Instance-based Learning

January 19, 2025

Instance-based learning

- Instance-based learning (IBL) is a type of ML that learns from specific instances or examples in the training data rather than abstracting from them to create a general model.
- IBL relies on storing the training instances and making predictions based on these stored examples.
- Some characteristics of IBL:
  - Lazy learning: It does not construct a general model during the training phase.
  - Local Decision Making: Predictions are made based on the specific instances closest to the query instance.
  - Memory-Based: The learning process involves storing training examples in memory.
- The most well-known instance-based method is KNN
- Similarity measure is required for comparison and is key in instance-based learning

Instance-based learning

- **Advantages:**
  - Adaptability: IBL methods can adapt to changes in the data without retraining a global model.
  - Simplicity: These methods are often straightforward to implement and understand.
- **Disadvantages:**
  - Storage Requirements: Storing all instances can be memory-intensive, especially for large datasets.
  - Computational Complexity: Prediction can be computationally expensive because it involves comparing the query instance to many stored instances.
  - Generalization issue: IBL methods do not generalize well to unseen data because its predictions are based on memorized examples rather than learned models

Some similarity measures

In the following, $x, y \in R^d$ and A, B are set.

- **Manhattan Distance**

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

- **Euclidean Distance**

$$d(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

- **Cosine Distance**

$$\cos(x, y) = \frac{\vec{x}.\vec{y}}{\|\vec{x}\|.\|\vec{y}\|}$$

- **Jaccard Distance**

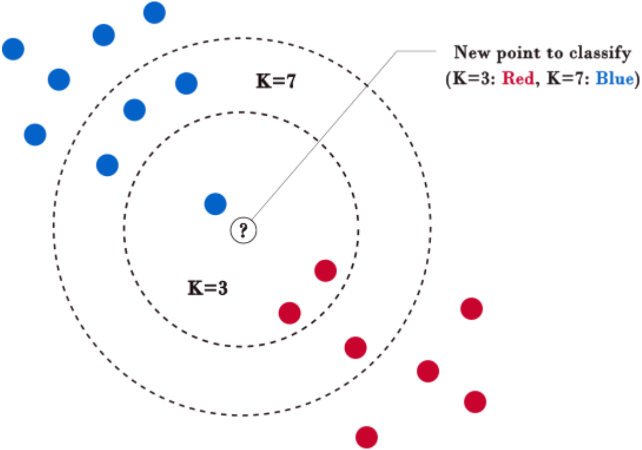
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- **Hamming Distance**

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

$$x, y \in \{0, 1\}^d,\ I(x_i, y_i) = 1 \text{ if } x_i = y_i,\ 0 \text{ otherwise.}$$

K-nearest neighbors

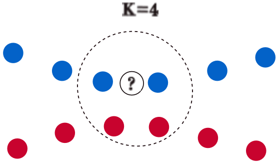


K-nearest neighbors

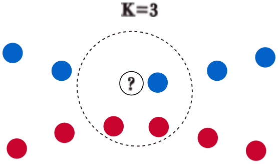
- **Input:**
  - $D = \{(x_i, y_i) \mid x_i \in R^d, y_i \in \{1, 2 \ldots, C\}\}, |D| = n$
  - K: # of considered neighbors
  - d: a distance measure
  - $\hat{x} \in R^n$: a new instance to predict
- **Output:**
  - $\hat{y}$: predicted label for $\hat{x}$
- **KNN**
  1. compute $d(\hat{x}, x_i), \ \forall x_i \in D$
  2. form $\mathbf{N} \subseteq D$ s.t
     - $|\mathbf{N}| = K$
     - $d(\hat{x}, x_i) \leq d(\hat{x}, x_j) \mid \forall x_i \in \mathbf{N} \ \& \ \forall x_j \in (D - \mathbf{N})$
  3. $\hat{y} = \text{argmax}_c\{\sum_{(x_i, y_i) \in N} I(y_i = c)\}$
- **Complexity:** $O(n.d + n.k)$
- **K:** Small K $\rightarrow$ overfitting; big K $\rightarrow$ underfitting

Problems with KNN

- KNN shares the same disadvantages with instance-based learning methods
- Tie issue: The model with randomly decide when multiple classes have the same number of nearest neighbors among the K closest instances.
- Neighbors with different distances are equally weighted.

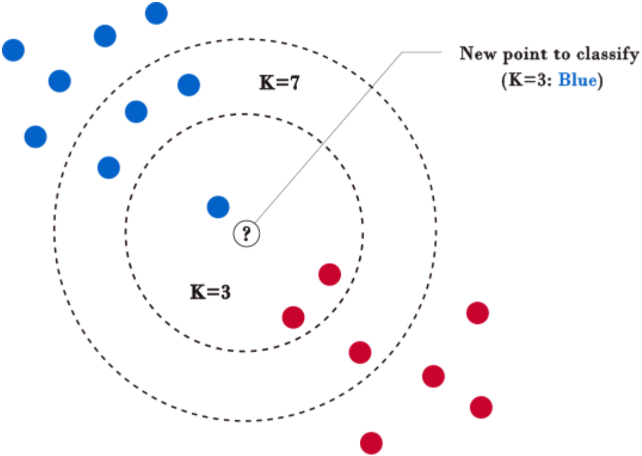


Tie issue



Equal weight issue

Weighted KNN



Weighted KNN

- **Input:**
  - $D = \{(x_i, y_i) \mid x_i \in R^d, y_i \in \{1, 2 \ldots, C\}\}, |D| = n$
  - K: # of considered neighbors
  - d: a distance measure
  - w: a weighted measure
  - $\hat{x} \in R^n$: a new instance to predict
- **Output:**
  - $\hat{y}$: predicted label for $\hat{x}$
- **Weighted KNN**
  1. compute $d(\hat{x}, x_i), \ \forall x_i \in D$
  2. form $\mathbf{N} \subseteq D$ s.t
     - $|\mathbf{N}| = K$
     - $d(\hat{x}, x_i) \leq d(\hat{x}, x_j) \mid \forall x_i \in \mathbf{N} \ \& \ \forall x_j \in (D - \mathbf{N})$
  3. $\hat{y} = \text{argmax}_c\{\sum_{(x_i, y_i) \in N} w(\hat{x}, x_i) \times I(y_i = c)\}$
- **Weighted measures**
  - Inverse of distance measure
  - Rank of instances in the neighbor set