EU FP7 CogX
ICT-215181
May 1 2008 (52months)

DR 5.2: Continuous learning of cross modal concepts

Danijel Skočaj, Matej Kristan, Alen Vrečko, Barry Ridge, Aleš Leonardis, Sergio Roa, and Geert-Jan Kruijff

University of Ljubljana, DFKI Saarbrücken
⟨danijel.skocaj@fri.uni-lj.si⟩

Due date of deliverable:	31 July 2010
Actual submission date:	29 July 2010
Lead partner:	UL
Revision:	final
Dissemination level:	PU

Cross-modal learning is an important characteristic of a system that is supposed to be capable of self-extension. The system should exploit different modalities and extend its current knowledge based on the information obtained from different sources. In this deliverable we address the cross-modal learning from different perspectives. We present methods that facilitate learning in one modality by being supervised by another modality. We also present approaches for self-supervised cross-modal learning on a lower level, on the level of extracted features. And finally, we also address the problem of high-level cross-modal learning, which is tightly related to the problem of cross-modal binding.

Executive Summary

An important characteristic of a robot that operates in a real-life environment is the ability to expand its current knowledge - and has to do so continuously, in a life-long manner. The system has to create and extend concepts by observing the environment while interacting with this environment as well as with other cognitive agents and humans. Interactive continuous learning, which is the main research topic of Workpackage 5, is therefore an essential characteristic of a self-extending cognitive system. In this deliverable we focus on continuous learning of cross-modal concepts.

Different types of cross-modality are addressed in this deliverable. Firstly, we address the cross-modal learning between language and vision; linguistic descriptions provide information, which drives learning in the visual subsystem. Next, we present cross-modal learning of object affordances and action effects; here the information arising from the visual subsystem is combined with the information from the manipulation subsystem. Also, different derived modalities, or cues, from the visual subsystem are taken into account: colour, depth (which is converted in 3D point cloud), and motion. And finally, we also address an approach that finds associations between higher-level modal concepts from different modalities and could be used for binding.

We have integrated many of the developed methods in a multi-modal cognitive system based on CAS. Continuous cross-modal learning is an important aspect of the George system we have been developing, so several approaches presented in this report will be demonstrated in the George scenario. Moreover, all the methods presented here were also individually extensively evaluated on different problem domains.

Some of the work presented in this deliverable is a continuation of the work performed in Year 1 and mostly presented in the deliverable DR.5.1. *Continuous learning of basic visual concepts*, while this deliverable also presents work that has been initiated in Year 2. In both cases, the novelty and the value added in Year 2 are clearly exposed in the sections below. This deliverable is also highly highly related to the DR.5.3 *Representations of gaps in categorical knowledge*, since detection and representations of knowledge gaps is an integral part of interactive continuous cross-modal learning. While the deliverable DR.5.3 focuses on mechanisms for detection of incompleteness in knowledge, this deliverable focuses on mechanisms for learning.

The work performed and described here addresses the problem of learning basic visual and cross-modal concepts in an interaction between a robot and a tutor and between the robot and objects in the scene. The work has been mainly performed as envisioned in the workplan and forms a solid basis for further research and extensions in the direction of more general and more active interactive learning of cross-modal concepts.

Role of Continuous learning of cross modal concepts in CogX

In the process of continuous interactive cross-modal learning, the system tries to understand what it does know and what it does not, and act or update the current knowledge accordingly. Therefore, the main research topic fits very well with the main motto of the project: to self-understand to be able to self-extend.

Contribution to the CogX scenarios and prototypes

In order to monitor and show progress on interactive continuous learning, we have designed the George scenario (Interactive cross-modal learning scenario) [24, 25]. This scenario has been designed as a use case for guiding and testing system-wide research and for demonstrating methods developed in WP 5 (and also some other workpackages) in a working system. Therefore, many of the methods presented in this deliverable have been integrated into the overall system, which is used in the George scenario.

An active learning loop in which a robot semi-autonomously selects actions that improve its prediction abilities was implemented for the Dexter scenario. These sensorimotor models encode the causal relationships between motor actions and the consequent object behaviour.

1 Tasks, objectives, results

1.1 Planned work

This deliverable mainly tackles the problems addressed in Task 5.2 of Workpackage 5:

> *Task 5.2: Continuous learning of cross-modal concepts. Extend the system to consider features of other modalities and to build cross-modal category systems. Analyse the trade-offs between unsupervised and supervised learning.*

Therefore, the main goal was to develop the theory and methods to be integrated into a robot capable of interactive continuous learning of cross-modal concepts.

Before we begin with the concrete plans we had set for this deliverable, let us first discuss some terminological issues. The terms related to cross-modal learning are not consistently used in the literature. First, let us cite, how we defined the term cross-modal learning (and modality) in our contribution about cross-modal learning in the Encyclopedia of the Sciences of Learning (see also Annex 2.6 [26]):

> Cross-modal learning refers to any kind of learning that involves information obtained from more than one modality. In the literature the term modality typically refers to a sensory modality, also known as stimulus modality. A stimulus modality provides information obtained from a particular sensorial input, for example visual, auditory, olfactory, or kinesthetic information. Examples from artificial cognitive systems (”robots”) include also information about detected range (by sonar or laser range-finders), movement (by odometry sensors), or motor state (by proprioceptive sensors). We adopt here the notion of modality that includes both the sensorial data, and further interpretations of that data within the modality. For example, from a pair of (depth-calibrated) images, a cloud of points in 3-dimensional space can be computed. We obtain both types of data (the image data, and the 3D points) from the same visual sensor. At the same time, they differ in what information they provide. We consider information sources derived from sensorial data as derived modalities that by themselves can be involved again in cross-modal learning.

We have, therefore, adopted a wider meaning of the term modality, and also of the term cross-modal learning. We discuss in [26] several types of cross-modal learning. In weakly-coupled cross-modal learning the models are built within individual modalities, and the other modalities only supervise learning, by, e.g., providing a label or reinforcement signal. In closely-coupled cross-modal learning, learning processes are more intertwined. A model is learnt by combining information from different modalities into a common level of representation, and then using this level as a starting point to build a common cross-modal classifier or predictor. We can identify also a third type of cross-modal learning that is performed on a higher level of abstraction. Here, a model is acquired that connects modal conceptual structures from different modalities by learning associations between them.

Coarsely according to these division we also performed our work within Task 5.2, which is reported here. In the following, we first present several mutually dependent research lines related to different types of cross-modal learning as depicted in Fig. 1. Here, three modalities or main sources of information are presented: vison (2D and 3D), audio (language), and manipulation (action, proprioceptive data). The data that is captured in each of these modalities is then further processed and can be even transformed in 'derived modalities' (depicted as two light grey matrices in the vision modality in Fig. 1 representing, e.g., colour and 3D information). All different modalities continually provide a huge amount of information, much of which is possibly highly redundant. This information is then processed and different kinds of low-level features or other types of structured (the most relevant) data are extracted (depicted as small blue vectors in Fig. 1). Based on these features, high-level modality-specific concepts are then formed (depicted as red circles in Fig. 1); i.e., in vision this would mean that an adequate representation of a visual concept has been learned, while in language a logical form containing linguistic concepts has been formed. Based on this description we can identify several different types of cross-modal learning, which in different ways involve information from several modalities:

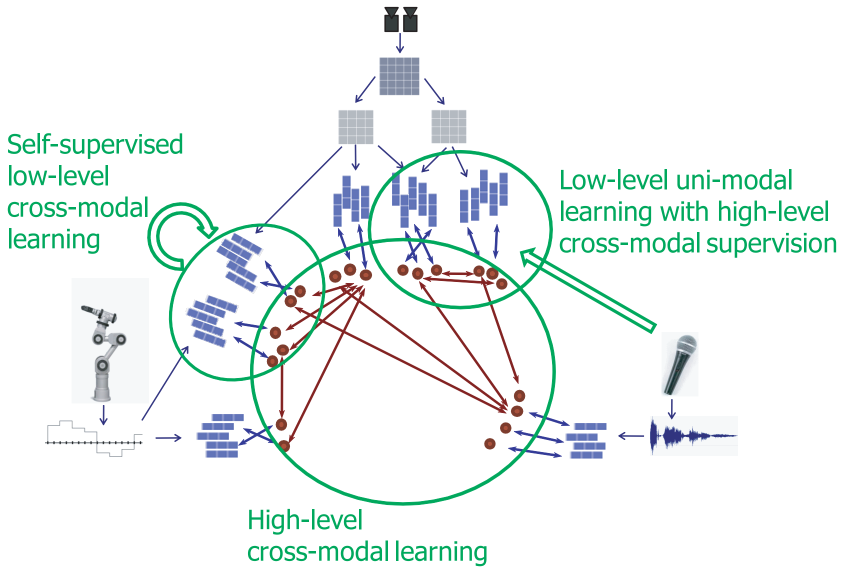


Figure 1: Different types of cross-modal learning.

- **Low-level uni-modal learning with high-level cross-modal supervision.** The primary goal was to learn associations between extracted low level features and modal concepts, in particular to learn association between automatically extracted low-level visual features and visual concepts, such as objects, object colour, and shape. The learning was therefore performed within a single (visual) modality (although considering several derived modalities like colour and 3D information), however this learning was driven by another modality or source of information, namely language. The training samples are, therefore, generated in interaction with the tutor.

- **Self-supervised low-level cross-modal learning.** The main goal here was to learn associations between extracted low-level visual, motion, and action features and concepts of simple object affordances and action effects. Here, the cross-modal learning takes place on a lower level; the information from different modalities is combined on the feature level already. This learning process does not require any supervision from other modalities on a higher level. The training samples are autonomously generated in interaction with an object by performing actions.

- **High-level cross-modal learning.** This type of cross-modal learning is performed at a higher level of abstraction. The aim of the learner is to learn associations between different high level modal representations (stemming from different modalities) of the same concepts and in this way improve the system's ability to relate different modal and a-modal representations of perceived instances. This type of knowledge is critical for the process of cross-modal binding. The training samples are generated on the basis of the robot's beliefs and successful past bindings. An example: if the colour of an object is assigned to the visual *colour model 1* by the uni-modal learner and the tutor attributes the colour of the same object to the linguistic *concept of red*, a cross-modal association between visual *colour model 1* and linguistic *concept of red* is made.

For each type of cross-modal learning we now briefly describe the concrete goals we had set were to be performed under the Task 5.2. The actual work performed is then described in the following section.

- **Low-level uni-modal learning with high-level cross-modal supervision.**

  Our algorithm for learning visual concepts (by being supervised by language) is based on generative representations modelled by Kernel Density Estimation. The aim was to further investigate the methodology for estimation of generative models from streams of data. We had planned to evaluate in detail the properties of the multivariate online Kernel Density Estimation (DR.5.1, Annex 2.3), which we had proposed in the previous year. Based on this analysis we planned to extend the framework and improve it for the task of learning generative models, but this time explicitly for the task of classification. In this way, we aimed at online generative discriminative models. Once the theory for the online discriminative models would be laid out, we planned to address the detection of uncertainties in the discriminative models in relation with the question of online learning in the "open world".

- **Self-supervised low-level cross-modal learning.**

  When the robotic system is interacting with objects in its environment, it is continuously gathering feature information in several sensory modalities. With regard to object affordance learning, one of our goals was to design a learning mechanism that could harness the statistical information in the data stream of a particular modality in order to recognise emerging affordance categories on-the-fly, and to use this information to train classifiers in other modalities. In particular, if one of the sensory modalities represents the effects of actions on objects, then effect categories, e.g. rolling versus non-rolling objects, could be discovered therein by clustering motion-specific features, e.g. object velocity or the distance traveled by the object after an action. Meanwhile, as co-occurring information is being gathered in a separate modality that represents the properties of objects, e.g. 2D and 3D shape features, the category information from the effect modality as well as the inter-modality co-occurrence information could be used to efficiently train a classifier. Given a number of interactive episodes with objects, the goal was to enable the system to predict from observed object properties the affordance classes of novel objects based on its own self-generated object affordance categories grounded in the effect modality.

  An additional goal is the acquisition of generative regression models through semiautonomous learning. Following the work in the first year, the specific objective was the incorporation of active learning experiments in the context of affordance learning with recurrent neural networks in a robot arm scenario. In the previous year, offline experiments were performed and some statistics were collected to analyse the generalisation abilities of LSTM machines for these tasks [19]. We planned to use the Intelligent Adaptive Curiosity method [16] for active selection of samples for learning. This algorithm was already used in our previous experiments [18], but we planned to adapt it for the problem of object trajectory prediction after a pushing action. This problem involves sequences of features in a time-series prediction setting, where the neural network represents a model of the underlying dynamical system.

- **High-level cross-modal learning.**

  The aim was to first define the problem of High-level cross-modal learning and the problem of binding, then implement a prototype system that would allow further investigation and experiments on synthetic data. If possible, the cross-modal learner would be integrated into the overall system as part of the binding subsystem. Once integrated, it would be possible to perform experiments on real-world objects and with real low level learners.

1.2 Actual work performed

In this section we briefly describe the main achievements related to the topic of this deliverable. For detailed descriptions of the work performed the reader is referred to the papers attached in the annex of this deliverable.

1.2.1 Low-level uni-modal learning with high-level cross-modal supervision.

We have analysed the properties of the oKDE and successfully extended the methodology to the class of online discriminative classifiers, which can also operate in the "open worlds".

- We have revised the theory behind the oKDE and simplified its mathematical formulation. In addition, we extensively analysed the performance of the oKDE. Specifically,(i) we have analysed the influence of the parameters and the data order, (ii) we have analysed the performance of non-stationary distribution estimation and (iii) we have performed the analysis of the discriminative performance of the oKDE.
- We have extended the oKDE theory by considering the problem of online learning as estimation of probability density function for building classifiers. The new approach is called the online discriminative Kernel Density Estimator (odKDE). The extensive analysis of the recently proposed multivariate online Kernel Density Estimators showed that it is possible to build probabilistic models from streaming data and these models perform very well in classification tasks, although they are not built in consideration for discrimination properties. While the oKDE builds a separate class model for each class, the new odKDE takes into account all classes while updating a particular class model. In this way the adaptation of a single class model is weakly supervised by other, uncertain, class models.

- We have derived a distance function that measures distance between the classifiers. This allows us to better compress the generative models such that their discrimination properties do not significantly decrease.
- The odKDE was successfully implemented within the concept of a probabilistic knowledge model, that allows detection of gaps in knowledge and explicitly accounts for the possibility of yet undiscovered classes (the open world question).

The analysis, along with the new methodology described in Annex 2.1 [9] and Annex 2.2 [10], showed that since the odKDE builds models that are complex just so much to retain their discrimination quality, these models tend do be significantly less complex than those built using the oKDE. On the one hand, less redundant (complex) models allow faster learning and inference of the concepts that we are dealing with. On the other hand, these models retain their generative properties, which is also central for knowledge gap detection, which we address in DR.5.3 .

1.2.2 Self-supervised low-level cross-modal learning.

With regard to object affordance learning, the following main developments have been made:

- We have developed a cross-modal competitive learning neural network based on Kohonen's self-organizing map (SOM) and learning vector quantisation (LVQ) paradigms. Sensory modalities are represented by layers of codebook vectors that are trained in either an unsupervised manner (SOM-based) or a self-supervised manner (LVQ-based) depending on the modality in question. The modality codebook vector layers are connected to each other via a Hebbian mapping which models the co-occurrence of data between modalities and serves as a memory of how well nodes in one modality correspond to nodes in another modality over time. The layer doing the unsupervised learning, typically representing the effect modality in our experiments, finds modes in the data distribution that represent potential categories. Meanwhile, the layer undergoing self-supervised learning, typically representing the object property modality in our experiments, compares the mode matching the sample in the effect layer to the corresponding Hebbian mapping in order to determine if the sample is being classified as expected. This training process proceeds without any class labels, self-generated or otherwise. Class labels are generated via clustering the effect layer nodes after training, or when required.
- Feature selection may be regarded as almost being a requirement if a classifier is to be trained successfully using a small number of training

samples; a typical scenario in robotic learning. Therefore, to augment the cross-modal learner, we have developed a feature relevance determination method for LVQ-based algorithms that uses the structure of the codebook vectors, as well as statistics gathered during training to apply Fisher's linear discriminant to individual feature dimensions in an online manner. This allows the system to determine during training that, for example, the object curvature feature is the most important feature in determining whether the object will roll or not.

- The above methods have been evaluated on a real-world dataset of a robotic arm performing pushing actions on eight household objects on a table surface. Four of the objects rolled across the table when pushed and the other four did not roll when pushed, thus forming two natural affordance classes of rolling versus non-rolling objects. Our self-supervised learning method performed favourably when trained on this data, even under the tough condition of leave-one-object-out cross-validation. It also produced performances close to those of fully-supervised LVQ methods.

This work is summarised in Annex 2.3 [17]. It mainly focuses on self-supervised learning of affordance classes. The main goal is to build a system that is able to assign an object into one of several previously autonomously learned affordance classes based only on its observation (2D and 3D) and by knowing the action that is to be performed. A pushing action and several household objects have been used in this experiment.

The second line of research on affordance learning focuses on a different aspect of this problem. Here, the main emphasis is on learning the dynamics of actions that are being performed and the effects of these actions. The main goal is to learn to predict what will happen with an object that has been pushed in terms of regression (what the resulting trajectory of its movement will be). We have continued our work on this problem that we have initiated (and reported) last year. To begin with, we improved the neural network treatment of features. This allowed the machines to achieve convergence when solving the regression problems they are tackling. Based on improved offline experiments, we used the same features in an active learning setting. The Intelligent Adaptive Curiosity algorithm was implemented and adapted for sequence prediction. In summary, the partition of the sensorimotor space into regions was added. We implemented a variance measure to split these regions, taking into account the temporal sensorimotor space or manifold formed after a whole sequence of object poses is obtained.

The results can be seen in the Annex 2.4 [20]. The experiments show that the active learning procedure forms regions that are statistically properly split. The selection of samples from a specific sensorimotor region is based on the learning progress measure, that allows the selection of actions that are more likely to improve the learning performance.

1.2.3 High-level cross-modal learning.

The high-level cross-modal learning problem is closely related to the association rule learning problem in data mining, which was first defined by Agrawal et al., therefore, we based our learning problem definition on Agrawal's definition. First we introduced the notion of *modality* and restrict the association rule to cross-modal associations. Then we defined *percepts* as subsets of features from one modality and *unions* as collections of percepts from different modalities. Percepts are uni-modal representations of perceived instances, while unions are multi-modal, shared representations of the same. We also defined the *binding function* as a function that maps a set of current percepts (percept configuration) into a set of unions (union configuration) and the *measure of binding confidence*, which expresses the confidence in a union configuration based on the accumulated cross-modal knowledge. Finally, we defined binding as finding the union configuration with the maximal confidence, given the current percept configuration, the set of possible binding functions and the current cross-modal knowledge.

We implemented a prototype of the high-level cross-modal learner and a prototype of the binding in *Markov logic networks (MLNs).* The MLN combines first-order logic and probabilistic graphical models in a single representation. This enables a smooth translation of the formal definitions of binding and cross-modal learning into MLN on the one hand, while retaining the ability of probabilistic knowledge representation and probabilistic inference based on graphical models on the other hand. In addition, the representation of cross-modal learning a set of weighted semi-grounded rules is also more suitable for online learning (we limited the learning to the weight learning, while we based the rule structure on a subset of possible association rules from the definition). Experiments have shown that with the growing number of samples the binding rate tends to grow and converge, though with some oscillations. This work is reported in Annex 2.5 [29].

1.3 Relation to the state-of-the-art

In this section we discuss how our work is related to, and goes beyond the current state-of-the-art.

1.3.1 Low-level uni-modal learning with high-level cross-modal supervision.

The analysis of the oKDE in [9] and the methodology therein were our basis for deriving online generative/discriminative models. The revised methodology and the extensive benchmark experiments have shown that in order to allow online adaptation, a model needs to hold sufficient information for it to adapt to future data (sufficient complexity), while at the same time it has to generalise the past observations well for good prediction (prediction capacity). This means that we require general methods that possibly allocate new components as new observations arrive as well as revise the components estimated in the previous time-steps. A model with a small number of components will generalise past observations, but might also underestimate the distribution by modeling it too smoothly (over-smoothing). Then as new data-points arrive, the model may lack the information required to increase its complexity. On the other hand, a drawback of keeping a very large number of components in the model is that the model might overfit the data which results in poor generalisation (under-smoothing). By definition, the oKDE adds new data to increase the generative properties and compresses the model, by trying not to lose those. Therefore, the only place where the oKDE might lose information (over-smooth) is the compression step. By proposing a new cost function, that measures the loss of discrimination in the compression, we derived an online discriminative Kernel Density Estimator (odKDE) [11, 10], that combines generative properties required for proper updates with the discriminative properties to simplify models without compromising the resulting classification performance. While there are many batch approaches to building classifiers, e.g., [15, 4, 5, 1], the odKDE allows online adaptation from as little as single data-point at a time, and since it is based on the oKDE, it produces models with generative properties, which are important when considering discovering gaps in knowledge. The approach has been compared to batch methods [15, 4, 5, 1] and an online method [9]. Results (see Annex 2.2 [10]) demonstrate that the proposed odKDE produces comparable classification performance to the state-of-the-art, and produces models of significantly lower complexity while allowing online adaptation.

1.3.2 Self-supervised low-level cross-modal learning.

With regard to object affordance learning, perhaps the most closely related work in the literature to our work is by Fitzpatrick *et al.* [3, 13]. The authors trained a humanoid robot to recognise "rolling" affordances of 4 household objects using a fixed set of actions to poke the objects in different directions as well as simple visual descriptors for object recognition. There are two main differences between their method and ours. Firstly, in [3, 13], the feature associated with the rolling direction affordance was predetermined, whereas in our system, the learning algorithm is provided with a number of different output features and it must determine for itself the affordance classes within that feature space. Secondly, their system used object recognition to identify the affordances of individual objects, whereas our system determines the affordance class of objects (grounded in output modality features) based, not on their individual identity, but on a broad set of input features (e.g. shape). In [14], the authors used a humanoid robot to push objects on a table and used a Bayesian network to form associations between actions, objects and effects. Though quite similar to our approach, their learning method may not be as amenable to full online learning, as they have to gather a certain amount of data initially to form categories within the various modalities before the network can be trained.

The selection of samples is a crucial problem in machine learning and specially in robotics [12]. Robots should be able to identify situations that are new or informative. Active Learning has been studied specially for classification tasks [23] but there is little work in that respect for generative models. Moreover, the state spaces considering in a robotic scenario are very high dimensional, considering also the temporal aspect of object behaviour prediction. This aspect even concerns scenarios where features are obtained from a simulator, in which case we are reducing to a big extent the dimensionality of the task, but the complexity of the state space of possible behaviours remains a challenge.

1.3.3 High-level cross-modal learning.

Many of the past attempts of binding information within cognitive systems were restricted to associating linguistic information to lower level perceptual information. Roy et al. tried to ground the linguistic descriptions of objects and actions in visual and sound perceptions and to generate descriptions of previously unseen scenes based on the accumulated knowledge [21, 22]. This is essentially a *symbol grounding problem* as first defined by Harnad [6]. Chella et al. proposed a three-layered cognitive architecture around the visual system with the middle, *conceptual layer* bridging the gap between linguistic and sub-symbolic (visual) layers [2]. Related problems were also often addressed by Steels [27].

Jacobsson et al. approached the binding problem in a more general way [8, 7] developing a cross-modal binding system that could associate between multiple modalities and could be part of a wider cognitive architecture. The cross-modal knowledge was represented as a set of binary functions comparing binding attributes in pair-wise fashion. A cognitive architecture using this system for linguistic reference resolution was presented in [28]. This system was capable of learning visual concepts in interaction with a human tutor. Recently, we have developed a probabilistic binding system that encodes the cross-modal knowledge into a Bayesian graphical model [30].

The need for a more flexible, but still probabilistic representation of the cross-modal knowledge motivated our work and led our research efforts in the direction of Markov graphical models. Using Markov logic networks as a template, the cognitive system can construct a specialised graphical model for each specific situation online. Since the cross-modal knowledge is learnt online on the MLN (template) level, the situated graphical models offer more flexibility than the static graphical models (e. g. Bayesian).

2 Annexes

2.1 Kristan and Leonardis "Multivariate Online Kernel Density Estimation with Gaussian Kernels"

Bibliography M. Kristan and A. Leonardis: "Multivariate Online Kernel Density Estimation with Gaussian Kernels" Submitted for journal publication, 2010

Abstract We propose a novel approach for online estimation of probability density functions. Our approach is based on the kernel density estimation (KDE) and produces models that enable online adaptation, which at the same time maintain a low (or bounded) complexity that scales sublinearly with the observed samples. During online adaptation, we maintain a non-parametric model of the data itself and use this model to calculate the corresponding KDE. The dominant parameter in the KDE is the kernel bandwidth and we propose an automatic bandwidth selection rule, which can be computed directly from the non-parametric model of the data. Low complexity of the model is maintained through a novel compression/revitalization scheme. We analyze the properties of the proposed online KDE and compare it to the state-of-the-art approaches on examples of estimating stationary and non-stationary distributions, and on examples of classification. The results show that the online KDE outperforms the state-of-the-art online Gaussian mixture model, and achieves a comparable performance to the batch approaches, while producing models with a significantly lower complexity and allowing online updating by using only a single observation at a time.

Relation to WP This paper extends the theory of the multivariate online Kernel Density Estimation (oKDE) and simplifying the mathematical formulation behind the approach. In addition, we extensively analyse the performance of the oKDE. Specifically, we (i) analyse the influence the parameters and the data order, (ii) we analyse performance of non-stationary distribution estimation and (iii) we perform the analysis of the discriminative performance of the oKDE. This analysis is important for proper use of the oKDE within the WP5 and to determine limitations and advantages of the approach.

2.2 Kristan and Leonardis "Online Discriminative Kernel Density Estimation"

Bibliography M. Kristan and A. Leonardis: "Online Discriminative Kernel Density Estimation" Accepted at International Conference on Pattern Recognition ICPR 2010, Istambul, Turkey, 23-26 August 2010.

Abstract We propose a new method for online estimation of probabilistic discriminative models. The method is based on the recently proposed online Kernel Density Estimation (oKDE) framework which produces Gaussian mixture models and allows adaptation using only a single data point at a time. The oKDE builds reconstructive models from the data, and we extend it to take into account the interclass discrimination through a new distance function between the classifiers. We arrive at an online discriminative Kernel Density Estimator (odKDE). We compare the odKDE to oKDE, batch state-of-the-art KDEs and support vector machine (SVM) on a standard database. The odKDE achieves comparable classification performance to that of best batch KDEs and SVM, while allowing online adaptation, and produces models of lower complexity than the oKDE.

Relation to WP The paper casts the problem of online learning into estimation of probability density function for building classifiers. The extensive analysis of the recently proposed multivariate online Kernel Density Estimators showed that it is possible to build probabilistic models from streaming data and these models perform very well in classification tasks, although they are not build with considering discrimination properties. We have shown in this paper that the oKDE can be extend by explicitly accounting for its discrimination properties, thus leading to models that perform similarly to oKDE, but are significantly less complex. This is achieved by directly building a classifier (not each class model separately from the others), therefore in adaptation of a single class model is weakly supervised by other, uncertain, class models. The paper thus provides the necessary methodology for building probabilistic models for classification in the WP5. The new methodology is called the online discriminative Kernel Density Estimation (odKDE), and since it is derived from the oKDE, it not only allows building classifiers from positive examples but also from negative examples, which is central for the learning tasks in WP5.

2.3 Ridge et al. "Self-supervised Cross-Modal Online Learning of Object Affordances for Developmental Robotic Systems"

Bibliography B. Ridge and D. Skočaj and A. Leonardis: "Self-supervised Cross-Modal Online Learning of Object Affordances for Developmental Robotic Systems" Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Anchorage, USA, May 2010

Abstract For a developmental robotic system to function successfully in the real world, it is important that it be able to form its own internal representations of affordance classes based on observable regularities in sensory data. Usually successful classifiers are built using labeled training data, but it is not always realistic to assume that labels are available in a developmental robotics setting. There does, however, exist an advantage in this setting that can help circumvent the absence of labels: co-occurrence of correlated data across separate sensory modalities over time. The main contribution of this paper is an online classifier training algorithm based on Kohonens learning vector quantization (LVQ) that, by taking advantage of this co- occurrence information, does not require labels during training, either dynamically generated or otherwise. We evaluate the algorithm in experiments involving a robotic arm that interacts with various household objects on a table surface where camera systems extract features for two separate visual modalities. It is shown to improve its ability to classify the affordances of novel objects over time, coming close to the performance of equivalent fully-supervised algorithms.

Relation to WP This paper presents a self-supervised cross-modal competitive learning neural network that can be trained online to form a classifier where the categories are automatically generated by the algorithm based on the statistical properties of the data. Therefore, it addresses the requirements of WP5 on multiple levels, primarily in the areas of continuous learning and the representation of cross-modal concepts. The algorithm can be trained online from scratch, without any prior model being in place, thereby addressing one of the main facets of continuous learning. Moreover, while at the beginning of training the algorithm doesn't have any concept or model of affordance categories, by the end of training it has developed its own affordance categories based on the statistics of the data it has observed during training, thus developing on another idea from continuous learning; the ability to form novel concepts over time. It is inherently cross-modal, in that separate learning layers that initially are trained independently, over time conspire to influence each other during training via cross-modal connectivity.

2.4 Roa and Kruijff. "Offline and active gradient-based learning strategies in a pushing scenario"

Bibliography Roa, S. and Kruijff, G.-J.M.: "Offline and active gradient-based learning strategies in a pushing scenario". In ERLARS, 2010 (To appear).

Abstract When operating in the real world, a robot needs to accurately predict the consequences of its own actions. This is important to guide its own behavior, and in adapting it based on feedback from the environment. The paper focuses on a specific problem in this context, namely predicting affordances of simple geometrical objects called polyflaps. A machine learning approach is presented for acquiring models of object movement, resulting from a robot performing pushing actions on a polyflap. Long Short-Term Memory machines (LSTMs) are used to deal with the inherent spatiotemporal nature of this problem. An LSTM is a gradient-based model of a Recurrent Neural Network, and can successively predict a sequence of feature vectors. The paper discusses offline experiments to test the ability of LSTMs to solve the prediction problem considered here. Cross-validation methods are applied as a measure of convergence performance. An active learning method based on Intelligent Adaptive Curiosity is also applied for improving the learning performance of learners trained offline, generating a combination of learners specialized in different sensorimotor spaces after the knowledge transfer.

Relation to WP This work is related to Tasks 5.2 and 5.3. Task 5.2 is related to cross-modal learning of visiomotor concepts. Task 5.3 is related to active learning. We design a feedback loop between sensing and acting for the robot to select actions on the basis of an intrinsic motivation. This work is also related to the WP 5, specifically task 2.10.

2.5 Vrečko et al. "Binding and Cross-modal Learning in Markov Logic Networks"

Bibliography A. Vrečko, D. Skočaj, A. Leonardis: "Binding and Cross-modal Learning in Markov Logic Networks". Submitted, 2010

Abstract Binding the ability to combine two or more modal representations of the same entity into a single shared representation is vital for every cognitive system operating in a complex environment. In order to successfully adapt to changes in an dynamic environment the binding mechanism has to be supplemented with cross-modal learning. In this paper we define the problems of high-level binding and cross-modal learning. By these definitions we model a binding mechanism and a cross-modal learner in Markov logic network and test the system on a synthetic object database.

Relation to WP The paper addresses the problems of high-level cross-modal binding and learning, as defined in Task 5.2. It defines both problems and shows, using the Markov logic networks, how can high-level cross-modal associations be learned in this framework.

2.6 Skočaj et al. "Cross-modal learning"

Bibliography D. Skočaj, G.-J. Kruijff, A. Leonardis: "Cross-modal learning". Submitted to Encyclopedia of the Sciences of Learning, 2010

Abstract Cross-modal learning refers to any kind of learning that involves information obtained from more than one modality. In the literature the term modality typically refers to a sensory modality, also known as stimulus modality. A stimulus modality provides information obtained from a particular sensorial input, for example visual, auditory, olfactory, or kinesthetic information. Examples from artificial cognitive systems ("robots") include also information about detected range (by sonar or laser rangefinders), movement (by odometry sensors), or motor state (by proprioceptive sensors). We adopt here the notion of modality that includes both the sensorial data, and further interpretations of that data within the modality. For example, from a pair of (depth-calibrated) images, a cloud of points in 3-dimensional space can be computed. We obtain both types of data (the image data, and the 3D points) from the same visual sensor. At the same time, they differ in what information they provide. We consider information sources derived from sensorial data as derived modalities that by themselves can be involved again in cross-modal learning.

Relation to WP In this contribution to the Encyclopedia of the Sciences of Learning we define and discuss the term "cross-modal learning", which is the central research topic of WP5 and Task 5.2.

References

[1] C. C. Chang and C. J. Lin. *LIBSVM: A library for support vector machines*, 2001.

[2] A. Chella, M. Frixione, and S. Gaglio. A cognitive architecture for artificial vision. *Artif. Intell.*, 89(1-2):73–111, 1997.

[3] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini. Learning about objects through action-initial steps towards artificial cognition. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, volume 3, 2003.

[4] M. Girolami and C. He. Probability density estimation from optimally condensed data samples. *ieeepami*, 25(10):1253–1264, 2003.

[5] P. Hall, S. J. Sheater, M. C. Jones, and J. S. Marron. On optimal data-based bandwidth selection in kernel density estimation. *Biometrika*, 78(2):263–269, 1991.

[6] S. Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42:335–346, 1990.

[7] H. Jacobsson, N. Hawes, G-J. Kruijff, and J. Wyatt. Crossmodal content binding in information-processing architectures. In *Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Amsterdam, The Netherlands, March 12–15 2008.

[8] H. Jacobsson, N. Hawes, D. Skočaj, and G-J. Kruijff. Interactive learning and cross-modal binding - a combined approach. In *Symposium on Language and Robots*, Aveiro, Portugal, 2007.

[9] M. Kristan and A. Leonardis. Multivariate Online Kernel Density Estimation with Gaussian Kernels. *submitted*, 2010.

[10] M. Kristan and A. Leonardis. Online discriminative kernel density estimation. In *International Conference on Pattern Recognition*, 2010.

[11] M. Kristan, D. Skočaj, and A. Leonardis. Towards probabilistic online discriminative models. In *Eighteenth International Electrotechnical and Computer Science Conference*, 2009.

[12] R. Martinez-Cantin, J. Peters, and A. Krause, editors. *Towards Closing the Loop: Active Learning for Robotics, Robotics: Science and Systems 2010 Workshop*, Zaragoza, Spain, June 2010.

[13] G. Metta and P. Fitzpatrick. Early integration of vision and manipulation. *Adaptive Behavior*, 11(2):109–128, 2003.

[14] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory-motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1):15–26, 2008.

[15] J. M. L. Murillo and A. A. Rodriguez. Algorithms for gaussian bandwidth selection in kernel density estimators. In *NIPS*, 2008.

[16] P-Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(1), 2007.

[17] B. Ridge, D. Skočaj, and A. Leonardis. Self-supervised cross-modal online learning of basic object affordances for developmental robotic systems. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, USA, May 2010.

[18] S. Roa, G.-J. Kruijff, and H. Jacobsson. Curiosity-driven acquisition of sensorimotor concepts using memory-based active learning. In *Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics*, pages 665–670, 2008.

[19] S. Roa and G.-J.M. Kruijff. Long short-term memory for affordances learning. In Lola Canamero, Pierre-Yves Oudeyer, and Christian Balkenius, editors, *Proceedings of the 9th International Conference on Epigenetic Robotics. Epigenetic Robotics (EpiRob-09), in Conjunction with Modeling Cognitive Development in Robotic Systems, November 12-14, Venice, Italy*, number 146 in Lund University Cognitive Studies, pages 235–236, 11 2009.

[20] S. Roa and G.-J.M. Kruijff. Offline and active gradient-based learning strategies in a pushing scenario. In *International Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems 2010. ERLARS 2010 (To appear)*, 2010.

[21] D. Roy. Learning visually-grounded words and syntax for a scene description task. *Computer Speech and Language*, 16(3-4):353–385, 2002.

[22] D. Roy. Grounding words in perception and action: computational insights. *TRENDS in Cognitive Sciences*, 9(8):389–396, 2005.

[23] B. Settles. Active learning literature survey. Technical Report 1648, University of Wisconsin-Madison, 2009.

[24] D. Skočaj, M. Janiček, M. Kristan, G.-J. M. Kruijff, A. Leonardis, P. Lison, A. Vrečko, and M. Zillich. A basic cognitive system for interactive continuous learning of visual concepts. In *Proceedings of the ICRA*

2010 Workshop on Interactive Communication for Autonomous Intelligent Robots (ICAIR) Making robots articulate what they understand, intend, and do., Anchorage, AK, USA, May 2010.

[25] D. Skočaj, M. Kristan, A. Leonardis, M. Mahnič, A. Vrečko, M. Janíček, G.-J. M. Kruijff, P. Lison, M. Zillich, C. Gretton, M. Hanheide, and M. Göbelbecker. A system approach to interactive learning of visual concepts. 2010. Submitted.

[26] D. Skočaj, G.J. Kruijff, and A. Leonardis. *Encyclopedia of the Sciences of Learning*, chapter Cross-modal learning. Springer, 2010. Submitted.

[27] L. Steels. *The Talking Heads Experiment. Volume 1. Words and Meanings.* Laboratorium, Antwerpen, 1999.

[28] A. Vrečko, D. Skočaj, N. Hawes, and A. Leonardis. A computer vision integration model for a multi-modal cognitive system. In *The 2009 IEEE/RSJ International Conference on Intelligent RObots and Systems*, pages 3140–3147, St. Louis, MO, USA, October 2009.

[29] A. Vrečko, D. Skočaj, and A. Leonardis. Binding and cross-modal learning in markov logic networks. July 2010. Submitted.

[30] J. L. Wyatt, A. Aydemir, M. Brenner, M. Hanheide, N. Hawes, P. Jensfelt, M. Kristan, G.-J. M. Kruijff, P. Lison, A. Pronobis, K. Sjöö, D. Skočaj, A. Vrečko, H. Zender, and M. Zillich. Self-understanding & self-extension: A systems and representational approach. *IEEE Transactions on autonomous mental development*, 2010. Accepted for publication.

Multivariate Online Kernel Density Estimation with Gaussian Kernels

Matej Kristan, *Member, IEEE* and Aleš Leonardis, *Member, IEEE*

***Abstract*—We propose a novel approach for online estimation of probability density functions. Our approach is based on the kernel density estimation (KDE) and produces models that enable online adaptation, which at the same time maintain a low (or bounded) complexity that scales sublinearly with the observed samples. During online adaptation, we maintain a non-parametric model of the data itself and use this model to calculate the corresponding KDE. The dominant parameter in the KDE is the kernel bandwidth and we propose an automatic bandwidth selection rule, which can be computed directly from the non-parametric model of the data. Low complexity of the model is maintained through a novel compression/revitalization scheme. We analyze the properties of the proposed online KDE and compare it to the state-of-the-art approaches on examples of estimating stationary and non-stationary distributions, and on examples of classification. The results show that the online KDE outperforms the state-of-the-art online Gaussian mixture model, and achieves a comparable performance to the batch approaches, while producing models with a significantly lower complexity and allowing online updating by using only a single observation at a time.**

***Index Terms*—Online models, probability density estimation, Kernel density estimation, Gaussian mixture models.**

I. Introduction

MANY tasks in machine learning and pattern recognition require building models from observing sequences of data. In some cases all the data may be available in advance, but processing all data at once (i.e., batch processing) becomes computationally infeasible for large data-sets. This was, for example, a major motivation for extending the batch algorithms such as the principal component analysis (PCA) to its incremental variants, e.g., [6], [25], [41]. Incrementlization allowed building models from large data-sets without being bounded by the computer's memory limitations. Furthermore, in many real-world scenarios all the data may not available in advance, or we even want to observe some process for an indefinite duration, while continually providing the best estimate of the model from the data observed so far. These scenarios are, for example, central for the problems in developmental and cognitive robotics [2], [35], [36], [43], in which a cognitive agent learns gradually from observing and interacting with its tutor or environment (e.g. [24]). A common point in applications of learning from large data-sets and applications of learning from sequences of data is that they all require a type of online construction of the models.

A popular approach to generating models from data is to model the probability density function (pdf) associated with the observed data. Traditionally, parametric models based on Gaussian mixture models (GMM) [27], [28] have been applied with some success in estimation of the pdf when all data are observed in advance. The parametric mixture models typically require specifying the number of components (or an upper bound on the number) in advance [28], [50] or implementing some data-driven criteria for selection of the appropriate number of components (e.g. [12], [48]). Improper choice of the number of components may lead to models which fail to capture the complete structure of the underlying pdf. Non-parametric methods such as Parzen kernel density estimators (KDE) [15], [31], [49] alleviate this problem by treating each observation as a component in the mixture model and assuming all components have equal bandwidths. In case of using Gaussian kernels, for example, the bandwidth is the kernel's covariance. The problem of KDE is then how to automatically set the kernel bandwidth. Indeed, efficient bandwidth estimation is a difficult problem which has been studied extensively in the literature (e.g., [10], [39], [40], [44], [47]). Usually, the approaches for the bandwidth estimation focus on one-dimensional or low-dimensional problems (see, e.g. [8], [11], [21], [44], [47]) and approximations in the case of multivariate application (see, e.g., [8], [17], [38], [39]). Recently Murillo and Rodriguez [30] have proposed an efficient cross-validation-based method which iteratively calculates the multivariate bandwidth. One drawback of the KDEs is that their complexity (number of components) increases linearly with the number of the observed data. To remedy this increase, methods have been proposed to reduce the number of components (compress) either to a predefined value [15], [49], or to optimize some data-driven criteria. Examples of the latter approach are reduced-set-density estimators [13] which minimize an approximate L_2 distance between the unknown density and the model, and MDL-based radial-basis-function (RBF) networks [26] which gradually remove the RBF units by optimizing a minimum-description-length criterion [37]. Alternatively, Pawlak and Stadmüller [32] have investigated estimation of one-dimensional KDEs from a pre-binned data (histograms) with a preset binning to reduce the model's complexity.

In an online scenario, we are faced with several difficulties when trying to estimate a compressed mixture model. The main difficulty is that the model has to maintain a sufficient information to adapt to future data (sufficient complexity),

M. Kristan is with the Faculty of Computer and Information Science, University of Ljubljana and with the Faculty of Electrical Engineering, University of Ljubljana

A. Leonardis is with the Faculty of Computer and Information Science, University of Ljubljana

while at the same time it has to generalize well the past observations for a good prediction (prediction capacity). This means that we require generally methods that possibly allocate new components as new observations arrive as well as revise the components estimated in the previous time-steps. A model with a small number of components will generalize the past observations, but might also underestimate the distribution as too smooth (over-smoothing). Then as new data-points arrive, the model may lack the information required to increase its complexity. On the other hand, a drawback of keeping a very large number of components in the model is that the model might overfit the data which results in poor generalization (under-smoothing).

There have been several attempts to address the problems of online estimation in the context of merging the non-parametric quality of the kernel density estimators with the Gaussian mixture models in online applications. Typically, authors constrain their models by imposing various assumptions about the distributions to be estimated, leading to various algorithms for online adaptation. Arandjelović et.al. [1] proposed a scheme for online adaptation of the Gaussian mixture model which can be updated by observing as little as a single data-point at a time. However, a strong restriction is made that data is temporally coherent in feature space, which prevents its use in general applications. In particular, the smoothness of their model depends heavily on the distance between successive data points. If the distance is smaller than assumed, the model will likely be over-smoothed. On the other hand, if the distance is greater than assumed, the model will be under-smoothed resulting in a poor prediction capacity. Priebe and Marchette [34] proposed an online EM algorithm, called active mixtures, which allows adaptation from a single observation at a time, assumes the data is radnomly sampled from the underlying distribution, and includes a heuristic for allocating new components, which makes it less sensitive to data ordering. Song et. al. [42] aimed to alleviate the restrictions on data orderings by processing data in large blocks. Bischof and Leonardis [5] use an RBF-based network and apply an MDL-based procedure for basis function allocation and deletion – they assume a predefined initial RBF size. Deleclerq and Piater [9] assume each data-point is a Gaussian with a predefined covariance. All data are stored in the model and a heuristic is used to determine when a subset of the data (Gaussians) can be replaced by a single component. Han et. al. [18] proposed an online approach inspired by the kernel density estimation in which each new observation is added to the model as a Gaussian kernel with a predefined bandwidth. The model's complexity is maintained through the assumption, that the underlying probabilty density function can be approximated sufficiently well by retaining only its modes. They therefore implement compression by searching for the modes of the distribution and approximating them by Gaussians. This approach deteriorates in situations when the assumed predefined bandwidths of kernels are too restrictive, and when the distribution is locally non-Gaussian. Such examples are a skewed distribution, a heavy tailed distribution and the uniform distribution. In general, a major problem with those methods which assume a predefined (known) kernel bandwidth, or intial RBF, is that they likely lead to over/under-smoothing problems when the bandwidth is not estimated (or guessed) properly. Recently, we [24] have proposed an incremental kernel density estimator, which uses the least assumptions in comparison to the related methods. The only assumption is that the target distribution is sufficiently smooth, which is a common assumption in the kernel density estimation framework. The method implements a variant of a recursive kernel density estimator (see, e.g., [10]) for bandwidht estimation and implements an optimization to compress the distribution. A drawback of that method is that it requires computationally intensive iterative optimization, it is applicable only to one-dimensional data, and can suffer from poor initialization in cases of a small number of samples. Generally, a positive side of imposing assumptions on the distribution, which we want to estimate, is that we can better constrain the problem of estimation and design efficient algorithms for the task at hand. A downside is that once the assumptions are violated, the algorithms will likely break down and deteriorate in performance. In this paper we therefore aim at an algorithm, which would be applicable to multivariate cases, would be minimally constrained by the assumptions and therefore efficiently tackle the difficulties of online estimation.

A. *Our approach*

We propose a new multivariate online version of the kernel density estimator, which enables adaptation from only a single observation at a time. Our approach is grounded in the following two key ideas. The first key idea is that, unlike the related approaches, we do not attempt to build a model of the target distribution directly, but rather maintain a non-parametric model of the data itself in a form of a *sample distribution* – this model can then be used to calculate the kernel density estimate of the target distribution. The sample distribution is constructed by online clustering of the data-points. The second key idea is that we treat each new observation as a distribution in a form of a Dirac-delta function and we model the *sample distribution* by a mixture of Gaussian and Dirac-delta functions. During online operation the sample distribution is updated by each new observation in essentially the following three steps (Figure 1a): (1) In the first step, we update the sample model with the observed data-point. (2) In the second step, the updated model is used to recalculate the optimal bandwidth for the KDE – here, the main issue is how to calculate the bandwidth without having access to the previously observed individual samples. (3) In the third step, the sample distribution is refined and compressed. This step is required because, without compression, the number of components in our model would increase linearly with the observed data. However, it turns out that a valid compression at one point in time might become invalid later, when new data-points arrive. The result of these invalid compressions is that the model misses the structure of the underlying distribution and produces significantly over-smoothed estimates (see, e.g., Figure 10). We therefore require a refinement algorithm to detect such events and to recover from them. The main issues here are: (i) how to devise an optimization which would

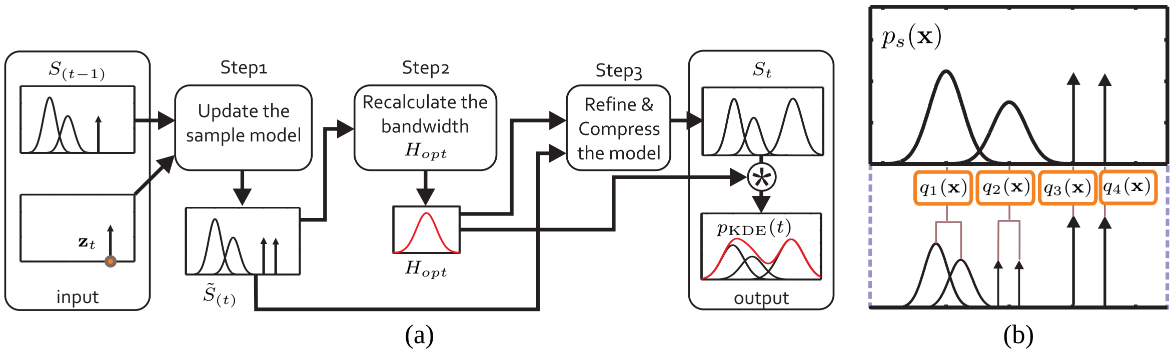
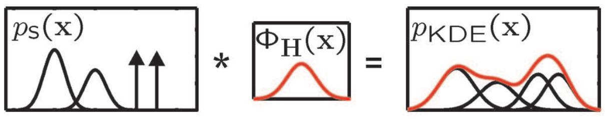


Fig. 1. A three-step summary of the online KDE iteration (a). The sample model $S_{(t-1)}$ is updated by a new observation $\mathbf{z}_t$ and compressed into a new sample model $S_{(t)}$. An illustration of the new sample model $S_{(t)}$ (sample distribution $p_s(\mathbf{x})$ along with its detailed model $\{q_i(\mathbf{x})\}_{i=1:4}$) is shown in (b).

efficiently compress the sample model, (ii) how to determine the extent of the allowed compression and (iii) how to recover from the early compressions. To allow the recovery from the early compression, we keep for each component in the sample distribution another model of the data that generated that component. This detailed model is in a form of a mixture model with at most two components (Figure 1b). The rationale behind constraining the detailed model to two components is that this is the simplest detailed model that allows detection of early over compressions. After the compression and refinement step, the KDE can be calculated as a convolution of the (compressed) sample distribution with the optimal kernel calculated at step 2 (see Figure 2).



$p_S(\mathbf{x})$
*
$\Phi_{\mathrm{H}}(\mathbf{x})$
=
$p_{\mathrm{KDE}}(\mathbf{x})$

Fig. 2. Calculation of the KDE $p_{\mathrm{KDE}}(\mathbf{x})$ through a convolution of the sample distribution $p_s(\mathbf{x})$ with a kernel (Dirac-deltas are depicted by upward arrows).

Our main contribution is the approach for multivariate online kernel density estimation (oKDE), which enables construction of a multivariate probability density estimate by observing only a single sample at a time and which can automatically ballance between its complexity and generalization of the observed data points. In contrast to the standard bandwidth estimators, which require access to all observed data, we derive a method which can use a mixture model (sample distribution) instead and can be applied to multivariate problems. To enable a controlled compression of the sample distribution, we propose a compression scheme which maintains low distance between the KDE before and after compression. To this end, we propose an approximation to the multivariate Hellinger distance on mixtures of Gaussians. Since over-compressions occur during online estimation, we propose a revitalization scheme, which detects over-compressed components and refines them, thus allowing efficient adaptation.

The remainder of the paper is structured as follows. In Section II we define our model. In Section III we derive a rule for automatic bandwidth selection. We propose the compression scheme in Section IV, where we also address the problem of over-compression. The online KDE (oKDE) algorithm is presented in Section V. In Section VI we analyze the recostructive and discriminative properties of the oKDE. We analyze how the parameters, the data ordering and the revitalization scheme affect the oKDE's performance and we compare the oKDE to existing online and batch state-of-the-art algorithms on examples of estimating distributions and on classification examples. We conclude the paper in Section VII.

II. The model definition

In this section we define our model. As stated in the introduction, we aim at maintaining a (compressed) model of the observed data-points in a form of a distribution model, and use this model to calculate the KDE when required. We therefore start with the definition of the distribution of the data-points. Each separate data-point can be presented in a distribution as a single Dirac-delta function, with its probability mass concentrated at that data-point. Noting that a Dirac-delta can be generally written as a Gaussian with zero covariance, we define the model of (potentially compressed) d-dimensional data as an N-component Gaussian mixture model

$$p_s(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\mathbf{\Sigma}_{si}}(\mathbf{x} - \mathbf{x}_i), \tag{1}$$

where

$$\phi_{\mathbf{\Sigma}}(\mathbf{x} - \mu) = (2\pi)^{-\frac{d}{2}} |\mathbf{\Sigma}|^{-\frac{1}{2}} e^{(-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu))} \tag{2}$$

is a Gaussian kernel centered at μ with covariance matrix $\mathbf{\Sigma}$. We call $p_s(\mathbf{x})$ a *sample distribution* and a kernel density estimate (KDE) is defined as a convolution of $p_s(\mathbf{x})$ by a kernel with a covariance matrix (bandwidth) $\mathbf{H}$ (see Figure 2):

$$\hat{p}_{\mathrm{KDE}}(\mathbf{x}) = \phi_{\mathbf{H}}(\mathbf{x}) * p_s(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\mathbf{H}+\mathbf{\Sigma}_{si}}(\mathbf{x} - \mathbf{x}_i). \tag{3}$$

To maintain a low complexity of the KDE during online operation, the sample distribution $p_s(\mathbf{x})$ is compressed from time to time by replacing clusters of components in the $p_s(\mathbf{x})$ by single Gaussian components. Details will be explained later in section IV. As noted in the introduction, compressions at some point in time may later become invalid as new data arrive. To detect and recover from these early over-compressions, we keep an additional model of data for each component in the mixture model. We therefore define our *model of the observed samples* as

$$\mathbf{S}_{\text{model}} = \{p_s(\mathbf{x}), \{q_i(\mathbf{x})\}_{i=1:N}\}, \tag{4}$$

where $p_{\text{s}}(\mathbf{x})$ is the *sample distribution* and $q_i(\mathbf{x})$ is a mixture model (with at most two components) for the i-th component in $p_{\text{s}}(\mathbf{x})$ (Figure 1b). To obtain a KDE, we need to compute the optimal bandwidth from all the observed samples, which are now summarized in the sample model $p_s(\mathbf{x})$ (step 2 in Figure 1a). In the following we propose a method for calculating this bandwidth.

III. Estimation of the bandwidth

If we retained (did not compress) all the observed samples in the sample model, then the sample distribution $p_s(\mathbf{x})$ would contain only components with zero covariances (i.e, $\mathbf{\Sigma}_{\text{s}i} = \mathbf{0}$ for all i) and the KDE (3) would be defined as $\hat{p}_{\text{KDE}}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_i)$. The goal of all KDE methods is to determine the kernel bandwidth $\mathbf{H}$ such that the distance between the $\hat{p}_{\text{KDE}}(\mathbf{x})$ and the unknown pdf $p(\mathbf{x})$, that generated the data, is minimized. A classical measure used to define the closeness of the estimator $\hat{p}_{\text{KDE}}(\mathbf{x})$ to the underlying pdf is the *asymptotic mean integrated squared error* (AMISE), defined as ([47], pp.95-98),

$$AMISE = (4\pi)^{-\frac{d}{2}} |\mathbf{H}|^{-\frac{1}{2}} N_\alpha^{-1} + \frac{1}{4} d^2 \int \text{tr}^2\{\mathbf{H}\mathcal{G}_p(\mathbf{x})\} d\mathbf{x}, \tag{5}$$

where $\text{tr}\{\cdot\}$ is the trace operator, $\mathcal{G}_p(\mathbf{x})$ is a Hessian of $p(\mathbf{x})$, and $N_\alpha = (\sum_{i=1}^{N} \alpha_i^2)^{-1}$. If we rewrite the bandwidth matrix in terms of scale β and structure $\mathbf{F}$, i.e., $\mathbf{H} = \beta^2 \mathbf{F}$, and assume for now that $\mathbf{F}$ is known, then (5) is minimized at scale

$$\beta_{\text{opt}} = [d(4\pi)^{\frac{d}{2}} N_\alpha R(p, \mathbf{F})]^{-\frac{1}{d+4}}, \tag{6}$$

where the term

$$R(p, \mathbf{F}) = \int \text{tr}^2\{\mathbf{F}\mathcal{G}_p(\mathbf{x})\} d\mathbf{x} \tag{7}$$

is a functional of the second-order partial derivatives, $\mathcal{G}_p(\mathbf{x})$, of the unknown distribution $p(\mathbf{x})$. In principle, this functional could be estimated using the plug-in methods [47], however, these are usually numeric, iterative, assume we have access to *all the observed samples* and often suffer from numerical instabilities. In our case, we maintain only a (compressed) mixture model of the samples, and we require an approximation to the functional using this mixture model.

We first note (see, eg., [47]) that $R(p, \mathbf{F})$ can be written in terms of expectations of the derivatives $\psi_{\mathbf{r}} = \int p^{(\mathbf{r})}(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}$. We can then use the sample distribution $p_{\text{s}}(\mathbf{x})$ to obtain the following approximations

$$p(\mathbf{x}) \approx p_{\text{s}}(\mathbf{x}) \ ; \ p^{(\mathbf{r})}(\mathbf{x}) \approx p_{\mathbf{G}}^{(\mathbf{r})}(\mathbf{x}), \tag{8}$$

where we approximate the derivative of $p(\mathbf{x})$, $p_{\mathbf{G}}^{(\mathbf{r})}(\mathbf{x})$, through the following kernel density estimate

$$p_{\mathbf{G}}(\mathbf{x}) = \phi_{\mathbf{G}}(\mathbf{x}) * p_s(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \phi_{\Sigma_{\text{g}j}}(\mathbf{x} - \mu_j). \tag{9}$$

The estimate $p_{\mathbf{G}}(\mathbf{x})$ plays a role of the so-called *pilot distribution* with covariance terms $\mathbf{\Sigma}_{\text{g}j} = \mathbf{G} + \mathbf{\Sigma}_{sj}$ and $\mathbf{G}$ is called the *pilot bandwidth*. Using the approximations in (8) and the derivation in Appendix A, we can approximate $R(p, \mathbf{F})$ by

$$\hat{R}(p, \mathbf{F}, \mathbf{G}) = \int \text{tr}\{\mathbf{F}\mathcal{G}_{p_{\mathbf{G}}}(\mathbf{x})\} \text{tr}\{\mathbf{F}\mathcal{G}_{p_s}(\mathbf{x})\}. \tag{10}$$

Since $p_{\text{s}}(\mathbf{x})$ and $p_{\mathbf{G}}(\mathbf{x})$ are both Gaussian mixture models, we can calculate the functional (10) using only matrix algebra:

$$\hat{R}(p, \mathbf{F}, \mathbf{G}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \phi_{\mathbf{\Sigma}_{\text{g}i} + \mathbf{\Sigma}_{\text{s}j}}(\mu_{\text{g}i} - \mu_{\text{s}j}) \times [2\text{tr}(\mathbf{F}\mathbf{A}_{ij}\mathbf{F}\mathbf{B}_{ij}) + \text{tr}^2(\mathbf{F}\mathbf{C}_{ij})], \tag{11}$$

where for each pair (i, j) we have used the following definitions

$$\begin{aligned} \mathbf{A}_{ij} &= (\mathbf{\Sigma}_{\text{g}i} + \mathbf{\Sigma}_{\text{sj}})^{-1}, \\ \mathbf{B}_{ij} &= \mathbf{A}_{ij}\{\mathbf{I} - 2(\mu_{\text{g}i} - \mu_{\text{sj}})(\mu_{\text{g}i} - \mu_{\text{sj}})^T \mathbf{A}_{ij}\}, \\ \mathbf{C}_{ij} &= \mathbf{A}_{ij}\{\mathbf{I} - (\mu_{\text{g}i} - \mu_{\text{sj}})(\mu_{\text{g}i} - \mu_{\text{sj}})^T \mathbf{A}_{ij}\}. \end{aligned} \tag{12}$$

Derivation of (11,12) is rather laborious, and is based on the M. P. Wand's [46] study of an integral similar to (10). For convenience, we have removed the required derivations into the Appendix B.

Note that we still have to determine the pilot bandwidth $\mathbf{G}$ of $p_{\mathbf{G}}(\mathbf{x})$ and the structure $\mathbf{F}$ of the bandwidth matrix $\mathbf{H}$. We use the empirical covariance of the observed samples $\hat{\mathbf{\Sigma}}_{\text{smp}}$ to approximate both. First we resort to a practical assumption [11], [47] that the *structure* of the bandwidth $\mathbf{H}$ can be reasonably well approximated by the structure of the covariance matrix of the observed samples, i.e., $\mathbf{F} = \hat{\mathbf{\Sigma}}_{\text{smp}}$. We estimate the pilot bandwidth $\mathbf{G}$ by a normal-scale rule [47]. Note that the normal-scale provides a bandwidth that is optimal in AMISE sense if the unknown distribution $p(\mathbf{x})$ is in fact normal. While this assumption is too restrictive to directly estimate $\mathbf{H}$, it is admissable in practice for estimation of the bandwidths for the derivatives (see, eg. [47] page 71). The pilot bandwidth using the multivariate normal-scale rule for the derivative ([47], page 111) is given by

$$\mathbf{G} = \hat{\mathbf{\Sigma}}_{\text{smp}} \left(\frac{4}{(d+2)N_\alpha}\right)^{\frac{2}{d+4}}. \tag{13}$$

IV. Compression of the sample model

Having approximated the optimal bandwidth, the next step is to compress and refine the resulting model (step 3 in Figure 1a). The objective of the compression is to approximate the original N-component sample distribution

$$p_{\text{s}}(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi_{\mathbf{\Sigma}_{si}}(\mathbf{x} - \mu_i) \tag{14}$$

by a M-component, $M < N$, equivalent $\hat{p}_{\text{s}}(\mathbf{x})$

$$\hat{p}_{\text{s}}(\mathbf{x}) = \sum_{j=1}^{M} \hat{w}_j \phi_{\hat{\mathbf{\Sigma}}_{sj}}(\mathbf{x} - \hat{\mu}_j), \tag{15}$$

such that the resulting (compressed) KDE does not change significantly. Since a direct optimization (e.g., [24]) of the parameters in $\hat{p}_s(\mathbf{x})$ can be computationally prohibitive, and prone to slow convergence even for moderate number of dimensions, we resort to a clustering-based approach. The main idea is to identify clusters of components in $p_s(\mathbf{x})$, such that each cluster can be sufficiently well approximated by a single component in $\hat{p}_s(\mathbf{x})$. Let $\Xi(M) = \{\pi_j\}_{j=1:M}$ be a collection of disjoint sets of indexes, which cluster $p_s(\mathbf{x})$ into M sub-mixtures. The sub-mixture corresponding to indexes $i \in \pi_j$ is defined as

$$p_s(\mathbf{x};\pi_j) = \sum_{i\in\pi_j} w_i \phi_{\boldsymbol{\Sigma}_{si}}(\mathbf{x}-\mu_i) \quad (16)$$

and is approximated by the j-th component $\hat{w}_j\phi_{\hat{\boldsymbol{\Sigma}}_{sj}}(\mathbf{x}-\hat{\mu}_j)$ of $\hat{p}_s(\mathbf{x})$. The parameters of the $j-th$ component are defined by matching the first two moments (mean and covariance) [4] of the sub-mixture:

$$\hat{w}_j = \sum\nolimits_{i\in\pi(j)} w_i \ , \ \hat{\mu}_j = \hat{w}_j^{-1} \sum\nolimits_{i\in\pi(j)} w_i\hat{\mu}_i$$
$$\hat{\Sigma}_j = \hat{w}_j \sum\nolimits_{i\in\pi(j)} w_i(\boldsymbol{\Sigma}_i + \mu_i\mu_i^T) - \hat{\mu}_j\hat{\mu}_j^T. \quad (17)$$

For better understanding, we illustrate in Figure 3 an example in which components of a sample distribution $p_s(\mathbf{x})$ are clustered to form another (compressed) sample distribution $\hat{p}_s(\mathbf{x})$ with a smaller number of components. We also show the KDEs corresponding to the original and the compressed KDE. While the number of components in the sample distribution is reduced, the resulting KDE does not change significantly.

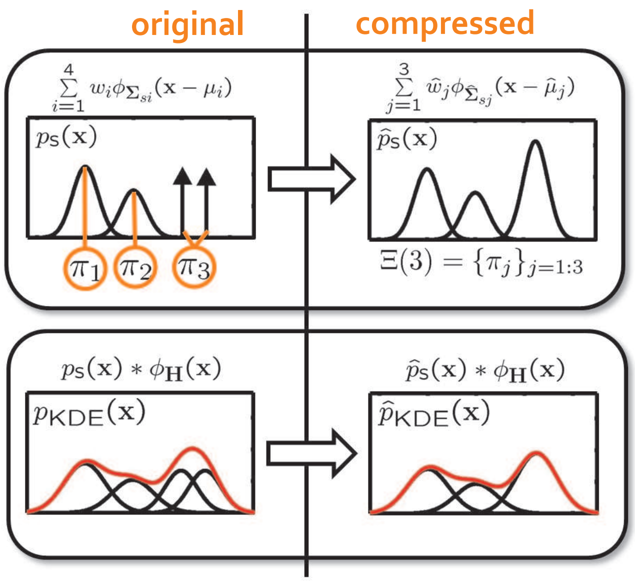

original
compressed
$\sum_{i=1}^{4} w_i\phi_{\Sigma_{si}}(\mathbf{x}-\mu_i)$
$p_s(\mathbf{x})$
π_1 π_2 π_3
$\sum_{j=1}^{3} \hat{w}_j\phi_{\hat{\Sigma}_{sj}}(\mathbf{x}-\hat{\mu}_j)$
$\hat{p}_s(\mathbf{x})$
$\Xi(3) = \{\pi_j\}_{j=1:3}$
$p_s(\mathbf{x}) * \phi_{\mathbf{H}}(\mathbf{x})$
$p_{KDE}(\mathbf{x})$
$\hat{p}_s(\mathbf{x}) * \phi_{\mathbf{H}}(\mathbf{x})$
$\hat{p}_{KDE}(\mathbf{x})$

Fig. 3. The images illustrate a compression of a four-component sample distribution $p_s(\mathbf{x})$ into a three-component counterpart $\hat{p}_s(\mathbf{x})$ using the clustering assignment $\Xi(3) = \{\pi_j\}_{j=1:3}$. The left and right columns show the sample distribution (upper row) and the corresponding KDE (lower row) before and after compression, respectively.

As indicated in Figure 3 the compression seeks to identify the clustering assignment $\Xi(M)$, along with the minimal number of clusters M, such that the error induced by each cluster is sufficiently low, i.e., it does not exceed a prescribed threshold D_{th},

$$\hat{M} = \arg\min_M E(\Xi(M)) \ , \ \text{s.t. } E(\Xi(\hat{M})) \leq D_{\text{th}}, \quad (18)$$

where we define $E(\Xi(M))$ as the largest local clustering error $\hat{E}(p_s(x;\pi_j),\mathbf{H}_{\text{opt}})$ under the clustering assignment $\Xi(M)$,

$$E(\Xi(M)) = \max_{\pi_j\in\Xi(M)} \hat{E}(p_s(x;\pi_j),\mathbf{H}_{\text{opt}}). \quad (19)$$

The local clustering error $\hat{E}(p_s(x;\pi_j),\mathbf{H}_{\text{opt}})$ tells us the error induced under the KDE with bandwidth $\mathbf{H}_{opt}$, if the cluster $p_s(x;\pi_j)$ is approximated by a single Gaussian. We define this error next.

A. The local clustering error

Let $\mathbf{H}_{\text{opt}}$ be the current estimated bandwidth, and let $p_1(x) = p_s(\mathbf{x};\pi_j)$ be a cluster, a sub-mixture of the sample distribution defined by (16), which we want to approximate with a single Gaussian $p_0(x)$ according to (17). We define the local clustering error as the distance

$$\hat{E}(p_1(\mathbf{x}),\mathbf{H}_{\text{opt}}) = D(p_{1KDE}(\mathbf{x}),p_{0KDE}(\mathbf{x})), \quad (20)$$

between the corresponding *KDEs*

$$p_{1\text{KDE}}(\mathbf{x}) = p_1(\mathbf{x}) * \phi_{\mathbf{H}_{\text{opt}}}(\mathbf{x})$$
$$p_{0\text{KDE}}(\mathbf{x}) = p_0(\mathbf{x}) * \phi_{\mathbf{H}_{\text{opt}}}(\mathbf{x}).$$

In particular, we can quantify the distance between distributions using the Hellinger distance [33], which is defined as

$$D^2(p_{1\text{KDE}}(\mathbf{x}),p_{0\text{KDE}}(\mathbf{x})) \triangleq$$
$$\frac{1}{2}\int (p_{1\text{KDE}}(\mathbf{x})^{\frac{1}{2}} - p_{0\text{KDE}}(\mathbf{x})^{\frac{1}{2}})^2 d\mathbf{x}. \quad (21)$$

Note that, while the Hellinger distance is a proper metric between distributions and is bounded to interval $[0,1]$ (see, e.g., [33]), it cannot be calculated analytically for the mixture models. We therefore calculate its approximation using the *unscented transform* [22], which has been recently applied in [14] for approximating the Kullback-Liebler divergence between mixture models. For convenience, the derivation of the unscented approximation of the Hellinger distance is given in the Appendix C.

B. Compression by hierarchical error minimization

In principle, the global optimization of (18) would require evaluation of all possible cluster assignments $\Xi(M)$ for the number of clusters M ranging from one to N, which becomes quickly computationally prohibitive. A significant reduction in complexity of the search can be obtained by a *hierarchical* approach to cluster discovery. Similar approaches have been previously successfully applied for speeding up the EM algorithm [29], online visual category discovery [16] and controlled data compression with Gaussian mixture models to a predefined number of clusters [15], [19].

In our implementation, the hierarchical clustering proceeds as follows. We start by splitting the entire sample distribution $p_s(\mathbf{x})$ into two sub-mixtures using the Goldberger's [15] K-means algorithm for mixture models[1] with $K = 2$. Each sub-mixture is approximated by a single Gaussian and the sub-mixture which yields the largest local error $\hat{E}(p_s(\mathbf{x};\pi_j),H_{\text{opt}})$

[1]Note that to avoid the singularities associated with the components in the sample distribution with zero covariance, the K-means algorithm for the Gaussian mixtures is carried out on the corresponding KDE.

is further splitted into two sub-mixtures. This process is recursively continued until the largest local error is sufficiently small and the condition $E(\Xi(M)) \leq D_{th}$ in (18) fulfilled. This approach generates a binary tree with $\hat{M}$ leafs among the components of the sample distribution $p_{\mathrm{s}}(\mathbf{x})$, in which the leafs of the tree represent the clustering assignments $\Xi(\hat{M}) = \{\pi_j\}_{j=1:M}$. Once the clustering $\Xi(\hat{M})$ is found, the compressed sample distribution $\hat{p}_{\mathrm{s}}(\mathbf{x})$ (15) is calculated using (16) and (17). An illustration of the hierarchical clustering on a one-dimensional example is shown in Figure 4.

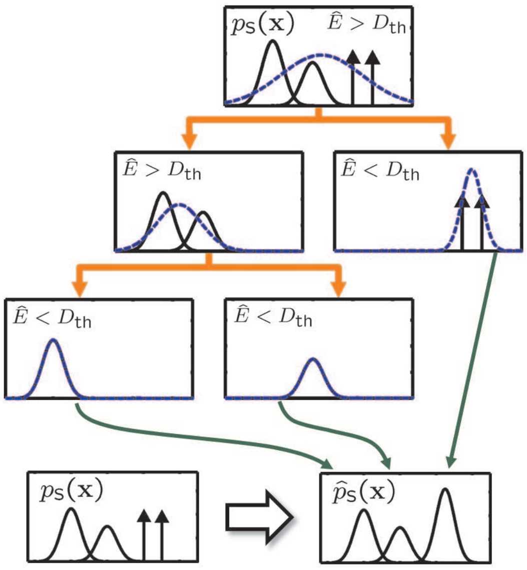

$p_{\mathrm{s}}(\mathbf{x})$ $\hat{E} > D_{\mathrm{th}}$
$\hat{E} > D_{\mathrm{th}}$
$\hat{E} < D_{\mathrm{th}}$
$\hat{E} < D_{\mathrm{th}}$
$\hat{E} < D_{\mathrm{th}}$
$p_{\mathrm{s}}(\mathbf{x})$
$\hat{p}_{\mathrm{s}}(\mathbf{x})$

Fig. 4. Illustration of the hierarchical clustering. The components of the sample distribution $p_{\mathrm{s}}(\mathbf{x})$ are hierachically clustered to form a tree. Each leaf of the tree is replaced by a single Gaussian and together form the compressed sample distribution $\hat{p}_{\mathrm{s}}(\mathbf{x})$.

Recall that we keep track of a detailed model for each component in the sample distribution (see, e.g., Figure 1b). The detailed model $\hat{q}_j(\mathbf{x})$ of the j-th component in the compressed model $\hat{p}_{\mathrm{s}}(\mathbf{x})$ is calculated as follows. If the set π_j contains only a single index, i.e., $\pi_j = \{i\}$, then the j-th component of the compressed sample distribution is equal to the i-th component in the original sample distribution and therefore the detailed model remains unchanged, i.e., $\hat{q}_j(\mathbf{x}) = q_i(\mathbf{x})$. On the other hand, if π_j contains multiple indexes, then the detailed models corresponding to these indexes are first concatenated into a single *extended* mixture model

$$\hat{q}_{j\mathrm{ext}}(x) = \sum_{i \in \pi_j} q_i(\mathbf{x}). \tag{22}$$

Then the required two-component detailed model $\hat{q}_j(\mathbf{x})$ is generated by splitting $\hat{q}_{j\mathrm{ext}}(\mathbf{x})$ into two sub-mixtures again using the Goldberger's K-means and each sub-mixture is approximated by a single Gaussian using (17). Note that the detailed model is constrained to at most two components, since this is the least complex model which enables detection of the early over compressions as discussed next.

C. Revitalizing the sample distribution

The compression identifies and compresses those clusters of components whose compression does not introduce a significant error into the KDE with the bandwith $\mathbf{H}_{\mathrm{opt}}$ estimated at the time of compression. However, during online operation, new samples arrive, the sample distribution and $\mathbf{H}_{\mathrm{opt}}$ change, and so does the estimated KDE. Therefore, a compression which may be valid for a KDE at some point in time, may become invalid later on. This is illustrated in Figure 5.

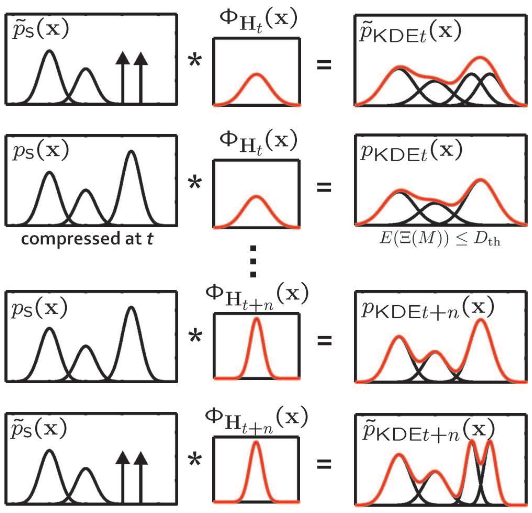

$\tilde{p}_{\mathrm{s}}(\mathbf{x})$
$\Phi_{\mathbf{H}_t}(\mathbf{x})$
$\tilde{p}_{\mathrm{KDE}t}(\mathbf{x})$
$p_{\mathrm{s}}(\mathbf{x})$
$\Phi_{\mathbf{H}_t}(\mathbf{x})$
$p_{\mathrm{KDE}t}(\mathbf{x})$
compressed at t
$E(\Xi(M)) \leq D_{\mathrm{th}}$
$p_{\mathrm{s}}(\mathbf{x})$
$\Phi_{\mathbf{H}_{t+n}}(\mathbf{x})$
$p_{\mathrm{KDE}t+n}(\mathbf{x})$
$\tilde{p}_{\mathrm{s}}(\mathbf{x})$
$\Phi_{\mathbf{H}_{t+n}}(\mathbf{x})$
$\tilde{p}_{\mathrm{KDE}t+n}(\mathbf{x})$

Fig. 5. Illustration of the early over compression. The left, middle and the right columns show the sample distribution, the bandwidth and the resulting KDEs, respectively. The first and the second row of left column show the sample distribution and its compressed counterpart, respectively. Note that the compressed counterpart is valid, since the KDE does not change significantly (right column). The third row illustrates a time-step $t + n$, in which the bandwidth changes and so does the KDE (right column). The last row shows the KDE calculated from the non compressed sample distribution using the bandwidth $\mathbf{H}_{t+n}$. This KDE is significantly different from its compressed version, which makes the early compression at time-step t invalid under the bandwidth at time-step $t + n$.

The over compression can be detected through inspection of the *detailed model* of each component in the sample distribution $p_{\mathrm{s}}(\mathbf{x})$. The local clustering error $\hat{E}(q_i(\mathbf{x}), \mathbf{H}_{\mathrm{opt}})$ (19) of each component $w_i \phi_{\mathbf{\Sigma}_{si}}(\mathbf{x})$ in the sample distribution can be evaluated against its detailed model $q_i(\mathbf{x})$ to verify whether the global clustering error from (18) exceeds the threshold D_{th}. Those components in $p_{\mathrm{s}}(\mathbf{x})$ for which $\hat{E}(q_i(\mathbf{x}), \mathbf{H}_{\mathrm{opt}}) > D_{th}$ are removed from the sample distribution and replaced by the two components of their detailed model. A detailed model is then created for each of the new components. For example, let $w_i \phi_{\Sigma_i}(\mathbf{x} - \mu_i)$ be one of the new components. If the determinant of $\mathbf{\Sigma}_i$ is zero, then this component corresponds to a single data-point and therefore its detailed model is just the component itself. However, in case the determinant is nonzero, it means that the component has been generated through clustering of several detailed models in the previous compression steps. Its detailed model is then initialized by splitting $\phi_{\mathbf{\Sigma}_i}(\mathbf{x} - \mu_i)$ along its principal axis into a two-component mixture, whose first two moments match those of the original component. More precisely, let $\mathbf{U}\mathbf{D}\mathbf{U}^{\mathrm{T}} = \mathbf{\Sigma}_i$ be a singular value decomposition of $\mathbf{\Sigma}_i$ with eigenvalues and eigenvectors ordered by the descending eigenvalues. Then the new detailed mixture model is defined as

$$q_i(\mathbf{x}) = \sum_{k=1}^{2} \alpha_k \phi_{\mathbf{\Sigma}_k}(\mathbf{x} - \mu_k), \tag{23}$$
$$\mu_1 = \mathbf{FM} + \mu_i \ ; \ \mu_2 = \mathbf{FM} - \mu_i,$$
$$\mathbf{\Sigma}_k = \mathbf{FCF}^{\mathrm{T}} \ ; \ \alpha_k = \frac{1}{2} w_i,$$

where $\mathbf{C} = diag([3/4, \mathbf{0}_{1\times(d-1)}])$, $\mathbf{M} = [0.5, \mathbf{0}_{1\times(d-1)}]^{\mathrm{T}}$, $\mathbf{F} = \mathbf{U}\sqrt{\mathbf{D}}$ and $\mathbf{0}_{1\times(d-1)}$ is all-zeros row vector of length $(d-1)$. An example of revitalization is illustrated in Figure 6, and the entire compression procedure along with the revitalization routine is summarized in the Algorithm 1.

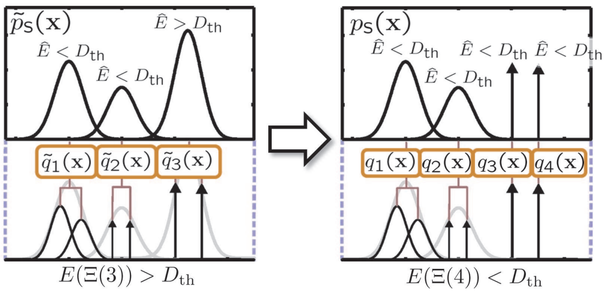

$\tilde{p}_s(\mathbf{x})$
$\hat{E} > D_{\mathrm{th}}$
$\hat{E} < D_{\mathrm{th}}$
$\hat{E} < D_{\mathrm{th}}$
$\tilde{q}_1(\mathbf{x})$ $\tilde{q}_2(\mathbf{x})$ $\tilde{q}_3(\mathbf{x})$
$E(\Xi(3)) > D_{\mathrm{th}}$
$p_s(\mathbf{x})$
$\hat{E} < D_{\mathrm{th}}$ $\hat{E} < D_{\mathrm{th}}$ $\hat{E} < D_{\mathrm{th}}$
$\hat{E} < D_{\mathrm{th}}$
$q_1(\mathbf{x})$ $q_2(\mathbf{x})$ $q_3(\mathbf{x})$ $q_4(\mathbf{x})$
$E(\Xi(4)) < D_{\mathrm{th}}$

Fig. 6. Illustration of the revitalization procedure. The left column shows the sample distribution $\tilde{p}_s(\mathbf{x})$ along with its detailed model $\{\tilde{q}(\mathbf{x})\}_{i=1:3}$ prior to revitalization. The $\hat{E} > D_{\mathrm{th}}$ indicates that an over compression was detected for the third component (from left to right) in the sample distribution. The revitalized sample distribution and new detailed model are shown in the right column, where the third component was replaced by its detailed model.

V. Online Kernel Density Estimation

A key point of the online kernel density estimation is the ability to properly update the sample distribution and to recalculate the bandwidth as new samples arrive. Furthermore, if the samples arrive from a non-stationary process, then a *forgetting factor* should be taken into account since the recently observed samples are better representatives of the current underlying distribution. In this section, we describe an iteration of the proposed online kernel density estimation algorithm, whose steps were outlined in the introduction (Figure 1a).

Let us denote the model of the samples observed up to time-step $(t-1)$ as

$$\mathbf{S}_{\mathrm{model}(t-1)} = \{p_{\mathrm{s}(t-1)}(\mathbf{x}), \{q_{i(t-1)}(\mathbf{x})\}_{i=1:M_{t-1}}\}, \tag{24}$$

where $p_{\mathrm{s}(t-1)}$ is a M_{t-1}-component sample distribution,

$$p_{\mathrm{s}(t-1)}(\mathbf{x}) = \sum\nolimits_{i=1}^{M_{t-1}} \alpha_i \phi_{\mathbf{\Sigma}_{si}}(\mathbf{x} - \mu_i). \tag{25}$$

Let N_{t-1} denote the *effective number* of observations[2] up to time-step $(t-1)$, let $N_{\alpha(t-1)}$ be the value of the parameter for bandwidth calculation (N_α in equation 6) and let f be a forgetting factor.

[2]Note that if there is no forgetting involved then all the data-points are equally important, regardless of the order in which they arrive. In this case the effective number of observations is just the number of all observed samples.

Algorithm 1 : Compression of the sample model

```
Input:
    $\tilde{\mathbf{S}}_{\mathrm{model}} = \{\tilde{p}_s(\mathbf{x}), \{\tilde{q}_i(\mathbf{x})\}_{i=1:\tilde{M}}\}$ ... the $\tilde{M}$-component sample model.
    $\mathbf{H}_{\mathrm{opt}}$ ... the current optimal bandwidth.
    $D_{\mathrm{th}}$ ... the maximal allowed local compression error.
Output:
    $\hat{\mathbf{S}}_{\mathrm{model}} = \{\hat{p}_s(\mathbf{x}), \{\hat{q}_j(\mathbf{x})\}_{j=1:M}\}$, ... the compressed $M$-component sample model.
    Procedure:
 1: Revitalize each $i$-th component in $\tilde{p}_{\mathrm{s}}(\mathbf{x})$ for which $\hat{E}(\tilde{q}_i(\mathbf{x}), \mathbf{H}_{opt}) > D_{th}$ according to Section IV-C and replace the sample model with the $N$-component revitalized model: $\mathbf{S}_{\mathrm{model}} \longleftarrow \{p_s(\mathbf{x}), \{q_i(\mathbf{x})\}_{i=1:N}\}$.
 2: Initialize the cluster set:
    $\Xi(M) = \{\pi_1\}$, $\pi_1 = \{1, \ldots, N\}$, $M = 1$
 3: while $D_{\mathrm{th}} < \max_{\pi_j \in \Xi(M)} \hat{E}(p_{\mathrm{s}}(\mathbf{x}; \pi_j))$ do
 4:     Select the cluster with the maximum local error:
        $\pi_j = \arg\max_{\pi_j \in \Xi(M)} \hat{E}(p_{\mathrm{s}}(\mathbf{x}; \pi_j))$
 5:     Split the sub-mixture $p_{\mathrm{s}}(\mathbf{x}; \pi_j)$ into two sets using the Goldberger's $K$-means: $\pi_j \longrightarrow \{\pi_{j1}, \pi_{j2}\}$.
 6:     Update the cluster set:
        $M \longleftarrow M + 1$, $\Xi(M) \longleftarrow \{\{\Xi(M) \setminus \pi_j\}, \pi_{j1}, \pi_{j2}\}$.
 7: end while
 8: Regroup the components of $p_{\mathrm{s}}(\mathbf{x})$ according to clustering $\Xi(M)$ and construct the compressed sample model $\hat{p}_{\mathrm{s}}(\mathbf{x})$.
 9: For each $j$-th component in $\hat{p}_{\mathrm{s}}(\mathbf{x})$ create its detailed model $\hat{q}_j(\mathbf{x})$ from the reference detailed models $\{q_i(\mathbf{x})\}_{i=1:N}$ according to the clustering $\Xi(M)$.
```

At time-step t we observe a sample $\mathbf{x}_t$ and reestimate the sample model $\mathbf{S}_{\mathrm{model}(t)} = \{p_{\mathrm{s}(t)}(\mathbf{x}), \{q_{i(t)}(\mathbf{x})\}_{i=1:M_t}\}$ (and hence the KDE) in the following steps.

Step 1: Update the sample model. The effective number of observed samples is augmented using the forgetting factor, $N_t = N_{t-1}f + 1$ and the weight $w_0 = N_t^{-1}$ of the new sample is computed. The sample distribution is updated by the new observation[3] as

$$\tilde{p}_{s(t)}(\mathbf{x}) = (1 - w_0)p_{s(t-1)}(\mathbf{x}) + w_0 \phi_{\mathbf{0}}(\mathbf{x} - \mathbf{x}_t). \tag{26}$$

The detailed model $\tilde{q}_{\tilde{M}_t}(\mathbf{x}) = \phi_{\mathbf{0}}(\mathbf{x} - \mathbf{x}_t)$ corresponding to $\mathbf{x}_t$ is added to the existing set of detailed models

$$\{\tilde{q}_{i(t)}(\mathbf{x})\}_{i=1:\tilde{M}_t} = \{\{q_i(\mathbf{x})\}_{i=1:M_{t-1}}, \tilde{q}_{\tilde{M}_t}(\mathbf{x})\}, \tag{27}$$

Thus yielding an updated sample model

$$\tilde{\mathbf{S}}_{\mathrm{model}(t)} = \{\tilde{p}_{s(t)}(\mathbf{x}), \{\tilde{q}_{i(t)}(\mathbf{x})\}_{i=1:\tilde{M}_t}\}. \tag{28}$$

Step 2: Reestimate the bandwidth. The empirical covariance of the observed samples $\hat{\mathbf{\Sigma}}_{\mathrm{smp}}$ is calculated by approximating $\tilde{p}_{\mathrm{s}(t)}(\mathbf{x})$ by a single Gaussian using the moment matching (17) and the parameter for bandwidth calculation is updated as $N_{\alpha t} = (N_{\alpha(t-1)}^{-1}(1 - w_0)^2 + w_0^2)^{-1}$. The

[3]Note that $\tilde{(\cdot)}$ denotes the updated model before the compression.

new optimal bandwidth is then approximated according to Section III as

$$\mathbf{H}_t = \mathbf{F}[d(4\pi)^{d/2}N_{\alpha t}\hat{R}(p,\mathbf{F},\mathbf{G})]^{\frac{-1}{d+4}} \tag{29}$$

with $\mathbf{F} = \hat{\mathbf{\Sigma}}_{\mathrm{smp}}$, $\mathbf{G} = \hat{\mathbf{\Sigma}}_{\mathrm{smp}}(\frac{4}{(2+d)N_{\alpha t}})^{\frac{2}{d+4}}$, and with the functional $\hat{R}(p,\mathbf{F},\mathbf{G})$ calculated according to (11).

Step 3: Refine and compress the model. After the current bandwidth $\mathbf{H}_t$ has been calculated, the sample model $\tilde{\mathbf{S}}_{\mathrm{model}(t)}$ is refined and compressed, using the Algorithm 1, into

$$\mathbf{S}_{\mathrm{model}(t)} = \{p_{\mathrm{s}(t)}(\mathbf{x}), \{q_{i(t)}(\mathbf{x})\}_{i=1:M_t}\}. \tag{30}$$

In our implementation, the compression is called after some threshold on number of components M_{thc} has been exceeded. Note that this threshold does not determine the number of components in the final model, but rather influences the *frequency* at which the compression is called. To avoid too frequent calls to compression, the threshold is also allowed to vary during the online operation using a simple hysteresis rule: If the number of components M_t still exceeds M_{thc} after the compression, then the threshold increases $M_{\mathrm{thc}} \leftarrow 1.5M_{\mathrm{thc}}$, otherwise, if $M_{\mathrm{t}} < \frac{1}{2}M_{\mathrm{thc}}$, then it decreases $M_{\mathrm{thc}} \leftarrow 0.6M_{\mathrm{thc}}$.

Recalculate the KDE: After the three steps of the online update have finished, the sample distribution is a M_t-component mixture model

$$p_{\mathrm{s}(t)}(\mathbf{x}) = \sum_{i=1}^{M_t}\alpha_i\phi_{\mathbf{\Sigma}_{\mathrm{s}i}}(\mathbf{x}-\mu_t), \tag{31}$$

and the current KDE is calculated from the sample distribution according to (3):

$$\begin{aligned} p_{\mathrm{KDE}t}(\mathbf{x}) &= p_{\mathrm{s}(t)}(\mathbf{x}) * \phi_{\mathbf{H}_t}(\mathbf{x}) \\ &= \sum_{i=1}^{M_t}\alpha_i\phi_{\mathbf{\Sigma}_{\mathrm{s}i}+\mathbf{H}_t}(\mathbf{x}-\mu_i). \end{aligned} \tag{32}$$

A. Preprocessing by removing the null-space

Since the samples used to build the oKDE model may not span the entire data space but rather lie on a lower-dimensional manifold, the updates in the oKDE may suffer from singular covariances. For example, this is usually true for the first few updates when we initialize the oKDE using smaller number of samples than the sample dimensionality. To avoid these pathologies, the oKDE and the new data-point are first (prior to update) projected into a subspace using a principal component analysis, the updates are carried out there, and then the oKDE is backprojected into the original space.

VI. Experimental study

A significant difference between the online and batch algorithms is that the batch algorithms have access to all data during the learning, while the online algorithms observe only portions of the data at a time, discard the old observations and retain only the models derived from them. Another difference is that the online approaches my deal with only a single data-point (or a few) at time, and thus different data orderings may affect the model's quality. To analyze the effects of these differences, the experiments were divided into five parts. In the first four parts, we have analyzed the oKDE's reconstructive performance, while the last experiment analyzed the oKDE's discriminative performance. In the first experiment (section VI-A) we have analyzed how the parameters and data orderings affect the reconstructive performance of the oKDE. An important part of the proposed oKDE algorithm is the revitalization of the sample distribution, which allows recovery from early over-compressions (as discussed in section IV-C) and helps to correctly adapt the model's structure as new data arrive. The second experiment (section VI-B) was therefore dedicated to the analysis of the revitalization effects. In the third experiment (section VI-C) we have compared the oKDEs reconstructive performance to the related state-of-the-art batch and online methods in estimating stationary distributions. In the fourth experiment (section VI-D) we have analyzed the oKDE's reconstructive performance in estimating a non-stationary distribution. In the last experiment (section VI-E) we have analyzed the oKDE's discriminative performance in application of online construction of a Bayes classifier. In all the experiments, except for the fourth which involved estimating a non-stationary distribution, the forgetting factor in the oKDE was set to $f = 1$. This effectively assigned equal importance to all data, regardless of the order. The experiments were performed on a standard 2GHz CPU, 2GB RAM PC in Matlab.

A. Influence of the compression parameter D_{th} and data order

The only free parameter in the oKDE is the compression parameter D_{th}, which quantifies the local approximation error during compression (and revitalization) in terms of the unscented Hellinger distance. The aim of the first part of the experiment was therefore to illustrate how the different values of this parameter affect the oKDE's performance. The experiments involved approximating a spiral-shaped two-dimensional stationary distribution defined as

$$\begin{gathered} \mathbf{x} = [(1+\theta)\cos(\theta), (1+\theta)\sin(\theta)]^{\mathrm{T}} + \mathbf{w} \\ \mathbf{w} \sim \phi_{\mathbf{\Sigma}_{\mathbf{w}}}(\cdot)\ ;\ \theta \sim \mathcal{U}(0,10) \end{gathered} \tag{33}$$

where $\mathbf{\Sigma}_{\mathbf{w}} = \mathrm{diag}\{0.9^2, 0.9^2\}$ and $\mathcal{U}(0,10)$ is a uniform distribution on interval $[0,10]$. An example of the spiral model along with its distribution is shown in Figure 7. A set of 1000 samples was generated from this distribution – the first ten samples were used for initialization and the rest were used one at a time to update the oKDE. After all 1000 samples have been observed, the reconstructive performance of the KDE model was evaluated as the average negative log-likelihood of additionally drawn 20,000 samples.

The performance of the oKDE with various compression values was compared with the *adaptive mixtures* (AM) approach [34], which is essentially an online EM algorithm for Gaussian mixture models with an automatic component-allocation heuristic. The performance results are shown under the "*random order*" label in Table I. We can see that after observing the 1000th sample, the oKDE with smallest compression threshold produced models with the smallest error and produced a mixture model which on average contained 37 components. By increasing the compression threshold, the

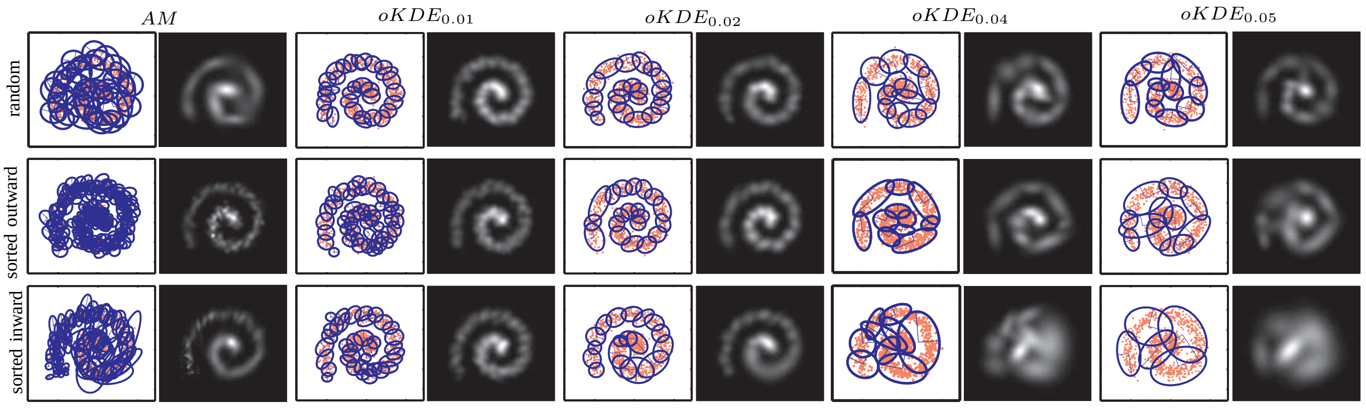


Fig. 8. Mixture models of the spiral distribution from Figure 7 obtained by different orderings of data: random (first row), sorted outward (second row) and sorted inward (third row). Each model is shown as a decomposed mixture model, and as an image of its distribution.

TABLE I
THE AVERAGE NEGATIVE LOG-LIKELIHOOD ($-\mathcal{L}$) AND THE NUMBER OF COMPONENTS IN THE MODEL (N_{cmp}) FOR oKDE$_{D_{\mathrm{th}}}$ AND AM W.R.T. TO THE DATA ORDERS: RANDOM, CENTER-TO-OUTERMOST AND OUTERMOST-TO-CENTER.

	[median, mean ± standard deviation]					
	random		sorted outward		sorted inward	
method	$-\mathcal{L}$	N_{cmp}	$-\mathcal{L}$	N_{cmp}	$-\mathcal{L}$	N_{cmp}
AM	$5.45,\ 5.45 \pm 0.0431$	$43,\ 43.3 \pm 7.25$	$5.46,\ 5.46 \pm 0.0139$	$217,\ 216 \pm 12.5$	$5.41,\ 5.41 \pm 0.00907$	$115,\ 115 \pm 6.3$
oKDE$_{0.01}$	**5.39, 5.39±0.00548**	$37,\ 37.7 \pm 3.22$	**5.39, 5.39±0.00526**	$49,\ 48.9 \pm 2.6$	**5.39, 5.39±0.0055**	$36,\ 36.8 \pm 2.8$
oKDE$_{0.02}$	$5.39,\ 5.39 \pm 0.00589$	$19,\ 19.6 \pm 1.43$	$5.39,\ 5.39 \pm 0.00602$	$20,\ 20.3 \pm 2.48$	$5.41,\ 5.41 \pm 0.011$	$18,\ 17.8 \pm 1.38$
oKDE$_{0.03}$	$5.42,\ 5.43 \pm 0.0157$	$14,\ 14.4 \pm 1.41$	$5.43,\ 5.43 \pm 0.0199$	$12,\ 12.4 \pm 1.4$	$5.57,\ 5.57 \pm 0.0458$	$12.5,\ 12.6 \pm 1.33$
oKDE$_{0.04}$	$5.48,\ 5.48 \pm 0.0302$	$12,\ 11.8 \pm 1.44$	$5.61,\ 5.63 \pm 0.0878$	$9,\ 8.97 \pm 1.5$	$5.77,\ 5.76 \pm 0.0602$	$8,\ 7.5 \pm 1.5$
oKDE$_{0.05}$	$5.56,\ 5.55 \pm 0.0591$	$10,\ 10 \pm 1.76$	$5.87,\ 5.83 \pm 0.0706$	$5,\ 5.03 \pm 1.4$	$5.84,\ 5.84 \pm 0.0372$	$5,\ 5.2 \pm 1.03$
oKDE$_{0.06}$	$5.58,\ 5.59 \pm 0.047$	$8,\ 8.27 \pm 1.05$	$5.9,\ 5.9 \pm 0.0247$	$3,\ 3.43 \pm 0.858$	$5.89,\ 5.88 \pm 0.0287$	$4,\ 3.93 \pm 0.74$
oKDE$_{0.07}$	$5.72,\ 5.72 \pm 0.113$	$6,\ 5.87 \pm 1.94$	$5.92,\ 5.91 \pm 0.0195$	$3,\ 2.7 \pm 0.535$	$5.91,\ 5.9 \pm 0.0345$	$3,\ 3.2 \pm 1.21$

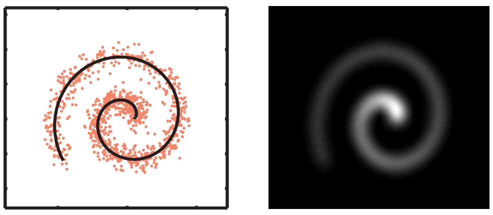


Fig. 7. The left image shows the mean of the spiral model (dark) along with some samples (bright) generated from the model, while the right image shows the corresponding spiral distribution.

number of components decreased, while the approximation error increased. In terms of the approximation error, the oKDE outperformed the AM for D_{th} values smaller than 0.03. Note that the AM-estimated models contained on average 45 components, while for example, the oKDE with $D_{\mathrm{th}} = 0.02$ produced more accurate models which contained on average 20 components. For convenience we show typical estimated models for AM and oKDE in the first row of Figure 8. By comparing these images with the image of the reference distribution in Figure 7, we can visualize the reduction of the complexity of the estimated distribution with increasing the compression threshold in the oKDE.

1) Sorted data: In the previous part of the experiment the data-points have been sampled randomly from the entire distribution. Therefore the information about *the scale* of the target distribution was present in the observed samples throughout the experiment. While online estimation of a distribution from a single-time-step observation is difficult as it is, it becomes more difficult if the information about the scale of the target distribution is not available from the start. To simulate these conditions, we have performed two variations of the experiment with the 2D spiral distribution in which we enforced a predefined order to the observed samples. The order was enforced by deterministically selecting the values of the position parameter θ (33) along the spiral at equal distances from the interval $[0, 10]$.

In the first variant, the position values θ were organized in an ascending order, thus yielding an *outward* ordering of data-points from the spiral's center, while the second variation used a descending order of position values, which yielded an *inward* ordering. In both orderings, the early samples indicated a small scale of the estimated distribution, and the entire scale became evident slowly at later time-steps. In the outward ordering the scale became apparent only slowly, since a large number of samples are concentrated at the center of the spiral. This trend was faster for the inward ordering. The results for the outward and inward ordering are given in the second and third columns of the Table I. With increasing D_{th} the oKDE produced models with lower number of components at a cost of larger approximation error. With respect to the values of D_{th}, the number of components remained comparable with the random sampling. On the other hand, the AM produced models whose complexity was significantly larger. This can be attributed directly to the missing scale information in the early samples, which initially caused allocation of a larger number of components in the AM model. The accuracy of the oKDE for outward ordering is comparable to random ordering for values $D_{\mathrm{th}} < 0.04$. However, for the inward ordering, the decrease in accuracy in comparison to random ordering becomes already apparent for values $D_{\mathrm{th}} = 0.03$ (Table I, last column).

With increasing the compression threshold D_{th}, the degradation of the models in oKDE was faster for inward than outward ordering. The reason is that greater D_{th} allows grater loss of information about the structure of the distribution during online estimation. In absence of the structure information the models deteriorate. Note, however, that even in the case of inward ordering with $D_{\mathrm{th}} = 0.05$, the models retained a rough structure of the spiral distribution (see, Figure 8, last row, fifth pair).

To estimate how the oKDE performs regardless of the data order, the results of Table I were averaged over the different data orders. The results are shown in Table II and visualized in Figure 9. We see that on average the oKDE with $D_{\mathrm{th}} < 0.03$ outperforms the AM by producing models with smaller errors and smaller number of components. With increasing the compression values, the number of components further decreases, while the errors increase.

TABLE II
THE AVERAGE NEGATIVE LOG-LIKELIHOOD ($-\mathcal{L}$) AND THE NUMBER OF COMPONENTS IN THE MODEL (N_{cmp}) FOR oKDE$_{D_{\mathrm{th}}}$ AND AM AVERAGED OVER THE DIFFERENT DATA ORDERS.

	[median, mean ± standard deviation]	
	$-\mathcal{L}$	N_{cmp}
AM	5.44, 5.44 ± 0.035	115, 125 ± 71.6
oKDE$_{0.01}$	**5.39, 5.39±0.00566**	39, 41.1±6.23
oKDE$_{0.02}$	5.39, 5.4±0.0138	19, 19.2±2.1
oKDE$_{0.03}$	5.44, 5.48±0.0745	13, 13.1±1.65
oKDE$_{0.04}$	5.61, 5.63±0.132	9, 9.43±2.33
oKDE$_{0.05}$	5.82, 5.74±0.149	6, 6.74±2.71
oKDE$_{0.06}$	5.87, 5.79±0.147	4, 5.21±2.35
oKDE$_{0.07}$	5.9, 5.85±0.111	3, 3.92±1.94

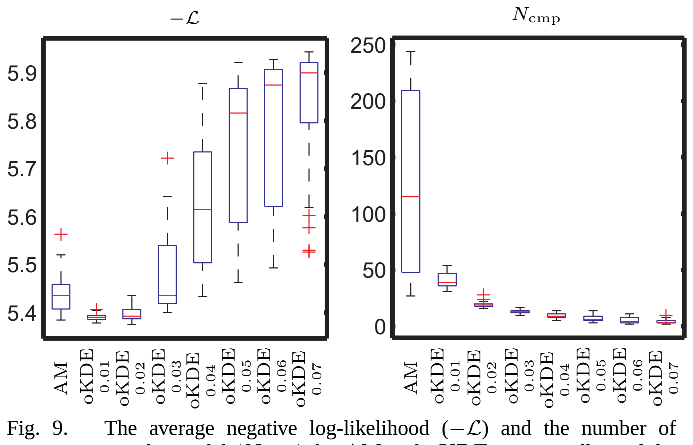


Fig. 9. The average negative log-likelihood ($-\mathcal{L}$) and the number of components in the model (N_{cmp}) for AM and oKDE$_{D_{\mathrm{th}}}$ regardless of the order of data.

B. Influence of the revitalization scheme

An important part of the oKDE is the revitalization of the sample distribution. As discussed in section IV-C, merging certain components in the sample distribution into a common cluster may be valid at some point in time, but this compression can become invalid later on as new data arrive and new structures in the distribution should become apparent. An example of such compression was illustrated in Figure 5. To more concretely demonstrate how the revitalization influences the quality of estimation with increasing the number of observations, we have performed density estimation using the oKDE with and without the revitalization scheme. In this experiment we have generated 1000 samples from a heavily skewed one-dimensional reference distribution (see, Figure 10), and used one sample at a time with the $oKDE$ to approximate this distribution. Figure 10 (upper row) shows examples of the estimated distributions after observing 100 and 1000 samples without using the revitalization scheme, with the compression threshold set to $D_{\mathrm{th}} = 0.02$. We see that after observing 100 samples the reference distribution is approximated by an over-smoothed four-component model. The error of this over-compression is propagated further as new samples arrive and prohibits a significant improvement of the model even after additional nine hundred samples. On the other hand, when using the oKDE with the revitalization scheme activated, the model better approximates the reference pdf already after the first 100 samples (Figure 10, lower row). As more samples are observed, the revitalization scheme revitalizes the over-compressed components and better adapts the model's complexity, resulting in a significantly better approximation. The graphs in Figure 11 (left column) show the L_1 distances between the reference distribution and its approximations, for the different values of the compression threshold D_{th}, w.r.t. the number of the observed samples averaged over 20 trials. Without the revitalization, the distances decrease only for the first 50 to 200 samples, and then remain approximately constant. For the first hundred samples, these results are very similar to the case when applying the revitalization scheme. However, in case of the revitalization scheme the distances further decrease with increasing the number of samples.

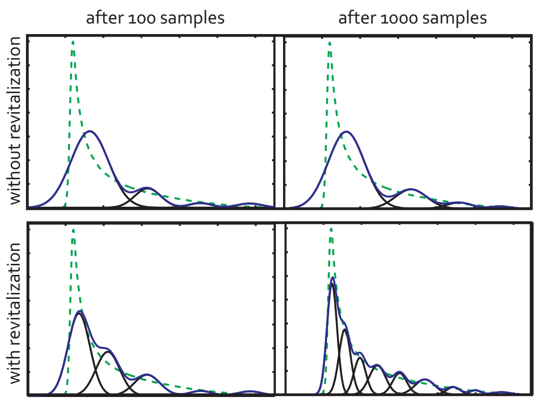


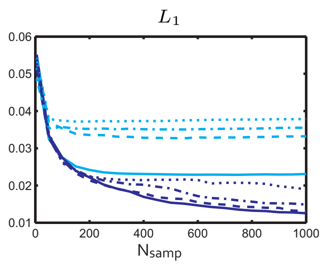
Fig. 10. The upper row shows two distributions estimated after observing 100 and 1000 samples without using the revitalization scheme in the oKDE, while the lower row shows the results of enabling the revitalization in the oKDE. The reference distribution is depicted by dashed line (green), while the estimated distribution is depicted by solid (blue) line.

To further illustrate the improvements of using the revitalization scheme, we have calculated the improvement factors

γ_i w.r.t. the number of samples as

$$\gamma_i = \frac{(\hat{\varepsilon}_i - \varepsilon_i)}{\hat{\varepsilon}_i}, \tag{34}$$

where $\hat{\varepsilon}_i$ is the L_1 distance between the reference distribution and the model without revitalization, ε_i is the L_1 distance between the reference distribution and the model with revitalization and i is the index of the observed sample. Figure 11 (right column) shows these results for the different values of the compression threshold D_{th}. We can see that the improvement of using the revitalization scheme increases with the number of samples regardless of the compression threshold D_{th}. For example, after observing 1000 samples, the improvement for all tested values D_{th} was between 45% and 65%.



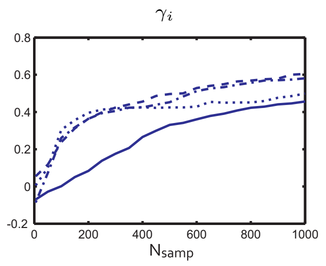


Fig. 11. The left plot shows the L_1 distance errors for the $oKDE$ with the revitalization scheme (bright cyan), and without the revitalization scheme (dark blue), w.r.t. the number of samples N_{samp}. The right graph shows the improvements in terms of error reduction (improvement factor γ_i). The results for $oKDE_{0.01}$, $oKDE_{0.02}$, $oKDE_{0.04}$ and $oKDE_{0.05}$ are depicted by solid, dashed, dash-dotted and dotted lines, respectively.

C. Comparison with batch methods

As noted earlier, a significant difference between the online and batch algorithms is that the batch algorithms have constant access to all data, while the online algorithms discard the data and retain only the models derived from the observed data. These models thus have to retain enough information to be able to successfully update when the new observations arrive. We have therefore compared the performance of the oKDE with three batch state-of-the-art KDE methods: Hall's et. al. [17] plug-in (implementation [20]), Murillo's et. al. [30] cross validation and Girolami's et. al. [13] reduced-set-density estimator initialized by the cross validation estimator. For reference, we have compared the oKDE also to the adaptive mixtures [34].

The experiment consisted of two parts. The first part of the experiment involved estimation of a 2D noisy sinusoidal distribution defined by

$$\mathbf{x} = [a, \sin(3a) + w]^T \tag{35}$$
$$a = 4(t - 1/2)\ ;\ w \sim \phi_{\sigma_w}(\cdot)$$

with $\sigma_w = 0.2^2$. The mean of the model along with some generated samples is shown in Figure 12a. Ten thousand samples were generated from this distribution – the first four were used for initialization and the rest were used one at a time to update the oKDE. The reconstructive performance of the models was evaluated by the average negative log-likelihood of additionally sampled 20,000 observations. The experiment was repeated 10 times for different random sets of samples. An example of an $\mathrm{oKDE}_{0.02}$ model after observing a thousand samples is shown in Figure 12b.

The second part of the experiment involved estimating a three-dimensional stationary spiral distribution defined by the following model

$$\mathbf{x} = [(13 - \frac{1}{2}t)\cos(t), -(13 - \frac{1}{2}t)\sin(t), t]^T + \mathbf{w} \tag{36}$$
$$\mathbf{w} \sim \phi_{\mathbf{\Sigma_w}}(\cdot)\ ;\ t \sim \mathcal{U}(0, 14),$$

where $\mathbf{\Sigma_w} = \mathrm{diag}\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}\}$, and $\mathcal{U}(1, 14)$ is a uniform distribution constrained to interval $[0, 14]$. A set of samples generated from the model (36) is shown in Figure 12c. Again, a set of ten thousand test samples was generated from this model – the first 10 samples were used for initialization and the rest were used one at a time with the oKDE, to approximate the underlying distribution. The reconstructive performance of the models was evaluated by the average negative log-likelihood of additionally sampled 20,000 observations. This experiment was also repeated 10 times. An example of the estimated distribution with $\mathrm{oKDE}_{0.02}$ after observing a thousand samples is shown in Figure 12d.

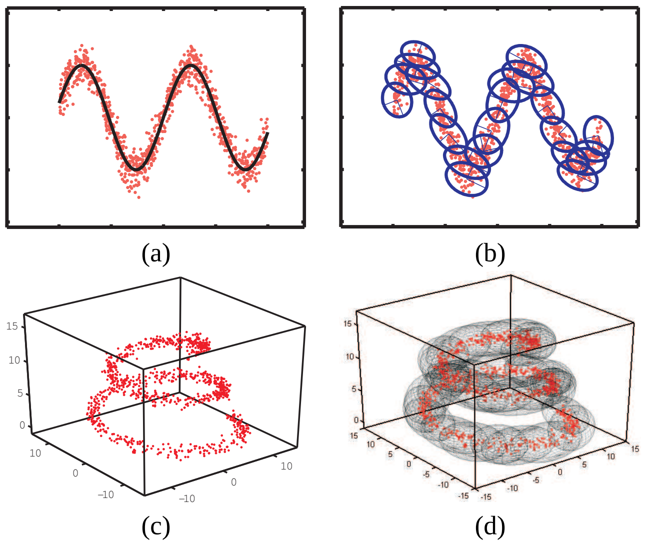


Fig. 12. First row shows the sinusoidal distribution and the second row shows the spiral distribution. Left column shows the reference distributions and the right column shows the estimated distribution using $\mathrm{oKDE}_{0.02}$ after observing a 1000 samples.

The results of the two experiments are summarized in Table III, Table IV and Figure 13. The different estimators are denoted as: cross validation (CV), Hall's plug-in (Hall), reduced-set-density estimator (RSDE), the adaptive mixtures (AM) and the online KDE with compression threshold D_{th} ($\mathrm{oKDE}_{D_{\mathrm{th}}}$).

Among the batch approaches, the CV outperformed the other two batch methods in terms of accuracy. While the advantage of the batch methods is that they optimize their parameters by having access to all the data-points, they become increasingly slow with increasing the number of data-points and can also run into computer's memory constraints. Indeed this was the case for the particular implementations of the

TABLE III
THE AVERAGE NEGATIVE LOG-LIKELIHOOD $-\mathcal{L}$ AND THE NUMBER OF COMPONENTS IN THE MODEL (N_{cmp}) W.R.T. THE NUMBER OF OBSERVED SAMPLES FOR THE EXPERIMENT WITH THE SINUSOIDAL DISTRIBUTION. THE SYMBOL "/" INDICATES THAT THE ESTIMATOR COULD NOT BE CALCULATED DUE TO MEMORY LIMITATIONS.

	50 samples		1000 samples		6000 samples		8000 samples		10000 samples	
Batch methods	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$
CV	**1.71±0.10**	50±0.0	**1.30±0.01**	1000±0.0	1.33±0.00	6000±0.0	**1.32±0.00**	8000±0.0	**1.35±0.00**	10000±0.0
Hall	2.35±0.03	50±0.0	1.98±0.01	1000±0.0	1.79±0.01	6000±0.0	1.76±0.00	8000±0.0	/	/
RSDE	1.98±0.15	23±4.2	1.30±0.01	380±11.4	**1.26±0.01**	2203±47.4	/	/	/	/
Online methods	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$
AM	2.17±0.12	11±2.7	1.72±0.14	22±3.9	1.54±0.14	38±5.8	1.52±0.13	41±6.3	1.50±0.13	43±6.3
$\mathrm{oKDE}_{0.01}$	**1.97±0.07**	16±1.6	**1.45±0.01**	34±1.8	**1.31±0.01**	48±2.6	**1.30±0.01**	51±2.8	**1.30±0.01**	54±2.6
$\mathrm{oKDE}_{0.02}$	1.98±0.09	12±1.7	1.48±0.01	21±2.1	1.45±0.03	28±1.9	1.45±0.03	29±2.6	1.44±0.03	30±2.3
$\mathrm{oKDE}_{0.04}$	1.97±0.05	8±1.1	1.65±0.05	11±1.7	1.61±0.03	13±1.6	1.61±0.03	14±1.5	1.60±0.03	14±1.6
$\mathrm{oKDE}_{0.05}$	2.00±0.05	6±1.1	1.73±0.03	9±1.1	1.65±0.04	10±1.4	1.64±0.04	11±1.3	1.64±0.04	11±1.6

TABLE IV
THE AVERAGE NEGATIVE LOG-LIKELIHOOD $-\mathcal{L}$ AND THE NUMBER OF COMPONENTS IN THE MODEL (N_{cmp}) W.R.T. THE NUMBER OF OBSERVED SAMPLES FOR THE EXPERIMENT WITH THE 3D SPIRAL. THE SYMBOL "/" INDICATES THAT THE ESTIMATOR COULD NOT BE CALCULATED DUE TO MEMORY LIMITATIONS.

	50 samples		1000 samples		6000 samples		8000 samples		10000 samples	
Batch methods	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$
CV	**8.07±0.25**	50±0.0	**6.61±0.01**	1000±0.0	**6.50±0.00**	6000±0.0	**6.53±0.01**	8000±0.0	**6.52±0.01**	10000±0.0
Hall	8.14±0.20	50±0.0	6.95±0.01	1000±0.0	6.69±0.01	6000±0.0	/	/	/	/
RSDE	8.64±0.67	30±7.6	6.69±0.03	516±82.5	6.57±0.01	2614±17.1	/	/	/	/
Online methods	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$	$-\mathcal{L}$	$\mathrm{N_{cmp}}$
AM	8.66±0.16	18±2.7	6.93±0.09	42±4.3	6.62±0.07	64±6.0	6.60±0.06	68±5.8	6.58±0.06	72±6.2
$\mathrm{oKDE}_{0.01}$	**8.03±0.20**	24±1.5	**6.75±0.01**	46±2.4	**6.49±0.00**	51±1.7	**6.47±0.00**	52±1.8	**6.45±0.00**	52±1.4
$\mathrm{oKDE}_{0.02}$	8.04±0.25	19±1.7	6.77±0.01	29±1.4	6.53±0.01	32±1.1	6.52±0.01	33±1.1	6.51±0.01	33±1.2
$\mathrm{oKDE}_{0.04}$	8.14±0.30	14±1.3	6.84±0.01	20±1.1	6.69±0.04	23±1.5	6.66±0.02	24±1.3	6.64±0.02	24±1.2
$\mathrm{oKDE}_{0.05}$	8.08±0.28	13±1.4	6.88±0.03	18±0.9	6.75±0.03	21±1.1	6.72±0.04	21±1.2	6.71±0.04	21±1.1

batch RSDE and Hall, which prohibited estimation for very large sets of samples. This is indicated in the Table III and Table IV by the symbol "/". For smaller number of samples, the batch CV outperformed the online methods in terms of accuracy, however, at a cost of severely increased model complexity. For example, after observing thousand data-points, the complexity of CV model was one-thousand components, while the complexity of the $\mathrm{oKDE}_{0.01}$ was only 5% of that. For increasing the number of samples over (approximately 6000), the $oKDE$ started to outperform the CV also in terms of better accuracy, while maintaining the number of components low. In Figure 13 we can see that in the case of CV, the number of components increased linearly with the number of samples, while in the case of oKDE models, the number of components initially increases, but then, after approximately 500 samples stabilizes and increases only little. For example, after observing 10000 samples, the complexity of the CV model was 10000 components, while the complexity of the $oKDE_{0.01}$ was approximately 52 components. While the number of components from 6000th to 10000th sample increased by 4000 in CV model, this increase was in the $oKDE_{0.01}$ on average by six components in the case of sinusoidal distribution (Table III) and only 1 component in the case of the spiral distribution (Table IV). All online methods (oKDE as well as AM) on average produced models with a smaller number of components than the batch RSDE. In all experiments, the $\mathrm{oKDE}_{0.01}$ and $\mathrm{oKDE}_{0.02}$ consistently outperformed the online AM model in terms of accuracy. The $\mathrm{oKDE}_{0.02}$ also consistently produced models of lower complexity than the AM. It is important to note that while the online methods (oKDE and AM) maintained a compressed model of the observed data, and were able to adapt the models with each new observation, the batch methods required processing also all of the previously observed samples along with the current observation, thus increasing their complexity as well as the complexity of their updates.

D. *Estimation of non-stationary distributions*

In the third experiment we applied the $oKDE_{0.02}$ to approximate a non-stationary distribution. This distribution was a mixture of two distributions,

$$p_0(\mathbf{x}, t) = w(t)p_1(\mathbf{x}) + (1 - w(t))p_2(\mathbf{x}), \tag{37}$$

whose mixing weight $w(t)$ was changing with time-steps t. The first distribution, $p_1(\mathbf{x})$, was a heavily skewed distribution (Figure 14a), while the second, $p_2(\mathbf{x})$, was a mixture of a uniform and a skewed distribution (Figure 14c). The weight was set to $w(t) = 1$ for the first 1000 samples and it gradually decreased to zero for the next 7000 samples at rate $w(t) = w(t-1)0.995$. Thus $p_0(\mathbf{x}, t)$ transited from pure $p_1(\mathbf{x})$ to pure $p_2(\mathbf{x})$. Figures 14(a,b,c) show the distribution at time-steps $t = 1$, $t = 1800$ and $t = 8000$, respectively.

Since the distribution was non-stationary, the forgetting factor in $oKDE_{0.02}$ was set to $f = 0.999$. Thus the effective sample size converges to $N_t = 1000$ and the $oKDE_{0.02}$ was performing as if it effectively observed only 1000 samples. The $oKDE_{0.02}$ was initialized from the first 3 samples and the rest were added one at a time. The quality of estimation at time-step t was measured by the $L1$ distance between the current estimate and $p_0(\mathbf{x}, t)$. The performance of the $oKDE_{0.02}$ was compared to the adaptive mixtures and the two sliding-window

TABLE V
THE L_1 DISTANCE AND THE NUMBER OF COMPONENTS IN THE MODEL (N_{cmp}) W.R.T. THE NUMBER OF OBSERVED SAMPLES FOR THE EXPERIMENT WITH THE NONSTATIONARY DISTRIBUTION.

	1000 samples		3000 samples		6000 samples		8000 samples	
	L_1	N_{cmp}	L_1	N_{cmp}	L_1	N_{cmp}	L_1	N_{cmp}
oKDE$_{0.02}$	**0.09±0.01**	10 ± 0.8	0.08±0.01	10 ± 0.92	**0.05±0.01**	10 ± 0.94	**0.04±0.00**	10 ± 1.3
CV	0.14 ± 0.01	1000 ± 0	**0.06±0.01**	1000 ± 0	0.06 ± 0.01	1000 ± 0	0.06 ± 0.00	1000 ± 0
Hall	0.16 ± 0.01	1000 ± 0	0.07 ± 0.00	1000 ± 0	0.08 ± 0.00	1000 ± 0	0.08 ± 0.00	1000 ± 0
AM	0.16 ± 0.06	8.4 ± 2.7	0.11 ± 0.02	9.6 ± 2.7	0.09 ± 0.03	11 ± 2.6	0.08 ± 0.03	11 ± 2.6

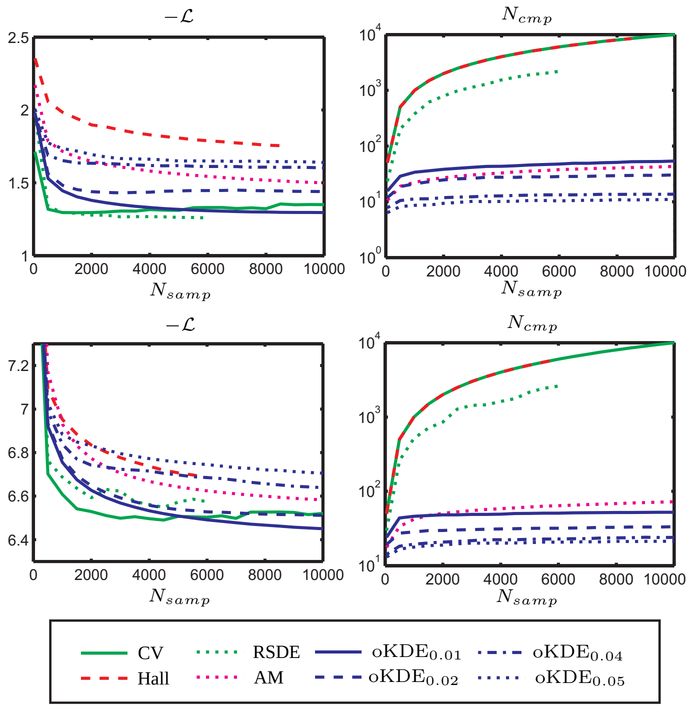


Fig. 13. The evolution of the average negative log likelihood and the number of components w.r.t. the number of samples for the sinusoidal distribution (upper row) and the spiral distribution (lower row). This image is best viewed in color.

batch methods, CV and Hall batch KDEs, which have been computed using the last 1000 observed samples. The forgetting factor in the AM was set as in the $oKDE_{0.02}$. Note that since the distribution was non-stationary, the AM algorithm was producing mixtures that contained many components whose weight was approximately zero. This comes from the fact, that the AM algorithm lacks a mechanism for removing redundant components. Therefore, during online operation of the AM, those components whose weight fell below 10^{-4} were removed from the mixture. Figure 15 and Table V summarize the results.

Both batch methods outperformed the AM model in terms of accuracy, however, they produced models of significantly greater complexity. On average, the $oKDE_{0.02}$ outperformed both, CV and Hall, batch KDEs by maintaining lower error and using one tenth as many components. The approximation error of models produced by the $oKDE_{0.02}$ was lower for 1000 samples, became slightly greater than that of the batch KDEs for 200 and 3000 samples, and then became again lower. We have noticed that in some (rare) cases, the CV produced an under-smoothed estimate of the distribution which temporarily increased the L_1 error. On the other hand, this behavior has not been observed for the oKDE, AM and the Hall's method. In all experiments, the $oKDE_{0.02}$ models with comparable accuracy to AM, however, it consistently outperformed the AM in terms of accuracy. An example of the model produced by the oKDE is shown in Figure 14(d).

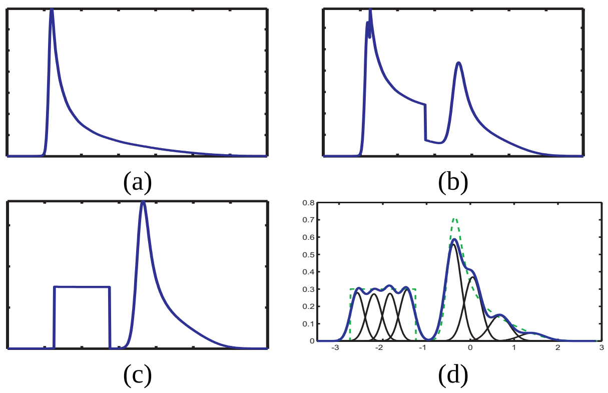


Fig. 14. The phases of the non-stationary distribution at $t = 1$ (a), $t = 1800$ (b) and $t = 8000$ (c), and the estimated distribution with oKDE after observing the 8000th sample (d). The components of the $oKDE_{0.02}$ model in (d) are depicted by solid thin lines and the oKDE is shown in solid thick line, while the reference distribution is depicted by a dashed green line.

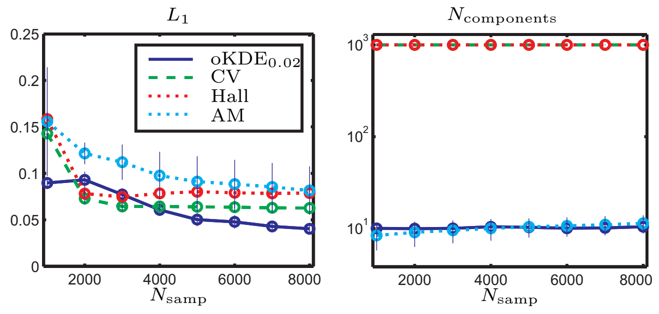


Fig. 15. The L_1 estimation error (left) and the number of components (right) w.r.t the time-step, along with one-standard-deviation bars. The results are shown for the oKDE (full line), CV (dashed), Hall (dotted dark) and AM (dotted bright).

E. Online models for discrimination

To analyze how the discriminative properties of the oKDE evolve during online estimation, we have compared the classification performance of the oKDE with three batch KDEs: the CV batch KDE [30], the reduced-set density estimator [13] initialized by the CV and the Hall's batch KDE [17] [4]. For the

[4]Since the Hall's KDE assumes a diagonal covariance matrix of the kernel, the data was pre-sphered before applying the Hall's KDE for better performance.

baseline classification, we have applied a multiclass SVM with an RBF kernel [7]. The methods were compared on a set of public classification problems [3] (Table VI). The classification performance of the KDE-based methods was tested using a simple Bayesian criterion

$$\hat{y} = \arg\max_{l} p_{\mathrm{KDE}}(\mathbf{x}|c_l). \quad (38)$$

The parameter for the SVM kernel was determined separately in each experiment via cross validation on the training data set. The classification experiment was conducted via four-fold cross validation. The oKDE was initialized from the first 10 samples and the rest were added one at a time. The compression threshold was set to $D_{\mathrm{th}} = 0.05$. The results of the experiments after observing all the data-points are summarized in Table VI, while the Fig. 16 shows the oKDE results during online estimation.

After observing all the data-points, the batch methods, SVM and CV, produced on average best classification. The oKDE outperformed batch RSDE and slightly Hall's KDE batch method, and produced a comparable classification to the SVM and the CV. An important observation is that the oKDE outperformed, or produced comparable performance, to the batch methods, even though the oKDE was constructed by observing only a single sample at a time. In contrast, the SVM and batch KDEs optimized their structure by having access to all the samples. A further thing to note is that, with the exception of the Pima dataset, the oKDE's classification performance quite closely matched that of an SVM, eventhough the oKDE is in its nature reconstructive, while the SVM optimizes its structure to maximize discrimination. Note also, that the complexity of the models learnt by batch KDEs is generally larger than that of the oKDE. For example, for the *letter* dataset, the oKDE required one third as many components to achieve a comparable performance to the CV and Hall's KDE. While the RSDE retained a significantly smaller number of components than the oKDE for the *letter* dataset, the RSDE's classification performance was significantly lower. By further inspection of the results in Fig. 16, we can also observe a general trend that the number of components initially significantly increases in the oKDE, but then stabilizes for larger number of samples, while the recognition score further improves. This is particularly evident for the *wine* and *Letter* datasets, in which the bound on the complexity is reached relatively early on after observing 90 and 200 samples per class (300th and 5200th sample), respectively, while the classification performance further increased through the model refinement. This makes the oKDE an appropriate tool for online operation since it produces compressed models with good classification performance, while at the same time allowing online refinements of the model without necessarily increasing their complexity with each training example.

VII. Conclusion

We have proposed an approach for a kernel density estimation which can be applied in online operation. The central point of the proposed scheme is that it maintains a compressed model of the observed samples and uses this model to compute the kernel density estimate of the underlying distribution. Since in online operation, the samples arrive continually, a low complexity of the model has to be maintained. We have therefore proposed a compression scheme, which searches for clusters of components and approximates them by a single component. To enable recovering from false compressions, each component is also described by a more detailed mixture model. The approach was analyzed using examples of online estimation of stationary as well as non-stationary distributions and on classification examples. In all experiments, the oKDE was able to produce comparable or better results to the state-of-the-art batch approaches, while producing models whose complexity was significantly lower. While the proposed oKDE is a contribution to the literature on kernel density estimation as such, parts of our approach can contribute to solutions of some other problems as well. The proposed unscented Hellinger distance may be used, for example, as a general metric in applications where one needs to compare mixtures of Gaussians (e.g., [15]). Recently, an approximate probability density estimator was proposed for visual tracking in [18]. The estimator is based on KDE, however, the kernel bandwidth is either predefined by user or an ad hoc scheme which is vulnerable to data order is applied. Our bandwidth selection rule can be directly applied to that estimator to provide means of an automatic and theoretically supported bandwidth selection.

Furthermore, since we maintain a non-parametric model of the data, it would be interesting to consider in the future work standard improvements of the bandwidth selectors such as sample point or baloon density estimator [38], [47]. Another extension might be adaptation of the cross-validation bandwidth selector [30]. While we have seen that the oKDE successfully approximates the nonstationary distributions, the "forgetting factor" was predetermined. An interesting venue of furhter research would be how to automatically determine this factor directly from streaming data.

Note that the update procedure in the oKDE makes it a reconstructive estimator, since the compression algorithm penalizes errors in the reconstruction. We can think about the compression algorithm itself as an approximate optimization, which seeks a minimum of the reconstructive cost function. We believe that replacing this cost function with some other criteria would yield different properties of the online KDE, without modification of the optimization algorithm. Indeed, we have already explored a possibility of replacing this cost function with a criteria that, instead of reconstruction errors, it penalizes discriminative errors in [23] and obtained encouraging preliminary results. We believe that this venue of research will lead to online probabilistic discriminative models based on the kernel density estimation, which will be based on the theory presented here. These are the topics of our ongoing research.

Appendix A
Functional approximation

In this appendix we derive an approximation to the functional $R(p, \mathbf{F})$ from Section III. We write the multiple partial

TABLE VI
AVERAGE CLASSIFICATION RESULTS (RECOG.) AND AVERAGE NUMBER OF COMPONENTS PER CLASS ($\mathrm{N_{cmp}}$) IN THE MODELS ALONG WITH $\pm$ ONE STANDARD DEVIATION. THE NUMBER OF SAMPLES IN EACH DATASET, THE DIMENSIONALITY AND THE NUMBER OF CLASSES ARE DENOTED BY N_{S}, N_{D} AND N_{C}, RESPECTIVELY.

				oKDE		CV		RSDE		Hall		SVM	
dataset	N_{S}	N_{D}	N_{C}	Recog.	$\mathrm{N_{cmp}}$	Recog.	$\mathrm{N_{cmp}}$	Recog.	$\mathrm{N_{cmp}}$	Recog.	$\mathrm{N_{cmp}}$	Recog.	$\mathrm{N_{cmp}}$
Iris	150	4	3	97±3%	31±0	96±3%	38±0	96±2%	10±5	97±4%	38±0	96±3%	16±1
Pima	768	8	2	70±1%	162±3	72±2%	288±0	65±3%	48±10	67±2%	288±0	78±3%	160±4
Wine	178	13	3	99±2%	45±0	92±4%	45±0	94±4%	44±0	99±2%	45±0	98±2%	22±4
Letter	20000	16	26	95±0%	222±5	96±0%	613±0	55±0%	25±0	95±0%	613±0	96±0%	322±0

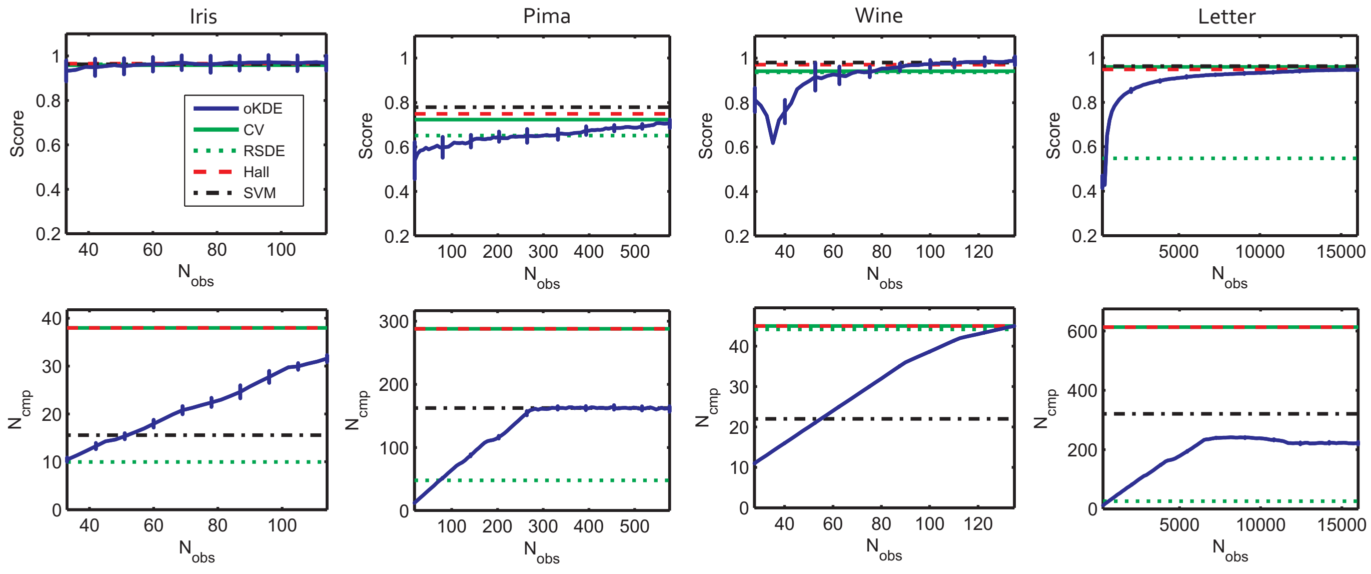


Fig. 16. Upper row shows classification results (Score) and the lower row shows the number of components per class (N_{cmp}) w.r.t. the number of all observed samples (N_{obs}). The results for the oKDE are depicted along with one standard deviation bars. For reference, we also show results for the batch methods after observing all the samples.

derivatives of a d-variate function $g(\mathbf{x})$ as

$$g^{(\mathbf{r})} = \frac{\partial^{|\mathbf{r}|}}{\partial x_1^{r_1} \ldots \partial x_d^{r_d}} g(\mathbf{x}), \tag{39}$$

with $\mathbf{r} = (r_1, \ldots, r_d)$ a vector of nonnegative integers and $|\mathbf{r}| = \sum_{i=1}^{d} r_i$. According to [47] (page, 98) we can rewrite $R(p, \mathbf{F})$ from (7) into

$$\begin{aligned} R(p, \mathbf{F}) &= \int \mathrm{tr}\{\mathbf{F}\mathcal{G}_{\mathrm{p}}(\mathbf{x})\}\mathrm{tr}\{\mathbf{F}\mathcal{G}_{\mathrm{p}}(\mathbf{x})\}d\mathbf{x} \\ &= \mathrm{vech}^T(\mathbf{F})\mathbf{\Psi}_{\mathcal{G}}\mathrm{vech}(\mathbf{F}), \end{aligned} \tag{40}$$

with $\mathbf{\Psi}_{\mathcal{G}}$ denoting a $\frac{1}{2}d(d+1) \times \frac{1}{2}d(d+1)$ matrix

$$\begin{aligned} \mathbf{\Psi}_{\mathcal{G}} = \int & \mathrm{vech}(2\mathcal{G}_p(\mathbf{x}) - \mathrm{dg}(\mathcal{G}_p(\mathbf{x}))) \times \\ & \mathrm{vech}^T(2\mathcal{G}_p(\mathbf{x}) - \mathrm{dg}(\mathcal{G}_p(\mathbf{x})))d\mathbf{x}, \end{aligned} \tag{41}$$

and where the notation dg denotes the diagonal matrix formed by replacing all off-diagonal entries by zeros. Since $\mathcal{G}_p(\mathbf{x})$ is a matrix of second partial derivatives, each entriy in $\mathbf{\Psi}_{\mathcal{G}}$ can be written in terms of functionals $\psi_{\mathbf{r}}$,

$$\begin{aligned} \psi_{\mathbf{r}} &= \int p^{(\mathbf{r}_1)}(\mathbf{x})p^{(\mathbf{r}_2)}(\mathbf{x})d\mathbf{x} \\ &= \int p^{(\mathbf{r})}(\mathbf{x})p(\mathbf{x})d\mathbf{x} \end{aligned} \tag{42}$$

with $\mathbf{r} = \mathbf{r}_1 + \mathbf{r}_2$ and with even $|\mathbf{r_1}|$ and $|\mathbf{r_2}|$. This means that entries in (40) are simply expectations of partial even derivatives of an unknown distribution $p(\mathbf{x})$. We now approximate the unknown distribution by a sample model $p_{\mathrm{s}}(\mathbf{x})$ and approximate its derivative through a kernel density estimate $p_{\mathbf{G}}^{(\mathbf{r})}(\mathbf{x})$ (approximations in equation 8), where

$$\begin{aligned} p_{\mathbf{G}}(\mathbf{x}) &= \sum_{i=1}^{N_{\mathrm{g}}} \alpha_{\mathrm{g}i}\phi_{\Sigma_{\mathrm{g}i}}(\mu_{\mathrm{g}i} - \mathbf{x}) \\ p_{\mathrm{s}}(\mathbf{x}) &= \sum_{j=1}^{N_{\mathrm{s}}} \alpha_{\mathrm{s}j}\phi_{\Sigma_{\mathrm{s}j}}(\mu_{\mathrm{s}j} - \mathbf{x}). \end{aligned}$$

This means that the entries $\psi_{\mathbf{r}}$ are approximated by

$$\begin{aligned} \hat{\psi}_{\mathbf{r}} &= \int p_{\mathbf{G}}^{(\mathbf{r})}(\mathbf{x})p_{\mathrm{s}}(\mathbf{x})d\mathbf{x} \\ &= \int p_{\mathbf{G}}^{(\mathbf{r}_1)}(\mathbf{x})p_s^{(\mathbf{r}_2)}(\mathbf{x})d\mathbf{x}. \end{aligned} \tag{43}$$

By matching the last line of the (43) with the first line of (42) we get the approximation of $\mathbf{\Psi}_{\mathcal{G}}$,

$$\begin{aligned} \hat{\mathbf{\Psi}}_{\mathcal{G}} = \int & \mathrm{vech}(2\mathcal{G}_{p_{\mathbf{G}}}(x) - \mathrm{dg}(\mathcal{G}_{p_{\mathbf{G}}}(x))) \times \\ & \mathrm{vech}^T(2\mathcal{G}_{ps}(x) - \mathrm{dg}(\mathcal{G}_{ps}(x)))d\mathbf{x}. \end{aligned} \tag{44}$$

Plugging $\hat{\mathbf{\Psi}}_{\mathcal{G}}$ back into (40) yields the following approximation of $R(p,\mathbf{F})$:

$$\hat{R}(p,\mathbf{F},\mathbf{G}) = \int \mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_{\mathbf{G}}}(\mathbf{x})\}\mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_s}(\mathbf{x})\}d\mathbf{x}. \quad (45)$$

APPENDIX B
CLOSED-FORM FUNCTIONAL CALCULATION

To derive a closed-form solution to

$$\hat{R}(p,\mathbf{F},\mathbf{G}) = \int \mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_{\mathbf{G}}}(\mathbf{x})\}\mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_s}(\mathbf{x})\} \quad (46)$$

which is based only on matrix algebra, we follow closely the derivation of a similar integral which was studied in the Appendix of M.P. Wand's paper [46]. As in that paper, we will require some established results:

$$\mathcal{G}_{\phi_{\mathbf{\Sigma}}(\cdot-\mu)} = \phi_{\mathbf{\Sigma}}(\mathbf{x})\{\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)(\mathbf{x}-\mu)^T - \mathbf{I}\}\mathbf{\Sigma}^{-1}, \quad (47)$$

$$\phi_{\mathbf{\Sigma}_i}(\mathbf{x}-\mu_i)\phi_{\mathbf{\Sigma}_j}(\mathbf{x}-\mu_j) = \phi_{\mathbf{\Sigma}_i+\mathbf{\Sigma}_j}(\mu_i-\mu_j)\phi_{\mathbf{\Sigma}_i(\mathbf{\Sigma}_i+\mathbf{\Sigma}_j)^{-1}\mathbf{\Sigma}_j}(\mathbf{x}-\mu^*) \quad (48)$$

where

$$\mu^* = \mathbf{\Sigma}_j(\mathbf{\Sigma}_i+\mathbf{\Sigma}_j)^{-1}\mu_i + \mathbf{\Sigma}_i(\mathbf{\Sigma}_i+\mathbf{\Sigma}_j)^{-1}\mu_j, \quad (49)$$

and

$$\mathrm{Cov}(\mathbf{X}^T\mathbf{A}\mathbf{X},(\mathbf{X}-\mathbf{c})^T\mathbf{B}(\mathbf{X}-\mathbf{c})) = 2tr[\mathbf{A}\mathbf{\Sigma}\mathbf{B}\{\mathbf{\Sigma}+2(\mu-\mathbf{c})\mu^T\}], \quad (50)$$

where $\mathbf{X}$ is random vector distributed in $\phi_{\mathbf{\Sigma}}(\mu-\mathbf{x})$, $\mathbf{A}$ and $\mathbf{B}$ are $d\times d$ symmetric constant matrices and $\mathbf{c}$ is a $d\times 1$ constant vector. We start by expanding the integral

$$\int \mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_{\mathbf{G}}}(\mathbf{x})\}\mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_s}(\mathbf{x})\} = \sum_{i=1}^{N_{\mathrm{g}}}\sum_{j=1}^{N_{\mathrm{s}}}\alpha_{\mathrm{g}i}\alpha_{\mathrm{s}j}\phi_{\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j}}(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j}) \times E[\mathrm{tr}\{\mathbf{F}\mathbf{\Sigma}_{\mathrm{g}i}^{-1}[(Y-\mu_{\mathrm{g}i})(Y-\mu_{\mathrm{g}i})^T\mathbf{\Sigma}_{\mathrm{g}i}^{-1}-\mathbf{I}]\} \times\mathrm{tr}\{\mathbf{F}\mathbf{\Sigma}_{\mathrm{s}j}^{-1}[(Y-\mu_{\mathrm{s}j})(Y-\mu_{\mathrm{s}j})^T\mathbf{\Sigma}_{\mathrm{s}j}^{-1}-\mathbf{I}]\}], \quad (51)$$

where $\mathbf{I}$ is an identity matrix and $\mathbf{Y}$ a random vector distributed in $\phi_{\mathbf{\Sigma}_{\mathrm{g}i}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}\mathbf{\Sigma}_{\mathrm{s}j}}(\mathbf{x}-\mu^*_{ij})$, with

$$\mu^*_{ij} = \mathbf{\Sigma}_{\mathrm{s}j}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}\mu_{\mathrm{g}i} + \mathbf{\Sigma}_{\mathrm{g}i}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}\mu_{\mathrm{s}j}. \quad (52)$$

Since $E(\mathbf{UV}) = \mathrm{Cov}(\mathbf{U},\mathbf{V}) + E(\mathbf{U})E(\mathbf{V})$ for two random variables $\mathbf{U}$ and $\mathbf{V}$ the expectation in (51) can be written as

$$\mathrm{Cov}\{(Y-\mu_{\mathrm{g}i})^T\mathbf{\Sigma}_{\mathrm{g}i}^{-1}\mathbf{F}\mathbf{\Sigma}_{\mathrm{g}i}^{-1}(Y-\mu_{\mathrm{g}i}), (Y-\mu_{\mathrm{s}j})^T\mathbf{\Sigma}_{\mathrm{s}j}^{-1}\mathbf{F}\mathbf{\Sigma}_{\mathrm{s}j}^{-1}(Y-\mu_{\mathrm{s}j}) +\mathrm{tr}\{\mathbf{F}\mathbf{\Sigma}_{\mathrm{g}i}^{-1}(E[(Y-\mu_{\mathrm{g}i})(Y-\mu_{\mathrm{g}i})^T]\mathbf{\Sigma}_{\mathrm{g}i}^{-1}-\mathbf{I})\} \times\mathrm{tr}\{\mathbf{F}\mathbf{\Sigma}_{\mathrm{s}j}^{-1}(E[(Y-\mu_{\mathrm{s}j})(Y-\mu_{\mathrm{s}j})^T]\mathbf{\Sigma}_{\mathrm{s}j}^{-1}-\mathbf{I})\}. \quad (53)$$

Since $\mu^*_{ij}-\mu_{\mathrm{s}i} = \mathbf{\Sigma}_{\mathrm{g}i}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}(\mu_{\mathrm{s}j}-\mu_{\mathrm{g}i})$, (50) and matrix algebra can be used to show that the covariance term is

$$2\mathrm{tr}\{\mathbf{F}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}\mathbf{F}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}\times [\mathbf{I}-2(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})^T(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})]\}.$$

Using $E[(Y-\mu_{\mathrm{g}i})(Y-\mu_{\mathrm{g}i})^T] = \mathbf{\Sigma}_{\mathrm{g}i}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}\mathbf{\Sigma}_{\mathrm{s}j} + (\mu^*_{ij}-\mu_{\mathrm{g}i})(\mu^*_{ij}-\mu_{\mathrm{g}i})^T$ it can be shown that each of the factor in the second term is equal to

$$-\mathrm{tr}\{\mathbf{F}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}[\mathbf{I}-(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})^T(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}]\}. \quad (54)$$

Combining these with (51) and applying the definitions

$$\begin{aligned}\mathbf{A}_{ij} &= (\mathbf{\Sigma}_{\mathrm{gi}}+\mathbf{\Sigma}_{\mathrm{sj}})^{-1},\\ \mathbf{B}_{ij} &= \mathbf{A}_{ij}\{\mathbf{I}-2(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})^T\mathbf{A}_{ij}\},\\ \mathbf{C}_{ij} &= \mathbf{A}_{ij}\{\mathbf{I}-(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})^T\mathbf{A}_{ij}\} \quad (55)\end{aligned}$$

leads to the result in (11).

APPENDIX C
THE UNSCENTED HELLINGER DISTANCE

The unscented transform is a special case of a Gaussian quadrature, which, similarly to Monte Carlo integration, relies on evaluating integrals using carefully placed points, called *the sigma points*, over the support of the integral. Therefore, as in Monte Carlo integration [45], we define an *importance* distribution $p_0(\mathbf{x}) = \gamma(p_1(\mathbf{x})+p_2(\mathbf{x}))$, which contains the support of both, $p_1(\mathbf{x})$ as well as $p_2(\mathbf{x})$, with γ set such that $\int p_0(\mathbf{x})d\mathbf{x} = 1$. In our case, $p_0(\mathbf{x})$ is a Gaussian mixture model of a form $p_0(\mathbf{x}) = \sum_{i=1}^N w_i\phi_{\mathbf{\Sigma}_i}(\mathbf{x}-\mathbf{x}_i)$, and we rewrite the Hellinger distance (21) into

$$D^2(p_1,p_2) = \frac{1}{2}\int g(\mathbf{x})p_0(\mathbf{x})d\mathbf{x} = \frac{1}{2}\sum_{i=1}^N w_i\int g(\mathbf{x})\phi_{\Sigma_i}(\mathbf{x}-\mathbf{x}_i)d\mathbf{x}, \quad (56)$$

where we have defined $g(\mathbf{x}) = \frac{(\sqrt{p_1(\mathbf{x})}-\sqrt{p_2(\mathbf{x})})^2}{p_0(\mathbf{x})}$. Note that the integrals in (56) are simply expectations over a nonlinearly transformed Gaussian random variable $\mathbf{X}$, and therefore admit to the unscented transform. According to [22] we then have

$$D^2(p_1,p_2) \approx \frac{1}{2}\sum_{i=1}^N w_i\sum_{j=0}^{2d+1} g({}^{(j)}\mathcal{X}_i){}^{(j)}\mathcal{W}_i, \quad (57)$$

where $\{{}^{(j)}\mathcal{X}_i, {}^{(j)}\mathcal{W}_i\}_{j=0:d}$ are weighted sets of sigma points corresponding to the i-th Gaussian $\phi_{\mathbf{\Sigma}_i}(\mathbf{x}-\mathbf{x}_i)$, and are defined as

$$\begin{aligned}{}^{(0)}\mathcal{X}_i = \mathbf{x}_i\ ;\ {}^{(0)}\mathcal{W}_i = \frac{\kappa}{1+\kappa}\\ {}^{(j)}\mathcal{X}_i = \mathbf{x}_i + s_j\sqrt{1+\kappa}(\sqrt{d\mathbf{\Sigma}_i})_j \quad (58)\end{aligned}$$

$${}^{(j)}\mathcal{W}_i = \frac{\kappa}{2(1+\kappa)}\ ;\ s_j = \begin{cases}1 & ;\quad j\le d\\ -1 & ;\quad otherwise\end{cases} \quad (59)$$

with $\kappa = \max([0,m-d])$, and $(\sqrt{\mathbf{\Sigma}_i})_j$ is the j-th column of the matrix square root of $\mathbf{\Sigma}_i$. Concretely, let $\mathbf{UDU}^T$ be a singular value decomposition of covariance matrix $\mathbf{\Sigma}$, such that $\mathbf{U} = \{U_1,\ldots,U_d\}$ and $\mathbf{D} = \mathrm{diag}\{\lambda_1,\ldots,\lambda_d\}$, then $(\sqrt{\Sigma})_k = \sqrt{\lambda_k}U_k$. In line with the discussion on the properties of the unscented transform in [22], we set the parameter m to $m=3$.

ACKNOWLEDGMENT

This research has been supported in part by: Research program P2-0214 (RS), Research program P2-0095 (RS), TP-MIR SOVZO (Ministry of Defense of Republic of Slovenia), and EU FP7-ICT215181-IP project CogX.

REFERENCES

[1] O. Arandjelovic and R. Cipolla, *Incremental learning of temporally-coherent gaussian mixture models*, British Machine Vision Conference, 2005, pp. 759–768.
[2] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida, *Cognitive developmental robotics: a survey*, IEEE Transactions on Autonomous Mental Development **1** (2009), no. 1, 1234.
[3] A. Asuncion and D.J. Newman, *UCI machine learning repository*, 2007.
[4] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation*, ch. 11, pp. 438–440, John Wiley & Sons, Inc., 2001.
[5] H. Bischof and A. Leonardis, *View-based object representations using rbf networks*, "IVC" **19** (2001), 619–629.
[6] J. R Bunch and C. P Nielsen, *Updating the singular value decomposition*, Numerische Mathematik **31** (1978), no. 2, 111129.
[7] Chih Chung Chang and Chih Jen Lin, *LIBSVM: A library for support vector machines*, 2001.
[8] D. Comaniciu, V. Ramesh, and P. Meer, *The variable bandwidth mean shift and data-driven scale selection*, Proc. Int. Conf. Computer Vision, vol. 1, 2001, pp. 438 – 445.
[9] A. Declercq and J. H. Piater, *Online learning of gaussian mixture models - a two-level approach*, Intl.l Conf. Comp. Vis., Imaging and Comp. Graph. Theory and Applications, 2008, pp. 605–611.
[10] L. Devroye and G. Lugosi, *Combinatorial methods in density estimation*, Springer Verlag, 2001.
[11] T. Duong and M. L. Hazelton, *Plug-in bandwidth matrices for bivariate kernel density estimation*, Nonparametric Statistics **15** (2003), no. 1, 1730.
[12] M. A. F. Figueiredo and A. K. Jain, *Unsupervised learning of finite mixture models*, IEEE Trans. Pattern Anal. Mach. Intell. **24** (2002), no. 3, 381–396.
[13] M. Girolami and C. He, *Probability density estimation from optimally condensed data samples.*, IEEE Trans. Pattern Anal. Mach. Intell. **25** (2003), no. 10, 1253–1264.
[14] J. Goldberger, H. Greenspan, and J. Dreyfuss, *Simplifying mixture models using the unscented transform*, IEEE Transactions on Pattern Analysis and Machine Intelligence (2008), 14961502.
[15] J. Goldberger and S. Roweis, *Hierarchical clustering of a mixture model*, Neural Inf. Proc. Systems, 2005, pp. 505–512.
[16] R. Gomes, M. Welling, and P. Perona, *Incremental learning of nonparametric Bayesian mixture models*, IEEE Conference on Computer Vision and Pattern Recognition, 2008.
[17] P. Hall, S. J. Sheater, M. C. Jones, and J. S. Marron, *On optimal data-based bandwidth selection in kernel density estimation*, Biometrika **78** (1991), no. 2, 263–269.
[18] B. Han, D. Comaniciu, Y. Zhu, and L. S. Davis, *Sequential kernel density approximation and its application to real-time visual tracking*, IEEE Trans. Pattern Anal. Mach. Intell. **30** (2008), no. 7, 1186–1197.
[19] A. T. Ihler, *Inference in sensor networks: Graphical models and particle methods*, Ph.D. thesis, Massachusetts Institute of Technology, 2005.
[20] A. T. Ihler and M. Mandel, *Kernel density estimation toolbox for MATLAB*, http://www.ics.uci.edu/ ihler/code/, 2007.
[21] M. C. Jones, J. S. Marron, and S. J. Sheather, *A brief survey of bandwidth selection for density estimation*, J. Amer. Stat. Assoc. **91** (1996), no. 433, 401–407.
[22] S. Julier and J. Uhlmann, *A general method for approximating nonlinear transformations of probability distributions*, Tech. report, Department of Engineering Science, University of Oxford, 1996.
[23] M. Kristan and A. Leonardis, *Online discriminative kernel density estimation*, International Conference on Pattern Recognition, 2010, p. accepted.
[24] M. Kristan, D. Skočaj, and A. Leonardis, *Online kernel density estimation for interactive learning*, Image and Vision Computing **28** (2010), no. 7, 1106–1116.
[25] F. De la Torre and M. J Black, *Robust principal component analysis for computer vision*, International conference on computer vision, vol. 52, 2001, p. 362369.
[26] A. Leonardis and H. Bischof, *An efficient mdl-based construction of rbf networks*, Neural Networks **11** (1998), no. 5, 963 – 973.
[27] G. J. Mc Lachlan and T. Krishan, *The EM algorithm and extensions*, Wiley, 1997.
[28] G. McLachlan and D. Peel, *Finite mixture models*, Wiley-Interscience, 2000.
[29] A. W. Moore, *Very fast EM-based mixture model clustering using multiresolution kd-trees*, Advances in Neural Information Processing Systems, 1999, p. 543549.
[30] J. M. L. Murillo and A. A. Rodriguez, *Algorithms for gaussian bandwidth selection in kernel density estimators*, Neural Inf. Proc. Systems, 2008.
[31] E. Parzen, *On estimation of a probability density function and mode*, Annals of Math. Statistics **33** (1962), 1065–1076.
[32] M. Pawlak and U. Stadtmuller, *Kernel density estimation with generalized binning*, Scandinavian Journal of Statistics **26** (1999), no. 4, 539561.
[33] D. E. Pollard, *A user's guide to measure theoretic probability*, Cambridge University Press, 2002.
[34] C. E. Priebe and D. J. Marchette, *Adaptive mixture density estimation*, Patt. Recogn. **26** (1993), 771–785.
[35] EU project FP6-004250-IP, *CoSy: Cognitive systems for cognitive assistants*, http://www.cognitivesystems.org, 2004-2008.
[36] EU project FP7-ICT215181-IP, *CogX: Cognitive systems that self-understand and self-extend*, http://cogx.eu, 2008-2012.
[37] J. Rissanen, *Stochastic complexity*, J. Royal Statistical Society **49** (1987), no. 3, 223–239.
[38] D. W. Scott, *Multivariate density estimation*, New York: Wiley, 1992.
[39] D. W Scott and S. R Sain, *Multi-dimensional density estimation*, Data mining and data visualization (2005), 229.
[40] B. W Silverman, *Density estimation*, London: Chapman and Hall, 1986.
[41] D. Skocaj and A. Leonardis, *Incremental and robust learning of subspace representations*, Image and Vision Computing **26** (2008), no. 1, 2738.
[42] M. Song and H. Wang, *Highly efficient incremental estimation of Gaussian mixture models for online data stream clustering*, SPIE: Intelligent Computing: Theory and Applications, 2005, pp. 174–183.
[43] A. Stoytchev, *Some basic principles of developmental robotics*, IEEE Transactions on Autonomous Mental Development **1** (2009), no. 2, 122130.
[44] B. A Turlach, *Bandwidth selection in kernel density estimation: A review*, CORE and Institut de Statistique (1993), 23–493.
[45] E. Veach and L. J. and Guibas, *Optimally combining sampling techniques for monte carlo rendering*, Computer graphics and interactive techniques, 1995, pp. 419 – 428.
[46] M. P. Wand, *Error analysis for general multivariate kernel estimators*, Nonparametric Statistics **2** (1992), 1–15.
[47] M. P. Wand and M. C. Jones, *Kernel smoothing*, Chapman & Hall/CRC, 1995.
[48] T. Xiang and S. Gong, *Model selection for unsupervised learning of visual context*, Int. J. Comput. Vision **69** (2006), no. 2, 181–201.
[49] K. Zhang and J. T. Kwok, *Simplifying mixture models through function approximation*, Neural Inf. Proc. Systems, 2006.
[50] Z. Živkovič and F. van der Heijden, *Recursive unsupervised learning of finite mixture models*, IEEE Trans. Pattern Anal. Mach. Intell. **26** (2004), no. 5, 651 – 656.

Matej Kristan received the Dipl.ing., M.Sc., and Ph.D. degrees in electrical engineering from the Faculty of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia, in 2003, 2005, and 2008, respectively.

He is currently an Assistant Professor at the Machine Vision Laboratory, Faculty of Electrical Engineering, University of Ljubljana and a Researcher with the Visual Cognitive Systems Laboratory, Faculty of Computer and Information Science, University of Ljubljana. His research interests include probabilistic methods for computer vision and pattern recognition with focus on tracking, probabilistic dynamic models, online learning and mobile robotics. Dr. Kristan has received several awards for his research in the field of computer vision and pattern recognition.

Aleš Leonardis received the Dipl.ing. and M.Sc. degrees in electrical engineering and the Ph.D. degree in computer science from the Faculty of Electrical Engineering and Computer Science, University of Ljubljana, Ljubljana, Slovenia, in 1985, 1988, and 1993, respectively.

From 1988 to 1991, he was a Visiting Researcher at the General Robotics and Active Sensory Perception Laboratory, University of Pennsylvania, Philadelphia. From 1995 to 1997, he was a Postdoctoral Associate with the Pattern Recognition and Image Processing Group, Vienna University of Technology, Vienna, Austria. He was also a Visiting Researcher and a Visiting Professor at the Swiss Federal Institute of Technology ETH, Zurich, and at the Technische Fakultaet der Friedrich-Alexander-Universitaet, Erlangen, respectively. He is a Full Professor and the Head of the Visual Cognitive Systems Laboratory, Faculty of Computer and Information Science, University of Ljubljana. He is also an Adjunct Professor with the Faculty of Computer Science, Graz University of Technology, Graz, Austria. He is an author or coauthor of more than 160 papers published in journals and conferences and he coauthored the book Segmentation and Recovery of Superquadrics (Kluwer, 2000). His research interests include robust and adaptive methods for computer vision, object and scene recognition and categorization, statistical visual learning, 3-D object modeling, and biologically motivated visions. Dr. Leonardis is an Editorial Board Member of Pattern Recognition, an Editor of the Springer Book Series Computational Imaging and Vision, and an Associate Editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. He has served on the program committees of major computer-vision and pattern-recognition conferences. He was also a Program Cochair of the European Conference on Computer Vision 2006. He has received several awards. In 2002, he coauthored a paper, Multiple Eigenspaces, which won the 29th Annual Pattern Recognition Society award. In 2004, he was awarded a prestigious national award for scientific achievements. He is a fellow of the International Association for Pattern Recognition and a member of the IEEE Computer Society.

Online Discriminative Kernel Density Estimation

Matej Kristan, Aleš Leonardis
Faculty of Computer and Information Science
University of Ljubljana
Ljubljana, Slovenia
{matej.kristan},{ales.leonardis}@fri.uni-lj.si

***Abstract*—We propose a new method for online estimation of probabilistic discriminative models. The method is based on the recently proposed online Kernel Density Estimation (oKDE) framework which produces Gaussian mixture models and allows adaptation using only a single data point at a time. The oKDE builds reconstructive models from the data, and we extend it to take into account the interclass discrimination through a new distance function between the classifiers. We arrive at an online discriminative Kernel Density Estimator (odKDE). We compare the odKDE to oKDE, batch state-of-the-art KDEs and support vector machine (SVM) on a standard database. The odKDE achieves comparable classification performance to that of best batch KDEs and SVM, while allowing online adaptation, and produces models of lower complexity than the oKDE.**

***Index Terms*—Online Estimation; Discriminative Models; Kernel Density Estimation**

I. Introduction

Building discriminative models of some process from the observed data is a central task of many applications in machine learning. A popular approach to generating models is to estimate the probability density function (pdf) associated with the observed data. In this respect, *reconstructive* models such as the Gaussian mixture models, (GMM), (e.g., [1]) have been successfully applied in *batch* operation, i.e., in situations in which all the data is observed in advance. In contrast to the reconstructive models, the discriminative models capitalize on the discriminative information, however, this may lead to decreased robustness [2]. A significant drawback of the purely batch methods is that their estimation becomes increasingly difficult when processing extremely large amounts of data. Furthermore, in real-world environments, all the data may not be available in advance, or we even want to observe some process for an indefinite duration, while continually providing the best estimate of the model from the data observed so far. This generates the need for models that can be constructed in an *online* operation.

Adapting the existing reconstructive GMM methods to work with online cases, in which as little as a single data-point may be observed at a time, is a nontrivial task. In contrast to the batch incremental models (e.g., [3]) who store and revisit all the data in multiple passes, the online models have to adapt from a (single) new data-point and then discard that data-point. The main difficulty is therefore that the online models have to maintain sufficient information to generalize well to the yet unobserved data and have to adjust their complexity without having access to all the observations (future as well as past). There have been various attempts to extend the *reconstructive* GMMs to online operation, however, these either imply strong spatio-temporal constraints on the data [4], [5], assume constraints on the shape of the target distribution [6] or require tuning of parameters to a specific application [7]. Recently, we have proposed a non-parametric approach called the *online Kernel Density Estimation* (oKDE) [8]. In contrast to the other approaches, the oKDE does not impose any of the above constraints but assumes only that the target pdf is sufficiently smooth and produces models with a high reconstructive performance. In [9] we have also considered a variant of the oKDE that allows adaptation from positive as well as negative examples.

While the purely reconstructive models may contain redundant information required for discrimination, the discriminative models disregard the reconstructive information required for online adaptation. Indeed, Fidler et al. [2] have shown that even in batch methods accounting for the reconstructive information leads to improved robustness of the discriminative models. Following their results, we adapt the oKDE framework to account for the discriminative power of the models along with the reconstructive, thus arriving at an *online discriminative Kernel Density Estimator*, which is the main contribution of the paper. The proposed method allows online adaptation of the discriminative models, by maintaining enough reconstructive power to efficiently adapt to new observations, and can be used to develop online classifiers. The remainder of the paper is structured as follows. In Section II we briefly review the oKDE framework, in Section III we extend the oKDE to discriminative models, in Section IV we evaluate the approach and the Section V concludes the paper.

II. The online Kernel Density Estimation

The online Kernel Density Estimation (oKDE) produces a generative model from the d-dimensional streaming data as an N-component Gaussian mixture model

$$p(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi_{\mathbf{\Sigma}_i}(\mathbf{x} - \mathbf{x}_i), \tag{1}$$

where $\phi_{\mathbf{\Sigma}}(\mathbf{x} - \mu) = (2\pi)^{-\frac{d}{2}} |\mathbf{\Sigma}|^{-\frac{1}{2}} e^{(-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu))}$ is a Gaussian kernel centered at μ with covariance matrix $\mathbf{\Sigma}$. We give here only a brief overview of the oKDE framework and refer the reader to [8] for more details.

Broadly speaking, the oKDE proceeds in two steps:

Update: Starting with the GMM from the previous time-step $p_{t-1}(\mathbf{x})$ and the new observation $\mathbf{x}_t$, the oKDE augments the $p_{t-1}(\mathbf{x})$ by a Gaussian kernel $\phi_{\mathbf{\Sigma}_t}(\mathbf{x}-\mathbf{x}_t)$ centered at $\mathbf{x}_t$, it automatically calculates the optimal covariance $\mathbf{\Sigma}_t$ for that kernel and accordingly readjusts the covariances of the existing kernels using the multivariate online plug-in rule proposed in [8]. If required, it also refines the GMM by splitting up some of the components.

Compress: To maintain a low complexity (i.e., low number of components) the GMM is simplified from time to time. The oKDE generates a binary tree among the components and iteratively merges pairs of components until some threshold of a cost function is exceeded. In [8], the cost function represents the distance between the distribution before and after the compression, which penalizes the reconstruction errors.

III. Discriminative KDE

We pose the online discriminative learning as a task to estimate from a stream of data a set of K discriminative classes, each class c_i described by a Gaussian mixture model $p(\mathbf{x}|c_i)$ and a prior probability $p(c_i)$. In principle, we could use the oKDE to construct each of these classes, but, due to its reconstructive nature, the produced models will be likely redundant for classification. Indeed, we require the models to contain just so much of the information to prevent a degraded classification. Recall that the oKDE simplifies the models under a certain cost function which measures the reconstruction error induced by compression [8]. This means that by redefining this cost function to rather take into account the classification error, the compression step in the oKDE will lead to models that *reduce their complexity* while *retaining their discriminative power*.

Assume that we want to compress the c_i-th class mixture model $p(\mathbf{x}|c_i)$ into $p_{\mathrm{cmp}}(c_i|\mathbf{x})$, while minimizing the induced classification errors. First we have to rewrite this model into a *classification model*. We consider the class c_i as a *positive example* class C^+, described by a mixture model $p(C^+|\mathbf{x}) = p(c_i|\mathbf{x})$. Then we collect *all the other classes* to form a single *negative example* class C^-, $p(C^-|\mathbf{x}) = \sum_{j\setminus i} p(c_j|\mathbf{x})$. The posterior over the resulting two-class model is then defined as

$$p(C|\mathbf{x}) = \delta_{C^+}(C)p(C^+|\mathbf{x}) + \delta_{C^-}(C)p(C^-|\mathbf{x}), \qquad (2)$$

where $\delta_{C^*}(C)$ is a Dirac function centered at C^*. The compressed counterpart of the posterior (2), is obtained by setting $p_{\mathrm{cmp}}(C^+|\mathbf{x}) = p_{\mathrm{cmp}}(c_i|\mathbf{x})$:

$$p_{\mathrm{cmp}}(C|\mathbf{x}) = \delta_{C^+}(C)p_{\mathrm{cmp}}(C^+|\mathbf{x}) + \delta_{C^-}(C)p(C^-|\mathbf{x}). \qquad (3)$$

From the classification point of view we can say that $p(\mathbf{x}|c_i)$ can be compressed into $p_{\mathrm{cmp}}(\mathbf{x}|c_i)$ as long as the distance between the corresponding posteriors $p(C|\mathbf{x})$ and $p_{\mathrm{cmp}}(C|\mathbf{x})$, does not change significantly. We therefore require a distance measure between the posterior before and after compression.

A. Distance between two classifiers

We define the distance between the posterior $p(C|\mathbf{x})$ and its compression $p_{\mathrm{cmp}}(C|\mathbf{x})$, given some value of $\mathbf{x}$, using the Hellinger distance [10],

$$\begin{aligned} D^2(p, p_{\mathrm{cmp}}|\mathbf{x}) &\overset{\Delta}{=} \frac{1}{2}\int_C (p(C|\mathbf{x})^{\frac{1}{2}} - p_{\mathrm{cmp}}(C|\mathbf{x})^{\frac{1}{2}})^2 \\ &= \frac{1}{2}\sum_{C\in[C^+,C^-]} (p(C|\mathbf{x})^{\frac{1}{2}} - p_{\mathrm{cmp}}(C|\mathbf{x})^{\frac{1}{2}})^2. \qquad (4)\end{aligned}$$

Integrating (4) over the relevant feature space $\mathbf{x}$ gives the *expected* Hellinger distance

$$\hat{D}^2(p, p_{\mathrm{cmp}}) = \int D^2(p, p_{\mathrm{cmp}}|\mathbf{x})p_0(\mathbf{x})d\mathbf{x}, \qquad (5)$$

where the expectation is calculated over the distribution $p_0(\mathbf{x}) = p(\mathbf{x}|C^+)p(C^+) + p(\mathbf{x}|C^-)p(C^-)$, with the priors $P(C^+) = p(c_i)$ and $P(C^-) = \sum_{j\setminus i} p(c_j)$. In our case, $p_0(\mathbf{x})$ can be written in the form of a Gaussian mixture model $p_0(\mathbf{x}) = \sum_{i=1}^M w_i\phi_{\mathbf{\Sigma}_i}(\mathbf{x}-\mathbf{x}_i)$ and (5) becomes

$$\hat{D}^2(p, p_{\mathrm{cmp}}) = \sum_{i=1}^M w_i \int D^2(p, p_{\mathrm{cmp}}|\mathbf{x})\phi_{\Sigma_i}(\mathbf{x}-\mathbf{x}_i)d\mathbf{x}. \quad (6)$$

Note that while $D(\cdot,\cdot|\mathbf{x})$ is a metric, constrained to the interval $[0, 1]$, it is a nonlinear function of $\mathbf{x}$, and the integrals in (6) cannot be evaluated analytically. However, they can be numerically approximated using the *unscented transform*, which has been proposed by [11] for calculating nonlinear transformations of Gaussian variables. Similarly to a Monte Carlo integration, the *unscented transform* relies on evaluating integrals using carefully placed points, called *the sigma points*, over the support of the integral. Therefore, (6) is approximated as

$$\hat{D}^2(p, p_{\mathrm{cmp}}) \approx \sum_{i=1}^M w_i \sum_{j=0}^{2d+1} D^2(p, p_{\mathrm{cmp}}|^{(j)}\mathcal{X}_i)^{(j)}\mathcal{W}_i, \qquad (7)$$

where $\{^{(j)}\mathcal{X}_i, {}^{(j)}\mathcal{W}_i\}_{j=0:d}$ are the weighted sets of sigma points corresponding to the i-th Gaussian $\phi_{\mathbf{\Sigma}_i}(\mathbf{x}-\mathbf{x}_i)$, and are defined as

$$\begin{aligned} {}^{(0)}\mathcal{X}_i &= \mathbf{x}_i \ ; \ {}^{(0)}\mathcal{W}_i = \frac{\kappa}{1+\kappa} \qquad (8)\\ {}^{(j)}\mathcal{X}_i &= \mathbf{x}_i + s_j\sqrt{1+\kappa}(\sqrt{d\mathbf{\Sigma}_i})_j \ ; \\ {}^{(j)}\mathcal{W}_i &= \frac{\kappa}{2(1+\kappa)} \ ; \ s_j = \begin{cases} 1 & ; \quad j \le d \\ -1 & ; \quad otherwise \end{cases}\end{aligned}$$

with $\kappa = \max([0, m-d])$, and $(\sqrt{\mathbf{\Sigma}_i})_j$ is the j-th column of the matrix square root of $\mathbf{\Sigma}_i$. Specifically, let $\mathbf{UDU}^T$ be a singular value decomposition of covariance matrix $\mathbf{\Sigma}$, such that $\mathbf{U} = \{U_1, \ldots, U_d\}$ and $\mathbf{D} = \mathrm{diag}\{\lambda_1, \ldots, \lambda_d\}$, then $(\sqrt{\Sigma})_k = \sqrt{\lambda_k}U_k$. In line with the discussion on the properties of the unscented transform in [11], we set the parameter m to $m = 3$.

B. The online discriminative KDE

The distance function $\hat{D}(p, p_{\text{cmp}})$ from Section III-A penalizes any change of the classification posterior which is induced through the compression of the mixture models. Small values, i.e., $\hat{D}(p, p_{\text{cmp}}) \approx 0$, mean that the classification does not change, while $\hat{D}(p, p_{\text{cmp}}) = 1$ implies a complete change. In an online operation, we can therefore simply adapt the oKDE (Section II) to build and compress a mixture model for each class separately from the observed data. To penalize the loss of discrimination during the compression, we can use the distance measure $\hat{D}(p, p_{\text{cmp}})$ as the compression cost function. We arrive at an online discriminative KDE (odKDE). In practice, we let the odKDE compress the model until the cost function exceeds some (small) threshold $\hat{D}_{\text{th}}$.

During learning, at time-step $t - 1$, we have a set of K models $\{p_{t-1}(\mathbf{x}|c_i), p_{t-1}(c_i)\}_{i=1:K}$. For simplicity assume that at time-step t, K new observations $\{\mathbf{z}_i\}_{i=1:K}$, one per each class[1], arrive and the models are updated into $\{p_t(\mathbf{x}|c_i), p_t(c_i)\}_{i=1:K}$. A single time-step iteration of the approach is outlined in Algorithm 1. In the classification phase a new observation $\mathbf{z}$ is classified into a class $\hat{c}$ by applying the Bayesian rule

$$\hat{c} = \arg\max_{c_i} p(\mathbf{z}|c_i)p(c_i). \tag{9}$$

```
Algorithm 1 : The online discriminative KDE
Input:
    $\{p_{t-1}(\mathbf{x}|c_i), p_{t-1}(c_i)\}_{i=1:K}$ ... the input models.
    $\{\mathbf{z}_i\}_{i=1:K}$ ... observations (one per each class)
Output:
    $\{p_t(\mathbf{x}|c_i), p_t(c_i)\}_{i=1:K}$ ... the output models.
Procedure:
1: for $i = 1 : K$ do
2:     Update $p_{t-1}(\mathbf{x}|c_i)$ with $\mathbf{z}_i$ into $\tilde{p}_t(\mathbf{x}|c_i)$ using the
       original update step of oKDE (Section II).
3:     Update the prior $p_t(c_i)$.
4: end for
5: for $i = 1 : K$ do
6:     Construct the two-class classification model (2) by
       treating $\tilde{p}_t(\mathbf{x}|c_i)$ as a positive example and the rest $K-1$
       models as the negative example.
7:     Compress $\tilde{p}_t(\mathbf{x}|c_i)$ into $p_t(\mathbf{x}|c_i)$ by hierarchical merg-
       ing components such that $\hat{D}(\tilde{p}_t, p_t) \leq \hat{D}_{\text{th}}$.
8: end for
```

IV. Experiments

We have compared the classification performance of the odKDE with the online reconstructive KDE, oKDE [8], and three state-of-the-art batch KDEs: the cross-validation (CV) KDE [12], the reduced-set density estimator [13] (RSDE) initialized by the CV, and the Hall's KDE [14] (Hall). For the baseline classification, we have applied a multiclass support vector machine (SVM) with an RBF kernel [15]. The methods were compared on a set of public classification problems [16] (Table I). In all experiments, the distance parameter in the odKDE was set to $\hat{D}_{\text{th}} = 0.005$.

The odKDE and oKDE were initialized for each class using the first 10 samples and the rest were added one at a time; this experiment was repeated via four-fold cross validation and for three random data orderings. This amounted to twelve repetitions per dataset. The parameter for the SVM kernel was determined separately in each experiment via cross validation on the training dataset. Table I shows the classification score and the number of components in the models after observing all the samples, while Fig. 1 shows the evolution of the results with respect to (w.r.t.) the number of observations for the oKDE and the doKDE. For reference, the graphs also show results for the batch methods after observing all the samples. From the results in Table I we see that the odKDE generated models with comparable classification performance as the oKDE, but generally with a significantly lower complexity. We can verify that this was also true during the online estmation from Fig. 1. Although the odKDE was learnt only by observing a *single* example at a time, the resulting models exhibit classification performance similar to the best batch KDE approaches and the SVM, who optimized their structure having access to *all* the data. By further inspection of the results we can also observe a general trend that the number of components initially significantly increases in doKDE (as well as in oKDE), but then stabilizes for larger number of samples, while the recognition score further improves. For example, in the case of the *Letter* dataset, the bound on the complexity is reached relatively early on after observing 200 samples per class (i.e., after 5200th sample), while the classification performance further increased through the model refinement. This makes the odKDE a very appropriate tool for online operation since it produces compressed models with good classification performance, while at the same time maintains sufficient reconstructive information to allow online refinements of the models from new observations.

V. Conclusion

We have proposed an approach for online estimation of discriminative models by adapting the framework of online Kernel Density Estimation. We have defined a distance measure which measures the discrimination of models and used this measure in the oKDE as a cost function for compression. Results demonstrate that the proposed odKDE produces comparable classification performance to the state-of-the-art, and produces models of significantly lower complexity while allowing online adaptation. This makes the approach ideal for online estimation of classifiers from streaming data. In our future work we will study how different distance measures and data orderings influence the performance of the proposed method as well as test how the method handles noise in labels.

[1]This restriction serves only for clarity of the presentation. Note that, in general, our approach can also handle cases in which observations come only from a subset of classes at a time.

TABLE I

AVERAGE CLASSIFICATION RESULTS ALONG WITH $\pm$ ONE STANDARD DEVIATION. THE NUMBER OF SAMPLES IN EACH DATASET, THE DIMENSIONALITY AND THE NUMBER OF CLASSES ARE DENOTED BY N_{S}, N_{D} AND N_{C}, RESPECTIVELY.

				Recognition accuracy[%] (Number of components per class)					
dataset	N_{S}	N_{D}	N_{C}	odKDE	oKDE	CV	RSDE	Hall	SVM
Iris	150	4	3	97±4%(6.3±1)	97±3%(31±0)	96±3%(38±0)	96±2%(10±5)	97±4%(38±0)	96±3%(16±1)
Pima	768	8	2	71±3%(108±6)	70±1%(162±3)	72±2%(288±0)	65±3%(48±10)	67±2%(288±0)	78±3%(160±4)
Wine	178	13	3	97±2%(2.5±1)	99±2%(45±0)	92±4%(45±0)	94±4%(44±0)	99±2%(45±0)	98±2%(22±4)
Letter	20000	16	26	94±0%(16±1)	95±0%(222±2)	96±0%(613±0)	55±0%(25±0)	95±0%(613±0)	96±0%(322±0)

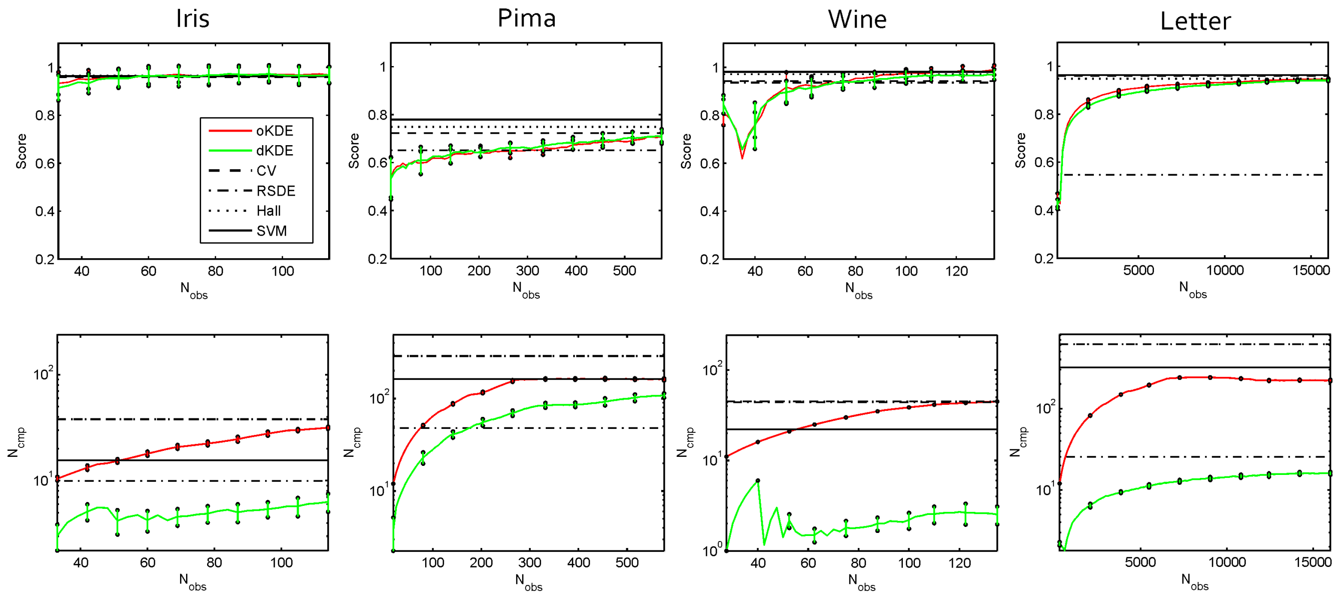


Fig. 1. Upper row shows the classification results (Score) and the lower shows the number of components per class (N_{cmp}) w.r.t. the number of samples (N_{obs}). The results for the oKDE and odKDE are depicted by darker (red) line and bright (green) line, respectively along with one standard deviation bars. For reference, we also show results for the batch methods after observing all samples.

ACKNOWLEDGMENT

This research was supported by: RP P2-0214 and P2-0094 (RS), ARRS project "Learning a large number of visual object categories for content-based retrieval in image and video databases", and EU FP7-ICT215181-IP project CogX.

REFERENCES

[1] M. A. F. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 381–396, 2002.

[2] S. Fidler, D. Skočaj, and A. Leonardis, "Combining reconstructive and discriminative subspace methods for robust classification and regression by subsampling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 3, pp. 337–350, March 2006. [Online]. Available: http://vicos.fri.uni-lj.si/data/publications/fidlerPAMI06.pdf

[3] D. Mansjur and B. Juang, "Incremental learning of mixture models for simultaneous estimation of class distribution and inter-class decision boundaries," in *International Conference on Pattern Recognition*, 2008, pp. 1–4.

[4] O. Arandjelovic and R. Cipolla, "Incremental learning of temporally-coherent gaussian mixture models," in *British Machine Vision Conference*, 2005, pp. 759–768.

[5] M. Song and H. Wang, "Highly efficient incremental estimation of Gaussian mixture models for online data stream clustering," in *SPIE: Intelligent Computing: Theory and Applications*, 2005, pp. 174–183.

[6] B. Han, D. Comaniciu, Y. Zhu, and L. S. Davis, "Sequential kernel density approximation and its application to real-time visual tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1186–1197, 2008.

[7] W. F. Szewczyk, "Time-evolving adaptive mixtures," National Security Agency, Tech. Rep., 2005.

[8] M. Kristan and A. Leonardis, "Multivariate online kernel density estimation," in *Computer Vision Winter Workshop*, 2010, pp. 77–84. [Online]. Available: http://vicos.fri.uni-lj.si/data/publications/KristanCVWW2010.pdf

[9] M. Kristan, D. Skočaj, and A. Leonardis, "Online kernel density estimation for interactive learning," *Image and Vision Computing*, vol. 28, no. 7, pp. 1106–1116, 2010.

[10] D. E. Pollard, *A user's guide to measure theoretic probability*. Cambridge University Press, 2002.

[11] S. Julier and J. Uhlmann, "A general method for approximating nonlinear transformations of probability distributions," Department of Engineering Science, University of Oxford, Tech. Rep., 1996.

[12] J. M. L. Murillo and A. A. Rodriguez, "Algorithms for gaussian bandwidth selection in kernel density estimators," in *Neural Inf. Proc. Systems*, 2008.

[13] M. Girolami and C. He, "Probability density estimation from optimally condensed data samples." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 10, pp. 1253–1264, 2003. [Online]. Available: http://dblp.uni-trier.de/db/journals/IEEEPAMI/IEEEPAMI25.html

[14] P. Hall, S. J. Sheater, M. C. Jones, and J. S. Marron, "On optimal data-based bandwidth selection in kernel density estimation," *Biometrika*, vol. 78, no. 2, pp. 263–269, 1991.

[15] C. C. Chang and C. J. Lin, *LIBSVM: A library for support vector machines*, 2001. [Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/libsvm

[16] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html

Self-Supervised Cross-Modal Online Learning of Basic Object Affordances for Developmental Robotic Systems

Barry Ridge, Danijel Skočaj, and Aleš Leonardis
Faculty of Computer and Information Science
University of Ljubljana, Slovenia
{barry.ridge, danijel.skocaj, ales.leonardis}@fri.uni-lj.si

***Abstract*— For a developmental robotic system to function successfully in the real world, it is important that it be able to form its own internal representations of affordance classes based on observable regularities in sensory data. Usually successful classifiers are built using labeled training data, but it is not always realistic to assume that labels are available in a developmental robotics setting. There does, however, exist an advantage in this setting that can help circumvent the absence of labels: co-occurrence of correlated data across separate sensory modalities over time. The main contribution of this paper is an online classifier training algorithm based on Kohonen's learning vector quantization (LVQ) that, by taking advantage of this co-occurrence information, does not require labels during training, either dynamically generated or otherwise. We evaluate the algorithm in experiments involving a robotic arm that interacts with various household objects on a table surface where camera systems extract features for two separate visual modalities. It is shown to improve its ability to classify the affordances of novel objects over time, coming close to the performance of equivalent fully-supervised algorithms.**

I. Introduction

The term *affordance*, introduced by Gibson [1], is used to characterise the action possibilities that an environment offers an agent acting within that environment. In this paper we address the issue of object affordance learning in a developmental robotic system by developing a self-supervised classifier that operates across two different sensory modalities mediated by object interactions. The main idea behind this is illustrated in Fig. 1 and Fig. 2(a). In our scenario, a robotic arm is mounted on a table surface while camera systems observe the scene. Objects are placed in the workspace where the arm is allowed to interact with them using pushing actions. Object features (e.g. shape features) derived from image data taken prior to arm-object interaction provide data for the first sensory modality, hereby referred to as the *input modality*. After an action has been initiated on an object, video footage is recorded of the object in motion and effect features are extracted from the video footage, forming the basis of the *output modality*. Often when different sensory modalities (or stimulus modalities) are discussed in the literature, they tend to be modalities from different sensory systems, e.g. auditory and visual. Here, instead, we consider two different sensory modalities

This research has been supported by: EU FP6 project VISIONTRAIN (MRTN-CT-2004-005439), EU FP7 project CogX (ICT-215181), and Research program P2-0214 Computer Vision (Republic of Slovenia).

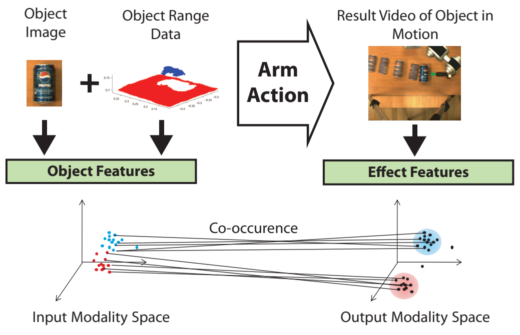
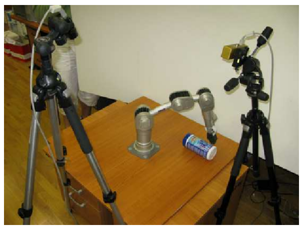


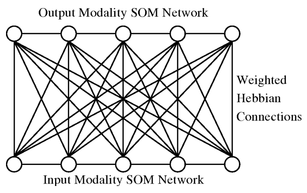
Fig. 1. The main idea of our affordance learning framework.

from the same sensory system (visual) dealing with shape and motion respectively. Though there is a temporal delay when gathering data from each modality during an interactive episode, for the purposes of our discussion here we consider such data in each modality to be *co-occurrences*. Given a series of interactive episodes, the learning task is to find clusters in the output modality feature space that may be identified as affordance classes, and use them to train a classifier in the input modality space. Thus, when the system encounters novel objects, it can predict their affordance classes by observing their respective object features.

The main contribution of our algorithm is that it removes the need for class labels of any kind during the training stage by introducing a probabilistic heuristic based on the co-occurrence information. When designing the algorithm, we were subscribing to an online learning paradigm suitable for developmental robotic systems. Requirements for such an algorithm include: 1) No, or limited, access to previously viewed training samples. 2) An incremental training mechanism. 3) Fixed, or limited, memory requirements. To meet these criteria, we use a cross-modal neural network, as in Fig. 2(b) consisting of two layers of codebook vectors fully connected via a Hebbian weight mapping. The codebook vector layers are of a fixed size, thus meeting the third online learning criterion. The learning algorithm that we present, through the use of Kohonen's *self-organizing map (SOM)* [2] method, as well as a variation of Kohonen's *learning vector quantization (LVQ)* [2], does not require access to previously viewed training samples and can be trained incrementally, thus satisfying the first and second criteria.



(a)



(b)

Fig. 2. (a) Experimental setup. (b) Our cross-modal neural network. Two SOM networks operating in separate modalities, fully connected via a weighted Hebbian mapping.

Perhaps the most closely related work in the literature with respect to affordance learning to our work is by Fitzpatrick *et al.* [3], [4]. The authors trained a humanoid robot to recognize "rolling" affordances of 4 household objects using a fixed set of actions to poke the objects in different directions as well as simple visual descriptors for object recognition. There are two main differences between their method and ours. Firstly, in [3], [4], the feature associated with the rolling direction affordance was pre-determined, whereas in our system, the learning algorithm is provided with a number of different output features and it must determine for itself the affordance classes within that feature space. Secondly, their system used object recognition to identify the affordances of individual objects, whereas our system determines the affordance class of objects (grounded in output modality features) based, not on their individual identity, but on a broad set of input features (e.g. shape). In [5], the authors used a humanoid robot to push objects on a table and used a Bayesian network to form associations between actions, objects and effects. Though quite similar to our approach, their learning method may not be as amenable to full online learning, as they have to gather a certain amount of data initially to form categories within the various modalities before the network can be trained.

Saxena *et al.* [6], [7] used the same robotic arm used in our work (see Sec. III-A) to attempt to grasp novel objects based on a probabilistic model trained on synthetic images of various other objects labeled with grasp points. What differs in our work is that rather than training our learning algorithm on synthetically generated object examples, we train on interactions with real objects. Moreover, their two possible affordances were specified in advance: graspable or non-graspable, whereas our system generates its own affordance classes through interaction with objects. In [8], the authors describe a mobile robotic system equipped with a 3D laser scanner that learns to perceive traversability affordances of various objects, such as spheres, cylinders and boxes in a room. The robot was provided with a set of seven possible actions and used its range scanner to gather angle and distance features aggregated over a grid-division of the range image. It then learned mappings between environmental situations and the results of its actions by first selecting relevant features from the full set, then using support vector machines to classify the relevant features into affordance categories. Though good results were acheived, the affordance categories were, again, pre-defined: traversable or non-traversable.

With regard to our learning algorithm, one of the first examples of Hebbian-linked SOMs was provided in [9], where they were used for developing an artificial neural network model of the mental lexicon. The structure of the network in [9] is identical to the one presented here: two SOMs fully connected via weighted Hebbian mappings. Moreover, the training scheme presented in [9] is the same as our phase 1 training (see Sec. II-B). However, this training scheme by itself, is not optimized for classification purposes as we shall see later in Sec. V, and the SOMs do not influence each other during training. de Sa *et* al. [10], [11] greatly improved upon this by creating a cross-modal neural network where two competitive learning maps in each modality influenced each others' training by learning to agree upon common class labels for co-occurring data samples. Similarly to us, they employed LVQ to train the maps in each modality based on the class information. One drawback, however, is that the class labels have to be determined a priori and maintained throughout the training process. A SOM with a Hebbian learning mechanism called a *Growing When Required (GWR)* network was used in [12] to aid a simulated mobile robot in learning affordances of objects with survival values such as nutrition and stamina so that it could prosper over time in its environment. The SOM was used to cluster visual sensor data in the input space where nodes were assigned weights based on the success or failure of actions. While our method also uses SOM training, in our case it is used on both the output data, where the nodes are meta-clustered to form affordance classes, and the input data, where at a certain point it is swapped for a variation of LVQ which is better suited for classification optimization.

II. The Learning Algorithm

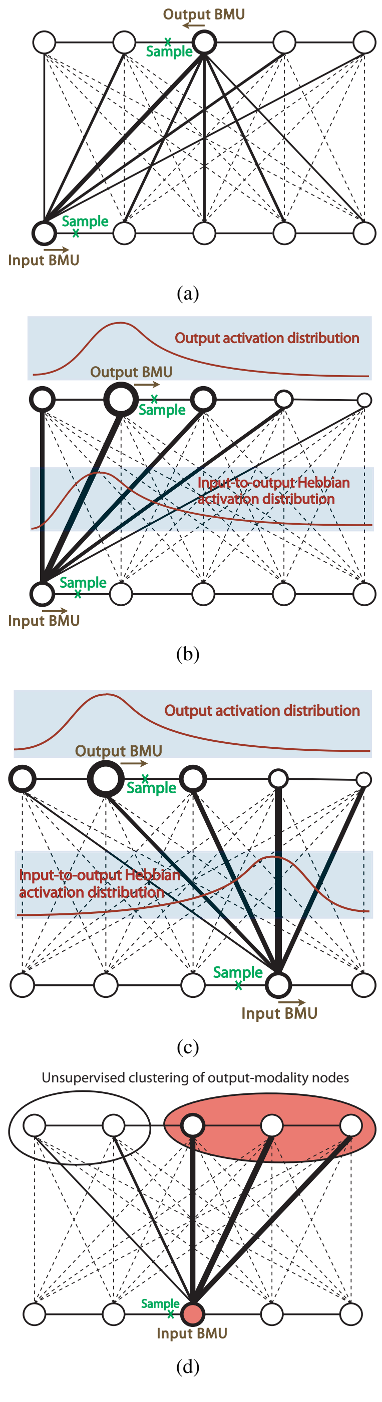
A classifier could be constructed in either of these modalities by attaching class labels to the training data and employing a supervised learning algorithm, but this is hardly ideal for an autonomous cognitive system like a robot since it assumes the existence of an external tutor who is willing to label the data. However, once we have noted that in this type of learning scenario correlated training data *co-occur* in each modality, this opens up some alternative possibilities. For example, k-means clustering or density estimation might be possible in the joint feature space, however, as was discussed in [11], these are not ideal solutions. The problem with simple k-means clustering or competitive learning in the joint space is that all feature dimensions would be required for the classification of test samples; these methods would not be able to marginalize over the missing dimensions when trying to predict the outcome of one modality from another. Density modeling would account for this problem, but requires fitting many parameters which would become infeasible in high dimensions. Moreover, neither approach complies naturally to our online learning criteria. Thus, perhaps a better approach would be to use the natural structure of the data in one modality, as well as the co-occurrence information, to train a supervised classifier in the other modality. This could be accomplished, for example, by using an unsupervised clustering algorithm like k-means to derive clusters in one modality which could be used as class labels to train a supervised classifier in the other ([11] provides a similar approach). However, it is computationally expensive to cluster at every training step in an online algorithm. One alternative, that of clustering early and maintaining the clusters over time, could potentially introduce inaccuracies as training progresses if the sample distribution changes significantly. In the following we describe a cross-modal neural network and a two-phased training scheme that aims to address these issues.

A. Cross-modal Neural Network

The structure of our cross-modal neural network is illustrated in Fig. 3 (a). Two codebook vector layers, one in the input modality and one in the output modality, are fully connected to each other via a weighted Hebbian mapping. The idea is that the codebook vector layers are trained to form a representation of the information contained in their respective modalities, while the Hebbian mapping is trained on the basis of the co-occurrence of data across these modalities.

Learning proceeds in two phases. In the first phase of training, as training samples for each modality are concurrently presented to the network, the codebook vector layers are trained separately using the usual SOM algorithm (described below). While training is ongoing, the Hebbian links that connect the best-matching unit nodes in each of the codebook vector layers are then updated appropriately based on co-occurence. Though SOM training is good for producing low-dimensional represenations of data distributions, it is not the best solution for optimizing decision borders, thus we employ a second phase of training that exploits the co-occurence information captured by the Hebbian mapping between the codebook vector layers. In the second phase, the codebook vector layer in the output modality continues to be trained in the usual way, as does the Hebbian mapping, while the codebook vector layer in the input modality is trained using our variation of LVQ. Rather than using class labels, the LVQ training rules are selected using a Hellinger distance-based heuristic that exploits the cross-modal Hebbian mapping to indicate whether a given codebook vector is of the "correct" or "incorrect" class for a given training sample.

A classifier can then be formed after training by performing unsupervised meta-clustering over the output modality nodes in order to form class labels, although it should be emphasised that these class labels are not required during training. It should also be noted that the algorithm, in largely unmodified form, could also perform regression, though results for this are not presented in this paper. The training and classification processes are described in more detail in the following sections.



Output BMU
Sample
Sample
Input BMU
(a)
Output activation distribution
Output BMU
Sample
Input-to-output Hebbian activation distribution
Sample
Input BMU
(b)
Output activation distribution
Output BMU
Sample
Input-to-output Hebbian activation distribution
Sample
Input BMU
(c)
Unsupervised clustering of output-modality nodes
Sample
Input BMU
(d)

Fig. 3. Training and classification. In each figure, the bottom layer is the input modality codebook & the top layer is the output modality codebook. Arrows indicate node movement during training. (a) Phase 1 training: regular SOM training performed in each modality while Hebbian links are also updated. (b) Phase 2 training: Hellinger distance heuristic indicates correct training sample classification based on activation distributions. Input best-matching unit node (BMU) is moved towards the training sample. (c) Phase 2 training: heuristic indicates incorrect sample classification because the activation distributions differ significantly. Input BMU is moved away from the training sample. (d) Classfication: output nodes are meta-clustered and the cluster with the strongest weighted connections to the input BMU wins.

B. Training: Phase 1

The following two sub-sections describe how both the individual modality networks and the Hebbian mapping that connects them are trained in the first phase of training. Phase 1 training is illustrated in Fig. 3 (a).

1) Modality Codebook Vector Layers: In the first phase, network training in each modality proceeds in accordance with Kohonen's original SOM formulation [2] which we summarise here. The nodes of the network layers of each modality contain codebook vectors $\mathbf{m}_i = [m_{i1}, \ldots, m_{id}]$, where d is the dimensionality of the modality feature vectors, that are randomly initialized before training begins. At each training step, a data vector $\mathbf{x} = [x_1, \ldots, x_d]$ is is measured against each codebook vector using the Euclidean distance metric, as follows:

$$||\mathbf{x} - \mathbf{m}_i||^2 = \sum_{j=1}^{d} w_j (x_j - m_{ij})^2, \tag{1}$$

where w_j is an element of weight vector $\mathbf{w} = [w_1, \ldots, w_d]$ which is used for a feature selection algorithm in the second phase of training, described in Sec. II-C. The node that is closest to the input data vector based on this metric is called the *best matching unit (BMU)* and both it and its neighbouring nodes are updated using the following update rule

$$\mathbf{m}_i(t{+}1) = \begin{cases} \mathbf{m}_i(t) + \alpha_{\text{SOM}}(t)\,[\mathbf{x}(t) - \mathbf{m}_i(t)] & \text{if } i \in N_c, \\ \mathbf{m}_i(t) & \textit{otherwise}, \end{cases} \tag{2}$$

operating over all $i \in [1, n]$, where $\alpha_{\text{SOM}}(t)$ is the learning rate at time t and N_c is the neighbourhood around the BMU c.

2) Cross-modal Hebbian Mapping: The following Hebbian weight training procedure is taken from Miikulainen [9]. In order to train the Hebbian mapping, we require a measurement of the activation of a given modality layer node a_i, formulated as follows:

$$a_i(t) = \begin{cases} 1 - \frac{||\mathbf{x}(t) - \mathbf{m_i}(t)|| - d_{\min}}{d_{\max} - d_{\min}} & \text{if } i \in N_c, \\ 0 & \textit{otherwise}, \end{cases} \tag{3}$$

where $d_{\min}$ is the smallest and $d_{\max}$ the largest distance of $\mathbf{x}(t)$ to a unit in the neighbourhood.

During training, the link weight changes are made proportional to the product of the activation of the two nodes in each modality that are being associated, as follows:

$$\Delta h_{kl}(t) = \alpha_{\text{HEB}}(t) a_k(t) a_l(t), \tag{4}$$

where Δh_{kl} is the unidirectional associative weight leading from node k in the input modality layer to node l in the output modality layer, $\alpha_{\text{HEB}}(t)$ is the Hebbian learning rate, and $a_k(t)$ and $a_l(t)$ are the activations on the two nodes at time t. Each link weight h_{kl} is then updated using the following normalization equation:

$$h_{kl}(t+1) = \frac{h_{kl}(t) + \Delta h_{kl}(t)}{\sqrt{\sum_l \left[h_{kl}(t) + \Delta h_{kl}(t)\right]^2}}. \tag{5}$$

The Hebbian mapping, when trained as above, provides a type of memory of previous training experience in terms of modality co-occurrences, or of what one modality "looks like" from the perspective of the other. As we shall see, this information can be effectively employed to augment classifier training.

C. Training: Phase 2

After Phase 1 training has proceeded for a reasonable amount of time, i.e., long enough to provide a robust Hebbian mapping, Phase 2 training may be initiated. In the algorithm presented in this paper, the network layer in the output modality continues to be trained with the usual SOM algorithm in Phase 2. The network layer in the input modality however, switches to a modified version of learning vector quantization (LVQ) [2] training that employs a probabilistic heuristic. In order to develop this heuristic, we require the Hellinger distance metric which we discuss next. Phase 2 training is illustrated in Figs. 3 (b) and 3 (c).

1) Hellinger Distance Heuristic: Using the definition from [13], for a countable state space Ω, given probability measures μ and ν,

$$d_H(\mu, \nu) := \left[\sum_{\omega \in \Omega} \left(\sqrt{\mu(\omega)} - \sqrt{\nu(\omega)} \right)^2 \right]^{\frac{1}{2}}. \tag{6}$$

We use the Hellinger distance metric as defined above to create a heuristic that allows us to measure the similarity between nodes in the input modality layer and the output modality layer with respect to the Hebbian mapping. The Hellinger distance takes values in the bounded interval $[0, \sqrt{2}]$, making it amenable to statistical analysis, e.g. calculating mean distance. Given input modality node k, we define

$$f_k(t) = \{h_{kl}(t) : \forall l \text{ in the output modality layer}\}, \tag{7}$$

or all the Hebbian link weights that connect node k in the input modality layer to the nodes in the output modality layer at time t. We define

$$g(t) = \{a_l(t) : \forall l \text{ in the output modality layer}\}, \tag{8}$$

or all the node activations in the output modality layer at time t.

f_k can be thought of as a distribution of the Hebbian map activity from node k in the input layer projected onto the output layer. Loosely put, this gives us a picture of what the output map looks like from the perspective of node k in the input map based on previous training experience. g, on the other hand, gives us a distribution of the output map activity with respect to the current training sample. Thus, when given a training sample for the input modality, if we employ the metric $d_H(f_k(t), g(t))$, we can get an impression of how well its best matching unit node in the input modality layer predicts the activity of the output modality layer given its co-occurring training sample. This heuristic can of course be used in the opposite direction, from the output modality layer to the input modality layer, but for the algorithm we present in this paper it is employed strictly in the above way to augment the training of the input modality layer. Now that we have the necessary tools in place, we may proceed to present our modified LVQ algorithm.

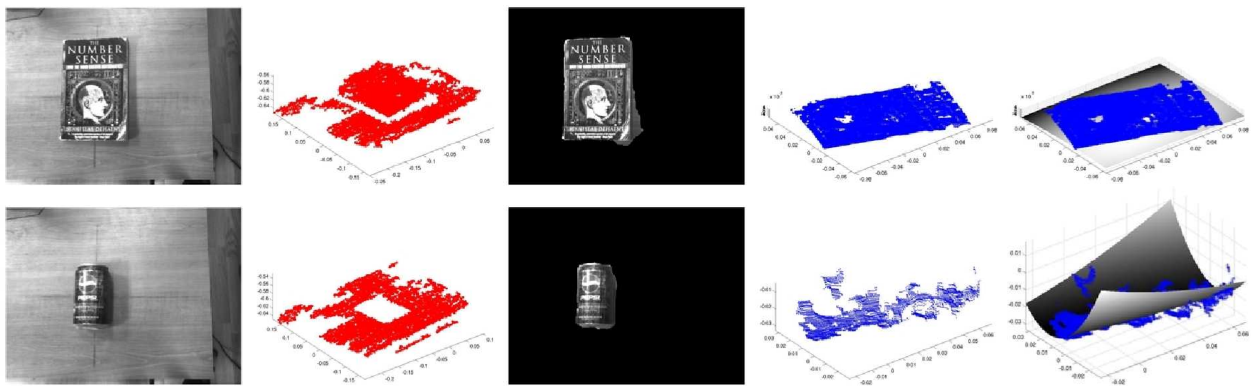


Fig. 4. Examples of image and range data taken with the stereo camera for two different types of objects: a book which slides when pushed by the robotic arm, and a Pepsi can which rolls when pushed by the arm. From left to right: intensity image, range data of the scene, segmented object, segmented object range data, object range data with a fitted quadric surface.

2) Learning Vector Quantization Without Class Labels: In traditional LVQ training [2], codebook vectors are given fixed class labels a priori. Subsequently, as training samples are presented with accompanying class labels, the best matching codebook vectors are updated according to a set of rules. If the best matching codebook vector class label matches that of the training sample, the codebook vector is moved towards the sample. If the labels do not match, the codebook vector is moved away from the sample.

In the modified form of LVQ we present here, which is the main contribution of this paper, we refrain from labeling the codebook vectors altogether. Codebook vectors are updated based on the heuristic presented in the previous section. Given the best matching node c in the input modality layer for a given input modality training sample x, we apply the following update rule:

$$\mathbf{m}_c(t+1) = \begin{cases} \mathbf{m}_c(t) + (1-\gamma)\alpha_{\mathrm{LVQ}\,c}\left[\mathbf{x}(t) - \mathbf{m}_c(t)\right] \\ \textit{if } d_H(f_c(t), g(t)) < \epsilon \\ \\ \mathbf{m}_c(t) + (\gamma-1)\alpha_{\mathrm{LVQ}\,c}\left[\mathbf{x}(t) - \mathbf{m}_c(t)\right] \\ \textit{otherwise}, \end{cases} \tag{9}$$

where, assuming $d_H(f_c(t), g(t))$ is normalised, ϵ is usually set to its mean value over all t, and γ is set to either 0 or $2d_H(f_c(t), g(t))$ if the rule is to be applied in either a binary or fuzzy fashion respectively.

In more simple terms, the effect of applying the above rule is that, when the output modality appears to have the same activity distribution as predicted by the best matching node in the input modality based on past experience, the best matching node in the input modality is moved closer to the training sample. Conversely, if the output modality appears to have a significantly different activity distribution, it is moved away from the training sample. These two alternative cases are visualised in Figs. 3 (b) and 3 (c).

Note that the above also incorporates optimized-learning-rate learning vector quantization (OLVQ), where separate $\alpha_{\mathrm{LVQ}\,c}$ are stored and updated for each node. See [2] for more details.

3) Feature Selection: As alluded to in Sec. II-B.1, we also employ a feature selection method to boost classifier training. We use the relevance determination learning vector quantization (RLVQ) algorithm from [14] to do this. Given the best matching node c in the input modality layer for a given input modality training sample x at time-step t, we perform the following operation on the w_j from (1). For each feature dimension j:

$$w_j(t+1) = \begin{cases} \max\left\{w_j(t) - \alpha_{\mathrm{F}}(t)|\mathbf{x}(t) - \mathbf{m}_c(t)|, 0\right\} \\ \textit{if } d_H(f_c(t), g(t)) < \epsilon \\ \\ w_j(t) + \alpha_{\mathrm{F}}(t)|\mathbf{x}(t) - \mathbf{m}_c(t)| \\ \textit{otherwise}. \end{cases} \tag{10}$$

We then normalise, as follows: for all j, $w_j := w_j/|\mathbf{w}|$.

D. Classification

After training, a classifier may be formed from the network by performing unsupervised meta-clustering over the nodes of the output modality codebook layer. To this end, we used k-means with automatic selection of k based on votes from the following validity indices: Davies-Bouldin [15], Calinski-Harabasz [16], Dunn [15], Krzanowski-Lai [16] and the silhouette index [16], [15]. These clusters define the output modality categories to be used for classification purposes. Given an input modality test sample to be classified, the best matching node in the input modality layer is found and its Hebbian weight links are mapped to the output modality layer. The weights for the links connecting to each cluster are summed, and the cluster with the highest score is deemed to be the winning class for that input test sample.

III. System Architecture & Setup

A. Robotic Arm

In our system, we use a Neuronics Katana 6M robotic arm which features 5 DC motors for main arm movement, as well as a 6th motor to power a 2 fingered gripper that houses both infrared and haptic sensors (note: these sensors are not used in the experiment presented here). The base of

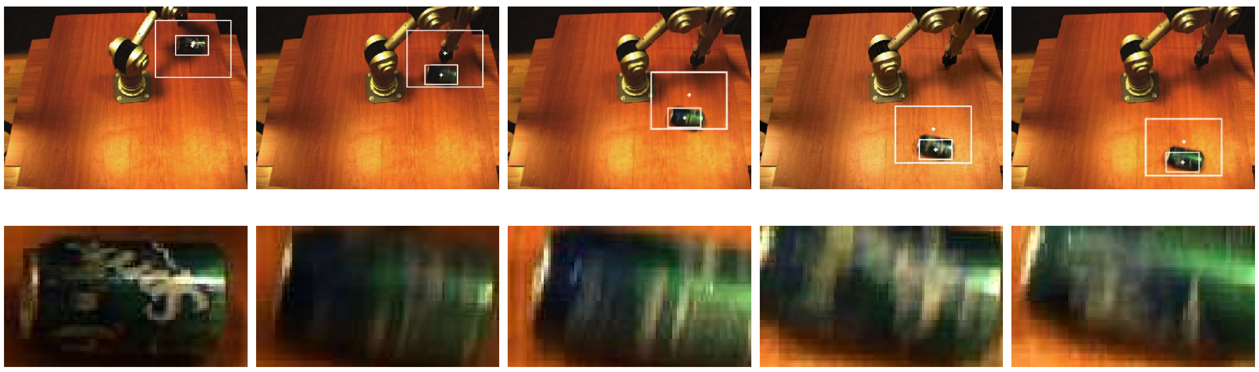


Fig. 5. An example of the object tracking mechanism described in Sec. IV-B using the images in the first row show a progression of frames tracking a Sprite can being pushed by the arm. The outer rectangle is a likelihood window around the object obtained using the particle filter tracker. The inner rectangle is the result of using histogram back-projection within that window to localise the object. The second row of close-up images shows how the appearance of the object within the inner rectangle changes during the course of object motion.

the arm is mounted on a flat table with a wooden laminate surface, and the arm is allowed to move freely in the area above the table surface, avoiding collisions with the table through the use of specialized control software. The arm control software that was used for this work is a modified version of Golem [1], control software for the Katana arm. Given desirable parameters, Golem uses forward kinematics to generate arm joint orientations and motion paths, then uses cost functions and searches to select the ones that most closely fit the parameters. In order to ensure that the actions, and by extension the object affordances, that are available to the system are as consistent and learnable as possible, we optimized for a linear end-effector motion trajectory when moving between workspace positions.

B. Camera Systems

2 Point Gray Research cameras- the Flea monocular camera (640x480 @ 60FPS or 1024x768 @ 30FPS) and the Bumblebee 2 grayscale stereo camera (640x480 at 48FPS or 1024x768 at 20FPS) were used to gather intensity images, range data and video for the experiment listed in Sec. V.

IV. Visual Feature Extraction

A. Input Modality Feature Extraction

With regard to the input modality features, for the purposes of this particular affordance learning scenario, we are mostly interested in extracting features that describe the global shape of an object as they are likely to be most relevant for determining how the object will behave. However, in theory, any types of features that describe properties of the objects under consideration could be used here.

1) Range Data: We have developed a method for segmenting the object from range images that uses RANSAC (RANdom SAmple Consensus) [17] to fit a plane to the table surface for removal, then mean-shift clustering [18] as well as a graph-cut segmentation in the corresponding intensity image to isolate the object range data with minimal noise. The graph-cut segmentation method we used was from [19], which uses the min-cut/max-flow algorithms oulined in [20], [21], [22] to apply the standard graph cut technique to segmenting multimodal tensor valued images. A quadratic surface may then be fitted to the object range data to derive curvature features from the object surface. We derive 2 curvature features in this way from the coefficients of the polynomial of the fitted quadratic surface that provide a good description of the global curvature of the object. This surface fitting technique is illustrated on the two objects shown in Fig. 4.

2) Image Data: The segmentation technique produces reasonably good intensity image segmentations of objects. These are then used to calculate the following 10 shape features: area, convex area, eccentricity, equivalent cicular diameter, Euler number, extent, filled area, and the major axis length.

B. Output Modality Feature Extraction

After an arm action has been performed on an object, the resulting videos of the interaction are processed for output modality features. This is primarily acheived by tracking the object in motion using a probabilistic tracker from [23]. This tracker is in essence a colour-based particle filter, which also makes use of background subtraction using a pre-learned background image. Background subtraction by itself is insufficient to localise the object in our experimental setup due to changes in lighting and the motion of the arm, but it is helpful in reducing ambiguities for the tracker. Object shapes are approximated by elliptical regions, while their colour is encoded using colour histograms. The dynamics of objects are modeled using a dynamic model from [24], which allows for tracking with a smaller number of particles, and consequently, near real-time tracking performance.

1) Global Object Motion Features: The following 9 features are calculated from the particle filter tracker output data: total distance traveled in x-axis, total distance traveled in y-axis, total Euclidean distance traveled, mean velocity in

[1]Golem was developed by researchers at the University of Birmingham who kindly provided us with a copy for our research. More information can be found at: http://www.cs.bham.ac.uk/˜msk/ .

x-axis, mean velocity in y-axis, velocity variance in x-axis, velocity variance in y-axis, final x position, final y position.

2) Object Appearance Changes: To estimate how the appearance of the objects change during motion, we chose to calculate the average difference of both colour and edge histograms between video frames of the objects, the aim being to detect both motion blur and the texture changes characteristic of many rotating objects. This required an extension to the particle filter tracker previously described. The tracker by itself is sufficient for tracking the motion of objects, but it is slightly inaccurate at times. For example, if an object is rolling and stops suddenly, the tracker sometimes briefly overshoots the object before returning to it a few frames later. To avoid this, we use the output of the tracker to define a broad window around the object in the video frames, before using colour histogram back-projection [25] to localise the object within the window. Histogram difference averages are then calculated from the start of object motion until the end. See Fig. 5 for sample frames from an interaction with an object that illustrates this technique at work. We derive 3 output modality features from this procedure: average colour histogram difference, average edge histogram difference, and the product of these two values.

V. Experiments

To test our affordance learning system, the experimental environment was set up as previously described and as shown in Fig. 2(a). During experiments, objects were placed at a fixed starting position prior to interaction. Two cameras were used to provide both sufficiently detailed close-up range data of the object surfaces and a sufficiently wide field of view to capture object motion over the entire work area. To achieve this, the stereo camera was positioned above the object start position, while the monocular camera was positioned at a higher position in front of the workspace.

We selected 8 household objects to be used in the experiments: 4 flat-surfaced objects; a book, a CD box, a box of tea and a drink carton, and 4 curved-surfaced objects; a box of cleaning wipes, a Pepsi can, a Sprite can and a tennis ball box. Each of these objects was placed centred at the start position with a consistent orientation, and the robotic arm pushed the object at a fixed speed using a fixed pushing action. During trials, the curved objects would tend to roll after being pushed, whereas the flat objects would stop suddenly after the push. Before an action was performed on an object, both intensity and range images were gathered from the stereo camera. This data was then processed to produce the 12 input modality features discussed in Sec. IV-A. After an action was performed on an object, images were gathered and passed to the tracking system described in Sec. IV-B to produce 12 output modality features.

To evaluate the algorithm, we first collected a dataset as follows. 20 object push tests were carried out for each of the 8 objects listed previously and the resulting data was processed, leaving 160 data samples. The samples were then hand-labeled with two ground truth labels: rolling and non-rolling. In the following evaluations, leave-one-out cross validation was performed by splitting the dataset into a training set of 140 samples consisting of all data for 7 of the objects and a test set of 20 samples consisting of all data for the remaining object. The classification task was then to train on 7 objects, find the affordance classes in the output modality and try to classify the remaining object on that basis. In the experiments for this paper, the training set was doubled and randomized, effectively allowing for 2 epochs of training over the training set, i.e. training over 280 samples. Cross validation was performed by using each of the 8 objects in turn as the test object and averaging classification scores across all 8 subsequent training and test sets and the 20 test samples contained therein. Fig. 6 shows the results of incrementally cross-validating 6 algorithms every 20 training steps and averaging over 40 trials. In 4 of the algorithms, each of the two codebook vector layers in the input and output modalities contained 100 nodes arranged in a 10×10 hexagonal lattice with a sheet-shaped topology. In the other 2 algorithms, the codebook vector layers contained 5 nodes arranged in a 1×5 linear topology.

The goal of the evaluation was to compare the performance of our self-supervised algorithm to fully-supervised learning using ground truth labels. In the case of the self-supervised algorithms, classification of a test sample was deemed to be correct if the output modality meta-cluster (c.f. Sec. II-D) matched the ground truth (c.f. [26] for more details on matching the meta-clusters to ground truth labels). Of the 6 algorithms evaluated, 2 were variations on fully-supervised LVQ1, OLVQ1 with 100 nodes and OLVQ1 with 5 nodes, while the remaining 4 were variations of self-supervised cross-modal learning. Of the 4 variations of cross-modal learning, one was cross-modal SOM training as in [9] with 100 nodes and the other 3 were modifications of our proposed heuristic-based LVQ algorithm: fuzzy heuristic OLVQ1 with RLVQ feature selection (HeurORLVQ), binary HeurORLVQ with 100 nodes, and binary HeurORLVQ with 5 nodes. For each of these, the initial α_{SOM} learning rate in each modality was set to 1 with a linear profile descending to 0 over the 280 timesteps in the output modality and 140 timesteps in the input modality. The RLVQ α_{F} learning rate was set to a constant 0.1. Training shifted from Phase 1 to Phase 2 halfway through the training set (140 timesteps), In Phase 2, α_{LVQ} was set to a constant 0.3. These learning rates were selected both through trial and error, and as advised by [2].

As can be seen in Fig. 6, the fully supervised algorithms performed the best, as expected, with the 5-node OLVQ reaching a correct classification rate of 97.11% by the end of training, and the 100-node OLVQ reaching a score of 93.53%. Of the self-supervised cross-modal classifiers, 100-node Fuzzy HeurORLVQ performed the best, reaching a correct classification rate of 91.64%, while 100-node HeurORLVQ and 5-node HeurORLVQ reached rates of 90.16% and 86.67% respectively. The cross-modal SOM finished with a score of 81.91%, thus justifying our two-phased learning approach. Our algorithm works best when there are enough nodes in the network to give a decent approximation of the

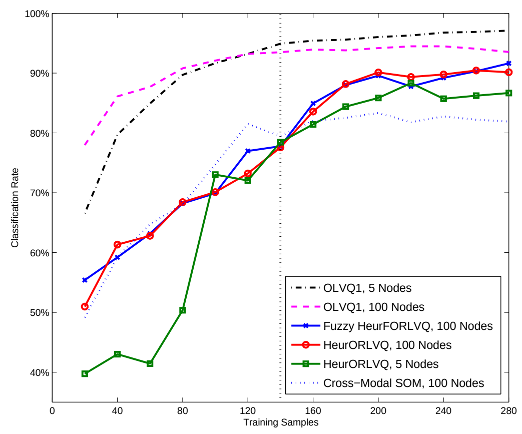


Fig. 6. Incremental leave-one-out cross-validation evaluation averaged over 40 trials (c.f. Sec. V). The switch from Phase 1 training to Phase 2 training (c.f. Sec. II) is indicated by the vertical dotted line.

sample density in the output modality, thus providing a more accurate Hellinger distance heuristic. The results show that, when evaluated with such a number of nodes, our algorithm performs almost as well as fully-supervised OLVQ1 using the same number of nodes. On the other hand, OLVQ1 works best when the number of training steps is 30 to 50 times the number of codebook vectors [2], i.e. for our small training set size of 280 samples, the number of codebook vectors should be low, e.g., 5. However, even when using such a small number of nodes, in this 2-class learning scenario our algorithm performed robustly, and still produced better results than the 100-node cross-modal SOM.

VI. Conclusion

In conclusion, we have presented a robotic system that uses a novel self-supervised cross-modal online classifier training algorithm to learn basic object affordances. We have shown that it can be successfully trained to learn affordances of household objects by interacting with them, and subsequently predict the affordance classes of novel objects by observing their object features, e.g. shape. The experimental results also demonstrated how the system, through the use of the proposed novel algorithm, can start learning with little or no experience, and improve results over time to the point where the classification rate is close to that of a fully-supervised system. Although the results presented here only account for one type of action, multiple classifiers may be trained to account for different types of actions. We aim to improve on this in future work by modifying the algorithm such that actions may be parameterized, perhaps in a separate modality. We would also like to test the algorithm on more challenging problems where there are more than two classes present in the training data.

References

[1] J. J. Gibson. *The ecological approach to visual perception*. Lawrence Erlbaum, 1986.

[2] T. Kohonen. *Self-organizing maps*. Springer, 1997.

[3] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini. Learning about objects through action-initial steps towards artificial cognition. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, volume 3, 2003.

[4] G. Metta and P. Fitzpatrick. Early integration of vision and manipulation. *Adaptive Behavior*, 11(2):109–128, 2003.

[5] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory-motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1):15–26, 2008.

[6] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng. Robotic grasping of novel objects. In *In Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems Conference*, Vancouver, Canada, 2006.

[7] A. Saxena, J. Driemeyer, and A. Y Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157, 2008.

[8] E. Ugur, M. R. Dogar, M. Cakmak, and E. Sahin. The learning and use of traversability affordance using range images on a mobile robot. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, page 1721–1726, 2007.

[9] R. Miikkulainen. Dyslexic and Category-Specific aphasic impairments in a Self-Organizing feature map model of the lexicon. *Brain and Language*, 59(2):334–366, 1997.

[10] V. R. de Sa. Learning classification with unlabeled data. In *In Proceedings of the Eight Annual Conference on Neural Information Processing Systems*, volume 93, pages 112–119, 1994.

[11] V. R. de Sa and D. H. Ballard. Category learning through multimodality sensing. *Neural Computation*, 10:1097–1117, 1998.

[12] I. Cos-Aguilera, L. Canamero, and G. Hayes. Using a SOFM to learn object affordances. In *Proceedings of the 5th Workshop of Physical Agents (WAF'04), Girona, Spain*, 2004.

[13] A. L. Gibbs and F. E. Su. On choosing and bounding probability metrics. *International Statistical Review/Revue Internationale de Statistique*, pages 419–435, 2002.

[14] T. Bojer, B. Hammer, D. Schunk, and K. T. von Toschanowitz. Relevance determination in learning vector quantization. In *European Symposium on Artificial Neural Networks*, pages 271–276, 2001.

[15] N. Bolshakova and F. Azuaje. Cluster validation techniques for genome expression data. *Signal Processing*, 83(4):825–833, 2003.

[16] S. Dudoit and J. Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, 3(7), 2002.

[17] M. A Fischler and R. C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[18] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[19] J. Malcolm, Y. Rathi, and A. Tannenbaum. A graph cut approach to image segmentation in tensor space. In *Workshop on Component Analysis Methods (CVPR)*, pages 18–25, 2007.

[20] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[21] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

[22] Y. Boykov and V. Kolmogorov. An experimental comparison of Min-Cut/Max-Flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[23] M. Kristan, J. Perš, M. Perše, and S. Kovačič. Closed-world tracking of multiple interacting targets for indoor-sports applications. *Computer Vision and Image Understanding*, 113(5):598–611, 2009.

[24] M. Kristan, J. Perš, A. Leonardis, and S. Kovačič. A hierarchical dynamic model for tracking in sports. In *Proceedings of the Sixteenth Electrotechnical and Computer Science Conference*, September 2007.

[25] M. J Swain and D. H Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[26] B. Ridge, D. Skočaj, and A. Leonardis. Unsupervised learning of basic object affordances from object properties. In *Proceedings of the Fourteenth Computer Vision Winter Workshop*, pages 21–27, Eibiswald, Austria, 2009.

Offline and active gradient-based learning strategies in a pushing scenario

Sergio Roa and Geert-Jan Kruijff
German Research Center for Artificial Intelligence / DFKI GmbH
{sergio.roa,gj}@dfki.de[1]

Abstract.
When operating in the real world, a robot needs to accurately predict the consequences of its own actions. This is important to guide its own behavior, and in adapting it based on feedback from the environment. The paper focuses on a specific problem in this context, namely predicting affordances of simple geometrical objects called polyflaps. A machine learning approach is presented for acquiring models of object movement, resulting from a robot performing pushing actions on a polyflap. Long Short-Term Memory machines (LSTMs) are used to deal with the inherent spatiotemporal nature of this problem. An LSTM is a gradient-based model of a Recurrent Neural Network, and can successively predict a sequence of feature vectors. The paper discusses offline experiments to test the ability of LSTMs to solve the prediction problem considered here. Cross-validation methods are applied as a measure of convergence performance. An active learning method based on Intelligent Adaptive Curiosity is also applied for improving the learning performance of learners trained offline, generating a combination of learners specialized in different sensorimotor spaces after the knowledge transfer.

1 Introduction

Robots need to learn in continuously changing environments. One way to learn from the world is by interacting with objects present in it. This work is inspired by the fact that humans and animals in general are able to properly adapt to a dynamic environment. Theories of cognitive development like the theory of affordances [5] attempt to explain how creatures are able to acquire sensorimotor skills when they are faced with the different features found in the environment. For instance, surfaces afford posture, locomotion, collision, manipulation, and in general behaviour [5]. Particular objects can afford sliding, flipping, rolling behaviour. A consequence of this is that the creature should then properly predict the consequences of actions on a surface given its own body configuration. We consider here a special case of affordances learning in robots, namely that of predicting consequences of pushing simple geometrical objects called *polyflaps*. Polyflaps have been proposed to design simple learning scenarios. A polyflap is a polygon (concave or convex) cut out of a flat sheet of some material (e.g. cardboard) and folded once (anywhere) to produce a 3-D object [17], cf. Fig. 1. By combining different objects and performing different actions, we can steadily increase the complexity of the learning environment.

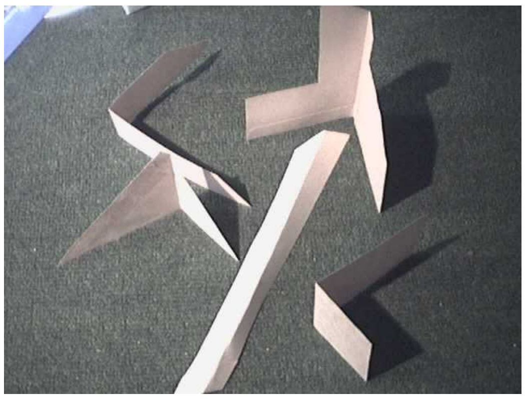


Figure 1. Polyflaps, http://www.cs.bham.ac.uk/∼axs/polyflaps/. Used here are polyflaps of the shape (bottom-right corner)

In this paper we discuss a learning scenario where a simulated robotic arm interacts with a polyflap. In the implementation we use the NVidia® PhysX™ library that allows us to perform realistic physical simulations and to obtain 3-dimensional feature vectors, so that we can easily re-adapt our algorithms to real scenarios. Although providing an idealized scenario, these experiments are necessary to establish a base line from which we can start facing noisy and incomplete features, where learning machines should be able to generalize and present outcomes in the presence of uncertainty. The learning machines we use are able to process spatio-temporal features. Specifically, we use the Long Short-Term Memory (LSTM) [7, 6] model of an Artificial Neural Network. The main objective is that the robot arm pushes the object and predicts a sequence of polyflap poses encoded as rigid body transformations during a certain time interval following the pushing action. To reduce the space- and time complexity of the problem, we select a discrete set of possible actions and starting positions for the arm to start the pushing movement. This reduction of dimensionality affords us also to evaluate and analyse more easily and carefully the learning algorithms and its corresponding results. In general, *sliding* and *flipping* affordances are obtained by applying pushing actions. The experiments show that the machines are able to model a sort of regression function that fits the data very accurate. This fact is also crucial from the point of view of dimensionality reduction, since the use of a learning machine together with its generalization abilities can highly reduce the need

[1] The research reported of in this paper is supported by EU FP7 IP "CogX" (ICT-215181)

of storage space. Moreover, the inherent recurrent topology of these networks affords the reduction of space needed for storing spatio-temporal information.

The characteristics of these learning machines are appropriate for autonomous development of robots [10, 14]. Robots should be able to autonomously acquire sensorimotor skills by interaction with the environment. Thus, machines that are able to learn in an online and active manner need to be used. Neural Networks in general are useful for these tasks, since their weights can be updated efficiently by using one forward and one backward pass when we use gradient-based methods. However, one has to be careful with the problem of overfitting data (bias-variance tradeoff). Therefore, a sufficiently big set of samples and iterations are needed in order to generalize sufficiently well a dataset.

We tested the topology of the neural network in order to find a good compromise between computational complexity and generalization ability. For that purpose, we extracted n-fold crossvalidation sets and analyzed the average sum of squares error for all training epochs. The problem that we tackle can be regarded as a time series prediction problem approached by regression techniques. Therefore, the sum of squares error is a good performance estimation. The experiments show that the machines are able to accurately predict a feature vector, given a history of precedent feature vectors that together form a sequence. After offline training, we applied an active learning technique based on the Intelligent Adaptive Curiosity algorithm [10, 14], by including an additional set of actions in order to test the autonomous generation of different regions in the sensorimotor space that allows an active selection of samples via maximization of a measure of learning progress and multiple learners specialized in each region. We also show that the generalization is improved by the set of machines (which are only "biased" for their corresponding regions).

This paper is organized as follows. In the next section, we present a current state of the art related to affordances learning in robots and recurrent neural networks. In section 3, we describe the learning scenario and the features we used for training LSTMs. In section 4 we present the offline learning mechanism and architecture employed. In section 5 we show and explain experimental results for offline experiments with LSTMs. In section 6 we explain the active learning mechanism and results and in section 8 we present some concluding remarks and planned work.

2 Related Work

Affordances learning has been introduced in the field of robotics in recent years. The reason is that aiming to autonomous behaviour in robots requires an inspiration from biological cognitive systems, which are very successful on acquiring sensorimotor skills by their own means. As a cognitive science theory, the field was introduced by the perceptual psychologist J.J. Gibson [5]. An affordance in this sense is a resource or support that the environment offers an agent for action, and that the agent can directly perceive and employ. From the robotic perspective, this concept implies that the robots should be able to predict consequences of actions given certain object features and robotic embodiment.

In the field of robotics, a compilation of works related to affordance-based robot control can be found in [15]. A similar approach to the one presented in this work is described in [12]. In that work, labels of object/action pairs and 11 features encoding the action performed and the object behaviour are used to train Self Organizing Maps. In this way, they cluster this space and map the features to the target function represented by such labels. Pushing actions were performed on different objects in a real environment. Other approaches have used also similar features and learning methods and have studied different kinds of affordances [2].

Perception of affordances has also been addressed with reinforcement learning techniques. In [11], the robot performs different learning stages starting from recognizing affordances and finally accomplishing some task given the affordances that the robot has already acquired in earlier stages. In that work, liftable vs. non-liftable objects are recognized and Markov Decision Processes are used for the goal-based task. In [10, 14] the robot autonomously enters different stages of development by interacting with objects or performing some action, which is selected according to a measure of "interestingness". Thus, robots are intrinsically motivated to perform actions that offer an opportunity to learn according to an estimation of learning progress calculated from prediction error histories.

However, we can consider that these aproaches use a kind of short-term memory or mapping approach that does not take into account the spatio-temporal processing of data when an action is performed in a given time interval. Moreover, in some approaches there is an explicit labelling of the recognized affordances or the robot has no means to evaluate the accuracy of its predictions. In order to evaluate the abilities of learning machines in processing a series of features like rigid body transformations that gives us a more accurate assessment of the object poses and behaviour, we are using recurrent neural networks that are known to process sequences and obtain proper generalizations by infering regression functions.

A simulated scenario using also polyflaps is described in [9]. The authors formalise the learning problem in a probabilistic framework. Explicit 3D rigid body transformations are predicted by that models and they are tested against novel objects similar in shape to polyflaps.

Long Short-Term Memory machines have been used for problems like time-series prediction, sequences classification, phoneme classification, reinforcement learning, among others [7, 6, 1]. They are appropriate to handle long-term dependencies in data sequences. Therefore, they seem to have a high potential to be used in learning tasks where compositionality and conditional dependencies of events or states is encountered through a relatively broad time period.

The work described in this paper is a follow-up of the one presented in [13].

3 Learning Scenario

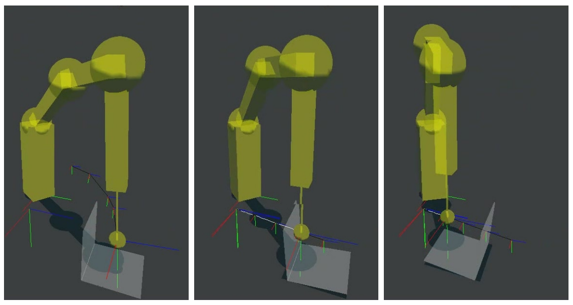


Figure 2. Learning scenario with a polyflap

The learning scenario is shown in Fig. 2. The simulated arm corresponds to a Neuronics® Katana 6M$^{\text{TM}}$ arm with a ball as a simple finger. In order to simulate a pushing action we apply a linear trajectory over a specified time period until it reaches the desired pose. The arm has 6 joints, including the last joint for the finger which is static. The representation of object poses are in Euler angles with respect to a reference frame which is the origin in the scene (6-D pose).

The features corresponding to the arm are a starting 6-D pose vector for the end-effector $\mathbf{e}_0$, and a real value denoting a direction angle Θ ranging from 60 to 120 degrees, parallel to the ground plane in the direction to the center of the standing polyflap side. Together, these features form the motor command feature vector denoted as $\mathbf{m}$. The values are all normalized to obtain vectors with mean 0 and standard deviation 1.0. A 6-D pose vector corresponding to the polyflap pose is denoted as $\mathbf{p}_t$ at time t. The pose $\mathbf{p}_0$ is fixed for all experiments.

Then, the concatenation $\mathbf{f}_0 = [\mathbf{m}\ \mathbf{e}_0\ \mathbf{p}_0]$ represents the feature vector to be fed initially to the neural network. The subsequent feature vectors fed to the machine have the form $\mathbf{f}_t = [\mathbf{0}\ \mathbf{e}_t\ \mathbf{p}_t]$, where the size of $\mathbf{0}$ is the size of $\mathbf{m}$. This representation affords the learning machine to attain a better convergence.

During the execution of the arm path, we obtain a series of poses $\langle \mathbf{p}_t, \mathbf{e}_t \rangle$ to construct a feature vector $\mathbf{f}_t$. We extract then n polyflap and effector poses and finally we build a sequence set $S = \{\mathbf{f}_{t=1}^{n}\}$. So, a particular sequence set (an instance) is used in each iteration of the experiment to be fed to the LSTM in $n+1$ steps. For the time step t, a training tuple $\langle \mathbf{f}_t, \mathbf{t}_t \rangle$ is used for the neural network learning procedure, where the feature vector $\mathbf{f}_t$ represents the input vector and $\mathbf{t}_t = \mathbf{p}_{t+1}$ the target (predicted) vector encoding the predicted polyflap pose.

This representation then encodes the rigid body transformations of polyflap and effector through these n steps and also encodes the given robot control command that performs the pushing movement. In order to discretize and reduce the dimensionality of the task, we only used a discrete number of different starting positions for the arm to start the pushing movement.

4 Offline Learning method

The learning process used for training LSTMs with the features described in section 3 is described here. As mentioned in the previous section, a dataset $\mathcal{D}$ containing a certain quantity of sequences S_i is obtained and we perform offline experiments with these data.

A LSTM machine is usually composed of an input layer, a hidden layer and an output layer. In general, recurrent neural networks can have recurrent connections for all their neurons. In particular, in this work we only use recurrent connections for the hidden layers. We also made preliminary experiments with networks with no recurrent connections and we found less performance. The LSTM [7, 6, 1] architecture was developed in order to solve some learning issues in recurrent neural networks related to long-term dependencies learning. These problems sum up to the problem that errors propagated back in time tend to either vanish or blow up. This is known as the problem of vanishing gradients.

LSTM's solution to this problem is to enforce *constant* error flow in a number of specialized units, called Constant Error Carrousels (CECs), corresponding to those CECs having linear activation functions not decaying over time. CECs avoid to transmit useless information from the time-series by adding other input gates that regulate the access to the units. Thus, they learn to open and close access to the CECs at appropriate moments. Likewise, the access from the CECs to output units is controlled by multiplicative output gates and they learn in a similar way how to open or close the access to the output side. Additionally, forget gates [3] learn to reset the activation of the CECs when the information stored in them is no longer useful, i.e., when previous inputs need to be forgotten. The combination of a CEC with its associated input, output and forget gate is called a memory cell, as depicted in Fig. 3. Other additions are peephole weights [4], which improve the LSTM's ability to learn tasks that require precise timing and counting of internal states, and bidirectional connections [16].

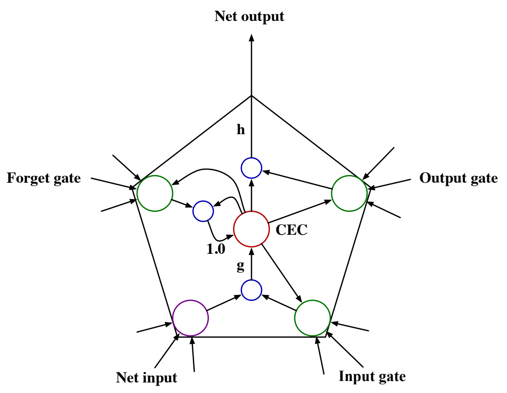


Figure 3. LSTM memory block with one cell. The internal state of the cell is maintained with a recurrent connection of fixed weight 1.0. The three gates collect activations from inside and outside the block, and control the cell via multiplicative units (small circles). The input and output gates scale the input and output of the cell while the forget gate scales the internal state. The cell input and output activation functions (g and h) are applied at the indicated places [6].

In this work, we used 10 memory blocks in the hidden layer, which was found to be a good compromise between computational complexity and convergence.

When some input vector is fed to the network, the forward pass is calculated as follows. Let us denote an output neuron (unit) activation y^o, an input gate activation y^{in}, and output gate activation y^{out} and a forget gate activation y^f. Then, for the time step t each of them are calculated in the following standard way:

$$y^i(t) = f_i(\sum_j w_{ij} y^j(t-1)), \tag{1}$$

where w_{ij} is the weight of the connection from unit j to unit i, and f the activation function. In this paper, we only consider one CEC activation (one cell) for each memory block. The CEC activation s_c for the memory cell c is computed as follows:

$$s_c(t) = y^{f_c}(t) s_c(t-1) + y^{\text{in}_c}(t) g(\sum_j w_{cj} y^j(t-1)), \tag{2}$$

where g is the cell input activation function. The memory cell output is then calculated by

$$y^{s_c}(t) = y^{\text{out}_c}(t) h(s_c(t)), \tag{3}$$

where h is the cell output activation function. The backward pass is a steepest (gradient) descent method which updates the weights of the different types of units. Consider a network input $a_j(t)$ to some unit j at time t. In general, the gradient is defined as:

$$\delta_j(t) = \frac{\delta E}{\delta a_j(t)}, \tag{4}$$

where E is the objective (error) function to be minimized and used for training. For a detailed explanation of the backward pass equations for each unit type cf. [6]. Since we are dealing with a regression problem, we consider the sum of squares error as a performance measure. The error function is defined as:

$$E_t = \frac{1}{2K} \sum_i (y_i - y_i')^2, \tag{5}$$

where K is a normalization factor which depends on the size of each sequence n_i and the total number of sequences in the dataset k. y_i is the output unit activation and y_i' is the expected value. The learning process is described in the Algorithm 1.

```
Data: A dataset $\mathcal{D}_1$ containing $k$ sequences of variable size $n_i$
      for training. A dataset $\mathcal{D}_2$ containing $z$ sequences of size
      $n_j$ for testing.
Result: An LSTM machine after error minimization.
Nr. of epochs $ep = 0$.
repeat
    for i=1 to $k$ do
        for j=1 to $n_i$ do
            Input: Present training tuple $\langle \mathbf{f}_{ij}, \mathbf{t}_{ij} \rangle$ ($j$th forward
                   pass step).
        end
        Calculate error $e_i$ associated to current training
        sequence $S_i$.
        Backward pass.
    end
    Evaluate error $E_t$ with the test set $\mathcal{D}_2$.
    Epoch $ep = ep + 1$.
until No new network found with lowest error after 20 epochs ;
```
Algorithm 1: Offline learning process

For the purpose of calculating the number of training sequences that are necessary so that convergence improves, we generated n-fold cross-validation sets. We split a dataset $\mathcal{D}$ into n disjoint sets of equal size that are used for testing. We used the remaining data for training n different networks.

5 Experimental results for Offline Learning

In order to test the convergence of LSTMs we used 10-fold cross-validation sets for three different dataset sizes, namely 100, 200 and 500. That allowed us to estimate the approximate number of samples that are needed to learn with high precision the prediction task.

In Fig. 4 a comparison of the average sum of squares error (SSE) and SSE standard deviation is shown. In this case, the SSE is averaged among all the cross-validation sets. The picture shows that the SSE is considerably reduced when more samples are used, as expected, and likewise the standard deviation of the SSEs.

6 Active Learning

The active learning procedure is based on the work of Oudeyer et al. [10] about Intrinsic Motivation Systems. The general idea of the Intelligent Adaptive Curiosity (IAC) algorithm is that a meta-learning system samples a set of actions and selects one that maximizes the learning progress, which is a measure based on the difference between smoothed current and previous mean error quantities. The learning progress $\mathcal{L}_r$ is associated to a region $\mathcal{R}_r$ in the sensorimotor space. Starting with one region, successive regions are obtained by splitting the sensorimotor space depending on a measure

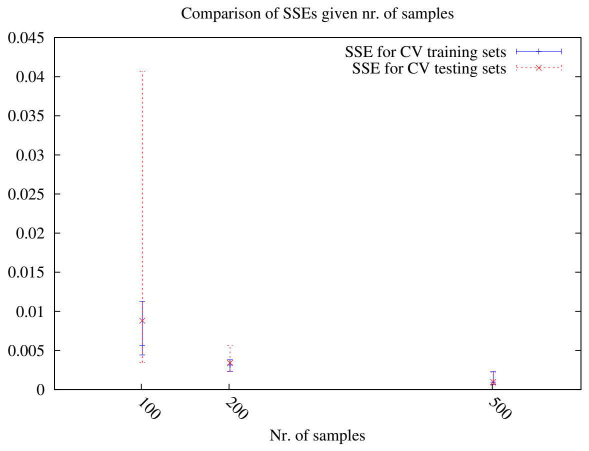


Figure 4. SSEs are reduced when increasing the dataset size.

of variance in the dataset $\mathcal{D}_r$ (exemplars used for Region $\mathcal{R}_r$). This division is performed after $|\mathcal{D}_r|$ achieves a certain threshold κ. A dataset $\mathcal{D}_r$ for a Region R_r is split in two datasets $\mathcal{D}_{r+1}, \mathcal{D}_{r+2}$ (for regions $\mathcal{R}_{r+1}, \mathcal{R}_{r+2}$). Let us denote

$$\mathcal{D}_r = \{S_i\}$$

the set of instances in region $\mathcal{R}_r$. Then the split of $\mathcal{D}_r$ defined by the index c with value v_c is performed when the following criterion (Γ) is met:

- all the instances S_i of $\mathcal{D}_{r+1}$ have the cth component of their motor command vector $\mathbf{m}_i$ smaller than v_c.
- all the instances S_i of $\mathcal{D}_{r+2}$ have the cth component of their motor command vector $\mathbf{m}_i$ greater than v_c.
- the quantity $|\mathcal{D}_{r+1}| \cdot \sigma(\{[\mathbf{e}_{ij}\ \ \mathbf{p}_{ij}]_{j=1}^{n_i} \in \mathcal{D}_{r+1}\}) + |\mathcal{D}_{r+2}| \cdot \sigma(\{[\mathbf{e}_{ij}\ \ \mathbf{p}_{ij}]_{j=1}^{n_i} \in \mathcal{D}_{r+2}\})$ is minimal, where

$$\sigma(\mathcal{S}) = \frac{\sum_{v \in \mathcal{S}} \|v - \frac{\sum_{v \in \mathcal{S}} v}{|\mathcal{S}|}\|^2}{|\mathcal{S}|}$$

where $\mathcal{S}$ is a set of vectors.

Each region stores all cutting dimension and values that were used in its generation as well as in the generation of its parent regions. For the region $\mathcal{R}_r$ a learning machine $\mathcal{M}_r$ is stored, and this machine is inherited by the child regions. The learning process is described in the Algorithm 2.

```
Data: An initial region $\mathcal{R}_0$ which encompasses the whole
      sensorimotor space.
Result: A set of regions $\{\mathcal{R}_r\}$ with corresponding LSTM
        machines $\{\mathcal{M}_r\}$.
for i=1 to $I$ do
    Choose a motor command action
    $\mathbf{m}_{r,i} = \arg\max_{\mathbf{m}\in\{R_r\}}\{\mathcal{L}_{r,i}\}$ among all current regions
    $\{\mathcal{R}_r\}$ by using a near to greedy policy with probability 0.3.
    if $\kappa$ then
        Split region $\mathcal{R}_r$ into $\mathcal{R}_{r+1}$ and $\mathcal{R}_{r+2}$ according to $\Gamma$.
    end
    Calculate error $e_{r,i}$ associated to current training sequence
    $S_{r,i}$.
    Update the machine $\mathcal{M}_r$ with a forward and backward pass.
    Calculate smoothed mean error $\varepsilon_{r,i+1}$ and $\varepsilon_{r,i+1-\tau}$ with a
    window parameter $\tau$ and a smoothing parameter $\theta$.
    Calculate the decrease in the mean error rate
    $\Delta_{r,i+1} = \varepsilon_{r,i+1} - \varepsilon_{r,i+1-\tau}$.
    Calculate the learning progress $\mathcal{L}_{r,i+1} = -\Delta_{r,i+1}$.
end
```

Algorithm 2: Active learning process

7 Experimental Results for Active Learning

In order to test the active learning mechanism, the main idea is to train offline a LSTM with a subset of all possible starting positions producing a partial set of actions and thus a dataset $\mathcal{D}_0 \subset P_0$, where P_0 is the sensorimotor manifold encompassing D_0. Then, we use this machine in the active learning loop allowing additional actions, so that at the end we generate a dataset $\mathcal{D}_1 \subset P_1$, where $P_0 \subset P_1$. The hypothesis is that the algorithm will start producing more frequently actions corresponding to the sensorimotor regions associated to the new actions.

Thus, we first trained offline a LSTM with a subset of possible starting positions for the arm movement and a number of sequences equal to 500. This generates the dataset $\mathcal{D}_0$. When initializing the active learning procedure, we allowed all possible starting positions for the arm movement. Then, we initialize the region R_0 with the already trained machine $\mathcal{M}_0$ that introduces better generalization performance according to the cross-validation sets. We apply a maximum number $I = 300$ of iterations, after which a new dataset $\mathcal{D}_1$ is generated. Then, we merge the datasets into a set $\mathcal{D} = \mathcal{D}_0 \cup \mathcal{D}_1$. We use the set $\mathcal{D}$ to test the errors of the machine trained offline and the ensemble of machines trained via active learning. The results are shown in Table 1.

Table 1. An ensemble trained via IAC against an offline trained machine.

Machine	Avg SSE
Offline	0.4251
Active	0.211991

The unique observation here is that the generalization performance is improved by using the new active learner, which is a expected result. In order to check the hypothesis presented above, we analysed the learning progress of the ensemble of machines created after splitting the sensorimotor space in different regions.

As expected, the algorithm starts to select very frequently actions that are new or "interesting". In Fig. 5, we can observe the frequency of actions generated from each set of starting positions for a window of 20 iterations. For instance, from index ∼150 to ∼250 the new set of actions are more frequent. This result also confirms the generation of different stages of development that the IAC algorithm produces [10]. We make the same observation for a specific region (Fig. 6). In Fig. 7 the curves of learning progress and error for the corresponding region are shown. We can observe that the learning progress curve rises and the error drops.

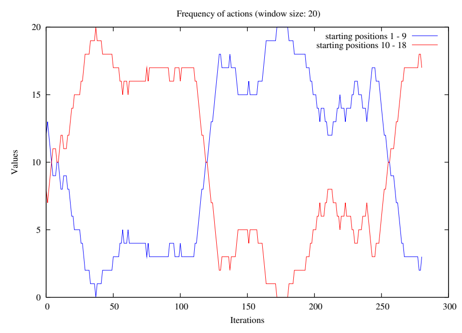


Figure 5. Frequency of actions in the experiment.

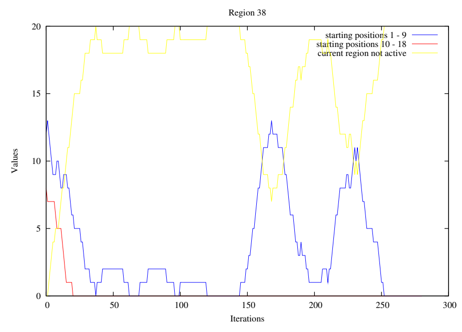


Figure 6. Frequency of actions for a specific region (window size: 20).

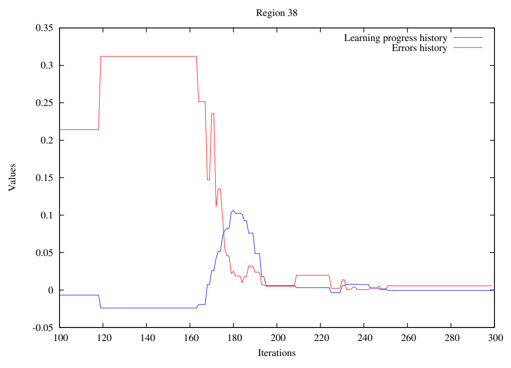


Figure 7. Learning progress increase for a certain region.

In Figures 8,9 the prediction ability of a learning machine over a sequence is illustrated.

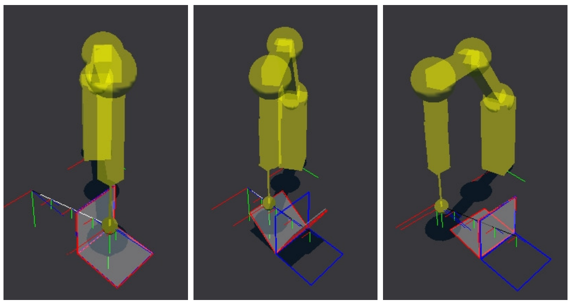


Figure 8. Prediction of the flipping affordance. The blue polyflap is the first predicted polyflap in the current sequence and the red one the last predicted one.

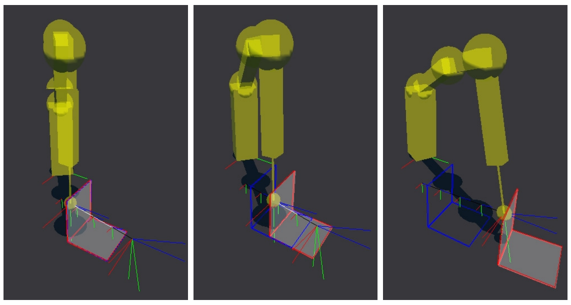


Figure 9. Prediction of the sliding affordance.

8 Conclusions and future work

The experiments shown in this paper demonstrate the ability of recurrent neural networks, in particular Long Short-Term Memory machines to approximate a regression function encoding the trajectory of simple geometrical objects when pushing actions are performed. Therefore, these machines are useful for predicting the affordances of pushing actions. We used 3-dimensional features and realistic simulations that we can then apply to real environments. Sequences of finger effector and polyflap poses were used to feed the LSTMs, showing the capacity of LSTM for prediction in relatively large time periods. The offline experiments showed great accuracy in prediction. The use of an active learning mechanism where machines are specialized in different parts of the sensorimotor space was also tested. The selection of actions is performed via a measure of learning progress that improves generalization.

In this work, the motivation to select an action via active learning is mainly based on the curiosity-driven mechanism introduced by the IAC algorithm. This mechanism forces the robot to select actions that maximize a learning progress measure. This encourages the reduction of error for sensorimotor regions that are still not accurately learned. The effectivity of the IAC-based strategy for assessing learning progress in sensorimotor regions with spatio-temporal features is confirmed. Moreover, machines that take into account spatio-temporal information fit well into the active learning loop. However, the gradient-based method for updating the networks still makes the process slow, so that many iterations are needed to observe high improvements. It is possible to add additional drives or measures for selecting actions in order to have different strategies for accelerating the learning progress. Additionally, alternative algorithms for LSTM training may also be considered.

The CrySSMEx[8] algorithm has been used to analyse recurrent networks as dynamical systems by using a conditional Entropy based method that extracts a probabilistic automaton associated to a machine. This method might be useful for active learning, because it represents uncertainty and predictability during the processing of spatio-temporal features.

REFERENCES

[1] Bram Bakker, ‘Reinforcement learning with long short-term memory’, in *Advances in Neural Information Processing Systems 14*, pp. 1475–1482. MIT Press, (2002).

[2] Paul Fitzpatrick, Giorgio Metta, Lorenzo Natale, Sajit Rao, and Giulio Sandini, ‘Learning about objects through action - initial steps towards artificial cognition’, in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3140–3145, (2003).

[3] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins, ‘Learning to forget: Continual prediction with lstm’, *Neural Computation*, **12**, 2451–2471, (1999).

[4] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber, ‘Learning precise timing with lstm recurrent networks’, *J. Mach. Learn. Res.*, **3**, 115–143, (2003).

[5] J. J. Gibson, ‘The theory of affordances’, in *Perceiving, Acting, and Knowing: Toward an Ecological Psychology*, eds., R. Shaw and J. Bransford, 67–82, Lawrence Erlbaum, (1977).

[6] Alex Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, Ph.D. dissertation, Technische Universität München, July 2008.

[7] Sepp Hochreiter and Jurgen Schmidhuber, ‘Long short-term memory’, *Neural Computation*, 1735–1780, (1997).

[8] H. Jacobsson, ‘The crystallizing substochastic sequential machine extractor - `CrySSMEx`’, *Neural Computation*, **18**(9), 2211–2255, (2006).

[9] M. Kopicki, J. Wyatt, and R. Stolkin, ‘Prediction learning in robotic pushing manipulation’, in *Proceedings of the 14th IEEE International Conference on Advanced Robotics (ICAR 2009)*, Munich, Germany, (June 2009).

[10] P-Y. Oudeyer, F. Kaplan, and V. V. Hafner, ‘Intrinsic motivation systems for autonomous mental development’, *IEEE Transactions on Evolutionary Computation*, **11**(1), (2007).

[11] L. Paletta, G. Fritz, F. Kintzler, J. Irran, and G. Dorffner, ‘Learning to perceive affordances in a framework of developmental embodied cognition’, in *Development and Learning, 2007. ICDL 2007. IEEE 6th International Conference on*, pp. 110–115, (July 2007).

[12] B. Ridge, D. Skočaj, and A. Leonardis, ‘A system for learning basic object affordances using a self-organizing map’, in *International Conference on Cognitive Systems CogSys 2008*, Karlsruhe, Germany, (2008).

[13] Sergio Roa and Geert Jan Kruijff, ‘Long short-term memory for affordances learning’, in *Proceedings of the 9th International Conference on Epigenetic Robotics*, eds., Lola Caamero, Pierre-Yves Oudeyer, and Christian Balkenius, number 146 in Lund University Cognitive Studies, pp. 235–236. o.A., (11 2009).

[14] Sergio Roa, Geert Jan Kruijff, and Henrik Jacobsson, ‘Curiosity-driven acquisition of sensorimotor concepts using memory-based active learning’, in *Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics*, pp. 665–670, (2008).

[15] *Towards Affordance-Based Robot Control - LNAI 4760*, eds., Erich Rome, Joachim Hertzberg, and Georg Dorffner, Lecture Notes in Computer Science LNAI 4760, Springer Verlag, Berlin, Germany, 2008.

[16] M. Schuster and K.K. Paliwal, ‘Bidirectional recurrent neural networks’, *Signal Processing, IEEE Transactions on*, **45**(11), 2673–2681, (Nov 1997).

[17] Aaron Sloman, ‘Polyflaps as a domain for perceiving, acting and learning in a 3-D world’, in *Position Papers for 2006 AAAI Fellows Symposium*, Menlo Park, CA, (2006). AAAI.

Binding and Cross-modal Learning in Markov Logic Networks

Alen Vrečko, Danijel Skočaj, Aleš Leonardis

Faculty of Computer and Information Science,
University of Ljubljana, Slovenia
`{alen.vrecko, danijel.skocaj, ales.leonardis}@fri.uni-lj.si`

Abstract. Binding — the ability to combine two or more modal representations of the same entity into a single shared representation is vital for every cognitive system operating in a complex environment. In order to successfully adapt to changes in an dynamic environment the binding mechanism has to be supplemented with cross-modal learning. In this paper we define the problems of high-level binding and cross-modal learning. By these definitions we model a binding mechanism and a cross-modal learner in Markov logic network and test the system on a synthetic object database.

1 Introduction

One of the most important abilities of any cognitive system operating in real world environment is to relate and merge information from different modalities. For example, when hearing a sudden, unexpected sound, humans automatically try to visually locate its source in order to relate the audio perception of the sound to the visual perception of the source. The process of combining two or more modal representations (grounded in different sensorial inputs) of the same entity into a single multimodal representation is called *binding*. While the term *binding* has many different meanings across various scientific fields, a very similar definition comes from the neuroscience, where it denotes the ability of the brain to converge perceptual data processed in different brain parts and segregate it in distinct elements [2] [14].

The binding process can operate on different types and levels of cues. In the above example the direction that the human percieves the sound from is an important cue, but sometimes this is not enough. If there are several potential sound sources in the direction of the percept, the human may have to relate higher level audio and visual properties. A knowledge base that associates the higher level perceptual features across different modalities is therefore critical for a successful binding process in any cognitive system.

In order to function properly in a dynamic environment, a cognitive system should also be able to learn and adapt in a continuous, open-ended manner. The ability to online update the cross-modal knowledge base, i. e. *cross-modal learning*, is therefore vital for any kind of binding process in such an environment.

Many of the past attempts of binding information within cognitive systems were restricted to associate linguistic information to lower level perceptual information. Roy et al. tried to ground the linguistic descriptions of objects and actions in visual and sound perceptions and to generate descriptions of previously unseen scenes based on the accumulated knowledge [12] [13]. This is essentially a *symbol grounding problem* first defined by Harnad [6]. Chella et al. proposed a three-layered cognitive architecture around the visual system with the middle, *conceptual layer* bridging the gap between linguistic and sub-symbolic (visual) layers [4]. Related problems were also often addressed by Steels [15].

Jacobsson et al. approached the binding problem in a more general way [8] [7] developing a cross-modal binding system that could associate between multiple modalities and could be part of a wider cognitive architecture. The cross-modal knowledge was represented as a set of binary functions comparing binding attributes in pair-wise fashion. A cognitive architecture using this system for linguistic reference resolution was presented in [16]. This system was capable of learning visual concepts in interaction with a human tutor. Recently, a probabilistic binding system was developed within the same group that encodes the cross-modal knowledge into a Bayesian graphical model [17]. The need for a more flexible, but still probabilistic representation of the cross-modal knowledge led our reasearch efforts in the direction of Markov graphical models, as presented in this paper.

In the next section we define the problems of cross-modal learning and binding. In section 3 we first briefly describe the basics of *Markov logic networks (MLNs)*. Then we desribe our binding and cross-modal learning model that is based on MLNs. Section 4 describes the experiments performed on the prototype system. We end the paper with the conclusion and future work.

2 The problem definition

The main idea of cross-modal learning is to use successful bindings of modal percepts as learning samples for the cross-modal learner. The improved cross-modal knowledge base thus enhances the power of the binding process, which is then able to bind together new combinations of percepts, i. e. new learning samples for the learner. For example, if a cognitive system is currently capable of binding an utterance describing something blue and round to a perceived blue ball only by color association, this particular instance of binding could teach the system to associate also the visual shape of the ball to linguistic concept of roundness. We see that at least on this level the binding process depends on the ability to associate between *modal features* (in this example the visual concepts of blue and shape are features of visual modality, while the linguistic concept of blue and ball belong to the linguistic modality).

We assume an open world in terms of modal features (new features can be added, old retracted). The cross-modal learner starts with just a small prior knowledge of how to associate between few basic features, which is then gradually expanded to the other features and the new ones that are created.

High level cross-modal learning problem is closely related to the *association rule learning problem* in data mining, which was first defined by Agrawal et al. [1]. Therefore, we will base our learning problem definition on Agrawal's definition and expand it with the notion of *modality.*

We have a set of n binary attributes called features $F = \{f_1, f_2, ..., f_n\}$ and a set of rules called knowledge database $K = \{t_1, t_2, ..., t_m\}$. A rule is defined as an implication over two subset of features:

$$t_i : X \Rightarrow Y \tag{1}$$

where $X, Y \subseteq F$ and $X \cap Y = \emptyset$. The features represent various higher level modal properties based on the sensorial input. We introduce the notion of modality to our definition — each feature is restricted to one modality only:

$$\begin{aligned} M_1 &= \{f_{11}, f_{12}, ..., f_{1n_1}\} \\ M_2 &= \{f_{21}, f_{22}, ..., f_{2n_2}\} \\ &\vdots \vdots \vdots \vdots \vdots \\ M_k &= \{f_{k1}, f_{k2}, ..., f_{kn_k}\} \end{aligned} \tag{2}$$

$$F = M_1 \cup M_2 \cup ... \cup M_k.$$

We modify the rule-making restrictions of (1) accordingly:

$$\begin{aligned} &1.\ N = M_{m_1} \cup M_{m_2} \cup ... \cup M_{m_r},\quad m_1, ..., m_r \in \{1, 2, ..., k\}, r < k \\ &2.\ Y \subseteq N \\ &3.\ X \subseteq F \setminus N \end{aligned} \tag{3}$$

Next, we define the binding problem. *Percepts* are collections of features from a single modality. A percept acts as a modal representation of a percieved entity. Let $\mathbf{P}$ be the set of current percepts – *the percept configuration*:

$$\mathbf{P} = \{P_1, P_2, ..., P_n\},\ P_i \subseteq M_j. \tag{4}$$

Unions are collections of percepts from different modalities. An union acts as a shared representation of a percieved entity, grounded through its percepts to different modalities. Let $\mathbf{U}$ be the set of current binding unions – *the union configuration*:

$$\mathbf{U} = \{U_1, U_2, ..., U_m\},\ U_i \subseteq \mathbf{P}. \tag{5}$$

The set of possible *binding functions* is then defined as

$$\beta : \mathbf{P} \rightarrow \mathbf{U}, \tag{6}$$

where the following restrictions apply:

$$\begin{aligned}&1.\ U_1 \cup U_2 \cup ... \cup U_m = \mathbf{P}\\ &2.\ \forall U_i, U_j \in \mathbf{U}, i \neq j : U_i \cap U_j = \emptyset\\ &3.\ \forall P_i, P_j \in U_k, i \neq j : P_i \subseteq M_l \wedge P_j \subseteq M_m \Rightarrow l \neq m.\end{aligned} \tag{7}$$

The first two restrictions assign each percept in the configuration to exactly one union, while the third restricts the number of percepts per modality in an union to one.

We also introduce a measure of confidence in an union configuration based on the current knowledge base K – *the binding confidence* $\mathrm{bconf_K}(\mathbf{U})$. Given the percept configuration $\mathbf{P}$ and the set of possible binding functions β, the task of the binding process is to find the optimal union configuration:

$$\mathbf{U}_{opt} = \underset{\mathbf{U}=\beta(P)}{\mathrm{argmax}}(\mathrm{bconf_K}(\mathbf{U})). \tag{8}$$

In this sense — i. e. considering $\mathrm{bconf_K}(\mathbf{U})$ as a predictor based on K — we can consider the high-level cross-modal learning a regression problem. Therefore, the aim of the cross-modal learner is to maintain and improve the cross-modal knowledge base, thus providing an increasingly more reliable measure of binding confidence.

3 Implementation in MLN

The *Markov logic networks*[1] [10] combines first-order logic and probabilistic graphical models in a single representation. A MLN knowledge base consists of a set of first-order logic formulae (rules) with a weight attached:

$$\textit{weight} \quad \textit{first-order logic formula}$$

The weight is a real number, which determines how strong a constraint each rule is, the higher the weight — less probable the world violating that rule. Together with a finite set of constants the MLN defines a *Markov network (MN)* (or *Markov random field*). A MN is an undirected graph where each possible grounding of a predicate (all predicate variables replaced with constants) represents a node, while the rules define the edges between the nodes. Each rule grounding defines a clique in the graph. A MLN can be thus viewed as a template for constructing the MN. The probability distribution over possible worlds

[1] We used *Alchemy* [9] to implement the prototype of our crossmodal binding and cross-modal learning mechanisms. Alchemy is a software package providing various inference and learning algorithms based on MLN.

x defined by a MN is given by

$$P(X = x) = \frac{1}{Z} \exp\big(\sum_i w_i n_i(x)\big), \tag{9}$$

where $n_i(x)$ is the number of true groundings of the rule i, w_i is the weight of the rule i, while Z is the partition function defined as $Z = \sum_x \exp\big(\sum_i w_i n_i(x)\big)$.

The inference in MN is P#-complete problem [11]. Methods for approximating the inference include various *Markov Chain Monte Carlo* sampling methods [5] and *belief propagation* [18].

3.1 Cross-modal knowledge base

We have two types of templates for the binding rules in our cross-modal knowledge base. The template for the *aggregative rule* is defined as

$$\text{perPart}(p_1, f_1) \wedge \text{uniPart}(u, p_1) \wedge \text{perPart}(p_2, f_2) \Rightarrow \text{uniPart}(u, p_2), \tag{10}$$

where the predicate perPart($percept, feature$) denotes that the feature $feature$ is part of the percept *percept*, while uniPart($union, percept$) denotes that *union* includes *percept*. In a very similar manner the template for the *segregative rule* is defined:

$$\text{perPart}(p_1, f_1) \wedge \text{uniPart}(u, p_1) \wedge \text{perPart}(p_2, f_2) \Rightarrow \neg\text{uniPart}(u, p_2). \tag{11}$$

If by the aggregative rules the percepts are merged in common unions, the segregative rules separate them in distinct unions. The template rules are equivalent to a subset of association rules (1), where each side is limited to one feature. We also define the binding domain that we will use to ground the network. An example of binding domain with two modalities:

$$\begin{aligned} modality &= \{Language, Vision\} \\ feature &= \{Red, Green, Blue, Compact, Flat, Elongated, \\ &\quad\; Color1, Color2, Color3, Shape1, Shape2, Shape3\}. \end{aligned} \tag{12}$$

Based on this example domain a small set of grounded and weighted binding rules could look like this:

$$\begin{aligned} &2.5 \quad \text{perPart}(p_1, Red) \wedge \text{uniPart}(u, p_1) \wedge \text{perPart}(p_2, Color1) \Rightarrow \text{uniPart}(u, p_2) \\ &1.9 \quad \text{perPart}(p_1, Red) \wedge \text{uniPart}(u, p_1) \wedge \text{perPart}(p_2, Color2) \Rightarrow \neg\text{uniPart}(u, p_2), \end{aligned} \tag{13}$$

The predicates forming the binding rules are not fully grounded yet. They are only grounded on conceptual level, with the known features like Red, Color1, etc., while the instances (objects) are still represented with the variables. The predicates get fully grounded each time an inference is performed, when based on the current situation (e. g. perceived objects that form the scene) a MN is constructed. This principle could be very beneficial for a cognitive system, since it decouples the general from specific, but allows to apply and adapt the general concepts learnt over longer periods of time to the current situation in a very flexible fashion.

Using the example domain in (12) an example percept configuration could look like

$$\begin{aligned}&\text{perPart}(1, \mathit{Color}1) \wedge \text{perPart}(1, \mathit{Shape}2) \wedge \\ &\text{perPart}(2, \mathit{Color}2) \wedge \text{perPart}(2, \mathit{Shape}3) \wedge \\ &\text{perPart}(3, \mathit{Red}), \end{aligned} \tag{14}$$

From (13) and (14) we could infer the following union configuration:

$$\text{uniPart}(1, 1) \wedge \text{uniPart}(1, 3) \wedge \text{uniPart}(2, 3).$$

Besides the binding rules, our database contains also features' priors in the following form:

$$\mathit{weight} \quad \text{perPart}(p, \mathit{ColorGrounding})$$

A feature prior denotes the default probability of a feature belonging to a percept (if there is no positive or negative evidence about it).

In addition, we use a special predicate to determine the partition of features between modalities in the sense of (2) (e. g. $modPart(Language, Red)$, $modPart(Vision, Color1)$.

3.2 Learning

After the rules and priors are grounded within the binding domain, we need to learn the weights for each rule and for each prior grounding. We use the generative learning method described in [10]. The learner computes a gradient from the weights based the number of true groundings ($n_i(x)$) in the learning database and the expected true groundings according the MLN ($E_w[n_i(x)]$):

$$\frac{\delta}{\delta w_i} \log P_w(x) = n_i(x) - E_w[n_i(x)], \tag{15}$$

and optimizes the weights accordingly. Since the expectations $E_w[n_i(x)]$ are very hard to compute, the method uses the *pseudo-likelihood* to approximate it [3].

Continuous learning is performed by feeding the learning samples to the system in small batches (3-6 examples of unions). Each mini batch represents a scene the system has resolved, described with perPart and uniPart predicates. In each learning step the learner accepts the rule's old weight in the knowledge database as the mean for the Gaussian prior, which it tries to adjust based on the new training mini batch. By setting the dispersion of the weight's Gaussian prior to an adequate value, we ensure the learning rate of each mini batch is proportional to the batch size.

3.3 The binding process

The binding is performed as inference over the knowledge base based on some evidence. In order for the binding inference to function properly we have to define some hard rules (formulae with infinite weight) that apply the restrictions of the binding functions β in (7):

1. $\forall p \exists u : \text{uniPart}(u, p)$
2. $\text{uniPart}(u1, p) \wedge \text{uniPart}(u2, p) \Rightarrow u1 = u2$
3. $\text{perPart}(p1, f1) \wedge \text{perPart}(p2, f2) \wedge (p1 \neq p2) \wedge modPart(m, f1)$
   $\wedge\, modPart(m, f2) \wedge \text{uniPart}(u, p1) \Rightarrow \neg\text{uniPart}(u, p2)$.

Usually the inference consists of querying for the predicate uniPart, where the evidence typically includes the description of the current percept configuration (using the predicate perPart), a list of known and potential unions and the description of the current partial union configuration (some percepts are already assigned to known unions). The binding result is expressed as a probability distribution for each unassigned percept over the known and potential unions.

4 Experimental results

We experimented with our system on a database of 42 synthetic objects. Objects had percepts from three modalities: vision, language and affordance. Visual modality had in total 13 features: 6 for object color and 7 for the shape. Language had 13 features that matched the visual features and 8 features for object type (e. g. book, box, apple). The affordance modality had 3 features describing the possible outcomes of pushing an object. Mini batches were designed to mimic robot interaction with a human tutor, where the tutor showed objects to the robot, describing their properties. Typically a mini batch contained 5-6 objects. The learning sequence consisted of 80 mini batches.

We designed 30 test-cases for evaluating the binding process. In each test-case we had three visual percepts and one non-visual percept. The binder had to determine whether and which visual percept the the non-visual belonged to (i. e. four possible choices: one for each visual percepts and one for no corresponding

$union = \{1,\ 2,\ 3,\ 4\}$
perPart$(1, VisRed)$, perPart$(1, VisFlat)$, perPart$(1, VisCylindrical)$
perPart$(2, VisBlue)$, perPart$(2, VisCompact)$, perPart$(2, VisSpherical)$
perPart$(3, VisGreen)$, perPart$(3, VisElongated)$, perPart$(3, VisConical)$
uniPart$(1, 1)$, uniPart$(2, 2)$, uniPart$(2, 2)$

perPart$(4, LinRed)$, perPart$(4, LinFlat)$, perPart$(4, LinCylindrical)$
uniPart$(\mathrm{u}, 4)$?

Fig. 1. An example of an easy test-case. We can see that objects represented with visual percepts (1,2 and 3) differ in all types of visual features. The system needs to determine which union the fourth, linguistic percept belongs to.

$union = \{1,\ 2,\ 3,\ 4\}$
perPart$(1, VisRed)$, perPart$(1, VisCompact)$, perPart$(3, VisConical)$
perPart$(2, VisGreen)$, perPart$(2, VisCompact)$, perPart$(2, VisSpherical)$
perPart$(3, VisGreen)$, perPart$(3, VisFlat)$,
uniPart$(1, 1)$, uniPart$(2, 2)$, uniPart$(2, 2)$

perPart$(4, LinApple)$
uniPart$(\mathrm{u}, 4)$?

Fig. 2. An example of an difficult test-case. We can see that the objects represented with visual percepts (1,2 and 3) are less distinct than in the easier test-case (fig. 1) and with some incomplete information. The system has to find out which visual percept could be an apple. The visual training samples for apples consisted of compact and spherical percepts of red or green color.

percept). Of the four possible choices there was always one that was more obvious than the others and deemed correct. The possibility that the system inferred as the most probable, was considered as its choice for the binding union. The test-cases varied in their level of difficulty: the easiest featured distinct features for visual percepts and complete information for all percepts (all percepts had a value for each feature type belonging to its modality, see figure 1), while more difficult cases could have features shared by more percepts and incomplete percept information (see figure 2). The tests were performed several times during the learning process in intervals of four batches.

Figure 3 shows the average success rate over 10 randomly generate learning sequences. We see that with the growing number of samples the binding rate

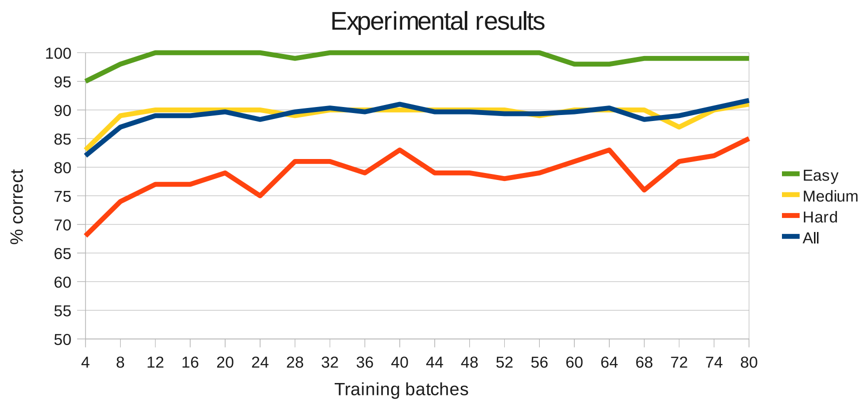


Fig. 3. Experimental results: the average rate of correct bindings relative to the number of training batches (10 randomly generated learning sequences were used). The green, yellow and red lines denote the easy, medium and hard test samples respectively, while the blue line denotes the overall success percentage.

tends to grow and converge, though with some oscillations. The oscillations for the difficult samples are more pronounced as for the easier samples. Analyzing the results example by example we can see that the test-cases with the most oscillations were the ones that depended on *many-to-one feature associations* (e. g. *red*, *compact*, *cylindrical* $\Rightarrow$ *cola can*). This can be explained with the current structure of our binding rules that directly support *one-to-one feature associations* only.

5 Conclusion

In this paper we defined the problems of high-level binding and cross-modal learning. By these definitions we modeled a binding mechanism and a cross-modal learner in MLNs. We tested the system on a synthetic object database and shown how the binding power of the system increases with the number of learnt samples.

In the future we will apply our binding and cross-modal learning models to a real cognitive architecture that includes visual and communication subsystems, thus gaining a platform for experiments on real-world data. We will also extend the structure of our database to more complex rules (or perhaps include a structure learning mechanism to our system) and improve and extend our experiments to better simulate the robot-tutor interaction.

References

1. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216, 1993.
2. A. Bartels and S. Zeki. The temporal order of binding visual attributes. *Vision research*, 46(14):2280–2286, 2006.
3. J. Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society. Series D (The Statistician),*, 24(3):179–195, 1975.
4. A. Chella, M. Frixione, and S. Gaglio. A cognitive architecture for artificial vision. *Artif. Intell.*, 89(1-2):73–111, 1997.
5. W. R. Gilks and D. J. Spiegelhalter. *Markov chain Monte Carlo in practice.* Chapman & Hall/CRC, 1996.
6. S. Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42:335–346, 1990.
7. H. Jacobsson, N. Hawes, G-J. Kruijff, and J. Wyatt. Crossmodal content binding in information-processing architectures. In *Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Amsterdam, The Netherlands, March 12–15 2008.
8. H. Jacobsson, N. Hawes, D. Skočaj, and G-J. Kruijff. Interactive learning and cross-modal binding - a combined approach. In *Symposium on Language and Robots*, Aveiro, Portugal, 2007.
9. S. Kok, M. Marc Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, J. Wang, and P. Domingos. The alchemy system for statistical relational ai. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2009.
10. M. Richardson and P. Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, 2006.
11. D. Roth. On the hardness of approximate reasoning. *Artif. Intell.*, 82(1-2):273–302, 1996.
12. D. Roy. Learning visually-grounded words and syntax for a scene description task. *Computer Speech and Language*, 16(3-4):353–385, 2002.
13. D. Roy. Grounding words in perception and action: computational insights. *TRENDS in Cognitive Sciences*, 9(8):389–396, 2005.
14. W Singer. Consciousness and the binding problem. *Annals of the New York Academy of Sciences*, 929:123–46, 2001.
15. L. Steels. *The Talking Heads Experiment. Volume 1. Words and Meanings.* Laboratorium, Antwerpen, 1999.
16. A. Vrečko, D. Skočaj, N. Hawes, and A. Leonardis. A computer vision integration model for a multi-modal cognitive system. In *The 2009 IEEE/RSJ International Conference on Intelligent RObots and Systems*, pages 3140–3147, St. Louis, MO, USA, October 2009.
17. J. Wyatt, A. Aydemir, M. Brenner, M. Hanheide, N. Hawes, P. Jensfelt, M. Kristan, G-J. Kruijff, P. Lison, A. Pronobis, K. Sjöö, D. Skočaj, A. Vrečko, H. Zender, and M. Zillich. Self-understanding & self-extension: A systems and representational approach, 2010. Accepted for publication.
18. J. S. Yedidia, W. T. Freeman, and Y. Weiss. *Understanding belief propagation and its generalizations.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

CROSS-MODAL LEARNING

Danijel Skočaj
Aleš Leonardis
Faculty of Computer and Information Science
University of Ljubljana
Ljubljana, Slovenia
{danijel.skocaj,ales.leonardis}@fri.uni-lj.si

Geert-Jan M. Kruijff
Language Technology Lab
German Research Center for Artificial Intelligence
Saarbrücken, Germany
gj@dfki.de

Synonyms

Multi-modal learning

Definition

Cross-modal learning refers to any kind of learning that involves information obtained from more than one modality. In the literature the term modality typically refers to a sensory modality, also known as stimulus modality. A stimulus modality provides information obtained from a particular sensorial input, for example visual, auditory, olfactory, or kinesthetic information. Examples from artificial cognitive systems ("robots") include also information about detected range (by sonar or laser range-finders), movement (by odometry sensors), or motor state (by proprioceptive sensors). We adopt here a notion of modality that includes both sensorial data, and further interpretations of that data within the modality. For example, from a pair of (depth-calibrated) images, a cloud of points in 3-dimensional space can be computed. We obtain both types of data (the image data, and the 3D points) from the same visual sensor. At the same time, they differ in what information they provide. We consider information sources derived from sensorial data as derived modalities that by themselves can be involved again in cross-modal learning.

Theoretical Background

We distinguish different types of cross-modal learning. The distinction is based on how the learnt model depends on the data from several modalities, and to what degree the model integrates information from these modalities.

In weakly-coupled cross-modal learning, models are built within individual modalities, while the rather high-level information from other modalities, such as a label or reinforcement signal, supervise the internal learning process. Inference based on the learned models can be done on the basis of a single modality, or the output of several modalities can be combined to achieve better performance or robustness. One example of this is speech recognition. In situated dialogue, recognizing sequences of words in an audio signal can be greatly improved by information about the situated context (what is there to be seen, what is there to be done, what have we talked before), and through observation of the speaker. Context and observations aid disambiguation during processing of the auditory signal, possibly also correcting misheard words, or filling in (grammatically) missing words. For example, lip reading can greatly aid recognition. In a continuous learning process, successfully recognized lip poses can supervise learning of audio-based speech recognition ability. The other way round, correctly recognized audio input can provide labels to aid the learning of lip reading. This process of coupled supervision during learning is also known as co-learning. In the end we obtain two classifiers, one in each modality, that can be used individually, or they can be combined to further increase the success of speech recognition. This type of cross-modal learning is thus based on a weakly coupled interaction of data from different modalities, which is done on a rather high level of abstraction. In the case above we assumed that both modalities mutually drive the learning in the other one. This process can in principle be uni-directional. If the information in one modality is much more reliable, it can drive the learning in another modality.

In closely-coupled cross-modal learning, learning processes are more intertwined. A model is learnt by combining information from different modalities into a common level of representation, and then using this level as a starting point to build a common cross-modal classifier or predictor. As a result, inference with the acquired model requires information coming from several modalities, and cannot be achieved within a single modality only. This approach is often used in sensorimotor learning. Here, low-level features from a visual modality and motor (or proprioceptive, or haptic) modalities are merged. Based on the obtained cross-modal features, higher-level sensorimotor concepts are learned. For example, from low-level visual features describing objects and low-level features parametrizing actions that could be performed, a model is learned that predicts what happens with a particular object if a particular action is applied (through classification or regression). A similar case is also when the feature vectors still reside in the individual modalities, but we construct several intermediate classifiers, which are no longer independent, and combine them. This requires a close-coupling of semi-synchronous learning processes, based on interconnected representations, and leading up to the formation of cross-modal concepts.

We can also identify a third type of cross-modal learning that is performed on a higher level of abstraction. Here, a model is acquired that connects modal conceptual structures from different modalities by learning associations between them. For example, let us suppose that we want to recognize a cup of coffee. A cup can be recognized visually. Yet, to recognize what is inside the cup we need another sense - smell. We need to combine information from both modalities to determine that there is a cup of coffee on the table and not a cup of a black tea. The learning of required concepts could be performed largely independently, in each modality individually. At some point though we need to learn to combine the concepts of the cup and the coffee into a concept of a "cup of coffee". The final representation therefore consists of representations from several modalities.

Cross-modal learning is related to principles of fusion of data from different sensors (Clark and Yuille), also known as multi-sensory processing in natural cognitive systems (Stein and Meredith). Different processes interact in a cognitive system to form a coherent interpretation of experience, based on the combination of information obtained through several modalities. The process of learning how to combine this information is a kind of cross-modal learning.

As already mentioned, we can consider the term modality in its wider sense. This includes derived modalities. In this case, the type of information that characterizes a modality is not attached directly to a sensor, but to a process which interprets the sensorial data. For example, suppose that we have a place recognition approach that is based on both visual images, and 3D point clouds representing geometrical structure. The images may be obtained using a camera. The 3D point clouds are obtained using a laser range finder, or, alternatively, both the images and the 3D data can be obtained using a stereo rig. In both cases we can conceive of the learning of representations of places as a kind of cross-modal learning, although in the second case we have one sensor only. In computer vision, it is very often favorable to extract several visual cues (such as color, texture, borders, shape, motion), and combine them in order to obtain better classifiers. We can look at the learning of such combined classifiers as at a kind of cross-modal learning as well.

The relevance of cross-modal learning is alike for natural and artificial cognitive systems (Christensen et al). Both continuously learn, to extend their knowledge of acting in dynamic environments. The ability to connect possibly asynchronously developed models across different modalities provides an important basis for a grounded form of self-understanding. The possibility to interconnect and thus form an interpretation that is coherent across multiple modalities indicates what is known relative to some experience. Failure to do so may indicate a kowledge gap, and can function as a trigger for self-aware learning.

Important Scientific Research and Open Questions

There are arguments for learning to be based on association, and for learning to be mediated by a (developing) categorical system. Very often, the interconnectivity between modalities is mediated by categorical structure. Effectively this establishes a triadic relation between modalities. The conceptual structures in the modalities can be connected because they can be understood as related by virtue of their reference to a shared categorical ground. The arguments for this type of learning, based on the formation of a mediating categorical structure, arise from for example childhood cognitive development. In word learning it is shown that a purely associative, unmediated account ("child-as-data-analyst") cannot appropriately account for categorical generalizations a child is able to make ("child-as-theorist"). The use of mediating categories both helps generalization of sensory input beyond actual experience, and allows for representations to be ultimately grounded in, and influenced by, the embodiment of the system (G. Lakoff & M. Johnson). On the other hand, in many cases the modalities interact on a much lower level, like in the case of sensorimotor learning. It still an open question what roles do these different forms of learning play in specific types of cross-modal learning, whether in natural or artificial cognitive systems.

A fundamental aspect of embodied cognition is that understanding is ultimately based in how a cognitive system experiences the world. Since the cross-modal learning is based on processing and relating information from several (sensory) modalities it may play an important role in bringing about grounded forms of cognition.

We also have to address the terminological issues, since the terms related to cross-modal learning are not consistently used in the literature. Sometimes, the term cross-modal learning is used only to refer to strongly-coupled types of cross-modal learning. Also, the term modality is sometimes used in its narrower sense, considering sensory modalities only. Here, we adopted the broader meaning of both terms. There is also another term in the literature that is often used to describe a similar phenomenon, the term multi-modal learning. One meaning of this term refers to (human) learning based on different multimedia material involving different human senses that facilitate learning. The second meaning of this term is very close to the meaning of cross-modal learning as defined above. Sometimes this term relates to forms of weakly-coupled cross-modal learning, while very often cross-modal and multi-modal learning are used interchangeably with the same meaning (synonyms).

Cross-References

→ Co-learning

→ Active learning

→ Adaptation and learning

→ Learning and understanding

→ Data fusion

→ Cognitive models of learning

→ Cognitive robotics

→ Embodied cognition

References

1. B.E. Stein and M.A. Meredith. The Merging Of The Senses. MIT Press. 1993.
2. G. Lakoff and M. Johnson. Philosophy In The Flesh: the Embodied Mind and its Challenge to Western Thought. Basic Books. 1999.
3. H.I. Christensen, G.J.M. Kruijff and J.L.Wyatt. Cognitive Systems. COSMOS 8. Springer Verlag, 2010.
4. D. Philipona, K. O'Regan and J.P. Nadal. "Is there something out there? Inferring space from sensorimotor dependencies." Neural Computation 15(9):2029-2049. 2003.
5. J. J. Clark, and A. L. Yuille. Data Fusion for Sensory Information Processing Systems. Kluwer Academic Publishers, Norwell, MA. 1990.
6. V.R. de Sa and D. Ballard. "Category Learning through Multi-Modality Sensing". Neural Computation 10(5): 1097-1117. 1998.