# Variational Learning for Multi-Layer Networks of Linear Threshold Units

Neil D. Lawrence,
Microsoft Research,
St George House,
1 Guildhall Street,
Cambridge, CB2 3NH, U.K.
neil@microsoft.com

February 5, 2001

**Abstract**

Linear threshold units (LTUs) 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. In this work we derive an algorithm in the context of a probabilistic models and show how it may be applied in multi-layer networks of linear threshold units. We demonstrate the performance of the algorithm on three data-sets.

## 1 Linear Threshold Unit Networks

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

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

When nodes of this type are to be implemented in pattern recognition, the model is often chosen to be one with a layered structure such as in Figure 1. In this example there is a layer of input nodes, which represent input variables, that can be real valued as well as binary. This layer is then connected to a layer of hidden nodes, whose activation function takes the form of Eqn (1), which are in turn connected to a layer of output nodes with the same activation function. The structure can be more general, containing more layers of hidden nodes and across layer connections (for example between input and output

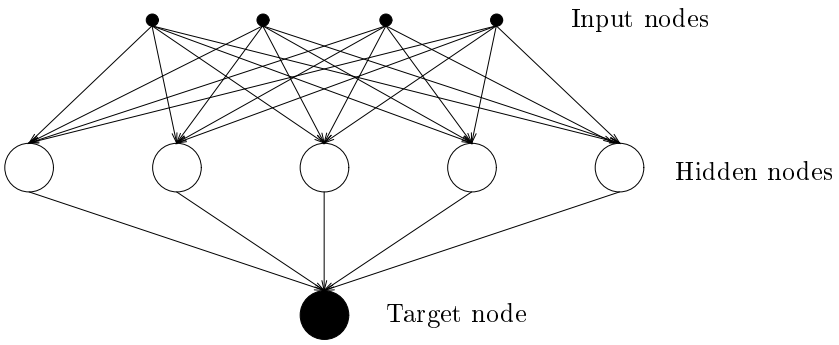


Figure 1: A linear threshold unit network with one layer of hidden units.

nodes), and our theoretical results apply to all models of this type. For simplicity though, we constrain our practical investigations to models of the type depicted in Figure 1. It is also possible to envisage cases where some or all of the input variables are unobserved. We only investigate the case where they are all observed, which is sometimes known as supervised learning. Models of this type were implemented by Rosenblatt (Rosenblatt, 1962) in his perceptron. Perceptrons, however, were limited by the use of fixed basis functions which could not be adapted to the particular data-set under consideration. In other words the parameters determining the output of the hidden nodes where taken to be fixed. The generalised delta rule popularised by Rumelhart *et al.* (1986) allowed networks to have adaptive basis functions through the use of multiple layers of adaptable 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 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 of the error surface as computed by the generalised delta rule is zero at almost all points.

One approach to learning in these networks 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, in particular the approach, without further approximation, only applies to regression networks containing only one hidden layer. In this work the algorithm we derive is inspired by the variational approach of mean field theory, in particular the work of Saul *et al.* (1996). Bounds used in our mean field approach can become exact in the limits of interest and present a tractable learning algorithm for networks of linear threshold units.

## 2 From Sigmoid Belief Networks ....

Consider a sigmoid belief network (Neal, 1992) with bi-valued nodes, $s_i \in \{-1, 1\}$. The probability of a node being equal to one given the states of its parents can be seen to be

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

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. 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. Consider a potential function given by

$$E_i(s_i, \mathbf{w}_i, \mathrm{pa}(s_i), T) = \left(\sum_j \frac{w_{ij}}{T} s_j\right) \frac{(s_i + 1)}{2}, \tag{3}$$

where we have absorbed the biases into the weight matrix $\mathbf{W}$ in the usual manner by introducing a dummy variable $s_0 = 1$. The probability of the combined states of the model, $\mathcal{S}$, is

$$P(\mathcal{S}) \quad = \quad \prod_i P(s_i|\mathrm{pa}(s_i)), \tag{4}$$

where we may write the conditional probability as

$$P(s_i|\mathrm{pa}(s_i)) \quad = \quad \frac{\exp\left[E_i(s_i, \mathbf{w}_i, \mathrm{pa}(s_i), T)\right]}{1 + \exp\left[E_i(1, \mathbf{w}_i, \mathrm{pa}(s_i), T)\right]} \tag{5}$$

$$\equiv \quad \frac{\exp\left[E_i(s_i, \mathbf{w}_i, \mathrm{pa}(s_i), T)\right]}{\sum_{s_i} \exp\left[E_i(s_i, \mathbf{w}_i, \mathrm{pa}(s_i), T)\right]}. \tag{6}$$

The variables may be partitioned into a visible set $\mathcal{V} = \{v_i\}$, and a hidden set $\mathcal{H} = \{h_i\}$. In the context of supervised learning, our observed data, $\mathcal{V}$, may be further split into two subsets: input data, $\mathcal{I}$, and data labels or output data, $\mathcal{O}$. When the input data has been fully observed it is unnecessary to infer its distribution, we therefore need only consider the distribution $P(\mathcal{H}, \mathcal{O}|\mathcal{I})$. Marginalising over the latent variables of this distribution gives $P(\mathcal{O}|\mathcal{I})$ which in supervised learning is the distribution of interest.

$$P(\mathcal{O}|\mathcal{I}) = \sum_{\mathcal{H}} P(\mathcal{H}, \mathcal{O}|\mathcal{I}). \tag{7}$$

A training set consists of a set of observations of the input variables, $\mathcal{I}_1, \ldots, \mathcal{I}_N$, and the output variables, $\mathcal{O}_1, \ldots, \mathcal{O}_N$. The log likelihood of the data is a sum over patterns

$$\ln P(\mathcal{O}|\mathcal{I}) = \sum_{n=1}^{N} \ln\left\{\sum_{\mathcal{H}_n} P(\mathcal{H}_n, \mathcal{O}_n|\mathcal{I}_n)\right\}. \tag{8}$$

Henceforth we suppress the summations over $n$ to avoid cluttering the notation. Additionally to maintain simple notation we will denote the observed values of the individual variables ($s_i$ for $i \in \mathcal{I}$ or $i \in \mathcal{O}$) as an expectation of that variable under the $Q(\mathcal{H})$, $\langle s_i \rangle_{Q(\mathcal{H})}$ or more concisely $\langle s_i \rangle$.

The size of the sum in Eqn (7) becomes impractical for large networks. We must therefore seek an alternative approach to learning without performing the sum explicitly.

## 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

$$\ln P(\mathcal{O}|\mathcal{I}) \quad \geq \quad \sum_{\mathcal{H}} Q(\mathcal{H}|\mathcal{O},\mathcal{I}) \ln \frac{P(\mathcal{H},\mathcal{O}|\mathcal{I})}{Q(\mathcal{H}|\mathcal{O},\mathcal{I})}. \tag{9}$$

The aim of this approach being 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. Variational approaches in sigmoid belief networks are already well known (Saul *et al.*, 1996). The derivation we give here however is novel. We re-derive the variational approach known as mean field theory for networks containing bi-polar valued nodes. We do so from the perspective of free-form variational optimisations rather than explicitly implementing a functional form for our approximation (see also (Haft *et al.*, 1999)). We also handle the additional intractability arising in the expectation of the normalising constant (the denominator in Eqn (6)) in different manner to previous works.

To implement mean field theory, we first assume the posterior approximation factorises across sub-sets of the full variable set. If we assume that the posterior factorises across each variable member, $s_j$, we obtain

$$Q(\mathcal{H}|\mathcal{O},\mathcal{I}) = \prod_{i \in \mathcal{H}} Q(s_i), \tag{10}$$

we may now perform a free-form optimisation (MacKay, 1995) over the functional form of each factor $Q(s_i)$ leading to

$$Q(s_j) \propto \exp\left( \langle \ln P(s_j|\mathrm{pa}(s_j)) \rangle_{\prod_{k \neq l} Q(s_k)} + \sum_{i \in \mathrm{ch}(s_j)} \langle \ln P(s_i|\mathrm{pa}(s_i)) \rangle_{\prod_{k \neq j} Q(s_k)} \right). \tag{11}$$

where $\mathrm{ch}(s_j)$ represents those nodes that are children of the node $s_j$. Substitut-

ing in the form of our conditional distributions from Eqn (6) we obtain

$$Q(s_j) \propto \exp\left( \langle E_j(s_j, \mathbf{w}_j, \mathrm{pa}(s_j)) \rangle_{\prod_{k\in \mathrm{pa}(s_j)} Q_k} - \sum_{i\in \mathrm{ch}(s_j)} \left\langle \ln \sum_{s_i} \exp\left[E_j(s_i, \mathbf{w}_i, \mathrm{pa}(s_i))\right] \right\rangle_{\prod_{k\in \mathrm{pa}(i), k\neq j} Q_k} \right). \quad (12)$$

Here we have used $Q_k$ as shorthand for $Q(s_k)$. The first term in the exponent is straightforward to evaluate, the second term, however, proves more difficult. The cause is the log of the sum. We choose to again apply a variational lower bound (9) to this term.

$$\ln \sum_{s_i} \exp\left[E_i(s_i, \mathbf{w}_i, \mathrm{pa}(s_i), T)\right] \geq \sum_{s_i} Q^-(s_i|\mathrm{pa}(s_i)) E_i(s_i, \mathbf{w}_i, \mathrm{pa}(s_i), T) - \sum_{s_i} Q^-(s_i|\mathrm{pa}(s_i)) \ln Q^-(s_i|\mathrm{pa}(s_i)) \quad (13)$$

It is straightforward to show, through free form optimisation, that this bound is maximised by setting

$$Q^-(s_i|\mathrm{pa}(s_i)) = \frac{\exp(E_i)}{\sum_{s_i} \exp(E_i)}, \quad (14)$$

where we have used the shorthand $E_i$ to represent $E_i(s_i, \mathbf{w}_i, \mathrm{pa}(s_i), T)$[1].

### 2.1.1 Expectations under $Q(\mathcal{H})$

The expectation of a variable under the distribution $Q^-(s_i|\mathrm{pa}(s_i))$ can easily be seen to be

$$\langle s_i \rangle_{Q_i^-} = \tanh\left(E_i(1, \mathbf{w}_i, \mathrm{pa}(s_i), T)\right). \quad (15)$$

we may now rewrite Eqn (12) as

$$Q(s_j) \propto \exp\left( \langle E_j(s_j, \mathbf{w}_j, \mathrm{pa}(s_j)) \rangle_{\prod_{k\in \mathrm{pa}(s_j)} Q_k} - \sum_{i\in \mathrm{ch}(s_j)} \left\langle \left(\frac{w_{ji}}{T} s_j\right) \frac{(\langle s_i \rangle_{Q_i^-} + 1)}{2} \right\rangle_{\prod_{k\in \mathrm{pa}(i), k\neq j} Q_k} + \sum_{i\in \mathrm{ch}(s_j)} \mathcal{H}\left(Q^-(s_i)\right) \right), \quad (16)$$

[1] In fact $Q^-(s_i|\mathrm{pa}(s_i))$ in Eqn (14) is recognised as the true posterior distribution of $s_i$ given its parents (Eqn (6)). Thus when $Q^-$ is set as shown in Eqn (14), the bound (13) becomes an equality.

where $\mathcal{H}(Q^-(s_i))$ is the entropy of the distribution $Q^-(s_i)$. The argument of the exponential in Eqn (16) can be written in terms of a function of $s_j$, which we denote $E'_j(s_j)$, a constant and the entropy term. Substituting Eqn (3) into Eqn (16) and collecting terms in $s_j$ we write

$$\log Q(s_j) = E'_j(s_j) + \sum_{i \in \mathrm{ch}(s_j)} \mathcal{H}\left(Q^-(s_i)\right) + \text{const.} \tag{17}$$

where the function $E'_j(s_j)$ can be seen to be

$$E'_j(s_j) \quad = \quad s_j \left( \sum_i \frac{w_{ij}}{2T} \langle s_i \rangle + \sum_{i \in \mathrm{ch}(s_j)} \frac{w_{ji}}{2T} \left( \langle s_i \rangle - \left\langle \langle s_i \rangle_{Q^-} \right\rangle \right) \right). \tag{18}$$

Now note that the expectation of $s_i$ under $Q^-$ is also dependent on $s_j$ (Eqn (15)). We therefore rewrite this expectation to bring out the dependence on $s_j$,

$$\begin{aligned}
\langle s_j \rangle_{Q^-} &= \tanh\left( \sum_k \frac{w_{ik}}{T} s_k \right) \\
&= \frac{1+s_j}{2} \tanh\left( \sum_{k \neq j} \frac{w_{ik}}{T} s_k + \frac{w_{ij}}{T} \right) + \frac{1-s_j}{2} \tanh\left( \sum_{k \neq j} \frac{w_{ik}}{T} s_k - \frac{w_{ij}}{T} \right) \\
&= \frac{s_j}{2} \left( \tanh\left( \sum_{k \neq j} \frac{w_{ik}}{T} s_k + \frac{w_{ij}}{T} \right) - \tanh\left( \sum_{k \neq j} \frac{w_{ik}}{T} s_k - \frac{w_{ij}}{T} \right) \right) + \text{const.} \\
&\stackrel{\text{def}}{=} s_j \phi_{ij} + \text{const.},
\end{aligned} \tag{19}$$

where const. is a term constant in $s_j$. Substituting this representation back into Eqn (18) we obtain,

$$E'_j(s_j) \quad = \quad s_j \left( \sum_i \frac{w_{ij}}{2T} \langle s_i \rangle + \sum_j \frac{w_{ji}}{2T} \left( \langle s_i \rangle - \langle \phi_{ij} \rangle \right) \right), \tag{20}$$

leading to

$$Q(s_j) = \frac{\exp\left(E'(s_j)\right)}{\exp\left(E'(1)\right) + \exp\left(E'(-1)\right)}, \tag{21}$$

This can be substituted into bound (9) along with Eqn (4) to determine the lower bound on the log-likelihood, as we shall see in Section 4.

# 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

marginal likelihood will then become discontinuous being equal to 1 for a subset of 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 then deterministic as in Eqn (1) and our network is a multi-layer network of linear threshold units.

We can now consider the behaviour of the distribution in Eqn (21) in the zero temperature limit, or more specifically we discuss the behaviour of the expectation of $s_i$ under this distribution in that limit. Using the distribution in Eqn (21) the expectation of any node can be seen to be

$$\langle s_k \rangle = \lim_{T \to 0} \tanh \left( E'(s_j) + \sum_{i \in \mathrm{ch}(s_j)} \mathcal{H}(Q^-(s_i)) \right), \tag{22}$$

When we take the limit the entropy term goes to zero and, through substitution of Eqn (18) for $E'(s_j)$, we find

$$\langle s_k \rangle = \mathrm{sign} \left( \sum_i w_{ij} \langle s_i \rangle + \sum_i \frac{w_{ji}}{2} \left( \langle s_i \rangle - \langle \phi_{ij} \rangle \right) \right). \tag{23}$$

Additionally the expectation $\langle \phi_{ij} \rangle$ also becomes tractable,

$$\begin{aligned} \langle \phi_{ij} \rangle &= \lim_{T \to 0} \frac{1}{2} \left\langle \tanh \left( \sum_{k \neq j} \frac{w_{ik}}{T} s_k + \frac{w_{ij}}{T} \right) - \tanh \left( \sum_{k \neq j} \frac{w_{ik}}{T} s_k - \frac{w_{ij}}{T} \right) \right\rangle \\ &= \frac{1}{2} \mathrm{sign} \left( \sum_{k \neq j} w_{ik} \langle s_k \rangle + w_{ij} \right) - \frac{1}{2} \mathrm{sign} \left( \sum_{k \neq j} w_{ik} \langle s_k \rangle - w_{ij} \right). \end{aligned} \tag{24}$$

Updates of each $\langle s_j \rangle$ for $j \in \mathcal{H}$ may then be undertaken for each pattern $n$ in the training set. Upon presentation of each example Eqn (23) may be applied to each node in random order until a stable solution is reached.

# 4 Learning

In the last section we discussed how inference of latent variables (or hidden units) may be performed in networks of LTUs. We now turn to the learning of the parameters $\mathbf{W}$.

The variational lower bound on the log likelihood may be written (from bound (9) substituting in Eqn (4), Eqn (6) and Eqn (10))

$$\ln P(\mathcal{O}|\mathcal{I}) \geq \sum_i \langle E_i \rangle - \sum_i \langle \ln \left( 1 + \exp(E_i(s_i)) \right) \rangle + \sum_i \mathcal{H} \left( Q(s_i) \right) \stackrel{\mathrm{def}}{=} \mathcal{L}. \tag{25}$$

We may treat the second term of the bound as in bound (13). Substituting in Eqn (3) and noting that the entropy term will always go to zero in the zero

temperature limit. We obtain

$$\mathcal{L} = \lim_{T \to 0} \sum_{ij} \frac{w_{ij}}{2T} \langle s_j \rangle \left( \langle s_i \rangle - \left\langle \langle s_i \rangle_{Q^-} \right\rangle \right). \tag{26}$$

Clearly this limit does not exist unless $\langle s_i \rangle = \left\langle \langle s_i \rangle_{Q^-} \right\rangle$ for all $i$.

Consider the function $T\mathcal{L}$

$$T\mathcal{L} = \sum_{ij} \frac{w_{ij}}{2} \langle s_i \rangle \langle s_j \rangle - \sum_{ij} \frac{w_{ij}}{2} \langle s_i \rangle \left\langle \langle s_j \rangle_{Q^-} \right\rangle. \tag{27}$$

First of all note that both terms take the form $\sum_{ij} \langle s_i \rangle w_{ij} x_j$. This form is maximised, for bi-valued $x$, if $x_j = \text{sign}\left(\sum_i w_{ij} \langle s_i \rangle\right)$. Now recall that the form of $\left\langle \langle s_j \rangle_{Q^-} \right\rangle = \text{sign}\left(\sum_i w_{ij} \langle s_i \rangle\right)$. Therefore Eqn (27) and Eqn (26) are bounded from above by zero. As we mentioned before, the limit in Eqn (26) only exists if $\langle s_i \rangle = \text{sign}\left(\sum_i w_{ij} \langle s_i \rangle\right)$ for all nodes in the latent set, $\mathcal{H}$, and the output (or target) set, $\mathcal{O}$. In other words the limit only exists when the pattern is classified correctly and all hidden nodes have their expected values. If this is the case the true likelihood for that pattern is 1, and our bound on the log-likelihood (which is $0 = \ln 1$) is exact. The objective in learning therefore is to adapt the parameters $\mathbf{W}$ so that this is the case. Of course, we wish to make this the case for every pattern in the training set so, reintroducing the index over the $N$ patterns, we really wish to optimise

$$\mathcal{L} = \lim_{T \to 0} \sum_{n=1}^{N} \sum_{ij} \frac{w_{ij}}{2T} \left\langle s_j^{(n)} \right\rangle \left( \left\langle s_i^{(n)} \right\rangle - \left\langle \left\langle s_i^{(n)} \right\rangle_{Q^-} \right\rangle \right). \tag{28}$$

We can envisage situations where the limit cannot exist for any parameterisation of $\mathbf{W}$. Consider, for example, the case where an identical input pattern has two different labels. We make it our objective, therefore, to satisfy the above equation for as many patterns as we can given the constraints of our chosen model structure.

Optimisation of the weight parameters in Eqn (28) is in effect a constrained optimisation with $N$ constraints imposed on the weight parameters associated with each hidden and output node where, as long as the constraints are all satisfied, the value of the objective function is constant. For values of $\mathbf{W}$ that satisfy the constraints the likelihood is one, for all other values it is zero.

## 4.1 Perceptron Learning Rule

The constraints imposed lead to an optimisation problem at each node which is identical to that of the perceptron with Eqn (27) providing the perceptron criterion for each node (given the inferred values of $\langle s_i \rangle$). To form a historical connection, we first simply derive the perceptron learning rule for each node.

Gradient based optimisation of the weight parameters could be achieved through differentiation of Eqn (27) with respect to $w_{kl}$. Ignoring the dependence

of $\left\langle \langle s_i \rangle_{Q^-} \right\rangle$ upon the $w_{kl}$ we obtain

$$\frac{\partial T\mathcal{L}}{\partial w_{kl}} = \frac{\langle s_k \rangle \langle s_l \rangle - \left\langle \langle s_l \rangle_{Q^-} \right\rangle \langle s_l \rangle}{2}. \tag{29}$$

If the weights are adjusted by simple gradient descent the algorithm can be recognised as the perceptron learning rule.

The solution to the constrained optimisation, which is equivalent to learning a perceptron for every hidden and output node, is not unique[2], see Figure 2, and may not exist, see Figure 3. If the solution does not exist then the

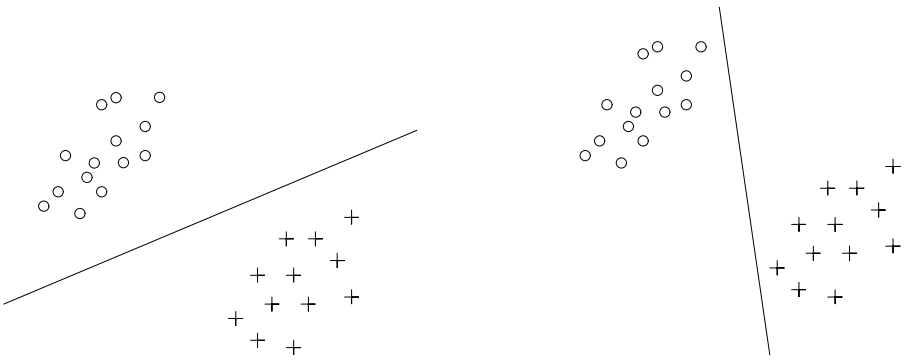

Figure 2: These two examples have decision boundaries which would both represent solutions to the linear programming problem.

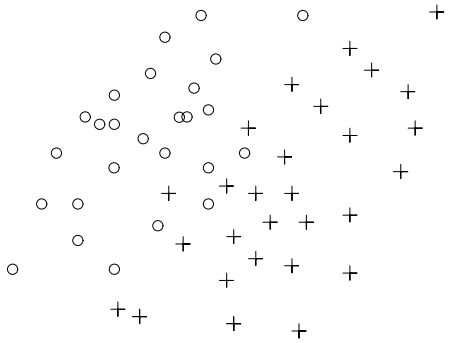

Figure 3: No linear decision boundary can be found which solves the linear constraints imposed by this data.

perceptron learning rule will fail to converge. Ideally we seek the solution which we expect to generalise best on unseen data. Therefore, rather than utilising the

[2]The objective function is zero once all constraints are satisfied, therefore any solution which satisfies the constraints is equally valid.

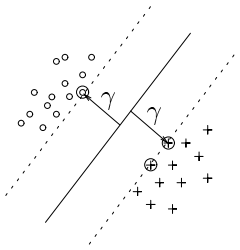


Figure 4: In linearly separable data, maximum margin provides a unique solution. Circled data-points indicate the support vectors and $\gamma$ indicates the size of the margin.

perceptron learning rule to optimise our weights, we utilise a maximum margin approach to finding a solution.

## 4.2 Maximum Margin

A maximum margin classifier, also known as the optimal perceptron, is a technique which can be justified from the perspective of statistical learning theory. In this work we only give a brief overview of these classifiers, for a more detailed introduction see Burges (1998).

### 4.2.1 Statistical Learning Theory

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 of such classifiers which hold with a certain confidence. These bounds are functions of the model capacity. The principle of structural risk minimisation is to minimise these bounds, through capacity reduction, to obtain the best classifier. This is the underlying theory of maximum margin classification. The margin size can be defined as the minimum distance the classification boundary needs to be moved for an example to be misclassified. The model capacity is known to be inversely related to margin size. Increasing the margin decreases the capacity and hence should lead to a classifier with better generalisation properties. Additionally, the objective function in the region where the constraints have been satisfied becomes convex, and it can be shown that the solution for the weight parameters becomes unique.

The maximum margin solution, shown in Figure 4, is sometimes 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 through quadratic programming.

There remains the issue of separability. If the problem is not linearly separable, there will be no solution for the standard formulation of the quadratic programming problem. 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, $\xi_n$, represents the

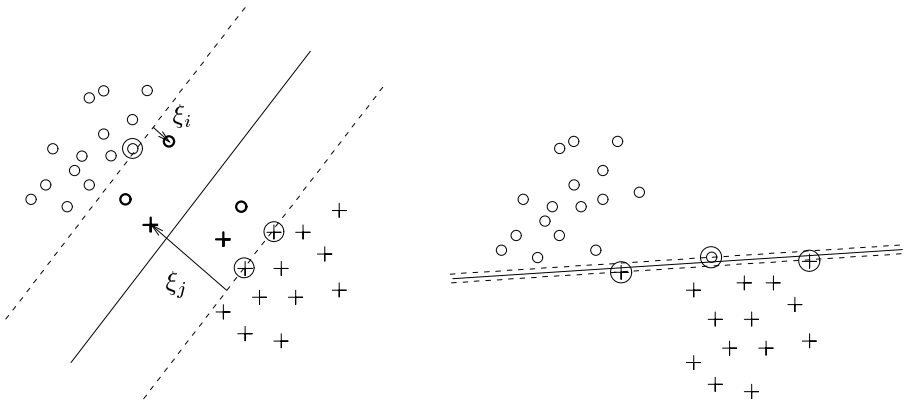


Figure 5: Depending on the penalty assigned to errors, $C$, different solutions will be found. The left hand figure is a low error penalty, the right hand figure is a high error penalty. Bold data-points indicate 'errors' and $\xi_i$ represents the extent of the error on data-point $i$.

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 5). The solution found will naturally then be sensitive to the choice of $C$.

# 5 Algorithm Overview and Implementation

The algorithm as we have described it involves two steps. The first step involves updating the expected value of the hidden nodes, $\left\langle s_i^{(n)} \right\rangle$, for each data point, $n$. The next step is to update the weight parameters, $\{w_{ij}\}$, by a quadratic programming problem, given the expected values of the variables.

## 5.1 Some Algorithm Heuristics

Having described the fundamental theory behind our algorithm, we also wish to mention some heuristics which were found useful in the practical implementation of the algorithm.

### 5.1.1 Simulated annealing

The updates of Eqn (23) 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 $\langle s_j \rangle$, gradually reducing the temperature to zero (see Waugh *et al.* (1990) for some theoretical justification of this approach).

### 5.1.2 Expectation Initialisation

The values for $\langle s_j \rangle$ for $j \in \mathcal{H}$ may be initialised with the values of $s_j$ obtained from a 'forward pass' through the network.

### 5.1.3 Convergence

The algorithm converges when every training pattern has been classified exactly. However, it is quite possible for the training error, in terms of the number of mis-classifications, to increase after the inference and learning steps. This may occur despite the overall objective function going down as it may coincide with an increase in the number of hidden nodes which are 'classified correctly'. It would therefore seem prudent to store the network best training error achieved at all times during learning and, in the case that learning doesn't converge (e.g. for the reasons outlined in Section 4) to utilise this network for the solution.

### 5.1.4 Weight Normalisation

The inference step for a node (Eqn (23)) involves a sum across the weights which propagate out from the node and a sum across weights which feed into the node. The positioning of the separating hyper-plane which is defined by the weights which feed into the node is invariant to a rescaling of the weights. Envisage, therefore, a situation where the magnitude of the weights which feed into the node is much larger than the magnitude of the weights which propagate out (this can be achieved without affecting the mapping which the network represents between the input set and the output set). The inferred value of the node as given by Eqn (23) will then be dominated by the variables which feed into the node. This situation is far from ideal. In our experiments, therefore, we implemented a heuristic where the values of the weights which propagate into a node are normalised by the magnitude of the vector of the weights which propagate out from the node.

### 5.1.5 Relaxing Quadratic Programming Convergence Conditions

Maximising the margin can be an expensive computation. In the early stages of the algorithm, when we expect our inference about the values of the latent nodes to be 'less correct', we may not wish to spend undue computational effort on precise optimisation of the margin. In our implementation we relaxed the conditions for convergence of the maximum margin solutions for the first four iterations. To be precise, we normally required the Karus-Kuhn-Tucker[3] (KKT) conditions to be satisfied to an accuracy of $1 \times 10^{-4}$, but for the first four iterations we replaced this equation with $1 \times 10^{-(k-1)}$ where $k$ is the iteration number. Note that for the first iteration, this is convergence of the KKT to within 1. In other words no maximisation of the margin is required for the first iteration.

[3] See (Burges, 1998) for details of these convergence conditions.

# 6 Experimental Assesment

In this section we apply the algorithm to three data-sets. The first two data-sets are well known benchmarking data-sets. The third data-set is from a medical imaging problem involving the analysis of fluorescent in-situ hybridisation (FISH) images. For these experiments, updates were undertaken in the form of a full inference step, i. e. updates of $\{s_i^{(n)}\}$ to convergence, followed by learning of the parameters $\mathbf{W}$ to convergence using the sequential minimal optimisation (SMO) algorithm (Platt, 1998) to solve the quadratic programming problem. The process of inference and learning was repeated seven times for all data-sets.

## 6.1 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 332 points. We normalised the data to be zero mean and lie between $-1$ and 1. We then trained networks with differing error penalties, $C = 10^k$ for $k = 0, 1$ and 2, and different numbers of hidden units, $H = 10, 15$ and 20. Ten different initialisations were used for each parameterisation. We limited the margin optimisation to a maximum of 100,000 iterations of Platt's SMO algorithm. Simulated annealing was implemented for updating hidden unit expectations. The temperature was initialised at 1000 and the decay rate was set to 0.8, i. e. at iteration $k$ the temperature was $1000 \times 0.8^{k-1}$. Once the temperature dropped below 0.001 it was taken to be zero. Once annealing was complete, a maximum of a further 70 passes updating the expectations of the latent nodes were permitted.

Results are summarised in Figure 6 and Table 1. The other presented results are taken from Ripley (1996) and Barber and Williams (1997).

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.

## 6.2 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 consisted of 80 training points and 120 test points. The data was tested in the same manner as the Pima Indians data-set, but with different numbers of

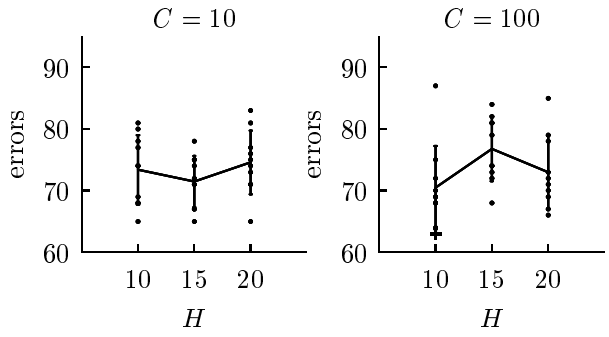


Figure 6: Results for the networks on the Pima Indians data-sets for $C = 10$ and $C = 100$. Dots represent each of the ten different networks initialisations. The network which achieved the overall minimum error is marked with +. Also shown is the mean and error-bars at one standard deviation for each parameterisation.

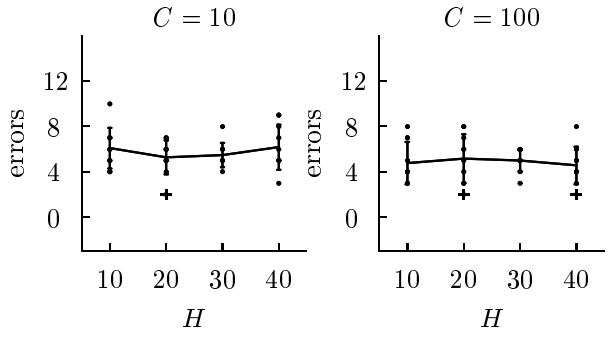


Figure 7: Results for the networks on the Leptrograpsus crabs data-sets for $C = 10$ and $C = 100$. Dots represent each of the ten different networks initialisations. The networks which achieved the overall minimum error are marked with +. Also shown is the mean and error-bars at one standard deviation for each parameterisation.

hidden units $H = 10, 20, 30$ and 40. The results are depicted in Figure 7, and the error rates are shown in Table 1.

Similar conclusions may be drawn from the results on the crabs data as were drawn for those on the Pima Indians 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.

### 6.3 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 hybridisation 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 la-

| 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 |
| Average Linear Threshold Unit | 73.97 | 6.08 |
| Best Linear Threshold Unit | 63 | 2 |

Table 1: Classification rates for the Pima Indians and Leptrograpsus crabs data-sets. The table shows some of the obtained results on the Pima Indians and Leptrograpsus crabs data-sets. The results labeled 'Best Linear Threshold Unit' are those which obtained the minimum errors on the test sets. Their parameterisations are depicted in Figure 7 and Figure 6. The result labeled 'Average Linear Threshold Unit' is the average of all the networks' classification errors.

belled 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 the LTU models as the results are initialisation dependent. The inputs were normalised to have zero mean and unit standard deviation for these experiments. The results of the FISH experiments are summarised in Table 2 and Figure 8.

The results for the linear threshold units stand up well in comparison to the other classifiers. The best performing linear threshold unit network gives results in line with that of the support vector machine. The committee of networks also gives a competitive result. The single network selected by utilising the validation set gives equivalent performance to a standard 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. There is some chance that performance could be improved by further training of the selected models on the validation set.

# 7 Discussion

In this work we have presented an algorithm for learning in networks of linear threshold units. We demonstrated the algorithm's implementation with a variety of data-sets.

The algorithm has an interesting parallel with the expectation-maximisation (EM) algorithm (Dempster *et al.*, 1977), the 'expectation' step would be the de-

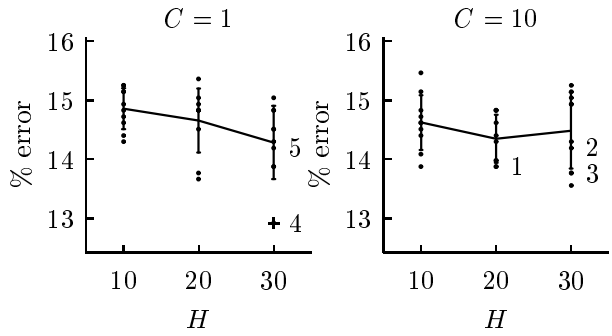


Figure 8: Results for the networks on the FISH data-sets for $C = 1$ and $C = 10$. Dots represent each of the ten different networks initialisations. The network which achieved the overall minimum error is marked with +. The networks with the five lowest validation set error are marked in order 1 to 5, with 1 having the lowest validation error. Also shown is the mean and error-bars at one standard deviation for each parameterisation.

| model | Classification Rate |
|---|---|
| Naive Bayesian Network | 17.0 % |
| Neural Network | 13.6 % |
| Bayesian Neural Network | 11.8 % |
| Support Vector Machine | 12.8 % |
| Linear Threshold Unit (Best) | 12.9 % |
| Linear Threshold Unit (Average) | 14.6 % |
| Linear Threshold Unit (Validation Set) | 13.9 % |
| Linear Threshold Unit (Committee) | 13.2 % |

Table 2: Classification rates for the FISH data. The table shows some of the obtained results on the FISH data-set. The result labelled 'Best' is that which obtained the minimum error on the test sets, the 'Average' result was the average error for all networks, the result labelled 'Validation Set' is that selected through the validation set and finally that labelled 'Committee' is a result from an unweighted committee of five networks selected according to the validation set error.

termination of the parameters $\{\langle s_i \rangle\}$ and the 'maximisation' step is a quadratic program in $\mathbf{W}$. It not strictly correct, however, to call the fixed point equation update of $\left\langle s_i^{(n)} \right\rangle$ 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.

We have considered only linear threshold units in this work. 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 $\langle s_i \rangle$ 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 criterion (Rosenblatt, 1962) and the weight update is the associated learning algorithm.

Finally this approach may provide a method of performing inference in expert systems which use `if ... then` rules for their knowledge base. Such a system can be viewed as the zero temperature limit of a DAG based on probability tables, just like networks of linear threshold units are the zero temperature limit of the sigmoid belief network. We could therefore apply this approach to performing inference in such systems.

## Acknowledgements

We would like to thank Boaz Lerner for providing the FISH data-set used in these experiments.

## 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.

Haft, M., R. Hoffmann, and V. Tresp (1999). Model-independent mean field theory as a local method for approximate propagation of information. *Network: Computation in Neural Systems* **10**, 93–105.

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. To appear in *Neural Computing and Applications*.

MacKay, D. J. C. (1995). Developments in probabilistic modelling with neural networks—ensemble learning. In *Neural Networks: Artificial Intelligence and Industrial Applications. Proceedings of the 3rd Annual Symposium on Neural Networks, Nijmegen, Netherlands, 14-15 September 1995*, pp. 191–198. Berlin: Springer.

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*, Series F: Computer and Systems Sciences, pp. 105–126. Springer-Verlag.

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.

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.