ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 189-194

A Neural Network Approach to Adaptive Pattern Analysis — the Deformable Feature Map *)

Axel Wismüller[1], Frank Vietze[1],
Dominik R. Dersch[2], Klaus Hahn[1],
and Helge Ritter[3]

[1]Institut für Radiologische Diagnostik,
Ludwig-Maximilians-Universität München,
Klinikum Innenstadt, Ziemssenstr. 1, D-80336 München, Germany
email: Axel.Wismueller@physik.uni-muenchen.de
[2]Integral Energy Corp., Sydney, Australia
[3]AG Neuroinformatik, Universität Bielefeld, Germany

Abstract. In this paper, we present an algorithm that provides adaptive plasticity in function approximation problems: the deformable (feature) map (DM) algorithm. The DM approach reduces a class of similar function approximation problems to the explicit supervised one-shot training of a *single* data set. This is followed by a subsequent, appropriate similarity transformation which is based on a self-organized deformation of the underlying multidimensional probability distributions. After discussing the theory of the DM algorithm, we use a computer simulation to visualize its effects on a two-dimensional toy example. Finally, we present results of its application to the real-world problem of fully automatic voxel-based multispectral image segmentation, employing magnetic resonance data sets of the human brain.

1. Introduction

Function approximation is a classical problem of neural network computation. Various algorithms have been proposed to solve this problem, e.g. multi-layer-perceptrons trained by the error-back-propagation algorithm [7] or (generalized) radial-basis-functions networks ((G)RBF networks, see e.g. [2],[5], [1]). These algorithms are based on the supervised training of a sample data set by adapting the neural network parameters in order to represent an appropriate model of the target function. The (G)RBF approach decouples the function approximation problem into two different computational steps: an initial unsupervised vector quantization (VQ) step is followed by a supervised training of the output weights.

In this paper, we refer to the problem of training a *changing* target function. For instance, the target function may represent a dynamical system in a changing environment involving an inevitable temporal shift of parameters. A different example are apparent similarities within pattern analysis problems when comparing different, but similar objects. In biomedical research data sets, this phenomenon can be observed frequently (see e.g. [9]). One may think of the interindividual variability of anatomical features: there are no completely identical biological individuals, but there may be obvious anatomical "resemblances" (see e.g. fig.3.a,b).

These examples imply the need for adaptive plasticity in order to avoid a complete re-training of the function approximation network. Within the framework of (G)RBF function approximation, it is usually the *supervised* training of the output weights which is kept flexible in order to meet the needs of learning a changing target function, whereas the parameters obtained in the initial VQ procedure are preserved. For example, this approach is frequently chosen in the so-called mixture-of-experts solution of time-series prediction by competing RBF networks (see e.g. [3]). This is motivated by the observation that the VQ step is computationally more expensive than the adaptive training of the output weights. However, there may be situations in which repetitive supervised training is a critical issue, as an appropriate training data set (i) may be expensive, e.g. require human working power, (ii) may not be available at all.

In this paper, we present an algorithm that provides a reverse, alternative approach to adaptive function approximation: The output weights of a (G)RBF network are kept constant, whereas the adaptive training is performed on the VQ level. Hereby, the explicit supervised training is restricted to a *single* data set. From a theoretical point of view, this approach reduces a class of "similar" function approximation problems to the one-shot training of a single data set, followed by an appropriate subsequent similarity transformation.

2. Theory

Given are two similar, but not identical data distributions in the n-dimensional feature spaces X and Y. Here, the total number of raw data vectors may differ between X and Y, i.e. "similarity" refers to probability densities. Let $\mathbf{x}^\mu \in X\,(\mu \in \{1,\ldots,q\})$ denote the so-called *source distribution*, and $\mathbf{y}^\nu \in Y\,(\nu \in \{1,\ldots,p\})$ the *target distribution*. Given this situation, the basic problem in this article can be addressed as follows: How can X and Y be matched onto each other in a somewhat optimal manner, including local nonlinear deformations.

In other words, how can we define a mapping $S : X \rightarrow Y$ that satisfies the following constraints: (i) optimal correspondence of probability densities f and f' before and after the match, i.e. minimization of $\int_X \|f'(S(\mathbf{x})) - f(\mathbf{x})\|\, d^n x$, where $\|\cdot\|$ denotes an appropriate norm in $\mathbb{R}^n$, e.g. the Euclidean norm, (ii) minimization of the total deformation $\int_X \|S(\mathbf{x}) - \mathbf{x}\|\, d^n x$, and (iii) topology perservation, i.e. neighboring points of the source distribution in X should be mapped on neighboring points of the target distribution in Y. There is no unique, optimal solution to this tough optimization problem, as the constraints may be weighted differently. In the following, we present an algorithm that can at least provide suboptimal solutions.

The target distribution in Y can be represented by a set C_Y of prototypical "codebook vectors" $\mathbf{r}_j$, i.e. $C_Y = \{\mathbf{r}_j \in \mathbb{R}^n \,|\, j \in \{1,\ldots,N\}\}$ as a result of a suitable VQ procedure, e.g. Kohonen's self-organizing map (SOM) algorithm [4] or minimal free energy VQ [6], [1] etc.

The basic idea of the DM algorithm is the slight adaption of the original codebook vector positions $\mathbf{r}_j \in C_Y$ of the *target space* Y by re-training the codebook vectors with the data points of the *source space* X. This procedure

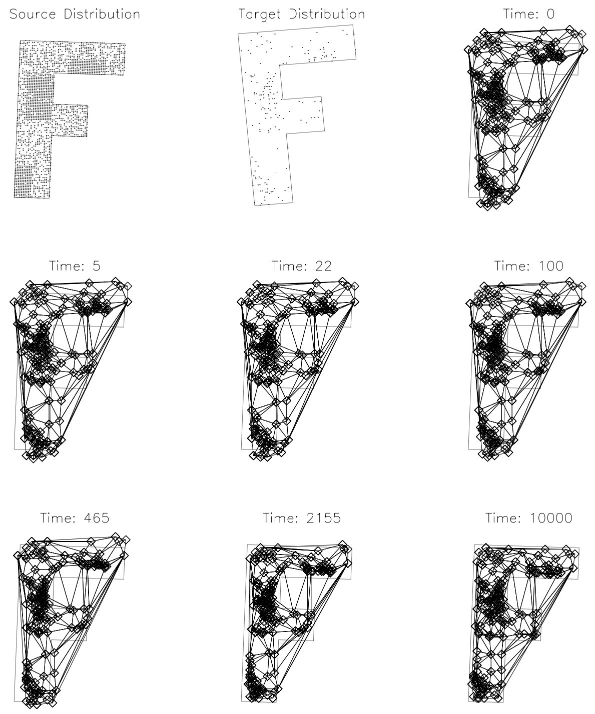

Figure 1: Application of the DM algorithm to a toy example. The tw odistributions are similar, but not identical. Here, they differ with respect to size and rotation. Note the gradual deformation of the target distribution onto the source distribution with increasing number of iterations. The lines represent a triangulation of the original target distribution. The absence of line crossings during the procedure can serve as an indicator for topology preservation without "twisting".

results in a new correspondingcodebook $C_X = \{\mathbf{w}_j \in \mathbb{R}^n \,|\, j \in \{1, \ldots, N\}\}$ representing the source distribution in X.

In detail, the desired codebook vectors $\mathbf{w}_j$ of the source space X are initialized with the codebook vectors $\mathbf{r}_j$ of the target space Y. Subsequently, the codebook vector positions $\mathbf{w}_j$ are adapted in an iterative procedure: After randomly choosing a data vector $\mathbf{x} \in X$, the codebook vectors $\mathbf{w}_j$ are updated according to

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \epsilon(t)\, h_j(\mathbf{x}(t), \sigma(t))\, (\mathbf{x}(t) - \mathbf{w}_j(t)), \tag{1}$$

employing the cooperation function

$$h_j(\mathbf{x}(t), \sigma(t)) = \exp\left(-\frac{(\mathbf{r}_j - \mathbf{r}_{\max}(\mathbf{x}(t)))^2}{2\sigma^2(t)}\right). \tag{2}$$

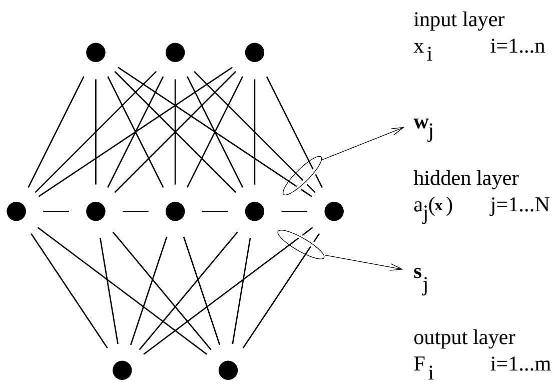

Figure 2: Architecture of a three-la yer(generalized) radial-basis-functions (RBF) netw ork.

and an appropriate (e.g. exponential) annealing scheme of the learning parameter $\epsilon(t)$ and the cooperation length $\sigma(t)$ for every training step t. The codebook v ector $\mathbf{r}_{max}(\mathbf{x}(t))$ represents the "winner neuron" with respect to the minimal distance to the presented data vector $\mathbf{x}(t)$ in the feature space X. It should be emphasized that the cooperation function $h_j(\mathbf{x}(t), \sigma(t))$ is based on the *metric of the target spac e* Y, whereas the update of the codebook vectors according to (1) occurs in the source space X! The positions of the $\mathbf{r}_j$ in the target space Y remain unchanged.

The DM algorithm, as described so far, shows close similarities to Kohonen's SOMs. How ev er, there are tw o *important differences* to the conv en tional use of SOMs: (i) The vectors $\mathbf{r}_j$ are not located on a (usually tw o-dimensional) regular grid. Their spatial positions are "meaningful" in the sense that they represent a codebook of the target distribution. (ii) There is no *random* initialization of the training procedure: the adaptive training of the codebook vectors $\mathbf{w}_j$ starts at the codebook vector positions $\mathbf{r}_j$ of the target distribution. Fig.1 shows an application of the DM algorithm to the matching of simple tw o-dimensional data sets.

The iterativ e training according to the update rule (1) results in a set of pairs $(\mathbf{w}_j, \mathbf{r}_j)$ of corresponding v ectors, representing reference points for the definition of a mapping $S : X \to Y, \mathbf{x} \mapsto \mathbf{y}$. Betw een these reference points, S has to be determined by interpolation. An elegant way to perform this task is the use of parametrized self-organizing maps (PSOMs) [8]

How can the DM algorithm be used for adaptive *sup ervise d* learning? Let $\mathcal{F}$ denote a function defined on the target space Y, i.e. $\mathcal{F} : \mathbb{R}^n \supset Y \to \mathbb{R}^m, \mathbf{y} \mapsto \mathcal{F}(\mathbf{y}), m, n \in \mathbb{N}$. In a (G)RBF scenario (Fig.2), the codebook vectors $\mathbf{r}_j$ can be interpreted as the input weigh ts of the hidden layer. The output weights s_{ij} can be trained in a supervised manner, employing a simple perceptron learning rule. The final result is a function approximator for $\mathcal{F}$.

Now, the goal is to train a netw ork in order to represent a function $\mathcal{F}' : \mathbb{R}^n \supset X \to \mathbb{R}^m, \mathbf{x} \mapsto \mathcal{F}'(\mathbf{x})$ with $\mathcal{F}(\mathbf{y}) = \mathcal{F}'(\mathbf{x})$ for pairs $(\mathbf{x}, \mathbf{y})$ of corresponding points of the source and the target space.

The central idea to solve this problem is to use the mapping S as trained b y the DM algorithm for the definition of "corresponding" points. Thus, a

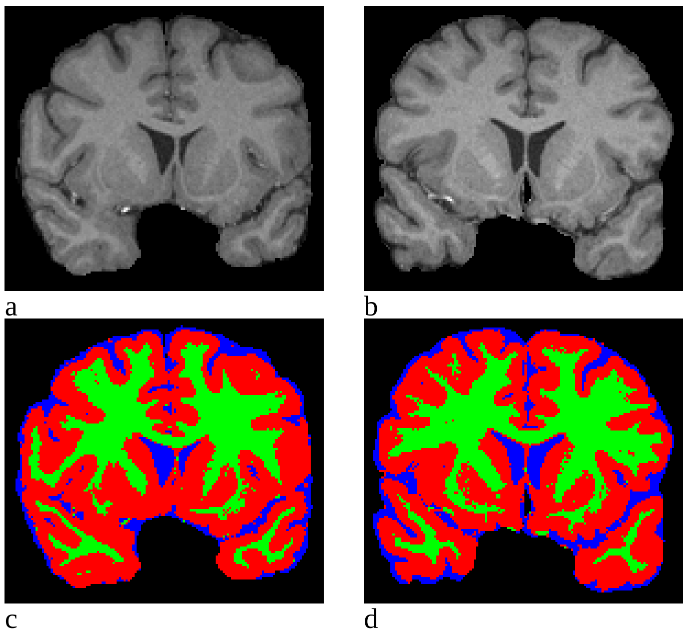

Figure 3: Results of fully automatic segmentation of multispectral magnetic resonance (MR) imaging data sets of the human brain using the DM approach. The upper line (a,b) shows so-called T1-weighted MR images. The lower line (c,d) shows the corresponding segmentations with respect to three classes "white matter" (light gray), "gray matter" (middle gray), and "liquor" (dark gray). The images of the left column (a,c) belong to an individual Y, the images of the right column (b,d) belong to a different individual X. The segmentation of Y (c) serverd as a reference data set for a fully automatic segmentation of X, shown in (d).

function approximator for $\mathcal{F}'$ can be trained in an *unsupervised* manner, just by exploiting the similarity between source and target distributions.

After completing the DM training of the mapping S, the information of the preceding supervised learning of $\mathcal{F}$ for a *single* target data set in Y can be employed in order to solve the function approximation problem for $\mathcal{F}'$. Given an arbitrary point $\mathbf{x} \in X$, this can be performed by the following computational steps: (i) Calculate $S(\mathbf{x}) \in Y$ as described above. (ii) Calculate the activations a_j of the codebook vectors $\mathbf{r}_j$ using the metric of Y. (iii) Calculate the output activations of the (G)RBF network using the output weights s_{ij} which have been determined by supervised learning of a single data set in Y.

3. Application to image segmentation

Fig.3 shows results of multispectral image segmentation employing the DM algorithm. This is an interesting problem in order to demonstrate its performance, as the creation of training data for supervised learning of image segmentation is a very time-consuming task that requires a considerable amount of human working power. The details of this application will be described elsewhere.

Fig.3a shows a coronal cross-section of a human brain obtained by magnetic resonance (MR) imaging of an individual Y. By changing several physical MR imaging parameters, k different images of the same cross-section can be obtained. By anatomically correct registration, these images form a so-called "multispectral" data set. Hereby, each pixel i can be characterized by a feature vector $\mathbf{y} = (g_1, \ldots, g_k, x_i, y_i)$ with $n = k + 2$, where g_j, $j \in \{1, \ldots, k\}$ denote the gray values of the different images, and x_i, y_i the spatial coordinates of the pixel. Thus, the data set can be described as a distribution in a n-dimensional feature space. Fig.3 refers to a data set with $n = 6$, i.e. $k = 4$. The supervised training of a GRBF classifier on this data set resulted in the image segmentation of fig.3c. Fig.3b shows a corresponding brain section of a *different* individual X. Note the differences between X and Y with respect to anatomical details and distribution of gray values. The DM algorithm provided a *fully automatic* image segmentation for data set X which can be seen in fig.3d, where Y served as a reference data set.

*) A similar paper has been presented at ICANN'98.

References

[1] D.R. Dersch. *Eigenschaften neuronaler Vektorquantisierer und ihre Anwendung in der Sprachverarbeitung.* Verlag Harri Deutsch, Reihe Physik, Bd. 54, Thun, Frankfurt am Main, 1996. ISBN 3-8171-1492-3.

[2] F. Girosi and T. Poggio. Networks and the best approximation property. *Biological Cybernetics*, 63:169–176, 1990.

[3] J. Kohlmorgen, K.R. Müller, and K. Pawelzik. Improving short-term prediction with competing experts. In *Proceedings of the International Conference on Artificial Neural Networks ICANN*, volume 2, pages 215–220, Paris, 1995. EC2 & Cie.

[4] T. Kohonen. *Self-Organization and Associative Memory.* Springer, Berlin, 1989.

[5] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294, 1989.

[6] K. Rose, E. Gurewitz, and G.C. Fox. Vector quantization by deterministic annealing. *IEEE Transactions on Information Theory*, 38(4):1249–1257, 1992.

[7] D.E. Rumelhart and J.L. McClelland. Learning internal representations by error propagation. In *Parallel Distributed Processing* volume I. M.I.T. Press, Cambridge, MA, 1986.

[8] J. Walter and H. Ritter. Investment learning with hierarchical PSOM. In A. Wismüller and D.R. Dersch, editors, *Symposion über biologische Informationsverarbeitung und Neuronale Netze – SINN '95, Konferenzband*, pages 161–168. Hanns-Seidel-Stiftung, München, 1996.

[9] A. Wismüller and D.R. Dersch. Neural network computation in biomedical research: chances for conceptual cross-fertilization. *Theory in Bioscienes*, 116(3), 1997.