Similarity Metric Learning for a Variable-Kernel Classifier

David G. Lowe

Computer Science Department
University of British Columbia
Vancouver, B.C., V6T 1Z4, Canada
E-mail: lowe@cs.ubc.ca

November 25, 1993

Abstract

Nearest-neighbour interpolation algorithms have many useful properties for applications to learning, but they often exhibit poor generalization. In this paper, it is shown that much better generalization can be obtained by using a variable interpolation kernel in combination with conjugate gradient optimization of the similarity metric and kernel size. The resulting method is called variable-kernel similarity metric (VSM) learning. It has been tested on several standard classification data sets, and on these problems it shows better generalization than back propagation and most other learning methods. An important advantage is that the system can operate as a black box in which no model minimization parameters need to be experimentally set by the user. The number of parameters that must be determined through optimization are orders of magnitude less than for back-propagation or RBF networks, which may indicate that the method better captures the essential degrees of variation in learning. Other features of VSM learning are discussed that make it relevant to models for biological learning in the brain.

UBC-TR-93-43. This paper will be published in *Neural Computation*.

1 Introduction

Classification methods based on nearest-neighbour interpolation have attracted growing interest in the neural network community. In part, this is because they support rapid incremental learning from new instances without degradation in performance on previous training data. Since the interpolation function is determined from a set of nearest neighbours at run time, it is easy to incrementally incorporate new training data and, if desired, to discount old data in a controlled manner (Atkeson, 1989; Omohundro, 1992). These capabilities are missing from the most popular neural network learning methods, yet they are necessary for models of biological learning and for on-line learning applications.

However, classical nearest-neighbour methods often exhibit poor generalization performance as compared to recent neural network learning methods. It has not been clearly recognized in the classical nearest-neighbours literature that the performance of these methods is highly dependent on the similarity (distance) metric that is used to select neighbours. In this paper, we combine a variable interpolation kernel with cross-validation optimization of the similarity metric and kernel size. The resulting system is called VSM (variable-kernel similarity metric) learning. It has much better generalization than the classical nearest-neighbour approach, and it performs as well or better than current neural network techniques on the comparison data sets to which it has been applied.

A particular advantage of this approach is that it solves for orders of magnitude fewer parameters than back propagation or radial basis function (RBF) methods, which greatly reduces the problem of overlearning. There is no need for the user to select model minimization parameters, such as the number of hidden units or basis function centers (in other learning methods these are usually determined by performing extensive cross-validation testing with each set of possible parameter values). VSM learning can be run as a black box without setting problem-specific parameters, which is a necessary requirement for biological models and for many on-line learning applications.

In addition to the problem of poor generalization, nearest-neighbour methods have been criticized for slow run-time performance and for increased memory requirements. The well-known k-d tree algorithm (Friedman, Bentley & Finkel, 1977; Sproull, 1991) can be used to identify the nearest neighbours, but its computation time is known to become large for random points in high dimensional spaces (in these cases, it must sometimes measure the distance to a large proportion of the inputs to find the single nearest neighbour). However, following similarity metric optimization, there is an effective dimensionality reduction as some dimensions are assigned higher weightings than others. This greatly speeds the k-d tree algorithm. In conjunction with the fact that VSM learning only looks at a small number (about 10) nearest neighbours, the run-time performance on many problems is actually better than competing methods such as back-propagation. If the k-d tree algorithm is still not efficient enough in certain cases, then an approximation to the nearest-neighbour can be used that looks at only a limited number of leaves of the k-d tree. The problem of increased memory requirements has been partly addressed by an editing procedure that removes unnecessary training data from regions where there is little uncertainty.

2 Previous research

Nearest neighbour classification techniques have been the topic of hundreds of papers over the past 40 years in the pattern recognition and statistical classification literature. An excellent survey of this area has recently been prepared by Dasarathy (1991). A surprising shortcoming of this extensive literature is that it gives little attention to the problem of selecting the optimal distance norm for determining nearest neighbours. In most papers, this issue is avoided by looking at the asymptotic performance as the number of training cases approaches infinity (in which case the metric is irrelevant). However, any reasonable learning method will converge to the Bayes optimal solution with infinite data, so it is the number of training cases required for a given level of performance that distinguishes learning methods.

The importance of an appropriate distance metric can be seen by the degradation in performance that often accompanies the addition of new input features. Each time an unimportant feature is added to the feature set and assigned a weight similar to an important feature, it increases the quantity of training data that is needed by a factor that allows for all combinations of the important and unimportant values. It is easy to create exponential increases in training data requirements by adding only a few poorly weighted features. This is why nearest-neighbours algorithms have sometimes shown excellent performance (when appropriate features and metrics have been used), but also often show poor performance in comparison with other learning methods (when poor metrics are chosen, usually on the basis of equal weighting for each feature).

One reason for the strong interest in neural network learning methods, such as back propagation, is that they are able to select useful input features from high-dimensional input vectors. Therefore, they do not suffer from the "curse of dimensionality" of classical nearest neighbours, in which higher dimensional inputs become less likely to provide accurate classifications with reasonable amounts of training data. This becomes even more important when it is necessary to take weighted combinations of noisy redundant inputs in order to produce the best classification. In these cases, it is even less likely that the initial assigned weights will be appropriate. In this paper, we use the same optimization techniques developed for other neural network methods, but apply them directly to determining relative feature weightings. The result is that equivalent or better generalization can be achieved while solving for far fewer parameters and gaining the other advantages of the nearest neighbour approach.

Research in the neural network field has recently been moving towards algorithms that interpolate between nearest neighbours. One of the most popular of these methods is radial basis function (RBF) networks (Broomhead & Lowe, 1988; Moody & Darken, 1989). This is quite similar to the classical Parzen window method of estimating probability density distributions (Duda & Hart, 1973), except that it uses somewhat fewer basis functions and adds a linear output layer of weights that are optimized during the learning process. However, neither the RBF nor Parzen window method provides any way to optimize the similarity metric. Therefore, they suffer from the same problem as standard nearest neighbours, in which performance will be good only when the appropriate feature weighting happens to be specified by the user. Poggio & Girosi (1989, 1990) have proposed extensions to the RBF method, which they call generalized RBFs and hyper basis functions, that optimize the centers of the basis functions and the global similarity metric. This provides a very flexible framework, but the large number of parameters (including those in the output layer) means that it is necessary to select some problem-specific subset of parameters to optimize and to determine some limited number of basis functions that is smaller than the number of training examples. In practice, this requires extensive cross-validation testing to determine the model size and the appropriate selection of free parameters, which is computationally very expensive and prevents the use of incremental learning as needed for biological models or on-line learning.

Some previous research on the problem of optimizing a similarity metric is the work of Atkeson (1991) on robot learning. He uses cross-validation to optimize not only a similarity metric but also other stabilization and cross-correlation terms. Similarly, the work of Wettschereck & Dietterich (1992) selects a similarity metric for the Wolpert approach to the NETtalk problem. Both of these methods use a distance weighted interpolation kernel that has the property of giving infinite weight to training data that exactly matches the current input. This is clearly undesirable for noisy inputs, as is the case with most real-world problems. This paper instead makes use of a variable kernel method that provides better interpolation and approximation in the presence of noise. VSM learning is aimed at classification problems with many input features, whereas the more extensive correlation matrix fitting of Atkeson may be more appropriate for continuous output problems based on low-dimensional inputs, as occurs in the problem of robot control.

Cleveland and Devlin (1988) describe the LOESS method for locally weighted regression, in which a local weighting kernel is used to smooth multivariate data. However, they use a similarity metric that is proportional to the variance of each input feature rather than being optimized according to its value in determining the output.

3 Choice of interpolation kernel

The choice of the interpolating kernel can have a substantial effect on the performance of a nearest-neighbours classifier. Cover & Hart (1967) showed that the single nearest-neighbour rule can have twice the error rate of a kernel that obtains an accurate measure of the local Bayes probability. A doubling of the error rate for a given set of training data would make even the best learning method appear to have poor performance relative to the alternatives.

One widely-used kernel is to place a fixed-width Gaussian at each neighbour, as in the Parzen window method. However, a fixed-width kernel will be too small to achieve averaging where data points are sparse and too large to achieve optimal locality where data points are dense. There is a trade-off between averaging points to achieve a better estimate of the local Bayes probability versus maintaining locality in order to capture changes in the output. As Duda & Hart (1973, p. 105) have shown, most of the benefits of local averaging are achieved from averaging small numbers of points. In fact, the k-nearest-neighbour method achieves a relatively good performance by maintaining a constant number of points within the kernel.

The benefits of the k-nearest-neighbour method can be combined with the smooth weighting fall-off of a Gaussian by using what is known as the variable kernel method (Silverman, 1986). In this method, the size of a Gaussian kernel centered at the input is set proportional to the distance of the k-th nearest neighbour. In this paper, we instead use the average distance of the first k neighbours, because this measure is more stable under a changing similarity metric. The constant relating neighbour distance to the Gaussian width is learned during the optimization process, which allows the method to find the optimal trade-off between localization and averaging for each particular data set.

In a classification problem, the objective is to compute a probability p_i for each possible output label i given any new input vector $\mathbf{x}$. In VSM learning, this is done by taking the weighted average of the known correct outputs of a number of nearest neighbours. Let n_j be the weight that is assigned to each of the J (e.g., $J = 10$) nearest neighbours, and s_{ij} be the known output probability (usually 0 or 1) for label i of each neighbour. Then,

$$p_i = \frac{\sum_{j=1}^{J} n_j s_{ij}}{\sum_{j=1}^{J} n_j}.$$

The weight n_j assigned to each neighbour is determined by a Gaussian kernel centered at $\mathbf{x}$, where d_j is the distance of the neighbour from $\mathbf{x}$:

$$n_j = \exp(-d_j^2/2\sigma^2).$$

The distance d_j depends on the similarity metric weights w_k that will be learned during the optimization process for each dimension k of the input vector. Let $\mathbf{c_j}$

be the input location of each neighbour. Then, the weighted Euclidean distance is

$$d_j^2 = \sum_k w_k^2 (x_k - c_{jk})^2.$$

The width of the Gaussian kernel is determined by σ, which is a multiple of the average distance to the M nearest neighbours. It is better if only some fraction (e.g., $M = J/2$) of the neighbours is used, so that the kernel becomes small even when only a few neighbours are close to the input. There is a multiplicative parameter r relating the average neighbour distance to σ which is learned as a part of the optimization process (a typical initial value is $r = 0.6$, which places the average neighbour near the steepest slope of the Gaussian):

$$\sigma = \frac{r}{M} \sum_{m=1}^{M} d_m.$$

If it is successful, the optimization will select a larger value for r for noisy but densely-sampled data, and a smaller value for data that is sparse relative to significant variations in output.

4 Optimization of the similarity metric

The similarity metric weights and the kernel width factor are optimized using the cross-validation procedure that has been widely adopted in neural network research. This minimizes the error resulting when the output of each exemplar in the training set is predicted on the basis of the remaining data without that exemplar. As Atkeson (1991) has discussed, this is simple to implement with the nearest-neighbour method because it is trivial to ignore one data item when applying the interpolation kernel. This avoids some of the problems of overtraining that are found in many other neural network learning methods that can not so easily remove a single exemplar to measure the cross validation error.

The particular optimization technique that has been used is conjugate gradient (with the Polak-Ribiere update), because it is efficient even with large numbers of parameters and converges rapidly without the need to set convergence parameters. One important technique in applying it to this problem is that the set of neighbours of each exemplar are stored before each line search, and the same neighbours are used throughout the line search. This avoids introducing discontinuities to the error measure due to changes in the set of neighbours with a changing similarity metric, which could lead in turn to inappropriate choices of step size in the line search. A nice side-effect is that this greatly speeds the line search, as repeatedly finding the nearest neighbours would otherwise be the dominant cost. For the problems we have studied, the conjugate gradient method converges to a minimum error in about 5 to 20 iterations.

In order to apply the conjugate gradient optimization in an efficient manner, it is necessary to compute the derivative of the cross validation error with respect to the parameters being optimized. The cross validation error E is defined as the sum over all training exemplars t and output labels i of the squared difference between the known correct output s_{ti} and the computed probability p_{ti} for that output label based on its nearest neighbours:

$$E = \sum_t \sum_i (s_{ti} - p_{ti})^2.$$

The derivative of this error can be computed with respect to each weight parameter w_k:

$$\frac{\partial E}{\partial w_k} = -2 \sum_t \sum_i (s_{ti} - p_{ti}) \frac{\partial p_{ti}}{\partial w_k}$$

where

$$\frac{\partial p_{ti}}{\partial w_k} = \frac{\sum_j (s_{ji} - p_{ti}) \partial n_j / \partial w_k}{\sum_j n_j}$$

and

$$\frac{\partial n_j}{\partial w_k} = \frac{-n_j w_k}{\sigma^2} \left((x_k - c_{jk})^2 - \frac{r d_j^2}{M\sigma} \sum_{m=1}^{M} \frac{(x_k - c_{mk})^2}{d_m} \right).$$

The sum in this last expression does not depend on the particular neighbour j and can therefore be precomputed for the set of neighbours.

In order to optimize the parameter r determining the width of the Gaussian kernel, we can substitute the derivative with respect to r for the last equation above:

$$\frac{\partial n_j}{\partial r} = \frac{n_j d_j^2}{r\sigma^2}.$$

As noted above, the error function has discontinuities whenever the set of nearest neighbours changes due to changing weights. This can lead to inappropriate selection of the conjugate gradient search direction, so the search direction should be restarted (i.e., switched to pure gradient descent for the current iteration) whenever the error or the gradient increases. In fact, simple gradient descent with line search seems to work well for this problem, with only a small increase in the number of iterations required as compared to conjugate gradient.

One final improvement to the optimization process is to add a stabilizing term to the error measure E that can be used to prevent large weight changes when there is only a small amount of training data. This is less important than for most other neural network methods because of the smaller number of parameters, but it can still be useful for preventing overfitting to small samples of noisy training data. The following stabilizing term S is added to the cross-validation error E:

$$S = \lambda^2 \sum_k \log^2 \left(\frac{w_k}{w_{k0}} \right)$$

which has a derivative of

$$\frac{\partial S}{\partial w_k} = \frac{2\lambda^2}{w_k} \log \left(\frac{w_k}{w_{k0}} \right).$$

This tends to keep the value of each weight w_k as close as possible to the initial weight value w_{k0} assigned by the user prior to optimization. We have used the stabilization constant $\lambda = 1$, which means that a one log-unit change in the weight carries a penalty equivalent to a complete misclassification of a single item of training data. This has virtually no effect when there is a large amount of training data—as in the NETtalk problem below—but will prevent large weight changes based on a statistically invalid sample for small data sets.

5 Minimizing memory requirements

One frequent criticism of nearest-neighbour methods is that they require much greater use of memory than neural network algorithms. However, this expectation of high memory requirements seems to be based on an invalid numerical comparison between the number of hidden units typically used in the back-propagation approach and the much larger number of exemplars in the training database. These are not directly comparable because each hidden unit normally maintains a weight for every possible discrete feature value or for each value range in a distributed representation, whereas each training exemplar requires only memory for a specific set of feature values. An example is provided by the NETtalk problem to be described below, in which an 80-hidden-unit back propagation network contains 18,629 weights, which actually requires more memory to store than the 1000 word training set. For other data sets with a less distributed representation, the nearest-neighbour approach will often require more memory, but the difference may not be as significant as is commonly implied.

On the other hand, it is clear that it is often unnecessary to retain all training data in regions of the input space that have unambiguous classifications. Nearest-neighbour learning algorithms can reduce their memory usage by only retaining the full density of training exemplars where they are needed near to classification boundaries and thinning them in other regions. There has been a considerable amount of research on this problem in the classical nearest-neighbours literature, as is summarized in the survey by Dasarathy (1991).

Most of this work is only relevant to a single-nearest-neighbour classifier, but the papers by Chang (1974) and Tomek (1976) give approaches that are relevant to a k-nearest-neighbour method.

For VSM learning, we have developed a simple editing procedure that removes data from regions that have unambiguous classifications. The method deletes an exemplar if its J nearest neighbours all agree on the same classification using the VSM classifier, and all of the neighbours assign this classification a probability above 0.6. For our experiments, we have used $J = 10$. There is no requirement that the removed exemplar have the same classification as its neighbours, so this can remove "noise" exemplars. The procedure is repeated until less than 5% of the remaining exemplars are deleted on any iteration. This is a very conservative method that deletes exemplars only within regions that have consistent classifications. The surprising result is that this actually improves generalization performance by a small but consistent amount in our experiments. This is clearly a topic on which further research would be useful.

6 Test results

The VSM learning method was first applied to synthetic data to test its ability to select features of interest and assign them appropriate relative weights. The task was to solve a noisy XOR problem, in which the first two real-valued inputs were randomly assigned values of 0 or 1 and the binary output class was determined by the exclusive-OR function of these inputs. Noise was added to these 2 inputs drawn from a normal distribution with a standard deviation of 0.3 (meaning that there was some overlap between the two classes). The next two inputs were assigned the same initial 0 or 1 values as the first two, but had noise with a standard deviation of 0.5. Finally another 4 inputs were added that had random zero-mean values with a standard deviation of 2.0. The presence of extra random inputs and varying noise levels results in poor performance of nearest-neighbours algorithms. Indeed, the performance of the basic nearest-neighbours algorithm on this data with 100 training and 100 test examples was only 54.3% correct (hardly better than the 50% achieved by random guessing). However, VSM learning achieved 94.6% correct, after 14 iterations of the conjugate gradient convergence, by assigning high weights to the first 2 input features, slightly smaller weights to the next 2, and much smaller weights to the random inputs. Note that because of the XOR determination of the output class, there would be no linear correlation between any individual input and the output, so any linear classifier or feature selection method would fail on this problem.

The next test was performed on the well-known NETtalk task (Sejnowski & Rosenberg, 1987) in which the input is a 7-letter window of an English word and the output is the pronunciation of the central letter. Recently, Wettschereck & Dietterich (1992) have tested many learning methods on this data, using a

Algorithm	*Letter*	*Phoneme*	*Stress*
Nearest neighbour	53.1	61.1	74.0
RBF	57.0	65.6	80.3
Back propagation	70.6	80.8	81.3
Wolpert	72.2	82.6	80.2
GRBF	73.8	84.1	82.4
VSM	73.4	83.7	81.2

Table 1: This table gives the percent correct generalization on the NETtalk task for different learning methods. All rows except the last are from (Wettschereck & Dietterich, 1992).

standardized test method that selects 1000 words of training data at random and a disjoint set of 1000 words of test data. Table 1 presents their results for many well-known learning algorithms, along with the results of running the VSM algorithm on the same data. As the table shows, VSM learning performs significantly better than back propagation or radial basis function (RBF) learning on this data. The one method that is slightly better is a generalized radial basis function (GRBF) method in which the center of each basis function is optimized. However, this required extensive cross-validation testing to select many parameters, such as number of centers and type of parameter adjustments, and the optimization failed to converge from some starting values. In contrast, VSM learning achieves almost the same generalization with only 10 minutes of training time on a SparcStation 2 and with no need for experimentation. The efficiency of the k-d tree access method is such that the distance to only 43 exemplars on average must be checked to determine the 10 nearest neighbours for classification. VSM learning optimizes only 8 parameters (the weights for the 7 inputs and the kernel size), whereas back propagation optimizes 18,629 parameters and GRBF optimizes 40,600 parameters for this problem.

The method by Wolpert listed in Table 1 is similar to VSM, in that it optimizes a distance metric for nearest neighbour interpolation. Wolpert (1990) originally selected the weights by hand, and he applied the method to an edited test set so that his comparison to previous NETtalk data was invalid. Wettschereck & Dietterich (1992) used a mutual information approach to compute the feature weights, and applied it to NETtalk using their standardized test procedure to get the results shown in Table 1. Wolpert's kernel is a distance-weighted kernel that gives infinite weight to an exemplar with zero distance, and these results show that the variable kernel method used in VSM learning has better generalization. Another approach to the NETtalk problem was taken by Stanfill & Waltz (1986), who computed a type of similarity metric from the nearest neighbours of each input. Although they do not perform systematic testing, they report that the results are at about the same level as back-propagation.

Algorithm	*Percent correct*
Back propagation	51
RBF	53
Gaussian node network	55
Nearest neighbour	56
VSM	61

Table 2: Percent correct generalization on vowel classification task for different learning methods. All rows except the last are from (Robinson, 1989).

VSM learning was also tested on Robinson's (1989) speaker-independent speech recognition data. Each item of training data corresponds to one of 11 vowel sounds, with the input features consisting of 10 real-valued numbers that were extracted from the speech signal using linear predictive analysis and other preprocessing. The training data is produced by 8 speakers saying each of the 11 vowels six times, while the test data is produced by 7 other speakers in the same format. In applying VSM learning, it is important that only neighbours produced by different speakers from the center input are accessed during training, as otherwise the weights will be optimized to recognize each vowel based on data from the same speaker. This is easy to accomplish by adding a field to each exemplar indicating the speaker. The result of VSM learning on this task was better than the other methods tested by Robinson, as shown in Table 2. In fact, the nearest neighbour algorithm performed very well for this task, and the reason for this is shown by the fact that the feature weights changed only a little from their initial value of 1.0 during VSM learning. Presumably, this is because of the careful preprocessing of the speech signal to extract a useful feature set. The further improvement of VSM learning over nearest neighbours is due to the use of the variable kernel.

Another data set on which VSM learning has been tested is Gorman & Sejnowski's (1988) sonar data set. Each exemplar in this data set consists of 60 real-valued inputs extracted from a sonar signal. The task is to classify the object from which the sonar is reflected as either a rock or a metal cylinder. Only their "aspect-angle dependent" test case was used, as the precise training and test data cannot be determined for their other series. In this case, VSM learning achieved 95.2% correct classification as compared to the best result of 90.4% obtained by Gorman and Sejnowski using back propagation. Given that only 104 training cases were available, the large number of input dimensions, and the inability to perform randomized trials, it is not clear whether this result is statistically significant.

7 Relevance to models of biological learning

A major long-term goal of learning research is to develop a model for the powerful learning mechanisms incorporated in the cerebral cortex of the brain. While many aspects of learning in the brain remain to be discovered, certain broad properties of its performance are well known. These include the capability of the brain to incrementally update its learned model with each new training stimulus and its ability to perform some tasks with as little as a single training exemplar (as when recognizing a new stimulus following a single exposure and then improving recognition performance with further exposures). Learning in one part of the input space does not produce any major degradation of performance in other parts. Of the currently proposed neural network learning methods, only those that perform some type of local interpolation seem capable of satisfying these constraints.

A biologically-plausible model of learning would need to develop a complete incremental learning method. It would be possible to start performing classification with very small numbers of inputs if the initial set of feature weights could be assigned by using weights from some similar previous task (for example, the recognition of a new person would initially be based on feature weights that have proved useful for recognizing other people). These weights would then need to be optimized incrementally rather than through a batch process such as conjugate gradient. One hypothetical model for implementing a variable kernel approach with neurons would be to initially assign a neuron to each new training experience. Each neuron would fire in proportion to its distance from a new input due to a Gaussian-shaped receptive field. The implementation of a variable kernel with a constant sum of neuron activations would require lateral inhibition between neurons at the same level, which is known to be a common aspect of cortical processing. To limit memory requirements, new inputs that are very similar to previous inputs would not be assigned to a new neuron, but instead would modify the output weights of the closest existing neurons to reflect the new output. This is similar to the role of the output layer of weights in RBF learning, so the learning would tend to switch from VSM to RBF approaches as the density of neighbours rose beyond what was needed to represent output variations. An open research problem is to derive a statistical test to determine when output variations are small enough to perform this combination of exemplars. One prediction that arises from VSM learning is that relative feature weights should be set on a more global basis than a single neuron (this differs from the separate feature weights of each unit in back-propagation). This could be accomplished in biological systems by determining feature weights from, for example, the activation level of a feature-encoding neuron rather than by changing individual synapse weights.

8 Conclusions and future directions

Nearest-neighbour methods have often shown poor generalization in comparison to other learning methods, and therefore have attracted little interest in the neural network community in spite of a number of attractive properties. This paper shows that with the choice of an appropriate kernel and optimization of the similarity metric, the generalization can be as good or better than the alternatives. In the data sets that have been tested, VSM learning achieves better generalization than the back-propagation algorithm and most forms of RBF networks. It also has a much reduced training time, and a large reduction in the number of parameters to be optimized. A particular advantage of the method is its ability to operate as a black box without the need for the user to assign critical parameter values.

One important area for further research is the ability to learn weights that vary between regions of the input space. Clearly, there are many problems for which the optimal feature weights vary for different regions of the input. On the other hand, there must be a fair quantity of training data to determine the feature weights with statistical reliability, so their optimization must also avoid being too local. One approach to this problem would be to partition the input space into regions using a data structure such as the k-d tree, and to perform the optimization separately in each region. The local parameters could be stabilized to also minimize their distance from the global values, which would reduce the problems of overlearning.

Another area of potential improvement would be to incorporate the learning of local linear models such as have been explored by Atkeson (1991) and Bottou & Vapnik (1992). These approaches fit a linear model to a set of neighbours around each input at classification time. At the cost of a large increase in run-time computation, the output can be based on a more accurate interpolation between inputs that accounts for their particular spatial distribution in the input space. This is likely to be particularly useful for continuous outputs.

9 References

Atkeson, C.G. 1989. Learning arm kinematics and dynamics. *Annual Review of Neuroscience* **12,** 157–83.

Atkeson, C.G. 1991. Using locally weighted regression for robot learning. *IEEE Conf. on Robotics and Automation,* Sacramento, CA, 958–963.

Bottou, L., and Vapnik, V. 1992. Local learning algorithms. *Neural Computation,* **4**, 888–900.

Broomhead, D.S., and Lowe, D. 1988. Multivariable functional interpolation and adaptive networks. *Complex Systems,* **2**, 321–355.

Chang, C.L. 1974. Finding prototypes for nearest neighbour classifiers. *IEEE Transactions on Computers,* **23,** 1179–84.

Cleveland, W.S., and Devlin, S.J. 1988. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association,* **83,** 596–610.

Cover, T.M., and Hart, P.E. 1967. Nearest neighbour pattern classification. *IEEE Transactions on Information Theory,* **IT-13,** 1, 21–27.

Dasarathy, B.V. 1991. NN concepts and techniques. *Nearest Neighbour (NN) Norms: NN Pattern Classification Techniques,* B.V. Dasarathy (Ed.), IEEE Computer Society Press, 1–30.

Duda, R.O. and Hart, P.E. 1973. *Pattern Classification and Scene Analysis.* New York: Wiley.

Friedman, J.H., Bentley, J.L., and Finkel, R.A. 1977. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software,* **3**, 209–226.

Gorman, R.P. and Sejnowski, T.J. 1988. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks,* **1**, 75–89.

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

Omohundro, S.M. 1992. Best-first model merging for dynamic learning and recognition, *Advances in Neural Information Processing Systems 4,* Morgan Kaufmann, Denver, 958–965.

Poggio, T., and Girosi, F. 1989. A theory of networks for approximation and learning. Report AI-1140, MIT Artificial Intelligence Laboratory, Cambridge, MA.

Poggio, T., and Girosi, F. 1990. Extensions of a theory of networks for approximation and learning: dimensionality reduction and clustering. Report AI-1167, MIT Artificial Intelligence Laboratory, Cambridge, MA.

Robinson, A.J. 1989. *Dynamic error propagation networks,* Ph.D. thesis, Cambridge University Engineering Department.

Sejnowski, T.J., and Rosenberg, C.R. 1987. Parallel networks that learn to pronounce English text. *Complex Systems,* **1,** 145–168.

Silverman, B.W. 1986. *Density Estimation for Statistics and Data Analysis,* Chapman and Hall, London.

Sproull, R.F. 1991. Refinements to nearest-neighbour searching in k-d trees. *Algorithmica,* **6**, 579–589.

Stanfill, C., and Waltz, D. 1986. Toward memory-based reasoning. *Communications of the ACM,* **29,** 1213–1228.

Tomek, I. 1976. An experiment with the edited nearest-neighbour rule. *IEEE Transactions on Systems, Man and Cybernetics,* **6,** 448–452.

Wettschereck, D., and Dietterich, T. 1992. Improving the performance of radial basis function networks by learning center locations, *Advances in Neural Information Processing Systems 4,* Morgan Kaufmann, Denver, 1133–40.

Wolpert, D.H. 1990. Constructing a generalizer superior to NETtalk via a mathematical theory of generalization. *Neural Networks,* **3,** 445–452.