CMSC 726
Lecture 10: Instance-based Learning

Lise Getoor

October 5, 2010

ACKNOWLEDGEMENTS: The material in this course is a synthesis of materials from many sources, including: Hal Daume III, Mark Drezde, Carlos Guestrin, **Rich Maclin,** Andrew Ng, Ben Taskar, Eric Xing, and others. I am very grateful for their generous sharing of insights and

Today's Topics

- Nearest Neighbor
- Locally weighted regression
- Case-based reasoning

Some Vocabulary

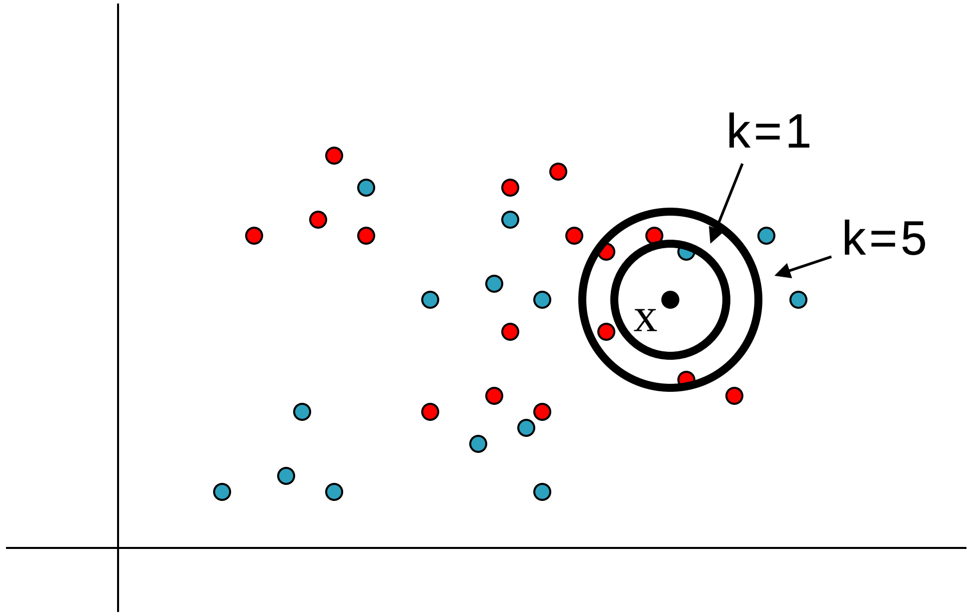
- **Parametric vs. Non-parametric:**
  - **parametric**:
    - A particular functional form is assumed, e.g., multivariate normal, naïve Bayes.
    - Advantage of simplicity – easy to estimate and interpret
    - may have high bias because the real data may not obey the assumed functional form.
  - **non-parametric:**
    - distribution or density estimate is data-driven and relatively few assumptions are made a priori about the functional form.
- Related terms: Instance-based, Memory-based, Lazy, Case-based, kernel methods…

Nearest Neighbor Algorithm

- Learning Algorithm:
  - Store training examples
- Prediction Algorithm:
  - To classify a new example **x** by finding the training example ($\mathbf{x}^i$,y^i) that is *nearest* to **x**
  - Guess the class $y = y^i$

K-Nearest Neighbor Methods

- To classify a new input vector x, examine the k-closest training data points to x and assign the object to the most frequently occurring class

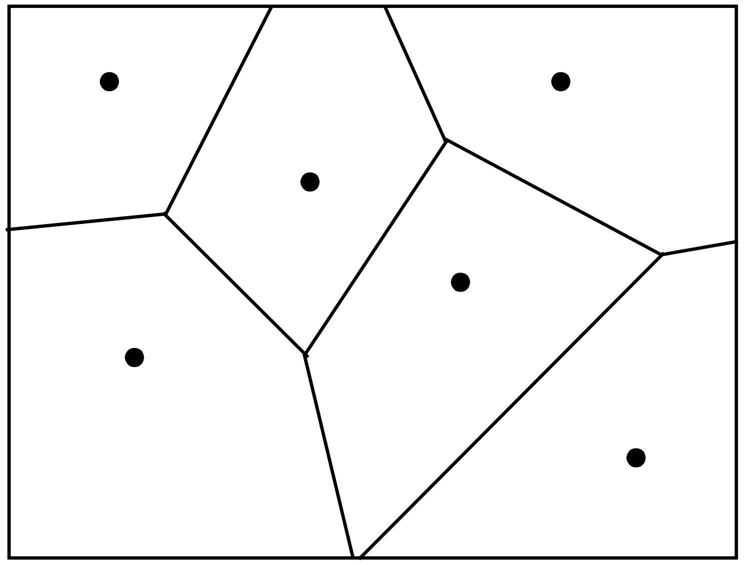


common values for k: 3, 5

Decision Boundaries

- The nearest neighbor algorithm does not explicitly compute decision boundaries. However, the decision boundaries form a subset of the Voronoi diagram for the training data.

1-NN Decision Surface



- Each line segment is equidistant between two points of opposite classes. The more examples that are stored, the more complex the decision boundaries can become.

Instance-Based Learning

Key idea : just store all training examples $< x_i, f(x_i) >$

Nearest neighbor (1 - Nearest neighbor) :

- Given query instance x_q, locate nearest example x_n, estimate

  $\hat{f}(x_q) \leftarrow f(x_n)$

k – Nearest neighbor :

- Given x_q, take vote among its k nearest neighbors (if discrete - valued target function)
- Take mean of f values of k nearest neighbors (if real - valued)

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

Distance-Weighted *k*-NN

Might want to weight nearer neighbors more heavily...

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

where

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

and $d(x_q, x_i)$ is distance between x_q and x_i

Note, now it makes sense to use *all* training examples instead of just k

Nearest Neighbor

When to Consider

- Instance map to points in R^n
- Less than 20 attributes per instance
- Lots of training data

Advantages

- Training is very fast
- Learn complex target functions
- Do not lose information

Disadvantages

- Slow at query time
- Easily fooled by irrelevant attributes

Issues

- Distance measure
  - Most common: Euclidean
- Choosing k
  - Increasing k reduces variance, increases bias
- For high-dimensional space, problem that the nearest neighbor may not be very close at all!
- Memory-based technique. Must make a pass through the data for each classification. This can be prohibitive for large data sets.

Distance Measures

- Many ML techniques (NN, clustering) are based on similarity measures between objects.

- Two methods for computing similarity:
  1. Explicit similarity measurement for each pair of objects
  2. Similarity obtained indirectly based on vector of object attributes.

- Normalize Feature Values. All features should have the same range of values. Otherwise, features with larger ranges will be treated as more important.

Distance

- Notation: object with p measurements

$$x^i = (x^i_1, x^i_2, \ldots, x^i_p)$$

- Most common distance metric is *Euclidean* distance:

$$d_E(x^i, x^j) = \left(\sum_{k=1}^{p} (x^i_k - x^j_k)^2 \right)^{\frac{1}{2}}$$

- efficiency trick: using squared Euclidean distance gives same answer, avoids computing square root
- ED makes sense when different measurements are commensurate; each is variable measured in the same units.
- If the measurements are different, say length and weight, it is not clear.

Standardization

When variables are not commensurate, we can standardize them by dividing by the sample standard deviation. This makes them all equally important.

The estimate for the standard deviation of x_k :

$$\hat{\sigma}_k = \left(\frac{1}{n} \sum_{i=1}^{n} \left(x_k^i - \bar{x}_k \right)^2 \right)^{\frac{1}{2}}$$

where x_k is the sample mean:

$$\bar{x}_k = \frac{1}{n} \sum_{i=1}^{n} x_k^i$$

Weighted Euclidean distance

Finally, if we have some idea of the relative importance
Of each variable, we can weight them:

$$d_{WE}(i, j) = \left(\sum_{k=1}^{p} w_k (x_k^i - x_k^j)^2 \right)^{\frac{1}{2}}$$

One option: weight each feature by its mutual information with the class.

Dealing with Correlation

- Standardize the variables, not just in direction of variable, but also taking into account covariances.
- Assume we have two variables or attributes X_j and X_k and n objects. The sample covariance is:

$$\mathrm{Cov}(X_j, X_k) = \frac{1}{n}\sum_{i=1}^{n}(x_j^i - \overline{x}_j)(x_k^i - \overline{x}_k)$$

- The covariance is a measure of how X_j and X_k vary together.
  - large and positive if large values of X_j are associated with large values of X_k, and small $X_j \Rightarrow$ small X_k
  - large and negative if large $X_j \Rightarrow$ small X_k
- More generally, we can form the covariance matrix Σ, in which each element (i,j) is the covariance of the ith and jth feature.

Sample correlation coefficient

- Covariance depends on ranges of X_j and X_k
- Standardize by dividing by standard deviation
- Sample correlation coefficient

$$\rho(X_j, X_k) = \frac{\sum_{i=1}^{n} (x_j^i - \overline{x})(x_k^i - \overline{y})}{\left(\sum_{i=1}^{n} (x_j^i - \overline{x})^2 (x_k^i - \overline{y})^2 \right)^{\frac{1}{2}}}$$

Mahalanobis distance

$$d_{MH}(x^i, x^j) = \left(\left(x^i - x^j\right)^T \Sigma^{-1} \left(x^i - x^j\right)\right)^{\frac{1}{2}}$$

1. It automatically accounts for the scaling of the coordinate axes
2. It corrects for correlation between the different features

Price:

1. The covariance matrices can be hard to determine accurately
2. The memory and time requirements grow quadratically rather than linearly with the number of features.

Other Distance Metrics

- Minkowski or L_λ metric:

$$d(i, j) = \left(\sum_{k=1}^{p} (x_k(i) - x_k(j))^{\lambda} \right)^{\frac{1}{\lambda}}$$

- Manhattan, city block or L_1 metric:

$$d(i, j) = \sum_{k=1}^{p} \left| x_k(i) - x_k(j) \right|$$

- L_∞

$$d(i, j) = \max_{k} \left| x_k(i) - x_k(j) \right|$$

Binary Data

	j=1	j=0
i=1	n_{11}	n_{10}
i=0	n_{01}	n_{00}

- matching coefficient

$$\frac{n_{11}+n_{00}}{n_{11}+n_{10}+n_{01}+n_{00}}$$

- Jaccard coefficient

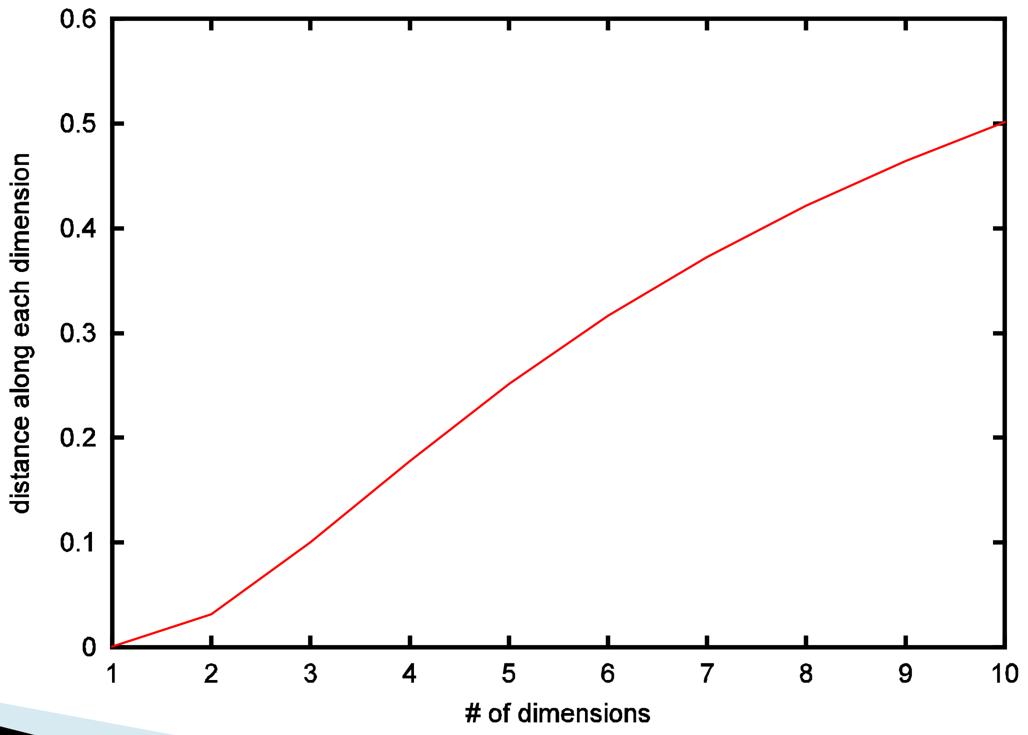
$$\frac{n_{11}}{n_{11}+n_{10}+n_{01}}$$

The Curse of Dimensionality

- Nearest neighbor breaks down in high-dimensional spaces because the "neighborhood" becomes very large.
- Suppose we have 5000 points uniformly distributed in the unit hypercube and we want to apply the 5-nearest neighbor algorithm.
- Suppose our query point is at the origin.
  - 1D –
    - On a one dimensional line, we must go a distance of $5/5000 = 0.001$ on average to capture the 5 nearest neighbors
  - 2D –
    - In two dimensions, we must go sqrt(0.001) to get a square that contains 0.001 of the volume
  - D –
    - In d dimensions, we must go $(0.001)^{1/d}$

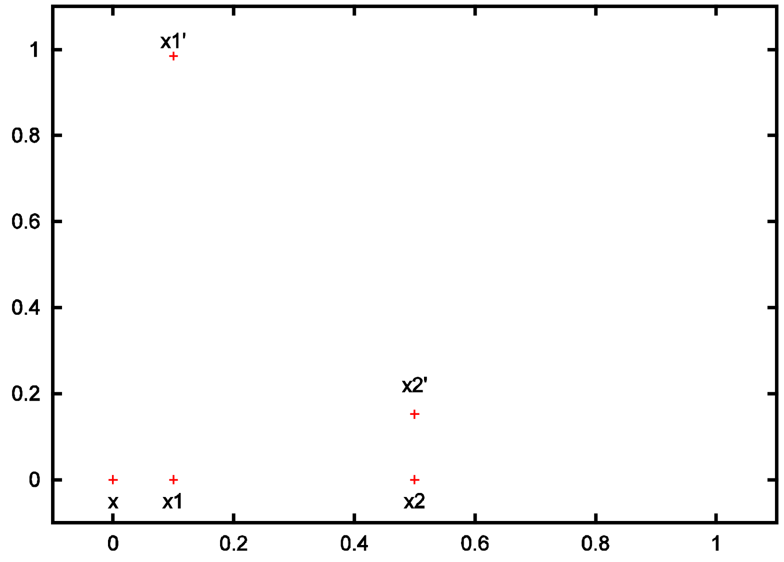
Curse of Dimensionality cont.

- With 5000 points in 10 dimensions, we must go 0.501 distance along each attribute in order to find the 5 nearest neighbors!



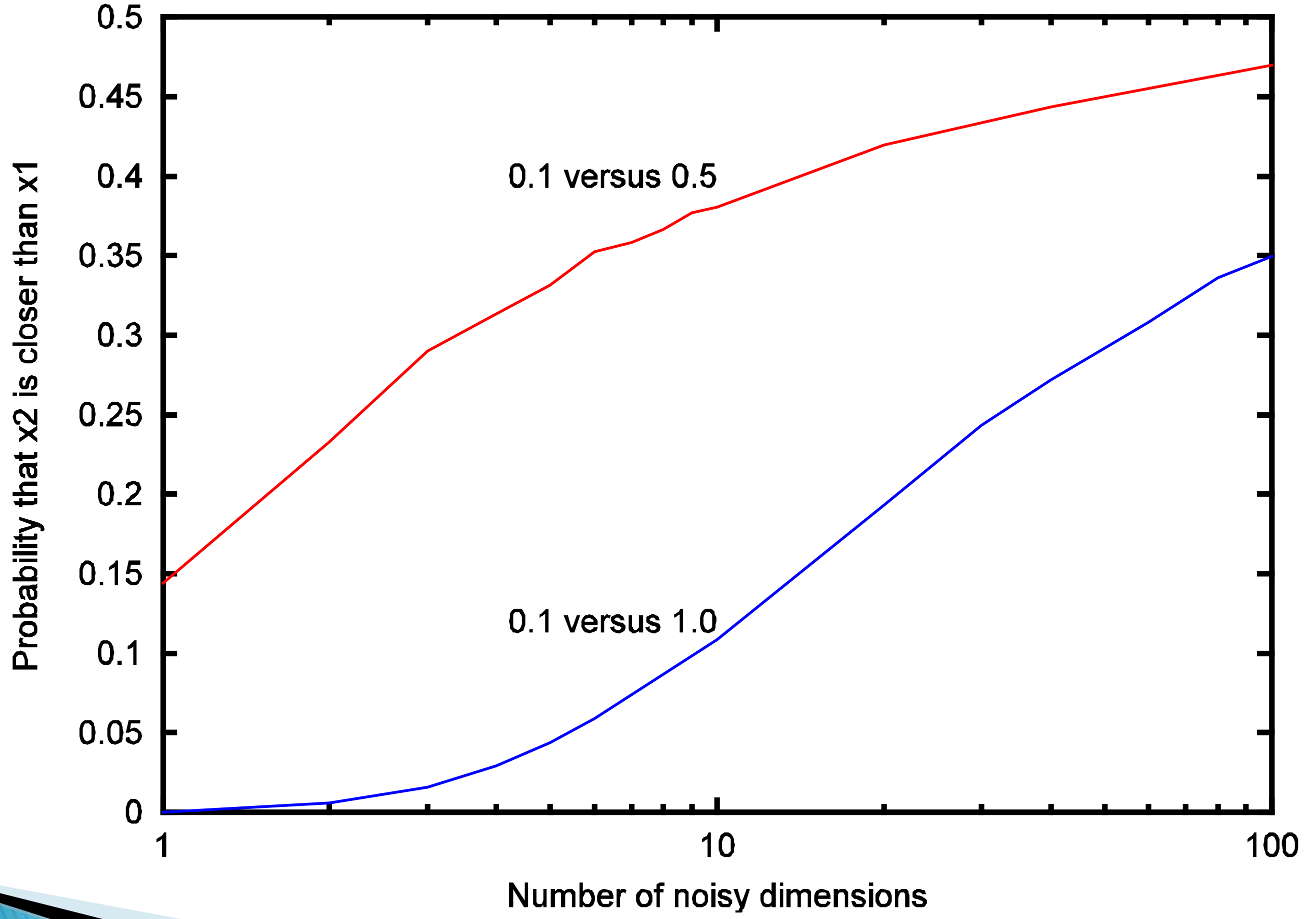
The Curse of Noisy/Irrelevant Features

- NN also breaks down when irrelevant, noisy features are included
- Suppose that our query point is at the origin, and there is one relevant feature and that our nearest neighbor is a positive example x1 at position 0.1 and our second nearest neighbor is a negative example x2 at 0.5.



- Now suppose we add one uniformly random feature. What is the probability that x2 will now be closer to x than x1?
- Approximately 0.15!

The Curse of Noisy cont.

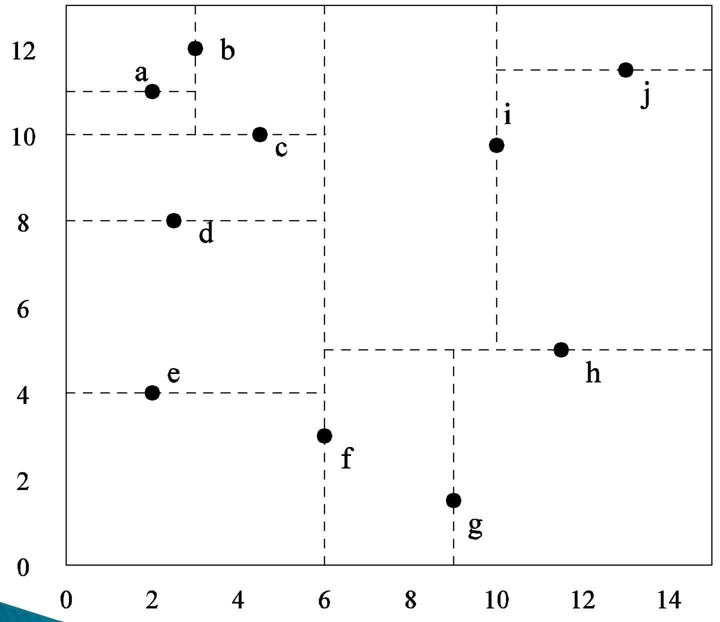


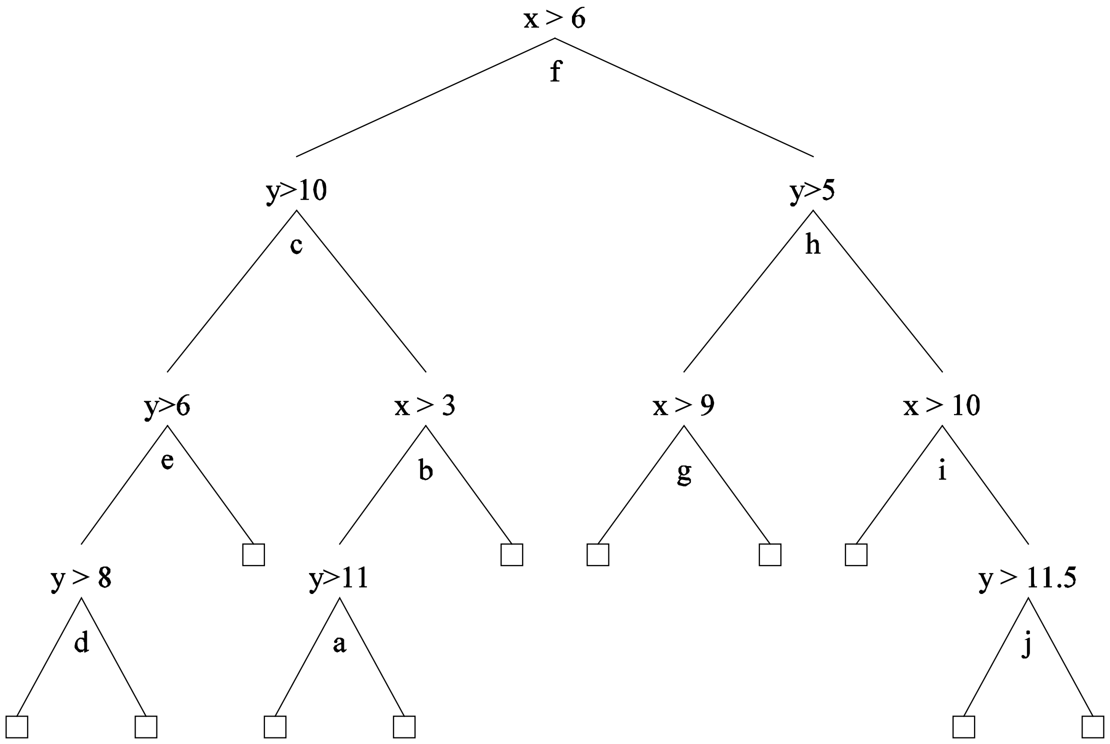
Efficient Indexing: Kd-trees

- A kd-tree is similar to a decision tree, except that we split using the median value along the dimension having the highest variance, and points are stored
- A kd-tree is a tree with the following properties
  - Each node represents a rectilinear region (faces aligned with axes)
  - Each node is associated with an axis aligned plane that cuts its region into two, and it has a child for each sub-region
  - The directions of the cutting planes alternate with depth

Kd-trees

- A kd-tree is similar to a decision tree, except that we split using the median value along the dimension having the highest variance, and points are stored





Edited Nearest Neighbor

- Storing all of the training examples can require a huge amount of memory. Select a subset of points that still give good classifications.
  - **Incremental deletion**. Loop through the training data and test each point to see if it can be correctly classified give the other points. If so, delete it from the data set.
  - **Incremental growth.** Start with an empty data set. Add each point to the data set only if it is not correctly classified by the points already stored.

KNN Advantages

- Easy to program
- No optimization or training required
- Classification accuracy can be very good; can outperform more complex models
- Easy to add *reject* option

Nearest Neighbor Summary

- Advantages
  - variable-sized hypothesis space
  - Learning is extremely efficient
    - however growing a good kd-tree can be expensive
  - Very flexible decision boundaries
- Disadvantages
  - distance function must be carefully chosen
  - Irrelevant or correlated features must be eliminated
  - Typically cannot handle more than 30 features
  - Computational costs: Memory and classification-time computation

Locally Weighted Regression

k - NN forms local approximation to f for each query point x_q

Why not form explicit approximation $\hat{f}(x)$ for region around x_q?

- Fit linear function to k nearest neighbors
- Or fit quadratic, etc.
- Produces "piecewise approximation" to f

Several choices of error to minimize :

- Squared error over k nearest neighbors

$$E_1(x_q) \equiv \tfrac{1}{2} \sum_{x \in \text{k nearest neighbors of } x_q} (f(x) - \hat{f}(x))^2$$

- Distance - weighted squared error over all neighbors

$$E_2(x_q) \equiv \tfrac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

Radial Basis Function Networks

- Global approximation to target function, in terms of linear combination of local approximations
- Used, for example, in image classification
- A different kind of neural network
- Closely related to distance-weighted regression, but “eager” instead of “lazy”

Case-Based Reasoning: NN for Relational Representations

- When examples are described in a relational language and used for tasks such as legal reasoning and planning, NN methods are usually referred to as case-based reasoning
- Computing similarity of two relationally described instances is computationally complex since it is basically subgraph isomorphism, an NP-complete problem. Combinatorics arise from how pieces of one case (nodes) are matched to pieces of the other case. Therefore, complex and frequently ad-hoc indexing methods are used to find the closest cases.
- For planning and problem solving complex adaptation methods are needed to adapt previous retrieved solutions to new problems.

Case-Based Reasoning

Can apply instance-based learning even when $X \neq R^n$

$\rightarrow$ need different “distance” metric

Case-Based Reasoning is instance-based learning applied to instances with symbolic logic descriptions:

```
((user-complaint error53-on-shutdown)
 (cpu-model PowerPC)
 (operating-system Windows)
 (network-connection PCIA)
 (memory 48meg)
 (installed-applications Excel Netscape VirusScan)
 (disk 1Gig)
 (likely-cause ???))
```

Ex. Case-Based Reasoning in CADET

CADET: stored examples of mechanical devices

- each training example:

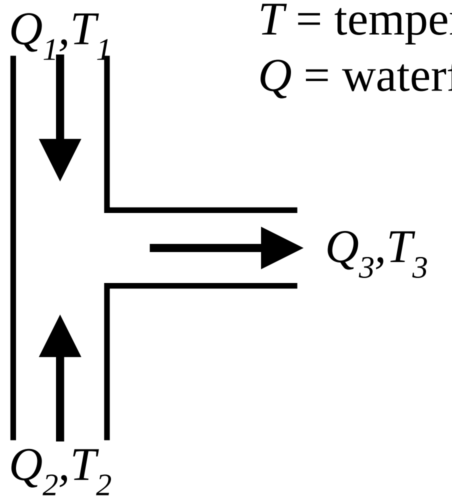
  <qualitative function, mechanical structure>
- new query: desired function
- target value: mechanical structure for this function

Distance metric: match qualitative function descriptions

Case-Based Reasoning in CADET

A stored case: T-junction pipe

Structure:



T = temperature
Q = waterflow

Function:

$$\begin{tikzcd}
Q_1 \arrow[dr, "+"] & \\
& Q_3 \\
Q_2 \arrow[ur, "+"'] &
\end{tikzcd}$$

$$\begin{tikzcd}
T_1 \arrow[dr, "+"] & \\
& T_3 \\
T_2 \arrow[ur, "+"'] &
\end{tikzcd}$$

A problem specification: Water faucet

Structure:

?

Function:

$$\begin{tikzcd}
C_c \arrow[r, "+"] \arrow[dd, "+"] & Q_c \arrow[dr, "+"] \arrow[ddddr, "-"] & \\
& & Q_m \\
C_h \arrow[uu, "+"] \arrow[r, "+"'] & Q_h \arrow[ur, "+"] \arrow[ddr] & \\
& T_c \arrow[dr, "+"] & \\
& & T_m \\
& T_h \arrow[ur, "+"] \arrow[ur, "+"'] &
\end{tikzcd}$$

Case-Based Reasoning in CADET

- Instances represented by rich structural descriptions
- Multiple cases retrieved (and combined) to form solution to new problem
- Tight coupling between case retrieval and problem solving

Bottom line:

- Simple matching of cases useful for tasks such as answering help-desk queries
- Area of ongoing research

What you need to know

- Instance-based learning
  - non-parametric
  - trade decreased learning time for increased classification time
- Issues
  - appropriate distance metrics
  - curse of dimensionality
  - efficient indexing