Memory-Based Learning

Instance-Based Learning

K-Nearest Neighbor

Motivating Example

Inductive Assumption

- Similar inputs map to similar outputs
  - If not true => learning is impossible
  - If true => learning reduces to defining "*similar*"
- Not all similarities created equal
  - predicting a person's height may depend on different attributes than predicting their IQ

1-Nearest Neighbor

$$Dist(c_1, c_2) = \sqrt{\sum_{i=1}^{N} \left(attr_i(c_1) - attr_i(c_2) \right)^2}$$

$$NearestNeighbor = MIN_j(Dist(c_j, c_{test}))$$

$$prediction_{test} = class_j \ (or \ value_j)$$

- works well if no attribute noise, class noise, class overlap
- can learn complex functions (sharp class boundaries)
- as number of training cases grows large, error rate of 1-NN is at most 2 times the Bayes optimal rate (i.e. if you knew the true probability of each class for each test case)

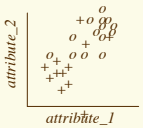
k-Nearest Neighbor

$$Dist(c_1, c_2) = \sqrt{\sum_{i=1}^{N} \left(attr_i(c_1) - attr_i(c_2)\right)^2}$$

$$k - NearestNeighbors = \left\{k - MIN(Dist(c_i, c_{test}))\right\}$$

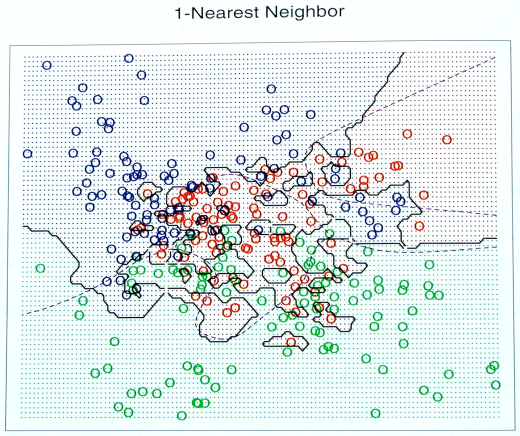
$$prediction_{test} = \frac{1}{k}\sum_{i=1}^{k} class_i \ (or \ \frac{1}{k}\sum_{i=1}^{k} value_i)$$

- Average of k points more reliable when:
  - noise in attributes
  - noise in class labels
  - classes partially overlap

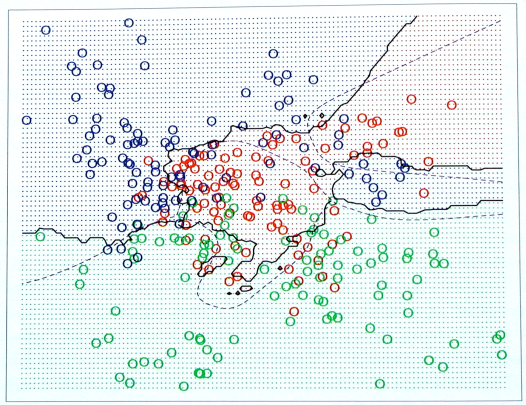


How to choose "k"

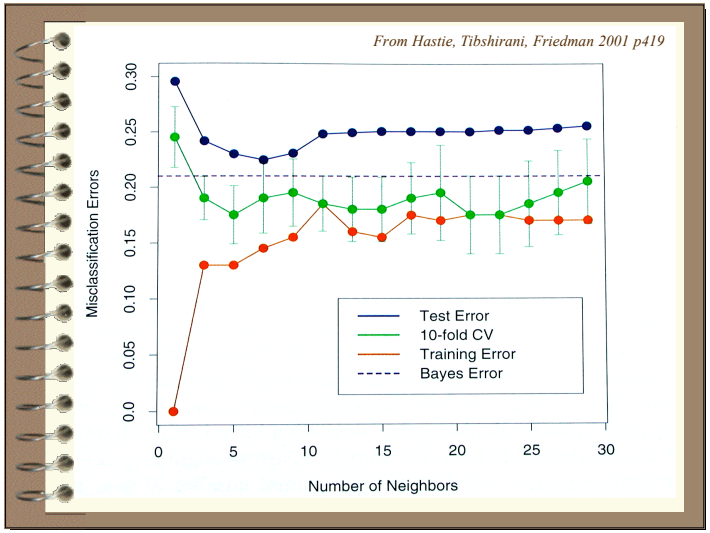
- Large k:
  - less sensitive to noise (particularly class noise)
  - better probability estimates for discrete classes
  - larger training sets allow larger values of k
- Small k:
  - captures fine structure of problem space better
  - may be necessary with small training sets
- Balance must be struck between large and small k
- As training set approaches infinity, and k grows large, kNN becomes Bayes optimal

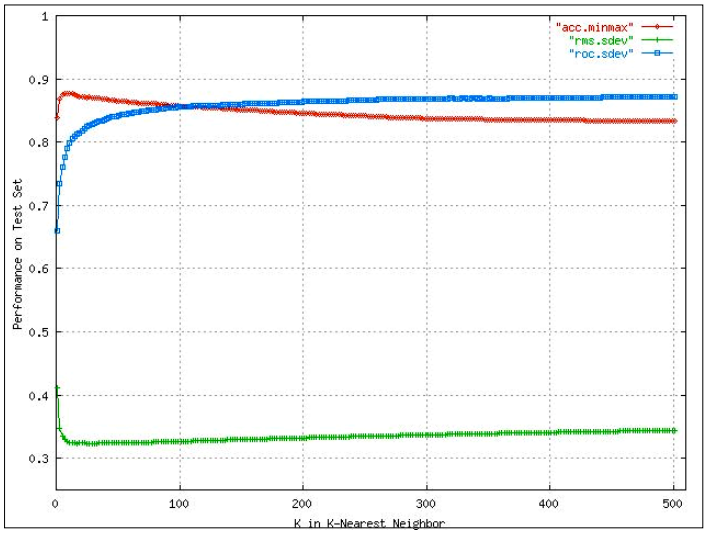


From Hastie, Tibshirani, Friedman 2001 p418



From Hastie, Tibshirani, Friedman 2001 p418





Cross-Validation

- Models usually perform better on training data than on future test cases
- 1-NN is 100% accurate on training data!
- Leave-one-out-cross validation:
  - "remove" each case one-at-a-time
  - use as test case with remaining cases as train set
  - average performance over all test cases
- LOOCV is impractical with most learning methods, but extremely efficient with MBL!

Distance-Weighted kNN

- tradeoff between small and large k can be difficult
  - use large k, but more emphasis on nearer neighbors?

$$prediction_{test} = \frac{\sum_{i=1}^{k} w_i * class_i}{\sum_{i=1}^{k} w_i} \left(or \frac{\sum_{i=1}^{k} w_i * value_i}{\sum_{i=1}^{k} w_i} \right)$$

$$w_k = \frac{1}{Dist(c_k, c_{test})}$$

Locally Weighted Averaging

- Let k = number of training points
- Let weight fall-off rapidly with distance

$$prediction_{test} = \frac{\sum_{i=1}^{k} w_i * class_i}{\sum_{i=1}^{k} w_i} \left(or \frac{\sum_{i=1}^{k} w_i * value_i}{\sum_{i=1}^{k} w_i} \right)$$

$$w_k = \frac{1}{e^{KernelWidth \cdot Dist(c_k, c_{test})}}$$

- KernelWidth controls size of neighborhood that has large effect on value (analogous to k)

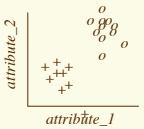
Locally Weighted Regression

- All algs so far are strict averagers: interpolate, but can't extrapolate
- Do weighted regression, centered at test point, weight controlled by distance and KernelWidth
- Local regressor can be linear, quadratic, n-th degree polynomial, neural net, …
- Yields piecewise approximation to surface that typically is more complex than local regressor

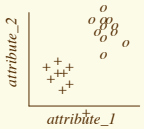
Euclidean Distance

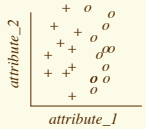
$$D(c1, c2) = \sqrt{\sum_{i=1}^{N} \left(attr_i(c1) - attr_i(c2) \right)^2}$$

- gives all attributes equal weight?
  - only if scale of attributes and differences are similar
  - scale attributes to equal range or equal variance
- assumes spherical classes



Euclidean Distance?





- if classes are not spherical?
- if some attributes are more/less important than other attributes?
- if some attributes have more/less noise in them than other attributes?

Weighted Euclidean Distance

$$D(c1,c2) = \sqrt{\sum_{i=1}^{N} w_i \cdot \left(attr_i(c1) - attr_i(c2)\right)^2}$$

- large weights => attribute is more important
- small weights => attribute is less important
- zero weights => attribute doesn't matter

- Weights allow kNN to be effective with axis-parallel elliptical classes
- Where do weights come from?

Learning Attribute Weights

- Scale attribute ranges or attribute variances to make them uniform (fast and easy)
- Prior knowledge
- Numerical optimization:
  - gradient descent, simplex methods, genetic algorithm
  - criterion is cross-validation performance
- Information Gain or Gain Ratio of single attributes

Information Gain

- Information Gain = reduction in entropy due to splitting on an attribute
- Entropy = expected number of bits needed to encode the class of a randomly drawn + or – example using the optimal info-theory coding

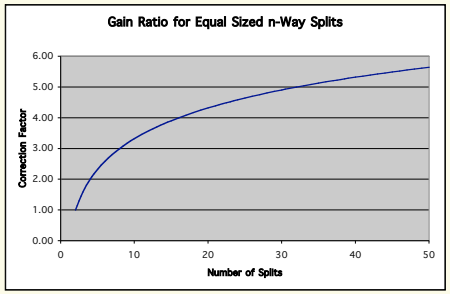
$$Entropy = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Splitting Rules

$$GainRatio(S,A) = \frac{Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)}{\sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}}$$

Gain_Ratio Correction Factor



GainRatio Weighted Euclidean Distance

$$D(c1,c2) = \sqrt{\sum_{i=1}^{N} gain\_ratio_i \cdot \left(attr_i(c1) - attr_i(c2)\right)^2}$$

- weight with gain_ratio after scaling?

Booleans, Nominals, Ordinals, and Reals

- Consider attribute value differences:
  $(attr_i(c1) - attr_i(c2))$
- Reals: easy! full continuum of differences
- Integers: not bad: discrete set of differences
- Ordinals: not bad: discrete set of differences
- Booleans: awkward: hamming distances 0 or 1
- Nominals? not good! recode as Booleans?

Curse of Dimensionality

- as number of dimensions increases, distance between points becomes larger and more uniform
- if number of relevant attributes is fixed, increasing the number of less relevant attributes may swamp distance

$$D(c1,c2) = \sqrt{\sum_{i=1}^{relevant} \left(attr_i(c1) - attr_i(c2)\right)^2 + \sum_{j=1}^{irrelevant} \left(attr_j(c1) - attr_j(c2)\right)^2}$$

- when more irrelevant than relevant dimensions, distance becomes less reliable
- solutions: larger k or KernelWidth, feature selection, feature weights, more complex distance functions

Advantages of Memory-Based Methods

- Lazy learning: don't do any work until you know what you want to predict (and from what variables!)
  - never need to learn a global model
  - many simple local models taken together can represent a more complex global model
  - better focused learning
  - handles missing values, time varying distributions, ...
- Very efficient cross-validation
- Intelligible learning method to many users
- Nearest neighbors support explanation and training
- Can use ***any*** distance metric: string-edit distance, …

Weaknesses of Memory-Based Methods

- Curse of Dimensionality:
  - often works best with 25 or fewer dimensions
- Run-time cost scales with training set size
- Large training sets will not fit in memory
- Many MBL methods are strict averagers
- Sometimes doesn't seem to perform as well as other methods such as neural nets
- Predicted values for regression not continuous

Combine KNN with ANN

- Train neural net on problem
- Use outputs of neural net or hidden unit activations as new feature vectors for each point
- Use KNN on new feature vectors for prediction
- Does feature selection and feature creation
- Sometimes works better than KNN or ANN

Current Research in MBL

- Condensed representations to reduce memory requirements and speed-up neighbor finding to scale to 10^6–10^{12} cases
- Learn better distance metrics
- Feature selection
- Overfitting, VC-dimension, ...
- MBL in higher dimensions
- MBL in non-numeric domains:
  - Case-Based Reasoning
  - Reasoning by Analogy

References

- *Locally Weighted Learning* by Atkeson, Moore, Schaal
- *Tuning Locally Weighted Learning* by Schaal, Atkeson, Moore

Closing Thought

- In many supervised learning problems, all the information you ever have about the problem is in the training set.
- Why do most learning methods discard the training data after doing learning?
- Do neural nets, decision trees, and Bayes nets capture *all* the information in the training set when they are trained?
- In the future, we'll see more methods that combine MBL with these other learning methods.
  - to improve accuracy
  - for better explanation
  - for increased flexibility