Statistical Learning. LAB 2

Xijia Liu

Task 1: About Kernel PCA

In this task, we implement kernel PCA in R.

Task 3.1: Derive the following formula of calculating the kth kernel principal components for a new observation.

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

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, $i=1,\dots,n$, and $\mathbf{C}=n^{-1}\mathbf{1}_n\mathbf{1}_n^\top$. For details, check my slides.

Tips: read lecture notes for KPCA.

Task 1.2: Use the R code 'Task3-KPCA' to generate the data. Write your own program[1] 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 2: About Kernel Ridge Regression

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

[1]Using function from any package is not acceptable

Task 2.1: Implement kernel ridge regression (KRR) in R. Write an R function ’predictKRR’. The inputs should include

- ’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 built-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 2.2: 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 observations 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 2.3: 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 [2] 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 $(0.1, 0.01, 0.001)^\top$ 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 3: About SVM

In this task, you solve the handwritten digit recognition problem again. For this task, you should report all the details of your tuning procedure. R package ’kernlab’ is recommended. Please set the random seed as 8312 to split the data into training set (80%) and testing set (20%).

Task 3.1: Train an SVM with ’vanilladot’ kernel function with a nonzero slackness parameter C, i.e. linear soft margin classifier. We apply 10-fold cross-validation to tune the hyper-parameter C. The slackness parameter C can be proposed as $(0.0001, 0.0005, 0.001, 0.005, 0.1, 1)^\top$.

[2] You may check the potential issues of skipping this step.

In the end, pick up the ’optimal’ value for C, train the final model, and estimate the performance by using the testing set.

Task 3.2: Train a kernel SVM with ’rbfdot’ kernel function. Optimal your final model by tuning the hyper-parameters, i.e. slackness parameter C and kernel parameter sigma. Estimate the performance of your final model by using the testing set.