Instance Based Learning

Alexander Skoglund
Machine Learning Course
AASS, April 2005
Instance Based Learning

- Nearest neighborhood
- Locally weighted regression
- Radial basis functions
- Case-based reasoning
- Lazy and eager learning
Main Idea of Nearest Neighbor

- The main idea is to store all training data in a data base
- When a prediction is needed we make a query and then use a clever way to get an approximation
- **Benefits:** training is very fast, can learn complex functions
- **Drawback:** the query can be very computationally heavy (i.e., large data set)
Nearest Neighbor

- Assume we have a set of data: \{a_1(x), a_2(x), ..., a_n(x)\}

- We then want to know the distance (how close) two values are to each other:

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

1-Nearest Neighbor:
- Given query instance $x_q$, first locate the nearest training example $x_n$, then estimate

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

Left: 5-NN classifies $x_q$ as negative
Right: decision surface of 1-NN
$k$-Nearest Neighbor:

- **Discreet case:** given $x_q$, take a vote among it $k$-nearest neighbors. Select the most common one.
- **Real value:** take the mean of $f$ values of $k$ nearest neighbors:

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

We can also introduce weights:
• Point close to the query -> high weights
• Point far from the query -> low weights

Extend the formula:

\[
\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}
\]
Remarks on NN-algorithm

- Robust to noise, smooth data
- Suffers from the curse of dimensionality: we can remedy this by using cross-validation:
  "extend" axis with high relevance
  "shorten" less relevant axis
- Other techniques: Principal Component Analysis or Singular Value Decomposition)
Terminology

- **Regression**: approximation of real valued function
- **Residual function**: the error in the approximation
- **Kernel function**: the distance function, i.e., the function $K$ such that:

$$w = K(d(x_i, x_q))$$
Locally Weighted Regression

LWR, a generalization of the previous topic. LWR makes a local approximation around $x_q$. It can use a linear, quadratic or some other kind of functional form.

- Locally: model around $x_q$
- Weighted: weighting according to distance
- Regression: statistical term for approximation
Locally Weighted Linear Regression

- Squared error over $k$-nearest neighbors
  \[
  E_1(x_q) = \frac{1}{2} \sum_{x \in k\text{-nearest nbrs of } x_q} (f(x) - \hat{f}(x))^2
  \]

- Distance weighted squared error over all nbrs
  \[
  E_2(x_q) = \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 K(d(x_q, x))
  \]

- Or a combination of the above
Radial Basis Functions

- RBF, a combination of LWR and ANN
  \[ \hat{f}(x) = w_0 + \sum_{u=1}^{k} w_u K_u(d(x_u, x)) \]

- Where K, the Kernel function, usually is a Gaussian function:
  \[ K_u(d(x_u, x)) = e^{-\frac{1}{2s_u^2}d^2(x_u, x)} \]
Overview of RBF

- Is a global approximation to target function, in terms of linear combination of local approximation
- Used for classification
- A different kind on neural network
- Eager instead of lazy
RBF layout

- $K_u(d(x_u, x)) = e^{-\frac{1}{2s_u^2}d^2(x_u, x)}$

  can be seen as a two layer network

- Design steps:
  - determine number of hidden units
  - Train weights using global EM
Training of RBF networks

- Determine number of hidden units, $k$

- Each unit, $u$, is defined by selecting $x_u$ and $\sigma^2_u$, this defines $K_u(d(x_u,x))$

- Train weights to fit the training data using

$$E \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2$$
Alternative approaches

- Allocate a kernel function for each training example, all with same $\sigma^2$.

- Allocate a set of kernel functions that is smaller than the number of training examples.
  - Centers may be spread uniformly over space or,
  - Identify clusters of instances, and add a kernel function centered around each cluster.
Case-Based Reasoning

Shared properties:
- Lazy, they postpone the generalization until later
- Classification is based on similarity,
- Instances are represented by a real-values in n-dimensional space
- Case Based Reasoning (CBR) include the first two properties but not the third
Case-Based Reasoning

- Uses symbolic descriptions, we need a measure of the distance between two instances (instead of the Euclidean distance)

Some properties that differs from NN:
- Instances are represented as symbolic descriptions
- Use knowledge-based reasoning instead of statistical methods for combining different instances
A case study: CADET

Containing approx. 75 mechanical devices:

- Each instance is the structure and function
- Query for the desired function
- Target is the structure of this function
- Closest functional description is chosen
- Distance metric: match qualitative function description
- If no exact match exists, use subgraphs instead
Lazy and Eager Learning

- **Lazy**: generalizes when given a query
  - k-Nearest Neighbor, LWR, CBR...
- **Eager**: generalize during training
  - RBF, ID3, Back propagation, etc...
- **Eager learner creates a global approximation**
- **Lazy learner can create many local approximations**
- **Lazy can represent more complex functions (given the same H), and we do not risk to “unlearn” distant data**