COMS 4771 Machine Learning
Problem Set #4

Jun Hu - jh3846@columbia.edu
Discussants:

July 12, 2017

Problem 1

(a) Generate the data set as:

```
set.seed(23)
p = 40
n = 2500
x = matrix(rnorm(n*p), n, p)
b = rnorm(p)
b[5]=b[7]=b[11]=b[13]=b[17]=b[19]=b[29]=b[31]=b[37] = 0
e = rnorm(n)
y = x %*% b + e
```

(b) Split the data set as:

```
train = sample(seq(n*0.7), n*0.7, replace=FALSE)
y.train = y[train,]
y.test = y[-train,]
x.train = x[train,]
x.test = x[-train,]
data.train = data.frame(y=y.train, x=x.train)
data.test = data.frame(y=y.test, x=x.test)
```

(c) Perform boosting on the training set, and predict on test set as:

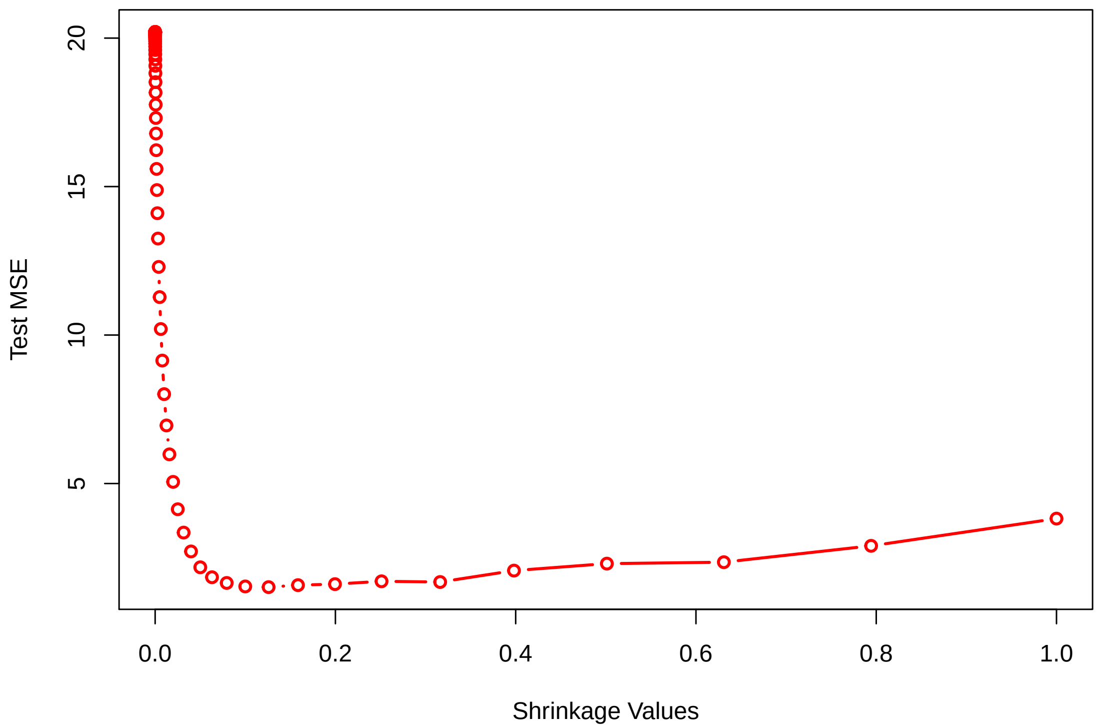
```
library(gbm)
set.seed(23)
pows = seq(-10, 0, 0.1)
lambdas = 10^pows
boost.test.err = rep(NA, length(lambdas))
for (i in 1:length(lambdas)) {
boost.data = gbm(y~., data=data.train, distribution="gaussian", n.trees
    =1000, shrinkage=lambdas[i])
pred.boost.test = predict(boost.data, data.test, n.trees=1000)
```

```
boost.test.err[i] = mean((pred.boost.test-y.test)^2)
}
```

(d) Produce the plot on corresponding test set MSE as:

```
plot(lambdas, boost.test.err, type="b", col="red", lwd=2, xlab="Shrinkage
    Values", ylab="Test MSE")
```

The plot is:



(e) Apply bagging as:

```
library(randomForest)
set.seed(23)
bag.data = randomForest(y~., data=data.train, mtry=40, importance=TRUE)
pred.bag.test = predict(bag.data, newdata=data.test)
bag.test.err = mean((pred.bag.test-y.test)^2)
```

Show the test set MSE:

```
bag.test.err
```

```
[1] 7.47377
```

The test set MSE for bagging is 7.47377.

(f) Apply random forest to the training set:

```
set.seed(23)
rf.data = randomForest(y~., data=data.train, mtry=13, importance=TRUE)
```

For the test set MSE:

```
pred.rf.test = predict(rf.data, newdata=data.test)
rf.test.err = mean((pred.rf.test-y.test)^2)
rf.test.err
```

```
[1] 7.732766
```

The test set MSE for the random forest is 7.732766.

Show importance of the variables:

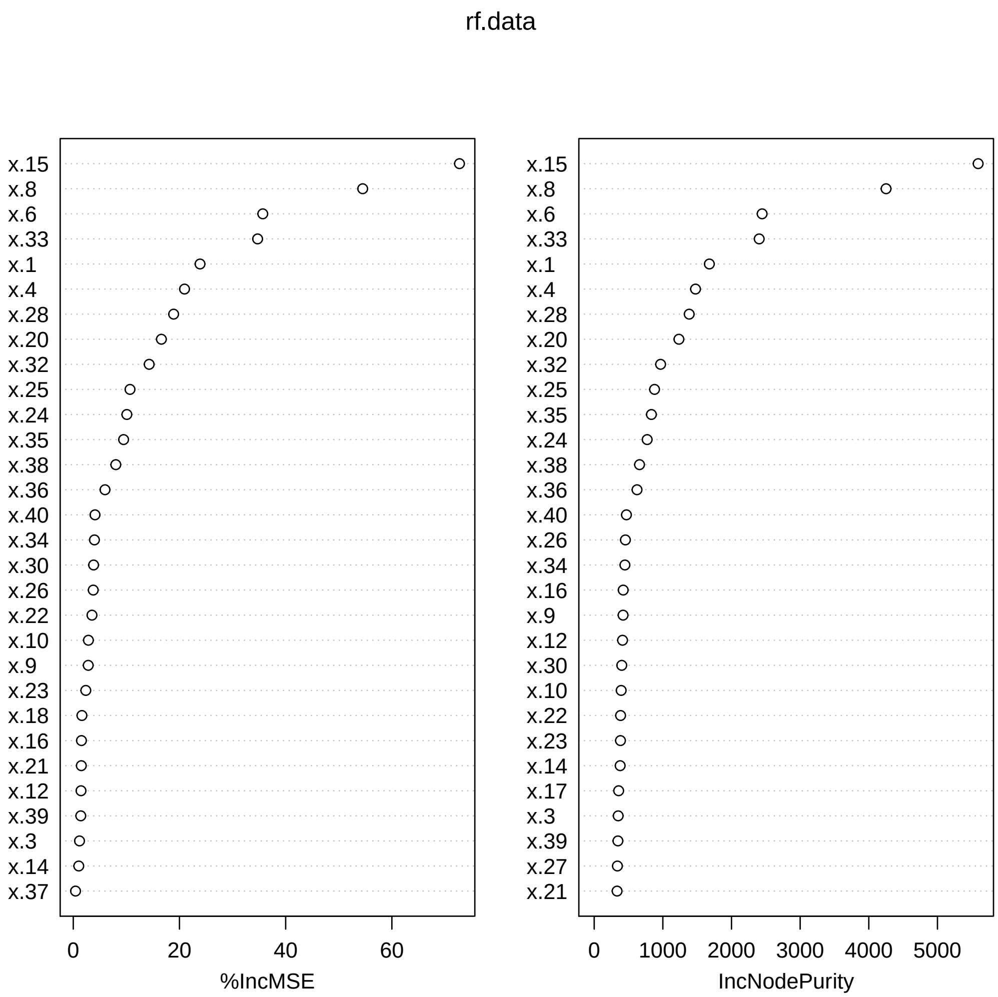
```
importance(rf.data)
```

```
          %IncMSE IncNodePurity
x.1   23.87090801     1677.6907
x.2    0.35810518      333.3559
x.3    1.17464376      349.5035
x.4   20.95290681     1475.7868
x.5   -1.34732389      316.3252
x.6   35.68869036     2446.1484
x.7   -1.55473484      316.0630
x.8   54.50399876     4251.4854
x.9    2.81823806      420.5238
x.10   2.85097449      393.0114
x.11  -0.41229165      327.6212
x.12   1.45707758      412.8536
x.13  -0.07614845      330.2071
x.14   1.03408645      378.3150
x.15  72.73041744     5592.5573
x.16   1.53779507      422.7106
x.17  -0.30807860      356.2791
x.18   1.62393917      311.9116
x.19  -1.12441079      326.3051
x.20  16.59394558     1234.1057
x.21   1.51444227      333.4318
x.22   3.52036654      384.4151
x.23   2.35868856      382.6603
x.24  10.09022068      771.6829
x.25  10.68455082      879.2054
x.26   3.75893632      454.7341
x.27  -1.09067523      338.1381
```

```
x.28 18.89861330     1383.9493
x.29  0.35818465      321.8541
x.30  3.82742307      401.3689
x.31 -1.15160979      326.7535
x.32 14.29438889      967.1681
x.33 34.71390399     2403.3239
x.34  3.98600438      447.5083
x.35  9.47775240      833.3425
x.36  5.97622152      623.8455
x.37  0.42569931      326.4059
x.38  8.00994315      660.3828
x.39  1.41219689      345.5444
x.40  4.09253787      469.6644
```

And we can plot them as:

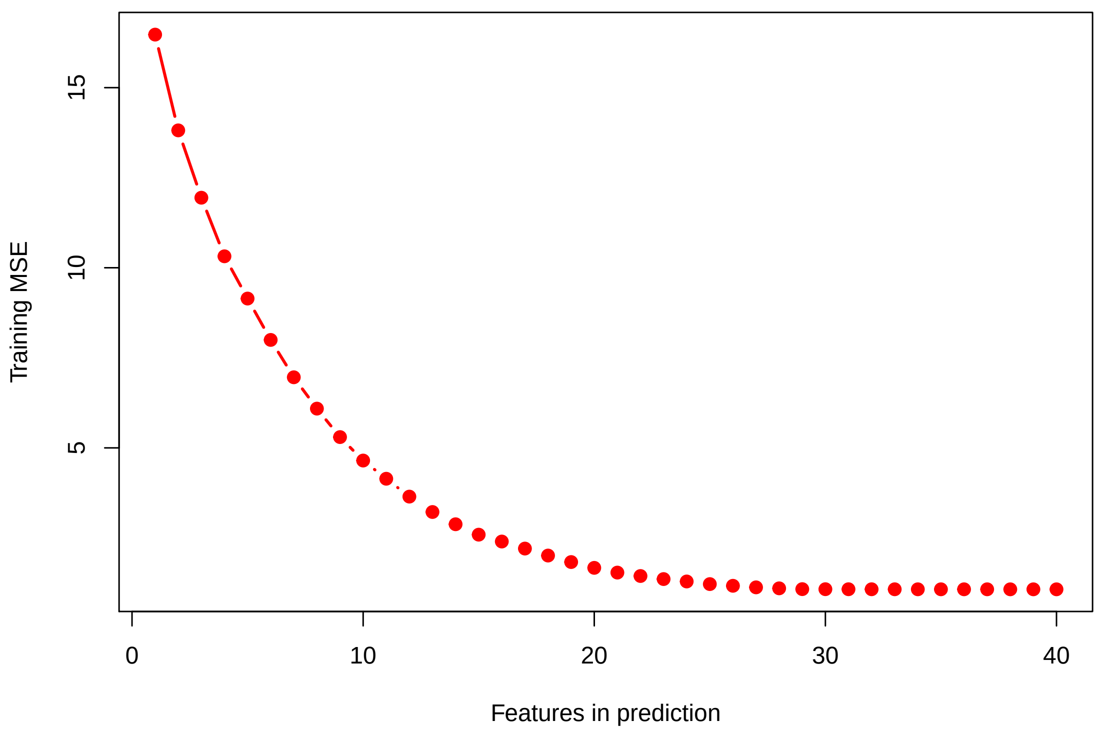
```
varImpPlot(rf.data)
```



Variables x.15 and x.8 appear to be the two most important predictors in the random forest model.

(g) Perform best subset selection on the training set, plot the training set MSE associated with the best model of each size:

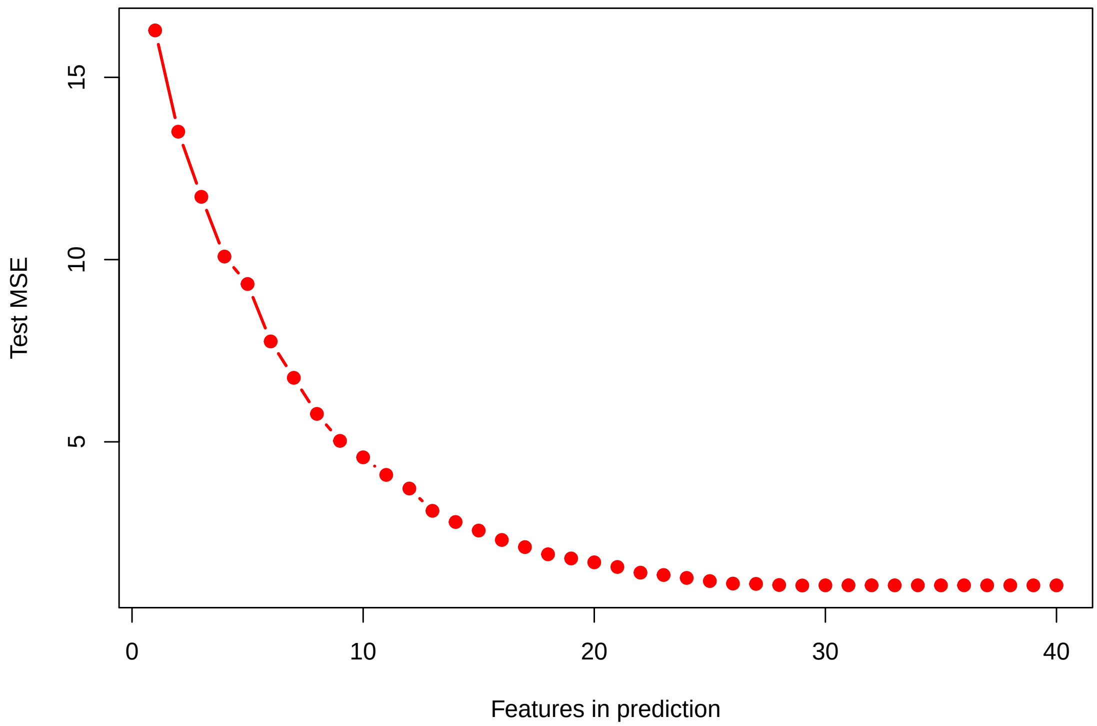
```
library(leaps)
set.seed(23)
regfit.train.err = rep(NA, 40)
regfit.data = regsubsets(y~., data=data.train, nvmax=40)
train.mat = model.matrix(y~., data=data.train, nvmax=40)
for (i in 1:40){
coefi = coef(regfit.data, id=i)
pred.regfit.train = train.mat[, names(coefi)] %*% coefi
regfit.train.err[i] = mean((pred.regfit.train-y.train)^2)
}
plot(regfit.train.err, type="b", col="red", pch=19, lwd=2, xlab="Features
    in prediction", ylab="Training MSE")
```



(h) Plot the test set MSE associated with the best model of each size:

```
set.seed(23)
regfit.test.err = rep(NA, 40)
test.mat = model.matrix(y~., data=data.test, nvmax=40)
for (i in 1:40){
coefi = coef(regfit.data, id=i)
pred.regfit.test = test.mat[, names(coefi)] %*% coefi
```

```
regfit.test.err[i] = mean((pred.regfit.test-y.test)^2)
}
plot(regfit.test.err, type="b", col="red", pch=19, lwd=2, xlab="Features
    in prediction", ylab="Test MSE")
```



(i) Check which model size has the minimum test MSE:

```
which.min(regfit.test.err)
```

```
[1] 29
```

So the model with 29 variables has the minimum test MSE.

(j) Check the coefficient values:

```
coef(regfit.data, id=29)
```

```
(Intercept)         x.1         x.2
0.00238244 -1.16720641  0.23645754
x.3         x.4         x.6
0.16699470  1.04589096 -1.44426869
```

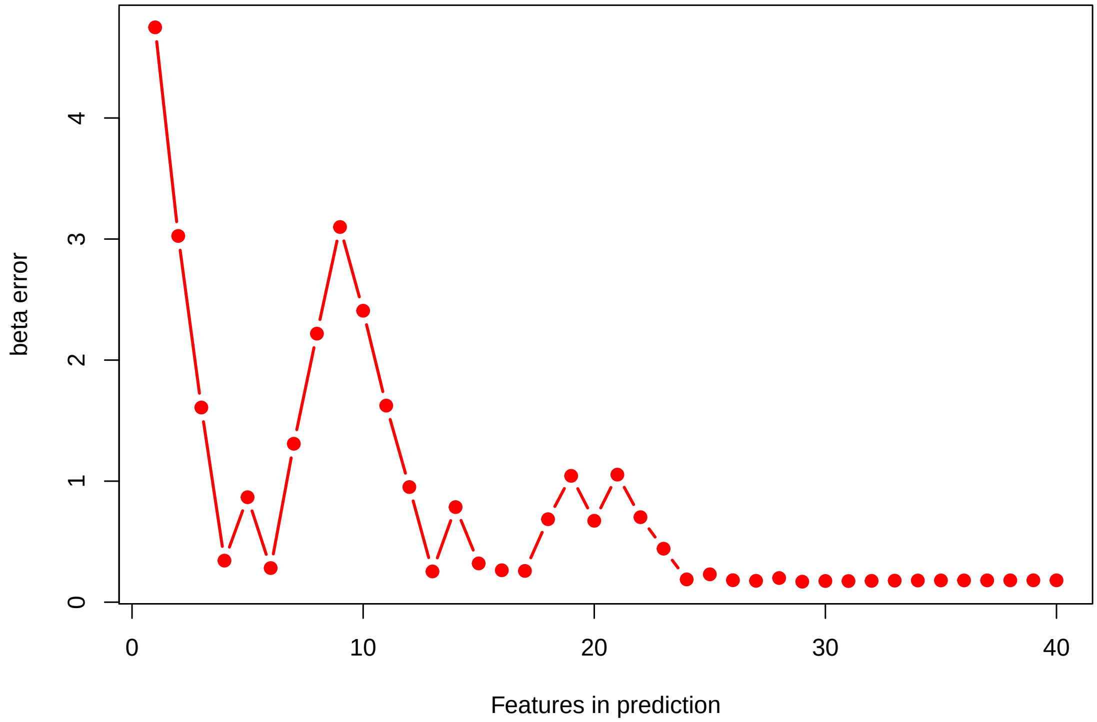
```
x.8         x.9        x.10
-1.71070751 -0.32336918 -0.28433311
x.12        x.14        x.15
-0.27297398  0.41538162 -2.03644594
x.16        x.20        x.22
-0.36922201  1.10242666 -0.26480135
x.23        x.24        x.25
0.34663161 -0.73662824 -0.65426426
x.26        x.27        x.28
-0.42748998 -0.15248357  0.89620285
x.30        x.32        x.33
0.42648698  0.92766354 -1.28265446
x.34        x.35        x.36
0.59904794 -0.75540164 -0.49063913
x.38        x.39        x.40
-0.72828204 -0.20553657  0.43477006
```

We can find that all the b elements with value 0 (b[5], b[7], b[11], b[13], b[17], b[19], b[29], b[31], b[37]) in (a) are not in the best model coefficients.

(k) Display b errors as:

```r
b.err = rep(NA, 20)
x.col = colnames(x, do.NULL=FALSE, prefix="x.")
for (i in 1:40) {
coefi = coef(regfit.data, id=i)
b.err[i] = sqrt(sum((b[x.col %in% names(coefi)]-coefi[names(coefi) %in% x
    .col])^2) + sum(b[!(x.col %in% names(coefi))])^2)
}
plot(b.err, type="b", col="red", pch=19, lwd=2, xlab="Features in
    prediction", ylab="beta error")
```

The plot is:



We can have the minimum error between the estimated and true coefficients at:

```
which.min(b.err)
```

```
[1] 29
```

So in this particular case, both the test error and the error between the estimated and true coefficients are at the model with 29 variables.

(l) Show the table as:

```
table.errs = matrix(c(boost.test.err[which.min(boost.test.err)], bag.test
    .err, rf.test.err, regfit.test.err[which.min(regfit.test.err)]), ncol
    =4, byrow=TRUE)
colnames(table.errs) = c("Boosting", "Bagging", "Random Forest", "Model
    Selection")
rownames(table.errs) = c("MSE")
table.errs
```

```
     Boosting Bagging Random Forests Model Selection
MSE 1.511832 7.47377        7.732766        1.058832
```

In this case, model selection by exhaustive search using the regsubsets() function has the minimum MSE. It maybe because some elements of b are intended for zero, so by model selection, these zero variables are neglected in the model, the errors of these useless variables are prevented from the model, so we have the minimum MSE by the best subset selection.

Problem 2

Given a two data points data set, one from each class, such that:

$$\mathbf{x_1} \in \mathcal{C}^+(t_1 = +1)$$
$$\mathbf{x_2} \in \mathcal{C}^-(t_1 = -1)$$

In order to maximize margin, there exist two constraints:

$$\mathbf{w}^{\mathrm{T}}\mathbf{x_1} + b - 1 = 0 \tag{1}$$
$$\mathbf{w}^{\mathrm{T}}\mathbf{x_2} + b + 1 = 0 \tag{2}$$

Denote λ_1 and λ_2 be the Lagrange multipliers, we can have the following Lagrange function:

$$L(\mathbf{w}, b, \lambda) = \frac{1}{2}\|\mathbf{w}\|^2 - \lambda_1(\mathbf{w}^{\mathrm{T}}\mathbf{x_1} + b - 1) - \lambda_2(\mathbf{w}^{\mathrm{T}}\mathbf{x_2} + b + 1)$$

Setting the derivatives of $L(\mathbf{w}, b, \lambda)$ with respect to $\mathbf{w}$ and b equal to zero:

$$\frac{\partial L(\mathbf{w}, b, \lambda)}{\partial \mathbf{w}} = \mathbf{w} - \lambda_1\mathbf{x_1} - \lambda_2\mathbf{x_2} = 0 \tag{3}$$
$$\frac{\partial L(\mathbf{w}, b, \lambda)}{\partial b} = -\lambda_1 - \lambda_2 = 0 \tag{4}$$

By equation **(4)**, we have:

$$\lambda_2 = -\lambda_1 \tag{5}$$

Substitute **(5)** in equation **(3)**, we have:

$$\mathbf{w} = \lambda_1(\mathbf{x_1} - \mathbf{x_2}) \tag{6}$$

By equation **(1)** and **(2)**, and substitute $\mathbf{w}$, we have:

$$\begin{aligned} b &= -\frac{1}{2}\mathbf{w}^{\mathrm{T}}(\mathbf{x_1} + \mathbf{x_2}) \\ &= -\frac{\lambda_1}{2}(\mathbf{x_1} - \mathbf{x_2})^{\mathrm{T}}(\mathbf{x_1} + \mathbf{x_2}) \\ &= -\frac{\lambda_1}{2}(\mathbf{x_1}^{\mathrm{T}} - \mathbf{x_2}^{\mathrm{T}})(\mathbf{x_1} + \mathbf{x_2}) \\ &= -\frac{\lambda_1}{2}(\mathbf{x_1}^{\mathrm{T}}\mathbf{x_1} - \mathbf{x_2}^{\mathrm{T}}\mathbf{x_2}) \end{aligned}$$

The values of $\mathbf{w}$ and b have demonstrate that, irrespective of the dimensionality of the data space, a two data points data set, one from each class, is sufficient to determine the location of the maximum-margin hyperplane.

Problem 3

By the exponential error function, we have:

$$\mathbb{E}_{\mathbf{x},t}[\exp\{-ty(\mathbf{x})\}] = \sum_t \int \exp\{-ty(\mathbf{x})\}p(t|\mathbf{x})p(\mathbf{x})\mathrm{d}\mathbf{x}$$

Perform a variational minimization with respect to all possible functions y(x), we obtain:

$$y(\mathbf{x}) = \frac{1}{2}\ln\left\{\frac{p(t=+1|\mathbf{x})}{p(t=-1|\mathbf{x})}\right\}$$

That is:

$$p(t=\pm 1|\mathbf{x}) = \frac{1}{1+e^{-2y(\mathbf{x})}}$$

Using a log likelihood of logistic regression model, which is well-behaved probabilistic model, on the $p(t|\mathbf{x})$, then we need to minimize

$$\ln(1+e^{-2\theta})$$

While, by definition, the Adaboost minimizes the average of

$$e^{-\theta}$$

Because $\ln(1+e^{-2\theta})$ is bounded to linear growth, while $e^{-\theta}$ is bounded to exponential growth. This has demonstrated that the corresponding conditional distribution $p(t|\mathbf{x})$ cannot be correctly normalized. So the above exponential error function, which is minimized by the AdaBoost algorithm, does not correspond to the log likelihood of any well-behaved probabilistic model.