EECS 545 – Machine Learning - HW#6 Solution

Changhan (Aaron) Wang Due: 11:00pm 04/18/2016

Homework Policy: Working in groups is fine, but each member must submit their own writeup. Please write the members of your group on your solutions. There is no strict limit to the size of the group but we may find it a bit suspicious if there are more than 4 to a team. **For coding problems, please include your code and report your results (values, plots, etc.)** in your PDF submission. You will lose points if your experimental results are only accessible through rerunning your code. Homework will be submitted via Gradescope (https://gradescope.com/).

1) **Principal Components Analysis (20 pts).**

(a) Let $\boldsymbol{w}_{K+1}, \cdots, \boldsymbol{w}_D$ extend $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_K$ to an orthonormal basis. Denote $M = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_D]$, then we have $M^T M = I_D$, and

$$0 \le h_j = \sum_{k=1}^{K} (\boldsymbol{w}_k^{(j)})^2 \le \sum_{k=1}^{D} (\boldsymbol{w}_k^{(j)})^2 = 1$$

Besides,

$$\sum_{j=1}^{D} h_j = \sum_{j=1}^{D}\sum_{k=1}^{K} (\boldsymbol{w}_k^{(j)})^2 = \sum_{k=1}^{K}\sum_{j=1}^{D} (\boldsymbol{w}_k^{(j)})^2 = \sum_{k=1}^{K} 1 = K$$

(b)

$$\begin{aligned}
\max_{\substack{W\\ W^TW=I_k}} \sum_{k=1}^{K} \boldsymbol{w}_k^T \Lambda \boldsymbol{w}_k &= \max_{\substack{W\\ W^TW=I_k}} \sum_{k=1}^{K}\sum_{j=1}^{D} \lambda_j (\boldsymbol{w}_k^{(j)})^2 \\
&= \max_{\substack{W\\ W^TW=I_k}} \sum_{j=1}^{D} \lambda_j \sum_{k=1}^{K} (\boldsymbol{w}_k^{(j)})^2 \\
&= \max_{\substack{h_j\\ W^TW=I_k}} \sum_{j=1}^{D} h_j \lambda_j
\end{aligned}$$

(c) Since $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_D \ge 0$, the optimal h_j are

$$h_j = \begin{cases} 1 & j = 1, \cdots, K \\ 0 & j = K+1, \cdots, D \end{cases}$$

It is easy to verify that $\boldsymbol{a}_k = \boldsymbol{u}_k$ $(k = 1, \cdots, K)$ is a feasible solution and leads to optimal h_j.

(d) For a set of optimal h_j, W must be in the following form:

$$W = \begin{bmatrix} \tilde{W} \\ N \end{bmatrix}$$

where $\tilde{W}$ is a $K \times K$ matrix and N is a $(D-K) \times K$ zero matrix. Moreover, $\tilde{W}^T\tilde{W} = I_K$. Hence $\tilde{W}^T = \tilde{W}^{-1}$ and $\tilde{W}\tilde{W}^T = I_K$, i.e., row vectors in $\tilde{W}$ form an orthonormal basis (and they can be any orthonormal basis). Hence the solution is not unique. For $\tilde{W}_1$ and $\tilde{W}_2$ in any 2 solutions, $\tilde{W}_2 = \tilde{W}_1(\tilde{W}_1^T\tilde{W}_2)$, i.e., column vectors of $\tilde{W}_2$ are linear combinations of that of $\tilde{W}_1$ (with transformation matrix $\tilde{W}_1^T\tilde{W}_2$). Thus, column vectors of $\tilde{W}_1$ and $\tilde{W}_2$ span the same subspace. As a result, column vectors of W_1 and W_2 span the same subspace as well.

However, the set of optimal h_j is not unique when $\lambda_K = \lambda_{K+1}$. And subspaces spanned by the solutions from different sets of h_j are different. That's because their vectors have non-zero entries from different dimensions of the D-dimensional vectors $\boldsymbol{w}_i$. With the above analysis, it is easy to verify that $\lambda_K > \lambda_{K+1}$ is a necessary and sufficient condition for the uniqueness of the subspace spanned by the solution.

(e) We have

$$\boldsymbol{a}_k^T S \boldsymbol{a}_k = \boldsymbol{w}_k^T \Lambda \boldsymbol{w}_k = \sum_{j=1}^{K} \lambda_j (\boldsymbol{w}_k^{(j)})^2 = \sum_{j=k}^{K} \lambda_j (\boldsymbol{w}_k^{(j)})^2 \tag{1}$$

where $\boldsymbol{w}_k^{(j)} = \boldsymbol{u}_j^T \boldsymbol{a}_k$. To maximize (1), we must have $(\boldsymbol{w}_k^{(j)})^2 = (\boldsymbol{u}_j^T \boldsymbol{a}_k)^2 = \begin{cases} 1 & j = k \\ 0 & \text{otherwise} \end{cases}$. And it is easy to verify that $\boldsymbol{a}_k = \boldsymbol{u}_k$ is a solution.

2) **Multi-layer Neural Network (20 pts).**

(a) According to the setting, we have

$$\begin{aligned} E(\mathbf{W}, \mathbf{S}, \mathbf{A}) &= -\sum_{n=1}^{N} 1(t = n) \log(P_n) \\ &= -\sum_{n=1}^{N} 1(t = n) \log\left(\frac{\exp(\sum_{j=1}^{D} w_{n,j} z_j + w_{n,D+1})}{\sum_{k=1}^{N} \exp(\sum_{j=1}^{D} w_{k,j} z_j + w_{k,D+1})} \right) \\ &= \log\left(\sum_{k=1}^{N} \exp(\sum_{j=1}^{D} w_{k,j} z_j + w_{k,D+1}) \right) - \sum_{n=1}^{N} 1(t = n) \left(\sum_{j=1}^{D} w_{n,j} z_j + w_{n,D+1} \right) \end{aligned}$$

For $n \in 1, 2, ..., N$ and $d \in 1, 2, ..., D$, we have

$$\begin{aligned} \frac{\partial}{\partial w_{n,d}} E(\mathbf{W}, \mathbf{S}, \mathbf{A}) &= \frac{\partial}{\partial w_{n,d}} \log\left(\sum_{k=1}^{N} \exp(\sum_{j=1}^{D} w_{k,j} z_j + w_{k,D+1}) \right) \\ &\quad - \frac{\partial}{\partial w_{n,d}} \sum_{k=1}^{N} 1(t = k) \left(\sum_{j=1}^{D} w_{k,j} z_j + w_{k,D+1} \right) \\ &= \frac{\exp(\sum_{j=1}^{D} w_{n,j} z_j + w_{n,D+1})}{\sum_{k=1}^{N} \exp(\sum_{j=1}^{D} w_{k,j} z_j + w_{k,D+1})} z_d - 1(t = n) z_d \\ &= (P_n - 1(t = n)) z_d \end{aligned}$$

Similarly, $\frac{\partial}{\partial w_{n,D+1}} E(\mathbf{W},\mathbf{S},\mathbf{A}) = P_n - 1(t=n)$.

Further, for $d \in 1,2,...,D$ we have

$$\begin{aligned}
\frac{\partial}{\partial z_d} E(\mathbf{W},\mathbf{S},\mathbf{A}) &= \frac{\partial}{\partial z_d} \log\left(\sum_{k=1}^{N} \exp(\sum_{j=1}^{D} w_{k,j} z_j + w_{k,D+1})\right) \\
&\quad - \frac{\partial}{\partial z_d} \sum_{k=1}^{N} 1(t=k)\left(\sum_{j=1}^{D} w_{k,j} z_j + w_{k,D+1}\right) \\
&= \frac{\sum_{k=1}^{N} \frac{\partial}{\partial z_d} \exp(\sum_{j=1}^{D} w_{k,j} z_j + w_{k,D+1})}{\sum_{k=1}^{N} \exp(\sum_{j=1}^{D} w_{k,j} z_j + w_{k,D+1})} - \sum_{k=1}^{N} 1(t=k) w_{k,d} \\
&= \frac{\sum_{k=1}^{N} \exp(\sum_{j=1}^{D} w_{k,j} z_j + w_{k,D+1}) w_{k,d}}{\sum_{k=1}^{N} \exp(\sum_{j=1}^{D} w_{k,j} z_j + w_{k,D+1})} - \sum_{k=1}^{N} 1(t=k) w_{k,d} \\
&= \sum_{k=1}^{N} (P_k - 1(t=k)) w_{k,d}
\end{aligned}$$

For $\mathbf{S}$ and $d \in 1,2,...,D$ and $k \in 1,2,...,K$ we have

$$\begin{aligned}
\frac{\partial}{\partial s_{d,k}} E(\mathbf{W},\mathbf{S},\mathbf{A}) &= \frac{\partial}{\partial z_d} E(\mathbf{W},\mathbf{S},\mathbf{A}) \frac{\partial}{\partial s_{d,k}} z_d \\
&= \frac{\partial}{\partial z_d} E(\mathbf{W},\mathbf{S},\mathbf{A}) * z_d * (1-z_d) * y_k
\end{aligned}$$

Similarly, we have $\frac{\partial}{\partial s_{d,K+1}} E(\mathbf{W},\mathbf{S},\mathbf{A}) = \frac{\partial}{\partial z_d} E(\mathbf{W},\mathbf{S},\mathbf{A}) * z_d * (1-z_d)$.

Further, for $k \in 1,2,...,K$ we have

$$\begin{aligned}
\frac{\partial}{\partial y_k} E(\mathbf{W},\mathbf{S},\mathbf{A}) &= \sum_{d=1}^{D} \frac{\partial}{\partial z_d} E(\mathbf{W},\mathbf{S},\mathbf{A}) \frac{\partial}{\partial y_k} z_d \\
&= \sum_{d=1}^{D} \frac{\partial}{\partial z_d} E(\mathbf{W},\mathbf{S},\mathbf{A}) * z_d * (1-z_d) s_{d,k}
\end{aligned}$$

Last, for $k \in 1,2,...,K$ and $m \in 1,2,...,M$ we have

$$\begin{aligned}
\frac{\partial}{\partial a_{k,m}} E(\mathbf{W},\mathbf{S},\mathbf{A}) &= \frac{\partial}{\partial y_k} E(\mathbf{W},\mathbf{S},\mathbf{A}) \frac{\partial}{\partial a_{k,m}} y_k \\
&= \frac{\partial}{\partial y_k} E(\mathbf{W},\mathbf{S},\mathbf{A}) * y_k * (1-y_k) * x_m
\end{aligned}$$

Similarly, we have $\frac{\partial}{\partial a_{k,M+1}} E(\mathbf{W},\mathbf{S},\mathbf{A}) * y_k * (1-y_k)$.

3) **Open Kaggle Challenge (20 pts).**

4) **AdaBoost (20 pts).**

(a) Assuming points on the line are classified as negative, we have two misclassified examples. In this case,

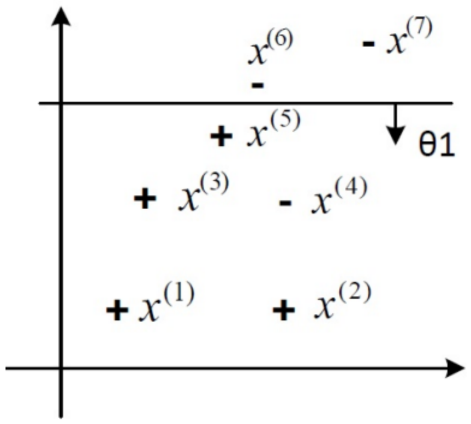
$$\alpha_1 = \log(\sqrt{\frac{5}{2}})$$

$$\tilde{W}_1(i) = \begin{cases} \frac{1}{4} & i \text{ is misclassified} \\ \frac{1}{10} & i \text{ is correctly classified} \end{cases}$$

Note that, you will get a different answer if you assume that, the points on the line are classified as positive or they are misclassified.

(b)

We will assume a numbering of the points as in the following figures. First, we initialize the weights $W_0(i) = 1, \forall i$. In the first $m = 1$ iteration, the AdaBoost algorithm adds a stump that minimizes the weighted training error ϵ_m. Let $h(\bar{x}; \theta_1) = sgn(-x_2 + 0.185)$, this will result in only $x^{(4)}$ being misclassified as in the following figure.



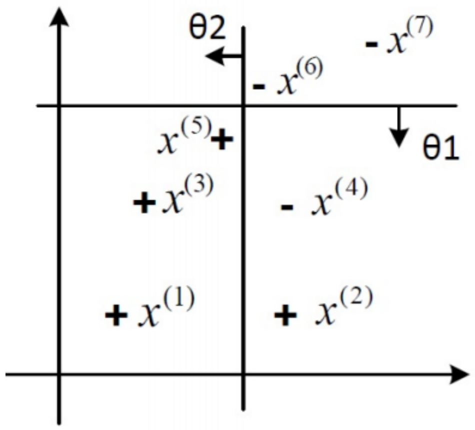
Now we have,

$$\epsilon_1 = \frac{1}{7}$$

$$\alpha_1 = \log(\sqrt{6})$$

$$\tilde{W}_1(i) = \begin{cases} \frac{1}{2} & i \in \{4\} \\ \frac{1}{12} & i \in \{1, 2, 3, 5, 6, 7\} \end{cases}$$

Notice that we haven't classified all points correctly yet and as a result we need to add a second stump in a way that minimizes the weighted training error ϵ_m. The weights for misclassified points have increased, and therefore to minimize ϵ_m we need to correctly classify the points that were misclassified in the previous iteration. So, AdaBoost algorithm puts more emphasis on misclassified examples. Let $h(\bar{x}; \theta_2) = sgn(-x_1 + 0.21)$, this will correctly classify $x^{(4)}$ and will result in only $x^{(2)}$ being misclassified as in the following figure.



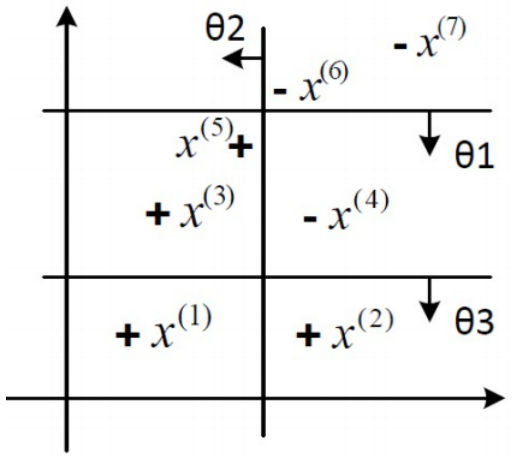
Now we have,

$$\epsilon_2 = \frac{1}{12}$$

$$\alpha_2 = \log(\sqrt{11})$$

$$\tilde{W}_2(i) = \begin{cases} \frac{3}{11} & i \in \{2\} \\ \frac{1}{2} & i \in \{4\} \\ \frac{1}{22} & i \in \{1, 3, 5, 6, 7\} \end{cases}$$

Now use the ensemble classifier to check that we misclassify $x^{(2)}$. So, we need to make sure we classify $x^{(2)}$ correctly in the next iteration. Let $h(\bar{x}; \theta_3) = sgn(-x_2 + 0.16)$ as in the following figure.



Now we have,

$$\epsilon_3 = \frac{1}{11}$$

$$\alpha_3 = \log(\sqrt{10})$$

$$\tilde{W}_3(i) = \begin{cases} \frac{3}{20} & i \in \{2\} \\ \frac{11}{40} & i \in \{4\} \\ \frac{1}{4} & i \in \{3, 5\} \\ \frac{1}{40} & i \in \{1, 6, 7\} \end{cases}$$

Now, the ensemble classifier classifies all points correctly. Notice that, the exponential loss is non-zero even for correctly classified examples:

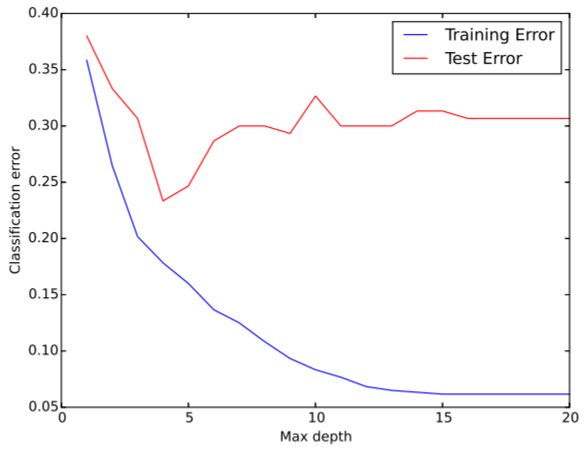
$$L(h_m) = \exp(\sum_{t=1}^{n} -y^{(t)} h_m(\bar{x}^{(t)})) \approx 1.5570$$

Note that, you may have different results based on your choice of decision stumps.

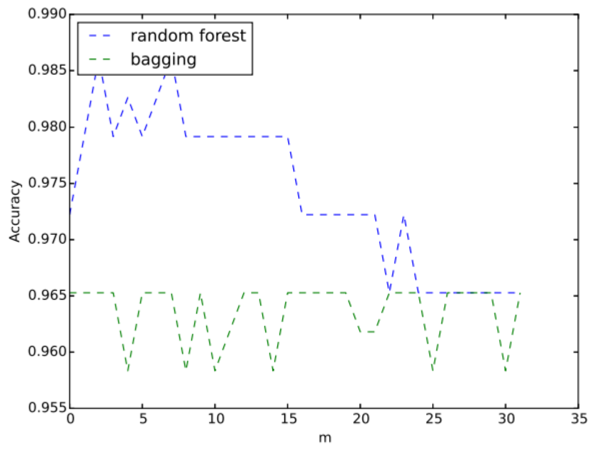
(c) The exponential loss function encourages the classifier to make confident decisions: the loss is exponentially high if the ensemble $h_m(x^{(i)})$ strongly disagrees with the label $y^{(i)}$ and it decreases exponentially as the agreement $y^{(i)} h_m(x^{(i)})$ increases. The exponential loss is non-zero even for correctly classified examples. Thus, we can use it to compare two different ensembles that both correctly classify the training data. Lower exponential loss means the classifier is more confident in its decision and should have less generalization error.

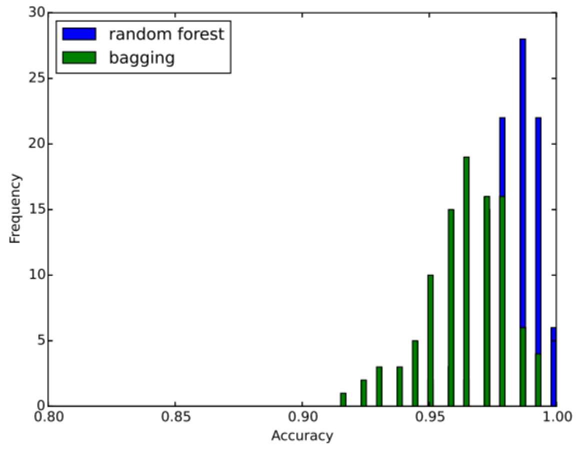
5) **Decision trees and random forest (20 pts).**

(a) The classifier performs best for `max_depth = 4`.



(b) Note that as m approach 64 the total number of features in the dataset the two methods become equivalent. The best m appears to be 8. Here, Random Forest slightly outperforms Bagging on average.


random forest
bagging
Accuracy
m
0.990
0.985
0.980
0.975
0.970
0.965
0.960
0.955
0
5
10
15
20
25
30
35


random forest
bagging
Frequency
Accuracy
30
25
20
15
10
5
0
0.80
0.85
0.90
0.95
1.00