

# Metody klasyfikacji danych - część 2

## *Inteligentne Usługi Informacyjne*

Jerzy Dembski

# Plan wykładu

- AdaBoost
- Klasyfikacja metodą wektorów wspierających (SVM)
- Klasyczne sztuczne sieci neuronowe
- Głębokie uczenie (*Deep Learning*)

# Algorytm AdaBoost

**for** każdy przykład do uczenia  $\{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2) \dots (\mathbf{x}_K, c_K)\}$

przypisz wagom wartości początkowe  $w_i = \frac{1}{K_c}$ , gdzie  $K_c$  jest liczbą przykładów, które należą do tej samej klasy  $c$  co  $i$ -ty przykład

**end for**

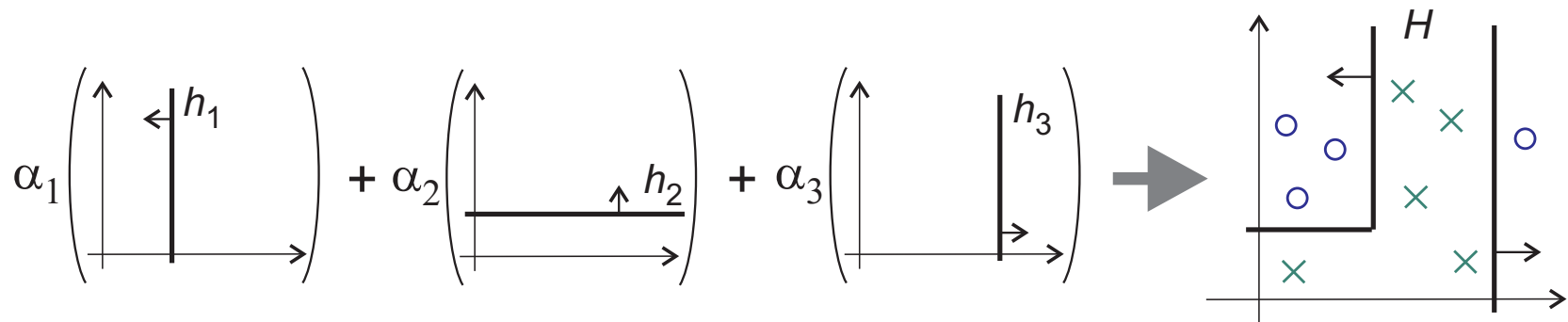
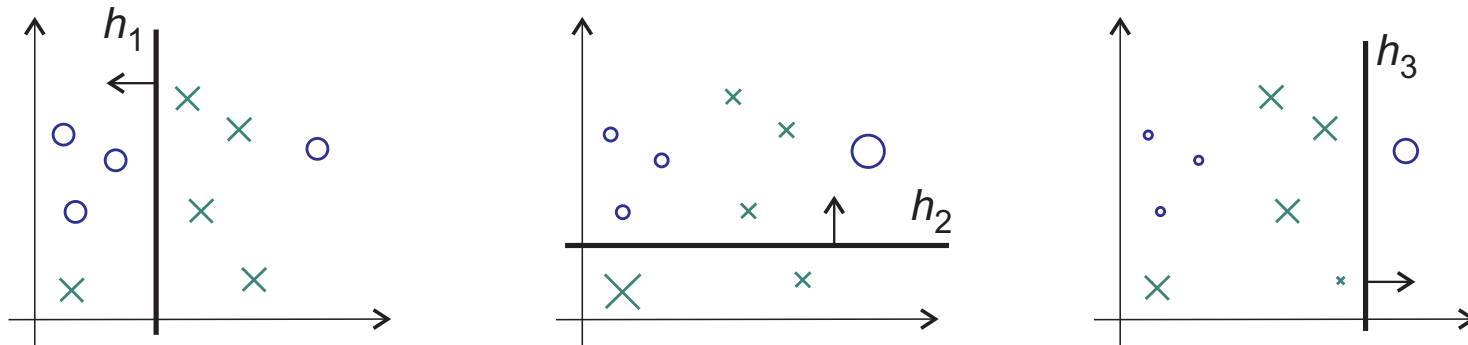
**while** błąd fałszywie pozytywnych  $f > f_{\max}$  **or** proporcja prawdziwie pozytywnych  $d < d_{\min}$

1. normalizuj wagi:  $w_i \leftarrow \frac{w_i}{\sum_{j=1}^K w_j}$
2. wybierz słaby klasyfikator  $h_t(\mathbf{x})$ , który minimalizuje ważony błąd klasyfikacji:  $\epsilon_t = \min_j \sum_{i=1}^K w_i |h_j(\mathbf{x}_i) - c_i|$
3. zmniejsz wagi poprawnie klasyfikowanych przykładów:  $w_i \leftarrow w_i \frac{\epsilon_t}{1 - \epsilon_t}$
4. znajdź wartość progu  $\Theta$  dla silnego klasyfikatora dającą jak najmniejszą wartość błędu fałszywie pozytywnych, podczas gdy  $d \geq d_{\min}$

**end while**

# Algorytm AdaBoost - prosty przykład

Słaby klasyfikator progowy jest reprezentowany funkcją 4-argumentową:  $h(\mathbf{x}, f, \theta, p)$ ,  
gdzie  $\mathbf{x}$  - wektor cech,  $f$  - cecha,  $\theta$  - wartość progu,  $p$  - polaryzacja



# Algorytm AdaBoost - wzór na silny klasyfikator

Silny klasyfikator zawierający  $T$  słabych klasyfikatorów:

$$H(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \geq \Theta \\ 0 & \text{w przeciwnym razie,} \end{cases}$$

gdzie  $\alpha_t = \log(1 - \epsilon_t) - \log \epsilon_t$  jest wagą  $t$ -tego słabego klasyfikatora,

$h_t(\mathbf{x})$  – wartość na wyjściu  $t$ -tego słabego klasyfikatora,

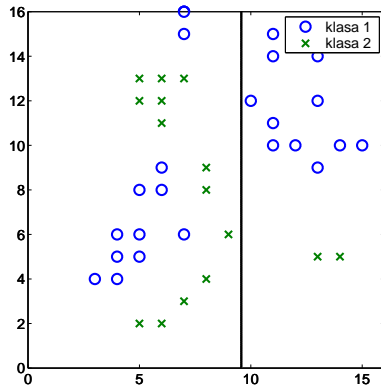
$\mathbf{x}$  – obraz wejściowy,

$\Theta$  – wartość progu.

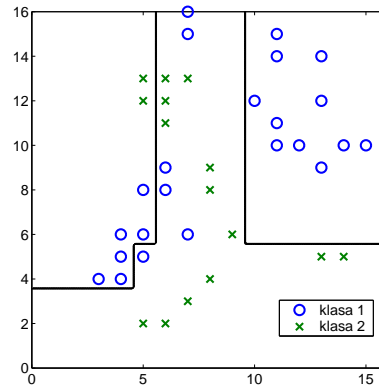
# Algorytm AdaBoost - granice decyzji

Granice decyzji dla zadania klasyfikacji punktów i przy różnej liczbie słabych klasyfikatorów  $T$ : a) 1, b) 5, c) 10, d) 50, e) 100, f) 1000.

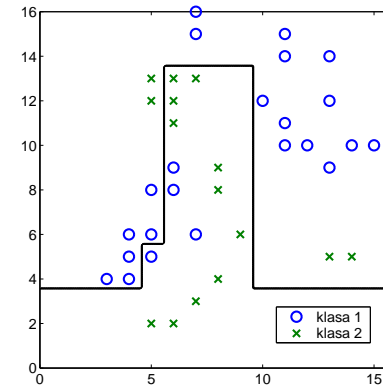
a)



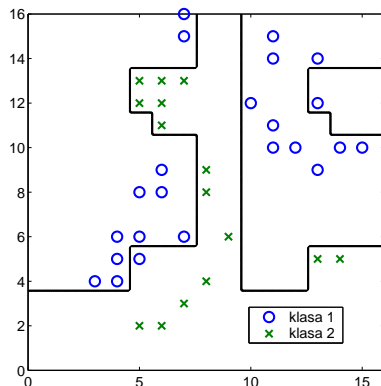
b)



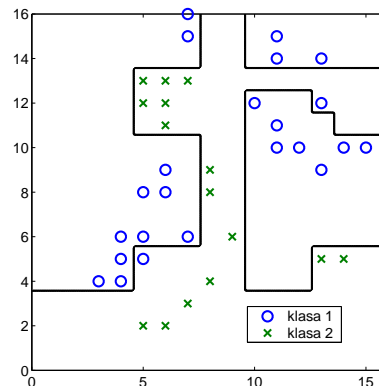
c)



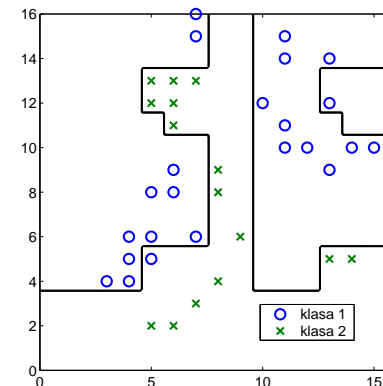
d)



e)



f)



# Algorytm AdaBoost w detekcji twarzy

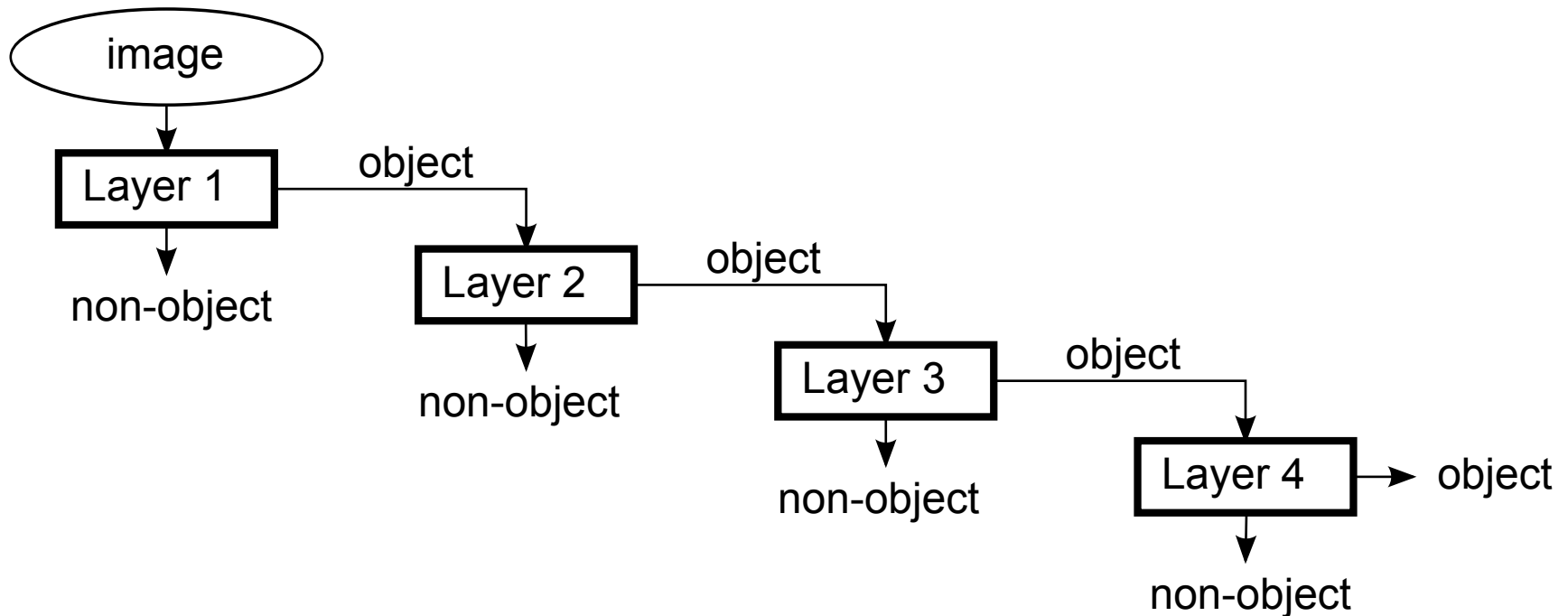
Wymagania:

- Znormalizowane położenie twarzy w przykładach uczących
- Znormalizowane średnia i wariancja jasności pikseli
- Kaskadowa wersja klasyfikatora AdaBoost (przewaga obrazów negatywnych)

Cechy obrazów:

- Różnice średnich jasności pikseli w obszarach prostokątowych (HAAR Features)
- Lokalne cechy binarne (*CENSUS*, *Local Binary Features* (LBF))
- Histogram kierunków gradientów (*Histogram of Oriented Gradients* (HOG))
- Zgrupowane binarne cechy prostokątowe, Lokalne zgrupowane cechy binarne (LAB)

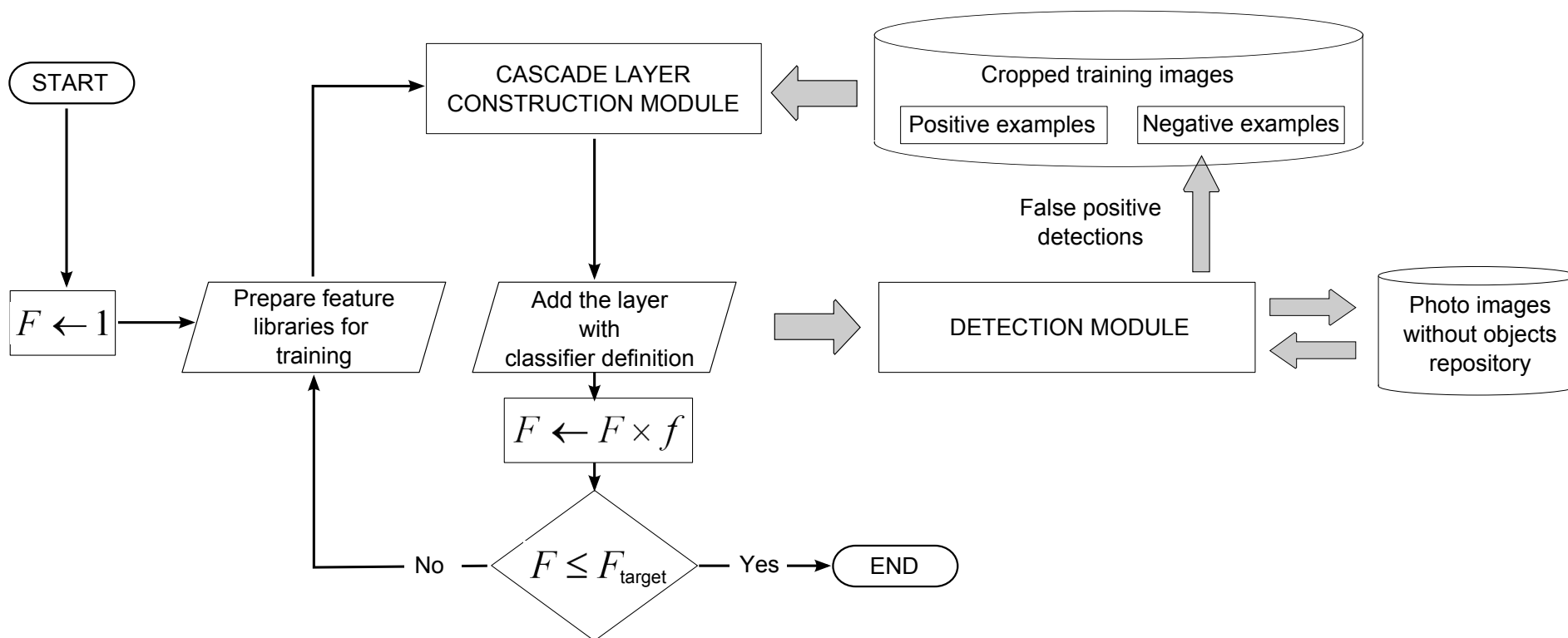
# Kaskada silnych klasyfikatorów



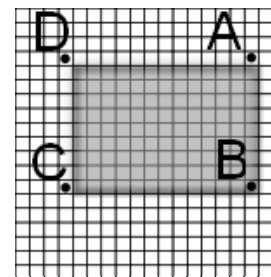
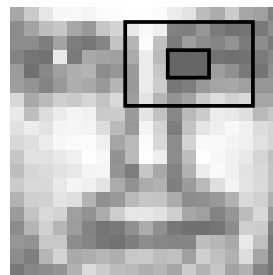
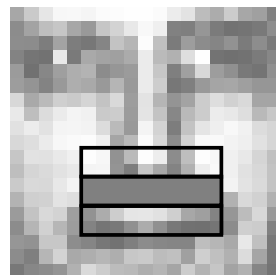
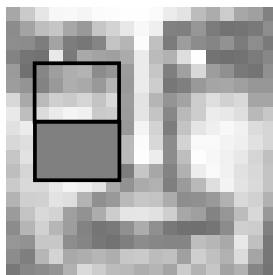
Zaletą kaskady - odrzucanie dużej liczby obrazów negatywnych małym kosztem obliczeniowym w przypadku dużej przewagi obrazów negatywnych podczas skanowania zdjęcia lub obrazu z kamery



# Schemat uczenia kaskady silnych klasyfikatorów w zadaniu detekcji obiektów



# Typy cech prostokątowych (Haar features) w detekcji twarzy



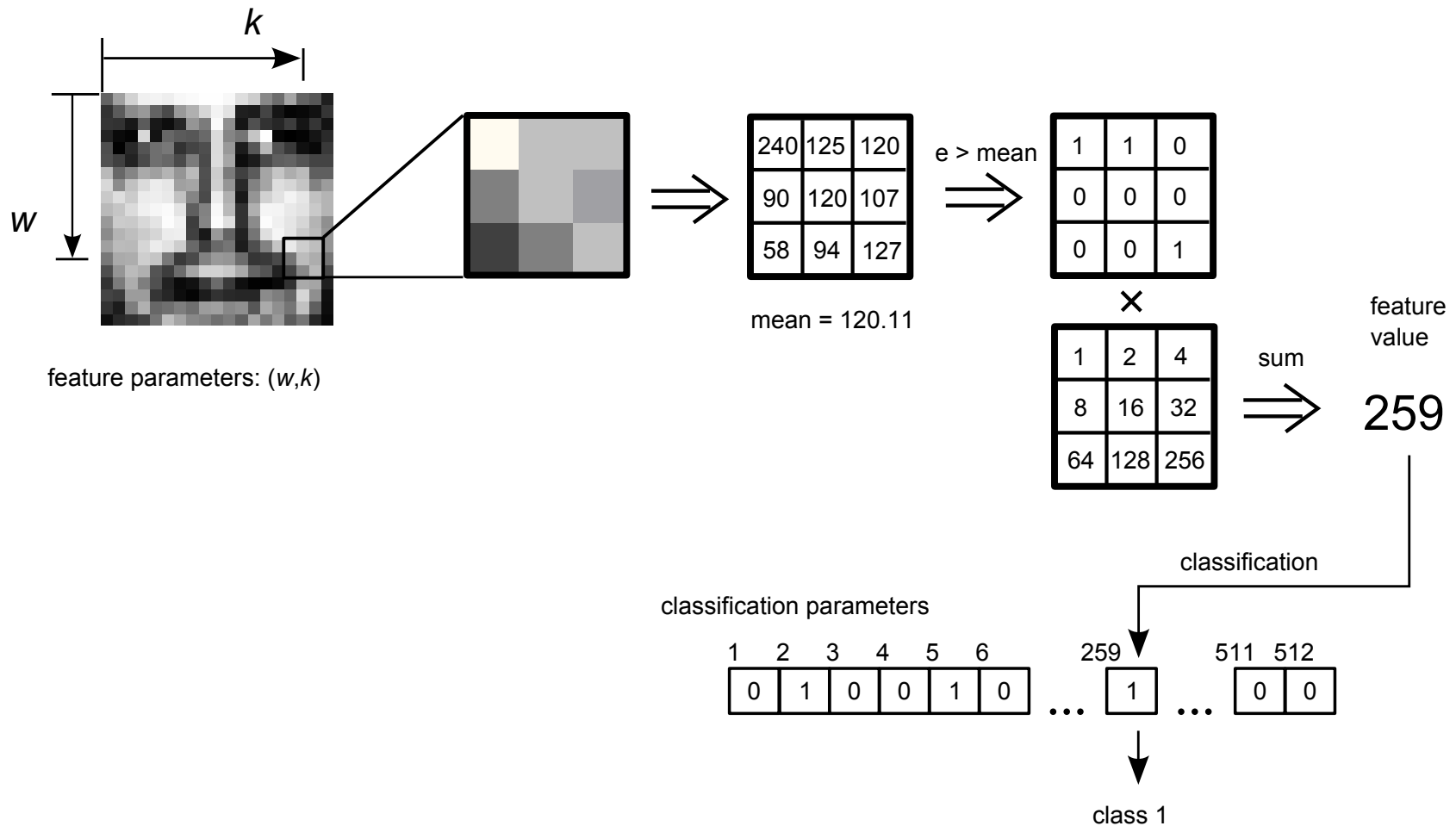
Użycie obrazu całkowego: Suma jasności pikseli w prostokącie ABCD = B+D-A-C

Dodatkowe założenia:

- każde okno prostokątne cechy może być skalowane i przesuwane w obszarze obrazu,
- średnie i wariancje jasności pikseli obrazów w zbiorze do uczenia i do testu (detekcji) muszą być takie same.

# Zmodyfikowany Census

Przykład obliczania wartości cech zmodyfikowanego Census:



# Zmodyfikowany Census

Uczenie słabego klasyfikatora:

$$\mathbf{x}_{\text{best}} = \operatorname{argmin}_{\mathbf{x}} (\epsilon(\mathbf{x}))$$

$$\epsilon(\mathbf{x}) = \sum_{\gamma} \min\{g^0(\mathbf{x}, \gamma), g^1(\mathbf{x}, \gamma)\},$$

gdzie

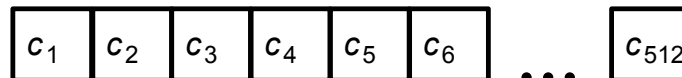
$\mathbf{x}$  - wektor parametrów cech:  $(w, k)$ ,

$\gamma$  - numer wzorca (od 1 do 512),

$g^c(\mathbf{x}, \gamma)$  - ważona suma przykładów uczących klasy  $c \in \{0, 1\}$

Numer klasy dla wybranej cechy  $\mathbf{x}_{\text{best}}$  i wzorca  $\gamma$ :

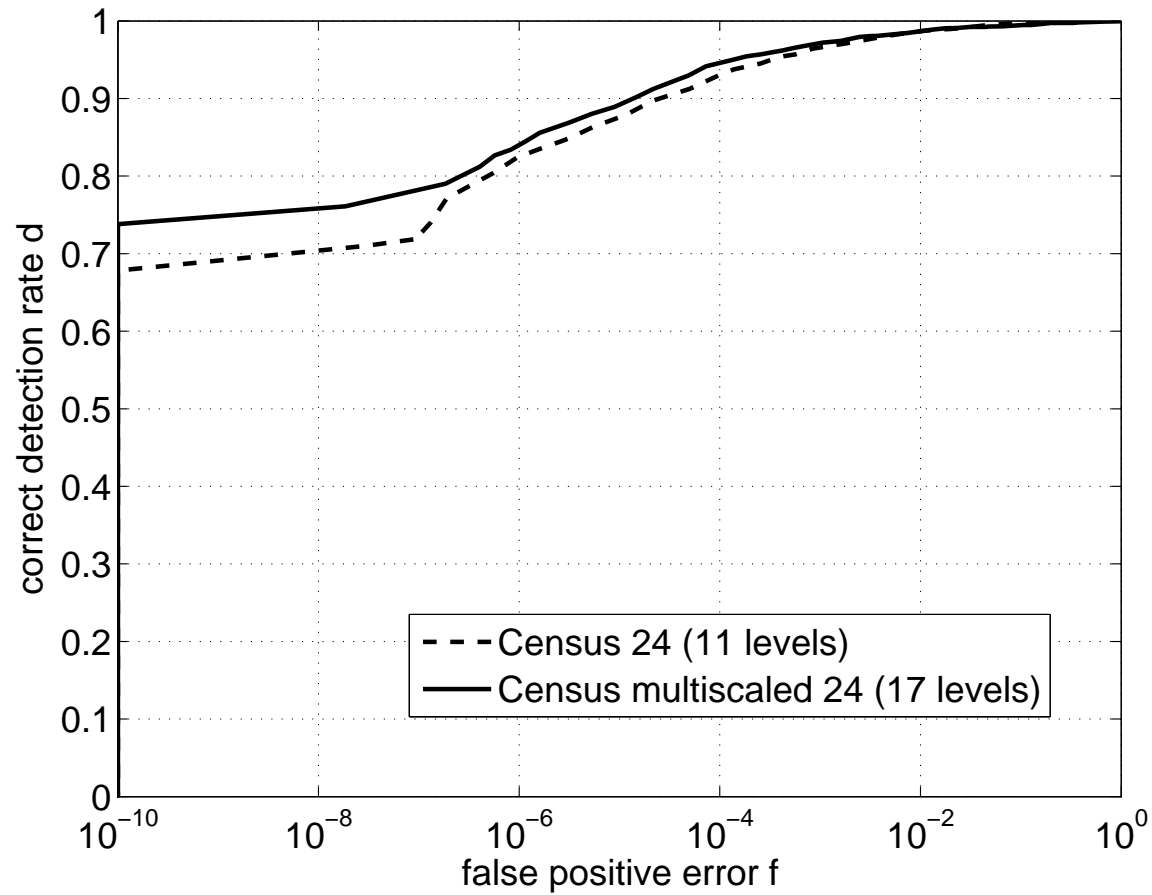
$$c_{\gamma} = \begin{cases} 0 & \text{jeśli } g^0(\mathbf{x}_{\text{best}}, \gamma) > g^1(\mathbf{x}_{\text{best}}, \gamma) \\ 1 & \text{w przeciwnym wypadku.} \end{cases}$$



# AdaBoost: zestawienie typów cech i słabych klasyfikatorów

typ cechy	typ klasyfikatora	parametry cechy $p_f$	parametry klasyfikatora $p_c$
prostokątowa (Haar)	progowy	współrzędne lewego górnego i prawego dolnego narożnika prostokąta, orientacja prostokąta (tylko cechy krawędziowe i liniowe)	próg decyzji, polaryzacja
Z.Census	tabelaryczny	współrzędne centralnego piksela siatki 3x3 ( $w, k$ )	wektor binarny klas: numery klas dla wszystkich możliwych wzorców

# Przykładowa krzywa ROC



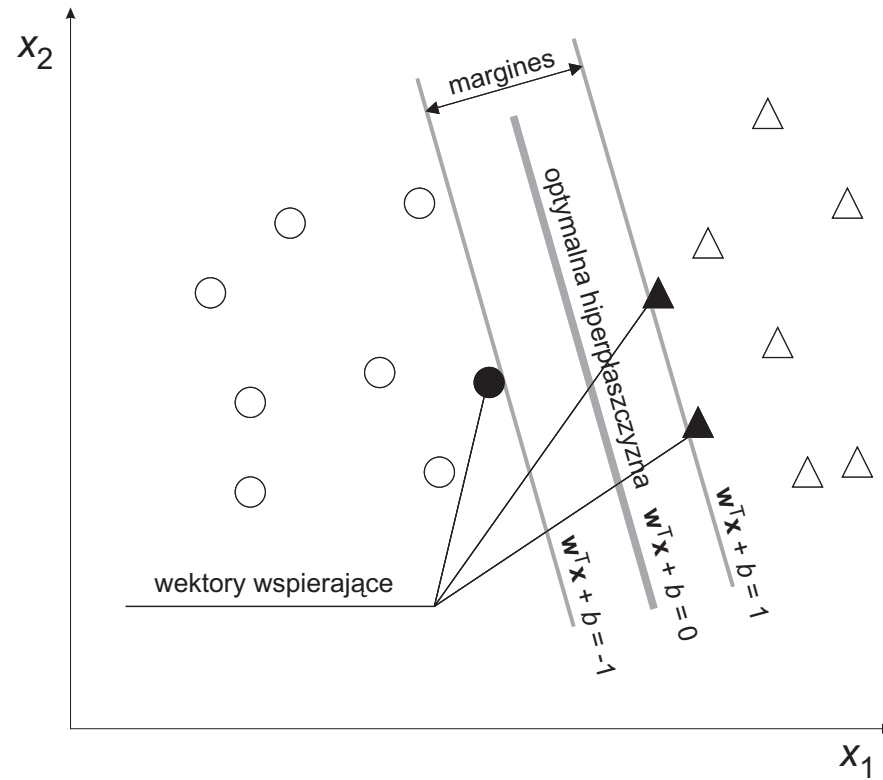
# SVM: problem klasyfikacji

Zbiór danych - przykładów (każdy przykład  $i$  zawiera wektor cech obrazu  $\mathbf{x}_i$  oraz klasę  $d_i$ , do której należy):

$$P = \{(\mathbf{x}_i, d_i)\}_{i=1}^K, \quad \mathbf{x}_i \in X \subset \mathbb{R}^N, \quad d_i \in \{-1, 1\}.$$

Cel uczenia: znaleźć hiperpłaszczyznę (np. prostą o równaniu  $w_1 * x_1 + w_2 * x_2 + b = 0$ ) dzielącą skupiska punktów należących do klas  $-1$  i  $1$  z jak najszerszym marginesem. Granice tego marginesu są wyznaczane przez wybrane wektory  $\mathbf{x}_i$  - tzw. wektory wspierające.

# SVM: optymalna hiperpłaszczyzna



$$\text{szerokość marginesu} = \frac{2}{\|w\|}$$



# SVM: szukanie optymalnej hiperpłaszczyzny

maksymalizacja marginesu:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2,$$

przy ograniczeniach:

$$d_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad i = 1, \dots, K.$$

# SVM: szukanie optymalnej hiperpłaszczyzny

Metoda mnożników Lagrange'a:

1. definicja lagrangiana:

$$L = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^K \alpha_i (d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1),$$

2. tzw. dualny problem optymalizacyjny (szukanie punktu siodłowego):

- minimalizacja  $L$  ze względu na  $\mathbf{w}$  oraz  $b$
- maksymalizacja  $L$  względem  $\alpha_i$ ,  $i = 1, \dots, K$ .

# SVM: minimalizacja $L$ ze względu na $w$ oraz $b$

przyporównanie pochodnych do zera:

$$\frac{\delta L}{\delta b} = 0, \quad \frac{\delta L}{\delta \mathbf{w}} = 0,$$

co daje

$$\sum_{i=1}^K \alpha_i d_i = 0,$$

$$\mathbf{w} = \sum_{i=1}^K \alpha_i d_i \mathbf{x}_i.$$

# SVM: maksymalizacja $L$ względem $\alpha$

problem programowania kwadratowego (QP):

$$\max_{\alpha} \left( \sum_{i=1}^K \alpha_i - \frac{1}{2} \sum_{i,j=1}^K \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j \right),$$

przy ograniczeniach:

$$\sum_{i=1}^K \alpha_i d_i = 0,$$

$$\alpha_i \geq 0, \quad i = 1, \dots, K,$$

$$\alpha_i (d_i (\mathbf{w}^T \mathbf{x}_i + b) - 1) = 0, \quad i = 1, \dots, K.$$

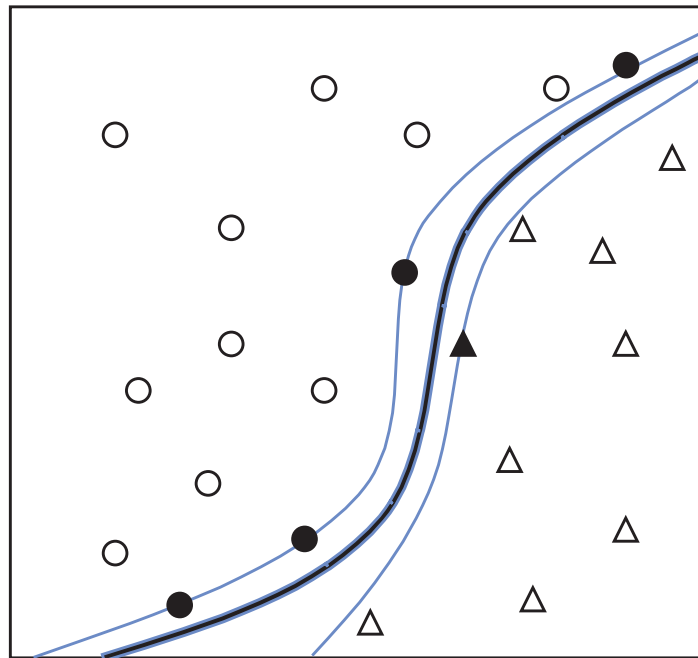
# SVM: klasyfikacja

numer klasy dla obrazu  $\mathbf{x}_j$ :

$$f(\mathbf{x}_j) = \operatorname{sgn}\left(\sum_{i=1}^S \alpha_i d_i \mathbf{x}_j^T \mathbf{x}_i + b\right),$$

gdzie  $S$  - liczba wektorów wspierających (zwykle dużo mniejsza niż liczba wszystkich przykładów  $K$ )

# SVM: przykład zadania nieseparowalnego



# SVM: zadania liniowo nieseparowalne

Metody rozwiązywania zadań nieseparowalnych:

- Ograniczenie wartości mnożników Lagrange'a  
 $\alpha_i \leq C, \quad i = 1, \dots, K$

# SVM: zadania liniowo nieseparowalne

Metody rozwiązywania zadań nieseparowalnych:

- Ograniczenie wartości mnożników Lagrange'a  
 $\alpha_i \leq C, \quad i = 1, \dots, K$
- Zastosowanie współczynników rozluźniających  $\xi_i \geq 0$  :

$$d_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, K$$



# SVM: zadania liniowo nieseparowalne

Metody rozwiązywania zadań nieseparowalnych:

- Ograniczenie wartości mnożników Lagrange'a  
 $\alpha_i \leq C, \quad i = 1, \dots, K$
- Zastosowanie współczynników rozluźniających  $\xi_i \geq 0$  :

$$d_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, K$$

- Przejście do nowej przestrzeni cech:  $\Phi : \mathbb{R}^N \rightarrow \mathbb{R}^M$

# SVM: funkcja jądra

dokonując podstawienia:

$$\mathbf{x}_i^T \mathbf{x}_j \longrightarrow \text{kernel}(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j),$$

otrzymujemy nową postać problemu QP:

$$\max_{\alpha} \left( \sum_{i=1}^K \alpha_i - \frac{1}{2} \sum_{i,j=1}^K \alpha_i \alpha_j d_i d_j \text{kernel}(\mathbf{x}_i, \mathbf{x}_j) \right),$$

oraz funkcji klasyfikacji SVM:

$$f(\mathbf{x}_j) = \text{sgn} \left( \sum_{i=1}^S \alpha_i y_i \text{kernel}(\mathbf{x}_i, \mathbf{x}_j) + b \right)$$

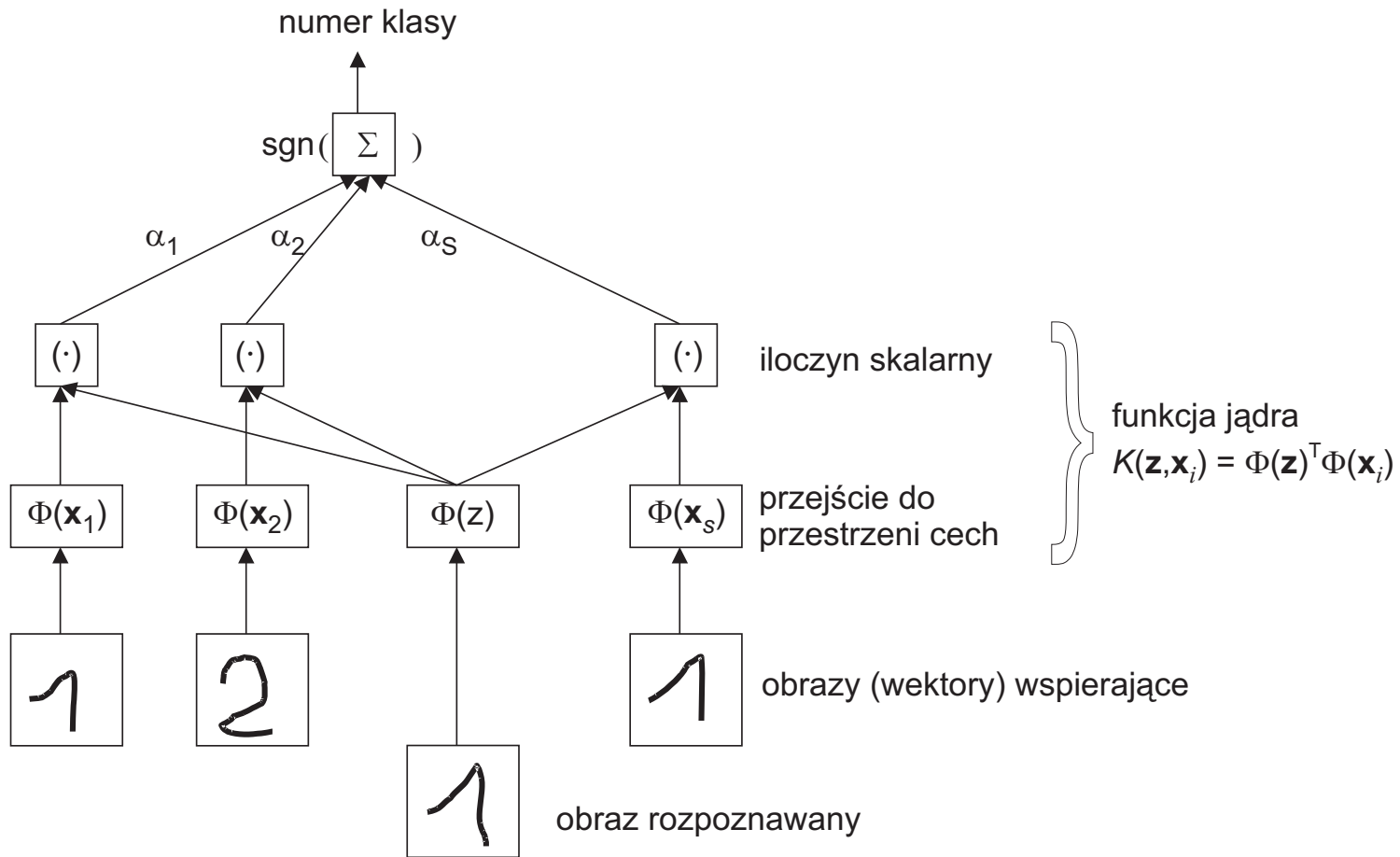
# SVM: typy funkcji jądra

$$\text{kernel}(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \mathbf{x}_i^T \mathbf{x}_j & \text{– liniowa,} \\ (\mathbf{x}_i^T \mathbf{x}_j + \theta)^p & \text{– wielomianowa,} \\ \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right) & \text{– radialna.} \end{cases}$$

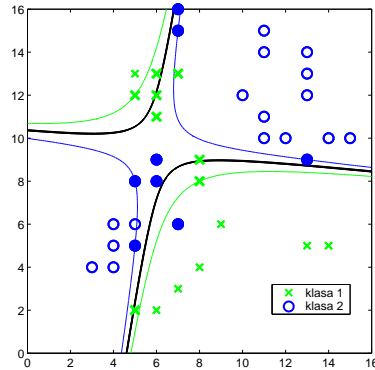
# SVM: funkcja kwadratowa

$$\begin{aligned}\text{kernel}(\mathbf{u}, \mathbf{v}) &= (\mathbf{u}^T \mathbf{v})^2 = ([u_1, u_2][v_1, v_2]^T)^2 = \\ &= u_1^2 v_1^2 + 2u_1 v_1 u_2 v_2 + u_2^2 v_2^2 = \\ &= [u_1^2, \sqrt{2}u_1 u_2, u_2^2][v_1^2, \sqrt{2}v_1 v_2, v_2^2]^T = \\ &= \Phi(\mathbf{u})^T \Phi(\mathbf{v}).\end{aligned}$$

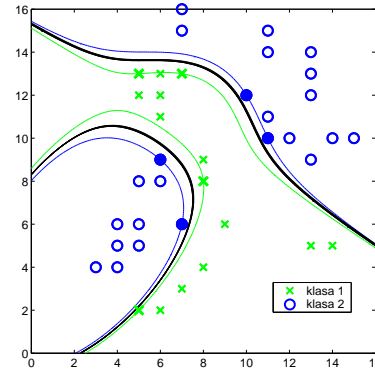
# SVM: schemat klasyfikacji



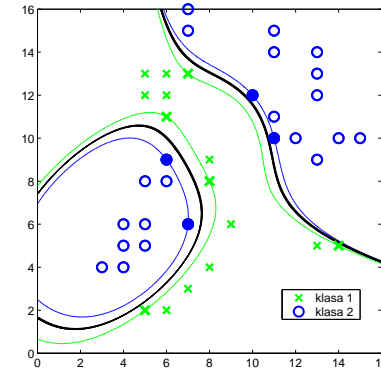
a)



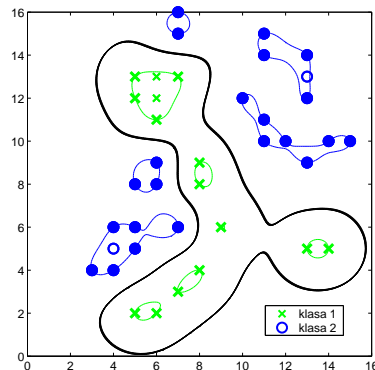
b)



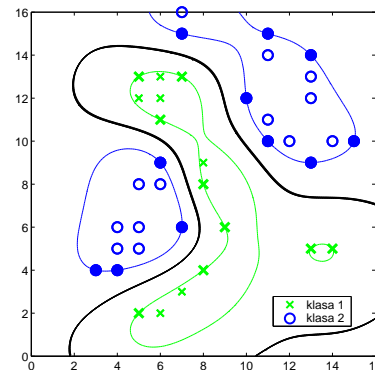
c)



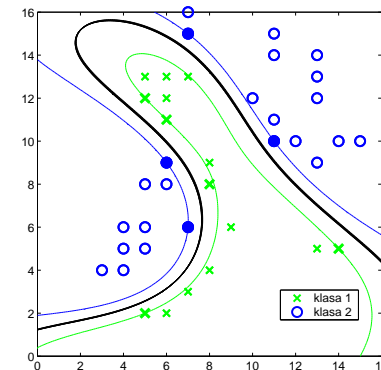
d)



e)

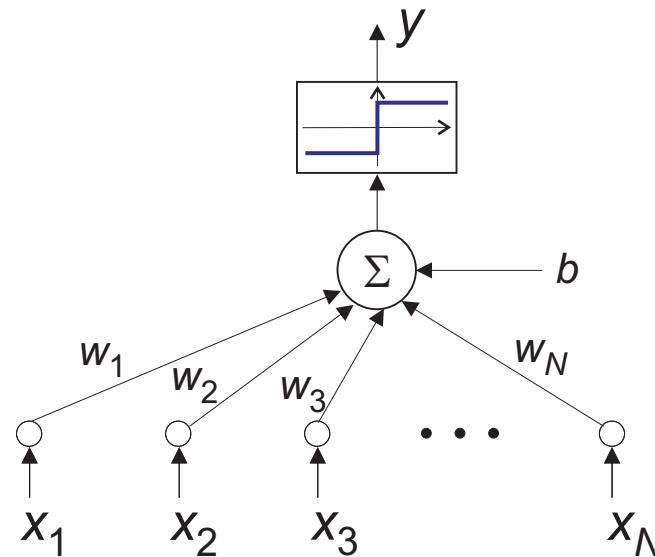


f)



Granice decyzji dla przykładowego zadania klasyfikacji wektorów obserwacji o  $N = 2$  punkty-1 dla klasyfikatora SVM z różnymi funkcjami jądra: a) wielomian 2-stopnia, b) wielomian 3-stopnia, c) wielomian 4-stopnia, d) funkcja radialna  $\sigma = 1.0$ , e) funkcja radialna  $\sigma = 2.0$ , f) funkcja radialna  $\sigma = 5.0$ . Wektory wspierające pogrubiono, granicę decyzji zaznaczono czarną, pogrubioną krzywą.

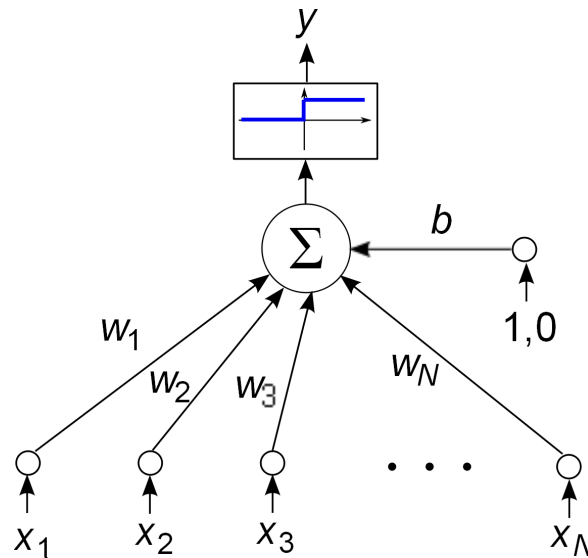
# Sztuczne sieci neuronowe: Model neuronu z progową bipolarną funkcją aktywacji



Funkcja, którą taka sieć reprezentuje wyraża się wzorem:

$$f(\mathbf{x}) = y = \text{sgn}\left(\sum_{i=1}^N w_i x_i + b\right) = \text{sgn}(\mathbf{w}^T \mathbf{x} + b),$$

# Model neuronu z progową unipolarną funkcją aktywacji



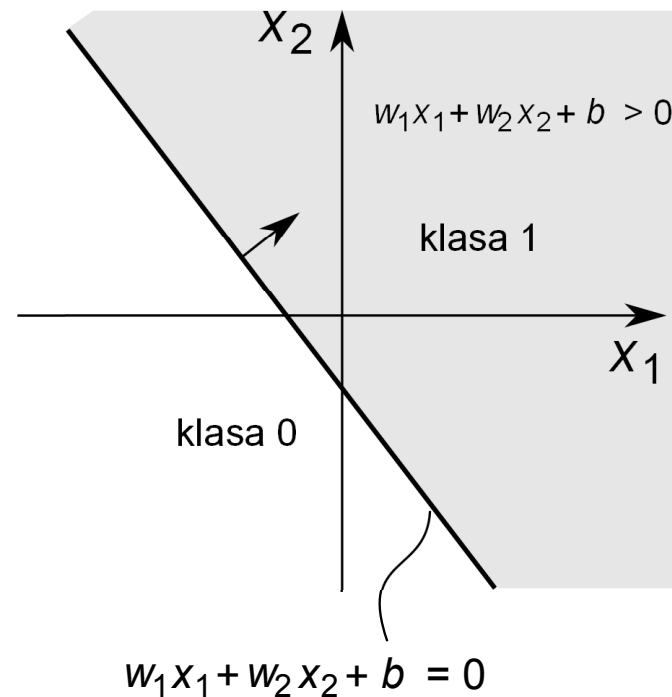
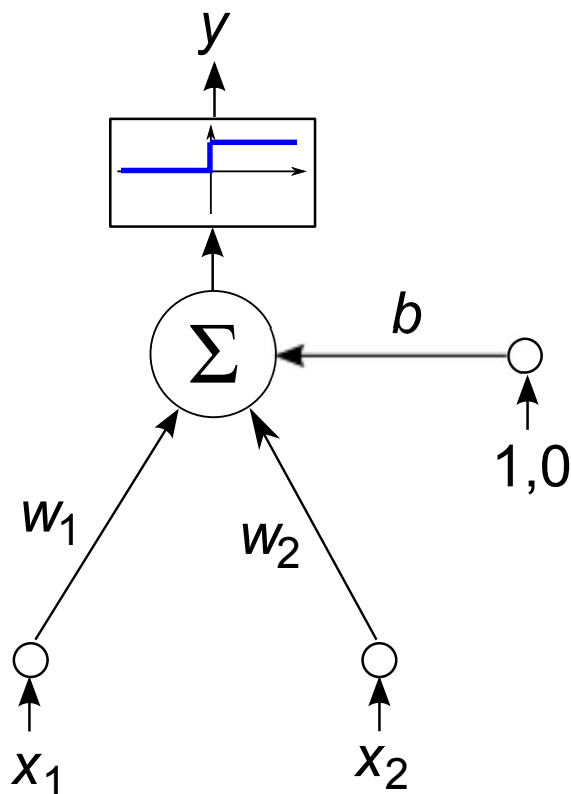
Funkcja, którą taka sieć reprezentuje wyraża się wzorem:

$$f(\mathbf{x}) = y = \text{hardlim}\left(\sum_{i=1}^N w_i x_i + b\right) = \text{hardlim}(\mathbf{w}^T \mathbf{x} + b),$$

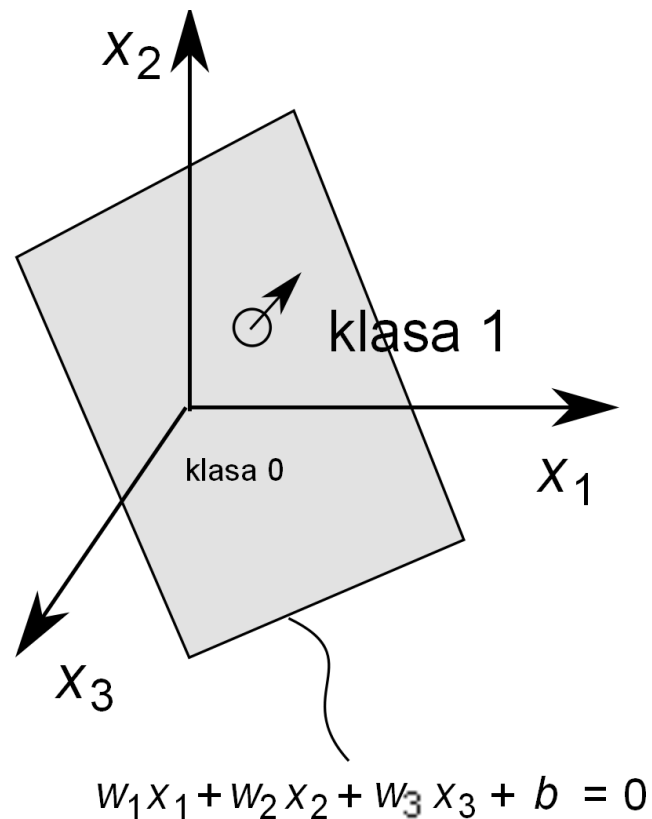
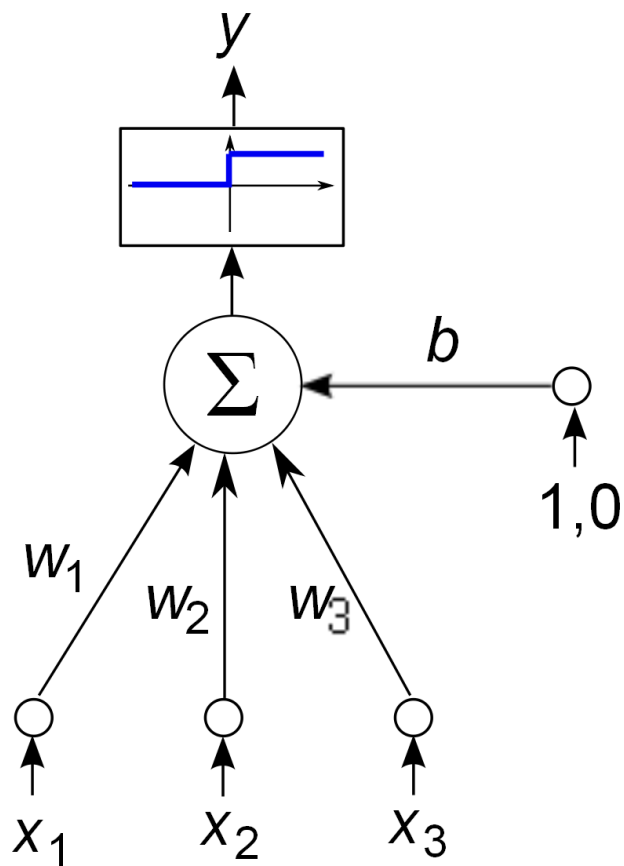


# Interpretacja geometryczna funkcji reprezentowanej przez neuron

Prosta wyznaczająca półpłaszczyznę klasy 1.



# Płaszczyzna separująca przestrzeń na dwie półprzestrzenie



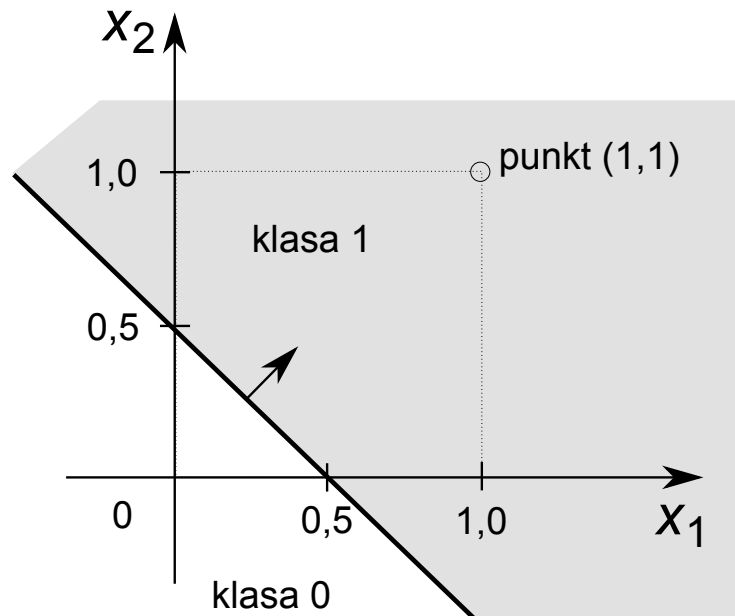
# Możliwość reprezentowania funkcji logicznych

alternatywa:  $x_1 \vee x_2$

$$w_1 = 1$$

$$w_2 = 1$$

$$b = -0.5$$

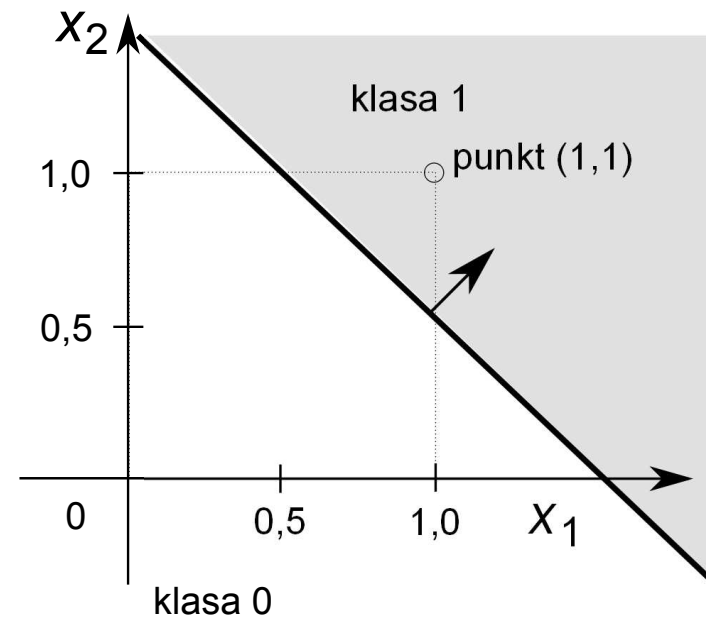


koniunkcja:  $x_1 \wedge x_2$

$$w_1 = 1$$

$$w_2 = 1$$

$$b = -1.5$$



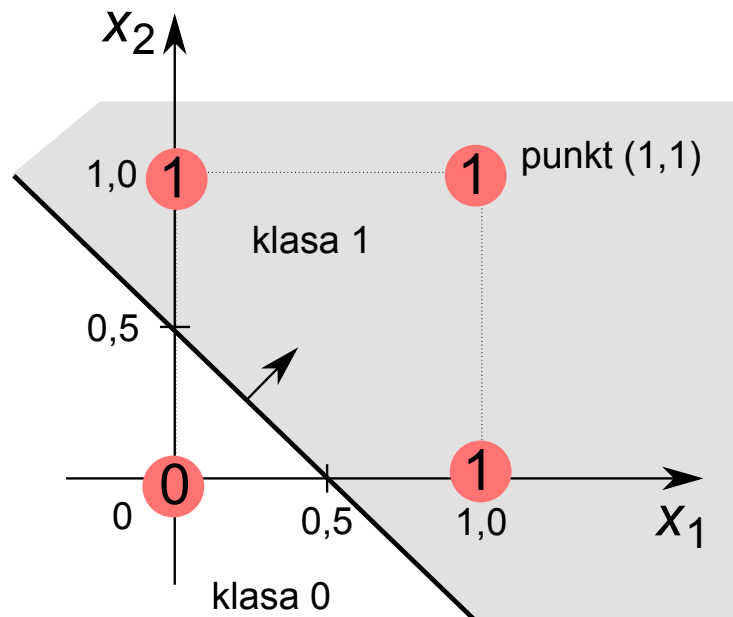
# Możliwość reprezentowania funkcji logicznych

alternatywa:  $x_1 \vee x_2$

$$w_1 = 1$$

$$w_2 = 1$$

$$b = -0.5$$

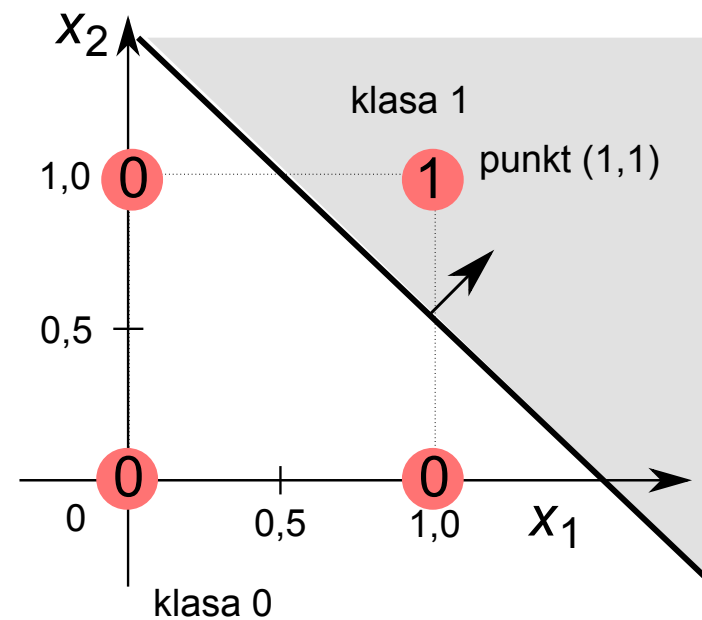


koniunkcja:  $x_1 \wedge x_2$

$$w_1 = 1$$

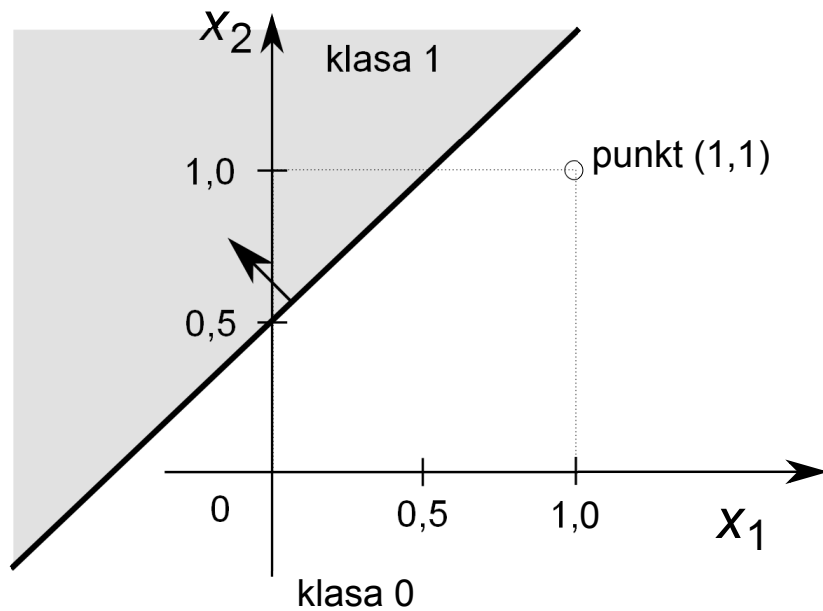
$$w_2 = 1$$

$$b = -1.5$$

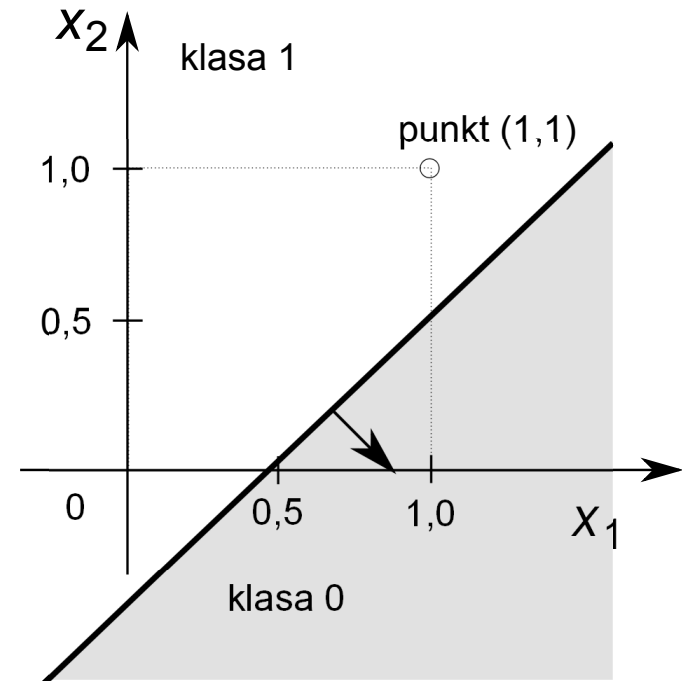


# Możliwość reprezentowania funkcji logicznych

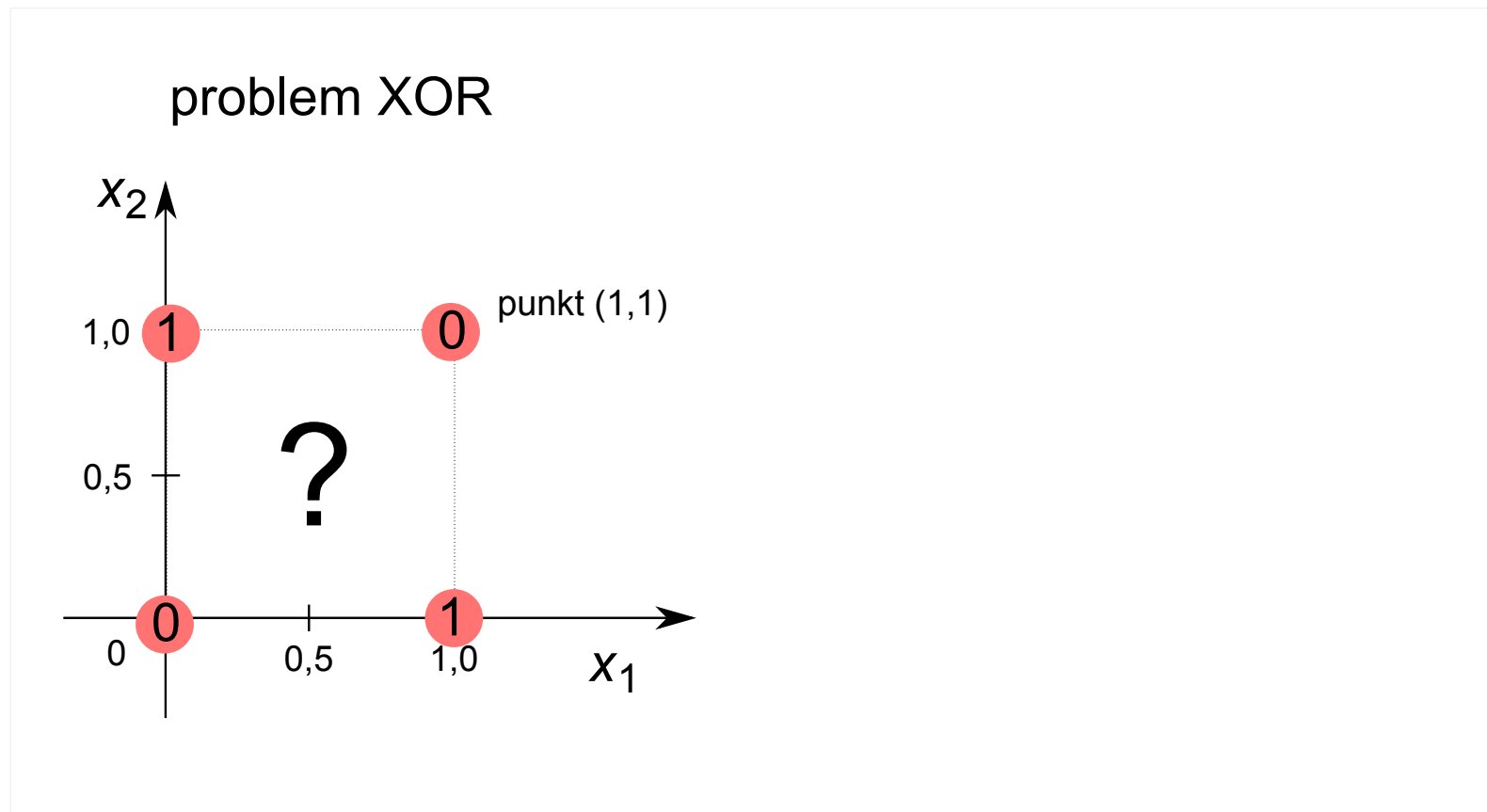
$$X_1 \wedge \sim X_2 \quad \begin{array}{l} w_1 = -1 \\ w_2 = 1 \\ b = -0.5 \end{array}$$



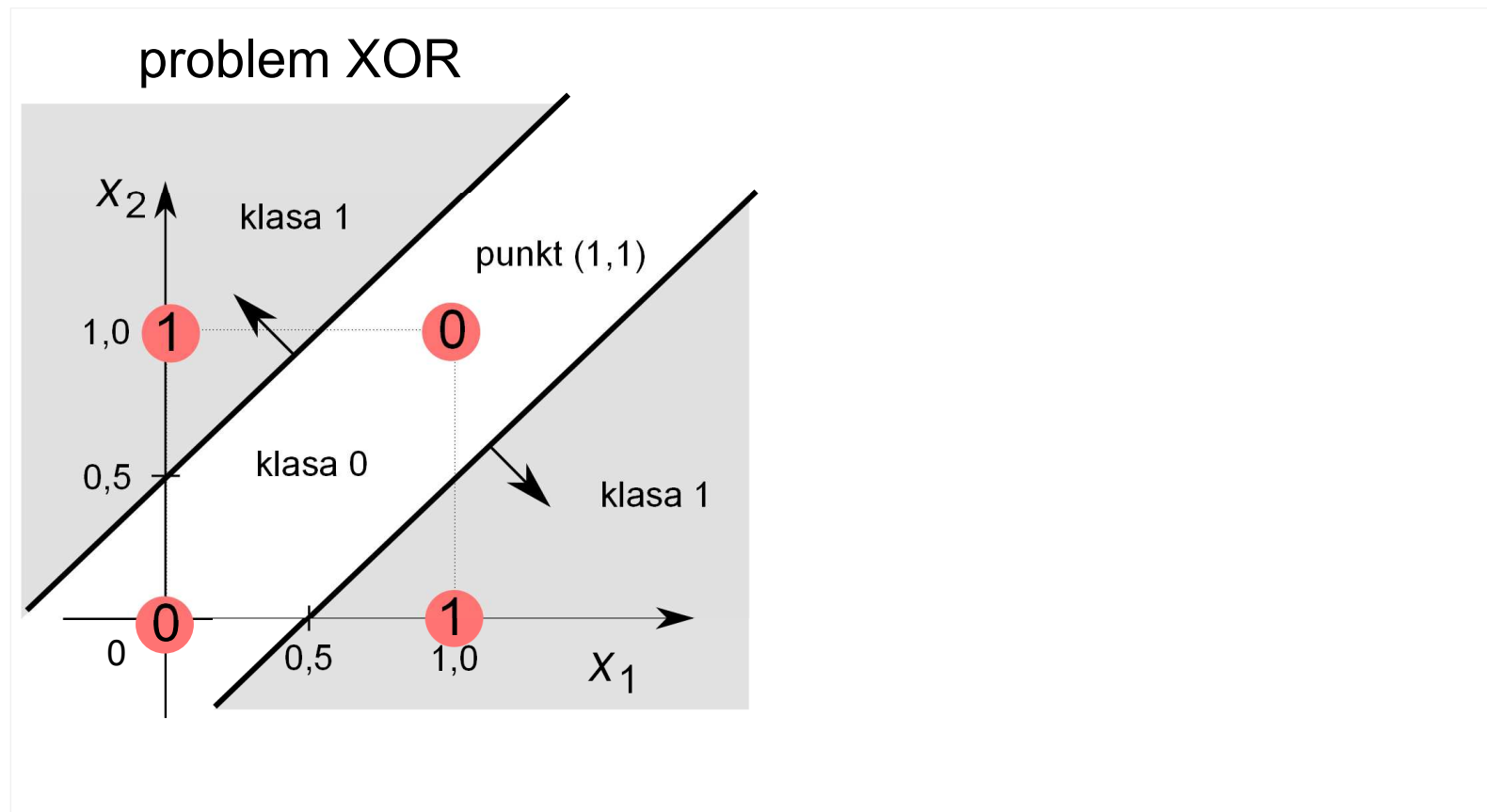
$$\sim X_1 \wedge X_2 \quad \begin{array}{l} w_1 = 1 \\ w_2 = -1 \\ b = -0.5 \end{array}$$



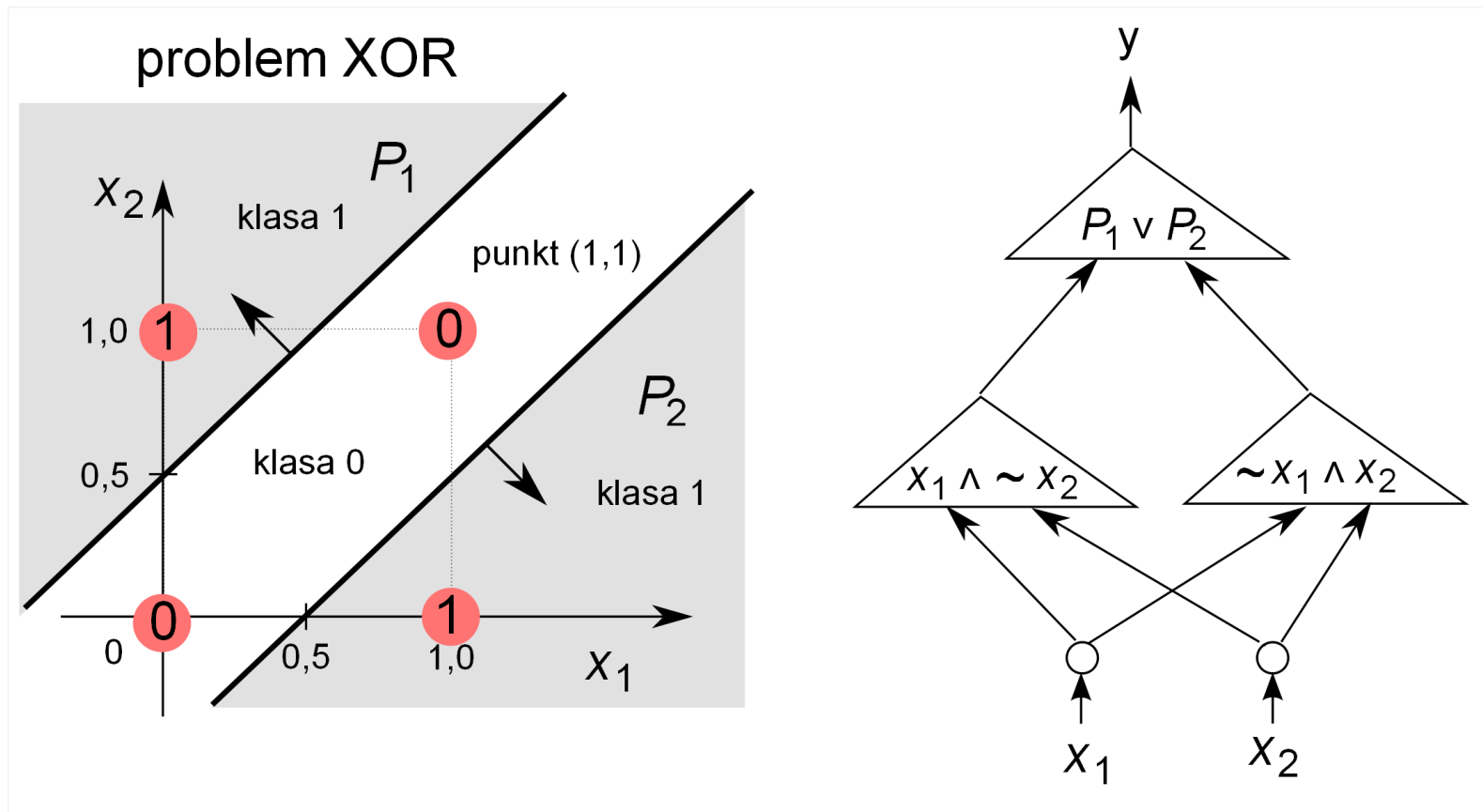
# Problem parzystości - XOR



# XOR - interpretacja geometryczna



# Sieć dwuwarstwowa reprezentująca XOR





# Metody uczenia perceptronu

Metody uczenia:

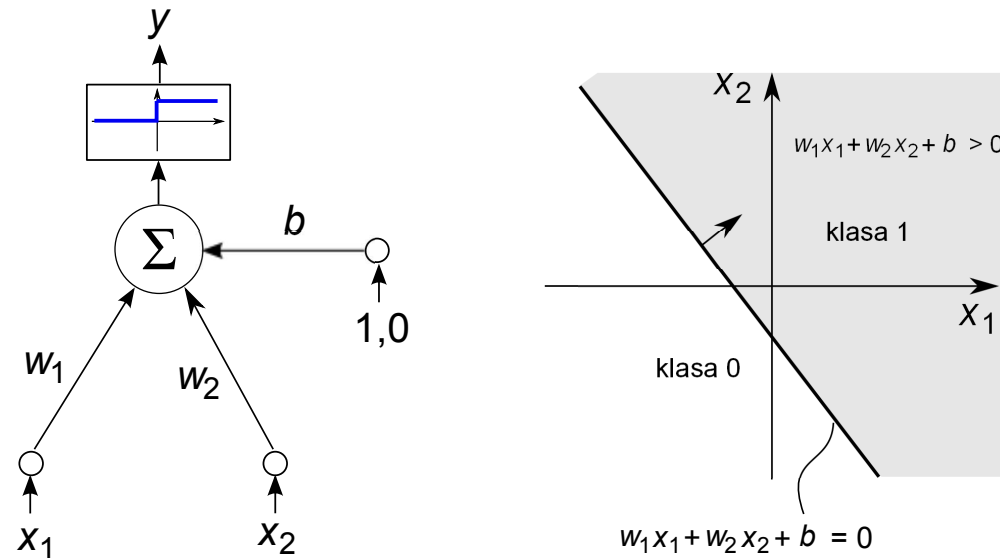
- Hebba:  $\mathbf{w}' \leftarrow \mathbf{w} + \alpha y_i \mathbf{x}_i$  (uczenie bez nauczyciela),
- Delta:  $\mathbf{w}' \leftarrow \mathbf{w} + \alpha (d_i - y_i) \mathbf{x}_i$ ,
- uogólniona Delta:  $\mathbf{w}' \leftarrow \mathbf{w} + \alpha (d_i - y_i) f'(\mathbf{x}_i) \mathbf{x}_i$ ,

gdzie  $y_i$  - wyjście po prezentacji  $i$ -tego przykładu  $\mathbf{x}_i$ ,  $d_i$  - pożądana wartość (klasa) na wyjściu,  $\alpha$  - współczynnik szybkości uczenia, np. 0,5,  $f'(\mathbf{x}_i)$  - pochodna funkcji aktywacji.

Sposoby prezentacji przykładów uczących:

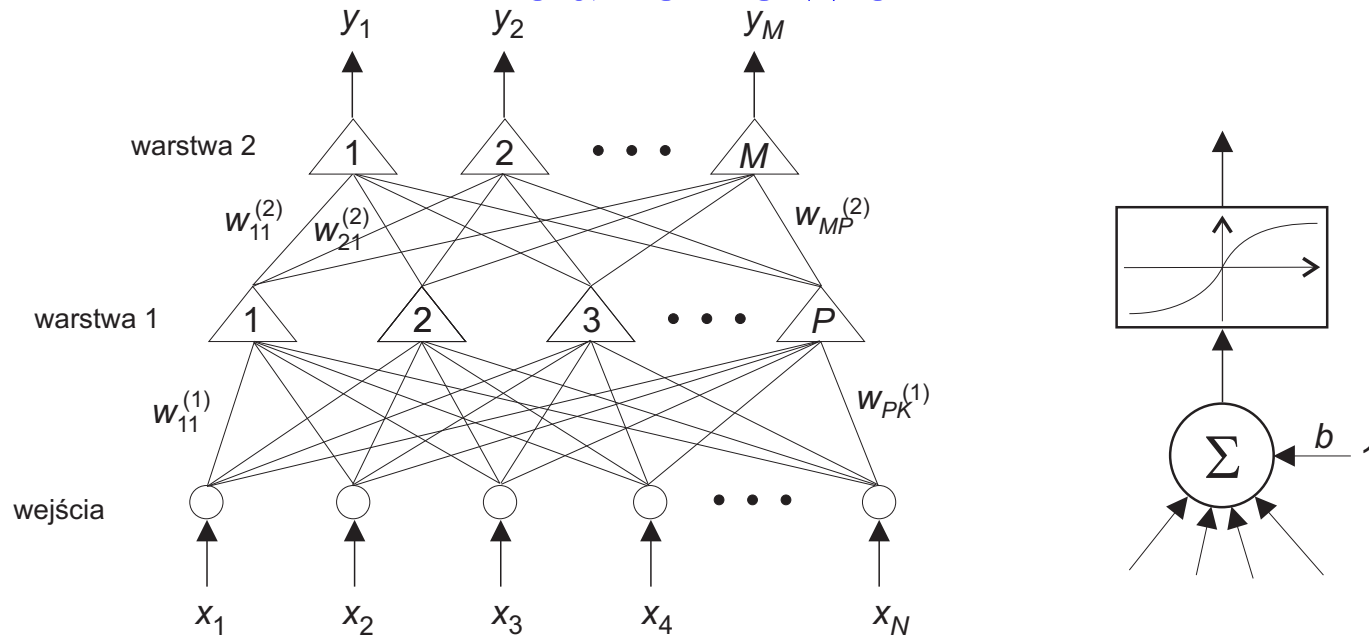
- Przyrostowy - aktualizacja wag po prezentacji każdego przykładu)
- Wsadowy (ang. *batch*) - aktualizacja wag po prezentacji wszystkich przykładów)
- *Mini-batch* - aktualizacja wag po prezentacji niewielkiego losowego podzbioru przykładów

# Przykład uczenia perceptronu z progową unipolarną funkcją aktywacji



- Warunki początkowe:  $w_1 = w_2 = 1, b = 0$
- Współczynnik szybkości uczenia:  $\alpha = 0.5$
- Reguła korekty wag Delta:  $w' \leftarrow w + \alpha(d_i - y_i)\mathbf{x}_i$
- Sposób prezentacji przykładów uczących: przyrostowy

# Klasyczne wielowarstwowe sztuczne sieci neuronowe

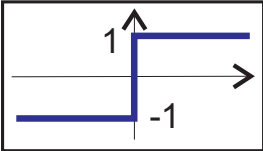
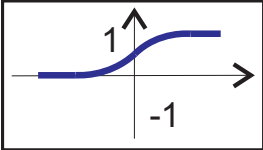
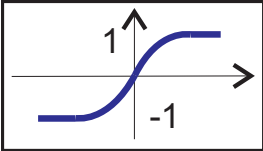
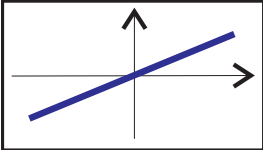
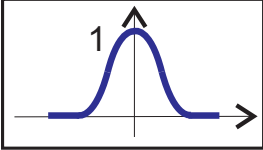


Przykładowa jednokierunkowa sztuczna sieć neuronowa, z prawej strony schemat pojedynczego neuronu

Funkcja reprezentowana przez  $i$ -te wyjście sieci:

$$y_i = f_{akt}^{(2)} \left( \sum_{j=1}^P w_{ij}^{(2)} f_{akt}^{(1)} \left( \sum_{k=1}^N w_{jk}^{(1)} x_k + b_j^{(1)} \right) + b_i^{(2)} \right)$$

# Zestawienie funkcji aktywacji neuronu

funkcja aktywacji $f_{akt}$	zakres wartości na wyjściu	wykres funkcji	opis
model neuronu z iloczynem skalarnym $s = \mathbf{w}^T \mathbf{x} + b$			
$\begin{cases} 0 & \text{gdy } s < 0 \\ 1 & \text{gdy } s \geq 0 \end{cases}$	$y \in \{-1, 1\}$		funkcja skoku jednostkowego
$1/(1 + e^{-s})$	$y \in (0, 1)$		funkcja sigmoidalna unipolarna
$2/(1 + e^{-2s}) - 1$	$y \in (-1, 1)$		funkcja sigmoidalna bipolarna
$s$	$y \in (-\infty, \infty)$		funkcja liniowa – identycznościowa
model z miarą odległościową $s = \ \mathbf{w} - \mathbf{x}\  b$			
$e^{-s^2}$	$y \in (0, 1)$		funkcja radialna

# Uczenie sieci wielowarstwowych: algorytm propagacji wstecznej błędu

Za miarę błędu klasyfikacji przyjmuje się w klasycznych SSN błąd średniokwadratowy:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{\mu=1}^{K_u} \sum_{i=1}^M (d_{i\mu} - y_{i\mu})^2,$$

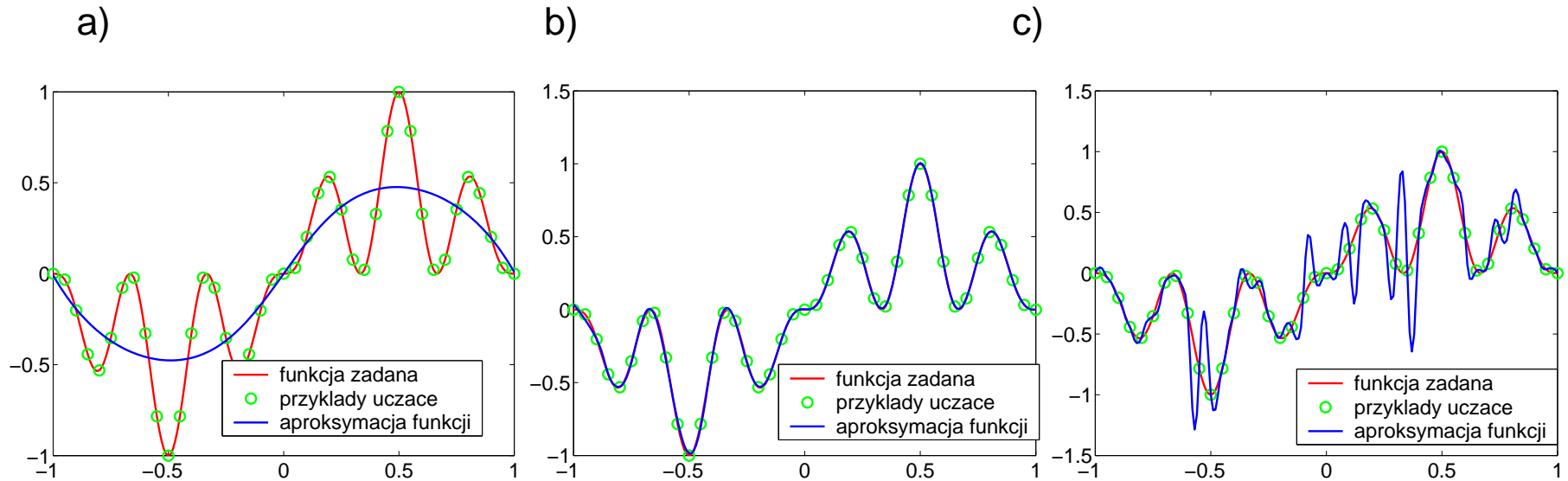
gdzie  $K_u$  jest liczbą przykładów do uczenia,  $M$  jest liczbą neuronów w warstwie wyjściowej, zaś  $d_{i\mu}$  jest pożądaną wartością na wyjściu  $i$ -tego neuronu, gdy na wejściu sieci podano  $\mu$ -ty obraz.

Uczenie polega na modyfikacji wektora wszystkich wag sieci w kierunku największego spadku funkcji błędu. Kierunek ten wyznaczany jest przez gradient funkcji błędu:  $\nabla E(\mathbf{w}) = [\partial E / \partial w_1, \partial E / \partial w_2, \dots, \partial E / \partial w_L]^T$ . Kierunek gradientu wyznacza kierunek największego wzrostu funkcji błędu w  $L$ -wymiarowej przestrzeni wag, więc w celu zmniejszenia błędu należy zmodyfikować wagi w kierunku przeciwnym:

$$\mathbf{w}' = \mathbf{w} + \Delta \mathbf{w} = \mathbf{w} - \alpha \nabla E(\mathbf{w}),$$

gdzie  $\alpha$  jest współczynnikiem szybkości uczenia.

# Uczenie a uogólnianie



Aproksymacja funkcji  $\sin^2(3\pi x) \sin(\pi x)$  z wykorzystaniem dwuwarstwowej sztucznej sieci neuronowej o a) 3, b) 10 oraz c) 50 neuronach w pierwszej warstwie

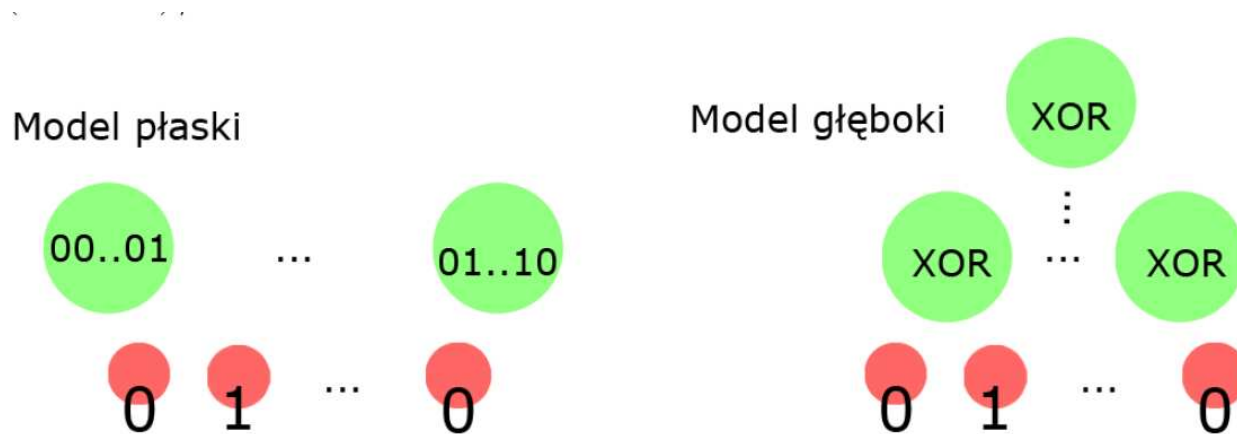
# Uczenie a uogólnianie

Metody poprawy uogólniania:

- Dobór architektury sieci (liczba warstw, liczby neuronów w warstwach) do problemu.
- Zatrzymanie uczenia, gdy błąd walidacji (testu) zaczyna rosnać (ang. *early stopping*).
- Regularyzacja:
  - Dodanie szumu do danych podczas uczenia.
  - Dodanie specjalnego członu do funkcji błędu:  $E' = E + \frac{\lambda}{2K} \sum w^2$  ograniczającego wzrost wartości wag.
  - Blokowanie aktywacji losowo wybranych neuronów podczas uczenia (ang. *dropout*)

# Klasyczne Sztuczne sieci neuronowe: problemy z uczeniem

- Zanikający gradient w przypadku sieci o dużej liczbie warstw.
- Słabe uogólnianie w przypadku dużej liczby warstw.
- Wymagana duża liczba neuronów w przypadku sieci dwuwarstwowych.



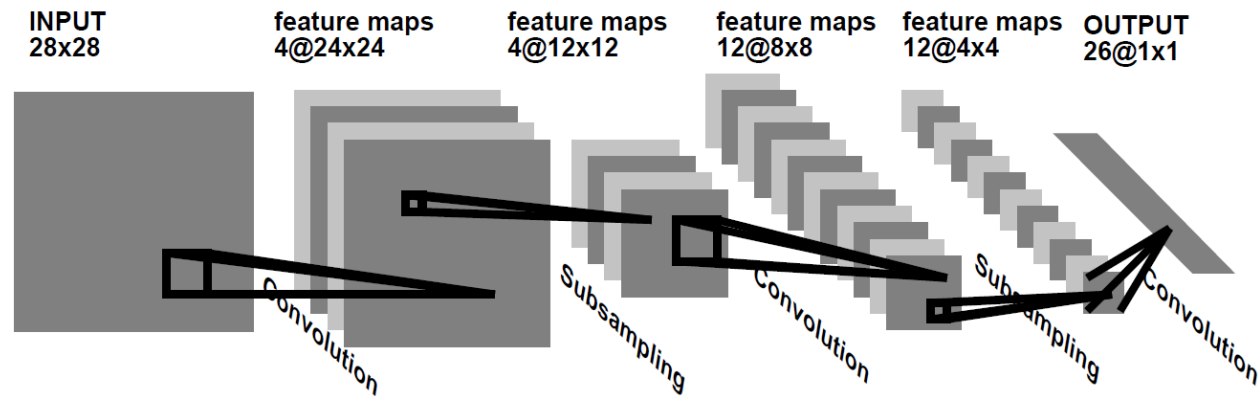


# Głębokie uczenie (*Deep Learning - DN*)

Dwa główne kierunki rozwoju:

- Głębokie sieci przekonań (*Deep Belief Networks - DBN*).
- Sieci realizujące funkcje splotu (*Convolutional Neural Networks - CNN*).
- Sieci zawierające warstwy LSTM do analizy sekwencji.
- Sieci zawierające warstwy kapsułowe (*Capsules*).

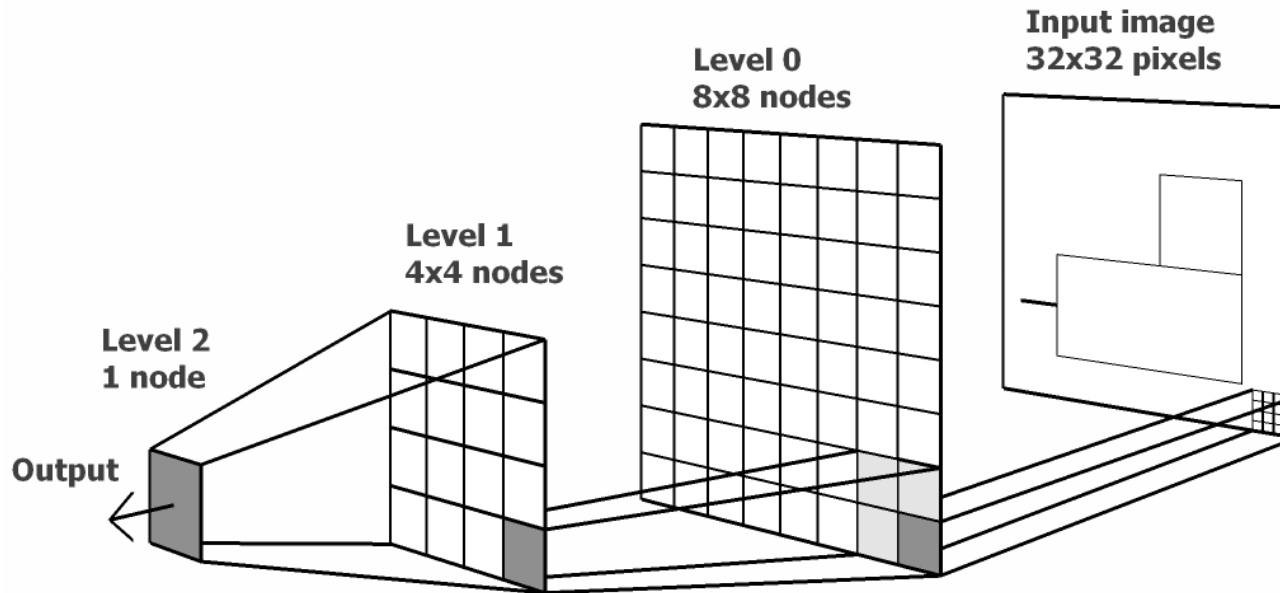
# Sieć realizująca funkcje splotu



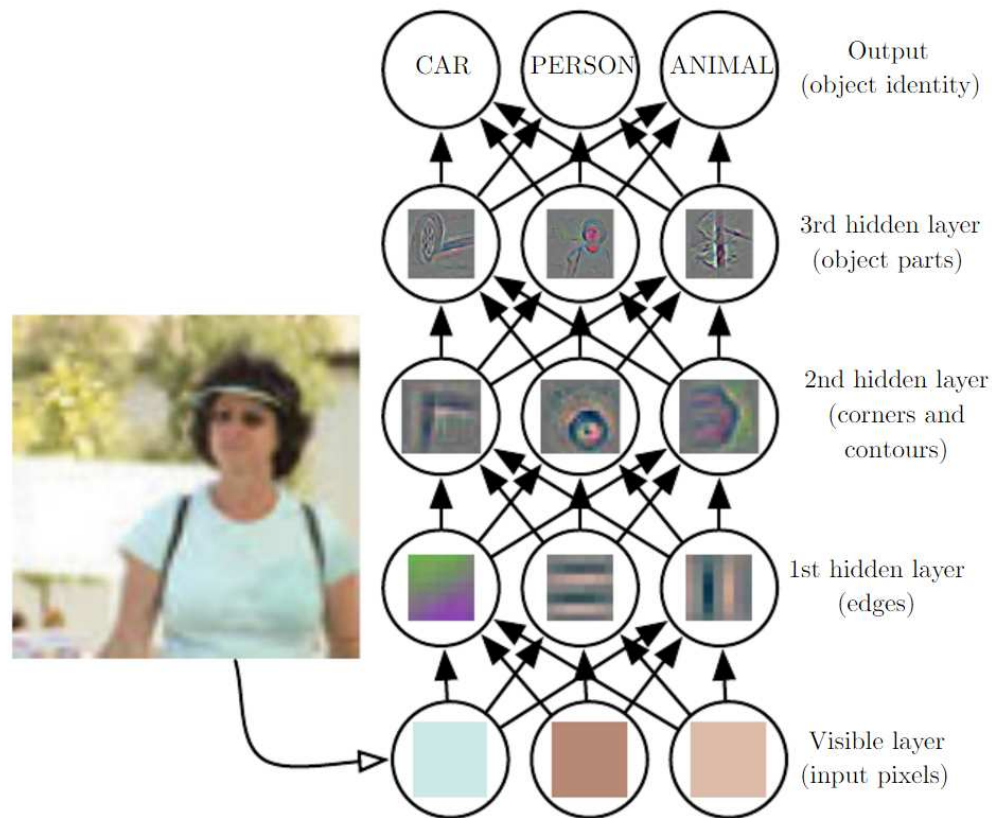
Techniki poprawiające szybkość uczenia i uogólnianie:

- Funkcja aktywacji ReLU:  $f(x) = \max(x, 0)$ .
- Redukcja rozdzielczości obrazu (*subsampling*).
- Regularyzacja metodą *Dropout*: aktywacja losowo wybranych neuronów.
- Uczenie metodą stochastycznego spadku gradientu (*SGD*).
- Funkcja entropii krzyżowej jako funkcja błędu zamiast średniego błędu kwadratowego.

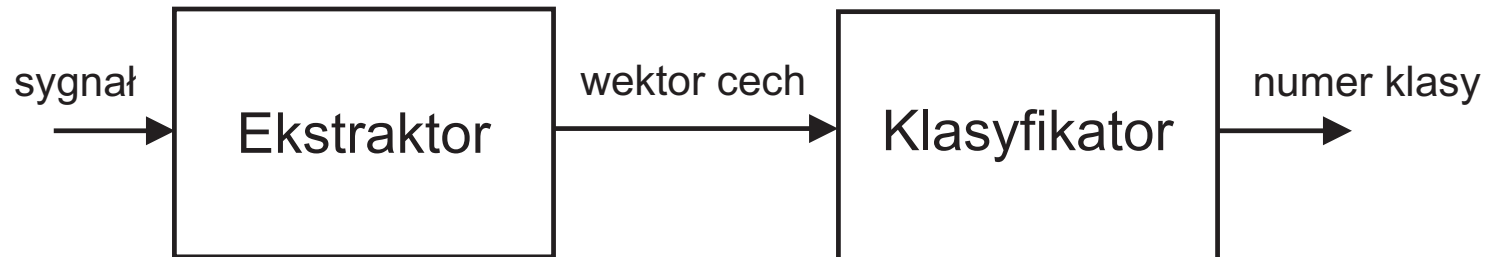
# Funkcja splotu i podpróbkowania



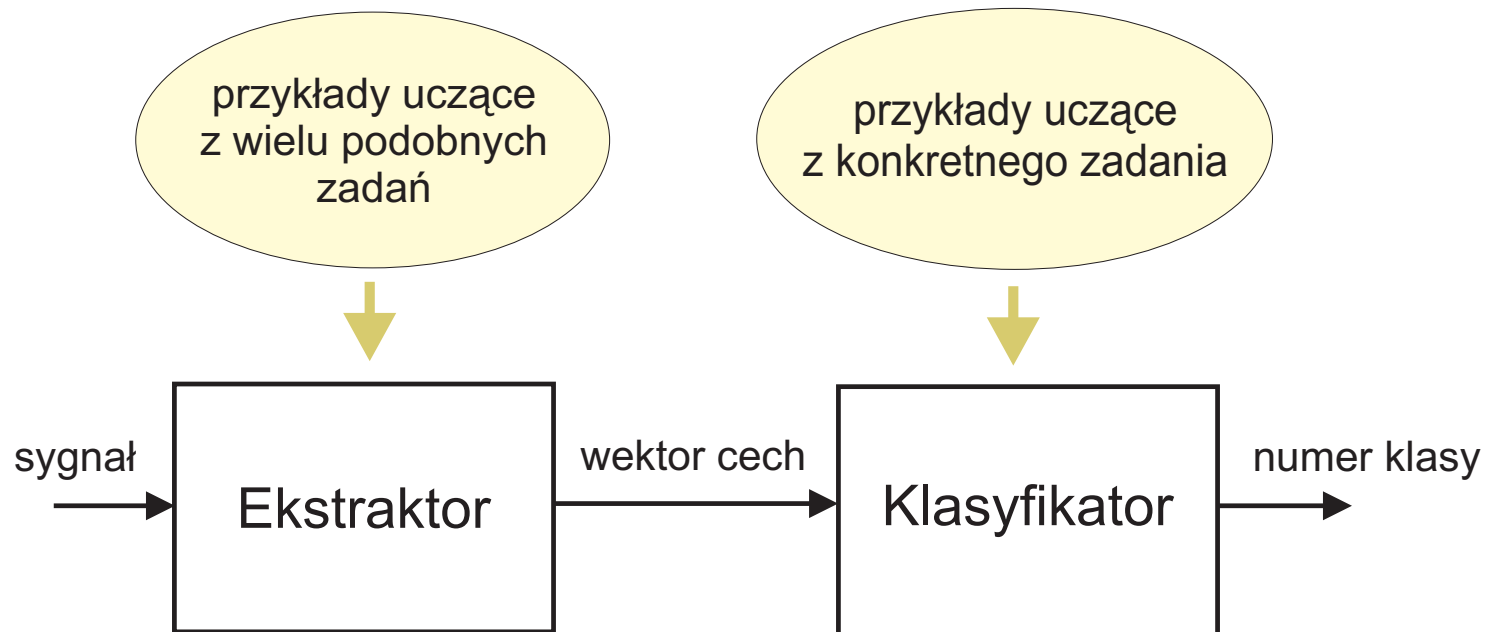
# Sieć realizująca funkcje splotu - hierarchia przekształceń



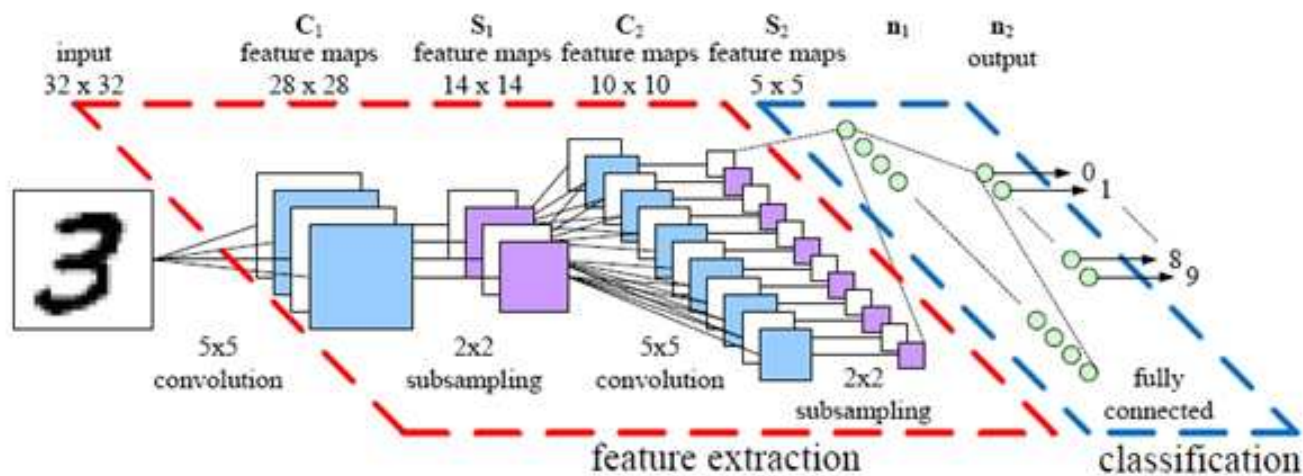
# Schemat procesu rozpoznawania



# Schemat procesu rozpoznawania - dane do uczenia



# Sieć realizująca funkcje splotu - ekstrakcja cech



# Sieć autoasocjacyjna - ekstrakcja cech

