To appear in: O. Simula (ed.) *Proceedings of the International Conference on ARTIFICIAL NEURAL NETWORKS, ICANN-91*, Elsevier Science Publishers, June 1991

LEARNING BY ERROR-DRIVEN DECOMPOSITION

Dieter Fox † Volker Heinze † Knut Möller†
Sebastian Thrun†‡ Gerd Veenker †

†Computer Science Department
Bonn University, Römerstr. 164
D 5300 Bonn 1, FR GERMANY

‡German National Research Center
for Computer Science, Postfach 1240
D 5205 St. Augustin, FR GERMANY

In this paper we describe a new selforganizing decomposition technique for learning high-dimensional mappings. Problem decomposition is performed in an error-driven manner, such that the resulting subtasks (patches) are equally well approximated. Our method combines an unsupervised learning scheme (Feature Maps [Koh84]) with a nonlinear approximator (Backpropagation [RHW86]). The resulting learning system is more stable and effective in changing environments than plain backpropagation and much more powerful than extended feature maps as proposed by [RS88, RMS89]. Extensions of our method give rise to active exploration strategies for autonomous agents facing unknown environments.
The appropriateness of our general purpose method will be demonstrated with an example from mathematical function approximation.

1 Introduction

Artificial neural networks have been found useful in widespread applications. One of the important problems concerning real-world applications is the slowness and instability of existing learning algorithms such as backpropagation [RHW86]. This slowness (which increases drastically with network size) is due to the fact, that

- all of the weights in the network are changed at once. Hidden units interact and are constantly facing changing information that delays the settling into useful hidden unit representations.
- the network architecture (number of hidden units, connectivity etc.) and parameter settings (e.g. learning rate, momentum) have to fit the problem, i.e. usually learning has to be repeated several times to find an appropriate network and parameter set.

Some approaches have already been proposed to eliminate the need for repeated trials with different architectures, such as the theoretical results on minimal hidden layer size [BH89] and constructive/destructive algorithms [FL90, Cha89].
In this paper we propose a radically different approach. Instead of constructing an architecture that fits a problem or function, we are decomposing the problem by splitting its input domain. Subdivision of input space is accomplished by an extended feature map. There is a small backpropagation (bp) network[1] associated with every feature map unit. Each bp-network approximates a region of the function, that is defined over the receptive field of its feature map unit. Instead of using equally defined patches or a decomposition that is just dependent on the probability distribution of the input, we describe a new difficultness-sensitive decomposition method. Local approximation error of the bp-networks drives the distribution of feature map units. Thus an error-driven decomposition of the input domain is created. Difficult parts of the function yield fine grained patches (more bp-nets), while easier ones lead to a coarse grained decomposition and are approximated by less bp-networks.
Our approach can be seen as an application of the devide and conquer-principle (or problem decomposition) that has a long tradition in computer science [HS78] and artificial neural network learning [JJB90, MT90]. Its usefulness is widely accepted (e.g. finite element method [BP84] in numerical mathematics, patching [Far90] in geometric modeling). But traditional methods suffer from the fact that either the decomposition has to be given in advance or is performed in a strict, predetermined manner (e.g. splitting in halves). We introduce an error-driven technique to automatically derive a suitable task decomposition. Some considerable benefits will be demonstrated:

[1] Actually just hidden units and their associated weight vectors are manipulated. Input and output units need not be duplicated.

- The learning process compared to backpropagation is sped up by using many small networks in a modular fashion rather than a single large one.
- Execution speed is enhanced by modularization, and it is possible to perform faster (gradient directed) search in input space, as it is used by a planning method described in[TML90, TML91].
- The problem of determining the number of hidden units is circumvented. The system performs an adaptive resource allocation in a way such that "difficult" parts of a function attract more networks (or hidden units) than easier ones. Destructive/constructive extensions can be incorporated easily.
- The instability (or relearning) problem is eased considerably, because additional patterns or slight changes of some old patterns only affect a small part of the whole network. Thus, their unstablizing influence is locally restricted to a small number of hidden units.
- An active, error-driven exploration strategy can be derived with our method, if adaption to an unknown environment is desired.

We will proceede by introducing our novel learning technique and its architecture.

2 Function approximation as a learning process

In the sequel we will compare several connectionist approximation methods, whose approximation capabilities will be demonstrated by the mexican hat function $f(x,y) = (4(x^2+y^2)-2)e^{\frac{-(x^2+y^2)}{0.72}}$.

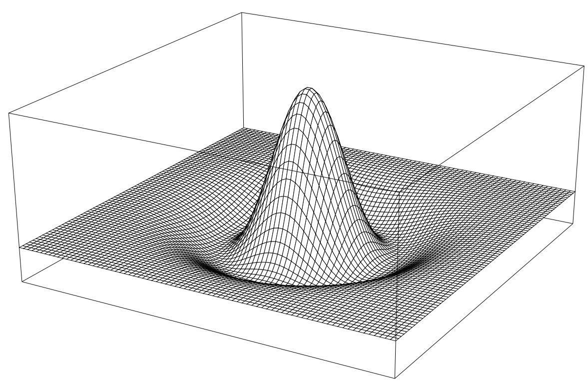


Figure 1: Mexican hat function

We start with approximation by a discrete step function using an extended feature map algorithm.

2.1 Approximation with a step function

In order to approximate a multidimensional function, we first introduce a technique which is related to feature maps [Koh84, RMS89]. In this approach the input space is devided into small regions, each of which is represented by a feature vector.
A feature map formally defines a step-function

$$FM : x \in I\!R^n \rightarrow p \in M$$

where M is a m-dimensional grid, i.e. the cartesian product of m indexsets I_i with $I_i \subset Z\!\!Z$ and some neighborhood function. Every point in the input domain is assigned to some element of M. If we consider any element $m_k \in M$ a formal neuron, its receptive field is defined as

$$\mathcal{R}_k = \{x \in I\!R^n | FM(x) = m_k\}$$

The mapping FM is formed by adapting internal parameters $\{w_i \in I\!R^n; i \in M\}$ (called weights) by the following algorithm.

1. Generate/read (next) input $x \in I\!R^n$.
2. Determine node $j \in M$ whose parameter set (weights) is closest[2] to x.
3. Let $\Gamma(j) \subset M$ be a symmetrically defined neighborhood of the winning node j in the m-dimensional grid. $\forall i \in \Gamma(j)$ adapt the parameters according to the rule:

[2]This can be done by minimizing the Euclidean distance $||w_j - x|| = \min_i ||w_i - x||$, $i \in M$ or, in case of normalized weights by maximizing the inner product: $w_j x = \max_i\{w_i x\}$, $i \in M$

$$w_i(t+1) = w_i(t) + \alpha_i(t)(x(t) - w_i(t))$$

where $\alpha_i(t) = \epsilon(t)h_{ij}(t)$ consists of a stepsize $\epsilon(t)$ and a neighborhood term $h_{ij}(t)$ that is usually taken to be gaussian centered at the winning node j.

4. Change stepsize $\epsilon(t)$ and neighborhood function $h_{ij}(t)$, such that these are monotonically decreasing.
5. Goto step 1.

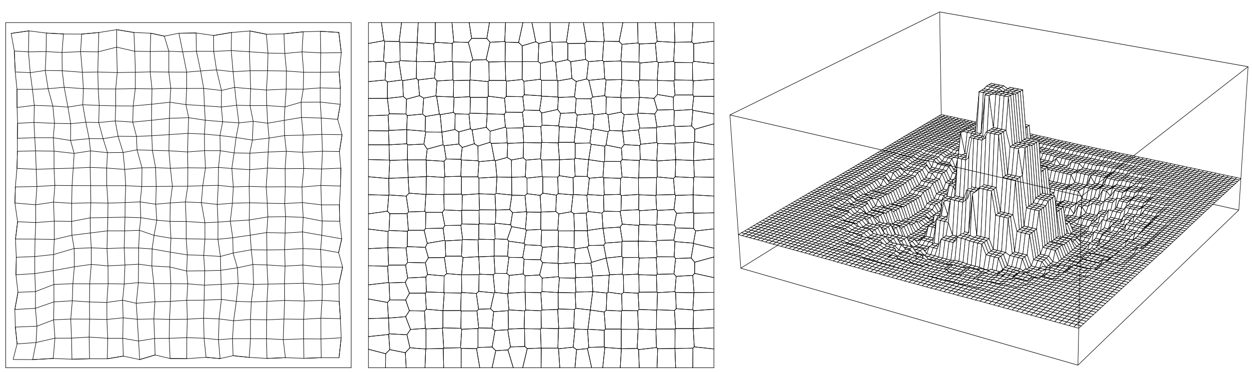
An extension of this algorithm was used by Ritter, Martinez and Schulten [RMS89] to perform function approximation. They assigned to every node in the FM-grid a set of output parameters that are adapted over time to minimize the LMS difference of the output value and the desired k-dimensional $f(x)$. Formally they introduced an output mapping OUT^3:

$$OUT : p \in M \rightarrow o \in I\!R^k$$

For each node of the feature map some output value is determined during learning, which minimizes the L_2-norm over its receptive field. Kohonen's algorithm has to be changed slightly by introducing an additional adaption step. It might, for example, be realized by a gradient descent technique on a quadratic error function (Widrow-Hoff or delta rule):

3.a $o_i(t+1) = o_i(t) + \alpha'_i(t)(f(t) - o_i(t))$ where $o_i(t)$ denotes the output weight vector of unit i, $f(t)$ the target value to be approximated and $\alpha'_i(t)$ the stepsize at time t.

This extended algorithm yields a stepwise approximation of the function that is governed by the pattern distribution.
We used it to approximate the mexican hat function. Results obtained with a uniform distribution of input patterns are given in figure 2. After 850000 sample presentations learning converged. A plot of the distribution of feature map units and a plot of the approximated mexican hat function is shown.



a b c

Figure 2: Approximation of mexican hat with a step function using feature maps. a. Distribution of units (20×20) b. Receptive fields c. Plot of approximation (tss=2.5).

Obviously this approximation is not optimal, i.e. with the same number of feature map units (patches) a better approximation can be derived (cf. figure 3.), if the size of receptive fields (subtasks) is allowed to change independently of the input distribution.

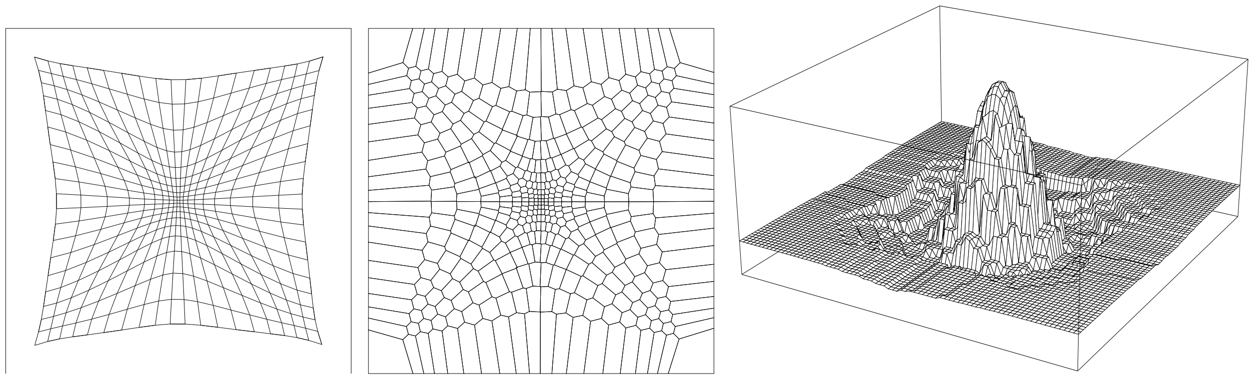
2.2 Error-driven decomposition

Obviously the above example is not optimal with respect to the L_2-norm. The overall error of the approximation $E = \int_M P(x)|o(x) - f(x)|^2 dx$ is influenced by two factors:

- the probability distribution $P(x)$ of the input domain and
- the squared error (L_2-norm) of the approximation

The contribution of a FM-unit i to this error is given by $E_i = \int_{\mathcal{R}_i} P(x)|o(x) - f(x)|^2 dx$, i.e. the size of receptive fields should be determined not by the pattern distribution alone, but by the quality of the local approximation as well. In this approach the decomposition of input space is done in an error-driven manner, i.e. the current approximation L_2-error of each region is used for changing region boundaries: the larger the error of a region, the more the region is narrowed. This leads to a very simple "learning" rule which also adapts to the underlying pattern distribution. The crucial point is to modify the above algorithm by an error term $pss(t) = \sum_j (o_j(t) - f_j(t))^2$, that indicates the "difficultness" of the local approximation task.

$$w_i(t+1) = w_i(t) + \alpha_i(t)pss(t)(x(t) - w_i(t))$$



a b c

Figure 3: Error-driven stepwise approximation: a. Distribution of units (20×20) b. Receptive fields c. Plot of approximation (tss=1.24).

In figure 3 results (with an even pattern distribution) after 900,000 pattern presentations are shown. Note the changed distribution of units (cf. fig. 2.) that is driven by the error term. A discussion of convergence properties of this method may be found in [Moe91].
So far we presented an error-driven decomposition method whose approximation capabilities are restricted due to the obvious limitations of stepfunctions. We will now extend this approach to nonlinear approximation.

3 Linear and nonlinear approximation

A nonlinear approximation can be achieved with plain backpropagation networks[4]. Unfortunately, a huge effort is required to approximate the mexican hat function. We used a network with 100 hidden units for the result presented in figure 4. Training was performed on a 16k Connection Machine. After almost 1 billion of pattern presentations (batch training with 16k patterns) learning converged. We tried a dozen of other architectures without any noticable improvements.

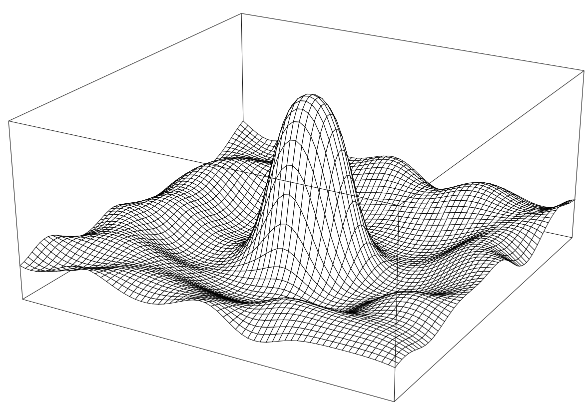


Figure 4: Approximation with pure Backpropagation (tss=7.3)

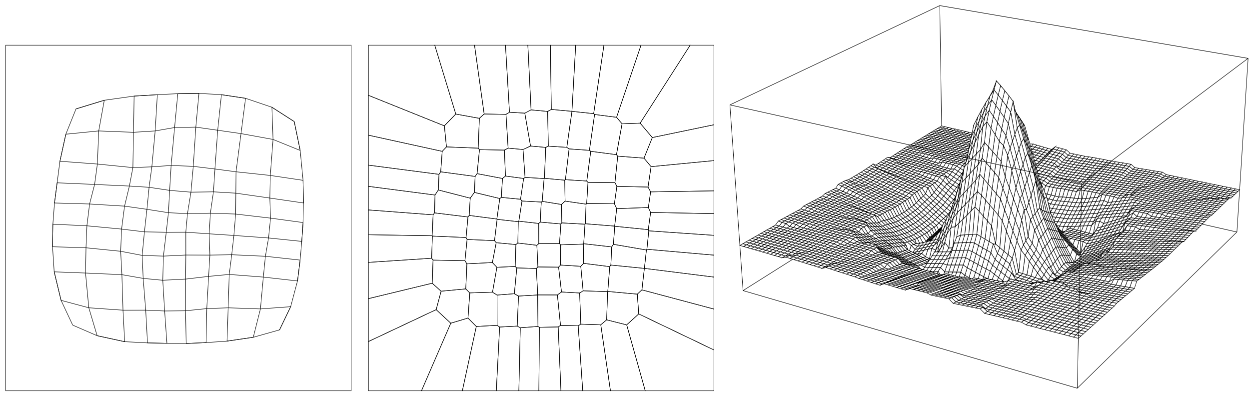
Since we are not really interested in approximating mexican hat functions with backpropagation, we omit a statistically valid investigation. But the slowness (almost one billion of pattern presentations!) of the pure bp-approach should be sufficiently clear.

The learning algorithm is now extended to nonlinear approximations by associating a small backpropagation networks with each feature map unit. Feature map units are used for gating the input to their associated bp-network. Thus, each receptive field (which are shown in fig. 5,6 b) represents the input (or task) space of one small bp-network.
In our pseudo-algorithmic notation: step 3.a has to be exchanged by a backpropagation learning step:

3.b Train the backpropagation network associated to the winning node with the actual pattern pair $(x, f(x))$.

[3] This was enhanced by an approximation of the Jacobian to linearly interpolate between OUT-values. Their method relies on knowledge about the inverse of the function f.

[4] This is what we actually started with.

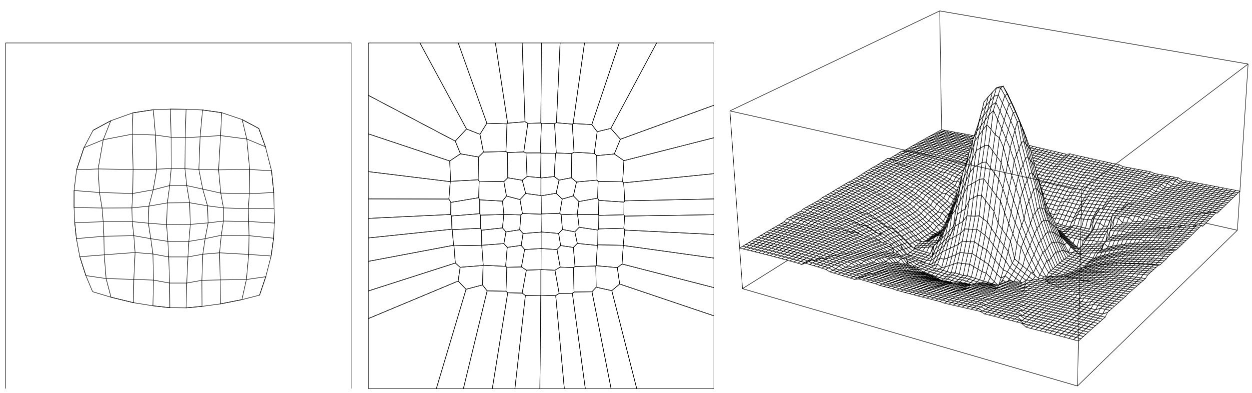


a b c

Figure 5: Linear approximation with decomposition (no hidden units). a. Distribution of linear nets (10×10) b. Receptive fields c. Plot of approximation (tss=0.36).

This decomposition method has a considerable effect on backpropagation learning. With a 10×10 feature map, i.e. 100 small backpropagation nets, (in the former demonstrations we used 400 feature map units) the following results were achieved:

- The approximation result after 8 million pattern presentations using no hidden units is shown in figure 5 (tss=0.36)
- The corresponding results (tss=0.07) with 3 hidden units are shown in figure 6.



a b c

Figure 6: Nonlinear approximation with decomposition (3 hidden units). a. Distribution of backprop nets (10×10) b. Receptive fields c. Plot of approximation (tss=0.07).

In our experiments the learning capabilities turned out to be robust with respect to parameter settings. We used a learning rate of .5 and a momentum term of .9 . A survey of results is given in the following table.

Method	Dimension	# hidden units per net	TSS	Patternpresentations in [millions]
Backprop	1	100	7.30	860
FM	20×20	0	2.52	1.0
FMerror-dvn	20×20	0	1.24	1.0
FMlinear	10×10	0	0.36	8.0
FMbackprop	10×10	3	0.07	8.0

4 Conclusion and future work

We described a nonlinear learning method based on error-driven decomposition. This technique combines an unsupervised learning method (feature maps) with a number of small supervised bp-networks. The local adaption error of the networks (*pss*) drives the learning of the feature map. The effect of feature map learning is an assignment of a receptive field to a bp-net in a way, that the variance of error over all FM-units is minimized. Usually a minimal overall error (with respect

to the L_2-norm) is approached.
It should be noted, however, that there is no general convergence proof for our method. But during our extensive testing we never observed divergence. Extensions, that are being pursued, include:

- smoothness at the receptive field borders could be enforced. We are currently investigating how to combine gradient informations in neighboring networks such that the resulting patches are smooth or even differentiable.
- combining several competing grids with different bp-architectures.
- active exploration, i.e. the focus of attention on those regions in workspace that are important and whose approximations are less satisfying. It is straightforward how our learning method can be used to generate input for those parts where knowledge is still inadequate [TM91].

To summerize: The motivation for this approach was

- to support coarse grained parallelization e.g. on transputer nets,
- to accelerate learning of real-world problems such as robot kinematics and dynamics. This is accomplished by function decomposition i.e. the use of many small networks (with few hidden units) instead of one big network, since in the latter case hidden unit representations are more difficult to develop.
- to ease relearning. If a learning system is facing a constantly changing environment, fast relearning is necessary. Clearly pure backpropagation is not well-suited for online training with open training sets. Error-driven decomposition allows to keep certain changes local.

In this paper we presented a first demonstration of the capabilities of our new technique. Scaling to real world applications in the robotics domain are under investigation.

References

[BH89] E. Baum and D. Haussler. What size net gives valid generalization. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, pages 81–89, IEEE, Morgan Kaufmann, San Mateo, CA, 1989.

[BP84] M. Bercovier and T. Pat. A C^0 finite element method for the analysis of inextensible pipe lines. *Computers and Structures*, 18(6):1019–1023, 1984.

[Cha89] Y. Chauvin. A back-propagation algorithm with optimal use of hidden units. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, pages 519–526, IEEE, Morgan Kaufmann, San Mateo, CA, 1989.

[Far90] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, San Diego, 1990.

[FL90] S. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, pages 524–532, IEEE, Morgan Kaufmann, San Mateo, CA, 1990.

[HS78] E. Horowitz and S. Sahni. *Fundamentals of computer algorithms*. Computer Science Press, 1978.

[JJB90] R. A. Jacobs, M. I. Jordan, and A. G. Barto. *Task decomposition through competition in a modular connectionist architecture: The what and where vision task*. Technical Report COINS TR 90-27, Dept. of Brain and Cognitive Science, Massachusetts Institute of Technology, Cambridge,MA, 1990.

[Koh84] T. Kohonen. *Self-Organization and Associative Memory*. Springer, Berlin New York, 1984.

[Moe91] K. Möller. *Error-driven decomposition*. Technical Report, Bonn University, *in preparation*, 1991.

[MT90] K. Möller and S. Thrun. Task modularization by network modulation. In J. Rault, editor, *Neuro-Nimes '90*, pages 419–432, 1990.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing. Vol. I + II*, MIT Press, 1986.

[RS88] H. Ritter and K. Schulten. Extending Kohonen's self organizing mapping algorithm to learn ballistic movements. In R. Eckmiller and C. von der Malsburg, editors, *Neural Computers*, Springer-Verlag, Berlin, 1988.

[RMS89] H. Ritter, T. Martinez, and K. Schulten. Topology-conserving maps for learning visuo-motor-coordination. *Neural Networks*, 2(3):159–168, 1989.

[TML90] S. Thrun, K. Möller, and A. Linden. Adaptive look ahead planning. In G. Dorffner, editor, *Konnektionismus in Artificial Intelligence*, Springer-Verlag, Berlin, 1990.

[TML91] S. Thrun, K. Möller, and A. Linden. Planning with an adaptive world model. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, IEEE, Morgan Kaufmann, San Mateo, CA, 1991. to appear.

[TM91] S. Thrun and K. Möller. *On planning and exploration in non-discrete environments*. Technical Report 528, GMD, St. Augustin, FRG, February, 1991.