Home Assignment 3

Xijia Liu

Task 1

In this task, we implement perceptron algorithm in R.

Task 1.1: Write your own program to implement the algorithm in R

Task 1.2: Use the R code 'DataGeneration.R' to generate the data. Apply your function of perceptron on the data and visualize the decision boundary in a plot.

Tips: The perceptron model can be represented as $Sign\left(\mathbf{w}^\top \mathbf{x}\right)$, where $\mathbf{x} = (1, x_1, \ldots, x_p)^\top$, $\mathbf{w} = (w_0, w_1, \ldots, w_p)^\top$

Task 2

In this task, we implement kernel PCA in R.

Task 2.1: Derive the following formula of calculating the kth kernel principal components for a new observation.

$$\lambda_k^{-1}\mathbf{u}_k^\top(\mathbf{I}-\mathbf{C})((\kappa(\mathbf{x}_1,\mathbf{x}_{new}),\ldots,\kappa(\mathbf{x}_n,\mathbf{x}_{new}))^\top - n^{-1}\mathbf{K}\mathbf{1}_n)$$

where $\mathbf{K}$ is the Gram matrix of the training set, $(\lambda_k, \mathbf{u}_k)$ is the kth eigen pairs of $\mathbf{K}$, $\mathbf{x}_i$ is the ith observation, $i = 1, \ldots, n$, and $\mathbf{C} = n^{-1}\mathbf{1}_n\mathbf{1}_n^\top$. For details, check my slides.

Task 2.2: Use the R code 'Task2-KPCA' to generate the data. Write a program to implement KPCA with kernel function $(\mathbf{x}_1^\top \mathbf{x}_2)^2$ over training set and visualize the 3rd kernel principal components in a figure. Calculate the 3rd kernel principal components for the testing set and add the two points into the figure.

Tips: To verify the results, you may compare them with the results of PCA with explicit feature mapping $\phi(\mathbf{x}) = (x_1^2, x_1x_2, x_2^2)^\top$

Task 3

In this task, we implement kernel (RBF kernel) ridge regression and train a predictive model on 'Boston data'.

Task 3.1: Implement kernel ridge regression (KRR) in R. Write a R function 'predictKRR'. The inputs should inlcude

- 'trainX': a data matrix which contains feature variables for training a KRR.
- 'trainY': a numeric vector contains the target variable for training a KRR.
- 'X': a data matrix that contains the feature variables for prediction.
- 'lambda': a scalar to specify the shrinkage parameter in KRR.
- 'sigma': a scalar to specify the parameter in RBF kernel function.

The output of function 'predictKRR' should be the predicted value of the target variable given the input features variable 'X'.

Tips: the RBF kernel can either be written by your own code or using the build-in function 'rbfdot' from the 'kernlab' package. The build-in function 'kernelMatrix' in package 'kernlab' can be applied to compute the Gram matrix (kernel matrix).

Task 3.2: Train a regular regression model on the 'Boston data' as the baseline model. The last column, 'medv', is the target variable and the rest are feature variables. You may follow the following procedure:

- Set the random seed as 2020
- Draw a random sample with 400 observations (use build-in function 'sample') from the Boston data as the training data set. Use the rest observation as the testing data.
- Train the regression model using the training data.

- Estimate the performance of your regress model on the testing data. The performance of the model should be quantified by square root mean square errors.

Task 3.3: Train KRR with RBF kernel function on the 'Boston data'. The last column, 'medv', is the target variable. You train the KRR with the Gaussian kernel function to predict 'medv' by using other variables as inputs. The following procedure is suggested:

- Normalize [1] the data by removing the mean and divided by the standard deviation.
- Set the random seed as 2020
- Draw a random sample with 400 observations from the Boston data as the training data set. Use the rest observation as the testing data.
- Propose potential values for hyper-parameters 'lambda' and 'sigma'. You may propose $(0.1, 0.01, 0.001)'$ both for 'lambda' and 'sigma', then use 'expand.grid' to generate all combinations of them.
- Run 10-fold cross-validation on the train data set and select the 'optimal' values for the hyper-parameters. Root mean square error can be used as a metric of performance.
- Estimate the performance of your final model on the testing data and compare it with the results of Task 1.2.

Task 4

'Titanic dataset' is an interesting and classical pedagogical dataset on Kaggle[2]. The main task of this dataset is to predict if the people can survive the disaster given different background variables.
In this task, we review a nice solution by Megan Risdal. By reviewing her solution, we will not only study an example of implementing random forest in a real problem but also learn what is feature engineering and data cleaning. Some advance R programming skills for data processing are also can be

[1] You may skip this step and check the potential issues
[2] Kaggle is an online community of data scientists and machine learners, owned by Google. www.kaggle.com

learned.

Task 4.1: Review this nice solution and summarize the key steps and results in your report. Her solution is written by markdown in R. You can find both the pdf file, markdown source code, and datasets from Cambro. The markdown source code and data files must be put in the same folder such that you can run it smoothly.

Task 4.2: Based on the processed data and extracted features, train your random forest model. Analysis the output of feature importance.

Task 4.2: Based on the optimal random forest model and the training dataset, calculate the average misclassification error rate of individual trees and the misclassification error rate of the random forest model.