Looking for lumps: boosting and bagging for density estimation

Greg Ridgeway
RAND Statistics Group
Santa Monica, CA 90407-2138
gregr@rand.org

Abstract

The solution to data mining problems often involves discovering non-linear relationships in large, noisy datasets. Bagging, boosting, and their variations have produced an interesting new class of techniques for finding these relationships in prediction problems. In this paper I extend these methods to the design of algorithms for density estimation for large, noisy, high dimensional datasets. Analogous to the boosting framework, the algorithms iteratively mix the current density estimator with an additional density chosen in a greedy fashion to optimize a fit criterion. A bagging step helps to control overfitting by providing better estimates of the fit criterion. I derive optimization algorithms for the boosting steps, discuss strategies for massive datasets, and show results from real and simulated problems.

Keywords: Density estimation, boosting, bagging, data mining

1 Introduction

This paper introduces a new algorithm for density estimation. Although that is the end product, I intend for this paper to show much more. Classification problems consume much of the focus from the researchers primarily associated with boosting and bagging methods. I will show that boosting and bagging are extremely flexible and that these methods provide a natural solution to the density estimation problem. Furthermore, they provide a transparent way to decrease bias and control variance.

In the remainder of this section I will briefly discuss heuristics of boosting and bagging. I will also discuss the density estimation problem. Section 2 will develop a boosting algorithm for density estimation that uses the EM algorithm and normal distributions as the base component of the estimator. I will also discuss a simple way to accelerate EM in this situation. Section 3 will demonstrate the algorithm on some test datasets.

1.1 Boosting and bagging

Computational learning theorists first started studying boosting methods in the context of classification problems. The AdaBoost algorithm (Freund and Schapire 1997) in particular brought boosting to the forefront of modern prediction methods. Early on this algorithm seemed to have an uncanny ability to produce some of the lowest misclassification rates on standard test datasets of any off-the-shelf classifier. Recent explanations of its performance characteristics have focused on boosting as a gradient descent algorithm that searches for the classifier that minimizes some loss function (Breiman 1999a, Friedman et al. 2000, Mason et al. 2000). Once understood as a familiar optimization problem, the idea behind boosting becomes applicable to a variety of other prediction problems including non-linear and robust non-linear regression (Friedman 1999) and non-linear exponential family and survival regression models (Ridgeway 1999). Boosting has a rigorous definition in computational learning theory. However, in this paper, although it might be a misuse of the term, I use "boosting" to describe gradient-based functional optimization algorithms that fit non-linear functions to data.

Functional gradient methods, like boosting, begin with a vague guess for the classifier, regressor, or density estimate. Subsequent iterations add or mix a new component that offers the largest local decrease in error (misclassification, squared-error loss, or entropy depending on the application). Generally these algorithms will overfit the training data if allowed to proceed unchecked. Each iteration progressively decreases prediction bias without monitoring prediction variance. The strategy has been to run the optimizer for a fixed number of iterations, selected via cross-validation, and stop before the increase in variance offsets the decrease in bias. A key ingredient for boosting to be successful is to use low-variance predictors on each iteration. Bagging (Breiman 1996a) reduces variance by combining several models, each fit to bootstrap samples of the dataset. Breiman (1999b) proposed combining boosting and bagging ideas together with the expectation that boosting's bias reduction together with bagging's variance reduction could produce excellent predictive models. He introduces a clever automatic procedure using out-of-bag observations for choosing when to halt the optimizer. In this paper I will adapt these ideas for use in density estimation.

1.2 Density estimation

Assume that we observe a dataset of n independent and identically distributed d-dimensional observations, $x_1, \ldots, x_n$ drawn from an unknown density $f(x)$. The problem is to produce an estimate, $\hat{f}(x)$, of $f(x)$ from the dataset alone. We can assess the quality of the estimate using the expected log-likelihood

$$J(\hat{f}) = \mathrm{E}_x \log \hat{f}(x) \tag{1}$$

where the expectation is taken with respect to the true density $f(x)$. The $\hat{f}(x)$ that maximizes $J(\hat{f})$ is indeed the true density. Since we do not know $f(x)$ we can

approximate $J(\hat{f})$ as

$$\hat{J}(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} \log \hat{f}(x_i) \tag{2}$$

The density that optimizes $\hat{J}(\hat{f})$ is one that puts point masses on the observed x_i. Although this maximizes $\hat{J}(\hat{f})$, the approximation breaks down as $\hat{f}(x)$ approaches such a point mass distribution since the variance of the empirical estimate of $J(\hat{f})$ becomes large. To prevent this we usually impose smoothness constraints. Much research in density estimation focuses on how to best smooth $\hat{f}(x)$.

Although we are dealing with density estimation, the problem is akin to previous applications of boosting. They share these common features.

- There is a functional measuring generalization error or model fit, J.
- We want to find a function that maximizes (or minimizes) the objective function, J.
- We cannot compute J explicitly but can only approximate it with our sample.

In fact, a large class of statistical problems matches this description. Boosting, or gradient style functional optimization, seems to offer a competitive solution in this scenario. In classification problems boosting sequentially mixes classifiers together, each additional classifier reducing misclassification error. In the next section I present a boosting algorithm for density estimation that sequentially mixes distributions together to maximize the expected log-likelihood.

2 A boosted density estimator using mixtures

In this section I present an algorithm for density estimation that approaches the problem as a functional optimization problem. That is, the algorithm involves gradient ascent style adjustments to a current guess for the density in order to optimize a likelihood.

As previously mentioned, a good density estimator has large expected likelihood. If we let $\hat{f}(x)$ be an initial guess for the density then we can increase $J(\hat{f})$ by mixing $\hat{f}(x)$ with some other density, $g(x)$, chosen from some class of distributions so that $J((1-\alpha)\hat{f} + \alpha g)$ is larger than $J(\hat{f})$. Two potentially ideal candidates for $g(x)$ are the multivariate uniform and normal densities. For a d-dimensional x, the uniform is compact since it requires $2d$ parameters. However, finding those parameters seems computationally challenging and I did not pursue that further. The normal density, $\varphi(x; \mu, \mathbf{\Sigma})$, requires more parameters, $2d + d(d-1)/2$, but estimating them is linear in the number of observations. Other candidates are possible but I will not consider these here.

Focusing now on the normal case, we can try to select μ, $\mathbf{\Sigma}$, and a mixing proportion parameter, α, so that

$$J((1-\alpha)\hat{f} + \alpha\varphi) = \mathrm{E}_x \log\left[(1-\alpha)\hat{f}(x) + \alpha\varphi(x; \mu, \mathbf{\Sigma})\right] \tag{3}$$

is larger than $J(\hat{f})$. Once those parameters have been found, we can update our density estimate as

$$\hat{f}(x) \leftarrow (1-\alpha)\hat{f}(x) + \alpha\varphi(x;\mu,\mathbf{\Sigma}). \tag{4}$$

By repeatedly finding $(\mu, \mathbf{\Sigma}, \alpha)$ and updating $\hat{f}(x)$ so that $J(\hat{f})$ increases, we move through the space of density functions on a path of densities approaching the true density.

The main difficulty, of course, is that in practice we cannot compute the expectation in (3). We can estimate (3) using our sample since

$$\begin{aligned} J((1-\alpha)\hat{f}+\alpha\varphi) &\approx \frac{1}{N}\sum_{i=1}^{N}\log\left[(1-\alpha)\hat{f}(x_i)+\alpha\varphi(x_i;\mu,\mathbf{\Sigma})\right] \\ &= \hat{J}((1-\alpha)\hat{f}+\alpha\varphi). \end{aligned} \tag{5}$$

We end up with an ill-posed optimization problem since we can only approximate the objective function. Furthermore, the $(\mu, \mathbf{\Sigma})$ that maximizes (5) is a point mass on any one of the x_i. With such values for the parameters, the approximation in (5) is poor.

Nevertheless, with our sample we can carefully estimate a sequence of $(\mu, \mathbf{\Sigma}, \alpha)$ constructing our path toward a good density estimate. Each step on this path will need to be cautious about the variance of $\hat{J}(\hat{f})$. We can then try to halt progress on this path when we can no longer be certain that a move that increases $\hat{J}(\hat{f})$ also increases $J(\hat{f})$.

2.1 Likelihood optimization using EM

The EM algorithm (Dempster et al. 1977) is one way to propose an update to the current guess, $\hat{f}(x)$, for the density estimate (Wasserman 2000). At a particular stage we can assume that each observation came from either $\hat{f}(x)$ with probability $1-\alpha$ or some normal distribution, $\varphi(x;\mu,\mathbf{\Sigma})$, with probability α. This is only a slight departure from the usual EM style of estimation for mixture models in that we fix $\hat{f}(x)$ and only need to estimate $(\mu, \mathbf{\Sigma}, \alpha)$. I will briefly derive the EM algorithm for this case.

Let z_i be the class indicator, 0 if $x_i \sim \hat{f}(x)$ and 1 if $x_i \sim \varphi(x;\mu,\mathbf{\Sigma})$. The E-step computes the expected value of the complete data log-likelihood.

$$\begin{aligned} Q(\mu,\mathbf{\Sigma},\alpha) &= \mathrm{E}\left[\sum_{i=1}^{N}(1-z_i)\log\hat{f}(x_i)+z_i\log\varphi(x_i;\mu,\mathbf{\Sigma})+\right. \\ &\qquad \left.(1-z_i)\log(1-\alpha)+z_i\log\alpha\,\middle|\,\mathbf{x},\mu,\mathbf{\Sigma},\alpha\right] \\ &= \sum_{i=1}^{N}(1-p_i)\log\hat{f}(x_i)+p_i\log\varphi(x_i;\mu,\mathbf{\Sigma})+ \\ &\qquad (1-p_i)\log(1-\alpha)+p_i\log\alpha \end{aligned} \tag{6}$$

where

$$p_i \;=\; P(z_i = 1 | x_i, \mu, \mathbf{\Sigma}, \alpha) = \frac{\alpha\varphi(x_i; \mu, \mathbf{\Sigma})}{(1-\alpha)\hat{f}(x_i) + \alpha\varphi(x_i; \mu, \mathbf{\Sigma})}$$

The M-step finds μ, $\mathbf{\Sigma}$, and α that maximize (6).

$$(\hat{\mu}, \hat{\mathbf{\Sigma}}) \;\leftarrow\; \arg\max_{\mu, \mathbf{\Sigma}} \sum_{i=1}^{N} p_i \log \varphi(x_i; \mu, \mathbf{\Sigma}) \tag{7}$$

$$\hat{\alpha} \;\leftarrow\; \arg\max_{\alpha} \sum_{i=1}^{N} (1-p_i)\log(1-\alpha) + p_i \log \alpha \tag{8}$$

Computation of the M-step produces $\hat{\mu}$ as the weighted mean of x_i and $\hat{\mathbf{\Sigma}}$ as the weighted covariance of x_i with weights p_i. The update for α is the average of the p_i's. Iterating between the E-step and the M-step yields a sequence of $(\mu, \mathbf{\Sigma}, \alpha)$ for which the log-likelihood does not decrease. Note, however, that early iterations of the EM algorithm might make for a poor modification to $\hat{f}(x)$. Subsequent iterations will make improvements with respect to the starting value so that at convergence the parameter values should offer a reasonable normal density to mix in with our current density estimate.

So given a current guess for the density, the EM algorithm proposes a move in the space of densities along a line segment connecting $\hat{f}(x)$ to some normal density that offers an increase in the log-likelihood. Figure 1 summarizes the boosting algorithm for density estimation. The EM algorithm moves quickly toward a region that increases the log-likelihood but can be painfully slow to actually reach a maximum.[1] With a little additional effort we can accelerate the EM algorithm in the neighborhood of the maximum.

2.2 Accelerating EM using Newton-Raphson

Each iteration of the EM algorithm described in figure 1 is fairly fast. It requires a single scan of the dataset to compute the updates for the model parameters. This algorithm, however, moves quickly to the rough location of the maximum but could spend many more, sometimes hundreds more, iterations moving to the maximum. While Newton-Raphson algorithms tend to be faster, they also require computing the gradient and Hessian of the observed data log likelihood function. Louis (1982) showed that the gradient is computable as the derivative of the expected complete data log-likelihood (6).

[1] Note that the true maximum puts a point mass on one of the x_i. The EM algorithm fortunately gets stuck in a more desirable mode. As the boosting algorithm progresses, the density estimate improves, and the likelihood surface tends to have only modes at $\mu = x_i$ and $\mathbf{\Sigma}$ for which $|\mathbf{\Sigma}|$ is small. At this point, the boosting algorithm simply needs to reject those proposals for which $|\mathbf{\Sigma}|$ is too small.

Initialize $\hat{f}(x) = \varphi(x; \hat{\mu}_1, \hat{\mathbf{\Sigma}}_1)$ where $\hat{\mu}_1$ and $\hat{\mathbf{\Sigma}}_1$ are the mean and covariance of the sample. I found that inflating the covariance on the first round by a factor of four or so improved the algorithm.

While stopping criterion is not satisfied
{
  $\alpha = \frac{1}{2}, \mu =$ randomly selected x_i, $\mathbf{\Sigma} =$ sample covariance of the x_i's
  Iterate the EM algorithm
  {
    $p_i = \frac{\alpha\varphi(x_i;\mu,\mathbf{\Sigma})}{(1-\alpha)f(x_i)+\alpha\varphi(x_i;\mu,\mathbf{\Sigma})}$
    $\mu =$ weighted mean of the x_i's with weights p_i
    $\mathbf{\Sigma} =$ weighted covariance of the x_i's with weights p_i
    $\alpha =$ mean of the p_i's
  }
  Update the density estimate as $\hat{f}(x) \leftarrow (1-\alpha)\hat{f}(x) + \alpha\varphi(x_i; \mu, \mathbf{\Sigma})$.
}

Figure 1: *Boosting algorithm for density estimation*

To ensure that the Newton updates always propose valid parameters I reparametrize α and $\mathbf{\Sigma}$ as

$$\alpha = \frac{1}{1+e^{-\gamma}} \text{ and } \mathbf{\Sigma} = (\mathbf{A}\mathbf{A}^t)^{-1}$$

where $\mathbf{A}$ is a lower triangular matrix, the Cholesky decomposition of the precision matrix. Letting α be the logistic transform of γ forces α to be in $[0, 1]$ and defining $\mathbf{\Sigma}$ in terms of $\mathbf{A}$ forces $\mathbf{\Sigma}$ to be positive definite. The gradient, $\mathbf{g}$, of the observed data log-likelihood is

$$\frac{d}{d\mu}Q(\mu, \mathbf{A}, \gamma) = \sum_{i=1}^{N} p_i \mathbf{A}\mathbf{A}^{\mathbf{t}}(x_i - \mu) = \mathbf{\Sigma}^{-1}\sum_{i=1}^{N} p_i(x_i - \mu) \quad (9)$$

$$\frac{d}{dA}Q(\mu, \mathbf{A}, \gamma) = \sum_{i=1}^{N} p_i \left[\text{diag}(\mathbf{A})^{-1} - \text{LT}\left((x_i - \mu)(x_i - \mu)^t\mathbf{A}\right)\right] \quad (10)$$

$$\frac{d}{d\gamma}Q(\mu, \mathbf{A}, \gamma) = \sum_{i=1}^{N} p_i - \frac{1}{1+e^{-\gamma}} = N(\bar{p} - \alpha) \quad (11)$$

where LT($\mathbf{M}$) indicates the lower triangle of $\mathbf{M}$. The gradient calculation occurs just after computing the p_i's, the E-step.

Meilijson (1989) proposed using the empirical Fisher information to estimate the Hessian, $\mathbf{H}$. That is, since the score function, $\frac{d}{d\theta}Q$, is a sum of N independent

terms we can simply estimate the variance of the score function using the empirical covariance of the N observed values of the score functions. In high dimensions the Hessian can become large growing quadratically with d. It requires only a single scan of the dataset and if we can manage to compute it to perform a few Newton-Raphson steps we can avoid the last slow stages of the EM algorithm. With the gradient computed and the Hessian approximated we can update our parameters as $\theta \leftarrow \theta - \mathbf{H}^{-1}\mathbf{g}$. This replaces the parameter updates in figure 1.

Jamshidian and Jennrich (1997) propose an EM style conjugate gradient ascent algorithm that also could show promise here. Conveniently, it avoids the Hessian calculation but replaces it with an extra line search.

2.3 Using bagging to automate the stopping decision

So far I have presented a method for iteratively adding mixture components to the current density estimate. As mentioned in the introduction, boosting methods, if allowed to proceed unchecked, will eventually overfit the data. In the density estimation case this would lead to a mixture of normals tightly peaked around each data point. The number of iterations (equivalently the number of mixture components) needs to be fixed so that the variance of the estimator does not overcome the gains in bias reduction. I propose a variant of bagging that reduces variance while also offering an automatic criterion for stopping the algorithm.

The original implementation of bagging proposed by Breiman (1996a) creates bootstrap replicates of the dataset, fits a model to each, and then averages across the models. A slightly simpler implementation fits models to half-samples rather than bootstrap datasets (Friedman and Hall 1999). The result is a "bagged" model that tends to have equal bias but lower variance than a single model.

Incorporating this strategy into the present problem, on each iteration of the boosting algorithm I will take one simple random sample of $N/2$ observations from the dataset. The EM algorithm will search for the best normal distribution to add to the mixture based only upon this half of the dataset. In practice this improves the model fit with the added benefit of reducing the number of observations sent to the EM algorithm.

Using only half of the observations for proposing the new adjustments leaves the other half for providing a nearly unbiased estimate of the likelihood gain. We can use these "out-of-bag" observations (Breiman 1996b) to compute an nearly unbiased estimate of $J((1-\alpha)\hat{f} + \alpha\varphi) - J(\hat{f})$, the *expected* improvement in the log likelihood of the density estimate. After running the EM algorithm to convergence we compute ΔJ as

$$\begin{aligned}\Delta J &= \sum_{i \in \text{out-of-bag}} \log\Big((1-\alpha)\hat{f}(x_i) + \alpha\varphi(x_i;\mu,\mathbf{\Sigma})\Big) - \log\Big(\hat{f}(x_i)\Big) \\ &= \sum_{i \in \text{out-of-bag}} \log\left((1-\alpha) + \alpha\frac{\varphi(x_i;\mu,\mathbf{\Sigma})}{\hat{f}(x_i)}\right) \qquad (12)\end{aligned}$$

Note that in (12) if the proposed normal puts a lot of mass in a small neighborhood then ΔJ will be negative, unless there are out-of-bag observations to support it. When ΔJ is negative I do not mix the proposed normal into the density estimate. This prevents "spiked" density estimates unless the data truly warrant it. Generally, as components accumulate in the mixture ΔJ becomes smaller and at times becomes negative. The quality of the estimate is not very sensitive to moderate changes in the number of iterations. I stop the boosting process when ΔJ is negative for several iterations in a row (*e.g.* 3 or 4). A few additional iterations are unlikely to affect the density estimator but I also do not wish to waste computation time.

Note that parts of this algorithm are stochastic. In particular I have elected to randomly choose an x_i as the starting value for μ in the EM iterations. The bagging steps further randomize the algorithm. The implication is that repeat runs of the algorithm will produce different density estimates.

The components are in place to put together a density estimator that sequentially reduces bias while controlling variance with an automatic stopping criterion. The next section demonstrates the algorithm on some simulated and real datasets.

3 Examples

In this section I apply the boosted density estimator to some real and simulated datasets. For high-dimensional problems there is no readily available density estimator. Therefore, for the simulated datasets, I compare the boosted estimator to density estimates based on knowing the structure of the true density.

3.1 Old faithful data

Scott (1992) presents data on the duration of 107 eruptions of the Old Faithful geyser. This dataset has become an often used example for density estimation. It is a small dataset and the algorithm presented in this paper is overkill for such a problem. However, for visual comparison and to stay in line with tradition I will assess the boosted density estimate for the Old Faithful dataset. I estimated the expected log-likelihood (using leave-one-out cross validation) when using a kernel method and when using the boosted density estimator. A kernel density estimator with the width selected using unbiased cross-validation (as implemented in S-plus) yielded an expected log-likelihood of -1.01. For the boosted density estimator, using all the defaults discussed in the previous section, the estimate of the expected log-likelihood was -1.07. Figure 2 shows a graphical comparison of the two density estimates.

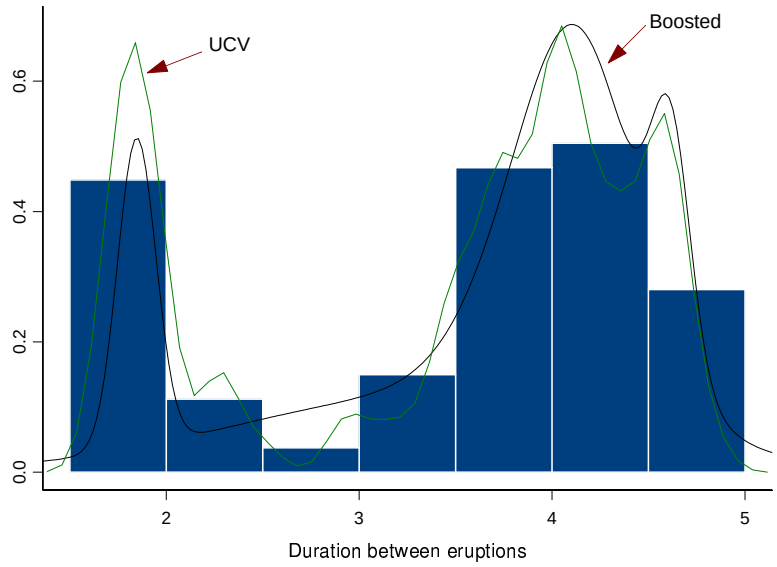

Figure 2: *Old faithful data. The density estimate labeled "UCV" is a kernel density estimate with a bandwidth selected using unbiased cross-validation. The boosted density estimate is a mixture of five normals.*

3.2 20 dimensional mixture distributions

I simulated 10 datasets each with 100,000 observations from a different 20 dimensional mixture distribution with five equally weighted normal components. Each component had roughly the same location but with varying shape. For each dataset I obtained a boosted density estimate. I will report all results on independent test datasets with 100,000 observations. I evaluated the average log-likelihood for the boosted density estimate and a density estimate based on knowing that the true density was a mixture of five normals. We cannot expect the boosted estimator to compete with more complete knowledge but we should hope that the performance is close.

When we assume that we know that the true density is a mixture of five 20 dimensional multivariate normal densities we can use the standard EM algorithm to estimate the components. Since the models are non-nested we cannot compare it to the boosted density estimator using simple procedures (the log likelihood ratio does not have a χ^2 distribution). At this point I can only present figures that seem to indicate that the model fit is adequate. The average log-likelihood when knowing the structure of the true density is -88.40 compared with -89.01 for the boosted density estimate. This says that for the average observation the boosted density estimate will be about 0.7% worse in terms of log-likelihood than if we had known the true structure.

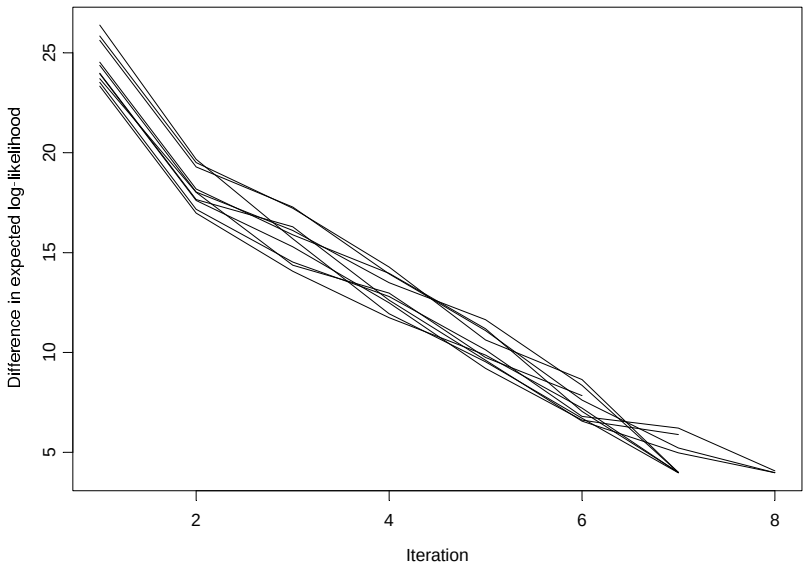

Figure 3: *20 dimensional uniform mixture example. The plot shows the decrease in the difference between the expected log-likelihood with the true density and the density estimate as the algorithm iterates. The 10 curves represent the 10 replicates of the experiment.*

The mixture of normals is somewhat well suited to this implementation of the boosted density estimate (one composed of normals). I repeated the above experiment on a 20 dimensional mixture of five uniforms. I randomly chose the bounds on each axis constrained to be between 0 and 1. The expected log likelihood under the true density was approximately 28.5 while the expected log-likelihood under the density estimate was, on average, about 4.6 lower. Figure 3 shows the difference between the expected log-likelihood under the true density and under the density estimate as the algorithm progressively added components to the mixture. On the 10 trials the algorithm added between six and eight components to the mixture. Figure 4 shows a two dimensional slice of one of the datasets and the associated density estimate. Upon termination the density estimate has captured the majority of the mass. However, it seems difficult to capture the corners when using a normal basis.

3.3 Classification problems

Density estimation can also play a role in constructing classifiers. Assume that Y is a discrete valued random variable taking on k possible states. The probability that Y takes on a particular state may depend on a feature vector X. Straightforward

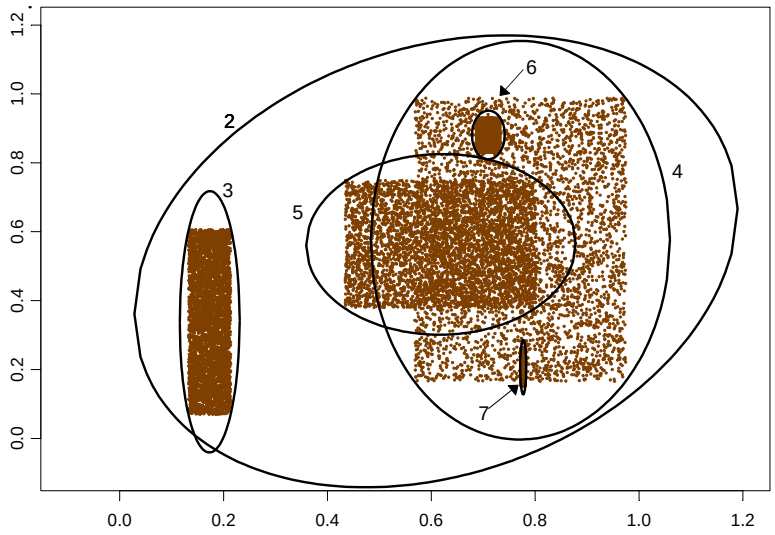

Figure 4: *20 dimensional uniform mixture example. A two dimensional slice of the data and the density estimate (95% contours). Each contour is labeled by the order it entered into the mixture. The first component's contour is outside the graphic.*

probability manipulations show that

$$P(Y = y|X) \propto P(X|Y = y)P(Y = y). \tag{13}$$

We can estimate $P(Y = y)$ by the prevalence of class y in the training dataset but $P(X|Y = y)$ requires some form of density estimate. Other classification methods based on boosting and bagging generally produce some of the best misclassification rates. These methods focus directly on estimating $P(Y|X)$ or the decision boundary and almost certainly outperform those that require an intermediate density estimation step. Friedman (1997) notes that "good probability estimates are not necessary for good classification" nor do low classification rates indicate accurate probability estimates. However, I include this section to argue that if boosted estimates of $P(X|Y = y)$ produce decent classifiers then there is some additional evidence that the boosted density estimator might be producing reasonable density estimates.

Blackard and Dean (2000) discuss predicting the main forest cover type in a region from cartographic variables. The dataset, which is available from the UCI KDD archive (Bay 1999), contains 581,012 observations. Each observation has 10 continuous features describing its location, shade, and distance to water and recent fire points. In addition we know the region (one of four undisturbed wilderness reserves) in which it resides and the soil type (one of 40 soil types). The goal is to predict the forest cover type (seven classes) from this information.

Algorithm	Misclassification
Neural net	30%
LDA	42%
Boosted density estimate	34%
CART	33%
Bagged CART	26%

Table 1: *Comparison of misclassification rates on the forest cover type dataset*

Let Y be the forest cover type class, X the vector of continuous features, and A and S refer to the area and the soil type respectively. I assume the following decomposition of the conditional class probabilities.

$$P(Y = y|X, A, S) \propto P(X|Y = y)P(A, S|Y = y)P(Y = y) \tag{14}$$

This assumes that, given the forest cover type, the area and soil type are independent of the cartographic measures. This is a mildly naïve assumption particularly since measures such as elevation and wilderness area are not entirely independent given that the forest cover type is, say, aspen. However, in classification problems these kinds of naïve assumptions are robust to violations and often perform well nonetheless.

Blackard and Dean (2000) used only a small portion of the dataset for training, 15,120 observations, since they were building a neural network classifier. They used the remainder as a validation dataset. Their neural network classifier achieved 30% misclassification compared with a 42% misclassification rate for linear discriminant analysis (LDA). It is unclear how they handled the discrete area and soil type features.

I used half of the dataset for training and the other half for validation. Density estimation in 10 dimensions does require a lot of observations. Table 1 summarizes the misclassification rates on this dataset for various procedures. The classifier based on the boosted density estimate performs reasonably well misclassifying 34% of the observations. It performs worse than other methods such as CART and bagged CART but its performance is still in the neighborhood.

4 Discussion

The algorithm decomposes into several stages, each of which is adaptable to massive dataset applications. I suggested sending half-samples to the EM algorithm to propose the direction in which to move the density estimate preserving the remaining half for an out-of-bag estimate of fit. For large datasets sending less than half to the EM algorithm should not degrade the performance. Research on "data squashing" (DuMouchel et al. 1999, Madigan et al. 2000) has shown that datasets can often be stratified into smaller, pseudo-datasets with observations weighted in

such a way that the likelihood for the pseudo-dataset approximates the likelihood for the massive dataset. Since the normal move that the EM algorithm proposes only needs to be approximate, an upfront data compression step could reduce the scale of the problem without degrading performance. Lastly, there is not really a need to iterate EM to convergence since we really only need a move that increases the expected log-likelihood.

I am aware of few implementations of high-dimensional density estimators. As far as other methods that are similar to the one presented here, Kloppenburg and Tavan (1997) take an annealing approach for finding a mixture of normals that maximizes the log-likelihood. Li and Barron (2000) provide a theoretical analysis of an "iterative likelihood maximization" density estimation method. As in this paper, they propose constructing density estimates by greedily mixing in additional model components and provide an assortment of interesting risk bounds for their estimators. Also Nichol et al. (2000) describe fitting mixtures of normals to astronomical datasets. They include a search for the optimal number of mixture components and utilize KD-trees for fast computation of the EM algorithm. They apply their method to simulated two-dimensional datasets with 100,000 observations and a real dataset with 11,000 galaxies. Polonik (1995) connects empirical process theory with a potential method for high-dimensional density estimation but stops short of proposing an algorithm.

This paper presented an easily implemented algorithm for boosted density estimation. By sequentially mixing normal densities, the algorithm can construct high dimensional density estimates. Utilizing the out-of-bag estimate of the increase in the expected likelihood, the algorithm has a completely automated stopping rule. Therefore, the algorithm involves no tuning parameters. Future work will speed up the EM proposal step and will consider methods for composing even more flexible density estimators by allowing more normal components into the mixture.

Simple but slow S code for `density.boost()` is available from the author.

Acknowledgements

Funding for this work came from a grant from the National Science Foundation (DMS 9704573) and from the RAND Corporation. The author is grateful to Werner Stuetzle for discussions on boosting with respect to density estimation, Larry Wasserman for his suggestion to use EM to make the modification proposals, and Dan McCaffrey for reviewing a draft of this paper.

References

Bay, S. (1999). The UCI KDD archive. http://kdd.ics.uci.edu. Irvine, CA: University of California, Department of Information and Computer Science.

Blackard, J. and D. Dean (2000). Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from carto-

graphic variables. *Computers and Electronics in Agriculture 24*(3), 131–151.

Breiman, L. (1996a). Bagging predictors. *Machine Learning 26*, 123–140.

Breiman, L. (1996b, November). Out-of-bag estimation. Technical report, University of California, Berkeley, Statistics Department.

Breiman, L. (1999a, October). Prediction games and arcing algorithms. *Neural computation 11*(7), 1493–1517.

Breiman, L. (1999b, February). Using adaptive bagging to debias regressions. Technical Report 547, University of California, Berkeley, Statistics Department.

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

DuMouchel, W., C. Volinsky, T. Johnson, C. Cortes, and D. Pregibon (1999). Squashing flat files flatter. In S. Chaudhuri and D. Madigan (Eds.), *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 6–15.

Freund, Y. and R. Schapire (1997). A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences 55*(1), 119–139.

Friedman, J. (1997). Bias, variance, 0/1 loss, and the curse of dimensionality. *Data Mining and Knowledge Discovery 1*(1), 55–77.

Friedman, J. (1999, February). Greedy function approximation: A gradient boosting machine. Technical report, Stanford University, Statistics Department. Available from http://www-stat.stanford.edu/∼jhf.

Friedman, J. and P. Hall (1999, May). On bagging and nonlinear estimation. Technical report, Department of Statistics, Stanford University.

Friedman, J., T. Hastie, and R. Tibshirani (2000). Additive logistic regression: a statistical view of boosting (with discussion). *Annals of Statistics 28*(2), 337–374.

Jamshidian, M. and R. I. Jennrich (1997). Acceleration of the EM algorithm by using quasi-Newton methods. *Journal of the Royal Statistical Society, Series B, Methodological 59*, 569–587.

Kloppenburg, M. and P. Tavan (1997, March). Deterministic annealing for density estimation by multivariate normal mixtures. *Physical Review E 55*(3), R2089–R2092.

Li, J. and A. Barron (2000). Mixture density estimation. In S. Solla, T. Leen, and K. Müller (Eds.), *Advances in Neural Information Processing Systems*, Volume 12. MIT Press.

Louis, T. A. (1982). Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society, Series B, Methodological 44*, 226–233.

Madigan, D., N. Raghavan, W. DuMouchel, M. Nason, C. Posse, and G. Ridgeway (2000). Likelihood-based data squashing: A modeling approach to instance construction. In H. Liu and H. Motoda (Eds.), *Instance Selection and Construction - A data mining perspective*, Chapter 12. Kluwer Academic Publishers.

Mason, L., J. Baxter, P. Bartlett, and M. Frean (2000). Boosting algorithms as gradient descent. In S. Solla, T. Leen, and K. Müller (Eds.), *Advances in Neural Information Processing Systems*, Volume 12. MIT Press.

Meilijson, I. (1989). A fast improvement to the EM algorithm on its own terms. *Journal of the Royal Statistical Society, Series B, Methodological 51*, 127–138.

Nichol, R., A. Connolly, A. Moore, J. Schneider, C. Genovese, and L. Wasserman (2000). Fast algorithms and efficient statistics: Density estimation in large astronomical datasets. Technical Report 719, Department of Statistics, Carnegie-Mellon University.

Polonik, W. (1995). Measuring mass concentrations and estimating density contour clusters – An excess mass approach. *The Annals of Statistics 23*, 855–881.

Ridgeway, G. (1999). The state of boosting. *Computing Science and Statistics 31*, 172–181.

Scott, D. (1992). *Multivariate Density Estimation: Theory, Practice and Visualization*. Wiley.

Wasserman, L. (2000). Personal communication.