Inteligenţă artificială
Cursul 7

Reţele neuronale
Sinteză

Clasificarea şi regresia

Aceste metode de învăţare automată presupun construirea unui model pe baza unei mulţimi de antrenare, formată dintr-o mulţime de instanţe (vectori de antrenare, obiecte). Instanţele au atribute. Fiecare instanţă are atribute cu anumite valori. Pentru o problemă de clasificare, de obicei, ultimul atribut este clasa. De exemplu:

Atribute

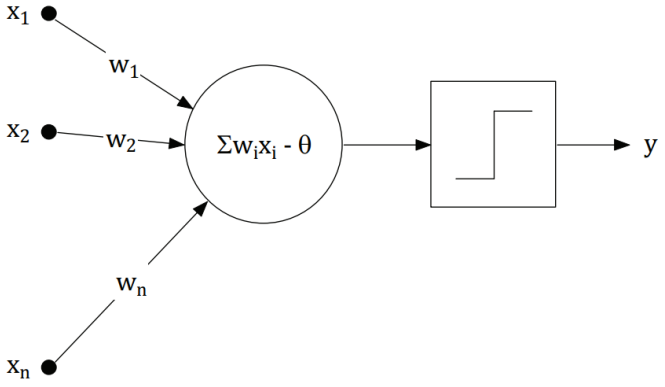
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Instanţe

Singura diferenţă între clasificare şi regresie este tipul ieşirii: discret, respectiv continuu. Au însă aceeaşi idee de bază: învăţarea unei relaţii între intrări (vectorul **x**) şi ieşire (y) din date. Odată construit modelul, acesta va putea fi folosit pentru predicţie, adică pentru instanţe diferite de cele conţinute de mulţimea de antrenare.

Perceptronul standard

Este un model de neuron artificial în care semnalele de intrare sunt sumate, iar un semnal de ieşire este generat doar dacă suma depăşeşte pragul.



$$y = F\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$$

Perceptronul are ponderi sinaptice ajustabile și funcție de activare semn sau treaptă:

$$F(a) = \begin{cases} -1, & dacă\ a < 0 \\ 1, & dacă\ a \geq 0 \end{cases}$$

$$F(a) = \begin{cases} 0, & dacă\ a < 0 \\ 1, & dacă\ a \geq 0 \end{cases}$$

Scopul perceptronului este să clasifice o instanță într-una din două clase. Spațiul intrărilor, *n*-dimensional, este împărțit în două de un hiperplan definit de ecuația:

$$\sum_{i=1}^{n} x_i w_i - \theta = 0$$

Pragul poate fi considerat o pondere suplimentară, cu intrarea tot timpul 1 sau –1. Acest fapt simplifică formula de calcul a ieșirii:

$$y = F\left(\sum_{i=1}^{n+1} w_i x_i\right)$$

Învățarea are loc prin ajustarea succesivă a ponderilor pentru a reduce diferența dintre ieșirile reale (y) și ieșirile dorite (y_d), pentru *toate* datele de antrenare.

Regula de învățare a perceptronului este:

$$\Delta w = \alpha \cdot x \cdot e$$

unde Δw este corecția ponderii, α este rata de învățare, x este intrarea, iar e este eroarea ($y_d - y$).

Perceptronul standard (cu un singur strat) poate învăța tot ce poate reprezenta, dar nu poate reprezenta multe funcții. Mai ales, nu poate reprezenta funcții neseparabile liniar, de exemplu, XOR.

Adaline

Adaline este un tip de neuron asemănător perceptronului, dar are funcție de activare liniară și se antrenează diferențial.

$$y = \sum_{i=1}^{n+1} w_i x_i$$

Eroarea folosită este eroarea pătratică:

$$E_i = \frac{1}{2}(y_{di} - y_i)^2$$

Antrenarea presupune minimizarea erorii în raport cu ponderile. Regula de ajustare a ponderilor este aceeași ca la perceptron.

În cazul unei funcții de activare *f* neliniare, dar derivabile, *regula delta* este:

$$\frac{\partial E_i}{\partial w_j} = -x_{ij}\,(y_{di} - y_i)\,f'(y_i)$$

de unde se deduce:

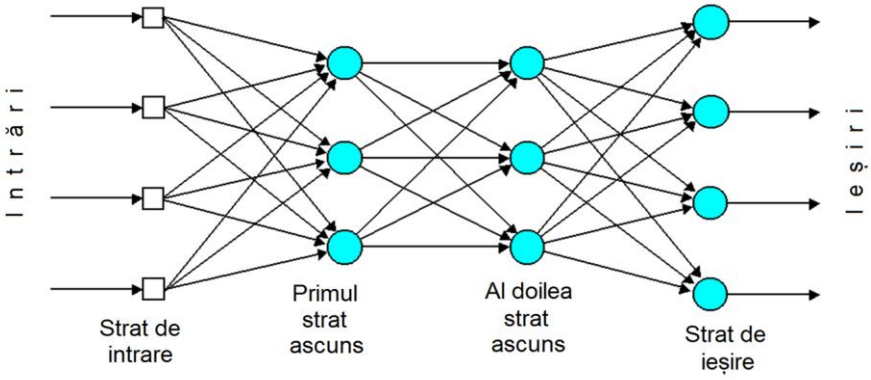
$$\Delta_i w_j = \alpha \cdot x_{ij} \cdot e_i \cdot f'(y_i)$$

Perceptronul multistrat

Perceptronul multistrat este o rețea neuronală cu propagare înainte (*feed-forward*) cu unul sau mai multe straturi ascunse. El are:

- Un strat de intrare (care nu face procesări);
- Unul sau mai multe straturi ascunse / intermediare;
- Un strat de ieșire.

De exemplu:

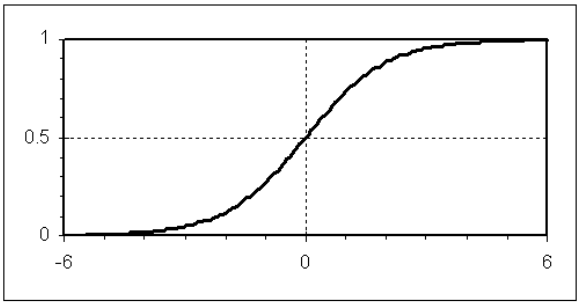


Proprietatea de aproximare universală. O rețea neuronală cu un singur strat ascuns, cu un număr posibil infinit de neuroni, poate aproxima orice funcție reală continuă.

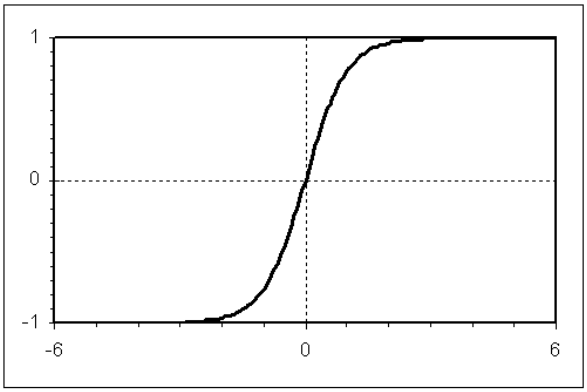
Din punct de vedere practic, un strat nu poate avea un număr infinit de neuroni. Un strat suplimentar poate reduce foarte mult numărul de neuroni necesari în straturile ascunse.

Funcțiile de activare pentru perceptronul multistrat trebuie să fie neliniare. Cel mai des utilizate sunt *sigmoida unipolară* (sau logistică), respectiv *sigmoida bipolară* (tangenta hiperbolică):

$$f(x) = \frac{1}{1+e^{-x}}$$



$$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$$



Un perceptron cu un singur strat are aceleași limitări chiar dacă folosește o funcție de activare neliniară.

Un perceptron multistrat cu funcții de activare liniare este echivalent cu un perceptron cu un singur strat.

Algoritmul de retro-propagare (back-propagation)

Are două faze:

- Rețeaua primește vectorul de intrare și propagă semnalul înainte, strat cu strat, până se generează ieșirea;
- Semnalul de eroare este propagat înapoi, de la stratul de ieșire către stratul de intrare, ajustându-se ponderile rețelei:

$$\Delta w_{jk} = \alpha \cdot y_j \cdot [y_k (1 - y_k)] \cdot e_k$$
$$\Delta w_{ij} = \alpha \cdot x_i \cdot [y_j (1 - y_j)] \cdot \sum_k \delta_k w_{jk}$$

Metoda momentului presupune adăugarea unui termen la regula delta, care accelerează și stabilizează căutarea:

$$\Delta w_{jk}(p) = \beta \cdot \Delta w_{jk}(p-1) + \alpha \cdot y_j(p) \cdot \delta_k(p)$$

Rata de învățare poate fi de asemenea modifica în mod adaptiv pe parcursul antrenării. Dacă variația sumei erorilor pătratice ΔE are același semn algebric pentru mai multe epoci consecutive, atunci rata de învățare α trebuie să crească. Dacă semnul lui ΔE alternează timp de câteva epoci consecutive, atunci rata de învățare α trebuie să scadă.

Rețele profunde

Primele straturi ale unui perceptron multistrat clasic cu un număr mai mare de straturi ascunse nu se antrenează bine. Când ponderile sunt mici sau funcțiile de activare sigmoide sunt saturate (gradienți mici), corecțiile ponderilor Δw sunt mici, iar antrenarea este foarte lentă. De asemenea, ultimele straturi, cele apropiate de ieșire, pot învăța problema „destul de bine" și deci semnalul de eroare trimis către primele straturi devine și mai mic. Sunt necesare alte metode de antrenare pentru a pune în valoare primele straturi.

În tabelul de mai jos se prezintă o comparație între rețelele clasice și cele profunde. Însă, În afară de numărul de straturi, diferențele nu sunt stricte.

Rețele clasice	**Rețele profunde**
• 1-2 straturi • Funcții de activare sigmoide • Funcții de cost bazate pe eroarea medie pătratică (MSE) • Algoritmi de antrenare: Backpropagation, RProp, Levenberg-Marquardt etc.	• Mai multe straturi • Funcții de activare mai simple: ReLU • Funcții de cost bazate pe estimarea verosimilității maxime (maximum likelihood estimation, MLE) • Algoritmi de antrenare: SGD, RMSProp, Adam etc. • Alte metode de inițializare a ponderilor, regularizare, pre-antrenare