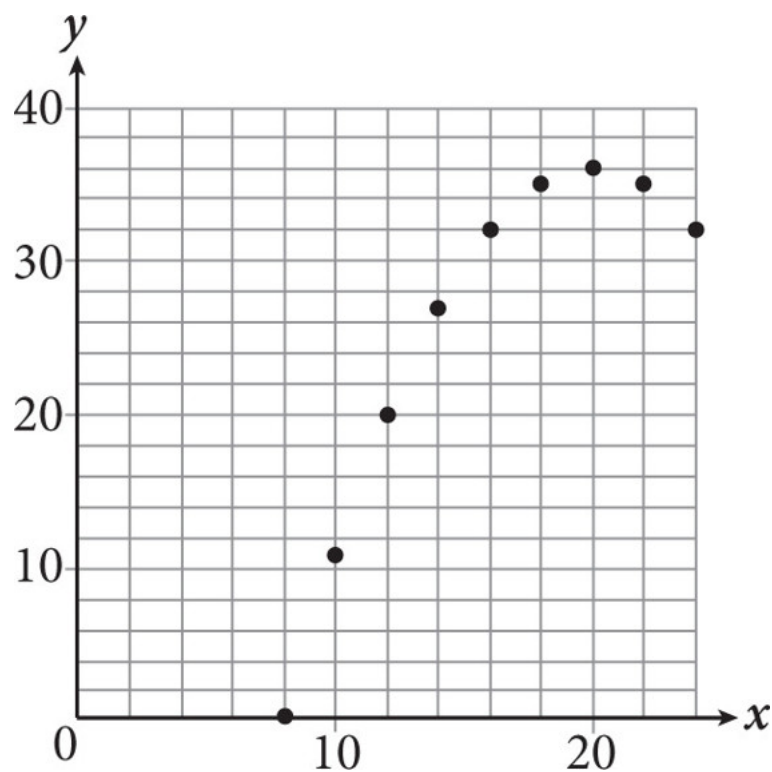


## Lecture 4 (Dt: 17.01.2020)

## Nearest Neighbour Classification

*Prepared by**Team 2 (Divyesh, Pulkit, Ankitha)***K Nearest Neighbour Algo (KNN)****Assumption**

Let's take an example:-We got a scatter plot and in this we need to predict the value of  $y$  at  $x=21$ .



We will predict the value of  $y$  to be around 36 but how can we predict the value, since the value can be any real number. We predicted by assuming that the data follows a distribution pattern and that it lies on the curve with respective value. Assumptions like this help us in most of the ML algorithms. Most ML algorithms start with an assumption and so, the same algorithm doesn't apply for all problems. KNN algorithm also starts with an assumption... i.e.

Similar points(K nearest neighbours) have similar labels.

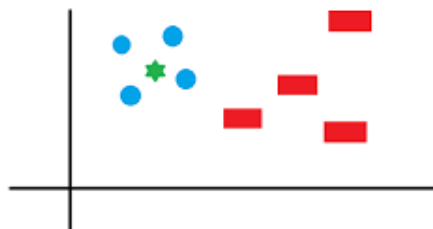
## Definition

### What is KNN?

- A powerful classification algorithm used in pattern recognition.
- K nearest neighbors stores all available cases and classifies new cases based on a *similarity measure* (e.g. **distance function**)
- One of the **top data mining algorithms** used today.
- A **non-parametric** lazy learning algorithm (An Instance-based Learning method).

6

## Illustration



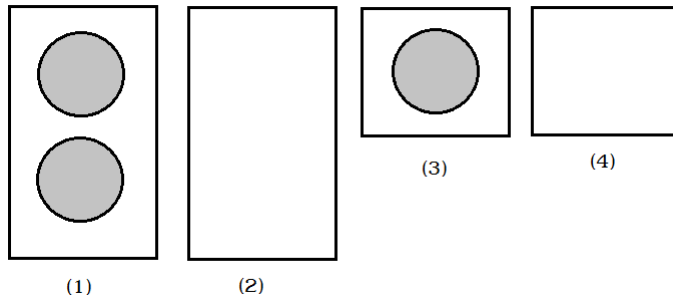
In above plot if we want to predict the class of point represented by star, then by KNN, it can be observed that its nearest neighbors are the points represented by blue circles. So the star most likely belongs to the class of blue circles not red rectangles.

In this case, the nearest neighbors are identified by the euclidean distance. The **distance metric or similarity measure** for each problem may be different depending upon the problem. Examples of few distance metrics are euclidean, manhattan, mahalanobis, minkowski, cosine similarity..., etc

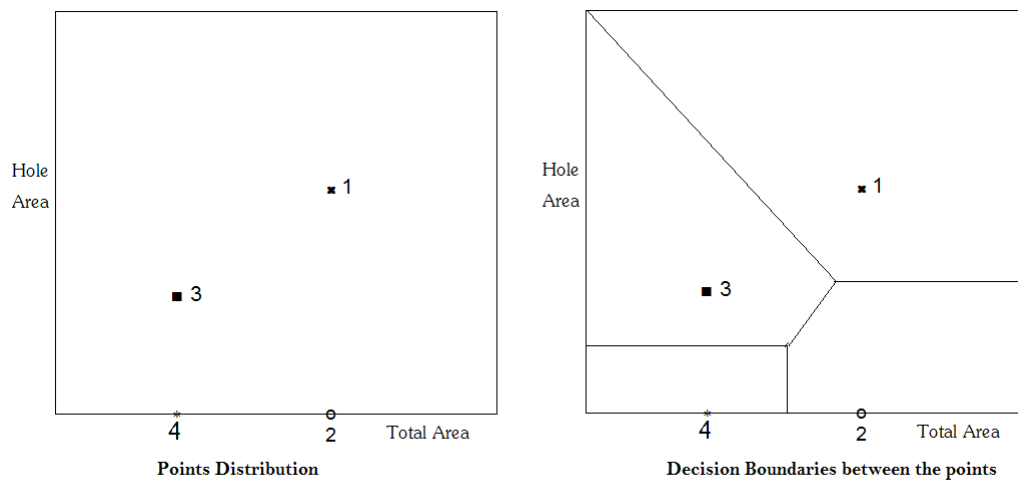
Most common application of KNN is **Information Retrieval**.

## Voronoi Diagram and Decision Boundary

### Intuition Sorting example



In the above example, given 4 instances, let us map Total Area Vs. Hole Area



Decision boundaries are the boundaries that divide the classes. Any point on one side of the boundary will be closest to the class on the respective side of the boundary. In the above figure, the decision boundaries have divided the space into 4 regions corresponding to classes 1, 2, 3, 4.

The diagram on the right, showing the decision boundaries and regions is called **Voronoi diagram**. Voronoi diagram partitions the space into regions. The points on boundary lie at the same distance from samples on both sides.

# Nearest-neighbor classification

- Use the intuition to classify a new point  $x$ :

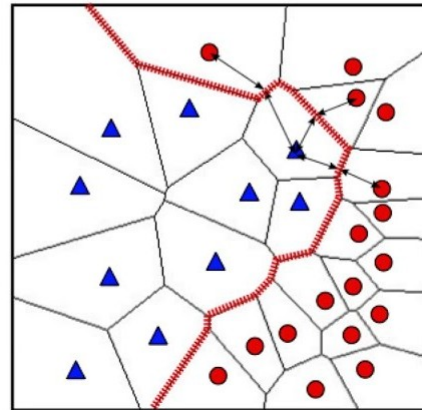
- find the most similar training example  $x'$
- predict its class  $y'$

- Voronoi tessellation

- partitions space into regions
- boundary: points at same distance from two different training examples

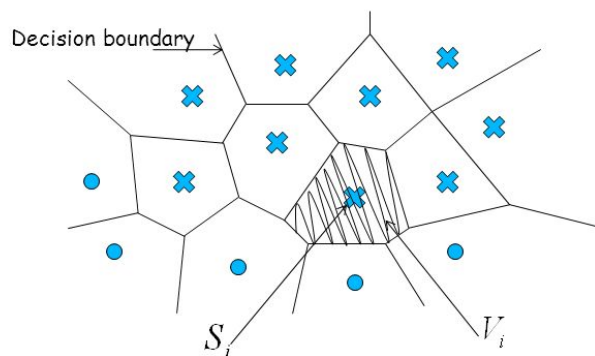
- decision boundary

- non-linear, reflects classes well
- compare to NB, DT, logistic
- impressive for simple method



Copyright © 2014 Victor Lavrenko

## Voronoi Diagrams



$V_i$  is a polygon such that any point that falls in  $V_i$  is closer to  $S_i$  than any other sample  $S_j$ .

Decision boundary helps us to distinguish the areas according to which we predict the nearest training example in Voronoi Diagram.

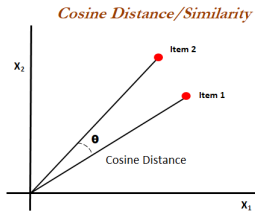
Decision boundaries are not necessarily linear. But in linear classification, the linear boundaries can be a point in 1D, line in 2D and a plane in 3D.

## Example

Distance or Similarity measures play an important role in identifying the nearest neighbours. We cannot always use same distance measures for every problem.

For example if we take a case in which we are given plot of celebrity magazine figure and wired magazine label and we need to predict the most likely article from set of articles. Does KNN work here? If so, how can we identify similar points? What should be the distance measure?

Here euclidean distance may not work, so we treat each point as a vector and use cosine distance or cosine similarity measure to compute distance.



$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}, \quad (1)$$

Another example where euclidean distance metric would not be suitable is finding distance between two cities on earth. Since the surface of earth is a spheroid, the cities lie on a manifold and hence, the distance should be calculated along the manifold.

Another similarity measure approach for word based problems is shape-context matching.

Assumption is important for an algorithm and hence, assumptions made should be true for the problem and data.

## KNN Classification

Given a pair of training examples  $x_i, y_i$ , we want to classify a testing point  $X$ .

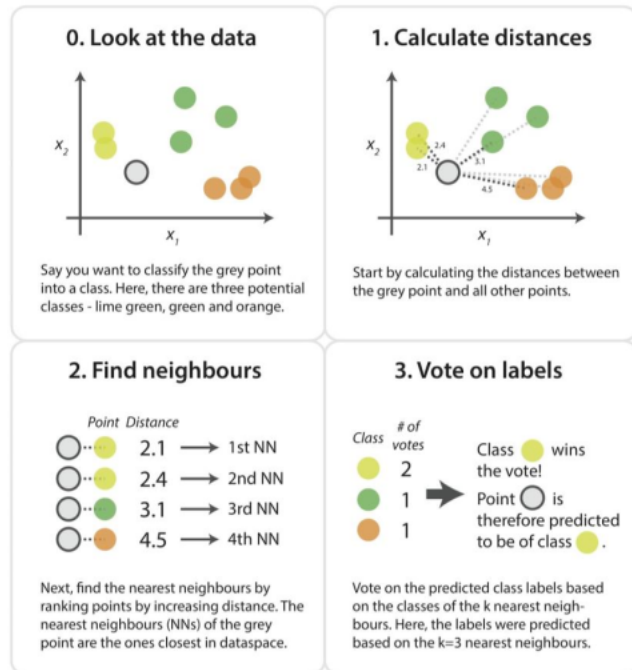
### Knn Algorithm

- Compute distance from the test point to all the training examples.
- Select  $k$  closest instances and their labels.
- Output the weighted mean of  $k$  labels.

Example of weights( $w$ ) for the  $k$  instances is distance based weighting (Inverse Distance Weighting)

where  $0 \leq w_i \leq 1$ . Algorithms should handle exceptions.

# k-NN algorithm in pictures



(2)

## Value of K

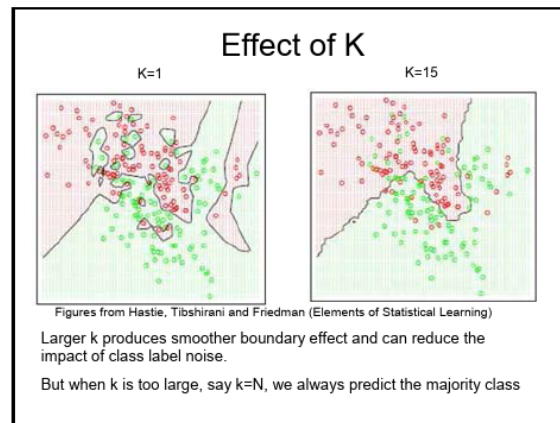
- Value of K has strong effect on KNN performance.
- Check accuracy/any other measure, for different values of K.
- K is usually an odd number.

## Large K

- Everything classified as most common class.
- Large value make it computationally expensive.

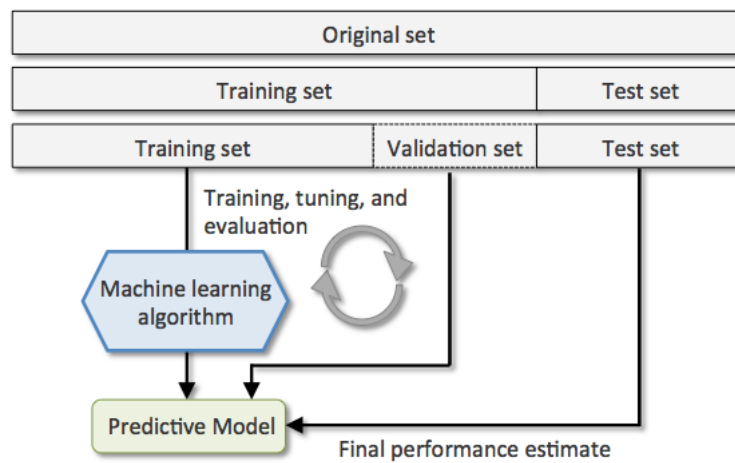
## Small K

- Small value of k means that noise will have a higher influence on the result.
- Highly variable unstable decision boundaries.
- Small change in training set will cause huge changes in classification and accuracy.



(3)

Rule of thumb:  $K < \sqrt{n}$ , where  $n$  is the no. of training examples.



## Distance Measures

### Key components

- Defines which examples are similar and which aren't.
- Can have strong effect on performance of the model.

### Distance Norms/Vector Norms

For a point  $(x_1, x_2, \dots, x_n)$  and a point  $(y_1, y_2, \dots, y_n)$ , the Minkowski distance of order  $p$  ( $p$ -norm distance) is defined as:

- 1 Norm Distance (Manhattan Distance)  
The distance between two points measured along axes at right angles. In a plane with  $p_1$  at  $(x_1, y_1)$  and  $p_2$  at  $(x_2, y_2)$ , it is  $\text{abs}(x_1 - x_2) + \text{abs}(y_1 - y_2)$ .  
Also known as rectilinear distance, Minkowski's  $L_1$  distance, taxi cab metric, or city block distance. Hamming distance can be seen as Manhattan distance between bit vectors.

$$\text{Manhattan distance} = \sum_{i=1}^n |x_i - y_i|$$


---

(4)

- 2 Norm Distance (Euclidean Distance)

The most familiar distance measure is the one we normally think of as distance. An n-dimensional Euclidean space is one where points are vectors of n real numbers. The conventional distance measure in this space, which we shall refer to as the L2-norm, is defined:

$$\text{Euclidean Distance} = d = \sqrt{\sum_{i=1}^N (X_i - Y_i)^2}$$
(5)

symmetric, spherical, treats all dimensions equally.  
sensitive to extreme differences in single attribute.

- p Norm Distance (Minkowski Distance)

The Minkowski distance is a metric in a normed vector space which can be considered as a generalization of both the Euclidean distance and the Manhattan distance.

Minkowski distance is typically used with p being 1 or 2, which correspond to the Manhattan distance and the Euclidean distance, respectively. In the limiting case of p reaching infinity, we obtain the Chebyshev distance:

The Minkowski distance can also be viewed as a multiple of the power mean of the component-wise differences.

$$\text{Minkowski} = \sqrt[p]{\sum_{i=1}^n |x_i - y_i|^p}$$
(6)

- Infinity Norm Distance (Chebyshev Distance)

Chebyshev distance (or Tchebychev distance), maximum metric, or L infinite metric is a metric defined on a vector space where the distance between two vectors is the greatest of their differences along any coordinate dimension.

It is also known as chessboard distance, since in the game of chess the minimum number of moves needed by a king to go from one square on a chessboard to another equals the Chebyshev distance between the centers of the squares, if the squares have side length one, as represented in 2-D spatial coordinates with axes aligned to the edges of the board.



## Chebyshev distance

In case of  $q \rightarrow \infty$ , the distance equals to the maximum difference of the attributes. Useful if the worst case must be avoided:

$$\begin{aligned} d_{\infty}(X, Y) &= \lim_{q \rightarrow \infty} \left( \sum_{i=1}^n |x_i - y_i|^q \right)^{1/q} \\ &= \max(|x_1 - y_1|, |x_2 - y_2|, \dots, |x_n - y_n|) \end{aligned}$$

Example:

$$d_{\infty}((2,8), (6,3)) = \max(|2-6|, |8-3|) = \max(4,5) = 5$$

(7)

## Hamming Distance

The number of bits which differ between two binary strings.

The Hamming distance can be interpreted as the number of bits which need to be changed (corrupted) to turn one string into the other. Sometimes the number of characters is used instead of the number of bits. Hamming distance can be seen as Manhattan distance between bit vectors.

No. of attributes where  $x, x'$  differ. (XOR)

When bits are same its 0 else 1.

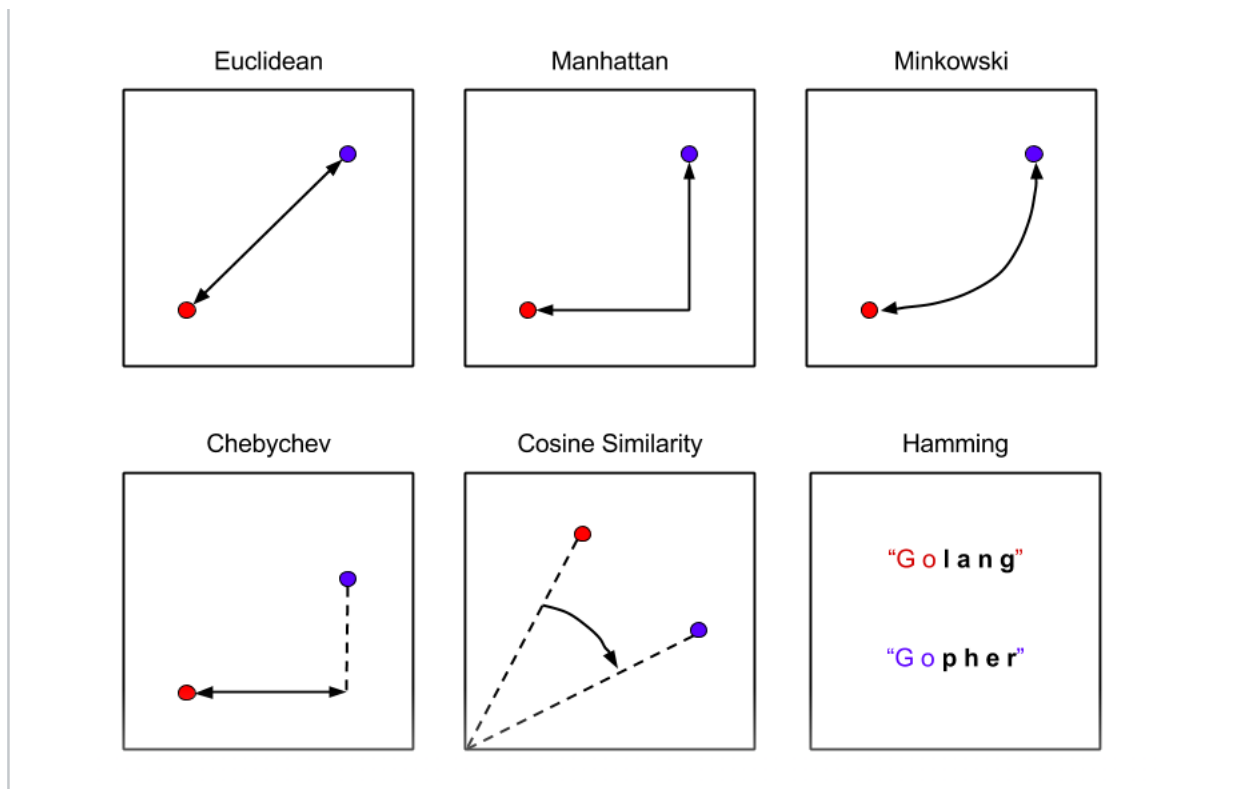
$p=0$  and logical and.

## Cosine distance

Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

(8)



## Kullback-Leiber(KL) Divergence

A measure of how one distribution is different from a second, reference distribution.

$$D_{KL}(P, Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

It is not a symmetric measure, i.e.  $D(P, Q) \neq D(Q, P)$

## KNN: practical issues

### Resolving Ties

When we will do nearest neighbours we will sometimes get ties i.e. we have equal number of positives and negatives.

- If we have odd number of classes then we can just use an odd value of K. But this does not resolve our issue for multi-class.
  - For multi-class we need to break ties:
    - random: flip a coin to decide positive/negative.
    - priority: we can pick a class with greater prior.
    - Nearest: we can use 1-nn classifier as it does not have ties.
- In practice these methods do have some effect on performance but that's not that much as we expect.

## Missing values

If in our representation we don't know the value, we cannot find the distance. We don't have a naive based (it handles missing values automatically).

- have to "fill in", because we want to compute the distance.
- Key concern: While filling the empty values we need to take care of distance, as distance should be affected as little as possible.
- Reasonable Choice: now to fulfill the above key concern we fill average value across entire dataset and use that instead of missing value.  
This does have some downsides but this is the best we can do and every strategy will have some loopholes.

## Some Use Cases for KNN

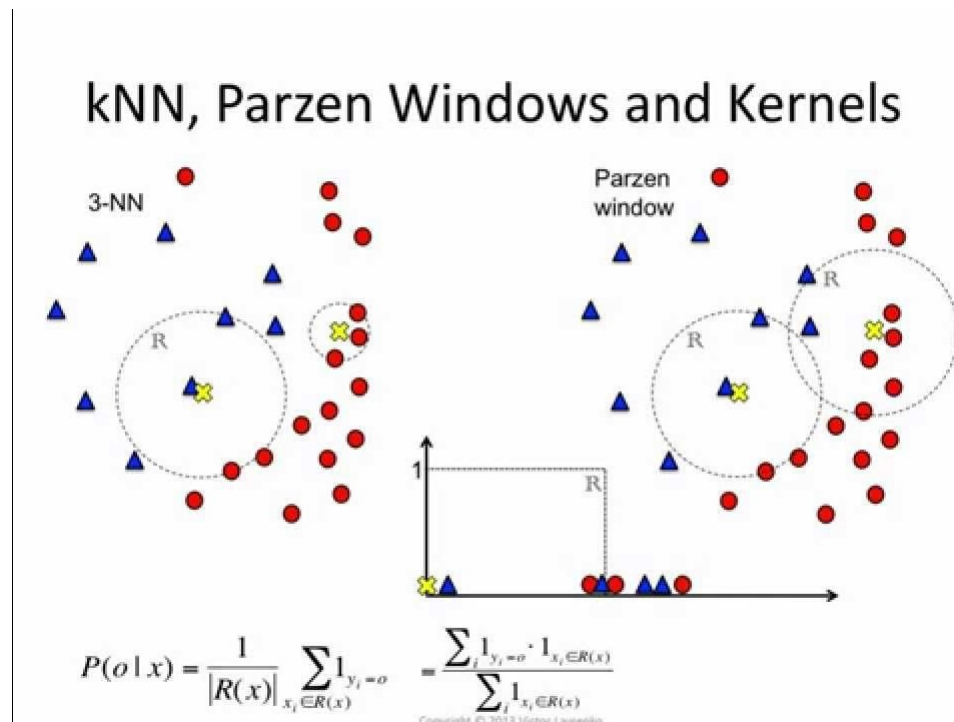
- **MNIST Dataset:** The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. A very low test error rate of 5% has been reached using kNN with only L2 Euclidean distance as distance measure (LeCun et al. 1998).
- **IM2GPS: estimating geographic information from a single image:** A simple algorithm (that uses kNN) was proposed for estimating a distribution over geographic locations from a single image using a purely data-driven scene matching approach, with a dataset of over 6 million GPS-tagged images from the Internet. The estimated image location is represented as a probability distribution over the Earth's surface.
- **Im2Text: Describing Images Using 1 Million Captioned Photographs:** A collection of 1 million images with associated visually relevant captions was created by filtering the noisy results from large number of Flickr queries. This collection helped in description generation using relatively simple non-parametric methods.
- **A Distributed Representation Based Query Expansion Approach for Image Captioning**

## Parzen Windows

- Circle of fixed radius  $R$  as a window around data point is drawn (for 2-dimensional data).
- Closer points inside window have more weight. Weight decreases rapidly as we move away from data point, Ex. In the form of probabilistic decaying function.

$$f(\mathbf{X}) = \text{sign}(\sum y_i K(\mathbf{X}_i, \mathbf{X})).$$

where the kernel function  $K$  is the radial basis function.



## KNN Pros and Cons

### Pros

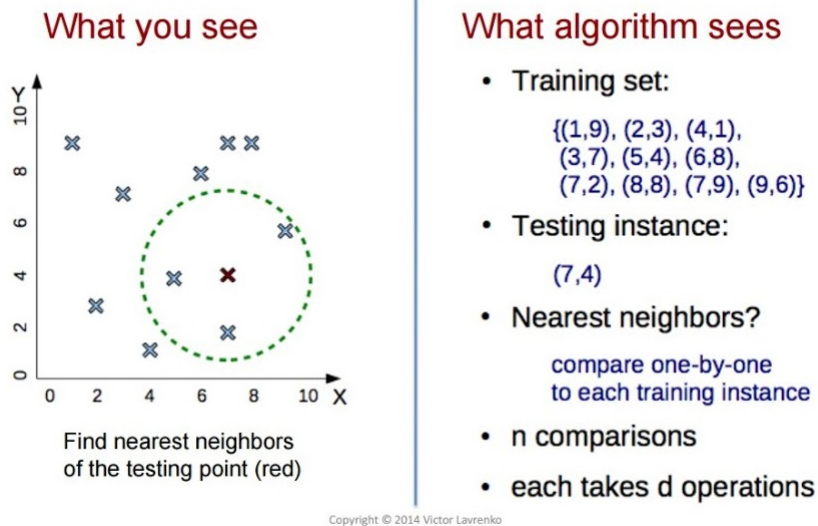
- Simple and Powerful
- No assumptions about data useful, for example, for nonlinear data.
- Non Parametric - data driven
- The 2 variables of choice are k and DM (Distance Metric) - Should be guided by the kind of data.
- Easy to update in an online setting

### Cons

- Missing values need to be handled
- Sensitive to outliers and noise
- Computationally expensive - Progress becomes slower as N grows (N - No. of training samples)
- Hence, it is required to make KNN faster for better performance.

## Making KNN Faster

### Why is kNN slow?



### Ideas

Reduce d - Dimensionality Reduction    Reduce N - Donot compare with all the training samples

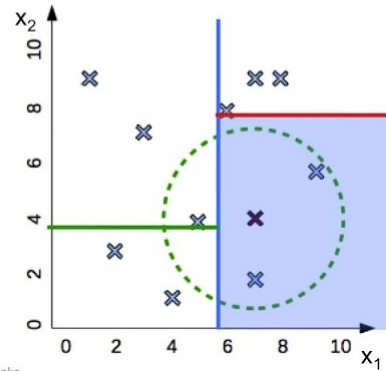
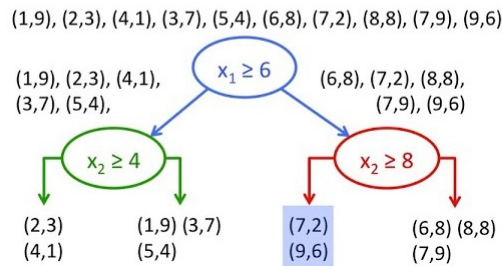
- Kd-Trees
- Local Sensitive Hashing
- Inverted Index

### Low Dimensional Data - Kd-Tree

Build Kd-Tree

## K-D tree example

- Building a K-D tree from training data:
  - pick random dimension, find median, split data, repeat
- Find NNs for new point (7,4)
  - find region containing (7,4)
  - compare to all points in region



- Pick random dimension
- Find median
- Split Data
- Repeat

In a Kd-tree, if  $d$  decisions are taken, the space is divided into  $2^d$  regions and  $2^d \ll N$ . Hence the algorithm needs to run for only  $2^d$  regions instead of  $N$  samples making KNN fast.

## High Dimensional Data - Local Sensitive Hashing

Local Sensitive Hashing (LSH)

- Generate  $h$  random hyperplanes
- The space is divided into  $2^h$  regions
- Run the point through  $h$  hyperplanes and return 0 or 1
- It returns 1 only if the point lies above the hyperplane,
- Each point generates a  $h$  digit hash.
- Data points that are similar in high-dimension will have a larger chance of generating the same hash value

$$[hash(a)]_i = \begin{cases} 1 & w^T a > 0 \\ 0 & w^T a \leq 0 \end{cases}, i = 1 \dots K$$

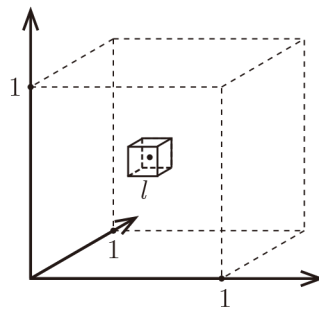
where  $w^T a = 0$  is the hyperplane

## High Dimensional Sparse Data - Inverted Index

- Data Structure used by search engines (eg. Google..., etc)
- For each word, find the indices of documents that have it
- Use these indices to find the appropriate document for a given query.

## The Curse of Dimensionality

Given a normalized cube of side = 1 with a total of N points.



**Assuming uniform distribution**, what should be the side of cube ( $l$ ) such that it contains  $k$  points out of total  $N$  points?

Since it is 3-Dimensional,

$$l^3 = k/N$$

$$\Rightarrow l = \sqrt[3]{k/N}$$

For  $d$ -Dimensions,

$$l^d = k/N$$

$$\Rightarrow l = \sqrt[d]{k/N}$$

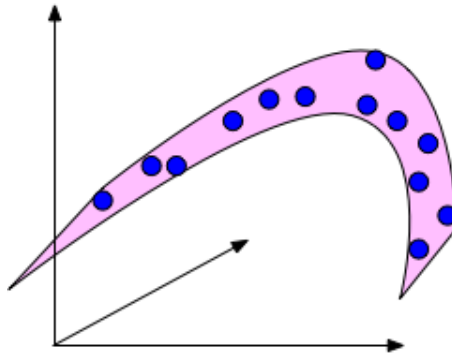
Now, as  $d$  increases,  $1/d$  tends to 0  $\Rightarrow \sqrt[d]{k/N}$  tends to 1  $\Rightarrow l$  tends to 1, expanding the search to entire space.

This is why KNN does not work well on a high dimensional uniformly distributed data.

But in real situations, data is generally not uniformly distributed, instead lies on a manifold.

## Manifold to Rescue

- Euclidean distance does not work for manifolds but most manifolds can have euclidean distances at local level.
- Here, the true dimensionality of the data can be much lower than its ambient space.



An example of a data set in 3d that is drawn from an underlying 2-dimensional manifold. The blue points are confined to the pink surface area, which is embedded in a 3-dimensional ambient space.