Wykład 3: Uczenie nadzorowane warstwowych sieci neuronowych

Sieci warstwowe złożone z neuronów nieliniowych

W przeciwieństwie do neuronów liniowych, łączenie neuronów nieliniowych w sieci ma sens - prowadzi do jakościowego zwiększenia możliwości adaptacyjnych tak powstałego systemu. Wielu odwzorowań realizowalnych przy pomocy nieliniowej sieci dwuwarstwowej nie da się osiągnąć pojedynczą warstwą neuronów.

Oznaczenia:

M liczba warstw

k indeks warstw (*k=0* – wejścia sieci)

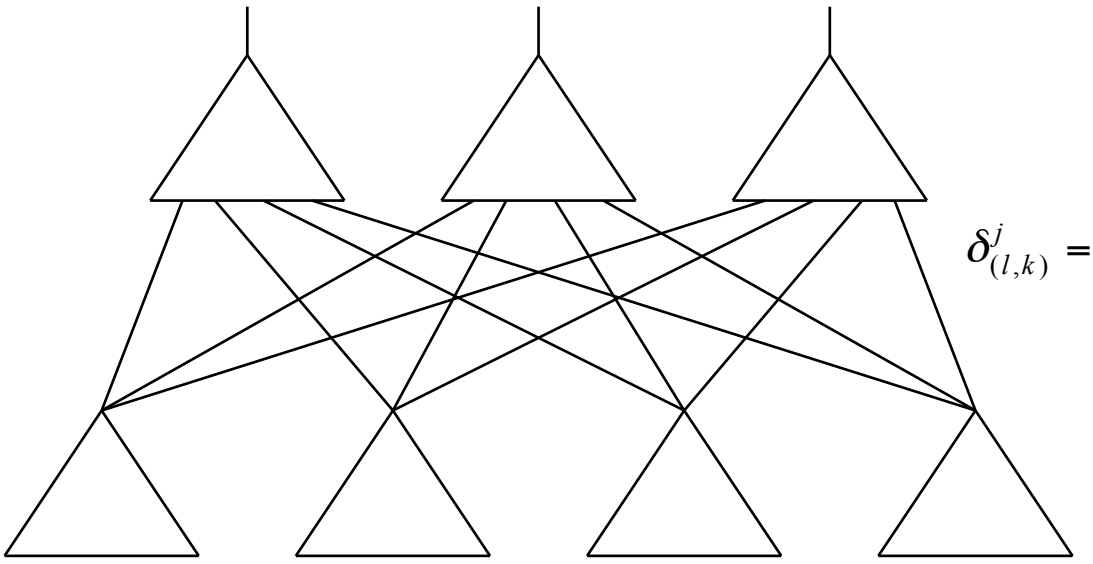
N_k liczba neuronów w k-tej warstwie, $k = 1 \ldots M$

(l,k) l-ty neuron w k-tej warstwie, $l = 1 \ldots N_k$

Problem:

Jak znaleźć błąd popełniany przez neurony z warstw ukrytych?

$$\delta^j_{(l,M)} = z^j_{(l,M)} - y^j_{(l,M)}$$

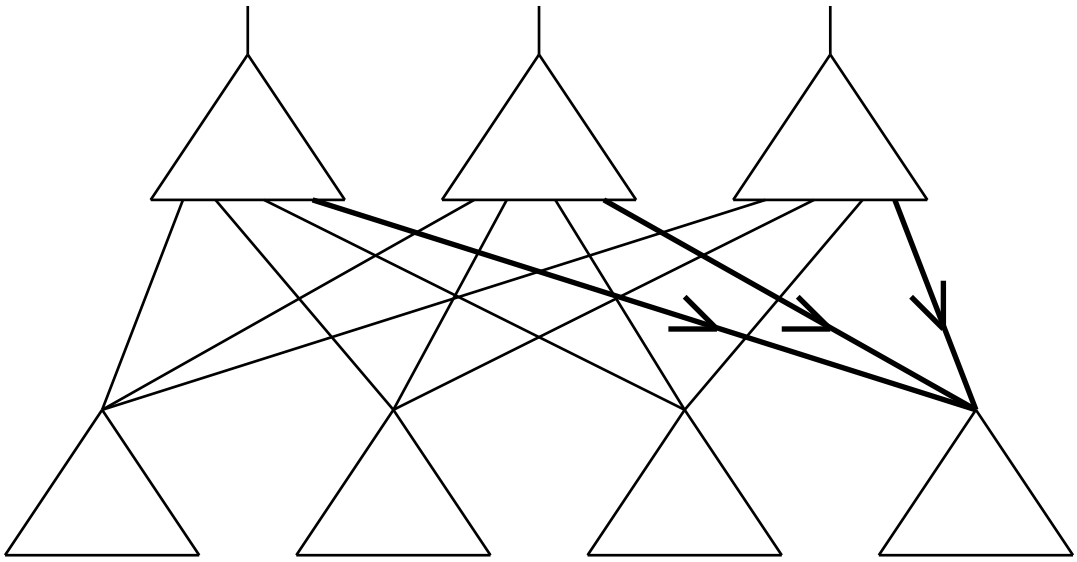


$$\delta^j_{(l,k)} = ?, k < M$$

Da się wykazać, że jeśli sieć jako całość ma stosować się do reguły największego spadku gradientu, to błędy neuronów w warstwach ukrytych muszą być zdefiniowane następująco [Rume86]:

$$\delta^{j}_{(k,l)} = \sum_{p=1}^{N_{l+1}} w^{j}_{k(p,l+1)} \delta^{j}_{(p,l+1)} \qquad ($$

Oznacza to, że błąd danego neuronu w l-tej warstwie jest równy sumie błędów popełnionych przez neurony z warstwy l+1-szej ważonych po wagach łączących ten neuron z neuronami tej warstwy. Potwierdza to intuicja: udział neuronu warstwy ukrytej w błędzie popełnionym przez następną warstwę musi być siłą rzeczy proporcjonalny do wag łączących go z "następnikami".



Natychmiastowym następstwem powyższej formuły jest tzw. algorytm wstecznego *progagacji/rzutowania błędu (backpropagation)*.

Algorytm wstecznej propagacji błędu (*backpropagation, BP*):

1. Podaj na wejście sieci kolejny wektor wymuszeń $\boldsymbol{x}$.
2. Przepropaguj wymuszenie przez sieć, obliczając pobudzenia neuronów w kolejnych warstwach, aż do warstwy wyjściowej.
3. Wektor wyjść otrzymany w warstwie wyjściowej $\boldsymbol{y}$ porównaj z wektorem uczącym/oczekiwanym $\boldsymbol{z}$ i oblicz na tej podstawie błędy δ popełnione przez neurony tej warstwy.
4. Zgodnie z formułą dokonaj wstecznej propagacji błędu do kolejnych warstw ukrytych, tj. do ostatniej, przedostatniej itd., aż do osiągnięcia warstwy wyjściowej.
5. Dla każdego neuronu w sieci dokonaj modyfikacji wartości wag stosownie do wielkości popełnionego błędu.
6. Sprawdź, czy błąd średniokwadratowy popełniany przez sieć dla wszystkich przykładów ze zbioru uczącego Q spadł poniżej zadanej wartości Q_{stop}; jeśli tak - zakończ pracę, w przeciwnym razie przejdź do kroku 1.

Autorstwo:
Bryson, Ho, 1969
Werbos, 1974
Parker, 1985
Rumelhart, Hinton, Williams: *Parallel distributed processing.* 1986

Zalety BP:
- **prostota:** w rzutowaniu błędu używa się tych samych współczynników co w propagacji pobudzenia, tj. wag;
- **lokalny charakter:** błąd δ neuronu w warstwach ukrytych zależy jedynie od błędów neuronów, do których przyłączone jest jego wyjście
  $\Rightarrow$ łatwość równoległych realizacji sprzętowych.

Interpretacja graficzna procesu uczenia

Przestrzeń konfiguracji

Rozpięta w tylu wymiarach, ile parametrów modyfikowanych podczas uczenia posiada sieć. Stan/konfiguracja sieci w danej chwili jest reprezentowana w tej przestrzeni jako punkt, który przemieszcza się w wskutek uczenia.

Uwaga!
Przestrzeni konfiguracji nie należy mylić z przestrzenią wzorców/przykładów.

Krajobraz energetyczny (energy landscape), inaczej *krajobraz funkcji błędu Q*

Każda sieć posiada funkcję energetyczną lub funkcję błędu, pozwalająca wyznaczyć energię lub błąd popełniany przez sieć w danej konfiguracji. Funkcję tę można przedstawić w postaci wykresu rozpiętego "ponad" przestrzenią konfiguracji (pojawia się dodatkowy wymiar). Przy takiej interpretacji działanie algorytmu polega na znajdowaniu minimum (najchętniej *globalnego*) w krajobrazie energetycznym.

Parametry reguły delta i algorytmu wstecznej propagacji błędu

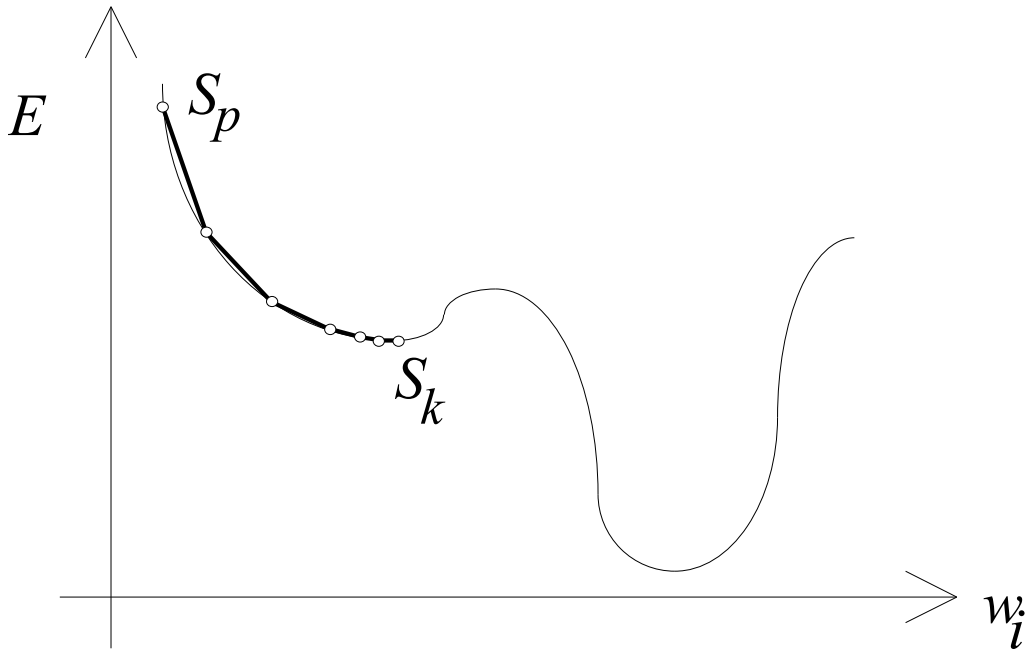
- początkowa konfiguracja *wektora wag* **w**:

  niewielkie wartości losowe (dotyczy to większości algorytmów uczących).

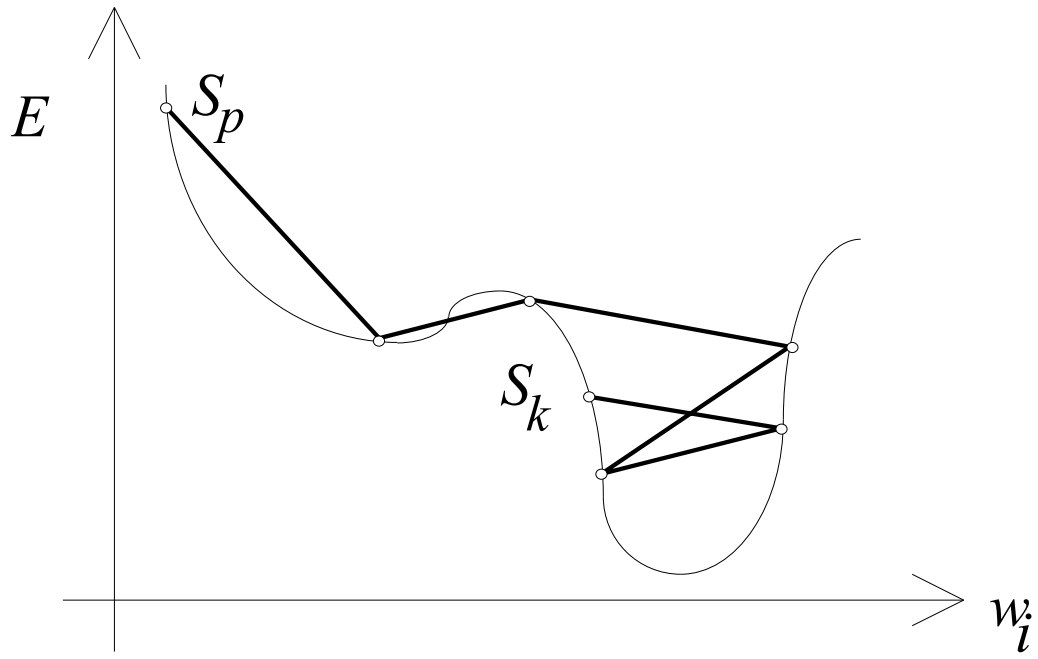
- *współczynnik prędkości uczenia* η

  Decyduje o wpływie błędu popełnianego przez neuron na korektę wartości wag. Właściwy dobór ma kluczowe znaczenie dla prędkości zbieżności algorytmu:

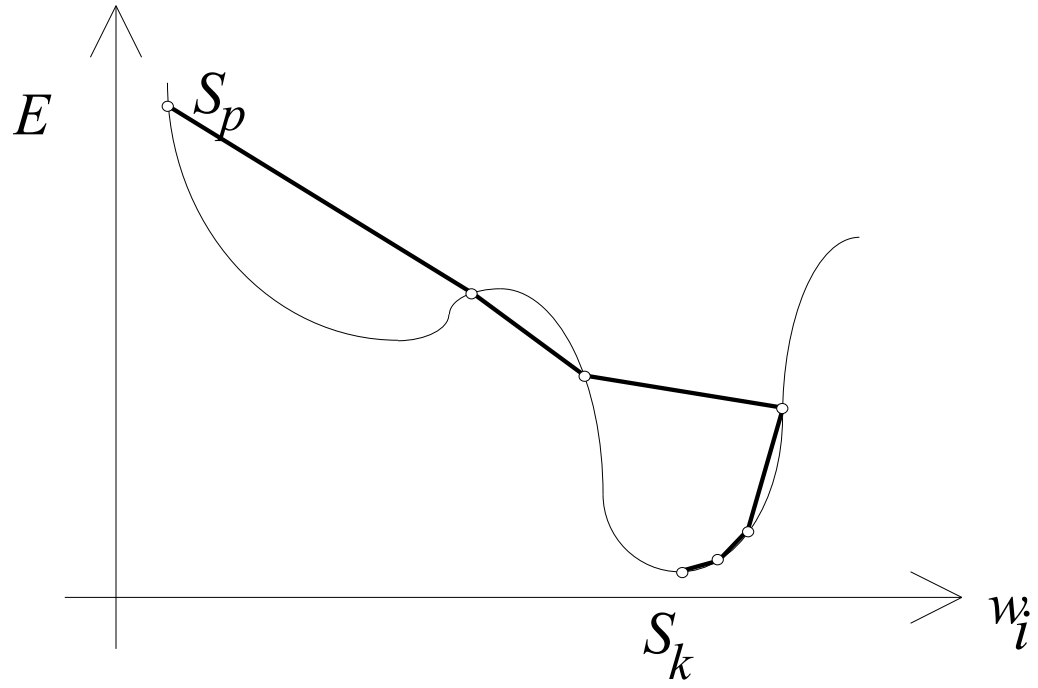
  - zbyt mała wartość spowalnia proces uczenia i zwiększa ryzyko wpadnięcia w pułapkę lokalnego minimum (punkt reprezentujący konfigurację sieci porusza się "małymi kroczkami" po krajobrazie energetycznym)



- przy zbyt dużej wartości istnieje ryzyko braku zbieżności:



- dynamiczna modyfikacja współczynnika η w trakcie uczenia (najczęściej duża wartość początkowa i stopniowe zmniejszanie)



Człon momentu (bezwładności)

Metoda największego spadku gradientu, opisana formułą (2.5):

$$\Delta w_i^j = -\eta \frac{\partial Q^j}{\partial w_i}$$

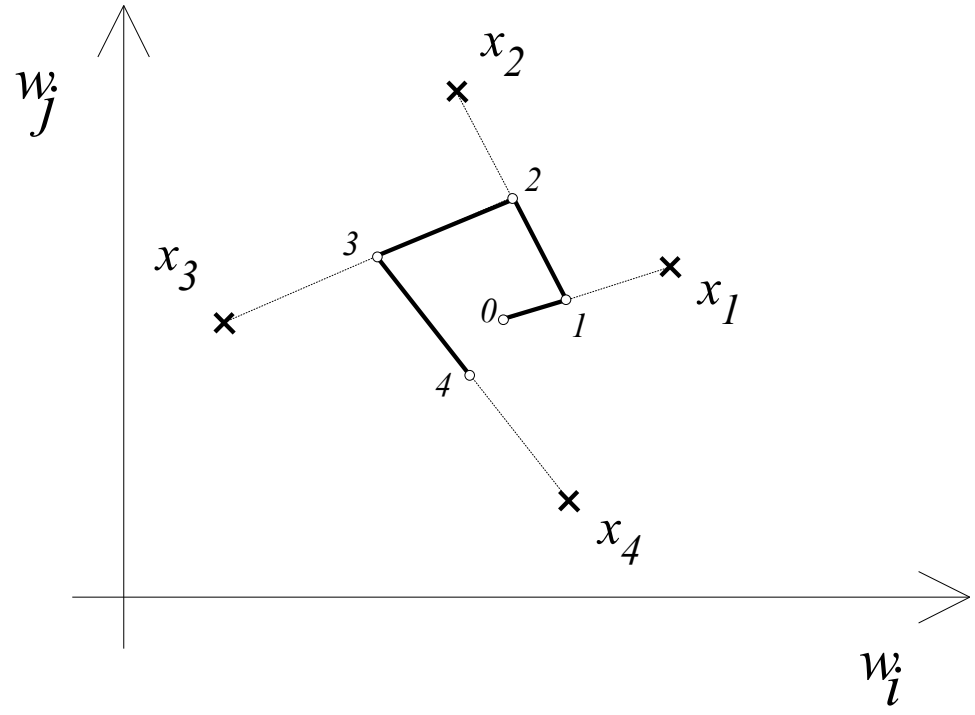
skłania *i*-tą wagę do zmiany wartości stosownie do bieżącej wartości gradientu w chwili *j*, bez względu na dotychczasowy przebieg uczenia. W wielu przypadkach powoduje to zbyt chaotyczne nadążanie wektora wag za wektorem pobudzeń; jest to szczególnie widoczne przy stosowaniu modyfikacji wag po prezentacji każdego wzorca.

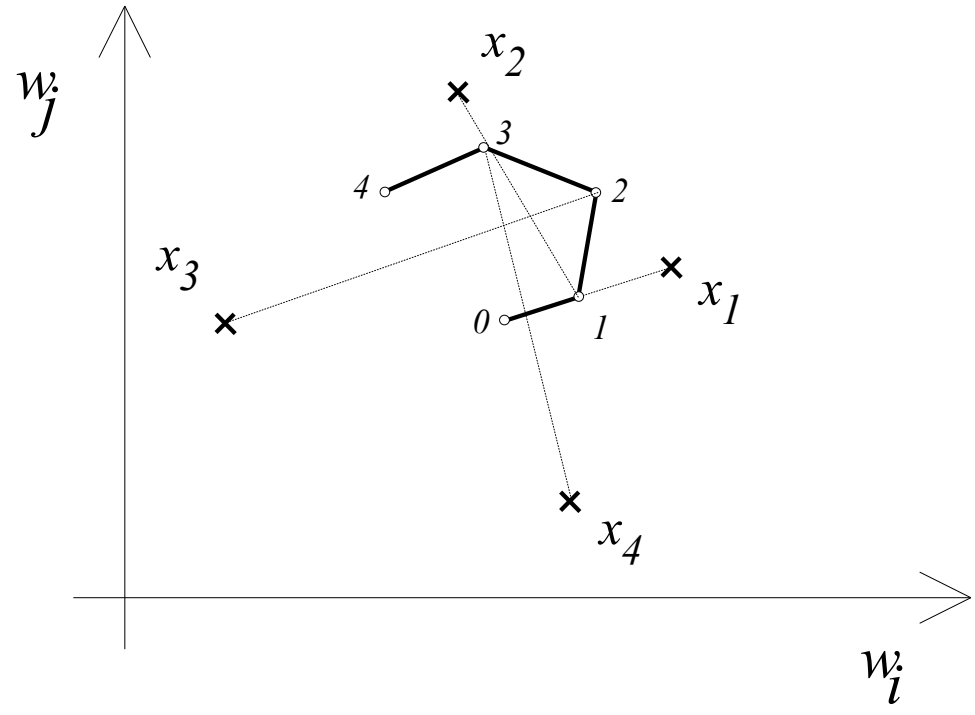
Dlatego formułę tę rozbudowuje się często o tzw. *człon bezwładności (momentum)*:

$$\Delta w_i^t = -\eta \frac{\partial Q^t}{\partial w_i} + \alpha \Delta w_i^{t-1} \qquad ($$

Uzależnia on (przez współczynnik α) wartość bieżącej modyfikacji wagi od modyfikacji przeprowadzonej w kroku poprzednim. Im większe α w stosunku do η, tym algorytm jest bardziej stabilny. Z reguły przyjmuje się α=0.9.

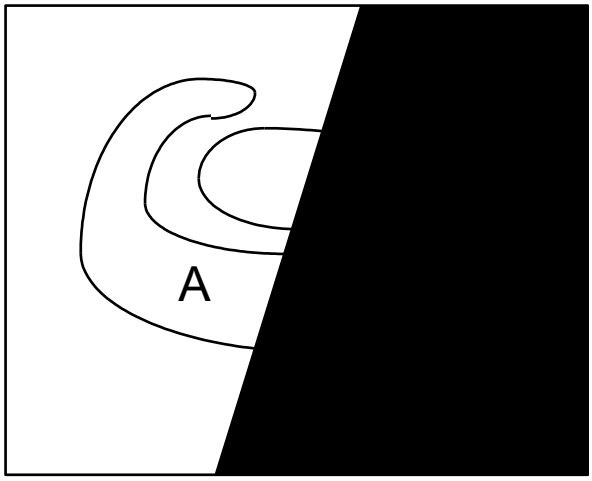
Porównanie procesu uczenia bez i z członem momentu

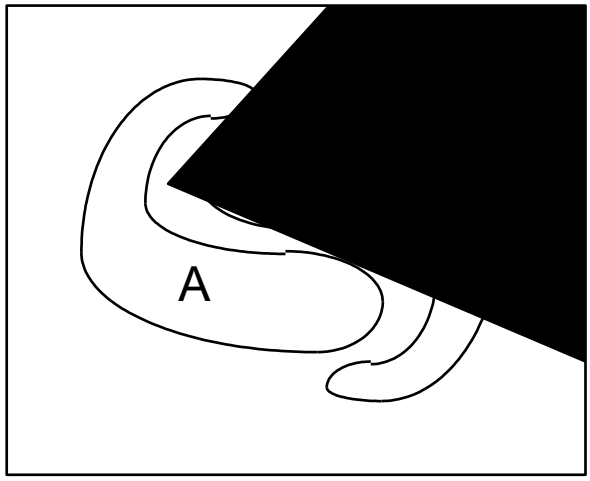




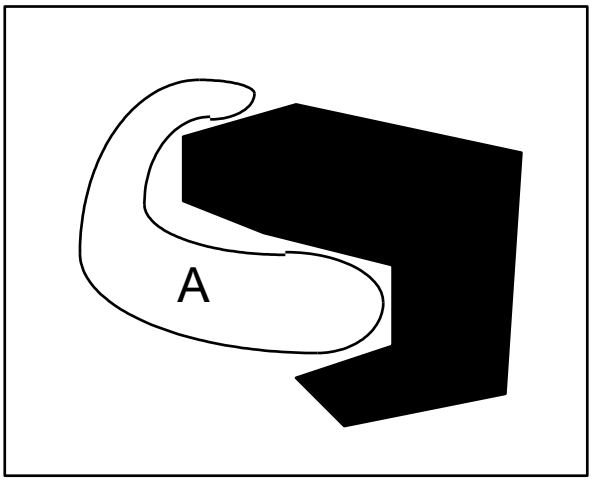
Problem doboru ilości warstw ukrytych

Pojedynczy neuron dzieli przestrzeń wejściową na dwie półprzestrzenie przedzielone (hiper)płaszczyzną.





Dwie warstwy neuronów są zdolne do wycięcia z przestrzeni wejściowej dowolnej wypukłej bryły.



Dodanie trzeciej warstwy neuronów umożliwia "sklejanie" ze sobą brył rozpoznawanych przez drugą warstwę, co daje możliwość definiowania dowolnych podziałów w przestrzeni wejściowej.

Problem doboru wielkości warstw ukrytych

Problem doboru rozmiaru pojedynczej warstwy pozostaje do dziś otwarty. Brak jednoznacznej reguły określającej optymalny rozmiar danej warstwy sieci przy danym zbiorze uczącym. Znane są jedynie ogólne zalecenia, podyktowane intuicją i doświadczeniem praktycznym:

- zbyt mała wielkość warstw czyni sieć niezdolną do adaptacji do zadanego zbioru przykładów/wymuszeń: w trakcie uczenia błąd średniokwadratowy utrzymuje dużą wartość
- zbyt duże warstwy wprowadzają ryzyko tzw. "uczenia na pamięć": dysponując dużą liczbą neuronów sieć "obejmuje" każdym z nich małą grupę przykładów (w skrajnym przypadku pojedynczy wzorzec), unikając bardziej kosztownego poszukiwania jakiejś generalizacji

Kiedy przeprowadzać modyfikację wag ?

Dwa modele

- tzw. *batch updating*:

  Przy prezentacji kolejnych przykładów poprawki wartości wag Δw są kumulowane. Co pewną liczbę prezentacji (z reguły równą rozmiarowi zbioru uczącego, tzw. *epoka/epoch*) wagi są modyfikowane przy pomocy tych skumulowanych poprawek.

- *modyfikacja przyrostowa*:

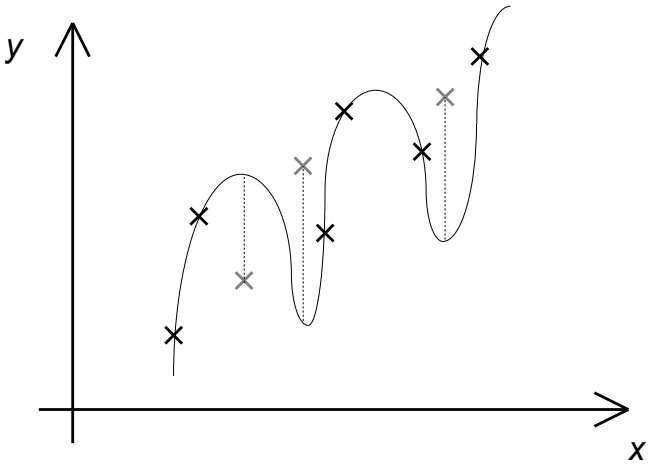
  Poprawki obliczone przy prezentacji wzorca są używane bezpośrednio (w tym samym kroku algorytmu) do modyfikacji wag.

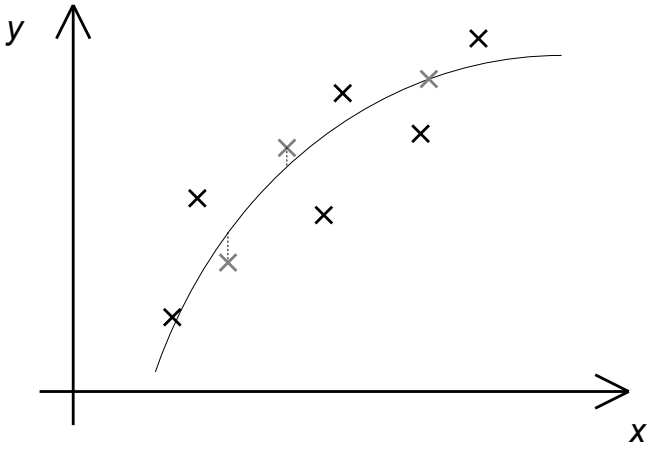
Powszechnie uważa się, że *batch updating* obciążone jest poważną wadą: podczas kumulacji poprawek wartości wag może zachodzić ich wzajemne znoszenie się. Wypadkowa poprawka może być nieznaczna, mimo że podczas prezentacji poszczególnych wzorców działanie neuronu obarczone było znacznym błędem. Powoduje to obniżenie "ruchliwości" procesu przeszukiwania przestrzeni wag, a co za tym idzie np. trudności z wyjściem z minimum lokalnego.

Modyfikacja przyrostowa pozbawiona jest tej wady: algorytm jest bardziej "ruchliwy", ryzyko utknięcia w minimum lokalnym jest mniejsze. Nie jest ono jednak całkowicie wyeliminowane: jeśli przykłady w kolejnych epokach prezentowane są stale w tej samej kolejności, trajektoria sieci/neuronu w przestrzeni wag może ulec "zapętleniu".

Problem "przeuczenia"

"Przeuczenie" (*overlearning*): sieć uczy się "zbyt dobrze" pojedynczych obiektów, nie generalizując (szczególnie istotne w interesującym nas zastosowaniu w uczeniu maszynowym, ML)





ak zapobiec przeuczeniu?

- dobrze dobrane kryterium stopu
- "erozja" wag; np.

$$w_i^j := (1-\varepsilon) w_i^j$$

  Ten sam efekt da się uzyskać dodając człon kary do błędu średniokwadratowego:

$$Q_{new} = Q + \frac{1}{2}\gamma \sum_i w_i^2$$

  Wada: bardziej karze za jedną dużą wagę, niż za wiele małych.
- usuwanie wag
- usuwanie nadmiarowych neuronów
- specjalizowane algorytmy uczące:
  - cascade correlation

Kiedy zatrzymać uczenie?

Kryterium stopu:

- z reguły błąd średniokwadratowy Q; stosowany głównie w zastosowaniach sieci do aproksymacji przebiegu funkcji
- niekiedy trafność klasyfikacji na zbiorze uczącym, zwłaszcza przy zastosowaniu w uczeniu maszynowym. Klasyfikacja kodowana jest wówczas najczęściej w postaci "1 z N"
  - podejście "progowe": wartość wyjścia l-tego neuronu w warstwie wyjściowej porównywana jest z pewnym progiem (najczęściej 0.5); jeżeli go przekracza, przykład klasyfikowany jest do l-tej klasy.
    - możliwość otrzymania decyzji niedeterministycznych.
  - podejście "maksimum" albo "zwycięzca bierze wszystko" (*winner takes all*): w warstwie wyjściowej znajdowany jest neuron o największej wartości wyjścia.
  - inne

Algorytmy modyfikujące topologię sieci podczas uczenia

- *Cascade-correlation* [Fahlman, Lebiere 1990]
  0. Początkowa konfiguracja sieci jest minimalna: sieć składa się tylko z warstwy wyjściowej
  1. Sieć uczona jest do chwili, gdy nie obserwuje się już dalszego spadku błędu.
  2. Tworzonych jest kilka nowych neuronów, tzw. *kandydatów*, z wejściami przyłączonymi do wszystkich wejść sieci i wszystkich dotychczas utworzonych neuronów warstwy ukrytej ; ich wyjścia nie są na razie przyłączone do neuronów warstwy wyjściowej.
  3. Kandydaci uczeni są w ten sposób, aby uzyskać możliwie dużą korelację z błędem popełnianym dotychczas przez neurony warstwy wyjściowej.
  4. Po pewnym czasie uczenia spośród kandydatów wybierany jest ten, którego korelacja z błędem popełnianym przez warstwę wyjściową jest największa; jest on wstawiany do sieci, tj. jego wyjście przyłączane jest do wszystkich neuronów warstwy wyjściowej; jego wagi wejściowe są "zamrażane"; pozostali kandydaci są usuwani.
  5. Jeśli spełnione jest kryterium stopu (wartość błędu spadła poniżej zadanej), algorytm kończy działanie; w przeciwnym przypadku następuje skok do punktu 1.