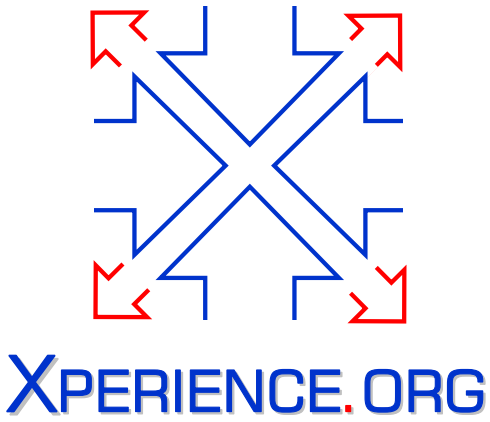
Project Acronym:	Xperience
Project Type:	IP
Project Title:	Robots Bootstrapped through Learning from Experience
Contract Number:	270273
Starting Date:	01-01-2011
Ending Date:	31-12-2015



Deliverable Number:	D3.1.3
Deliverable Title:	Transfer of structural bootstrapping on sensorimotor experience: Report or scientific publication on implementation of structural bootstrapping on sensorimotor experience within the architecture and in the final demonstration.
Type (Internal, Restricted, Public):	PU
Authors:	Sandor Szedmak, Emre Ugur, Hanchen Xiong, Justus Piater, Tamim Asfour, Norbert Krüger, Aleš Ude, Rok Vuga, and Alejandro Agostini, Florentin Wörgötter
Contributing Partners:	ALL

Contractual Date of Delivery to the EC:	31-01-2015
Actual Date of Delivery to the EC:	31-01-2015

Contents

Chapter 1

Executive Summary

This deliverable presents the implementation framework and the scientific contributions to realize the main objectives of Work Package 3.1.

The summary of that objective reads as "The general objective of WP3 is to investigate and implement structural bootstrapping of semantic sensorimotor categories at the continuous control level as well as at the planning level. Within WP3 WP3.1 investigates and implements structural bootstrapping of semantic sensorimotor categories. The goal is to learn cognitive categories faster than with existing methods, as well as to learn more elaborate cognitive categories than can feasibly be done with existing methods."

The deliverable consists of two main parts in two chapters.

Transfer report on the implementation of WP3.1 Chapter 2 contains the details of the learning infrastructure of the structural bootstrapping, the ROAR, which is the central part of the whole system. The ROAR is designed to behave as a general, intelligent database server which can work independently from the concrete robot applications. The most important advantage of this independence is the high flexibility and the easy transferability of the learning module to other areas of robot experiments and applications. The task of the ROAR is to synthesize a consistent collection of information out of noisy and inconsistent sources of the sensorimotor data while significantly reducing the cost of building a reliable knowledge base.

Scientific contribution Chapter 3 comprises the results and publications achieved in Year 4 of the Xperience project relating to WP3.1. Some of these pieces of work provide background knowledge to implement different learning modules for the ROAR system; others enhance the usefulness and the quality of sources inserted into the database.

Additionally the description of a demonstrative application of the ROAR system is included in this part of the deliverable; see Section 3.5.

Chapter 2

Transfer report on structural bootstrapping

2.1 General description of the ROAR

The infrastructure of learning of object-action relation and the replacement of the objects or actions with a suitable one is built around the ROAR module. That module behaves as a certain type of object memory where the set of available or potentially available objects together with the affordances related attributes are stored. The ROAR stands for *repository of objects&attributes with roles*. The database of prior knowledge can be created by hand or by prior experience. It allows objects to be retrieved by their attributes, and the attributes of novel objects can be inferred. General description of the ROARs main tasks can be found in Deliverable D3.1.2.

The ROAR module serves as an active database system, which not only stores and returns the data items, but via machine learning tools it extends the database with predicted elements. In this way it can provide data not observed earlier by the users connected to the database. This type of active database might also be called as "Intelligent Relational Database". The details about the implementation can be found in the technical report, [SP14], attached to this deliverable.

ROAR can learn from various data sources and can make reasoning in different ways. While ROAR has the potential of representing any type of relation, the current interface, which is designed based on the requirements of the demonstration scenario, supports learning from and reasoning on (object, action, score) tuples. Figure 2.1 shows that how this intelligent database can learn from different data sources and how the reasoning capability of ROAR can be exploited by different modules. First of all, the relations represented in ROAR can be automatically bootstrapped by common sense knowledge extracted from text [2], (action, object, score) tuples in particular as depicted in Module A in the figure. Alternatively, ROAR can learn object-action relations directly from the domain descriptions used by the planners that UGOE and KIT/UEDIN realized for the Xperience scenarios. Note that, ROAR not only stores the known action-object relations, but has the ability to infer the scores of the missing ones. In the current scenarios, reasoning capability of ROAR is used when the plan execution fails due to missing target objects, and the plan execution monitoring module searches for alternative objects that can replace the target object. The learning from domain description and reasoning data-flow in object-replacement scenario are depicted in Modules B and C in Figure 2.1. Modules A-C shows example scenarios where learning and inference are based on object and action labels. However ROAR can also represent continuous data, and have the potential to make inferences based on (for example) features obtained from perception as shown in Module D. In the current state, prototypes of the connections to Modules A and B have already been implemented, and connections to other modules, which are also under construction, are targets of the next year.

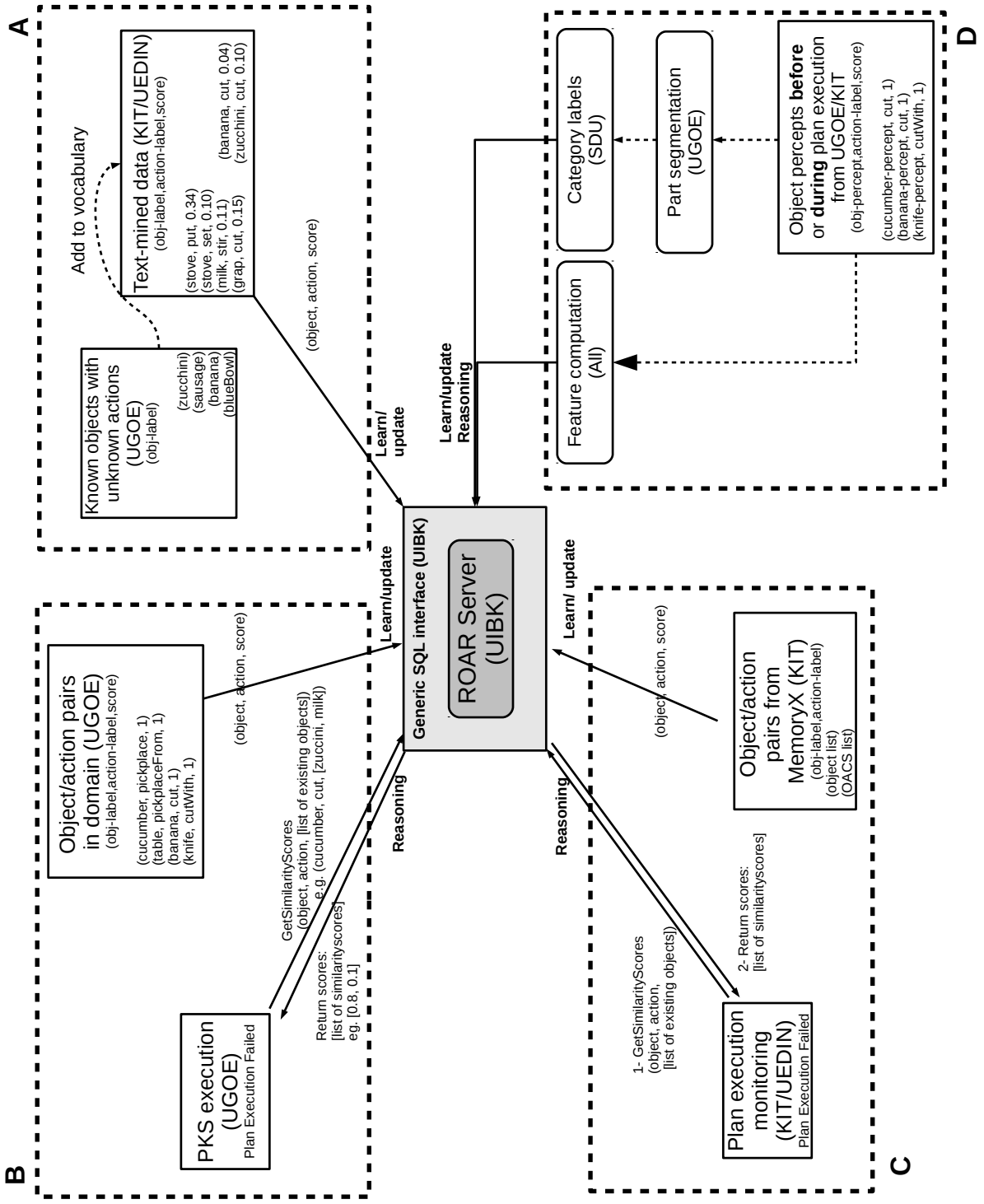


Figure 2.1: How the ROAR as an intelligent database is connected to the potential data sources and potential clients exploiting the accumulated knowledge about the object-action relations.

Chapter 3

Structural Bootstrapping on sensorimotor experience: scientific contribution in Year 4

3.1 Affordance learning

In the previous Deliverable D3.1.2, we discussed that learning object-action relations can benefit from our powerful knowledge propagation approach [7] that exploits kernelized maximum-margin methods outperforming standard discriminative learning methods in large, noisy and sparse datasets. In [SUP14], we extended and applied this method, namely Maximum Margin Multi Valued Regression (MMMVR), to a learning task of predicting the effects of the actions that are applied on pairs of objects. In the experiments, we evaluated this method with a dataset composed of 83 objects and 83x83 interactions. We compared the prediction performance with standard classifiers that predict the effect category given object pairs low-level features or single-object affordances. The experiments showed that proposed method achieves significantly higher prediction performance especially when supported with Active Learning.

The knowledge propagation approach, which we verified above, achieves bootstrapping in one level of learning. In the technical report [8] attached to the previous Deliverable D3.1.2, we verified that stacking the learners, i.e. forming hierarchical affordance prediction structures that use outputs of simple affordance predictors as inputs of complex affordance predictors, speeds-up learning of complex affordances compared to a non-stacked flat prediction structure where all learners use the same low-level visual attributes. We presented these results in [USP14b] and further showed that by actively selecting the next objects and by increasing the diversity of the training set using a distance measure based on learned simple-object affordances, the effect of bootstrapping can be further increased in [USP14a]. A truly developmental system, on the other hand, should be able to self-discover this manually designed hierarchical structure along with a suitable learning order. In [UP14], we showed that the hierarchical structure and the development order can emerge through use of two principles, namely Intrinsic Motivation and prediction based on the most distinctive attributes. We implemented our method in an online learning setup, and tested it in a real dataset that includes 83 objects and manually coded discrete effects (such as pushed, rolled, inserted) of three poke actions and one stack action.

While, the above work provides a natural setting for bootstrapping affordance learning in different levels, goal-oriented complex task execution requires use of the learned affordances, concepts and predictions as components of complex reasoning and planning. In [UP15], we studied the problem of symbol formation for planning. The robot discovers object categories, discrete effects, and the corresponding logical rules through its single-object and paired-object explorations, and directly use them as components of symbolic domain representation and symbolic planning. Development of the symbolic knowledge is achieved in two stages. In the first stage, the robot explores the environment by executing different actions on single objects, forms effect and object categories, and learns to predict object/effect categories from visual properties of the objects. In the next stage, with further interactions that involve stack actions on pairs of objects, the system learns logical high-level rules that return stack effect category given the categories of the involved objects and discrete relations between them. Finally, these categories and rules

are encoded in Planning Domain Definition Language (PDDL), enabling symbolic planning. We realized our method by learning the categories and rules in a physics-based simulator through exploration. The learned symbols and operators were verified by generating and executing non-trivial symbolic plans in the real robot in a tower building task.

3.2 Novel Learning for Multi-Label Prediction

Affordance prediction can be cast into a multi-label learning framework where each affordance is considered as a label and object attributes (or properties) are fed as inputs. For example, in our construction of ROAR, various attributes can be extracted from visual appearance/features of objects or other sources, while objects' affordances and functionalities are role labels. In [XSP14b], [XSP14d], [XSP14a], [XSP14c], we investigated the difficulties in multi-label learing from different perspective (*e.g.* inter-label dependency, efficiency, structural dependency regularization and heterogeneous information fusion), and developed two novel learning methods: Joint-SVM and KGHA (Kernel Generalized Homogeneity Analysis) which considerably improved the performance in predicting multiple labels.

3.2.1 Joint SVM

Structural SVM (support vector machine) is an extension of SVM for structured-outputs, in which, however, the margin to be maximized is defined as the score gap between the desired output and the runner-up. Assume that inputs $\mathbf{x} \in \mathcal{X}$, structured outputs $\mathbf{y} \in \mathcal{Y}$, and the score function is linear in some *combined feature representation* of inputs and outputs $\mathbf{\Psi}(\mathbf{x}, \mathbf{y})$: $F(\mathbf{x}, \mathbf{y}; \mathbf{W}) = \langle \mathbf{W}, \mathbf{\Psi}(\mathbf{x}, \mathbf{y}) \rangle$, then the objective function of structural SVM is:

$$\arg \min_{\mathbf{W} \in \mathbb{R}^{\mathbf{\Psi}}} \frac{1}{2}||\mathbf{W}||^2 + C \sum_{i=1}^{m} \max_{\mathbf{y}' \in \mathcal{Y}} \left\{ d(\mathbf{y}^{(i)}, \mathbf{y}') - \Delta_F(\mathbf{y}^{(i)}, \mathbf{y}') \right\} \tag{3.1}$$

where $\Delta_F(\mathbf{y}^{(i)}, \mathbf{y}') = F(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}; \mathbf{W}) - F(\mathbf{x}^{(i)}, \mathbf{y}'; \mathbf{W})$ and $d(\mathbf{y}^{(i)}, \mathbf{y}')$ is a distance function defined on structured outputs. In multi-label scenario, given a set of T labels, then outputs are T-dimensional binary vector $\mathbf{y} = [y_1, \cdots, y_t, \cdots, y_T]^\top \in \mathbb{B}^T$. When we define the score function $F\left(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}; \mathbf{W}\right) = \langle \mathbf{W}, \phi(\mathbf{x}^{(i)}) \otimes \mathbf{y}^{(i)} \rangle$, and use *Hamming distance* on outputs, then because of linear decomposability, (3.1) can be rewritten as:

$$\begin{array}{rl} \arg \min\limits_{\mathbf{W} \in \mathbb{R}^{\mathcal{H}_\phi \times \mathbb{R}^T}} & \frac{1}{2}||\mathbf{W}||_F^2 + C \sum_{i=1}^{m} \sum_{t=1}^{T} \max_{y'_t = \{-1,+1\}} \left\{ d(y_t^{(i)}, y'_t) - \Delta_F(y_t^{(i)}, y'_t) \right\} \\ \Downarrow & \\ \arg \min\limits_{\mathbf{w}_1, \cdots, \mathbf{w}_T \in \mathbb{R}^{\mathcal{H}_\phi}} & \sum_{t=1}^{T} \left\{ \frac{1}{2}||\mathbf{w}_t||^2 + C \sum_{i=1}^{m} \max \left\{ 0, d(y_t^{(i)}, -y_t^{(i)}) - \Delta_F(y_t^{(i)}, -y_t^{(i)}) \right\} \right\} \end{array} \tag{3.2}$$

where $\langle \cdot, \cdot \rangle_F$ denotes Frobenius product and $||\mathbf{W}||_F$ is the Frobenius norm of matrix $\mathbf{W}$.

It can be seen (in (3.2)) that, with linearly decomposable score functions and output distances, using structural SVM on multi-label learning is equivalent to learning T SVMs jointly. This is closely related to multi-task learning frameworks, where different learning tasks are connected by summing up their objectives and constraints respectively:

$$\begin{array}{rl} \min & \frac{1}{2} \sum_{t=1}^{T} ||\mathbf{w}_t||^2 + C \sum_{t=1}^{T} \sum_{i=1}^{m} \xi_t^{(i)} \\ \text{w.r.t.} & \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_T \in \mathbf{R}^{\mathcal{H}_\phi \times 1} \\ \text{s.t.} & \sum_{t=1}^{T} y_t^{(i)} \left(\mathbf{w}_t^\top \phi(x^{(i)}) \right) \geq T - \sum_{t=1}^{T} \xi_t^{(i)} \end{array} \tag{3.3}$$

By denoting $\mathbf{y}^{(i)} = [y_1^{(1)}, \ldots, y_T^{(i)}]$, and $\mathbf{W} = [\frac{\mathbf{w}_1^\top}{T}; \ldots; \frac{\mathbf{w}_T^\top}{T}]^\top$, we can rewrite (3.3) as:

$$\begin{array}{rl} \arg \min\limits_{\mathbf{W} \in \mathbb{R}^T \times \mathcal{H}_\phi} & \frac{1}{2}||\mathbf{W}||_F^2 + C \sum_{i=1}^{m} \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)}) \right\rangle \geq 1 - \bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1, \ldots, m\} \end{array} \tag{3.4}$$

which is referred to as *joint SVM*. When linear output kernels ($K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) = \langle \psi(\mathbf{y}^{(i)}), \psi(\mathbf{y}^{(j)}) \rangle$) are applied on outputs, (3.4) will be:

$$\begin{array}{rl} \arg \min\limits_{\mathbf{W} \in \mathbb{R}^{H_\psi \times \mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{W}||_F^2 + C \sum_{i=1}^{m} \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \psi(\mathbf{y}^{(i)}), \mathbf{W}\phi(x^{(i)}) \right\rangle \geq 1 - \bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1, \ldots, m\} \end{array} \tag{3.5}$$

Since the linear decomposability of $\Delta_F(\mathbf{y}^{(i)}, \mathbf{y}')$ is still preserved, join SVM solves the same problem as structural SVM. However, one strength of joint SVM is that its training complexity is almost the same as a single SVM, by contrast to the exponential complexity in structural SVM. Similarly to regular SVM, joint SVM can be converted to its dual form

$$\begin{array}{rl} \arg \min\limits_{\alpha_1,\cdots,\alpha_m} & \sum_{i=1}^m \alpha_i - \sum_{i,j=1}^m \alpha_i \alpha_j K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) K_\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \\ \text{s.t} & \forall i, 0 \leq \alpha_i \leq C \end{array} \tag{3.6}$$

with $\mathbf{W} = \sum_i^m \alpha_i \psi(\mathbf{y}^{(i)}) \phi(\mathbf{x}^{(i)})^\top$. It can be seen that, with the kernel matrix on outputs pre-computed, the computational complexity of joint SVM (3.6) is the same as the learning of one single SVM, which is a great advantage in efficiency. Meanwhile, when more general output kernels are used, then the linear decomposability of $\Delta_F(\mathbf{y}^{(i)}, \mathbf{y}')$ will be violated, then joint SVM becomes a special case of max-margin regression, which seeks to learn linear operators $\mathbf{W} : \mathcal{H}_\phi \to \mathcal{H}_\psi$ from general $\phi(\mathbf{x}) \otimes \psi(\mathbf{y})$.

Assume that the statistics of tags' pairwise co-occurrence can be encoded in a $T \times T$ matrix $\mathbf{P}$, via which the output vectors can be linearly mapped as $\psi(\mathbf{y}) = \mathbf{P}\mathbf{y}$, and thus the corresponding linear output kernel is:

$$K_\psi^{Lin}(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) = \mathbf{y}^{(i)\top} \mathbf{\Omega} \mathbf{y}^{(j)} \tag{3.7}$$

where $\mathbf{\Omega} = \mathbf{P}^\top \mathbf{P} = \mathbf{P}\mathbf{P}^\top$. By denoting $\mathbf{U} = \mathbf{P}^\top \mathbf{W}$, we can rewrite joint SVM (3.5) as:

$$\begin{array}{rl} \arg \min\limits_{\mathbf{W} \in \mathbb{R}^{H_\psi \times \mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{W}||_F^2 + C \sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \mathbf{y}^{(i)}, \mathbf{U}\phi(x^{(i)}) \right\rangle \geq 1 - \bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1, \ldots, m\} \end{array} \tag{3.8}$$

Meanwhile, we need to control the scale of $\mathbf{P}$, otherwise the constraints in (3.8) will be pointless. Different regularizations on $\mathbf{P}$ have been proposed in previous work. Here, we would like to add a regularization to control overfitting from output dependency-structures. By merging regularization on $\mathbf{W}$ and $\mathbf{P}$, we obtain a more compact regularizer, $\frac{1}{2}\mathbf{W}^\top \mathbf{\Omega} \mathbf{W}$, resulting in:

$$\begin{array}{rl} \arg \min\limits_{\mathbf{U} \in \mathbb{R}^{H_\psi \times \mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{U}||_F^2 + C \sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \mathbf{y}^{(i)}, \mathbf{U}\phi(x^{(i)}) \right\rangle \geq 1 - \bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1, \ldots, m\} \end{array} \tag{3.9}$$

Remarkably, (3.9) is equivalent to (3.5) with $\mathbf{W}$ substituted by $\mathbf{U}$, which suggests that a linear output kernel is implicitly learned, and absorbed in $\mathbf{W}$, when we training a plain joint SVM with no explicit kernel on outputs. In addition, a regularization on the output kernel is also implicitly added.

More details on other derivations of Joint-SVM and experiment results are referred to [XSP14b], [XSP14d], [XSP14a].

3.2.2 Kernel Generalized Homogeneity Analysis

Canonical correlation analysis (CCA) and homogeneity analysis (HA) are two popular methods for analyzing multivariate data. Although they are applied to different data types (CCA is used on two sets of continuous variables while HA operates on multivariate categorical variables), we reveal that they are actually closely related. Building on this relation, we generalize homogeneity analysis to continuous variables, which leads to a relaxed variant of multiple-set-CCA. Furthermore, kernel functions are also utilized to enable generalized HA to learn nonlinear dependencies within data.

In [XSP14c], we study KGHA in the multi-label learning case, although it can also be applied to more general multiple output regression. We show that when used for learning vector-valued functions (*e.g.* in multi-label, multi-class predictions), KGHA works as low-rank output kernel learning with manifold regularization on heterogeneous information fusion (HIF). Low-rank output kernel learning, to a certain degree, coincides with multi-label dimensionality reduction, which enables learners to gain higher efficiency and accuracy. Also, manifold regularization on HIF is related to multiple kernel learning (MKL), in which heterogeneous information is encoded in an ensemble of kernels to match outputs. Instead of regularizing on smoothness of instance-based manifold, the regularization of KGHA is based on the smoothness of prediction space from heterogeneous information.

3.2.3 Conditional Boltzmann Machines

It is also possible to conduct multi-label learning with conditional random fields with each binary node denoting the presence/absence of a label. To take inter-label dependency into account, the random fields can be constructed as a Boltzmann machine, which thus results in a conditional Boltzmann machine (CBM) (see Figure 3.1),

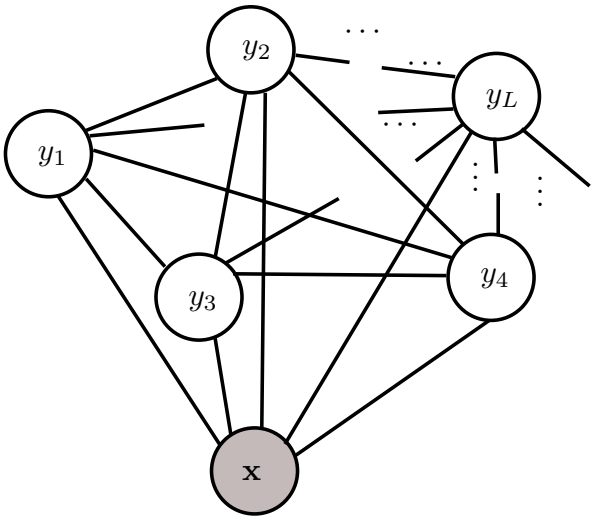


Figure 3.1: Conditional Boltzmann machine for multi-label learning.

Similar to other undirected graphical models, extract learning with maximum likelihood is intractable even for medium-scale model. For our purpose of multi-label learning, we plan to train CBMs with our novel approximation learning method, persistent sequential Monte Carlo [9], which is detailed in Deliverable D2.3.3.

3.3 Structural Bootstrapping for Action Learning

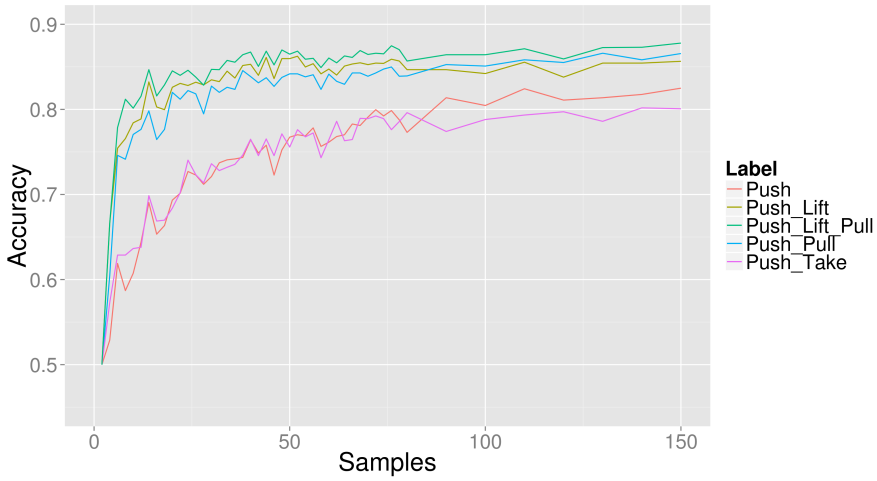


Figure 3.2: Learning to push with and without ***bootstrapping***

Learning the effect of actions in complex cluttered environments using vision input is a difficult task. Especially the effect of multi-object manipulating actions requires many training samples to learn reliable outcome predictors.

In [FKKG14] we learn Random Forest based success predictors from high level visual input histograms

(see [4, 1]) and demonstrate *bootstrapping* by reusing already learnt knowledge from previously learnt actions in two different ways.

We demonstrate both, direct and abstract knowledge transfer for *bootstrapping*. In direct *bootstrapping*, predictors are trained with the outputs of previously learnt predictors as additional inputs beside the state space. For abstract knowledge transfer we learn abstract concepts that capture underlying commonalities of two or more existing predictors and use the output of these concepts as additional data inputs for new predictors.

Figure 3.2 illustrates the prediction accuracy for a pushing action involving 2 objects. Related to this pushing action are the actions Pull and Lift and the Lift_Pull concept which is created from the Lift and Pull predictor knowledge. The Take action is unrelated. It can be seen in the Figure 3.2 that unrelated knowledge does not help to bootstrap learning. Related knowledge, however, leads to remarkable *bootstrapping* effects with the abstract knowledge concepts *bootstrapping* surpassing direct *bootstrapping*.

3.4 Action Learning through Directed Exploration

In our previous work we showed that exploration as realised by standard reinforcement learning techniques [3] can be integrated with statistical learning techniques [6, 5]. Based on a database of example movements, which solve the given task in specific situations, statistical learning can generate new movements for similar but not the same situations as in the example database. For example, the knowledge of reaching movements to a number of different locations can facilitate the generation of reaching movements to nearby locations. Based on this insight we defined a new, structured reinforcement learning algorithm, which can find new example movement patterns in the neighborhood of the example trajectory manifold much quicker than standard reinforcement learning algorithms. The key to this accelerated convergence of the autonomous learning process (see Figure 3.3) lies in the fact that thanks to the statistical generalization of existing movements, a significant part of the search process could be moved from the high-dimensional trajectory space spanned by DMPs to the low-dimensional space determined by query points, which are defined as task-relevant parameters that characterize the task. Another important feature of our approach is that statistical generalization provides a reference movement, which can be used to define intermediate rewards in terms of the distance between trajectories explored by reinforcement learning and the reference trajectory. This is needed to ensure that the newly found trajectories are similar to the trajectories in the example database. In this way the robot can autonomously augment the database of example movements.

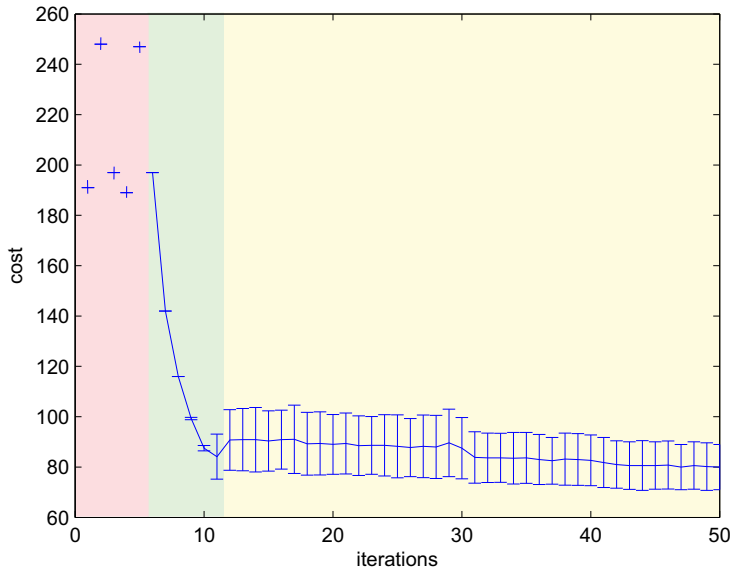


Figure 3.3: Convergence of the learning process. The bars show standard deviation of 8 learning trials. The red shaded part of the graph shows costs of trials using policies from prior knowledge. The green shaded area corresponds to learning where exploration was performed using ILC, while the yellow shaded area corresponds to final, random based exploration.

However, learning in reduced dimensionality can not guarantee that an optimal policy will be obtained in general because the optimal solution might lie outside of the reduced dimensionality space. Therefore, in Year 4 we explored how to accelerate the exploration of space outside the existing trajectory manifold. For this purpose we developed a new formulation of dynamic movement primitives (DMPs), which enables nonuniform time scaling of trajectories (see the attached paper [VNU14]). By normalizing the DMPs with respect to the arc length of the underlying trajectory, we were able to compare movement patterns across different tasks and employ example movements from non-related tasks when searching for the optimal movement for a new given task. In this way we were able to significantly accelerate the learning process even if no prior movements specific to the new task were available.

In our experiments we focused on trajectory speed adaptation. We showed experimentally that the proposed approach is effective at learning high precision tasks such as fast transfer of a cup full of liquid without spilling.

3.5 Integrating Symbolic Planning and ROAR for the Final Demonstration in Scenario 1

This section introduces the theoretical outlines of the integration of symbolic planning and ROAR for the final demonstration in Scenario 1 (salad scenario, module B in Figure 2.1). The complete demonstration will be presented in deliverable D5.2.4.

The salad scenario contains several actions: pick and place, cut, chop, drop, pour, and stir, with different objects: knife, cutting board, bowl, cleaver, cucumber, carrot, etc. The main idea is that the robot prepare a salad executing a symbolic plan. For the plan execution, the robot must make sure that all the required objects are in the scene. If any of these objects is missing, the robot searches for its replacement using ROAR, update the plan accordingly, and execute, in a structural bootstrapping process.

Figure 3.4 shows the general diagram of the plan generation and object replacement process. First, a plan is generated from the original list of objects of the recipe. Once the plan is generated, the robot checks if all the required objects are in the scenario. If there are missing objects, the robot connects to ROAR to find out which of the existing objects have the same affordances of the missing ones. If replacements are found, the plan is updated and executed. This process could be compared to when we are at the office thinking of what we will have for dinner when we arrive home. We first recall the food we have in the kitchen and then plan the dinner. Once at home, we may find out that some of the elements are actually missing. In this case, we search for replacements of these elements and update our dinner plan.

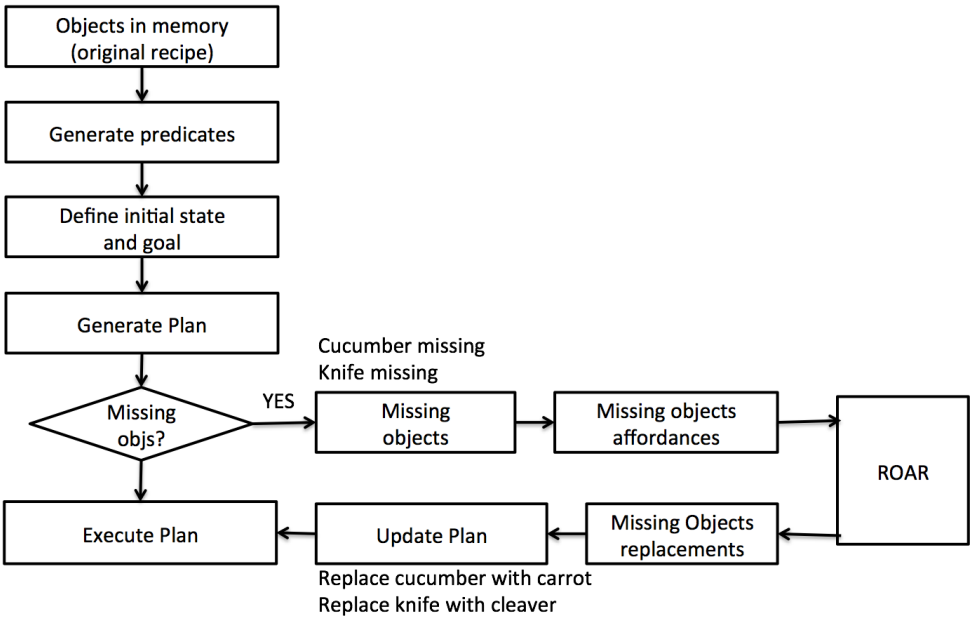


Figure 3.4: General diagram for plan generation and object replacement.

For the successful updating of the plan, it is mandatory that the replacements have the same affordances as the missing objects. This demands that the predicates, coding the object affordances in the planning domain definition, and the notation used in ROAR, are fully compatible. Table 3.1 presents an example list of objects and their affordances in ROAR notation and the corresponding predicates defined for planning. ROAR notation involves the object name, the related action, the preposition, indicating the specific function of the object in that action, and the score. We can see in the table that a cucumber and a banana are two objects that can be cut, dropped, or picked and placed, facts represented in planning notation with the predicates *cutObj*, *dropObj*, *PPObj*, respectively.

Table 3.1: Object Affordances Notation Example.

Object	Action	Preposition	Predicate	Score
cucumber	cut	null	cutObj	1
banana	cut	null	cutObj	1
knife	cut	with	cutWith	1
cleaver	cut	with	cutWith	1
cucumber	drop	null	dropObj	1
banana	drop	null	dropObj	1
board	drop	from	dropFrom	1
cucumber	pick place	null	PPObj	1
banana	pick place	null	PPObj	1
board	pick place	from	PPFrom	1

To provide a more concrete idea of the role of ROAR in the planning domain definition, Figure 3.5 presents a detailed description of the elements involved in this process. Note that the diagram represents the same four modules of the left-hand side of Figure 3.4. We can see that the initial specification of the

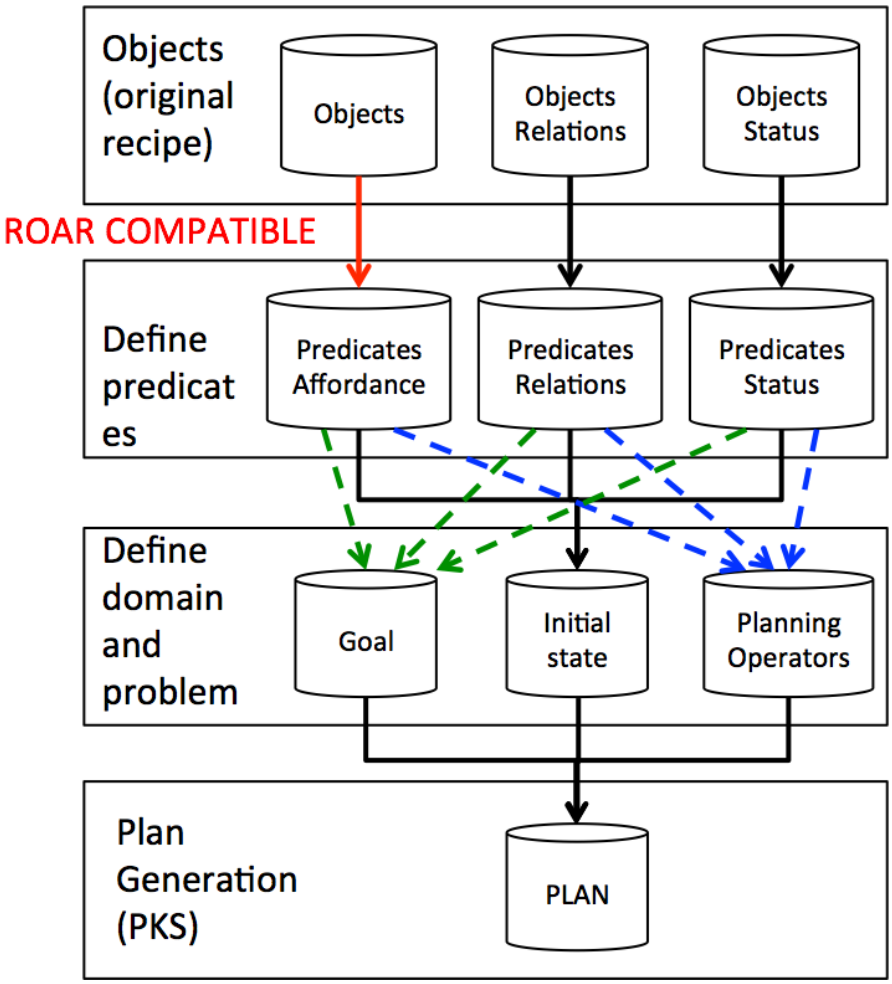


Figure 3.5: General diagram the planning domain and problem definitions and plan generation.

objects for the recipe not only consider the list of objects, but also the relation between them and their status. From the list of objects in memory (original recipe) the system generates the predicates coding the affordances (e.g. cucumber is cuttable: $cutObj(cucumber)$), relations (e.g. cucumber on the board: $on(cucumber, board)$), and status (e.g. cucumber not cut ($!cut(cucumber)$)). These predicates are used, in turn, to describe the initial state and goal, as well as to code the planning operators. Finally, a plan is generated, using the PKS planner. Example planning operators in PKS notation are shown in Figure 3.6 while Figure 3.7 presents an example plan generated by the PKS planner for the preparation of a salad using a cucumber.

```
action pick_place( ?obj, ?with, ?from, ?to){
   preconds:
          K(PPobj(?obj)) &
          K(on(?obj, ?from)) &
          K(!on(?obj, ?to)) &
          K(PPwith(?with)) &
          K(free(?with)) &
          K(PPto(?to))
   effects:
          add(Kf, !on(?obj,?from)),
          add(Kf, on(?obj,?to))   }

action cut( ?obj, ?with, ?on){
   preconds:
          K(cutObj(?obj)) &
          K(on(?obj, ?on)) &
          K(!cut(?obj)) &
          K(cutWith(?with)) &
          K(cutOn(?on))
   effects:
          add(Kf, cut(?obj))   }
```

Predicates Affordances:
ROAR COMPATIBLE

```
action drop( ?obj, ?with, ?from, ?in){
   preconds:
          K(dropObj(?obj)) &
          K(on(?obj, ?from)) &
          K(!in(?obj, ?in)) &
          K(dropWith(?with)) &
          K(free(?with)) &
          K(dropInto(?in))
   effects:
          add(Kf, !on(?obj, ?from)),
          add(Kf, in(?obj, ?in)) }

action stir( ?obj, ?with){
   preconds:
          K(stirObj(?obj)) &
          K(!stirred(?obj)) &
          K(stirWith(?with))
   effects:
          add(Kf, stirred(?obj)) }
```

Figure 3.6: Example planning operators in PKS notation for the salad scenario. The predicates coding affordances compatible with ROAR are marked in red.

pick_place(cucumber,hand,table,board)
cut(cucumber,knife,board)
pour(sand,hand,jar,bowl)
drop(cucumber,hand,board,bowl)
stir(bowl,spoon)

Figure 3.7: Example plan generated by the PKS planner for the preparation of a cucumber salad. In the demonstration the sand replaces the salad dressing.

Finally, to shed more light on the object replacement process, Figure 3.8 shows an example of the process taking place when the object to be replaced is a cucumber. In this case, the cucumber is missing from the scenario but a banana is present (among other objects). The predicates coding affordances of the cucumber involved in the plan generation are: $PPobj(cucumber)$ (pick and place-*able*), $cutObj(cucumber)$ (cuttable), $dropObj(cucumber)$ (droppable). For each of the objects in the scene, the robot evaluates, via ROAR, which of them has all these three affordances. The first object found fulfilling these affordances is a banana. Then, the plan is updated replacing the cucumber with a banana, as shown in the figure.

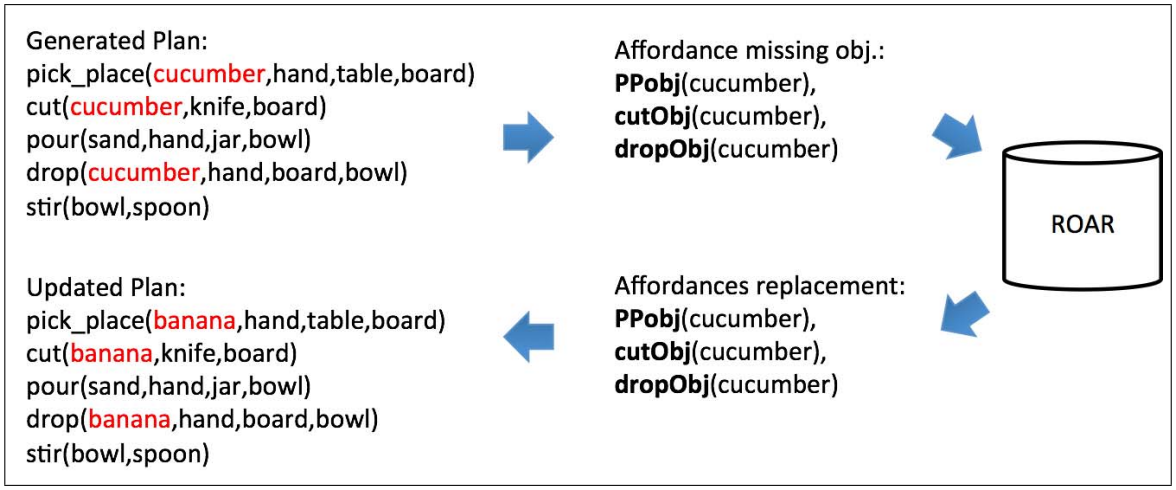


Figure 3.8: Example of object replacement and plan updating.

Chapter 4

Conclusion

In the fourth year of the project the members of the consortium focused in WP3.1 on designing and building a general, broadly applicable learning infrastructure. The high level autonomy of the ROAR system can open up new application areas where the idea of structural bootstrapping can be exploited in the future.

References

[1] Severin Fichtl, Andrew McManus, Wail Mustafa, Dirk Kraft, Norbert Krüger, and Frank Guerin. Learning spatial relationships from 3D vision using histograms. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 501–508, Hong Kong, May 2014. IEEE.

[2] Peter Kaiser, Mike Lewis, Ronald P. A. Petrick, Tamim Asfour, and Mark Steedman. Extracting common sense knowledge from text for robot planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2014)*, Hong Kong, China, 31 May– 7 June 2014.

[3] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[4] Wail Mustafa, Nicolas Pugeault, and N Krüger. Multi-View Object Recognition using View-Point Invariant Shape Relations and Appearance Information. In *IEEE International Conference on Robotics and Automation*, 2013.

[5] Bojan Nemec, Denis Forte, Rok Vuga, Minija Tamosiunaite, Florentin Wörgötter, and Aleš Ude. Applying statistical generalization to determine search direction for reinforcement learning of movement primitives. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 65–70, Osaka, Japan, 2012.

[6] Bojan Nemec, Rok Vuga, and Aleš Ude. Exploiting previous experience to constrain robot sensorimotor learning. In *2011 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 727–732, Bled, Slovenia, 2011.

[7] Sandor Szedmak and Justus Piater. Learning object-action relations via knowledge propagation. Technical report, University of Innsbruck, 2012.

[8] Emre Ugur. Bootstrapping multi-object affordance learning using learned single-affordance features. Technical report, University of Innsbruck, 2014.

[9] Hanchen Xiong, Sandor Szedmak, and Justus Piater. Towards maximum likelihood: Learning undirected graphical models using persistent sequential Monte Carlo. In *6th Asian Conference on Machine Learning (ACML14)*, 2014. (To appear.).

Attached Articles

[FKKG14] Severin Fichtl, Dirk Kraft, Norbert Krüger, and Frank Guerin. Bootstrapping: Knowledge Transfer from Learned Action Preconditions to new Actionsaffordance learning with learned single-affordance features. 2014. To be submitted.

[SP14] Sandor Szedmak and Justus Piater. Roar server interface. Technical report, University of Innsbruck, 2014.

[SUP14] Sandor Szedmak, Emre Ugur, and Justus Piater. Knowledge Propagation and Relation Learning for Predicting Action Effects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 623–629. IEEE, 09 2014.

[UP14] Emre Ugur and Justus Piater. Emergent Structuring of Interdependent Affordance Learning Tasks. In *The Fourth Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics*, 10 2014. To appear.

[UP15] Emre Ugur and Justus Piater. Bottom-Up Learning of Object Categories, Action Effects and Logical Rules: From Continuous Manipulative Exploration to Symbolic Planning. Technical report, University of Innsbruck, 2015.

[USP14a] Emre Ugur, Sandor Szedmak, and Justus Piater. Bootstrapping paired-object affordance learning with learned single-affordance features. In *The Fourth Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics*, 10 2014. To appear.

[USP14b] Emre Ugur, Sandor Szedmak, and Justus Piater. Complex affordance learning based on basic affordances. In *22nd Signal Processing and Communications Applications Conference*, pages 698–701. IEEE, 04 2014.

[VNU14] Rok Vuga, Bojan Nemec, and Aleš Ude. Speed profile optimization through directed explorative learning. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 547–553, Madrid, Spain, 2014.

[XSP14a] Hanchen Xiong, Sandor Szedmak, and Justus Piater. Implicit Learning of Simpler Output Kernels for Multi-Label Prediction. In *NIPS workshop on Representation and Learning for Complex Outputs*, 2014. To appear.

[XSP14b] Hanchen Xiong, Sandor Szedmak, and Justus Piater. Joint SVM for Accurate and Fast Image Tagging. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 4 2014.

[XSP14c] Hanchen Xiong, Sandor Szedmak, and Justus Piater. Multi-Label Learning with Kernel Generalized Homogeneity Analysis. Technical report, 2014.

[XSP14d] Hanchen Xiong, Sandor Szedmak, and Justus Piater. Scalable, Accurate Image Annotation with Joint SVMs and Output Kernels. *Neurocomputing*, 2014. To appear.

Bootstrapping: Knowledge Transfer from Learned Action Preconditions to new Actions

Severin Fichtl, Dirk Kraft, Norbert Krüger and Frank Guerin

CONTENTS

***Abstract*—Learning the effect of actions in complex cluttered environments using vision input is a difficult task. Especially the effect of multi-object manipulating actions requires many training samples to learn reliable outcome predictors.**

We learn Random Forest based success predictors from high level visual input histograms (see [1], [2]) and demonstrate *bootstrapping* by reusing already learnt knowledge from previously learnt actions in two different ways.

We demonstrate both, direct and abstract knowledge transfer for *bootstrapping*. In direct *bootstrapping*, predictors are trained with the outputs of previously learnt predictors as additional inputs beside the state space. For abstract knowledge transfer we learn abstract concepts that capture underlying commonalities of two or more existing predictors and use the output of these concepts as additional data inputs for new predictors.

Figure 1 illustrates the prediction accuracy for a pushing action involving 2 objects. Related to this pushing action are the actions Pull and Lift and the Lift_Pull concept which is created from the Lift and Pull predictor knowledge. The Take action is unrelated. It can be seen in the Figure 1 that unrelated knowledge does not help to bootstrap learning. Related knowledge, however, leads to remarkable *bootstrapping* effects with the abstract knowledge concepts *bootstrapping* surpassing direct *bootstrapping*.

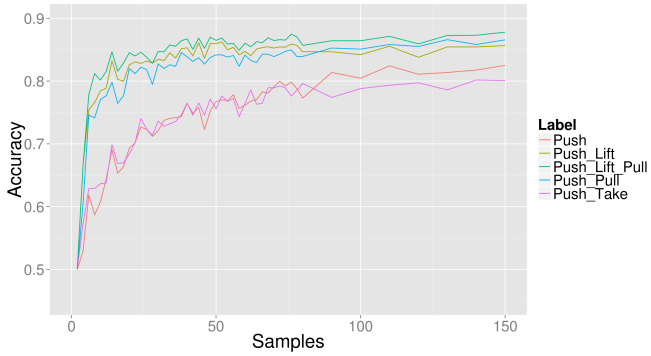


Fig. 1: Learning to push with and without ***bootstrapping***

I. INTRODUCTION

We are interested in grounding knowledge in a robot's own sensorimotor experience, this approach is an accepted principle of developmental robotics [3]. A major problem with this approach is that it takes a long time to learn through robot experience. The state of the art in artificial development and learning methods does not permit a robot to learn from experience as rapidly as an infant. This mirrors problems in other areas of artificial intelligence such as speech recognition where systems need orders of magnitude more data to learn from than a small child is exposed to [4]. The added difficulty in robotics is that we do not have a large data set to run the learning algorithm on, a robot must first go through the slow process of trying actions out in the world. This problem of slow learning has generated interest in methods that can bootstrap the learning [5], [6][1]. The basic idea is that if we have some grounded knowledge we should be able to learn more or similar things faster. In this paper we focus on one type of knowledge that is important to a robot that must manipulate various objects, that is the knowledge of spatial relationships that determine the outcome of actions on object pairs. In summary our problem has three aspects:

[1]Note that the use of 'bootstrap' here is not the standard use in Machine Learning, but more akin to the layman's use.

1) Learning spatial relationships that determine the outcome of actions.
2) Learning by grounding in the robot's own action.
3) Learning rapidly.

In previous work we have developed a histogram feature which is good at capturing knowledge of spatial relationships for a variety of objects [2]. This paper uses this feature and focuses on speeding up the learning of preconditions that determine the outcome of actions, such as pushing an object which is under another, or lifting an object which contains another.

We now discuss the importance of the three aspects of our problem. Firstly spatial relationships are extremely important for robotic manipulation e.g. in service robots, to determine the outcome of actions in everyday home environments; for example, trying to reach something that is partly obstructed by another object, or pulling or lifting something when another object is on top or inside of it. These are the kinds of situations that children are competent with, and in looking at how infants first acquire this knowledge we see the close connection with means-ends behaviours involving more than one object (i.e. when one action is used as a means to achieve some other end goal). Infants start to appreciate importance of spatial relationships when they begin exploring means-end actions at around 8 months [7], [8], and means-end behaviour is the start of planning: how to use one action as a step to achieve some more distant goal. Hence learning spatial relationships that determine the outcome of actions are a crucial part of the problem of robots learning planning operators for manipulation.

Secondly, grounding in a robot's own actions helps to avoid problems with classical AI which relied on a human's judgement of what knowledge and representation might be appropriate for the robot [9]. Handcoded knowledge tended to result in brittle systems (i.e. they broke down when the task went outside the scenarios that the human had foreseen). Knowledge learnt from action can be expected to be more useful to the robot, and more robust, in line with the "Verification Principle" [3], [10]:

> An AI system can create and maintain knowledge only to the extent that it can verify that knowledge itself.

Thirdly rapid learning is important because there is a great deal to be learnt in order to achieve a basic level of manipulation competence in everyday environments, such as a child has. Spatial relationships are only one small segment of the commonsense knowledge that is required. Also it typically takes a large amount of data to ground knowledge in the robot's own actions, and this data is usually hard to generate. If any technique can reduce the requirement for data it would be enormously beneficial.

Previous work has tackled learning spatial relationships from vision [2], [11], and grounding knowledge [12], [13], [14], our focus in this paper is the rapid bootstrapping of the learning.

In artificial Intelligence in general various methods have been tried to accelerate learning. For example intrinsic motivation or active learning [15], [16] which can avoid wasting time on training examples which are not useful. This seems to be one of the techniques employed by humans in development, but it alone does not account for the rapidity of learning; what is needed in addition is some way to reuse past similar knowledge when it is relevant to the current learning task. This is also called transfer learning [17]. There are good examples of transfer learning for motor policies for example [18], [19], [20], but we do not find many examples of transfer learning in the learning of preconditions determining the success of actions (recent work by Ugur et al. [6] being an exception). This is a relatively new area of investigation.

Our key technique for bootstrapping is inspired by works in cognitive science where symbolic knowledge is learnt from interaction and can then be reused where it benefits subsequent learning, for example the 'synthetic item' of Drescher and Chaput [21], [22]. These techniques have not been applied to high dimensional robot manipulation scenarios to the best of our knowledge (Ugur et al. [6] being an exception). In our work we learn categories from early experience, where these categories correspond with spatial relationships like 'on top' or 'inside'. Note that these categories are not imposed by the human designer, but rather are learnt by a classifier whose task is to discriminate situations, from visual input, where a certain action will have one outcome or another. These categories can be then used as a binary input to subsequent learning, resulting in a speedup where they have discriminative power. The visual input used for learning is RGB-D data which is compiled into histograms (as in [2]). We learn from simulated robot actions as this was the only feasible way to generate the thousands of examples we needed to compare different approaches. Our results show that learning from these categories gives a greater speedup in learning than other approaches, e.g. to achieve 90% accuracy take 10% of the number of training samples (see Sec. V-B).

This work is related to and partly builds on previous work [12], [23], [2] which did learn classifiers predicting the outcomes of actions. The main new contribution here is the bootstrapping of the learning, evaluating various new techniques (Sect. V); in addition Sect. IV-B1 provides new more extensive results than presented in Fichtl et al. [2].

II. Related Work

Our work learns classifiers which discriminate between possible effects of an action on a pair of objects in some relationship. This is quite close to work on learning relational "affordances" (i.e. not just the affordance of a single object, but a pair). Ugur et al. [6] learn "paired object affordances" for the action of stacking. In this work the stack action was attempted with 18 pairs of random objects. The input to the learner was a set of shape features for each object (consisting of histograms of normal vectors for various points on the object's surface). The effects observed were "tumbled over", "piled up" (i.e. successfully stacked), "covered" (when the top object is a cup that covers and completely contains the lower object),

and "inserted in" (when the lower object is the container and the top object drops into it). Classifiers were learnt to predict which effect would occur given the visual features of the pair of objects. There are some similarities and differences with our work. There is a significant difference in the training data in that we are looking at objects already in a relationship, in order to determine the effect of an action, whereas Ugur et al. are looking at the features of the two objects before they are put in a relationship, in order to determine what relationship they might end up in after an action. Because of this it is clear that it is important for us to look at relative position, whereas that would not make sense in Ugur et al.'s case. Ugur et al. give as input the combined set of shape features of the two objects, hence their classifier has the opportunity to learn relationships among the objects' features. This is similar to our work, except that we go even further, and to some extent "rig the game" in favour of paying attention to relationships, because our histogram feature is compiled from relationships among parts of the objects.

Our histogram approach is inspired by the approach of Mustafa et al. [1]. That work compiles histograms over relationships between surface patches (distances and angles) in a single object. These histograms characterise the object, and are quite robust to variations in viewpoint. Mustafa et al. use this for object recognition. In our work we borrow the idea of compiling histograms over relationships among surface patches; however we look at pairs of objects, and compile histograms which relate every patch on the first object with every patch on the second. Our idea is that these histograms should characterise the relationship between the objects. We are not aware of any other work which uses a feature computed from relationships among parts of two different objects.

Although there is rather a lot of work on affordances, it has been noted by Moldovan et al. [24] that there is very little work on relational affordances (i.e. the actions afforded by a pair of objects in a particular spatial relationship). In their work the relational features considered are relative distance between two objects, the relative orientation of one with respect to the other, and whether or not they are touching. We go for a much richer relational description looking at much finer grained details of the objects, which can capture such things as one object having elements enclosing another, or in front of another etc. Rosman and Ramamoorthy [11] learn spatial relationships between objects using a support vector machine based approach. In this approach the support vectors are picked from for their ability to differentiate the point cloud into two objects. This has the effect that the subset of points considered by the classifier are on the edges of the object. Relations are then learnt based upon the relative positions of clusters of the support vectors (the scene is reduced to clusters with xyz coordinates). Compared to both of the above cited works believe that our histogram based approach captures a higher proportion of the important information about the relations between objects in the scene, whereas much of this information is discarded by the above approaches.

One particularly interesting work on support relations is by Panda et al. [25]. This is considerable more elaborate and advanced than our approach in a number of respects. With regard to manipulating objects in clutter the authors state that "The interaction with surrounding objects in the environment must be considered in order to perform the task without causing the objects fall or get damaged." The work exploits a number of visually derived features regarding the relationship between the objects: proximity, boundary overlap, depth boundary, containment, relative stability. In addition a rule based method is employed to infer what supports what, when multiple objects are stacked or leaning on each other. This method implicitly embodies significant commonsense knowledge about how things can be supported in a hierarchy. The system allows for more sophisticated reasoning about support relations than is possible with our method. However, we have approached the problem more from a developmental robotics perspective; we are attempting to see what the system can learn without significant prior knowledge, and learning from the effects of its actions. Our approach to reach the level of sophistication of Panda et al.'s work would be to attempt to get the system itself to learn about the rules governing hierarchical support orders in gradual incremental steps. The reasons for our preferring the developmental approach have been discussed elsewhere [12]. Briefly: such an approach could be more robust because the system's knowledge (e.g. naive physics) is grounded in its own experience, and can be revisited at any time if new data calls it into question and requires adjustment for cases not encountered before [10]. In contrast, where knowledge is input by a human designer the system will be brittle, and will only be reliable in situations that the human designer has foreseen, and it is unlikely that all eventualities can be foreseen for robots working in everyday environments.

A second aspect of our work which deserves comparison with others is the bootstrapping. Ugur et al. [6] bootstrap the learning of a "stacking" affordance by first learning a "rolling" affordance (items which can roll are usually not good candidates to stack something on top of). This is in the same spirit as our approach; where Ugur has rollable as an input to the second stage of learning, we have several categories. We have attempted to learn a variety of categories because the robot does not know in advance which (if any) might be useful in later stages of learning.

We do not feel that our work is particularly close to computer vision work in scene understanding (e.g. [26]) because those works typically recognise all objects, and then can use higher level knowledge to assist in understanding. Our work in contrast is at a lower level, and is more concerned with the physical relationships among surfaces without regard for object knowledge. We think of it more like how an infant might recognise simple physical relationships between household objects without any idea of what their names are or what their typical purposes are.

We can also relate our work to infant development. In the period from six months of age through to two years human infants undergo significant development in their skills and understanding relating to physical world objects and their manipulation. Observations of infants show that, at any particular age, they possess a repertoire of behaviours or manual skills which they apply to various objects or surfaces

they encounter [7], [27]. Each such behaviour could be seen as roughly analogous to a planning operator in Artificial Intelligence, because there are situations which make them likely to be executed (like the precondition of a planning operator), and expected effects (postcondition), as well as some motor control program describing the behaviour executed. As infants develop they solve the problems of (i) identifying when a new behaviour should be created, (ii) learning the new precondition, (iii) postcondition, and (iv) motor program for the new behaviour. In this paper, we focus on learning the precondition for a new behaviour. This is a particularly interesting problem in the case of means-ends behaviours (i.e. where one action is used in order to facilitate another [28]), because it is through learning means-ends behaviours that infants begin to learn about relationships between objects [29]. The precondition must capture the relationship between objects which determines where the behaviour works or does not work. In preconditions the infant is learning new important abstractions over its sensor space. This can change how an infant understands a scene because the infant can begin to see things at a higher level of abstraction, seeing precisely those relationships which are important in determining what object manipulations are possible (by itself or other agents).

III. Methods

In this section we describe the simulated environment, the objects and actions used, the robotic perception system and the classifier algorithm used for experiments and learning in section IV.

A. Experimental Setup

In this work, we collected data using a physically realistic simulation environment [30], [31] designed for robot simulations and a vision system using a simulated Kinect camera (See III-B). As robot we use a simulated six degrees of freedom (DOF) arm mounted on a table with a two finger gripper as its hand.

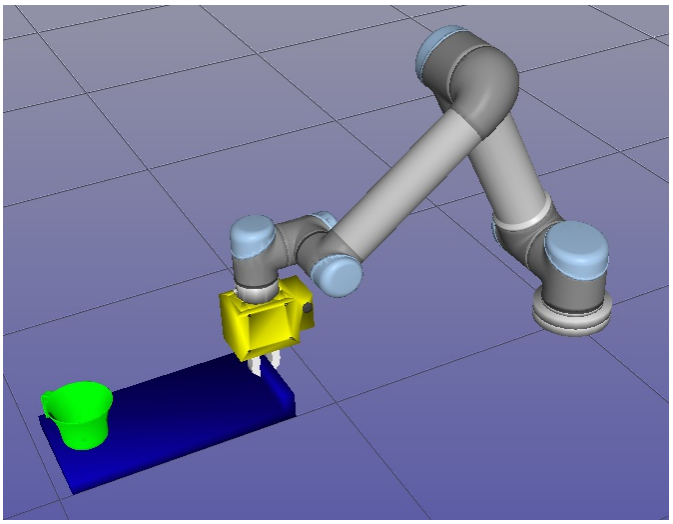


Fig. 2: Illustration of Robot Simulation Environment.

In the experiments, the robot used a set of object pairs (See III-A1) to perform a series of actions (See III-A2) on them and recorded the initial and final *state spaces* (See III-B) and also action success/failure labels. To compare the contribution of this work to earlier work [2], the spatial relations between object pairs in the environment during the initial states have also been labeled[2]. Labels can be one of the following three:

1) "Ontop": One object resides on top of another object, e.g. a cup on top of a kitchen tray
2) "Inside": One object is inside of another object, e.g. a die inside of a box
3) "Beside": The two objects are just lying around, and neither of the two other relations holds.

Code was written to automate the labelling, based on the known positions of the objects from the simulator, and their dimensions (e.g. it was possible to work out if one object was on top of another). The data gathered during these experiments is used in all subsequent learning sections.

In the following, we will describe in more detail the objects used, their distribution in the environment and the actions performed during the experiments.

1) Objects: In our experiments we used an overall set of 24 Objects, which can be grouped into four different groups (See Figures 3 to 6)[3]:

1.	Toys	(5 Objects)
2.	Bases	(15 Objects)
3.	Obstacles	(5 Objects)
4.	Rakes	(5 Objects)

2) Actions: The robot was equipped with nine actions it could perform.

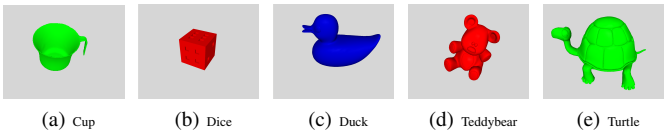
1) Lift: Grasp base object and Lift both objects in one go.
2) Move: Move to toy object and push both objects aside together by pushing against toy.
3) Pull: Grasp base object and Pull both objects in one go.
4) Push: Grasp base object and Push both objects in one go.
5) Rake: Use rake to bring toy object closer.
6) Take: Grasp and Lift toy object without running into other objects, e.g. side of a container.
7) Tilt (pour): Lift and tilt base object with toy object *remaining* in relative position to base object, e.g. Dice in Cup.
8) Tilt (slide): Lift and tilt base object with toy object *changing* relative position to base object, e.g. Cup on Plate.
9) Unobstruct: Grasp and move obstacle object to turn toy object reachable.[4]

The Figures 7 to 15 Illustrate the how the actions work. The subfigures a) and b) show the scene either before or after the initial part of the action, e.g. in Figure 7a shows the robot after moving to grasp the base object, but before Lifting it

[2]Note that these spatial relations recorded here are independent of any actions.

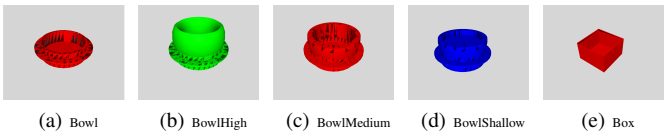
[3]One Object (Cup) is member of two Groups (Toys and Supports/Containers)

[4]Here, reachability of toy object before and after moving potentially obstructing object was tested via inverse kinematics

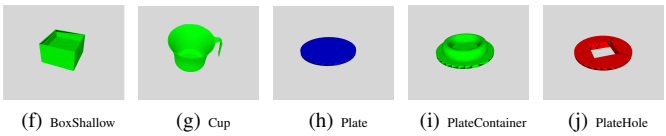


(a) Cup (b) Dice (c) Duck (d) Teddybear (e) Turtle

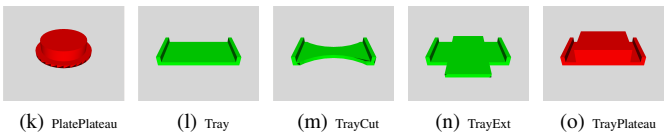
Fig. 3: Toy Objects



(a) Bowl (b) BowlHigh (c) BowlMedium (d) BowlShallow (e) Box

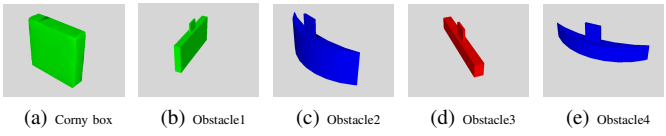


(f) BoxShallow (g) Cup (h) Plate (i) PlateContainer (j) PlateHole



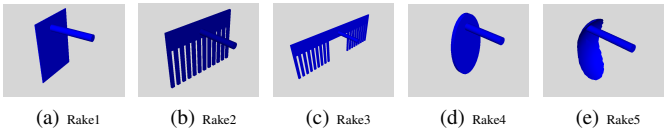
(k) PlatePlateau (l) Tray (m) TrayCut (n) TrayExt (o) TrayPlateau

Fig. 4: Base Objects



(a) Corny box (b) Obstacle1 (c) Obstacle2 (d) Obstacle3 (e) Obstacle4

Fig. 5: Obstacle Objects

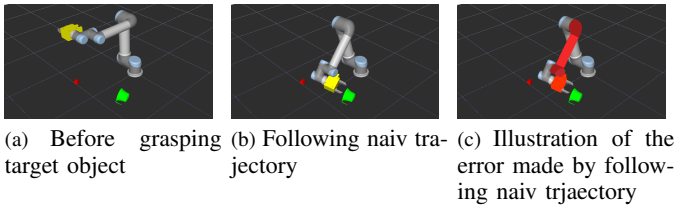


(a) Rake1 (b) Rake2 (c) Rake3 (d) Rake4 (e) Rake5

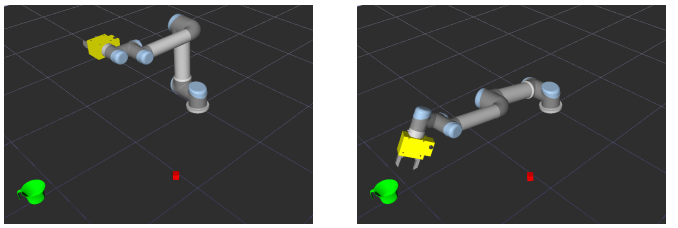
Fig. 6: Rake Objects

as seen in 7b. Subfigures c) and d) show the same action but demonstrate a scene where the action fails. Note, that the actions where preprogrammed and followed simple paths, not taking into account potentially obstructing objects or changes in the environment caused or happening during the action. For example in the case of the Take action illustrated in Figures 12c and 12d, the robot follows a simple straight path towards the toy object to grasp. By doing so the robot runs into the side of the object containing the target toy, moving the container with the toy away. The robot then proceeds with the grasp where it expected to find the toy, leading to action failure.

If the object selected to perform the action on is out of reach, naturally the action will fail (See Figures 16d and 16e.) Another reason for failure is the fact that the action motor program follows a too simple approach. The robot calculates a target position for the gripper to manipulate an object and calculates approriate joint angles via *forward kinematics*. The robot then drives the joint angles directly to the calculated values even if such a path is not possible. See Figures 16a to 16c as examples of such cases.



(a) Before grasping target object (b) Following naiv trajectory (c) Illustration of the error made by following naiv trjaectory



(d) Before attempting grasp on out of reach object (e) After attempting grasp on out of reach object

Fig. 16: Action Failures

For data collection purposes, every action was performed many times on an overall of 57 Object pair combinations (Random combinations of Toy and Base objects). Two exceptions exist; a) the Unobstruct action was performed on 61 object pair combinations as the *Obstacle* objects where used alongside the base objects as potential obstructions. And b) the Rake action, which was perfomed on only 23 object pairs[5]

B. The Robotic Perception System

A system's *state space* represents the robot and its environment to the degree that it is observable to the robot itself. The *state space* of our robot consists of two parts, an internal and an external *state space*.

The internal *state space* describes the state of the robot embodiment (see section III-B1). The external *state space* describes the perceptible environment outside of the robot (see section III-B2).

1) Sensed Internal State Space: In our system, the robot is aware of its own configuration that makes up the internal *internal state space*. In particular the robot "senses" the following properties:

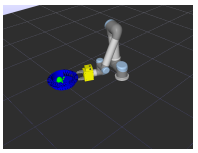
- The Joint Angles of the 6DOF Arm (6 values)
- The resulting Gripper Position (3 values)
- The resulting Gripper Orientation (3 values)
- The finger configuration/openness (1 value)

These 13 values make for the internal *state space*.

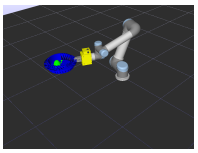
2) Vision Based External State Space: To perceive its environment, the robot is equipped with Kinect-based vision capabilities, enabling the robot to extract information about objects in the scene.

A Kinect is a 3D scanner camera system developed by Microsoft as motion sensing input device for the Microsoft game console Xbox 360. It is a popular alternative to expensive

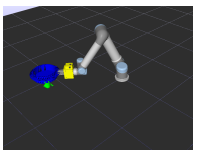
[5]All Rake & Toy object combinations appart from Rake4 & Cup and Rake4 & TeddyBear, due to problems with the simulation with these two object pairs.



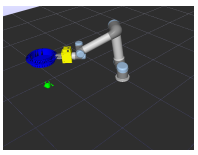
(a) Lift Before Success



(b) Lift After Success

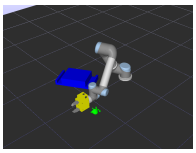


(c) Lift Before Fail

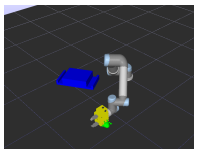


(d) Lift After Fail

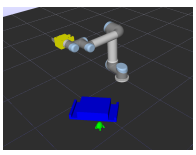
Fig. 7: Lift



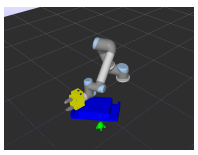
(a) Move Before Success



(b) Move After Success

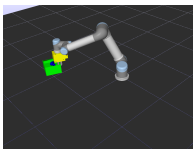


(c) Move Before Fail

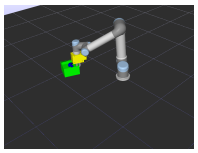


(d) Move After Fail

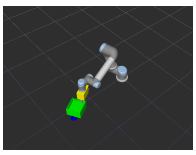
Fig. 8: Move



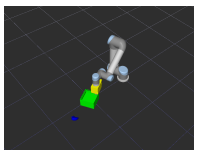
(a) Pull Before Success



(b) Pull After Success

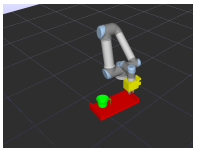


(c) Pull Before Fail

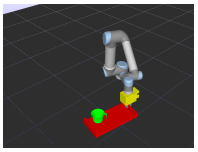


(d) Pull After Fail

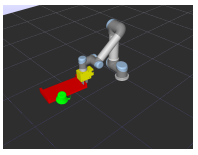
Fig. 9: Pull



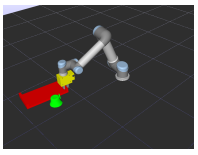
(a) Push Before Success



(b) Push After Success

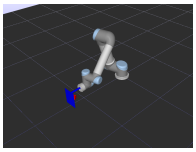


(c) Push Before Fail

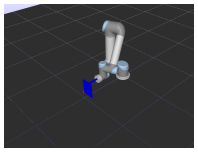


(d) Push After Fail

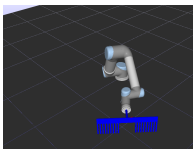
Fig. 10: Push



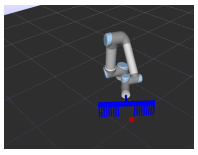
(a) Rake Before Success



(b) Rake After Success

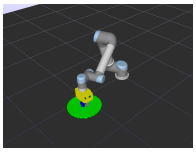


(c) Rake Before Fail

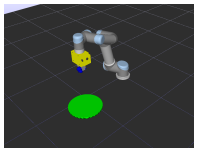


(d) Rake After Fail

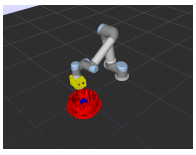
Fig. 11: Rake



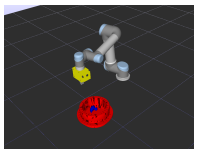
(a) Take Before Success



(b) Take After Success

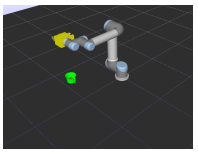


(c) Take Before Fail

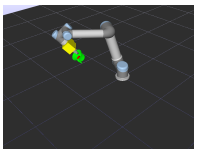


(d) Take After Fail

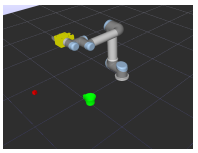
Fig. 12: Take



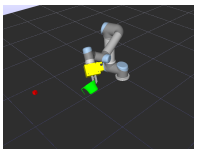
(a) Tilt(pour) Before Success



(b) Tilt(pour) After Success

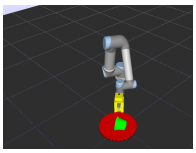


(c) Tilt(pour) Before Fail

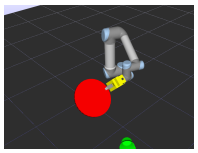


(d) Tilt(pour) After Fail

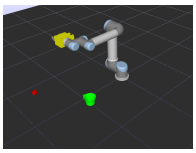
Fig. 13: Tilt(pour)



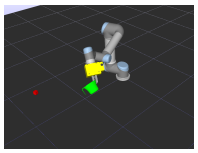
(a) Tilt(slide) Before Success



(b) Tilt(slide) After Success

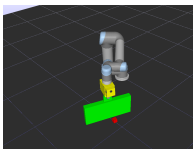


(c) Tilt(slide) Before Fail

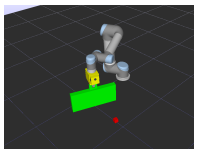


(d) Tilt(slide) After Fail

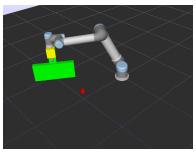
Fig. 14: Tilt(slide)



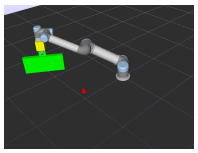
(a) Unobstruct Before Success



(b) Unobstruct After Success



(c) Unobstruct Before Fail



(d) Unobstruct After Fail

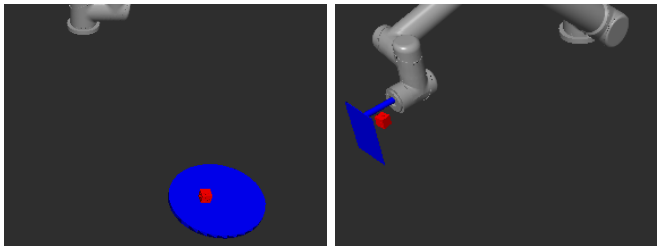
Fig. 15: Unobstruct

stereo camera systems and provides good results in close range applications with up to three meters distance from the Kinect device [32].

Using a simulator for the robotic experiments, we also simulate the Kinect camera, inclusive the noise of real Kinect devices. This gives us data about the depth to the objects in our 3D scene just as we would have obtained from a real Kinect looking at a real scene with 3D objects. The data from the simulated vision system is hence more noisy and less accurate than the perfectly accurate data which could be provided our simulator.

The Kinect camera sensor is mounted at a high position of the work space at the opposite side of the robot, looking down towards the performing robot, as illustrated in Figure 17.

The Kinect records images with *VGA* resolution (640x480 pixels) and the vision system calculates a 3D point cloud



(a) Dice on a Plate (b) Robot using Rake to retrieve Dice

Fig. 17: Kinect camera looking at workspace. Scene (a) shows a die on a plate-like object, with the robot "shoulder" visible at the top. Scene (b) shows the robot using a rake-like tool to pull another object.

point per pixel (307200 points per scene) (See Figure 18). For performance reasons we sub sampled this point cloud and used point clouds with only around 50000 points per cloud (the actual amount of points in the sub sampled cloud varies between 40k and 50k points). Figure 19 highlights the difference between the high resolution and the sub sampled point clouds.

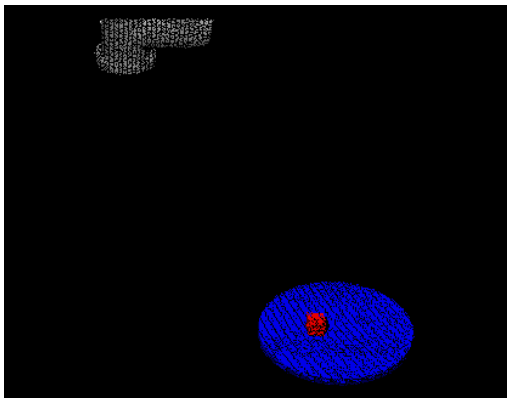
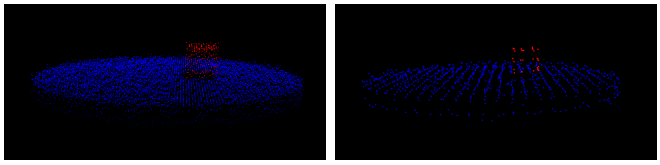


Fig. 18: High resolution point cloud for the same scene as Figure 17a.



(a) High Resolution Point Cloud (b) Low Resolution Point Cloud

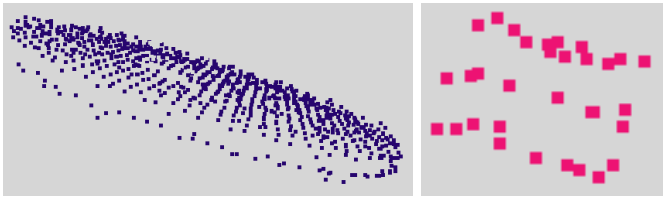
Fig. 19: Illustration of the difference between high resolution (a) and sub sampled point clouds (b).

We acknowledge that highly sophisticated object segmentation algorithms exist [33] and we assume they could be employed to work in a more complex environment. In this work, however, we used a trivial method for Object Segmentation. The method we present here is based on colour information of the " Point Cloud".

For this simple method to work, it is assumed that the objects are coloured in one of a known set of colours. This is a strong assumption also made by others, e.g. Rosman & Ramamoorthy [11], but it could be relaxed by using more sophisticated segmentation methods, which could take into consideration factors like discontinuities of surface curvatures and colour differences.

We coloured our objects either red, blue or green and the background was black and the robot arm/hand Grey. The points where then grouped based on their colour or were neglected if they were Black or Grey (or any colour other than Red, Green or Blue).

After segmentation, each object is assigned its unique set of points. Two segmented point clouds can be seen in Figure 20.



(a) Segmented Plate (b) Segmented Die

Fig. 20: Illustration of the sub sampled and segmented point clouds. (a) shows the Plate and (b) shows the Die

From each segmented point cloud, our vision system extracts, using PCA, the position of the object's centre of gravity, the object's orientation and the object's dimensions. Each of these are described by three variables. These are X, Y and Z for the position, Roll, Pitch and Yaw for the orientation and three size values for the elongation along the objects three PCA axes.

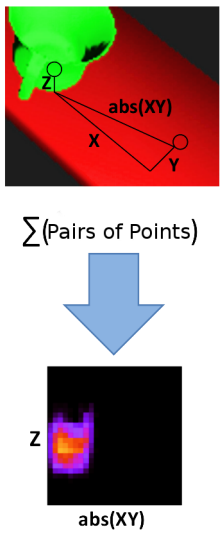


Fig. 21: Illustration of the Histogram creation process from point clouds to histograms.

Using the segmented point cloud based scene representation, we create *Relational Histograms* to capture the spatial relations between objects. These *Relational Histograms* form a relational space into which the absolute geometric information

(3D position and orientation) of the 3D points is transferred. To achieve this transfer, we define a set of relational features which encode the spatial relationship structure of the objects in the scene.

More specifically, for each scene we have two point clouds Π^1 and Π^2 representing the segmented objects 1 and 2 in the scene. For each cross object pair of points of the form $\Pi_i^1 \oplus \Pi_j^2$ we calculate four *Euclidean Distances* $R_d(\Pi_i^1, \Pi_j^2)$ (The Euclidean Distances along the X, Y and Z axes respectively and in the XY plane) and three *Angle Relations* $R_a(\Pi_i^1, \Pi_j^2)$ (The line through the two points is projected onto one of the planes XY, XZ, or YZ, and we look at the angle between the projected line and the axes X, Z and Z respectively). The size of these feature vectors, describing the relation between the two objects in the scene, is variable and determined by the amount of points extracted by the vision system. As we want to apply *Supervised Learning Algorithms*, we need the input vector to be generic and of fixed length, for all possible scenarios.

For this, instead of using the data vectors $R_d(\Pi_i^1, \Pi_j^2)$ and $R_a(\Pi_i^1, \Pi_j^2)$ directly, we compute 1-, 2- or 3 Dimensional *"Relational" Histograms* from the data vectors and use these as learning data input, similar to Mustafa et al. [1].

In this work we experimented with three different kinds of Histograms for learning *spatial relations*. A composite of 1D Relational Histograms, one 2D and one 3D Histogram.

- 1D Histograms capture simple relational features between inter-object pairs of points. For the first 1D composite Relational Histogram, we calculate three 1D histograms capturing the distances between points along each of the three main axes X, Y and Z respectively and put them together as 1D learning input. For the second 1D composite Relational Histogram we compute the angle relations in the 3 planes in the space opened by the three main axes (XY, XZ and YZ planes) and calculate the angles between points in these planes as described above. These angle relations put together alongside the three distance histograms, make up the second 1D Relational Histogram.
- The 2D Histogram used in this work, captures the absolute distance of inter-object pairs of points in the XY plane and puts it into relation with the height difference of the two points (i.e. Z difference). This process is illustrated in Figure 21.
- The 3D Relational Histogram captures distances between points amongst three Dimensions, in a similar fashion as the 2D Histogram does for two Dimensions. For the 3D Histogram, however, we used the actual position differences amongst all three main axes (X, Y and Z). 3D Histograms have not been graphically illustrated in this paper mainly because they did not give particularly good results, so it was less interesting to inspect them visually.

Mustafa et al. [1] have demonstrated the potential of histograms for object recognition. However, we found the lack of generalisation capabilities of Random Forests to be a limitation in their applicability when it comes to learning spatial relations, as it is of major importance to be able, to not only recognise relations between known objects, but also for never before seen objects. In this we differ from Mustafa et al. who aimed at recognizing known objects. Therefore in order to not only increase the learning performance, but especially increase the robustness of recognising spatial relations among novel objects, we implemented some feature vector and histogram post processing methods.

The efficiency of these post processing methods on the spatial recognition rate and robustness was investigated in prior work [34]. Given the clear improvements shown there, we use these methods for all learning in this paper.

- *Histogram Normalisation*

Histogram Normalisation proved to vastly increase the robustness of the recognition rate when it comes to novel object pairs and their relations [34]. This is not surprising as the numbers in the un-normalised histograms rely heavily on the sizes of the objects, and the amount of point cloud points extracted for them by the vision system. Hence, two large objects would generate bigger numbers than two small objects in the same relation. The according histograms would hence look very similar but with different scales. Normalising these histograms removes these scaling effects caused by the sizes of the objects.

E.g. the two imaginary histograms $[1|2|4|1]$ and $[2|4|8|2]$ could describe the same relations for objects pairs with different sized objects. Normalisation would bring both histograms down to $[0.25|0.5|1|0.25]$ and hence remove the differences caused by the object sizes, allowing them to be recognised as the same *spatial relation*.

- *Histogram Smoothing*

Histogram Smoothing using normal Gaussian smoothing considering only direct neighbours (i.e. Window size 3) was also found to increase performance, but with a smaller effect on the robustness in case of novel objects [34]. For Smoothing we applied a standard Gaussian Smoothing algorithm with a variance $\sigma^2 = 1$ and a window size of three bins, i.e. only direct neighbours to values are taken into account for the smoothing.

Smoothing was found especially useful when used on 2D and 3D Histograms as these are naturally quite sparse, also compared to the according 1D histograms. The smoothing accounts for noise in the histograms caused by Kinect camera and the limits in its resolution.

- *Logarithmic Scaling*

We apply Logarithmic Scaling to the feature vectors preceding the creation of Histograms. This logarithmic scaling had the biggest impact on general classification performance [34] but was only applied on distance features; the angle relation features were not scaled as this would not be sensible.

To scale the data, we replaced the original values of the feature vectors, i.e. distances, with $f(x) = ln(x + 1)$. This logarithmic scaling has the effect that in the histograms created

from the scaled feature vectors, for small distances there is a higher resolution than for larger distances. This has a positive effect because in the smaller distances lies the most useful information about *spatial relationships*. It is evident, that if the distance between inter-object pairs of points is large, the two objects are unlikely to be in a "Ontop" or "Inside" relation, but instead are unrelated distributed in the scene.

C. Classifiers

To predict success of actions in a particular scene, we use ensemble classifiers based on the Random Forest [35] algorithm. Random Forests (RF) are particularly well suited for our use case, as they inherently do feature selection and hence identify the relevant features from the (potentially) large amount of *state space* variables (See III-B).

Figure 22 illustrates the a schematic of predicting action success based on internal and external *state space* inputs.

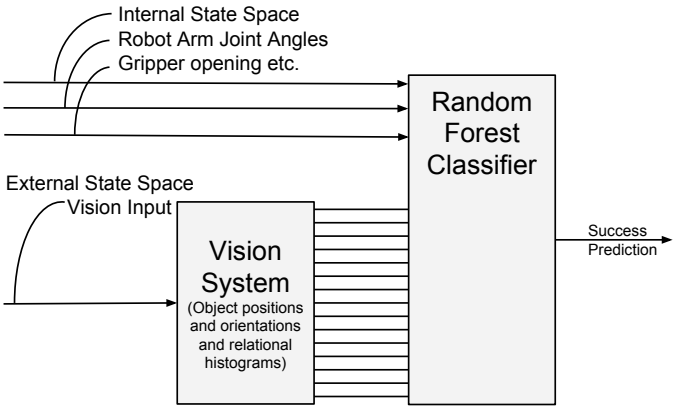


Fig. 22: Learning of Visual Preconditions.

To parameterise the Random Forests we followed Breiman's et al. guidelines as layed out in his paper [35] and on his web page about Random Forests [36]. The amount of trees (ntrees) in our forests is dependent on both, the amount of samples available for training ($nSamples$), and the amount of variables in the *state space* ($nDims$), where

$$nTrees = 150 + \sqrt{nSamples * nDims}$$

For each tree in the forrest we use $\frac{2}{3}$ of the samples in the training set to train the tree. We use standard C4.5 Trees as base classifiers using *Gini impurity criterion* to split nodes and do not prune the Trees in any way. To grow each tree as much as possible, we set the minimum size for a partition is set to 1 and the maximum depth of the tree to the largest possible signed 32bit integer value

$$depthMax = 2^{31} - 1$$

For every split, each tree uses all samples available to it ($\frac{2}{3}$ of the samples available to the Forest). But, for every split, at each node, out of $nDims$, only $m << nDims$ are randomly picked and the best split on these is used to split the node, where

$$m = \sqrt{nDims}$$

IV. Learning Preconditions and Categories for Bootstrapping

In this paper we aim to demonstrate how the learning of preconditions for robot manipulation actions can be accelerated by knowledge transfer from already learnt actions (*Bootstrapping*). More specifically, we find that preconditions learnt for certain actions may implicitly capture a *category*, for example the category of being 'on top', or 'not on top, or 'inside', etc. We avoid calling these *concepts* because a concept suggests a complex package of information, e.g. knowledge of situations or associated actions (see Barsalou []), whereas we are talking about something more restricted: a simple classifier determining the presence of a critical aspect of a scene, e.g. spatial relationship category. These implicit categories can be extracted and treated as an explicit symbol to be used as input to the learning of subsequent preconditions. Our results found that this approach effectively bootstrapped the learning.

A. Learning (Visual) Preconditions

In this section, we will first demonstrate and evaluate learning action preconditions using a "standard" approach without any bootstrapping.

The goal of the classifiers trained here is to accurately predict whether their associated action can be executed successfully in a given scene, depending on the objects and their relative positions. The training input for, and the classifiers themselves, are collected and created as described in section III-A. Figure 22 illustrates a schematic of predicting action success.

These basic precondition classifiers serve two purposes: Firstly, they illustrate the "standard" learning rate for learning preconditions when learning without bootstrapping and will be used as a baseline for comparison in the following sections. Secondly, they constitute the "knowledge" which will be used for bootstrapping in the following sections.

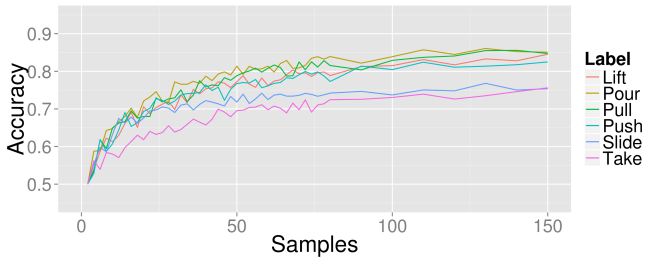
Figures 23a to 23d illustrate the learning rate for the different actions using different *state space* representations.

As can be seen by comparing the different Figures, an appropriate *state space* representation is of significant importance for learning. The more expressive state representation using Histograms massively outperforms learning without Histograms. The "Take" action, however, serves as a good example of the potential shortcomings of hand designed *state spaces*. The extended state space representation does not benefit the "Take" precondition classifier, instead, the increased amount of input variables causes a decrease of it's learning speed (curse of Dimensionality).

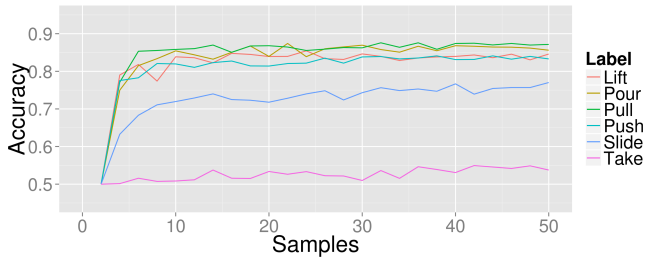
In Section V we not only demonstrate that competetive improvements can be achieved using bootstrapping in a developmental robotics approach, rather than engineering state space representations. We also demonstrate that both approaches can work together where bootstrapping further increases the learning performance in hand designed state spaces.

B. Learning of (Visual) Categories

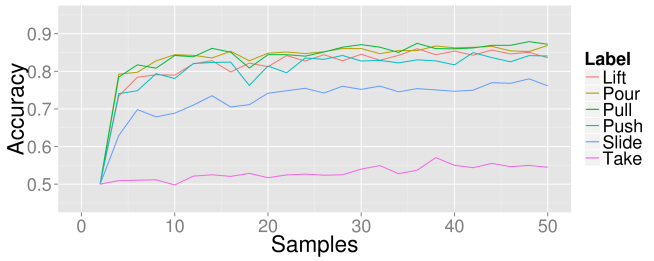
In this section we compare two different approaches for obtaining categories. These categories will subsequently be



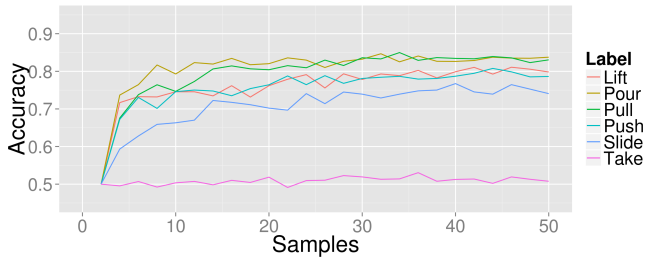
(a) Learning without Histogram based state space.



(b) Learning with 1D Histogram based state space.



(c) Learning with 2D Histogram based state space.



(d) Learning with 3D Histogram based state space.

Fig. 23: Illustration of the precondition learning speed for 6 actions in different state spaces without bootstrapping.

used to bootstrap learning of preconditions in Section V.

1) Manual Learning of Categories: Our first approach for learning categories relies on human supervision to define categories expected to be useful, e.g. the spatial second order categories "On top", "Inside" and "Beside" (See Section III-A for a more detailed description of the spatial categories used). This is an approach we followed previously in [2], where we trained classifiers to recognise *spatial categories*.

In this work we partly repeated this work on our new and extended sample data base, but focussed on the spatial categories mentioned above.

These manually learnt categories will serve as another baseline for comparison. They demonstrate the performance boost achievable via bootstrapping when using engineered categories, and will compete against autonomously found categories.

Section V-B will demonstrate how these recognised categories can be used to improve the speed of convergence when learning action precondition classifiers.

Figure 24 illustrates the performance of recognising the manually chosen categories in Different State spaces. As can be seen, the Histogram based state spaces allow for higher performance. These results, however, are based on only 300 training samples and 150 validation samples. Both, training set and validation set, incorporate data from the same object pair combinations and only a subset of two toy objects (Duck and Dice) and six base objects (Bowl, Box, BoxShallow, Plate, PlateContainer and Tray) have been used. For a more thorough analysis of spatial category learning we refer the reader to [2].

2) Automatic Detection and Learning of Categories: One contribution of this work is the automatic detection of categories in the environment. Unlike above, the goal is to have the robot find these categories of the environment through

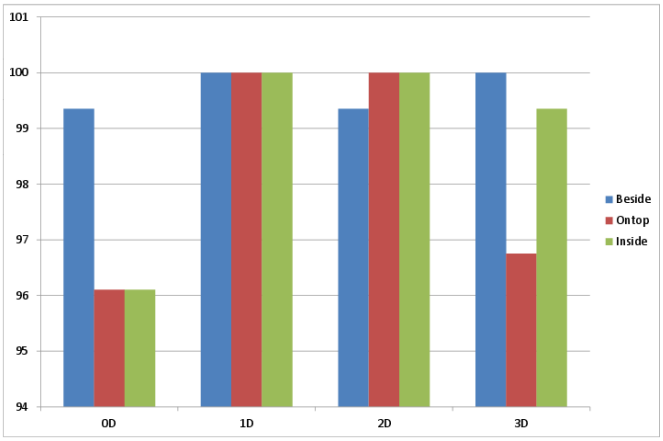


Fig. 24: Illustration of spatial category recognision performance in different state spaces. 0D represents a state space without Histogram extension, while 1D, 2D and 3D stand for respective Histograms used to extend the state space.

interaction, without utilising human knowledge.

Our approach aims to focus on categories that are indeed useful, not just any category in the environment as e.g. unsupervised Learning / clustering could find them. Our hypothesis here is, that basing the creation of categories on already established predictors and their commonalities, lets us find categories that describe exactly these commonalities that are in fact important for predicting some already existing action outcomes. We assume that categories important for at least two predictors might be important for further action predictors and are therefore worthwhile to create and extend the state space with. Categories in the environment that are unpredictive, i.e.

not useful for predicting the outcome of actions, would only clutter up the state space without adding any benefit.

The robot can automatically find the categories as follows: After learning at least two action precondition classifiers, the system records the correlation between pairs of classifiers. To compute the correlation between classifiers, both classifiers are presented with the same inputs, e.g. as the robot interacts with the environment. The robot then records the prediction of each of the classifiers for all inputs and calculates the correlation between these predictions.

This correlation serves as a heuristic, indicating that there might be a common underlying category shared between the two actions and their preconditions. If the correlation between two classifiers exceeds a certain threshold, a category classifier is created. Figure 25 illustrates the search for correlations between two action precondition classifiers.

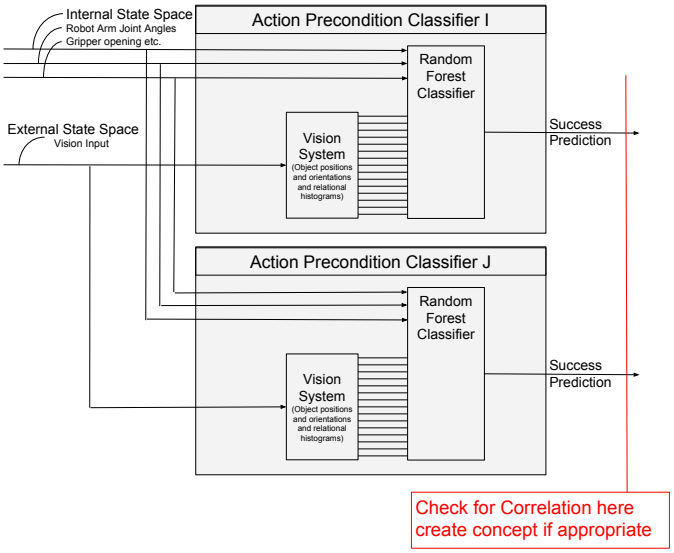


Fig. 25: Searching for correlations between precondition classifier as heuristic for learning new categories.

There are different possibilities how a category classifier could be created from two individual classifiers. In this work, we describe and present results of the simplest method.

To create a category classifier, we combine the training data that was used to train the two or more individual classifiers that were found to correlate. This combined training set is then used to train a new classifier, that encodes the underlying commonalities of the original classifiers. Figure 26 illustrates the creation of a category classifier.

Internally we use supervised learning to create classifiers here, however we do not report any learning performance results at this point. The idea here is that the classifiers capture arbitrary potential underlying commonalities. Reporting performance is meaningless without understanding what the classifiers actually captured. Our hyptothesis is, that these automatically created categories resemble, to some degree, the manually created spatial categories from Section IV-B1. We are in the process of analysing the similarities between manually and automaticaly created categories and will report our findings in a future publication. Most likely, automatically

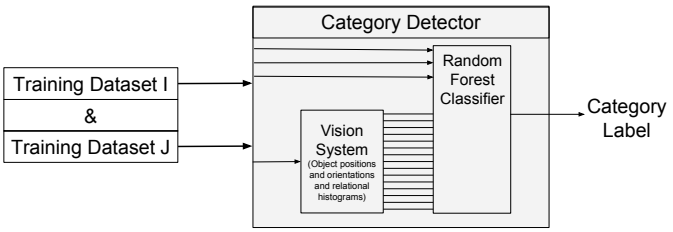


Fig. 26: Illustration of creating a new category classifier. Using the combined data, used for learning individual preconditions, a new classifier is created to detect common underlying categories.

created category classifiers capture even more knowledge, e.g. the fact that objects can not be manipulated if they are too far away from the robot. Unlike the manually created categories, the automatically created ones will capture multiple such commonalities between actions at the same time.

V. Transfer of Knowledge for Bootstrapping

In this Section we describe our approach to bootstrapping in more detail and present results using different knowledge *Sources* for bootstrapping.

The idea behind bootstrapping is to reuse knowledge to increase learning speed of new actions. We achieve bootstrapping by using the classifiers presented in Section IV-A as knowledge *sources*. In particular, we add the output (prediction) of already learnt action precondition classifiers or *category* classifiers as input for learning of new problems. Figure 27 illustrates the general structure of learning with bootstrapping.

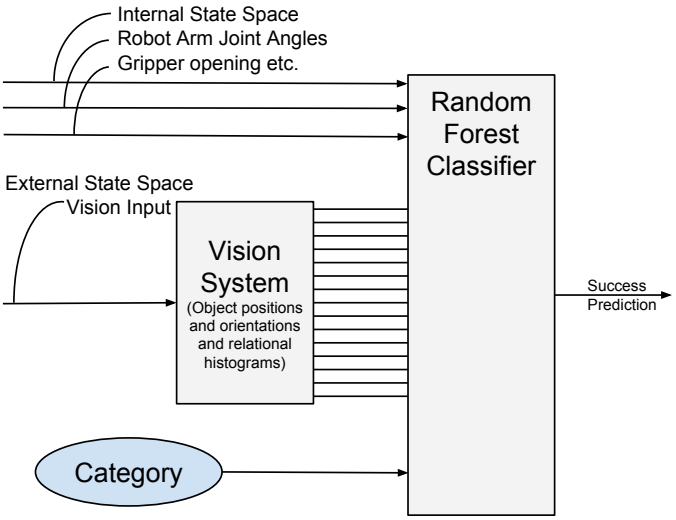


Fig. 27: Bootstrapping learning by reusing existing (category) knowledge.

In the following subsection we will first present brief results of the individual approaches to bootstrapping. A more thorough comparison of the different approaches to bootstrapping can be found in section V-D.

A. Bootstrapping With Already Learnt Action Preconditions As Knowledge Source

The most basic approach to bootstrapping followed here, is extending the state space of a new action precondition classifier with the prediction output of a precontidion classifier learnt for a different action.

If these two actions are related or at least have similar preconditions required for the actions to succeed, then knowing the prediction for one action can be seen as a heuristic to predict success of the other action.

Figure 28 demonstrates the bootstrapping results of this approach in the different *state spaces*. Comparing 28a with Figure 23a from above, the increase in learning speed achievable through bootstrapping becomes evident.

Figure 29 illustrates the bootstrapping effect on the "Lift" action when using different actions as Input. As can be seen bootstrapping "Lift" with "Pull" a significant increase in learning speed can be achieved, while bootstrapping "Lift" with "Take" does not improve learning speed, but is slightly counter productive instead. This confirms that positive results require a "good" knowledge source for bootstrapping, e.g. in this case similar or related actions.

B. Bootstrapping With Manualy Learned Categories

In our previous work [2], we argued for the usefulness of recognising *spatial relations* for (multi object) manipulation tasks in complex environments. In this work, we want to put our claims to the test and evaluate whether - and if so, to what extend - our manually learnt abstractions can be used to improve a robotic system's performance.

Following the approach outlined above, we extend the state space of the action precondition classifiers with the output of the *spatial category* recognising classifiers.

Figure 30 demonstrates the bootstrapping result of this approach in different state spaces. Again, with the plain state space without Histogram in Figure 30a, the bootstrapping effect is more evident than in state spaces with Histograms.

Figure 31 illustrates the bootstrapping effect on the "Lift" action when using different spatial categories as Input. As in Figure 29, one can see in Figure 31 that the availability of an appropriate knowlegde source is crucial.

C. Bootstrapping With Automatically Created Categories

The final approach to bootstrapping followed here, is to extend the state space of a new action precondition classifier with the output of an automatically created category recognising classifier (see Section IV-B2 above).

Similar to our reasoning in Section V-A and Section V-B, we expect positive bootstrapping results if the actions used for creating the category are related to the new action (as in Section V-A) or if the underlying category abstracted from these actions describes a more universal aspect of the state space (as in Section V-B).

Figure 32 Illustrates the bootstrapping result of this approach in different state spaces (as before, the bootstrapping effect is most apparent in the state space without histogram 32a).

Figure 33 illustrates the bootstrapping effect on the "Lift" action when using different automatically created categories as Input. Two things can be noted here: a) the amount of categories potentially created in our approach increases polynomial with the number of actions available. A heuristic for pruning out candidates before creating and trying categories is inevitable, b) the different category inputs are less clearly separable into useful/useless than in the results above. This is due to both, more inputs than before means more mediocre outcomes and even from unrelated actions (unrelated to the target action "Lift" in this case), universally "useful" knowledge can be extracted and used for bootstrapping with many actions.

D. Comparison of Results

Here we will more thoroughly compare the different approaches to bootstrapping presented above.

Figure 34 presents the results of the different approaches to bootstrapping in the state space without Histograms.

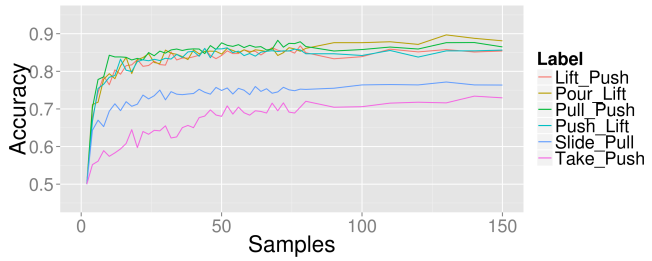
Plot 34a shows learning without bootstrapping.

Plot 34b shows learning with bootstrapping using another action as knowledge source. These results show that this approach to bootstrapping can compete with hand designed state space representations, in terms of achievable learning speed of actions. Similar to the results presented in Figures 23b to 23d, we can achieve near final performance with only 10 - 15 training samples, even in simple and unoptimised state spaces.

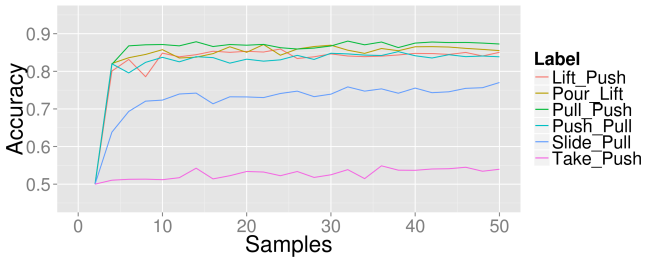
Plot 34c The results show that manually learnt spatial relationship categories outperform other action classifiers when it comes to bootstrapping.

Plot 34d This result shows that automatically created categories outperform other action classifiers as knowledge source for bootstrapping, but they fall behind the manually learnt categories from Figure 34c. This is not surprising, however, as we created our manually learnt categories with the aim of improving performance of actions like the ones used in this work. Also, manually created categories cover single fundamental categories, unlike the automatically created categories which might weakly capture multiple aspects of the environment at the same time (See Section IV-B2). If not all these captured aspects are useful for bootstrapping a new action precondition classifier, the result will be inferior to a category that is more selective in the aspects covered. It is part of our planned future work, to change the automatic creation of categories in a way that leads to more individual categories, instead of a mixture thereof. Hierarchical learning or Unsupervised learning approaches might be the way forward.

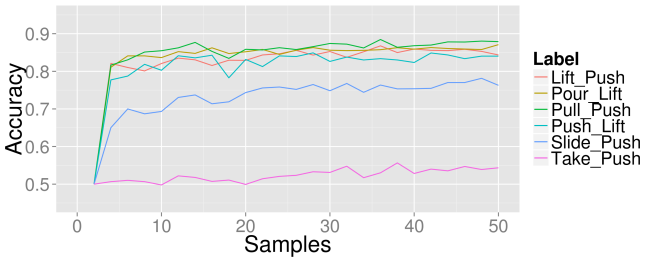
Figure 35 shows the different approaches to bootstrapping on three selected actions learnt with simple state space without Histograms (left column) and 3D Histogram based state space (right column). It can be seen, that even in hand designed state spaces like the 3D Histogram based one, bootstrapping can further improve the learning speed. More than in Figure 34, the order of performance gains of the different approaches is evident here. In both state spaces and for all three actions, the spatial category based bootstrapping has the best performance.



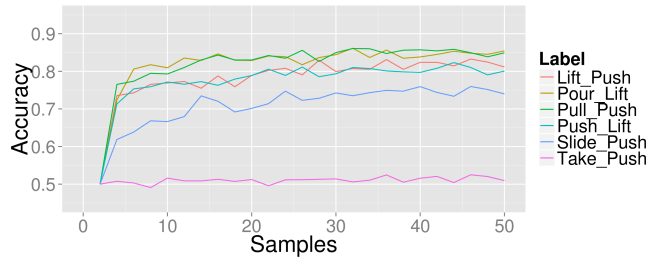
(a) Learning without Histogram based state space



(b) Learning with 1D Histogram based state space

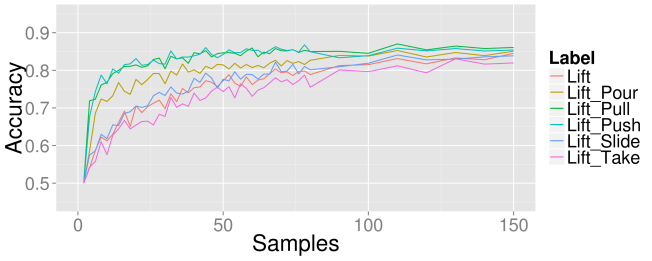


(c) Learning with 2D Histogram based state space

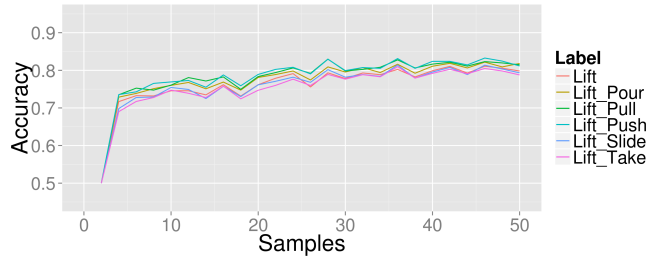


(d) Learning with 3D Histogram based state space

Fig. 28: Illustration of the precondition learning speed for 6 actions in different state spaces using another action precondition output for bootstrapping. Of all the potential knowledge sources for bootstrapping, for each action, only the best result is presented.

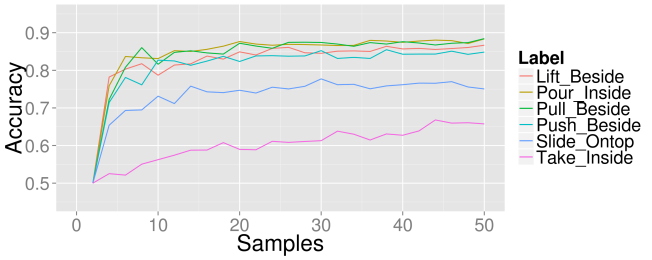


(a) Learning without Histogram based state space

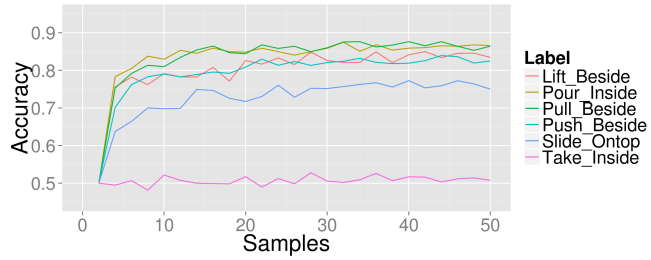


(b) Learning with 3D Histogram based state space

Fig. 29: Illustration of different bootstrapping results for Learning preconditions for the “Lift” action in (a) 0D & (b) 3D Histogram based state spaces using different action classifiers as knowledge sources.



(a) Learning without Histogram based state space



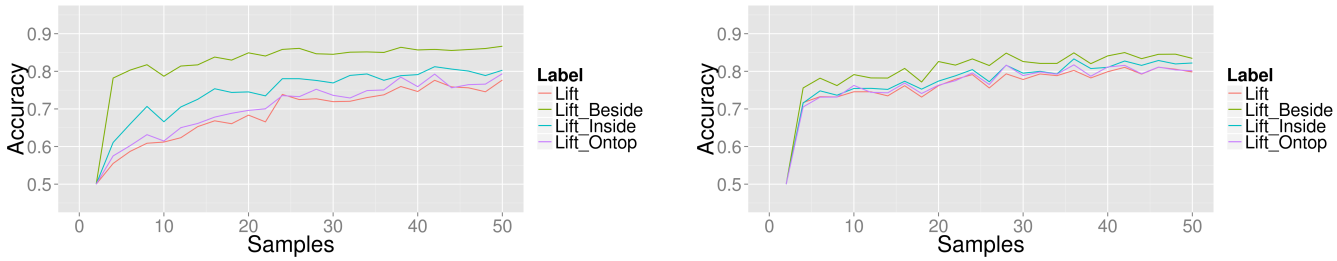
(b) Learning with 3D Histogram based state space

Fig. 30: Illustration of the precondition learning speed for 6 actions in different state spaces using *spatial categories* for bootstrapping. Of all the potential knowlede sources for bootstrapping, for each action, only the best result is presented.

For bootstrapping with other actions and bootstrapping with automatically created categories, this is less clear, however the latter seems to tend to perform better than the first.

ACKNOWLEDGMENTS

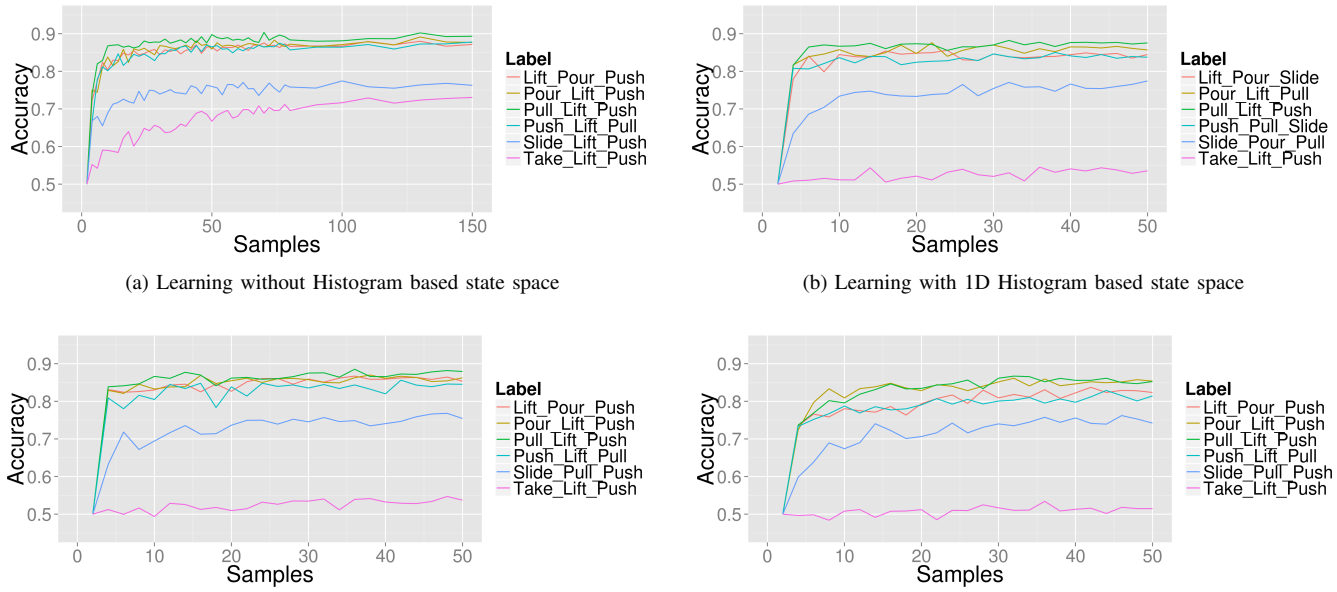
This work was supported by the EU Cognitive Systems project XPERIENCE (FP7-ICT-270273).



(a) Learning without Histogram based state space

(b) Learning with 3D Histogram based state space

Fig. 31: Illustration of different bootstrapping results for Learning preconditions for the "Lift" action in (a) 0D & (b) 3D Histogram based state spaces using different *spatial category* classifier outputs as knowledge sourse for bootstrapping



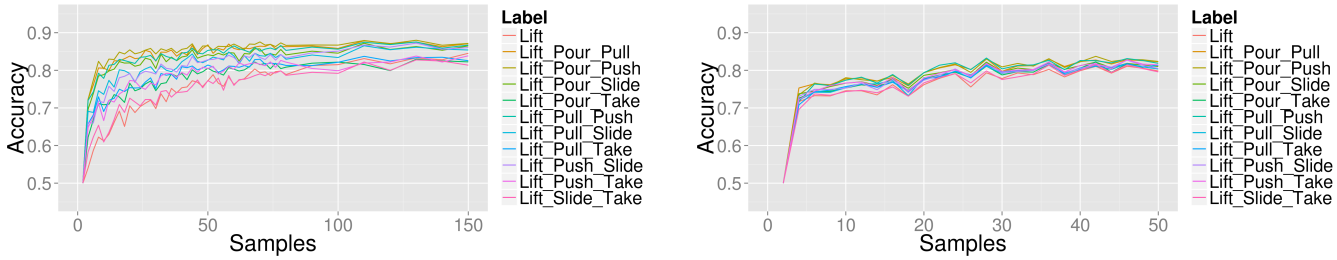
(a) Learning without Histogram based state space

(b) Learning with 1D Histogram based state space

(c) Learning with 2D Histogram based state space

(d) Learning with 3D Histogram based state space

Fig. 32: Illustration of the precondition learning speed for 6 actions in different state spaces using automatically created categories for bootstrapping. From all categories available for bootstrapping each action, only the best result is presented



(a) Learning without Histogram based state space

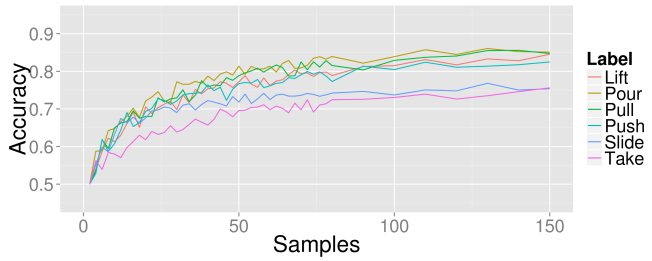
(b) Learning with 3D Histogram based state space

Fig. 33: Illustration of different bootstrapping results for Learning preconditions for the "Lift" action in (a) 0D & (b) 3D Histogram based state spaces using different automatically created category classifier outputs for bootstrapping.

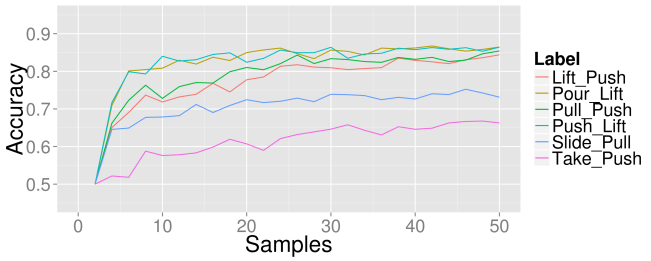
REFERENCES

[1] W. Mustafa, N. Pugeault, and N. Krüger, "Multi-view object recognition using view-point invariant shape relations and appearance information," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

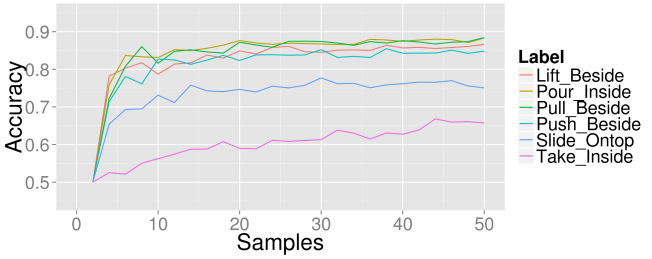
[2] S. Fichtl, A. McManus, W. Mustafa, D. Kraft, N. Kruger, and F. Guerin, "Learning spatial relationships from 3D vision using histograms," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong: IEEE, May 2014, pp. 501–508. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6906902



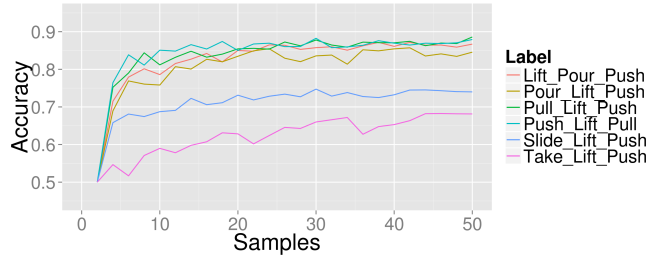
(a) Learning without bootstrapping



(b) Learning with another action as knowledge source for bootstrapping



(c) Learning with spatial categories as knowledge source for bootstrapping



(d) Learning with automatically created categories as knowledge source for bootstrapping

Fig. 34: Comparisson of approaches to bootstrapping in state space without Histogram. Only the best result of each action with bootstrapping is presented

[3] A. Stoytchev, "Some basic principles of developmental robotics," *IEEE Transactions on Autonomous Mental Development*, vol. 1, no. 2, pp. 1–9, 2009.

[4] R. Moore, "Spoken Language Processing: Where Do We Go from Here?" in *Your Virtual Butler*, ser. Lecture Notes in Computer Science, R. Trappl, Ed. Springer Berlin Heidelberg, 2013, vol. 7407, pp. 119–133. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-37346-6_10

[5] M. Do, J. Schill, J. Ernesti, and T. Asfour, "Learn to wipe: A case study of structural bootstrapping from sensorimotor experience," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 1858–1864.

[6] E. Ugur, S. Szedmak, and J. Piater, "Bootstrapping paired-object affordance learning with learned single-affordance features," in *The Fourth Joint IEEE Intl. Conf. on Development and Learning and on Epigenetic Robotics (ICDL-Epirob), Genoa, Italy*, 2014, pp. 468–473.

[7] J. Piaget, "The Origins of Intelligence in Children," 1952.

[8] P. Willatts, "Pulling a support to retrieve a distant object," *Developmental Psychology*, vol. 35, no. 3, pp. 651–667, 1999.

[9] R. A. Brooks, "Intelligence without representation," *Artificial Intelligence*, vol. 47, pp. 139–159, 1991.

[10] R. S. Sutton, "Verification, the key to ai," 2006, unpublished document, available on author's webpage http://www.cs.ualberta.ca/∼sutton /IncIdeas/KeytoAI.html.

[11] B. Rosman and S. Ramamoorthy, "Learning spatial relationships between objects," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1328–1342, Sep. 2011. [Online]. Available: http://dx.doi.org/10.1177/0278364911408155 http://ijr.sagepub.com/content/30/11/1328.abstract

[12] S. Fichtl, J. Alexander, D. Kraft, J. Jø rgensen, N. Krüger, and F. Guerin, "Learning object relationships which determine the outcome of actions," *Paladyn*, vol. 3, no. 4, pp. 188–199, 2012. [Online]. Available: http://dx.doi.org/10.2478/s13230-013-0104-x

[13] E. Ugur, E. Oztop, and E. Sahin, "Goal emulation and planning in perceptual space using learned affordances," *Robotics and Autonomous Systems*, vol. 59, no. 7–8, pp. 580–595, 2011.

[14] J. Sinapov, C. Schenck, K. Staley, V. Sukhoy, and A. Stoytchev, "Grounding semantic categories in behavioral interactions: Experiments with 100 objects," *Robotics and Autonomous Systems*, vol. 62, no. 5, pp. 632–645, May 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092188901200190X

[15] A. Baranes and P.-Y. Oudeyer, "Robust intrinsically motivated exploration and active learning," in *Development and Learning, 2009. ICDL 2009. IEEE 8th International Conference on*, Jun. 2009, pp. 1–6.

[16] S. Fichtl, J. Alexander, F. Guerin, J. A. Jorgensen, D. Kraft, and N. Krueger, "Rapidly learning preconditions for means-ends behaviour using active learning," in *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, Nov. 2012, pp. 1–2.

[17] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[18] A. Lazaric, "Transfer in Reinforcement Learning: A Framework and a Survey," in *Reinforcement Learning: State of the Art*, ser. Adaptation, Learning, and Optimization, M. Wiering and M. van Otterlo, Eds. Springer Berlin Heidelberg, 2012, pp. 143–173. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-27645-3_5

[19] F. Doshi-Velez and G. D. Konidaris, "Transfer Learning by Discovering Latent Task Parametrizations," in *NIPS 2012 Workshop on Bayesian Nonparametric Models for Reliable Planning And Decision-Making Under Uncertainty*, 2012.

[20] B. C. da Silva, G. D. Konidaris, and A. G. Barto, "Learning Parameterized Skills," in *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[21] G. Drescher, "Made-Up Minds: A Constructivist Approach to Artificial Intelligence," 1991.

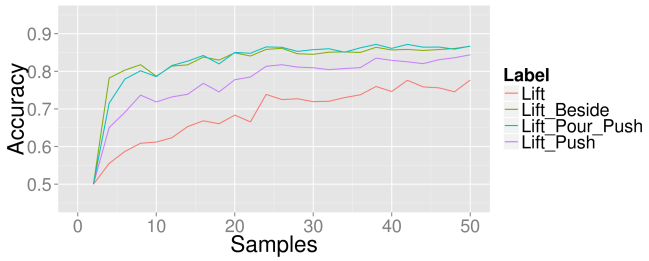
[22] H. Chaput, "The Constructivist Learning Architecture: A Model of Cognitive Development for Robust Autonomous Robots," *PhD Thesis*, no. The University of Texas at Austin, Artificial Intelligence Laboratory, 2004.

[23] S. Fichtl, J. Alexander, F. Guerin, W. Mustafa, D. Kraft, and N. Kruger, "Learning spatial relations between objects from 3D scenes," in *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*, Aug. 2013, pp. 1–2.

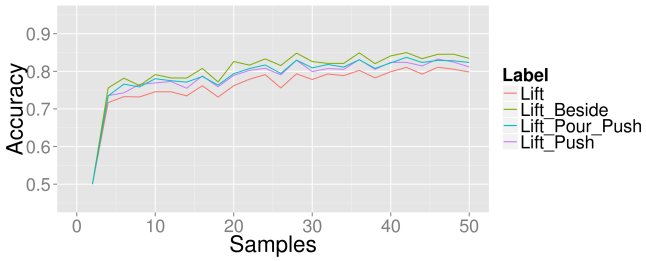
[24] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt, "Learning relational affordance models for robots in multi-object manipulation tasks," in *IEEE Intl. Conf. on Robotics and Automation*, 2012, pp. 4373–4378.

[25] S. Panda, A. H. A. Hafez, and C. V. Jawahar, "Learning support order for manipulation in clutter," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov. 2013, pp. 809–815.

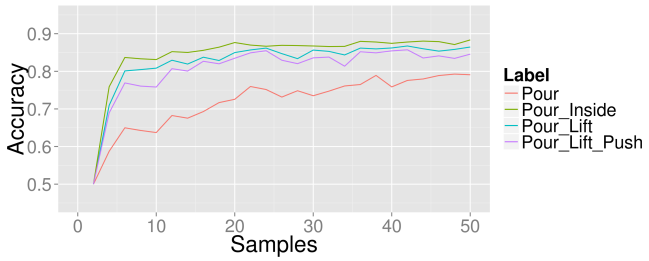
[26] W. Choi, Y.-W. Chao, C. Pantofaru, and S. Savarese, "Understanding indoor scenes using 3d geometric phrases," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.



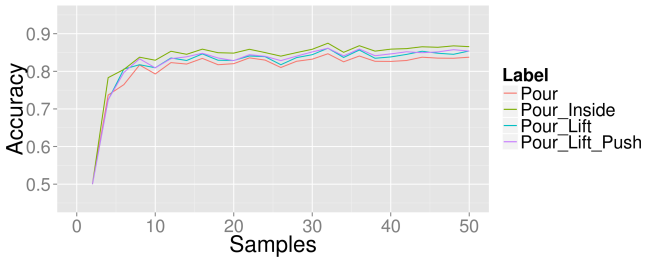
(a) Learning "Lift" without Histogram based state space



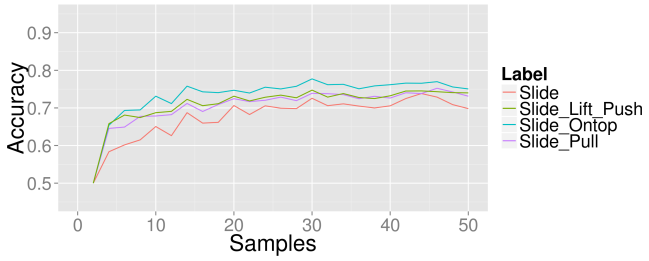
(b) Learning "Lift" with 3D Histogram based state space



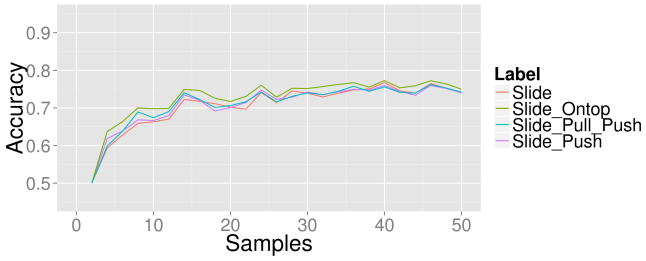
(c) Learning "Pour" without Histogram based state space



(d) Learning "Pour" with 3D Histogram based state space



(e) Learning "Slide" without Histogram based state space



(f) Learning "Slide" with 3D Histogram based state space

Fig. 35: Comparison of the different approaches to bootstrapping on 3 selected actions learnt with simple state space without Histograms (left column) and 3D Histogram based state space (right column). In each Subfigure, each approach to bootstrapping is represented with it's best performing candidate.

[27] J. J. Lockman, "A perception-action perspective on tool use development," *Child Development*, vol. 71, no. 1, pp. 137–144, 2000.

[28] P. Willatts, "Development of problem-solving strategies in infancy," in *Children's Strategies: Contemporary Views of Cognitive Development*, D. Bjorklund, Ed. Lawrence Erlbaum, 1990, pp. 23–66.

[29] J. Piaget, *The Construction of Reality in the Child*. London: Routledge & Kegan Paul, 1937, (French version 1937, translation 1955).

[30] G. Kootstra, M. Popović, J. A. Jø rgensen, K. Kuklinski, K. Miatliuk, D. Kragic, and N. Krüger, "Enabling grasping of unknown objects through a synergistic use of edge and surface information," *Int. J. Rob. Res.*, vol. 31, no. 10, pp. 1190–1213, Sep. 2012. [Online]. Available: http://dx.doi.org/10.1177/0278364912452621

[31] J. A. Jørgensen, L.-P. Ellekilde, and H. G. Petersen, "RobWorkSim - an Open Simulator for Sensor based Grasping," in *ISR/ROBOTIK 2010 (41st International Symposium)*. VDE-Verlag, Jun. 2010. [Online]. Available: http://www.vde-verlag.de/proceedings-en/453273198.html

[32] K. Khoshelham and S. O. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012. [Online]. Available: http://www.mdpi.com/1424-8220/12/2/1437

[33] K. M. Varadarajan and M. Vincze, "Object part segmentation and classification in range images for grasping," in *Advanced Robotics (ICAR), 2011 15th International Conference on*, Jun. 2011, pp. 21–27.

[34] A. McManus, "Learning Spatial Relationships," *Master Thesis*, 2013.

[35] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[36] L. Breiman and A. Cutler, "Random Forests," 2004. [Online]. Available: https://www.stat.berkeley.edu/ breiman/RandomForests/cc_home.htm

Object-action relation learning and replacement infrastructure: ROAR server interface

Sandor Szedmak
University of Innsbruck
sandor.szedmak@uibk.ac.at

January 12, 2015

Contents

1 Introduction

The object-action relation learning and replacement infrastructure is built around the ROAR module. That module behaves as a certain type of object memory where the set of available or potentially available objects together with their affordances are stored. The ROAR stands for *repository of objects and attributes with roles*. This prior knowledge can be recorded by hand made annotations or by experiences carried out on real robots or simulators. It allows objects to be retrieved by their attributes, and the attributes of novel objects can be inferred. General description and details about the aim of the ROAR can be found in Deliverable D3.1.2. Here we focus on the interface of the ROAR and its internal structure.

2 General view

The ROAR behaves as an active database(DB), which not only stores and returns the data items, but via machine learning tools it extends the database with predicted elements. In this way it can provide data not observed earlier by the users connected to the database. The active database might be called as "Intelligent Relational Database" as well.

Considering the ROAR as an enriched DB has several advantages:

- The operations acting on the DB, inserting new data, updating the available ones, and executing queries on the available knowledge stored in the DB, can be implemented in a general, standard way.
- The standard realization of the DB operations allow the users to apply sophisticated program packages, e.g. object-relational database management systems (ORDBMS).
- Object-relational database management systems can provide
  - consistent handling of complicated relationships between the different data categories,
  - complex querying tools, which allow conditional data selections to be combined with different kind of set operations, e.g. automatic intersection, union, on the data satisfying some given conditions.
  - a unified programming environment, e.g. SQL, to implement the data handling tasks.
  - concurrent handling of the database, thus several users can load and request data in the same time without additional programming efforts.

In the reliable implementation of the ROAR all the above mentioned properties can play crucial roles. The learning system of the ROAR has to extend the available knowledge concurrently with users loading new information, e.g. web mining, or users requesting predicted data to carry out object or action substitutions, e.g. plan execution monitor.

Example 1. *Consider the situation:*

By an insertion of new data the learner is triggered to predict the missing items and incorporate them into the database. The learner starts a new training process and when it is finished inserts the new data or updates relating to the predictions. During the execution of the training some the new data insertion, update and query requests could arrive in the DB system from any users. The ROAR control system has to deal with the synchronization of the training and those requests, and preserve the consistency of the database.

The general layout of the ROAR with its environment is presented by Figure 1.

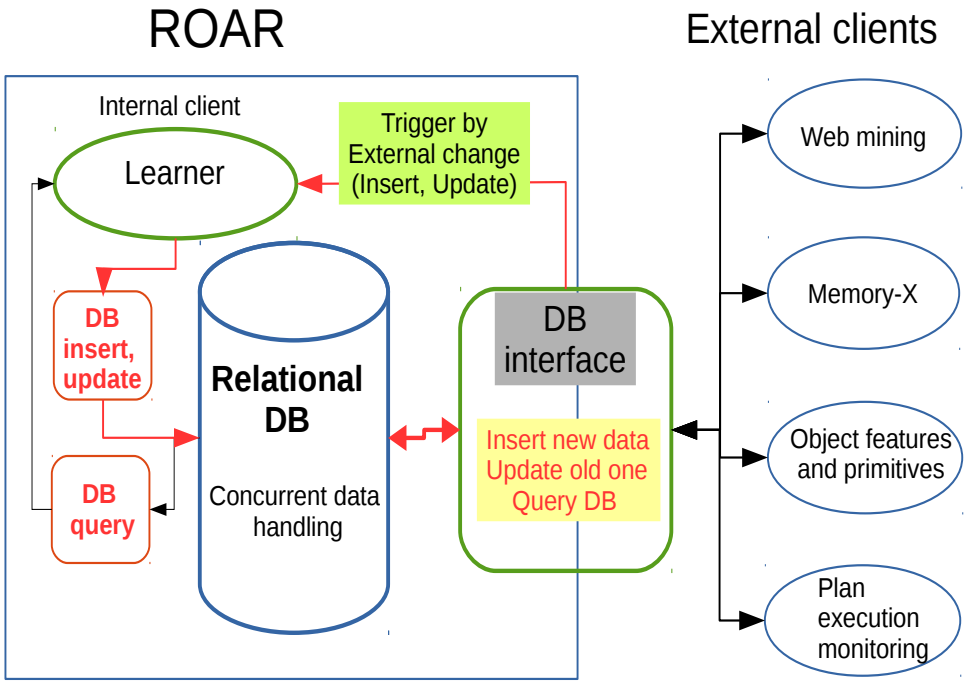


Figure 1: General layout of the internal structure and the interface of the ROAR

The base structure of the DB is displayed in Figure 2

An additional requirement that the system needs to fulfill is the high level of independence from the programming languages (e.g. C++, Java, Python), of operating systems(e.g. Linux based systems, Windows) and of the applied network connection system (e.g. ICE, ROS).

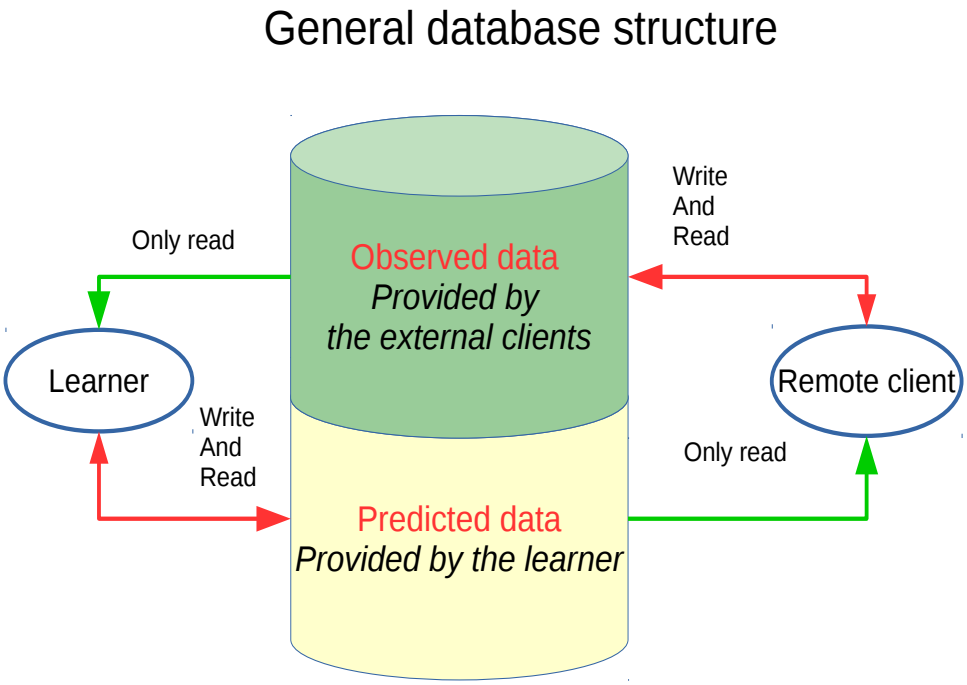


Figure 2: Database structure: the learner generates new, predicted database records based on the available observations

Further requirement is that the components which addresses the problems mentioned above have to have an open-source implementation, to be freely available and to have a large, active developing community.

Based on all these requirements the PostgreSQL (see general description on the web site: `postgresql.org`) ORDMBS system seems to be a good choice. It has highly advanced capability to handle concurrent processes, and its SQL implementation imposes very few restrictions on the general SQL specification, in this way all the SQL commands needed for the realize ROAR functionality can be used without restrictions.

PostgreSQL runs a full server-client model and has an extensive support to work in an online environment, e.g. www, by directly connecting remote users to the database without the need for additional packages.

3 SQL-type interface

The ROAR formally behaves as a RDBMS (Relational Database Management System). It provides a table view on all the stored data.

The basic commands allowing the database handling are implemented via SQL (Structures Query Languages). The insertion of the new data items, the updates of the earlier stored ones and returning the contents of

the database are carried out via SQL commands.

The SQL interface can provide a highly transparent interface which can be implemented in several different programming languages (C++, Python, JAVA, etc.) The SQL uses string based commands to realize all the data handling functionality, see details in Section "Database interface".

Some alternatives of the connection of the users to the database are shown in Figure 3.

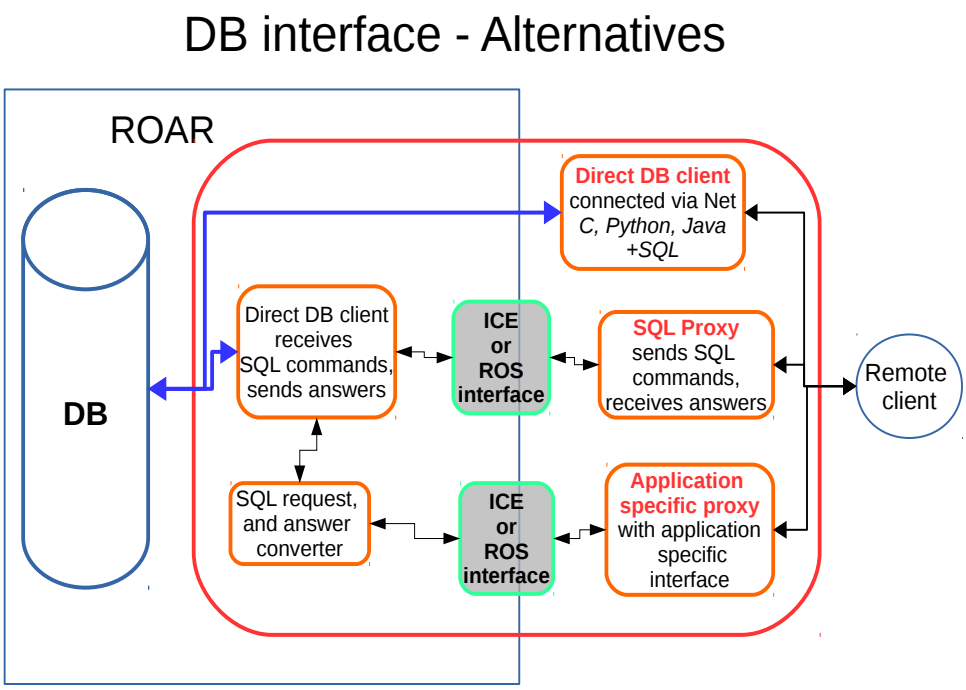


Figure 3: Alternatives of the DB interface; the remote Direct DB client interface could be the default

The three alternatives presented in Figure 3 can be summarized in the following points:

Direct DB client: The users are connected to the DB directly via a client module provided by PostgreSQL, or some other vendors. This module can be linked to almost all well known programming languages, C++, Java, Python, Perl etc. This client behaves as an SQL wrapper. The clients are connected to the DB via the standard internet protocol available in all known operating systems.

SQL Proxy: The users generate the SQL commands by their own program modules and sending the SQL commands as strings to the ROAR module, where those commands are channeled into a DB client for execution. In case of queries the ROAR can send back the results as

tables in a comma separated text file format(CVS). The communication can be built on the top of Zeroc-ICE or on the proper modules of the ROS.

Application specific Proxy: The users can send and receive ROS/ICE specific messages containing the data for insertion and updates, and the answers on queries. The network communication can follow the protocol of the SQL Proxy interface mentioned in the previous point.

It should be emphasized that this specific approach could be the source of several programming and communication errors and ought to be avoided as much as possible!

The most general approach is the Direct DB client which has the smallest overhead, and the broadest available support.

4 Learning infrastructure

The ROAR database system is built around those learning modules which have been introduced in earlier deliverables, the maximum margin based recommender- , and the homogeneity analysis systems, see in D3.1.1 and D3.1.2.: [2] and [3].

Here only the general structure of the learning modules is presented, since the robot system has no direct connection to the learner itself, it can see only the tables of the database which are completed by the learners in the background.

5 Database structure

Here some examples are shown about the possible correspondences between the terminology of the SQL and the functionality of the ROAR. Since the general DB framework allows high level of flexibility in representing the available knowledge therefore the contents of these examples can be easily adapted to new problems.

SQL database tables: The DB tables correspond to the potential relationships between the categories of the ROAR. Those categories are:

Actions,
Objects,
Action features,
Object Features.

The categories correspond to the DB fields (columns of the tables).

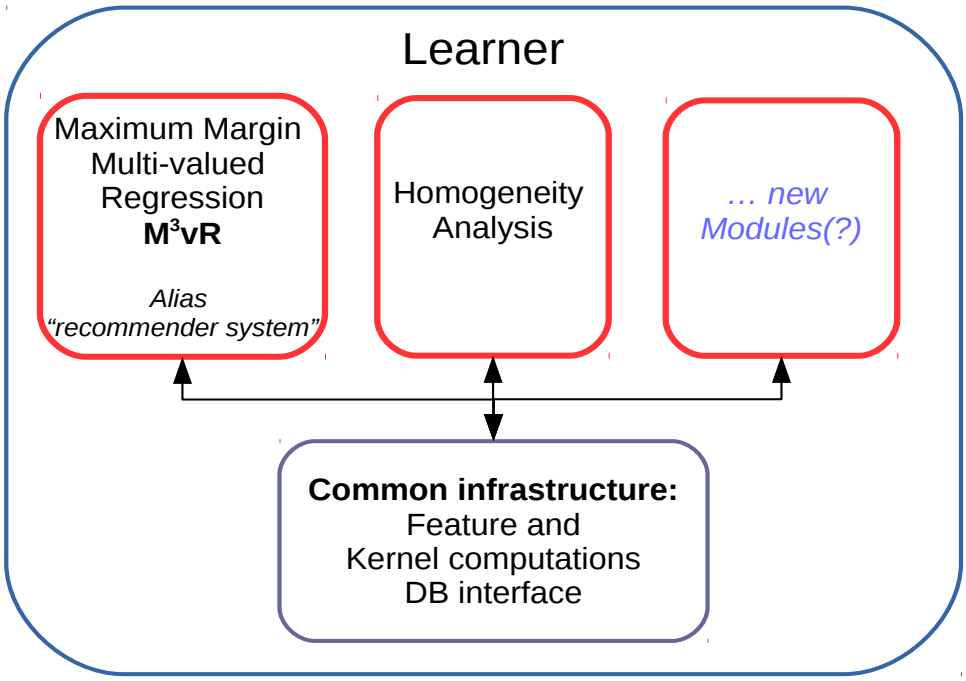


Figure 4: The internal structure of the learning module working behind the DB interface.

The possible relations, and in turn the SQL tables, are

Relations	Examples stored in that table
Action-Object	"cut" - "knife"
Object-Object,	"knife" - "cucumber"
Object-Object features,	"knife" - "elongated"
Action-Object features,	"cut" - "soft"
Action-Action features,	"cut" - "oscillatory trajectory"

The structure of the tables follow the general schema:

Category 1	Category 2	Value 1 of the relation, Value 2 of the relation, ...

Examples:

Table of the "Action-Object" relations

Category 1	Role	Category 2	Value 1	Value 2
Action	Object	Relation	Confidence	
"cut"	"with"	"knife"	"True"	0.95
"cut"	NULL	"cucumber"	"True"	0.95
"stir"	NULL	"juice"	"True"	0.9

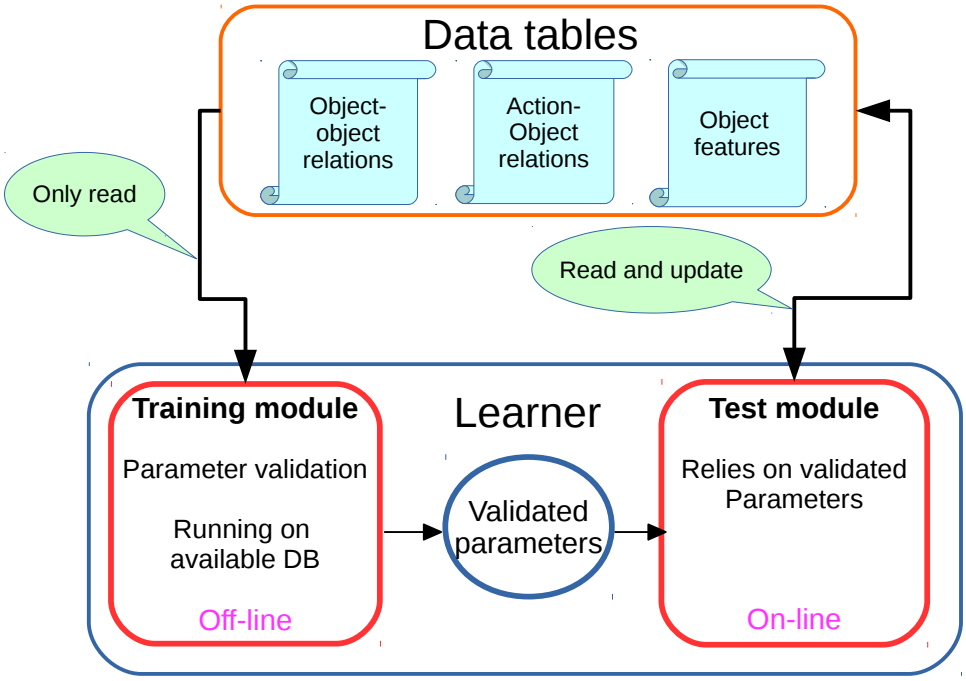


Figure 5: The connection between the main learning modules.

"NULL" corresponds to missing or omitted value.

Table of the "Object-Object feature" relations

Category 1	Category 2	Value 1	Value 2
Object	Object-feature	Relation	Confidence
"cucumber"	"cuttable"	"True"	0.95
"cucumber"	"soft"	"True"	0.9
"juice"	" liquid"	"True"	0.9

Here we can see that any category can occur several time in a table, and all the available values of the other category can be paired with that.

The DB system can be extended in any time with further tables (relations) and columns or fields(categories) without interfering with the earlier system, i.e. only the SQL commands created before the extension need not be modified.

6 Database interface

6.1 SQL commands

Here the most important commands are presented with simple examples. Very detailed tutorials, user guides with plenty of examples can be found on

these sites:

```
postgresql.org/docs/9.3/interactive/index.html
postgresql.org/docs/8.0/static/tutorial.html
tutorialspoint.com/postgresql/
```

In the SQL the "NULL" symbol represent the missing values. It is possible to load the "NULL" everywhere into the tables if it is not prohibited in the "CREATE TABLE" command by a "NOT NULL" statement. The "NULL" marks those fields for the ROAR which have to be be replaced with predicted values.

6.1.1 Create table

Basic format:

```
CREATE TABLE table_name
(
column_name1 data_type(size),
column_name2 data_type(size),
column_name3 data_type(size),
....
);
```

Example:

```
CREATE TABLE action_object
(
action      text, NOT NULL,
object      text,
affordance  boolean,
confidence  real,
....
);
```

6.1.2 Insert or update

Basic format:

```
INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)]
VALUES (value1, value2, value3,...valueN);
```

Example:

```
INSERT INTO object_object_feature (object, cutable, rollable,
main_color,edible)]
VALUES (carrot, true, NULL, orange, true );
```

Basic format:

```
UPDATE table_name
SET column1 = value1, column2 = value2...., columnN = valueN
WHERE [condition];
```

Example:

```
UPDATE object_object_feature
SET main_color = lighrred,
WHERE [object=carrot];
```

6.1.3 Query

```
SELECT object FROM action_object where action = cut
INTERSECT
SELECT object FROM object_object_feature where edible=true;
```

6.2 Meta information - DB structure

The internal structure of the DB, available tables, the field (column) names and their types of each table can be read similarly to all other data stored in the DB. These information can be gained from predefined tables of the DB. For example the names of the tables can be asked from the table "pg_class". Similarly the names of the fields (columns) are available from the table "pg_attribute" and there types from "pg_type".

6.3 Programming interfaces

Here the representation of the SQL "INSERT INTO table" command is shown for C++ and Python, the same example is similarly in other languages as well. See the implementation of other commands for example:

C++	tutorialspoint.com/postgresql/postgresql_c_cpp.htm http://pqxx.org/development/libpqxx/
Python	tutorialspoint.com/postgresql/postgresql_python.htm wiki.postgresql.org/wiki/Psycopg2_Tutorial

6.4 C++

```
#include <iostream>
#include <pqxx/pqxx>

using namespace std;
using namespace pqxx;
```

```
int main(int argc, char* argv[])
{
   char * sql;

   try{
      connection C("dbname=ROAR user=postgres password="pswrd" \
      hostaddr=127.0.0.1 port=8888");
      if (C.is_open()) {
         cout << "Database successfully opened: " << C.dbname() << endl;
      } else {
         cout << "Can't open database" << endl;
         return 1;
      }
      /* Create SQL statement */
      sql = "INSERT INTO object_object_feature ( object, cutable," \
            "rolable, main_color, edible ) " \
            "VALUES ('carrot' , true, NULL, 'orange' , true ); ";

      /* Create a transactional object. */
      work W(C);

      /* Execute SQL query */
      W.exec( sql );
      W.commit();
      cout << "Records successfully created" << endl;
      C.disconnect ();
   }catch (const std::exception &e){
      cerr << e.what() << std::endl;
      return 1;
   }

   return 0;
}
```

6.5 Python

```
#!/usr/bin/python

import psycopg2

conn = psycopg2.connect(database="testdb",  \
       user="postgres", password="pswrd",   \
       host="127.0.0.1", port="8888")
```

```
print("Database successfully opened")  ## Python Version 3

cur = conn.cursor()

cur.execute("INSERT INTO object_object_feature ( object, cutable, \
            rolable, main_color, edible )  \
            VALUES ('carrot' , true, NULL, 'orange' , true );")

conn.commit()
print "Records successfully created";
conn.close()
```

6.6 Basic workflow

A creation, insertion and a query sequence

```
CREATE TABLE object_action(
object  VARCHAR(50),
action  VARCHAR(50),
preposition  VARCHAR(50),
score  REAl
);

INSERT INTO object_action( object, action, preposition, score)
VALUES ('bowl_blue','pour','to',1.0)
);

SELECT object FROM object_action WHERE action='cut' and
preposition=NULL and MAX(score);
```

7 Setting up the database

This series of commands with some modification is based on the following source : [1]

INSTALL POSTGRESQL 1. Install latest PostgreSQL, the most recent version is 9.4, for example via the console:

sudo apt-get install postgresql libpq-dev

[1] http://stackoverflow.com/questions/8200917/postgresql-create-a-new-db-through-pgadmin-ui

2. PostgreSQL has a super user is called postgres. Change user to the PostgreSQL user:

   sudo su - postgres

3. Change password of postgres user:

   psql -d postgres -U postgres

   After entering into **psql** the following prompt is shown:

   ```
   postgres=#
   ```

   The prompt text refers to the name of database to which the user is connected. In the following sequence the lines not containing

   ```
   postgres=#
   ```

   are the messages of the system. To change the user password we might have this protocol:

   ```
   psql (9.1.3) Type "help" for help
   postgres=# alter user postgres with password 'YOUR_NEW_PASSWORD';
   ALTER ROLE
   postgres=# \q
   #logout postgres user
   logout
   ```

4. Restart the PostgreSQL server:

   sudo /etc/init.d/postgresql restart

   pgAdmin III: provide PostgreSQL administration and management tools. If pgAdminIII is not installed, the installation is easy:

   sudo apt-get install pgadmin3

ADD A NEW DATABASE Here we assume that the database is on the localhost.

psql -d postgres -h localhost -U postgres

```
postgres=#create database database_name;
```

where "database_name" can be "salad_scenario".

ADD A NEW USER Creating a new user with password for a database

psql -d postgres -h localhost -U postgres

```
postgres=# create user user_name with password 'user_password';
```

Grant all privileges to a user on the database salad_scenario:

```
postgres=# grant all privileges on database salad_scenario  to user_name;
postgres=# \q
```

Try to login as new user to test the user creation:

psql -d salad_scenario -h localhost -U user_name

The prompt text is similar to this:

```
salad_scenario=>
```

Then the user with granted privileges can add new tables to the database "salad_scenario", and inserting new data items by following the work-flow outline in Section 6.6.

SAVE(EXPORT) DATABASE INTO SQL FILE Saving the basic postgres template database into SQL we need the command:

pg_dump -U user_name -d database_name -h localhost > db_save.sql

or saving the database salad_scenario

pg_dump -U user_name -d salad_scenario -h localhost > salad_scenario.sql

In both cases the user with "user_name" has to have all granted privileges on the database saved, see in the point "ADD A NEW USER".

LOAD(IMPORT) DATABASE FROM SQL FILE This is the inverse operation of the pg_dump database exporting utility. First a new database has to be created and then the SQL dump file, see in the previous point, can be loaded into the new database.

pg_restore -U user_name -d salad_scenario_new -h localhost -f salad_scenario.sql

8 Connection to a DB server via pgadmin3 utility

ADD A SEVER Open pgAdminIII and add new localhost server. Go to menu File > Add Server

Set up pgAdmin III server instrumentation:

When connecting to a PostgreSQL database using pgAdmin you may receive an error letting you know that the server instrumentation is not installed.

Install postgresql-contrib package:

sudo apt-get install postgresql-contrib

Install adminpack extension:

sudo -u postgres psql

```
postgres=# CREATE EXTENSION "adminpack";
postgres=# \q
```

9 Sample DB

Here on the next page an example of the data tables is presented. This table is part of the database comprising the information extracted from web about the potential relations between objects and actions. The "preposition" field shows the type of those relations, and "score" field provides the estimated confidence on relations. In the table presented only those object-action pairs are enumerated which have higher estimated confidence than 0.95. The creation of web related object-action database is relying on the approach presented in [1].

object	action	preposition	score
knife	drop	from	1
spatula	drop	from	1
potato	cut	from	1
butter	get	from	1
yogurt	get	from	1
oven	get	from	1
wok	drain	from	1
fork	drop	from	1
cucumber	stir	in	1
zucchini	stir	in	1
banana	roll	in	1
knife	cut	in	1
artichoke	stir	in	1
broccoli	stir	in	1
lettuce	place	in	1
spinach	stir	in	1
canister	put	in	1
fork	put	in	0.9874
zucchini	stir	into	1
artichoke	put	into	1
carrot	stir	into	1
broccoli	stir	into	1
pepper	cut	into	1
spinach	stir	into	1

yogurt	stir	into	1
microwave	put	into	1
fridge	put	into	1
toaster	put	into	1
canister	put	into	1
fork	put	into	1
fridge	get		1
tomato	pour	on	1
pepper	cut	on	1
potato	put	on	1
pot	put	on	0.999
microwave	cook	on	1
fridge	put	on	1
toaster	put	on	1
kettle	put	on	0.9923
spatula	pour	onto	1
stove	put	onto	1
dish	put	onto	1
banana	put	with_by	1
cleaver	cut	with_by	1
artichoke	cut	with_by	1
carrot	mix	with_by	1
lettuce	put	with_by	1
oven	cook	with_by	1
canister	cut	with_by	1
container	get	with_by	1
dish	put	with_by	1

References

[1] Peter Kaiser, Mike Lewis, Ronald P. A. Petrick, Tamim Asfour, and Mark Steedman. Extracting common sense knowledge from text for robot planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2014)*, Hong Kong, China, 31 May– 7 June 2014.

[2] Sandor Szedmak. Learning object-action relations via knowledge propagation. Technical report, University of Innsbruck, 2012. Technical report.

[3] Hanchen Xiong, Sandor Szedmak, and Justus Piater. Homogeneity analysis for object-action relation reasoning in kitchen scenarios. In *2nd Workshop on Machine Learning for Interactive Systems, (Workshop at IJCAI)*, page 3744. 2013.

2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

Knowledge Propagation and Relation Learning for Predicting Action Effects

Sandor Szedmak, Emre Ugur and Justus Piater

Intelligent and Interactive Systems, Institute of Computer Science,
University of Innsbruck

Abstract— **Learning to predict the effects of actions applied to pairs of objects is a difficult task that requires learning complex relations with sparse, incomplete and noisy information. Our Knowledge Propagation approach propagates affordance predictions by exploiting similarities among object properties, action parameters and resulting effects. The knowledge is propagated in a graph where a missing edge, corresponding to an unknown interaction between two objects (nodes), is predicted via the superposition of all paths connecting those objects in the graph. The high complexity of affordance representation is addressed through the use of Maximum Margin Multi-Valued Regression (MMMVR), which scales well to complex problems of multiple layers. With increased diversity and size of object databases and the addition of other parametric combinatory actions, we expect to achieve complex systems that leverage learned structure for subsequent learning, achieving *structural bootstrapping* over lifelong development and learning.**

In this paper, we extend MMMVR for learning of paired-object affordances, i.e., for predicting the effects of actions applied to pairs of objects. In our experiments, we evaluated this method on a dataset composed of 83 objects and 83×83 interactions. We compared the prediction performance with standard classifiers that predict the effect category given the object pair's low-level features or single-object affordances. The experiments show that our proposed method achieves significantly higher prediction performance especially when supported with Active Learning.

I. Introduction

Learning object-object relations is a difficult task. The difficulty comes from two main sources. First, the structure of descriptions of particular objects is very complex, while those descriptions are generally incomplete. The descriptions are derived from several sources, and the corresponding feature spaces are high-, even infinite-dimensional. Some features possess intriguing internal structure, e.g. graphs, which require computationally intensive preprocessing. The second source of difficulties is the small number of experiments which can confirm our hypotheses about the relationships between paired objects and the corresponding action. The experiments might even provide contradicting outcomes; thus the reliability of our knowledge is limited.

A general framework to learn sparse incomplete relations between several data sources was introduced by Szedmak et al. [1]. This general framework has been applied to recommender systems [2], [3]. Recommender systems connect users to objects, e.g. books, movies etc., and have to tackle problems of very high level of sparsity, since most of the user-object pairs are missing. Very frequently less than one percent of pairs can be observed; therefore the others need to be predicted. We face a similar problem in learning object-object relations, where actions can only be tried on a very small number object pairs in real experiments. Therefore, predicting the outcome of an action executed on a pair of objects can borrow the approach applied on recommender systems.

The learning task in this paper is to predict the effect of an action that is applied to a pair of objects. In object-object relation learning we consider the case where large numbers of objects are available but the tried actions on pairs of these objects is small. In this case, if the structure of the space spanned by the objects is sufficiently rich and the feature specific properties between any two objects can be transformed into each other by exploiting the similarity among those objects, then this similarity can be used to propagate information in a network of object pairs to other interaction instances. In other words, object information can be propagated over the object-pair space. We will call this approach Knowledge Propagation in the rest of the paper.

In our paper, we compare three different approaches with the aim of predicting the effect of a *stacking* action. The first approach is a standard one where low level features such as shape and dimensions of the object pairs are used as input attributes for state-of-the-art classifiers to predict the effect categories. The second approach utilizes the same classification method [4], but instead of using low-level features, it uses pre-learned single-object affordance features that already include some invariance related to object-robot-environment dynamics. The third method uses Knowledge Propagation that assumes the existence and knowledge of object identifiers. This is indeed a strong assumption, but we discuss that object identifiers can be derived from object features and affordances, which is a challenge not in the scope of this paper.

In the context of robot affordance learning research, multi-object affordance learning has not been studied extensively with exceptions of [5] where 'tool objects' interact with other objects, and [6] where two-object relational interaction models were learned. Several recent studies focused on learning of expected sensory feedback to predict the outcome of future trials [7], [8] and to predict the discrete changes in the system's dynamics to guide manipulation actions [9]. Our

978-1-4799-6934-0/14/$31.00 ©2014 IEEE

focus, on the other hand, is on predicting the action outcomes using Knowledge Propagation that enables propagation of the predictions through exploiting the similarities among objects, in sparse and incomplete datasets.

A. Knowledge Propagation

The learning problem we are facing can be summarized in the following way. There is given a matrix whose rows and columns are indexed by labels of objects. The elements of the matrix express the outcome of the interactions between the object pairs. The matrix elements might contain not only simple numbers but complex, structured elements, categories of multi-class systems, vectors or graphs. In the concrete case applied in this paper, these elements are categories. The elements of the matrix are only partially given. For example, if we are given 1000 objects, then collecting the interactions between all possible object pairs will be infeasible in a real robot environment. Therefore the matrix is incomplete and even very sparse; hence the learning task is to learn a function which can be predict all the missing elements based on the available ones. Additionally we might be given feature vectors describing the objects as well, e.g. shape descriptors, which can be another source of exploitable information.

In learning the outcome of unknown interactions we can exploit the geometric structure of the feature space spanned by the known elements. Based on that structure, the objects can be connected by certain similarity measures, and along these connections the knowledge can be transferred from the outcome of the small number of known object pairs to those that have not been tried so far. One can imagine the underlying geometry as a graph with objects represented by nodes which are connected by edges, and the edges equipped with weights expressing the similarity of the incident objects. One can refer to a similar, semi-supervised learning based, model presented for example in [10].

The base learning method, described in Section II, can predict the missing elements of the matrix, and can provide confidences of prediction of each missing element. This method grew out of a combination of different maximum-margin based structured learning approaches. One group of those models build upon Markov Random Fields [11], [12], [13], [14]. These models can provide highly accurate predictions but at a very high cost of computation. The other group is derived directly from the Support Vector Machine, by preserving the same computational complexity but those models can predict complex output structures as well [15], [16]. In [16] the performance of models of both groups are directly compared in predicting hierarchical structures. The approach used in this paper is based on a synthesis of those mentioned above, and can tackle very large data sets containing millions of potential interactions [1], [2], [3].

II. Methods

In this section, we explain the methods used to learn paired-object affordances and the Active Learning approach that speeds up affordance learning. In order to learn the mapping from objects' features to the paired-object affordances, we will use SVM-like classifiers as summarized in Section II-B. On the other hand, in order to learn to predict paired-affordances given object identifiers, we will use KnowPropC that is detailed in Section II-C. The last subsection provides the algorithmic details of applying Active Learning in our framework.

A. Problem Description

The aim is to learn to predict the effect of actions that are applied to pair of objects. In other words, given objects the classifier learns to predict the effect category of an action. Objects are represented in three different ways:

- *Basic features* correspond to conventional, manually-designed features computed mostly from visual perception with no explicit link to robot's actions.
- *Affordance features* encode the list of affordances offered by single objects considering robot actions. For example, (pinch-graspable, not-power-graspable, front-rollable, side-pushable) is an affordance feature vector composed of categorical values. Affordance features are assumed to be learned and can be computed from basic features of the object.
- *Object ids* correspond to labels of the objects. Known object ids can be computed based on object feature similarity or can be given by a human expert.

In a standard learning approach where object features are input and effects are predicted, as object ids have no generalization power, they do not yield high accuracy. The Knowledge Propagation method detailed below can propagate the effect information across different object ids.

B. Maximum Margin Classifier (MMC)

This classifier is used to predict the effect of stack action given object features or given object affordances. For this purpose we use Maximum Margin Regression (MMR) as we showed that MMR can improve classification accuracy in multi-class learning problems [4]. MMR is realized by the same optimization problem of SVM but can produce vector-valued output. Fig. 1 shows the main differences between SVM and MMR. For more details, please refer to [1], [16].

C. Knowledge Propagation based relational Classifier (KnowPropC)

In this paper, we propose an extension to MMR (see formulation in Eqn. 6) to learn the effect category of interactions given the two object identifiers. This problem is not a pure supervised learning problem since one object might appear in interactions with several other objects, or we have no observation of the interaction between several object pairs, thus we face the problem of missing values, an incomplete database.

1) Description of the relational learner: The details of an earlier version of the learning model can be found for example in [1], [2] and [3]. Here we provide a summary of the model applied.

We are given two sets $\mathcal{B}$ and $\mathcal{U}$, and we can assume that they are finite. Let $\mathscr{R}$ be an arbitrary relation between $\mathcal{B}$ and

	Binary class learning	Vector label learning
	Support Vector Machine	Maximum Margin Regression
min	$\frac{1}{2}\underbrace{\boxed{\mathbf{w}'\mathbf{w}}}_{\|\mathbf{w}\|_2^2}+C\mathbf{1}'\boldsymbol{\xi}$	$\frac{1}{2}\underbrace{\boxed{\mathbf{tr}(\mathbf{W}'\mathbf{W})}}_{\|\mathbf{W}\|_F^2}+C\mathbf{1}'\boldsymbol{\xi}$
w.r.t.	$\boxed{\mathbf{w}:\mathcal{H}_\phi\to\mathbb{R},}$ normal vec. $\boxed{b\in\mathbb{R},}$ bias, $\boldsymbol{\xi}\in\mathbb{R}^m,$ error vector,	$\boxed{\mathbf{W}:\mathcal{H}_\phi\to\mathcal{H}_\psi,}$ linear op., $\boxed{\mathbf{b}\in\mathcal{H}_\psi,}$ translation(bias), $\boldsymbol{\xi}\in\mathbb{R}^m,$ error vector,
s.t.	$\boxed{y_i(\mathbf{w}'\boldsymbol{\phi}(\mathbf{x}_i)+b)}$ $\geq 1-\xi_i,$ $\boldsymbol{\xi}\geq\mathbf{0},\ i=1,\ldots,m,$	$\boxed{\langle\boldsymbol{\psi}(\mathbf{y}_i),\mathbf{W}\boldsymbol{\phi}(\mathbf{x}_i)+\mathbf{b}\rangle_{\mathcal{H}_\psi}}$ $\geq 1-\xi_i,$ $\boldsymbol{\xi}\geq\mathbf{0},\ i=1,\ldots,m.$

(1)

Fig. 1. Primal problems for maximum margin learning: Support Vector Machine for binary classification, and Maximum Margin Regression for general feature represented outputs. H_ϕ and H_ψ denote the input and output feature spaces.

$\mathcal{U}$ given by a subset $\mathcal{D}$ of the ordered pairs of the elements of $\mathcal{U}$ and $\mathcal{B}$. Assume that to any pair of $(b,u)\in\mathcal{D}$ there is given a certain collection of information describing how the items in that pair relate to each other. This information can be obtained by some experiments. For a pair (b,u) it is described by z_{bu} taken from a set of possible descriptions $\mathcal{Z}$.

The information characterizing the relation between the pairs might be given by a binary variable, e.g. (b,u) relate to each other or not, or by a real number, e.g. the probability of observing b given u. If the sets $\mathcal{B}$ and $\mathcal{U}$ consist of complex objects, e.g. $\mathcal{B}$ contains objects and $\mathcal{U}$ is a collection of action identifiers, then the relationship between a pair (b,u) can be described by the affordances where action u is applied on object b. In the current work where effects of paired-objects are learned for the stacking action, both $\mathcal{B}$ and $\mathcal{U}$ will correspond to objects, one being dropped and the other on the table. On the other hand, z_{bu} will refer to the effect of the stacking action applied on these objects.

The available sample of observations that can be used to capture the structure of the relation can be given by a set of triples (b,u,z_{bu}), a pair of objects and the description of the available information.

The proposed learning model is summarized in the following points:

- Given two finite sets $\mathcal{B}$ and $\mathcal{U}$.
- There is given a function $f:\mathcal{B}\times\mathcal{U}\to\mathcal{Z}$, where $\mathcal{Z}$ is a set whose elements are the descriptions of the available information connecting a pair $(b,u)\in\mathcal{B}\times\mathcal{U}$ and it is denoted by z_{bu}.
- Let the subset $\mathcal{D}\subseteq\mathcal{B}\times\mathcal{U}$ express a relation between $\mathcal{B}$ and $\mathcal{U}$. Let $f(b,u)=\varnothing$ if $(b,u)\notin\mathcal{D}$; the function f is only partially given on its domain.
- We can collect for all b the relating values of u, $\mathcal{D}_b=\{u|(b,u)\in\mathcal{D}\}$, and similarly for all u the relating b, $\mathcal{D}_u=\{b|(b,u)\in\mathcal{D}\}$. We might call them b- or u-related sections or projections of $\mathcal{D}$.
- $\phi_Z:\mathcal{Z}\to\mathcal{H}_z$ a feature mapping into the Hilbert space $\mathcal{H}_Z$.
- Feature vectors of the elements of $\mathcal{B}$ and $\mathcal{U}$ can be defined by the mappings

$$\phi_B:\times_{\text{card}(\mathcal{U})}\mathcal{H}_Z\to\mathcal{H}_B$$

and by

$$\phi_U:\times_{\text{card}(\mathcal{B})}\mathcal{H}_Z\to\mathcal{H}_U,$$

where $\mathcal{H}_B$ and $\mathcal{H}_U$ are Hilbert spaces.

2) Learning task: The learning task is given by a sample set of tuples consisting of three elements (b,u,z_{bu}). The sample might not contain references to all elements of the sets $\mathcal{B}$ and $\mathcal{U}$, and consequently the set $\mathcal{D}$ describing the relation is also partially given. Let $\mathcal{B}^{(o)}\subseteq\mathcal{B}$ and $\mathcal{U}^{(o)}\subseteq\mathcal{U}$ be the sets whose references are observed in the sample, and let $\mathcal{D}^{(o)}\subseteq(\mathcal{B}^{(0)}\times\mathcal{U}^{(o)})\cap\mathcal{D}$ be the set of ordered pairs (b,u) for which the corresponding information z_{bu} is available. Furthermore we have the projections $\mathcal{D}_b^{(o)}=\{u|(b,u)\in\mathcal{D}^{(o)}\}$ and $\mathcal{D}_b^{(o)}=\{u|(b,u)\in\mathcal{D}^{(o)}\}$ into the observed sets $\mathcal{U}^{(o)}$ and $\mathcal{B}^{(o)}$.

Now the task is to find the function $f:\mathcal{B}\times\mathcal{U}\to\mathcal{Z}$ based on the knowledge given by $\{z_{bu}|z_{bu}\in\mathcal{D}^{(o)}\}$.

3) Optimization problem, first approach: The aim of the learning problem can be rephrased as finding a multilinear function $F:\mathcal{H}_Z\times\mathcal{H}_B\times\mathcal{H}_U\to\mathbb{R}$, which has higher value if the feature vectors of two items b and u can better predict the feature vector of z_{bu}

$$\phi_Z(z_{bu})\sim\mathbf{W}(\phi_B(b)\otimes\phi_U(u)). \tag{2}$$

The function F as a multilinear function can therefore be expressed as

$$\begin{aligned}F(\phi_Z(z_{bu}),\phi_B(b),\phi_U(u))\\ =\langle\mathbf{W},\phi_Z(z_{bu})\otimes\phi_B(b)\otimes\phi_U(u)\rangle,\end{aligned} \tag{3}$$

where $\mathbf{W}$ is a tensor describing the function itself, and the variables are connected by the tensor product; see details for example in [17].

We can rewrite the function F as

$$\begin{aligned}F(\phi_Z(z_{bu}),\phi_B(b),\phi_U(u))\\ =\langle\mathbf{W},\phi_Z(z_{bu})\otimes\phi_B(b)\otimes\phi_U(u)\rangle\\ =\langle\phi_Z(z_{bu}),\mathbf{W}(\phi_B(b)\otimes\phi_U(u))\rangle,\end{aligned} \tag{4}$$

where $\mathbf{W}$ plays the role of a linear operator mapping the tensor product $(\phi_B(b)\otimes\phi_U(u))$ into the space of $\mathcal{H}_Z$ and then the inner product is computed between the image of that map $\mathbf{W}(\phi_B(b)\otimes\phi_U(u))$ and $\phi_Z(z_{bu})$. This inner product will have a higher value if the correlation between the image vector $\mathbf{W}(\phi_B(b)\otimes\phi_U(u))$ and $\phi_Z(z_{bu})$ is higher.

4) Poly-learning – learning via an ensemble of weakly-coupled learners: We can reformulate the optimization problem by bearing in mind two problems that can occur in a real application:

- Solving the problem when the cardinality of the observed tuples, i.e. $\text{card}(\mathcal{D}^{(o)})$, is high can require too much resources measured in memory and also in time.

For example in a recommender system, where books are offered to users, the number of users can grow up to millions or even more, and the number of books can be several tens of thousands as well, and the possible observed pairs of users and books can be more than a billion.

- In a large data set built upon several sources the distribution of the items could be a highly multimodal one, namely a mixture of plenty of different distributions.

To overcome these difficulties we introduce a learning model which decomposes the entire problem into weakly coupled subproblems. In this model the role of the object sets $\mathcal{B}$ and $\mathcal{U}$ are asymmetric. To every element of $\mathcal{B}^{(o)}$ a learner $\mathscr{L}_b$ is assigned. To connect those learners the following assumption is applied. A learner assigned to a b tries to predict the interaction between b and the set of $u \in \mathcal{D}_b^0$. If there are given b_1 and b_2 and the intersection $\mathcal{D}_{b_1}^0 \bigcap \mathcal{D}_{b_2}^0$ is not empty, then both learners have to predict the common u elements in the same way in the sense that the values of the loss functions of both learners on those elements have to be equal.

The learners are defined via the multilinear function $F_b : b \times \mathcal{D}_b^{(o)} \times \{z_{bu} | u \in \mathcal{D}_b^{(o)}\} \to \mathbb{R}$. Each of the multilinear functions is given by the linear operator $\mathbf{W}_i$ which plays the role of a similarity measure between its parameters.

The loss functions are defined by applying a version of the hinge loss:

$$L_b(b, u, z_{bu}) = \begin{cases} 0, \text{ if } F_b(b, u, z_{bu}) \geq 1, \\ \max_u(1 - F_b(b, u, z_{bu})), \text{ otherwise,} \end{cases}$$
$$\text{for all } u \in \mathcal{D}_b^{(o)}. \tag{5}$$

The optimization problem expressing the ideas introduced above can be stated as

$$\begin{array}{ll} \min & \frac{1}{2}\|\mathbf{W}\|_b^2 + C\sum_{u\in\mathcal{U}^{(o)}} \xi_u \\ \text{w.r.t.} & \mathbf{W}_b \in (\mathcal{H}_Z \otimes \mathcal{H}_B \otimes \mathcal{H}_U)^*, \text{ linear operator} \\ \text{s.t.} & \langle \phi_Z(z_{bu}), \mathbf{W}_b \phi_U(u) \rangle \geq 1 - \xi_u, \ b \in \mathcal{B}^{(o)}, \ u \in \mathcal{D}_b^{(o)}, \\ & \xi_u \geq 0, , \ u \in \mathcal{U}^{(o)}, \end{array} \tag{6}$$

Note that the roles of $\mathcal{B}$ and $\mathcal{U}$ can be swapped. In our case since $\mathcal{B} = \mathcal{U}$ this casting of the roles can be ignored.

After solving the joint optimization problem we have

$$\mathbf{W}_b = \textstyle\sum_{u\in\mathcal{D}_b} \alpha_{bu}(\phi_Z(z_{bu}) \otimes \phi_U(u)), \ b \in \mathcal{B}^0, \tag{7}$$

where α_{bu} are optimal Lagrangian multipliers, and $\otimes$ denotes the tensor product.

The prediction for a given $(\hat{b}, \hat{u})$ pair can be derived by

$$\begin{aligned} & z_{\hat{b}\hat{u}} = \max_t \phi_Z(z_t) \mathbf{W}_{\hat{b}} \phi_U(\hat{u}) \\ & = \max_t \phi_Z(z_t) \textstyle\sum_{u\in\mathcal{D}_{\hat{b}}} \alpha_{bu}(\phi_Z(z_{bu}) \otimes \phi_U(u)) \phi_U(\hat{u}) \\ & = \max_t \textstyle\sum_{u\in\mathcal{D}_{\hat{b}}} \alpha_{bu} \underbrace{\langle \phi_Z(z_t), \phi_Z(z_{bu}) \rangle}_{K(z_t, \phi_Z(z_t)} \underbrace{\langle \phi_U(u), \phi_U(\hat{u}) \rangle}_{K(u,\hat{u})}, \end{aligned} \tag{8}$$

where $K(z_t, \phi_Z(z_t)$ and $K(u, \hat{u})$ are kernel matrices expressing the inner product between the corresponding elements.

D. Active Learning

To implement knowledge propagation, an active leaning algorithm is applied. The main steps of this algorithm are:

1) Start on a small subset of all object pairs as training set of the learner.
2) Train the learner on the available outcomes.
3) Predict all untried elements of the matrix, and compute confidences for those predictions. This confidence is equal to the difference of the probability of the predicted category of the effect minus the probability assuming uniform distribution over all categories.
4) Choose that untried pair of objects for which the confidence is smallest, and check the interaction between this pair, and then include that into the training set.
5) Repeat the procedure from Step 2.

The confidences relate to information that we can gain if the predicted elements are included into the training set. If the confidence is high then that element does not provide much useful information since it is highly similar to those appearing in the training set. The low confidence points to the least similar elements, thus they can yield sufficient new information about the general structure of the interactions. Technically, the learner yields a distribution over the possible outcome categories. If the entropy of this distribution is high, categories are predicted with similar probabilities, and the confidence of the prediction is low.

This learning scheme can be straightforwardly extended by including different kinds of object features, e.g. shape descriptors. These features can be seen as new nodes of the graph connecting the objects. Those feature-related nodes can be connected by edges expressing the similarity between the object features. In this way the interaction between two objects can be predicted even if they have never appeared earlier in any experiments.

III. Experiments

In this section, we report our experimental results obtained from a database of 83 objects and their pairwise stacking interactions.

1) Interaction Dataset: We collected data from 83 objects (Fig. 2) by placing them on the table. In order to analyze our learning algorithms, we aimed to create an interaction database composed of (object, action, effect) triples with 4 actions that enable single- and multi-object manipulation. Poking actions, namely *s-poke*, *f-poke*, and *t-poke*, are designed to poke the object from its side, front and top, respectively. The *stack* action, on the other hand, is designed to release one object above another.

In order to collect this database, a robot was required to make $3 \times 83 + 1 \times (83 \times 83) = 7138$ interactions which is not feasible in the real world. Thus, we used a human expert who observed robot action executions on different sample objects and then filled the complete table in analogy with his observations. We used a 7 dof. Kuka arm and a 7 dof. Schunk gripper to let the expert develop an intuition of the robot-object-object dynamics. In cases where the effect is

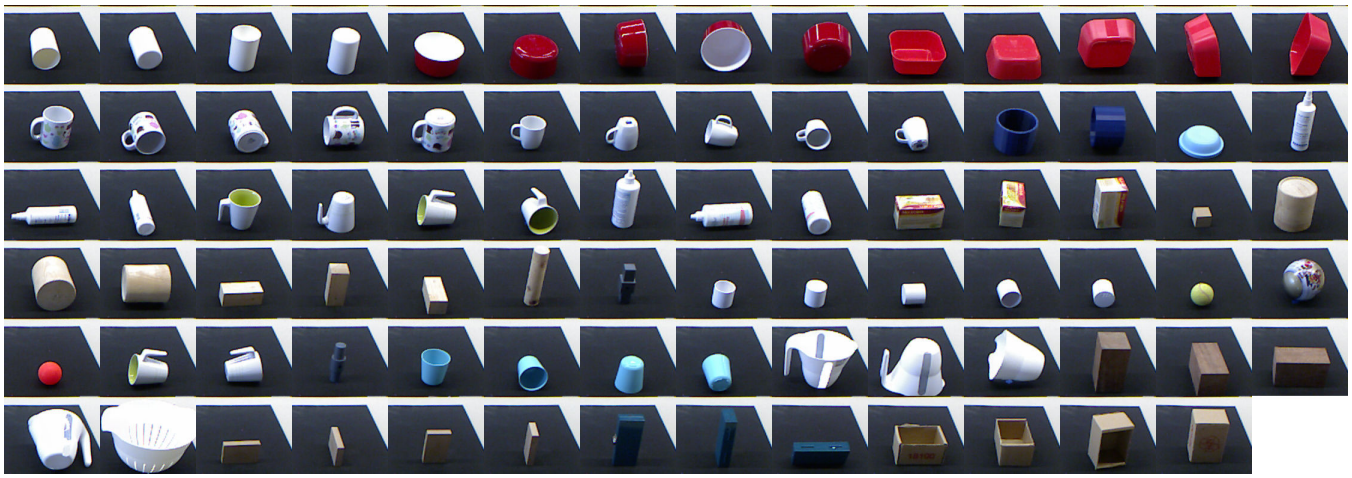


Fig. 2. Objects used in the experiments. Each object-orientation pair is assigned to a new object index in the experiments.

difficult to assess, the human emulated the robot's actions physically.

2) Action effects: The effect of stacking objects on top of each other depends on their relative size. For example, while 'inserted-in' effect is generated when a small box is stacked on a hollow cylinder, the 'piled-up' effect is observed when the box is larger than the opening on top of the cylinder. Using the objects, we marked the interaction results for each object pair for the stacking action. Different poking actions also generate different effects even on the same objects. For example, when poked from side, lying cylinders will roll away, boxes will be pushed, objects with holes in the poking direction will not be affected as finger would go into the hole without any interaction, and tall objects will topple over. The set of manually encoded actions and their effects are as follows:

- Actions: {side-poke, top-poke, front-poke, stack}
- Poking effects: {pushed, rolled, toppled, resisted, nothing}
- Stacking effects: {piled-up, inserted-in, covered, tumbled-over}

3) Object representation: The objects are segmented based on depth information of a Kinect sensor that is placed on the torso of the robot. In these experiments an object can represented by object id (assigned index), basic features or affordance features.

- *Object id* (*object-id*^{o}) is the index of the object.
- *Basic features (basic-feat^o)* are encoded in a continuous-valued vector composed of features relating to shape, size and local distance for object o:

$$\textit{basic-feat}^o = (shape^o, dim^o, dist^o)$$

Shape features are encoded as the distribution of local surface normal vectors from object surface[1]. Specifically histograms of normal vectors along each axis, 8 bins each, are computed to form $3 \times 18 = 54$ sized feature vector. dim encodes the object size in different axes. $dist$ features encode the distribution of the local distance of all pixels to the neighboring pixels. For this purpose, for each pixel we computed distances to the neighboring pixels along each 4 direction on Kinect's 2D depth image. For each direction, we created a histogram of 20 bins with bin size of $0.5cm$, obtaining a $4 \times 20 = 100$ sized vector for the $dist$.

- *Affordance features (afford-feat^o)* are encoded as the list of single-object action effects:

$$\textit{afford-feat}^o = (\varepsilon^o_{\textit{s-poke}}, \varepsilon^o_{\textit{p-poke}}, \varepsilon^o_{\textit{t-poke}})$$

where

$$\varepsilon^o \in \{\text{pushed, rolled, resisted, no-change}\}$$

refers to the effect categories of the corresponding poking action on object o. Although ε^o is manually coded for each object category by the human expert, we previously showed that this can be learned in previous stages of development and can be computed from basic features (*basic-feat*) [18].

4) Paired affordance learning: Our system learns to predict the effect of stacking actions given the descriptors of the two objects. This learning refers to building classifiers that predict a multi-class value of the stacking effect:

$$\varepsilon_{\textit{stack}} \in \{\text{piled, inserted-in, covered, tumbled-over}\}$$

Depending on the object description, different learning methods that are detailed in Section II will be used:

- When basic features or affordance features are used, learning corresponds to finding a mapping from these features to the effect class using Maximum Margin Classification (MMC):

$$\begin{aligned} \varepsilon^{o_1,o_2}_{\textit{stack}} &= \text{MMC}(\textit{basic-feat}^{o_1}, \textit{basic-feat}^{o_2}) \\ \varepsilon^{o_1,o_2}_{\textit{stack}} &= \text{MMC}(\textit{afford-feat}^{o_1}, \textit{afford-feat}^{o_2}) \end{aligned}$$

- When object ids are used, then our Knowledge Propagation based relational Classifier (KnowPropC) is employed. Note that this classifier learns and propagates

[1]Point Cloud Library normal estimation software is used to compute normal vectors.

the effects of the underlying graph in the object-object interaction table. We will add the additional meta-term *learned-pairwise-relations* to denote this characteristics:

$$\varepsilon_{stack}^{o_1,o_2} = \text{KnowPropC}^{\ \textit{learned-pairwise-relations}}(id^{o_1}, id^{o_2})$$

A. Action propagation results

In this section, we compare the performance of the classifiers that are trained with different object descriptors: basic-feature based MMC, affordance-feature based MMC, and object-id based KnowPropC. We evaluated the performance of each classifier type by systematically changing the number of training samples. Using the stacking interaction dataset composed of 83×83 samples of (object-pair, effect) triples, we created training datasets of increasing sizes. We trained a classifier with each training set, and tested the performance of the classifier using the remaining samples. The prediction results obtained from different classifier types using different training sets are provided in Fig. 3. Mean and variance of the prediction performance with 10-fold cross-validation are provided with the corresponding bars.

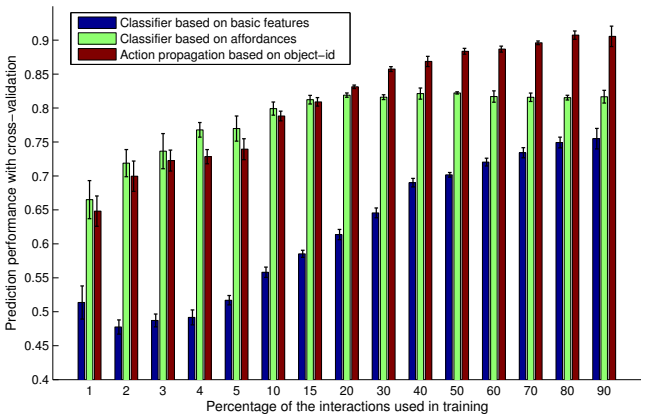


Fig. 3. The prediction performance of the classifiers based on basic features and affordances, and action propagation system based on object id. All predictors predict the effect of stack action in a database composed of 83 objects and 83×83 interactions in total.

As shown in the figure, even with very small training datasets (1% of the complete data), *object-id* based and *afford-feat* based classifiers perform significantly better than *basic-feat* based classifier. While the performance of *afford-feat* based classifiers levels off at training sets of around 15%, $basicf$ based classifier performance continually improves with increasing training size and becomes similar to *afford-feat* performance at the end. The performance of *object-id* based classifiers also increases continually, but at a higher level, and outperforms *afford-feat* performance at the end. We can interpret these results as follows:

- *basic-feat* and *afford-feat* based classifiers use the same classification method. Thus, their performance difference with small training sets should be related to the representation of these features. As *afford-feat* include the effects of single-object actions, they already include some properties related to object-robot-environment dynamics. As a result, this feature bootstraps the learning in the beginning. On the other hand, *basic-feat* is a high-dimensional, continuous-valued vector without explicit interaction properties of the objects. However, *afford-feat* can be computed based on *basic-feat* as we discussed before, so information encoded in *basic-feat* should be compatible with *afford-feat*. Therefore, as expected, with increased size of training data, *basic-feat* and *afford-feat* based classifier performances become similar.
- The *object-id* based classifier also has high performance with small datasets and its performance gets significantly higher compared to the feature based classifiers. This result demonstrates the success of the knowledge-propagation approach where instead of object properties alone, the classifier uses the connected nodes in the underlying graph of interactions to transfer knowledge from one object pair to another.
- It is difficult to compare *afford-feat* based and *object-id* based classifiers as they use different structures in training and predictions. We can assume that the similarity in their performance in initial steps of learning might be mere coincidence but the significantly higher performance of *object-id* based classifiers is a result of the powerful knowledge propagation mechanism.

B. Bootstrapping through active learning results

In this section, we evaluate the effect of active learning on prediction performance of *object-id* based classifier. Starting with a random set of few elements (1%), we increased the training sample set by selecting the next sample based on the criteria described in Section II-D. The prediction performances of the classifiers with active learning and with random sampling of training data are provided in Fig. 4. As shown, the performance of the classifier remains same until the underlying interaction network that is used for knowledge propagation is established. Then, after around 6% of the dataset, the performance boosts up and quickly surpasses that without active learning. The initial phase of slow learning is related to the exploration-exploitation dilemma, i.e. it can be viewed as preparing the underlying structure of KnowPropC for the active learner for bootstrapping in the later phases.

IV. Conclusion

In this study, we introduced a method that was successfully used in recommender systems to the robotics community and proposed to use this method in learning paired object affordances in robots with large datasets. We evaluated the results of such learning, which is based on knowledge propagation of the relations on the underlying (object-pair, effect) graph nodes. We compared the results with more conventional approaches, state-of-the-art classifiers that use low-level object features or previously learned object affordances as input attributes. With a focus on Knowledge Propagation in the current paper, we showed that the prediction performance of the proposed method is significantly higher compared to

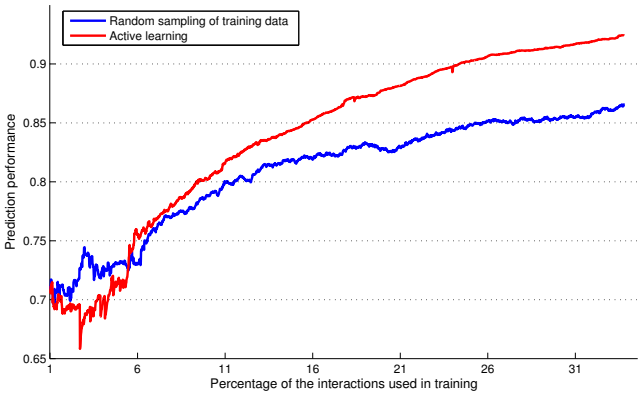


Fig. 4. The prediction performance of predictors that are trained with increasing number of training samples that is either randomly sampled from the dataset or selected according to active learning criteria. Both classifiers use randomly sampled initial training sets to achieve a connected underlying graph. As shown, the active learning speed and performance is superior compared to random sample selection.

the standard classifiers especially when active learning is applied.

The disadvantage of the proposed knowledge propagation method is the requirement of using object id's for learning. We discussed that this id can be provided by an expert or found by the robot by comparing its features against the objects in its database. The incorporation of novel objects into the object id dataset and prediction of action effects using the knowledge propagation algorithm is a challenging problem. In our future work, we will study creating object id's on the fly based on the feature and effect similarities and based on the prediction failures.

Collecting the interaction dataset by experts is very time consuming and probably introduces bias to the system. On the other hand, executing thousands of action to create this dataset is not realistic with real robots as well. In this paper, we showed that knowledge propagation based classification with active learning significantly speeds up learning. Yet, we need datasets to test and analyze our algorithms. In the future, we plan to use physics-based simulators for this purpose with the purpose of transferring the results to the real world.

Finally, we provide the object perception and interaction dataset used in this paper along with the open-source code of the learning method at
`https://iis.uibk.ac.at/public/szedmak/IROS2014-KnowProb/`.

ACKNOWLEDGEMENTS

This research was supported by European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

REFERENCES

[1] S. Szedmak, Y. Ni, and S. R. Gunn, "Maximum margin learning with incomplete data: Learning networks instead of tables," *Journal of Machine Learning Research, Proceedings*, vol. 11, Workshop on Applications of Pattern Analysis, pp. 96–102, 2010, jmlr.csail.mit.edu/proceedings/papers/v11/szedmak10a/szedmak10a.pdf.

[2] M. Ghazanfar, S. Szedmak, and A. Prugel-Bennett, "Incremental kernel mapping algorithms for scalable recommender systems," in *IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Special Session on Recommender Systems in e-Commerce (RSEC)*, 2011.

[3] M. Ghazanfar, A. Prugel-Bennett, and S. Szedmak, "Kernel mapping recommender system algorithms," *Information Sciences*, 2012.

[4] S. Szedmak, J. Shawe-Taylor, and E. Parado-Hernandez, "Learning via linear operators: Maximum margin regression," PASCAL Southampton, UK, Southampton, Tech. Rep., 2006, technical Report.

[5] J. Sinapov and A. Stoytchev, "Detecting the functional similarities between tools using a hierarchical representation of outcomes," in *Proceedings of the 7th IEEE International Conference on Development and Learning*. IEEE, Aug. 2008, pp. 91–96.

[6] M. B., P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt, "Learning relational affordance models for robots in multi-object manipulation tasks," in *Prof. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 4373–4378.

[7] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3828–3834.

[8] P. Pastor, M. Kalakrishnan, L. Righetti, and S. Schaal, "Towards associative skill memories," in *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*. IEEE, 2012, pp. 309–315.

[9] O. Kroemer, H. van Hoof, G. Neumann, and J. Peters, "Learning to predict phases of manipulation tasks as hidden states," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014.

[10] O. Chapelle, B. Schölkopf, and A. Z. Editors, *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2010.

[11] B. Taskar, C. Guestrin, and D. Koller, "Max-margin markov networks," in *NIPS 2003*, 2003.

[12] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *Journal of Machine Learning Research (JMLR)*, vol. 6(Sep), pp. 1453–1484, 2005.

[13] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor, "Kernel-based learning of hierarchical multilabel classification models," *Journal of Machine Learning Research*, vol. Special issue on Machine Learning and Large Scale Optimization, 2006.

[14] J. Rousu, C. Saunders, S. Szedmak, and J. ShaweTaylor, "Efficient algorithms for maxmargin structured classification," in *Predicting Structured Data*. MIT Press, 2007, pp. 105–129.

[15] S. Szedmak and Z. Hussain, "A universal machine learning optimization framework for arbitrary outputs," 2009, http://eprints.pascal-network.org.

[16] K. Astikainen, L. Holm, E. Pitkänen, S. Szedmak, and J. Rousu, "Towards structured output prediction of enzyme function," in *BMC Proceedings, 2(Suppl 4):S2*, 2008.

[17] J. Lee, *Introduction to Smooth Manifolds*, ser. Graduate Texts in Mathematics. Springer, 2003, vol. 218.

[18] E. Ugur, E. Oztop, and E. Sahin, "Goal emulation and planning in perceptual space using learned affordances," *Robotics and Autonomous Systems*, vol. 59, no. 7–8, pp. 580–595, 2011.

4th International Conference on
Development and Learning and on Epigenetic Robotics
October 13-16, 2014. Palazzo Ducale, Genoa, Italy

Emergent Structuring of Interdependent Affordance Learning Tasks

Emre Ugur and Justus Piater

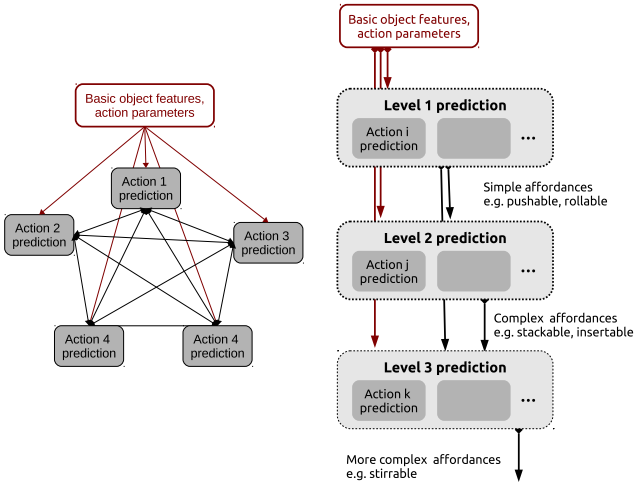
Intelligent and Interactive Systems, Institute of Computer Science,
University of Innsbruck

***Abstract*— In this paper, we study the learning mechanisms that facilitate autonomous discovery of an effective affordance prediction structure with multiple actions of different levels of complexity. A robot can benefit from a hierarchical structure where pre-learned basic affordances are used as inputs to bootstrap learning of complex affordances. In a developmental setting, links from basic affordances to the related complex affordances should be self-discovered by the robot, along with a suitable learning order. In order to discover the developmental order, we use Intrinsic Motivation approach that can guide the robot to explore the actions it should execute in order to maximize the learning progress. During this learning, the robot also discovers the structure by discovering and using the most distinctive object features for predicting affordances. We implemented our method in an online learning setup, and tested it in a real dataset that includes 83 objects and the discrete effects (such as pushed, rolled, inserted) created by three poke and one stack action. The results show that the hierarchical structure and the development order emerged from the learning dynamics that is guided by Intrinsic Motivation mechanisms and distinctive feature selection approach.**

I. Introduction

Studies with infant chimpanzees[1] and human infants[2] revealed that there is a dramatic increase in exploration and success of object-object combinatory actions at around 1.5 years of age while such actions were at a very low frequency before that period. This data suggests that the infants first develop basic skills and affordances that are precursors of combinatory manipulation actions. They also probably use the learned action grounded object properties in further development of complex action affordances.

In learning complex action affordances, i.e. affordances that are provided by pairs of objects, we proposed a learning framework where a developmental robotic system learns object affordances[1] in two-stages [3]. In the first stage, the robot learns predicting single-object affordances (such as pushability and rollability) by pushing single objects in different directions, and learning the relations between visual object features and the created discrete effects. In the second stage, these single-object affordance predictions, i.e. effects predicted to be obtained by the single-object actions, were used along with other object features to learn paired-object affordances in stacking task. In this context, we showed how complex affordance learning was bootstrapped by using pre-learned basic-affordances encoded as additional features. While such an approach was effective in efficient learning of increasingly more complex affordances, the development order and hierarchical prediction structure was manually designed based on the pre-defined complexity levels of actions and affordances. A truly developmental system, on the other hand, should be able to self-discover such a structure (see Fig. 1(b)), i.e. links from basic to related complex affordances, along with a suitable learning order.



(a) Flat prediction (b) Hierarchical prediction

Fig. 1. (a) shows a flat affordance learning structure, where the affordances are predicted based on low level object features, action parameters, and all other perceived affordances. (b) shows a simple hierarchical structure where simple affordance predictions can be used to detect complex affordances. This paper aims automatic discovery of such a hierarchical structure along with the corresponding development order of its components.

E. J. Gibson argued that learning affordances is neither the construction of representations from smaller pieces, nor the association of a response to a stimulus. Instead, she claimed, learning is "discovering distinctive features and invariant properties of things and events" [4]. Learning is not "enriching the input" but discovering the critical perceptual information in that input. We will argue that learning and prediction based on the most distinctive features not only provide perceptual economy (as in [5]), but can be used to autonomously determine the structure of the learning problem.

[1]In this study, the affordances provided by an object is defined as the list of discrete effects (e.g. pushed, rolled, inserted) predicted to be obtained by the discrete actions such as 'poke a single object' and 'stack a pair of objects'. Learning affordances refers to building a multi-category classifier for each action that predicts the effect of that action given continuous visual features and other predicted affordances of the object(s) involved. The discrete actions and discrete effects are assumed to be discovered before.

Affordance learning through exploration requires the embodied agent to deal with the high-dimensional nature of the sensorimotor development in an open-ended learning framework. Intrinsic Motivation approach, which can be regarded as a set of active learning mechanisms for developmental robots, enables efficient and effective learning in such environments by guiding the robot learning with intelligent exploration strategies[6]. Intrinsic Motivation (IM) approach in developmental robots [7] was inspired from curiosity based motivation mechanisms in human development, and has recently been effectively applied to cognitive robots where object knowledge is developed through self-exploration and social guidance [8]. This approach adaptively partitions agent's sensorimotor space into regions of exploration and guides the agent to select the regions that are in intermediate level of difficulty. This is achieved by maximizing reduction in prediction error, in other words by maximizing the learning progress. In this paper, we propose to use this approach to guide the robot to explore different affordances by adaptively selecting the actions to execute, and updating the models of the affordance predictions based on the results of these actions. Through IM approach, we aim to achieve a developmental progression similar to those of infants in learning simple-to-complex skills and affordances.

In summary, we study the mechanisms that enable autonomous structuring of affordance learning problem along with development order of its components. Our prediction system starts in a flat form as shown in Fig. 1(a) with no assumption on the relative complexity of actions and predictions. In each learning step, the robot actively selects the most "interesting" action to explore based on Intrinsic Motivation[9], and updates the prediction model of the corresponding action based on the observed effect. The robot also distinguishes "the most distinctive features" for prediction of each different affordance in order to "discover the information that specifies an affordance"[10] in training the prediction model. We expect these two mechanisms, namely (i) the Intrinsic Motivation based selection of actions to explore, and (ii) the use of the most distinctive features in affordance predictions, enable emergence of a hierarchical structure, similar to the one shown in Fig. 1(b) along with the corresponding developmental order of its components.

II. Active Learning of Affordances with Distinctive Features

This section gives the outline of the online learning algorithm. In our scenario, the robot needs to interact with the objects using its action repertoire in order to learn their affordances. Learning affordances corresponds to training a classifier for each action that predicts the effect of that action given object features. Thus, in each learning episode, the robot selects an action, executes this action on a number of objects, observes the effects created on these objects, and updates the predictor of the explored action with the acquired experience.

Algorithm 1 gives the online learning outline. At line 1, visual object features are computed for the objects observed in the environment. Next, predictors and their learning-progresses are initialized with an initial phase of random exploration which correspond to 10 interactions for each action. The first step of the main loop (line 4) is to select the next action to explore with the highest learning progress based on Intrinsic Motivation criteria (see Section II-C). Next, a number of objects are selected for exploration by this action (Section II-D). The selected action is executed on each selected object and the effects generated by these executions are observed (line 6). Based on the observed effects, the predictor of the executed action, along with its learning progress, is updated (lines 7 and 8). The most distinctive features used for predicting the effect of this action are also updated by finding the relevant features of the updated predictor at the same step (Section II-B). Finally, the effect predictions for all objects are updated. Note that we described the algorithm with single-object actions in order to provide a clear overall picture, thus omitted several details.

Algorithm 1 Active learning of affordances with distinctive features

```
1: compute object features
2: initialize predictors and affordance predictions
3: for each online learning time-step do
4:     select action based on Intrinsic Motivation
5:     select objects to explore
6:     execute the selected action on the objects and observe effects
7:     update the effect predictor of the selected action
8:     find the most distinctive features for the updated predictor
9:     update learning progress of the selected action
10:    update the effect predictions for all objects for the selected
       action
11: end for
```

A. Learning of affordances

Learning of affordances corresponds to learning the relations between objects, actions and effects [11]. In this study, object affordances are encoded as the list of effects achievable by executing different actions of the robot:

$$affordances^o = (\varepsilon^o_{a_1}, \varepsilon^o_{a_2}, ...)$$

where $\varepsilon^o_{a_1}$ is the discrete effect created on object o by action a_1.

Predicting the effect of each action is learned by executing the corresponding action on different objects. The resulting effect of one action depends on various features of objects, and is related to the other affordances the object provide. For example, stackability affordance can be related to rollability affordance and some other object features such as the object sizes. Here, object features corresponds to general-purpose basic ones computed mostly from visual perception with no explicit link to robot's actions. These may include standard features used in literature, related to size and shape properties of the objects. On the other hand, as we defined above, affordances encode object-action interaction dynamics for the available actions.

In order to learn affordances, and acquire the ability to predict action effect based on object features and affordances,

the following classifier ($Pred$) is trained for each action. Specifically, we use Support Vector Machine (SVM) classifiers with Radial Basis Function (RBF) kernel and optimized parameters to learn these predictors[12]. The multi-category classifier, after training, can predict the effect category given features and affordances as follows:

$$\varepsilon^o_{a_i} = Pred_{a_i}(\textit{features}^o, \textit{affordances}^o \backslash \varepsilon^o_{a_i})$$

Here, $\textit{affordances}\backslash\varepsilon^o_{a_i}$ denotes 'other affordances', i.e. the effect predictions of other classifiers. The general input/output structure is provided in Fig. 2.

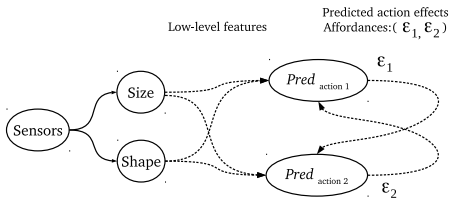


Fig. 2. Input/output links of the affordance predictors.

The recurrent connections in Fig. 2 might seem counter-intuitive in a non-dynamical system, where both of the predictors expect input from each other. This is achieved by keeping the predictions for all the objects in memory, and using the input values from the memory. Before training, the predictions for all objects are fixed to non-existing effect categories (-1). When a predictor is updated with exploration and learning, its predicted effects on all objects are updated in the memory as well. However these updated predictions are not propagated to other predictors immediately; they are used only by a new predictor that is being updated in the next time-step. In this way, we avoided potential instability issues of interdependent predictors.

Finally, this prediction mechanism can be generalized to actions that involve more than one object, by simply including all object features and affordances as the input attributes of the predictors. In this paper, we indeed use an action that involves two objects, where the predictor takes the following form:

$$\varepsilon^{(o_1,o_2)}_{a_i} = Pred_{a_i}(\textit{features}^{o_1}, \textit{features}^{o_2}, \textit{affordances}^{(o_1,o_2)} \backslash \varepsilon^{(o_1,o_2)}_{a_i}) \quad (1)$$

Paired-object affordances, i.e. affordances offered by the corresponding two objects, correspond to collection of the effects (expected to be) obtained by the execution of all available actions:

$$\textit{affordances}^{(o_1,o_2)} = (\varepsilon^{o_1}_{a_1}, \varepsilon^{o_1}_{a_2}, ...\varepsilon^{o_2}_{a_1}, \varepsilon^{o_2}_{a_2}, ..., \varepsilon^{(o_1,o_2)}_{a_i}, \varepsilon^{(o_1,o_2)}_{a_j})$$

B. Discovering the most distinctive features

The most distinctive features that specify an affordance correspond to the minimal set of inputs of the corresponding effect predictor with the maximum achievable prediction accuracy. We used *Sequentialfs* (sequential features selection) method to select these features. The Sequentialfs method generates near-optimal relevant feature sets in a way similar to the one used in Schemata Search[13]. Starting from an empty relevance feature set, it selects one feature and adds it to the feature set of previous iteration. At each iteration, a candidate feature set for each not-yet-selected feature is formed by adding the corresponding feature to the previous feature set. Then, the candidate feature sets are evaluated through 10-fold cross-validations on SVM classifiers that use these candidate feature sets. The best performing candidate set is then transferred to the next iteration. In the experiments, we empirically observed that not more than 10 features were necessary to achieve best accuracy, thus we limited the iteration number to 10. We also eliminated the ones that have no effect in accuracy increase, finalizing the most distinctive features for each trained predictor $Pred$.

C. Action selection with Intrinsic Motivation

Intrinsic Motivation, in its original formulation by Oudeyer et. al.[7], is used to adaptively partition agent's sensorimotor space into regions of exploration, and to guide the agent to select the regions that provide maximal learning progress. In our study, Intrinsic Motivation is used to guide our robot to select actions in order to maximize the learning progress, which is defined as the increase in prediction accuracy of the corresponding action.

The robot keeps learning progress of each action and in each time-step, it selects an action to explore based on the learning progress using ϵ-greedy strategy[14] where ϵ is set to 0.05. If an action (a_i) and a number of objects are selected for exploration at time-step t, the robot first computes the effects predicted to be achieved on these objects using $Pred_{a_i}$. Next, the action is executed on these objects and the generated effects are observed. The success of the robot in predicting the effects, denoted by $\gamma_{a_i}(t)$, is defined as the ratio of the correct predictions on objects explored by a_i, and is used to update the learning progress of the action.

The learning progress (LP) of action a_i is formally defined as the actual increase in the mean prediction accuracy of the predictor ($Pred_{a_i}$) of the corresponding action:

$$LP_{a_i}(t+1) = \overline{\gamma}_{a_i}(t+1) - \overline{\gamma}_{a_i}(t+1-\tau)$$

where $\overline{\gamma}_{a_i}(t+1)$ and $\overline{\gamma}_{a_i}(t+1-\tau)$ are defined as the current and previous mean prediction accuracies of the effect predictor, and τ is a time window, set to 2.

Here we define mean prediction accuracy by setting a smoothing parameter θ to 5:

$$\overline{\gamma}_{a_i}(t+1) = \frac{\sum_{j=0}^{\theta} \gamma_{a_i}(t+1-j)}{\theta+1}$$

This is only a local measure that approximates the real accuracy. We used this local accuracy measure in our online incremental learning setup as the robot cannot access to ground truth, i.e. it cannot know the effect categories of the objects without actually executing its actions on all of them in a real setting.

Fig. 3. A subset of the objects used in the experiments.

D. Active object selection

The aim is to select the next object set so that the diversity of the objects in the training set is maximized. For this purpose, the Euclidean distance between objects are computed using one of the three feature types randomly (as computing distance in the joint space of different types would be sensitive to relative weighting of the features). We select the next object from the set of possible objects (PossObjs) by maximizing the total distance from the next object to the already explored objects (ExpObjs) as follows:

$$nextObj = \underset{o_1 \in \text{PossObjs}}{\arg\max} \sum_{o_2 \in \text{ExpObjs}} dist_t(o_1, o_2)$$

where $dist_t(o_1, o_2)$ is the Euclidean distance between two objects in space t, which is sampled uniformly from the set of feature types {size, shape, distance}.

III. Experiment Setup

1) Interaction Dataset: We collected data from 83 objects (Fig. 3) by placing them on the table in front of our robot. Using these objects, we aimed to create an interaction database composed of (object, action, effect) tuples. In order to collect such a dataset, the robot, for example, was required to make $(83 \times 83) = 6889$ interactions for an action that involves two objects, which is not feasible in the real world. Thus, we used a human expert to fill-up the effect field of the complete table[2]

2) Actions: The robot is equipped with a number of manually coded actions that enable single and multi object manipulation. The robot can poke a single object from different sides using *front-poke*, *side-poke*, and *top-poke* actions. It can also stack one object on the other using *stack* action, where it grasps the first object, move it on top of the other one and release it.

[2]Guessing the effects of actions and filling up the table without any reference to robot's real world performance have the risk of creating a human-biased interaction dataset. In order to reduce this risk, we implemented poke and stack in our hand-arm robot system and let the expert observe the robot action executions on a number of different sample objects; and generalize his observations to other objects.

3) Action effects: The effect of stacking objects on top of each other depends on their relative size. For example, while 'inserted-in' effect is generated when a small box is stacked on a hollow cylinder, 'piled-up' effect is observed when the box is larger than the opening on top of the cylinder. Using the objects, we marked the interaction results for each object pair for stack action. Different poke actions also generate different effects even on the same objects. For example, when poked from side, lying cylinders will roll away, boxes will be pushed, objects with holes in poke direction will not be affected as finger would go through the hole without any interaction, and tall objects will topple down. The set of manually encoded actions and their effects are as follows

- Actions: {side-poke, top-poke, front-poke, stack}
- Poke-effects: {pushed, rolled, toppled, resisted, nothing}
- Stack-effects: {piled-up, inserted-in, covered, tumbled-over}

Note that all the effects can be differentiated based on changes in visual features of the objects, except for the effects 'resisted' and 'nothing', which require force readings from the end effector of the robot.

4) Object features: The objects are segmented based on depth information of Kinect sensor that is placed over the torso of the robot. Features are encoded in a continuous vector composed of shape, size and local distance related properties for object o:

$$features^o = (shape^o, dim^o, dist^o)$$

Shape features are encoded as the distribution of local surface normal vectors from object surface. Specifically histograms of normal vectors along each axis, 8 bins each, are computed to form $3 \times 8 = 24$ sized feature vector. dim encodes the object size in different axes. $dist$ features encode the distribution of the local distance of all pixels to the neighboring pixels. For this purpose, for each pixel we computed distances to the neighboring pixels along each 4 direction on Kinect's 2D depth image. For each direction, we created a histogram of 20 bins with bin size of $0.5cm$, obtaining a $4 \times 20 = 100$ sized vector for the $dist$.

IV. Experiment Results

Using the database of 83 objects, 4 actions, and their corresponding effects, we applied active learning of affordances with distinctive features method (Algorithm 1) to discover the structure and development order of the affordance learning system.

A. Discovered development order

This section provides the obtained development order of the affordance predictors. Recall that development order refers to maturation order of the action predictors, and can be analyzed by examining the order and frequency of actions, selected during each iteration of the online learning of the complete system (Algorithm 1, Step 7). The action selected for exploration in each iteration step is shown in Fig. 4. As shown, the less complex poke actions are learned first, and more complex stack action is learned later. As the effect of paired-object actions depend on the relations between

properties of two objects, stack is a more complicated action, difficult to learn. Prediction of stack action can also benefit from simple-affordances (as we will show in the next section). Thus, stack action is explored and learned automatically after all other simpler actions are explored. In the figure, the stack action is observed to be explored also in the beginning of the learning in a number of steps either because of momentarily increases in local accuracy or due to the ϵ-greedy strategy.

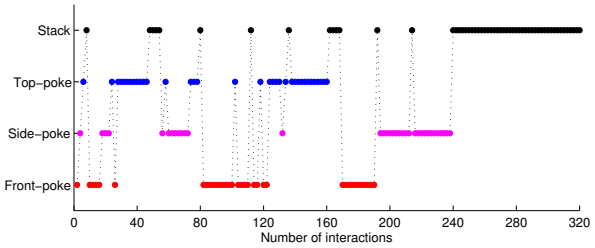


Fig. 4. The action selected for exploration and learning in each iteration of online learning of affordances. As shown, single-object affordances, i.e. prediction of effects of top-poke, side-poke and front-poke. are learned initially. As prediction of the effect of stack action requires learning of features (and probably affordances) of both objects, paired-object affordances are explored later.

We also plotted the local prediction accuracy γ evolution of each action in Fig. 5. The actions that are selected in the corresponding iteration step is illustrated with a mark along with its accuracy plot. As we defined in Section II-A, the accuracy of the predictors are computed using the small number of objects (denoted by objs in Algorithm 1; 4 objects in this experiment) explored in that iteration. This causes a jerky performance evolution as shown in the figure. We run the same algorithm with different initial conditions (initial objects), and observed that the exact shape of each accuracy plot and the exact order among single-object actions change. However, the tendency of learning single-object affordances first, and paired-object affordances later, remained consistent.

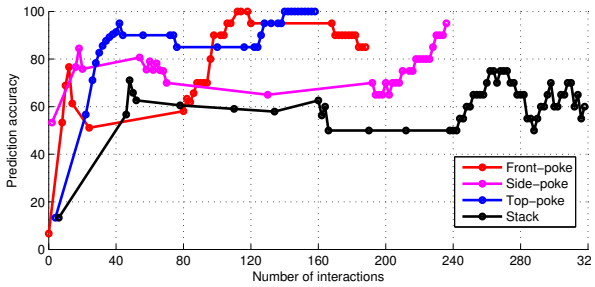


Fig. 5. The evolution of prediction accuracy of each predictor during online learning. In each time-step, one of the four predictors is being updated depending on the selected action, where this selection is illustrated by the marks on the plots.

B. Discovered affordance prediction structure

This section gives the results of the structure evolution of the affordance prediction system. Recall that the prediction

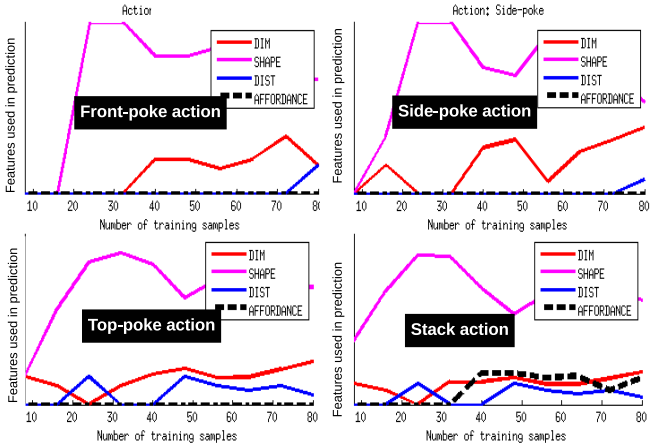


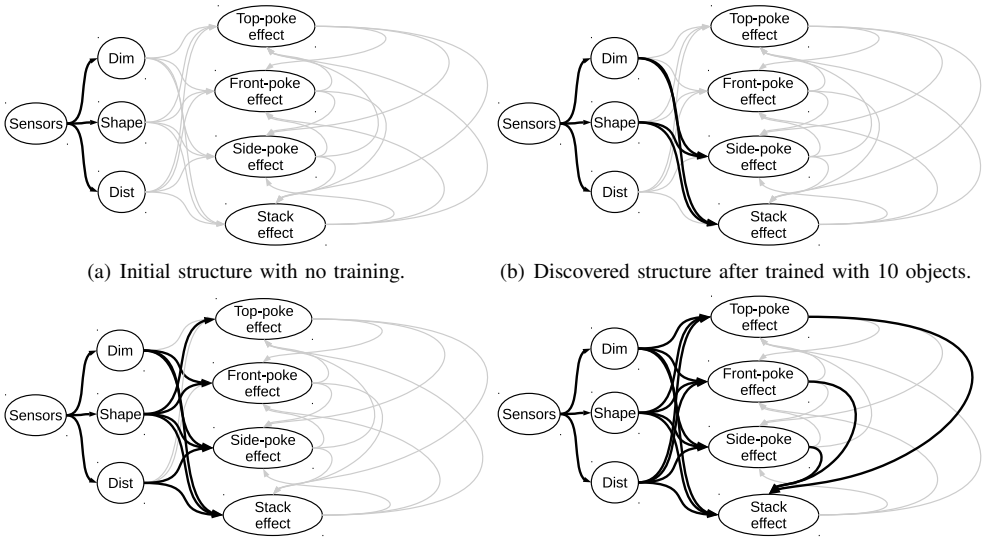
Fig. 6. Evolution of the distinctive features for prediction, where solid lines correspond distance, shape and dimension features, and dashed lines correspond to the predicted affordances. As shown, only effect prediction of the stack action uses affordances as distinctive features. Note that use of affordances for prediction starts after step 30, probably after effect predictors of poke actions (and their affordance corresponding predictions) have developed.

structure is defined over the most distinctive features that are discovered to be most effective in predicting affordances (Section II-B). The robot learns the affordances similar to the previous experiment, but in order to analyze the discovered structure independent of an action selection strategy, the next action in each iteration is selected randomly in this experiment. The ratio of the types of distinctive features used in prediction in different phases of the online learning are shown in Fig.6. Each plot in this figure corresponds to evolution of the used features and affordances for a different action. As shown in the plots, each low-level feature affects affordance predictions in different levels, and shape features are observed to be the first discovered distinctive features especially in the initial phases of development for all actions. However, more important in the context of this paper, affordances are observed not to be used in the initial phases, and are only found to be used in predicting effect of stack action, i.e. predicting stackability affordances. Note that stack predictor starts using single-object affordances after around 30 samples, probably because the single-object affordance prediction was not good enough before that time-point.

We also plotted the exact structure, i.e. features and affordances used by different effect predictors by highlighting the links in the prediction system in Fig.7. Different plots provide the structure in different iterations of learning. As shown, at the end, a hierarchical structure is formed as expected, where learned simple affordances are used in learning and prediction of more complex affordances.

V. Conclusion

In this paper, we studied how interdependent affordance learning tasks can be autonomously structured along with its developmental order. In an online learning framework, we showed that intrinsic motivation mechanism, which select



(a) Initial structure with no training.

(b) Discovered structure after trained with 10 objects.

(c) Discovered structure after trained with 20 objects.

(d) Discovered structure after trained with 80 objects.

Fig. 7. Evolution of the structure discovery of the affordance prediction system. Black lines correspond to the "distinctive" features that are used by affordance predictors. As shown, a hierarchy gradually emerges where at the end (d), single-object affordances are predicted based on only object features, and further used in predicting paired-object affordances.

the next action to explore, based on learning progress of the model of that action, can discover such a development order where paired-object affordance learning follows maturation of single-object affordances learning. Next, we showed that by using the most discriminative features for affordance prediction, the expected hierarchical structure emerged autonomously where the learning system discovered that predictions of the single-object affordances are connected to the paired-object affordances. We validated our approach in a real dataset composed of 83 objects and pairs of these objects along with the effects of three poke actions and one stack action. The results show that hierarchical structure and development order emerged from the learning dynamics that is guided by Intrinsic Motivation mechanisms and feature selection approach. In order to further verify our approach, we are currently working on realizing the learning cycle in the real robot with the aim of self-discovering the effect categories autonomously and analyzing the results with multiple independent trials.

In this paper, we assumed existence of discrete action primitives and effect categories. We safely made such simplifications and assumptions in the developmental setting of this paper, as we already showed that a set of basic primitive actions can be self-discovered through in interaction based on observed tactile profiles in [15], and effect categories can be autonomously found for different actions, such as rolled-out-of-table, pushed, no-change, grasped in [11].

ACKNOWLEDGEMENTS

This research was supported by European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

REFERENCES

[1] M. Hayashi and T. Matsuzawa, "Cognitive development in object manipulation by infant chimpanzees," *Animal Cognition*, vol. 6, pp. 225–233, 2003.

[2] M. Ikuzawa, *Development diagnostic tests for children*, 2000.

[3] E. Ugur, S. Szedmak, and J. Piater, "Bootstrapping paired-object affordance learning with learned single-affordance features," in *4th International Conference on Development and Learning and on Epigenetic Robotics*, Genoa, Italy, 2014.

[4] E. J. Gibson, "Perceptual learning in development: Some basic concepts," *Ecological Psychology*, vol. 12, no. 4, pp. 295–302, 2000.

[5] E. Ugur and E. Şahin, "Traversability: A case study for learning and perceiving affordances in robots," *Adaptive Behavior*, vol. 18, no. 3-4, 2010.

[6] M. Lopes, P.-Y. Oudeyer, *et al.*, "Guest editorial active learning and intrinsically motivated exploration in robots: Advances and challenges," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 2, pp. 65–69, 2010.

[7] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 265–286, 2007.

[8] S. Ivaldi, S. Nguyen, N. Lyubova, A. Droniou, V. Padois, D. Filliat, P.-Y. Oudeyer, and O. Sigaud, "Object learning through active exploration," *IEEE Transactions on Autonomous Mental Development*, pp. 56–72, 2013.

[9] P.-Y. Oudeyer and F. Kaplan, "What is intrinsic motivation? a typology of computational approaches," *Frontiers in neurorobotics*, vol. 1, pp. 1–6, 2007.

[10] A. Szokolszky, "An interview with Eleanor Gibson," *Ecological Psychology*, vol. 15, no. 4, pp. 271–281, 2003.

[11] E. Ugur, E. Oztop, and E. Sahin, "Goal emulation and planning in perceptual space using learned affordances," *Robotics and Autonomous Systems*, vol. 59, no. 7–8, pp. 580–595, 2011.

[12] C.-C. Chang and C.-J. Lin, "Libsvm : a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 27, pp. 1–27, 2011.

[13] A. W. Moore and M. S. Lee, "Efficient algorithms for minimizing cross validation error," in *Proceedings of the 11th International Conference on Machine Learning*, R. Greiner and D. Schuurmans, Eds. Morgan Kaufmann, 1994, pp. 190–198.

[14] R. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[15] E. Ugur, E. Sahin, and E. Oztop, "Self-discovery of motor primitives and learning grasp affordances," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 3260–3267.

Bottom-Up Learning of Object Categories, Action Effects and Logical Rules: From Continuous Manipulative Exploration to Symbolic Planning

Emre Ugur and Justus Piater

Abstract— **This work aims bottom-up and autonomous development of symbolic planning operators from continuous interaction experience of a manipulator robot that explores the environment using its action repertoire. The core idea is that the system discovers compact, descriptive, and predictable effect categories by grouping the interactions that produce similar continuous effects, and forms object categories in the form of list of generated discrete effects for different actions. After forming object categories, it learns the relations between continuous properties of the objects and the created effects, effectively building mechanisms that detect object categories from the observed properties of the objects.**

Development of the symbolic knowledge is achieved in two stages. In the first stage, the robot explores the environment by executing actions on single objects, forms effect and object categories, and learns to predict object/effect categories from visual properties of the objects. In the next stage, with further interactions that involve stack actions on pairs of objects, the system learns logical high-level rules that return stack effect category given the categories of the involved objects and discrete relations between them. Finally, these categories and rules are encoded in Planning Domain Definition Language (PDDL), enabling symbolic planning. We realized our method by learning the categories and rules in a physics-based simulator through exploration. The learned symbols and operators are further verified by generating and executing non-trivial symbolic plans in the real robot in tower building task.

I. INTRODUCTION

There exists a representational gap between continuous sensorimotor world of a robot and discrete symbols and operator used by advanced AI planning methods. Learning of the mapping between the sensorimotor readings and these symbols is one approach to bridge the gap and is a part of the so called symbol grounding problem [1]. The learning studies in this context typically assume that the planning symbols are pre-coded, and the relations between continuous sensorimotor readings and these pre-coded symbols are learned [2]. They generally define transition rules as actions linked by logical preconditions and effect predicates, and sensorimotor experience of the robot is used to associate the predicates of the transition rules. For example in [3], these preconditions were pre-defined binary functions of sensor readings, where the robot learned to combine the preconditions with effects and effects through human assistance. In [4], pre-defined high-level object and environment properties are used as the predicates of the transition rules. Similarly, [5] studied learning of predefined action effects using kernel perceptrons for STRIPS and ADL planning domains.

On the other hand, Sun argued that symbols "are not formed in isolation" and that "they are formed in relation to the experience of agents, through their perceptual/motor apparatuses, in their world and linked to their goals and actions" [6]. In this vein, symbol formation in a robot interacting with its world was studied in [7], where self-organizing maps were used to cluster low-level sensory data and to form perceptual states; and the planning was is performed by successively predicting the next perceptual states. Our previous work [8] also addresses planning in perceptual space, where the affordances that the robot learned during its interactions with the environment were used to develop multi-step plans in perceptual space with effect predictions and forward chaining in continuous domain. However, in these studies, the structures used for multi-step planning were still in continuous space, limiting the use of powerful AI planning techniques. [9], on the other hand, took a path in between, and used a teacher to learn grounded relational non-predefined symbols.

Mugan and Kuiper's recent work [10] is probably the closest one to our current study, in terms of the target research goal of acquiring discrete representations; while the methods differ significantly. They also proposed a method to learn qualitative representations of states and predictive models in a bottom-up manner by discretizing the continuous variables of the environment. Based on the predictive models that are learned using Dynamic Bayesian Networks (DBNs), Markov decision processes (MDP) framework is used to plan goal-oriented hand/arm control in the simulator.

The current study is part of a research effort where a robot system gradually develops skills and competencies in subsequent stages of development, similar to human infants. Previously we showed that a robot that is initialized with a basic reach-and-grasp movement capability can discover a set of action primitives [11], learn a library of affordances and associated predictors [8], and finally use these structures to bootstrap complex imitation with the help of a cooperative tutor [12]. The methods provided in the current study enable the robot to reach a higher-level of cognitive competence where it starts forming symbols and reasoning in symbolic level, and makes non-trivial plans in real world with the symbols developed in a bottom-up manner.

This research was supported by European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

University of Innsbruck Institute of Computer Science, IIS Innsbruck, Austria firstname.lastname@uibk.ac.at

II. METHODS

A. Representation

The environment and the generated effects are observed by the robot and originally represented in continuous variables, such as continuous object percept and continuous proprioceptive signals. With the methods provided, to use in symbolic planning, the robot learns categorical representation of the environment and the effects. The following notation will be used in our paper.

The robot is equipped with a number of manually coded **actions** that enable single and multi object manipulation. The robot can poke a single object from different sides using front-poke (*f-poke*), side-poke (*s-poke*), and top-poke (*t-poke*); grasp it using *pick*, and drop it at a given position using *release* actions. It can also stack one object on the other using *stack* action, which is a combination of *pick* and *release* actions, where the vertical aligned gripper grasps the object first, carries it on top of the other one and releases it.

Continuous object state is represented by $\boldsymbol{f}_o$ and includes the list of object *f*eatures obtained from the visual perception, including features related to the existence, shape and size of the object. The continuous proprioceptive state of the robot, which includes the gripper position, is represented by $\boldsymbol{f}_r$. Thus, the **continuous world state** with n objects is represented by

$$(\boldsymbol{f}_{o_1}, \boldsymbol{f}_{o_1}, ..\boldsymbol{f}_{o_n}, \boldsymbol{f}_r)$$

Continuous effect of action a on object o is represented by $\Delta(\boldsymbol{f}_o, \boldsymbol{f}_r)^a$ and corresponds to the changes in object features and proprioceptive readings.

Discrete effect (ε_o^a), on the other hand, is discovered by the robot during its development, and used for symbolic planning. Planner also uses **discrete object state**, which is a collection of discrete effects expected to be obtained by the available m actions:

$$S_o = (\varepsilon^{a_1}, \varepsilon^{a_2}, ..\varepsilon^{a_m})_o \tag{1}$$

The discrete object state can also be represented by **object category**, where each category is assigned an index that is uniquely represented in the vector. Object state and category will be used interchangeably through the document as they refer to the same concept.

Finally, **discrete world state** is represented by the list of discrete object states and discrete relations between pairs of these objects:

$$S = (S_{o_1}, S_{o_2}..S_{o_n}, R1_{o_1,o_2}, R2_{o_1,o_2}...)$$

where R corresponds a discrete relation between the corresponding pairs of objects, such as the size relation, 'below-bigger'.

J.J. Gibson coined the term **affordance** to refer to the action possibilities that objects offer to an organism in an environment [13]. In our formalism, object states include the list of action possibilities encoded with the corresponding expected effects. The robot will learn the relations between object features and action effects, effectively learning to perceive object categories given object features. In other words, it learns to perceive the possibilities provided by the objects with the explored set of actions, i.e. it learns object affordances. The learned affordances are used for planning as detailed in the next sections.

B. Development of symbols and prediction capability

The robot explores the environment using its action repertoire, and encodes its observations in terms of continuous world states and continuous effects. Because our aim is to build discrete symbols for the symbolic planning, the robot needs to form categories in these continuous spaces, i.e. learn encoding environment in discrete variables. For this purpose, the robot progressively learns discrete and abstract structures and rules to represent the world and do reasoning; and uses these high-level symbolic structures for symbolic planning at the end. The stages of development and use of knowledge representations are as follows:

1) Stage I: Learning of single-object affordances: In the first stage, the robot explores the environment with its actions that are executed on single objects such as poke, pick and release. It observes and stores the continuous effects generated during these interactions. After experiencing N^a number of interactions for any action a, unsupervised clustering methods are applied in continuous effect space

$$\{\varepsilon_i^a\}_{i=1}^I = \text{Cluster}(\{\Delta(\boldsymbol{f}_o, \boldsymbol{f}_r)_j^a\}_{j=1}^{N^a}, \tag{2}$$

in order to find I discrete effect categories for each action. Recall that in (1), we defined a discrete object state as a collection of the discrete effect categories. Thus, through this clustering process, the robot learns the object categories it encountered in terms of the action effects.

Next, the robot learns to relate these categories to the features of the objects in order to detect object categories without any action execution. For this purpose, it learns the mapping from continuous object features to the effect categories by training multi-category classifiers ($\text{Predictor}^{a_i}(\boldsymbol{f}_o)$) for each action a_i, with the following training data:

$$\{(\boldsymbol{f}_o, \boldsymbol{f}_r)_j, \varepsilon_j^a\}_{j=1}^{N^a} \tag{3}$$

After learning, given object features, the robot can predict the list of offered discrete effects, and perceive the discrete object state:

$$S_o = (\text{Predictor}^{a_1}(\boldsymbol{f}_o), ...\text{Predictor}^{a_m}(\boldsymbol{f}_o)) \tag{4}$$

For instance, if one object is predicted to be rolled with poke action, lifted with pick action, and tumbled off when it is released, it is categorized with (rolled, lifted, and stable) discrete variables. At the end of this phase, the robot acquires the ability to categorize the observed objects with the type of effects its action repertoire can generate.

2) Stage II.a: Discovering the effect categories of stack action: In this stage, the robot explores the environment with its actions that involve pairs of objects. *Stack* action is used for this purpose. Similar to Stage I, the robot executes this action on different pairs of objects, observes the continuous

effects generated, and finds representative and meaningful discrete effect categories created on the pair.

For effect category generation in this stage, we observed that a naive clustering in continuous effect space was not effective as the interacting objects can generate various effects due to the complex interactions between them. Thus the robot applies a number of further exploratory actions on the objects after stack action, and groups the effects generated during the execution of this sequence. For example, after stack action, it can grasp the object below, lift and rotate it; and can observe the generated effects on both object directly. If they move together then they are properly-inserted. In the current work, poke action is used as the exploratory action. After stacking, the robot pokes both of the objects one by one, observes the additional generated effects, and includes all these data to find a grouping in interactions:

$$\{\varepsilon_i^{\text{stack}}\}_{i=1}^{I} = \text{Cluster}(\{\Delta(\boldsymbol{f}_{o_1}, \boldsymbol{f}_{o_2}, \boldsymbol{f}_r)^{a_{\text{stack}}}, \Delta(\boldsymbol{f}_{o_1}, \boldsymbol{f}_{o_2}, \boldsymbol{f}_r)^{a_{\text{f-poke},o_1}}, \Delta(\boldsymbol{f}_{o_1}, \boldsymbol{f}_{o_2}, \boldsymbol{f}_r)^{a_{\text{f-poke},o_2}}\}) \quad (5)$$

where $a_{\text{f-poke},o_i}$ denotes front-poke action applied to the object pair:(o_1, o_2). The order of pairing is important, and the indexes will be replaced with the 'below' and 'above' keywords, in the rest of the text.

3) Stage II.b: Learning logical rules for stack affordances: In this stage, the robot builds rules for stack action. It already learned how to categorize single-objects with the list of the predicted action effects based on perceived object features. It uses the stack instances, and use the object categories and discretized relations between them (such as below-bigger or below-higher) to represent the states. Then, it attempts to obtain logical rules from the noisy and possibly inconsistent interactions, so that the rules can be used to predict stack effects. It uses decision tree rule learning methods to build a decision tree for a compact rule set using the following training data:

$$\{(S_{o_1}, S_{o_2}, R1_{o_1,o_2}, R2_{o_1,o_2}) \rightarrow \varepsilon^{a_{\text{stack}}}\} \quad (6)$$

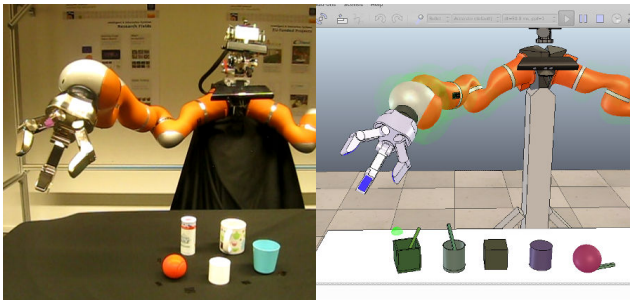
where $R1$ and $R2$ represent discrete relations between objects, and ε^{a_s} is the effect category obtained by the stack action, that was found in (5).

4) Stage III: Symbolic planning: In Stage III, the robot builds symbolic domain and problem descriptions based on the object states and the rules learned in the previous stages. This description, realized in STRIPS notation, includes all the predicates and actions. The predicates correspond to automatically discovered discrete object categories and relations, and actions correspond to the learned rules. The actions in PDDL (Planning Domain Definition Language) contain the following three fields:

- Action name: We used stack action in this work.
- Preconditions: The list of the predicates that should be valid in order to apply the action. This corresponds to the discrete object states and their discrete relations (left part of (6)) for each learned rule.
- Effects: The list of the predicates that change if the preconditions are satisfied and the action is executed. The predicted effect categories (right part of (6)) are provided in this field.

Therefore, domain description includes a separate action for each learned rule along with the corresponding preconditions and effects.

The system, after automatically constructing the domain description, can solve various problems depending on the state of the world and the goal. The initial state of the world, i.e. states of the objects and discrete relations between them, and the goal is defined in the problem description, in STRIPS notation as well. Given domain and problem descriptions, off-the-shelf symbolic planners are used to acquire the desired tasks. Please see Section III-E for the implementation details.



(a) Real robot (b) Simulation

Fig. 1: The experimental setup. KUKA LWR robot arm and Schunk gripper are used for manipulation and Kinect is used to extract object features including object's position and to compute object affordances. The simulation environment includes one and two objects during single-object and paired-object affordance learning experiments, respectively. Several objects are included while testing the planning capability for tower-building task in the real world.

III. EXPERIMENTS

In this section, we will present the results obtained through staged development of the robot and its planning performance. The single and paired affordances are learned through interactions and self-exploration in the simulated platform. After rules are learned, the plan generation and execution is verified in the real robot.

A. Robot Platform

Our experimental setup is composed of a KUKA Light Weight Robot (LWR) arm and a Schunk gripper for manipulation, a Kinect sensor for environment perception, and a number of objects that are placed on the table for exploration (Fig. 1a).

For environment perception, Kinect sensor placed over the torso is used. The robot's workspace consists of several objects and a table where the region of interest is defined as the volume over the table. The objects are then segmented by the Connected Component Labeling algorithm which differentiates object regions that are spatially separated by a preset threshold value (3 cm in the current implementation).

Features are encoded in a continuous vector composed of different properties for object o:

$$\boldsymbol{f}_o = (vis_o, pos_o, shape_o, dim_o, dist_o)$$

vis feature corresponds to object visibility which encodes the knowledge regarding the existence of the object. $shape$ features are encoded as the distribution of local surface normal vectors from object surface[1]. Specifically histograms of normal vectors along each axis, 8 bins each, are computed to form $3 \times 8 = 24$ sized feature vector. pos and dim correspond to the center and size of the object, respectively. $dist$ features encode the distribution of the local distance of all pixels to the neighboring pixels. For this purpose, for each pixel we computed distances to the neighboring pixels along each 4 direction on Kinect's 2D depth image. For each direction, we created a histogram of 20 bins with bin size of $0.5cm$, obtaining a $4 \times 20 = 100$ sized vector for the $dist$.

The robot can manipulate objects using its action repertoire that includes pick, release, f-poke, t-poke, s-poke, and stack actions. These actions are parameterized with target object position (pos_o), and the orientation of the gripper, depending on the action type. Once the target pose in task space is identified, the corresponding target joint angles are computed using inverse kinematics functions provided by the Kinematics and Dynamics Library (KDL)[2]. Finally, given the current and target joint angles, a smooth trajectory is computed by Reflexxes library [14], and this point-to-point movement is executed. The object is gripped from top using the built-in spherical grip of the Schunk hand.

For exploration and learning, the robot is required to make large number of interactions with the objects, which is time-consuming and risky in real world. For this purpose, we used the V-Rep[3] robot simulation platform (Fig. 1b) with Bullet physics engine. The complete simulated setup was built using the publicly available models of the arm and the gripper; and Robot Operating System (ROS) is used to communicate with both the real robot and the simulated one with exactly same interfaces. The objects used in training include boxes, cylinders, spheres, box walls, cylinder walls as shown in Fig. 1b, where small sticks are inserted into the hollow objects to illustrate the hollow objects.

B. Learned single-object affordances

In the first stage of learning, the simulated robot executes *poke*, *pick*, and *release* actions on single objects, monitors the environment, and stores the continuous object states along with the continuous effects generated by different actions. After collecting the interaction instances $\{\boldsymbol{f}_o, \boldsymbol{f}_r, \Delta(\boldsymbol{f}_o, \boldsymbol{f}_r)^{a_j}, a_j\}$, first it applies X-means clustering method in continuous effect space, $\{\Delta(\boldsymbol{f}_o, \boldsymbol{f}_r)^{a_j}\}$, and finds a number of effect categories for each different action. In our previous work [8], we already showed that meaningful and predictable effect categories cannot be obtained if clustering is performed in complex spaces such as size and shape features space. Therefore, we applied clustering in object position, object visibility and gripper position spaces and obtained the effect categories presented in Table I.

TABLE I: Discovered effect categories for single-object actions.

Action	Effect prototype	ε^{a_i}	Interpretation
Pick	-	0	Grasped
Release	change in object position	0	Tumbled
	no change	1	Stable
Front-poke	change in object visibility	0	Rolled off the table
	no change	1	Pushed
Side-poke	change in object visibility	0	Rolled off the table
	no change	1	Pushed
Top-poke	large change in gripper pos	0	Finger goes through
	small change in gripper pos	1	Finger obstructed

As all the objects used in the experiments were graspable, the effect of applying *pick* action were always same. For other actions, different distinguishable effects were created in different spaces. When these effects are enumerated, the following compact object category representation is obtained:

$$S_o \in \{\text{SOLID}, \text{ROLLABLE}, \text{HOLLOW}, \text{UNSTABLE}\}$$

where

SOLID = (0, 1, 1, 1, 1) ROLLABLE = (0, 1, 0, 0, 1)
HOLLOW = (0, 1, 1, 1, 0) UNSTABLE = (-, 0, -, -, -)

The effect categories for UNSTABLE objects are unknown, as they are generally thin objects that lie on the ground (after being released on the table and tumbled-off), and risky to interact with manipulation actions.

Finally, the mapping from object features to object categories, i.e. $\boldsymbol{f}_o \rightarrow S_o$, is learned using Support Vector Machine (SVM) with with Radial Basis Function (RBF) kernel and optimized parameters [15].

C. Discovered stack effect categories

In this stage, the simulated robot executes *stack* actions on pairs of objects, monitors the environment, and stores the continuous object states along with the continuous effects generated by different actions. As we described before, the effects do not only correspond to immediate effect of *stack* action, but is a collection of effects generated by the following exploratory *front-poke* actions applied both of the objects. Furthermore, as the effect space was very complex, only the changes in the visibility and position of the object are used as the effect features. The grouped instances of these features are provided in Table II. We interpreted these effect categories based on position and visibility changes, and provided labels for each effect category in the table. STACKED1 and STACKED2 effect categories correspond to SOLID-on-SOLID and ROLLABLE-on-SOLID stacking in general. TUMBLED1 and TUMBLED2 effect categories on the other hand both correspond to SOLID-on-ROLLABLE stacking interactions, where in the first case poking one object does not affect the other, and in the second case, the poked ROLLABLE object also pushes the SOLID object. The labels

[1]PCL normal estimation software is used to compute normal vectors.
[2]http://www.orocos.org/kdl
[3]http://coppeliarobotics.com

TABLE II: Discovered effect categories of stack action. Number of occurrences of the clusters, the corresponding changes in object features, and interpretation.

	After $stack_{o_1,o_2}$		After *f-poke*$_{o_1}$				After *f-poke*$_{o_2}$				Interpretation
#	Δvis_{o_1}	Δvis_{o_2}	Δpos_{o_1}	Δpos_{o_2}	Δvis_{o_1}	Δvis_{o_2}	Δpos_{o_1}	Δpos_{o_2}	Δvis_{o_1}	Δvis_{o_2}	
49	1	1	1	1	1	1	1	1	1	1	INSERTED
127	1	1	1	1	1	0	1	1	1	1	STACKED1
28	1	1	1	1	1	0	0	1	1	1	STACKED2
67	1	1	0	0	0	1	0	1	0	1	TUMBLED1
13	1	1	0	1	0	1	0	1	0	1	TUMBLED2
6	1	1	1	1	1	1	0	0	1	1	-
4	1	1	0	0	0	0	0	0	0	0	-
..	..	..	..	..	..	..	..	..	..	..	..

TABLE III: K-means clustering in effect category space.

ε^{stack}	μ_1	σ_1	μ_2	σ_2	Cat.
INSERTED	[0,0,0]	[0,0,-9]	[0, 2, 0]	[2, 4, 4]	0
STACKED1	[0,0,0]	[1,0,-2]	[1, 1, 0]	[4, 3, 5]	1
STACKED2	[0,0,0]	[1,-1,-1]	[0, 1, 0]	[4, 4, 2]	1
TUMBLED1	[2,0,0]	[0,-9,-12]	[3, 3, 0]	[7, 11, 4]	2
TUMBLED2	[-1,0,0]	[1,-5,-16]	[3, 2, 0]	[9, 13, 3]	2

of the effect categories will be used in the rest of the text (similar to use of object category labels) solely to ease the understandability of the text.

Note that only the most occurring effect categories are used in the rest. If we assume that the number of meaningful effect categories is 3, and apply further clustering (K-Means) using mean and variance of the object position changes for each effect category, a more 'intuitive' categorization can be obtained at the end as shown in Table III. As seen in the table, STACKED1 and STACKED2 are assigned to a single category; and TUMBLED1 and TUMBLED2 are assigned to another category. While these results with a hierarchical clustering give some insights about the complexity of the problem, and the necessity for human intervention to obtain the most meaningful clusters, we are not going to use these clusters, and let the system continue learning bottom-up in a completely unsupervised way.

D. Rule Learning

Based on the discovered object categories (Table II) and the discovered effect categories (Table III), now the robot can represent the interactions with discrete variables. Here, to represent discrete world state, we use discovered object categories, and size and height relations. The set of interactions the robot observed is encoded as follows:

$$\{(S_{o_1}, S_{o_2}, \text{Rel-Width}_{o_1,o_2}, \text{Rel-Height}_{o_1,o_2}), (\varepsilon^{stack})\}$$

where

$$S_o = \{\text{SOLID, ROLLABLE, HOLLOW, tumbled}\}$$
$$\text{Rel-Width} = \{\text{below-bigger, is-same, below-smaller}\}$$
$$\text{Rel-Height} = \{\text{below-higher, is-same, below-shorter}\}$$
$$\varepsilon^{stack} = \{\text{INSERTED, STACKED1, STACKED2, TUMBLED1, TUMBLED2}\}$$

A number of sample interaction instances obtained during experiments and encoded in the discovered discrete structures are shown below, where the first and second objects corresponds to the object below and above during stacking.

'HOLLOW','HOLLOW','below-smaller','below-shorter','STACKED1'
'HOLLOW','HOLLOW','below-bigger','below-shorter','INSERTED'
'HOLLOW','SOLID','same-width','below-higher','STACKED1'
'SOLID','HOLLOW','below-smaller','below-shorter','STACKED1'
'SOLID','ROLLABLE','below-smaller','same-height','STACKED2'
'ROLLABLE','SOLID','below-bigger','same-height','TUMBLED1'

	Below = HOLLOW
	— Rel-Width = below-smaller
01	— — Above = HOLLOW: STACKED1
02	— — Above = SOLID: STACKED1
03	— — Above = ROLLABLE: INSERTED
04	— — Above = UNSTABLE: INSERTED
	— Rel-Width = same-width
05	— — Above = HOLLOW: STACKED1
06	— — Above = SOLID: STACKED1
07	— — Above = ROLLABLE: INSERTED
08	— — Above = UNSTABLE: INSERTED
09	— Rel-Width = below-bigger: INSERTED
	Below = SOLID
10	— Above = HOLLOW: STACKED1
11	— Above = SOLID: STACKED1
12	— Above = ROLLABLE: STACKED2
13	— Above = UNSTABLE: STACKED1
14	Below = ROLLABLE: TUMBLED1
15	Below = UNSTABLE: STACKED1

Fig. 2: Results of decision tree learning

C4.5 decision tree learning with pruning is used to find a compact representation of rules with Weka software package [16]. The obtained decision tree is presented in Fig. 2. As shown, if the object below is a HOLLOW one, then depending on the size of the object above, the effects are different. For example, if the object below is bigger, then independent of any other factor, the resulting effect is always INSERTED. Otherwise, depending on the category of the above object and its relative width, it can get INSERTED in or stacked on the HOLLOW object. If the object below is SOLID, then, again depending on the object above, different types of stacked effects are expected to be generated. Note that in both STACKED1 and STACKED2, the above object stacks on the below object, but the STACKED2 case is more UNSTABLE (as detected and learned by the successive exploratory poke actions). Finally according to the decision tree, if the below object is ROLLABLE or UNSTABLE, independent of the object above, the effect would be tumbled or stacked, respectively. The tumbled effect in response to a stack action on ROLLABLE objects was expected. However, the stacked effect in response

```
(define (domain stack)
(:requirements :strips)
(:predicates
(hollow ?x)                (below-smaller ?x ?y)
(solid ?x)                 (below-bigger ?x ?y)
(rollable ?x)              (same-size ?x ?y)
(unstable ?x)              (below-shorter ?x ?y)
(pickloc ?x)               (below-higher ?x ?y)
(stackloc ?x)              (same-height ?x ?y)
(instack ?x))
```

Fig. 3: Predicates that are used in domain definition.

to stack action on UNSTABLE objects is not intuitive, and needs more elaborate analysis of the decision tree learner. Note that the rule learner was able to find out that the relative height of the objects do not have any significant influence on the generated effects in the simulated interactions; thus height relation is not included in any learned rule.

E. Planning

Based on the rules learned in the previous section, the system automatically constructs domain definition in STRIPS notation. We already mentioned that the domain is described by a number of predicates (discovered object and effect categories, and discrete relations between objects in our case), and a number of actions along with their preconditions and effects. Here, the stack action is included into the domain file, with the preconditions and effects given in Fig. 2. The predicates used in planning are given in Fig. 3. `(pickloc ?x)` and `(stackloc ?x)` predicates are set to true if the objects are at pick-up and stacked locations, respectively. `(instack ?x)` predicate is set to true if the effect category is in the following set: {INSERTED, STACKED1, STACKED2}.

Fig. 4 gives sample action descriptions generated from rules 01, 03, and 14, which have effects of TUMBLED1, INSERTED and STACKED1, respectively. We also added predicates that represent the height of the stack ('H') and the number of the objects ('S') in the stack. As the increment operator is not supported in STRIPS notation, a separate action is automatically generated for each level of height and stack increment.

While INSERTED does not affect the height of the stack significantly, STACKED1 and STACKED2 do increase the height, as shown by the mean position changes of the objects in Table III. Thus, 'H' is increased in rule (01), but not in rule (03). Similarly, the number of objects is increased with STACKED1 and INSERTED effects, not with the TUMBLED1effect; thus there is no change in 'S' in TUMBLED1 effect and the corresponding rule no 14. Finally, in this study we assumed that the category of the tower/stack is determined by the latest included element. For example, if a ROLLABLE object is added to the stack with SOLID or HOLLOW object on top, the next category of the stack is set as ROLLABLE. We plan to relax this constraint in our next work, by learning a rule that predicts the next 'category' of the stack, depending on the categories and relations of the pairs of objects being stacked. In this way, the system should learn that the category of the stack should be kept

```
(:action stack ;;; rule no: 01
:parameters (?Below ?Above)
:precondition (and (pickloc ?Above)
(stackloc ?Below) (hollow ?Below) (H0) (S0)
(below-smaller ?Below ?Above) (hollow ?Above) )
:effect
(not (pickloc ?Above)) (not (H0)) (H1) (not (S0)) (S1)
(instack ?Above) (stackloc ?Above)
(not (stackloc ?Below)))
```

```
(:action stack ;;; rule no: 03
:parameters (?Below ?Above)
:precondition (and (pickloc ?Above)
(stackloc ?Below) (hollow ?Below) (S0)
(below-smaller ?Below ?Above) (rollable ?Above) )
:effect
(not (pickloc ?Above)) (not (S0)) (S1)
(instack ?Above) (stackloc ?Above)
(not (stackloc ?Below)))
```

```
(:action stack ;;; rule no: 14
:parameters (?Below ?ABOVE)
:precondition (and (pickloc ?Above)
(stackloc ?Below) (rollable ?Below))
:effect
(not (pickloc ?ABOVE)))
```

Fig. 4: A number of automatically generated actions in domain definition

```
(define (problem simple-1)
(:domain stack)
(:objects o1 o2 table)
(:init (stackloc table)
(solid table)
(pickloc o1) (pickloc o2)
(hollow o1) (solid o2)
(below-bigger o1 o2)
(H0) (S0))
(:goal (and (S2) (H2))))
```

```
PLAN:
1 (stack table o2)
2 (stack o2 o1)
```

Fig. 5: Left: Sample domain and problem descriptions that include initial world state and the goal. Initial world contains one HOLLOW and one SOLID object, and an empty stack `(S0)`, with 0 height, `(H0)`. The goal is to build a stack of height 2 `(H2)` with 2 objects `(S2)`. Right: The generated plan first stacks SOLID object on the empty table, then stacks the HOLLOW object on top of SOLID, further increasing the height of the tower.

HOLLOW, if a very small ROLLABLE object is dropped-on a big HOLLOW object.

The goal/problem is defined with STRIPS notation as well. When we define the initial world state, we always assume that the objects are placed to a location/object with `(stackloc table)` predicate where a is a stackable object (SOLID table). Fig. 5 shows a sample world with two objects, that were categorized as HOLLOW and SOLID. The goal is to obtain a predicate with `(H2)`, which means to obtain a stack of height 2. In order to achieve this goal, the system plans to stack HOLLOW object on top of the SOLID object. But with the same objects, if the goal is set to have two objects in a stack of height 1, `(S2) (H1)`, then the plan would be different, stacking SOLID object on the bigger HOLLOW one:

```
1 (stack table o1) 2 (stack o1 o2)
```

Note that, if the height of the stack is not important,

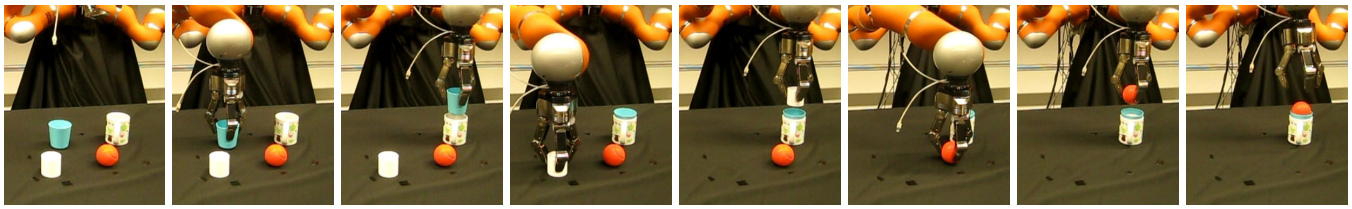


Fig. 6: Tower building experiment. The goal is to build a tower of `(S4)` and `(H1)`, i.e. a tower with 4 objects of height 1. The plan was first stacking the HOLLOW objects on top of each other, starting from the bigger ones; and finally stacking the ball at the end. As the objects are predicted to be inserted into each other, the height of the tower is correctly predicted to be kept fixed.

the tower-building goal can also be defined by using the `(instack o)` predicate. For plan generation, we use an off-the-shelf planning software, namely Blackbox[4], which transforms the facts and operators defined in STRIPS notation into propositional satisfiability (SAT) problem and solves the problem it with randomized systematic solvers [17].

F. Real World Experiments

In this section, our system is partially verified by the real robot experiments, where the effect category prediction (learned in the real world), and the rules (learned in the simulator as described above) are used to build and execute plans in tower building task. Given a number of objects, the task of the robot is to build compact towers with all objects included. Such a task is useful in transporting objects together, to ensure the stability by keeping the stack compact. Thus, the robot needs to plan a sequence of stack actions, and ensure that all objects are stacked with minimal final height of a single tower.

Given objects, the robot first computes the categories using the classifiers learned, then encodes world state in terms of these categories and the categorical relations among the objects. Finally, it runs the planner, setting 'S' predicate to the number of objects, and 'H' predicate to the minimum number (1) initially. In case, no plan is found, the constraint is relaxed, i.e. the goal 'H' predicate is increased by one in a loop, until a plan is found. After the plan is constructed, the robot selects the next objects to stack according to the plan and sequentially executes the stack actions. As the planned effect categories are grounded, the robot can also monitor the plan execution, and check whether the observed effect categories are in accordance with the expected ones; but this feature is not implemented in the current system.

In case study 1, three cups and one ball is presented to the robot as shown in Fig. 6. The robot first detects that they belong to 'HOLLOW' and 'ROLLABLE' categories, and encodes the category information along with the relative size and height relations in the initial world state as follows:

```
(define (problem tower-real-1)
(:domain stack) (:objects o0 o1 o2 o3 table)
(:init (stackloc table) (solid table) (S0) (H0)
(pickloc o0) (pickloc o1) (pickloc o2) (pickloc o3)
(hollow o0) (hollow o1) (rollable o2) (hollow o3)
;;; width and height relations
(:goal (and (S4) (H1))))
```

Above, the objects correspond to the white-cup, the green-cup, the ball, and the bigger coffee mug, respectively. The robot generated a plan where the HOLLOW objects are stacked on top of each other in decreasing order of width first, and the ball is stacked as the final action. The actions were successfully executed, and a compact and complete tower is built as shown in the snapshots of Fig. 6.

In the next case study, a SOLID object, which is a cylindrical shaped salt container, is included to the scene as shown in Fig 6, left-most snapshot. The robot is asked to make a plan with the same goal:

```
(define (problem tower-simple-2)
(:domain stack) (:objects o0 o1 o2 o3 o4 table)
(:init (stackloc table) (solid table) (H0) (S0)
(rollable o0) (hollow o1) (solid o2) (hollow o3)
(hollow o4) (pickloc o0) (pickloc o1) (pickloc o2)
;;; relations here
(:goal (and (S5) (H1))))
```

A plan was not generated with these constraints, thus the constraints were relaxed by incrementing the goal height of the tower: `small (:goal (and (S5) (H2)))`. The plan generated for this goal is as follows:

```
1 (stack table o3)
2 (stack o3 o4)
3 (stack o4 o2)
4 (stack o2 o1)
5 (stack o1 o0)
```

where the largest two HOLLOW objects and the salt-container are 'inserted' in each other first; then the white-cup is 'stacked', increasing the size; and finally the ball is 'inserted'. This plan was executed two times, leading to successful and unsuccessful results at the end, as shown in the final snapshots of Fig. 7. One can notice that the height of the tower is more than `(H2)`, i.e. more than height of 2 objects, because, contrary to the action predictions, the 'inserted' tall salt container and the 'inserted' ball increased the height considerably. A more compact tower could have been obtained by inserting all cups into each other and

[4]www.cs.rochester.edu/ kautz/satplan/blackbox/

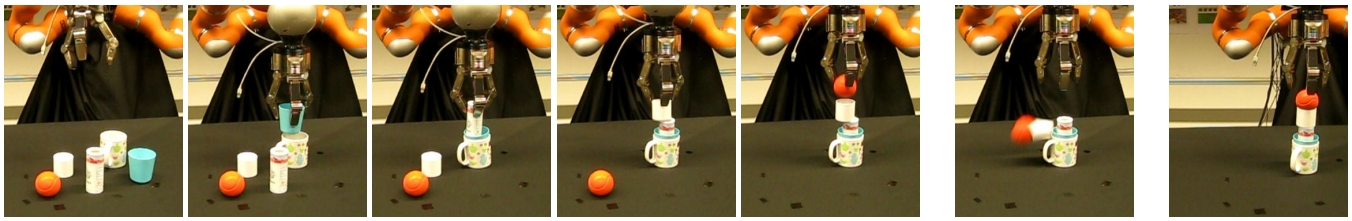


Fig. 7: Tower building experiment with an additional SOLID object. The goal is to build a tower of (S5) and (H2). The generated plan first stacks two HOLLOW objects on top of each other, starting from the biggest one; adding salt-container and small cup next, and finally stacking the ball at the end. The execution of the same plan fails in one execution and succeeds in the other one due to the noise in perception and actuation.

inserting salt-container into the inserted cups. Also, note that while the action with 'inserted' effect reliably works as planned, the action with 'stacked' effect is not reliable in the real world. This brings the necessity to use probabilistic planners and re-learning in the real world while building the towers, instead of learning only from paired stacks.

IV. Conclusion

In this study, we developed a system that forms symbols and operators in the continuous sensorimotor experience of the robot through self-exploration. These formed structures were used to generate symbolic plans in the robot in a sample tower building task. While our study provides a proof-of-the-concept realization of the proposed system, through analysis of the learned symbols and operators, and execution of plans in the real world; realizing a learning robotic system with the long-term goal of developing high-level cognitive skills stands as a big challenge.

The most immediate extension of this research is to incorporate probabilistic techniques in discovering symbols, learning operators and generating plans. Particularly, we plan to construct probabilistic rules from robot's noisy interactions based on category prediction confidences, and use planners that can deal with incomplete and noisy information (e.g. Planning with Knowledge and Sensing planning system [18]). During our experiments, we observed that, in addition to noise in sensing and perception, failures in plan execution were partially due to the simple encoding of the currently used motion primitives and the object perception that only relies on vision. Therefore, our system should incorporate vision and force guided closed-loop motion representation frameworks such as Dynamic Motor Primitives [19] that can also be used as exploratory actions to acquire other properties of objects, such as mass and material properties [20].

Finally, we provide the data and source code used in this study, along with the robot videos in `http://emreugur.net/icra2015/`.

References

[1] S. Harnad, "The symbol grounding problem," *Physica D*, vol. 42, no. 1-2, pp. 335–346, 1990.

[2] V. Klingspor, K. Morik, and A. D. Rieger, "Learning concepts from sensor data of a mobile robot," *Machine Learning*, vol. 23, no. 2-3, pp. 305–332, 1996.

[3] F. Wörgötter, A. Agostini, N. Krüger, N. Shylo, and B. Porr, "Cognitive agents: A procedural perspective relying on the predictability of Object-Action-Complexes OACs," *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 420–432, Apr. 2009.

[4] R. Petrick, D. Kraft, K. Mourao, N. Pugeault, N. Krüger, and M. Steedman, "Representation and integration: Combining robot control, high-level planning, and action learning," in *Proceedings of the 6th international cognitive robotics workshop*, 2008, pp. 32–41.

[5] K. Mourao, R. P. Petrick, and M. Steedman, "Using kernel perceptrons to learn action effects for planning," in *Int. Conf. on Cognitive Systems (CogSys 2008)*, 2008, pp. 45–50.

[6] R. Sun, "Symbol grounding: A new look at an old idea," *Philosophical Psychology*, vol. 13, no. 149–172, 2000.

[7] J. Pisokas and U. Nehmzow, "Experiments in subsymbolic action planning with mobile robots," in *Adaptive Agents and Multi-Agent Systems II, Lecture Notes in AI*. Springer, 2005, pp. 80–87.

[8] E. Ugur, E. Oztop, and E. Sahin, "Goal emulation and planning in perceptual space using learned affordances," *Robotics and Autonomous Systems*, vol. 59, no. 7–8, pp. 580–595, 2011.

[9] J. Kulick, M. Toussaint, T. Lang, and M. Lopes, "Active learning for teaching a robot grounded relational symbols," in *Proc. 23rd Int. Joint. Conf. AI*. AAAI Press, 2013, pp. 1451–1457.

[10] J. Mugan and B. Kuipers, "Autonomous learning of high-level states and actions in continuous environments," *Autonomous Mental Development, IEEE Transactions on*, vol. 4, no. 1, pp. 70–86, 2012.

[11] E. Ugur, E. Sahin, and E. Oztop, "Self-discovery of motor primitives and learning grasp affordances," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 3260–3267.

[12] E. Ugur, Y. Nagai, and E. Oztop, "Parental scaffolding as a bootstrapping mechanism for learning grasp affordances and imitation skills," *Robotica*, 2014, in press.

[13] J. J. Gibson, *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, 1986.

[14] T. Kroger, "Opening the door to new sensor-based robot applications – the reflexxes motion libraries," in *ICRA*, 2011, pp. 1–4.

[15] C.-C. Chang and C.-J. Lin, "Libsvm : a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 27, pp. 1–27, 2011.

[16] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[17] H. Kautz and B. Selman, "Unifying sat-based and graph-based planning," in *Proc. IJCAI-99*, 1999.

[18] R. P. A. Petrick and F. Bacchus, "Extending the knowledge-based approach to planning with incomplete information and sensing," in *Proceedings of the Int. Conference on Automated Planning and Scheduling (ICAPS)*. AAAI Press, 2004, pp. 2–11.

[19] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *Int. Conf. on Intelligent Robots and Systems (IROS)*.

[20] S. Takamuku, G. Gomez, K. Hosoda, and R. Pfeifer, "Haptic discrimination of material properties by a robotic hand," in *Int. Conf. on Development and Learning ICDL*, 2007, pp. 1–6.

4th International Conference on
Development and Learning and on Epigenetic Robotics
October 13-16, 2014. Palazzo Ducale, Genoa, Italy

Bootstrapping paired-object affordance learning with learned single-affordance features

Emre Ugur, Sandor Szedmak, and Justus Piater

Intelligent and Interactive Systems, Institute of Computer Science,
University of Innsbruck

***Abstract*— The aim of this paper is to propose a system where complex affordance learning is bootstrapped through using pre-learned basic-affordances as additional inputs of the complex affordance predictors or as cues in selecting the next objects to explore during learning. In the first stage, the robot learns affordances in the form of developing classifiers that predict effect categories given object features for different discrete actions applicable to single objects. These predictions are later added to robot's feature set as higher-level *affordance features*. In the second stage, the robot learns more complex multi-object affordances using object and affordance features. We first applied our idea in an artificial interaction database which includes discrete actions, several manually coded object categories, and actions effects. Finally, we validated our bootstrapping approach in a real robot with poke and stack actions. We expected to obtain higher performance with *affordance-features* especially in small training datasets as the object-robot-environment dynamics should have already been partially learned and encoded in affordances. The experiment results showed that complex affordance learning significantly speeds up with predictors that are bootstrapped with *affordance-features* compared to predictors that use low-level features such as shape descriptors. We also showed that by actively selecting the next objects and by increasing the diversity of the training set using a distance measure based on learned single-object affordances, the effect of bootstrapping can be further increased.**

I. Introduction

This study is part of a research effort where a robot system gradually develops skills and competencies in subsequent stages of development, similar to human infants. In our previous work, we showed that similar to human infants who learn a set of actions by the age of 7 months such as grasp, hit and drop [1], a robot could also self-discover a number of behavior primitives such as push, grasp and release by interacting with objects using its crude 'reach' action and grasp reflex, and observing the changes in its tactile perception [2]. Next, we showed that similar to infants who learn object dynamics after 7-9 month of age, our robot could learn affordances in an unsupervised way by first discovering the effect categories it could generate in the environment, and then by learning the mapping from the object features to the effect categories. After learning, the robot was shown make plans to achieve desired goals, emulate end states of demonstrated actions, monitor the plan execution and take corrective actions using the perceptual structures employed or discovered during learning. Finally, we showed that more complex actions that involve multiple objects (such as bring object 1 over object 2) can be taught to the robot through imitation using the structures developed in the previous stages with mechanisms inspired from parental scaffolding and motionese [**?**]. In the current study, we assume that a number of actions (such as push and stack) and effect categories (such as rolled and pushed), which were discovered in the previous stages as summarized above, are transferred to the next stage where complex affordances such as stackability are learned. We study how this complex affordance learning can be bootstrapped by use of learned simple affordances as (i) additional inputs in prediction, and (ii) in active selection of objects to explore next in an active learning setting.

One hallmark feature of bootstrapped learning is that learning problems stack in the sense that higher-level learners use as input attributes concepts produced by lower-level learners. These higher-level attributes should allow faster learning than if the higher-level concepts had to be learned from the lower-level attributes alone. The aim of this paper is to propose a learning system where a developmental robotic system benefits from bootstrapping where learned simpler structures (affordances) that encode robot's interaction dynamics with the world are used in learning of complex affordances. In detail, our robot learns the affordances of single objects and uses these affordances as additional features in the next stages of development where paired-object affordances are discovered. The use of learned similarities in the form of affordances are expected to bootstrap the learning in the next stages.

Our approach can be explained by the following intuitive example: Let us assume that the robot learned rollability affordances of the objects in the first development stage, and can now predict the rollability based on object shape properties. In the next stage, robot learns a more complex affordance such as stackability from two sample interactions where it observes that stacked two balls tumble over and stacked two boxes pile up. The robot, trained only with those two stacking interactions, can find a correspondence between stackability and rollability. Then, even if the robot does not have any stacking experience with cylindrical objects, it can make better predictions for stackability depending on the roll orientation (and affordance) of the cylinders.

In the context of robot affordance learning research, paired-object affordance learning has not been studied ex-

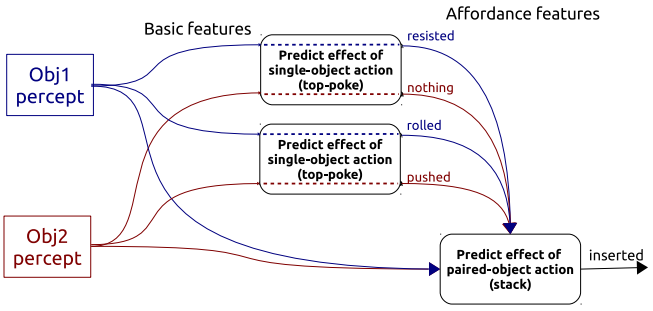


Fig. 1. Learning and prediction of action effects using *basic-features* (visual features) are shown with solid lines. The learned *affordance-features*, i.e. predicted effects, can be further used as input to classifiers which predict action effects of object pairs, i.e. in learning paired-object affordances.

tensively with exceptions of [3] where 'tool objects' are interacting with other objects, and [4] where two-object relational interaction models were directly learned. However none of these studies attempted to bootstrap their robot-object-object interaction dynamics with previously obtained skills and affordance knowledge.

Our method summarized in the next section is validated with an artificial interaction dataset that includes rich set of objects and interactions in Section III, in real world with robot experiments in Section IV.

II. Method

Learning of affordances corresponds to learning the relations between objects, actions and effects [5]. In this study, affordances are acquired through learning to predict what type of effects, i.e. discrete effect categories, can be generated given discrete robot actions and continuous object properties. To achieve this, we simply train a classifier for each action, which takes object features as input and predicts the effect category.

A. Object features

Here, we distinguish two different sets of object features. The first set includes hand-coded basic general-purpose features, computed from visual perception[1] with no explicit link to robot's actions. These may include standard features used in literature related to size, shape and local distance properties of the objects. The second set of features are acquired through interaction and they correspond to the higher-level learned ones that are computed from basic features. They encode the dynamics between robot actions (A) and object response (effect, ε). The first set of features is called *basic-features* whereas the second that is learned through interaction is called *affordance-features* as the latter includes the relations between objects, actions and effects.

[1]In this paper, we limit ourselves with the features that can be captured by vision only. However, object properties such as object friction or weight plays an important role on object-robot interaction dynamics. Thus, exploratory actions that can be used to perceive such properties should be implemented in a full-fledged scenario.

The straightforward approach to learn effect prediction is to train a classifier c for each action a that takes *basic-features* as input:

$$c^a_{\text{basic}}(\textit{basic-feat}) \rightarrow \text{effect}$$

whereas we propose to speed up learning of complex effect prediction using *affordance-features* that are computed using the learned basic effect prediction:

$$c^a_{\text{complex}}(\textit{basic-feat}, c_{\text{basic}}(.)) \rightarrow \text{effect}$$

which, in a *flat* form, corresponds to:

$$c^a_{\text{complex}}(\textit{basic-feat}, \textit{affordance-feat}) \rightarrow \text{effect}$$

Our approach is summarized in Fig. 1. The features shown with blue and red solid lines correspond to *basic-features* and action predictions based on these features give rise to *affordance-features*. The dashed lines correspond to *affordance-features*, that are learned in previous stages. The learning and prediction of complex affordances benefit from previously learned affordance features as shown in 'Predict effect of action k' predictor. Note that action k is considered to be a complex action as two objects are involved in execution.

In particular affordance features are represented as a vector of categorical variables, i.e. the list of the effect categories, predicted to be generated by single-object actions:

$$\textit{affordance-feat} = (\varepsilon^o_{a_1}, \varepsilon^o_{a_2}, ...)$$

where

$$\varepsilon^o_{a_i} = c^{a_i}_{\text{basic}}(\textit{basic-feat})$$

Complex affordance learning can be realized in different ways. In this paper, the action possibilities that are provided by two (or more) objects are considered to be complex. For instance, the effects created by a *stack* action (where the object is grasped and released over another one) is determined by the properties of both objects. We will use *affordance-features* (such as rollability, pushability, etc) and *basic-features* to learn and predict stackability affordances, and show that this learning significantly speeds up with predictors that are bootstrapped with *affordance-features*.

B. Active object selection based on affordances

We claim that the bootstrapping effect can be further increased if the objects to be explored (and learned next) are selected intelligently. A learner which is provided with a rich set of qualitatively different objects in its initial phases of development can perform better compared to the ones trained with complete random objects. Thus, an online learning system actively selects the next object to maximize the diversity of the training set, and the learned single-object affordances will be used as 'high-level' similarity measures between objects in computing this diversity.

The next object is selected from the set of possible objects (PossObjs) by maximizing the total distance from the next object to the already explored objects (ExpObjs) as follows:

$$nextObj = \underset{o_1 \in \text{PossObjs}}{\arg\max} \sum_{o_2 \in \text{ExpObjs}} dist_t(o_1, o_2)$$

where $dist_t(o_1, o_2)$ is the distance between two objects in the space of affordances.

$$dist_t(o_1, o_2) = \sum_a (1 - \delta_{\varepsilon_a^{o_1}, \varepsilon_a^{o_2}})$$

where $a \in A_{\text{Basic}}$, and $\delta_{i,j}$ is Kronecker delta function.

III. Bootstrapping in Artificial Data

In this section, we report our bootstrapping results obtained from a manually prepared artificial database of objects and interactions. The set of objects include cylinders, boxes, spheres and triangular prisms in different orientations and with/without holes as shown in Fig. 2. The set of manually encoded actions and their effects are as follows:

- Actions: {side-poke, top-poke, front-poke, stack}
- Poke-effects: {pushed, rolled, toppled, resisted, nothing}
- Stack-effects: {piled-up, inserted-in, covered, tumbled-over}

When poked from different directions, hypothetically, different effects can be generated with these objects. For example, when poked from side, lying cylinders would roll away, boxes would be pushed, objects with holes in poke direction would not be affected as finger would go through the hole without any interaction, and the tall objects would topple down. The effect of stacking objects on top of each other depends not only on their shape but also on their relative size as well. For example, while 'inserted-in' effect would generated when a small box is stacked on a hollow cylinder, 'piled-up' effect would be observed when the box is larger than the opening on top of the cylinder. Based on these assumptions, we manually created a hypothetical set of rules that give the effect based on object categories and their relative sizes.

A. Basic and affordance features

The classifier trained with *basic-features* uses the following features for training (and prediction later):

$$\text{TS}_{\text{basic}} = \{(shape^{o_1}, shape^{o_2}, dim^{o_1}, dim^{o_2})\}$$

where $shape$ includes mean and variance of the normals of the lateral surfaces, and the direction of the hole if it exists; and dim encodes the object size in different axes.

The classifier trained with *affordance-features* uses the following features:

$$\text{TS}_{\text{aff}} = \{(\varepsilon^{o_1}_{*\text{-}poke}, \varepsilon^{o_2}_{*\text{-}poke}, dim^{o_1}, dim^{o_2})\}$$

where ε^o refers to the effects of the corresponding poke action on the object o.

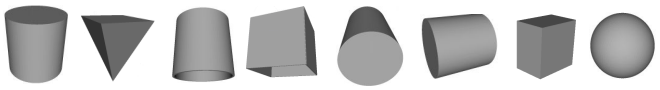


Fig. 2. The set of objects used in the artificial interaction database.

B. Bootstrapping Results

The performances of the classifiers trained with *basic-features* and *affordance-features* are provided in Fig. 3. We evaluated the classifiers by systematically changing the number of categories used in training set. For each number of categories, we trained 10 classifiers by selecting 5 objects of random size from each training category. To test these classifiers, we created test sets with random sized object from the remaining categories. Each bar corresponds to mean performance of these 10 classifiers. As shown, the prediction performance of both *basic-features* and *affordance-features* based classifiers improve by including more categories into the training set. We also included the performance of a category based predictor (which takes category index as input) to show the baseline. Because the categories used in training set are never included into test set, category-based predictors do fail independent of the training set size.

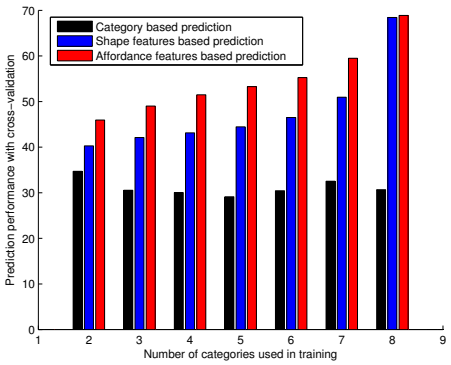


Fig. 3. The effect prediction performance of stack action obtained in artificial interaction database. The training of classifiers are done with the indicated number of categories with either shape features or affordance features. As shown, use of affordance features enable bootstrapping of the learning system.

These results show that because *affordance-features* already include properties related to object dynamics (pushability, rollability etc), classifiers that use these features have better performance especially for small training sets. With the increasing training set size, the effect of using high-level features is reduced as the *basic-features* classifier can also find the invariance related to stackability affordance with large dataset. Finding this invariance with small datasets is easier with *affordance-features* as they already include some properties of the agent-object-environment interactions.

IV. Bootstrapping in Real World

This section provides the details of the real world experiments where the effect of bootstrapping is analyzed. We first present the robot setup along with the details of robot's action

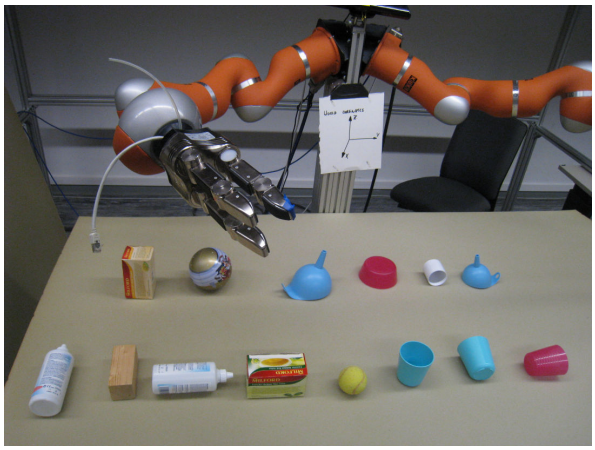


Fig. 4. The experiment setup. The environment includes one or two objects during experiments depending on the action type.

control and perception. Next, we showed that single-object affordances can be learned through interaction, and use of these acquired single-object affordances bootstraps paired-object affordance learning.

A. Robot system

The robot system employs a 7 DOF Kuka Light Weight Robot (LWR) arm, which is placed on a vertical bar similar to human arm in Fig. 4. A 3 fingered Schunk gripper is mounted on the arm to enable manipulation. For environment perception, Kinect sensor placed over the torso is used. The objects shown in Fig. 4 are used in learning single-object affordances as well as pairwise-affordances.

1) Object features: The robot's workspace consists of several objects and a table where the region of interest is defined as the volume over the table. First, the point cloud obtained from Kinect is transformed to robot's task space, then table is removed from the point cloud with a filtering along z-axis (see Fig 4), and finally objects are segmented based on depth information. Point Cloud Library normal estimation software is used next to compute a normal vector for each point of the object. The projection of each normal vector along each axis is separated, and histograms of normal vectors along each axis are computed. Using 8 bins for each histogram, $3 \times 18 = 54$ sized feature vector is obtained for shape related features. Note that in these experiments, an object is represented by a feature vector composed of only shape related features. Please see [5] for more details on histogram representation of normal vectors.

2) Robot Actions: The robot is equipped with a number of manually coded actions that enable single and paired object manipulation. The robot can 'poke' a single object from its side, front and top with *s-poke*, *f-poke*, and *t-poke* actions, respectively. It can also stack one object on the other using *stack* behavior, where it grasps the first object, move it on top of the other one and release it. The object position in world coordinate (shown in Fig. 4) is computed using the depth image of Kinect sensor. An inverse kinematic solver is used to compute the joint angles for initial and final

Fig. 5. The training set used for learning single-object affordances.

points defined in Cartesian space, and Reflexxes library [6] is utilized to generate smooth trajectories to achieve point-to-point movement. The action execution is as follows:

- Regarding to *poke* actions, the robot gripper is placed on one side of the object with $5cm$ distance with an orientation depending on the poke type. Two of the fingers are flexed to enable only the third finger to physically interact with the object (similar to index finger poking in humans). Next, the robot hand moves in the corresponding direction for $10cm$ towards the object and it is retracted after the poking is completed.
- Regarding to *stack* action, one object is grasped from above first by placing the gripper in a vertical orientation $10cm$ over of the object, then moving the wide-open gripper towards the object and finally enclosing it. Next, the gripper that carries the grasped object is repositioned over the second object in a vertical orientation again, and the object in the gripper is released over the first one by extending all the fingers.

3) Effect Categories: In the real world experiments, depending on the object(s) and the action executed, different effects were generated. When *poke* action was executed, the object was pushed, toppled over or rolled away depending on its shape. There was no effect in object state or robot's sensors if the robot finger went through the hole on the object. Finally, for *t-poke* action, all solid objects created resistance and obstructed gripper's movement that was detected using the force sensors. When *stack* action was executed, the objects in general piled up on top of each other if the object below provided a proper support (for example if it had a flat top surface). Depending on the existence of concave surfaces and holes, the released object was inserted in or hided the object below by encapsulating it. The released object also tumbled over due to the lack of stable support. Based on the above possibilities that we observed empirically, the sets of effect categories ($\mathcal{E}$) were set same as in previous section.

B. Experiment Results

1) Learning single-object affordances: The robot executed its poke actions on the objects (Fig. 4) placed in different orientations, and it collected 24 interaction instances for each poke action. The object shape features along with generated effect categories are stored for learning affordances. Support Vector Machine classifiers are used to learn the mapping between object features and effect categories. In order to analyze if the affordances for *poke* action are generalizable, we divide the interaction set into training and

Object from robot's view	Explanation	Front-poke prediction	Side-poke prediction	Top-poke prediction
	Lying cylinder	pushed	rolled	resisted
	Lying mug	rolled	pushed	resisted
	Cylinder wall	pushed	pushed	no-change
	Tennis ball	rolled	rolled	resisted
	Lying mug	pushed	rolled	resisted
	Wooden block	pushed	pushed	resisted
	Lying mug	pushed *	rolled	resisted
	Tea box	pushed	pushed	resisted
	Thin wooden block	pushed	toppled ***	resisted
	Thin wooden block	toppled **	pushed	resisted

Fig. 6. Robot's basic-affordance prediction on objects which are not included in the training set with the same orientations. Prediction fails in the examples with star (*), which are difficult cases to predict.

test sets with the deliberate purpose of distributing objects with same affordances into different sets. For each poke action, we trained a classifier using the objects given in Fig. 5. Then, we tested these classifiers by predicting the action effects on novel objects given in Fig. 6. As shown, the robot was able to detect the affordances of the object (in terms of effect prediction) correctly, except a small number of cases shown with stars (*), where the prediction also required perception and learning of material properties.

The trained three classifiers (for *s-poke*, *t-poke*, and *f-poke*) are transferred to the next stage and their predictions are used as high-level features to learn complex affordances.

2) Learning paired object affordances: In this section, the robot learns the paired-object affordances by exploring the two-object environments with its *stack* action. This learning is again achieved by training an SVM classifier that predicts the effect of the *stack* action given object features. Here we compare the prediction performance of the classifiers that are trained either with *basic-features* or *affordance-features*. Regarding to *basic-features*, normal vector histograms are used as we did in learning single-affordances in the previous subsection. Regarding to *affordance-features*, the list of effect predictions (provided by the classifiers transferred from the previous stage) for the poke actions are used.

The robot executed *stack* action with 18 pairs of random objects. A number of snapshots taken during these interactions are given in Fig. 7, where all the possible effects were observed with different object pairs. In each interaction, *basic-features* and *affordance-features* of both objects are computed and stored along with the observed effect category.

The classifier trained with *basic-features* uses the following features for training (and prediction later):

$$\text{TS}_{\text{basic}} = \{(shape^{o_1}, shape^{o_2})\}$$

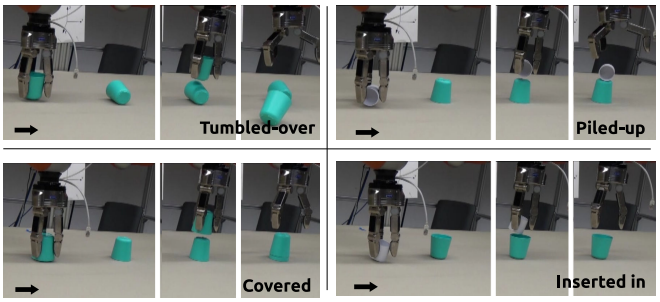


Fig. 7. Sample interactions observed during *stack* action execution.

and the classifier trained with *affordance-features* uses the following features:

$$\text{TS}_{\text{aff}} = \{(\varepsilon^{o_1}_{\textit{s-poke}}, \varepsilon^{o_1}_{\textit{f-poke}}, \varepsilon^{o_1}_{\textit{t-poke}}, \varepsilon^{o_2}_{\textit{s-poke}}, \varepsilon^{o_2}_{\textit{f-poke}}, \varepsilon^{o_2}_{\textit{t-poke}})\}$$

where $\{\}$ corresponds to the set operator.

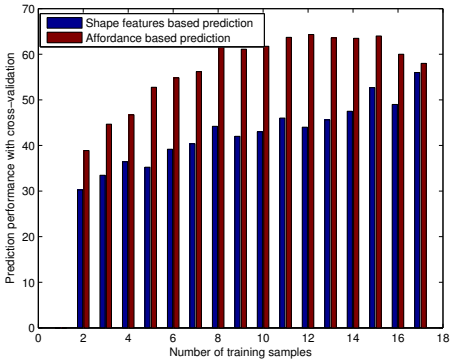


Fig. 8. The effect prediction performance of stack action that involves two objects. The training of classifiers are done with the indicated number of samples (interactions) with either shape features or affordance features. The initial high performance of *affordance-features* based classifiers demonstrates the advantage of using bootstrapping.

We evaluated the performance of these classifiers by systematically changing the size of the training set. For each training set size, we trained 10 classifiers using randomly selected samples. We tested each classifier using the remaining sample interactions. Fig. 8 gives these cross-validation results. Real-world experiment results are similar to the results obtained from the synthetic interaction dataset. As the *affordance-features* were obtained through interaction with the environment, they already encode the object-environment dynamics, which provides bootstrapping effect in learning multi-object affordances as shown. The *basic-features* are real valued larger sized vectors that encode object shape properties independent of robot-object dynamics. Thus they require more training data for learning. Additionally, *basic-features* are used in computation of *affordance-features*, so they contain the information to make predictions with the performance of affordance features. As shown in the figure, with increasing number of training samples, *basic-features* based classifier's performance indeed approached to the bootstrapped classifier's performance.

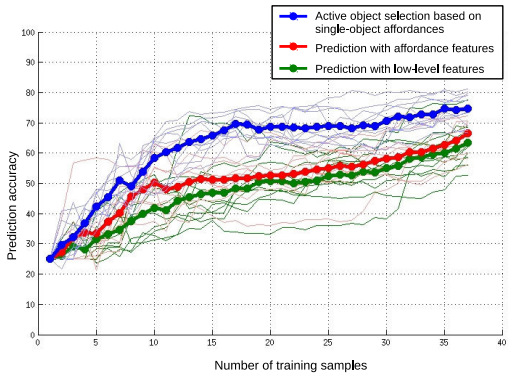


Fig. 9. The bootstrapping obtained by active selection of pairs of objects for learning of stack affordances.

V. Active selection of objects based on affordances

We used affordance distance measure defined in Section II.B, to maximize the diversity in training of paired-object affordance learning in an online learning setting. 83 objects and their poke and stack effects are used in this experiment (please see [7] for the complete list of objects). The object features are computed from Kinect depth image as in previous section, however, we used a human expert to label stack effect categories and it was not feasible to execute $83 \times 83 = 6889$ stack actions in the real robot. Fig. 9 shows the resulting performance of the basic-features and affordance-features based effect predictors trained with randomly selected objects, and of the effect predictors trained with the proposed active object selection strategy. Each type of predictor was trained 10 times, starting from a different random set of objects, and the thick lines correspond to the average accuracy for each predictor type. As shown, active selection of objects based on single-affordances provides a further bootstrapping effect in learning paired-object affordances. The best predictor was trained with affordance-features, but we observed that a similar performance was achieved even if it was trained with basic-features.

VI. Conclusion

Single-object affordances encode characteristics related to robot-object-environment dynamics as they are learned through robot's interaction with the objects. In this study, we showed that learned basic affordances can be used as additional features in order to bootstrap the next stage of development where complex paired-object affordances are learned. In our general model, affordances are used as 'additional features' for learning complex affordances, but in the experiments, in order to compare their independent performances, we used either only basic features or only affordance features. If an important basic action (such as top-poke) was unavailable, the affordance features, i.e. effect predictions for side-poke and front-poke actions, would have failed to predict insertability. Therefore, both channels should be used during learning and possibly a feature selection algorithm can filter out unnecessary channels.

While this work serves as one of the proof-of-concept application of the structural bootstrapping idea, we need to adapt advanced representations and learning methods (such as knowledge propagation framework of [8]) that can truly exhibit the real potential of this idea. We showed that this bootstrapping enabled the robot to speed up its learning particularly with small training data. Recently generative models have been proved to be effective in their ability in capturing object-action-effect dynamics, and in making predictions in different directions, for example in inferring the required actions to achieve desired effects given object properties [9], [4]. Particularly, hierarchical Bayesian networks directly encodes the desired structure and allows inference in several directions [10]. We discuss that our 'discriminative' model still provides powerful mechanisms as it can effectively map the continuous object feature and behavior parameter spaces to the corresponding effects [11] without any initial categorization of object properties as in [9], [4]. Furthermore, while bi-directional relations are not explicitly encoded in our system, we showed that our robot was able to make predictions in different directions, and made plans that involved sequence of actions on automatically selected objects[5].

Acknowledgements

This research was supported by European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

References

[1] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida, "Cognitive developmental robotics: a survey," *IEEE Tran. Auton. Mental Dev.*, vol. 1-1, 2009.

[2] E. Ugur, E. Sahin, and E. Oztop, "Self-discovery of motor primitives and learning grasp affordances," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 3260–3267.

[3] J. Sinapov and A. Stoytchev, "Detecting the functional similarities between tools using a hierarchical representation of outcomes," in *Proceedings of the 7th IEEE International Conference on Development and Learning*. IEEE, Aug. 2008, pp. 91–96.

[4] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt, "Learning relational affordance models for robots in multi-object manipulation tasks," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 4373–4378.

[5] E. Ugur, E. Oztop, and E. Sahin, "Goal emulation and planning in perceptual space using learned affordances," *Robotics and Autonomous Systems*, vol. 59, no. 7–8, pp. 580–595, 2011.

[6] T. Kroger, "Opening the door to new sensor-based robot applications – the reflexxes motion libraries," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 1–4.

[7] S. Szedmak, E. Ugur, and J. Piater, "Knowledge propagation and relation learning for predicting action effects," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[8] S. Szedmak and J. Piater, "An active learning based sampling design for structural bootstrapping," Univ. of Innsbruck, Tech. Rep., 2014.

[9] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory–motor maps to imitation," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 15–26, 2008.

[10] E. Gyftodimos and P. A. Flach, "Hierarchical bayesian networks: A probabilistic reasoning model for structured domains," in *ICML WS on Development of Representations*, 2002, pp. 23–30.

[11] E. Ugur, E. Oztop, and E. Sahin, "Going beyond the perception of affordances: Learning how to actualize them through behavioral parameters," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

TEMEL SAĞLARLIK TABANLI KARMAŞIK SAĞLARLIK ÖĞRENİMİ

COMPLEX AFFORDANCE LEARNING BASED ON BASIC AFFORDANCES

Emre Ugur, Sandor Szedmak ve Justus Piater
Intelligent and Interactive Systems, Institute of Computer Science
Innsbruck University
{emre.ugur,sandor.szedmak,justus.piater}@uibk.ac.at

***Özetçe* —Biz bu bildiride, objelerin sunduğu 'karmaşık sağlarlıkların' robotlar tarafından nasıl etkili bir şekilde öğrenilebileceğini ve bu amaçla önce öğrenilen yapıların nasıl kullanılabileceğini çalışmaktayız. Standart görsel özelliklerin yanısıra, robotun daha önce öğrendiği temel obje sağlarlıklarının kullanılmasının karmaşık sağlarlık öğrenmesini hızlandıracağını savunmaktayız. Bu hipotezimizi kanıtlamak için karmaşık sağlarlıkları öğrenen iki prediktör tipinin performansını karşılaştırdık: Objelerin şekil özelliklerine göre tahmin yapanlar ile objelerin temel sağlarlıklarına göre tahmin yapanlar. Yapay olarak yaratılmış bir (obje, aksiyon) etkileşim veritabanı kullanarak elde edilen sonuçlar, temel-sağlarlık bazlı prediktörlerin küçük öğrenme kümeleriyle eğitilseler bile daha önce karşılaşmadıkları objeler üzerinde önemli ölçüde genelleme yapabildiklerini göstermiştir. Bu sonuçlar göstermektedir ki karmaşık aksiyon etkileri öğrenmede kullanılan temel-sağlarlıklar, basit aksiyonlardan öğrenilmiş olsalar da, içlerinde obje-robot-ortam dinamiklerini barındırmakta ve karmaşık aksiyon öğrenmesini hızlandırmaktadır.**

Anahtar Kelimeler—sağlarlık, gelişimsel robotik

***Abstract*—In this paper, we study how complex object affordances can be efficiently learned and how previously learned structures can be used for this purpose. We discuss that besides standard visual features, using previously learned basic affordances in predicting complex affordances would speed up this complex learning task. In order to prove our hypothesis, we compared two different types of complex affordance predictors: The predictors that are based on shape features and the ones that use basic affordances. The results obtained from a synthetic (object, action) interaction database showed that basic-affordance based predictors can generalize over novel objects even with small training sets. This result shows that although the basic affordances are related to basic simpler actions, as they encode object-robot-environment dynamics, they can speed up learning of complex actions.**

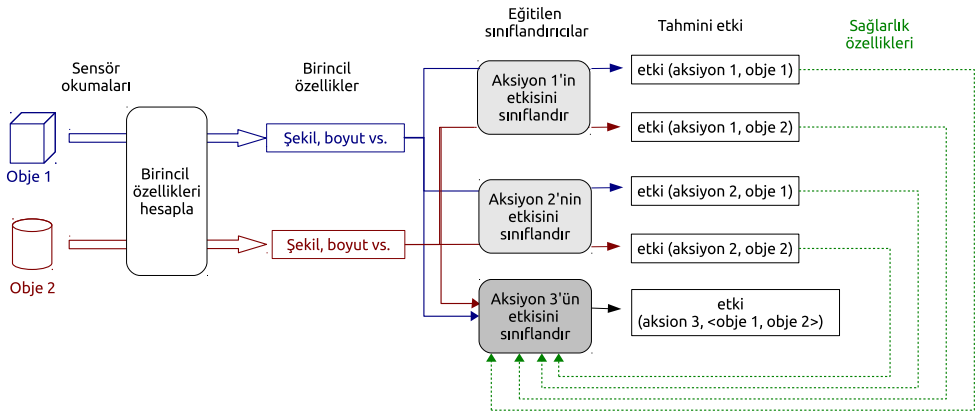
Keywords—affordances, developmental robotics

I. GİRİŞ

Sağlarlıklar (affordances), Ekolojik Psikoloji'de ortam tarafından canlılara sunulan ve canlıların direkt algılayabildiği aksiyon potansiyelleri olarak tanımlanmaktadır [1]. Örnek olarak önünde top gören bir insan bu objenin yuvarlanabilirlik sağladığını, yani ittirme davranışı ile yuvarlanacağını objenin şekilsel özelliklerinden hızlıca tahmin edebilmektedir. Araştırmacılar, ortam ve cisim sağlarlıklarının makine öğrenme yöntemleriyle robotlar tarafından öğrenilmesi konusunda özellikle son on yıldır yoğun olarak çalışmaktadırlar. Bu bildiri, çoklu objelerin sunduğu 'üstüstekoyulabilme' gibi karmaşık sağlarlıkları öğrenirken, robotun bir önceki gelişim evrelerinde öğrendiği daha temel sağlarlıkları kullanarak, bu karmaşık öğrenmenin nasıl hızlandırılabileceğini çalışmaktadır. Daha net bir şekilde ifade etmek istersek, önceki gelişim aşamasında bir objenin sunduğu sağlarlıkları algılamayı öğrenen robot, bir sonraki gelişim aşamasında çoklu obje sağlarlıklarını algılarken önceki aşamadan aktardığı tek-obje sağlarlıklarını kullanacaktır. Karmaşık sağlarlıkları en baştan öğrenmek yerine basit sağlarlıklar üzerinden öğrenmesi robot gelişimini özellikle ilk aşamalarında çok hızlandıracaktır.

Yukarıda özetlediğimiz yaklaşım şu örnek ile daha rahat anlaşılabilir: Robotun ilk gelişim aşamasında objelerin yuvarlanabilirliklerini (rollability affordance), çeşitli objeleri değişik noktalarından çeşitli yönlerde iterek öğrendiğini varsayalım. Öğrenme sonunda bu robot, önüne gelen bir objenin görsel algısından hesapladığı şekil özelliklerini kullanarak o objenin yuvarlanabilirliğini tahmin edebilme yeteneğine kavuşmuştur. Bir sonraki aşamada robotun objelerin *üstüstekoyulabilirliklerini* (stackability affordance) öğrenmeye çalıştığını düşünelim. Bu ikinci aşama öğrenme sırasında robot iki örnek etkileşimde bulunmuş olsun. Birinci etkileşimde robot bir kutuyu diğerinin üzerine koyduğunda kutuların devrilmediğini gözlemlemekte, ikinci etkileşimde ise üst üste koymaya çalıştığı kürelerin devrildiğini görmektedir. Bu gözlemlerden yola çıkan öğrenme sistemi yuvarlanabilirlik ile yığılabilirlik/devrilebilirlik arasında bir bağlantı kurarsa, ikinci aşamada daha önce etkileşimde bulunmadığı yeni nesnelerin sağlarlıklarına dair genellenebilen daha iyi tahminlerde bulunabilecektir. Örnek olarak ikinci aşamada silindirleri üst üste koyma deneyimi olmasa da, silindirlerin yuvarlanabilirliklerini bir önceki aşamada öğrendiği için, yatay silindirlerin üst üste konulduklarında devrilirken dikey silindirlerin devrilmeyeceğini tahmin edebilecektir.

978-1-4673-5563-6/13/$31.00



Şekil 1: Basit aksiyon etkilerinin öğrenim ve tahmininde kullanılan *birincil-özellikler* kesintisiz çizgilerle gösterilmektedir. Kesintili çizgilerle gösterilen öğrenilmiş sağlarlık-özellikleri ise, çoklu obje etkilerini tahmin eden, bir başka deyişle çoklu obje sağlarlıklarını öğrenen sınıflandırıcıya tarafından kullanılmaktadır.

Literatürü taradığımız zaman, tek-obje sağlarlıklarının yoğun olarak çalışılmasına karşın çoklu-obje sağlarlıklarına dair çalışmaların kısıtlı olduğunu görüyoruz. İstisna olarak yakın zamanda çoklu-obje sağlarlıkları, ittirilen objelerin masa üzerindeki diğerler objelerle etkileşebileceği ortamlarda göreceli uzaklık ve oryantasyonları hesaba katılarak öğrenilmiştir [2]. Bunun yanında araç sağlarlıklarının (tool affordances) çalışıldığı, yani ortamdaki objeler ile robotun kullandığı araç objeler arasındaki dinamiklerin öğrenildiği çalışmalar çoklu obje sağlarlıkları kategorisinde değerlendirilebilir [3]. Fakat geçmiş çalışmaların hiçbiri çoklu sağlarlık öğrenmesini, öğrenilmiş tek obje sağlarlık bilgisi üzerine inşa etmemektedir.

II. METOD

Genel olarak sağlarlık öğrenimi, objeler, robot aksiyonları ve aksiyonların uygulanması sonucunda yaratılan etkiler arasındaki ilişkileri öğrenmeye karşılık gelmektedir [4]. Bu çalışmada ise sağlarlık öğrenimi, robotun özelliklerini algıladığı obje üzerinde aksiyonlarından birisini uyguladığı zaman ne çeşit etkiler yaratacağını tahmin etme yeteneği kazanmasına denk gelmektedir. Örnek olarak, robot önündeki bir objeyi ittiği zaman obje masa üzerinde sürüklenebilir, yuvarlak olursa yuvarlanıp masadan düşebilir yada çok ağır olup hareket etmeyebilir. Robotun herhangi bir aksiyonda ne çeşit bir etki yaratacağını objenin özelliklerine göre tahmin etmesi, objenin sağlarlığını tahmin etmesine denk düşmektedir. Bu yeteneği elde edebilmek için robot, aksiyonlarını özellikleri farklı objeler üzerinde uygulayarak ve oluşan etki kategorilerini gözleyerek (obje-özellikleri, ekti-kategorileri) deneyimlerini biriktirir. Daha sonra her aksiyonu için obje özelliklerini girdi olarak alıp etki kategorisini çıktı olarak veren sınıflandırıcılar (classifiers) eğitmektedir.

Burada obje özelliklerini iki sınıfa ayırmaktayız. Birinci sınıf el ile kodlanmış genel amaçlı özellikleri kapsamaktadır. Bu özellikler görsel algıdan hesaplanan ve robotun aksiyonlarıyla belirgin ilişkisi olmayan özelliklerdir. Örnek olarak literatürde standart olarak kullanılan büyüklük, şekil, renk ve obje-parçalarıyla ilgili özellikler bu sınıfa dahildir. İkinci sınıftaki özellikler ise robotun çevresiyle etkileşimi sonucunda elde edilmiş olup birinci seviyeli özellikler kullanılarak hesaplanan ikincil yüksek sıralı (higher-order) özelliklerdir. Dolayısıyla ikincil özellikler robotun aksiyonlarıyla objeler arasındaki dinamiği kodlamaktadırlar. Bu noktadan itibaren el ile kodlanmış genel özellikleri *birincil-özellikler*, etkileşim ve öğrenme yoluyla elde edilenleri ise *sağlarlık-özellikleri* olarak adlandıracağız.

Etki kategorilerini tahmin etmenin en dolambaçsız yolu objenin birincil özellikleri ve aksiyonları girdi olarak alıp etkiyi bulan fonksiyonlar bulmak iken

$$f_{\text{birincil}}(\text{aksiyon}, \textit{birincil-özellikler}) \rightarrow \text{etki}$$

biz, etkilerin karmaşık aksiyonlarda ve karmaşık ortamlarda tahmininin öğrenimini kolaylaştırmak ve hızlandırmak için sağlarlık-özelliklerinin de girdi olarak kullanılmasını önermekteyiz:

$$f_{\text{karmaşık}}(\text{aksiyon}, \textit{birincil-özellikler}, f_{\text{birincil}}()) \rightarrow \text{etki}$$

Karmaşık etki tahminini yatay bir şekilde yazacak olursak, aşağıdaki forma dönüşecektir:

$$f_{\text{karmaşık}}(\text{aksion}, \textit{birincil-özellikler}, \textit{sağlarlık-özellikleri}) \rightarrow \text{etki}$$

Yaklaşımımız Şekil 1'de özetlenmektedir. Mavi ve kırmızı kesintisiz oklarla ile gösterilen özellikler objenin *birincil-özelliklerine* denk gelmekte ve bu özellikleri kullanarak yapılan etki tahminleri sağlarlık-özelliklerini (kesikli yeşil çizgiler) oluşturmaktadır. 'k aksiyonunun etkisinin tahmini' ile gösterilen sınıflandırıcı, karmaşık sağlarlıkları öğrenmek ve tahmin etmekte önceki aşamalarda öğrenilmiş sağlarlık özelliklerinden faydalanmaktadır (k aksiyonunun karmaşık aksiyon olarak adlandırılmasının bu figürdeki nedeni, birden fazla obje üzerinde uygulanmasıdır).

Bu çalışmada iki yada daha fazla objenin sunduğu sağlarlıklar karmaşık sağlarlık olarak nitelendirilmektedir. Örnek olarak üst üste koyma aksiyonunun sonucunda oluşacak etki, hem yukarıdan bırakılan hem de aşağıdaki objelerin ilişkisel özelliklerine bağlıdır. Yuvarlanabilirlik, ittirilebilirlik gibi *sağlarlık-özellikleri* ve şekil, büyüklük gibi *birincil-özellikler* kullanılarak *üstüstekoyulabilirlik* sağarlığı öğrenilecektir. Amacımız, bu öğrenmenin hızının *sağlarlık-özellikleri* kullanılarak önemli ölçüde arttığını göstermektir.

Tablo I: Üst üste koy aksiyonu sonucu oluşan etkileri belirleyen kurallar kümesi. Üst satırda gösterilen objelerin solda gösterilen objelerin üzerine düşmesi sonucu oluşan etkiler, denk gelen tablo hücrelerindeki kurallara göre belirlenmektedir. Etkiler objelerin şekillerinin yanında büyüklükleri tarafından da etkilenmektedir. Burada d, g ve u, cisimlerin derinlik, genişlik ve uzunluklarını ifade etmektedir.

	Dik silindir (DS)	Boş dik silindir (BDS)	Yatay silindir 1 (YS1)	Yatay silindir 2 (YS2)	Küp	Boş Küp (BKüp)	Küre	Üçgen Prizma (UP)
DS	d1>d2 → devril d1<d2 → otur	d1>d2 → kapsa at. → otur	u1<d2 → otur at. → devril	u1<d2 → otur at. → devril	g1<d2 & d1<d2 → otur at. → devril	g1<d2 & d1<d2 → otur g1>d2 & d1>d2 → kapsa at. → devril	d2 > C → otur at. → devril	g1>d2 & d1>d2 → otur at. → devril
BDS	d1>d2 → devril d1<d2 → içine	d1>d2 → kapsa d1<d2 → içine	u1<d2 → içine at. → devril	u1<d2 → içine at. → devril	g1<d2 & d1<d2 → içine at. → devril	g1<d2 & d1<d2 → içine g1>d2 & d1>d2 → kapsa at. → devril	içine	g1>d2 & d1>d2 → içine at. → devril
YS1	devril	d1>u2 & d1>d2 & u1>d2/2 → kapsa at. → devril	devril	devril	devril	g1>u2 & d1>d2 & u1>d2/2 → kapsa at. → devril	devril	devril
YS2	devril	d1>u2 & d1>d2 & u1>d2/2 → kapsa at. → devril	devril	devril	devril	g1>d2 & d1>u2 & u1>d2/2 → kapsa at. → devril	devril	devril
Küp	d1<g2 & d1<d2 → otur at. → devril	d1<g2 & d1<d2 → otur d1>g2 & d1>d2 → kapsa at. → devril	u1<g2 & d2>C → otur at. → devril	u1<d2 & g2>C → otur at. → devril	g1<g2 & d1<d2 → otur at. → devril	g1>g2 & d1>d2 → kapsa g1<g2 & d1<d2 → otur at. → devril	d2>C & g2>C → otur at. → devril	g1<g2 & d1<d2 → otur at. → devril
BKüp	d1<g2 & d1<d2 → içine at. → devril	d1<g2 & d1<d2 → içine d1>g2 & d1>d2 → kapsa at. → devril	u1<g2 → içine at. → devril	u1<d2 → içine at. → devril	g1<g2 & d1<d2 → içine at. → devril	g1>g2 & d1>d2 → kapsa g1<g2 & d1<d2 → içine at. → devril	içine	g1<g2 & d1<d2 → içine at. → devril
Küre	devril	d1>d2 & u1>d2/2 → kapsa at. → devril	devril	devril	devril	d1>d2 & g1>d2 & u1>d2/2 → kapsa at. → devril	devril	devril
UP	devril	devril d1>d2 → kapsa at. → devril	devril	devril	devril	d1>d2 → kapsa at. → devril	devril	devril

III. DENEYLER

A. Deney Düzeni

Deneylerde farklı şekil ve büyüklükte boş ve dolu silindir, kutu, küre ve üçgen prizmalardan oluşan bir yapay obje kümesi kullanılmıştır. Bu cisimler değişik yönlerde ittirildiklerinde değişik etkiler gözlemlenebilir. Örnek olarak yandan itildiğinde yana yatmış silindirler yuvarlanacak, kutular bir miktar sürüklenecek, uzun objeler devrilecektir. Boş silindirin tepesinden bir dürtme hiçbir etkileşim yaratmayacağı için ne robotun kuvvet sensörlerinde ne de objenin kendisinde bir etki yaratmayacak, tepeden dürtülen dolu dik bir silindir ise robotun hareketine direnç gösterecek ve robotun kuvvet sensörlerinde ters yönde yüksek etki yaratacaktır. Manuel olarak kodlanmış aksiyonlar ve olası etkileri aşağıda verilmiştir:

- Aksiyonlar: {yandan-dürt (*y-dürt*), tepeden-dürt (*t-dürt*), önden-dürt (*ö-dürt*), üst-üste-koy}
- Dürtme etkileri: {sürüklendi, yuvarlandı, devrildi, direndi, etkisiz}
- Üstüste-koyma etkileri: {Üstünde-durdu, içine-girdi, üzerini-örttü, devrildi}

Üst üste koyma sonucunda oluşan etkiler, objelerin şekillerine ve göreceli büyüklüklerine bağlıdır. Örnek olarak 'içine-girdi' etkisi küçük bir kutu büyük boş bir silindirin üzerine koyulunca gözlenirken, eğer kutu silindirin boşluğundan büyükse 'üstünde-durdu' etkisi oluşacaktır. Bu ve buna benzer varsayımları kullanarak, obje kategorilerine ve büyüklüklerine bağlı olarak etkileri hesaplayan hipotetik bir kurallar kümesi Tablo I'de açıklanmaktadır. Burada, d, g ve u objeların farklı eksenlerdeki boyutlarına denk gelmektedir ve sırasıyla derinlik, genişlik ve uzunluk olarak adlandırılabilir.

B. Birincil özellikler ve sağlarlık özellikleri

Birincil özelliklerle eğitilmiş sınıflandırıcılar öğrenme ve daha sonra tahmin için aşağıdaki eğitim kümesini (EK) kullanmaktadır:

$$\text{EK}_{\text{birincil}} = \{(şekil^{o_1}, şekil^{o_2}, boyut^{o_1}, boyut^{o_2})\}$$

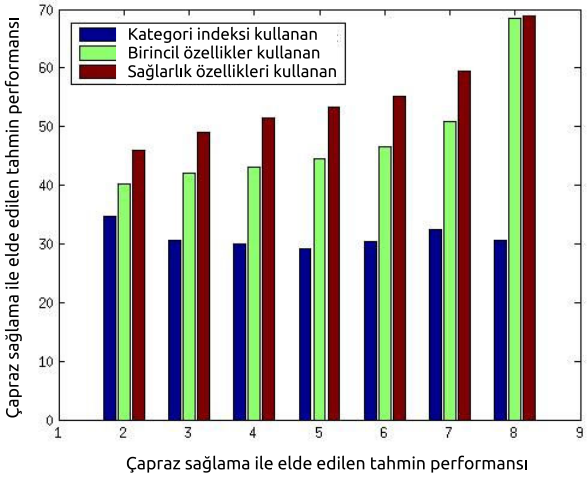
Burada *şekil* özellikleri yüksek seviyeli eğim bilgisini ve eğer bir boşluk varsa boşluğun yönünü içermekte, $boyut$ ise objenin boyutlarını farklı eksenlerde kodlamaktadır. Sağlarlık özellikleri ile eğitilen sınıflandırıcılar ise girdi olarak aşağıdaki özellikleri kullanmaktadır.

$$\text{EK}_{\text{sağlarlık}} = \{(\varepsilon^{o_1}_{\text{y-dürt}}, \varepsilon^{o_1}_{\text{ö-dürt}}, \varepsilon^{o_1}_{\text{t-dürt}}, \varepsilon^{o_2}_{\text{y-dürt}}, \varepsilon^{o_2}_{\text{ö-dürt}}, \varepsilon^{o_2}_{\text{t-dürt}}, boyut^{o_1}, boyut^{o_2})\}$$

Burada ε^o, objeye uygulanan dürtme aksiyonuna karşılık gelen etki kategorisini ifade etmektedir ($\varepsilon^o \in$ {sürüklendi, yuvarlandı, devrildi, direndi, sıfır}). ε^o her dürtme aksiyonu ve obje kategorisi için manuel olarak kodlanmış olsa da, bu etkinin şekil özelliklerinden hesaplanabileceğini ve bir önceki gelişim aşamasında öğrenilebildiğini daha önceki çalışmalarımızda göstermiştik [5].

C. Deney Sonuçları

Birincil-özellikler ve *sağlarlık özelliklerini* kullanan sınıflandırıcıları, öğrenmede kullanılan obje kategorilerinin sayısını (n) sistematik olarak arttırarak değerlendirdik ve performans sonuçlarını Şekil 2'de sunduk. Sınıflandırıcı olarak RBF kernelli Support Vector Machine kullandık. Her sınıflandırıcının öğrenme kümesinde kullanması için 9 kategori içerisinden n kategori rastgele seçilmiş, seçilen her kategoriden 5 rastgele büyüklükte obje yaratılmıştır. Yaratılan her obje çifti için üst üste koy aksiyonu sonucunda oluşan etkiler Tablo I kurallar kullanarak otomatik olarak hesaplanmış ve $((n \times 5) \times (n \times 5))$ elemandan oluşan etkileşimler kümesi öğrenme için kullanılmıştır. Sınıflandırıcıyı test etmek



Şekil 2: Yapay etkileşim verikümesi kullanılarak elde edilen etki tahmin performansı. Kategori indeks numarası, birincil özellikler yada sağlarlık özellikleri kullanan 3 sınıflandırıcı türünün sonuçları karşılaştırılmaktadır. Gösterildiği üzere, sağlarlık özellikleri, zaten obje-dünya dinamiklerini barındırdıkları için, eğitimlerinde az sayıda obje kategorisi kullansalar bile çok daha iyi genelleme yapabilmekte ve yüksek performans gösterebilmektedir.

için ise geri kalan $(9 - n)$ kategoriden benzer şekilde oluşturulan $((9 - n) \times 5) \times ((9 - n) \times 5)$ elemanlı etkileşim kümesi kullanılmıştır. Grafikte sunulan değerler, her kategori sayısı için eğitilen 10 farklı sınıflandırıcının ortalama performans değerleridir. *Birincil-özellikler* ve *sağlarlık-özellikleri* kullanan sınıflandırıcıların yanısıra, objelerin kategori indeks numarasını girdi olarak alan üçüncü bir sınıflandırıcı kullandık. Kategori indeks numarası üzerinden bir genelleme yapılamayacağı için, bu sınıflandırıcılardan elde edilen sonuçlar sistemin baz performansı kabul edildi. Şekilde görüldüğü üzere, hem birincil özellikleri kullanan hem de sağlarlık özelliklerini kullanan sınıflandırıcıların eğitimindeki kategori sayısı arttıkça performansı artmaktadır. Oysa, gelişim kümesinde kullanılan kategoriler test kümesinde yer almadığı için, kategori indeks bazlı sınıflandırıcılar kullanılan kategori sayısından bağımsız olarak başarısız kalmaktadır.

Bu sonuçlar göstermektedir ki, *sağlarlık-özellikleri* obje dinamiklerini (itilebilirlik, yuvarlanabilirlik vs.) içinde barındırdıkları için, bu özellikleri kullanan sınıflandırıcıların performansı özellikle küçük eğitim kümelerinde daha yüksektir. Eğitim kümesi büyüklüğü ve çeşitliliğinin artması ile sağlarlık özelliklerinde zaten kısmi olarak varolan değişmezlikler (invariance), *birincil-özellikler* tarafından da bulunmaya başlamaktadır. Değişmezliklerin sağlarlık özellikleri üzerinden küçük veri kümelerinde dahi bulunabilmesinin nedeni, robot-obje-ortam dinamiklerini temsil eden yapısal özelliklerin zaten bu özelliklerin içinde (basit aksiyonlarla) öğrenilmiş ve kodlanmış olmasıdır.

D. SONUÇ

Bu çalışmada robotun bir önceki aşamada öğrendiği temel sağlarlıkların sonraki aşamalara aktarıldığında karmaşık sağlarlık öğrenimini hızlandırabileceğini gösterdik. Yapay olarak yaratılmış veritabanından elde ettiğimiz sonuçlar bu yaklaşımımızı desteklese de sistemimizi gerçek robotlarla yapacağımız benzer deneylerle desteklemek zorundayız. Ayrıca bu bildirede önerilen metoddaki en önemli sınır, tahmin ve öğrenmenin tek yönlü yapısıdır (F.igür 1). Örnek olarak bu öğrenme sistemi istenen bir etkiyi yaratmak için gerekli aksiyonları direkt tahmin etmek mümkün değildir. Fakat bu aksiyonlar ve kullanılıyorsa aksiyon parametreleri iteratif yöntemler yoluyla dolaylı olarak bulunabilmektedir [6]. Diğer yandan sağlarlık özelliklerinin temsili gücü arttırılarak bu öğrenme sistemi önemli ölçüde geliştirilebilir. Mevcut durumunda her aksiyon için sağlarlık özelliği ayrık aksiyonun üreteceği tek değişkenli etki olarak kodlanmaktadır. Bu, parametrik aksiyonun etkilerinin dağılımını özetleyen bir yapısal model ile değiştirilebilir. Örnek olarak, Detry ve diğerleri'nin önerdiği [7], nesne yüzeylerinin o bölgelerin robot eliyle nasıl kavranılacağını temsil eden vektörel yoğunluklar öğrenilebilir ve daha sonra *sağlarlık-özellikleri* olarak kodlanarak çok objeli karmaşık sağlarlıkları öğrenmede kullanılabilir.

TEŞEKKÜR

Bu araştırma Avrupa Birliği Yedinci Çerçeve Programı (European Community's Seventh Framework Programme FP7/2007-2013, Specific Programme Cooperation, Theme 3, Information and Communication Technologies) 270272 numaralı ödenek sözleşmesi altındaki Xperience projesi tarafından tarafından desteklenmiştir.

KAYNAKÇA

[1] J. J. Gibson, *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, 1986.

[2] M. B., P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt, "Learning relational affordance models for robots in multi-object manipulation tasks," in *ICRA*, 2012, pp. 4373–4378.

[3] J. Sinapov and A. Stoytchev, "Detecting the functional similarities between tools using a hierarchical representation of outcomes," in *ICRA*, 2008, pp. 91–96.

[4] E. Şahin, M. Çakmak, M. R. Doğar, E. Ugur, and G. Üçoluk, "To afford or not to afford: A new formalization of affordances toward affordance-based robot control," *Adaptive Behavior*, vol. 15, no. 4, pp. 447–472, 2007.

[5] E. Ugur, E. Oztop, and E. Sahin, "Goal emulation and planning in perceptual space using learned affordances," *Robotics and Autonomous Systems*, vol. 59, no. 7–8, pp. 580–595, 2011.

[6] E. Ugur, E. Sahin, and E. Oztop, "Self-discovery of motor primitives and learning grasp affordances," in *IROS*, 2012, pp. 3260–3267.

[7] R. Detry, E. Başeski, M. Popović, Y. Touati, N. Krüger, O. Kroemer, J. Peters, and J. Piater, "Learning continuous grasp affordances by sensorimotor exploration," in *From Motor Learning to Interaction Learning in Robots*, 2010, vol. 264, pp. 451–465.

2014 14th IEEE-RAS International Conference on
Humanoid Robots (Humanoids)
November 18-20, 2014. Madrid, Spain

Speed profile optimization through directed explorative learning

Rok Vuga, Bojan Nemec and Aleš Ude[1]

Abstract— **In this paper we propose a new skill learning framework based on fusing prior knowledge with programming by demonstration and explorative learning methodologies. Prior knowledge as well as all partially known models guide the search process within the proposed adaptation method. The proposed methodology is based on algorithms originating in iterative learning control and reinforcement learning. The developed approach was experimentally verified on the problem of speed profile optimization for a challenging task of transferring vessels filled with liquid without spilling. In order to explicitly encode the speed profiles and to allow their transfer between tasks, a modified form of dynamic movement primitives has been developed.**

I. Introduction

Autonomous acquisition and refinement of skills is one of the major challenges of contemporary humanoid robotics. An often used paradigm to initiate knowledge transfer from humans to humanoid robots is programming by demonstration [1], [2], where the demonstrated actions are used to seed the learning process. The initially obtained knowledge should then be adjusted and refined to account for different kinematic and dynamic capabilities of a human demonstrator and a target humanoid robot [3]. Adaptation process, where the robot modifies the available movements by exploring its action space in the neighborhood of previously acquired movements, is usually based on reinforcement learning (RL) techniques [4], [5], [6]. A major problem here is the huge search space that needs to be explored, which is affected not only by the high number of degrees of freedom of a humanoid robot, but also by the underlying policy representation and the robot's environment. Recently proposed probabilistic RL algorithms like PI^2 [7] and PoWER [8] can scale to significantly more complex problems and reduce the number of tuning parameters. However, due to the high dimensionality of the parameter space, the adaptation speed of these algorithms is still low compared to humans, who are able to quickly adapt to new situations by exploiting previous experience and by generalizing from previously solved similar situations. Therefore, the issues of how to speed up the adaptation process and how to include knowledge from previous, similar situations are among the biggest challenges that need to be overcome to develop truly autonomous humanoid robots.

[1]Humanoid and Cognitive Robotics Lab, Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia. rok.vuga@ijs.si,bojan.nemec@ijs.si, ales.ude@ijs.si

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

Many of the recent approaches address the above problem by reducing the set of adaptation parameters, consequently reducing the search space. Kormushev et al. [9] proposed the parameter update formed as a linear combination of parameters reweighted according to the reward in previous rollouts. Grollman and Billard [10] constrained the search space to the area between two appropriately chosen unsuccessful demonstrations. Kober et al. [11] proposed adaptation of the control policy based on a small set of global parameters, called meta-parameters. Nemec et al. [12] combined the ideas of RL and statistical generalization, where a normally low dimensional query into a database of movements was adapted by means of RL. Stulp et al. [13] investigated the adaptation of the covariance matrix, which governs the generation of exploration noise in the RL algorithm.

However, learning in reduced dimensionality can not guarantee that an optimal policy will be obtained in general because the optimal solution might lie outside of the reduced dimensionality space. Therefore, in our previous research [14] we suggested that learning in reduced dimensionality should be followed by learning in the full parameter space. Unfortunately, such approaches require ad hoc hard coding of switching between the learning algorithms. In this paper we focus on how to transfer the knowledge between similar tasks and how to improve the adaptation speed by combining the ideas of Iterative Learning Control (ILC) [15] and reinforcement learning. In contrast to the standard approach, where Gaussian noise is used for parameter exploration, we propose to combine knowledge transferred from similar tasks and ILC for the initial parameter exploration. The approach can be related to the covariance matrix adaptation methods, which follow a similar idea, i. e. to apply guided search in the parameter space in order to increase the adaptation speed. The initial adaptation based on ILC is much faster than what can be achieved by reinforcement learning and results in skill knowledge that is much better adapted to the capabilities of the robot than knowledge originating in user demonstrations. RL, however, is still needed to achieve the final fine tuning of the task. The proposed approach was verified on velocity adaptation task, where the robot carried a cup of liquid as fast as possible without spilling.

The paper is organized as follows. In Section II we outline the dynamic movement primitives framework along with the extension for speed profile encoding. Next, in Section III we present the main contribution of this paper, the integrated application of ILC and reinforcement learning to accelerate policy iteration. An application of the proposed approach to speed learning is outlined in Section IV. Finally, Section V presents experimental evaluation.

978-1-4799-7173-2/14/$31.00 ©2014 IEEE

II. DMPs AND SPEED PROFILE

We start with initial user demonstration acquired either in joint or Cartesian space

$$\mathcal{G} = \{\mathbf{y}(k), \dot{\mathbf{y}}(k), \ddot{\mathbf{y}}(k), t(k)\}_{k=1}^{T}, \tag{1}$$

where $\mathbf{y}(k) \in \mathbb{R}^D$ are the corresponding coordinates and T is the number of samples on the demonstrated trajectory. In the following we use notation $\mathbf{y}(k), \dot{\boldsymbol{y}}(k), \ddot{\boldsymbol{y}}(k), x(k)$ to denote the measurements on the trajectory and $\mathbf{y}(x), \dot{\boldsymbol{y}}(x), \ddot{\boldsymbol{y}}(x), x$ to denote values obtained through integration of (2) – (4).

Next, we parameterize the given policy with dynamic movement primitives (DMP) [16]. In the original formulation, the speed profile is embedded into the representation with the DMP parameters, which are normally obtained with regression. In [17] we proposed a formulation which allows to non-uniformly change the speed profile, but the initial speed profile was still encoded by the initial set of DMP parameters. In order to allow the transfer of speed profiles between policies, the speed profile has to be explicitly encoded. As in the original representation every degree of freedom is described by its own dynamic system, but with a common phase to synchronize them. For discrete movements, the trajectory of each robot degree of freedom y is described by the following system of nonlinear differential equations

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) + f(x), \tag{2}$$

$$\tau \dot{y} = z, \tag{3}$$

$$\tau \dot{x} = -\alpha_x x \nu(t). \tag{4}$$

where x is the phase variable and z is an auxiliary variable. This way, the system converges to the unique equilibrium point $(z, y, x) = (0, g, 0)$. The nonlinear forcing term f contains free parameters that are used to modify the dynamics of the second-order differential equation system. These parameters can be calculated to approximate any smooth point-to-point trajectory from the initial position y_0 to the final configuration g

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x(g - y_0),$$

$$\Psi_i(x) = \exp\left(-h_i \left(x - c_i\right)^2\right), \tag{5}$$

with the given initial velocity and final velocity equal to zero. Here c_i are the centers of radial basis functions distributed along the trajectory and $h_i > 0$ their widths. Weights w_i determine the shape of the system's response. The time constant is set to $\tau = t(T) - t(1)$, whereas α_x, α_z, and β_z are set so that the underlying second order linear dynamic system is critically damped [16], e. g. $\alpha_z = 12$, $\beta_z = 3$, $\alpha_x = 2(t(T) - t(1))$. Note that (4) is independent of (2) – (3) and can be solved analytically

$$x(t) = \exp\left(-\frac{\alpha_x}{\tau} \int_0^t \nu(r) \mathrm{d}r\right), \tag{6}$$

where we assumed that $t(1) = 0$. This can easily be checked by calculating the derivative of (6).

ν is the temporal scaling function which encodes the speed profile of the trajectory. It is defined as a time dependent function of the form

$$\nu(t) = \frac{\|\dot{\boldsymbol{y}}(t)\|}{\int_0^\tau \|\dot{\boldsymbol{y}}(t)\| \mathrm{d}t}. \tag{7}$$

By combining (6) and (7) we obtain

$$x(t) = \exp\left(-\frac{\alpha_x}{\tau} \frac{\int_0^t \|\dot{\boldsymbol{y}}(r)\| \mathrm{d}r}{\int_0^\tau \|\dot{\boldsymbol{y}}(t)\| \mathrm{d}t}\right). \tag{8}$$

Note that expression

$$\int_0^t \|\dot{\boldsymbol{y}}(r)\| \mathrm{d}r \tag{9}$$

defines the arc length of trajectory $\mathcal{G}$ up to time t. Recall that standard DMPs [16] use the canonical equation $\tau \dot{x} = -\alpha_x x$ instead of (4), which solution is given by $x(t) = \exp(-\alpha_x t/\tau)$. So by defining a new variable

$$s(t) = \frac{\int_0^t \|\dot{\boldsymbol{y}}(r)\| \mathrm{d}r}{\int_0^\tau \|\dot{\boldsymbol{y}}(t)\| \mathrm{d}t}, \tag{10}$$

we have parameterized the phase with normalized arc length s. Motion control is simpler if trajectories are parameterized by arc length instead of time. To move at constant speed along the path of an arc-length parameterized curve, the controller needs only to evaluate the parametric function at parameter values separated by the speed times the inter-frame time interval [18].

By numerical integration of (7) and (8) we obtain the following approximations for scaling function ν and phase x at times $t(k)$

$$\nu(k) = \frac{\|\dot{\boldsymbol{y}}(k)\|}{\mathrm{Trapzd}(T)} \tag{11}$$

and

$$x(k) = \begin{cases} 1, \ k = 1 \\ \exp\left(-\frac{\alpha_x}{\tau} \frac{\mathrm{Trapzd}(k)}{\mathrm{Trapzd}(T)}\right), \ k > 1 \end{cases} \tag{12}$$

In the above equations we applied the trapezoidal rule for numerical integration [19]

$$\mathrm{Trapzd}(k) = \Delta t \left(\frac{1}{2}\|\dot{\boldsymbol{y}}(1)\| + \sum_{n=2}^{k-1} \|\dot{\boldsymbol{y}}(n)\| + \frac{1}{2}\|\dot{\boldsymbol{y}}(k)\|\right). \tag{13}$$

In (13) we assumed that integration step Δt is constant.

Next, we rewrite the system of two first order equations (2) – (3) as one second order equation (by replacing z with $\tau \dot{y}$)

$$\tau^2 \ddot{y} = \alpha_z(\beta_z(g - y) - \tau \dot{y}) + f(x). \tag{14}$$

We then define

$$f(k) = \tau^2 \ddot{y}(k) + \tau \alpha_z \dot{y}(k) - \alpha_z \beta_z (g - y(k)), \tag{15}$$

where $y(k), \dot{y}(k), \ddot{y}(k)$ are the samples of one of the components of the initially demonstrated trajectory $\mathcal{G}$. For simplicity we omitted the index denoting the dimension of the trajectory. To calculate the weights of nonlinear forcing term $f(x)$ from (5), we need to solve the linear equation system

$$\frac{\sum_{i=1}^{N} w_i \psi_(x(k))}{\sum_{i=1}^{N} \psi_i(x(k))} = f(k), k = 1, \ldots, T, \tag{16}$$

which can be rewritten in a vector form, resulting in the following system of linear equations

$$\mathbf{A}\mathbf{w} = \mathbf{f}, \tag{17}$$

with

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} f(1) \\ \vdots \\ f(T) \end{bmatrix}, \tag{18}$$

and the $(T \times N)$ system matrix $\mathbf{A}$ defined as

$$\mathbf{A} = \begin{bmatrix} \frac{\psi_1(x(1))x(1)}{\sum_{j=1}^{N} \psi_j(x(1))} & \cdots & \frac{\psi_N(x(1))x(1)}{\sum_{j=1}^{N} \psi_j(x(1))} \\ \vdots & \vdots & \vdots \\ \frac{\psi_1(x(T))x(T)}{\sum_{j=1}^{N} \psi_j(x(T))} & \cdots & \frac{\psi_N(x(T))x(T)}{\sum_{j=1}^{N} \psi_j(x(T))} \end{bmatrix}.$$

A set of weights that solves linear equation system (17) in least square sense is then calculated using

$$\mathbf{w} = \mathbf{A}^{+}\mathbf{f}, \tag{19}$$

where $\mathbf{A}^{+}$ denotes the Moore-Penrose pseudo-inverse of the system matrix $\mathbf{A}$.

To write ν as a function phase we again use a weighted sum of Gaussian kernels

$$\nu(x) = 1 + \frac{\sum_{j=1}^{M} \theta_j \Psi_j(x)}{\sum_{j=1}^{M} \Psi_j(x)} x = 1 + \tilde{\mathbf{g}}^T(x)\boldsymbol{\theta}, \tag{20}$$

where

$$\tilde{\mathbf{g}}(x) = \begin{bmatrix} \tilde{g}_1(x) \\ \cdots \\ \tilde{g}_M(x) \end{bmatrix}, \; \tilde{g}_i(x) = \frac{\Psi_i(x)x}{\sum_{j=1}^{M} \Psi_j(x)} \tag{21}$$

To calculate the weights θ_j we first define

$$r(k) = \nu(k) - 1. \tag{22}$$

We then need to solve

$$\mathbf{B}\boldsymbol{\theta} = \mathbf{r}, \tag{23}$$

with

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_M \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} r(1) \\ \vdots \\ r(T) \end{bmatrix}, \tag{24}$$

and the system matrix $\mathbf{B} \in \mathbb{R}^{T \times M}$ defined as

$$\mathbf{B} = \begin{bmatrix} \tilde{g}_1(1) & \cdots & \tilde{g}_M(1) \\ \vdots & \vdots & \vdots \\ \tilde{g}_1(T) & \cdots & \tilde{g}_M(T) \end{bmatrix}.$$

In this way we expressed the scaling function ν as a function of phase. We can therefore rewrite Eq. (4) as

$$\tau\dot{x} = -\alpha_x x \nu(x) = -\alpha_x x + \mathbf{g}(x)^{\mathrm{T}}\boldsymbol{\theta}, \tag{25}$$

where

$$\mathbf{g}(x) = -\alpha_x x \tilde{\mathbf{g}}(x). \tag{26}$$

Eqs. (2), (3), and (25) are used when reproducing the learned trajectory.

By the above construction the rate of phase change becomes related to the speed of movement. Lower values of $\nu(x)$ correspond to slower movements and higher values to faster movements, respectively. Note the differences between the proposed approach and our previous work [20]. Here we added the term $1+$ to the definition of (20), which ensures that ν tends to 1 as x tends to 0, which is the desired behavior. We also inverted relationship between $\nu(x)$ and τ. In [20] slowing down was achieved by increasing the value of ν, which caused near singularity behavior at parts of the phase corresponding to low speed. Finally, the main difference is the initial definition of scaling factor ν. In [20], ν was initially set to 1, whereas here it is defined as the function of the normalized arc length, which ensures that the phase moves uniformly along the trajectory regardless of the current velocity of motion. This new formulation enables the transfer of speed profiles between different trajectories by simply replacing the weights of the temporal scaling function $\nu(x)$ in the DMP.

III. Reinforcement learning with directed exploration

Policy search algorithms, the type of reinforcement learning most commonly used in robotics, follow the general recipe:

1) Perform exploratory trials with the current policy,
2) Analyze the collected rewards,
3) Improve policy and repeat at step 1.

In essence, different algorithms vary in how steps 2 and 3 are implemented. Drawing from the normal distribution with mean at the current policy parameters is typically used to generate random exploration. A class of policy gradient algorithms exploits the obtained knowledge to estimate the gradient of the reward function with respect to the policy parameters. They have strict convergence properties, but in general suffer from low adaptation speed [4]. Recently, a novel class of methods gained popularity, which return directly the update as a weighted combination of exploration policies [8], [7], [21]. These algorithms employ some notion of "eliteness" of exploratory trials, from which the update is computed, while the "non-elite" trials are rejected [13].

Note that in gradient based methods, performing random exploration makes sense, as sampling of the local vicinity of the policy is needed to obtain a good gradient estimate. In the case of ranking samples by eliteness, however, every step made in the wrong direction is rejected by the algorithm. Obviously, the mapping between policy and reward is unknown in general and random exploration is a safe way to

achieve the optimal desired policy. However, in many cases, some estimate of this mapping can be obtained, either from prior knowledge, user input, reward function design, etc. Therefore, we propose to exploit this knowledge as much as possible by way of using policies inferred from exploration trials in the earlier stages of learning. Random exploration can still be used in combination to preserve the generality of the RL method.

We propose the following exploration strategy to find the optimal policy:

1) assess policies of known, similar tasks,
2) apply iterative learning control (ILC),
3) perform random exploration guided by PI^2 reinforcement learning algorithm.

The first step relies on a database of known control policies, which the agent executes and collects the corresponding rewards. After every execution, we update the policy using the RL method of choice. The RL algorithm can take care of filtering out policies that are not suitable for the task.

Once the agent runs out of prior knowledge, the exploration is changed from experience based to ILC based. The basic idea of iterative learning control (ILC, [22], [15]) is that information about the tracking error can be used to improve performance in the next repetition of the same trajectory. The general form of ILC is to update the control signal as follows

$$u(k, j+1) = \kappa(u(k, j) + \eta_1 e(k+1, j)), \qquad (27)$$

where u is the control signal, k denotes the k-th time sample, j denotes iteration, and κ and η_1 are the learning parameters. ILC is distinguished from simple feedback control by the prediction of the error $e(k+1, j)$, which serves to anticipate the error caused by the action taken at the k-th time step. ILC modifies the control input in the next iteration based on the control input and error in the previous iteration. In the context of policy learning, the error is associated with the policy cost and u is parameterized. In our case, the policy is encoded by DMP parameters defining the speed profile $\boldsymbol{\theta}$, which is mapped to the control signal $\mathbf{u} = [u(1), \ldots, u(N)]$ (see also Section IV). Mapping from DMP parameters $\boldsymbol{\theta}$ to the control signal $\mathbf{u}$ is accomplished with DMP integration (2), (3), (25). Similarly, mapping from time or phase dependant signal $\mathbf{u}$ to $\boldsymbol{\theta}$ is accomplished with regression [23], [16].

The main requirements for the application of ILC are to carefully design the error signal e in Eq. (27) and to manually tune the learning parameters κ and η_1. Note that ILC works in a controller-like fashion: at every iteration it makes a step in the direction given by the sign of the obtained error. This is hard to achieve for problems where a balance needs to be found between opposing criteria. In such cases, reinforcement learning with random exploration can find a better solution. Therefore, once we observe that ILC does not improve the cost anymore, a switch to step 3 is made.

Algorithm 1 summarizes the proposed approach, where we applied reinforcement learning algorithm PI^2 [7] for parameter update. This way, a significant increase in the speed of convergence can be achieved without sacrificing convergence properties.

The PI^2 implementation in Algorithm 1 contains two subtle adjustments compared to the original implementation described in [7]. First, we omitted the quadratic control cost term from the calculation of S, as it is not justified in our case. Penalizing high values of policy parameters may slow the trajectory down, which is the opposite of what we want to achieve. Furthermore, in the original formulation the update rule was defined as $\Delta\boldsymbol{\theta}_n = \sum_{k=1}^{N_b} P_k^b(n)\mathbf{M}_k^b(n)\boldsymbol{\epsilon}_k$, where $\boldsymbol{\epsilon}_k$ is the current exploration noise of the policy parameters. Instead, we replaced $\boldsymbol{\epsilon}_k$ with the exploration of the current parameters with respect to the k-th best known policy from the whole history of trials, i. e. $\boldsymbol{\epsilon}_k = \boldsymbol{\theta} - \boldsymbol{\theta}_k^b$. In this way we can perform the policy update after only one trial, as opposed to the standard implementation of the PI^2 algorithm which requires several trials before the policy can be updated.

IV. Learning of speed profile using ILC

We evaluate the proposed approach on the problem of learning speed profiles paramaterized as described in Section II, which enables easy transfer of speed profiles between different tasks.

The goal of modifying the speed profile is to accelerate the task execution without degrading the overall performance of the task. A similar idea was exploited in [17] in order to gradually increase the speed of an assembly task until contact forces remained within the prescribed tolerances. The idea is simple: increase the speed of task execution until some essential task constrains are violated. These task constrains together with policy execution time determine the error function $e(x)$, used for parameter update with ILC, as well as the cost function used for RL. However, it is important to note that the two criteria are not the same. RL algorithms require unsigned value to determine the cost of an experiment, while ILC operates with signed error function.

A suitable error function for ILC to adapt the speed of motion is given by

$$e(x) = \xi_\nu(\nu_{\max} - \nu(x)) - \xi_d b(x), \qquad (28)$$

where ν is the previously defined temporal scaling function, b is a scalar function quantifying deviations from the prespecified task constraints, $\nu_{\max} > 1$ is the upper bound for the temporal scaling factor, x is the phase along the trajectory at time t, $x = x(t)$, and ξ_ν, ξ_d are the corresponding weighting factors. Function b describing deviations from task constraints might be anything from a norm of excessive forces and torques as in [17] to a signal indicating the spilling of liquid as in this paper.

As noted in [17], standard ILC approach cannot be applied to the trajectory speed learning in time domain, since one of the assumptions of ILC is that each trial has the same number of samples [15]. Therefore, in [17] the ILC was implemented in phase domain, where the phase signal was anticipated for a fixed phase offset. Since mapping from the phase to the displacement is not linear, this choice is not optimal. Instead, we propose the following ILC-type algorithm to realize the learning of the temporal scaling function

$$\nu(x_n, j+1) = \kappa(\nu(x_n, j) + \eta_1 e(x_n + \delta_{x_n}, j), \qquad (29)$$

Algorithm 1: PI^2 algorithm augmented with prior knowledge and ILC directed exploration.

Input: parameterized policy $\boldsymbol{\theta}$ (Eq. (20) and (25))
basis functions $\boldsymbol{g}$ for parameterization (Eq. (26))
intermediate cost functions $r(x_n), n = 1, \ldots, N$
terminal cost function $r(x_{N+1})$
ILC error function $e(x_n)$
learning parameters $\boldsymbol{\Sigma}$, N_b, κ, η_1, δ_s
initial approximation $\boldsymbol{\theta}_1$
knowledge base containing policies $\boldsymbol{\theta}^{kb}_1, \ldots, \boldsymbol{\theta}^{kb}_{N_{kb}}$
Output: optimal policy $\boldsymbol{\theta}_{opt}$

$use\_random = false$
for $i = 1$: max iterations **do**
  calculate exploration step:
  if $i < N_{kb}$ **then**
    $\boldsymbol{\theta} = \boldsymbol{\theta}^{kb}_i$
  else
    if C *decreasing* **and** $use\_random = false$ **then**
      calculate trajectory using ILC:
      for $n = 1 : N$ **do**
        $u(x_n, i) =$
        $\kappa(u(x_n, i-1) + \eta_1 e(x_n + \delta_{x_n}, i-1))$
        find $\boldsymbol{\theta}$ which parameterizes $\boldsymbol{u}_i$ with $\boldsymbol{g}$
    else
      use random exploration policy:
      draw $\boldsymbol{\theta}$ from $\mathcal{N}(\boldsymbol{\theta}_i, \boldsymbol{\Sigma})$
      $use\_random = true$

  perform exploration experiment using $\boldsymbol{\theta}$ and collect costs $r(x_n)$, $n = 1, \ldots, N+1$

  calculate total cost $C = \sum_{n=1}^{N+1} r(x_n)$
  sort all past trials by C so that $\boldsymbol{\theta}^b_k$ is k-th best policy and r^b_k its return, i. e. C
  if $i < N_b$ **then**
    $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_1$
  else
    update policy using algorithm PI^2:
    for $n = 1 : N$ **do**
      for $k = 1 : N_b$ **do**
        $S^b_k(n) = r^b_k(N+1) + \sum_{j=n}^{N} r^b_k(j)$
        $P^b_k(n) = \dfrac{\exp(-S^b_k(n)/\lambda)}{\sum_{m=1}^{N_b} \exp(-S^b_m(n)/\lambda)}$
        $\mathbf{M}^b_k(n) = \dfrac{\boldsymbol{g}(n)^b_k \boldsymbol{g}(n)^{b\,T}_k}{\boldsymbol{g}(n)^{b\,T}_k \boldsymbol{g}(n)^b_k}$
      $\Delta\boldsymbol{\theta}_n = \sum_{k=1}^{N_b} P^b_k(n) \mathbf{M}^b_k(n)(\boldsymbol{\theta} - \boldsymbol{\theta}^b_k)$
    for $j = 1 : M$ **do**
      $\delta\theta_j = \dfrac{\sum_{n=1}^{N} (N+1-n) g_j(n) \Delta\theta_{n,j}}{\sum_{n=1}^{N} g_j(n)(N+1-n)}$
    $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + [\delta\theta_1, \ldots, \delta\theta_M]^{\mathrm{T}}$
$\boldsymbol{\theta}_{opt} = \boldsymbol{\theta}_i$

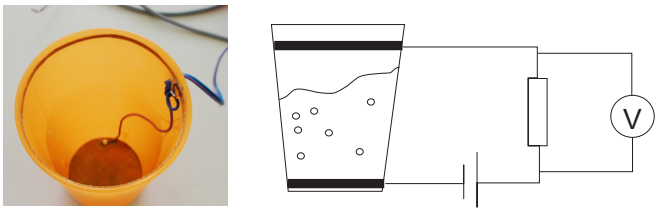


Fig. 1. Sensory system used to detect spilling. The circuit was powered with a 9 volt battery and voltage measured on a 10 MΩ resistor. When the water closes the circuit, the resistor takes an overwhelming part of the voltage drop and the reading increases from 0 to 9 V.

where x_n denotes the phase, k is the sampling index, j is the learning iteration index, and κ and η_1 are the manually selected ILC gains. The error signal $e(x_n + \delta_{x_n}, j)$ is used to anticipate the deviation from the desired behavior, with δ_{x_n} denoting the magnitude of the step in the phase domain from which the prediction is calculated. δ_{x_n} can be obtained by calculating the derivative of (8)

$$\delta_{x_n} = -\frac{\alpha_x}{\tau} \exp\left(-\frac{\alpha_x}{\tau} s(t_n)\right) \nu(t_n) \delta_t, \tag{30}$$

where $\delta_t = (t_T - t_1)/N$ and $t_n = (t_T - t_1)(n-1)/N$, $n = 1, \ldots, N$. N is the parameter chosen by the user and defines the number of steps in the ILC algorithm.

V. Experimental evaluation

The proposed methodology was used for speeding up the task of delivering a glass full of liquid from a tray onto a table. The goal of adaptation was to speed up the execution time as much as possible, without spilling the liquid. This is an example of a process, for which it would be very difficult to find a model-based solution. Our experimental setup consisted of an antropomorphic Kuka LWR arm with FRI interface and a three finger Barret hand, shown in Fig. 3. The desired path of the robot was demonstrated using kinesthetic guiding. The cup was equipped with a resistance sensor to detect spilling (see Figure 1). Whenever the level of the liquid reached the conductive ring at the edge of the cup, an increase in voltage was detected.

The intermediate cost function was defined as

$$r(x) = \begin{cases} 0 & \text{if } U(x) < U_0 \\ \gamma & \text{if } U(x) \geq U_0 \end{cases}, \tag{31}$$

where $r(x)$ denotes an intermediate cost at phase x, $U(x)$ the current voltage, U_0 the threshold voltage, which signifies spilling, and γ is a positive constant. In our experiment we used $\gamma = 10$, $U_0 = 5V$. The terminal cost was defined as duration of the trajectory in seconds multiplied by scaling factor 50.

Prior knowledge consisted of several recordings of two similar tasks, shown in Fig. 2: first, placing a cup of water from a tray onto a table, and second, inserting a small cubical object (resembling a small candy) into an ice cube tray (resembling a candy box). Even though the cup moving task seems similar to the assigned robot's task, the path by which the movement was done was substantially different from

the one demonstrated to the robot; therefore, just directly transferring the recorded speed profile is not feasible.

The human demonstrated motion was captured using NaturalPoint Optitrack motion capture system with passive markers. In total, 5 trajectories were captured: 3 for cup moving and 2 for cube moving. We encoded them into DMPs as presented in Section II.

The speed profiles (20) gathered from the demonstrated motions were then used as initial exploration policies in the first stage of learning. Figure 4 shows convergence results. The first five data points (red shaded area) correspond to the costs obtained during execution of these policies.

Next, starting from sixth iteration, policies were calculated and executed using ILC. Error function (28) was finalized by monitoring the spilling sensor voltage

$$b(x) = U(x) - U_0, \tag{32}$$

where $U(x)$ denotes the sensor's output voltage and U_0 denotes the threshold voltage. For $U(x)$ smaller than U_0, signifying no spilling, b will be negative and will cause the trajectory at the corresponding phase to speed up. Conversely, $U(x)$ higher than U_0 slows down the motion. The speed profile for the next iteration ν_{j+1} was obtained by Eq. (29), with constants chosen as $\kappa = 1$ and $\eta_1 = 0.01$. The policy update $\Delta\boldsymbol{\theta}_{j+1}$ was then obtained by parameterizing ν_{j+1} as per Eq. (20) and calculating the difference to the current policy provided by reinforcement learning as explained in Section III.

As shown in Fig. 4, the policy improves quickly. However, the main drawback of ILC is that it cannot efficiently find a balance between opposing effects. Imagine that during an episode, no liquid spilling is detected. Therefore, for the next learning iteration, ILC will generate a faster trajectory. As a result, some liquid will spill and in the next iteration,

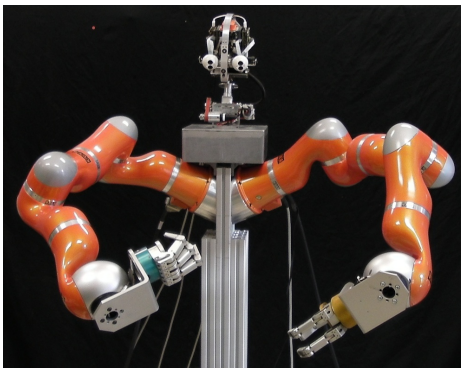


Fig. 3. Upper body humanoid system used in the experiments.

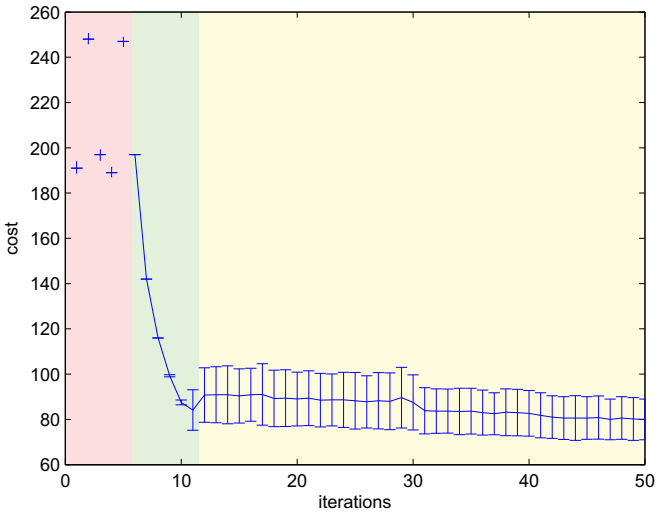


Fig. 4. Convergence of the learning process. The bars show standard deviation of 8 learning trials. The red shaded part of the graph shows costs of trials using policies from prior knowledge. The green shaded area corresponds to learning where exploration was performed using ILC, while the yellow shaded area corresponds to final, random based exploration.

Fig. 2. Recording trajectories for prior knowledge. Top: placing a cube into a compartment. Bottom: placing a cup on a table.

the movement will be slower again, catching the learning process in an oscillatory cycle. This is due to the controller-like nature of ILC - it tries to reduce the error signal to zero, which is sometimes not possible to achieve.

Thus, after 11th iteration, the cost has been observed to stop converging. Random exploration is then used to perform fine tuning of the policy.

VI. Conclusions

In this paper we presented a novel method for autonomous learning of control policies. Reinforcement learning methods typically use random exploration to obtain information about reward fluctuation in the local vicinity of the policy. We proposed to supplement random exploration by prior knowledge and iterative learning control (ILC). This way, prior knowledge about promising search directions is inserted into the system, which is followed by a few iterations of ILC. Since ILC's convergence properties are not as strong as those of RL, the algorithm stops converging at some point, and the exploration switches to standard random exploration.

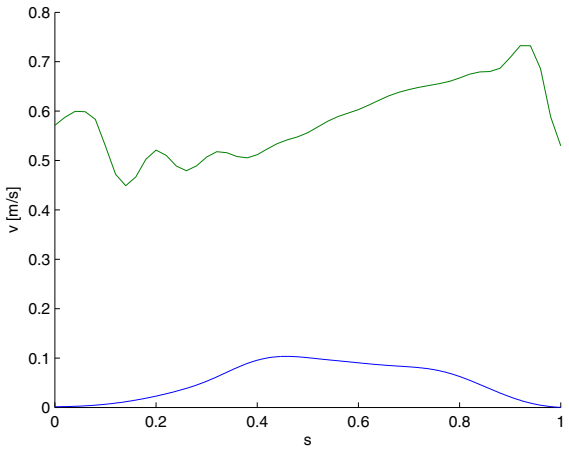


Fig. 5. Result of the learning process. Blue line shows speed of the initially demonstrated task. Green line shows an example of a learned speed profile after 50 learning trials. Note that the profiles are plotted against normalized arc length s, not time.

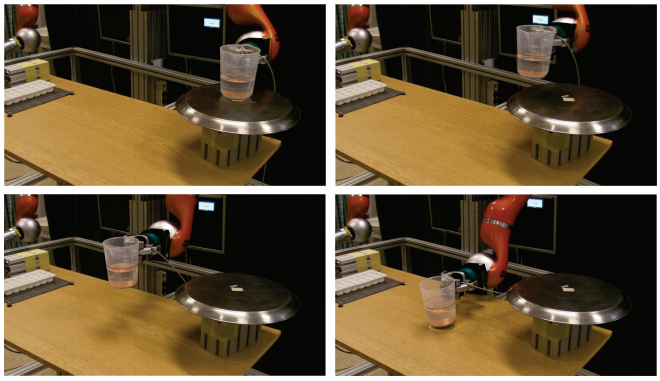


Fig. 6. In this experiment the robot moves the cup from the tray onto the table. The path was obtained using kinesthetic guiding and the speed profile (shown in Fig. 5) was learned using our novel approach.

This way we retain the best of both worlds, the initial fast convergence of ILC and fine tuning achievable by RL.

In our experiments we focused on trajectory speed adaptation. For this purpose we developed a new formulation for speed profile parameterziation within a DMP framework. Our experimental results showed that the proposed approach is effective at learning high precision tasks such as fast transfer of a cup full of liquid without spilling.

REFERENCES

[1] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[2] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109–116, 2004.

[3] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, 2008, pp. 1371–1394.

[4] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, pp. 682–697, 2008.

[5] M. Tamosiunaite, B. Nemec, A. Ude, and F. Wörgötter, "Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives," *Robotics and Autonomous Systems*, vol. 59, no. 11, pp. 910–922, 2011.

[6] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotic Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[7] E. A.Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, vol. 11, pp. 3137–3181, 2010.

[8] J. Kober and J. Peters, "Policy search for motor primitives in robotics," in *Advances in Neural Information Processing Systems 21 (NIPS)*, Vancouver, B. C., Canada, 2008, pp. 852–859.

[9] P. Kormushev, S. Calinon, R. Saegusa, and G. Metta, "Learning the skill of archery by a humanoid robot iCub," in *IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, 2010, pp. 352–357.

[10] D. H. Grollman and A. Billard, "Donut as I do: Learning from failed demonstrations," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 3804–3809.

[11] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Autonomous Robots*, vol. 33, no. 4, pp. 361–379, 2012.

[12] B. Nemec, R. Vuga, and A. Ude, "Efficient sensorimotor learning from multiple demonstrations," *Advanced Robotics*, vol. 27, no. 13, pp. 1023–1031, 2013.

[13] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," in *29th International Conference on Machine Learning*, Edinburgh, UK, 2012, pp. 281–288.

[14] B. Nemec, R. Vuga, and A. Ude, "Exploiting previous experience to constrain robot sensorimotor learning," in *IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, 2011, pp. 727–723.

[15] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems*, vol. 26, no. 3, pp. 96–114, 2006.

[16] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[17] B. Nemec, F. Abu-Dakka, J. A. Jørgensen, T. R. Savarimuthu, B. Ridge, J. Jouffroy, H. G. Petersen, N. Krüger, and A. Ude, "Transfer of assembly operations to new workpiece poses by adaptation to the desired force profile," in *International Conference on Advanced Robotics*, Montevideo, Uruguay, 2013.

[18] H. Wang, J. Kearney, and K. Atkinson, "Arc-length parameterized spline curves for real-time simulation," in *5th International Conference on Curves and Surfaces*, San Malo, France, 2002, pp. 387–396.

[19] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipies; The Art of Scientific Computing*. Cambridge: Cambridge University Press, 2007.

[20] B. Nemec, A. Gams, and A. Ude, "Velocity adaptation for self-improvement of skills learned from user demonstrations," in *IEEE-RAS International Conference on Humanoid Robots*, Atlanta, Georgia, USA, 2013, pp. 423–428.

[21] V. Heidrich-Meisner and C. Igel, "Neuroevolution strategies for episodic reinforcement learning," *Journal of Algorithms*, vol. 64, no. 4, pp. 152–168, 2009.

[22] K. L. Moore, Y. Chen, and H.-S. Ahn, "Iterative learning control: A tutorial and big picture view," in *IEEE Conference on Decision and Control*, San Diego, CA, 2006, pp. 2352–2357.

[23] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

Implicit Learning of Simpler Output Kernels for Multi-Label Prediction

Hanchen Xiong Sandor Szedmak Justus Piater
Institute of Computer Science, University of Innsbruck
Innsbruck, A-6020, Austria
{*hanchen.xiong, sandor.szedmak, justus.piater*}*@uibk.ac.at*

Abstract

It has been widely agreed that, in multi-label prediction tasks, capturing and utilizing dependencies among labels is quite critical. Therefore, a research tendency in multi-label learning is that increasingly more sophisticated dependency structures on labels (*e.g.* output kernels) are proposed. We show that, however, over-complex dependency structures will harm more than help learning when the underling dependency is relatively weak. To avoid overfitting on structures, a regularization on label-dependency is desirable. In this paper, we put forward a novel *joint-SVM* for multi-label learning. Compared to other discriminative learning schemes, joint-SVM has two strengths: at first, the complexity of training joint-SVM is almost the same as training a single regular SVM, which is quite efficient; secondly, in joint-SVM, a linear output kernel on multi-label is implicitly learned and a regularization on the output kernel is implicitly added, which enhances generalization ability. In our experimental results on image annotation, joint-SVM compares favorably state-of-the-arts methods.

1 Predict Multi-label as Structured Outputs

In the past two decades, support vector machines (SVMs) have displayed remarkable successes in various application domains. The achievements of SVMs mainly stems from its two advantageous components: *maximum margins* and *input kernels*. The maximum-margin principle is a reflection of statistical learning theory [12] on linear binary classification. Kernels provide powerful mechanisms enabling the linear classifier to separate highly non-linear data. The critical observation of kernel methods is that a kernel function can be defined on a pair of data instances to implicitly map them to a reproducing kernel Hilbert space (RKHS):

$$K_\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \langle \phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}^{(j)}) \rangle \tag{1}$$

where $\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathbb{R}^d$ are two input training instances, ϕ is the feature map induced by kernel function K_ϕ, and $\phi(\mathbf{x}^{(i)})$ is the representation of $\mathbf{x}^{(i)}$ in the RKHS $\mathcal{H}_\phi$. Given the training dataset $\{\mathbf{x}^{(i)} \in \mathbb{R}^d, y^{(i)} \in \{+1, -1\}\}_{i=1}^m$, the primal form of training SVM is:

$$\begin{aligned} \arg \min_{\mathbf{w} \in \mathbb{R}^{\mathcal{H}_\phi}} \quad & \tfrac{1}{2}||\mathbf{w}||^2 + C \textstyle\sum_{i=1}^m \xi^{(i)} \\ \text{s.t.} \quad & y^{(i)} \left(\mathbf{w}^\top \phi(\mathbf{x}^{(i)}) \right) \geq 1 - \xi^{(i)}, \xi^{(i)} \geq 0, i \in \{1, \ldots, m\} \end{aligned} \tag{2}$$

where $\mathbf{w}$ is a linear hyperplane in $\mathcal{H}_\phi$, $\xi^{(i)}$ are slack variables for the tolerance of noise, and C is a trade-off parameter. (2) differs from usual SVM formulation slightly at the absence of a bias term. Here we ignore the bias since it can be absorbed in $\mathbf{w}$. The computational advantage of kernels become obvious when the primal form of SVM (2) is reformulated to its dual form:

$$\begin{aligned} \arg \min_{\alpha_1, \alpha_2, \ldots, \alpha_m} \quad & \textstyle\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y^{(i)} y^{(j)} K_\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \\ \text{s.t.} \quad & \forall i, 0 \leq \alpha_i \leq C \end{aligned} \tag{3}$$

The dual representation of $\mathbf{w}$ is $\sum_{i=1}^{m} \alpha_i y^{(i)} \phi(\mathbf{x}^{(i)})$, and thus the prediction of a test instance $\hat{\mathbf{x}}$ is

$$\hat{y} = \text{sgn}\left(\mathbf{w}^\top \phi(\hat{\mathbf{x}})\right) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i y^{(i)} K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}})\right). \tag{4}$$

We can denote $y^{(i)}\left(\mathbf{w}^\top \phi(\mathbf{x}^{(i)})\right)$ in the constraints of (2) as a score function $F(\mathbf{x}^{(i)}, y^{(i)}; \mathbf{w})$, then for binary outputs $y^{(i)}$, $F\left(\mathbf{x}^{(i)}, y^{(i)}; \mathbf{w}\right) - F\left(\mathbf{x}^{(i)}, -y^{(i)}; \mathbf{w}\right) = 2 \times F\left(\mathbf{x}^{(i)}), y^{(i)}; \mathbf{w}\right)$. Also, a distance function between binary outputs can be denoted as $d(y^{(i)}, -y^{(i)}) = |y^{(i)} - (-y^{(i)})| = 2$. Then by replacing C with $\frac{C}{2}$, (2) can be rewritten as:

$$\begin{aligned} \arg \min_{\mathbf{w} \in \mathbb{R}^{\mathcal{H}_\phi}} \quad & \tfrac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} \xi^{(i)} \\ \text{s.t.} \quad & \forall i, \underbrace{F\left(\mathbf{x}^{(i)}, y^{(i)}; \mathbf{w}\right) - F\left(\mathbf{x}^{(i)}, -y^{(i)}; \mathbf{w}\right)}_{\Delta_F(y^{(i)}, -y^{(i)})} \geq d(y^{(i)}, -y^{(i)}) - \xi^{(i)}, \xi^{(i)} \geq 0 \end{aligned} \tag{5}$$

which is a binary-output case of structural SVM [11] (see later). By using hinge-loss representation for $\xi^{(i)}$, (5) is:

$$\arg \min_{\mathbf{w} \in \mathbb{R}^{\mathcal{H}_\phi}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} \max\{0, d(y^{(i)}, -y^{(i)}) - \Delta_F(y^{(i)}, -y^{(i)})\} \tag{6}$$

Structural SVM [11] is an extension of SVM for structured-outputs, in which, however, the margin to be maximized is defined as the score gap between the desired output and the runner-up. Assume that structured outputs $\mathbf{y} \in \mathcal{Y}$, and the score function is linear in some *combined feature representation* of inputs and outputs $\mathbf{\Psi}(\mathbf{x}, \mathbf{y})$: $F(\mathbf{x}, \mathbf{y}; \mathbf{W}) = \langle \mathbf{W}, \mathbf{\Psi}(\mathbf{x}, \mathbf{y}) \rangle$, then the objective function of structural SVM is:

$$\arg \min_{\mathbf{W} \in \mathbb{R}^{\Psi}} \frac{1}{2}||\mathbf{W}||^2 + C\sum_{i=1}^{m} \max_{\mathbf{y}' \in \mathcal{Y}} \left\{d(\mathbf{y}^{(i)}, \mathbf{y}') - \Delta_F(\mathbf{y}^{(i)}, \mathbf{y}')\right\} \tag{7}$$

where $\Delta_F(\mathbf{y}^{(i)}, \mathbf{y}') = F(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}; \mathbf{W}) - F(\mathbf{x}^{(i)}, \mathbf{y}'; \mathbf{W})$ and $d(\mathbf{y}^{(i)}, \mathbf{y}')$ is a distance function defined on structured outputs. In multi-label scenario, given a set of T labels, then outputs are T-dimensional binary vector $\mathbf{y} = [y_1, \cdots, y_t, \cdots, y_T]^\top \in \mathbb{B}^T$. When we define the score function $F\left(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}; \mathbf{W}\right) = \langle \mathbf{W}, \phi(\mathbf{x}^{(i)}) \otimes \mathbf{y}^{(i)} \rangle$, and use *Hamming distance* on outputs, then because of linear decomposability, (7) can be rewritten as:

$$\begin{aligned} \arg \min_{\mathbf{W} \in \mathbb{R}^{\mathcal{H}_\phi \times \mathbb{R}^T}} \quad & \tfrac{1}{2}||\mathbf{W}||_F^2 + C\sum_{i=1}^{m}\sum_{t=1}^{T} \max_{y'_t = \{-1,+1\}} \left\{d(y_t^{(i)}, y'_t) - \Delta_F(y_t^{(i)}, y'_t)\right\} \\ \Downarrow \\ \arg \min_{\mathbf{w}_1, \cdots, \mathbf{w}_T \in \mathbb{R}^{\mathcal{H}_\phi}} \quad & \sum_{t=1}^{T} \left\{\tfrac{1}{2}||\mathbf{w}_t||^2 + C\sum_{i=1}^{m} \max\left\{0, d(y_t^{(i)}, -y_t^{(i)}) - \Delta_F(y_t^{(i)}, -y_t^{(i)})\right\}\right\} \end{aligned} \tag{8}$$

where $\langle \cdot, \cdot \rangle_F$ denotes Frobenius product and $||\mathbf{W}||_F$ is the Frobenius norm of matrix $\mathbf{W}$.

2 Joint SVM

It can be seen (by linking (6) and (8)) that, with linearly decomposable score functions and output distances, using structural SVM on multi-label learning is equivalent to learning T SVMs jointly. This is closely related to multi-task learning frameworks [1], where different learning tasks are connected by summing up their objectives and constraints respectively:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\sum_{t=1}^{T} ||\mathbf{w}_t||^2 + C\sum_{t=1}^{T}\sum_{i=1}^{m} \xi_t^{(i)} \\ \text{w.r.t.} \quad & \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_T \in \mathbf{R}^{\mathcal{H}_\phi \times 1} \\ \text{s.t.} \quad & \sum_{t=1}^{T} y_t^{(i)}\left(\mathbf{w}_t^\top \phi(x^{(i)})\right) \geq T - \sum_{t=1}^{T} \xi_t^{(i)} \end{aligned} \tag{9}$$

By denoting $\mathbf{y}^{(i)} = [y_1^{(1)}, \ldots, y_T^{(i)}]$, and $\mathbf{W} = [\frac{\mathbf{w}_1^\top}{T}; \ldots; \frac{\mathbf{w}_T^\top}{T}]^\top$, we can rewrite (9) as:

$$\begin{aligned} \arg \min_{\mathbf{W} \in \mathbb{R}^T \times \mathcal{H}_\phi} \quad & \tfrac{1}{2}||\mathbf{W}||_F^2 + C\sum_{i=1}^{m} \bar{\xi}^{(i)} \\ \text{s.t.} \quad & \left\langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)}) \right\rangle \geq 1 - \bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1, \ldots, m\} \end{aligned} \tag{10}$$

which is referred to as *joint SVM*. When linear output kernels ($K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) = \langle \psi(\mathbf{y}^{(i)}), \psi(\mathbf{y}^{(j)}) \rangle$) [4, 7, 13] are applied on outputs, (10) will be:

$$\begin{aligned} \arg \min_{\mathbf{W} \in \mathbb{R}^{H_\psi \times \mathcal{H}_\phi}} \quad & \tfrac{1}{2}||\mathbf{W}||_F^2 + C \sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} \quad & \left\langle \psi(\mathbf{y}^{(i)}), \mathbf{W}\phi(x^{(i)}) \right\rangle \geq 1 - \bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1, \dots, m\} \end{aligned} \tag{11}$$

Since the linear decomposability of $\Delta_F(\mathbf{y}^{(i)}, \mathbf{y}')$ is still preserved, join SVM solves the same problem as structural SVM. However, one strength of joint SVM is that its training complexity is almost the same as a single SVM, by contrast to the exponential complexity in structural SVM. Similarly to regular SVM, joint SVM can be converted to its dual form

$$\begin{aligned} \arg \min_{\alpha_1, \cdots, \alpha_m} \quad & \textstyle\sum_{i=1}^m \alpha_i - \sum_{i,j=1}^m \alpha_i \alpha_j K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) K_\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \\ \text{s.t} \quad & \forall i, 0 \leq \alpha_i \leq C \end{aligned} \tag{12}$$

with $\mathbf{W} = \sum_i^m \alpha_i \psi(\mathbf{y}^{(i)}) \phi(\mathbf{x}^{(i)})^\top$. It can be seen that, with the kernel matrix on outputs precomputed, the computational complexity of joint SVM (12) is the same as the learning of one single SVM (3), which is a great advantage in efficiency. Meanwhile, when more general output kernels are used, then the linear decomposability of $\Delta_F(\mathbf{y}^{(i)}, \mathbf{y}')$ will be violated, then joint SVM becomes a special case of max-margin regression [10], which seeks to learn linear operators $\mathbf{W} : \mathcal{H}_\phi \to \mathcal{H}_\psi$ from general $\phi(\mathbf{x}) \otimes \psi(\mathbf{y})$.

Given a test input $\hat{\mathbf{x}}$, the prediction $\psi(\hat{\mathbf{y}})$ in $\mathcal{H}_\psi$ is

$$\psi(\hat{\mathbf{y}}) = \mathbf{W}\phi(\hat{\mathbf{x}}) = \sum_{i=1}^m \alpha_i \psi(\mathbf{y}^{(i)}) K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}}). \tag{13}$$

Meanwhile, there is no direct way (say, by inverting Eq.(13)) to map $\psi(\hat{\mathbf{y}})$ back to $\hat{\mathbf{y}}$. Therefore, we can find the optimal solution $\hat{\mathbf{y}}^*$, out of all possible $\mathbf{y} \in \{+1, -1\}^T$, such that its projection in $\mathcal{H}_\psi$ is closest to $\mathbf{W}\phi(\hat{\mathbf{x}})$:

$$\begin{aligned} \hat{\mathbf{y}}^* &= \text{argmax}_{\mathbf{y} \in \{+1,-1\}^T} \langle \psi(\mathbf{y}), \mathbf{W}\phi(\hat{\mathbf{x}}) \rangle \\ &= \text{argmax}_{\mathbf{y} \in \{+1,-1\}^T} \textstyle\sum_{i=1}^m \alpha_i \underbrace{K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}})}_{\beta_i} K_\psi(\mathbf{y}^{(i)}, \mathbf{y}) \end{aligned} \tag{14}$$

In general, there is no closed-form solution to Eq.(14), so here we use a similar neighbour-based label transferring theme as [9, 6]:

$$\hat{\mathbf{y}}^* = \left(\sum_{k=1}^K \mathbf{y}^{(k)} w_k \right) \Big/ \sum_{k=1}^K w_k \qquad w_j = \sum_{i=1}^m \alpha_i \beta_i K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) \tag{15}$$

where $k = \{j \in [1, m] : w_j > 0\}$ and maximum $K = 10$ neighbours are taken into account. Since α_i are $K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)})$ were already computed in the training phase, only the computation of $\{\beta_i\}_{i=1}^m$ is needed during testing. Thus, the complexity in predicting is $\mathcal{O}(m)$.

3 Implicit Learning and Regularization of Output Kernels

Assume that the statistics of tags' pairwise co-occurrence can be encoded in a $T \times T$ matrix $\mathbf{P}$[3, 4, 7, 13], via which the output vectors can be linearly mapped as $\psi(\mathbf{y}) = \mathbf{P}\mathbf{y}$, and thus the corresponding linear output kernel is:

$$K_\psi^{Lin}(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) = \mathbf{y}^{(i)\top} \mathbf{\Omega} \mathbf{y}^{(j)} \tag{16}$$

where $\mathbf{\Omega} = \mathbf{P}^\top \mathbf{P} = \mathbf{P}\mathbf{P}^\top$. By denoting $\mathbf{U} = \mathbf{P}^\top \mathbf{W}$, we can rewrite joint SVM (11) as:

$$\begin{aligned} \arg \min_{\mathbf{W} \in \mathbb{R}^{H_\psi \times \mathcal{H}_\phi}} \quad & \tfrac{1}{2}||\mathbf{W}||_F^2 + C \sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} \quad & \left\langle \mathbf{y}^{(i)}, \mathbf{U}\phi(x^{(i)}) \right\rangle \geq 1 - \bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1, \dots, m\} \end{aligned} \tag{17}$$

Meanwhile, we need to control the scale of $\mathbf{P}$, otherwise the constraints in (17) will be pointless. Different regularizations on $\mathbf{P}$ have been proposed in previous work. In [4] one extra regularization on $\mathbf{\Omega}$, $\frac{1}{2}||\mathbf{\Omega}||_F^2$, was added into the objective function, while $||P||_F = 1$ was used in [13]. By

	Corel5K			Espgame			Iaprtc12		
Method	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
MBRM [5]	24.0	25.0	24.0	18.0	19.0	18.0	24.0	23.0	23.0
JEC [9]	27.0	32.0	29.0	24.0	19.0	21.0	29.0	19.0	23.0
TagProp [6]	33.0	42.0	37.0	39.0	27.0	**32.0**	45.0	**34.0**	**39.0**
FastTag [3]	32.0	**43.0**	37.0	**46.0**	22.0	30.0	**47.0**	26.0	34.0
JSVM	**48.5**	38.0	**42.6**	32.7	**31.6**	**32.2**	42.2	29.4	34.6
JSVM+Pol(2)	46.6	37.0	41.3	32.6	24.4	27.9	37.9	26.6	31.2
JSVM+Pol(3)	41.5	31.3	35.7	28.5	21.3	24.4	38.0	26.1	31.0

Table 1: Comparison between different versions of joint SVM and other related methods on three benchmark databases. P, R and F1 denote precision, recall and F1 measure respectively.

contrast, a pseudo regularization on $\mathbf{P}$ is used in [3] via the re-construction loss from manually-corrupted data and $\mathbf{P}$. Similar to [4], we want to add a regularizer to control overfitting from output dependency-structures. Meanwhile, by merging regularization on $\mathbf{W}$ and $\mathbf{P}$, we obtain a more compact regularizer, $\frac{1}{2}\mathbf{W}^\top\mathbf{\Omega}\mathbf{W}$, resulting in:

$$\begin{aligned} \arg \min_{\mathbf{U}\in\mathbb{R}^{H_\psi\times\mathcal{H}_\phi}} \quad & \tfrac{1}{2}||\mathbf{U}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} \quad & \left\langle \mathbf{y}^{(i)}, \mathbf{U}\phi(x^{(i)})\right\rangle \geq 1-\bar{\xi}^{(i)}, \xi_i \geq 0, i\in\{1,\ldots,m\} \end{aligned} \tag{18}$$

Remarkably, (18) is equivalent to (11) with $\mathbf{W}$ substituted by $\mathbf{U}$, which suggests that a linear output kernel is implicitly learned, and absorbed in $\mathbf{W}$, when we training a plain joint SVM with no explicit kernel on outputs. In addition, a regularization on the output kernel is also implicitly added.

4 Experiments

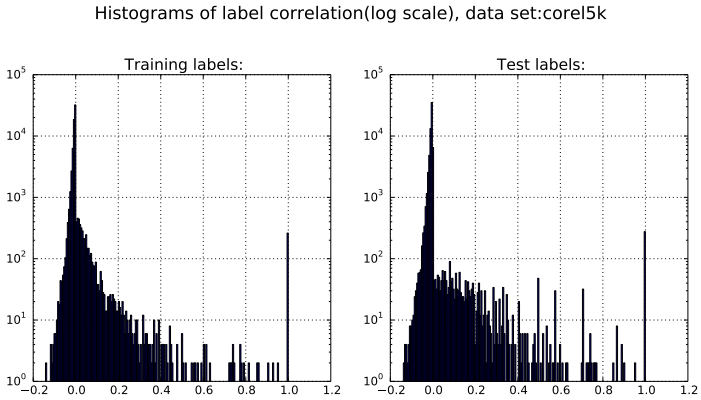
In our experiments, we evaluated the propose joint SVM on image annotation tasks. Here, we used three benchmark datasets, Corel5k, Espgame and Iaprtc12. These three datasets have been widely used in image annotation studies [8, 2, 5, 6, 9, 3] with performance evaluations reported therein. Therefore, we can easily compare our method with others. We used the same visual features as in [6, 3]. Three types of joint SVMs with different output kernels are tested: plain joint SVM (JSVM), 2-degree polynomial (JSVM+Pol(2)) and 3-degree polynomial (JSVM+Pol(3)).

The experimental results, together with the reported results from other related work, are presented in Table 1. We can see that plain joint SVM (JSVM) outperforms all other results on Corel5k and Espgame datasets. JSVM is also the second best result on Iaprtc12 dataset. JSVM+Pol(2) also worked better than some old methods [5, 9]. Meanwhile, JSVM+Pol(3) is worse than JSVM+Pol(2).

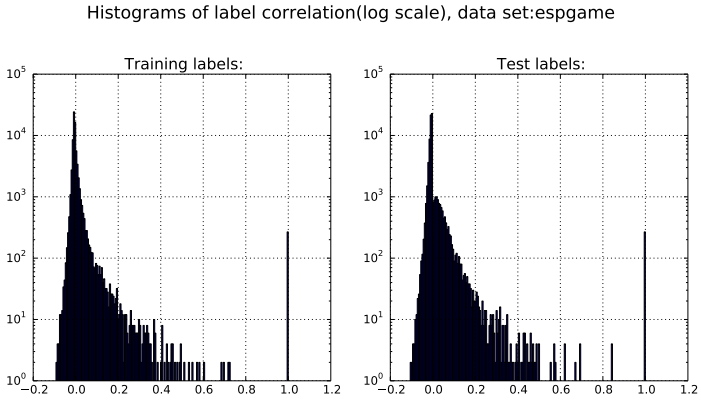
Discussions Based on our experiments, it seems that plain joint SVM (JSVM) works more robustly than the joint SVMs with explicit output kernels. In order to dig deeper to find an explanation, we can study the correlation matrices of output tag-sets in three datasets. In Figure 1, for each dataset, we plot the histograms (in log scale) of all correlation values in both training sets and testing sets. We found that most entries in correlation matrices are 0, which means that the pairwise correlation (or roughly speaking, dependencies) is rather sparse. Although JSVM, JSVM+Pol(2) both encode pairwise dependencies, it should be reminded that the implicit linear output kernel in JSVM is in regularization term, which implies that simpler output kernels (dependencies) are encouraged. However, JSVM+Pol(2) does not have this preference. Therefore, JSVM can implicitly learned most simple output kernels when no more complex ones are needed. Analogously, the same principle can explain why even JSVM+Pol(3) led to worse results.

5 Conclusions

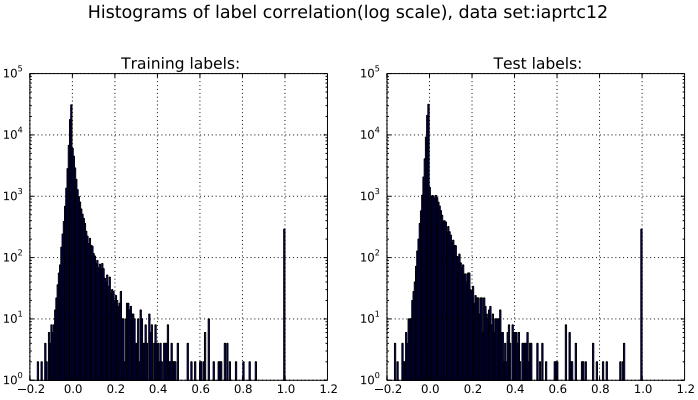
A novel joint SVM was presented for multi-label learning. One benefit of using joint SVM is that the learning and regularization of a linear output kernel are implicitly conducted. Moreover, both training joint SVM and predicting with joint SVM are efficient. As a possible work direction, we might investigate more interesting output kernel regularization schemes to fit different applications.



(a)



(b)



(c)

Figure 1: The histograms (in log scale) of all correlation values in both training sets and testing sets: (a) Corel5k, (b) Espgame (c) Iartc12.

Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

References

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[2] David M. Blei and Michael I. Jordan. Modeling annotated data. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, 2003.

[3] Minmin Chen, Alice Zheng, and Kilian Q. Weinberger. Fast image tagging. In *ICML*, 2013.

[4] Francesco Dinuzzo, Cheng Soon Ong, Peter V. Gehler, and Gianluigi Pillonetto. Learning output kernels with block coordinate descent. In *ICML*, 2011.

[5] S. L. Feng, R. Manmatha, and V. Lavrenko. Multiple bernoulli relevance models for image and video annotation. In *CVPR*, 2004.

[6] Matthieu Guillaumin, Thomas Mensink, Jakob Verbeek, and Cordelia Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, 2009.

[7] Bharath Hariharan, S. V. N. Vishwanathan, and Manik Varma. Efficient max-margin multi-label classification with applications to zero-shot learning. *Machine Learning*, 88(1-2):127–155, 2012.

[8] Victor Lavrenko, R. Manmatha, and Jiwoon Jeon. A model for learning the semantics of pictures. In *NIPs*. 2004.

[9] Ameesh Makadia, Vladimir Pavlovic, and Sanjiv Kumar. Baselines for image annotation. *International Journal of Computer Vision*, 90:88–105, 2010.

[10] Sandor Szedmak and John Shawe-taylor. Learning via linear operators: Maximum margin regression. Technical report, University of Southampton, UK, 2005.

[11] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

[12] Vladimir Vapnik. *Statistical learning theory*. Wiley, 1998.

[13] Yu Zhang and Dit-Yan Yeung. Multilabel relationship learning. *ACM Trans. Knowl. Discov. Data*, 7(2):1–30, August 2013.

Joint SVM for Accurate and Fast Image Tagging

Hanchen Xiong Sandor Szedmak Justus Piater *

Institute of Computer Science, University of Innsbruck
Technikerstr.21a A-6020, Innsbruck, Austria

Abstract. This paper studies how joint training of multiple support vector machines (SVMs) can improve the effectiveness and efficiency of automatic image annotation. We cast image annotation as an output-related multi-task learning framework, with the prediction of each tag's presence as one individual task. Evidently, these tasks are related via correlations between tags. The proposed joint learning framework, which we call *joint SVM*, can encode the correlation between tags by defining appropriate kernel functions on the outputs. Another practical merit of the joint SVM is that it shares the same computational complexity as one single conventional SVM, although multiple tasks are solved simultaneously. According to our empirical results on an image-annotation benchmark database, our joint training strategy of SVMs can yield substantial improvements, in terms of both accuracy and efficiency, over training them independently. In particular, it outperforms many other state-of-the-art algorithms.

1 Introduction

Automatic image annotation is an important yet challenging machine learning task. The importance is based on the fact that the number of images grows exponentially on the internet, and most of them have no link to semantic tags. Therefore, automatic annotation is of great significance to generate meaningful metadata for image retrieval from textual queries. The challenges are usually considered from two perspectives: first, to directly apply mature binary classification methods, e.g. Support Vector Machines (SVMs), assumes the independence of the labels; secondly, the image data on internet is usually presented in large volumes, so the desired learning method should be capable of working on large-scale data with high learning and prediction efficiency. One straightforward yet naive strategy is to consider each tag's presence as a binary classification problem. Then, multiple SVMs can be trained independently for different tags. This method, however, will suffer from high computational complexity in both training and prediction phases when the number of tags is relatively large. Independently learning multiple SVMs is not expected to work well because it ignores the correlation between the presences of tags, which is a phenomenal characteristic of image annotation tasks (e.g., sky and cloud often co-occur). In this paper, we propose to interpret image annotation as the learning of multiple related tasks. However, different from most existing multi-task learning frameworks [1] in which tasks are related through their *inputs*, our joint learning

*The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

method focuses on the relation between *outputs*. Our strategy is motivated by two intuitions. First, by connecting multiple SVM classifiers together, the correlation between their outputs (the presences of tags), presumably, can be more easily encoded. Secondly, if the outputs of multiple SVMs can be merged into a single vector entity, the optimization problem can be established and solved over vectors, greatly reducing the computational complexity. These two objectives, surprisingly, can be easily achieved by summing up the objectives and constraints in different SVMs, plus an appropriately designed kernel on outputs.

2 Joint Learning of Multiple SVMs

2.1 Support Vector Machines and Input Kernels

In the past two decades, support vector machines (SVMs) have seen remarkable successes in various domains. The achievements of SVMs mainly stems from its two advantageous components: *maximum margins* and *input kernels*. The maximum-margin principle is a reflection of statistical learning theory [2] on linear binary classification. Kernels provide powerful mechanisms enabling the linear classifier to separate highly non-linear data. The critical observation of kernel methods is that a kernel function can be defined on a pair of data instances to implicitly map them to a reproducing kernel Hilbert space (RKHS):

$$K_\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \langle \phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}^{(j)}) \rangle \tag{1}$$

where $\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathbb{R}^d$ are ith and jth input training instances, ϕ is the feature map induced by kernel function K_ϕ, and $\phi(\mathbf{x}^{(i)})$ is the representation of $\mathbf{x}^{(i)}$ in the RKHS $\mathcal{H}_\phi$. Most popularly, a Gaussian (or radial basis function) kernel

$$K_\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(\frac{-\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{2\sigma^2}\right) \tag{2}$$

is employed because its corresponding RKHS is an unnormalized Gaussian density function

$$\phi(\mathbf{x}^{(i)}) \propto \mathcal{N}(\tau; \mathbf{x}^{(i)}, \sigma) \tag{3}$$

which is of infinite dimension, and thus greatly improves the representational capability of input data. Given the training dataset $\{\mathbf{x}^{(i)} \in \mathbf{R}^d, y^{(i)} \in \{+1, -1\}\}_{i=1}^m$ of one binary classification problem, the primal form of training SVM is written

$$\begin{array}{rl} \min & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^m \xi^{(i)} \\ \text{w.r.t.} & \mathbf{w} \in \mathbb{R}^{\mathcal{H}_\phi \times 1}, b \in \mathbb{R} \\ \text{s.t.} & y^{(i)}\left(\mathbf{w}^\top \phi(\mathbf{x}^{(i)}) + b\right) \geq 1 - \xi^{(i)}, \xi^{(i)} \geq 0, i \in \{1, \ldots, m\} \end{array} \tag{4}$$

where $\mathbf{w}$ is a linear hyperplane in $\mathcal{H}_\phi$, b is bias term, $\xi^{(i)}$ are slack variables for the tolerance of noise, and C is trade-off parameter between training error and max-margin regularization. The computational advantage of kernels become

more obvious when the primal form of SVM (4) is reformulated to its dual form by introducing Lagrange multipliers α_i for each constraints:

$$\begin{array}{rl} \max & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} K_\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \\ \text{w.r.t.} & \alpha_1, \alpha_2, \ldots, \alpha_m \\ \text{s.t.} & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0; \forall i, 0 \leq \alpha_i \leq C \end{array} \tag{5}$$

The dual representation of $\mathbf{w}$ is $\sum_{i=1}^{m} \alpha_i y^{(i)} \phi(\mathbf{x}^{(i)})$, and thus the prediction of a test instance $\hat{\mathbf{x}}$ is

$$\hat{y} = \operatorname{sgn}\left(\mathbf{w}^\top \phi(\hat{\mathbf{x}}) + b\right) = \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i y^{(i)} K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}}) + b\right). \tag{6}$$

It can be seen that the kernel function K_ϕ enables the learning of a infinite-dimensional $\mathbf{w}$ without explicit computation in $\mathcal{H}_\phi$. (6) shows that the kernel function yields a similarity measurement between two input instances.

2.2 Joint SVM

Automatic image annotation tasks seek to predict the presence of tags given an input image. If we consider prediction of each tag's occurrence as a binary classification problem, we can enlist as many SVMs as the number of tags. Similar to other multi-task learning frameworks [1, 3], we connect the learning tasks of different SVMs by simply summing up their objectives and constraints respectively in the primal form

$$\begin{array}{rl} \max & \frac{1}{2} \sum_{t=1}^{T} ||\mathbf{w}_t||^2 + C \sum_{t=1}^{T} \sum_{i=1}^{m} \xi_t^{(i)} \\ \text{w.r.t.} & \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_T \in \mathbf{R}^{\mathcal{H}_\phi \times 1}, b_1, b_2, \ldots, b_T \in \mathbb{R} \\ \text{s.t.} & \sum_{t=1}^{K} y_t^{(i)} \left(\mathbf{w}_t^\top \phi(x^{(i)}) + b_t\right) \geq T - \sum_{k=1}^{T} \xi_t^{(i)} \end{array} \tag{7}$$

where t indexes different tags or learning tasks, and T is the total number of tags. By denoting $\mathbf{y}^{(i)} = [y_1^{(1)}, \ldots, y_T^{(i)}]$, $\mathbf{b} = [\frac{b_1}{T}, \ldots, \frac{b_T}{T}]$ and $\mathbf{W} = [\frac{\mathbf{w}_1^\top}{T}; \ldots; \frac{\mathbf{w}_T^\top}{T}]^\top$, we can rewrite (7) as a joint SVM:

$$\begin{array}{rl} \min & \frac{1}{2} ||\mathbf{W}||_F^2 + C \sum_{i=1}^{m} \bar{\xi}^{(i)} \\ \text{w.r.t.} & \mathbf{W} \in \mathbb{R}^{K \times \mathcal{H}_\phi}, \mathbf{b} \in \mathbb{R}^K \\ \text{s.t.} & \left\langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)}) + \mathbf{b} \right\rangle \geq 1 - \bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1, \ldots, m\} \end{array} \tag{8}$$

where $||\mathbf{W}||_F$ is the Frobenius norm of matrix $\mathbf{W}$, and $\bar{\xi}^{(i)} = \frac{1}{T} \sum_{t=1}^{T} \xi_t^{(i)}$. One rationale of (7) is that within the joint form of objectives and constraints, learning easy tasks can help the learning of challenging tasks. For example, if training data $(\mathbf{x}^{(i,)}, y_p^{(i)})$ can be easily classified correctly in the pth task (i.e., $y^{(i)}(\mathbf{w}_p^\top \mathbf{x}^{(i)} + b)/T > \frac{1}{T}$), it can offer some 'freedom' to other challenging tasks before violating constraint $\left\langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)}) + \mathbf{b} \right\rangle_{\mathcal{H}} \geq 1$. In addition, a key functionality this joint form (8) can afford is that we can define kernel functions on outputs $\mathbf{y}$ to improve their representational power (e.g. correlations). Assume the kernel function defined on outputs are $K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)})$ (the design of

the output kernel will be explained later) and the corresponding feature map is $\psi : \mathbb{R}^K \to \mathcal{H}_\psi$, then (8) is modified to

$$\begin{array}{rl} \min & \frac{1}{2}||\mathbf{W}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{w.r.t.} & \mathbf{W} \in \mathbb{R}^{H_\psi \times \mathcal{H}_\phi}, \mathbf{B} \in \mathbb{R}^{\mathcal{H}_\psi \times 1} \\ \text{s.t.} & \left\langle \psi(\mathbf{y}^{(i)}), \mathbf{W}\phi(x^{(i)}) + \mathbf{B} \right\rangle \geq 1 - \bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1, \ldots, m\} \end{array} \tag{9}$$

Interestingly, although derived from a rather different starting point, our joint SVM (9) is the same as Maximum Margin Regression (MMR) [4], wherein the motivation is to seek a linear operator in arbitrary tensor product space $\psi(\mathbf{y}^{(i)}) \otimes \phi(\mathbf{x}^{(i)})$. In addition, (9) is also related to structured-output learning [5] by sharing the same objective, yet with different constraints. Basically, the differences originate from two types of margins used in (9) and [5]. An empirical comparison of these two methods on structured-output learning is in [6]. The solution of the MMR stands close to the Minimum Description Length Principle, see for example in [7], by providing a highly compressed description to complex learning problems.

2.3 Output Kernels and Solutions

Similarly to a single conventional SVM, the joint SVM learning (9) can be converted to its dual form

$$\begin{array}{rl} \max & \sum_{i=1}^m \alpha_i - \sum_{i,j=1}^m \alpha_i \alpha_j K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) K_\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \\ \text{w.r.t} & \alpha_1, \cdots, \alpha_m \\ \text{s.t} & \sum_{i=1}^m \alpha_i \psi(\mathbf{y}^{(i)}) = 0; \quad \forall i, 0 \leq \alpha_i \leq C \end{array} \tag{10}$$

with $\mathbf{W} = \sum_i^m \alpha_i \psi(\mathbf{y}^{(i)}) \phi(\mathbf{x}^{(i)})^\top$. It can be seen, with kernel matrix on outputs pre-computed, that the computational complexity of joint learning (10) is the same as the learning of one single SVM (5), which is a great advantage in efficiency. In this paper, the Gaussian kernel function (2) is used on $\mathbf{y}$, hence $\psi(\mathbf{y}^{(i)})$ corresponds to an unnormalized density function (which is non-negative everywhere), and the bias-induced constraint $\sum_{i=1}^m \alpha_i \psi(\mathbf{y}^{(i)}) = 0$ will lead to a trivial solution $\forall i, \alpha_i = 0$. Since the Gaussian kernel is translation invariant, the bias in output space $\mathbf{y}$ has no effect, and we can ignore the bias $\mathbf{B}$ in (9) and its corresponding constraint in (10). Therefore, given a test data $\hat{\mathbf{x}}$, the prediction $\phi(\hat{\mathbf{y}})$ in $\mathcal{H}_\psi$ is

$$\psi(\hat{\mathbf{y}}) = \mathbf{W}\phi(\hat{\mathbf{x}}) = \sum_{i=1}^m \alpha_i \psi(\mathbf{y}^{(i)}) K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}}). \tag{11}$$

With identical scalar σ, the Gaussian kernel (2) on $\mathbf{y}$ can be decomposed into independent Gaussian kernels on each element y_t. To preserve the correlation between every pair of tags, one simple remedy is to use a full covariance matrix Σ. Here, we use a scaled empirical covariance from outputs in training data. Another computational issue is that there is no direct way (say, by inverting (11)) to map $\psi(\hat{\mathbf{y}})$ back to $\hat{\mathbf{y}}$. Therefore, instead of finding a closed form solution,

we can find the optimal solution $\hat{\mathbf{y}}^*$, out of all possible $\mathbf{y} \in \{+1, -1\}^T$, such that its projection in $\mathcal{H}_\psi$ is closest to $\mathbf{W}\phi(\hat{\mathbf{x}})$:

$$\begin{aligned}\hat{\mathbf{y}}^* &= \text{argmax}_{\mathbf{y}\in\{+1,-1\}^T} \langle \psi(\mathbf{y}), \mathbf{W}\phi(\hat{\mathbf{x}}) \rangle \\ &= \text{argmax}_{\mathbf{y}\in\{+1,-1\}^T} \sum_{i=1}^m \alpha_i K_\psi(\mathbf{y}^{(i)}, \mathbf{y}) \underbrace{K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}})}_{\beta_i} \\ &= \text{argmax}_{\mathbf{y}\in\{+1,-1\}^T} \sum_{i=1}^m \alpha_i \beta_i \exp\left(-\tfrac{1}{2}(\mathbf{y} - \mathbf{y}^{(i)})^\top \Sigma^{-1} (\mathbf{y} - \mathbf{y}^{(i)})\right)\end{aligned} \tag{12}$$

In general, there is no closed-form solution to (12), so usually approximate dynamic programming (ADP) is applied in searching for the optimum $\hat{\mathbf{y}}^*$. Here, we employ a simpler strategy. Since the number of tags associated with one image is relatively small, most of the $\mathbf{y}$ in $\{+1, -1\}^T$ space are bad solutions. Therefore, when the training data size is large, the most likely solutions of (12), presumably, are covered by the outputs in training data $\{\mathbf{y}\}_{i=1}^m$. Consequently, we can find the optimum $\hat{\mathbf{y}}^*$ via

$$\text{argmax}_{\{\mathbf{y}^{(j)}\}_{j=1}^m} \sum_{i=1}^m \alpha_i \beta_i \underbrace{\exp\left(-\frac{1}{2}(\mathbf{y}^{(i)} - \mathbf{y}^{(j)})^\top \Sigma^{-1} (\mathbf{y}^{(j)} - \mathbf{y}^{(i)})\right)}_{\gamma_{ij}} \tag{13}$$

where $\{\gamma_{ij}\}_{i,j=1}^m$ were already computed in the training phase, $\{\alpha_i\}_{i=1}^m$ are training results, and only the computation of $\{\beta_i\}_{i=1}^m$ is needed during testing.

3 Experiment

In our experiment, we used the Corel5K benchmark dataset with image features extracted as in [8]. The dataset contains 5,000 images of different scenarios and objects, out of which 4500 images are used as training data and 500 images are test data. In the database, there exist 260 tags, and on average each image is annotated with 3.5 tags. For all images in the database, 15 different features [8] were extracted.

In our experiment, we applied both a joint SVM and independent SVMs for comparison. To ensure fairness, in the learning phase, the optimization problems (5) and (10) were solved with the same coordinate descent method [9]. In addition, for both independent SVMs and the joint SVM, 500 instances of training data were taken out as validation data to find the best parameter C. In the testing phase, the performance was measured with precision, recall and F1 score. To measure the efficiency, training and testing times were recorded as well. We tried two learners on 15 different input visual features and found that the global "RgbV3H1" feature yields best results for both cases. All experiments were run on the same simulation and hardware conditions (Python, Intel Core i7). The comparison of accuracy and efficiency between independent SVMs and joint SVM is presented in Figure 1. While the learning and testing time of independent SVMs scale with the number of tags, the computation time of the joint SVM approximately equals a SVM for single-tag classification. At the same

	Training Time (sec)	Testing Time (sec)	Testing Performance		
			Precision	Recall	F1
Independent SVMs	6285.11	317.20	0.1049	0.1225	0.1130
Joint SVM (Gaussian)	80.68	**6.92**	**0.4078**	**0.3713**	**0.3887**
Joint SVM (Polynomial)	**76.48**	9.11	0.3908	0.3565	0.3728
The best result in [10]	–	–	0.27	0.32	0.292

Fig. 1: Performance of different algorithms.

time, in terms of accuracy, the joint SVM worked much better than independent SVMs. In addition, to test the higher-order dependency among tags, a 3rd-degree polynomial kernel function was also applied. The performance of this type kernel falls very close to that the Gaussian kernel provided. However, it is worth noting that the proposed joint SVM, with either Gaussian or polynomial kernel, outperforms many other state-of-the-art methods by a large margin (see a survey in [10]) on the same database.

4 Conclusions

A novel joint SVM was presented for automatic image tagging. Its superiority over conventional SVMs is obvious from our mathematical derivation and empirical results. Although, in this preliminary work, simple individual features and kernels already display good results, yet more improvements are expected when more features and sophisticated kernels, e.g. multi-kernel learning, are employed, which is a promising direction of future work.

References

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *NIPS*, pages 41–48, 2006.

[2] Vladimir Vapnik. *Statistical learning theory*. Wiley, 1998.

[3] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[4] Sandor Szedmak and John Shawe-taylor. Learning via linear operators: Maximum margin regression. Technical report, University of Southampton, UK, 2005.

[5] Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In *ICML*, 2005.

[6] Katja Astikainen, Liisa Holm, Esa Pitkänen, Sandor Szedmak, and Juho Rousu. Towards structured output prediction of enzyme function. In *BMC Proceedings, 2(4):S2*. 2008.

[7] Peter D. Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007.

[8] Matthieu Guillaumin, Thomas Mensink, Jakob Verbeek, and Cordelia Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, 2009.

[9] Francesco Dinuzzo, Cheng Soon Ong, Peter V. Gehler, and Gianluigi Pillonetto. Learning output kernels with block coordinate descent. In *ICML*, 2011.

[10] Ameesh Makadia, Vladimir Pavlovic, and Sanjiv Kumar. Baselines for image annotation. *International Journal of Computer Vision*, 90:88–105, 2010.

Scalable, Accurate Image Annotation with Joint SVMs and Output Kernels

Hanchen Xiong[1], Sandor Szedmak, Justus Piater

Institute of Computer Science, University of Innsbruck, Technikerstr.21a, A-6020 Innsbruck, Austria

Abstract

This paper studies how joint training of multiple support vector machines (SVMs) can improve the effectiveness and efficiency of automatic image annotation. We cast image annotation as an output-related multi-task learning framework, with the prediction of each tag's presence as one individual task. Evidently, these tasks are related via dependencies between tags. The proposed joint learning framework, which we call *joint SVM*, is superior to other related models in its impressive and flexible mechanisms in exploiting the dependencies between tags: first, a linear output kernel can be implicitly learned when we train a joint SVM; or, a pre-designed kernel can be explicitly applied by users when prior knowledge is available. Also, a practical merit of joint SVM is that it shares the same computational complexity as one single conventional SVM, although multiple tasks are solved simultaneously. Although derived from the perspective of multi-task learning, the proposed joint SVM is highly related to structured-output learning techniques, *e.g.* max-margin regression [1], structural SVM [2]. According to our empirical results on several image-annotation benchmark databases, our joint training strategy of SVMs can yield substantial improvements, in terms of both accuracy and efficiency, over training them independently. In particular, it compares favorably with many other state-of-the-art algorithms. We also develop a "perceptron-like" online learning scheme for joint SVM to enable it to scale up better to huge data in real-world practice.

Keywords: image annotation, multi-label learning, output kernels, maximum margin
2014 MSC: 00-01, 99-00

1. Introduction

Automatic image annotation is an important yet challenging machine learning task. The importance is based on the fact that the number of images grows increasingly fast on the internet, and most of them have no link to semantic tags (or keywords, labels). Therefore, automatic annotation is of great significance to generate meaningful meta-data for organizing image collections, and in particular, retrieving images from textual queries. The challenges are usually considered from two classical perspectives [3]: first, *semantic-gap*, *i.e.* the gap from low-level image features to textual tags is large and there exist no reliable way to extract dependencies between them; secondly, *absence of correspondence*, *i.e.* for each tag associated with one image, there is no corresponding region annotated, which hinders learning worse. Meanwhile, when considering contemporary image annotation, one more difficulty to bear is big data. The image data on internet is usually presented in large volumes (million or billion level), so the desired learning method should be capable of working on large-scale data with high learning and prediction efficiency. One straight-forward yet naive strategy is to consider each tag's presence as a binary classification problem. Then, multiple binary classifiers, *e.g.* support vector machines (SVMs), can be trained independently for different tags. This method, however, will suffer from high computational complexity in both training and prediction

Email addresses: hanchen.xiong@uibk.ac.at (Hanchen Xiong), sandor.szedmak@uibk.ac.at (Sandor Szedmak), justus.piater@uibk.ac.at (Justus Piater)
[1]Corresponding author

phases when the number of tags is relatively large. In addition, independently learning multiple SVMs is not expected to work well because it ignores the dependencies between the presences of tags [4], which is a phenomenal characteristic of image annotation tasks (*e.g.*, sky and cloud often co-occur).

In this paper, we propose to interpret image annotation as the learning of multiple related tasks. However, different from most existing multi-task learning frameworks [5] in which tasks are related through their *inputs*, our joint learning method focuses on the relation between *outputs*. Our strategy is motivated by two intuitions. First, by connecting multiple SVM classifiers together, the dependencies between their outputs (the presences of tags), presumably, can be more easily encoded. Secondly, if the outputs of multiple SVMs can be merged into a single vector entity, the optimization problem can be established and solved over vectors, greatly reducing the computational complexity. These two objectives, surprisingly, can be easily achieved by summing up the objectives and constraints in different SVMs, plus an appropriately designed kernel on outputs. We refer to the proposed training strategy as *joint SVM*. The key strength of joint SVM is that it can flexibly offer two mechanisms to exploit the dependencies between tags: first, when there is no prior knowledge on the dependencies, a linear output kernel can be implicitly learned when we train a joint SVM; or a pre-designed, prior-oriented, kernel can be applied on outputs when prior knowledge is available (see section 4). In addition, as we will see in section 3, the training of joint SVM shares almost the same computation complexity as a single regular SVM, which is a practical merit when the number of tags is relative large. Interestingly, although derived from the perspective of multi-task learning, the proposed joint SVM highly relates to structured-output learning techniques, such as max-margin regression [1], structural SVM [2] or max-margin Markov network (M^3N) [6]. More connections between them will be exploited in section 5. In addition, to enable joint SVM to scales up to huge data (million or billion level) in real-world practice, we develop a "perceptron-like" online learning algorithm for joint SVM in section 6. In our experiment (section 7), we tested joint SVM on several benchmark image-annotation databases, with comparison against independent SVMs and other results reported in state-of-the-art algorithms. The experimental results show that our joint SVM can gain impressive improvement over training SVMs independently. In particular, it compares favorably with many other state-of-the-art algorithms.

2. Related Work

Prior to our work, there exist many literatures on image annotation in computer vision and machine learning communities [7, 8, 9, 10, 3, 11, 12, 13, 14, 4, 15, 16]. Roughly speaking, all algorithms can be categorized into generative methods or discriminative methods according to how the relevance between image features and textual tags are modeled. On one hand, generative methods, mostly inspired by linguistic translation studies, model the generating or formating procedure of visual features and tags, then tags prediction from a novel image is inferred by leveraging co-occurrence statistics between visual features and tags in training data. Continuous Relevance Model (CRM) [7], Correlation Latent Dirichlet Allocation (CorLDA) [8] and Multiple Bernoulli Relevance Model (MBRM) [9] belong to the generative category. However, one drawback of these method is that usually some statistical assumptions (*e.g.* conditional independence) are imposed on models, which restricts their modeling capabilities. Furthermore, another practical obstacle of most generative methods is the intractability of inference in tag prediction phase, therefore, usually some approximation techniques are applied. On the other hand, discriminative methods directly model the tag-predicting function, out of which TagProp [10], JEC [3] are metric-learning based approaches, rank-SVM [17], LM-K [18] are rank-learning based approaches, M3L [4] and Multi-Label Relationship Learning (MLRL) [16] are maximum-margin based approaches. One notable issue, and also difficulty, in discriminative methods is the dependencies between output tags, of which many state-of-the-art studies [4, 11, 18] have being aware. In several recent studies [10, 3, 11], discriminative methods were reported to displayed empirically superior performance than generative ones on image annotation task. More comparison and analysis on different representative methods can be found in up-to-date reviews [3, 4, 11].

The proposed joint SVM in this paper is a maximum-margin based, discriminative learning framework. Although joint SVM displays strong connections with structured-output learning, the staring point of our work is to improve the annotation performance by exploiting the relationship between individual tag-predictors. A conceptually-related work was concurrently, but independently from us, presented in MLRL

[16], of which the authors explicitly model the relationship as a covariance matrix in matrix-variate normal distribution over individual model parameters. In contrast, in joint SVM, the dependency between different tags are encoded in output kernels. In this sense, our work is also similar to LM-K [18] and M3L [4]. Interestingly, when the output kernel is linear, it is equivalent to the explicit relationship learning in MLRL. Meanwhile, more sophisticated output kernel can flexibly be constructed and utilized in joint SVM, to afford nonlinear, higher-order dependencies, although they are not always of help in practice.

3. Joint Learning of Multiple SVMs

3.1. Support Vector Machines and Input Kernels

In the past two decades, support vector machines (SVMs) have displayed remarkable successes in various application domains. The achievements of SVMs mainly stems from its two advantageous components: *maximum margins* and *input kernels*. The maximum-margin principle is a reflection of statistical learning theory [19] on linear binary classification. Kernels provide powerful mechanisms enabling the linear classifier to separate highly non-linear data. The critical observation of kernel methods is that a kernel function can be defined on a pair of data instances to implicitly map them to a reproducing kernel Hilbert space (RKHS):

$$K_\phi(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \langle \phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}^{(j)}) \rangle \tag{1}$$

where $\mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathbb{R}^d$ are ith and jth input training instances, ϕ is the feature map induced by kernel function K_ϕ, and $\phi(\mathbf{x}^{(i)})$ is the representation of $\mathbf{x}^{(i)}$ in the RKHS $\mathcal{H}_\phi$. Most popularly, a Gaussian (or radial basis function) kernel

$$K_\phi^{Gau}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2 / 2\sigma^2\right) \tag{2}$$

is employed because its corresponding RKHS is an unnormalized Gaussian density function:

$$\phi^{Gau}(\mathbf{x}^{(i)}) \propto \mathcal{N}(\tau; \mathbf{x}^{(i)}, \sigma) \tag{3}$$

which is of infinite dimension, and thus greatly improves the representational capability of input data. Another popular kernel function is Polynomial kernel:

$$K_\phi^{Pol}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \left(\langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle + c\right)^d \tag{4}$$

In particular, when the degree $d = 1$ and constant term $c = 0$, Polynomial is a simple inner product. Meanwhile, in 2-degree ($d = 2$) Polynomial kernel, corresponding feature map is:

$$\phi^{Pol}(\mathbf{x}) = [x_d^2, \cdots, x_1^2, \sqrt{2}x_d x_{d-1}, \cdots, \sqrt{2}x_2 x_1, \sqrt{2c}x_d, \cdots, \sqrt{2c}x_1, c]^\top \tag{5}$$

Given the training dataset $\{\mathbf{x}^{(i)} \in \mathbb{R}^d, y^{(i)} \in \{+1, -1\}\}_{i=1}^m$ of one binary classification problem, the primal form of training SVM is written

$$\begin{array}{ll} \arg\min_{\mathbf{w} \in \mathbb{R}^{\mathcal{H}_\phi \times 1}} & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^m \xi^{(i)} \\ \text{s.t.} & y^{(i)}\left(\mathbf{w}^\top \phi(\mathbf{x}^{(i)})\right) \geq 1 - \xi^{(i)}, \xi^{(i)} \geq 0, i \in \{1, \ldots, m\} \end{array} \tag{6}$$

where $\mathbf{w} \in \mathbb{R}^{\mathcal{H}_\phi \times 1}$ is the normal vector of a linear hyperplane in $\mathcal{H}_\phi$ (here and later we use $\mathcal{H}$ as $dim(\mathcal{H})$ for simplicity when we denote dimensionality) , $\xi^{(i)}$ are slack variables for the tolerance of noise, and C is trade-off parameter de between training error and max-margin regularization. Eq.(6) differs from usual SVM formulation slightly at the absence of a bias term. Here we ignore the bias since it can be absorbed in $\mathbf{w}$ [2]. Actually, eliminating the bias is more critical in predicting multiple dependent labels, check [4] for

[2]When a Polynomial kernel is used, a bias term is already in its corresponding feature map. When a Gaussian kernel is used, an input vector can be augmented with one extra constant.

detailed explanations. The computational advantage of kernels become obvious when the primal form of SVM (Eq.(6)) is reformulated to its dual form by introducing Lagrange multipliers α_i for each constraints:

$$\begin{array}{rl} \arg\min_{\alpha_1,\alpha_2,\ldots,\alpha_m} & \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m \alpha_i\alpha_j y^{(i)}y^{(j)}K_\phi(\mathbf{x}^{(i)},\mathbf{x}^{(j)}) \\ \text{s.t.} & \forall i, 0 \le \alpha_i \le C \end{array} \tag{7}$$

The dual representation of $\mathbf{w}$ is $\sum_{i=1}^m \alpha_i y^{(i)}\phi(\mathbf{x}^{(i)})$, and thus the prediction of a test instance $\hat{\mathbf{x}}$ is

$$\hat{y} = \operatorname{sgn}\left(\mathbf{w}^\top\phi(\hat{\mathbf{x}})\right) = \operatorname{sgn}\left(\sum_{i=1}^m \alpha_i y^{(i)}K_\phi(\mathbf{x}^{(i)},\hat{\mathbf{x}})\right). \tag{8}$$

It can be seen that the kernel function K_ϕ enables the learning of a high-dimensional (even infinite) $\mathbf{w}$ without explicit computation in $\mathcal{H}_\phi$. Eq.(8) shows that the kernel function yields a similarity measurement between two input instances, and the prediction is working as a weighted-sum of all outputs in the training data.

3.2. Joint SVM

The automatic image annotation task seeks to predict the presence of tags given an input image. Assume d-dimensional visual features are extracted from input images and there are T tags in a pre-defined dictionary, then the annotation learning task is to seek a function $f : \mathbf{x} \in \mathbb{R}^d \to \{-1,+1\}^T$. If we consider prediction of each tag's occurrence as a binary classification problem, we can list as many SVMs as the number of tags. Similar to other multi-task learning frameworks [5], we connect the learning tasks of different SVMs by simply summing up their objectives and constraints respectively in the primal form

$$\begin{array}{rl} \min & \frac{1}{2}\sum_{t=1}^T ||\mathbf{w}_t||^2 + C\sum_{t=1}^T\sum_{i=1}^m \xi_t^{(i)} \\ \text{w.r.t.} & \mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_T \in \mathbf{R}^{\mathcal{H}_\phi\times 1} \\ \text{s.t.} & \sum_{t=1}^T y_t^{(i)}\left(\mathbf{w}_t^\top\phi(x^{(i)})\right) \ge T - \sum_{t=1}^T \xi_t^{(i)} \end{array} \tag{9}$$

where t indexes different tags or learning tasks, and T is the total number of tags. By denoting $\mathbf{y}^{(i)} = [y_1^{(1)},\ldots,y_T^{(i)}]$ and $\mathbf{W} = [\frac{\mathbf{w}_1^\top}{T};\ldots;\frac{\mathbf{w}_T^\top}{T}]^\top$, we can rewrite (Eq.(9)) as a joint SVM:

$$\begin{array}{rl} \arg\min_{\mathbf{W}\in\mathbb{R}^{T\times\mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{W}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)})\right\rangle \ge 1 - \bar{\xi}^{(i)}, \xi_i \ge 0, i \in \{1,\ldots,m\} \end{array} \tag{10}$$

where $||\mathbf{W}||_F$ is the Frobenius norm of matrix $\mathbf{W}$, and $\bar{\xi}^{(i)} = \frac{1}{T}\sum_{t=1}^T \xi_t^{(i)}$. Eq.(10) is referred to as *joint SVM*. One rationale of Eq.(10) is that within the joint form of objectives and constraints, learning easy tasks can help the learning of challenging tasks. For example, if training data $(\mathbf{x}^{(i,)}, y_p^{(i)})$ can be easily classified correctly in the pth task (*i.e.*, $y^{(i)}(\mathbf{w}_p^\top\mathbf{x}^{(i)})/T > \frac{1}{T}$), it can offer some 'freedom' to other challenging tasks before violating constraint $\left\langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)})\right\rangle_{\mathcal{H}} \ge 1$. Meanwhile, a more critical strength of Eq.(10) is that a linear output kernel is implicitly learned and absorbed in the model parameters $\mathbf{W}$. More rigorous explanation will be presented later in section 4.1. In addition, another key functionality joint SVM can afford is that we can also, based on our prior knowledge, explicitly define kernel functions on outputs $\mathbf{y}$ to improve their representational power (*e.g.* dependencies). Assume the kernel function defined on outputs are $K_\psi(\mathbf{y}^{(i)},\mathbf{y}^{(j)})$ (the output kernel will be explained later) and the corresponding feature map is $\psi : \{-1,+1\}^T \to \mathcal{H}_\psi$, then Eq.(10) is modified to

$$\begin{array}{rl} \arg\min_{\mathbf{W}\in\mathbb{R}^{H_\psi\times\mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{W}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \psi(\mathbf{y}^{(i)}), \mathbf{W}\phi(x^{(i)})\right\rangle \ge 1 - \bar{\xi}^{(i)}, \xi_i \ge 0, i \in \{1,\ldots,m\} \end{array} \tag{11}$$

Similarly to a single conventional SVM, joint SVM Eq.(11) can be converted to its dual form

$$\begin{array}{rl} \arg\min_{\alpha_1,\cdots,\alpha_m} & \sum_{i=1}^m \alpha_i - \sum_{i,j=1}^m \alpha_i\alpha_j K_\psi(\mathbf{y}^{(i)},\mathbf{y}^{(j)})K_\phi(\mathbf{x}^{(i)},\mathbf{x}^{(j)}) \\ \text{s.t} & \forall i, 0 \le \alpha_i \le C \end{array} \tag{12}$$

with $\mathbf{W} = \sum_i^m \alpha_i \psi(\mathbf{y}^{(i)})\phi(\mathbf{x}^{(i)})^\top$. It can be seen, with the kernel matrix on outputs pre-computed, that the computational complexity of joint learning (Eq.(12)) is the same as the learning of one single SVM (Eq.(7)), which is a great advantage in efficiency.

Given a test input $\hat{\mathbf{x}}$, the prediction $\phi(\hat{\mathbf{y}})$ in $\mathcal{H}_\psi$ is

$$\psi(\hat{\mathbf{y}}) = \mathbf{W}\phi(\hat{\mathbf{x}}) = \sum_{i=1}^{m} \alpha_i \psi(\mathbf{y}^{(i)}) K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}}). \tag{13}$$

Meanwhile, one computational issue is that there is no direct way (say, by inverting Eq.(13)) to map $\psi(\hat{\mathbf{y}})$ back to $\hat{\mathbf{y}}$. Therefore, we can find the optimal solution $\hat{\mathbf{y}}^*$, out of all possible $\mathbf{y} \in \{+1, -1\}^T$, such that its projection in $\mathcal{H}_\psi$ is closest to $\mathbf{W}\phi(\hat{\mathbf{x}})$:

$$\begin{aligned} \hat{\mathbf{y}}^* &= \text{argmax}_{\mathbf{y}\in\{+1,-1\}^T} \langle \psi(\mathbf{y}), \mathbf{W}\phi(\hat{\mathbf{x}}) \rangle \\ &= \text{argmax}_{\mathbf{y}\in\{+1,-1\}^T} \sum_{i=1}^m \alpha_i \underbrace{K_\phi(\mathbf{x}^{(i)}, \hat{\mathbf{x}})}_{\beta_i} K_\psi(\mathbf{y}^{(i)}, \mathbf{y}) \end{aligned} \tag{14}$$

In general, there is no closed-form solution to Eq.(14), so usually approximate dynamic programming (ADP) is applied in searching for the optimum $\hat{\mathbf{y}}^*$. Here, we employ a simpler yet effective strategy. Since the number of tags associated with one image is rather small, most of the $\mathbf{y}$ in $\{+1, -1\}^T$ space are bad solutions. Therefore, when the training data size is large, the most likely solutions of Eq.(14), presumably, are covered by the outputs in training data $\{\mathbf{y}\}_{i=1}^m$. Consequently, we can find the optimum $\hat{\mathbf{y}}^*$ via a similar neighbour-based label transferring theme as [3, 10]:

$$\hat{\mathbf{y}}^* = \left(\sum_{k=1}^{K} \mathbf{y}^{(k)} w_k \right) / \sum_{k=1}^{K} w_k \tag{15}$$

$$w_j = \sum_{i=1}^{m} \alpha_i \beta_i K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) \tag{16}$$

where $k = \{j \in [1, m] : w_j > 0\}$ and maximum $K = 10$ neighbours are taken into account. Since α_i are $K_\psi(\mathbf{y}^{(i)}, \mathbf{y}^{(j)})$ were already computed in the training phase, only the computation of $\{\beta_i\}_{i=1}^m$ is needed during testing. Thus, the complexity in predicting is $\mathcal{O}(m)$.

4. Implicit and Explicit Linear Output Kernels on Tag-Sets

To transform the pairwise and triplet-wise dependencies between tags into the inner product of two outputs containing those tags, 2-degree and 3-degree Polynomial kernels are tried in [18] and it was reported that 2-degree is better than 3-degree. In [4, 16, 20], linear feature maps were exploited also for pairwise dependencies. In particular, linear output kernels and models were simultaneously learned in [16, 20], while the output kernel in [4] is pre-computed as a correlation matrix over output vectors. In this paper, based on the experience from previous literatures, we also only focus pairwise dependencies and study linear kernels (although higher-order kernels will also be tried in our experiments, and the performance among different kernels can be checked in section 7). Here we adopted strategies both in [16, 20] and in [4]. At first, we present that the linear output kernel can be implicitly, but more simply compared to [16, 20], learned when we train a joint SVM. Secondly, we developed a novel pre-designed linear kernel function, which can be seen as a replacement of the kernel with correlation matrix used in [4].

4.1. Implicit linear output kernel learning

Assume that the statistics of tags' pairwise co-occurrence can be encoded in a $T \times T$matrix $\mathbf{P}$, via which the output vectors can be linearly mapped as $\psi(\mathbf{y}) = \mathbf{P}\mathbf{y}$, and thus output kernel is:

$$K_\psi^{Lin}(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) = \mathbf{y}^{(i)\top} \mathbf{\Omega} \mathbf{y}^{(j)} \tag{17}$$

where $\mathbf{\Omega} = \mathbf{P}^\top\mathbf{P} = \mathbf{P}\mathbf{P}^\top$. By denoting $\mathbf{U} = \mathbf{P}^\top\mathbf{W}$, we can rewrite joint SVM (Eq.(11)) as:

$$\begin{array}{rl} \arg\min_{\mathbf{W}\in\mathbb{R}^{H_\psi\times\mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{W}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \mathbf{y}^{(i)}, \mathbf{U}\phi(x^{(i)})\right\rangle \geq 1-\bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1,\ldots,m\} \end{array} \tag{18}$$

Meanwhile, we need to control the scale of $\mathbf{P}$, otherwise the constraints in Eq.(18) will be pointless. In [20] one extra regularization on $\mathbf{\Omega}$, $\frac{1}{2}||\mathbf{\Omega}||_F^2$, was added into the objective function, while $||P||_F = 1$ was used in [16]. By contrast, a pseudo regularization on $\mathbf{P}$ is used in [11] via the re-construction loss from manually-corrupted data and $\mathbf{P}$. Here we apply a simpler strategy by using a compact regularizer, $\frac{1}{2}\mathbf{W}^\top\mathbf{\Omega}\mathbf{W}$, resulting in:

$$\begin{array}{rl} \arg\min_{\mathbf{U}\in\mathbb{R}^{H_\psi\times\mathcal{H}_\phi}} & \frac{1}{2}||\mathbf{U}||_F^2 + C\sum_{i=1}^m \bar{\xi}^{(i)} \\ \text{s.t.} & \left\langle \mathbf{y}^{(i)}, \mathbf{U}\phi(x^{(i)})\right\rangle \geq 1-\bar{\xi}^{(i)}, \xi_i \geq 0, i \in \{1,\ldots,m\} \end{array} \tag{19}$$

Remarkably, Eq.(19) is equivalent to Eq.(11) with $\mathbf{W}$ substituted by $\mathbf{U}$, which suggests that a linear output kernel is implicitly learned, and absorbed in $\mathbf{W}$, when we training a simple joint SVM with no explicit kernel on outputs.

4.2. Odds-ratio based kernel

In this paper, we also explicitly design an odds-ratio based kernel over tag-sets to capture pairwise dependencies. The dependency between tags measures how much the appearance of one tag increases or decreases the chance of another tag to occur in the same label set. At first, we can estimate the probability of co-occurrence of two labels, w_r and w_s, form training data:

$$P(w_r, w_s) = \frac{\sum_{i=1,\ldots,m} \mathbf{y}_r^{(i)} = 1 \text{ and } \mathbf{y}_s^{(i)} = 1}{m}. \tag{20}$$

according to which, we can compute the odds ratio, a measure, of the dependency between those words by the well known formula [21]:

$$O_{rs} = \frac{P(w_r, w_s)P(\overline{w_r}, \overline{w_s})}{P(w_r, \overline{w_s})P(\overline{w_r}, w_s)}, \tag{21}$$

where $\overline{w_r}$ means the complement of w_r (counting those sample items where w_r does not occur). Then the odds ratio is symmetrized by taking its logarithm, where the 0 value expresses the independence and the positive (or negative) value corresponds to higher (or lower) co-occurrence of those words than the random case. The higher of the magnitude of the log-odds-ratio shows stronger deviation from the independence.

$$Q_{rs} \leftarrow \log(O_{rs}) \tag{22}$$

The odds-ratio based kernel on a pair of outputs can then be computed:

$$K_\psi^{Odd}(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}) = \mathbf{y}^{(i)\top}\mathbf{Q}\mathbf{y}^{(j)} \tag{23}$$

where $\mathbf{Q}$ is the log-odds-ratio matrix with $\mathbf{Q}_{rs} = Q_{rs}$.

5. Relation to Structured-Output Learning

Interestingly, although derived from a rather different starting point, our joint SVM (Eq.(11)) is the same as Maximum Margin Regression (MMR) [1], wherein the motivation is to seek a linear operator in arbitrary tensor product space $\psi(\mathbf{y}^{(i)}) \otimes \phi(\mathbf{x}^{(i)})$. In addition, Eq.(11) is also related to structural SVM [6, 2] by sharing the same objective, yet with different constraints. An empirical comparison of these two methods on hierarchical-label learning is in [22]. The solution of the MMR stands close to the Minimum Description Length Principle, see for example in [23], by providing a highly compressed description to complex learning problems. In particular, when a linear output kernel and Hamming loss function are used in structural SVM. Structural SVM can be converted to a rather similar formulation as joint SVM by decomposing Hamming loss and $\mathbf{y}$ element-wisely. The detailed derivation was presented in [4].

6. Online Learning of Joint SVM

In real-word applications, the number of images can be very huge and beyond the memory storage and computing capacities of normal PCs. For instance, millions of images are uploaded to Facebook^{TM} and Flicker^{TM} every day. Obviously, the computation of kernel matrix for even daily volume is impractical. The formulation of joint SVM also suggests an implementation of a "perceptron-like" algorithm . For simplicity, here we present the case where no output kernel is applied. We aim to demonstrate the transparency of the formulation of joint SVM, which allows us to inherit most of the machine learning techniques developed earlier. Consider the optimization problem in Eq.(10) when only the error term is minimized

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} h(\lambda - \langle \mathbf{y}^{(i)}, \mathbf{W}\boldsymbol{\phi}(\mathbf{x}^{(i)})\rangle_{\mathcal{H}_y}) \\ \text{subject to} \quad & \{\mathbf{W} | \mathbf{W} : \mathcal{H}_x \to \mathcal{H}_y, \mathbf{W} \text{ a linear operator}\}, \end{aligned} \tag{24}$$

where λ is a prescribed margin, and the function $h(u)$ denotes the Hinge loss, that is

$$h(u) = \begin{cases} u & \text{if } u > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

The error function that we are going to minimize has subgradient with respect to $\mathbf{W}$ and this can be computed independently in an incremental way for each term occurring in the summation Eq.(24). The reader can consult to [24] and [25] for details of incremental subgradient methods. The term-wise subgradient is equal to

$$\partial h(\lambda - \langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)})\rangle_{\mathcal{H}_y})|_{\mathbf{W}} = \begin{cases} -\mathbf{y}^{(i)}\phi(x^{(i)})^T & \text{if } \lambda - \langle \mathbf{y}^{(i)}, \mathbf{W}\phi(x^{(i)})\rangle_{\mathcal{H}_y} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{26}$$

We can define the learning speed with a step size, denoted by s, and we obtain the "perceptron-like" algorithm given in Figure 1. In that algorithm $\mathbf{W}^{norm}$ denotes the L_2 normalized linear operator.

Input of the learner: The sample S, step size s
Output of the learner: $\mathbf{W} \in \mathbb{R}^{\mathcal{H}_y \times \mathcal{H}_x}$
Initialization: $t = 0$; $\mathbf{W}_t = \mathbf{0}$; $\mathbf{W}_t^{norm} = \mathbf{0}$; $\|\mathbf{W}_t\| = 0$
Repeat
  for $i = 1, 2, \ldots, m$ **do**
    read input-output pair: $(\mathbf{x}_i, \mathbf{y}_i)$
    $\beta_i = \langle \mathbf{y}_i, \mathbf{W}_t^{norm}\boldsymbol{\phi}(x_i)\rangle_{\mathcal{H}_y}$
    if $\beta_i < \lambda$ **then**
      $\mathbf{W}_{t+1} = \mathbf{W}_t + s\mathbf{y}_i\boldsymbol{\phi}(x_i)^T$
      $t = t + 1$
      $\|\mathbf{W}_{t+1}\|^2 = \|\mathbf{W}_t\|^2 + s^2\|\mathbf{y}_i\|^2\|\phi(\mathbf{x}_i)\|^2 + 2s\beta_i$
      $\mathbf{W}_{t+1}^{norm} = \mathbf{W}_{t+1}/\|\mathbf{W}_{t+1}\|$
    end if
  end for
until

(27)

Figure 1: Primal "perceptron-like" online learning algorithm for joint SVM.

The departure from the original perceptron algorithm, see for example in [26] and [27], is very moderate. Here we need to learn a matrix realizing the projection of the input vectors into the output space. The incremental subgradient based update employs the direct product of the corresponding output and input vectors to update the projection matrix. Furthermore a normalization step is also included as a certain regularization step, similar approach is proposed in [28].

A dual version of perceptron algorithm can be derived to learn vector outputs. Assume $\mathbf{W}$ is expressible by the training instances, then we have the optimization problem

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} h(\lambda - \sum_{j=1}^{m} \alpha_j \overbrace{\langle \mathbf{y}^{(i)}, \mathbf{y}^{(j)} \rangle}^{\kappa_{ij}^{y}} \overbrace{\langle \boldsymbol{\phi}(\mathbf{x}^{(i)}), \boldsymbol{\phi}(\mathbf{x}^{(j)}) \rangle}^{\kappa_{ij}^{\phi}}) \\ \text{subject to} \quad & \alpha_j \geq 0, \ j = 1, \ldots, m, \end{aligned} \tag{28}$$

The partial derivatives for $\alpha_i, \ k = 1, \ldots, m$ equals to

$$\partial h(\lambda - \sum_{j=1}^{m} \alpha_j \kappa_{ij}^{y} \kappa_{ij}^{\phi})|_{\alpha_i} = \begin{cases} -\kappa_{ij}^{y} \kappa_{ij}^{\phi} & \text{if } h(\lambda - \sum_{j=1}^{m} \alpha_j \kappa_{ij}^{y} \kappa_{ij}^{\phi}) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

Finally the corresponding dual perceptron algorithm is formulated according to Figure 2. An analogue of

Input of the learner: The training set S, step size s,
Output of the learner: $(\alpha_j), \ j = 1, \ldots, m,$
Initialization: $\alpha_j = \mathbf{0}; \ j = 1, \ldots, m,$
Repeat
  for $i = 1, 2, \ldots, m$ **do**
    read input: $\mathbf{x}^{(i)} \in \mathbb{R}^n$;
    if $(\sum_{j=1}^{m} \alpha_j \kappa_{ij}^{y} \kappa_{ij}^{\phi}) < \ \lambda$ **then**
      for $j = 1, 2, \ldots, m$ **do**
        $\alpha_j = \alpha_j + s \kappa_{ij}^{y} \kappa_{ij}^{\phi}$
      endif
    end if
  end for
until

(30)

Figure 2: Dual "perceptron-like" online learning algorithm for joint SVM.

the standard Novikoff theorem provides an upper bound on the number of updates and a lower bound on the achievable margin in the primal formulation. Here we follow the derivation that was presented in [29]. Let us define the margin for perceptron learner as

$$\gamma(\mathbf{W}, S, \phi) = \min_{(\mathbf{y}^{(i)}, \mathbf{x}^{(i)}) \in S} \frac{\langle \mathbf{y}^{(i)}, \mathbf{W} \boldsymbol{\phi}(\mathbf{x}^{(i)}) \rangle_F}{\|\mathbf{W}\|_F}. \tag{31}$$

Then we can claim the following statement not assuming the normalization step in the algorithm:

Theorem 1. *Let $S = \{(\mathbf{y}^{(i)}, \mathbf{x}^{(i)})\} \subset (\mathcal{Y} \times \mathcal{X}), \ i = 1, \ldots$ be a sample set independently and identically drawn from an unknown distribution and let $\phi : \mathcal{X} \to \mathcal{H}_\phi$ be an embedding into a Hilbert space, furthermore assume that $\|\boldsymbol{\phi}(\mathbf{x}^{(i)})\| = 1$ and $\|\mathbf{y}^{(i)}\| = 1$ for all i, and that the learning rate, the step size, s is a fixed positive real number. Suppose there exists a linear operator $\mathbf{W}^*$ such that $\|\mathbf{W}^*\|_F = 1$ and*

$$\gamma(\mathbf{W}^*, S, \phi) \geq \Gamma, \tag{32}$$

and the algorithm stops when the functional margin 1 is achieved.

1. *Then the number of updates made by Algorithm (1) is bounded by*

$$t \leq \frac{1}{\Gamma^2} \left(1 + \frac{2}{s}\right). \tag{33}$$

2. *Then for the solution* $\mathbf{W}_t$ *in Algorithm (1) we have*

$$\gamma(\mathbf{W}_t, S, \phi) \geq \frac{\Gamma}{s+2}. \tag{34}$$

Proof 1. 1. *Following the proof of the original Novikoff theorem [30], we first upper bound the norm of the matrix* $\mathbf{W}_t$ *obtained after* t *updates:*

$$\begin{aligned} \|\mathbf{W}_t\|_F^2 &= \|\mathbf{W}_{t-1}\|_F^2 + 2s\langle \mathbf{y}^{(i)}\mathbf{W}_{t-1}\boldsymbol{\phi}(x^{(i)})\rangle_{\mathcal{H}_y} + s^2\|\mathbf{y}^{(i)}\boldsymbol{\phi}(x^{(i)})^T\|_F^2 \\ &\leq \|\mathbf{W}_{t-1}\|_F^2 + 2s + s^2\|\mathbf{y}^{(i)}\|^2\|\boldsymbol{\phi}(x^{(i)})\|^2 \\ &\leq \|\mathbf{W}_{t-1}\|_F^2 + 2s + s^2 \\ &\leq ts(s+2). \end{aligned} \tag{35}$$

We now provide a reverse inequality for the inner product with $\mathbf{W}^\star$*:*

$$\begin{aligned} \langle \mathbf{W}_t, \mathbf{W}^\star\rangle_F &= \langle \mathbf{W}_{t-1}, \mathbf{W}^\star\rangle_F + s\left\langle \mathbf{y}^{(i)}\boldsymbol{\phi}(x^{(i)})^T, \mathbf{W}^\star\right\rangle_F \\ &= \langle \mathbf{W}_{t-1}, \mathbf{W}^\star\rangle_F + s\left\langle \mathbf{y}^{(i)}, \mathbf{W}^\star\boldsymbol{\phi}(x^{(i)})\right\rangle_{\mathcal{H}_y} \\ &\geq \langle \mathbf{W}_{t-1}, \mathbf{W}^\star\rangle_F + s\Gamma \\ &\geq ts\Gamma. \end{aligned}$$

Then we can create the squeezing inequality:

$$ts(s+2)\|\mathbf{W}^\star\|_F^2 \geq \|\mathbf{W}_t\|_F^2\|\mathbf{W}^\star\|_F^2 \geq \langle \mathbf{W}_t, \mathbf{W}^\star\rangle_F^2 \geq (ts\Gamma)^2. \tag{36}$$

implying the result.

2. *Taking the bound Eq.(33) for* t *and substituting into Eq.(35) we arrive at*

$$\|\mathbf{W}_t\|_F \leq \frac{s+2}{\Gamma}. \tag{37}$$

Then for the margin we have

$$\gamma(\mathbf{W}_t, S, \phi) \geq \min_{(\mathbf{y}^{(i)}, \mathbf{x}^{(i)})\in S} \frac{\langle \mathbf{y}^{(i)}, \mathbf{W}_t\boldsymbol{\phi}(\mathbf{x}^{(i)})\rangle_F}{\|\mathbf{W}_t\|_F} \tag{38}$$

$$\geq \frac{1}{\|\mathbf{W}_t\|_F} \tag{39}$$

$$\geq \frac{\Gamma}{s+2}, \tag{40}$$

which proves the statement.

Sparsity bounds [31] can also be used to translate this bound on the number of updates into a corresponding bound on the generalization of the resulting classifier.

All results included in this paper are assumed the normalization conditions, $\|\boldsymbol{\phi}(\mathbf{x}^{(i)})\| = 1$ and $\|\mathbf{y}^{(i)}\| = 1$, of Theorem 1. By forcing the normalization of $\|\mathbf{W}_t\|$ for all t in Algorithm 1 allows us to simplify and sharpen the proof of Theorem 1. In this case Expression (35) collapses into a identity of both sides of the equation, therefore instead of (36) we have

$$1 = \|\mathbf{W}_t\|_F^2\|\mathbf{W}^\star\|_F^2 \geq \langle \mathbf{W}_t, \mathbf{W}^\star\rangle_F^2 \geq (ts\Gamma)^2, \tag{41}$$

from which we gain that

$$t \leq \frac{1}{s\Gamma}, \tag{42}$$

Dataset	labels	Number of training instances	test instances	average labels
Corel5k	260	4500	500	3.3965
Espgame	268	18689	2081	4.6859
Iaprtc12	291	17665	1962	5.7187

Table 1: Statistics of three benchmark datasets.

and in case of the margin we can write

$$\gamma(\mathbf{W}_t, S, \phi) \quad \geq \quad 1, \tag{43}$$

which statements are significantly stronger than those appearing in the general case. The price that we need to pay for this result is the slower algorithm.

In comparing our algorithm with other online learning schemes of maximum margin based learning methods, e.g. SVM, (see some realizations in [32] and [33]), we need to bear in mind that our methods learns to predict all components of the label vector within one optimization problem. Those methods which can deal only with binary classification problems have to solve as many binary label subproblems as the number of labels independently, therefore their overall computational complexity turns to be significantly higher than our approach.

7. Experiment

7.1. Databases

In our experiments, we used three benchmark datasets, Corel5k, Espgame and Iaprtc12. These three datasets have been widely used in image annotation studies [7, 8, 9, 10, 3, 11] with performance evaluations reported therein. Therefore, we can easily compare our method with others. Statistics of three benchmark datasets are summarized in Table 1. Readers are referred to [3] for more details of three datasets.

7.2. Feature Extraction

In our experiment, we worked with 15 visual features extracted in [10]. More concretely, they contain one Gist descriptor, six global color histograms and eight histograms of local bag-of-words texture features [3]. The description of 15 features are summarized in Table 2. Readers are referred to [10] for more detail on extracting these features. These features were also used in [10] and [11]. A similar visual feature set without layout was extracted and used in [3], while 30 visual feature with spatial layouts were used in [9].

7.3. Evaluation metric

In our experiment, we evaluated annotation performance using *precision (P), recall (R), F-1 measure (F)*, which were commonly used in previous studies. For each tag, the precision is computed as ratio between the number of images assigned the tag correctly and total number of images predicted to have the tag, while the recall is the number of images assigned the tag correctly, divided by the number of images which truly have the tag. Then precision and recall are averaged across all tags. At last, F1 measure is calculated as $F = 2\frac{P \times R}{P+R}$.

7.4. Model selection

In three original databases, training/test data are already divided in advance. Therefore, given a learned model, there exist no variance in prediction performance on fixed test data. Hyper-parameters in Gaussian kernels, polynomial kernels and odds-ratio based kernels are found by cross validation restricted to the training data, namely it is divided into validation test and validation training parts. Then the learner is trained only on the validation training items. At the end those values of the parameters have been chosen which maximize the $F1$ score on the validation test.

[3] All features are available on http://lear.inrialpes.fr/people/guillaumin/data.php.

Feature	Dimension	Source	Descriptor	Location	Layout
DenseHue	100	texture	hue	dense	no
DenseHueV3H1	300	texture	hue	dense	yes
DenseSift	1000	texture	sift	dense	no
DenseSiftV3H1	3000	texture	sift	dense	yes
Gist	512	-	holistic	-	-
HarrisHue	100	texture	Hue	Harris points	no
HarrisHueV3H1	300	texture	Hue	Harris points	yes
HarrisSift	1000	texture	sift	Harris points	no
HarrisSiftV3H1	3000	texture	sift	Harris points	yes
Hsv	4096	color	HSV	-	no
HsvV3H1	5184	color	HSV	-	yes
Lab	4096	color	LAB	-	no
LabV3H1	5184	color	LAB	-	yes
Rgb	4096	color	RGB	-	no
RgbV3H1	5184	color	RGB	-	yes

Table 2: Description of 15 visual features tried in our experiments.

	Corel5k			Espgame			iaprtc12		
Feature	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
DenseHue	33.3	26.0	29.2	28.5	16.4	20.8	26.7	17.5	21.1
DenseHueV3H1	38.1	30.7	34.0	32.9	18.8	23.9	31.8	21.0	25.3
DenseSift	40.2	32.2	35.8	33.3	24.6	28.3	38.4	26.5	31.4
DenseSiftV3H1	**43.7**	**34.6**	**38.6**	**35.2**	**26.3**	**30.1**	**40.5**	**28.3**	**33.3**
Gist	33.7	26.9	29.9	28.3	20.6	23.9	33.2	23.5	27.5
HarrisHue	31.0	24.6	27.4	27.4	16.3	20.4	27.6	18.3	22.0
HarrisHueV3H1	34.5	27.7	30.7	31.5	18.4	23.2	31.9	21.9	26.0
HarrisSift	39.9	32.1	35.6	33.2	25.5	28.9	39.4	26.9	32.0
HarrisSiftV3H1	40.2	33.4	36.5	**34.6**	**26.2**	**29.8**	**40.7**	**29.7**	**34.3**
Hsv	38.3	30.6	34.0	30.0	18.7	23.1	32.6	21.1	25.7
HsvV3H1	40.8	33.8	37.0	33.8	21.6	26.4	35.4	24.1	28.7
Lab	35.1	27.5	30.8	27.2	16.4	20.5	28.4	17.9	22.0
LabV3H1	39.7	30.7	34.6	30.0	18.9	23.1	32.7	20.8	25.4
Rgb	42.0	33.4	37.2	26.2	16.4	20.2	32.8	20.6	25.3
RgbV3H1	**42.1**	**34.5**	**38.0**	29.6	19.2	23.3	35.7	23.0	28.0

Table 3: Performance of joint SVM without explicit output kernel on different individual features.

7.5. Selecting optimal features

In [10, 11], all 15 features were used for predicting tags. However, we believe that there exist some redundancies in all 15 features. Also, some features might be weakly relevant to the annotation task. A more efficient way is to identify a few most relevant features and use them for prediction. To this end, we apply joint SVM without explicit output kernel on different features, and list their discriminative abilities in Table 3, in which the best and second runner-up features are highlighted with bold font. We can see that `DenseSfitV3H1` is consistently more reliable than other features in three datasets. In addition, `HarrisSiftV3H1` is also optimal or close to optimal in Espgame and Iaprtc12 respectively. However, `HarrisSiftV3H1` is inferior to `RgbV3H1` in Corel5k. Therefore, in our later experiments, we used `DenseSfitV3H1`+`RgbV3H1` on Corel5k, while `DenseSfitV3H1`+`HarrisSiftV3H1` on Espgame and Iaprtc12. We combined two features by simply concatenating one feature vector after the other one.

	Training Time (sec)	Testing Time (sec)	Testing Performance		
			Precision (%)	Recall (%)	F1 (%)
Independent SVMs (Gau)	6285.11	117.20	15.3	22.1	18.1
Independent SVMs (Pol)	4612.23	147.9	15.1	29.7	20.0
Joint SVM (Gau)	80.68	**6.92**	40.8	37.1	38.9
Joint SVM (Pol)	**76.48**	9.11	**48.5**	**38.0**	**42.6**

Table 4: Comparison between one joint SVM and multiple SVMs on Corel5k dataset. Two input kernels (Gaussian and 2-degree polynomial) are tried in both learners.

	Corel5K			Espgame			Iaprtc12		
Method	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
MBRM [9]	24.0	25.0	24.0	18.0	19.0	18.0	24.0	23.0	23.0
JEC [3]	27.0	32.0	29.0	24.0	19.0	21.0	29.0	19.0	23.0
TagProp [10]	33.0	42.0	37.0	39.0	27.0	**32.0**	45.0	**34.0**	**39.0**
FastTag [11]	32.0	**43.0**	37.0	**46.0**	22.0	30.0	**47.0**	26.0	34.0
JSVM	**48.5**	38.0	**42.6**	32.7	**31.6**	**32.2**	42.2	29.4	34.6
JSVM+Odd	**48.8**	37.1	**42.2**	27.4	27.1	27.2	32.9	28.6	30.6
JSVM+Pol(2)	46.6	37.0	41.3	32.6	24.4	27.9	37.9	26.6	31.2
JSVM+Pol(3)	41.5	31.3	35.7	28.5	21.3	24.4	38.0	26.1	31.0
JSVM-Per	37.5	29.8	33.2	25.0	19.0	21.6	29.2	20.8	24.3

Table 5: Comparison between different versions of joint SVM and other related methods on three benchmark databases.

7.6. Comparison with Independent SVMs

At first, we applied both a joint SVM, and many independent SVMs on Corel5k dataset with the feature combination selected above. To ensure fairness, no user-designed kernel is used on output for the joint SVM (plain joint SVM), while Gaussian kernel and 2-degree polynomial kernel are tried for inputs in both learners. In the learning phase, the optimization problems (Eq.(7)) and (Eq.(12)) were solved with the same coordinate descent method [20]. In addition, the same cross-validation procedure is used for both many independent SVMs and the joint SVM to find the best hyper-parameters C, d, c, σ. To measure the efficiency, training and testing time were recorded as well. All experiments were run on the same simulation and hardware conditions (Python 3, Intel Core i7). The comparison of accuracy and efficiency between independent SVMs and joint SVM is presented in Table 4. While the learning and testing time of independent SVMs scale with the number of tags, the computation time of joint SVM approximately equals a SVM for single-tag classification. At the same time, in terms of accuracy, joint SVM also worked much better than independent SVMs. We can also see that 2-degree polynomial input kernel worked better than Gaussian input kernel for both learners.

7.7. Comparison with state-of-the-art

More intensive experiments of joint SVM were conducted with different pre-designed, explicit output kernels: odds-ratio based kernel (JSVM+Odd), 2-degree polynomial (JSVM+Pol(2)), 3-degree polynomial (JSVM+Pol(3)). Also, online learning algorithm of joint SVM (JSVM-Per) was also implemented. All configurations were run on all three datasets, with optimal feature combination and 2-degree polynomial kernel on inputs. The experimental results, together with the reported results from other related work, are presented in Table 5. We can see that plain joint SVM (JSVM) outperforms all other results on Corel5k and Espgame datasets, yielding the best results so far. JSVM is also the second best result on Iaprtc12 dataset. The results of JSVM+Odd and JSVM+Pol(2) are similar on all three datasets. It is worth noting that JSVM+Odd and JSVM+Pol(2) also worked better than previous methods by a large margin. Meanwhile, JSVM+Pol(3) is worse than JSVM+Pol(2). JSVM-Per's performance is inferior to other JSVM versions, although it is still better than two classic methods [9, 3].

7.8. Discussions

Based on our experiments, it seems that plain joint SVM (JSVM) works more robustly than the joint SVMs with explicit output kernels. In order to dig deeper to find an explanation, we can study the correlation matrices of output tag-sets in three datasets. In Figure 3, for each dataset, we plot the histograms (in log scale) of all correlation values in both training sets and testing sets. We can see that most entries in correlation matrices are 0, which means that the pairwise correlation (or roughly speaking, dependencies) is rather sparse. Although JSVM, JSVM+Odd both encode linear pairwise dependencies, it should be reminded that the implicit output kernel in JSVM is in regularization term, which implies that simpler output kernels (dependencies) are encouraged. However, JSVM+Odd does not have this preference. Therefore, JSVM can implicitly learned most simple output kernels when no more complex ones are needed. Analogously, the same principle can explain why JSVM+Pol(2), or even JSVM+Pol(3) led to worse results. If we look closer, we can observe that in Corel5k datasets, stronger correlations are displayed in its testing set, and correspondingly, the performance gaps between JSVM, JSVM+Odd and JSVM+Pol(2) are also rather small.

As for JSVM-Per, one reason of its inferiority is that the regularization is computed instance-wisely, which might conflict the global effect it is supposed to have. However, we gain tractability, for extremely large datasets, with acceptable accuracy cost. As a future direction work, we will investigate some alternative online regularization strategies.

8. Conclusions

A novel joint SVM was presented for automatic image tagging. It is superior to conventional SVMs based on our empirical results. In particular, it compares favorably with state-of-the-art methods. As possible future work directions, we would like to apply and improve joint SVM in other multi-label learning domains.

Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

Reference

[1] S. Szedmak, J. Shawe-taylor, Learning via linear operators: Maximum margin regression, Tech. rep., University of Southampton, UK (2005).

[2] I. Tsochantaridis, T. Hofmann, T. Joachims, Y. Altun, Support vector machine learning for interdependent and structured output spaces, in: ICML, 2004.

[3] A. Makadia, V. Pavlovic, S. Kumar, Baselines for image annotation, International Journal of Computer Vision 90 (2010) 88–105.

[4] B. Hariharan, S. V. N. Vishwanathan, M. Varma, Efficient max-margin multi-label classification with applications to zero-shot learning, Machine Learning 88 (1-2) (2012) 127–155.

[5] A. Argyriou, T. Evgeniou, M. Pontil, Convex multi-task feature learning, Machine Learning 73 (3) (2008) 243–272.

[6] B. Taskar, V. Chatalbashev, D. Koller, C. Guestrin, Learning structured prediction models: A large margin approach, in: ICML, 2005.

[7] V. Lavrenko, R. Manmatha, J. Jeon, A model for learning the semantics of pictures, in: NIPs, 2004.

[8] D. M. Blei, M. I. Jordan, Modeling annotated data, in: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, 2003.

[9] S. L. Feng, R. Manmatha, V. Lavrenko, Multiple bernoulli relevance models for image and video annotation, in: CVPR, 2004.

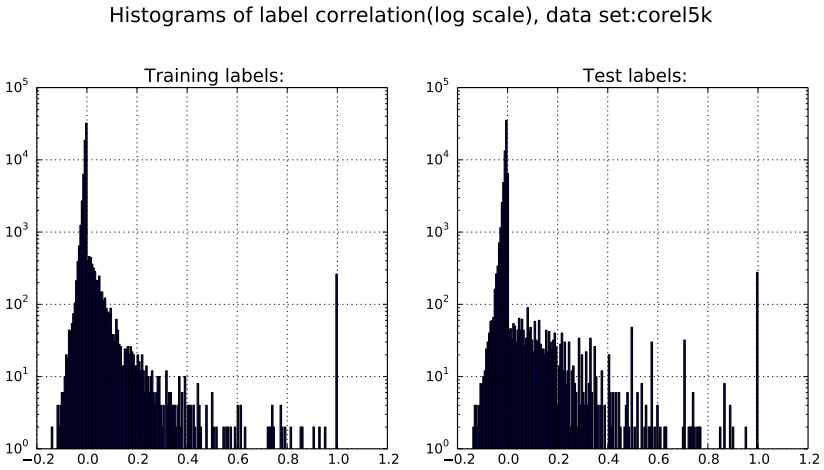
[10] M. Guillaumin, T. Mensink, J. Verbeek, C. Schmid, Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation, in: ICCV, 2009.

[11] M. Chen, A. Zheng, K. Q. Weinberger, Fast image tagging, in: ICML, 2013.

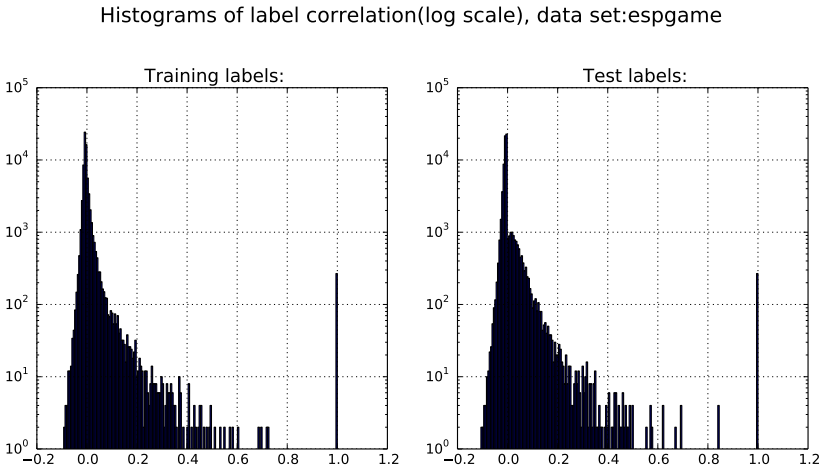
[12] D. R. Hardoon, S. Szedmak, J. Shawe-Taylor, Canonical correlation analysis: An overview with application to learning methods, Neural Computation 16 (2004) 2639–2664.

[13] X. Qi, Y. Han, Incorporating multiple svms for automatic image annotation, Pattern Recogn. 40 (2) (2007) 728–741.

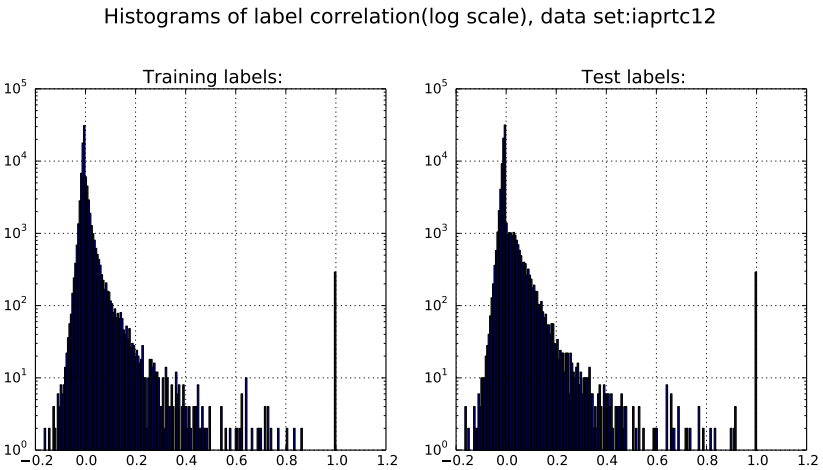
[14] S. Szedmák, T. D. Bie, D. R. Hardoon, A metamorphosis of canonical correlation analysis into multivariate maximum margin learning, in: ESANN, 2007.
[15] J. Rousu, C. Saunders, S. Szedmák, J. Shawe-Taylor, Kernel-based learning of hierarchical multilabel classification models, Journal of Machine Learning Research 7 (2006) 1601–1626.
[16] Y. Zhang, D.-Y. Yeung, Multilabel relationship learning, ACM Trans. Knowl. Discov. Data 7 (2) (2013) 1–30.
[17] A. Elisseeff, J. Weston, A kernel method for multi-labelled classification, in: NIPs, 2001.
[18] Y. Guo, D. Schuurmans, Multi-label classification with output kernels, in: ECML/PKDD, 2013.
[19] V. Vapnik, Statistical learning theory, Wiley, 1998.
[20] F. Dinuzzo, C. S. Ong, P. V. Gehler, G. Pillonetto, Learning output kernels with block coordinate descent, in: ICML, 2011.
[21] S. M. Hailpern, P. F. Visintainer, Odds ratios and logistic regression: further examples of their use and interpretation, Stata Journal 3 (3) (2003) 213–225.
[22] K. Astikainen, L. Holm, E. Pitkänen, S. Szedmak, J. Rousu, Towards structured output prediction of enzyme function, in: BMC Proceedings, 2(4):S2, 2008.
[23] P. D. Grünwald, The Minimum Description Length Principle, MIT Press, 2007.
[24] D. Bertsekas, Nonlinear Programming, 2nd Edition, Athena Scienctific, 1999.
[25] K. Kiwiel, Convergence of approximate and incremental subgradient methods for convex optimization, Journal of Optimization 14, 3 (2004) 807–840.
[26] N. Cristianini, J. Shawe-Taylor, An introduction to Support Vector Machines and other kernel-based learning methods, Cambridge University Press, 2000.
[27] N. Cesa-Bianchi, G. Lugosi, Prediction, Learning and Games, Cambridge University Press, 2006.
[28] C. Gentile, A new approximate maximal margin classification algorithm, Journal of Machine Learning Research 2 (2001) 213–242.
[29] Y. Li, H. Zaragoza, R. Herbich, J. Shawe-Taylor, J. Kandola, The perceptron algorithm with uneven margins, in: Proceedings of the International Conference of Machine Learning (ICML'2002), 2002.
[30] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods, Cambridge University Press, New York, NY, USA, 2000.
[31] T. Graepel, R. Herbrich, J. Shawe-Taylor, Generalisation error bounds for sparse linear classifiers, in: Proceedings of the Thirteenth Annual Conference on Computational Learning Theory, Morgan Kaufmann Publishers Inc., 2000, pp. 298–303.
[32] A. Bordes, S. Ertekin, J. Weston, L. Bottou, Fast kernel classifiers with online and active learning, Journal of Machine Learning Research 6(Sep) (2005) 1579–1619.
[33] S. Shalev-Shwartz, T. Zhang, Stochastic dual coordinate ascent methods for regularized loss minimization, Journal of Machine Learning Research 14(Feb) (2013) 567–599.



(a)



(b)



(c)

Figure 3: The histograms (in log scale) of all correlation values in both training sets and testing sets: (a) Corel5k, (b) Espgame (c) Iartc12.