K Nearest Neighbors

POLS 8500

(Adapted from various sources)

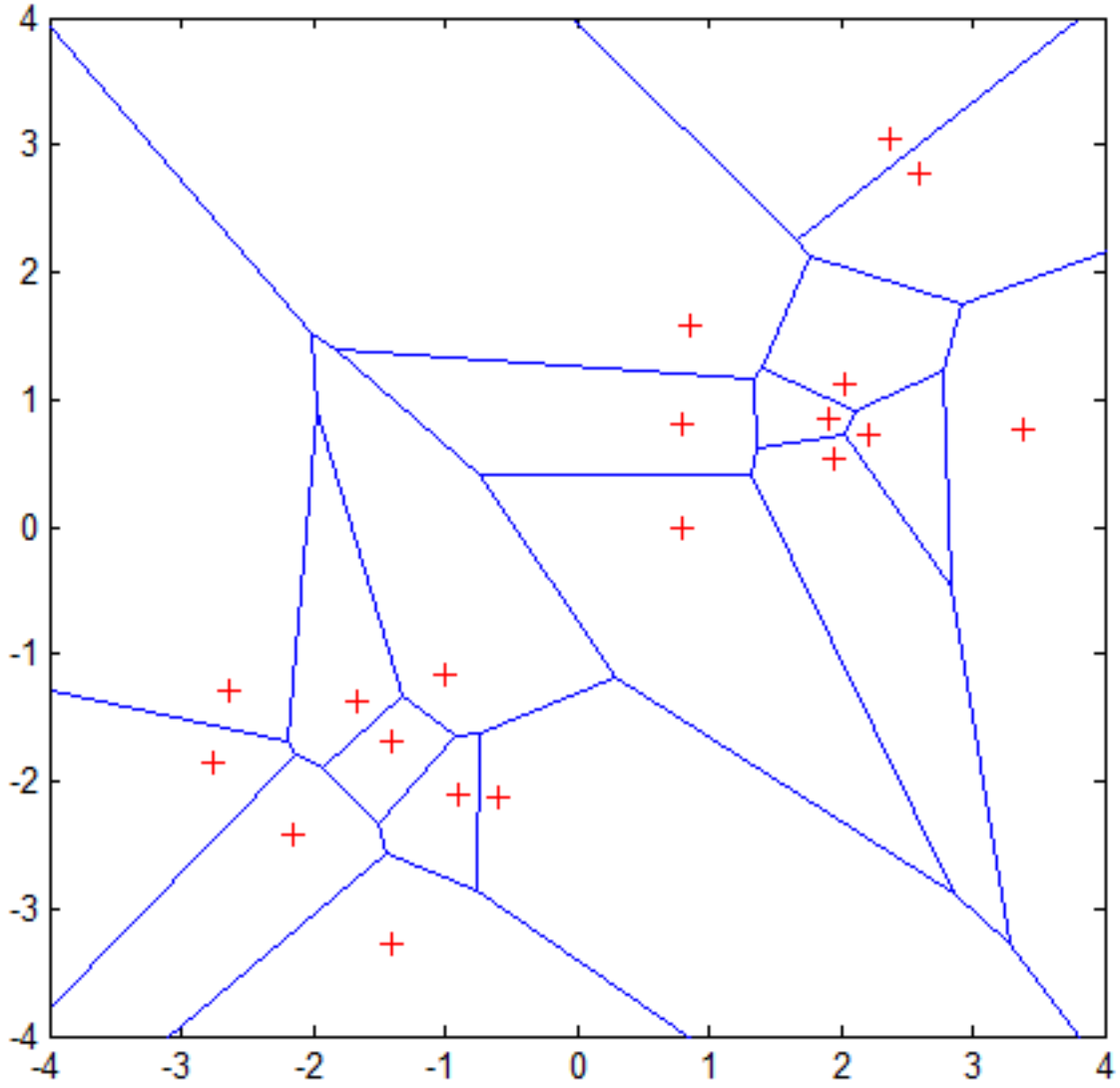
Introduction

- Instance-based learning is often termed *lazy* learning.
- Typically no “transformation” of training instances into more general “statements.”
- Instead, the presented training data is simply stored and, when a new query instance is encountered, a set of similar, related instances is retrieved from memory and used to classify the new query instance
- Hence, instance-based learners never form an explicit general hypothesis regarding the target function. They simply compute the classification of each new query instance as needed

k-NN Approach

- The simplest, most used instance-based learning algorithm is the *k*-NN algorithm
- *k*-NN assumes that all instances are points in some *n*-dimensional space and defines neighbors in terms of distance (usually Euclidean in R-space)
- *k* is the number of neighbors considered

Graphic Depiction



Properties:

1) All possible points within a sample's Voronoi cell are the nearest neighboring points for that sample
2) For any sample, the nearest sample is determined by the closest Voronoi cell edge

Basic Idea

- Using the second property, the *k*-NN classification rule is to assign to a test sample the majority category label of its *k* nearest training samples
- In practice, *k* is usually chosen to be odd, so as to avoid ties
- The *k* = 1 rule is generally called the nearest-neighbor classification rule

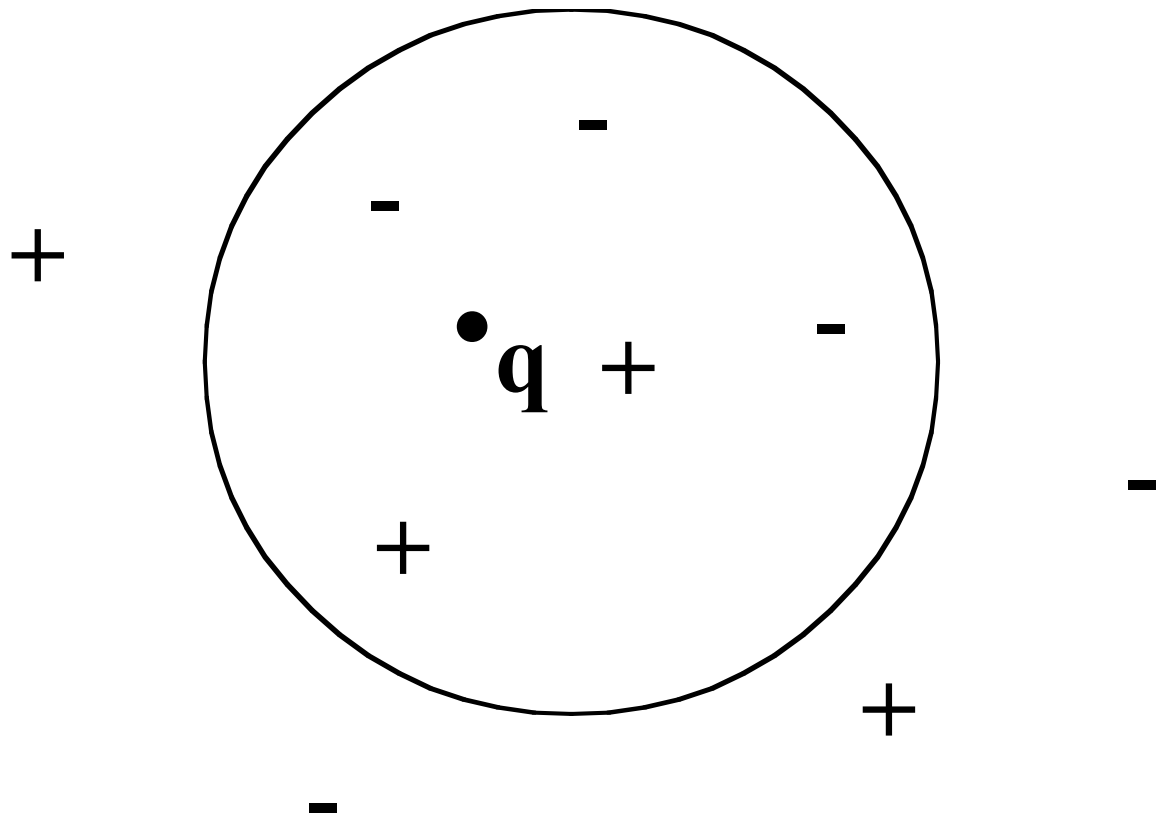
k-NN Algorithm

- For each training instance $t=(x, f(x))$
  - Add t to the set *Tr_instances*
- Given a query instance q to be classified
  - Let $x_1, \ldots, x_k$ be the k training instances in *Tr_instances* nearest to q
  - Return

- Where V is the finite set of target class values, and $\delta(a,b)=1$ if $a=b$, and 0 otherwise (Kronecker function)
- Intuitively, the k-NN algorithm assigns to each new query instance the majority class among its k nearest neighbors

$$\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$$

Simple Illustration



q is + under 1-NN, but – under 5-NN

Distance-weighted *k*-NN

- Replace $\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$ by:

$$\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} \frac{1}{d\left(x_i, x_q\right)^2} \delta(v, f(x_i))$$

Scale Effects

- Different features may have different measurement scales
  - E.g., patient weight in kg (range [50,200]) vs. blood protein values in ng/dL (range [-3,3])
- Consequences
  - Patient weight will have a much greater influence on the distance between samples
  - May bias the performance of the classifier

Standardization

- Transform raw feature values into z-scores

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

  - x_{ij} is the value for the *i*th sample and *j*th feature
  - μ_j is the average of all x_{ij} for feature *j*
  - σ_j is the standard deviation of all x_{ij} over all input samples
- Range and scale of z-scores should be similar (providing distributions of raw feature values are alike)

Distance Metrics

Minkowsky:

$$D(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{i=1}^{m} |x_i - y_i|^r \right)^{1/r}$$

Euclidean:

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

Manhattan / city-block:

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

Camberra:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:

$$D(\boldsymbol{x},\boldsymbol{y}) = \max_{i=1}^{m} |x_i - y_i|$$

Quadratic:

$$D(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T Q(\boldsymbol{x} - \boldsymbol{y}) = \sum_{j=1}^{m} \left(\sum_{i=1}^{m} (x_i - y_i) q_{ji} \right) (x_j - y_j)$$

Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:

$$D(\boldsymbol{x},\boldsymbol{y}) = [\det V]^{1/m} (\boldsymbol{x} - \boldsymbol{y})^{\mathrm{T}} V^{-1} (\boldsymbol{x} - \boldsymbol{y})$$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_{i=1}^{m} (x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m} (x_i - \overline{x_i})^2 \sum_{i=1}^{m} (y_i - \overline{y_i})^2}}$$

$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{1}{sum_i} \left(\frac{x_i}{size_{\boldsymbol{x}}} - \frac{y_i}{size_{\boldsymbol{y}}} \right)^2$$

sum_i is the sum of all values for attribute i occuring in the training set, and $size_x$ is the sum of all values in the vector $\boldsymbol{x}$.

Kendall's Rank Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^{m} \sum_{j=1}^{i-1} \mathrm{sign}(x_i - x_j)\,\mathrm{sign}(y_i - y_j)$$

sign(x)=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

Figure 1. Equations of selected distance functions.
($\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of m attribute values).

Issues with Distance Metrics

- Most distance measures were designed for linear/real-valued attributes
- Two important questions in the context of machine learning:
  - How best to handle nominal attributes
  - What to do when attribute types are mixed

Distance for Nominal Attributes

Value Difference Metric (VDM)

[Stanfill & Waltz, 1986]

Providing appropriate distance measurements for nominal attributes.

$$vdm_a(x,y) = \sum_{c=1}^{C} \left(\frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right)^2$$

$N_{a,x}$ = # times attribute a had value x

$N_{a,x,c}$ = # times attribute a had value x and class was c

C = # output classes

Two values are considered closer if they have more similar classifications, i.e., if they have more similar correlations with the output classes.

Distance for Heterogeneous Data

In this section, we define a heterogeneous distance function *HVDM* that returns the distance between two input vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. It is defined as follows:

$$HVDM(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{a=1}^{m} d_a^2(x_a, y_a)} \tag{11}$$

where m is the number of attributes. The function $d_a(x,y)$ returns a distance between the two values x and y for attribute a and is defined as:

$$d_a(x,y) = \begin{cases} 1, & \text{if } x \text{ or } y \text{ is unknown; otherwise...} \\ normalized\_vdm_a(x,y), & \text{if } a \text{ is nominal} \\ normalized\_diff_a(x,y), & \text{if } a \text{ is linear} \end{cases} \tag{12}$$

*Wilson, D. R. and Martinez, T. R., Improved Heterogeneous Distance Functions, Journal of Artificial Intelligence Research, vol. **6**, no. 1, pp. 1-34, 1997*

Some Remarks

- *k*-NN works well on many practical problems and is fairly noise tolerant (depending on the value of *k*)
- *k*-NN is subject to the curse of dimensionality (i.e., presence of many irrelevant attributes)
- *k*-NN needs adequate distance measure
- *k*-NN relies on efficient indexing

How is kNN Incremental?

- All training instances are stored
- Model consists of the set of training instances
- Adding a new training instance only affects the computation of neighbors, which is done at execution time (i.e., lazily)

  - Note that the storing of training instances is a violation of the strict definition of incremental learning.

Predicting Continuous Values

- Replace by:

$$\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

- Note: unweighted corresponds to w_i=1 for all i

$$\hat{f}(q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

Case-based Reasoning

- An extension of IBL
  - Cases represented by rich symbolic descriptions, rather than $\Re^n$
  - Multiple retrieved cases are combined using knowledge-based reasoning rather than statistical methods
  - Strong interaction among case retrieval, knowledge-base reasoning and problem solving (e.g., adapting cases for matching)