Section 10 k-Nearest Neighborhoods

1. k-NN regression
2. k-NN classification
3. The curse of dimensionality

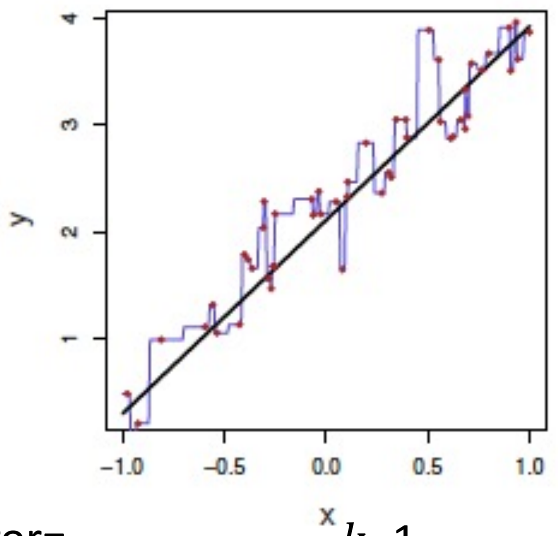
➢ k-Nearest Neighbors regression.

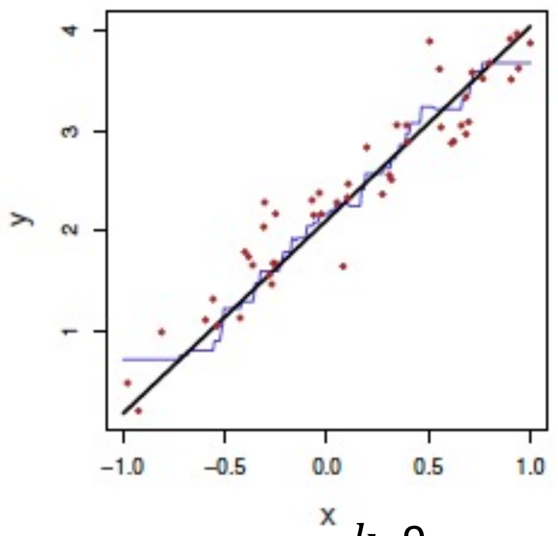
k-NN regression is one of the simplest and best-known non-parametric methods.

Assumption: Similar Inputs have similar outputs.

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 $h(x_0)$ using the average of all the training responses in $\mathcal{N}_0$

$$\tilde{h}(x_0) = \frac{1}{k} \sum_{x_i \in \mathcal{N}_0} y_i$$

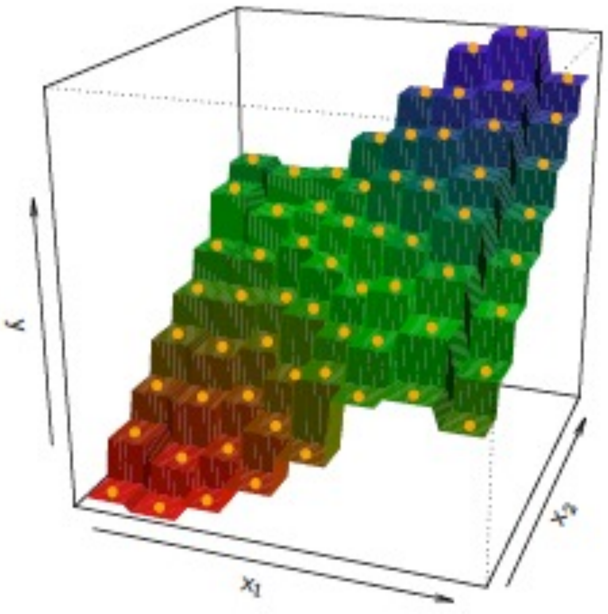




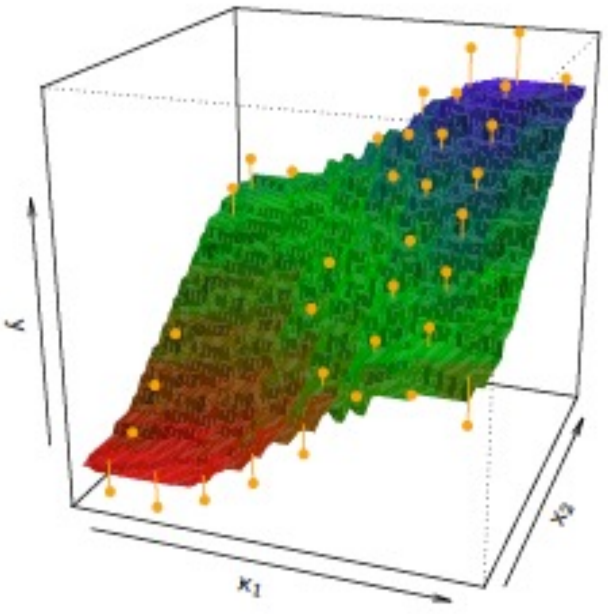
Training error=

k=1

k=9



k=1



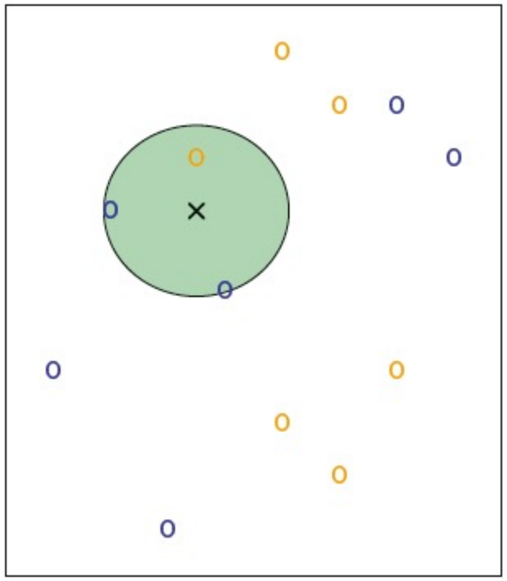
k=9

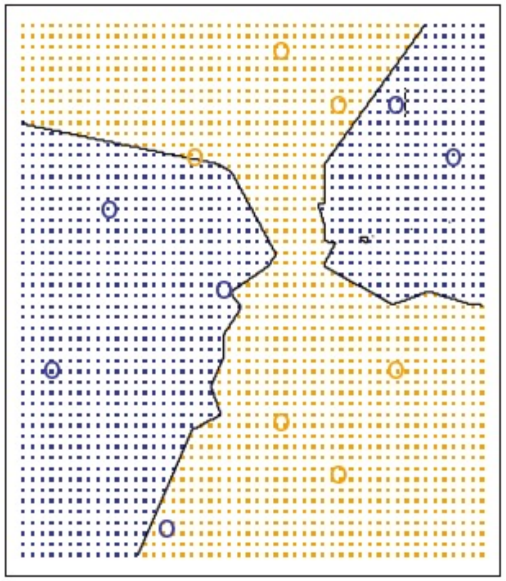
- When $k = 1$, the kNN fit perfectly interpolates the training observations, and consequently takes the form of a step function.
- When $k = 9$, the kNN fit still is a step function, but averaging over nine observations results in much smaller regions of constant prediction, and consequently a smoother fit.
- In general, the optimal value for k will depend on the bias-variance tradeoff.

➢ *k*-NN Classification.

Data: $D = \{(\vec{x}^{(i)}, y^{(i)})\}, i = 1 \dots n$

Assumption:
Similar Inputs have *similar* labels.
Classification rule:
For a test input $\vec{\boldsymbol{x}}$, assign the most common label amongst its k most similar training inputs.





- Let S_x be the subset of the data set D such that

  1. $|S_x| = k$
  2. For any $(\vec{x}', y') \in D \backslash \mathrm{S}_x$

  $$\mathrm{dist}(\vec{x}, \vec{x}') \geq \max_{(\vec{x}'', y'') \in S_x} (\mathrm{dist}(\vec{x}, \vec{x}''))$$

- The k-NN classifier $h(\vec{x})$ is defined as

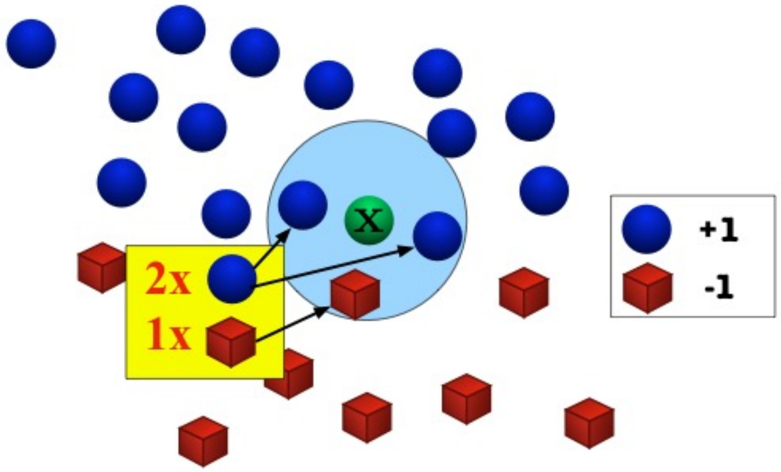
$$h(\vec{x}) = \mathrm{mode}\ (\{y'' \mid (\vec{x}'', y'') \in S_x\})$$

where mode(·) means to select the label of the highest occurrence.

- Under the assumption, k-NN classifier maximizes the conditional probability

$$P(y = j \mid \vec{x}\,) \approx \frac{1}{\mathrm{k}} \sum_{(\vec{x}'', y'') \in S_x} \mathbb{I}(y'' = j)$$

where $\mathbb{I}$ is the indicator function.



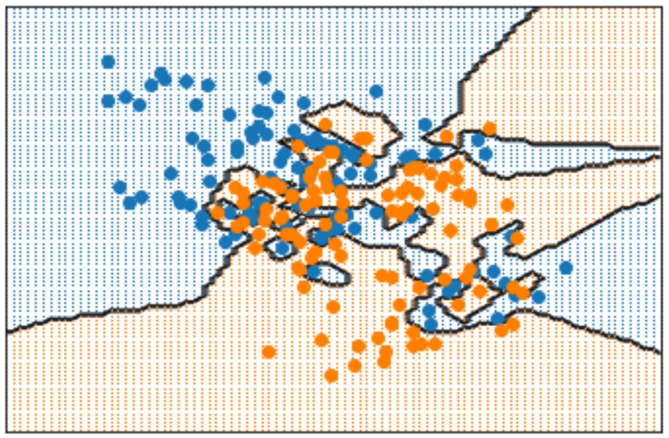
- The **zero-one loss function:**

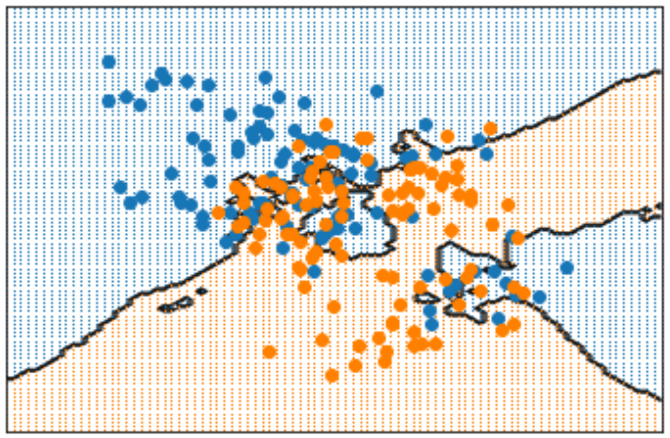
$$J_{0/1}(h) = \frac{1}{n} \sum_{i=0}^{n} \mathbb{I}(h(\vec{x}^{(i)}) \neq y^{(i)})$$

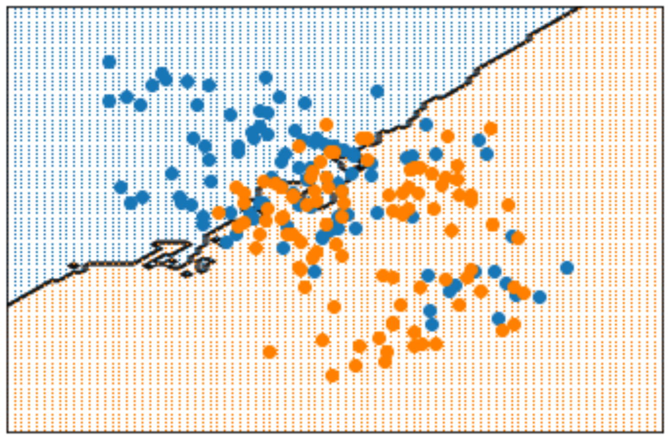
This loss function returns the *error rate* on this data set.

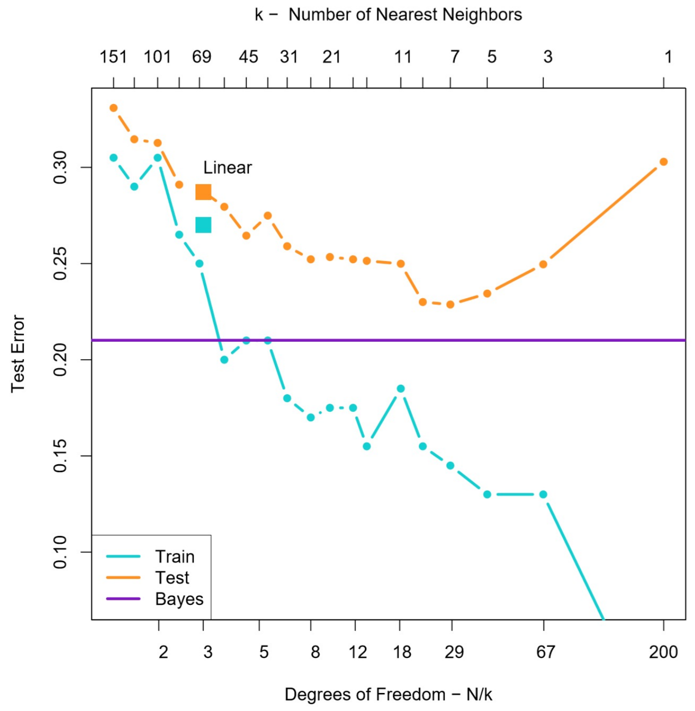
- The k-NN classifier $h(\vec{x}) \approx \underset{y}{\operatorname{argmax}}\, P(y \mid \vec{x})$, which is the Bayes optimal classifier.

The Bayes optimal error is $\epsilon_B = 1 - P(y^* \mid \vec{x})$








k − Number of Nearest Neighbors
151 101 69 45 31 21 11 7 5 3 1
Test Error
0.10 0.15 0.20 0.25 0.30
Linear
Train
Test
Bayes
2 3 5 8 12 18 29 67 200
Degrees of Freedom − N/k

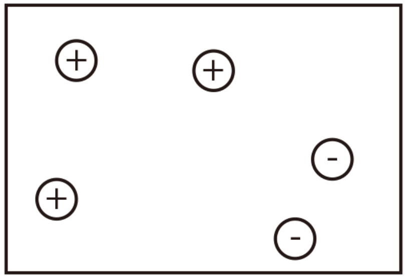
Theorem: (*Cover and Hart 1967*)
As $n \to \infty$, the 1-NN error is no more than twice the error of the Bayes Optimal classifier. (Similar guarantees hold for k>1)

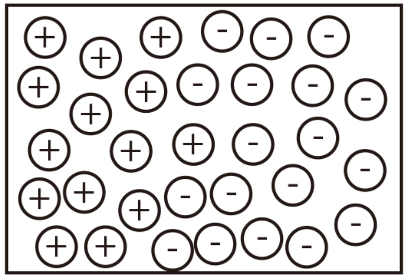
Proof:

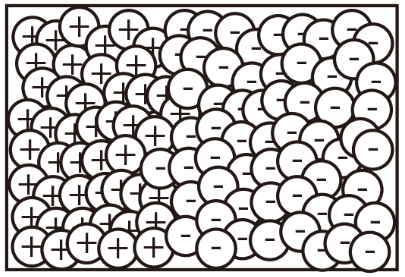
Let $\vec{x}_{NN}$ be the nearest neighbor of our test point $\vec{x}_t$.

As $n \to \infty$, $dist(\vec{x}_{NN}, \vec{x}_t) \to 0$.

$$\begin{aligned}
\epsilon_{NN} &= P(y^*|\vec{x}_t)\big(1 - P(y^*|\vec{x}_{NN})\big) + P(y^*|\vec{x}_{NN})\big(1 - P(y^*|\vec{x}_t)\big) \\
&\le \big(1 - P(y^*|\vec{x}_{NN})\big) + \big(1 - P(y^*|\vec{x}_t)\big) \\
&= 2\big(1 - P(y^*|\vec{x}_t)\big) \\
&= 2\epsilon_B
\end{aligned}$$







➢ **Distance function (Metric space $\mathbb{R}^n$)**

The k-nearest neighbor classifier fundamentally relies on a distance metric.

The better that metric reflects label similarity, the better the classified will be.

The most common choice is the **Minkowski distance.**

$$\operatorname{dist}(\vec{x}, \vec{z}) = \left(\sum_{i=1}^{d} |x_i - z_i|^p \right)^{1/p}$$

K-NN method is an example of non-parametric method.

Non-parametric methods do not make explicit assumptions about the functional form of $h(\vec{x})$. Instead, they seek an estimate of $h(\vec{x})$ that gets as close to the data points as possible without being too rough or wiggly.

Advantage over parametric approaches: by avoiding the assumption of a particular functional form for $h(\vec{x})$, they have the potential to accurately fit a wider range of possible shapes for $h(\vec{x})$.

Disadvantage of non-parametric approaches: since they do not reduce the problem of estimating $h(\vec{x})$ to a small number of parameters, a very large number of observations (far more than is typically needed for a parametric approach) is required in order to obtain an accurate estimate for $h(\vec{x})$.

For k-NN method,

1. It may be computationally hard to find the k-nearest neighbors for a large amount of data of high dimension. It must store and search through the entire training set in order to classify a single test point.
2. Geometry in higher dimensions often behaves counter-intuitively.

➢ Bad News: Curse of dimensionality

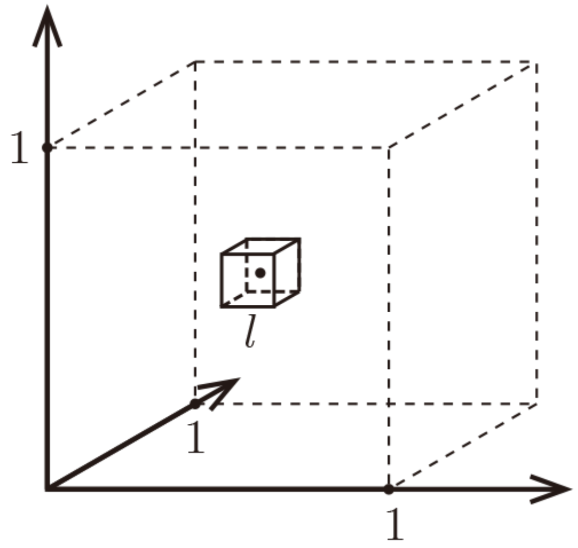
The k nearest neighbors classifier **assume** that similar points share similar labels.

However, in high dimensional spaces, points that are drawn from a probability distribution, tend to never be close together.

Distances between points

Draw n points **uniformly** at random within the unit cube $D = [0,1]^d$.

Investigate how much space the k nearest neighbors of a test point inside this cube.



Considering the k nearest neighbors of such a test point.

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

Then $l^d \approx \frac{k}{n}$ and $l = \left(\frac{k}{n}\right)^{1/d}$

Suppose n=1000 and k=10,

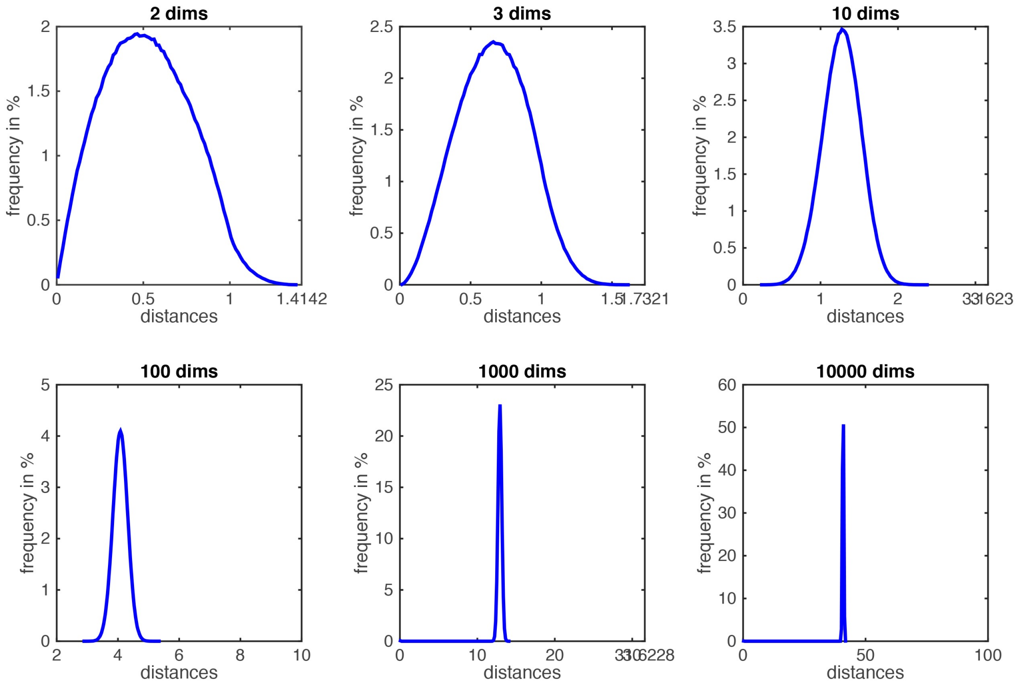
d	l
2	0.1
10	0.63
100	0.955
1000	0.9954

So as d large enough, almost the entire space is needed to find the 10-NN.

This breaks down the k-NN assumptions.

Distance between two randomly drawn data points $\vec{x}$ and $\vec{y}$ increases drastically with their dimensionality.

$$dist(\vec{x}, \vec{y}) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_d - y_d)^2}$$



The histogram plots show the distributions of all pairwise distances between randomly distributed points within d-dimensional unit squares.

As the dimensions d grows, all distances concentrate within a very small range.

Distances to hyperplanes

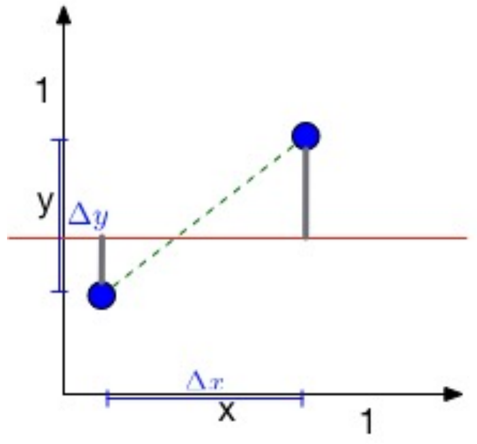
Two random data points in $D = [0,1]^d$, the distance between datapoints is

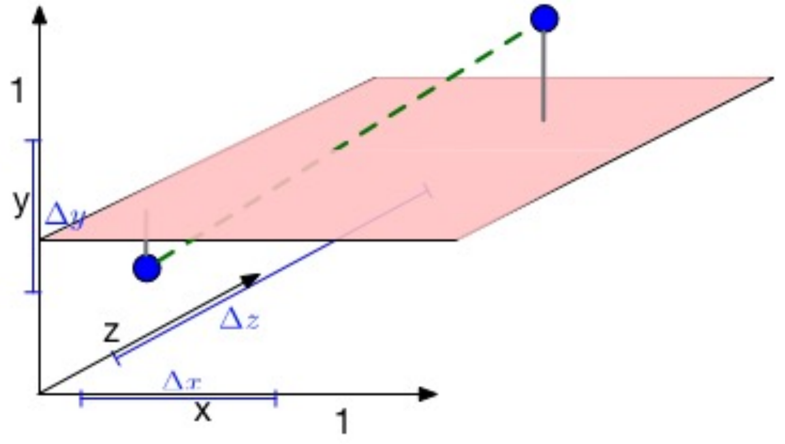
$$\mathrm{dist}(\vec{x}, \vec{y}) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_d - y_d)^2}$$

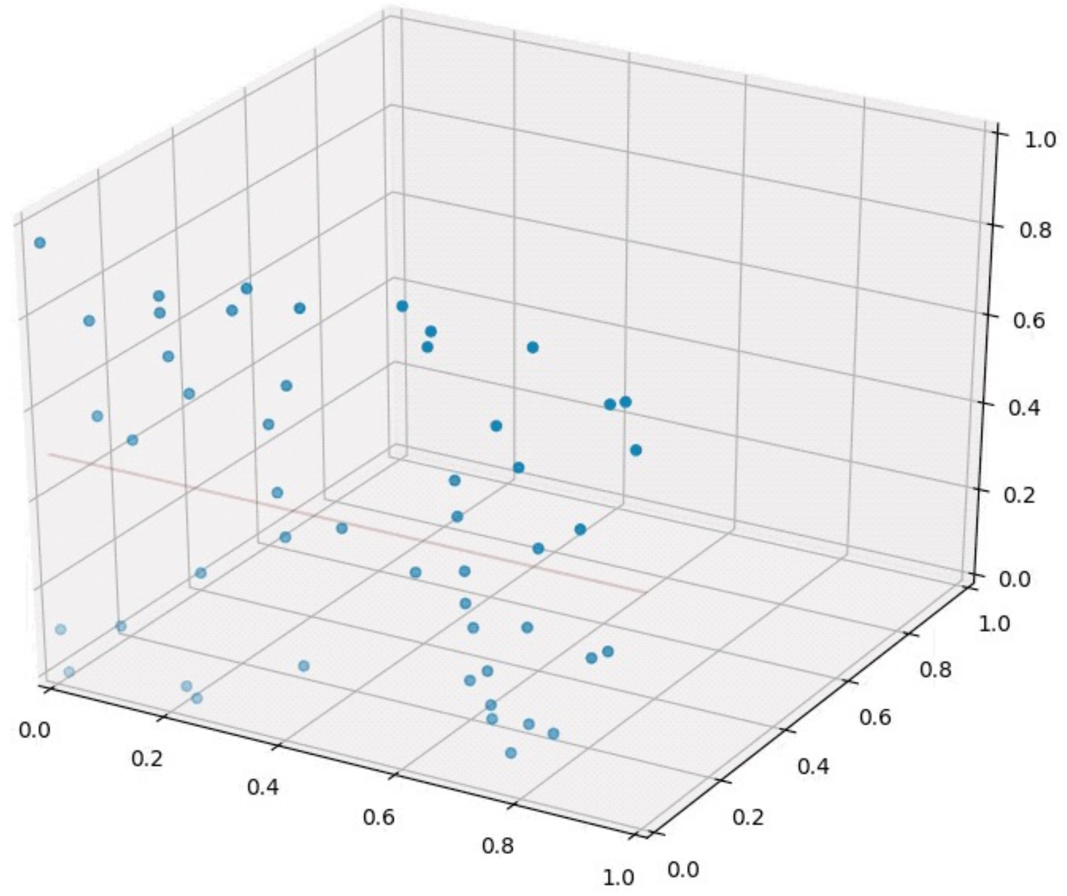
Distance between data points $\vec{x}$ and $\vec{y}$ increases from $[0,1]^2$ to $[0,1]^3$

The distance to the red hyperplane remains unchanged as a third dimension is added.

The reason is that the normal of the hyperplane is orthogonal to the new dimension.







We use hyperplanes between concentrations of different classes for classification.

As distances between pairwise points become very large in high dimensional spaces, distances to hyperplanes become relatively tiny.

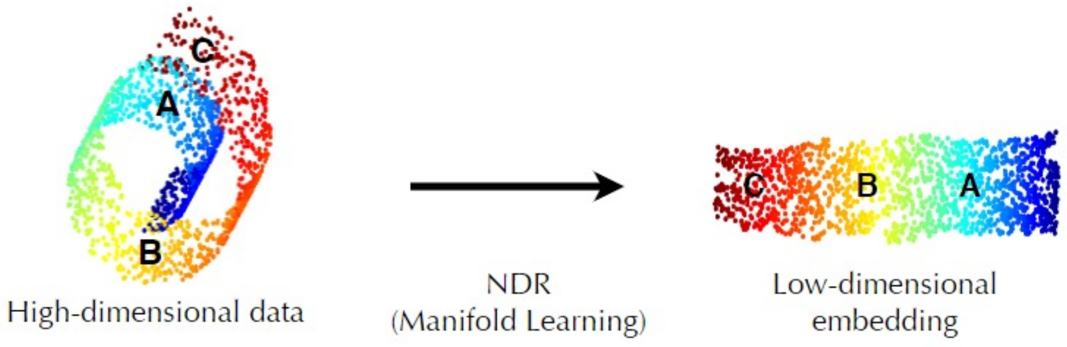
- **Good News: Data with low dimensional structure.**

Data may lie in low dimensional subspace or on sub-manifolds.

The true dimensionality of the data can be much lower than its ambient space.

For example, human faces are a typical example of an intrinsically low dimensional data set. Although an image of a face may require 18 million pixels, a person may be able to describe this person with less than 50 attributes (e.g. male/female, blond/dark hair, ...) along which faces vary.

Methods:
1. Principal component analysis. 2. Manifold Learning. 3. Topological data analysis



Use geodesic rather than Euclidean distances in manifold learning.