

# *Principles of Statistical Machine Learning*

## *Introduction to kNearest Neighbors Learning*

*Ernest Fokoué*  
福尔特 教授

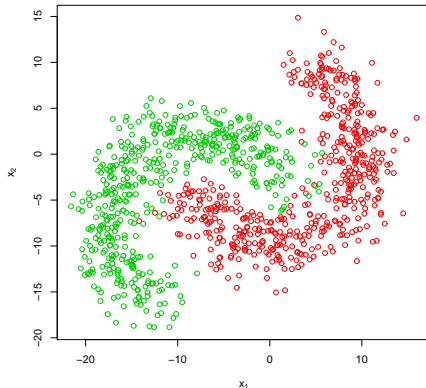
*School of Mathematical Sciences*  
*Rochester Institute of Technology*  
*Rochester, New York, USA*

*Statistical Machine Learning and Data Science*  
*African Institute for Mathematical Sciences (AIMS)*  
*Kigali (Rwanda)-January 2018*

*“To understand God’s thoughts, one must study statistics ... the measure of His purpose.”*  
*Florence Nightingale*

# Binary Classification in the Plane (2D space)

Consider the following two dimensional binary classification task.



Question: Can the two classes ever be separated by a line? If not, what is the "best" classifier here?

# Binary Classification in the Plane

*For the binary classification problem introduced earlier:*

- A collection  $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$  of i.i.d. observations is given
  - $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^2, i = 1, \dots, n$ .  $\mathcal{X}$  is the input space.
  - $y_i \in \{-1, +1\}$ .  $\mathcal{Y} = \{-1, +1\}$  is the output space.
- What is the probability law that governs the  $(\mathbf{x}_i, y_i)$ 's?
- What is the functional relationship between  $\mathbf{x}$  and  $y$ ?
- What is the "best" approach to determining from the available observations, the relationship between  $\mathbf{x}$  and  $y$  in such a way that, given a new (unseen) observation  $\mathbf{x}^{\text{new}}$ , its class  $y^{\text{new}}$  can be predicted as accurately as possible.

*“When the character of a man is not clear to you, look at his friends”*  
*Japanese Proverb*

# Intuition of $k$ -Nearest Neighbor Classification

To help gain insights into the principles of  $k$ Nearest Neighbors classification, we consider the deliberately very simple and straightforward 2D binary classification task shown below

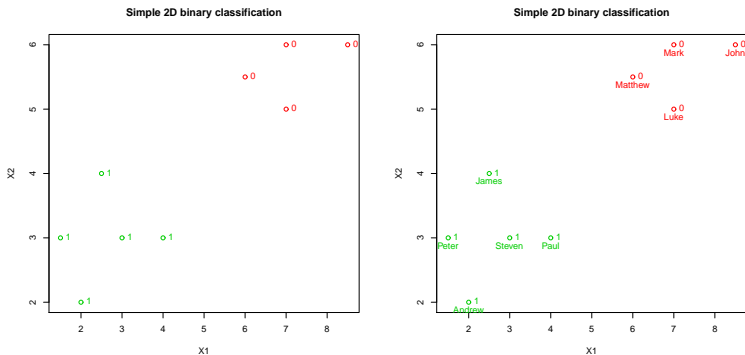
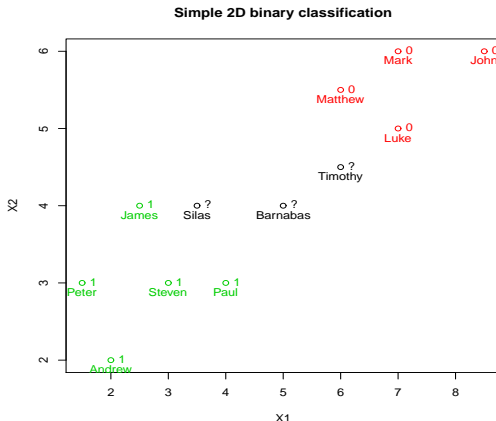


Figure: (left) Plot without names . (right) Plot with names.

# Intuition of $k$ -Nearest Neighbor Classification

**Task:** We are given new people {*Timothy*, *Barnabas*, *Silas*} as shown below, and we want to assign each one of them to their "correct" class.



We will herein build the so-called  $k$ -Nearest Neighbors ( $k$ NN) classifier.

# Intuition of $k$ -Nearest Neighbor Classification

- **$k$ -Nearest Neighbor Principle:** The reasonable class/category for a given object is the most prevalent class among its nearest neighbors
- **$k$ -Nearest Neighbor Steps:** Given a new point to be classified,
  - Choose a distance for measuring how far a given point is from another
  - Set the size of the neighborhood  $k$
  - Compute the distance from each existing point to the new point
  - Identify the class labels of the  $k$  points closest/nearest to the new point
  - Assign the most frequent label to the new point
- **$k$ -Nearest Neighbor Classification:** The estimated class of a vector  $x$  is the most frequent class label in the neighborhood of  $x$ .



*“Tell me who your friends are, and I will tell you who you are.”*  
*Mexican Proverb*

# Intuition of $k$ -Nearest Neighbor Classification

We consider applying the  $k$ NN algorithm to the above simple example. First, let's use  $k$ NN to find the neighbors of *Timothy* = (6.0, 4.5).

*Matthew* is his nearest neighbor. *Luke* is his second nearest neighbor.

*Mark* is his third nearest neighbor.

	X1	X2	class	dist	rank
Peter	1.5	3.0	1	22.50	9
Andrew	2.0	2.0	1	22.25	8
James	2.5	4.0	1	12.50	7
Paul	4.0	3.0	1	6.25	4
Steven	3.0	3.0	1	11.25	6
Matthew	6.0	5.5	0	1.00	1
Mark	7.0	6.0	0	3.25	3
Luke	7.0	5.0	0	1.25	2
John	8.5	6.0	0	8.50	5

**Question:** What is the correct class for *Timothy*?

# Intuition of $k$ -Nearest Neighbor Classification

What are the 3 nearest neighbors of *Barnabas* = (5.0, 4.0)?

	X1	X2	class	dist	rank
Peter	1.5	3.0	1	13.25	8
Andrew	2.0	2.0	1	13.00	7
James	2.5	4.0	1	6.25	5
Paul	4.0	3.0	1	2.00	1
Steven	3.0	3.0	1	5.00	3
Matthew	6.0	5.5	0	3.25	2
Mark	7.0	6.0	0	8.00	6
Luke	7.0	5.0	0	5.00	4
John	8.5	6.0	0	16.25	9

*Question:* What is the correct class for *Barnabas*?

# Intuition of $k$ -Nearest Neighbor Classification

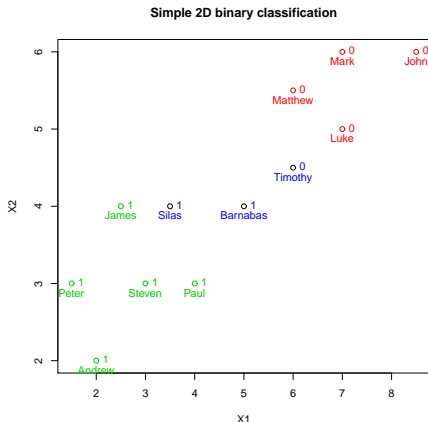
Finally, determine the 4 nearest neighbors of *Silas* = (3.5, 4.0)

	X1	X2	class	dist	rank
Peter	1.5	3.0	1	5.00	4
Andrew	2.0	2.0	1	6.25	5
James	2.5	4.0	1	1.00	1
Paul	4.0	3.0	1	1.25	2
Steven	3.0	3.0	1	1.25	3
Matthew	6.0	5.5	0	8.50	6
Mark	7.0	6.0	0	16.25	8
Luke	7.0	5.0	0	13.25	7
John	8.5	6.0	0	29.00	9

*Question:* What is the correct class for *Silas*?

# Intuition of $k$ -Nearest Neighbor Classification

The final classification of all the three new individuals is below



We now give more ample description of the  $k$ Nearest Neighbors Algorithm.

# Some Distances Used in kNN classification

*Three of the most commonly used distances are:*

- Euclidean distance: *also known as the  $\ell_2$  distance*

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{l=1}^q (x_{il} - x_{jl})^2} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$$

- Manhattan distance (city block): *also known as  $\ell_1$  distance*

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^q |x_{il} - x_{jl}| = \|\mathbf{x}_i - \mathbf{x}_j\|_1$$

- Maximum distance: *also known as the infinity distance*

$$d(\mathbf{x}_i, \mathbf{x}_j) = \max_{l=1, \dots, q} |x_{il} - x_{jl}| = \|\mathbf{x}_i - \mathbf{x}_j\|_\infty$$

*Other distances:* Minkowski; canberra; binary or Jaccard.

# Additional Distances for $k$ Nearest Neighbors

Given two vectors  $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^q$ , The most common distances are

- Minkowski distance:  $\ell_p$  distance

$$d(\mathbf{x}_i, \mathbf{x}_j) = \left\{ \sum_{\ell=1}^q |\mathbf{x}_{i\ell} - \mathbf{x}_{j\ell}|^p \right\}^{1/p}$$

- Canberra distance:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{\ell=1}^q \frac{|\mathbf{x}_{i\ell} - \mathbf{x}_{j\ell}|}{|\mathbf{x}_{i\ell} + \mathbf{x}_{j\ell}|}$$

- Jaccard/Tanimoto distance: For binary vectors ie  $\mathbf{x}_i \in \{0, 1\}^q$

$$d(\mathbf{x}_i, \mathbf{x}_j) = 1 - \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{|\mathbf{x}_i|^2 + |\mathbf{x}_j|^2 - \mathbf{x}_i \cdot \mathbf{x}_j}$$

$$\mathbf{x}_i \cdot \mathbf{x}_j = \sum_{\ell=1}^q \mathbf{x}_{i\ell} \mathbf{x}_{j\ell} = \sum_{\ell=1}^q \mathbf{x}_{i\ell} \wedge \mathbf{x}_{j\ell} \text{ and } |\mathbf{x}_i|^2 = \sum_{\ell=1}^q \mathbf{x}_{i\ell}^2$$

## *k*-Nearest Neighbors (*k*NN) classification

$\mathcal{D} = \left\{ (\mathbf{x}_1, Y_1), \dots, (\mathbf{x}_n, Y_n) \right\}$ , with  $\mathbf{x}_i \in \mathcal{X}^q$ ,  $Y_i \in \{1, \dots, g\}$ .

- 1 Choose the value of  $k$  and the distance to be used
- 2 Let  $\mathbf{x}^*$  be a new point. Compute

$$d_i^* = d(\mathbf{x}^*, \mathbf{x}_i) \quad i = 1, \dots, n$$

- 3 Rank all the distances  $d_i^*$  in increasing order

$$d_{(1)}^* \leq d_{(2)}^* \leq \dots \leq d_{(k)}^* \leq d_{(k+1)}^* \leq \dots \leq d_{(n)}^*$$

- 4 Form  $\mathcal{V}_k(\mathbf{x}^*)$ , the  $k$ -Neighborhood of  $\mathbf{x}^*$

$$\mathcal{V}_k(\mathbf{x}^*) = \left\{ \mathbf{x}_i : d(\mathbf{x}^*, \mathbf{x}_i) \leq d_{(k)}^* \right\}$$

- 5 Compute the predicted response  $\hat{Y}^*$  as

$$\hat{Y}_{\text{kNN}}^* = \text{Most frequent label in } \mathcal{V}_k(\mathbf{x}^*)$$



# *k*-Nearest Neighbors (*k*NN) classification

- ① Compute the predicted response  $\hat{Y}^*$  as

$$\hat{Y}_{\text{kNN}}^* = \hat{f}_{\text{kNN}}(\mathbf{x}^*) = \underset{j \in \{1, \dots, g\}}{\operatorname{argmax}} \left\{ p_j^{(k)}(\mathbf{x}^*) \right\}$$

where

$$p_j^{(k)}(\mathbf{x}^*) = \frac{1}{k} \sum_{\mathbf{x}_i \in \mathcal{V}_k(\mathbf{x}^*)} I(Y_i = j)$$

estimates the probability that  $\mathbf{x}^*$  belongs to class  $j$  based on  $\mathcal{V}_k(\mathbf{x}^*)$ .

- ② **Posterior probability estimate:** Indeed,  $\frac{1}{k} \sum_{\mathbf{x}_i \in \mathcal{V}_k(\mathbf{x}^*)} I(Y_i = j)$  can be thought of as a rough estimate of  $\pi_j(\mathbf{x}^*) = \Pr[Y^* = j | \mathbf{x}^*]$ , the posterior probability of class membership of  $\mathbf{x}^*$ , ie

$$p_j^{(k)}(\mathbf{x}^*) = \frac{1}{k} \sum_{\mathbf{x}_i \in \mathcal{V}_k(\mathbf{x}^*)} I(Y_i = j) \approx \widehat{\pi_j(\mathbf{x}^*)}$$

# *k*-Nearest Neighbors (*k*NN) classification

*Some properties of kNN estimators include*

- *kNearest Neighbors (kNN) essentially performs classification by voting for the most popular response among the  $k$  nearest neighbors of  $x^*$ .*
- *kNN provides the most basic form of nonparametric classification*
- *Thanks to the fact that the estimated response  $\hat{Y}_{kNN}^*$  for  $x^*$  is - at least - a crude nonparametric estimator of Bayes classifier's response*
- *Since the fundamental building block of kNN is the distance measure, one can easily perform classification beyond the traditional setting where the predictors are numeric. For instance, classification with kNN can be readily performed on indicator attributes*

$$x_i = (x_{i1}, \dots, x_{ip})^\top \in \{0, 1\}^p$$

- *kNN classifiers are inherently naturally multi-class, and are used extensively in applications such as image processing, character recognition and general pattern recognition tasks*

# *k*-Nearest Neighbors (*k*NN) classification

*Limitations of the basic k-Nearest Neighbors approach:*

- ① **Equidistance:** *All neighbors are given the same contribution to the estimate of the response; Indeed, in the estimated probability*

$$p_j^{(k)}(\mathbf{x}^*) = \frac{1}{k} \sum_{\mathbf{x}_i \in \mathcal{V}_k(\mathbf{x}^*)} I(Y_i = j) = \sum_{\mathbf{x}_i \in \mathcal{V}_k(\mathbf{x}^*)} w_i I(Y_i = j)$$

*the weight  $w_i = \frac{1}{k} = \text{constant}$  for all points in  $\mathcal{V}_k(\mathbf{x}^*)$  regardlessly of how far they are from  $\mathbf{x}^*$ .*

- ② **No model, no interpretability:** *There is no underlying model, therefore no interpretation of the response relative to the predictor variables. There is no training per se, since all happens at prediction. For this reason, kNN is referred to as **lazy method**.*
- ③ **Computationally intensive:** *Predictions are computationally very intensive, due to the fact that for each new observation, the whole dataset must be traversed to compute the response*

# Weighted $k$ Nearest Neighbors Classification

$k$ NN classification can be improved by weighting the votes as a function of the distance from  $\mathbf{x}^*$ . Some of the common weighting schemes include:

- *Exponential decay:*

$$w_i = \frac{e^{-d_i^*}}{\sum_{l=1}^k e^{-d_l^*}}$$

- *Inverse distance:*

$$w_i = \frac{\frac{1}{1+d_i^*}}{\sum_{l=1}^k \frac{1}{1+d_l^*}}$$

The weights are defined so as to preserve convexity, namely  $\sum_{i=1}^k w_i = 1$

**Question:** How can non convexity affect the weighted  $k$ NN method?

# *k*-Nearest Neighbors Classification Software

*As we said earlier, one of the greatest advantages of the *k*-NN classifier is its simplicity. The R command for it is also pretty straightforward and is found in the R package [class](#).*

```
library(class)
knn(train, test, cl, k = 1, l = 0, prob = FALSE)
```

*k*-Nearest Neighbors Algorithm on the famous Pima Indian Diabetes

```
library(MASS)
p <- ncol(Pima.tr)-1
Xtr <- Pima.tr[,-(p+1)]
Xte <- Pima.te[,-(p+1)]
Ytr <- Pima.tr[, (p+1)]
Yte <- Pima.te[, (p+1)]

Ytr.hat <- knn(Xtr, Xtr, Ytr, k=5)
Yte.hat <- knn(Xtr, Xte, Ytr, k=5)
```

*“When it comes to skills or character, you often tend to be as good or as bad as your constant companions are”*

*Unknown*

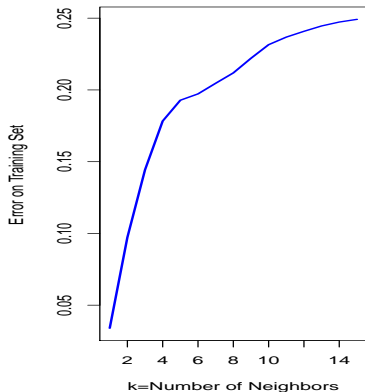
# Effect of $k$ on $k$ -Nearest Neighbors Classification

- ① *If  $k$  is small, ie the estimated class of  $x^*$  is determined based on very few neighbors, the resulting  $k$ NN classifier will have very low bias, but very high variance. In fact, in the limit, if  $k=1$ , the decision boundary will perfectly separate the classes on the training set (think of Mr Memory), but will perform poorly on the test set*
- ② *If  $k$  is large, ie the estimated class of  $x^*$  is determined based on very many neighbors from far and wide, the resulting  $k$ NN classifier will have very large bias, but very low variance. In fact, for truly large  $k$ , the decision boundary will be a constant hyperplane*
- ③ *We see the need for the determination of an optimal  $k$ , one that achieves the trade-off between bias and variance.*
  - ① *Determine  $k$  by cross validation*
  - ② *Determine  $k$  by direct minimization of the estimated prediction error via a suitably chosen test set*

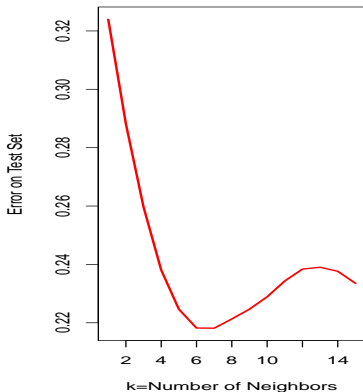
# Effect of $k$ on $k$ -Nearest Neighbors Classification

The following uses both the training and test sets of the Pima Indian data

**Train Error as a function of  $k$**



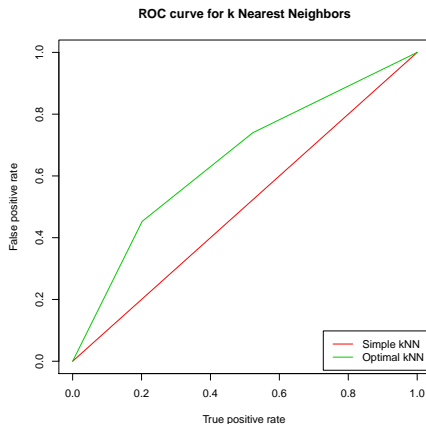
**Test Error as a function of  $k$**



*Training Error is overly optimistic. Test error is more realistic.*



# Effect of $k$ on $k$ -Nearest Neighbors Classification



*The optimistic classifier based of  $k = 1$  cannot even beat random guessing.*

# Various Effects in $k$ -Nearest Neighbors Classification

- 1 **Effect of  $k$ :** It is now clear that in  $k$ NN,  $k$  controls the complexity of the underlying classifier, with small  $k$  yielding very complex classifiers and large  $k$  yielding rather simple ones
- 2 **Effect of  $n$ :** The sample size  $n$  plays a crucial role because  $k$ NN being a lazy method, all happens at prediction. A large  $n$  would therefore lead to intense prediction
- 3 **Effect of  $p$ :** The dimensionality  $p$  of the input space is only felt by the function that computes the distances. If the function is optimized,  $k$ NN should be unaffected by this dimensionality
- 4 **Effect of distance:** It is known that some distances are more robust to extreme observations

# Distinct Strengths/Pros of Nearest Neighbors Approach

- ① The  $k$ NN method is intuitively appealing and very easy to understand, explain, program/code and interpret
- ② The  $k$ NN method provides a decent estimate of  $\Pr[Y = j|\mathbf{x}]$ , the posterior probability of class membership
- ③ The  $k$ NN method easily handles missing values (by restricting distance calculations to subspace)
- ④ As the number of training samples grows larger, the asymptotic misclassification error rate is bounded by twice the Bayes risk.

$$\lim_{n \rightarrow \infty} R(\hat{f}_n^{(kNN)}) \leq 2R^*$$

where  $R^*$  is the Bayes risk, that is, the small possible error, the error made by the generator of the data.

- ⑤ The  $k$ NN method is naturally suitable for sequential/incremental machine learning
- ⑥ The  $k$ NN method is also suitable where the hypothesis space is variable in size

## 定理

**(Vapnik and Chervonenkis, 1971)** Let  $\mathcal{H}$  be a class of functions implementing so learning machines, and let  $\zeta = VCdim(\mathcal{H})$  be the VC dimension of  $\mathcal{H}$ . Let the theoretical and the empirical risks be defined as earlier and consider any data distribution in the population of interest. Then  $\forall f \in \mathcal{H}$ , the prediction error (theoretical risk) is bounded by

$$R(f) \leq \hat{R}_n(f) + \sqrt{\frac{\zeta \left( \log \frac{2n}{\zeta} + 1 \right) - \log \frac{\eta}{4}}{n}} \quad (1)$$

with probability of at least  $1 - \eta$ .

# *Distinct Strengths/Pros of Nearest Neighbors Approach*

- ① *The kNN method serves a basic and easy to understand foundational machine learning and data mining technique*
- ② *The kNN method is an excellent baseline machine learning technique, and also allows many extensions*
- ③ *The kNN method can handle non-numeric data as long as the distance can be defined*
- ④ *The kNN method is usually performs reasonable well or sometimes very well when compared to more sophisticated techniques*
- ⑤ *kNN methods can handle mixed types of data as long as the distance are computed as hybrid or combinations*
- ⑥ *The kNN method is inherently multi-class, and this is very important because for some other methods, going beyond binary classification requires some sophisticated mathematics. It also handles very flexible decision boundaries*

# Distinct Weaknesses/Cons of Nearest Neighbors Approach

- ① The computational complexity of  $k$ NN is very high in prediction. Specifically, it is  $\mathcal{O}(nmp)$  where  $n$  is the training set size,  $m$  is the test set size and  $p$  is the number of predictor variables. This means that  $k$ NN requires large amount of memory, and therefore does NOT scale well. This failure in scalability is address using various heuristics and strategies
- ②  $k$ NN methods suffer from the Curse of Dimensionality (COD). When  $p$  is large and  $n$  is small, especialy in the context of the so-called short fat data where  $p \ggg n$ ,
  - The concept of nearness becomes meaningless to the point of being ill-defined when the dimension of the space is very high, because the "neighborhood" becomes very large
  - In a sense, one's nearest neighbor could be very far when the space is high dimensional and there are very few observations

# Distinct Weaknesses/Cons of Nearest Neighbors Approach

- 1 The  $k$ NN method does not yield a model, and therefore no parameter to help explain why the method performs as it does
- 2 The  $k$ NN method is heavily affected by the local structure
- 3 The  $k$ NN method is very sensitive to both irrelevant and correlated features
- 4 Unless the distance is well chosen and properly calibrated,  $k$ NN methods will be sensitive to outliers and all sorts of noise in the data
- 5 Unless the distance is used in some way to weight the neighbors, more frequent classes will dominate in the determination of the estimated label. This means one has to be careful with  $k$ NN when one class has a far larger proportion of observations than the others
- 6 The measurement scale of each variable affect the  $k$ NN method more than most methods. This issue is usually tackled by simply standardizing, unitizing or cubizing/squeezing the data.

# Discriminant Adaptive Nearest Neighbors (DANN)

*A very well established extension of the kNN method in binary classification consists of finding a new metric in a larger neighborhood of the test point, thereby avoiding the pitfall that arises from local influence and the assumption of constant local class probabilities*

$$d(\mathbf{x}^*, \mathbf{x}_i) = (\mathbf{x}^* - \mathbf{x}_i)^\top \Sigma (\mathbf{x}^* - \mathbf{x}_i)$$

*where*

$$\Sigma = \mathbf{W}^{-1} \mathbf{B} \mathbf{W}^{-1}$$

*with*

- $\mathbf{W} = S_1^2 + S_2^2$  denoting the within class covariance
- $\mathbf{B} = (\bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_1)(\bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_1)^\top$  denoting the between class covariance

*A modified version of  $\Sigma$  helps avoid overstretching neighborhood, namely*

$$\Sigma = \mathbf{W}^{-1/2} [\mathbf{W}^{-1/2} \mathbf{B} \mathbf{W}^{-1/2} + \epsilon I] \mathbf{W}^{-1/2}$$

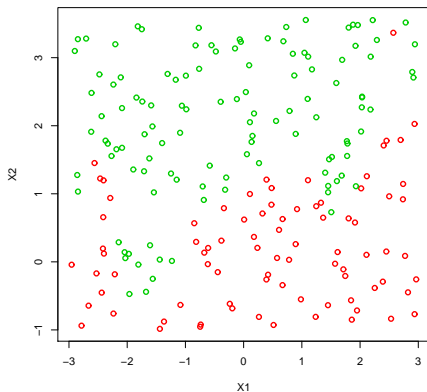
*Clearly,  $\Sigma = I$ , we have the Euclidean distance.*



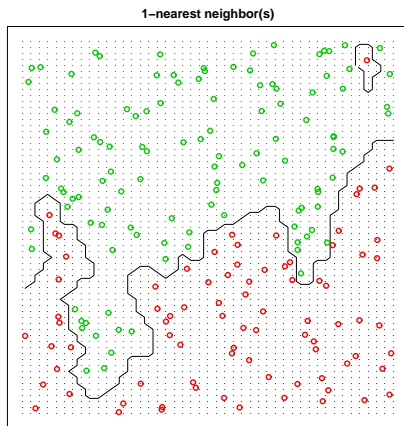
# First Two Dimensional Example

Let  $x_1 \in [-3.0, 3.0]$  and  $x_2 \in [-1.0, 3.6]$  and a decision boundary

$$\left\{ (x_1, x_2) : x_2 - \frac{1}{4}x_1^2 + \cos(x_1^2) + \sin\left(3 - \frac{1}{2}x_1\right) = 0 \right\}$$

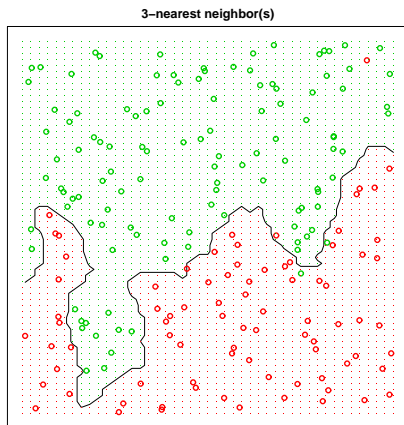


# *k*-Nearest Neighbors Decision Boundary



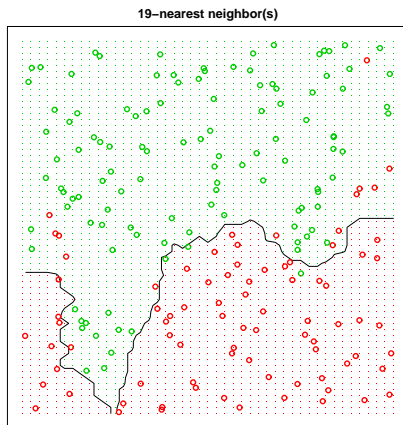
*This decision boundary is as complex as it can get for this data.*

# *k*-Nearest Neighbors Decision Boundary



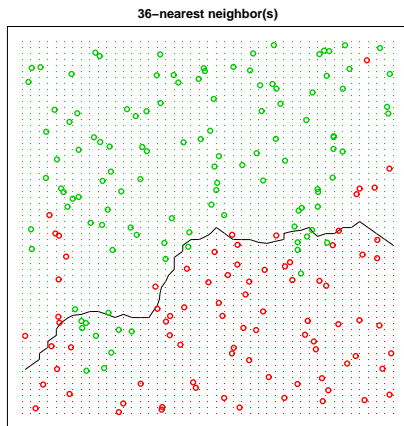
*This decision boundary is quite complex for this data.*

# *k*-Nearest Neighbors Decision Boundary



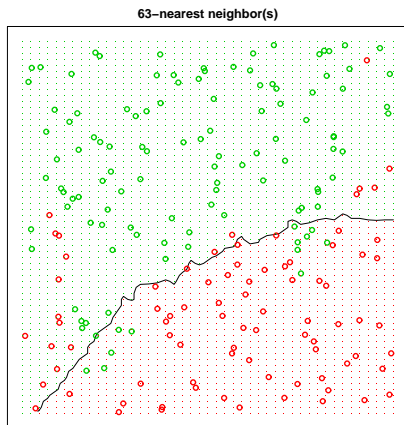
*This decision boundary is getting somewhat simpler.*

# *k*-Nearest Neighbors Decision Boundary



*This decision boundary is getting maybe too simple.*

# *k*-Nearest Neighbors Decision Boundary

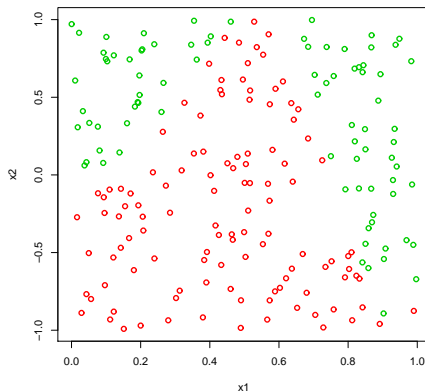


*This decision boundary is decidedly too simple*

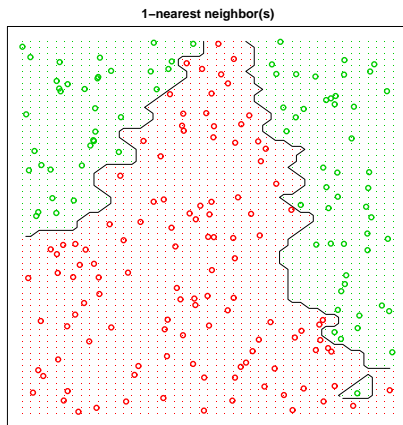
## Second Two Dimensional Example

Let  $x_1 \in [0, 1]$  and  $x_2 \in [-1, 1]$  and a decision boundary

$$\left\{ (x_1, x_2) : x_2 - \sin\left(\pi^{3/2}x_1^2\right) = 0 \right\}$$



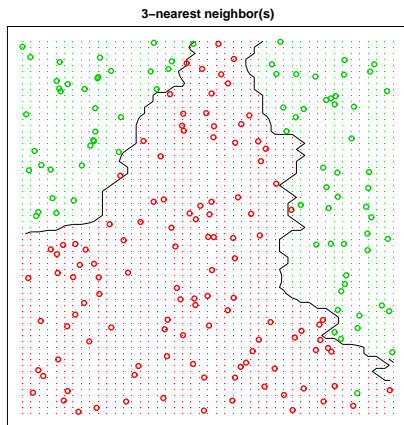
# *k*-Nearest Neighbors Decision Boundary



*This decision boundary is as complex as it can get for this data.*

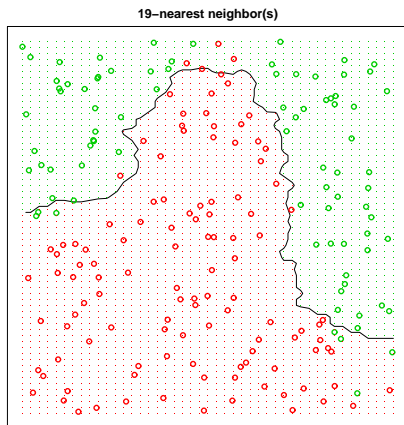


# *k*-Nearest Neighbors Decision Boundary



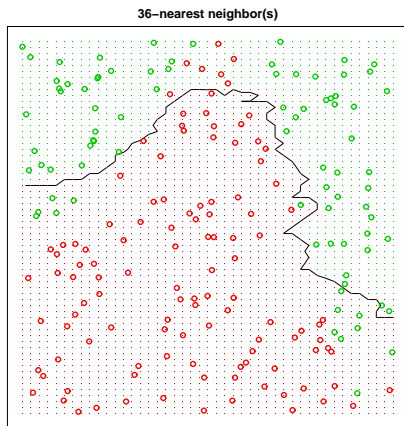
*This decision boundary is quite complex for this data.*

# *k*-Nearest Neighbors Decision Boundary



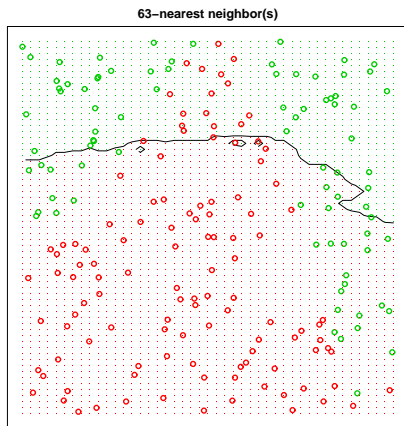
*This decision boundary is getting somewhat simpler.*

# *k*-Nearest Neighbors Decision Boundary



*This decision boundary is getting maybe too simple.*

# *k*-Nearest Neighbors Decision Boundary



*This decision boundary is decidedly too simple*

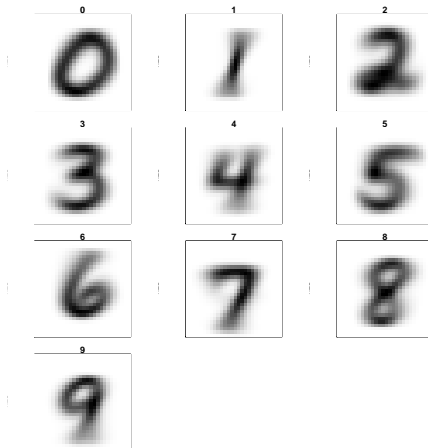
# Applications of the $k$ Nearest Neighbors Method

*The  $k$ -Nearest Neighbors approach to Machine Learning and Data Mining has been successfully applied to wide variety of important fields. Amongst others:*

- 1 The famous and somewhat ubiquitous **handwritten digit recognition** (See example below from with data taken Hastie, Tibshirani and Friedman). This is usually the first task in some Data Analytics competitions.
- 2 More recently, **text mining** and specific topic of **text categorization/classification** has made successful use of  $k$ Nearest Neighbors approach (See Assigned article)
- 3 **Credit Scoring** is another application that has been connected with  $k$  Nearest Neighbors Classification (See assigned Article)
- 4 **Disease diagnostics** also has been tackled using  $k$  Nearest Neighbors Classifiers

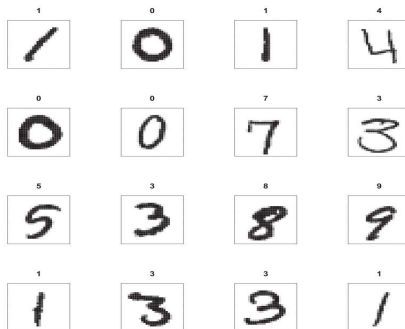
# Handwritten Digit Recognition

*Handwritten digit recognition is a fascinating problem that captured the attention of the machine learning and neural network community for many years, and has remained a benchmark problem in the field.*



# Handwritten Digit Recognition

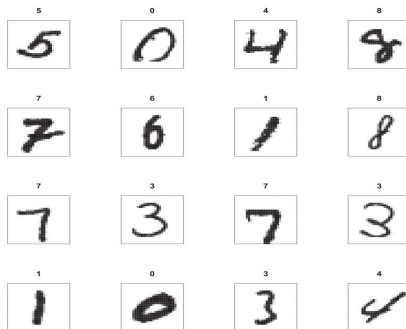
*Below is a portion of the benchmark training set*



**Note:** *The challenge here is building classification techniques that accurately classify handwritten digits taken from the test set.*

# Handwritten Digit Recognition

*Below is a portion of the benchmark training set*

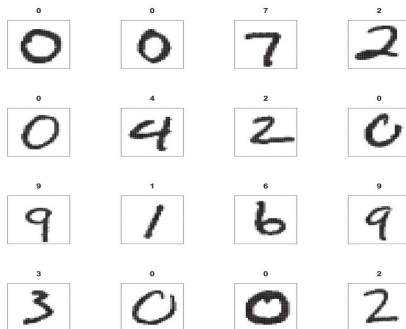


**Note:** *The challenge here is building classification techniques that accurately classify handwritten digits taken from the test set.*



# Handwritten Digit Recognition

*Below is a portion of the benchmark training set*



**Note:** *The challenge here is building classification techniques that accurately classify handwritten digits taken from the test set.*

## *kNN Predictive performance on Digit Recognition*

```
library(ElemStatLearn)

data(zip.train)
data(zip.test)

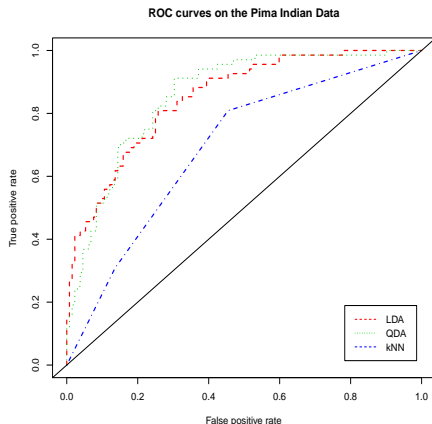
xtrain <- zip.train[,-1]
ytrain <- zip.train[, 1]
xtest  <- zip.test[,-1]
ytest  <- zip.test[, 1]

yhat.test <- knn(xtrain, xtest, ytrain, k=3)

pcc <- sum(diag(prop.table(table(ytest, yhat.test))))
cat('\n Accuracy is ', 100*pcc, '%\n')
```

# kNN vs others on Pima Indian Data

*Below is a comparative ROC curve on the Pima Indian Data*



*Clearly, kNN looks inferior on this data. But ... Can it be improved?  
Different distance? Better  $k$ ?*

# Existing Computing Tools

- *Do the following*




```
install.packages('ctv')  
library{ctv}  
install.views('MachineLearning')  
install.views('HighPerformanceComputing')  
install.views('TimeSeries')  
install.views('Bayesian')
```

- *R packages for big data*

```
library{biglm}  
library(foreach)  
library(glmnet)  
library(kernlab)  
library(randomForest)  
library(ada)  
library(audio)  
library(rpart)
```

# Some Remarks and Recommendations

- **Applications:** Sharpen your intuition and your commonsense by questioning things, reading about interesting open applied problems, and attempt to solve as many problems as possible
- **Methodology:** Read and learn about the fundamental of statistical estimation and inference, get acquainted with the most commonly used methods and techniques, and consistently ask yourself and others what the natural extensions of the techniques could be.
- **Computation:** Learn and master at least two programming languages. I strongly recommend getting acquainted with **R**  
<http://www.r-project.org>
- **Theory:** "Nothing is more practical than a good theory" (Vladimir N. Vapnik). When it comes to data mining and machine learning and predictive analytics, those who truly understand the inner workings of algorithms and methods always solve problems better.

-  James, G, Witten, D, Hastie, T and Tibshirani, R (2013). *An Introduction to Statistical Learning with Applications in R*. Springer, New York, (e-ISBN: 978-1-4614-7138-7),(2013)
-  Clarke, B, Fokoué, E and Zhang, H (2009). *Principles and Theory for Data Mining and Machine Learning*. Springer Verlag, New York, (ISBN: 978-0-387-98134-5), (2009)
-  J. J. Faraway(2002). *Practical Regression and ANOVA using R*. Lecture Notes contributed to the R project, (2002)