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. Such a task requires learning complex relations with sparse, incomplete and noisy information. Knowledge Propagation approach, can deal with such relations where affordance predictions can be propagated through exploiting the similarities among object properties, action parameters and generated effects. The high complexity of affordance representation can be addressed through use of Maximum Margin Multi Valued Regression (MMMVR) which is a large scale approach to complex problems of several layers. With increased variety and size of object database and addition of other parametric combinatory actions, we expect to achieve complex systems which truly uses 'structural' bootstrapping in its lifelong learning and development.**

In this paper, we extended MMMVR for learning of paired-object affordances, i.e. for predicting the effects of the actions that are applied object pairs. In the experiments, we evaluated this method with a dataset composed of 83 objects and 83$\times$83 interactions. We compared the prediction performance with standard classifiers that predict the effect category given object pair's low-level features or single-object affordances. The experiments showed that 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 infite-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 conform 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 is introduced by Szedmak et al. [1]. That general framework has been applied for recommender systems in [2] and [3]. The 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 other ones need to be predicted. We face similar problem in learning the 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 for the recommender systems. We will call this approach Knowledge Propagation in the rest of the paper.

The learning task in this paper is to predict the effect of an action that is applied on 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.

In our paper, we will compare three different approaches with the aim of to predicting the effect of *stack* 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' are interacting with other objects, and [6] where two-object relational interaction models were directly learned. However none of these studies attempted to bootstrap their multi-object affordance learning system through knowledge propagation.

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 indexed by labels of objects. The elements of the matrix expresses 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 system, graphs, vectors. 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 could 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 other kind of sources 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 a certain similarity measures, and along these connections the knowledge can be transferred from the outcome of the small number 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 [7].

The base learning method, described in the Section II, can predict the missing elements of the matrix, and can provide confidences to prediction of each missing elements. The method used has been grown out of a combination of different maximum margin based structured learning approaches. One group of those models build up on Markov Random Fields, see for example [8], [9], [10] and [11]. 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, see in [12] and [13]. In [13] the performance of models of both groups directly compared in predicting hierarchical structures. The approach used in this paper based on a synthesis of those mentioned above, and developed to tackle very large data set, millions of potential interactions, see details in the Appendix, **??**, and in the following papers, [1], [2] and [3].

II. Method

In this section, we will explain the methods used to learn paired-object affordances and how we can increase the speed through Active 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 predicting 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 predicting 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 standard manually coded features computed mostly from visual perception with no explicit link to robot's actions.
- Affordance-features encodes 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 corresponds to label of the objects. Known object-ids' can be computed based on object feature similarity or can be given by the 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. Knowledge Propagation method detailed below can propagate the effect information through 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 but it can deal with vectorial output. Fig. 1 shows the main differences between SVM's and MMR's. For more details, please refer to [1], [13].

C. Knowledge Propagation based relational Classifier (KnowPropC)

This classifier is used to learn the effect category of interactions given the two object identifiers.

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.

Given two sets $\mathcal{B}$ and $\mathcal{U}$, we can assume that they are finite ones. Let $\mathscr{R}$ be an arbitrary relation between $\mathcal{B}$ and $\mathcal{U}$ 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 which describe how the items in that pair relate to each other. This information can be obtained by some experiments and for a pair (b, u) it is described by z_{bu} taken out of 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

	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.

or not to each other, or by a real number, e.g. what is 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 stack action, both $\mathcal{B}$ and $\mathcal{U}$ will correspond objects, the one being dropped and the other one on the table. z_{bu} on the other hand will refer to the effect of the stack action being applied on these objects.

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

An application of the learning framework in developing recommender systems can be found in [2] and [3].

The formal 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 relations 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.
- For all b $\mathcal{D}_b=\{u|(b,u)\in\mathcal{D}\}$ is a cover of the set $\mathcal{U}$, and for all u $\mathcal{D}_u=\{b|(b,u)\in\mathcal{D}\}$ is a cover of the set $\mathcal{B}$.
  Cover means a set of subsets $\{\mathcal{U}|\mathcal{U}\subseteq\mathcal{X}\}$ of a given set $\mathcal{X}$ whose union contains the original set, namely $\mathcal{X}=\bigcup_{\mathcal{U}\in\mathcal{S}}\mathcal{U}$.
- $\phi_Z:\mathcal{Z}\to\mathcal{H}_z$ 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 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) to 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 $F:\mathcal{H}_F\times\mathcal{H}_B\times\mathcal{H}_U\to\mathbb{R}$, which has higher value if the feature vectors of two items b and u can predict better 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 therefore can be expressed as

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

where $\mathbf{W}$ as a tensor describing the function itself and the variables are connected by tensor product, see details for example in [14].

We can rewrite function F as

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

where $\mathbf{W}$ plays a 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 image of that map $\mathbf{W}(\phi_B(b)\otimes\phi_U(u))$ and $\phi_Z(z_{bu})$. Now this inner product can have higher value if correlation between the image vector $\mathbf{W}(\phi_B(b)\otimes\phi_U(u))$ and the outcome of the match $\phi_Z(z_{bu})$ is higher.

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

- To solve the problem when the cardinality of the observed tuples, i.e. 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 ten thousands as well, an 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, a mixture of plenty of different distributions.

To overcome on these potential difficulties we can introduce a learning model which decomposes the entire problem into weakly coupled subproblems. To this end consider a cover of the one of the sets $\mathcal{B}^{(o)}$ and $\mathcal{U}^{(o)}$ say $\mathcal{B}^{(o)}$, obviously the role of $\mathcal{B}^{(o)}$ and $\mathcal{U}^{(o)}$ in what follows can be exchanged. Let $\mathcal{I}$ be a finite index set, then the cover is given by sets $\{\mathcal{B}_i^{(o)}\}$, $\mathcal{B}_i^{(o)} \subseteq \mathcal{B}^{(o)}$, $\mathcal{B}^{(o)} = \bigcup_i \mathcal{B}_i^{(o)}$, $i \in \mathcal{I}$. There is learner $\mathscr{L}_i$ defined on each of the sets $\mathcal{B}_i^{(o)}$, given by the multilinear function $F_i : \mathcal{B}_i^{(o)} \times \{z_{bu} | b \in \mathcal{B}_i^{(o)},\ u \in \mathcal{D}_b^{(o)}\} \to \mathbb{R}$. Each of the multilinear function is given by the linear operator $\mathbf{W}_i$. We might say the domain of learner $\mathscr{L}_i$ is $\mathcal{B}_i^{(o)}$ since if $\mathcal{B}_i^{(o)}$ and $\mathcal{D}^{(o)}$ are given then the entire domain of the multilinear function F_i is determinated.

We define the conditions connecting the learners into one assemble. Let L_i be a learner dependent loss function, and it is defined by using a version of the hinge loss

$$L_i(b, z_{bu}) = \begin{cases} 0 & \text{if } F_i(b, z_{bu}) \geq 1, \\ \max_u(1 - F_i(b, z_{bu})) & \text{otherwise,} \end{cases}$$
$$b \in \mathcal{B}_i^{(o)}, u \in \mathcal{D}_b^{(o)}. \tag{5}$$

As a consequence of the definition L_i does not depend directly on u, obviously the implicit dependency via the function F_i remained valid, thus it depends on b and F_i. Now we then require for any subset of the learners indexed by $\mathcal{J} \subseteq \mathcal{I}$, and satisfying that $\bigcap_{i \in \mathcal{J}} \mathcal{B}_i^{(o)} \neq \emptyset$

$$L_i(b, F_i) = L_j(b, F_j),\ \forall b \in \bigcap \mathcal{B}_i^{(o)},\ \text{and } \forall i, j \in \mathcal{J}, \tag{6}$$

hence the loss on a b has to be the same for all learners which domain contains that b. We might rephrase this condition by saying that there is an nonnegative additive loss measure define on $\mathcal{B}^{(o)}$.

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

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\mathbf{W}\|_i^2 + C \textstyle\sum_{b \in \mathcal{B}^{(o)}} \xi_b \\ \text{w.r.t.} \quad & \mathbf{W}_i \in (\mathcal{H}_Z \otimes \mathcal{H}_B \otimes \mathcal{H}_B)^* \\ \text{s.t.} \quad & \langle \phi_Z(z_{bu}), \mathbf{W}_i(\phi_B(b)) \rangle \geq 1 - \xi_b,\ b \in \mathcal{B}_i^{(o)},\ u \in \mathcal{D}_b^{(o)}, \\ & \xi_b \geq 0, ,\ b \in \mathcal{B}_i^{(o)}, \end{aligned} \tag{7}$$

There is an obvious simple way to define the index set $\mathcal{I}$ by saying it is equal to $\mathcal{U}^{(o)}$. It means in our example, thus every object has its own learner and every action identifier has its own loss function.

D. Active Learning

To implement the knowledge propagation an active leaning based algorithm is applied. The main steps of this algorithm are summarized here.

1) Start on a small subset of all object pairs as training of the learner.
2) Train the learner on the available outcomes.
3) Predict all untried elements of the matrix, and compute confidences to those predictions.
4) Choose that untried pair of objects for which the confidence is the smallest one, 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 included into the training set. If the confidence is high then that element provides not much useful information since that element highly similar to those ones appearing in the training set. The low confidence marks the least similar elements, thus they can yield sufficient new information about the general structure of the interactions. Technically, the learner yields a distribution on the possible outcome categories, if the entropy of this distribution is high, the categories are predicted by closely the same probabilities, then the confidence of the prediction is low.

III. Experiments

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

A. Experiment Setup

The robot system is composed of a 7 DOF Kuka Light Weight Robot (LWR) arm placed on a vertical bar similar to human arm, a A 7 DOF 3 fingered Schunk gripper mounted to the robot arm, and a Kinect sensor placed over the 'torso' with a view of the table in front of the robot. 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 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.

1) Interaction Dataset: We collected data from 83 objects (Fig. 2) by placing them on the table in front of our robot. In order to analyze our learning algorithms, we aimed to create an interaction database composed of (object, action, effect) tuples with the action repertoire mentioned above. In order to collect this database, the 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 to fill-up the complete table through his observations. In cases where the effect is difficult to assess, the human 'simulated' 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, '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

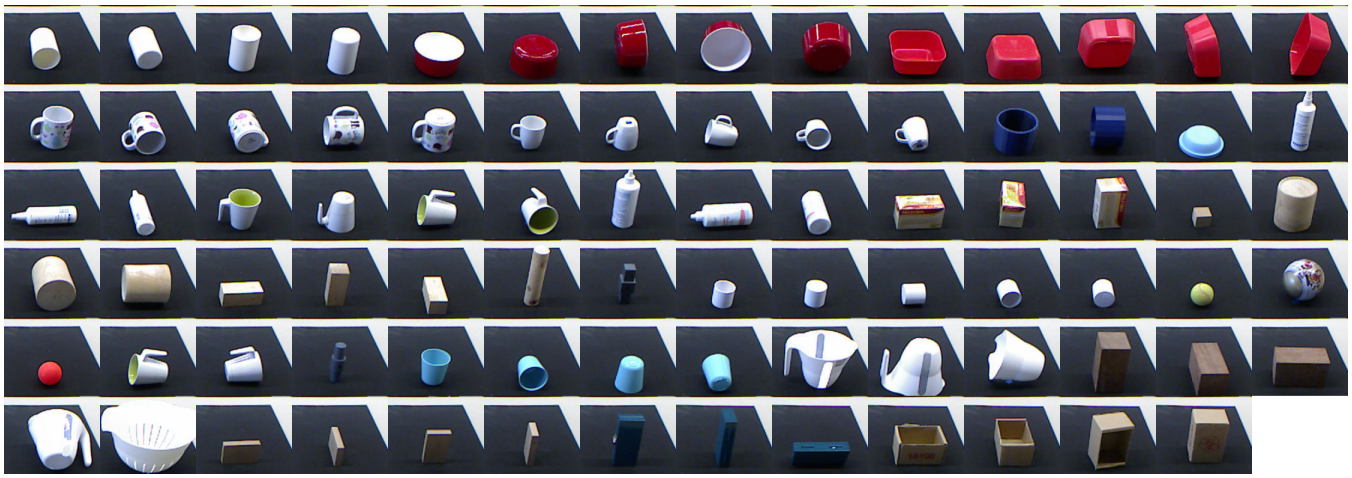

Fig. 2. Objects used in the experiments. Any object-orientation pair is assigned to a new object index in the experiments.

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[1]:

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

3) Object representation: The objects are segmented based on depth information of Kinect sensor that is place over 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 feature vector composed of shape, size and local distance related features 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[2]. 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 $dist$.

[1]Although in this paper the actions and their effect categories are manually coded, we showed that a robot can self-discover action primitives and effect categories through exploration in [15] and [16], respectively. Thus, we can safely assume that these skills were learned in previous stages.

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

- *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 ε^o refers to the effect categories ($\in$ {pushed,rolled,resisted,no-change}) of the corresponding poke 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*) in [15].

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

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

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

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

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

B. Action propagation results

In this section, we will compare the performance of the classifiers that are trained with different object descriptors: basic-features based MMC, affordance-features 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 stack interaction dataset, that is composed of 83×83 samples of (object, effect) tuples, 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.

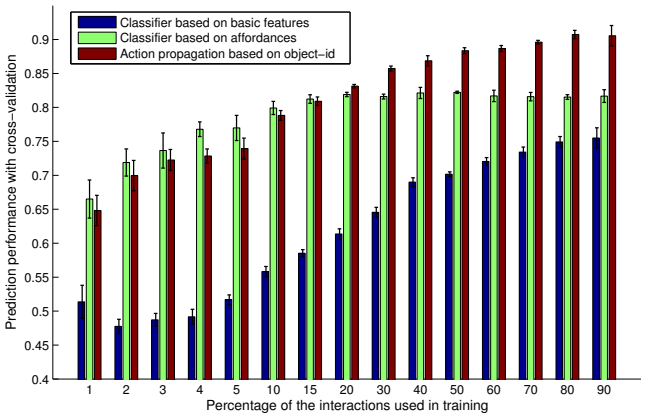

Fig. 3. The prediction performance of the classifiers based on basic features and affordances, and action propagation system based on object id. All predictors predict the effect of stack action in a database composed of 83 objects and 83×83 interactions in total.

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 *afford-feat* based classifiers performance convergence with a training set of around 15%, $basicf$ based classifiers performance smoothly increases with increasing training size and becomes similar to *afford-feat* performance at the end. On the other hand *object-id* based classifiers performance increases continuously and it outperforms *afford-feat* performance at the end. We can interpret the 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 with no link to 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.
- *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 knowledge propagation based 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 one.
- 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 initial performance might be mere coincidence but the significantly higher performance of *object-id* based classifiers is a result of the powerful knowledge propagation mechanism.

C. Bootstrapping through active learning results

In this section, we evaluated the effect of active learning in prediction performance of id based classifier. Starting with a random set of few elements (1%), we increased the training sample set 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. We suggest Then, after around 6% of the dataset, the performance boosts up and quickly surpasses the one without active learning. The initial phase of slow learning is related to exploration-exploitation dillema, i.e. it can be viewed as preparing the underlying structure of KnowPropC for the active learner for bootstrapping in the latter phases.

IV. Conclusion

In this study, we introduced a method that was successfully used in recommender systems to 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 standard' approaches, state-of-the-art classifiers that use low-level object features or previously learned object affordances as input attributes. We already studied the bootstrapping effect of affordance features in detail in [17] where effects were obtained from real robot interactions. 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 the standard classifiers especially when active learning is applied.

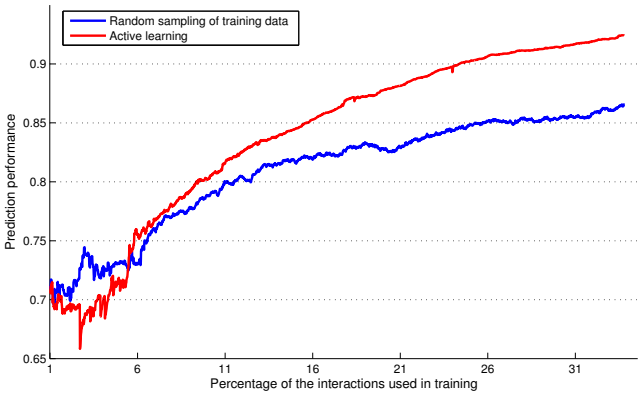

Fig. 4. The prediction performance of predictors that are trained with increasing number of training samples that is either randomly sampled from the dataset or selected according to active learning criteria. Both classifiers use randomly sampled initial training sets to achieve a connected underlying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

The 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 introduce 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 using object affordance instead of low-level features, or knowledge propagation based classification with active learning significantly speeds up learning. Yet, we need datasets to test and analyses our algorithms. In future, we plan to use physics based simulators for this purpose with the condition 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 in
`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] O. Chapelle, B. Schölkopf, and A. Z. Editors, *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2010.

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

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

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

[15] 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.

[16] 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.

[17] E. Ugur, S. Szedmak, and J. Piater, "Bootstrapping paired-object affordance learning with learned single-affordance features," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, submitted.