Instance Based Learning

Based on "Machine Learning", T. Mitchell, McGRAW Hill, 1997, ch. 8

Acknowledgement:

The present slides are an adaptation of slides drawn by T. Mitchell

Key ideas:

training: simply store all training examples

classification: compute only locally the target function

inductive bias:

the classification of query/test instance x_q will be most similar to the classification of training instances that are nearby

Advantages:

can learn very complex target functions
training is very fast
don't lose information
robust to noisy training

Disadvantages:

slow at query time
easily fooled by irrelevant attributes

Methods

1. **k-Nearest Neighbor;**
   Distance-weighted k-NN
2. **A generalization of k-NN:**
   Locally weighted regression
3. **Combining instance-based learning and neural networks:**
   Radial basis function networks

1. k-Nearest Neighbor Learning

[**E. Fix,J. Hodges, 1951**]

Training:

Store all training examples

Classification:

Given a query/test instance x_q,
first locate the k nearest training examples $x_1, \ldots, x_k$,
then estimate $\hat{f}(x_q)$:

- in case of discrete-valued $f : \Re^n \to V$,
  take a vote among its k nearest neighbors

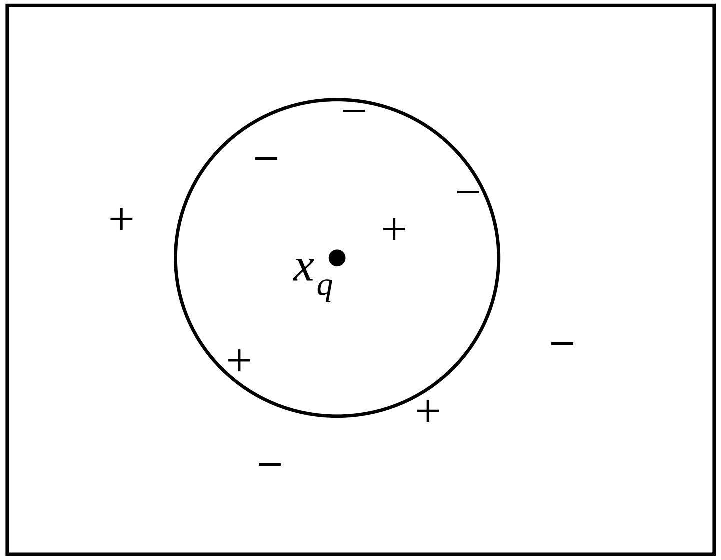
$$\hat{f}(x_q) \leftarrow \operatorname*{argmax}_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$$

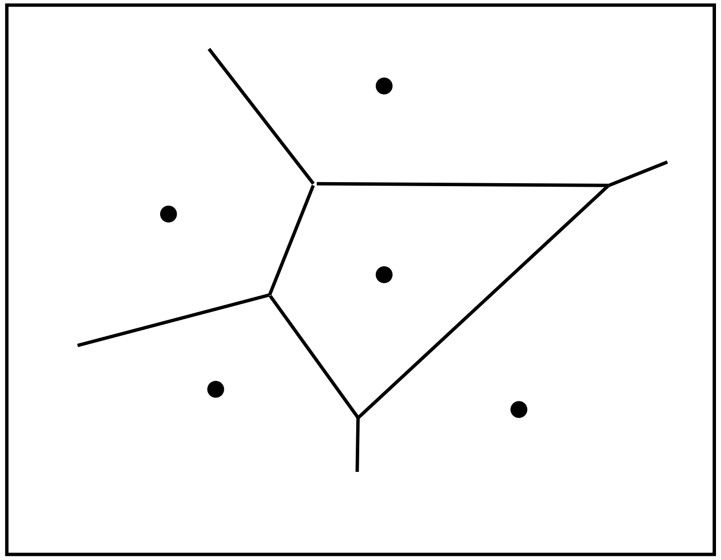
  where $\delta(a, b) = 1$ if $a = b$, and $\delta(a, b) = 0$ if $a \neq b$

- in case of continuous-valued f,
  take the mean of the f values of its k nearest neighbors

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

Illustratring k-NN; Voronoi Diagram





Note that
1-NN classifies x_q as $+$
5-NN classifies x_q as $-$

Above: The decision surface induced by 1-NN for a set of training examples.

The convex polygon surrounding each training example indicates the region of the instance space closest to that examples;

1-NN will assign them the same classification as the corresponding training example.

When To Consider k-Nearest Neighbor

- instances map to points in $\Re^n$
- less than 20 attributes per instance
- lots of training data

Efficient memory indexing for the retrieval of the nearest neighbors

kd-trees ([Bentley, 1975] [Friedman, 1977])

Each leaf node stores a training instance. Nearby instances are stored at the same (or nearby) nodes.

The internal nodes of the tree sort the new query x_q to the relevant leaf by testing selected attributes of x_q.

k-NN: The Curse of Dimensionality

Note: **k-NN is easily mislead when X is highly-dimensional, i.e. irrelevant attributes may dominate the decision!**

Example:

Imagine instances described by $n = 20$ attributes, but only 2 are relevant to the target function. Instances that have identical values for the 2 attributes may be distant from x_q in the 20-dimensional space.

Solution:

- **Stretch the j-th axis by weight z_j, where $z_1, \ldots, z_n$ are chosen so to minimize the prediction error.**
- **Use an approach similar to cross-validation to automatically choose values for the weights $z_1, \ldots, z_n$ (see [Moore and Lee, 1994]).**
- **Note that setting z_j to zero eliminates this dimension altogether.**

A k-NN Variant: Distance-Weighted k-NN

We might want to weight nearer neighbors more heavily:

- **for discrete-valued f:** $\hat{f}(x_q) \leftarrow \text{argmax}_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$

  where

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

  $d(x_q, x_i)$ **is the distance between** x_q **and** x_i
  but if $x_q = x_i$ **we take** $\hat{f}(x_q) \leftarrow f(x_i)$

- **for continuous-valued f:** $\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$

Remark: **Now it makes sense to use *all* training examples instead of just k. In this case k-NN is known as Shepard's method (1968).**

A link to Bayesian Learning (Ch. 6)

k-NN: Behavior in the Limit

Let $p(x)$ be the probability that the instance x will be labeled 1 (positive) versus 0 (negative).

k-Nearest neighbor:

- If the number of training examples $\rightarrow \infty$ and k gets large, k-NN approaches the Bayes optimal learner.

  Bayes optimal: if $p(x) > 0.5$ then predict 1, else 0.

Nearest neighbor ($k = 1$):

- If the number of training examples $\rightarrow \infty$, 1-NN approaches the Gibbs algorithm.

  Gibbs algorithm: with probability $p(x)$ predict 1, else 0.

2. Locally Weighted Regression

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

Why not form an explicit approximation $\hat{f}(x)$ for the region surrounding x_q:

- **Fit a linear function (or: a quadratic function, a multi-layer neural net, etc.) to k nearest neighbors**

  $$\hat{f} = w_0 + w_1 a_1(x) + \ldots + w_n a_n(x)$$

  where $a_1(x), \ldots, a_n(x)$ are the attributes of the instance x.

- **Produce a "piecewise approximation" to f, by learning $w_0, w_1, \ldots, w_n$**

Minimizing the Error in Locally Weighted Regression

- Squared error over k nearest neighbors

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

- Distance-weighted squared error over all neighbors

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

  where the "kernel" function K decreases over $d(x_q, x)$

- A combination of the above two:

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in \ k \ nearest \ nbrs \ of \ x_q} (f(x) - \hat{f}(x))^2 \ K(d(x_q, x))$$

  In this case, applying the gradient descent method, we obtain the training rule $w_j \leftarrow w_j + \Delta w_j$, where

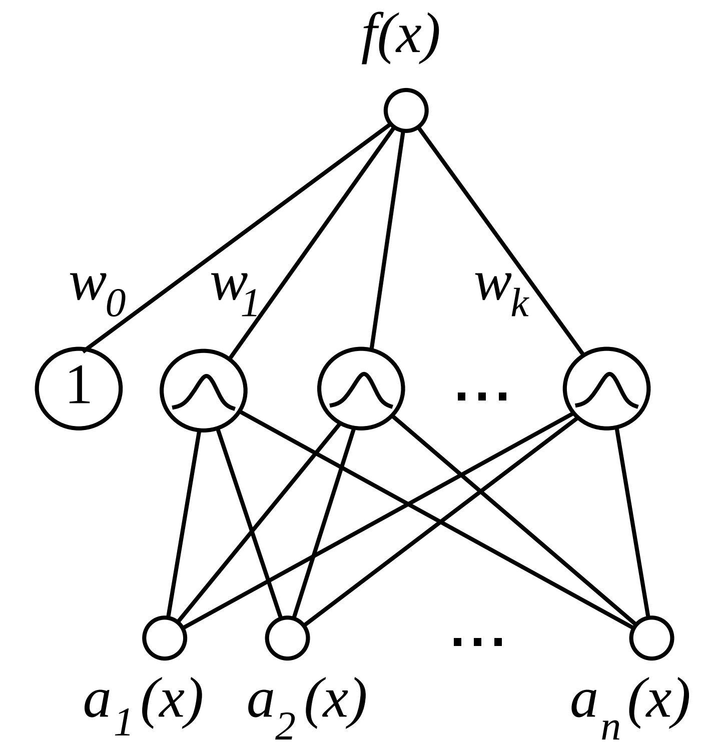
$$\Delta w_j = \eta \sum_{x \in \ k \ nearest \ nbrs \ of \ x_q} K(d(x_q, x))(f(x) - \hat{f}(x)) a_j(x)$$

Combining instance-based learning and neural networks:
3. Radial Basis Function Networks

- Compute a global approximation to the target function f, in terms of linear combination of local approximations (“kernel” functions).
- Closely related to distance-weighted regression, but “eager” instead of “lazy”.
- Can be thought of as a different kind of (two-layer) neural networks:

  The hidden units compute the values of kernel functions. The output unit computes f as a liniar combination of kernel functions.
- Used, e.g. for image classification, where the assumption of spatially local influencies is well-justified.

Radial Basis Function Networks



a_i are the attributes describing the instances.

Target function:

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

The kernel functions are commonly chosen as Gaussians:

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

The activation of hidden units will be close to 0 unless x is close to x_u.

As it will be shown on the next slide, the two layers are trained separately (therefore more efficiently than in NNs).

Training Radial Basis Function Networks

Q1: What x_u to use for each kernel function $K_u(d(x_u, x))$:

- use the training instances;
- or scatter them throughout the instance space, either uniformly or non uniformly (reflecting the distribution of training instances);
- or form prototypical clusters of instances, and take one K_u centered at each cluster.
  We can use the EM algorithm (see Ch. 6.12) to automatically choose the mean (and perhaps variance) for each K_u.

Q2: How to train the weights:

- hold K_u fixed, and train the linear output layer to get w_i

Theorem
[Hartman *et al.*, 1990]

The function f can be approximated with arbitrarily small error, provided

- **a sufficiently large k, and**
- **the width σ_u^2 of each kernel K_u can be separately specified.**

Remark

Instance-based learning was also applied to instance spaces $X \neq \Re^n$, usually with rich symbolic logic descriptions. Retrieving similar instances in this case is much more elaborate. It is

This learning method, known as Case-Based Reasoning, was applied for instance to

- conceptual design of mechanical devices,
  based on a stored library of previous designs
  [Sycara, 1992]
- reasoning about new legal cases,
  based on previous rulings
  [Ashley, 1990]
- scheduling problems,
  by reusing/combining portions of solutions to similar problems
  [Veloso, 1992]