Report of Homework 3, STAT 542

Wenke Huang

671105713

Question 1

1.1 Introduction

A retrospective sample of males in a heart-disease high-risk region of the Western Cape, South Africa. There are roughly two controls per case of CHD (Coronary Heart Disease). Many of the CHD positive men have undergone blood pressure reduction treatment and other programs to reduce their risk factors after their CHD event. In some cases the measurements were made after these treatments. These data are taken from a larger dataset, described in Rousseauw et al, 1983, South African Medical Journal. We can have access to these data from the R package `ElemStatLearn`.

Based on the description part of this data set, we can know that the variable `chd` is the response. It is a binary variable which contains only two values `0` and `1`. We can read from the description article that they are all related to whether a particular individual will have coronary heart disease or not. We decide to consider all of them to be the predictors of this case.

The model SVM (Support Vector Machine) is considered here. We will talk about different types of both kernels and their corresponding tuning parameters (such as, cost, degree and sigma) in this case and 5-fold cross-validation will help us to find out the best tuning parametes of different methods we choose and this can finally lead us to the best model.

1.2 Linear SVM

For non-separable Linear SVM, we set the optimization problem to be minimize $\frac{1}{2}||\beta||^2 + C\sum_{i=1}^{n}\xi_i$, subject to $y_i(x^T\beta + \beta_0) \geq (1 - \xi_i)$ and $\xi_i \geq 0$ (i $= 1, 2, ..., n$). Because in almost every case of the real world, zero-error is not always attainable. Here, slack variables ξ_i are created by us that help us account for these errors. C which is positive is a tuning parameter for "cost".

We can easily find that the objective function consists of two parts. For thoses observations which cannot be classified correctly, we set $\xi_i > 1$. So we know $\sum_i \xi_i$ is an upper bound on number of training errors. We also minimize the inverse margin $\frac{1}{2}||\beta||^2$. The tuning parameter C is used to balance the error and margin width. Larger C puts more weight on misclassification rate than margin width while smaller C puts more attention on data further away from the boundary. We can and will choose cross-validation to find our C. And for the seperable cases which are very rare, $C = \infty$. The inequality constraints are used to soft classification to allow errors.

The tuning parameters C we try and their corresponding cross-validation accuracies are shown below. Further analysis and comparation will be discussed in the following part.

Table 1: Tuning Parameters and Accuracies (Linear SVM)

C	Accuracy
0.25	0.7035943
0.50	0.7035640
1.00	0.7031078

C	Accuracy
2.00	0.7021921
4.00	0.7021974

1.3 Non-Linear SVM

In many cases, linear classifier is not flexible enough. We need to pay attention on creating the nonlinear boundaries. The decision function becomes $f(x) = \langle \Phi(x), \beta \rangle$ and we have two ways to figure it out, Kernel trick and Naive approach.

Here, we only talk about Kernel trick. It is because that we could calculate $\Phi(x)$ for all of x_i if we know it and do the further calculation, which is not necessary and a total waste of computation time. For the Kernel trick, the kernel function $K(x, z) = \langle \Phi(x), \Phi(z) \rangle$ is considered and we only pay attention on inner product. Using Kernel trick is also the reason why we went all the way from the primal form to dual form to deal with SVM. Because there is no inner product involved in the primal form.

1.3.1 Radial SVM

Here the Radial basis is chosen. Its corresponding $K(x_1, x_2) = exp(-\frac{||x_1 - x_2||^2}{2\sigma^2})$. In some examples, $1/(2\sigma^2)$ can also be written as γ. Here, we do not do that so that we can separate this one from the previous C which stands for cost.

The tuning parameters C and γ that we try and their corresponding cross-validation accuracies are shown below. Further analysis and comparation will be discussed in the following part.

Table 2: Tuning Parameters and Accuracies (Radial SVM)

	C	sigma	Accuracy
1	0.25	0.125	0.7068236
6	0.50	0.125	0.7113139
11	1.00	0.125	0.7036972
16	2.00	0.125	0.6885008
21	4.00	0.125	0.6723960
2	0.25	0.250	0.6897470
7	0.50	0.250	0.6921028
12	1.00	0.250	0.6800893
17	2.00	0.250	0.6651786
22	4.00	0.250	0.6489705
3	0.25	0.500	0.6587155
8	0.50	0.500	0.6676808
13	1.00	0.500	0.6494598
18	2.00	0.500	0.6424301
23	4.00	0.500	0.6362082
4	0.25	1.000	0.6585332
9	0.50	1.000	0.6500379
14	1.00	1.000	0.6361659
19	2.00	1.000	0.6332433
24	4.00	1.000	0.6339525
5	0.25	2.000	0.6585332
10	0.50	2.000	0.6577800
15	1.00	2.000	0.6545217

	C	sigma	Accuracy
20	2.00	2.000	0.6526086
25	4.00	2.000	0.6521468

1.3.2 Polynomial SVM

In this part, we are about to talk about another way to create nonlinear boundaries. Its Kernal function that we consider is $K(x_1, x_2) = (x_1^T x_2 + 1)^d$. Here, d stands for degrees which is the tuning parameter of this case. However, we also should notice that this function may suffer from numerical instability. When $x_1^T x_2 + 1 < 1$, $K(x_1, x_2) = (x_1^T x_2 + 1)^d$ tends to zero with increasing d. On the other hand, when $x_1^T x_2 + 1 > 1$, $K(x_1, x_2) = (x_1^T x_2 + 1)^d$ tends to infinity with increasing d.

The tuning parameters C and d (which will be shown as *degree* in the output tables) that we try and their corresponding cross-validation accuracies are shown below. Further analysis and comparation will be discussed in the following part.

Table 3: Tuning Parameters and Accuracies (Polynomial SVM)

	C	degree	Accuracy
1	0.25	1	0.7194371
4	0.50	1	0.7207731
7	1.00	1	0.7212534
10	2.00	1	0.7217342
13	4.00	1	0.7214238
2	0.25	2	0.6917705
5	0.50	2	0.6883859
8	1.00	2	0.6853653
11	2.00	2	0.6814675
14	4.00	2	0.6782396
3	0.25	3	0.6170845
6	0.50	3	0.6182932
9	1.00	3	0.6171653
12	2.00	3	0.6173697
15	4.00	3	0.6178494

1.4 Comparation and conclusion

Table 4: Best accuracies of models above

Accuracy.Linear	Accuracy.Radial	Accuracy.Polynomial
0.7035943	0.7113139	0.7217342

We extract the best accuracies of these three methods we have tried in previous parts. The results are shown above. We can find that the cross-validation accuracy of polynomial SVM is the greatest of these three which is 0.7217342, which shows that this model can give us a better prediction than any of the rest two models. Hence, we think that in this case, the SVM model with polynomial kernal is the best model which is between over and under fitting. And the tuning parameters we will choose and the corresponding 5-fold cross-validation accuracy are shown below.

Table 5: Final choose of this model

	degree	C	Accuracy
10	1	2	0.7217342

Question 2

2.1 Introduction

In this case, we would like to construct the natural cubic spline basis from the cubic spline basis. The truncated power series basis representation for cubic splines with K interior knots $\xi_1, \xi_2, ..., \xi_K$ is given by $f(X) = \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3$, where$(.)_+$ denotes the positive part. We also set the following contrain to construct this natural cubic spline: $f(X)$ is linear for both $X \leq \xi_1$ and $X \geq \xi_K$.

Out ultimate goal of this case is to proof that the original model can be also re-writed as $f(X) = \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \alpha_k (d_k(X) - d_{K-1}(X))$, where $d_k(X) = \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k}$ and $\alpha_k = \theta_k(\xi_K - \xi_k)$.

2.2 Proof

2.2.1 Pre-preparation and proof

It is easy to find that when $X \leq \xi_1$, $X < \xi_k$ for all of $k = 2, 3, ..., K$. So we re-write the function and get $f(X) = \sum_{j=0}^{3} \beta_j X^j$ when $X \leq \xi_1$. This model is a linear one if and only if $\beta_2 = \beta_3 = 0$. Here we get that both β_2 and β_3 are 0 here.

When $X \geq \xi_K$, $X > \xi_k$ for all of $k = 1, 2, ..., K - 1$. Hence, when $X \geq \xi_K$, we can get the function $f(X) = \beta_0 + \beta_1 X + \sum_{k=1}^{K} \theta_k (X - \xi_i)^3$. Based on the constrain, we can know that it is also a linear model. So we know that its corresponding first derivative is a constant (which may and may not be 0) and second derivative is the constant 0 which is shown below.

$f''(X) = 6 \sum_{k=1}^{K} \theta_k (X - \xi_k) = 0 \Rightarrow X \sum_{k=1}^{K} \theta_k - \sum_{k=1}^{K} \theta_k \xi_k = 0$, which is true if and only if $\sum_{k=1}^{K} \theta_k = 0$ and $\sum_{k=1}^{K} \theta_k \xi_k = 0$ for all of the $X \geq \xi_K$.

Here, we summary all of these following results that we have proved so far:

(1) β_2 and β_3 are both 0, (2) $\sum_{k=1}^{K} \theta_k = 0$, and (3) $\sum_{k=1}^{K} \theta_k \xi_k = 0$.

And we can also re-write the original function to $f(X) = \beta_0 + \beta_1 X + \sum_{k=1}^{K} \theta_k (X - \xi_i)_+^3$.

2.2.2 Rigorous calculation

Here, we only focus on the third part of this function, because the first two parts will not be changed anyway. Apart from those results which we have shown in the previous part, we also define two new functions to help us finish this mission, which are $d_k(X) = \dfrac{(X-\xi_k)^3_+ - (X-\xi_K)^3_+}{\xi_K - \xi_k}$ and $\alpha_k = \theta_k(\xi_K - \xi_k)$. The proof will be shown below, which contains all of the possible transformations we have discussed so far.

$f(X) - \beta_0 - \beta_1 X$

$= \sum_{k=1}^{K} \theta_k (X-\xi_k)^3_+.$

$= \sum_{k=1}^{K-1} \theta_k (X-\xi_k)^3_+ + \theta_K (X-\xi_K)^3_+$

$= \sum_{k=1}^{K-1} \theta_k (X-\xi_k)^3_+ + 0 - \sum_{k=1}^{K-1} \theta_k (X-\xi_K)^3_+$

$= \sum_{k=1}^{K-1} \theta_k [(X-\xi_k)^3_+ - (X-\xi_K)^3_+]$

$= \sum_{k=1}^{K-2} \theta_k [(X-\xi_k)^3_+ - (X-\xi_K)^3_+] + \theta_{K-1}[(X-\xi_{K-1})^3_+ - (X-\xi_K)^3_+]$

$= \sum_{k=1}^{K-2} \theta_k [(X-\xi_k)^3_+ - (X-\xi_K)^3_+] - (\theta_{K-1}\xi_{K-1} - \theta_{K-1}\xi_K)\dfrac{(X-\xi_{K-1})^3_+ - (X-\xi_K)^3_+}{\xi_K - \xi_{K-1}}$

$= \sum_{k=1}^{K-2} \theta_k [(X-\xi_k)^3_+ - (X-\xi_K)^3_+] - [(0 - \theta_{K-1}\xi_K - \theta_K\xi_K) - (0 - \theta_{K-1}\xi_{K-1} - \theta_K\xi_K)]d_{K-1}(X)$

Above we do the transformation: $d_k(X) = \dfrac{(X-\xi_k)^3_+ - (X-\xi_K)^3_+}{\xi_K - \xi_k}$

$= \sum_{k=1}^{K-2} \theta_k(\xi_K - \xi_k)\dfrac{(X-\xi_k)^3_+ - (X-\xi_K)^3_+}{\xi_K - \xi_k} - (\sum_{k=1}^{K-2}\theta_k\xi_K - \sum_{k=1}^{K-2}\theta_k\xi_k)d_{K-1}(X)$

$= \sum_{k=1}^{K-2} \alpha_k d_k(X) - \sum_{k=1}^{K-2} \theta_k(\xi_K - \xi_k)d_{K-1}(X)$

Above we do the transformation: $d_k(X) = \dfrac{(X-\xi_k)^3_+ - (X-\xi_K)^3_+}{\xi_K - \xi_k}$

$= \sum_{k=1}^{K-2} \alpha_k d_k(X) - \sum_{k=1}^{K-2} \alpha_k d_{K-1}(X)$

Above and below we do the transformation: $\alpha_k = \theta_k(\xi_K - \xi_k)$

$= \sum_{k=1}^{K-2} \alpha_k (d_k(X) - d_{K-1}(X))$

Then we re-write the function above. We add $\beta_0 + \beta_1 X$ to both sides of the function. Hence we can get the equation which is

$$f(X) = \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \alpha_k (d_k(X) - d_{K-1}(X))$$

where $d_k(X) = \dfrac{(X-\xi_k)^3_+ - (X-\xi_K)^3_+}{\xi_K - \xi_k}$ and $\alpha_k = \theta_k(\xi_K - \xi_k)$.

2.3 Conclusion

According to the rigorous mathematical proof with clear logic, we can draw our final conclusion. Suppose we construct a natural cubic spline: $f(X) = \sum_{j=0}^{3} \beta_j X^j$ when $X \leq \xi_1$ with the constrain which is that $f(X)$ is linear for both $X \leq \xi_1$ and $X \geq \xi_K$. We get to know that the power series representation can also re-wrote as $f(X) = \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \alpha_k (d_k(X) - d_{K-1}(X))$, where $d_k(X) = \dfrac{(X-\xi_k)_+^3 - (X-\xi_K)_+^3}{\xi_K - \xi_k}$ and $\alpha_k = \theta_k(\xi_K - \xi_k)$.

Question 3

3.1 Introduction

In this case, we would like to study the degrees of freedom for random forest. We have tried to studied the degees of freedom for k-nearest-neighbor, and this time we will use the same method. The degrees of freedom can be calculated by using the formular $\sum_i Cov(\hat{y_i}, y_i)/\sigma^2$. The differences between these two is that this time there will not be theoretical degrees of freedom any more and we only focus on our simulations.

The predictor X is a 200×4 matrix while the respotnse Y is a 200×1 vector. All of the elements of matirx X follow independent standard normal distribution. The relationship between predictor X and response Y can be written like

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon$$

where β is $(1, 1, 1, 2)^T$ and noise ε follows independent standard normal distribution. Easily we can know $E(\mathbf{Y}) = \mathbf{X}\beta$ and $Var(\mathbf{Y}) = \mathbf{I}_{200}$, where $\mathbf{I}_{200}$ is a 200×200 identity matrix.

First we generate matrix $\mathbf{X}$. All of its elements follow the standard normal distribution, which we have discussed in the previous paragraph. Second, we use the function $E(\mathbf{Y}) = \mathbf{X}\beta$ to calculate the mean value of response vector $\mathbf{Y}$. Both matrix $\mathbf{X}$ and vector $E(\mathbf{Y})$ will be fixed for the rest of this question. Then, we add an independent standard normal noise ε to each observation of the mean response vector $E(\mathbf{Y})$ to get the response vector $\mathbf{Y}$ and use it and matrix $\mathbf{X}$ to fit the random forest model to predict the $\hat{\mathbf{Y}}$. We will repeat this procedure for many times. Finally, we calculate the covariance between each observation of $\mathbf{Y}$ and its corresponding predicted value, which leads us to the degrees of freedom.

3.2 Attempt with fixed tuning parameters

Random forests are an ensemble learning method for classification, regression and other tasks, which operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (when we fit classification model) or mean prediction (when we fit regression model) of the individual trees. Random forests correct for decision trees' habit of overfitting to their training data set.

Three tuning parameters are considered in this model, `ntree`, `mtry` and `nodesize`. `ntree` means the number of trees. We usually select a large number. `mtry` stands for the number of variables which is considered at each split. At each split, randomly select `mtry` variables from the entire set of feature. Search the best variable and splitting point out of these mtry variables. Split and proceed to child nodes. We need to notice that this procedure turns out working remarkably well, even in high-dimensional data. Most of the time, we choose $p/3$ for regression and $\sqrt{p}$ for classification. Random forests do not perform pruning anymore. Instead, splitting does not stop until the terminal node size is less or equal to the number which we set at the beginning and is `nodesize` here. The entire tree is used. `nodesize` controls the trade-off between bias and variance in each tree. In the most extreme case, we have `nodesize`=1 which means fit each observation exactly.

Table 6: Fixed Tuning

Parameter	ntree	mtry	nodesize	DoF
Value	500	2	25	53.37679

In this part, the tuning parameters are considered as fixed values: `ntree`=1000, `mtry`=2, and `nodesize`=25. We repeat fitting response $\mathbf{Y}$ and predict $\hat{\mathbf{Y}}$ for 10 times. Neither $\mathbf{X}$ nor $E(\mathbf{Y})$ values are changed. Then, we calculate the covariance value between each observation of $\mathbf{Y}$ and its corresponding predicted value. The degrees of freedom of this case is 53.3767863 which we get from the formula $\sum_i Cov(\hat{y_i}, y_i)/\sigma^2$.

3.3 Attempt with vary tuning parameters

In the previous part, we tried the model with fixed tuning parameters. Here, we consider varying all of them and calculate their corresponding estimated degrees of freesom. The possible values of `ntree` are 500, 1000 and 1500. The possible values of `mtry` are 1, 2 and 3. And the possible values of `nodesize` are 25, 50 and 75. We use the same method as the previous part to calculate degrees of freedom. All of the possible combinations (which is 27 in total) of these three tuning parameters are attempted. Within each fitting and repeating, the tuning parameters are fixed.

Table 7: Vary Tuning(nodesize=25)

	mtry=1	mtry=2	mtry=3
ntree=500	46.71272	54.39218	58.88506
ntree=1000	44.70204	54.83712	56.42052
ntree=1500	45.27241	53.85462	57.66212

Table 8: Vary Tuning(nodesize=50)

	mtry=1	mtry=2	mtry=3
ntree=500	33.10840	42.96217	47.59825
ntree=1000	33.13119	41.66074	46.92940
ntree=1500	33.68870	43.25872	47.85336

Table 9: Vary Tuning(nodesize=75)

	mtry=1	mtry=2	mtry=3
ntree=500	23.27289	32.67966	38.07900
ntree=1000	23.24304	30.79277	36.63859
ntree=1500	23.31866	30.55195	37.39277

3.4 Comparation and conlcusion

The degrees of freedom we got based on the different tuning parameters are shown above. We can choose to fix two tuning parameters at the same time and find out the influence from the rest tuning parameter at one time. When we fix parameters `ntree` and `nodesize`, we find that degrees of freedom tend to go higher with increasing `mtry`. When we fix parameters `ntree` and `mtry`, we find that degrees of freedom tend to go higher with decreasing `nodesize`. When we fix parameter `mtry` and `nodesize`, we find that degrees of freedom seem not to change significantly if we try different `ntree`.

We can conclude that two of these three tuning parameters have influence on the degrees of freedom in this model. `mtry` positively influences with the degrees of freedom while `nodesize` negatively influences with the degrees of freedom. `ntree` does not influence the degrees of freedom in random forest model.