# MS1b Statistical Data Mining
# Part 3: Supervised Learning
# Nonparametric Methods

**Yee Whye Teh**
Department of Statistics
Oxford

http://www.stats.ox.ac.uk/~teh/datamining.html

# Outline

## Supervised Learning: Nonparametric Methods

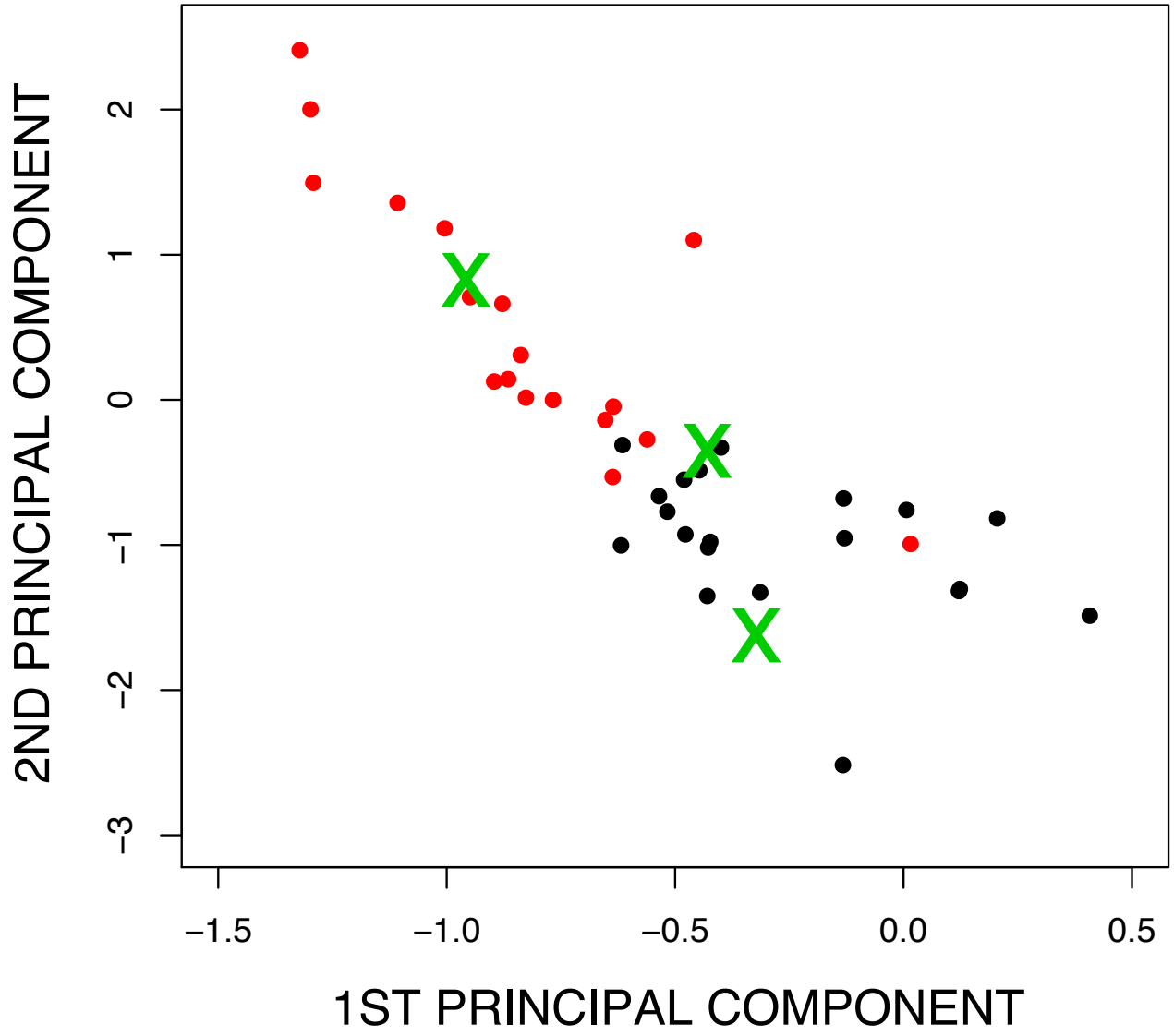
- Nearest Neighbours and Prototype Methods
- Learning Vector Quantization
- Classification and Regression Trees
- Determining Model Size and Parameters
- Neural Networks
- Lasso

# Outline

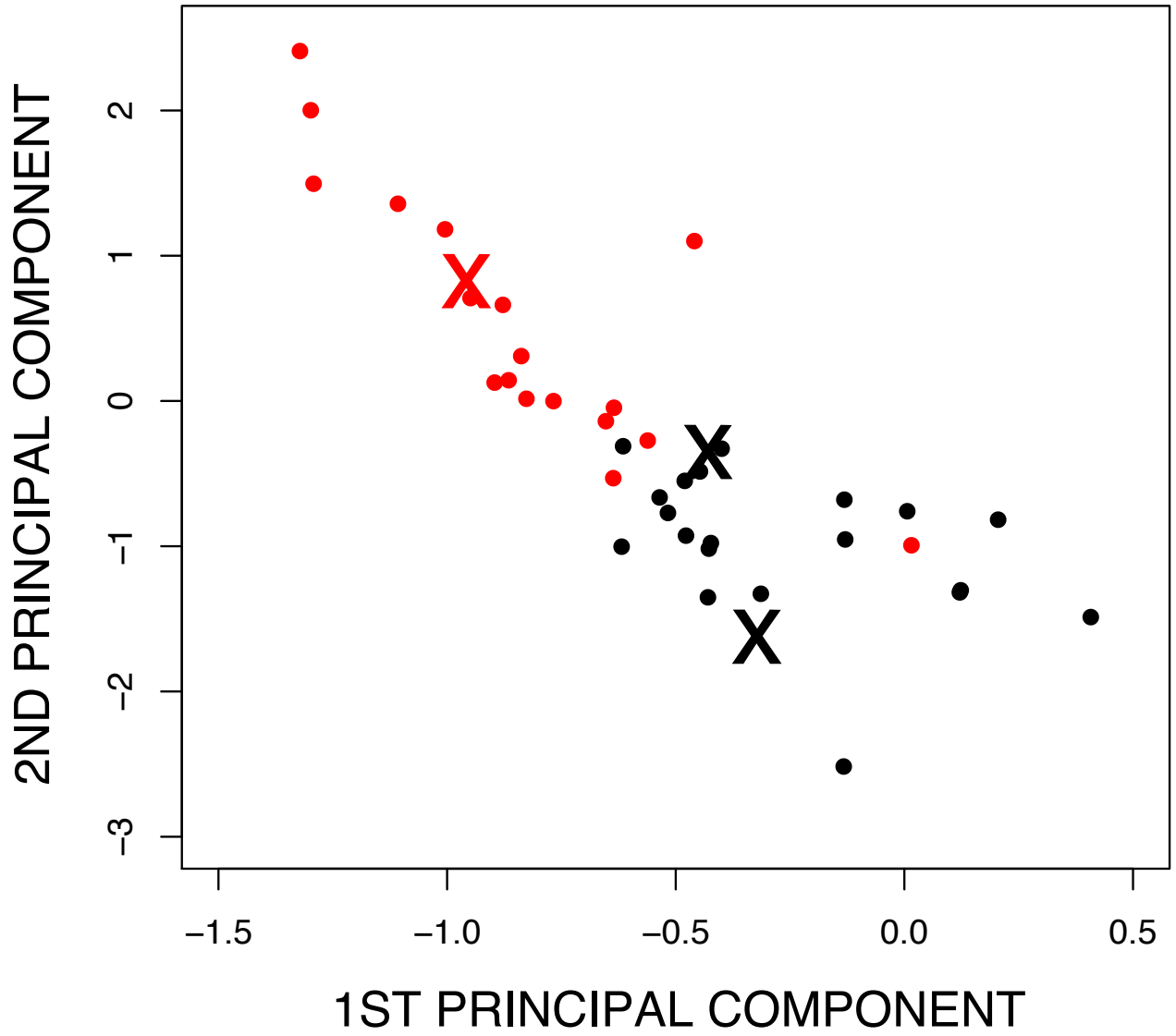
# k-Nearest Neighbours

- Nearest neighbours are simple and essentially model-free methods for classification.
- These methods are not very useful for understanding relationships between attributes and class predictions.
- Makes weaker modelling assumptions than e.g. LDA, Naïve Bayes and logistic regression.
- As *black box* classification methods however, they are often reasonable performers on real life problems (at least in lower-dimensional data) and provide a good benchmark as they are trivial to set up.

Example: Spam dataset in 2 dimensions. Using first 2 principal components of the predictor variables $X$ for illustration. Plotted are 50 emails (red: spam, black: no spam).
Task: predict category spam/no-spam for 3 new emails at the green crosses.



True categories.



Suppose again that the training data are $(X_i, Y_i)$ for $i = 1, \ldots, n$, where as usual $X_i \in \mathbb{R}^p$ and $Y_i \in \{1, \ldots, K\}$.
Assuming $\mathcal{X} \in \mathbb{R}^p$, Euclidean distance is often used to measure distance $d(X, X') = \|X - X'\|_2$. The nearest neighbours of a point $X$ is the set $ne_k(X) \subset \{1, \ldots, n\}$ such that

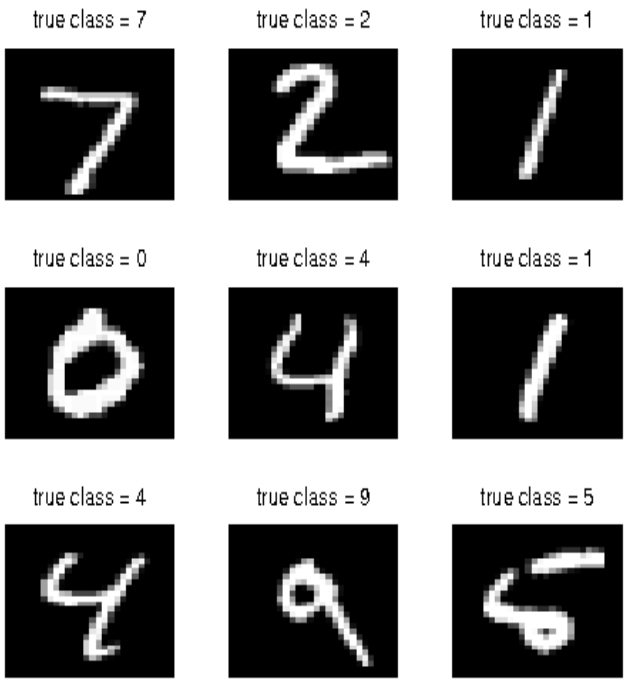
$$\max_{j \in ne_k(X)} d(X, X_j) \quad \leq \quad \min_{i \in \{1,\ldots,n\} \setminus ne_k(X)} d(X, X_i).$$

Given a new $X$, we find the $k$ observations in the training data 'closest' to $X$ and then classify using a majority vote amongst the $k$ neighbours (ties are broken at random – choose $k$ odd preferably).

$$\hat{Y}(X) = \text{argmax}_l \ |\{j \in ne_k(X) : Y_j = l\}|.$$

# Application to Handwritten Character Recognition

**Objective**: recognizing isolated (i.e., non-overlapping) digits, as in ZIP or postal codes.



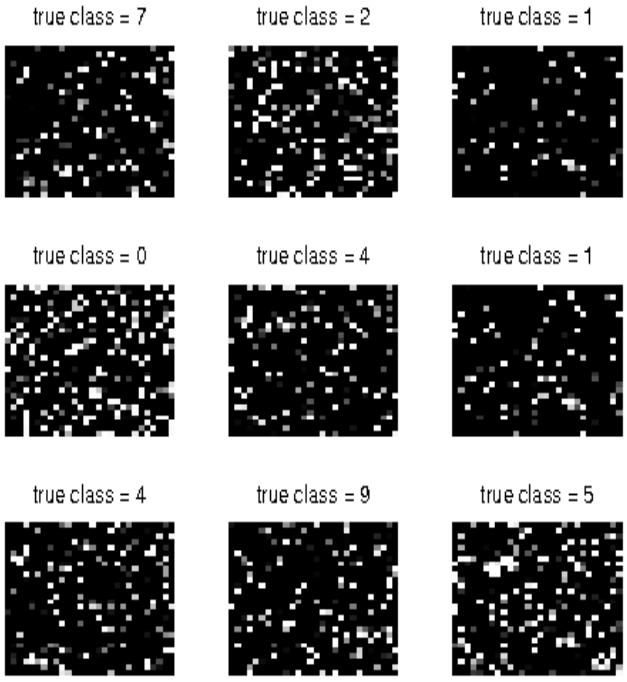

**Training and Test Data**: The MNIST15 dataset contains 60,000 training images and 10,000 test images of the digits 0 to 9, as written by various people.

**Details**: Images are $28 \times 28$ and have grayscale values in the range 0:255.

# Application to Handwritten Character Recognition

**Results**: 1-NN obtains a misclassification rate of only 3.09% on the test data using the Hamming distance!

This problem might look easy to you but remember that we do not use any spatial information. The K-NN classifier would obtain exactly the same results if the training and test data were permuted as it is invariant to the order of the features.



# Asymptotic Performance of 1 NN

Let $(X_i, Y_i)_{i=1}^n$ be some training data where $X_i \in \mathbb{R}^p$ and $Y_i \in \{1, 2, ..., K\}$. We define

$$\widehat{y}_{\text{Bayes}}(x) = \underset{l \in \{1,\ldots,K\}}{\arg\max}\ \pi_l f_l(x)$$

and

$$\widehat{y}_{\text{1NN}}(x) = y\,(\text{nearest neigbour of } x)\,.$$

Define

$$\begin{aligned} R_{\text{Bayes}} &= \mathbb{E}\left[\mathbb{I}\left(y \neq \widehat{y}_{\text{Bayes}}(x)\right)\right], \\ R_{\text{1NN}} &= \mathbb{E}\left[\mathbb{I}\left(y \neq \widehat{y}_{\text{1NN}}(x)\right)\right], \end{aligned}$$

then, as $n \to \infty$, we have the following powerful result

$$R_{\text{Bayes}} \leq R_{\text{1NN}} \leq 2R_{\text{Bayes}} - \frac{K}{K-1} R_{\text{Bayes}}^2.$$

- Despite its simplicity, $k$-NN is often a very good classifiers to use in a wide range of classification problems. At the very least, it is a good 'baseline' against which classifiers can be compared.
- This method allows for extremely flexible (nonparametric) decision boundaries.
- Due to the importance of distance to $k$-NN, it is important to standardise the data before use and find a suitable metric.
- It is also important to determine an appropriate $k$.

# Influence of k on Decision Boundaries

- $k$ determines the complexity of the decision boundary.
- For $k = 1$, we have no training error but are exposed to **overfitting**.
- Increasing $k$ yields smoother predictions, since we average over more data.
- For $k = n$, we predict the same output whatever being $X$.

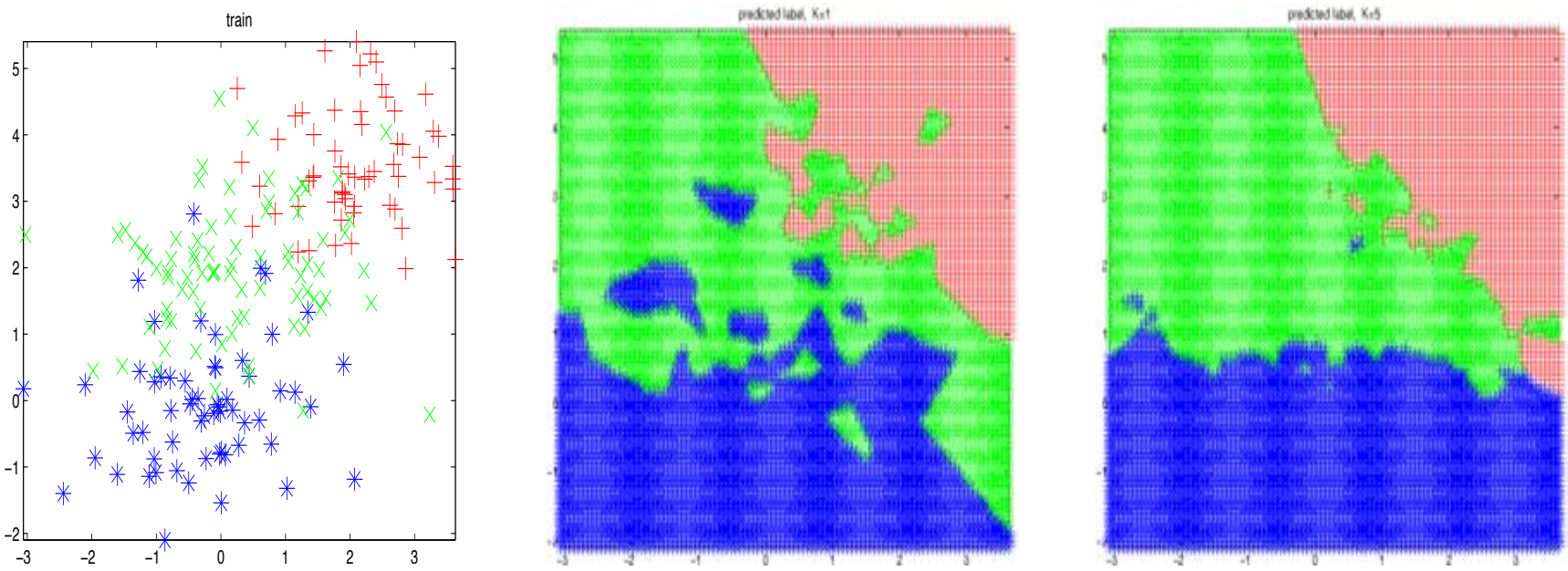



Figure: Training data (left), 1-NN (center) and 5-NN (right)

Using the SPAM dataset properly (without PC). Writing the code from scratch.

```
X <- scale(X)
knn <- 3

predicted <- numeric(length(test))
for (k in 1:length(test)){
  DIFF <- X[train,] -
          outer(rep(1,length(train)),X[test[k],],FUN="*")
  distance <- apply(DIFF^2,1,mean)
  nearestneighbors <- order(distance)[1:knn]
  predicted[k] <- mean(Y[train[nearestneighbors]])
}
```

Predict on the test set.

```
predicted_knn <- as.numeric(predicted > 0.5)
> table(predicted_knn, Y[test])

          actual  0    1
 predicted  0 1343  131
            1  110  818
```

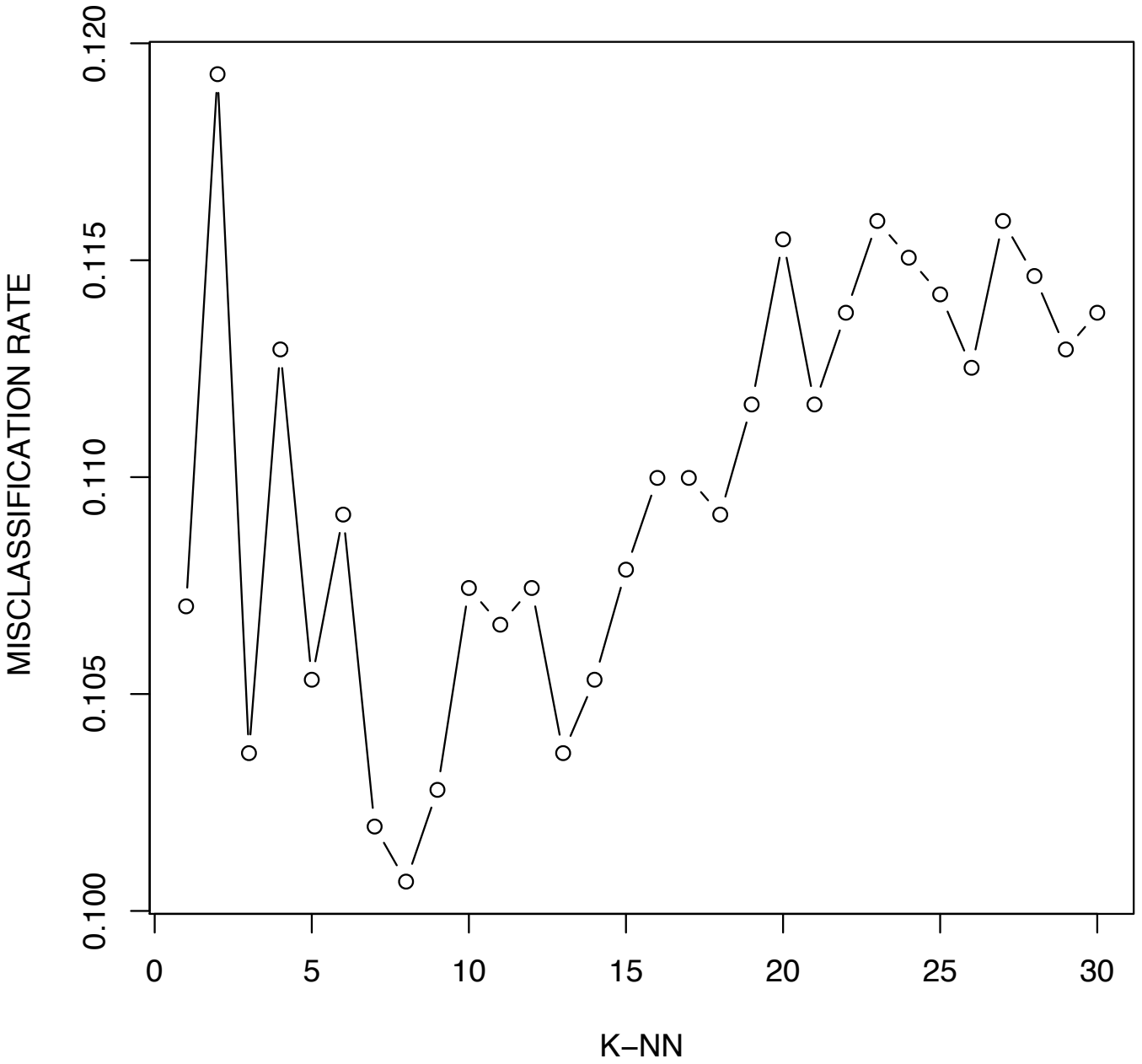
Using $k = 9$ nearest neighbours.

```
predicted_knn <- as.numeric(predicted > 0.5)
> table(predicted_knn, Y[test])

           actual 0    1
 predicted  0 1349  156
            1  104  793
```

Misclassification rate is thus about 10.8% for $k = 9$.

Compute misclassification rate as a function of $k$.



# K-means clustering

- A disadvantage of $k$-nn is the high memory requirement as each new observations has to be matched against *all* observations to find the nearest neighbour(s).
- As discussed before, K-means is a method for finding clusters and cluster centres in a set of unlabeled data. Considering each class in isolation, we can apply the K-means algorithm (each with $R$ clusters) to characterise observations from each class.
- These $K \times R$ labeled *prototypes* can be used to summarise the distribution of class data over $\mathcal{X}$. For a new observations $X$, we predict to the class with the nearest prototype. An advantage over $k$-NN is thus that much fewer observations/prototypes have to be kept in memory.

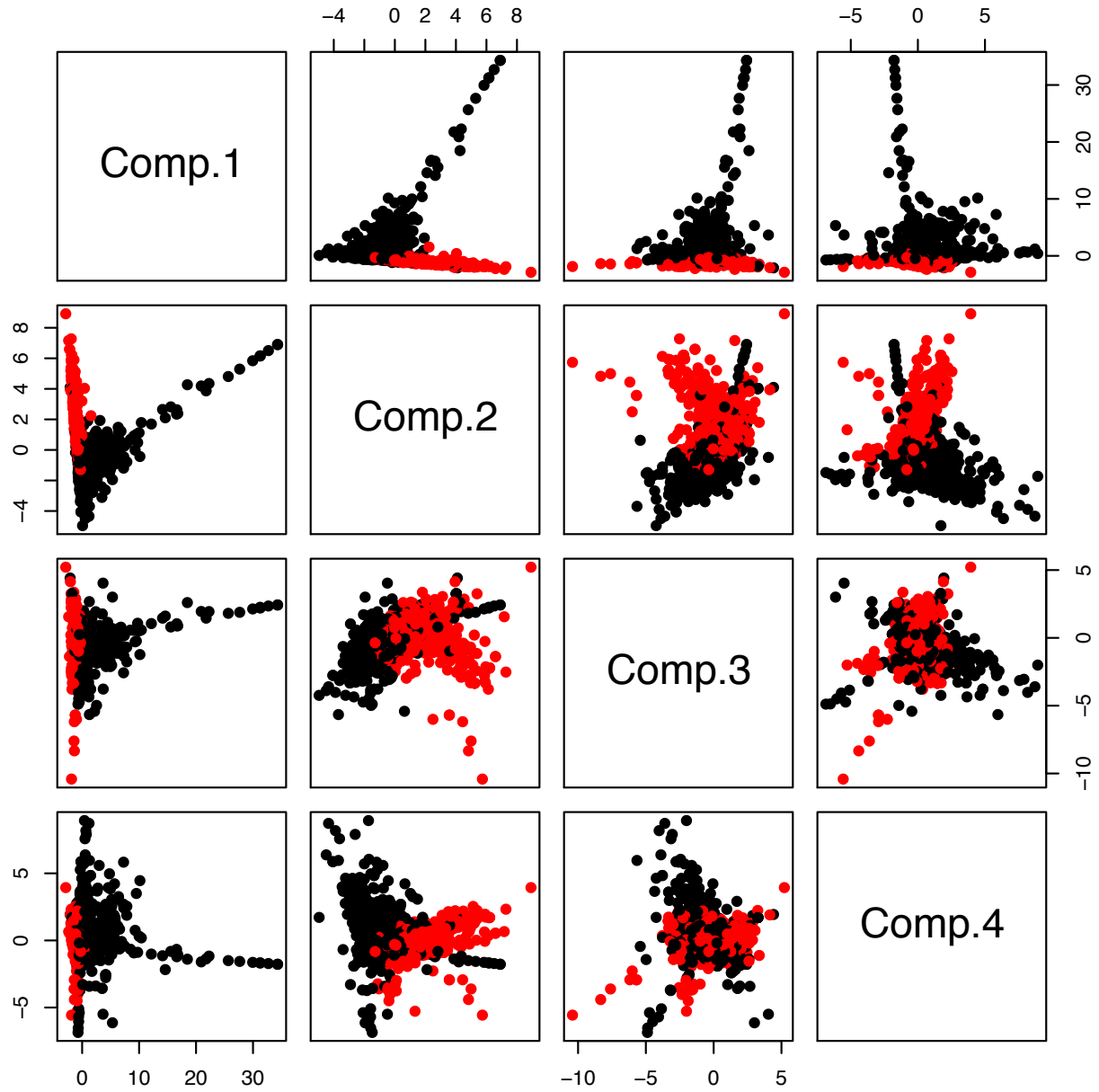
Plot Spam data in the space of the first 4 Principal Components.

```
library(kernlab)
data(spam)
n <- nrow(spam)
p <- ncol(spam)-1


Y <- as.numeric(spam[, p+1])-1
X <- predict(princomp(spam[,-(p+1)] ,cor=TRUE))[,1:4]

pairs(X,col=Y+1,cex=1.5,pch=20)
```

Red: spam
Black: not spam.



Now replace each class by 30 cluster centers.

```
nk <- 30
KMX <- kmeans(X[Y==0,], nk)
KMY <- kmeans(X[Y==1,], nk)


pairs( rbind(KMX$centers,KMY$centers),
           col= rep(1:2,each=nk), cex=1.5, pch=20)
```

And then use these as prototypes for $k$-NN classification.