Mining Classification Knowledge

Remarks on Non-Symbolic Methods

JERZY STEFANOWSKI
Institute of Computing Sciences,
Poznań University of Technology

COST Doctoral School, Troina 2008

Outline

1. Bayesian classification
2. K nearest neighbors
3. Linear discrimination
4. Artificial neural networks
5. Support vector machines

Bayesian Classification: Why?

- Probabilistic learning: Calculate explicit probabilities for hypothesis (decision), among the most practical approaches to certain types of learning problems.
- Probabilistic prediction: Predict multiple hypotheses, weighted by their probabilities.
- Standard: Even in cases where Bayesian methods prove computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured.

Bayesian Theorem

- Given training data *D, posteriori probability of a hypothesis h, P(h|D)* follows the Bayes theorem:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- MAP (maximum posteriori) hypothesis:

$$h_{MAP} \equiv \arg\max_{h \in H} P(h|D) = \arg\max_{h \in H} P(D|h)P(h).$$

- Practical difficulty: require initial knowledge of many probabilities, significant computational cost.

Naïve Bayes Classifier (I)

- A simplified assumption: attributes are conditionally independent:

$$P(C_j|V) \propto P(C_j)\prod_{i=1}^{n} P(v_i|C_j)$$

- Greatly reduces the computation cost, only count the class distribution.

Probabilities for weather data

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Probabilities for weather data

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

- A new day:

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Likelihood of the two classes

For “yes” = $2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$

For “no” = $3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0206$

Conversion into a probability by normalization:

P(“yes”) = 0.0053 / (0.0053 + 0.0206) = 0.205

P(“no”) = 0.0206 / (0.0053 + 0.0206) = 0.795

Missing values

- Training: instance is not included in frequency count for attribute value-class combination
- Classification: attribute will be omitted from calculation
- Example:

Outlook	Temp.	Humidity	Windy	Play
?	Cool	High	True	?

Likelihood of "yes" = 3/9 × 3/9 × 3/9 × 9/14 = 0.0238

Likelihood of "no" = 1/5 × 4/5 × 3/5 × 5/14 = 0.0343

P("yes") = 0.0238 / (0.0238 + 0.0343) = 41%

P("no") = 0.0343 / (0.0238 + 0.0343) = 59%

Naïve Bayes: discussion

- Naïve Bayes works surprisingly well (even if independence assumption is clearly violated)
- Why? Because classification doesn't require accurate probability estimates *as long as maximum probability is assigned to correct class*
- However: adding too many redundant attributes will cause problems (e.g. identical attributes)
- Note also: many numeric attributes are not normally distributed ($\rightarrow$ *kernel density estimators*)

Instance-Based Methods

- Instance-based learning: Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified.
- Typical approaches:
  - *k*-nearest neighbor approach:
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression:
    - Constructs local approximation.

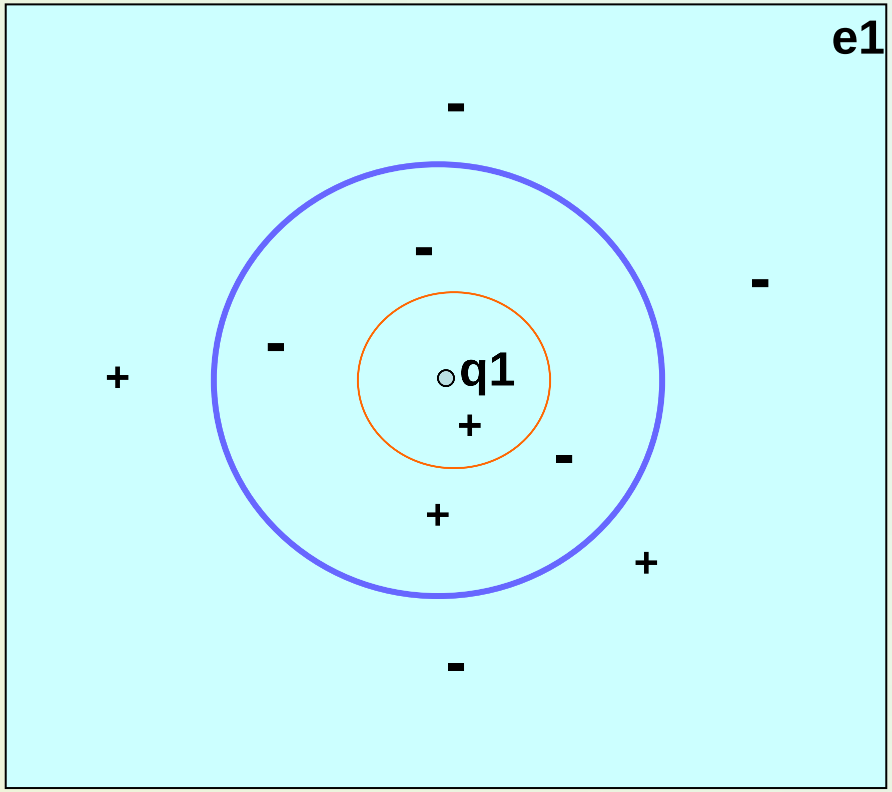
k-Nearest-Neighbor Algorithm

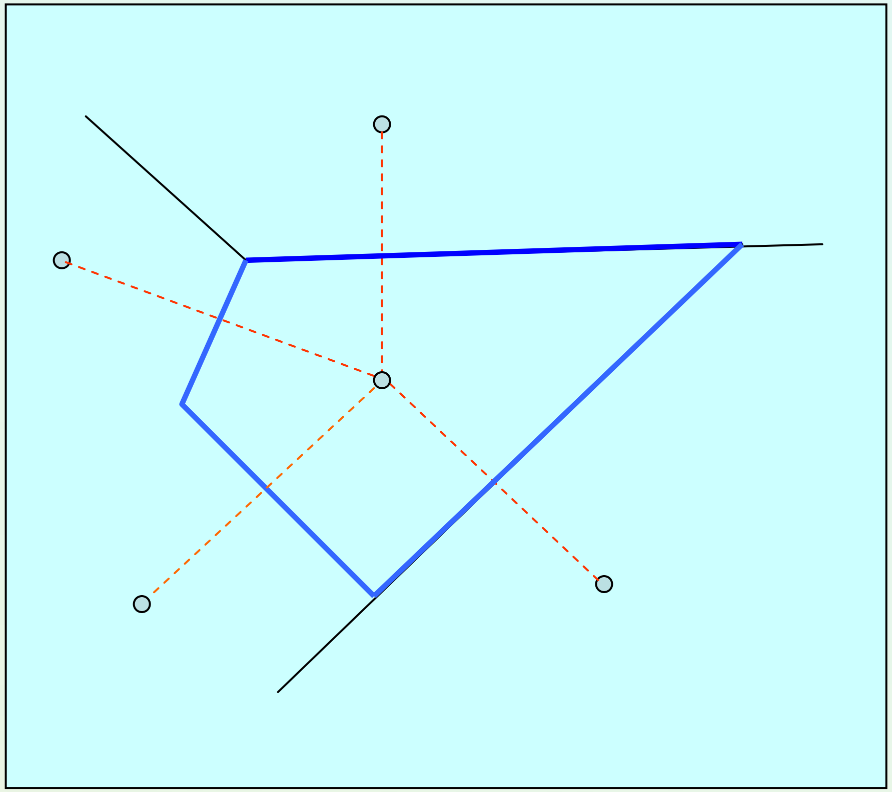
The case of discrete set of classes.

1. Take the instance x to be classified
2. Find k nearest neighbors of x in the training data.
3. Determine the class c of the majority of the instances among the k nearest neighbors.
4. Return the class c as the classification of x.

The distance functions are composed from difference metric d_a defined for each two instances x_i and x_j.

Classification & Decision Boundaries





1-nn: q1 is positive
5-nn: q1 is classified as negative

1-nn:

Discussion on the *k*-NN Algorithm

- The k-NN algorithm for continuous-valued target functions.
  - Calculate the mean values of the *k* nearest neighbors.
- Distance-weighted nearest neighbor algorithm.
  - Weight the contribution of each of the k neighbors according to their distance to the query point *xq*.
    - giving greater weight to closer neighbors: $w \equiv \frac{1}{d(x_q, x_i)^2}$
  - Similarly, we can distance-weight the instances for real-valued target functions.
- Robust to noisy data by averaging k-nearest neighbors.
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes. To overcome it,
  - axes stretch or elimination of the least relevant attributes.

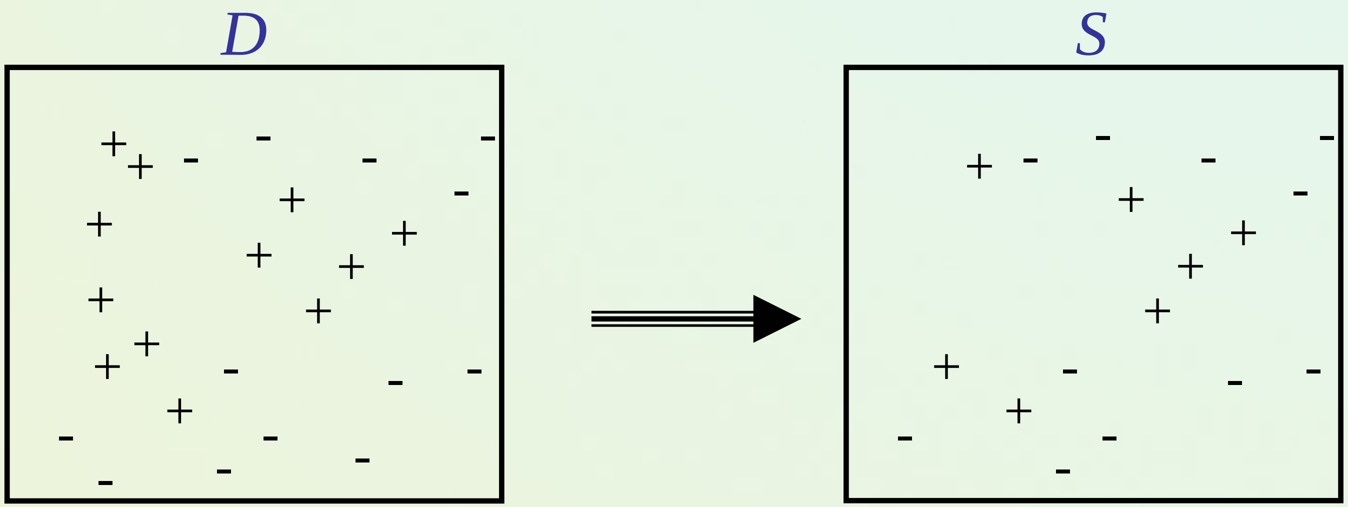
Disadvantages of the NN Algorithm

- the NN algorithm has large storage requirements because it has to store all the data;
- the NN algorithm is slow during instance classification because all the training instances have to be visited;
- the accuracy of the NN algorithm degrades with increase of noise in the training data;
- the accuracy of the NN algorithm degrades with increase of irrelevant attributes.

Condensed NN Algorithm

The Condensed NN algorithm was introduced to reduce the storage requirements of the NN algorithm.

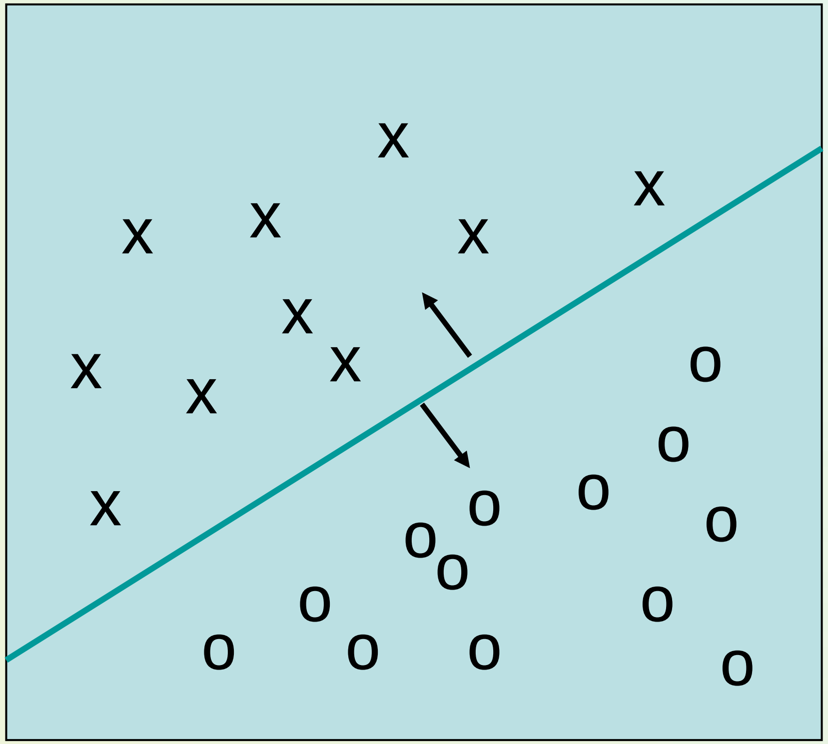
The algorithm finds a subset *S* of the training data *D* s.t. each instance in *D* can be correctly classified by the NN algorithm applied on the subset *S*. *The average reduction of the algorithm varies between 60% to 80%.*



Remarks on Lazy vs. Eager Learning

- Instance-based learning: lazy evaluation
- Decision-tree and Bayesian classification: eager evaluation
- Key differences
  - Lazy method may consider query instance *xq* when deciding how to generalize beyond the training data *D*
  - Eager method cannot since they have already chosen global approximation when seeing the query
- Efficiency: Lazy - less time training but more time predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

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

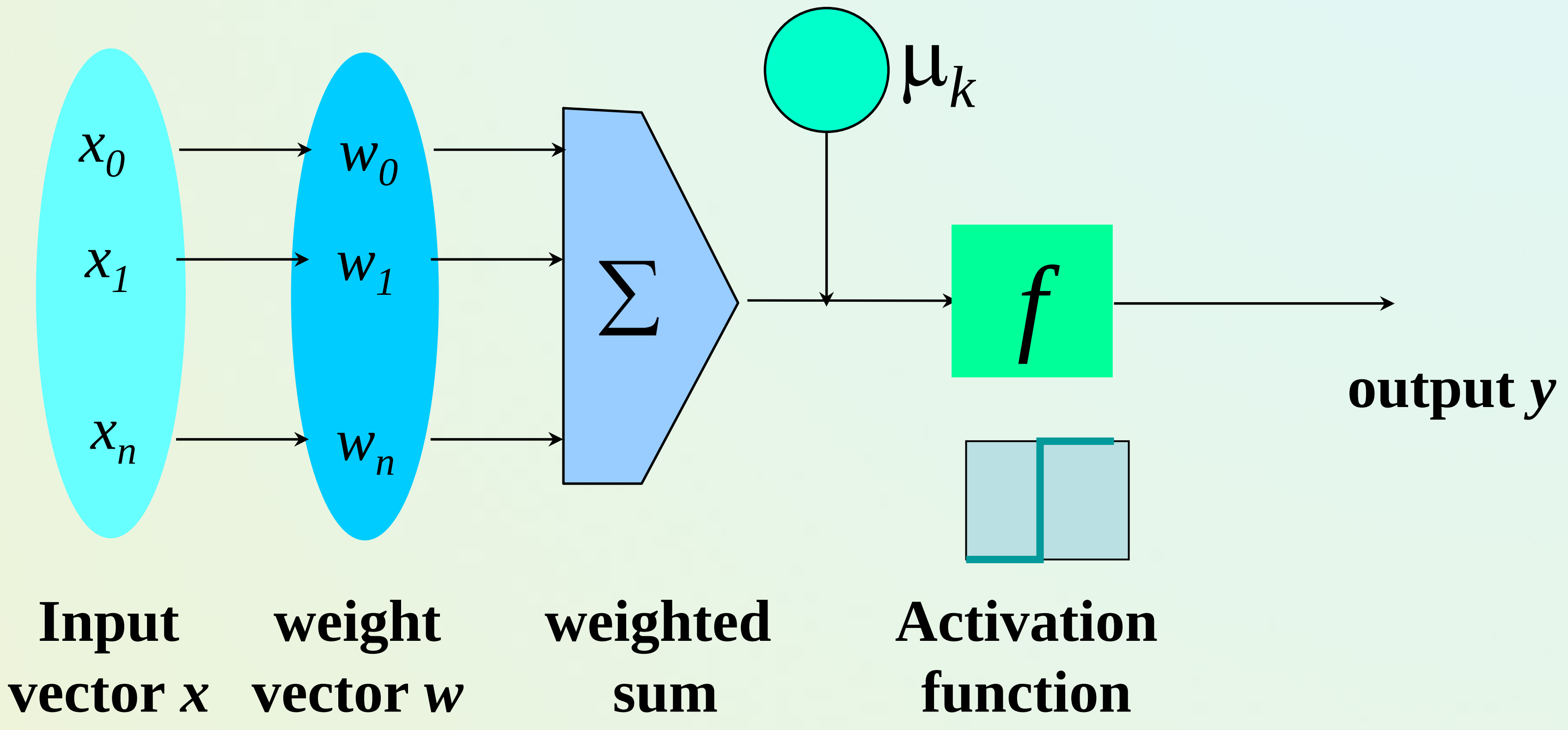
Discriminative Classifiers

- 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
- Criticism
  - long training time
  - difficult to understand the learned function (weights)
  - not easy to incorporate domain knowledge
    - (easy in the form of priors on the data or distributions)

Neural Networks

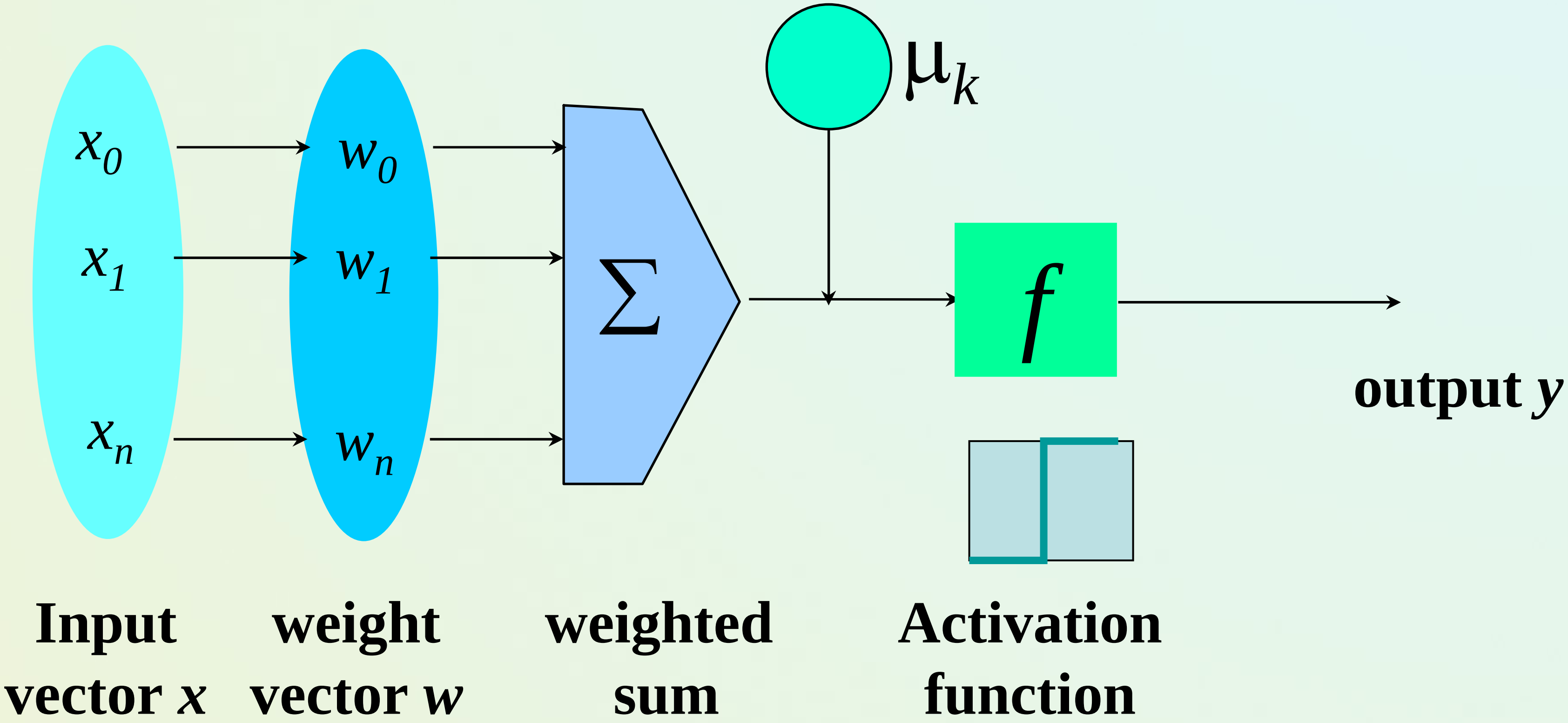
- Analogy to Biological Systems (Indeed a great example of a good learning system)
- Massive Parallelism allowing for computational efficiency
- The first learning algorithm came in 1959 (Rosenblatt) who suggested that if a target output value is provided for a single neuron with fixed inputs, one can incrementally change weights to learn to produce these outputs using the perceptron learning rule

A Neuron



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

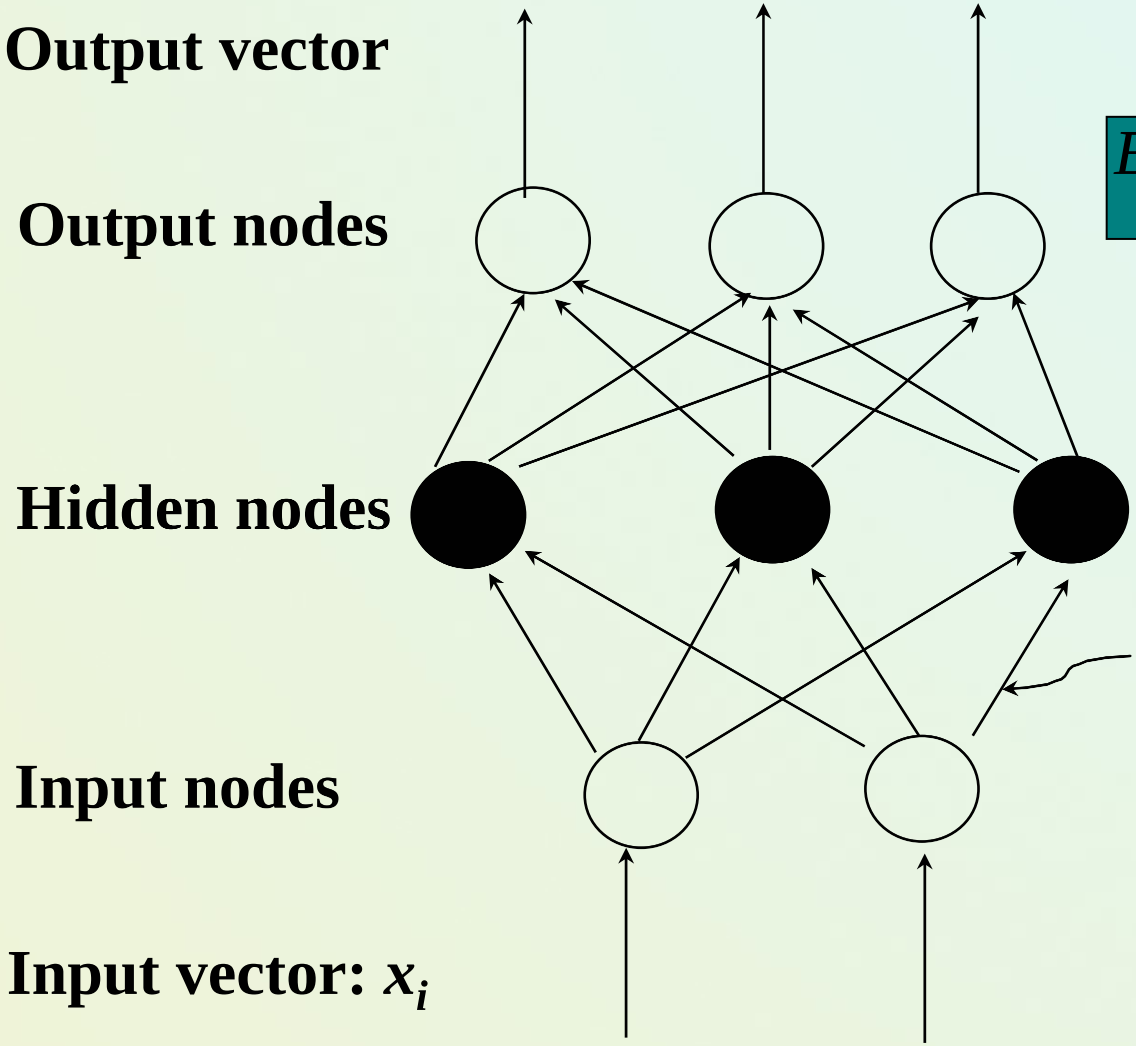
A Neuron



For Example

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

Multi-Layer Perceptron



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

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

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

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

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

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

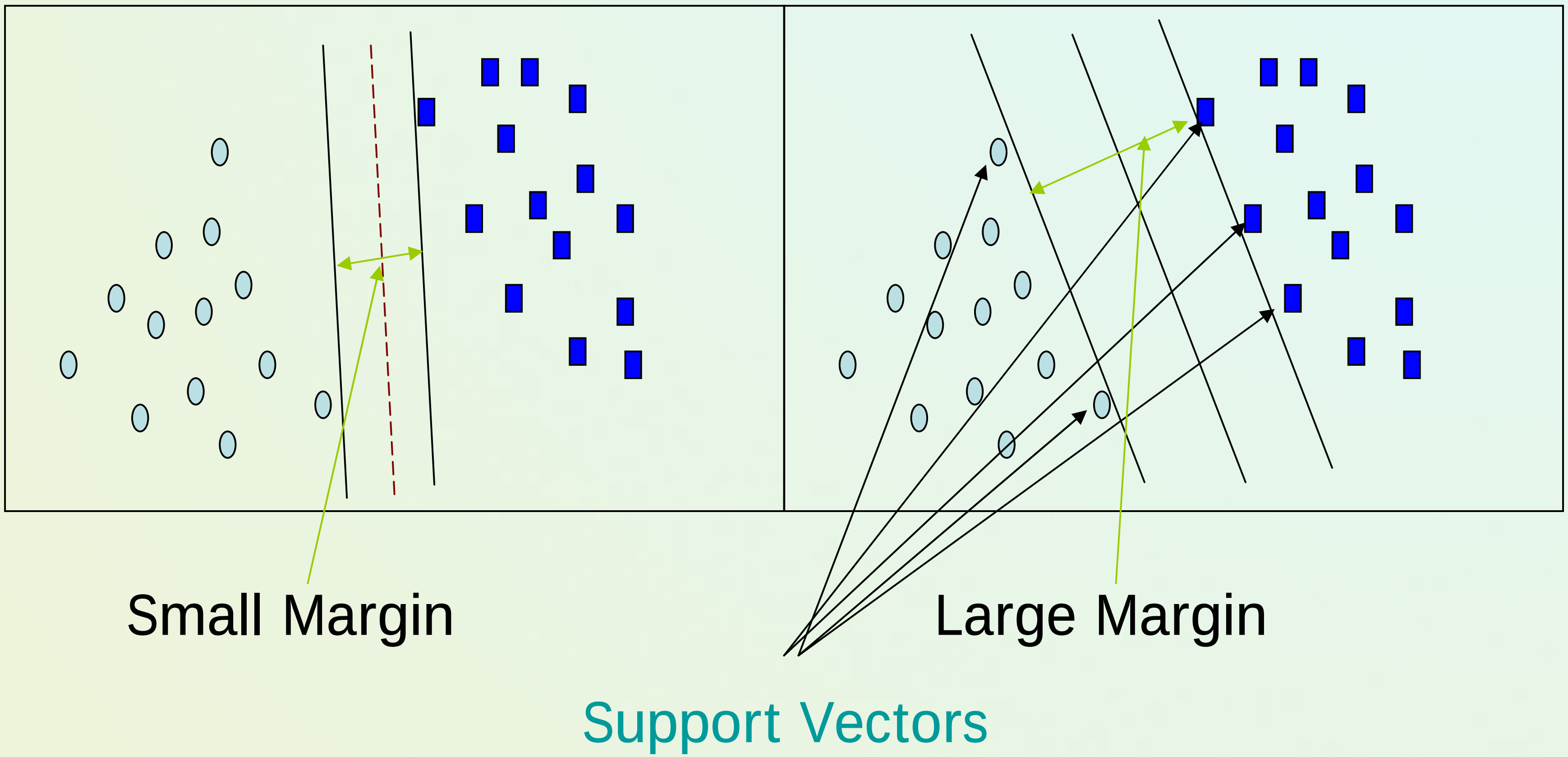
Network Training

- The ultimate objective of training
  - obtain a set of weights that makes almost all the examples in the training data classified correctly
- Steps:
  - Initial weights are set randomly
  - Input examples are fed into the network one by one
  - Activation values for the hidden nodes are computed
  - Output vector can be computed after the activation values of all hidden node are available
  - Weights are adjusted using error
    (desired output - actual output)

Neural Networks

- Advantages
  - prediction accuracy is generally high
  - robust, works when training examples contain errors
  - output may be discrete, real-valued, or a vector of several discrete or real-valued attributes
  - fast evaluation of the learned target function.
- Criticism
  - long training time
  - difficult to understand the learned function (weights).
  - not easy to incorporate domain knowledge

SVM – Support Vector Machines



SVM – Cont.

- Linear Support Vector Machine

Given a set of points $x_i \in \Re^n$ with label $\mathrm{y_i} \in \{-1,1\}$

The SVM finds a hyperplane defined by the pair $(\mathrm{w,b})$

(where w is the normal to the plane and b is the distance from the origin)

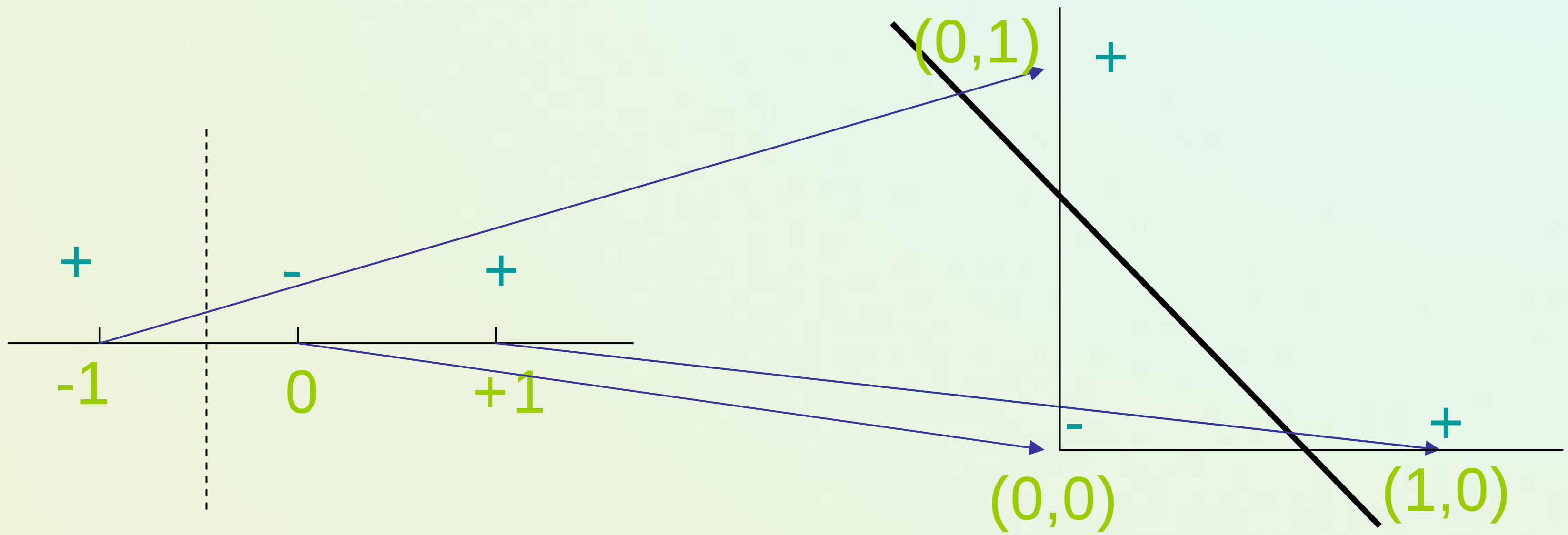
s.t.

$$y_i(x_i \cdot w + b) \geq +1 \quad i = 1,..., N$$

x – feature vector, b- bias, y- class label, ||w|| - margin

SVM – Cont.

- What if the data is not linearly separable?
- Project the data to high dimensional space where it is linearly separable and then we can use linear SVM – (Using Kernels)



Non-Linear SVM

Classification using SVM (w,b)

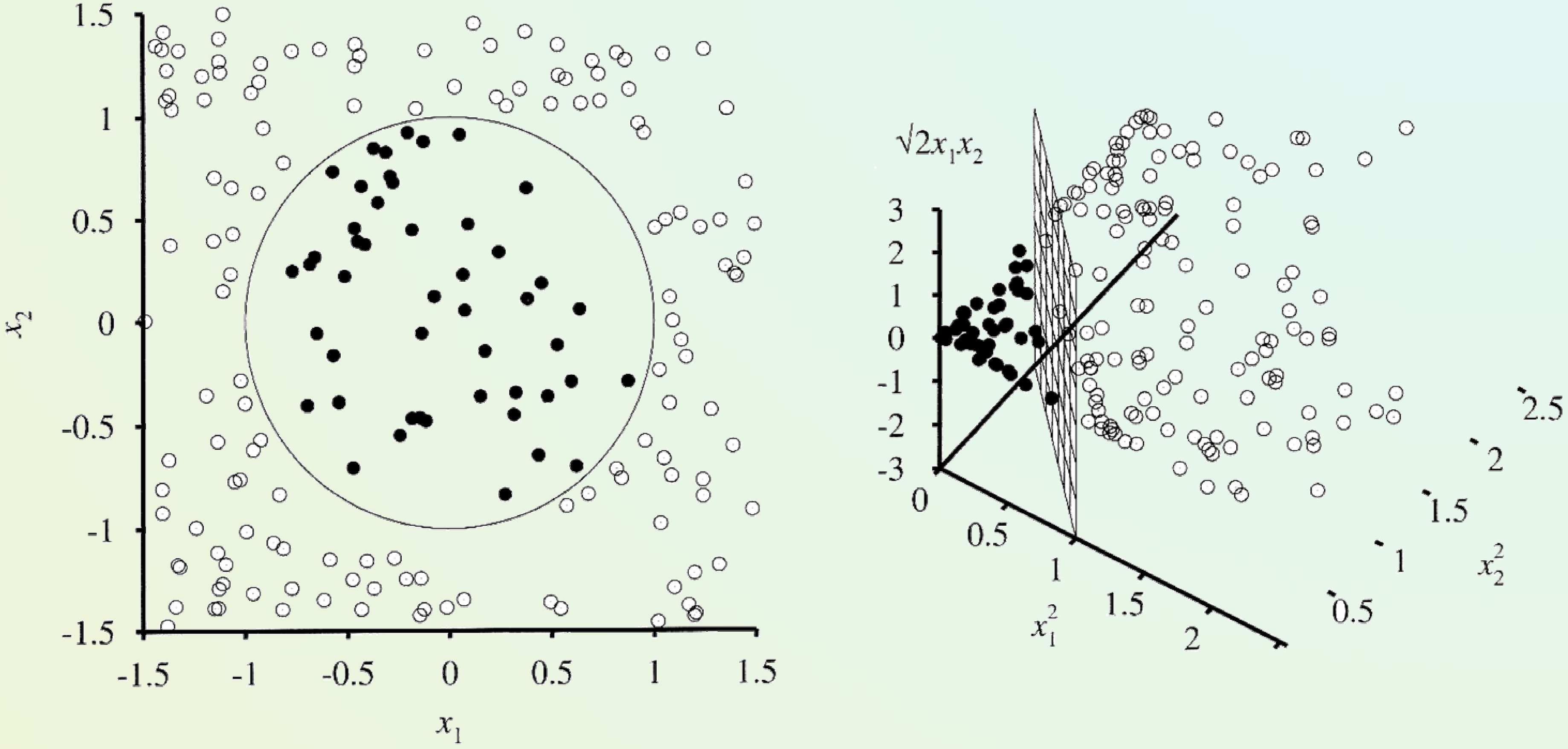
$$x_i \cdot w + b \overset{?}{>} 0$$

In non linear case we can see this as

$$K(x_i, w) + b \overset{?}{>} 0$$

Kernel – Can be thought of as doing dot product in some high dimensional space

Changing Attribute Space



SVM vs. Neural Network

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