STA 208 Mid term

Guicheng Wu

April 30, 2016

1 Description of the data and task

The task is a binary classifier problem. The data has four features for us to predict the response value which is either 1 or 0. Considering the data is 4-dimensional, we cannot efficiently visualize the data. However, we can use the Principal component analysis(PCA) method to get the 2 or 3 principal components, then these principal components can represent the data. With the principal components of data, we can draw 2-dimension or 3-dimension graphs. As I draw the graphs, I discover that the two classes (1 and 0) of data are not effectively divided into two classes. Therefore, I decide to first use kernel methods to get high dimensional data, then reduce to 2 or 3 dimension by PCA. I try four different kernels which are polynomial kernel, RBF(gaussian) kernel, laplacian kernel and sigmoid kernels. The PCA graphs are shown in Figure 1.

The polynomial kernel is as follows.

$$K(x_i, x_j) = (x_i^T x_j + c)^d \tag{1}$$

The RBF kernel is as follows.

$$K(x_i, x_j) = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}} \tag{2}$$

The laplacian kernel is as follows.

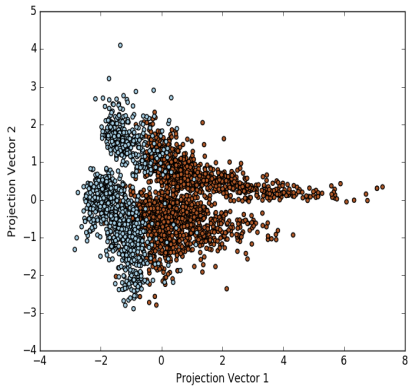
$$K(x_i, x_j) = e^{-gamma||x_i - x_j||} \tag{3}$$

The sigmoid kernel is as follows.

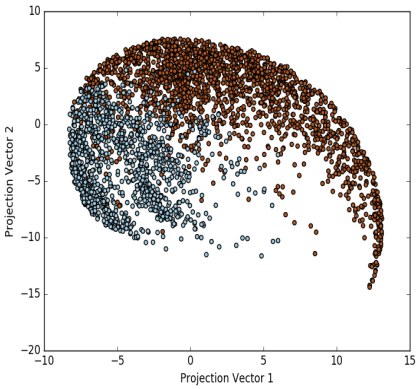
$$K(x_i, x_j) = tanh(\gamma x_i^T x_j + c_0) \tag{4}$$

From the figures, we can see that it's hard to tell which kernel methods work best, but we obviously the PCA graphs using kernels work far better than the one who doesn't use the kernel. Relatively, the Laplacian kernel and RBF kernel have the best performance.

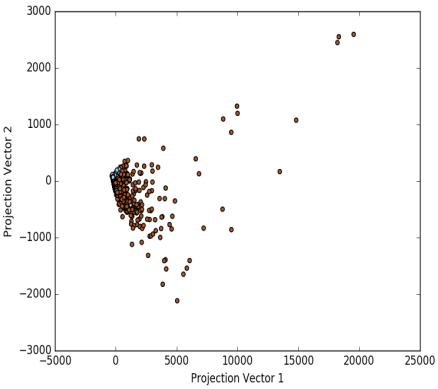
In summary, I preprocess the data as follows. First, I standardize the data by calling the preprocessing.scale() method. Second, considering the laplacian kernel has the best performance, I apply this kernel to all the following classification methods except SVM. For SVM methods, I try to apply different kernels and tune the parameters to show the difference of performances. For the K nearest neighbor method, I tune the parameter K to find the best performance.



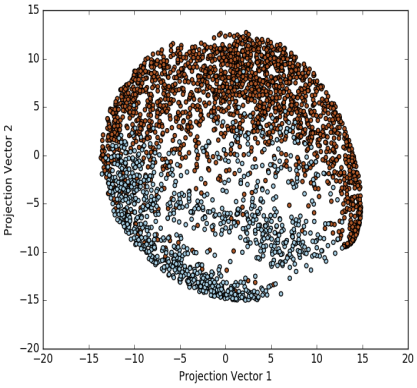
(a) Original Data 2D PCA



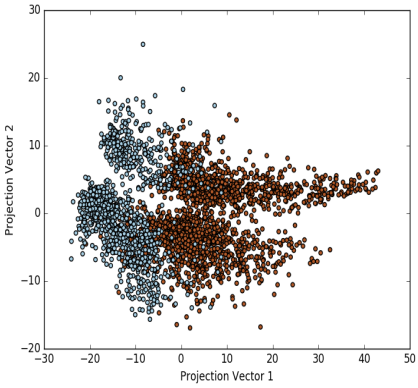
(b) Laplacian kernel 2D PCA



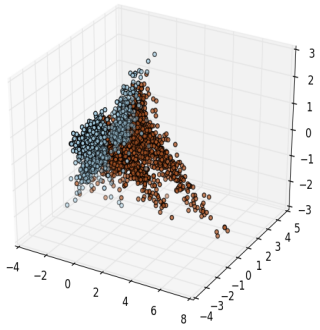
(c) Polynomial Kernel 2D PCA



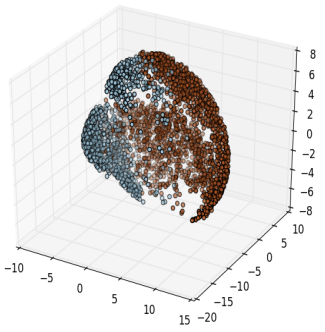
(d) rbf kernel 2D PCA



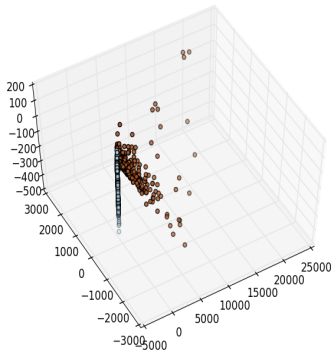
(e) Sigmoid kernel 2D PCA



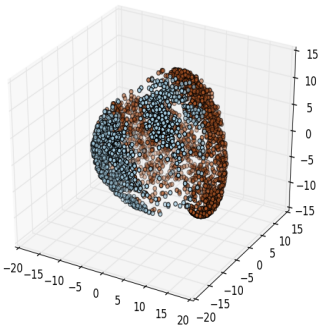
(f) Original Data 3D PCA



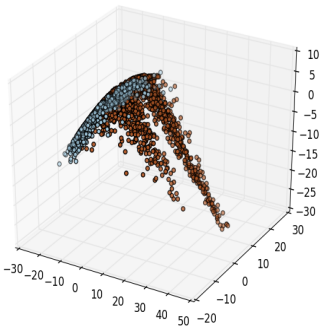
(g) Laplacian kernel 3D PCA



(h) Polynomial Kernel 3D PCA



(i) rbf kernel 3D PCA



(j) Sigmoid kernel 3D PCA

Figure 1: Data process comparison graphs

2 Description of the methods and quality metrics

For the binary classifier problem, there are many methods there. For example, linear regression, ridge regression, K nearest neighbors, naive bayes, logistic regression, linear discriminant analysis, lasso regression, support vector machine(SVM), decision tree, random forest, Adaboost and quadratic discriminant analysis. Different methods have different performances in a certain scenario. Since the sklearn python package has already implemented all these methods and they are easy to call, I decide to try out all these methods and find the most significant one.

For different methods, I can apply different kernels and try different parameters, However, I decide to only tune the parameter of K nearest neighbors, polynomial-kernel SVM, sigmoid-kernel SVM and RBF-kernel SVM because of the time limit. For the K nearest neighbors methods, I tune the parameters K to find the relationship of neighbors K and prediction correctness. For support vector machine(SVM), I apply different kernels and tune related parameter to find out the best performance of each kernel svm. For all other methods, I apply Laplacian kernel on the data, use the default the parameter to train the classifier.

To quantifying the quality of predictions, I use two simple but efficient metrics which are correct rate and test error. The correct rate is got by dividing the correct prediction number by the total test size. The correct rate is defined as follows.

$$\text{correct rate} = \frac{c}{n} \tag{5}$$

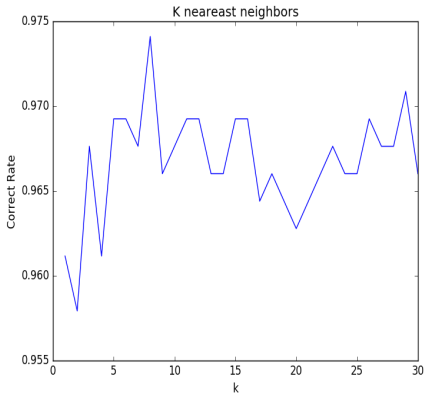
where c stands for correct prediction numbers and n stands for the size of test sample.

The test error is defined as

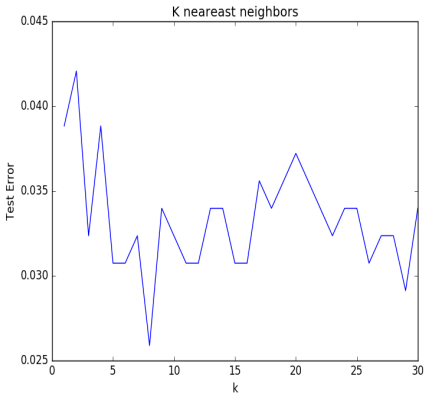
$$\text{test error} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y} - y)^2}{n}} \tag{6}$$

Where $\hat{y}$ is the prediction value, y is the true value and n is the size of the test sample.

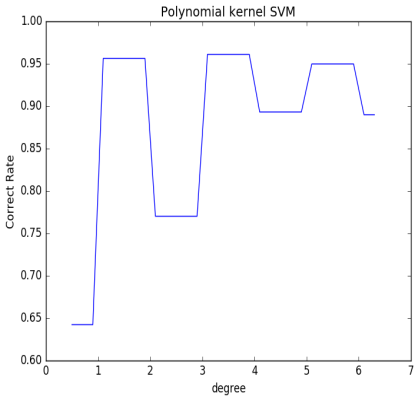
The relationship of tuning parameters and performances are shown in figure 2. For knn, we know form the graph when the neighbor size equals 9, the classifier has the best performance. For polynomial-kernel SVM, when the degree equals to 2 or the correct rate is as high as 95%. For RBF-kernel SVM, when the gamma parameter equals 0.25, the correct rate reaches more than 97%. For sigmoid-kernel SVM, the smaller the gamma parameter, the better the performance.



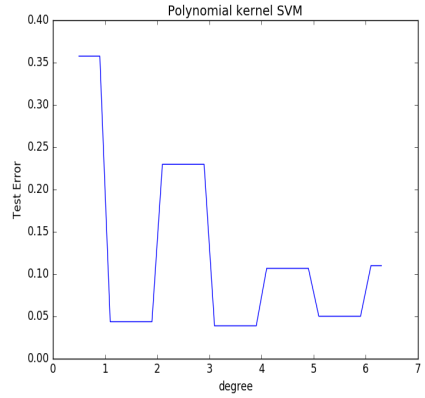
(a) KNN correct rate



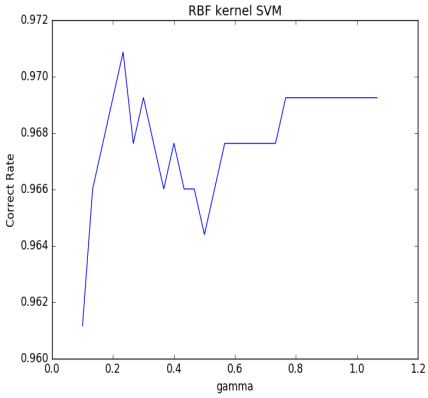
(b) KNN test error



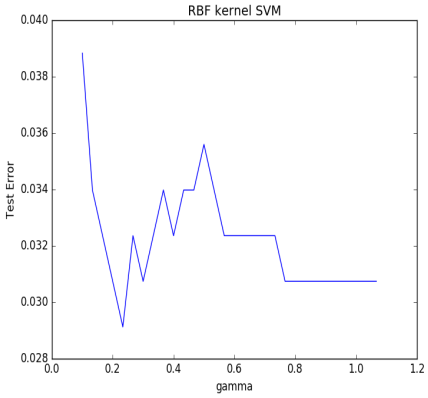
(c) polynomial SVM correct rate



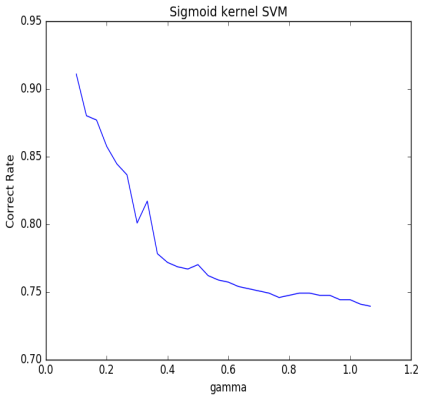
(d) polynomial SVM test error



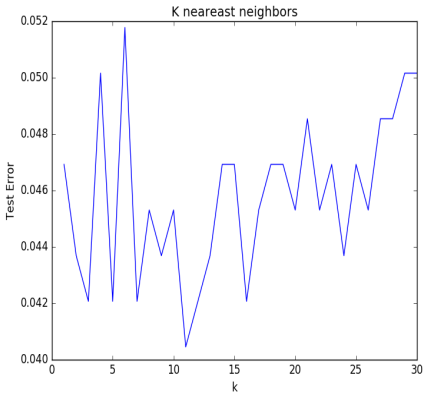
(e) RBF SVM correct rate



(f) RBF SVM test error



(g) Sigmoid SVM correct rate



(h) Sigmoid SVM test error

Figure 2: Evaluation metrics with tune parameter

3 Evaluation and conclusions

The table below shows the quality of differnent methods. According to the test results, we can see that RBF-kernel SVM with parameter gamma as 0.25 reaches the best performance with correct rate 98.54% and test error 0.0146. The worst method is Lasso which only have correct rate 65.7% and test error 0.3430. Except Lasso and QDA, almost all other methods can reach the correct rate at least 90%. The reason behind this fact is that the Laplacian kernel has already put the data with the same label together efficiently.

Prediction results				
Algorithm	Kernel	Parameters	Correct rate(%)	Test Error
Linear regression	Laplacian	—	94.33	0.0566
Ridge regression	Laplacian	—	95.95	0.0405
KNN	Laplacian	K=9	97.25	0.0275
SVM	linear	—	96.44	0.0356
SVM	polynomial	degree=3	95.79	0.2379
SVM	RBF	gamma=0.25	98.54	0.0146
SVM	sigmoid	gamma=0.1	91.10	0.0890
Naive bayes	Laplacian	—	92.72	0.0728
Logistic regression	Laplacian	—	96.76	0.0324
LDA	Laplacian	—	95.63	0.0437
Lasso	Laplacian	—	65.70	0.3430
Decision Tree	Laplacian	—	94.98	0.0502
Random Forest	Laplacian	—	95.63	0.03883
AdaBoost	Laplacian	—	96.28	0.0372
QDA	Laplacian	—	67.63	0.3236

Table 1: Prediction Results

To visualize the behaviors to different methods with different kernels, I draw the figure 3 to for users to see the differences of ten methods. These method are ridge regression, knn, linear SVM, RBF-kernel SVM, LDA, random forest, Adaboost, naive bayes, QDA and logistic regression. In figure 3, there are three rows, and each row represents the ten different methods applying on the same kernel. The first row is based on laplacian kernel. The second row is based on RBF kernel. The thrid row is based on sigmoid kernel. The colors in the graphs represent the boundaries of decisions. We can see that different methods have very different decision boundaries. Also the correct rates of each method with kernel are shown on the bottom right of each graph.

.

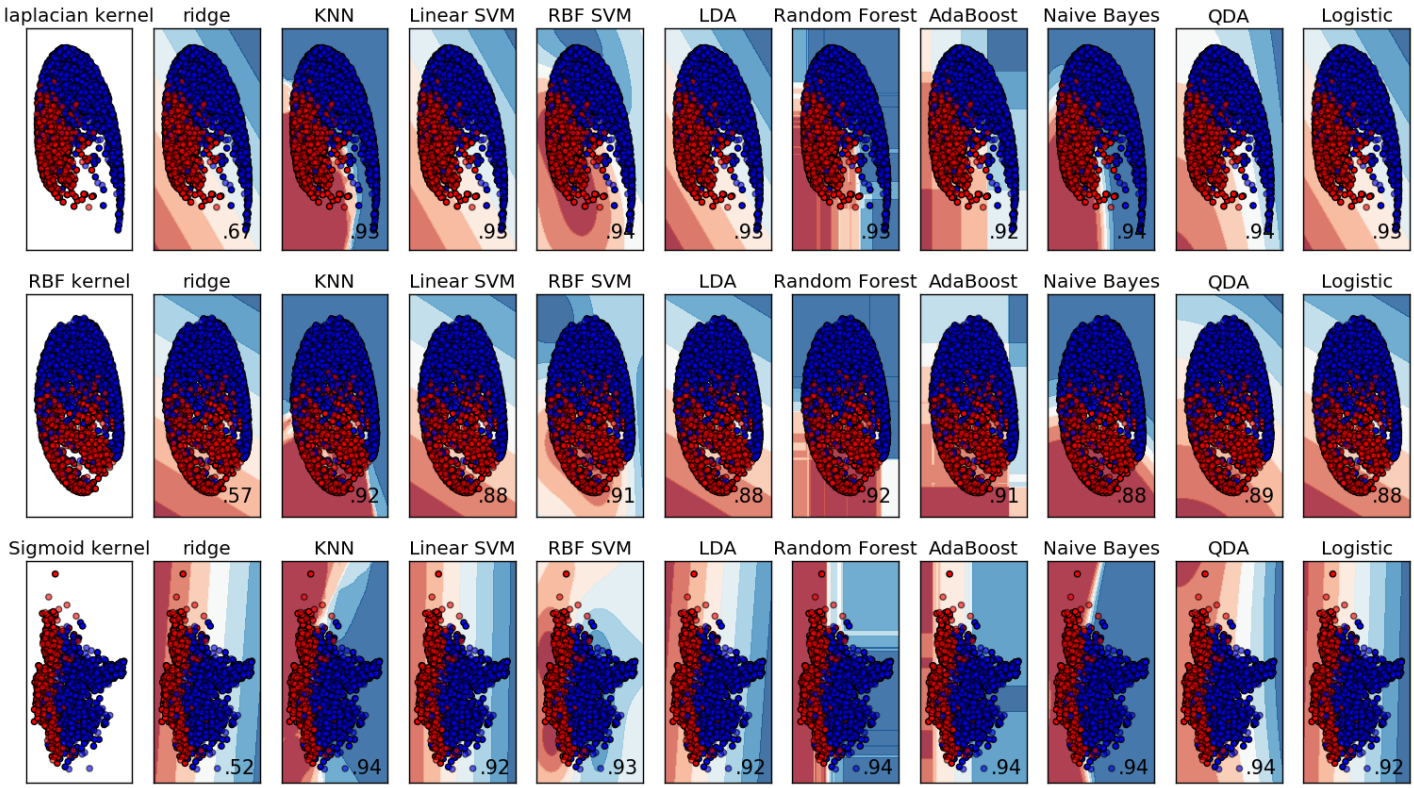


Figure 3: Algorithms comparison by applying different kernels

In summary, I think preprocess data with the appropriate kernel is the first but most important step for the predictions. As we have seen before, laplacian kernel and RBF kernel work very good in separating our data apart. The second step is trying different methods with appropriate parameter. As I show in part(2), tuning the parameters will significantly affect the prediction results. The third step is selecting the best algorithms with the optimal parameter. As the experiments I conduct, I will choose the RBF-kernel SVM with gamma parameter 0.25 as my final model.