CMPSC 448: Machine Learning

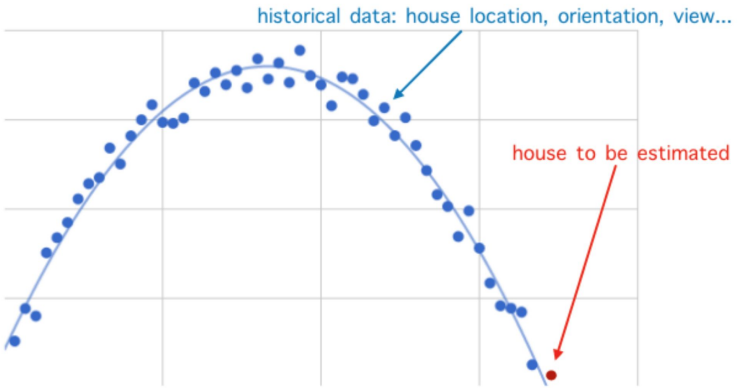
Lecture 8. Nearest Neighbor Classifiers

Rui Zhang

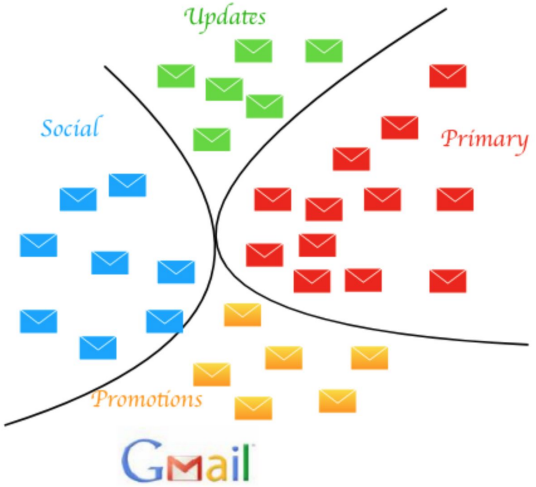
Fall 2021

Regression vs Classification

- Tracking a mortgage
- Houses data
- Banks: Estimate the loan based on location, orientation, view, etc
- Output values: **continuous**



- Spam classification
- Incoming emails
- How to group email in categories
- Output values: **discrete**, **categorical**



Regression vs Classification

- Ordinary Least Squares
- Ridge Regression
- Lasso Regression

- Nearest Neighbor
- Artificial Neural Networks: Perceptron and Deep Neural Networks
- Logistic Regression
- Decision Trees
- Support Vector Machines (SVMs)
- Bagging: Random Forests
- Boosting: AdaBoost and Gradient Boosted Trees

Outline

- The nearest neighbor (NN) search for classification and regression
- The importance of distance and distance metric learning problem
- The effect of number of neighbors
- Model selection for NN algorithm
- Curse of Dimensionality

Binary vs multi class classification

- Binary classification: the label set is binary

$$\mathcal{Y} = \{0, 1\} \text{ or } \mathcal{Y} = \{-1, +1\}$$

- Multi-class classification: the label set is has more than two labels

$$\mathcal{Y} = \{1, 2, 3, \ldots, K\}$$

Example: digit recognition (OCR)

Problem: classify images of handwritten digits by the actual digits they represent:

Input data is a set of black and white images labeled by the number in the image.

Each input is matrix of 0 and 1 values (0 for white and 1 for black)

Can be cast as an instance of multi-class classification problem with class labels:

$$\mathcal{Y} = \{0, 1, 2, \ldots, 9\}$$

Evaluation

Error rate of a classifier f on a set of labeled examples

$$\mathcal{S} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$$

$$\frac{\#\ \text{of}\ \ (\boldsymbol{x}, y) \in \mathcal{S}\ \ \text{such that}\ \ f(\boldsymbol{x}) \neq y}{n}$$

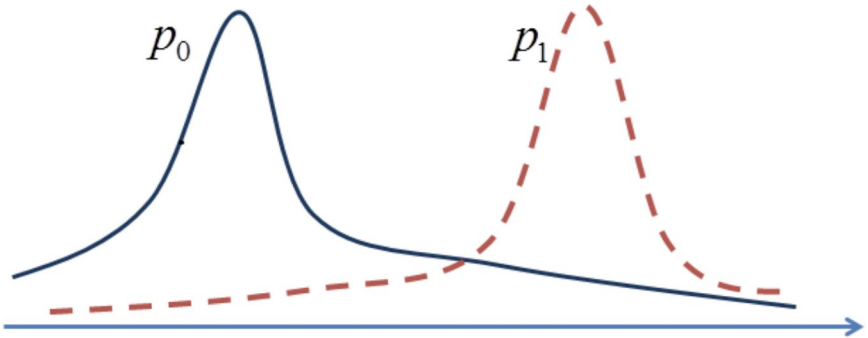
(i.e., the fraction of $\mathcal{S}$ on which f disagrees with paired label).

Bayes optimal classifier

Assume you know $\mathbb{P}(y \mid \boldsymbol{x})$. What would you predict?

Example:

$$\mathbb{P}(+1 \mid \boldsymbol{x}) = 0.8$$
$$\mathbb{P}(-1 \mid \boldsymbol{x}) = 0.2$$

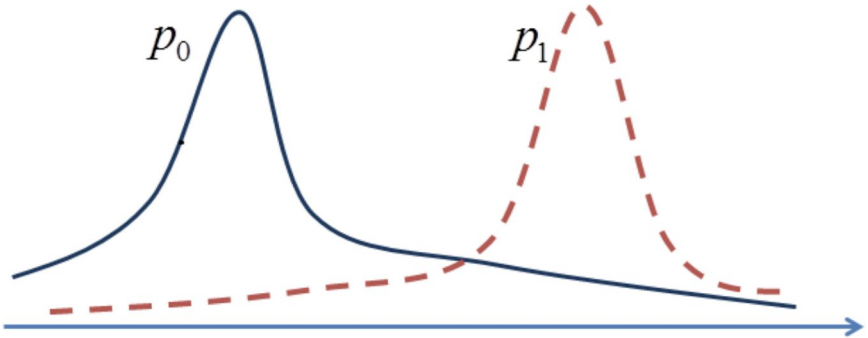


Bayes optimal classifier

Assume you know $\mathbb{P}(y \mid \boldsymbol{x})$. What would you predict?

Example:

$$\mathbb{P}(+1 \mid \boldsymbol{x}) = 0.8$$
$$\mathbb{P}(-1 \mid \boldsymbol{x}) = 0.2$$



Predict the most likely class:

$$y_* = \arg \max_{y \in \{+1,-1\}} \mathbb{P}(y \mid \boldsymbol{x})$$

The error (risk) of Bayes optimal classifier is called the Bayes risk:

$$1 - \mathbb{P}(y_* \mid \boldsymbol{x})$$

We can never do better than Bayes optimal classifier.

Nearest Neighbor (NN) classifier

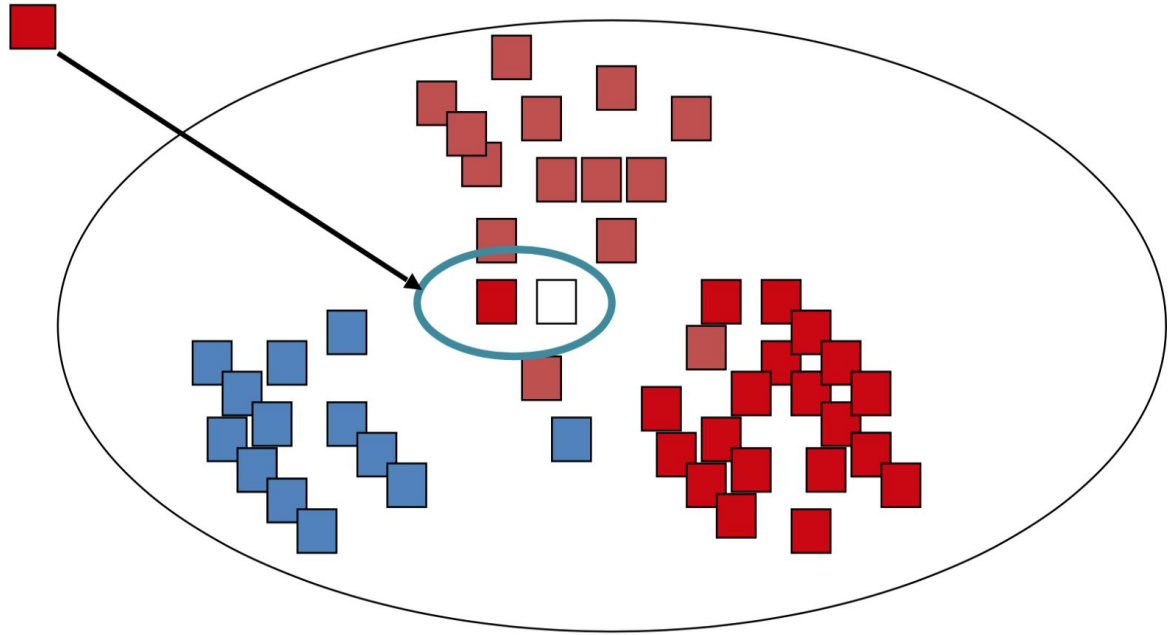
Given training data: $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$

Intuition:

Similar inputs have similar outputs.

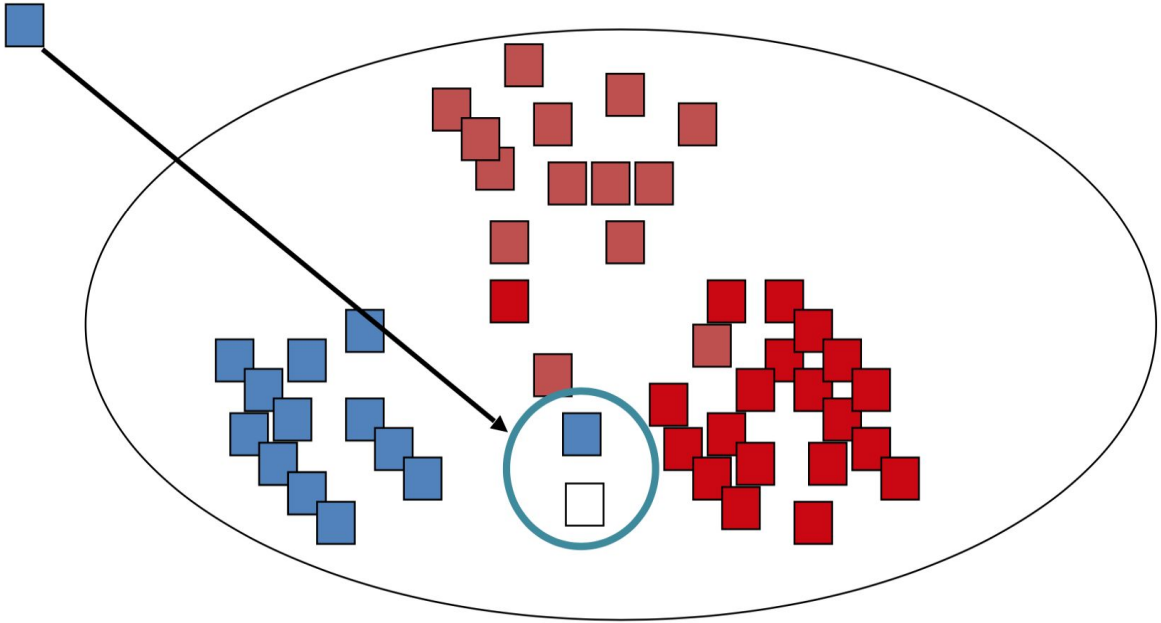
Nearest Neighbor (NN) classifier

$k = 1$



Nearest Neighbor (NN) classifier

$k = 1$



1-Nearest Neighbor (NN) classifier

Given training data: $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$

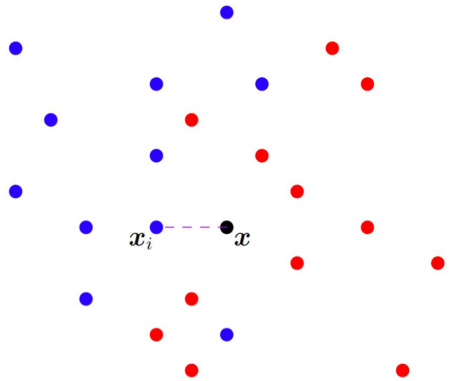
Do what your neighbor does

Predictor ($k = 1$):

On input (test) data $\boldsymbol{x} \in \mathbb{R}^d$

Find the closets point $\boldsymbol{x}_i$ in training dat to $\boldsymbol{x}$ (the **nearest neighbor**)

Return the label of nearest neighbor y_i



1-Nearest Neighbor (NN) classifier

Given training data: $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$

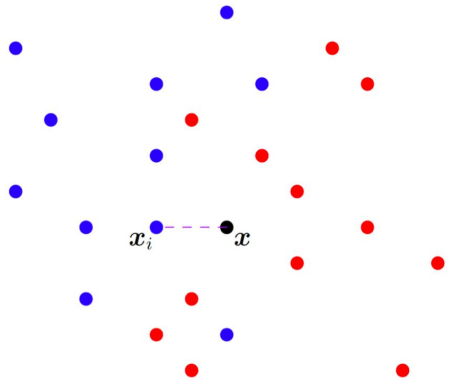
Do what your neighbor does

Predictor ($k = 1$):

On input (test) data $\boldsymbol{x} \in \mathbb{R}^d$

Find the closets point $\boldsymbol{x}_i$ in training dat to $\boldsymbol{x}$ (the **nearest neighbor**)

Return the label of nearest neighbor y_i



Question: how should we measure distance between points in training?

How to measure distance?

A default choice for distance between points in $\mathbb{R}^d$ is the Euclidean distance

$$||\boldsymbol{x} - \boldsymbol{y}||_2 = \sqrt{\sum_{i=1}^{d} (\boldsymbol{x}_i - \boldsymbol{y}_i)^2}$$

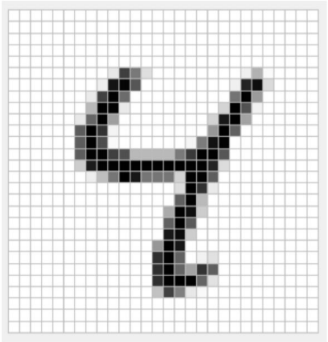
How to measure distance?

A default choice for distance between points in $\mathbb{R}^d$ is the Euclidean distance

$$||\boldsymbol{x} - \boldsymbol{y}||_2 = \sqrt{\sum_{i=1}^{d} (\boldsymbol{x}_i - \boldsymbol{y}_i)^2}$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]^\top$

and $\boldsymbol{x}' = [x'_1, x'_2, \ldots, x'_d]^\top$



Grayscale 28×28 pixel images.

Treat as *vectors* (of 784 real-valued *features*) that live in $\mathbb{R}^{784}$.

Distances

There are many other options (which could be much better than Euclidean distance)

- Minkowski distance:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{d} |\boldsymbol{x}_i - \boldsymbol{y}_i|^p \right)^{1/p}$$

p=1: Manhattan distance

p=2: Euclidean distance

Distances

There are many other options (which could be much better than Euclidean distance)

- Minkowski distance:

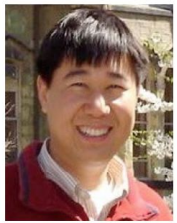
$$d(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{d} |\boldsymbol{x}_i - \boldsymbol{y}_i|^p \right)^{1/p}$$

  p=1: Manhattan distance

  p=2: Euclidean distance

- Edit distance (for strings): how add/delete/substitutions are required to transform one string to the other.
- Shape distance (for images): how much "warping" is required to change one to the other.
- Audio waveforms: dynamic time warping
- Mahalanobis distance (will talk about this in the following slides)

The importance of distance






$\boldsymbol{x}_1$ $\boldsymbol{x}_2$ $\boldsymbol{x}_3$ $\boldsymbol{x}_4$ $\boldsymbol{x}_5$ $\boldsymbol{x}_6$

Each image represented by a feature vector: $\boldsymbol{x} \in \mathbb{R}^d$

What Euclidean distance says about the similarity of images:

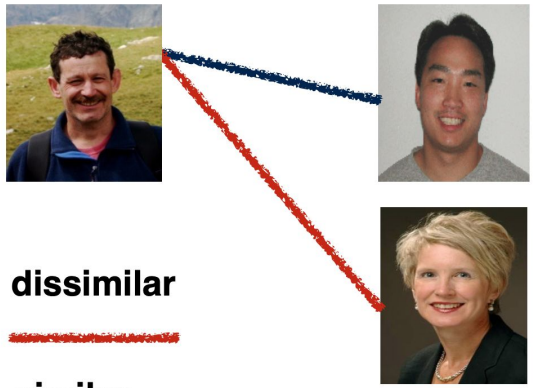
$$d(\boldsymbol{x}, \boldsymbol{y}) = ||\boldsymbol{x} - \boldsymbol{y}||_2 = \sqrt{\sum_{i=1}^{d} (\boldsymbol{x}_i - \boldsymbol{y}_i)^2}$$

The NN classifier fundamentally relies on a distance metric. The better that metric reflects label similarity, the better the classified will be.

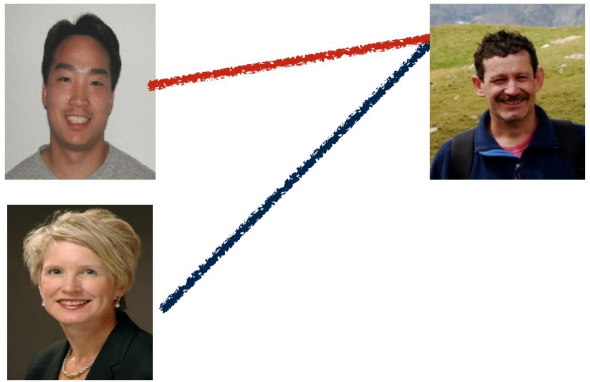
The importance of distance

Same feature vectors, but different applications

Male v.s. Female



Young v.s. Old



The meaning of distance depends on the application

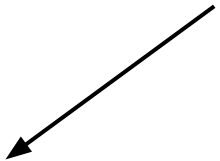
Mahalanobis distance

The Mahalanobis distance is the generalization of Euclidean distance defined by:

$$d_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^\top \boldsymbol{M} (\boldsymbol{x} - \boldsymbol{y})}$$

$\boldsymbol{M} \in \mathbb{R}^{d \times d}$

$$d_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{d} \sum_{j=1}^{d} (\boldsymbol{x}_i - \boldsymbol{y}_i) \boldsymbol{M}_{i,j} (\boldsymbol{x}_j - \boldsymbol{y}_j)}$$



A weight for each pair of features
Euclidean distance treats all features in a same way!

When $\boldsymbol{M}$ is the identity matrix, Mahalanobis distance is just Euclidean Distance.

Mahalanobis distance

The Mahalanobis distance is the generalization of Euclidean distance defined by:

$$d_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^\top \boldsymbol{M} (\boldsymbol{x} - \boldsymbol{y})}$$

$\boldsymbol{M} \in \mathbb{R}^{d \times d}$ is symmetric, positive definite matrix (positive eigenvalues). This guarantees:

$$d_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{y}) \geq 0$$

$$d_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{y}) = 0 \quad \text{if and only if} \quad \boldsymbol{x} = \boldsymbol{y}$$

$$d_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{y}) = d_{\boldsymbol{M}}(\boldsymbol{y}, \boldsymbol{x})$$

$$d_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{y}) + d_{\boldsymbol{M}}(\boldsymbol{y}, \boldsymbol{z}) \geq d_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{z})$$

Mahalanobis distance - Triangular Inequality

Since $\boldsymbol{M}$ is symmetric, so it has eigen decomposition $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top$. Since $\boldsymbol{M}$ is P.D., it has positive eigenvalues, then we can further decompose $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\Sigma}^{\frac{1}{2}}$. Therefore,

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{U}^\top = \boldsymbol{U}\boldsymbol{\Sigma}^{\frac{1}{2}}(\boldsymbol{U}\boldsymbol{\Sigma}^{\frac{1}{2}})^\top = \boldsymbol{V}^\top\boldsymbol{V}$$

where $\boldsymbol{V} := (\boldsymbol{U}\boldsymbol{\Sigma}^{\frac{1}{2}})^\top$ So,

$$d_{\boldsymbol{M}}(\boldsymbol{x},\boldsymbol{y}) = \sqrt{(\boldsymbol{x}-\boldsymbol{y})^\top\boldsymbol{M}(\boldsymbol{x}-\boldsymbol{y})} = \sqrt{(\boldsymbol{x}-\boldsymbol{y})^\top\boldsymbol{V}^\top\boldsymbol{V}(\boldsymbol{x}-\boldsymbol{y})} = \sqrt{(\boldsymbol{V}(\boldsymbol{x}-\boldsymbol{y}))^\top\boldsymbol{V}(\boldsymbol{x}-\boldsymbol{y})} = ||\boldsymbol{V}(\boldsymbol{x}-\boldsymbol{y})||_2$$

This means the Mahalanobis distance with $\boldsymbol{M}$ on $(\boldsymbol{x},\boldsymbol{y})$ is the same as the Euclidean distance on $(\boldsymbol{V}\boldsymbol{x},\boldsymbol{V}\boldsymbol{y})$. Then, we can apply the triangular inequality for Euclidean distance:

$$||\boldsymbol{a}||_2 + ||\boldsymbol{b}||_2 \geq ||\boldsymbol{a}+\boldsymbol{b}||_2$$

So,

$$\begin{aligned} d_{\boldsymbol{M}}(\boldsymbol{x},\boldsymbol{y}) + d_{\boldsymbol{M}}(\boldsymbol{y},\boldsymbol{z}) &= ||\boldsymbol{V}(\boldsymbol{x}-\boldsymbol{y})||_2 + ||\boldsymbol{V}(\boldsymbol{y}-\boldsymbol{z})||_2 \\ &\geq ||\boldsymbol{V}(\boldsymbol{x}-\boldsymbol{y}) + \boldsymbol{V}(\boldsymbol{y}-\boldsymbol{z})||_2 \\ &= ||\boldsymbol{V}(\boldsymbol{x}-\boldsymbol{z})||_2 \\ &= d_{\boldsymbol{M}}(\boldsymbol{x},\boldsymbol{z}) \end{aligned}$$

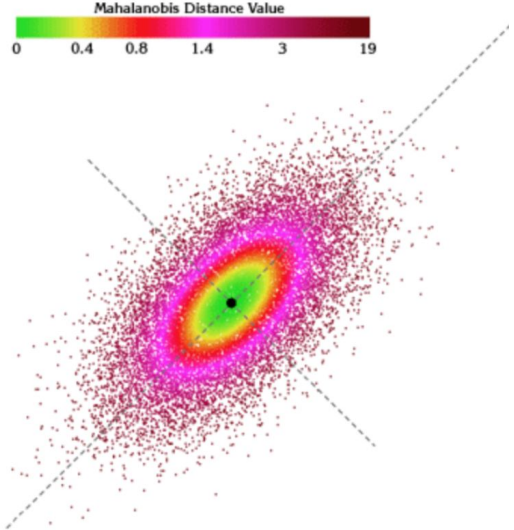
So the triangular inequality holds.

Mahalanobis distance

The Mahalanobis distance is the generalization of Euclidean distance defined by:

$$d_M(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^\top \boldsymbol{M} (\boldsymbol{x} - \boldsymbol{y})}$$

$\boldsymbol{M} \in \mathbb{R}^{d \times d}$ is symmetric, positive definite matrix (positive eigenvalues).



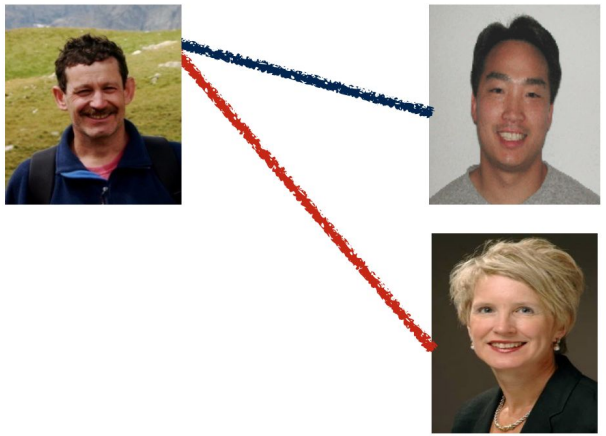
$$d_M(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{d} \sum_{j=1}^{d} (\boldsymbol{x}_i - \boldsymbol{y}_i) \boldsymbol{M}_{i,j} (\boldsymbol{x}_j - \boldsymbol{y}_j)}$$

A weight for each pair of features
Euclidean distance treats all features in a same way!

Distance metric learning

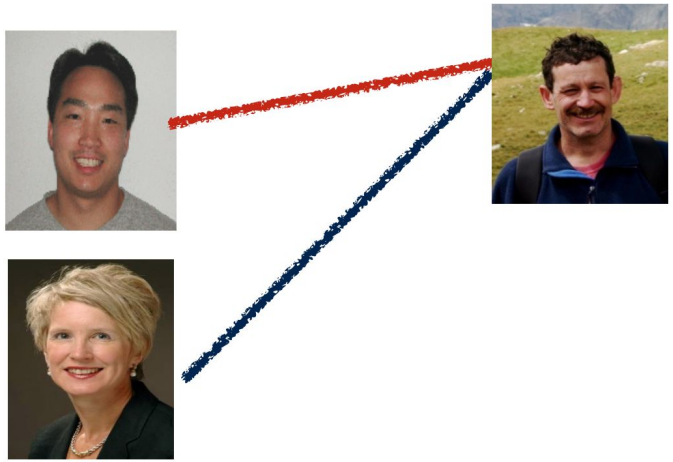
Given pairs of similar and dissimilar examples, can we learn the positive semi-definite matrix in Mahalanobis distance?

Male v.s. Female



$$\boldsymbol{M} \in \mathbb{R}^{d \times d}$$

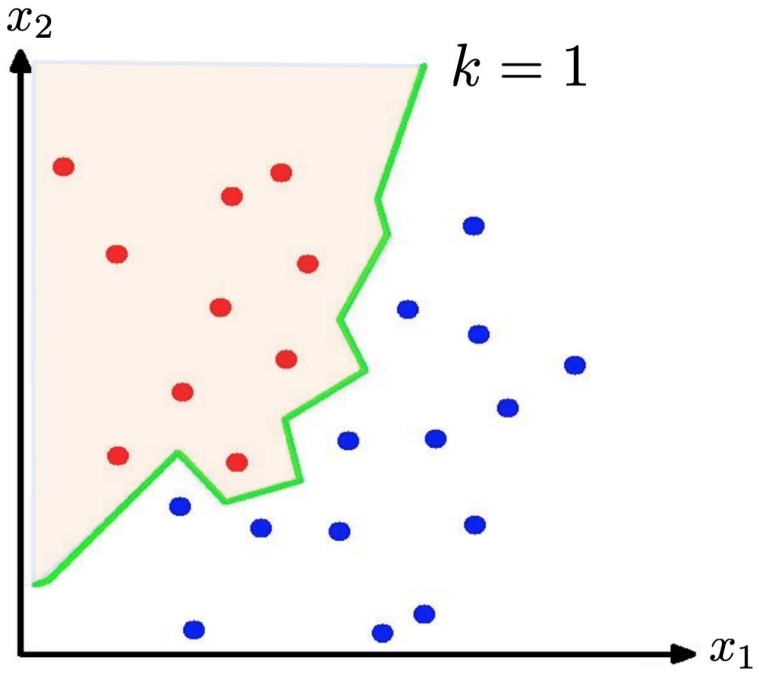
Young v.s. Old



$$\boldsymbol{M}' \in \mathbb{R}^{d \times d}$$

Decision boundary

Let’s assume you have two features with binary labels (blue and red)
We can draw the decision boundary of NN on test data points:



Error Analysis for k = 1

Some mistakes made by the NN classifier

Error Analysis for k = 1

Some mistakes made by the NN classifier

First mistake (correct label is “2”) could have been avoided by looking at the three nearest neighbors (whose labels are “8”, “2”, and “2”).

test point three nearest neighbors

Let’s upgrade the NN classifier

k-Nearest Neighbor (NN) classifier

Do what your neighbors do

Predictor ($k \geq 2$):

On input (test) data $\boldsymbol{x} \in \mathbb{R}^d$

Find k closets points in **training** data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ to $\boldsymbol{x}$

Return the plurality of labels of nearest neighbors $y_1, y_2, \ldots, y_k$

(Break ties in both steps arbitrarily)

k-Nearest Neighbor (NN) regression

Do what your neighbors do

Predictor ($k \geq 2$):

On input (test) data $\boldsymbol{x} \in \mathbb{R}^d$

Find k closets points in **training** data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ to $\boldsymbol{x}$

Return the plurality of labels of nearest neighbors $y_1, y_2, \ldots, y_k$

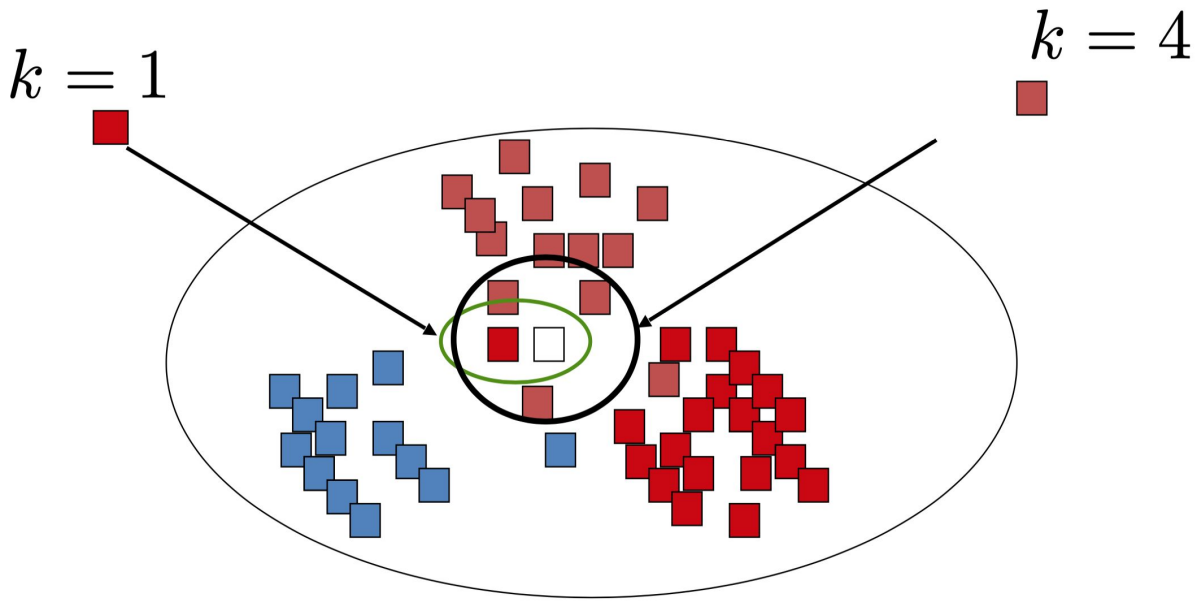
(Break ties in both steps arbitrarily)

Question: How can we change this if we want to do regression instead of classification?

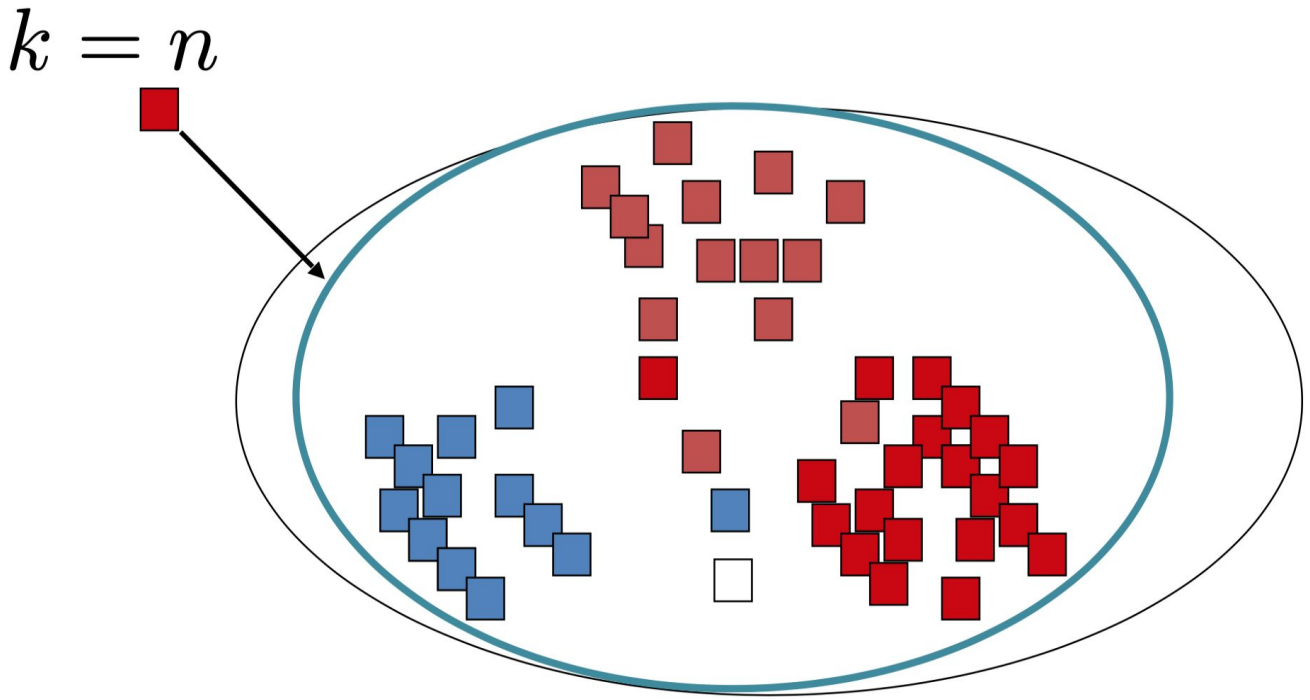
We can calculate the average of the numerical target of the K nearest neighbors!

Number of neighbors

How many neighbors should we count?



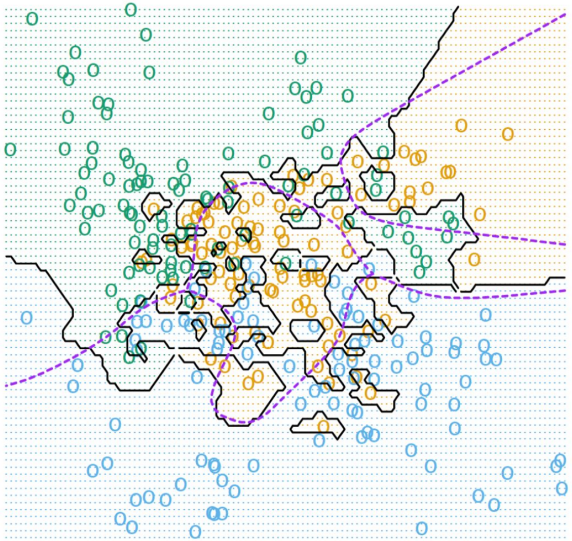
Nearest Neighbor (NN) classifier



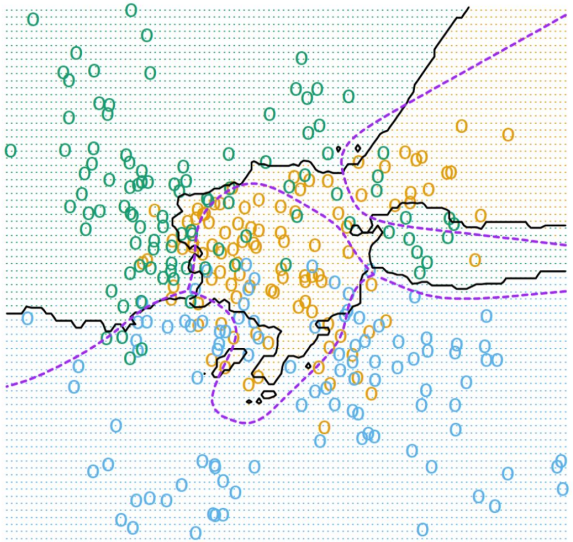
The effect of k

Smaller k : smaller training error rate (k = 1 has zero training error).

Larger k : higher training error rate, but predictions are more “stable” due to voting.



1-NN



15-NN

Purple dotted lines: Bayes optimal classifier decision boundaries.
Black solid lines: k-NN decision boundaries.

Choosing k

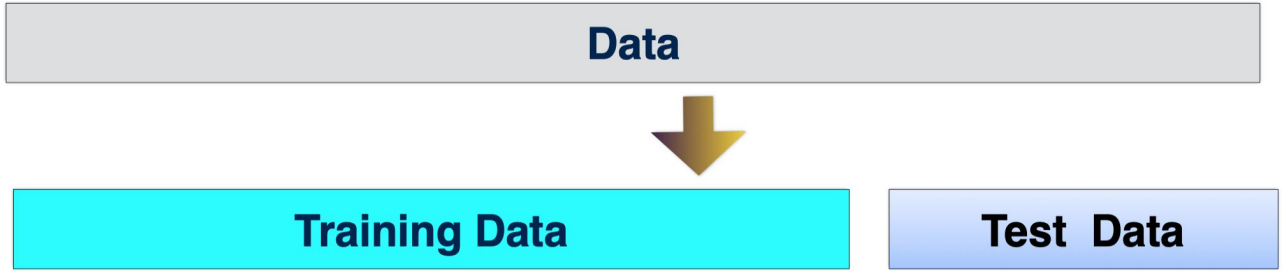
Number of neighbors k is the only hyper-parameter of the model!
Divide training examples into two sets

- A training set (80%) and a validation set (20%)

Predict the class labels for validation (a.k.a. holdout) set by using the examples in training set

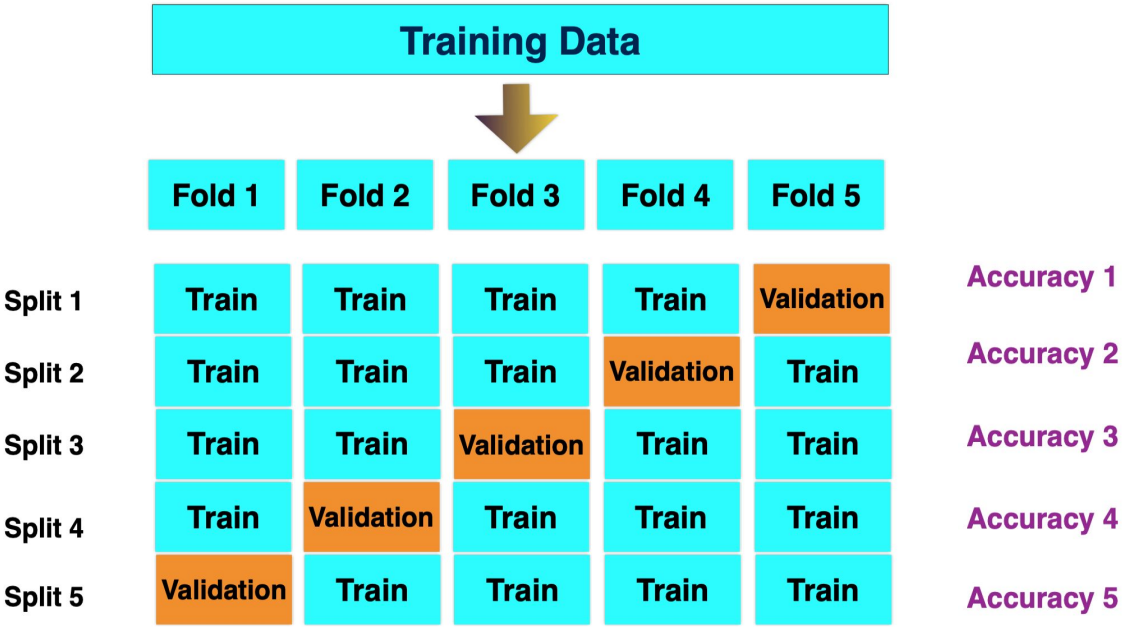
Choose the number of neighbors k that maximizes the classification accuracy (smallest hold-out error rate)

Training and testing




```
from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier


X_train, X_test, y_train, y_test = train_test_split(X, y)

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)

print("accuracy: {:.2f}".format(knn.score(X_test, y_test)))

y_pred = knn.predict(X_test)
```

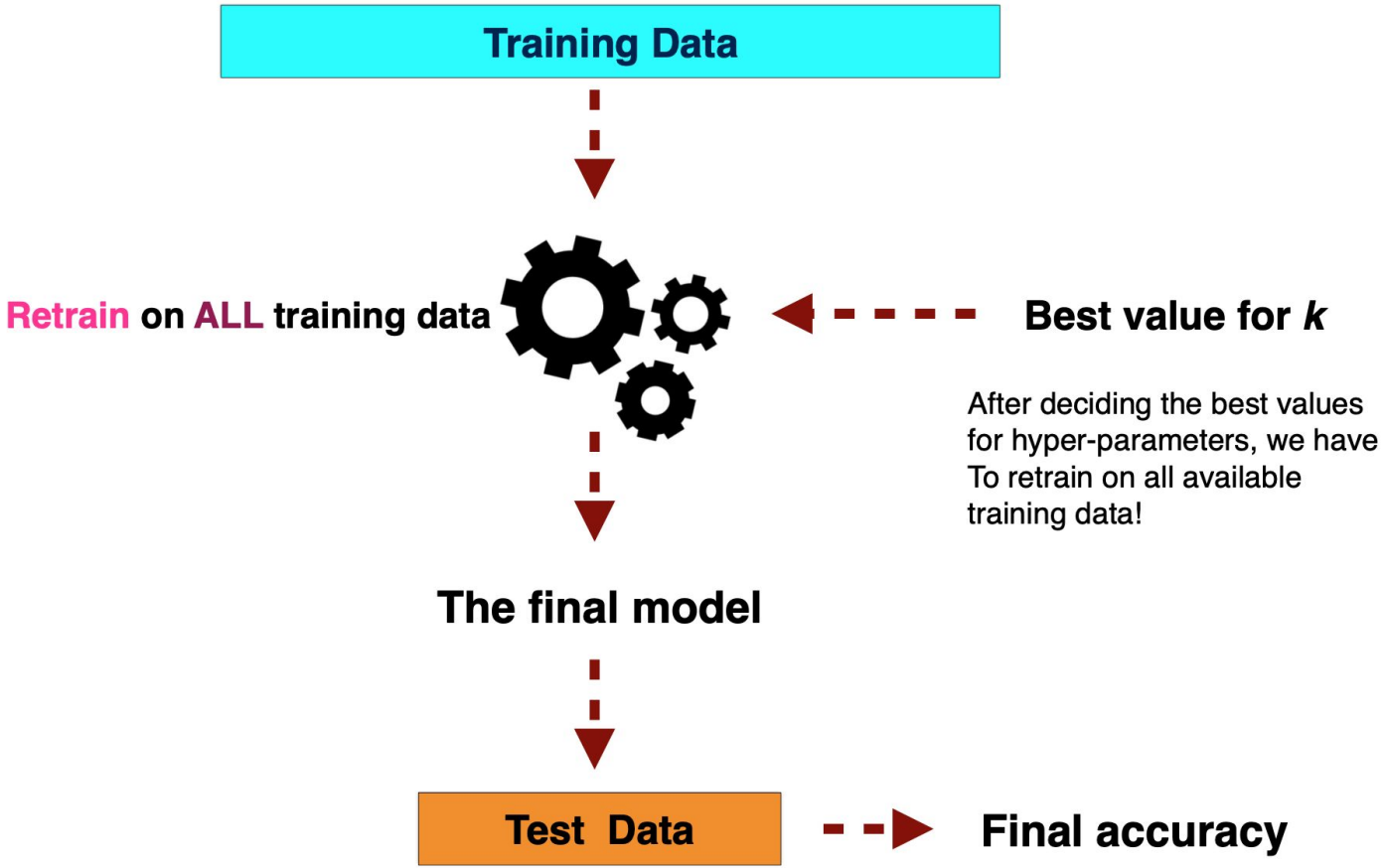
Cross validation



Example: 5-fold cross validation where we split the training data into 5 folds and for every split, we train on 4 folds and evaluate on the 5th fold.

Total Accuracy = average accuracy over all splits!

Final model


Training Data
Retrain on ALL training data
Best value for k
After deciding the best values for hyper-parameters, we have To retrain on all available training data!
The final model
Test Data
Final accuracy

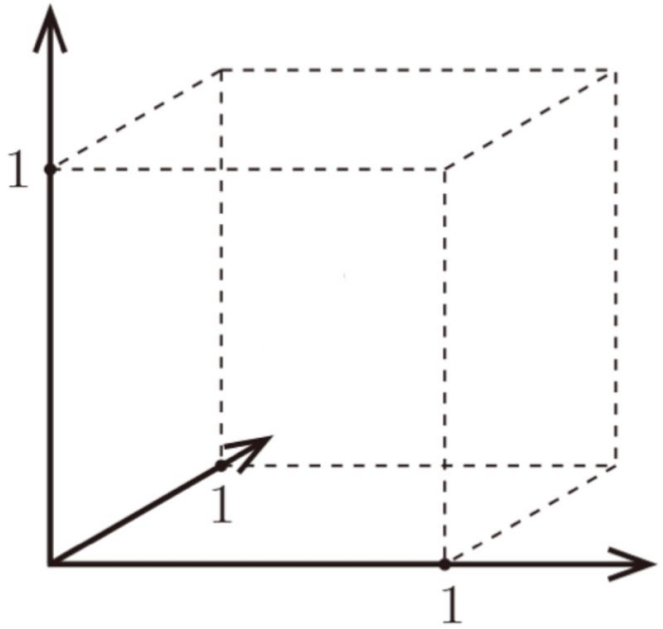
Cross validation in sklearn

```
from sklearn.model_selection import cross_val_score
X_train, X_test, y_train, y_test = train_test_split(X, y)
cross_val_scores = []
for i in neighbors:
    knn = KNeighborsClassifier(n_neighbors=i)
    scores = cross_val_score(knn, X_train, y_train, cv=5)
    cross_val_scores.append(np.mean(scores))

best_n_neighbors = neighbors[np.argmax(cross_val_scores)]

knn = KNeighborsClassifier(n_neighbors=best_n_neighbors)

knn.fit(X_train, y_train)

print("Test score: {:.3f}".format(knn.score(X_test, y_test)))
```

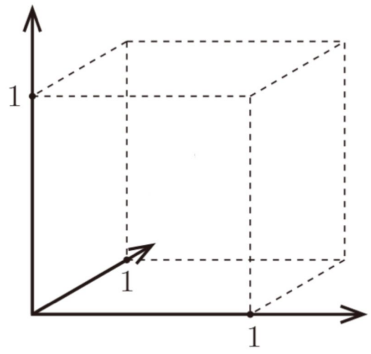
Curse of dimensionality

Unfortunately, when the dimension d increases, the notion of “nearest points” vanishes.

Let’s plot the histograms of the distribution of the pairwise-distances for 100 random data points in cube



Curse of dimensionality



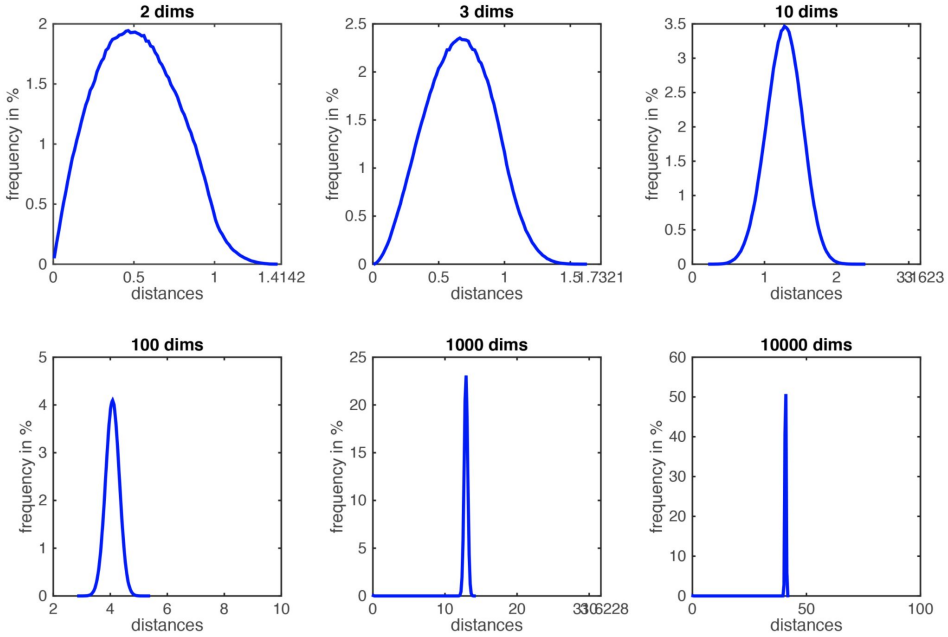
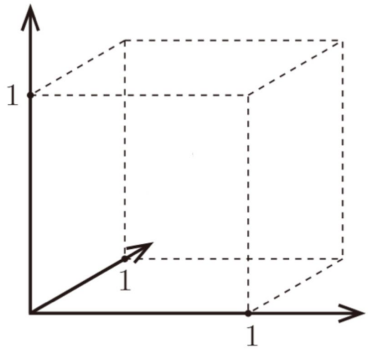


Figure demonstrating "the curse of dimensionality". The histogram plots show the distributions of all pairwise distances between randomly distributed points within d-dimensional unit squares. As the number of dimensions d grows, all distances concentrate within a very small range.

https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html

Curse of dimensionality



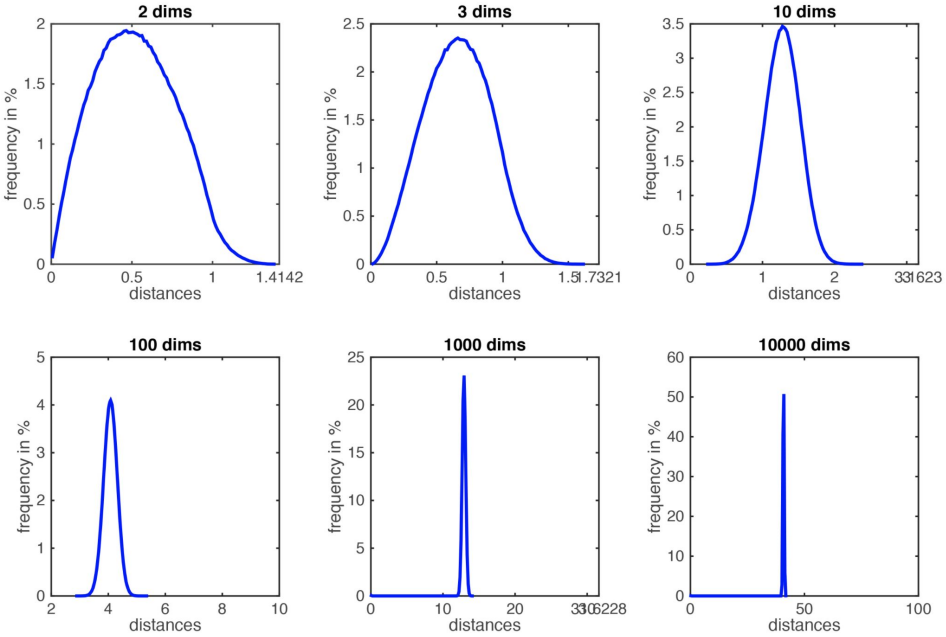


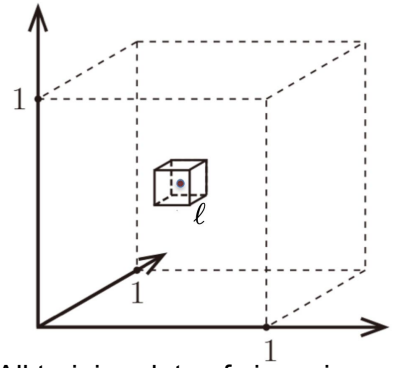
Figure demonstrating "the curse of dimensionality". The histogram plots show the distributions of all pairwise distances between randomly distributed points within d-dimensional unit squares. As the number of dimensions d grows, all distances concentrate within a very small range.

When dimension is 1000, all the points are at a similar distance one from the others, so the notion of “nearest-points” vanishes.

If the number n of observations remains fixed while the dimension d of the observations increases, the observations get rapidly very isolated and local methods such as NN can NOT work.

https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html

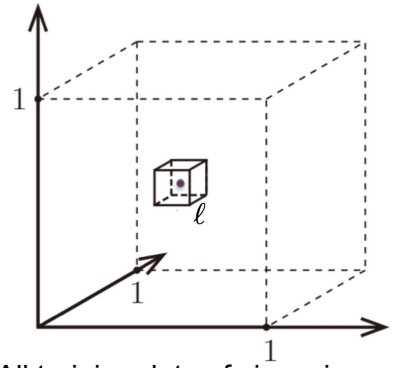
How many observations do we need?



All training data of size n is sampled uniformly within this cube.

ℓ the edge length of the smallest hyper-cube that contains all k-nearest neighbor of a test point.

How many observations do we need?



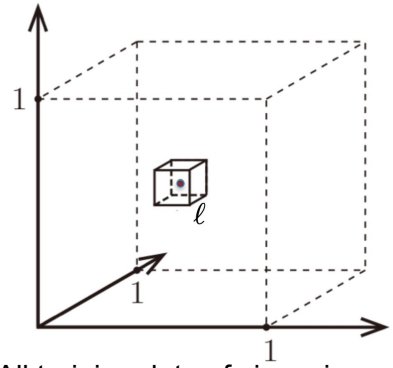
All training data of size n is sampled uniformly within this cube.

ℓ the edge length of the smallest hyper-cube that contains all k-nearest neighbor of a test point.

$$\ell^d \approx \frac{k}{n}$$

$$\ell \approx \left(\frac{k}{n}\right)^{1/d}$$

How many observations do we need?



All training data of size n is sampled uniformly within this cube.

$n = 1000 \qquad k = 10$

ℓ the edge length of the smallest hyper-cube that contains all k-nearest neighbor of a test point.

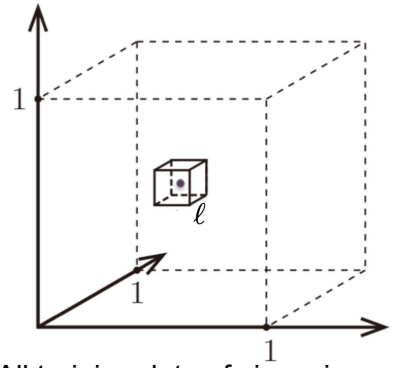
$$\ell^d \approx \frac{k}{n}$$

$$\ell \approx \left(\frac{k}{n}\right)^{1/d}$$

d	ℓ
2	0.1
10	0.63
100	0.955
1000	0.9954

Almost the entire space is needed to find the 10-NN for d = 1000!

How many observations do we need?



All training data of size n is sampled uniformly within this cube.

$n = 1000 \qquad k = 10$

ℓ the edge length of the smallest hyper-cube that contains all k-nearest neighbor of a test point.

$$\ell^d \approx \frac{k}{n}$$

$$\ell \approx \left(\frac{k}{n}\right)^{1/d}$$

d	ℓ
2	0.1
10	0.63
100	0.955
1000	0.9954

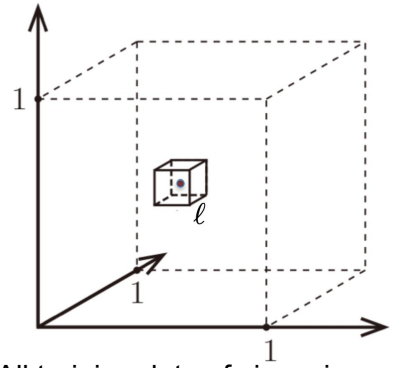
Almost the entire space is needed to find the 10-NN for d = 1000!

One might think that one rescue could be to increase the number of training samples, n, until the nearest neighbors are truly close to the test point.

$$\ell = \frac{1}{10} = 0.1 \Rightarrow n = \frac{k}{\ell^d} = k \cdot 10^d$$

The number of points n should grow more than exponentially fast with d.

How many observations do we need?



All training data of size n is sampled uniformly within this cube.

$n = 1000 \qquad k = 10$

ℓ the edge length of the smallest hyper-cube that contains all k-nearest neighbor of a test point.

$$\ell^d \approx \frac{k}{n}$$

$$\ell \approx \left(\frac{k}{n}\right)^{1/d}$$

d	ℓ
2	0.1
10	0.63
100	0.955
1000	0.9954

Almost the entire space is needed to find the 10-NN for d = 1000!

One might think that one rescue could be to increase the number of training samples, n, until the nearest neighbors are truly close to the test point.

$$\ell = \frac{1}{10} = 0.1 \Rightarrow n = \frac{k}{\ell^d} = k \cdot 10^d$$

The number of points n should grow more than exponentially fast with d.

This is called curse of dimensionality! The moral of this example is that we have to be very careful with our geometric intuitions in high-dimensional spaces. These spaces have some counterintuitive geometric properties