Lecture 7 Instance-based Learning

- Introduction of Instance-based learning
- K-Nearest Neighbor

Instance-based Learning - Overview

- In contrast to learning methods that construct a general, explicit description of the target function when training examples are provided, instance-based learning methods simply store the training examples.

- Generalizing beyond these examples is postponed until a new instance must be classified.

- Each time a new query instance is encountered, its relationship to the previously stored examples is examined in order to assign a target function value for the new instance.

Instance-based Learning - Overview

1. Instance-based learning methods are sometimes referred to as delayed/lazy learning methods because they delay processing until a new instance must be classified.

2. A key advantage of delayed/lazy learning is that instead of estimating the target function one for the entire instance space, these methods can estimate it locally and differently for each new instance to be classified.

Instance-based Learning - Overview

- Instance-based learning methods such as Nearest Neighbor are conceptually straightforward approaches to approximating real-valued or discrete-valued target functions.

    1. Learning in these algorithms consists of simply storing the presented training data.
    2. 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.

Instance-based Learning - Overview

- Many techniques construct only a local approximation to the target function that applies in the neighborhood of the new query instance, and never construct an approximation designed to perform well over the entire instance space.

- This has significant advantages when the target function is very complex, but can still be described by a collection of less complex local approximations.

Instance-based Learning - Disadvantages

1. One disadvantage of instance-based approaches is that the cost of classifying new instances can be high.

   - This is due to the fact that nearly all computation takes place at classification time rather than when the training examples are first encountered.

Instance-based Learning - Disadvantages

2. A second disadvantage to many instance-based approaches, especially nearest neighbor approaches, is that they typically consider all attributes of the instances when attempting to retrieve similar training examples from memory.

   - If the target concept depends on only a few of the many available attributes, then the instances that are truly most "similar" may well be a large distance apart.

K-nearest Neighbor Learning (Intro.)

- The most basic instance-based method is the k-NEAREST NEIGHBOR algorithm.

- This algorithm assumes all instances correspond to points in the n-dimensional space $\Re^n$. That means inputs of data are numeric ones.

- The nearest neighbors of an instance are defined in terms of the standard Euclidean distance.

K-nearest Neighbor Learning (Euclidean distance)

- Let an arbitrary instance x be described by the feature vector $\langle a_1(x), a_2(x), \ldots a_n(x) \rangle$

- where $a_r(x)$ denotes the value of the r^{th} attribute of instance x. Then the distance between two instances x_i and x_j is defined to be $d(x_i, x_j)$, where

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

K-nearest Neighbor Learning (output type)

- In nearest-neighbor learning the target function may be either discrete-valued or real-valued.

- Let us first consider learning discrete-valued target functions of the form $f : \mathscr{R}^n \rightarrow V$, where V is the finite set $\{v_l, \ldots v_s\}$.

- The k-NEAREST NEIGHBOR algorithm for approximating a discrete-valued target function is given in next page ->

K-nearest Neighbor Algorithm for approximating a discrete-valued function f : $\Re^n$ -> V,

- Training algorithm:
  - For each training example <x, f(x)>, add the example to the listing training_examples
- Classification algorithm
  - Given a query instance x_q to be classified,
  - Let $x_1 \ldots x_k$ denote the k instances from training_examples that are nearest to x_q
  - Return

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

- Where $\delta(a, b) = 1$ if a = b and 0 otherwise.

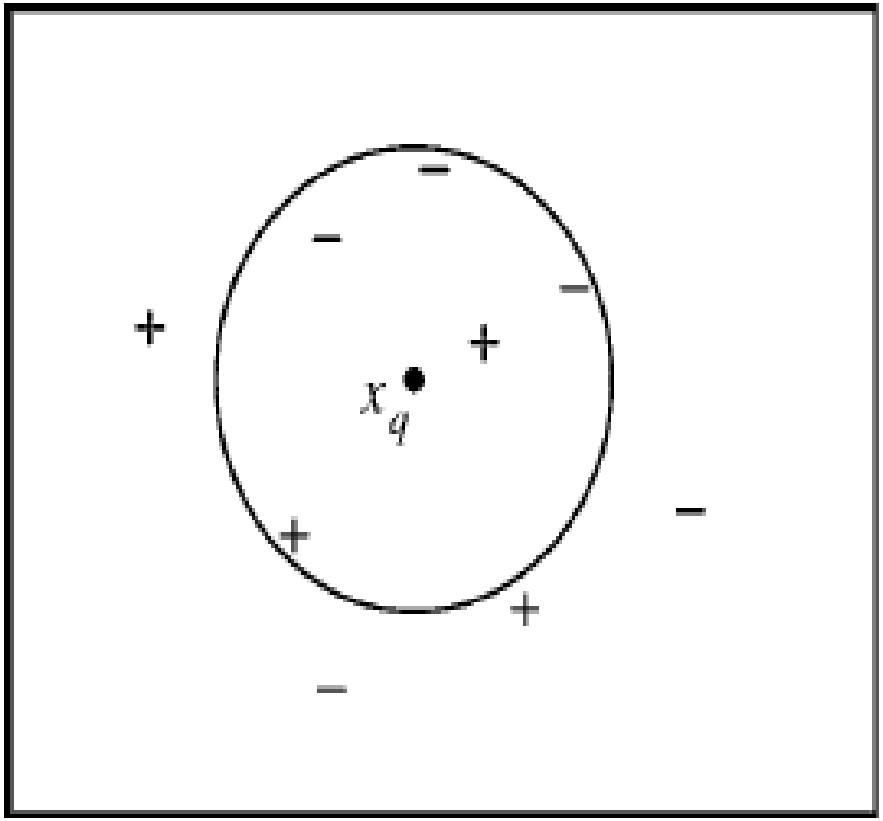
K-nearest Neighbor Algorithm for approximating a discrete-valued function $f : \Re^n \rightarrow V$,

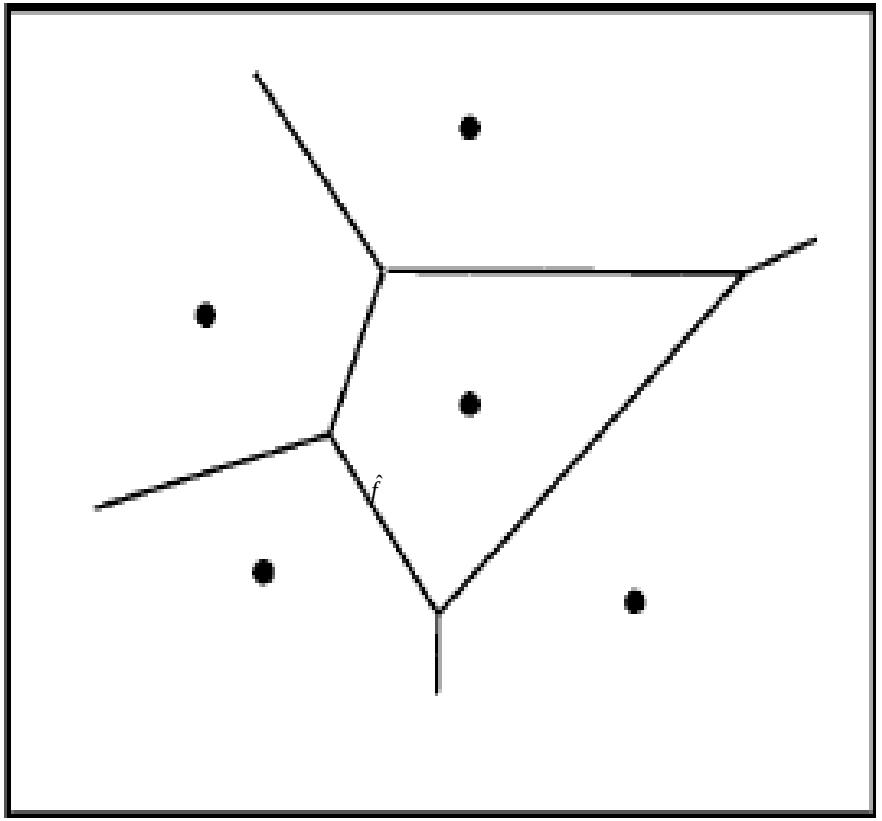
- As shown there, the value $\hat{f}(x_q)$ returned by this algorithm as its estimate of $f(x_q)$ is just the most common value of f among the k training examples nearest to x_q.

- If we choose k = 1, then the 1-NEAREST NEIGHBOR algorithm assigns to $\hat{f}(x_q)$ the value $f(x_i)$ where x_i is the training instance nearest to x_q.

- For larger values of k, the algorithm assigns the most common value among the k nearest training examples.

K-nearest Neighbor Algorithm for approximating a discrete-valued function f : $\Re^n$ -> V





1 Nearest Neighbor classifies x_q positive, 5 NearestNeighbor classifies x_q as negative. The diagram on the right handside shows the decision surface induced by 1-Nearest Neighbor.

K-nearest Neighbor Algorithm for approximating a discrete-valued function f : $\Re^n$ -> V

- Calculate the mean value of the k nearest training examples.

- Replace the final line of previous algorithm by:

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

Distance-Weighted Nearest Neighbor Algorithm for discrete-valued target functions

- Weight the contribution of each of the k neighbors according to their distance to the query point x_q, giving greater weight to closer neighbors.

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

where

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

- If x_q exactly match one of the training instances x_i, the denominator $d(x_q,x_i)^2$ is zero and, we assign $\hat{f}(x_q)$ to be $f(x_i)$.

Distance-Weighted Nearest Neighbor Algorithm for continuous- valued target functions

- We can distance-weight the instances for real-valued target functions in a similar way.

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

where

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

Remark

- Note all of the above variants of the k-NEAREST NEIGHBOR algorithm consider only the k nearest neighbors to classify the query point.

- Once we add distance weighting, there is no harm in allowing all training examples because very distant examples will have very little effect on $\hat{f}(x_q)$

- The only disadvantage of considering all examples is that our classifier will run more slowly.

- If all training examples are considered when classifying a new query instance, we call the algorithm a global method.

- If only the nearest training examples are considered, we call it a local method.

Remark

- The distance-weighted k-NEAREST NEIGHBOR algorithm is a highly effective inductive inference method for many practical problems.

- It is robust to noisy training data and quite effective when it is provided a sufficiently large set of training data.

- Note that by taking the weighted average of the k neighbors nearest to the query point, it can smooth out the impact of isolated noisy training examples.

Remark

- Since the algorithm delays all processing until a new query is received, significant computation can be required to process each new query.

- Various methods have been developed for indexing the stored training examples so that the nearest neighbors can be identified more efficiently at some additional cost in memory.

- One such indexing method is the kd-tree
- https://www.ri.cmu.edu/pub_files/pub1/moore_andrew_1991_1/moore_andrew_1991_1.pdf