Sieci Neuronowe

Wykład 4
Wariacje na temat propagacji wstecznej
Sieci CP

wykład przygotowany na podstawie.

S. Osowski, „Sieci neuronowe w ujęciu algorytmicznym”, WNT, 1996
J. Hertz, A. Krogh, R.G.Palmer, „Wstęp do teorii obliczeń neuronowych”, WNT, 1993
R. Tadeusiewicz, “Sieci Neuronowe”, Rozdz. 5. Akademicka Oficyna Wydawnicza RM, Warszawa 1993.

Uczenie sieci nieliniowej

Dla *sieci wielowarstwowych*, dla *warstw wewnętrznych* nie ma możliwości bezpośredniego określenia oczekiwanych (wymaganych) wartości sygnałów wejściowych $z^{(j)}$, a tym samych określenia wartości błędu $\delta^{(j)}$.

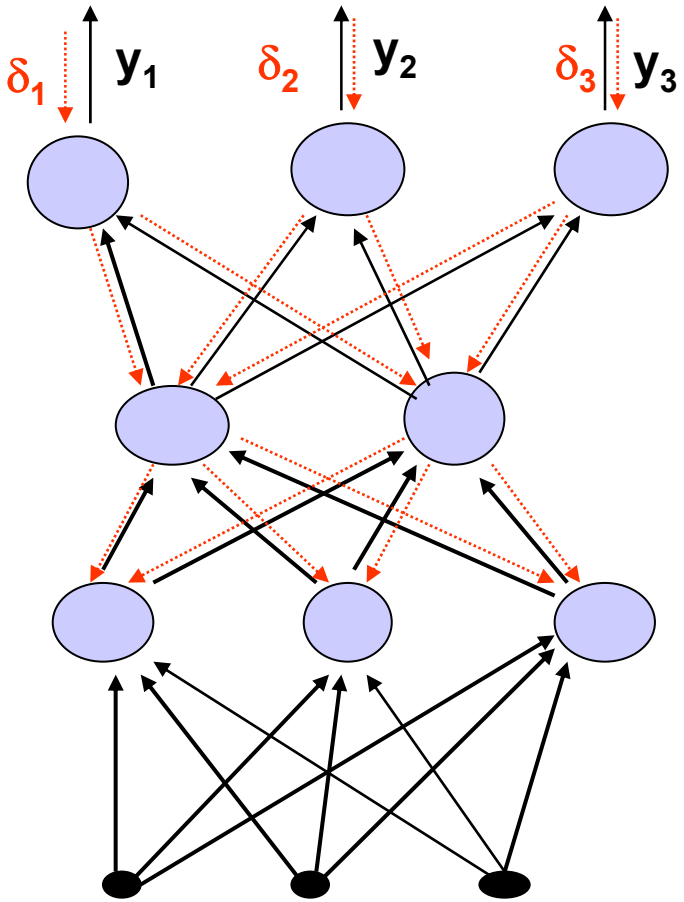
Rozważając ciąg

$$U = < <X^{(1)} , Z^{(1)} >, < X^{(2)} , Z^{(2)} > , ... , <X^{(N)} ,Z^{(N)} > >$$

mamy do dyspozycji n-wymiarowe wektory wejściowe X oraz k-wymiarowe wektory wyjściowe Z z neuronów terminalnych. Jeżeli odnotujemy błąd, czyli różnice $(X^{(j)} - Z^{(j)})$, to nie będziemy w stanie ustalić w jaki sposób za pojawienie się błędu odpowiadają neurony warstwy wyjściowej a jaki sposób powstał w elementach wcześniejszych (wewnętrznych) warstw.
Noszą one nazwę *warstw ukrytych*, “hidden layers”.

Uczenie sieci nieliniowej

W latach 80-tych zaproponowano algorytm
tzw. *wstecznej propagacji błędów* (backpropagation),
polegający na tym że mając wyznaczony błąd $\delta^{(m)(j)}$ (j-ty krok uczenia m-tego neuronu) możemy "rzutować" ten błąd wstecz do wszystkich tych neuronów, których sygnały stanowiły wejścia do m-tego neuronu.



Uczenie sieci nieliniowej

Przy założeniu ciągłości funkcji kryterialnej, najskuteczniejszymi metodami uczenia pozostają gradientowe metody optymalizacyjne, w których uaktualnianie wektora wag odbywa się zgodnie ze wzorem

$$W(k+1) = W(k) + \Delta W$$

gdzie

$$\Delta W = \eta\, p(W)$$

η = wspolczynnik uczenia
$p(W)$ = kierunek w przestrzeni wielowymiarowej W

Uczenie przy zastosowaniu metod gradientowych wymaga wyznaczenia kierunku $p(W)$, czyli wyznaczenia wektora gradientu względem wszystkich wag sieci.

$$\Delta W = - \eta\, \nabla(W)$$

∇ - oznacza gradient

Uczenie sieci nieliniowej

Na samym początku wyznacza się poprawki dla neuronów stanowiących wyjściową warstwę sieci. Dla poszczególnych sygnałów $y_m^{(j)}$ istnieją w ciągu uczącym wzorcowe (oczekiwane) wartości $z_m^{(j)}$, z którymi można je porównywać, wyznaczając bezpośrednio błąd $\delta_m^{(j)}$.

$$\delta_m^{(j)} = (y_m^{(j)} - z_m^{(j)})$$

$$\Delta\omega_i^{(m)(j)} = \eta\, \delta_m^{(j)}\, d\phi(e)/de_m^{(j)}\, y_i^{(j)}$$

dla funkcji logistycznej wzór ten ulega znacznemu uproszczeniu:

$$\Delta\omega_i^{(m)(j)} = -\eta\, (z_m^{(j)} - y_m^{(j)})\, (1 - y_m^{(j)})\, y_i^{(j)}\, y_m^{(j)}$$

$k \in \Omega_0$

Uczenie sieci nieliniowej

Dla warstwy ukrytej, przez analogię możemy zapisać

$$\Delta\omega_i^{(m)(j)} = \eta\, \delta_m^{(j)}\, d\phi(e)/de_m^{(j)}\, y_i^{(j)}$$

ale teraz nie mamy możliwości bezpośrednego wyznaczenia $\delta_m^{(j)}$.
Załóżmy, że rozważany neuron należy do warstwy ukrytej, ale sygnały od niego docierają tylko do warstwy wyjściowej (dla której potrafimy określić $\delta_k^{(j)}$).
Wówczas (*backpropagation*)

$$\delta_m^{(j)} = \Sigma\ \omega_m^{(k)(j)}\, \delta_k^{(j)}$$

$\omega_m^{(k)(j)}$ – waga w neuronie o numerze k, przy jego wejściu m

Rzutowane wstecznie błędy przemnażane są przez te same współczynniki, przez które mnożone były sygnały, tyle tylko, że kierunek przesyłania informacji zostaje w tym przypadku odwrócony; zamiast od wejścia do wyjścia przesyła się je od wyjścia kolejno w kierunku wejścia.

Uczenie sieci nieliniowej

- Powyższą technikę propagacji wstecznej błędów należy powtarzać dla kolejno coraz głębszej warstwy sieci. Każdy neuron z warstwy ukrytej albo przesyła sygnały do wartości wyjściowych, albo znajduje się w jednej z głębszych warstw, wówczas jego błąd można oszacować z chwilą określenia błędów dla wszystkich neuronów w sieci które są odbiorcą jego sygnałów.

- Uaktualnianie wag może odbywać się po każdorazowej prezentacji próbki uczącej lub jednorazowo (w sposób skumulowany) po prezentacji wszystkich próbek tworzących cykl uczący.

Uczenie sieci nieliniowej

Wprowadzając oznaczenia:

K – neurony w warstwie ukrytej,

v – sygnał wychodzący z warstwy ukrytej

M – neurony w warstwie wyjsciowej,

N – neurony wejsciowe,

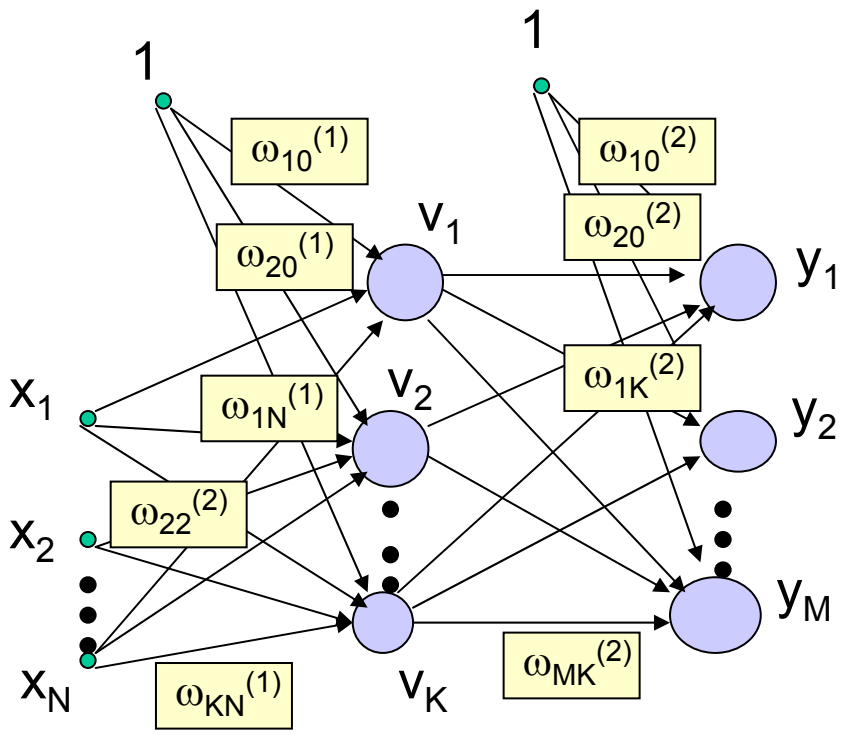
i,j=0 odpowiada jednostkowej polaryzacji $x=[1,x_1,x_2,....x_N]^T$

$$Q = \tfrac{1}{2} \sum_{k=1}^{M} \left(\phi \sum_{i=0}^{K} \left(\omega^{(2)}_{ki} v^{(i)} - z^{(k)} \right)^2 \right.$$

$$= \tfrac{1}{2} \sum_{k=1}^{M} \left(\phi \sum_{i=0}^{K} \left(\omega^{(2)}_{ki} \phi \left(\sum_{j=0}^{N} \omega^{(1)}_{ij} x^{(i)} \right) - z^{(k)} \right)^2 \right.$$

Odpowiednie składniki gradientu otrzymuje się przez zróżniczkowanie powyższej zależności.

Uczenie sieci nieliniowej



W sieci *feedforward* zawsze daje się określić taką kolejność wstecznej propagacji błędów, która pozwala obliczyć błędy dla wszystkich elementów sieci.
Uczenie tą metoda jest stosunkowo skuteczne ale powolne.

Propagacja wsteczna

Algorytm podstawowy jest wolnozbieżny w sieci wielowarstwowej, liczne wariacje proponują jak go przyspieszyć. Inne cele modyfikacji dotyczą unikania minimów lokalnych i poprawy zdolności generalizacji.

Istnieje wiele parametrów które można rozpatrywać w zakresie usprawnienia ogólnej metody propagacji wstecznej, włącznie z architekturą (liczbą warstw, liczbą jednostek w warstwie), wielkością i naturą zbioru treningowego i regułą modyfikacji wag.

Dyskusja dotyczyć będzie przede wszystkim zmian reguł modyfikacji, utrzymując architekturę bez zmian.

Metody gradientowe

Ponieważ funkcja kryterialna Q = Q(W), zatem poszukiwanie minimum może być dokonywane metodą gradientową. Skupiając uwagę na i-tej składowej wektora W, możemy więc zapisać:

$$\omega'_i - \omega_i = \Delta \omega_i = \eta \frac{\partial Q}{\partial \omega_i}$$

Wzór ten można interpretować w sposób następujący: poprawka $\Delta \omega_i$ jakiej powinna podlegać i-ta składową wektora W, musi być proporcjonalna do i-tej składowej gradientu funkcji Q.
Znak – w omawianym wzorze wynika z faktu, że gradient Q wskazuje kierunek najszybszego wzrastania tej funkcji, podczas gdy w omawianej metodzie zależy nam na tym, by zmieniać W w kierunku najszybszego malenia błędu popełnianego przez sieć, czyli w kierunku najszybszego malenia funkcji Q. Współczynnik proporcjonalności $\eta^{(i)}$ określa wielkość kroku $\Delta \omega_i$ i może być w zasadzie wybierany dowolnie.

Metody gradientowe

Gradientowa minimalizacja funkcji błędu (funkcji kryterialnej)

Rozpisując wzór gradientowego uczenia dla kroku j, otrzymujemy:

$$\omega_i^{(j+1)} - \omega_i^{(j)} = \Delta \omega_i^{(j)} = - \frac{\partial Q^{(j)}}{\partial \omega_i} \eta$$

Uwzględniając fakt, że Q zależne jest od y, a dopiero y jest funkcją wektora wag W, możemy zapisać wzór, odpowiadający pochodnej funkcji złożonej

$$\frac{\partial Q^{(j)}}{\partial \omega_i} = \frac{\partial Q^{(j)}}{\partial y^{(j)}} \frac{\partial y^{(j)}}{\partial \omega_i}$$

Na podstawie zależności $Q^{(j)} = \frac{1}{2} (z^{(j)} - y^{(j)})^2$ można ustalić że,

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

Funkcje kryterialna

Kwadratowa funkcja kryterialna,

$$Q = \tfrac{1}{2} \sum [\, y\text{-}z \,]^2$$

Nie jest jedyną możliwością. Możemy zastąpić czynnik $(y\text{-}z)^2$ dowolną inną funkcją różniczkowalną, która ma minimum jeżeli jej argumenty są sobie równe, oraz możemy wprowadzić odpowiednią regułę modyfikacji. Bezpośrednie różniczkowanie wskazuje, że tylko wyrażenie bezpośrednio definiujące $\delta^{(m)}$ w warstwie wyjściowej zmienia się, a wszystkie pozostałe równania propagacji wstecznej pozostają bez zmian.

Do funkcji kryterialnej można też dodać parametry zmieniające jej stromość lub pofałdowanie w procesie uczenia.

Spadek gradientu może być bardzo powolny, jeżeli η jest małe, i może mieć duże oscylacje jeżeli η jest duże.

Składnik momentu w funkcji kryterialnej

Najprostsze podejście to dodanie momentu, pomysł polega na nadaniu każdej wadze pewnej bezwładności, czyli momentu, w wyniku czego ma ona skłonność do zmian "kierunku średniej " zamiast popadania w oscylacje przy każdym małym impulsie.

$$\Delta \omega_{pq}^{(j+1)} = -\eta \, \partial Q/\partial \omega + \alpha \, \Delta \omega_{pq}^{(j)}$$

Jeżeli $\partial Q/\partial \omega$ jest prawie stałe to wówczas

$$\Delta \omega_{pq}(t+1) = -\eta/(1-\alpha) \; \partial Q/\partial \omega$$

Przy wartości współczynnika $\alpha = 0.9$ oznacza to 10-krotny wzrost efektywności wartości współczynnika uczenia, a więc przyspieszenie.

Algorytm największego spadku

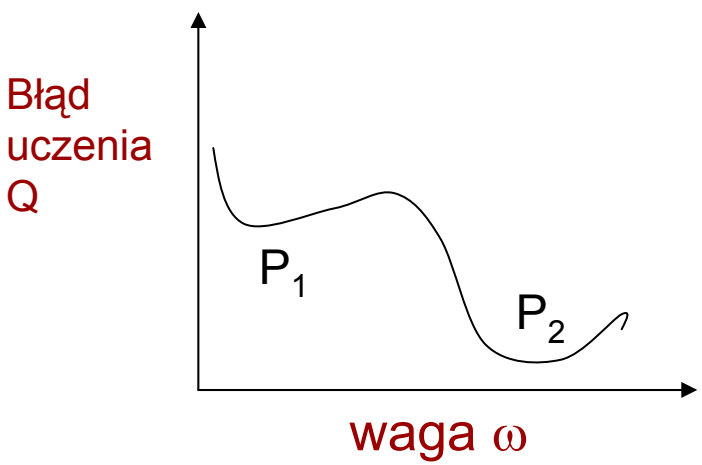
Nie jest łatwo wybrać odpowiednie wartości parametrów η i α dla określonego problemu. Ponadto najlepsze wartości w początkowej fazie uczenia mogą nie być dość dobre później.

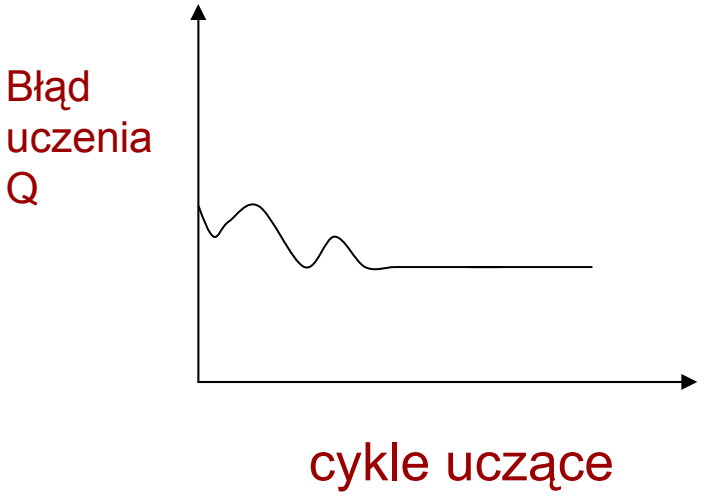
Spadek gradientu może być bardzo powolny, jeżeli η jest małe, i może mieć duże oscylacje jeżeli η jest duże.

Ważne jest aby czynnik momentu nie stał się dominujący w procesie uczenia, gdyż prowadziłoby to do niestabilności algorytmu.
Zwykle kontroluje się zmiany wartości funkcji celu w procesie uczenia, dopuszczając do jej wzrostu jedynie w ograniczonym zakresie, np. 5%.
Jeżeli kolejna iteracja nie spełnia tego warunku to krok jest pomijany.
W tym momencie składnik gradientowy odzyskuje dominację ponad składnikiem momentu i proces przebiego zgodnie z kierunkiem minimalizacji wyznaczonym przez wektor gradientu.

Algorytm największego spadku

Ilustracja działania czynnika momentu prowadzącego do wyprowadzenia procesu uczenia z minimum lokalnego. Charakterystyka ma dwa minima: P_1, P_2. Przy wybranym warunku startowym wagi algorytm znalazł się w lokalnym minimum P_1 ktore opuścił dzięki czynnikowi momentu. Ostatecznie zakonczył proces uczenia w minimum globalnym P_2, ale po kilku oscylacjach dopiero, spowodowanych "pasożytniczym" wpływem współczynnika momentu..





Algorytm zmiennej metryki

W metodzie tej wykorzystuje się kwadratowe przybliżenie funkcji kryterialnej $Q(W)$ w sasiedztwie znanego rozwiązania W_k.
Możemy więc rozwinąć:

$$Q(W+p) = Q(W) + [g(W)^T]p + 1/2p^T H(W)p$$

Gdzie $g(W) = \nabla Q = [\ \partial Q/\partial \omega_1, \partial Q/\partial \omega_n, ..., \partial Q/\partial \omega_n\ ]^T$ jest wektorem gradientu, a symetryczna macierz kwadratowa $H(W)$ jest macierza drugich pochodnych $(\partial^2 Q/(\partial w_1\, \partial w_2\ ))$.

Minimum funkcji wymaga aby $dQ(W_k+p)/dp = 0$. Dokonując odpowiedniego różniczkowania otrzymuje się

$$g(W_k)+H(W_k)p_k=0$$

A więc ostatecznie

$$p_k= -[H(W_k)]^{-1}\, g(W_k)$$

Algorytm zmiennej metryki

W praktyce wyznaczenie hesjanu H(W) w każdym kroku nie jest stosowane, stosuje się przybliżenie G(W) oparte na znajomości wartości g(W), tak aby

$$G(W_k)\,(W_k - W_{k-1}) = g(W_k) - g\,(W_{k-1})$$

Oznaczmy:

$$s_k = W_k - W_{k-1},\ r_k = g(W_k) - g(W_{k-1}),$$
$$V_k = [G(W_k)]^{-1},\ V_{k-1} = [G(W_{k-1})]^{-1}$$

To wówczas proces uaktualniania wartości macierzy V_k opisuje się zależnością rekurencyjną

$$V_k = V_{k-1} + (1 + r_k^T V_{k-1} r^k)/(s_k^T r_k) - s_k r_k^T V_{k-1} + V_{k-1} r_k s_k^T/(s_k^T r_k)$$

Algorytm zmiennej metryki

Metoda zmiennej metryki charakteryzuje się zbieżnością superliniową. Jest więc znacznie lepsza od liniowo zbieżnej metody największego spadku. Fakt, że hesjan w każdym kroku spełnia warunek dodatniej określoności daje pewność, że spełnienie warunku $g(W_k) = 0$ odpowiada rozwiazaniu problemu optymalizacji.

Metoda ta jest uważana za jedną z najlepszych metod optymalizacji funkcji wielu zmiennych. Jej wadą jest stosunkowo duża złożoność obliczeniowa (konieczność wyznaczenia n^2 elementów hesjanu), a także duże wymagania co do pamięci przy przechowywaniu macierzy hesjanu. Z tego względu stosuje się ją do niezbyt dużych sieci.

Metoda gradientów sprzężonych

W metodzie tej podczas wyznaczania kierunku minimalizacyjnego rezygnuje się z bezpośredniej informacji o hesjanie. Kierunek poszukiwań p_k jest konstruowany w taki sposób, aby był ortogonalny oraz sprzężony ze wszystkimi poprzednimi kierunkami p_0, p_1, ..., p_{k-1}, tzn.

$$p_i^T G p_j = 0 \quad i \neq j$$

Wektor p_k ktory spełnia powyższe założenia ma postać:

$$p_k = - g_k + \sum \beta_{kj} p_j$$

Przy czym $g_k = g(W_k)$ oznacza aktualną wartość wektora gradientu, a sumowanie dotyczy poprzednich kierunków minimalizacyjnych. Korzystając z warunku ortogonalności oraz uwzględniając sprzeżenie między wektorami, możemy zapisać:

$$p_k = -g_k + \beta_{k-1} p_{k-1}$$

Metoda gradientów sprzężonych

Współczynnik sprzężenia odgrywa bardzo ważną rolę, kumulując w sobie informację o poprzednich kierunkach poszukiwań. Istnieje wiele odmiennych reguł wyznaczania tego współczynnika.
Najbardziej znane z nich to:

$$\beta_{k-1} = g_k^T (g_k - g_{k-1}) / (g^T_{k-1}\, g_{k-1})$$

$$\beta_{k-1} = g_k^T (g_k - g_{k-1}) / (-p_{k-1}\, g_{k-1})$$

Ze względu na kumulację błędów zaokrąglenia podczas kolejnych cykli obliczeniowych metoda gradientów sprzężonych w praktyce zatraca właściwość ortogonalności między wektorami kierunków minimalizacyjnych. Po n-iteracjach przeprowadza się ponowny start.
Metoda wykazuje zbieżność zbliżoną do liniowej, jest mniej skuteczna niż metoda zmiennej metryki, ale zdecydowanie szybsza niż metoda największego spadku.
Stosuje się ją jako skuteczny algorytm przy dużej liczbie zmiennych.

Metody doboru współczynnika uczenia

Po określeniu właściwego kierunku p_k i wyborze na nim nowego rozwiązania W_{k+1}, należy tak dobrać wartość η_k, aby nowy punkt rozwiązania

$$W_{k+1} = W_k + \eta_k p_k$$

leżał możliwie blisko minimum funkcji $Q(W)$ na kierunku p_k.

Właściwy wybór współczynnika η_k ma ogromny wpływ na zbieżność algorytmu optymalizacji do minimum funkcji celu.

→ przyjęcie zbyt małej wartości η powoduje niewykorzystanie możliwości zminimalizowania wartości funkcji celu w danym kroku i konieczność jego powtórzenia w następnym.

→ zbyt duży krok powoduje "przeskoczenie" minimum funkcji i podobny efekt jak poprzednio.

Empirycznym "przepisem" stosowanym w sieciach neuronowych jest dobór

$$\eta < \min (1/n_i)$$

gdzie n_i oznacza liczbe wejść i-tego neuronu.

Sieci CP (Counter Propagation)

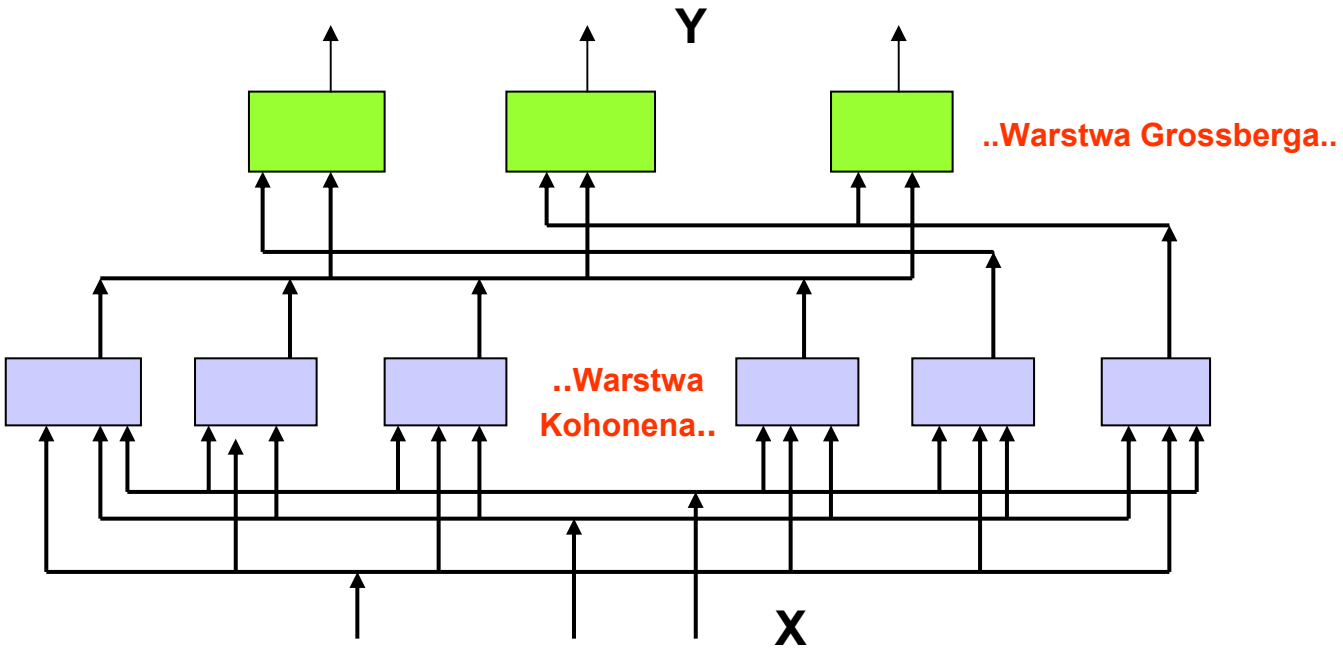
Wada, jaką jest powolny i uciążliwie pracochłonny proces uczenia w sieciach ze wsteczną propagacją błędów sprawiła, że pojawiły się jej sprawniejsze modyfikacje.

Jedną z najbardziej znanych jest propozycja Hecht-Nielsena, określana jako sieć

"Counter-Propagation" (CP).

Sieci CP (Counter Propagation)

Sieć CP właściwie nie jest oryginalną propozycją, lecz stanowi kompilację sieci Kohonena i sieci Grossberga. Zestawienie tych sieci w strukturze sieci CP wprowadziło istotnie nową jakość – sieć stosunkowo szybko się uczącą i mającą (potencjalnie) nieograniczony zakres możliwych odwzorowań pomiędzy sygnałem wejściowym X i wyjściowym Y.



Przypomnienie: metoda gwiazdy wyjsc

Metoda gwiazdy wyjść (Grossberga):

W koncepcji tej rozważa się wagi wszystkich neuronów całej warstwy, jednak wybiera się wyłącznie wagi łączące te neurony z *pewnym ustalonym wejściem*.
W sieciach wielowarstwowych wejście to pochodzi od pewnego ustalonego neuronu wcześniejszej warstwy i to właśnie ten neuron staje się *"gwiazdą wyjść"* (outstar).

$$\omega_i^{(m)(j+1)} = \omega_i^{(m)(j)} + \eta^{(j)} \left(y_m^{(j)} - \omega_i^{(m)(j)} \right)$$

W powyższym wzorze i jest ustalone, natomiast m jest zmienne i przebiega wszelkie możliwe wartości (m = 1,2,.....,k).
Reguła zmieniania $\eta^{(j)}$ jest dana wzorem

$$\eta^{(j)} = 1 - \lambda j$$

Przypomnienie: uczenie Kohonena

Uczenie z rywalizacja **(competitive learning)**
wprowadził Kohonen przy tworzeniu sieci neuronowych uczących się realizacji *dowolnych odwzorowań* $X \Rightarrow Y$.
Zasada uczenia z rywalizacją jest formalnie identyczna z regułą "instar"

$$\omega_i^{(m^*)(j+1)} = \omega_i^{(m^*)(j)} + \eta^{(j)} \left(x_i^{(j)} - \omega_i^{(m^*)(j)} \right)$$

z dwoma dość istotnymi uzupełnieniami.
$\Rightarrow$ Wektor wejściowy X jest przed procesem uczenia normalizowany tak, aby $||X|| = 1$.
$\Rightarrow$ Numer podawanego treningowi neuronu m* nie jest przypadkowy czy arbitralnie wybierany, jest to bowiem ten (i tylko ten) neuron którego sygnał wyjściowy $y_{m^*}^{(j)}$ jest największy. Przy każdorazowym podaniu sygnału wejściowego $X^{(j)}$ neurony rywalizują ze sobą i wygrywa ten, który uzyskał największy sygnał wyjściowy $y_{m^*}^{(j)}$. Tylko ten zwycięski neuron podlega uczeniu, którego efektem jest jeszcze lepsze dopasowanie wag $W^{(m^*)(j+1)}$ do rozpoznawania obiektów podobnych do $X^{(j)}$.

Przypomnienie: uczenie Kohonena

Reguła uczenia Kohonena bywa często wzbogacana o dodatkowy element związany z topologią uczącej się sieci. Neurony w sieci są *uporządkowane*, można więc wprowadzić pojęcie *sąsiedztwa*. Uogólniona metoda samoorganizującej się sieci Kohonena polega na tym, że uczeniu podlega nie tylko neuron m* wygrywający w konkurencji z innymi neuronami sieci, ale także neurony które z nim sąsiadują.

Formalnie regułę można zapisać wzorem:

$$\omega_i^{(m^*)(j+1)} = \omega_i^{(m^*)(j)} + \eta^{(j)}\, h(m,m^*)\, (x_i^{(j)} - \omega_i^{(m)(j)})$$

formuła uczenia może być zapisana w formie:

$$\omega_i^{(m^*)(j+1)} = \omega_i^{(m^*)(j)} + \eta^{(j)}\, x_i^{(j)}\ (2\, y_m^{(j)} - 1)$$

Przypomnienie: uczenie Kohonena

Funkcjonowanie powyższego wzoru w istotny sposób oparto na fakcie, że $y_m^{(j)} \in \{0,1\}$.

Wzór ten nazywamy *regułą Hebb/Anti-Hebb*.

Funkcje h(m,m*) można definiować na wiele różnych sposobów, na przykład:

$$h(m,m^*) = \begin{cases} 1 & \text{dla } m=m^* \\ 0.5 & \text{dla } |m-m^*| =1 \\ 0 & \text{dla } |m-m^*| > 1 \end{cases}$$

$$h(m,m^*) = 1/\rho(m,m*)$$

$$h(m,m^*) = \exp (- [\rho(m,m*)]^2)$$

Sieci CP

Podstawowym założeniem przy stosowaniu sieci CP jest normalizacja sygnałów wejściowych. Każdy wektor X wprowadzany do systemu musi spełniać warunek $||X|| = 1$ (czyli $X^T X = 1$). Normalizacja X jest dokonywana poza siecią neuronową.

Normalizacja wejść jest potrzebna ze względu na element konkurencji (rywalizacji) występujący w pierwszej warstwie sieci CP. Do dalszego przetwarzania w kolejnej warstwie sieci przesyłany jest zaledwie jeden sygnał, pochodzący od tego elementu warstwy pierwszej który był najbardziej optymalnie dopasowany do przedstawionego sygnału wejściowego X.

Pierwsza warstwa sieci CP jest warstwą realizującą algorytm Kohonena.

Sieci CP

Pierwsza warstwa sieci CP jest warstwą realizującą algorytm Kohonena.

Wektory wejściowe X mnożone są przez wektory wag W_j poszczególnych neuronów sieci dostarczając wartości e_j będących sumarycznym (ważonym) pobudzeniem każdego neuronu.

$$e_j = W_j^T X$$

a następnie wybierany jest element o największej wartości pobudzenia e_j *("zwycięzca")* i tylko jego sygnał wyjściowy przyjmuje wartość 1. To jest właśnie ten tytułowy *counter* zastępujący i symbolizujący wszystkie sygnały wejściowe.

Zasada pierwszej warstwy sieci CP

Zasada działania sieci Kohonena zakłada, że sygnał wejściowy e_j każdego neuronu jest miarą stopnia podobieństwa pomiędzy aktualnym sygnałem wejściowym X, a abstrakcyjnym wzorcem sygnału, na którego wykrywanie wytrenowany jest j-ty neuron.
Ten wzorzec idealnego sygnału dla j-tego neuronu zawarty jest w jego wektorze wag W_j.

Jeżeli:

$X = W_j$ neuron odpowiada sygnałem o maksymalnej wartości

$X \neq W_j$ wówczas e_j jest miarą kąta pomiędzy wektorami X i W_j

$$ej = W_j^T X = ||W_j^T|| \, ||X|| \cos(\alpha)$$

miara ta jest wiarygodna jeżeli $||W_j^T|| = 1$ oraz $||X|| = 1$

Zasada pierwszej warstwy sieci CP

Przykład sieci CP ilustrujący wady jej działania:

$W^T_1 = [1\ 2\ 3]$
$W^T_2 = [0\ 1\ 0]$

$X = [1\ 2\ 3]$ to $e_1 = 14$, $e_2 = 2$ $\Rightarrow$ wgrywa neutron 1
$X = [0\ 1\ 0]$ to $e_1 = 2$, $e_2 = 1$ $\Rightarrow$ wygrywa neuron 1 ŹLE

Błąd pojawił się ponieważ wejścia X oraz wektor wag nie były znormalizowane.

Zasada drugiej warstwy sieci CP

Druga warstwa sieci realizuje algorytm *Outstar* Grossberga.

Jeżeli oznaczymy, że sygnały wejściowe do tej warstwy tworzą wektor K, a sygnał wyjściowy Y obliczany jest wg. klasycznej reguły $Y = V K$, gdzie macierz współczynników wagowych V składa się z transponowanych wektorów V_i odpowiadających zestawom wag kolejnych neuronów warstwy wyjściowej.
Z formalnego punkty widzenia, sygnał z neuronów warstwy wyjściowej ma postać

$$y_i = \sum_{j=1}^{m} v_{ij} k_j$$

gdzie m ma z reguły dużą wartość. W praktyce tylko jeden element wektora K ma wartość 1, pozostałe maja wartość 0 i wystarcza utożsamić wyjścia y_j z pewnym współczynnikiem v_{ij}. Na wszystkich wyjściach pojawiają się tylko te wartości v_{ij} które odpowiadają numerowi j dla których $k_j=1$.

Zauważmy, że działanie to przypomina odczyt gotowej tabeli.

Uczenie pierwszej warstwy sieci CP

Uczenie sieci CP przebiega równocześnie w obu warstwach sieci. Jest to proces *uczenia z nauczycielem*, wraz z każdym wektorem wejściowym X podany jest wektor wyjściowy, jaki użytkownik chce uzyskać z sieci.

$\Rightarrow$ przy uczeniu pierwszej warstwy stosuje się technikę Kohonena, która jest formą uczenia bez nauczyciela

$\Rightarrow$ przy uczeniu drugiej warstwy stosuje się algorytm Grossberga do bezpośredniego wymuszania pożądanych odpowiedzi sieci.

Zgodnie z regułą Kohonena uczenie przebiega następująco:

Uczenie pierwszej warstwy sieci CP

Na k-tym kroku pokazuje się wektor wejściowy $X^{(k)}$, a dysponując (z wcześniejszych kroków procesu uczenia) wartościami wszystkich wektorów $W_j^{(k)}$ można obliczyć wszystkie wartości $e_j^{(k)}$

$$e_j^{(k)} = W_j^{(k)T} X^{(k)} \quad , j=1,2,...,m$$

oraz wyznaczyć numer "zwycięskiego" neuronu (tzn. tego, dla którego zachodzi)

$$\forall \; (j \neq z) \qquad e_z^{(k)} > e_j^{(k)}$$

Korekcie podlegają wyłącznie wagi "zwycięskiego"neuronu według reguły

$$W_z^{(k+1)} = W_z^{(k)} + \eta_1 (X^{(k)} - W_z^{(k)})$$

współczynnik uczenia η_1 jest przyjmowany początkowo jako 0.7, potem stopniowy zmniejszany.

Uczenie pierwszej warstwy sieci CP

Przy realizacji metody Kohonena najważniejsze są pierwsze kroki.

Najpierw należy nadać współczynnikom wagowym w_{ij} wartości początkowe.

Należy zapewnić unormowanie wszystkich wag $||W_j^{(1)}|| = 1$ oraz wskazane jest takie dobranie kierunków, by w sposób równomierny rozkładały się na powierzchni jednostkowej w przestrzeni n-wymiarowej.

To nie jest takie proste w realizacji.

Uczenie pierwszej warstwy sieci CP

Technika "*convex combination method*".

Początkowo, wszystkim składowym wszystkich wektorów wag nadaje się te sama wartość początkowa

$$\omega_{ij}^{(1)} = \text{sqrt}(1/n)$$

W procesie uczenia jako wektory wejściowe podajemy

$$x_i^{(k)\prime} = \eta_2(k)\, x_i^{(k)} + [1 - \eta_2(k)]\, \text{sqrt}(1/n)$$

gdzie

$\eta_2(k)$ – funkcja adaptująca, która dla małych k przyjmuje małe wartości a potem stopniowo rośnie do wartości 1 i tą wartość zachowuje podczas całego procesu uczenia.

Uczenie drugiej warstwy sieci CP

Uczenie drugiej warstwy sieci jest wykonywane wg. następującej reguły:

$$v_{ij}^{(k+1)} = v_{ij}^{(k)} + \eta_3 (y_i - z_i) k_j$$

Ponieważ tylko jedna wartość k_j jest różna od zera i w każdym kroku procesu uczenia korygowane są tylko te wagi, które łączą poszczególne neurony wyjściowej warstwy z jednym tylko – "zwycięskim" elementem poprzedniej warstwy.

Ta zasada (zwana regułą "outstar") znacznie zmniejsza pracochłonność procesu uczenia.

Parametr η_3 wybiera się "ostrożnie" tak, aby proces uczenia nie spowodował wpisania do "look-up" tablicy błędnych wartości.

Sieci CP

Sieć CP ***"potrafi uogólniać i kojarzyć dostarczone jej informacje".***

W rozbudowanej wersji jest ona dość chętnie i z powodzeniem stosowana.

Doskonale zdają egzamin jako systemy klasyfikacji i rozpoznawania obrazów, są wykorzystywane w automatyce i robotyce, są cenione jako systemy do redukcji ilości przesyłanych informacji (transmisji obrazów).

Autoasocjacyjna siec CP

Wersja autoasocjacyjna oznacza, że sieć nauczona realizacji odwzorowania

$$X \Rightarrow Y$$

może również realizować odwzorowanie

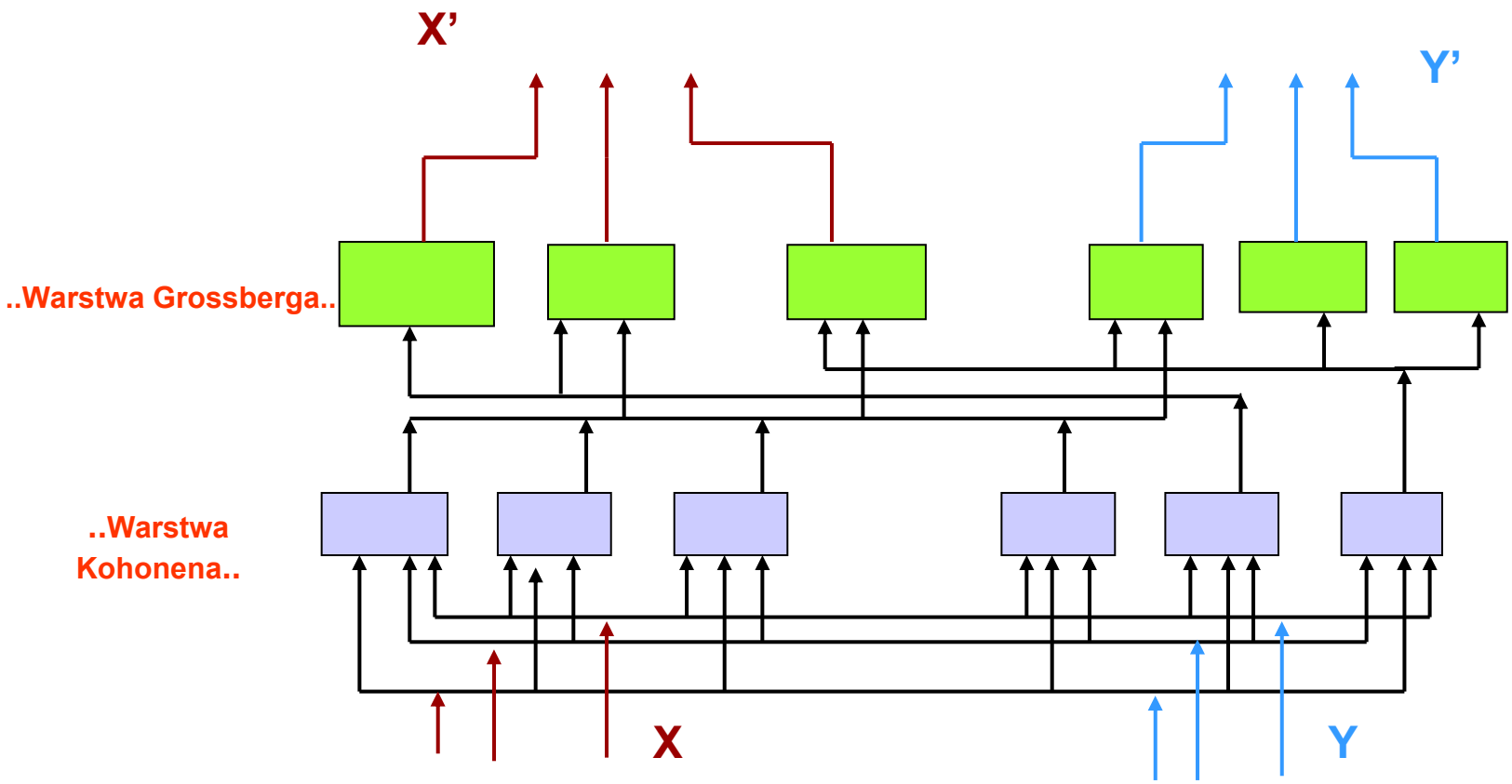
$$Y \Rightarrow X$$

Uczenie sieci polega na tym że na wejście podaje się X, Y (jako sygnały wejściowe) na wyjściu oczekuje się również X,Y.

Sieć uczy się realizacji odwzorowania tożsamościowego.

Eksploatuje się sieć podając tylko sygnał $X^{(k)}$ (wejścia Y bez sygnału), na wyjściu otrzymuje się odtworzony $X^{(k)}$ oraz również $Y^{(k)}$ który na etapie uczenia był kojarzony z $X^{(k)}$.

Autoasocjacyjna siec CP



Autoasocjacyjna sieć CP

Ponieważ na etapie uczenia sygnały X, Y są całkowicie równoprawne, sieć potrafi także odtwarzać odwzorowanie odwrotne. Wystarczy na wejściu podać sygnał $Y^{(k)}$, pozostawiając wejścia $X^{(k)}$ bez sygnału, na wyjściu otrzymamy $X^{(k)}$ oraz oczywiście odtworzony przez sieć $Y^{(k)}$.

Sieci CP znalazły liczne zastosowania:

$\Rightarrow$ Doskonale zdają one egzamin jako systemy klasyfikacji i rozpoznawania obrazów,

$\Rightarrow$ Są wykorzystywane w automatyce i robotyce do sterowania określonych systemów,

$\Rightarrow$ Są bardzo cenione jako systemy służące do przesyłania informacji – na przykład podczas transmisji obrazów.

Zestaw pytań do testu

1. Co to znaczy wsteczna propagacja błędów?
2. Podaj znane ci nazwy metod do uczenia wg. techniki propagacji wstecznej
3. Czy metody wstecznej propagacji błędów stosujemy do uczenia sieci liniowych?
4. Co to jest sieć CP?
5. Na czym polega uczenie Kohonena.
6. Czy sieć CP może być autoasocjacyjna?
7. Na czym polega uczenie autoasocjacyjnej sieci CP?