

Neuron liniowy

Najprostsza jest jednostka liniowa:

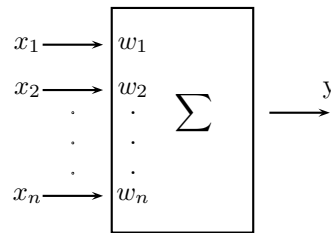
potrafi ona “rozpoznawać” wektor wejściowy $X = (x_1, x_2, \dots, x_n)^T$
zapamiętany we współczynnikach wagowych $W = (w_1, w_2, \dots, w_n)$,

Zauważmy, że $y = W * X$

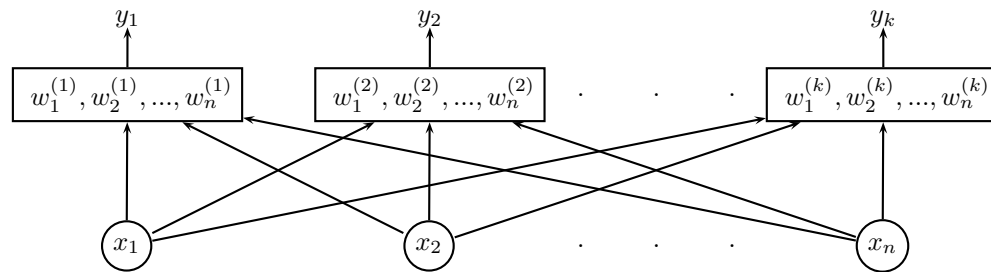
Założmy, że wektor wejściowy i wektor wag są znormalizowane

$\|X\| = 1$ i $\|W\| = 1$

wówczas widać, że $y = \cos(\phi)$



Najprostsza sieć — warstwa elementów liniowych



- warstwa zawiera k neuronów
- każdy neuron ma ten sam zestaw sygnałów wejściowych $X = (x_1, x_2, \dots, x_n)^T$
- każdy neuron ma swój własny wektor wag $W^{(m)} = (w_1^{(m)}, w_2^{(m)}, \dots, w_n^{(m)})$

wyjście m -tego elementu można zapisać: $y^{(m)} = W^{(m)} * X = \sum_{i=1}^n w_i^{(m)} x_i$

Na działanie tej sieci można patrzeć jak na:

- *Grandmother Cell*
- filtr liniowy

Warstwa elementów liniowych realizuje odwzorowanie liniowe

Wyjście sieci zbierzemy w wektor $Y = (y_1, y_2, \dots, y_k)^T$

Z wektorów wag zbudujemy macierz $[k \times n]$ $W = \begin{bmatrix} w_1^{(1)} & w_2^{(1)} & \dots & w_n^{(1)} \\ w_1^{(2)} & w_2^{(2)} & \dots & w_n^{(2)} \\ \vdots & \vdots & & \vdots \\ w_1^{(k)} & w_2^{(k)} & \dots & w_n^{(k)} \end{bmatrix}$

Widać, że $Y = W * X$, czyli nasza warstwa dokonuje pewnego przekształcenia liniowego $X \in \mathcal{R}^n$ w $Y \in \mathcal{R}^k$ zadanego macierzą W .

Problem: jak znaleźć macierz W dla konkretnego odwzorowania?

Uczenie pojedynczego neuronu — reguła delta

Reguła delta jest przykładem uczenia z nauczycielem.

Zamiast znajdować macierz W od razu można ją znaleźć iteracyjnie w procesie "uczenia" sieci.

Jeśli dla każdego wektora wejściowego X umiemy podać porządaną wartość wyjścia neuronu z , to możemy obliczyć błąd jaki ten neuron popełnił: $\delta = z - y$

Okazuje się, że aby neuron lepiej realizował nasze odwzorowanie, należy skorygować jego wagi w następujący sposób:

$$W' = W + \eta \delta X$$

Dlaczego?

- Dlaczego dodawać fragment wejścia X do wektora wag W ?
- dlaczego δ ?
- dlaczego η ?

Dowód zbieżności reguły delta (1)

W procesie uczenia posługujemy się ciągiem par $\{(X^{(j)}, z^{(j)})\}_{j=1, \dots, N}$ opisującym wejście i rządzane wyjście sieci. W j -tym kroku sieć popelnia błąd

$$\delta^{(j)} = z^{(j)} - y^{(j)}$$

gdzie

$$y^{(j)} = W^{(j)} * X^{(j)}.$$

Celem uczenia jest jak najlepsze odtwarzanie przez sieć ciągu uczącego w sensie *najmniejszych kwadratów*, czyli minimalizacja funkcji:

$$Q = \frac{1}{2} \sum_{j=1}^N (\delta^{(j)})^2 = \frac{1}{2} \sum_{j=1}^N (z^{(j)} - y^{(j)})^2 = \sum_{j=1}^N Q^{(j)}$$

gdzie wprowadziliśmy: $Q^{(j)} = \frac{1}{2} (z^{(j)} - y^{(j)})^2$

Dowód zbieżności reguły delta (2)

Ponieważ $Q = Q(W)$ więc w j -tym kroku uczenia musimy zmieniać i -tą składową wektora wag w kierunku przeciwnym do gradientu Q :

$$w_i^{(j+1)} - w_i^{(j)} = \Delta w_i^{(j)} = -\frac{\partial Q^{(j)}}{\partial w_i} = -\frac{\partial Q^{(j)}}{\partial y^{(j)}} \frac{\partial y^{(j)}}{\partial w_i} = \delta^{(j)} x_i^{(j)}$$
$$\underbrace{\frac{\partial Q^{(j)}}{\partial y^{(j)}}}_{-\left(z^{(j)} - y^{(j)}\right) = -\delta^{(j)}} \quad \underbrace{\frac{\partial y^{(j)}}{\partial w_i}}_{x_i^{(j)}}$$

ostatecznie: $\Delta w_i^{(j)} = \delta^{(j)} x_i^{(j)}$

Ponieważ Q jest unimodalną paraboloidą eliptyczną, więc ma jedno minimum, a więc procedura minimalizacji gradientowej jest zbieżna. Ponieważ ciąg uczący jest stochastyczny, więc dla zapewnienia płynniejszej zbieżności szybkość zmiany wektora wag jest kontrolowana przez parametr $0 < \eta < 1$:

$$\Delta w_i^{(j)} = \eta \delta^{(j)} x_i^{(j)}$$

Z powodów praktycznych istotne jest aby $\forall_{i,j} w_i^{(0)} \neq w_j^{(0)}$.

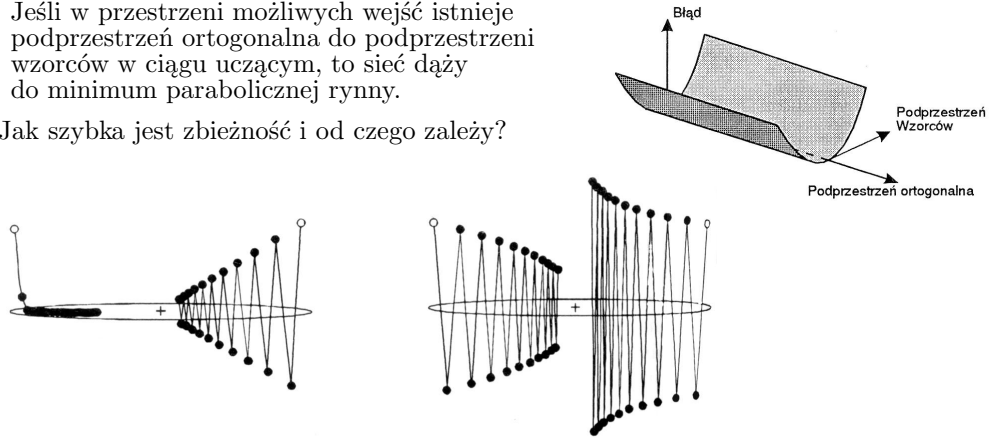
Dla liniowego odwzorowania $y = f(X)$ mamy zagwarantowane, że znajdziemy W zapewniające dokładne dopasowanie, zaś dla f nieliniowego osiągniemy najlepsze dopasowanie w sensie minimum błędu średniokwadratowego.

Reguła delta: uwagi

⇒ Jak wpływa dobór ciągu uczącego na to, czego sieć się nauczyła?

- Jeśli ciąg uczący rozpina przestrzeń możliwych wejść, to sieć dąży do globalnego minimum.
- Jeśli w przestrzeni możliwych wejść istnieje podprzestrzeń ortogonalna do podprzestrzeni wzorców w ciągu uczącym, to sieć dąży do minimum parabolicznej rynny.

⇒ Jak szybka jest zbieżność i od czego zależy?



Kolejne trajektorie dla: $\eta = 0.2$, $\eta = 0.0476$, $\eta = 0.049$, $\eta = 0.0505$. Powierzchnia:
 $E = x^2 + 20y^2$

Uczenie sieci elementów liniowych

Reguła delta przenosi się w naturalny sposób na sieć elementów liniowych w postaci warstwy:

- w ciągu uczącym $\{(X^{(j)}, Z^{(j)})\}_{j=1, \dots, N}$ zamiast wartości z podajemy wektor wartości porządkanych Z
- zamiast modyfikować wektor wag, modyfikujemy macierz wag W

$$W_k^{(j+1)} = W_k^{(j)} + \eta (Z^{(j)} - Y^{(j)}) (X^{(j)})^T$$

Sieci liniowe MADALINE (Many Adaptive Linear Elements) z tym algorytmem uczenia, są wykorzystywane jako filtry adaptacyjne np.:

- do tłumienia "echa" w liniach telefonicznych
- do poprawiania stosunku sygnału do szumu

Uczenie bez nauczyciela

Reguła Hebba: Wzmocnieniu ulegają te wagi $w_i^{(m)(j)}$, dla których wartość wejścia $x_i^{(j)}$ jest duża podczas gdy neuron jest pobudzony ($y_m^{(j)}$ jest duże):

$$w_i^{(m)(j+1)} = w_i^{(m)(j)} + \eta x_i^{(j)} y_m^{(j)}$$

gdzie: m — numer neuronu, j — numer kroku uczenia, i — numer wejścia neuronu

$$y_m^{(j)} = \sum_{i=1}^n w_i^{(m)(j)} x_i^{(j)}$$

Dlaczego ta reguła działa?

Jeśli w chwili początkowej, któryś neuron miał zestaw wag zbliżony do prezentowanego sygnału to w następnych pokazach tego samego sygnału “rozpoznawanie” go przez ten neuron będzie coraz silniejsze. \Rightarrow Sieć tak uczona zaczyna klasyfikować sygnały.

Ograniczenia:

- przebieg uczenia zależy od wartości początkowych wag
- nie ma gwarancji, że jednej klasie wzorców będzie odpowiadał jeden neuron
- nie ma gwarancji, że wszystkie klasy wzorców będą miały oddzielne reprezentacje w postaci oddzielnych zbiorów neuronów.

Modyfikacje reguły Hebba

- przyrostowa reguła Hebba — uzależniamy proces zmiany wag od zmian wartości wejścia i wyjścia w kolejnych krokach uczenia:

$$w_i^{(m)(j+1)} = w_i^{(m)(j)} + \eta \left[(x_i^{(j)} - x_i^{(j-1)}) (y_m^{(j)} - y_m^{(j-1)}) \right]$$

- “gwiazda wejść” (*Grossberg 1974: Instar training*): w każdym kroku, uczeniu rozpoznawania bieżącego bodźca podlega tylko jeden wybrany neuron:

$$w_i^{(m)(j+1)} = w_i^{(m)(j)} + \eta^{(j)} (x_i^{(j)} - w_i^{(m)(j)})$$

przy czym z praktyki $\eta = 0.1 - \lambda j$

- i wiele innych

Uczenie z forsowaniem

Techniki uczenia bez nauczyciela można zaadaptować do przypadku kiedy znana jest porządana odpowiedź. Pomysł sprowadza się do podstawienia w miejsce prawdziwej odpowiedzi sieci y porządanym wartości z :

- metoda Hebba:

$$w_i^{(m)(j+1)} = w_i^{(m)(j)} + \eta x_i^{(j)} z_m^{(j)}$$

- przyrostowa metoda Hebba:

$$w_i^{(m)(j+1)} = w_i^{(m)(j)} + \eta \left[(x_i^{(j)} - x_i^{(j-1)}) (z_m^{(j)} - z_m^{(j-1)}) \right]$$

Przekonajmy się, że reguła Hebba z forsowaniem działa:

Dla całej warstwy reguła ta wygląda tak:

$$W^{(m)(j+1)} = W^{(m)(j)} + \eta Z^{(j)} [X^{(j)}]^T$$

Efekt uczenia:

$$W = \sum_{j=1}^N \eta Z^{(j)} [X^{(j)}]^T + W^{(1)}$$

Niech $W^{(1)} = 0$ oraz

1. wektory w ciągu uczącym są ortonormalne $\forall_j [X^{(j)}]^T X^{(m)} = \begin{cases} 1 & \text{dla } j = m \\ 0 & \text{dla } j \neq m \end{cases}$

wtedy po prezentacji bodźca $X^{(m)}$ sieć odpowie:

$$Y = WX^{(m)} = \sum_{j=1}^N \eta Z^{(j)} [X^{(j)}]^T X^{(m)} = \eta Z^{(j)}$$

2. wektory w ciągu uczącym są skrajnie skorelowane tzn: $X^{(j)} = X + \epsilon^{(j)}$, a porządana odpowiedź sieci jest stale Z , wówczas:

$$W = \sum_{j=1}^N \eta Z^{(j)} [X^{(j)}]^T = \sum_{j=1}^N \eta Z^{(j)} [X + \epsilon^{(j)}]^T =$$

$$\eta Z \left(NX^T + \sum_{j=1}^N [\epsilon^{(j)}]^T \right)$$

tzn: sieć tworzy reprezentację uśrednionego wzorca.

W rzeczywistych sytuacjach oba efekty występują równocześnie. Empirycznie szacowana pojemność sieci k elementowej jest $N_{max} \approx \frac{k}{2 \log k}$

Przyspieszanie uczenia

- kontrolowanie wartości $\eta^{(j)}$
- dodanie składnika bezwładności:
 $W^{(j+1)} = W^{(j)} + \eta_1 (Z^{(j)} - Y^{(j)}) [X^{(j)}]^T + \eta_2 M^{(j)}$ gdzie
 $M^{(j)} = W^{(j)} - W^{(j-1)}$
- wygładzanie wykładnicze:
 $W^{(j+1)} = W^{(j)} + \eta_1 [(1 - \eta_2) (Z^{(j)} - Y^{(j)}) [X^{(j)}]^T + \eta_2 M^{(j)}]$
- technika kumulowania błędów: zbiór uczący dzieli się na podzbiory o pewnej długości η_3 ($30 \leq \eta_3 \leq 50$) numerowanych indeksami $j^* = 0, \dots, P$ skumulowany błąd:

$$S^{(j^*)} = \sum_{j=\eta_3 j^*+1}^{\eta_3(j^*+1)} \eta_1 (Z^{(j)} - Y^{(j)}) [X^{(j)}]^T$$
i używając tej wielkości modyfikuje się wagi:
 $W^{(j^*+1)} = W^{(j^*)} + S^{(j^*)}$
ta korekta wag występuje w co η_3 kroku uczenia.

Nie ma teorii mówiącej, jak dobierać parametry η_1 , η_2 . Zwykle uczenie rozpoczyna się od dużych wartości, stopniowo je redukując.

Uczenie z rywalizacją: sieci Kohonena

Uczenie na pierwszy rzut oka wygląda identycznie z "instar", uczeniu w danym kroku podlega tylko jeden neuron:

$$w_i^{(m^*)(j+1)} = w_i^{(m^*)(j)} + \eta^{(j)} (\tilde{x}_i^{(j)} - w_i^{(m^*)(j)})$$

ale są dwie zasadnicze różnice:

- wektor wejściowy musi być znormalizowany: $\|X\| = 1$
- neuron podlegający uczeniu w danym kroku nie jest wybierany dowolnie, lecz jest to ten m^* , który dla danego bodźca produkuje największe wyjście
 $y_{m^*}^{(j)} = \max_m (y_m^{(j)})$

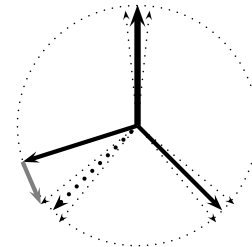
Co nam dają te warunki?

normalizacja — największa wartość wyjścia sieci dla neuronu m^* :

$y_{m^*} = \sum_{i=1}^N w_i^{(m^*)} \tilde{x}_i$
zapewnia to, że wektor wag tego neuronu znajduje się najbliżej wektora bodźca na sferze jednostkowej.

uczenie neuronu m^* — szybkość nauki.

zbieżność do klastrów — jeśli w zbiorze uczącym występują jakieś klastry, to wektory wag są modyfikowane w ten sposób, że dążą do wartości średnich w klastrach, przy czym my nie musimy wiedzieć *a priori* o istnieniu tych klastrów.



Sąsiedztwo

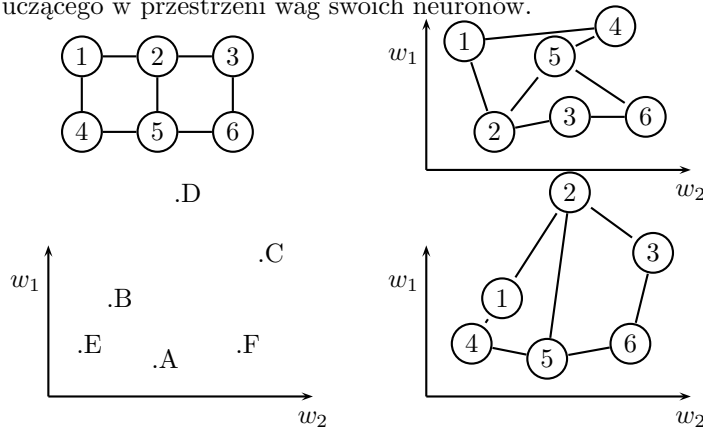
Neurony w tej sieci mogą być uporządkowane. Można więc mówić o relacji sąsiedztwa. Rozszerzeniem oryginalnej koncepcji Kohonena jest uczenie nie tylko jednego wygrywającego neuronu, lecz także jego sąsiadów.

$$w_i^{(m)(j+1)} = w_i^{(m)(j)} + \eta^{(j)} h(m, m^*) (\tilde{x}_i^{(j)} - w_i^{(m)(j)})$$

Pojęcie sąsiedztwa może być różne np.:

$$h(m, m^*) = \frac{1}{\rho(m, m^*)} \text{ albo } h(m, m^*) = \exp(-\rho(m, m^*)^2)$$

Sz szczególnie ciekawe efekty daje więcej niż jedno-wymiarowe sąsiedztwo: sieci Kohonena mają własność odwzorowywania topograficznych własności zbioru uczącego w przestrzeni wag swoich neuronów.



Ograniczenia sieci elementów liniowych

- sieć może jedynie realizować liniowe odwzorowania $X \Rightarrow Y$
- w odróżnieniu od sieci nieliniowych, liniowe sieci wielowarstwowe nie mają sensu, bo złożenie operacji liniowych da nam i tak operację liniową