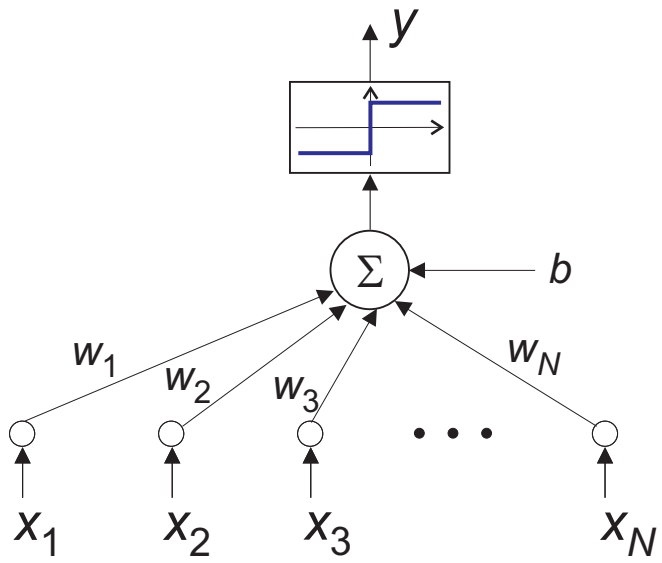
Metody klasyfikacji danych - część 2

Jerzy Dembski

Plan wykładu

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

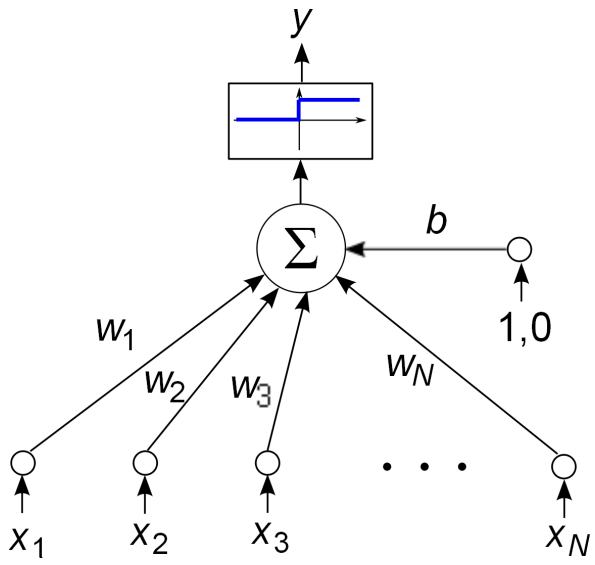
Sztuczne sieci neuronowe: Model neuronu z progową bipolarną funkcją aktywacji



Funkcja, którą taka sieć reprezentuje wyraża się wzorem:

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

Model neuronu z progową unipolarną funkcją aktywacji

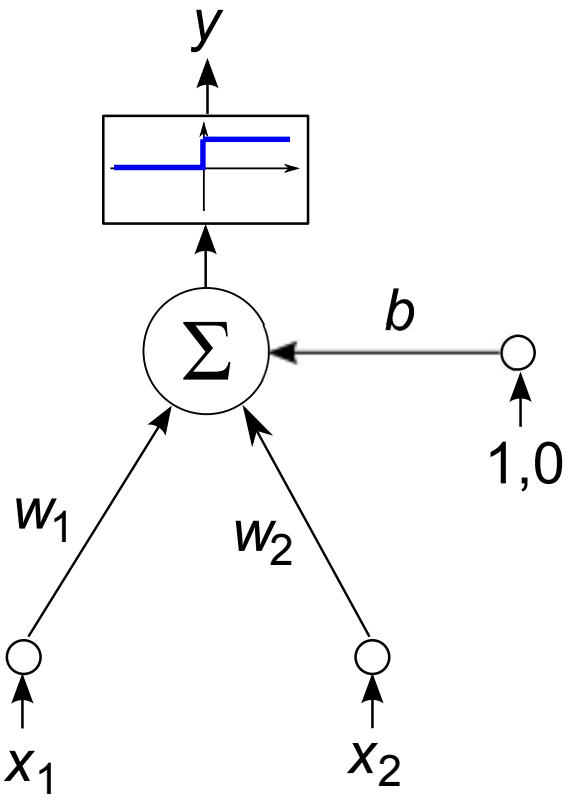


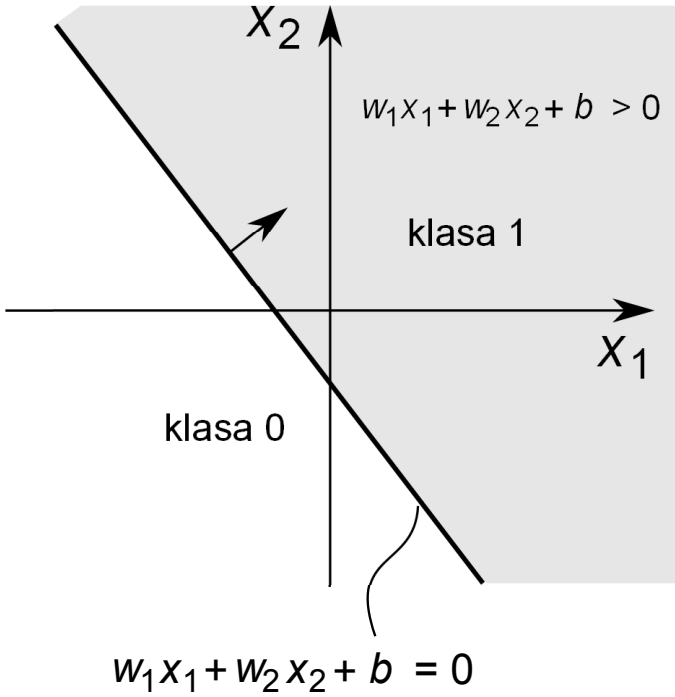
Funkcja, którą taka sieć reprezentuje, wyraża się wzorem:

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

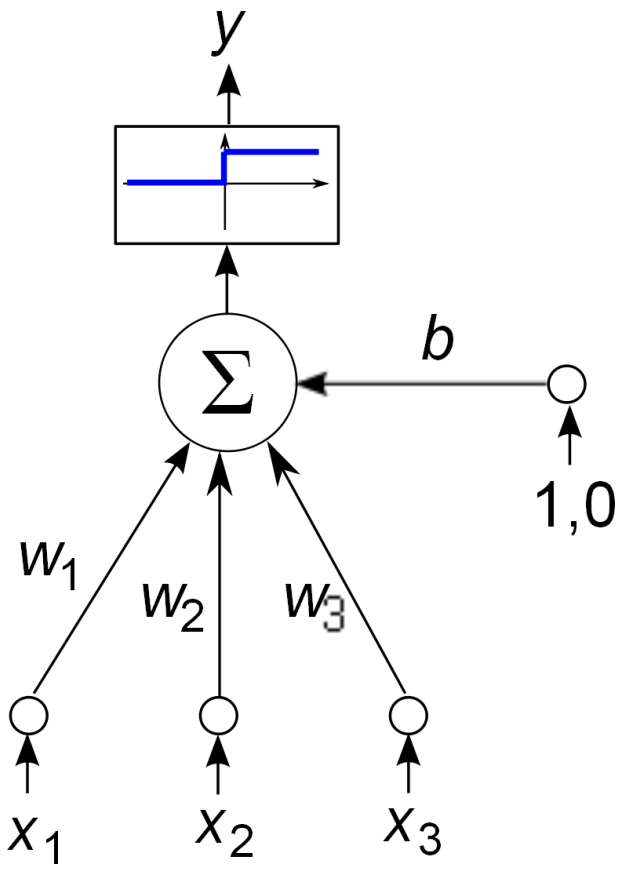
Interpretacja geometryczna funkcji reprezentowanej przez neuron

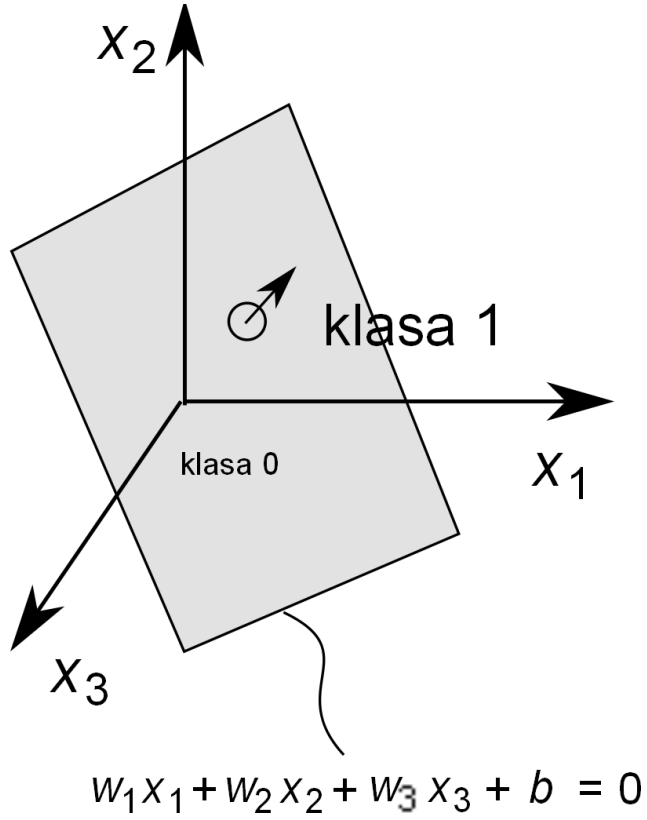
Prosta wyznaczająca półpaszczyznę klasy 1.





Płaszczyzna separująca przestrzeń na dwie półprzestrzenie

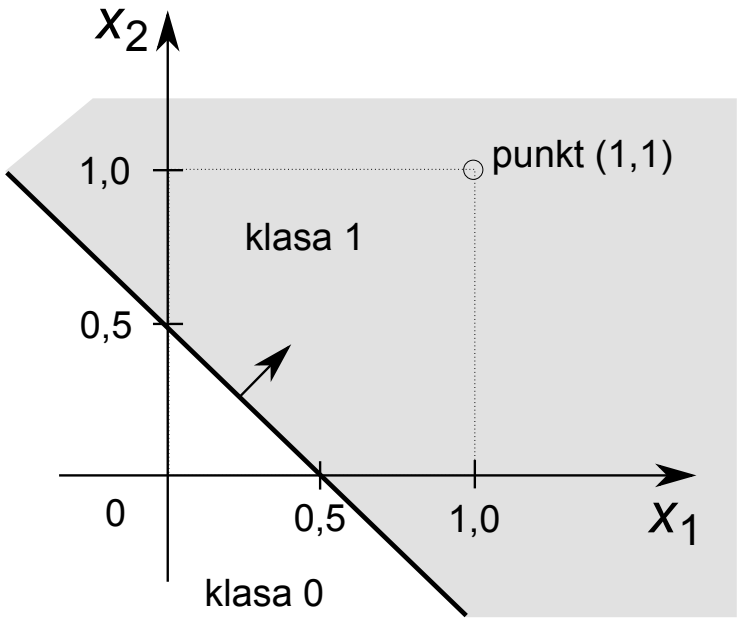




Możliwość reprezentowania funkcji logicznych

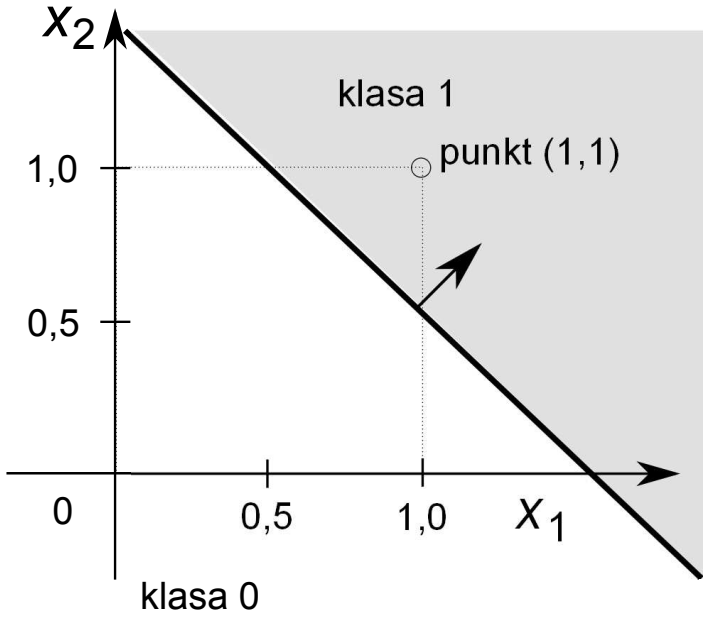
alternatywa: $x_1 \vee x_2$

$w_1 = 1$
$w_2 = 1$
$b = -0.5$



koniunkcja: $x_1 \wedge x_2$

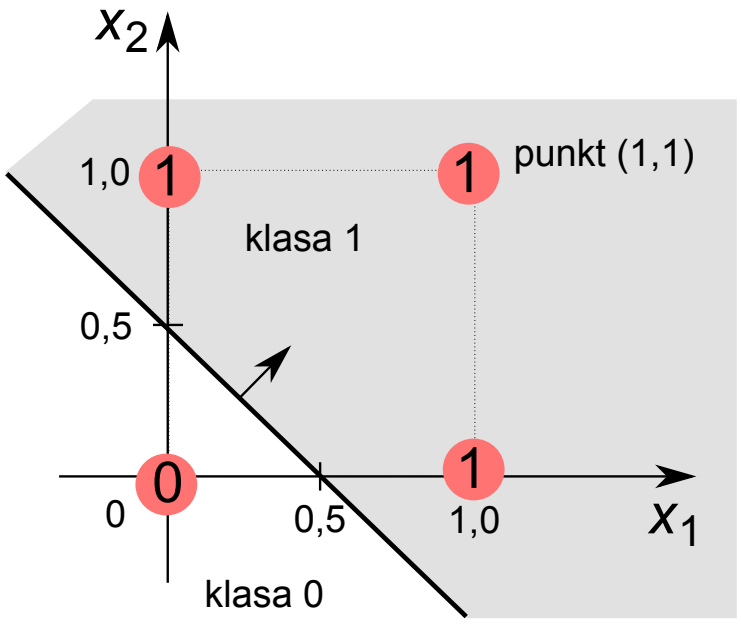
$w_1 = 1$
$w_2 = 1$
$b = -1.5$



Możliwość reprezentowania funkcji logicznych

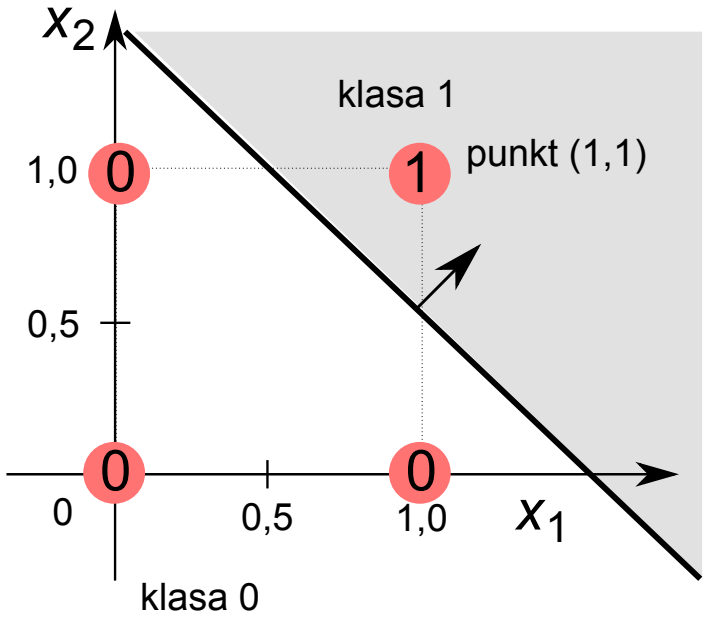
alternatywa: $x_1 \vee x_2$

$w_1 = 1$
$w_2 = 1$
$b = -0.5$



koniunkcja: $x_1 \wedge x_2$

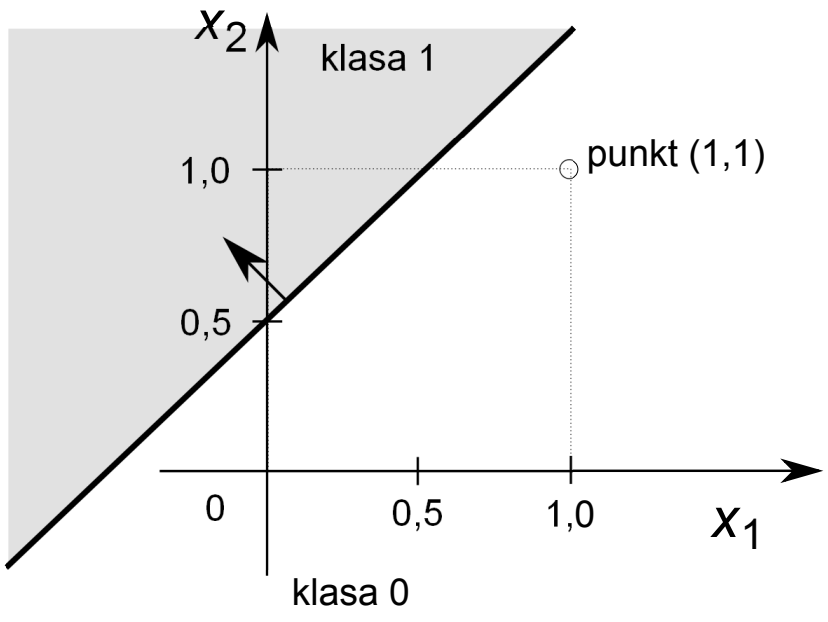
$w_1 = 1$
$w_2 = 1$
$b = -1.5$



Możliwość reprezentowania funkcji logicznych

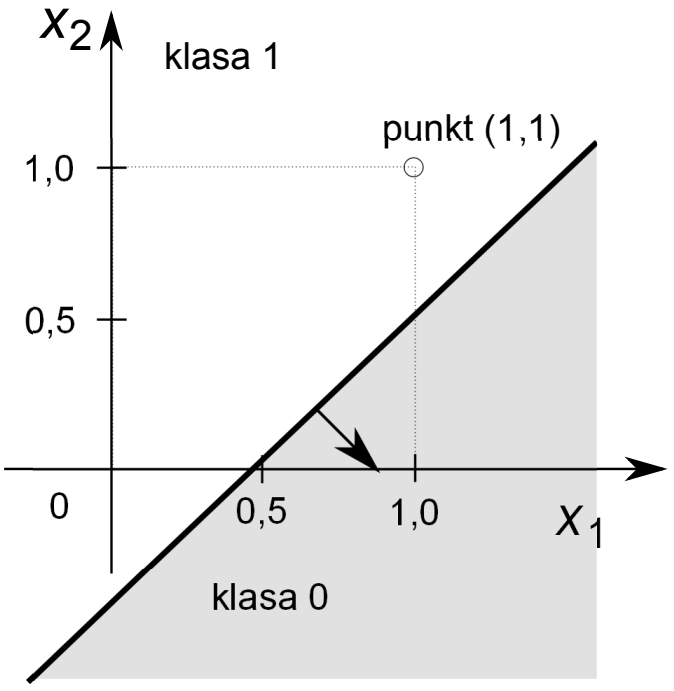
$x_1 \wedge \sim x_2$

$w_1 = -1$
$w_2 = 1$
$b = -0.5$

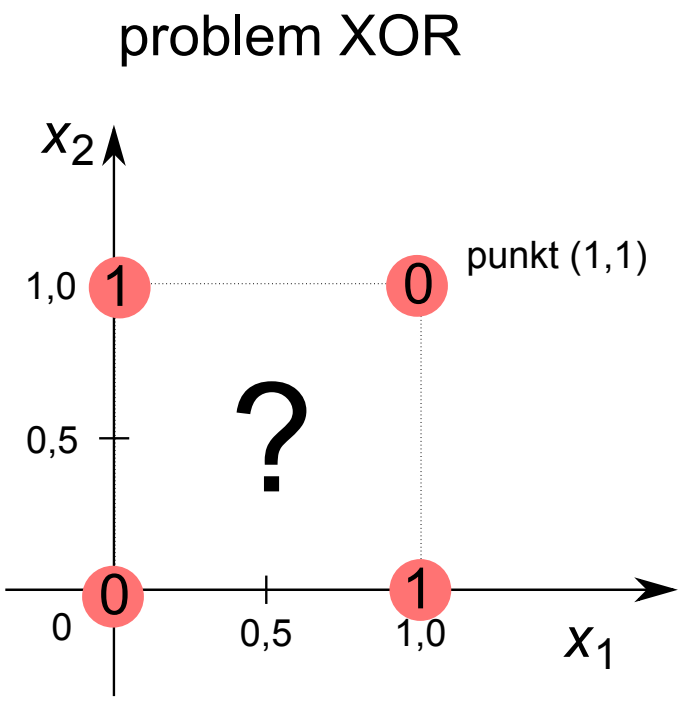


$\sim x_1 \wedge x_2$

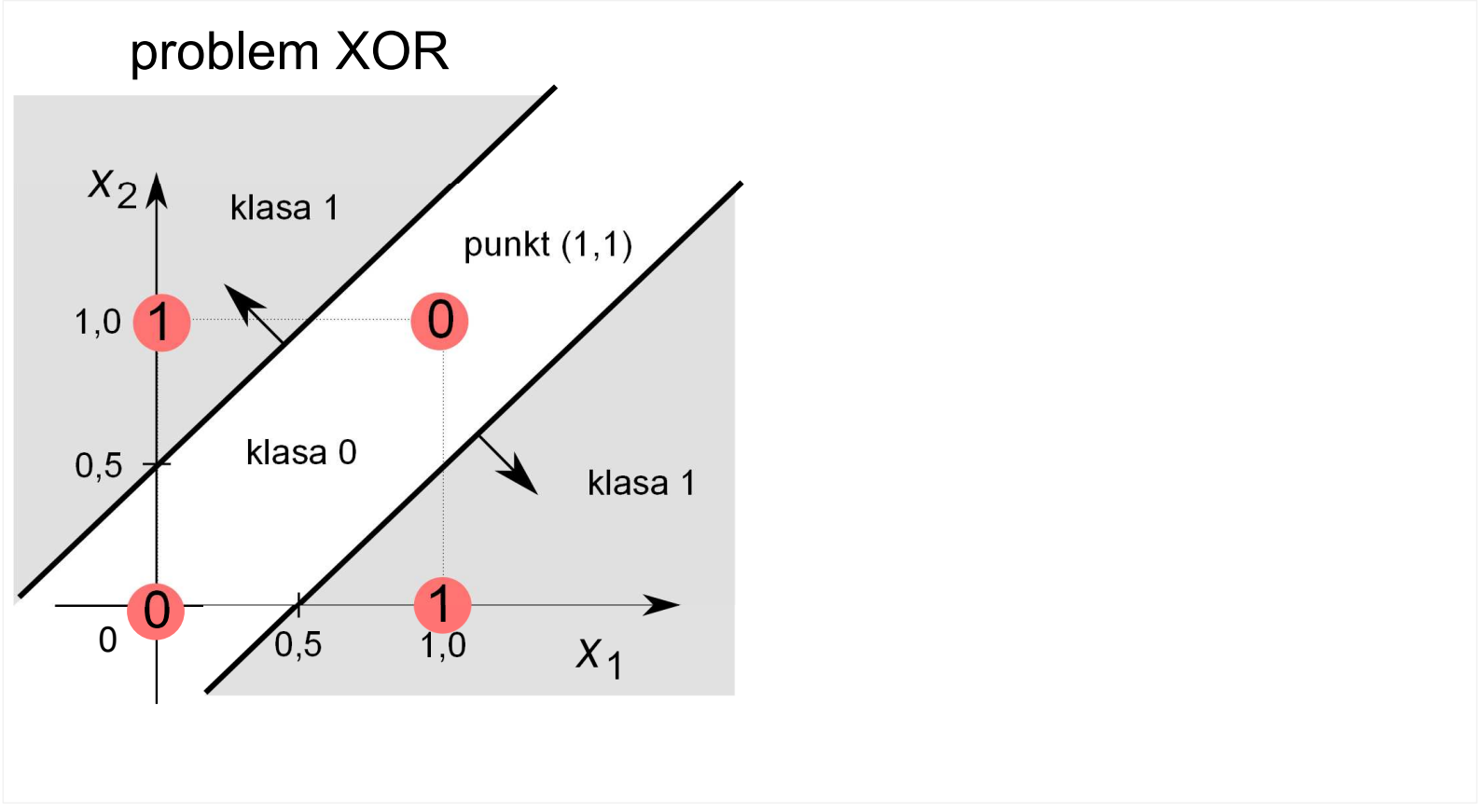
$w_1 = 1$
$w_2 = -1$
$b = -0.5$



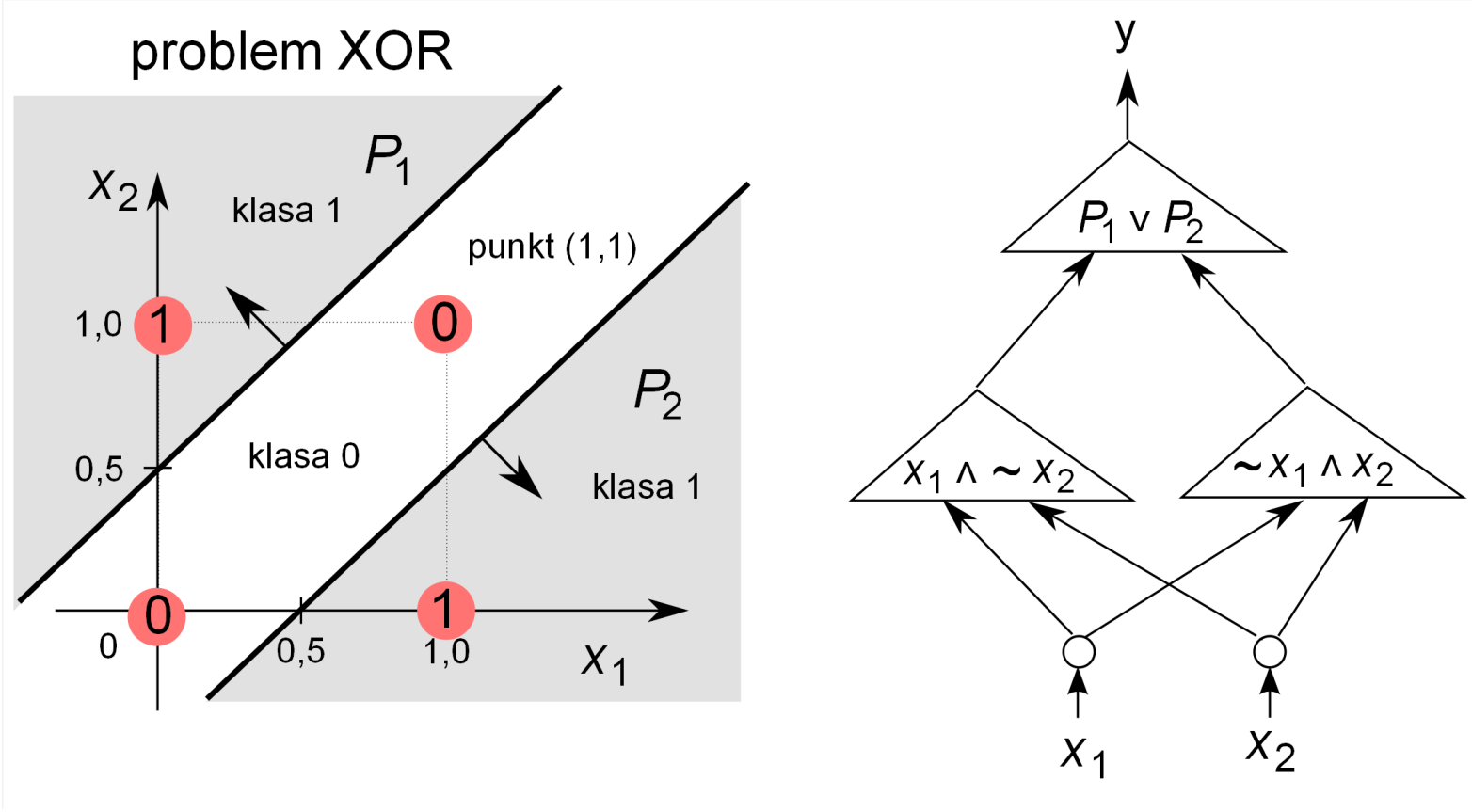
Problem parzystości - XOR



XOR - interpretacja geometryczna



Sieć dwuwarstwowa reprezentująca XOR



Metody uczenia perceptronu

Metody uczenia:

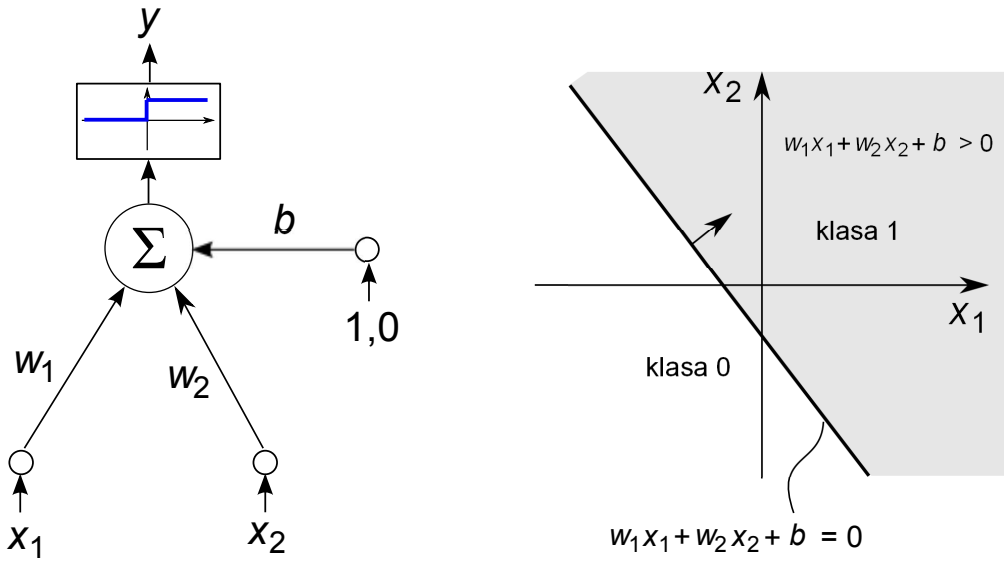
- Hebba: $\mathbf{w'} \leftarrow \mathbf{w} + \alpha y_i \mathbf{x_i}$ (uczenie bez nauczyciela),
- Delta: $\mathbf{w'} \leftarrow \mathbf{w} + \alpha(d_i - y_i)\mathbf{x_i}$,
- uogólniona Delta: $\mathbf{w'} \leftarrow \mathbf{w} + \alpha(d_i - y_i)f'(\mathbf{x_i})\mathbf{x_i}$,

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

Sposoby prezentacji przykładów uczących do aktualizacji wag:

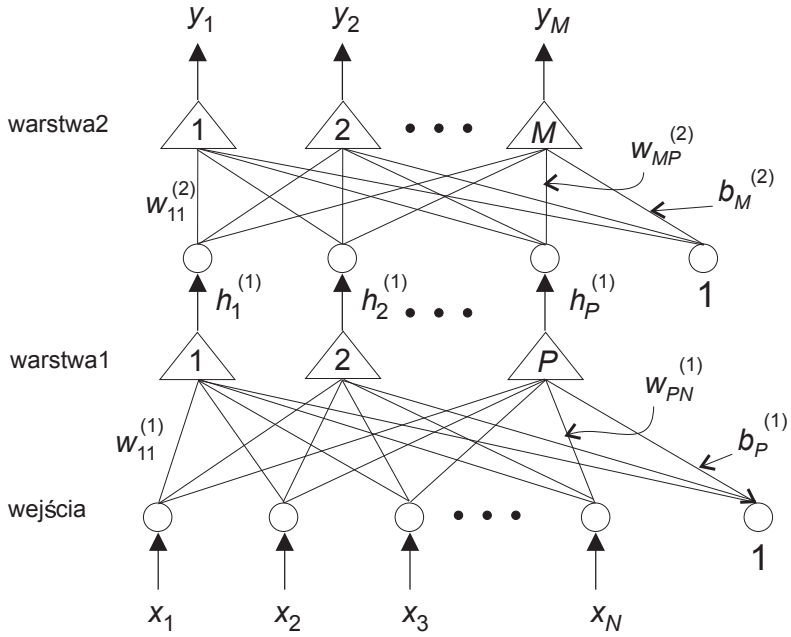
- Przyrostowy - aktualizacja wag po prezentacji każdego przykładu)
- Wsadowy (ang. *batch*) - aktualizacja wag po prezentacji wszystkich przykładów)
- *Mini-batch* - aktualizacja wag po prezentacji niewielkiego losowego podzbioru przykładów

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: $\mathbf{w}' \leftarrow \mathbf{w} + \alpha(d_i - y_i)\mathbf{x_i}$
- Sposób prezentacji przykładów uczących: przyrostowy

Klasyczne wielowarstwowe sztuczne sieci neuronowe

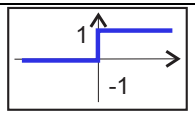
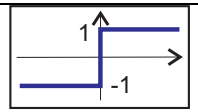
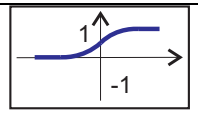
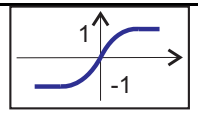
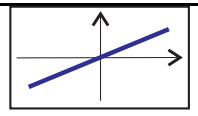
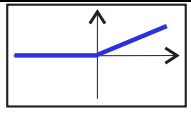
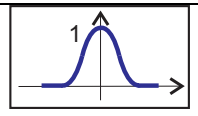


Przykładowa jednokierunkowa dwuwarstwowa sztuczna sieć neuronowa, z prawej strony schemat pojedynczego neuronu.

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 $a = \mathbf{w}^T\mathbf{x} + b$			
$\begin{cases} 0 & \text{gdy} \quad a < 0 \\ 1 & \text{gdy} \quad a \geq 0 \end{cases}$	$y \in \{0, 1\}$		funkcja progowa unipolarna
$\begin{cases} -1 & \text{gdy} \quad a < 0 \\ 1 & \text{gdy} \quad a \geq 0 \end{cases}$	$y \in \{-1, 1\}$		funkcja progowa bipolarna
$1/(1 + e^{-a})$	$y \in (0, 1)$		funkcja sigmoidalna unipolarna
$2/(1 + e^{-2a}) - 1$	$y \in (-1, 1)$		funkcja sigmoidalna bipolarna
a	$y \in (-\infty, \infty)$		funkcja liniowa – identycznościowa
$\begin{cases} 0 & \text{gdy} \quad a < 0 \\ a & \text{gdy} \quad a \geq 0 \end{cases}$	$y \in (0, 1)$		funkcja ReLU (*rectified linear unit*)
model z miarą odległościową $a = \|\mathbf{w} - \mathbf{x}\|\, b$			
e^{-a^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:

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

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 μ-ty obraz wejściowy.
Uczenie polega na obliczaniu gradientu funkcji błędu:
$\nabla J(\mathbf{w}) = [\partial J/\partial w_1, \partial J/\partial w_2, \ldots, \partial J/\partial w_L]^T$. Kierunek gradientu wyznacza kierunek największego wzrostu funkcji błędu w L-wymiarowej przestrzeni wag, więc w celu zmniejszenia błędu jest modyfikowany w kierunku przeciwnym:

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

gdzie α jest współczynnikiem szybkości uczenia.

Uczenie sieci wielowarstwowych: algorytm propagacji wstecznej błędu

Po rozwinięciu wyjść $y_{i\mu}$ z ostatniej warstwy:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{\mu=1}^{K_u}\sum_{i=1}^{M}\left[d_{i\mu} - f_{akt}^{(2)}\left(\sum_{j=1}^{P} w_{ij}^{(2)} h_{j\mu} + b_i^{(2)}\right)\right]^2 .$$

Po rozwinięciu wyjść $h_{j\mu}$ z pierwszej warstwy:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{\mu=1}^{K_u}\sum_{i=1}^{M}\left[d_{i\mu} - f_{akt}^{(2)}\left(\sum_{j=1}^{P} w_{ij}^{(2)} f_{akt}^{(1)}\left(\sum_{n=1}^{N} w_{jn}^{(1)} x_{n\mu} + b_j^{(1)}\right) + b_i^{(2)}\right)\right]^2 .$$

Uczenie sieci wielowarstwowych: algorytm propagacji wstecznej błędu

Korekta wag ostatniej warstwy:

$$\Delta w_{ij}^{(2)} = -\alpha \frac{\partial J(\mathbf{w})}{\partial w_{ij}^{(2)}} = \alpha \sum_{\mu=1}^{K_u} \sum_{m=1}^{M} (d_{m\mu} - y_{m\mu}) \frac{\partial y_{m\mu}}{\partial w_{ij}^{(2)}} = \alpha \sum_{\mu=1}^{K_u} (d_{i\mu} - y_{i\mu}) \frac{\partial y_{i\mu}}{\partial w_{ij}^{(2)}}.$$

Suma po neuronach w warstwie wyjściowej redukuje się do i-tego neuronu, ze względu na zerowanie się pochodnych $\frac{\partial y_{m\mu}}{\partial w_{ij}^{(2)}}$ dla $m \neq i$. Różniczkując $y_{i\mu} = f_{akt}^{(2)}(\sum_{j=1}^{P} w_{ij}^{(2)} h_{j\mu} + b_i^{(2)}) = f_{akt}^{(2)}(a_{i\mu}^{(2)})$ względem $w_{ij}^{(2)}$ otrzymujemy:

$$\Delta w_{ij}^{(2)} = \alpha \sum_{\mu=1}^{K_u} (d_{i\mu} - y_{i\mu}) f_{akt}^{(2)\prime}(a_{i\mu}^{(2)}) h_{j\mu}$$

gdzie $a_{i\mu}^{(2)}$ jest sygnałem pobudzenia, czyli ważoną sumą sygnałów na wyjściu neuronów z poprzedniej warstwy, zaś $f_{akt}^{(2)\prime}(a_{i\mu}^{(2)})$ pochodną funkcji aktywacji względem $a_{i\mu}^{(2)}$.

Uczenie sieci wielowarstwowych: algorytm propagacji wstecznej błędu

Korektę wag warstwy wyjściowej wygodnie jest przedstawić w postaci:

$$\Delta w_{ij}^{(2)} = \alpha \sum_{\mu=1}^{K_u} \delta_{i\mu}^{(2)} h_{j\mu},$$

gdzie $\delta_{i\mu}^{(2)} = (d_{i\mu} - y_{i\mu}^{(2)}) f_{akt}^{(2)\prime}(a_{i\mu}^{(2)})$ jest tzw. sygnałem błędu i-tego neuronu w warstwie wyjściowej.
Podobnie obliczane są korekty wag w niższych warstwach sieci, z tą różnicą, że sygnał błędu neuronu niższej warstwy jest sumą ważoną sygnałów błędów neuronów wyższej warstwy przemnożoną przez pochodną funkcji aktywacji neuronu niższej warstwy. Dla sieci dwuwarstwowej jest to $\delta_{j\mu}^{(1)} = f_{akt}^{(1)\prime}(a_{j\mu}^{(1)}) \sum_{i=1}^{M} w_{ij}^{(2)} \delta_{i\mu}^{(2)}$.
W przypadku, gdy niższa warstwa jest jednocześnie pierwszą warstwą, korekta wag zależy bezpośrednio od wartości na wejściu sieci:

$$\Delta w_{ij}^{(1)} = \alpha \sum_{\mu=1}^{K_u} \delta_{i\mu}^{(1)} x_{j\mu}.$$

Uczenie sieci wielowarstwowych: algorytm propagacji wstecznej błędu

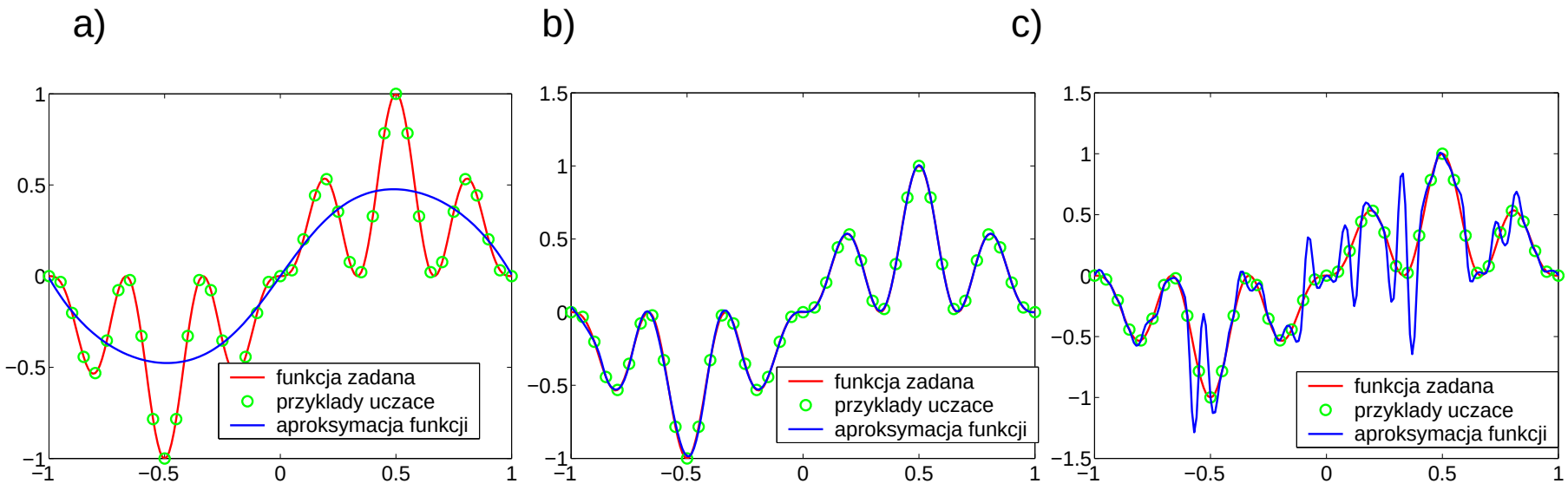
W ogólności, dla sieci o wielu warstwach ukrytych, sygnał błędu dla warstwy t:

$$\delta_{j\mu}^{(t)} = f_{akt}^{(t)\prime}(a_{j\mu}^{(t)}) \sum_{i=1}^{P^{(t)}} w_{ij}^{(t+1)} \delta_{i\mu}^{(t+1)},$$

gdzie $P^{(t)}$ jest liczbą neuronów w warstwie t, zaś korekta wag:

$$\Delta w_{ij}^{(t)} = \alpha \sum_{\mu=1}^{K_u} \delta_{i\mu}^{(t)} h_{j\mu}^{(t-1)}.$$

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

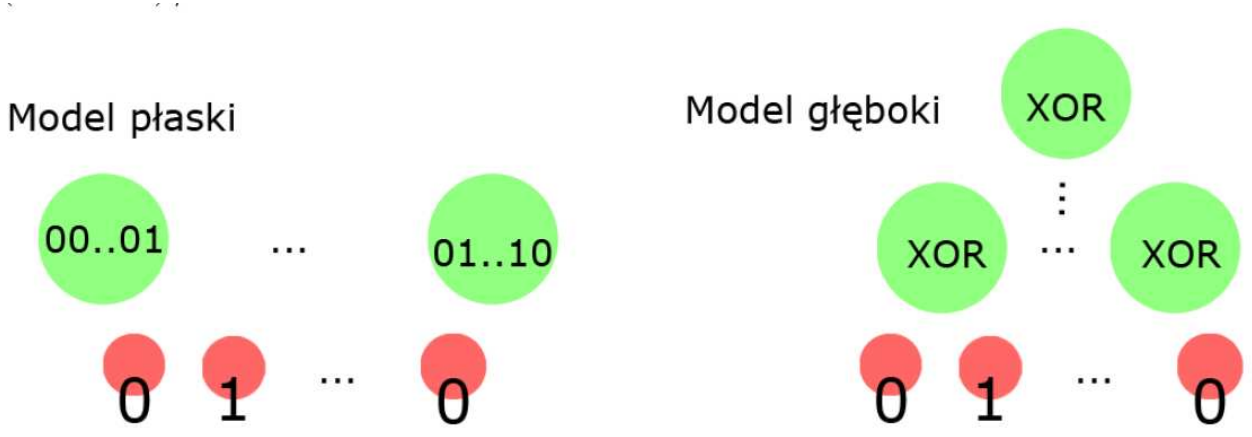
Uczenie a uogólnianie

Metody poprawy uogólniania:

- dobór architektury sieci (liczba warstw, liczby neuronów w warstwach) do problemu,
- zatrzymywanie uczenia, gdy błąd walidacji (testu) zaczyna rosnąć (ang. *early stopping*),
- regularyzacja:
  - dodanie szumu do danych podczas uczenia,
  - dodanie specjalnego członu do funkcji błędu: $J' = J + \frac{\lambda}{2K} \sum w^2$ 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 (teoretycznie wystarczających do reprezentowania dowolnych funkcji).

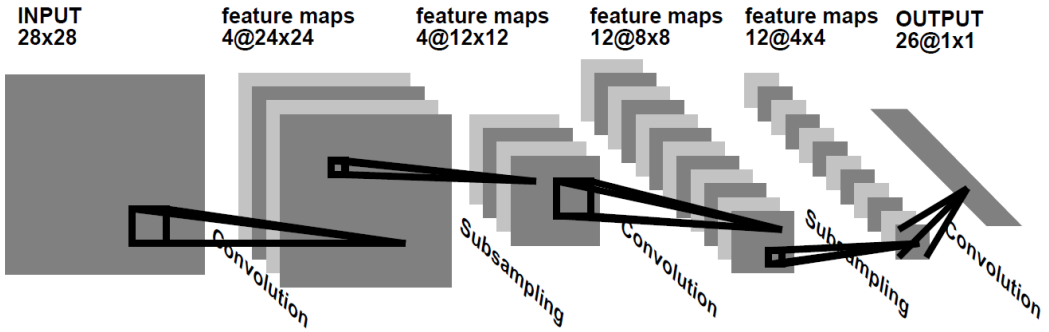


Głębokie uczenie (*Deep Learning - DN*)

Główne kierunki rozwoju:

- głębokie sieci przekonań (*Deep Belief Networks - DBN*),
- sieci realizujące funkcje splotu (*Convolutional Neural Networks - CNN*),
- sieci rekurencyjne zawierające warstwy GRNN lub LSTM do analizy sekwencji,
- sieci zawierające warstwy kapsułowe (*Capsules*),
- biblioteki nauczonych sieci do wykorzystania w podobnych problemach klasyfikacyjnych do douczenia z użyciem niewielkiego zbioru przykładów uczących(*transfer learninig*).

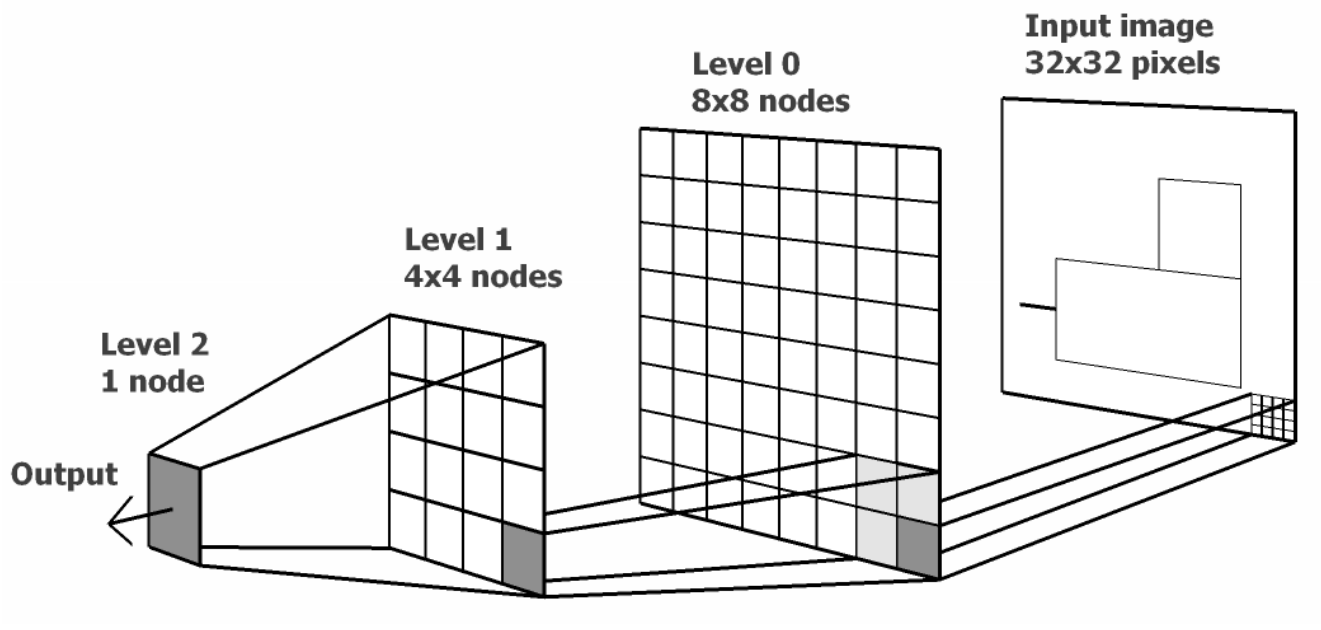
Sieć realizująca funkcję splotu



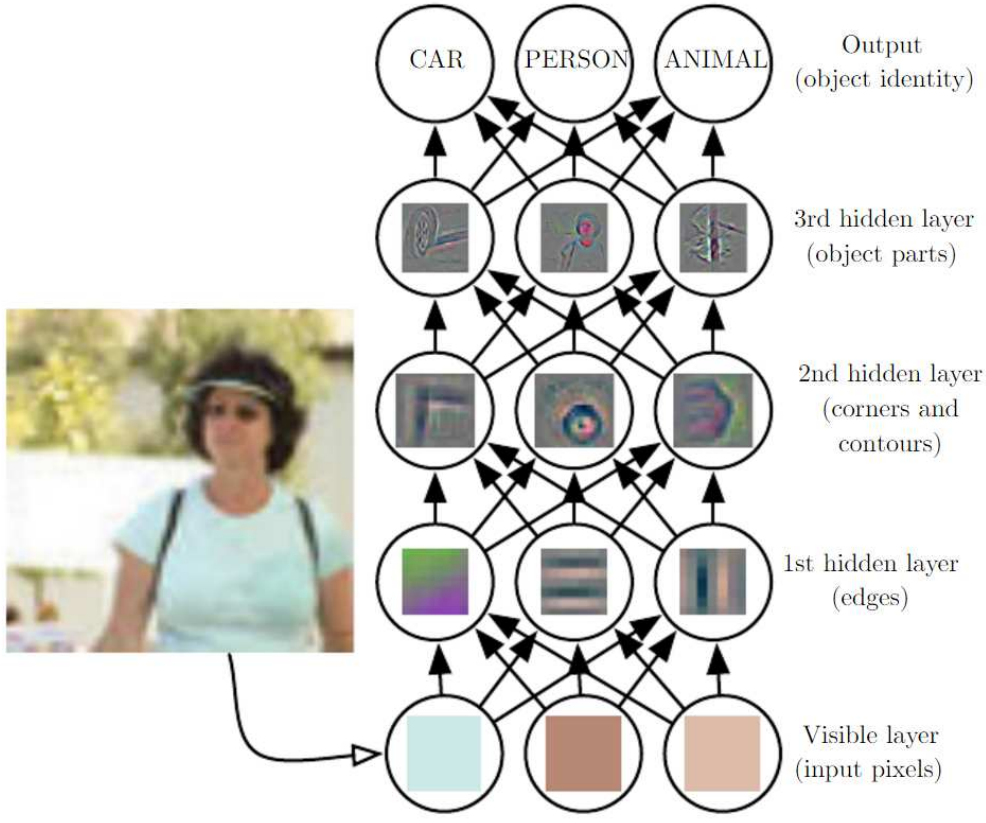
Techniki poprawiające szybkość uczenia i uogólnianie:

- 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 skrośnej (*cross entropy*) $J = -\frac{1}{L}\sum_{i=1}^{M}(d_i \log y_i + (1 - d_i)\log(1 - y_i))$ jako funkcji błędu zamiast średniego błędu kwadratowego,
- warstwa wyjściowa *softmax* pozwala na uzyskanie ciągu prawdopodopieństw przypisania wektora obserwacji do poszczególnych klas.

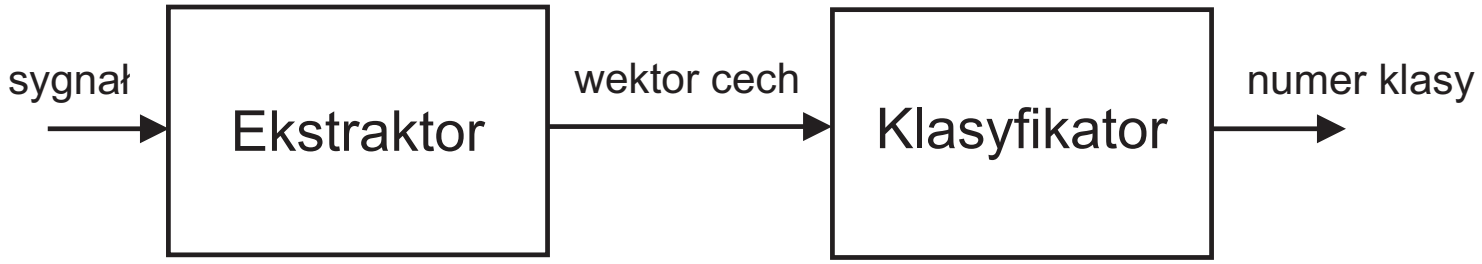
Funkcja splotu i podpróbkowania



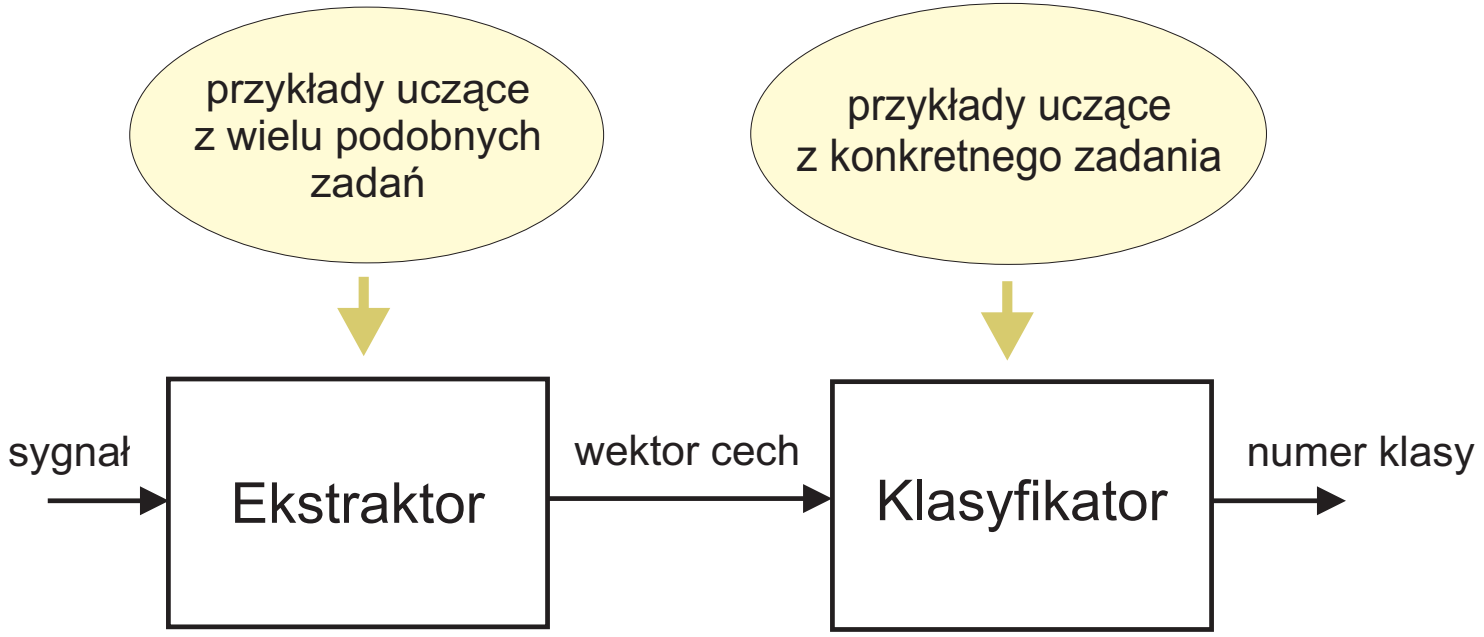
Sieć realizująca funkcję splotu - hierarchia przekształceń



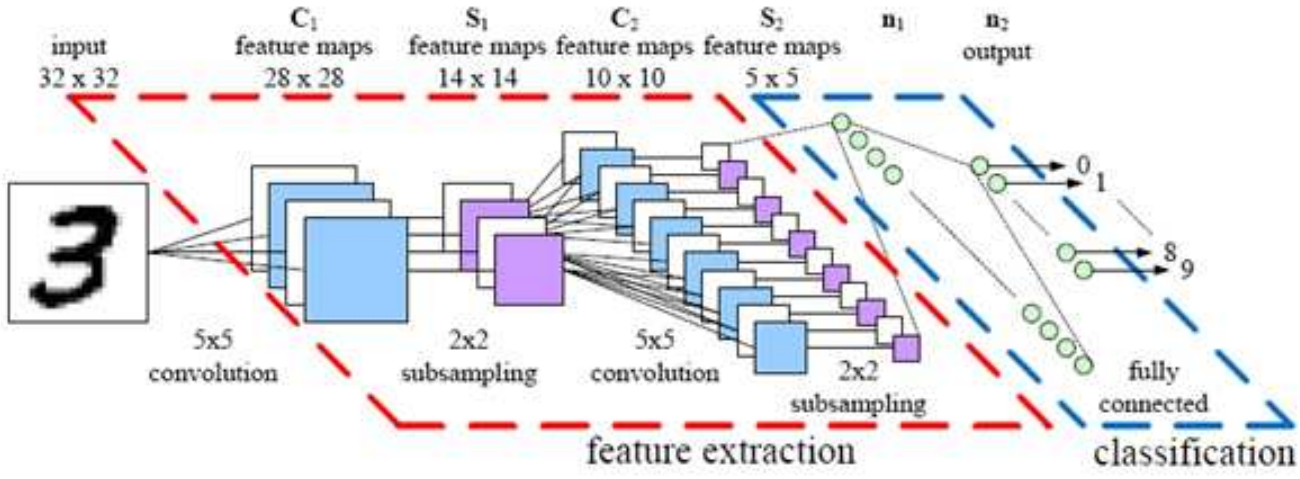
Schemat procesu rozpoznawania



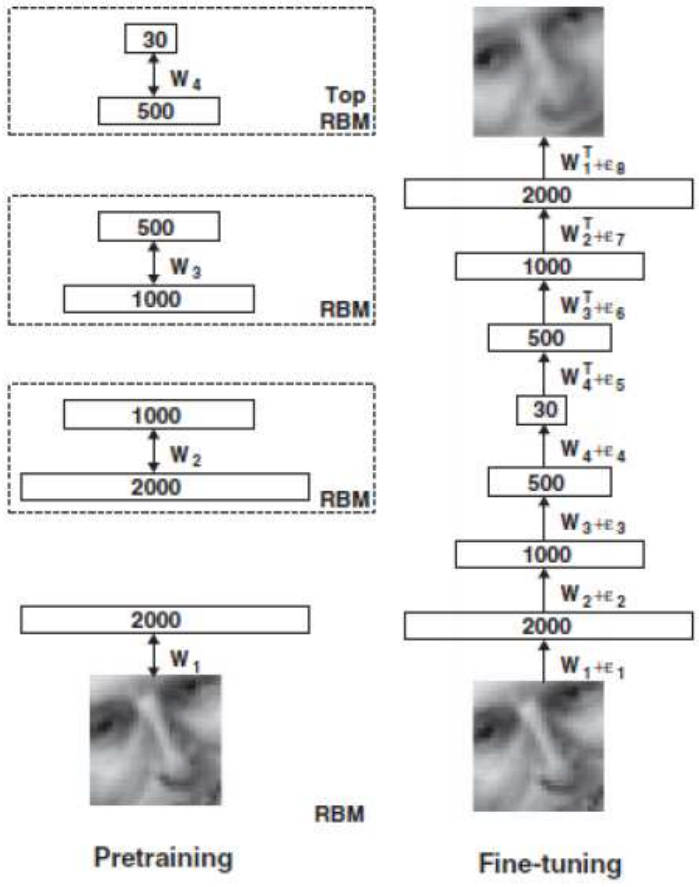
Schemat procesu rozpoznawania - dane do uczenia



Sieć realizująca funkcję splotu - ekstrakcja cech



Sieć autoasocjacyjna - ekstrakcja cech



Algorytm AdaBoost

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

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

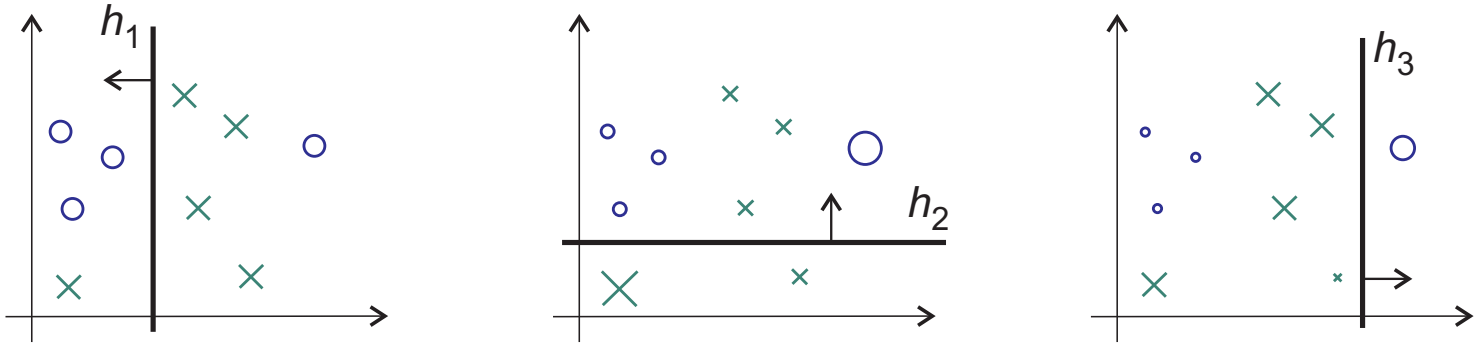
while błąd fałszywie pozytywnych $f > f_{\max}$ **or** proporcja prawdziwie pozytywnych $d < d_{\min}$

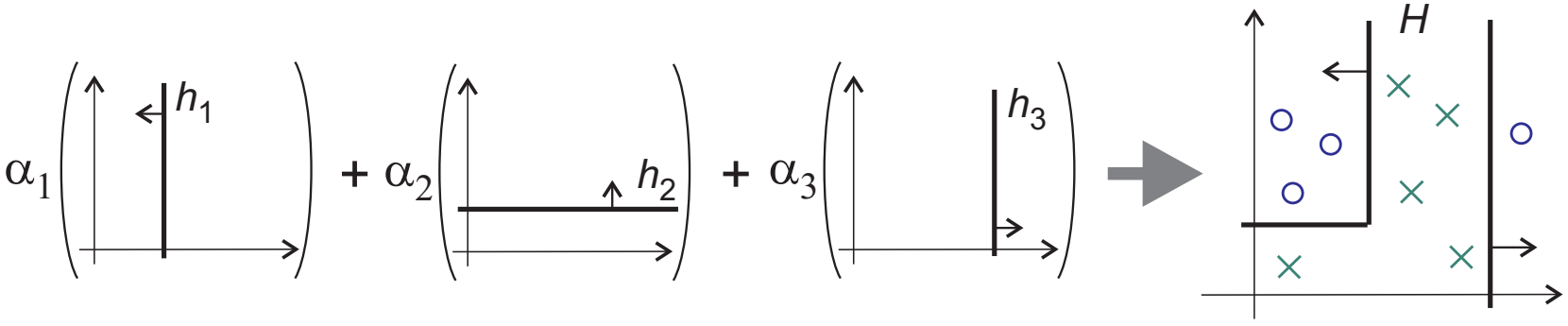
1. normalizuj wagi: $w_i \leftarrow \frac{w_i}{\sum_{j=1}^{K} w_j}$
2. wybierz słaby klasyfikator $h_t(\mathbf{x})$, który minimalizuje ważony błąd klasyfikacji: $\epsilon_t = \min_j \sum_{i=1}^{K} w_i |h_j(\mathbf{x}_i) - c_i|$
3. zmniejsz wagi poprawnie klasyfikowanych przykładów: $w_i \leftarrow w_i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

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, θ - wartość progu, p - polaryzacja





Algorytm AdaBoost - wzór na silny klasyfikator

Silny klasyfikator zawierający T słabych klasyfikatorów:

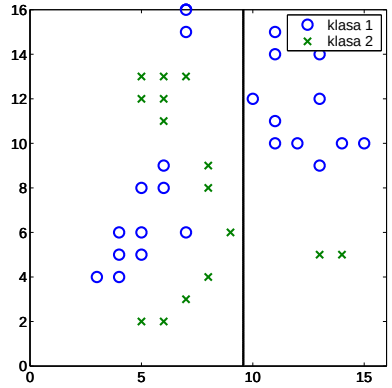
$$H(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \geq \Theta \\ \\ 0 & \text{w przeciwnym razie,} \end{cases}$$

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,
Θ – wartość progu.

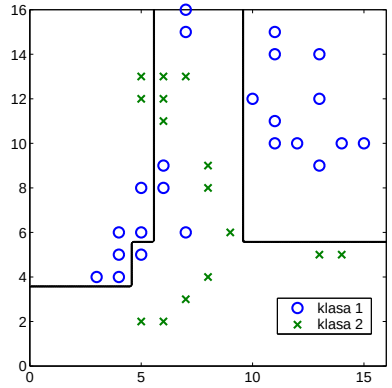
Algorytm AdaBoost - granice decyzji

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.

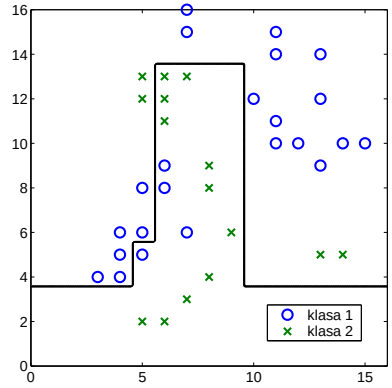
a)



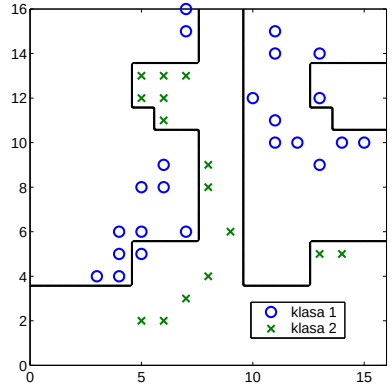
b)



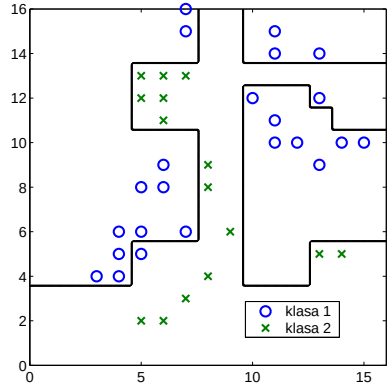
c)



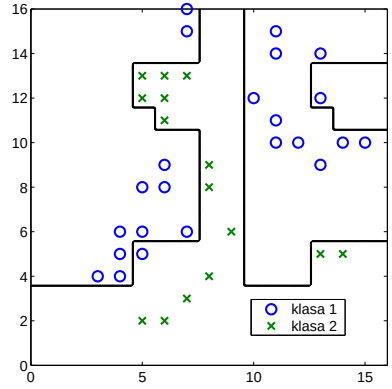
d)



e)



f)



Algorytm AdaBoost w detekcji twarzy

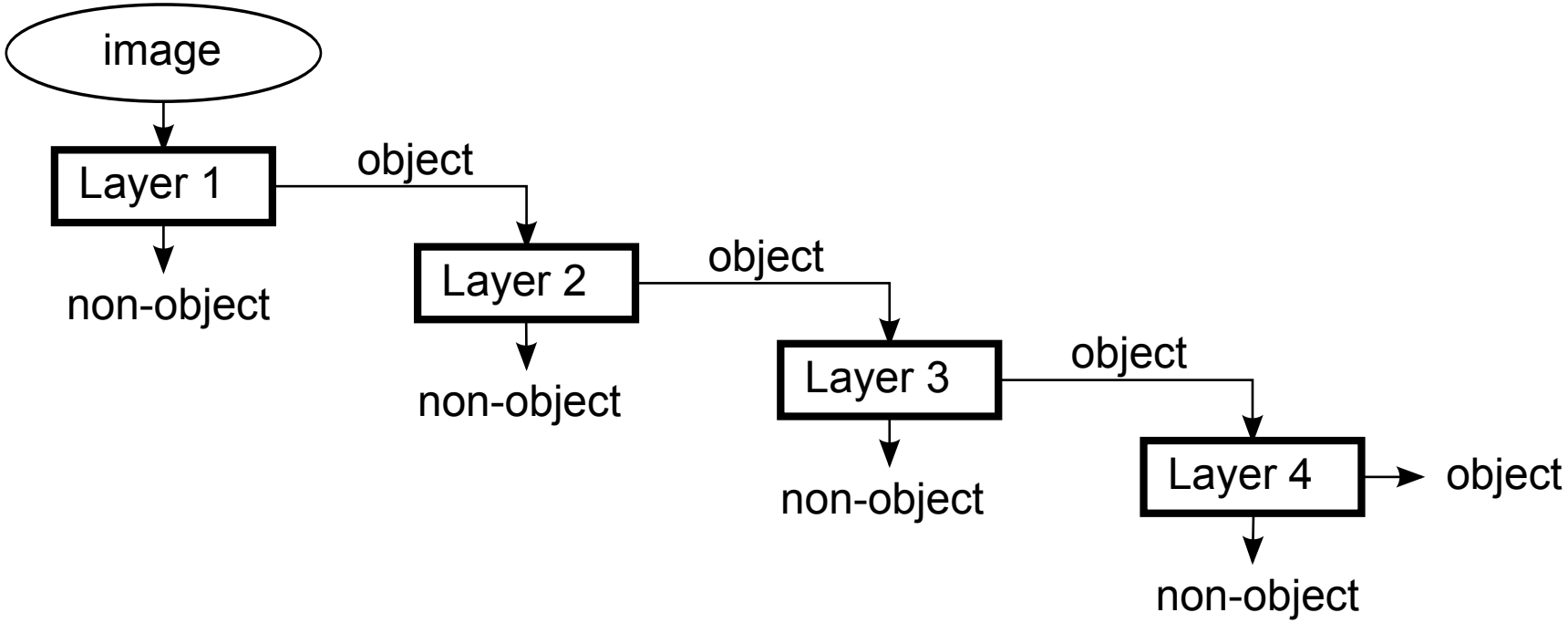
Wymagania:

- znormalizowane położenie twarzy w przykładach uczących,
- znormalizowane średnia i wariancja jasności pikseli,
- kaskadowa wersja klasyfikatora AdaBoost (przewaga obrazów negatywnych).

Cechy obrazów:

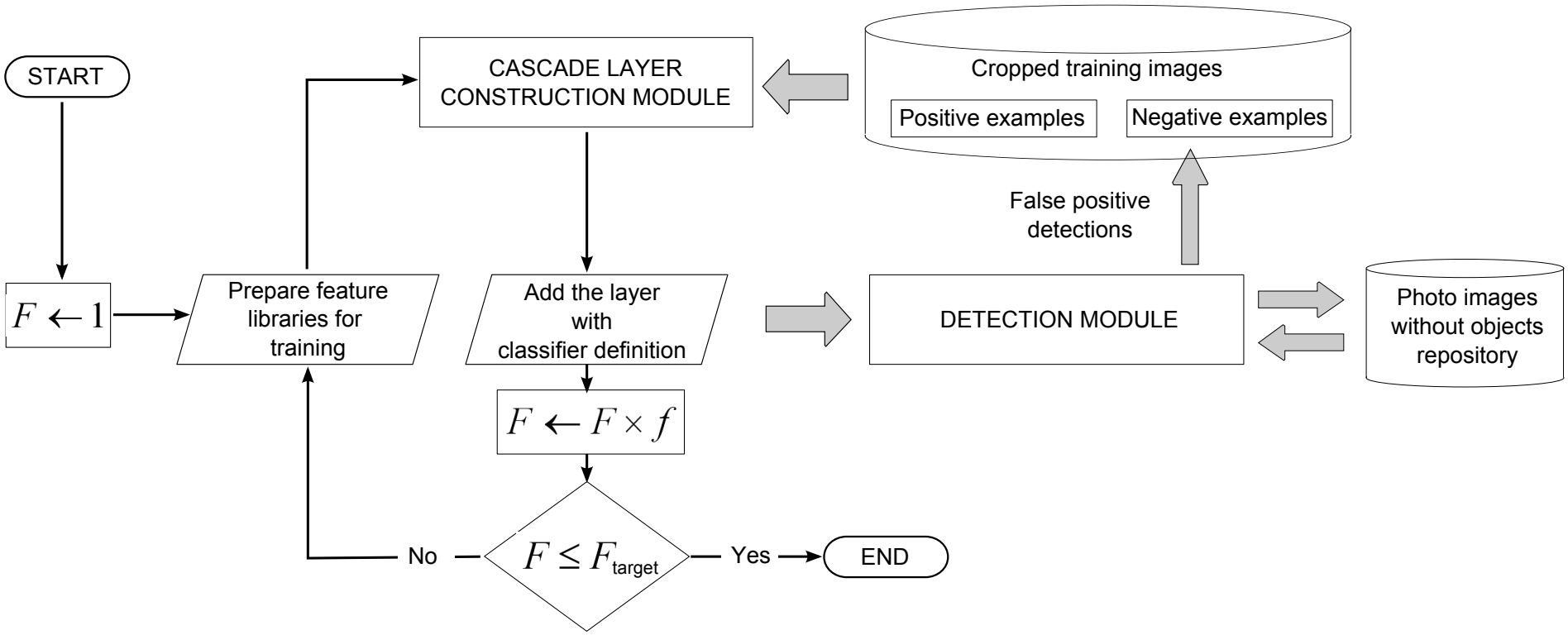
- różnice średnich jasności pikseli w obszarach prostokątowych (HAAR Features),
- lokalne cechy binarne (*CENSUS*, *Local Binary Features* (LBF)),
- histogram kierunków gradientów (*Histogram of Oriented Gradients* (HOG)),
- zgrupowane binarne cechy prostokątowe, Lokalne zgrupowane cechy binarne (LAB).

Kaskada silnych klasyfikatorów

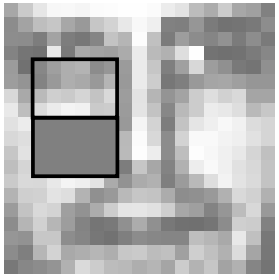
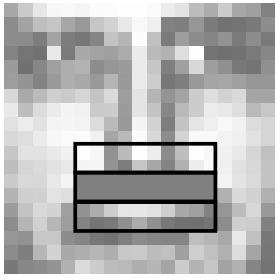
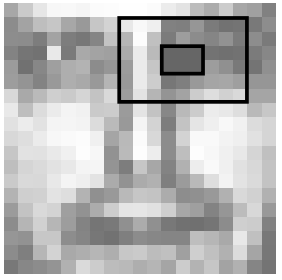
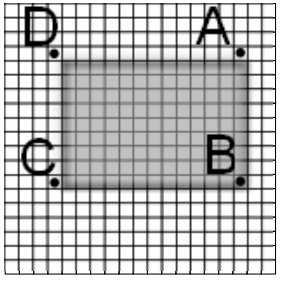


Zaleta kaskady - odrzucanie dużej liczby obrazów negatywnych małym kosztem obliczeniowym w przypadku dużej przewagi obrazów negatywnych podczas skanowania zdjęcia lub obrazu z kamery

Schemat uczenia kaskady silnych klasyfikatorów w zadaniu detekcji obiektów



Typy cech prostokątowych (Haar features) w detekcji twarzy

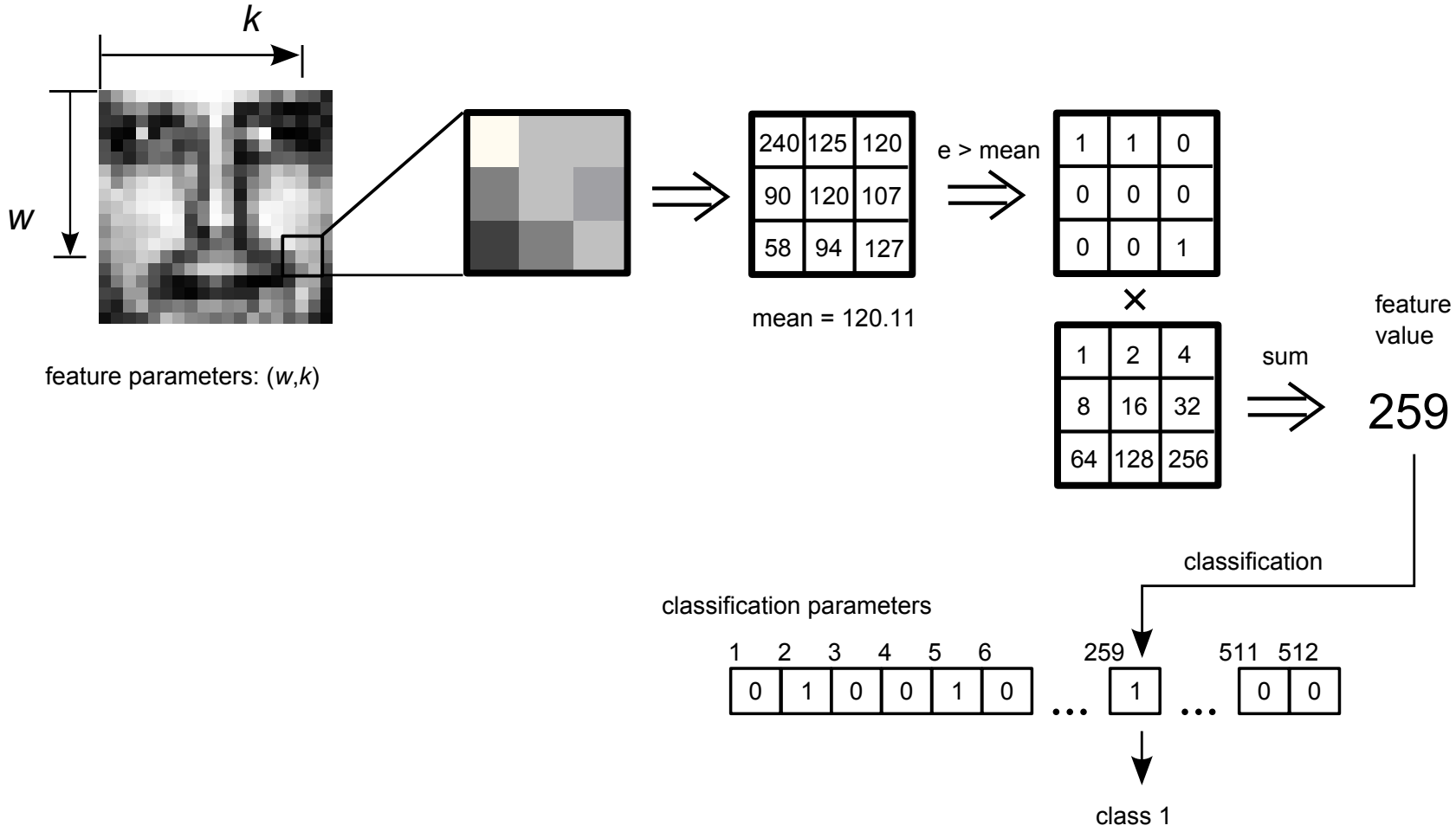
Użycie obrazu całkowego: Suma jasności pikseli w prostokącie ABCD = B+D-A-C

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 (to nie było omawiane)

Przykład obliczania wartości cech zmodyfikowanego Census:



Zmodyfikowany Census (to nie było omawiane)

Uczenie słabego klasyfikatora:

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

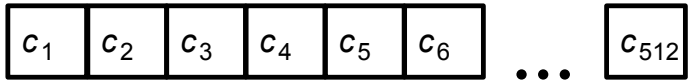
$\epsilon(\mathbf{x}) = \sum_{\gamma} \min\{g^0(\mathbf{x}, \gamma), g^1(\mathbf{x}, \gamma)\},$

gdzie

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

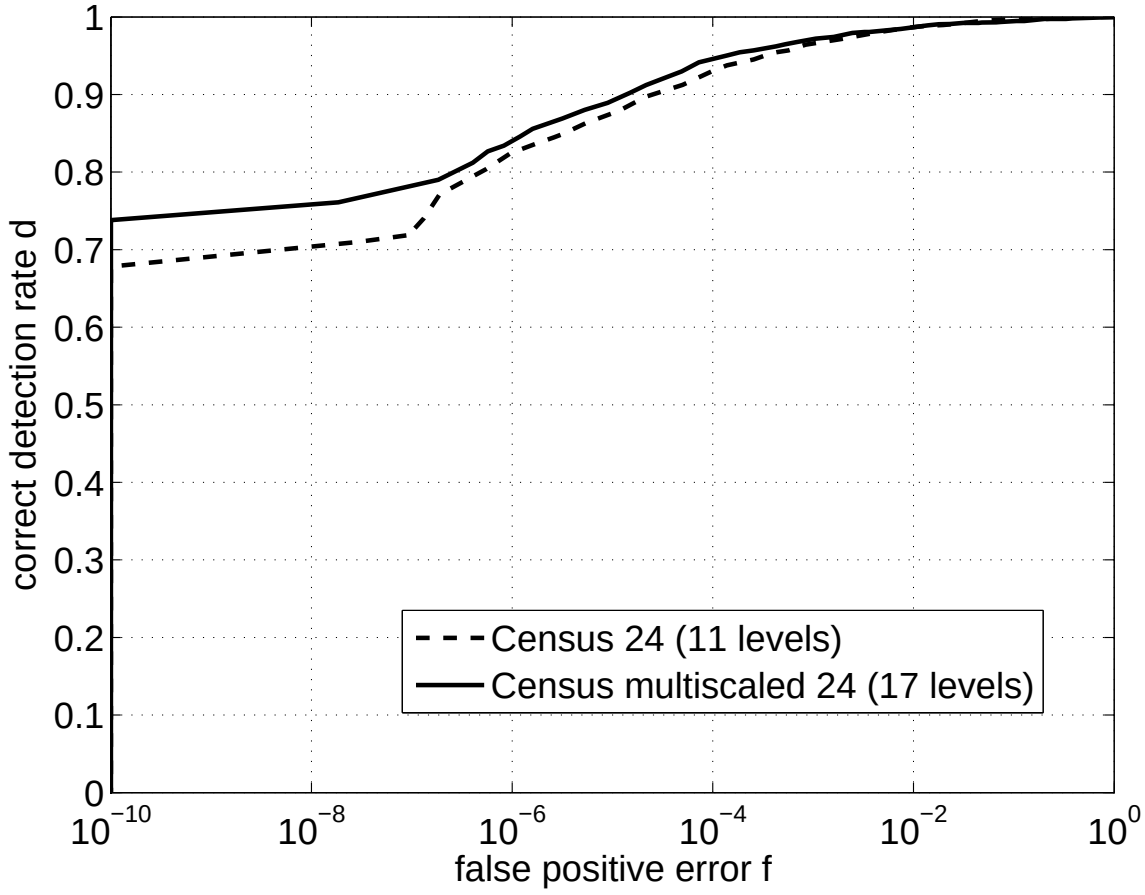
$$c_\gamma = \begin{cases} 0 & \text{jeśli} \quad g^0(\mathbf{x}_{\text{best}}, \gamma) > g^1(\mathbf{x}_{\text{best}}, \gamma) \\ 1 & \text{w przeciwnym wypadku.} \end{cases}$$



AdaBoost: zestawienie typów cech i słabych klasyfikatorów (to nie było omawiane)

typ cechy	typ klasyfikatora	parametry cechy $\mathbf{p}_f$	parametry klasyfikatora $\mathbf{p}_c$
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 (to nie było omawiane)



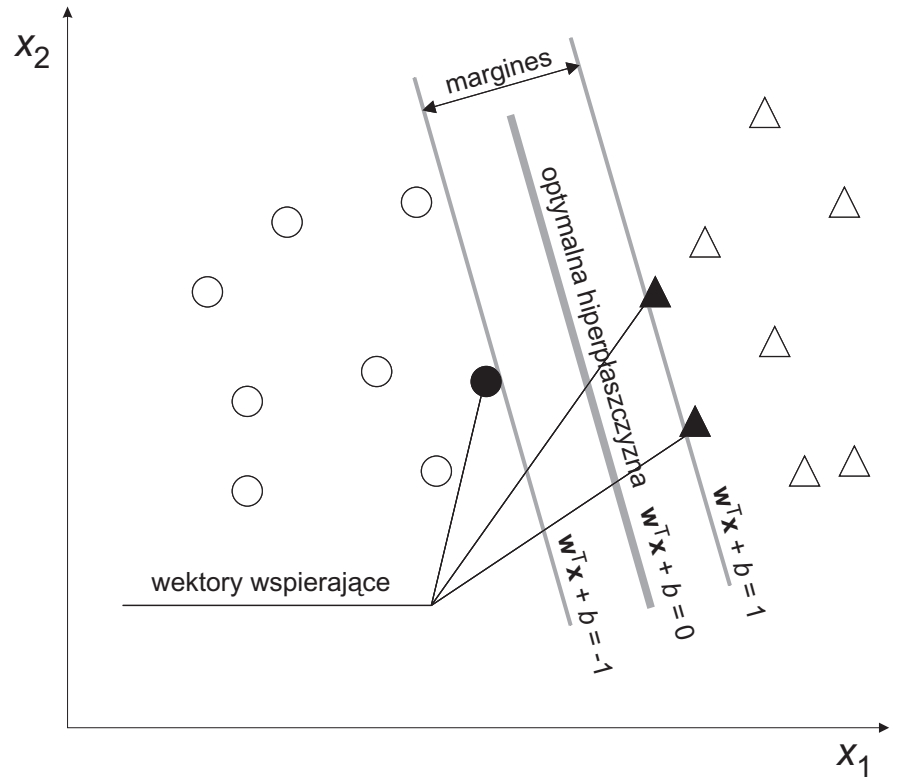
SVM: problem klasyfikacji

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



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

SVM: szukanie optymalnej hiperpłaszczyzny

maksymalizacja marginesu:

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

przy ograniczeniach:

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

SVM: szukanie optymalnej hiperpłaszczyzny

Metoda mnożników Lagrange'a:

1. definicja lagrangiana:
   $L = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{K} \alpha_i (d_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) - 1),$
2. tzw. dualny problem optymalizacyjny (szukanie punktu siodłowego):
   - minimalizacja L ze względu na $\mathbf{w}$ oraz b
   - maksymalizacja L względem $\alpha_i, \quad i = 1, \ldots, K$.

SVM: minimalizacja L ze względu na $\mathbf{w}$ oraz b

przyrównanie pochodnych do zera:

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

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 α

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:

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

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

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

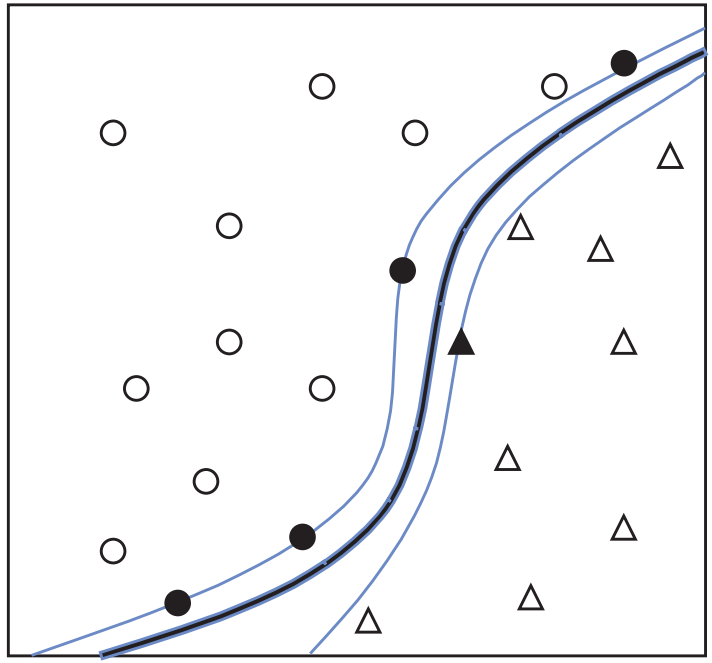
SVM: klasyfikacja

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

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

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
  $\alpha_i \leq C, \quad i = 1, \ldots, K$

SVM: zadania liniowo nieseparowalne

Metody rozwiązywania zadań nieseparowalnych:

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

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

SVM: zadania liniowo nieseparowalne

Metody rozwiązywania zadań nieseparowalnych:

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

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

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

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:

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

SVM: typy funkcji jądra

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

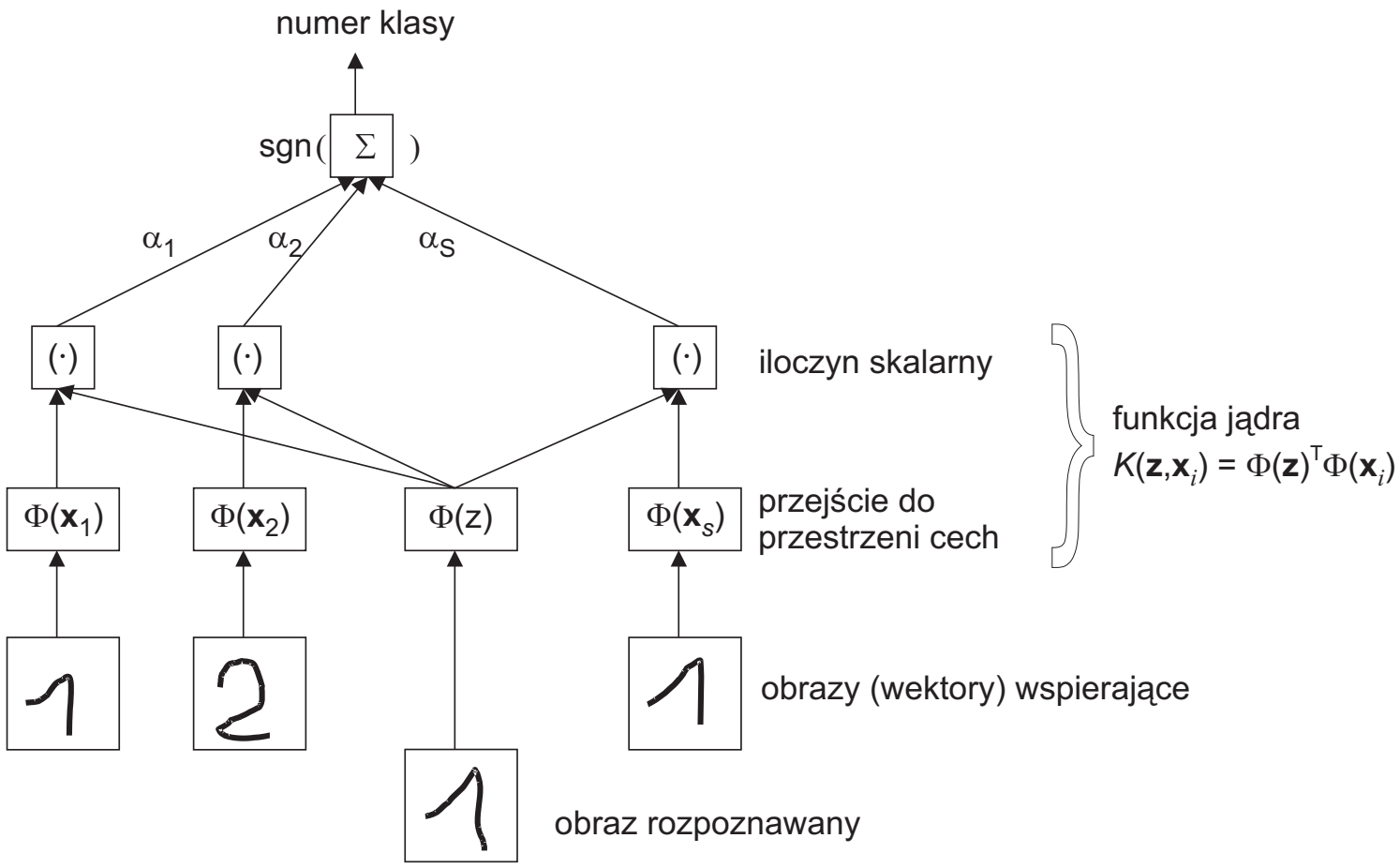
SVM: funkcja kwadratowa

Wyznaczenie nowych wektorów cech:

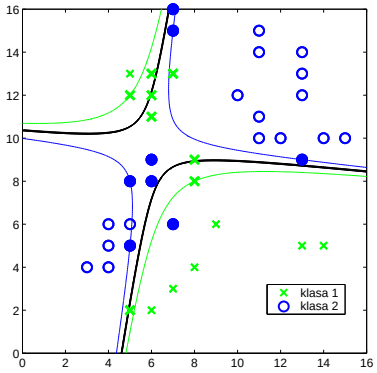
$$\begin{aligned}
\text{kernel}(\mathbf{u},\mathbf{v}) &= (\mathbf{u}^T\mathbf{v})^2 = ([u_1,u_2][v_1,v_2]^T)^2 = \\
&= u_1^2v_1^2 + 2u_1v_1u_2v_2 + u_2^2v_2^2 = \\
&= [u_1^2, \sqrt{2}u_1u_2, u_2^2][v_1^2, \sqrt{2}v_1v_2, v_2^2]^T = \\
&= \Phi(\mathbf{u})^T\Phi(\mathbf{v}).
\end{aligned}$$

W praktyce nie ma potrzeby wyznaczania wektorów Φ, które mogą mieć bardzo dużą wymiarowość.

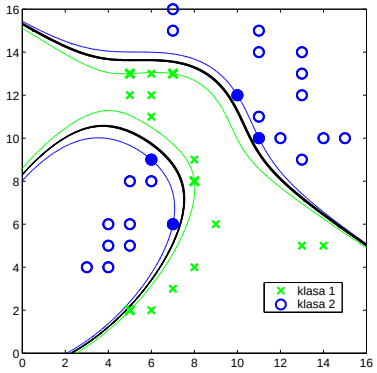
SVM: schemat klasyfikacji



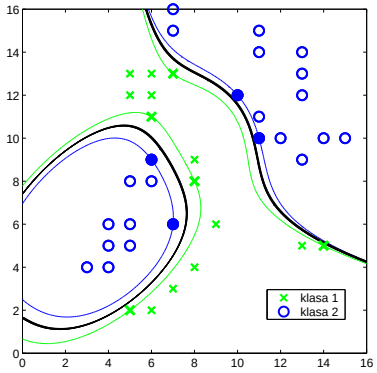
a)



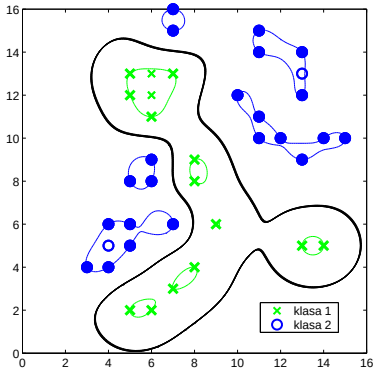
b)



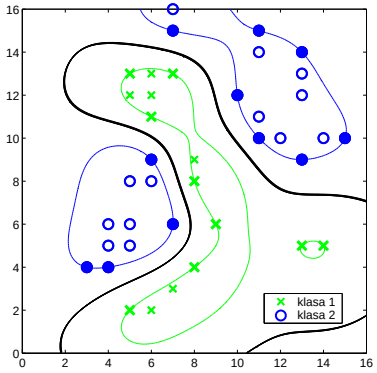
c)



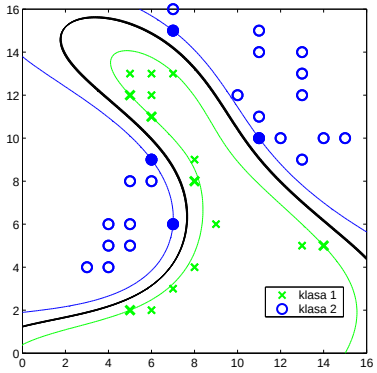
d)



e)



f)



Granice decyzji dla przykładowego zadania klasyfikacji wektorów obserwacji o $N = 2$ *punkty-1* dla klasyfikata 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.