Instance-Based Learning

Instance-Based Learning

- Unlike most learning algorithms, *case-based*, also called *exemplar-based* or *instance-based*, approaches do not construct an abstract hypothesis but instead base classification of test instances on similarity to specific training cases. (e.g. Aha et al. (1991))
- Training is typically very simple: Just store the training instances.
- Generalization is postponed until a new instance must be classified. Therefore, case-based methods are sometimes called "lazy" learners. (Aha,1997)
- Given a new instance, its relationship to the stored examples is examined in order to assign a target function value for the new instance.
- In the worst case, testing requires comparing a test instance to every training instance. As a result, case-based methods may require more expensive indexing of cases during training to allow for efficient retrieval.
- Rather than estimating the target function for the entire instance space, case-based methods estimate the target function locally and differently for each new instance to be classified.

Distance Metrics

- Case-based methods assume a function for calculating the (dis)similarity of two instances.

- For continuous feature vectors, just use Euclidean distance

$$d(X, Y) = \sqrt{\sum_{i=1}^{n} (X_{f_i} - Y_{f_j})^2}$$

- For discrete features, just assume distance between two values is 0 if they are the same, 1 if different (e.g. Hamming distance).

- To compensate for differences in units, scale all continuous values to normalize their values to be between 0 and 1.

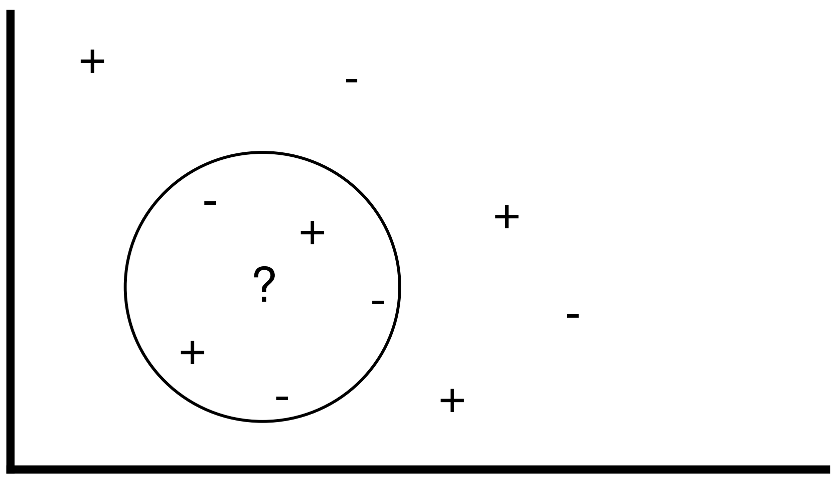
- Can develop specialized distance metric to account for relationships that exist among feature values. E.g. when comparing parts-of-speech feature, a proper noun and common noun are probably more similar than a proper noun and a verb.

K-Nearest Neighbor

(Cover & Hart, 1967; Duda & Hart, 1973)

- Calculate the distance between a test instance and every training instance.

- Pick the *k* closest training examples and assign the test example to the most common category among these "nearest neighbors"



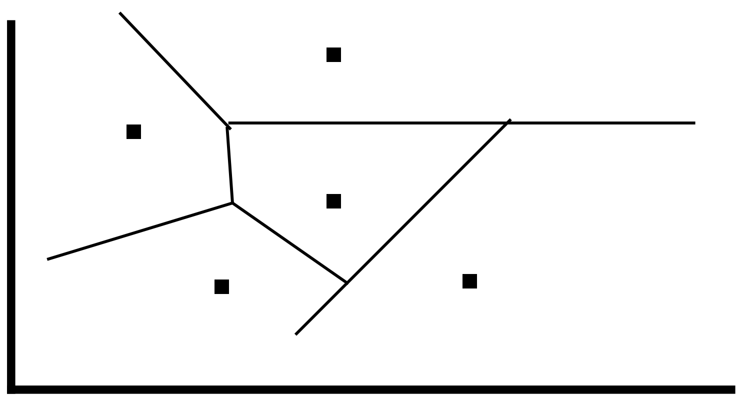
1-nearest neighbor says +, 5-nearest neighbor says -

- Voting multiple neighbors helps increase resistance to noise. For binary classification tasks, odd values of *k* are normally used to avoid ties. *k*=1,3,5 most common values used.

Implicit Classification Function

- Although it is not necessary to explicitly calculate it, the learned classification rule is based on the regions of feature space closest to each training example.

- For 1-nearest neighbor, the *Voronoi diagram* gives the complex polyhedra that segment the space into the region of points closest to each training example.



Nearest Neighbor Variations

- Since linear search is not a very efficient classification procedure, a data structure called a *k-d tree* can be used to index training examples and find nearest neighbors in logarithmic time on average. Nodes branch on threshold tests on individual features and leaves terminate at nearest neighbors. (Bentley, 1975; Friedman et al., 1977)

- *K*-nearest neighbor can be used to approximate the value of a continuous function (regression) by taking the average function value of the *k* nearest neighbors. (Atkeson et al., 1997; Bishop, 1995)

- All training examples can be used to contribute to the classification by giving every example a vote that is weighted by the inverse square of its distance from the test example. (Shepard, 1968)

Feature Relevance

- The standard distance metric weights each feature equally, which can cause problems if only a few of the features are relevant to the classification task, since the method could be mislead by similarity along many irrelevant dimensions.
- *Wrapper* methods for feature selection generate a set of candidate features, run the induction algorithm with these features, use the accuracy of the resulting concept description to evaluate the feature set. (John et al., 1994)
- *Filter* methods for feature selection consider attributes independently of the induction algorithm that will use them.
- *Global* feature weighting methods compute a single weight vector for the classification task.
- *Local* feature weighting methods allow feature weights to vary for each training instance, for each test instance, or both. (Wettschereck et al.,,1997)

Wrapper Methods for Feature Selection

- Generate a set of candidate features, run the induction algorithm with these features, use the accuracy of the resulting concept description to evaluate the feature set.
- *Forward selection.* Start with no features and successively add attributes. Continue until performance degrades.
- *Backward elimination.* Start with all features and successively remove attributes. Continue until performance degrades.
- General method for feature selection in that it can be used in conjunction with any induction algorithm.
- Disadvantages: Computational cost.

Filter Methods
for Feature Selection and Weighting

- Consider attributes independently of the induction algorithm that will use them.

- *Decision trees for feature selection.* In addition to storing training cases in a case base, use them to induce a decision tree. Features that do not appear in the decision tree are considered irrelevant for the learning task and can be discarded from the instance representation for the case-based learning system. (Cardie, 1993)

- *Information gain for feature weighting.* Use the information gain of each feature as the weight for each feature in the similarity metric. (Daelemans et al., 1999; Cardie & Howe,1997)

- *Value-difference metric.* Local feature weighting method that computes a weight for each feature based on the particular feature value exhibited in the test case and in the training case. Weight is based on feature value distributions across possible class values. (Stanfill & Waltz, 1986)

- *Per-category feature importance.* Weight for each feature-class combination is P (f | c). Training case specific weight vector. (Creecy et al., 1992)

- Filter methods use a separate measure/method for feature selection with an inductive bias that is entirely different from the bias employed in the induction algorithm.

Linguistic Biases and Feature Weighting

- Assign weights based on linguistic or cognitive preferences (Cardie, 1996).
  - recency bias: assign higher weights to features that represent temporally recent information
  - focus of attention bias: assign higher weights to features that correspond to words or constituents in focus, e.g. subject of a sentence
  - restricted memory bias: keep the n features with the highest weights
- Use cross validation to determine which biases apply to the task and which actual weights to assign.
- Hybrid filter-wrapper approach.
  - Wrapper approach for bias selection
  - Selected biases direct feature weighting using filter approach

Example Storage

- Some algorithms only store a subset of the most informative training examples in order to focus the system and make it more efficient. Sometimes called *instance editing*.

- *Edit superfluous regular instances,* e.g. the “exemplar growing” IB2 algorithm (Aha et al., 1991):

  Initialize set of stored examples **to** the empty set.
  For each example **in** the training set **do**
      If the example is correctly classified by picking the nearest neighbor in the current stored examples
        then do not store the example
        else add it to the stored examples.

- In some experimental results this approach works as well if not better than storing all training examples.

- *Edit unproductive exceptions*, e.g. the IB3 algorithm (Aha et al., 1991): Delete all instances that are bad class predictors for their neighborhood in the training set.

- There is evidence that keeping *all* training instances is best in case-based approaches to natural language processing problems (Daelemans et al.,1999).

IBL Conclusions

- IBL methods base decisions on similarity to specific past instances rather than constructing abstractions.
- Instance-based methods abandon the goal of maintaining concept "simplicity."
- Consequently, they trade decreased learning time for increased classification time.
- Important issues are:
  - Defining appropriate distance metrics.
  - Efficient indexing of training cases.
  - Handling irrelevant features.