EFFICIENT TRAINING OF RBF NETWORKS FOR CLASSIFICATION

Ian T. Nabney

Neural Computing Research Group
Aston University, BIRMINGHAM, B4 7ET, UK

ABSTRACT

Radial Basis Function networks with linear outputs are often used in regression problems because they can be substantially faster to train than Multi-layer Perceptrons. For classification problems, the use of linear outputs is less appropriate as the outputs are not guaranteed to represent probabilities. In this paper we show how RBFs with logistic and softmax outputs can be trained efficiently using algorithms derived from Generalised Linear Models. This approach is compared with standard non-linear optimisation algorithms on a number of datasets.

1 INTRODUCTION

Radial Basis Function (RBF) networks with linear outputs are often used in regression problems because they can be substantially faster to train than Multi-layer Perceptrons (MLP). This is because it is possible to choose suitable parameters for the basis function parameters by an unsupervised technique (such as selecting a subset of the data for centres, using a clustering algorithm such as K-means, or training a mixture model with EM) so that the hidden unit activations model the unconditional input data density $p(\mathbf{x})$. With the hidden unit parameters fixed, and a sum of squared error function, the optimisation of the outputs weights is a quadratic problem that can be solved using the methods from numerical linear algebra.

In classification problems, rather than directly outputting a classification it is advantageous to estimate posterior probabilities $p(C_k|\mathbf{x})$, since this allows us to compensate for different prior probabilities, combine the outputs of several networks, make minimum risk classifications under different cost functions and to set rejection thresholds: see [3]. For classification problems, the use of linear outputs is less appropriate as then the network outputs are not guaranteed to represent probabilities. With MLPs it is common practice to use logistic (for two class) and softmax (for multiple classes) output nodes and appropriate cross-entropy error function so as to ensure that the outputs sum to one and all lie in the interval $[0,1]$. This does not add significantly to the time taken to train an MLP since even with linear outputs, general purpose optimisation routines must be used.

However, an RBF with logistic or softmax outputs no longer has a quadratic error surface for the output layer. If general purpose optimisation algorithms are used, much of the speed advantage over MLPs is lost. In this paper we show how RBFs with logistic and softmax outputs can be trained efficiently using algorithms derived from Generalised Linear Models. We compare these models with standard RBF's on both synthetic and real datasets.

2 TRAINING GENERALISED LINEAR MODELS

2.1 GENERALISING LINEAR REGRESSION

This brief outline of generalised linear models is based on that in [8]. In linear regression theory, it is assumed that the errors follow a normal distribution with constant variance σ^2. The output of the model represents the mean conditioned on the input vector $\mathbf{x}$:

$$\mu = \mathbf{x}\boldsymbol{\beta} \tag{1}$$

In a generalised linear model we replace the normal distribution for the target random variable Y by a distribution from the exponential family, which has the form:

$$P_Y(y, \eta, \phi) = \exp\{(\eta y - b(\eta))/a(\phi) + c(y, \phi)\} \tag{2}$$

where η is the 'natural parameter' and ϕ is the 'dispersion parameter'. For the normal distribution, $\theta = \mu$ and $\phi = \sigma^2$. The model now has the form

$$\eta = \mathbf{x}\boldsymbol{\beta} \tag{3}$$

The mean μ of P_Y is modelled as a function of η: $\mu = b'(\eta)$, where $f = b'$ is called the *link* function. If $\eta = \theta$, then the output of the generalised linear model is the natural parameter of the noise model, and f is the *canonical* link.

The normal distribution is not an appropriate error model for classification problems, where the output variable is discrete. For a two class problem, we use a Bernoulli distribution

$$P(y) = \pi^y(1-\pi)^{1-y} = \exp\{\eta y - \ln(1+e^\eta)\} \tag{4}$$

where π, the probability of 'success' is the mean, and $\eta = \ln(\pi/(1-\pi))$. This corresponds to using a logistic function at the output of the generalised linear model. For an m class problem, we use the multinomial distribution on m variables:

$$P(y_1, y_2, \dots, y_m) = \frac{M!}{(y_1!)(y_2!)\cdots(y_m!)} p_1^{y_1} p_2^{y_2} \cdots p_m^{y_m} \tag{5}$$

where p_i is the probability of the ith class and $M = \sum_{i=1}^{m} y_i$ is generally taken to be equal to one. The model has m outputs and the canonical link function is the familiar softmax function:

$$p_i = \frac{e^{\eta_i}}{\sum_{j=1}^{m} e^{\eta_j}} \tag{6}$$

In the statistical literature this is known as multiple logistic regression. The target data should have a 1-of-m encoding.

2.2 PARAMETER ESTIMATION

The obvious starting point for training these models is to use maximum likelihood. For linear regression, this is equivalent to minimising the quadratic form

$$(\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta) \tag{7}$$

with respect to β, where $\mathbf{X}$ is the data matrix and $\mathbf{Y}$ is the target matrix. Equating the derivative to zero yields the normal equations

$$(\mathbf{X}^T\mathbf{X})\beta = \mathbf{X}^T\mathbf{Y} \tag{8}$$

which can be solved efficiently by computing the pseudo-inverse $\mathbf{X}^\dagger$ of $\mathbf{X}$ and setting $\beta = \mathbf{X}^\dagger\mathbf{Y}$. This is numerically more stable than computing explicitly the inverse of the square matrix $\mathbf{X}^T\mathbf{X}$.

Maximum likelihood for both generalised linear models we are considering does not lead to a quadratic form, and so iterative methods are used instead. In principle there is no reason why general purpose nonlinear optimisation algorithms should not be used, but it is more efficient to take advantage of the special 'near-linear' form of the model. Let $\mathcal{L}$ denote the log likelihood and $\mathbf{H} = (\partial^2\mathcal{L}/\partial\beta\partial\beta^T)$ the Hessian of $\mathcal{L}$. The Fisher scoring method updates the parameter estimates β at the rth step by

$$\beta_{r+1} = \beta_r - \{E\,[\mathbf{H}]\}^{-1}\frac{\partial l}{\partial \beta} \tag{9}$$

This is the same as the Newton-Raphson algorithm, except that the expected value of the Hessian replaces the Hessian[1]. Normally taking a full Newton step is not a good idea, as it is easy to overshoot the maximum. However, there are two special features of the generalised linear model that make this procedure work well in practice: the log likelihood of logistic models has a single maximum, and it is possible to initialise the parameter β reasonably close to the maximum.

The Hessian of the logistic model is equal to $-\mathbf{X}^T\mathbf{W}\mathbf{X}$, where $\mathbf{W}$ is a diagonal *weight* matrix whose elements are $\pi^{(n)}(1 - \pi^{(n)})$. The gradient is equal to $\mathbf{X}^T\mathbf{W}\mathbf{e}$, where the nth row of $\mathbf{e}$ is given by

$$e^{(n)} = (y^{(n)} - \pi^{(n)})/f'(\eta^{(n)}) \tag{10}$$

We form the variable $\mathbf{z}_r = \mathbf{X}\beta_r + \mathbf{e}$, which is the linearisation of the link function around the current value of the mean. Then the equation (9) reduces to:

$$(\mathbf{X}^T\mathbf{W}_r\mathbf{X})\beta_{r+1} = \mathbf{X}^T\mathbf{W}_r\mathbf{z}_r \tag{11}$$

which is the normal form equation for a least squares problem with input matrix $\mathbf{X}^T\mathbf{W}_r^{1/2}$ and dependent variables $\mathbf{W}_r^{1/2}\mathbf{z}_r$. The weights change at each iteration, since they are a function of the parameters β_r. The algorithm is known as Iterated Re-weighted Least Squares (IRLS).

The reduction of the Newton step to the normal form equation (11) depends on being able to find a square root of $\mathbf{W}$ (which is easy in this case, as it is non-negative diagonal), and compute $\mathbf{X}^T\mathbf{W}^{1/2}$ efficiently (which can be done without a full matrix multiplication again as $\mathbf{W}$ is diagonal). We initialise the procedure by using the values $(y^{(n)} + 0.5)/2.0$ as a first estimate for $\pi^{(n)}$ and from this deriving the other quantities needed. The uniqueness of the maximum of $\mathcal{L}$ was shown in [1].

The case of multiple logistic or softmax regression is a little more complicated (and not so well documented in the literature). The gradient and Hessian for a single input pattern $\mathbf{x}$ are given by

$$\frac{\partial\mathcal{L}}{\partial\beta_{ki}} = (p_k - y_k)x_i \qquad \frac{\partial^2\mathcal{L}}{\partial\beta_{ki}\partial\beta_{lj}} = (p_l\delta_{kl} - p_l p_k)x_i x_j \tag{12}$$

To show that there is a unique maximum, it is sufficient to prove that the Hessian $\mathbf{H}$ is positive semi-definite. If $\mathbf{a}$ is an arbitrary vector, and we write $\mathbf{C} = (p_l\delta_{kl} - p_l p_k)$, then

$$\mathbf{a}^T\mathbf{H}\mathbf{a} = \mathbf{a}^T\mathbf{x}\mathbf{C}\mathbf{x}^T\mathbf{a} = (\mathbf{x}^T\mathbf{a})^T\mathbf{C}(\mathbf{x}^T\mathbf{a}) \geq 0 \tag{13}$$

since $\mathbf{C}$ is the covariance matrix of the multinomial distribution and is therefore positive semi-definite[2]. However, when we write the Hessian in the form $\Xi^T\mathbf{W}\Xi$, where Ξ is the $(mn) \times (mp)$ block matrix containing m copies of $\mathbf{X}$ along the diagonal (and p is the input dimension), the matrix $\mathbf{W}$ is an $m \times m$ block matrix, where each block is an $n \times n$ diagonal matrix containing the corresponding entries from $\mathbf{C}$ for each input pattern. $\mathbf{W}$ is no longer diagonal, but to compute its square root, we need only find a Cholesky decomposition of $\mathbf{C}$. However, because $\mathbf{C}$ has m^2 non-zero entries, it is no longer clear that this representation of the problem offers practical advantage. We have chosen to implement two alternative algorithms. In the first we calculate the exact Hessian by summing terms given by equation (12) for each row in the dataset. The resulting matrix is usually very ill-conditioned, but using singular value decomposition, it is numerically tractable to solve the original Fisher scoring equation

[1] In any case, for the canonical link, the Hessian coincides with its expected value.

[2] Thanks to Chris Williams for pointing this out.

(9). Alternatively, in a simplified algorithm, we can treat each output as independent, which yields the same update rule as for the logistic model (this is no surprise, since the marginal distribution for a single output in a multiple logistic model is binomial with probability p_i), although this is not a good approximation to the true Hessian. In either case, we initialise the parameters using the same procedure as for logistic regression, but treating each output independently.

3 NONLINEAR RBF NETWORKS FOR CLASSIFICATION

The output of RBF networks is usually given as a linear combination of basis functions:

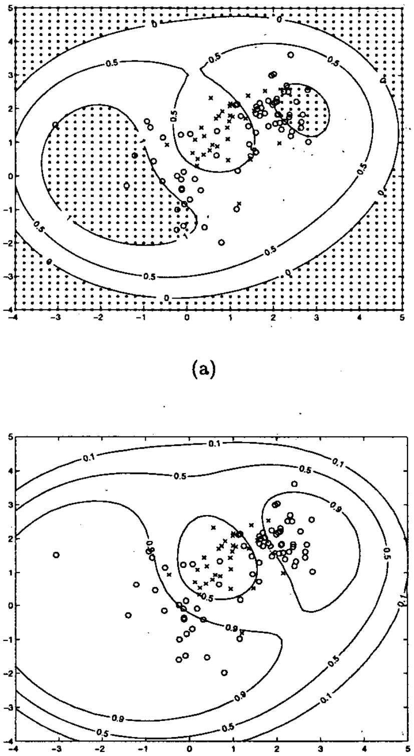
$$o_k(x) = \sum_j \phi_j(\|\mathbf{x} - \mathbf{x}^{(j)}\|)w_{jk} \tag{14}$$

where $\mathbf{x}^{(j)}$ is the 'centre' of the jth basis function. Once the parameters of the basis functions are fixed, the computation of the output weights is a linear regression problem, $\mathbf{Y} = \mathbf{\Phi W}$, with $\mathbf{\Phi}$ denoting the design matrix. As in equation (7), this is solved by computing the pseudo-inverse of $\mathbf{\Phi}$. Because of the form of the solution, any linear constraint on the training targets is necessarily satisfied by the network outputs [7]. For a multi-class classification problem, where a 1-of-m encoding is used, the network outputs will sum to one just as the targets do. However, the network outputs need not lie in the range $[0, 1]$ and so it may not always be possible to interpret them as probabilities. Instead, we replace the linear output layer with logistic (for 2 class) and softmax (for more than 2 classes) models and use the relevant IRLS algorithm from section 2.2 for training. Software implementing this model was developed using the NETLAB neural network toolbox[3].

4 EXPERIMENTAL RESULTS

In this section we present the results of using our method of training logistic output RBF networks on classification problems. Their performance is compared with linear output models, and the training algorithm with scaled conjugate gradient (SCG) and quasi-Newton optimisation algorithms ([3] is a useful reference). The initial output layer weights for the logistic models trained by SCG and quasi-Newton algorithms are taken from a linear output model trained with a pseudo-inverse, which is an efficient way to get a much better than random start point. The initialisation of the output weights for IRLS training was discussed in Section 2.2 and applies IRLS to a 'smoothed' version of the targets. The RBF networks used thin plate splines as basis functions (for the reasons given in [5]). The centres were adjusted using either K-means or the EM algorithm (so that they approximate the unconditional density of the input data). Note that in all results reported here, the reported computational effort does *not* include the centre selection phase and is solely for the training of the output layer weights and biases. All algorithms had the same stopping criterion; both the absolute change in the weight vector and the error function should be less than 1×10^{-4}.

4.1 SYNTHETIC DATASETS



(a)

(b)

Figure 1: 2 class synthetic data. (a) Contour plot of linear output RBF. Contours at 0, 0.5 and 1.0. In the shaded region the output cannot be interpreted as a probability. (b) Contour plot of logistic output RBF.

Two simple synthetic datasets have been created with a two-dimensional input space. In the two class case, data is drawn from a mixture of three Gaussians, two of which are assigned to one class. The generating parameters were selected so that the decision boundaries are non-linear.

The graph in figure 1a shows that with linear

[3] Available from http://www.ncrg.aston.ac.uk/netlab/index.html

outputs, there are regions of the input space where the outputs are not confined to the interval [0, 1], and that this can occur even for training data. The learning curves in figure 2 show that the IRLS training algorithm is significantly faster than either SCG or quasi-Newton, as is also shown in table 1.

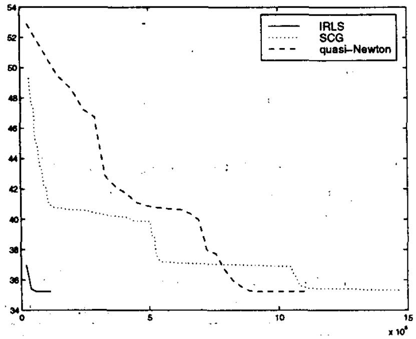


Figure 2: 2 class synthetic data. Learning curves (measured in flops) for logistic output RBF.

Algorithm	flops	Error
Linear outputs	22323	—
IRLS	122087	35.2262
SCG	1467342	35.3275
quasi-Newton	1142511	35.2262

Table 1: Results on a 2 class synthetic dataset.

To test the generality of this result, 10 replicated datasets were created by randomly sampling from the same mixture model. Table 2 contains the results of training 10 networks on these datasets. The error ratio is computed for each training set by dividing the error of each algorithm by the minimum error across all the algorithms. It shows that the IRLS is on average 6 times faster than the other two algorithms, and the computational effort is more consistent.

	IRLS	SCG	q-N
Mean flops ($\times 10^6$)	0.1635	1.0036	1.0623
S.d. flops ($\times 10^5$)	0.2137	4.0293	1.1322
Mean error ratio	1.000	1.0042	1.000

Table 2: Results on replicated 2 class synthetic dataset.

The three class synthetic dataset is drawn from a mixture of five Gaussians. Again, a linear output RBF can have non-probabilistic outputs in regions of input space where the density of training data is high (see figure 3a). The specialised training algorithms were an order of magnitude more efficient than SCG and quasi-Newton (see table 2). The 'simplified' algorithm was fastest to converge, but tends to take a large up-hill step when close to the maximum likelihood, so that the algorithm terminates at a sub-optimal value.

Algorithm	flops	Error
Linear outputs	59687	—
Softmax: exact Hessian	2050376	94.95
Softmax: simplified	478152	95.60
SCG	11031430	95.01
quasi-Newton	11370906	94.95

Table 3: Results on 3 class synthetic data

The results from 10 replicas of the 3 class data are given in Table 4. The IRLS algorithm is on average more than 4.6 times faster than SCG, and the computational effort is again more consistent. For 7 of the 10 datasets, the approximate Hessian failed to improve on the initial weights, and so the results are not tabulated.

	softmax exact H	SCG	q-N
Mean flops ($\times 10^7$)	0.2271	1.0586	1.2256
S.d. flops ($\times 10^6$)	0.3024	3.0739	0.6538
Mean error ratio	1.000	1.0042	1.000

Table 4: Results on replicated 3 class synthetic dataset.

4.2 REAL DATASETS

We have tried out our method on three well known classification problems: Leptograpsus crabs, diabetes in Pima women and forensic glass[4].

In the Leptograpsus crabs problem, the task is to determine the sex of crabs on the basis of 6 measurements. Using the same procedure reported in [9], we took 80 training examples and 120 test examples. The results, with some selected comparisons from [2], are reported in table 5. Note that SCG remained stuck in a local minimum despite several restarts, while IRLS and quasi-Newton achieved a training set log likelihood of near zero. Ten hidden units were used, since this was the smallest number for which the logistic RBF trained to a sufficiently low error.

In the diabetes diagnosis problem, the task is to diagnose whether a subject has diabetes or not on the basis of 8 variables measuring various disease indicators. There are 200 training examples, and 332 test examples. The default classifier (assigining every subject to the healthy class) has an error rate of 33%. The optimal RBF network had 8 hidden

[4] Available from
`http://markov.stats.ox.ac.uk/pub/PRNN`

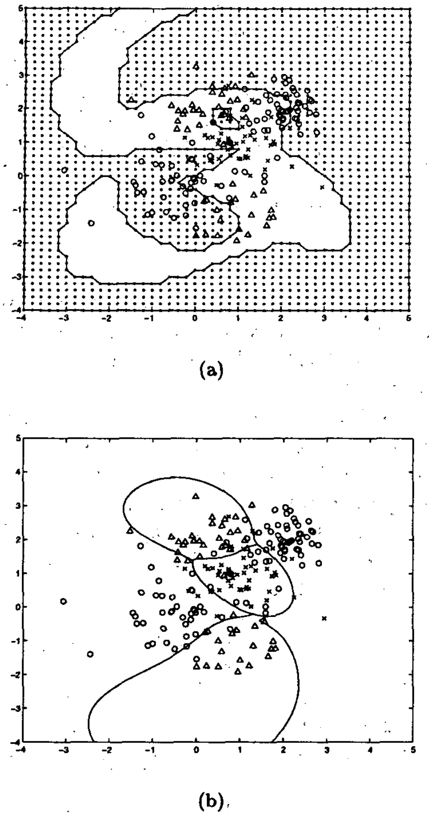


Figure 3: 3 class synthetic data. (a) Non-probabilistic predictions from linear output RBF (dotted region). (b) Decision boundaries of softmax output RBF.

units: its results are compared with those achieved by other models (as given in [10]) in table 6.

In the forensic glass problem, the task is to determine the type of a glass sample from the refractive index and composition (weight fraction of eight oxides). There are 214 examples with 6 classes, so performance is estimated using 10-fold cross-validation [10]. To improve performance, a committee of 10 networks was used for each partition, as was done by Ripley in [10] for the MLP. The results are contained in table 7; for comparison, the default rule (assigning to the largest class) has a misclassification rate of 65%. It should be noted that the computational effort for both MLP and Gaussian Process methods on this problem was very large (the latter required 24 hours on an SGI Challenge) compared with the softmax RBF approach (which took about 20 minutes on a less powerful computer). On the third dataset, the optimal number of hidden units for the linear output RBF was 25, while it was 12 for the non-linear output RBF.

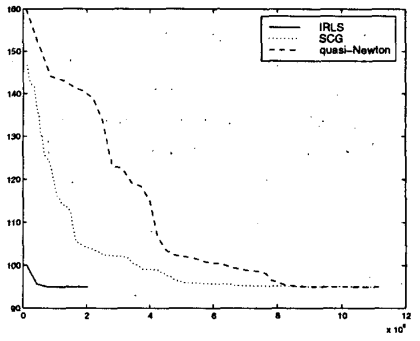


Figure 4: 3 class synthetic data. Learning curves (measured in flops) for softmax output RBF.

Algorithm	flops	Test Set Misclassifications
RBF: Linear outputs	48733	6
RBF: IRLS	364836	4
SCG	3330815	—
quasi-Newton	3245640	—
MLP	—	3
Linear Discriminant	—	8
Logistic Regression	—	4
MARS	—	22.6

Table 5: Results on crab data

5 DISCUSSION AND CONCLUSIONS

In this paper we have demonstrated that the benefits of using non-linear output functions for classification problems can be achieved with RBF networks while still retaining their significant training speed advantage over MLPs. In addition, the IRLS algorithm is considerably faster than using more general non-linear optimisation methods. In our experiments, IRLS achieved the same final error values as the quasi-Newton algorithm (to 4 decimal places), while scaled conjugate gradient often terminated at error values that were larger in the third sig-

Algorithm	flops	Miclassification Rate %
RBF: Linear outputs	96723	19.9
RBF: IRLS	436145	21.4
SCG	1469126	—
quasi-Newton	6493136	—
MLP	—	22.6
Linear Discriminant	—	20.2
Logistic Regression	—	19.9

Table 6: Results on diabetes data

Algorithm	Misclassification Rate %
RBF: Linear outputs	31.4
RBF: exact Hessian	30.3
MLP	23.8
Linear Discriminant	36.0
MARS	32.2
PP regression	35.0
Gaussian Process	23.3
Gaussian Mixture	30.8

Table 7: Results on forensic glass data. The simplified IRLS algorithm failed to converge.

nificant figure.

In the future we hope to extend many useful results for RBFs that depend on the pseudo-inverse solution for the output weights to the non-linear output models considered in this paper using equation (11). For example, [6] explains the link between the degrees of freedom of an RBF model and the eigenvalues of the design matrix and [11] gives an interpretation of the hidden units. The single maximum of the log likelihood means that a Bayesian approach to regularisation with the Laplace approximation is likely to be effective and we intend to pursue this further. In [4] there is an explanation of how to calculate the degrees of freedom for a generalised additive model, and it should be possible to apply this to RBFs.

Acknowledgements

This paper has benefitted from discussions with David Lowe and Chris Williams.

6 References

[1] P. Auer, M. Herbster, and M. K. Warmuth. Exponentially many local minima for single neurons. In *Neural Information Processing Systems 8*, pages 316–322, 1996.

[2] D. Barber and C. K. I. Williams. Gaussian Processes for Bayesian classification via hybrid Monte Carlo. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Neural Information Processing Systems 9*, 1997.

[3] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[4] T. J. Hastie and R. J. Tibsharani. *Generalized Additive Models.* Chapman and Hall, London, 1990.

[5] D. Lowe. On the use of nonlocal and non positive definite basis functions in radial basis function networks. In *IEE ANN 1995*, pages 206–211, 1995.

[6] D. Lowe. Characterising complexity in a radial basis function network. In *IEE ANN 1997*, pages 19–23, 1997.

[7] D. Lowe and A. R. Webb. Optimized feature extraction and the Bayes decision in feed-forward classifier networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4:355–364, 1991.

[8] P. McCullagh and J. A. Nelder. *Generalized Linear Models.* Chapman and Hall, London, 1983.

[9] B. D. Ripley. Flexible non-linear approaches to classification. In V. Cherkassy, J. H. Friedman, and H. Wechsler, editors, *From Statistics to Neural Networks*, pages 105–126. Springer, 1994.

[10] B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[11] A. R. Webb and D. Lowe. The optimised internal representation of multilayer classifier networks performs nonlinear discriminant analysis. *Neural Networks*, 3:367–375, 1990.