ШТУЧНИЙ НЕЙРОН

Нейронні мережі – це моделі, що імітують діяльність людського мозку. Незважаючи на велику розмаїтість варіантів нейронних мереж вони мають спільні ознаки. Вони, подібно до мозку людини, складаються з великого числа однотипних елементів – нейронів, які імітують нейрони головного мозку і є пов'язані між собою. На рис.1. показано схему нейрона

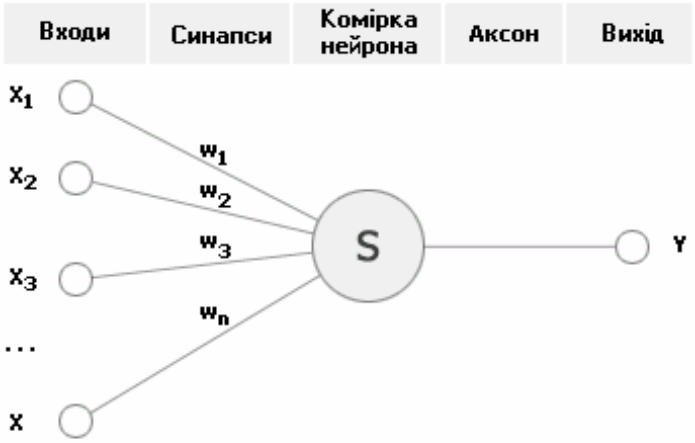

Рис. 1. Схема нейрона

З рис.1. видно, що штучний нейрон, подібно до біологічного, складається з синапсів, які зв'язують входи нейрона з ядром, ядра нейрона, яке здійснює обробку вхідних сигналів та аксона, що зв'язує нейрон з нейронами наступного шару. Кожен синапс має вагу, яка визначає наскільки відповідний вхід нейрона впливає на його стан. Стан нейрона визначається за формулою:

$$S = \sum_{i=1}^{n} x_i w_i \qquad (1)$$

де n - число входів нейрону, x_i – значення i-го входу нейрона, w_i – вага i-го синапсу

Значення виходу визначається за формулою:

$$Y=f(S) \qquad (2)$$

де f – певна функція, що називається передатною.

Найчастіше в якості передатної функції використовують, так звану, сигмоїду, що має наступний вид:

$$f(x) = \frac{1}{1+e^{ax}} \qquad (3)$$

Основною перевагою сигмоїди є, що вона диференційована по всій осі абсцис і має просту похідну:

$$f'(x) = af(x)(1-f(x)) \qquad (4)$$

При зменшенні параметра a сигмоїда стає більш пологою, вироджуючись у горизонтальну лінію на рівні 0,5 при a=0. При збільшенні a сигмоїда наближається до функції одиничного стрибка.

НЕЙРОННІ МЕРЕЖІ ЗВОРОТНЬОГО ПОШИРЕННЯ ПОХИБКИ

Нейронні мережі зворотного поширення – най поширена модель для пошуку закономірностей, прогнозування, якісного аналізу. Таку назву – мережі зворотного поширення (back propagation) вони мають через свій алгоритм навчання, де похибка поширюється від вихідного прошарку до вхідного, тобто в напрямку, протилежному напрямку поширення сигналу при нормальному функціонуванні мережі.

Нейрона мережа зворотного поширення складається з кількох прошарків нейронів, причому кожен нейрон прошарку i зв’язаний з кожним нейроном прошарку $i+1$, тобто мова йде про повнозв’язану мережу.

В загальному випадку задача навчання мережі зводиться до знаходження певної функціональної залежності $Y=F(X)$ де X - вхідний вектор, а Y - вихідний вектор. В загальному випадку така задача, за наявності обмеженого набору вхідних даних має нескінченну множину

розв'язків. Для обмеження простору пошуку при навчанні ставиться задача мінімізації цільової функції похибки мережі, що знаходиться за методом найменших квадратів:

$$E(w) = \frac{1}{2}\sum_{j=1}^{p}(y_j - d_j) \qquad (5)$$

де y_j - значення *j-го* виходу мережі, d_j - цільове значення *j-го* виходу, *p* – кількість нейронів в вихідному шарі.

$$\Delta w = -\eta \cdot \frac{\partial E}{\partial w_{ij}} \qquad (6)$$

де η - параметр, що визначає швидкість навчання.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{dS_j} \cdot \frac{\partial S_j}{\partial w_{ij}} \qquad (7)$$

де y_j - значення *j-го* виходу нейрону, S_j - зважена сума вхідних сигналів, що визначається за формулою (1), при цьому множник

$$\frac{\partial S_j}{\partial w_{ij}} \equiv x_i \qquad (8)$$

де x_i - значення *i-го* входу нейрону

Далі розглянемо визначення першого множника формули (7)

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{dS_k} \cdot \frac{\partial S_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{dS_k} \cdot w_{jk}^{(n+1)} \qquad (9)$$

де *k* – кількість нейронів в прошарку *n+1*

Введемо допоміжну змінну.

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{dS_j} \qquad (10)$$

Визначимо рекурсивну формулу для знаходження $\delta_j^{(n)}$ *n*-ного прошарку, якщо відоме $\delta_j^{(n+1)}$ наступного *n+1-го* шару.

$$\delta_j^{(n)} = \left[\sum_k \delta_j^{(n+1)} \cdot w_{jk}^{(n+1)}\right] \cdot \frac{dy_j}{dS_j} \qquad (11)$$

Знаходження $\delta_j^{(n)}$ для наступного прошарку ШНМ не викликає труднощів, оскільки є відомим цільовий вектор, тобто вектор тих значень, котрі повинна видавати мережа при даному наборі вхідних значень.

$$\delta_j^{(N)} = (y_j^{(N)} - d_j) \cdot \frac{dy_j}{dS_j} \qquad (12)$$

Напишемо формулу (6) в розкритому вигляді

$$\Delta w_{ij}^{(n)} = -\eta \cdot \delta_j^{(n)} \cdot x_i^n \qquad (13)$$

Тепер розглянемо повний алгоритм навчання нейронної мережі:

1. Подати на вхід мережі один з потрібних зразків і визначити значення виходів нейронів мережі.
2. Розрахувати $\delta_j^{(N)}$ для вихідного прошарку мережі за формулою (12) і визначити зміни ваг $\Delta w_{ij}^{(N)}$ вихідного прошарку *N* за формулою (13).
3. Розрахувати за формулами (11) і (13) відповідно $\delta_j^{(N)}$ і $\Delta w_{ij}^{(N)}$ для інших прошарків мережі, *N=N-1..1*
4. Скоректувати всі ваги мережі

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t) \qquad (14)$$

5. Якщо помилка є значною, то перейти на крок 1.

На другому етапі навчання мережі почергово в довільному порядку вибираються вектори з навчальної послідовності

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ НАВЧАННЯ МЕРЕЖІ ЗВОРОТНЬОГО ПОШИРЕННЯ .

Найпростіший метод градієнтного спуску виявляється неефективним у випадку, коли похідні по різних вагах сильно відрізняються. Це відповідає ситуації, коли значення функції *S* для деяких нейронів є близьким по модулю до 1. В цьому випадку для плавного зменшення похибки потрібно вибрати дуже малу швидкість навчання, але при цьому навчання може зайняти дуже багато часу.

Найпростішим методом вдосконалення градієнтного спуску є введення моменту μ, коли вплив градієнту на зміну ваг змінюється з часом.

Тоді формула (13) набуде вигляду

$$\Delta w_{ij}^{(n)}(t) = -\eta \cdot \delta_j^{(n)} + \mu \Delta w_{ij}^{(n)}(t-1) \qquad (15)$$

Додатковою перевагою від введення моменту є здатність алгоритму подолати дрібні локальні мінімуми.

ПРЕДСТАВЛЕННЯ ВХІДНИХ ДАНИХ.

Основна відмінність нейронних мереж у тому, що в них усі вхідні і вихідні параметри представлені у виді чисел із плаваючою крапкою в діапазоні [0..1]. У той же час дані предметної області часто мають інше кодування. Це можуть бути числа в довільному діапазоні, дати, символьні рядки. Тобто дані про проблему можуть бути як кількісними так і якісними. Розглянемо спочатку перетворення якісних даних у числові, а потім розглянемо спосіб перетворення вхідних даних у необхідний діапазон.

Якісні дані можна розділити на дві групи: впорядковані (ординальні) і невпорядковані. Для розгляду способів кодування цих даних розглянемо задачу про прогнозування успішності лікування якого-небудь захворювання. Прикладом впорядкованих даних можуть наприклад бути дані, наприклад, про додаткові фактори ризику при даному захворюванні.

нема	гіпертонія	куріння	алкоголь	ожиріння

А також можливим прикладом може бути наприклад вік хворого.

до 25 р.	25 – 39 р.	40 – 49 р.	50 – 59 р.	більше 60

Небезпека кожного фактора зростає в таблицях при просуванні зліва направо.

У першому випадку ми бачимо, що в хворого може бути кілька факторів ризику одночасно. У такому випадку нам необхідно використовувати таке кодування, при якому відсутня ситуація, коли різним комбінаціям факторів відповідає те саме значення. Розповсюдженим способом кодування є коли кожному фактору ставиться у відповідність розряд двійкового числа. 1 у цьому розряді говорить про наявність фактора, а 0 про його відсутність. В такий спосіб для представлення всіх факторів достатньо 4-х розрядного двійкового числа. Тоді число $1010_2 = 10_{10}$ означає наявність у хворого гіпертонії і вживання алкоголю, а число 0000_2 відповідає відсутності у хворого факторів ризику. Таким чином фактори ризику можна представити числами в діапазоні [0..15].

В другому випадку також можна кодувати всі значення двійковими цифрами, але це буде недоцільним, бо набір можливих значень буде нерівномірним. В цьому випадку вірнішим буде встановлення у відповідність кожному значенню своєї ваги, що відрізняється на 1 від ваги сусіднього значення. Так число 3 буде відповідати віку 50 – 59 років. У такий спосіб вік буде закодований числами в діапазоні [0..4].

Аналогічно можна вчинити і для невпорядкованих даних, поставивши у відповідність кожному значенню певне число. Однак це вводить небажану впорядкованість, що може спотворити дані, і ускладнити процес навчання. В якості одного із способів вирішення проблеми можна поставити у відповідність кожному значенню один із входів мережі. В такому випадку за наявності цього значення відповідний йому вхід встановлюється в 1 або в 0 у противному випадку. На жаль даний спосіб не є панацеєю, тому що при великій кількості варіантів вхідного значення число входів мережі розростається до величезної кількості. Це різко збільшує витрати часу на навчання. Можна використати дещо інший підхід. У відповідність кожному значенню вхідного параметра ставиться бінарний вектор, кожен розряд якого відповідає окремому входу мережі. Наприклад, якщо число можливих значень параметра 128, то можна використовувати *7-ми* розрядний вектор. Тоді першому значенню буде відповідати вектор 0000000 а сто двадцять восьмому - вектор 1111111, а, наприклад, двадцять шостому значенню – 0011011. Тоді число необхідних для кодування параметрів входів можна визначити як

$$N = \mathbf{log}_2\, n \qquad (16)$$

де *n*- кількість значень параметра, *N*- кількість входів

Для мережі необхідно щоб вхідні дані були в діапазоні [0..1], у той час як дані предметної області можуть бути у довільному діапазоні. Припустимо що дані по одному з параметрів є в діапазоні [Min..Max]. Тоді, найбільш простим способом нормування буде

$$\tilde{x} = \frac{x - Min}{Max - Min} \qquad (17)$$

де *x*- вихідне значення параметра, $\tilde{x}$ -значення, що подається на вхід мережі.

На жаль цей спосіб кодування не позбавлено недоліків. Так у випадку якщо *Max* >> $\tilde{x}$, то розподіл даних на вході може бути вкрай нерівномірним, що приведе до погіршення якості навчання. Тому в подібних ситуаціях, а також у випадку, коли значення входу лежить у діапазоні $[0;\infty]$ можна використати нормування за допомогою функції виду

$$\tilde{x} = \frac{1}{1 + e^{-x}} \qquad (18)$$