Podstawy sztucznej inteligencji

wykład 5

Sztuczne sieci neuronowe (SSN)

Joanna Kołodziejczyk

4 czerwca 2011

Plan wykładu

Neuron biologiczny

Neuron

Jest podstawowym budulcem układu nerwowego. Jest komórką, która jest w stanie odbierać i przekazywać sygnały elektryczne.

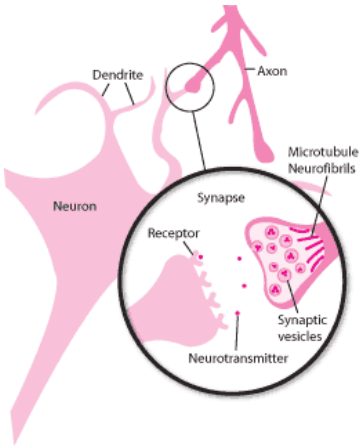
Neuron działanie

Jeżeli wartość sygnału elektrycznego przekazywana do neuronu przekracza pewną wartość, neuron ulega depolaryzacji, powodując pobudzenie komórki nerwowej. Pobudzenie polega na wyładowaniu neuronu, a powstały sygnał przesyłany jest do innych neuronów. Siła wyładowania neuronu jest niezależna od bieżącej depolaryzacji, siły która go spowodowała. Dopóki prąd przekracza określony próg, wielkość wyładowanie pozostanie taka sama (w danym neuronie).

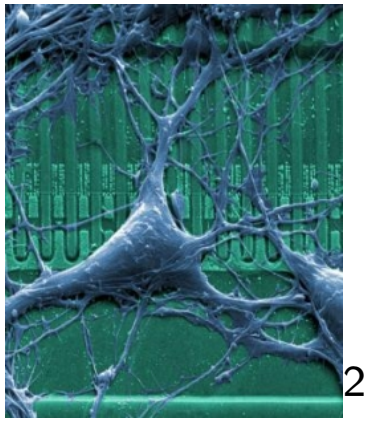
Budowa neuronu biologicznego

Neuron składa się z:

1. Wielu dendrytów, których celem jest pobieranie impulsów od innych neuronów.
2. Ciała komórki z jądrem.
3. Jednego aksonu, który przekazuje wyładowanie do kolejnych komórek.
4. Synaps – neuroprzekaźników osłabiających lub wzmacniających sygnał wyjściowy



1



2

[1] http://www.smart-publications.com/articles/view/choline-uridine-builds-new-neurons-and-protect-against-Alzheimers/

Plan wykładu

Sztuczna sieć neuronowa

Sztuczna sieć neuronowa

Sztuczne sieci neuronowe (SSN) łączą w sobie małe procesory (jednostki przetwarzania), a każdy naśladuje funkcje pojedynczego neuronu biologicznego. W ten sposób tworząc sztuczny „mózg" do przetwarzania informacji. Dzięki połączeniu wielu takich sztucznych neuronów, z których każdy posiada jedno wyjście, SSN może wykonywać złożone funkcje.

Sztuczna neuron

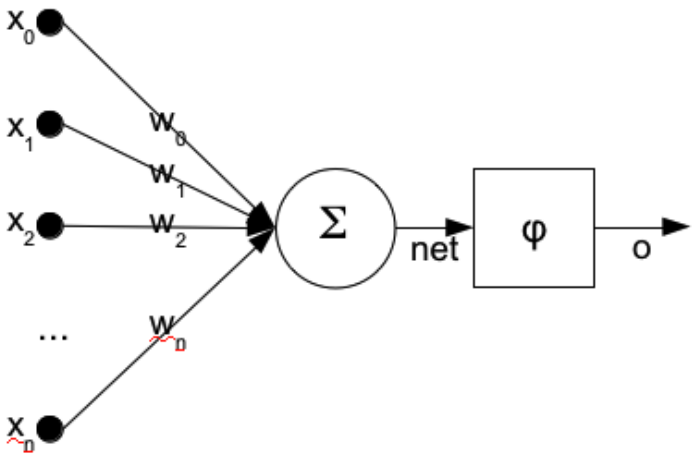
Każdy procesor, podobnie jak neuron, ma poziom progowy, który musi być przekroczony nim nastąpi transmisja sygnału. Binarny charakter procesora można porównać do włącznika światła.

Sztuczny Neuron

Działanie neuronu

$$o = \varphi(net) = \varphi(\sum_{i=0}^{n} x_i * w_i)$$

gdzie x_i — wejścia i $x_0 = 1$, w_i — wagi, φ — funkcja aktywacji, net — pobudzenie neuronu, o — wyjście z neuronu



Cechy neuronu

- Wejścia i wagi są liczbami rzeczywistymi dodatnimi i ujemnymi.
- Jeżeli jakaś cecha (wejście) powoduje odpalenie neuronu, waga będzie dodatnia, a jeżeli cecha ma działanie hamujące to waga jest ujemna.
- Funkcję aktywacji dobiera się do rozwiązywanego zadania.
- Wagi są dopasowywane przez pewną regułę uczenia tak, by zależność wejście/wyjście w neuronie spełniało pewien określony cel.

Stosowane w SSN funkcje aktywacji

Name	Input/Output Relation	Icon
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$	
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$	
Linear	$a = n$	
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$	

[3]

[3] Neural Network Design, M. T. Hagan, H. B. Demuth, M. H. Beale

Stosowane w SSN funkcje aktywacji

Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \le n \le 1$ $a = 1 \quad n > 1$	
Log-Sigmoid	$a = \frac{1}{1+e^{-n}}$	
Hyperbolic Tangent Sigmoid	$a = \frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$	
Positive Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \le n$	
Competitive	$a = 1$ neuron with max n $a = 0$ all other neurons	

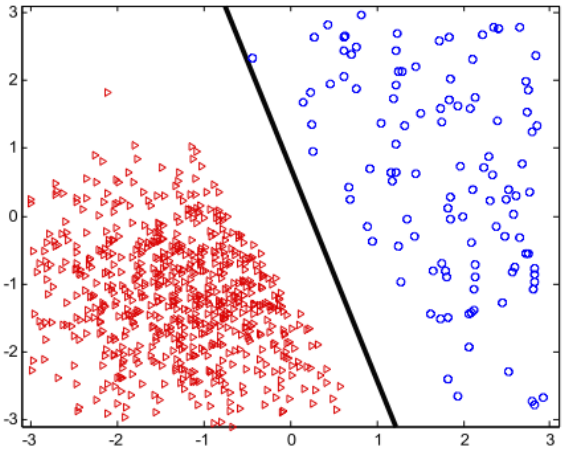
[4]

[4] Neural Network Design, M. T. Hagan, H. B. Demuth, M. H. Beale

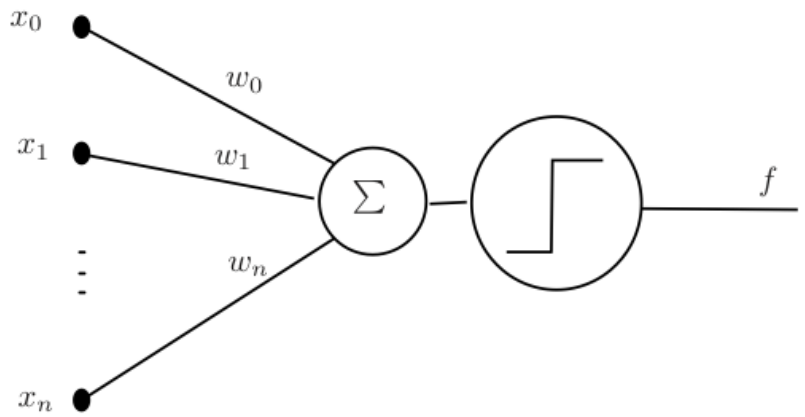
Plan wykładu

Perceptron prosty (Rosenblatt 1962)

Perceptron prosty jest najprostszą SSN, używaną do klasyfikacji binarnej. Perceptron składa się z pojedynczego neuronu z regulacją wag. Rosenblatt zaproponowała twierdzenie zbieżności perceptronu, stwierdzając, że konwergencja jest gwarantowana wtedy i tylko wtedy, gdy wzorce wykorzystywane do uczenia perceptronu tworzą dwie separowalne klasy. Powierzchnia decyzyjna zostanie umieszczona w formie hiperpłaszczyzny gdzieś pomiędzy tymi dwiema grupami.



Perceptron — model



Funkcja aktywacji

$$f(\langle \mathbf{w}, \mathbf{x} \rangle) = \begin{cases} 1 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle > \mathbf{0}; \\ 0 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle \leq \mathbf{0}. \end{cases}$$

Zadanie klasyfikacji binarnej

Klasyfikacja binarna

Klasyfikacja polega na przydzieleniu obiektu do pewnej klasy na podstawie jego atrybutów (danych wejściowych). W klasyfikacji binarnej wyjście jest dwupunktowe (możliwe dwie klasy na wyjściu).

Niech dany będzie zbiór uczący (zbiór par) $(\mathbf{x}_i, y_i)$ $i = 1, \ldots, l$, gdzie $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \in \mathbb{R}^n$ są danymi wejściowymi a $y_i \in \{0, 1\}$ są skojarzonymi z nimi klasami (wyjściami).

Płaszczyzna separacji klas

Równanie płaszczyzny w przestrzeni $\mathbb{R}^n$

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = 0,$$

czyli, należy znaleźć taki zestaw $w_0, w_1, w_2, \ldots, w_n$, że

$$\forall_{i, y_i = 1} w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0$$

oraz

$$\forall_{i, y_i = 0} w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \leq 0,$$

gdy dodamy jeszcze jeden punkt do wejść $x_i = (1, x_{i1}, x_{i2}, \ldots, x_{in})$, to równanie płaszczyzny:

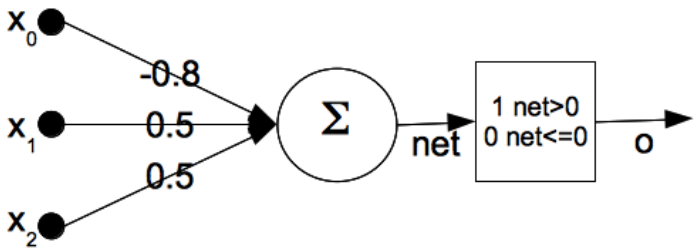
$$\langle \mathbf{w}, \mathbf{x_i} \rangle = \sum_{j=0}^{n} w_j x_{ij}$$

Przykład dwuwymiarowy

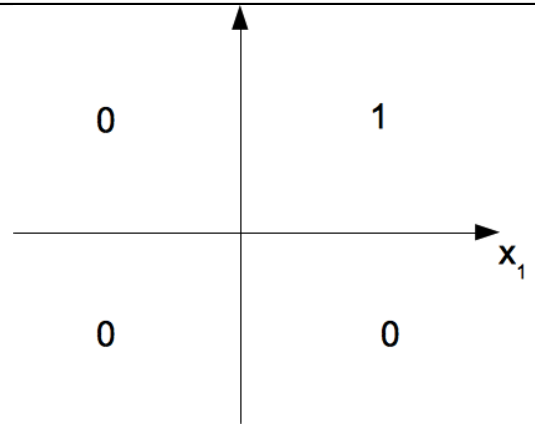
Dane uczące:

x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	0

Nauczony neuron (ustalone wagi):



Przestrzeń wejść:



Uczenie SSN

Uczenie

Uczenie polega na automatycznym doborze wag w SSN, na podstawie zbioru przykładów nazwanych zbiorem uczącym. Zaczyna się z losowymi małymi wagami i iteracyjnie zmienia się wagi, dopóki wszystkie przykłady uczące nie zostaną poprawnie zaklasyfikowane (lub z niewielkim błędem)

Wyróżnia się dwa typy uczenia, zależy od wykorzystanej sieci i przykładów (zbioru uczącego):

1. nadzorowane (z nauczycielem), gdy w zbiorze danych do każdej próbki podana jest poprawna klasa
2. nienadzorowane (bez nauczyciela), zbiór danych nie ma wektora odpowiedzi.

Algorytm uczenia perceptronu

1. Niech $\mathbf{w}(0) = (0, \ldots, 0)$ lub wartości losowe z przedziału $[-1, 1]$, $k = 0$
2. Dopóki zbiór punktów uczących pozostaje błędnie klasyfikowany tj. zbiór $A = \{\mathbf{x}_i : y_i \neq f(\langle \mathbf{w}, \mathbf{x}_i \rangle)\}$ pozostaje niepusty, powtarzaj:
   1. Wylosuj ze zbioru A dowolny punkt
   2. Aktualizuj wagi według następującej reguły:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta e \mathbf{x}_i$$

   3. $k = k + 1$

Gdzie η współczynnik uczenia $\eta \in (0, 1]$. A e jest wartością błędu popełnianego na prezentowanej próbce.

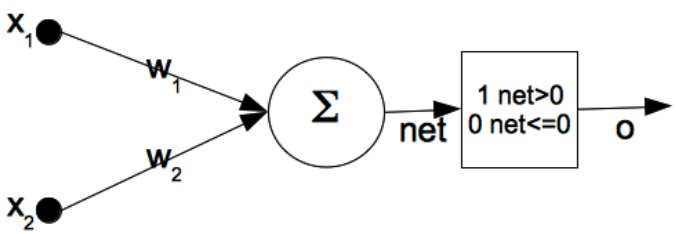
Analiza algorytmu na przykładzie testowym

Przykład pochodzi z książki „Neural Network Design", M. T. Hagan, H. B. Demuth, M. H. Beale.

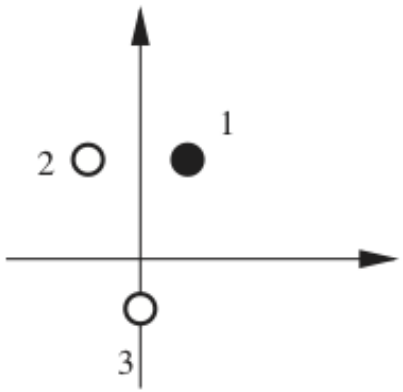
Zbiór uczący

$\{\mathbf{x}_1 = (1, 2), y_1 = 1\}$ $\{\mathbf{x}_2 = (-1, 2), y_2 = 0\}$
$\{\mathbf{x}_3 = (0, -1), y_3 = 0\}$

Architektura sieci:



Przestrzeń wejść:

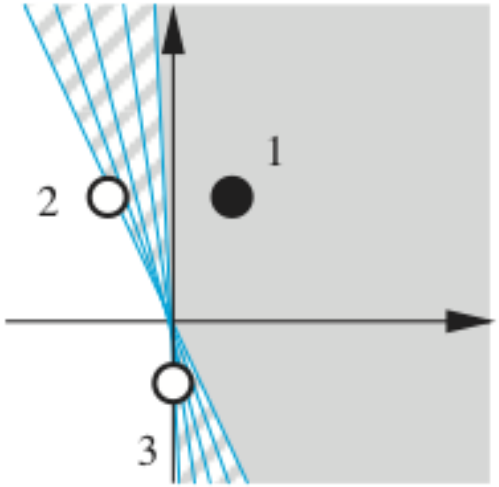


Konsekwencje braku wejścia x_0

Usunięcie dodatkowego wejścia powoduje, że płaszczyzna decyzyjna musi przejść przez początek układu współrzędnych, bo:

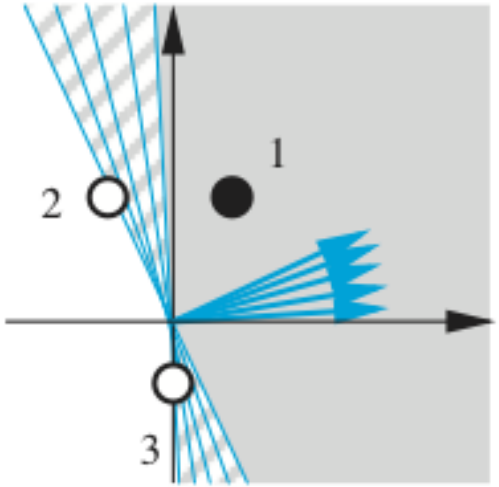
$$w_1 x_1 + w_2 x_2 = 0$$

to równanie prostej przechodzącej przez punkt $(0, 0)$.
Trzeba być pewnym, że taka sieć rozwiąże problem separacji punktu x_1 od punktów x_2 i x_3.
Jak widać rozwiązań jest nieskończenie wiele.



Cel uczenia

Rysunek pokazuje wektory wag, które odpowiadają granicom decyzyjnym. Wektor jest prostopadły do granicy decyzyjnej. Reguła ucząca ma znaleźć wektor wag, który wskazuje w jednym ze wskazanych kierunków. Długość wektora wag nie ma znaczenia, ważny jest tylko kierunek.

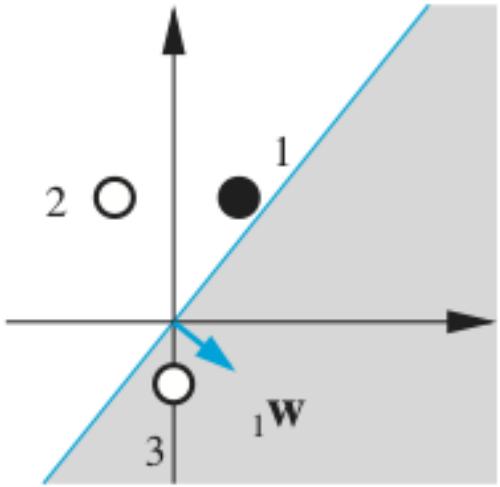


Ustalenie wartości początkowych wag

Zakłada się losowo ustalenie wartości wag początkowych i tak:

$$\mathbf{w}(0) = (1, -0.8)$$

Na wykresie zaprezentowano początkowy wektor wag i granicę decyzyjną dla takich wag.



Prezentacja pierwszego punktu ze zbioru uczącego

Pierwszym prezentowanym punktem jest $\mathbf{x}_1$.

$$f(\langle \mathbf{w}, \mathbf{x}_i \rangle) = f(1 \cdot 1 + (-0.8) \cdot 2) = f(-0.6) = 0$$

Wiemy, że $y_1 = 1$, zatem sieć popełniła błąd i wskazała złą klasę dla punktu $\mathbf{x}_1$.
Jak widać na rysunku z poprzedniego slajdu granica decyzyjna jest nieprawidłowo położona w stosunku do punktu uczącego. Powinno się zatem przesunąć wektor wag nieznacznie w kierunku punktu $\mathbf{x}_1$. Można tego dokonać dodając do wektora wag wektor $\mathbf{x}_i$.

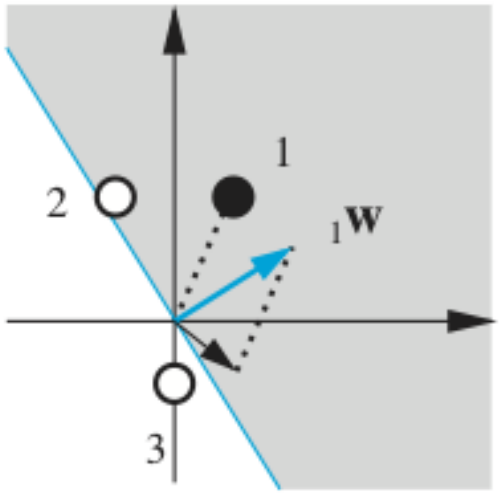
Wnioski co do poprawki wag

Reguła korekty wag (1)

Jeżeli $y_i = 1$ i $f(\langle \mathbf{w}, \mathbf{x}_i \rangle) = 0$ to

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{x}_i$$

Wagi po korekcie: $\mathbf{w}(1) = (1+1, -0.8+2) = (2, 1.2)$



Prezentacja drugiego punktu ze zbioru uczącego

Drugim prezentowanym punktem jest $\mathbf{x}_2$.

$$f(\langle \mathbf{w}, \mathbf{x}_i \rangle) = f(2 \cdot (-1) + 1.2 \cdot 2) = f(0.4) = 1$$

Wiemy, że $y_2 = 0$, zatem sieć popełniła błąd i wskazała złą klasę dla punktu $\mathbf{x}_2$.
Jak widać na rysunku z poprzedniego slajdu granica decyzyjna jest nieprawidłowo położona w stosunku do punktu uczącego. Powinno się zatem odsunąć wektor wag nieznacznie od punktu $\mathbf{x}_2$.
Można tego dokonać odejmując do wektora wag wektor $\mathbf{x}_i$.

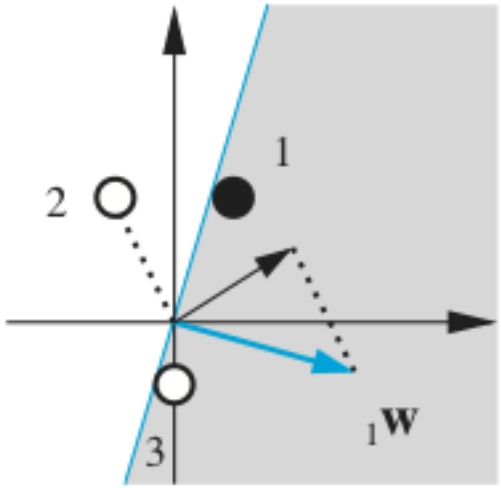
Wnioski co do poprawki wag

Reguła korekty wag (2)

Jeżeli $y_i = 0$ i $f(\langle \mathbf{w}, \mathbf{x}_i \rangle) = 1$ to

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mathbf{x}_i$$

Wagi po korekcie: $\mathbf{w}(2) = (2 - (-1), 1.2 - 2) = (3, -0.8)$

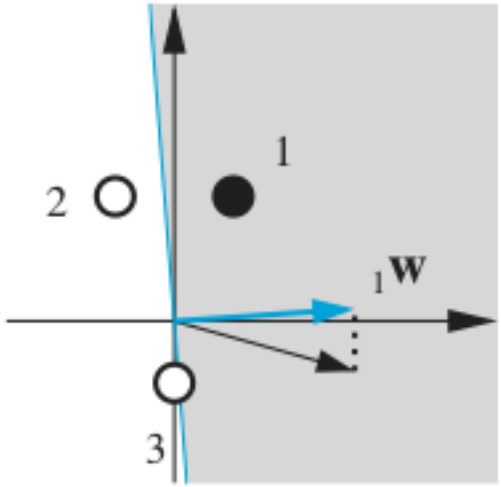


Prezentacja trzeciego punktu ze zbioru uczącego

Trzecim prezentowanym punktem jest $\mathbf{x}_3$.

$$f(\langle \mathbf{w}, \mathbf{x}_i \rangle) = f(3 \cdot 0 + (-0.8) \cdot (-1)) = f(0.8) = 1$$

Wiemy, że $y_2 = 0$, zatem sieć popełniła błąd i wskazała złą klasę dla punktu $\mathbf{x}_2$. Stosując regułę 2 poprawia się wagi:
$\mathbf{w}(3) = (3 - 0, (-0.8) - (-1)) = (3, 0.2)$



Kolejna seria prezentacji wzorców uczących

Prezentując ponownie wszystkie punkty uczące uzyskujemy bezbłędne sklasyfikowanie wszystkich 3 punktów. Oznacza to, że wagi $\mathbf{w} = (3, 0.2)$ są rozwiązaniem dla uczonego klasyfikatora.

Korekty wag

1. Jeżeli $y_i = 1$ i $f(\langle \mathbf{w}, \mathbf{x}_i \rangle) = 0$ to $\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{x}_i$
2. Jeżeli $y_i = 0$ i $f(\langle \mathbf{w}, \mathbf{x}_i \rangle) = 1$ to $\mathbf{w}(k+1) = \mathbf{w}(k) - \mathbf{x}_i$
3. Jeżeli $y_i = f(\langle \mathbf{w}, \mathbf{x}_i \rangle)$ to $\mathbf{w}(k+1) = \mathbf{w}(k)$

Reguła uczenia perceptronu

Potrójna reguła na korektę wag może być zapisana w postaci pojedynczego równania.
Niech błąd popełniany przez sieć neuronową:

$$e = y_i - f(\langle \mathbf{w}, \mathbf{x}_i \rangle)$$

1. Jeżeli $e = 1$ to $\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{x}_i$
2. Jeżeli $e = -1$ to $\mathbf{w}(k+1) = \mathbf{w}(k) - \mathbf{x}_i$
3. Jeżeli $e = 0$ to $\mathbf{w}(k+1) = \mathbf{w}(k)$

Korekta regułą perceptronu (delta)

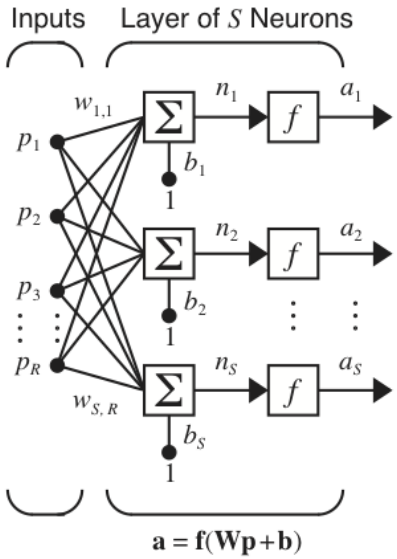
$$\mathbf{w}(k+1) = \mathbf{w}(k) + e\mathbf{x}_i$$

Plan wykładu

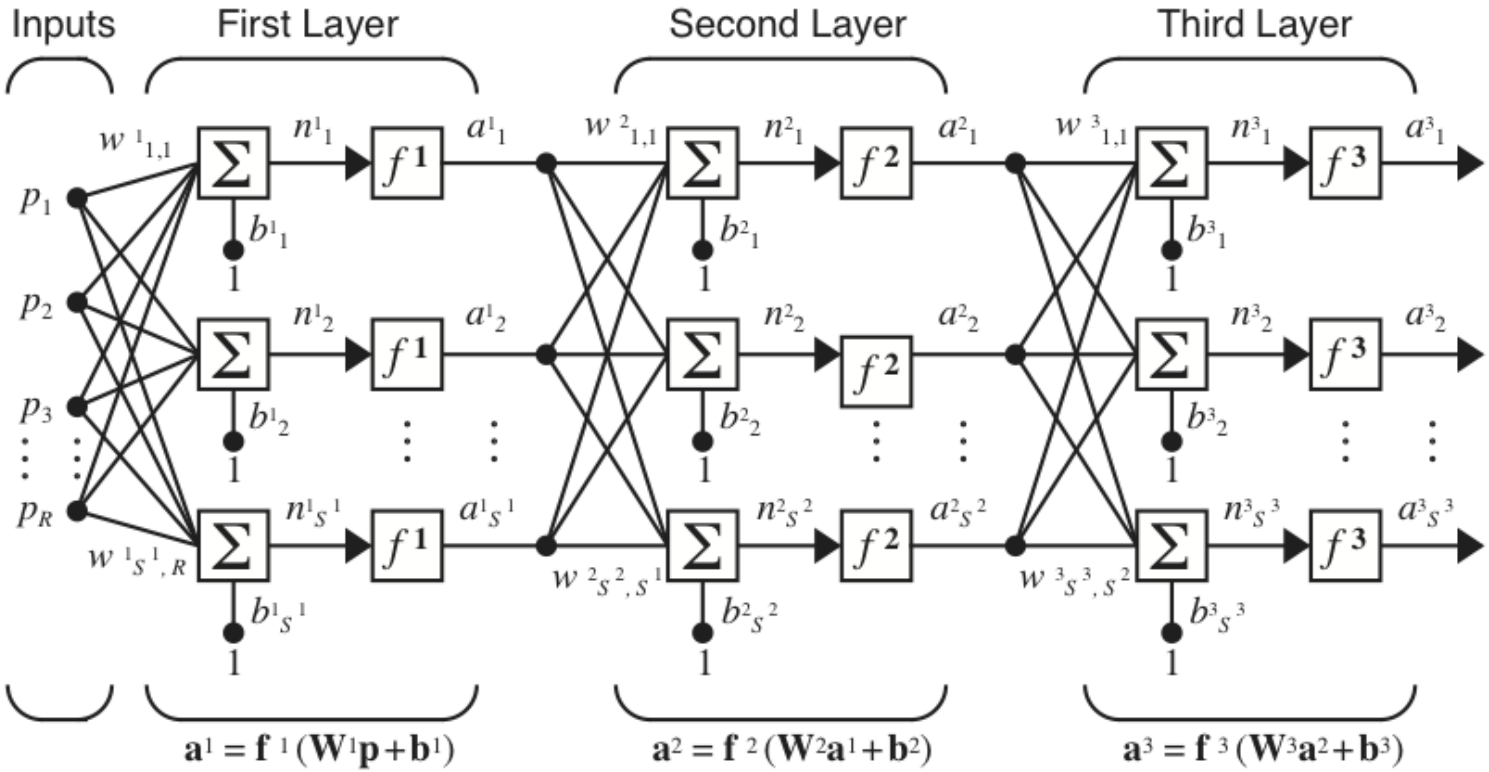
Perceptron prosty jednowarstowy

Pojedyncza warstwa neuronów (Single layer perceptron) - cechy

1. Pewna liczba neuronów S jest ułożona w warstwie i wszystkie wejścia R sieci zasilają wszystkie neurony. $S \neq R$.
2. Wyjście z sieci jest wektorem. Wagi są macierzą $S \times R$.
3. Umożliwia klasyfikację do większej liczby klas.
4. Zazwyczaj wszystkie neurony mają tą samą funkcję aktywacji.



Sieć wielowarstwowa MLP



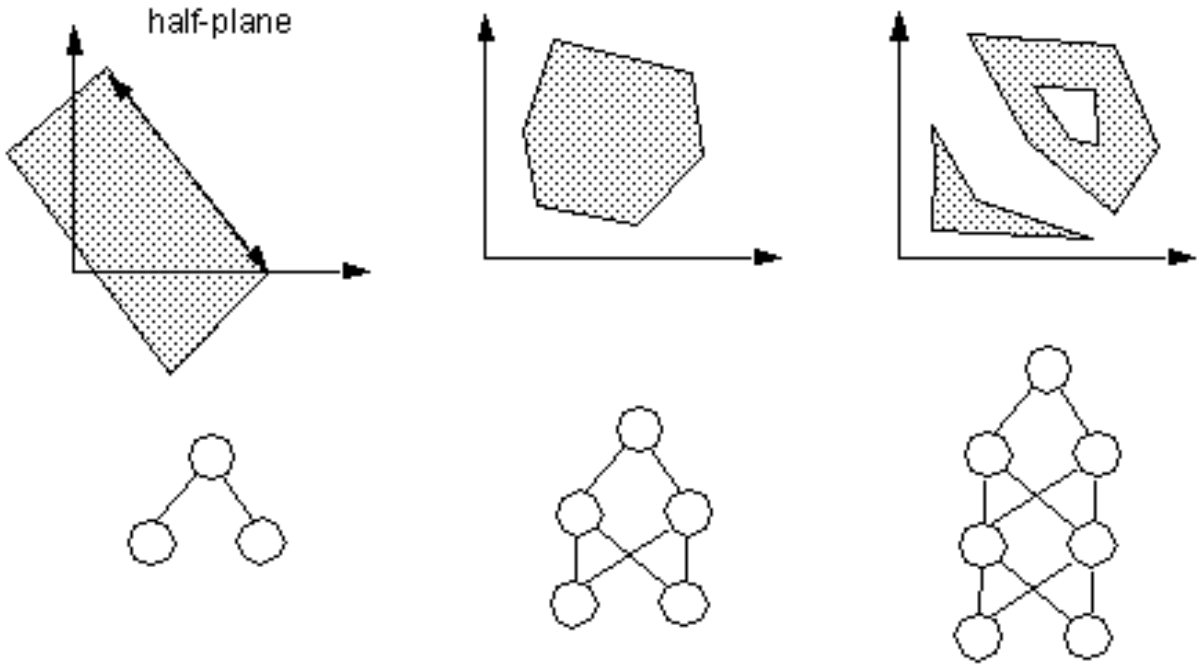
$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

Cechy sieci MLP

Wielowarstwowy perceptron

1. Ostatnia warstwa nazywana jest wyjściową, a wewnętrzne warstwami ukrytymi.
2. MLP ma dużo większe zastosowania niż pojedyncza warstwa.
3. Liczba wejść i neuronów wyjściowych jest określona przez rozwiązywane zadanie (zbiór danych wejściowych).
4. Liczba warstw ukrytych powinna być ograniczona do jednej lub dwóch. Wyjaśnienie na kolejnym slajdzie.
5. Nie ma wskazania ile powinno być neuronów w każdej warstwie. Źle dobrana ta liczba będzie skutkować złym dopasowaniem (nadmiernym lub za słabym) sieci do danych uczących.

Graficzna prezentacja możliwości separowania klas przez MLP



Dane uczące dla SSN

1. Należy zebrać dane niezbędne w procesie uczenia. Mogą pochodzić z baz danych.
2. Wybrać zmienne mające znaczenia dla rozwiązywanego zadania. Zazwyczaj na początku przyjmuje się wszystkie a potem dokonuje się redukcji.
3. Zebrana liczba rekordów powinna być reprezentatywna. Powinno się zadbać, by pokazać dane obejmujące pełne dziedziny zmiennych.
4. W rzeczywistości liczba potrzebnych rekordów jest również uzależniona od złożoności zależności funkcyjnej poddawanej modelowaniu.

Dane uczące dla SSN - obróbka wstępna

1. Dane numeryczne są przeskalowywane do właściwego dla sieci przedziału (normalizacja).
2. Wartości brakujące są zastępowane wartościami średnimi (lub innymi statystykami) obliczanymi na podstawie wartości zmiennych dostępnych w ciągu uczącym.
3. Typ danych nominalnych takich jak Płeć = Mężczyzna, Kobieta zamienia się na wartości numeryczne. Przy dużej liczbie klas łączy się je w grupy.
4. Wartości ciągłe poddaje się procesowi dyskretyzacji.

Plan wykładu

Uczenie nadzorowanie sieci wielowarstwowej

Wsteczna propagacja błędów

Dla sieci typu MLP (wielowarstwowy perceptron), najczęściej wykorzystuje się algorytm uczenia BP (backpropagation), czyli wstecznej propagacji błędów.

Uczenie nadzorowane

Sieć w tej metodzie uczy się przez przykłady, czyli, należy dostarczyć zestaw próbek składający się z par wejście-wyjście. Wyjście to znany poprawny wynik dla każdego przypadku. Tak więc znane jest oczekiwane zachowanie sieci neuronowej, a algorytm BP pozwala na takie dopasowanie.

Algorytm BP (wersja uproszczona

1. Ustaw wagi na wartości losowe z przedziału $[-1, 1]$, $k = 0$
2. Dopóki nie osiągnięto założonego progu błędu $e(L) = Err$ lub nie wykonano założonej liczby kroków $k < MaxEpoch$:
   1. Wylosuj ze zbioru próbek dowolny punkt i oblicz odpowiedź sieci , $i = L$
   2. Wykonuj dla warstw $i > 0$
      1. Aktualizuj wagi połączeń w warstwie i zgodnie ze wzorem

$$\mathbf{w}(k+1, i) = \mathbf{w}(k, i) + \eta \mathbf{e}(i)\mathbf{x}_i(i)$$

      2. $i = i - 1$
   3. $k = k + 1$

Gdzie $\mathbf{e}(i)$, to błędy popełniane przez neurony w warstwie (i). W warstwie wyjściowej błąd obliczany jest na podstawie różnicy sygnału oczekiwanego i generowanego przez sieć. W warstwach ukrytych obliczany jest z błędu warstwy następnej.

Słowny opis algorytmu BP

- W kolejnych iteracjach wykonuje się kroki: jedna z próbek uczących jest wprowadzana do sieci.
- Na jej podstawie w oparciu o aktualny stan wag (początkowo wyniki będą losowe) sieć będzie dawała odpowiedź.
- Odpowiedź ta jest porównywana do znanego wyjścia z próbki, obliczany jest średni błąd kwadratowy.
- Wartość błędu jest następnie propagowana wstecz przez sieć i wprowadzane są małe zmiany w wartościach wag w każdej warstwie.
- Zmiany wag wykonuje się tak, by zmniejszyć wartość błędu dla danej próbki.
- Cały proces jest powtarzany dla każdej próbki, i może być wykonywany wielokrotnie.
- Cykl ten powtarza się, aż całkowita wartość błędu spada poniżej progu, który jest z góry określony.

Ważne uwagi do BP

- Metoda działa tak, by minimalizować funkcję błędu $E = \frac{1}{2}\sum_{o=1}^{N_o}(y_o - output_o)$, gdzie y_o — oczekiwana wartość wyjścia a $output_o$ wartość obliczona przez sieć, a N_o — liczba wyjść.
- Podczas nauki, wagi każdego połączenia są zmieniane przez wartość, która jest proporcjonalna do sygnału błędu.
- Można powiedź, że sieć nauczy się rozwiązywania problemu „wystarczająco dobrze". Właściwie sieć nigdy nie będzie tworzyć dokładnego odwzorowania ($Err \neq 0$), ale będzie raczej asymptotycznie zbiegać do funkcji idealnej.
- Ważne, by funkcja aktywacji była funkcją różniczkowalną, gdyż minimalizacja funkcji błędu wiąże się z wyznaczeniem gradientu.

Współczynnik uczenia η

Znaczenie

Reguła uczenia wymaga, by zmiana wag była proporcjonalna do gradientu. Gradientowa reguła największego spadku oczekuje wykonywania nieskończonej liczby kroków. Stałą skalującą jest współczynnik uczenia.

Wartości

W praktyce wybiera się maksymalnie duży współczynnik uczenia, ale taki, który nie prowadzi do oscylacji.

Współczynnik momentum α

Wzór na poprawkę wagi z momentum

$$\mathbf{w}(k+1,i) = \mathbf{w}(k,i) + \underbrace{\eta \mathbf{e}(i)\mathbf{x}_i(i)}_{\Delta w(k+1,i)} + \alpha \Delta w(k,i)$$

Przesłanką do zastosowania współczynnika momentum jest to, że poprzednie zmiany w wagach powinny mieć wpływ na aktualny kierunek przesuwania się w przestrzeni wag. Zatem momentum zapobiega oscylacjom.
Czyli, gdy wagi zaczną przemieszczać się w określonym kierunku w przestrzeni wag, to będą dążyć do dalszego ruchu w tym samym kierunku. Momentum pomaga przeskoczyć przez lokalne minima.

Dwa tryby aktualizowania wag

Tryb onliine

Aktualizacja wag po każdym zaprezentowanym wzorcu.

Tryb wsadowy (batch)

Aktualizacja po prezentacji wszystkich wzorców.

Jeśli współczynnik uczenia jest niewielki, to różnica pomiędzy dwoma procedurami jest niewielka. Znaczne różnice można zaobserwować, gdy współczynnik uczenia jest duży, jako że algorytmu wstecznej propagacji błędów zakłada, że pochodne błędów są zsumowane po wszystkich wzorcach.

Generalizacja

Generalizacja

Uogólnienie, operacja myślowa, polegająca na wykryciu przez uczącego się cechy lub zasady wspólnej dla danej klasy przedmiotów lub zjawisk (psych.). Jest to najbardziej oczekiwana cecha sieci.

Przeuczenie

Sieci bardzo skomplikowane (duża liczba wag) tworzą bardziej złożony model i są przez to bardziej skłonne do nadmiernego dopasowania. Przeuczenie rozpoznaje się po tym, że błąd uczenia jest bardzo mały, a błąd testowania jest duży.

Niedouczenie

Sieci z niewielką liczną neuronów (wag) mogą być niewystarczająco silne do zamodelowania poszukiwanej funkcji odwzorowania wejść w wyjście. Niedouczenie daje duży błąd uczenia. Prawie zawsze większe sieci generują mniejszy błąd.

Kiedy stosować MLP

- Dostępna jest duża liczba danych typu wejście-wyjście, ale nieznana jest relacja pomiędzy wejściem i wyjściem.
- Problem wydaje się mieć ogromną złożoność, ale wyraźnie istnieje rozwiązanie.
- Łatwo jest wygenerować przykłady właściwego zachowania/działania.
- Rozwiązanie problemu może ulec zmianie, w granicach podanych parametrów wejściowych i wyjściowych (tzn. dzisiaj $2 + 2 = 4$, ale w przyszłości może się okazać, że $2 + 2 = 3.8$).
- Wyjścia mogą być rozmyte lub nienumeryczne.