

# Rozpoznávanie obrazcov

Sonka, Hlavac, Boyle: Image Processing, Analysis and Machine vision,  
kapitola: Object recognition

<http://dip.sccg.sk/>

# Rozpoznávanie obrazcov

- Rozpoznávanie obrazcov sa používa na **klasifikáciu** oblastí a objektov a predstavuje dôležitú súčasť zložitejších systémov počítačového videnia.
- Žiadne rozpoznávanie nie je možné bez znalostí. Vyžadujú sa špecifikácie tak o objektoch, ktoré sa rozpoznávajú ako aj všeobecnejšie znalosti o triedach objektov.

# Prístupy rozpoznávania obrazcov

- Štatistický založený na pochopení štatistického modelu obrazca a tried obrazcov
- Neurónové siete: klasifikátor je reprezentovaný ako siet' buniek modelujúcich neuróny ľudského mozgu
- Syntaktický (štrukturálny): triedy obrazcov reprezentované cez formálnu štruktúru ako gramatiky automaty ret'azce atď.

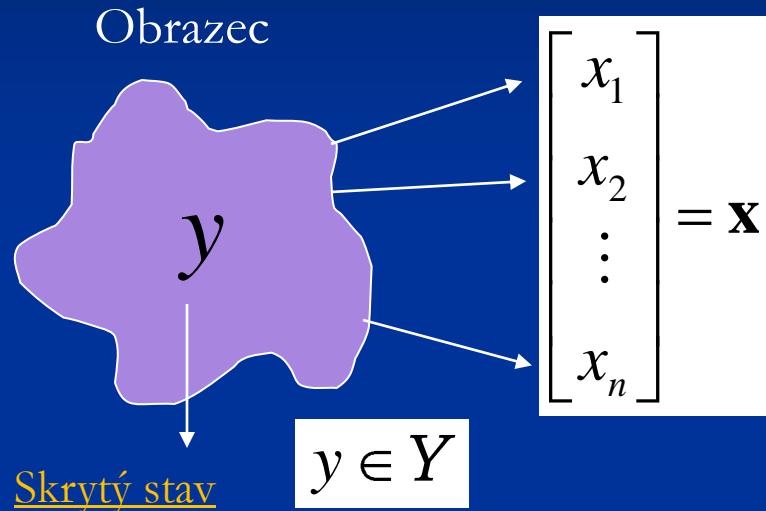
# Základné pojmy

- Obrazec (**pattern**) je objekt, proces alebo udalosť, ktorú vieme pomenovať.
- Trieda obrazcov (**pattern class**) je množina obrazcov s podobnými atribútmi a obvykle pochádzajú s rovnakého zdroja
- Počas rozpoznávania (klasifikácie) daný objekt (obrazec) je priradený do jednej triedy

# Štatistické rozpoznávanie obrazcov

- Rozhodovanie na základe Bayesovo pravidla
- Support Vector Machines
- Zhlukovacie metódy

# Základný koncept



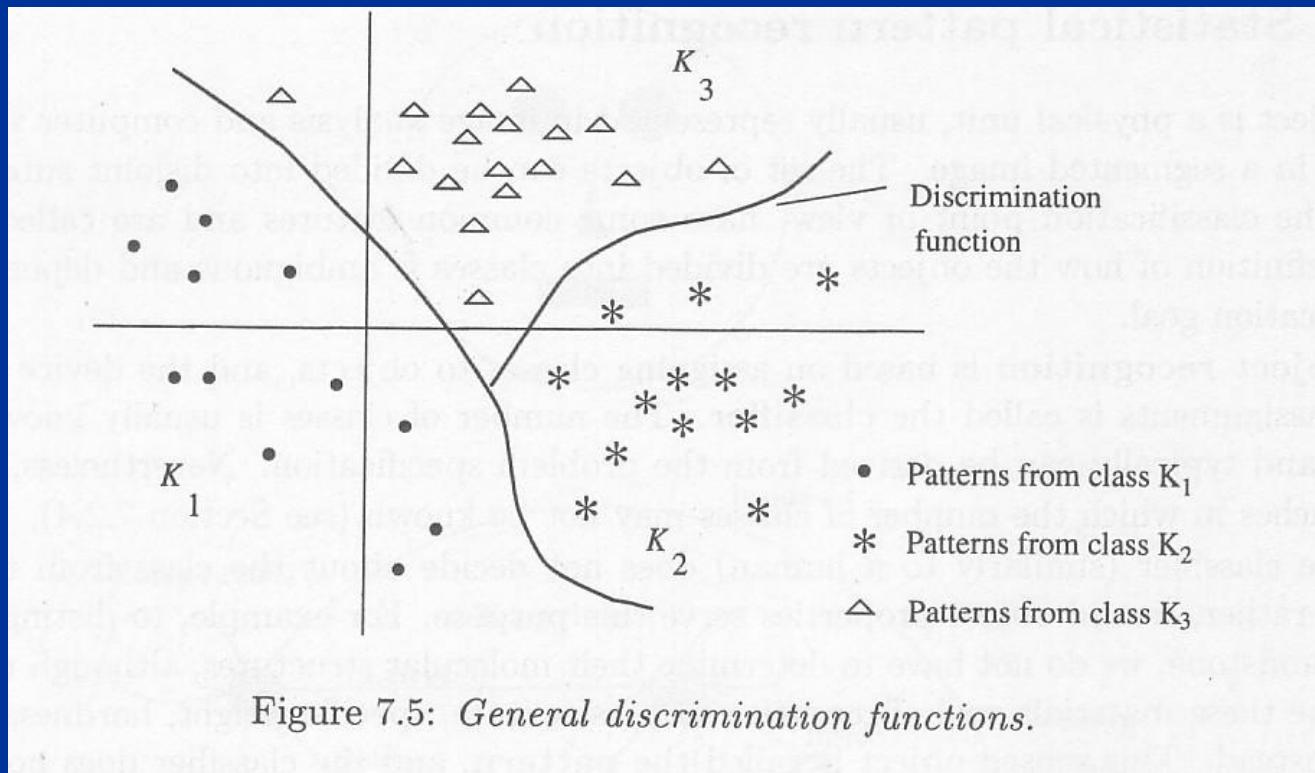
Vektor príznakov  $\mathbf{x} \in X$

- Vektor pozorovaní (merania).
- $\mathbf{x}$  je bod v priestore príznakov  $X$

- Nemôže byť priamo pozorovaný.
- Obrazce s rovnakým skrytým stavom patria do rovnakej triedy
- Počet tried sa obyčajne vie dopredu.
- Určiť klasifikátor (rozhodovacie pravidlo), ktorý rozhodne o skrytom stave na základe pozorovania.

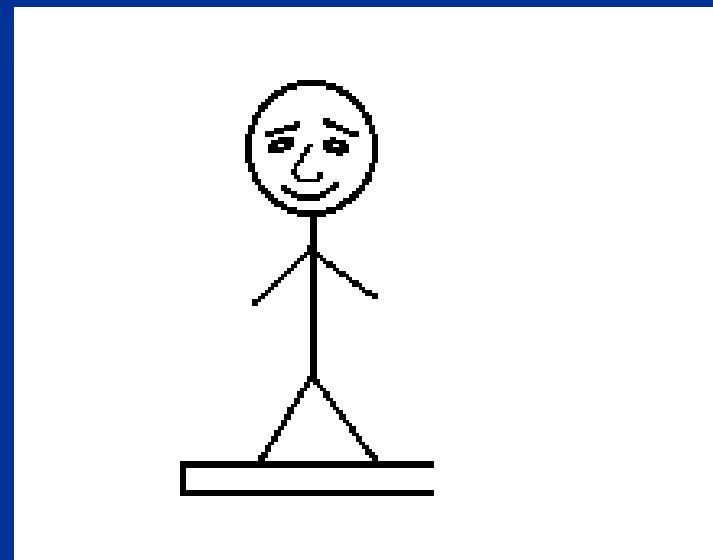
$$q : X \rightarrow Y$$

Triedy obrazcov vytvárajú v príznakovom priestore zhľuky, ktoré je možné rozdeliť diskriminačnými (oddelujúcimi) hyperkrivkami.



# Príklad

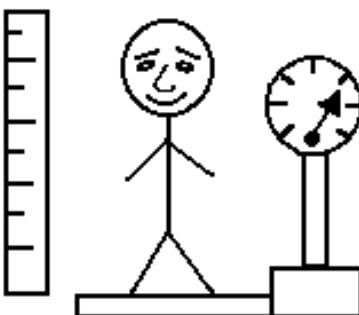
Rozpoznávanie džokej – basketbalista.



# Príklad

Rozpoznávanie džokej – basketbalista.

výška



$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{x}$$

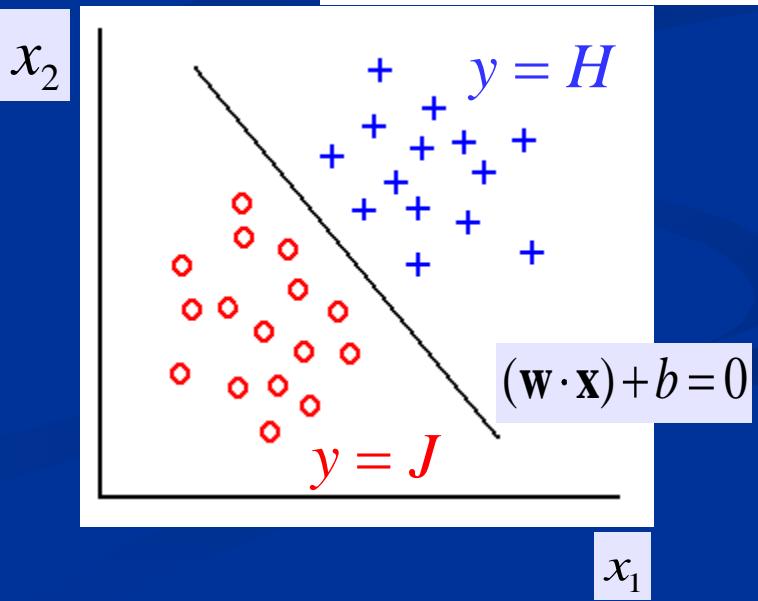
Množina skrytých stavov  $Y = \{H, J\}$

Priestor príznakov  $X = \mathcal{R}^2$

Lineárny klasifikátor

$$q(\mathbf{x}) = \begin{cases} H & \text{if } (\mathbf{w} \cdot \mathbf{x}) + b \geq 0 \\ J & \text{if } (\mathbf{w} \cdot \mathbf{x}) + b < 0 \end{cases}$$

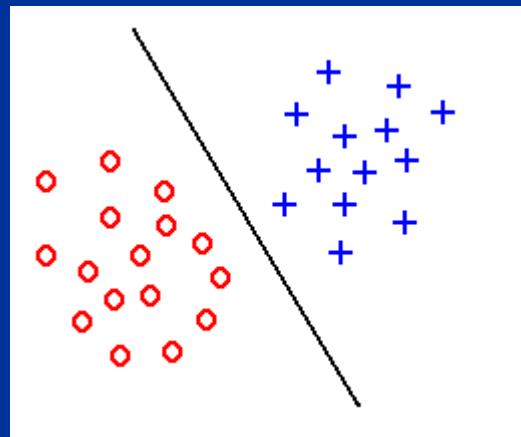
Trénovacie vzorky  $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_l, y_l)\}$



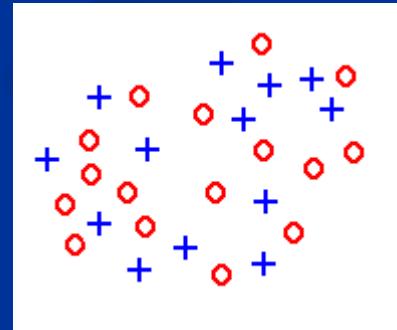
# Príznaky a obrazce

Kvalita vektora príznakov závisí od schopnosti rozdeliť vzorky do rôznych tried

- Vzorky z rovnakých tried by mali mať podobné hodnoty príznakov
- Vzorky z rôznych tried by mali mať odlišné hodnoty príznakov



“Good” features

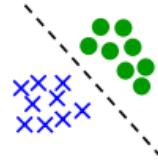


“Bad” features

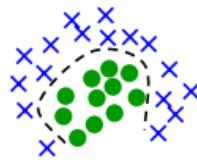
Separabilné triedy - také, pre ktoré existujú oddelujúce nadplochy.

Lineárne separabilné – také, pre ktoré existujú oddelujúce nadroviny.

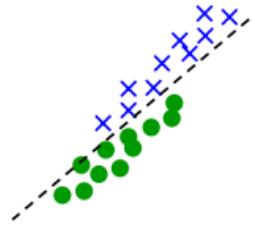
Väčšina reálnych problémov je reprezentovaná ako neseparabilné triedy, teda klasifikátor sa pomýli.



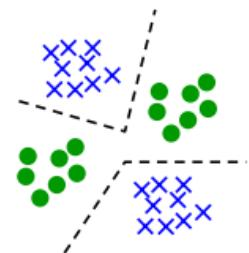
*Linear separability*



*Non-linear separability*

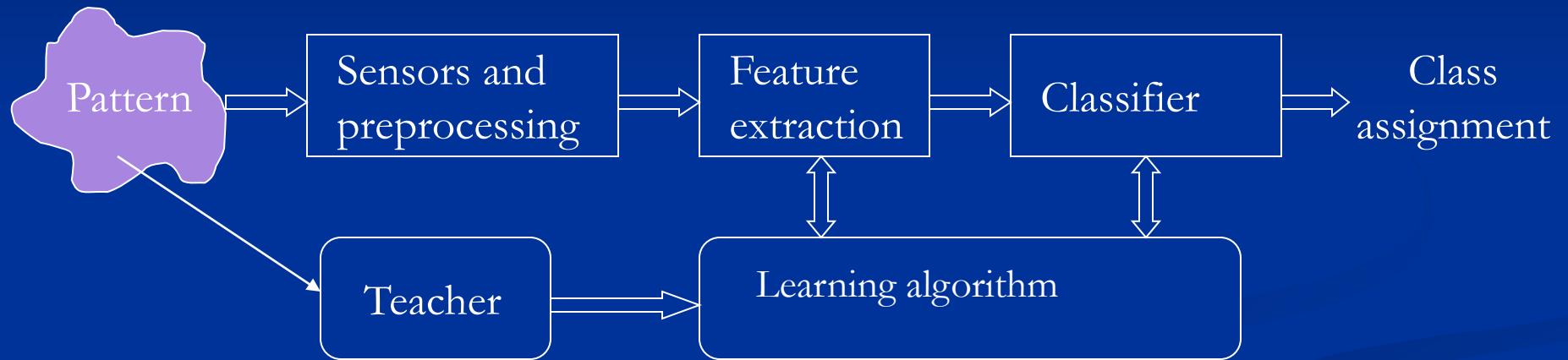


*Highly correlated features*



*Multi-modal*

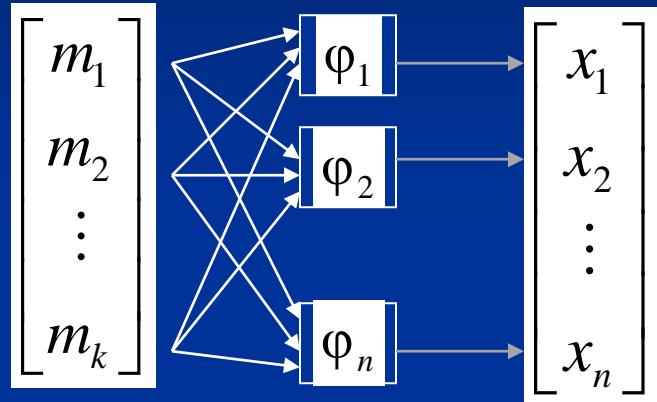
# Komponenty PR systému



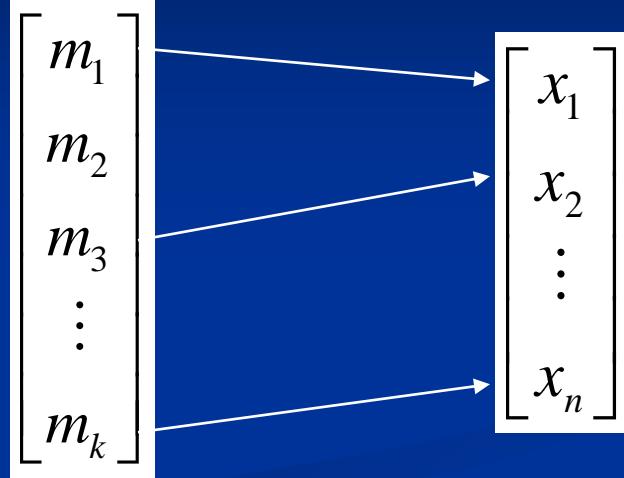
- **Sensors and preprocessing.**
- A **feature extraction** aims to create discriminative features good for classification.
- A **classifier**.
- A **teacher** provides information about hidden state -- supervised learning.
- A **learning algorithm** sets PR from training examples.

# Feature extraction methods

Redukcia príznakov



Výber príznakov



Problém môže byť vyjadrený ako optimalizácia parametrov  $\varphi(\theta)$

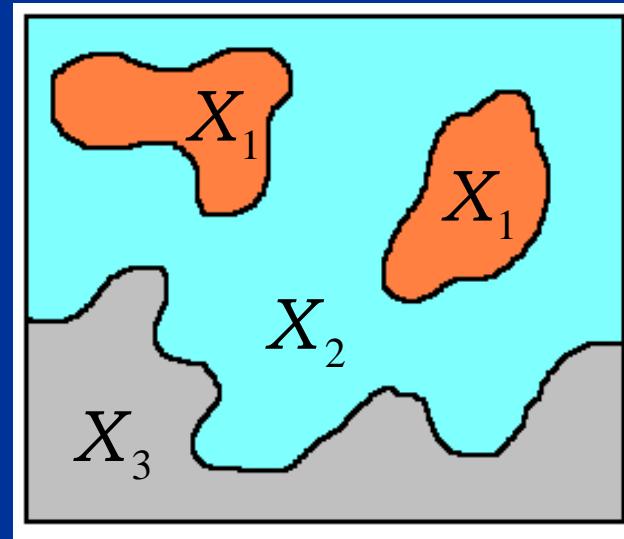
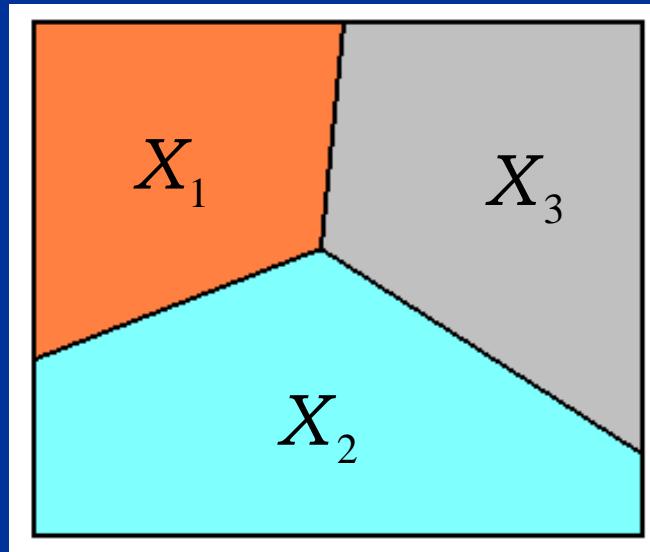
**Metódy s učiteľom:** objektívou funkciou je kritérium separability označených vzoriek, e.g., linear discriminant analysis (LDA).

**Metódy bez učiteľa:** hľadá sa reprezentácia nižšej dimenzie, ktorá zachová dôležité charakteristiky vstupných dát, e.g., principal component analysis (PCA).

# Klasifikátor

Klasifikátor rozdelí priestor príznakov  $X$  na regióny označené triedou také že

$$X = X_1 \cup X_2 \cup \dots \cup X_{|Y|} \quad \text{and} \quad X_1 \cap X_2 \cap \dots \cap X_{|Y|} = \{0\}$$



Klasifikácia pozostáva z rozhodnutia, do ktorého regiónu vektor príznakov  $x$  patrí.

Hranice medzi regiónmi rozhodovania sa nazývajú rozhodovacie hranice (**decision boundaries**)

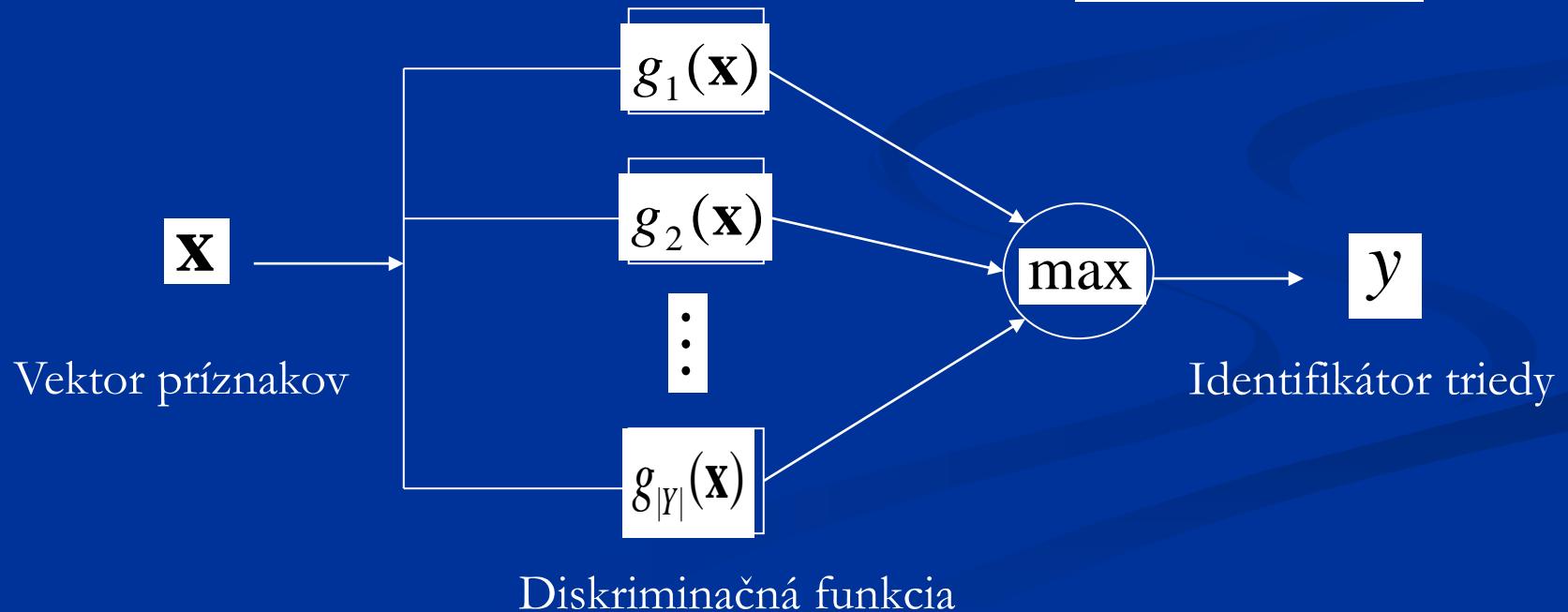
# Reprezentácia klasifikátora

Klasifikátor je typicky reprezentovaný množinou diskriminačných funkcií

$$g_i(\mathbf{x}) : X \rightarrow \mathfrak{R}, i = 1, \dots, |Y|$$

Klasifikátor priradí vektor príznakov  $\mathbf{x}$  do  $i$ -tej triedy ak

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \quad \forall j \neq i$$



$$\omega_r$$
$$\omega_s$$

Potom je nadplocha medzi triedami a určená rovnicou  $|g_r(\mathbf{x}) - g_s(\mathbf{x})| = 0$ .  
Z toho vyplýva rozhodovacie pravidlo

$$d(\mathbf{x}) = \omega_r \Leftrightarrow g_r(\mathbf{x}) = \max_{s=1,2,\dots,R} g_s(\mathbf{x})$$

Lineárne diskriminačné funkcie sú najjednoduchšie a široko používané

$$g_r(\mathbf{x}) = q_{r0} + q_{r1}x_1 + q_{r2}x_2 + \dots + q_{rn}x_n$$

Druhá možnosť je založená na princípe **minimálnej vzdialenosťi**.  
Predpokladajme, že je definovaných  $R$  etalónov  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_R$   
ktoré reprezentujú ideálnych reprezentantov  $R$  tried.  
Rozhodovacie pravidlo je potom

$$d(\mathbf{x}) = \omega_r \Leftrightarrow |\mathbf{v}_r - \mathbf{x}| = \min_{s=1,2,\dots,R} |\mathbf{v}_s - \mathbf{x}|$$

- Dá sa ukázať, že pravidlo najbližšieho suseda je špeciálnym prípadom lineárnych diskriminačných funkcií.
- Ak sú triedy lineárne separabilné, možno použiť metódu lineárnych diskriminačných funkcií alebo pravidlo najbližšieho suseda
- ak nie sú triedy lineárne separabilné, možno použiť  $\Phi$ -kernel alebo nelineárne diskriminačné funkcie.

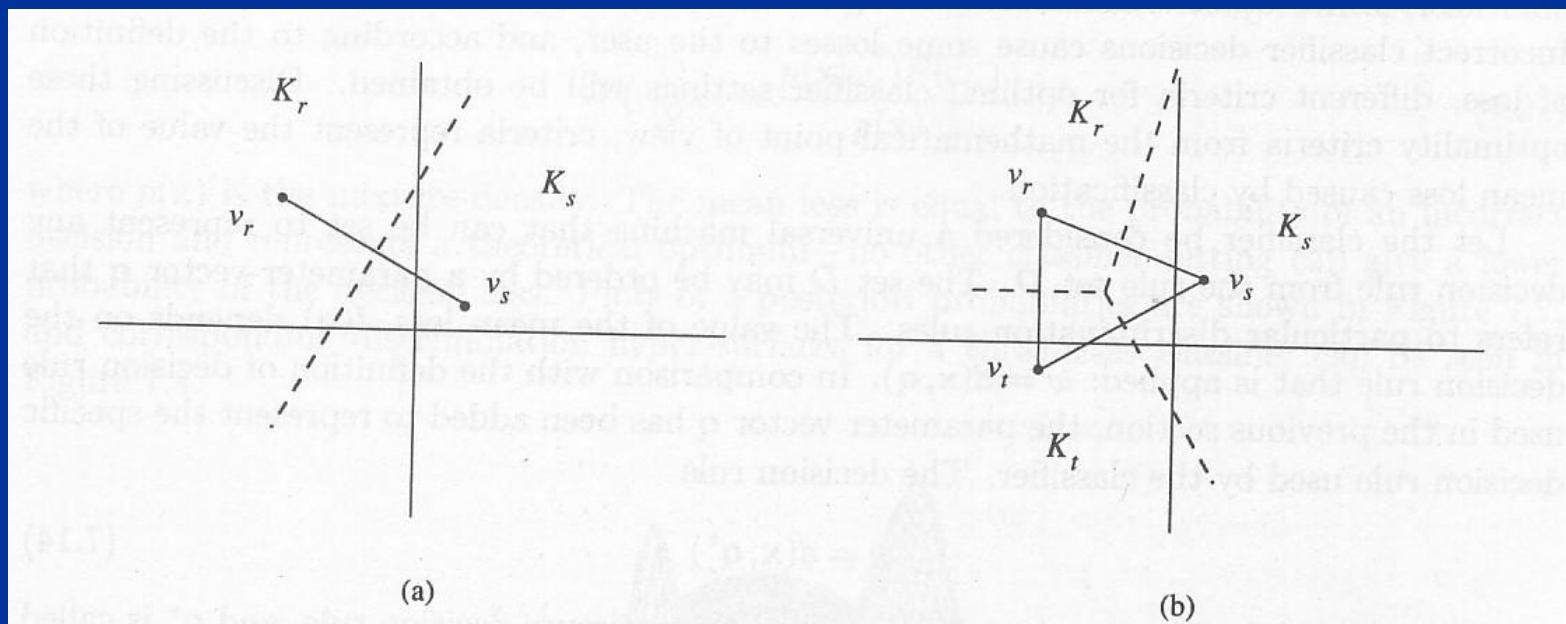


Figure 7.6: *Minimum distance discrimination functions:* (a) two-class problem; (b) three-class problem.

Definujme  $J(\mathbf{q})$  ako stratu spojenú s použitím konkrétneho rozhodovacieho pravidla  $\omega = d(\mathbf{x}, \mathbf{q})$  v prípade nesprávnej klasifikácie.

Potom pravidlo  $\omega = d(\mathbf{x}, \mathbf{q}^*)$  sa nazýva optimálne rozhodovacie pravidlo, ak dáva **minimálnu strednú stratu**, teda platí, že

$$J(\mathbf{q}^*) = \min_q J(\mathbf{q}), \forall d(\mathbf{x}, \mathbf{q})$$

Klasický príklad štatistického rozpoznávania, ktoré dáva minimálnu strednú stratu, je Bayesovo pravidlo, ktoré pracuje s 2 apriórnymi a 1 aposteriórnej pravdepodobnosťou.

Pretože pri neseparabilných triedach nevieme jednoznačne rozhodnúť o triede, pokladáme ju za náhodnú premennú s možnými hodnotami

$$\omega_1, \omega_2, \dots, \omega_R$$

s apriórnymi pravdepodobnosťami pre ktoré platí

$$P(\omega_1), P(\omega_2), \dots, P(\omega_R)$$

$$\sum_{r=1}^R P(\omega_r) = 1.$$

Väčšinou máme viac informácie ako iba apriórne pravdepodobnosti tried, nech máme aj podmienené pravdepodobnosti  $p(\mathbf{x}|\omega_r)$

ktoré vyjadrujú pravdepodobnosť objavenia sa príznakového vektora  $\mathbf{x}$  za podmienky, že je známa trieda  $\omega_R$

Potom podmienená pravdepodobnosť, že vektor príznakov  $\mathbf{x}$  patrí do triedy  $\omega_R$  je daná vztahom

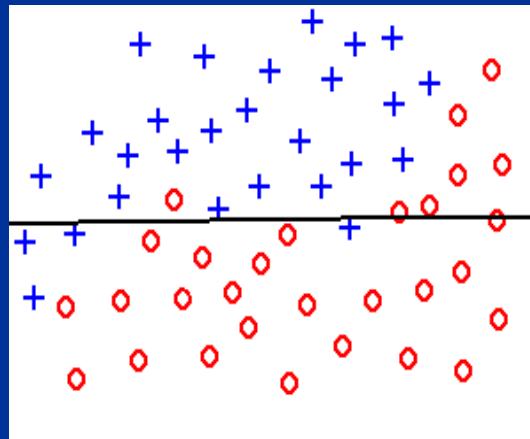
$$P(\omega_r|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_r).P(\omega_r)}{p(\mathbf{x})},$$

kde

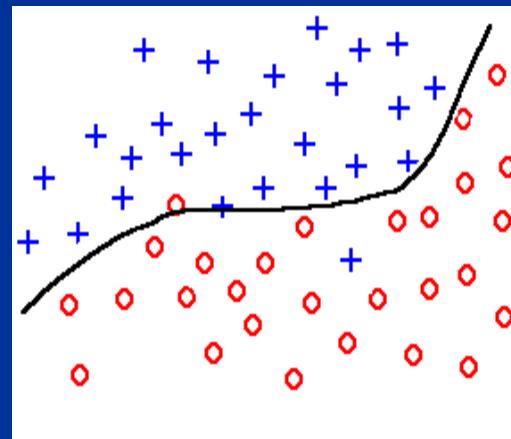
$$p(\mathbf{x}) = \sum_{r=1}^R p(\mathbf{x}|\omega_r).P(\omega_r)$$

je hustota pravdepodobnosti rozloženia príznakového vektora v príznakovom priestore bez ohľadu na triedu.

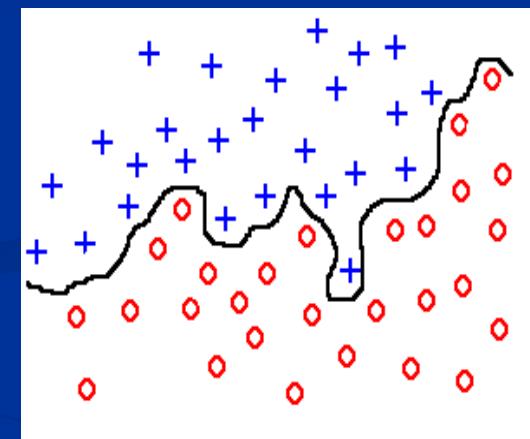
# Overfitting and underfitting



underfitting



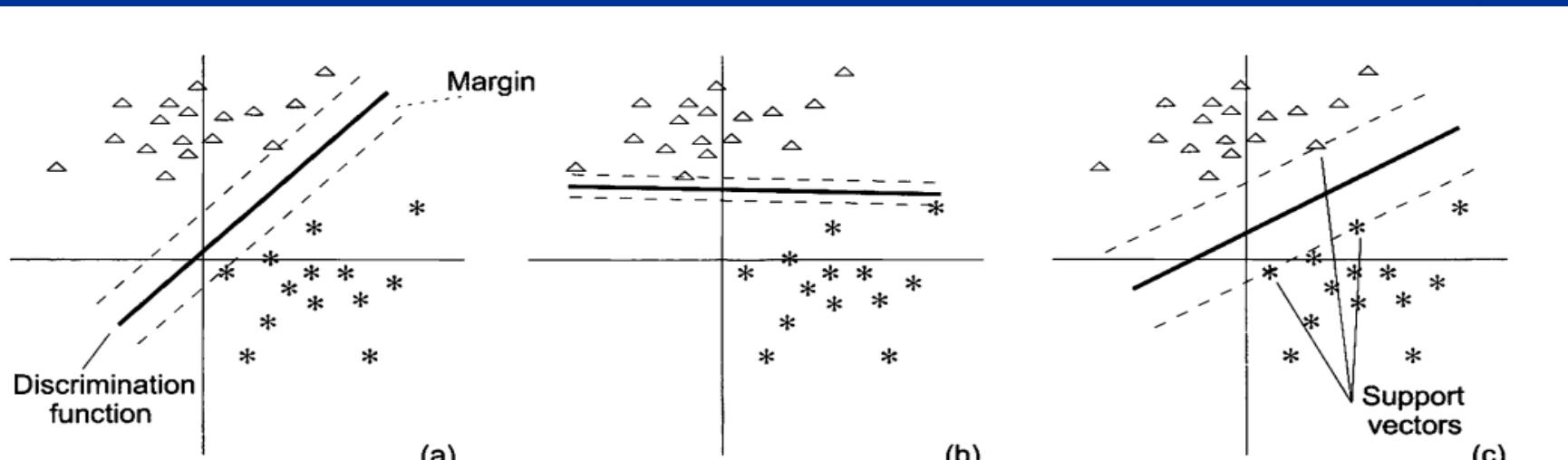
good fit



overfitting

# Support vector machines (SVM)

- Snaží sa pri rozdeľovaní 2 separovatelných tried maximalizovať vzdialenosť medzi nimi (**margin**)
- Vektory, ktoré sa nachádzajú najbližšie sa nazývaju podporné vektory (**support vectors**)



**Figure 9.9:** Basic two-class classification idea of support vector machines. (a,b) Two examples of non-optimal linear discrimination function. (c) Optimal linear discrimination function maximizes the margin between patterns of the two separable classes. The optimal hyperplane is a function of the support vectors.

# SVM

- 2 triedy označené  $\omega$ , kde  $\omega \in \{-1, 1\}$
- Oddelujúca hyperplocha  $\mathbf{w} \cdot \mathbf{x} + b = 0$
- Maximalizovať okraje (margin) 2 hyperplochy sú definované

$$\mathbf{w} \cdot \mathbf{x} + b = -1$$

$$\mathbf{w} \cdot \mathbf{x} + b = 1$$

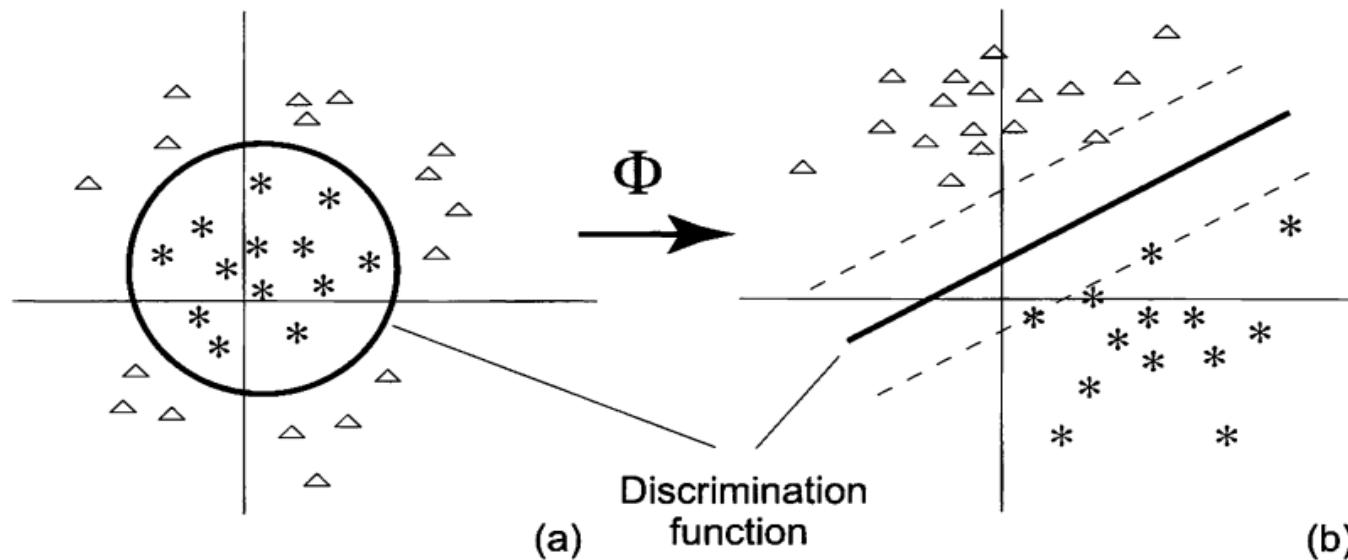
ktoré prechádzajú cez podporné vektory

- Aby sme zaručili, že ziadne trénovacie vektory nebudú medzi týmito hyperplochami musíme dodržať nerovnosť

$$\omega_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

# Kernel funkcie

Nelineárna transformácia priestoru, v ktorom použijeme lineárnu hyperplochu na oddelenie tried zodpovedá použitiu nelineárneho hyperpovrchu v pôvodnom priestore



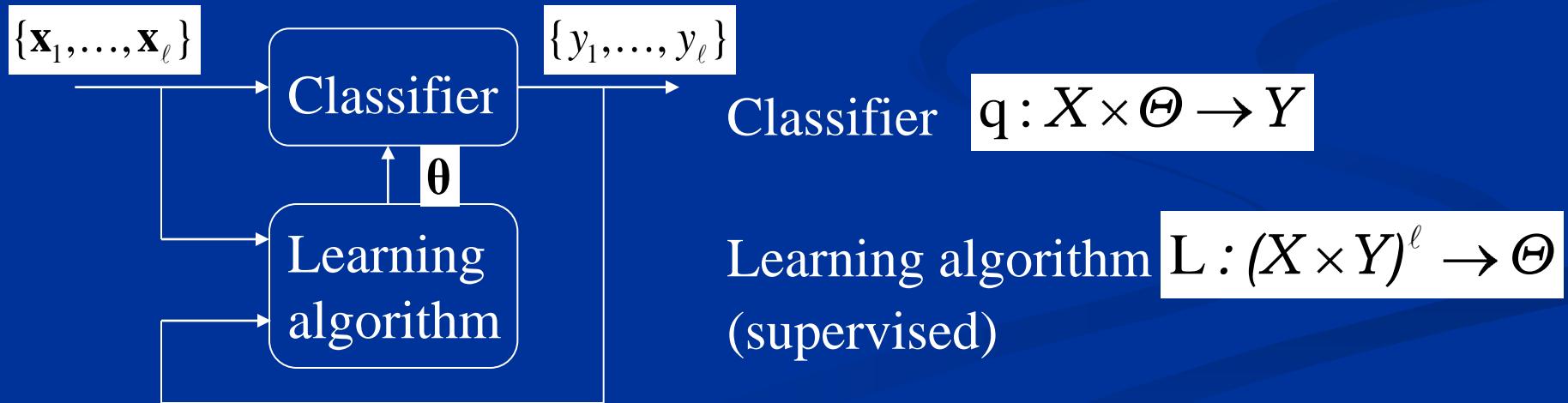
**Figure 9.10:** Non-linear separability approach to support vector machine classification. (a) Input space showing classes that are not linearly separable. (b) Non-linear  $\Phi$  mapping of the input space patterns to a new feature space in which the two classes are linearly separable.

# Učenie bez učiteľa

Vstup: trénovacie vektorové súbor  $\{\mathbf{x}_1, \dots, \mathbf{x}_\ell\}$  bez informácie o skrytom stave

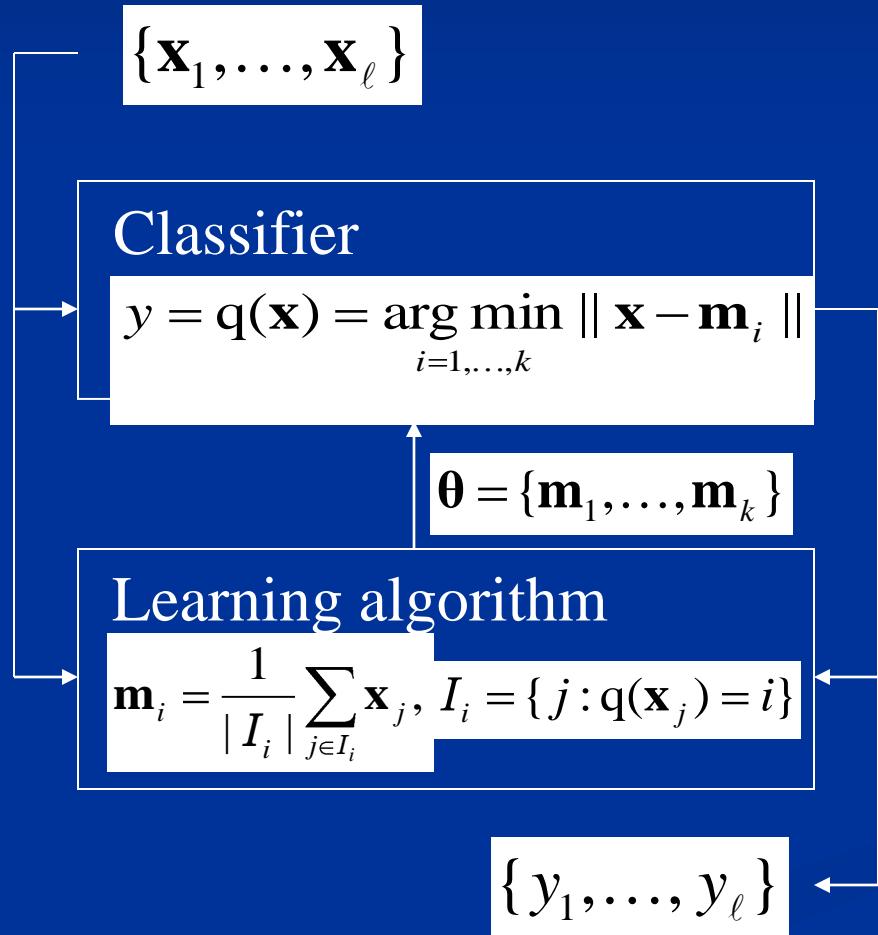
Zhlukovanie: cieľom je nájsť zhluky dát s podobnými vlastnosťami

Trieda algoritmov učiacich sa bez učiteľa:



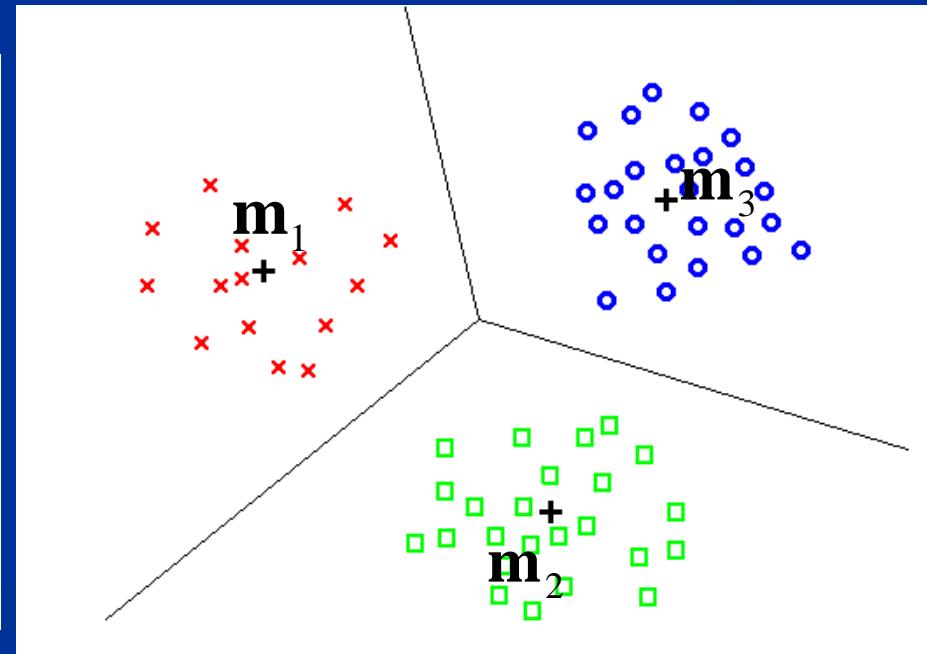
# Príklad učiaceho sa algoritmu bez učiteľa

k-Means clustering:



Ciel' je minimalizovať

$$\sum_{i=1}^{\ell} \|\mathbf{x}_i - \mathbf{m}_{q(\mathbf{x}_i)}\|^2$$



# Neurónové siete

- Viaceré **neurónové prístupy** sú založené na kombinácii elementárnych procesorov (neurónov), z ktorých každý má viacero vstupov a jeden výstup.
- Každému vstupu je priradená váha a výstup je sumou váhovaných výstupov.
- Rozpoznávanie obrazcov je jednou z mnohých oblastí použitia neurónových sietí.

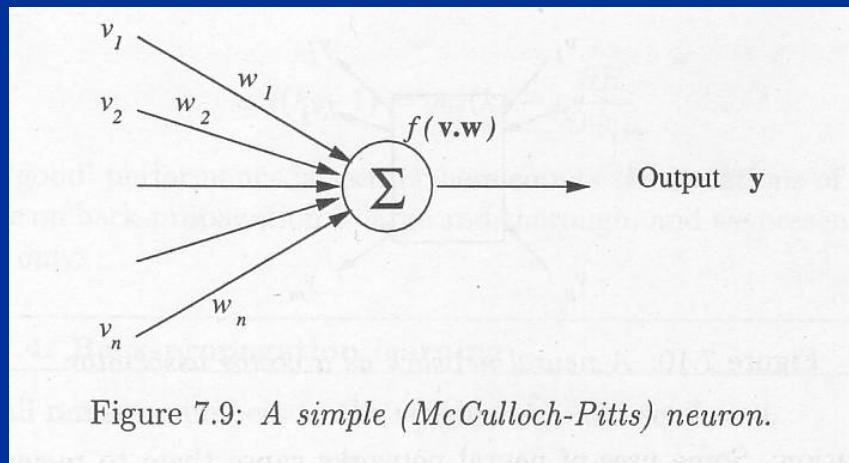
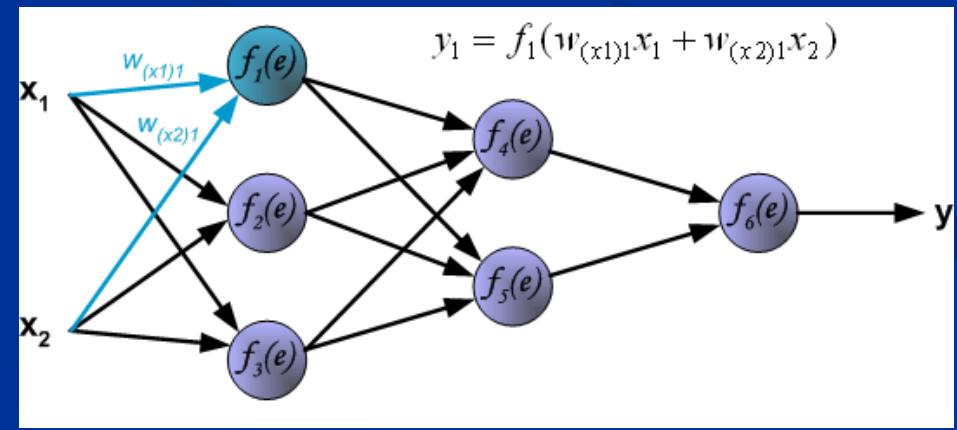
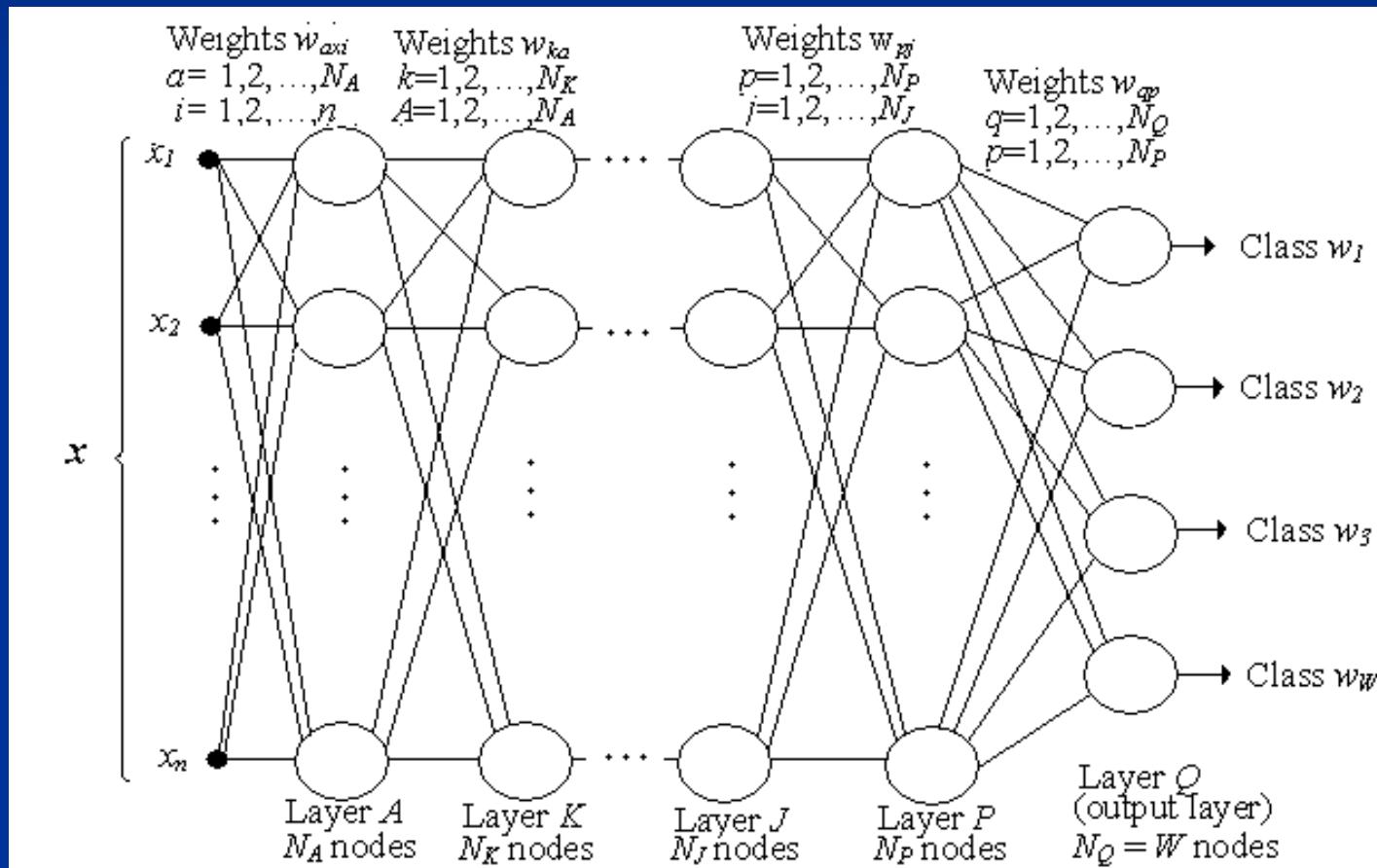


Figure 7.9: A simple (McCulloch-Pitts) neuron.



# Viacúrovňová dopredná neurónová siet'

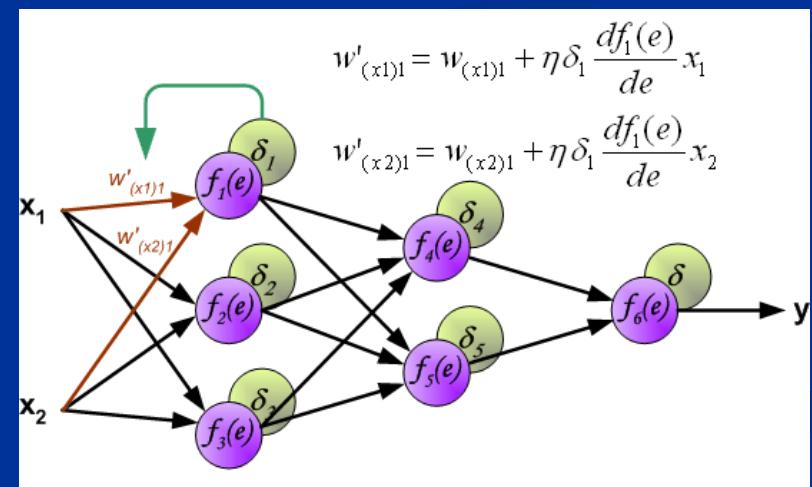
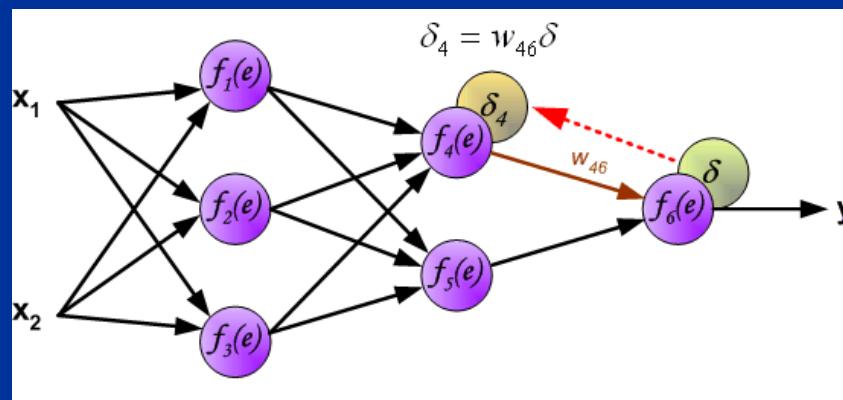
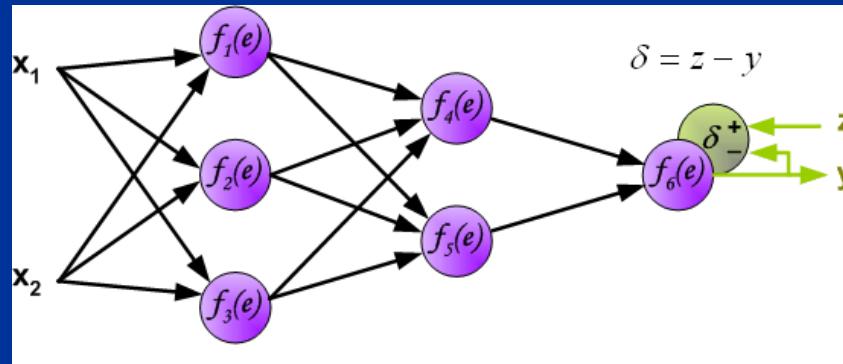
## ■ Basic structure



Structure of multilayer feedforward neural networks

# Viacúrovňová dopredná neurónová siet'

## ■ Training algorithm : back propagation



# Viacúrovňová dopredná neurónová siet'

## 1. Initialization

Assigning an arbitrary set of weights throughout the network (not equally).

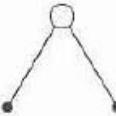
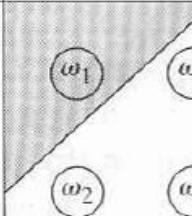
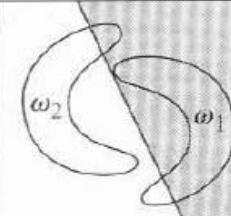
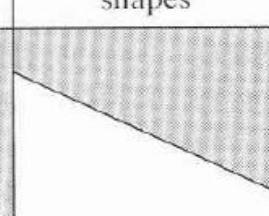
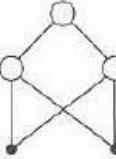
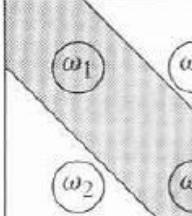
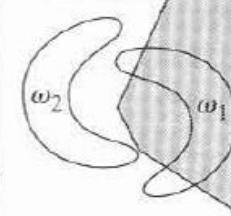
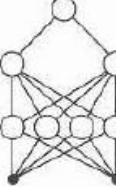
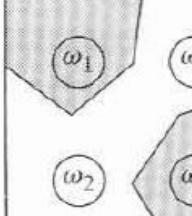
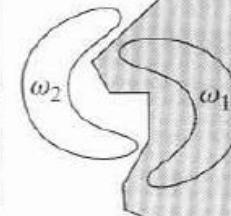
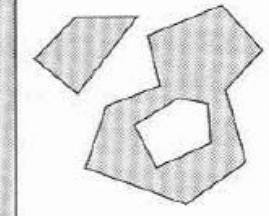
## 2. Iterative step

- a. Computing  $\gamma$  for each node by using training vector, then generating the error terms for output  $\delta$ ,  
 $\zeta$  is the desired response.
- b. Backward passing - appropriate error signal is passed to each node and the corresponding weight changes are made.

# Neural Networks

## ■ Decision surface complexity

Table2 : Decision surface complexity of multilayer feedforward neural networks[1]

Network structure	Type of decision region	Solution to exclusive-OR problem	Classes with meshed regions	Most general decision surface shapes
Single layer 	Single hyperplane			
Two layers 	Open or closed convex regions			
Three layers 	Arbitrary (complexity limited by the number of nodes)			

# Syntactic Recognition : String Case

- Input to the automata are unknown sentences generated by the corresponding grammars respectively.
  - ◆ The grammar  $G = (N, \Sigma, P, S)$   
 $N$  is a finite set of variables called *nonterminals*,  
 $\Sigma$  is a finite set of constants called *terminals*,  
 $P$  is a set of rewriting rules called *productions*, and  
 $S$  in  $N$  is called the *starting symbol*.

# Syntactic Recognition : String Case

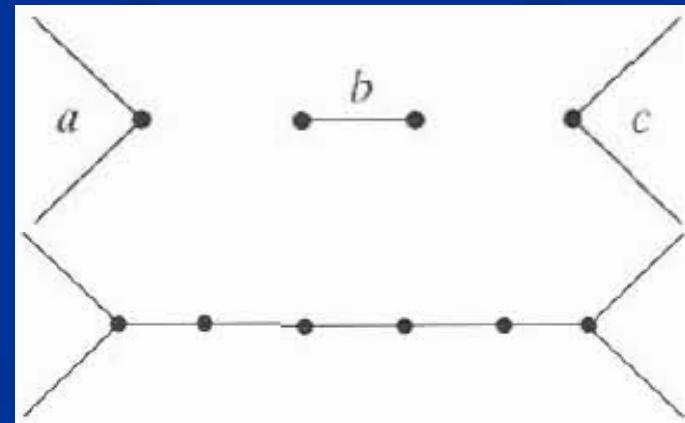
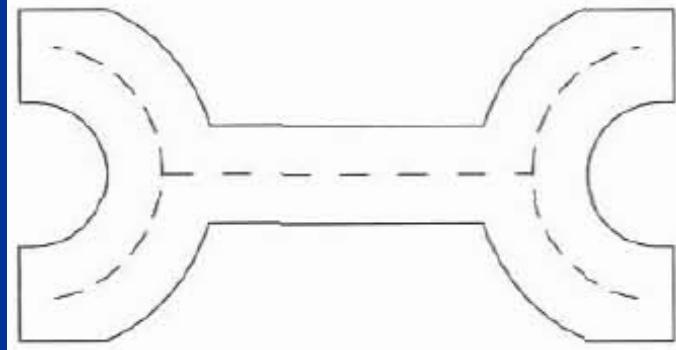
## ■ An example

$$N = \{A, B, S\}, \Sigma = \{a, b, c\}$$

$$P = \{S \rightarrow aA, A \rightarrow bA, A \rightarrow bB, B \rightarrow c\}$$

$$S \rightarrow aA \rightarrow abA \rightarrow abbA \rightarrow \dots \rightarrow abbbbbc$$

$$L(G) = \{ab^n c \mid n \geq 1\}$$

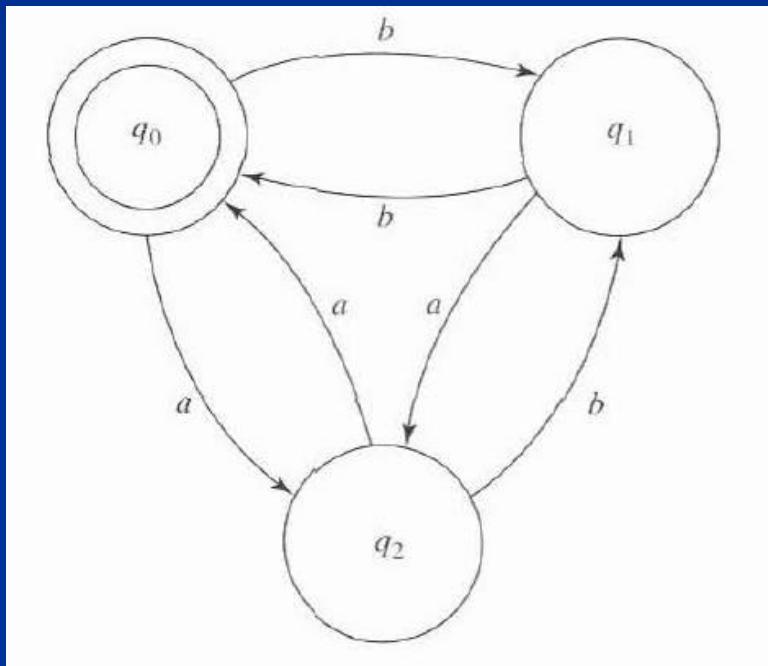


# Syntactic Recognition : String Case

- ◆ The finite automata  $A_f = (Q, \Sigma, \delta, q_0, F)$   
 $Q$  is a finite, nonempty set of *states*,  
 $\Sigma$  is a finite input *alphabet*,  
 $\delta$  is a *mapping* from  $Q \times \Sigma$  into the collection of all  
subsets of  $Q$ ,  
 $q_0$  is the *starting state*, and  
 $F$  is a set of *final states*.

# Syntactic Recognition : String Case

## ■ A simple automaton



$$A_f = (Q, \Sigma, \delta, q_0, F)$$

$$Q = \{q_0, q_1, q_2\}$$

$$\Sigma = \{a, b\}$$

$$F = q_0$$

$$\delta(q_0, a) = \{q_2\}$$

$$\delta(q_0, b) = \{q_1\}$$

$$\delta(q_1, a) = \{q_2\}$$

$$\delta(q_1, b) = \{q_0\}$$

$$\delta(q_2, a) = \{q_0\}$$

$$\delta(q_2, b) = \{q_1\}$$

Invalid input string : bababbb

Valid input string : aaabbba

# Syntactic Recognition : String Case

- Conversion between regular grammar and corresponding automaton states.

$$G = (N, \Sigma, P, S)$$



$$Af = (Q, \Sigma, \delta, q_0, F)$$

$$X_0 \equiv S$$

$$Q = \{q_0, q_1, \dots, q_n, q_{n+1}\}$$

$$N = \{X_0 \sim X_n\}$$

The mappings in  $\delta$  are obtained by using the following two rules, for  $a$  in  $\Sigma$ , and each  $i$  and  $j$ , with  $0 \leq i \leq n$ ,  $0 \leq j \leq n$ ,

1. If  $X_i \rightarrow aX_j$  is in  $P$ , then  $\delta(q_i, a)$  contains  $q_j$ .
2. If  $X_i \rightarrow a$  is in  $P$ , then  $\delta(q_i, a)$  contains  $q_{n+1}$ .