CS498 Homework 2

Xinchen Pan, Fangxiaozhi Yu, Jiefei Shi

February 4, 2017

3.6

(a)

Firstly we loaded the data set using `fread` functions setting `na.strings = "?"` since they are the missing values. There are 31420 instances and 5410 attributes in the original data set. we removed the $5410th$ columns as it is all NULL. We did not use `as.numeric` because after reading in the data, all our values are automatically transformed to `num`. The response variable is the 5409th attribute which has label **active** and **inactive**. After removing missing values, we have 31159 instances left.

Then we splitted the data set into training and testing, 90% for training and 10% for testing.

Firstly we created a new variable based on the labels of the response variable. If the label is **active**, we set as 1. If the label is **inactive**, we set as -1. Then we removed the original response variable and used the new variable as the response variable. So we still have 5409 attributes in total.

To reach 50% of the training data, we used 150 seasons and 100 steps. For each step, we selected one point at random unformly and then stepped down using gradient descent. The total instances we touched were 15000. The training data set about 28000 instances, thus we **indeed touched** more 50% of the training data set. For the steplength, we choose $m = 0.01$ and $n = 0.01$. So $\eta^r = \frac{0.1}{0.01+r}$ where r represents r'th season. The λ we chose was determined later in cross validation. We used it to choose the λ which would have the lowest cv error. We initialized a and b randomly from a uniform distribution. Our $\mathbf{a}$ is a 5408×1 matrix and b is a number.

We want to compute

$$\nabla(\max(0, 1 - y_k(\mathbf{a} \cdot \mathbf{x_k} + b)) + \frac{\lambda}{2}\mathbf{a}^T\mathbf{a})$$

We then write the gradient as

$$p_k = \begin{cases} \begin{bmatrix} \lambda\mathbf{a} \\ 0 \end{bmatrix} & \text{if } y_k(\mathbf{a} \cdot \mathbf{x}_k + b) \geq 1 \\ \begin{bmatrix} \lambda\mathbf{a} - y_k\mathbf{x} \\ -y_k \end{bmatrix} & \text{otherwise} \end{cases}$$

We chose a steplength η then keep updating our a and b using

$$\mathbf{a}^{n+1} = \mathbf{a}^{(n)} - \eta \begin{cases} \lambda\mathbf{a} & \text{if } y_k(\mathbf{a} \cdot \mathbf{x}_k + b) \geq 1 \\ \lambda\mathbf{a} - y_k\mathbf{x} & \text{otherwise} \end{cases}$$

and

$$b^{n+1} = b^{(n)} - \eta \begin{cases} 0 & \text{if } y_k(\mathbf{a} \cdot \mathbf{x}_k + b) \geq 1 \\ -y_k & \text{otherwise} \end{cases}$$

After we finished updating $\mathbf{a}$ and b, we used test data set to test. If the sign of $\mathbf{a} \cdot \mathbf{x} + b \geq 1$, we classified it to 1 else we classified to -1. We then compared it to the label of the testing data.

Cross Validation

We tested three regularization constant $\lambda = 0.001, 0.01, 0.1$ using cross validation. We used 10 fold cross validation. The least cv error we got is from $\lambda = 0.01$. The result is below

lambda = `0.001`	lambda = 0.01	lambda = 0.1
0.0084879	0.0078103	0.0090228

We them used $\lambda = 0.01$ as the regularization constant. We trained our model using **all** the training set(90% of original) and then tested using 10% of the data set. The accuracy is 99.22953%

The following is the plot of the accuracy during the 150 season, each season is 100 steps.

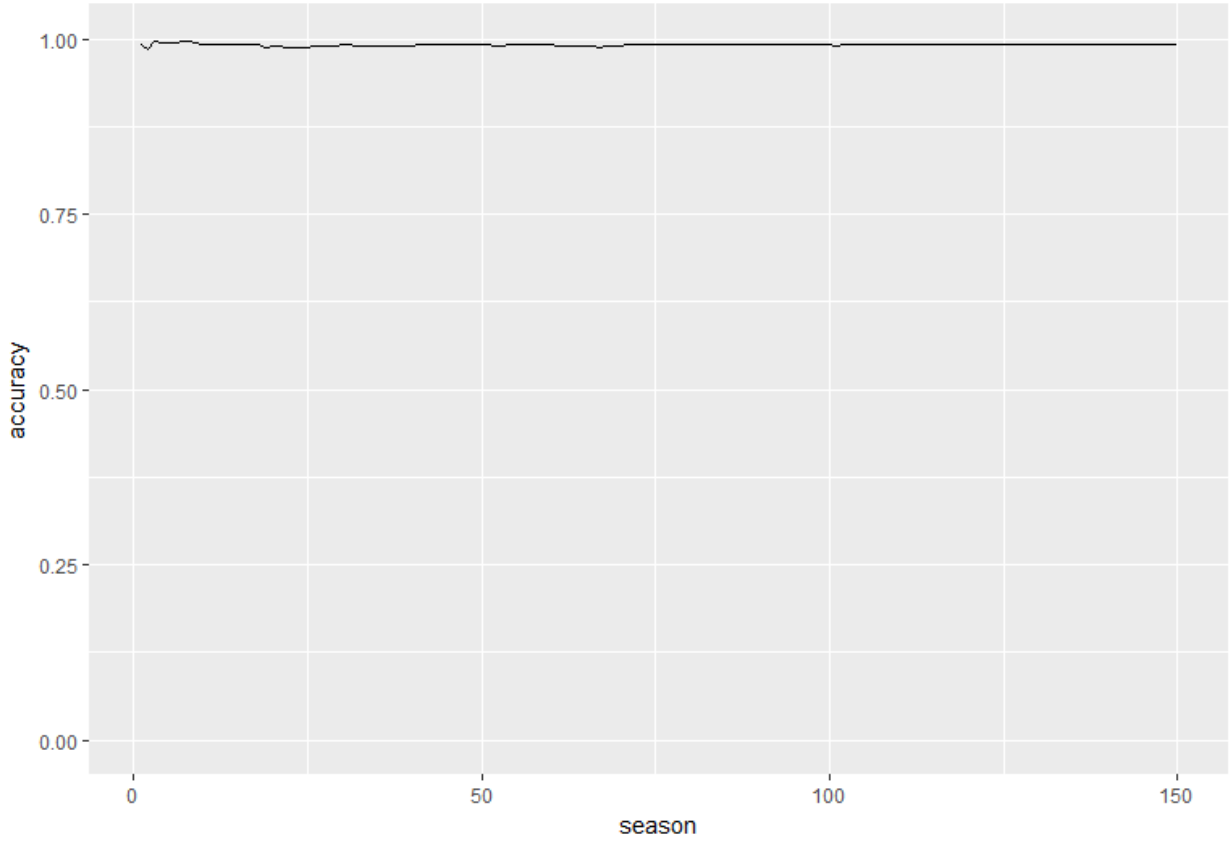

Figure 1: "Season vs Accuaracy"

(b)

For second question, we used `train` function from `caret` package. Because of the computational time, we only used 5000 instances. We got an accuracy about 99.40828%.

(c)

Both SVM and naive bayes classifier work well. Since the response variable is very unbalanced as there are much more **inactive** observations than **active** observations. Thus during training, the classifier turned to classify more to **inactive**. However, naive bayes method is slower than the svm classifier we built. The reason can be we used stochastic gradient descent which only use a portion of the data each time.