Programming for AI

Cindi Thompson[1]

[1]University of San Francisco
Email: cathompson4@usfca.edu

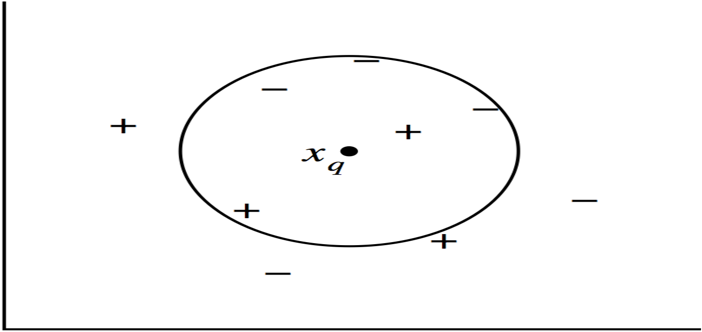
Instance-based Learning

Instance-based Learning

Still in supervised learning scenario:

- Training Examples: $\langle x_i, y_i \rangle$ (assume discrete Y for now)
- How to label a new example x_q?
- Key idea of IBL: store all training examples, compute "hypothesis" at classification time
- Also called lazy, memory-based, exemplar-based, and non-generalizing
- Types:
  - k-Nearest Neighbor
  - Locally weighted regression
  - Radial basis functions
  - Case-based Reasoning

5-NN Example



Overview of KNN classification

- Given a query point x_q, K-nearest neighbor (KNN) classification finds the k training points closest in distance to x_q, and then classifies using majority vote[1] among the k neighbors.
- The nearest-neighbor method goes back at least to Fix and Hodges (1951).
- These classifiers are memory-based, and require no model to be fit.

[1] Bases the assignment of a label on the predominance of a particular class in this neighborhood.

Overview of KNN classification

There are three key elements of the KNN classifier:

1. a set of labeled examples,
2. a distance or similarity metric to compute "nearness" between examples
   - For example, Euclidean distance
   - Or for discrete features similarity is 1 if same and 0 if different
3. the value of k - the number of nearest neighbors.

Conceptual Design of the KNN algorithm

- To classify an unlabeled example, the distance of this example to the labeled examples is computed,
- its k-nearest neighbors are identified and
- the class labels of these nearest neighbors are then used to determine the class label of the example.
- Once the k-nearest neighbors are found, classification of test example is based on the 'majority class' of its nearest neighbors

Formal definition

Given: $D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_m}, y_m)\}$

Parameters:

- Similarity function: $K\colon\ X \times X \rightarrow \mathbb{R}$
- Number of nearest neighbors to consider: k

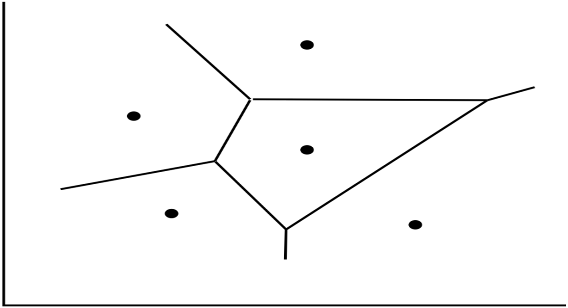
Prediction rule: for $\mathbf{x_0}$ use $nn(k, \mathbf{x_0})$ to denote the k examples in D with largest $K(\mathbf{x_i}, \mathbf{x_0})$. Then classify using:

$$h(\mathbf{x_0}) = \underset{y \in Y}{\arg\max} \sum_{i \in nn(k, \mathbf{x_0})} \delta(y, y_i)$$

where $\delta(a, b) = 1$ if $a = b$, otherwise $\delta(a, b) = 0$

Decision boundary Visualization

For 1-nearest neighbor with Euclidian distance, the *Voronoi diagram* gives the complex polyhedra segmenting the space into the regions closest to each point.



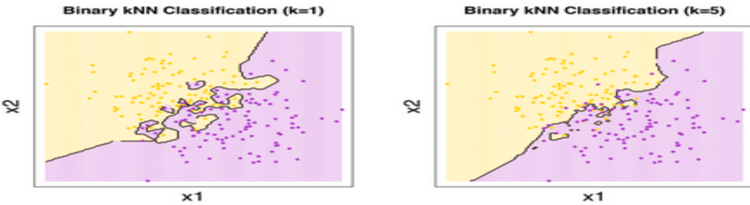
KNN for regression

We've been talking about classification - discrete Y

To extend to real-valued Y, we merely need to take the mean of the y values of the k-NNs:

$$h(\mathbf{x_0}) = \frac{\sum_{i \in nn(k, \mathbf{x_0})} y_i}{k}$$

Effect of k


Binary kNN Classification (k=1)
x2
x1
Binary kNN Classification (k=5)
x2
x1

Accuracy of K-Nearest Neighbor Classifiers

Several key issues that affect the accuracy of KNN classifiers:

- Choice of k:
  - If k is too small, then the result can be sensitive to noise points.
  - If k is too large, then the neighborhood may include too many points from other classes and blur the boundaries
  - In binary (two class) classification problems, it is helpful to choose k to be an odd number as this avoids tied votes

Accuracy of K-Nearest Neighbor Classifiers

- Another issue: Majority voting:
  - Can be a problem if the nearest neighbors vary widely in their distance and the closer neighbors more reliably indicate the class of the example.
  - Potential solution is to weight each examples's vote by its similarity, K so that

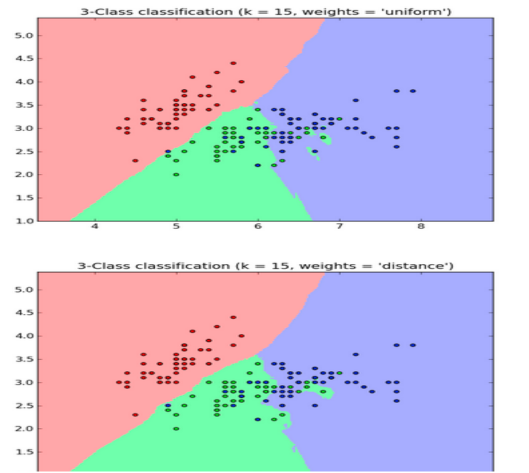
$$h(\mathbf{x_0}) = \arg\max_{y \in Y} \sum_{i \in nn(k,\mathbf{x_0})} K(\mathbf{x_i}, \mathbf{x_0})\delta(y_i, y)$$

    Or for regression:

$$h(\mathbf{x_0}) = \frac{\sum_{i \in nn(k,\mathbf{x_0})} K(\mathbf{x_i}, \mathbf{x_0}) y_i}{\sum_{i \in nn(k,\mathbf{x_0})} K(\mathbf{x_i}, \mathbf{x_0})}$$

  - Weighted majority voting decreases sensitivity of classifier to k

Effect of



3-Class classification (k = 15, weights = 'uniform')
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8
3-Class classification (k = 15, weights = 'distance')
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5

Scaling Tuples

Scaling is necessary to perform the distance calculation being invalidated by scale effects.

Sample of approaches

- Min-max normalizing the attribute to give $\hat{x} \in [0, 1]$

$$\hat{x} = \frac{x - min(x)}{max(x) - min(x)}$$

  e.g. 2,3,4,5 $\rightarrow$ 0, 2/5, 3/5, 1

- Standardizing the attribute

$$\hat{x} = \frac{x - \mu_x}{\sigma_x}$$

Strengths of KNN classification

- KNN classification is an easy to understand and easy to implement classification technique
- KNN is particularly well suited for multi-modal classes
- Often successful when the decision boundary is very irregular
- Training is easy!

Weaknesses of KNN classification

- KNN is computationally complex because the model needs to be "re-trained" for each test example
- The accuracy of the k-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance
- The performance of KNN is sensitive to the choice of k and the 'majority voting' approach
- For 'majority voting' classification, classes with the more frequent examples tend to dominate the prediction of the new vector as they tend to occur in the k nearest neighbors due to their large number
- The decision boundary is piecewise linear