k Nearest Neighbors algorithm (kNN)

László Kozma
Lkozma@cis.hut.fi

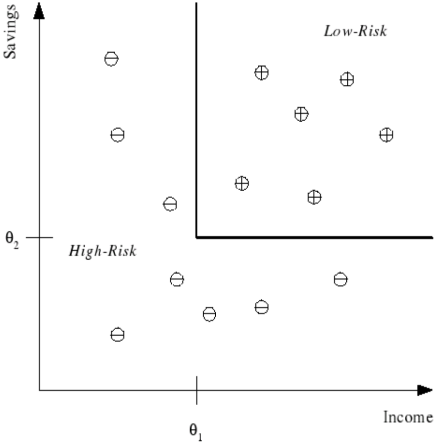
Helsinki University of Technology
T-61.6020 Special Course in Computer and Information Science

20. 2. 2008

Supervised Learning

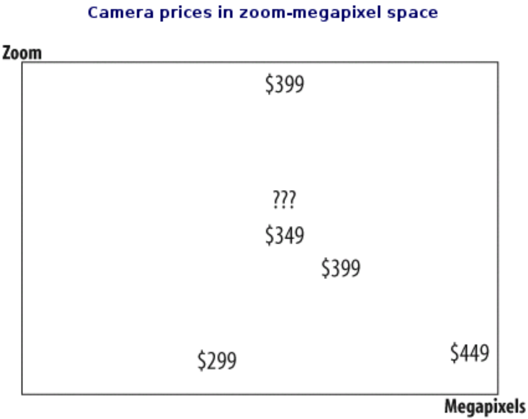
- Data set:
  - Training (labeled) data: $T = \{(x_i, y_i)\}$
  - $x_i \in \mathbb{R}^p$
  - Test (unlabeled) data: $x_0 \in \mathbb{R}^p$
- Tasks:
  - Classification: $y_i \in \{1, \ldots, J\}$
  - Regression: $y_i \in \mathbb{R}$
- Given new x_0 predict y_0
- Methods:
  - Model-based
  - Memory-based

Classification



credit risk assessment (source: Alpaydin ...)

Regression

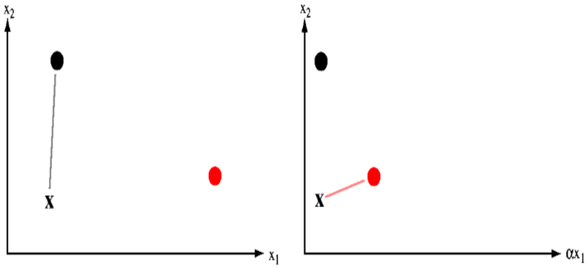


source: O'Reilly ...

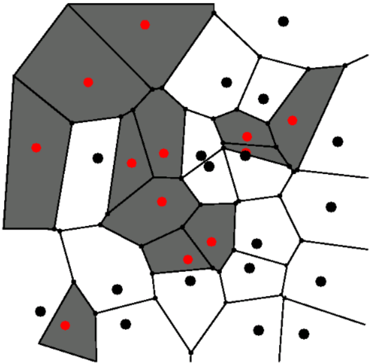
k NN Algorithm

- 1 NN
  - Predict the same value/class as the nearest instance in the training set
- k NN
  - find the k closest training points (small $\|x_i - x_0\|$ according to some metric, for ex. euclidean, manhattan, etc.)
  - predicted class: majority vote
  - predicted value: average weighted by inverse distance

1 NN

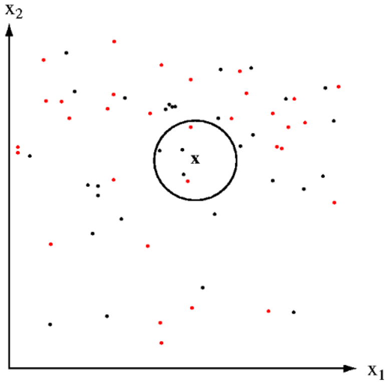


1 NN - Voronoi diagram



source: Duda, Hart ...

k NN - Example



source: Duda, Hart ...

k NN

- Classification
  - use majority voting
- Binary classification
  - k preferably odd to avoid ties
- Regression
  - $y_0 = \sum_{i=1}^{k} w_i y_i$
  - weights:
    - $w_i = \frac{1}{k}$
    - $w_i \sim 1 - \|x_i - x_0\|$
    - $w_i \sim k - rank\|x_i - x_0\|$

k NN Classification

1. Calculate distances of all training vectors to test vector
2. Pick k closest vectors
3. Calculate average/majority

k NN Algorithm

- Memory-based, no explicit training or model, "lazy learning"
- In its basic form one of the most simple machine learning methods
- Gives the maximum likelihood estimation of the class posterior probabilities
- Can be used as a baseline method
- Many extensions

k NN

- + Easy to understand and program
- + Explicit reject option
  - if there is no majority agreement
- + Easy handling of missing values
  - restrict distance calculation to subspace
- + asymptotic misclassification rate (as the number of data points $n \to \infty$) is bounded above by twice the Bayes error rate. (see Duda, Hart...)

k NN

- - affected by local structure
- - sensitive to noise, irrelevant features
- - computationally expensive O(nd)
- - large memory requirements
- - more frequent classes dominate result (if distance not weighed in)
- - curse of dimensionality: high nr. of dimensions and low nr. of training samples:
    - "nearest" neighbor might be very far
    - in high dimensions "nearest" becomes meaningless

Neighborhood size

- Choice of k
  - smaller $k \Rightarrow$ higher variance (less stable)
  - larger $k \Rightarrow$ higher bias (less precise)
  - Proper choice of k dependends on the data:
    - Adaptive methods, heuristics
    - Cross-validation

Distance metric

- Distance used:
  - Euclidean, Manhattan, etc.
- Issue: scaling of different dimensions
- Selecting/scaling features: common problem for all methods
- but affects k NN even more

$\rightarrow$ use mutual information between feature and output

- "Euclidean distance doesn't need any weights for features": just an illusion !!

Extensions

- Reducing computational load:
  - Space partitioning (quad-tree, locality sensitive hashing, etc.)
  - Cluster training data, check input vector only against nearest clusters
  - Editing (remove useless vectors, for example those surrounded by same-class vectors)
  - Partial distance (take distance in less dimensions first)
  - Reduce training set (just sample, or use vector quantization)

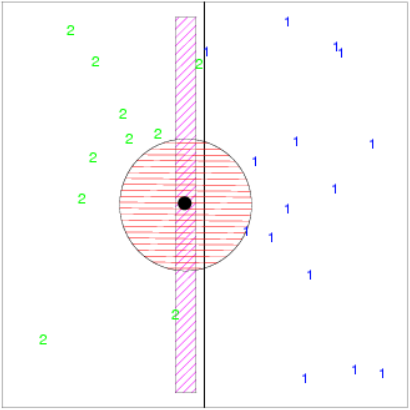
Extensions

- Improving results
  - Preprocessing: smoothing the training data (remove outliers, isolated points)
  - Adapt metric to data

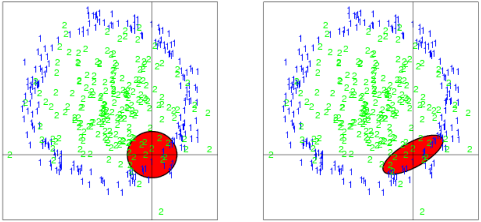
Discriminant Adaptive Nearest Neighbor Classification (DANN)

- k NN is based on the assumption that class probabilities are locally approximately constant
- Not true for most neighborhoods
- Solution: change the metric, so that in the new neighborhoods, class probabilities are "more" constant

DANN - Motivation

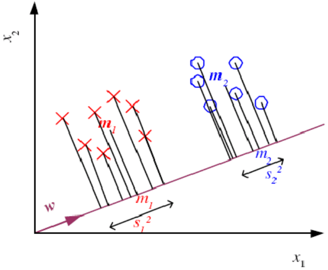


DANN - Example



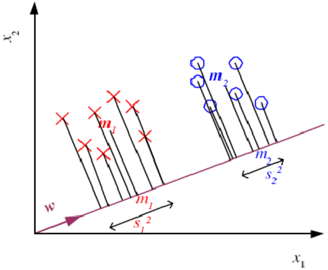
- Idea: DANN creates a neighborhood that is elongated along the "true" decision boundary, flattened orthogonal to it.
- Question: What is the "true" decision boundary?

Linear Discriminant Analysis



- Find w that maximizes $J(w) = \frac{(m_1 - m_2)^2}{{s_1}^2 + {s_2}^2}$ (source: Alpaydin)

Linear Discriminant Analysis



- Solution: $w = (S_1^2 + S_2^2)^{-1}(m_2 - m_1)$
- S_i - class covariance
- Idea: find nearest neighbor using distance between projected points (same as elongating the neighborhood parallel to boundary)
- Squared distance becomes: $D(x, x_0) = (x - x_0)^T w w^T (x - x_0)$

- Squared distance between projections:

$$D(x, x_0) = (x - x_0)^T w w^T (x - x_0) \tag{1}$$

- But we had $w = (S_1^2 + S_2^2)^{-1}(m_2 - m_1)$
- Denote:
  - $W = S_1^2 + S_2^2$ (within-class covariance)
  - $B = (m_2 - m_1)(m_2 - m_1)^T$ (between-class covariance)
- We get $ww^T = W^{-1}BW^{-1}$ (denote by Σ)

DANN

- Squared distance using 'metric' Σ (just a matrix with weights)

$$D(x, x_0) = (x - x_0)^T \Sigma (x - x_0),$$

- if $\Sigma = I \Rightarrow$ Euclidean squared distance
- Reminder: Σ is approximation of local LDA distance

$$\Sigma = W^{-1} B W^{-1} \quad (2)$$

- to avoid neighborhoods infinitely stretching in one direction:

$$\Sigma = W^{-1/2}[W^{-1/2} B W^{-1/2} + \epsilon I] W^{-1/2} \quad (3)$$

DANN

- x_0 - test point
- d_i - distance of x_i from x_0 according to metric Σ

$$d_i = \|\Sigma^{1/2}(x_i - x_0)\| \tag{4}$$

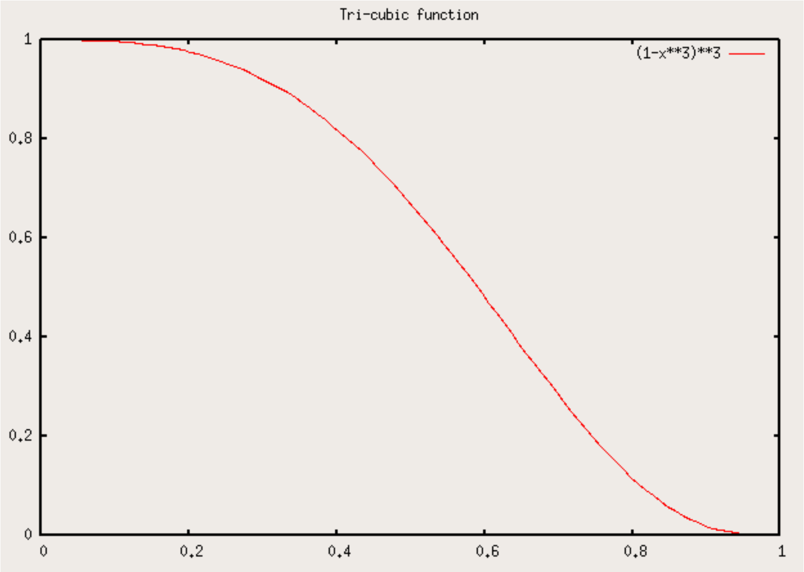
- h - size of the neighborhood

$$h = max_{i \in N_k(x_0)} d_i \tag{5}$$

- assign a weight w_i to each point x_i around x_0 (depending on how far away it is in the neighborhood)
- Use tri-cubic function

$$w_i = (1 - (\frac{d_i}{h})^3)^3 \tag{6}$$

Tri-cubic function



- We now have the weights w_i for each x_i
- The weights depend on the distances (d_i), which depend on the metric (Σ)
- We can calculate B and W, taking the weights into account

$$B = \sum_{j=1}^{J} \alpha_j (\bar{x_j} - \bar{x})(\bar{x_j} - \bar{x})^T \tag{7}$$

$$\alpha_j = \frac{\sum_{y_j = j} w_i}{\sum_{i=1}^{N} w_i} \tag{8}$$

$$W = \sum_{j=1}^{J} \sum_{y_i = j} w_i (x_i - \bar{x_j})(x_i - \bar{x_j})^T / \sum_{i=1}^{N} w_i \tag{9}$$

- $\bar{x}$ - the center of all vectors in the neighborhood
- $\bar{x_j}$ - the center of all vectors belonging to class j

- We started with a metric Σ and a neighborhood around x_0
- Now we have B and W
- But from (3):

$$\Sigma = W^{-1/2}[W^{-1/2}BW^{-1/2} + \epsilon I]W^{-1/2} \tag{10}$$

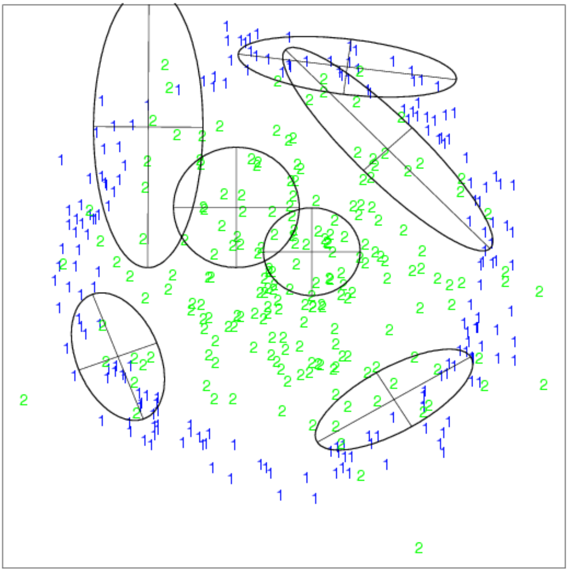
- From Σ we obtain Σ'
- Iterative algorithm can be deviced (see article for proof of convergence and more details)

DANN Algorithm

Predicting y_0 for test vector x_0:

1. Initialize the metric $\Sigma = I$
2. Spread out a nearest neighborhood of K_M points around x_0, using the metric Σ
3. Calculate the weighted 'within-' and 'between-' sum-of-squares matrices W and B using the points in the neighborhood (using class information)
4. Calculate the new metric Σ from (10)
5. Iterate 2,3 and 4 until convergence
6. With the obtained Σ metric perform k NN classification around test point x_0

Result



Choice of parameters

- K_M: number of nearest neighbors for estimating the metric
  - should be reasonably large, especially for high nr. of dimensions
  - $K_M = max(N/5, 50)$
- K: number of nearest neighbors for final k NN rule
  - $K \ll K_M$
  - find using (cross-)validation
  - $K = 5$
- ϵ: 'softening' parameter in the metric
  - fixed value seems OK (see article)
  - $\epsilon > 0$
  - $\epsilon = 1$

Summary

- Nearest Neighbor and k Nearest Neighbor algorithms
  - Baseline methods for classification/regression
  - Have some weak points
  - Several variants exist
- Discriminant Adaptive NN Classification
  - Finds a new metric in a larger neighborhood of the test point
  - Uses class information in a way similar to LDA
  - Uses new metric to perform regular k NN

Sources

1. Hastie, Tibshirani: Discriminant Adaptive Nearest Neighbor Classification (1996)
2. Duda, Hart, Stork: Pattern Classification (Wiley, 2000)
3. Hand, Mannila, Smyth: Principles of Data Mining (MIT Press, 1999)
4. sample images from:
   - Alpaydin: Introduction to Machine Learning (MIT Press, 2004)
   - Segaran: Programming Collective Intelligence (O'Reilly, 2007)
   - D'Silva: DANN presentation, www.lans.ece.utexas.edu/ srean/dann.ppt