Polynomial Softmax Functions for Pattern Classification

A. Turk, S.J. Young

CUED/F-INFENG/TR ???

January 2001

Cambridge University Engineering Department
Trumpington Street
Cambridge CB2 1PZ
England

E-mail: `at233@eng.cam.ac.uk, sjy@eng.cam.ac.uk`

Contents

1 Introduction **3**

2 Basic properties of the softmax functions **4**

3 Strict convexity and the minimisation of the error function **6**
3.1 Convexity of the error function 6
3.2 Minimisation of the error function 11

4 Practical parameter reestimation **12**
4.1 The one-dimensional two-class problem 13
4.2 Higher dimensional 2 class problems 18
4.3 Higher dimensional multi class problems 22

5 Conclusions and further work **26**

1 Introduction

In the neural network literature softmax functions are typically used if the output units of a neural network have to be interpreted as posterior probabilities. In this situation softmax functions are the activation units of the final layer [5, 7], and the outputs of the neural net are therefore given by

$$\frac{\exp(a_k)}{\sum_j \exp(a_j)} \tag{1}$$

Here a_j is the j-th output of the neural network before applying the softmax function. The definition of the softmax functions ensure that the network outputs lie between 0 and 1 and that their sum is equal to 1. This justifies their interpretation as probabilities. The weights of such a neural network are then optimised with respect to the cross-entropy criterion which results in a tractable reestimation theory [4, 5]. The main motivation for studying neural networks as input to softmax functions is their great versatility and the possibility of approximating a function with arbitrary accuracy as long as the neural network has enough hidden layers and nodes. However, this also means that such a model shares the same problems as all neural networks. For instance, it is not clear if the error surface has any local minima which correspond to suboptimal choices of parameters, and if there exists a global minimum at all. This problem has, for instance, been discussed in [2].

The present paper proposes an alternative to the combination of general neural networks and softmax functions that substitutes the network outputs with independent polynomials or more general linear combinations of non-linear functions. In [5] an example has been discussed where the input to the softmax functions was a polynomial, however, this polynomial depended non-linearly on the model parameters and is therefore not covered by the theory developed in this report. As a means to solve classification problems, polynomials have also been studied in the Volterra connectionist model [10, 8]. Here the polynomials are directly applied to the observation vectors without the subsequent use of softmax functions. Since the polynomials can take any values this means that the problem of assigning proper classification indices to the training data has to be addressed [11]. This is not necessary for softmax functions because, here, the training data are labelled, in the usual way, by assigning a probability of 1 to the class the data point belongs to. Although, as compared to the Volterra model, there is no closed form solution for the optimal set of parameters of a set of polynomial softmax functions it will be shown in section 3 that under very general assumptions on the training data the cross-entropy error surface is always strictly convex and that there exists a unique global minimum. Section 4 discusses the practical implications of these results and gives several examples of the usefulness of polynomial softmax functions. It will, for instance, be shown that even if the assumptions in section 3 are not fulfilled, it is still possible to find a sequence of softmax functions that approximate the training data with arbitrary accuracy. Furthermore, polynomials will be shown to exist that solve classification problems that are normally used to motivate the use of neural networks with two or more hidden layers. In addition, these polynomials use a similar number of parameters as a neural network that is appropriate for such a classification task. Another advantage of polynomials as compared to neural networks as input to softmax functions is that their "topology" is much simpler. Therefore, to solve a particular classification problem only the degree of the polynomial has to be considered. However, in section 4.3, the possibility of introducing a "topology" into a set of more general probability functions will be suggested by using algebraic combinations of polynomial softmax functions.

2 Basic properties of the softmax functions

In this paper softmax functions will be applied to various classification problems. The data set that defines a K-class classification problem consists of a set of points $x_n, n = 1, \ldots, N$ in the observation space and a set of posterior probabilities $p_{n,k}$ for each point x_n and each class $k = 1, \ldots, K$. The $p_{n,k}$ therefore have to satisfy

$$\sum_{k=1}^{K} p_{n,k} = 1 \tag{2}$$

for each point x_n. Such data sets include the classical situation where one has a set of observations x_n that are labelled according to their class-membership. In this case $p_{n,k}$ is 1 if k is the label of the class x_n belongs to and 0 otherwise. One can further enlarge this data set by adding positive weights w_n to each point x_n. These weights can be interpreted as the importance of the data point x_n in the classification problem and will later be used in the definition of the error function. Data sets of this kind will be denoted by (x, p, w) in the following, i.e.

$$(x, p, w) = \{x_n, p_{n,k}, w_n : n = 1, \ldots, N \wedge k = 1, \ldots, K\} \tag{3}$$

and they will be referred to as d-dimensional K-class problems, where d is the dimension of the x_n. Given a data set (x, p, w), the task is to model the relationship between the x_n and the $p_{n,k}$ as accurately as possible and thereby reliably predict the class membership of the x_n. Here, the relationship between the x_n and the $p_{n,k}$ will be modelled by softmax functions with polynomial exponents. Later this frame-work will be extended to include exponents of a more general form. For a K-class problem these softmax functions are defined by

$$S_k(x) = \begin{cases} \frac{e^{q_k(x)}}{1+\sum_{j=1}^{K-1} e^{q_j(x)}} & : \quad 1 \leq k \leq K-1 \\ \frac{1}{1+\sum_{j=1}^{K-1} e^{q_j(x)}} & : \quad k = K \end{cases} \tag{4}$$

where the $q_j(x)$ are polynomials, i.e.

$$q_j(x) = \sum_{l=0}^{L_j} a_{j,l} x^l \tag{5}$$

Here $a_{j,l}$ are the coefficients of the j-th polynomial, L_j is its degree, and l can be a multi-index if the dimension of x is greater than one. The set of all model parameters will from now on be denoted by $\mathbf{a}$, i.e.

$$\mathbf{a} = \{a_{j,l} : 1 = 2, \ldots, K-1 \wedge l = 0, \ldots, L_j\} \tag{6}$$

It is immediate from (4) that the softmax functions satisfy

$$\sum_{k=1}^{K} S_k(x) = 1 \quad \text{and} \quad 0 < S_k(x) < 1 \tag{7}$$

for each x. Therefore the numbers $S_k(x)$, $k = 1, \ldots, K$ constitute a probability distribution for each x. Before tackling the problem of reestimating the model parameters it is necessary to define more precisely the criterion which characterises the ideal set of model parameters. In this work the criterion will be linked to an error function that is the sum of the K-L distances between the true and the estimated posteriors at each point of the training set, i.e.

$$E(x, p, \mathbf{a}) = \sum_{n=1}^{N} \sum_{k=1}^{K} p_{n,k} \log \left(\frac{p_{n,k}}{S_k(x_n)} \right) \tag{8}$$

This error function is called the cross-entropy error function and is closely related to the MMI criterion [1, 12] that is used in speech recognition to train the parameters of a set of hidden Markov models. Additionally, one can weight the K-L distance in (8) at the individual data points with the weights of the classification problem w_n which results in the following more general error function.

$$E(x, p, w, \mathbf{a}) = \sum_{n=1}^{N} w_n \sum_{k=1}^{K} p_{n,k} \log \left(\frac{p_{n,k}}{S_k(x_n)} \right) \tag{9}$$

This allows the case where the x_n are samples of a non-uniform distribution to be handled. The information about the distribution can be included into the search for an optimal set of parameters by setting w_n to the value of the distribution at x_n. In this way, the error function puts a high weight on errors in areas where the value of the density is high. Assuming that,

$$0 \log(0) = 0 \tag{10}$$

the error function (9) is always positive and equals zero only if the approximating functions model the training data perfectly. The optimal parameter set $\mathbf{a}$ will be the one that minimises the error function (9). Since the entropy of the probability distribution $p_{n,k}, k = 1, \ldots, K$ at each point x_n is not affected by changes in the parameter set $\mathbf{a}$, minimising (9) is equivalent to minimising the following expression

$$-\sum_{n=1}^{N} w_n \sum_{k=1}^{K} p_{n,k} \log(S_k(x_n)) \tag{11}$$

A necessary condition for a set of parameters $\mathbf{a}$ to minimise the error function is that the gradient vanishes at such a point. For a component $a_{j,l}$ in $q_j(x)$ the partial derivative of the error function is given by

$$\begin{aligned}
\frac{\partial}{\partial a_{j,l}} E(x, p, w, \mathbf{a}) &= \frac{\partial}{\partial a_{j,l}} \sum_{n=1}^{N} w_n \sum_{k=1}^{K} p_{n,k} \log \left(\frac{p_{n,k}}{S_k(x_n)} \right) \\
&= -\sum_{n=1}^{N} w_n \sum_{k=1}^{K} p_{n,k} \frac{\partial}{\partial a_{j,l}} \log(S_k(x_n)) \\
&= -\sum_{n=1}^{N} w_n \sum_{k=1}^{K} p_{n,k} \left(\frac{\partial}{\partial a_{j,l}} q_k(x_n) - \frac{\partial}{\partial a_{j,l}} \log \left(1 + \sum_{i=1}^{K-1} e^{q_i(x_n)} \right) \right) \\
&= -\sum_{n=1}^{N} w_n (p_{n,j} - S_j(x_n)) \frac{\partial}{\partial a_{j,l}} q_j(x_n)
\end{aligned} \tag{12}$$

The gradient of the error function with respect to the model parameters $\mathbf{a}$ is therefore given by

$$\nabla E(x,p,w,\mathbf{a}) = \left\{ \sum_{n=1}^{N} w_n (S_j(x_n) - p_{n,j}) \frac{\partial}{\partial a_{j,l}} q_j(x_n) : j = 2,\ldots,K-1 \wedge l = 0,\ldots,L_j \right\} \tag{13}$$

Here it was used that $a_{j,l}$ is a parameter that appears only in $q_j(x)$ and therefore the following holds

$$\frac{\partial}{\partial a_{j,l}} q_k(x) = 0 \qquad \text{if} \qquad k \neq j \tag{14}$$

Apart from the independence of the parameters of different $q_j(x)$'s, the derivation of equation (12) did not make use of the special form of the $q_j(x)$ and is therefore valid for any function $q_j(x)$ which is partially differentiable with respect to one of its parameters $a_{j,l}$. If $q_j(x)$ is a polynomial of the form (5) then equation (12) becomes

$$\frac{\partial}{\partial a_{j,l}} E(x,p,w,\mathbf{a}) = \sum_{n=1}^{N} w_n (S_j(x_n) - p_{n,j}) x_n^l \tag{15}$$

This is the l-th moment of the weighted difference between $p_{n,j}$ and $S_j(x_n)$. Since the gradient has to vanish at a local minimum of the error function the moments up to order L of the weighted difference have to vanish at such a point. Note that the requirement that the gradient of the error function is 0 has a similar form as the requirement that a polynomial minimises the sum of square error on the training data. However, since the $S_j(x)$ do not depend linearly on the parameters $a_{j,l}$ there is no closed form solution for the optimal set of parameters in this case. This means that an iterative scheme has to be employed to find a solution. This will be discussed in section 4.

3 Strict convexity and the minimisation of the error function

This section will investigate some special properties of the error function $E(x,p,w,\mathbf{a})$ that are important with respect to the optimisation of the parameter set $\mathbf{a}$. In particular it will discuss matrices that arise as second order derivatives of the error function $E(x,p,w,\mathbf{a})$ and will analyse them in terms of their associated inner products.

3.1 Convexity of the error function

As can be seen from (12), the second order derivatives of the error function require the derivatives of the softmax functions. These can be calculated as follows.

$$\frac{\partial}{\partial a_{j,l}} S_k(x) = \frac{e^{q_k(x)} \frac{\partial}{\partial a_{j,l}} q_k(x) (1 + \sum_{k=1}^{K-1} e^{q_k(x)}) - e^{q_k(x)} \sum_{k=1}^{K-1} e^{q_k(x)} \frac{\partial}{\partial a_{j,l}} q_k(x)}{(1 + \sum_{k=1}^{K-1} e^{q_k(x)})^2} \tag{16}$$

For $k = j$, equation (16) becomes

$$\frac{\partial}{\partial a_{j,l}} S_j(x) = \left(\frac{e^{q_j(x)}}{1 + \sum_{k=1}^{K-1} e^{q_k(x)}} - \frac{(e^{q_j(x)})^2}{(1 + \sum_{k=1}^{K-1} e^{q_k(x)})^2} \right) \frac{\partial}{\partial a_{j,l}} q_j(x) \tag{17}$$

and for $k \neq j$ equation (16) reduces to

$$\frac{\partial}{\partial a_{j,l}} S_k(x) = -\frac{e^{q_k(x)} e^{q_j(x)}}{(1 + \sum_{k=1}^{K-1} e^{q_k(x)})^2} \frac{\partial}{\partial a_{j,l}} q_j(x) \tag{18}$$

Substituting the definitions of the softmax functions, these last two equations can be reformulated as follows

$$\frac{\partial}{\partial a_{j,l}} S_k(x) = \begin{cases} (S_k(x) - S_k(x)^2) \frac{\partial}{\partial a_{j,l}} q_j(x) & : \ j = k \\ \\ -S_j(x) S_k(x) \frac{\partial}{\partial a_{j,l}} q_j(x) & : \ j \neq k \end{cases} \tag{19}$$

Under the assumption that the following holds

$$\frac{\partial}{\partial a_{j,l} a_{j,l'}} q_j(x) = 0 \tag{20}$$

the second order derivatives of the error function $E(x, p, w, \mathbf{a})$ are therefore given by

$$\frac{\partial}{\partial a_{j,l} a_{k,l'}} E(x, p, w, \mathbf{a}) = \begin{cases} \sum_{n=1}^{N} w_n (S_k(x_n) - S_k(x_n)^2) \frac{\partial}{\partial a_{k,l}} q_k(x_n) \frac{\partial}{\partial a_{k,l'}} q_k(x_n) & : \ j = k \\ \\ -\sum_{n=1}^{N} w_n S_k(x_n) S_j(x_n) \frac{\partial}{\partial a_{j,l}} q_j(x_n) \frac{\partial}{\partial a_{k,l'}} q_k(x_n) & : \ j \neq k \end{cases} \tag{21}$$

Integrating (20) one can see that the most general $q_j(x)$ for which the derivations up until equation (21) are valid are given by

$$q_j(x) = \sum_{l=0}^{L_j} a_{j,l} \phi_{j,l}(x) + \phi_j(x) \tag{22}$$

Because of (14) $a_{j,l}$ and $a_{k,l'}$ are independent of each other if $j \neq k$, and $\phi_{j,l}(x)$ and $\phi_j(x)$ are functions independent of the $a_{j,l}$. In keeping with the terminology of polynomials, L_j will be called the degree of $q_j(x)$. Due to the special form of the $q_j(x)$ in (22), equation (21) can be rewritten as

$$\frac{\partial}{\partial a_{j,l} a_{k,l'}} E(x, p, w, \mathbf{a}) = \begin{cases} \sum_{n=1}^{N} w_n (S_k(x_n) - S_k(x_n)^2) \phi_{k,l}(x_n) \phi_{k,l'}(x_n) & : \ j = k \\ \\ -\sum_{n=1}^{N} w_n S_k(x_n) S_j(x_n) \phi_{j,l}(x_n) \phi_{k,l'}(x_n) & : \ j \neq k \end{cases} \tag{23}$$

For the special case where the $q_j(x)$ are polynomials, equation (21) becomes

$$\frac{\partial}{\partial a_{j,l} a_{k,l'}} E(x, p, w, \mathbf{a}) = \begin{cases} \sum_{n=1}^{N} w_n (S_k(x_n) - S_k(x_n)^2) x_n^{l+l'} & : \ j = k \\ \\ -\sum_{n=1}^{N} w_n S_k(x_n) S_j(x_n) x_n^{l+l'} & : \ j \neq k \end{cases} \tag{24}$$

Equation (23) gives the components of the second order derivative, which is also called the Hessian, of the error function $E(x, p, w, \mathbf{a})$ regarded as a function of $\mathbf{a}$. This matrix will be denoted by $H(x, p, w, \mathbf{a})$ and abbreviated as H if the particular classification problem (x, p, w)

is not important. For a two class problem with an exponential function $q(x)$ of degree L the second order derivative of the error function $E(x, p, w, \mathbf{a})$ is given by the following matrix

$$H = \begin{pmatrix} \sum_{n=1}^N c_n \phi_0(x_n)^2 & \sum_{n=1}^N c_n \phi_0(x_n)\phi_1(x_n) & \cdots & \sum_{n=1}^N c_n \phi_0(x_n)\phi_L(x_n) \\ \sum_{n=1}^N c_n \phi_1(x_n)\phi_0(x_n) & \sum_{n=1}^N c_n \phi_1(x_n)^2 & \cdots & \sum_{n=1}^N c_n \phi_1(x_n)\phi_L(x_n) \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \sum_{n=1}^N c_n \phi_L(x_n)\phi_0(x_n) & \sum_{n=1}^N c_n \phi_L(x_n)\phi_0(x_n) & \cdots & \sum_{n=1}^N c_n \phi_L(x_n)^2 \end{pmatrix} \tag{25}$$

Here the c_n are defined as follows

$$c_n = w_n(S_1(x_n) - S_1(x_n)^2) = w_n S_1(x_n) S_2(x_n) \tag{26}$$

Since w_n is positive and $0 < S_k(x) < 1$ holds for all x, the c_n are always positive. Given this fact, it is straight forward to show that the matrix H is always positive semi-definite. This holds because H can be rewritten in the following form

$$H = \sum_{n=1}^{N} c_n \Phi(x_n)' \Phi(x_n) \tag{27}$$

where $\Phi(x_n)$ is the vector given by

$$\Phi(x_n) = (\phi_0(x_n), \ldots, \phi_L(x_n))' \tag{28}$$

Applying the quadratic form that is represented by H to a vector v results therefore in

$$vHv' = \sum_{n=1}^{N} c_n < \Phi(x_n), v >^2 \tag{29}$$

where $< \cdot, \cdot >$ is the standard Euclidean inner product. Since the c_n are positive this shows that H is always positive semi-definite. If there exist $L+1$ linearly independent $\Phi(x_n)$ the quadratic form H is strictly positive definite. This shows that for a two class problem of arbitrary dimension the error surface $E(x, p, w, \mathbf{a})$ is always convex and strictly convex if there are $L+1$ linearly independent $\Phi(x_n)$.

For a multi-class problem where the number of classes is larger than 2 the Hessian of the error function has a more complex form. For a 3-class problem, for instance, the Hessian is given by

$$H = \begin{pmatrix} H_{11} & H_{12} \\ H_{12}' & H_{22} \end{pmatrix} \tag{30}$$

Here H_{11} is an $L_1 + 1 \times L_1 + 1$ matrix and H_{22} is an $L_2 + 1 \times L_2 + 1$ matrix of the form (25), where L_1 is the degree of q_1 and L_2 is the degree of q_2. For matrix H_{11} the c_n, which will be denoted as h_{n11} are given by

$$h_{n11} = c_n = w_n(S_1(x_n) - S_1(x_n)^2) \tag{31}$$

and for matrix H_{22} the c_n, which will be denoted as h_{n22}, are given by

$$h_{n22} = c_n = w_n(S_2(x_n) - S_2(x_n)^2) \tag{32}$$

Matrix H_{12} is an $L_1+1 \times L_2+1$ matrix of the following form

$$H_{12} = \left(\begin{array}{cccc} \sum_{n=1}^{N} c_n \phi_0(x_n)^2 & \sum_{n=1}^{N} c_n \phi_0(x_n)\phi_1(x_n) & \cdots & \sum_{n=1}^{N} c_n \phi_0(x_n)\phi_{L_2}(x_n) \\ \sum_{n=1}^{N} c_n \phi_1(x_n)\phi_0(x_n) & \sum_{n=1}^{N} c_n \phi_1(x_n)^2 & \cdots & \sum_{n=1}^{N} c_n \phi_1(x_n)\phi_{L_2}(x_n) \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \sum_{n=1}^{N} c_n \phi_{L_1}(x_n)\phi_0(x_n) & \sum_{n=1}^{N} c_n \phi_{L_1}(x_n)\phi_1(x_n) & \cdots & \sum_{n=1}^{N} c_n \phi_{L_1}(x_n)\phi_{L_2}(x_n) \end{array} \right) \tag{33}$$

for which the c_n, in the following denoted as h_{n12}, are given by

$$h_{n12} = c_n = -w_n S_1(x_n) S_2(x_n) \tag{34}$$

To show that matrix H is positive semi-definite it is convenient to decompose it, as in the case of a two-class problem, into a sum of matrices. The terms of the sum are the matrices with a fixed index n which will be denoted by H_n. These matrices are again block matrices whose blocks can be derived from matrices H_{11}, H_{12} and H_{22}. These blocks will be denoted as H_{n11}, H_{n12} and H_{n22} in the following. Matrix H_{n11} can therefore be written as

$$H_{n11} = h_{n11} \Phi_1(x_n)' \Phi_1(x_n) \tag{35}$$

and H_{n22} is given by

$$H_{n22} = h_{n22} \Phi_2(x_n)' \Phi_2(x_n) \tag{36}$$

where Φ_1 and Φ_2 are defined as follows

$$\Phi_1(x_n) = (\phi_{1,0}(x_n), \ldots, \phi_{1,L_1}(x_n)) \tag{37}$$
$$\Phi_2(x_n) = (\phi_{2,0}(x_n), \ldots, \phi_{2,L_2}(x_n)) \tag{38}$$
$$\tag{39}$$

Therefore, it follows that H_{n12} is given by

$$H_{n12} = h_{n12} \Phi_1(x_n)' \Phi_2(x_n) \tag{40}$$

The positive semi-definiteness of H_n will be shown by proving that the following holds

$$v H_n v' \geq 0 \tag{41}$$

where v is an arbitrary vector of dimension $L_1 + L_2 + 2$. To simplify the problem this vector will be written as

$$v = (v_1, v_2) \tag{42}$$

where v_1 are the first $L_1 + 1$ components of vector v and v_2 are the remaining $L_2 + 1$ components. Proving that (41) holds, therefore amounts to showing that the following is true

$$h_{n11} < \Phi_1(x_n), v_1 >^2 + 2h_{n12} < \Phi_1(x_n), v_1 >< \Phi_2(x_n), v_2 > + h_{n22} < \Phi_2(x_n), v_2 >^2 \geq 0 \tag{43}$$

Since the inner products $< \Phi_1(x_n), v_1 >$ and $< \Phi_2(x_n), v_2 >$ can take any value, proving equation (43) is equivalent to showing that the following matrix is positive semi-definite.

$$\mathbf{h}_n = \begin{pmatrix} h_{n11} & h_{n12} \\ h_{n12} & h_{n22} \end{pmatrix} \tag{44}$$

These considerations can easily be generalised to a K-class problem with more than 3 classes. In this case, proving that the individual terms of the second order derivative are positive semi-definite is equivalent to showing that matrix $\mathbf{h}_n = (h_{nij})_{i,j=1}^{K-1}$ is positive semi-definite. Here the h_{nij} are given by

$$h_{nij} = \begin{cases} S_i(x_n) - S_i(x_n)^2 & : \quad i = j \\ \\ -S_i(x_n)S_j(x_n) & : \quad i \neq j \end{cases} \tag{45}$$

Matrix $\mathbf{h}_n$ is diagonally dominant since

$$\begin{aligned} |h_{nii}| &= S_i(x_n) - S_i(x_n)^2 = S_i(x_n) \sum_{j=1, j\neq i}^{K} S_j(x_n) \\ &> S_i(x_n) \sum_{j=1, j\neq i}^{K-1} S_j(x_n) = \sum_{j=1, j\neq i}^{K-1} |h_{nij}| \end{aligned} \tag{46}$$

and since its diagonal elements are positive the matrix is positive definite. If there exists a j with $1 \leq j \leq K-1$ for which there are $L_j + 1$ data points x_n such that the $\Phi_j(x_n)$ are linearly independent then the second order derivative of the error function $E(x, p, w, \mathbf{a})$ is positive definite. Therefore, the following Theorem has been proven

Theorem 1 *The error function $E(x, p, w, \mathbf{a})$ of a K-class problem of arbitrary finite dimension, regarded as a function of the model parameters $\mathbf{a}$, is always convex. If there exists a j with $1 \leq j \leq K-1$ for which there are at least $L_j + 1$ data points x_n such that the $\Phi_j(x_n)$ are linearly independent then the error function $E(x, p, w, \mathbf{a})$ is strictly convex.*

If the observation space is one-dimensional and the functions $q_j(x)$ are polynomials then the linear independence of $L_j + 1$ vectors $\Phi_j(x_n)$ means that the Vandermonde matrix has to be non-degenerate. This again is equivalent to the requirement that there are $L_j + 1$ different x_n. Therefore the following corollary holds

Corollary 1 *The error function $E(x, p, w, \mathbf{a})$ of a one-dimensional K-class problem is strictly convex if the $q_j(x)$ are polynomials and there exist at least $\min_j L_j + 1$ different x_n.*

The operations that are necessary to calculate H_n for a K-class problem involve the calculation of the products $\Phi_i(x_n)'\Phi_j(x_n)$ and of the h_{nij}. If the functions $\phi_{j,l}(x)$ are the same for different j then only the products $\Phi_j(x_n)'\Phi_j(x_n)$ have to be calculated for the j whose function $q_j(x)$ has the highest degree. The total number of operations for these products is therefore $2\max_j L_j + 1$. Adding the number of operations necessary for the h_{nij} therefore shows that the total number of operations necessary to calculate H_n is

$$2 \max_j L_j + 1 + \frac{K(K-1)}{2} \tag{47}$$

This means that the number of operations to calculate H_n is of the order of K^2 and of the order of $\max_j L_j$, which is a considerable reduction in the number of necessary operations as compared to a general neural network where it is of the order of the squared number of weights [3, 6].

3.2 Minimisation of the error function

The previous section showed that the error function for a multi-class problem of arbitrary finite dimension is always convex. This means that the following holds

$$\lambda_1 E(x,p,w,\mathbf{a_1}) + \cdots + \lambda_n E(x,p,w,\mathbf{a_n}) \geq E(x,p,w,\lambda_1\mathbf{a_1} + \cdots + \lambda_n\mathbf{a_n}) \tag{48}$$

where $\mathbf{a_i}$ are arbitrary points in the parameter space and $\lambda_1 + \cdots + \lambda_n = 1$. If the error function is strictly convex the above inequality is strict. This implies that there can at most exist one local minimum. Once it has been shown that there exists a local minimum of the error function it therefore follows that this is a unique global minimum. To show the existence of a local minimum of the error function it is sufficient to prove that there exists a local minimum along each direction in the parameter space. This together with the fact that the set of directions is compact and taking the minimum of the error function along each direction is a continuous function will show that there exists a global minimum of the error function. To show that there exists a local minimum along each direction it is necessary to observe that the error function restricted to a particular direction in parameter space becomes a one-dimensional strictly convex function. It is therefore sufficient to show that the error function tends to infinity if the one-dimensional parameter goes to infinity or minus infinity. For a K-class problem of arbitrary finite dimension this can be shown as follows. Let $q_j(x)$, $j = 1, \ldots, K-1$, be an arbitrary set of polynomials such that not all of them are zero, then the direction specified by the $q_j(x)$ is the set of polynomials given by $cq_j(x)$, $j = 1, \ldots, K-1$, where c is a real number. Suppose there exists an x_n, such that $q_j(x_n) \neq 0$ for at least one j. Without restricting the generality of the proof one can assume that $q_j(x_n) > 0$. Now let J be defined by

$$J = \arg\max_j q_j(x_n) \tag{49}$$

Then $q_J(x_n) > 0$ and therefore the following holds true.

$$\frac{e^{cq_J(x_n)}}{1 + \sum_{j=2}^{K} e^{cq_j(x_n)}} \to 1 \quad \text{for} \quad c \to \infty \tag{50}$$

This implies that $S_j(x_n) \to 0$ if $j \neq J$. Similarly, one has

$$\frac{e^{cq_J(x_n)}}{1 + \sum_{j=2}^{K} e^{cq_j(x_n)}} \to 0 \quad \text{for} \quad c \to -\infty \tag{51}$$

This implies that for $|c| \to \infty$ there is always one j such that $S_j(x_n)$ converges to zero. For such a j the following therefore holds if $p_{n,j} \neq 0$.

$$p_{n,j} \log\left(\frac{p_{n,j}}{S_j(x_n)}\right) \to \infty \tag{52}$$

And as a result the error function $E(x,p,w,\mathbf{a})$ tends to infinity. Together with Theorem 1 the following theorem has therefore been proved

Theorem 2 *Let (x,p,w) be a multi-class problem of arbitrary finite dimension. Suppose none of the $p_{n,j}$ are zero and that there exists for each set of functions $q_j(x)$, $j = 1, \ldots, K-1$, and each x_n at least one j for which $q_j(x_n) \neq 0$ then there exists a point $\mathbf{a}$ in the parameter space which minimises $E(x,p,w,\mathbf{a})$. If, furthermore, there exists at least one j for which there are $L_j + 1$ data points x_n such that the $\Phi_j(x_n)$ are linearly independent then this solution is unique.*

Since a one-dimensional polynomial $q_j(x)$ of degree L_j has only L_j roots, together with Corollary 1, Theorem 2 immediately implies the following

Corollary 2 *Let (x, p, w) be a one-dimensional K-class problem and suppose there exist at least $\min_j L_j + 1$ different x_n such that $p_{n,j} \neq 0$ for all the j. Then there exists exactly one point $\mathbf{a}$ in the parameter space which minimises $E(x, p, w, \mathbf{a})$.*

The requirement that $p_{n,j} \neq 0$ for all j is essential in the above statements. As will be shown in the next section, one cannot expect to find a global minimum of the error function if there are $p_{n,j}$ which are zero. In this case, however, experimental evidence suggests that there still exists an approximate solution that is unique to within multiplication by a constant.

4 Practical parameter reestimation

This section will discuss some of the problems that are involved in determining an optimal set of parameters in practice. Although Theorem 2 guarantees the existence of a unique solution to the optimisation problem for a very general class of functions $q_j(x)$ the functions that will be considered in this section are exclusively polynomials.

The proof of Theorem 2 in the last section was not constructive. Initially, it is therefore not clear how to find the global minimum that is guaranteed by this theorem. There are a number of standard optimisation schemes, like conjugated gradients and gradient descent that could be used for this task. The method investigated here is the Newton algorithm with line search and backtracking as described in [9]. This algorithm will be used to find a solution of the equation

$$\nabla E(x, p, w, \mathbf{a}) = 0 \tag{53}$$

which characterises the global minimum of the error function $E(x, p, w, \mathbf{a})$. Now one step of the Newton iteration with line-search is given by

$$\mathbf{a_{i+1}} = \mathbf{a_i} - \lambda(\mathbf{a_i}) H(x, p, w, \mathbf{a})^{-1} \nabla(E(w, p, x, \mathbf{a})) \tag{54}$$

where $\lambda(\mathbf{a_i})$ is a positive variable that determines how far one moves in the Newton direction $H(x, p, w, \mathbf{a})^{-1} \nabla E(w, p, x, \mathbf{a})$, and, as before, $H(x, p, w, \mathbf{a})$ is the second order derivative of the error function. The variable λ is required to decrease the length of the gradient in going from $\mathbf{a_i}$ to $\mathbf{a_{i+1}}$ and will therefore quite often be lower than 1. On the other hand, to avoid slow or even spurious convergence, λ is not allowed to become arbitrarily small. In the Newton algorithm, it is essential that $H(x, p, w, \mathbf{a})$ is well conditioned to allow for an effective matrix inversion. This might not be the case if the x_n have high values and the degrees of the exponential polynomials are large. In this situation the absolute values of the entries in different rows of H might vary by some orders of magnitude. This undesirable effect can be avoided by rescaling the data points x_n. In principle, one can apply an arbitrary affine linear transformation A to the x_n and use the Newton iteration for this transformed problem. This results in a set of polynomials q_j from which the polynomials of the original problem can be derived by substituting $q_j(x)$ by $q_j(Ax)$. Since A was assumed to be affine linear the function $q_j(Ax)$ is again a polynomial. Thanks to this observation the Newton algorithm can be efficiently applied to most practical situations.

In the following, the reestimation of the softmax parameters will be illustrated with the help of several examples. First, a number of one-dimensional examples will be discussed that are intended to show basic properties of the softmax functions. Subsequently, the reestimation procedure will be applied to two-dimensional classification tasks with a special focus on the comparison of softmax functions and neural networks. The data points x_n in all of these experiments were chosen to lie on an equidistant grid. This is, however, not necessary for the reestimation to work. Such data points were only chosen to investigate the basic ability of softmax functions to approximate an arbitrary posterior. The $p_{n,j}$ in these examples were either 0 or 1. This corresponds to the usual situation where the training data x_n are labelled according to their class membership. The examples will show that the requirement in Theorem 2, that the $p_{n,j}$ are not zero, was essential because the sequence of $\mathbf{a_i}$ does not converge in this case. However, the value of the error does converge to zero and therefore the Newton algorithm is still applicable in this situation.

4.1 The one-dimensional two-class problem

In the following experiments, the training data were generated by prescribing two posterior probabilities to a set of 2000 equidistant points over an interval ranging from -10 to 10. One of the posteriors was chosen to be constant and equal to 1 in the interval from -5 to 5 and was set to zero elsewhere. The other was determined by the requirement that both posteriors sum to one at each data point. Each of the following figures shows only the first posterior probability.

Figure 1 shows approximations of a true posterior (solid line) by softmax functions that were derived from an initial guess by a sequence of Newton iterations of the form (54). As can be seen from figure 1, the approximations become increasingly accurate with an increasing number of iterations. The initial guess in this experiment was the polynomial

$$p_{init}(x) = x \tag{55}$$

which gives the softmax function in the upper graph of figure 1 where it is represented by the dotted curve. Since the initial polynomial was chosen at random the corresponding softmax function is quite far from the posterior it is meant to approximate. Although the initial polynomial had degree one, the reestimation procedure was required to produce a polynomial of degree two. The upper graph in figure 1 shows that already after 5 iterations the estimated posterior had changed from an asymmetric to a symmetric shape. Until about iteration 15 this estimate changed only little which was also reflected by a small value of $\lambda(\mathbf{a})$ in (54) during these iterations. Only after iteration 15 did the reestimation procedure start to take larger steps in the Newton direction again and after 30 iterations there was hardly any difference between the true and the estimated posterior. This was also reflected by the value of the error function which dropped from 5164.49 for the initial guess to 14.84 for the approximating softmax function after 30 iterations. The estimated polynomial after 30 iterations was

$$p_{30}(x) = 2.21442x^2 + 0.02222x - 55.35773 \tag{56}$$

Although the sequence of softmax functions converges, Theorem 2 cannot be applied in this case because all the posterior probabilities $p_{n,j}$ are either 0 or 1. It is therefore not immediately clear if there exists a unique solution in the parameter space. And indeed, further iterations show that the polynomial parameters do not converge. After 60 Newton iterations the estimated polynomial was

$$p_{60}(x) = 502.12755x^2 + 5.02127x - 12553.18762 \tag{57}$$

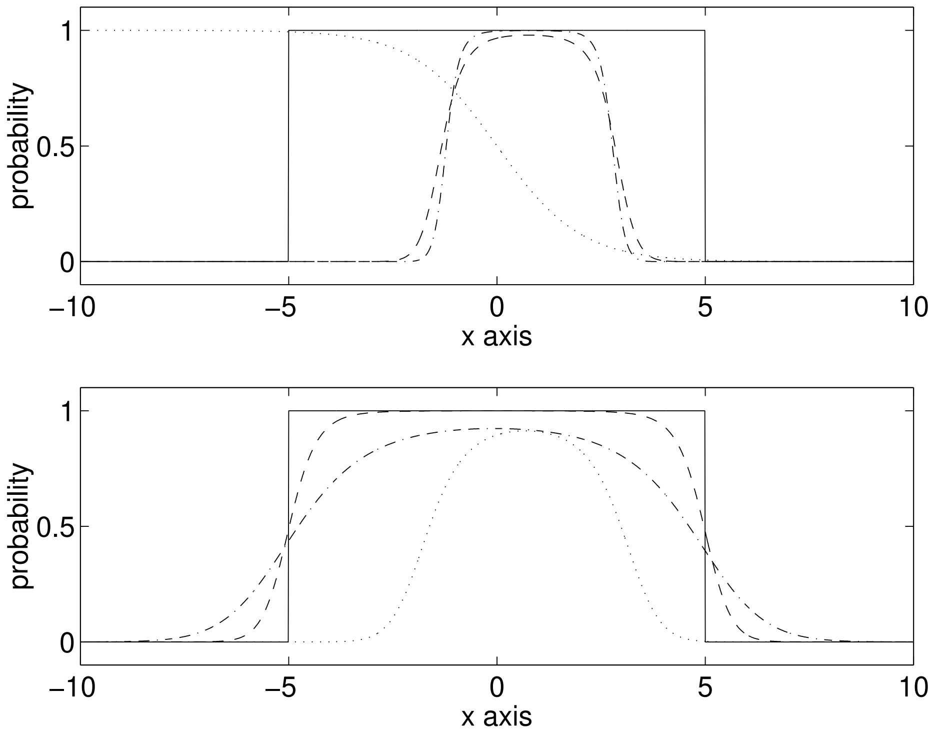

Figure 1: Sequence of approximations to posterior probability. In both graphs the solid line is the true posterior. In the upper graph the dotted line is the initial posterior, the dash-dotted line and the dashed line are the approximations after 5 respectively 10 Newton iterations. In the lower graph the dotted, dash-dotted and dashed line are the approximations after 15, 20 and 25 Newton iterations, respectively.

This is roughly $226 \cdot p_{30}$ and therefore the second 30 iterations amount to a much larger change in the polynomial parameters than the first 30 iterations. However, the change in the value of the error function is relatively small. After 60 iterations this value was $4.99 \cdot 10^{-11}$. Even though this is a very small value it could still have been further decreased by multiplying p_{60} with a high positive number. The reason for this behaviour is the result of the relationship between the softmax function and the exponential polynomial in this example which is illustrated by figure 2. Here the dashed line shows the estimated polynomial after 30 iterations and the solid line gives the true posterior. The polynomial has been scaled to fit to the axes of the plot. The true maximum of the polynomial on this interval was 166.3 and was attained at the boundaries. As can be seen, the polynomial vanishes at the discontinuities of the input posterior where the average is 0.5. This is the main requirement for the polynomial of the approximating softmax function, because the set of points where $q(x)$ vanishes is the set of points where the two softmax functions are 0.5 and has therefore to coincide with the decision boundary of the classification problem. Once this relationship between the points x where $q(x) = 0$ and the decision boundary of the classification problem has been established, every multiple $cq(x)$ of the polynomial, where $c > 0$, will also satisfy this requirement. Furthermore, since the values of the softmax function are always strictly higher than 0 and strictly lower than 1 every multiple $cq(x)$, with $c > 0$, will result in an approximation of the true posterior which becomes better with increasing c. This shows that in the current example the solution to the classification problem is the point at infinity of the line $cq_{30}(x)$ in the parameter space.

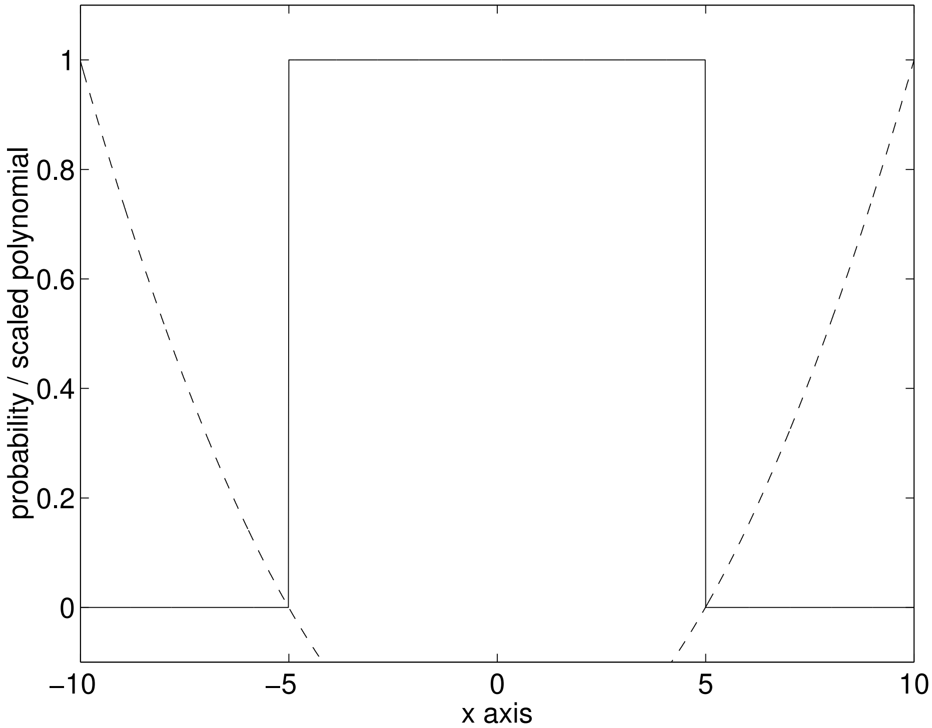

Figure 2: Input posterior probability (solid line) and scaled estimated polynomial of softmax function (dashed line) after 30 iterations.

It was mentioned before that the reestimation procedure in the current example was required to produce a polynomial of degree two. Figure 3 shows that the choice of the degree is important. The upper graph in figure 3 shows the result of the reestimation procedure when the polynomial was restricted to have degree one. After 30 iterations the reestimation procedure had produced the following result

$$p_{30}(x) = 2.99550 \cdot 10^{-4}x + 9.99501 \cdot 10^{-4} \tag{58}$$

This polynomial is almost identical zero. The least biased approximation in this case is therefore a posterior probability that is 0.5 for every value of x. The opposite problem occurs when the degree of the polynomial is too high. If the reestimation procedure is required to produce a polynomial of degree three one arrives after 30 iterations at the following polynomial

$$p_{30}(x) = -3.68155x^3 + 252.71314x^2 + 94.56630x - 6318.74809 \tag{59}$$

As can be seen from the second graph in figure 3, the corresponding softmax function models the input posterior perfectly over the interval from -10 to 10, i.e. it is almost exactly one between -5 and 5 and zero elsewhere, but the approximation produces an artifact at the value $x = 68.65$. Although subsequent iterations move the artifact further away from the

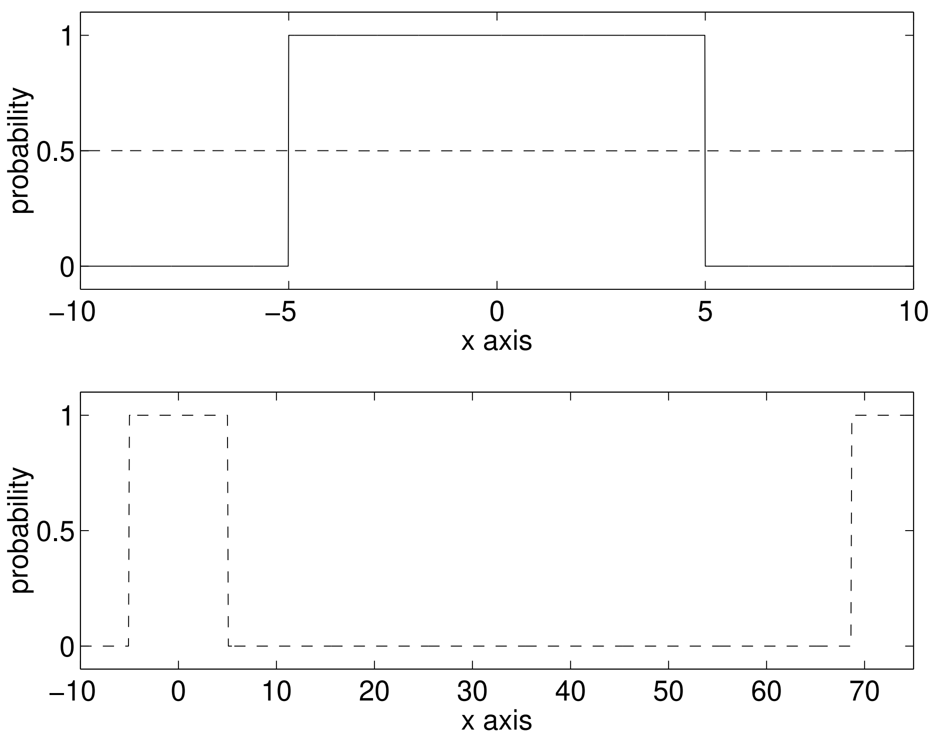

Figure 3: In the upper graph the polynomial in the softmax function was required to have degree not greater than 1. In the lower graph the polynomial was allowed to have degree up to 3.

interval[1] it can never be removed completely. The reason for this behaviour lies in the fact that, as mentioned before, the reestimation forces the polynomial to be zero at the decision boundary of the classification problem. This means that, in the current example, the third order polynomial of the approximating softmax function has to have two real roots and therefore the third root has to be real as well, since all the polynomial coefficients are real. The third root, however, is not associated with any of the discontinuities of the input posterior. This example shows that the correct degree of the polynomial in a one-dimensional two class problem can be found by counting the number of points in the decision boundary of the classification problem. At these points the polynomial $q(x)$ has to vanish. The correct degree of the polynomial is therefore equal to the number of points in the decision boundary. In the current example, this degree can also be characterised as the smallest degree of a polynomial such that the reestimation procedure produces a series of softmax functions whose error function values converge to zero.

In the previous example the weights w_n of the error function $E(x, p, w, \mathbf{a})$ were assumed to be constant. Figures 4 and 5 show the effect of introducing weights w_n which are samples of Gaussian densities. In both figures the posterior probability, which is represented by the solid line, was approximated by a softmax function with a polynomial of degree two. Since, according to the considerations above, the degree of the polynomial has to be 4 for a

[1]after 40 iterations it is located at $x = 122.98$, after 50 at $x = 177.30$

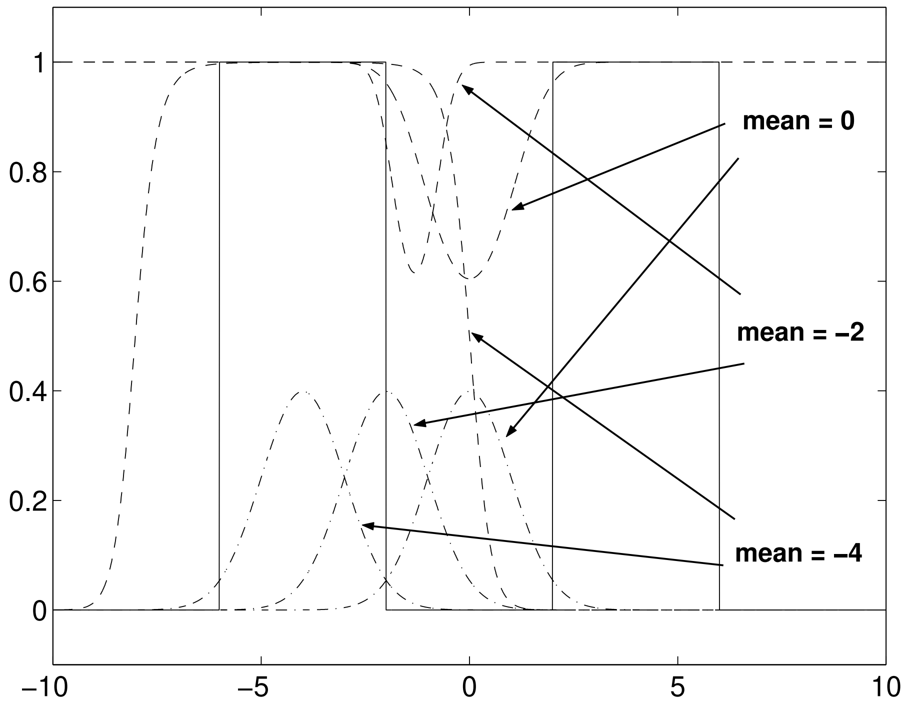

Figure 4: Approximation of posterior with softmax functions with second degree polynomial and non-uniform weights w_n. The weights in this graph are represented by the dash-dotted lines. These are Gaussians with unit variance and mean at $x = -4$, $x = -2$ and $x = 0$ respectively. The approximations are the dashed curves.

perfect approximation, the approximating softmax functions in the following examples have to be suboptimal. In figure 4 one can see that shifting the Gaussian density along the x-axis changes the area of good approximation. Here the densities are represented by the dash-dotted lines. For the Gaussian with mean $x = -4$, for instance, the approximating softmax function resembles the left bump of the posterior distribution. For the density functions with means $x = -2$ and $x = 0$ the two bumps are not modelled that well anymore but the approximation of the zero region between the two bumps becomes better. At first, these approximations might seem worse than one would expect. One would probably hope that the resulting softmax function models the first bump of the posterior almost perfectly for the density with mean $x = -4$. Instead the softmax function appears to be too wide. Similarly, one might expect the softmax function to model the zero region in the middle much better for the density with mean $x = 0$. The fact that these expectations are not met is not the result of a failure of the reestimation procedure to converge. On the contrary, the error function in all these examples was almost zero. The reason for this seemingly bad approximation behaviour is the relatively high variance of the density functions. As can be seen from figure 5, the approximation becomes better with decreasing variance, although the area in which the approximation has to be good becomes smaller and smaller.

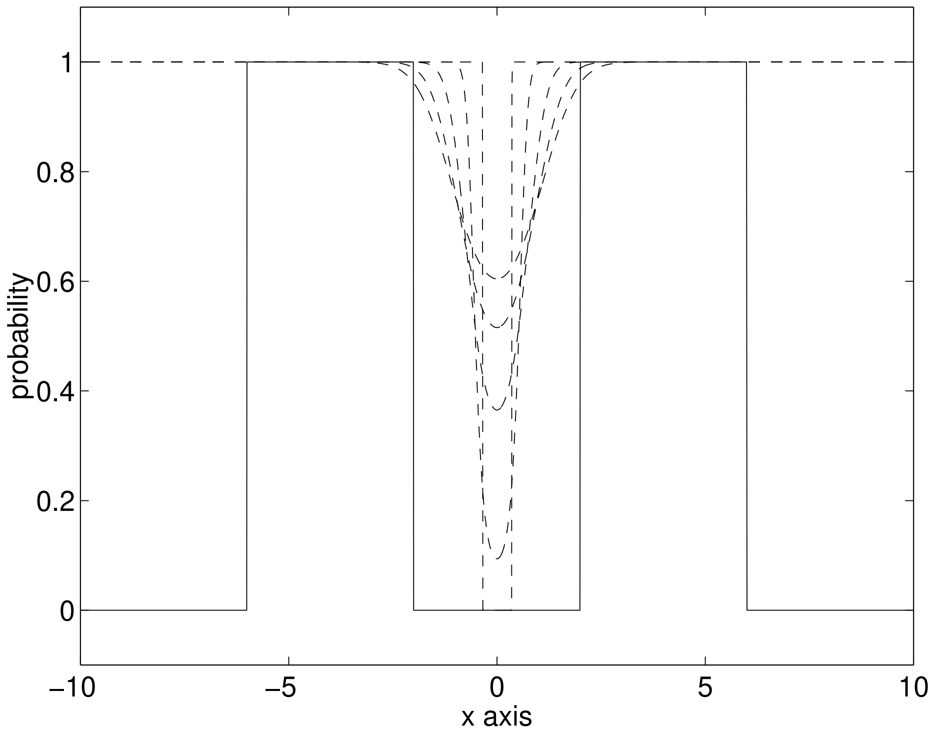

Figure 5: Approximation for weights that are distributed according to Gaussian densities with fixed mean and variable variance.

4.2 Higher dimensional 2 class problems

The following examples will discuss the softmax parameter reestimation for two-dimensional classification problems. Again the exponential functions are polynomials. In the context of a two-dimensional problem this means that the polynomials are given by

$$q(x_1, x_2) = a_{00} + a_{10}x_1 + a_{01}x_2 + a_{20}x_1^2 + a_{11}x_1x_2 + a_{02}x_2^2 + \cdots \tag{60}$$

The degree of such a polynomial is defined to be the smallest number L such that $a_{i_1 i_2} = 0$ for $i_1 + i_2 > L$. Equation (60) shows that the parameters of a two-dimensional polynomial can be arranged in the form of a triangular matrix as follows

$$\begin{pmatrix} a_{00} & a_{10} & a_{20} & \cdots\cdots & & \cdots & a_{d0} \\ a_{01} & a_{11} & a_{21} & \cdots & & a_{(d-1)1} & 0 \\ a_{02} & a_{12} & a_{22} & \cdots & a_{(d-1)2} & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{0d} & 0 & 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix} \tag{61}$$

Here the degree of the polynomial was assumed to be L. A two-dimensional polynomial of degree L therefore has $(L+1)(L+2)/2$ parameters. The determination of the proper degree of the polynomial is not as straight-forward for a higher dimensional classification problem as it is in one dimension. This is the case because in higher dimensions the decision boundary of the classification problem is not given by a set of discrete points. Usually, in m dimensions the decision boundary is given by an $m-1$ manifold. Such manifolds can be described locally by the following equation

$$f(x) = 0 \tag{62}$$

where $f(x)$ is a scalar valued function. Here, the term "locally" means that for each point x in the observation space there exists a neighbourhood U such that the decision boundary in this neighbourhood is given by equation (62). Such manifolds are usually categorised according to the properties of the functions $f(x)$ in (62). If $f(x)$ can always be chosen to be analytic, for instance, the manifold is called analytic. Analyticity means that at each point in U the function $f(x)$ can be expanded into an exponential series. Consequently, analytical manifolds can be locally approximated by polynomial manifolds. This means that the set of points x that satisfy equation (62) can be approximated by the set of points x that satisfy

$$q(x) = 0 \tag{63}$$

where $q(x)$ is a polynomial. Since equation (63) also defines the decision boundary for a two-class problem with polynomial softmax functions, one might expect that softmax functions solve at least locally every classification problem with a reasonably well behaved decision boundary. In order to obtain a good approximation to the decision boundary one has to make sure that the degree of the polynomial $q(x)$ is sufficiently high. This requirement does not place an upper bound on the number of parameters necessary to achieve good approximation. However, the next example will show that for a two-dimensional two-class problem, that is normally used to motivate the introduction of neural networks with two hidden layers, softmax functions can be found that solve this problem with a similar number of parameters as an appropriate neural network.

The left upper graph in figure 10 illustrates a typical classification problem that necessitates the use of neural networks with more than one hidden layer. Here the plane is divided into three regions, the innermost and the outermost belong to class 2 the one in the middle belongs to class 1. The boundaries between each of these regions are defined by 4 non-rectangular straight lines. The neural network in figure 6 could therefore be used to learn this classification problem. Here I_1 and I_2 are the input units for the components of the two-dimensional input vector and $I_0 = 1$ is the unit corresponding to the bias. Since in total there are 8 different straight lines to learn, the first hidden layer has 8 hidden units plus a bias unit $h_{1,0} = 1$. Each of the 8 hidden units $h_{1,1} - h_{1,8}$ corresponds to the question as to whether the inner product of the input vector with the normal direction to one of the lines is positive or negative. It will be assumed here that the units $h_{1,1} - h_{1,4}$ in the first hidden layer correspond to the lines that define the outer rectangle in the first plot of figure 10 and that the units $h_{1,5} - h_{1,8}$ correspond to the lines defining the inner rectangle. The first node in the second layer $h_{2,1}$ can therefore realise the question if the input vector lies inside or outside the outer rectangle and the second hidden unit $h_{2,2}$ can realise the question if the input vector lies inside or outside the inner rectangle. The answers to these questions are finally combined with a bias unit $h_{2,0} = 1$ to infer if the input vector belongs to class one which is represented by a value of one in the output unit O_1 or to class two which is represented by a value of one in the output unit O_2. The activation units in this example can be chosen to be Heaviside functions. The total number of weights in this neural network is given by

$$8 \cdot 3 + 2 \cdot 5 + 2 \cdot 3 = 40 \tag{64}$$

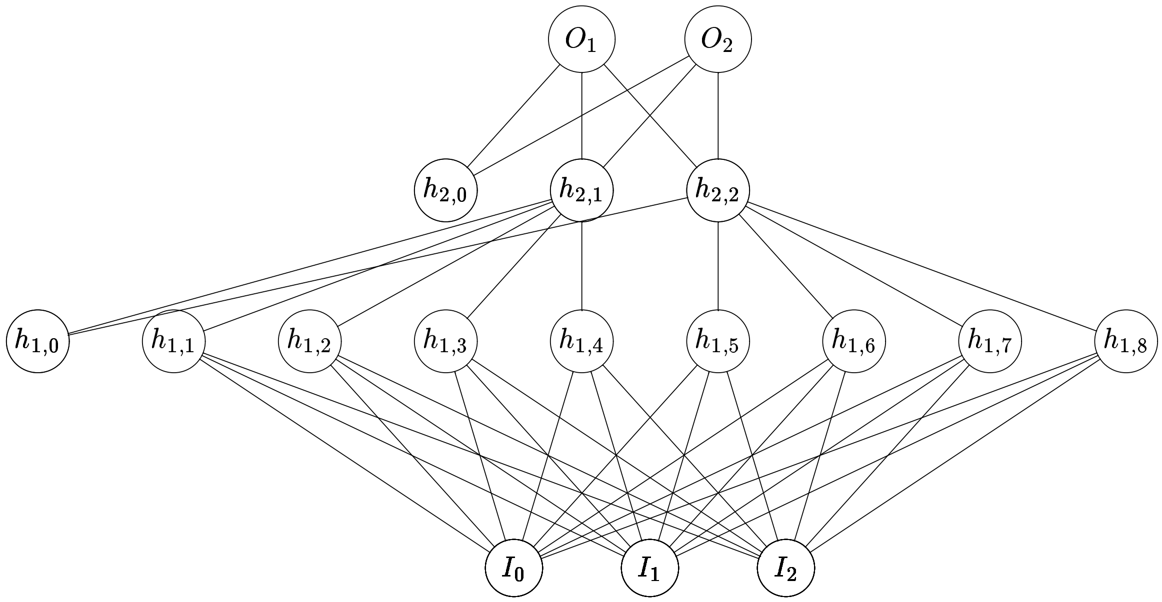

Figure 6: Neural net for classification task in figure 10

Apart from the bias, only the units $h_{1,1} - h_{1,4}$ in the first layer are connected to node $h_{2,1}$ in the second hidden layer, and only the units $h_{1,5} - h_{1,8}$ are connected to node $h_{2,2}$. The topology of this network therefore makes use of the fact that the current classification problem can be solved by answering the question as to whether the input vector lies inside or outside of two structures in the plane. Without making use of this a priori information all the nodes in the first hidden layer have to be connected to all the units in the second hidden layer. The total number of parameters in such a general neural network is therefore given by

$$8 \cdot 3 + 2 \cdot 9 + 2 \cdot 3 = 48 \tag{65}$$

Both these numbers correspond roughly to the number of parameters in a two-dimensional polynomial of degree 8 which is given by

$$\frac{(8+1)(8+2)}{2} = 45 \tag{66}$$

In the following experiments the posteriors for the two classes were sampled on an equidistant grid of $51 \cdot 51 = 2601$ points. As in the one-dimensional example discussed previously the posterior probabilities $p_{j,n}$ were either 0 or 1. This again rules out the application of Theorem 2 and in principal every approximate solution of the classification problem can be multiplied by a positive constant to find a better approximation. Here, however, the Newton iterations were applied as if nothing was known about the posteriors $p_{j,n}$ and the reestimation was considered to be completed if the length of the gradient of the error function was virtually zero. The initial polynomial in all the experiments was randomly chosen to be

$$q(x_1, x_2) = x_1 + x_2 \tag{67}$$

This choice is very far away from the final solution. In fact, the corresponding softmax functions have a decision boundary that is the straight line through $(0,0)$ with slope -1. Despite this poor initial choice, figure 10 shows that after a number of iterations the true posterior will be approximated with arbitrary precision if the degree of the exponential polynomial is high enough. Figure 10 shows the contour plots of the softmax approximations for polynomials having degree two up to eight. As can be seen, below degree four the approximation of the true posterior is relatively poor. This is because the inner rectangle cannot be modelled at all by a polynomial of such small degree. The approximations with polynomials of degree four and higher show an increasing sharpness of the edges and better approximation of the inner rectangle. For a polynomial of degree eight the decision boundary is modelled perfectly. The closed regions inside class 1 which start to appear from the 6th degree polynomial onwards between the outer and inner rectangle are the regions where the estimated softmax function is 1 to within machine precision. Up to a polynomial of degree 5 the reestimation procedure needed 50 iterations to converge. For the polynomials of degree 6, 7 and 8 the reestimation procedure was completed after 130, 200 and 350 iterations respectively. The parameters that

	0	1	2	3	4	5	6	7	8
0	286.6	9.4	-1088.9	369.9	857.7	-471.8	-331.7	156.1	78.6
1	-333.1	-301.6	854.9	2453.1	65.8	-1768.5	-312.4	260.0	*
2	-1130.7	-1312.3	-1807.6	3351.7	-26.2	-1214.1	778.7	*	*
3	-261.9	-3514.8	519.3	924.9	-570.7	949.1	*	*	*
4	-179.1	-806.3	1742.5	-463.1	-809.6	*	*	*	*
5	-251.8	1353.4	17.8	-1090.5	*	*	*	*	*
6	82.8	583.7	-7.3	*	*	*	*	*	*
7	139.0	112.5	*	*	*	*	*	*	*
8	60.9	*	*	*	*	*	*	*	*

Table 1: Parameters of 8th degree polynomial after 350 iterations.

were estimated for the polynomial of degree 8 are given in Table 1. The entries in this table represent the a_{ij} parameters of the polynomial where i is the row index and j is the column index of the table. This table shows that although there is some variation in the magnitude of the polynomial parameters, the triangular matrix is far from sparse. The full parameter set of the 8th degree polynomial has therefore to be employed to solve the current classification problem. The good convergence in all the experiments shows again the great insensitivity of the reestimation procedure to the initial choice of the exponential polynomial. This illustrates once more that there is only one global minimum which is the point at infinity of a line in the parameter space. Figure 10 shows that only for a polynomial of degree 8 the approximation to the true posterior becomes perfect. From a classification point of view, however, one might already be satisfied with a softmax function with a relatively small degree. Table 2 shows the percentages of data points in the training data correctly classified for different degrees of the exponential polynomial. The biggest increase in classification accuracy occurs when the degree of the polynomial is increased from 3 to 4. As mentioned before, this is a result of the fact that the inner rectangle is not modelled at all for polynomials of degree 2 and 3. Since the increase in classification performance is relatively small in going from degree 4 to 8 one might, in practice, already by satisfied with a 4th degree polynomial model, which has only $(4+1)(4+2)/2 = 15$ parameters. Note that the multiplication of the exponential polynomial by a positive constant does not have an effect on the classification performance of the softmax functions, because multiplication of the exponential polynomial does not change the decision boundary. Finally, figure 11 shows the contour plot of the 8th degree polynomial given by

2	3	4	5	6	7	8
89.2	90.6	96.6	97.8	98.9	99.5	100

Table 2: Percentage of training data points correctly classified. The first row gives the degree of the corresponding exponential polynomial.

Table 1. As can be seen from this figure, in the area under consideration the polynomial has 5 local extrema, 4 of which are local minima and one of them is a local maximum. Outside this region the polynomial tends to infinity. Each local maximum corresponds to a value of the softmax function which is close to 1 and each local minimum corresponds to a value close to 0. Furthermore, figure 11 shows the decision boundary that divides the inner rectangle from class 1. This is the curve in the middle of the graph with the pointed right upper corner and corresponds to the contour line of the polynomial for a value of 0.

4.3 Higher dimensional multi class problems

This section will discuss some two-dimensional K-class problems where $K > 2$ and will show that the Newton algorithm from the last section can also be applied in this situation. This leads to a reestimation algorithm that is very stable and can approximate locally every analytic decision boundary with arbitrary precision as long as the degrees of the exponential polynomials are high enough. Figure 7 shows a typical two-dimensional three-class problem. The first graph in this figure shows the true decision boundaries between the 3 different classes. Here the classes are denoted by $C1$, $C2$ and $C3$. The second and third graph in figure 7 show the approximations of these decision boundaries by 3 softmax functions with 2 polynomials of degree 1 and 2 respectively after 50 iterations. The initial polynomials were both chosen to be equal to the polynomial in (67). The curves in the second and third graph in figure 7 are the contour lines of the corresponding softmax functions at value 0.5. This explains why in the right upper graph there is a region in the middle of the plane where these lines do not intersect. This is the case, because all the decision boundaries intersect in this region and consequently the values of the approximating softmax functions are close to 1/3. As can be seen from figure 7, softmax functions with polynomials of degree 1 are only capable of learning one of the three decision boundaries. In general, they can learn any 3 class problem that is defined by only two independent decision boundaries. The current example is therefore too complicated for this type of softmax functions. However, already for softmax functions with polynomial exponents of degree 2 the approximation of the decision boundaries is almost perfect. This shows again that polynomials can be found that have a similar number of parameters as a neural network that is appropriate for this classification task. Such a neural network is given in figure 8. Here the weights of the first layer learn the location of the decision boundaries and the hidden layer learns where the classes are located with respect to the boundaries. The output units are 1 or 0 depending on whether the input pattern belongs to the corresponding class or not. This neural network has a total of $3 \cdot 3 + 3 \cdot 4 = 21$ parameters as compared to the softmax functions with two second degree polynomials which have $2 \cdot 3 \cdot 4/2 = 12$ parameters. Unfortunately, these softmax functions do not generalise in the obvious way, as can be seen from the last graph in figure 7. Here the region

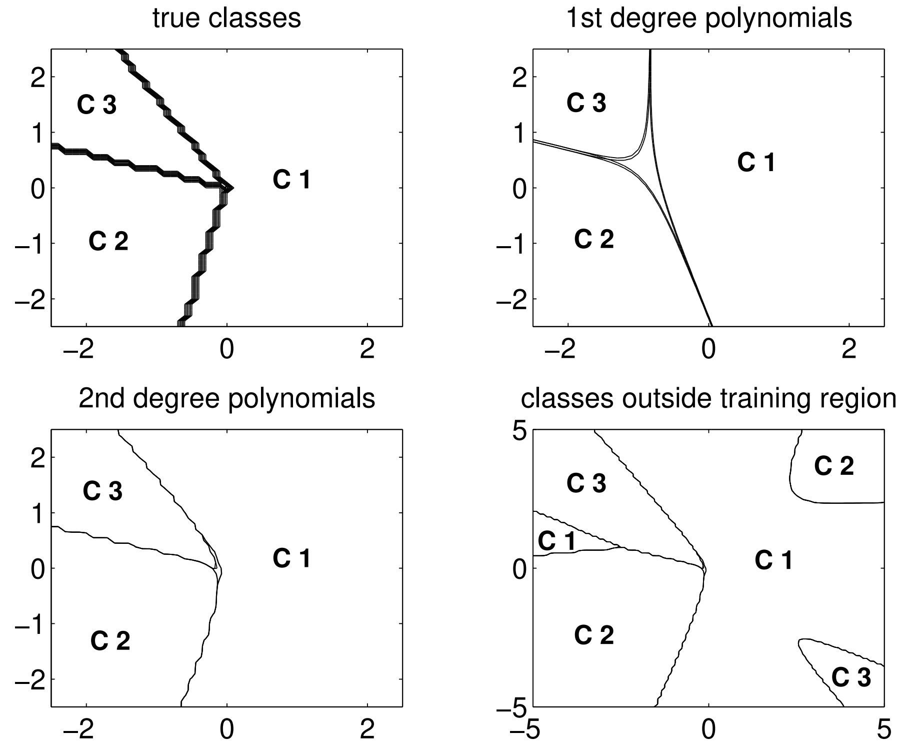

Figure 7: Two dimensional three-class problem.

where the training data were located was extended to the rectangle bounded by -5 and 5 in both directions. This shows that outside the training region the softmax functions introduce several artifacts. This is due to the fact that the reestimation procedure chooses polynomials whose contour lines are hyperbolic curves. This is also illustrated by the location of classes $C2$ and $C3$ in this graph which lie opposite each other and have a distinctive hyperbolic shape. It is not possible to remove these artifacts with polynomials of higher degree. In fact, for higher degree polynomials the artifacts become even more pronounced. Of course, one cannot expect that a model is capable of predicting data it has never seen. In principal what appear to be artifacts outside the training region are all valid extrapolations of the training data, and could be associated with the true posterior. However, it is somewhat contrived to assume such properties and it is therefore much more satisfactory to find a representation of the training data that does not make such assumptions. This problem of finding the simplest explanation for a set of observations is known as Occam's razor. In practice, this problem will be solved by finding the model with the smallest number of parameters. To achieve this task, in the current example, it is useful to consider the following posterior functions.

$$\begin{aligned} P_1(x) &= S_1^1(x)S_2^2(x) \\ P_2(x) &= S_1^2(x)S_2^3(x) \\ P_3(x) &= S_1^3(x)S_2^1(x) \end{aligned} \tag{68}$$

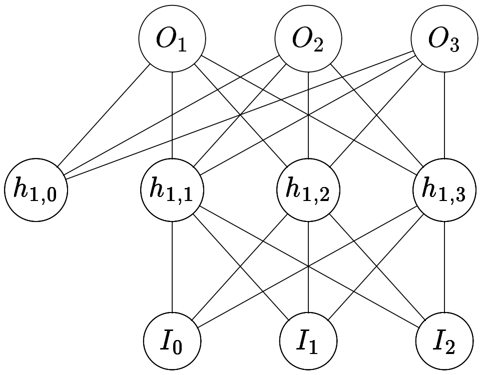

Figure 8: Neural network for a two-dimensional three-class problem.

Here the $S^i_{1,2}$ are softmax functions that are given by

$$S^i_1(x) = \frac{e^{q_i(x)}}{1+e^{q_i(x)}} \tag{69}$$

$$S^i_2(x) = \frac{1}{1+e^{q_i(x)}} \tag{70}$$

and the $q_i(x)$ are linear polynomials. It is sensible to assume that there exists a unique set of polynomials $q_i(x)$ such that the $P_i(x)$ in (68) solve the classification problem in figure 7, because each of the softmax functions $S^i_{1,2}(x)$ can model exactly one decision boundary of this problem and the products in the $P_i(x)$ can therefore model the intersections of the corresponding classes. A set of posteriors that are given by (68) which solves this classification problem will therefore have the minimal number of parameters and generalises in the desired way. It is interesting to note that the definition of the posteriors (68) contains a logical structure that is usually modelled by the hidden layers of a neural network. In this structure a logical "and" is realised by the product of two softmax functions and a logical "or" is realised by 1 minus the product of two softmax functions. As compared to neural networks, here, the logical structure is imposed a priori and is not learned automatically. It might, however, be possible to find the appropriate structure of the classification problem by applying a finite predefined set of structures and choosing the one that has the best trade-off between the number of parameters and a small value of the error function. Note that the functions in (68) are not posterior probabilities in the strict sense because they do not sum to one, i.e. the following holds

$$0 < P_1(x) + P_2(x) + P_3(x) < 1 \tag{71}$$

One can, nevertheless, use the error function $E(x, p, w, \mathbf{a})$ in this case whose individual terms are given by

$$\sum_{j=1}^{3} p_{n,j} \log\left(\frac{p_{n,j}}{C_n P_j^{norm}(x_n)}\right) \tag{72}$$

where

$$C_n = \sum_{j=1}^{3} P_j(x_n) \tag{73}$$

and P_j^{norm} is the normalised version of the posteriors

$$P_j^{norm}(x) = \frac{P_j(x)}{C_n} \tag{74}$$

Therefore (72) is the KL distance between $p_{n,j}$ and the $P_j^{norm}(x_n)$ minus $\log C_n$. Since (71) holds, $-\log C_n$ is always positive and therefore minimisation of $E(x,p,w,\mathbf{a})$ can only be achieved by modelling the posteriors $p_{n,j}$ as accurately as possible. If $-\log C_n$ could become negative, minimisation of $E(x,p,w,\mathbf{a})$ could be achieved by minimising $-\log C_n$ independently of the $p_{n,j}$. The components of the gradient of $E(x,p,w,\mathbf{a})$ are given by

$$\begin{aligned}
\frac{\partial}{\partial a_{1,l}} E(x,p,w,\mathbf{a}) &= \sum_{n=1}^{N} w_n (p_{n,3} S_1^1(x_n) - p_{n,1} S_2^1(x_n)) \frac{\partial}{\partial a_{1,l}} q_1(x_n) && (75) \\
\frac{\partial}{\partial a_{2,l}} E(x,p,w,\mathbf{a}) &= \sum_{n=1}^{N} w_n (p_{n,1} S_1^2(x_n) - p_{n,2} S_2^2(x_n)) \frac{\partial}{\partial a_{2,l}} q_2(x_n) && (76) \\
\frac{\partial}{\partial a_{3,l}} E(x,p,w,\mathbf{a}) &= \sum_{n=1}^{N} w_n (p_{n,2} S_1^3(x_n) - p_{n,3} S_2^3(x_n)) \frac{\partial}{\partial a_{3,l}} q_3(x_n) && (77)
\end{aligned}$$

As in section 3.1, in order to study the second order derivatives of the error function it is sufficient to study the matrices $\mathbf{h}_n = (h_{nij})_{i,j=1}^3$ whose diagonal elements are in this case given by

$$\begin{aligned}
h_{n11} &= w_n S_1^1(x_n) S_2^1(x_n)(p_{n,1} + p_{n,3}) && (78) \\
h_{n22} &= w_n S_1^2(x_n) S_2^2(x_n)(p_{n,2} + p_{n,1}) && (79) \\
h_{n33} &= w_n S_1^3(x_n) S_2^3(x_n)(p_{n,3} + p_{n,2}) && (80)
\end{aligned}$$

Since these numbers are all positive and the off-diagonal elements are all zero, this shows that the error function $E(x,p,w,\mathbf{a})$ is convex. In the light of this result, it seems promising to study more general “logical combinations” of softmax functions. It is interesting to note that the error function for the normalised posteriors $P_j^{norm}(x)$ does not have the same pleasant behaviour. Although the gradient has an intuitively understandable form, in this case, the matrix $\mathbf{h}_n$ is not necessarily positive definite as its diagonal elements can become negative if the estimated posteriors are far from the solution.

Finally, figure 9 shows that the reestimation procedure also works for classification problems with more than 3 classes and decision boundaries of various types. In this figure the left graph shows the decision boundaries of a classification problem with 6 different classes denoted by $C1-C6$. The right graph shows the contour plots of the approximating softmax functions after 60 iterations. The exponential polynomials in this example had degree 2 and the initial polynomials were all chosen to be equal to the polynomial in (67). This graph shows that the approximation is almost perfect. The contour lines that do not lie on the decision boundaries determine regions where the value of the softmax function was 1 to within machine precision. It is interesting to note, that in this example the reestimation procedure chose polynomials $q_j(x), j = 1, \ldots, 5$ whose contour lines are always ellipsoids. This is in contrast to the previous example where these lines were hyperbolic curves. The parameters of the 5 estimated polynomials for this 6 class problem are given in Table 3. The total number of parameters in the softmax functions was $5 \cdot 6 = 30$. Since these 30 parameters almost

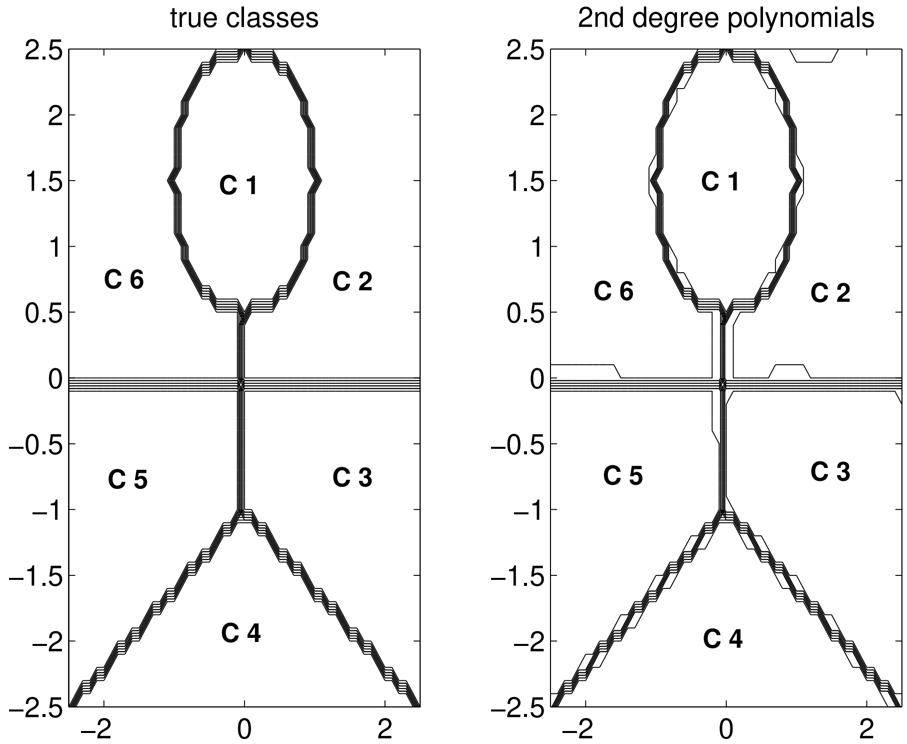

Figure 9: Contour plot of true and approximated posteriors of 2-dimensional 6-class problem.

perfectly represent the 2601 data points in this section of the plane, the softmax reestimation might be considered a feasible data compression scheme in this example. Table 3 also shows that relative to their absolute value the parameters $a_{j,0,1}$ and $a_{j,1,1}$ are the ones with the highest variation. Compared with these parameters the variation of the other parameters is relatively small. This points to the fact that the decision boundaries in the current example are not independent of one another. If all the parameters in the polynomials, apart from the $a_{j,0,1}$ and $a_{j,1,1}$, are tied together this results in a model with a total of 14 parameters. Compared to this, the number of parameters in a model where the lines and the circle are modelled with 3 parameters each is 15. This suggests that the polynomials $q_j(x)$ also contain information about the number of parameters in a minimal model.

5 Conclusions and further work

This report introduced softmax functions with polynomial and more general exponents and discussed their reestimation in the context of the cross-entropy error function. It was shown in section 3 that under relatively general assumptions on the training data, the error surface of a classification problem of arbitrary finite dimension and for a finite number of classes is always strictly convex and that there exists exactly one optimal set of parameters. Section 4 showed that a standard Newton algorithm with line-search and backtracking can be used to find the solution to the parameter reestimation problem. This algorithm was found to be very robust and insensitive to the choice of initial parameters. Even if the assumptions

polynomial 1				polynomial 2			
	0	1	2		0	1	2
0	9317.04	163.30	7331.32	0	9298.22	-210.60	7317.33
1	-22500.12	8.55	$*$	1	-22499.22	5.91	$*$
2	7498.01	$*$	$*$	2	7500.57	$*$	$*$
polynomial 3				polynomial 4			
	0	1	2		0	1	2
0	9209.01	-256.95	7308.93	0	9197.84	368.49	7214.73
1	-23665.20	670.49	$*$	1	-24087.57	-858.52	$*$
2	6442.65	$*$	$*$	2	6053.75	$*$	$*$
polynomial 5							
	0	1	2				
0	8819.48	192.25	7205.94				
1	-22819.85	122.16	$*$				
2	7670.21	$*$	$*$				

Table 3: Parameters of 2nd degree polynomials after 60 iterations.

that guarantee the existence of a unique solution were not satisfied this algorithm produced a sequence of posteriors that converged to the proper solution. Finally, in section 4.3 a more general way to define posterior probabilities as algebraic combinations of softmax functions was briefly introduced. Future work will investigate if such probability functions are a useful tool in implementing a "logical structure" with softmax functions. Furthermore, the experimental work presented here was mainly aimed at illustrating the basic features of the softmax reestimation theory and as such was restricted to toy examples. Future work will therefore also focus on more realistic problems.

References

[1] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proc. ICASSP*, pages 49–52, 1986.

[2] M. Bianchini, M. Gori, and M. Maggini. On the Problem of Local Minima in Recurrent Neural Networks. *IEEE Transaction on Neural Networks*, 5(2):167–177, 1994.

[3] C. Bishop. Exact Calculation of the Hessian Matrix for the Multilayer Perceptron. *Neural Computation*, 4:494–501, 1992.

[4] C. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

[5] J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs. In F. Fogelman-Soulie and J. Herault, editors, *Neurocomputing: Algorithms, Architectures and Applications*, pages 227–236. Springer-Verlag, Berlin, 1989.

[6] W. L. Buntine and A. S. Weigend. Computing Second Order Derivatives in Feed-Forward Networks: A Review. *IEEE Transactions on Neural Networks*, 5(3):480–488, 1994.

[7] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3:79–87, 1991.

[8] M. R. Lynch, P. J. Rayner, and S. B. Holden. Removal of degeneracy in adaptive Volterra networks by dynamic structuring. In *Proc. ICASSP*, pages 2069–2072, 1991.

[9] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1997.

[10] P. J. W. Rayner and M. R. Lynch. A new connectionist model based on a non-linear adaptive filter. In *Proc. ICASSP*, pages 1191–1194, 1989.

[11] P. J. W. Rayner and M. R. Lynch. Complexity Reduction in Volterra Connectionist Modelling by Consideration of Output Mapping. In *Proc. ICASSP*, pages 885–888, 1990.

[12] V. Valtchev, J. J. Odell, P. C. Woodland, and S. J. Young. MMIE training of large vocabulary recognition systems. *Speech Communication*, 22:303–314, 1997.

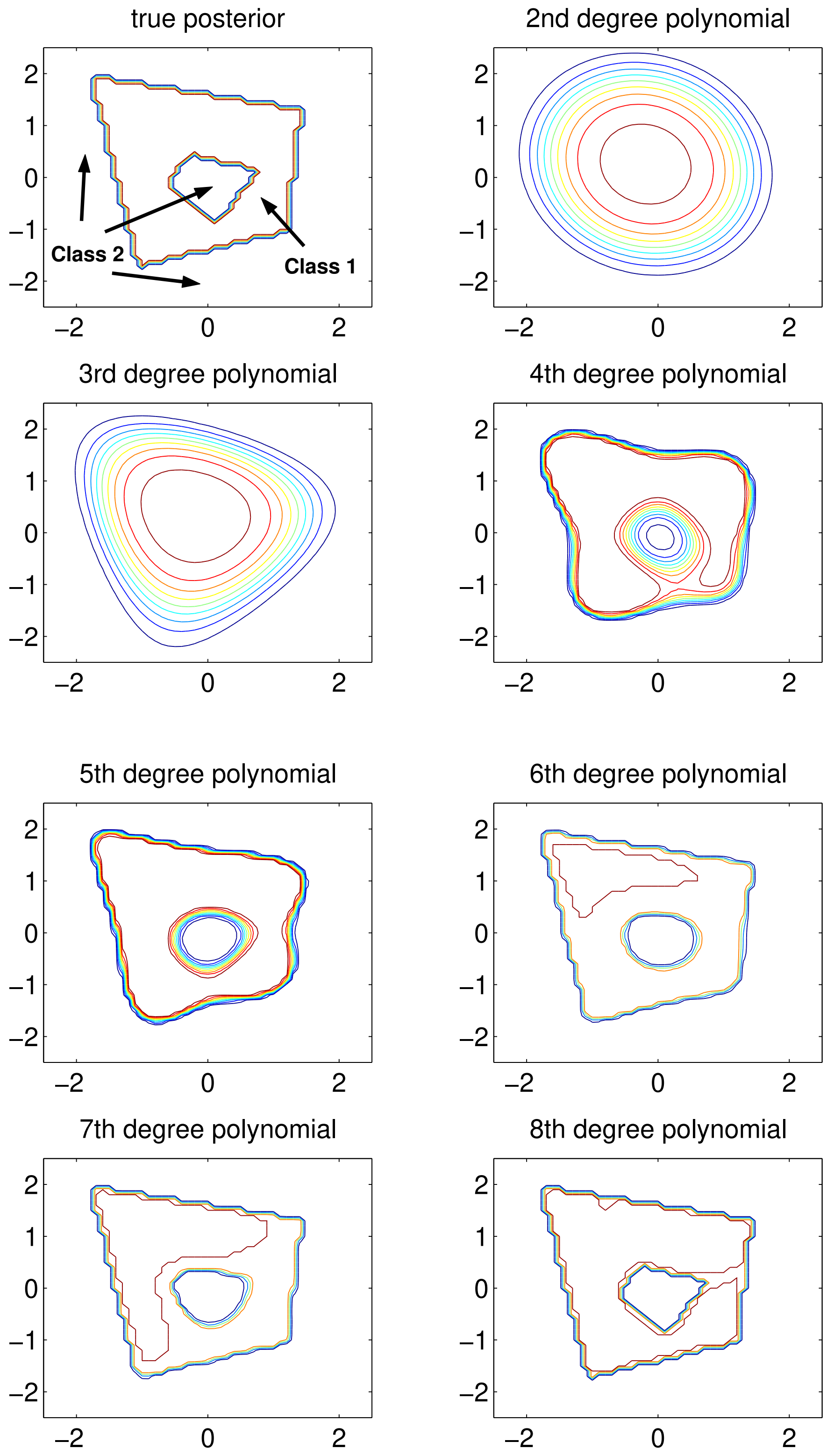

Figure 10: Contour plots of softmax approximations to true posterior of two-dimensional two-class problem. The degrees of the exponential polynomials range from 2 to 8.

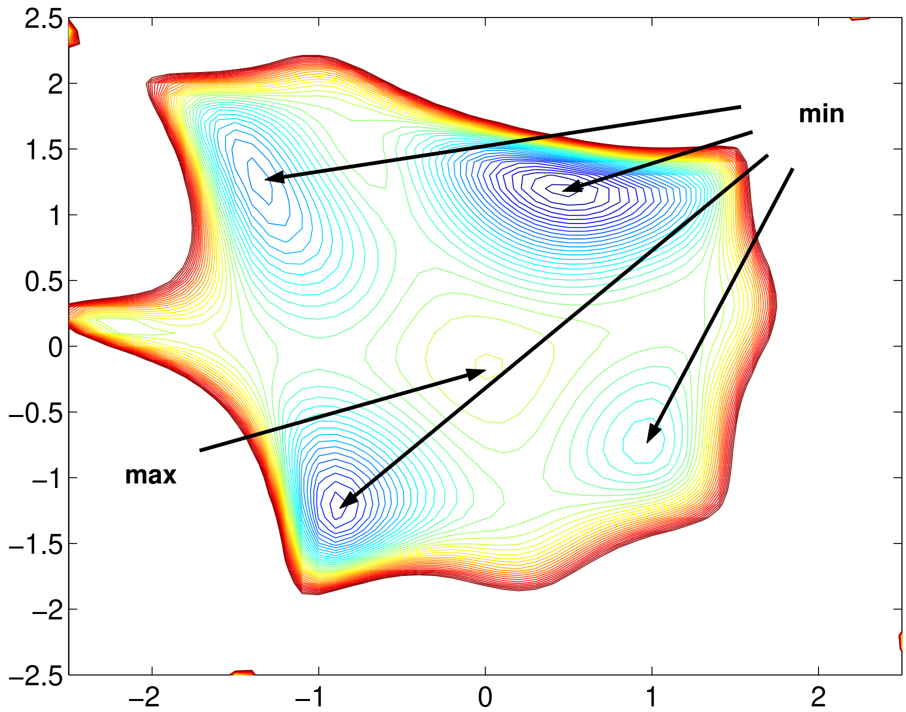

Figure 11: Estimated 8th degree polynomial