Zakład Sztucznej Inteligencji
ZUT Szczecin

Sztuczna inteligencja i maszynowe uczenie w systemach interaktywnych

Joanna Kolodziejczyk
jkolodziejczyk@zut.edu.pl
pokój nr 18 WI1
konsultacje: środa 14:00 - 15:30

December 3, 2020

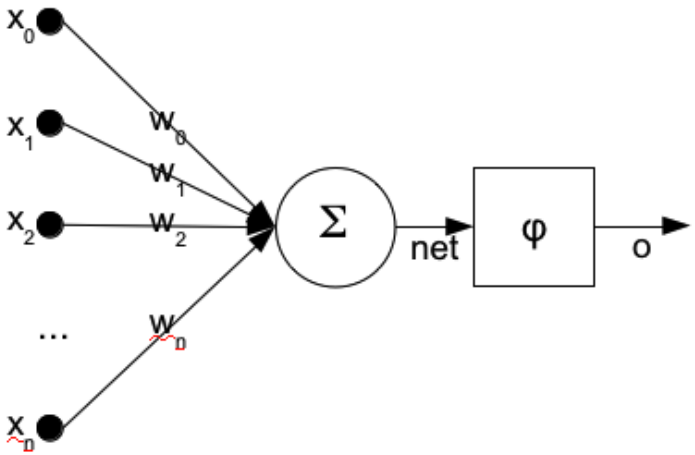
Spis treści

Sztuczny Neuron

Działanie neuronu

$$o = \varphi(net) = \varphi(\sum_{i=0}^{n} x_i * w_i)$$

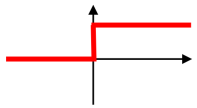
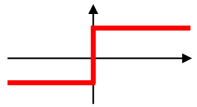
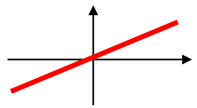
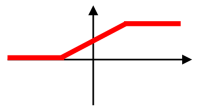
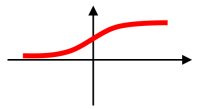
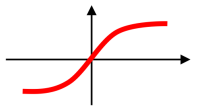
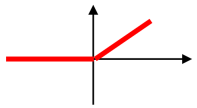
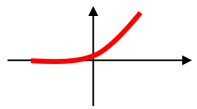
gdzie x_i — wejścia i $x_0 = 1$, w_i — wagi, φ — funkcja aktywacji, net — pobudzenie neuronu, o — wyjście z neuronu



Cechy neuronu

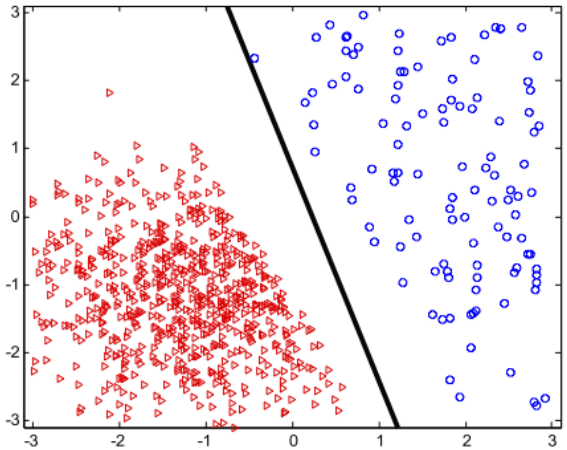
- Wejścia i wagi są liczbami rzeczywistymi dodatnimi i ujemnymi.
- Jeżeli jakaś cecha (wejście) powoduje odpalenie neuronu, waga będzie dodatnia, a jeżeli cecha ma działanie hamujące to waga jest ujemna.
- Funkcję aktywacji dobiera się do rozwiązywanego zadania.
- Wagi są dopasowywane przez pewną regułę uczenia tak, by zależność wejście/wyjście w neuronie spełniało pewien określony cel.

Stosowane w SSN funkcje aktywacji

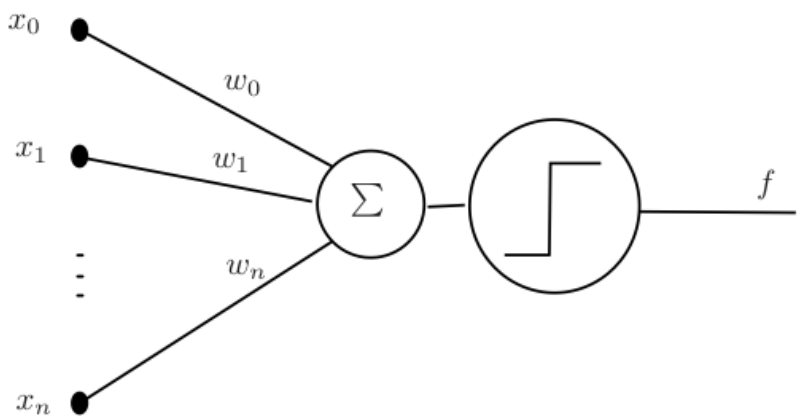
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

Perceptron prosty (Rosenblatt 1962)

Perceptron prosty jest najprostszą SSN, używaną do klasyfikacji binarnej. Perceptron składa się z pojedynczego neuronu z regulacją wag. Rosenblatt zaproponowała twierdzenie zbieżności perceptronu, stwierdzając, że konwergencja jest gwarantowana wtedy i tylko wtedy, gdy wzorce wykorzystywane do uczenia perceptronu tworzą dwie separowalne klasy. Powierzchnia decyzyjna zostanie umieszczona w formie hiperpłaszczyzny gdzieś pomiędzy tymi dwiema grupami.



Perceptron — model



Funkcja aktywacji

$$f(\langle \mathbf{w}, \mathbf{x} \rangle) = \begin{cases} 1 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle > \mathbf{0}; \\ 0 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle \leq \mathbf{0}. \end{cases}$$

Zadanie klasyfikacji binarnej

Klasyfikacja binarna

Klasyfikacja polega na przydzieleniu obiektu do pewnej klasy na podstawie jego atrybutów (danych wejściowych). W klasyfikacji binarnej wyjściem jest dwupunktowe wyjście (możliwe dwie klasy na wyjściu).

Niech dany będzie zbiór uczący (zbiór par) $(\mathbf{x}_i, y_i)$ $i = 1, \ldots, l$, gdzie $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \in \mathbb{R}^n$ są danymi wejściowymi a $y_i \in \{0, 1\}$ są skojarzonymi z nimi klasami (wyjściami).

Płaszczyzna separacji klas

Równanie płaszczyzny w przestrzeni $\mathbb{R}^n$

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = 0$$

czyli należy znaleźć taki zestaw $w_0, w_1, w_2, \ldots, w_n$, że

$$\forall_{i, y_i = 1} w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0$$

oraz

$$\forall_{i, y_i = 0} w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \leq 0,$$

gdy dodamy jeszcze jeden punkt do wejść $x_i = (1, x_{i1}, x_{i2}, \ldots, x_{in})$, to równanie płaszczyzny:

$$\langle \mathbf{w}, \mathbf{x_i} \rangle = \sum_{j=0}^{n} w_j x_{ij}$$

Uczenie SSN

Uczenie

Uczenie polega na automatycznym doborze wag w SSN, na podstawie zbioru przykładów nazwanych zbiorem uczącym. Zaczyna się z losowymi małymi wagami i iteracyjnie zmienia się wagi, dopóki wszystkie przykłady uczące nie zostaną poprawnie zaklasyfikowane (lub z niewielkim błędem).

Wyróżnia się dwa typy uczenia, zależy od wykorzystanej sieci i przykładów (zbioru uczącego):

1. nadzorowane (z nauczycielem), gdy w zbiorze danych do każdej próbki podana jest poprawna klasa
2. nienadzorowane (bez nauczyciela), zbiór danych nie ma wektora odpowiedzi.

Algorytm uczenia perceptronu

1. Niech $\mathbf{w}(0) = (0, \ldots, 0)$ lub wartości losowe z przedziału $[-1, 1]$, $k = 0$
2. Dopóki zbiór punktów uczących pozostaje błędnie klasyfikowany tj. zbiór $A = \{\mathbf{x}_i : y_i \neq f(\langle \mathbf{w}, \mathbf{x}_i \rangle)\}$ pozostaje niepusty, powtarzaj:
   - 2.1 Wylosuj ze zbioru A dowolny punkt
   - 2.2 Aktualizuj wagi według następuącej reguły:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta e \mathbf{x}_i$$

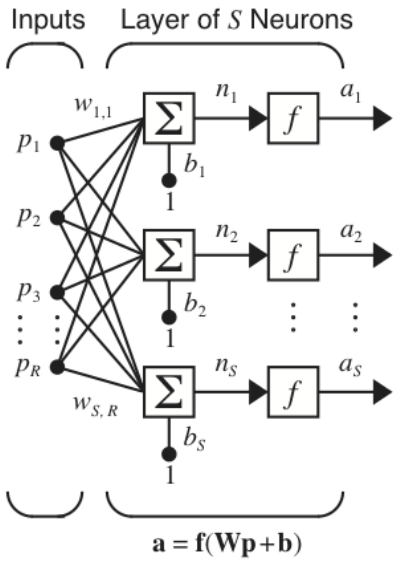
   - 2.3 $k = k + 1$

Gdzie η współczynnik uczenia $\eta \in (0, 1]$. A e jest wartością błędu popełnianego na prezentowanej próbce.

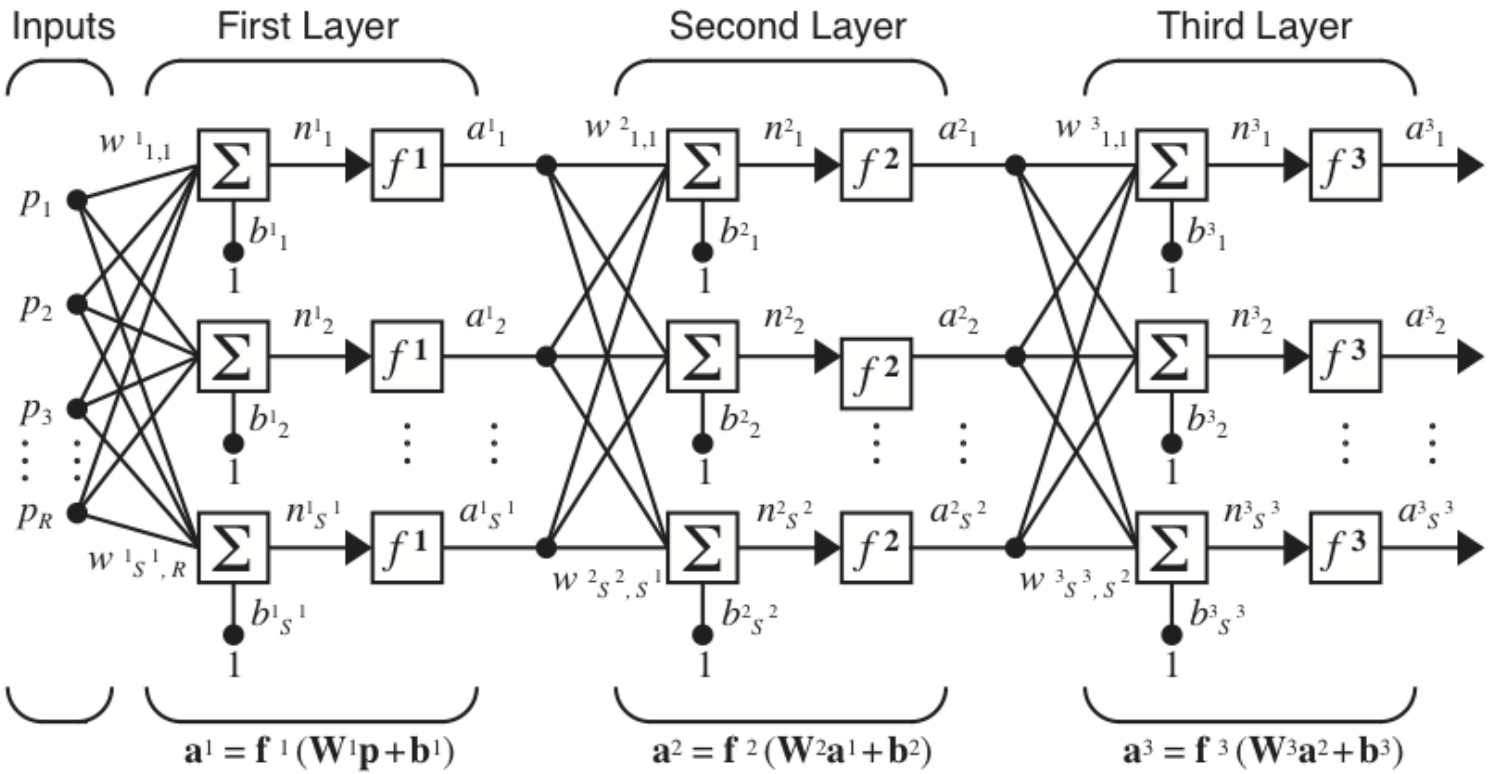
Perceptron prosty jednowarstowy

Pojedyncza warstwa neuronów (Single layer perceptron) - cechy

1. Pewna liczba neuronów S jest ułożona w warstwie i wszystkie wejścia R sieci zasilają wszystkie neurony. $S \neq R$.
2. Wyjście z sieci jest wektorem. Wagi są macierzą $S \times R$.
3. Umożliwia klasyfikację do większej niż dwie liczby klas.
4. Zazwyczaj wszystkie neurony mają tą samą funkcję aktywacji.



Sieć wielowarstwowa MLP



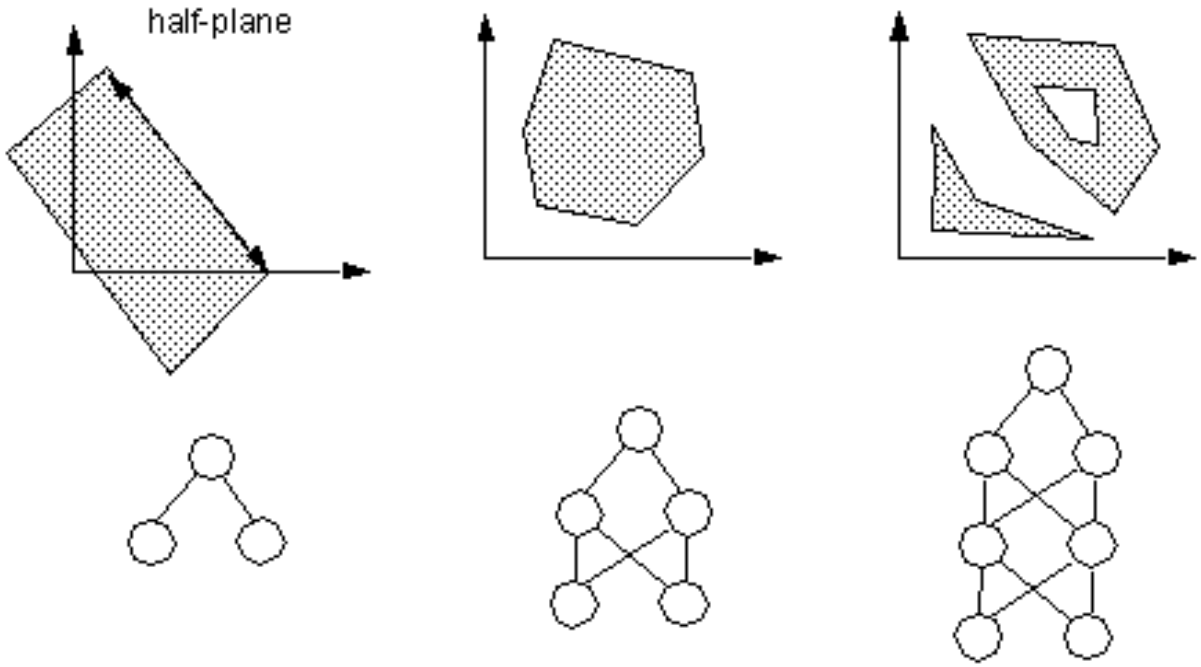
$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

Cechy sieci MLP

Wielowarstwowy perceptron

1. Ostatnia warstwa nazywana jest wyjściową, a wewnętrzne warstwami ukrytymi.
2. MLP ma dużo większe zastosowania niż pojedyncza warstwa.
3. Liczba wejść i neuronów wyjściowych jest określona przez rozwiązywane zadanie (zbiór danych wejściowych).
4. Liczba warstw ukrytych powinna być ograniczona do jednej lub dwóch. Wyjaśnienie na kolejnym slajdzie.
5. Nie ma wskazania ile powinno być neuronów w każdej warstwie. Źle dobrana ta liczba będzie skutkować złym dopasowaniem (nadmiernym lub za słabym) sieci do danych uczących.

Graficzna prezentacja możliwości separowania klas przez MLP



Dane uczące dla SSN

1. Należy zebrać dane niezbędne w procesie uczenia. Mogą pochodzić z baz danych.
2. Wybrać zmienne mające znaczenia dla rozwiązywanego zadania. Zazwyczaj na początku przyjmuje się wszystkie a potem dokonuje się redukcji.
3. Zebrana liczba rekordów powinna być reprezentatywna. Powinno się zadbać, by pokazać dane obejmujące pełne dziedziny zmiennych.
4. W rzeczywistości liczba potrzebnych rekordów jest również uzależniona od złożoności zależności funkcyjnej poddawanej modelowaniu.

Dane uczące dla SSN - obróbka wstępna

1. Dane numeryczne są przeskalowywane do właściwego dla sieci przedziału (normalizacja).
2. Wartości brakujące są zastępowane wartościami średnimi (lub innymi statystykami) obliczanymi na podstawie wartości zmiennych dostępnych w ciągu uczącym.
3. Typ danych nominalnych takich jak Płeć = Mężczyzna, Kobieta zamienia się na wartości numeryczne. Przy dużej liczbie klas łączy się je w grupy.
4. Wartości ciągłe poddaje się procesowi dyskretyzacji.

Uczenie nadzorowanie sieci wielowarstwowej

Wagi

„Inteligencja” SSN kryje się w wartościach wag. Zatem konieczna jest metoda dostosowywania wag w celu rozwiązania danego problemu.

Wsteczna propagacja błędów

Dla sieci typu MLP (wielowarstwowy perceptron), najczęściej wykorzystuje się algorytm uczenia BP (backpropagation), czyli wstecznej propagacji błędów.

Uczenie nadzorowane

Sieć w tej metodzie uczy się przez przykłady, czyli, należy dostarczyć zestaw próbek składający się z par wejście-wyjście. Wyjście to znany poprawny wynik dla każdego przypadku. Tak więc znane jest oczekiwane zachowanie sieci neuronowej, a algorytm BP pozwala na dopasowanie wejście-wyjście.

Algorytm BP (wersja uproszczona)

1. Ustaw wagi **w** na wartości losowe z przedziału $[-1, 1]$, $k = 0$
2. Dopóki nie osiągnięto założonego progu błędu $e(L) = Err$ lub nie wykonano założonej liczby kroków $k < MaxEpoch$:
   - 2.1 Wylosuj ze zbioru próbek dowolny punkt i oblicz odpowiedź sieci ,
   - 2.2 Warstwę analizowaną ustaw na ostanią $i = L$
   - 2.3 Wykonuj dla warstw $i > 0$
     - 2.3.1 Aktualizuj wagi połączeń w warstwie i zgodnie ze wzorem

$$\mathbf{w}(k+1, i) = \mathbf{w}(k, i) + \eta \mathbf{e}(i)\mathbf{x}_i(i)$$

     - 2.3.2 $i = i - 1$
   - 2.4 $k = k + 1$

Gdzie $\mathbf{e}(i)$, to błędy popełniane przez neurony w warstwie (i). W warstwie wyjściowej błąd obliczany jest na podstawie różnicy sygnału oczekiwanego i generowanego przez sieć. W warstwach ukrytych obliczany jest z błędu warstwy następnej. η - współczynnik uczenia.

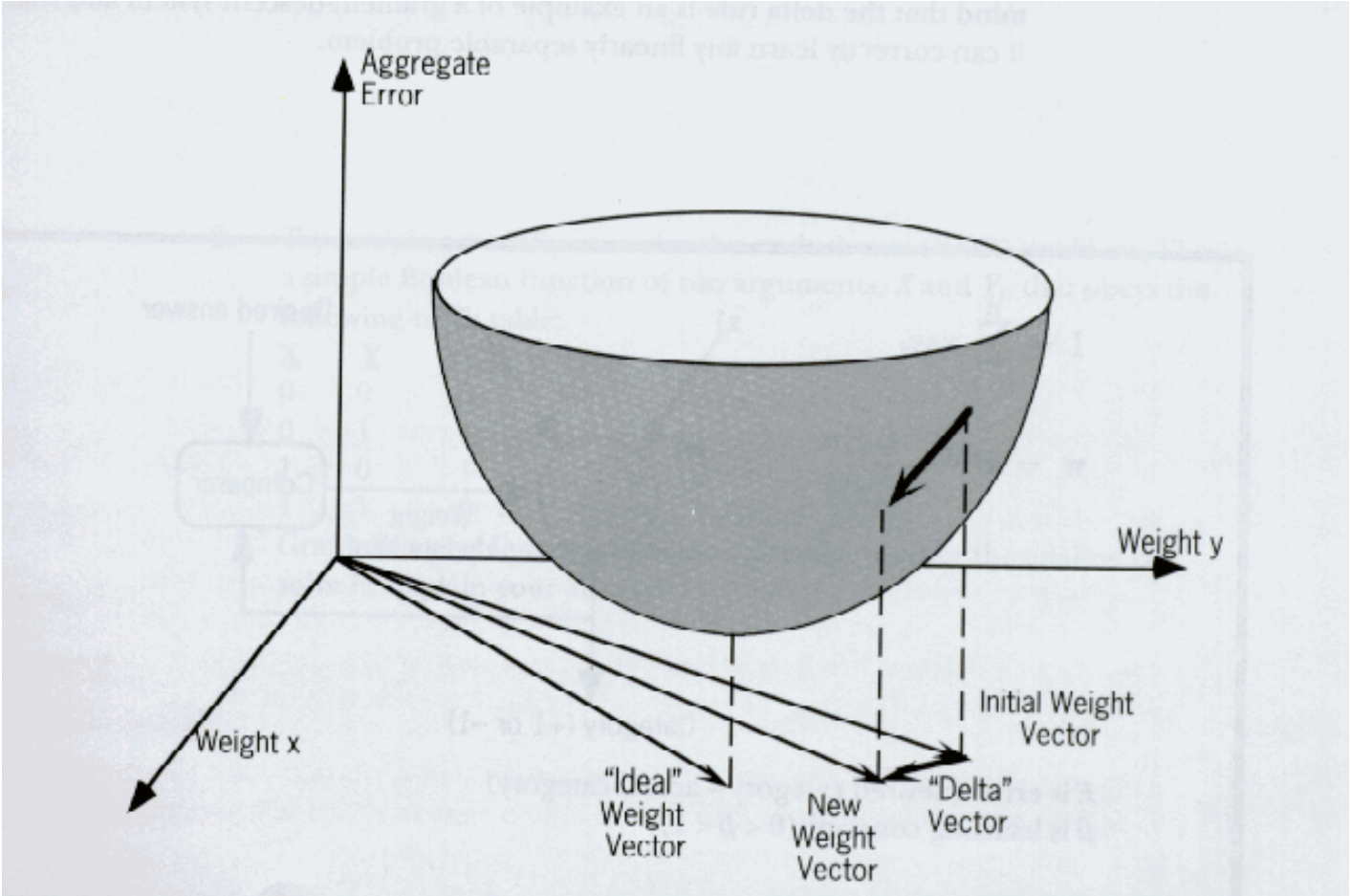
Słowny opis algorytmu BP

- W kolejnych iteracjach wykonuje się kroki: jedna z próbek uczących jest wprowadzana do sieci.
- Na jej podstawie w oparciu o aktualny stan wag (początkowo wyniki będą losowe) sieć będzie dawała odpowiedź.
- Odpowiedź ta jest porównywana do znanego wyjścia z próbki, obliczany jest średni błąd kwadratowy.
- Wartość błędu jest następnie propagowana wstecz przez sieć i wprowadzane są małe zmiany w wartościach wag w każdej warstwie.
- Zmiany wag wykonuje się tak, by zmniejszyć wartość błędu dla danej próbki.
- Cały proces jest powtarzany dla każdej próbki, i może być wykonywany wielokrotnie.
- Cykl ten powtarza się, aż całkowita wartość błędu spada poniżej progu, który jest z góry określony.

Ważne uwagi do BP

- Metoda działa tak, by minimalizować funkcję błędu $E = \frac{1}{2}\sum_{o=1}^{N_o}(y_o - output_o)$, gdzie y_o — oczekiwana wartość wyjścia, $output_o$ —wartość obliczona przez sieć, a N_o — liczba wyjść.
- Podczas nauki, wagi każdego połączenia są zmieniane przez wartość, która jest proporcjonalna do sygnału błędu.
- Można powiedź, że sieć nauczy się rozwiązywania problemu „wystarczająco dobrze”. Właściwie sieć nigdy nie będzie tworzyć dokładnego odwzorowania ($Err \neq 0$), ale będzie raczej asymptotycznie zbiegać do funkcji idealnej.
- Ważne, by funkcja aktywacji była funkcją różniczkowalną, gdyż minimalizacja funkcji błędu wiąże się z wyznaczeniem gradientu.

Ważne uwagi do BP



http://reocities.com/ResearchTriangle/3626/backprop.htm

Współczynnik uczenia η

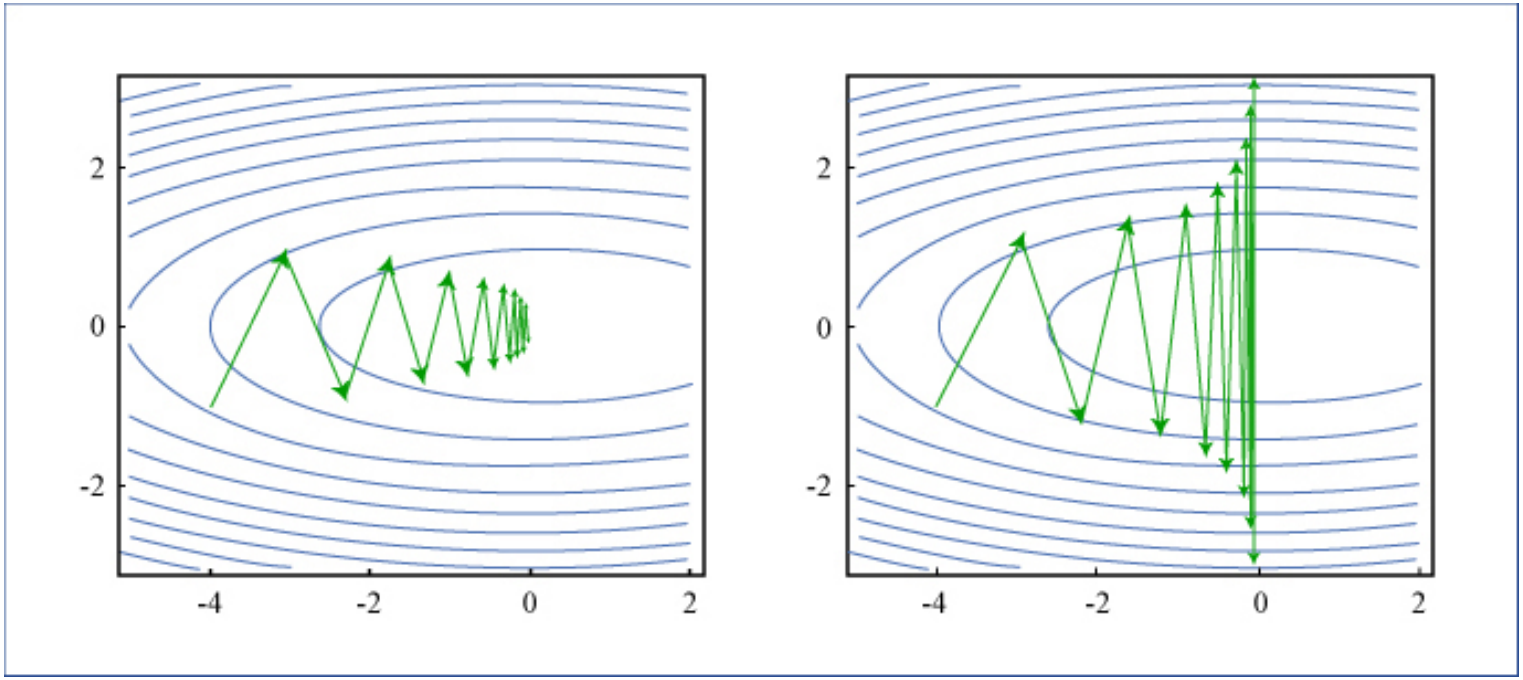
Znaczenie

Reguła uczenia wymaga, by zmiana wag była proporcjonalna do gradientu. Gradientowa reguła największego spadku oczekuje wykonywania nieskończonej liczby kroków. Stałą skalującą jest współczynnik uczenia.

Wartości

W praktyce wybiera się maksymalnie duży współczynnik uczenia, ale taki, który nie prowadzi do oscylacji.

Ważne uwagi do BP



Feathergraphics by MIT OpenCourseWare. HST.951J / 6.873J Medical Decision Support, Fall 2005

Współczynnik momentum α

Wzór na poprawkę wagi z momentum

$$\mathbf{w}(k+1,i) = \mathbf{w}(k,i) + \underbrace{\eta \mathbf{e}(i)\mathbf{x}_i(i)}_{\Delta w(k+1,i)} + \alpha \Delta w(k,i)$$

Przesłanką do zastosowania współczynnika momentum jest to, że poprzednie zmiany w wagach powinny mieć wpływ na aktualny kierunek przesuwania się w przestrzeni wag. Zatem momentum zapobiega oscylacjom.
Czyli, gdy wagi zaczną przemieszczać się w określonym kierunku w przestrzeni wag, to będą dążyć do dalszego ruchu w tym samym kierunku. Momentum pomaga przeskoczyć przez lokalne minima.

Dwa tryby aktualizowania wag

Tryb onliine

Aktualizacja wag po każdym zaprezentowanym wzorcu.

Tryb wsadowy (batch)

Aktualizacja po prezentacji wszystkich wzorców.

Jeśli współczynnik uczenia jest niewielki, to różnica pomiędzy dwoma procedurami jest niewielka. Znaczne różnice można zaobserwować, gdy współczynnik uczenia jest duży, jako że algorytmu wstecznej propagacji błędów zakłada, że pochodne błędów są zsumowane po wszystkich wzorcach.

Generalizacja

Generalizacja

Uogólnienie, operacja myślowa, polegająca na wykryciu przez uczącego się cechy lub zasady wspólnej dla danej klasy przedmiotów lub zjawisk (psych.). Jest to najbardziej oczekiwana cecha sieci.

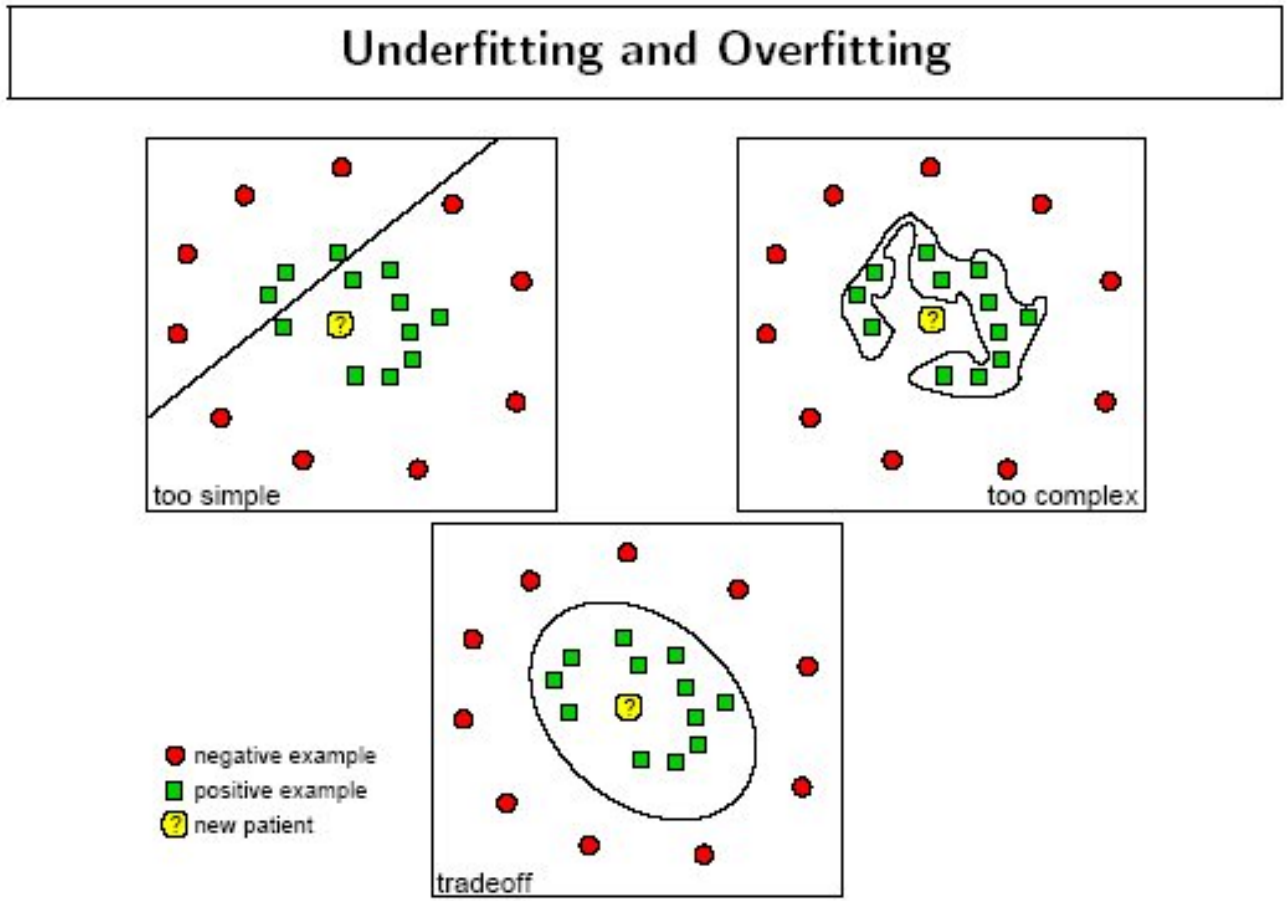
Przeuczenie

Sieci bardzo skomplikowane (duża liczba wag) tworzą bardziej złożony model i są przez to bardziej skłonne do nadmiernego dopasowania. Przeuczenie rozpoznaje się po tym, że błąd uczenia jest bardzo mały, a błąd testowania jest duży.

Niedouczenie

Sieci z niewielką liczną neuronów (wag) mogą być niewystarczająco silne do zamodelowania poszukiwanej funkcji odwzorowania wejść w wyjście. Niedouczenie daje duży błąd uczenia. Prawie zawsze większe sieci generują mniejszy błąd.

Ważne uwagi do BP



http://www.dtreg.com/svm.htm

Kiedy nie stosować MLP

- Czy można zapisać schemat lub wzór, który dokładnie opisuje problem? Jeśli tak, to należy stosować tradycyjne metody.
- Czy istnieje już rozwiązanie sprzętowe lub softwarowe, które rozwiązuję większość problemu? Jeśli tak, to czas poświęcony na aplikację SSN może nie być tego wart.
- Czy funkcje mają „ewoluować' w kierunku, który nie jest z góry określony? Jeśli tak, wykorzystaj algorytm genetyczny.
- Czy łatwo wygenerować/pozyskać znaczną liczbę próbek wejście-wyjście pokazujących poprawne działanie? Jeśli nie, to nie będzie można trenować SNN.
- Czy problem jest bardzo „dyskretny"? Czy poprawną odpowiedź można znaleźć metodą Look-Up Table? Look-Up Table jest znacznie prostszy i bardziej dokładny.
- Czy wymagane są precyzyjne wartości liczbowe na wyjściach? SSN nie jest korzystne, gdy należy podać dokładne odpowiedzi liczbowe.

Kiedy stosować MLP

- Dostępna jest duża liczba danych typu wejście-wyjście, ale nieznana jest relacja pomiędzy wejściem i wyjściem.
- Problem wydaje się mieć ogromną złożoność, ale wyraźnie istnieje rozwiązanie.
- Łatwo jest wygenerować przykłady właściwego zachowania/działania.
- Rozwiązanie problemu może ulec zmianie, w granicach podanych parametrów wejściowych i wyjściowych (tzn. dzisiaj 2 +2 = 4, ale w przyszłości może się okazać, że 2 +2 = 3.8).
- Wyjścia mogą być rozmyte lub nienumeryczne.

Zastosowania sieci MLP

Jednym z najbardziej typowych zastosowań SSN jest przetwarzanie obrazów:

- rozpoznawanie pisma odręcznego
- dopasowanie fotografii twarzy do zdjęć z bazy danych
- kompresja obrazu przy minimalnej utracie jakości.

Inne aplikacje:

- rozpoznawanie mowy
- analiza podpisu radarowego
- przewidywania rynku akcji.

Uwaga

Stosując MLP nie trzeba znać rozwiązania problemu. W tradycyjnych metodach trzeba rozumieć zależność pomiędzy wejściem i wyjściem. Z siecią postępujemy tak, że pokazujemy jakie jest poprawne wyjście dla pewnego zestawu danych wejściowych.

Po odpowiednim czasie uczenia sieć odkryje zależność pomiędzy wejściami i wyjściami. Czasami dobrze jest zaprezentować sieci przykłady, które są nieistotne i sieć nauczy się je ignorować.

Natomiast ominięcie istotnych danych przyczyni się do tego, że sieć zbuduje niewłaściwą zależność wejście-wyjście.

Miles per Gallon

Zazwyczaj problemy rozwiązywane siecią neuronową są związane ze zbiorem statystyk [1]. Celem będzie wykorzystanie pewnych danych by przewidywać inne dane. Rozważmy bazę danych samochodów. Zawiera ona następujące pola:

- Waga samochodu
- Pojemność silnika
- Liczba cylindrów
- Konie mechaniczne
- Typ hybrydowy lub benzynowy
- Wydajność: Mil na galon

[1] Przykład zastosowania sieci `http://www.heatonresearch.com/content/non-mathematical-introduction-using-neural-networks`

Miles per Gallon cd.

Cel

Zakłada się, że zostały zebrane dane dla wymienionych atrybutów, zatem możliwe jest zbudowanie sieci neuronowej, która jest w stanie przewidzieć, wartość jednej cechy, na podstawie wartości innych atrybutów. W tym przykładzie omówiony zostanie przykład prognozowania zużycia mil-per-galon.

Normalizacja danych

Należy zdefiniować problem zapisując w macierzy wejść z wartościami rzeczywistymi z przyporządkowaną macierzą wyjść zawierającą również wartości rzeczywiste. Należy zapewnić, by zakres liczbowy każdego elementu w macierzy zawierał się między 0 a 1 lub -1 i 1. Przekształcenie takie nazywane jest normalizacją.

Miles per Gallon cd.

Architektura SSN

Baza zawiera sześć atrybutów/cech. Pięć z nich będzie użyte, by przewidywać szóstą. Zatem sieć będzie miała pięć wejść i jedno wyjście.

Sieć będzie skonstruowana następująco:

Input Neuron 1: Waga samochodu
Input Neuron 2: Pojemność silnika
Input Neuron 3: Liczba cylindrów
Input Neuron 4: Konie mechaniczne
Input Neuron 5: Typ: Hybrydowy lub benzynowy

Output Neuron 1: Wydajność: Mil na galon

Miles per Gallon cd.

Zakresy wartości dla danych:

Waga samochodu: 100-5000 lbs
Pojemność silnika: 0.1 to 10 litrów
Liczba cylindrów: 2-12
Konie mechaniczne: 1-1000
Hybrydowy lub benzynowy: true lub false
Mil na galon: 1-500

Przyjęte zakresy nie koniecznie są rzeczywiste, przykład jest poglądowy. Jednakże, sieć można douczyć na nowych danych. Niekorzystnym jest, by zbiór uczący zawierał zbyt wiele danych na skrajnych końcach przedziałów.

Miles per Gallon cd.

Normalizacja do zakresu 0-1

Jak jest znormalizowana wartość dla wagi samochodu: 2000 lbs?

- Różnica pomiędzy wagą rozpatrywanego samochodu a wagą minimalną to $2000 - 100 = 1900$.
- Zebrane w bazie auta mają wagi w zakresie 100 do 5000lbs. Czyli zakres wag to $5000 - 100 = 4900$.
- Obliczamy udział masy 2000 w maksymalnym zakresie $1900/4900 = 0.38$.
- To jest wartość znormalizowana i taka zasili wejście sieci neuronowej.

Wartości obliczane zgodnie z przedstawioną procedurą będą spełniały kryteria zakresu 0-1.

Miles per Gallon cd.

Wartości nominalne

Wejście „hybrydowy” ma wartość true/false. Do reprezentowania tych wartości używa się 1 dla hybrydowych i 0 dla benzynowych. Wartości prawda/fałsz sprowadza się do dwóch wartości.

W następnym kroku można wykorzystać algorytm uczenia BP i po jego zakończeniu sieć będzie umiała wskazać jakie jest przeciętna wydajność auta na podstawie 5 atrybutów.

Jak zaprezentować obraz?

Obrazy są popularnym źródłem informacji dla sieci neuronowych. Przedstawiona niżej metoda kodowania obrazu jest prosta ale często skuteczna.

Zakłada się, że obraz, ma rozmiar 300x300 pikseli. Dodatkowo obraz jest kolorowy. Należy zatem 90000 pikseli pomnożyć przez trzy kolory RGB, co daje 270.000. W tym przypadku powinno się przyjąć sieć z 270.000 neuronami wejściowymi. To jest zdecydowanie za dużo dla sztucznej sieci neuronowej.

Jak zaprezentować obraz? cd.

Wynik downsamplingu do rozmiaru 32x32.

Sieć neuronowa wymaga teraz 1024 wejścia, i zakłożono, że patrzy się tylko na intensywność każdego kwadratu. Sieci neuronowe widzi zatem w bieli i czerni.

Jak zaprezentować obraz? cd.

- Dana wejściowa do każdego z 1024 wejść, to poziom nasycenia mający wartość od 0 do 255.
- By dokonać normalizacji wartość wejściową wystarczy podzielić przez 255, na przykład: intensywność określona liczbą 10 staje się wejściem o wartości 10/255 lub 0.039.

Jak zaprezentować obraz? cd.

Wyjścia z sieci

W prezentowanym przykładzie oczekuje się od neuronów wyjściowych określania jaki obraz widzi sieć neuronowa. Zwykle rozwiązaniem jest utworzenie jednego neuronu wyjściowego dla każdego rodzaju rozpoznawanego przez SSN obrazu. Sieć zostanie nauczona, aby wystawić na wyjściu neuronu odpowiadającemu za dany obraz wartość 1.

Sieć neuronowa do danych zmieniających się w czasie

Finansowe sieci neuronowe

Są bardzo popularną formą temporalnej sieci neuronowych. Temporalne sieci neuronowe to takie, które akceptują na wejściu wartości zmienne w czasie. Istnieją dwa sposoby prezentowania danych zmiennych w czasie w sieci neuronowej. Jedną z nich jest użycie okna wejścia i okna predykcji.

Rozważmy sieć neuronową do przewidywania rynku akcji. Dostępne są następujące informacje na temat cen akcji, które reprezentują kursu zamknięcia dla akcji w ciągu kilku dni.

Sieć neuronowa do danych zmieniających się w czasie cd.

Pierwszym krokiem jest normalizacja danych. W tym celu zamienia się każdą wartość na procent w stosunku do wartości *t* z poprzedniego dnia. Na przykład, dzień 2 da 0,04, bo przesunięcie z 45 dolarów do 47 to 4% wzrostu w stosunku do 45.

Dzień	Dane przed normalizacją	Dane po normalizacji
Day 1	$45	
Day 2	$47	0.04
Day 3	$48	0.02
Day 4	$40	-0.16
Day 5	$41	0.02
Day 6	$43	0.04
Day 7	$45	0.04
Day 8	$57	0.04
Day 9	$50	-0.12
Day 10	$41	-0.18

Sieć neuronowa do danych zmieniających się w czasie cd.

Cel

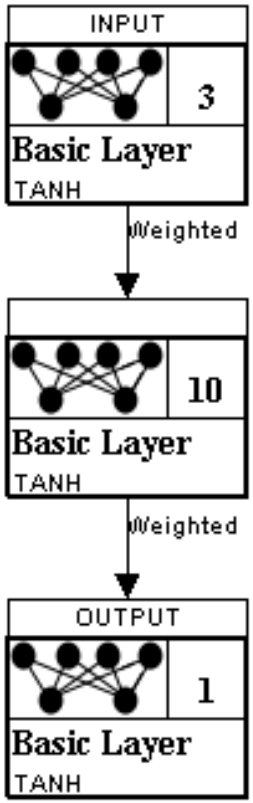
Utworzyć sieć neuronową, która będzie przewidywać wartości na następny dzień. Jak zakodować dane, które zostaną wprowadzone do sieci neuronowych? Sposób, w jaki to zrobimy zależy od tego, czy sieć neuronowa jest rekurencyjna.

Sieć neuronowa do danych zmieniających się w czasie cd.

Wielowarstwowy perceptron

Aby korzystać z tej sieci do przewidywania cen akcji musimy korzystać z „okien czasowych”. Będziemy korzystać z okna trzech cen, by przewidzieć czwartą wartość. Tak więc, by przewidzieć jutrzejszy kurs patrzymy na ostatnie trzy dni. Oznacza to, że mamy trzy neurony wejściowe i jeden w warstwie wyjściowej.

Sieć neuronowa do danych zmieniających się w czasie cd.



Sieć neuronowa do danych zmieniających się w czasie cd.

Dane uczące dla wielowarstwowego perceptronu:

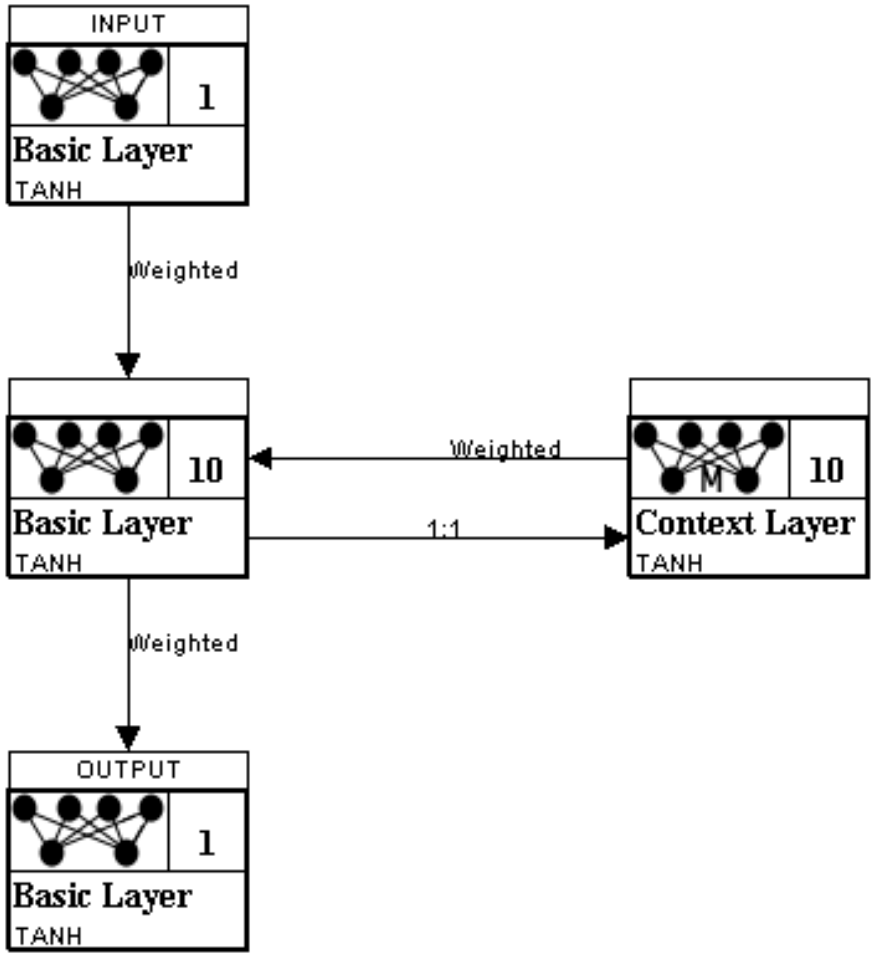
- Próbka 1: $(0.04, 0.02, -0.16) \rightarrow (0.02)$
- Próbka 2: $(0.02, -0.16, 0.02) \rightarrow (0.04)$
- Próbka 3: $(-0.16, 0.02, 0.04) \rightarrow (0.04)$
- Próbka 4: $(0.02, 0.04, 0.04) \rightarrow (0.04)$
- Próbka 5: $(0.04, 0.04, 0.04) \rightarrow (-0.12)$
- Próbka 6: $(0.04, 0.04, -0.12) \rightarrow (-0.18)$

Sieć neuronowa do danych zmieniających się w czasie cd.

Sieć rekurencyjna

Istnieje kilka różnych typów rekurencyjnych sieci neuronowych. Używana w tym przykładzie jest nazywana siecią neuronową Elmana. W sieci tej zastosowany zostanie jeden neuron wejściowy oraz pojedynczy neuron wyjściowy. Będzie można dostarczać tylko informacje z jednego dnia i oczekiwać zmiany kursy w dniu następnym.

Sieć neuronowa do danych zmieniających się w czasie cd.



Sieć neuronowa do danych zmieniających się w czasie cd.

Architektura sieci

Sieć neuronowa Elmana ma również 10 ukrytych neuronów. Tak jak było w przypadku sieci jednokierunkowych, wartość 10 została wybrana arbitralnie. W tej sieci funkcjonuje rekurencyjna warstwa neuronów kontekstowa. Pozwala ona zapamiętać pewien stan. To spowoduje, iż wyjście z sieci neuronowej zależy od większej liczby wejść niż tylko ostatnie.

Sieć neuronowa do danych zmieniających się w czasie cd.

Dane uczące dla sieci Elmana:

$(0.04) \rightarrow (0.02)$
$(0.02) \rightarrow (-0.16)$
$(-0.16) \rightarrow (0.02)$
$(0.02) \rightarrow (0.04)$
$(0.04) \rightarrow (0.04)$
$(0.04) \rightarrow (0.04)$
$(0.04) \rightarrow (-0.12)$
$(-0.12) \rightarrow (-0.18)$

Sieć neuronowa do danych zmieniających się w czasie cd.

Jak widać z powyższych danych, bardzo ważna jest kolejność prezentowanych próbek. Sieć neuronowa nie może przewidzieć przyszłej ceny widząc tylko cenę bieżącą. Dlatego, że warstwa kontekstowa pamiętająca ostatnie kilka dni będzie miała wpływ na wyjście. Można to porównać z rozpoznaniem piosenki. Trzeba usłyszeć przynajmniej kilka nut w sekwencji. Oczywiście kolejność dźwięków jest bardzo ważna.

Wprowadzanie tekstu do sieci neuronowej

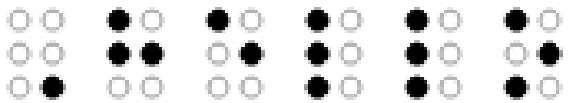
Wprowadzanie tekstu do sieci neuronowej jest wyjątkowo kłopotliwe z dwóch powodów:

1. słowo ma zmienną długość (sieć wymaga stałej liczby wejść i wyjść)
2. kodowanie znaków (a-z powinno być przechowywane w 1 czy 26 neuronach?)

Zastosowanie sieci Elmana rozwiązuje część z tych problemów. Stosując sieć Elmana z wystarczającą liczbą neuronów w warstwie wejściowej, by zapamiętać litery alfabetu i stosując warstwę kontekstową, by pamiętać kolejność liter można rozwiązać problem rozpoznawania tekstu. Analiza tekstu będzie się zatem wiązać z wprowadzaniem strumienia znaków.

Wprowadzanie tekstu do sieci neuronowej cd.

Kodowanie można rozwiązać stosując np. kody ASCII.
Niniejszy przykład pokazuje zastosowanie znaków Braille'a, które łatwiej zakodować. Braille stosuje 6 kropek by reprezentować literę i znaki interpunkcyjne. Zakłada się, że te 6 kropek będzie wejściami do sieci neuronowej. Słowo „Hello" podane poniżej:



Pierwsze 6 kropek oznacza, że następny znak jest wielką literą. Zatem litera „h" (drugi znak od lewej) można zakodować jako (1, -1, 1, 1 -1, -1) (1 dla czarnej kropki, -1 dla białej).

Wprowadzanie tekstu do sieci neuronowej cd.

Cel

Sieć ma rozpoznać słowo „Hello” i nie dać sygnału na wyjściu tak długo, jak długo wyraz się nie pojawi w całości.

Sieć Elmana będzie miała 6 wejść i 1 wyjście. Na wyjściu pojawi się 0 gdy wyraz nie rozpoznany i 1, gdy rozpoznano sekwencję znaków „hello”.

Dane uczące:

$[-1,-1,-1,-1,-1,1] \rightarrow [0]$
$[1,-1,1,1,-1-1] \rightarrow [0]$
$[1,-1,-1,1,-1-1] \rightarrow [0]$
$[1,-1,1,-1,1,-1] \rightarrow [0]$
$[1,-1,1,-1,1,-1] \rightarrow [0]$
$[1,-1,-1,1,1,-1] \rightarrow [1]$

Wprowadzanie tekstu do sieci neuronowej cd.

Prezentowany przykład pokazuje rozpoznanie pojedynczego słowa. Sieć Elmana powinna sobie radzić z wyrazami o zmiennej długości.

Aby rozpoznawać więcej wyrazów, trzeba dodać kolejny neuron wyjściowy. Rozpoznawanie języka naturalnego staje się więc przy takim systemie modelowania zbyt złożone.

Dziękuję za uwagę
Czas na pytania ????