Neural network experiences between perceptrons and support vectors

Robert P.W. Duin and Dick de Ridder
Pattern Recognition Group, Faculty of Applied Physics
Delft University of Technology, The Netherlands
duin@tn.tudelft.nl

Abstract
After years of hesitation neural networks are now widely accepted as important tools for pattern recognition. As their characteristics differ strongly from the traditional techniques, neural networks have had a great influence on the recent developments of non-neural classifiers as well. A summary will be presented of the insights and fruits resulting from the neural network hype with an emphasis on the recently developed support vector machines.

1 The Challenge

The purpose of this paper is to sketch the neural network contribution to the pattern recognition area. We will concentrate on feed-forward networks and pattern classifiers. As this is not review paper the references will be mainly restricted to a few general books and papers ([1]-[7]) that might be useful for further reading and to some contributions by the authors and others that support the private observations and evaluations of the discussed approaches.

The challenge for the pattern recognition researcher can be simply formulated as: define an automatic procedure that, given a collection of sets of different real world objects is able to find the correct set for a new object. These sets are usually called classes and their names labels. The procedure is a pattern classifier as it has to find the patterns distinguishing the classes from the given collection of examples, often referred to as the training set: a collection of object with known labels. Examples of such pattern recognition problems are the recognition of apples and pears, gothic and roman churches, or more applicable, handwritten characters, spoken words and microscopically observed biological tissues and cells. Human beings are reasonably able to learn to recognize these types of objects using relatively small sets of examples: tens to hundreds. How about an automatic procedure?

It has been realized for a long time that a single, generally applicable pattern recognition system cannot work without any context information. For that reason the abstraction has been made to represent objects by features: measurements from which an expert in the application knows that they are useful for finding the patterns. The training set is herewith reduced to a set of labeled points in a feature vector space and the pattern classifier to a discriminant function defining or separating areas for different classes in this space.

There is a major problem with this approach: for some application areas no expert can be found able to define a small set of good features. For instance, for the character recognition problem many feature types have been proposed, cumulating in thousands of dif-

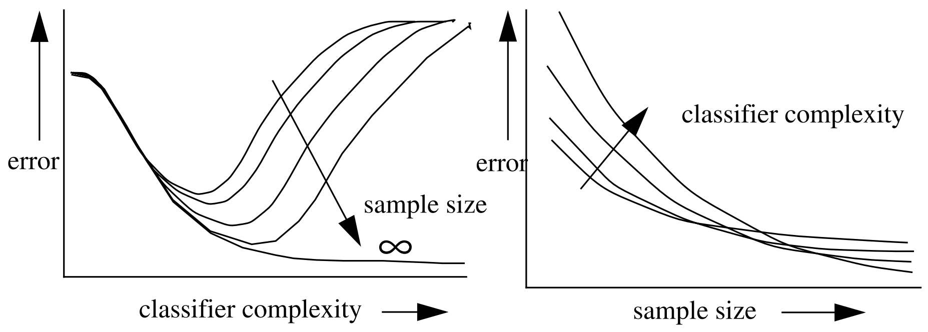


Fig. 1. The behavior of the classification error as a function of the classifier complexity (e.g. feature size or number of degrees of freedom) and the size of the training set.

ferent features. As a direct consequence the question has to be faced: how does the size of the feature space (its dimensionality) affect the generalization (performance) of a pattern classifier? If also the possibility of patterns defined by nonlinear combinations of features has to be included, the question can be reformulated as: How are the complexity and the generalization capability of a classifier related?

This question cannot be answered without considering the size of the training set as well. If we have an infinitely large set of examples the most complex classifier will describe the pattern better or equally good as more simple systems. For finite sample sizes, however, very complex classifiers may also be able to find very detailed patterns, in fact caused by the noise. These classifiers will have a bad performance. As a consequence, more simple classifiers might perform better than more complex classifiers in case of finite sample size, see fig. 1. This phenomenon is known under various names in the pattern recognition literature: peaking, the curse of dimensionality, Rao's paradox and Hughes' phenomenon.

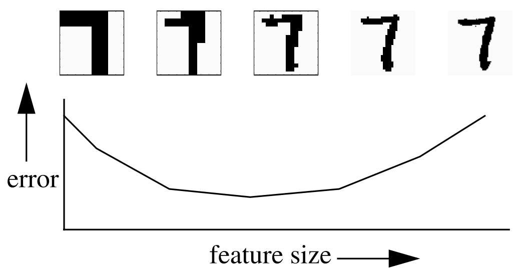


Fig. 2. Higher resolutions may result in more features and thereby in a worse performance.

For a given classifier a peaking performance can only be avoided by a sufficiently large sample size. For difficult classification problems (those that need complex classifiers) it appears sometimes to be necessary to have hundreds of thousands of training objects. More complex classifiers are caused, among others, by growing feature sizes. It remains counterintuitive that better object descriptions, e.g. more pixels caused by higher resolution images, may result in a worse performance and thereby demand more object examples, see fig 2. We will return to this point at the end of the paper.

The classifier complexity still lacks a useful definition. For classifiers that are linear in their parameters it can be defined as the number of degrees of freedom. Vapnik [4] has given a definition for nonlinear classifiers as well, which is, however, related to a worst case training set of unknown probability. Moreover, the above curves are problem dependent. So what is really needed for an optimal classifier choice is some method that connects the classifier complexity with the difficulty of the problem: the data complexity. This is still an open issue.

This approach of estimating first the desired complexity (which includes the choice of feature set) finally aims at an automatic procedure that chooses for a given problem the best classifier (in some sense) and thereby constitutes a 'super classifier', which has to be used for all pattern classification problems. Herewith the question is raised why we cannot build directly a classifier that automatically adapts itself to the optimal complexity.

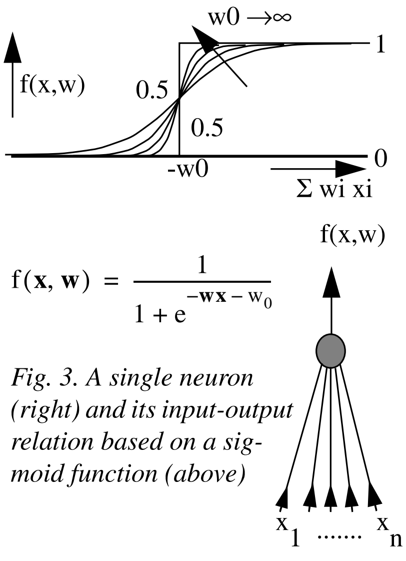
This point of view makes it necessary to reconsider the traditional optimization criteria for classifiers. The two most frequently used are the *error probability* and the *mean square error*. The first is based on some integration of the classification error and is directly related to what is considered as the performance of the final classifier. The second is based on distance measures, which are usually easier to calculate and to optimize but which are only indirectly related to the classifier performance. Both measures, however, do not include any complexity measure. Thereby they are not protected against peaking during the optimization of a classifier with an adjustable complexity. A possibly third approach is the minimization of the *description length*, which seems directly related to the complexity. For a straight forward use of this approach it is necessary to fix the classification error.

In any discussion on comparing and selecting classifiers it is necessary to realize that the overall performance of a classifier depends on the set of problems that is considered. No classifier will be the best on any problem or on any set of problems. So it is necessary to make a choice on a wide set of problems. See [2] for an example of such a study. In [11] the problem of comparing classifier is discussed more extensively.

In the remainder of this paper the feed-forward neural network classifier is discussed in the light of the above considerations. It will be summarized what can be learnt from this classifier and these insights will be applied to the extension of traditional classifiers and the development of new classifiers.

2 Neural Network Properties

The feed-forward neural networks that are discussed here are configured by single neurons such as shown in fig. 3. Their inputs are linearly weighted and summed. The result is mapped by a squashing function on the (0,1) interval. Note that for very small weights this function is almost linear and that for very large weights it almost resembles the step function.



$$f(\mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{wx} - w_0}}$$

Fig. 3. A single neuron (right) and its input-output relation based on a sigmoid function (above)

An entire neural network with a single layer of hidden neurons is shown in fig. 4. It has an input for each feature and an output for each class. Classification is done by assigning the label of the class with the highest output to the incoming feature vector. The number of hidden neurons can be freely chosen and determines the maximum nonlinearity that can be reached. Almost any decision surface can be constructed for sufficiently large sets of neurons.

It is important to distinguish the architecture of a neural network from the way it is trained. It has been shown in literature that almost any classifier can be mapped on a neural network. This shows that the architecture is very general and it is thereby not specific. Consequently, from a scientific point of view a neural network implementation of an otherwise trained classifier is thereby not of interest, it does not specifically contribute to a better generalization.

The real interesting point is therefor the way neural networks are trained. If this makes use of the specific architecture, we will call it a neural network classifier. If the training is done otherwise it belongs to the large class of non-neural classifiers and such methods are not considered in this section. Specific neural network training procedures train the network as a whole and iteratively minimize some error criterion based on the network outputs and their targets (derived from the class labels).

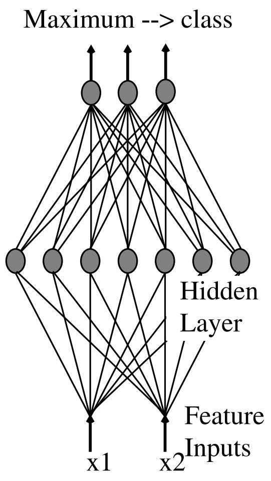


Fig. 4. Feedforward network for 2 features and 3 classes

Traditionally the MSE criterion is minimized using a gradient descent technique resulting in the well-known error back-propagation training rule. This rule is very slow and only feasible by the heavily improved computer speed in the last decade. Methods that use the second derivative in one way or another (conjugated gradient, Levenberg-Marquardt) can be much faster but might also yield instable results.

A number of observations can be made on the properties of neural network classifiers. They are given here without much argumentation:

1. The backpropagation rule and its derivatives very rarely finds the global minimum. The computational effort as well as the large number of local minima in the error landscape are prohibitive.
2. For almost all networks with more than one or two hidden neurons the global minimum is undesirable as an unrestricted neural network classifier has a too large complexity for most problems.
3. It can thereby be concluded that the traditional neural network classifiers work because of the lousy performance of the backpropagation rule. Put it more friendly: this rule has sufficiently built in regularization possibilities, most prominently its large computer demand resulting in early stopping by an impatient analyst.
4. One of the most important reasons why the optimization of complex nonlinear machines like neural networks works, is due to the tradition of starting with small weights. This corresponds with linear neurons (see fig. 3) and thereby with an entire linear neural network. Consequently, in the first step an optimal linear solution is approximated, thereafter, due to the growing weights, the network represents a moderate nonlinearity. After that it stops at one of the nearest local minima, see [9] for a more extensive discussion.

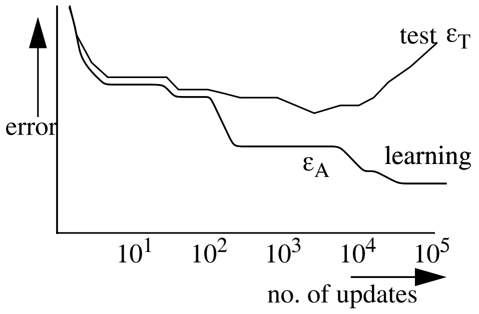


Fig. 5. A neural networks classifier shows increasingly more overtraining for more updates due to an increasing effective classifier complexity.

5. The above may also be formulated as: a neural network shows an increasing complexity as a function of the training effort. This is illustrated by the behavior of the error as a function of the training time, see fig. 5 compared with fig. 1.
6. Faster neural methods have less built in regularization and should therefor be more explicitly protected against overtraining, e.g. by the addition of noise or by weight decay.
7. Most neural networks are redundant: they have more neurons than needed for implementing the final classifier. This redundancy, however, helps during the training procedure.
8. The neural network solution is hardly ever the best classifier for a given problem. However, very often they do reasonably well. It has to be realized that due to the very large set of possible neural network solutions, results highly depend on the skills of the analyst. A unique neural network classifier (like the well defined 1-nearest neighbor rule) does not exist, see [11].

3 Further Observations

One of the most important lessons that can be extracted from the neural network invasion that entered the pattern recognition area is that it appears to be possible to train large, complex machines that often have more degrees of freedom than the size of the training set. The widely accepted dogma, based on the work by Cover, that one should never do this has therefore to be reconsidered.

A simplified version of the traditional argumentation is that a linear classifier in R_k, having k degrees of freedom, is always able to separate perfectly an arbitrarily labeled training set of $n \leq k$ training samples and has thereby no generalization capability. Computations by Cover [8] and later by Vapnik [4] show that $n >> k$, e.g. $n = 10k$, in order to get some generalization. In these studies it is not taken into account that in many applications it is known on apriori grounds that the classes are reasonably well separable. On the same arguments as given for the linear classifier, the more complex and nonlinear neural network classifier should not be used in small sample size situations, but appeared to give good results in practice.

Stimulated by these observations the first author started to reinvestigate the linear classifier for $n < k$. One of the first things that could be observed is that an adaptation of the traditional Fisher's Linear Discriminant using a pseudo-inverse technique (the PFLD) yields a surprising result for the separation of two 30-dimensional Gaussian distributions [10], see fig. 6. The mean error peaks for $n = k = 30$. On the left side of this peak it pays off to throw away a part of the training set at random. If this data set reduction is done more systematically using some heuristic strategy (the Small Sample Size Classifier, SSSC, [10]) an even better result can be obtained. This behavior can be understood by realizing that for $n < k$ the data is situated in a linear subspace, such that there is a preference for directions with large variances. If these directions coincide with the class differences, the additions of more samples, and thereby more feature directions will mainly introduce more noise and will yield worse results. A series of experiments has shown that the above behavior is not an artifact of the artificial example that was used, but also exists in real world problems. A mathematical analysis of the phenomenon is undertaken by Raudys and Duin [14].

It should be noted that in the step from PFLD to SSSC a shift in criterion has been made. Fisher's discriminant is based on a MSE criterion. In case of the PFLD this criterion is still used but yields a zero MSE as the number of parameters (the feature size) is larger than the sample size. The SSSC now heuristically minimizes the number of objects in the training set under the condition that still a zero MSE is obtained. So this follows a minimum description length (MDL) strategy.

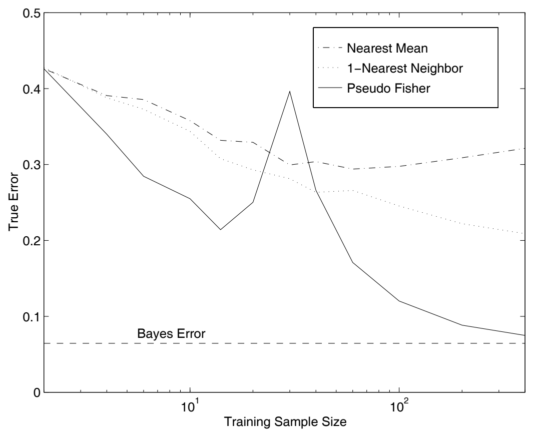


Fig. 6. The averaged error (50 runs) of some classifiers for a 30-dimensional Gaussian problem.

More systematically this is done by a recent proposal by Vapnik [5], the Support Vector Classifier (SVC). If the training set is given by a n×k matrix X the weight vector w for a linear classifier C(y) for a new object vector y can be written as a linear combination of the original training set:

$$C(y) = wy^T = \alpha X y^T \tag{1}$$

The MSE solution of α for a given set of class labels (output targets) λ is:

$$\alpha = \lambda^T (XX^T)^{-1}\text{, so} \tag{2}$$

$$C(y) = \lambda^T (XX^T)^{-1} X y^T = \lambda^T K_{xx}^{-1} K_{xy} \tag{3}$$

in which K_{xy} stands for the matrix of inner products between X and Y (or y). The FLD and the PFLD can be written like this. If n <= k the MSE is zero. In the SVC the training set X is reduced to a minimal set. Vapnik [5] uses a quadratic programming technique which allows also nonzero MSE solutions and thereby training set sizes n > k. He also shows that by replacing K by K^p a polynomial classifier is obtained having the same complexity as the linear one!. This is due to the fact that the length of the parameter vector α equals to size of the support set.

In [12] and [13] we proposed to replace the inner product matrix by an arbitrary similarity matrix based on direct measurements of object (dis)similarities, resulting in a featureless approach to pattern recognition. This procedure seems to be a good candidate to meet the challenge presented in the first section: higher image resolutions now result in more accurate computations (of object similarities) only. The classifier itself doesn't change. As a result of the fixed parameter size there are no problems with an increasing complexity.

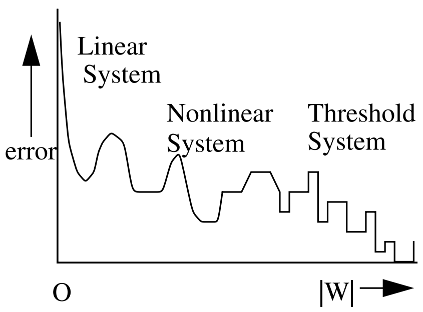


Fig. 7. Neural network error behavior as a function of the total weight size.

At this point in our discussion we report an interesting perceptron study recently undertaken by Raudys [15]. His aim is to get a better understanding of the neural network learning behavior. In [9] we summarized this using fig 7. Due to the small weight initialization a neural network starts as a linear system, for growing weights the nonlinear part of the sigmoid function is used, see fig 3, and for very large weights the sigmoid behaves as a threshold function that just counts the errors. Some of Raudys' observations will now be summarized and commented.

In order to study the linear initialization better, targets of 0.4 and 0.6 are used. Raudys proves formally and experimentally that by this the perceptron has a MSE behavior and approaches Fisher's Linear Discriminant. If the targets are released to extreme values, e.g. 0.001 and 0.999 the outputs are allowed to use the nonlinear part of the sigmoid. As a result a minimum error probability behavior is found for just weakly overlapping classes: the perceptron approaches the linear classifier that minimizes the apparent error.

If training is proceeded further the perceptron becomes insensitive for the most remote objects and training becomes determined by the most nearby objects only. We suggest that if at this point the weights of the most remote objects are set to zero, the perceptron might be used to find a support vector classifier. So now we have a MDL solution. In the next section an experimental examples is given. Raudys concludes that the target value is a very important regularization parameter, in combination with the training time. It has to be studied further how Raudys' results should be extended to the nonlinear case for neural networks with more layers.

A remarkable conclusion can be drawn from Raudys' study. A neural network is not just able to implement almost any classification function, but it can also be trained with various strategies. We add to this conclusion that this neural network ability may only be judged positively if it appears to be possible to formulate explicit rules how this has to be done for an arbitrary problem. Otherwise this neural network ability remains hidden in the skills of the analyst.

4 Experiments

In this section a number of character recognition experiments will be presented that illustrate the possibilities of some of the above discussed classifiers. In particular examples will be given that support the following observations:

1. The nearest neighbor classifier is still alive and doing well, even in comparison with the most advanced neural network procedures.
2. Support vector classifiers may perform very well.
3. Perceptrons are a good candidate for fast training of support vector classifiers.

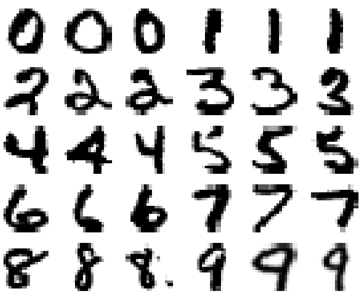


Fig. 8. Some examples of the used 16x16 character representations.

These experiments are partially published before [16], [17]. They are based on one of the NIST databases [18] from which we extracted 1250 samples for each of the ten handwritten numerals 0, 1,..., 9. In the raw data the characters are represented in binary 128 x 128 images. We used the normalization software supplied with this dataset and applied it for position, size, angle and line-width, resulting in 16 x 16 gray value images. See fig 8 for some examples. The data was split into a fixed set of 250 characters per class for training and 1000 characters per class for testing. All experiments were run only once, using constant subsets of 5 until 250 characters per class.

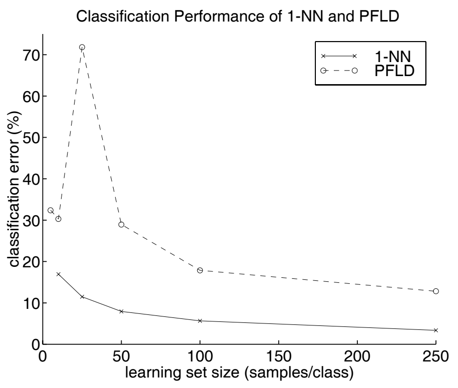


Fig. 9. Performance of the nearest rule (1-NN) and the pseudo-Fisher liner discriminant.

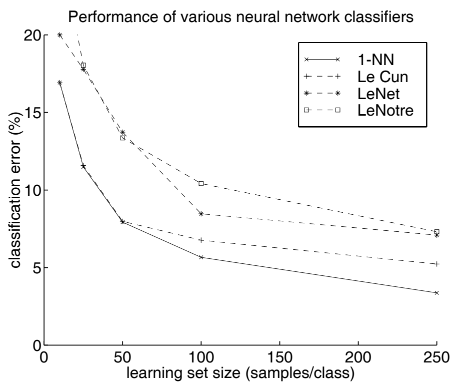


Fig. 10. Various neural networks compared with the nearest neighbor rule.

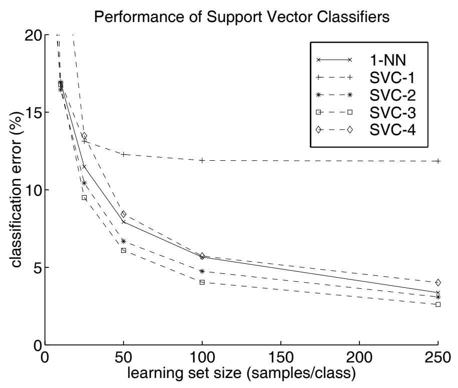


Fig. 11. Original support vector classifier compared with the nearest neighbor rule.

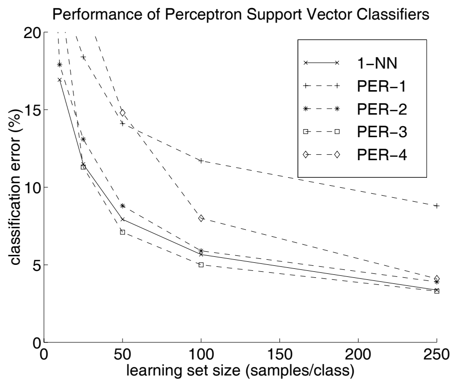


Fig. 12. Perceptron support vector classifier compared with nearest the neighbor rule.

As a reference we computed first the (pseudo) Fisher's linear discriminant (PFLD) and the nearest neighbor rule (1-NN), see fig. 9. The PFLD shows the peaking behavior as discussed before, see fig. 6, for a sample size (10×25) that is about the feature size (256). The nearest neighbor rule performs much better, indicating that nonlinear classifiers might be useful. In the following figures the 1-NN error curve is repeated as a reference. For the neural network experiments so called shared weights networks have been applied known as "LeCun" (in our implementation 1361 neurons, 63660 connections and 9760 weights and biases), see [19], its extension "LeNet" (4634, 94952, 6434), see [20] and the much smaller "LeNotre" (394, 2210, 764), see [21]. Results are shown in fig. 10. For these sample sizes the 1-NN rule appears to perform better.

The training of these large neural networks is computationally very heavy. Moreover, this also holds for the application of a neural network with many weights for the recognition of new objects. On both points SVC's might perform better. In our experiments, however, the training of a SVC using the by Vapnik proposed quadratic programming technique [5] took about 10 days on a Sun 200MHz Ultra-2 system for 250 objects per class. On the other side, the resulting performances are very promising, see fig. 11, where we show the results for classifiers upto degree 4.

Stimulated by the performance of Vapnik's support vector technique and by Raudys' observations on the general possibilities of the perceptron we developed a perceptron technique for optimizing the SVC. For initialization the nearest mean classifier is used. As targets we set 0.001^{d}, in which d is the degree of the classifier and we trained for just 1000 epochs using batch training with step size 0.001. The computational effort is herewith reduced to about 10% compared with the quadratic programming technique. Performances, shown in fig. 12, are for large sample sizes just slightly worse or even better.

5 Discussion

Neural networks have highly stimulated the development of pattern classifiers. Their architecture supports the implementation of almost any classification function. The wide set of possibilities to train them allow for almost any training strategy. Even the most degenerated neural network, the single perceptron, can be trained according various strategies like the minimization of the probability of error, the mean square error and the description length by just playing with targets, step sizes and by neglecting small weights.

What has been learned from studying neural network training and behavior is that it is possible to handle very large nonlinear machines using a large set of regularization tools like early stopping, weight decay and noise injection. By this it is possible that neural networks much larger than necessary for the problem still find good solutions. So it is not a necessary condition for obtaining generalization to have a sufficiently small machine that by its nature damps all noise. By studying neural networks it can be observed that during training the influence of remote objects is diminished. Inspired by this we developed a support vector perceptron technique. The initial experiments reported here are stimulating. Support vector methods in general should be welcomed for their small dependency on the feature size. They are thereby expected to suffer hardly or not at all from the peaking phenomenon for increasing resolutions as shown in fig. 2.

The performance of neural network techniques themselves are always somewhat disappointing in relation with their computational effort and compared with more dedicated techniques and with its eternal competitor, the nearest neighbor rule.

We have now some powerful possibilities for attacking the very small sample size problem (sample sizes smaller than the dimensionality). On the other side, the large sample size problem certainly needs some attention: Researcher like Le Cun [20] use larger and larger networks in order to be able to use all separation possibilities offered by the data. Support vector machines have difficulties to handle large datasizes simultaneously for nonlinear classifiers. A solution might be the use of combining classifiers trained by moderate sample sizes. This technique is widely studied now and it may be a fruitful option for training support vector machines by large data sizes.

6 Acknowledgment

This work is partially supported by the Foundation for Applied Sciences (STW), the Foundation for Computer Science in the Netherlands (SION) and the Dutch Organization for Scientific Research (NWO).

The line of research described here is initiated during the author's stay at VSSP group of the EE Department of the University of Surrey. Prof. Josef Kittler and colleagues are gratefully acknowledged for their hospitality and for offering a stimulating environment.

7 References

[1] B.Cheng and D.M. Titterington, *Neural Networks: A review from statistical perspective*, Statistical Science, vol. 9, no. 1, pp. 2-54, 1994.

[2] D. Michie, D.J. Spiegelhalter and C.C. Taylor, *Machine learning, neural and statistical classification*, Ellis Horwood, 1994.

[3] B.D. Ripley, *Neural networks and related methods for classification*, Journal of the Royal Statistical Society B, vol. 56, no. 3, pp. 409-456, 1994.

[4] V. Vapnik, *Estimation of Dependences based on Empirical Data*, Springer, 1982.

[5] V.N. Vapnik, *The nature of statistical learning theory*, Springer Verlag, Berlin, 1995.

[6] C.M. Bishop, *Neural networks for pattern recognition*, Clarendon, 1995.

[7] B. Ripley, *Pattern Recognition and Neural Networks*, Cambridge Univ. Press, 1996.

[8] T.M. Cover, *Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition*, IEEE Trans. EC, EC-14, 1965, 326-334.

[9] R.P.W. Duin, *Superlearning capabilities of neural networks*?, SCIA'93, Proc. of the 8th Scandinavian Conf. on Image Analysis, Tromsø, Norway, 1993, 547-554.

[10]R.P.W. Duin, *Small sample size generalization*, SCIA'95, Proc. 9th Scandinavian Conf. on Image Analysis, Volume 2, 1995, 957-964.

[11]R.P.W. Duin, *A note on comparing classifiers*, Pattern Recognition Letters, vol. 17, no. 5, 1996, 529-536.

[12]R.P.W. Duin, D. de Ridder, and D.M.J. Tax, *Featureless Classification*, First Int. Workshop Statistical Techn. in Pattern Recognition, Prague, 1997, 37-42.

[13]R.P.W. Duin, D. de Ridder, and D.M.J. Tax, *Experiments with object based discriminant functions; a featureless approach to pattern recognition*, Pattern Recognition Letters, 1997 (accepted).

[14]S. Raudys and R.P.W. Duin, *On expected classification error of the Fisher Linear Classifier with pseudo-inverse covariance matrix*, Pattern Recognition Letters, 1997, (submitted).

[15]S. Raudys, *Evolution and Generalization of a Single Neurone. I. Single Layer Perceptron as Seven Statistical Classifiers*, Neural Computation, 1997 (accepted)

[16]D. de Ridder, *Shared weights neural networks in image analysis*, Master Thesis, Delft University of Technology, 1996, February, 1-152.

[17]D. de Ridder, A. Hoekstra, and R.P.W. Duin, *Feature extraction in shared weights neural networks*, Proc. ASCI'96, Lommel, Belgium, 1996, 289-294.

[18]C.L. Wilson, M.D. Marris, *Handprinted character database 2*, april 1990. National Institute of Standards and Technology; Advanced Systems division.

[19]Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel, *Backpropagation applied to handwritten zip code recognition*, Neural Computation, vol. 1, 1989, 541-551.

[20]Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel, *Handwritten digit recognition with a back-propagation network*, in: Touretzky, D. (eds.), Advances in Neural Information Processing Systems 2, Morgan Kaufmann Publishers, San Mateo, CA., 1990.

[21]E. Viennet, *Architectures Connexionistes Multi-Modulaires, Application a l'Analyse de Scene*, Ph.D. Thesis, Université de Paris-Sud, Centre d'Orsay, 1993.