Likelihood–based Boosting

Radford M. Neal
Depts. of Statistics and Computer Science, Univ. of Toronto,
M5S 1A4, Canada. radford@stat.toronto.edu

David J.C. MacKay
Cavendish Laboratory, Cambridge, CB3 0HE,
United Kindom. mackay@mrao.cam.ac.uk

November 23, 1998

Abstract

We present a probabilistic interpretation of 'boosting' in terms of a mixture of experts model, and give a boosting algorithm that is a crippled maximum likelihood algorithm.

Boosting is a method for training multiple classifiers on a dataset and combining their outputs to make predictions for new cases (Schapire, 1990; Freund and Schapire, 1995; Freund and Schapire, 1996; Freund and Schapire, 1997; Schapire and Singer, 1998). In this paper we present a probabilistic interpretation of 'boosting' in terms of a simple mixture of experts model, and show that this model has a likelihood–increasing algorithm which looks a lot like 'adaboost'.

1 The likelihood boosting algorithm

It is assumed that we have a probabilistic model whose prediction for a categorical variable t given an input $\mathbf{x}$ and given the parameters $\theta \equiv (\pi, \{\mathbf{w}\})$ is:

$$P(t|\mathbf{x}, \pi, \{\mathbf{w}\}) = \sum_{c=1}^{C} \pi_c P(t|\mathbf{x}, \mathbf{w}^{(c)}) \tag{1}$$

where each of the *component* models has a probability distribution $P(t|\mathbf{x}, \mathbf{w}^{(c)})$ which is simple to learn given a bunch of labelled data $\{\mathbf{x}^{(n)}, t^{(n)}\}$. We assume access to a learning algorithm which returns the maximum likelihood parameters $\mathbf{w}$ in response to a weighted data set with weights $r^{(n)}$. This model is known as a mixture of experts model, with fixed mixing coefficients $\{\pi_c\}$. In a general hierarchical mixture of experts, the mixing coefficients would be parametric functions of the input $\mathbf{x}$.

An example of a component model is:

$$P(t|\mathbf{x}, \mathbf{w}^{(c)}) = \begin{cases} 0.9 & \mathbf{w}^{(c)} \cdot \mathbf{x} \geq 0 \\ 0.1 & \mathbf{w}^{(c)} \cdot \mathbf{x} < 0 \end{cases}, \tag{2}$$

which could be called a noisy threshold function. Another example is the standard logistic classifier:

$$P(t|\mathbf{x}, \mathbf{w}^{(c)}) = \frac{1}{1 + \exp\left(-\mathbf{w}^{(c)} \cdot \mathbf{x}\right)}. \tag{3}$$

1.1 What this model is not

This mixture of experts model combines together C sigmoid functions, but it is not the same as the neural network model studied by Friedman *et al.* (1998). Contrast the model we are studying,

$$P(t|\mathbf{x}, \pi, \{\mathbf{w}\}) = \sum_{c=1}^{C} \pi_c f(\mathbf{w}^{(c)} \cdot \mathbf{x}), \tag{4}$$

where $f(a) = 1/(1 + e^{-a})$, with their

$$\log \frac{P(t = 1|\mathbf{x}, \pi, \{\mathbf{w}\})}{P(t = 0|\mathbf{x}, \pi, \{\mathbf{w}\})} = \sum_{c=1}^{C} f_c(\mathbf{x}), \tag{5}$$

or equivalently,

$$P(t|\mathbf{x}, \pi, \{\mathbf{w}\}) = f\left(\sum_{c=1}^{C} f_c(\mathbf{x})\right), \tag{6}$$

where f_c is a nonlinear function computed by component c.

We believe that the first of these two models has more in common with boosting than the second, as illustrated by the learning algorithm that follows.

1.2 Initialization

We initialize $\pi = (1, \epsilon, \epsilon \ldots \epsilon)$, where ϵ is a tiny number, and all classifiers' parameters to $\mathbf{w} = 0$ (or whatever value of $\mathbf{w}$ corresponds to a classifier that is insensitive to the value of $\mathbf{x}$).

We denote all the parameters $\{\pi, \mathbf{w}\}$ by θ.

1.3 E–step

We compute the responsibilities r_{cn} of each expert c for each data point n.

$$r_{cn} \equiv P(c|\mathbf{x}^{(n)}, t^{(n)}, \theta) = \frac{P(t^{(n)}|\mathbf{x}^{(n)}, \mathbf{w}^{(c)})\pi_c}{P(t^{(n)}|\mathbf{x}^{(n)}, \pi, \{\mathbf{w}\})} \tag{7}$$

At the first iteration, we have initialized π such that only the first expert is responsible. We have $r_{1n} = 1$ for all n, and $r_{cn} \propto \epsilon$ for all c not equal to 1.

1.4 Partial M–step

We update the parameters $\mathbf{w}^{(1)}$ of expert 1 only. We do this using the optimization algorithm for $\mathbf{w}$, which is fed with the entire data set $\{t^{(n)}, \mathbf{x}^{(n)}\}$, weighted by the responsibilities $\{r_{cn}\}$ for $c = 1$, which at this stage are all equal. The optimizer returns optimized parameters $\mathbf{w}^*$ that might satisfy, for example,

$$\mathbf{w}^* = \operatorname{argmax} \sum_n r_{cn} \log P(t^{(n)}|\mathbf{x}^{(n)}, \mathbf{w}^{(c)}). \tag{8}$$

1.5 E–step

We now reevaluate the responsibilities r_{cn}. Expert 1 still has overwhelming responsibilities for all points still, but that does not matter. The reponsiblities of all the other experts for the nth data point will be equal, but those responsibilities r_{cn} will vary from data point to data point. A data point $(t^{(n)}, \mathbf{x}^{(n)})$ that was well modelled by the optimized expert number 1 will have a smaller r_{2n} than a point that was poorly modelled by expert number 1.

1.6 Partial M–steps

We now adjust the parameters of expert 2 only, using the responsibilities $\{r_{2n}\}$ as weights for the optimizer.

(All this seems very similar to the reweighting of data made by boosting algorithms.)

We also update the prior probabilities of the two experts π_1 and π_2. We constrain the probabilities $\pi_3 \ldots$ to still equal ϵ. The details of this π update are flexible. It would probably be best to perform a sequence of E and M steps in which π is updated until it converges. (Otherwise the prior probability of expert 2 would remain of order ϵ.) The cost of this π optimization would be small because, as long as the parameters $\mathbf{w}$ are unchanged, few computations would be required to update the responsibilities.

1.7 And so forth

Having refreshed the responsibilities, we can wake up another expert. In order to make this algorithm most similar to boosting, we would only allow the new expert's parameters $\mathbf{w}$ and the weights π to be adjusted.

1.8 Properties

This algorithm is a crippled maximum likelihood algorithm. The likelihood increases with every step, but we don't expect to reach the optimum setting of the parameters unless we can tweak all the parameter vectors $\mathbf{w}$.

The model itself is not capable of overfitting in a likelihood sense, at least for the mixture of experts models given earlier. What we mean by this is that given a generic data set, it is not possible, given a huge number of experts C, to fit that data arbitrarily closely, with the likelihood $P(\{t\}|\theta)$ going arbitrarily close to 1. While it is possible to make a very complicated decision boundary (the surface in $\mathbf{x}$ space on which $P(t|\mathbf{x}, \theta)$ is 0.5), such a predictive function could only be made by mixing together the predictions of many mutually contradictory experts, so the predictive distribution would become closer to 0.5 in most places, the more wiggly the decision boundary becomes.

2 Discussion

Any model that is implemented with a boosting algorithm could also be implemented as a mixture of experts optimized with maximum likelihood.

Several questions can be raised: Does normal boosting offer any advantages over this crippled maximum likelihood method? If not, then does the crippled maximum likelihood (and boosting) offer any advantages over real maximum likelihood? If yes, (for example, if the crippled ML method avoids overfitting) is the avoidance of overfitting obtained by crippling maximum likelihood in this way a better approach than limiting model complexity or adding penalty terms? In particular, does boosting or crippling maximum likelihood really avoid the need to set some fudge parameter such as the magnitude of the penalty by hand? If so, is its solution better than other automatic methods, such as setting the magnitude of the penalty by cross-validation?

Acknowledgements

We thank the Isaac Newton Institute, Cambridge for hospitality.

References

Freund, Y., and Schapire, R. (1995) A decision–theoretic generalization of on–line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pp. 23–37.

Freund, Y., and Schapire, R. (1996) Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*.

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

Friedman, J., Hastie, T., and Tibshirani, R., (1998) Additive logistic regression: a statistical view of boosting. Tech. report available from `http://www-stat.stanford.edu/~tibs/research.html`.

Schapire, R. (1990) The strength of weak learnability. *Machine learning*.

Schapire, R., and Singer, Y. (1998) Improved boosting algorithms using confidence–rated predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*.

Version 1.1