Machine Learning

AI3002

kNN Algorithm: Overview, Analysis, Convergence and Extensions

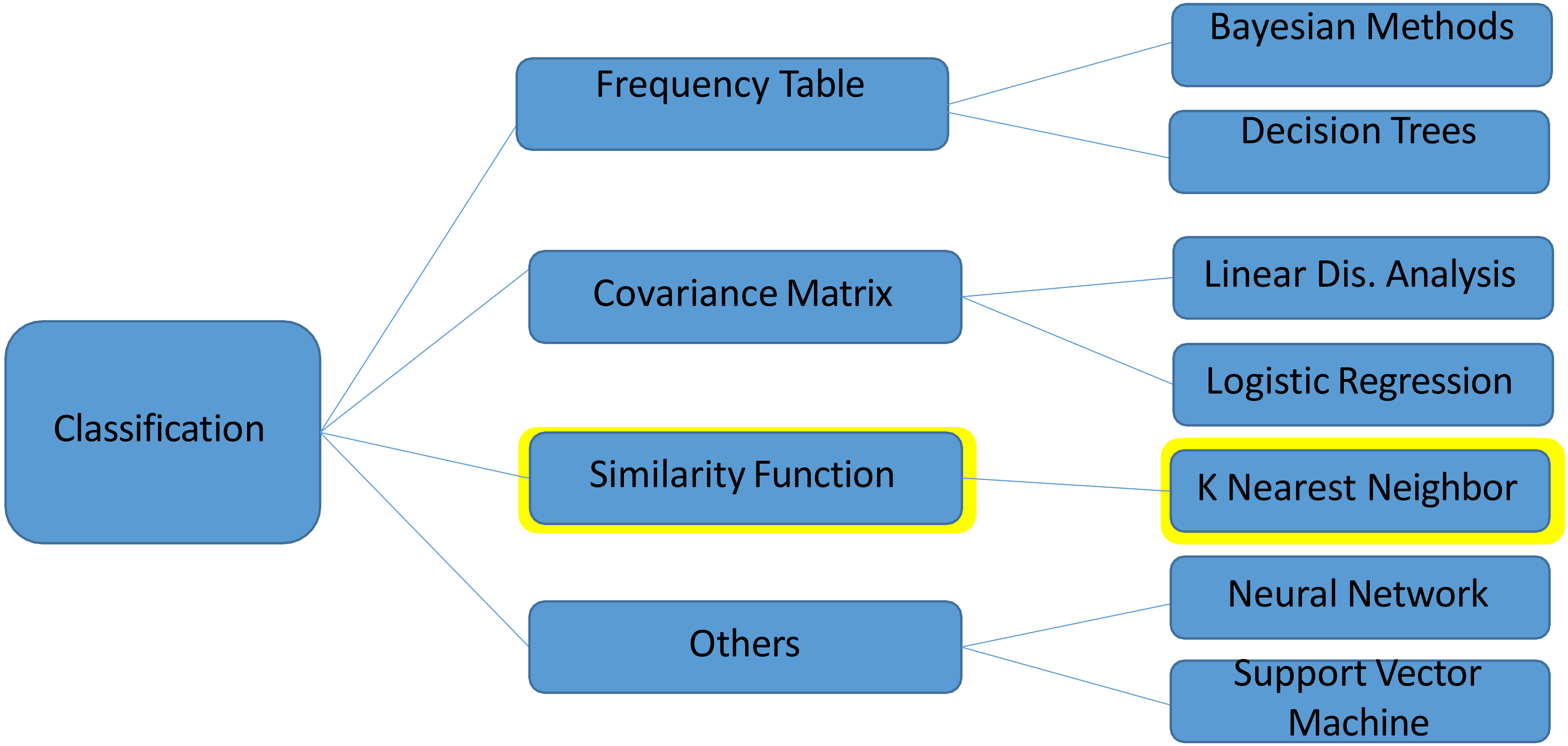
Outline

- k-Nearest Neighbor (kNN) Algorithm Overview
- Algorithm Formulation
- Distance Metrics
- Choice of k
- Algorithm Convergence
- Storage, Time Complexity Analysis
- Fast kNN
- The Curse of Dimensionality

Supervised Learning

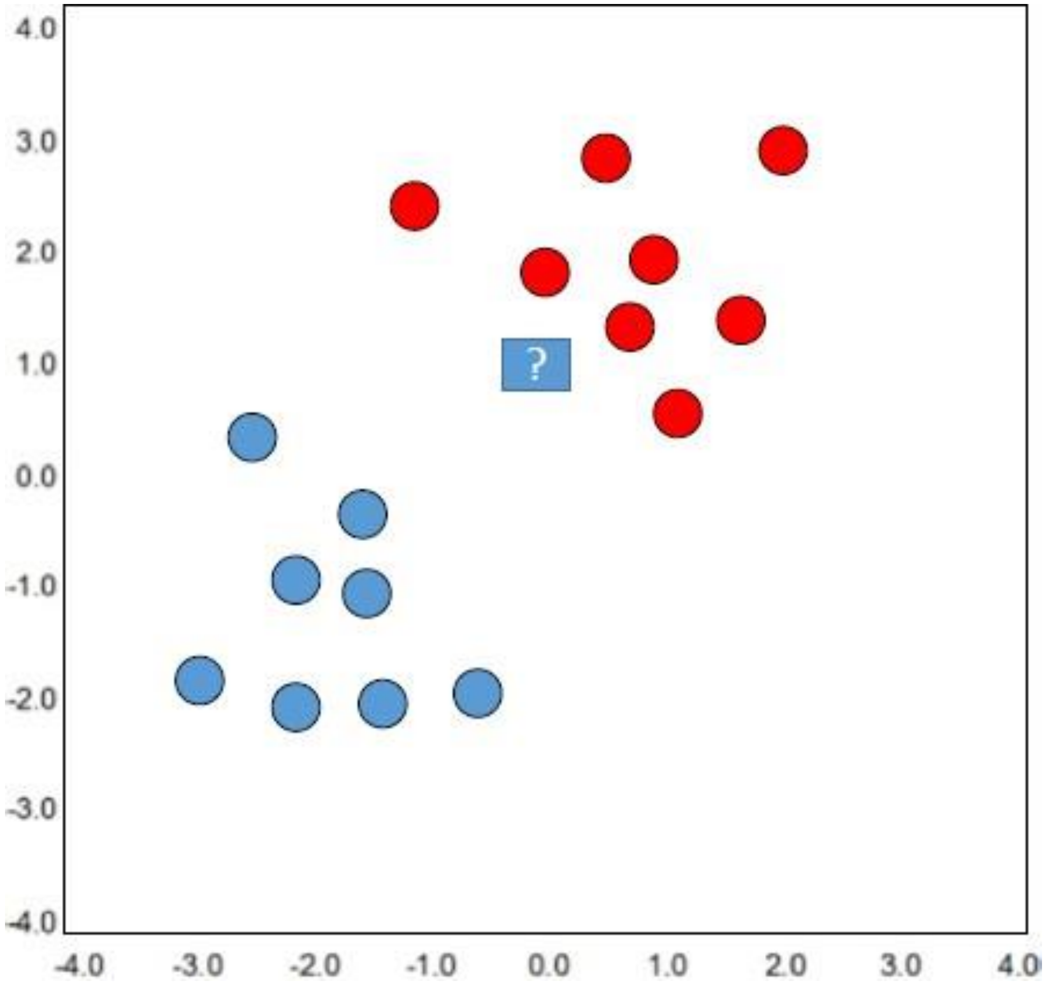
Classification Algorithms or Methods

Predicting a categorical output is called classification



k-Nearest Neighbor (kNN) Algorithm

Idea:



- Two classes, two features
- We want to assign label to unknown data point?
- Label should be red.

k-Nearest Neighbor (kNN) Algorithm

Idea:

- We have similar labels for similar features.
- We classify new test point using similar training data points.

Algorithm overview:

- Given some new test point x for which we need to predict the class y.
- Find **most similar** data-points in the training data.
- Classify x "like" these **most** similar data points.

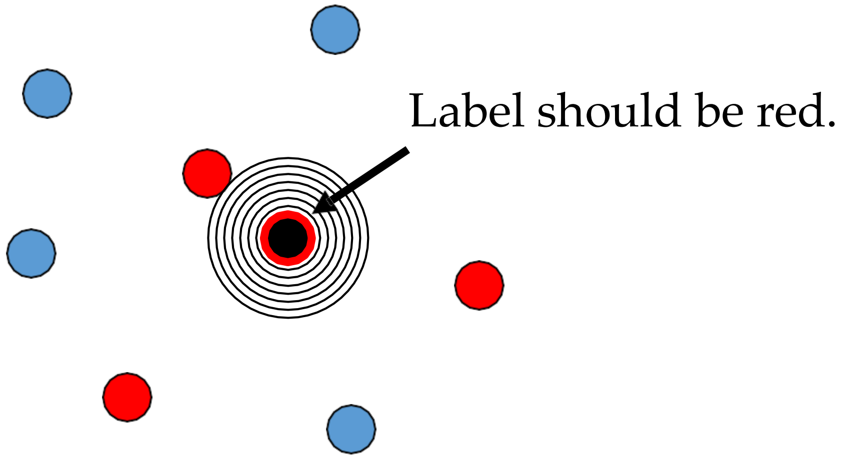
Questions:

- How do we determine the similarity?
- How many similar training data points to consider?
- How to resolve inconsistencies among the training data points?

k-Nearest Neighbor (kNN) Algorithm

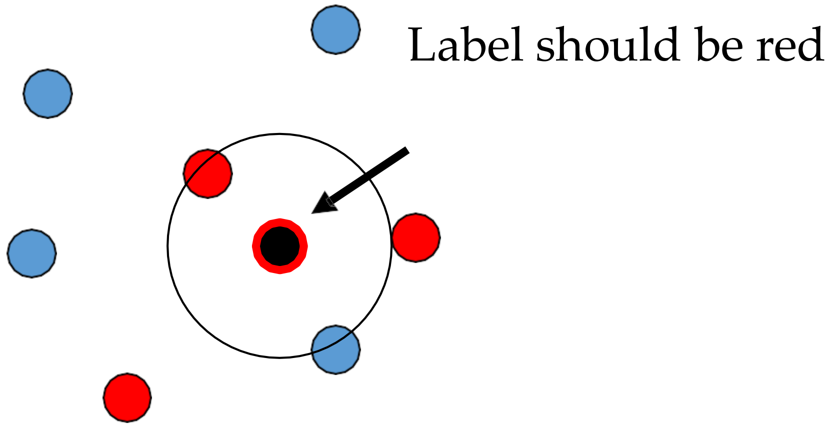
1-Nearest Neighbor:

Simplest ML Classifier
*Idea: Use the label of the **closest** known point*

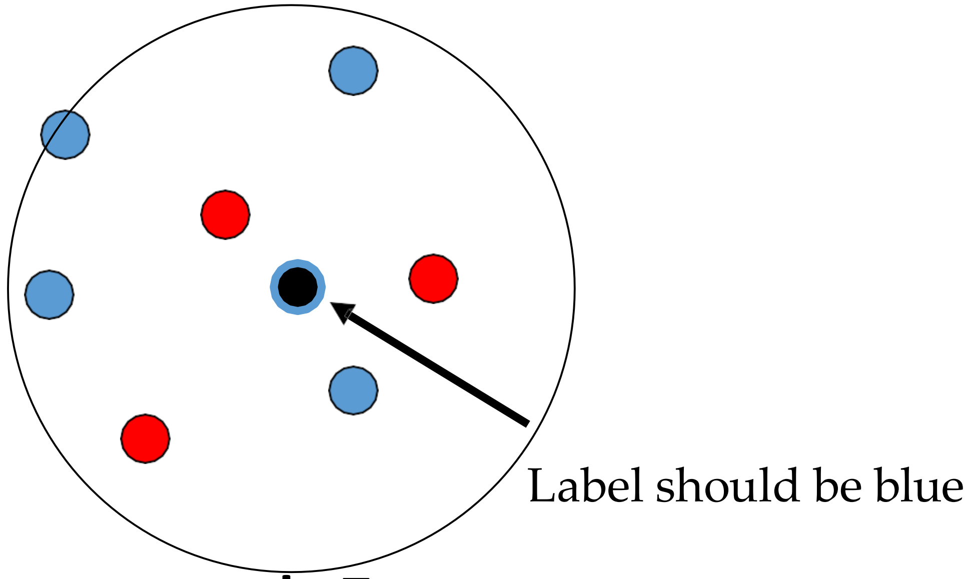


Generalization:

*Determine the label of **k** nearest neighbors and assign the most frequent label*



k=3



k=7

k-Nearest Neighbor (kNN) Algorithm

Formal Definition:

- We assume we have training data D given by

$$D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\} \subseteq \mathcal{X}^d \times \mathcal{Y}$$

- $\mathcal{Y} = \{1, 2, \ldots, M\}$ (M-class classification)

Denote the set of the k nearest neighbors of x as S_x.

- For a point $\mathbf{x} \in \mathcal{X}^d$, we define a set $S_\mathbf{x} \subseteq D$ as a set of k neighbors.

- Using the function 'dist' that computes the distance between two points in $\mathcal{X}^d$, we can define a set $S_\mathbf{x}$ of size k as

$$\mathrm{dist}(\mathbf{x}, \mathbf{x}') \geq \max_{(\mathbf{x}'', y'') \in S_\mathbf{x}} \mathrm{dist}(\mathbf{x}, \mathbf{x}''), \quad \forall (\mathbf{x}', y') \in D \backslash S_\mathbf{x}$$

Interpretation:

Every point in D but not in $S_\mathbf{x}$ is at least as far away from $\mathbf{x}$ as the furthest point in $S_\mathbf{x}$.

k-Nearest Neighbor (kNN) Algorithm

Formal Definition:

- Using the $S_{\mathbf{x}}$, we can define a classifier as a function that gives us most frequent label of the data points in $S_{\mathbf{x}}$

$$h(\mathbf{x}) = \text{mode}(\{y'' : (x'', y'') \in S_{\mathbf{x}}\})$$

where mode$(\cdot)$ means to select the label of the highest occurrence.

- Instance-based learning algorithm; easily adapt to unseen data

k-Nearest Neighbor (kNN) Algorithm

Decision Boundary:

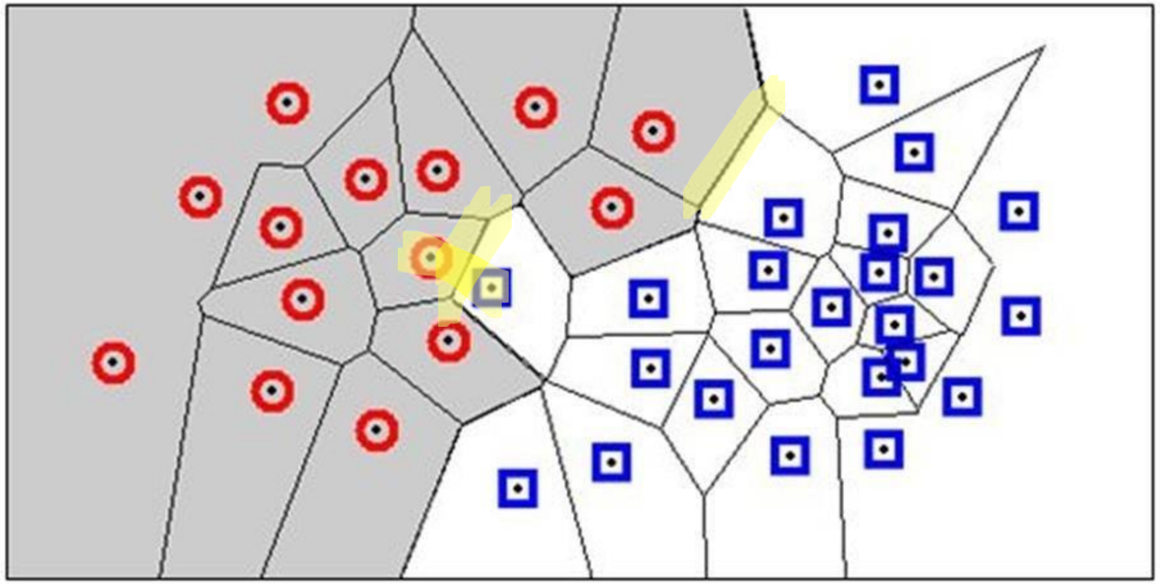
For $k = 1$, kNN defines a region, called decision boundary or region, in the space. Such division of the featue space is referred to s Voronoi partitioning.

We can define a region R_i associated with the feature point $\mathbf{x}_i$ as

$$R_i = \{\mathbf{x} : \text{dist}(\mathbf{x}, \mathbf{x}_i) < \text{dist}(\mathbf{x}, \mathbf{x}_j),\ i \neq j\}$$

For example, Voronoi partitioning using Euclidean distance in two-dimensional space.

Classification boundary changes with the change in k and the distance metric.



k-Nearest Neighbor (kNN) Algorithm

Characteristics of kNN:

- No assumptions about the distribution of the data
- Non-parametric algorithm
  - No parameters
- Hyper-Parameters
  - k (number of neighbors)
  - Distance metric (to quantify similarity)

k-Nearest Neighbor (kNN) Algorithm

Characteristics of kNN:

- Complexity (both time and storage) of prediction increases with the size of training data.

- Can also be used for regression (average or inverse distance weighted average)
  - For example,

$$y = \frac{1}{k}\sum_{i=1}^{k} y_i, \quad (\mathbf{x}_i, y_i) \in S_{\mathbf{x}}$$

k-Nearest Neighbor (kNN) Algorithm

Practical issues:

- For binary classification problem, use odd value of k. Why?
- In case of a tie:
  - Use prior information
  - Use 1-nn classifier or k-1 classifier to decide
- Missing values in the data
  - Average value of the feature.

k-Nearest Neighbor (kNN) Algorithm

We need to define distance metric to find the set of k nearest neighbors, S_x

- We defined a set $S_{\mathbf{x}}$ of size k as

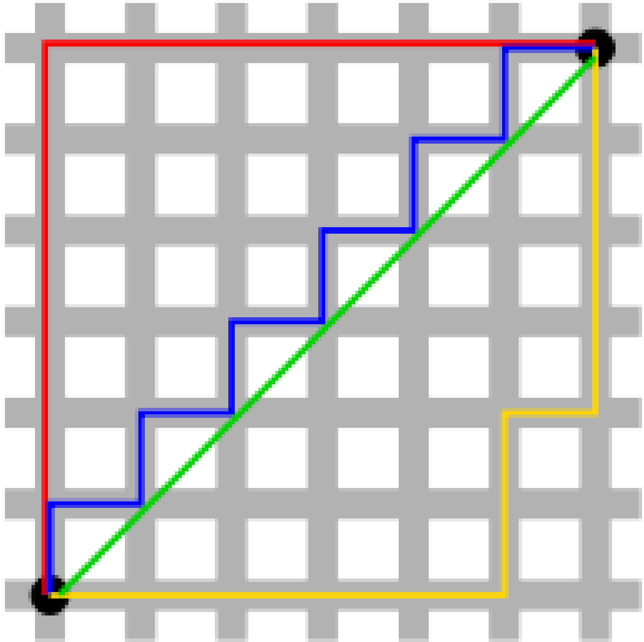
$$\mathrm{dist}(\mathbf{x}, \mathbf{x}') \geq \max_{(\mathbf{x}'', y'') \in S_{\mathbf{x}}} \mathrm{dist}(\mathbf{x}, \mathbf{x}''), \quad \forall (\mathbf{x}', y') \in D \backslash S_{\mathbf{x}}$$

k-Nearest Neighbor (kNN) Algorithm

Distance Metric:

- Euclidean $$\mathrm{dist}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2 = \sqrt{\sum_{i=1}^{d} (x_i - x'_i)^2}$$

- Manhattan $$\mathrm{dist}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_1 = \sum_{i=1}^{d} |x_i - x'_i|$$



Euclidean

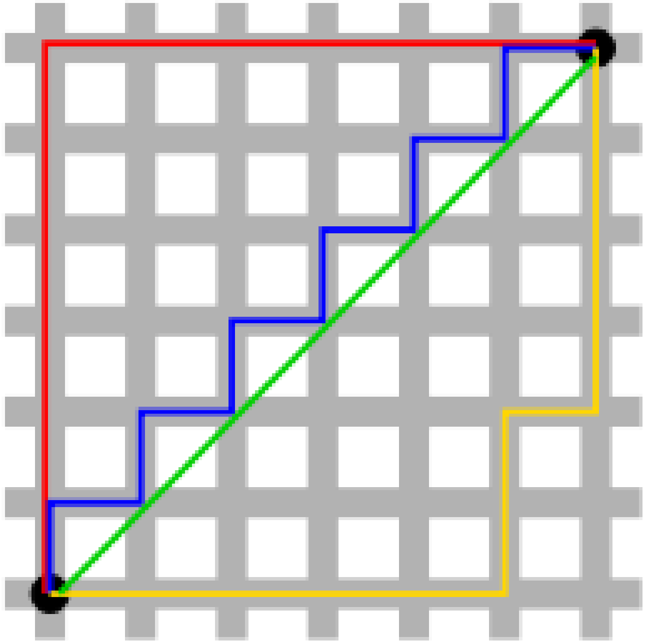
Manhattan
Manhattan
Manhattan

k-Nearest Neighbor (kNN) Algorithm

Norm of a vector

- p-norm of a vector $\mathbf{x} \in \mathbf{R}^d$

$$\|\mathbf{x}\|_p = \Big(\sum_{i=1}^{d} |x_i|^p\Big)^{1/p}, \quad p \geq 1$$



Euclidean $6\sqrt{2}$

Manhattan
Manhattan 12
Manhattan

Properties of Norm

Non-negative, $\|\mathbf{x}\|_p \geq 0$

Absolutely homogenous: $\|\alpha\mathbf{x}\|_p = |\alpha|\|\mathbf{x}\|_p$

$\|\alpha\mathbf{x}\|_p = 0 \iff \mathbf{x} = 0$

Triangular inequality, $\|\mathbf{x} + \mathbf{x}'\|_p \leq \|\mathbf{x}\|_p + \|\mathbf{x}'\|_p$

$$\|\mathbf{x}\|_q \leq \|\mathbf{x}\|_p, \quad p \leq q$$

k-Nearest Neighbor (kNN) Algorithm

Norm of a vector

$$p\text{-norm distance} = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

$$\begin{aligned}\text{infinity norm distance} &= \lim_{p\to\infty}\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} \\ &= \max\left(|x_1 - y_1|, |x_2 - y_2|, \ldots, |x_n - y_n|\right).\end{aligned}$$

p need not be an integer, but it cannot be less than 1, because otherwise the triangle inequality does not hold.

k-Nearest Neighbor (kNN) Algorithm

Distance Metric:

- Euclidean $\text{dist}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2 = \sqrt{\sum_{i=1}^{d} (x_i - x_i')^2}$

- Manhattan $\text{dist}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_1 = \sum_{i=1}^{d} |x_i - x_i'|$

$p = 2$

$p = 1$

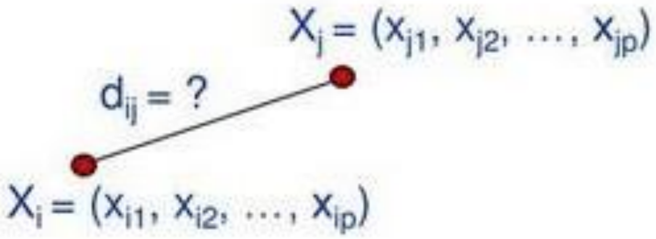
- Minkowski

$$\text{dist}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_p = \Big(\sum_{i=1}^{d} (|x_i - x_i'|)^p\Big)^{1/p}, \quad p \geq 1$$

$p = \infty$

$$\text{dist}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_\infty = \max_{i=1,2,\ldots,d} (|x_i - x_i'|)$$ Chebyshev Distance

k-Nearest Neighbor (kNN) Algorithm



- **Minkowski distance**

$$d(i,j) = \sqrt[q]{\underbrace{|x_{i1} - x_{j1}|}_{\text{1st dimension}}^{q} + \underbrace{|x_{i2} - x_{j2}|}_{\text{2nd dimension}}^{q} + \ldots + \underbrace{|x_{ip} - x_{jp}|}_{\text{pth dimension}}^{q}}$$

- **Euclidean distance**

$q = 2$

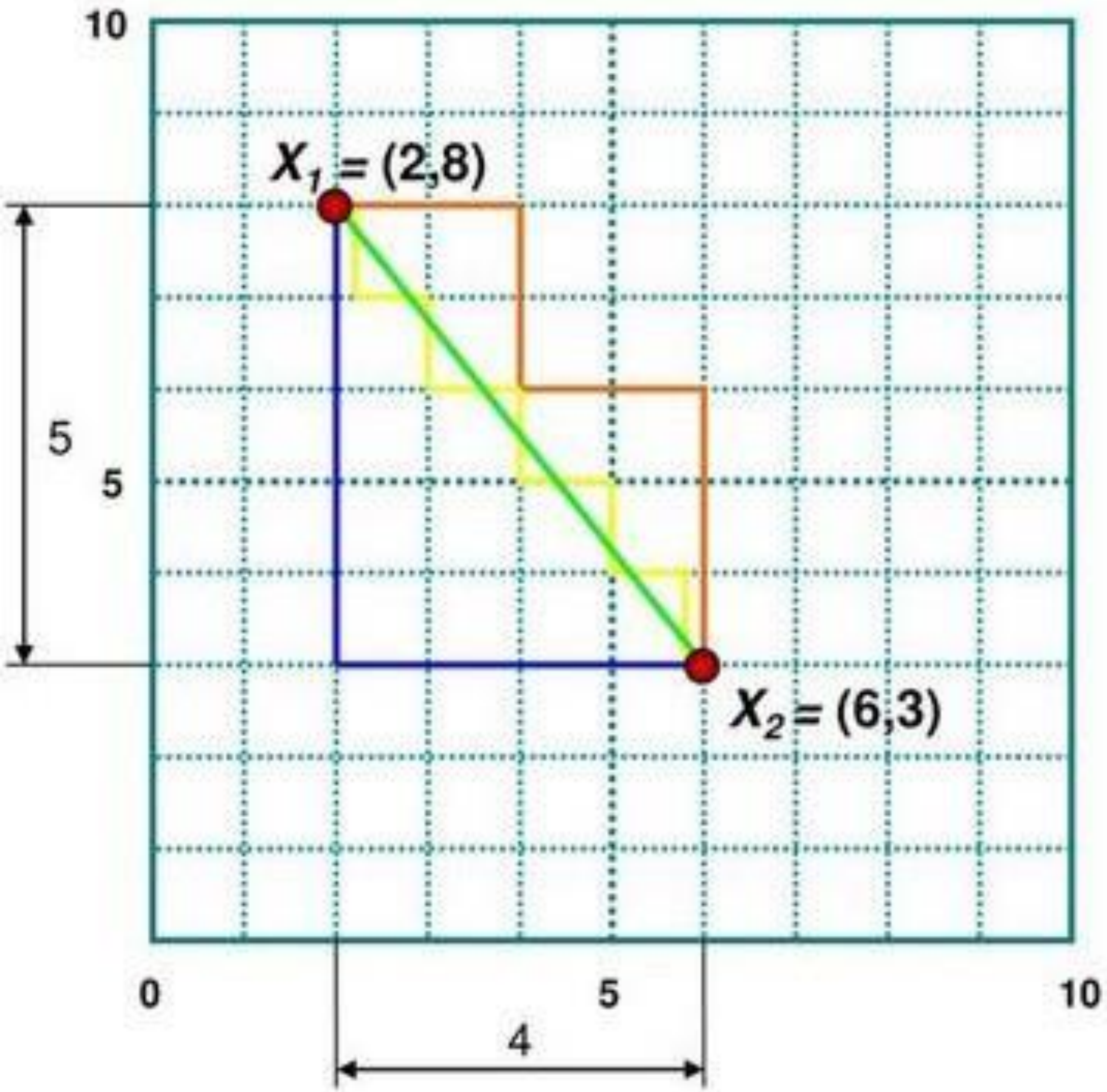
$$d(i,j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{ip} - x_{jp}|^2}$$

- **Manhattan distance**

$q = 1$

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ip} - x_{jp}|$$

k-Nearest Neighbor (kNN) Algorithm



2D example

$x_1 = (2,8)$

$x_2 = (6,3)$

Euclidean distance

$$d(1,2) = \sqrt{|2-6|^2 + |8-3|^2} = \sqrt{41}$$

Manhattan distance

$$d(1,2) = |2-6| + |8-3| = 9$$

k-Nearest Neighbor (kNN) Algorithm

Chebyshev distance

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

$$d_\infty(X,Y) = \lim_{q\to\infty}\left(\sum_{i=1}^{n}|x_i - y_i|^q\right)^{1/q}$$

$$= \max\left(|x_1 - y_1|, |x_2 - y_2|, \ldots, |x_n - y_n|\right)$$

Example:

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

k-Nearest Neighbor (kNN) Algorithm

Distance Metric:

Properties of Distance Metrics:

$$\text{Non-negative, } \operatorname{dist}(\mathbf{x}, \mathbf{x}') \geq 0$$

$$\text{Symmetric, } \operatorname{dist}(\mathbf{x}, \mathbf{x}') = \operatorname{dist}(\mathbf{x}', \mathbf{x})$$

$$\operatorname{dist}(\mathbf{x}, \mathbf{x}') = 0 \iff \mathbf{x} = \mathbf{x}'$$

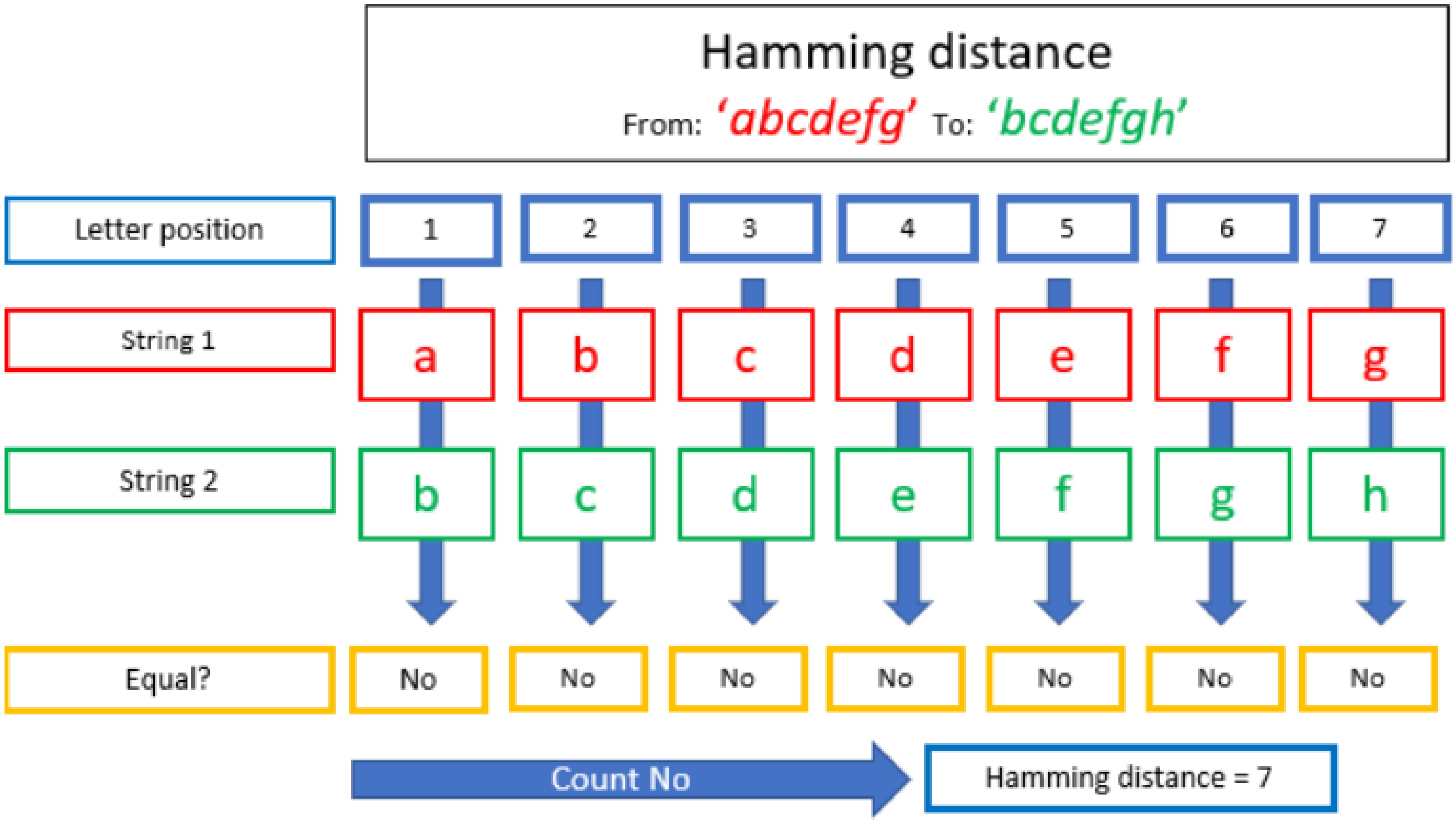
$$\text{Triangular inequality, } \operatorname{dist}(\mathbf{x}, \mathbf{x}') \leq \operatorname{dist}(\mathbf{x}', \mathbf{x}'') + \operatorname{dist}(\mathbf{x}'', \mathbf{x})$$

k-Nearest Neighbor (kNN) Algorithm

Distance Metric:

- For categorical variable, use Hamming Distance

$$\text{dist}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{d} 1 - \delta_{x_i - x_i'}$$



The **Hamming distance** between two words is the number of differences between corresponding bits.

Example:

Hamming distance 10101, 11110) is 3

k-Nearest Neighbor (kNN) Algorithm

Practical issues in computing distance:

- Mismatch in the values of data
  - Issue: Distance metric is mapping from d-dimensional space to a scaler. The values should be of the same order along each dimension.
  - Solution: Data Normalization

Outline

- k-Nearest Neighbor (kNN) Algorithm Overview
- Algorithm Formulation
- Distance Metrics
- **Choice of k**
- Algorithm Convergence
- Storage, Time Complexity Analysis
- Fast kNN
- The Curse of Dimensionality

k-Nearest Neighbor (kNN) Algorithm

Choice of k:

- k=1

Sensitive to noise

High variance

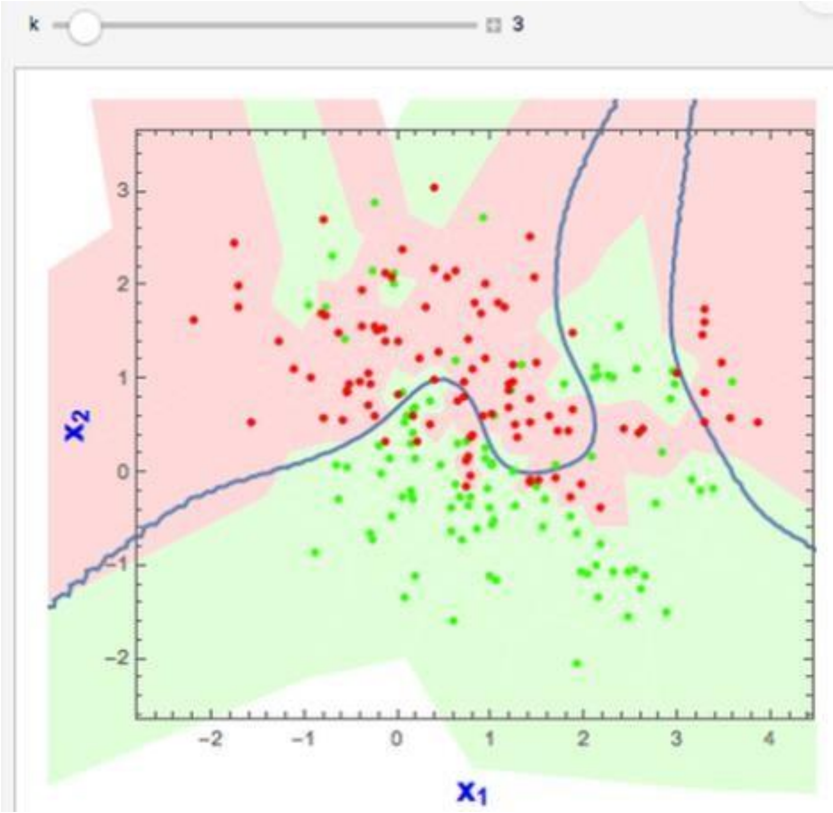
Increasing k makes algorithm less sensitive to noise

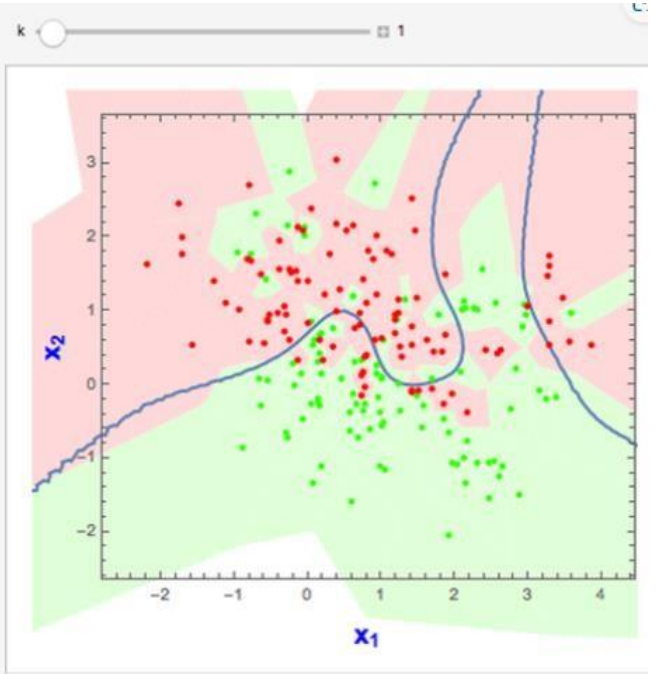
- k=n

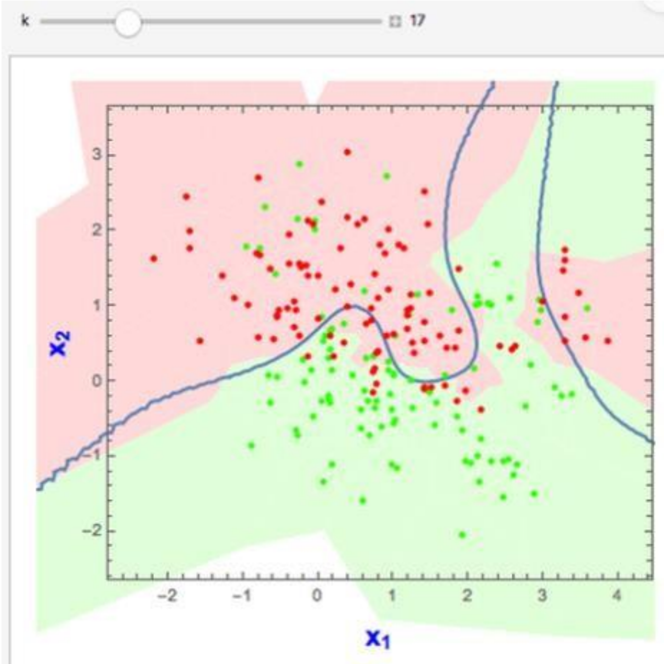
Decreasing k enables capturing finer structure of space

Idea: Pick k not too large, but not too small (depends on data)

How?







k-Nearest Neighbor (kNN) Algorithm

Choice of k:

- Learn the best hyper-parameter, k using the data.
- Split data into training and validation.
- Start from k=1 and keep iterating by carrying out (5 or 10, for example) cross-validation and computing the loss on the validation data using the training data.
- Choose the value for k that minimizes validation loss.
- This is the only learning required for kNN.

Outline

- k-Nearest Neighbor (kNN) Algorithm Overview
- Algorithm Formulation
- Distance Metrics
- Choice of k
- Algorithm Convergence
- Storage, Time Complexity Analysis
- Fast kNN
- The Curse of Dimensionality

k-Nearest Neighbor (kNN) Algorithm

Error Convergence:

We wish to analyze the error rate of the kNN classifier.

We will show that the error of 1-NN classifer converges as number of points in D increases.

To show the convergence, we will derive that 1-NN classifier is only a factor 2 worse than the best possible classifier.

k-Nearest Neighbor (kNN) Algorithm

Learning Problem

We represent the entire training data as

$$D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\} \subseteq \mathcal{X}^d \times \mathcal{Y}$$

Recall a problem in hand. We want to develop a model that can predict the label for the input for which label is unknown using kNN.

We assume that the data points $(\mathbf{x_i}, y_i)$ are drawn from some (unknown) distribution $P(X, Y)$.

k-Nearest Neighbor (kNN) Algorithm

Algorithm Computational and Storage Complexity:

Input/Output:

- We have a feature vector, $\mathbf{x}$ for which we want to predict label y.
- We have k and dist function.

Steps:

- We defined a set $S_{\mathbf{x}}$ of size k as

$$\operatorname{dist}(\mathbf{x}, \mathbf{x}') \geq \max_{(\mathbf{x}'', y'') \in S_{\mathbf{x}}} \operatorname{dist}(\mathbf{x}, \mathbf{x}''), \quad \forall (\mathbf{x}', y') \in D \backslash S_{\mathbf{x}}$$

- Classifier that gives us most frequent label of the data points in $S_{\mathbf{x}}$

$$h(\mathbf{x}) = \operatorname{mode}(\{y'' : (x'', y'') \in S_{\mathbf{x}}\})$$

k-Nearest Neighbor (kNN) Algorithm

Algorithm:

Steps:

Computational Complexity

1. Find distance between given test point and feature vector of every point in D.

   Noting n number of data points we have and each feature vector $\mathbf{x}$ is d-dimensional. $\mathcal{O}(dn)$

2. Find k points in D closest to the given test point vector to form a set S_x.

   Finding k-th smallest distance using median of medians method. $\mathcal{O}(n)$

   Finding k data-points in D with distance less than the k-th smallest distance $\mathcal{O}(n)$

3. Find the most frequent label in the set S_x and assign it to the test point. $\mathcal{O}(k)$

Computational Complexity: $\mathcal{O}(dn)$

Space Complexity: $\mathcal{O}(dn)$

Outline

- k-Nearest Neighbor (kNN) Algorithm Overview
- Algorithm Formulation
- Distance Metrics
- Choice of k
- Algorithm Convergence
- Storage, Time Complexity Analysis
- Fast kNN
- The Curse of Dimensionality

k-Nearest Neighbor (kNN) Algorithm

Fast kNN:

- kNN Computational complexity: $O(nd)$
- How to make it faster?
  - Dimensionality Reduction
    - Feature Selection (to be covered later)
    - PCA (to be covered)
  - Use efficient method to find nearest neighbors
    - KD Tree

k-Nearest Neighbor (kNN) Algorithm

K-D Tree:

- k-Dimensional tree
  - Extended version of binary search tree in higher dimension
- Pick the splitting dimension
  - Randomly
  - Large variance dimension
- Pick the middle value of the feature along the selected dimension after sorting along that dimension.
- Use this value as the root node and construct a binary tree and keep going.

k-Nearest Neighbor (kNN) Algorithm

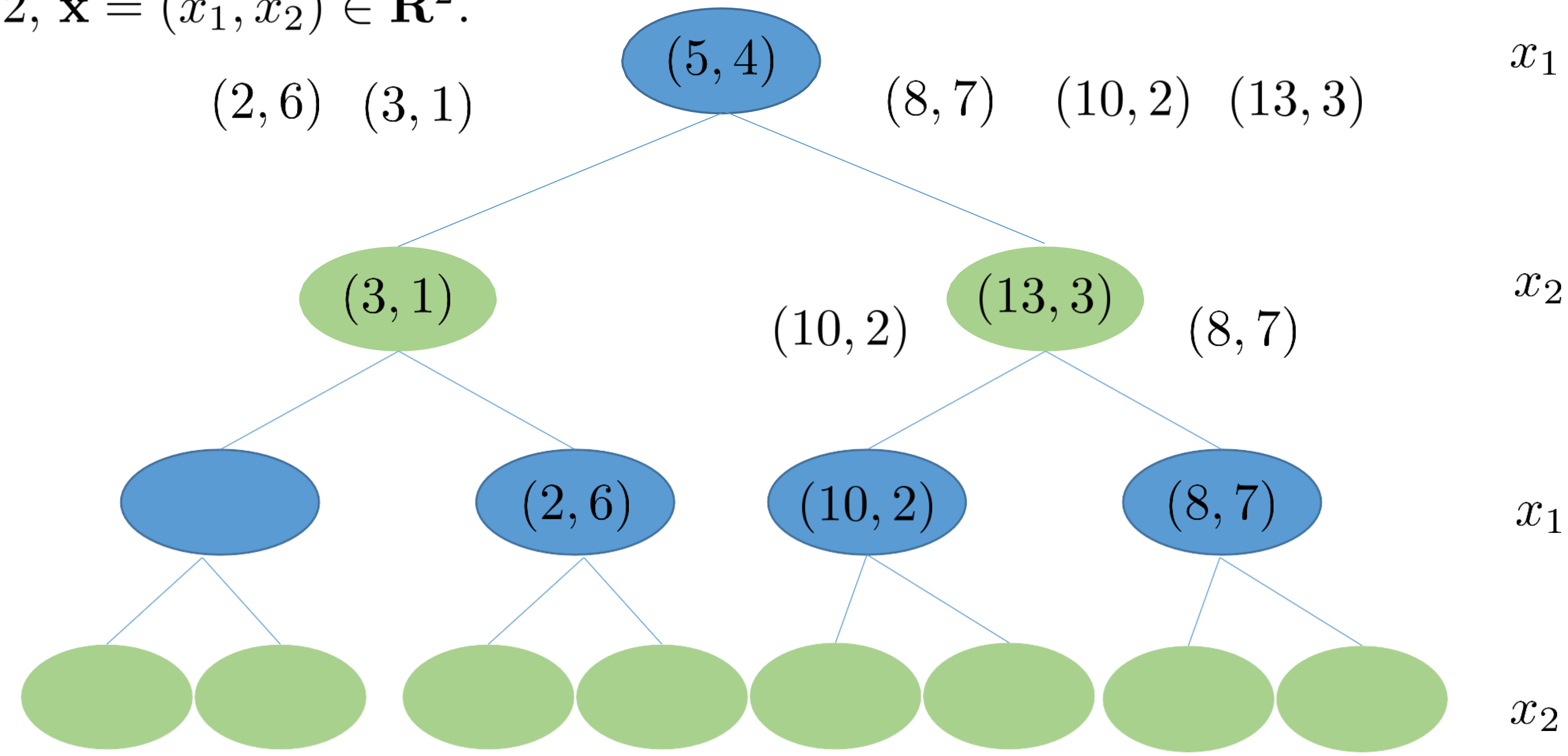
K-D Tree:

Example:

Splitting dimension

Consider $d = 2$, $\mathbf{x} = (x_1, x_2) \in \mathbf{R}^2$.

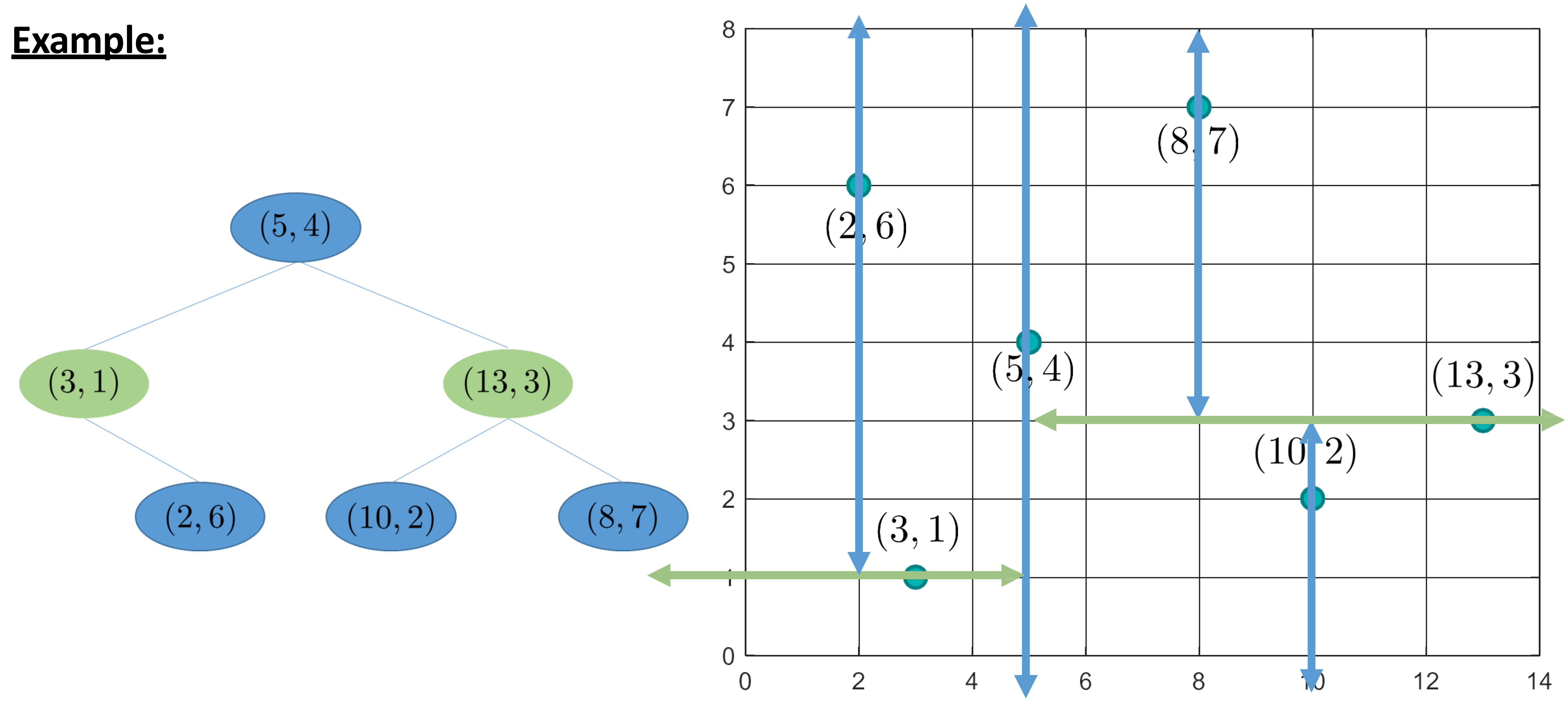
$(3, 1)$
$(5, 4)$
$(2, 6)$
$(10, 2)$
$(13, 3)$
$(8, 7)$



k-Nearest Neighbor (kNN) Algorithm

K-D Tree:

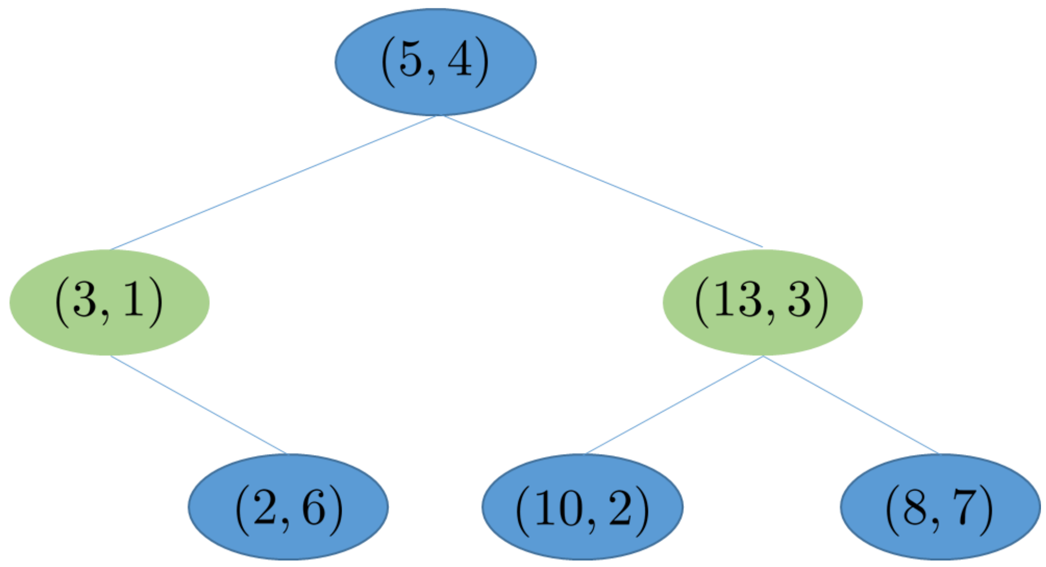
Example:

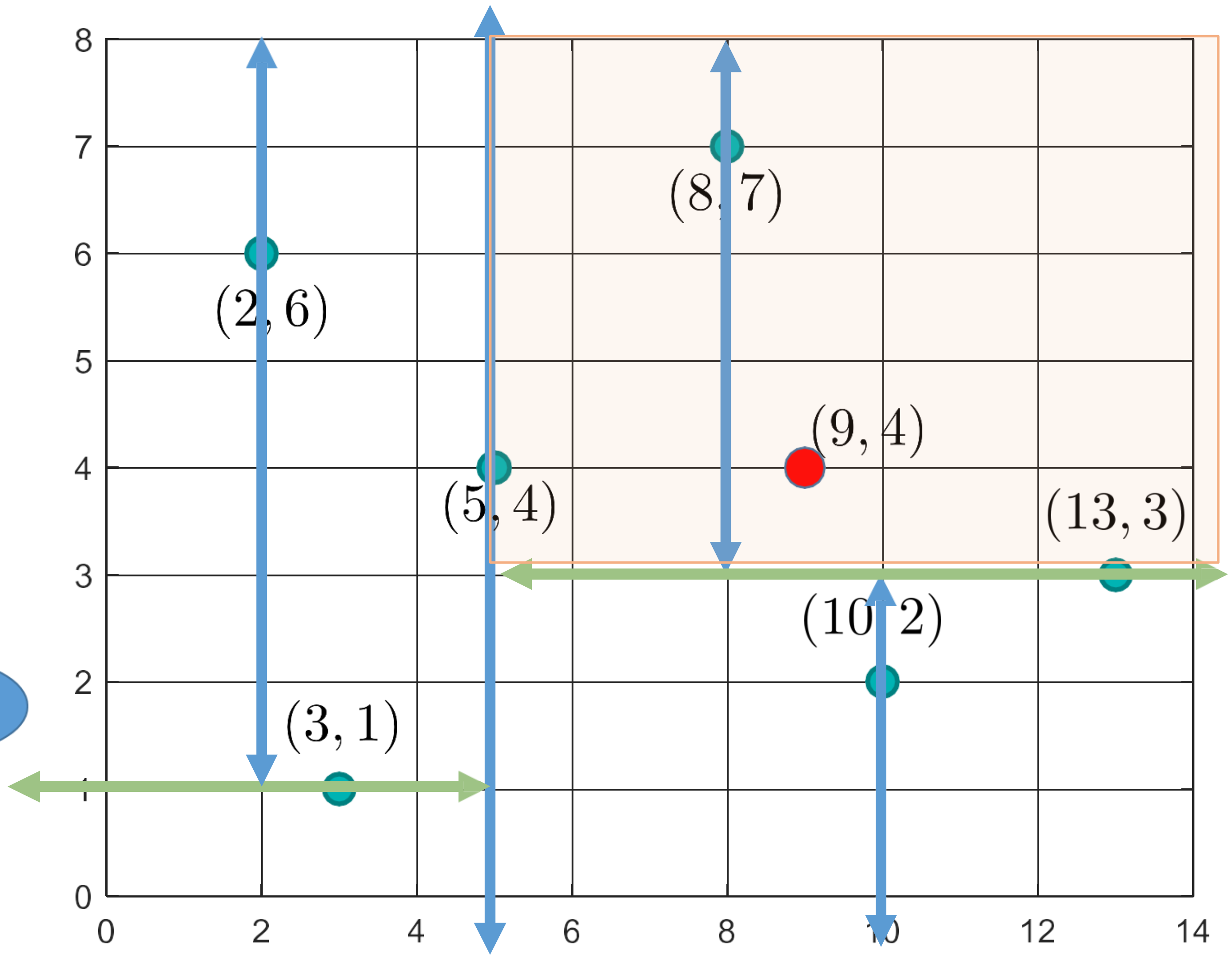


k-Nearest Neighbor (kNN) Algorithm

K-D Tree:

Connection with kNN:
Finding nearest neighbor





Issue: May miss neighbors! Trick to handle this.

k-Nearest Neighbor (kNN) Algorithm

K-D Tree - Summary:

- Enables **significant** reduction in the time complexity to support nearest neighbor algorithm.
  - Search to $O(\log n)$.

- Trade-offs:
  - Computational overhead to construct a tree $O(n \log n)$.
  - Space complexity: $O(n)$.
  - May miss neighbors.
  - Performance is degraded with the increase in the dimension of future space (**Curse of Dimensionality**).

Outline

- k-Nearest Neighbor (kNN) Algorithm Overview
- Algorithm Formulation
- Distance Metrics
- Choice of k
- Algorithm Convergence
- Storage, Time Complexity Analysis
- Fast kNN
- The Curse of Dimensionality

k-Nearest Neighbor (kNN) Algorithm

The Curse of Dimensionality:

- Refers to the problems or phenomena associated with classifying, analyzing and organizing the data in high-dimensional spaces that do not arise in low-dimensional settings.

- For high-dimensional datasets, the **size** of data **space** is **huge**.

- In other words, the size of the feature space grows exponentially with the number of dimensions (d) of the data sets.

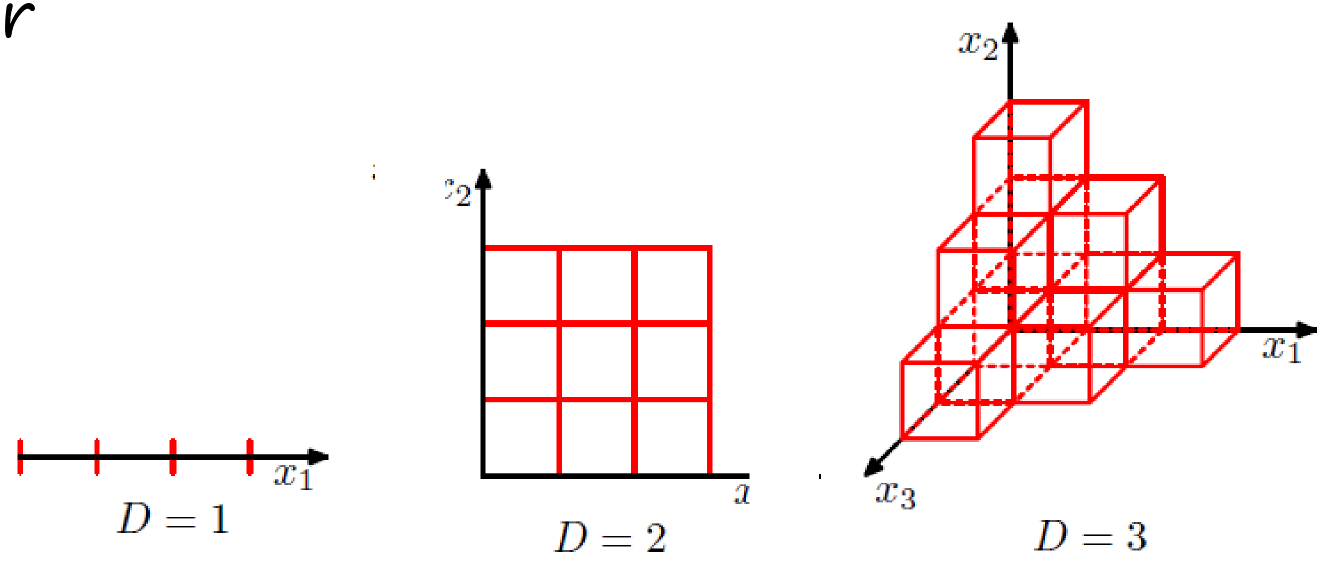
- To ensure the points stay close to each other, the size (n) of the data set must also have exponential growth. That means, we need a very large dataset to maintain the density of points in the high dimensional space.

k-Nearest Neighbor (kNN) Algorithm

The Curse of Dimensionality:

- For high-dimensional datasets, the **size** of data **space** is **huge**.

For an exponentially large number of cells, we need an exponentially large amount of training data to ensure that the cells are not empty.



Ref: CB

k-Nearest Neighbor (kNN) Algorithm

The Curse of Dimensionality:

Connection with kNN:

- With the increase in the number of features or number of dimensions of the feature space, data-points are never near to one another.

- kNN algorithm carries out predictions about the test point assuming we have data-points near to the test point that are similar to the test point.

- As we do not have neighbors in the high dimensional space, kNN becomes vulnerable and sensitive to the Curse of Dimensionality.

k-Nearest Neighbor (kNN) Algorithm

The Curse of Dimensionality: **Why does kNN work?**

Two related explanations:

- Real-world data in the higher dimensional space is confined to a region with effective lower dimensionality.
  - Dimensionality Reduction (to be covered later in the course)

- Real-world data exhibits smoothness that enables us to make predictions exploiting interpolation techniques.

- For example,
  - Data along a line or a plane in higher dimensional space
  - detection of orientation of object in an image; data lies on effectively 1 dimensional manifold in probably 1million dimensional space.
- Face recognition in an image (50 or 71 features).
  - Spam filter