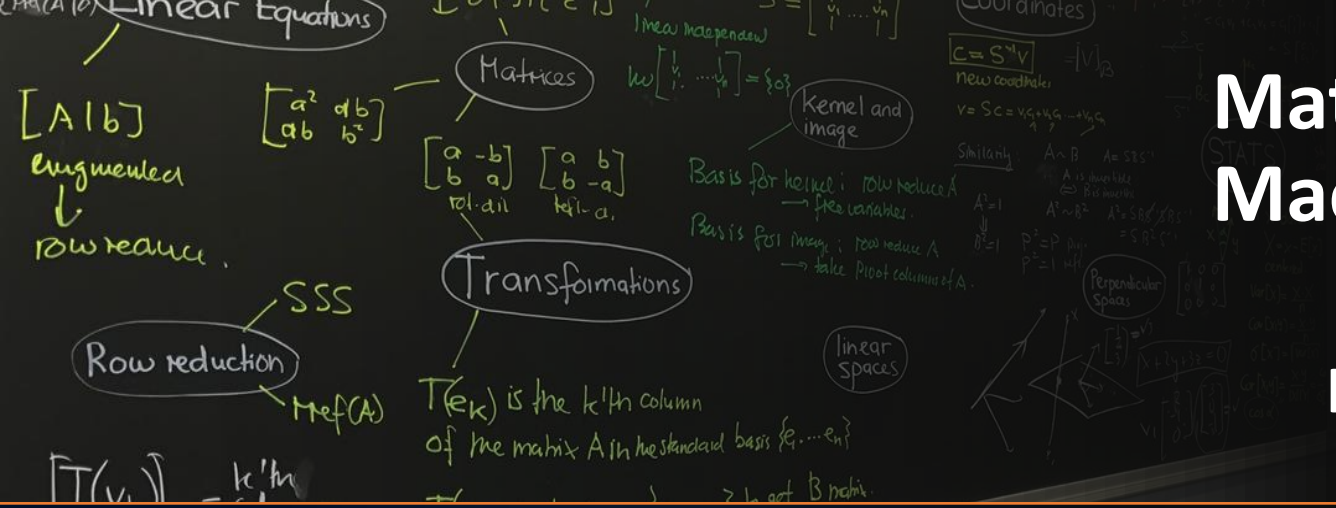
Mathematical Foundations for Machine Learning and Data Science

kNN Algorithm: Overview and Analysis



Dr. Zubair Khalid

Department of Electrical Engineering
School of Science and Engineering
Lahore University of Management Sciences

https://www.zubairkhalid.org/ee212_2021.html

LUMS
A Not-for-Profit University

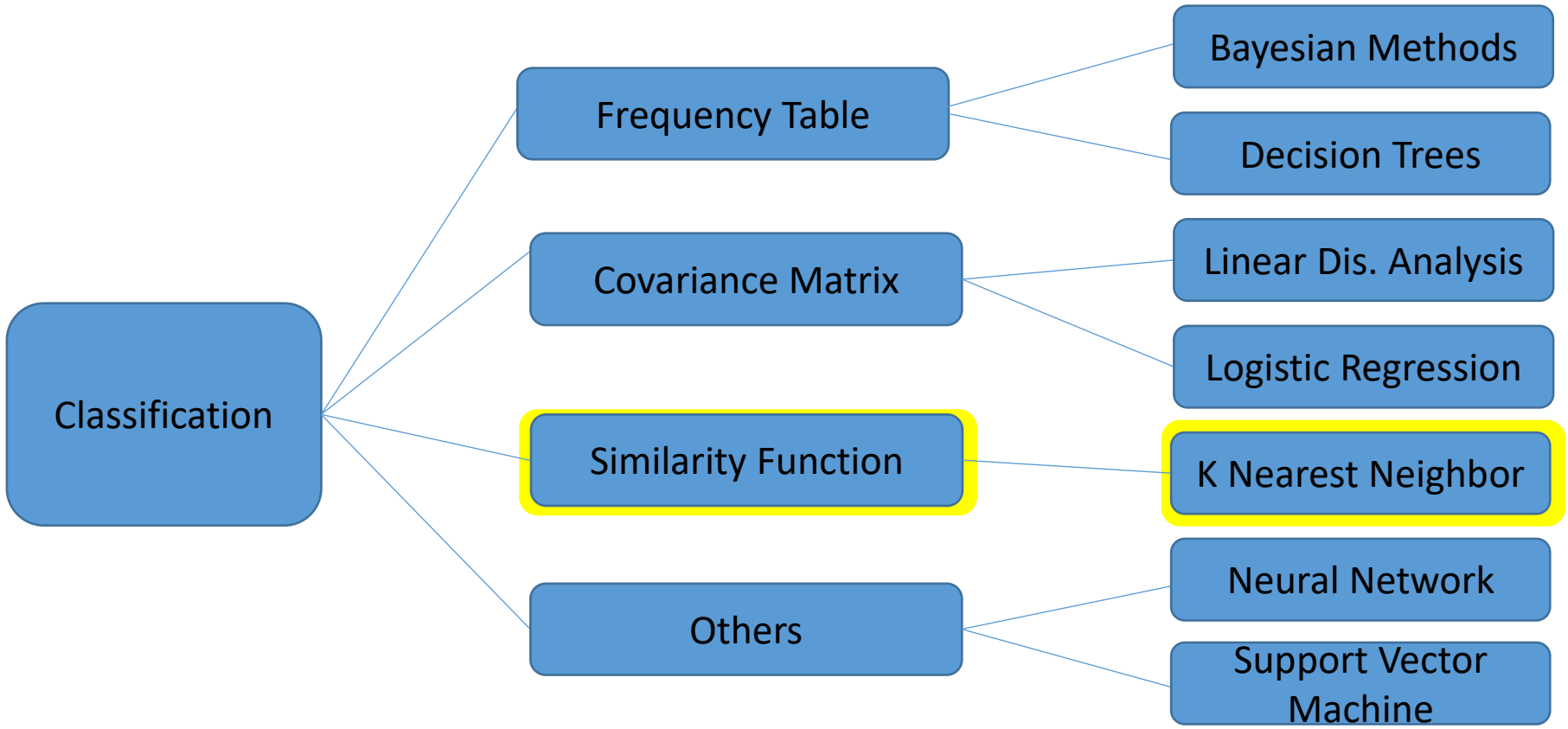
Outline

- k-Nearest Neighbor (kNN) Algroithm Overview
- Algorithm Formulation
- Distance Metrics
- Choice of k

Supervised Learning

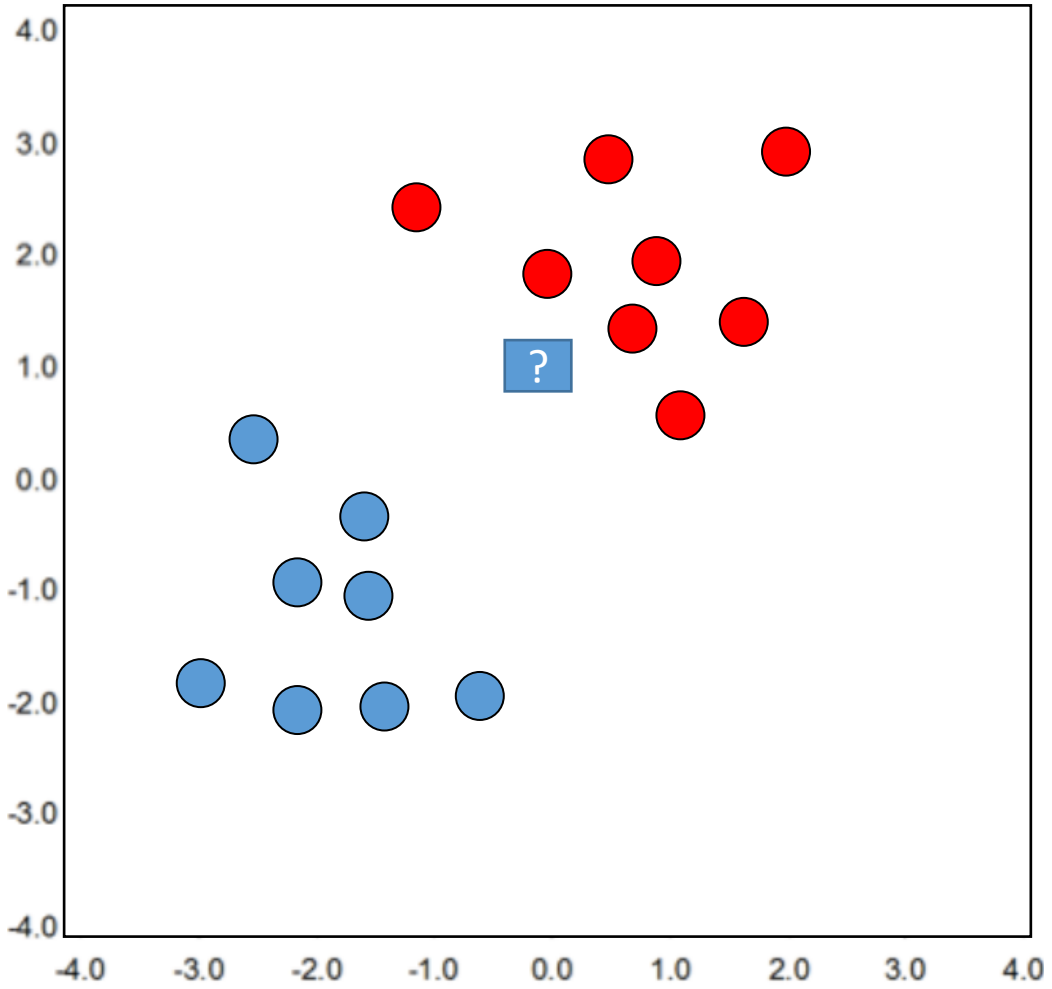
Classification Algorithms or Methods

Predicting a categorical output is called classification



Concept of Decision Boundary

Idea:

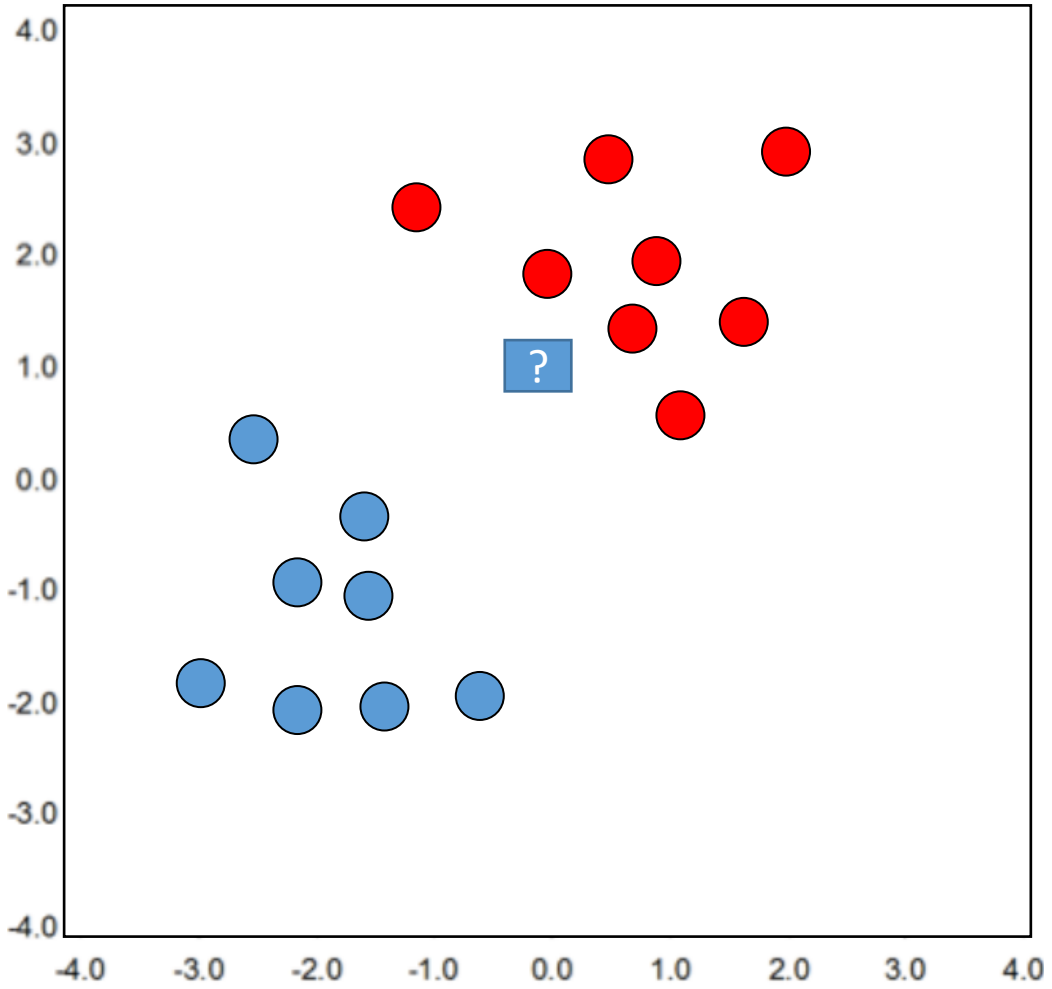


- Two classes, two features

- In classification, we learn a boundary that separates different classes.

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.

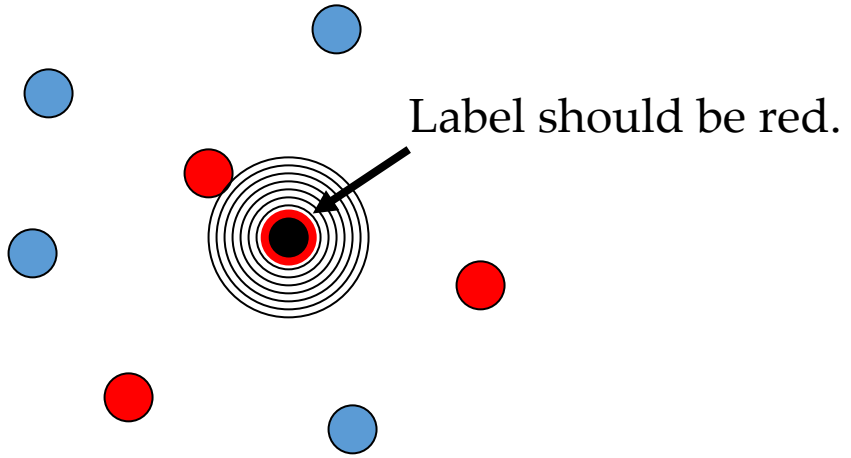
Questions:

- 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

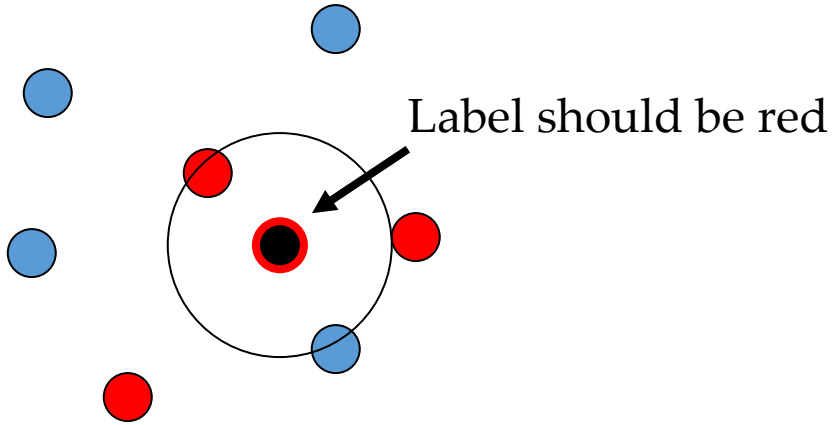
1-Nearest Neighbor:

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

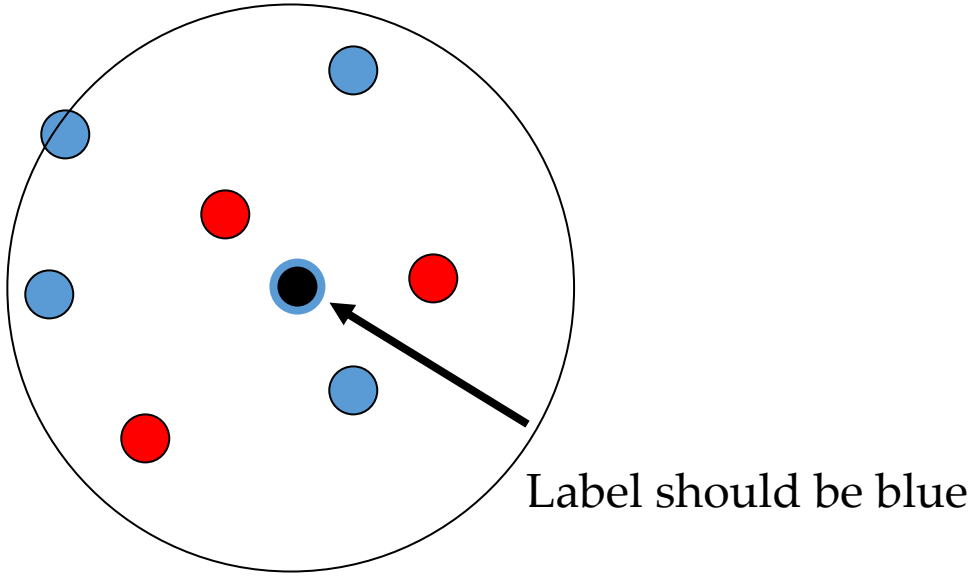


Generalization:

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)
- 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

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

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 $S_{\mathbf{x}}$

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

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

k-Nearest Neighbor (kNN) Algorithm

Decision Boundary:

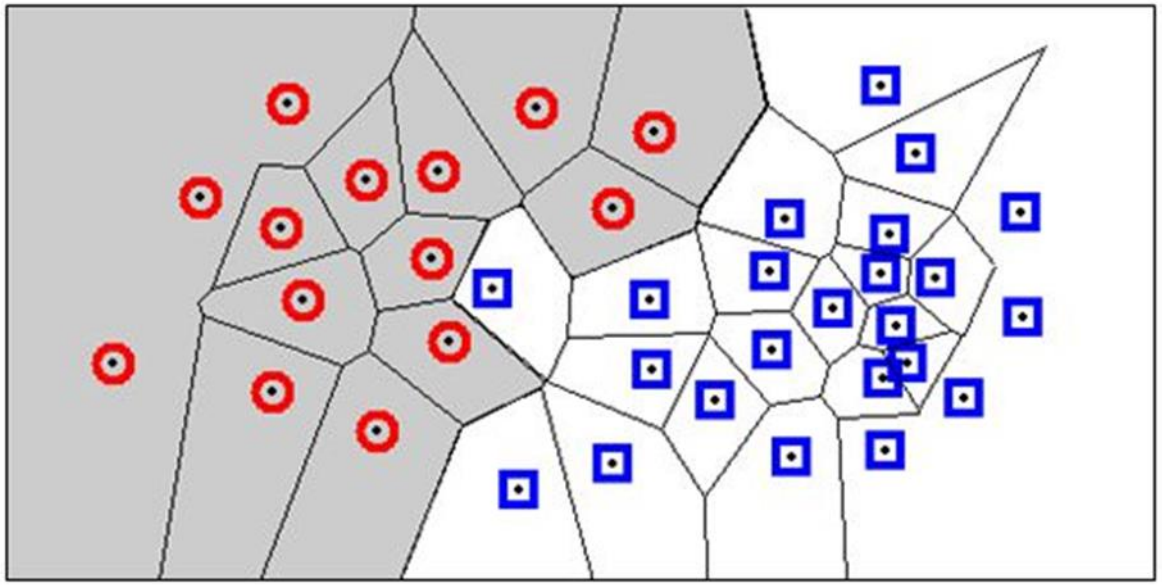
For $k = 1$, kNN defines a region, called decision boundary or region, in the space. Such division of the featue space is referred to s Voronoi partitioning.

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

Decision Boundary:

Demonstration

https://demonstrations.wolfram.com/KNearestNeighborKNNClassifier/

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.

Outline

- k-Nearest Neighbor (kNN) Algroithm Overview
- Algorithm Formulation
- **Distance Metrics**
- Choice of k

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

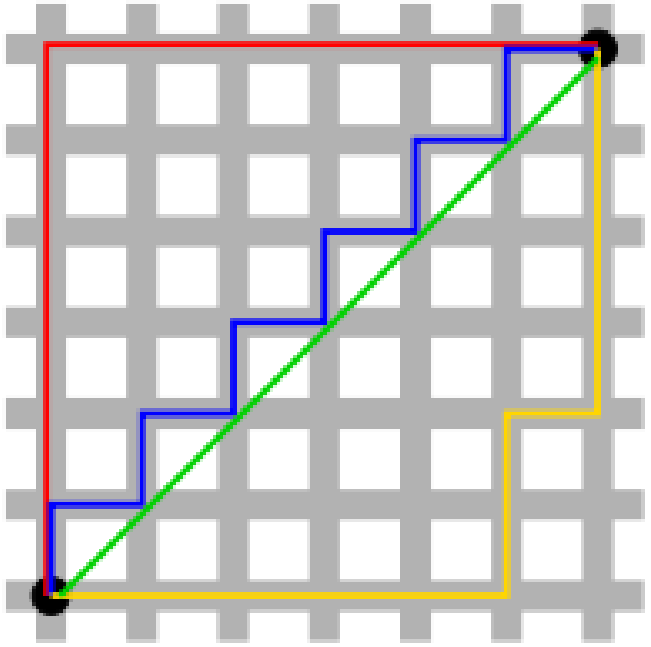
$$\text{dist}(\mathbf{x}, \mathbf{x}') \geq \max_{(\mathbf{x}'', y'') \in S_{\mathbf{x}}} \text{dist}(\mathbf{x}, \mathbf{x}''), \quad \forall (\mathbf{x}', y') \in D \backslash S_{\mathbf{x}}$$

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



Euclidean $6\sqrt{2}$

Manhattan
Manhattan 12
Manhattan

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

- Minkowski

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

$p = 2$

$p = 1$

$p = \infty$

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

k-Nearest Neighbor (kNN) Algorithm

Distance Metric:

Properties of Distance Metrics:

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

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

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

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

k-Nearest Neighbor (kNN) Algorithm

Distance Metric:

- For categorical vaiable, use Hamming Distance

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

k-Nearest Neighbor (kNN) Algorithm

Cosine Distance

- Cosine distance, though does not satisfy the properties we defined for distance metric, is however used to meaasure the angular distance between the vectors.
- It follows from the standard definition of inner (dot) product between the vectors, that is,

$$\mathbf{x}^T\mathbf{x}' = \|\mathbf{x}\|_2\|\mathbf{x}'\|_2 \cos\theta$$

or

$$\cos\theta = \frac{\mathbf{x}^T\mathbf{x}'}{\|\mathbf{x}\|_2\|\mathbf{x}'\|_2}$$

What is the range of values of angular distance and what is the interpretation of these values?

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) Algroithm Overview
- Algorithm Formulation
- Distance Metrics
- **Choice of k**

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.