Autonomous generation of internal representations for associative learning

Michaël Garcia Ortiz, Benjamin Dittes, Jannik Fritsch, and Alexander Gepperth

michael.garcia.ortiz@gmail.com, benjamin.dittes@honda-ri.de, jannik.fritsch@honda-ri.de, alexander.gepperth@honda-ri.de

Honda Research Institute Europe GmbH
Carl-Legien-Str.30, 63073 Offenbach,
Germany

CoR-Lab, Bielefeld University
Universitatsstr. 25, 33615 Bielefeld,
Germany

Abstract. In this contribution, we explore the possibilities of learning in large-scale, multimodal processing systems operating under real-world conditions. Using an instance of a large-scale object detection system for complex traffic scenes, we demonstrate that there is a great deal of very robust correlations between high-level processing results quantities, and that such correlations can be autonomously detected and exploited to improve performance. We formulate requirements for performing *system-level learning* (online operation, scalability to high-dimensional inputs, data mining ability, generality and simplicity) and present a suitable neural learning strategy. We apply this method to infer the identity of objects from multimodal object properties ("context") computed within the correlated system and demonstrate strong performance improvements as well as significant generalization. Finally, we compare our approach to state-of-the-art learning methods, Locally Weighted Projection Regression (LWPR) and Multilayer Perceptron (MLP), and discuss the results in terms of the requirements for system-level learning.

1 Introduction

In contrast to many previous approaches focussing on learning close to sensory signals [1,2], this contribution supports an alternative view, namely that learning in large-scale environment perception systems gets more feasible and beneficial when applied to heavily preprocessed, abstract/invariant quantites to which we will summarily refer to as *system-level quantities*. We propose that this kind of *system-level learning* has, by construction, a high generalization ability which supports system performance particularly in difficult sensory conditions (high-dimensionality, noise, ambiguity, ...).

This study is conducted based on an instance of a large-scale object detection system in road traffic environments [3] which integrates multimodal information (laser, video) as well as a wide variety of vision-based algorithms such as stereo, tracking, classification, and free-area detection. The motivation for this study arose when trying to obtain multimodal object models (in this case: car models) for excluding obviously incorrect object detections.

1.1 Requirements for learning strategies

For being applicable for system-level learning in a large-scale system, any learning method must fulfill several requirements:

Online regression, i.e, the ability to train and perform regression (in contrast to binary classification) within a running system, which implies that the number of training examples is not known in advance, while avoiding catastrophic forgetting [4] when new examples are presented to an already trained system.

Generality and simplicity, a requirement which concerns the applicability of an algorithm to a very wide range of data. Thus, e.g., support vector machines with problem-specific kernels such as, e.g. [5], are inapplicable. Any system-level learning method must also be *simple* in the sense that it does not contain a large number of parameters that must be tuned to problem-specific values.

Scalability. Using a simple set of system-level learning methods requires the conversion of system-level quantities into a common representational format (see [6]). Such a format should not be optimized for a specific kind of data, and should give the possibility of repesenting probabilistic information. Therefore, any system-level learning algorithm must cope with the fact that the number of data dimensions can grow very large.

Data mining ability. A learning algorithm must be able to ignore irrelevant and possibly noisy dimensions and to approximately identify the relevant ones, especially in high-dimensional data.

Reusability of internal representations. Any advanced learning algorithm constructs internal representations of the input data, e.g. the hidden layers of a MLP or the set of support vectors constructing the separating hyperplane of a support vector machine. Internal representations should not be task-specific, but should be reusable for other tasks if possible.

1.2 Neural projection-prediction as a system-level learning strategy

We present a first instance of an extensible, composite neural method able to exploit correlations and interrelationships between system-level quantities, especially higher-order correlations not directly available to direct associative learning. This contribution focuses on the autonomous generation of internal representations by the proposed *neural projection-prediction* (NPP) method. We demonstrate the benefits and limitations of our composite method by rigorously benchmarking it. For this purpose, we employ data in the form of system-level quantities recorded from a large-scale real-world system dedicated to object detection in challenging traffic environments. In order to realistically assess the value of NPP as a system-level learning method, we evaluate two other learning methods on the benchmark task: a nonlinear MLP and the LWPR [7].

1.3 Related Work

LWPR [7] is explicitly designed to be an online method for high-dimensional data, and it avoids catastrophic forgetting by incrementally partitioning the input space into volumes where individual linear regression models are applied. Here, we evaluate the performance of LWPR w.r.t. suitability for system-level learning. We particularly intend to determine whether LWPR can deal with the requirements of scalability and generality (sec. 1.1).

Another related model is the Radial Basis Function network (RBFN) model (see, e.g., [4]) which requires a preprocessing step in the input data by passing it through a set of Radial Basis Functions (RBF) with predefined centers and widths before performing linear regression. In the RBFN model, the RBF centers and widths are fixed and empirically chosen by the user. This necessarily generates a large number of problem-specific parameters, which violates the requirements of generality, simplicity and reusability (Sec. 1.1). We therefore consider the RBFN model, although related to NPP, to be incompatible with our requirements.

The multilayer perceptron (MLP) model trained by a batch-mode back-propagation algorithm (see, e.g., [8,4]) fulfills many of the requirements specified in Sec. 1, although the online property and the reusability of internal representations are not guaranteed (since the back-propagation mechanism tunes the hidden layer exclusively w.r.t. the learning task). We will study this model in order to assess its potential as a system-level learning algorithm.

2 Methods

2.1 Common Representational format for system-level learning

We used population coding in our system to realize a common representational format (see [6] for details) which allows to use learning methods fulfilling the requirement of scalability presented in Sec. 1.1. A system-level quantity is represented by an activation on a two-dimensional surface, where the position and amplitude of the activation code for the (possibly two-dimensional) value and the confidence, respectively (see Fig. 1 for examples). Simple quantities (distance, height, size, ..) can be represented naturally in this way; information of higher intrinic dimensionality must be subjected to a suitable *projection*, which will be elaborated in more detail in Sec. 2.3. This way of storing information is not optimized for storage efficiency, which leads to high-dimensional data even though the intrinsic dimensionality of represented quantities is low.

2.2 Notation

We denote local activity in a two-dimensional population code A by $z^A(\boldsymbol{x}, t)$. A neuron at position $\boldsymbol{x}$ of population-code A is connected to another neuron at position $\boldsymbol{y}$ of population-code B by a weight matrix denoted $w^{AB}_{\boldsymbol{xy}}$.

2.3 Neural Projection-Prediction

The technique we present consists of two steps: a self-organizing map (SOM) [9] performing a *projection* of the input space on a two-dimensional *internal representation*, and a neural network mapping the internal representation to a

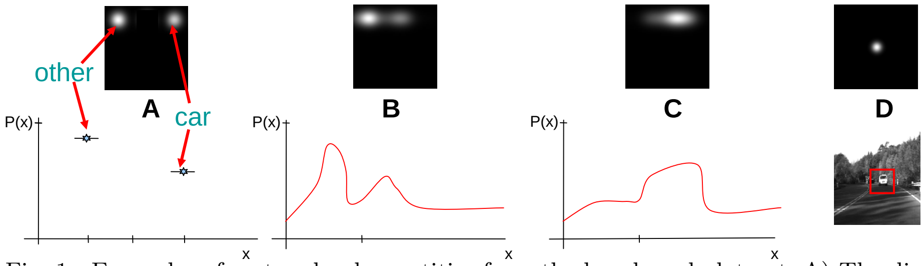


Fig. 1: Examples of system-level quantities from the benckmark dataset. A) The discrete distribution from an object classifier is translated into a population code where only certain locations carry information. B),C) Quasi-continuous one-dimensional measurements (here: object elevation and distance) are encoded into population codes that are extended along one axis. The uncertainty (multimodality) of measured distributions is transferred to the resulting population code. D) Quasi-continuous two-dimensional measurements (here: object position in camera image) are naturally encoded into a two-dimensional population code.

population-coded output using a normalized form of logistic regression. NPP is an online method: adaptation of the SOM and the logistic regression weights are performed in parallel to data transmission. The weight vectors of both projection and prediction are initialized with small random values.

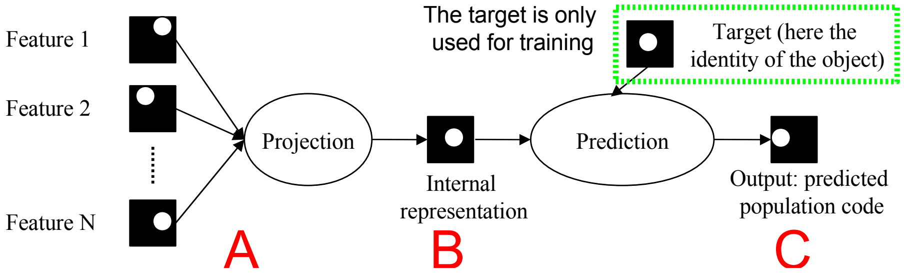

Feature 1
Feature 2
Feature N
The target is only used for training
Target (here the identity of the object)
Projection
Internal representation
Prediction
Output: predicted population code
A
B
C

Fig. 2: Description of the system.

Neural Projection-Prediction: Projection step In the projection step, the input A consisting of a concatenation of population codes is projected onto a two-dimensional output B using a self-organizing map (SOM, see [9]). At the same time, the SOM weights are updated according to the conventional rule based on input activity $z^A(\boldsymbol{x},t)$, output activity $z^B(\boldsymbol{y},t)$ and the position of the best-matching unit, $\boldsymbol{c}$. Projection and updating are governed by:

$$\begin{aligned} z^B(\boldsymbol{y},t) &= \sum_{\boldsymbol{x}} w^{AB}_{\boldsymbol{xy}} z^A(\boldsymbol{x},t) \\ w^{AB}_{\boldsymbol{xy}}(t+1) &= w^{AB}_{\boldsymbol{xy}}(t) + \epsilon_{\text{proj}}\, g^\sigma_{\boldsymbol{c}}(\boldsymbol{y})[w^{AB}_{\boldsymbol{xy}}(t) - z^A(\boldsymbol{x},t)] \\ \text{where } g^\sigma_{\boldsymbol{c}}(\boldsymbol{y}) &= \exp - \frac{\boldsymbol{y}-\boldsymbol{c}}{2\sigma^2}. \end{aligned} \qquad (1)$$

The values of ϵ_{proj} and σ are usually decreased over time; the precise form of this variation can influence projection quality considerably.

Neural Projection-Prediction: Prediction step We employ a supervised learning strategy where the supervision signal can come from annotated data or can be generated within the system (bootstrapping). Based on activity $z^B(\boldsymbol{x}, t)$ in the internal representation B, we use logistic regression [10] for training, minimizing the error of the prediction for C, $z^C(\boldsymbol{y}, t)$, based on the teaching signal $t^C(\boldsymbol{y}, t)$. The transmission and learning rules for the prediction step read:

$$\begin{aligned} z^C(\boldsymbol{y}, t) &= \sigma\left(\sum_{\boldsymbol{x}} w^{BC}_{\boldsymbol{xy}} z^B(\boldsymbol{x}, t)\right) \\ w^{BC}_{\boldsymbol{xy}}(t+1) &= w^{BC}_{\boldsymbol{xy}}(t) + \epsilon_{\text{pred}}\ z^B(\boldsymbol{x})[z^C(\boldsymbol{y}, t) - t^C(\boldsymbol{y}, t)] \end{aligned} \tag{2}$$

where the logistic sigmoid function $\sigma(x) \equiv \frac{1}{1+\exp{(\mu - x)}}$ is applied point-wise, correcting its argument by subtracting the long-time mean μ.

2.4 Baseline techniques

The following techniques serve as a baseline to demonstrate the advantages of NPP and the possible ways to improve it.

Locally weighted projection regression Locally weighted projection regression (LWPR) is a method for learning high-dimensional function approximators based on the superposition of multiple linear models in the input space. We used the publicly available implementation of LWPR [7] by the authors for all decribed experiments. Since LWPR stores a covariance matrix for each used linear model, it cannot deal with very high-dimensional data of $d > 1000$ due to memory consumption.

The multilayer perceptron model The multimayer perceptron (MLP) model [4] is a standard nonparametric regression method using gradient-based learning. It is a rather simple model, the only real free parameters being the number and size of hidden layers. The hidden layer may be viewed as an abstract internal representation where it is however unclear what is being represented. For network training, we employ the backpropagation algorithm with weight-decay and a momentum term (see, e.g., [8]). We used the pyBrain-library [11] for all described MLP experiments.

3 Experiments and Results

3.1 Data sets

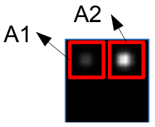
The *standard dataset* consists of 30000 samples of system-level quantities recorded in a large-scale integrated object detection system. A car is equipped with two front cameras and a laser in order to acquire data which is used by several processing layers [3]. The output of these layers is converted into population-codes (PCs) following the method described in Sec 2.1. Each sample of the *standard dataset* consists of 8 population-coded quantities representing abtract, invariant visual and spatial properties of objects detected by the system: distance, size, image position, elevation, two measures for distance-to-road-area, retinal size and depth (see [6]). Each PC consists of 64x64 elements, one sample therefore has a dimensionality of 32768. For training purposes, each PC is downsampled to a size of 16x16, making the dimensionality of one example $d = 2048$. The *reduced dataset* contains three PC per sample: retinal XY position, retinal size and distance. The *noised dataset* contains the 8 PC, plus 5 PC of uniform noise. As explained in Sec 2.1, the effective dimensionality is lower than the size of the PC, because some areas of the PC are not used. However, we keep the unnecessary dimensions to show that NPP is robust under the addition of irrelevant dimensions (*data mining ability*, see Sec. 1.1).

We generate several *LWPR datasets* of roughly the same content but reduced dimensionality since LWPR cannot deal with the high dimensionality of the default dataset. This is achieved by computing the coordinates (2 numbers) of the center of gravity for each of the 8 PCs contained in a single data sample. Depending on the type of encoded quantity, the y-coordinate of the center of gravity is irrelevant and can be omitted. The dataset LWPR-1 is the approximation of the *reduced dataset* and has 4 dimensions since image position carries two-dimensional information. Dataset LWPR-2 has dimensionality 9 approximating the 8 PCs (in most cases the y=coordinate of the center of gravity can be disregarded), and Dataset LWPR-3 has dimensionality 18, where 9 dimensions are taken from LWPR-2 and 9 dimensions contain uniform noise.

The examples are either positive (coming from car objects) or negative (not coming from cars). The ratio of positive to negative examples is approximately 1:10. For all training runs, we split the used dataset: the first 15000 examples are used for training, the last 15000 examples for evaluation.

3.2 Evaluation measures

The output of the trained Neural Projection-Prediction architecture is a discrete population code $z^C(\boldsymbol{x}, t)$ predicting the identity of a detected object (see Fig. 3.2). The population code is formed by two regions of activations, a_1 coding for the class "car" and a_2 for the class "non-car". We calculate a confidence value $C = A_1 - A_2$ with $A_{1,2} = \sum_{a_{1,2}} z^C(\boldsymbol{x}, t)$ for each example. A decision is made by comparing C to a variable threshold θ. This is visualized in Fig. 3.



$$C = A_1 - A_2$$

$$\text{class} = \begin{cases} \text{"car(positive)"} & C > \theta \\ \text{"non-car(negative)"} & \text{otherwise} \end{cases}$$

Fig. 3: Decision making process for performance evaluation.

This decision can be compared to the known "true" class of the object given by the identity. For different values of θ, we calculate the *false positive rate* $\mathrm{fpr} = \frac{\#(\text{incorrect positive classifications})}{\#(\text{negative examples})}$ and the *false negative rate* $\mathrm{fnr} = \frac{\#(\text{incorrect negative classifications})}{\#(\text{positive examples})}$ for the whole stream. We will assess classification quality by fnr-fpr plots also known as *receiver-operator characteristics* (ROCs).

3.3 Experiments with Baseline techniques

We evaluated LWPR using the datasets LWPR-1, LWPR-2, LWPR-3. Default LWPR parameters were used, except for an initial distance metric of 0.0625 and enabled meta learning. By changing the decision threshold θ applied to the real-valued output of LWPR, ROCs were produced which can be seen in Fig. 4. The figure shows that LWPR is able to solve the classification task well and improves greatly from the first to the second experiment by adding more input dimensions. However, it has difficulty coping with unnecessary dimensions, as one can observe when we add 9 random input dimensions in dataset LWPR-3. Furthermore, LWPR training failed due to memory limitations when using the *default dataset* or the *reduced dataset* due to the number of receptive fields. It is not suitable to work with such high dimensionalities in its present form. It takes approximately 4 *rounds* of data for the LWPR to converge (one round is one iteration over the whole dataset). It creates 170 receptive fields for LWPR-1, 790 for LWPR-2, and 730 for LWPR-3.

We trained an MLP as described in Sec. 2.4 using the standard and the reduced dataset (see Sec. 3.1). We employ MLP networks with an input layer of size 2560, one hidden layer of size 50 and one output neuron, using a sigmoid nonlinearity for each neuron. We verified that the results are similar for a number of hidden units between 50 and 100 hidden units. Training of the MLP requires 5 rounds (gradient steps) before early-stopping occurs. Training convergence was fast in spite of the high input dimensionality, resulting in ROCs given in Fig. 4 by applying a varying threshold θ to the real-valued MLP output.

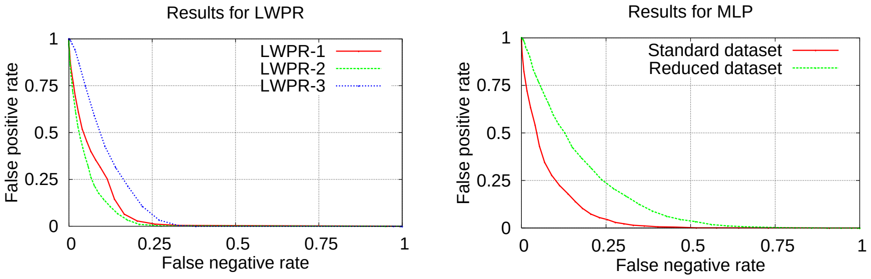


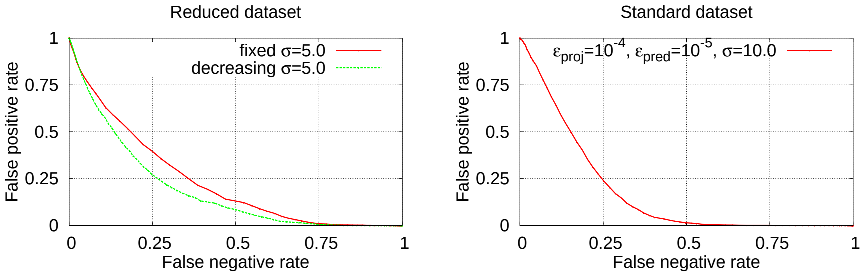
Fig. 4: Results for LWPR and MLP algorithms

3.4 Experiments with Neural Projection-Prediction

We trained our algorithm on the *standard, reduced and noised datasets*, and we limit the training to one round of data, so each example is presented once. We impose this constraint to our algorithm in order to evaluate its potential on online scenarios.

Reduced Dataset In order to characterize the qualities and the flaws of our algorithm, we evaluate the quality of the prediction depending on several parameters of the learning. We take a standard value for the learning constant of the prediction: $\epsilon_{pred} \simeq 1/15000$, 15000 being the number of training examples. We verified during our experiments that the value of the learning constant of the projection can vary between 0.1 and 0.0001 without affecting the results. We also performed experiments with different values and decreasing functions for the SOM radius. SOMs are usually trained with a decreasing radius, which implies a fixed number of training samples and a converging network. As we aim to work on real-world online problem, we verify that we obtain comparable results for a fixed radius. One can observe on Fig. 5, for $\epsilon_{proj} = 0.001$ and $\epsilon_{pred} = 0.0001$, that having a fixed radius does not change the results of the prediction drastically.

Performance with a high dimensionality We now run our algorithm on the *standard dataset*. Our algorithm requires a very small amount of parameter tuning, as it performs similarly for a large span of values. We keep a constant radius $\sigma = 10$. One can observe on Fig. 5 that the performances with the *standard dataset* are better than the performances with the *reduced dataset*, especially in the high false-negative rate regime.


Reduced dataset

Standard dataset

Fig. 5: We can observe on the left figure the results for a fixed radius and a lineary decreasing radius (from 5.0 to 1.0), using the reduced dataset. On the right we have the result for the standard dataset, using a fixed radius. The results for the standard dataset are indeed better than for the reduced dataset.

We show with this experiment that the NPP complies with several requirements from Sec. 1.1: simplicity and scalability. For now, as the SOM projection does not depend on the target value, we can say that the internal representations are not task-specific, and thus reusable. The technique is less efficient than the LWPR or the MLP, which have a task-specific internal representation. This lower performance is the price to pay for the reusability of the internal representation. Another reason for the lower performance is the fact that we perform only one round of learning.

Resistance to noise Fig. 6 shows the results of the experiments conducted in order to study the performance of our algorithm with noisy data. We used the *noised dataset* which has a dimentionality of 3328, with standard parameter values taken from the previous section. We can observe that the noise does not affect the NPP. As expected, the SOM algorithm is resistant to noisy dimensions.

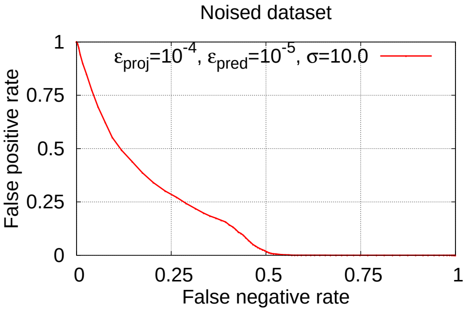


Fig. 6: Analysis of the resistance to noise.

4 Conclusion and key findings

Based on data obtained from a large-scale processing system, we study three learning techniques to determine whether they fulfill the requirements described in Sec. 1.1 related to system-level learning in large scale architectures or not.

The LWPR algorithm gives the best results regarding the quality of the prediction. It also prevents catastrophic forgetting, and so fulfils the requirement of *online regression*. However, the internal representation is task-specific, it is unable to cope with too high dimensions or with additional noise, so it does not meet several requirements: *scalability*, *data mining ability*, and *reusability of internal representations*. Also, we observe that it performs best with a certain amount of manual tuning (selection of input dimensions and parametrization of distance metrics), which goes against the requirement of *simplicity*. LWPR is then not suitable for our system-level learning requirements.

The MLP algorithm also performs well in terms of the quality of the prediction. The internal representation (hidden layer) is not easily reusable since it is strongly influenced by the chosen learning task. The MLP algorithm meets the criteria of *scalability, data mining* and obviously *generality and simplicity*, since the internal representation size is found to be uncritical. However it is not an online algorithm and it faces known issues of catastrophic forgetting (see [8]), and so it is unsuitable for an online real-world system, where new combinaisons of inputs can overwrite previously learned combinaisons.

The NPP meets all the requirements described in Sec. 1.1. It is a *generic and simple* method with a small amount of parameters, and usable in a plug and play manner. It is able to handle very high dimensions (*scalability*), and is not disturbed by unnecessary dimensions (*data mining ability*). The internal representation does not depend on the task, only on the input space (*reusability of internal representations*). Finally, it performs *online regression* where LWPR and MLP need to use the training data several times. We want to emphasize the fact that our goal is the system-level applicability. As we have shown, we pay the price for this in the form of reduced performance. Even so, the benefits for the whole system can be huge, especially in real-world applications.

5 Future works

As future research topics, we propose several additional mechanisms, most notably in the projection step. We want to improve the performance while retaining the advantages we gained. We will first improve the projection in order to represent the input space as faithfully as possible. We also plan to derive a feedback signal in order to modulate the SOM clustering depending on the quality of the prediction.

References

1. A Gepperth, B Mersch, J Fritsch, and C Goerick. Color object recognition in real-world scenes. In JM de Sa, editor, *ICANN 2007, part II*, number 4669 in Lecture Notes in Computer Science. Springer Verlag Berlin Heidelberg New York, 2007.
2. H Wersing and E Körner. Learning optimized features for hierarchical models of invariant object recognition. *Neural Computation*, 15(7), 2003.
3. J. Schmüdderich, N. Einecke, S. Hasler, A. Gepperth, B. Bolder, S. Rebhan, M. Franzius, R. Kastner, B. Dittes, and H. Wersing. System approach for multi-purpose representations of traffic scene elements. In *Proceedings of the ITSC*, 2010. submitted.
4. S Haykin. *Neural networks: a comprehensive foundation.* Prentice Hall, 1999.
5. B Mersch, T Glasmachers, P Meinicke, and C Igel. Evolutionary optimization of sequence kernels for detection of bacterial gene starts. *Int J Neural Syst*, 17(5), 2007.
6. A Gepperth, J Fritsch, and C Goerick. Cross-module learning as a first step towards a cognitive system concept. In *Proceedings of the First International Conference On Cognitive Systems*, 2008.
7. Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: Incremental real time learning in high dimensional space. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1079–1086, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.
8. RJ Reed and RJ Marks II. *Neural smithing: Supervised Learning in Feedforward Artificial Neural Networks.* MIT Press, 1999.
9. T Kohonen. Self-organized formation of topologically correct feature maps. *Biol. Cybernet.*, 43:59–69, 1982.
10. CM Bishop. *Pattern recognition and machine learning.* Springer-Verlag, New York, 2006.
11. Tom Schaul, Justin Bayer, Daan Wierstra, Sun Yi, Martin Felder, Frank Sehnke, Thomas Rckstie, and Jrgen Schmidhuber. Pybrain. *Journal of Machine Learning Research*, 2010. to appear.