Machine Learning: Lecture 7

Instance-Based

Learning (IBL)

(Based on Chapter 8 of Mitchell T.., Machine Learning, 1997)

General Description

- IBL methods learn by simply storing the presented training data.
- 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.
- IBL approaches can construct a different approximation to the target function for each distinct query. They can construct ***local*** rather than ***global*** approximations.
- IBL methods can use complex symbolic representations for instances. This is called ***Case-Based Reasoning (CBR).***

Advantages and Disadvantages of IBL Methods

Advantage: IBL Methods are particularly well suited to problems in which the target function is very complex, but can still be described by a collection of less complex local approximations.

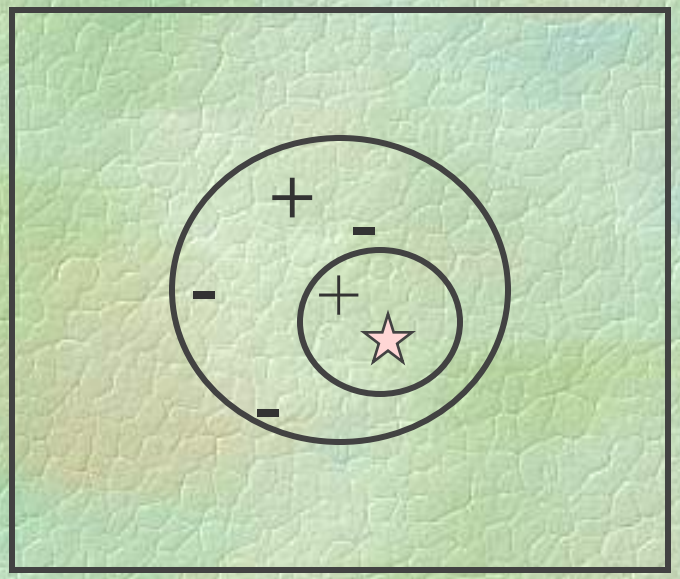
Disadvantage I: The cost of classifying new instances can be high (since most of the computation takes place at this stage).

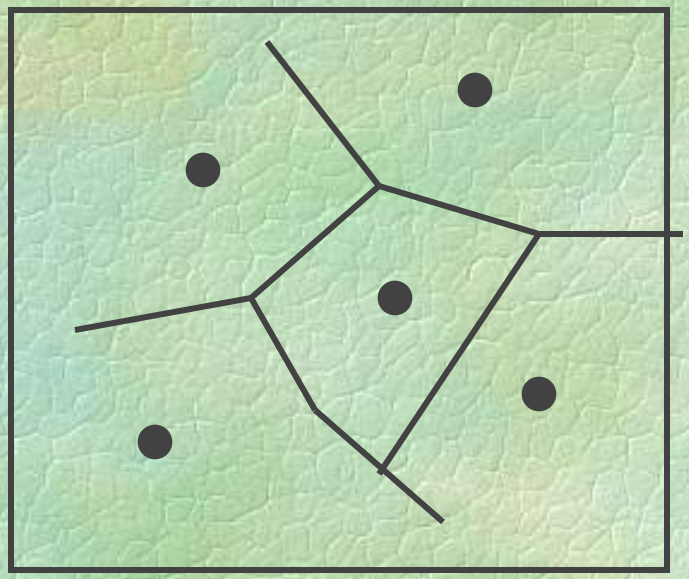
Disadvantage II: Many IBL approaches typically consider all attributes of the instances ==> they are very sensitive to the curse of dimensionality!

k-Nearest Neighbour Learning

- **Assumption:** All instances, x, correspond to points in the n-dimensional space $\mathbf{R}^n$. $x = \langle a_1(x), a_2(x) \ldots a_n(x) \rangle$.
- **Measure Used:** ***Euclidean Distance:***
  $d(x_i,x_j) = \sqrt{\Sigma_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$
- **Training Algorithm:**
  - For each training example $\langle x, f(x) \rangle$, add the example to the list ***training_examples***.
- **Classification Algorithm:** Given a query instance x_q to be classified:
  - Let $x_1 \ldots x_k$ be the k instances from training_examples that are nearest to x_q.
  - Return $\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \Sigma_{r=1}^{n} \delta(v, f(x_i))$
  - where $\delta(a,b)=1$ if $a=b$ and $\delta(a,b)=0$ otherwise.

Example





: query, x_q

1-NN: +

5-NN: -

Decision Surface for 1-NN

Distance-Weighted Nearest Neighbour

- k-NN can be refined by weighing the contribution of the k neighbours according to their distance to the query point x_q, giving greater weight to closer neighbours.
- To do so, replace the last line of the algorithm with

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{r=1}^{n} w_i \delta(v, f(x_i))$$

where $w_i = 1/d(x_q, x_i)^2$

Remarks on k-NN

- k-NN can be used for ***regression*** instead of classification.
- k-NN is ***robust to noise*** and, it is generally quite a good classifier.
- k-NN's ***disadvantage*** is that it uses all attributes to classify instances
  - **Solution 1:** weigh the attributes differently (use cross-validation to determine the weights)
  - **Solution 2:** eliminate the least relevant attributes (again, use cross-validation to determine which attributes to eliminate)

Locally Weighted Regression

- Locally weighted regression generalizes nearest-neighbour approaches by constructing an explicit approximation to f over a local region surrounding x_q.
- In such approaches, the contribution of each training example is weighted by its distance to the query point.

An Example: Locally Weighted Linear Regression

- f is approximated by: $\hat{f}(x)=w_0+w_1a_1(x)+\ldots+w_na_n(x)$
- Gradient descent can be used to find the coefficients $w_0, w_1,\ldots w_n$ that minimize some error function.
- The error function, however, should be different from the one used in the Neural Net since we want a ***local solution***. **Different possibilities:**
  - Minimize the squared error over just the ***k*** nearest neighbours.
  - Minimize the squared error over the entire training set but weigh the contribution of each example by some decreasing function ***K*** of its distance from x_q.
  - Combine 1 and 2

Radial Basis Function (RBF)

- **Approximating Function:**

$$\hat{f}(x) = w_0 + \sum_{u=1}^{k} w_u K_u(d(x_u, x))$$

- $K_u(d(x_u, x))$ is a ***kernel function*** that decreases as the distance $d(x_u, x)$ increases (e.g., the Gaussian function); and k is a user-defined constant that specifies the number of kernel functions to be included.
- Although $\hat{f}(x)$ is a ***global*** approximation to $f(x)$ the contribution of each kernel function is ***localized***.
- RBF can be implemented in a neural network. It is a very efficient two step algorithm:
  - Find the parameters of the kernel functions (e.g., use the EM algorithm)
  - Learn the linear weights of the kernel functions.

Case-Based Reasoning (CBR)

- CBR is similar to k-NN methods in that:
  - They are ***lazy*** learning methods in that they defer generalization until a query comes around.
  - They classify new query instances by analyzing similar instances while ignoring instances that are very different from the query.
- However, CBR is different from k-NN methods in that:
  - They do not represent instances as real-valued points, but instead, they use a ***rich symbolic*** representation.
- CBR can thus be applied to complex conceptual problems such as the design of mechanical devices or legal reasoning

Lazy versus Eager Learning

- ***Lazy methods***: k-NN, locally weighted regression, CBR
- ***Eager methods***: RBF + all the methods we studied in the course so far.
- **Differences in Computation Time:**
  - Lazy methods learn quickly but classify slowly
  - Eager methods learn slowly but classify quickly
- **Differences in Classification Approaches:**
  - Lazy methods search a larger hypothesis space than eager methods because they use many different local functions to form their implicit global approximation to the target function. Eager methods commit at training time to a single global approximation.