Pattern Recognition

Kjell Elenius

Speech, Music and Hearing
KTH

Ch 4. Pattern Recognition 1(3)

- Bayes' Decision Theory
  - Minimum-Error-Rate Decision Rules
  - Discriminant Functions
- How to Construct Classifiers
  - Gaussian Classifiers
  - The Curse of Dimensionality
  - Estimating the Error Rate
  - Comparing Classifiers (McNemar's test)

Pattern Recognition 2 (3)

- Discriminative Training
  - Maximum Mutual Information Estimation
  - Minimum-Error-Rate Estimation
  - Neural networks
- Unsupervised Estimation Methods
  - Vector Quantization
  - The K-Means Algorithm
  - The EM Algorithm
  - Multivariate Gaussian Mixture Density Estimation

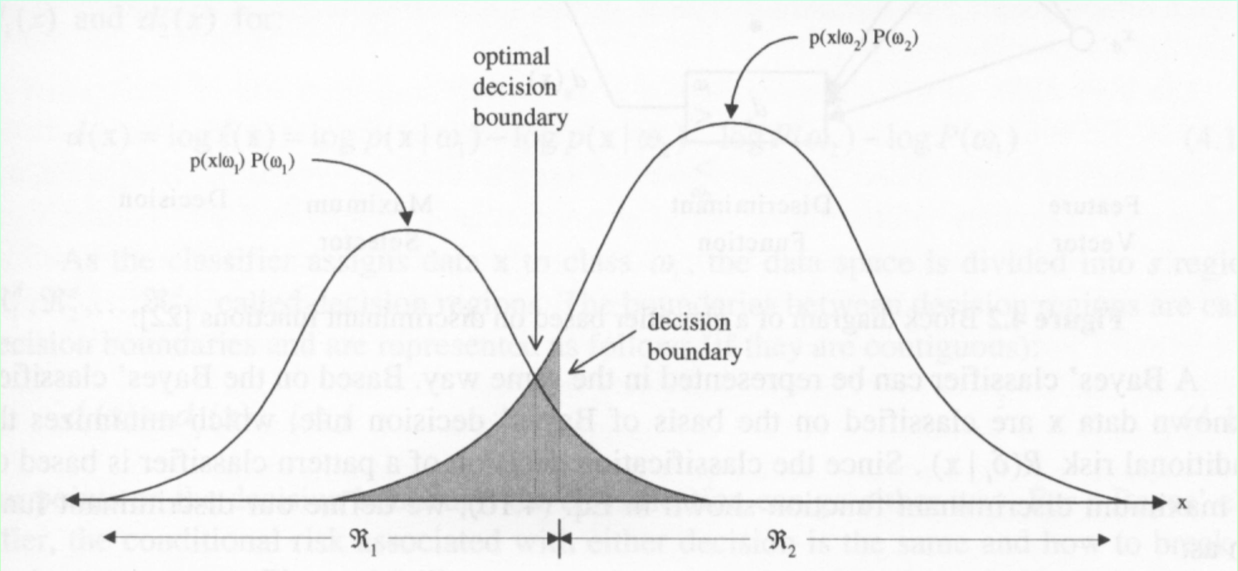
Pattern Recognition 3 (3)

- Classification and Regression Trees (CART)
  - Choice of question set
  - Splitting criteria
  - Growing the tree
  - Missing values and conflict resolution
  - Complex questions
  - The Right-Sized Tree

4.1.1 Minimum-Error-Rate Decision Rules

– Bayes' decision rule
– The decision is based on choosing the candidate that maximizes the posterior probability (results in minimum decision error)

$$k = \underset{i}{\arg\max}\, P(\omega_i \mid x) = \underset{i}{\arg\max}\, p(x \mid \omega_i) P(\omega_i)$$



4.1.2 Discriminant Functions

- The decision problem viewed as classification problem
  - Classify unknown data into one of *s* known categories
  - Using *s* discriminant functions
- Minimum-error-rate classifier:
  - Maximize a posteriori probability: Bayes' decision rule
- For two-class problem:

$$\ell(x) = \frac{p(x \mid \omega_1)}{p(x \mid \omega_2)}$$

$$\ell(x) > T : \omega_1$$

$$\ell(x) < T : \omega_2$$

$$T = \frac{P(\omega_2)}{P(\omega_1)}$$

  - Likelihood ratio:

Discriminant Functions

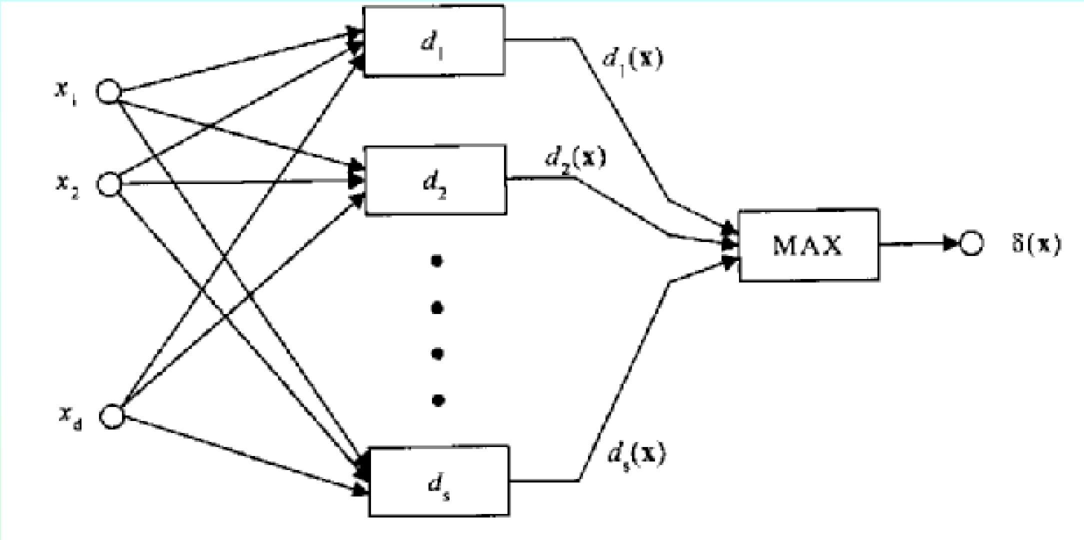
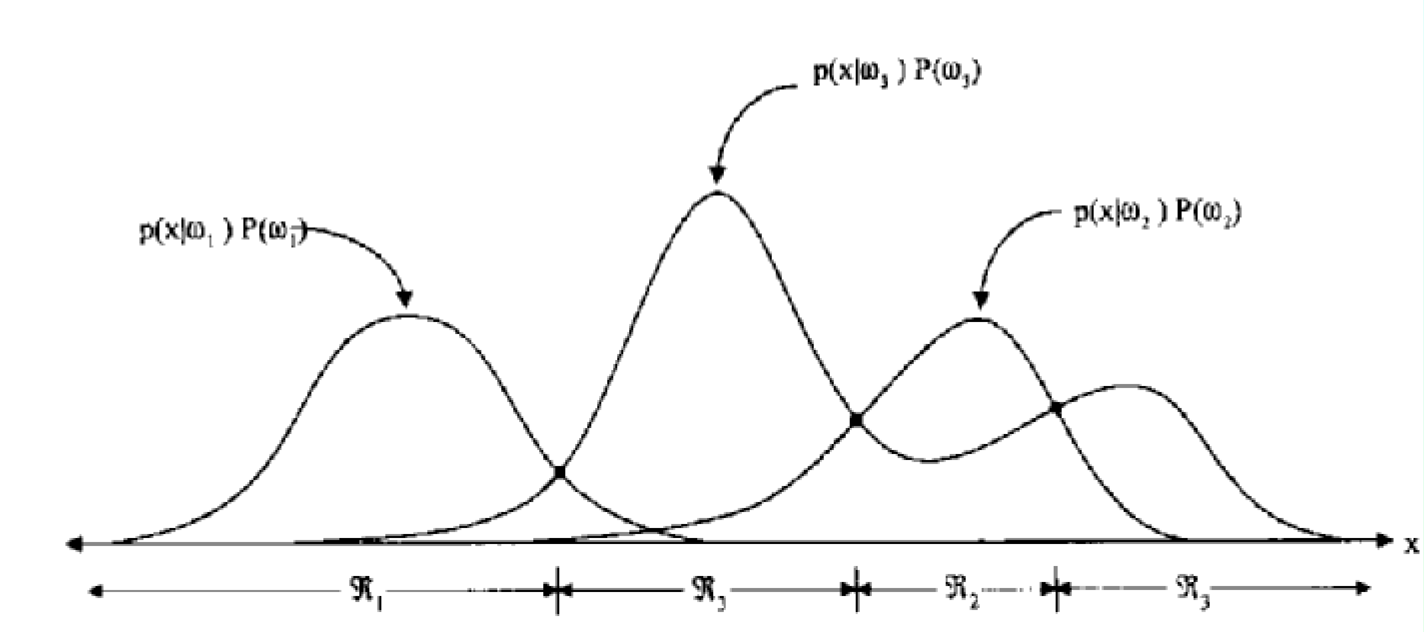


Fig 4.2 A classifier based on discriminant functions

Fig 4.3 Decision boundaries

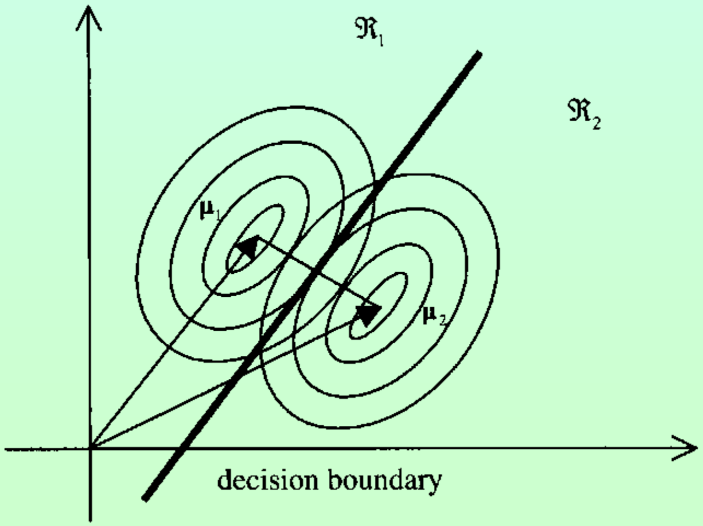


4.2.1 Gaussian classifiers

- The class-conditional probability density is assumed to have a Gaussian distribution

$$p(\mathbf{x}\,|\,\omega_i) = \frac{1}{(2\pi)^{d/2}\,|\,\Sigma_i\,|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right]$$

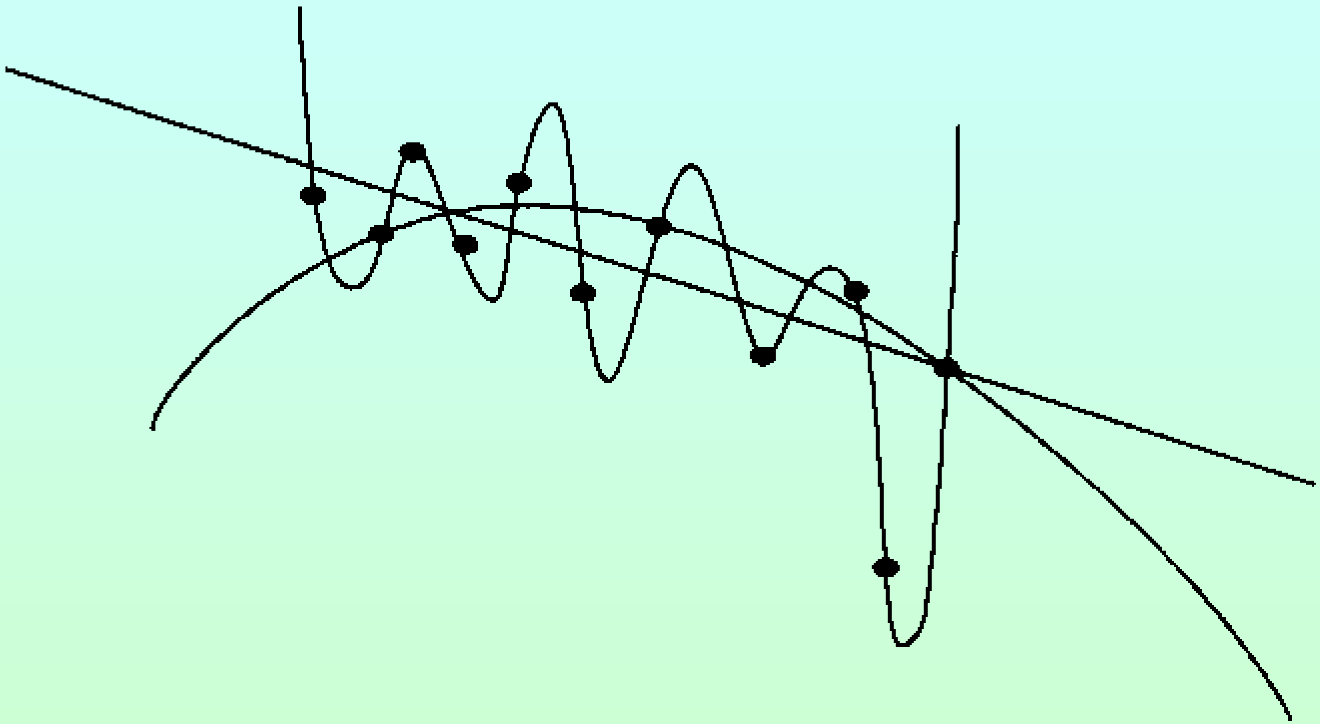
- Decision boundary



4.2.2 The Curse of Dimensionality

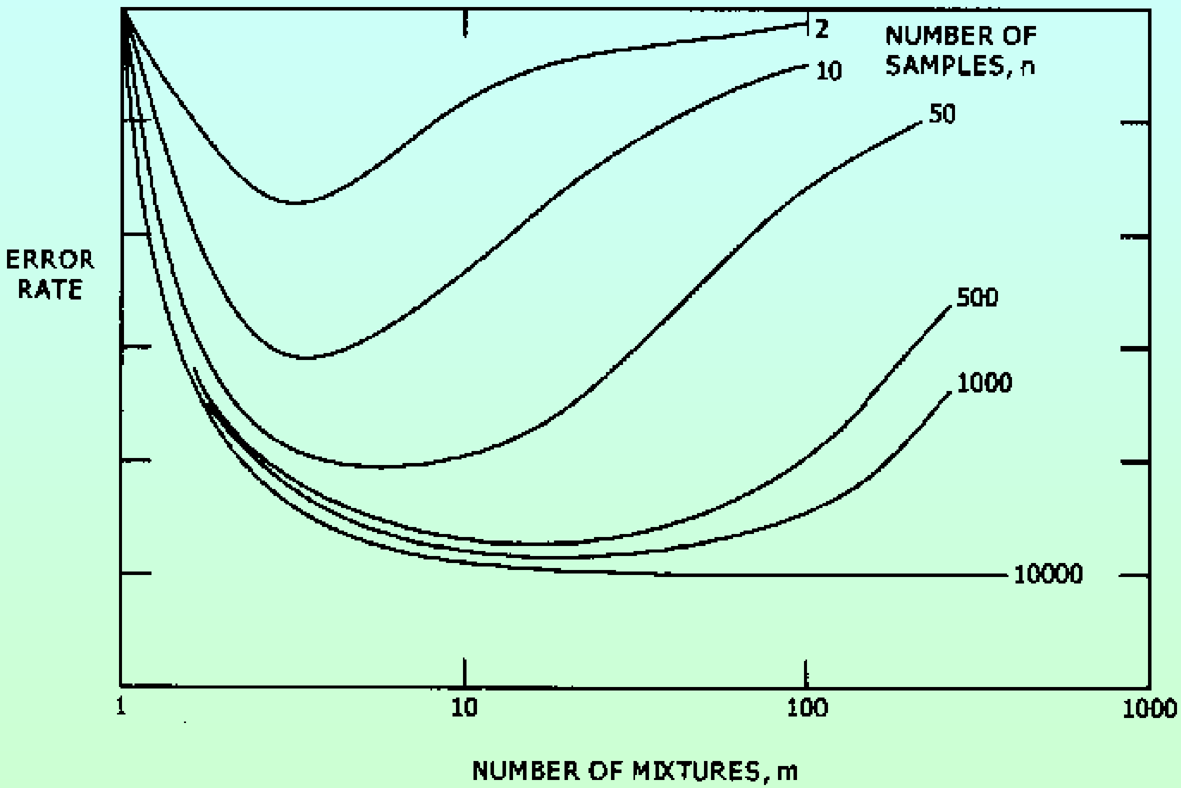
- More features (e.g. higher dimensions or more parameters in density function) lead (in theory) to lower classification error rate
- In practice: may lead to worse results due to too little training data
- Paradox called *The curse of dimensionality*
- Fig 4.6 Curve fitting
- Fig 4.7 Phoneme classification

The Curse of Dimensionality
Curve Fitting



First order, linear, second and
10th order polynomial fitting curves

The Curse of Dimensionality
Two-phoneme classification



Error rate as a function of the number of Gaussian mixture densities and the number of training samples (2 – 10000)

4.2.3 Estimating the Error Rate

- Computation from parametrical model has problems (error under-estimation, bad model assumptions, very difficult)
- Recognition error on training data is a lower bound (Warning!)
- Use independent test data
- How to partition the available speech data
  - Holdout method
  - V-fold cross validation (Leave-one-out method)

Is Algorithm/System A better than B?

- Compare results on the same test data
  - McNemar Test
    - Compares classification results (next slide)
  - Sign Test
    - May be used if the results are considered as Matched Pairs
    - Only thing measured is whether A or B is better
  - Magnitude Difference Test
    - Measures how much better A or B is

4.2.4 Comparing classifiers

- McNemar's test
  - Compares two classifiers by looking at samples where only one made an error

		Q_2 Correct	Q_2 Incorrect
Q_1	Correct	N_{00}	N_{01}
Q_1	Incorrect	N_{10}	N_{11}

$n = N_{01} + N_{10}$ N_{xx} has binomial distribution $B(n, 1/2)$

Test the null hypothesis that the classifiers have the same error rates (z-test)

Confidence

- The true result is within an interval around the measured value with a certain probability
  - Confidence level and interval
- Doddington’s “Rule of 30”
  - *To be 90 percent confident that the true error rate is within +/- **30** percent of the observed error rate, there must be at least **30** errors.*

Confidence Intervals

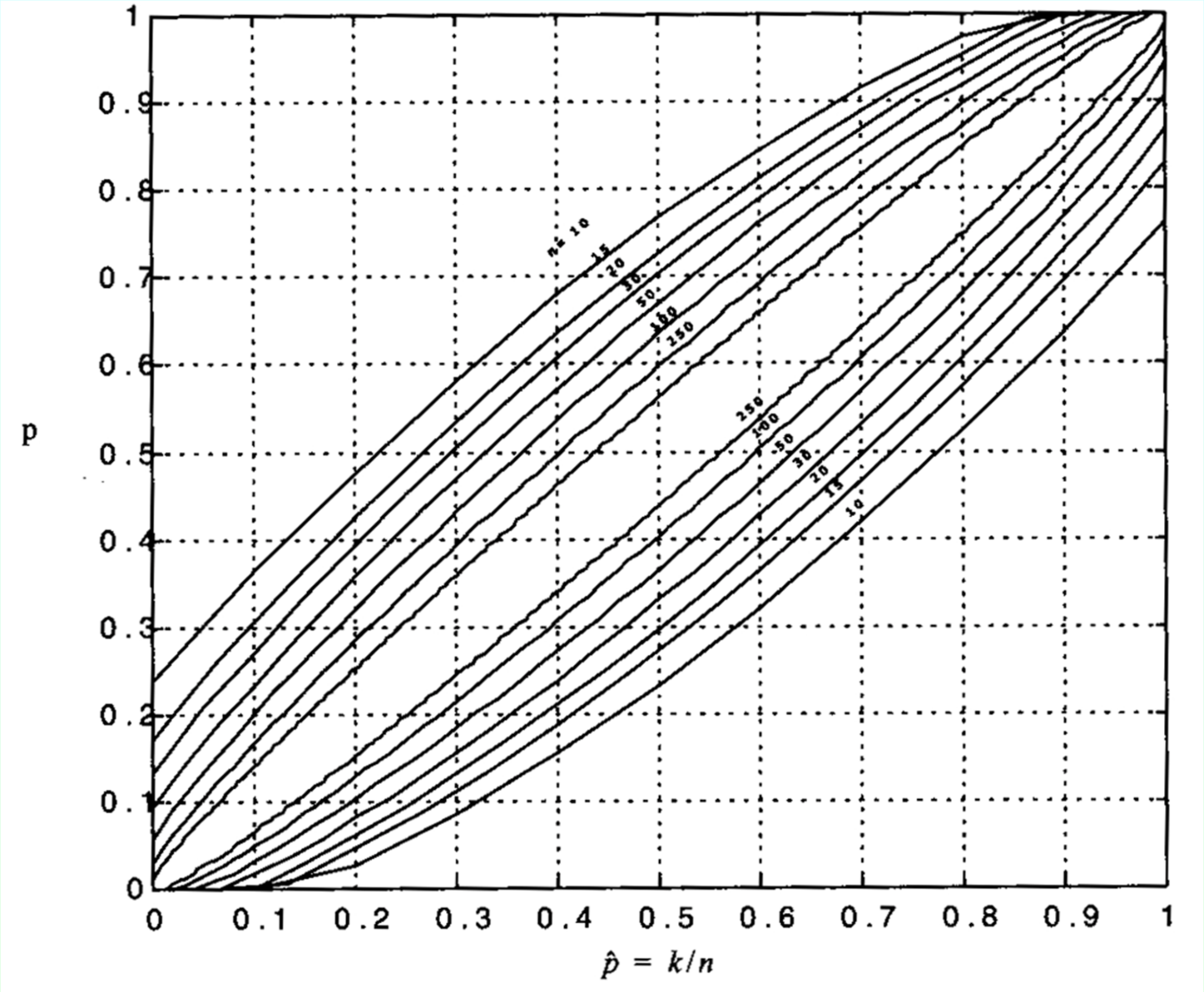


Figure 4.8 95 % confidence intervals for classification error rate estimation with normal test
n = 10, 15, 20, 30, 50, 100, 250

4.3 Discriminative Training

- Maximum Likelihood Estimation models each class separately, independent of other classes

- Discriminative Training aims at models that maximize the discrimination between the classes
  - Maximum Mutual Information Estimation (MMIE)
  - Minimum-Error-Rate Estimation
  - Neural networks

4.3.1 Maximum Mutual Information Estimation (MMIE)

- Discriminative criterion:
  - For each model to estimate, find a setting that maximizes the probability ratio between the model and the sum of all other models. Maximizes the posterior probability.
- Maximize $$\frac{p(\mathbf{x}|\omega_i)\,p(\omega_i)}{\sum_{k\neq i} p(\mathbf{x}|\omega_k)\,p(\omega_k)}$$
- Gives different result compared to MLE.
  MLE maximizes the numerator only
- Theoretically appealing but computationally expensive
  - Every sample used for all classes
  - Use gradient descent algorithm

4.3.2 Minimum-Error-Rate Estimation

- Also called Minimum-Classification-Error (MCE) training, discriminative training,
- Iterative procedure (gradient descent)
  - Re-estimate models, classification, improve correctly recognized models and suppress mis-recognized models
- Computationally intensive, used for few classes
- Corrective training
  - Simple and faster error-correcting procedure
  - Move the parameters of the correct class towards the training data
  - Move the parameters of the near-miss class away from the training data
  - Good results

4.3.3 Neural Networks

- Inspired by nerve cells in biological nervous systems
- Many simple processing elements connected to a complex network.
- Single-Layer Perceptron Fig. 4.10
- Multi-Layer Perceptron (MLP) Fig 4.11
  - Back propagation training

Artificial Neural Network - ANN

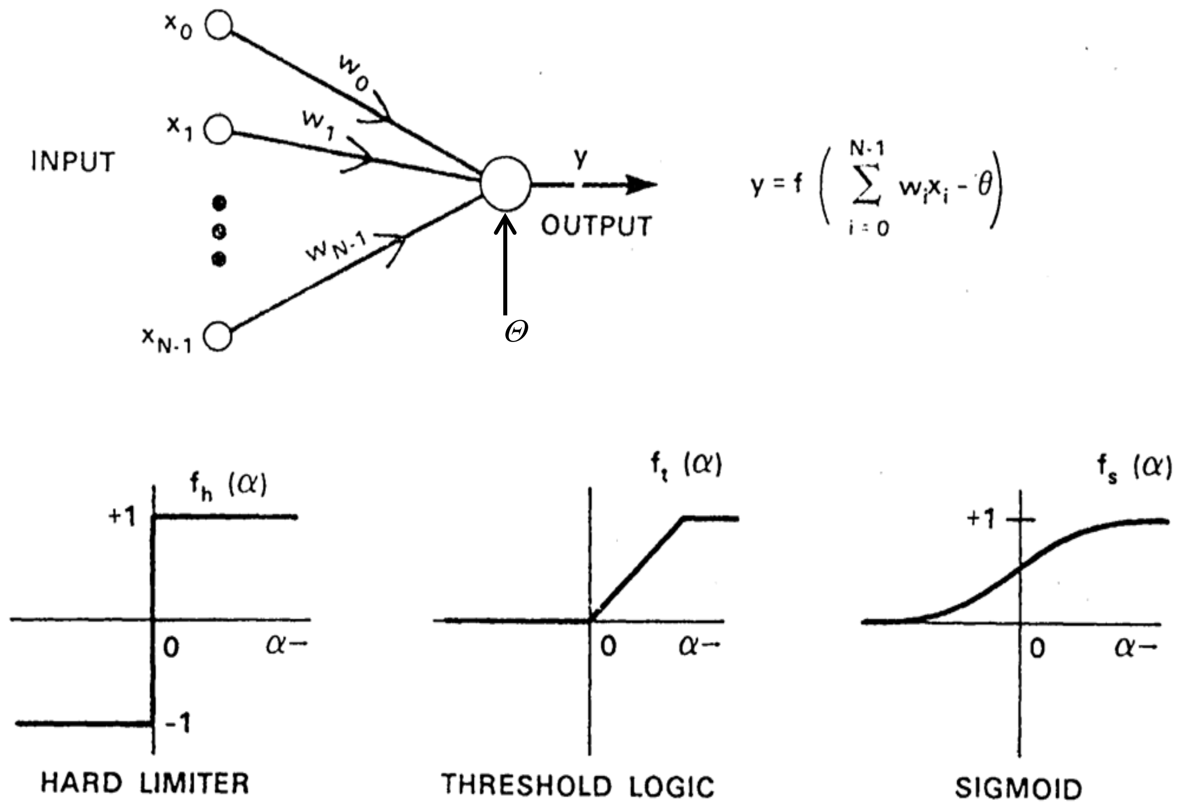
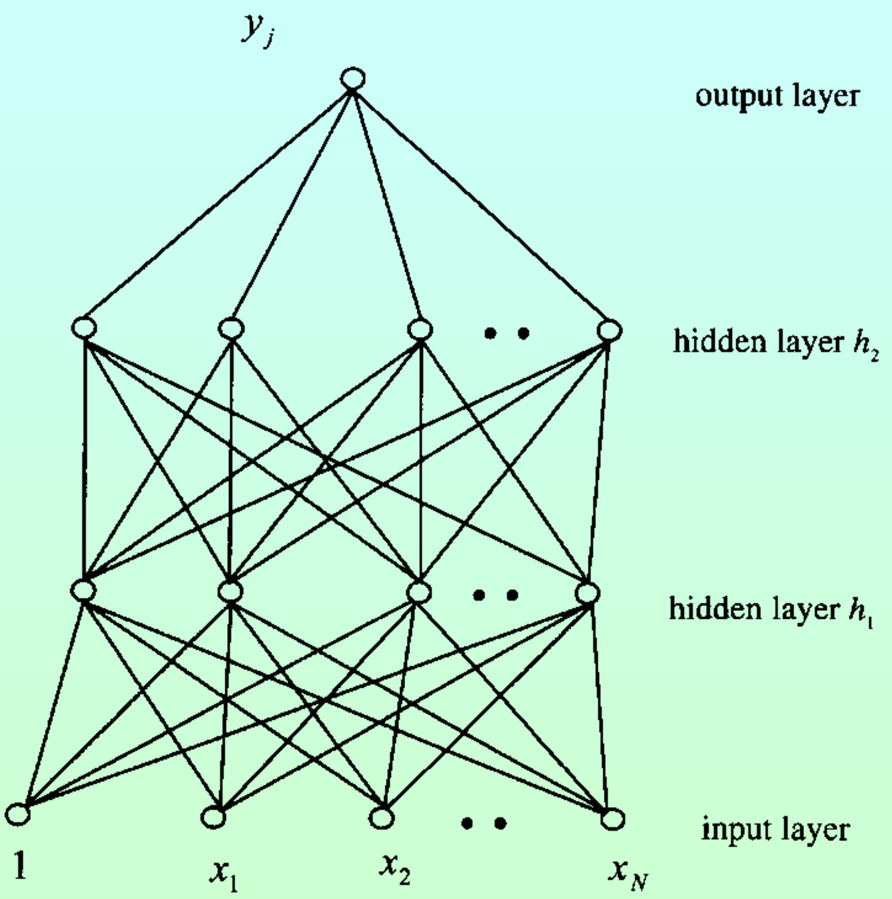


Figure 1: Computation performed in a single node. Three representative nonlinearities are shown.

4.3.3 Multi-Layer Perceptron

$$\mathbf{y} = sigmoid(\mathbf{W'}_y(\mathbf{h}_2))$$

$$y_j = \frac{1}{1 + e^{-(w_{0j} + \sum_i w_{ij} h_{2i})}}$$



The Back Propagation Algorithm

ALGORITHM 4.1: *THE BACK PROPAGATION ALGORITHM*

Step 1: Initialization: Set $t=0$ and choose initial weight matrices $\mathbf{W}$ for each layer. Let's denote $w_{ij}^k(t)$ as the weighting coefficients connecting i^{th} input node in layer $k-1$ and j^{th} output node in layer k at time t.

Step 2: Forward Propagation: Compute the values in each node from input layer to output layer in a propagating fashion, for $k=1$ to K

$$v_j^k = sigmoid(w_{0j}(t) + \sum_{i=1}^{N} w_{ij}^k(t) v_i^{k-1}) \quad \forall j \tag{4.72}$$

where $sigmoid(x) = \dfrac{1}{1+e^{-x}}$ and v_j^k is denoted as the j^{th} node in the k^{th} layer

Step 3: Back Propagation: Update the weights matrix for each layer from output layer to input layer according to:

$$\overline{w}_{ij}^k(t+1) = w_{ij}^k(t) - \alpha \frac{\partial E}{\partial w_{ij}^k(t)} \tag{4.73}$$

where $E = \sum_{i=1}^{s} \| y_i - o_i \|^2$ and $(y_1, y_2, \ldots y_s)$ is the computed output vector in Step 2.

α is referred to as the learning rate and has to be small enough to guarantee convergence. One popular choice is $1/(t+1)$.

Step 4: Iteration: Let $t = t+1$. Repeat Steps 2 and 3 until some convergence condition is met.

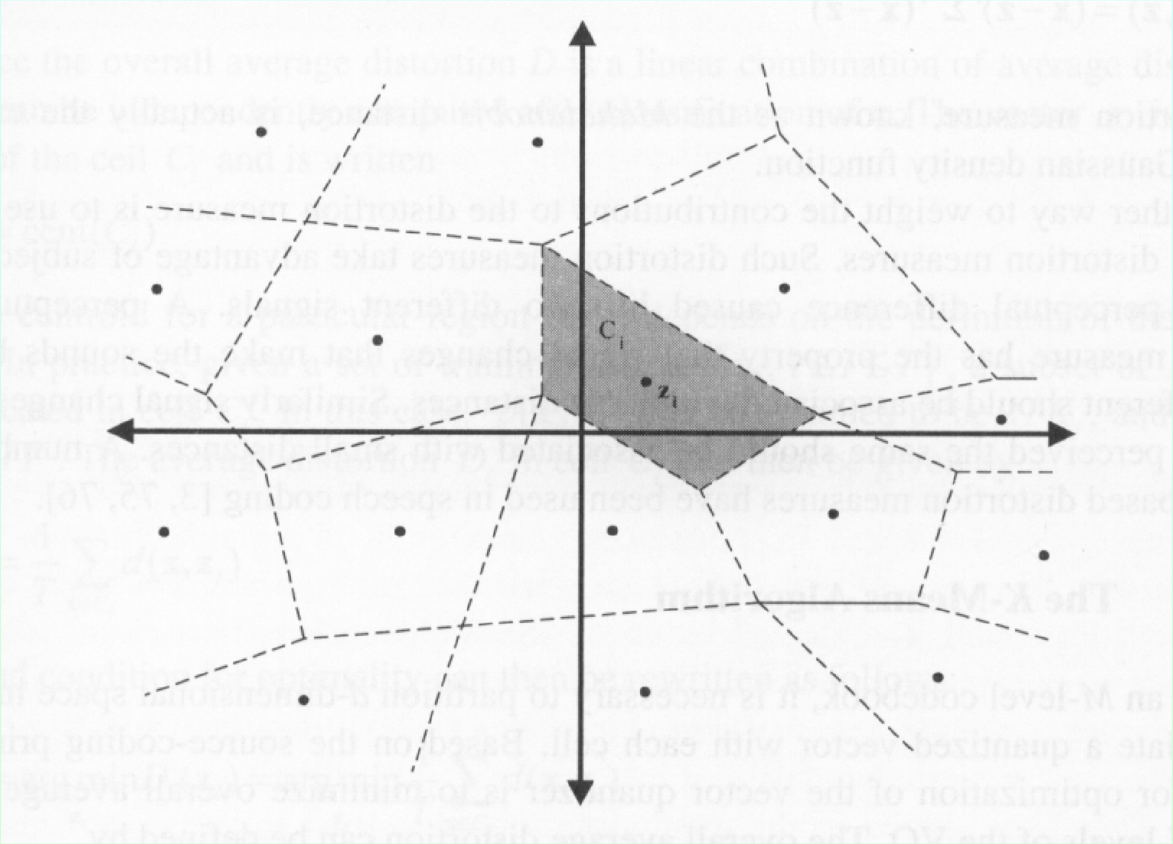
4.4 Unsupervised Estimation Methods

- Vector Quantization
- The EM Algorithm
- Multivariate Gaussian Mixture Density Estimation

4.4.1 Vector Quantization (VQ)

- Described by a codebook, a set of prototype vectors (codewords)
- An input vector is replaced by the index of the codeword with the smallest distortion
- Distortion Measures
  - Euclidean
    - sum of squared error
  - Mahalanobis distance
    - exponential term in Gaussian density function
- Codebook generation algorithms
  - The K-Means Algorithm
  - The LBG Algorithm

Vector Quantization



Partitioning of a two-dimensional space into 16 cells

The K-Means Algorithm

- 1. Choose an initial division between the codewords
- 2. Classify each training vector into one of the cells by choosing the closest codeword
- 3. Update all codewords by computing the centroids of the training vectors
- 4. Repeat steps 2 and 3 until the distortion ratio between current and previous codebook is above a preset threshold

- Comment
  - Converges to *local* optimum
  - Initial choice is critical

The LBG Algorithm

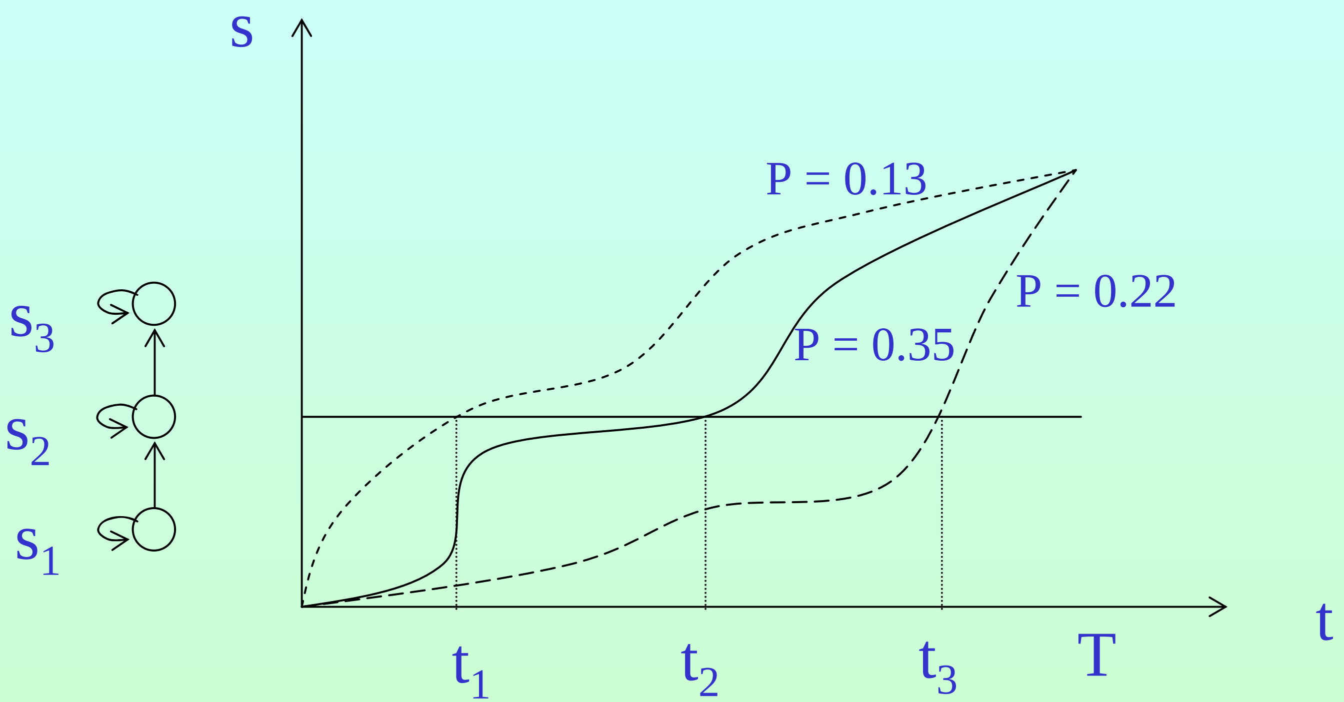
- 1. Initialization.
  - Set number of cells M = 1. Find the centroid of all training data.
- 2. Splitting.
  - Split M into 2M by finding two distant points in each cell. Set these as centroids for 2M cells.
- 3. K-Means Stage.
  - Use K-Means algorithm to modify the centroids for minimum distortion.
- 4. Termination
  - If M equals the required codebook size, STOP. Otherwise go to 2.

4.4.2 The Expectation Maximization (EM) Algorithm

- Used for training of hidden Markov models
- Generalisation of Maximum-Likelihood Estimation
- Problem approached
  - Estimate distributions (ML) of several classes when the training data is not classified (e.g. into states of the models)
  - Is it possible to train the classes anyway? (Yes - *local* maximum)
- Simplified iterative procedure (similar to K-Means procedure for VQ)
  - 1. Initialise class distributions
  - 2. Using current parameters, compute the class probability for each training sample.
  - 3. Each sample updates *each* class distribution by the probability weights
    - Maximum-likelihood estimate of distributions, replace current distr.
  - 4. Repeat 2+3 until convergence (Will converge)

Simplified illustration of EM estimation

Say, three paths have been found in a training utterance.
The probabilities of the state sequences for the initial HMM are 0.13, 0.35 and 0.22.



New $E(s_2) = (0.13\ X(t_1) + 0.35\ X\ (t_2) + 0.22\ X(t_3))\ /\ 0.70$
Not as simple as it may look, though!

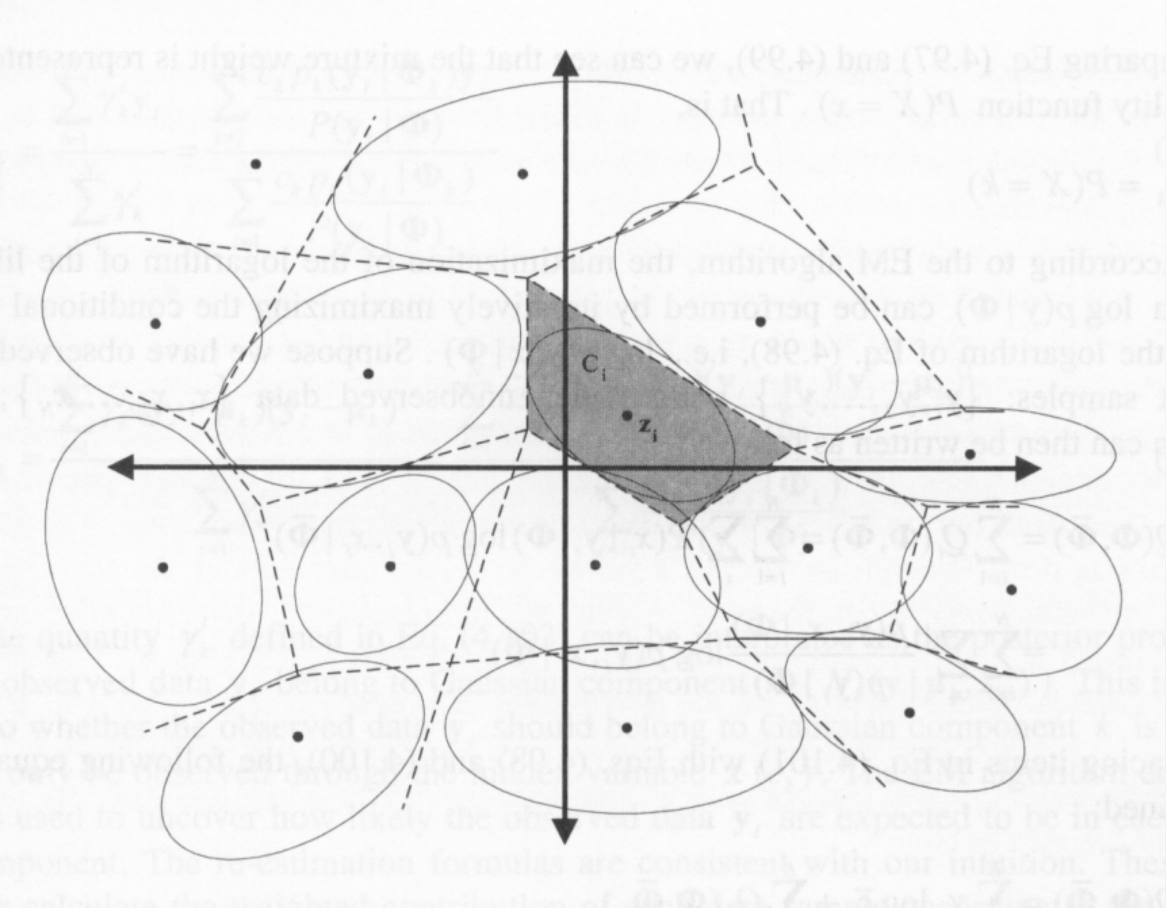
4.4.3 Multivariate Gaussian Mixture Density Estimation

- Probability density is weighted sum of Gaussians:

$$p(\mathbf{y} \mid \mathbf{\Phi}) = \sum_{k=1}^{K} c_k \, p_k(\mathbf{y} \mid \mathbf{\Phi}_k) = \sum_{k=1}^{K} c_k N_k(\mathbf{y} \mid \boldsymbol{\mu}_k, \mathbf{\Sigma}_k)$$

  - c_k is the probability of component k, $c_k = P(X = k)$
- Analogy
  - GM vs VQ,
  - EM algorithm vs K-means algorithm
  - VQ minimizes codebook distortion
    GM maximizes the likelihood of the observed data
  - VQ performs hard assignment
    EM performs soft assignment

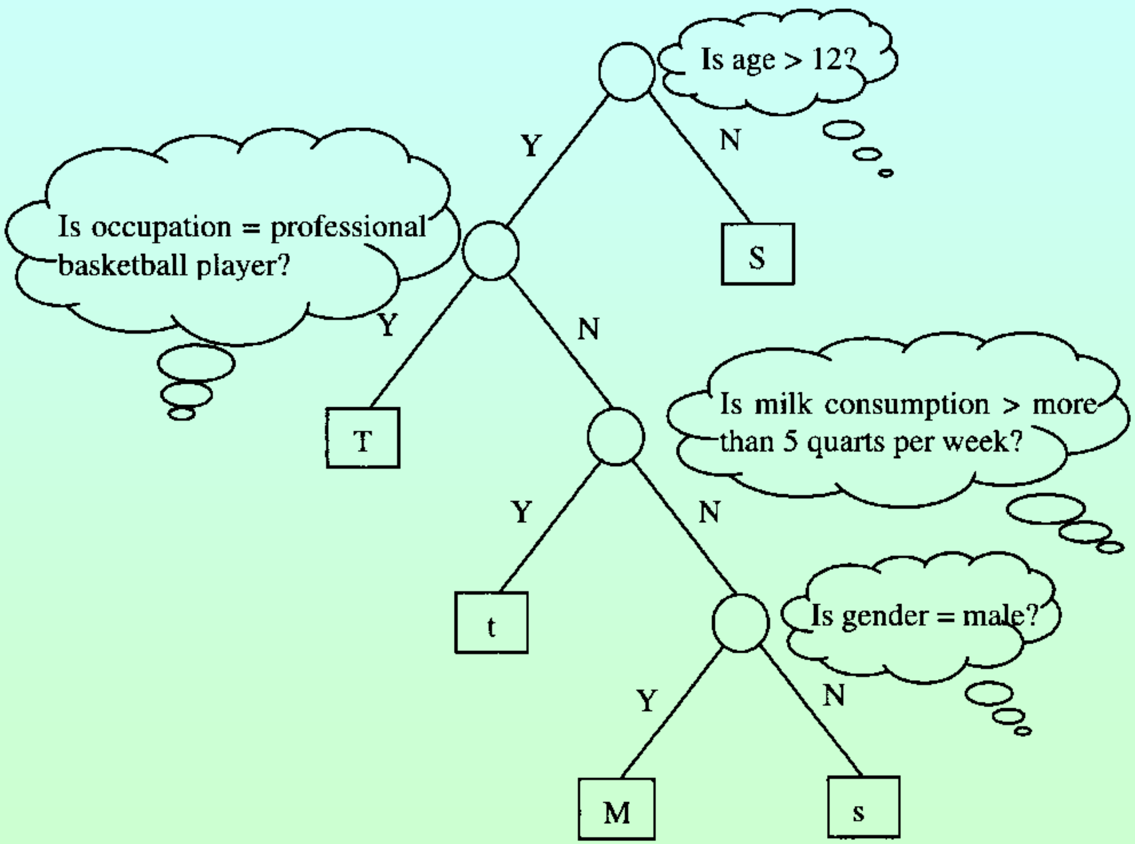
Partitioning Space into Gaussian Density Functions



4.5 Classification And Regression Trees CART

- Binary decision tree
- An automatic and data-driven framework to construct a decision process based on objective criteria
- Handles data samples with mixed types, nonstandard structures
- Handles missing data, robust to outliers and mislabeled data samples
- Used in speech recognition for model tying

Binary Decision Tree for Height Classification



Steps in Constructing a CART

- 1. Find set of questions
- 2. Put all training samples in root
- 3. Recursive algorithm
  - Find the best combination of question and node. Split the node into two new nodes
  - Move the corresponding data into the new nodes
  - Repeat until right-sized tree is obtained
- Greedy algorithm, only locally optimal, splitting without regard to subsequent splits
  - Dynamic programming would help but computationally heavy
  - Works well in practice

4.5.1 Choice of Question Set

- Can be manually selected
- Automatic procedure:
- Simple (singleton) or complex questions
  - Simple questions about a single variable
- Discrete variable questions
  - Does x_i belong to set S?
  - S is any possible subset of the training samples
- Continuous variable questions
  - Is $x_i <= c_n$?
  - c_n is midpoint between two training samples

4.5.2 Splitting Criteria

- Find the pair of node and question for which split gives
  - Discrete variable
    - Maximum reduction in entropy

$$\Delta\overline{H}_t(q) = \overline{H}_t(Y) - (\overline{H}_l(Y) + \overline{H}_r(Y))$$

  - Continuous variables
    - The maximum gain in likelihood

$$\Delta\overline{L}_t(q) = L_1(\mathbf{X}_1 | N) + L_2(\mathbf{X}_2 | N) - L_X(\mathbf{X} | N)$$

  - For regression purposes
    - The largest reduction in squared error from a regression of the data in the node

4.5.3 Growing the Tree

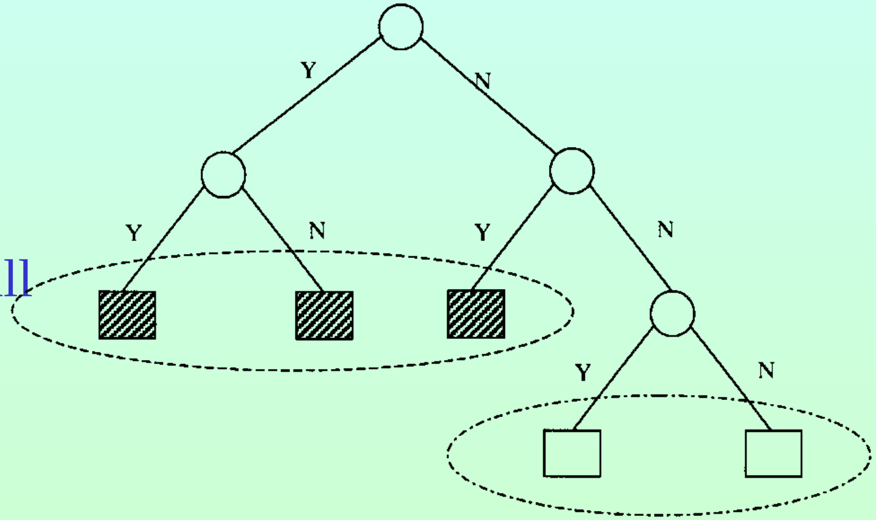
- Stop growing a node when either
  - All samples in the node belong to the same class
  - The greatest entropy reduction falls below threshold
  - The number of data samples in the node is too small

4.5.4 Missing Values and Conflict Resolution

- Missing values in input
  - Handle by surrogate question(s)
- Conflict resolution
  - Two questions may achieve the same entropy reduction and the same partitioning
  - One question may be sub-question to the other
  - Select the sub-question (since more specific)

4.5.5 Complex Questions

- Problem
  - Simple (one-variable) questions may result in similar leaves in different locations
  - Over-fragmenting
- Solution
  - Form composite questions by all possible combinations of the simple-question leaf nodes in the tree using all AND and OR combinations. Select best composite question.



4.5.6 The Right-Sized Tree

- Too many splits improves classification on training data but reduction on test data (Curse of dimensionality)
- Use a pruning strategy to gradually cut back the over-grown tree until the minimum misclassification on the test data is achieved
  - Minimum Cost-Complexity Pruning
    - Produces a sequence of trees with increased pruning
  - Select the best tree using either of
    - Independent Test Sample Estimation (fixed test data)
    - V-fold Cross Validation (train on (v-1) parts, test on 1, circulate)