Nearest Neighbor Models

Outline

- Concept of Instance-based classification
- Constructing k-Nearest Neighbor classifier

Nonparametric Models

- Parametric model:
  - Summarizes data with a set of parameters of fixed size
  - Assumes that the data is drawn from a model of known form

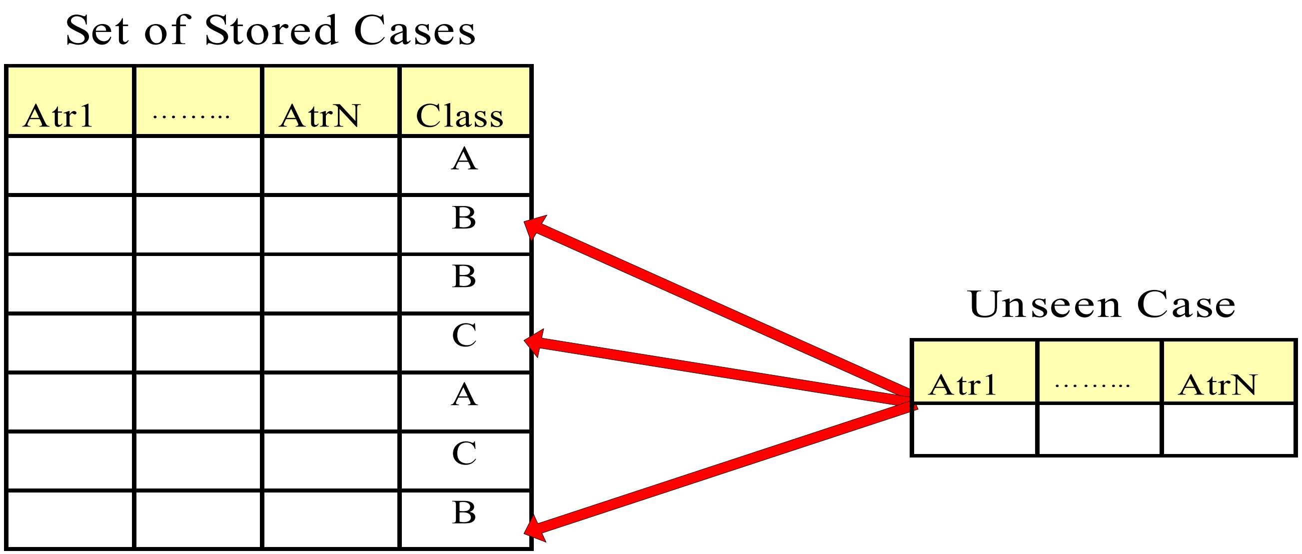
    E.g., linear regression
- Nonparametric model:
  - When the data cannot be characterized by a bounded set of parameters
  - We let the data speak for themselves

    (especially when a large volume of data are available)

    E.g., instance-based (memory-based) learning

Instance-Based Learning

- Store the training records
- Use training records to predict the class label of unseen cases

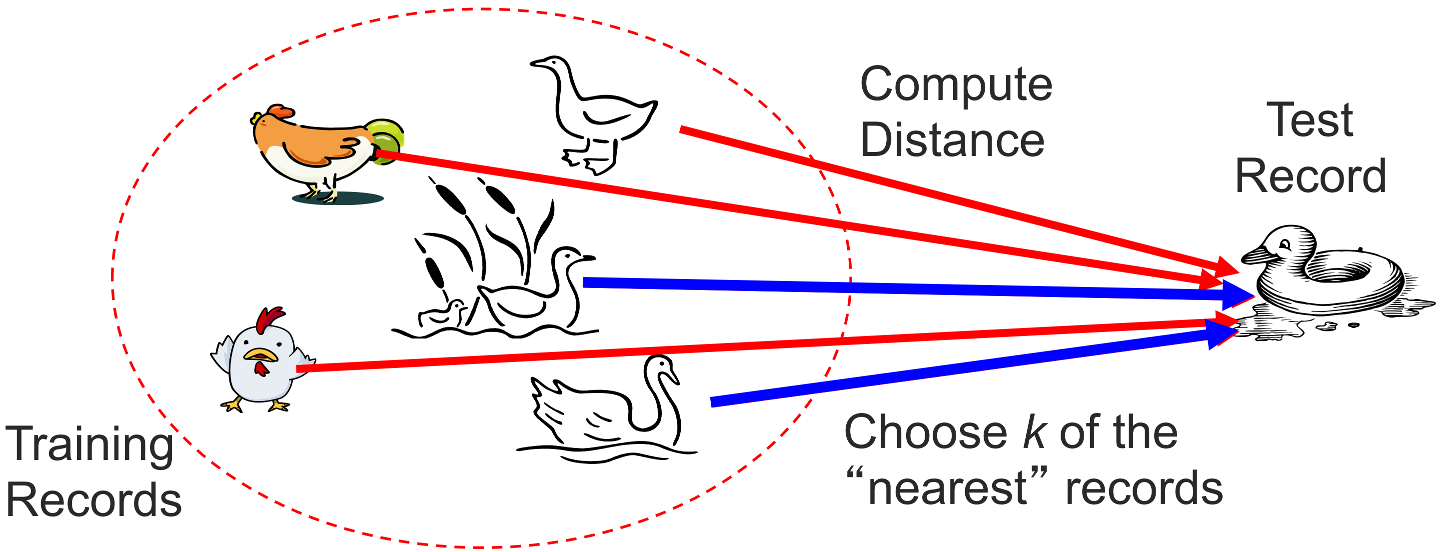


Instance-Based Learning

- Examples:
  - Rote-learner
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - Nearest neighbor
    - Uses *k* “closest” points (nearest neighbors) for performing classification

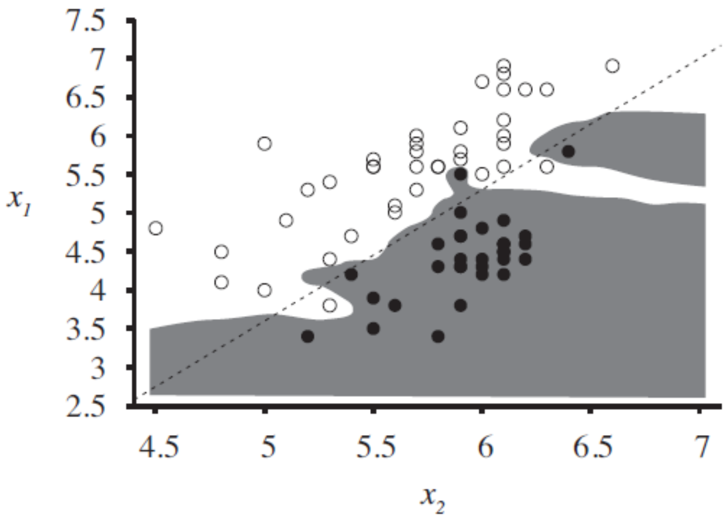
Nearest Neighbor Models

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck

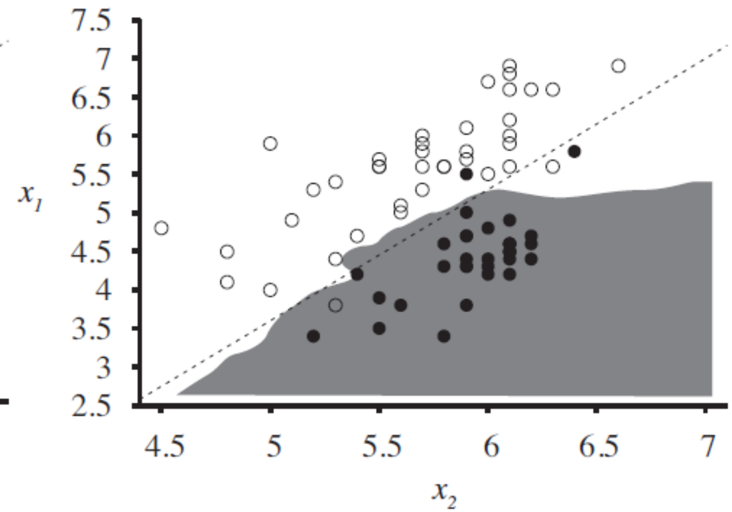


Nearest Neighbor Models

- Given a query $\mathbf{x}_q$, find the k nearest neighbors $NN(k, \mathbf{x}_q)$
  - Classification: plurality vote of $NN(k, \mathbf{x}_q)$
  - Regression: mean or median of $NN(k, \mathbf{x}_q)$ or
    solve a linear regression problem on $NN(k, \mathbf{x}_q)$
  - Can use cross-validation to select the best k

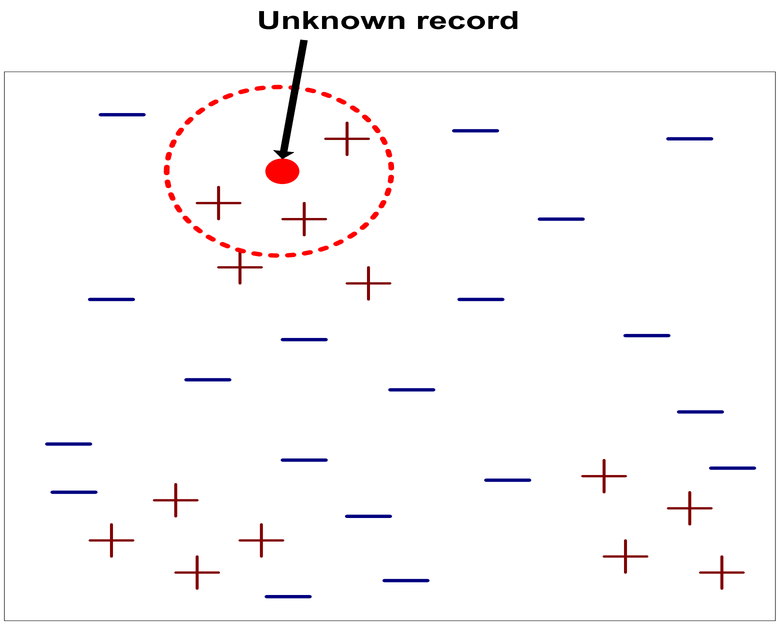


Overfitting with $k = 1$



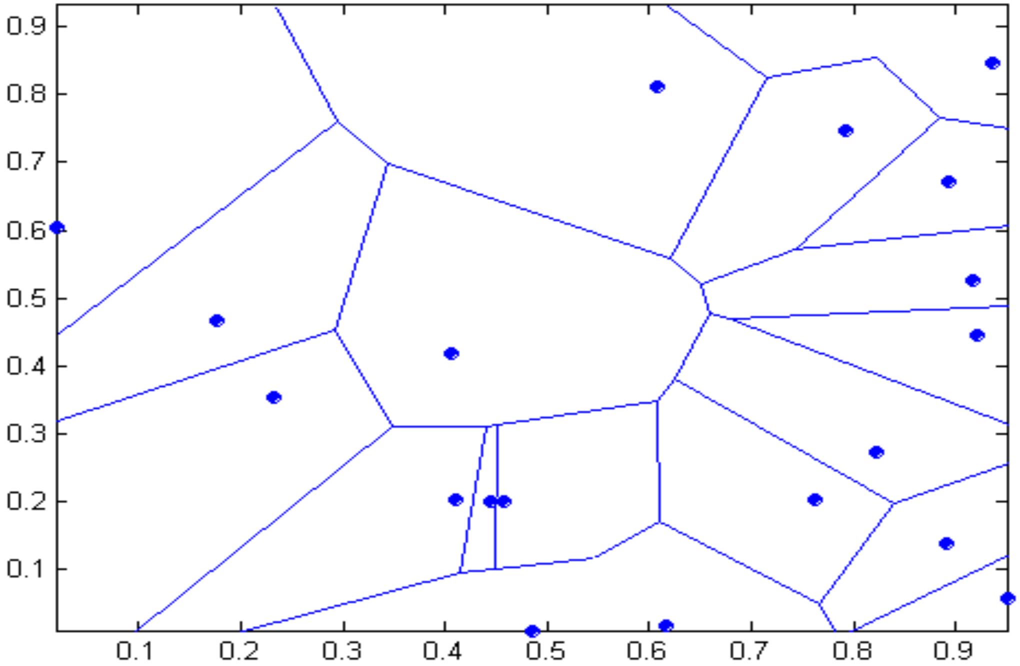
O.K. with $k = 5$

Nearest Neighbor Classifier



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

1-Nearest Neighbor Classifier



Distance and Normalization

- Distance metric: Minkowski distance or L^p norm

$$L^p(\mathbf{x}_j, \mathbf{x}_q) = \left(\sum_i \left| x_{j,i} - x_{q,i} \right|^p \right)^{1/p}$$

  - $p = 2$: Euclidean distance
  - $p = 1$: Manhattan distance
  - $p = 1$ with Boolean attributes: Hamming distance

- Normalization:

$$x_{j,i} \rightarrow (x_{j,i} - \mu_i)/\sigma_i$$

  - μ_i: mean of the values in the ith dimension
  - σ_i: standard deviation of the values in the ith dimension

Nearest Neighbor Models

- Curse of dimensionality: nearest neighbors are not very near!

  l: average side length of a neighborhood

  l^n: volume of the neighborhood hypercube

  - If N points are uniformly distributed in the full cube of volume 1,

    $$l^n/1 = k/N \;\longrightarrow\; l = (k/N)^{1/n}$$

  - E.g., let $k = 10$ and $N = 1{,}000{,}000$
    - $n = 3 \;\longrightarrow\; l = 0.02$
    - $n = 17 \;\longrightarrow\; l = 0.5$
    - $n = 200 \;\longrightarrow\; l = 0.94$

Nearest Neighbor Models

- Time complexity:
  - $O(N)$ with a sequential table
  - $O(\log N)$ with a binary tree $\rightarrow$ k-d tree
  - $O(1)$ with a hash table $\rightarrow$ locality-sensitive hashing

Finding Nearest Neighbors with k-d Trees

- k-d tree
  - A balanced binary tree over data with k dimensions
- Construction:
  - Select the dimension along which the variance of the data is the greatest
  - Choose the median and split at that point
  - Continue recursively

Finding Nearest Neighbors with k-d Trees

- k nearest neighbor lookup:
  - Suppose the query point is in a rectangle and the distance to the dividing boundary is r
  - If we cannot find k examples within the rectangle that are closer than r we need to search the rectangle on the other side of the boundary
  - ➡ k-d trees work well when there are at least 2^n examples (up to $n = 10$ with thousands of examples, and 20 with millions)

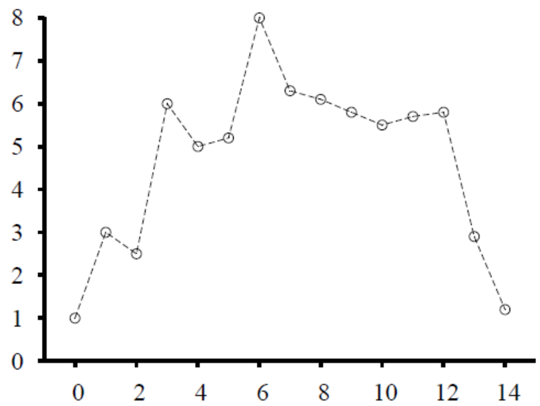
Locality-Sensitive Hashing

- Approximate near-neighbors problem:
  - Given a data set of example points and a query point $\mathbf{x}_q$, find, with high probability, an example point (or points) that is near $\mathbf{x}_q$
- Locality-sensitive hash:
  - A hash table can be created by projecting the data onto a line and discretize the line into hash bins
  - We choose l random projections and create l hash tables, $g_1(\mathbf{x}), \ldots, g_l(\mathbf{x})$
  - Given a query point $\mathbf{x}_q$, we fetch the set of points in bin $g_j(\mathbf{x}_q)$ for each j, and union them into a set of candidate points C
  - We find the k closest points from C by computing the actual distance to $\mathbf{x}_q$

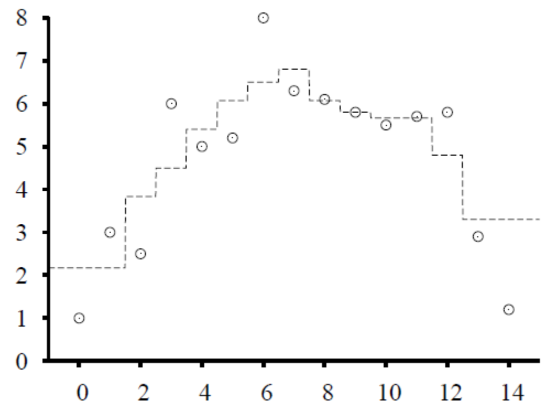
Nonparametric Regression

- *k*-nearest-neighbors regression:
  - *k*-NN average: $h(x) = \sum y_j / k$
    - Poor estimate at outlying points
      (evidence comes from one side, ignores trend)
  - *k*-NN linear regression
    - Finds best line through k examples
    - Captures trend at outliers
- Locally weighted regression:
  - Avoids discontinuities in $h(x)$
  - Examples are weighted by a kernel function
  - Weight decreases gradually as the distance to the query point increases

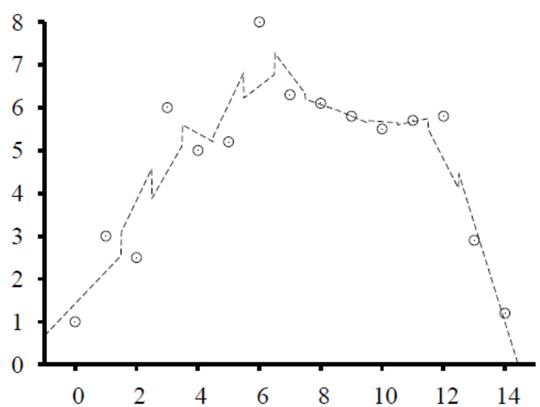
Nonparametric Regression



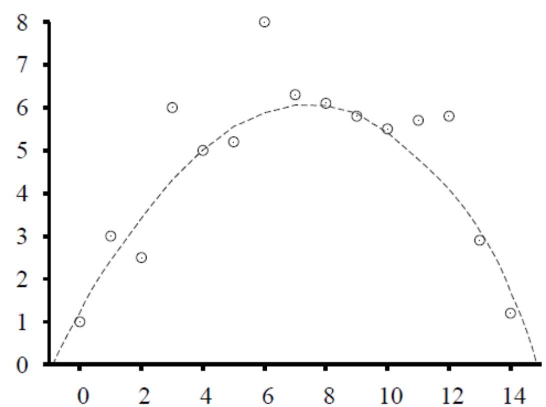
(a) connect the dots



(b) 3-NN average



(c) 3-NN linear regression



(d) locally weighted regression (quadratic kernel, $k = 10$)

Nonparametric Regression

- Kernel function:
  - Should be symmetric around 0, have a maximum at 0
  - Area under the kernel must be bounded
  - The shape does not matter much, kernel width is more important (underfitting vs. overfitting)
  - Best kernel width can be chosen by cross-validation

A quadratic kernel,
$\mathcal{K}(x) = \max(0, 1 - (2|x|/k)^2)$,
with kernel width $k = 10$,
centered on the query point $x = 0$