K nearest neighbor

LING 572
Fei Xia
Week 2: 1/14/2010

Outline

- Demo
- kNN

Demo

- ML algorithms: Naïve Bayes, Decision stump, boosting, bagging, SVM, etc.

- Task: A binary classification problem with only two features.

- http://www.cs.technion.ac.il/~rani/LocBoost/

The term “weight” in ML

Some Xs are more important than others given everything else in the system

- Weights of features
- Weights of instances
- Weights of classifiers

The term “binary” in ML

- Classification problem
  - Binary: the number of classes is 2
  - Multi-class: the number is classes is > 2

- Features
  - Binary: the number of possible feature values is 2.
  - Real-valued: the feature values are real numbers

- File format:
  - Binary: human un-readable
  - Text: human readable

kNN

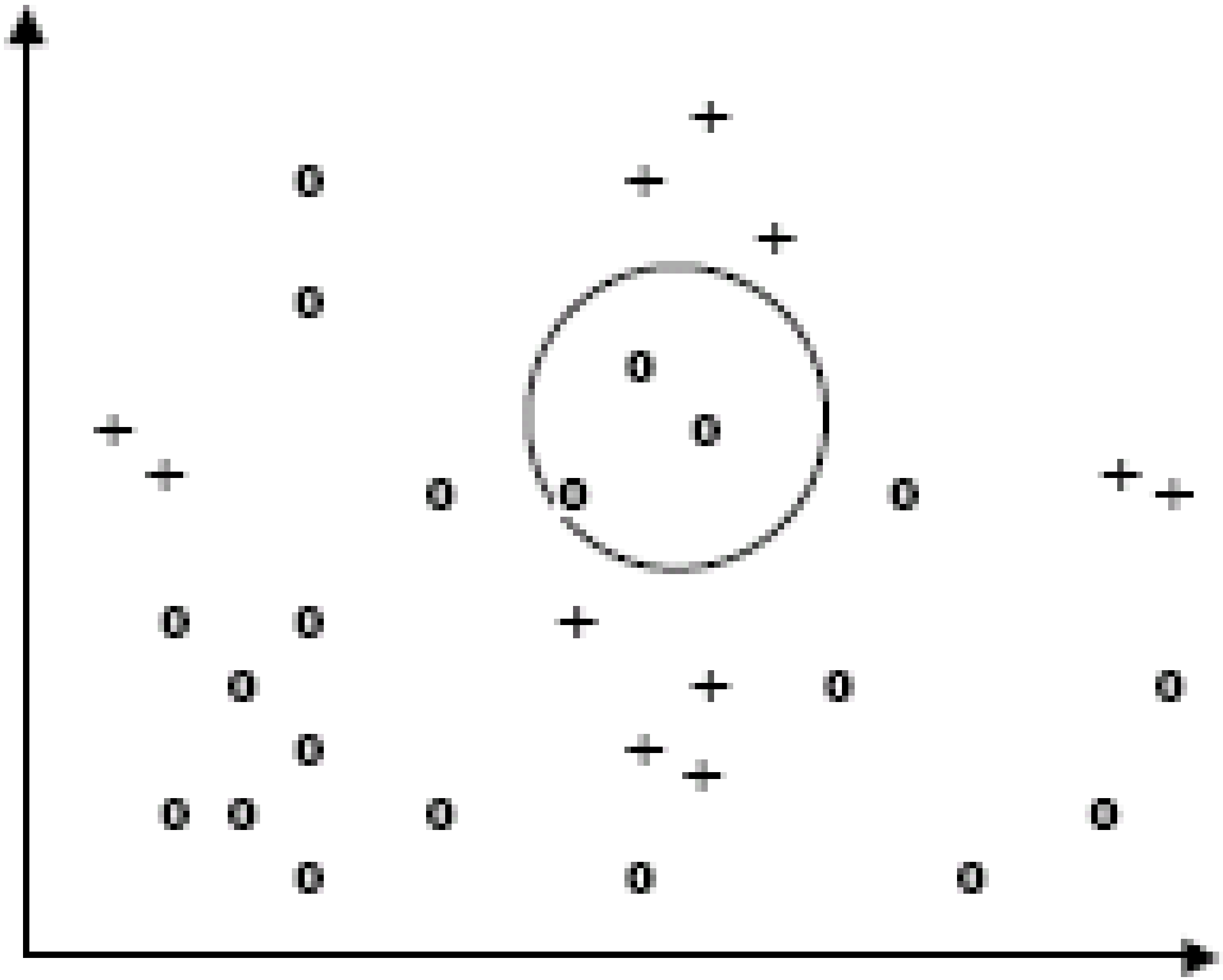
Instance-based (IB) learning

- No training: store all training instances.
  ➔ "Lazy learning"

- Examples:
  – kNN
  – Locally weighted regression
  – Radial basis functions
  – Case-based reasoning
  – …

- The most well-known IB method: kNN

kNN



kNN

- For a new instance d,
  - find k training instances that are closest to d.
  - perform majority voting or weighted voting.

- Properties:
  - A “lazy” classifier. No training.
  - Feature selection and distance measure are crucial.

The algorithm

- Determine parameter K
- Calculate the distance between query-instance and all the training instances
- Sort the distances and determine K nearest neighbors
- Gather the labels of the K nearest neighbors
- Use simple majority voting or weighted voting.

Picking K

- Use the validation data: pick the one that minimizes cross validation error.
  - Training data: true training data and validation data
  - Dev data
  - Test data

- N-fold cross validation:

Normalizing attribute values

- Distance could be dominated by some attributes with large numbers:
  - Ex: features: age, income
  - Original data: x_1=(35, 76K), x_2=(36, 80K), x_3=(70, 79K)
  - Assume: age $\in$ [0,100], income $\in$ [0, 200K]
  - After normalization: x_1=(0.35, 0.38), x_2=(0.36, 0.40), x_3 = (0.70, 0.395).

The Choice of Features

- Imagine there are 100 features, and only 2 of them are relevant to the target label.

- kNN is easily misled in high-dimensional space.

➔Feature weighting or feature selection
(It will be covered in Week #4)

Feature weighting

- Stretch j-th axis by weight w_j

- Use cross-validation to automatically choose weights $w_1, \ldots, w_n$

- Setting w_j to zero eliminates this dimension altogether.

Some similarity measures

- Euclidean distance:

$$dist(d_i, d_j) = \sqrt{\sum_k (a_{i,k} - a_{j,k})^2}$$

- Weighted Euclidean distance:

$$dist(d_i, d_j) = \sqrt{\sum_k w_k (a_{i,k} - a_{j,k})^2}$$

- Cosine

$$cos(d_i, d_j) = \frac{\sum_k a_{i,k} a_{j,k}}{\sqrt{\sum_k a_{i,k}^2}\sqrt{\sum_k a_{j,k}^2}}$$

Voting by k-nearest neighbors

- Suppose we have found the k-nearest neighbors.

- Let f_i (x) be the class label for the i-th neighbor of x.

  $\delta(c, f_i(x))$ is the identity function; that is,
  it is 1 if $f_i(x) = c$, and is 0 otherwise.

  Let $g(c) = \sum_i \delta(c, f_i(x))$; that is, g(c) is the number of neighbors with label c.

Voting

- Majority voting:
  $c^* = \arg\max_c g(c)$

- Weighted voting: weighting is on each neighbor
  $c^* = \arg\max_c \sum_i w_i \, \delta(c, f_i(x))$

- Weighted voting allows us to use more training examples:

  e.g., $w_i = 1/\mathrm{dist}(x, x_i)$

  ➔ We can use all the training examples.

Summary of kNN algorithm

- Decide k, feature weights, and similarity measure
- Given a test instance x
  - Calculate the distances between x and all the training data
  - Choose the k nearest neighbors
  - Let the neighbors vote

- Strengths:
  - Simplicity (conceptual)
  - Efficiency at training: no training
  - Handling multi-class
  - Stability and robustness: averaging k neighbors
  - Predication accuracy: when the training data is large

- Weakness:
  - Efficiency at testing time: need to calc all distances
  - Theoretical validity
  - It is not clear which types of distance measure and features to use.

=> Extension: e.g., Rocchio algorithm