Chapter 5: K - Nearest Neighbours

- KNN classification
- KNN regression

K - Nearest Neighbours - Classfication

KNN is to classify a given observation to the class with highest estimated probability. Given a positive integer K and a test observation x_0, the KNN classifier first identifies the K points in the training data that are closest to x_0, represented by $\mathcal{N}_0$. It then estimates the conditional probability for class j as the fraction of points in $\mathcal{N}_0$ whose response values equal j:

$$\mathbb{P}(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} \mathbb{I}_{\{y_i = j\}}.$$

Finally, KNN applies Bayes rule and classifies the test observation x_0 to the class with the largest probability.

We will apply the KNN approach to the Caravan data set, which is part of the ISLR library. This data set includes 85 predictors that measure demographic characteristics for 5,822 individuals. The response variable is Purchase, which indicates whether or not a given individual purchases a caravan insurance policy. In this data set, only 6% of people purchased caravan insurance.

K - Nearest Neighbours - Classfication

For instance, imagine a data set that contains two variables, salary and age, measured in dollars and years, respectively. If we measured salary in Japanese yen, or if we measured age in minutes, then we'd get quite different classification results from what we get if these two variables are measured in dollars and years. A good way to handle this problem is to standardize the data so that all variables are given a mean of zero and a standard deviation of one. Then all variables will be on a comparable scale.

```
> library(ISLR)
> caravan = na.omit(Caravan) # laod the data and delete the missing
    values
> dim(caravan)
[1] 5822   86
> plot(caravan$Purchase)     # Proportion of the two classes
> table(caravan$Purchase)    # Count the Numbers
  No  Yes
5474  348

> y = caravan[,86]           # y - vecttor: Class Lable
> x = scale(caravan[,-86])   # x - standardized table: Variables
> n = nrow(caravan)          # No. of the observations
```

K - Nearest Neighbours - Classfication

The test data is a vector, with values from 1 to 2,000. The rest of data is regarded for training purposes. And try $K = 3$.

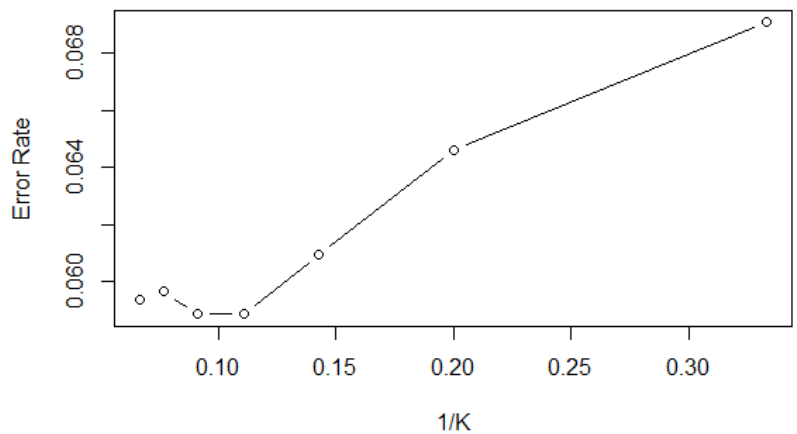
```
> test = 1:2000   # test set
> library(class)  # knn() - train and predict at the same time
> knn_pred_3 = knn(train = x[-test,],  # train dataset
+                  test = x[test,],    # test dataset
+                  cl = y[-test],      # class label of the train
+                  k = 3)
> (contTable3 = table(knn_pred_3, y[test]))

knn_pred_3   No  Yes
       No  1850  114
       Yes   29    7
```

When K = 3, the success rate increases to $7/(7 + 29) = 19.44\%$.

K - Nearest Neighbours - Classfication

Now we apply the cross validation to determine the value of K.

```
> k_test = seq(3, 15, 2) # odds numbers
> err = rep(NA, 7)
> set.seed(1234)
> for (i in 1:7) {
+   out = knn.cv(x[-test,], cl = y[-test], k = k_test[i])
+   err[i] = mean(out != y[-test])
+ }
> plot(k_test, err, type = "b", ylab = "Error Rate", xlab = "K")
```



K - Nearest Neighbours - Classfication

Using K = 7, the success rate increases to 2/(2+6) = 25%, and with K = 3 the rate is 19.44%. This is nearly four times the rate that results from random guessing. It appears that KNN is finding some real patterns in a difficult data set.

```
> knn_pred_7 = knn(train = x[-test,],   # train dataset
+                  test = x[test,],     # test dataset
+                  cl = y[-test],       # class label of the train
+                  k = 7)
> (contTable7 = table(knn_pred_7, y[test]))

knn_pred_7   No  Yes
       No  1873  119
       Yes    6    2
```

K - Nearest Neighbours - Classfication

As a comparison, we can also fit a logistic regression model to the data.

```
> dat = data.frame(y, x) # build a data frame using y and x
> logit_fit = glm(formula = y ~., # formula: y on all the rest
+                 family = "binomial", # Logit Regression
+                 data = dat,      # the dataset we are using
+                 subset = -test) # subset, the training data
>
> # predict() using the estimated model to predict on new dataset
> logit_prob = predict(logit_fit, # the estimated model
+                      newdata = dat[test,], # new dataset
+                      type = "response") # predicted probability
```

K - Nearest Neighbours - Classfication

If we instead predict a purchase any time the predicted probability of purchase exceeds 0.25, we predict that 63 people will purchase insurance, and we are correct for about $15/(15 + 48) = 23.8\%$ of these people.

```
> logit_pred = rep("No", length(test))
> logit_pred[logit_prob > 0.25] = "Yes"
> (contTable = table(logit_pred, y[test]))

logit_pred    No   Yes
       No   1831   106
       Yes    48    15
```

K - Nearest Neighbours - Regression

Linear regression is an example of a parametric approach because it assumes a linear functional form for $f(X)$. In contrast, non-parametric methods do not explicitly assume a parametric form for $f(X)$, and thereby provide an alternative and more flexible approach for performing regression. Here we consider one of the simplest and best-known non-parametric methods, K-nearest neighbours regression (KNN regression).

The KNN regression method is closely related to the KNN classifier discussed above. Given a value for $K$and a prediction point x_0, KNN regression first identifies the K training observations that are closest to x_0, represented by $\mathcal{N}_0$. It then estimates $f(x_0)$ using the average of all the training responses in $\mathcal{N}_0$. In other words,

$$\hat{f}(x_0) = \frac{1}{K} \sum_{x_i \in \mathcal{N}_0} y_i.$$

K - Nearest Neighbours - Regression

We will apply the KNN approach to the Advertising data set, which we have met in the regression section. This data set includes 3 predictors that advertising expenses in TV, Radio and Newspapers for 200 records. The response variable is Sales, which indicates whether or not the three predictors are associated with sales.

```
> # install.packages("FNN") - knn.reg()
> library(FNN)
> ad = read.csv(file = "Advertising.csv")[,-1] # Read in data
>
> y = ad$Sales          # y
> x = scale(ad[,-4])    # x - regressors
> n = nrow(ad)          # n - sample size
```

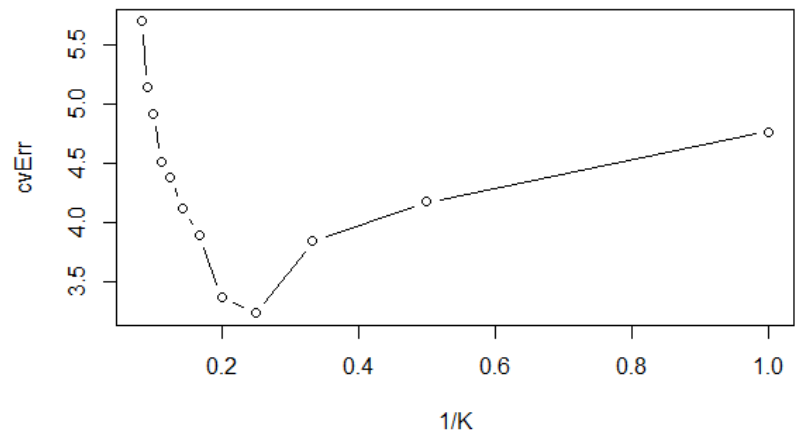
K - Nearest Neighbours - Regression

The knn.reg() function is used to fitting a data by KNN. We split the advertising data a forty-sixty ratio as test and training data respectively. To pick up the best K, we also need to apply cross validation.

```
> set.seed(1234)
> test = sample(1:n, size = 80, replace = F)
> cvErr = rep(x = 0, times = 12)
> for(i in 1:12){
+   knnReg = knn.reg(train = x[-test,],
+                    y = y[-test],
+                    k = i)
+   cvErr[i] = knnReg$PRESS / length(test)
+ }
> plot(1:12, cvErr, type = "b", xlab = "K")
```

K - Nearest Neighbours - Regression

Cross validation errors with respect to K. The minimum MSE for training data is at $k = 4$. And apply the model to the testing data, the the test MSE is 1.69.




```
> knnReg = knn.reg(train = x[-test,], test = x[test,],
+                  y = y[-test], k = 4)
> mean((knnReg$pred - y[test])^2)
[1] 1.69168
```

K - Nearest Neighbours - Regression

Now we apply multiple linear regression to fit the same data. The test MSE of multiple linear regression, 2.30, is much larger than the test MSE of KNN 1.69. We have to consider a question listed below.

```
> dat = data.frame(y, x)        # build a data frame using y and x
> ls_fit = lm(formula = y~., data = dat, subset = -test)
> ls_pred = predict(ls_fit, newdata = dat[test,])
> mean((y[test] - ls_pred)^2)
[1] 2.298314
```

In what setting will a parametric approach such as least squares linear regression (logistic regression) outperform a non-parametric approach such as KNN regression (KNN classification)?

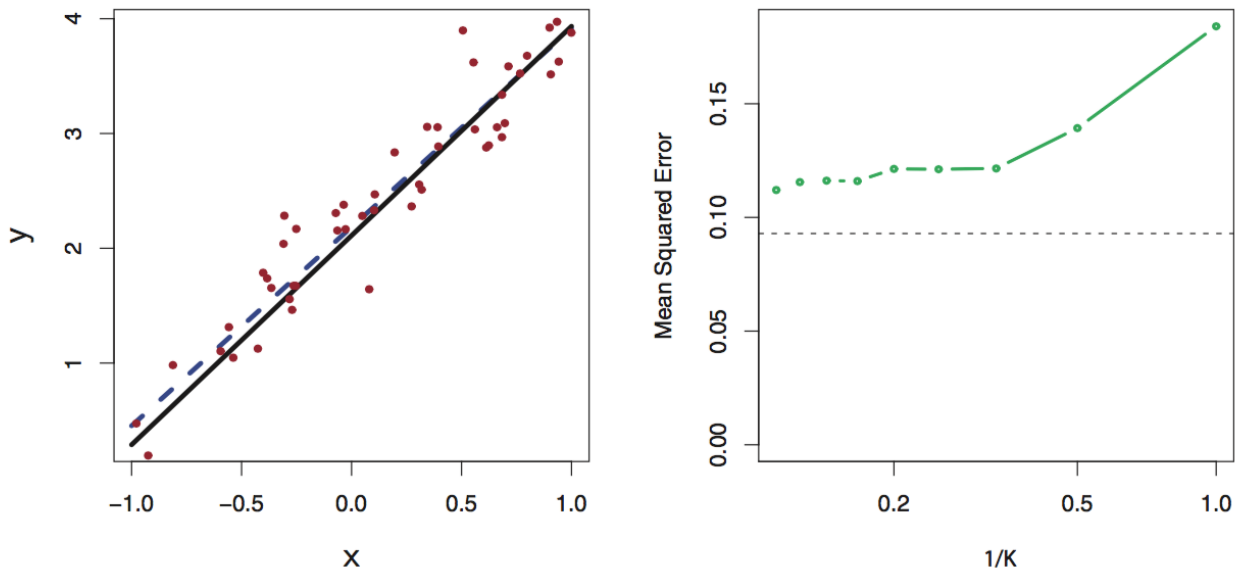
The answer is simple: the parametric approach will outperform the non-parametric approach if the parametric form that has been selected is close to the true form of f.

K - Nearest Neighbours - Comparison

In the left plot, the blue dashed line is the least squares fit to the data. Since $f(X)$ is in fact linear (displayed as the black line), the least squares regression line provides a very good estimate of $f(X)$.

In the right plot, the dashed horizontal line represents the least squares test set MSE, while the green solid line corresponds to the MSE for KNN as a function of 1/K (on the log scale).

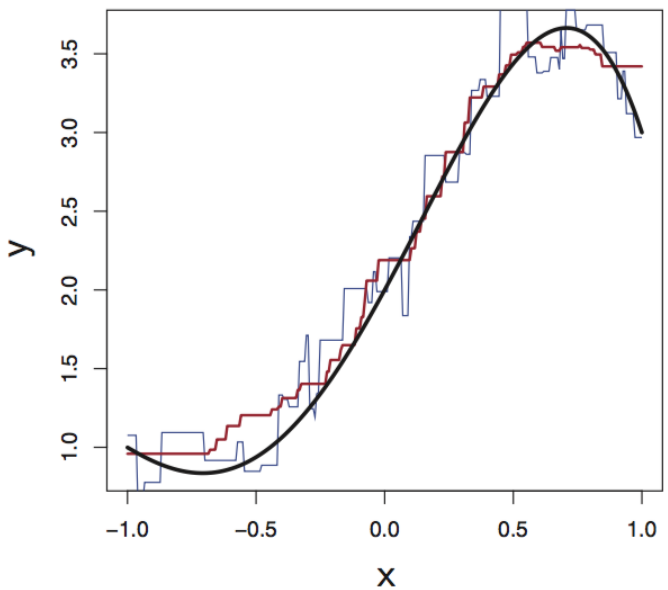
Linear regression achieves a lower test MSE than does KNN regression, since $f(X)$ is in fact linear. For KNN regression, the best results occur with a very large value of K, corresponding to a small value of $1/K$.

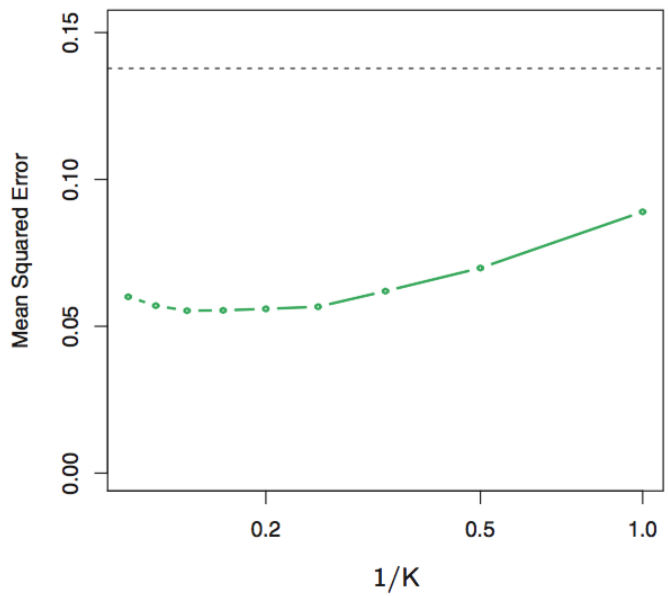


K - Nearest Neighbours - Comparison

In the left plot, a setting with a slightly non-linear relationship between X and Y (solid black line), the KNN fits with $K = 1$ (blue) and $K = 9$ (red) are displayed.

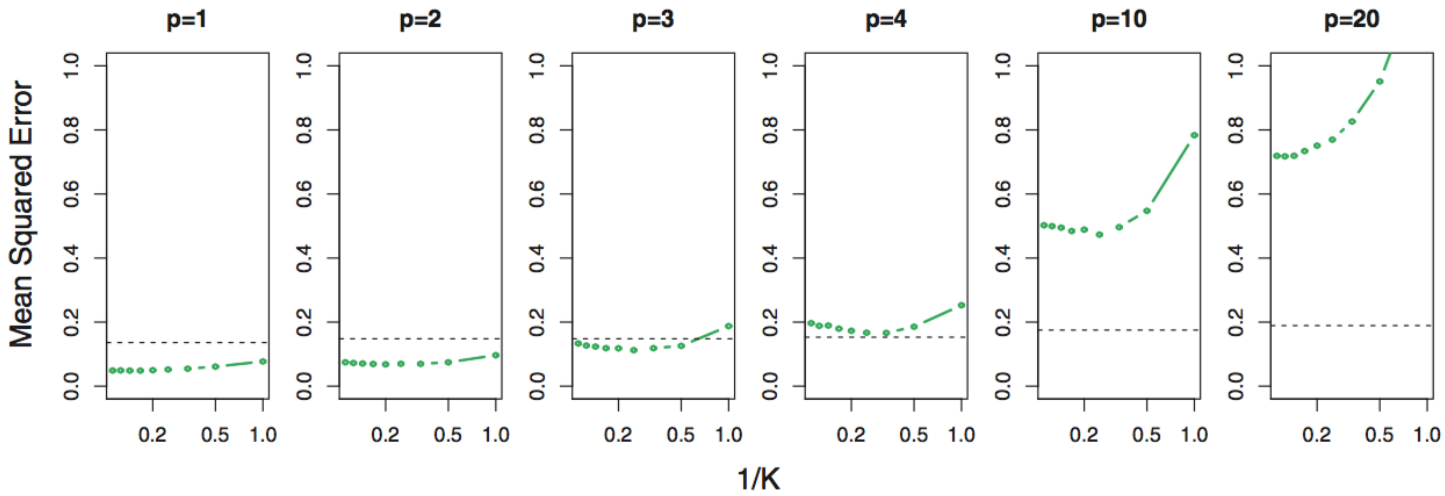
In the right plot, for the slightly non-linear data, the test set MSE for least squares regression (horizontal black) and KNN with various values of 1/K (green) are displayed.





K - Nearest Neighbours - Comparison

Test MSE for linear regression (black dashed lines) and KNN (green curves) as the number of variables p increases. The true function is non-linear in the first variable, as in the plots in previous slides, and does not depend on the additional variables. The performance of linear regression deteriorates slowly in the presence of these additional noise variables, whereas KNN's performance degrades much more quickly as p increases.



As a general rule, parametric methods will tend to outperform non-parametric approaches when there is a small number of observations per predictor.

K - Nearest Neighbours - Comparison

Even in problems in which the dimension is small, we might prefer linear regression to KNN from an interpretability standpoint. If the test MSE of KNN is only slightly lower than that of linear regression, we might be willing to forego a little bit of prediction accuracy for the sake of a simple model that can be described in terms of just a few coefficients, and for which p-values are available.