Variational Learning for Multi-Layer Networks of Linear Threshold Units

Neil D. Lawrence

Computer Laboratory,
New Museums Site, Pembroke Street,
Cambridge, CB2 3QG, U.K.
neil.lawrence@cl.cam.ac.uk

Abstract

Linear threshold units were originally proposed as models of biological neurons. They were widely studied in the context of the perceptron (Rosenblatt, 1962). Due to the difficulties of finding a general algorithm for networks with hidden nodes, they never passed into general use. We derive an algorithm in the context of graphical models and show how it may be applied in multi-layer networks of linear threshold units. We demonstrate the performance of the algorithm on several real-world datasets.

1 Introduction

Linear threshold units were proposed by McCulloch and Pitts (1943) as a simple model of the biological neuron. The state of a node, s_i, depends on its parents' states and the parameters $\mathbf{W} = \{w_{ij}\}$ in the following way

$$s_i = \operatorname{sign}\left(\sum_j w_{ij} s_j + w_{i0}\right). \tag{1}$$

These types of node were implemented in by Rosenblatt in his perceptron. However, perceptrons were limited by the use of fixed basis functions which could not be adapted to the particular data set under consideration. The generalised delta rule popularised by Rumelhart *et al.* (1986) allowed networks to have adaptive basis functions through the use of multiple layers of weights. Unfortunately this rule may not be applied when the basis functions are linear threshold units. A common misconception is that this is due to the basis functions' discontinuity; a larger problem is that they have a gradient of zero

Submitted to Neural Networks *13 October 1999*

at all points except at the discontinuity. This means that any attempt to find the optimal parameters in such a network gains no useful information from the gradient: the gradient in the error surface as computed by the generalised delta rule is zero at almost all points.

One approach to this problem has been to use 'noisy weights' (Hinton and van Camp, 1993). This gives a probability of a threshold unit being active which is a smooth function of the weights. However, the approach leads to some restrictions on the structure of the network; namely only two layers of weights may be optimised. In this work we seek to derive a more general algorithm. Our inspiration is the variational method known as mean field theory, in particular the work of Saul *et al.* (1996). The approximations used in this approach become exact in the limits of interest, and as such present a tractable learning algorithm for networks of linear threshold units.

2 From Sigmoid Belief Networks ....

Sigmoid belief networks (Neal, 1992) are a form of probabilistic graphical model in which the nodes take states $\mathcal{S} = \{s_i\}$, where each s_i is taken to be bi-valued, we take $s_i \in \{-1, 1\}$, with a probability which is dependent on a non-linear function of a weighted sum of their parents. In the case of a sigmoid belief network the non-linear function is taken to be the logistic sigmoid⋆. The conditional probability of a node being on given the states of it's parents can therefore be seen to be

$$P(s_i = 1|\mathrm{pa}(s_i)) = \sigma\left(\sum_j \frac{w_{ij}}{T} s_j + \frac{w_{i0}}{T}\right) \tag{2}$$

where $\mathrm{pa}(s_i)$ is the set of nodes which are parents of node i, and the parameters $\mathbf{W}$ are a matrix of weights and biases. We absorb the bias parameters w_{i0} into the connectivity matrix, $\mathbf{W}$, by introducing an additional variable $s_0 = 1$. The 'temperature' parameter T in eqn. 2 is in principle redundant since it can be absorbed into the connectivity matrix, however it will prove convenient to separate it. We may write down the probability of the combined states of the model as:

$$P(\mathcal{S}) = \prod_i P(s_i|\mathrm{pa}(s_i)) \tag{3}$$

$$= \prod_i \frac{\exp\left[\left(\sum_j \frac{w_{ij}}{T} s_j\right) \frac{s_i+1}{2}\right]}{1 + \exp\left[\sum_j \frac{w_{ij}}{T} s_j\right]}. \tag{4}$$

⋆ The logistic function is given by $\sigma(x) = \frac{1}{1+\exp(-x)}$

The variables $\mathcal{S}$ maybe partitioned into a *visible* set $\mathcal{V} = \{v_i\}$ whose values are observed, and a *hidden* set $\mathcal{H} = \{h_i\}$ whose values are unobserved. The marginal probability of the observed states is obtained by summing over the hidden variables to give

$$P(\mathcal{V}) = \sum_{\mathcal{H}} P(\mathcal{H}, \mathcal{V}), \tag{5}$$

which can be viewed as a function of the parameters $\mathbf{W}$ in which case it represents a likelihood function. A training set then consists of a set of observations of the visible variables $\mathcal{V}_1, \ldots, \mathcal{V}_N$. The log likelihood of the data is a sum over patterns

$$\log P(\mathcal{V}) = \sum_{n=1}^{N} \log \left\{ \sum_{\mathcal{H}_n} P(\mathcal{H}_n, \mathcal{V}_n) \right\}. \tag{6}$$

Here we are implicitly assuming that it is the same set of variables which are observed in each pattern. From now on we suppress the summations over n to avoid cluttering the notation.

Maximum likelihood learning for sigmoid belief networks requires the evaluation of the likelihood function. Evaluation of $P(\mathcal{V})$ requires summing over exponentially many configurations of the hidden variables $\mathcal{H}$, and is computationally intractable for larger networks with dense connectivity.

2.1 Variational Inference

If we introduce a distribution $Q(\mathcal{H}|\mathcal{V})$, which we regard as an approximation to the true posterior distribution, then we may bound the likelihood below by

$$\log P(\mathcal{V}) \geq \sum_{\mathcal{H}} Q(\mathcal{H}|\mathcal{V}) \log \frac{P(\mathcal{V}, \mathcal{H})}{Q(\mathcal{H}|\mathcal{V})} \tag{7}$$

$$\equiv \mathcal{L}. \tag{8}$$

The difference between the true log likelihood and the bound 7 can be shown to be the Kullback-Leibler divergence between the true posterior distribution $P(\mathcal{H}|\mathcal{V})$ and the approximation $Q(\mathcal{H}|\mathcal{V})$.

$$\mathrm{KL}(Q||P) = \sum_{\mathcal{H}} Q(\mathcal{H}|\mathcal{V}) \log \frac{P(\mathcal{V}|\mathcal{H})}{Q(\mathcal{H}|\mathcal{V})} \tag{9}$$

Thus the correct log likelihood is reached when $Q(\mathcal{H}|\mathcal{V})$ exactly equals the true posterior. The aim of this approach is therefore to choose an approximating distribution which leads to computationally tractable algorithms and yet which is also flexible enough to permit a good representation of the true posterior. In this work we consider parametrised distributions, and we adapt the parameters to maximise the bound. This gives the best approximating distribution within the particular parametric family.

2.2 Mean Field Theory

Considerable simplification results if the model distribution is chosen to be factorial over the individual variables, so that $Q(\mathcal{H}|\mathcal{V}) = \prod_i Q(h_i)$, which gives *mean field theory.* Saul *et al.* (1996) have applied mean field theory to the problem of learning in sigmoid belief networks. Here we briefly review the framework of Saul *et al.* since this forms the basis of our algorithm. The mean field distribution is chosen to be a product of Bernoulli distributions of the form

$$Q(\mathcal{H}|\mathcal{V}) = \prod_i \left(\frac{\mu_i + 1}{2}\right)^{\frac{h_i+1}{2}} \left(\frac{\mu_i - 1}{2}\right)^{\frac{h_i-1}{2}} \tag{10}$$

in which we have introduced mean-field parameters μ_i. This leads to a lower bound on the log likelihood,

$$\mathcal{L} = \sum_{ij} \frac{w_{ij}}{T} \frac{\mu_i - 1}{2} \mu_j - \sum_i \left\langle \log \left[1 + \exp\left(\sum_j \frac{w_{ij}}{T} s_j \right) \right] \right\rangle_Q + \sum_i S(\mu_i) \tag{11}$$

where $S(x)$ is the binary entropy of the variable x and $\langle \cdot \rangle_Q$ denotes an expectation with respect to the distribution $Q(\mathcal{H}|\mathcal{V})$.

3 ... to Linear Threshold Units

The sigmoid belief network may be converted into a network of linear threshold units by taking the limit of eqn. 2 as the temperature goes to zero. The conditional probability now becomes discontinuous being equal to 1 for a subset the set of all possible values of $\mathcal{S}$, and 0 at all other times. This subset is the patterns which the network has stored. The output of each node is now deterministic as in eqn. 1. If we define the model structure, $\mathcal{M}$, to be one of R rows of nodes, each row being indexed by r and $\mathcal{S} \in \mathcal{V}$ iff $r = 1$ or $r = R$, our network becomes a multi-layer network of linear threshold units.

There are two benefits derived from exploring the bound in the zero temperature limit. First of all the mean field approximation becomes exact in this limit. Secondly we may easily handle the remaining expectation in eqn. 11

$$\lim_{T\to 0}\left\langle \log\left(1+\exp\left[\sum_j \frac{w_{ij}}{T}s_j\right]\right)\right\rangle_Q = \lim_{T\to 0}\frac{\lambda_i+1}{2}\sum_j \frac{w_{ij}}{T}\mu_j, \tag{12}$$

where $\lambda_i = \text{sign}\left(\sum_j w_{ij}\mu_j\right)$. We may now express our bound as

$$\mathcal{L} = \lim_{T\to 0}\sum_{ij}\frac{w_{ij}}{T}\mu_j\frac{(\mu_i-\lambda_i)}{2} + \sum_i S(\mu_i) \tag{13}$$

where $\mu_{i\in\mathcal{V}}$ is defined to be equal to the observed value.

4 Inference

It remains to determine the parameters of our approximating distribution. This may be achieved by setting the gradient $\frac{\partial\mathcal{L}}{\partial\mu_k}$ to zero revealing a fixed point equation for μ_k:

$$\mu_k = \text{sign}\left(\sum_i \frac{w_{ik}}{2}(\mu_i-\lambda_i) + \sum_j w_{kj}\mu_j\right). \tag{14}$$

Updates of each $\mu_{k\in\mathcal{H}}$ may then be undertaken for each pattern n in the training set. Upon presentation of each example, the equation may be applied to each node in random order until a stable solution is reached. These updates are taking place in a highly discontinuous space. It may be advantageous to reintroduce the temperature parameter and perform *deterministic annealing* during the updates of μ_k, gradually reducing the temperature to zero

$$\mu_k = \tanh\left(\sum_i \frac{w_{ik}}{2T}(\mu_i-\lambda_i) + \sum_j \frac{w_{kj}}{T}\mu_j\right). \tag{15}$$

See Waugh *et al.* (1990) for some theoretical justification of this approach. Initial values for $\mu_{k\in\mathcal{H}}$ may be chosen to be the values of $s_{k\in\mathcal{H}}$ obtained from a 'forward pass' through the network.

5 Learning

The values of $\mu_{k\in\mathcal{H}}$ obtained in the inference step can now be considered as hidden targets. The optimisation of the weight parameters eqn. 13 has become a linear programming problem in $\mathbf{W}$ with N constraints imposed for each hidden node. For values of $\mathbf{W}$ that satisfy the constraints the likelihood is one, for all other values it is zero.

5.1 The Perceptron Criteria

The linear programming problem provided at each node is identical to that of the perceptron, eqn. 13 provides the perceptron criteria for each node given the inferred values of μ_i. To form a historical connection, we first simply derive the perceptron learning rule for each node.

Gradient based optimisation of the weight parameters would normally be achieved through differentiation of eqn. 13 with respect to w_{kl}, however we would obtain a function which is inversely proportional to T and this would lead to gradients of infinity or zero. Instead, noting that the maxima of $\mathcal{L}$ are also the maxima of $T\mathcal{L}$ we choose instead to ascend the gradient of this product,

$$\frac{\partial T\mathcal{L}}{\partial w_{kl}} = \frac{\mu_k \mu_l - \lambda_k \mu_l}{2}. \tag{16}$$

The weights may now be adjusted according to a gradient descent algorithm.

The solution to this optimisation is not unique (figure 1) and may not exist (figure 2). If the solution does not exist then the perceptron learning rule will

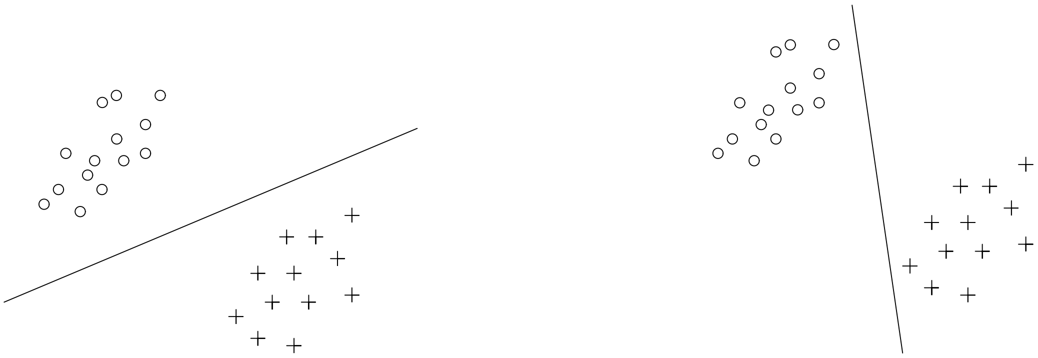

Fig. 1. These two examples have decision boundaries which would both represent solutions to the linear programming problem.

fail to converge. One heuristic which enables convergence is using the use of learning rate decay. However, we are still faces with the non-uniqueness of the solution. Ideally we seek the solution which we expect to generalise best on unseen data. We look to statistical learning theory to provide such a solution.

5.2 Maximum Margin Classifiers

The use of maximum margin in classification is a technique which arises from the field of *statistical learning theory*. In this work we only give a brief overview

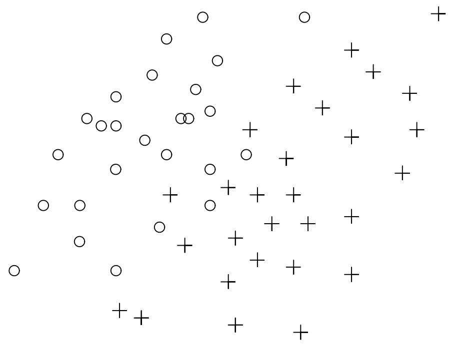

Fig. 2. No linear decision boundary can be found which solves the linear constraints imposed by this data.

of these classifiers (for a more detailed introduction see Burges (1998)). Statistical learning theory relies on notions of capacity which reflect the number of patterns that a classifier may store (Vapnik, 1982). It is possible to place bounds on the expected generalisation error which hold with a certain confidence, normally taken to be 95%. These bounds are functions of the model capacity which is often quantified in terms of the *VC dimension* (Vapnik and Chervonenkis, 1971)

$$R(\mathbf{W}) \leq R_{\text{emp}}(\mathbf{W}) + R_{\text{struct}}(\mathcal{M}) \tag{17}$$

where R is the expected generalisation error, known as the risk, R_{emp} is the error on the training set, known as the empirical risk, and R_{struct} is known as the structural risk and is increases with increasing model capacity.

The principal of structural risk minimisation is to minimise not only the empirical risk but also the structural risk, through capacity reduction, to obtain the best classifier. This is the underlying theory of maximum margin classification. The model capacity is known to be inversely related to margin size. Increasing the margin decreases the structural risk. We therefore seek the classifier with the maximum margin.

As shown in figure 3, the maximum margin solution is defined by data points which lie on that margin, known as support vectors, and by Lagrange multipliers associated with those points. It may be found using quadratic programming.

There remains the issue of separability. Fortunately the quadratic programming problem may be reformulated to allow errors through the introduction of slack variables and a parameter, C, known as the error penalty. A slack variable, ξ_n, represents the extent of the error on data point n. A penalty term of the form $C \sum_{n=1}^{N} \xi_n$ may then be introduced into the quadratic programming problem (figure 4). The solution found will naturally then be sensitive to the choice of C.

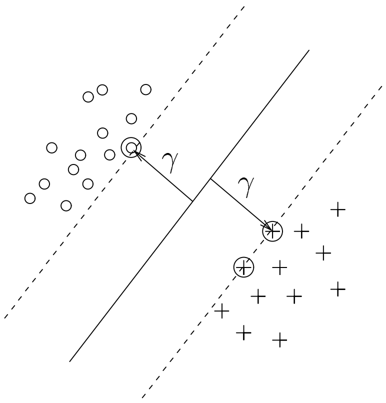

Fig. 3. In linearly separable data, maximum margin provides a unique solution. Circled data points indicate the support vectors and γ indicates the size of the margin.

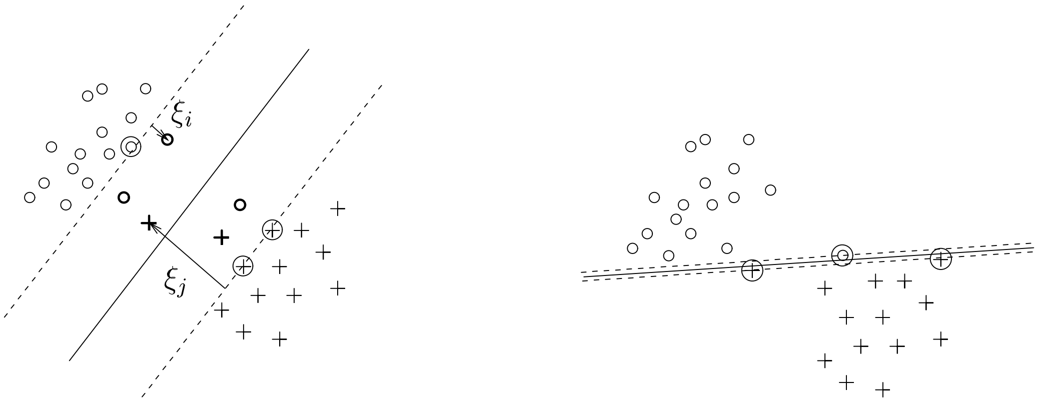

Fig. 4. Depending on the penalty assigned to errors, C, different solutions will be found. The left hand figure is a low error penalty, the right hand figure is a high error penalty.

5.2.1 Algorithm Overview

The algorithm has an interesting parallel with the expectation-maximisation (EM) algorithm (Dempster *et al.*, 1977), the 'expectation' step would be the determination of the parameters $\{\mu_k\}$ and the 'maximisation' step is of the negative error function in $\mathbf{W}$. It is incorrect, however, to call the fixed point equation update an expectation step because the function $Q(\mathcal{H}|\mathcal{V})$ is no longer probabilistic. This is analogous to the relationship between K-means clustering and EM optimisation of Gaussian mixture models (Bishop, 1995). K-means clustering may be seen as the limit as the variance goes to zero of EM update of a Gaussian mixture model. This method of learning in networks of linear threshold units is merely the limit as the temperature goes to zero of mean field theory for sigmoid belief networks.

One problem with the algorithm is its behaviour when the true posterior $P(\mathcal{H}|\mathcal{V})$ is zero for all possible $\mathcal{H}$. As a result there may be no valid solution for the update eqn. 14 and no guarantee of convergence**. Convergence could be guaranteed by implementing deterministic annealing to unit rather than zero temperature. The resulting value for each μ_k could then be thresholded.

We have considered only linear threshold units in this section. However it is possible to start with probabilistic graphs containing *soft-max* nodes and *linear-Gaussian* nodes and to arrive at networks containing *winner-takes-all* and *linear* activation functions. For the case of linear activation functions in the input rows and winner-takes-all activation functions in the output layer the derived equations are identical to the above. For linear inputs the value of μ_i for $r = 1$ becomes continuous. For the case of a network with linear inputs and only one layer of weights the product $T\mathcal{L}$ is identical to the perceptron criteria (Rosenblatt, 1962) and the weight update is the associated learning algorithm.

6 Results

6.1 Perceptron Criteria with Handwritten Digit Data

Evaluation of the algorithm was carried out using a dataset of 8×8 binary digits***. Single hidden layer networks containing $30 \rightarrow 270$ hidden nodes at 20 node intervals were studied. The weight matrix was structured with full connectivity available between adjacent rows of nodes and no connections between rows of nodes which were not adjacent. The first layer of weights was initialised with samples from a normal distribution which had a mean of 0 and a variance of $1/65$. The second layer of weights was initialised with samples which had a variance of $1/(H + 1)$ where H is the number of hidden nodes. The ten output nodes were winner-takes-all nodes.

The networks were each trained by both allowing the input to hidden layer weights to adapt and by fixing them at their initial values. Fixing these parameters is equivalent to perceptron learning as described by Rosenblatt; the hidden nodes are simply non-adaptive basis functions. On-line gradient descent was used with a learning rate initialised at $1/H$. The learning rate was then

**Although convergence is not guaranteed, it may still occur. A simple demonstration of this would be a network with only one hidden layer containing one node. Convergence will always occur in this case.

***The digit dataset was made available by the U.S. Postal Service Office of Advance Technology but was pre-processed by Hinton *et al.* (1995). It consists of 700 training examples of each digit and 400 test examples. Previous results on this dataset are reported by Saul *et al.* (1996) and Hinton *et al.*.

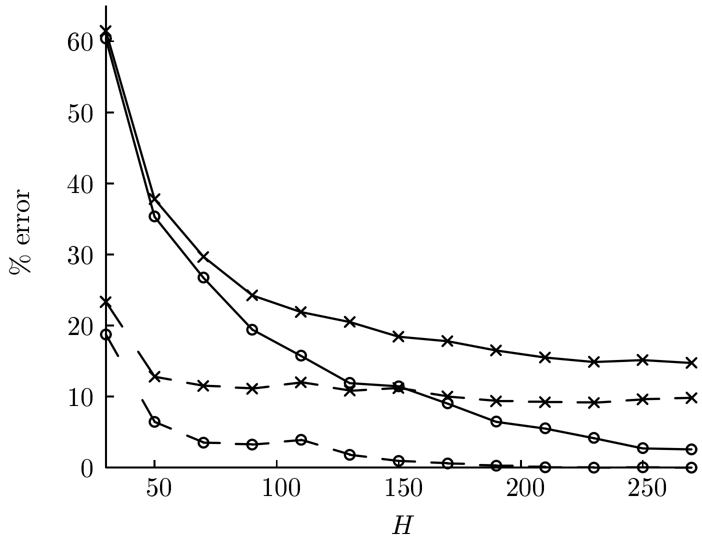

Fig. 5. The dependence of the error on the number of hidden nodes for the digit classification problem. The solid line is for networks with non-adaptive input to hidden weights. The dashed line is for networks with adaptive input to hidden weights. The error on the training data labelled by o, the error on the test data is labelled by $\times$.

algorithm	classification error
nearest neighbour	6.7 %
back-propagation	5.6 %
wake-sleep	4.8 %
mean field theory	4.6 %
Adaptive linear threshold units	9.1 %
Non-adaptive linear threshold units	14.8%

Table 1
Classification rates for different algorithms, the first three were reported by Hinton *et al.* (1995) and the fourth by Saul *et al.* (1996).

halved every ten iterations until one hundred iterations were complete to ensure convergence. Iterations of gradient descent were alternated with updates of the hidden targets. Figure (5) shows the performance of the two algorithms for the different networks and table (1) shows the performance of other algorithms and the minimum test error achieved by adaptive and non-adaptive linear threshold unit networks. The networks of adaptive Linear Threshold Units were able to completely separate the training data. As opposed to those with fixed basis functions. We have therefore overcome the fundamental limitations of the fixed basis functions with this method. The performance of the trained networks on the test set was poor though. We contend that this is due to a lack of any regularisation framework for the algorithm. Unfortunately this data set is too large to practically apply maximum margin techniques for every node.

6.2 *Maximum Margin*

In section 5.2 we proposed the use of maximum margin for training. In this section we apply the algorithm using maximum margin to three data sets. The first data set is a well known benchmarking data set. The second data set is from a medical imaging problem involving the analysis of fluorescent in-situ hybridization (FISH) images. For these experiments, updates were undertaken in the form of a full inference step, followed by learning of the parameters $\mathbf{W}$ to convergence using the sequential minimal optimisation algorithm (Platt, 1998). The process of inference and learning was repeated seven times for both data sets.

6.2.1 *The Pima Indians Diabetes Data*

This well known benchmark data set is demonstrated here with the partitions used by Ripley (1996). The objective is to create a classifier which predicts whether the subjects are diabetic. The data consists of seven inputs and a set of class labels. Ripley split the data into a training set of 200 points and a test set of 331 points. We normalised the data to lie between -1 and 1 and trained networks with differing error penalties, $C = 10^k$ for $k = 0, 1$ and 2, and different numbers of hidden units, $H = 10, 20$ and 30. Ten different initialisations were used for each parameterisation. The other presented results are taken from Ripley (1996) and Barber and Williams (1997). Results are summarised in figure 6 and table 2.

For model selection we implemented a bound on the risk due to Mason *et al.* (1999). Unfortunately there is a constant κ in the bound which is not explicitly defined. We therefore display results for several different values.

The results on this data appear encouraging. Note though the variability in performance with different initialisations shown in figure 6. However the average performance was still in line with that of the neural network results. Some caution has to be exercised with interpretation of these results. The best linear threshold unit classifier had a better performance on the test set than on the training set. This gives rise to some suspicion that these results may be somewhat partition sensitive due to the low number of samples. There appears to be some merit in the bound used for model selection. The bound has managed to extract some of the better performing networks.

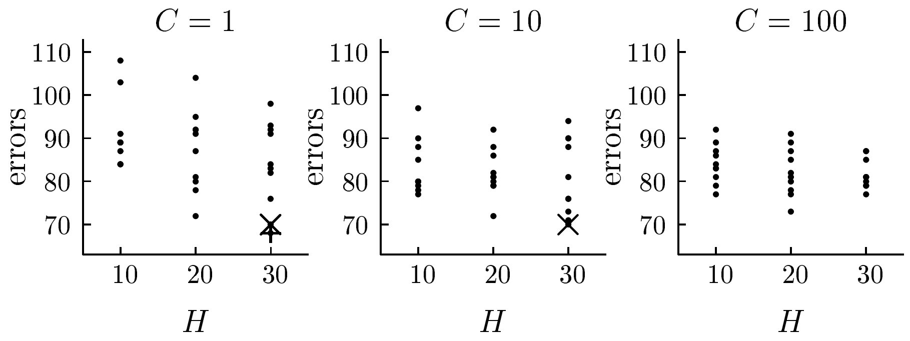

Fig. 6. Results for the networks on the Pima Indians datasets for $C = 1$, $C = 10$ and $C = 100$. Dots represent each of the ten different networks initialisations. Networks which achieved the overall minimum error are marked with +. Those which were selected via examination of bounds on the risk are marked with $\times$.

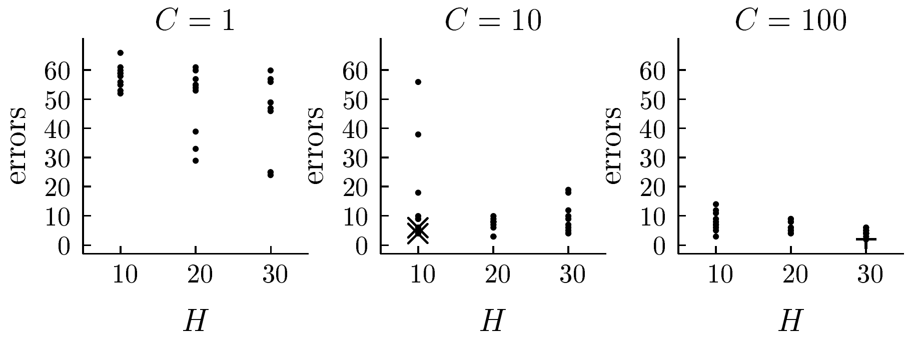

Fig. 7. Results for the networks on the Leptrograpsus Crabs datasets for $C = 1$, $C = 10$ and $C = 100$. Dots represent each of the ten different networks initialisations. Networks which achieved the overall minimum error are marked with +. Those which were selected via examination of bounds on the risk are marked with $\times$.

6.2.2 The Leptrograpsus Crabs Data

This is another benchmark data set. The partitions implemented here are from Ripley (1994). Five input features are used to describe crabs of two different species. The objective is to predict the sex of the crab given these features The data consists of 80 training points and 120 test points. The data was tested in the same manner as the Pima Indians data set. The results are depicted in figure 7, and the error rates are shown in table 2.

Similar conclusions may be drawn from the results on the crabs data as were drawn for those on the Pima data but with the same reservations also. For a better evaluation of the algorithm we turned to a larger data set from a medical application.

model	Pima	Crabs
Neural Network	75+	3
Linear Discriminant	67	8
Logistic Regression	66	4
MARS (degree = 1)	75	4
PP (4 ridge functions)	75	6
2 Gaussian Mixture	64	-
Gaussian Process Classifier	68	3
Best Linear Threshold Unit	68	2
Linear Threshold Unit $\kappa = .01$	70	6
Linear Threshold Unit $\kappa = .1$	70	6
Linear Threshold Unit $\kappa = 1$	70	6
Linear Threshold Unit $\kappa = 10$	70	4
Linear Threshold Unit $\kappa = 100$	70	4

Table 2
Classification rates for Pima Indians dataset.

6.2.3 FISH data

The FISH data set was acquired from 400 images of cell nuclei. The objective is to classify signals produced by fluorescent in-situ hybridization as either real signals, or artifacts of the slide preparation. The data consists of twelve features of the images which are based on shape and colour. The data were hand labelled by a trained cytologist. The problem is more fully described in Lerner and Lawrence (1999). We followed Lerner and Lawrence in partitioning the data into 2200 training points and 944 test points. For training the linear threshold units we made use of a validation set by further sub-dividing the training data. The validation set contained 314 points. Cross validation was not a viable option for these networks as once again the results were very initialisation dependent. The inputs were normalised to have zero mean and unit standard deviation for the experiments.

The results for the linear threshold units are bettered by all but one of the other classifiers. However the best performing linear threshold unit network gives results in line with that of the neural network. The linear threshold units are placed at a disadvantage with respect to the other techniques which were trained on all the available training data. This may go some way to explaining the discrepancy between its performance and that of the neural network. The selection on the basis of the risk bound was not as effective on this data set as it was for the smaller data sets. Whilst the networks selected for low κ appear

model	Classification Rate
Naive Bayesian Network	17.0 %
Neural Network	13.6 %
Bayesian Neural Network	11.8 %
Support Vector Machine	12.8 %
Best Linear Threshold Unit	14.3 %
Linear Threshold Unit (Validation set)	14.6 %
Linear Threshold Unit $\kappa = .01$	15.5 %
Linear Threshold Unit $\kappa = .1$	15.3 %
Linear Threshold Unit $\kappa = 1$	19.6 %
Linear Threshold Unit $\kappa = 10$	20.7 %
Linear Threshold Unit $\kappa = 100$	20.7 %

Table 3
Classification rates for the FISH data.

promising, those selected with higher κ are amongst the worst performing networks.

7 Discussion

We have presented an algorithm for learning in networks of linear threshold units. We demonstrated the algorithm's implementation with a variety of datasets. We note first of all that the algorithm was able to separate completely a digits data set and achieved zero training error. Poor generalisation results for the perceptron criteria were overcome through the use of maximum margin. The maximum margin results appeared to be very initialisation sensitive. We suggest this is due to a random elements in the inference step for which convergence is not guaranteed. We believe that the generalisation performance of the linear threshold unit networks would be much improved if the dependence on initialisation could be overcome.

References

Anderson, J. A. and E. Rosenfeld (Eds.) (1988). *Neurocomputing: Foundations of Research*, Cambridge, MA. MIT Press.

Barber, D. and C. K. I. Williams (1997). Gaussian processes for Bayesian classification via hybrid Monte Carlo. In M. C. Mozer, M. I. Jordan, and

T. Petsche (Eds.), *Advances in Neural Information Processing Systems*, Volume 9. Cambridge, MA: MIT Press.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* **2** (2), 121–167.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B* **39** (1), 1–38.

Hinton, G. E., P. Dayan, B. J. Frey, and R. M. Neal (1995). The wake-sleep algorithm for unsupervised neural networks. *Science* **268**, 1158–1161.

Hinton, G. E. and D. van Camp (1993). Keeping neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Anuual Conference on Computational Learning Theory*, pp. 5–13.

Lerner, B. and N. D. Lawrence (1999). A comparison of state-of-the-art classification techniques with application to cytogenetics. Technical report, Cambridge University. Submitted to *Neural Computing and Applications.*

Mason, L., P. Bartlett, and M. Golea (1999). Generalization error of combined classifiers. *Journal of Computer and System Sciences*. To appear.

McCulloch, W. S. and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* **5**, 115–133. Reprinted in Anderson and Rosenfeld (1988).

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial Intelligence* **56**, 71–113.

Platt, J. C. (1998). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola (Eds.), *Advances in Kernel Methods: Support Vector Learning*, pp. 185–208. Cambridge, MA: MIT Press.

Ripley, B. D. (1994). Flexible non-linear approaches to classification. In V. Cherkassky, J. H. Friedman, and H. Wechsler (Eds.), *From Statistics to Neural Networks. Theory and Pattern Recognition Applications.*

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks.* Cambridge, U.K.: Cambridge University Press.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning internal representations by error propagation. In *Parallel Distributed Programming: Explorations in the Microstructure of Cognition*, Volume 1: Foundations, pp. 318–362. Cambridge, MA: MIT Press. Reprinted in Anderson and Rosenfeld (1988).

Saul, L. K., T. S. Jaakkola, and M. I. Jordan (1996). Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research* **4**, 61–76.

Vapnik, V. N. (1982). *Estimation of Dependences Based on Empirical Data.*

New York: Springer-Verlag.

Vapnik, V. N. and A. Y. Chervonenkis (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications* **16** (2), 264–280.

Waugh, F. R., C. M. Marcus, and R. M. Westervelt (1990). Fixed-point attractors in analog neural computation. *Physical Review Letters* **64** (16), 1986–1989.