Stochastic Optimization Methods for Fitting Polyclass and Feed-Forward Neural Network Models

Charles KOOPERBERG and Charles J. STONE

Kooperberg, Bose, and Stone introduced POLYCLASS, a methodology that uses adaptively selected linear splines and their tensor products to model conditional class probabilities. The authors attempted to develop a methodology that would work well on small and moderate size problems and would scale up to large problems. However, the version of POLYCLASS that was developed for large problems was computationally impractical beyond a certain point. In this article we gain further insight into the fitting of large POLYCLASS models by simultaneously considering the fitting of large feed-forward neural network models with a single hidden layer (NEURALNET). In this combined setting, the stochastic gradient method, as used in the online version of the backpropagation method for fitting neural network models, and a stochastic version of the conjugate gradient method for fitting such models emerge as being computationally attractive. In particular, these stochastic methods are successfully applied to the fitting of POLYCLASS and NEURALNET models in the context of a phoneme recognition problem involving 45 phonemes, 81 features, 150,000 cases in the training sample, up to 1,000 basis functions and 44,000 parameters for POLYCLASS, and up to 800 hidden nodes and about 100,000 parameters for NEURALNET.

Key Words: Backpropagation; Conjugate gradient method; Linear splines; MARS; Multiple classification; Speech recognition; Stochastic approximation.

1. INTRODUCTION

Polychotomous regression and multiple classification are well studied subjects in statistics. In particular, Kooperberg, Bose, and Stone (1997) developed the POLYCLASS methodology, which uses adaptively selected linear splines and their tensor products to model conditional class probabilities. The methodology performed quite well on small and moderate size problems. A version of POLYCLASS was applied to a fairly large problem with some success, but it required approximately two months of CPU time (which was reduced to one day on a network of 64 workstations). The bulk of this time was spent in obtaining the maximum likelihood estimates of the coefficients of the POLYCLASS model

Charles Kooperberg is Associate Member, Division of Public Health Sciences, Fred Hutchinson Cancer Research Center, Seattle, WA 98109-1024 (Email: clk@fhcrc.org). Charles J. Stone is Professor, Department of Statistics, University of California, Berkeley, CA 94720-3860 (Email: stone@stat.berkeley.edu).

Journal of Computational and Graphical Statistics, Volume 8, Number 2, Pages 169–189

corresponding to a given set of basis functions. The quasi-Newton algorithm, which was used for this purpose, is really impractical for such a large problem.

Feed-forward neural networks (see, e.g., Cheng and Titterington 1994; Ripley 1994, 1996) are frequently used for large classification problems. They have several advantages, including generally good performance and ease of implementation. A POLYCLASS model can be viewed as a special type of single layer (no hidden layer) feed-forward neural network, in which the inputs are not the raw features, but adaptively selected basis functions. Inspired by the success of stochastic (online) gradient versions of the backpropagation algorithm for fitting such neural networks (Rumelhart and McClelland 1986), we investigate in this article the use of the stochastic gradient method instead of the quasi-Newton algorithm in fitting POLYCLASS models corresponding to given sets of basis functions. We also develop and study a stochastic version of the conjugate gradient method of optimization for the fitting of such models.

In Section 2 of this article, we discuss POLYCLASS. There we also discuss feed-forward neural networks. In this investigation we concentrate on networks of this form having a single hidden layer, which we dub NEURALNET. The stochastic gradient method for fitting POLYCLASS and NEURALNET models is discussed in Section 3. In Section 4 we discuss the conjugate gradient method for fitting such models and a stochastic version of this method. In Section 5 we compare the performance of various optimization methods in fitting POLYCLASS and NEURALNET models in the context of a phoneme recognition problem involving 45 phonemes and also in the context of a three-vowel subproblem. Our concluding remarks are given in Section 6.

2. POLYCHOTOMOUS REGRESSION

Consider a qualitative random variable Y that takes on a finite number K of values, which we refer to as classes. We can think of Y as ranging over $\mathcal{K} = \{1, \ldots, K\}$. Suppose the distribution of Y depends on features $x_1, \ldots, x_M$, where $\mathbf{x} = (x_1, \ldots, x_M)$ ranges over a subset $\mathcal{X}$ of $\mathbb{R}^M$. Let $\mathbf{x}$ now be distributed as a random vector; that is, consider the random pair $(\mathbf{X}, Y)$, where $\mathbf{X}$ is an $\mathcal{X}$-valued random vector and Y is a $\mathcal{K}$-valued random variable. In the multiple classification problem we want to predict Y based on $\mathbf{X}$. The well known optimal rule is to predict Y to be $\arg\max_k P(Y = k|\mathbf{X})$. While many classification methods try to find $\arg\max_k P(Y = k|\mathbf{X})$ directly, there are problems in which conditional class probabilities are required and direct classification does not suffice.

Suppose that $P(Y = k|\mathbf{X} = \mathbf{x}) > 0$ for $\mathbf{x} \in \mathcal{X}$ and $k \in \mathcal{K}$. Let $\psi(\mathbf{x})$ be any function on $\mathcal{X}$ and set

$$\theta(k|\mathbf{x}) = \log P(Y = k|\mathbf{X} = \mathbf{x}) - \psi(x), \qquad \mathbf{x} \in \mathcal{X} \quad \text{and} \quad k \in \mathcal{K}.$$

Then

$$P(Y = k|\mathbf{X} = \mathbf{x}) = \frac{\exp \theta(k|\mathbf{x})}{\sum_k \exp \theta(k|\mathbf{x})}, \qquad \mathbf{x} \in \mathcal{X} \quad \text{and} \quad k \in \mathcal{K}. \tag{2.1}$$

Thinking of $\theta(1|\mathbf{x}), \ldots, \theta(K|\mathbf{x})$ as unknown functions, we refer to (2.1) as the polychotomous regression model; when $K = 2$ it is referred to as the logistic regression

model. Observe that the model in (2.1) is nonidentifiable in that it does not involve the function ψ. To obtain an identifiable model, we add the restriction that

$$\theta(K|\mathbf{x}) = 0, \qquad \mathbf{x} \in \mathcal{X}. \tag{2.2}$$

With this restriction, (2.1) is now an identifiable model; indeed,

$$\theta(k|\mathbf{x}) = \log \frac{P(Y = k|\mathbf{X} = \mathbf{x})}{P(Y = K|\mathbf{X} = \mathbf{x})}, \qquad \mathbf{x} \in \quad \mathcal{X} \quad \text{and} \quad k \in \mathcal{K}.$$

2.1 POLYCLASS

There are a number of attractive approaches to the fitting of a polychotomous regression model to observed data. One such approach is to model the functions $\theta(1|\mathbf{x}), \ldots, \theta(K|\mathbf{x})$ in a suitable common finite-dimensional linear space G of functions on $\mathcal{X}$. Let p denote the dimension of G and let $B_1, \ldots, B_p$ be a basis of this space. Consider the model

$$\theta(k|\mathbf{x}; \boldsymbol{\beta}_k) = \sum_{j=1}^{p} \beta_{j\,k} B_j(\mathbf{x}), \qquad \mathbf{x} \in \mathcal{X} \quad \text{and} \quad k \in \mathcal{K}, \tag{2.3}$$

for the unknown functions $\theta(1|\mathbf{x}), \ldots, \theta_K|\mathbf{x})$; here $\boldsymbol{\beta}_k = (\beta_{1k}, \ldots, \beta_{pk})^T$, $1 \le k \le K$, are the unknown vectors of parameters. Correspondingly, we get the finite-dimensional polychotomous regression model

$$P(Y = k|\mathbf{X} = \mathbf{x}; \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K) = \frac{\exp \theta(k|\mathbf{x}; \boldsymbol{\beta}_k)}{\sum_k \exp \theta(k|\mathbf{x}; \boldsymbol{\beta}_k)}, \qquad \mathbf{x} \in \mathcal{X} \quad \text{and} \quad k \in \mathcal{K}. \tag{2.4}$$

To obtain an identifiable model, we impose the restriction (2.2) or, equivalently that $\theta(K|\mathbf{x}; \boldsymbol{\beta}_K) = 0$ or that $\boldsymbol{\beta}_K = \mathbf{0}$. [Except for floating point roundoff errors, the POLYMARS and POLYCLASS methodologies, defined in the following, are invariant under permutations of the values of Y. Without some such identifiability restriction, the POLYCLASS log-likelihood function would fail to be concave, its Hessian would fail to be negative definite and invertible, and the corresponding maximum likelihood estimate would fail to be uniquely defined. This would complicate some of the numerical techniques for obtaining the maximum likelihood estimates. Also, the Rao and Wald statistics that are used in stepwise addition and stepwise deletion, respectively, are conveniently calculated by using the inverse of the Hessian; see Kooperberg et al. (1997).] Let $\boldsymbol{\beta}$ denote the $p(K-1)$-dimensional column vector consisting of the entries of $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{K-1}$. Then $\boldsymbol{\beta}$ ranges over $\mathbb{R}^{p(K-1)}$. Correspondingly, we rewrite $\theta(k|\mathbf{x}; \boldsymbol{\beta}_k)$ as $\theta(k|\mathbf{x}, \boldsymbol{\beta})$, and we rewrite (2.4) as

$$P(Y = k|\mathbf{X} = \mathbf{x}; \boldsymbol{\beta}) = \frac{\exp \theta(k|\mathbf{x}; \boldsymbol{\beta})}{\sum_k \exp \theta(k|\mathbf{x}; \boldsymbol{\beta})}, \qquad \mathbf{x} \in \mathcal{X} \quad \text{and} \quad k \in \mathcal{K}. \tag{2.5}$$

An important advantage of a finite-dimensional linear model for $\theta(k|\mathbf{x})$ over such a model for $P(k|\mathbf{X} = \mathbf{x})$ directly is that in the former case the resulting model for $P(k|\mathbf{X} = \mathbf{x})$

given by (2.5) automatically yields estimated probabilities that are strictly between 0 and 1 and hence yields well-defined estimates of $\log P(k|\mathbf{X} = \mathbf{x})$, which are needed in some applications such as the approach to speech recognition described by Bourlard and Morgan (1994).

For theoretical purposes, we think of the data $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$ as the observed values of a random sample of size n from the distribution of $(\mathbf{X}, Y)$. In principle, the space G or, equivalently, a basis of this space could be determined either in an adaptive manner (i.e., based on all of the observed data) or in a nonadaptive manner (i.e., not based on the observed data or, more generally, based only on $\mathbf{X}_1, \ldots, \mathbf{X}_n$). Theoretical motivation for using nonadaptively selected polynomial splines and their low-order tensor products in polychotomous regression and other extended linear modeling contexts (e.g., regression, generalized regression, density estimation, hazard regression, spectral density estimation, and event history analysis) is given in Stone, Hansen, Kooperberg, and Truong (1997) and the references cited therein. In short, this approach can be used for dimensionality reduction (i.e., to ameliorate the "curse of dimensionality") and to obtain more easily interpretable models. In these theoretical results, when G is chosen optimally its dimension increases at a rate proportional to n^γ for some $\gamma \in (0, 1/2)$ that depends explicitly on specified parameters in the theoretical formulation.

2.1.1 Adaptive Model Selection

In the adaptive POLYCLASS methodology developed in Kooperberg et al. (1997), linear splines and their tensor products are used to model the unknown functions $\theta(\cdot|\mathbf{x})$. Specifically, the basis functions that are used are of the following types:

- linear functions in one of the features;
- piecewise linear functions in a feature x of the form $(x - t)_+ = (x - t)$ if $x > t$ and 0 otherwise, where the knot t is a number within the range of the feature x;
- tensor products of two basis functions that depend on different features.

A new function can be added to an existing set of basis functions only if the resulting space is *allowable* as described in Kooperberg et al. (1997). For small and medium size problems, basis functions are selected using the following algorithm:

1. fit the initial model $\theta(k|\mathbf{x}) = \beta_{1k}$ for $k = 1, \ldots, K - 1$;
2. determine which basis functions can be added to the model so that the resulting space is allowable; among these basis functions find the one having the largest Rao (score) statistic;
3. add that basis function to the space and fit the resulting model using maximum likelihood;
4. return to Step 2 until the linear space has reached a prespecified maximum dimension;
5. for those basis functions that can be removed from the allowable space so that the resulting subspace is also allowable, find the one having the smallest Wald statistic;

6. remove that basis function and refit the resulting model using maximum likelihood;
7. return to Step 5 until only the constant basis function remains;
8. out of the double sequence of models, select the model with the lowest BIC value, the highest test set log-likelihood, or the lowest test set classification error.

Since Rao and Wald statistics are quadratic approximations to differences in log-likelihood, they are reasonable measures to use in the selection of basis functions when a maximum likelihood fitting procedure is employed. See Kooperberg et al. (1997, secs. 3.1 and 3.2, app. A) for more details.

Kooperberg et al. (1997) pointed out that for large datasets this model-selection procedure requires approximately $O(K^2 p_{\max}^3 n)$ floating point operations (flops), where $p_{\max}$ is the maximum number of basis functions in any model, K is the number of classes, and n is the sample size. For the phoneme recognition example discussed in Section 5, $n \approx 150{,}000$, $K = 45$, and $p_{\max}$ may be as large as 1,000. Thus, more than 10^{17} flops would be required, which would be equivalent to decades of CPU time on the workstations that were available.

As a much faster alternative, Kooperberg et al. (1997) proposed using a least-squares approximation to the stepwise addition process when dealing with large models and datasets. The estimate $\widehat{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$ is obtained by minimizing $V(\boldsymbol{\beta}) = \sum_i \sum_k [\text{ind}(Y_i = k) - \theta(k|\mathbf{X}_i; \boldsymbol{\beta})]^2$, where $\theta(k|\mathbf{X}_i; \boldsymbol{\beta}) = \sum_{j=1}^{p} \beta_{j\,k} B_j(\mathbf{X}_i)$. The selection of the new basis function is carried out by minimizing $V(\widehat{\boldsymbol{\beta}})$, while the same allowable spaces as in POLYCLASS are used. The stepwise addition part of the model selection procedure now takes approximately $O(M p_{\max}^2 n)$ flops, where M is the number of features. Thus, for the same phoneme recognition dataset, it would take approximately one day of CPU time to select 400 basis functions and approximately one week to select 1,000 basis functions. We refer to the selection of basis functions using Rao statistics as POLYCLASS selection of basis functions and selection using least squares approximation as POLYMARS selection.

After selecting the basis functions, we need to obtain a reasonably accurate approximation to the maximum likelihood estimate $\widehat{\boldsymbol{\beta}}$ of the coefficient vector $\boldsymbol{\beta}$ in (2.5). For a given collection of basis functions the corresponding log-likelihood function is concave, so this numerical approximation problem is conceptually straightforward. There is a large literature about optimization of concave functions. A reference that we found particularly useful is Kennedy and Gentle (1980). When basis functions are selected with POLYCLASS, the approximation to $\widehat{\boldsymbol{\beta}}$ is obtained using a Newton–Raphson algorithm, which takes an order of magnitude less CPU time than POLYCLASS selection of the basis functions since more full Hessians have to be computed for the Rao statistics than for the Newton–Raphson iterations to compute $\widehat{\boldsymbol{\beta}}$.

On the other hand, when $K p_{\max}$ is large and the basis functions are selected using POLYMARS, there are many optimization methods that are orders of magnitude faster than Newton–Raphson, one being a quasi-Newton algorithm, in which a "pseudo-Hessian" is updated (we have used a BFGS updating formula) at each iteration by combining information about the gradient at the previous iteration and the current iteration. Kooperberg et al. (1997) pointed out that this would require $O(200 K p_{\max}^2 n)$ flops, or several months of CPU time, for the phoneme recognition problem. The main reason that a quasi-Newton

algorithm takes so long is that a large number of gradients have to be computed before a workable pseudo-Hessian is obtained; thus, the initial speed of convergence is very slow. An additional disadvantage of quasi-Newton methods is that we have to store the pseudo-Hessian. For the phoneme recognition problem this can be as large as a $45{,}000 \times 45{,}000$ matrix, which is infeasible on typical current computers.

2.2 FEED-FORWARD NEURAL NETWORKS

Feed-forward neural networks (also known as multilayer perceptrons) can also be used to construct polychotomous regression models. Consider such a neural network having L layers including the output layer L and hidden layers $1, \ldots, L-1$. We refer to the inputs x_m, $m \geq 1$, as forming layer 0. Let θ_{lk} denote the value of the kth unit in layer l for $l, k \geq 1$, where the number of units is allowed to vary from layer to layer. Also, set $\pi_{l0} = 1$ for $0 \leq l < L$, $\pi_{0m} = x_m$ for $m \geq 1$, $\pi_{lk} = (\exp\theta_{lk})/(1+\exp\theta_{lk})$ for $1 \leq l < L$ and $k \geq 1$, and $\pi_{Lk} = (\exp\theta_{Lk})/\sum_k \exp\theta_{Lk}$ for $1 \leq k \leq K$ (which is sometimes referred to as *softmax* in the neural network literature). Then $\theta_{lk} = \sum_m \beta_{lmk}\pi_{l-1,m}$ for $l, k \geq 1$, where the coefficients β_{lmk} for $1 \leq l \leq L$, $k \geq 1$, and $m \geq 0$ are to be determined. Given the data $(\mathbf{x}, y)$ for a single case in the training set, set $\delta_y = 1$ and $\delta_k = 0$ for $k \neq y$. The corresponding log-likelihood is given by $\ell = \sum_k \delta_k\theta_{Lk} - \log(\sum_k \exp\theta_{Lk})$. (Again, we impose the restriction (2.2); that is, we set $\theta_{LK} = 0$ and $\beta_{LmK} = 0$ for $m \geq 0$. In the context of neural networks this restriction is harmless, but it doesn't help either since the log-likelihood function is highly multimodal even after the imposition of the restriction; see the discussion following (2.4).

Think of the log-likelihood as a function of the coefficients. To determine the gradient of this function, we observe that

$$\frac{\partial\ell}{\partial\beta_{lmk}} = \pi_{l-1,m}\frac{\partial\ell}{\partial\theta_{lk}}, \qquad 1 \leq l \leq L, \tag{2.6}$$

$$\frac{\partial\ell}{\partial\theta_{lk}} = \sum_j \frac{\partial\ell}{\partial\theta_{l+1,j}}\frac{\partial\theta_{l+1,j}}{\partial\theta_{lk}}, \qquad 1 \leq l < L, \tag{2.7}$$

and

$$\frac{\partial\theta_{l+1,j}}{\partial\theta_{lk}} = \pi_{lk}(1-\pi_{lk})\beta_{l+1,k,j}, \qquad 1 \leq l < L. \tag{2.8}$$

We conclude from (2.7) and (2.8) that

$$\frac{\partial\ell}{\partial\theta_{lk}} = \pi_{lk}(1-\pi_{lk})\sum_j \beta_{l+1,k,j}\frac{\partial\ell}{\partial\theta_{l+1,j}}, \qquad 1 \leq l < L. \tag{2.9}$$

Moreover,

$$\frac{\partial\ell}{\partial\theta_{Lk}} = \delta_k - \pi_{Lk}. \tag{2.10}$$

We use (2.9) and (2.10) to compute $\partial\ell/\partial\theta_{lk}$ successively for $l = L, \ldots, 1$ and use (2.6) to compute $\partial\ell/\partial\beta_{lmk}$. This method of computing the gradient of the log-likelihood function is known as *backpropagation*.

3. STOCHASTIC GRADIENT METHOD

3.1 POLYCLASS

We can think of polychotomous regression as corresponding to a single-layer network (no hidden layers) having inputs $B_m(\mathbf{x})$, $m \geq 1$, and outputs θ_k, $k \geq 1$. Set $\pi_k = (\exp\theta_k)/(1 + \exp\theta_k)$ for $k \geq 1$. Then $\theta_k = \sum_m \beta_{mk} B_m(\mathbf{x})$ for $k \geq 1$, where the coefficients β_{mk}, $k \geq 1$ and $m \geq 0$, are to be determined. Given the data $(\mathbf{x}, y)$ for a single case in the training set, set $\delta_y = 1$ and $\delta_k = 0$ for $k \neq y$. The corresponding (contribution to the) log-likelihood is given by $\ell = \sum_k \delta_k \theta_k - \log(1 + \sum_k \exp\theta_k)$. Think of the log-likelihood as a function of the coefficients. Since

$$\frac{\partial \ell}{\partial \theta_k} = \delta_k - \pi_k,$$

the gradient of the log-likelihood function is given by

$$\frac{\partial \ell}{\partial \beta_{mk}} = (\delta_k - \pi_k) B_m(\mathbf{x}).$$

In the stochastic gradient method, we successively update the coefficients on a case-by-case basis according to the formula

$$\beta_{mk}^{(i+1)} = \beta_{mk}^{(i)} + r_i \frac{\partial \ell}{\partial \beta_{mk}}, \tag{3.1}$$

where ℓ is the log-likelihood at $\boldsymbol{\beta}^{(i)} = (\beta_{mk}^{(i)})$ based on a single case. We go through the cases in the training set in random order and make a number of passes through the data, choosing a fresh random order on each pass. Note that each pass requires $O(Kpn)$ flops. Note also that, in contrast to the quasi-Newton method, the stochastic gradient method requires a separate fit for each model (collection of basis functions) under consideration.

To apply the stochastic gradient method we need to address several issues:

- How should we adjust the *learning rate* r_i?
- Do the parameter estimates based on the stochastic gradient method "converge" and, if so, do they get close to the maximum likelihood estimates $\widehat{\boldsymbol{\beta}}$?
- How do we assess the convergence?
- How many passes through the data do we need?

These issues are simpler for POLYCLASS than for neural networks, since for POLYCLASS the log-likelihood function is strictly concave and hence there is a unique maximum of the log-likelihood function.

For selected sets of basis functions we fit a POLYCLASS model to the corresponding linear spaces using the stochastic gradient method. Initially, we set $\beta_{mk} = 0$ for all m and k. (Since the log-likelihood for POLYCLASS is concave, the initial values used in the stochastic gradient method are largely irrelevant.)

We tried a number of schemes for adjusting the learning rate. If this rate is reduced too slowly, the algorithm converges too slowly; although it appears that the misclassification error and log-likelihood have stabilized, the parameter estimates remain unstable

and they do not get close to the corresponding maximum likelihood estimates. If the learning rate is reduced too rapidly, the change in the parameters may become too small, so that the log-likelihood does not get close to its maximum.

Eventually, we settled on starting with an initial rate r_0 and dividing this rate by two after every ten full passes through the data. We found a reasonable, simple rule for determining the initial rate in our examples (see Sec. 5), but we also noticed that the accuracy of the approximations after a few passes through the data is very indicative of the accuracy after a much larger number of passes.

3.2 FEED-FORWARD NEURAL NETWORKS

In the context of feed-forward neural networks, the stochastic gradient method is as described in Section 3.1 with (3.1) replaced by

$$\beta_{lmk}^{(i+1)} = \beta_{lmk}^{(i)} + r_i \frac{\partial \ell}{\partial \beta_{lmk}}.$$

It is easily seen that a feed-forward neural network with M input variables, K features, and p_l hidden units in the lth hidden layer requires $O(RN)$ flops per pass through the data, where $R = Mp_1 + \sum_{l=1}^{L-2} p_l p_{l+1} + p_{L-1}K$. In particular, such a network with one hidden layer having p hidden units requires $O((M+K)pn)$ flops per pass.

In the experiments that are described in Section 5, we use feed-forward neural networks with a single hidden layer, which we refer to for convenience as NEURALNET. In fitting these models, we use roughly the same scheme for adjusting the learning rate as in POLYCLASS, except that in addition to dividing the rate by two after every ten full passes through the data, we divide it by ten after the first five full passes. [By comparison, Bourlard and Morgan (1994, p. 270) ultimately started with an initial learning rate such as .01 and successively divided the learning rate by 2 after each complete pass through the data.]

3.3 STOCHASTIC APPROXIMATION

There is a rich and voluminous literature on stochastic approximation, starting with Robbins and Monro (1951), which included the one-pass version of the stochastic gradient method as a special case. In particular, in the context of fitting POLYCLASS models with a fixed collection of basis functions or NEURALNET models with a fixed number of hidden units, White (1989) applied results of Ljung (1977) to obtain conditions under which convergence should occur as $n \to \infty$. In these conditions the random pairs $(\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots$ should be independent and have a common distribution with compact support and the learning rate r_n should be (say) of the form $A/(B+n)$. With probability one, the successive iterates $\boldsymbol{\beta}^{(n)}$ for $\widehat{\boldsymbol{\beta}}$ then either converge to a local maximum of the log-likelihood function or diverge ($|\boldsymbol{\beta}^{(n)}| \to \infty$). In particular, when the log-likelihood function is strictly concave, as in the fitting of a POLYCLASS model, the successive iterates either converge to $\widehat{\boldsymbol{\beta}}$ or diverge. We are unaware of similar theoretical results that deal with repeated passes through the data or with models whose size

(e.g., the number of basis functions of POLYCLASS or the number of hidden units of the NEURALNET model) depends on n.

4. STOCHASTIC CONJUGATE GRADIENT METHOD

4.1 CONJUGATE GRADIENT METHOD

The conjugate gradient method for numerically approximating the maximum likelihood estimate $\widehat{\boldsymbol{\beta}} = \operatorname{argmax} \ell(\boldsymbol{\beta})$ is an iterative method. At the beginning of the $(\nu + 1)$th iteration, where ν is a nonnegative integer, we know the approximation $\boldsymbol{\beta}^{(\nu)}$ determined during the νth iteration, the value $\mathrm{g}^{(\nu+1)}$ of the gradient of the log-likelihood function at $\boldsymbol{\beta}^{(\nu)}$, and the search direction $\boldsymbol{\gamma}^{(\nu)}$ used during the νth iteration ($\boldsymbol{\gamma}^{(0)} = \mathbf{0}$). The search direction used during the $(\nu + 1)$th iteration is given by $\boldsymbol{\gamma}^{(\nu+1)} = \mathrm{g}^{(\nu+1)} + b\boldsymbol{\gamma}^{(\nu)}$ for some number b, and the corresponding approximation to $\widehat{\boldsymbol{\beta}}$ is given by $\boldsymbol{\beta}^{(\nu+1)} = \boldsymbol{\beta}^{(\nu)} + a\boldsymbol{\gamma}^{(\nu+1)}$, where a is determined by a line search. (In a minimization problem, $\boldsymbol{\gamma}^{(\nu+1)} = -\mathrm{g}^{(\nu+1)} + b\boldsymbol{\gamma}^{(\nu)}$.)

Dixon (1975, eq. 6–9) listed four choices of b. We experimented with all of these choices, but found that, for our current problem, the choice of b has little influence on the results. We eventually settled on

$$b = \frac{\mathrm{g}^{(\nu+1)^T}(\mathrm{g}^{(\nu+1)} - \mathrm{g}^{(\nu)})}{\mathrm{g}^{(\nu)^T}\mathrm{g}^{(\nu)}},$$

which Dixon (1975) attributed to Polak and Ribiere (1969). Some authors advocate resetting the search direction to the steepest ascent direction (i.e., setting $b = 0$) after every so many iterations. In our experiments, however, such a modification did not yield a significant improvement in performance.

In many references about optimization it is suggested that good line searches are particularly important for conjugate gradient methods. In our setting this does not appear to be the case. Although more accurate line searches do reduce the number of required passes through the data somewhat, the CPU time spent on each individual pass increases sufficiently to completely offset any gains. A reason for that may be that in none of our problems do we get very close to the exact parameter vector that maximizes the log-likelihood function.

We ended up by using the following algorithm to approximate the value $\tilde{a}$ of a that maximizes $\ell(\boldsymbol{\beta} + a\boldsymbol{\gamma})$, where $\boldsymbol{\gamma}$ is the search direction found, for example, by the conjugate gradient method described previously:

1. Rescale $\boldsymbol{\gamma}$ to have the same norm as $a\boldsymbol{\gamma}$ at the end of the previous step (so that $a = 1$ may be a reasonable choice for a).
2. Find three numbers a_0, a_1, a_2 in the set $\{0\} \cup \{\pm 2^i : i \text{ is a nonnegative integer}\}$ such that $a_0 < a_1 < a_2$ and $\ell(\boldsymbol{\beta} + a_1\boldsymbol{\gamma}) > \max\big(\ell(\boldsymbol{\beta} + a_0\boldsymbol{\gamma}), \ell(\boldsymbol{\beta} + a_2\boldsymbol{\gamma})\big)$.
3. Find $\tilde{a}$ using quadratic interpolation.

For both POLYCLASS and NEURALNET, computing $\ell(\boldsymbol{\beta} + a\boldsymbol{\gamma})$ requires computing $P(Y = k|\mathbf{X} = \mathbf{x}; \boldsymbol{\beta} + a\boldsymbol{\gamma})$. For POLYCLASS this can be done very rapidly for many

values of a. To see this, note that

$$\exp\theta(k|\mathbf{x};\boldsymbol{\beta}+a\boldsymbol{\gamma}) = \exp\left(\sum_{j=1}^{p}\beta_{jk}B_j(\mathbf{x})\right)\left[\exp\sum_{j=1}^{p}\gamma_{jk}B_j(\mathbf{x})\right]^a.$$

Thus, if we store $\exp\theta(k|\mathbf{x}_i;\boldsymbol{\beta})$ and $\exp\theta(k|\mathbf{x}_i;\boldsymbol{\gamma})$ for all i and k, we can compute $\ell(\boldsymbol{\beta}+a\boldsymbol{\gamma})$ for every integer a in $O(nK)$ flops without additional exponentiations. Although, unfortunately, there is no similar approach for NEURALNET, substantial saving in CPU time can still be achieved by computing $\ell(\boldsymbol{\beta}+a\boldsymbol{\gamma})$ for several values of a at the same time.

Kennedy and Gentle (1980) noted that for exact maximization of a quadratic function, a conjugate gradient method would need as many computations of the gradient as there are variables, which in our situation would require $O(K^2p_{\max}^2n)$ flops. In practice, far fewer iterations are needed to obtain a reasonable solution, but, as in the quasi-Newton method, the initial convergence may be very slow. An important advantage of the conjugate gradient method over the quasi-Newton method when $Kp_{\max}$ is large is that the former method does not require the storage of a Hessian matrix.

4.2 STOCHASTIC VERSION

Let us refer to any subset of the training set as a *block*. Given such a block, we can write the corresponding normalized log-likelihood as

$$\bar{\ell}_{\text{block}}(\boldsymbol{\beta}) = \frac{1}{\text{blocksize}}\sum_{i\in\text{ block}}\ell_i(\boldsymbol{\beta}).$$

We can think of $\bar{\ell}_{\text{block}}$ as an estimate of $\Lambda(\boldsymbol{\beta}) = E[\ell(\boldsymbol{\beta})]$, where $\ell(\boldsymbol{\beta})$ is the log-likelihood based on a single random case. The maximum likelihood estimate $\widehat{\boldsymbol{\beta}} = \operatorname{argmax}\bar{\ell}_{\text{trainingset}}(\boldsymbol{\beta})$ can thereby be viewed as a Monte Carlo estimate of $\boldsymbol{\beta}^* = \operatorname{argmax}\Lambda(\boldsymbol{\beta})$. If n is large, it may be computationally attractive to use $\widehat{\boldsymbol{\beta}}_{\text{block}} = \operatorname{argmax}\bar{\ell}_{\text{block}}(\boldsymbol{\beta})$ as an estimate of $\boldsymbol{\beta}^*$.

Consider now the conjugate gradient method for finding the MLE. It may be worthwhile to use a different block at each iteration. This leads to a modification of the conjugate gradient method in which, at the $(\nu+1)$th iteration, the gradient $\mathbf{g}^{(\nu+1)}$ and the line search are based on $\bar{\ell}_{\text{block}^{(\nu+1)}}$.

Suppose we want to make a single pass through the data. A natural approach would be to partition the data randomly into S blocks of prespecified size and then iterate, starting with a prespecified $\boldsymbol{\beta}^{(0)}$. In practice, however, we need to make repeated passes through the data. With this in mind, it is reasonable to let each block size in a given pass be approximately n/S and to let S increase from pass to pass, as more accuracy is needed at later passes. On each pass, we should use a fresh random partition of the training set.

The problem remains of choosing the initial value $S^{(0)}$ of S and of coming up with a rule for increasing S. Based on some experimenting, we propose using $S^{(i)} = 2^{L_0}$ for $i = 0, 1, 2$ and $S^{(i)} = 2^{\min(0, L_0+2-i)}$ for $i > 2$. If L_0 is too small, the initial block size is

very large and the method is not much faster than a (nonstochastic) conjugate gradient method. If L_0 is too large, the initial passes could have an adverse effect on accuracy; in particular, $\boldsymbol{\beta}^{(1)}$ could be further away from $\boldsymbol{\beta}$ than is $\boldsymbol{\beta}^{(0)}$. Fortunately, we have found that even a single pass through the data is almost always indicative of the best choice of L_0. When i increases $S^{(i)}$ eventually equals 1, after which the method essentially coincides with a nonstochastic conjugate gradient method.

We refer to this method as the *stochastic conjugate gradient method*, as does Møller (1993), where a similar method is proposed in the context of fitting neural networks (see also Ripley 1997, p. 154). The motivation for using such a method is that it can perform numerous iterations and get close to convergence with only a moderate number of passes through the data. The larger the sample size n, the more redundancy there is in the data and hence the more attractive is this approach.

5. EXPERIMENTS

5.1 DATA

In our experiments we used the Numbers93 database, whose source is the Center for Spoken Language Understanding in Portland, OR (Cole, Roginski, and Fanty 1992; Cole et al. 1994). This database involves 2,165 utterances from telephone calls, which are numbers that typically are parts of addresses, zip codes, and street numbers. Each utterance was processed by one or more listeners, who produced a time-aligned phonetic description of the utterance. For example, for one particular utterance, "3o3" (three-oh-three), it was determined that from 1 millisecond (ms) to 167 ms, the speaker produced phoneme `T`, followed by phoneme `r` from 167 ms to 193 ms, and so on. It should be noted that the person who classified the phoneme being spoken was not aware of the text of the utterance. The phoneme transcription, which we obtained from the International Computer Science Institute (ICSI) in Berkeley, CA, is based on the LIMSI phonetic alphabet (Gauvain, Lamel, Adda, and Adda-Decker 1994).

The utterances were also processed to produce perceptual linear predictive (PLP) features (Hermansky 1990; Rabiner and Juang 1993; Bourlard and Morgan 1994). Every 12.5 ms the audible spectrum was determined from a concentric 25 ms interval of sound. Because we are using telephone data, which is sampled at the frequency of 8 kHz, there are 200 observations of the sound in such a 25 ms interval. A Hamming window was applied to these 200 observations, after which the spectrum was estimated using the discrete Fourier transform. The estimated spectrum was next transformed to yield a critical-band integrated power spectrum with an equal-loudness preemphasis and a cube root nonlinearity to simulate the auditory intensity-loudness relation. Then the eighth-order autoregressive all-pole model of the transformed spectrum was obtained. The coefficients of the Fourier transform representation of the log-magnitude of this model are known as its cepstral coefficients. The nine PLP features that we used for the 25 ms interval of sound under consideration are the log-gain of the model (similar to the variance) and the next eight cepstral coefficients (similar to autoregressive coefficients).

The main task in our experiments was to estimate the conditional probability distribution of the phoneme being spoken at a certain time, given the nine PLP features for

the interval of length 25 ms centered at the time in question as well as the nine features for each of the four intervals centered at 100 ms, 75 ms, 50 ms, and 25 ms before the time in question and each of the four intervals centered at 25 ms, 50 ms, 75 ms, and 100 ms after that time. Thus, we have 81 features in all.

Such a conditional probability distribution (or, more precisely, a likelihood that is obtained by weighting the estimated probabilities by the empirically determined frequencies of the phonemes) can be used as input to train (estimate) a hidden Markov model, which in turn can be used for automatic speech recognition (Bourlard and Morgan 1994). In the hybrid approach described by Bourlard and Morgan, a feed-forward neural network is used to estimate these probabilities.

For each utterance in the database, we used the phonemes being spoken at times 12.5 ms apart to define the cases. We eliminated the phonemes being spoken at the beginning and end of the utterance in order to avoid missing values for any of the features. We also eliminated one phoneme represented only by three cases. In this manner, we obtained 45 phonemes and about 250,000 cases. We randomly divided the data into a training set of 153,426 cases and a test set of 102,239 cases. We used both the complete set of data and the subset of the data consisting of all occurrences of three phonemes: the vowels in b*ee*t, b*e*t, and b*ou*ght. These phonemes were selected so as to get data for which much better classification would be possible than for the complete phoneme data and selection of the basis functions using POLYCLASS would be computationally feasible. There are 14,736 cases in the corresponding training subsample and 10,167 cases in the test subsample. In the remainder of this article we refer to the complete data set as the all-phoneme data, and to the subset of the data consisting of all occurrences of three phonemes as the three-vowel data.

5.2 COMPARISON OF THE OPTIMIZATION METHODS

For both datasets we selected basis functions by applying the POLYMARS algorithm to the complete training set. We selected 100 basis functions for the three-vowel data and 1,000 bases functions for the all-phoneme data. For the three-vowel data we also selected 100 basis functions using the POLYCLASS algorithm. However, although the classification results using basis functions selected with POLYCLASS were better, this approach requires more than ten times as much CPU time as that required by the combined use of POLYMARS to select the basis functions and one of the faster optimization methods among those discussed in Sections 3 and 4 to obtain the maximum likelihood estimates of the corresponding coefficients.

We applied a variety of numerical optimization methods to determine the maximum likelihood estimate of the coefficient vector β corresponding to the basis functions selected by POLYMARS. In particular, we used a conjugate gradient method, a stochastic gradient method, and a stochastic conjugate gradient method. For the three-vowel data we also used a quasi-Newton method. On the all-phoneme data we could only apply the quasi-Newton method on problems with 100 or fewer basis functions since, for significantly more basis functions, the size of the Hessian matrix became too large to be handled by our Sun ULTRA II workstations with 64M of memory.

As mentioned in Section 4.1, the actual updating formulas used in the conjugate gradient and stochastic conjugate gradient methods appeared to have only a minor effect on performance. In addition, we found that a fairly rough line search typically sufficed. The one remaining parameter in the stochastic gradient method (initial learning rate r_0) and the stochastic conjugate gradient method (initial number $S^{(0)} = 2^{L_0}$ of batches) was optimized separately for each number of basis functions for POLYCLASS and for each number of hidden nodes for NEURALNET. The effect of the choice of $S^{(0)} = 2^{L_0}$ for the stochastic conjugate gradient algorithm is discussed in more detail in the next subsection.

In Figure 1 we show the (test set) performance of the various optimization methods on the three-vowel data with 25 and 100 basis functions as a function of CPU time. Because the various methods require considerably different amounts of computation, we felt that CPU time rather than the number of passes through the data was the appropriate quantity to compare. Note that we do not include swap time in any of our plots. (In particular, for the quasi-Newton algorithm the swap time can be substantial, since the pseudo-Hessians require a lot of storage space.) The horizontal axis in Figure 1 (CPU time) does not start at 0 since we included the time spent in selecting the POLYMARS basis functions—about .7 minutes for 25 basis functions and 5.5 minutes for 100 basis functions. The vertical axis in this plot shows the mean

$$\exp\left(\frac{1}{n_{\text{test}}}\sum_{i\in\text{testset}}\ell_i(\boldsymbol{\beta})\right)$$

of the exponential of the fitted log-likelihood over the test set. This quantity is the same as the geometric mean of the estimated probabilities of the correct class.

For the computations reported in this section we compared the amount of CPU time needed per pass through the data with the predicted number of flops in the previous

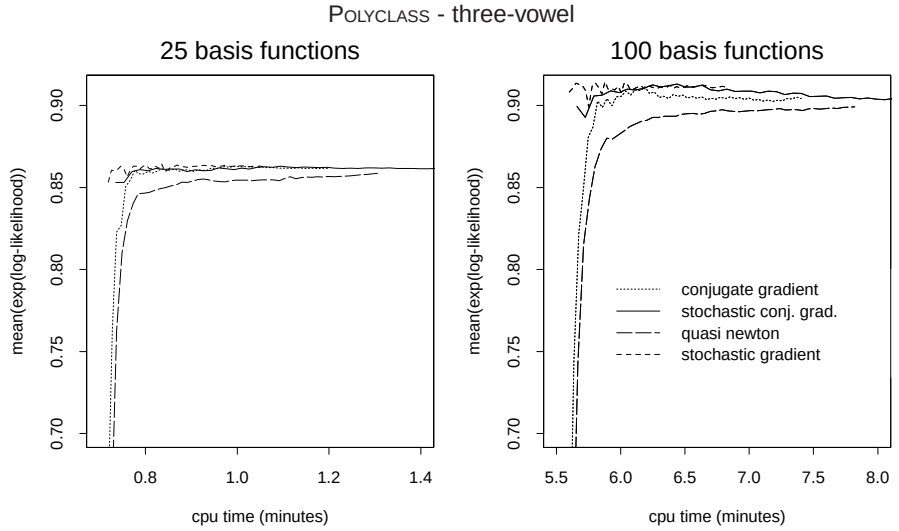

Figure 1. Performance of the various optimization methods for POLYCLASS on the three-vowel data.

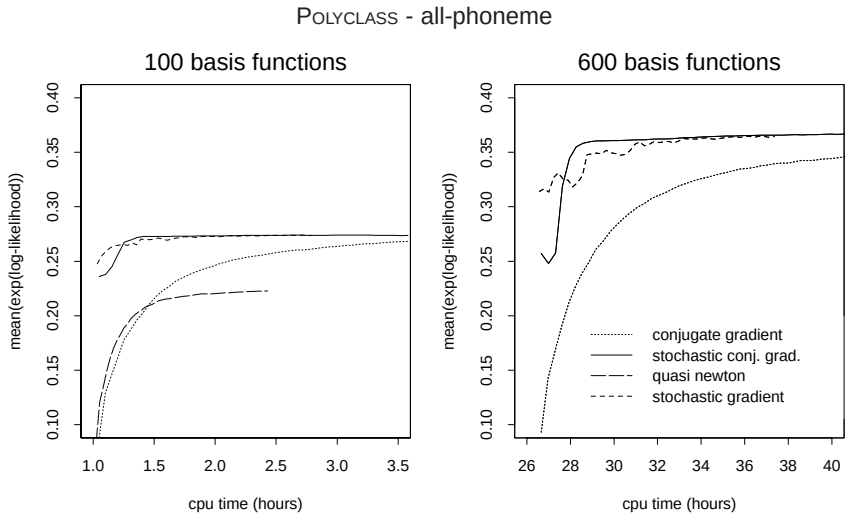

Figure 2. Performance of the various optimization methods for POLYCLASS on the all-phoneme data.

sections. We found a surprisingly good agreement between these predictions and the actual amount of cpu time needed.

We can see from this figure that the two stochastic methods outperform the two nonstochastic methods on the three-vowel data. In addition, the conjugate gradient method seems to be considerably faster than the quasi-Newton method. This conclusion also holds in the context of Figure 2, which summarizes the results of the various optimization methods on the all-phoneme data with 100 and 600 basis functions. From this figure it appears that although the stochastic gradient method initially performs slightly better than the stochastic conjugate gradient method, after a few iterations the stochastic conjugate gradient method takes over the lead.

The performance of the various optimization methods for NEURALNET is summarized in Figure 3 for the three-vowel data and in Figure 4 for the all-phoneme data for smaller and larger numbers of hidden nodes. In these two figures we see more extreme differences than in Figures 1 and 2 for POLYCLASS. We attribute the superiority of the stochastic conjugate gradient method over the stochastic gradient method in the fitting of NEURALNET models to the lack of concavity of the corresponding log-likelihood functions. This causes stochastic gradient steps to be more frequently in a poor direction than they were for POLYCLASS, and the corresponding estimate of β that is eventually obtained may be in a neighborhood of an inferior local maximum of the log-likelihood function. Moreover, since this function is not concave, computing the gradients exactly, as in the conjugate gradient method, is less helpful for NEURALNET than for POLYCLASS. We also noticed that both sized networks clearly overfit the three-vowel data; after about 30 iterations (one minute of CPU time for 5 hidden nodes, six minutes for 50 hidden nodes) the test-set log-likelihood starts to decrease, while the training-set log-likelihood still increases (data not shown).

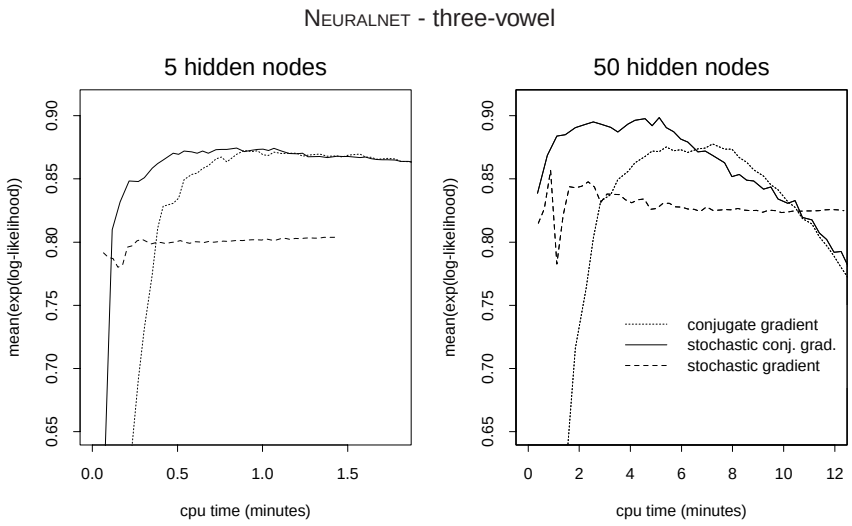

Figure 3. Performance of the various optimization methods for NEURALNET on the three-vowel data.

5.3 CHOICE OF THE INITIAL NUMBER OF BATCHES

From Figures 1 through 4, it appears that the most efficient optimization method for fitting POLYCLASS and NEURALNET models to the phoneme datasets is the stochastic conjugate gradient method. In our implementation this method has only one free parameter: the initial number $S^{(0)}$ of batches. In Figure 5 (for POLYCLASS) and Figure 6 (for NEURALNET) we show the performance of these optimization methods for a range of choices of $S^{(0)}$ and for a variety of model sizes. In these figures we show the performance after five passes through the data, but the relative performance as a function of

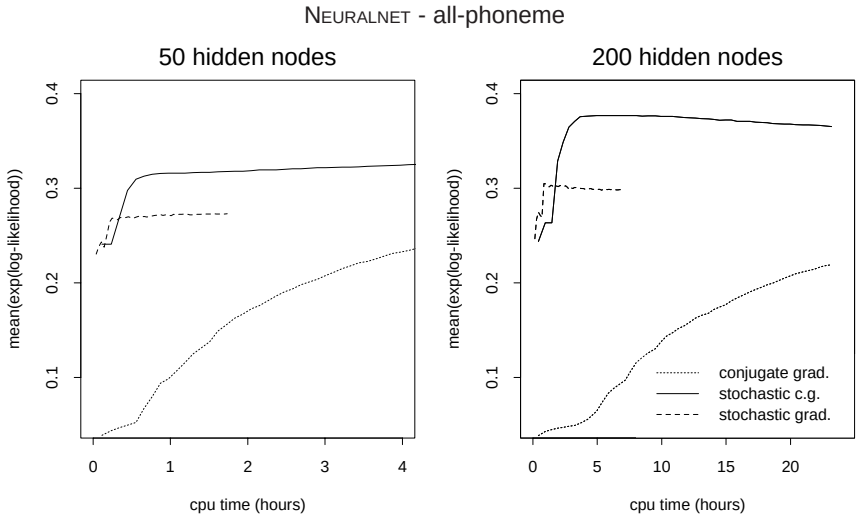

Figure 4. Performance of the various optimization methods for NEURALNET on the all-phoneme data.

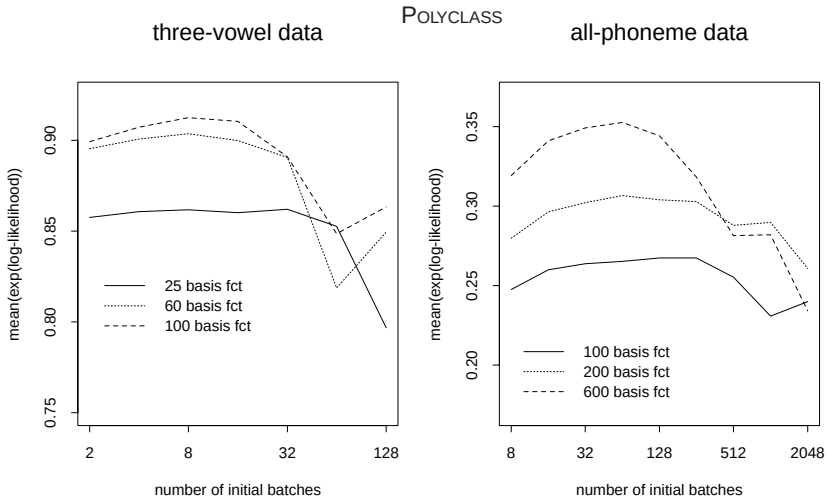

Figure 5. Performance of the stochastic conjugate gradient method for POLYCLASS as a function of the number of initial batches.

$S^{(0)}$ remains mostly unchanged for different numbers of passes. (The CPU time per pass does not depend significantly on $S^{(0)}$.)

As can be seen from Figure 5, for POLYCLASS the optimal value of $S^{(0)}$ does not depend significantly on the model size. Rather, for the three-vowel data $S^{(0)} = 8$ or $S^{(0)} = 16$ seems to be uniformly optimal, and for the all-phoneme data $S^{(0)} = 64$ seems optimal. Actually, these values of $S^{(0)}$ would already have been picked after the first pass through the data, suggesting that an automated algorithm could easily be implemented. It seems reasonable that $S^{(0)}$ should be larger for the larger problem with its much larger set of training data.

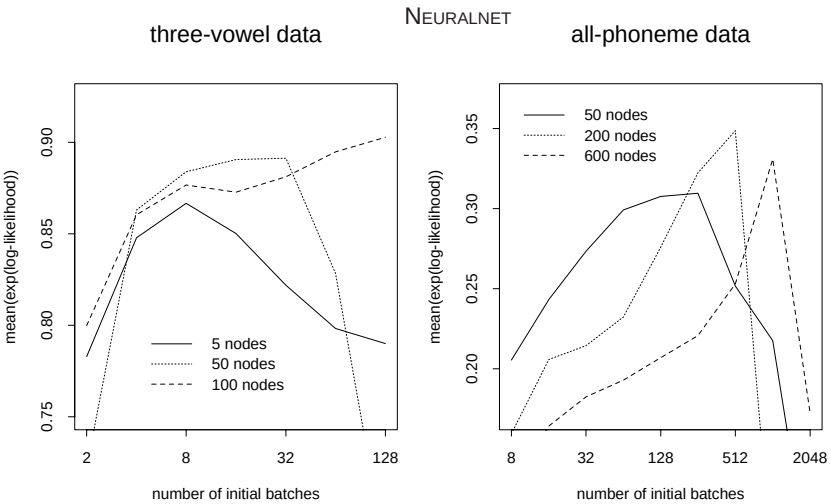

Figure 6. Performance of the stochastic conjugate gradient method for POLYCLASS as a function of the number of initial batches.

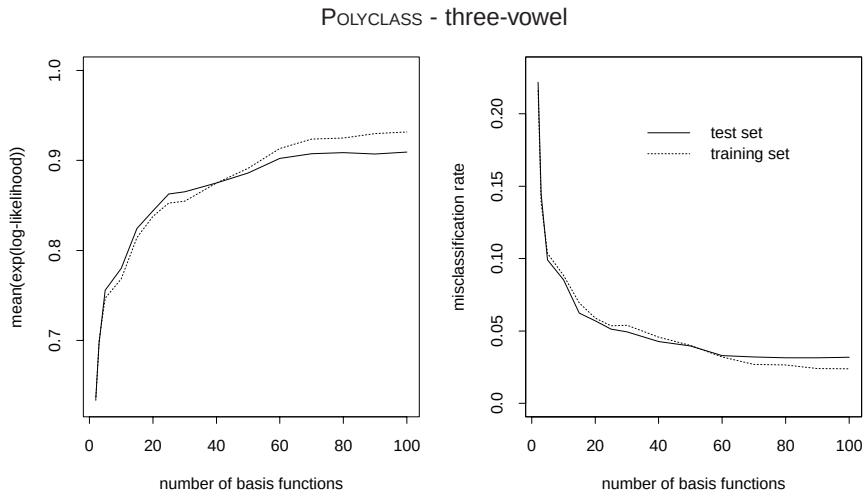

Figure 7. Performance of POLYCLASS on the three-vowel data.

Figure 6 suggests that for NEURALNET the best choice of $S^{(0)}$ does depend significantly on the number of hidden nodes. For example, for the all-phoneme data we notice that for 50 hidden nodes $S^{(0)} = 256$ is optimal, for 200 hidden nodes $S^{(0)} = 512$ is optimal, and for 600 hidden nodes $S^{(0)} = 1024$ is optimal. For the three-vowel data we notice a similar increase in the optimal number of initial batches, and hence a decrease in the initial batch size, when the number of hidden nodes is increased. Fortunately, for NEURALNET as well as for POLYCLASS, approximately the same values of $S^{(0)}$ would have been picked after the first pass through the data,

5.4 PERFORMANCE OF POLYCLASS AND NEURALNET

While the main goal of this article is to discuss various optimization methods for large problems, we looked briefly at the comparative performance of POLYCLASS and NEURALNET on the three-vowel and all-phoneme data sets. In Figures 7 and 8 we show the mean of the exponential of the log-likelihood and the misclassification rate for POLYCLASS as applied to the two datasets. For these plots we used the results after the 20th pass through the data of the stochastic conjugate gradient optimization. We note from these plots that even at the largest number of basis functions, the test-set performance still seems to be improving a bit, suggesting that still larger models may have a slightly better performance.

Figures 9 and 10 show graphs for NEURALNET similar to those in Figures 7 and 8 for POLYCLASS. Again, the results are shown for the stochastic conjugate gradient method at the end of 20 passes through the data. For NEURALNET this corresponds approximately to the best test set performance. For the three-vowel data, we notice that NEURALNET with about 25 hidden nodes roughly reaches the same performance as POLYCLASS reaches with 40–60 basis functions. However, POLYCLASS still improves when the model size is further increased, while NEURALNET seems only to overfit the data further, as the gap between

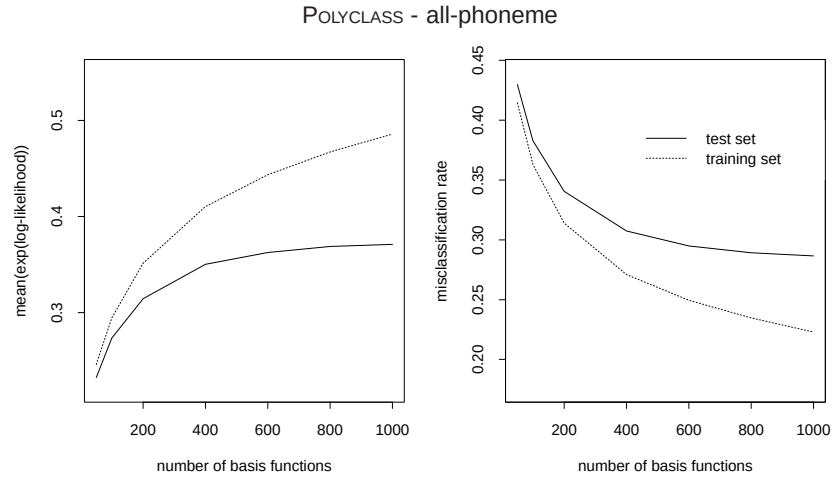

Figure 8. Performance of POLYCLASS on the all-phoneme data.

test set and training set performance increases. The best performance for NEURALNET on the all-phoneme data, reached with 400 hidden nodes, is not reached by POLYCLASS. If the number of nodes is further increased, the test set performance decreases considerably. (The training set performance for larger numbers of hidden nodes would still improve if we would let the optimization routine make more passes through the data.)

6. DISCUSSION

In this article we have considered a variety of numerical methods for approximating the maximum likelihood estimates of the unknown parameters in large POLYCLASS

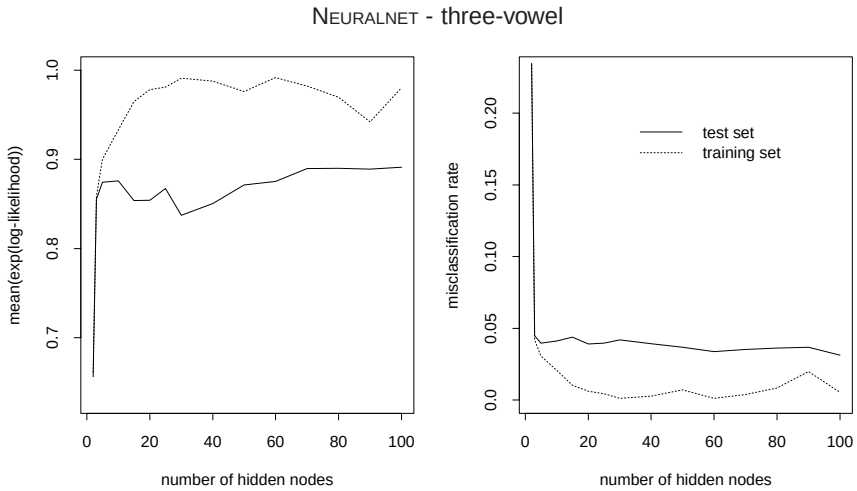

Figure 9. Performance of NEURALNET on the three-vowel data.

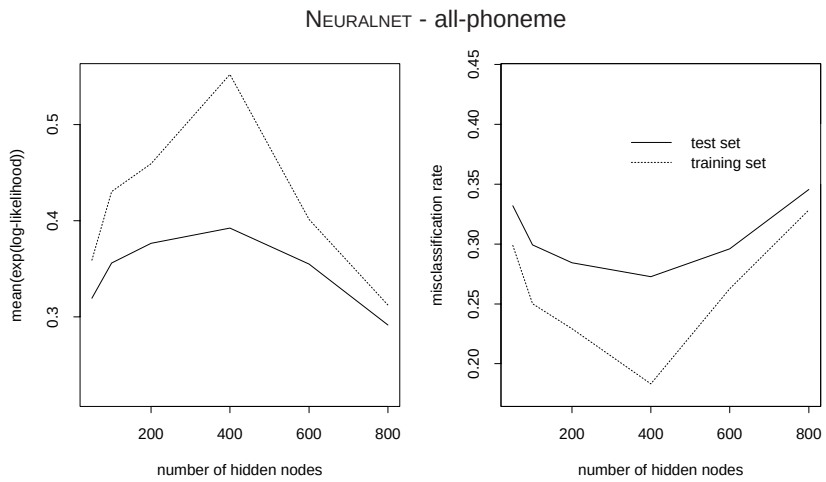

Figure 10. Performance of NEURALNET on the all-phoneme data.

and NEURALNET (feed-forward neural network having a single hidden layer) models. It turned out that stochastic gradient and stochastic conjugate gradient methods considerably outperformed the traditional, nonstochastic, quasi-Newton and conjugate gradient optimization methods. In particular, the stochastic methods found parameter estimates that nearly maximized the log-likelihood in much less CPU time. Stochastic optimization methods may not find the exact maximum likelihood estimate in a POLYCLASS or NEURALNET model, but this is not a serious deficiency in practice. Indeed, we have seen that the test set results may be better when the estimated parameters are close to the maximum likelihood estimates than when they are exactly equal to them.

The stochastic optimization methods performed better than the nonstochastic methods in all of our examples: for POLYCLASS and NEURALNET; for large and moderate numbers of basis functions and hidden nodes; and for the large all-phoneme dataset and the smaller three-vowel subset. We agree with Møller (1993) that the demonstrated superiority of the stochastic methods is due to the large amount of "redundancy" in large datasets: unless we are very close to the maximum likelihood estimate, we do not need to use the precise value of the gradient, so an acceptable Monte Carlo approximation can be obtained from a relatively small subset of the data.

In our implemention of the stochastic optimization methods, we were able to settle on relatively simple algorithms with just a single parameter, which could be optimized by letting the algorithm run for a short time for various choices of the parameter. There are obviously other reasonable choices of algorithms and corresponding parameters that could be considered. We are convinced that properly designed stochastic algorithms will consistently outperform nonstochastic algorithms in fitting models to (sufficiently) large datasets.

We realize that advocating stochastic optimization methods for numerically determining the maximum likelihood estimates of the unknown parameters in a POLYCLASS or NEURALNET model based on a large dataset is still controversial since a common opinion is in agreement with Ripley (1996), where the reader is advised to use a well tested,

expert implementation of a general purpose (nonstochastic) optimization algorithm. We suspect that such advice is based on experience with smaller datasets than those used in the present article.

The stochastic conjugate gradient method that we implemented performed much better than the stochastic gradient method in fitting NEURALNET models, but it performed only a little better in fitting POLYCLASS models. This varying relative performance may be related to the highly multimodal nature of the NEURALNET log-likelihood function in contrast to the strict concavity and hence unique maximum of the POLYCLASS log-likelihood function.

Møller (1993) gave a similar, but more complicated implementation and treatment of the stochastic conjugate gradient method in the context of fitting NEURALNET models. His article is reasonably well known by those in the neural network community, but is rarely cited in the literature. The authors hope that the simplifications introduced in this article, the extension of the method to the fitting of POLYCLASS models, and its success in the experiments we have described will increase the popularity of the stochastic gradient method.

Although it is computationally more challenging to fit large NEURALNET models than to fit large POLYCLASS models, suitably fit NEURALNET models performed as well as or a little better than POLYCLASS models in the context of the phoneme data that we used in our experiments. This surprised us somewhat, since the basis functions of POLYCLASS are chosen adaptively by POLYMARS. Nevertheless, we feel that there is definitely a place for POLYCLASS because such models are more insightful and may well perform better than NEURALNET models on other types of data, and their numerical and statistical properties are mathematically more tractable. In any case, the main goal of the present article has been to show that stochastic gradient and stochastic conjugate gradient methods are computationally feasible in the fitting of POLYCLASS models to much larger datasets than are deterministic optimization methods such as those treated in Kooperberg et al. (1997).

ACKNOWLEDGMENTS

The authors are grateful to a referee for comments that led to this revision. Charles Kooperberg was supported in part by National Science Foundation grant DMS-9403371 and National Institutes of Health grant CA74841. Charles J. Stone was supported in part by National Science Foundation grants DMS-9504463 and DMS-9802071.

[Received April 1996. Revised August 1998.]

REFERENCES

Bourlard, H. A., and Morgan, N. (1994), *Connectionist Speech Recognition*, Boston: Kluwer.

Cheng, B., and Titterington, D. M. (1994), "Neural Networks: A Review from a Statistical Perspective" (with discussion), *Statistical Science*, 9, 2–54.

Cole, R., Noel, M., Burnett, D. C., Fanty, M., Lander, T., Oshika, B., and Sutton, S. (1994), "Corpus Development Activities at the Center for Spoken Language Understanding," Technical Report, CSLU, Portland, OR.

Cole, R. A., Roginski, K., and Fanty, M. (1992), "A Telephone Speech Database of Spelled and Spoken Names," in *Proceedings of the International Conference on Spoken Language Processing*, Edmonton: Quality Color Press, University of Alberta, pp. 891–893.

Dixon, L. C. W. (1975), "Conjugate Gradient Algorithms: Quadratic Termination without Linear Searches," *Journal of the Institute of Mathematics and its Applications*, 15, 9–18.

Gauvain, J. L., Lamel, L., Adda, G., and Adda-Decker, M. (1994), "Speaker-Independent Continuous Speech Dictation," *Speech Communication*, 15, 21–37.

Hermansky, H. (1990), "Perceptual Linear Predictive (PLP) Analysis of Speech," *Journal of the Acoustical Society of America*, 87, 1738–1752.

Kennedy, W. J., and Gentle, J. E. (1980), *Statistical Computing*, New York: Marcel Dekker.

Kooperberg, C., Bose, S., and Stone, C. J. (1997), "Polychotomous Regression," *Journal of the American Statistical Association*, 92, 117–127.

Ljung, L. (1977), "Analysis of Recursive Stochastic Algorithms," *IEEE Transactions on Automatic Control*, 22, 551–575.

Møller, M. (1993), "Supervised Learning on Large Redundant Training Sets," *International Journal of Neural Systems*, 4, 15–25.

Polak, E., and Ribiere, G. (1969), "Note sur la Convergence de Methodes de Directions Conjugees," *Revue Francaise d'Informatique et de Recherche Operationnelle*, 3, 35–43.

Rabiner, L., and Juang, B.-H. (1993), *Fundamentals of Speech Recognition*, Englewood Cliffs, NJ: Prentice Hall.

Ripley, B. (1994), "Neural Networks and Related Methods for Classification," *Journal of the Royal Statistical Society*, Ser. B, 56, 409–456.

——— (1996), *Pattern Recognition and Neural Networks*, Cambridge: Cambridge University Press.

Robbins, H., and Monro, S. (1951), "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, 22, 400–407.

Rumelhart, D. E., and McClelland, J. L. (eds) (1986), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1. Foundations*, Cambridge, MA: MIT Press.

Stone, C. J., Hansen, M. H., Kooperberg, C., and Truong, Y. K. (1997), "Polynomial Splines and Their Tensor Products in Extended Linear Modeling" (with discussion), *The Annals of Statistics*, 25, 1371–1470.

White, H. (1989), "Some Asymptotic Results for Learning in Single Hidden-Layer Feedforward Networks," *Journal of the American Statistical Association*, 84, 1003–1013. Correction, 87, 1252.