CSE 4553
Machine Learning
Lecture 11: KNN

Winter 2022

Hasan Mahmud | hasan@iut-dhaka.edu

Contents

- Introduction
- A simple example to understand the intuition behind KNN
- How does the KNN algorithm work?
- Methods of calculating distance between points
- How to choose the k factor?

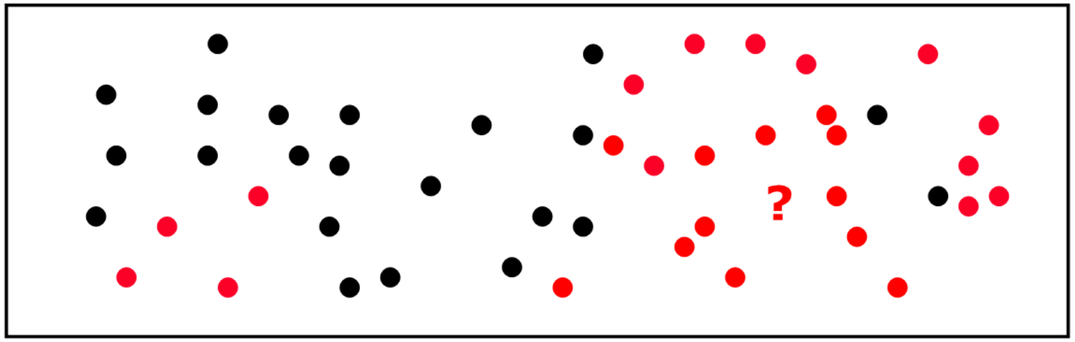
Introduction

- The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.
- The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other, which called “feature similarity”.

KNN

- At test time, find closest example in training set, and return corresponding label

$$\hat{y}(x) = y_{n^*} \text{ where } n^* = \arg\min_{n \in D} dist(x, x_n)$$

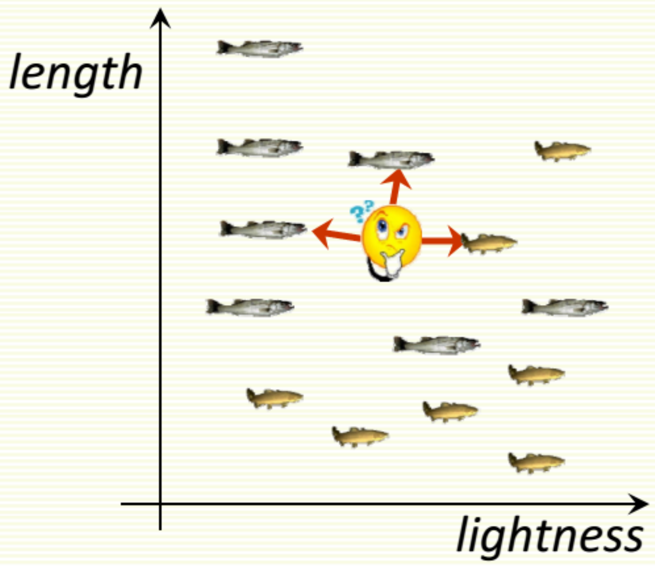


- We can find the K nearest neighbors, and return the majority vote of their labels

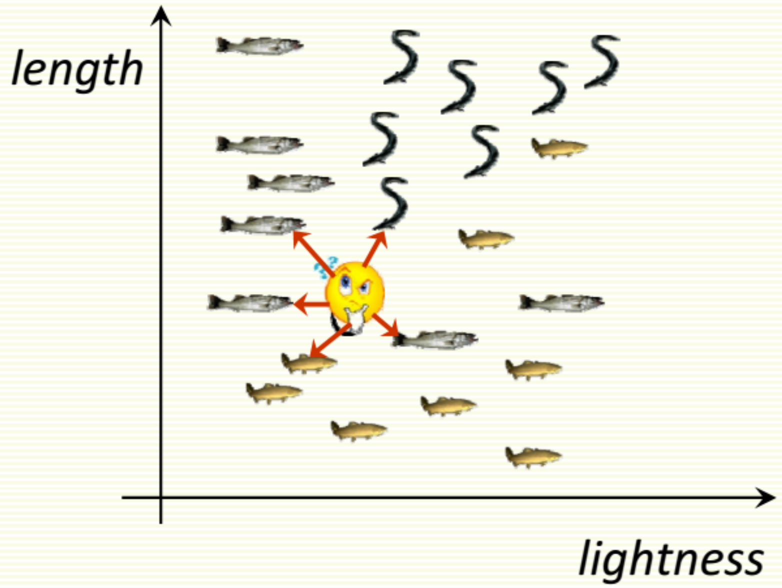
  Eg y(X1) = x, y(X2) = o

- classify an unknown example with the most common class among ***k*** closest examples
  - *"tell me who your neighbors are, and I'll tell you who you are"*

- Example:
  - ***k*** = 3
  - 2 sea bass, 1 salmon
  - Classify as sea bass



- Easy to implement for multiple classes
- Example for ***k*** = 5
  - 3 fish species: salmon, sea bass, eel
  - 3 sea bass, 1 eel, 1 salmon ⇒ classify as sea bass



The KNN Algorithm

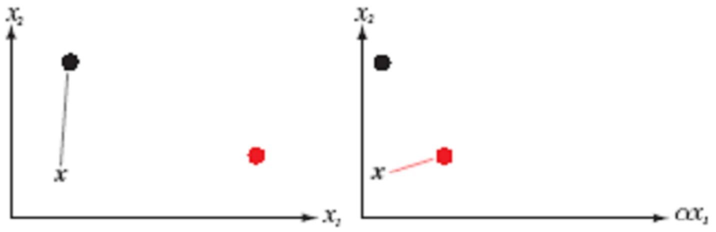
1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data
    1. Calculate the distance between the query example and the current example from the data.
    2. Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

Similarity measure

- Euclidean distance
  - For real-valued feature vectors, we can use Euclidean distance

$$D(u,v)^2 = ||u-v||^2 = (u-v)^T(u-v) = \sum_{i=1}^{d}(u_i - v_i)^2$$

  - If we scale x1 by 1/3, NN changes!



Similarity measure

- Manhattan distance

$$d(x, y) = \sum_{i=1}^{d} |x_i - y_i|$$

- Hamming distance

$$D_H = \sum_{i=1}^{k} \left| x_i - y_i \right|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

Similarity measure

- Mahalanobis distance

  - Mahalanobis distance lets us put different weights on different comparisons

$$\begin{aligned} D(u,v)^2 &= (u-v)^T\Sigma(u-v) \\ &= \sum_i \sum_j (u_i - v_i)\Sigma_{ij}(u_j - v_j) \end{aligned}$$

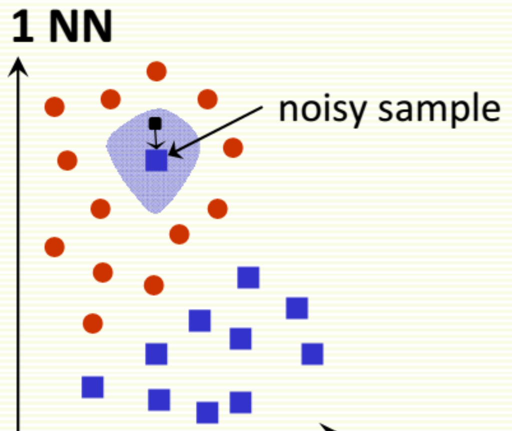
  where Σ is a symmetric positive definite matrix

  - Euclidean distance is Σ=I

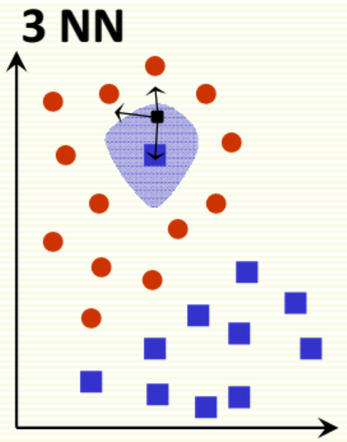
How to choose K?

- In theory, if infinite number of samples available, the larger is ***k***, the better is classification
- But the caveat is that all ***k*** neighbors have to be close
  - Possible when infinite # samples available
  - Impossible in practice since # samples is finite

- Rule of thumb is ***k*** = sqrt(***n***), ***n*** is number of examples
  - interesting theoretical properties
- In practice, ***k*** = 1 is often used for efficiency, but can be sensitive to "noise"

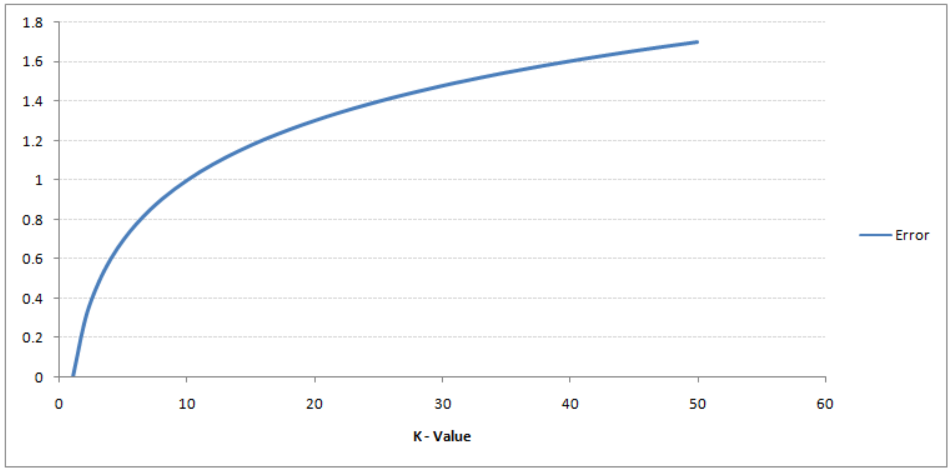


every example in the blue shaded area will be misclassified as the blue class

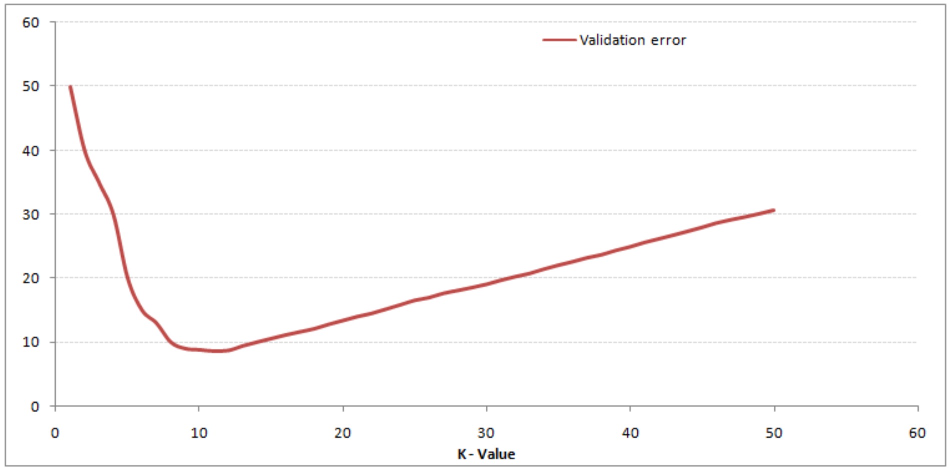


every example in the blue shaded area will be classified correctly as the red class

- Training Error



- Validation Error


Validation error
K - Value