Journal of Machine Learning Research 8 (2007) 2017-2045 Submitted 4/05; Revised 2/07; Published 9/07

Very Fast Online Learning of Highly Non Linear Problems

Aggelos Chariatis CHARIATISAD@DIAS.COM.GR
Alamanas 2
15125 Athens, Greece

Editor: Léon Bottou

Abstract

The experimental investigation on the efficient learning of highly non-linear problems by online training, using ordinary feed forward neural networks and stochastic gradient descent on the errors computed by back-propagation, gives evidence that the most crucial factors for efficient training are the hidden units' differentiation, the attenuation of the hidden units' interference and the selective attention on the parts of the problems where the approximation error remains high. In this report, we present global and local selective attention techniques and a new hybrid activation function that enables the hidden units to acquire individual receptive fields which may be global or local depending on the problem's local complexities. The presented techniques enable very efficient training on complex classification problems with embedded subproblems.

Keywords: neural networks, online training, selective attention, activation functions, receptive fields

1. Framework

Online supervised learning is in many cases the only practical way of learning. This includes situations where the problem size is very big, or situations where we have a non-recurring stream of input vectors that are unavailable before training begins. We examine online supervised learning using a particular class of adaptive models, the very popular feed forward neural networks, trained with stochastic gradient descent on the errors computed by back-propagation.

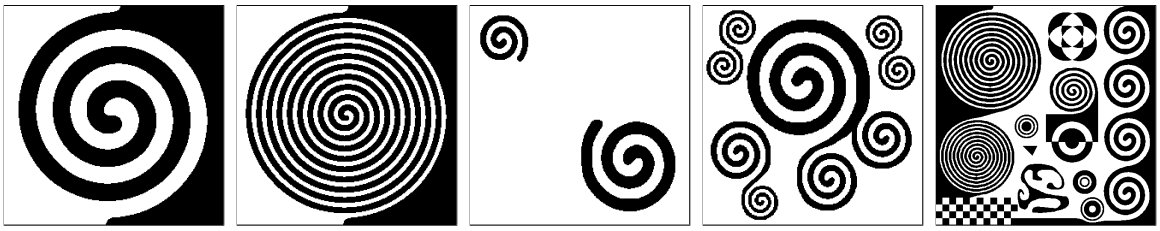
In order to easily visualize the online training dynamics of highly complex non linear problems, we are experimenting on 2:η:1 networks where the input is a point in a two dimensional image and the output is the value of the pixel at the corresponding input position. This framework allows the creation of very complex non-linear problems, just by hand-drawing the problem on a bitmap and presenting it to the network. Most problems' images in this report are 256×256 pixels in size, producing in total 65536 different samples each one.

Classification and regression problems can be modeled as black & white and gray scale images respectively. In this report we only examine training on classification problems. However, since mixed problems are possible, we are only interested on techniques that can be applied to both classification and regression.

The target of this investigation is online training where the input is not known in advance, so the input samples are treated as random and non-recurring vectors from the input space and are discarded after being used. We select and train on random samples until the average classification or RMS error is acceptable. Since both the number of training exemplars and the complexity of the underlying function are assumed unknown, we require from our training mechanism to have "initial state invariance" as a fundamental property. Thus we deliberately exclude from our arsenal any

training techniques that require a schedule to be decided ahead of training. Ideally we would like from the training mechanism to be totally invariant to the initial training parameters and network state.

This report is organized as follows: Sections 2 and 3 describe techniques for global and local selective attention. Section 4 is devoted to acceleration of training. In Section 5 we present experimental results and in Section 6 we discuss the presented techniques and give some directions for future research. Finally, Appendix A contains a description of the notations that have been used. In Figure 1 you can see some examples of problems that can be learned very efficiently using the techniques that are presented in the following sections.

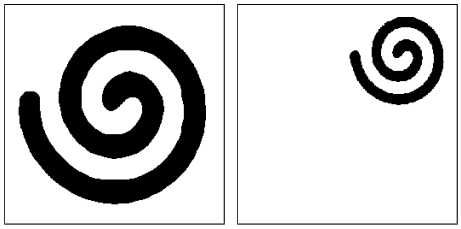


(a) (b) (c) (d) (e)

Figure 1: Examples of complex non-linear problems that can be learned efficiently.

2. Global Selective Attention - Dynamic Training Set Evolution

Consider the two problems depicted in Figure 2. Clearly, both problems are of approximately equal complexity, since they encapsulate the same image in a different scale and position within the input space. We would like to have a mechanism that will make the network capable of learning both problems at about the same speed.



(a) (b)

Figure 2: Approximately equal complexity problems.

Intuitively, the samples on the boundaries, which are the samples on positions with the highest contrast, are those that determine the complexity of each problem. During training, these samples have the property that they produce the highest errors. We thus need a method that will focus attention on samples with high error relatively to the rest. Previous work on such global selective attention has been published by many authors (Munro, 1987; Yu and Simmons, 1990; Bakker, 1992, 1993; Schapire, 1999; Zhong and Ghosh, 2000).

Of particular interest are the various boosting algorithms, such as AdaBoost (Schapire, 1999), which work by placing more emphasis on training samples that the system is currently getting wrong. Unfortunately, the most successful of these algorithms require a predefined set of samples on which training will be performed, something that is excluded from our scenario. Nevertheless, in a less constrained scenario, boosting can be applied on top of our techniques as a meta-learning algorithm, using our techniques as the base-learning algorithm.

A simple method that can provide such an adaptive selective attention mechanism, by keeping an exponential trace of the average error in the training set, is described in Algorithm 1.

$\bar{e} \leftarrow 0$
Repeat
    Pick a random sample
    Evaluate the error e for the sample
    If $e > 0.5\,\bar{e}$
        $\bar{e} \leftarrow e\,\alpha + \bar{e}\,(1-\alpha)$
        Train
    End
Until a stopping criterion is satisfied

In this report's context, *error evaluation* and *train* are defined as:
Error Evaluation: Computation of the output values by forward propagating the activations from the input to the output layer for a single sample, plus computation of the output errors. The sample's error e is set to the quadratic mean (RMS) of the output units' errors.
Train: Back-propagation of the output errors to the hidden layer and immediate weights' adjustment.

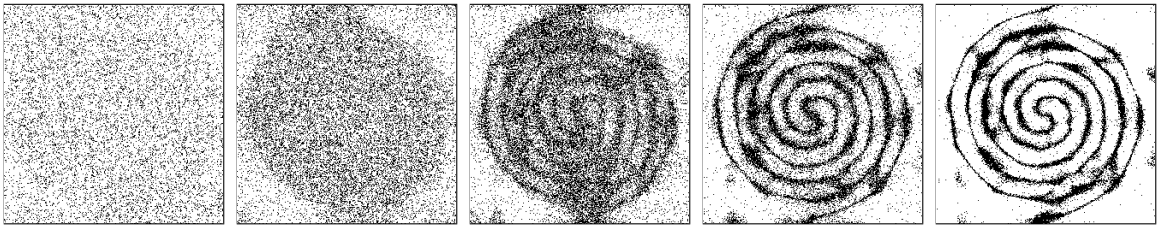
Algorithm 1: The dynamic training set evolution algorithm.

The algorithm evaluates the errors of all samples, but trains only for samples with error greater than half the average error of the current training set. Training is initially performed for all samples, but gradually, it is concentrated on the samples at the problem's boundaries. When the error for these samples is reduced, other previously excluded samples enter the training set. Thus, samples enter and leave the training set automatically, with a tendency to train on samples with high error.

The magnitude of the constant α that determines the time scale of the exponential trace is problem specific, but in all experiments in this report it was kept fixed to 10^{-4}. The fraction of 0.5 was determined experimentally to give a good balance between sample selectivity and training set size. If it is close to 0 then we train for almost all samples. If it is close to 1 then we are at risk of making the training set starve from samples. Of course, one can choose to vary it dynamically in order to have a fixed percentage of samples in the training set, or, to not allow the training percentage to fall below a pre-specified limit.

Figure 3 shows the training set evolution for the two-spirals problem (in Figure 1a) in various training stages. The network topology was 2:64:1. You can see the training set forming gradually and tracing the problem boundaries where the error is the highest.

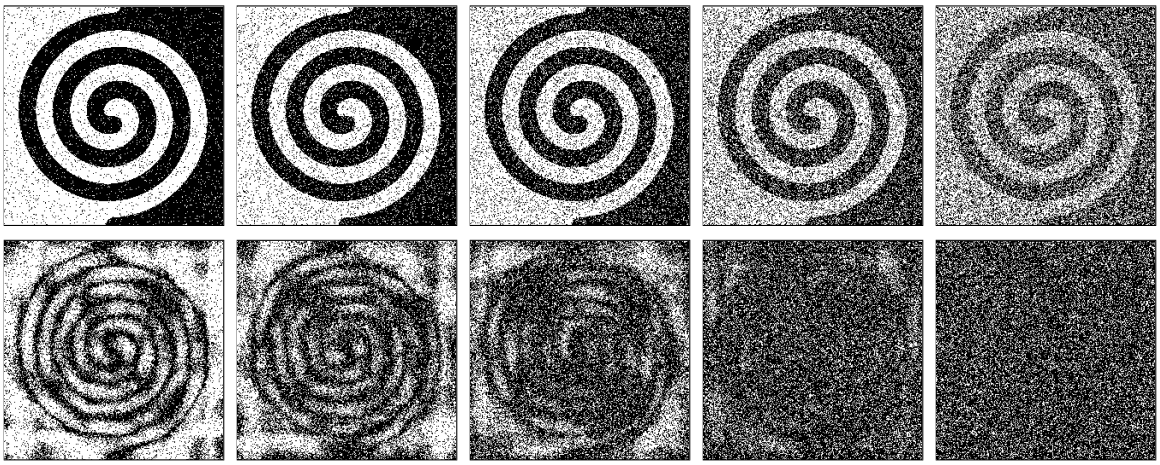
One could argue that such a process may be very sensitive to outliers. Experiments have shown that this does not happen. The algorithm does not try to recognize the outliers, but at least, adjusts naturally by not allowing the training set size to shrink. So, at the presence of heavy noise, the algorithm becomes ineffective, but does not introduce any additional harm. Figure 4 shows the two-



10000-11840-93% 20000-25755-76% 40000-66656-41% 60000-142789-22% 90000-296659-18%

Figure 3: Training set evolution for the two-spirals problem. Under each image you can see the stage of training in trains, error-evaluations and the percentage of samples for which training is performed at the corresponding stage.

spirals problem distorted by dynamic noise and the corresponding training set after 90000 trains with 64 hidden units. You can see that the algorithm tolerates noise by not allowing the training set size to shrink. It is also interesting that at noise levels as high as 30% the algorithm can still exclude large areas of the input space from training.



10%-42% 20%-62% 30%-75% 50%-93% 70%-99%

Figure 4: Top row shows the model with a visualization of the applied dynamic noise. Bottom row shows the corresponding training sets after 90000 trains. Under each pair of images you can see the percentage of noise distortion applied to the original input and the percentage of samples for which training is performed.

3. Local Selective Attention - Receptive Fields

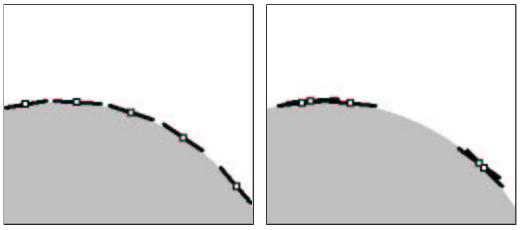
Having established a global method to focus attention on the important parts of a problem, we now come to address the main issue, which is the network training. Let first discuss the roles of the hidden and output layers in a feed forward neural network with a single hidden layer and without shortcut input-to-output connections.

The hidden layer is responsible for transforming a non linear input-to-output mapping, into a non linear input-to-hidden layer mapping, that can be mapped linearly to the output.

The output layer is responsible for learning a linear hidden-to-output mapping (which is an easy job), but most importantly, it must provide to the hidden layer error gradient information that will be used for the error credit assignment problem. In this respect, it becomes apparent that all hidden units should receive the most possibly accurate error information. That is why, we must train all hidden to output connections and back propagate the error through all these connections.

This is not the case for the hidden layer. Consider, for a classification problem, how the hidden units with sigmoidal activations partition the input space into sub areas. By adjusting the input-to-hidden weights and biases, each hidden unit develops a hyperplane that bi-partitions the input space in the most useful sense.

We would like to limit the number of hyperplanes in order to reduce the system's available degrees of freedom and obtain better generalization capabilities. At the same time, we would like to thoroughly use them in order to optimize the input output approximation. This can be done by arranging the hyperplanes to touch the problem's boundaries at regular intervals dictated by the boundary curvature, as it is shown in Figure 5a. Figure 5b, shows a suboptimal placement of the hyperplanes which causes a waste of resources. Each hidden unit must be differentiated from the others and ideally not interfere with the subproblems that the other units are trying to solve. Suppose that two hidden units are governed by the same, or nearly the same, parameters. How can we differentiate them? There are many possibilities.



(a) (b)

Figure 5: Optimal vs. suboptimal hyperplanes.

One could be, to just throw one unit away and make the output weight of the other equal to the sum of the two original output weights. That would leave the function unchanged. However, identifying these similar units during training is not easy computationally. In addition, we would have to figure out a method that would compute the best initial placement for the hyperplane of the new unit that would substitute the one that was thrown away.

Another possibility would be to add noise in the weight updates, gradually reduced with a simulated annealing schedule which should be decided before training begins. Unfortunately, the loss of initial state invariance would complicate training for unknown complex non linear problems.

To our thinking, it is much better to embed constraints into the system, so that it will not be possible for two hidden units to develop the same hyperplane. Two computationally efficient techniques to embed such constraints are described in sections 3.1 and 3.2.

Many other authors have also examined methods for local selective attention. For the related discussions see Huang and Huang (1990), Ahmad and Omohundro (1990), Baluja and Pomerleau (1995), Flake (1998), Duch et al. (1998), and Phillips and Noelle (2004).

3.1 Fixed Cascaded Inhibitory Connections

A problem with the hidden units of conventional feed forward networks is that they are all fed with the same inputs and back propagated errors and that they operate without knowing each other's existence. So, nothing prevents them from behaving identically. This lack of communication between hidden units has been addressed by researchers through hidden unit lateral connections. Agyepong and Kothari (1997) use unidirectional lateral interconnections between adjacent hidden layer units, claiming that these connections facilitate the controlled assignment of role and specialization of the hidden units. Kothari and Ensley (1998) use Gaussian lateral connections which enable the hidden decision boundaries to be global in nature but also be able to represent local structure. Numerous neural network algorithms employ bidirectional lateral inhibitory connections in order to generate competition between the hidden units. In an interesting variation described by Spratling and Johnson (2004), competition is provided by each hidden unit blocking its preferred inputs from activating other units.

We use a single hidden layer where the hidden units are considered sequenced. Each hidden unit is connected to all succeeding hidden units with a fixed connection with weight set to minus one. The hidden units get differentiated, because they receive different inputs, they produce different activations and they get back different error information. Another benefit is that they can generate higher order feature detectors, that is, the resulting hidden hyperplanes are no longer strictly linear, but they may also be curved. Considering the fixed value, -1 is used just to avoid a multiplication. Values from -0.5 to -2 give good results as well.

As it is shown in Section 5.1.1, the fixed cascaded inhibitory connections are very effective at reducing a problem's asymptotic residual error. This should be attributed to both of their abilities, to generate higher order feature detectors and to hasten the hidden units' symmetry breaking.

These connections can be implemented very efficiently with just one subtraction per hidden unit for both hidden activation and hidden error computation. In addition, the disturbance to the parallelism of the backpropagation algorithm is minimal. Most operations on the hidden units can still be done in parallel and only the final computations must be performed sequentially. We include the algorithms for the hidden activation and error computations as examples of sequential implementations. These changes can be very easily incorporated into conventional neural network code.

Hidden Activations	Hidden Error Signals
$\delta \leftarrow 0$	$\delta \leftarrow 0$
For $j \leftarrow 1 \ldots \eta$	For $j \leftarrow \eta \ldots 1$
$\quad n_j \leftarrow \delta + \vec{x} \cdot \vec{w}_j$	$\quad e_j \leftarrow \delta + \vec{r} \cdot \vec{u}_j$
$\quad h_j \leftarrow f(n_j)$	$\quad g_j \leftarrow e_j f'(n_j)$
$\quad \delta \leftarrow \delta - h_j$	$\quad \delta \leftarrow \delta - g_j$
End	End

Algorithm 2: Hidden unit activation and error computation with Fixed Cascaded -1 Connections.

3.2 Selective Training of the Hidden Units

The hidden units' differentiation can be farther magnified if each unit is not trained on all samples, but only on the samples for which it receives a high error.

We train all output units, but only the hidden units for which the error signal is higher than the RMS of the error signals of all hidden units. Typically about 10% of the hidden units are trained on each sample during early training and the percentage falls up to 2% when the network is close to the solution.

This is intuitively justified by the observation that at the beginning of training the hidden units are largely undifferentiated and receive high error signals from the whole input space. At the final stage of training, the hidden hyperplanes' linear soft decision boundaries are combined by the output layer to define arbitrarily shaped decision boundaries. For μ input dimensions, from 1 up to μ units can define an open sub-region and $\mu+1$ units are enough to define a closed convex region. With such high level constructs, each sample may be discriminated from the rest with very few hidden units. These, are the units that receive the highest error signal for the sample.

Experiments on various problems have shown that training on a fraction of the hidden units is always better (in respect to number of trains to convergence), than training all or just one hidden unit. It seems that training only one hidden unit on each sample is not sufficient for some problems (Thornton, 1992). Measurements for one of these experiments are reported in Section 5.1.1.

In addition to the convergence acceleration, the combined effect of training a fraction of the hidden units on a fraction of the samples, gives big savings in CPU usage per sample as well. This sparseness of training in respect to evaluation provides further opportunities for speedup as it is discussed in Section 4.

3.3 Centering On The Input Space

It is a well known recommendation (Schraudolph, 1998a,b; LeCun et al., 1998) that the input values should be normalized to have zero mean and unit standard deviation over each input dimension. This is achieved by subtracting from each input value the mean and dividing by the standard deviation.

For some problems, like the one in Figure 2b, the center of the input space is not equal to the center of the problem. When the input is not known in advance, the later must be computed adaptively. Moreover, since the hidden units are trained on different input samples, we should compute for each hidden unit its own mean and standard deviation over each input dimension.

For the connection between hidden unit j and input unit i we can adaptively compute the approximate mean m_{ji} and standard deviation s_{ji} over the inputs that train the hidden unit, using either exponential traces:

$$m_{ji_{(t)}} \leftarrow \beta\, x_i + (1-\beta)\, m_{ji_{(t-1)}},$$

$$q_{ji_{(t)}} \leftarrow \beta\, x_i^2 + (1-\beta)\, q_{ji_{(t-1)}},$$

$$s_{ji_{(t)}} \leftarrow (q_{ji_{(t)}} - m_{ji_{(t)}}^2)^{1/2},$$

or perturbated calculations:

$$m_{ji_{(t)}} \leftarrow m_{ji_{(t-1)}} + \beta\, (x_i - m_{ji_{(t-1)}}),$$

$$v_{ji_{(t)}} \leftarrow v_{ji_{(t-1)}} + \beta \left((x_i - m_{ji_{(t)}})\, (x_i - m_{ji_{(t-1)}}) - v_{ji_{(t-1)}} \right),$$

$$s_{ji_{(t)}} \leftarrow v_{ji_{(t)}}^{1/2},$$

where β is a constant that determines the time scale of exponential averaging, vector $\vec{x}$ holds the input values, matrix Q holds the means of the squared input values and matrix V holds the variances.

The means and standard deviations of a hidden unit's input connections are updated only when the hidden unit is trained. The result of this treatment is that each hidden unit is centered on a different part of the input space. This center is indirectly affected by the error that the inputs produce on the hidden unit.

The magnitude of the constant β is problem specific, but in all experiments in this report it was kept fixed to 10^{-3}. This constant must be selected large enough, so that the centers will rapidly move to their optimum locations, and small enough, so that the hidden units will see a relatively static view of the input space and the gradient descent algorithm will not be confused. As the hidden units jitter around their centers, we effectively train them on slightly shifted views of the input space, something that can assist generalization. We get something analogous to training with jitter (Reed et al., 1995), at no extra cost.

In Figure 6, the squares show where each hidden unit is centered. You can see that most are centered on the problem boundaries at regular intervals. The crosses show the standard deviations. On some directions the standard deviations are very small, which results in very high normalized input values, causing the hidden units to act as threshold units at those directions. The sloped lines show the hyperplane distance from center and the slope. These are computed for display purposes, from their theoretical formulas for a conventional network, without considering the effect of the cascaded connections.

For some units the hyperplanes shown are not exactly on the boundaries. This is because of the fixed cascaded connections that cause the hidden units to be not exactly linear discriminants. In the last picture you can see the decision surface of a hidden unit which is a bit curved and coincides with the class boundary although its calculated hyperplane is not on the boundary.

An observant reader may also notice that the hyperplane distances from the centers are very small, which implies that the corresponding biases are small as well. On the contrary, if all hidden units were centered on the center of the image, we would have the following problem. The hyperplanes of some hidden units must be positioned on the outer parts of the image. For this to happen, these units should develop large biases in respect to the weights. This would make their activations to have small variances. These small variances might need to be compensated by large output weights and biases, which would saturate the output units and in addition ill-condition the problems.

One may wonder if the hidden biases are still necessary. Since the centers are individually set, it may seem at first that they are not. However, the centers are not trained through error backpropagation, and the hyperplanes do not necessarily pass over them. The biases role is to drive the hyperplanes to the correct location and thus pull the centers in the corresponding direction.

The individual centering of the hidden units based on the samples' positions is feasible, because we train only on samples with high errors and only the hidden units with high errors. By ignoring the small errors, we effectively position the center of each hidden unit near the center of mass of the high errors that it receives. However, this centering technique can still be used even if one chooses to train on all samples and all hidden units. Then, the statistics interval should be differentiated for each hidden unit and be recomputed for each sample relatively to the normalized absolute error that each hidden unit receives. A way to do it is to set the effective statistics interval for hidden unit j

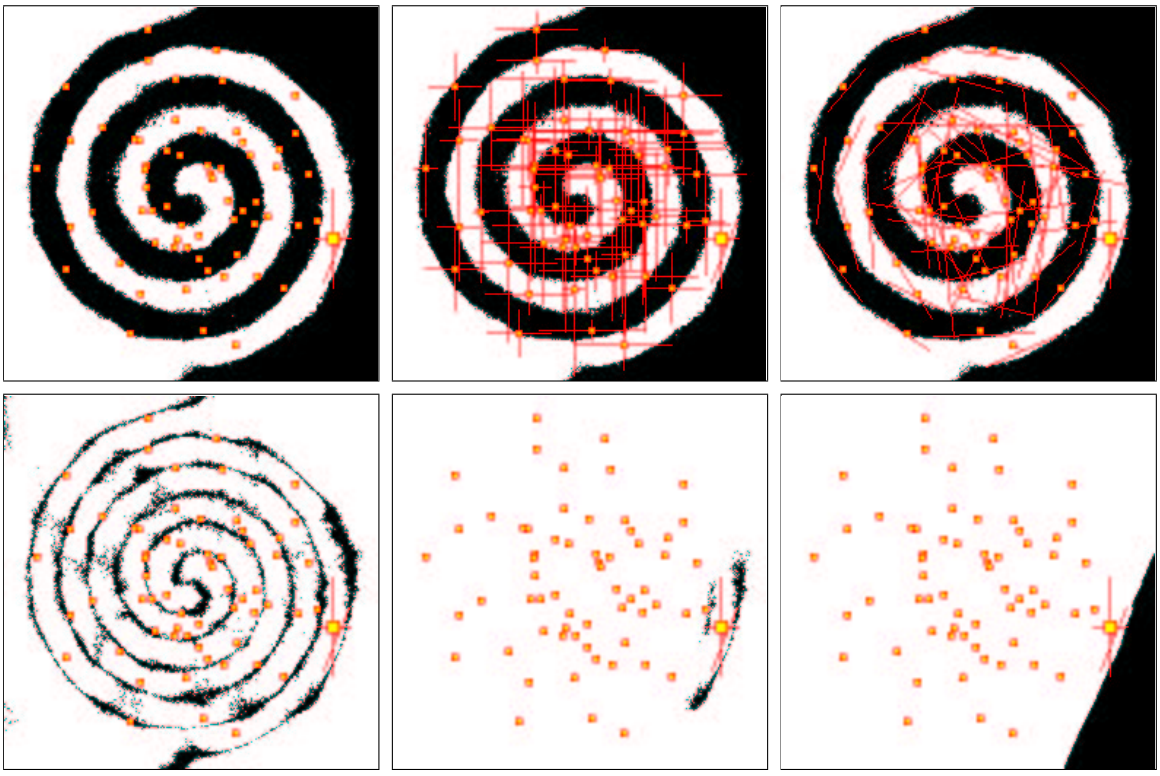


Figure 6: Hidden unit centers, standard deviations, hyperplanes, global and local training sets and a hidden unit's output. The images were captured at the final stage of training, of the problem in Figure 1a with 64 hidden units.

and sample s to:

$$\beta \frac{|e_{j,s}|}{\langle |e_j| \rangle}$$

where β is the global statistics interval, $e_{j,s}$ is the hidden unit's backpropagated error for the sample and $\langle |e_j| \rangle$ is the mean of the absolute backpropagated errors that the hidden unit receives, measured via an exponential trace. The denominator acts as a normalizer, which makes the hidden unit's mobility to be independent of the average magnitude of the errors.

Centering on other factors has been extensively investigated by Schraudolph (1998a,b). These techniques can provide further convergence acceleration, but we chose not to use them because of the additional computational overhead that they require.

3.4 A Hybrid Activation Function

As it is shown in Section 5, the aforementioned techniques enable successful training on some difficult problems like those in Figures 1a and 1b. However, if the problem contains subproblems, or put in another way, if the problem generates more than one cluster of high error density, the centering mechanism does not manage to drive the hidden unit centers to the most suitable locations. The centers are attracted by the larger subproblem or get stuck in areas between the subproblems, as shown in Figure 7.

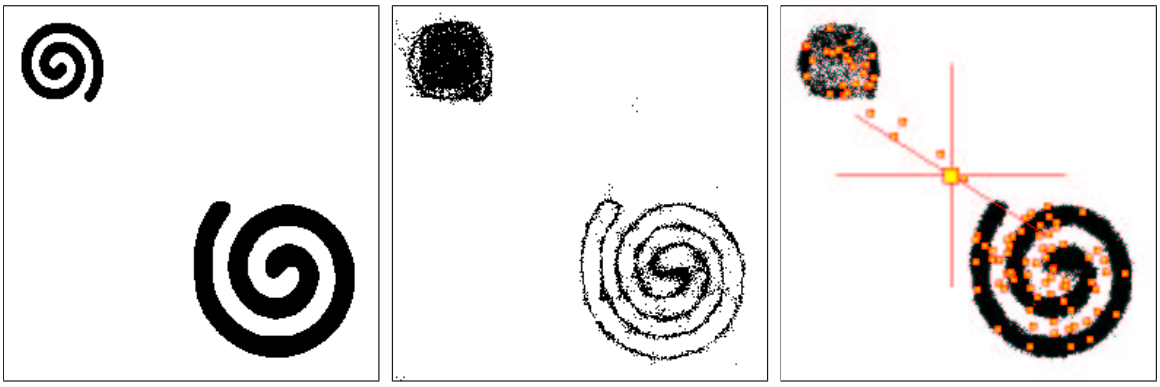


Figure 7: Model, training set, and inadequate centering

We need a mechanism that can force a hidden unit to get out of balanced but suboptimal positions. It would be nice if this mechanism could also allow the centers to migrate to various points in the input space as the need arises. It has been found that both of these requirements are fulfilled by a new hybrid activation function.

Sigmoid activations have the property that they produce hyperplanes that separate the input space globally. Our intention is to use a sigmoid like hidden activation function, because it can provide global separability, and at the same time, reduce the activation value towards zero on inputs which are not important to a hidden unit.

The Gaussian function is commonly used within radial basis function (RBF) neural networks (Broomhead and Lowe, 1988). When this function is applied to the distance of a sample to the unit's center, it produces a local response which is stronger near the center. We can then enclose the sigmoidal activation within a Gaussian envelope, by multiplying the activation with a value between 0 and 1, which is provided by applying the Gaussian function to the distance that is measured in the normalized input space.

When the number of input dimensions is large, the distance metric that must be used is not an obvious choice. Table 1 contains the distance metrics that we have considered. The most suitable distance metric seems to depend on the distribution of the samples that train the hidden units.

$\sqrt{\sum_{i=1}^{\mu} x_i^2}$	$\sqrt{\frac{1}{\mu}\sum_{i=1}^{\mu} x_i^2}$	$\sum_{i=1}^{\mu} \lvert x_i \rvert$	$\frac{1}{\mu}\sum_{i=1}^{\mu} \lvert x_i \rvert$	$\max\left(\lvert x_i \rvert\right)$
Euclidean	Euclidean Scaled	Manhattan	Manhattan Scaled	Chebyshev

Table 1: Various distance metrics that have been considered for the hybrid activation function.

In particular, if the samples follow a uniform distribution over a hypercube, then the Euclidean distance has the disturbing property that the average distance grows larger as the number of input dimensions increases and consequently the corresponding average Gaussian response decreases towards zero. As suggested by Hegland and Pestov (1999), we can make the average distance to center independent of the input dimensions, by measuring it in the normalized input space and then dividing it by the square root of the input's dimensionality. The same problem occurs for the Manhattan distance which has been claimed to be a better metric in high dimensions (Aggarwal et al., 2001). We can normalize this distance by dividing it by the input's dimensionality. A problem that

appears for both of the above rescaled distance metrics, is that for the samples that are near the axes the distances will be very much attenuated and the corresponding Gaussian responses will be close to one, something that will make the Gaussian envelopes ineffective.

A more suitable metric for this type of distributions is the Chebyshev metric whose average magnitude is independent of the dimensions. However, for reasons analogous to those mentioned above, this metric is not the most suitable if the distribution of the samples is spherical. In that case, the Euclidean distance does not need any rescaling and is the most natural distance measure. We can obtain spherical distributions by adaptively whitening them. As Plumbley (1993) and Laheld and Cardoso (1994) independently proposed, the whitening matrix Z can be adaptively computed as:

$$Z_{t+1} = Z_t - \lambda \left[\vec{z}_t \vec{z}_t^{\,T} - I\right] Z_t$$

where λ is the learning rate parameter, $\vec{z}_t = Z_t \, \vec{x}_t$ is the whitened vector and $\vec{x}_t$ is the input vector. However, we would need too many additional parameters to do it individually for each subset of samples on which each hidden unit is trained.

For the above reasons (and because of lack of a justified alternative), in the implementation of these techniques we typically use the Euclidean metric when the number of input dimensions is up to three and the Chebyshev metric in all other cases. We have also replaced the usual tanh (sigmoidal) and Gaussian (bell-like) functions, by similar functions which do not involve exponentials (Elliott, 1993).

For each hidden unit j we first compute the net-input n_j to the hidden unit (that is, the weighted distance of the sample to the hyperplane), as the inner product of normalized inputs and weights plus the bias:

$$\begin{aligned} z_{ji} &= \frac{x_i - m_{ji}}{s_{ji}}, \\ n_j &= \vec{z}_j \cdot \vec{w}_j. \end{aligned}$$

We then compute the sample's distance d_j to the center of the unit which is measured in the normalized input space:

$$d_j = \left\|\vec{z}_j\right\|.$$

Finally, we compute the activation h_j as:

$$\begin{aligned} a_j &= Elliott(n_j) = \frac{n_j}{(1+\left|n_j\right|)}, \\ b_j &= bell(d_j) = \frac{1}{(1+d_j^2)}, \\ h_j &= a_j \, b_j. \end{aligned}$$

Since d_j is not a function of $\vec{w}_j$, we treat b_j as a constant for the calculation of the activation derivative with respect to n_j, which becomes:

$$\frac{\partial h_j}{\partial n_j} = b_j \, (1 - \left|a_j\right|)^2.$$

The hybrid activation function, which by definition may only be used for hidden units connected to the input layer, enables these units to acquire selective attention capabilities on the input space. Each hidden unit may have a global or local receptive field on each input dimension. The size of this dimensional receptive field depends on the standard deviation which is computed for the corresponding dimension.

This activation makes balanced positions between subproblems to be unstable. As soon as the center is changed by a small amount, it will be attracted by the nearest subproblem. This is because the unit's activation and the corresponding error will be increased for samples towards the nearest subproblem and decreased at the other direction. Hidden units can still be centered between subproblems but only if their movement at either direction causes a large error for samples at the opposite direction, that is, if they are absolutely necessary at their current position.

Additionally, if a unit is centered near a subproblem that produces low errors and the unit is not necessary in that area, then it may migrate to other areas that still have high errors. This unit center migration has been observed in all experiments on complex problems. This may be due to the non-linear response of the bell function, and its long tails which keep the activation above zero for all input samples.

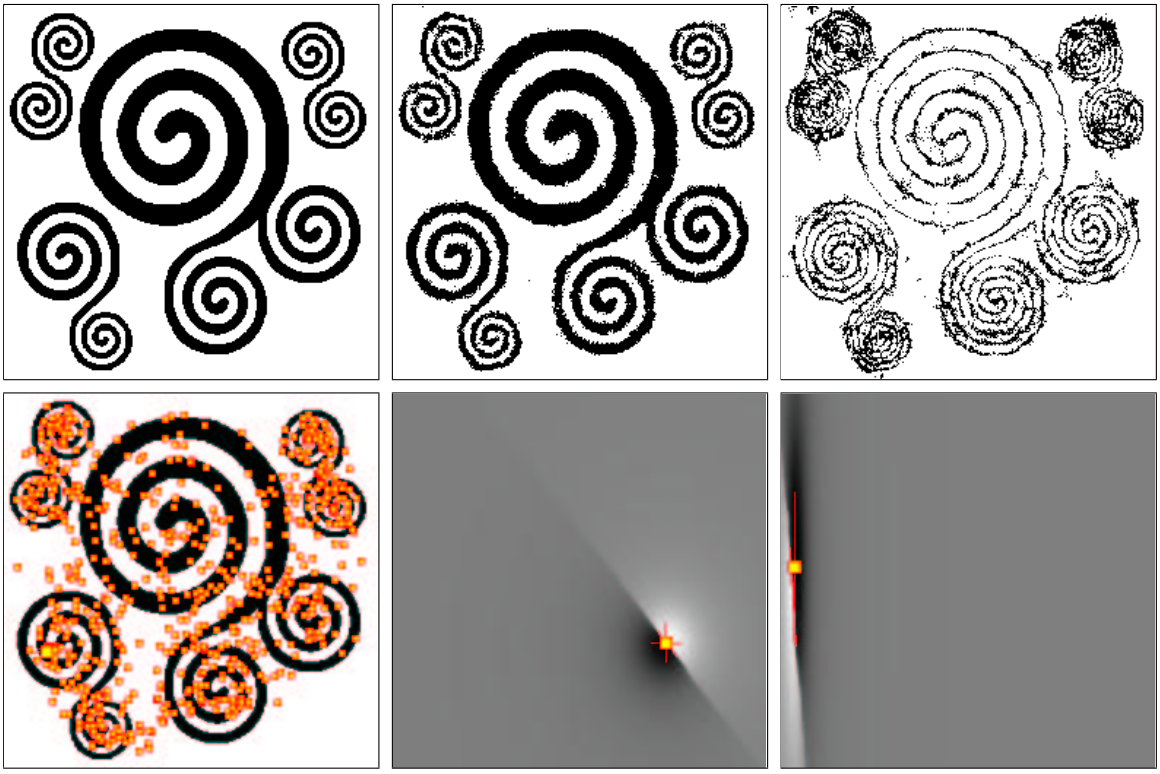


Figure 8: Model, evaluation, training set, hidden unit centers and two hidden unit outputs showing the effect of the hybrid activation function. The images were captured at the final stage of training, of the problem in Figure 1d with 700 hidden units.

In Figure 8 you can see a complex problem with 9 clusters of high errors. The hidden units place their centers on all clusters and are able to solve the problem. In the last two images, you can see the

effect of the hybrid activation function which attenuates the activation at points far from the center in respect to the standard deviation on each dimension. One unit develops a near circular local receptive field and one other develops an elongated ellipsoidal receptive field. The later provides almost global separation in the vertical direction and becomes a useful discriminant for two of the subproblems.

One may find similarities between this hybrid activation function and the Square-MLP architecture described by Flake (1998). The later, partially implements higher order neurons by duplicating the number of input units and setting the new input values equal to the squares of the original inputs. This architecture enables the hidden units to form local features of various shapes, but not the locally constrained sigmoid formed by our proposal. In contrast, the hybrid activation function does not need any additional parameters beyond those that are already used for centering and it has the additional benefit, which is realized by the local receptive fields in conjunction with the small biases and the symmetric sigmoid, that the hidden activations will have a mean close to zero. As discussed by Schraudolph (1998a,b) and LeCun et al. (1998), this is very beneficial for the output layer training.

However, there is still room for improvement. As it was also observed by Flake (1998), the orientations of the receptive field ellipses are always at the direction of one of the input axes. This limitation is expected to hinder performance when training hidden units which have sloped hyperplanes. Figure 9 shows a complex problem at the middle of training. Units with sloped hyperplanes are trained on samples whose input values are highly correlated. This can slowdown learning by itself, but in addition, the standard deviations cannot get sufficiently small and as a result the receptive field cannot be sufficiently shrunk at the direction perpendicular to the hyperplane. As a result the hidden unit's activation unnecessarily interferes with the activations of nearby units.

Although it may be possible to address the correlation problem with a more sophisticated training method that uses second order gradient information, like Stochastic Meta Descent (Schraudolph, 1999, 2002), the orientations of the receptive fields will still be limited. In Section 6.2 we discuss possible directions for further research that may circumvent this limitation.

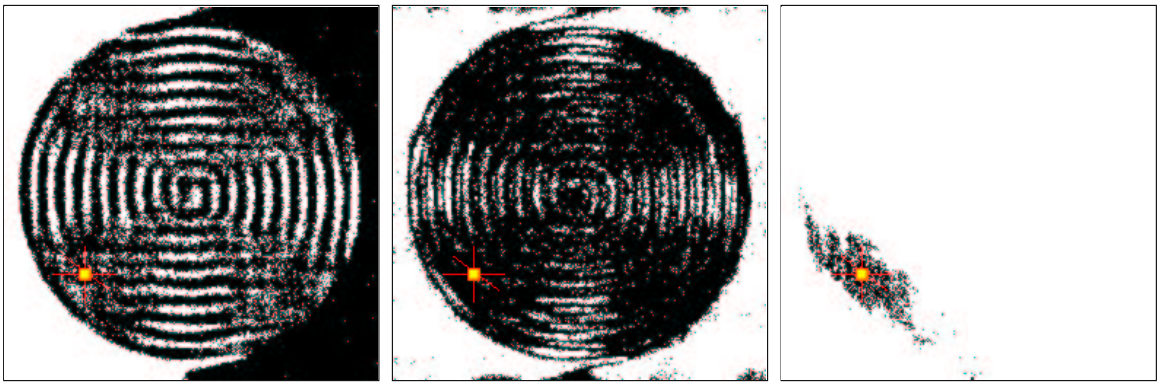


Figure 9: Evaluation and global and local training sets during middle training for the problem in Figure 1b. It can be seen that a hidden unit with a sloped hyperplane is trained on samples with highly correlated input values. Samples that are separated by horizontal or vertical hyperplanes are easier to be learned.

4. Further Speedups

In this section we first describe an implementation technique that reduces the computational requirements of the error evaluation phase and then we give references to methods that have been proposed by other authors for the acceleration of the training phase.

4.1 Evaluation Speedup

Two of the discussed techniques, training only for samples with high errors, and then, training only the hidden units with high error, make the error-evaluation phase to be the most processing demanding phase for the solution of a given problem. In addition, some other techniques, like board game learning through temporal difference methods, require many evaluations to be performed before each train. We can speedup evaluation by the following observation:

For many problems, only part of the input is changed on successive samples. For example, for a backgammon program with 200 input units (with raw board data and not any additional features encoded), very few inputs will change on successive positions. Even on two dimensional problems such as images, we can arrange to train on samples selected by random changes on the X and Y dimensions alternatively. This process of only resampling one coordinate at a time is also known as "Gibbs sampling" and it nicely generalises to more than two coordinates (Geman and Geman, 1984).

Thus, we can keep in memory all intermediate results from the evaluation, and recalculate only for the inputs that have changed. This implementation technique requires more storage, especially for high dimensional inputs. Fortunately, storage is not an issue on modern hardware.

4.2 Training Speedup

Many authors have proposed methods for speeding-up online training by using second order gradient information in order to dynamically vary either the learning rate or the momentum (see LeCun et al., 1993; Leen and Orr, 1993; Murata et al., 1996; Harmon and Baird, 1996; Orr and Leen, 1996; Almeida et al., 1997; Amari, 1998; Schraudolph, 1998c, 1999, 2002; Graepel and Schraudolph, 2002).

As it is shown in the next section, our techniques enable standard stochastic gradient descent with momentum to efficiently solve all the highly non-linear problems that have been investigated. However, the additional speed up that an accelerating algorithm can give is a nice thing to have. Moreover, these accelerating algorithms automatically reduce the learning rate when we are close to a solution (by sensing the oscillations in the error gradient) something that we should do through annealing if we wanted the best possible solution.

We use the Incremental Delta-Delta (IDD) accelerating algorithm (Harmon and Baird, 1996), an incremental nonlinear extension to Jacobs' (1988) Delta-Delta algorithm, because of its simplicity and relatively small processing requirements. IDD computes an individual learning rate λ for each weight w as:

$$\lambda(t) = e^{\xi(t)},$$

$$\xi(t+1) = \xi(t) + \theta \frac{\Delta w(t+1)}{\lambda(t)} \Delta w(t),$$

where θ is the meta-learning rate which we typically set to 0.1.

5. Experimental Results

In order to measure the effectiveness of the described techniques on various classes of problems, we performed several experiments. Each experiment was replicated 10 times with different random initial weights using matched random seeds and the means and standard deviations of the results were plotted in the corresponding figures.

For the experiments we used a single hidden layer, the cross entropy error function, the logistic or softmax activation function for the output units and the Elliott or hybrid activation function for the hidden units. Output to hidden layer weights and biases were initialized to zero. Hidden to input layer weights were initialized to random numbers from a normal distribution and then rescaled so that the incoming weights to each hidden unit had norm unity. Hidden unit biases were initialized to a uniform random number between zero and one.

The curves in the figures are labelled with a combination of the following letters which indicate the techniques that were applied:

B – Adjust weights using stochastic gradient descent with momentum 0.9 and fixed learning rate $0.1/\sqrt{c}$ where c is the number of incoming connections to the unit.

A – Adjust weights using IDD with meta-learning rate 0.1 and initial learning rate $1/\sqrt{c}$ where c is as above.

L – Use fixed cascaded inhibitory connections as described in Section 3.1.

S – Skip weights adjustment for samples with low error as described in Section 2.

U – Skip weights adjustment for hidden units with low error as described in Section 3.2.

C – Use individual means and stdevs for each hidden to input connection as described in Section 3.3.

H – Use the hybrid activation function as described in Section 3.4.

For the ‘B’ training method we deliberately avoided an annealing schedule for the learning rate, since this would destroy the initial state invariance of our techniques. Instead, we used a fixed small learning rate which we compensated with a large momentum. For the ‘A’ method, we used a small meta-learning rate, to avoid instabilities due to the high non-linearities of the examined problems. It is important to note that for both training methods the learning parameters were fixed to the above values and not optimized to each individual problem.

For the ‘C’ technique, the centers of the hidden units where initially set to the center of the input space and the standard deviations were set to one third of the distance between the extreme values of each dimension. When the technique was not used, a global preprocessing was applied which normalized the input samples to have zero mean and unit standard deviation.

5.1 Two Input Dimensions

In this section we give experimental results for the class of problems that we have mainly examined, that is, problems in two input and one output dimensions, for which we have dense and noiseless training samples from the whole input space. In the figures, we measure the average classification error in respect to the stage of training. The classification error was averaged via an exponential trace with time scale 10^{-4}.

5.1.1 COMPARISON OF TECHNIQUE COMBINATIONS

For these experiments we used the two-spirals problem shown in Figures 1a, 3, 4 and 6. We chose this problem as a non trivial representative of the class of problems that during early training generate a single cluster of high error density. The goal of this experiment is to measure the effectiveness of various technique combinations and then to measure how well the best technique combination scales with the size of the hidden layer.

Figures 10 and 11 show the average classification error in respect to the number of evaluated samples and processing cycles respectively for 13 technique combinations. For these experiments we used 64 hidden units. The standard deviations were not plotted in order to keep the figures uncluttered. Figure 10 has also been split to Figures 12 and 13 in order to show the related error bars.

Comparing the curves B vs. BL and BS vs. BLS on Figures 10 and 11, we can see that the fixed cascaded inhibitory connections reduce the asymptotic residual error by more than half. This also applies, but to a lesser degree, when we skip weight updates for hidden units with low errors (B vs. BU, BS vs. BSU). When used in combination, we can see a speed-up of convergence but the asymptotic error is only marginally further improved (BLU and BLSU).

In Figure 11, it can be seen that skipping samples with low errors can speed-up convergence and reduce the asymptotic error as well (BLU vs. BLSU). This is a very intriguing result, in the sense that it implies that the system can learn faster and better by throwing away information.

Both Figures 10 and 11 show the BLUCH curve to diverge. Considering the success of the BLSUCH curve, we can imply that skipping samples is necessary for the hybrid activation. However, the real problem, which was found out by viewing the dynamics of training, is that the centering mechanism does not work correctly when we train on all samples. A possible remedy may be to modify the statistics interval which is used for centering, as it is described at the end of Section 3.3.

BLSUC vs. BLSU shows that centering further reduces the remaining asymptotic error to half and converges much faster as well.

Comparing curve BLSUCH vs. BLSUC, we see that the hybrid activation function does better, but only marginally. This was expected since this problem has a single region of interest, so the ability of H to focus on multiple regions simultaneously is not exercised. This is the reason for the additional experiments in Section 5.1.2.

BLSUCH and ALSUCH were the most successful technique combinations, with the later being a little faster. Nevertheless, it is very impressive that standard stochastic gradient descent with momentum can approach the best asymptotic error in less than a second, when using a modern 3.2 GHz processor.

Figure 14 shows the average classification error in respect to the number of evaluated samples, for the ALSUCH technique combination and various hidden layer sizes. It can be seen that the asymptotic error is almost inversely proportional to the number of hidden units. This is a good indication that our techniques use the available resources efficiently. It is also interesting, that the convergence rates to the corresponding asymptotic errors are quite fast and about the same for all hidden layer sizes.

5.1.2 HYBRID VS. CONVENTIONAL ACTIVATION

For these experiments we used the two dimensional problem depicted in Figures 1c and 7. We chose this problem as a representative of the class of problems that during early training generate

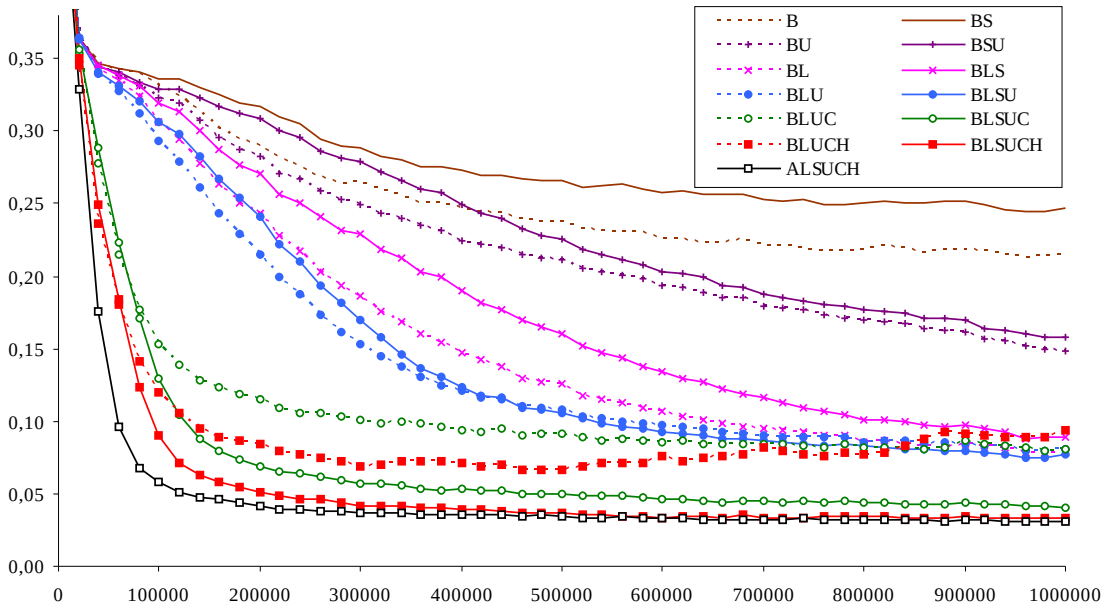


Figure 10: Average classification error vs. number of evaluated samples for various technique combinations, while training the problem in Figure 1a with 64 hidden units. The standard deviations have been omitted for clarity.

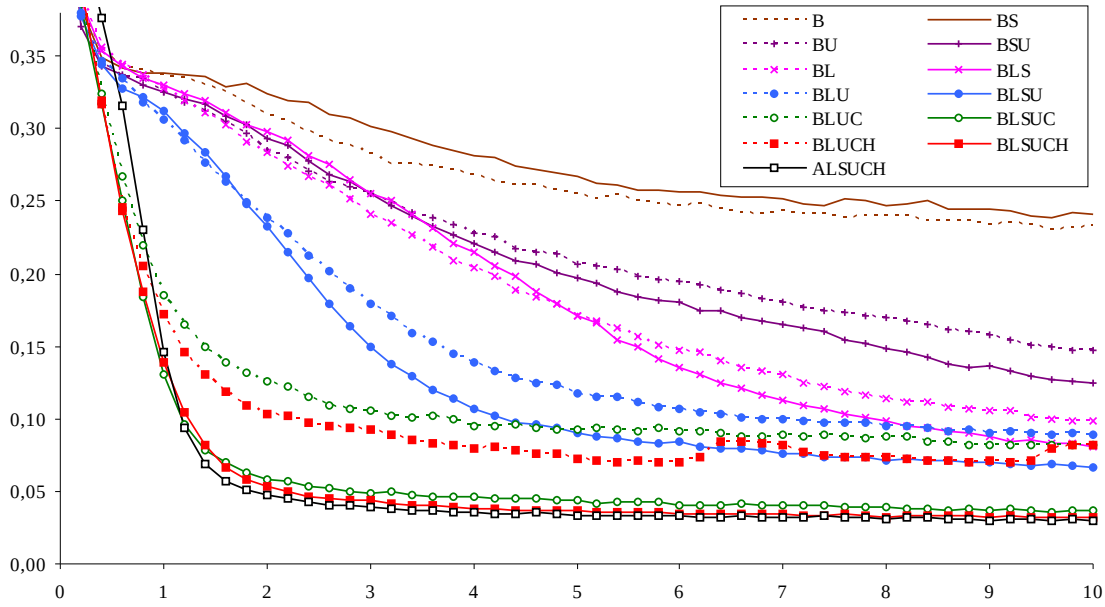


Figure 11: Average classification error vs. Intel IA32 CPU cycles in billions, for various technique combinations, while training the problem in Figure 1a with 64 hidden units. The horizontal scale also corresponds to seconds when run on a 1 GHz processor. The standard deviations have been omitted for clarity.

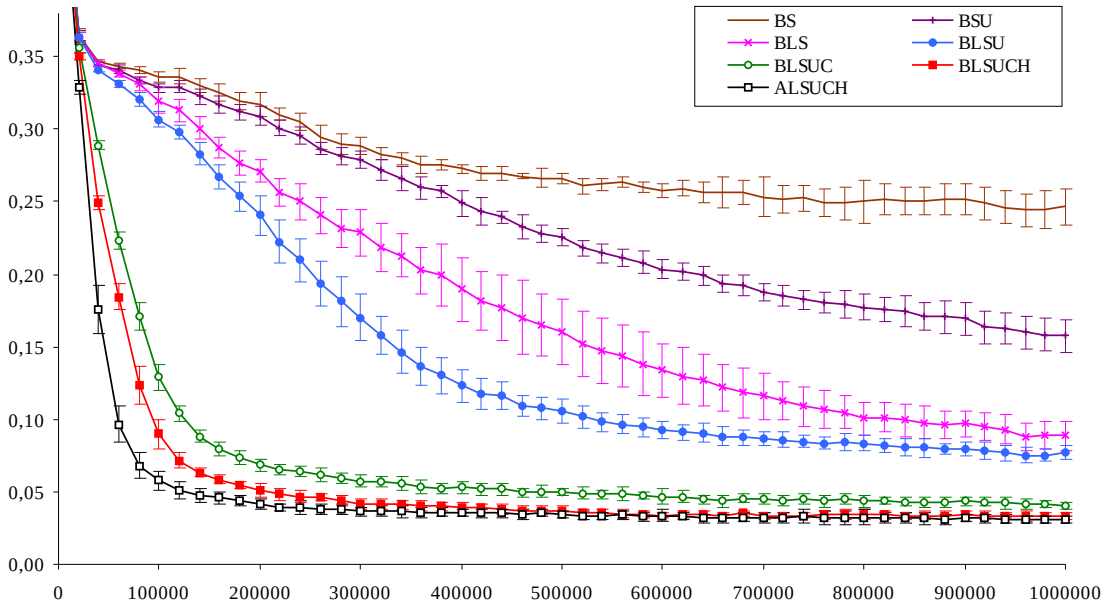


Figure 12: Part of Figure 10 showing error bars for technique combinations which employ S.

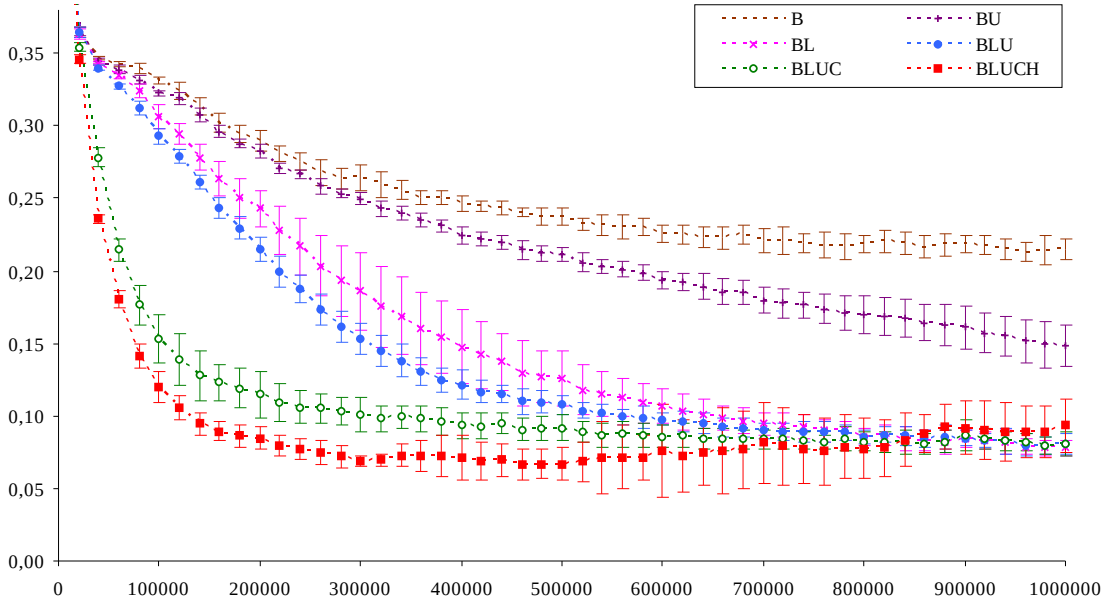


Figure 13: Part of Figure 10 showing error bars for technique combinations which do not employ S.

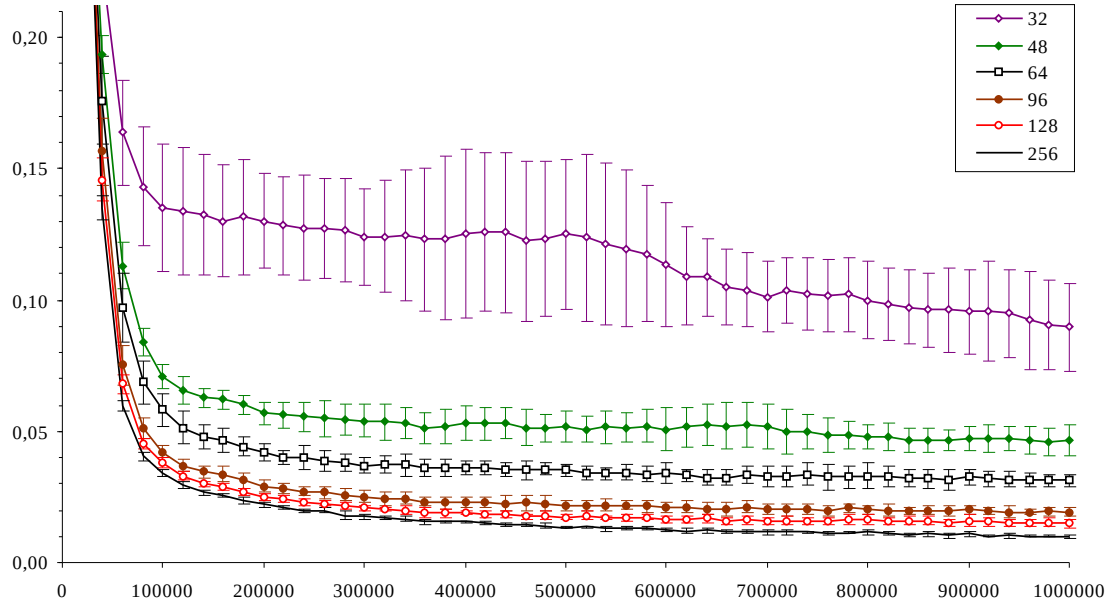


Figure 14: Average classification error vs. number of evaluated samples for various hidden layer sizes, while training the problem in Figure 1a with the ALSUCH technique combination.

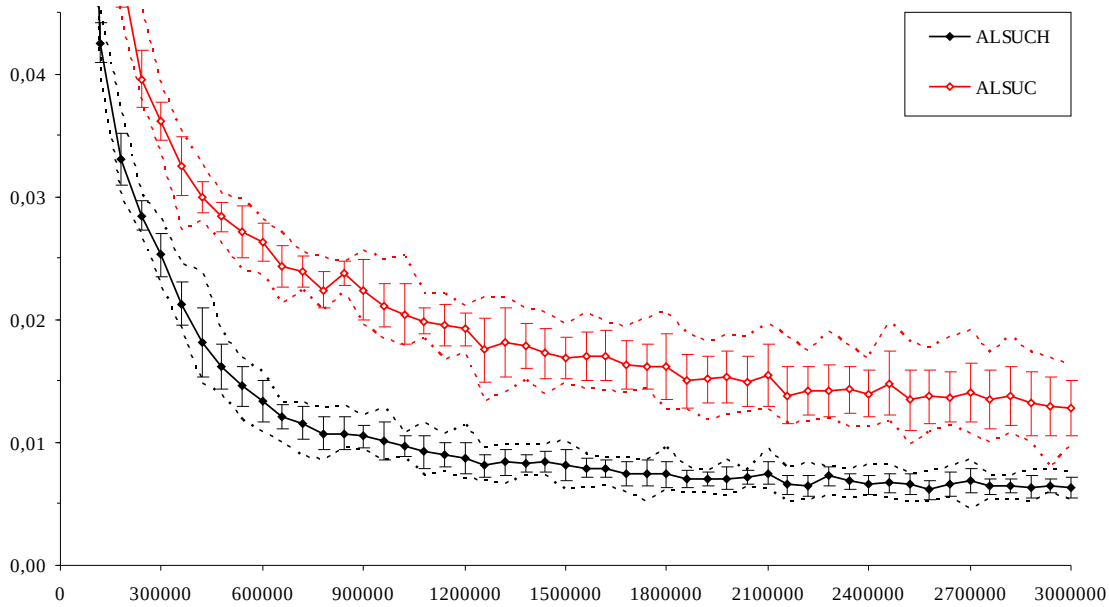
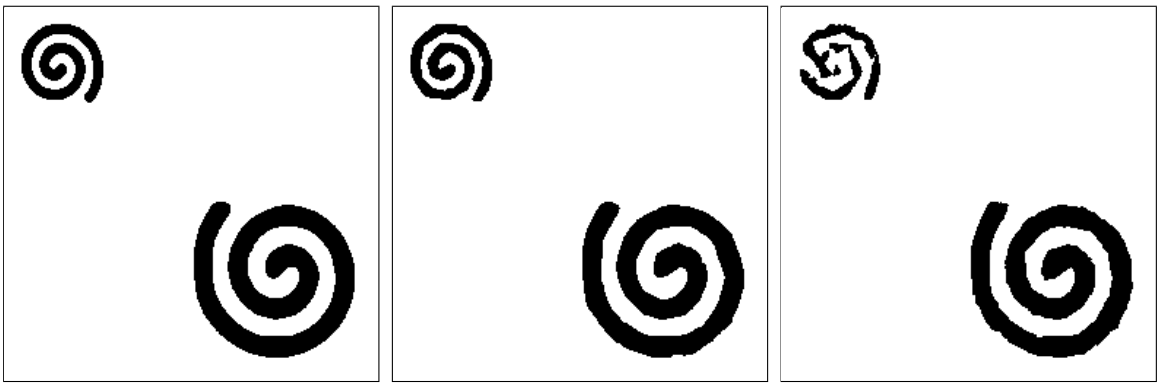


Figure 15: Average classification error vs. number of evaluated samples for the ALSUCH and ALSUC technique combinations, while training the problem in Figure 1c with 100 hidden units. The dashed lines show the minimum and maximum observed values.

small clusters of high error density of various sizes. For this kind of problems we typically obtain very small residuals for the classification error, although the problem may not have been learned. This is because we measure the error on the whole input space and for these problems most of the input space is trivial to be learned. The problem's complexities are confined in very small areas. The dynamic training set evolution algorithm is able to locate these areas, but we need much more sample presentations, since most of the samples are not used for training.

The goal of this experiment is to measure the effectiveness of the hybrid activation function at coping with the varying sizes of the subproblems. For these experiments we used 100 hidden units.

Figure 15 shows that the ALSUCH technique, which employs the hybrid activation function, reduced the asymptotic error to half in respect to the ALSUC technique. As all of the visual inspections revealed, one of which is reproduced in Figure 16, the difference in the residual errors of the two curves is due to the insufficient approximation of the smaller subproblem by the ALSUC technique.



Model ALSUCH ALSUC

Figure 16: ALSUCH vs. ALSUC approximations for a problem with two sub-problems.

5.2 Higher Input and Output Dimensions

In order to evaluate our techniques on a problem with higher input and output dimensions, we selected a standard benchmark, the Letter recognition database from the UCI Machine Learning Repository (Newman et al., 1998).

This database consists of 20000 samples that use 16 integer attributes to classify the 26 letters of the English alphabet. This problem is characterized by a medium input dimensionality and a large output dimensionality. The later, makes it a very challenging problem for any classifier.

This problem differs from those on which we have experimented so far, in that we do not have the whole input space at our disposal for training. We must train on a limited number of samples and then test the system's generalization abilities on a separate test set. Although we have not taken any special measures to assist generalization, the experimental results indicate that our techniques have the inherent ability to generalize well, when given noiseless exemplars.

An observation that applies to this problem is that the IDD accelerated training method could not do better than standard stochastic gradient descent with momentum. Thus, we report results using

the BLSUCH technique combination which is computationally more efficient than the ALSUCH technique.

For this experiment, which involves more than two output classes, we used the softmax activation function at the output layer.

Table 2 contains previously published results showing the classification accuracy of various classifiers. The most successful of them were the AdaBoosted versions of the C4.5 decision-tree algorithm and of a feed forward neural network with two hidden layers. Both classifier ensembles required quite a lot of machines in order to achieve that high accuracy.

Classifier	**Test Error %**	**Reference**
Naive Bayesian classifier	25,3	Ting and Zheng (1999)
AdaBoost on Naive Bayesian classifier	24,1	Ting and Zheng (1999)
Holland-style adaptive classifier	17,3	Frey and Slate (1991)
C4.5	13,8	Freund and Schapire (1996)
AdaBoost on C4.5 (100 machines)	3,3	Freund and Schapire (1996)
AdaBoost on C4.5 (1000 machines)	3,1	Schapire et al. (1997)
CART	12,4	Breiman (1996)
AdaBoost on CART (50 machines)	3,4	Breiman (1996)
16-70-50-26 MLP (500 online epochs)	6,2	Schwenk and Bengio (1998)
AdaBoost on 16-70-50-26 MLP (20 machines)	2,0	Schwenk and Bengio (1998)
AdaBoost on 16-70-50-26 MLP (100 machines)	1,5	Schwenk and Bengio (2000)
Nearest Neighbor	4,3	Fogarty (1992)

Table 2: A compilation of previously reported best error rates on the test set for the UCI Letters Recognition Database.

Figure 17 shows the average error reduction in respect to the number of online epochs, for the BLSUCH technique combination and various hidden layer sizes. As suggested in the database's documentation, we used the first 16000 samples for training and for measuring the training accuracy and the rest 4000 samples to measure the predictive accuracy. The solid and dashed curves show the test and training set errors respectively. Similarly to ensemble methods, we can observe two interesting phenomena which both seem to contradict the Occam's razor principle.

The first observation is that the test error stabilizes or continues to slightly decrease even after the training error has been zeroed. What is really happening is that the RMS error for the training set (which is related to the confidence of classification) continues to decrease even after the classification error has been zeroed, something that is also beneficiary for the test set's classification error.

The second observation is that increasing the network's capacity does not lead to over fitting. Although the training set error can be zeroed with just 125 hidden units, increasing the number of hidden units reduces the residual test error as well. We attribute this phenomenon to the conjecture that the hidden units' differentiation results in a smoother approximation (as suggested by Figure 5 and the related discussion).

Comparing our results with those in Table 2, we can also observe the following: The 16-125-26 MLP (5401 weights) reached a 4.6% misclassification error on average, which is 26% better than the 6.2% of the 16-70-50-26 MLP (6066 weights), despite the fact that it had fewer weights, a simpler

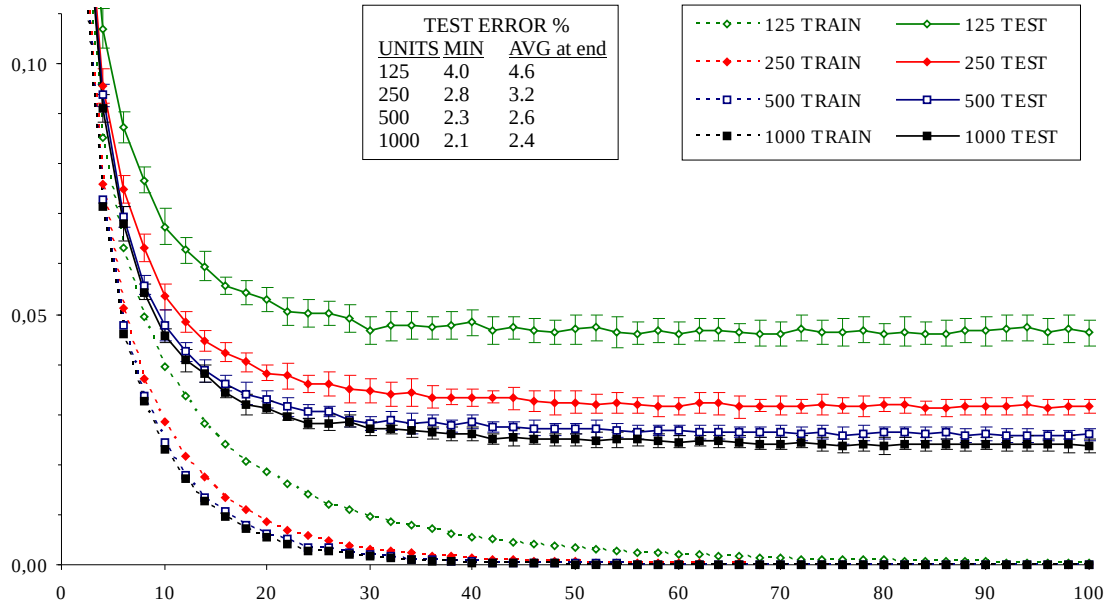


Figure 17: Average error reduction vs. number of online epochs for various hidden layer sizes, while training on the UCI Letters Recognition Database with the BLSUCH technique combination. The solid and dashed curves show the test and training set errors respectively. The standard deviations for the training set errors have been omitted for clarity. The embedded table contains the minimum observed errors across all trials and epochs, and the average errors across all trials at epoch 100.

architecture with one hidden layer only and it was trained for a far less number of online epochs. It is indicative that the asymptotic residual classification error on the test set was reached in about 30 online epochs.

The 16-1000-26 MLP (43026 weights) reached a 2.4% misclassification error on average, which is the third best published result following the AdaBoosted 16-70-50-26 MLPs with 20 and 100 machines (121320 and 606600 weights respectively). The lowest observed classification error was 2.1% and was reached in one of the 10 runs at the 80th epoch. It must be stressed that the above results were obtained without any optimization of the learning rate, without a learning rate annealing schedule and within a by far shorter training time.

All MLPs with 250 hidden units and above, gave results which put them at the top of the list of non-ensemble techniques and they even outperformed Adaboost on C4.5 with 100 machines.

Similarly to Figure 14, we also see that the convergence rates to the corresponding asymptotic errors on the test set are quite fast and about the same for all hidden layer sizes.

6. Discussion and Future Research

We have presented global and local selective attention techniques that can help neural network training to concentrate on the difficult parts of complex non-linear problems. A new hybrid activation function has also been presented that enables the hidden units to acquire individual receptive fields

in the input space. These individual receptive fields may be global or local depending on the problem's local complexities.

The success of the new activation function is due to the fact that it depends on two distances. The first is the weighted distance of a sample to the hidden unit's hyperplane. The second is the distance to the hidden unit's center. We need both distances and neither of them is sufficient. The first helps us discriminate and the second helps us localize.

The dynamic training set evolution algorithm locates the sub-areas of the input space where the problem resides. The fixed cascaded inhibitory connections and the selective training of a subset of the hidden units on each sample, force the hidden units to get differentiated and attack different subproblems. The individual centering of the hidden units at different points in the input space, adaptively conditions the network to the problem's local structures and enables each hidden unit to solve a well-conditioned subproblem. In coordination with the above, the hidden units' limited receptive fields allow training to follow a divide and conquer paradigm where each hidden unit only solves a local subproblem. The solutions to the subproblems are then combined by the output layer to give a solution to the original problem.

In the reported experiments we initialized the hidden weights and biases so that the hidden hyperplanes would cover the whole input space at random positions and orientations. The initial norm of the weights was also adjusted so that the net-input to each hidden unit would fall in the transition between the linear and non-linear range of the activation function. These specific initializations were necessary for standard backpropagation. On the contrary, we have found that the combined techniques are insensitive to the initial weights and biases, as long as their values are small. We have repeated the experiments with hidden biases set to zero and hidden weight norms set to 10^{-3} and the results where equivalent to those reported in Section 5. However, the choice of the best initial learning rate is still problem specific.

An additional and important characteristic of these techniques is that training of the hidden layer does not depend solely on gradient information. Gradient based techniques can only perform local optimization by locating a local minimum of the error function when the system is already at the basin of attraction of that minimum. Stochastic training has a potential of escaping from a shallow basin, but only when the basin is not very wide. Once there, the system cannot escape towards a different basin with a lower minimum. On the contrary, in our model some of the hidden layer's free parameters (the weights) are trained through gradient descent on the error, whereas some other (the means and standard deviations) are "trained" from the statistical properties of the back-propagated errors. Each hidden unit places its center near the center of mass of the error that it receives and limits its visibility only to the area of the input space where it produces a significant error. This model makes the hidden error landscape to constantly change. We conjecture that during training, paths connecting the various error basins are continuously emerging and vanishing. As a result the system can explore much more of the solution space. It is indicative that in all the reported experiments, all trials converged to a solution with more or less the same residual error irrespectively of the initial network state.

The combination of the presented techniques enables very fast training on complex classification problems with embedded subproblems. By focusing on the problem's details and efficiently utilizing the available resources, they make feasible the solution of very difficult problems (like the one in Figure 1e), provided that the adequate number of hidden units has been used. Although other machine learning techniques can do the same, to our knowledge this is the first report that this can be done using ordinary feed forward neural networks and backpropagation, in an online, adaptive

and memory-less scenario, where the input exemplars are unknown before training and discarded after being used.

In the following we discuss some areas that deserve further investigation.

6.1 Generalization and Regression

For the classes of problems that were investigated, we had noiseless exemplars and the whole input space at our disposal for training, so there was no danger of overfitting. Thus, we did not use any mechanism to assist generalization. This does not mean of course that the network just stored the input output mapping, as a lookup table would do. By putting constraints on the positions and orientations of the hidden unit hyperplanes and by limiting their receptive fields, we reduced the system's available degrees of freedom, and the network arranged its resources in a way to achieve the best possible input-output mapping approximation.

The experiments on the Letter Recognition Database showed remarkable generalization capabilities. However, when we train on noisy samples or when the number of training samples is small in respect to the size and complexity of the input space, we have the danger of overfitting. It remains to be examined how the described techniques are affected by methods that avoid overfitting, such as, training with jitter, error regularization, target smoothing and sigmoid gain attenuation (Reed et al., 1995). This consideration also applies to regression problems which usually require smoother approximations. Although early experiments give evidence that the presented techniques can be applied to regression problems as well, we feel that some smoothing technique must be included in the training framework.

6.2 Receptive Fields Limited Orientations

As it was noted in Section 3.4, the orientations of the receptive field ellipses are limited to have the direction of one of the input axes. This hinders training performance by not allowing the receptive fields to be adequately shrunk at the direction perpendicular to the hyperplane. In addition, hidden units with sloped hyperplanes are trained on highly correlated input values. These problems are expected to be exaggerated in high dimensional input spaces.

We would cure both of these problems simultaneously, if we could individually transform the input for each hidden unit through adaptive whitening, or, if we could present to each hidden unit a rotated view of the input space, such that, one of the axes to be perpendicular to the hyperplane and the rest to be parallel to the hyperplane. Unfortunately, both of the above transformations would require too many additional parameters. An approximation (for 2 dimensional problems) that we are currently investigating upon is the following:

For each input vector we compute K vectors rotated around the center of the input space with successive angle increments equal to $\pi/(2K)$. Our purpose is to obtain uniform rotations between 0 and $\pi/4$. Every a few hundred training steps, we reassign to each hidden unit the most appropriate input representation and adjust the affected parameters (weights, means and stdevs). The results are promising.

6.3 Dynamic Cascaded Inhibitory Connections

Regarding the fixed cascaded inhibitory connections, it must be examined whether it is better to make the strength of the connections, dynamic. Minus one is OK when the weights are small. How-

ever as the weights get larger, the inhibitory connections get less and less effective to differentiate the hidden units. We can try to make them relative to each hidden unit's average absolute net-input or alternatively to make them trainable. It has been observed that increasing the strength of these connections enables the hidden units to generate more curved discriminant functions, which is very beneficiary for some problems.

6.4 Miscellaneous

More experiments need to be done, in order to evaluate the effectiveness of the hybrid activation function on highly non-linear problems in many dimensions. High dimensional input spaces have a multitude of disturbing properties in regard to distance and density metrics, which may affect the hybrid activation in yet unknown ways.

Last, we must devise a training mechanism, that will be invariant to the initial learning rate and that will vary automatically the number of hidden units as each problem requires.

Acknowledgments

I would like to thank all participants in my threads in usenet comp.ai.neural-nets, for their fruitful comments on early presentations of the subjects in this report. Special thanks to Aleks Jakulin for his support and ideas on further research that can make these results even better and to Greg Heath for bringing to my attention the perturbated forms for the calculation of sliding window statistics. I also thank the area editor Léon Bottou and the anonymous reviewers for their valuable comments and for helping me to bring this report in shape for publication.

Appendix A. Notational Conventions

The following list contains the meanings of the symbols that have been used in this report. Symbols with subscripts are used either as scalars or as vectors and matrices when the subscripts are omitted. For example, w_{ji} is a single weight, $\vec{w}_j$ is a weight vector and W is a weight matrix.

α – A constant that determines the time scale of the exponential trace of the average training-set error within the dynamic training set evolution algorithm.

β – A constant that determines the time scale of the exponential trace of the input means and standard deviations.

δ – An accumulator for the efficient implementation of the fixed cascaded inhibitory connections.

η – The number of hidden units.

μ – The number of input units.

f – The hidden units' squashing function.

i – Index enumerating the input units.

j – Index enumerating the hidden units.

k – Index enumerating the output units.

a_j – The hidden unit's activation computed from the sample's weighted distance to the hidden unit's hyperplane.

b_j – The hidden unit's activation attenuation computed from the sample's distance to the hidden unit's center.

d_j – The sample's distance to the hidden unit's center.

e_j – The hidden unit's accumulated back propagated errors.

g_j – The hidden unit's error signal $\left(f'(n_j)\, e_j\right)$.

h_j – The hidden unit's activation.

m_{ji} – The mean of the values received by hidden unit j from input unit i.

n_j – The net-input to the hidden unit.

q_{ji} – The mean of the squared values received by hidden unit j from input unit i.

r_k – The error of output unit k.

s_{ji} – The standard deviation of the values received by hidden unit j from input unit i.

u_{jk} – The weight of the connection from hidden unit j to output unit k.

v_{ji} – The variance of the values received by hidden unit j from input unit i.

w_{ji} – The weight of the connection from hidden unit j to input unit i.

x_i – The value of input unit i.

z_{ji} – The normalized input value received by hidden unit j from input unit i. It is currently computed as the z-score of the input value. A better alternative would be to compute the vector $\vec{z}_j$ by multiplying the input vector $\vec{x}$ with a whitening matrix Z_j.

References

C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In J. Van den Bussche and V. Vianu, editors, *Proceedings of the 8th International Conference on Database Theory (ICDT)*, volume 1973 of *Lecture Notes in Computer Science*, pages 420–434. Springer, 2001.

K. Agyepong and R. Kothari. Controlling hidden layer capacity through lateral connections. *Neural Computation*, 9(6):1381–1402, 1997.

S. Ahmad and S. Omohundro. A network for extracting the locations of point clusters using selective attention. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society, MIT*, 1990.

L. B. Almeida, T. Langlois, and J. D. Amaral. On-line step size adaptation. Technical Report INESC RT07/97, INESC/IST, Rua Alves Redol 1000 Lisbon, Portugal, 1997.

S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

P. Bakker. Don't care margins help backpropagation learn exceptions. In A. Adams and L. Sterling, editors, *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, pages 139–144, 1992.

P. Bakker. Exception learning by backpropagation: A new error function. In P. Leong and M. Jabri, editors, *Proceedings of the 4th Australian Conference on Neural Networks*, pages 118–121, 1993.

S. Baluja and D. Pomerleau. Using the representation in a neural network's hidden layer for task-specific focus of attention. In *IJCAI*, pages 133–141, 1995.

L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California, 1996.

D. S. Broomhead and D. Lowe. Multivariate functional interpolation and adaptive networks. *Complex Systems*, 2(3):321–355, 1988.

W. Duch, K. Grudzinski, and G. H. F. Diercksen. Minimal distance neural methods. In *World Congress of Computational Intelligence*, pages 1299–1304, 1998.

D. L. Elliott. A better activation function for artificial neural networks. Technical Report TR 93-8, The Institute for Systems Research, University of Maryland, College Park, MD, 1993.

G. W. Flake. Square unit augmented, radially extended, multilayer perceptrons. In G. B. Orr and K. R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pages 145–163. Springer, 1998.

T. C. Fogarty. Technical note: First nearest neighbor classification on frey and slate's letter recognition problem. *Machine Learning*, 9(4):387–388, 1992.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *ICML*, pages 148–156, 1996.

P. W. Frey and D. J. Slate. Letter recognition using holland-style adaptive classifiers. *Machine Learning*, 6:161–182, 1991.

S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

T. Graepel and N. N. Schraudolph. Stable adaptive momentum for rapid online learning in nonlinear systems. In J. R. Dorronsoro, editor, *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, volume 2415 of *Lecture Notes in Computer Science*, pages 450–455. Springer, 2002.

M. Harmon and L. Baird. Multi-player residual advantage learning with general function approximation. Technical Report WL-TR-1065, Wright Laboratory, Wright-Patterson Air Force Base, OH 45433-6543, 1996.

M. Hegland and V. Pestov. Additive models in high dimensions. *Computing Research Repository (CoRR)*, cs/9912020, 1999.

S. C. Huang and Y. F. Huang. Learning algorithms for perceptrons using back propagation with selective updates. *IEEE Control Systems Magazine*, pages 56–61, April 1990.

R.A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1: 295–307, 1988.

R. Kothari and D. Ensley. Decision boundary and generalization performance of feed-forward networks with gaussian lateral connections. In S. K. Rogers, D. B. Fogel, J. C. Bezdek, and B. Bosacchi, editors, *Applications and Science of Computational Intelligence, SPIE Proceedings*, volume 3390, pages 314–321, 1998.

B. Laheld and J. F. Cardoso. Adaptive source separation with uniform performance. In *Proc. EUSIPCO*, pages 183–186, September 1994.

Y. LeCun, P. Simard, and B. Pearlmutter. Automatic learning rate maximization by on-line estimation of the hessian's eigenvectors. In S. Hanson, J. Cowan, and L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 156–163. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Mueller. Efficient backprop. In G. B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pages 9–50. Springer, 1998.

T. K. Leen and G. B. Orr. Optimal stochastic search and adaptive momentum. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Proceedings of the 7th NIPS Conference (NIPS), Advances in Neural Information Processing Systems 6*, pages 477–484. Morgan Kaufmann, 1993.

P. W. Munro. A dual back-propagation scheme for scalar reinforcement learning. In *Proceedings of the 9th Annual Conference of the Cognitive Science Society, Seattle, WA*, pages 165–176, 1987.

N. Murata, K. Müller, A. Ziehe, and S. Amari. Adaptive on-line learning in changing environments. In M. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9 (NIPS)*, pages 599–605. MIT Press, 1996.

D. J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998.

G. B. Orr and T. K. Leen. Using curvature information for fast stochastic search. In M. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9 (NIPS)*, pages 606–612. MIT Press, 1996.

J. L. Phillips and D. C. Noelle. Reinforcement learning of dimensional attention for categorization. In *Proceedings of the 26th Annual Meeting of the Cognitive Science Society*, 2004.

M. Plumbley. A hebbian/anti-hebbian network which optimizes information capacity by orthonormalizing the principal subspace. In *Proc. IEE Conf. on Artificial Neural Networks, Brighton, UK*, pages 86–90, 1993.

R. Reed, R.J. Marks, and S. Oh. Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter. *IEEE Transactions on Neural Networks*, 6(3):529–538, 1995.

R. E. Schapire. A brief introduction to boosting. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1401–1406. Morgan Kaufmann, 1999.

R. E. Schapire, Y. Freund, P. Barlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In D. H. Fisher, editor, *Proceedings of the 14th International Conference on Machine Learning (ICML)*, pages 322–330. Morgan Kaufmann, 1997.

N. N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.

N. N. Schraudolph. Centering neural network gradient factors. In G. B. Orr and K. R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pages 207–226. Springer, 1998a.

N. N. Schraudolph. Accelerated gradient descent by factor-centering decomposition. Technical Report IDSIA-33-98, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, 1998b.

N. N. Schraudolph. Online local gain adaptation for multi-layer perceptrons. Technical Report IDSIA-09-98, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Galleria 2, CH-6928 Manno, Switzerland, 1998c.

N. N. Schraudolph. Local gain adaptation in stochastic gradient descent. In *ICANN*, pages 569–574. IEE, London, 1999.

H. Schwenk and Y. Bengio. Boosting neural networks. *Neural Computation*, 12(8):1869–1887, 2000.

H. Schwenk and Y. Bengio. Training methods for adaptive boosting of neural networks for character recognition. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*. MIT Press, Cambridge, MA, 1998.

M. W. Spratling and M. H. Johnson. Neural coding strategies and mechanisms of competition. *Cognitive Systems Research*, 5(2):93–117, 2004.

C. Thornton. The howl effect in dynamic-network learning. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 211–214, 1992.

K. M. Ting and Z. Zheng. Improving the performance of boosting for naive bayesian classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 296–305, 1999.

Y. H. Yu and R. F. Simmons. Descending epsilon in back-propagation: A technique for better generalization. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 167–172, 1990.

S. Zhong and J. Ghosh. Decision boundary focused neural network classifier. In *Intelligent Engineering Systems Through Artificial Neural Networks (ANNIE)*. ASME Press, 2000.