* †

INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

Cyclical Local Structural Risk Minimization with Growing Neural Networks

Jan Matti Lange

TR-96-015

April 1996

Abstract

With that paper a new concept for learning from examples called Cyclical Local Structural Risk Minimization (CLSRM) minimizing a global risk by cyclical minimization of residual local risks is introduced. The idea is to increase the capacity of the learning machine cyclically only in those regions where the effective loss is high and to do a stepwise local risk minimization, restricted to those regions. An example for the realization of the CLSRM principle is the TACOMA (**TA**sk Decomposition, **CO**rrelation **M**easures and local **A**ttention neurons) learning architecture. The algorithm generates a feed-forward network bottom up by cyclical insertion of cascaded hidden layers. The Output of a hidden unit is locally restricted with respect to the network input space using a new kind of activation function combining the local characteristic of radial basis functions with sigmoid functions. The insertion of such hidden units increases the capacity only locally and leads finally to a neural network with a capacity well adapted to the distribution of the training data. The performance of the algorithm is shown for classification and function approximation benchmarks.

*on leave from Research Group Statistical Learning and Pattern Recognition, Center for Applied Computer Science (GFaI) Berlin, jl@gfai.fta-berlin.de

†This work is supported by the Staatsminister für Wirtschaft und Arbeit des Freistaates Sachsen as part of the project Intelligent Inspection Engine

1 Introduction

Information processing systems realize mappings from inputs to outputs. The computation of the mapping rule using a set of exemplary pairs of input and output data is called learning from examples. The goal of a learning machine is minimization of a risk resulting from a faulty mapping. The definition of what faulty means as function of desired and actually produced output depends on the application and is called loss function. The expectation value of the loss is called risk and the empirical mean of the loss on an identically distributed subset of observations of input and outputs is called empirical loss. The principle to approximate the mapping rule minimizing the risk by a mapping rule minimizing the empirical risk is called Empirical Risk Minimization (ERM) [Vap95]. The set of possible mapping rules, quasi the search space for the ERM, is restricted by additional assumptions called a priori knowledge.

There is a need for learning machines as well with high convergence speed needing only a small number of training samples as with a small risk. These are two contradict claims. As smaller the set of possible mapping rules is as greater is the convergence speed but as greater is also the minimal possible risk. Vice versa as greater the set of possible mapping rules is, as more training data are required, as lower is the convergence speed but as smaller is the minimal possible risk.

Vapnik and Chervonenkis did an important contribution with their fundamental works on the theory of statistical learning. They introduced the concept of the entropy of a set of functions as a measure for the number of realizable mapping rules describing the capacity of a learning machine. Based on that they formulated the conditions for the consistence of the ERM principle and derived an equation describing an upper bound of the risk depending on the capacity of the learning machine and the number of training samples. For each learning problem there exist a capacity of the learning machine depending on the number of training samples for which the upper bound of the risk is a minimum. The concept to take that dependency into account while one constructs a learning machine and to control the capacity by a priori knowledge on the problem to be solved so that the upper bound resulting from the application of the ERM principle becomes minimal is called Structural Risk Minimization (SRM).

The most frequently used method for the implementation of the SRM principle is to minimize the empirical risk holding the capacity of the learning machine fixed. One applies the ERM principle to a bounded set of mapping rules defined by all possible different parametrizations of a function. For instance in case of neural networks these are all mapping rules which one achieves by all possible weight values. The maximum capacity is controlled by the choice of the topology and the node functions of the network.

Another method is to minimize the capacity and to hold the empirical risk. Realizations are the Support Vector Machine of Vapnik [Vap95] and different pruning algorithms for neural networks.

A new concept realizing the SRM principle is the Cyclical Structural Risk Minimization (CSRM). The idea is to increase the set of possible mapping rules and to minimize the residual empirical risk alternately over a number of cycles. The capacity is controlled by a priori knowledge of how to increase the set of possible mapping rules. The goal with each cycle is to reduce the residual empirical risk as much as possible by a minimum growth

of capacity. The learning stops if the capacity exceeds a threshold or if the empirical risk is small enough. The Cascade Correlation Learning Architecture (CASCOR) [FL90] is an example for the realization of that principle. The realization of that idea can be used with advantage if it is more clear how to increase the capacity at a certain point as what is a good maximum capacity a priori. In that case the engineering effort might decrease significantly because it is not more necessary to do a number of time-consuming experiments with different sets of mapping rules.

The upper bound of the risk, the worst case solution of the ERM principle, depends on the number of training samples. If one would cut the input space in subspaces of same size and there would be different numbers of samples in the subspaces, resulting from a non-uniform density, one could outcome with different upper bounds for the risk. The upper bound of the risk is a local measure and the capacity of the learning machine has to be controlled locally. The concept dealing with such cases, called Local Risk Minimization (LRM) principle, is a generalization of the SRM principle [Vap95], [BV92].

With that paper a new concept called Cyclical Local Structural Risk Minimization (CLSRM) is introduced. By combination of LRM and CSRM it is possible to minimize a global risk by cyclical minimization of residual local risks. From the general setting of learning as risk minimization it is clear that the capacity has only to be increased in such regions where the local empirical risk is still high because to increase the local capacity where the local empirical risk is already small increases only the upper bound of the risk in that region and does probably not reduce significantly the local risk. In generalization of the CSRM principle the local capacity get increased and the local empirical risk becomes minimized alternately until the global empirical risk is smaller than a threshold. The idea is first to estimate regions with high residual empirical risk and than to extend the set of possible mapping rules by a set of new rules producing a more complex mapping from this regions and secondly to do a local empirical risk minimization procedure over this set. The problem arising with that approach is to compute the regions automatically from the training set. A technique called Error Mapping doing that task well is introduced.

The CLSRM principle is also plausible from the AI point of view. It realizes the paradigm of task decomposition to divide a large problem in several subproblems and to solve them separately by well adapted experts or modules. The concept to build the learning machine bottom up by cyclically local extension of the mapping rule leads to a global controlled development of locally adapted experts.

An example for the realization of the CLSRM principle is the TACOMA (**TA**sk Decomposition, **CO**rrelation **M**easures and local **A**ttention neurons) learning Architecture [LVW94b], [LVW94a]. The algorithm generates a feed-forward network bottom up by cyclically inserting cascaded hidden layers. The Output of a hidden unit is locally restricted with respect to the input space of the network using an activation function combining the local characteristic of a radial basis function with a sigmoid unit. The insertion of such hidden units increases the set of mapping rules and therefore the capacity only locally. The number of hidden units inserted with a cycle depends on the number of regions with high local empirical risk estimated by the error mapping procedure. They are trained to reduce the empirical local residual risk.

The outline of the paper is as follows. First the general setting of learning from examples as risk minimization and the concepts ERM, SRM and LRM are explained. The second

section introduces the new concept of CLSRM. It starts with a description of the CSRM concept. Based on that the need for combination of LRM and CSRM is shown and the idea of cyclical local extension of the set of mapping rules is explained. In the third section the TACOMA learning architecture as realization of the CLSRM principle is explained and the advantages are shown at hand of different examples. The fourth section gives a conclusion and an outlook.

2 Learning from Examples as Risk Minimization

2.1 General Setting

Information processing systems realize mappings from inputs to outputs $g : x \rightarrow y$. To use the classical engineering approach to build such a system one has to know the mapping rule a priori. In that case the inductive inference leading from observation and experience to the mapping rule is done by humans. The other approach, to compute the mapping rule using a set of exemplary pairs of input and output data, is called learning from examples. The learning machine has to do an inductive inference based on observations and a priori knowledge restricting the set of possible mapping rules. The general setting in terms of statistics is as follows.

Given is a generator producing independent random input vectors $x \in R^n$ from an unknown stationary[1] distribution $F(x)$ and a meta model[2] producing for each x an output vector $y \in R^m$ from an unknown stationary joint distribution $F(y|x)$. The goal of learning is to find a mapping rule $g(x)$ based on a identically distributed subset D of observations of input-outputs pairs$\{(x_i, y_i) : i = 1 \ldots l\}$ minimizing the expectation value of a loss $L(y, g(x))$ resulting from a faulty mapping.

The definition of what faulty means as function of desired and actually produced output of the mapping depends on the application and is called loss function.

- pattern recognition, classification

$$L(y, g(x)) = \begin{cases} 0 & \text{if} \quad y = g(x) \\ 1 & \text{if} \quad y \neq g(x) \end{cases}$$

- regression, function approximation

$$L(y, g(x)) = (y - g(x))^2$$

The expectation value of the loss is called risk

$$R = \int L(y, g(x)) dF(x, y)$$

[1] The stationarity of the process is a necessary condition for the type of learning machines described in that paper.

[2] A meta module can be a human or another model, for example in case of classifying different objects from pictures the meta model has to give the class-descriptor y for each picture x.

and the empirical mean of the loss on D

$$R_{\text{emp}} = \frac{1}{l}\sum_{i=1}^{l} L(y_i, g(x_i))$$

is called empirical loss. The set G is a subset of all possible mapping rules restricted by a priori knowledge.

$$G = \{g : \mathrm{R}^n \rightarrow \mathrm{R}^m\} \subset \{f : \mathrm{R}^n \rightarrow \mathrm{R}^m\}$$

The principle to approximate the mapping rule g_* minimizing the risk $R(G)$ by a mapping rule g minimizing the empirical risk $R_{\text{emp}}(G)$ is called Empirical Risk Minimization (ERM) [Vap95].

2.2 Balancing the A Priori Knowledge

To get the right g by the ERM using an optimization procedure one has to take the condition

$$\lim_{l \rightarrow \infty} P\{\sup_{g \in G} \left(R(g) - R_{\text{emp}}(g)\right) > \varepsilon\} = 0 \quad \forall \varepsilon > 0 \tag{1}$$

into account for which the ERM method is nontrivial consistent and to choose the right set G fulfilling condition 1. With other words, the ERM is consistent if the probability of the maximum difference between the risk and the empirical risk resulting from the mapping rule g, chosen in worst case from G, converges to zero with increasing number of samples. The problem of learning from examples is to get a small probability with small sample sizes, that is to get high convergence speed.

From that point it is also clear that there exist no universal learning machine based on the ERM principle. One would need the universal a priori knowledge which would result in an unrestricted set of possible mapping rules not fulfilling condition 1. One can only build learning machines for distinct classes of problems which are bounded by similar a priori knowledge.

The problem of small sample sizes is described by the Bias Variance Dilemma [GBD92]. As smaller the set of possible mapping rules is as greater is the convergence speed but as greater is also the minimal possible risk. That is the case of high bias and low variance. The variance of the loss resulting from the multiple application of the ERM method to the same problem, but with different sample sets, becomes small. But the expectation value of the loss at each x which is sometimes called bias, and therefore the risk is high, if the small set of mapping rules is not choosen with high amount of a priori knowledge. In terms of the function approximation community that is called underfitting. Vice versa as greater the set of possible mapping rules, as more training data are required, as lower is the convergence speed but as smaller is the minimal possible risk. That is the case of low bias and high variance also called overfitting. The learning machine can fit the samples at mean very well resulting in low biases but the outcoming mapping rules are quite different leading to a high variance of the loss.

There is a need for well generalizing[3] learning machines as well with high convergence speed needing only a small number of training samples as with a small risk. But as shown

[3]footnote The term generalization is borrowed by the neural network community from the psychology and describes how well a learning machine can generalize from the training set to unseen samples

these are two contradict claims representing the fundamental question of statistical learning pointed out by Vapnik [Vap95]: *What must one know a priori about an unknown functional dependency in order to estimate it on the basis of observations ?*

2.3 The Upper Bound of the Risk

Vapnik and Chervonenkis did an important contribution with their fundamental works on the theory of statistical learning. They introduced the concept of the entropy of a set of functions describing the capacity of a learning machine. Based on that they formulated the conditions for the consistence of the ERM principle and derived an equation describing an upper bound of the risk depending on the capacity of the learning machine, the number of training samples and the empirical risk [Vap95]:

> Let $A \leq L(y, g(x)) \leq B$, $g \in G$ be a set of totally bounded functions. Then following inequalities hold with probability of at least $1-\eta$ simultaneously for all functions $L(y, g(x))$, $g \in G$ (including the function that minimizes the empirical risk):
>
> $$R(g) \leq R_{\text{emp}}(g) + \frac{B-A}{2}\sqrt{\varepsilon}, \tag{2}$$
> $$R(g) \geq R_{\text{emp}}(g) - \frac{B-A}{2}\sqrt{\varepsilon},$$
>
> with
>
> $$\varepsilon = 4\frac{h\left(\ln\frac{2l}{h}+1\right) - \ln\frac{\eta}{4}}{l}$$
>
> and h the VC dimension of the set of lossfunctions.

2.4 Structural Risk Minimization

The curve of the upper bound of the risk over the capacity mirrors the Bias-Variance Dilemma, see figure 1. For each learning problem there exist a capacity of the learning machine depending on the number of training samples for which the upper bound of the risk is a minimum.

When l/h is large, ε is small. Therefore, the second summand on the right-hand side of the inequalities 2 becomes small, the upper bound depends mostly from the empirical risk. But with small sample sizes l/h is small to and a small empirical risk does not guarantee a small upper bound. The concept to take the second summand into account while one constructs a learning machine and to control the capacity by a priori knowledge on the problem to be solved so that the upper bound resulting from the application of the ERM principle becomes minimal is called Structural Risk Minimization (SRM) [Vap95]. The concept of the SRM is to structure a set of possible loss functions resulting from G in nested subsets S_k and to choose the smallest S_k which contains the lossfunction and so fare the mapping rule minimizing the upper bound of the risk.

The SRM principle encompasses two parts. Choose a set of mapping rules by a priori knowledge and apply of the ERM principle.

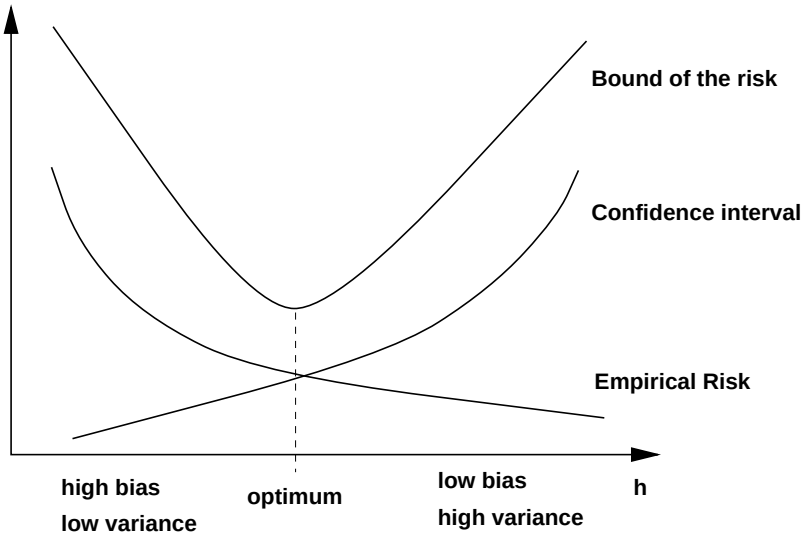

Figure 1: The upper bound of the risk as sum of empirical risk and confidenc interval.

The most frequently used method for the implementation of the SRM principle is to minimize the empirical risk holding the capacity of the learning machine fixed. One applies the ERM principle to a bounded set of mapping rules defined by all possible different parametrizations of a function. For example in case of neural networks these are all mapping rules which one achieves by all possible weight values. The maximum capacity is controlled by the choice of the topology and the node functions of the network. An additional possibility for the control of the capacity is given by extension of the empirical risk function by a regularizations term. The regularization term modifies the objective function for the optimization process in order to prefer mappings better fitting a priori assumptions. In case of feed forward nets with sigmoid activation functions often a so called weight decay term is used for regularization[4]. The a priori knowledge based is to prefer smooth mappings. The degree of regularization is controlled by a factor of the regularization term in the objective function. To estimate this factor from data one can use the cross validation technique.

Another SRM method is to minimize the capacity and to hold the empirical risk fixed. Realizations of that are the Support Vector Machine of Vapnik [Vap95] and pruning algorithms for neural networks.

2.5 The Local Risk Minimization Principle

If it is impossible to find a mapping reducing the empirical risk clearly on the basis of a given set of possible mapping rules then there are two chances to overcome this problem. Extension of the set of possible mapping rules acording to the SRM principle or local approximation of the desired function at any point of interest. The second one is called Local Risk Minimization (LRM) principle [Vap95]. The idea is to use a non-negative window

[4]This type of regularization is in the statisics community called ridge regularization

function $0 \leq w(x, x_0; \beta) \leq 1$, $w(x_0, x_0; \beta) = 1$ with

$$w(x_0, \beta) = \int w(x, x_0; \beta) dF(x)$$

to introduce the concept of locality. The window function depends on the point x_0, the mean of the window and β, describing the range of the window. For example one can use a gaussian window

$$w(x, x_0; \beta) = e^{-\frac{(x-x_0)^2}{\beta^2}}. \tag{3}$$

The goal is to minimize the local risk functional

$$R(G, \beta; x_0) = \int L(y, g(x, G)) \frac{w(x, x_0; \beta)}{w(x_{0;\beta})} dF(x, y) \tag{4}$$

over both, the set of functions G and different vicinities of the point x_0 defined by β. The upper bound of local risk which holds with $\eta - 1$ for all functions $A \leq L(y, g(x)) \leq B$ and all functions $0 \leq w(x, x_0; \beta) \leq 1$ is given by Vapnik [Vap95] as

$$R(G, \beta; x_0) \leq \frac{\frac{1}{l} \sum_{i=1}^{l} L(y_i, g(x_i)) w(x_i, x_0; \beta) + (B - A) \varepsilon(l, h_{\sum})}{(\frac{1}{l} \sum_{i=1}^{l} w(x_i, x_0; \beta) - \varepsilon(l, h_\beta))_+} \tag{5}$$

with

$$\varepsilon(l, h) = \sqrt{\frac{lh(\ln(2l/h + 1) - \ln \eta/2}{l}}.$$

l is the number of training samples, $h_{\sum}$ is the VC dimension of the set of functions $L(y, g(x)) w(x, x_0; \beta)$ and h_β is the VC dimension of the set of window functions. The global risk minimization problem is a special case of local risk minimization which occurs for $w(x, x_0; \beta) = 1$.

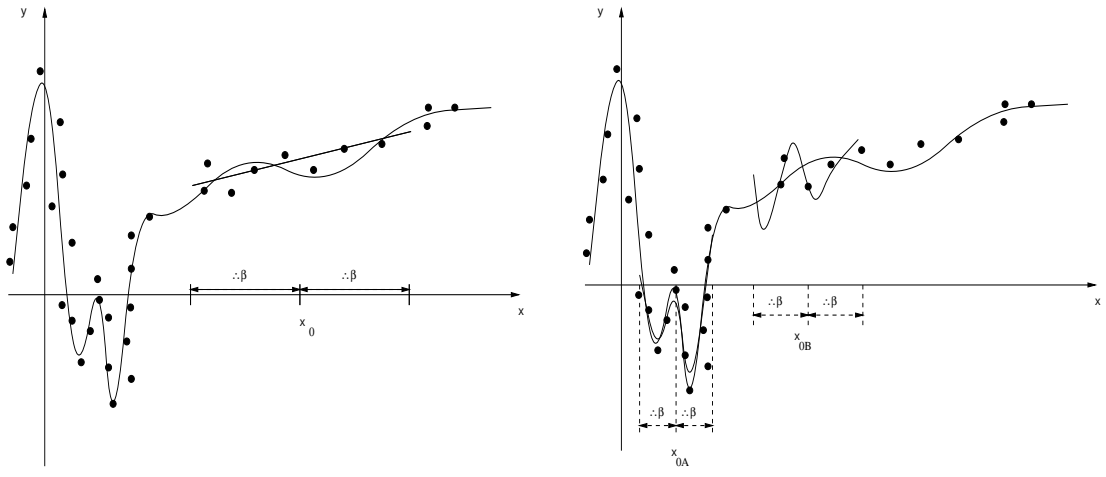

Figure 2: The principle of local risk minimization

The LRM principle becomes more clearly at a function approximation example at figure 2, left side. Using a set containing only linear functions it is impossible to reduce the global risk successfully but it is possible to give a good approximation in the vicinity of x_0.

But there is a second property the LRM principle comes with. It is possible to use different sets of mapping rules for different areas of the input space to adapt the capacity locally to achieve lower upper bounds of the local risks. For example see figure 2 right side. The set of mapping rules used for the LRM at x_{0A} leads to a good approximation of the underlying function in the vicinity of x_{0A}. There is a low upper bound of the local risk. Using the same set of mapping rules in the vicinity of x_{0B}, one would achieve a higher upper bound of the local risk than using a set of linear mapping rules because the capacity is not well controlled (to big) in that vicinity. That becomes clear from inequation 5. The empirical risk in both cases would be nearly the same because the capacity of both sets of functions is high enough to fit the trainig data well. But the VC dimension of the set of linear functions is much smaller than from the high-capacity set. Because of that the probability of comming out with an overfitted, bad generalizing solution using the high capacity set locally at x_{0B} or globally is higher.

3 Cyclical Local Structural Risk Minimization

3.1 The concept of cyclical structural risk minimization

The basic concept of cyclical structural risk minimization (CSRM) is inspired by the SRM principle to structure a set of possible mapping rules in nested subsets and to choose the set that minimizes the upper bound of the risk. But instead of explicit structuring the set, the structure is given by a growth rule how to build a set S_{n+1} from a set S_n fulfilling the following conditions:

1. The VC dimension of the set of loss functions L_1 resulting from the set of mapping rules S_1 is one.
$$h(\{L_1\}) = 1$$

2. The VC dimension of the set of loss functions L_{n+1} resulting from the application of the growth rule on the set of mapping rules S_n is $h_n + 1$.
$$h(\{L_{n+1}\}) = h(\{L_n\}) + 1$$
Or with other words, the rule increments the VC dimension of the set of loss functions by one.

The multiple application of the growth rule leads to a structured set of loss functions similar as needed for the SRM principle. The CSRM principle can be formulated as follows:

> Starting with the set S_1 the CSRM algorithm applies alternately the ERM principle and the growth rule to S_n over a number of cycles until the empirical risk is equal or smaller than a threshold t.

The CSRM principle chooses at the basic of the growth rule, the threshold and the initial set automatically the set of loss functions with the smallest VC dimension that minimizes the empirical risk. It minimizes so fare the upper bound of the risk.

There are some neural network based heuristics realizing the CSRM principle. The algorithms differ in the growth rule and in the optimization procedure used for minimization of the empirical risk. A good overview is given by Fiesler [Fie94].

A common general approach used in feed forward networks, useful as well for function approximation as for classification tasks, is the cyclical minimization of the residual risk. The idea is to ask what additional input to the output layer is necessary to minimize the residual risk. One useful answer, proposed by Fahlman with the Cascade Correlation Architecture [FL90] is to let the new input be correlated with the residual loss and than adapt the weights of output units again. The cyclical extension of the output layers input leads to a minimization of the empirical risk. Whether it also leads to a low upper bound depends strongly on the type of hidden units inserted and their input connections, that is how the set of possible mapping rules is increased actually with each step.

3.2 Cyclical Local Risk Minimization – How to Control the Growth of the Capacity Locally

Remember the risk functional

$$R = \int L(y, g(x)) dF(x, y) \tag{6}$$

is defined as the expectation value of the loss $L(y, g(x))$. X and Y are random values with

$$F(x, y) = F(x)F(y|x).$$

Under assumption of a continuous density of X we can write

$$F(x) = \int_{-\infty}^{x} p(x)dx$$

and

$$dF(x) = p(x)dx.$$

Now equation 6 can be written as follows

$$R = \int \int L(y, g(x))p(x)dF(y|x)dx.$$

$$R = \int \left[\int L(y, g(x))dF(y|x) \right] p(x)dx. \tag{7}$$

With

$$\int L(y, g(x))dF(y|x) = E\left[L(y, g(x))|x\right]$$

the risk becomes

$$R = \int E\left[L(y, g(x))|x\right] p(x)dx. \tag{8}$$

Equation 8 shows that the risk depends on the curve of $E[L(y, g(x))|x]p(x)$ over the input space. We will call this term effective loss

$$L_{\text{eff}}(y, g(x)) = E[L(y, g(x))|x]p(x).$$

If one considers the curve of the effective loss than it is obviously that the capacity of the learning machine has only to be increased in those regions where the effective loss is high. Because to increase the capacity and so far the VC dimension in regions with small effective loss would not reduce the empirical risk significantly and only increase the upper bound of the risk in those regions.

> The principle to increase the capacity of the learning machine cyclically and only in those regions where effective loss is high and to do a stepwise local risk minimization, restricted to those regions, is called Cyclical Structural Local Risk Minimization (CLSRM).

That principle can be used with advantage to construct neural network based learning machines which recruit new hidden units step by step to minimize the residual risk. The idea is to approximate at each grow step a number of maxima of the effective loss using the concept of competitive reference vectors and to place non-overlapping window functions implementing the concept of neighborhood at the biggest maxima, see figure 3.

The concept of competitive reference vectors means that the vectors try to force out each other whereby the force increases with decreasing distance between vectors. That causes that in the case of two or more very close maxima only one reference vector marks all these maxima, the other vectors have been "forced out".

The number of big maxima gives the number of new hidden units to be inserted. The window functions, one unit per window, are used as gate for the new hidden units. That restricts the influence of the units and therefore the increase of the VC dimension to the regions described by the windows. If the insertion of a hidden unit does not reduce the residual local empirical risk adequately then a second or more units are inserted with respect to that region step by step. This heuristic leads to a very well adapted network complexity.

4 The TACOMA Learning Architecture as Realization of the CLSRM Principle

The TACOMA (**TA**sk decomposition, **CO**rrelation **M**easures and local **A**ttention neurons) algorithm [LVW94b], [LVW94a] generates a feed forward network bottom up by cyclical insertion of cascaded hidden layers starting with only output units. The capacity of the initial network is low and becomes increased locally with each growth step. The essential elements of the algorithm are:

- hidden units locally restricted with respect to the input space using a new kind of activation function combining the local character of radial basis functions with sigmoid functions realizing the concept of local extension of the set of mapping rules

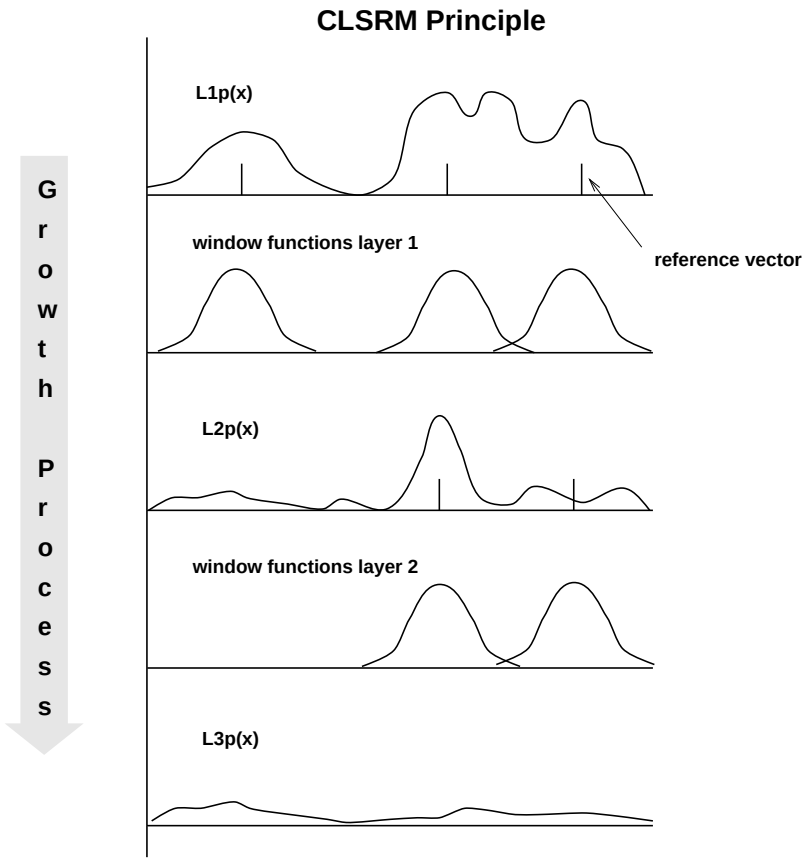

Figure 3: The CLSRM principle.

- an error mapping procedure for the approximation of the regions with high effective loss
- a connection routing procedure connecting cooperative hidden units of different layers
- a compounded objective function of different correlation measures for the training of hidden units featuring different goals

The main loop of the algorithm consists of three parts.

```
do loop {
        training of output layer;
        exit loop if error smaller than threshold;
        approximation of the effective loss and growth step;
        training of new inserted hidden layer;
}
```

The algorithm stops in case of function approximation if the mean square error at the output units is smaller than a threshold or in case of classification if the training set is classified correctly.

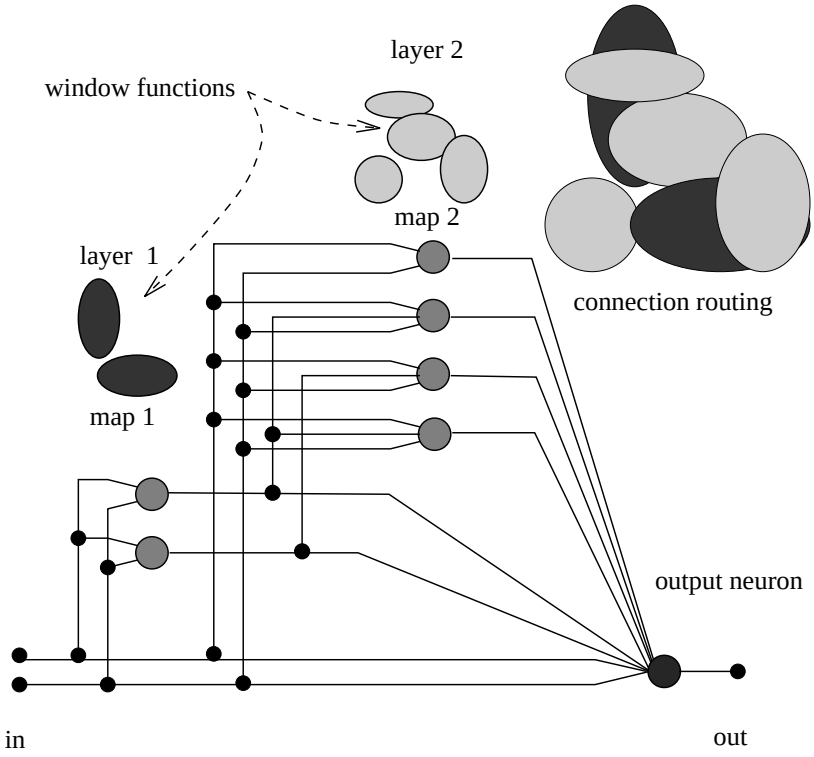

Figure 4: A TACOMA example network.

4.1 Training of the Output Layer

The output units have a sigmoid or linear activation function. They are trained cyclically after each insertion of a new hidden layer to minimize the mean square error at the output of the network by gradient descent using the Quickprop [Fah88] algorithm. The output units get the network input vector x and the outputs of all hidden units as input. At each cycle the number of inputs of an output unit is incremented by the number of units in the new inserted hidden layer, see figure 4.

4.2 Approximation of the Effective Loss and Growth Step

The approximation of a number of maxima of the effective loss is done using a mapping procedure called error mapping. It works similar to a Kohonen net but there are no neighborhood relations therefore there is no lattice structure. The error mapping net consists of a number of randomly initialized reference vectors v with the same dimension as the input space. The mapping works as follows.

> For a number of epochs compute for each pattern x_i the v_{k*} for which $\|v_{k*} - x_i\| < \|v_k - x_i\|$ holds for all $k* \neq k$ and update v_{k*} by

$$v_{k*,t+1} = v_{k*,t} + \alpha(t) \sum_{j=1}^{Outdim} |E_j(x_i)| \quad (x_i - v_{k*,t}).$$

α decreases with time to guarantee convergence. The residual error E is given by

$$E_j(x_i) = (y - o(s_j(x_i)))\frac{\partial o(s_j(x_i))}{\partial s_j(x_i)},$$

with $o(s_j(x_i))$ the output and $s_j(x)$ the weighted sum of the inputs of the j'th output unit.

The error mapping procedure computes the nearest reference vector v_{k*} to x_i and shifts it a small step in direction to x_i. The step size is proportionally to the residual error. Doing that for a number of epochs the reference vectors point approximately to the maxima of the effective loss.

To restrict a hidden units influence and so fare also the increase of the networks capacity locally a new kind of activation function for the hidden units is used combining the local character of a radial basis function with a sigmoid function. The output of a sigmoid unit is weighted by a Gaussian window function in the input space

$$h(x) = w(x)\left(\frac{1}{1+e^{-s}} - \frac{1}{2}\right)$$

with s the weighted sum of the hidden units input and $w(x)$ as window function

$$w(x) = \exp\left(-\sum_{i=1}^{Indim}\left(\frac{x_i - m_i}{r_i}\right)^2\right).$$

After error mapping each reference vector is a center point of a voronoi region in the input space, see figure 5.

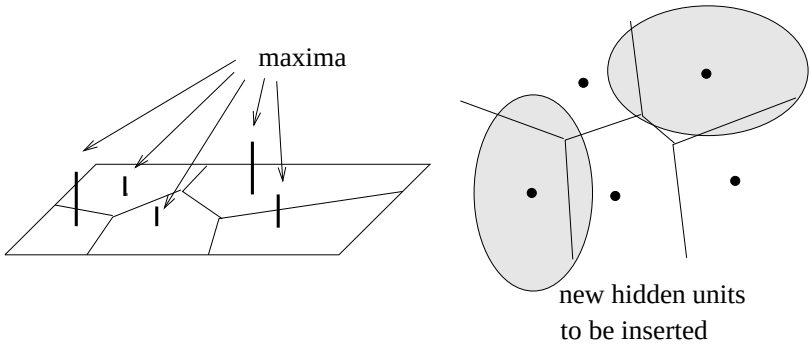

Figure 5: Estimation of the window functions.

To achieve information about the local risk in the subspaces described by the voronoi regions the local empirical risks $R_{\mathrm{emp}k}$ have to be computed for each subspace. If the local empirical risks $R_{\mathrm{emp}k}$ exceeds a threshold, then a new hidden unit has to be inserted. The number of units per layer can be different from layer to layer.

The window function of the new hidden unit is initialized to be centered at the reference vector and to encompass all patterns of the training set falling in that region. To initialize

the radii r_i of a window function the means $\overline{d_i} = 1/N_k \sum_{j=1}^{N_k} |x_{i,j} - m_i|$ for each dimension i of the network input space for all N_k patterns x_j falling in that region are computed. The r_i is initialized so that the value of the window function for $\overline{d_i}$ is equal β (for example $\beta = 0.55$).

$$r_i = \sqrt{\frac{-\left(\overline{d_i}\right)^2}{2 \ln \beta}}$$

The next step is to insert the new hidden units into the existing network using the following connection routing procedure.

1. The units in the output layer get their inputs from the outputs of all hidden units. That means the output of new hidden unit has to be connected with all output units.

2. Every unit in the network as well hidden units as output units gets the network input.

3. It is only necessary to connect cooperative units of different layers. Units are cooperative if their regions of interest overlap significantly. A new hidden unit to be inserted gets input connections from all previously inserted hidden layer units if the overlap computed as correlation coefficient between two window functions h_l and h_m over the training set exceeds a threshold γ

$$R_{l,m} = \frac{\sum_{i=1}^{N} [h_l(x_i) h_m(x_i)]}{\sum_{i=1}^{N} h_l(x_i)^2 \ \sum_{i=1}^{N} h_m(x_i)^2} \quad > \quad \gamma.$$

4.3 Training of New Inserted Hidden Layers

Parameters of a hidden unit which have to be optimized are the weight vector of the sigmoid function and the mean and radii vector of the window function. There are two optimization objectives: First to minimize the residual local empirical risk in the different regions and secondly to fine tune the window functions and to avoid overlapping between the regions of a layer. The units of a hidden layer have to solve different tasks represented by different regions in the input space.

The weight vector of the sigmoid function of the k'th unit of a hidden layer is trained to maximize the covariance S_k between the output h_k of the hidden unit and the residual error by a gradient procedure using Quickprop. The covariance is given by

$$S_k = \sum_{o=1}^{Outdim} |S_{k,o}| \quad \rightarrow \max$$

with

$$S_{k,o} = \frac{\sum_{i=1}^{N} h_k(x_i) E_o(x_i) - N \overline{h_k E_o})}{\sum_{o=1}^{Outdim} \sum_{i=1}^{N} E_o(x_i)^2}$$

and N the number of training patterns. That is the same as in the cascade correlation algorithm. The gradients to change the l'th element of the k'th hidden units weight vector w_k are given by

$$\frac{\partial S_k}{\partial w_{k,l}} = \sum_{i=1}^{N} \left(\frac{\partial S_k}{\partial h_k(x_i)} \ \frac{\partial h_k(x_i)}{\partial w_{k,l}} \right)$$

For the training of the parameters of the window function the two optimization objectives have to be compounded. We use the functional F which has to be maximized.

$$F = \frac{\sum_{k=1}^{L} S_k}{\sum_{k=1}^{L-1} \sum_{j=k+1}^{L} |R_{k,j}| + \eta} \quad \rightarrow \max$$

with

$$R_{k,j} = \frac{\sum_{i=1}^{N} h_k(x_i)\ h_j(x_i) - N\overline{h_k}\ \overline{h_j}}{\sqrt{\sum_{i=1}^{N} \left(h_k(x_i) - \overline{h_k}\right)^2 \left(h_j(x_i) - \overline{h_j}\right)^2}}$$

and L the number of units inserted. The nominator maximizes the covariance like above and the denominator minimizes the correlation between the outputs of the hidden layer units. That avoids that the window functions become overlapping and guarantees that the units of a hidden layer do a local risk minimization in different regions of the input space. The maximization of F is done by a simple gradient ascent with respect to the parameters $r_{k,l}$ and $m_{k,l}$ of the window function.

$$\frac{\partial F}{\partial m_{k,l}} = \sum_{i=1}^{N} \left(\frac{\partial F}{\partial y_{i,k}} \frac{\partial y_{i,k}}{\partial m_{k,l}} \right)$$

$$\frac{\partial F}{\partial r_{k,l}} = \sum_{i=1}^{N} \left(\frac{\partial F}{\partial y_{i,k}} \frac{\partial y_{i,k}}{\partial r_{k,l}} \right)$$

The update of weights and window function parameters is done simultaneously. The training of a hidden layer stops if F is stagnant or after a maximal number of epochs.

4.4 Experimental Results and Discussion

The performance of the TACOMA algorithm has been evaluated for a number of difficult benchmarks, as well classification as function approximation tasks.

4.4.1 The Two Spirals and Two Twin Spirals Classification Benchmark

These benchmarks are used because they are difficult tasks, they are simple to visualize and the quality of different network solutions can be compared by visual inspection. The Two Spirals Classification task [LW88] is to learn to discriminate between two sets of training points which lie on two distinct spirals in the x-y plane. These spirals coil three times around the origin and around one another. The Two Twin Spirals Classification task [LVW94b] has been designed to show the performance of the CLSRM principle realized with TACOMA. It is similar to the Two Spirals Classification task, see figure 6. For the evaluation of the Cascade Correlation algorithm the original program of S. Fahlman and Christian Lebiere was used. In figures 7 the decision regions are shown. In all cases the networks are trained to classify the training set without errors. For the first example the much better generalization of TACOMA in comparison to CASCOR is obviously. For the second example, see figure 8 it is shown that CASCOR can not solve the task because of the mutual disturbances of the hidden units. The insertion of hidden units which increase the capacity of the network

globally in the input space as done with the Cascade Correlation Algorithm leads to a much higher network capacity which is not adapted to the regions with different sample frequencies.

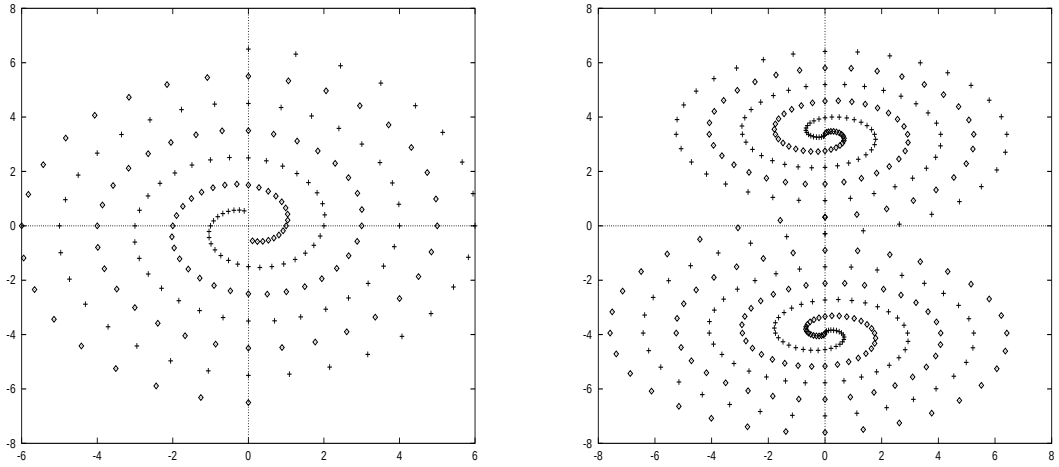

Figure 6: Training set for the Two Spirals Problem (left) and the Two Twin Spirals Problem (right).

Figure 7: Solutions of the Two Spirals Problem: left CASCOR and right TACOMA.

The solution of the TACOMA algorithm distinguishes well between the two intertwined spirals. The network structure reflects two domains of expertise, the upper and lower spirals called task1 and task2, see figure 9. The grey marked hidden units have an intermediate attention area between task 1 and task 2. Black and white marked hidden units are focused exclusively on the corresponding task. As can be seen from the number of hidden units the lower part of the Two-Twin-Spirals, i.e. task 2, is more difficult to solve because of the non-symmetric pattern distribution.

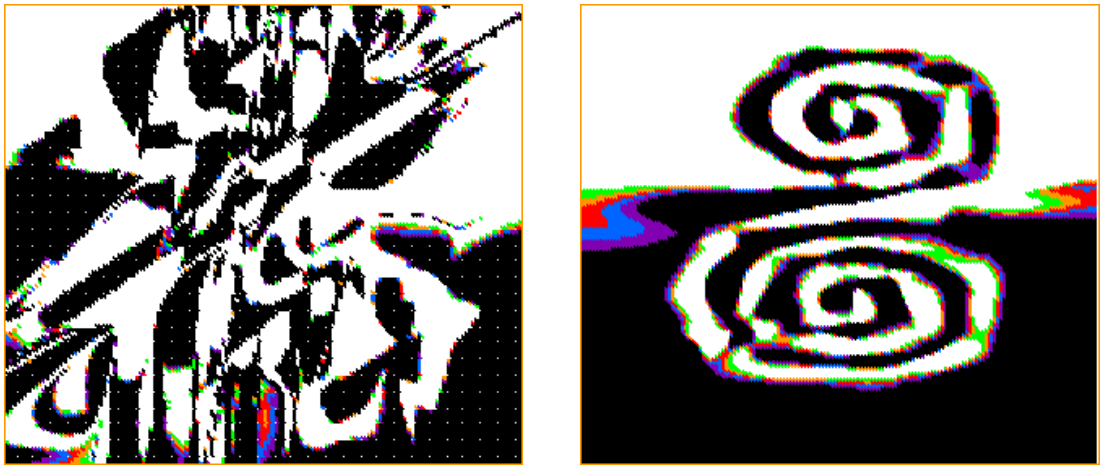

Figure 8: Solutions of the Two Twin Spirals Problem: left CASCOR and right TACOMA.

4.4.2 Function Approximation Tasks

The figures 10-12 show solutions of the TACOMA algorithm for different function approximation tasks. The first task was been used as benchmark by Hwang [HYLJ93] for the Projection Pursuit Learning algorithm and Back Propagation. The function to be approximated is

$$z(x,y) = 1.9(1.35 + e^x \sin(13(x-6)^2)e^{-y}\sin(7y)).$$

The training set contains the function values from 225 $[0,1]$-uniform distributed x, y values. The approximation achieved by TACOMA is very precise as shown at figure 10.

The next two benchmarks have been used by Frank Smieja [Smi95] for the evaluation of the Pandemonium system. These task are hard to approximate because the function is mostly very soft and only at some places are very strong chances. They are very well suited to show the good performance of the TACOMA algorithm because the capacity of the set of mapping rules has to be very different to achieve a low upper bound of the risk and solutions with good generalization. The training set contains in both cases 400 pattern lying at the nodes of a 20x20 lattice. Also in these examples the solutions of the TACOMA algorithm are very precise showing the advantages of the CLSRM principle.

5 Conclusions

The upper bound of the risk is a probability measure describing the worst case realization of a learning machine resulting from an ERM based learning process. It depends on the number of examples, the problem to be solved itself and especially on the set of possible mapping rules which can be realized by the learning machine. The curve of the upper bound over the capacity has minimum and that minimum one is looking for if one constructs a learning machine. The art of engineering is to restrict the set off possible mapping rules or

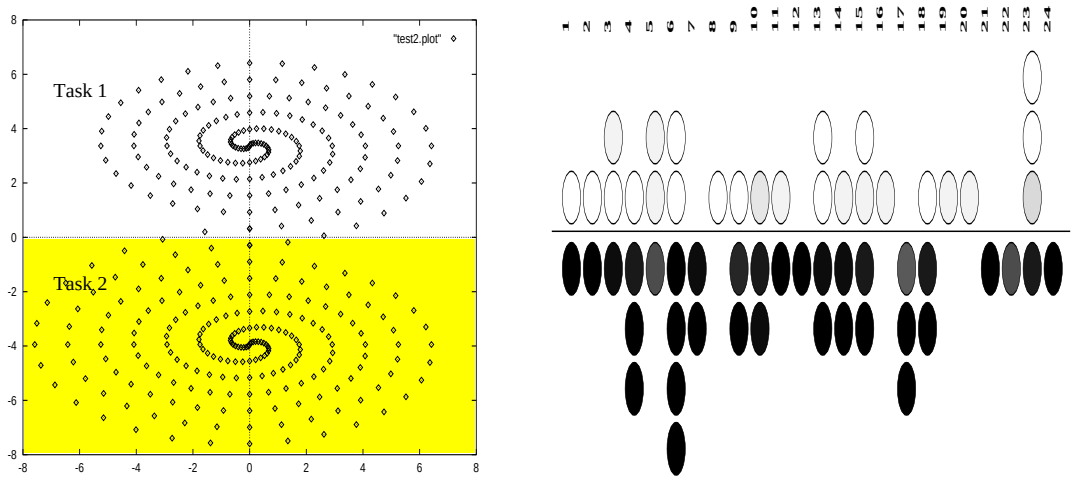

Figure 9: Attention assignment of the hidden units to different tasks, task 1 white and task 2 black, light and dark grey reflect intermediate attention areas, left hidden layer 1, right hidden layer 24

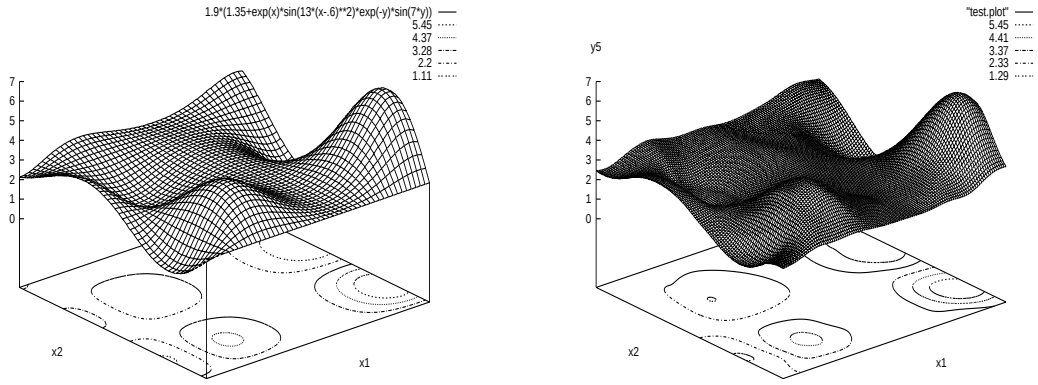

Figure 10: Function approximation Hwang, left original function, right approximation by TACOMA

the capacity by a priori knowledge. If the sample distribution varies over the input space then it is necessary to adapt the capacity locally. We proposed the CLSRM principle to build a learning machine bottom up incrementing the capacity locally step by step with respect to the residual error. We introduced the effective loss as a measure to identify regions where the capacity has to be increased. The TACOMA algorithm for growing neural networks is an implementation of the CLSRM principle. The local increase of capacity is controlled by window functions restricting the influence of hidden units with respect to the input space. The shape and center of the window functions is estimated from the approximation of the effective loss, done by the error mapping procedure.

The results of the experiments show that the proposed algorithm is able to generate complex neural networks with very good generalization abilities. It is well suited as well for classification as for function approximation tasks. Another experiment we have still to do is to estimate the bias and variance at hand of some benchmarks. We assume that both

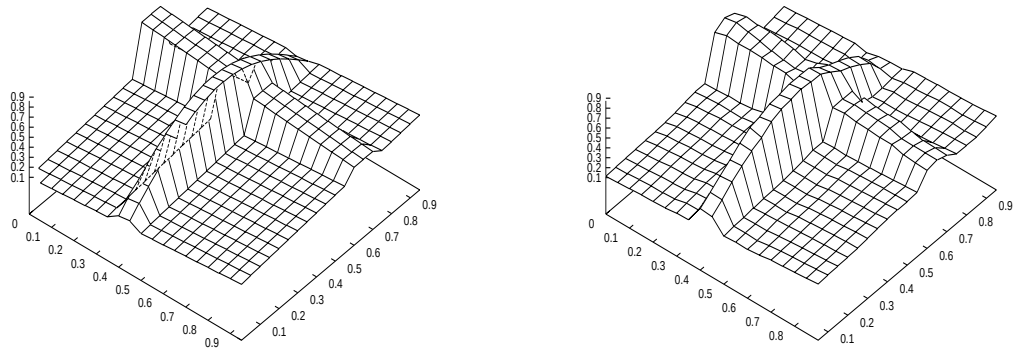

Figure 11: Function approximation, Smieja, example 1: left original function, right approximation by TACOMA

is small and well balanced showing that the CLSRM principle is able to balance bias and variance and to end with a solution where the upper bound of the risk is near the minimum.

The experiments show further that the TACOMA algorithms builds a network structure reflecting the complexity of the problem. One can identify subnets consisting of substructures of hidden units with similar regions of interest. That feature is called task decomposition. The whole task is decomposed in subtasks with respect to the input space. The network consists of subnets and can be compared with a number of cooperating experts, solving different sub problems.

The algorithm has been implemented on a MIMD parallel computer and will also be part of the free available Stuttgart Neural Network Simulator supporting different UNIX platforms.

References

[BV92] L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992.

[Fah88] Scott E. Fahlman. An empirical study of learning speed in back-propagation networks. Technical Report CMU-CS-88-162, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, September 1988.

[Fie94] E. Fiesler. Comparative bibliography od ontogenetic neural networks. In P. G. Morasso M. Mariano, editor, *Proceedings of the International Conference on Artificial Neural Networks 1994 Sorrento*, pages 735 – 738. Springer, 5 1994.

[FL90] Scott E. Fahlman and Christian Lebiere. The Cascade-Correlation learning architecture. Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, February 1990.

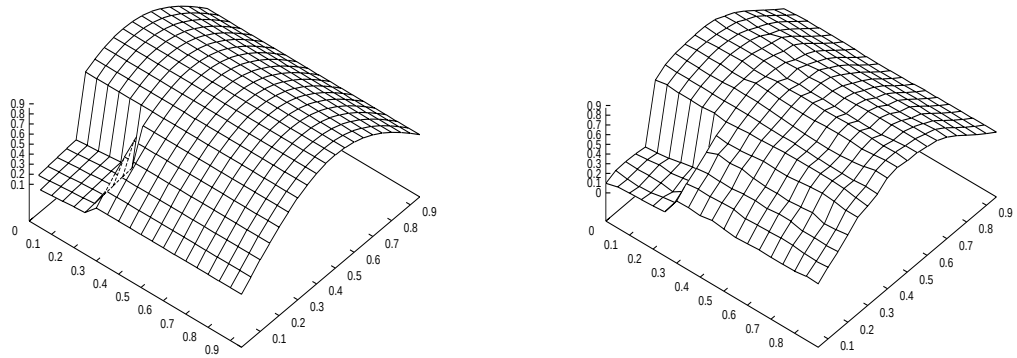

Figure 12: Function approximation, Smieja, example 2: left original function, right approximation by TACOMA

[GBD92] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[HYLJ93] Jenq-Neng Hwang, Shih-Shien You, Shyh-Rong Lay, and I-Chang Jou. What's wrong with a cascaded correlation learning network: A projection pursuit learning perspective. In *Int. Symposium on Artificial Neural Networks*, pages E11–E20, 1993.

[LVW94a] J. M. Lange, H.-M. Voigt, and D. Wolf. Growing artificial neural networks based on correlation measures, task decomposition and local attention neurons. In *Proceedings of the IEEE International Conference on Neural Networks 1994 as Part of the IEEE World Congress on Computational Intelligence Orlando*, pages 1355 – 1358. IEEE, 1994.

[LVW94b] J. M. Lange, H.-M. Voigt, and D. Wolf. Task decomposition and correlations in growing artificial neural networks. In P. G. Morasso M. Mariano, editor, *Proceedings of the International Conference on Artificial Neural Networks 1994 Sorrento*, pages 735 – 738. Springer, 5 1994.

[LW88] K. J. Lang and M. J. Witbrock. Learning to tell two spirals apart. In *Proceedings of the 1988 Connectionist Models Summer School*. Morgan Kaufmann, 1988.

[Smi95] Frank Smieja. private correspondence. GMD Sankt Augustin, Research Group Adaptive Systems, 1995.

[Vap95] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.