Review: K-Nearest Neighbors (KNN)

❖ Summary: Pipeline of K-Nearest Neighbor Regression/Classification

1. Prepare Training Data
(Features & Labels)

2. Configure value K
k-NN(k = K)

- ✓ Decide proper distance function
- ✓ K can be different values from 1 to N (training size)
- ✓ Optimal K should be determined by cross-validation

3. Training K-NN model
k-NN.fit(Features, Labels)

- ✓ Just store the entire training dataset
- ✓ Build indices for all training points
- ✓ Fast in training time(advantage), but greater storage requirement

4. Testing K-NN model
k-NN.predict(Features)

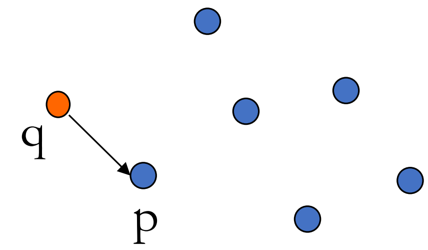
Instance-based similarity comparison

- ✓ Find the top-K training examples $(x_1, y_1),\ldots,(x_k, y_k)$ that are closest to test data
- ✓ For **classification**, use majority voting among neighbors' labels
- ✓ For **regression**, average neighbors' labels for each query data
- ✓ Prediction is slow, high memory cost(disadvantage)

5. Evaluate K-NN model
k-NN.evaluate(Features, Labels)

- ✓ **Regression**: Mean-Squared Error, etc.
- ✓ **Classification**: Accuracy, etc.

Review: K-Nearest Neighbors for Regression (KNN)



❖ **Algorithm Steps:**

(1) Calculate distance $d(q, p_i)$ between test point q and all training points $\{p_i\}$ ($i \in [1, \mathrm{N}]$)

(2) Sort training points $\{p_i\}$ by distance $d(q, p_i)$ ascendingly, ($i \in [1, \mathrm{N}]$)

(3) Select top K neighbors from training samples

(4) Calculate the average from the labels of top K nearest neighbors, defined by

$$\hat{y}_q = \frac{1}{K}\sum_{i=1}^{K} y_i$$

y_i is continuous label of neighbor i

Review: K-Nearest Neighbors for Classification (KNN)

❖ **Algorithm Steps:**

(1) Calculate distance $d(q, p_i)$ between test point q and all training points $\{p_i\}$ ($i \in [1, \mathrm{N}]$)

(2) Sort training points $\{p_i\}$ by distance $d(q, p_i)$ ascendingly, ($i \in [1, \mathrm{N}]$)

(3) Select top K neighbors from training samples

(4) Derive the Majority class (voting) from the labels of top K nearest neighbors, defined by

$$\hat{y}_q = mode(\{y' : (x', y') \in S_x\})$$

- S_x is set of the top k nearest neighbors of test point x
- y' is categorical label of a neighbor
- mode(·) means to select the label of the highest occurrence.

This class: Distance weighting and Computational Efficiency

Question 1: Should all features be treated equally important in distance calculation?

✓ **Euclidean distance** between vectors $\boldsymbol{x_A}$ and $\boldsymbol{x_B}$ **is:**

$$d(\boldsymbol{x_A}, \boldsymbol{x_B}) = \sqrt{\sum_{i=1}^{d}(a_i - b_i)^2} = \sqrt{\underbrace{(a_1 - b_1)^2}_{\text{Feature 1}} + \underbrace{(a_2 - b_2)^2}_{\text{Feature 2}} + \cdots + \underbrace{(a_d - b_d)^2}_{\text{Feature d}}}$$

Any modification can be applied?

Question 2: How distance function can be applied to both numeric and categorical attributes?

Question 3: When we average labels of neighbors for prediction, should all K closest neighbors be treated equally important regardless of distance values?

$$\hat{y} = \frac{1}{K}\sum_{i=1}^{K} y_i = \frac{1}{K} * (\underbrace{y_{nb-1}}_{\text{Neighbor-1}} + \underbrace{y_{nb-2}}_{\text{Neighbor-2}} + \cdots + \underbrace{y_{nb-K}}_{\text{Neighbor-K}})$$

Any modification can be applied?

Question 4: What distance function is best for the data (Euclidean, Cosine, Correlation);

Question 5: How different hyper-parameters affect performance under different applications?

Question 6: How to improve Neighbor Search Efficiency (i.e., $O(Nd)$) using Parallel Computing, GPU acceleration; Kd-Trees

Pre-processing: Feature Scaling and Transformation

Question 1: Should all features be treated equally important in distance calculation?

✓ **Euclidean distance** between vectors $\boldsymbol{x_A}$ and $\boldsymbol{x_B}$ **is:**

$$d(\boldsymbol{x_A}, \boldsymbol{x_B}) = \sqrt{\sum_{i=1}^{d}(a_i - b_i)^2} = \sqrt{\underbrace{(a_1 - b_1)^2}_{\text{Feature 1}} + \underbrace{(a_2 - b_2)^2}_{\text{Feature 2}} + \cdots + \underbrace{(a_d - b_d)^2}_{\text{Feature d}}}$$

Any modification can be applied?

A more general form

$$d(\boldsymbol{x_A}, \boldsymbol{x_B}) = \sqrt{\sum_{i=1}^{d} w_i \delta(\boldsymbol{x_{A,i}}, \boldsymbol{x_{B,i}})}$$

w_i: the weight for each feature

$\delta(\boldsymbol{x_{A,i}}, \boldsymbol{x_{B,i}})$: distance metrics between i-th feature value in vectors $\boldsymbol{x_A}$ and $\boldsymbol{x_B}$

Assumption of Naïve Euclidean Distance:

- All dimensions are treated equally: $w_i = 1$ for all features
- Straight-line distance between two points, $\delta(\boldsymbol{x_{A,i}}, \boldsymbol{x_{B,i}}) = (\boldsymbol{x_{A,i}} - \boldsymbol{x_{B,i}})^2$
- Have the same units/scales of measurement, i.e., $x \in [0,1]$