

Intro. to machine learning (CSI 5325)

Lecture 18: Instance-based learning

Greg Hamerly

Some content from Tom Mitchell.

1 k -Nearest Neighbor

Instance-Based Learning

Key idea: just store all training examples $\langle x_i, f(x_i) \rangle$

Nearest neighbor algorithm:

- Given query instance x_q , first locate nearest training example x_i , then estimate $\hat{f}(x_q) \leftarrow f(x_i)$

This is also called the '1-nearest neighbor' algorithm.

Using multiple neighbors

Why not use more than just one neighbor?

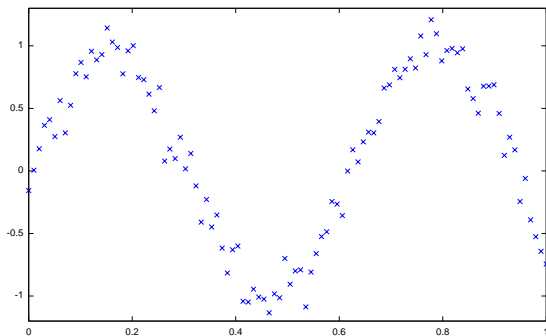
k -Nearest neighbor:

- Given x_q , take vote among its k nearest neighbors (if discrete-valued target function)
- Let $N(x_q, k)$ be the indexes of the k nearest neighbors of x_q
- Use the nearest neighbors to predict x_q

Regression applications

Regression: mean of f values over k nearest neighbors:

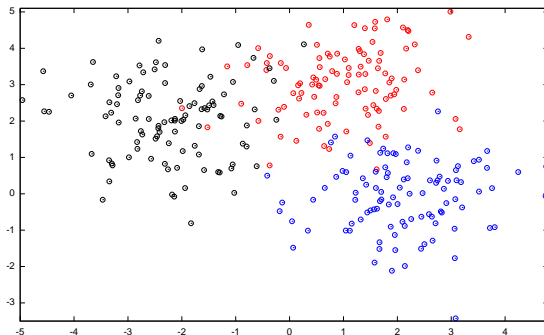
$$\hat{f}(x_q) \leftarrow \frac{1}{k} \sum_{i \in N(x_q, k)} f(x_i)$$



Classification applications

Classification: most popular f value over k nearest neighbors:

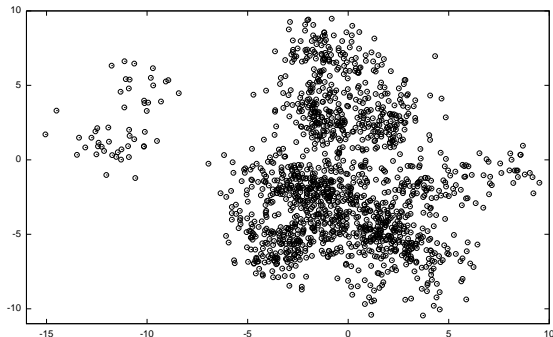
$$\hat{f}(x_q) \leftarrow \underset{i \in N(x_q, k)}{\text{mode}} f(x_i)$$



Probability density applications

$$\hat{f}(x_q) \leftarrow \frac{k-1}{nV(k, x_q, d)}$$

n is the number of examples in the dataset, and V is the volume of the smallest d -sphere centered at x_q that covers the k neighbors



Similarity and p -norms

How do we compare two points for similarity?

→ Similarity $\propto 1/\text{distance}$

Common distance metrics for vectors a and b :

- Euclidean:

$$d(a, b) = \sqrt{\sum_i (a_i - b_i)^2}$$

- Manhattan (aka city-block):

$$d(a, b) = \sum_i |a_i - b_i|$$

- General (Minkowski) p -norm:

$$d_p(a, b) = \left(\sum_i |a_i - b_i|^p \right)^{1/p}$$

When To Consider Using Nearest Neighbor

- Instances map to points in \mathcal{R}^d
- Less than 20 attributes per instance
- Lots of training data

Advantages:

- Training is very fast
- Learn complex target functions
- Don't lose information

Disadvantages:

- Slow at query time
- Easily fooled by irrelevant attributes

Classifier behavior in the Limit

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

Nearest neighbor:

- As $n \rightarrow \infty$, approaches Gibbs Algorithm
Gibbs: with probability $p(x)$ predict 1, else 0

k -Nearest neighbor:

- As $n \rightarrow \infty$ and k gets large, approaches Bayes optimal
Bayes optimal: if $p(x) > .5$ then predict 1, else 0

Recall Gibbs has at most twice the expected error of Bayes optimal

Bias and variance in k -NN

What is the inductive bias of k -NN?

What is the *statistical* bias and *statistical* variance of k -NN?

- In particular, how do they vary with k ?
- Think about extreme k values, e.g. $k = 1$ or $k = n$.

Distance-Weighted k -NN

Might want to weight nearer neighbors more heavily...

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i \in N(x_q, k)} w_i f(x_i)}{\sum_{i \in N(x_q, k)} w_i}$$

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: we could use *all* training examples instead of just k

→ This is called Shepard's method

Curse of Dimensionality

Imagine instances described by 20 attributes, but only 2 are relevant to target function

Nearest neighbor is easily misled when high-dimensional X (a symptom of the 'curse of dimensionality')

Why?

Can we fix this problem?

Adapting distance metrics

When not all attributes are equally relevant, we can give them different weights – equivalent to stretching their ranges (see Moore and Lee 1994).

- Stretch j th axis by weight z_j , where z_1, \dots, z_d chosen to minimize prediction error
- Use cross-validation to automatically choose weights z_1, \dots, z_d
- Note setting z_j to zero eliminates this dimension altogether

There has been a lot of recent work on learning distance metrics (Yang 2006, Davis et al. 2008, etc.)