CPS 571 — HW 3

Shengxin Qian, sq16

1 Convex Hull

Assume two sets of points $\{\boldsymbol{x}_n\}$ and $\{\boldsymbol{y}_m\}$ is linearly separable, so:

$$\boldsymbol{w}^T\boldsymbol{x}_n + \omega_o > 0, \forall \boldsymbol{x}_n \in \{\boldsymbol{x}_n\} \tag{1}$$

$$\boldsymbol{w}^T\boldsymbol{y}_m + \omega_o < 0, \forall \boldsymbol{y}_m \in \{\boldsymbol{y}_m\} \tag{2}$$

We can derive the following equations from (1), (2):

$$\sum_n a_n \boldsymbol{w}^T\boldsymbol{x}_n + \omega_o > 0, \forall \boldsymbol{x}_n \in \{\boldsymbol{x}_n\}, a_n \geq 0 \; and \; \sum_n a_n = 1 \tag{3}$$

$$\sum_m a_m \boldsymbol{w}^T\boldsymbol{y}_m + \omega_o < 0, \forall \boldsymbol{y}_m \in \{\boldsymbol{y}_m\}, a_m \geq 0 \; and \; \sum_m a_m = 1 \tag{4}$$

Equations (3) and (4) are same as:

$$\boldsymbol{w}^T \sum_n a_n \boldsymbol{x}_n + \omega_o > 0, \forall \boldsymbol{x}_n \in \{\boldsymbol{x}_n\}, a_n \geq 0 \; and \; \sum_n a_n = 1 \tag{5}$$

$$\boldsymbol{w}^T \sum_m a_m \boldsymbol{y}_m + \omega_o < 0, \forall \boldsymbol{y}_m \in \{\boldsymbol{y}_m\}, a_m \geq 0 \; and \; \sum_m a_m = 1 \tag{6}$$

Based on the definition of convex hull and the condition that two convex hulls have intersect, there must exist at least one point $\boldsymbol{z}$ which belongs to both of them.

Because $\boldsymbol{z}$ belongs to convex hull of $\{\boldsymbol{x}_n\}$:

$$\boldsymbol{w}^T\boldsymbol{z} + \omega_o > 0, \forall \boldsymbol{x}_n \in \{\boldsymbol{x}_n\} \tag{7}$$

Also because $\boldsymbol{z}$ belongs to convex hull of $\{\boldsymbol{y}_m\}$:

$$\boldsymbol{w}^T\boldsymbol{z} + \omega_o < 0, \forall \boldsymbol{y}_m \in \{\boldsymbol{y}_m\} \tag{8}$$

Obviously, equations (7) and (8) conflict. So, if the convex hulls of any sets of data points $\{\boldsymbol{x}_n\}$ and $\{\boldsymbol{y}_m\}$ intersect, two sets can not be linearly separable

2 "Cost sensitive" Adaboost

Because we change the weight of each examples, the new exponential loss function is:

$$R^{train}(\lambda) = \frac{2}{n+10}\sum_{i=1}^{10} e^{(M\lambda)_i} + \frac{1}{n+10}\sum_{i>10} e^{-(M\lambda)_i} \tag{9}$$

Then we do the coordinate descent:

$$\begin{aligned} j_t &\in argmax_j \left[\frac{\partial R^{train}(\lambda_t + \alpha e_j)}{\partial \alpha} \mid_{\alpha=0}\right] \\ &= argmax_j \left[\frac{2}{n+10}\sum_{i=1}^{10} M_{ij} e^{-(M\lambda_t)_i} + \frac{1}{n+10}\sum_{i>10} M_{ij} e^{-(M\lambda_t)_i}\right] \end{aligned} \tag{10}$$

We define $d_{t,i}$ as:

$$d_{t,i} = e^{-(M\lambda_t)_i}/Z_t, \ where \ Z_t = \sum_{i=1}^{n} e^{-(M\lambda_t)_i} \tag{11}$$

To get $argmax_j$, set derivative to 0

$$\begin{aligned} 0 = &-\frac{2}{n+10}\sum_{i=1,M_{i,j}=1}^{10} M_{ij} e^{-(M\lambda_t)_i} + \frac{2}{n+10}\sum_{i=1,M_{i,j}=-1}^{10} M_{ij} e^{-(M\lambda_t)_i} \\ &-\frac{1}{n+10}\sum_{i>10,M_{i,j}=1} M_{ij} e^{-(M\lambda_t)_i} + \frac{1}{n+10}\sum_{i>10,M_{i,j}=-1} M_{ij} e^{-(M\lambda_t)_i} \end{aligned} \tag{12}$$

After get rid of denominator:

$$\begin{aligned} (2d_{+,\leq 10} + d_{+,>10})e^{-\alpha t} &= (2d_{-,\leq 10} + d_{-,>10})e^{\alpha t} \\ \alpha_t &= \frac{1}{2}\ln\frac{2d_{+,\leq 10} + d_{+,>10}}{2d_{-,\leq 10} + d_{-,>10}} \end{aligned} \tag{13}$$

So, the new AdaBoost algorithm is:
Define cost variable:

$$g_i = \begin{cases} 2, \ i \in [1,10] \\ 1, \ i > 10 \end{cases}$$

Assign observation i the weight of d_{1i} is:

$d_{1i} = g_i/(n+10)$
$loop \ t = 1...T$
$choose \ h_j(x) \ with \ lowest \ weighted \ error$:
$d_- = \sum_{M_{i,j_t}=-1} g_i d_{t,i}$
$\alpha_t = \frac{1}{2}\ln(\frac{1-d_-}{d_-})$
$d_{t+1,i} = d_{t,i}e^{-M_{ij_t}\alpha_t}/Z_{t+1}, \ where \ Z_{t+1} \ is \ a \ normalization \ factor$
end

3 Margin Matrix of AdaBoost

3.1 When weak learning assumption holds

Adaboost could always classify this dataset perfectly if weak learning assumption holds. Because in the worst case we could overfit the dataset by creating tiny blocks for each sample and making the label of the block consistent with that of each sample. Rough proof: Because Adaboost misclassification error decays exponentially fast:

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}_{[y_i\neq H(x_i)]} \leq e^{-2\gamma_{WLA}^2 T} \tag{14}$$

If weak learning assumption holds, $\gamma_{WLA}^2 > 0$ and the misclassification error will eventually decline to zero.

3.2 When weak learning assumption does not holds

Adaboost may not perfectly separate the dataset without weak learning assumption. For example:
Given 2D dataset $\{\boldsymbol{x}_n\}$, where $\boldsymbol{x}_n = [x_1, x_2],\ n \in [1, 2]$
The $h_1(\boldsymbol{x})$ classifier choose $x_1 = 0$ as threshold.

$$h_1(\boldsymbol{x}) = \begin{cases} 1,\ x_1 \geq 0 \\ -1,\ x_1 < 0 \end{cases} \tag{15}$$

The $h_2(\boldsymbol{x})$ classifier choose $x_2 = 0.5$ as threshold.

$$h_2(\boldsymbol{x}) = \begin{cases} 1,\ x_2 \geq 0.5 \\ -1,\ x_2 < 0.5 \end{cases} \tag{16}$$

sample points	label	h_1	$-h_1$	h_2	$-h_2$
[-1,1]	1	-1	1	1	-1
[-1,1]	-1	1	-1	-1	1

Table 1: symmetric matrix of margins

Because the points in the dataset are overlapped, there is no way we can find a classifier which is better than random guess. So, obviously, we can not separate the dataset perfectly. Actually, the symmetric classifier does not help because it does not input any new information.

4 Empirical Comparision

In this section, the dataset I choose is "Tic-Tac-Toe Endgame Data Set" on UCI machine learning repository. The target of the classification is to anticipate the winner of Tic-Tac-Toe game when the game reaches the end. The input information is configuration of the board at the end. The attribute information is:

(x=player x has taken, o=player o has taken, b=blank)
1. top-left-square: x,o,b
2. top-middle-square: x,o,b
3. top-right-square: x,o,b
4. middle-left-square: x,o,b
5. middle-middle-square: x,o,b
6. middle-right-square: x,o,b
7. bottom-left-square: x,o,b
8. bottom-middle-square: x,o,b
9. bottom-right-square: x,o,b
10. Class: positive,negative

In the data preprocess stage, the algorithm transforms the categorical value into dummy label($x => 1,\ o => -1,\ b => 0, positive => 1, negative => 0$).

In empirical comparison, I choose 5 different machine learning algorithms. They are Boosted Trees, CART, Logistic Regression, Random Forest and SVM. I use nested 10-fold cross validation to determine "number of trees" parameter of Random Forest and Boosted Trees. In the nested cross validation, I choose {5, 15, 30} three potential parameters. The experiment result shows when the number of trees is 30, Random Forest and Boosted Trees have best performance. For the SVM classifier, the kernel function I chose is radial basis function kernel.

4.1 ROC Curves of Random Forest and Boosted Trees for nested 10 test folds

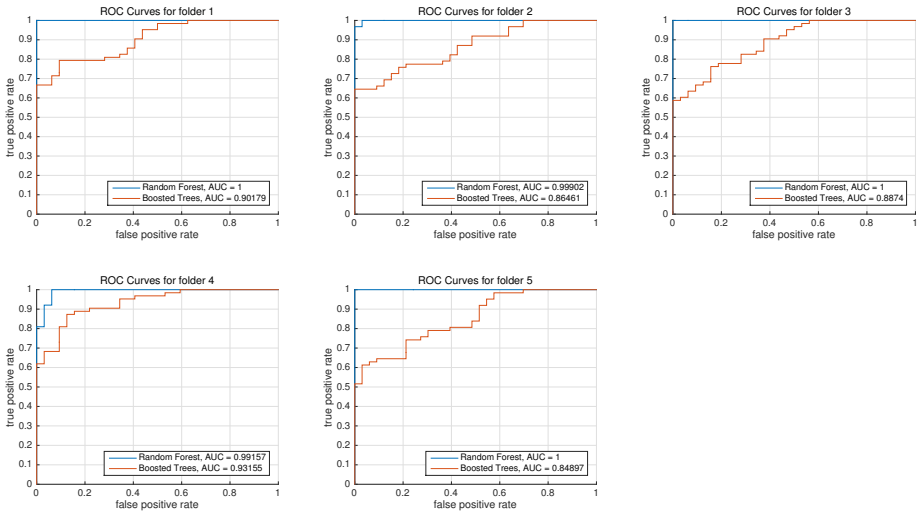

Figure 1: 10-fold nested cross validation for Random Forest and Boosted Trees 1-5 folds

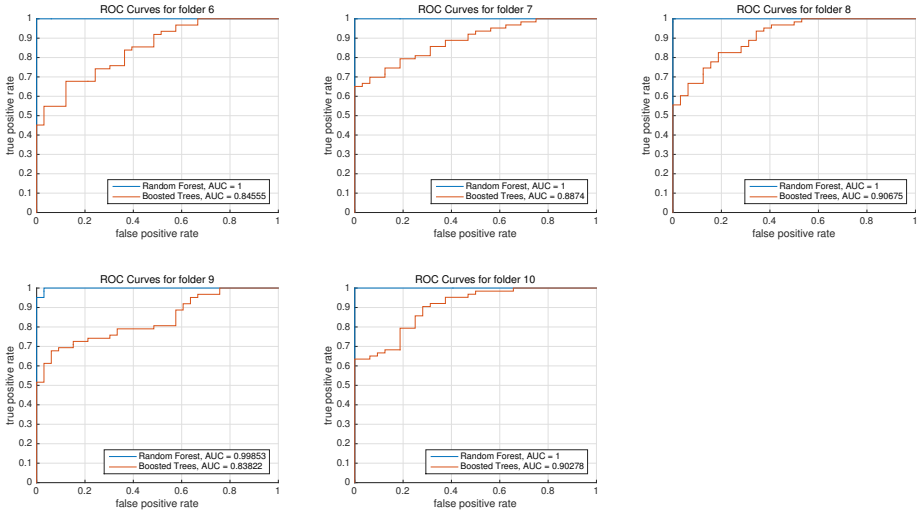

Figure 2: 10-fold nested cross validation for Random Forest and Boosted Trees 6-10 folds

4.2 ROC for each individual feature on 10 test folds

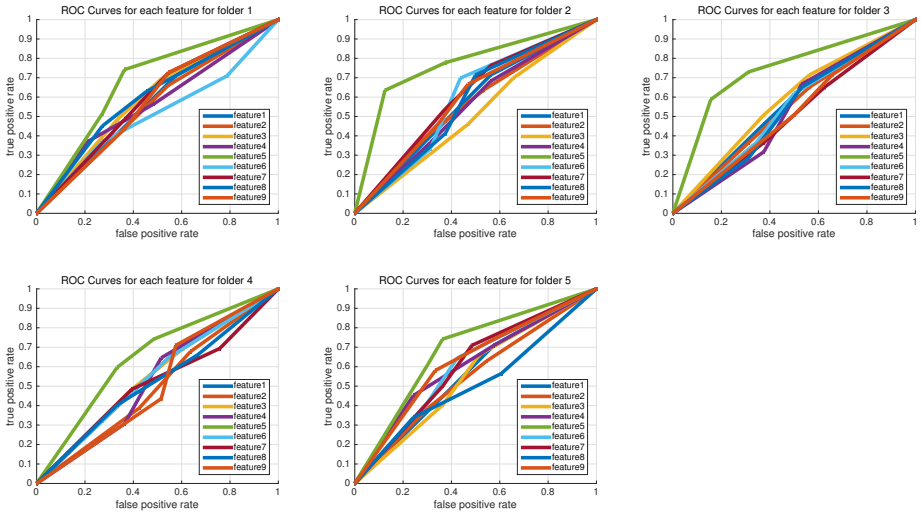

Figure 3: ROC for feature 1-5 on 10 test folds

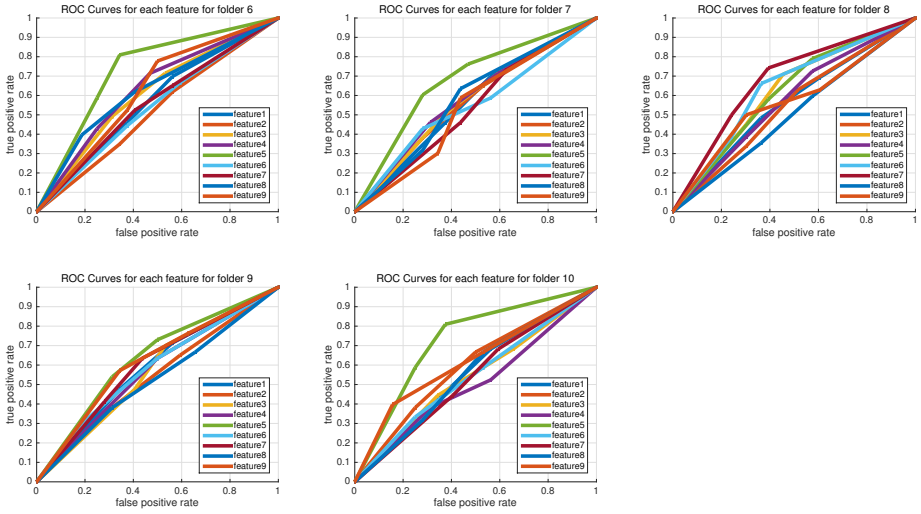

Figure 4: ROC for feature 6-10 on 10 test folds

4.3 ROC Curves of each algorithm for 10 test folds

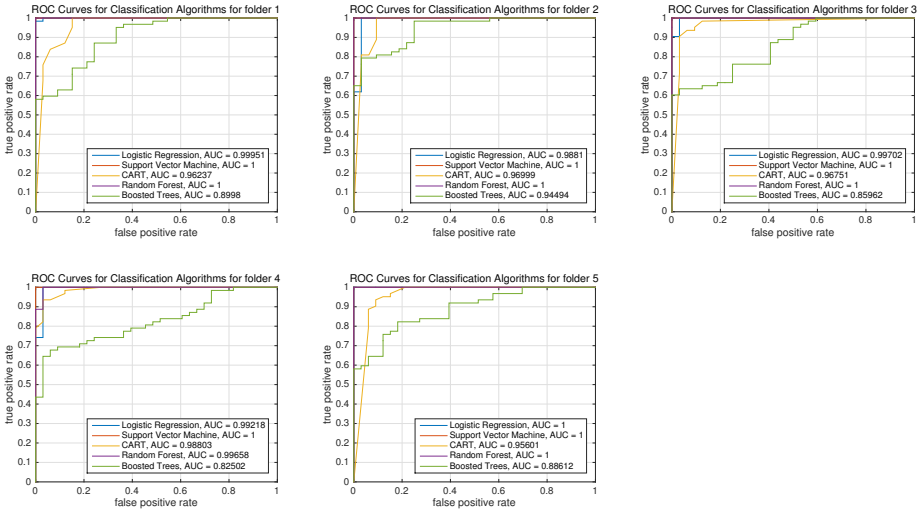

Figure 5: 10-fold cross validation for different classifiers 1-5 folds

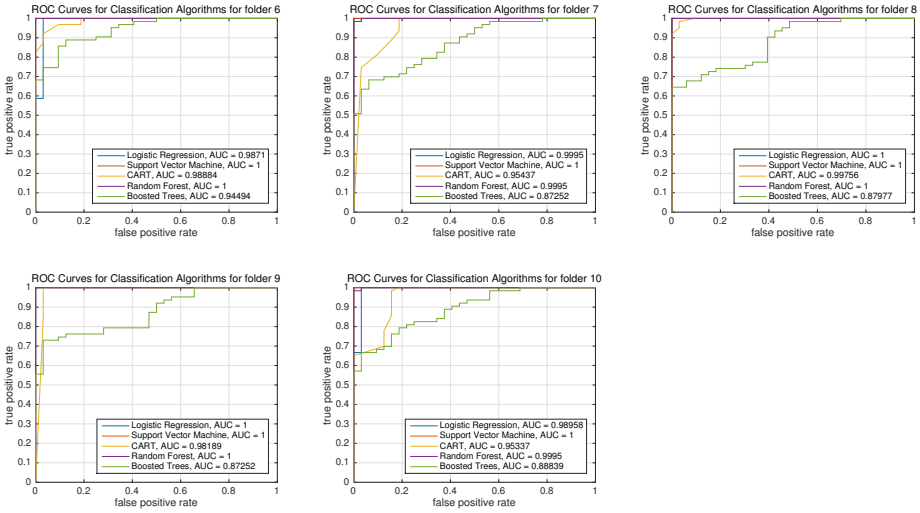

Figure 6: 10-fold cross validation for different classifiers 6-10 folds

As we can see in Figure.5 and Figure.6, 5 different algorithms perform pretty well. Compared with Figure.3 and Figure.4, combining classifiers and multiple features could obtain higher accuracy than classification with each individual feature. The ROC curves of Figure.5 and Figure.6 are more close to the top left corner than that of Figure.3 and Figure.4. This result does make sense because the points in the end of the Tic-Tac-Toe board configuration space is clearly easy to be separated. In the end game board space, only one player have

three continues filling could win the other. The three continues fillings in the board space would add some critical information help classifier to classify. On the contrary, if we only have one board position information, even though some position is more important than the other according to the rule of Tic-Tac-Toe game, but this does not help a lot. Obviously, the more board filling information we know, the more we know about the state of the game. With all 9 position information, it is supposed to be a perfect classifier exist. This also explain why SVM has perfect result.

4.4 Mean and standard deviation over 10 folds

Statistics	Boosted Trees	CART	Logistic Regression	Random Forest	SVM
mean	0.8874	0.9720	0.9953	0.9996	1
standard deviation	0.0364	0.0161	0.0054	0.0011	0

Table 2: mean and standard deviation over the 10 folds

As we can see in the table.2, even though all algorithms perform well, there is still some noticeable performance difference. The reason of Boosted Trees worst performance could be the number of trees I chose is lower than the optimal.