SVM and PCA

Yulu Jin, Kaiming Fu

Reason for applying SVM

As we know, linear regression explores the linear relationship between predictors and the corresponding variable(y here). However, for the data collected in the real world, the relationship between predictors and the response is always quite complex and non-linear. The bank data for this project contains 20 variables as predictors and it's trivial that the relationship between those variables and y is non-linear and hard to explore.

Thus, supporting vector machines is proposed to characterize such a complex non-linear relationship. To solve this problem, the input data is mapped into a higher dimensional space. By applying Kernel functions, a nonlinear problem in the lower dimensional space has been transferred into a linear one and thus an optimal separating hyper-plane can be learned. Actually, the hyper-plane is a boundary plane to perform the classification task.

SVM full model

We use all the 20 variables to predict the response y by the SVM algorithm. Since the SVM model can only deal with numerical data, first we convert all characteristic variables into numerical ones. Then we seperate the data into training and testing set and the training set accounts for 70%.

Since there is randomness in the training set generation part and also in the SVM training part, we apply set.seed() to fix the result. After training, the prediction accuracy on the test set is 90.928%.

```
##
## Call:
## svm(formula = y ~ age + job + marital + education + housing + loan +
##     contact + month + day_of_week + duration + campaign + pdays +
##     previous + poutcome + emp.var.rate + cons.price.idx + cons.conf.idx +
##     euribor3m + nr.employed + default, data = training, importance = T,
##     type = "C-classification")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  5987
```

```
##       predict.SVM..test...1.20...type....response..
## test.y     0     1
##      0 10725   254
##      1   867   511
```

```
## [1] 0.9092822
```

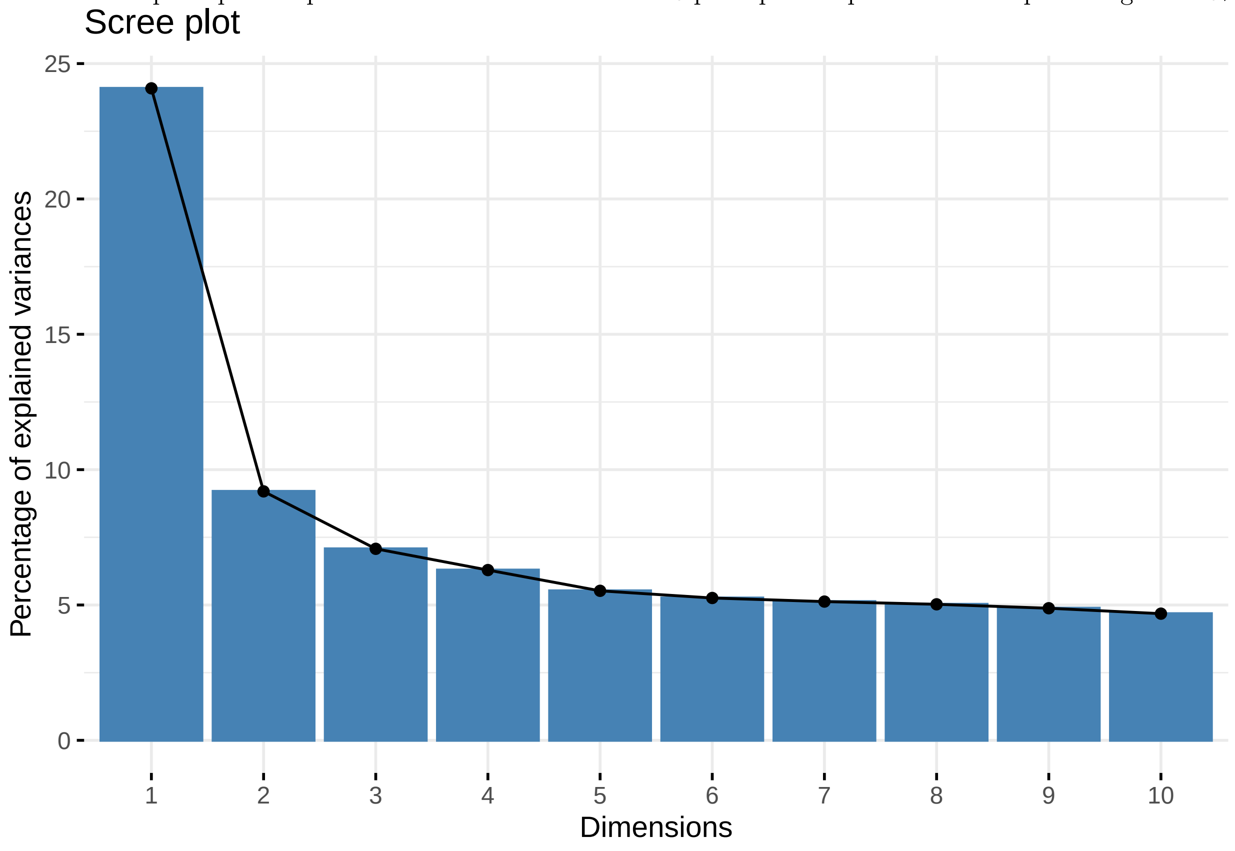
Principal component analysis

Principal component analysis(PCA) is widely used for dimensionality reduction. PCA computes the principal components and then the data points can be projected onto the first few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible. Moreover, it also eexplores the data and tells us the most important variables in the dataset.

We perform PCA on the dataset. By the summary result listed below, we have that the first 13 principal components give a cumulative proportion of 0.89615, which well represent the data.

```
## Importance of components:
##                           PC1     PC2     PC3    PC4     PC5     PC6     PC7
## Standard deviation     2.1948 1.35624 1.18958 1.1216 1.05106 1.02550 1.01252
## Proportion of Variance 0.2409 0.09197 0.07076 0.0629 0.05524 0.05258 0.05126
## Cumulative Proportion  0.2409 0.33282 0.40358 0.4665 0.52172 0.57430 0.62556
##                            PC8    PC9   PC10   PC11    PC12    PC13    PC14
## Standard deviation     1.00255 0.9879 0.9674 0.9497 0.91067 0.87377 0.82992
## Proportion of Variance 0.05026 0.0488 0.0468 0.0451 0.04147 0.03817 0.03444
## Cumulative Proportion  0.67581 0.7246 0.7714 0.8165 0.85797 0.89615 0.93059
##                           PC15    PC16    PC17    PC18    PC19    PC20
## Standard deviation     0.77229 0.63194 0.52046 0.30219 0.14289 0.09942
## Proportion of Variance 0.02982 0.01997 0.01354 0.00457 0.00102 0.00049
## Cumulative Proportion  0.96041 0.98038 0.99392 0.99848 0.99951 1.00000
```

The screen plot of principal components is shown below. The y-axis explains the percentage of explained variance for each principal component and we see that the first 8 principal components have a percentage over 5%.



By calculating the eigenvalue corresponding to each principal component, we have the following result. We would eliminate any variable that didn't have an eigenvalue greater than 1. The idea behind this is that if the eigenvalue is less than 1, then the component accounts for less variance than a single variable contributed.

Thus, we view the first 8 principal components as important ones.

```
##  [1] 4.817087520 1.839383411 1.415111190 1.258025265 1.104729596 1.051643444
##  [7] 1.025195603 1.005114510 0.976005532 0.935940119 0.901938282 0.829320176
## [13] 0.763474234 0.688765080 0.596425595 0.399346889 0.270875507 0.091316810
## [19] 0.020416860 0.009884378
```

By summarizing the first 8 principal components, we see that several variables are less significant than others, such as "contact", "day_of_week", "month", since they contribute little in the first 8 principal components. We then drop those variables and train the SVM model based on the same training and testing set. The test accuracy is then 90.807%. Thus, there is no big difference between this model and the previous one trained by all the variables, which indicates that the dropped variables do not have much effect in the prediction process.

```
##
## Call:
## svm(formula = y ~ age + job + marital + education + housing + loan +
##     duration + campaign + pdays + previous + poutcome + emp.var.rate +
##     cons.price.idx + cons.conf.idx + euribor3m + nr.employed + default,
##     data = training, importance = T, type = "C-classification")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  5951
```

```
##       predict.SVM..test...1.20...type....response..
## test.y     0     1
##      0 10730   249
##      1   887   491
```

```
## [1] 0.9080683
```