- Retrieval as K-nearest neighbor search

--- 1-NN search for retrieval

-- The first step is we need to compute the distance between this query article and all other articles.

-- 1-Nearest Neighbor

Input: Query article - x_q

Corpus of documents - $x_1, x_2, \dots, x_N$

Output: Most similar article

-- Formally, $x^{NN} = min_{x_i} distance(x_q, x_i)$

- 1-NN algorithm

--- Initialize $Dist2NN = \infty, \{closest\ document\} = \emptyset$

For i = 1, 2, …, N:

Compute: $\delta = distance(x_i, x_q)$

If $\delta < Dist2NN$:

Set $\{closest\ document\} = x_i$

Set $Dist2NN = \delta$

Return $most\ similar\ document$ $\{closest\ document\}$

- K-Nearest neighbor

--- Input: Query article - x_q

Corpus of documents: $x_1, x_2, \dots, x_N$

--- Output: List of k similar articles.

--- Formally, $x^{NN} = \{x^{NN_1}, \dots, x^{NN_K}\}$

For all x_i, not in x^{NN},

$Distance(x_i, x_q) \geq \max distance(x^{NN_q}, x_q)$

- K-NN algorithm

--- Initialize $Dist2KNN = sort(\delta_1, \dots, \delta_k),\ \ corpus = \ sort(doc_1, \dots, doc_k)$

For I = k+1, …, N:

Compute: $\delta = distance(doc_1, doc_q)$

If $\delta < dist2KNN[k]$:

Find j such that $\delta > Dist2kNN[j-1]$ but $\delta < Dist2kNN[j]$

Remove furthest article and shift queue:

$corpus[j+1{:}\,k] = corpus[j{:}\,k-1]$

$$Dist2kNN[j+1:k] = Dist2kNN[j:k-1]$$

Set $Dist2kNN[j] = \delta$ and $corpus[j] = doc$

Return k most similar articles.

- Critical elements of NN search

--- Item (e.g., doc) representation: x_q

--- Measure of distance between items: $\delta = distance(x_i, x_q)$.

- Word count document representation

--- Bag of words model: 1) Ignore order of words; 2) Count # of instances of each word in vocabulary.

- Issues with word counts – Rare words

- TF-IDF document representation

--- Emphasizes important words

-- Appears frequently in document (common locally);

-- Appears rarely in corpus (rare globally).

$$Term\ Frequency = word\ counts$$

$$Inverse\ doc\ freq = log \frac{\#docs}{1 + \#(docs\ using\ word)}$$

$$TF - IDF = Term\ frequency * Inverse\ doc\ freq$$

- Distance metrics: Defining notion of “closest”

--- In 1D, just Euclidean distance:

$$distance(x_i, x_q) = |x_i - x_q|$$

--- In multiple dimensions:

-- Can define many interesting distance functions;

-- most straightforwardly, might want to weight different dimensions differently;

-- Reasons: 1) Some features are more relevant than others; 2) some features vary more than others.

-- Specify weights:

For feature j:

$$\frac{1}{max_i(x_i[j]) - min_i(x_i[j])}$$

- Scaled Euclidean distance:

$$Distance(x_i, x_q) = \sqrt{a_1(x_i[1] - x_q[1])^2 + \cdots + a_d(x_i[d] - x_q[d])^2}$$
$$= \sqrt{(x_i - x_q)^T A (x_i - x_q)}$$

- (non-scaled) Euclidean distance

--- Defined in terms of inner product:

$$Distance(x_i, x_q) = \sqrt{(x_i - x_q)^T (x_i - x_q)} = \sqrt{(x_i[1] - x_q[1])^2 + \cdots + (x_i[d] - x_q[d])^2}$$

- Another natural inner product measure:

$$Similarity = x_i^T x_q = \sum_{j=1}^{d} x_i[j] x_q[j]$$

- Cosine similarity – normalize

$$similarity = \frac{\sum_{j=1}^{d} x_i[j] x_q[j]}{\sqrt{\sum_{j=1}^{d} (x_i[j])^2} \sqrt{\sum_{j=1}^{d} (x_q[j])^2}} = \frac{x_i^T x_q}{\|x_i\| \|x_q\|} = \cos\theta$$
$$distance = 1 - similarity$$

- Normalizing can make dissimilar objects appear more similar.