Boosting, applied to Classification of Gene Expression Data

Niv Efron

nefron@post.tau.ac.il

Overview

- Boosting
  - Theory [AdaBoost]
  - Statistical point of view [additive models, loss-functions, LogitBoost]
- Gene Expression Classification
  - Characteristics & Challenges
- Applications on DNA data
  - Modifications needed for LogitBoost
  - Stumps
  - ROC curves
  - Results & Simulations
  - BagBoosting

Classification

- Notations:
  - x_i – an observation (p-dimensional - R^p)
  - y_i – class label { -1,1}
  - n – number of training samples
- Classification task:
  - Build a classifier $C : R^p \rightarrow \{-1,1\}$ such that $P[C(x) \neq y]$ is minimal

Boosting

- (Freund & Schapire)
- "Weak learner"
  - $f : R^p \rightarrow \{-1,1\}$
  - A classifier with performance (error on training data) guaranteed (with high probability) to be better than random.
- Boosting
  - Train a series of weak learners $\{f_1,..,.f_M\}$, and ensemble them (a "committee" vote) to boost performance.

AdaBoost

- Adaptive Boosting
- In each AdaBoost iteration $m = 1,\ldots,M$:
  - The classifier f_m is trained on a weighted version of the training set $\{x_1,\ldots,x_n\}$ (weights $\{w_1,\ldots,w_n\}$)
  - According to the classification results, the training points are re-weighted for the next iteration
    - Increase the weight for misclassified samples
- Finally, the combined classifier:
  - a weighted committee vote of $\{f_1,\ldots,f_M\}$ (weights $\{\alpha_1,\ldots,\alpha_M\}$)

$$\hat{F}_M(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m f_m(\mathbf{x})$$

- By adaptively re-weighting the training samples, we force classifier f_m to focus on the samples which were difficult to classify in the previous iterations.

AdaBoost

- Choosing components:
  - weak learner
  - weights for data – $\{w_1,\ldots,w_n\}$
  - weights for aggregating classifiers – $\{\alpha_1,\ldots,\alpha_M\}$
  - number of boosting iterations – M
- Common choices:
  - Weak learner – decision trees
  - M – fixed (usually 100) or decided by CV

Discrete AdaBoost Algorithm

Discrete AdaBoost(Freund & Schapire 1996*b*)

1. Start with weights $w_i = 1/N$, $i = 1, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights w_i on the training data.
   (b) Compute $\text{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (c) Set $w_i \leftarrow w_i \exp[c_m \cdot 1_{(y_i \neq f_m(x_i))}]$, $i = 1, 2, \ldots N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} c_m f_m(x)]$

Algorithm 1: *E_w represents expectation over the training data with weights $w = (w_1, w_2, \ldots w_n)$, and $1_{(S)}$ is the indicator of the set S. At each iteration AdaBoost increases the weights of the observations misclassified by $f_m(x)$ by a factor that depends on the weighted training error.*

(this notation uses $\{c_1, \ldots, c_M\}$ instead of $\{\alpha_1, \ldots, \alpha_M\}$)

recall that $f_m(x) : \mathcal{X} \mapsto \{-1, 1\}$

Statistical View of Boosting

- (work by Friedman, Hastie, Tibshirani)
- Show that
  - AdaBoost fits an *additive model*
  - can be interpreted as stage-wise estimation of an additive *logistic regression* model
- Introduce LogitBoost

Additive Models

- Additive regression models
  - A separate function $f_j(x_j)$ for each of the input variables x_j.

$$F(x) = \sum_{j=1}^{p} f_j(x_j)$$

- Extended additive models
  - Each function $f_m(x)$ potentially uses all of the p input features of x.
    - γ_m – parameters set
    - β_m – multiplier
    - $b(x,\gamma_m)$ – generally called "basis functions"
  - E.g., $b(x, \gamma) = \sigma(\gamma^T x)$: single hidden layer neural network

$$f_m(x) = \beta_m b(x;\gamma_m)$$

$$F_M(x) = \sum_{m=1}^{M} \beta_m b(x;\gamma_m)$$

Additive Models

- Iterative optimization
  - Fitting criterion: least squares

$$\{\beta_m, \gamma_m\} \leftarrow \arg\min_{\beta,\gamma} E\left[y - F_{m-1}(x) - \beta b(x;\gamma)\right]^2$$

  - Forward stepwise algorithm

$$y_m = y_{m-1} - f_{m-1}(x)$$

    - At each step m, we change the output values y_m, so that the next iteration will only deal with the "residuals" left from the previous iteration

  - Only requires an algorithm for fitting a ***single*** weak learner to data
    - can be viewed as boosting $f(x) = \beta\, b(x, \gamma)$, forming a committee $F_M(x)$

Classification Problems

- Using regression for classification:
  - Use the class probabilities $P(y = j|x)$ as output values
  - Problem: regression estimates are not constrained to [0,1]
  - Solution:
    Logistic transformation: $$\log \frac{P(y=1|x)}{P(y=-1|x)} = \sum_{m=1}^{M} f_m(x) = \boldsymbol{F(x)}$$
  - Inverting: $$p(x) = P(y=1|x) = \frac{e^{F(x)}}{1+e^{F(x)}}$$
  - Guarantees that for every F(X), p(x) ∈ [0,1]

Loss Functions

- We need to define a loss function J(F) to be minimized in each iteration
- AdaBoost builds an additive logistic regression model using the exponential loss function.
- The exponential loss function:

$$J(F) = E(e^{-yF(x)})$$

Loss Functions

- The exponential loss function is similar (2nd order equivalence) to the statistically motivated, well-known:
- Binomial log-likelihood
  - $y^* = (y+1)/2$ (transforms [-1,1] to [0,1])
  - loss function:

$$\begin{aligned} \ell(y^*, p(x)) &= y^* \log(p(x)) + (1-y^*)\log(1-p(x)) \\ &= -\log(1+e^{-2yF(x)}) \end{aligned}$$

- Introduce LogitBoost – an additive logistic regression model using the binomial log-likelihood criterion.

Loss Functions

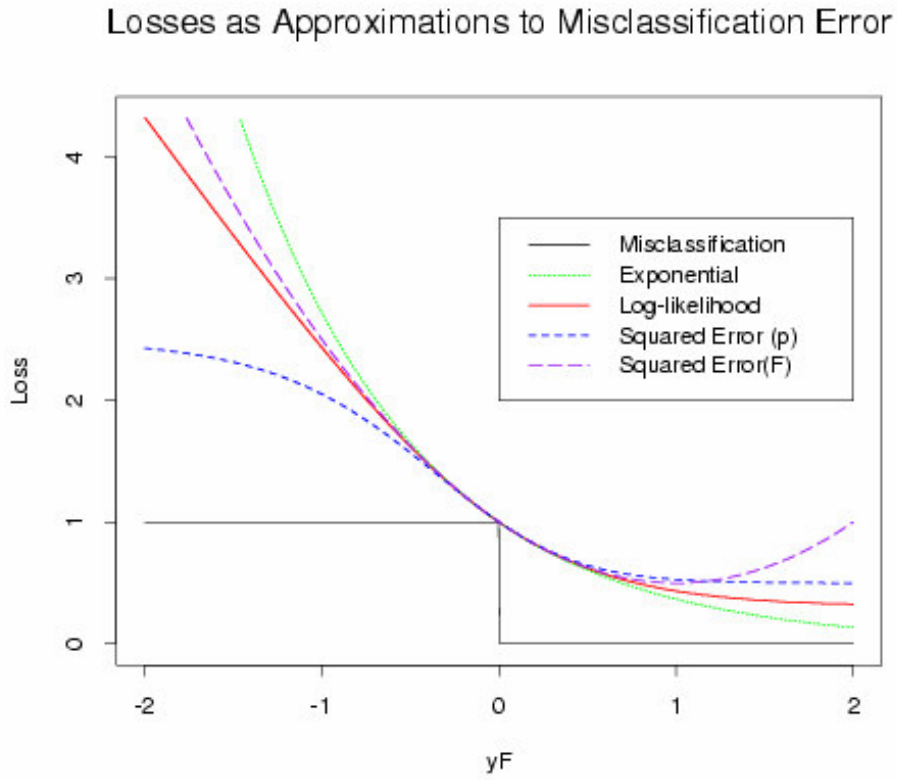


Figure 2: *A variety of loss functions for estimating a function $F(x)$ for classification. The horizontal axis is yF, which is negative for errors and positive for correct classifications. All the loss functions are monotone in yF, and are centered and scaled to match e^{-yF} at $F = 0$. The curve labeled "Log-likelihood" is the binomial log-likelihood or cross-entropy $y^* \log p + (1 - y^*) \log(1 - p)$. The curve labeled "Squared Error(p)" is $(y^* - p)^2$. The curve labeled "Squared Error(F)" is $(y - F)^2$, and* increases *once yF exceeds 1, thereby increasingly penalizing classifications that are "too correct".*

LogitBoost

LogitBoost (2 classes)

1. Start with weights $w_i = 1/N$ $i = 1, 2, \ldots, N$, $F(x) = 0$ and probability estimates $p(x_i) = \frac{1}{2}$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Compute the working response and weights

$$z_i = \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))}$$
$$w_i = p(x_i)(1 - p(x_i))$$

   (b) Fit the function $f_m(x)$ by a weighted least-squares regression of z_i to x_i using weights w_i.
   (c) Update $F(x) \leftarrow F(x) + \frac{1}{2} f_m(x)$ and $p(x) \leftarrow \frac{e^{F(x)}}{e^{F(x)} + e^{-F(x)}}$.
3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$

Algorithm 3: *An adaptive Newton algorithm for fitting an additive logistic regression model.*

Fitting f(x) (using Newton updates):

$$\min_{f(x)} E_{w(x)} \left(F(x) + \frac{1}{2} \frac{y^* - p(x)}{p(x)(1 - p(x))} - (F(x) + f(x)) \right)^2$$

Gene Expression Data

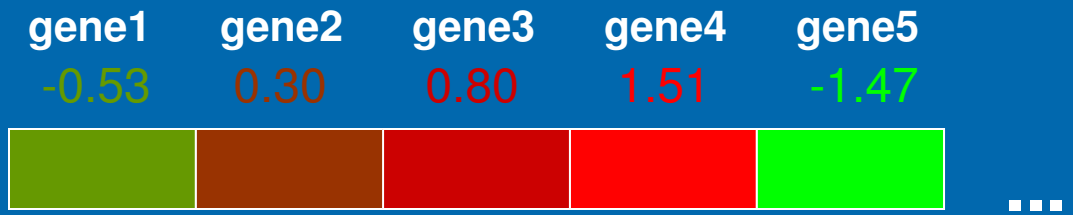
- Tissue classification using gene-expression data
- Using DNA microarray technology and computational methods for:
  - distinguishing between different cancer types
  - identifying normal / diseased tissues
- Correct classification can assist
  - Clinical diagnosis
  - Efficient and focused drug-design
  - Better understanding of gene functionality

DNA Microarrays

- Mechanism based on:
  - known sequences of genes
  - specifically designed DNA chips
  - hybridization experiments
  - image processing of the results
- In order to:
  - measure the expression levels of the genes in a given tissue

Gene Expression Data - Visualization

- One Tissue



- Several labeled tissues

The Classification Problem - Definition

- Input:
  - n observations (tissues), each has:
    - p values (gene expression levels)
    - a class label
  - Formally:
    - $X_{i,j}$ = nxp matrix of gene expression data
    - y_i = class label $(1..K)$ of observation i
- Output:
  - a trained classifier

Challenges

- "Curse of Dimensionality"
  - p (# genes) >> n (# samples)
  - very high-dimensional data (thousands)
  - few observations (dozens)
- Noisy data (DNA chips)
  - usually highly-correlated

Applying Boosting on Expression Data

- (work by Dettling, Bühlmann)
- Modifications needed:
  - Feature selection
  - LogitBoost with decision trees (stumps)
  - Dealing with multi-class problems
- Results
  - ROC curves
  - Simulations

Feature Selection

- Dimensionality reduction is essential ($p>>n$)
- Non-parametric method (Park)
  - Equivalent to Wilcoxon's test
- Individual scoring for each gene, then choosing $p^{\sim}$ genes with top scores
- $p^{\sim}$ can be chosen using cross-validation

Gene Scoring

- Scoring gene g

$$\text{Score}(g) = s(g) = \sum_{i \in \mathcal{N}_0} \sum_{j \in \mathcal{N}_1} 1_{[x_j^{(g)} - x_i^{(g)} \leq 0]}$$

- For symmetry purposes

$$q(g) = \max(s(g), n_0 n_1 - s(g))$$

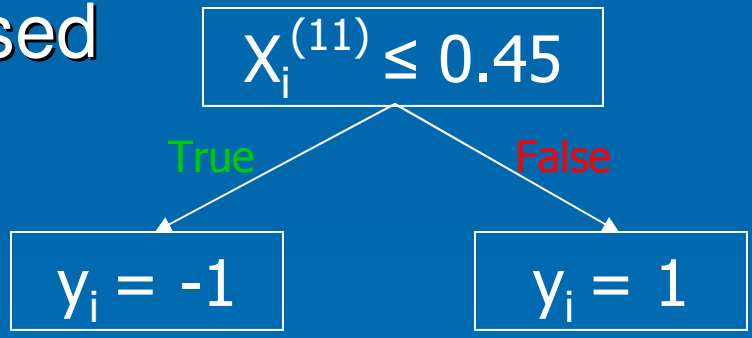
- Choose p˜ genes with highest q values

Gene Scoring - Example

	Gene 1	Gene 2	Gene 3	y
	0.6	0.5	0.1	0
	0.7	0.6	0.1	0
	0.8	0.8	0.3	0
	0.2	0.7	0.4	1
	0.1	0.4	0.4	1
	0.4	0.7	0.2	1
s(g)	9	5	1	
q(g)	9	5	8	

LogitBoost with Trees

- Boosting requires choosing a weak learner
- Decision trees are a common choice
  - Regression fitting using decision trees
- In this case, stumps are used
  - decision trees with 2 terminal nodes



  - Indeed, a weak learner…
  - No fine-tuning, or sophisticated optimization needed

Multi-Class Problems

- The algorithm deals with binary cases
- Enhancing to multi-class (J>2) problems through *one-against-all* approach:
  - Creating J binary problems
  - Each returns an estimate $\widehat{\mathbb{P}}[Y^{(j)} = 1|X]$
  - Normalize to get class probabilities:

$$\widehat{\mathbb{P}}[Y = j|X] = \frac{\widehat{\mathbb{P}}\left[Y^{(j)} = 1|X\right]}{\sum_{k=1}^{J} \widehat{\mathbb{P}}\left[Y^{(k)} = 1|X\right]}$$

  - Take the largest probability as the final classification

Results

- Employing the algorithm for 6 published real-world microarray data sets.
- Produce Leave-One-Out Cross-Validation rates
- Different values of $p^{\sim}$ (feature selection)
- Compare with other algorithms

Results

Table 1. Test set error rates based on leave one out cross validation for leukemia, colon, estrogen, nodal, lymphoma and NCI data with gene subsets from feature selection ranging between 10 to all genes for several classifiers. LogitBoost error rates are reported with optimal stopping (minimum cross-validated error across iterations), after a fixed number of 100 iterations as well as with the estimated stopping parameter. The cross validation with estimated stopping parameters for the lymphoma and NCI data with all genes was not feasible

Leukemia	10	25	50	75	100	200	3571
LogitBoost, optimal	4.17%	2.78%	4.17%	2.78%	2.78%	2.78%	2.78%
LogitBoost, estimated	6.94%	5.56%	5.56%	4.17%	4.17%	5.56%	5.56%
LogitBoost, 100 iterations	5.56%	2.78%	4.17%	2.78%	2.78%	2.78%	2.78%
AdaBoost, 100 iterations	4.17%	4.17%	4.17%	4.17%	4.17%	2.78%	4.17%
1-nearest-neighbor	4.17%	1.39%	4.17%	5.56%	4.17%	2.78%	1.39%
Classification tree	22.22%	22.22%	22.22%	22.22%	22.22%	22.22%	23.61%
Colon	10	25	50	75	100	200	2000
LogitBoost, optimal	14.52%	16.13%	16.13%	16.13%	16.13%	14.52%	12.90%
LogitBoost, estimated	22.58%	19.35%	22.58%	20.97%	22.58%	19.35%	19.35%
LogitBoost, 100 iterations	14.52%	22.58%	22.58%	19.35%	17.74%	16.13%	16.13%
AdaBoost, 100 iterations	16.13%	24.19%	24.19%	17.74%	20.97%	17.74%	17.74%
1-nearest-neighbor	17.74%	14.52%	14.52%	20.97%	19.35%	17.74%	25.81%
Classification tree	19.35%	22.58%	29.03%	32.26%	27.42%	14.52%	16.13%
Estrogen	10	25	50	75	100	200	7129
LogitBoost, optimal	4.08%	4.08%	2.04%	2.04%	2.04%	4.08%	2.04%
LogitBoost, estimated	6.12%	6.12%	6.12%	6.12%	6.12%	6.12%	6.12%
LogitBoost, 100 iterations	8.16%	6.12%	6.12%	4.08%	4.08%	8.16%	6.12%
AdaBoost, 100 iterations	8.16%	8.16%	2.04%	2.04%	6.12%	4.08%	4.08%
1-nearest-neighbor	4.08%	8.16%	18.37%	12.24%	14.29%	14.29%	16.33%
Classification tree	4.08%	4.08%	4.08%	4.08%	4.08%	4.08%	4.08%

Results – cont.

Nodal	10	25	50	75	100	200	7129
LogitBoost, optimal	16.33%	18.37%	22.45%	22.45%	22.45%	18.37%	20.41%
LogitBoost, estimated	22.45%	30.61%	30.61%	34.69%	28.57%	26.53%	24.49%
LogitBoost, 100 iterations	18.37%	20.41%	26.53%	42.86%	42.86%	18.37%	22.45%
AdaBoost, 100 iterations	18.37%	16.33%	28.57%	40.82%	36.73%	22.45%	28.57%
1-nearest-neighbor	18.37%	30.61%	30.61%	42.86%	36.73%	36.73%	48.98%
Classification tree	22.45%	20.41%	20.41%	20.41%	20.41%	20.41%	20.41%
Lymphoma	10	25	50	75	100	200	4026
LogitBoost, optimal	1.61%	3.23%	1.61%	1.61%	1.61%	3.23%	8.06%
LogitBoost, estimated	3.23%	3.23%	3.23%	1.61%	3.23%	3.23%	-%
LogitBoost, 100 iterations	1.61%	3.23%	1.61%	1.61%	1.61%	3.23%	8.06%
AdaBoost, 100 iterations	4.84%	3.23%	1.61%	1.61%	1.61%	1.61%	3.23%
Nearest neighbor	1.61%	0.00%	0.00%	0.00%	0.00%	1.61%	1.61%
Classification tree	22.58%	22.58%	22.58%	22.58%	22.58%	22.58%	25.81%
NCI	10	25	50	75	100	200	5244
LogitBoost, optimal	32.79%	31.15%	27.87%	22.95%	26.23%	24.59%	31.15%
LogitBoost, estimated	36.07%	44.26%	36.07%	39.34%	44.26%	47.54%	-%
LogitBoost, 100 iterations	37.70%	44.26%	34.43%	29.51%	26.23%	24.59%	36.07%
AdaBoost, 100 iterations	50.82%	37.70%	34.43%	29.51%	32.79%	29.51%	36.07%
Nearest neighbor	36.07%	29.51%	27.87%	24.59%	22.95%	22.95%	27.87%
Classification tree	70.49%	68.85%	65.57%	65.57%	60.66%	62.30%	62.30%

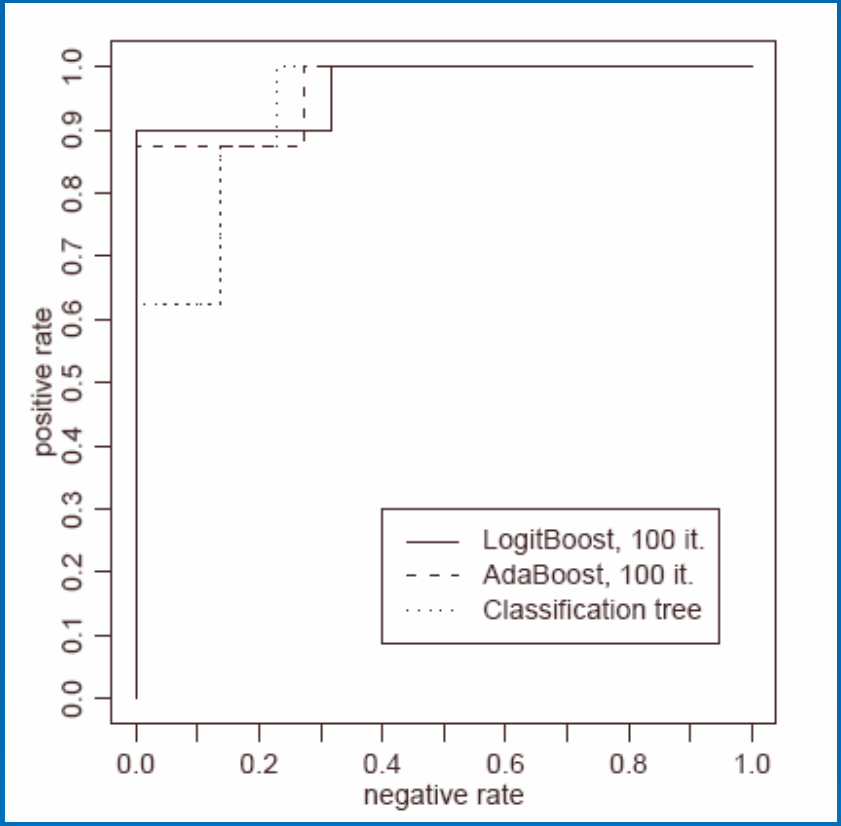
Asymmetric Losses

- The algorithm uses equal misclassification costs
  - False positive – predicting a normal tissue as diseased
  - False negative – predicting a diseased tissue as normal
- Clinical implications:
  - False negatives are much more dangerous
  - Further tests will be performed on false positives and rule them out

ROC Curves

- Receiver Operator Characteristic
  - from WW-II (radars distinguishing between ships and Noise)
- Used to evaluate the power of a classifier for different asymmetric losses
- $\beta \in [0,1]$ – a threshold for positive classifications
- We plot the fractions of positive and negative samples that are classified as positive
- Each point (x,y) represent these fractions for a specific β

ROC Curves



Colon data, $p^{\sim} = 2000$

Both boosting algorithms are closer to the ideal ROC curve

Simulations

- Benefits:
  - Limited amount of real data
  - Allows more accurate comparisons with other classifiers
- Requires a “realistic” simulation model
  - Multivariate distribution
  - Proportion of relevant genes
  - Response model (f(x) => y)

Simulations

- $X \sim N_p(0,\Sigma)$ (normal distribution)
  - Take Σ from a real data set (colon, $p=2000$)

- Response model $Y \mid X = x \sim Bernoulli(p(x))$
  - where $p(x)$ is determined by: $$\log \frac{p(x)}{1-p(x)} = \sum_{j=1}^{10} \beta_j \bar{x}^{(\mathcal{C}_j)} \left(1 + \gamma_j \bar{x}^{(\mathcal{C}_j)}\right) \left(1 + \delta_j \bar{x}^{(\mathcal{C}_j)}\right)$$
  - and $\bar{x}^{(\mathcal{C}_j)} = \sum_{g \in \mathcal{C}_j} x^{(g)} / |\mathcal{C}_j|$ is the mean value across a random gene set C_j of size ~ U([1,10])
  - Coefficients $\beta_j, \gamma_j, \delta_j$ ~ N(0,σ), σ = 2,1,0.5

Simulations – Results

- *Hypothesis*: LogitBoost outperforms benchmark methods.

Table 2. Percentual improvement and p-values of LogitBoost (stopped optimally and after a fixed number of 150 iterations) against the generic 1-nearest-neighbor method and classification trees in 20 independent realizations from our simulation model. The p-values are from paired two-sided Wilcoxon signed rank tests for equal test set error and are always in favor of LogitBoost

	1-Nearest-Neighbor	
LogitBoost, optimal	12.37%,	$p = 1.7 \cdot 10^{-4}$
LogitBoost, 150 iterations	7.54%,	$p = 1.4 \cdot 10^{-3}$
	Classification Tree	
LogitBoost, optimal	10.21%,	$p = 1.1 \cdot 10^{-3}$
LogitBoost, 150 iterations	5.27%,	$p = 1.7 \cdot 10^{-2}$

BagBoosting

- A recent enhancement (Dettling)
- Adding Bagging into the algorithm
- More simulation models
- Bias / Variance analysis
- Adding a “model recovery” capability

Bagging

- Bagging – Bootstrapping and Aggregating
- In each boosting iteration
  instead of using a *single* base learner,
  aggregate *several* learners generated from
  bootstrap samples of the re-weighted data.
- Motivation:
  - Boosting ensemble – reduces bias
  - Bagging ensemble – reduces variance

BagBoosting

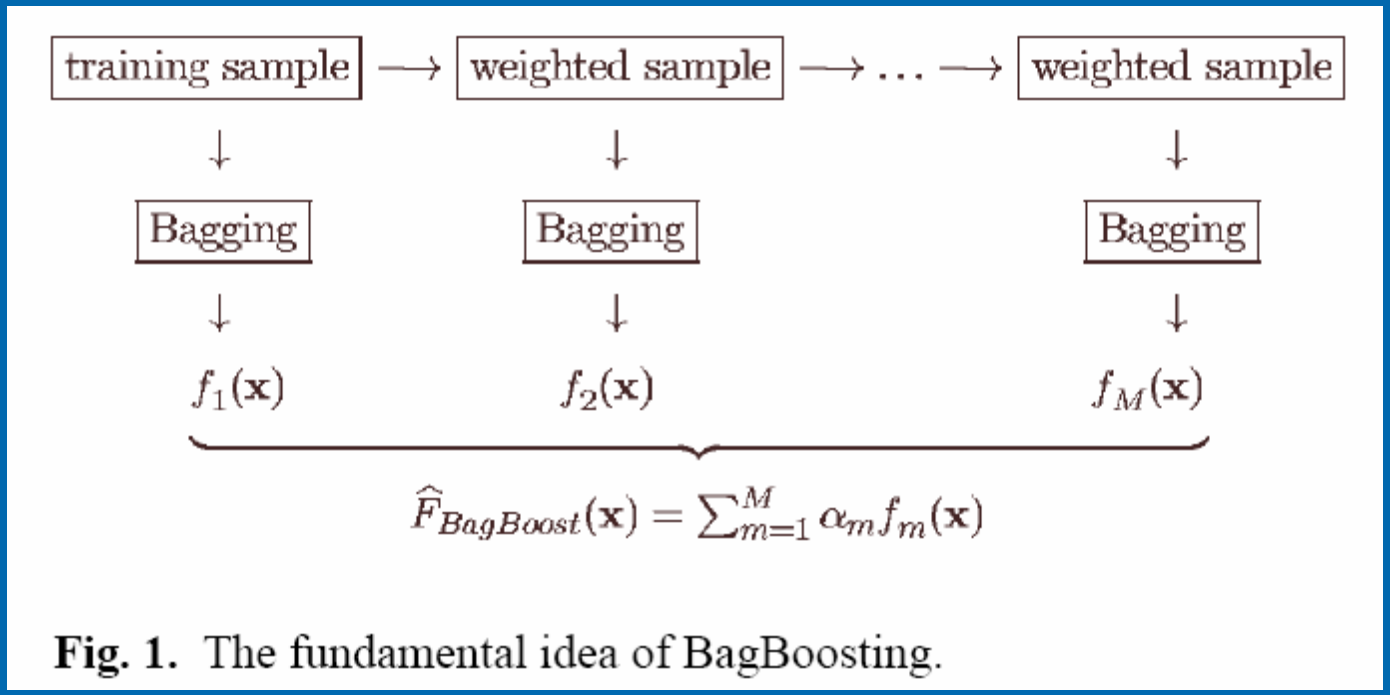


Fig. 1. The fundamental idea of BagBoosting.

- Aggregation of the B bootstrapped learners is done by averaging the B stumps

$$f_m(\mathbf{x}_i) \leftarrow \frac{1}{B} \sum_{b=1}^{B} g^{(b)}(\mathbf{x}_i)$$

Results

Table 1. Misclassification rates for seven classifiers on six microarray data-sets based on 50 random partitions into learning sets (two-thirds of the data) and test sets (one-third of the data)

	Leukemia (%)	Colon (%)	Prostate (%)	Lymphoma (%)	SRBCT (%)	Brain (%)
BagBoost	4.08	16.10	7.53	1.62	1.24	23.86
Boosting	5.67	19.14	8.71	6.29	6.19	27.57
RanFor	1.92	14.86	9.00	1.24	3.71	33.71
SVM	1.83	15.05	7.88	1.62	2.00	28.29
PAM	3.75	11.90	16.53	5.33	2.10	25.29
DLDA	2.92	12.86	14.18	2.19	2.19	28.57
*k*NN	3.83	16.38	10.59	1.52	1.43	29.71

- Real Data sets
- 50 random splits to train/test (2/3-1/3)
- $\tilde{p} = 200$

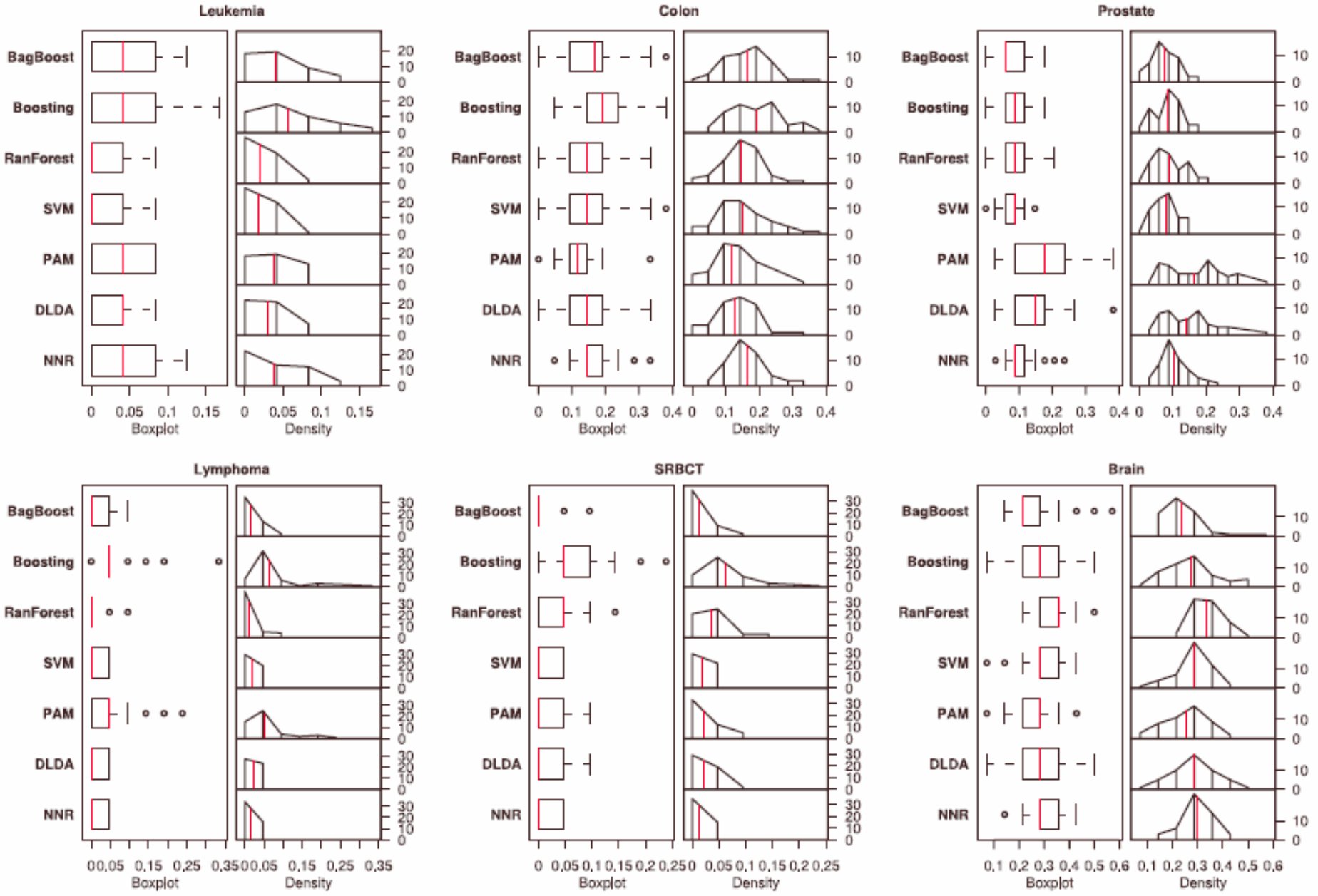


Fig. 2. Boxplots and density curves of the misclassification rates for seven classifiers on six microarray datasets based on 50 random splits into learning and test sets. The vertical red lines highlight the median (boxplots) and the mean value (density curves).

Simulations

- Take $X \sim N(\mu, \Sigma)$ according to real data
  - Leukemia data set ($p=3571$)
- 3 response models:
  - Additive (a)

$$F(\mathbf{x}) = \sum_{j=1}^{10} x_j$$

  - Weighted additive (b)

$$F(\mathbf{x}) = \sum_{j=1}^{25} \beta_j x_j$$

  - Complex interaction with 25 genes (c)

$$F(\mathbf{x}) = \sum_{j=1}^{25} \beta_j x_j \cdot \left(1 + \sum_{j=1}^{25} \gamma_j x_j\right) \cdot \left(1 + \sum_{j=1}^{25} \delta_j x_j\right)$$

- Set class labels:

$$p(\mathbf{x}) = \frac{1}{1 + \exp[-F(\mathbf{x})]},$$

$$y(\mathbf{x}) \sim \text{Bernoulli}\,[p(\mathbf{x})].$$

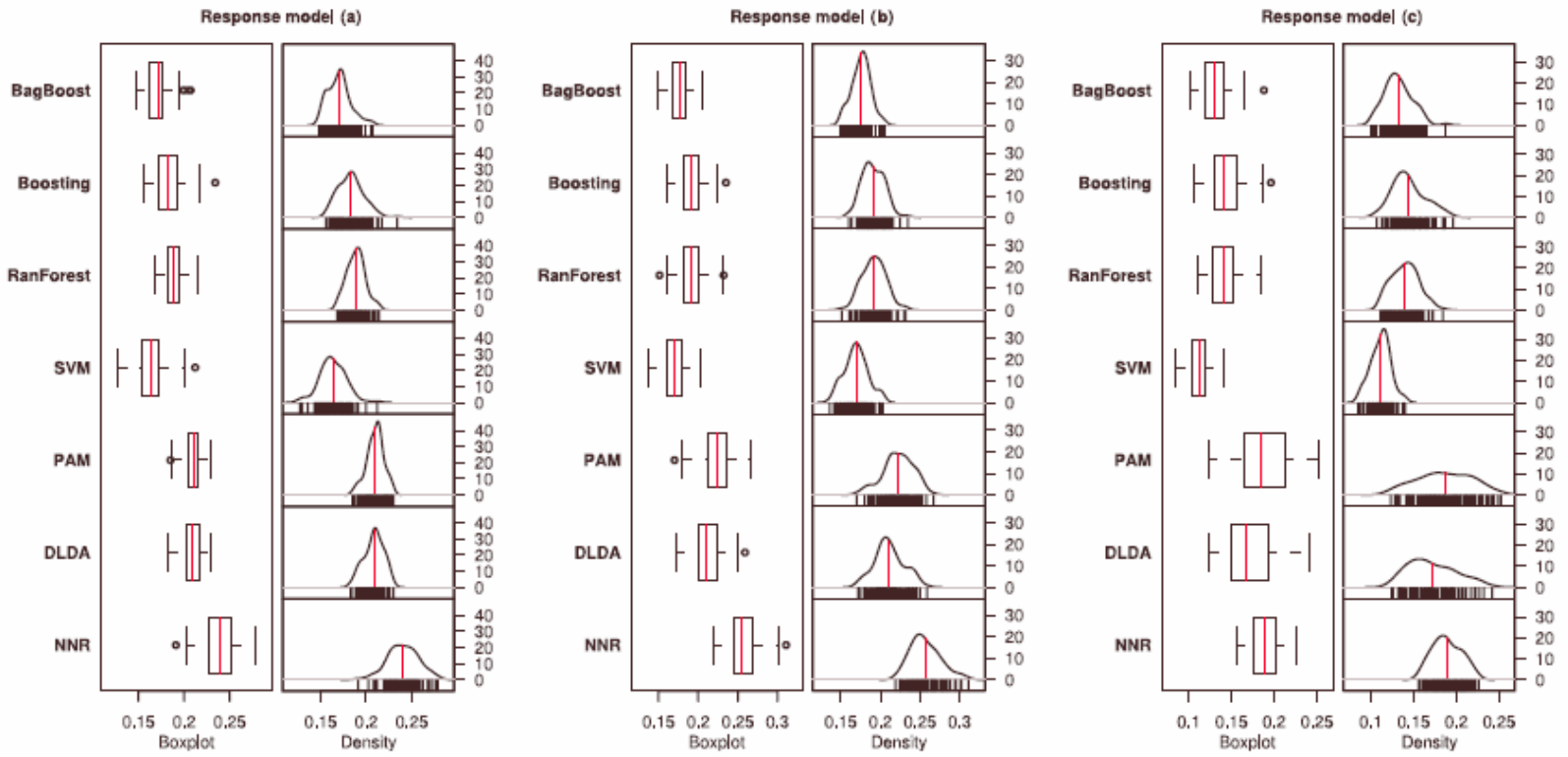


Fig. 3. Misclassification rates for outsample classification on simulated gene expression data with various classifiers and three different response models. The left panels show boxplots where the median is highlighted in red, the right panels show density curves where the red vertical line corresponds to the mean error rate.

Table 2. Misclassification rates for outsample classification on 100 iid simulation experiments with various classifiers and three different response models, as well as the number of simulations where each of the classifiers was worse than BagBoosting, and the *P*-value for the two-sided sign test that the performance is equal

	Response (a) Error (%)	Worse	*P*-value	Response (b) Error (%)	Worse	*P*-value	Response (c) Error (%)	Worse	*P*-value
BagBoost	17.02	—	—	17.57	—	—	13.19	—	—
Boosting	18.32	89	0.0000	19.09	96	0.0000	14.42	97	0.0000
RanFor	18.87	93	0.0000	19.19	87	0.0000	14.00	68	0.0004
SVM	16.33	36	0.0066	16.94	31	0.0002	11.13	7	0.0000
PAM	20.99	100	0.0000	22.25	99	0.0000	18.61	96	0.0000
DLDA	20.83	99	0.0000	21.15	98	0.0000	17.18	91	0.0000
*k*NN	24.04	100	0.0000	25.67	100	0.0000	18.90	100	0.0000

Bias – Variance

- IMSE = Integrated Mean Squared Error
- Estimations using simulations and T=1000 fixed test points:

$$\mathrm{IMSE}(\hat{p}) = \int_{\mathcal{X}} \mathrm{Var}(\hat{p}(\mathbf{x})) + \mathcal{E}[\hat{p}(\mathbf{x}) - p(\mathbf{x})]^2 d\mathcal{F}(\mathbf{x})$$

$$\mathrm{Var}(\hat{p}) = \frac{1}{T}\sum_{i=1}^{T}\left[\frac{1}{S}\sum_{k=1}^{S}[\hat{p}_k(\mathbf{x}_i) - \bar{p}(\mathbf{x}_i)]^2\right]$$

$$\mathrm{Bias}(\hat{p})^2 = \frac{1}{T}\sum_{i=1}^{T}\left[\frac{1}{S}\sum_{k=1}^{S}\hat{p}_k(\mathbf{x}_i) - p_k(\mathbf{x}_i)\right]^2$$

Table 3. Estimates of IMSE, variance and squared bias for conditional class probabilities, obtained from four different prediction methods on simulated gene expression data with three different response models

	Response (a)			Response (b)			Response (c)		
	IMSE	Var	Bias^2	IMSE	Var	Bias^2	IMSE	Vari	Bias^2
Stumps	0.102	0.041	0.061	0.116	0.040	0.075	0.128	0.037	0.091
Bagging	0.069	0.010	0.059	0.083	0.010	0.073	0.096	0.008	0.088
Boosting	0.076	0.048	0.028	0.086	0.056	0.030	0.090	0.047	0.043
BagBoost	0.045	0.020	0.026	0.050	0.023	0.027	0.056	0.019	0.037

Model Recovery

- Does BagBoosting recover the true response model – identifies the affecting genes?

- Note that the BagBoosted aggregation of stumps can be represented as an additive combination:

$$\hat{F}_M(\mathbf{x}) = \sum_{j=1}^{p} \theta_j h(x_j)$$

  - $h_j(\cdot)$ – aggregated step functions
  - θ_j – "importance coefficient"
  - Genes with high θ_j values – considered "important" by the classifier

h()

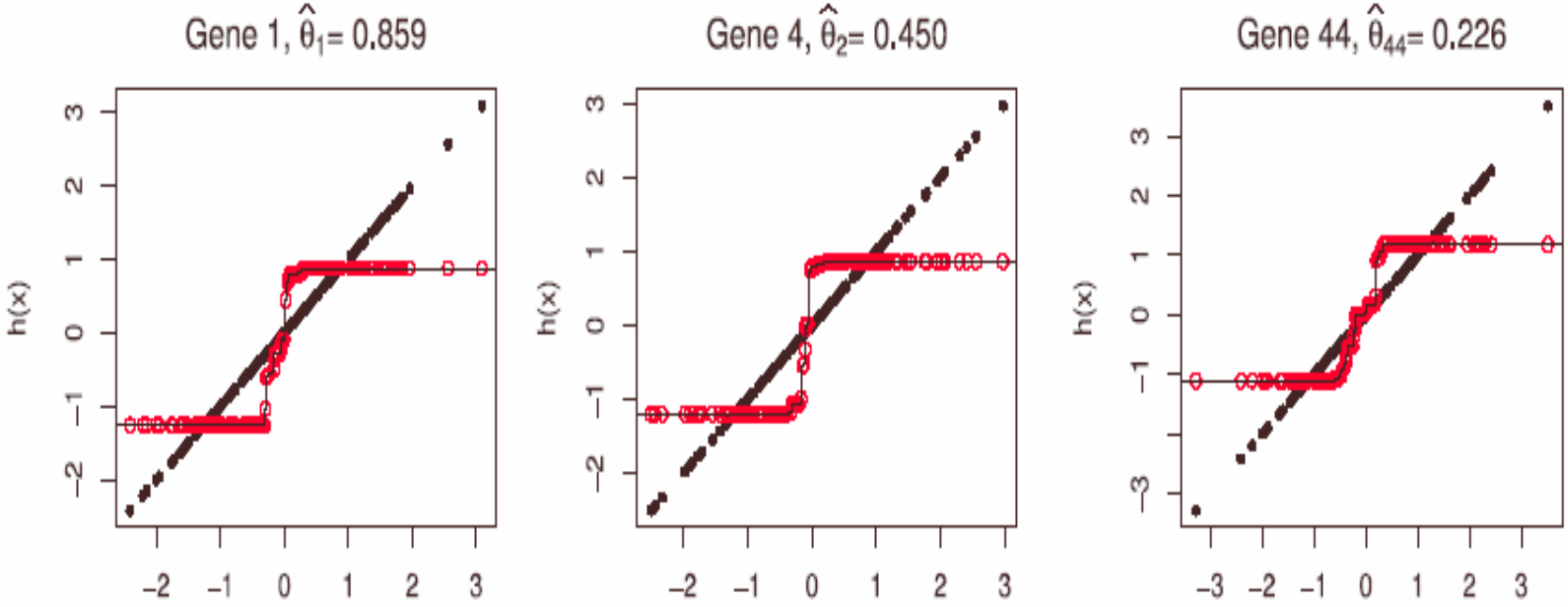


Fig. 4. BagBoosting model fit and true predictors: the black dots represent the linear univariate functions for three genes in the simulation model. Superimposed are the smoothed univariate step functions and their fitted values (grey circles), obtained by BagBoosting with stumps.

Model Recovery – Results

- Using simulations with the additive (a) response model
  - The "true" affecting genes are 1,..,10

$$F(\mathbf{x}) = \sum_{j=1}^{10} x_j$$

- Compare top 10 "important" genes to true ones

- Results:
  - Intersection: not impressive
  - Correlation: quite high
    - This is the case with highly correlated data

Table 4. Comparison of the 10 true and the 10 most important BagBoosting genes: given are their estimated model coefficients $\hat{\theta}_j$, the ranking $R(\hat{\theta}_j)$ of the coefficients according to their magnitude and the maximal correlation of each gene to 1 of the 10 genes from the other group

Important genes				True genes			
Gene	$R(\hat{\theta}_j)$	$\hat{\theta}_j$	Correlation	Gene	$R(\hat{\theta}_j)$	$\hat{\theta}_j$	Correlation
1	1	0.859	1.000	1	1	0.859	1.000
2	2	0.450	1.000	2	2	0.450	1.000
44	3	0.226	0.494	3	19	0.129	0.810
582	4	0.220	0.646	4	61	0.047	0.752
1026	5	0.217	0.623	5	66	0.041	0.864
1072	6	0.212	0.482	6	74	0.033	0.846
520	7	0.204	0.710	7	88	0.026	0.832
930	8	0.197	0.582	8	137	0.012	0.779
1894	9	0.188	0.661	9	149	0.009	0.655
261	10	0.183	0.859	10	155	0.008	0.694

Abstract

- This talk presents boosting methods and their application on DNA microarray classification. We present the theory of boosting from the computational learning perspective. We then show a statistical point of view, focusing on the relation to additive models with different loss functions.
  A brief introduction on DNA microarrays is given, followed by 2 modified applications of boosting on gene expression data. We cover the different methods and show real-world and simulation results.

- References
  - Additive Logistic Regression: A Statistical View of Boosting
    - (Friedman, Hastie & Tibshirani, 1999)
  - Boosting for tumor classification with gene expression data
    - (Dettling & Bühlmann, 2002)
  - BagBoosting for tumor classification with gene expression data
    - (Dettling, 2004)