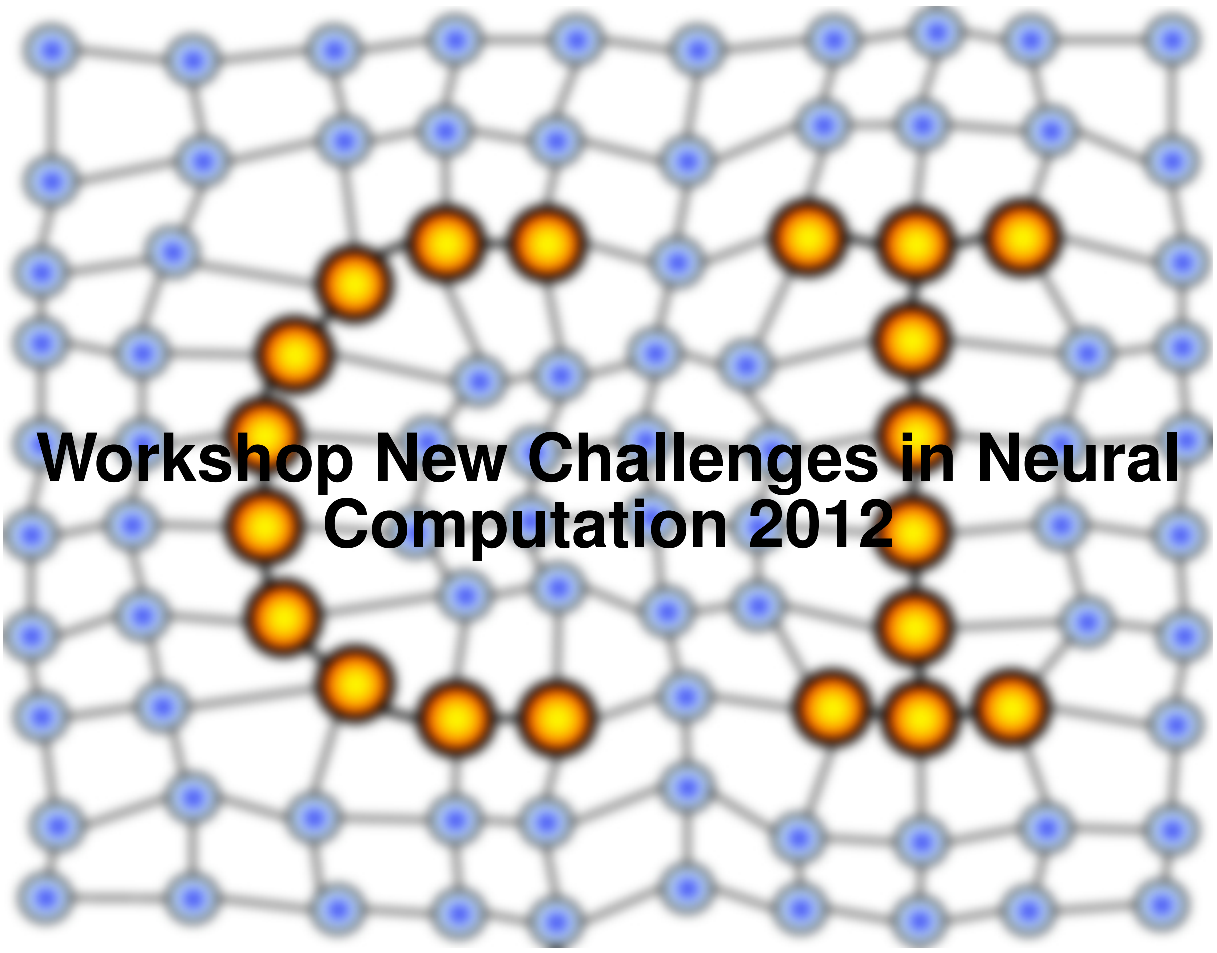


Workshop New Challenges in Neural Computation 2012

Report 03/2012
Submitted: 18.08.2012
Published: 21.08.2012

Barbara Hammer[1], Thomas Villmann[2] (Eds.)
(1) University of Bielefeld, Dept. of Technology CITEC - AG Computational Intelligence, Universitätsstrasse 21-23, 33615 Bielefeld
(2) University of Applied Sciences Mittweida, Technikumplatz 17, 09648 Mittweida, Germany

Table of contents

New Challenges in Neural Computation - NC^2 2012

(B. Hammer, T. Villmann) 4

Keynote Talk: Information Theoretic Feuture Selection for High-dimensional Data Analysis

(M. Verleysen) 5

Explorative Learning of Right Inverse Functions: Theoretical Implications of Redundancy

(M. Rolf, J. J. Steil) 6

Modeling Human Movements with Self-Organizing Maps using Adaptive Metrics

(M. Klingner, S. Hellbach, M. Kästner, T. Villmann, H.-J. Böhme) 14

Learning Motor Primitives with Echo State Networks

(J. Gütschow, J. Lohmann, D. Koryakin, M. V. Butz) 20

Robotics with the Head in the Clouds

(S. Hellbach, H.-J. Böhme) 34

Mixture of Gaussians for Distances Estimations with Missing Data

(E. Eirola, A. Lendasse, V. Vandewalle, C. Biernacki) 37

Enhancement Learning in Functional Relevance Learning

(T. Villmann, M. Kästner, D. Nebel, M. Riedel) 46

Discriminative Probabilistic Prototype Based Models in Kernel Space

(D. Hofmann, A. Gisbrecht, B. Hammer) 58

No Perplexity in Stochastic Neighbor Embedding
(M. Strickert) …........….. 68

How to Visualize a Classifier?
(A. Schulz, A. Gisbrecht, K. Bunte, B. Hammer)…... 73

Locally Weighted Regression using an Error-based Allocation Strategy
(S. Vukanovic, N. Carey, R. Haschke, H. Ritter)..84

Classification in High-dimensional Spectral Data - Precision vs. Interpretability vs. Model Size
(A. Backhaus, U. Seiffert) …..............…... 88

Feature Extraction from Occupancy Grid Maps Using Non-negative Matrix Factorization
(M. Himstedt, S. Hellbach, H.-J. Böhme)... 97

Learning of Invariant Object Recognition in a Hierarchical Network
(M. Lessmann, R. P. Würtz) ....…..104

Experience in Training (Deep) Multi-Layer Perceptrons to Classify Digits
(J. Hocke, T. Martinetz) ...…..............…..113

New Challenges in Neural Computation NC^2 – 2012

Barbara Hammer[1] and Thomas Villmann[2]

1 – Cognitive Interaction Technology – Center of Excellence,
Bielefeld University, Germany

2 – Faculty of Mathematics / Natural and Computer Sciences,
University of Applied Sciences Mittweida, Germany

The workshop New Challenges in Neural Computation, NC^2, took place for the third time, accompanying the prestigious DAGM conference in Graz, Austria. The workshop centers around exemplary challenges and novel developments of neural systems covering recent research concerning theoretical issues as well as practical applications of neural research. This year, fourteen contributions from international participants have been accepted as short or long contributions, respectively, covering diverse areas connected to challenges in robotics, visualization, interpretation, and sparsity, learning in the context of non-Euclidean data, and invariances and feature learning, respectively. In addition, we welcome an internationally renowned researcher, Prof. Michel Verleysen from Université Catholique de Louvain, who gives a tutorial about 'Information theoretic feature selection for high-dimensional data analysis'. The invitation of an invited speaker became possible due to the generous sponsoring of the European Neural Networks Society (ENNS) and the German Neural Network Society (GNNS). Following the workshop, a meeting of the GI working group on Neural Networks took place.

We would like to thank our international program committee for their work in reviewing the contributions in a short period of time as well as the organizers of DAGM for their excellent support of the workshop.

Keynote talk: Information theoretic feature selection for high-dimensional data analysis

Prof. Dr. Michel Verleysen, Université catholique de Louvain, Engineering Faculty - Electricity Department

Abstract:
Machine learning methods are used to build models for classification and regression tasks, among others. Models are built on the basis of information contained in a set of samples, with few or no information about the underlying process.

The more information there is in the set of samples, the better the model should be. However, this natural assumption does not always hold, since most machine learning paradigms suffer from the 'curse of dimensionality'. The curse of dimensionality means that strange phenomena appear when data are represented in a high-dimensional space. These phenomena are most often counter-intuitive: the conventional geometrical interpretation of data analysis in 2- or 3-dimensional spaces cannot be extended to much higher dimensions.

Among the problems related to the curse of dimensionality, the feature redundancy and concentration of the norm are probably those that have the largest impact on data analysis tools. Feature redundancy means that models will lose the identifiability property (for example they will oscillate between equivalent solutions), will be difficult to interpret, etc.; although it is an advantage on the point of view of information content in the data, the redundancy makes the learning of the model more difficult. The concentration of the norm is a more specific unfortunate property of high-dimensional vectors: when the dimension of the space increases, norms and distances will concentrate, making the discrimination between data more difficult. Most data analysis tools are not robust to these phenomena. Their performance collapse when the dimension of the data space increases, in particular when the number of data available for learning is limited.

This tutorial will start by a presentation of phenomena related to the curse of dimensionality. Then, feature selection will be discussed, as a possible remedy to this curse. Feature selection consists in selecting some of the variables/features among those available in the dataset, according to a relevance criterion. The goal is twofold: to avoid redundancy between features, and to discard irrelevant ones. State-of-the-art feature selection methods based on information theory criteria will be presented, together with the respective advantages of filter, wrapper and embedded methods.

The tutorial will conclude by opening new research questions about feature selection with informatics theoretic criteria.

Explorative learning of right inverse functions: theoretical implications of redundancy

Matthias Rolf and Jochen J. Steil

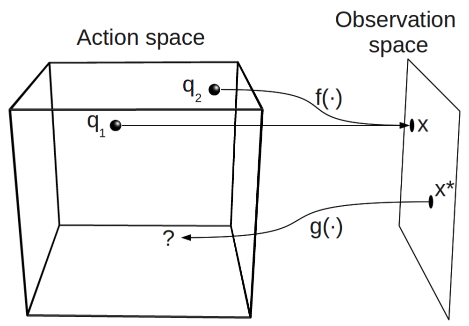
Research Institute for Cognition and Robotics, Bielefeld University
{mrolf,jsteil}@cor-lab.uni-bielefeld.de

Abstract. We investigate the role of redundancy for exploratory learning of inverse functions, where an agent learns to achieve goals by performing actions and observing outcomes. We present an analysis of linear redundancy and investigate goal-directed exploration approaches, which are empirically successful [1], but hardly theorized except negative results for special cases [2], and prove convergence to the optimal solution.

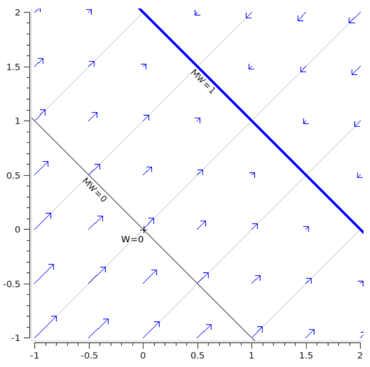
1 Introduction

In many learning scenarios, agents perform actions in some action space, whereas outcomes are measured in a different observation space. We assume that these two spaces are connected by a forward function that turns actions into observations. In order to achieve some desired behavior, an inverse function is needed that returns an appropriate action. A standard example is motor learning, in which action are motor commands like joint angles or forces. A forward function turns the actions into outcomes like effector positions. Learning a corresponding inverse has to rely on exploration schemes that generate examples for supervised learning. One substantial challenge is to deal with the redundancy in such domains: often multiple actions are mapped on the same outcome, such as different joint angles of an arm resulting in the same hand position. In this case learning can not be phrased as standard regression problem because multiple correct solutions exist. Often, the action space is very high-dimensional, which makes exhaustive exploration unfeasible. Yet, a number of practically efficient schemes have been proposed based on "goal-directed" exploration. This idea has been used for tuning of well initialized inverse functions [3–5]. Goal-directed exploration is particularly beneficial for learning from scratch, because it is applicable in very high-dimensional spaces [1]. Only very few theoretical results are available why and when such schemes can be successful. To the opposite, Sanger [2] proved that certain formulations can fail systematically even in simple domains.

This paper aims to deepen the theoretical understanding of such learning schemes in redundant domains. We first formalize the general problem and discuss its difficulties. Then, we provide a throughout analysis of the linear case with redundancy, which is applied to goal-directed exploration. To our knowledge, we thereby provide the first positive theoretical outcomes on such learning by proving convergence to an optimal solution if exploratory noise is added.



(a) Relation of action and observation space



(b) Learning gradient $-\frac{\partial E^X}{\partial W}(W)$

Fig. 1. Left: a forward function f maps actions into a lower dimensional outcome. An inverse g must suggest an action for a given target. Right: The performance gradient pulls any value W towards $MW = 1$. Gray contours show paths along the gradient.

2 Two Spaces and their Gradients

We consider an agent that can execute actions q in the action space $\mathbf{Q} \subseteq \mathbb{R}^m$. An action results in an outcome $x \in \mathbf{X} \subseteq \mathbb{R}^n$ in the observation space. Both variables are coupled by the forward function $f(q) = x$. The agent is asked to achieve some "goal" observation $x^* \in \mathbf{X}^* \subseteq \mathbf{X}$. It has to generate an action $\hat{q}$, such that $x = f(\hat{q}) = x^*$. The agent's selection of an action can be denoted by a function $g(x^*) = \hat{q}$. The learning task is to obtain a function g that can realize all goals:

$$f(g(x^*)) = x^* \ \forall \ x^* \in \mathbf{X}^* \tag{1}$$

Hence, g must be a *right inverse function* of f on the set of goals $\mathbf{X}^*$. Inverse functions do not always exist, so we need to require that f is surjective with $n \leq m$. For $n < m$ different actions can result in the same outcome, which is referred to as redundancy. An exemplary situation is shown in figure 1a.

2.1 The learning task in the observation space

In the observation space, obtaining a right inverse function can be directly formulated as optimization problem. We parametrize the function g with W. For some set of goals $\mathbf{X}^* = \{x_0^*, ..., x_{K-1}^*\}$, the *performance error* E^X naturally measures how much an inverse estimate g deviates from the solution in (1).

$$E^X(W, \mathbf{X}^*) = \frac{1}{2K} \sum_{k=0}^{K-1} ||f(g(x_k^*, W)) - x_k^*||^2 \tag{2}$$

Learning can be formulated as gradient descent on this error [6, 7]. The central difficulty is that computing the *performance gradient* $\partial E^X/\partial W$ requires analytic knowledge about the forward function: Since W appears inside $f(\cdot)$, differentiating $\partial E^X/\partial W$ requires to know the derivative of f. In general, inverse problems do not provide a teacher to indicate such optimal gradient directions.

2.2 Explorative learning in the action space

If the performance gradient is not available, a feasible way to probe knowledge is to generate examples (x, q) by exploration [1–5, 8, 9]. The setup starts by performing some action q_l, and observing the outcome $x_l = f(q_l)$. The *action error* on $D=\{(x_l, q_l)\}_l$ measures how well the inverse estimate fits the data:

$$E^Q(W, D) = \frac{1}{2L}\sum_{l=0}^{L-1} ||g(x_l, W) - q_l||^2 \ . \tag{3}$$

Learning is performed by descending the *action gradient* $\partial E^Q/\partial W$. Importantly, this scheme is not a data-driven version of minimizing E^X. In (3) we can replace $x_l = f(q_l)$ and see that the error evaluates on $g(f(q_l)) - q_l$. Hence, reducing E^Q corresponds to learning a *left inverse function* g, while the learning task is to obtain a right inverse function, which corresponds to minimizing E^X in observation space. Empirical results show that a right inverse function can be learned by minimizing E^Q [1, 3, 8]. Why this is possible is not theoretically understood for the general case. In fact, this kind of learning largely depends on how the data set chosen, whether f is linear or not, and whether the system contains redundancy: For the redundant case, left inverse functions do not exist on general data sets because different q_l can have the same outcome x_l. Trying to fit such inconsistent examples results in averaging, leading to invalid results in non-linear domains [6]. Sanger [2] investigated goal-directed exploration in the non-linear case without redundancy and showed that learning is not guaranteed to work.

This paper complements these previous, negative outcomes and investigates redundancy in linear domains. As a first positive result, we show that performance- and action-gradient have a non-negative angle and provide fix-point conditions.

3 Gradients in linear domains

In the linear domain, the relation between actions $q \in \mathbf{Q} \subseteq \mathbb{R}^m$ and outcomes $x \in \mathbf{X} = f(\mathbf{Q}) \subseteq \mathbb{R}^n$ is given by the linear forward function:

Definition 1 (Linear Forward Function). *We define the forward function as $f(q) = M \cdot q$ where M is a matrix $M \in \mathbb{R}^{n\times m}$ with $n \leq m$ and rank$(M) = n$.*

Requiring M to have full rank implies solvability of the right inverse problem. Correspondingly, we use a linear inverse estimates, with parameters W:

Definition 2 (Linear Inverse Estimate). *We define the inverse estimate as $g(x^*, W) = W \cdot x^*$ where W is a real-valued parameter matrix $W \in \mathbb{R}^{m\times n}$.*

Using these two definitions, we can re-write the right inverse equation (1) as linear equation. Assuming that the goals x^* span the entire space $\mathbf{X}$ we get:

$$f(g(x^*)) = x^* \;\; \forall\, x^* \quad \Leftrightarrow \quad MW = \mathbb{1}_n. \tag{4}$$

Hence, W must be a right inverse matrix of M. This equation is exactly solvable in W. For $n<m$ it is ill-posed and multiple solutions W exist.

We can now insert these definitions in the error-functionals defined in the last section and compute the gradients. For the *performance gradient* we get:

$$\frac{\partial E^X(W,\mathbf{X}^*)}{\partial W} = \frac{\partial E^X(W,\mathbb{X}^*)}{\partial W} = M^T(MW - \mathbb{1}_n)\mathbb{X}^* \tag{5}$$
$$\text{with } \mathbb{X}^* = \frac{1}{K}\sum_{k=0}^{K-1} x_k^* x_k^{*T} \in \mathbb{R}^{n\times n}.$$

Fig. 1b shows the performance gradient in relation to correct right inverse solutions. As an example we have chosen a forward matrix $M = (0.5, 0.5) \in \mathbb{R}^{1\times 2}$. The figure shows the parameter space of $W \in \mathbb{R}^{2\times 1}$. Right inverse matrices fulfill $MW = \mathbb{1}_1$ or in scalar notation $MW = 1$. These solutions give $\frac{\partial E^X}{\partial W} = 0$. The performance gradient drives any value of W straight to that solution manifold.

Considering a data set $D = \{(x_l, q_l)\}_l$ with $x_l = Mq_l$, we first define

$$\mathbb{Q} = \sum_{k=0}^{L-1} q_l {q_l}^T \quad \text{and} \quad \mathbb{X} = \sum_{k=0}^{L-1} x_l {x_l}^T = M\mathbb{Q}M^T \; .$$

Using this notation the *action gradient* is:

$$\frac{\partial E^Q(W,D)}{\partial W} = \frac{\partial E^Q(W,\mathbb{Q})}{\partial W} = (WM - \mathbb{1}_m)\mathbb{Q}M^T. \tag{6}$$

We can now show a tight relation between minimizing E^X and E^Q:

Theorem 1. *For any data set D, the action gradient is related to the performance gradient on the observed $\{x_l\}$ positions by*

$$M^T M \frac{\partial E^Q(W,\mathbb{Q})}{\partial W} = \frac{\partial E^X(W,\mathbb{X})}{\partial W} \; . \tag{7}$$

Proof.

$$M^T M \frac{\partial E^Q(W,\mathbb{Q})}{\partial W} \overset{(6)}{=} M^T M (WM - \mathbb{1}_m)\mathbb{Q}M^T = M^T(MWM - M)\mathbb{Q}M^T$$
$$= M^T(MW - \mathbb{1}_n)M\mathbb{Q}M^T = M^T(MW - \mathbb{1}_n)\mathbb{X} \overset{(5)}{=} \frac{\partial E^X(W,\mathbb{X})}{\partial W} \qquad \square$$

Both gradients have a *non-negative angle* since $M^T M$ is a positive semi-definite matrix. For $n=m$, $M^T M$ is even positive definite which guarantees a positive angle. Hence, learning a right inverse function is generally possible by minimizing

E^Q. For $n < m$, $M^T M$ is singular and the action gradient can project in its nullspace. This makes the redundant case mildly more complicated, but not as difficult as the general non-convex case, for which no angle can be guaranteed for arbitrary data sets. However, this theorem does *not* give a direct relation to the performance on the actual goals $E^X(W, \mathbb{X}^*)$. Whether a right inverse can be learned depends on whether the observations $\{x_l\}$ in D span the entire space $\mathbf{X}$.

4 Fixpoint analysis

In this section we investigate how the data set D, generated by exploration, shapes the learning process. Starting from some initial parameter value W_0, the parameters are iteratively updated with the learning equation

$$W_{t+1} = W_t - \eta \frac{\partial E^Q(W, \mathbb{Q})}{\partial W} \ . \tag{8}$$

Our main concern is whether learning converges to a W that satisfies the right inverse function condition $MW = \mathbb{1}_n$. In order to check for this behavior, we analyze the fixpoints $\partial E^Q / \partial W = 0$ of the learning equation depending on D.

The following two theorems give general conditions for which combinations of a parameter value W and data set D the action gradient becomes zero.

Theorem 2 (Sufficient fixpoint condition). *If W is a partial left inverse of M on the actions q_l (i.e. $WMq_l = q_l \forall q_l \in D$), then W is a fixpoint of eqn. (8).*
Proof.

$$\begin{aligned} WMq_l = q_l \forall q_l \in D \Leftrightarrow WM\mathbb{Q} = \mathbb{Q} \Leftrightarrow (WM - \mathbb{1}_m)\mathbb{Q} = 0 \\ \Rightarrow (WM - \mathbb{1}_m)\mathbb{Q}M^T = 0 = \frac{\partial E^Q}{\partial W} \end{aligned} \qquad \square$$

Sanger [2] showed for goal-directed exploration that this condition is also sufficient in the non-linear case with $n = m$. In fact, this condition is very general because it indicates that the action error in eqn. (3) is zero. The learner already fits the data. In a linear system with $n = m$, the condition is also necessary because M is square with full rank. Therefore the right-multiplication with M^T in the proof is reversible and we get equivalence between both statements. For redundant systems the condition is not necessary, since a left inverse does not exist on arbitrary data sets. If, for instance, D contains data $q_i \neq q_j$ and $x_i = x_j$, these samples can not be fitted. A more general condition is given by:

Theorem 3 (Necessary fixpoint condition). *If W is a fixpoint of learning equation (8), then W is a (partial) right inverse of M on the observed positions x_l, i.e. $MWx_l = x_l \ \forall \ x_l \in D$.*
Proof.

$$\begin{aligned} &\frac{\partial E^Q(W)}{\partial W} = 0 \quad \Rightarrow \quad M\frac{\partial E^Q(W)}{\partial W} = 0 \\ &\Leftrightarrow M(WM - \mathbb{1}_m)\mathbb{Q}M^T = (MW - \mathbb{1}_n)M\mathbb{Q}M^T = (MW - \mathbb{1}_n)\mathbb{X} = 0 \\ &\Leftrightarrow MW\mathbb{X} = \mathbb{X} \Leftrightarrow MWx_l = x_l \ \ \forall \ \ l \end{aligned} \qquad \square$$

Like theorem 2 this statement becomes an equivalence for $n = m$ (here because the left-multiplication with M is reversible). We can summarize both theorems:

$$WMq_l = q_l \forall\, l \quad \Rightarrow \quad \frac{\partial E^Q(W)}{\partial W} = 0 \quad \Rightarrow \quad MWx_l = x_l \forall\, l$$

Only for $n = m$ we get a full equivalence between these conditions. This asymmetry for $n < m$ is the second result on the impact of redundancy, additionally to the gradients losing their strictly positive relation in theorem 1. According to theorem 3, learning from examples will always result in a right inverse solution on the outcomes x_l contained in the data set. If the outcomes do not span the entire space $\mathbf{X}^*$, the solution will only be valid in the corresponding subspace.

4.1 Goal-directed Exploration

With these fixpoint conditions we can investigate right inverse learning driven by particular exploration processes. Goal-directed exploration has been discussed for the generation of data D in [3], but using a formulation without exploratory noise that can possibly fail even in non-redundant domains [2].

A data set $D_t = \{(x_k^{(t)},\ q_k^{(t)})\}_k$ is newly generated for each learning step t. The current inverse estimate $g(x^*, W_t)$ is evaluated on $\mathbf{X}^* = \{x_0^*, ..., x_{K-1}^*\}$ to select actions $q_k^{(t)}$. In order to inject exploratory noise, we follow [1] and add a perturbation function $E(x_k^*)$ to the inverse estimate. In the linear case we obtain this by choosing actions with some generating matrix W_{gen}:

$$q_k^{(t)} = g(x_k^*, W_t) + E(x_k^*) = W_{gen}x_k^* \quad \text{with} \quad W_{gen} \sim W + \varepsilon \ .$$

The components of the perturbation $\varepsilon \in \mathbb{R}^{m \times n}$ are chosen i.i.d. with zero mean and variance σ^2. Examples for multiple perturbations are collected and used for one gradient step according to equation (8). For our analysis we assume that enough data is collected to approximate the learning process by the expectation of this exploration process. First, we derive the expected action matrix:

$$\mathbb{Q} = E\left[(W+\varepsilon)\mathbb{X}^*(W+\varepsilon)^T\right]_\varepsilon = E\left[W\mathbb{X}^*W^T + W\mathbb{X}^*\varepsilon^T + \varepsilon\mathbb{X}^*W + \varepsilon\mathbb{X}^*\varepsilon^T\right]_\varepsilon$$

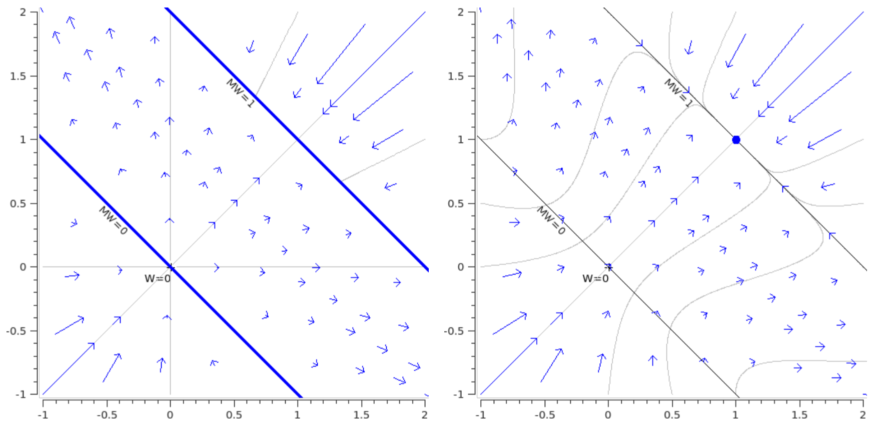
Here we get $E[W\mathbb{X}^*\varepsilon^T] = E[\varepsilon\mathbb{X}^*W] = 0$ because $E[\varepsilon] = 0$. Further we can derive that $E[\varepsilon\mathbb{X}^*\varepsilon^T] = trace(\mathbb{X}^*)\sigma^2\mathbb{1}_m$ which gives the action matrix

$$\mathbb{Q} = W\mathbb{X}^*W^T + trace(\mathbb{X}^*)\sigma^2\mathbb{1}_m \ . \tag{9}$$

Without noise ($\sigma = 0$) this matrix has a rank of at most n, but can also become zero if W is zero. This degeneration can cause failures of learning [2]. With noise, it has full rank, which implies that all fixpoints are valid right inverse functions:

Proposition 1. *For $\sigma^2 > 0$: $\quad rank(\mathbb{Q}) = m \ , \quad rank(\mathbb{X}) = n$*

Proof. $rank(\mathbb{Q}) = m$: The symmetric form $W\mathbb{X}^*W^T$ in eqn. 9 is positive-semidefinite. The second term is positive-definite for $\sigma^2 > 0$. The sum of a positive-semidefinite and a positive-definite matrix is also positive-definite, which implies full rank. $rank(\mathbb{X}) = rank(M\mathbb{Q}M^T) = n$ then follows from basic linear algebra. □



(a) Learning gradient without noise (b) Learning gradient with noise $\sigma^2 = 0.2$

Fig. 2. Goal-directed exploration with noise converges to the pseudoinverse of M.

Proposition 2. *For $\sigma^2 > 0$, any fixpoint W of the learning equation 8 is a right inverse of M.*

Proof. We know from theorem 3 that $MW\mathbb{X} = \mathbb{X}$ for any fixpoint. Since $\mathbb{X}$ has full rank we can right-multiply with $\mathbb{X}^{-1}$ and get: $MW\mathbb{X} = \mathbb{X} \Rightarrow MW = \mathbb{1}_n$ $\square$

For a full analysis we insert $\mathbb{Q}$ into the gradient equation (6):

$$\frac{\partial \hat{E}^Q(W)}{\partial W} = (WM - \mathbb{1}_m)(W\mathbb{X}^* W^T + trace(\mathbb{X}^*)\sigma^2 \mathbb{1}_m)M^T.$$

Using this equation we can show that the exploration results in a unique fixpoint:

Theorem 4. *For $\sigma^2 > 0$, the unique fixpoint of the learning equation 8 is the Moore-Penrose pseudoinverse: $W = M^\# = M^T(MM^T)^{-1}$.*

Proof. Expanding the gradient first gives for $\alpha = trace(\mathbb{X}^*)\sigma^2 > 0$:

$$0 = \frac{\partial \hat{E}^Q}{\partial W} = WMW\mathbb{X}^* W^T M^T + WM\alpha \mathbb{1}_m M^T - \ W\mathbb{X}^* W^T M^T - \alpha \mathbb{1}_m M^T$$

We can now use the previous result $MW = \mathbb{1}_n$ and substitute MW with $\mathbb{1}_n$:

$$\begin{aligned} W\mathbb{X}^* + \alpha WMM^T - W\mathbb{X}^* - \alpha M^T &= \alpha WMM^T - \alpha M^T = 0 \\ \Leftrightarrow WMM^T = M^T \quad \Leftrightarrow \quad W &= M^T(MM^T)^{-1} \end{aligned}$$ $\square$

Fig. 2 illustrates the learning gradient for goal-directed exploration in the example with $M = (0.5, 0.5)$ without (a) and with noise (b). Without noise, the procedure can end up in any of the fixpoints described by theorem 3. It stops in any correct solution $MW = 1$, but also in the entire nullspace of M $(MW = 0)$. Gray lines show exemplary trajectories on which W_t is changed during learning. Without noise, these trajectories are entirely concentric: the exploration never leaves the initial column space and does not allow to orient for new stimuli. With noise, the qualitative behavior is drastically changed (Fig. 2b). Noise removes the erroneous fixpoints on $MW = 0$. The gradient is not concentric around $W = 0$ anymore and allows W to change the column space. On the solution manifold $MW = 1$ the gradient pulls W towards the pseudoinverse, which is $W = M^{\#} = \binom{1.0}{1.0}$ in the example.

5 Discussion

We have investigated the explorative learning of right inverse functions, in order to let an agent perform actions that achieve some goal. It turns out that learning from examples corresponds to learning left inverse function. For the redundant linear case we have shown that such learning satisfies a non-negative gradient-relation to the actual right inverse problem. The analysis of goal-directed exploration as previously discussed in [2] shows that learning can lead to the discovery of valid right inverse functions, but may also get stuck in subspaces if the observation matrices lose rank. Redundancy causes additional failure modes in the Nullspace of the forward function. The new results for exploratory noise are particularly encouraging. Obviously, exploratory noise spans the entire observation space which eliminates undesirable fixpoints in subspaces. If applied in a redundant domain, noise even leads to the selection of the least-squares solution among the infinite set of solutions. Here our analysis gives the first affirmative results on goal-directed exploration from a theoretical perspective.

References

1. Rolf, M., Steil, J.J., Gienger, M.: Goal babbling permits direct learning of inverse kinematics. IEEE Trans. Auto. Mental Development **2**(3) (2010) 216–229
2. Sanger, T.D.: Failure of motor learning for large initial errors. Neural Computation **16**(9) (2004) 1873–1886
3. Oyama, E., Tachi, T.: Goal-directed property of online direct inverse modeling. In: IJCNN. (2000)
4. D'Souza, A., Vijayakumar, S., Schaal, S.: Learning inverse kinematics. In: IROS. (2001)
5. Peters, J., Schaal, S.: Reinforcement learning by reward-weighted regression for operational space control. In: ICML. (2007)
6. Jordan, M., Rumelhart, D.: Forward models: supervised learning with distal teacher. Cognitive Science **16** (1992) 307–354
7. Kawato, M.: Feedback-error-learning neural network for supervised motor learning. In: Advanced Neural Computers. Volume 6. Elsevier (1990) 365–372
8. DeMers, D., Kreutz-Delgado, K.: Learning global direct inverse kinematics. In: NIPS. (1992)
9. Haruno, M., Wolpert, D.M., Kawato, M.: Multiple paired forward-inverse models for human motor learning and control. In: NIPS. (1999)

Modeling Human Movements with Self-Organizing Maps using Adaptive Metrics

Mathias Klingner[1], Sven Hellbach[1], Marika Kästner[2], Thomas Villmann[2], Hans-Joachim Böhme[1]

[1] University of Applied Sciences Dresden, Artificial Intelligence and Cognitive Robotics Labs, POB 12 07 01, 01008 Dresden, Germany
{klingner, hellbach, boehme}@informatik.htw-dresden.de
[2] University of Applied Sciences Mittweida, Computational Intelligence and Technomathematics, POB 14 57, 09648 Mittweida, Germany
{kaestner, thomas.villmann}@hs-mittweida.de

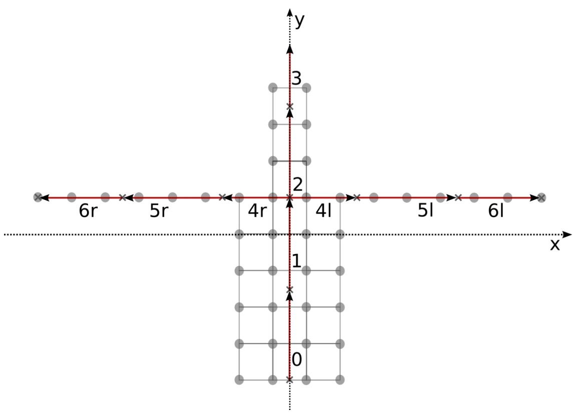
1 Introduction

A suitable human robot interface is of great importance for the practical usability of mobile assistance and service systems whenever such systems have to directly interact with persons. These interaction is commonly based on the learning and interpretation of the gestures and facial expressions of the dialog partner in order to avoid collision and to infer the intention of them. Therefore it is necessary to track the motion of the human body or rather the movements of individual parts thereof.

This work proposes an enhancement of the approach presented in [1]. It relies on the image of a depth camera from which a Self-Organizing Map (SOM) is extracted to model the human upper body. In contrast to the approach presented in [2], which is based on a extensive generation of training data and time-consuming training of classifiers for the individual body parts, a SOM is far less expensive but just as quick, in particular without the need of an extensive prior training process. Crucial in that context is the correct assignment of the SOM neurons to a specific region of the tracked person's upper body. In [1] the best-matching neuron (BMN) for a presented stimulus is determined based on the Euclidean distance with the three spatial dimensions x, y and z. Computation of the minimal Euclidean distance seems to be the most straight forward solution. However, it turned out to be insufficient for structured three-dimensional objects like the upper body, irrespective of the applied learning paradigm. Furthermore, depending on the specific technology used, additional problems may appear due to various sources of error and noise. This leads to the situation that sometimes neurons migrate from one part of the upper body to another. Without a verification of the SOM a future subsequent classification of the pose will produce incorrect results and maybe lead to wrong interpretation of the actual situation. Therefore we extended the approach in [1] by reshaping the trained SOM to a skeleton model to estimate the anatomical correctness of the pose. Having generated the skeleton model, incorrect Self-Organizing Maps will be rejected if the subsequent verification failed. This reduces the number of false classifications but it does not prevent the migration of the neurons. As it could be confirmed with a first naive classifier. The further goal is to eliminate this problem by integrating adaptive metrics [3]. The integration of the adaptive metrics presented in this paper is to be seen as ongoing work.

The goal of our approach is the generation of a problem specific parameter vector for a group of neurons. Therefor, the weight vector of a SOM neuron is extended compared to [1] including features like RGB color values, texture and neighborhood descriptors and the three spatial dimensions. By determining the relevant dimensions, a parameter vector is computed which includes a weight for each dimension of the weight vector of a group of neuron. Afterwards, it is possible to apply a simple feature selection or compute a metric based on the weights of the specific parameter vector.

An approach for data driven metric adaption within a Kohonen framework is described in [4] and [3]. The benefit of this method is binding neurons to a specific region by considering additional parameters in the input vector - rather than only their coordinates. The estimation of the relevant parameter and the computation of the metric is commonly estimated offline. In contrast to that, we intend to perform both online enabling the system to automatically adapt to the requirements of the scene. As an example, texture descriptors are more significant for the description of the chest of a person with a checkered shirt than for a topless person where color descriptors may be more meaningful.



(a) Initial skeleton model.

0	Lower torso
1	Middle torso (point of origin)
2	Upper torso
3	Head
4r/l	Left and right Scapula
5r/l	Left and right upper arm
6r/l	Left and right lower arm/hand

(b) Related joint description.

Fig. 1. Subfigure 1(a) shows the skeleton model at the point of initialization. In our context a bone is simply a vector. The origin of each bone is a joint which itself is the origin of a local coordinate system. All joints of the model are named in 1(b). They are connected in a global space above the point of origin of the skeleton model. Based on this we compare the bone length and the joint angles with corresponding reference values from a table of body measurements [5] for the pose validation in reference to the anatomical constraints.

2 Person detection and tracking

The separation of foreground and background in the captured scene image data is based on the method presented in [1]. The assumption is that only the foreground contains data of a person. In addition, we use a Viola Jones face detector in each frame to confirm this hypothesis. If a face is found, we perform a basic segmentation by clipping the point cloud 90 cm in front and behind the position of the detected face. Having successfully detected a face in the scene, the detector will only be active every tenth frame. The remaining data space is segmented more precisely by the Otsu threshold algorithm [6] to get only the person's point cloud data.

After extracting we use the resulting point cloud data for the training of our pre shaped Self-Organizing Map. Pre shaped means that the SOM's topology is created in the form of an human upper body with horizontally outstretched arms (Fig. 1(a)). At the initialization of the SOM the midpoint of the upper 4 head neurons will be placed over the center of gravity of the previously detected face in the input space. After initialization, the SOM is trained in the common way as shown in [7]. For this, the best-matching neuron $w_B(t)$ for each data point x is computed and adapted to $w_B(t+1)$ using the adaption rate $\eta(t)$:

$$w_B(t+1) = w_B(t) + \eta(t) \cdot (x - w_B(t)) \tag{1}$$

Hence, the best-matching neuron is determined by the computation of the minimal Euclidean distance between the current data point x and all neuron w_k of the SOM.

$$\min_{\forall k} ||x - w_k|| \tag{2}$$

The condition for the best-matching neuron is therefore

$$\forall k \neq B \quad ||x - w_B|| \leq ||x - w_k|| \,. \tag{3}$$

Exists more than one best-matching neuron under this condition, then one of the found will randomly selected. The adaption rate $\eta(t)$ is set to $\eta_{w_B}(t)$ for the best matching neuron and is defined to

$$\eta_{w_B}(t) = \eta_{in} \cdot (\frac{\eta_{fi}}{\eta_{in}})^{(\frac{t}{t_{max}})} \tag{4}$$

with η_{in} as the initial and η_{fi} as the final adaption rate, t is the number of the current training step and the maximum number of training steps is defined as t_{max}. For each best-matching neuron a limited neighborhood is defined by a Manhattan distance of 1 between a neighborhood neuron $w_n(t)$ and the best-matching neuron $w_B(t)$ on the grid structure of the SOM. Hence, the adaption rate for the neighborhood is set to $\eta_n(t)$ and defined as:

$$\eta_n(t) = \eta_{w_B}(t)/2 \,. \tag{5}$$

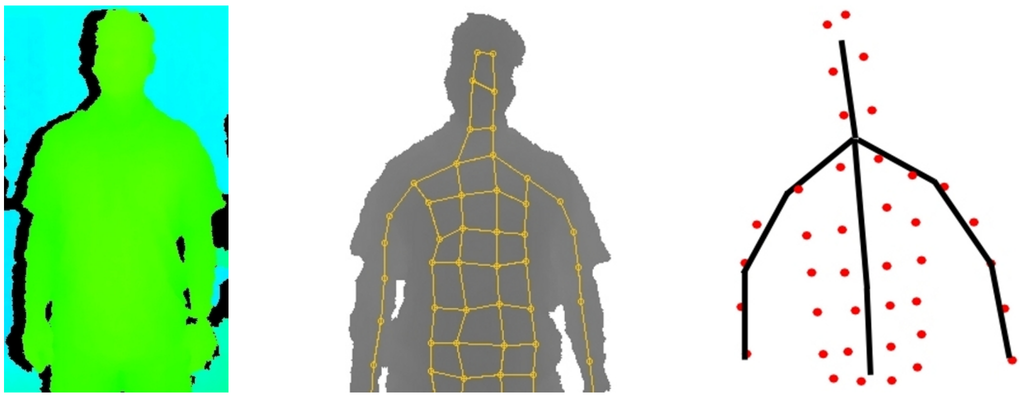
A step t contains a one-time presentation of all data points, which will t_{max} times per frame repeated.

As shown in equation 4, the learning rate used in the model decreases with the number of the training steps t [1]. A reasonable choice of the data presentation of the point cloud is therefore essential. The data presentation in the image processing usually starts in the upper left corner and the data points are presented row by row and column by column until the lower right corner is reached.

This kind of data presentation leads to a problematic situation. At the beginning of the training cycle of a single frame each best-matching neuron strongly adapts to the presented stimuli because $\eta_{w_B}(t)$ is on its initially high value. Presenting the data points as they arrive from the sensor in the mentioned order would adapt the SOM to the last presented stimuli, which lie at the lower border of the image. In each subsequent training step of a single frame with the decreasing $\eta_{w_B}(t)$ and $\eta_n(t)$ the current adaption distance decreases as well. Hence, the neurons will not as strongly adapted as in the training step before. Consequently, the neuron would stay at the lower border of the image and the SOM would not adapt correctly. The logical consequence is to present the data in a random order. However, it might also happen that a randomly chosen but static permutation of the input data leads to a bad adaption of the SOM. To avoid this, it makes sense to choose an individual random order for each frame. To reduce computational efforts it is useful to generate a permutation on a smaller set in each frame. Hence, we suggest a two level indexing method for a maximum permutation of the index values.

The first level contains a maximum quantity of m precomputed sets of permutations, being over the interval between 0 and the maximum resolution of the camera, with m being much smaller than the maximum resolution of the camera. As described above, the first index level ensures the correct development of the self-organizing map. For the second level, a frame dependent index set contains a permutation between 0 and m, where m is the number of precomputed sets of the first level. As a result, the algorithm uses a different randomly chosen set of permuted indices for each frame and each training step, with lower computational costs. Experiments suggest that this is a numerically stable method which assures a maximal unfolding of the SOM within the input space. Figure 2(b) shows a final status of the adapted SOM.

Furthermore, our current implementation extends the original approach by means of detection of the currently tracked person leaving the image. If this happens, the current SOM is discarded. For this purpose, the between class variance also known as inter-class variance from the Otsu threshold algorithm is monitored [6]. This variance describes the spread between the classes for foreground and background. The maximum of this value in the current frame is stored and compared to its subsequent value from the next frame. If the person leaves the scene these two values will differ strongly. Using a threshold for this difference, the exit of the tracked person can easily be detected.



(a) Part of a depth image captured with the kinect sensor.

(b) Cropped depth image with foreground data and adapted SOM.

(c) The related skeleton model with the neurons of the SOM.

Fig. 2. Subfigure a) shows the part from the captured scene containing a person in the foreground. Subfigure b) shows the cropped view of this image part. Only the cropped data is included in the training of the SOM. Finally subfigure c) shows the generated skeleton model with the neurons of the adapted SOM. All three images correspond to a valid case.

2.1 Skeleton Model

Validating the trained SOM is essential as the information of the SOM serves as input for a future motion classifier for human motions. Defective single poses are expected to be the main reason for classification errors and subsequently for errors in the situation assessment. These might eventually lead to invalid reactions of the robot.

To compensate this source of error we use a skeleton model with 10 bones and 9 joints. Figure 1 shows the model in its initialization state. It is generated in every frame out of a converged SOM and is subsequently checked for anatomical correctness. Therefor, it was evaluated that only two steps t_{max} are needed for the SOM to be in a stable or also converged state.

Every joint holds a local polar coordinate system in a forward kinematic chain. The reference figures for the test of anatomical correctness are taken from a standard table of body measurements [5]. The check includes the comparison of angles between child bone and their parent bones as well as the bone length. Every angle as well as every bone length of the standard table has an average measurement with a range of tolerance in proportion to the body size. If all parameters of the model are within acceptable tolerances, than the current SOM would be passed to a classification algorithm.

3 Adaptive Metrices

As described before, one of the most challenging problems for a to be implemented classification algorithm is the erroneous migration of neurons through the borders of the body parts. The most common cause of such an error are fast movements of the tracked person's arms near to another body part. Such sudden movements result in large changes of the point cloud in a short time. The approach presented in [1] introduces a modification of the learning rule forcing the neighboring neurons to stay closer together. This is by itself a substantial improvement. However, in situations with significant changes of the point cloud or if body parts are too close together, structural errors of the SOM still occur. A solution for such problems is to provide the neurons with information about their neighborhood in the input space.

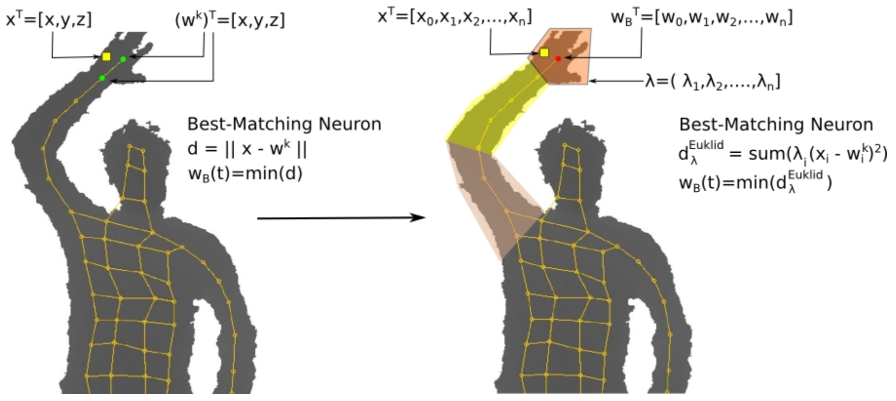

$x^T=[x,y,z]$
$(w^k)^T=[x,y,z]$
Best-Matching Neuron
$d = || x - w^k ||$
$w_B(t)=\min(d)$
$x^T=[x_0,x_1,x_2,\ldots,x_n]$
$w_B{}^T=[w_0,w_1,w_2,\ldots,w_n]$
$\lambda=(\lambda_1,\lambda_2,\ldots,\lambda_n]$
Best-Matching Neuron
$d_\lambda^{Euklid} = \text{sum}(\lambda_i(x_i - w_i^k)^2)$
$w_B(t)=\min(d_\lambda^{Euklid})$

Fig. 3. The left side of the graphic shows the typical input vector with three spatial dimensions, the vector of the single neuron of the SOM and the Euclidean distance function. For each stimulus x^T of the point cloud, the best matching neuron $w_B(t)$ will be determined by the estimation of the minimal Euclidean distance. The right side shows the enhancement of the stimulus and neuron vectors. In addiction to that a possible metric for the adaption is given by d_λ^{Euclid}. The vector λ describes a specific region of the upper body after the metric adaption.

Depth camera systems usually deliver far more information than only the point cloud data. As an example, the Microsoft Kinect also contains a RGB camera. Therefore we can work with the RGB data in combination with the depth information shown by Fig. 2(a). This enlargement of the input vector with further information results in the necessity for the adaption of the used metric.

The method described in [4] is such an extension of the common used metrics like Mahalanobis or Euclid. Defining a set of parameters λ which control the influence of each input dimension n for initialization:

$$\lambda = (\lambda_1, \cdots, \lambda_n) \quad \text{where} \quad \sum_i \lambda_i = 1 \tag{6}$$

and the corresponding metric:

$$d_\lambda^{Euclid}(x, w^k) = \sum_{i=1}^{n} \lambda_i (x_i - w_i^k)^2 \tag{7}$$

In combination with the existing framework it also provides the possibility for online learning of the metrics and the relevant parameters λ_i for a class of neurons. To keep the complexity of the metric adaption as small as possible, the SOM illustrated by figure 2(b) is partitioned in 4 different parts: the head, the torso, the hands and the arms. Each part contains at least one neuron. Furthermore, the single parts are variable in regard to the number of assigned neurons. Hence, different styles of clothes, e.g., long-arm shirts and t-shirts can be processed by the algorithm. The weights λ_i are initially chosen to be the same value for each input dimension:

$$\forall i : \lambda_i = 1/n \tag{8}$$

The training scheme works by detecting the person in the image, generating the SOM and adapting them one time with the classic training rule based on the three space dimensions. After this we obtain a stable, trained map with the neurons in the areas of the input space where they are expected. Figure 2(b) illustrates the trained map with the remaining foreground data.

At that time it is possible to use the neurons to obtain initial information for the image descriptors of the input vector. The λ parameters for each dimension i in the vector are subsequently weighted at run-time. Appropriate parameters are formally defined or they are determined at run-time by a stochastic gradient descent. The weights λ_i for the respective parameters w_i^B of the actual best-matching neuron are adapted with the following rule:

$$\lambda_i(t+1) := \lambda_i(t) - \epsilon(t) \cdot (x_i - w_i^B)^2 \quad \text{with} \quad \epsilon \in (0,1)\,. \tag{9}$$

In this equation is the factor $\epsilon(t)$ the learning rate comparably to $\eta(t)$ in equation 1. To obtain $||\lambda|| = 1$ and thereby to avoid numerical instabilities of the weighting factors a normalization is added. The principle idea is to find a unique set of weighting factors for each defined body region of the SOM. For more information see [8] and [9].

After the adaption of the weighting factors all following frames are trained with the adapted metrics. At this point it should not longer be possible for neurons of a specific body part to migrate to another one. This procedure leads to another possible advantage with regard to the resources of a mobile system.

The hardware of such a system defines the amount of possible parallel processes, and there are a lot more sensor systems than only the depth camera. Therefore we have to handle our resources with care. We assume that the adaptation of the metrics was correct and the neurons only adapt to their specific region of the body. The metric, so to speak, describes the spatial and textural characteristics of each region. Furthermore, we assume that these characteristics are different in particular to those of the background. Hence, it should not be possible that the neurons migrate to regions belonging to the background. Which allows to switch off the foreground-background segmentation without a loss of precision in the detection process after the metric has been adapted.

4 Conclusion

We presented a combination of an existing framework for people tracking and pose estimation based on depth camera systems with a theoretical approach for a more specific teaching algorithm. The basic approach by [1] is unsupervised, environment and also person independent. The adaptive metrics extension maintains these advantages while simultaneously stabilizing the adaption process, is expected to provide a reduction of the classification errors and reduces the computational effort for the computation. As a next step the implemented system needs to be evaluated. In particular, the advantages of the adapted metric need to be shown under real-world conditions.

References

1. Haker, M., Böhme, M., Martinetz, T., Barth, E.: Self-Organizing Maps for Pose Estimation with a Time-of-Flight Camera. In: Proceedings of the DAGM 2009 Workshop on Dynamic 3D Imaging. Volume 5742 of Dyn3D '09., Berlin, Heidelberg (2009) 142–153
2. Shotton, J., Fitzgibbon, A.W., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-Time Human Pose Recognition in Parts from Single Depth Images. In: CVPR, IEEE (2011) 1297–1304
3. Hammer, B., Villmann, T.: Effizient Klassifizieren und Clustern: Lernparadigmen von Vektorquantisierern. Künstliche Intelligenz **3**(6) (August 2006) 5–11
4. Hammer, B., Villmann, T.: Estimating Relevant Input Dimensions for Self-organizing Algorithms. In: Advances in Self-Organizing Maps, Springer (2001) 173–180
5. Ryf, C., Weymann, A.: Range of Motion - AO Neutral-0-Method Measurements and Documentation. Georg Thieme Verlag (1999)
6. Otsu, N.: A Threshold Selection Method from Gray-level Histograms. IEEE Transactions on Systems, Man and Cybernetics **9**(1) (1979) 62–66
7. Kohonen, T.: The Self-Organizing Map. Proceedings of the IEEE **78**(9) (1990) 1464–1480
8. Bojer, T., Hammer, B., Schunk, D., von Toschanowitz, K.T.: Relevance Determination in Learning Vector Quantization. In: Proc. of European Symposium on Artificial Neural Networks. (2001)
9. Hammer, B., Villmann, T.: Generalized Relevance Learning Vector Quantization. Neural Networks **15** (2002) 1059–1068

Learning Motor Primitives with Echo State Networks

Jakob Gütschow, Johannes Lohmann, Danil Koryakin, and Martin V. Butz

Cognitive Modeling, Department of Computer Science, University of Tübingen, Sand 14, 72076 Tübingen, Germany

Abstract. Movement control based on the combination of simple motor patterns (movement primitives) has proven to be a suitable approach for modeling human and robot behavior. Learning movement primitives involves the approximation of movement trajectories that can be described in terms of non-linear functions, different techniques have been proposed to solve this task. Here we investigate the ability of a special type of recurrent neural network, so called *echo state networks*, to perform this learning task. Our results based on a simple 2D simulation show that even quite small and simple structured networks are able to reproduce complex motion patterns.

Keywords: Movement Primitives, ESN, Reservoir computing

1 Introduction

The planning and execution of precise reaching movements is a highly complex challenge in robotics. Due to the high degrees of freedom of modern manipulators, there is a high level of redundancy inherent in these tasks. Targets can be reached on a large variety of trajectories and with a large variety of end-postures.

A promising approach to resolve this redundancy is to decompose the motions into basic patterns, so-called motor primitives [6]. The foundation for these primitives comes from biology, where they are described as “motor patterns that are considered basic units of voluntary motor control, thought to be present throughout the life-span” or simply “building blocks of movement generation” (see [11], p. 1). In vertebrates, the neuronal realization of motor primitives is likely located in the spinal cord. Here populations of neurons were found that recruit groups of muscles whose activities remained proportionally fixed throughout their recruitment [2]. By flexibly combining just a relatively small number of these building blocks, uncountable movement patterns and behaviors can be generated. It has been speculated, that such motor primitives may be crucial elements in early motor learning [2]. The existence of motor primitives implies that there might be some kind of *central pattern generator* (CPG) that combines simple low dimensional inputs to produce complex high dimensional outputs. A combination of low dimensional inputs can be achieved for instance by blending different motor primitives together.

To model such CPGs, two problems have to be addressed. First, the movement patterns have to be defined in terms of trajectories; for instance as time-series of task-space coordinates or angles in joint space. Second, a control process is necessary to monitor the movement execution and to adapt it dynamically if necessary. Apart from other approaches to learn and combine motor primitives like dynamic movement primitives [15] or Gaussian mixture models [1,10], [5] suggested an implementation via recurrent neural networks (RNNs). Especially the ability to autonomously generate rhythmic activation in a self-recurrent way is an interesting feature of RNNs as it seems to reflect the properties of biologic pattern generators.

The idea to learn motor primitives with RNNs is not new (see for instance [14]), but more recent research focuses on the capabilities of *reservoir computing*[18] to model a CPG. One architecture frequently used in studies on reservoir computing is based on the echo state networks (ESNs) presented in [7]. Compared to other RNNs, training of even very large ESNs is fast, easy to implement, and quite robust. ESNs are especially well-suited for time-series learning, and there are some examples for a successful application in the domain of movement planning and movement control. [3] compares ESNs with other frequently used algorithms used for time-series learning. [17] learned a bidirectional mapping of forward and inverse kinematics to perform goal-directed reaching movements within a single ESN. With an ESN applying leaky integrator neurons, [9] were able to reproduce trajectories like the *lazy figure eight*. Additionally, [9] also showed that the time scale of the reproduction of a certain ESN can be dynamically tuned. There have also been attempts to solve the mentioned control problem, i.e. the ability to switch between different learned motor primitives, within ESNs. As it was shown by [14], network dynamics can be shifted by bifurcation inputs. For instance, the architectures proposed by [19] and [13] proved that this approach is also viable for ESNs. Both architectures fulfill the requirements of a model for a CPG. To sum up, there is plenty of evidence for the ability of ESNs to account for the learning of movement primitives. With architectures like the one proposed by [19] or [13] it is also possible to solve the control problem within ESNs.

Even if these findings are promising, the applied networks were quite large (300 neurons in [13], and up to 2200 neurons in [19]), used special types of neurons with filter properties [20,19], or required more elaborate training mechanisms than the original ESNs [1]. Therefore, we are interested in the ability of “vanilla” ESNs to reproduce motor patterns in a completely self-recurrent way, without any additional input. As a first step, we use a simple 2D-simulator of an arm to define and execute basic movements to be learned and reproduced by the networks. We are focusing on the kinematics of the movement. Hence, we do not directly learn any dynamics. We concentrate on finding a coding scheme for motion trajectories that can be accurately learned, that can generate stable control commands, and that generalize well. Different encodings were employed

[1] For instance [19] used *ridge regression*[4], instead of the simple linear regression, proposed by [7].

to learn particular trajectories with ESNs. For instance, [13] as well as [3] used two-dimensional task space coordinates, whereas [19] learned different joint angle evolutions. In this work we assess the impact of different coding schemes on the learning performance. More precisely, we investigate if different coding schemes require different network parameters for successful learning.

As we are using networks that are guided by a constant output feedback loop without any additional external input the notion of stability is crucial. Our aim is to find *output feedback stable* networks in the sense of [16], i.e. networks that are not driven into an extreme attractor state by there own recurrent feedback. Some recent papers on output driven ESNs investigated which network properties are necessary to avoid error amplification due to recurrent feedback. While [16] pointed out that regularized[2] networks are less prone to error amplification, [12] suggested a balancing between the scaling of the output feedback strength and the amplitude of the learned time-series.

In the next section we provide a short introduction to ESNs. After this we sketch out our experimental setup. Next, we evaluate performances with respect to different coding schemes. A short discussion concludes the paper.

2 Echo State Networks

While a detailed description of the features and working of this architecture can be found in [7] and [8], this section only gives an overview of its most important features.

The basic ESN structure is quite similar to that of standard RNNs, consisting of an (optional) input layer, a layer of the hidden neurons (the so-called, dynamic reservoir), and an output layer. The crucial differences of ESNs compared to the standard RNNs are the pre-wired dynamic reservoir and the simple training procedure, which only optimizes the output weights. Figure 5 shows an overview of the architecture. Within the dynamic reservoir different dynamics unfold over time. These dynamics are combined to produce the output of the network.

It is necessary that the dynamic reservoir adheres to the so-called *echo state property*. This property is a requirement for stable dynamics and leads to a fading memory of the reservoir with respect to the input history. The *echo state property* can be achieved by restricting the spread of the reservoir weights through their division by the largest eigenvalue of the reservoir weight matrix and multiplying them with λ^*. This results in a matrix where the desired largest absolute eigenvalue equals λ^*, in other words the *spectral radius* of the matrix equals λ^*.

The state of the dynamic reservoir at time-step $n+1$ can be described by:

$$\mathbf{x}(n+1) = f(\mathbf{W^{IN}}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W^{OFB}}\mathbf{y}(n)), \tag{1}$$

where bold letters denote matrices and vectors. More precisely $\mathbf{x}(n)$ denotes the state of the dynamic reservoir at the n^{th} time-step, $\mathbf{u}(n+1)$ and $\mathbf{y}(n)$ denote the current input and the previous output, respectively. The weight matrices $\mathbf{W^{IN}}$,

[2] The term regularized refers to a low norm of the different weight matrices.

$\mathbf{W}$, and $\mathbf{W^{OFB}}$ contain weights of connections from the input neurons to the reservoir neurons, recurrent connections within the reservoir, and output feedback connections from the output neurons to the reservoir neurons, respectively. Finally, f denotes the transfer function of the units of the dynamic reservoir — usually a sigmoid. The state of the units in the output layer is obtained as follows:

$$\mathbf{y}(n+1) = f^{OUT}(\mathbf{W^{OUT}}\mathbf{u}(n+1), \mathbf{x}(n+1), \mathbf{y}(n)), \qquad (2)$$

where f^{OUT} denotes the transfer function of the output units — usually a linear function.

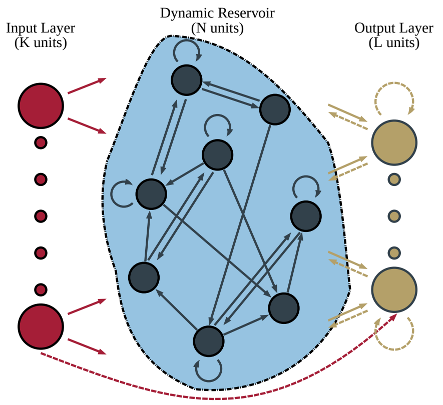


Fig. 1. Overview of the ESN-architecture. Dashed lines represent optional connections. Optimized connections are those leading to the output layer.

As noted above, a unique feature of ESNs is that the randomly initialized reservoir weights stay fixed. The only weights that are optimized during training are those connecting the dynamic reservoir and optionally the input layer with the output layer. To optimize the output weights the squared difference of the network output and the desired time-series is minimized.

The training process itself is executed in three consecutive phases. First, the initial transients of the dynamic reservoir are extinguished during a *washout phase*.

Second, the crucial part of the training is executed in the *sampling phase*. During this phase, teacher forcing is applied, feeding the intended output values back into the network via the output feedback connection weights $\mathbf{W^{OFB}}$. The resulting excitation of the reservoir is recorded in a matrix M (time-steps $\times$ internal nodes) while the corresponding teacher output values are collected within a vector T. The optimal weights are then calculated by

$$W^{OUT} = M^{-1}T \qquad (3)$$

where W^{OUT} denotes the connections to the output layer. In the third phase, the quality of the estimated weights is evaluated. To allow a direct comparison between different target time series we use the normed root mean squared error between the target signal and the network output to evaluate the quality of a certain ESN:

$$NRMSE = \sqrt{\sum_{i=1}^{I} \sum_{t=0}^{T-1} \frac{\|d_i(t) - y_i(t)\|^2}{T\sigma_i^2}}, \tag{4}$$

where I refers to the output dimension, T is the target sequence length int the exploitation phase, $y_i(t)$ is the network output at time step t in output dimension i, and σ_i^2 is the variance of the target dynamics $d_i(n)$ in the i'th dimension. There also exist online-learning procedures for training ESNs, which we do not consider here.

3 Experimental Setup

We use a simple simulator of a 2D-arm. This arm can be customized with regard to the number of segments and the length of these segments. Movement generation is carried out by first manually defining a trajectory for the endpoint. The trajectory is then reproduced by the arm, while data is recorded according to one of the coding schemes described below. The inverse kinematics are solved directly, without considering additional planning techniques.

We either recorded the position of the end-effector of the arm in target space, or the evolution of joint angles over time. Angles are measured in radiants separately for each joint. The location of the end point is recorded in pixels of a coordinate system laid over the task space. To investigate the ability of the networks to generalize the learned trajectories we did not only collect the absolute angles, or target locations, but also the differences in joint angles or end-effector coordinates between two subsequent steps. For the training we applied the raw data as well as data normalized to unity.

This data is then used for teacher-forcing. In case of joint angle data one output unit per joint is used. For the recordings of the location in target space, one output unit is used each for x- and y-coordinate, independent of the number of arm-segments. After training the ESNs, the activations of the output units in the exploitation phase are once again stored. These activations are the movement data that are then again used by the simulator to reproduce the recorded trajectory.

For our main evaluation trials, we used an arm setup with three segments, to draw different 8-shaped trajectories and recorded them with different coding schemes. Hence, the task we applied is similar to the *lazy figure eight* task described in [9]. With respect to these schemes varied crucial network parameters to identify the most effective ESN setup settings that yield the most successful learning of the trajectories. In the first set of trials, we investigated the influence of the spectral radius λ^*, the connectivity of the dynamic reservoir, as well as the initialization range of the output feedback weights W^{OFB} on the performance of the ESNs. We used seven different spectral radii, nine different connectivities, and nine different initialization ranges for the output feedback weights. Given that for every factor combination 20 networks were trained, the whole sequence required the training of 11340 networks. Ranges and step sizes of the parameter variations are displayed in the figures. As we were interested in the reproduction of the motion patterns with small networks, we kept the size of the dynamic reservoir fixed to 20 units in these runs. To estimate the influence of the different parameters, we trained 20 networks per parameter combination.

In the second set of trials another parameter setup was used. First, the connectivity of the reservoir was fixed to 0.9, as it did not show any significant influence on the results in the first trials. Instead, we varied the reservoir size from 10 to 20 units. Second, we varied the values of the initialization range of the output feedback weights W^{OFB} over a broader range. We used the same variations of the spectral radius λ^* as in the first set of trials. In this set of trials we applied seven different spectral radii, 11 reservoir sizes, and 24 initialization ranges for the output feedback weights. Again, we trained 20 networks per parameter combination, yielding a total of 30800 networks.

In the third set of trials we further investigated the influence of the reservoir size, creating networks with 25 to 100 units. The other parameters were the same as in the second run, yielding a total of 19600 networks.

Due to the large body of data and the fact that we would like to present the characteristics of the error distributions as detailed as possible, we decided to use box and whisker plots. The grayed area indicates the inter-quartile range covering 50% of the data, the outer whiskers indicate the 5% nad 95% quantile, repectively. Values out of this range but within the doubled inter-quartile range are considered as outliers (marked as open circles), values outside the doubled inter-quartile range are considered as extreme values and, if existent, are indicated with open triangles. The arithmetic mean is indicated by a black dot, whereas the smallest error value in each distribution is indicated by a gray square.

4 Results

As noted above, the coding schemes can be divided into schemes relying on joint angles and task space coordinates.

4.1 Relative Location in Target Space

We first investigated the network performance with data obtained with the normalized relative scheme, i.e. subsequent changes of the end effector position were recorded and normalized to unity. Fig. 2 displays the results of our first series of experiments. As can be deduced from the figure, the inter-quartile ranges of the different parameter setups overlap. There was not much variance due to the parameter variation and the overall performance was quite good. As it was pointed out by [8], the optimal spectral radius depends on the frequency of the desired signal. Interestingly, the best performing network was found for a spectral radius of 0.9 despite the fact that most networks generated with such a high spectral radius performed worse than networks with a smaller spectral radius.

The initialization of the output feedback weights W^{OFB} did not show a clear influence in the investigated interval and all values resulted in similarly good performance, the best results were found for an initialization interval of $[-0.08, 0.08]$. The overall effect of variations of the output feedback range was surprisingly small, therefore we decided to sample a broader interval in the next experiments.

The influence of the connectivity was completely indistinct, therefore we did not considered this parameter in the further analysis but fixed it to 0.9. Apparently, this task can be easily learned with ESNs, as even the naïve initialization could result in suitable networks.

In the second series of experiments, the variations once again failed to show more than minor effects (see Fig. 3). The error-medians remained nearly constant over the whole W^{OFB} initialization interval, but the variance increased for the extreme ranges of the initialization interval. However, the best networks were found in these ranges as well: Either with an initialization interval of $[-1.0, 1.0]$, or with an initialization interval of $[-10^{-15}, 10^{-15}]$. For this scheme varying the reservoir size had no general effect on the majority of the error-values. The effect of the variation of the error interval had no strong effect. The best performing networks were found for a reservoir size of 12 units. Concerning λ^*, a value of 0.9 led to the best results.

To sum up, the two-dimensional progression of end-effector changes in task space can be learned with quite small networks. The only parameter with more than a small effect was the spectral radius, with values above 0.9 increasing the error variance. The data pattern obtained with absolute task space coordinates was quite similar, hence we did not include the results.

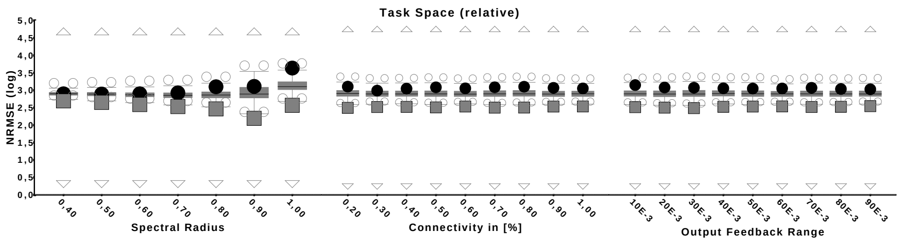


Fig. 2. Box-plots for results of parameter optimization for task space data (first experimental setup). Scaling is logarithmic. Extreme values (outside the doubled interquartile range) are indicated by arrows. The minimum of each distribution is indicated by a gray rectangle.

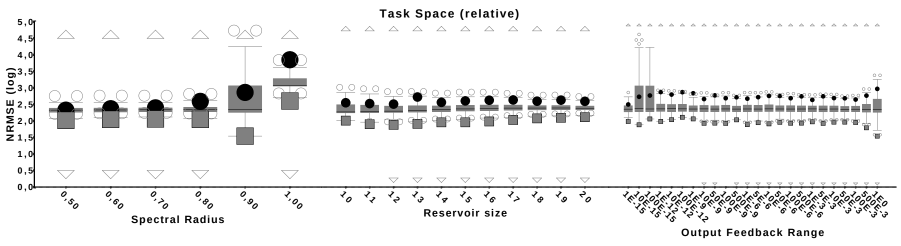


Fig. 3. Box-plots for results of parameter optimization for task space data (second experimental setup). Scaling is logarithmic. Extreme values (outside the doubled inter-quartile range) are indicated by arrows. The minimum of each distribution is indicated by a gray rectangle.

4.2 Joint Angles

Given the fact that the small networks we applied were able to learn a two-dimensional sequence we proceeded to the more complex task of learning a three-dimensional joint angle evolution within one dynamic reservoir. We considered two options of coding the movements in joint space. Either we recorded the absolute angles of each joint or we recorded the angular changes in consecutive time-steps. This latter relative coding proved to be problematic. Even if it was possible to learn and reproduce the time-series for the original initial arm posture, generalization was extremely poor. The reproduction of the trajectories was severely flawed when the initial arm posture was changed. Even small displacements led to major distortions of the original movement. The overall results are display in Fig. 5. Please note that these results were obtained for setups were the initial arm-posture during the reproduction was the same as during training. As noted before in this case the results are quite good, and only small effects of the parameter variations are visible. Once again a spectral radius of 0.9 turned out to produce the best results, even if the error distribution becomes broader for higher spectral radii. The variation of the initialization range of the output feedback weights only affected the broadness of the error distributions, the minimal values remained nearly unaffected. It is noteworthy that the reservoir size had only a very small effect on the overall performance — again quite small networks were able to reproduce the intended distribution. Because of the low generalization ability of networks trained with data obtained with this encoding scheme, we did not further investigate it.

More promising results were obtained with the absolute coding scheme (cf. Fig. 6). Although the output of the most successful network had a quite high NRMSE of 4.72, it produced a smooth trajectory very close to the original one (cf. Fig. 4). To reach this value, we identified several important parameters of the network. It turned out that the small reservoir size of 20 units was sufficient to reproduce the three different joint trajectories. This is in line with the results reported by [3] where it was also shown that quite small networks are able to reproduce at least three dimensional output series. For the spectral radius a value of 0.8 provided the best results. The overall effect of the spectral radius resembles the one observed in the other setups. The minimum error decreases with higher spectral radii, but the error distribution gets broader. We also varied the initialization interval of the output feedback weights W^{OFB} based on the considerations in [12], where it was pointed out that these bounds should be balanced with the interval of the target dynamics. Here, the best results were achieved with a value of 1.0 for the weight initialization interval. Again, this was a quite surprising result as the error distribution becomes much broader with higher initialization ranges, i.e. the most networks initialized with these values performed quite bad. But as the variation of the overall error range also increased the best network belonged to this sub-sample. It is noteworthy that even the smallest networks with a reservoir consisting of only 10 units were able to reproduce the intended trajectory with only slight distortions (cf. Fig. 7).

For the third experimental setup we increased the reservoir size up to 100 in several steps to examine whether we could find a maximal suitable size as reported in [12]. A major effect of this increase was a wider distribution of error values (see Fig. 8), but also a major increase in the error magnitude (see the different scaling of the y-axis in Fig. 6 and Fig. 8). Especially networks with small spectral radii and networks with large reservoirs tend to produce extreme error values. Nonetheless, a network with a reservoir consisting of 100 units also produced the lowest errors of all tested networks. For a better comparison of the minima of the different distributions, Fig. 9 displays the lower ranges of the error distributions. No clear effect of the variation of the initialization bound of the output feedback weights could be found.

To sum up, our results show that even small networks with reservoirs consisting of only 10 units are suitable to adapt to at least three-dimensional time-series. Once again, it is noteworthy that the reproduction took place in a completely self-recurrent manner, without any additional inputs. Surprisingly, the parameter variations we investigated primarily affected the broadness of the error distributions but not the extreme values. Nearly all of the investigated setups lead to the generation of at least a few suitable networks. This effect was especially prominent for the variation of the reservoir sizes, where the largest reservoir size of 100 units resulted in a nearly uniform error distribution.

5 Discussion

In this paper we described our current work on employing ESNs to learn and reproduce simple motor patterns. We mainly concentrated on finding encoding schemes to record movement data, which can be easily learned by ESNs. Compared to other approaches to learning motor primitives with reservoir computing techniques, we focused on small and simple networks similar to those applied by [3]. The first scheme we found to be viable codes the absolute angle of each joint of an arm over time. Although the quality of data reproduction depends on the parametrization of the ESN, we showed that it is generally possible to reach high reproduction accuracy with very simple networks. Additionally, this

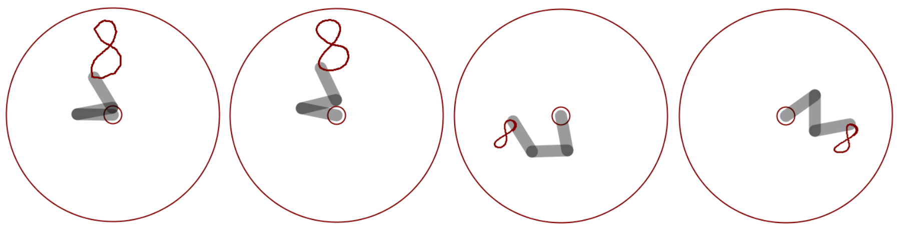


Fig. 4. Left to right: Comparison of original and reproduced trajectory coded in joint space (absolute), two examples of reproduction with the same learned target space dataset.

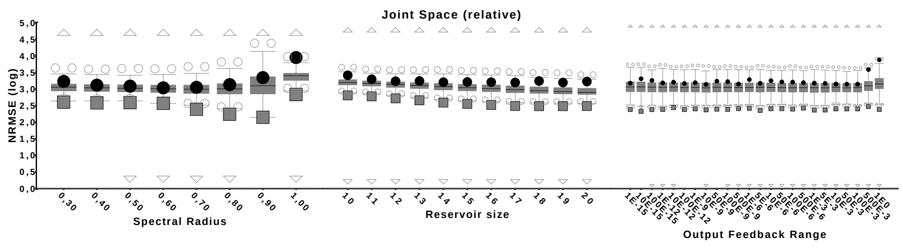


Fig. 5. Box-plots for results of parameter optimization for joint space data recorded with the relative coding scheme (second experimental setup). Scaling is logarithmic. Extreme values (outside the doubled inter-quartile range) are indicated by arrows. The minimum of each distribution is indicated by a gray rectangle.

scheme showed the interesting emergent effect of generating a smoothed and more symmetrical version of the original trajectory.

Scaling this scheme to 3D-movements might result in problems of the training due to the increased complexity of the data and the need for learning more time-series data simultaneously. So far we did not investigate the scaling properties of ESNs with respect to the movement data. [19] started with a network consisting of 300 neurons to learn the movement pattern of a single joint-angle, and 2400 neurons to account for a system with 22 DoF. Given our results we are optimistic to learn equally complex systems with a much smaller dynamic reservoir.

Another promising scheme encodes the movement in task space. It records the relative movement of the end-effector between consecutive time steps. This scheme has several advantages, apart from the general benefit that learning seems quite easy. First, due to the relative nature of the data, the learned movements can be simply transferred to any place in the task space, as long as it is not too close to the boundaries. Second, the scheme is completely independent of the number of arm-segments, as it only refers to the end-effector. Hence, there should be no scaling problem when extending this coding scheme to a 3D task space.

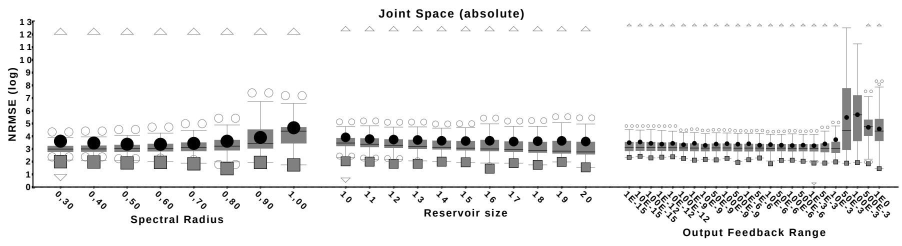


Fig. 6. Box-plots for results of parameter optimization for joint space data recorded with the absolute coding scheme (second experimental setup). Scaling is logarithmic. Extreme values (outside the doubled inter-quartile range) are indicated by arrows. The minimum of each distribution is indicated by a gray rectangle.

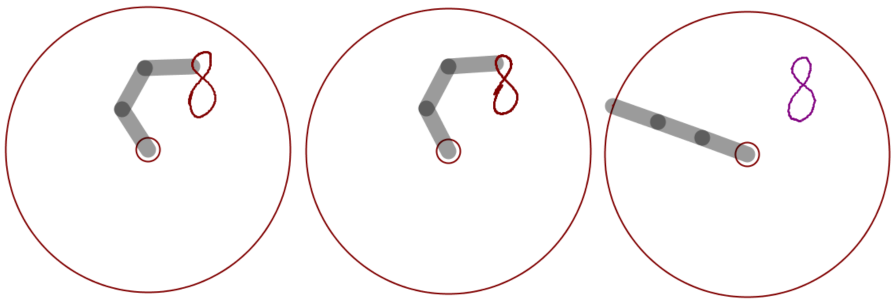


Fig. 7. The left and the middle panel display the reproduction performance of two small networks trained with absolute joint angle evolution. The rightmost panel displays the intended trajectory.

To sum up, our experiments showed that it is possible to learn simple motor pattern with quite small and simple ESNs. Even without regularization of the weight matrices (see for instance [16]), we found completely output-feedback driven networks that remained stable, but analyses regarding the long term stability are pending. As it was mentioned in the introduction, the second aspect of movement control with motor primitives, the adaptivity of movement control, is not realized yet. But as it was shown by [19] and [13], it is possible to switch between different dynamics within the same reservoir to produce different movements. This was achieved with a special input layer that served as a dynamic selection mechanism that enabled the selective activation of two different movement patterns.

On the other hand, dealing with perturbations is a much greater problem, which might not be solvable with our current approach. As it was mentioned in [1], *open-loop* approaches like the one proposed here cannot adapt very well to perturbations or delays. In [19], it was shown that reservoirs with suitable weight matrices are able to recover from perturbations and to converge back to the learned dynamic. At the moment experiments are missing that investigate if

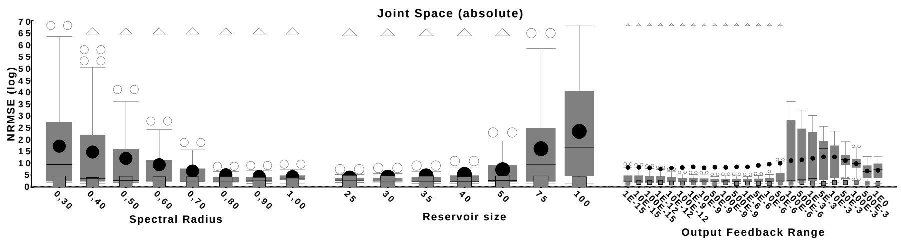


Fig. 8. Box-plots for results of parameter optimization for joint space data recorded with the absolute coding scheme (third experimental setup). Scaling is logarithmic. Extreme values (outside the doubled inter-quartile range) are indicated by arrows. The minimum of each distribution is indicated by a gray rectangle.

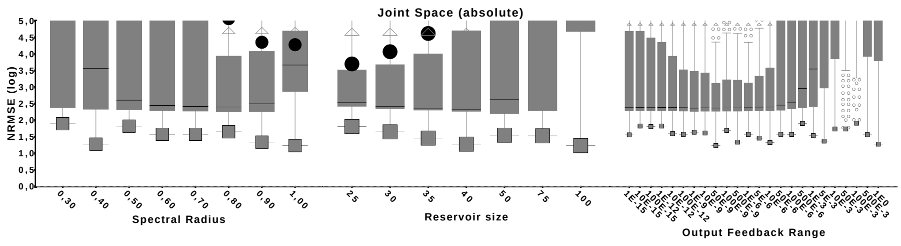


Fig. 9. Box-plots for results of parameter optimization for joint space data recorded with the absolute coding scheme (third experimental setup). Scaling is logarithmic and restricted to a range of 10^5. Extreme values (outside the doubled inter-quartile range) are indicated by arrows. The minimum of each distribution is indicated by a gray rectangle.

the simple networks that we trained are also able to recover from perturbations. So far we have shown that much simpler ESNs than previously thought are able to learn stable movement primitives. Our next step is to investigate if a simulated central pattern generator, like the ones proposed by [19] and [3], can be realized with small and simple ESNs.

References

1. Gribovskaya, E., Khansari Zadeh, S.M., Billard, A.: Learning Nonlinear Multivariate Dynamics of Motion in Robotic Manipulators [accepted]. International Journal of Robotics Research (2010)
2. Hart, C.B., Giszter, S.F.: A neural basis for motor primitives in the spinal cord. J. Neurosci. 30(4), 1322–1336 (2010), `http://dx.doi.org/10.1523/JNEUROSCI.5894-08.2010`
3. Hellbach, S., Eggert, J.P., Körner, E., Gross, H.M.: Time series analysis for long term prediction of human movement trajectories. In: Köppen, M., Kasabov, N., Coghill, G. (eds.) Advances in Neuro-Information Processing, pp. 567–

574. Springer-Verlag, Berlin, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-03040-6_69

4. Hoerl, A.E., Kennard, R.W.: Ridge regression: Biased estimation for nonorthogonal problems. Technometrics 12(1), 55–67 (1970)
5. Ijspeert, A.J.: Central pattern generators for locomotion control in animals and robots: a review. Neural Networks 21(4), 642–653 (2008)
6. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Movement imitation with nonlinear dynamical systems in humanoid robots. In: Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on. pp. 1398–1403 (2002)
7. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks. Tech. Rep. GMD Report 148, German National Research Center for Information Technology (2001)
8. Jaeger, H.: Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the echo state network approach. Tech. Rep. GMD Report 159, Fraunhofer Institute AIS (2002)
9. Jaeger, H., Lukoševičius, M., Popovice, D.: Optimization and applications of echo state networks with leaky integrator neurons. Neural Networks 20(3), 335–352 (2007)
10. Khansari-Zadeh, S.M., Billard, A.: Learning Stable Non-Linear Dynamical Systems with Gaussian Mixture Models. IEEE Transaction on Robotics (2011), http://lasa.epfl.ch/khansari
11. Konczak, J.: On the notion of motor primitives in humans and robots. In: Berthouze, L., Kaplan, F., Kozima, H., Yano, H., Konczak, J., Metta, G., Nadel, J., Sandini, G., Stojanov, G., Balkenius, C. (eds.) Proceedings of the Fifth International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems. pp. 47–53 (2005), http://cogprints.org/4963/
12. Koryakin, D., Lohmann, J., Butz, M.V.: Balanced echo state networks. Neural Networks (2012)
13. Krause, A.F., Bläsing, B., Dürr, V., Schack, T.: Direct control of an active tactile sensor using echo state networks. In: Ritter, H., Sagerer, G., Dillmann, R., Buss, M. (eds.) Human Centered Robot Systems, Cognitive Systems Monographs, vol. 6, pp. 11–21. Springer Berlin Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-10403-9_2
14. Nishimoto, R., Tani, J.: Learning to generate combinatorical action sequences utilizing the initial sensitivity of deterministic dynamical systems. Neural Networks 17(7), 925–933 (2004)
15. Pastor, P., Kalakrishnan, M., Chitta, S., Theodorou, E., Schaal, S.: Skill learning and task outcome prediction for manipulation. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on (May 2011). pp. 3828–3834 (2011), http://www-clmc.usc.edu/publications/P/pastor-ICRA2011.pdf
16. Reinhart, F., Steil, J.: Regularization and stability in reservoir networks with output feedback. Neurocomputing 90, 96–105 (2012), http://www.sciencedirect.com/science/article/pii/S0925231212001749, advances in artificial neural networks, machine learning, and computational intelligence (ESANN 2011)
17. Reinhart, R.F., Steil, J.J.: Reaching movement generation with a recurrent neural network based on learning inverse kinematics for the humanoid robot icub. In: Humanoids. pp. 323–330 (2009)
18. Verstraeten, D., Schrauwen, B., D'Haene, M., Stroobandt, D.: An experimental unification of reservoir computing methods. Neural Networks 20, 391–403 (2007)

19. Wyffels, F., Schrauwen, B.: Design of a central pattern generator using reservoir computing for learning human motion. In: ECSIS Symp. on LAB-RS. pp. 118–122 (2009)
20. Wyffels, F., Schrauwen, B., Stroobandt, D.: Stable Output Feedback in Reservoir Computing Using Ridge Regression. In: Kurková, V., Neruda, R., Koutník, J. (eds.) Artificial Neural Networks (ICANN), Lecture Notes in Computer Science, vol. 5163, pp. 808–817. Springer Berlin / Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-87536-9_83

Robotics with the head in the clouds

Sven Hellbach and Hans-Joachim Böhme

University of Applied Sciences Dresden, Artificial Intelligence and Cognitive Robotics Labs,
POB 12 07 01, 01008 Dresden, Germany
{hellbach;boehme}@htw-dresden.de

1 Introduction

Within the last decades an increasing application of methods from the field of artificial intelligence in the domain of robotics has taken place. In this context, the demands on the computational power of the robotic systems has also increased dramatically. However, the computer hardware does not always meet these demands. This leads to solutions, where complex calculations are performed at external high-performance computers or even distributed over the internet. The latter has attracted plenty of research interest with the emerging of cloud computing [1]. Even at conferences with an artificial intelligence background the topic is already present. At the 2012 World Congress of Computational Intelligence (WCCI) Piero P. Bonissone mentioned in his invited talk: "*The cloud is here and it is here to stay – like the internet.*" With this comes the fact that four accepted papers at the same conference bear the word *cloud* at least in their title.

In most cases the approaches for distributed computation are kept simple. In the easiest case, complete algorithms are computed off-line over the network and the system has to wait until the solution is available (e.g. text recognition system on smart phones [2]). More complex ideas, divide their methods into sub-tasks and have those sub-tasks computed on external computers [3].

2 Application Problems

Since the goal of robotics is to facilitate every-day situations, the research on assistance robots needs to be evaluated on real-world applications. Hence, some approaches have to be rejected, which were only developed under laboratory conditions. Especially such approaches that only slightly fulfill real-time performance requirements. For these methods it sounds reasonable not to rely solely on the on-board computer systems.

In particular, in the field of robotics two problems arise while distributing computations over the network: On the one hand, the usage of the network itself and the fact that a computation slot has to be available produces incalculable latencies not suitable for real-time performance. On the other hand, with a network or server failure the computational results would be missing at all. The absence of computational results might even lead to a failure of the robot system. To clarify, we consider an example where the global occupancy map is kept only on the server and only local maps for the close neighborhood are transferred to the robot for further processing. If the connection to the server would be lost the localization of the robot could not be guaranteed any more and hence the robot would need to be stopped.

Because of the high demands on the reactivity of the system and hence on the real time conditions, from our point of view, there is a high necessity on approaches or methods to eliminate the drawbacks coming with the use of distributed computing. In particular, existing methods should be evaluated with respect to the possibilities for only distributing parts of the methods and still being able to guarantee an adequate, not necessarily global optimal solution.

A principle approach is imaginable in a way that for local computing on the robot, computational fast methods are applied that allow to find a sub-optimal partial solution as a hypothesis for further processing. Simultaneously, the idea of cloud-computing is used to solve the problem with more computational power with an optimality constraint. For further processing, it is necessary that with delayed arrival of the optimal partial solution the overall solution can be corrected with respect to the error originating from the use of the sub-optimal partial solution.

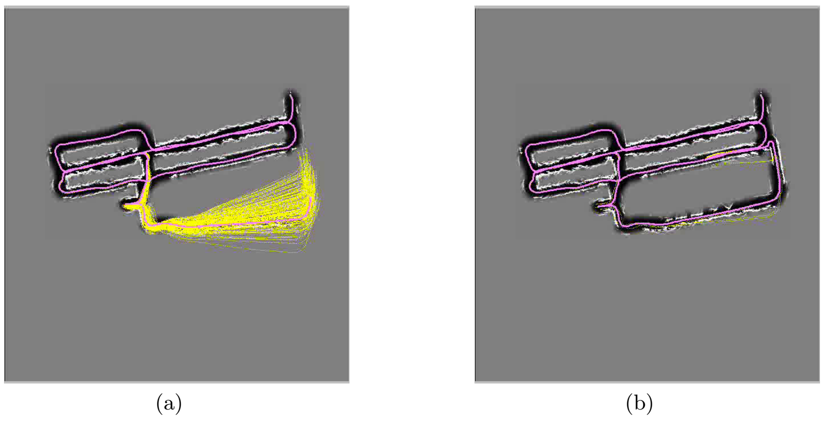


Fig. 1. Both figures show a visualization of a particle filter based SLAM implementation. The yellow and magenta lines show trajectory hypotheses for different particles. The particle with the highest probability is visualized in magenta together with the resulting occupancy map. (a) indicates the high variance within the hypotheses before a loop closing could be detected. In (b) the variance collapses to almost a single hypothesis. Figures have been captured from a video in the context of [5].

For illustrating this still abstract conceptional idea, the well known simultaneous-localization and mapping (SLAM) problem [4] is taken as an example for further description. For a mobile robot one of the most important tasks is to navigate within its environment. For this, the robot always needs to know its position – the result of the localization. Usually, the navigation takes place in unknown or changing environment. Hence, at the same time the robot needs to build a map of the yet unknown environment (see Fig. 1). Since, only the odometry of the robot is not sufficient to achieve this goal, different sensor systems are used, allowing the robot to perceive the local surroundings. The basic idea of the SLAM approaches then is to recognize already visited areas and with that information correct the wrong map hypotheses. In SLAM literature this idea is referred to as the loop closing problem.

Figure 1 shows a particle filter based approach [6], visualizing the trajectory hypotheses for different particles in Fig. 1(a). After loop-closing, particles propagating wrong hypothesis can be identified comparing the current sensor readings with previous ones coded in the occupancy map. Those particles are extinct.

Most state-of-the art approaches implement the entire SLAM framework on the robot's local computer, leaving not much room for further processing needed for human-machine interaction. For example in [3] however, first attempts have been made to solve this problem from a more engineering-like point of view.

Reconsidering our example in Fig. 1, it should be clear that the extinction of wrong particles might as well take place several steps after being at the position of the loop closure. The first idea that comes into mind, is to store the history of each particle for a predefined time window. Basically, each particle needs to carry the information from which particle in previous time steps it has been re-sampled. As soon as the particle specific computation from the remote system arrives, the modeled distribution can be corrected by deleting particles that had survived because of incomplete knowledge in the past.

Still the underlying approach – as well as other approaches in the field of robotics – promises to allow a more methodical solution, fulfilling the above mentioned needs to be distributable and to have a loose dependence on intermediate results. For this problem the key to a solution might be the fact that the particle filter models a stochastic process with a strong Markov assumption considering only the previous time step. However, extending the Markov assumption to previous time steps as needed, offers the possibility to correct the position of the previously misplaced particles.

3 Summary

This paper was meant to serve as a statement for future research directions. Hence, our goal was not to provide a practically applicable solution for the discussed problem. However, an example was given to show a possible direction. In the following the demands on algorithms developed for the suggested scenario are summarized:

- The algorithm should be able to cope with latencies occurring while parts are being computed remotely.
- With that comes the possibility that results can be corrected afterwards with low computational costs
- A possible loss of intermediate results should only have minor influence on the accuracy of the final solution.
- In particular, a final solution should be possible at all.
- The results that are being computed locally on the robot need to be fast, but not necessarily global optimal.

After all, the intention of the paper is to give a thought-provoking impulse for spending research efforts in this direction – in particular from a methodical or mathematical standpoint. The problems are already there and might become of increasing interest in the near future.

References

1. Jaatun, M.G., Zhao, G., Rong, C., eds.: Cloud Computing First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings. Lecture Notes in Computer Science. DOI: 10.1007/978-3-642-10665-1.
2. Berry, P.M., Gervasio, M., Peintner, B., Yorke-Smith, N.: Ptime: Personalized assistance for calendaring. ACM Trans. Intell. Syst. Technol. **2**(4) (July 2011) 40:1–40:22
3. Barkby, S., Williams, S., Pizarro, O., Jakuba, M.: A featureless approach to efficient bathymetric slam using distributed particle mapping. Journal of Field Robotics **28**(1) (2011) 19–39
4. Thrun, S., Burgard, W., Fox, D.: Probabilistic robotics. Intelligent robotics and autonomous agents. MIT Press (2005)
5. Schröter, C.: Probabilistische Methoden für die Roboter-Navigation am Beispiel eines autonomen Shopping-Assistenten. PhD thesis, TU Ilmenau (2009)
6. Doucet, A., Johansen, A.M.: A tutorial on particle filtering and smoothing: fifteen years later. The Oxford Handbook of Nonlinear Filtering (December 2009) 4–6

Mixture of Gaussians for distance estimation with missing data

Emil Eirola[a], Amaury Lendasse[abce],
Vincent Vandewalle[df], Christophe Biernacki[ef]

[a] Department of Information and Computer Science, Aalto University, FI–00076 Aalto, Finland

[b] IKERBASQUE, Basque Foundation for Science, 48011 Bilbao, Spain

[c] Computational Intelligence Group, Computer Science Faculty, University of the Basque Country, Paseo Manuel Lardizabal 1, Donostia/San Sebastián, Spain

[d] EA 2694, Université Lille 2, 1 place de Verdun, 59045 Lille Cedex, France

[e] Laboratoire P. Painlevé, UMR 8524 CNRS Université Lille I, Bât M2, Cité Scientifique, F–59655 Villeneuve d'Ascq Cedex, France.

[f] INRIA Lille – Nord Europe Parc Scientifique de la Haute Borne Park Plazza – Bât A – 40 avenue Halley 59650 Villeneuve d'Ascq, France

Abstract. In order to estimate distances between samples in a data set with missing values, a Gaussian mixture model is estimated from the data. For fitting the model, an extension to the EM algorithm is presented to make it applicable to missing data scenarios. The Gaussian mixture model enables the calculation of conditional means and variances for each missing value, and these are sufficient to calculate an estimate of the distance between any two samples in the data set. Experimental simulations confirm that the proposed method provides accurate estimates compared to alternative methods for estimating distances.

1 Introduction

Finite mixture models have proven to be a versatile and powerful modelling tool in a wide variety of applications. Particularly mixture models of Gaussians have been studied extensively to describe the distributions of data sets. The general approach to estimating the model from data is maximum likelihood (ML) estimation by the EM algorithm [5].

In this paper, we present how to apply the EM algorithm to estimate a Gaussian mixture model when the data contains missing values. Furthermore, the estimated model is used to estimate pairwise distance between samples. Using a Gaussian mixture model is ideal for this scenario, as it 1) can be optimised efficiently even in the presence of missing values, 2) allows one to derive estimates of pairwise distances, 3) is flexible enough to cover any distribution of samples, and 4) is sufficiently sophisticated to provide non-linear imputation.

In many real world machine learning tasks, data sets with missing values (also referred to as incomplete data) are all too common to be easily ignored. Values could be missing for a variety of reasons depending on the source of the data, including measurement error, device malfunction, operator failure, etc. However, many modelling approaches start with the assumption of a data set with a certain number of samples, and a fixed set of measurements for each sample. Such methods can not be applied directly if some measurements are missing. Simply discarding the samples or variables which have missing components often means throwing out a large part of data that could be useful for the model. It is relevant to look for better ways of dealing with missing values in such scenarios.

In this paper, the particular problem of estimating distances between samples in a data set with missing values is studied. Being able to appropriately estimate distances between samples, or between samples and prototypes, has numerous applications. It directly enables the use of several standard statistical and machine learning methods which are based only on distances and not the direct values, e.g.: nearest neighbours (k-NN), support vector machines (SVM), or radial basis function (RBF) neural networks [4].

An important consideration when dealing with missing data is the missingness mechanism. We will assume that a missing value represents a value which is defined and exists, but for an unspecified reason is not known. Following the conventions of [8], the assumption here is that data are Missing-at-Random (MAR):

$$P(M|x_{\text{obs}}, x_{\text{mis}}) = P(M|x_{\text{obs}}),$$

i.e., the event of a measurement being missing is independent from the value it would take, conditional on the observed data. The stronger assumption of Missing-Completely-at-Random (MCAR) is not necessary, as MAR is an ignorable missingness mechanism in the sense that, for instance, maximum likelihood estimation still provides a consistent estimator.

The paper is organised as follows. Section 2 reviews the EM algorithm for mixtures of Gaussians, and introduces the extension to missing data. Section 3 presents the estimation of pairwise distances. Section 4 includes comparison experiments on simulations of data with missing values.

2 EM for mixture of Gaussians with missing data

2.1 The standard EM algorithm

For the conventional EM algorithm for fitting a mixture of Gaussians, we follow the presentation of [4, Section 9.2]. Given data $\mathbf{X}$ consisting of a set of N observations $\{\boldsymbol{x}_i\}_{i=1}^N$, we wish to model the data using a mixture of K Gaussian distributions. The log-likelihood is given by

$$\log \mathcal{L}(\theta) = \log p(\mathbf{X} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right), \tag{1}$$

where $\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the probability density function of the multivariate normal distribution. The log-likelihood can be maximised by applying the EM-algorithm. Initialisation consists in choosing values for the means $\boldsymbol{\mu}_k$, covariances

$\boldsymbol{\Sigma}_k$, and mixing coefficients π_k for each component k. The E-step is to evaluate the probabilities t_{ik} using the current parameter values:

$$t_{ik} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{2}$$

In the M-step, the parameters are re-estimated with the updated probabilities:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} t_{ik} \boldsymbol{x}_i, \quad \boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{N} t_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T, \quad \pi_k = \frac{N_k}{N}, \tag{3}$$

where $N_k = \sum_{i=1}^{N} t_{ik}$. The E and M-steps are alternated repeatedly until convergence is observed in the log-likelihood or parameter values.

2.2 Missing data extension of the EM algorithm

The input data $\mathbf{X}$ is now a set of observations $\{\boldsymbol{x}_i\}_{i=1}^{N}$ such that for each sample there is an index set $O_i \subseteq \{1, \ldots, d\}$ enumerating the observed samples. The indices in the complement set M_i correspond to missing values in the data sample $\boldsymbol{x}_i$. In the case with missing values, the log-likelihood can be written as

$$\log \mathcal{L}(\theta) = \log p(\mathbf{X} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N} \left(\boldsymbol{x}_i^{O_i} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \right) \right) \tag{4}$$

where as a shorthand of notation, $\mathcal{N}(\boldsymbol{x}_i^{O_i} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is also used for the *marginal* multivariate normal distribution probability density of the observed values of $\boldsymbol{x}_i$.

In the EM algorithm, in order to account for the missing data, some additional expectations need to be computed in the E-step. These include the conditional expectations of the missing components of a sample ($\tilde{\boldsymbol{\mu}}_{ik}^{M_i}$) with respect to each Gaussian component k, and their conditional covariance matrices ($\tilde{\boldsymbol{\Sigma}}_{ik}^{MM_i}$). For convenience, we also define corresponding imputed data vectors $\tilde{\boldsymbol{x}}_{ik}$ and full covariance matrices $\tilde{\boldsymbol{\Sigma}}_{ik}$ which are padded with zeros for the known components. Then the E-step is:

$$t_{ik} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i^{O_i} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_i^{O_i} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{5}$$

$$\tilde{\boldsymbol{\mu}}_{ik}^{M_i} = \boldsymbol{\mu}_k^{M_i} + \boldsymbol{\Sigma}_k^{MO_i} (\boldsymbol{\Sigma}_k^{OO_i})^{-1} (\boldsymbol{x}_i^{O_i} - \boldsymbol{\mu}_k^{O_i}), \qquad \tilde{\boldsymbol{x}}_{ik} = \begin{pmatrix} \boldsymbol{x}_i^{O_i} \\ \tilde{\boldsymbol{\mu}}_{ik}^{M_i} \end{pmatrix}, \tag{6}$$

$$\tilde{\boldsymbol{\Sigma}}_{ik}^{MM_i} = \boldsymbol{\Sigma}_k^{MM_i} - \boldsymbol{\Sigma}_k^{MO_i} (\boldsymbol{\Sigma}_k^{OO_i})^{-1} \boldsymbol{\Sigma}_k^{OM_i}, \qquad \tilde{\boldsymbol{\Sigma}}_{ik} = \begin{pmatrix} \mathbf{0}^{OO_i} & \mathbf{0}^{OM_i} \\ \mathbf{0}^{MO_i} & \tilde{\boldsymbol{\Sigma}}_{ik}^{MM_i} \end{pmatrix} \tag{7}$$

The notation $\boldsymbol{\mu}_k^{M_i}$ refers to using only the elements from the vector $\boldsymbol{\mu}_k$ specified by the index set M_i, and similarly for $\boldsymbol{x}_i^{O_i}$, etc. For matrices, $\boldsymbol{\Sigma}_k^{MO_i}$ refers to elements in the rows specified by M_i and columns by O_i, and so on. The expressions for the parameters in equations (6) and (7) originate from the observation that the conditional distribution of the missing components also follows a multivariate normal distribution, with these parameters [2, Thm. 2.5.1].

The M-step remains functionally equivalent, the only changes being that the component means are estimated from the imputed data vectors and the covariance matrix estimate requires an additional term to include the covariances concerning the imputed values.

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{i=1}^{N} t_{ik}\tilde{\boldsymbol{x}}_{ik},\ \ \boldsymbol{\Sigma}_k = \frac{1}{N_k}\sum_{i=1}^{N} t_{ik}\left[(\tilde{\boldsymbol{x}}_{ik}-\boldsymbol{\mu}_k)(\tilde{\boldsymbol{x}}_{ik}-\boldsymbol{\mu}_k)^T + \tilde{\boldsymbol{\Sigma}}_{ik}\right],\ \ \pi_k = \frac{N_k}{N} \tag{8}$$

2.3 Implementation

In our implementation of the EM algorithm with missing data, the means $\boldsymbol{\mu}_k$ are initialised by a random selection of samples from the data (any missing components can be imputed with the sample mean), and the covariances $\boldsymbol{\Sigma}_k$ are initialised with the sample covariance of the data (ignoring samples with missing values). Alternatively, the covariances can be initialised as diagonal matrices, using only the sample variance of each variable.

The number of components is selected according to the Akaike information criterion [1] with the small sample (second-order) bias adjustment [7]. Using the corrected version is crucial, as the number of parameters grows relatively large when increasing the number of components. The corrected Akaike information criterion is a function of the log-likelihood:

$$AIC_C = -2\log\mathcal{L}(\theta) + 2P + \frac{2P(P+1)}{N-P-1} \tag{9}$$

where P is the number of free parameters. $P = Kd + K - 1 + \frac{1}{2}Kd(d+1)$ in the case of full, separate, covariance matrices for each of the K components. With high-dimensional data sets, the number of parameters quickly tends to become larger than the number of available samples when increasing the number of components, and the criterion would not be valid anymore. This effect can be mitigated by imposing restrictions on the structure of the covariance matrices, but this would also make the model less powerful.

3 Distance estimation with missing data

One application for the mixture of Gaussians model is to use it for distance estimation. The problem of estimating distances between samples with missing data is non-trivial, since even perfect imputation (by the conditional expectation) results in biased estimates for the distance. Using additional knowledge about the distribution of the data leads to more accurate estimates.

In the following, we focus on calculating the expectation of the *squared* Euclidean (ℓ^2) distance. Estimating the ℓ^2-norm itself could be feasible, but due to the square-root, the expressions do not simplify and separate as cleanly. Another motivation for directly estimating the squared distance is that many methods for further processing of the distance matrix actually only make use of the squared distances (e.g., RBF and SVM), while others only consider the ranking of the distances (nearest neighbours).

Given two samples $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^d$ with components $x_{i,l}$, $x_{j,l}$ $(1 \le l \le d)$, which may contain missing values, denote by $M_i \subseteq \{1, \dots, d\}$ the set of indices of the missing components for each sample $\boldsymbol{x}_i$. Partition the index set into four parts based on the missing components, and the expression for the squared distance $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$ can be split accordingly:

$$\begin{aligned}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = \sum_{l=1}^{d}(x_{i,l} - x_{j,l})^2 = \sum_{l \notin M_i \cup M_j} (x_{i,l} - x_{j,l})^2 + \sum_{l \in M_j \setminus M_i} (x_{i,l} - x_{j,l})^2 \\ + \sum_{l \in M_i \setminus M_j} (x_{i,l} - x_{j,l})^2 + \sum_{l \in M_i \cap M_j} (x_{i,l} - x_{j,l})^2\end{aligned}$$

The missing values can be modelled as random variables, $X_{i,l}$, $l \in M_i$. Taking the expectation of the above expression, by the linearity of expectation:

$$\begin{aligned}\mathrm{E}\left[\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right] = \sum_{l \notin M_i \cup M_j} (x_{i,l} - x_{j,l})^2 + \sum_{l \in M_j \setminus M_i} \big((x_{i,l} - \mathrm{E}[X_{j,l}])^2 + \mathrm{Var}[X_{j,l}]\big) \\ + \sum_{l \in M_i \setminus M_j} \big((\mathrm{E}[X_{i,l}] - x_{j,l})^2 + \mathrm{Var}[X_{i,l}]\big) \\ + \sum_{l \in M_i \cap M_j} \big((\mathrm{E}[X_{i,l}] - \mathrm{E}[X_{j,l}])^2 + \mathrm{Var}[X_{i,l}] + \mathrm{Var}[X_{j,l}]\big)\end{aligned}$$

In the final summation, it is necessary to consider $X_{i,l}$ and $X_{j,l}$ to be uncorrelated, given the known values of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

It thus suffices to find the expectation and variance of each random variable separately. If the original samples $\boldsymbol{x}_i$ are thought to originate as independent draws from a multivariate distribution, the distributions of the random variables $X_{i,l}$ can be found as the conditional distribution when conditioning their joint distribution on the observed values. Then finding the expected squared distance between two samples reduces to finding the (conditional on the observed values) expectation and variance of each missing component separately. Define $\tilde{\boldsymbol{x}}_i$ to be an imputed version of $\boldsymbol{x}_i$ where each missing value has been replaced by its conditional mean. Define $\sigma_{i,l}^2$ correspondingly as the conditional variance:

$$\tilde{x}_{i,l} = \begin{cases} \mathrm{E}[X_{i,l} \,|\, \boldsymbol{x}_i^{O_i}] & \text{if } l \in M_i, \\ x_{i,l} & \text{otherwise} \end{cases} \qquad \sigma_{i,l}^2 = \begin{cases} \mathrm{Var}[X_{i,l} \,|\, \boldsymbol{x}_i^{O_i}] & \text{if } l \in M_i, \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

With these notations, the expectation of the squared distance can conveniently be expressed as:

$$\mathrm{E}\left[\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right] = \|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j\|^2 + s_i^2 + s_j^2, \quad \text{where} \quad s_i^2 = \sum_{l \in M_i} \sigma_{i,l}^2 \tag{11}$$

This form of the expression particularly emphasises how the uncertainty of the missing values is accounted for. The first term – the distance between imputed samples – already provides an estimate of the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, but including the variances of each imputed component is the deciding factor.

The conditional means and covariances can be calculated using the Gaussian mixture model. These are calculated separately for each component in the

M-step, and it only remains to determine the overall conditional mean and covariance matrix. These are found weighted by the memberships as follows:

$$\tilde{\boldsymbol{x}}_i = \sum_{k=1}^{K} t_{ik} \tilde{\boldsymbol{x}}_{ik}, \qquad \tilde{\boldsymbol{\Sigma}}_i = \sum_{k=1}^{K} t_{ik} \left(\tilde{\boldsymbol{\Sigma}}_{ik} + \tilde{\boldsymbol{x}}_{ik} \tilde{\boldsymbol{x}}_{ik}^T \right) - \tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_i^T \tag{12}$$

The expression for the covariance is found by direct calculation of the second moments. In order to estimate pairwise distances, the conditional variances $\sigma_{i,l}^2 = \tilde{\boldsymbol{\Sigma}}_{i,ll}$ can be extracted from the diagonal of the conditional covariance matrix.

4 Experiments

To study the performance of the algorithm, some simulated experiments are conducted to compare the proposed algorithm to alternate methods on several data sets with two different performance criteria. Starting with a complete data set, values are removed at random with a fixed probability. As the true distances between samples are known, the methods can then be compared on how well they estimate the distances after values have been removed. Five different data sets are selected from the UCI Machine Learning Repository [3]:

Iris Fisher's famous Iris data set. $N = 150$ (samples), $d = 4$ (variables)
Glass Glass identification data set (ignoring id). $N = 214$, $d = 9$
CPU Computer hardware data of relative CPU performance. $N = 209$, $d = 6$
Servo Data from a simulation of a servo system $N = 167$, $d = 4$
Housing Concerns housing values in suburbs of Boston. $N = 506$, $d = 13$

To make distances meaningful, the variables in each data set are standardised to zero mean and unit variance before values are removed. As the problem of pairwise distance estimation is unsupervised, the labels for the samples are ignored.

A total of six different methods are compared:

PDS The Partial Distance Strategy [6]. Calculate the sum of squared differences of the mutually known components and scale to the missing components:

$$\hat{d}_{ij}^2 = \frac{d}{d - |M_i \cup M_j|} \sum_{l \notin M_i \cup M_j} (x_{i,l} - x_{j,l})^2 \tag{13}$$

For samples which have no common known components, the method is not defined. For such pairs, the average of the pairwise distances which were possible to estimate is returned instead.

ICkNNI Incomplete-case k-NN imputation [9]. An improvement of complete-case k-NN imputation, here any sample with a valid missingness pattern is viable nearest neighbour. In accordance to the suggestions in [9], up to $k = 5$ neighbours are considered. The imputation fails whenever there are no samples with valid missingness patterns. For such cases, the missing value is imputed by the sample mean for that variable.

1 Gaussian The expected squared distances as calculated by the proposed algorithm with only one Gaussian component in the mixture. The square root of the result is used to get an estimate of the distance.

Regression imputation Imputation by the conditional expectation when restricted to one Gaussian component. This is equivalent to least-squares linear regression, if the covariance matrix and mean were exactly known.

Mixture model The proposed method, where the number of Gaussian components is selected by the AIC_C criterion.

Imputation only Using the mixture model to conduct non-linear imputation, and calculating distances from the imputed data set.

The methods are evaluated by two different performance criteria. First, the methods are compared by the root mean squared error (RMSE) of all the estimated pairwise distances in the data set,

$$C_1 = \big(\frac{1}{\lambda}\sum_{i>j}(\hat{d}_{ij} - d_{ij})^2\big)^{1/2}$$

where, d_{ij} is the true Euclidean distance between samples i and j calculated without any missing data, and $\hat{d}_{ij}$ is the estimate of the distance provided by each method after removing data. The scaling factor λ is determined so that the average is calculated only over those distances which are estimates, discarding all the cases where the distance can be calculated exactly because neither sample has any missing components: $\lambda = MN - \frac{M(M+1)}{2}$.

A common application for pairwise distances is a nearest neighbour search, and thus we also consider the average (true) distance to the predicted nearest neighbour,

$$C_2 = \frac{1}{N}\sum_{i=1}^{N} d_{i,\mathrm{NN}(i)}, \quad \text{where} \quad \mathrm{NN}(i) = \arg\min_{j\neq i} \hat{d}_{ij}$$

Here, $\mathrm{NN}(i)$ is the nearest neighbour of the ith sample as estimated by the method, and $d_{i,\mathrm{NN}(i)}$ is the true Euclidean distance between the samples as calculated without any missing data. The criterion measures how well the method can identify samples which actually are close in the real data.

Values are removed from the data set independently at a fixed probability p. For each value of p, 100 repetitions are conducted for the Monte Carlo simulation, and simulations are run for value of p of 0.10, 0.25, and 0.50. The EM algorithm is run for 200 iterations, and repeated for a total of 5 times for each number of components. The best solution in terms of log-likelihood is selected. Runs are aborted if a covariance matrix becomes too poorly conditioned (condition number over 10^{12}).

Having 100 repetitions of the same set-up enables the use of statistical significance testing to assess the difference between the mean errors of different methods. The testing is conducted as a two-tailed paired t-test, with a significance level of $\alpha = 0.05$. Comparing the performance of the best method to that of every other method results in a multiple hypothesis scenario, and thus the Bonferroni correction is used to control the error rate.

The average RMSE values for each method are presented in Table 1, and the table also displays the average for the number of selected Gaussians components (K). It can clearly be seen that including the variance terms of equation (11)

Table 1. Average RMSE of estimated pairwise distances, and standard deviations in parenthesis. The best result for each row is underlined, and any results which are not statistically significantly different from the best result are bolded.

		PDS	ICkNNI	1 Gaussian	Regression imputation	Mixture model	Imputation only	Mean K
Iris	10%	0.517	0.287	0.290	0.297	**0.259**	0.268	2.44
	25%	0.763	0.457	0.437	0.465	**0.391**	0.419	2.29
	50%	1.108	0.895	0.763	0.894	**0.736**	0.840	3.28
Glass	10%	0.671	0.458	**0.324**	**0.320**	**0.320**	**0.320**	1.74
	25%	1.146	0.848	**0.649**	**0.649**	**0.652**	**0.660**	1.77
	50%	2.085	1.513	**1.205**	1.287	**1.210**	1.284	1.45
CPU	10%	0.808	**0.510**	0.551	0.534	0.545	**0.534**	3.82
	25%	1.280	**0.827**	0.844	0.854	**0.845**	**0.848**	3.34
	50%	1.944	1.346	**1.271**	1.346	1.323	1.367	2.28
Servo	10%	0.672	0.515	**0.427**	0.486	**0.428**	0.487	1.68
	25%	0.951	0.692	**0.553**	0.683	0.558	0.683	1.73
	50%	1.217	1.088	**0.766**	1.115	0.785	1.102	2.03
Housing	10%	0.668	0.438	**0.428**	0.447	**0.432**	0.449	1.28
	25%	1.179	0.808	**0.687**	0.761	**0.687**	0.757	1.18
	50%	2.278	1.589	**1.066**	1.321	**1.081**	1.317	1.18

tends to lead to an improvement in the accuracy compared to only conducing imputation. For the Iris data set, the mixture model is notably more accurate than any other method, whereas for the other data sets the errors are on par with using one Gaussian for the model. This suggests that either using a single Gaussian is enough to sufficiently model the data, or there are insufficient samples to accurately fit a model of several Gaussians.

Table 2 shows the corresponding performances in terms of the true distance to the predicted nearest neighbour. While the trends are similar, it is interesting to note that in terms of this performance measure the mixture model appears somewhat more capable than only looking at the RMSE. This seems to suggest that the mixture model is more accurate when estimating small distances.

5 Conclusions

The problem of estimating distances in a data set with missing values can be reduced to finding the conditional means and variances separately for each missing value. Having a Gaussian mixture model of the distribution of the data enables these quantities to be estimated. In order to fit the mixture model, certain extensions to the standard EM algorithm are presented in Section 2.

The combination of these ideas provides for a method to estimate distances, and the simulations in Section 4 show that the method is competitive, if not better, than alternative methods in terms of accuracy.

For future work, it remains to investigate the most effective ways to extend the method to high-dimensional cases where the number of parameters would exceed the number of samples. Furthermore, it will be interesting to study the influence of the distance estimation when used with machine learning methods such as SVM and RBF neural networks.

Table 2. Average of the mean distance to the estimated nearest neighbour, and standard deviations in parenthesis. The best result for each row is underlined, and any results which are not statistically significantly different from the best result are bolded.

		PDS	ICkNNI	1 Gaussian	Regression imputation	Mixture model	Imputation only	Mean K
	10%	0.670	0.423	0.385	0.418	$\underline{\mathbf{0.379}}$	0.413	2.44
Iris	25%	1.172	0.616	0.530	0.612	$\underline{\mathbf{0.506}}$	0.596	2.29
	50%	1.509	1.063	0.857	1.040	$\underline{\mathbf{0.839}}$	1.024	3.28
	10%	1.256	1.020	**0.945**	0.958	$\underline{\mathbf{0.943}}$	0.957	1.74
Glass	25%	1.998	1.378	$\underline{\mathbf{1.203}}$	1.241	**1.204**	1.250	1.77
	50%	2.861	2.147	$\underline{\mathbf{1.747}}$	1.890	1.768	1.912	1.45
	10%	0.852	0.631	0.610	0.622	$\underline{\mathbf{0.592}}$	0.626	3.82
CPU	25%	1.414	0.963	0.907	0.952	$\underline{\mathbf{0.862}}$	0.944	3.34
	50%	1.809	1.499	$\underline{\mathbf{1.320}}$	1.462	**1.327**	1.475	2.28
	10%	1.437	0.976	$\underline{\mathbf{0.818}}$	0.936	**0.819**	0.939	1.68
Servo	25%	1.851	1.286	$\underline{\mathbf{1.035}}$	1.239	**1.038**	1.244	1.73
	50%	2.041	1.774	$\underline{\mathbf{1.463}}$	1.733	1.509	1.759	2.03
	10%	1.270	**1.046**	1.045	**1.045**	$\underline{\mathbf{1.039}}$	**1.043**	1.28
Housing	25%	2.067	1.548	1.420	1.448	$\underline{\mathbf{1.415}}$	1.448	1.18
	50%	3.659	2.746	**2.094**	2.221	$\underline{\mathbf{2.088}}$	2.220	1.18

References

1. Akaike, H.: A new look at the statistical model identification. Automatic Control, IEEE Transactions on 19(6), 716–723 (Dec 1974)
2. Anderson, T.W.: An Introduction to Multivariate Statistical Analysis. Wiley-Interscience, New York, third edn. (2003)
3. Asuncion, A., Newman, D.J.: UCI machine learning repository (2011), http://archive.ics.uci.edu/ml/, University of California, Irvine, School of Information and Computer Sciences
4. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
5. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B (Methodological) 39(1), pp. 1–38 (1977)
6. Dixon, J.K.: Pattern recognition with partly missing data. Systems, Man and Cybernetics, IEEE Transactions on 9(10), 617–621 (Oct 1979)
7. Hurvich, C.M., Tsai, C.L.: Regression and time series model selection in small samples. Biometrika 76(2), 297–307 (1989)
8. Little, R.J.A., Rubin, D.B.: Statistical Analysis with Missing Data. Wiley-Interscience, second edn. (2002)
9. Van Hulse, J., Khoshgoftaar, T.M.: Incomplete-case nearest neighbor imputation in software measurement data. Information Sciences (2011), in Press, Corrected Proof

Enhancement Learning in Functional Relevance Learning Vector Quantization

T. Villmann*, M. Kästner, D. Nebel, and M. Riedel

Computational Intelligence Group,
University of Applied Sciences Mittweida,09648 Mittweida, Germany
{villmann,kaestner,nebel,riedel}@hs-mittweida.de

Abstract. We present a modification of relevance and matrix learning for the generalized learning vector quantization algorithm. This modification allows an enhanced learning of the relevance or matrix weights by neighborhood cooperativeness between the relevance weights. Thus relvance learning of a weight profits from the weight neighbors leading to an accelerated adaptation and smooth relevance profiles or relevamce matrices. Further we show that this enhancement scheme can be seen as a new dissimilarity measure in standard generalized learning vector quantization such that theoretical aspects like margin analysis remain valid.

1 Introduction

Relevance learning in learning vector quantization (GRLVQ) is a very well established and powerful method in prototype based classification for weighting of data attributes, which are important for the classification of respective data [5]. The automatic weighting of the data dimensions in online learning is obtained as a gradient learning scheme of a cost function based on that of the generalized learning vector quantization algorithm (GLVQ) proposed by SATO&YAMADA in [13], which assigns to each attribute a relevance weight indicating the importance to separate the classes in the given problem. As the basis, the GLVQ can be seen as a theoretically proven generalization of the standard learning vector quantization algorithms (LVQ) introduced by T. KOHONEN [8]. All relevance weights form the relevance profile, however, their adaptation is performed independently in GRLVQ. Extensions of the GRLVQ method consider some classification dependent correlation matrices of the data attributes to obtain better classifications (GMLVQ,[15,16]). Recent developments concern the application of relevance learning to specific data, in particular, functional data like spectra or time series, taking their specific functional property into account [7]. Functional data are vectorized representations of continuous functions such that their relevance profile can be taken as functions, too. In particular, if the functional data are representing smooth functions, the resulting relevance profile should be also smooth. In the mentioned approach [7] the (functional) relevance profile was determined by a superposition of adaptive smooth basis functions, for example Gaussians or Lorentzians with different widths end positions. This functional approach of relevance learning usually converges faster because of the lower degree of free parameters to be optimized and, obviously, carries some kind of inherent regularization according to the choice of the set of used basis functions. As it is known

* *corresponding author*, email: thomas.villmann@hs-mittweida.de

from radial basis function networks or Parzen windows estimators in density estimating, the learning of Gaussians is not trivial but suffers from unstable behavior [6]. Another drawback is that the set of adaptive basis functions has to be chosen in advance.

In this contribution we propose a different approach to obtain smooth relevance profiles for functional data classification based on GLVQ. Again, the functional property of the data is assumed to motivate the adaptation of the relevance profile. The new approach does not handle the relevance profile entries independently as in GRLVQ. It introduces a neighborhood cooperativeness between them under the assumption of their functional correspondence. However, it does not require the determination of some kind of an a priori given set of basis functions, to compose the relevance profile from that. The neighborhood cooperativeness leads to a faster convergence. Further, the achived profiles are smoother compared to standard relevance and matrix learning because the neighborhood cooperativeness can also be seen as a kind of regularization mechanism in relevance or matrix learning.

2 Relevance Learning and Metric Adaptation in Learning Vector Quantization

Relevance learning in learning vector quantization is a paradigm that weights the data attributes (dimensions) in such a way that a given classification problem can be solved better than using standard Euclidean metric. We suppose that the data are given as vectors $\mathbf{v} \in V \subseteq \mathbb{R}^n$, and the prototypes of the LVQ model are the set $W = \{\mathbf{w}_k \in \mathbb{R}^n, k = 1 \ldots M\}$. Each data vector $\mathbf{v}$ of the training data belongs to a class $x_{\mathbf{v}} \in \mathcal{C} = \{1, \ldots, C\}$. The prototypes have labels $y_{\mathbf{w}_k} \in \mathcal{C}$ indicating their responsibility to the several classes. The dissimilarity of data and prototypes is judged by the dissimilarity $d(\mathbf{v}, \mathbf{w}) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ given in the data space V and frequently chosen as the (quadratic) Euclidean metric. Yet, it is only required that the dissimilarity measure is differentiable with respect to the second argument, but not necessarily symmetric [12].

Standard LVQ distributes the prototypes in such a way that the classification error is optimized but in a heuristically manner [8]. The generalization thereof, the generalized LVQ (GLVQ) minimizes an approximated classification error based on a stochastic gradient descent scheme [13]. The cost function minimized by GLVQ is

$$E(W) = \frac{1}{2} \sum_{\mathbf{v} \in V} f(\mu(\mathbf{v})) \tag{1}$$

where the function

$$\mu(\mathbf{v}) = \frac{d^+(\mathbf{v}) - d^-(\mathbf{v})}{d^+(\mathbf{v}) + d^-(\mathbf{v})} \tag{2}$$

is the classifier function with $d^+(\mathbf{v}) = d(\mathbf{v}, \mathbf{w}^+)$ denotes the dissimilarity between the data vector $\mathbf{v}$ and the closest prototype $\mathbf{w}^+$ with the same class label $y_{\mathbf{w}^+} = x_v$, and $d^-(\mathbf{v}) = d(\mathbf{v}, \mathbf{w}^-)$ is the dissimilarity degree for the best matching prototype $\mathbf{w}^-$ with a class label $y_{\mathbf{w}^-}$ different from $x_{\mathbf{v}}$. The transformation function f is a monotonically increasing function usually chosen as sigmoidal or the identity function. A typical sigmoidal choice is the Fermi function

$$f(x) = \frac{1}{1 + a \cdot \exp\left(-\frac{(x - x_0)^2}{2\varsigma^2}\right)} \tag{3}$$

with $x_0 = 0$ and $a = 1$ as standard parameter values. The ς-parameter allows a control of the sensitivity of the classifier at class borders but usually fixed with $\varsigma = 1$. Learning of $\mathbf{w}^+$ and $\mathbf{w}^-$ is performed in GLVQ by the stochastic gradient with respect to the cost function $E(W)$ for a given data vector $\mathbf{v}$ according to

$$\frac{\partial_S E(W)}{\partial \mathbf{w}^+} = \xi^+ \cdot \frac{\partial d^+}{\partial \mathbf{w}^+} \text{ and } \frac{\partial_S E(W)}{\partial \mathbf{w}^-} = \xi^- \cdot \frac{\partial d^-}{\partial \mathbf{w}^-} \tag{4}$$

with

$$\xi^+ = f' \cdot \frac{2 \cdot d^-(\mathbf{v})}{\left(d^+(\mathbf{v}) + d^-(\mathbf{v})\right)^2} \tag{5}$$

and

$$\xi^- = -f' \cdot \frac{2 \cdot d^+(\mathbf{v})}{\left(d^+(\mathbf{v}) + d^-(\mathbf{v})\right)^2}. \tag{6}$$

For the quadratic Euclidean metric we simply have the derivative $\frac{\partial d^\pm(\mathbf{v})}{\partial \mathbf{w}^\pm} = -2\left(\mathbf{v} - \mathbf{w}^\pm\right)$ realizing a vector shift of the prototypes.

2.1 Standard Relevance Learning and Metric Adaptation

Standard relevance learning replaces the quadratic Euclidean metric in GLVQ by a parametrized dissimilarity

$$d_\Lambda(\mathbf{v}, \mathbf{w}) = (\mathbf{v} - \mathbf{w})^\top \Lambda (\mathbf{v} - \mathbf{w}) \tag{7}$$

with Λ being a positive semi-definite diagonal matrix [5]. The diagonal elements $\lambda_i = \sqrt{\Lambda_{ii}}$ form the relevance profile weighting the data dimensions. During the learning phase, the relevance parameter λ_i are adapted in addition to the prototype update, again as a stochastic gradient descent

$$\frac{\partial E_S(W)}{\partial \lambda_j} = \xi^+ \cdot \frac{\partial d_\Lambda^+}{\partial \lambda_j} + \xi^- \cdot \frac{\partial d_\Lambda^-}{\partial \lambda_j} \tag{8}$$

of the cost function and subsequent normalization such that $\sum_i \lambda_i^2 = \sum_i \Lambda_{i,i} = 1$. The respective algorithm is denoted as GRLVQ. The obvious generalization of this scheme is to generalize the relevance metric (7) to be a positive semi-definite quadratic form choosing $\Lambda = \Omega^\top \Omega$ such that (7) can be written in the form

$$d_\Omega(\mathbf{v}, \mathbf{w}) = \left(\Omega(\mathbf{v} - \mathbf{w})\right)^2 \tag{9}$$

with $\Omega \in \mathbb{R}^{m \times n}$ [2,15,16]. The resulting derivatives $\frac{\partial d^\pm}{\partial \mathbf{w}^\pm}$ for the prototype update in (4) are obtained as $\frac{\partial d^\pm(\mathbf{v})}{\partial \mathbf{w}^\pm} = -2\Lambda\left(\mathbf{v} - \mathbf{w}^\pm\right)$ accompanied by the Ω-update

$$\frac{\partial_S E(W)}{\partial \Omega_{r_1,r_2}} = \xi^+ \cdot \frac{\partial d_\Omega(\mathbf{v}, \mathbf{w}^+)}{\partial \Omega_{r_1,r_2}} + \xi^- \cdot \frac{\partial d_\Omega(\mathbf{v}, \mathbf{w}^-)}{\partial \Omega_{r_1,r_2}} \tag{10}$$

with

$$\frac{\partial d_\Omega(\mathbf{v}, \mathbf{w})}{\partial \Omega_{r_1,r_2}} = 2\left[\mathbf{\Omega}(\mathbf{v} - \mathbf{w})\right]_{r_1} [\mathbf{v} - \mathbf{w}]_{r_2} \tag{11}$$

for the determination of the Ω-update. Again, a normalization has to take place to ensure $\sum_{i,j} \Omega_{i,j}^2 = \sum_i \Lambda_{i,i} = 1$. We refer to this matrix variant as GMLVQ.

2.2 Functional Relevance Learning and Metric Adaptation

In functional vector quantization the data vectors are representations of functions $v(t)$ with given values $v_i = v(t_i)$. Examples are hyper-spectra in remote sensing, time series or funtion profiles.. In case of GRLVQ and GMLVQ this may lead to a huge number of relevance or matrix parameter to be adjusted during the metric adaptation[1]. Yet, if the data vector are discrete representations of functions, both relevance and matrix learning can make use of this functional property to reduce the number of parameters in relevance learning. More precisely, we assume in the following that data vectors $\mathbf{v} = (v_1, \ldots, v_n)^T$ are representations of functions $v(t)$ with given values $v_i = v(t_i)$.

In *functional relevance learning* the relevance profile in (7) is also interpreted as a function $\lambda(t)$ with $\lambda_j = \lambda(t_j)$ in agreement with the data vector interpretation. In the recently proposed *generalized functional relevance* LVQ (GFRLVQ) [7], the relevance function $\lambda(t)$ is taken as a superposition

$$\lambda(t) = \sum_{l=1}^{K} \beta_l \mathcal{K}_l(t) \tag{12}$$

of simple basis functions $\mathcal{K}_l$ depending on only a few parameters with the restriction $\sum_{l=1}^{K} \beta_l = 1$. Famous examples are standard Gaussians or Lorentzians:

$$\mathcal{K}_l(t) = \frac{1}{\sigma_l \sqrt{2\pi}} \exp\left(-\frac{(t-\Theta_l)^2}{2\sigma_l^2}\right) \tag{13}$$

and

$$\mathcal{K}_l(t) = \frac{1}{\eta_l \pi} \frac{\eta_l^2}{\eta_l^2 + (t-\Theta_l)^2}\ , \tag{14}$$

respectively.

Relevance learning in GFRLVQ takes place by adaptation of the parameters β_l, Θ_l,σ_l and η_l, respectively. For this purpose, again the stochastic gradient scheme is applied as before, such that for an arbitrary parameter ϑ_l of the functional relevance profile (12) we have

$$\frac{\partial_S E}{\partial \vartheta_l} = \xi^+ \cdot \frac{\partial d_\Lambda^+}{\partial \vartheta_l} + \xi^- \cdot \frac{\partial d_\Lambda^-}{\partial \vartheta_l}$$

in complete analogy. Obviously, this idea can be easily transferred to matrix learning assuming here that the matrix Ω in (9) is a discrete representation of a continuous function $\Omega(t_1, t_2)$ [11]. We again assume a superposition

$$\boldsymbol{\Omega}(t_1, t_2) = \sum_{l=1}^{K} \beta_l \mathbf{K}_l(t_1, t_2) \tag{15}$$

[1] For GMLVQ the number of free parameters scales with the square of the number of input dimensions although a slight self-regularizing mechanism leads to the fact that the effective number of free parameters is decreased, because their is a weak tendency of the (squared) matrix Ω to converge to a degenerated state such that the columns represent the first eigenvector of the data covariance matrix [1,14].

of two-dimensional basis functions $\mathbf{K}_l(t_1,t_2)$, such that we have

$$\mathbf{\Lambda}(t_1,t_2)=\int\mathbf{\Omega}(t_1,t)\,\mathbf{\Omega}(t,t_2)\,dt$$

and, therefore,

$$\mathbf{\Lambda}(t_1,t_2)=\sum_{l=1}^{K}\sum_{m=1}^{K}\beta_l\beta_m\cdot\int\mathbf{K}_l(t_1,t)\cdot\mathbf{K}_m(t,t_2)\,dt\ . \tag{16}$$

The basis functions $\mathbf{K}_l(t_1,t_2)$ are now two-dimensional. For the (symmetric) Gaussian example we have

$$\mathbf{K}_l(t_1,t_2)=\frac{1}{\sigma_{1,l}\cdot\sigma_{2,l}\cdot 2\pi}\exp\left(-\left(\frac{(t_1-\Theta_{1,l})^2}{2\sigma_{1,l}^2}+\frac{(t_2-\Theta_{2,l})^2}{2\sigma_{2,l}^2}\right)\right) \tag{17}$$

whereas for the (symmetric) Lorentzian we get

$$\mathbf{K}_l(t_1,t_2)=\frac{1}{\eta_{1,l}\cdot\eta_{2,l}\cdot\pi^2}\left(\frac{\eta_{1,l}^2}{\eta_{1,l}^2+(t_1-\Theta_{1,l})^2}\cdot\frac{\eta_{2,l}^2}{\eta_{2,l}^2+(t_2-\Theta_{2,l})^2}\right). \tag{18}$$

Yet, numerical simulation have shown that simpler handling can be obtained if a factorized version of (15) is applied instead without significant loss in performance [11]. This factorization approach is based on the consideration about the structure of the matrix Ω in GMLVQ, see footnote 1.

2.3 Gaussian Enhancement Learning for Functional Relevance and Matrix Learning

Although functional (matrix) relevance learning shows stable dynamic and good performance, the learning of the parameters like the position, the widths and the weights of the basis functions is difficult and requires a good fine tuning of the learning parameters [11]. This behavior is also known from radial basis function networks, where the determination of the parameters for the radial basis functions is crucial [6]. Therefore, a more robust alternative for functional (matrix) relevance learning is mandatory but keeping the idea to take the functional character of data into account.

2.3.1 Enhancement of Relevance Learning The idea presented here is to enhance the learning of the relevance weights around a certain data dimension according by an neighborhood term. This term is chosen as a Gaussian decay as it is known from neighborhood learning in self-organizing maps (SOM, [8]) or neural gas (NG, [10]), but there applied to neighborhood cooperativeness between prototypes.. More specific, we introduce a Gaussian enhancement function

$$h_{\sigma_\lambda}(i,j)=\exp\left(\frac{-(i-j)^2}{2\sigma_\lambda^2}\right) \tag{19}$$

for updating the relevance weights λ_i of around the data dimension j. Then the relevance enhancement update should become

$$\frac{\partial d^{\delta}(\mathbf{v},\mathbf{w})}{\partial \lambda_i} = \sum_{j=1}^{n} h_{\sigma_\lambda}(i,j) \cdot [\mathbf{v}-\mathbf{w}]_j^2 \,. \tag{20}$$

This update corresponds to a dissimilarity measure

$$d_\lambda^{\delta}(\mathbf{v},\mathbf{w}) = \sum_{i=1}^{n} \delta(i,\mathbf{v},\mathbf{w},\lambda) \tag{21}$$

with local distortions

$$\delta(i,\mathbf{v},\mathbf{w},\lambda) = \sum_{j=1}^{n} \lambda_i \cdot h_{\sigma_\lambda}(i,j) \cdot [\mathbf{v}-\mathbf{w}]_j^2 \tag{22}$$

to be used in the cost function (1).

The value σ_λ describes the influence range of the enhancement learning. The idea is illustrated in Fig. 1

2.3.2 Matrix Learning In analogy to the enhanced relevance learning we introduce for matrix learning of $\Lambda = \Omega^\top \Omega$ an enhancement matrix according to

$$H = \{h_{\sigma_\lambda}\}_{s,i=1}^{n} \in \mathbb{R}^{n\times n}, \tag{23}$$

based on the enhancement function (19). Hence, the dissimilarity measure becomes

$$d_\Omega^{\delta}(\mathbf{v},\mathbf{w}) = \sum_{i=1}^{m} \left(\sum_{k=1}^{n} \Omega_{i,k} \sum_{s=1}^{n} h_{\sigma_\lambda}(s,k)\,[\mathbf{v}-\mathbf{w}]_s \right)^2 \tag{24}$$

for $\Omega \in \mathbb{R}^{m\times n}$ instead of (9). Setting now $\delta_k = \sum_{s=1}^{n} H_{s,k}\,[\mathbf{v}-\mathbf{w}]_s$ we easily calculate

$$\begin{aligned} d_\Omega^{\delta}(\mathbf{v},\mathbf{w}) &= \sum_{i=1}^{m} \left(\sum_{k=1}^{n} \Omega_{i,k}\delta_k \right)^2 \\ &= \left(\Omega H (\mathbf{v}-\mathbf{w})\right)^2 . \end{aligned} \tag{25}$$

Thus, enhancement learning of the matrix corresponds to a matrix multiplication of Ω with the enhancement matrix H. To emphasize this more general description we denote $d_\Omega^{\delta}(\mathbf{v},\mathbf{w})$ by $d_\Omega^{H}(\mathbf{v},\mathbf{w})$. The derivatives for the enhanced matrix update are determined by

$$\frac{\partial d_\Omega^{H}(\mathbf{v},\mathbf{w})}{\partial \Omega_{a,b}} = 2\,[H(\mathbf{v}-\mathbf{w})]_b\,[\Omega H(\mathbf{v}-\mathbf{w})]_a \,, \tag{26}$$

which is similar to the usual determination (11).

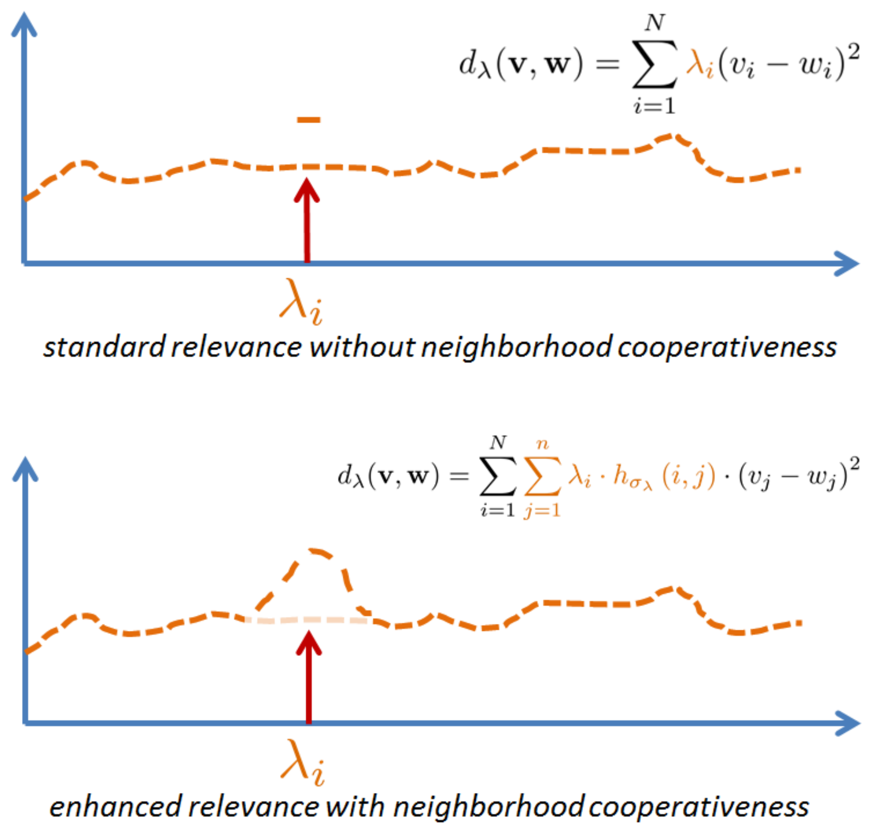

$d_\lambda(\mathbf{v},\mathbf{w}) = \sum_{i=1}^{N} \lambda_i (v_i - w_i)^2$
λ_i
standard relevance without neighborhood cooperativeness
$d_\lambda(\mathbf{v},\mathbf{w}) = \sum_{i=1}^{N} \sum_{j=1}^{n} \lambda_i \cdot h_{\sigma_\lambda}(i,j) \cdot (v_j - w_j)^2$
λ_i
enhanced relevance with neighborhood cooperativeness

Fig. 1. Illustration of the idea of the enhancement learning for relevance weights λ_i including a neighborhood cooperativeness (top - standard relevance learning without neighborhood learning; bottom - enhanced relevance learning.

3 Exemplary Application

To verify the e GMLVQ we choose a well known real world example the Indian Pine data set [9]. The whole data set was generated by means of an AVIRIS sensor capturing an area with 145×145 pixels in the Indian Pine test site in the northwest of Indiana. The spectrometer operates in the visible and mid-infrared wavelength range ($0.4 - 2.4$nm) with $n = 220$ equidistant bands. The area include 16 different kinds of forest or other natural perennial vegetation and some non-agriculture sectors. The latter one we removed from the data set as usual. Additionally, we remove 20 wavelengths, mainly affected by water content. Finally all spectra were normalized according to the l_2-norm. This overall preprocessing is usually applied for this data set [9].

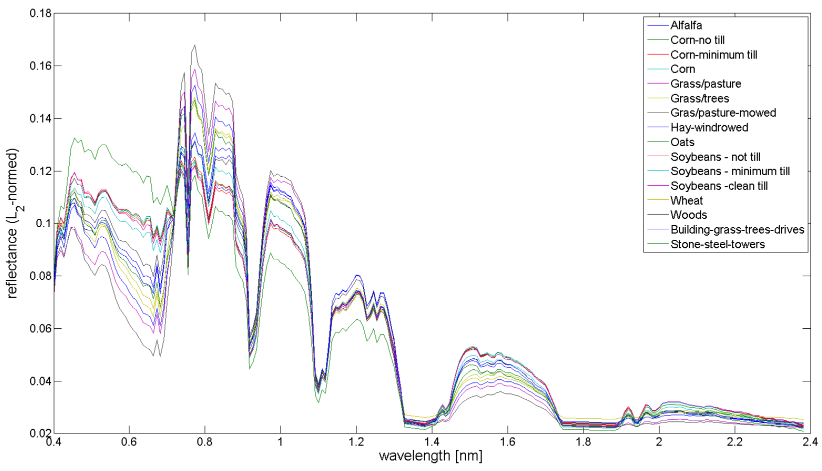


Fig. 2. Mean spectra of the 16 ground cover classes of the Indian Pine data set [9]. The spectra show relative smooth shapes according the functional character of the data.

The mean spectra of the 16 classes are depicted in Fig.2. We observe smooth mean spectra emphasizing their functional character.

The data set of overall $N = 145 \times 145$ data points was divided into 25% of training data points per class and the remaining for test. The number of prototypes per class was chosen according to the class distributions, because of the high variance in the number of pixel per class. Altogether we used 108 prototypes. With the standard GMLVQ and $m = 11$, i.e. $\Omega \in \mathbb{R}^{m \times n}$, we achieved for the test data set an accuracy of 74.1 % (training: 81.1%). The standard functional GFMLVQ obtains 74.0 % for test and 67.7 % for training, where $K = 20$ Gaussian basis functions (17) were applied for the matrix decomposition of Ω according to (15). The reduced accuracy compared to standard GMLVQ shows the difficulty to learn an appropriate matrix representation by the Gaussians. This effect can be observed also in the visualization of the matrix $\Lambda = \Omega^\top \Omega$. Significant correlations are mainly obtained for lower wavelengthes, see Fig. 3.

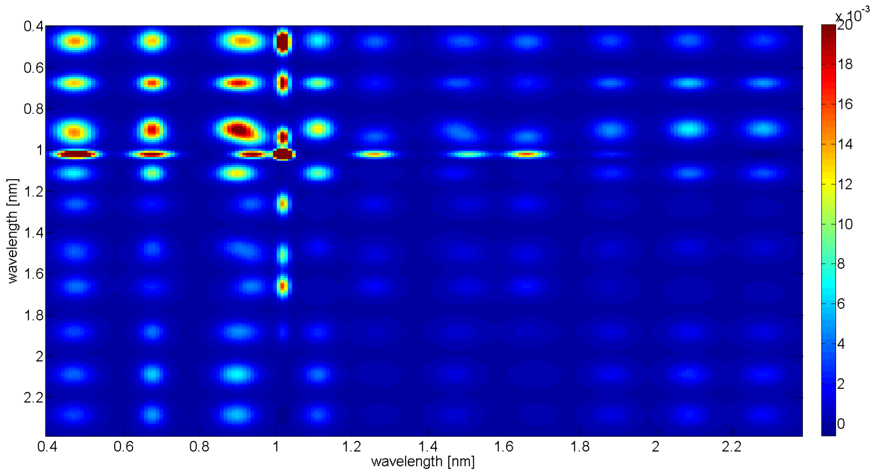


Fig. 3. Visualization of matrix $\Lambda = \Omega^\top \Omega$ for the GFMLVQ. Significant correlations are primarily obtained for lower wavelengths.

This is in contrast to the matrix Λ obtained for standard GMLVQ, see Fig. 4.

GMLVQ also takes correlations between higher wavelengths into account for classification, which results in significant better accuracy. However, the correlation matrix Λ for GMLVQ is not smooth as we would expect from the smooth shape of the data, compare to Fig. 2. Applying now the eGMLVQ a drastically increased test accuracy of 81.7% is achieved with a slightly improved training accuracy 85.0%, both compared to GMLVQ. The obtained correlation matrix Λ is visualized in Fig. 5.

We recognize that for eGMLVQ the matrix is smoother compared to that from GMLVQ, which demonstrates the regularizing ability of the enhancement learning. This property can also concluded from the decreased discrepancy between training and test error compared to GMLVQ and GFMLVQ. Moreover, the resulted correlation matrix also takes correlations between higher wavelengths into account as GMLVQ, which is, however, in contrast to GFMLVQ ignoring these correlations. Moreover, we observe a speedup in matrix learning convergence for eGMLVQ compared to GMLVQ emphasizing the neighborhood cooperativeness learning aspect as a stabilizing learning behavior, which also known from SOM and NG.

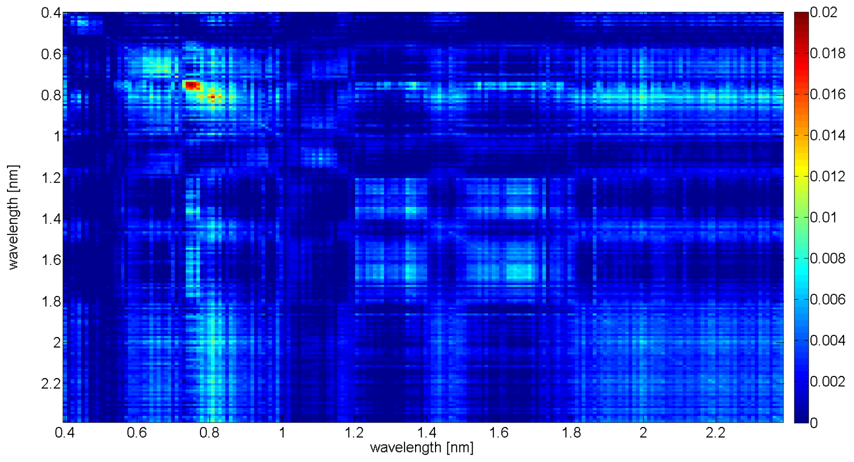


Fig. 4. Visualization of matrix $\Lambda = \Omega^\top \Omega$ for the GMLVQ. Significant correlations are obtained for a wide range of wavelengths and are not restricted to the lower wavelengthes like in GFMLVQ, see Fig. 3. However, the correlation matrix Λ is not smooth as expected for smooth data like the spectra in Fig. 2.

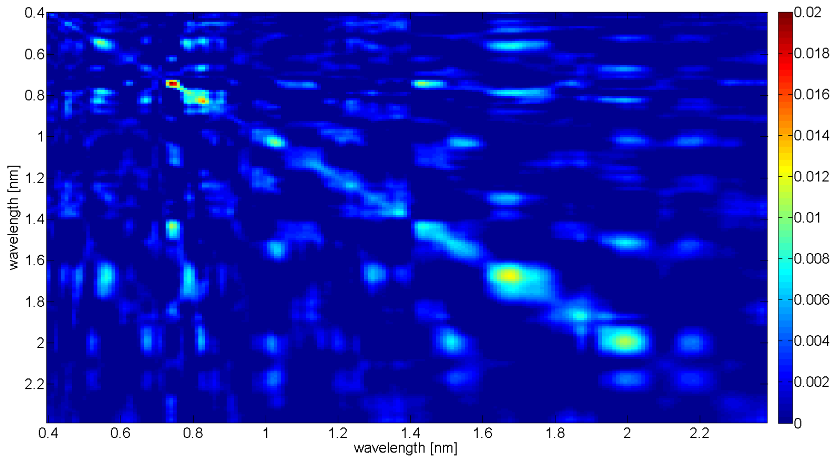


Fig. 5. Visualization of matrix $\Lambda = \Omega^\top \Omega$ for the eGMLVQ. Significant correlations are especially arround the diagonal and not restricted to the lower wavelengthes like in GFMLVQ, see Fig. 3. Yet, the matrix is smoother compared to that from GMLVQ, Fig. 4.

4 Discussion and concluding remarks

We introduced in this paper an enhancement scheme for relevance and matrix learning in learning vector quantization in case of functional data. The underlying idea is the utilization of the functional character of the data, in particular its domain correlation, for enhancement of matrix learning by neighborhood cooperativeness. This leads to a speedup in matrix learning and the enhancement model shows regularization effects obtaining smoother relevance profiles or matrices. Further we have shown that the enhancement learning can be translated into usual GLVQ learning with modified dissimilarities d_λ^δ and d_Ω^H, respectively. Thus, margin propositions from standard GRLVQ and GMLVQ remain valid [3]. Moreover, the dissimilarity d_λ^δ could be related to an earlier approach used in splice site recognition, i.e. the locality improved kernel (LIK) [4]. The difference is that Gaussians are defined on an unlimited domain and only operational during relevance adaptation, while the LIK is focusing on *local triangular convolutions* between attributes (data dimensions).

So far Gaussian neighborhood functions were investigated for enhancement learning. Looking at eq. (25) we can draw more general conclusions: The matrix H can be seen in the dissimilarity measure $d_\Omega^H(\mathbf{v},\mathbf{w})$ as a kind of structural expert knowledge about the data and their dissimilarity relations. For example, asymmetric choices of H could reflect aspects of time series with knowledge about the past. However, one has to take care for such modifications of H regarding the resulting properties of $d_\Omega^H(\mathbf{v},\mathbf{w})$ to fulfill at least minimum standards of dissimilarity measures [12].

References

1. M. Biehl, K. Bunte, F.-M. Schleif, P. Schneider, and T. Villmann. Large margin discriminative visualization by matrix relevance learning. In H. Abbass, D. Essam, and R. Sarker, editors, *Proc. of the International Joint Conference on Neural Networks (IJCNN), Brisbane*, pages 1873–1880, Los Alamitos, 2012. IEEE Computer Society Press.
2. K. Bunte, P. Schneider, B. Hammer, F.-M. S. T. Villmann, and M. Biehl. Limited rank matrix learning, discriminative dimension reduction and visualization. *Neural Networks*, 26(1):159–173, 2012.
3. B. Hammer, M. Strickert, and T. Villmann. Relevance LVQ versus SVM. In L. Rutkowski, J. Siekmann, R. Tadeusiewicz, and L. Zadeh, editors, *Artificial Intelligence and Soft Computing (ICAISC 2004)*, Lecture Notes in Artificial Intelligence 3070, pages 592–597. Springer Verlag, Berlin-Heidelberg, 2004.
4. B. Hammer, M. Strickert, and T. Villmann. Prototype based recognition of splice sites. In U. Seiffert, L. Jain, and P. Schweitzer, editors, *Bioinformatic using Computational Intelligence Paradigms*, pages 25–56. Springer-Verlag, 2005.
5. B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002.
6. S. Haykin. *Neural Networks. A Comprehensive Foundation*. Macmillan, New York, 1994.
7. M. Kästner, B. Hammer, M. Biehl, and T. Villmann. Functional relevance learning in generalized learning vector quantization. *Neurocomputing*, 90(9):85–95, 2012.
8. T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995. (Second Extended Edition 1997).
9. D. Landgrebe. *Signal Theory Methods in Multispectral Remote Sensing*. Wiley, Hoboken, New Jersey, 2003.
10. T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. 'Neural-gas' network for vector quantization and its application to time-series prediction. *IEEE Trans. on Neural Networks*, 4(4):558–569, 1993.
11. D. Nebel and M. Riedel. Generalized functional matrix learning vector quantization. Master's thesis, University of Applied Sciences Mittweida, Germany, 2012.
12. E. Pekalska and R. Duin. *The Dissimilarity Representation for Pattern Recognition: Foundations and Applications*. World Scientific, 2006.
13. A. Sato and K. Yamada. Generalized learning vector quantization. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8. Proceedings of the 1995 Conference*, pages 423–9. MIT Press, Cambridge, MA, USA, 1996.
14. P. Schneider, K. Bunte, H. Stiekema, B. Hammer, T. Villmann, and M. Biehl. Regularization in matrix relevance learning. *IEEE Transactions on Neural Networks*, 21(5):831–840, 2010.
15. P. Schneider, B. Hammer, and M. Biehl. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21:3532–3561, 2009.
16. P. Schneider, B. Hammer, and M. Biehl. Distance learning in discriminative vector quantization. *Neural Computation*, 21:2942–2969, 2009.

Discriminative probabilistic prototype based models in kernel space

Daniela Hofmann, Andrej Gisbrecht, and Barbara Hammer

CITEC centre of excellence, Bielefeld University, Germany
{dhofmann|agisbrec|bhammer}@techfak.uni-bielefeld.de

Abstract. Robust soft learning vector quantization (RSLVQ) constitutes a supervised prototype-based classification scheme which models data by a mixture of Gaussians representing the classes. Training optimizes the likelihood ratio of the model and since the class centers can be inspected directly, it is interpretable. While offering intuitive and flexible classifiers for standard vectorial data, RSLVQ cannot be used if more general data formats are given. We propose an extension towards a kernelized variant which, together with standard preprocessing, makes the method applicable to data described by pairwise similarities only. The resulting model, kernel RSLVQ, represents prototypes implicitly as linear combinations in feature space only such that direct interpretability is lost. We propose to approximate prototypes by the k nearest exemplars, this way arriving at an efficient and interpretable discriminative model for general similarity data. The suitability is tested on a couple of benchmarks.

1 Introduction

Machine learning techniques such as neural networks, self-organizing maps, support vector machines, or Bayesian modeling constitute standard techniques in many application areas ranging from time series forecasting in the financial area, biomedical applications, up to robotics. Relying on such models, it is often possible to infer a classification or regression prescription with excellent accuracy directly from the data. In recent years, a paradigm shift can be observed in several application areas: due to more and more complex learning scenarios and application domains, often, not only an excellent classification or prediction is aimed at, but interpretability of the model constitutes another vital aspect [24]. One prominent example for this demand can be found in the area of biomedical applications, where e.g. a medical test should be substantiated by relevant biomarkers to accompany the results by a medical explanation, and to be capable of developing efficient high-throughput diagnostic technology.

In machine learning, many state-of-the-art techniques act as black boxes such that a direct interpretation is not possible. Examples are k-nearest neighbor classifiers, which are particularly suited if storage is not an issue, but instantaneous learning is required. Due to its dependency on all data seen so far, interpretability of the full model is usually difficult. Another example is given by the support vector machine which represents a solution in terms of support vectors which describe the decision boundary as accurately as possible. Due to this focus on the class boundaries, excellent classification accuracy is usually obtained, but interpretability is lost and the size of the solution usually scales with the size of the training set. In consequence, applicants cannot interpret the results in such settings and it is hardly possible to substantiate a machine classification by a semantic explanation.

Prototype-based methods represent solutions in terms of prototypical class representatives and, thus, they offer a direct interface for the user: prototypes can be inspected in the same way as data points, and class typical characteristics can be inferred thereof. There exists a variety of different prototype-based learning schemes in the supervised as well as the unsupervised domain [14]. For supervised classification, the family of learning vector quantization (LVQ) type algorithms constitutes one of the most popular technique to train such a model based on given data. While many basic LVQ techniques rely on heuristics only, a variety of learning rules has been derived from a formal objective function [20, 23]. In this contribution, we focus on robust soft learning vector quantization (RSLVQ) as proposed in [23] since it can be derived from a probabilistic model where underlying assumptions are explicit and can easily be changed depending on the application area. Training is derived as a maximization of the likelihood ratio. RSLVQ resembles the classical LVQ2.1 algorithm in the limit of small bandwidth with the restriction that corrections are performed only if classifcation errors are present. While the limit case as well as standard LVQ2.1 do not achieve optimum behavior already in simple model situations [1], RSLVQ displays excellent generalization ability in the original probabilistic setting [22].

In many application areas, data sets are becoming more and more complex, and dedicated similarity measures are used to compare such data. Examples are bioinformatics sequences, graphs, or tree structures as they occur in linguistics, time series data, functional data arising in mass spectrometry, relational data stored in relational databases, etc. These data are no longer represented as Euclidean vectors, rather, pairwise similarities are available. Several machine learning techniques can deal with such non-vectorial data: technologies include structure kernels, recursive and graph networks, functional methods, relational approaches, and similar [7, 21, 9, 19, 11]. In the last years, several popular prototype-based algorithms have been extended to deal with more general data. Some techniques rely on a matrix of pairwise similarities or dissimilarities only rather than explicit feature vectors. In this setting, median clustering as provided by median self-organizing maps, median neural gas, or affinity propagation characterizes clusters in terms of typical exemplars [8, 15, 6]. More general smooth adaptation is offered by relational extensions such as relational neural gas or relational learning vector quantization [10]. A further possibility is offered by kernelization such as proposed for neural gas, self-organizing maps, or different variants of learning vector quantization [17, 4, 18]. By formalizing the interface to the data as a general similarity or dissimilarity matrix, complex structures can be easily dealt with: structure kernels for graphs, trees, alignment distances, string distances, etc. open the way towards these general data structures [16, 9].

In this contribution, we present an extension of RSLVQ to a general kernel. A statistically well motivated model is obtained which achieves excellent results. For more general similarities, the technique can be combined with standard preprocessing such as clip or flip, as we will demonstrate in experiments. The prototypes are represented implicitly as linear combinations in the feature space, thus interpretability is lost. We investigate the possibility to approximate the model by using the k nearest neighbors of the prototypes, to arrive at an accurate and interpretable model for general similarity data in many cases.

2 Robust soft learning vector quantization

Learning vector quantization (LVQ) offers a class of discriminative prototype based learning schemes which can naturally deal with any number of classes [14]. Unlike SVM, the model complexity is defined by the applicant, and usually only few prototypes are sufficient to represent the given classes. Since classes are represented by typical examples rather than its boundaries, a representation and inspection of the classes in terms of the prototypes is possible. This feature has been used e.g. in the context of life-long learning models, see e.g. [13]. Basic LVQ learning algorithms as proposed by Kohonen include LVQ1 which is directly based on Hebbian learning, and improvements such as LVQ2.1, LVQ3, or OLVQ which aim at a higher convergence speed or better approximation of the Bayesian borders. These types of LVQ schemes are essentially heuristically motivated and a valid cost function does not exist [2]. One of the first proposals of a cost function of LVQ can be found in [20, 12]. An alternative which is based on a probabilistic model of the data has been proposed in [23]. This method, robust soft LVQ (RSLVQ) models data by a mixture of Gaussians. Learning rules are derived thereof by means of a maximization of the log likelihood ratio of the given data. In the limit of small bandwidth, a learning rule which is similar to LVQ2.1 but which performs adaptation in case of misclassification only, is obtained.

Now we formally define the basic RSLVQ scheme. Assume data $\xi_k \in \mathbb{R}^n$ are labeled with discrete labels y_k. A RSLVQ network models data by a mixture distribution characterized by m prototypes $w_j \in \mathbb{R}^n$ with priorly fixed labels $c(w_j)$ and bandwidths σ_j. The mixture component j defines the probability

$$p(\xi|j) = K(j) \cdot \exp(f(\xi, w_j, \sigma_j^2))$$

with normalization constant $K(j)$ and function f

$$f(\xi, w_j, \sigma_j^2) = -\|\xi - w_j\|^2/\sigma_j^2$$

based on the Euclidean distance or a generalization thereof. This induces the probability of data point ξ:

$$p(\xi|W) = \sum_j P(j) \cdot p(\xi|j)$$

with prior $P(j)$ and parameters W of the model. The probability of a data point ξ and label y is

$$p(\xi, y|W) = \sum_{c(w_j)=y} P(j) \cdot p(\xi|j)\,.$$

Learning aims at an optimization of the log likelihood ratio

$$L = \sum_k \log \frac{p(\xi_k, y_k|W)}{p(\xi_k|W)}\,.$$

A stochastic gradient ascent yields the following update rules, given (ξ_k, y_k)

$$\Delta w_j = \alpha \cdot \begin{cases} (P_y(j|\xi_k) - P(j|\xi_k)) \cdot K(j) \cdot \partial f(\xi_k, w_j, \sigma_j^2)/\partial w_j & \text{if } c(w_j) = y_k \\ -P(j|\xi_k) \cdot K(j) \cdot \partial f(\xi_k, w_j, \sigma_j^2)/\partial w_j & \text{if } c(w_j) \neq y_k \end{cases}$$

with learning rate $\alpha > 0$ and the probabilities

$$P_y(j|\xi_k) = \frac{P(j)\exp(f(\xi_k, w_j, \sigma_j^2))}{\sum_{c(w_j)=y_j} P(j)\exp(f(\xi_k, w_j, \sigma_j^2))}$$

and

$$P(j|\xi_k) = \frac{P(j)\exp(f(\xi_k, w_j, \sigma_j^2))}{\sum_j P(j)\exp(f(\xi_k, w_j, \sigma_j^2))}\,.$$

If the standard Euclidean distance is used, class priors are equal, and small bandwidth is present, a learning rule similar to LVQ2.1, learning from mistakes, results.

Given a novel data point ξ, its class label is the most likely label y corresponding to a maximum value $p(y|\xi, W) \sim p(\xi, y|W)$. For typical settings, bandwidths are chosen equally $\sigma_j^2 = \sigma^2$, and the same holds for the priors $P(j) = \text{const}$. Further, the simple Euclidean distance is used. Then, this rule can be approximated by a simple winner takes all rule, i.e. ξ is mapped to the label $c(w_j)$ of the closest prototype w_j. It has been shown in [23] that RSLVQ yields excellent classification results while preserving interpretability of the model due to prototypical representatives of the classes given by w_j.

3 Kernel robust soft learning vector quantization

RSLVQ is restricted to Euclidean vectors. A kernelization of the method makes the technique applicable for more general data sets which are characterized in terms of a Gram matrix only. We assume that a kernel k is fixed corresponding to a feature map Φ. Then, it holds

$$k_{kl} := k(\xi_k, \xi_l) = \Phi(\xi_k)^t \Phi(\xi_l)$$

for all data points ξ_k, ξ_l. We assume that prototypes are represented by linear combinations of data

$$w_j = \sum_m \gamma_{jm} \Phi(\xi_m)\,.$$

Note that this assumption is not necessarily fulfilled for vectorial RSLVQ due to the error correction, i.e. anti-Hebbian terms. However, it is reasonable to assume that prototypes are contained in the convex hull of the data if interpretability should be guaranteed. In this case coefficients γ_{jm} are non-negative and sum up to 1. The cost function of RSLVQ becomes

$$L = \sum_k \log \frac{\sum_{c(w_j)=y_k} P(j)p(\Phi(\xi_k)|j)}{\sum_j P(j)p(\Phi(\xi_k)|j)} \,.$$

We assume equal bandwidth $\sigma^2 = \sigma_j^2$, constant prior $P(j)$ and mixture components induced by normalized Gaussians. These can be computed in the data space based on the Gram matrix because of the identity

$$\|\Phi(\xi_i) - w_j\|^2 = \|\Phi(\xi_i) - \sum_m \gamma_{jm}\Phi(\xi_m)\|^2 = k_{ii} - 2\cdot\sum_m \gamma_{jm}k_{im} + \sum_{s,t}\gamma_{js}\gamma_{jt}k_{st}$$

where the distance in the feature space is referred to by $\|\cdot\|^2$. Thus the update rules become $\Delta w_j = \sum_m \Delta\gamma_{jm}\Phi(\xi_m) =$

$$\alpha\cdot K(j)\cdot\begin{cases} (P_y(j|\Phi(\xi_k)) - P(j|\Phi(\xi_k)))\,(\Phi(\xi_k) - \sum_m \gamma_{jm}\Phi(\xi_m)) & \text{if } c(w_j) = y_k \\ -P(j|\Phi(\xi_k))\,(\Phi(\xi_k) - \sum_m \gamma_{jm}\Phi(\xi_m)) & \text{if } c(w_j) \neq y_k \end{cases}$$

A stochastic gradient ascent yields the following adaptation rules for γ_{jm}:

$$\Delta\gamma_{jm} = \alpha\cdot K(j)\cdot\begin{cases} -(P_y(j|\Phi(\xi_k)) - P(j|\Phi(\xi_k)))\gamma_{jm} & \text{if } \xi_m \neq \xi_k, c(w_j) = y_k \\ (P_y(j|\Phi(\xi_k)) - P(j|\Phi(\xi_k)))(1-\gamma_{jm}) & \text{if } \xi_m = \xi_k, c(w_j) = y_k \\ P(j|\Phi(\xi_k))\gamma_{jm} & \text{if } \xi_m \neq \xi_k, c(w_j) \neq y_k \\ -P(j|\Phi(\xi_k))(1-\gamma_{jm}) & \text{if } \xi_m = \xi_k, c(w_j) \neq y_k \end{cases}$$

This adaptation performs exactly the same updates as RSLVQ in the feature space if prototypes are in the convex hull of the data. To guarantee non-negativity and normalization, a correction takes place after every adaptation step.

4 *k*-approximation of the prototypes

Kernel RSLVQ yields prototypes which are implicitly represented as linear combinations of data points

$$w_j = \sum_m \gamma_{jm}\Phi(\xi_m)\,.$$

Since the training algorithm and classification depends on pairwise distances only, simple linear algebra allows us to compute the distance of a data point and a prototype based on the pairwise similarity of the data point and all training data only, i.e. the given Gram matrix, as specified above. However, direct interpretability of the prototype is lost this way.

Here we propose to use a simple approximation of the prototypes to maintain interpretability and flexibility of the clustering. As already proposed in the context of life long learning for relational approaches, a prototype is approximated by the k nearest exemplars in the given training set [10]. These exemplars can easily be determined by a linear scan of the training set.

5 Experiments

We compare RSLVQ and its k-approximation with different values of k on a variety of benchmarks as introduced in [5]. The data sets consist of similarity matrices which are, in general, non-euclidean. Non-euclideanity can be quantified by the signature of the data set, i.e. the number of positive and negative eigenvalues of the Gram matrix. The matrices are symmetrized and normalized before processing. In general, data do not constitute a valid kernel such that a probabilistic representation using the above formulas is no longer well-defined due to potentially negative distances. There exist standard preprocessing tools which transfer a given similarity matrix into a valid kernel, as presented e.g. in [5, 16]. Typical corrections are:

– *Spectrum clip:* set negative eigenvalues of the matrix to 0. Since this can be realized as a linear projection, it directly transfers to out-of-sample extensions.
– *Spectrum flip:* negative eigenvalues are substituted by their positive values. Again, this can be realized by means of a linear transformation.

These transforms are tested for kernel RSLVQ with according preprocessing. We use training data in analogy to [5].

– *Amazon47*: This data set consists of 204 books written by 47 different authors. The similarity is determined as the percentage of customers who purchase book j after looking at book i. The signature is $(192, 12, 0)$. The class label of a book is given by the author.
– *Aural Sonar*: This data set consists of 100 wide band solar signals corresponding to two classes, observations of interest versus clutter. Similarities are determined based on human perception, averaging over two random probands for each signal pair. The signature is $(62, 38, 0)$. Class labeling is given by the two classes: target of interest versus clutter.
– *Face Rec*: 945 images of faces of 139 different persons are recorded. Images are compared using the cosine-distance of integral invariant signatures based on surface curves of the 3D faces. The signature is $(794, 151, 0)$. The labeling corresponds to the 139 different persons.
– *Patrol*: 241 samples representing persons in eigth different patrol units are contained in this data set. Similarities are based on responses of persons in the units about other members of their groups. The signature is $(117, 124, 0)$. Class labeling corresponds to the eigth patrol units.
– *Protein*: 213 proteins are compared based on evolutionary distances comprising four different classes according to different globin families. The signature is $(171, 42, 0)$. Labeling is given by four classes corresponding to different globin families.
– *Voting*: Voting contains 435 samples with categorical data compared by means of the value difference metric. The signature is $(226, 209, 0)$. Class labeling into two classes is present.

	kernel RSLVQ	1-approx	2-approx	3-approx	4-approx
Amazon47	15.37 (0.36)	36.83 (0.35)	29.27 (0.42)	29.65 (0.53)	30.97 (0.44)
clip	15.37 (0.41)	31.65 (0.26)	26.29 (0.48)	28.06 (0.43)	29.96 (0.42)
flip	16.34 (0.42)	31.28 (0.25)	28.91 (0.35)	29.85 (0.39)	30.95 (0.40)
Aural Sonar	11.50 (0.37)	25.13 (1.15)	20.62 (1.56)	21.62 (0.96)	21.62 (0.79)
clip	11.25 (0.39)	24.75 (0.78)	21.75 (1.06)	18.50 (0.66)	17.00 (0.83)
flip	11.75 (0.35)	24.75 (0.99)	21.50 (0.48)	21.00 (0.57)	21.62 (0.53)
Face Rec	3.78 (0.02)	3.70 (0.02)	5.92 (0.02)	8.99 (0.03)	11.70 (0.04)
clip	3.84 (0.02)	3.76 (0.02)	6.00 (0.02)	8.95 (0.03)	11.90 (0.04)
flip	3.60 (0.02)	3.33 (0.02)	5.64 (0.03)	8.58 (0.04)	11.57 (0.03)
Patrol	17.50 (0.25)	54.94 (0.96)	46.69 (1.02)	39.33 (0.77)	35.46 (0.49)
clip	17.40 (0.29)	32.46 (0.90)	21.89 (0.34)	22.36 (0.43)	20.28 (0.31)
flip	19.48 (0.34)	37.42 (0.85)	26.36 (0.45)	22.27 (0.25)	21.96 (0.27)
Protein	26.98 (0.37)	55.12 (0.67)	49.97 (0.77)	49.57 (0.75)	47.38 (0.92)
clip	4.88 (0.17)	22.44 (0.51)	25.81 (0.98)	28.20 (0.92)	29.42 (0.89)
flip	1.40 (0.05)	23.26 (0.26)	22.77 (0.34)	22.56 (0.47)	23.37 (0.52)
Voting	5.46 (0.04)	8.56 (0.06)	8.71 (0.07)	8.59 (0.07)	8.56 (0.05)
clip	5.34 (0.04)	8.65 (0.07)	9.22 (0.09)	9.08 (0.09)	8.82 (0.09)
flip	5.34 (0.03)	7.84 (0.04)	7.82 (0.03)	8.13 (0.03)	8.56 (0.04)

Table 1: Results of kernel RSLVQ and its k-approximation for $k \in \{1, \ldots, 4\}$. The classification error in % and standard deviation in parenthesis are given.

Prototypes are initialized by means of normalized random coefficients γ_{jm} where the prior class label $c(w_j)$ determines the non-zero elements. The number of prototypes is taken as a small multiple of the number of classes (Amazon47: 94, Aural Sonar: 10, Face Rec: 139, Patrol: 24, Protein: 20, Voting: 20). Other meta-parameters are optimized on the data sets using cross-validation. The results reported in Tab. 1 stem from a 20-fold cross-validation.

The results obtained with kernel RSLVQ are generally good and reach state-of-the art accuracy as reported in [5]. In general, preprocessing using spectrum clip or flip can be beneficial. Surprisingly, a naive application of kernel RSLVQ for the (non-euclidean) similarity matrix already yields surprisingly good results. The results of a k-approximation are heterogeneous. For some of the data sets, the k-approximation is generally acceptable and yields results comparable to kernel RSLVQ itself (Voting). For others, the 1-approximation yields worse results with a decrease of the accuracy by more than 10%, but a sufficient number k yields acceptable results (Aural Sonar, Patrol). Interestingly, there is also the opposite case, a 1-approximation yielding acceptable results, but larger values k leading to more than 5% loss of accuracy (FaceRec). For Amazon47 and Protein, the classification results of all approximations are significantly worse as compared to direct kernel RSLVQ.

To further inspect this behavior, we exemplarily plot the results obtained by kernel RSLVQ and its 1-approximation for the data sets Aural Sonar and Voting in Fig. 1. For visualization, we use multidimensional scaling, which represents measurements of similarity or dissimilarity among pairs of objects as distances between points of low-dimensional multidimensional space [3]. In all cases, the training data, the prototypes obtained by kernel RSLVQ, and its 1-approximation are shown. It is obvious that the obtained prototypes constitute good representatives of the classes. For Aural Sonar, this also holds for the 1-approximation, while it indicates for Voting that prototypes are partially approximated by exemplars stemming from the wrong class. This bevahior can potentially be attributed to initialization issues or a wrong choice of the bandwidth, since kernel RSLVQ corresponds to a learning from mistakes rule for small bandwidth, hence prototypes are not necessarily driven towards the class centers if the classification is correct.

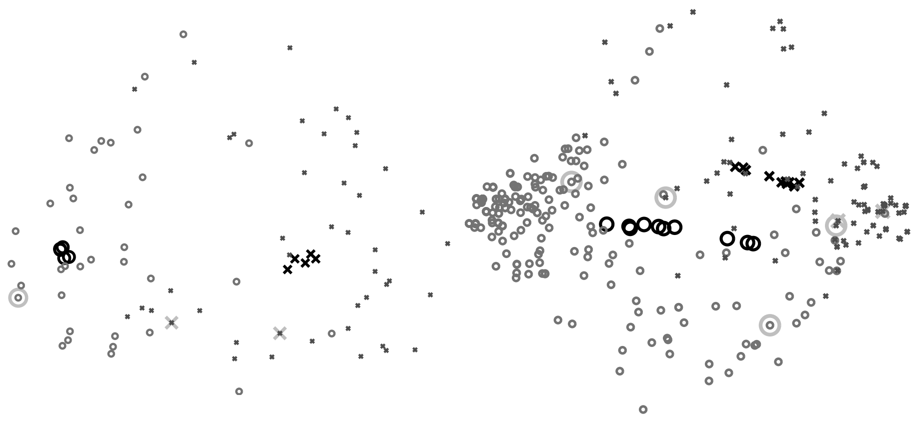


Fig. 1: Visualizing the Aural Sonar flip (left) and Voting clip (right) sets together with prototypes (black) and 1-approximation (pale) using multidimensional scaling.

6 Discussion

We have investigated kernel robust soft LVQ and the possibility to obtain interpretable sparse models by means of a k-approximation. While kernel RSLVQ generally yields very good results comparable to SVM, the situation is less clear for the k-approximations. In some cases, a high classification accuracy is maintained, while the classification accuracy is decreased by more than 15% for others. So far, approximation is based on the distance of the exemplar to the prototype only, neglecting label information or classification accuracy on the training set. It is the subject of ongoing work to incorporate this information in the selection process.

Acknowledgement

This work has been supported by the DFG under grant number HA2719/7-1 and by the CITEC center of excellence.

References

1. M. Biehl, A. Ghosh, and B. Hammer. Dynamics and generalization ability of LVQ algorithms. *Journal of Machine Learning Research*, 8:323–360, 2007.
2. M. Biehl, B. Hammer, M. Verleysen, and T. Villmann, editors. *Similarity Based Clustering.* Springer Lecture Notes Artificial Intelligence Vol. 5400/2009, Springer, 2009.
3. I. Borg and P. J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications.* Springer, 2nd edition, 2005.
4. R. Boulet, B. Jouve, F. Rossi, and N. Villa. Batch kernel SOM and related Laplacian methods for social network analysis. *Neurocomputing*, 71(7-9): 1257-1273, 2008.
5. Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti. Similarity-based classification: Concepts and algorithms. *JMLR*, 10:747–776, June 2009.
6. M. Cottrell, B. Hammer, A. Hasenfuss, and T. Villmann. Batch and median neural gas. *Neural Networks*, 19:762–771, 2006.
7. P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE TNN*, 9(5):768–786, 1998.
8. B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.
9. T. Gärtner. *Kernels for Structured Data.* PhD thesis, Univ. Bonn, 2005.
10. B. Hammer and A. Hasenfuss. Topographic mapping of large dissimilarity datasets. *Neural Computation*, 22(9):2229–2284, 2010.
11. B. Hammer, A. Micheli, and A. Sperduti. Universal approximation capability of cascade correlation for structures. *Neural Computation*, 17:1109–1159, 2005.
12. B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002.
13. S. Kirstein, H. Wersing, H.-M. Gross, and E. Körner. A life-long learning vector quantization approach for interactive learning of multiple categories. *Neural Networks*, 28:90–105, 2012.
14. T. Kohonen. *Self-Oganizing Maps.* Springer, 3rd edition, 2000.
15. T. Kohonen and P. Somervuo. How to make large self-organizing maps for nonvectorial data. *Neural Networks*, 15(8-9): 945-952. 2002.
16. E. Pekalska and R. P. Duin. *The Dissimilarity Representation for Pattern Recognition: Foundations and Applications.* World Scientific, 2005.
17. A. K. Qin and P. N. Suganthan. Kernel neural gas algorithms with application to cluster analysis. In *Proceedings of the 17th International Conference on Pattern Recognition Volume 4* (ICPR '04), pages 617–620, Washington, DC, USA, 2004.
18. A. K. Qin and P. N. Suganthan. A novel kernel prototype-based learning algorithm. In *Proceedings of the 17th International Conference on Pattern Recognition Volume 4* (ICPR '04), pages 621–624, Cambridge, UK 2004.
19. F. Rossi and N. Villa-Vialaneix. Consistency of functional learning methods based on derivatives. *Pattern Recognition Letters*, 32(8):1197–1209, 2011.
20. A. Sato and K. Yamada. Generalized Learning Vector Quantization. In *NIPS*, 1995.

21. F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. Computational capabilities of graph neural networks. *IEEE TNN*, 20(1):81–102, 2009.
22. P. Schneider, M. Biehl, and B. Hammer. Distance learning in discriminative vector quantization. *Neural Computation*, 21:2942–2969, 2009.
23. S. Seo and K. Obermayer. Soft learning vector quantization. *Neural Computation*, 15:1589-1604, 2003.
24. A. Vellido, J.D. Martin-Guerroro, and P. Lisboa. Making machine learning models interpretable. In *ESANN'12*, 2012.

No Perplexity in Stochastic Neighbor Embedding

Marc Strickert

Department of Mathematics and Computer Science, SYNMIKRO,
Philipps Universität Marburg, Germany
marc.strickert@uni-marburg.de

Abstract. Excellent results of stochastic neighbor embedding (SNE) methods are strongly dependent on the definition of data neighborhood probabilities. A rigorous definition is proposed that circumvents typical trial and error search or heuristics for the so-called perplexity value parametrizing data-specific neighborhood ranges. The new definition is parameter-free and yields excellent embeddings, and it indicates that the assumption of symmetric neighborhood probabilities in the original t-distributed SNE formulation is problematic, especially when utilizing rigorous optimization schemes.

1 Background of stochastic neighbor embedding

Stochastic neighbor embedding is an increasingly popular visualization technique for embedding very general pairwise data relationships in the Euclidean space [3, 6]. The flexibility of treating metric relationships, dissimilarities and pairwise scoring data in the same manner can be achieved by using only the probabilities of data items being neighbored, more precisely, of the pairwise neighborhood degree. Thereby, neighborhood scores can be provided as very general measurements, such as Minkowski distances, but also normalized compression distance or greedy string tiling [5], such scores allowing for intuitive visual exploration of document topics and transcriptome data [2], for example.

Formally, SNE optimization aims at placing points $\mathbf{y}_i$ in a Euclidean space such that the Kullback-Leibler divergence KL between the original neighborhood probabilities $\mathbf{P}$ of n given objects and those of their corresponding embedded points $\mathbf{Q}(\mathbf{Y})$ are minimized, i.e.

$$\mathbf{Y} = \operatorname{argmin}_{\mathbf{Y}'} \sum_{i=1}^{n} \mathsf{KL}(\mathbf{P}_i, \mathbf{Q}_i(\mathbf{Y}'))$$

which can be conveniently accomplished by gradient-based methods. Recent work shows how other divergence measures can be used in application-specific scenarios [1]. After all, the neighborhood probability distributions of points in the embedding space should reflect best the distributions of their original neighborhoods. Alternatively, the non–point-specific distribution of all neighborhood probabilities can be reconstructed, i.e. the reconstruction is not specific to rows $\mathbf{P}_i$ of $\mathbf{P}$; this is misleadingly called symmetric SNE in [6], because the authors in addition claim symmetry $\mathbf{P}_{\text{sym}} \propto \mathbf{P} + \mathbf{P}^{\mathsf{T}}$ for improved convergence.

2 Neighborhood probability

SNE minimization of KL depends both on adjustable $\mathbf{Q}(\mathbf{Y})$, i.e. on the embedded points $\mathbf{Y}$, but also on the original neighborhood distribution $\mathbf{P}$. The original density estimates are based on Gaussians centered over each high-dimensional data point with a common variance value determining the entropy, that is the rather counter-intuitive *log-perplexity*, of the neighborhood probability distribution. Implicitly this approach assumes input and output spaces with spatially constant data densities. Gaussian widths can be manually set or fit for a given perplexity by information-theoretically motivated interval bisection optimization [6]. For ease of optimization and for faster convergence, perplexity-based neighborhood probability matrices are rendered symmetric, as indicated above. Such an assumption is in contrast to commonly asymmetric Euclidean neighborhood probability relationships such as required for describing even simple systems like three points $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ randomly located in the plane where for $\boldsymbol{b}$, the nearest neighbor of $\boldsymbol{a}$, $\boldsymbol{c}$ might be nearest neighbor.

An observable incompatibility of concentrated distances encountered in high-dimensional Euclidean input spaces and widely distributed distances in low-dimensional output spaces is addressed by modeling the embedding relationships by Student t-distribution instead of a Gaussian. This approach is referred to as t-SNE and was found to be more successful in neighborhood preservation of high-dimensional data [6].

For using perplexity-based neighborhood probability estimations with SNE methods, score data and dissimilarities are usually turned into pseudo distances with minimum of zero by using some 'standard', yet distorting, transformation. Alternatively, in the following, pairwise relationships of item j are assumed to be located in rows of the data score matrix $\mathbf{S}$, the degree of neighborhood being related to the rank of scores in that row, thereby ignoring self-pairings (j, j). Using all data pairs, the largest values constitute the score distribution of nearest neighbors, the smallest values characterize the farthest neighbors. The probability of a specific score to exceed other scores can be approximated by the mid-rank, compensating for ties, of each score s_{ij} within all scores in $\mathbf{S}$. This is summarized in Figure 1 together with an exact neighbor probability estimation approach. Both procedures allow for asymmetric source probability matrices $\mathbf{P}^*$.

```
Precise, but slow:
1a:  $\mathbf{P}^* \leftarrow \mathbf{S} - \mathbf{S}$ #  empty matrix
2a:  For all unique scores $s_{ij}$ in $\mathbf{S}$
3a:    $\boldsymbol{\Omega} \leftarrow \{ (k,l) \mid s_{kl} \geq s_{ij} \in \mathbf{S} \}$
4a:    $\mathbf{P}^*_{\Omega} \leftarrow \mathbf{P}^*_{\Omega} + | \{ s_{kl} = s_{ij} \} |$
5a:  End
```

```
Approximation, fast:
1b:  $\mathbf{R} \leftarrow$ Mid-rank column-major order vectorized $\mathbf{S}$
2b:  $\mathbf{P}^* \leftarrow$ Column major order (n x n)-constructed $\mathbf{R}$
End:  Scale rows of $\mathbf{P}^*$ to unity sum.
```

Fig. 1. Neighborhood probability estimation procedure based on the marginal probability of pairwise score exceedance (left) and tie-averaging approximation (right).

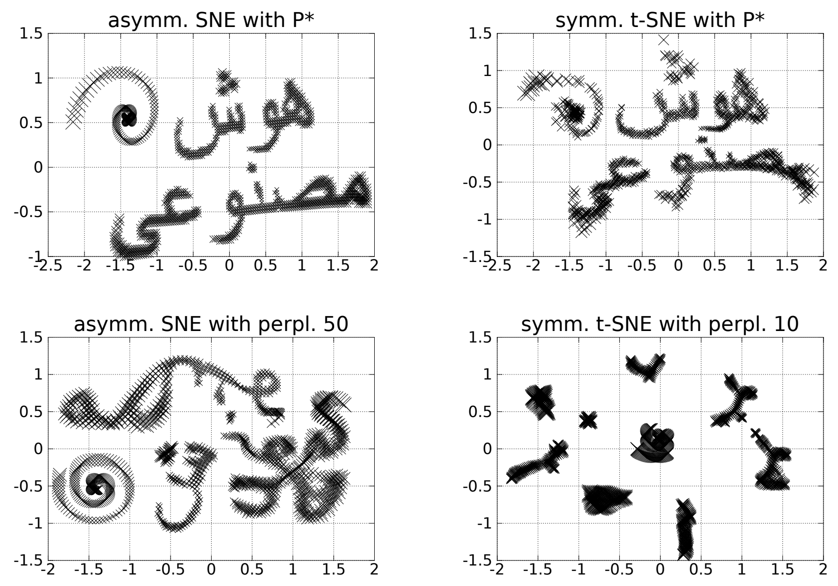


Fig. 2. Alternative stochastic neighbor embeddings of the Persian words 'artificial intelligence' being combined with a logarithmic spiral. The top left panel provides an almost perfect reconstruction of the 2D source data by asymmetric SNE of the proposed neighborhood probability inference. Slight tilt is caused by PCA normalization being applied to all four embeddings. Asymmetric t-SNE would yield very good results too, but trying to embed the asymmetric relationships P^* with a symmetric t-SNE implementation results in a compromise solution with scatter artifacts for this unresolvable configuration, shown in the top right panel. The two panels on the bottom show typical SNE (left) and t-SNE (right) results for symmetric perplexity-based neighborhood probability estimators. Point sizes reflect the total absolute forces, attracting and repelling, acting on each point according to the contributions to KL; larger points indicate higher tension and highlight potentially problematic embeddings.

3 Embedding results

Persian words for 'artificial intelligence' were rasterized in the plane and combined with a logarithmic spiral. This data set of 1164 points exhibits cluster structures, distance ties caused by the raster grid, and asymmetric properties in the spiral with increasing gaps between points. In the experiments, this set of 2D points is either directly fed into perplexity-based SNE methods or the corresponding distance matrix is turned into 'pseudo-scores' S by just switching signs before calculating P^*. Resulting embeddings are shown in Figure 2. The size of the crosses is proportional to the magnitude of forces, i.e. the sum of squared derivatives of the KL divergence, acting on the corresponding points. Note that the sum of repelling and attracting forces are zero, resulting in absence of point movement and leading to convergence after all.

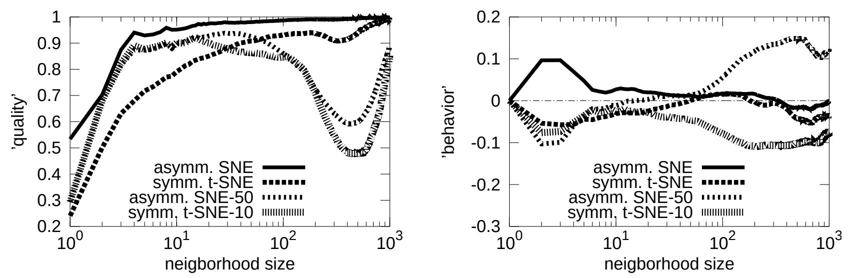


Fig. 3. Embedding quality assessment according to [4]. Left: quality of neighborhood preservation over neighborhood size with one being optimum. Right: behaviour of embedding; positive values indicate undesired mapping to neighbored points (intrusion), negative values indicate missed mappings of truly neighbored points (extrusion).

The embedding by asymmetric SNE with $\mathbf{P}^*$ in the top left panel is visually identical to the original set of points (not shown). Also the symmetric t-SNE (top right panel) comes quite close, but it can be seen that boundary points like those of the logarithmic spiral being exposed to asymmetric neighborhoods fail to be recovered validly under symmetric modeling assumptions. Asymmetric SNE with a perplexity of 50 (bottom left) takes into account a rather high number of effective neighbors and yields rather smooth reconstructions. As a result, connected shapes stay together, but the overall organization is locally twisted, and the logarithmic spiral does not exhibit its exponential radius growth. The t-SNE plot based on a perplexity of 10 (bottom right panel) exhibits the typical starfield-like structure that can be often observed for t-SNE embeddings. Natural 'clusters' get well-separated, but some connected shapes are torn apart, and the overall connectivity is lost. As an often observed result, local outlier structures like the spiral get placed in the center of the embedding to become 'equally unlikely' neighbors to other points. No perplexity value could be found, though, leading to embeddings similar to the ones based on $\mathbf{P}^*$.

The quality of the plots is assessed by neighborhood rank violations between the original points and those in the embeddings [4]. For each neighborhood size, these violations are measured similar to the Spearman rank correlation between source and target neighborhoods around each point, shown in the left panel of Figure 3. Asymmetric SNE on the proposed $\mathbf{P}^*$ yields almost perfect reconstructions along all neighborhood ranges. Its symmetric t-SNE counterpart improves over the two perplexity-based versions for ranges greater than about 60 neighbors. For these two, a dramatic quality loss is indicated at a range of 400–500 neighbors. The type of embedding failure is displayed in the right 'behaviour' panel which is in visual correspondence to the embeddings in Figure 2. Each method has certain problems for ranges up to 4–6 neighbors; these are caused by the nature of tied data neighborhood distances that are unlikely to become reconstructed as exact ties in the embedding. Perplexity-based SNE creates false neighborhood relations for neighborhood ranges greater than 100, for which perplexity-based t-SNE misses reconstruction of neighbors. Less of such intrusive and extrusive behaviour is shown by using $\mathbf{P}^*$

4 Discussion

The proper estimation of neighborhood probability distributions has been neglected since the invention of SNE 10 years ago. Allowing for an incompatibility between distributions in the input and output space like in t-SNE, the present approach completely decouples estimation procedures in both spaces, now aiming at the best description P^* of the input space, as described by most general pairwise score assumptions, and using a reasonable estimation for Euclidean embedding spaces. Experiments with a controlled data set show the benefits of such an alternative distribution estimation. Due to the structural difference of P^* and perplexity-based estimates, final KL divergence values might be lower for perplexity-based neighborhood estimation although the results for P^* look more realistic. Optimizing embeddings for $\mathsf{Q}^*(\mathsf{Y})$ calculated like P^* would be the best, yet, computationally much more demanding approach.

MATLAB/GNU-Octave estimators of P^* and (l-)BFGS gradient-based optimizers for (t-)SNE are available at http://mloss.org as package 'xSNE'.

References

1. Kerstin Bunte, Sven Haase, Michael Biehl, and Thomas Villmann. Stochastic neighbor embedding (sne) for dimension reduction and visualization using arbitrary divergences. *Neurocomputing*, 90:23–45, 2012.
2. Natascha Bushati, James Smith, James Briscoe, and Christopher Watkins. An intuitive graphical visualization technique for the interrogation of transcriptome data. *Nucleic Acids Research*, 39(17):7380–7389, 2011.
3. Geoffrey Hinton and Sam T. Roweis. Stochastic neighbor embedding. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Neural Information Processing Systems 15 (NIPS)*, volume 15, pages 857–864. MIT Press, 2002.
4. J.A. Lee and M. Verleysen. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing*, 72:1431–1443, 2009.
5. Bassam Mokbel, Sebastian Gross, Markus Lux, Niels Pinkwart, and Barbara Hammer. How to quantitatively compare data dissimilarities for unsupervised machine learning? In *Proceedings of the 5th International Workshop on Artificial Neural Networks in Pattern Recognition (ANNPR)*, to appear, 2012.
6. Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

How to visualize a classifier?

Alexander Schulz, Andrej Gisbrecht, Kerstin Bunte, and Barbara Hammer

University of Bielefeld - CITEC centre of excellence, Germany
{aschulz|agisbrec|kbunte|bhammer}@techfak.uni-bielefeld.de

Abstract. We propose a general framework to nonlinearly visualize a given classifier and training set. We assume a classifier, which not only provides crisp classification results, but also smooth values indicating the distance from the nearest class boundary. Furthermore, we discuss the requirements which this framework poses on a nonlinear dimensionality reduction method used for the visualization.

1 Introduction

More and more complex data sets and learning scenarios pose new challenges for standard data analysis tools: often, an exact objective is not clear a priori; rather, the users specify their interests and demands interactively when applying data mining techniques and inspecting the results [27]. Due to this fact, analysis tools have to offer an intuitive interface for the users such that they can directly interpet or refine the results on demand. Hence, apart from a good accuracy, interpretability of machine learning techniques becomes a central issue [24, 20]. Naturally, the notion of what means 'interpretable' is not clear a priori and it severely depends on the given data set and problem setting. However, visualization often plays an essential part in this context with the human visual system being one of our most advanced senses.

Classification constitutes one of the standard tasks in data analysis. At present, the major way to display the result of a classifier and to judge its suitability is by means of the classification accuracy. Visualization is used in only a few places when inspecting a classifier: If the data lives in a low dimensional space, such that direct visualization of the data points and classification boundaries in 2D or 3D is possible. For high D data, which constitutes the standard case, a direct visualization of the classifier is not possible. One line of research addresses visualization techniques to accompany the accuracy by an intuitive interface to set certain parameters of the classification procedure, such as e.g. ROC curves to set the desired specificity, or more general interfaces to optimize parameters connected to the accuracy [11]. Surprisingly, there exists relatively little work to visualize the underlying classifier itself for high dimensional settings. For the popular support vector machine (SVM), for examples, only some specific approaches have been proposed: one possibility is to let the users decide an appropriate linear projection dimension by means of tour methods [5]. As an alternative, some techniques rely on the distance of the data points to the class boundary and present this information using e.g. nomograms [12] or by using linear projection techniques on top of this distance [19]. A few nonlinear techniques exist such as SVMV [26], which visualizes the given data by means of a self-organizing map and displays the class boundaries by means of sampling. These techniques offer first steps to visually inspect an SVM solution such that the users can judge e.g. remaining error regions, the modes of the given classes, outliers, or the smoothness of the separation boundary based on a visual impression.

However, so far, these techniques are often only linear, they require additional parameters, and they provide combinations of a very specific classifier such as SVM and a specific visualization technique. In this contribution we discuss a general framework which allows to visualize the result of a given classifier and its training set in general, using a suitable nonlinear dimensionality reduction technique. We specify demands which such a nonlinear dimensionality reduction technique should fulfill to be eligible. Further, we demonstrate the method in an example.

2 The general framework

We assume the following scenario: a data set including points $\mathbf{x}_i \in X = \mathbb{R}^n$ is given. Every data point is labeled with $l_i \in L$ belonging to a finite set of different labels L. In addition, a classifier $f : X \to L$ has been trained on the given training set, such as a support vector machine or a learning vector quantization (LVQ) network. The standard way to evaluate the performance of the classifier f is by inspecting the classification error of the function on the given training set or a hold out test set. This gives us an indication whether the classifier is nearly perfect, corresponding to 100% accuracy, or whether errors occur. However, the classification error does not give us a hint about the geometric distribution of the errors (are they equally distributed in the space, or do they accumulate on specific misclassified regions), whether errors are unavoidable (due to overlapping regions of the data or outliers), whether the class boundaries are complex (e.g. due to multiple modes in the single classes), etc. A visualization of the given data set and the classifier would offer the possibility to visually inspect the classification result and to answer such questions. We propose a general framework how to visualize a classifier and a given data set such as the training set of the classifier.

In recent years, many different nonlinear dimensionality reduction techniques have been proposed to project a given data set onto low dimensions (usually 2D or 3D), see e.g. [3, 14, 23]. These techniques substitute the points $\mathbf{x}_i \in X$ by low-dimensional counterparts $p(\mathbf{x}_i) = \mathbf{y}_i \in Y = \mathbb{R}^2$, such that the structure of the original data points $\mathbf{x}_i$ is preserved by the low dimensional projections $p(\mathbf{x}_i) = \mathbf{y}_i$ as much as possible. It is in-general ill-posed what 'structure-preservation' means in this context, and, consequently, the techniques differ in the formalization of this demand as a mathematical objective. Several popular approaches aim at a preservation of distances, similarities, pairwise probabilities, or similar. Some methods aim at a preservation of the underlying manifold such as [22, 17, 2].

First approaches how to formally evaluate the success of such a technique have been proposed in the last years, one generally accepted way e.g. measuring the preservation of local k-ary neighborhoods while projecting the data [15].

These techniques, however, map a given finite set of data points only. They do neither represent the structure of the data points as concerns a given classifier nor their relation to the classification boundary. Which possibilities exist to extend a given nonlinear dimensionality reduction method such that an underlying classifier is displayed as well?

We assume a classifier f is present. In addition, we assume that the label $f(\mathbf{x})$ is accompanied by a nonnegative real value $r(\mathbf{x}) \in \mathbb{R}$ which indicates the distance from the closest class boundary. Assuming a nonlinear dimensionality reduction method is given, a naive approach could be like follows:

- Sample the full data space X by points $\mathbf{z}_i$.
- Project these points nonlinearly to two dimensional points $p(\mathbf{z}_i)$ using some nonlinear dimensionality reduction technique.
- Display the data points $\mathbf{x}_i$ and the contours induced by the sampled function $(p(\mathbf{z}_i), r(\mathbf{z}_i))$, the latter approximating the boundaries of the classifier.

This simple method, however, fails unless X is low dimensional because of two reasons:

- Sampling X sufficiently requires an exponential number of points, hence it is infeasible for high dimensional X.
- It is impossible to map a full high dimensional data set $\mathbf{z}_i$ faithfully to low dimensions, hence topological distortions are unavoidable when projecting the class boundaries.

The problem lies in the fact that this procedure tries to visualize the class boundaries in the full data space X. It would be sufficient to visualize only those parts of the boundaries which are relevant for the given training data $\mathbf{x}_i$, the latter usually lying on a low-dimensional sub-manifold of the data space X.

How can this sub-manifold be sampled? We propose the following three steps, which are displayed in Fig. 1:

- Project the data $\mathbf{x}_i$ using a nonlinear discriminative visualization technique leading to points $p(\mathbf{x}_i) \in Y = \mathbb{R}^2$.
- Sample the projection space Y leading to points $\mathbf{z}'_i$. Determine points $\mathbf{z}_i$ in the data space X which are projected to these points $p(\mathbf{z}_i) \approx \mathbf{z}'_i$.
- Visualize the training points $\mathbf{x}_i$ together with the contours induced by the sampled function $(\mathbf{z}'_i, r(\mathbf{z}_i))$.

Unlike the naive approach, sampling takes place in $\mathbb{R}^2$ only and, thus, it is feasible. Further, only those parts of the space X are considered which correspond to the observed data manifold $\mathbf{x}_i$, i.e. the class boundaries are displayed only as concerns these training data.

Two questions remain in this context: what does it mean to consider a *discriminative* nonlinear dimensionality reduction technique? How can we determine inverse points $\mathbf{z}_i$ for given projections $\mathbf{z}'_i$ which correspond to inverse images in the data manifold?

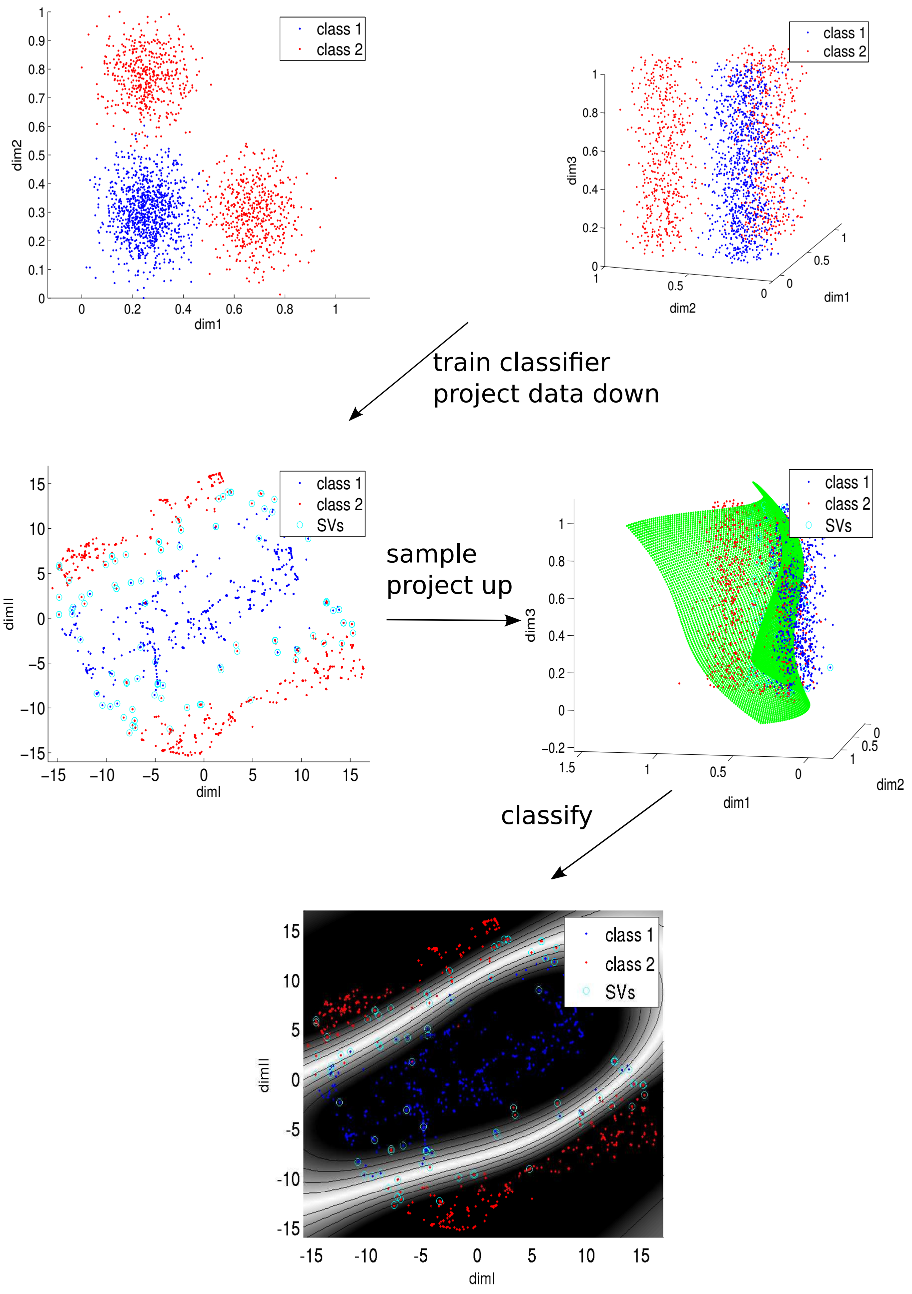


Fig. 1: Principled procedure how to visualize a given data set and a trained classifier. The example displays a SVM trained in 3D.

3 Discriminative nonlinear visualization

Dimensionality reduction is an inherently ill-posed problem, and the result of a dimensionality reduction tool largely varies depending on the chosen technology, the parameters, and partially even random aspects for non-deterministic algorithms. Often, the reliability and suitability of the obtained visualization for the task at hand is not clear at all since a dimensionality reduction tool might focus on irrelevant aspects or noise in the data. Discriminative dimensionality reduction, i.e. the integration of auxiliary information by an explicit labeling of data can help to partially overcome these problems: in discriminative dimensionality reduction, the aim is to visualize those aspects of the data which are particularly relevant for the given class information. Thus, the information which is neglected by the dimensionality reduction method is no longer arbitrary but directly linked to its relevance for the given classes.

In the given setting, an explicit labeling is available by the given classifier. To visualize the behavior of the classifier, it is relevant to visualize those aspects of the data which contribute to the class labeling. Since this captures the information which is relevant in this case, we propose to use a nonlinear discriminative visualization technique which can capture the underlying data manifold as concerns the class labeling.

A variety of different discriminative dimensionality reduction techniques has been proposed, such as the linear techniques Fisher's linear discriminant analysis (LDA), partial least squares regression (PLS), informed projections [7], or global linear transformations of the metric to include auxiliary information [10, 4], kernelization of such approaches [16, 1], or a combination of standard techniques with an adaptation of the underlying metric according to information theoretic principles, see e.g. [13, 18, 25, 8].

In principle, every such technique could be used in the given approach. Here, we use Limited Rank Matrix LVQ (LiRaM LVQ) [4] since it constitutes a very fast nonlinear discriminative dimensionality reduction technique which preserves the local manifold structure as much as possible, being a locally linear approach.

4 Inverse nonlinear dimensionality reduction

Assume a nonlinear projection of points $\mathbf{x}_i \in X$ to $p(\mathbf{x}_i) = \mathbf{y}_i \in \mathbb{R}^2$ and additional data points $\mathbf{z}'_i \in \mathbb{R}^2$ are given. What are points $\mathbf{z}_i$ such that its projections approximate $\mathbf{z}'_i \approx p(\mathbf{z}_i)$ and, in addition, $\mathbf{z}_i$ are contained in the data manifold? Some problems have to be considered: usually, an explicit mapping p is not given, rather only discrete projections of the data, albeit a few approaches to extend a mapping of points to a mapping of data have recently been proposed for the general case [9, 3]. Second, since X is high dimensional, the projection p is in general not invertible.

Here, we propose an interpolation technique similar to the kernel mapping as introduced in [9] for nonlinear dimensionality reduction which interpolates the given pairs $(p(\mathbf{x}_i), \mathbf{x}_i)$. Since this mapping will be trained on the points corresponding to the data manfiold X only, inverse points which correspond to this subset result for arbitrary $\mathbf{z}'_i$. We assume the following functional form

$$p^{-1} : Y \to X, \mathbf{y} \mapsto \frac{\sum_i \alpha_i k_i(\mathbf{y}_i, \mathbf{y})}{\sum_i k_i(\mathbf{y}_i, \mathbf{y})} = \mathbf{A}\mathbf{k}$$

where $\alpha_i \in X$ are parameters of the mapping and $k_i(\mathbf{y}_i, \mathbf{y}) = \exp(-\|\mathbf{y}_i - \mathbf{y}\|^2/\sigma_i^2)$ constitutes a Gaussian kernel with bandwidth determined by σ_i. The matrix $\mathbf{A}$ contains the vectors α_i in its columns and $\mathbf{k}$ is a column vector with the ith element being $\frac{k_i(\mathbf{y}_i, \mathbf{y})}{\sum_i k_i(\mathbf{y}_i, \mathbf{y})}$. Summation is over a subset of the given data projections $\mathbf{y}_i = p(\mathbf{x}_i)$, often including the most informative points such as the projections of support vectors for an SVM or the prototypes of an LVQ classifier. Depending on the choice of these data and the bandwidth, the problem can constitute an overdetermined system of equations such that we rely on a least squares solution of

$$\min_{\mathbf{A}} \sum_j \left\|\mathbf{x}_j - \mathbf{A}\mathbf{k}_j\right\|^2.$$

This can directly be computed using the Moore-Penrose pseudo inverse, since the mapping is of generalized linear form. Depending on the form of the original data manifold, this inverse function maps the Euclidean plane to that part of the data manifold which centers around the data, thereby neglecting irrelevant dimensions due to the choice of p. In addition to an optimization of the parameters α_i, an optimization of the bandwidth parameters σ_i is possible by means of a gradient technique.

5 Experiment

We exemplarily demonstrate this technique for a standard SVM and a Generalized Matrix LVQ (GMLVQ) classifier as introduced in [21]. Both techniques provide a distance of a data point to the closest class boundary in addition to the class label itself. For LVQ, this is given by the normalized hypothesis margin, given by $(d_2 - d_1)/(d_1 + d_2)$ assuming d_1 and d_2 constitute the distances to the closest two prototypes with different class labels. For the SVM, a one versus one classification and majority vote is used for multiple classes based on the LIBSVM implementation [6]. The margin is determined by the SVM which label corresponds to the final output and which displays the smallest margin. For visualization, discriminative LiRaM LVQ as described in [4] is used.

Gene expression data

Using matrix LVQ techniques has the benefit that, due to the local linearity of the classifier and the explicit mapping provided by (local) LiRaM LVQ [4], an explicit computation of class boundaries is possible. The visualization result obtained by our framework can be compared based on ground truth. Nevertheless, our visualization technique can be applied to any given classifier.

The gene expression data set consists of 83 samples described by 50 characteristics of small round blue cell childhood tumor. Four different classes are given. For the projection to the high dimensional space, 43 randomly chosen points are used. The width $\sigma = 0.8$ is assigned to all kernels in the 2D space. In Fig. 2, the original class boundaries obtained with a local matrix LVQ technique and the class separation obtained using our nonlinear mapping technique are displayed. In addition to the class boundaries, the values of the normalized hypothesis margin are displayed. These values are coded as the color intensity in the image, allowing for a visual judgement of the certainty of the classification results. Furthermore, black contour lines code regions with the same margin value. Obviously, very good agreement between the ground truth and our approximation can be observed.

Letter data set

The UCI letter data set consists of $20,000$ handwritten letters with a resolution of 16×16 grey values. For training and visualization, a subset of $1,300$ points is used. The visualization of a trained matrix LVQ network and a SVM is displayed in Fig. 3. For both classifiers, the same visualization parameters are used: The kernel widths are calculated per class by averaging the distances from each point of that class to its nearest neighbour with additional scaling by a constant. As before, the margin is coded by the color intensity, while this time, the contour lines are omitted for the sake of clarity.

Again, a very clear visualization of the classification result is obtained this way.

6 Conclusions

We have proposed a general framework how to nonlinearly visualize a data set and a given classifier in low dimensions. We have demonstrated the usefulness of this technique in two examples. Further experiments incorporating alternative discriminative visualization techniques, further classifiers, and more complex data sets which include a significant overlap are the subject of ongoing work.

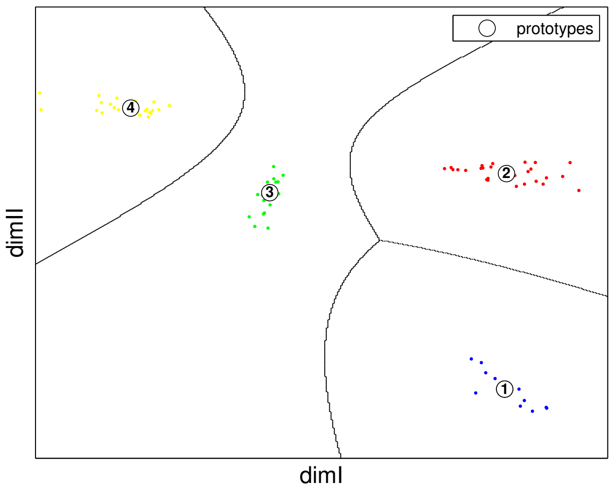
Further open problems concern a formal evaluation measure for the given visualization. Possibilities are given by a comparison of the classification accuracy of the classifier in the original space and in the projection. Further, extensions of the standard co-ranking framework [15] to include the Fisher information as proposed e.g. in [8] could be extended to also take the class boundaries into account.

References

1. G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12:2385–2404, 2000.

2. M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396, 2003.
3. K. Bunte, M. Biehl, and B. Hammer. A general framework for dimensionality reducing data visualization mapping. *Neural Computation*, 24(3):771–804, 2012.
4. K. Bunte, P. Schneider, B. Hammer, F.-M. Schleif, T. Villmann, and M. Biehl. Limited rank matrix learning, discriminative dimension reduction and visualization. *Neural Networks*, 26:159–173, 2012.
5. D. Caragea, D. Cook, H. Wickham, and V. Honavar. Visual methods for examining svm classifiers. In S. J. Simoff, M. H. Böhlen, and A. Mazeika, editors, *Visual Data Mining*, volume 4404 of *Lecture Notes in Computer Science*, pages 136–153. Springer, 2008.
6. C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
7. D.Cohn. Informed projections. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS*, pages 849–856. MIT Press, 2003.
8. A. Gisbrecht and B. Hammer. Discriminative dimensionality reduction mappings, submitted.
9. A. Gisbrecht, W. Lueks, B. Mokbel, and B. Hammer. Out-of-sample kernel extensions for nonparametric dimensionality reduction. In *ESANN 2012*, pages 531–536, 2012.
10. J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17*, pages 513–520. MIT Press, 2004.
11. J. Hernandez-Orallo, P. Flach, and C. Ferri. Brier curves: a new cost-based visualisation of classifier performance. In *International Conference on Machine Learning*, June 2011.
12. A. Jakulin, M. Možina, J. Demšar, I. Bratko, and B. Zupan. Nomograms for visualizing support vector machines. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 108–117, New York, NY, USA, 2005. ACM.
13. S. Kaski, J. Sinkkonen, and J. Peltonen. Bankruptcy analysis with self-organizing maps in learning metrics. *IEEE Transactions on Neural Networks*, 12:936–947, 2001.
14. J. A. Lee and M. Verleysen. *Nonlinear dimensionality redcution.* Springer, 2007.
15. J. A. Lee and M. Verleysen. Scale-independent quality criteria for dimensionality reduction. *Pattern Recognition Letters*, 31:2248–2257, 2010.
16. B. Ma, H. Qu, and H. Wong. Kernel clustering-based discriminant analysis. *Pattern Recognition*, 40(1):324–327, 2007.
17. R. Memisevic and G. Hinton. Multiple relational embedding. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 913–920. MIT Press, Cambridge, MA, 2005.
18. J. Peltonen, A. Klami, and S. Kaski. Improved learning of riemannian metrics for exploratory analysis. *Neural Networks*, 17:1087–1100, 2004.
19. F. Poulet. Visual svm. In C.-S. Chen, J. Filipe, I. Seruca, and J. Cordeiro, editors, *ICEIS (2)*, pages 309–314, 2005.
20. S. Rüping. *Learning Interpretable Models.* PhD thesis, Dortmund University, 2006.
21. P. Schneider, M. Biehl, and B. Hammer. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21:3532–3561, 2009.
22. J. Tenenbaum, V. da Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

23. L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.
24. A. Vellido, J. Martin-Guerroro, and P. Lisboa. Making machine learning models interpretable. In *ESANN'12*, 2012.
25. J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *Journal of Machine Learning Research*, 11:451–490, 2010.
26. X. Wang, S. Wu, X. Wang, and Q. Li. Svmv - a novel algorithm for the visualization of svm classification results. In J. Wang, Z. Yi, J. Zurada, B.-L. Lu, and H. Yin, editors, *Advances in Neural Networks - ISNN 2006*, volume 3971 of *Lecture Notes in Computer Science*, pages 968–973. Springer Berlin / Heidelberg, 2006.
27. M. Ward, G. Grinstein, and D. A. Keim. *Interactive Data Visualization: Foundations, Techniques, and Application.* A. K. Peters, Ltd, 2010.



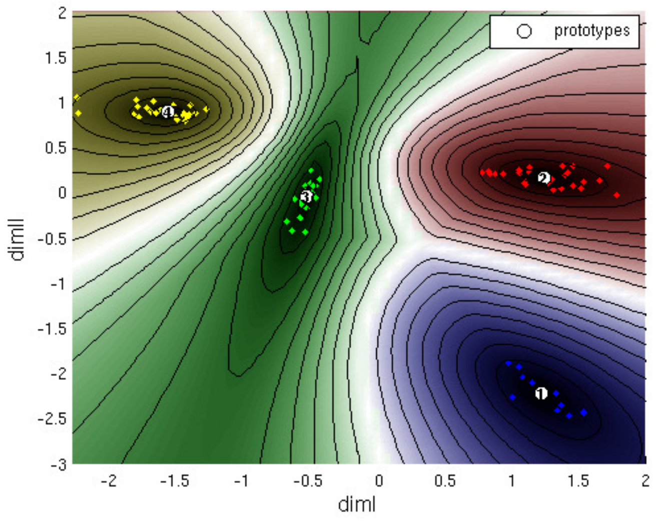
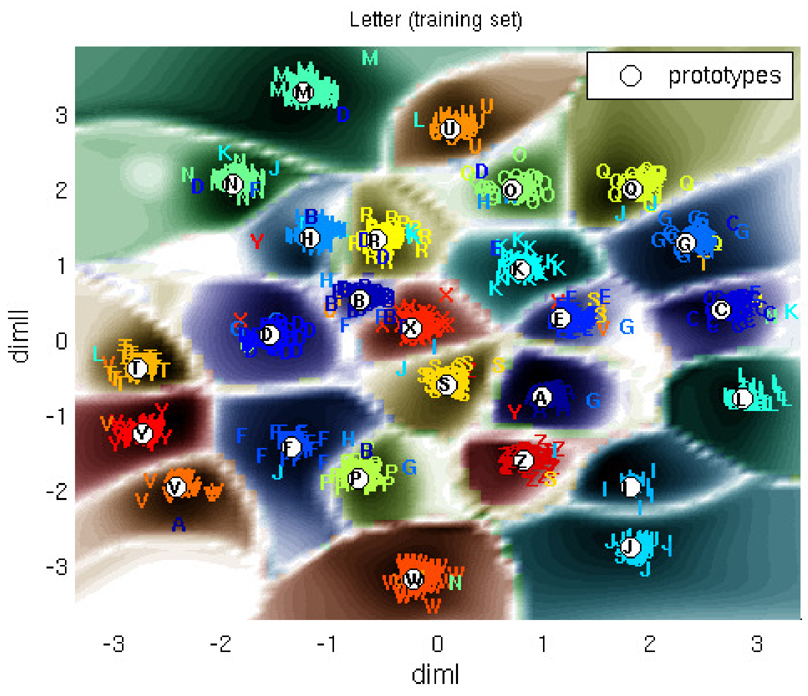


Fig. 2: Visualization of the Gene data set showing the original class boundaries of LiRaM LVQ (top) and our sampling strategy (bottom).



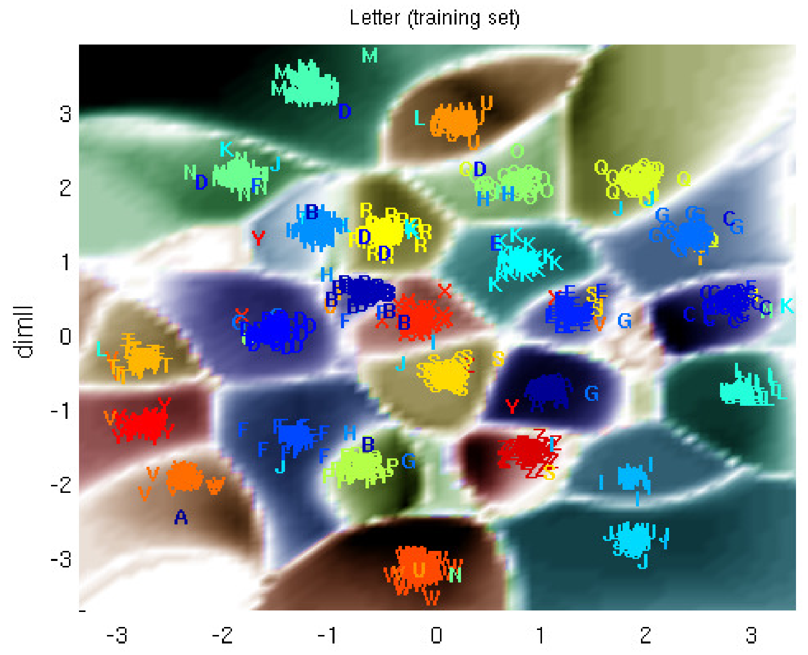


Fig. 3: Visualization of the Letter data set and a matrix LVQ classifier (top) or a SVM (bottom), respectively.

Locally Weighted Regression using an Error–based Allocation Strategy

Slobodan Vukanović, Nicole Carey, Robert Haschke, Helge Ritter

Cognitive Interaction Technology Excellence Cluster (CITEC)*, Bielefeld University

Locally weighted regression (LWR) approximates the target function with a number of spatially localized Gaussian weighted linear models. The output to a given query is computed as a combination of the outputs of the local regression models [1]. LWR is robust to destructive interference, and the complexity of the approximating function can be incrementally increased by adding new models, making it a suitable choice for learning in online scenarios, where data is discarded after updates and where input and output distributions are unknown and prone to change over time [2]. Implementations of LWR must address the structure of the models, the definition of locality of its models, and the strategy that allocates new models [3]. This work focuses on adapting the model allocation strategy of LWPR [4], a state–of–the–art [3] receptive field based LWR algorithm, with the goal of simplifying its initialization.

A receptive field – the region that defines locality of each local model – is described by an ellipsoid with the center $\mathbf{c}$ in the n–dimensional input space and a distance metric $\mathbf{D}$, a positive–definite matrix:

$$(\mathbf{x} - \mathbf{c})^T \mathbf{D}(\mathbf{x} - \mathbf{c}) = 1 \tag{1}$$

The decision to insert a new model is based in the input space. If no existing model yields an activation above a certain threshold, w_{gen}, a new local model is created with its center $\mathbf{c}$ set to the input datum and its distance metric $\mathbf{D}$ set to $\mathbf{D}_{init}$, a critical open parameter. Certain values of $\mathbf{D}_{init}$ will create many models while others will create only a few models. Each of these situations may pose problems: training many models is computationally expensive, and, with a small number of models, many examples are necessary for the network to converge to the target function.

We propose an allocation strategy that adds local models based on the current predictive ability of the network, as measured by its error. The aim is to achieve a good approximation of the target function with as few models as possible. The strategy will simplify the initialization of LWPR by removing the $\mathbf{D}_{init}$ and w_{gen} parameters.

* This work was supported by the DFG CoE 277: CITEC.

Rather than allocating new models in the parts of the input space that have not yet been covered, our strategy starts with an initial model that covers as much of the input space as possible. The model is then subdivided (split) based on the comparison of the desired error to the network's current error. The splitting continues until the desired error is reached. A candidate for splitting is chosen based on its local error e_{rf}, which is the history of its contributions to the prediction error:

$$e_{rf}^{n+1} = \frac{w^n e_{rf}^n + w(||\mathbf{y} - \hat{\mathbf{y}}||^2)}{w^n + w} \tag{2}$$

where w is the current activation weight of the receptive field, w^n are the accumulated weights, $\mathbf{y} \in R^m$ is the target output, and $\hat{\mathbf{y}} \in R^m$ is the output of the network. The existing receptive field is split geometrically into two equal receptive fields that cover the same space. An example of splitting in two dimensions is shown in Figure 1.

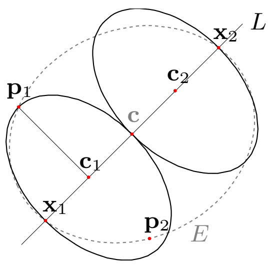


Fig. 1. A two–dimensional receptive field described by the ellipse E (dashed) centered at $\mathbf{c}$ split into two equal parts along the line L. The centers $\mathbf{c}_1$, $\mathbf{c}_2$ of the new receptive fields lie on L, halfway between $\mathbf{c}$ and the points $\mathbf{x}_1$, $\mathbf{x}_2$ of intersection of L with E. One of the semi–axes of the new receptive fields lies on L. The other is perpendicular to it, and its length is $max(||\mathbf{p_1} - \mathbf{c_1}||, ||\mathbf{p_2} - \mathbf{c_1}||)$.

The initial results of training LWPR with default and our allocation strategies are shown in Figure 2. The training set of size 20000 was generated from $y = max\{exp(-10x_1^2), exp(-50x_2^2), 1.25exp(-5(x_1^2 + x_2^2))\} + N(0, 0.05)$. For the default implementation of LWPR, the initial shape of the receptive fields and the threshold for adding local models were set to the recommended values of $\mathbf{D}_{init} = 50\mathbf{I}$ (where $\mathbf{I}$ is the identity matrix) and $w_{gen} = 0.2$ respectively [4]. The testing set consisted of 1681 noiseless samples from a 41×41 grid on the unit square in input space. LWPR with the default allocation strategy creates 59 receptive fields and has a mean square error of 0.002353 on the testing data. LWPR with our allocation strategy creates 50 receptive fields and has a mean square error of 0.002924 on the testing data. We thus achieve comparable approximation of the target function without having to specify the $\mathbf{D}_{init}$ and w_{gen} parameters. The accuracy of the network using our error–based allocation strategy is slightly lower than that of the default LWPR strategy This is attributed to the empty spaces between the receptive fields that result from the adaption of their shapes. LWPR's receptive field shape update rule considers the network's current prediction error, and the receptive fields can shrink if the error is too high. Since our strategy starts with only a few receptive fields, they will initially shrink as the network does not have enough resources to learn the complex target function. This is evident in Figure 2**(e)**, where the initial model is split vertically into two receptive fields that shrink in the location of high curvature where the error is high. Since the existing receptive fields are only split, no new receptive fields will be allocated in locations that are not covered. As a result, the peak of the cross function is not learned well (Figure 2**(c)**).

Our current work involves increasing the accuracy of our strategy and extending it to handle shifting input distributions.

References

1. C. G. Atkeson, A. W. Moore, and S Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.
2. Stefan Schaal and Christopher G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10:2047–2084, 1997.
3. O. Sigaud, C. Salaun, and V. Padois. On-line regression algorithms for learning mechanical models of robots: a survey. *Robotics and Autonomous Systems*, 59(12):1115–1129, December 2011.
4. Sethu Vijayakumar, Aaron D'Souza, and Stefan Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17:2602–2634, 2005.

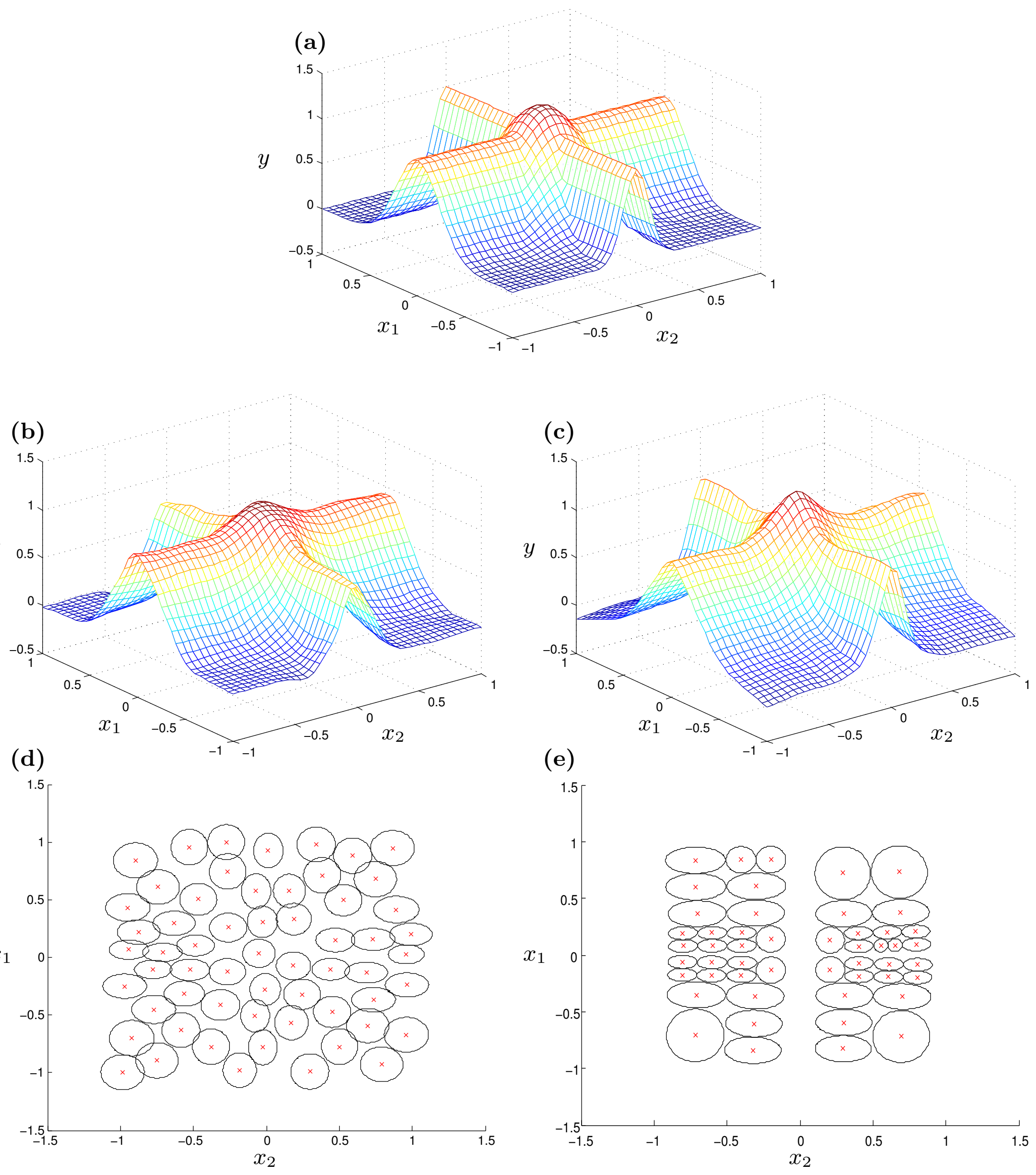


Fig. 2. **(a)** The target function. **(b)** The function learned with LWPR's default allocation strategy. **(c)** The function learned with our allocation strategy. **(d)** Receptive fields created by LWPR's default allocation strategy visualized in the input space. **(e)** Receptive fields created by our allocation strategy visualized in the input space.

Classification in High-dimensional Spectral Data – Precision vs. Interpretability vs. Model Size

Andreas Backhaus and Udo Seiffert

Fraunhofer IFF, Magdeburg, Germany
[Andreas.Backhaus, Udo.Seiffert]@iff.fraunhofer.de

Abstract. This paper evaluates aspects of *precision*, *interpretability* and *model size* of several computational intelligence based classification methods in the context of hyperspectral imaging and Raman spectroscopy. It is focussed on state-of-the-art representative paradigms of a number of different concepts, such as prototype based, kernel based, and support vector based approaches.

Keywords: classification, Learning Vector Quantization (LVQ), Radial Basis Function (RBF) networks, Support Vector Machines (SVM), hyperspectral imaging, Raman spectroscopy.

1 Introduction

Hyperspectral imaging as recent extension to traditional non-invasive spectroscopic analysis techniques (e.g. NIR spectroscopy) has paved the way to obtain the biochemical constitution of inspected solid materials with the additional advantage of a two-dimensional spatial resolution. For the examination of liquid sample, Raman spectroscopy has been shown to be a viable tool to gather information without sample preparation [1].

Regarding pattern recognition and data mining in the acquired spectral data, computational intelligence based methods are still providing powerful tools to cope with this kind of high-dimensional and complex data (see Fig. 1, left panel).

From the computational intelligence point of view the recent developments in hyperspectral camera technology with increasingly high resolution in both the spectral and spatial domain have led to high-dimensional input spaces and a large number of training vectors. Both aspects even more motivate and demand computational intelligence based algorithms.

Besides unsupervised visualisation and clustering typically used to get a (first) graphical representation of the acquired spectral data, classification and multivariate regression is often required by the underlying application. Here, corresponding labelled data is necessary. Since suitable wet lab analysis to provide continuously valued reference data are typically expensive, frequently categorical labels are provided. This leads to a classification task. Industrial applications in

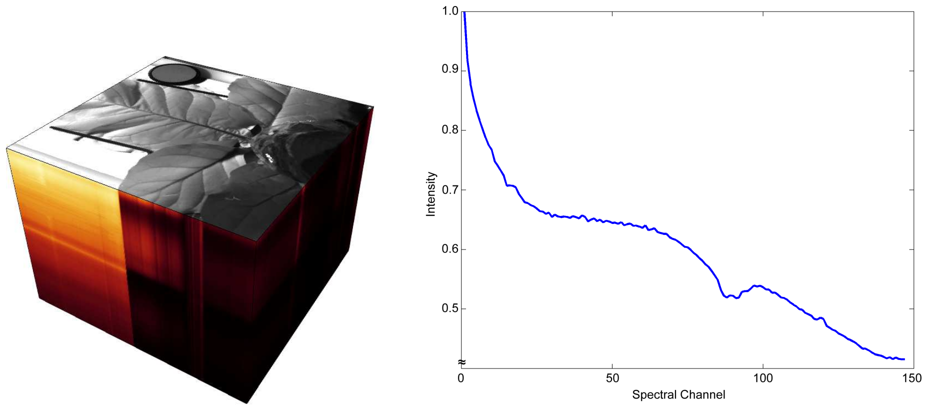


Fig. 1. Left: This figure illustrates the typical data cube obtained by hyperspectral imaging. It consists of three dimensions – two dimensions covering the spatial resolution and a third one containing the spectral data at each spatial position. A particular pixel is characterised by a vector containing the spectral reflectance along the acquired wavelength bands. The number of acquired wavelength bands represents the input dimensionality of subsequent data processing, whereas each pixel is one training sample. **Right**: This figure demonstrates a sample spectral fingerprint acquired at a particular position of a hyperspectral image. It shows the normalised intensity over the acquired spectral bands.

product quality control and sorting also demand on-line classification at a low systems cost.

Therefore this classification task has in general three, sometimes conflicting, objectives to address. The first objective is a classification model of high accuracy. The second objective is a as small as possible classification model for quick calculation. A third objective is the restriction to necessary information / features of the examined objects for the classification task at hand. In spectral data processing this means the restriction of necessary spectral bands. This not only speeds up calculation but also leads to less expensive spectral sensor systems. Therefore classification models need to offer a certain degree of interpretability. Relevance profiles for example can indicate the importance of the used input variables, in this case the acquired spectral bands. Additionally, classification models should require small or no expert interference in order to tune model parameters which could lead to biased, non-optimal decisions by the user.

Keeping these requirements in mind, a number of computational intelligence paradigms appear to be particularly suitable. Among them are prototype-based neural networks, such as the LVQ (Learning Vector Quantisation) family, RBF (Radial Basis Function) networks, and Support Vector Machines (SVM). These three approaches span the scope of the presented paper. The qualification of these three different approaches regarding classification data from the hyperspectral imaging domain as well from Raman spectra data in terms of several theoretical considerations as well as practical aspects is evaluated. In order to

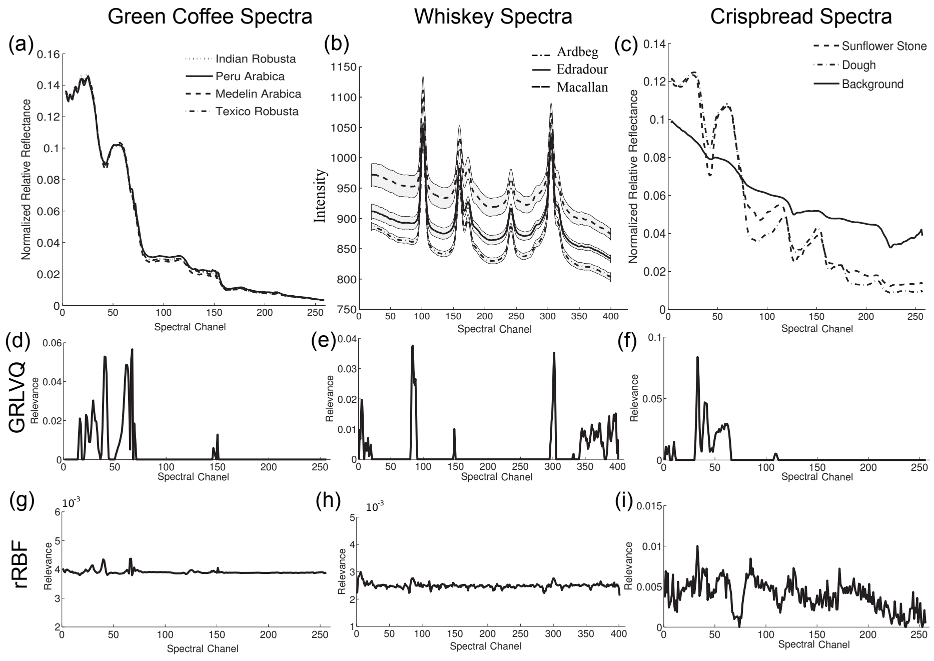


Fig. 2. Datasets and relevance profiles: (a)-(c) show the mean spectra per class for the different datasets used in this study; (d)-(i) depict the relevance profiles acquire for with the rRBF and GRLVQ for the different classification tasks; GRLVQ clearly acquired highly sparse profiles while the rRBF profiles are flat or less specific.

derive practically relevant information from this study, several real-world data set are used.

2 Classification Problems

2.1 Green Coffee Spectra

Quality control of coffee products, from basic green coffee to the finished roasted coffee by hyperspectral imaging offers the means for a non-invasive, on-line and automated screening method to control large product quantities [2, 3]. For example green coffee has to be inspected for the Robusta or Arabica varieties since Arabica based coffee is sold at a different price then Robusta based coffee. The spatial resolution of hyperspectral imaging makes it the ideal tool for loose material sorting especially in the case where information from color, shape or texture is not sufficient for differentiation.

For the hyperspectral image acquisition coffee beans of four different green coffee varieties, two varieties of Arabica and two varieties of Robusta and a standard optical PTFE (polytetrafluoroethylene) calibration pad were positioned on a translation table. Hyperspectral images were recorded using a HySpex SWIR-320m-e line camera (Norsk Elektro Optikk A/S). Spectra are from the short-wave infra-red range (SWIR) of 970 nm to 2,500 nm at 6 nm resolution yielding a 256 dimensional spectral vector per pixel. The camera line has a spatial resolution of 320px and can be recorded with a maximum frame rate of 100fps. Radiometric calibration was performed using the vendors software package. Coffee beans were segmented from background via Neural Gas clustering. Spectra are normalized to a vector length of one. The dataset comprised of the four green coffee varieties forming a 4-class problem with 2000 spectra per class. Figure 2a shows average spectra for the four green coffee classes.

2.2 Whisky Spectra

The automated, on-line assessment of high-priced liquor products is essential for the standardization and quality monitoring in liquor production as well as potential fraud detection. An ideal sensor should be compact for mobile applications and require no special sample preparation while measure quality instantaneously. In [1] an optofluidic chip was presented that uses Raman spectroscopy to acquire a Raman spectrum of the fluid sample.

The procedure to acquire the Raman spectra from whisky samples is shown in detail in [1]. In Raman spectroscopy a sample is illuminated with a laser beam. The laser light interacts with molecular vibrations, phonons or other excitations in the system, resulting in the energy of the laser photons being shifted up or down. The shift in energy gives information about the vibrational modes in the system. Raman spectroscopy is commonly used in chemistry, since vibrational information is specific to the chemical bonds and symmetry of molecules. Therefore, it provides a fingerprint by which molecules can be identified.

Whisky samples of $20\mu l$ were directly loaded into the microfluidic chip without any preparation. After Raman acquisition, any remaining liquid at the sample inlet was wiped off and 40 μl of deionized water rinsed the system. Raman excitation was performed with 200 mW of laser power at a wavelength of 785 nm.

Six commercially available Scotch whisky brands and their variants were used to build the dataset. All available data was labeled according to their distillery of origin resulting in a 6-class problem. For each class, 400 Raman spectra were available. Each dataset was scaled so the maximum across spectral bands was one. Figure 2b shows average spectra for three whisky classes with standard deviation.

2.3 Crispbread Spectra

Hyperspectral imaging offers the possibility to examine the spatial distribution and degree of coverage of food ingredients on the product surface in order to check with the design reference. As an example we consider the segmentation problem of sunflower stones on crispbread. Here a classifier of very high accuracy is needed to label each pixel with their respective class. Subsequent processing steps could remove some classification errors but have to become more sophisticated with lower classification accuracy.

Crispbread Samples containing sunflower stones where placed with a standard optical PTFE calibration pad on a translation table. Hyperspectral images were recorded using the same HySpex SWIR-320m-e line camera as used in the coffee dataset. Dough, sunflower stones and background material where manually marked and formed a 3-class problem with 616 spectra per class. Radiometric calibration and spectra preprocessing was identical to the coffee dataset. Figure 2c shows the mean spectra per class.

3 Machine Learning

For machine learning, three different classification models are considered, the Radial Basis Function (RBF) Network with Relevance Learning [4, 5], Generalized Relevance Learning Vector Quantization (GRLVQ) [6] as well as a Support Vector Machine [7]. RBF and GRLVQ Networks are similar in terms that they process the input data in a layer of prototypical data points. While the RBF generates activation due to the similarity with prototypes which are accumulated in a second layer for the network output, the GRLVQ directly assigns classes to prototypical data points. Prototypes usually represent central positions in a data cloud. In contrast, the Support Vector Machines stores support vectors, e.g. data points from the border of a data cloud. The used Support Vector Machine implementation from the freely available libSVM package[1] takes up a variable amount of support vectors.

In order to compute the distance of spectral data point v and a prototype w in the rRBF and GRLVQ, we used the weighted Euclidean distance metric

$$d\left(\mathbf{v}, \mathbf{w}_r, \boldsymbol{\lambda}\right) = \sum_i \lambda_i \left(v_i - w_{ir}\right)^2 \tag{1}$$

where λ_i is the relevance factor per spectral band which is adapted during the learning process to form the relevance profile. The rRBF and GRLVQ learning approach is in both cases essentially an energy minimization problem. In the standard learning scheme, stochastic gradient descent with step-sizes manually set for different parameters are used. In order to avoid a manually chosen parameter, we used the non-linear conjugate gradient approach with automatic step size from the freely available Matlab optimization toolbox 'minFunc' [2]. For this purpose we had to provide the objective/energy function along with the first derivatives according to the optimization parameters.

For the rRBF the objective function is the accumulated quadratic error of the network output y and target value t across network outputs and data samples.

[1] *www.csie.ntu.edu.tw/˜cjlin/libsvm/*
[2] http://www.di.ens.fr/ mschmidt/Software/minFunc.html

$$E(\mathbf{V}, \mathbf{W}, \boldsymbol{\lambda}) = \frac{1}{2} \sum_j \sum_k \left\{ y_k \left(\mathbf{v}^j\right) - \mathbf{t}_k^j \right\}^2 \tag{2}$$

with $y_k(\mathbf{v}) = \sum_r u_{rk} \phi(d(\mathbf{v}, \mathbf{w}_r, \boldsymbol{\lambda}))$ and $\phi(x) = \exp\left(-\frac{x}{2\sigma^2}\right)$. The partial derivatives are as follows

$$\frac{\partial E}{\partial w_{ir}} = \sum_j \sum_k \left\{ y_k \left(\mathbf{v}^j\right) - \mathbf{t}_k^j \right\} u_{rk} \phi\left(d\left(\mathbf{v}^j, \mathbf{w}_r, \boldsymbol{\lambda}\right)\right) \frac{\left(x_i^j - w_{ir}\right)}{\sigma_r^2} \tag{3}$$

$$\frac{\partial E}{\partial \sigma_r} = \sum_j \sum_k \left\{ y_k \left(\mathbf{v}^j\right) - \mathbf{t}_k^j \right\} u_{rk} \phi\left(d\left(\mathbf{v}^j, \mathbf{w}_r, \boldsymbol{\lambda}\right)\right) \frac{\sum_i \lambda_i \left(v_i - w_{ir}\right)^2}{\sigma_r} \tag{4}$$

$$\frac{\partial E}{\partial \lambda_i} = -\sum_j \sum_k \left\{ y_k \left(\mathbf{v}^j\right) - \mathbf{t}_k^j \right\} \sum_r u_{rk} \phi\left(d\left(\mathbf{v}^j, \mathbf{w}_r, \boldsymbol{\lambda}\right)\right) \frac{\left(v_i^j - w_{ir}\right)^2}{2\sigma_r^2} \tag{5}$$

The output weight u_{rk} are yielded by direct update $\mathbf{U}^T = \Phi^\dagger \mathbf{T}$ where † denotes the pseudo inverse [8]. For the classification task a 1-out-of-N coding scheme for the target vector was used. For the GRLVQ the objective function is the accumulated difference in shortest distance of a data point to a prototype representing its class d_r^+ and a prototype representing any other class d_r^-:

$$E(\mathbf{V}, \mathbf{W}, \boldsymbol{\lambda}) = \sum_{v \in V} \Phi\left(\frac{d_r^+ - d_r^-}{d_r^+ + d_r^-}\right). \tag{6}$$

The partial derivatives are as following

$$\frac{\partial E}{\partial w_{ir}^+} = -\frac{2 \cdot d_r^-}{\left(d_r^+ + d_r^-\right)^2} 2\left(v_i - w_{ir}^+\right) \qquad \frac{\partial E}{\partial w_{ir}^-} = \frac{2 \cdot d_r^+}{\left(d_r^+ + d_r^-\right)^2} 2\left(v_i - w_{ir}^-\right) \tag{7}$$

$$\frac{\partial E}{\partial \lambda_i} = \frac{2 \cdot d_r^-}{\left(d_r^+ + d_r^-\right)^2} \left(v_i - w_{ir}^+\right)^2 - \frac{2 \cdot d_r^+}{\left(d_r^+ + d_r^-\right)^2} \left(v_i - w_{ir}^-\right)^2 \tag{8}$$

All partial derivatives not belonging to the winning prototype of same class w_r^+ and any other class w_r^- is set to zero. The derivatives are accumulated for all data points (batch learning).

Both networks, rRBF and GRLVQ are trained till the step size fell below a threshold with a maximum number of allowed iterations and function evaluations respectively. For the classification, the number of prototypes per class is varied in the GRLVQ. To ensure similar model sizes, the rRBF number of prototypes is always the number of class times number of prototypes per class in the GRLVQ. Before the training with the respective method was started, prototypes were pre-trained using the Neural Gas algorithm. In the rRBF all training data is used to

Table 1. Test accuracies for the three classification tasks; Results are averaged across a 5-fold cross-validation with standard deviation in brackets. The model size denotes the total number of prototypes (rRBF, GRLVQ) and support vectors (SVM) respectively.

Method	Green Coffee Data 4 classes		Crispbread Data 3 classes		Whiskey Data 6 classes	
	Accuracy	Size	Accuracy	Size	Accuracy	Size
rRBF	0.430 (0.018)	4	0.964 (0.011)	3	0.753 (0.012)	6
	0.901 (0.081)	12	0.988 (0.004)	9	0.946 (0.013)	12
	0.962 (0.020)	16	0.989 (0.006)	15	0.957 (0.009)	24
	0.963 (0.018)	24	0.990 (0.004)	30	0.954 (0.002)	36
	0.967 (0.011)	32	0.983 (0.004)	45	0.963 (0.007)	48
GRLVQ	0.821 (0.011)	4	0.958 (0.003)	3	0.803 (0.009)	6
	0.870 (0.017)	12	0.986 (0.006)	9	0.885 (0.013)	12
	0.821 (0.016)	16	0.987 (0.005)	15	0.918 (0.012)	24
	0.872 (0.033)	24	0.988 (0.006)	30	0.929 (0.011)	36
	0.902 (0.027)	32	0.984 (0.004)	45	0.939 (0.014)	48
SVM (linear)	0.969 (0.003)	3159.2 (7.855)	0.984 (0.006)	176 (5.292)	0.841 (0.007)	1590.4 (44.97)

adapt all prototypes while in GRLVQ prototypes are learnt only on the training data of their respective class. Especially for the GRLVQ using a large number of prototypes this proved to increase model performance significantly in comparison to random initialisation or initialisation at central class data positions. Data was divided into training and test set according to a 5-fold cross validation. Classification accuracy was averaged and standard deviation was computed.

4 Results

Table 1 shows the test accuracy values for all three classification tasks for all methods and chosen model sizes. Methods generally reached high accuracy values in all classification task showing that all three problems are solvable by machine learning of spectral data. In order to achieve a high classification accuracy, the SVM classifier took up a very high number of support vectors for example over three thousand for the coffee sample. This poses a significant obstacle for the implementation of real time classification methods. In comparision, GRLVQ and rRBF reached similar level of accuracy with significantly smaller number of prototype, e.g. model sizes. For the obviously 'challenging' classification tasks of classifying green coffee and whiskey distilleries, the rRBF also showed better performance at smaller model sizes then the GRLVQ, with exception in the case of one prototype per class. However, the RBF contains a second layer which contributes to the computational steps as well as the calculation of the exponential function while the GRLVQ only needs to perform the comparison of a data vector with all prototypes vectors. Still, a SVM is much easier to handle in terms of model training. Optimization methods with automatic step sizes in the GRLVQ and rRBF however decrease the complexity for the user significantly.

In Figure 2d-i the relevance profile of the rRBF and GRVLQ are depicted. This offers some interpretability to the classification model and the information it bases its decision on which is lacking in the Support Vector Machine approach. It is very obvious that the relevance learning in the GRLVQ lead to much more pronounced and sparse relevance profiles. This ability is especially useful to reduce the number of spectral bands or narrow down the spectral range that is important to the classification task to reduce system cost significantly. On the other hand, relevance profiles of the rRBF are flat or very unspecific and not as clear as the GRLVQ profiles. This might explain the ability of the rRBF to classify data with a smaller model size using the full range of spectral information. In further work, classification accuracy and model sizes of method with and without relevance learning should be compared.

5 Conclusion

The practical application of machine learning methods in spectral data processing offers an efficient approach to generate classification models without deeper insight into the chemical and physical processes underlying a spectral signature. This approximation ability and the flexibility of machine learning method opens the way to inspection systems for multiple applications based on pattern producing sensor hardware. However, the creation of classification models from reference data is still an expert task.

Support Vector Machine gained an increasing popularity due to its relative easiness of use at high accuracy levels. As this study showed, SVM can become impractical for real time implementation due to the high number of support vectors. GRLVQ and rRBF, offering smaller model sizes, are still depending on too many learning parameters. Using more advanced optimisation method, parameters that have to be fine tuned by the user could be reduced. In further research, the application of growing model structures should be considered to avoid the need to choose an explicit model size. The question remains how to integrate the change of model size into the objective function. Optimization approaches that do not need a differentiable objective function like genetic algorithms or simplex algorithms should be considered here.

The relevance learning offers a level of interpretability that becomes very useful in the area of spectral data processing. System cost are still too high for a number of applications. Reducing the number of spectral information needed for a particular task will help reduce cost as well as increase the processing speed. The GRLVQ showed very good capability to classify spectral data with a minimal amount of information while the rRBF did not produce sparse relevance profiles. For applications, where the full spectral range is available, the rRBF might proof advantageous in terms of accuracy at small model sizes.

Acknowledgements

The authors want to thank Barbara Hammer and Thomas Villmann for motivating this study under practical considerations. We would like to thank Röstfein Magdeburg as well as Thomas Villmann and Marika Kästner for providing coffee samples. Furthermore, the authors are grateful to Praveen Ashok, Bavishna Praveen and Kishan Dholakia for providing us with extensive data sets of Scotch Whisky samples.

References

1. Praveen C. Ashok, Bavishna B. Praveen, and K. Dholakia. Near infrared spectroscopic analysis of single malt scotch whisky on an optofluidic chip. *Opt Express*, 19(23):22982–22992, Nov 2011.
2. A.G. Fiore, R. Romaniello, G. Peri, and C. Severini. Quality assessment of roasted coffee blends by hyperspectral image analysis. In *In Proc. 22nd International Conference on Coffee Science*, Campinas, Brazil, 2008.
3. Andreas Backhaus, Felix Bollenbeck, and Udo Seiffert. High-throughput quality control of coffee varieties and blends by artificial neural networks from hyperspectral imaging. In F. Travaglia, M. Bordiga, J.D. Coïsson, M. Locatelli, V. Fogliano, and M. Arlorio, editors, *Proceedings of the 1st International Congress on Cocoa, Coffee and Tea (CoCoTea)*, volume 1, pages 88–92, Novara, Italy, 2011.
4. John Moody and Christian J. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, 1:281–294, 1989.
5. Andreas Backhaus, Felix Bollenbeck, and Udo Seiffert. Robust classification of the nutrition state in crop plants by hyperspectral imaging and artificial neural networks. In *In Proc. 3rd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*, Lissabon, Portugal, 2011.
6. Barbara Hammer and Thomas Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15:1059–1068, 2002.
7. Vladimir Vapnik and Alexey Chervonenkis. *Theory of Pattern Recognition.* Nauka, 1974.
8. Cristopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press., 1995.

Feature extraction from Occupancy Grid Maps using Non-negative Matrix Factorization

Marian Himstedt, Sven Hellbach, Hans-Joachim Boehme*

Artificial Intelligence Lab, University of Applied Sciences, Dresden, Germany
{ himstedt; hellbach; boehme }@htw-dresden.de

1 Introduction

Localization and mapping are fundamental problems in the field of mobile robotics. In order to navigate autonomously a mobile robot requires the knowledge about its position and orientation within the surrounding environment. If both, map and position, are unknown, the robot has to concurrently maintain estimates about its position as well as the traversable environment which is well known as the Simultaneous Localization and Mapping (SLAM) problem. If, in contrast, a prior map of the environment is available, the goal is to continously localize the robot within this representation. This paper focusses on the latter problem motivating a novel approach to sparse map description and localization on this structure. An occupancy grid map is successively decomposed into basis primitives using non-negative matrix factorization. These can be understood as geometric features which are not determined a-priori but instead are automatically extracted from the given environment description. As a result a sparse description of the map containing only basis primitives and their distributions is obtained. In opposite to other approaches, we do not presuppose specific geometric features like lines or edges, as, for instance, presented in [1]. This enables optimal representations of different environment types while simultaneuously preserving a generic model. Tipaldi and Arras proposed an approach to region of interest (ROI) extraction from 2D range data [2]. Bosse and Zlot perform laser scan based map matching using local surface orientations within a SLAM framework [3]. Either of the methods [1–3] work on raw sensor data, specifically measurements from a laser range finder. Contrary to these methods, Schroeter et al. introduced Map Match SLAM [4, 5] utilizing occupancy grid maps which is similiar to our approach. This enables a more abstract representation of the underlying sensor data. Localization is then carried out in a Bayesian manner through association of locally observed feature distributions with those extracted a-priori from an occupancy grid map. Our approach enables efficient localization on highly compact environment representations. First experiments on a mobile robot navigating in a museum are carried out. In addition to that, the proposed approach is applied to publicly available datasets. First promising results are qualitatively depicted motivating further investigation in the application of NMF for localization and mapping.

2 Non-negative Matrix Factorization

Like other approaches, e. g. PCA and ICA, non-negative matrix factorization (NMF) [6] is meant to solve the source separation problem. Hence, a set of training data is decomposed into basis primitives $\mathbf{W}$ and their respective activations $\mathbf{H}$:

$$\mathbf{V} \approx \mathbf{W} \cdot \mathbf{H} \tag{1}$$

Each training data sample is represented as a column vector $\mathbf{V}_i$ within the matrix $\mathbf{V}$. Each column of the matrix $\mathbf{W}$ stands for one of the basis primitives. In matrix $\mathbf{H}$ the element H_i^j determines how the basis primitive $\mathbf{W}_j$ is activated to reconstruct training sample $\mathbf{V}_i$.

* This work was supported by ESF grand number 100076162

For generating the decomposition, optimization-based methods are used. Hence, an energy function E has to be defined:

$$E(\mathbf{W},\mathbf{H}) = \frac{1}{2} \|\mathbf{V} - \mathbf{T} \cdot \mathbf{W} \cdot \mathbf{H}\|^2 + \lambda \sum_{i,j} H_i^j \tag{2}$$

By minimizing the energy equation, it is now possible to achieve a reconstruction using the matrices $\mathbf{W}$ and $\mathbf{H}$. This reconstruction is aimed to be as close as possible to the training data $\mathbf{V}$.

Theoretically the basis primitives are invariant to several transformations as rotation, translation and scale. This is achieved by adding a transformation matrix $\mathbf{T}$ to the decomposition formulation [7]. For each allowed transformation the corresponding activity has to be trained individually. To avoid trivial or redundant solutions a further sparsity constraint is necessary. Its influence can be controlled using the parameter λ [8].

Enabling only translational invariance reduces the problem to a convolution over all translations. This is efficiently implemented by transforming the data into the frequency domain based on FFT. In order to reduce the complexity, rotation and scale are not taken into correspondence for the presented approach.

The minimization of the energy function can be done by gradient descent. The factors $\mathbf{H}$ and $\mathbf{W}$ are updated alternately with a variant of exponentiated gradient descent or using NMFs multiplicative update rule until convergence.

3 Localization and mapping using NMF

The data basis of our NMF-based approach to localization and mapping is given by occupancy grid maps. Introduced by Elfes [9] occupancy grid maps have demonstrated to be very advantageous in robot navigation. The entire environment structure is decomposed into grid cells with each expressing a likelihood of being occupied. They are well suited for a multitude of range measuring sensors, though our approach focusses on laser range finders. Occupancy grid maps can be interpreted as an abstraction of the actual sensor data. Instead of correlating the current scan with previously captured ones, a continuous representation associating adjacent scans is built. The majority of existing methods for grid map based localization matches entire laser scans against a prior occupancy grid map. The simplest model projects measured laser beams into the map tracing them until they hit an occupied cell (raycasting model) as detailed in [10]. Though ignoring some of the laser beams enormously reduces the complexity, this method is still expensive. The proposed approach, in contrast, does not consider each laser beam or grid cell itself, but introduces a further abstraction level working on geometric primitives. This enables efficient map matching while simultaneuously considering semantic environment structures.

The states expressed by the occupancy grid cells are converted into a binary representation differing only free and cells. This procedure does not significantly modify the actual states when the data is obtained from a laser range finder since distributions of occupancy likelihoods are trimodal possessing peaks around the states *occupied*, *free* and *unknown*. The occupancy likelihood p_i of grid cell i is transfered to the binary representation b_i as follows:

$$b_i = \begin{cases} occupied & (p_i > 0.5) \\ free & (p_i \leq 0.5) \end{cases} \tag{3}$$

Excluding the state *unknown* simply reduces the influence of noise arising in unexplored areas of the environment. Otherwise unknown areas would have to be modeled by the representation.

3.1 Training Phase

Given a binary occupancy grid map the training phase aims at gaining a set of basis primitives. The map is represented by the first column of matrix $\mathbf{V}$ and hence directly taken as input for the NMF approach. The standard NMF approach enriched by translation invariance and a sparsity constraint is applied on $\mathbf{V}$ as mentioned in Section 2.

3.2 Application Phase

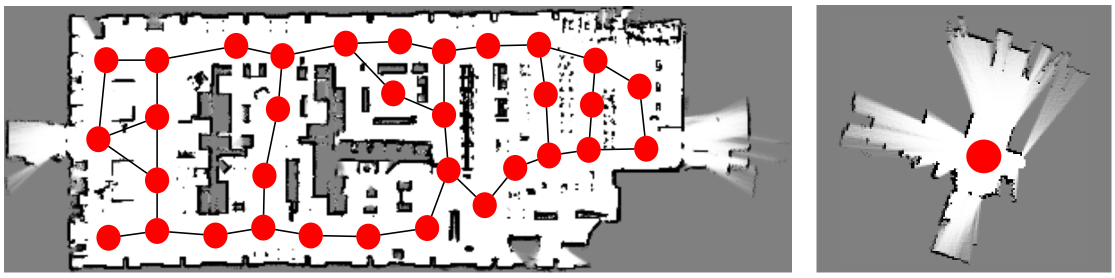

(a) Graph on global map (b) Submap

Fig. 1. Graph construction. The global map is partionioned into submaps. A graph connecting the submaps is constructed (Figure 1(a)). The red nodes of the graph denote the submaps which is exemplarily shown in Figure 1(b).

Similiar to previous SLAM approaches, as for instance [11, 12], we make use of a graph based representation. This step is not essential for NMF based localization and mapping, however, provides an initial model for further investigations. The global occupancy grid map is firstly partionioned into submaps. A graph m connecting all submaps is built ensuring the global consistency of the map as illustrated by Figure 1(a). Each submap m_i contains a sparse representation of the local distribution of NMF features. In particular, the activities of basis primitives within the submaps are encoded. The sizes of the submaps depend on their local feature diversities. Regions consisting of repetitive structures are stored more compactly summarizing the local features in one submap. For instance, environments like long corridors only consisting of line primitives can be described by a minimum number of NMF features which is exemplarily demonstrated by Figure 3.

Having completed a graphical model of the map, a particle filter[1] is used in order to perform global localization. The robot's state $X_t = (x_t, y_t, \phi_t)^T$ containing the 2D location (x_t, y_t) and orientation ϕ_t is predicted according to our motion model given odometry readings at timestep t. The NMF features z_t observed at timestep t are associated with those stored for each submap m_i using the observation model. More specifically, the observation likelihood $p(z_t|m_i, X_t)$ is estimated. The observation model raises the problem of transition from the sparse NMF-based representation to a local encoding. To be more precisely, the submap m_i has to describe the basis primitives efficiently.

The NMF based localization involves a number of spatial uncertainties that have to be incorporated. For instance, the local feature distributions change when submaps are perceived from different viewpoints. Robust position estimates require a certain level of rotation and viewpoint invariance of the NMF features. As mentioned in Section 2, a rotation invariance within NMF is generally possible, however requires additional computations. Hence we directly utilize the orientation space covered by the particles' states. In particular, the orientation $\phi_t^{(i)}$ of the particle i is used to rotate the local map captured by the robot. This enables global alignment of the local map used for localization with reference to the previously built map. The reconstruction accuracy serves as a measure of quality and thus poses the basis to estimate the weight w_i of particle i with $w_i \propto p(z_t|m, X_t^{(i)})$.

4 Future Work

The proposed approach is part of on-going research on non-negative matrix factorization with application to mobile robot localization and mapping. Based on current implementations we pre-

[1] A detailed introduction to particle filters with application to mobile robot localization is given by [10].

sented first experimental results obtained from multiple real-world datasets. These motivate further investigation promising substantial contributions to robust and efficient localization while significantly reducing the storage sizes of maps by exploiting semantic environment structures. In addition to that, in-depth research on estimating optimal numbers of basis primitives is intended. Currently these are determined a-priori depending on the environment stucture. Similarly to the approach presented in [13] we are investigating incremental learning of basis primitives.

A Experimental results

The training phase of the NMF given binary grid maps is carried out according to Section 3.1. The following figures exemplarily show priliminary results of the presented approach. The results contain the initial occupancy grid maps, the input binary grid maps and the NMF reconstructions which are followed by the basis primitives and activations of latter. The colors of the reconstructed maps are set according to an HSV color map with blue being the maximum and red the minimum values. Zero elements are plotted white emphasizing NMFs sparsity. The results of Figure 2 are obtained using datasets collected within the museum *Technical Exhibitions* in Dresden. Figures 3 and 4 show results achieved by applying the presented approach to datasets made publicly available through the Radish repository [14].

References

1. Wulf, O., Arras, K.O., Christensen, H.I., Wagner, B.: 2D mapping of cluttered indoor environments by means of 3D perception. In: Proc. IEEE International Conference on Robotics and Automation (ICRA'04), New Orleans, USA (2004)
2. Tipaldi, G., Arras, K.: Flirt - interest regions for 2d range data. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on. (2010) 3616 –3622
3. Bosse, M., Zlot, R.: Map matching and data association for large-scale two-dimensional laser scan-based slam. I. J. Robotic Res. **27**(6) (2008) 667–691
4. Schröter, C., Böhme, H.J., Gross, H.M.: Memory-efficient gridmaps in rao-blackwellized particle filters for slam using sonar range sensors. In: EMCR. (2007)
5. Schröter, C., Gross, H.M.: A sensor-independent approach to rbpf slam - map match slam applied to visual mapping. In: IROS. (2008) 2078–2083
6. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. Advances in Neural Information Processing **13** (2001) 556–562
7. Eggert, J., Wersing, H., Körner, E.: Transformation-invariant representation and NMF. In: IJCNN. (2004) 2535 – 2539
8. Eggert, J., Körner, E.: Sparse Coding and NMF. In: IJCNN. (2004) 2529 – 2533
9. Elfes, A.: Using Occupancy Grids for Mobile Robot Perception and Navigation. Computer **12**(6) (June 1989) 46–57
10. Thrun, S., Burgard, W., Fox, D.: Probabilistic robotics. Intelligent robotics and autonomous agents. MIT Press (2005)
11. Blanco, J.L., Fernandez-Madrigal, J.A., Gonzalez, J.: A new approach for large-scale localization and mapping: Hybrid metric-topological slam. In: Robotics and Automation, 2007 IEEE International Conference on. (2007) 2061 –2067
12. Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g2o: A general framework for graph optimization. In: ICRA, Shanghai (2011)
13. Rebhan, S., Sharif, W., Eggert, J.: Incremental learning in the non-negative matrix factorization. In: Advances in Neuro-Information Processing. Volume 5507. (2009) 960–969
14. Howard, A., Roy, N.: The robotics data set repository (radish) (2003)

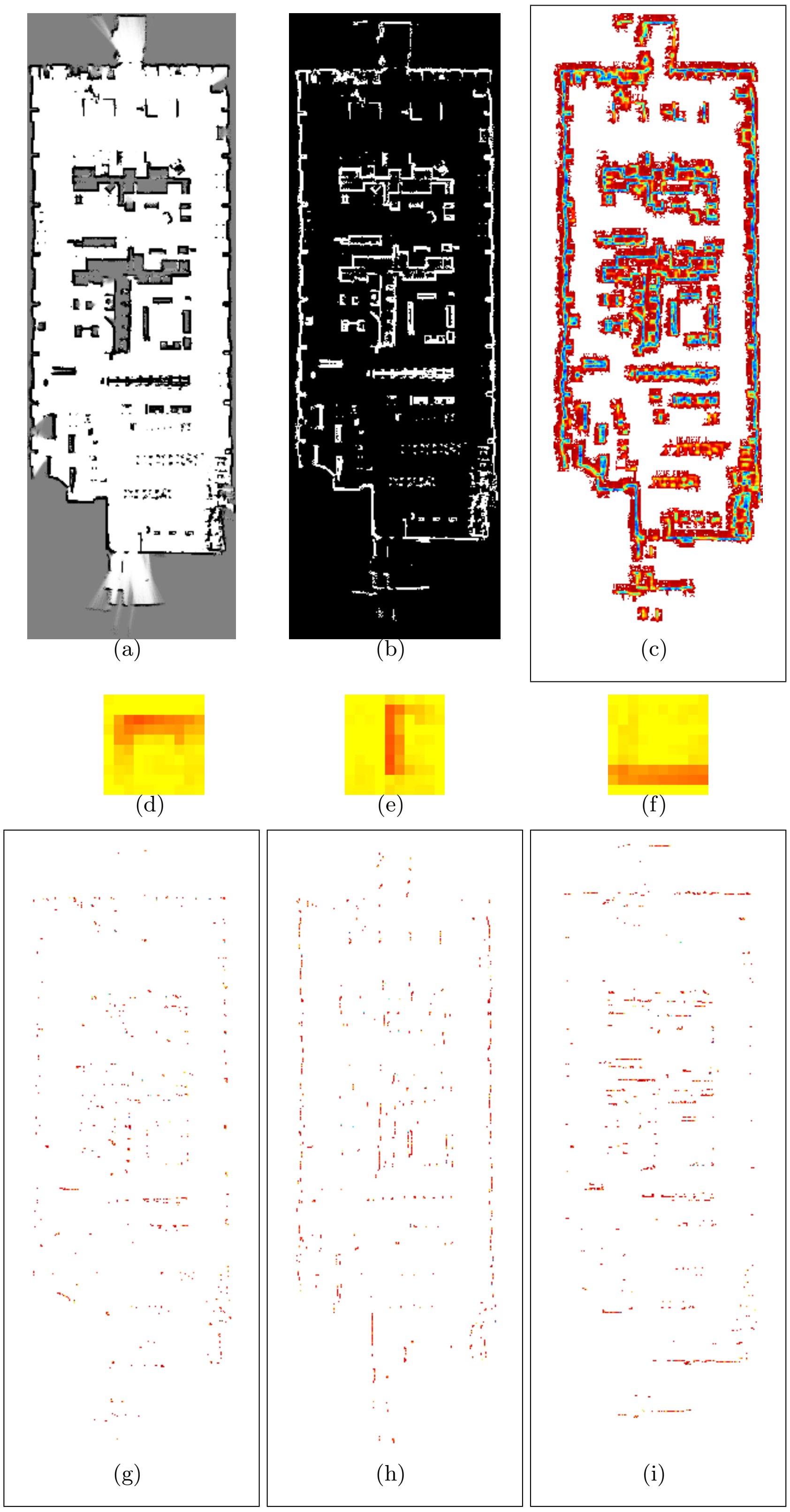


Fig. 2. Experiments within the Technical Exhibitions Dresden. Figure (a) shows the initial occupancy grid map, (b) the binary grid map, (c) the NMF reconstruction, (d) - (f) the basis primitives and (g) - (i) their activities.

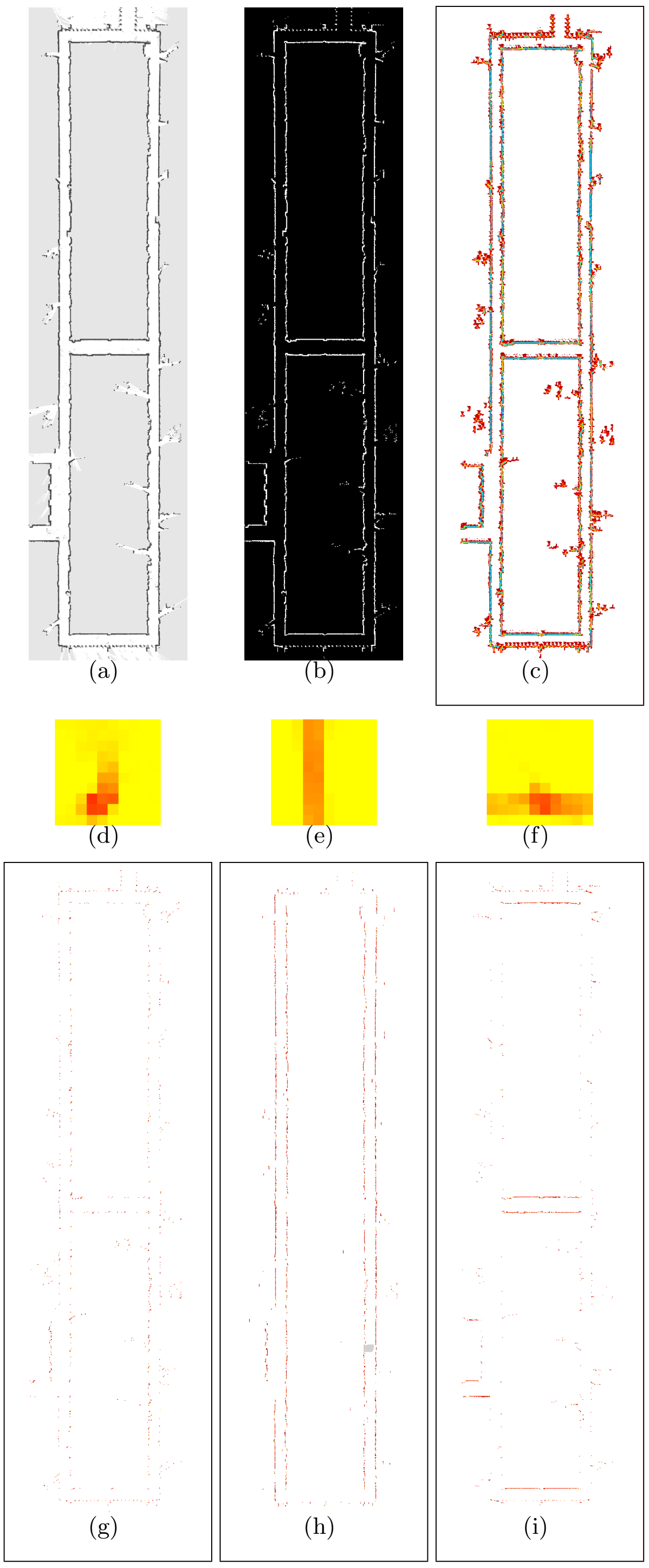


Fig. 3. Long corridor experiment based on a segment extracted from the MIT Killian Court dataset. Figure (a) shows the initial occupancy grid map, (b) the binary grid map, (c) the NMF reconstruction, (d) - (f) the basis primitives and (g) - (i) their activities.

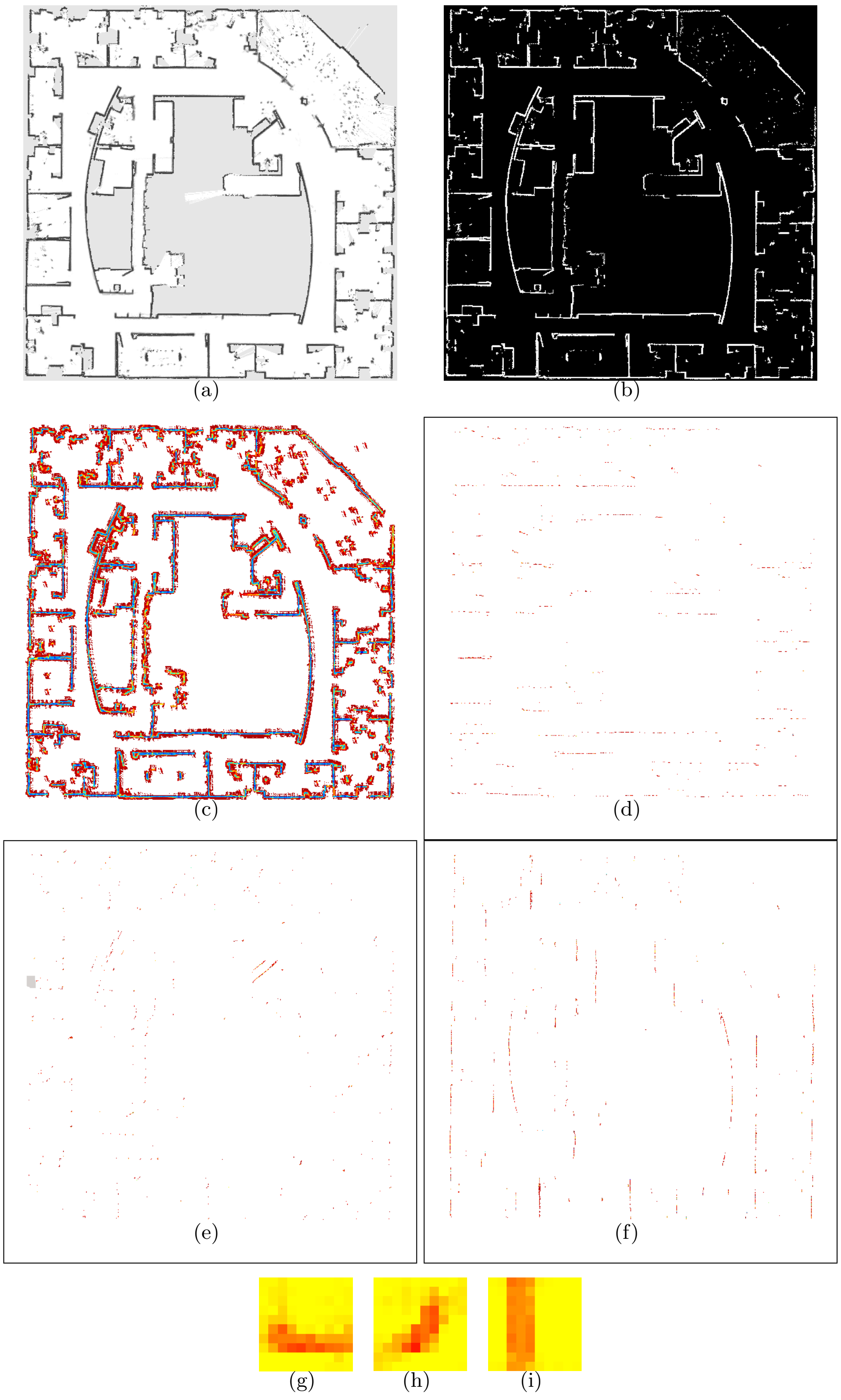


Fig. 4. Results obtained using the Intel Research Labs dataset. Figure (a) shows the initial occupancy grid map, (b) the binary grid map, (c) the NMF reconstruction, (g) - (i) the basis primitives and (d) - (f) their activities.

Learning of Invariant Object Recognition in a Hierarchical Network

Markus Lessmann and Rolf P. Würtz

Institut für Neuroinformatik, Ruhr-Universität, Bochum, Germany
{markus.lessmann,rolf.wuertz}@ini.rub.de

Abstract. In this paper we propose an object recognition system implementing three basic principles: forming of temporal groups of features, learning in a hierarchical structure and using feedback for predicting future input. It gives very good results on public available datasets. Precondition for successful learning is that training images are presented to the system in an appropriate order such that images of the same object under similar viewing conditions follow each other. The system has moderate memory demands and a very big fraction of computing resources (during recognition) is spent on nearest neighbor search in a codebook of visual features, which can be sped up using locality sensitive hashing methods [1].

1 Introduction

Visual processing is probably the most examined brain function in all fields of neuroscience. A huge amount of diverse and partly contradicting data about it has been gathered by myriads of studies in all fields of neuroscience. Since inductive reasoning alone is insufficient for getting a full understanding one has to look for general concepts that allow deductive conclusions. The lack of such theoretical concepts was criticized by Jeff Hawkins in his 2004 book "On Intelligence" [2] where he filled this gap by introducing a "Memory Prediction Framework". This collects a lot of known ideas about neural information processing in a coherent framework and has led to a software system named Hierarchical Temporal Memory (HTM), which implements most of the main concepts [3]. Three main ideas can be found in the HTM:

1. Learning of temporal sequences for creating invariance to transformations contained in the training data.
2. Learning in a hierarchical structure such that lower level knowledge can be reused in higher level contexts and thereby makes memory usage efficient.
3. Prediction of future signals for disambiguation of noisy input by usage of feedback.

In this article we will present a novel system which is able to do invariant object recognition implementing these basic ideas. The layout of the article is the following: in the next chapter we present our system in detail. Chapter 3 describes

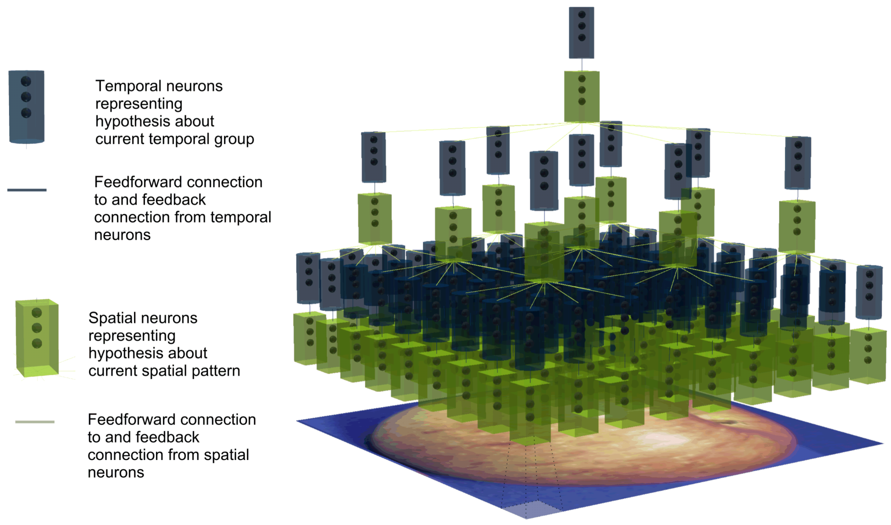

Temporal neurons representing hypothesis about current temporal group
Feedforward connection to and feedback connection from temporal neurons
Spatial neurons representing hypothesis about current spatial pattern
Feedforward connection to and feedback connection from spatial neurons

Fig. 1. Visualization of the network architecture. Connections of nodes represent possible synaptic connections between all neurons in one node and all in the other.

how learning works in the network. Results of experiments are presented in chapter 4. Chapter 5 closes the article with a conclusion and an outlook on future work.

2 Our System

Our system is an artificial neural network built up of neurons with associated activity values. Neurons have feedforward and feedback connections of differing strengths that are determined using a mostly unsupervised learning algorithm. Figure 1 shows the general structure of the network. It consists of 3 levels of node positions. Each position contains one node for neurons representing spatial patterns (spatial neurons) and one for neurons representing temporal groups (temporal neurons). A neuron can be seen as a hypothesis about which spatial or temporal pattern is observed at the current node position and time step. Each node is implemented as a hash map and stores for each active neuron its index and its activity. On the lowest level a spatial pattern is just a visual feature (a parquet graph [1]). On a higher level a spatial pattern is built up from temporal groups at the nodes that converge onto the current one (represented by their indices). A temporal group is a group of spatial patterns that appeared often at the same position close in time during learning. The temporal groups in the top level node represent the different object categories that have been learned.

The system learns from image sequences showing objects undergoing transformations in viewing conditions. It first builds a database (codebook) of features that are typical for these objects using vector quantization. Then it learns groups of features that occur at the same position when the objects transform. Groups at adjacent positions constitute the spatial patterns which are the input to the next level of the network. Again, a codebook is learned by vector quantization and then temporal groups are formed and the process is repeated until the top level of the hierarchy. This means the system has to be trained level by level.

3 Inference and Learning

The following chapter explains how the system does inference and how it is trained for doing this. Both steps are not independent since during training inference needs to be done on the already trained levels of the network.

3.1 Inference

Inference is done by computing activities of all neurons in our model for a given input image and then reading out the index of the most active temporal neuron at the top level. This neuron identifies the recognized object category. Calculation of activities is done from bottom to top level for one node position after another. When learning is finished this can also be done in parallel for node positions on the same level. Activities are calculated and stored in a temporary memory. If their calculation is completed they are transfered to a bigger container storing activities of the last T time steps, where they are used for learning and for feedback calculation.

Computing spatial feedforward input: First the spatial pattern at the current node position is extracted. This may either be a parquet graph on the corresponding image position for the lowest level or a concatenation of the indices of the most active temporal groups at adjacent node positions on the previous level. During learning the nearest neighbor in the codebook of the current level is determined. If the similarity to it is below a threshold S the pattern is added to the codebook. A neuron with the index of the new pattern is created and gets an activity value of 1. If the similarity to the nearest neighbor is above the threshold its corresponding neuron gets activated with the similarity as activity. How similarities of spatial patterns are defined is explained later. When learning on the actual level is completed no new patterns are added to the codebook and activities on higher levels can be calculated more neuron-like. Since all temporal groups have similarities to each other, a lower level temporal group can activate each possible spatial pattern of the current level using its similarity to the temporal group at its relative position in that pattern. Therefore, the activity of the temporal group is multiplied with this similarity divided by the number of temporal groups within a spatial pattern and this is added to the activity of the neuron representing that spatial pattern. This has the effect that a spatial pattern can be activated even if only one of its parts is observed (caused by, e.g., occlusion).

Inhibiting spatial neurons: After calculation of the feedforward input the amount of active neurons at the node position is reduced by setting the activity of all but the K most active neurons to zero and deleting them from the hash map. During learning K is 1, during recall it can also be higher. This step can be seen as application of inhibition between neurons at the same node position. Without it all neurons would be kept active, the network would run into a kind of overexcitation and the first most active neuron at the top level would remain the winner for all following images.

Computing spatial feedback input: The next step is to add feedback input to the remaining active neurons coming from temporal groups which have been active on the previous images.

Application of the activation function: The neuron activities are now processed by the activation function. The hyperbolic tangent was used for all experiments. The activation function prevents activity values from growing to infinity, which could happen because of feedback connections. It also provides one of the two nonlinearities in the system enabling it to robust classification (the other one being the inhibition of neurons).

Transfer into permanent memory: Then activities are written into the permanent memory of the last T images. This is done in a special way. If currently the same neurons are active as in the memory for time step 0 only their activities are updated. If a neuron has become inactive or a new one was activated all stored activities of the last time steps are shifted by one position (and the activity at time step $T-1$ is deleted) and current activity is copied to position 0. This has the effect that not only one neuron occupies all stored positions and is the only one giving feedback. Also the system is prevented from only recording transitions of one neuron to itself while learning temporal groups. During learning activity in the permanent memory is deleted when a new object category is presented. This prevents the system from learning temporal transitions between different categories.

Computations for temporal neurons: Now the same kind of calculations are done for neurons representing temporal groups. At first temporal neurons collect their feedforward input, then inhibition is applied and feedback is given to remaining temporal neurons. At last the activation function is used again and activities are moved to the permanent memory.

No inference possible: If learning data was too sparse or too few hypotheses are kept it may happen that no active temporal group can activate a spatial pattern on the next level because there doesn't exist any connection. Then no decision can be made about the category of the object on the current image. This problem diminishes with more learning data and more active hypotheses (higher K) during testing.

3.2 Learning

Learning on one level is done in two steps: first a codebook of input patterns is learned using vector quantization, then temporal groups are established. This is done on all positions of the network, but globally with only one codebook and one container for the groups per level. For establishing temporal groups training images are browsed after codebook learning and it is counted and stored in a global matrix $\underline{M}$ how often spatial patterns follow each other at the same node position within a certain time frame of T images. After normalization these counts can be seen as temporal similarities of spatial patterns. Now $\underline{M}$ is clustered using spectral clustering. On the lower levels groups of maximum size GS are formed, on the top level $\underline{M}$ is clustered in as many groups as there are categories in the training set. This is one of three supervised steps in learning. Another one is the deletion of the permanent memory between training images of different object categories. Of course there has to be a last supervised processing step, in which each temporal neuron at the top level is assigned the name or the number of the category that it indicates.

After clustering for each spatial pattern the sum of similarities to all spatial patterns in a cluster is computed (see figure 2 b)). This is a kind of membership measure, which is of course biggest for the cluster the pattern was put into. This gives potential connection weights from spatial neurons to temporal groups at the same level. The G strongest of these weights are used and normalized with the sum of all employed weights.

For doing vector quantization on higher spatial patterns similarities between them are defined as average similarities of their components (which are temporal groups). Thus a similarity measure for temporal groups must be devised. This can be reached using the temporal similarities stored in matrix $\underline{M}$. If one considers each similarity as an edge between two patterns and looks at two distinct clusters A and B there are 3 different sets of edges: edges within set A, edges within set B and edges that connect patterns of A with elements of B. This can be seen in figure 2 a). The sum of edge weights in the last set is the so called cut between A and B ($cut(A, B)$, the sum of the weights of all purple edges in figure 2 a)). The sum of edge weights in A (respectively B) is called $V_r(A)$ (resp. $V_r(B)$) because it is the volume of A restricted to the set. In figure 2 a) these are the sums of the weights of all blue respective all red edges. The cut is now divided by the sum of the restricted volumes of A and B and itself:

$$\mathrm{sim}(A, B) = \frac{\mathrm{cut}(A, B)}{\mathrm{cut}(A, B) + V_r(A) + V_r(B)} \tag{1}$$

This means that the similarity between sets A and B is divided by the similarities within the complete set of patterns. Using this measure spatial patterns on higher levels can be compared by taking the average similarity of temporal groups at corresponding positions. Thus codebooks on these levels can be learned using the same quantization threshold S as on the lowest level.

4 Experiments

The system was tested on the ETH80 [4] (in the “cropped close perimg” version), which contains images of 80 different objects belonging to 8 different categories (apple, car, cow, cup, dog, horse, pear, tomato), and the COIL100 [5], which

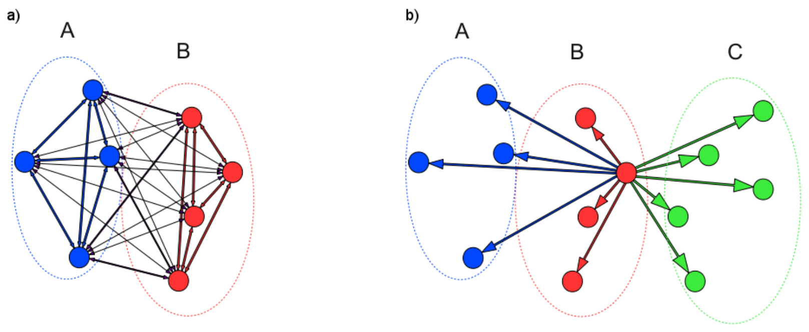


Fig. 2. a) Illustration of two clusters. Line widths indicate similarities of elements. Blue edges connect elements in A, red one elements in B and purple ones elements in A with elements in B. b) Scheme of three clusters. A membership value is computed for the central vertex using the drawn edges. Weights of all edges with the same color are added and divided by the sum of all edges irrespective of color.

consists of images of 100 different objects, each being its own category. In both databases a black background was used instead of additional segmentation information. All tests used a one-fold cross validation scheme: the number of views per object was split into two groups, the first set of views of every object was used for training, the rest for testing.

All images have a size of 128×128 pixels. A 3-level network was used as shown in figure 1 with 9×9 nodes on the lowest level, placed with a spacing of 14 pixels and an offset of 7 on the input images, 3×3 on the intermediate level and a single node on the top level. Since the network learns temporal sequences/groups the training images have to be in a meaningful order. Therefore views were sorted according to their great-circle distance on the viewing hemisphere. This can be computed using the two or one viewing angles given in the filename of each image. For fifty-fifty-partitioning every other view of the order was taken for training and the rest for testing, for other split-ups only every third or fourth and so forth.

The system has several parameters whose influence on recognition performance was tested in the following experiments. First the relevant parameters are listed again for a better overview:

K: number of neurons kept for computing the input to the next level/sublevel
G: number of temporal groups that are activated by a spatial neuron
T: number of past images whose activities are kept in memory
S: threshold used for vector quantization
GS: determines how big a temporal group can be at most

Several tests were conducted to find an appropriate set of parameter values for the ETH80, resulting in the following values: $S = 0.92$, $T = 13$, $K = 10$, $G = \infty$ (all possible connections are used) and $GS = 50$. These gave a recognition

Table 1. Tests for generalization over viewing angle. Given are the percentage of images of each data set the system is asked to use for training, the number of images that are actually used and the actual percentage of training images (which differs from the requested one because a fixed number of view points is used per object) and the recognition rate reached on the remaining images. On the left results for ETH80 on basic category level (apple, car, cow and so forth) and on the right results for COIL100 on name level.

% requested	# obtained	% obtained	RR	# obtained	% obtained	RR
	ETH80			COIL100		
2.50	160	4.88	48.14	200	2.78	59.04
5.00	240	7.32	18.45	400	5.56	78.15
10.00	400	12.20	21.28	800	11.11	91.02
20.00	720	21.95	88.48	1500	20.83	97.72
30.00	1040	31.71	95.00	2200	30.56	99.12
40.00	1360	41.46	93.54	2900	40.28	99.67
50.00	1680	51.22	97.69	3600	50.00	99.92
60.00	2000	60.98	97.27	4300	59.72	99.86
70.00	2320	70.73	96.67	5100	70.83	99.95
80.00	2640	80.49	95.31	5800	80.56	100.00
90.00	2960	90.24	91.56	6500	90.28	99.43

rate of 97.69%. For COIL100 the parameters were kept except K, which was set to 1, yielding a recognition rate of 99.92%.

The next test shows the generalization capabilities of the system for ETH80 and COIL100. The system was trained on (roughly) 10,20,30,40,50,60,70,80 and 90% of all images in the database and then recognition on the remaining images was done using the two optimal parameter sets.

The table above shows in the first column the percentage of all images of each database that the system was requested to use, in the second and fifth column the actual number of used images of ETH80 respectively COIL100, in the third and sixth the factual percentage of used images and in the fourth and last column the recognition rate achieved on the remaining images. The results in table 1 demonstrate the very good generalization capabilities of the network. Even with only 20% of the data recognition rates of almost 89% and 98% can be reached. As next experiment we conducted the standard test with both databases without feedback to demonstrate its beneficial impact and with feedback but using nearest neighbor search in the higher level codebooks for activation of spatial patterns instead of using the neural connections. The outcome for the ETH80 reveals the advantage of using both feedback and neural connections, since both improve recognition performance considerably. With codebook the recognition rate drops to 83.69% and without feedback to 83.62%. The possible positive effects of feedback (resolution of ambiguities) have been described before, the neural mechanism causes more spatial patterns to become active and offers the system more valuable hypotheses to chose from. For COIL100 both effects are

Table 2. Tests for generalization over viewpoints on COIL100 on name level.

viewpoint difference	proposed system	Westphal	Linde/Lindeberg
10°	99.92	99.68	**100.00**
20°	98.50	97.97	**99.96**
30°	96.15	92.93	**99.88**
40°	**93.14**	88.45	-
45°	91.02	-	**99.37**
50°	**88.25**	83.20	-
60°	84.42	76.61	**97.99**
70°	**78.57**	75.79	-
80°	**78.15**	72.39	-
90°	78.15	65.63	**97.13**

nearly negligible, recognition rates with codebook respectively without feedback are over 99%. Since for COIL100 K is set to 1 feedback doesn't have to disambiguate between several hypotheses. Using the codebook probably has a similar effect as using neural connections with $K = 1$.

Table 2 subsumes results for the generalization test on the COIL100 of our system and two others. For comparison we took the system of Westphal [6], which uses the same features as ours, and the system from [7], which gave the best results that we could find. It computes high dimensional histograms from images and classifies them using an SVM. The results shown here have been obtained using a SVM and 14-dimensional histograms. It has to be noted that results of Westphal have been obtained using 5-fold cross validation and not 1-fold cross validation as for both other systems. The first column shows the distance in viewing angle of two consecutive images in training or test set. The second through forth columns show the obtained recognition rates. The comparison reveals that our system outperforms the approach of Westphal. Nevertheless it cannot compete with the system of Linde and Lindeberg for very sparse training sets. Whereas they reach recognition rates of over 97% for 90° distance between training images our system drops to 78.15%. However, it needs to be considered that they include color information, which is not used in our system.

At last leave-one-out cross-validation was performed on the "normal" ETH80 using $S = 0.94$ and $K = 18$, the remaining parameters remained unchanged. Results are in table 3. The results from Leibe and Schiele [4] are the best ones using texture features. What can be seen is that our system is the best in all but one category and shows the best overall performance.

5 Conclusion and Future Work

We have presented a powerful object recognition system that generalizes very well over different views of the same object and also (but not as good) over different identities of the same category on standard test data sets. A major goal of future work is to let the system learn connection weights using a biologically more

Table 3. Leave-one-out-cross-validation on ETH80 on basic category level.

category	prop. Sys.	Westphal	Leibe/Schiele
apple	**94.15**	91.22	80.24
car	**99.76**	80.98	77.56
cow	80.98	49.76	**94.39**
cup	**100.00**	98.05	77.80
dog	**82.20**	35.85	74.39
horse	**80.98**	57.80	70.98
pear	**91.71**	87.80	85.37
tomato	**98.78**	95.12	97.07
complete	**91.07**	74.56	82.23

plausible local learning rule. The integration of horizontal connections within a layer is also a possible enhancement of the model which is tested currently.

Acknowledgments

The authors gratefully acknowledge funding from the DFG in the priority program "Organic Computing" (WU 314/5-3) and from the land of Northrhine-Westphalia in the project *MoGES*, which is co-financed by the EFRE program from the European Commission .

References

1. Lessmann, M., Würtz, R.P.: Fast nearest neighbor search in pseudosemimetric spaces. In: Proc. VISAPP. (2012) 667–674
2. Hawkins, J., Blakeslee, S.: On Intelligence. Times Books (October 2004)
3. George, D.: How the brain might work: a hierarchical and temporal model for learning and recognition. PhD thesis, Stanford University (2008) AAI3313576.
4. Leibe, B., Schiele, B.: Analyzing appearance and contour based methods for object categorization. In: CVPR (2). (2003) 409–415
5. Nene, S.A., Nayar, S.K., Murase, H.: Columbia Object Image Library (COIL-100). Technical report, Department of Computer Science, Columbia University (Feb 1996)
6. Westphal, G., Würtz, R.P.: Combining feature- and correspondence-based methods for visual object recognition. Neural Computation **21**(7) (2009) 1952–1989
7. Linde, O., Lindeberg, T.: Object recognition using composed receptive field histograms of higher dimensionality. In: Proceedings of ICPR. ICPR '04, Washington, DC, USA, IEEE Computer Society (2004) 1–6

Experience in Training (Deep) Multi-Layer Perceptrons to Classify Digits

Jens Hocke, Thomas Martinetz

Institute for Neuro- and Bioinformatics, University of Lübeck

1 Introduction

Multi-Layer Perceptrons (MLPs) have been in use for decades. It seemed for a long time, that MLPs have reached their limits, but recent advances caught our attention. Ciresan et al. [1] show that by using a proper training set deep MLPs can outperform all other state of the art machine learning algorithms on the MNIST dataset for handwritten digits. However, a vast amount of training samples is needed, which has to be generated artificially with special transformations. The drawback is that appropriate transformations might be known for handwritten digits but not in general. Work by Hinton et al. and Bengio et al. [2, 3] suggest that unsupervised pre-training helps to find deep MLPs with better generalization from the training set, and thus avoiding to generate extra training data. We are interested in how the better generalization is archived and tested therefore alternative similar architectures. Here we present some observations we made in our first tests on the MNIST dataset. There is a lot of room for improvements to reach the error rates of Ciresan's approach.

2 Training of a Multi-Layer Perceptron

The standard approach to training a MLP is gradient descent on a cost function. To archive a faster convergence than with batch learning on a data set like MNIST with many repeating patterns, usually stochastic gradient descent is used. The most common cost functions are Mean Squared Error (MSE) and Cross-Entropy (CE). For the results shown below we used CE because it converged faster in our experiments. Applying it to a fully connected single layer network of 10 output units (one for each class) we get a test error of about 8.08 percent. By adding a hidden layer with 1000 fully connected units the error rate decreases to 1.72 percent.

Already the above network with 1000 hidden units has many more weights than training samples and reaches zero percent training error. It is an underdetermined system with many solutions having zero percent training error. By adding another layer, the classifier becomes even more powerful and the solution space for zero training error becomes even larger. There is no reason to expect a better generalization. Our experiments confirm this. A network with two hidden layers of 1000 and 500 units performs even worse on the test set (1.82 %). Ciresan et al. [1] benefited from an increased number of layers, but they circumvented the underdeterminacy by artificially creating a basically infinitely large training set by applying elastic deformations to the original data. But to generate this data it must be known which transformations are appropriate for the dataset.

Is it possible to make use of the additional power of the classifier without generating extra data? Hinton and Bengio [2, 3] suggest the use of autoencoders. These ensure that the information loss in every layer is minimal. After this kind of unsupervised training the entire network is retrained in the usual way using back-propagation. Interestingly, this yields better generalization than plain back-propagation without pretraining. It is not clear what the autoencoder does. If the hidden layer is smaller than the previous layer, an encoding similar to PCA is found. But for a larger hidden layer, the problem is again underdetermined. The simplest solution would be the identity (Direct connection of input and output). However, in practice this is not the solution found. Bengio et al. hypothesize this may be caused by a weight-decay they used preventing large weights, or stochastic gradient descent finding an arbitrary solution. The weights found by the autoencoder are usually only used as initialization, as the starting point for the back propagation learning of large (deep) underdetermined networks. It seems, that this choice of the starting point lets the network converge to a good solution in the solution space. Erhan et al. [4] hypothesize that pretraining is a regularizer with an infinite penalty on certain regions of the parameter space. Would it not be better to use autoencoders explicitly as a regularization for the network? This strategy can be motivated by the fact that the brain is not only performing one specific classification task with its visual input, but many different ones. Then for every task different features are used, thus almost all features need to be encoded in the hidden layer, which is enforced by the autoencoder. We have tested this approach, but did not archive a good generalization (1.82 %).

Bengio et al. use a layer-wise training for the autoencoder. Would it also work to add iteratively one hidden layer and train only the newly added layer and the output neurons to classify correctly? This approach should ensure that all information needed for the classification is passed from the lower layers to the output layer. However, our experiments show that this scenario does not lead to better results. On the test set the performance is just as good as or even worse than using only one hidden layer (1.73 %). By retraining the entire network using the previously found weights as initialization leads to a slight improvement (1.70 %), but still worse than the autoencoder result. It is interesting to note that in all cases the features of the first hidden layer resemble parts of digits, if the networks are trained with noisy samples.

To summarize, so far the effect of pretraining deep MLPs with autoencoders is not yet really understood, but also does not really lead to competitive results on the MNIST dataset.

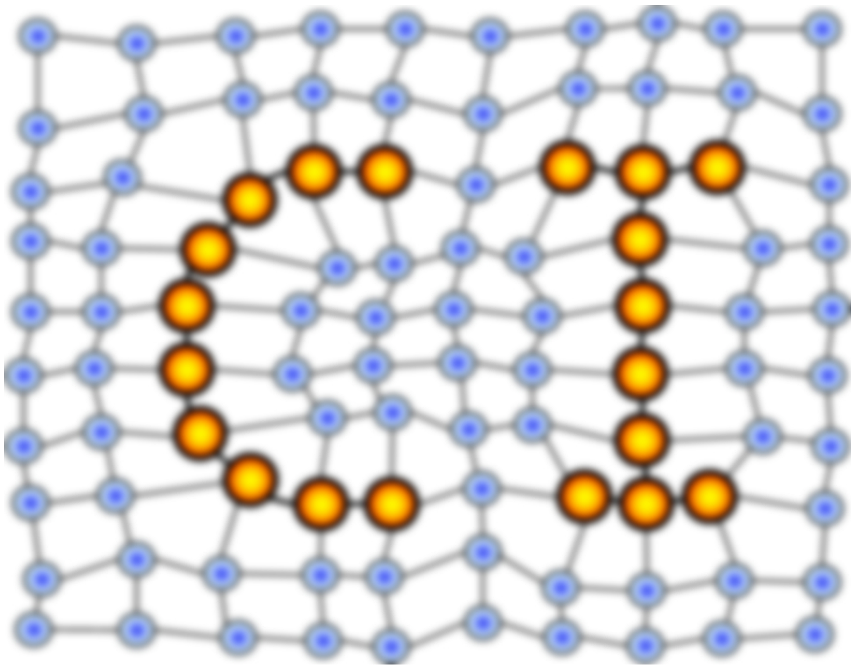
References

1. Ciresan, D., Meier, U., Gambardella, L., Schmidhuber, J.: Deep, big, simple neural nets for handwritten digit recognition. Neural computation **22**(12) (2010) 3207–3220

2. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science **313**(5786) (2006) 504–507
3. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. Advances in neural information processing systems **19** (2007) 153
4. Erhan, D., Manzagol, P., Bengio, Y., Bengio, S., Vincent, P.: The difficulty of training deep architectures and the effect of unsupervised pre-training. Artificial Intelligence **5** (2009) 153–160

MACHINE LEARNING REPORTS

Report 03/2012



Impressum

Machine Learning Reports ISSN: 1865-3960

▽ Publisher/Editors

Prof. Dr. rer. nat. Thomas Villmann
University of Applied Sciences Mittweida
Technikumplatz 17, 09648 Mittweida, Germany
• http://www.mni.hs-mittweida.de/

Dr. rer. nat. Frank-Michael Schleif
University of Bielefeld
Universitätsstrasse 21-23, 33615 Bielefeld, Germany
• http://www.cit-ec.de/tcs/about

▽ Copyright & Licence

Copyright of the articles remains to the authors.

▽ Acknowledgments

We would like to thank the reviewers for their time and patience.