Boosted Optimization for Network Classification

Timothy Hancock
Bioinformatics Center
Institute for Chemical Research
Kyoto University, Japan

Hiroshi Mamitsuka
Bioinformatics Center
Institute for Chemical Research
Kyoto University, Japan

Abstract

In this paper we propose a new classification algorithm designed for application on complex networks motivated by algorithmic similarities between boosting learning and message passing. We consider a network classifier as a logistic regression where the variables define the nodes and the interaction effects define the edges. From this definition we represent the problem as a factor graph of local exponential loss functions. Using the factor graph representation it is possible to interpret the network classifier as an ensemble of individual node classifiers. We then combine ideas from boosted learning with network optimization algorithms to define two novel algorithms, Boosted Expectation Propagation (BEP) and Boosted Message Passing (BMP). These algorithms optimize the global network classifier performance by locally weighting each node classifier by the error of the surrounding network structure. We compare the performance of BEP and BMP to logistic regression as well state of the art penalized logistic regression models on simulated grid structured networks. The results show that using local boosting to optimize the performance of a network classifier increases classification performance and is especially powerful in cases when the whole network structure must be considered for accurate classification.

Appearing in Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9. Copyright 2010 by the authors.

1 Introduction

In this paper we propose a novel way of optimizing the parameters of a network classifier with a known structure by guiding boosted ensemble learning with explicit knowledge of a network structure. This is achieved by considering a network classifier as a logistic regression where the variables define the nodes and the interaction effects define the edges. Our proposed models seek to accurately classify the response variable from the set of predictor variables with specified iterations that form our known network structure. Furthermore, we do not assume a set network type but we exploit the flexiblity of loopy message algorithms to allow our proposed methods to optimize classification performance for an arbitrary network structure.

Real world networks often contain large complex structures which provide in themselves a complex optimization task. Message passing algorithms are particularly well suited to optimization on large complex network structures as they exploit the factorized nature of network distributions to convert the problem to many local optimization tasks (Bickson et al., 2008). The global network structure is incorporated into each local optimization task through messages sent along the edges of the graph (Kschischang et al., 2001). These messages contain distribution information about the current state of the network and are used to guide each local optimization towards the global network optimum. Message passing algorithms are flexible methods as they do not specify the form of the node potentials and also loopy variants allow for approximation over an arbitrary network structure (Kschischang et al., 2001).

Boosting performs a forward stage-wise addition of new classifiers into an ensemble to build a weighted linear classifier for the response (Freund and Schapire, 1997, Friedman et al., 2000). The construction of each new classifier is explicitly weighted to correct the errors of the previous ensemble. If we consider the boosted weights as messages passed between models within an already known boosted ensemble, then the process of optimizing each new ensemble member is analogous to optimizing a local potential function within a message passing regime.

In the paper we investigate the algorithmic similarity between message passing and boosted ensemble learning and propose two novel network classifier models, Boosted Expectation Propagation (BEP) and Boosted Message Passing (BMP). These models treat each node within a network classifier as a member of a boosted classifier for a common global response. The information used to estimate each ensemble member is constrained by the messages sent through the known network structure. In the next section we describe the structure of our network classifier by rephrasing a global network logistic regression as a factor graph with exponential loss functions as the local potential functions. We then describe the details of BEP and BMP and show that employing both methods will minimize the Conditional Kullback Leilber divergence about the known network structure. The performance of BEP and BMP is then evaluated on grid structured networks with simulated data and compared to state of the art LASSO (Tibshirani, 1996, Zou and Hastie, 2005) and ridge (Park and Hastie, 2007a) penalized logistic regression models.

2 Preliminaries

Consider a classification problem with a binary response variable $y \in [-1, 1]$ which assigns a label to each observation $i = 1 \dots N$. There are K predictor variables $X = [x_1, \dots, x_K]$ which are organized into a network structure where $ne(x_k)$ defines the set of variables that are connected neighbors of x_k within the network. We define the set of network parameters to be β where β_k models each node effect and β_{km} models each edge effect. We additionally define an arbitrary function $F(X)$ which defines the classification function over all X, and $f_k = f_k(x_k, ne(x_k))$ the classification contribution given the local network structure defined by $[x_k, ne(x_k)]$. Further we define the marginal distribution of a node x_k to be q_k and denote message passed along edges within a network by μ.

3 Network Classifiers

We construct our network classifier in the form of a standard logistic regression where each variable x_k is a network node and each edge is an interaction effect $x_k x_m$,

$$P(y = 1|X) = \frac{e^{F(X)}}{1 + e^{F(X)}} \tag{1}$$

and,

$$F(X) = \sum_k \left[\beta_k x_k + \sum_{m \in ne(x_k)} \beta_{km} x_k x_m \right] \tag{2}$$

where β in (2) are the parameters to be estimated to maximize correct classification (1).

The task is to optimize the performance of our network classifier by accurately estimating the parameters β. This problem is made difficult by the increasing network size and complexity required for modern applications. Commonly this curse of dimensionality is overcome by using a feature selection approach such as the LASSO (Tibshirani, 1996, Zou and Hastie, 2005). However we argue that when the network structure is known a priori then a feature selection approach will break the structure of the network and lead to an invalid result. In this work we propose to directly use the network structure to overcome dimensionality and build an accurate classifier for realistic sized networks.

3.1 Network Classifiers and Exponential Loss

We reorganize (1) to see that maximizing the probability of correct classification is equivalent to maximizing an exponential potential function of the global network classifier,

$$\frac{P(y = 1|X)}{1 - P(y = 1|X)} = e^{F(X)} \ . \tag{3}$$

To achieve optimal performance from the network the magnitude of $F(X)$ must be increased in the direction of correct classification of $y \in [-1, 1]$.

An analogous procedure to maximizing the exponential potential of a network classifier can be found in minimizing its exponential loss. To convert (2) into an exponential loss requires flipping of the sign of the exponent, $F(X)$. Then assuming $y \in [-1, 1]$ the general form of the exponential loss for a binary classifier can be stated (4),

$$e^{-yF(X)} = e^{-y \sum_k \left[\beta_k x_k + \sum_{m \in ne(x_k)} \beta_{km} x_k x_m \right]} \ . \tag{4}$$

Maximizing $yF(X)$ in the direction of correct classification will reduce $e^{-yF(X)}$. This link between optimizing the performance of a logistic regression and minimizing an exponential loss has been previously observed with boosted learning (Friedman et al., 2000). Boosted models have a major advantages over standard logistic regression; specifically increased accuracy and resistance to over-fitting. Boosting's resistance to overfitting is a highly desirable property when considering constructing a classifier from a realistic sized network where the number of terms within the model can easily exceed the limits of standard logistic regression.

3.2 Network Classifiers as an Ensemble of Factors

From equation (4) we can observe that our problem can be expressed as separate exponential loss functions for each network node:

$$e^{-yF(X)} = \prod_k e^{-y\left(\beta_k x_k + \sum_{m \in ne(x_k)} \beta_{km} x_k x_m\right)} \qquad (5)$$

The factorized form of (5) is equivalent to a factor graph view of the network classifier. Factor graphs provide a flexible framework for modeling an arbitrary network structure. A factor graph is a bipartite graph where the first set of nodes are the original network nodes, X, and the second set of nodes are the factor nodes, f_k, which are functional nodes that specify the dependency relationships between the network variables (Kschischang et al., 2001). In equation (5) each factor is defined to be the exponential loss of a local logistic regression involving node x_k and the interaction terms with its neighbors $ne(x_k)$. The entire network loss function (5) is then simply the combination of these local logistic regressions, similar to bagging (Breiman, 1996). We therefore can view a network classifier as an ensemble model of local node classifiers.

However, simply bagging the network factors is only making limited use of the known interaction structure within the classifier. A major advantage of the factor graph representation of a network is that it provides an easy entry point into message passing optimization algorithms (Kschischang et al., 2001) which are optimization algorithms designed for efficient parameter estimation of large complex network structures. Message passing algorithms are efficient as they assume that all information required to estimate the parameters of any node can be summarized as messages sent to that node from its neighbors. In this work we propose two algorithms for optimizing the parameters of large or complex network classifiers by combining ideas from boosting with two variants of message passing algorithms, Expectation Propagation (EP) (Minka, 2001) and loopy message passing (Murphy et al., 1999). We define the messages propagated through the network to be the combined exponential loss function of all neighboring nodes. These messages are then used as weights within the optimization at each local node classifier. Therefore in an analogous step to a boosted update the classifiers at each node are optimized to correctly classify errors made by the neighboring node classifiers. For completeness, in the following section we briefly describe boosting and then progress onto defining our proposed network classifier algorithms, Boosted Expectation Propagation (BEP) and Boosted Message Passing (BMP).

4 Boosting

The boosting algorithm used in this paper is discrete Adaboost (Freund and Schapire, 1997) which constructs a linear combination of models $F_M(X)$ of individual classifiers $f_m(X)$ (6),

$$F_M(X) = \sum_{m=1}^{M} c_m f_m(X) \qquad (6)$$

where c_m are the estimated coefficients for each model $f_m(X)$ and each model produces a discrete classification for $y \in [-1, 1]$. Boosting iteratively performs a stage-wise addition of a new classifier $f_m(X)$ to the current ensemble $F_{m-1}(X)$ to minimize the exponential loss function:

$$\text{Loss} = \underset{c_m}{\operatorname{argmin}}\; w_{m-1} e^{-y c_m f_m(X)} \qquad (7)$$

At each iteration the minimization of (7) is achieved by a weighted parameter estimation of $f_m(X)$, where the weights $w_{m-1} = e^{-yF_{m-1}(X)}$ are the errors of the previous ensemble. Then for each new model c_m is estimated to weight the importance of $f_m(X)$ to the overall ensemble. For each new model c_m is estimated by (8),

$$e_m = E_{w_{m-1}}[1_{y \neq \hat{f}_m(X)}] \text{ and } c_m = \frac{1}{2}\log\frac{1-e_m}{e_m} \qquad (8)$$

where $\hat{f}_m$ are the predictions of f_m and e_m is the error of the ensemble with $f_m(X)$ added.

5 Network Optimization

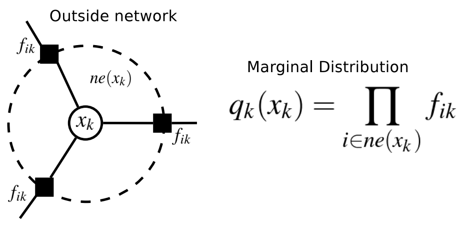

Figure 1: Example node marginal q_k in a factor graph.

Message passing and EP work by marginal estimation with respect to each variable within the network (Figure 1). Both EP and message passing define the marginal distribution, q_k of a variable x_k as the product of marginal contributions from all factors connected to x_k (9).

$$q_k(x_k) = \prod_{i \in ne(x_k)} f_{ik} \qquad (9)$$

The marginal distribution (9) contains contributions f_{ik} from each factor that specifies how x_k is connected to the rest of the network.

The final network posterior distribution is then simply the product of these marginal distributions,

$$p(X) = \prod_k q_k(x_k) = \prod_k \prod_{i \in ne(x_k)} f_{ik} \ . \qquad (10)$$

Both EP and message passing seek to maximize the overall network posterior (10) by locally optimizing each marginal contribution f_{ik}. For simplicity we represent the marginal contribution of each node as a pairwise contribution to x_k from each interaction term,

$$f_{ik} = f_{ik}(x_k, x_i) = e^{-y(\beta_k x_k + \beta_{ik} x_k x_i)} \ . \qquad (11)$$

6 Boosted Expectation Propagation (BEP)

Expectation propagation is a way to efficiently and accurately model complex posterior distributions (Minka, 2001). Expectation Propagation (EP) consistently refines the estimates of each marginal contribution by a three step process:

1. Remove the effect of the marginal contribution $\hat{f}_{ik}$ from the current posterior.
$$\hat{p}(X)^{/\hat{f}_{ik}} = \hat{p}(X)/\hat{f}_{ik}$$
2. Re-estimate f_{ik} with knowledge of the surrounding network.
$$\hat{f}_{ik} = \min \left\{ \hat{p}(X)^{/\hat{f}_{ik}} f_{ik} \right\}$$
3. Re-insert back into the network.
$$\hat{p}(X) = Z(x_k)\hat{p}(X)^{/\hat{f}_{ik}} \hat{f}_{ik}$$

where $Z(x_k)$ is a normalizing constant for the marginal distribution of x_k.

Considering the exponential form of our network classifier (2) and each marginal contribution (11) it is more efficient to work in the log space where steps (1) and (3) become simple linear operations. Additionally, $\hat{p}(X)^{/\hat{f}_{ik}}$ can now be interpreted as the current estimate of the exponential loss of the network classifier minus the marginal contribution from the interaction between nodes i and k. The addition of the f_{ik} weighted by the current exponential loss of the network classifier is clearly a boosted update (7). Performing a boosted update at step (2) defines our Boosted Expectation Propagation (BEP) algorithm as iteration of these three steps until the classifer's performance converges - Algorithm 1.

Additionally, by performing the boosted update for step (2) in Algorithm 1 we add an extra parameter for every edge within the network, c_{ik}. The parameter c_{ik} is assigned to each edge and equates to 1/2 the log odds of classification with f_{ik} . Therefore the boosted update assigns a probabilistic weight to each edge within the network based on its importance to the overall network classification.

Algorithm 1 Boosted Expectation Propagation (BEP)

```
1: for $iter \in [1 \dots \max_{iter}]$ do
2:   for $x_k \in X$ do
3:     for $i \in ne(x_k)$ do
4:       Step 1:
             $\hat{p}(X)^{/\hat{f}_{ik}} \propto e^{-y\left(F(X) - c_{ik}\hat{f}_{ik}\right)}$
5:       Step 2:
             $\hat{f}_{ik} = \min_{c_{ik}} \left\{ w_{ik} e^{-y c_{ik} f_{ik}} \right\}$
         where $w_{ik} = \hat{p}(X)^{/\hat{f}_{ik}}$.
6:       Step 3:
             $\hat{p}(X) \propto w_{ik} e^{-y\hat{f}_{ik}}$
7:     end for
8:   end for
9: end for
```

7 Boosted Message Passing (BMP)

Message passing algorithms are used to compute the marginal distribution of each node within a network (Kschischang et al., 2001). The most commonly employed message passing algorithm is the sum-product algorithm (Kschischang et al., 2001). The sum-product algorithm can estimate the node marginal distributions exactly for tree structured networks and through loopy variants allows for approximate marginal estimation for more complex network structures (Murphy et al., 1999). Simple variations of the sum-product algorithm such as the max-product algorithm, redefine the messages propagated such that they maximize the node marginal distributions. In this paper our goal is to optimize classification performance for arbitrary network structures. Therefore we employ a loopy max-product algorithm with a parallel message schedule. The parallel message schedule iterates the algorithm and requires every message within the network to be sent at each iteration.

There are two types of message propagated in the max-product algorithm:

1. From an original node x_k to factor node f_{ik}:

$$\mu_{x_k \to f_{ik}} = \prod_{\substack{j \in ne(x_k) \\ j \neq i}} \mu_{f_{jk} \to x_k} \tag{12}$$

2. From a factor node f_{ik} to original node x_k:

$$\mu_{f_{ik} \to x_k} = \max \left\{ f_{ik}(x_k, x_i) \prod_{\substack{j \in ne(f_{ik}) \\ j \neq k}} \mu_{x_j \to f_{ik}} \right\} \tag{13}$$

The action of the first message $\mu_{x_k \to f_{ik}}$ (12) is to remove the effect of f_{ik} from the marginal distribution at x_k and the action of the second message $\mu_{f_{ik} \to x_m}$ (13) is to re-estimate f_{ik} with the knowledge of local network structure of f_{ik}.

We note here that the nodes and the factors are identical to those used in EP (Section 6). It is also known that for node marginalization with the sum-product algorithm the messages sent in loopy message passing are identifical to the EP updates (Minka, 2001). However we point out here that when considering marginal maximization the information contained within the weights used to optimize each factor at each iteration of EP and loopy message passing are different. We show this by expanding the BEP weights $\hat{p}(X)^{/\hat{f}_{ik}}$ to show the contribution made from each node within the network,

$$\hat{p}(X)^{/\hat{f}_{ik}} = \prod_{\substack{j \in ne(f_{ik}) \\ j \neq k}} \hat{f}_{jk} \prod_{m \neq k} \hat{f}_m \ . \tag{14}$$

The BEP weights (14) have two terms: the first being the marginal contributions to node x_k excluding the component currently being re-estimated f_{ik}; and the second being the combined marginal distributions of all network nodes except x_k. The first term in (14) is exactly the messages passed to a factor node f_{ik} in BMP (13). This allows the BEP optimization step (Step 2 in Algorithm 1) to be expressed in terms of the BMP messages (15):

$$\hat{p}(X)^{/\hat{f}_{ik}} f_{ik} = \min_{c_{ik}} \left\{ f_{ik} \prod_{\substack{j \in ne(f_{ik}) \\ j \neq k}} \mu_{x_j \to f_{ik}} \prod_{m \neq k} \hat{f}_m \right\} \tag{15}$$

If we were to consider node marginalization with respect to x_k and use the sum-product algorithm rather than BMP's max-product algorithm, the minimization operation in (15) would be replaced with a summation. Therefore as the term $\prod_{m \neq k} f_m$ in (15) does not involve x_k it would be constant and could be removed from the summation. The EP and message passing updates would now be the same (Minka, 2001, Bishop, 2006). However we are considering a minimization problem where the second term in the BEP update corresponds to the exponential loss of the network classifier excluding node x_k and cannot be removed from the minimization operation. Therefore to optimize classification performance the weights propagated through the network by BEP and BMP are different.

The effect of the additional network terms within the BEP weights is that the boosted update in BEP is adding $\hat{f}_{ik}$ to the full network ensemble involving all terms except f_{ik}. Message passing however only provides local network messages at each optimization step (13). Therefore, using a boosted update as the optimization step in (13) is adding the new estimate of f_{ik} only to the current ensemble of local messages. This means that using BMP to propagate messages through the network is essentially building one boosted ensemble along every edge of the network in both directions. This leads to our boosted message passing algorithm - Algorithm 2.

The implications of using a boosted update as the optimization step in a loopy message passing algorithm is that each message now becomes a boosted ensemble with a size equal to the number of iterations performed. Therefore we have additional space requirements as we are required to maintain a history of each message propagated.

Algorithm 2 Boosted Message Passing (BMP)

```
1: for $iter \in [1 \dots \max_{iter}]$ do
2:     For all messages from $x_k$ to $f_m$:
           $\mu^{iter}_{x_k \to f_{ik}} = \prod_{j \in ne(x_k), j \neq i} \left[ e^{-y c_{ik} f_{ik}} \right]^{iter-1}$
3:     For all messages from $f_m$ to $x_k$:
           $\mu^{iter}_{f_{ik} \to x_k} = \min_{c_{ik}} \left\{ \left( w^{iter-1}_{ik} \right) e^{-y c_{ik} f_{ik}} \right\}$
       where $w^{iter-1}_{ik} = \prod_{j \in ne(f_{ik}), i \neq k} \mu^{iter-1}_{x_j \to f_{ik}}$
4: end for
```

8 Convergence and Complexity

We assess the convergence of BEP and BMP using the conditional Kullback Leiblier divergence (CKL). Given a two conditional distributions $P(X|y)$ and $Q(X|y)$ the conditional Kullback Lieblier divergence is:

$$CKL(P||Q) = \sum_{y,X} P(X|y) \log \frac{P(X|y)}{Q(X|y)} \tag{16}$$

We motivate the choice of the CKL by observing that both boosting and logistic regression in the standard setting have been shown to be optimizers for the CKL (Kivinen and Warmuth, 1999, Lebanon and Lafferty, 2002). Furthermore, the CKL allows us to explicitly state the probabilistic form of our network classifier.

Proposition 1 *If we assume the true distribution of our network classifier is $Q(X|y)$ which we want accurately approximate with $P(X|y)$. We further assume $P(X|y)$ has the same factorization as $Q(X|y)$ and each factor of $P(X|y)$ is modeled with an exponential loss potential,*

$$P(X|y) = \frac{1}{Z(X)} \prod_k e^{-yf_k} \ , \ Q(X|y) = \prod_k q_k \tag{17}$$

where $Z(X)$ is a normalizing constant for $P(X|y)$. We propose that approximating each true factor q_k with a boosted update to get f_k will optimize the conditional Kullback-Leilber divergence.

To address Proposition 1 we express the CKL (16) in terms of our assumed distributions (17) to get:

$$CKL = \sum_{y,X} P(X|y) \sum_k [-yf_k - \log Z(X) q_k] \tag{18}$$

We now state that the true distribution should perfectly classify the response y which results in $q_k = 1 \ \forall \ y$. Therefore we only need to consider the terms from $P(X|y)$:

$$CKL = -N \log Z(X) - \sum_{y,X} yF(X)P(X|y) \tag{19}$$

Given that $Z(X)$ is simply the exponential loss $Z(X) = \sum_{i=1}^{N} e^{-y_i F(X_i)}$ and $P(X|y)$ is the normalized exponential loss, we see that the magnitude of $Z(X)$ and $P(X|y)$ will decrease exponentially as $yF(X)$ increases. Furthermore as the boosting can only increase $yF(X)$ linearly, the second term in (19) will be dominated by $P(X|y)$. Therefore increasing $yF(X)$ with a boosted update will reduce the magnitude of both terms in (19) and the CKL will approach 0. This finding confirms Proposition 1 and agrees with previous research by (Kivinen and Warmuth, 1999, Lebanon and Lafferty, 2002).

Both BEP and BMP update the same number of marginal contributions at each iteration by computing $2E$ logistic regressions, where E is the number of edges in the network. As we are using the standard Iteratively Reweighted Least Squares (IRLS) optimizer for each logistic regression this takes $O(EKN^2)$ where K is the number of variables in each logistic regression and N is the number of observations. The high computation cost of both methods means that we require a good indicator for convergence. We see that optimizing the CKL is proportional to optimizing the exponential loss of the entire network classifier. However the exponential loss does not contain the linear term $yF(X)$ that dampens the CKL it will converge faster. Therefore considering time and space limitations in practice the relative change in exponential loss as a good approximate measure to assess convergence.

9 Experiments

We assess the performance of BEP and BMP we embed a simulated exponentially distributed network within a noise distribution. Our simulated network model has the form,

$$p(X) = \prod_i e^{\theta_i x_i + \sum_{j \in ne(x_i)} \theta_{ij} x_i x_j} \tag{20}$$

where θ are the parameters of the network, $X \in \{-1, 1\}$ and the network neighbors $ne(x_i)$ are defined over a 2D grid.

To simulate our model N observations for each predictor variable x_i are generated as random draws from a uniform distribution such that each $x_i \in \{-1, 1\}$. Then parameters θ are also drawn from a random uniform $\theta \in [-1, 1]$. All observations from our network model distribution are then assigned the response label $y = 1$. Then our noise distribution is generated as N observations for each predictor variable x_i taken from a random continuous uniform distribution Unif$\{min(X[y = 1]), max(X[y = 1])\}$ and are assigned the response label $y = -1$. These two datasets are augmented to create our base classifier distribution where the task is to correctly classify the observations of the network from the noise distribution.

However we wish to assess the performance of BEP and BMP given the same underlying data distribution but under varying network interaction strengths. To do this we hold the data distribution X and θ constant and apply a scaling parameter α across all theta $\theta_{ij} = \alpha\theta_{ij}$. We then generate different network distributions by increasing α from 0.5 to 3 in increments of 0.25. By maintaining the same underlying data distribution and network structure, as we increase α the network interactions become stronger and classification should become easier.

The performance of BEP and BMP is evaluated using the Area Under the Curve (AUC) of Receiver Oper-

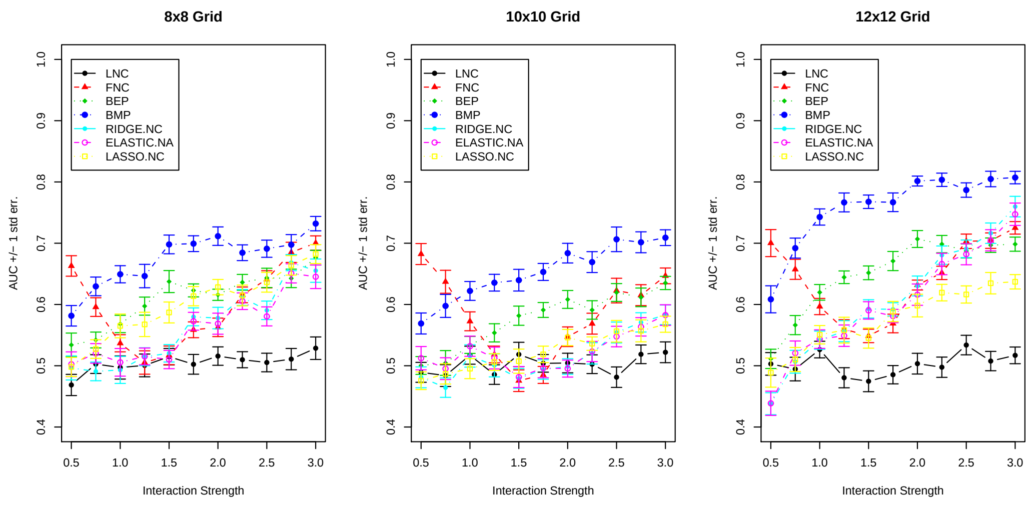

Figure 2: Mean AUC for all models for three grid sizes. The x-axis is the network scaling factor α. Larger AUC means better performance.

ating Curve (ROC). We compare the mean AUC of BEP and BMP from 5 separate runs of 5 fold cross-validation with the following logistic regression models:

- **LNC**: The complete network classifier model stated in (2) and solved with the standard IRLS algorithm.
- **FNC**: A bagged logistic network classifier model which factorizes the NC model (4) and optimizes each node classifier independently.
- **RIDGE.NC**: The LNC model with a ridge penalty (Park and Hastie, 2007a).
- **LASSO.NC**: The LNC model with a lasso penalty (Tibshirani, 1996).
- **ELASTIC.NC**: The LNC model with an elastic net penalty which is a combination of the ridge and lasso penalties (Zou and Hastie, 2005).

All non-penalized logistic regressions were performed with the *glm* package in R (Ihaka and Gentleman, 1996). The RIDGE.NC implementation used was found in the *stepPlr* R package (Park and Hastie, 2007b) and the LASSO.NC and ELASTIC.NC implementations were found in the *glmnet* R package (Friedman et al., 2009).

We note here that the connection between network classifiers and logistic regression has been well established for Bayesian networks (Greiner et al., 2005, Roos et al., 2005). Further the link between boosting through logistic regression to Bayesian networks has already been exploited by ensemble methods that use Bayesian network classifiers as a base learner (Jing et al., 2008). However the main thrust of this previous work is network structure learning which limits the complexity of the networks considered. Due to their assumption of simple network structures we cannot compare either BEP or BMP to these methods.

10 Results and Discussion

The performances of all models are compared on 3 grid sizes 8, 10 and 12 and the results are presented in Figure 2. Observation of Figure 2 quickly shows BMP to be the superior model over all grid sizes and for increasing network strengths. The fact that BMP is outperforming all other models is not surprising given the model's increased complexity which requires the construction of a boosted ensemble classifier for each message sent. BEP has a similar performance profile to BMP however only shows increased performance compared to the benchmark methods in the middle network strength range between 1 and 2.5.

The performance drop between BEP relative to BMP can be explained by the amount of classification information contained within the boosted weights supplied at each optimization step (14, 15). The weights supplied to BEP at each iteration contain terms from the entire network posterior. Therefore, if the current BEP model is performing well the boosted weights used to update each marginal contribution classifier will be sparse. This sparse weight distribution will reduce the size of the new marginal contribution to be added to the ensemble. This reduction of the marginal contribution reflects in the convergence history of BEP which is shown to converge consistently in under 10 iterations whereas BMP usually requires significantly more. The dampening of the marginal contributions combined with rapid convergence has the result of reducing the overall performance of the BEP classifier. Perhaps BEP's performance could be increased by penalizing the step size taken by each boosted update in a similar approach to (Friedman, 2001) however this is not explored in this paper.

Of the benchmark classifiers the performance of FNC yields the most insights into the performance of BEP and BMP. For small and large network strengths, FNC's performance is equivalent to and sometimes superior to that of BEP and BMP. This is because when the network interactions are small or large the classification task at each node becomes simple and does not require input from the surrounding network structure. Therefore, each node classifier can be treated independently. As bagging ensembles perform well when each model is essentially independent (Breiman, 2001) it is not surprising that at small and large network strengths the performance of FNC increases and the improvement offered by a boosted ensemble decreases.

The penalized models are shown not to perform particularly well. We stated earlier that they are not well suited to this learning task. Penalized models are powerful feature selection methods. However the networked data structure simulated in these experiments has no reduced subset of variables that offer an improved classification solution. In this case, as no best subset can be found, the performance improvements offered by penalized approaches are not realized.

11 Conclusions

In this paper we have proposed two methods, BEP and BMP, that use boosted learning within a message passing framework to optimize the performance of a network classifier. We use simulation experiments to show that BEP and BMP generate superior classification models in the case where noise disguises the true network structure. Both methods show much promise for future development as they offer improved classification performance. Furthermore they are flexible as they do not assume a constant data type across the network or a specific type of network structure.

Acknowledgements

Timothy Hancock was supported by JSPS and BIRD. Hiroshi Mamitsuka was supported in part by BIRD and JST.

References

D. Bickson, E. Yom-Tov, and D. Dolev. A gaussian belief propagation solver for large scale support vector machines. *5th European Complex Systems Conference*, 2008.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN 0-387-31073-8.

L. Breiman. Bagging predictors. *Mach. Learn.*, 26:123–140, 1996.

L. Breiman. Random forests. *Mach. Learn.*, 45:5–32, 2001.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1997.

J. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:337–374, 2000.

J. Friedman, T. Hastie, and R. Tibshirani. glmnet: Lasso and elastic-net regularized generalized linear models. 2009.

R. Greiner, X. Su, B. Shen, and W. Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Mach. Learn.*, 59(3): 297–322, 2005. ISSN 0885-6125.

R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

Y. Jing, V. Pavlovic, and J. M. Rehg. Boosted Bayesian network classifiers. *Machine Learning Journal*, 2008.

J. Kivinen and M. K. Warmuth. Boosting as entropy projection. In *COLT*, pages 134–144, 1999.

F. R. Kschischang, B. J. Frey, and H. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

G. Lebanon and J. Lafferty. Boosting and Maximum Likelihood for Exponential Models. In *NIPS*. MIT Press, 2002.

T. Minka. Expectation propagation for approximate bayesian inference. In Jack S. Breese and Daphne Koller, editors, *UAI*, pages 362–369. Morgan Kaufmann, 2001.

K.P. Murphy, Y. Weiss, and M.I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *UAI*, 1999.

M. Y. Park and T. Hastie. Penalized logistic regression for detecting gene interactions. *Biostatistics*, 2007a.

M. Y. Park and T. Hastie. stepplr: L2 penalized logistic regression with a stepwise variable selection, 2007b.

T. Roos, P. Grünwald, P. Myllymäki, H. Tirri, P. Larrañaga, J. A. Lozano, J M. Peña, and I. Inza. On discriminative Bayesian network classifiers and logistic regression. In *Machine Learning*, pages 267–296, 2005.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, 67:301–320, 2005.