Sztuczne sieci neuronowe

Jerzy Stefanowski

Plan wykładu

1. Wprowadzenie
2. Model sztucznego neuronu.
3. Topologie sieci neuronowych
4. Reguły uczenia sieci neuronowych.
5. Klasyfikacja sieci neuronowych.
6. Sieci warstwowe uczone algorytmem BP.
7. Zastosowania.

Poznań, 2006

Wprowadzenie

Sztuczna sieć neuronowa (SSN) - definicje:

> Zbiór prostych jednostek obliczeniowych przetwarzających dane, komunikujących się ze sobą i pracujących równolegle.

Lub inaczej:

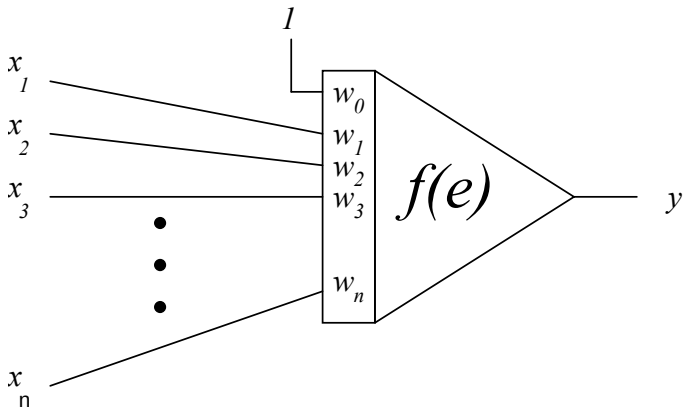
> Zbiór połączonych ze sobą jednostek wejściwo-wyjściowych. Z każdym połączeniem skojarzona jest waga, która może zostać zmieniona w trakcie uczenia.

Dowolna sztuczna sieć neuronowa może być zdefiniowana poprzez określenie:

- modelu sztucznego neuronu,
- topologii,
- reguły uczenia sieci.

Model sztucznego neuronu

Sztuczny neuron = neuron: można rozpatrywać jako specyficzny przetwornik sygnałów.



Podstawowe elementy składowe:

- n wejść neuronu wraz z wagami w_i (wektor wag **w** i wektor sygnałów wejściowych **x**)
- jeden sygnał wyjściowy y
- **pobudzenie** e neuronu jako suma ważona sygnałów wejściowych pomniejszona o próg Θ

$$e = \sum_{i=1}^{n} w_i \cdot x_i - \Theta = \mathbf{w}^T \mathbf{x} - \Theta$$

wprowadźmy wagę $w_0 = \Theta$, podłączonej do stałego sygnału $x_0 = 1$; wówczas:

$$e = \sum_{i=0}^{n} w_i \cdot x_i = \mathbf{w}^T \mathbf{x}$$

- **funkcja aktywacji** (przejścia):

$$y = f(e)$$

Funkcje aktywacji

Ma istotne znaczenie dla działania neuronu.

Podstawowe typy funkcji:

- liniowa $y = k \cdot e$
- nieliniowe (ciągłe i nieciągłe, unipolarne i bipolarne)

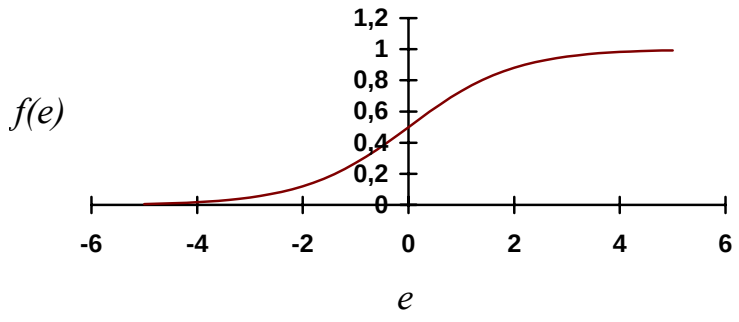
funkcja skoku jednostkowego, progowa (McCulloch i Pitts):

$$f(e) = \begin{cases} 1 \text{ dla } e \geq \Theta \\ 0 \text{ dla } e < \Theta \end{cases}$$

funkcja sigmoidalna:

$$f(e) = \frac{1}{1 + \exp(-\beta e)}$$

współczynnik stromości β

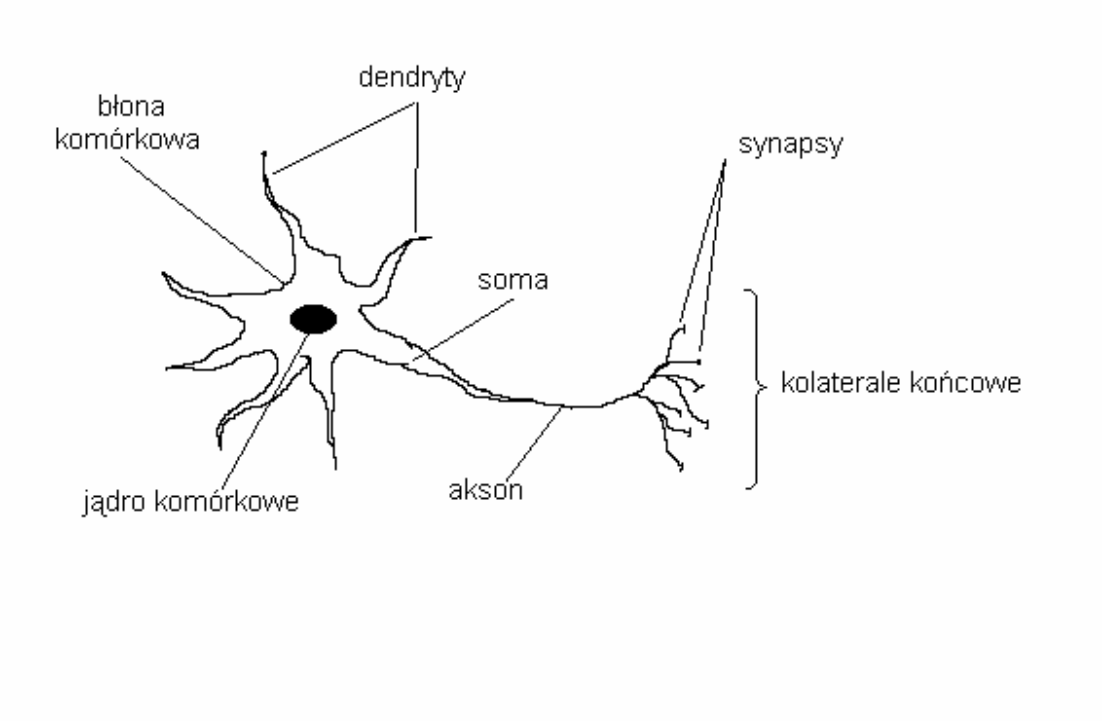


funkcja tangens hiperboliczny:

$$f(e) = tgh(\frac{\alpha e}{2}) = \frac{1 - \exp(-\alpha e)}{1 + \exp(\alpha e)}$$

Inspiracja: neuron “biologiczny”

- budowa: *soma, akson, dendryty, synapsy*

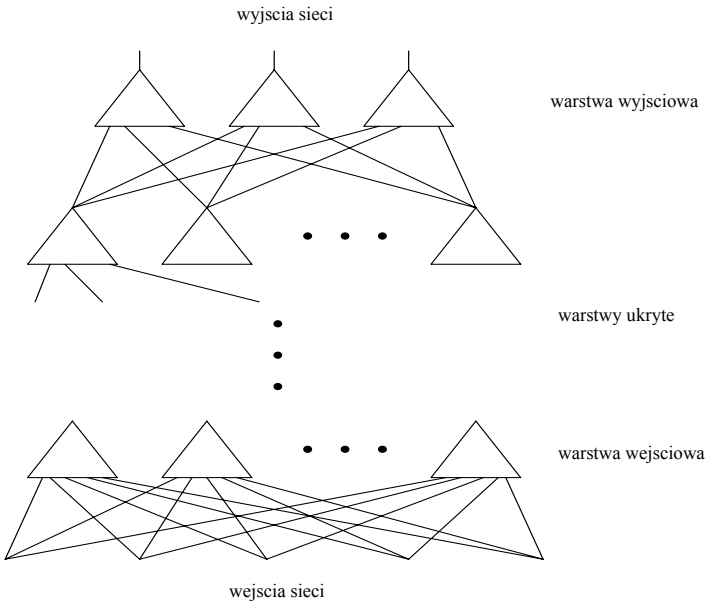


- znaczenie błony komórkowej w przesyłaniu sygnału; polega ono na propagacji zaburzenia różnicy potencjałów pomiędzy wnętrzem a zewnętrzem komórki; przyczyną tych zaburzeń jest chwilowa utrata „szczelności” przez błonę komórkową
- zasada działania: „wpływające” dendrytami bodźce (modulacja częstotliwości) sumują się (oddziaływają ze sobą) na błonie komórkowej i przy pomocy aksonu zakończonego synapsą/synapsami przekazywane są do innego neuronu/neuronów
- po propagacji sygnału różnica potencjałów odbudowywana jest przez tzw. pompy jonowe
- neuronów mamy $\sim 10^{10}$, dendrytów $\sim 10^{14}..10^{15}$
- różne rodzaje neuronów

Topologia sieci neuronowej (architektura)

Ogólnie wyróżnia się dwa typy architektur SSN:

1) sieci jednokierunkowe *(* ang. *feedforwarded)* tj. sieci o jednym kierunku przepływu sygnałów; Szczególnym przypadkiem architektury jednokierunkowej jest sieć **warstwowa**, reprezentująca zdecydowanie najpopularniejszą topologię;



2) Inne, np. **sieci rekurencyjne** *(feedback, bidirectional)* tj. sieci ze sprzężeniami zwrotnymi (sieć Hopfielda) albo sieci uczenia się przez współzawodnictwo (Kohonena)

Zasady łączenia neuronów między sobą

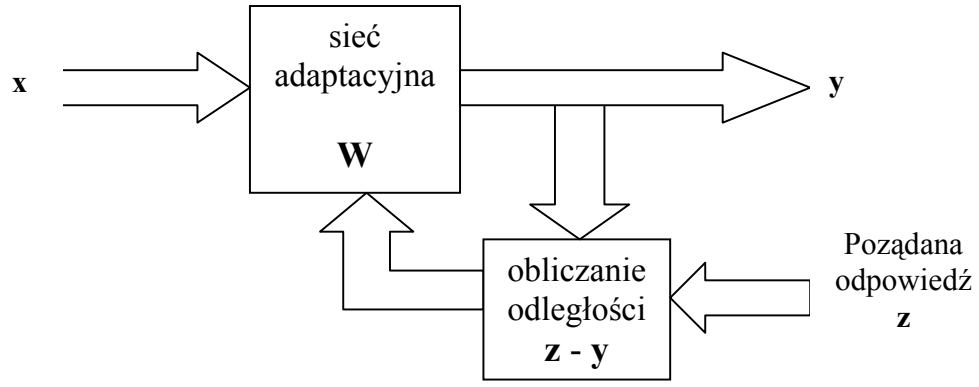
- „każdy z każdym”,
- połączenia między kolejnymi warstwami w sieciach warstwowych,
- tylko z pewną grupą neuronów, najczęściej z tzw. *sąsiedztwem*.

Charakterystyka procesu uczenia sieci

Wyróżnia się dwa podstawowe sposoby uczenia sieci:

1. uczenie nadzorowane (ang. *supervised learning*),
2. uczenie nienadzorowane (ang. *unsupervised learning*).

Uczenie nadzorowane – dany jest zbiór przykładów uczących składający się z par wejście-wyjście ($\boldsymbol{x}_j$, $\boldsymbol{z}_j$), gdzie $\boldsymbol{z}_j$ jest pożądana odpowiedzią sieci na sygnały wejściowe $\boldsymbol{x}_j$ (j=1,..m). Zadaniem sieci jest nauczyć się możliwie jak najdokładniej funkcji przybliżającej powiązanie wejścia z wyjściem.



Odległość pomiędzy rzeczywistą a pożądaną odpowiedzią sieci jest miarą błędu używaną do korekcji wag sieci.

Typowym przykładem jest uczenie sieci wielowarstwowej algorytmem wstecznej propagacji błędu; każdy neuron lokalnie zmniejsza swój błąd stosując metodę spadku gradientu.

Reguły uczenia sieci neuronowych

Reguła Widrowa-Hoffa

Dotyczy uczenia nadzorowanego sieci jednokierunkowych, gdzie minimalizuje się błąd pomiędzy pożądaną a aktualną odpowiedzią.

$$\delta^j = z^j - y^j = z^j - \mathbf{w}^T \mathbf{x}^j$$

Korekta wag jest następująca (Widrow,Hoff 1962):

$$\Delta w_i = \eta \cdot \delta^j \cdot x_i^j$$

Reguła delta

Obowiązuje dla neuronów z ciągłymi funkcjami aktywacji i nadzorowanego trybu uczenia. Regułę delta wyprowadza się jako wynik minimalizacji kryterium błędu średniokwadratowego Q.

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

Korekta wag:

$$\Delta w_i = \eta \cdot \delta^j \cdot f'(e^j) \cdot x_i^j$$

gdzie f'() oznacza pochodną funkcji aktywacji. W przypadku funkcji sigmoidalnej:

$$\Delta w_i = \eta \cdot \delta^j \cdot (1 - y^j) \cdot y^j \cdot x_i^j$$

Stosowana jest do uczenia wielowarstwowych sieci neuronowych wraz z algorytmem wstecznej propagacji błędów (Rumelhart, McClelland 1986)

Charakterystyka procesu uczenia

Uczenie się iteracyjne / uczenie się w jednym kroku

Algorytm uczenia:

- globalny (w kolejnej iteracji uczenie obejmuje całą sieć),
- lokalny (w kolejnej iteracji uczenie obejmuje część sieci)

Sposób propagacji sygnałów przez sieć:

- **synchroniczny**
- **asynchroniczny**:
- „przesyłanie żetonów” (*counter-propagation*): specyficzny model propagacji sygnału bazujący na „dyskretnym” pobudzeniu w postaci tzw. „żetonu”.

Charakterystyka wybranych typów sieci

Typ sieci	Topologia	Propagacja pobudzenia	Połączenia	Uczenie	
Perceptron	warstwowa	synchron.	każdy z każdym warstwami	nadzor.	iteracyjne
SOM	warstwowa	synchron.	spec. jedno-warstwa	nienadzor.	iteracyjne
Hopfield	rekurencyjna	synchr. lub asynchr	każdy z każdym	nienadzor.	w jednym kroku

Wybrane typy sieci neuronowych

- Warstwowe sieci liniowe
  - Adaline/Madaline,
- Warstwowe sieci nieliniowe
  - wielowarstwowa uczona algorytmem wstecznej propagacji błędów,
  - sieci wielowarstwowe z modyfikacjami algorytmu wstecznej propagacji błędów,
  - sieci z funkcjami o symetrii kołowej – RBF.
- Sieci ze sprzężeniem zwrotnym
  - sieci Hopfielda,
  - dwukierunkowa pamięć asocjacyjna – BAM,
- Sieci uczone przez współzawodnictwo
  - sieci Kohonena, LVQ,
  - odwzorowanie cech istotnych oraz sieci samoorganizujące się SOM,
- Sieci rezonansowe ART oraz sieci z kontrpropagacją,
- Sieci neuronowe zintegrowane z algorytmami metaheurystycznymi
  - symulowane wyżarzanie – maszyna Boltzmana,
  - algorytmy genetyczne,
- Metody hybrydowe wykorzystujące sieci neuronowe,
- Systemy rozmyto-neuronowe.

Uwagi na temat stosowanie sieci neuronowych

R.Tadeusiewicz (2000):

„Sieci neuronowe mogą być stosowane z dużym prawdopodobieństwem odniesienia sukcesu wszędzie tam, gdzie pojawiają się problemy związane z tworzeniem modeli matematycznych pozwalających odwzorowywać złożone zależności pomiędzy pewnymi sygnałami wejściowymi a wybranymi sygnałami wyjściowymi”

Potrzeba automatycznego – w wyniku tzw. procesu uczenia – modelowania złożonych zależności.

T.Mitchell (1997):

Cechy charakterystyczne problemów dla SNN

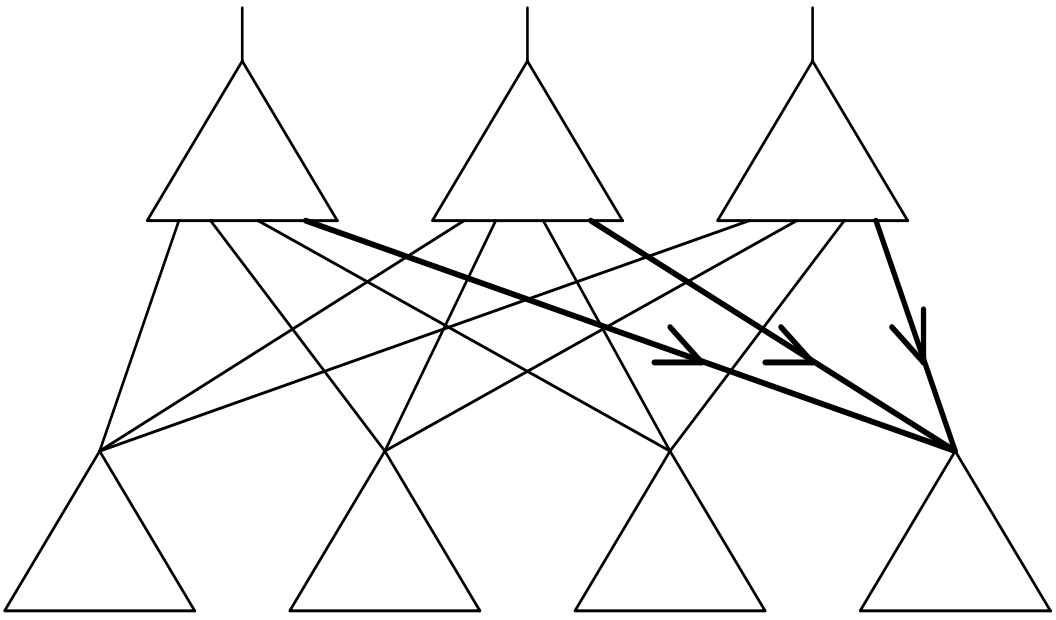
- Przykłady uczące opisane są przez pary atrybut-wartość (na ogół zdefiniowanych na skalach liczbowych; np. różnego rodzaju sygnały lub rezultaty pomiarów),
- Przybliżana funkcja może mieć wartości dyskretne lub rzeczywiste; może być także wektorem wartości,
- Dane mogą zawierać błędy lub podlegać „zaszumieniu”; SSN są odporne na różnego rodzaju uszkodzenia danych,
- Akceptowalny jest długi czas uczenia sieci,
- Akceptacja dla potencjalnie dużej liczby parametrów algorytmu, które wymagają dostrojenia metodami eksperymentalnymi,
- Zadanie nie wymaga rozumienia przez człowieka funkcji nauczonej przez SNN - trudności z interpretacją wiedzy nabytej przez sieć.

Algorytm wstecznej propagacji błędów (backpropagation)

Jak znaleźć błąd popełniany przez neurony z warstw ukrytych?

Błąd *k*-tego neuronu w *l*-tej warstwie jest równy sumie błędów popełnionych przez neurony (*p*) z warstwy *l*+1-szej ważonych po wagach $w_{k(p,l+1)}$ łączących ten neuron z neuronami tej warstwy:

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



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

1. Podaj na wejście sieci kolejny wektor wymuszeń $\boldsymbol{x}^j$.
2. Przepropaguj wymuszenie przez sieć, obliczając pobudzenia neuronów w kolejnych warstwach, aż do warstwy wyjściowej.
3. Wektor wyjść otrzymany w warstwie wyjściowej $\boldsymbol{y}^j$ porównaj z wektorem uczącym/oczekiwanym $\boldsymbol{z}^j$ i oblicz na tej podstawie błędy δ^j popełnione przez neurony tej warstwy.
4. 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.

Parametry reguły delta i algorytmu wstecznej propagacji błędu

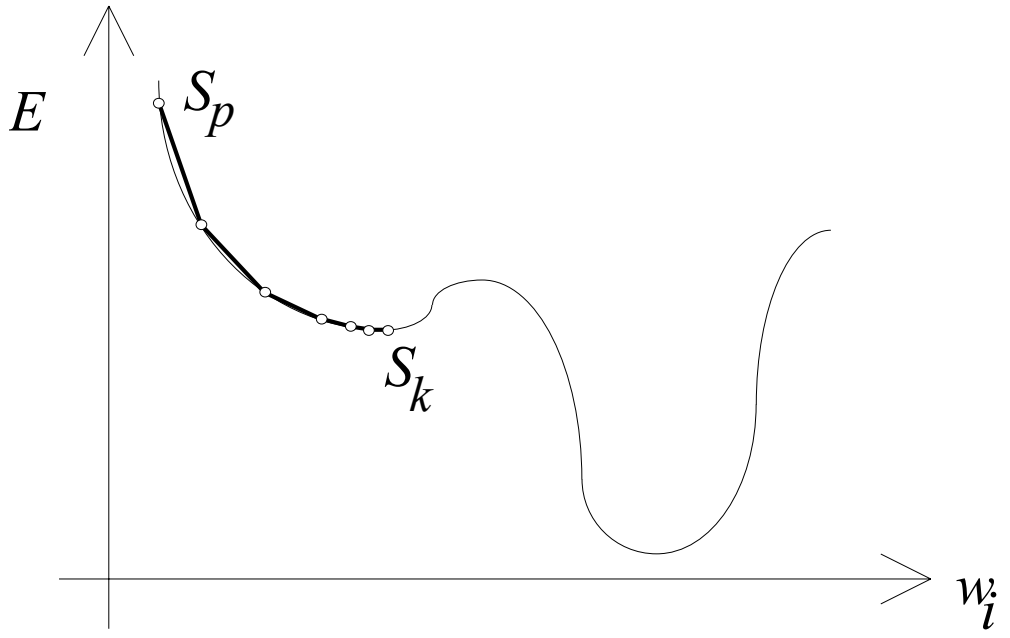
- początkowa konfiguracja *wektora wag* **w**:

  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 konfiguracje sieci porusza się "małymi kroczkami" po krajobrazie energetycznym)



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)*:

$$\Delta w_i^t = -\eta \frac{\partial Q^t}{\partial w_i} + \alpha \Delta w_i^{t-1} \tag{0.1}$$

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.

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 podejścia:

- 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*:

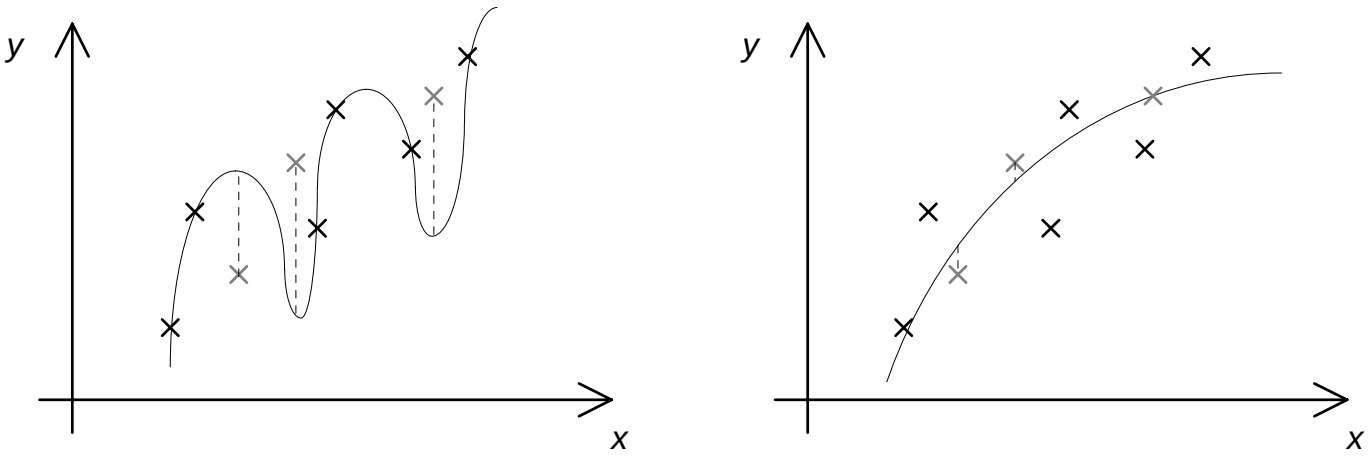
  Poprawki obliczone przy prezentacji wzorca są używane bezpośrednio (w tym samym kroku algorytmu) do modyfikacji wag.

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)



Jak 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

Uwagi na temat stosowanie sieci neuronowych

R.Tadeusiewicz (2000):

„Sieci neuronowe mogą być stosowane z dużym prawdopodobieństwem odniesienia sukcesu wszędzie tam, gdzie pojawiają się problemy związane z tworzeniem modeli matematycznych pozwalających odwzorowywać złożone zależności pomiędzy pewnymi sygnałami wejściowymi a wybranymi sygnałami wyjściowymi”

Potrzeba automatycznego – w wyniku tzw. procesu uczenia – modelowania złożonych zależności.

T.Mitchell (1997):

Cechy charakterystyczne problemów dla SNN

- Przykłady uczące opisane są przez pary atrybut-wartość (na ogół zdefiniowanych na skalach liczbowych; np. różnego rodzaju sygnały lub rezultaty pomiarów),
- Przybliżana funkcja może mieć wartości dyskretne lub rzeczywiste; może być także wektorem wartości,
- Dane mogą zawierać błędy lub podlegać „zaszumieniu”; SSN są odporne na różnego rodzaju uszkodzenia danych,
- Akceptowalny jest długi czas uczenia sieci,
- Akceptacja dla potencjalnie dużej liczby parametrów algorytmu, które wymagają dostrojenia metodami eksperymentalnymi,
- Zadanie nie wymaga rozumienia przez człowieka funkcji nauczonej przez SNN - trudności z interpretacją wiedzy nabytej przez sieć.