Data Mining CS573: Midterm

March 11, 2016

Yu-Chen Chang

Contents

Problem 1

a.

i. Naive Bayes Classifier:

From the naive bayes classifier, we have the formula:
$P(C|\mathbf{X}) = \frac{P(\mathbf{X}|C)P(C)}{P(\mathbf{X})} \propto P(\mathbf{X}|C)P(C)$ (Bayes rule), where C is the class random variable and $\mathbf{X}$ is the attribute random vector.

In NBC, there is an assumption that attributes are conditionally independent given the class, Therefore, we have the naive Bayes classifier:
$P(C|\mathbf{X}) \propto P(\mathbf{X}|C)P(C) \propto \prod_{i=1}^{m} P(X_i|C)P(C)$, where m is the number of attributes and X_i is the i-th attribute random variable.

Because we don't know the distribution of $P(X_i|C)$ and $P(C)$, Therefore, we need likelihood function to determine unknown parameters based on known outcomes. Assume the data D are independently sampled from the same distribution. Let $D = [\mathbf{x}_1, ..., \mathbf{x}_n]$, where n is the number of samples:

$$L(\theta|D) = \prod_{i=1}^{n} P(\mathbf{x}_i, c_i|\theta) \quad \text{(general likelihood)} \tag{1}$$

$$\propto \prod_{i=1}^{n} P(\mathbf{x}_i|c_i, \theta)P(c_i|\theta) \quad \text{(Bayes rule)} \tag{2}$$

$$\propto \prod_{i=1}^{n}\prod_{j=1}^{m} P(x_{ij}|c_i, \theta)P(c_i|\theta) \quad \text{(Naive assumption)} \tag{3}$$

We apply Maximum Likelihood estimation to learn the best parameters by finding the value θ that maximizes likelihood:

$$\hat{\theta}_{MLE} = arg\ \max_{\theta} L(\theta) \tag{4}$$

For Multinomials, Let $A \in \{1, ..., k\}$ be a discrete random variable with k values, where $P(A = j) = \theta_j$. Then P(A) is a multinomial distribution:

$$P(A|\theta) = \prod_{j=1}^{k} \theta_j^{I(A=j)} \text{ , where } I(A = j) \text{ is an indicator function.} \tag{5}$$

The likelihood for a data set D is:

$$P(D|\theta) = \prod_{i=1}^{n}\prod_{j=1}^{k} \theta_j^{I(A=j)} = \prod_{j} \theta_i^{n_j} \tag{6}$$

Therefore, by using Lagrange multipliers, the maximum likelihood estimates for each parameter are:

$$\hat{\theta_j} = \frac{n_j}{n} \tag{7}$$

which means that in multinomial case, MLE can be determined analytically by counting.

For continuous inputs X_i, the common way to represent the distributions $P(X_i|Y)$ to assume that

Problem 1 continued on next page...

for each possible discrete value y_k of Y, the distribution of each continuous X_i is Gaussian, and is defined by a mean and standard deviation specific to X_i and y_k.

$$\mu_{ik} = E[X_i|Y = y_k] \tag{8}$$
$$\sigma_{ik}^2 = E[(X_i - \mu_{ik})^2|Y = y_k] \tag{9}$$

Again, by MLE, we get:

$$\hat{\mu}_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j X_i^j \delta(Y^j = y_k) \tag{10}$$
$$\hat{\sigma}_{ik}^2 = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j (X_i^j - \hat{\mu_{ik}})^2 \delta(Y^j = y_k) \tag{11}$$

Then we can estimate continuous attributes using Gaussian distribution with $\hat{\mu}_{ik}$ and $\hat{\sigma}_{ik}^2$.

In this question, we are given 11 attributes.

1. Record Number
2. Amount Requested
3. Interest Rate Percentage
4. Loan Length in Months
5. Loan Title
6. Loan Purpose
7. Monthly Payment
8. Total Amount Funded
9. Debt-To-Income Ratio Percentage
10. FICO Range
11. Status

The Record Number is used as id and will not be considered as an attribute and Status is the classification goal that we are interested in and used as the class random variable. Therefore, the potential attributes are from the 2 to 10 entry, which forms our attribute random vector.

In the step of classifying out-of-sample items, we will use the above shown formula to calculate the P(C|X) and compare $P(C = c_1|X)$ with $P(C = c_2|X)$ to see whether the out-of-sample with its attributes given in X should belong to c_1 or c_2 class.

ii. From the MLE, we have the formula

$$\hat{\theta_j} = \frac{n_j}{n} \tag{12}$$

The prior is estimated from the dataset by counting the number of each class among the entire dataset. However, if the real value prior is far from the estimated one, it will have significant impacts on the correctness of the prediction. For example, if we have a dataset with half of people with cancer and other half are healthy while in really life the probability that a person has a cancer is nearly 0.01%, then in this situation the prior will be estimated wrong (50%), which should be 0.01% for cancer class and 99.99% for healthy class, and cause large false positive in this prediction. Therefore, we can see that the wrong prior in NBC will cause either false positive or false negative to increase depending on the difference between real prior and the estimated one. That's the reason why prior in NBC is important.

Problem 1 continued on next page...

b. Logistic Regression:

i. In logistic regression, we make the assumption that

$$log\frac{P(\mathbf{x}, y=1)}{P(\mathbf{x}, y=0)} = \mathbf{w}^T\mathbf{x} + w_0 \tag{13}$$

which is equivalent to

$$P(y=1|\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}+w_0)}} \tag{14}$$

$$P(y=0|\mathbf{x}) = \frac{e^{-(\mathbf{w}^T\mathbf{x}+w_0)}}{1+e^{-(\mathbf{w}^T\mathbf{x}+w_0)}} \tag{15}$$

Using the canonical representation of the data (adding a dummy feature of value 1 to each input vector), we have

$$P(y=1|\mathbf{x}) = g(\mathbf{x}, \mathbf{w}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}} \tag{16}$$

$$P(y=0|\mathbf{x}) = 1 - g(\mathbf{x}, \mathbf{w}) = \frac{e^{-\mathbf{w}^T\mathbf{x}}}{1+e^{-\mathbf{w}^T\mathbf{x}}} \tag{17}$$

These equations mean that given a training data set $\{(\mathbf{x}_i, y_i) : i = 1, ..., N\}$, and $\mathbf{x}_i \in R^{d+1}$, where N is the total number of training examples and d is the original feature dimension, the learning goal is to find the optimal weight vector $\mathbf{w}$.
The next step is to learn the parameters by using MLE. The log likelihood function is as follows:

$$L(\mathbf{w}) = \sum_{i=1}^{N} log\ p(y_i|\mathbf{x}_i) = \sum_{i=1}^{N} log\ g(\mathbf{x}_i, \mathbf{w})^{y_i}(1 - g(\mathbf{x}_i, \mathbf{w}))^{1-y_i} \tag{18}$$

Taking gradient of L with respect to $\mathbf{w}$, we have

$$\sum_{i=1}^{N}(y_i - g(\mathbf{x}_i, \mathbf{w}))\mathbf{x}_i = \mathbf{\Phi}^T(\mathbf{y} - g(\mathbf{x}, \mathbf{w})) \tag{19}$$

Now we use Newton-Raphson update for gradient descent

$$\mathbf{H} = \sum_{i=1}^{N} g(\mathbf{x}_i, \mathbf{w})(1 - g(\mathbf{x}_i, \mathbf{w}))\mathbf{x}_i\mathbf{x}_i^T \tag{20}$$

we donote it as:

$$\mathbf{H} = \sum_{i=1}^{N} g(\mathbf{x}_i, \mathbf{w})(1 - g(\mathbf{x}_i, \mathbf{w}))\mathbf{x}_i\mathbf{x}_i^T = \mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi} \tag{21}$$

where $R_{nn} = g(\mathbf{x}_i, \mathbf{w})(1 - g(\mathbf{x}_i, \mathbf{w})$
Then the iterative parameter update is

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} + (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T(\mathbf{y} - g(\mathbf{x}, \mathbf{w})) \tag{22}$$

$$= (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{R}\mathbf{z} \tag{23}$$

where $\mathbf{z}$ is an N-dimensional vector with elements

$$\mathbf{z} = \mathbf{\Phi}\mathbf{w}^{(\text{old})} - \mathbf{R}^{-1}(g(\mathbf{x}, \mathbf{w}) - \mathbf{y}) \tag{24}$$

In this question, we are given 11 attributes.

1. Record Number
2. Amount Requested
3. Interest Rate Percentage
4. Loan Length in Months
5. Loan Title
6. Loan Purpose
7. Monthly Payment
8. Total Amount Funded
9. Debt-To-Income Ratio Percentage
10. FICO Range
11. Status

The Record Number is used as id and will not be considered as an attribute and Status is the classification goal that we are interested in and used as the class random variable. Therefore, the potential attributes are from the 2 to 10 entry, which forms our attribute random vector. We also feed weight vector $\mathbf{w}$ to the logistic regression to train our model.

Once the model is trained with the $\mathbf{w}$, in the step of classifying out-of-sample items, we will use the above shown formula to calculate the P($\mathbf{y}|\mathbf{x}$) and compare $P(y = 0|\mathbf{x})$ with $P(y = 1|\mathbf{x})$ to see whether the out-of-sample with its attributes given in X should belong to $y = 0$ or $y = 1$ class.

ii. We know that the singular matrix doesn't have inverse matrix, if the $\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi}$ is singular, we cannot update the $\mathbf{w}$ for the updata formula is:

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} + (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T(\mathbf{y} - g(\mathbf{x}, \mathbf{w})) \tag{25}$$

$$= (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{R}\mathbf{z} \tag{26}$$

The reason why $\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi}$ is singular is that one or more features are a linear combination of other features.

There are two possible solutions:

The first one is to remove the highly correlated features from the dataset to prevent singular matrix happens or we can handle the feature in different way to prevent such this occur.

The second one is to modify the matrix by adding small value in the diagonal of the matrix to solve the singular matrix situation. However, this way will slightly modify the result of our prediction so the there will have slight difference between the original feature and modified one.

Problem 1 continued on next page...

c. i. To find support point for SVM (assuming linearly separable data), Back to our linear model with non-linear features ϕ.

$$y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b \tag{27}$$

For two classes, if class $t_n \in \{-1, 1\}$ of item $\mathbf{x}_n$ Then $t_n y(\mathbf{x}_n) > 0$ means correctly classified.
Also, the distance to the hyperplane is:

$$\frac{y(\mathbf{x})}{||\mathbf{w}||} \tag{28}$$

Thus, distance of $\mathbf{x}_n$ from decision hyperplane is the maximum minimum distance.

$$\underset{\mathbf{w},b}{arg\ max}\left\{\frac{1}{||\mathbf{w}||}\underset{n}{min}[t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b)]\right\} \tag{29}$$

However, because the problem is too complicated to compute, so we recast problem into another optimization problem. Then the original problem becomes (as $arg\ max||\mathbf{x}||^{-1} = arg\ min||\mathbf{w}||^2$) a quadratic programming problem.

$$\underset{\mathbf{w},b}{arg\ min}\frac{1}{2}||\mathbf{w}|| \tag{30}$$

s.t.

$$t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) \geqslant 1, \quad n = 1, ..., N \tag{31}$$

We can solve constrained optimization problem via Lagrange multipliers $a_n \geqslant 0$

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{n=1}^{N} a_n\left\{t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) - 1\right\} \tag{32}$$

Setting derivative of $L(\mathbf{w}, b, \mathbf{a})$ w,r,t $\mathbf{w}$ and b to zero, we get:

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n) \tag{33}$$

$$0 = \sum_{n=1}^{N} a_n t_n \tag{34}$$

Eliminating $\mathbf{w}$ and b from previous equation using these conditions:

$$\widetilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{n=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \tag{35}$$

s.t.

$$a_n \geqslant 0, \quad n = 1, ..., N \tag{36}$$

$$\sum_{n=1}^{N} a_n t_n = 0 \tag{37}$$

where $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}')$, and k is the kernel.
For Linear kernels:

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}' \tag{38}$$

Problem 1 continued on next page...

For Gaussian kernels:

$$k(\mathbf{x}, \mathbf{x}') = exp(\frac{1}{2}||\mathbf{x} - \mathbf{x}'||^2) \tag{39}$$

For non-linearly separable data, the

$$a_n \geqslant 0, \quad n = 1, ..., N \tag{40}$$

becomes

$$0 \leqslant a_n \leqslant \mathbf{C}, \quad n = 1, ..., N \tag{41}$$

where $\mathbf{C}$ can be seen as a penalty for misclassification.

In this question, we are given 11 attributes.

1. Record Number
2. Amount Requested
3. Interest Rate Percentage
4. Loan Length in Months
5. Loan Title
6. Loan Purpose
7. Monthly Payment
8. Total Amount Funded
9. Debt-To-Income Ratio Percentage
10. FICO Range
11. Status

The Record Number is used as id and will not be considered as an attribute and Status is the classification goal that we are interested in and used as the class random variable (t_n in the previous formula). Therefore, the potential attributes are from the 2 to 10 entry, which forms our attribute random vector. We also feed weight vector $\mathbf{w}$, kernel type $k(\mathbf{x}, \mathbf{x}')$ and $\mathbf{C}$ to the train our SVM model.

Once the model is trained, in the step of classifying out-of-sample items, we will use the above shown formula($\mathbf{w}^T\phi(\mathbf{x}) + b$) to see its sign to determine which class the out-of-sample with its attributes given in X should belong to $\{-1, 1\}$.

ii. The Gaussian kernel work best with the data. The reason is that there are just a few features with approximate 3500 training data. If we use linear kernel, the model is too simple so that the bias of the model is high and cause the higher error than Gaussian kernel. If we have lots of feature that may cause Gaussian kernel become too complicated, than at that case the Gaussian kernel will have high variance that it didn't perform well on testcases. Then at that case, we should choose linear kernel to reduce the complexity of the model.

d.

1) how to do k-fold validation.
The k-fold validation is done by randomly splitting the training dataset into k pieces, and each time we use 1 piece as testing dataset and others as training dataset and repeat this procedure for k time. Then we can avoid the situation that the model is only doing well in special cases.

2) give the average F1 score (possibly also the variance if you want)

The F1 score formula:

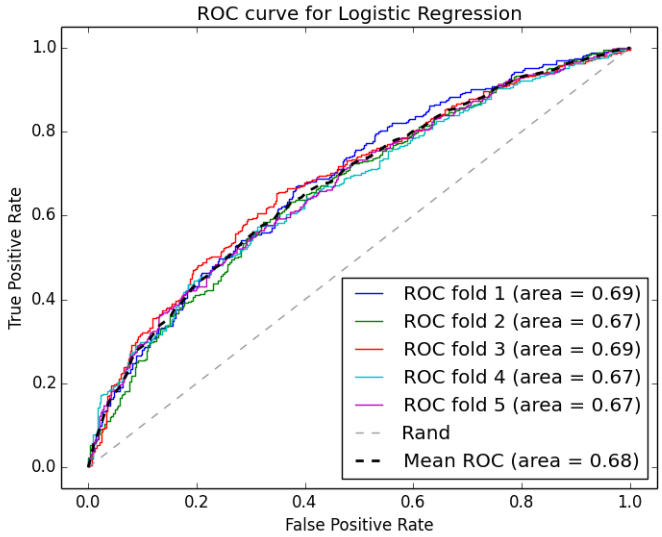
$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{42}$$

5-fold:
NBC: 0.606583184549
Logistic Regression: 0.618686739271
Linear SVM: 0.477387708457
Gaussian SVM: 0.540240372093
10-fold:
NBC: 0.605173788179
Logistic Regression: 0.610975291637
Linear SVM: 0.452656020523
Gaussian SVM: 0.536445423083

3) comment the results you obtained [which classifier seems to do best]
The Logistic Regression seems to do better bases on my result. We will use paired t-test to test the significance in the next question.

4) explain how to get the ROC curve from the logistic regression output
In logistic regression, we compute the probabilities to decide which class the node should belong to. If we adjust the probability threshold and re-classify all individuals using new threshold, then we get new true positive rate and false positive rate. If we calculate lots of these threshold, we can then draw the ROC curve with true positive rate as y axis and false negative rate as x axis.

5) Plot the ROC for logistic



6) Give the AUC score (say AUC is the area under the ROC curve)
0.68 as shown in the figure.

e.
The null hypothesis is that logistic regression has the same F1 score with NBC , and the p-value of

F1-score between logistic regression and NBC is 0.19062380751022309 $> \alpha = 0.05$. Therefore, we cannot reject the null hypothesis.
The null hypothesis is that logistic regression has the same F1 score with linear SVM and the p-value of F1-score between logistic regression and linear SVM is 1.9670734624844568e-12 $< \alpha = 0.05$. Therefore, we can say that there is significant difference that logistic regression is better than linear SVM.
The null hypothesis is that logistic regression has the same F1 score with Gaussian SVM and the p-value of F1-score between logistic regression and Gaussian SVM is 4.3700619261450163e-06 $< \alpha = 0.05$. Therefore, we can say that there is significant difference that logistic regression is better than Gaussian SVM.

f.

I select the logistic regression as my algorithm and choose 'Interest Rate Percentage', 'Loan Purpose', 'Monthly PAYMENT', 'Debt-To-Income Ratio Percentage', 'FICO Range' as my features.

Problem 2

a.

1. Shortest Path
   Since we are given the undirected graph G_0, for each node with more than 10 neighbors, we apply shortest path algorithm to find the nearest node that is not its neighbor and predict the edge between them as the missing edge.
   Advantage:
   Based on existing algorithm and it is intuitive.

   Disadvantage:
   Network diameter often very small and distribution very concentrated.

2. Common Neighbors
   Common neighbors usess as score the number of common neighbors between vertices u and v. For each node with more than 10 neighbors, we calculate the score between the node and others that is not its neighbor to predict the one with highest score as missing edge.

$$score(u, v) = |N(u) \cap N(v)| \tag{43}$$

   Advantage:
   The algorithm is efficient and works well on small society community.

   Disadvantage:
   Large scores for vertices with too many neighbors

3. Jaccard Similarity
   The Jaccard Similarity is aimed to compensate the problem in Common Neighbors. It has same idea with Common Neighbors but with modified formula:

$$score(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \tag{44}$$

   Advantage:
   Fixes the issue of Common Neighbors, "u or v have too many neighbors", by dividing the intersection by the union.

   Disadvantage:
   Although u, v are not problem, but the method fails to measure the quality of their neighbors.

4. Adamic / Adar
   It is improved version of Jaccard Similarity and can be applied similarly as previous two methods, the formula it use is:

$$score(u, v) = \sum_{z \epsilon N(u) \cap N(v)} \frac{1}{log|N(z)|} \tag{45}$$

   where N(z) is the neighbors of the common neighbors of u and v.
   Advantage:
   This score gives more weight to neighbors that are not shared with many others.

Disadvantage:

It is based on the hypothesis that the neighbor of u and v with less neighbors imply that u and v should know each other, which may not be true is some cases.

5. Katz score

In Katz score, we calculate the score to find out the missing edges. The Katz score uses adjacency matrix A in its formula:

$$score(u,v) = \sum_{l=1}^{\infty} \alpha^l (A^l)_{u,v} \tag{46}$$

The α is to ensure that the sum isn't divergent.

Advantage:

The adjacency matrix A is easy to express the relationship between nodes. A itself expresses the nodes with path of length 1 and AA can express the number of common neighbors between $node_i$ and $node_j$.

Disadvantage:

For $\alpha << 1$, predictions will be approximated to common neighbors, which share the same defect.

6. Page Rank

Each edge's vote is proportional to the importance of its source node. Therefore, if node j with importance r_j has n out-edges, each link gets $\frac{r_j}{n}$ votes. Node j's own importance is the sum of the votes on its in-links.

For the stochastic adjacency matrix M, let node i has d_i out-links. If $i \to j$, then $M_{ji} = \frac{1}{d_i}$ else $M_{ji} = 0$, where M is a colum stochastic matrix and the columns sum to 1.

Rank vector r: vector with an entry per page

r_i is the importance score of node i and $\sum_i r_i = 1$ Then the flow equations can be written as:

$$r = M \cdot r \tag{47}$$

$$(r_j = \sum_{i \to j} \frac{r_i}{d_i}) \tag{48}$$

The pagerank also has random teleports to solve dead ends and spider trap problem, the modified formula is:

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N} \tag{49}$$

In order to use pagerank to do link prediction, we have to apply rooted pagerank, which is also called personalized pagerank. The idea is based on hitting time . We first choose a rooted node say x, then for each step, there is α probability that goes back to x ('reset'), and there is $1-\alpha$ probability that goes from current node to its random neighbor. So the formula becomes:

$$r_{root} = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta) \tag{50}$$

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} \ , \quad \text{where } j \mathrel{!}= root \tag{51}$$

For each node, we use it as root to calculate its pagerank and retrieve the top 5 score nodes that is original not the neighbor of root as its missing neighbor.

Advantage:

The matrix will always converge, so we don't need α value as in Katz score to prevent divergent.

Disadvantage:

It share the same idea as hitting time and it is a randomized algorithm. Therefore, it relies on the probabilistic and may get bad result in chance.

b.

In this question, I choose rooted pagerank as my best algorithm for it is well developed and efficient. To verify this approach, I randomly remove an edge from nodes with more than 10 edges and put those edges into validation set. By this way, I can test the prediction accuracy in my approach. For each node we report five candidates and if any of the candidate is the node of the removed edge, we view it as a successful hit. After comparing the accuracy of different methods, we select rooted pagerank as the best algorithm.

Then I put those edges back and re-run the algorithm to report the real missing candidate.

The data contains more than ten thousand nodes with approximate four thousand nodes with more than 10 edges. Each one hundred rooted pagerank procedure takes about three minutes. Therefore, to calculate all the four thousand nodes, it takes us about two hours.