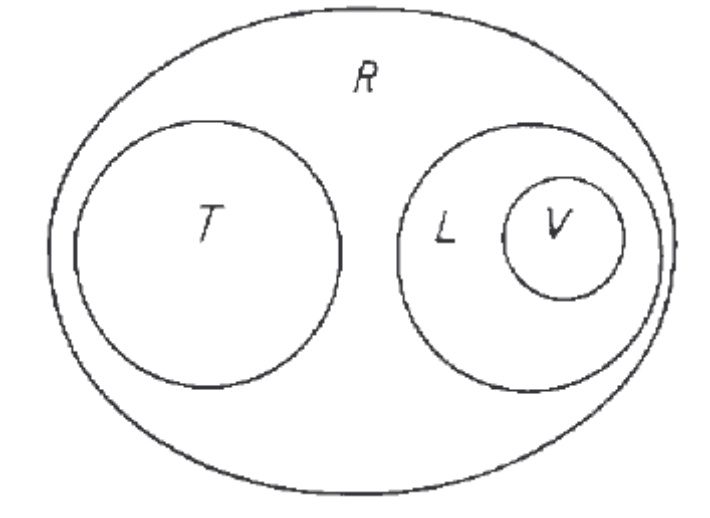
Widzenie komputerowe

Uczenie maszynowe na przykładzie sieci neuronowych (3)

źródła informacji:

S. Osowski, „Sieci neuronowe w ujęciu algorytmicznym”, WNT 1996

Zdolność uogólniania sieci neuronowej



R – oznaczenie reguły

L – zbiór uczący

V – zbiór walidujący (sprawdzający poziom nauczenia sieci)

T – zbiór testujący

Miarą zdolności uogólniania jest zdolność do generowania właściwych rozwiązań dla danych należących do zbioru *T*, na których sieć nigdy nie była trenowana.

Założenie: zarówno elementy zbioru *L* jak i *T* są typowymi reprezentantami tworzącymi regułę *R*.

Najczęściej stosowana miara: VCdim (Vapnika-Chervoneskisa; 1992 r.):

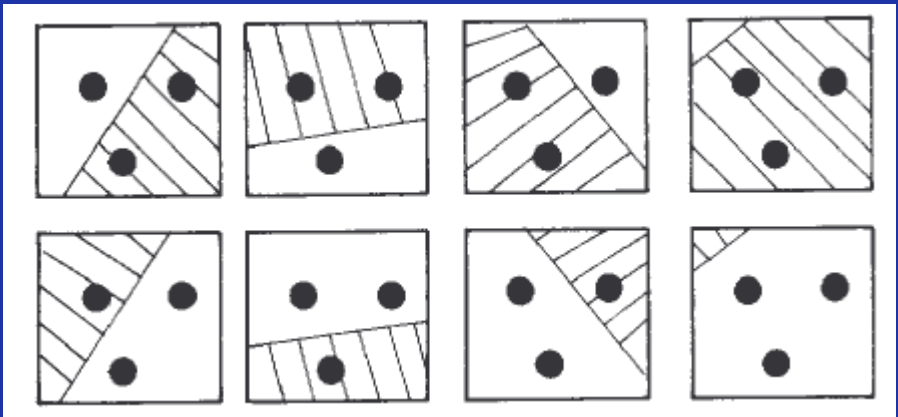
– liczebność największego zbioru S danych wzorców, dla których system może zrealizować wszystkie możliwe 2^n podziały zbioru S na 2 części za pomocą linii

W ogólności, dla neuronu o N wejściach miara VCdim wynosi N+1. Miara ta określa więc maksymalną liczbę danych uczących, które mogą być bezbłędnie odtworzone we wszystkich możliwych konfiguracjach

Przykład

Dla neuronu o dwóch wejściach n=3.

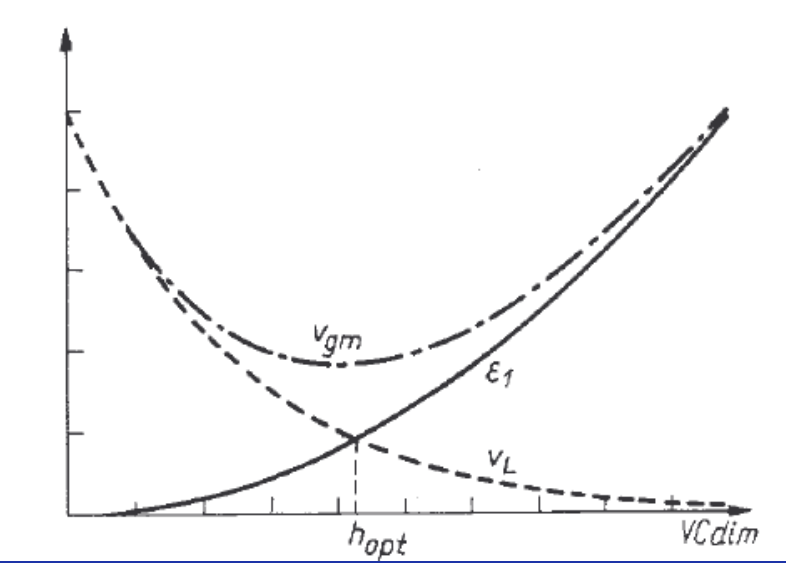
Uzasadnienie:



Zbiór złożony z 3 danych uczących jest największym zbiorem, w którym można przeprowadzić podział na dwie liniowo separowalne grupy na 2^3 sposobów.

Dodanie kolejnej próbki uczącej powoduje, że neurony nie są w stanie zrealizować wszystkich 2^4 podziałów, liniowo odseparowanych.

Wykres błędu uczenia i uogólniania w funkcji miary VCdim:



v_L – błąd uczenia

ε_I – przedział ufności

v_{gm} – ich superpozycja (błąd uogólniania)

h_{opt} – optymalna miara VCdim (optymalna liczba danych uczących)

Trudności w określaniu optymalnego rozmiaru zbioru uczącego

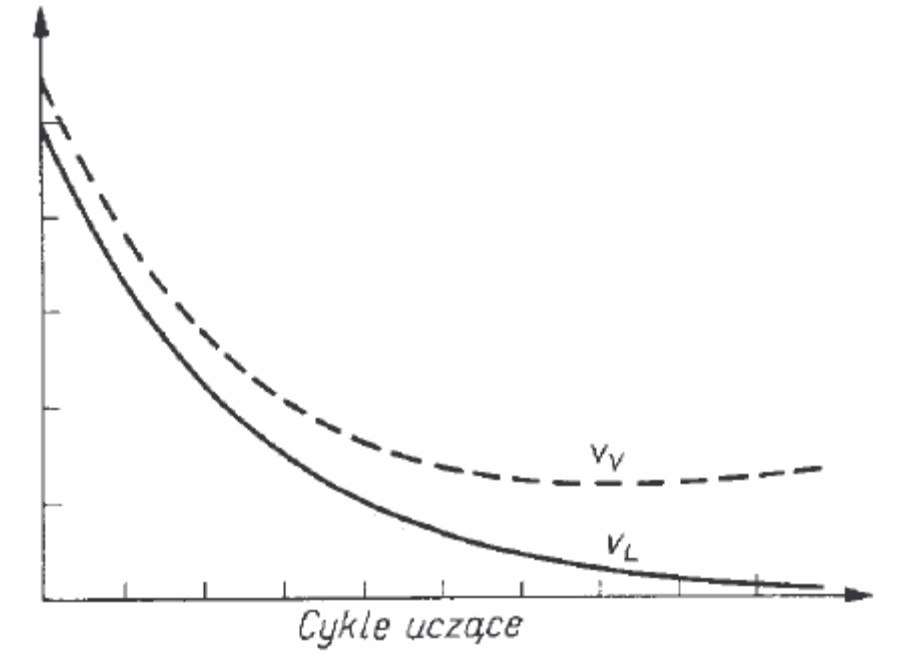
– trudno dokładnie określić miarę VCdim – często jest ona jedynie oszacowywana; ogólnie – jej wartość wzrasta przy wzrastającej liczbie wag

– w praktyce należy więc ograniczać liczbę neuronów ukrytych oraz liczbę powiązań międzyneuronowych

Wpływ sposobu i uczenia sieci na zdolność uogólniania

– ogólnie: błędy uczenia i testowania maleją w czasie (przy ustalonej liczbie próbek i mierze VCdim)

– taka sytuacja trwa tylko do pewnego momentu, od którego błąd testowania pozostaje stały a nawet rośnie



V_V – błąd testowania

V_L – błąd uczenia

Sytuacja taka zwykle wynika z ograniczonej liczby próbek uczących, które mogą być też niedoskonałe.

Wniosek:

Proces uczenia zwykle przeplatamy z procesem sprawdzania, jak dalece jest zaawansowany proces uczenia. Z tego względu dane uczące dzieli się zwykle na część podstawową L i uczącą V. Zbiór V jest testerem, wskazującym na moment, w którym należy zakończyć proces uczenia.

Dobór optymalnej architektury sieci

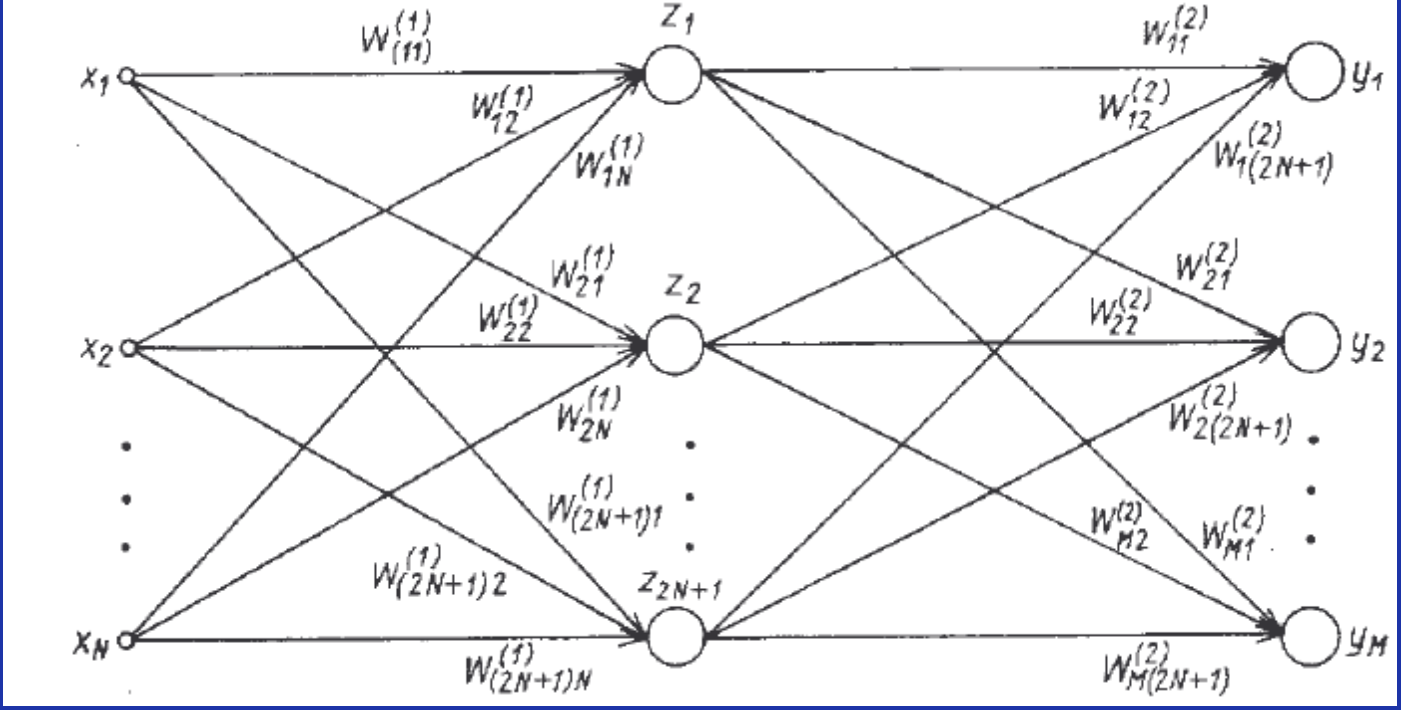
– wybór liczby warstw sieci, neuronów w warstwie i połączeń międzyneuronowych

– warstwa wejściowa: N – wymiar wektora danych x.

– warstwa wyjściowa: M – wymiar wektora zadanego d.

– pozostaje więc dobór warstw ukrytych i neuronów w każdej tej warstwie

Podstawy teoretyczne do określenia architektury sieci daje tzw. twierdzenie Kołmogorowa (1957, 1991 r.). Twierdzenie to mówi, że skuteczną aproksymację N-wymiarowego zbioru wejściowego x w M-wymiarowy zbiór wyjściowy d, można uzyskać przy użyciu sieci neuronowych już przy jednej warstwie ukrytej, zawierającej $2N+1$ neuronów.



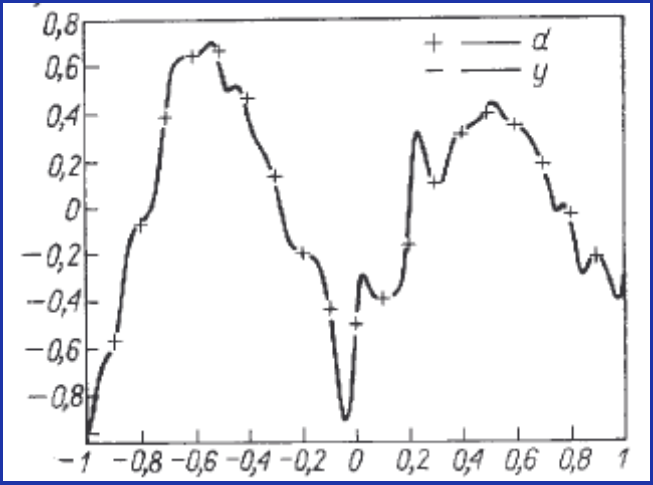
Uwaga: Twierdzenie Kołmogorowa udowadnia, że aproksymacja jest możliwa przy 2N+1 neuronach w warstwie ukrytej, ale nie mówi, że jest to architektura optymalna!

Ogólne wskazówki:

– istotny jest stosunek próbek uczących do miary VCdim (określającej złożoność sieci)

– mała liczba próbek uczących przy ustalonej mierze VCdim oznacza bardzo dobre dopasowanie sieci do próbek uczących, ale złe uogólnienie (w procesie uczenia nastąpił nadmiar parametrów dobieranych wag względem dopasowywanych wartości); może nastąpić niepotrzebne odtworzenie wszelkich nieregularności i szumów danych uczących; funkcja jest dobrze dopasowana jedynie w zadanych punktach uczących.

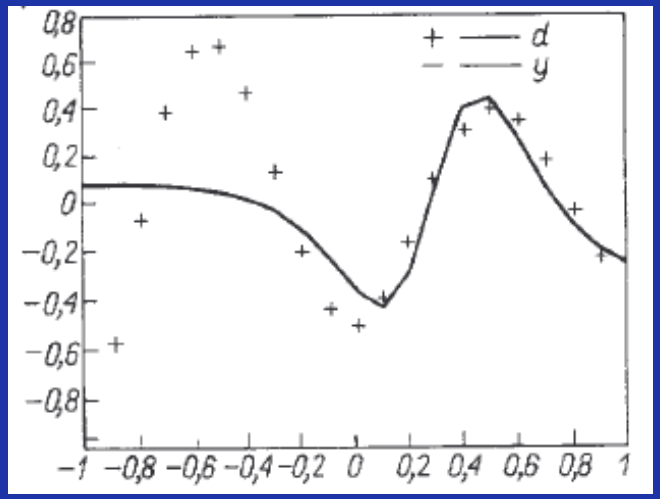
Ilustracja: zbyt duża liczba neuronów (20 neuronów, 40 próbek uczących):



Przeuczenie (przewymiarowanie) sieci

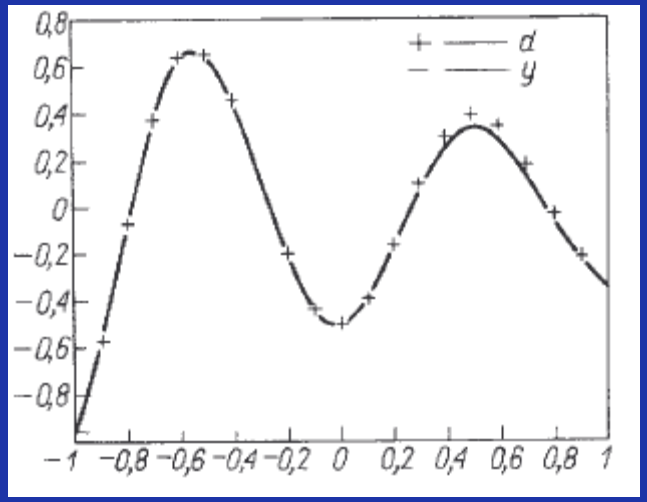
– zbyt duża liczba próbek uczących (zbyt mała liczba neuronów) powoduje problemy z dopasowaniem odpowiedzi już na etapie uczenia; liczba wag jest zbyt mała, żeby spowodować dobre dopasowanie.

Ilustracja: zbyt mała liczba neuronów (3 neurony, 40 próbek uczących):



Niedouczenie (niedowymiarowanie) sieci

Prawidłowy dobór wielkości sieci do wielkości próbek (5 neuronów, 40 próbek uczących):



Mały zarówno błąd uczenia, przy tym dobra zdolność uogólniania

Oszacowanie górnego i dolnego zakresu miary VCdim:

$$2\left[\frac{K}{2}\right]N \le VC\dim \le 2N_w\left(1+\log N_n\right)$$

[ ] – część całkowita liczby, N – wymiar wektora wejściowego; K – liczba neuronów w warstwie ukrytej;

N_w – całkowita liczba wag sieci; N_n – całkowita liczba neuronów w sieci

– dolny zakres: ≅ liczba wag między warstwą wejściową a ukrytą, górny zakres: ponad 2 x liczba wag między wszystkimi neuronami w sieci

Doświadczenie mówi, że próbek powinno być ok. 10–20 razy więcej niż wynosi miara VCdim.

Proces uczenia sieci

Kontynuuj

1. uczenie na podstawie zbioru treningowego L
2. testowanie na zbiorze V

Dopóki błąd testowania na danych ze zbioru V nie zacznie wzrastać, albo uzyskano minimum funkcji celu

Takie manipulowanie danymi uczącymi jednak często nie wystarcza! Trzeba aktywnie wpływać na strukturę sieci, dostosowując jej rozmiar do danego problemu.

Zwykle używa się sieci z jedną warstwą ukrytą. Jak konkretnie dobrać liczbę neuronów w tej warstwie?

1. Zakładamy wstępną liczbę neuronów na podstawie teorii Kołmogorowa ($2N + 1$ neuronów w warstwie ukrytej) bądź własnego doświadczenia; zwykle dobrym punktem startu jest też wartość

$$K \approx \sqrt{NM}$$

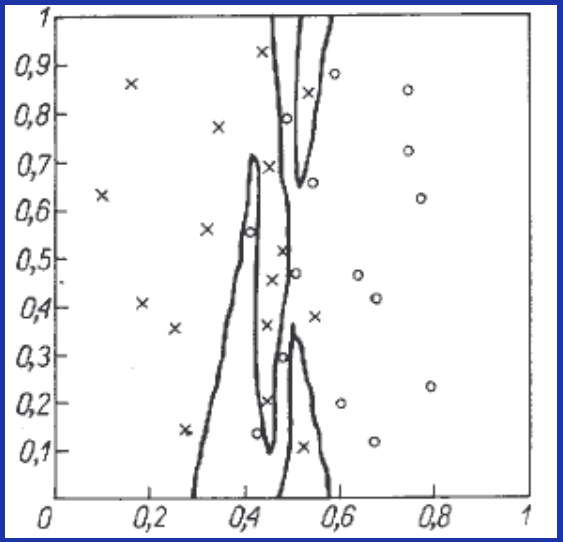
a następnie przeprowadzamy ew. redukcję podczas uczenia

2. Startujemy od minimalnej (nawet zerowej) liczby neuronów ukrytych, stopniowo je dodajemy aż do uzyskania dobrego wytrenowania sieci (czego miarą jest np. zdolność uogólniania na podzbiorze V).

Redukcja sieci

– jej zadaniem jest zmniejszenie liczby wektorów ukrytych i liczby powiązań międzyneuronowych

Przykład (klasyfikacja wzorca):

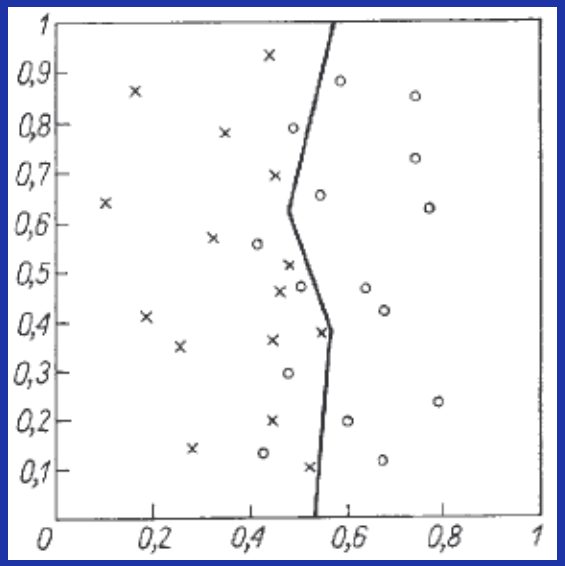


Struktura 2-50-10-1 (wejścia, warstwa ukryta, wyjścia)

Sieć przeuczona (przewymiarowana)

Wagi dopasowały się do nieistotnych szczegółów klasyfikacji

Zła zdolność uogólniania



Właściwie dobrana liczba neuronów, sieć dokonała prawidłowej klasyfikacji danych wejściowych na dwie klasy (gładka granica)

Struktura 2-2-1 (2 wejścia, 2 neurony w warstwie ukrytej, 1 wyjście)

Dobra zdolność uogólniania

Najprostsza metoda redukcji: próby dla różnych sieci o różnej liczbie neuronów ukrytych i wybranie tej o zadowalającym wyniku uczenia

– bardzo czasochłonna i najmniej efektywna

– decyzja może być podjęta dopiero po pełnym sprawdzeniu wielu struktur

– nie daje pewności, co do wyników

Inne, efektywniejsze metody:

– szacowanie wrażliwości funkcji celu względem wagi lub neuronu; wagi o najmniejszej wrażliwości (najmniej wpływające na wartość funkcji celu) są z sieci usuwane, a proces uczenia jest kontynuowany dla zredukowanej sieci

– wprowadzamy dodatkowy współczynnik do każdej wagi, α_j, który może przyjmować wartości 1 lub 0

$$y_i = f\left(\sum_j W_{ij}\alpha_j y_j\right)$$

– obliczamy współczynnik wrażliwości funkcji celu względem $\alpha_j=1$ (Mozer, 1989):

$$\rho_j = -\frac{\partial E}{\partial \alpha_j}$$

– obcinamy wagę (kładąc $\alpha_j=0$) jeżeli współczynnik ten jest mały

– kary za nieefektywną strukturę; w praktyce wprowadza się do funkcji celu składniki faworyzujące małe amplitudy wag, zmuszając algorytm do ich ciągłej redukcji; może być to nieefektywne, ponieważ małe wagi nie zawsze oznaczają automatycznie mały wpływ na działanie sieci

– zmodyfikowana funkcja celu: $$E(W) = E^{(0)}(W) + \gamma\sum_{i,j} W_{ij}^2$$

W obu przypadkach ważne, żeby po obcięciu wag douczyć sieć (daje to bardo dobre rezultaty w postaci bardzo płaskiej funkcji uczenia)

OBD (Optimal Brain Damage) (1990 r.)

– jedna z metod oszacowania wpływu wagi na działanie sieci, oparta na rozwinięciu aktualnego rozwiązania w szereg Taylora

– jedna z najlepszych, powszechnie stosowana

Zmiana wartości funkcji celu pod wpływem zmiany wagi:

$$\Delta E = \sum_i g_i \Delta W_i + \frac{1}{2}\left[\sum_i h_{ii}\left[\Delta W_{ii}\right]^2 + \sum_{i \neq j} h_{ij}\Delta W_i \Delta W_j\right] + O\left(\|\Delta W\|^2\right)$$

gdzie $g_i = \frac{\partial E}{\partial W_i}$ $h_{ij} = \frac{\partial^2 E}{\partial W_i \partial W_j}$

Upraszczając (małe zmiany wag), otrzymujemy:

$$\Delta E \approx \frac{1}{2}\left[\sum_i h_{ii}\left[\Delta W_{ii}\right]^2 + \sum_{i \neq j} h_{ij}\Delta W_i \Delta W_j\right]$$

Szukamy takich wag, które jak najmniej zmienią wartość funkcji celu – liczmy parametr $S_{ij} = \frac{1}{2} h_{kk} W_{ij}^2$

Procedura:

1. Selekcja wstępna (wstępny wybór struktury warstw ukrytych)

2. Przeprowadzenie programu uczenia przy zastosowaniu dowolnej metody gradientowej

3. Określenie elementów diagonalnych hesjanu

$$h_{kk} = \frac{\partial^2 E}{\partial W_{ij}^2}$$

odpowiadających każdej wadze sieci

4. Obliczenie parametru

$$S_{ij} = \frac{1}{2} h_{kk} W_{ij}^2$$

określającego znaczenie danego połączenia synaptycznego dla działania sieci

5. Posortowanie wag wg. przypisanych im parametrów S_{ij} i obcięcie najmniejszych wartości

6. GOTO 2

Modyfikacja (1993 r.): OBS (Optimal Brain Surgeon) (zachowuje dotychczas osiągnięte minimum)

Procedura:

1. Przeprowadzenie procesu uczenia aż do uzyskania minimum funkcji celu

2. Obliczenie macierzy odwrotnej hesjanu, H^{-1}

3. Znalezienie wagi, która ma najmniejszą wartość współczynnika $L_i = \frac{1}{2}\frac{W_i^2}{[H^{-1}]_{ii}}$

Jeżeli zmiana wartości funkcji celu, towarzysząca obcięciu tej wagi jest dużo mniejsza niż E, to wagę się obcina i przechodzimy do punktu 4. W przeciwnym wypadku, przechodzimy do punktu 5.

4. Następuje modyfikacja wartości wag (po obcięciu tej i-tej) o wartości: $\Delta W = \frac{W_i}{[H^{-1}]_{ii}} H^{-1} e_i$

e_i – wektor jednostkowy

5. Koniec procesu redukcji

Algorytm propagacji wstecznej

Założenie: celem uczenia sieci jest określenie wag neuronów wszystkich warstw sieci w taki sposób, aby przy zadanym wektorze wejściowym *x*, uzyskać na wyjściu sieci wartości sygnałów wyjściowych *y* równające się z dostateczną dokładnością wartościom żądanym *d*.

Gdy funkcja celu jest ciągła, stosujemy zwykle gradientowe metody optymalizacyjne, w których:

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

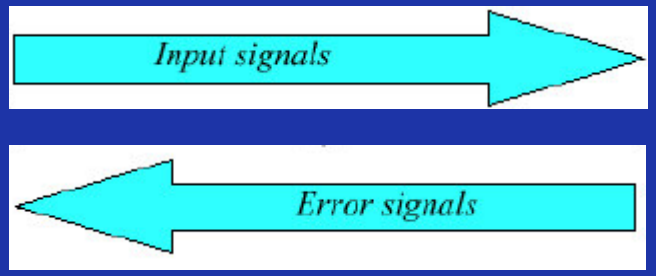
$$\Delta W = \eta p(W)$$

$p(W) = -\nabla E(W)$ – kierunek w przestrzeni wielowymiarowej W; E – funkcja celu.

Uczenie sieci wielowarstwowej wymaga określenia wektora gradientu względem wszystkich warstw sieci.

– dla warstwy wyjściowej wynik jest natychmiastowy

– dla innych warstw stosujemy algorytm propagacji wstecznej (ang. *backpropagation, bp*)



Ogólna zasada postępowania

1. Analiza sieci neuronowej o zwykłym kierunku przepływu sygnałów przy założeniu sygnałów wejściowych równych elementom aktualnego wektora x. Otrzymujemy wartości sygnałów wyjściowych neuronów warstw ukrytych oraz warstwy wyjściowe, a także odpowiednie pochodne funkcji aktywacji w poszczególnych warstwach:

$$\frac{df\left(u^{(1)}\right)}{du_i^{(1)}}, \frac{df\left(u^{(2)}\right)}{du_i^{(2)}}, \ldots, \frac{df\left(u^{(m)}\right)}{du_i^{(m)}}$$

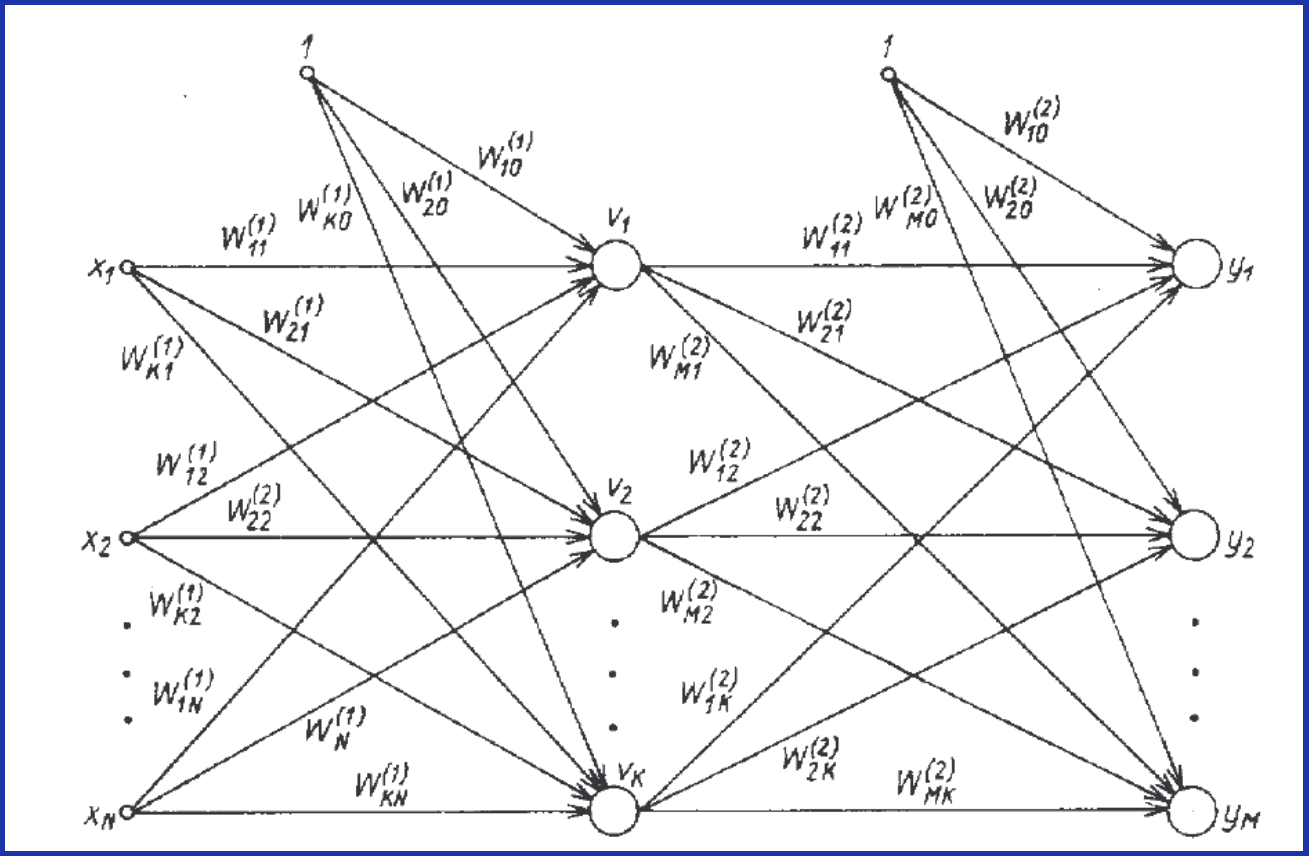
2. Utworzenie sieci propagacji wstecznej przez odwrócenie kierunku przepływu sygnałów, zastąpienie funkcji aktywacji przez ich pochodne oraz podanie do byłego wyjścia (obecnie wejścia) wymuszenia w postaci różnicy między wartością aktualną a żądaną. Dla tak utworzonej sieci należy obliczyć wartości odpowiednich różnic wstecznych.

3. Adaptacja wag (uczenie sieci) na podstawie wyników uzyskanych w punkcie 1 i 2 dla sieci zwykłej i sieci o propagacji wstecznej wg. odpowiednich wzorów.

4. Procesy 1–3 należy powtórzyć dla wszystkich wzorców uczących, aż do zatrzymania algorytmu (zwykle gdy norma gradientu spadnie poniżej pewnej, ustalonej wartości).

Wzory dla sieci o jednej warstwie ukrytej

Schemat oznaczeń:



N – liczba neuronów wejściowych

K – liczba neuronów w warstwie ukrytej

M – liczba neuronów w warstwie wyjściowej

Funkcja celu dla jednej pary uczącej:

$$E = \frac{1}{2}\sum_{i=1}^{M}(y_i - d_i)^2$$

dla wielu par uczących:

$$E = \frac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{M}(y_i(j) - d_i(j))^2$$

Zakładamy aktualizację wag po każdorazowej prezentacji próbki, otrzymując funkcję celu:

$$E = \frac{1}{2}\sum_{k=1}^{M}\left[f\left(\sum_{i=0}^{K} W_{ki}^{(2)} v_i \right) - d_k \right]^2 = \frac{1}{2}\sum_{k=1}^{M}\left[f\left(\sum_{i=0}^{K} W_{ki}^{(2)} f\left(\sum_{j=0}^{N} W_{ij}^{(1)} x_j \right) \right) - d_k \right]^2$$

(sumujemy od zera, włączając tzw. jednostkowy sygnał polaryzacji, dodając na początku każdego wektora *x* wartość 1)

Różniczkujemy funkcję celu, obliczając składniki gradientu dla warstwy wyjściowej:

$$\frac{\partial E}{\partial W_{ij}^{(2)}} = (y_i - d_i)\frac{df\left(u_i^{(2)}\right)}{du_i^{(2)}} v_j$$

$$u_i^{(2)} = \sum_{j=0}^{K} W_{ij}^{(2)} v_j$$

Oznaczmy

$$\delta_i^{(2)} = (y_i - d_i)\frac{df\left(u_i^{(2)}\right)}{du_i^{(2)}}$$

Teraz odpowiedni składnik gradientu względem wag neuronów warstwy wyjściowej zapisujemy:

$$\frac{\partial E}{\partial W_{ij}^{(2)}} = \delta_i^{(2)} v_j$$

Dla warstwy ukrytej otrzymujemy:

$$\frac{\partial E}{\partial W_{ij}^{(1)}} = \sum_{k=1}^{M} (y_k - d_k) \frac{dy_k}{dv_i} \frac{dv_i}{dW_{ij}^{(1)}}$$

$$\frac{\partial E}{\partial W_{ij}^{(1)}} = \sum_{k=1}^{M} (y_k - d_k) \frac{df\left(u_k^{(2)}\right)}{du_k^{(2)}} W_{ki}^{(2)} \frac{df\left(u_i^{(1)}\right)}{du_i^{(1)}}$$

Oznaczmy:

$$\delta_k^{(2)} = (y_k - d_k) \frac{df\left(u_k^{(2)}\right)}{du_k^{(2)}}$$

$$\delta_1^{(1)} = \sum_{k=1}^{M} \delta_k^{(2)} W_{ki}^{(2)} \frac{df\left(u_i^{(1)}\right)}{du_i^{(1)}}$$

otrzymując:

$$\frac{\partial E}{\partial W_{ij}^{(1)}} = \delta_i^{(1)} x_j$$

Problemy

jak nie wpaść w minima lokalne?

– dodawanie do wag małych, przypadkowych wartości

– dodawanie niewielkiego losowego zakłócenia do danych uczących

– podawanie wzorców w losowej kolejności

– kilkakrotne powtarzanie procesu uczenia z różnymi wagami początkowymi

jak dobrać współczynnik uczenia?

– wartości od 10^{-3} do 10 (w zależności od problemu); mniejsze wartości powodują zwolnienie algorytmu ale polepszenie jego stabilności

– stały lub zmienny

jak dobrać wagi początkowe?

– duże wagi początkowe: wolnozbieżny proces uczenia lub wręcz zatrzyma się w minimum lokalnym

– wagi bliskie zeru: zbiegają często do zera i później już się nie zmieniają lub też proces uczenia jest bardzo powolny

Zalecenie: losowy wybór wag, aby pobudzenie łączne było nieco mniejsze od jedności

Technika momentu (momentum)

– gdy gradient i współczynnik uczenia są niewielkie, zmniejszanie się błędu jest bardzo powolne

– gdy są z kolei duże, mogą wystąpić oscylacje wokół minimum

– metoda mementu służy do przyspieszenia zbieżności

Korekta wag:

$$\Delta w(k) = -\eta \nabla E(k) + \alpha \Delta w(k-1)$$

α – współczynnik momentu z przedziału (0; 1)

Działanie:

– jeżeli w kolejnych krokach gradienty wskazują ten sam kierunek, to ich działanie się kumuluje i przyrosty stają się coraz większe

– jeżeli w kolejnych krokach gradienty wskazują przeciwny kierunek, to ich działanie się częściowo znosi, to ruch punktu jest łagodnie hamowany (bez techniki momentu zwykle występują w takiej sytuacji oscylacje)

W praktyce najczęściej: $\alpha = 0.1$, $\eta = 0.9$