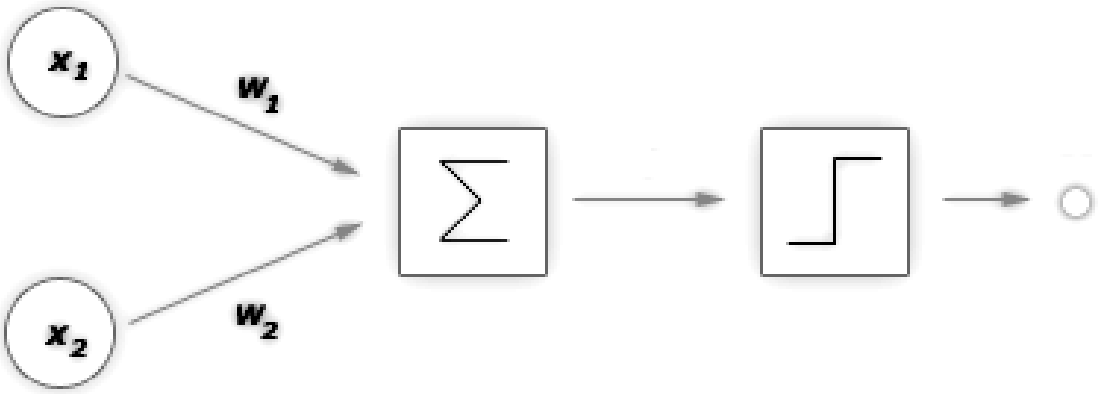
Проблема лінійного розподілу в НМ

Часто, для того, щоб продемонструвати обмежені можливості одношарових персептронів при рішенні завдань удаються до розгляду так званої проблеми XOR - що виключає АБО.

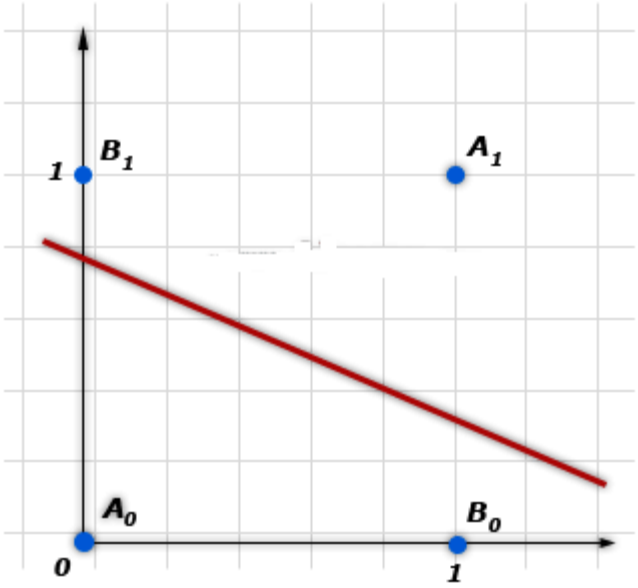
Суть завдання полягають в наступному. Дана логічна функція XOR - що виключає АБО. Це функція від двох аргументів, кожен з яких може бути нулем або одиницею. Вона набуває значення, коли один з аргументів дорівнює одиниці, але не обоє, інакше . Проблему можна проілюструвати за допомогою одношарової одно нейронної системи з двома входами, показаної на малюнку нижче.


x_1
w_1
x_2
w_2

Позначимо один вхід через X1, а інший через X2, тоді усі їх можливі комбінації складатимуться з чотирьох точок на площині.

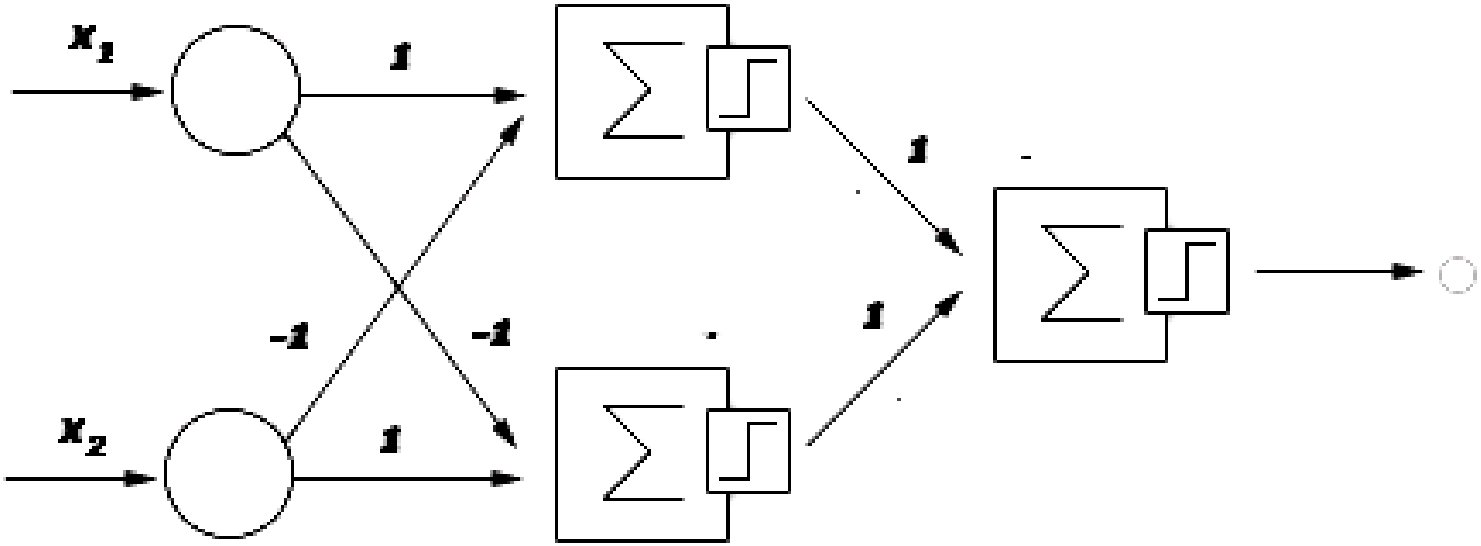
Один нейрон з двома входами може сформувати вирішальну поверхню у вигляді довільної прямої:

$$X1*W1+X2*W2 =T$$

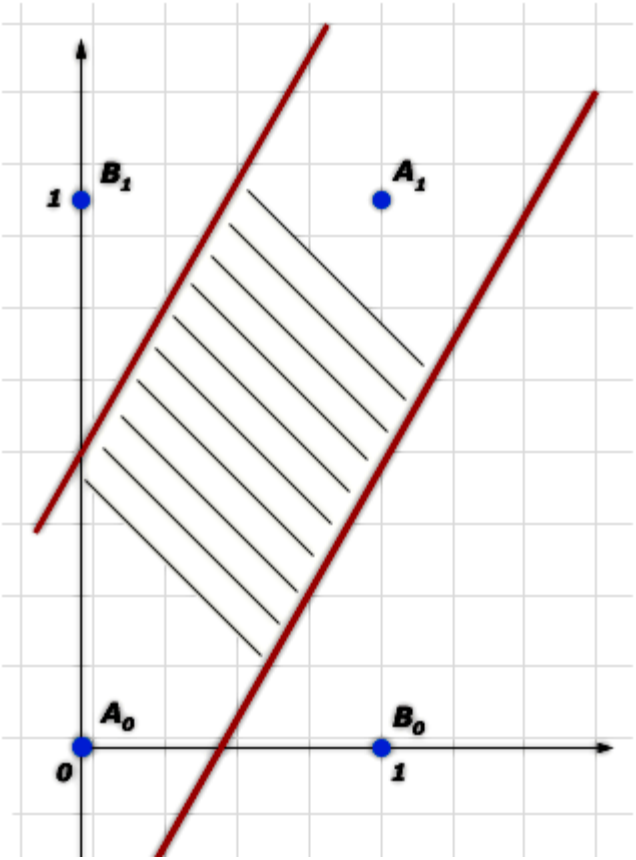

1
B_1
A_1
A_0
B_0
0
1

- Це означає, що які б значення не приписувалися вагам і порогу, одношарова нейронна мережа нездатна відтворити співвідношення між входом і виходом, потрібне для представлення функції XOR.

Проте функція XOR легко формується вже двошаровою мережею, причому багатьма способами. Розглянемо один з таких способів. Модернізуємо мережу, додавши ще один прихований шар нейронів


x_1
x_2
1
1
-1
-1
1
1

- Кожен з двох нейрон першого шару формує вирішальну поверхню у вигляді довільної прямої (ділить площину на дві напівплощини), а нейрон вихідного шару об'єднує ці два рішення, утворюючи вирішальну поверхню у вигляді смуги, утвореної паралельними прямими нейронів першого шару,


B1
A1
1
A0
B0
0
1

Навчання НМ без вчителя

Навчання людського мозку, на перший погляд, відбувається без вчителя: на зорові, слухові, тактильні та інші рецептори поступає інформація ззовні, і усередині нервової системи відбувається якась самоорганізація.

Проте, не можна заперечувати і того, що в житті людини не мало вчителів - і в буквальному, і у переносному розумінні, - які координують зовнішні дії.

Разом в тим, чим би не закінчилася суперечка прихильників цих двох концепцій навчання, вони обидві мають право на існування. Головна межа, що робить навчання без вчителя привабливим, - це його "самостійність".

Очевидно, що підстроювання синапсів може проводитися тільки на підставі інформації, доступної в нейроні, тобто його стану і вже наявних вагових коефіцієнтів. Виходячи з цього міркування і, що важливіше, по аналогії з відомими принципами самоорганізації нервових клітин, побудовані алгоритми навчання Хебба.

Сигнальний метод навчання Хебба

полягає в зміні ваг за наступним правилом:

$$w_{ij}(t)=w_{ij}(t-1)+\eta * y_i^{(n-1)} * y_j^{(n)} \qquad (1)$$

де $y_i^{(n-1)}$ - вхідне значення нейрону і шару; $y_i^{(0)} = x_i$;

$y_j^{(n)}$ - вихідне значення нейрона j шару n;

$w_{ij}(t)$, $w_{ij}(t-1)$ - ваговий коефіцієнт синапсу, що сполучає ці нейрони, на ітераціях t і $t-1$ відповідно;

η- коефіцієнт швидкості навчання.

Тут і далі, для спільності, під n мається на увазі довільний шар мережі. При навчанні по даному методу посилюються зв'язки між збудженими нейронами.

Існує також і диференціальний метод навчання Хебба

$$w_{ij}(t)=w_{ij}(t-1)+\eta * [y_i^{(n-1)}(t) - y_i^{(n-1)}(t-1)] * [y_i^{(n)}(t) - y_i^{(n)}(t-1)] \quad (2)$$

Повний алгоритм Хебба навчання без учителя

1. На стадії ініціалізації всім ваговим коефіцієнтам привласнюються невеликі випадкові значення.
2. На вході мережі подається вхідний образ, і сигнали збудження розповсюджуються по всіх шарах згідно принципам класичних прямоточних (FeedForward) мереж тобто для кожного нейрона розраховується зважена сума його входів, до якої потім застосовується активаційна функція нейрона, внаслідок чого виходить його вихідне значення ***yi(n)***, i=0...Mi, де Mi - число нейронів в шарі i; n=1...N, а N - число шарів в мережі.
3. На підставі набутих вихідних значень нейронів по формулах проводиться зміна вагових коефіцієнтів.
4. Цикл повторюється з кроку 2, поки вихідні значення мережі не стабілізуються із заданою точністю. Застосування цього нового способу визначення завершення навчання, відмінного від зворотного розповсюдження, обумовлене тим, що підстроювані значення синапсів фактично не обмежені.

На другому кроці циклу поперемінно пред'являються всі образи з вхідного набору. Слід зазначити, що вид відгуків на кожен клас вхідних образів невідомий заздалегідь і буде довільним поєднанням станів нейронів вихідного шару, обумовлене випадковим розподілом ваг на стадії ініціалізації.

Разом з тим, мережа здатна узагальнювати схожі образи, відносячи їх до одного класу.

Алгоритм навчання без учителя - алгоритм Кохонена

- передбачає підстроювання синапсів на підставі їх значень від попередньої ітерації

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \eta[y_i^{(n-1)} - w_{ij}^{(n)}(t-1)] \quad (3)$$

З наведеної вище формули видно, що навчання зводиться до мінімізації різниці між вхідними сигналами нейрона, що поступають з виходів нейронів попереднього шару

$y_i^{(n-1)}$ і ваговими коефіцієнтами його синапсів $w_{ij}^{(n)}$.

Повний алгоритм навчання має приблизно таку ж структуру, як в методах Хебба, але на кроці 3 з усього шару вибирається нейрон, значення синапсів якого максимально схожі на вхідний образ, і підстроювання ваг по формулі (3) проводиться тільки для нього.

Алгоритм навчання без учителя - алгоритм Кохонена

- Повний алгоритм навчання має приблизно таку ж структуру, як в методах Хебба, але на кроці 3 з усього шару вибирається нейрон, значення синапсів якого максимально схожі на вхідний образ, і підстроювання ваг по формулі (3) проводиться тільки для нього.
- Ця, так звана, акредитація може супроводжуватися гальмуванням усіх інших нейронів шару і введенням вибраного нейрона в насичення.
- Вибір такого нейрона може здійснюватися, наприклад, розрахунком скалярного добутку вектору вагових коефіцієнтів на вектор вхідних значень. Максимальне значення дає нейрон, що виграв.

Алгоритм навчання без учителя - алгоритм Кохонена

Інший варіант - розрахунок відстані між цими векторами в p - мірному просторі, де p - розмірність векторів

$$D_j = \sqrt{\sum_{i=0}^{p-1} (y_i^{(n-1)} - w_{ij})^2} \quad (4)$$

де j - індекс нейрона в шарі n, i - індекс підсумовування по нейронах шару (n - 1),

w_{ij} - вага синапсу, що сполучає нейрони; виходи нейронів шару (n - 1) є вхідними значеннями для шару n.

Корінь у формулі (4) брати не обов'язково, оскільки важлива лише відносна оцінка різних Dj. В даному випадку, "перемагає" нейрон з найменшою відстанню.

Іноді занадто часто нейрони, що одержують акредитацію примусово виключаються з розгляду, щоб "зрівняти права" усіх нейронів шару. Простий варіант такого алгоритму полягає в гальмуванні нейрона, що тільки що виграв.

Алгоритм навчання без учителя - алгоритм Кохонена

При використанні навчання по алгоритму Кохонена існує практика нормалізації вхідних образів, а так само - на стадії ініціалізації - і нормалізації початкових значень вагових коефіцієнтів

$$x_i = \frac{x_i}{\sqrt{\sum_{j=1}^{n-1} x_j^2}} \qquad (5)$$

де *xi* - і -а компонента вектору вхідного образу або вектору вагових коефіцієнтів, а n - його розмірність.

Це дозволяє скоротити тривалість процесу навчання. Ініціалізація вагових коефіцієнтів випадковими значеннями може привести до того, що різні класи, яким відповідають щільно розподілені вхідні образи, зіллються або, навпаки, роздрібняться на додаткові підкласи у разі близьких образів одного і того ж класу.

Алгоритм навчання без учителя - алгоритм Кохонена

Для уникнення такої ситуації використовується метод опуклої комбінації[3]. Суть його зводиться до того, що вхідні нормалізовані образи піддаються перетворенню :

$$x_i = \alpha x_i + \frac{1 - \alpha}{\sqrt{n}} \qquad (6)$$

де xi - i -а компонента вхідного образу, n - загальне число його компонент, α(t) - коефіцієнт, що змінюється в процесі навчання від нуля до одиниці, внаслідок чого спочатку на входи мережі подаються практично однакові образи, а з часом вони все більше сходяться до початкових. Вагові коефіцієнти встановлюються на кроці ініціалізації рівними величині

$$w_i = \frac{1}{\sqrt{n}} \qquad (7)$$

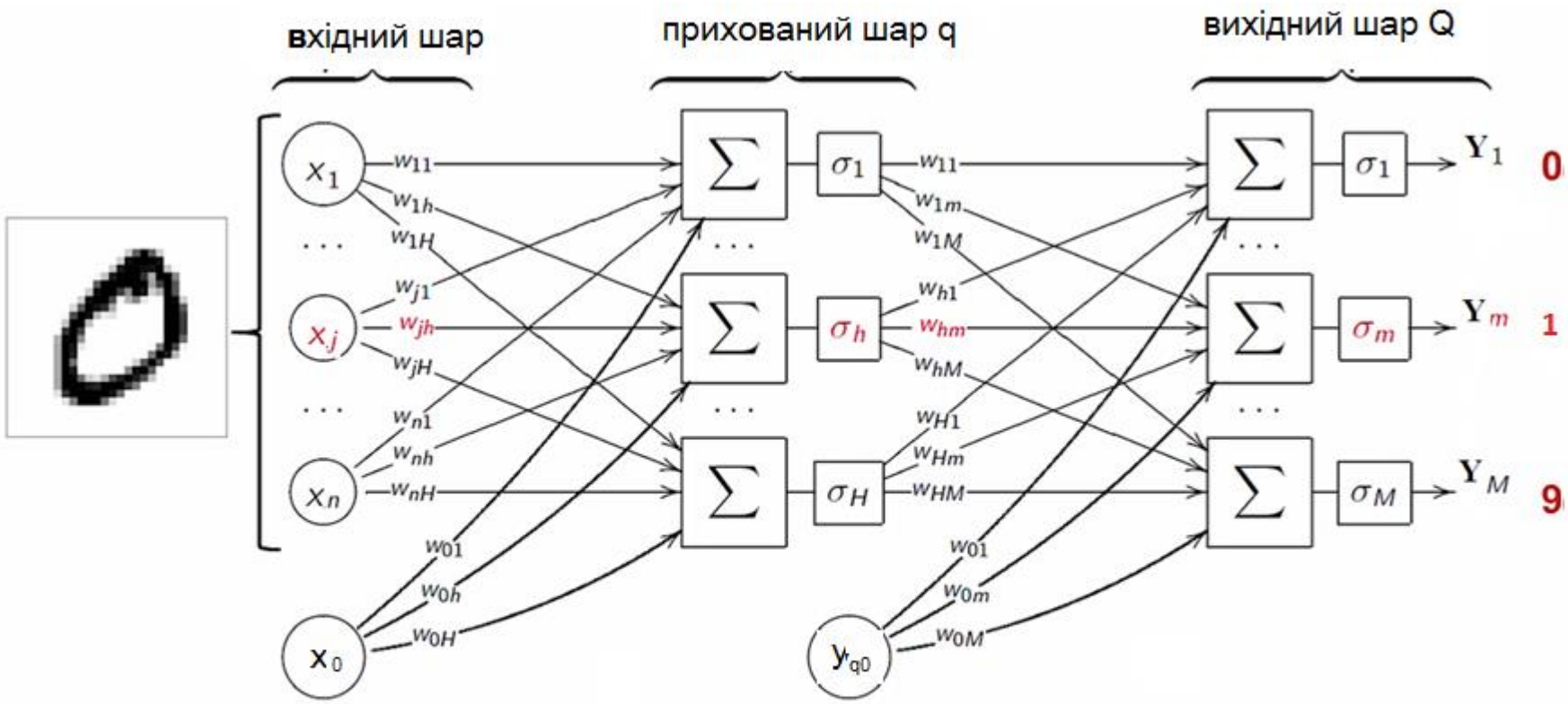
де n - розмірність вектору ваг для нейронів ініціалізованого шару.

Алгоритм навчання без учителя - алгоритм Кохонена

На основі розглянутого вище методу будуються нейронні мережі особливого типу - так звані структури, що самоорганізуються, - **self - organizing feature maps** (цей сталий переклад з англійського, на мій погляд, не дуже вдалий, оскільки, йдеться не про зміну структури мережі, а тільки про підстроювання синапсів). Для них після вибору з шару n нейрона j з мінімальною відстанню Dj (4) навчається по формулі (3) не лише цей нейрон, але і його сусіди, розташовані в околі R.

Величина R на перших ітераціях дуже велика, так що навчаються усі нейрони, але з часом вона зменшується до нуля. Таким чином, чим ближче кінець навчання, тим точніше визначається група нейронів, що відповідають кожному класу образів.

Багатошарова НМ



Градієнтний спуск

Цільова функція помилки

$$E(w_{ij}^{(q)}) = \frac{1}{2}\sum_{p}\sum_{j=1}^{M} (y_{j,p}^{(Q)} - d_{j,p})^2 \to \min$$

Мінімізуємо функцію помилки для кожного прикладу по черзі

$$E_p(w_{ij}^{(q)}) = \frac{1}{2}\sum_{j=1}^{M} (y_{j,p}^{(Q)} - d_{j,p})^2 = \frac{1}{2}\sum_{j=1}^{M} \Delta_{j,p}^2 \to \min$$

Тоді

$$E(w_{ij}^{(q)}) = \sum_{p=1}^{P} E_p(w_{ij}^{(q)})$$

Градієнтний спуск

Згідно з методом градієнтного спуску - крок

$$w_{ij}^{(q)}(t+1) = w_{ij}^{(q)}(t) + \Delta w_{ij}^{(q)} = w_{ij}^{(q)}(t) - \eta \cdot \frac{\partial E_p(w_{ij}^{(q)}(t))}{\partial w_{ij}^{(q)}}$$

Так як $\Delta_{j,p} = (y_{j,p}^{(Q)} - d_{j,p})$

То при $q=Q$ $$\frac{\partial E_p}{\partial w_{ij}^{(Q)}} = \frac{\partial E_p}{\partial \Delta_{j,p}} \cdot \frac{\partial \Delta_{j,p}}{\partial y_{j,p}^{(Q)}} \cdot \frac{\partial y_{j,p}^{(Q)}}{\partial w_{ij}^{(Q)}}$$

$$\frac{\partial E_p}{\partial \Delta_{j,p}} = \Delta_{j,p}; \frac{\partial \Delta_{j,p}}{\partial y_{j,p}^{(Q)}} = 1;$$

Оскільки $$y_{j,p}^{(Q)} = F(\sum_{i=0}^{n_{Q-1}} w_{ij}^{(Q)} y_{i,p}^{(Q-1)}) = F(s_{j,p}^{(Q)})$$

Градієнтний спуск

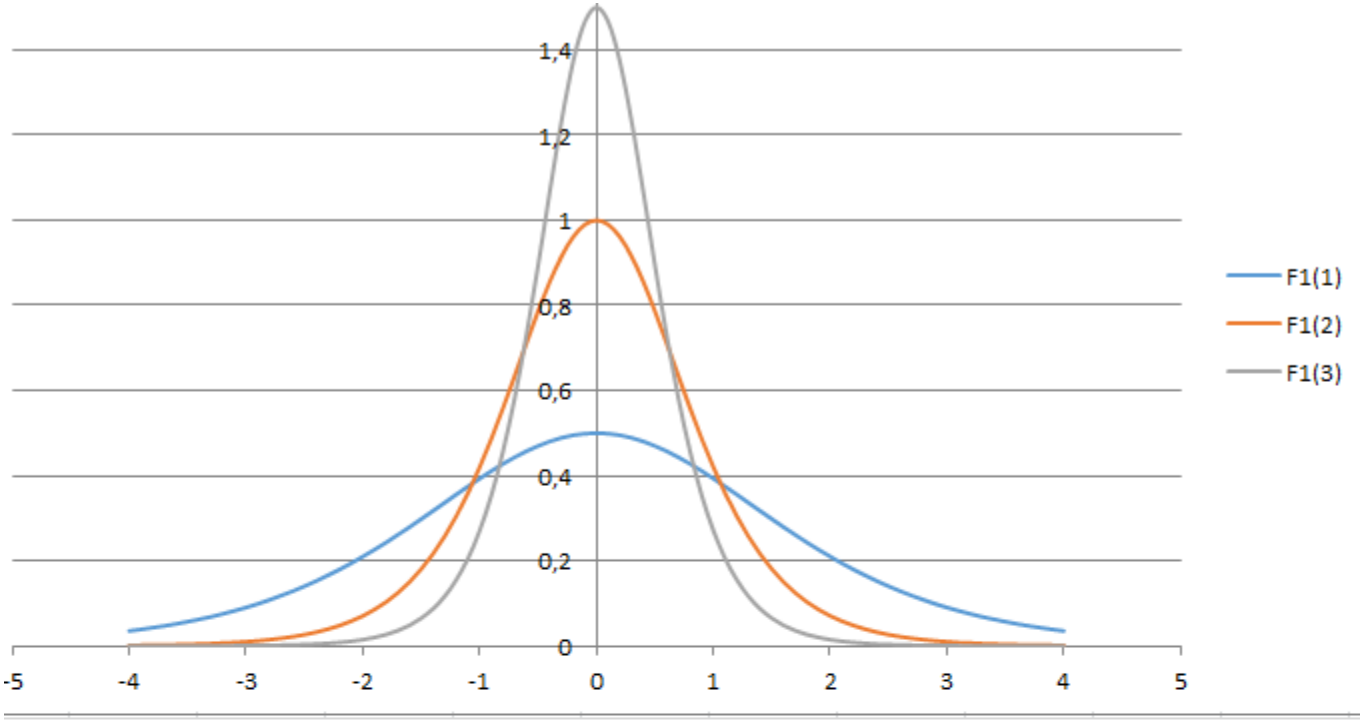
то

$$\frac{\partial y_{j,p}^{(Q)}}{\partial w_{ij}^{(Q)}} = F'(s_{j,p}^{(Q)}) \cdot \frac{\partial s_{j,p}^{(Q)}}{\partial w_{ij}^{(Q)}} = F'(s_{j,p}^{(Q)}) \cdot y_{i,p}^{(Q-1)}$$

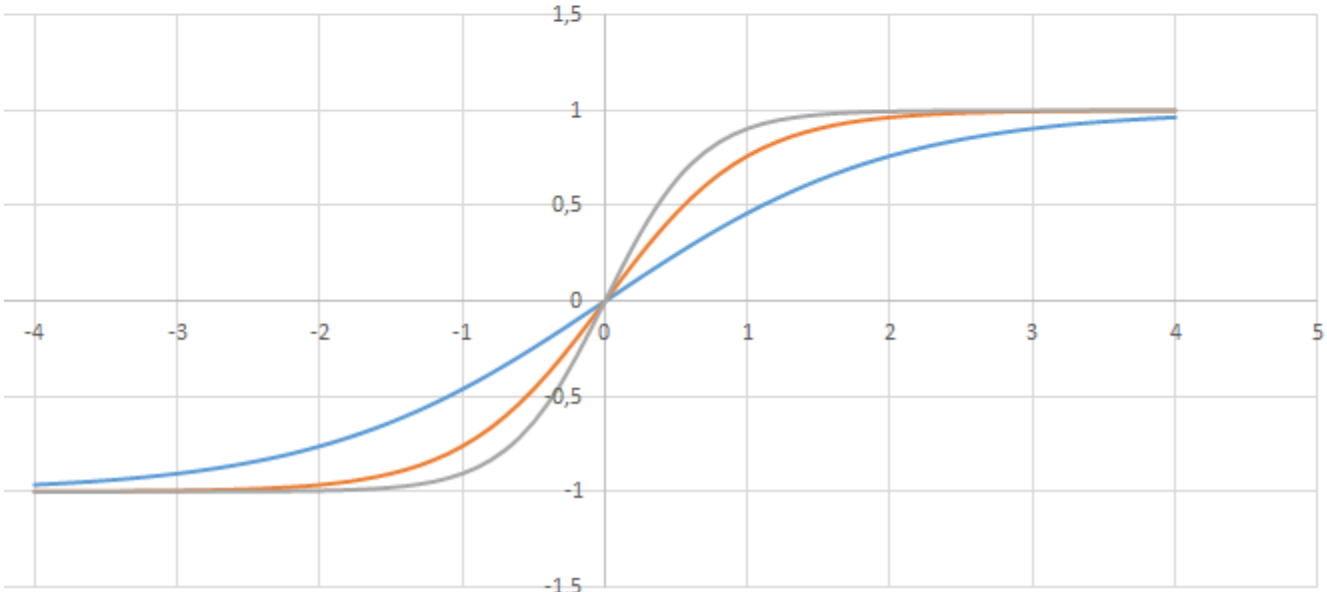
Оскільки множник $\frac{\partial y_{j,p}^{(Q)}}{\partial w_{ij}^{(Q)}}$ є похідною функції по її аргументу, з цього виходить, що похідна активаційної функція має бути визначена на усій осі абсцис, тому застосовуються такі функції, як гіперболічний тангенс або класичний сигмоїд з експонентою.

$$F(s) = \frac{2}{1+e^{-a \cdot s}} - 1 = th(\frac{a \cdot s}{2})$$

$$F'(s) = \frac{a}{2 \cdot ch^2(\frac{a \cdot s}{2})} = \frac{a}{2}(1 - th^2(\frac{a \cdot s}{2})) = \frac{a}{2}(1 - F^2(s))$$


1,4
1,2
1
0,8
0,6
0,4
0,2
0
-5
-4
-3
-2
-1
0
1
2
3
4
5
F1(1)
F1(2)
F1(3)

Сігмоїд


1,5
1
0,5
0
-0,5
-1
-1,5
-4
-3
-2
-1
0
1
2
3
4
5

Градієнтний спуск

При одношаровій мережі Q=1, $y^{(0)}=x_i$

$$\frac{\partial E_p}{\partial w_{ij}^{(Q)}} = \Delta_{j,p} \cdot F'(s_{j,p}^{(Q)}) \cdot x_{i,p}$$

Формула для уточнення $w_{ij}^{(Q)}$ матиме вигляд

$$w_{ij}^{(Q)}(t+1) = w_{ij}^{(Q)}(t) - \eta \cdot (\Delta_{j,p} \cdot F'(s_{j,p}^{(Q)})) \cdot x_{i,p}$$

І процес навчання буде повторювати процедуру Відроу-Хебба, тобто уточнення

Для багатошарових НМ невідомі помилки у проміжних шарах

Алгоритм Back propagation error

Запропонований Дэвидом И. Румельхартом, Дж. Е. Хинтоном и Рональдом Дж. Вильямсом в работе

***Rumelhart D.E., Hinton G.E., Williams R.J.*, Learning Internal Representations by Error Propagation. In: Parallel Distributed Processing, vol. 1, pp. 318—362. Cambridge, MA, MIT Press. 1986.**

Згідно з методом найменших квадратів, цільовою функцією помилки НС, що мінімізується, є величина:

$$E(w) = \frac{1}{2}\sum_{j,p}(y_{j,p}^{(Q)} - d_{j,p})^2 \qquad (2.1)$$

де $y_{j,p}^{(N)}$ реальний вихідний стан нейрона j вихідного шару N нейронної мережі при подачі на її входи p -го образу;

d_{jp} - ідеальний (бажане) вихідний стан цього нейрона.

Підсумовування ведеться по усіх нейронах вихідного шару і по усіх оброблюваних мережею образах.

Алгоритм Back propagation error

Мінімізація ведеться методом градієнтного спуску, що означає підстроювання вагових коефіцієнтів таким чином:

$$w_{ij}^{(q)}(t+1) = w_{ij}^{(q)}(t) + \Delta w_{ij}^{(q)} = w_{ij}^{(q)}(t) - \eta \cdot \frac{\partial E_p}{\partial w_{ij}^{(q)}} \qquad (2)$$

Тут $\boldsymbol{w_{ij}}$ - ваговий коефіцієнт синаптичного зв'язку, що сполучає і –й нейрон шару n – 1 з j –м нейроном шару n,

η - коефіцієнт швидкості навчання, 0< η <1.

Очевидно що,

$$\frac{\partial E}{\partial w_j^{(q)}} = \frac{\partial E}{\partial y_j^{(q)}} \cdot \frac{dy_j^{(q)}}{ds_j^{(q)}} \cdot \frac{\partial s_j^{(q)}}{\partial w_j^{(q)}} \qquad (3)$$

Тут під y_j, як і раніше, мається на увазі вихід нейрона j, а під s_j – зважена сума його вхідних сигналів, тобто аргумент активаційної функції.

$$\frac{dy}{ds} = F'(s) = \frac{a}{2}(1 - y^2) \qquad (4)$$

Алгоритм Back propagation error

Третій множник ds_j/dw_{ij}, очевидно, дорівнює виходу нейрона попереднього шару: $y_i^{(q-1)}$.

$$s_j^{(q)} = \sum_{i=0}^{n_q} y_i^{(q-1)} \cdot w_{ij}^{(q)} \qquad s_j^{(q+1)} = \sum_{i=0}^{n_q} y_i^{(q)} \cdot w_{ij}^{(q+1)}$$

Що стосується першого множника в (3), він легко розкладається таким чином:

$$\frac{\partial E}{\partial y_j^{(q)}} = \sum_{i=0}^{n_{q+1}} \frac{\partial E}{\partial y_i^{(q+1)}} \cdot \frac{dy_i^{(q+1)}}{ds_i^{(q+1)}} \cdot \frac{\partial s_i^{(q+1)}}{\partial y_i^{(q+1)}} = \sum_{i=0}^{n_{q+1}} \left(\frac{\partial E}{\partial y_i^{(q+1)}} \cdot \frac{dy_i^{(q+1)}}{ds_i^{(q+1)}}\right) \cdot w_{ji}^{(q+1)} \qquad (5)$$

Тут підсумовування по і виконується серед нейронів шару *n+1*.

Ввівши нову змінну

$$\delta_j^{(q)} = \frac{\partial E}{\partial y_j^{(q)}} \cdot \frac{dy_j^{(q)}}{ds_j^{(q)}} \qquad (6)$$

отримаємо рекурсивну формулу для розрахунків величин (j(n) шару n з величин (k(n+1) більше старшого шару n+1.

$$\delta_j^{(q)} = \left[\sum_{i=0}^{n_{q+1}} \delta_k^{(q+1)} \cdot w_{jk}^{(q+1)}\right] \cdot F'(s_j^{(q)}) \qquad (7)$$

Для вихідного ж шару

$$\delta_j^{(Q)} = (y_j^{(Q)} - d_j) \cdot F'(s_j^{(Q)})$$

Алгоритм Back propagation error

(8)

Тепер ми можемо записати (2) в розкритому виді:

$$\delta_j^{(q)} = \left[\sum_{i=0}^{n_{q+1}} \delta_k^{(q+1)} \cdot w_{jk}^{(n+1)} \right] \cdot F'(s_j^{(q)}) \qquad (9)$$

Іноді для надання процесу корекції ваг деякої інерційності, що згладжує різкі скачки при переміщенні по поверхні цільової функції, (9) доповнюється значенням зміни ваги на попередній ітерації

$$\Delta w_{ij}^{(q)} = -\eta \cdot \delta_j^{(q)} \cdot y_i^{(q-1)} \qquad (10)$$

$$\Delta w_{ij}^{(q)}(t+1) = -\eta \cdot (\mu \cdot \Delta w_{ij}^{(q)}(t) + (1-\mu) \cdot \delta_j^{(q)} \cdot y_i^{(q-1)})$$

де μ - коефіцієнт інерційності, t - номер поточної ітерації.

Реалізація Backpropogation

Алгоритм навчання багатошарової НМ за допомогою процедури зворотного поширення помилки :

0. Зформувати навчальну вибірку $\{\mathbf{x}_p, \mathbf{d}_p\}$, p=1…P

ініціалізувати початкові значення вагових параметрів

$$w_{ij} = 0.01*rand(-1,+1).$$

1. Подати на входи мережі один прикладів навчальної вибірки і в режимі звичайного функціонування НМ, коли сигнали поширюються від входів до виходів, розрахувати значення останніх. Нагадаємо, що

$$s_j^{(q)} = \sum_{i=0}^{n_{q-1}} y_i^{(q-1)} \cdot w_{ij}^{(q)}$$

де n_{q-1} - число нейронів в шарі q-1 з урахуванням нейрона з постійним вхідним станом +1, що задає зміщення;

$$y_j^{(q)} = F(s_j^{(q)}),$$

де F() – сигмоїд

$$y_i^{(0)} = x_i$$

де x_i - і -та компонента вектору вхідного образу.

2. Розрахувати для вихідного шару по формулі

$$\delta_j^{(Q)} = (y_j^{(Q)} - d_j) \cdot F'(s_j^{(Q)})$$

Реалізація Backpropogation

3. Розрахувати по формулах відповідні для усіх інших шарів, q=(Q – 1),..1.

$$\delta_j^{(q)} = \left[\sum_{i=0}^{n_{q+1}} \delta_k^{(q+1)} \cdot w_{jk}^{(q+1)} \right] \cdot F'(s_j^{(q)})$$

4. Розрахувати по формулі

$$\Delta w_{ij}^{(q)} = -\eta \cdot \delta_j^{(q)} \cdot y_i^{(q-1)}$$

зміни ваг шару q.

5. Скоригувати усі ваги в НМ

$$w_{ij}^{(q)}(t) = w_{ij}^{(q)}(t-1) + \Delta w_{ij}^{(q)}(t)$$

6. Перейти на крок 1 до вичерпання всіх прикладів навчальної вибірки.

7. Якщо функція помилки мережі E для всіх M прикладів навчальної вибірки істотна, перейти на крок 1. Інакше - кінець.

$$E' = \frac{1}{2} \sum_{p=1}^{P} \sum_{j=1}^{n_Q} \Delta_{jp}^2$$

де

$\Delta_{jp} = (y_j^{(Q)} - d_j)_p$

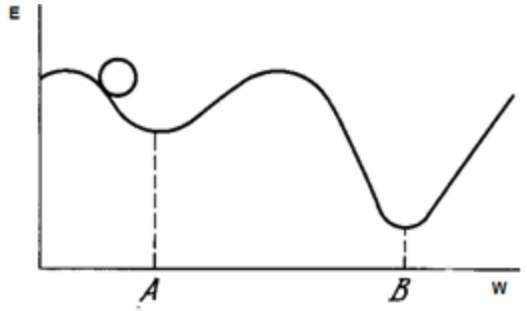
Мережі на кроці 1 попеременно у випадковому порядку пред'являються усі тренувальні приклади.

Налаштування НМ для вирішення задач

- Локальні мінімуми.
- Параліч НМ
- Розмір кроку.
- Тимчасова нестійкість.
- Попередня обробка вхідних даних.
- Згасання помилки у алгоритмі Backprop.

Локальні мінімуми

BackProp використовує різновид градієнтного спуску, тобто здійснює спуск вниз по поверхні помилки, безперервно налаштовуючи ваги у напрямі до мінімуму. Поверхня помилки складної мережі дуже порізана і складається з пагорбів, долин, складок і ярів в просторі високої розмірності.



Мережа може потрапити в локальний мінімум (неглибоку долину), коли поруч є набагато глибший мінімум. У точці локального мінімуму усі напрями ведуть вгору, і мережа нездатна з нього вибратися.

Статистичні методи навчання можуть допомогти уникнути цієї пастки, але вони повільні. Застосовується також метод «струшування», тобто зміна деяких вагових параметрів для виходу з локального мінімуму і продовження процесу навчання.

Параліч мережі

В процесі навчання мережі значення ваг можуть в результаті корекції стати дуже великими величинами. Це може привести до того, що усі або більшість нейронів функціонуватимуть при дуже великих значеннях S, в області, де похідна функції дуже мала. Оскільки помилка пропорційна цій похідній, то процес навчання може практично завмерти.

Різні евристики використовувалися для оберігання від паралічу або для відновлення після нього, наприклад тимчасове зменшення параметра сигмоїду **a**, оскільки причиною паралічу НМ часто є нульове значення похідної від активаційної функції $F'(S) = \boldsymbol{\alpha}(1-F^2(S))$ для великих значень S і це зупиняє процес навчання. Після відновлення навчання значення параметра $\boldsymbol{\alpha}$ повертається.

Розмір кроку

Уважний розбір збіжності показує, що корекції ваг передбачаються нескінченно малими. Ясно, що це нездійсненно на практиці, оскільки веде до нескінченного часу навчання.

Розмір кроку **η** повинен братися скінченним, і в цьому питанні доводиться спиратися тільки на досвід.

Якщо розмір кроку **η** дуже малий, то збіжність занадто повільна, якщо ж дуже великий, то може виникнути параліч або постійна нестійкість.

Автоматичне налаштування кроку може здійснюватись по формулі: $\eta = \eta_0/(1+b*t)$

t - номер епохи навчання,
η_0 і **b –** обираються експериментально.

Тимчасова нестійкість

Якщо мережа вчиться розпізнавати букви, то немає сенсу учити "Б", якщо при цьому забувається "А". Процес навчання має бути таким, щоб мережа навчалася на усій навчальній множині без пропусків того, що вже вивчене.

Необхідні зміни ваг повинні обчислюватися **на усій множині прикладів,** а це вимагає додаткової пам'яті; після ряду таких навчальних циклів вони зійдуться до мінімальної помилки.

Цей метод може виявитися даремним, якщо мережа знаходиться в зовнішньому середовищі, що постійно міняється, так що другий раз один і той же вектор може вже не повторитися. В цьому випадку процес навчання може ніколи не зійтися, безцільно блукаючи або сильно осцилюючи. У цьому сенсі зворотне поширення не схоже на біологічні системи.

Попередня обробка вхідних даних.

Процес навчання можна суттєво покращити попередньою обробкою навчальної вибірки.

Приклади для кожного класу повинні бути типовими проте різними. При майже однакових прикладах спостерігається синдром «перенавчання», коли нейронна мережа у процесі функціонування добре розпізнає образи близькі до прикладів і не розпізнають ся віддалені образи.

Разом з тим, дуже «зашумлені» приклади у навчальній вибірці не дозволяють навчити НС взагалі.

Суттєво може прискорити навчання попередня нормалізація навчальної вибірки, наприклад методом природної нормалізації.

Функції активації

- Логістичний сигмоїд $F(s)=\frac{1}{1+e^{-a\cdot s}}$ $F'(s)=a\cdot F(s)\cdot(1-F^2(s))$
- Гіперболічний тангенс $F(s)=th(as)$ $F'(s)=a(1-F^2(s))$
- SoftSign $F(s)=\frac{a\cdot s}{1+|a\cdot s|}$ $F'(s)=\frac{1}{(1+|a\cdot s|)^2}$
- SoftPlus $F(s)=\ln(1+e^{a\cdot s})$ $F'(s)=\frac{a}{1+e^{-a\cdot s}}$
- ReLU(rectified linear units) $F(s)=\max(0,s)$

$$F'(s)=\max(0,1)$$

Згасання помилки у алгоритмі BackProp.

Спостерігається згасання помилки при зворотному розповсюдженні, що зупиняє процес навчання.

Виходом можуть бути застосування інших методів навчання, зокрема, чисельних методів оптимізації чи обмеженої машини Больцмана.