Mining Classification Knowledge

Remarks on Non-Symbolic Methods

JERZY STEFANOWSKI
Institute of Computing Sciences,
Poznań University of Technology

SE lecture - revision 2013

Outline

1. Bayesian classification
2. K nearest neighbors
3. Linear discrimination
4. Artificial neural networks
5. Other remarks

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
- Applications: Quite effective in some problems, e.g. text classification
- Good mathematical background and reference point to other methods

Bayesian Theorem: Basics

- Let X be a data sample whose class label is unknown
- Let H be a hypothesis that X belongs to class C
- For classification problems, determine P(H/X): the probability that the hypothesis holds given the observed data sample X
- P(H): prior probability of hypothesis H (i.e. the initial probability before we observe any data, reflects the background knowledge)
- P(X): probability that sample data is observed
- P(X|H) : probability of observing the sample X, given that the hypothesis holds

Bayesian Theorem

- Given training data *X*, *posteriori probability of a hypothesis H, P(H|X)* follows the Bayes theorem

$$P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)}$$

- Informally, this can be written as

  posterior =likelihood x prior / evidence

- MAP (maximum posteriori) hypothesis

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

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

Bayesian Classifiers

- Consider each attribute and class label as random variables

  Given a record with attributes (A_1, A_2,…,A_n)

  - Goal is to predict class C
  - Specifically, we want to find the value of C that maximizes P(C| A_1, A_2,…,A_n)

- Can we estimate P(C| A_1, A_2,…,A_n) directly from data?

Bayesian Classifiers

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes theorem

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

  - Choose value of C that maximizes
    $P(C \mid A_1, A_2, \ldots, A_n)$
  - Equivalent to choosing value of C that maximizes
    $P(A_1, A_2, \ldots, A_n \mid C)\, P(C)$

- How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?

Naïve Bayes Classifier

- Assume independence among attributes A_i when class is given:
  - $P(A_1, A_2, \ldots, A_n \mid C) = P(A_1 \mid C_j)\, P(A_2 \mid C_j) \ldots P(A_n \mid C_j)$
  - Can estimate $P(A_i \mid C_j)$ for all A_i and C_j.
  - New point is classified to C_j

    if $P(C_j) \prod P(A_i \mid C_j)$ is maximal.
- Greatly reduces requirements to collect enough data , 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” = $3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0238$

Likelihood of “no” = $1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0343$

P(“yes”) = 0.0238 / (0.0238 + 0.0343) = 41%

P(“no”) = 0.0343 / (0.0238 + 0.0343) = 59%

Naïve Bayes: discussion

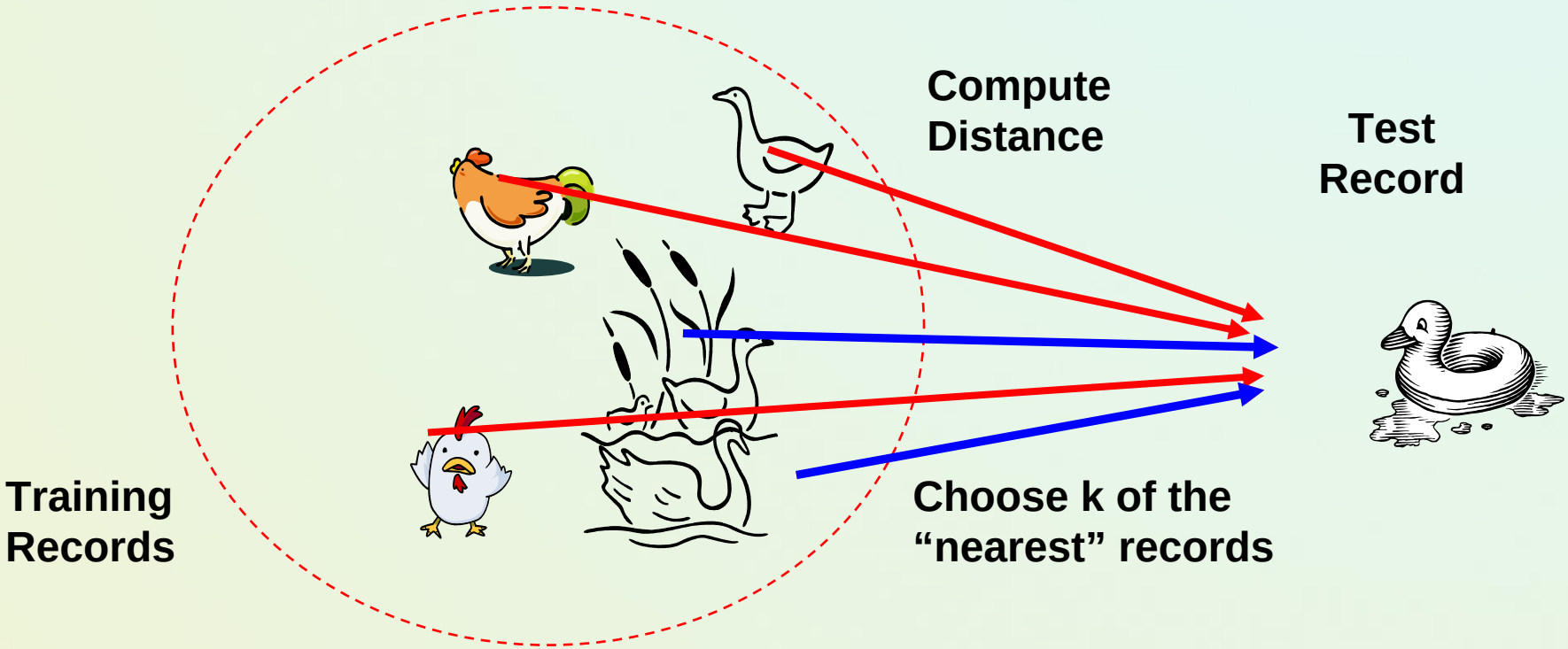
- Naïve Bayes works surprisingly well (even if independence assumption is clearly violated)
- Why? Because classification doesn't require accurate probability estimates *as long as maximum probability is assigned to correct class*
- However: adding too many redundant attributes will cause problems (e.g. 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.

Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



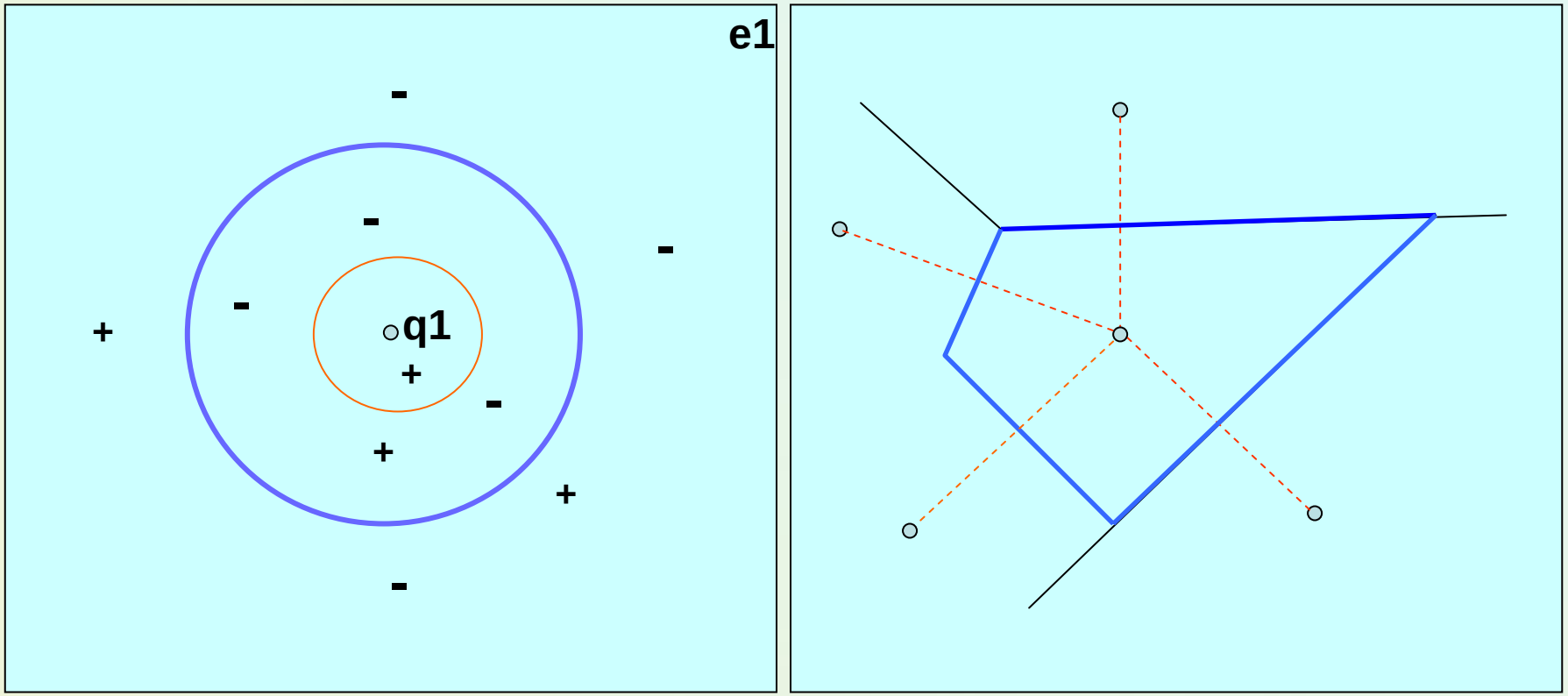
k-Nearest-Neighbor Algorithm

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:

The distance function

- Simplest case: one numeric attribute
  - Distance is the difference between the two attribute values involved (or a function thereof)
- Several numeric attributes: normally, Euclidean distance is used and attributes are normalized
- Nominal attributes: distance is set to 1 if values are different, 0 if they are equal
- Are all attributes equally important?
  - Weighting the attributes might be necessary

Instance-based learning

- Distance function defines what's learned
- Most instance-based schemes use *Euclidean distance*:

$$\sqrt{(a_1^{(1)} - a_1^{(2)})^2 + (a_2^{(1)} - a_2^{(2)})^2 + \ldots + (a_k^{(1)} - a_k^{(2)})^2}$$

  $\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$: two instances with *k* attributes

- Taking the square root is not required when comparing distances
- Other popular metric: *city-block (Manhattan) metric*
  - Adds differences without squaring them

Normalization and other issues

- Different attributes are measured on different scales
  $\Rightarrow$ need to be *normalized*:

$$a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i} \qquad \text{or} \qquad a_i = \frac{v_i - Avg(v_i)}{StDev(v_i)}$$

  v_i : the actual value of attribute *i*

- Nominal attributes: distance either 0 or 1
- Common policy for missing values: assumed to be maximally distant (given normalized attributes)

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.

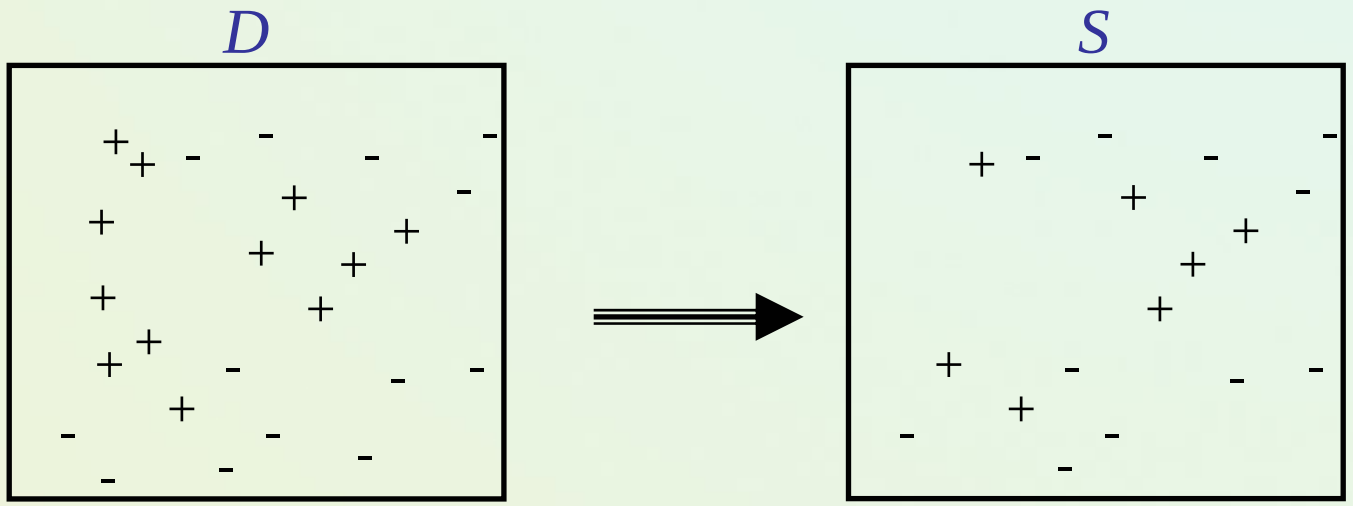
Disadvantages of the NN Algorithm

- 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.

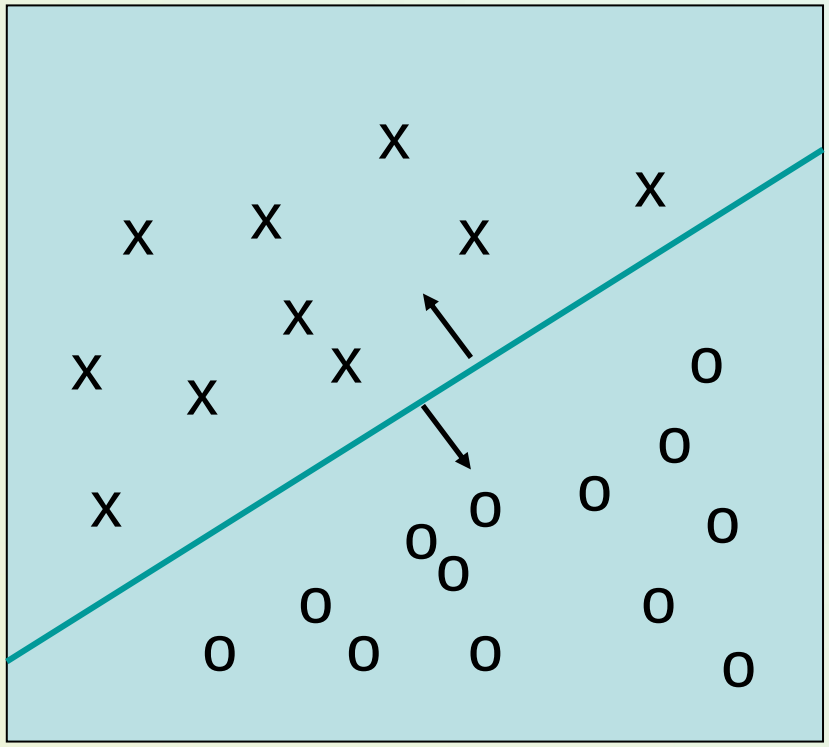
The algorithm finds a subset *S* of the training data *D* s.t. each instance in *D* can be correctly classified by the NN algorithm applied on the subset *S*. *The average reduction of the algorithm varies between 60% to 80%.*



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 blue line belongs to class ‘x’
- The data below red line belongs to class ‘o’
- Examples – SVM, Perceptron, Probabilistic Classifiers

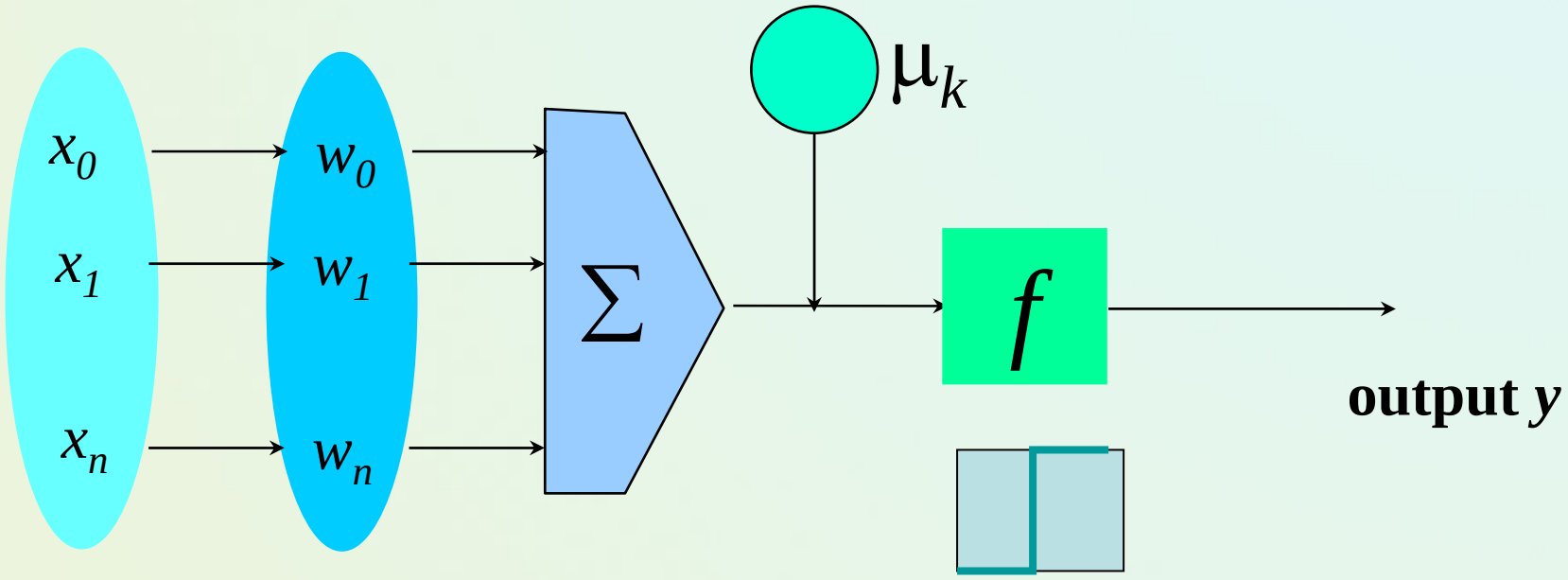
Discriminative Classifiers

- 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

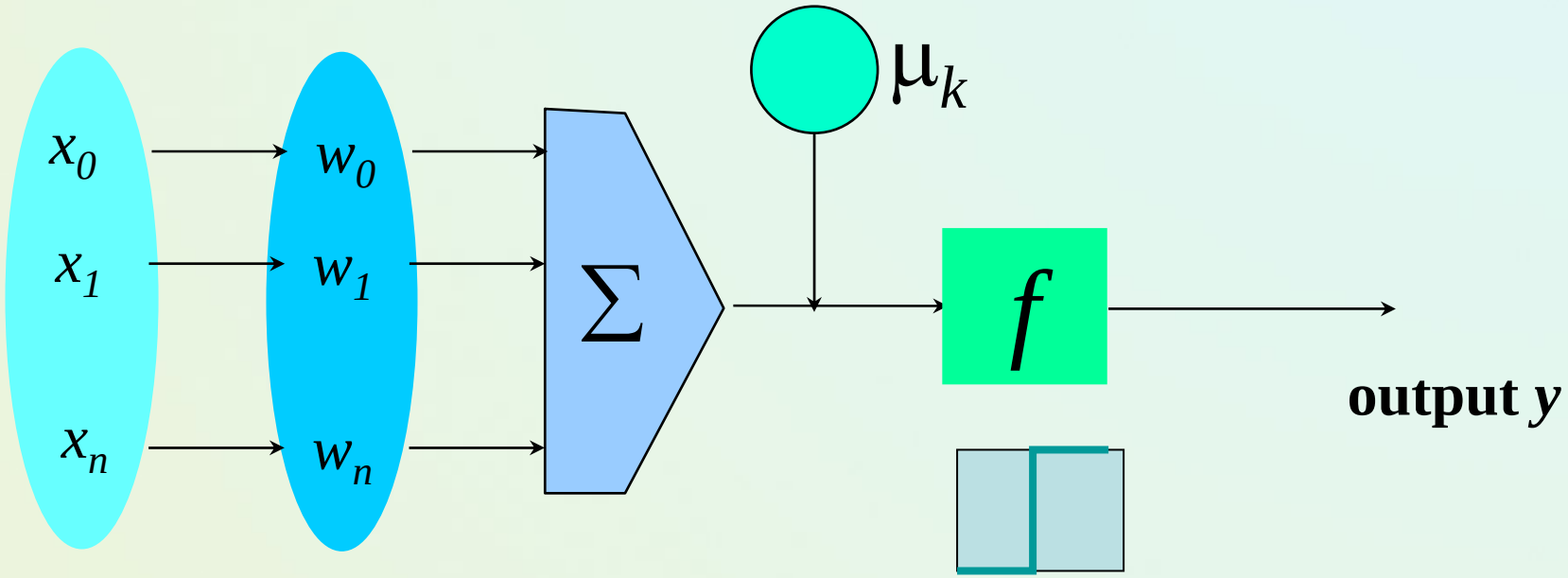
A Neuron



Input vector x **weight vector w** **weighted sum** **Activation function**

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

A Neuron



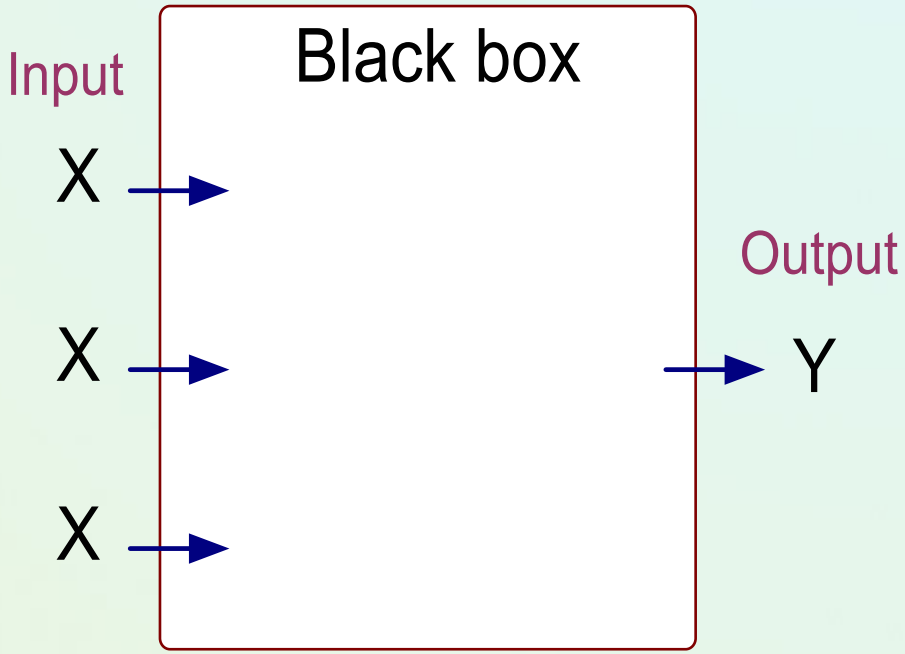
Input vector *x* **weight vector *w*** **weighted sum** **Activation function**

For Example

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

Artificial Neural Networks (ANN)

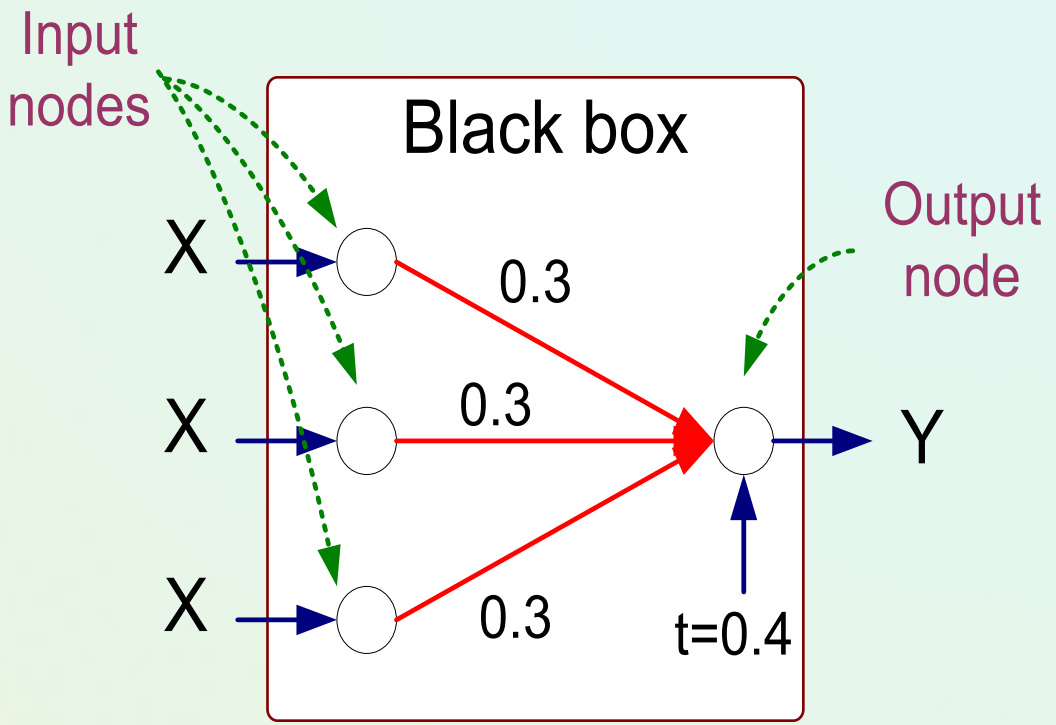
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output Y is 1 if at least two of the three inputs are equal to 1.

Artificial Neural Networks (ANN)

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0

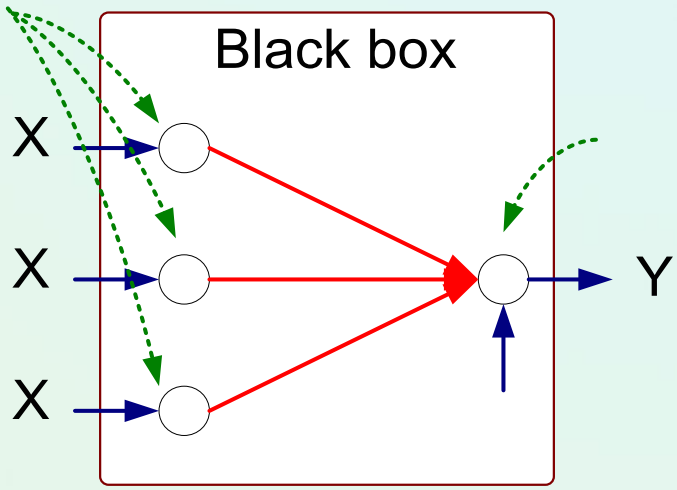


$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Artificial Neural Networks (ANN)

- Model is an assembly of inter-connected nodes and weighted links
- Output node sums up each of its input value according to the weights of its links
- Compare output node against some threshold t

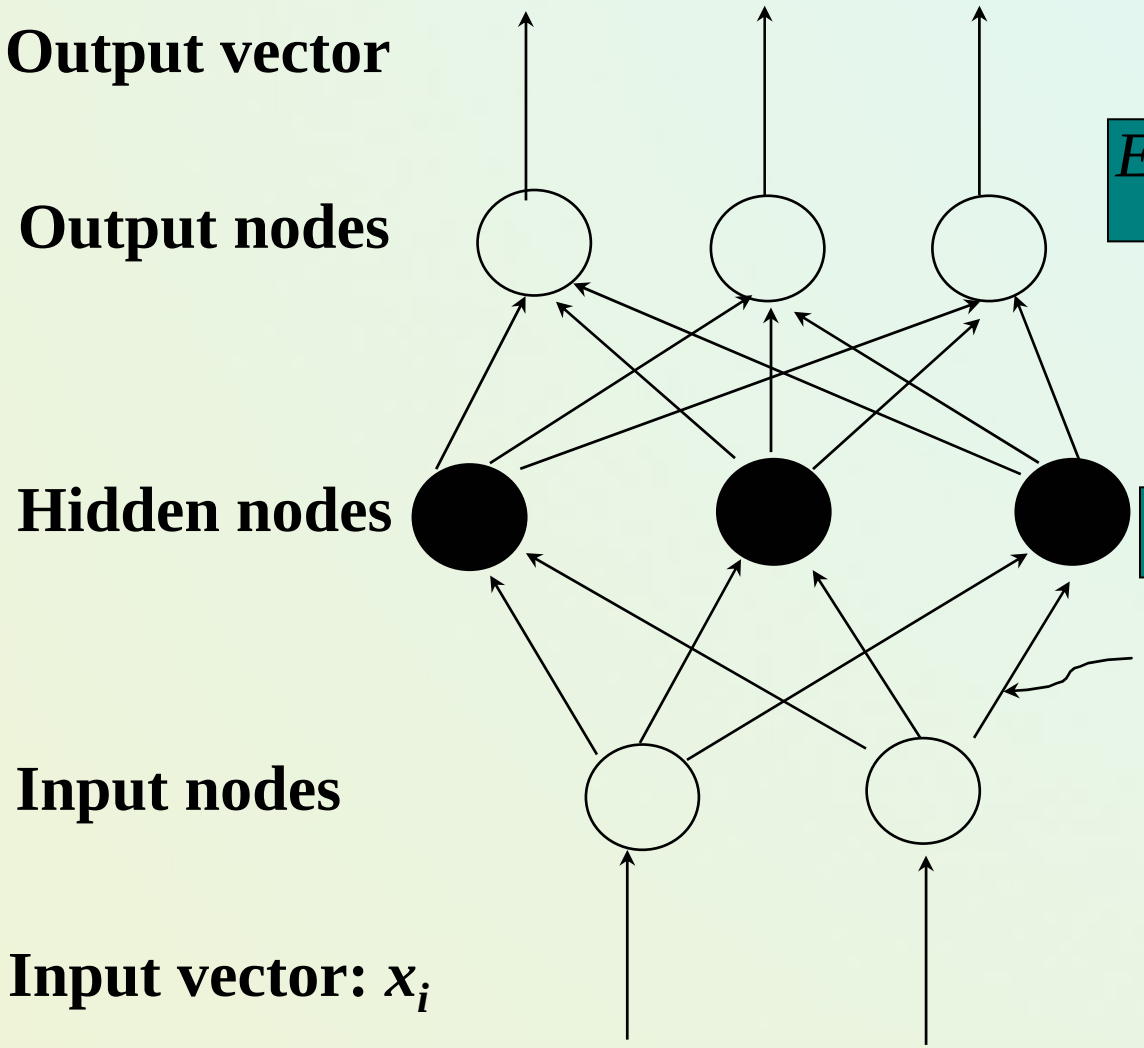


Perceptron Model

$$Y = I(\sum_i w_i X_i - t) \quad \text{or}$$

$$Y = sign(\sum_i w_i X_i - t)$$

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