5.1. Алгоритм зворотного розповсюдження помилки. Розглянемо ідею одного з найпоширеніших алгоритмів навчання – алгоритм зворотного розповсюдження помилки (back propagation). Це ітеративний градієнтний алгоритм навчання, який використовується з метою мінімізації середньоквадратичного відхилення поточного виходу і бажаного виходу багатошарових нейронних мереж.

Алгоритм зворотного розповсюдження використовується для навчання багатошарових нейронних мереж з послідовними зв'язками. Як зазначено вище, нейрони в таких мережах діляться на групи із загальним вхідним сигналом – шари, при цьому на кожний нейрон першого шару подаються всі елементи зовнішнього вхідного сигналу, а всі виходи нейронів того шару подаються на кожний нейрон шару *(q+1).* Нейрони виконують зважене (з синаптичними вагами) підсумовування елементів вхідних сигналів; до даної суми додається зсув нейрона. Над отриманим результатом виконується активаційною функцією нелінійне перетворення. Значення функції активації є вихід нейрона.

У багатошарових мережах оптимальні вихідні значення нейронів всіх шарів, окрім останнього, як правило, невідомі, і трьох- або більш шаровий персептрон вже неможливо навчити, керуючись тільки величинами помилок на виходах НМ. Найбільш прийнятним варіантом навчання в таких умовах виявився градієнтний метод пошуку мінімуму функції помилки з розглядом сигналів помилки від виходів НМ до її входів, в напрямку, зворотному прямому розповсюдженню сигналів у звичайному режимі роботи. Цей алгоритм навчання НМ одержав назву процедури зворотного розповсюдження.

У такому алгоритмі функція помилки є сумою квадратів помилки мережі бажаного виходу і реального. При обчисленні елементів вектора-градієнта був використаний своєрідний вид похідних функцій активації сигмоїдального типу. Алгоритм діє циклічно (ітеративно), і його цикли прийнято називати епохами. На кожній епосі на вхід мережі по черзі подаються всі навчальні спостереження, вихідні значення мережі порівнюються з цільовими значеннями і вираховується помилка. Значення помилки, а також градієнта поверхні помилок використовується для корегування вагів, після чого всі дії повторюються. Початкова конфігурація мережі обирається випадковим чином, і процес навчання припиняється коли пройдена певна кількість епох, або коли помилка досягне деякого певного рівня малості, або коли помилка перестане зменшуватися (користувач може сам вибрати потрібну умову зупинки).

Приведемо мовний опис алгоритму.

Крок 1. Вагам мережі присвоюються невеликі початкові значення.

Крок 2. Вибирається чергова навчальна пара *(X, Y)*; вектор *X* подається на вхід мережі.

Крок 3. Обчислюється вихід мережі.

Крок 4. Обчислюється різниця між виходом мережі, що вимагається (цільовим, *Y*) і реальним (обчисленим).

Крок 5. Вага мережі корегується так, щоб мінімізувати помилку.

Крок 6. Кроки з 2-го по 5-й повторюються для кожної пари навчальної множини до тих пір, поки помилка на всій множині не досягне прийнятної величини.

Кроки 2 і 3 подібні тим, які виконуються у вже навченій мережі.

Обчислення у мережі виконуються пошарово. На кроці 3 кожний з виходів мережі віднімається з відповідної компоненти цільового вектора з метою отримання помилки. Ця помилка використовується на кроці 5 для корекції вагів мережі.

Кроки 2 і 3 можна розглядати як «прохід вперед», оскільки сигнал розповсюджується по мережі від входу до виходу. Кроки 4 і 5 складають «зворотний прохід», оскільки тут обчислюваний сигнал помилки розповсюджується назад по мережі і використовується для підстроювання вагів.

Математичне представлення алгоритму розглянемо спочатку на прикладі найпростішої нейронної мережі, що містить тільки один нейрон (умовно позначений колом (рис. 5.2).

Вважатимемо, що вихід (output) мережі o = *f(net)* визначається функцією активації сигмоїдного типу.

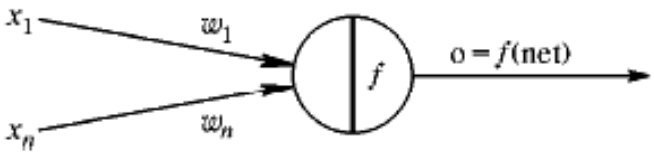

Рис. 5.2. НМ із одного нейрона

Припустимо, що для навчання мережі використовується вибірка:

$$x^1 = (x_1^1, x_2^1, \ldots, x_n^1)^T, \quad y^1,$$
$$x^2 = (x_1^2, x_2^2, \ldots, x_n^2)^T, \quad y^2,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$X^N = (x_1^N, x_2^N, \ldots, x_n^N)^T, \quad y^N.$$

де: y^k – значення бажаного (цільового) виходу.

Як функція помилки для *k*-го зразка (*k*-го елемента навчальної вибірки) приймемо величину, пропорційну квадрату різниці бажаного виходу і виходу мережі:

$$E_k = \frac{1}{2}(y^k - o^k)^2 = \frac{1}{2}(y^k - o^k(w^T x^k))^2 = \frac{1}{2}(y^k - \frac{1}{1+e^{-w^T x^k}})^2. \quad (5.1)$$

Відповідно, сумарна функція помилки по всіх елементах вибірки:

$$E = \sum_{k=1}^{N} E_k \quad (5.2)$$

Очевидно, як E_k, так і E є функціями вектора вагів мережі w. Задача навчання мережі зводиться у даному випадку до підбору такого вектора w, при якому досягається мінімум E. Таку задачу оптимізації вирішуватимемо градієнтним методом, використовуючи співвідношення:

$$w := w - \eta E_k'(w), \quad (5.3)$$

де: « : = » – оператор присвоєння,

$E'_k(w)$ – апозначення вектора-градієнта,

η – деяка константа.

Представляючи вектор у розгорненому вигляді для похідної сигмоїдної функції, одержимо:

$$E'_k(w) = \frac{d}{dw}\left(\frac{1}{2}\left(y^k - \frac{1}{1+e^{-w^T x^k}}\right)^2\right) = -(y^k - o^k)o^k(1-o^k)x^k. \quad (5.4)$$

Це дає можливість записати алгоритм корекції (підстроювання) вектора вагових коефіцієнтів мережі у формі:

$$w := w + \eta(y^k - o^k)o^k(1-o^k)x^k = w + \eta\delta_k x^k, \quad (5.5)$$

де:

$$\delta_k = (y^k - o^k)o^k(1-o^k). \quad (5.6)$$

Одержані математичні вирази повністю визначають алгоритм навчання даної НМ, який може бути представленний тепер у наступному вигляді.

1. Задаються деякі η $(0<\eta<1)$, E_{max} і деяка мала випадкова вага w_i мережі.
2. Задаються $k=1$ і $E=0$.
3. Вводиться чергова навчальна пара (x^k, y^k).

Проводяться позначення

$$x := x^k, \quad y := y^k$$

обчислюється величина виходу мережі:

$$o = o(w^T x) = \frac{1}{1+e^{-w^T x}}. \quad (5.7)$$

4. Обновляється (корегується) вага:

$$w := w + \eta(y-o)o(1-o)x. \quad (5.8)$$

5. Корегується (нарощується) значення функції помилки:

$$E := E + \frac{1}{2}(y-o)^2. \quad (5.9)$$

6. Якщо $k < N$, тоді $k := k+1$ і перехід до кроку 3, у протилежному випадку – перехід до кроку 7.
7. Завершення циклу навчання. Якщо $E<E_{max}$, то закінчення всієї процедури навчання. Якщо $E \geq E_{max}$, тоді починається новий цикл навчання переходом до кроку 2.

Розглянемо тепер більш загальний випадок, вважаючи, що двошарова НМ містить декілька (L) прихованих нейронів і один вихідний (рис. 5.3).

У даному випадку функція помилки залежить від векторів вагів прихованого шару і вектора вагів, пов'язаних з вихідним нейроном. Вихід мережі описується виразом:

$$O^k = \frac{1}{1+e^{-W^T o^k}}, \tag{5.10}$$

де: W – вектор вагів вихідного нейрона,

o^k – вектор виходів нейронів прихованого шару з елементами.

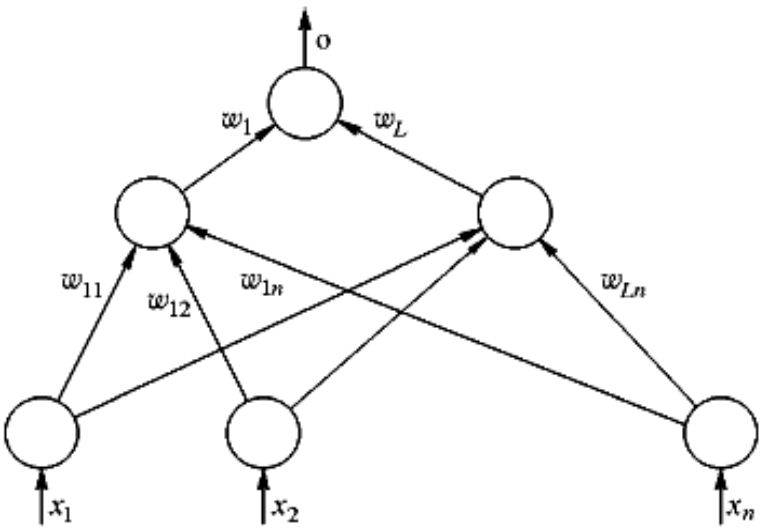

Рис. 5.2. Двошарова нейронна мережа

$$o_i{}^k = \frac{1}{1+e^{-w_i{}^T o^k}}, \tag{5.11}$$

де: w_i – вектор вагів, пов'язаних з i-м прихованим нейроном, $i=1,2,...,L$.

Правило коректування вагів у даній НМ також засновано на мінімізації квадратичної функції помилки градієнтним методом на основі виразів:

$$W := W - \eta \frac{\partial E_k(W,w)}{\partial W}, \tag{5.12}$$

$$w_i := w_i - \eta \frac{\partial E_k(W,w)}{\partial w_i}, \tag{5.13}$$

де: $\eta = const$ – коефіцієнт швидкості навчання $(0 < \eta < 1)$, $i=1,2,...,L$.

Використовуючи правило диференціювання складної функції і вираз для похідної сигмоїдної функції активації, отримаємо:

$$\frac{\partial E_k(W,w)}{\partial W} = \frac{1}{2}\frac{\partial}{\partial W}(y^k - \frac{1}{1-e^{-W^T O^k}})^2 = -(y^k - O^k)O^k(1-O^k)o^k, \tag{5.14}$$

звідки слідує:

$$W := W + \eta(y^k - O^k)O^k(1-O^k)o^k = W + \eta\delta_k o^k, \tag{5.15}$$

або в скалярній формі:

$$W_i := W_i + \eta\delta_k o_i^k, i = 1,2,\ldots,L, \quad (5.16)$$

де:

$$\delta_k = (y^k - O^k)O^k(1 - O^k). \quad (5.17)$$

Поступаючи аналогічно, знайдемо:

$$\frac{\partial E_k(W,w)}{\partial w_i} = -(y^k - O^k)O^k(1 - O^k)W_i o_i^k(1 - o_i^k)x^k, \quad (5.18)$$

звідки одержуємо:

$$w_i := w_i + \eta\delta_k W_i o_i^k(1 - o_i^k)x^k, \quad (5.19)$$

або (в скалярній формі):

$$w_{ij} = w_{ij} + \eta\delta_k W_i o_i^k(1 - o_i^k)x_j^k, i = 1,2,\ldots,L, j = 1,2,\ldots,n. \quad (5.20)$$

Алгоритм навчання може бути тепер представлений у вигляді слідуючих кроків:

1. Задаються деякі η $(0<\eta<1)$, E_{max} і деяка мала випадкова вага w_i мережі.
2. Задаються $k=1$ і $E=0$.
3. Вводиться чергова навчальна пара (x^k, y^k). Вводяться позначення:

$$x := x^k, \quad y := y^k, \quad (5.21)$$

і обчислюється величина виходу мережі:

$$O = \frac{1}{1 + e^{-W^T o}}, \quad (5.22)$$

де: W – вектор вагів вихідного нейрона, o^k – вектор виходів нейронів прихованого шару з елементами:

$$o_i = \frac{1}{1 + e^{-w_i^T x}}, \quad (5.23)$$

w_i позначає вектор вагів, пов'язаних з i-м прихованим нейроном, $i=1,2,\ldots,L$.

4. Проводиться коректування терезів вихідного нейрона:

$$W := W + \eta\delta o, \quad (5.24)$$

де:

$$\delta = (y - O)O(1 - O). \quad (5.25)$$

5. Коректується вага нейронів прихованого шару:

$$w_i := w_i + \eta\delta W_i o_i(1 - o_i), i = 1,2,\ldots,L. \quad (5.26)$$

6. Коректується (нарощується) значення функції помилки:

$$E := E + \frac{1}{2}(y - o)^2. \qquad (5.27)$$

Якщо $k<N$, тоді $k:=k+1$ і перехід до кроку 3, у протилежному випадку перехід на крок 8.

7. Завершення циклу навчання. Якщо $E<E_{max}$, то закінчення всієї процедури навчання. Якщо $E \geq E_{max}$, тоді починається новий цикл навчання з переходом до кроку 2.

Розглянута процедура може бути легко узагальнена на випадок мережі з довільною кількістю шарів і нейронів в кожному шарі.

Звернемо увагу, на те що в даній процедурі спочатку відбувається корекція вагів для вихідного нейрона, а потім – для нейронів прихованого шару, тобто від кінця мережі до її початку. Звідси і назва – зворотне розповсюдження помилки. Зважаючи на використання для позначень грецької букви δ, цю процедуру навчання називають ще іноді *узагальненим дельта-правилом*.

Дамо висловленому геометричну інтерпретацію.

У алгоритмі *зворотного розповсюдження* обчислюється вектор градієнта поверхні помилок. Цей вектор вказує напрямок найкоротшого спуску по поверхні з даної точки, тому, якщо ми «трохи» просунемося по ньому, помилка зменшиться. Послідовність таких кроків (сповільнююча по мірі наближення до дна) врешті-решт приведе до мінімуму того або іншого типу. Певну складність тут представляє питання про те, яку потрібно брати довжину кроків (що визначається величиною коефіцієнта швидкості навчання η).

При великій довжині кроку збіжність буде більш швидкою, але є небезпека перестрибнути через рішення або (якщо поверхня помилок має особливо складну форму) піти у неправильному напрямі.

Класичним прикладом такого явища при навчанні нейронної мережі є ситуація, коли алгоритм дуже повільно просувається по вузькому яру з крутими схилами, стрибаючи з однієї його сторони на іншу. Навпаки, при маленькому кроці, ймовірно, буде схоплений вірний напрям, проте при цьому буде потрібно дуже багато ітерацій.

На практиці величина кроку береться пропорційній крутизні схилу (отже алгоритм уповільнює хід поблизу мінімуму) з деякою константою (η), яка, як наголошувалося, називається коефіцієнтом швидкості навчання. Правильний вибір швидкості навчання залежить від конкретної задачі і звичайно здійснюється дослідним шляхом; ця константа може також залежати від часу, зменшуючись у міру просування алгоритму.

Звичайно алгоритм видозмінюється так, щоб включати доданок імпульсу (або інерції). Цей член сприяє просуванню у фіксованому напрямі, тому якщо було зроблено декілька кроків в одному і тому ж напрямі, то алгоритм «збільшує швидкість», що (іноді) дозволяє уникнути локального мінімуму, а також швидше проходити плоскі ділянки.

Таким чином, алгоритм діє ітеративно, і його кроки прийнято називати епохами. На кожній епосі на вхід мережі почерзі подаються всі навчальні

спостереження, вихідні значення мережі порівнюються з цільовими значеннями і обчислюється помилка. Значення помилки, а також градієнта поверхні помилки використовується для корегування вагів, після чого всі дії повторюються. Початкова конфігурація мережі вибирається випадковим чином, і процес навчання припиняється, або коли пройдена певна кількість епох, або коли помилка досягне деякого певного рівня малості, або коли помилка перипиняє зменшуватися (користувач може сам обрати потрібну умову зупинки).

Класичний метод зворотного розповсюдження відноситься до алгоритмів з лінійною збіжністю і відомими недоліками його є: невисока швидкість збіжності (велике число необхідних ітерацій для досягнення мінімуму функції помилки), можливість сходитися не до глобального, а до локальних рішень (локальним мінімумам відзначеної функції), можливість паралічу мережі (при якому більшість нейронів функціонує при дуже великих значеннях аргументу функцій активації, тобто на її пологій ділянці; оскільки помилка пропорційна похідній, яка на даних ділянках мала, то процес навчання практично завмирає).

Для усунення цих недоліків були запропоновані численні модифікації алгоритму зворотного розповсюдження, які зв'язані з використанням різних функцій помилки, різних процедур визначення напряму і величини кроку і т.ін.