The Iterative Bayesian Model Averaging Algorithm: an improved method for gene selection and classification using microarray data

Ka Yee Yeung, Roger E. Bumgarner, and Adrian E. Raftery

October 17, 2016

1 Introduction

Classification is a supervised learning technique that in the context of microarray analysis is most frequently used to identify genes whose expression is correlated with specific phenotypes of the samples. Typically, the interest is in identifying genes that are predictive of disease. In such cases, both the accuracy of the prediction and the number of genes necessary to obtain a given accuracy is important. In particular, methods that select a small number of relevant genes and provide accurate classification of microarray samples aid in the development of simpler diagnostic tests. In addition, methods that adopt a weighted average approach over multiple models have the potential to provide more accurate predictions than methods that do not take model uncertainty into consideration. Hence, we developed the Bayesian Model Averaging (BMA) method for gene selection and classification of microarray data (Yeung et al., 2005). Typical gene selection and classification procedures ignore model uncertainty and use a single set of relevant genes (model) to predict the class labels. BMA is a multivariate technique that takes the interaction of variables (typically genes) and model uncertainty into consideration. In addition, the output of BMA contains posterior probabilities for each prediction which can be useful in assessing the correctness of a given call (diagnosis).

1.1 Bayesian Model Averaging (BMA)

Bayesian Model Averaging (BMA) takes model uncertainty into consideration by averaging over the posterior distributions of a quantity of interest based on multiple models, weighted by their posterior model probabilities (Raftery, 1995). The posterior probability that a test sample belongs to a given class is equal to the weighted average of the posterior probability that the test sample belongs to the given class computed using the set of relevant genes in model M_k multiplied by the posterior probability of model M_k, summed over a set of "good" models M_k.

1.2 Iterative Bayesian Model Averaging (BMA) algorithm

The BMA algorithm we have described is limited to data in which the number of variables is greater than the number of responses. In the case of classifying samples using microarray

data, there are typically thousands or tens of thousands of genes (variables) under a few dozens samples (responses).

In this package, the iterative BMA algorithm for the binary classification problem is implemented. In the binary iterative BMA algorithm, we start by ranking the genes in order using a univariate measure such as the ratio of between-group to within-group sum of squares (BSS/WSS) (Dudoit et al., 2002). In this initial preprocessing step, genes with large BSS/WSS ratios (i.e., genes with relatively large variation between classes and relatively small variation within classes) receive high rankings. We then apply the traditional BMA algorithm to the $maxNvar$ top ranked genes, where we used $maxNvar = 30$. This is because the traditional BMA algorithm employs the leaps and bounds algorithm that is inefficient for numbers of genes (variables) much greater than 30. Then genes to which the BMA algorithm gives low posterior probabilities of being in the predictive model are removed. In our study, we used 1% as the threshold and removed genes with posterior probabilities $< 1\%$. Suppose m genes are removed. The next m genes from the rank ordered BSS/WSS ratios are added back to the set of genes so that we maintain a window of $maxNvar$ genes and apply the traditional BMA algorithm again. These steps of gene swaps and iterative applications of BMA are continued until all genes are subsequently considered.

2 Some examples

The R packages `BMA`, `leaps` and `Biobase` are required to run the key commands in this package.

```
> library ("Biobase")
> library("BMA")
> library("iterativeBMA")
```

An adapted leukemia dataset (Golub et al., 1999) is included for illustration purposes. The adapted leukemia dataset consists of the top 100 genes selected using the BSS/WSS statistic. The training set consists of 38 samples, while the test set consists of 34 samples.

```
> ## use the sample training data. The {\it ExpressionSet} is called trainData.
> data(trainData)
> ## the class vector (0,1) is called trainClass
> data(trainClass)
```

The function `iterateBMAglm.train` selects relevant variables by iteratively applying the `bic.glm` function from the `BMA` package. The data is assumed to consist of two classes. Logistic regression is used for classification. In the training phase, only the training dataset and the corresponding class labels are required as input. The parameter p represents the number of top BSS/WSS ranked genes used in the iterative process. Our studies showed that a relatively large p typically yields good results (Yeung et al., 2005). In this example, there are 100 genes in the sample training set, and we used p= 100 in the iterative BMA algorithm.

```
> ## training phase: select relevant genes
> ret.bic.glm <- iterateBMAglm.train (train.expr.set=trainData, trainClass, p=100)
```

```
[1] "5: explored up to variable ## 100"

> ret.bic.glm$namesx

 [1] "U50136_rna1_at"           "X95735_at"
 [3] "M30703_s_at"              "AFFX-HUMTFRR/M11507_3_at"
 [5] "Y12670_at"                "M23197_at"
 [7] "X17042_at"                "Z69881_at"
 [9] "X51521_at"                "U05572_s_at"
[11] "U62136_at"                "X12451_at"
[13] "X62535_at"                "M54995_at"
[15] "M11722_at"                "M63379_at"
[17] "M31523_at"                "X85116_rna1_s_at"
[19] "X59350_at"                "M20203_s_at"
[21] "Y00339_s_at"              "M27783_s_at"
[23] "J04615_at"                "X97748_s_at"
[25] "X76648_at"                "M13792_at"
[27] "U85767_at"                "M91432_at"
[29] "U41813_at"                "D26308_at"

> ret.bic.glm$probne0

 [1] 19.8 47.2  6.6  6.6 13.2 26.4  6.6  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
[16]  0.0  0.0 13.2  0.0  0.0 13.2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  6.6

> ## get the selected genes with probne0 > 0
> ret.gene.names <- ret.bic.glm$namesx[ret.bic.glm$probne0 > 0]
> ret.gene.names

 [1] "U50136_rna1_at"           "X95735_at"
 [3] "M30703_s_at"              "AFFX-HUMTFRR/M11507_3_at"
 [5] "Y12670_at"                "M23197_at"
 [7] "X17042_at"                "X85116_rna1_s_at"
 [9] "Y00339_s_at"              "D26308_at"

> ## get the posterior probabilities for the selected models
> ret.bic.glm$postprob

 [1] 0.40650525 0.06594386 0.06594386 0.06594386 0.06594386 0.06594386
 [7] 0.06594386 0.06594386 0.06594386 0.06594386
```

The `iterateBMAglm.train` function returns an object of class `bic.glm` from the last iteration of `bic.glm`. The object is a list consisting of many components. Here are some of the relevant components:

- `namesx`: the names of the variables in the last iteration of `bic.glm`.

- `postprob`: the posterior probabilities of the models selected. The length of this vector indicates the number of models selected by BMA.
- `which`: a logical matrix with one row per model and one column per variable indicating whether that variable is in the model.
- `probne0`: the posterior probability that each variable is non-zero (in percent) in the last iteration of `bic.glm`. The length of this vector should be identical to that of `namesx`.
- `mle`: matrix with one row per model and one column per variable giving the maximum likelihood estimate of each coefficient for each model.

In the training phase, the relevant variables (genes) are selected using the training data and the corresponding class labels. In the test phase, we use the selected variables (genes), the selected models, and the corresponding posterior probabilities to compute the predicted posterior probabilities that each sample belongs to class 1. The predicted posterior probability of a test sample is equal to the weighted average of the predicted probability of the test sample under each selected model, multiplied by the predicted posterior probability of each model. Note that in this case, a model consists of a set of genes, and different models can potentially have overlapping genes. The posterior probability of a gene is equal to the sum of the posterior probabilities of all the models that the gene belongs to.

```
> ## The test ExpressionSet is called testData.
> data (testData)
> ## get the subset of test data with the genes from the last iteration of bic.glm
> curr.test.dat <- t(exprs(testData)[ret.gene.names,])
> ## to compute the predicted probabilities for the test samples
> y.pred.test <- apply (curr.test.dat, 1, bma.predict, postprobArr=ret.bic.glm$postprob, mleA
> ## compute the Brier Score if the class labels of the test samples are known
> data (testClass)
> brier.score (y.pred.test, testClass)

[1] 2.106802
```

The Brier Score computes the sum of squares of the differences between the true class and the predicted probability over all test samples. If the predicted probabilities are constrained to equal to 0 or 1, the Brier Score is equal to the total number of classification errors.

The function `iterateBMAglm.train.predict` combines the training and prediction phases, and returns the predicted posterior probabilities that each test sample belongs to class 1.

```
> ## train and predict
> ret.vec <- iterateBMAglm.train.predict (train.expr.set=trainData, test.expr.set=testData, t

[1] "5: explored up to variable ## 100"

> ## compute the Brier Score
> data (testClass)
> brier.score (ret.vec, testClass)
```

```
[1] 2.221017
```

The function `iterateBMAglm.train.predict.test` combines the training, prediction and test phases, and returns a list consisting of the numbers of selected genes and models using the training data, the number of classification errors and the Brier Score on the test set.

```
> iterateBMAglm.train.predict.test (train.expr.set=trainData, test.expr.set=testData, trainCl
```

```
[1] "5: explored up to variable ## 100"
$num.genes
[1] 10

$num.model
[1] 10

$num.err
[1] 1

$brierScore
[1] 2.221017
```

This package also contains the `imageplot.iterate.bma` function, which allows us to create a heatmap-style image to visualize the selected genes and models (see Figure 1).

In Figure 1, the BMA selected variables are shown on the vertical axis, and the BMA selected models are shown on the horizontal axis. The variables (genes) are sorted in decreasing order of the posterior probability that the variable is not equal to 0 (`probne0`) from top to bottom. The models are sorted in decreasing order of the model posterior probability (`postprob`) from left to right.

Acknowledgements

We would like to thank Drs. Ian Painter and Chris Volinsky. Yeung is supported by NIH-NCI 1K25CA106988. Bumgarner is funded by NIH-NIAID grants 5P01AI052106, 1R21AI052028 and 1U54AI057141, NIH-NIEHA 1U19ES011387, NIH-NHLBI grants 5R01HL072370 and 1P50HL073996. Raftery is supported by NIH 8R01EB002137 and ONR N00014-01-10745.

References

S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of discrimination methods for the classification of tumors in gene expression data. *Journal of the American Statistical Association*, 97:77–87, 2002.

T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

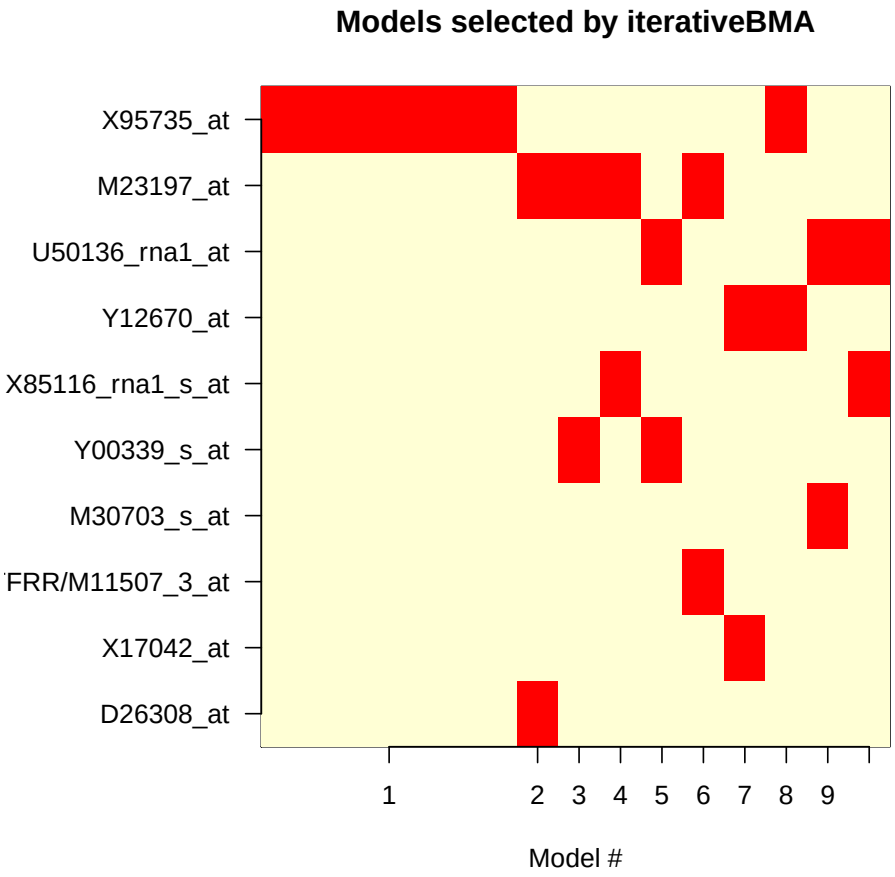

Figure 1: An image plot showing the selected genes and models.

A.E. Raftery. Bayesian model selection in social research (with discussion). *Sociological Methodology*, 25:111–196, 1995.

K.Y. Yeung, R.E. Bumgarner, and A.E. Raftery. Bayesian Model Averaging: Development of an improved multi-class, gene selection and classification tool for microarray data. *Bioinformatics*, 21(10):2394–2402, 2005.