3 Fundamental Algorithms (part 5 of 5)

k-Nearest Neighbors (kNN)

Training in k-NN consists only of storing the training data.

- **k-NN classification** assigns to a new \( \mathbf{x} \) the ________ label among its \( k \) nearest neighbors.
- **k-NN regression** assigns to a new \( \mathbf{x} \) the ________ numeric value of its \( k \) nearest neighbors.

**Distance metrics**

Euclidean distance is the typical metric. Others may be worth exploring. For two vectors \( \mathbf{a} \) and \( \mathbf{b} \),

- **Euclidean** distance(\( \mathbf{a}, \mathbf{b} \)) = \( \sqrt{\sum_{i=1}^{D} (a^{(i)} - b^{(i)})^2} \) (also called the 2-norm)
  
e.g. This is the familiar length of the line segment joining two points.
- **Manhattan** distance(\( \mathbf{a}, \mathbf{b} \)) = \( \sum_{i=1}^{D} |a^{(i)} - b^{(i)}| \) (also called the 1-norm or absolute-value norm or taxicab distance)
  
e.g. This is the ________ in blocks between two points on a square grid street layout.
- **Minkowski** distance(\( \mathbf{a}, \mathbf{b} \)) = \( \left( \sum_{i=1}^{D} |a^{(i)} - b^{(i)}|^{p} \right)^{\frac{1}{p}} \) for integer \( p \). \( p = \) _______: Manhattan; \( p = \) _______: Euclidean; \( p = \) _______: \( \max(\{|a_i - b_i|\}) \); \( p = \) _______: \( \min(\{|a_i - b_i|\}) \).
- **Negative cosine similarity** \( ^1 \) distance(\( \mathbf{a}, \mathbf{b} \)) = \( -\cos \angle(\mathbf{a}, \mathbf{b}) = -\frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}|| \cdot ||\mathbf{b}||} = -\frac{\sum_{i=1}^{D} a^{(i)} b^{(i)}}{\sqrt{\sum_{i=1}^{D} (a^{(i)})^2} \sqrt{\sum_{i=1}^{D} (b^{(i)})^2}} \)
  
e.g. In information retrieval, give each word in an \( n \)-word dictionary its own coordinate in \( n \)D space. Make a vector of word counts for each document. Then \( \cos \angle(\mathbf{a}, \mathbf{b}) \) for documents \( \mathbf{a} \) and \( \mathbf{b} \) measures ________ without regard for document length, so \( -\cos \angle(\mathbf{a}, \mathbf{b}) \) measures topical “distance”.
- **Hamming** distance(\( \mathbf{a}, \mathbf{b} \)), for two equal-length strings \( \mathbf{a} \) and \( \mathbf{b} \), is the is the ________ at which the corresponding characters are different.
  
e.g. Hamming distance could be used in a spell-checker.
  
e.g. In information theory, Hamming distance is the minimum number of errors that could account for transmitting bit sequence \( \mathbf{a} \) but receiving \( \mathbf{b} \).

Both \( k \) and the choice of metric are hyperparameters set before running k-NN.

---

\(^1\)Replacing \( \mathbf{a} \) and \( \mathbf{b} \) with \( \mathbf{a} - \bar{a} \) and \( \mathbf{b} - \bar{b} \) yields the **negative centered cosine similarity**, which is equivalent to \( -r \).
Normalization

Normalization or rescaling is necessary if the features have different units or scales (coming in §5).

E.g. Consider the distance of points from (0, 0). If \( x \) and \( y \) are each uniformly distributed, \( x \) in [0, 1] and \( y \) in [0, 0.01] (draw _ ), then the \( y \) values would hardly influence the distance. They could matter if we rescaled them (e.g. by \( y \mapsto y \times 100 \) so that they would span [0, 1]).

Weighted \( k \)-NN

E.g. Weighting each of a new point’s \( k \) nearest neighbors by ______, where \( d_i \) is the distance to the \( i \)th nearest neighbor, would make near neighbors ________________________ than distant ones.

Python

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import KNeighborsRegressor

clf = KNeighborsClassifier(n_neighbors=5, weights='uniform', p=2, metric='minkowski')
The default \( p=2, \) metric='minkowski' give Euclidean distance. metric options include 'euclidean', 'manhattan', and 'minkowski' for vectors in \( \mathbb{R}^D \); 'hamming' for vectors in \( \mathbb{Z}^D \); and a user-defined function that takes two 1D NumPy arrays and returns a distance.

weights='uniform' gives unweighted \( k \)-NN. weights='distance' uses weights \( \left\{ \frac{1}{d_i} \right\} \) (above).

clf = KNeighborsRegressor(n_neighbors=5)
clf.fit(X, y) fits the model to array \( X \times_D \) and \( y \times_1 \)
clf.predict(X) does classification or regression on examples in \( X \)

classification:
- clfr.predict_proba(X)[:, 1] gives \( P(y_i = 1) \) \( [:, 0] \) gives \( P(y_i = 0) \)
- clfr.score(X, y) gives the average accuracy on \( X \) with respect to \( y \)

regression: clfr.score(X, y) gives \( R^2 \), the coefficient of determination
```

To learn more:

- Reference manual: