Instance-Based Learning

(a.k.a. memory-based learning)
Part I: Nearest Neighbor Classification

Ronald J. Williams
CSG220
Spring 2007

Note to other teachers and users of these slides. Andrew would be delighted if you found this source material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. PowerPoint originals are available. If you make use of a significant portion of these slides in your own lecture, please include this message, or the following link to the source repository of Andrew's tutorials: http://www.cs.cmu.edu/~awm/tutorials . Comments and corrections gratefully received.

Containing selected slides adapted from the Andrew Moore tutorial with the same main title

1-Nearest-Neighbor Classification

Given: datapoints $(\boldsymbol{x}_1,y_1)$ $(\boldsymbol{x}_2,y_2)$..$(\boldsymbol{x}_N,y_N)$, where each $\boldsymbol{x}_i$ is an attribute vector and y_i is the corresponding class label, determined according to $y_i=f(\boldsymbol{x}_i)$ for some unknown function f (possibly corrupted by noise). Given query point $\boldsymbol{x}_q$, your job is to predict

$$\hat{y} = \hat{f}(\mathbf{x}_q).$$

Nearest Neighbor Classifier:

Find the closest $\boldsymbol{x}_i$ in the set of datapoints

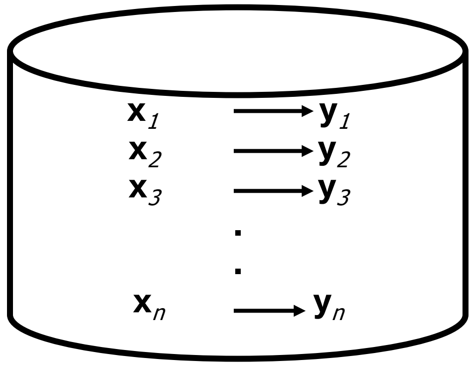
$$i(nn) = \arg\min_i d(\mathbf{x}_i, \mathbf{x}_q)$$

where d is some distance metric. Then classify $\boldsymbol{x}_q$ using

$$\hat{y} = y_{i(nn)}$$

1-Nearest-Neighbor is an example of….
Instance-based learning

A classifier and function approximator that has been around since about 1910.



To make a prediction, search database for similar datapoints, and predict based on these nearby points.

Four things make a memory-based learner:

- A distance metric
- How many nearby neighbors to look at?
- A weighting function (optional)
- How to fit with the local points?

Nearest Neighbor

Four things make a memory based learner:

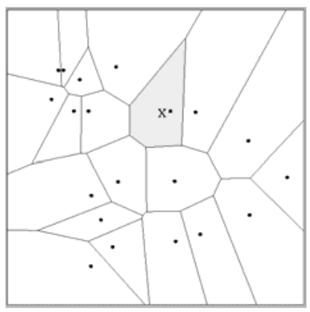
1. *A distance metric*
   Euclidian or related generalizations
2. *How many nearby neighbors to look at?*
   One
3. *A weighting function (optional)*
   N/A
4. *How to fit with the local points?*
   Just predict the same output as the nearest neighbor

Multivariate Distance Metrics

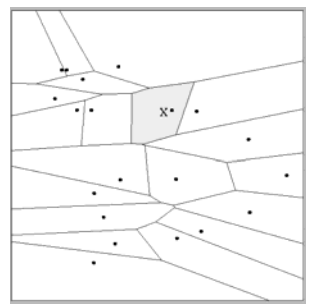
Suppose the input vectors x1, x2, …xn are two dimensional:

$\mathbf{x}_1 = (x_{11} , x_{12})$, $\mathbf{x}_2 = (x_{21} , x_{22})$, …$\mathbf{x}_N = (x_{N1} , x_{N2})$.

One can draw the nearest-neighbor regions in input space.



$d^2(\mathbf{x}_i,\mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2$

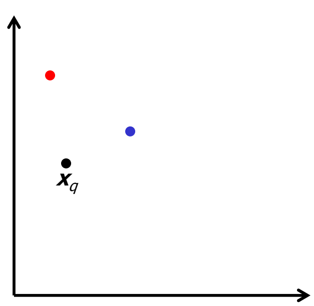


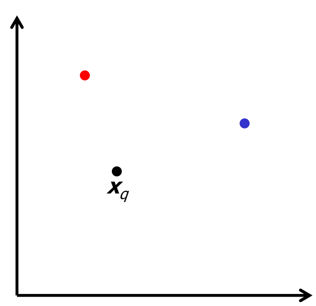
$d^2(\mathbf{x}_i,\mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (3x_{i2} - 3x_{j2})^2$

The relative scalings in the distance metric affect region shapes.

Effect of Axis Scaling

Which data point is closer to the query point?





Horizontal axis rescaled by a factor of 2

Euclidean Distance Metric

$$d(\mathrm{x},\mathrm{x}') = \sqrt{\sum_i \sigma_i^{\,2} \left(x_i - x'_i\right)^2}$$

Or equivalently,

$$d(\mathrm{x},\mathrm{x}') = \sqrt{(\mathrm{x} - \mathrm{x}')^T \sum (\mathrm{x} - \mathrm{x}')}$$

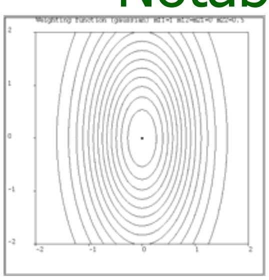
where

$$\sum = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma_N^2 \end{bmatrix}$$

Other Metrics…

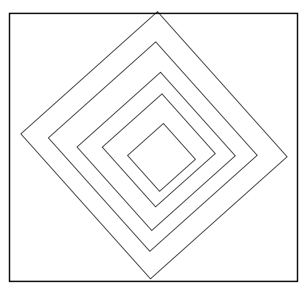
- Mahalanobis, Rank-based, Correlation-based (Stanfill+Waltz, Maes' Ringo system…)

Notable Distance Metrics

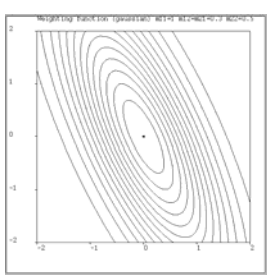


Scaled Euclidian (L_2)

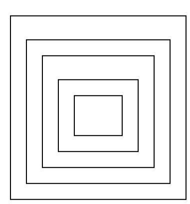
L_1 norm (absolute)



Mahalanobis (here, Σ on the previous slide is not necessarily diagonal, but is symmetric



$L_{infinity}$ *(max) norm*



Distance Metric: Things to Worry about

- In practice, should at least rescale every axis to have approximately the same range
  - Gives every feature more nearly equal weight
  - Can use cross-validation to fine-tune axis weightings or other metric parameters
- But try to eliminate irrelevant features
  - Many irrelevant features could dominate the distance measure and create a bad classifier
  - Cross-validation can be very helpful with this

What about noise?

- If each data point has a noisy class label, what would be better than using only the single nearest neighbor? …

k-Nearest Neighbor

Four things make a memory based learner:

1. *A distance metric*
   Euclidian or related generalizations
2. *How many nearby neighbors to look at?*
   k
3. *A weighting function (optional)*
   N/A
4. *How to fit with the local points?*
   Just take the majority vote among the k nearest neighbors

Weighted Nearest Neighbor Classifier

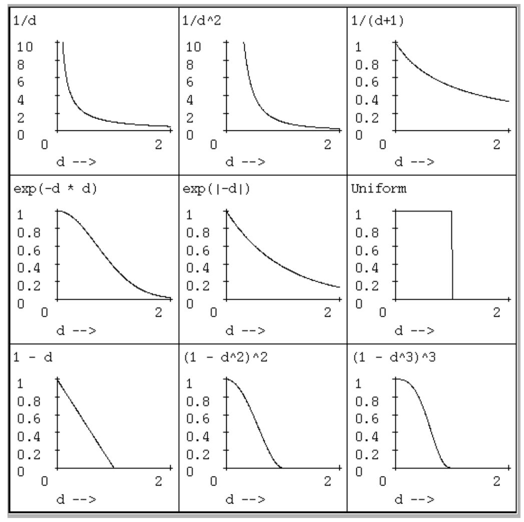
Four things make a memory based learner:

1. *A distance metric*
   Euclidian
2. *How many nearby neighbors to look at?*
   Potentially all of them
3. *A weighting function (optional)*
   Nearby points to the query are weighted strongly, far points weakly.
4. *How to fit with the local points?*
   Give each point a vote based on this weighting function. Classify according to this vote.

Weighting functions

Let $d = d(x_i, x_{query})$

Then here are some commonly used weighting functions…



Categorical Attributes

- If attribute $x[j]$ can take on one of several discrete values $v_1, v_2, \ldots, v_k$ (and there is no reason to consider some pairs of these values as being "closer" than some other pairs), then a sensible interpretation of

$$x_i[j] - x_q[j]$$

in the distance computation is

$$0 \text{ if } x_i[j] = x_q[j] \text{ or}$$
$$1 \text{ if } x_i[j] \neq x_q[j]$$

- This has a similar effect as when a 1-out-of-k encoding is used for input to a neural network

