CS 570 Data Mining

Classification and Prediction 3

Cengiz Gunay

Partial slide credits: Li Xiong, Han, Kamber, and Pan, Tan,Steinbach, Kumar

Collaborative Filtering Examples

- Movielens: movies
- Moviecritic: movies again
- My launch: music
- Gustos starrater: web pages
- Jester: Jokes
- TV Recommender: TV shows
- Suggest 1.0 : different products

Chapter 6. Classification and Prediction

- Overview
- Classification algorithms and methods
  - Decision tree induction
  - Bayesian classification
  - Lazy learning and kNN classification
  - Support Vector Machines (SVM)
  - Others
- Prediction methods
- Evaluation metrics and methods
- Ensemble methods

Prediction

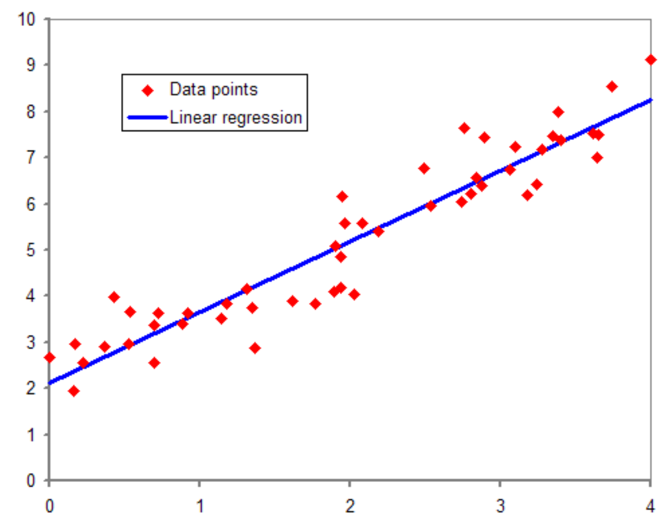
- Prediction vs. classification
  - Classification predicts categorical class label
  - Prediction predicts continuous-valued attributes
- Major method for prediction: regression
  - model the relationship between one or more *independent* or **predictor** variables and a *dependent* or **response** variable
- Regression analysis
  - Linear regression
  - Other regression methods: generalized linear model, logistic regression, Poisson regression, regression trees

Linear Regression

- Linear regression: $Y = b_0 + b_1X_1 + b_2X_2 + \ldots + b_PX_P$
  - Line fitting: $y = w_0 + w_1 x$
  - Polynomial fitting: $Y = b_2x^2 + b_1x + b_0$
  - Many nonlinear functions can be transformed
- Method of least squares: estimates the best-fitting straight line

$$w_1 = \frac{\sum_{i=1}^{|D|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|}(x_i - \bar{x})^2}$$

$$w_0 = \bar{y} - w_1 \bar{x}$$



Linear Regression- Loss Function

$$y = f(x) = w_1 x + w_0$$

$$\mathrm{Loss}(f) = \sum_j (y_j - f(x_j))^2 = \sum_j (y_j - (w_1 x_j + w_0))^2$$

Minimum is where the derivative is zero:

$$\frac{\partial}{\partial w_0} \sum_j (y_j - (w_1 x_j + w_0))^2 = 0, \quad \frac{\partial}{\partial w_1} \sum_j (y_j - (w_1 x_j + w_0))^2 = 0$$

Solution is:

$$w_1 = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum y_j)^2}, \quad w_0 = \left(\sum y_j - w_1(\sum x_j)\right) / N$$

Other Regression-Based Models

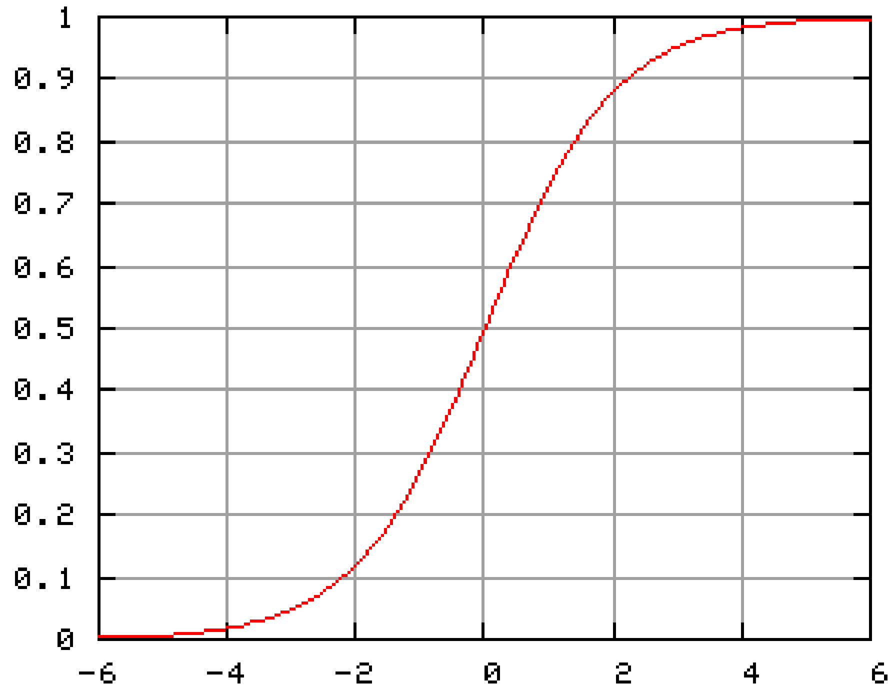
- General linear model
  - Logistic regression: models the probability of some event occurring as a linear function of a set of predictor variables
    - vs. Bayesian classifier
    - Assumes logistic model
  - Poisson regression (log-linear model): models the data that exhibit a Poisson distribution
    - Assumes Poisson distribution for response variable
- Maximum likelyhood method

Logistic Regression

- Logistic regression: models the probability of some event occurring as a linear function of a set of predictor variables
- Logistic function

$$f(z) = \frac{1}{1+e^{-z}}$$

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_k x_k,$$



Poisson Regression

- Poisson regression (log-linear model): models the data that exhibit a Poisson distribution
  - Assumes Poisson distribution for response variable
  - Assumes logarithm of its expected value follows a linear model
  - Simplest case: $\log(\mathrm{E}(Y)) = a + bx.$

Lasso

- Subset selection
- Lasso is defined

$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \Big(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \Big)^2$$

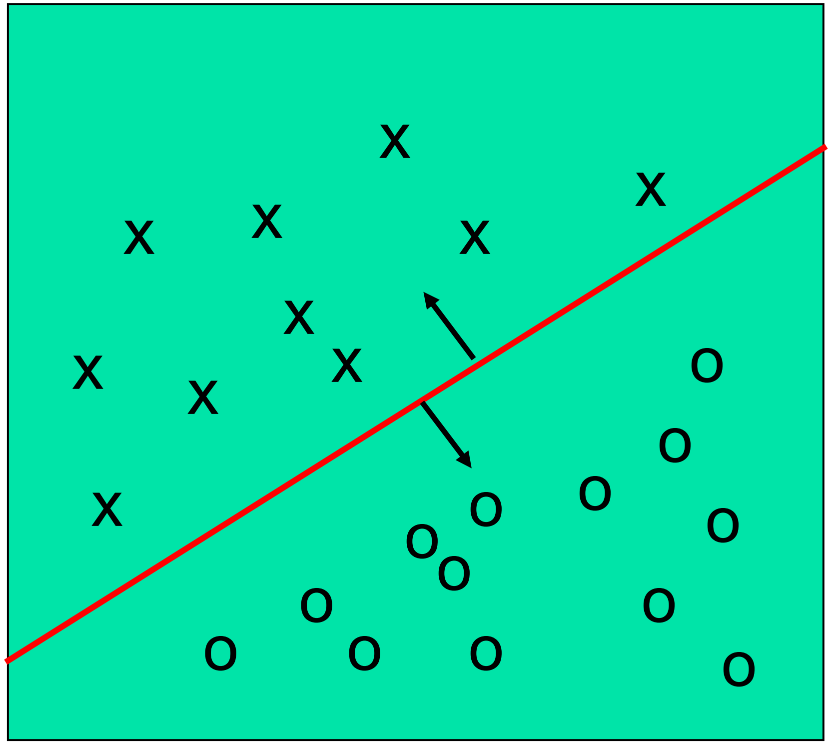
$$\text{subject to } \sum_{j=1}^{p} |\beta_j| \leq t.$$

- Using a small *t* forces some coefficients to 0
- Explains the model with fewer variables
- Ref: Hastie, Tibshirani, Friedman. The Elements of Statistical Learning

Other Classification Methods

- Rule based classification
- Neural networks
- Genetic algorithms
- Rough set approaches
- Fuzzy set approaches

Linear Classification



- Binary Classification problem
- The data above the red line belongs to class 'x'
- The data below red line belongs to class 'o'
- Examples: SVM, Perceptron, Probabilistic Classifiers

Classification: A Mathematical Mapping

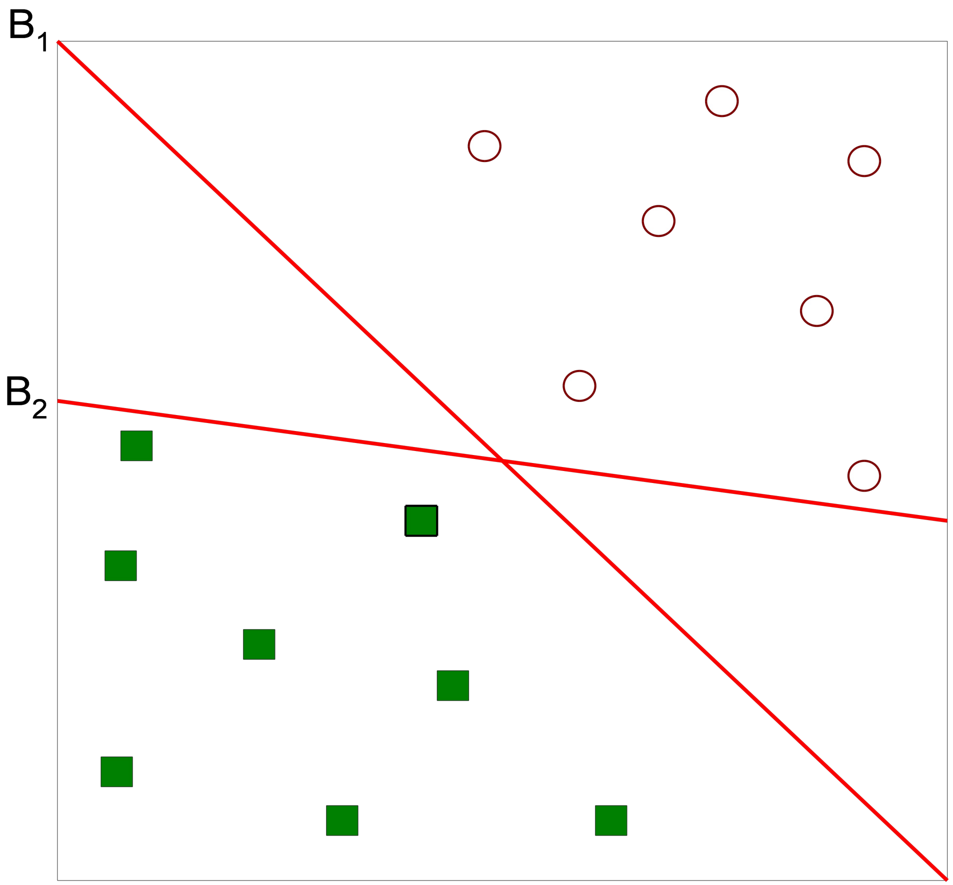
- Mathematically
  - $x \in X = \Re^n$, $y \in Y = \{+1, -1\}$
  - We want a function $f: X \rightarrow Y$
- Linear classifiers
  - Probabilistic Classifiers (Naive Bayesian)
  - SVM
  - Perceptron

Discriminative Classifiers

- Advantages
  - prediction accuracy is generally high
    - As compared to Bayesian methods – in general
  - robust, works when training examples contain errors
  - fast evaluation of the learned target function
    - Bayesian networks are normally slow
- Criticism
  - long training time
  - difficult to understand the learned function (weights)
    - Bayesian networks can be used easily for pattern discovery
  - not easy to incorporate domain knowledge
    - Easy in the form of priors on the data or distributions

Support Vector Machines (SVM)

- Find linear separation in input space

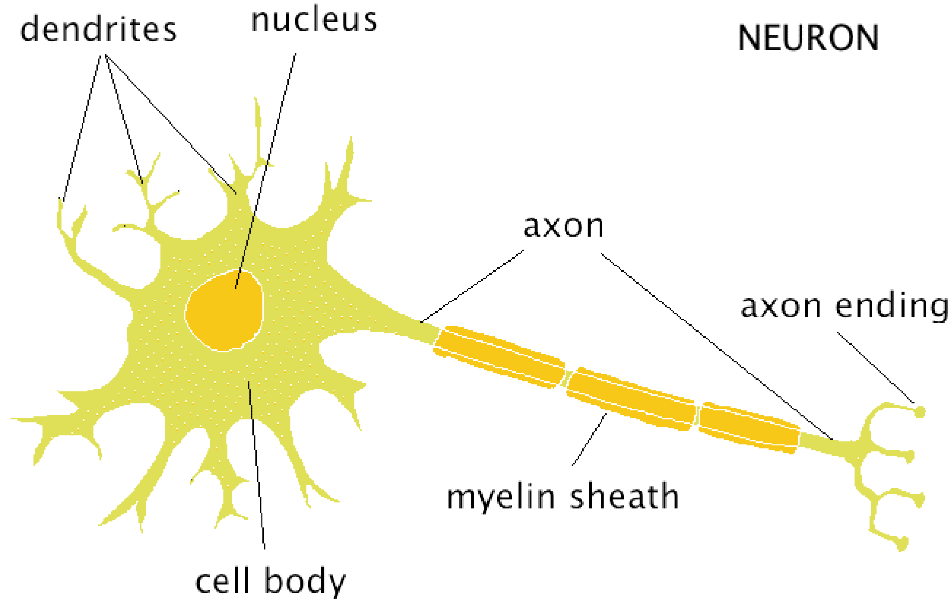


SVM vs. Neural Network

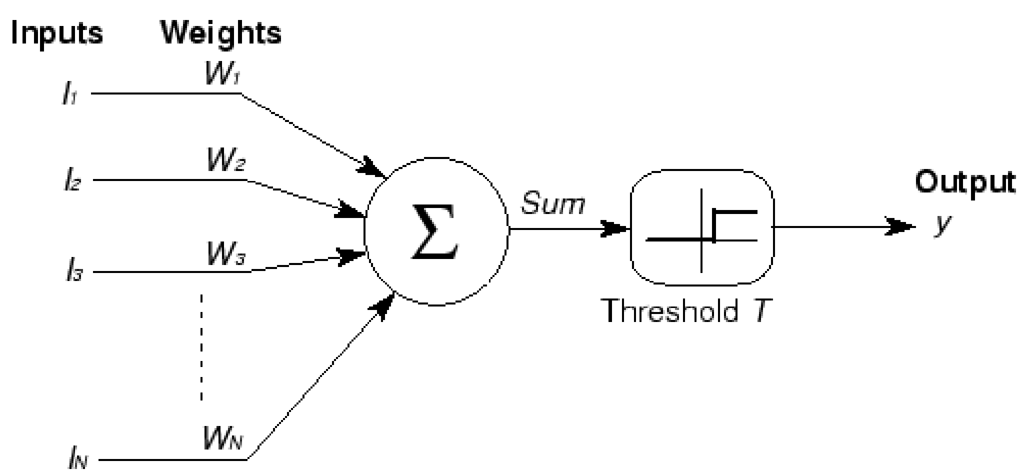
- SVM
  - Relatively new concept
  - Deterministic algorithm
  - Nice Generalization properties
  - Hard to learn – learned in batch mode using quadratic programming techniques
  - Using kernels can learn very complex functions
- Neural Network
  - Relatively old
  - Nondeterministic algorithm
  - Generalizes well but doesn't have strong mathematical foundation
  - Can easily be learned in incremental fashion
  - To learn complex functions—use multilayer perceptron (not that trivial)

Why Neural Networks?

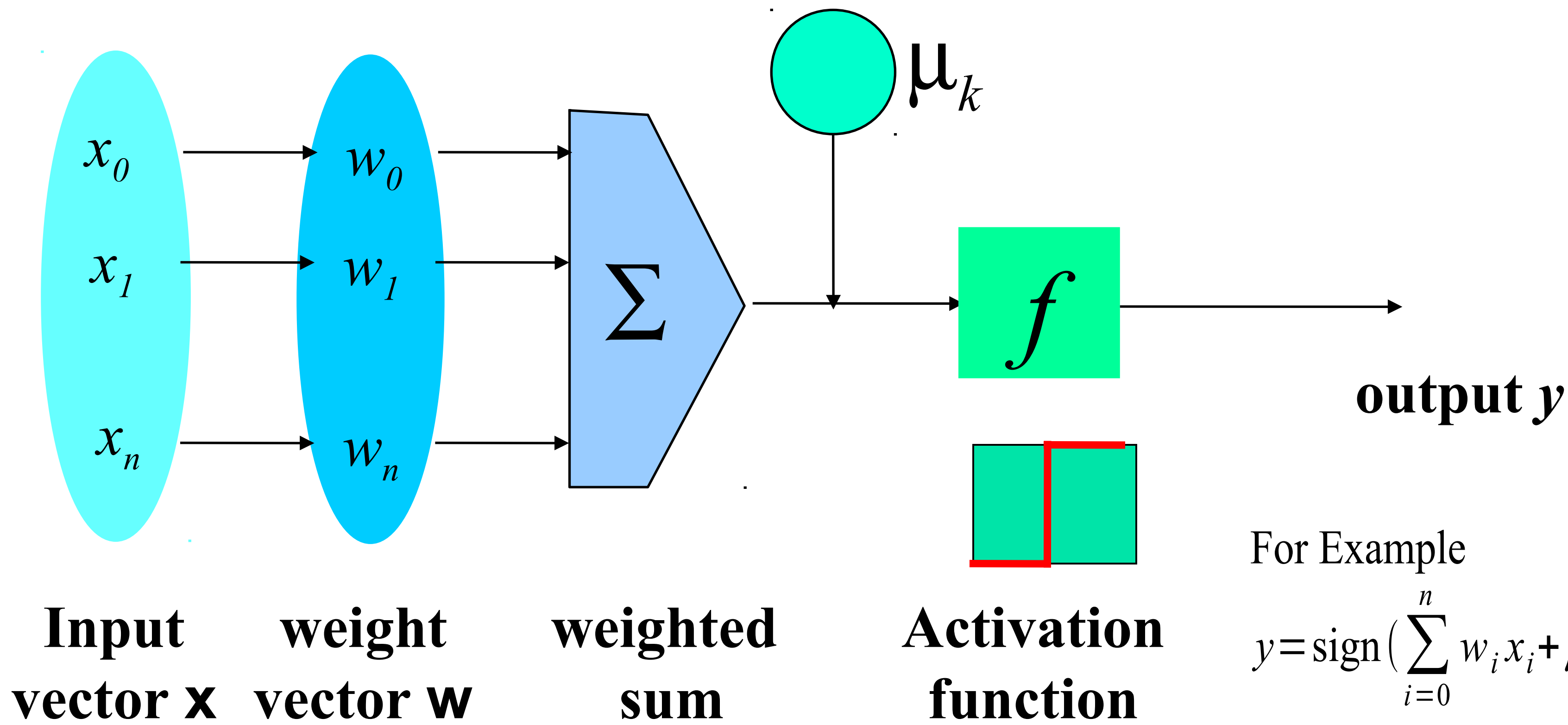
- Inspired by the nervous system:



- Formalized by McCullough & Pitts (1943) as **perceptron**



A Neuron (= a perceptron)

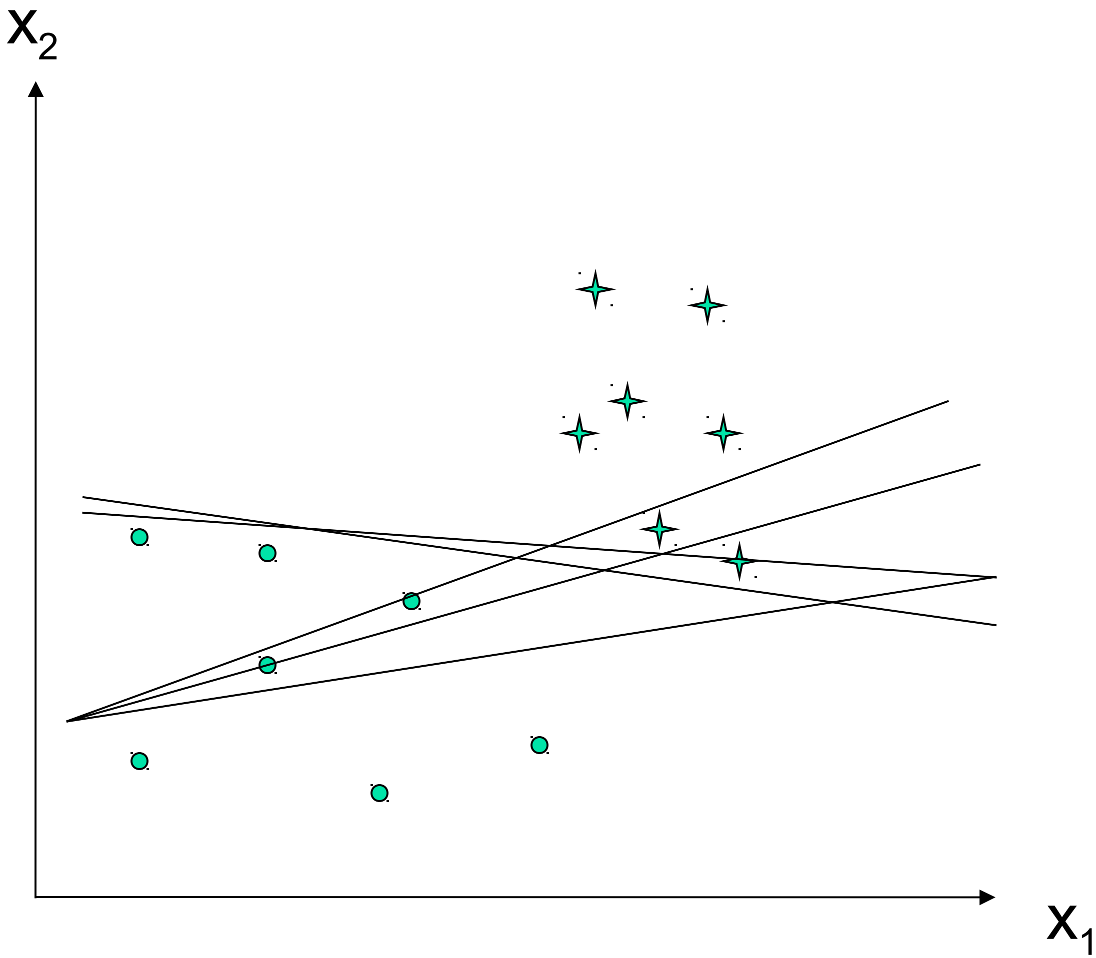


For Example

$$y = \text{sign}\left(\sum_{i=0}^{n} w_i x_i + \mu_k\right)$$

- The *n*-dimensional input vector **x** is mapped into variable y by means of the scalar product and a nonlinear function mapping

Perceptron & Winnow Algorithms



- Vector: **x**; scalar: x

Input: $\{(\mathbf{x}^{(1)}, y^{(1)}), \ldots\}$

Output: classification function $f(\mathbf{x})$

$f(\mathbf{x}^{(i)}) > 0$ for $y^{(i)} = +1$

$f(\mathbf{x}^{(i)}) < 0$ for $y^{(i)} = -1$

f(x) => uses inner product

$\mathbf{w}\ \mathbf{x} + \mathbf{b} = 0$

or $w_1x_1+w_2x_2+b = 0$

Learning updates **w** :

- Perceptron: additively
- Winnow: multiplicatively

Use the Loss Function, Perceptron

Perceptron:

$$y = f_{\mathbf{w}}(\mathbf{x})$$

Over all samples:

$$\mathrm{Loss}(\mathbf{w}) = \sum_i (y_i - f_{\mathbf{w}}(\mathbf{x}_i))^2$$

Trying to find

$$\arg\min_{\mathbf{w}} \mathrm{Loss}(\mathbf{w})$$

An incremental rule:

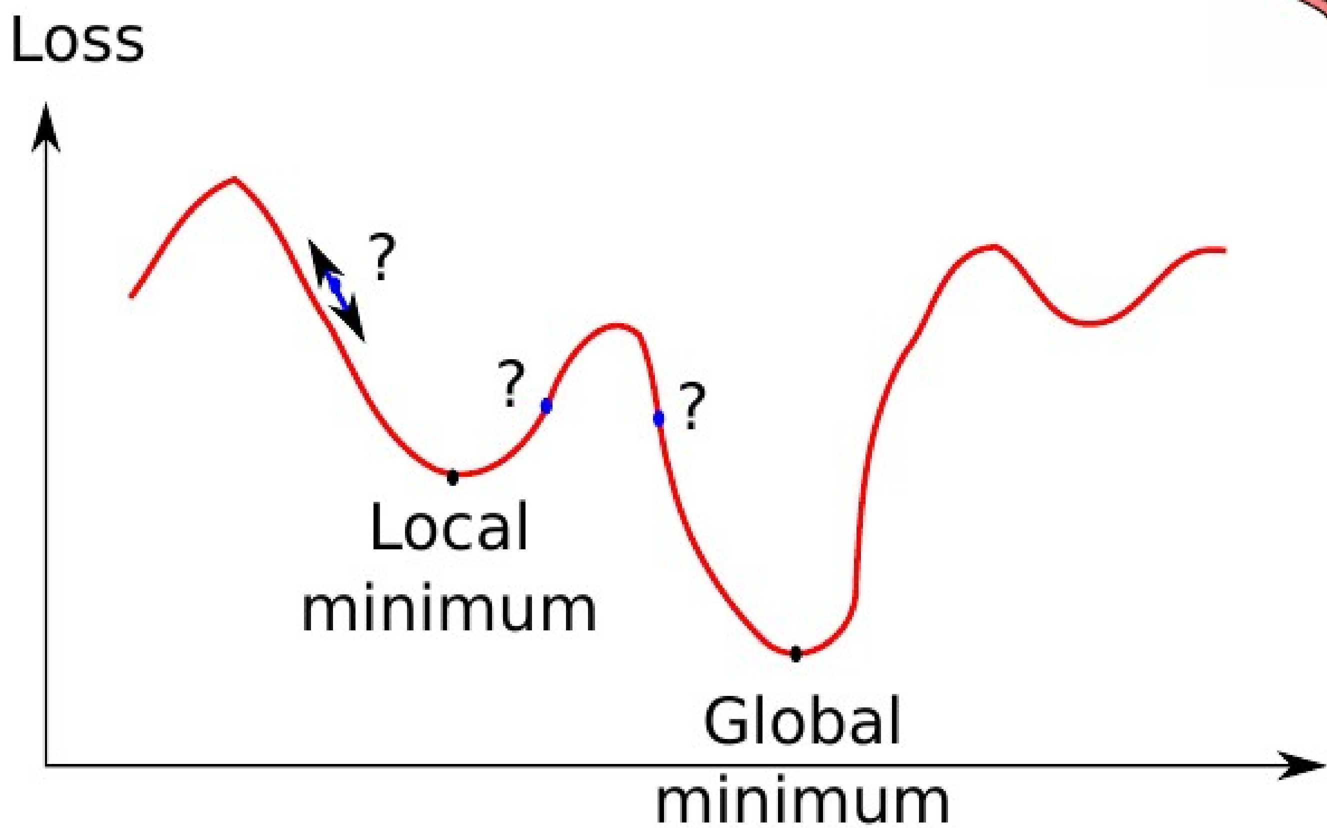
$$w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} \mathrm{Loss}(\mathbf{w})$$

$$w_j \leftarrow w_j + \alpha(y - f_{\mathbf{w}}(\mathbf{x})) \times x_j$$

Gradient Descent on the Loss Function

In general,

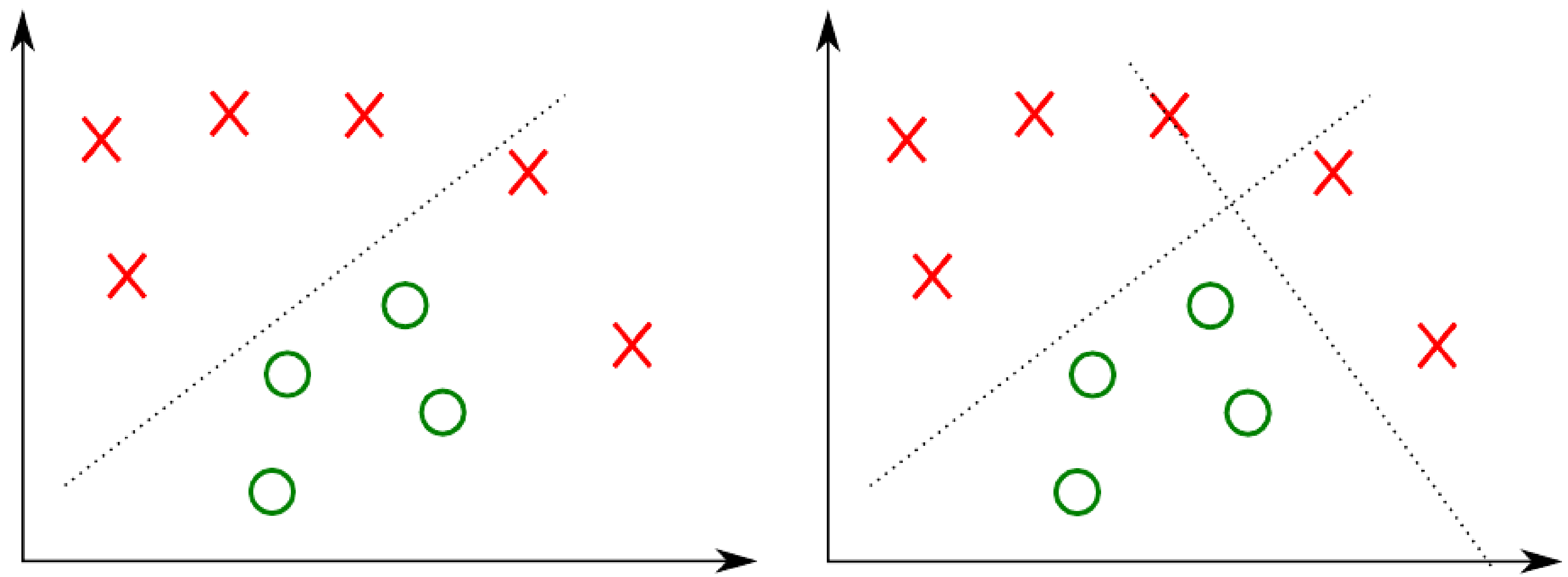
$$w_j \leftarrow w_j + \alpha \frac{\partial}{\partial w_j} \mathrm{Loss}(\mathbf{w})$$



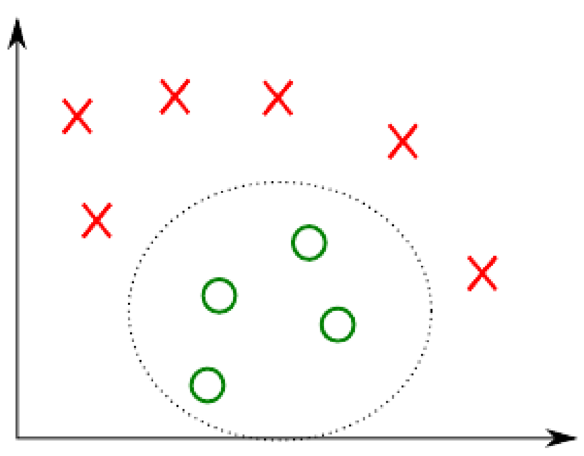
Adaptive α $\Rightarrow$**Simulated Annealing**

Major problem: **local minima**

Linearly non-separable input? Use multiple perceptrons

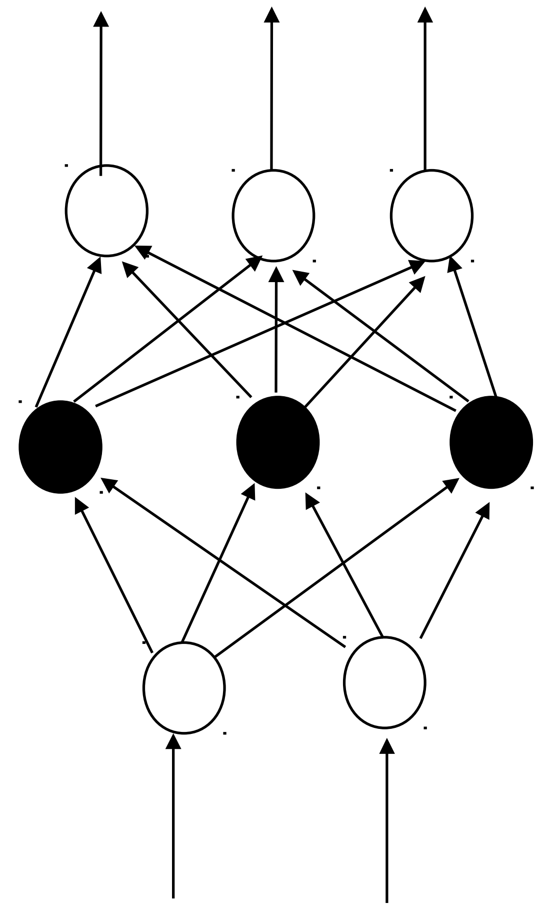


Advantage over SVM?
No need for kernels, although
Kernel Perceptron algorithm exists.



Neural Networks

- A neural network: A set of connected input/output units where each connection is associated with a **weight**
- Learning phase: adjusting the weights so as to predict the correct class label of the input tuples
  - Backpropagation
- From a statistical point of view, networks perform **nonlinear regression**



A Multi-Layer Feed-Forward Neural Network

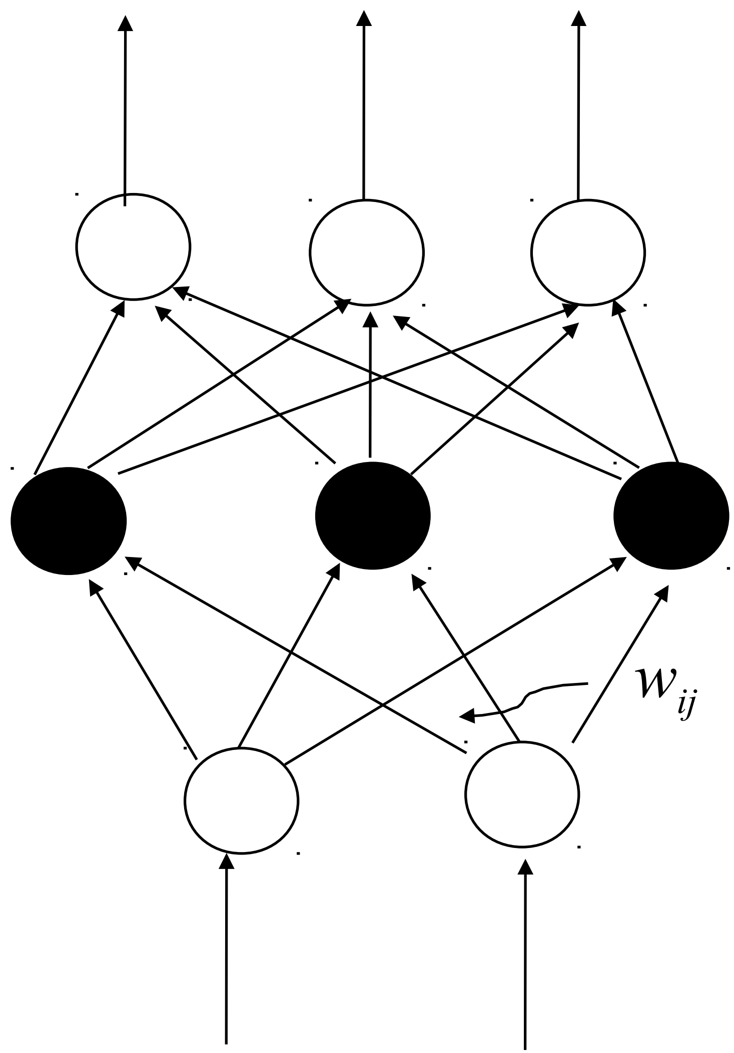
Output vector

Output layer

Hidden layer

Input layer

Input vector: *X*



A Multi-Layer Neural Network

- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction
- The network is **feed-forward** in that none of the weights cycles back to an input unit or to an output unit of a previous layer
- From a statistical point of view, networks perform **nonlinear regression**: Given enough hidden units and enough training samples, they can closely approximate any function

Defining a Network Topology

- First decide the **network topology:** # of units in the *input layer*, # of *hidden layers* (if > 1), # of units in *each hidden layer*, and # of units in the *output layer*
- Normalizing the input values for each attribute measured in the training tuples to [0.0—1.0]
- One **input** unit per domain value, each initialized to 0
- **Output**, if for classification and more than two classes, one output unit per class is used
- Once a network has been trained and its accuracy is **unacceptable**, repeat the training process with a *different network topology* or a *different set of initial weights*

Backpropagation

- For each training tuple, the weights are modified to **minimize the mean squared error** between the network's prediction and the actual target value
- Modifications are made in the “**backwards**” direction: from the output layer, through each hidden layer down to the first hidden layer, hence “**backpropagation**”
- Steps
  - Initialize weights (to small random #s) and biases in the network
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

A Multi-Layer Feed-Forward Neural Network

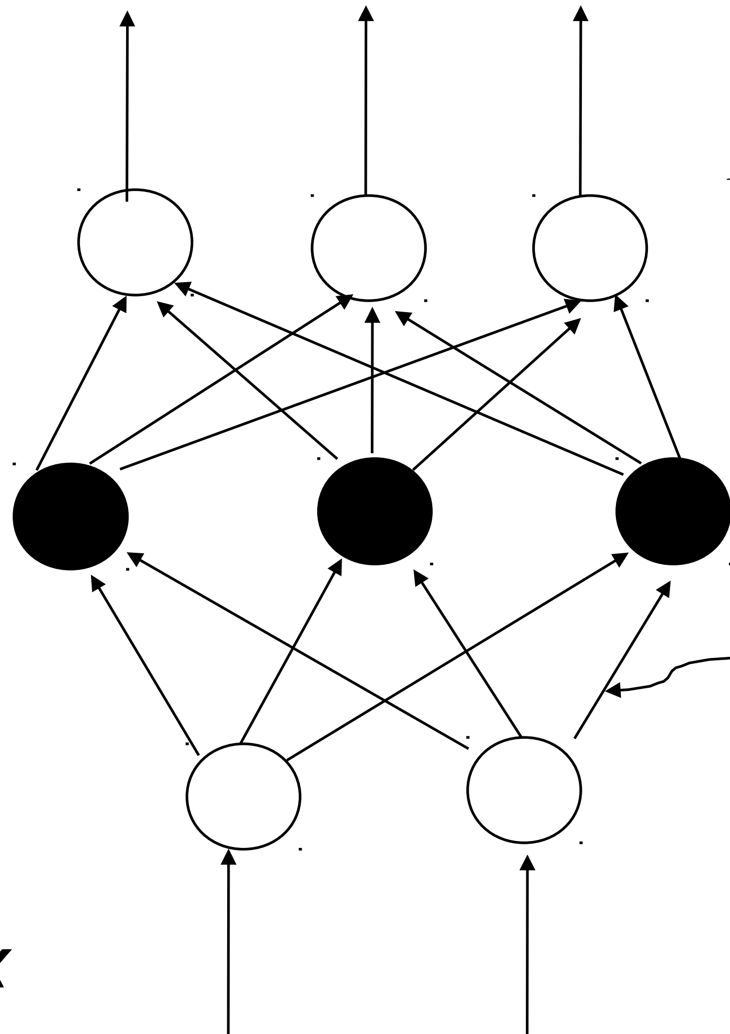
Output vector

Output layer

Hidden layer

Input layer

Input vector: *X*



w_{ij}

$$Err_j = O_j(1 - O_j)\sum_k Err_k w_{jk}$$

$$\theta_j = \theta_j + (l)\,Err_j$$

$$w_{ij} = w_{ij} + (l)\,Err_j O_i$$

$$Err_j = O_j(1 - O_j)(T_j - O_j)$$

$$O_j = \frac{1}{1 + e^{-I_j}}$$

$$I_j = \sum_i w_{ij} O_i + \theta_j$$

Backpropagation and Interpretability

- Efficiency of backpropagation: Each epoch (one interation through the training set) takes O(|D| * *w*), with |D| tuples and *w* weights, but # of epochs can be exponential to n, the number of inputs, in the worst case
- Rule extraction from networks: network pruning
  - Simplify the network structure by removing weighted links that have the least effect on the trained network
  - Then perform link, unit, or activation value clustering
  - The set of input and activation values are studied to derive rules describing the relationship between the input and hidden unit layers
- Sensitivity analysis: assess the impact that a given input variable has on a network output. The knowledge gained from this analysis can be represented in rules

Neural Network as a Classifier: Comments

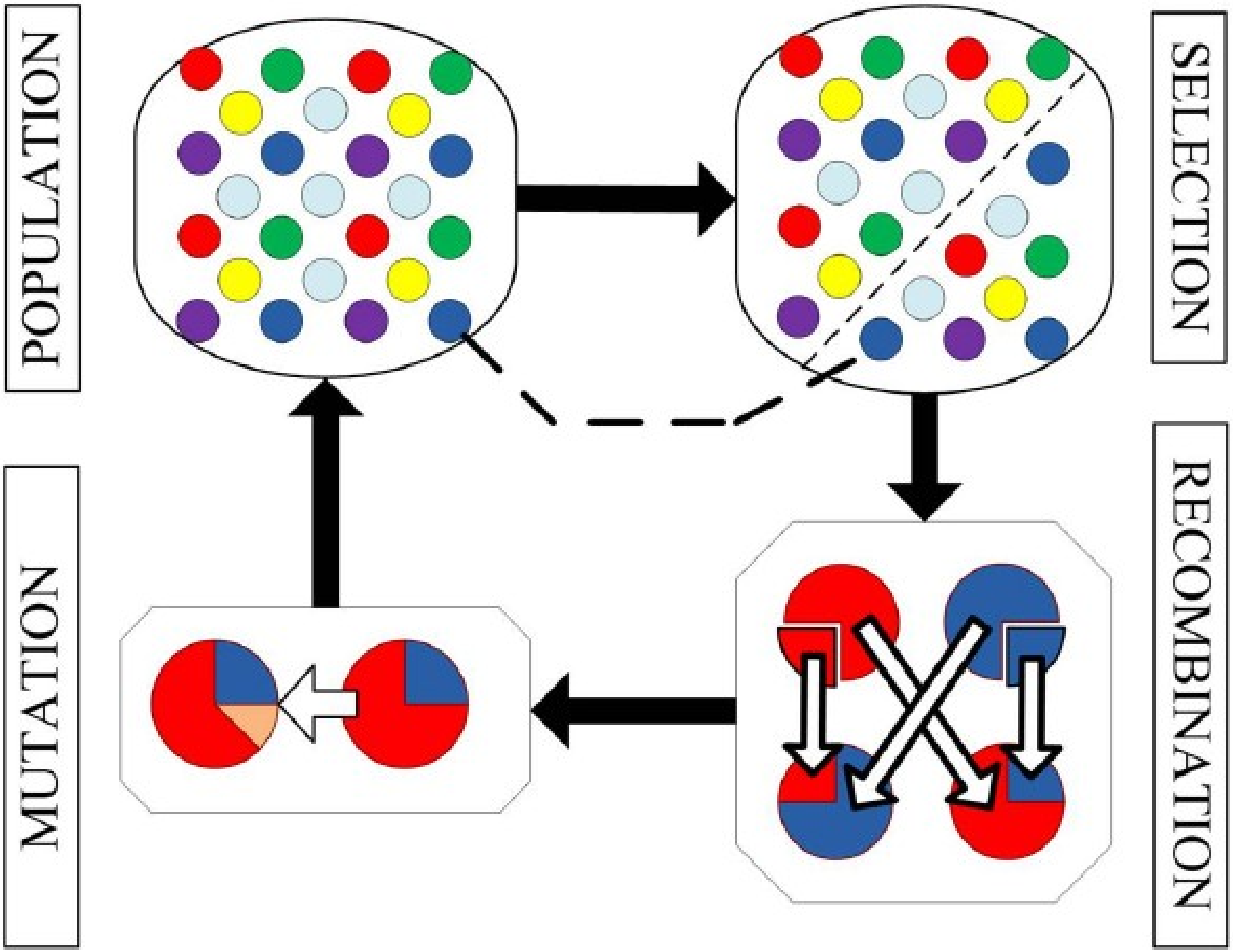
- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g., the network topology or ``structure."
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of ``hidden units" in the network
- Strength
  - High tolerance to noisy data
  - Ability to classify untrained patterns
  - Well-suited for continuous-valued inputs and outputs
  - Successful on a wide array of real-world data
  - Algorithms are inherently parallel
  - Techniques have recently been developed for the extraction of rules from trained neural networks

Other Classification Methods

- Rule based classification
- Neural networks
- Genetic algorithms
- Rough set approaches
- Fuzzy set approaches

Genetic Algorithms (GA)

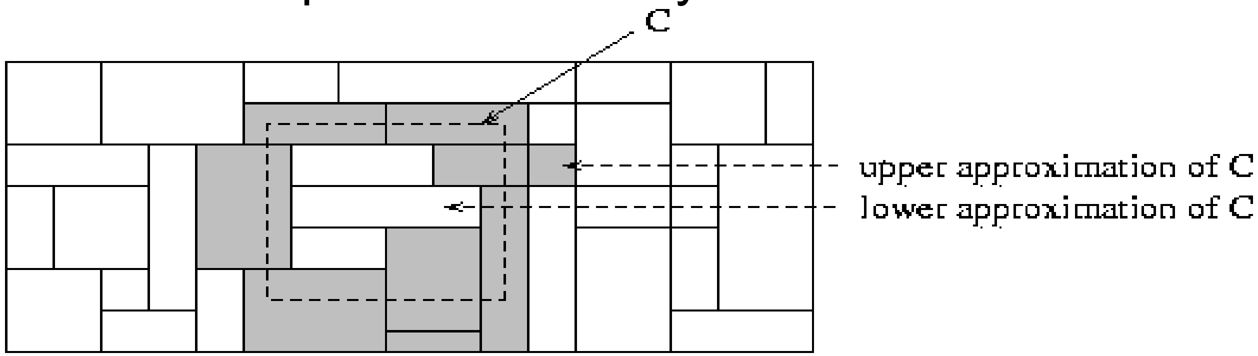
- Genetic Algorithm: based on an analogy to biological evolution
- An initial **population** is created consisting of randomly generated rules
  - Each rule is represented by a string of bits
  - E.g., if A_1 and $\neg A_2$ then C_2 can be encoded as 100
- Based on the notion of survival of the **fittest**, a new population is formed to consist of the fittest rules and their offsprings
- The fitness of a rule is represented by its *classification accuracy* on a set of training examples
- Offsprings are generated by *crossover* and *mutation*
- The process continues until a population P evolves *when each rule in P satisfies a prespecified threshold*
- Slow but easily parallelizable



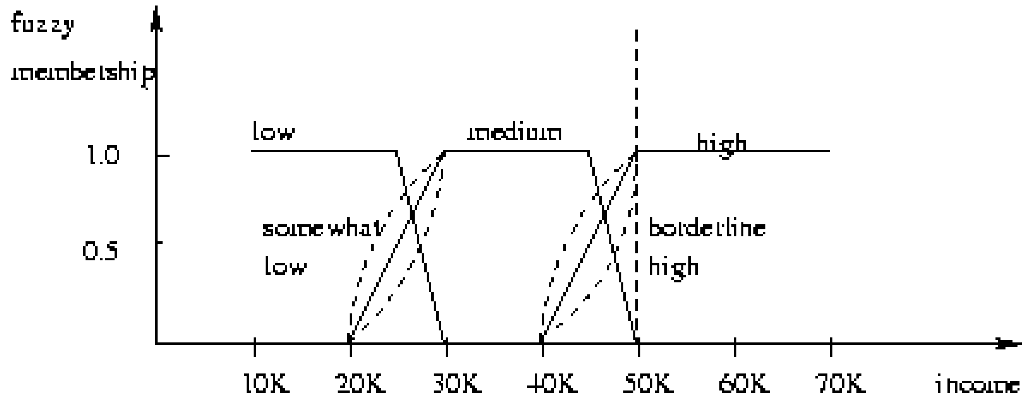
No local minima, but takes longer, must design problem well.

Rough Set Approach

- Rough sets are used to **approximately or “roughly” define equivalent classes**
- A rough set for a given class C is approximated by two sets: a lower approximation (certain to be in C) and an upper approximation (cannot be described as not belonging to C)
- Finding the minimal subsets (**reducts**) of attributes for feature reduction is NP-hard but a **discernibility matrix** (which stores the differences between attribute values for each pair of data tuples) is used to reduce the computation intensity



Fuzzy Set Approaches



- Fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership (such as using fuzzy membership graph)
- Attribute values are converted to fuzzy values
  - e.g., income is mapped into the discrete categories {low, medium, high} with fuzzy values calculated
- For a given new sample, more than one fuzzy value may apply
- Each applicable rule contributes a vote for membership in the categories
- Typically, the truth values for each predicted category are summed, and these sums are combined