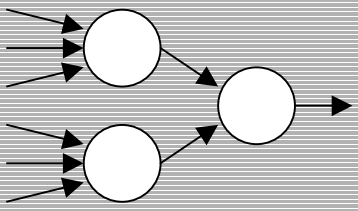
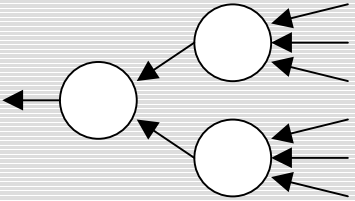


В. І. Дубровін
С.О. Субботін

МЕТОДИ ОПТИМІЗАЦІЇ ТА ЇХ ЗАСТОСУВАННЯ В ЗАДАЧАХ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ

навчальний посібник

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ



Міністерство освіти і науки України
Запорізький національний технічний університет

В.І. Дубровін, С.О. Субботін

МЕТОДИ ОПТИМІЗАЦІЇ ТА ЇХ ЗАСТОСУВАННЯ В ЗАДАЧАХ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ

Рекомендовано
Міністерством освіти і науки України
як навчальний посібник
для студентів вищих навчальних закладів

Запоріжжя
2003

ББК 22.18:32.97
Д79
УДК 519.6:004.93:007.52

Дубровін В.І., Субботін С.О.
Д79 Методи оптимізації та їх застосування в задачах навчання нейронних мереж: Навчальний посібник.-Запоріжжя: ЗНТУ, 2003.- 136 с.

ISBN 966-7809-24-2

Гриф надано Міністерством освіти і науки України
(лист № 14/18.2-486 від 12.03.2003 р.)

Рецензенти: **В.М. Порохня**, доктор технічних наук, професор

Л.О. Галкін, доктор технічних наук, професор

Розглянуто основні методи й алгоритми чисельної оптимізації. Приведено опис оригінальних авторських розробок і досліджень методів оптимізації. Особливу увагу приділено застосуванню методів оптимізації в задачах навчання штучних нейронних мереж.

Призначено для студентів спеціальності «Програмне забезпечення автоматизованих систем», а також може використовуватися інженерами, аспірантами і студентами різних спеціальностей для придбання практичних навичок вирішення оптимізаційних задач.

ISBN 966-7809-24-2

ЗМІСТ

ВСТУП

Методи оптимізації призначені для знаходження екстремумів функцій і точок, у яких вони досягаються при наявності обмежень або без них.

Сфера застосування методів оптимізації дуже широка. Математичне моделювання, планування і прогнозування в економіці та техніці сполучені з необхідністю пошуку найкращого з можливих варіантів, поліпшення якості, надійності та достовірності моделей, скорочення їхньої розмірності.

Сама задача розрахунку параметрів чисельної математичної моделі в більшості випадків зводиться до вирішення деякої оптимізаційної задачі. Одним з таких випадків є штучні нейронні мережі (НМ), керувальні параметри яких при моделюванні на ЕОМ розраховують, вирішуючи оптимізаційну задачу мінімізації помилки моделі.

У цій книзі розглянуті методи чисельної оптимізації, що найбільш широко застосовуються на практиці, детально описані приклади застосування методів оптимізації при вирішенні задач навчання штучних НМ, а також вирішення задач багатовимірної нелінійної апроксимації.

Для полегшення пошуку визначень термінів і понять у тексті книги вони виділені **жирним шрифтом**. У додатку приведений тематичний путівник за списком літератури.

Автори висловлюють щиру подяку своїм колегам, що приймали участь в окремих дослідженнях, результати яких приведені в цій книзі, рецензентам, а також тим, хто надавав дружню підтримку авторам при написанні книги.

РОЗДІЛ 1. ЛІНІЙНЕ ПРОГРАМУВАННЯ

1.1 Задачі лінійного програмування

Загальною задачею лінійного програмування (ЛП) називається задача, що полягає у визначені максимального (мінімального) значення функції

$$F = \sum_{j=1}^{n} c_j \cdot x_j, \tag{1.1}$$

при умовах:

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \le b_i \ , \quad i = 1, 2, ..., k; \tag{1.2}$$

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \ge b_i \ , \ i = k+1, k+2, ..., m; \tag{1.3}$$

$$x_j \ge 0 \, , \, j = 1, 2, ..., n;$$

$$i = 1, 2, ..., L, \ L \le n \, , \tag{1.4}$$

де a_{ij}, b_i, c_j - задані постійні величини, x_j – шукані змінні та $k \le m$.

Функція (1.1) називається **цільовою функцією** (або лінійної формою) задачі (1.1) - (1.4), а умови (1.2) - (1.4) - **обмеженнями** даної задачі.

Стандартною (або симетричною) задачею лінійного програмування називається задача, що полягає у визначенні максимального значення функції (1.1) при виконанні умов (1.2) та (1.4), де $k = m$ та $L = n$.

Основною (або канонічною) задачею лінійного програмування називається задача, що полягає у визначенні максимального значення функції (1.1) при виконанні умов (1.3) та (1.4), де $k = 0$ та $L = n$.

Сукупність чисел $x = (x_1, x_2, ..., x_n)$, що задовольняє обмеженням (1.2) - (1.4), називається **припустимим рішенням (або планом)**.

План $x^* = (x_1^*, x_2^*, ..., x_n^*)$, при якому цільова функція задачі (1.1) приймає максимальне (мінімальне) значення, називається **оптимальним.**

Значення цільової функції (1.1) при плані x будемо позначати через F(x). Отже, x^* - оптимальний план задачі, якщо для будь-якого x виконується нерівність $F(x) \le F(x^*)$ (відповідно $F(x) \ge F(x^*)$).

Зазначені вище три форми задачі ЛП (загальна, стандартна й основна) еквівалентні в тому змісті, що кожна з них за допомогою стандартних перетворень може бути переписана у формі іншої задачі. Це означає, що якщо є спосіб знаходження рішення однієї з зазначених задач, то тим самим може бути визначений оптимальний план кожної з трьох задач.

Щоб перейти від однієї форми запису задачі ЛП до іншої, потрібно в загальному випадку вміти, по-перше, **зводити задачу мінімізації функції до задачі максимізації**, по-друге, **переходити від обмежень-нерівностей до обмежень-рівностей** і навпаки, по-третє, **замінять змінні**, які не відповідають умові невід'ємності.

У тому випадку, коли потрібно знайти мінімум функції

$$F = c_1x_1+c_2x_2+...+c_nx_n,$$

можна перейти до знаходження максимуму функції

$$F_1 = -F = -c_1x_1 - c_2x_2 - ... -c_nx_n,$$

оскільки min F = max (-F).

Обмеження-нерівність вихідної задачі ЛП, що має вигляд "$\le$", можна перетворити в обмеження-рівність додаванням невід'ємної змінної, а обмеження-нерівність вигляду "$\ge$" - в обмеження-рівність - вирахуванням з його лівої частини додаткової невід'ємної змінної.

Таким чином, обмеження-нерівність

$$a_{i1}x_1 + a_{i2}x_2 +...+ a_{in}x_n \le b_i$$

перетворюється в обмеження-рівність

$$a_{i1}x_1 + a_{i2}x_2 +...+ a_{in}x_n + x_{n+i} = b_i \ (x_{n+i} \ge 0),$$

а обмеження-нерівність

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \geq b_i$$

в обмеження-рівність

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n - x_{n+i} = b_i \ (x_{n+i} \geq 0).$$

У той же час кожне рівняння системи обмежень

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n = b_i$$

можна записати у вигляді нерівностей:

$$\begin{cases} a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \leq b_i, \\ a_{i1}x_1 - a_{i2}x_2 - ... - a_{in}x_n \leq -b_i. \end{cases} \quad (1.5)$$

Число невід'ємних змінних, що вводяться при перетворенні обмежень-нерівностей в обмеження-рівності дорівнює числу нерівностей, що перетворюються.

Відзначимо, нарешті, що якщо змінна x_k не відповідає умові невід'ємності, то її варто замінити двома невід'ємними змінними u_k та v_k, поклавши $x_k = u_k - v_k$.

1.2 Властивості основної задачі лінійного програмування

Запишемо основну задачу лінійного програмування у векторній формі. Знайти максимум функції

$$F = cx \quad (1.6)$$

при умовах

$$x_1p_1 + x_2p_2 + ... + x_np_n = p_0; \quad (1.7)$$

$$x \geq 0, \quad (1.8)$$

де $c = (c_1, c_2, ..., c_n)$, $x = (x_1, x_2, ..., x_n)$, cx – скалярний добуток, $p_1, ..., p_n$ та p_0 - m-вимірні вектори-стовпці, складені з коефіцієнтів при невідомих і вільних членах системи рівнянь задачі:

$$p_0 = \begin{bmatrix} b_1 \\ b_2 \\ ... \\ b_n \end{bmatrix}, \; p_1 = \begin{bmatrix} a_{11} \\ a_{21} \\ ... \\ a_{m1} \end{bmatrix}, \; p_2 = \begin{bmatrix} a_{12} \\ a_{22} \\ ... \\ a_{m2} \end{bmatrix}, ..., \; p_n = \begin{bmatrix} a_{1n} \\ a_{2n} \\ ... \\ a_{mn} \end{bmatrix}.$$

План $x = (x_1, x_2, ..., x_n)$ називається **опорним планом** основної задачі лінійного програмування, якщо система векторів p_j, що входять у розкладання (1.7) з позитивними коефіцієнтами x_j, лінійно незалежна.

Так як вектори p_j є m-вимірними, то з визначення опорного плану випливає, що число його позитивних компонентів не може бути більше, ніж m.

Опорний план називається **невиродженим**, якщо він складається рівно з m позитивних компонентів, у протилежному випадку він називається **виродженим.**

Властивості основної задачі лінійного програмування (1.6) - (1.8) тісно пов'язані з властивостями опуклих множин.

Нехай $x_1, x_2, ..., x_n$ - довільні точки евклідового простору E_n. **Опуклою лінійною комбінацією** цих точок називається сума $1x_1 + 2x_2 + ... + nx_n$.

Множина називається **опуклою**, якщо разом з будь-якими двома точками вона містить і їхню довільну опуклу лінійну комбінацію.

Точка x опуклої лінійної множини називається **кутовою**, якщо вона не може бути представлена у вигляді опуклої лінійної комбінації яких-небудь двох інших довільних точок даної множини.

Множина планів основної задачі лінійного програмування є опуклою (якщо вона не порожня).

Непорожня множина планів основної задачі лінійного програмування називається **багатогранником рішень**, а всяка кутова точка багатогранника рішень **- вершиною**.

Якщо основна задача лінійного програмування має оптимальний план, то максимальне значення цільова функція задачі приймає в

одній з вершин багатогранника рішень. Якщо максимальне значення цільова функція приймає більш ніж в одній вершині, то вона приймає його у всякій точці, що є опуклою лінійною комбінацією цих вершин.

Якщо система векторів p_1, p_2, ..., p_k ($k \le n$) у розкладанні (1.7) лінійно незалежна і така, що $x_1p_1+x_2p_2+...+x_kp_k = p_0$, де всі $x_j \ge 0$, то точка $x = (x_1, x_2, ..., x_k, 0, ..., 0)$ є вершиною багатогранника рішень.

Якщо $x = (x_1, x_2, ..., x_n)$ - вершина багатогранника рішень, то вектори p_j, що відповідають позитивним x_j у розкладанні (1.7), лінійно незалежні.

1.3 Симплексний метод вирішення задачі лінійного програмування

Вирішення задачі лінійного програмування можна знайти **симплексним методом**. Перш ніж застосовувати цей метод, варто записати вихідну задачу у формі основної задачі ЛП, якщо вона не має такої форми запису.

Симплексний метод вирішення задачі ЛП заснований на переході від одного опорного плану до іншого, при якому значення цільової функції збільшується (при умові, що дана задача має оптимальний план і кожен її опорний план є невиродженим).

Зазначений перехід можливий, якщо відомий який-небудь вихідний опорний план. Розглянемо задачу, для якої цей план можна безпосередньо записати.

Нехай потрібно знайти максимальне значення функції

$$F = c_1x_1 + c_2x_2 + ... + c_nx_n$$

при умовах

$$\begin{cases} x_1 + a_{1m+1}x_{m+1} + ... + a_{1n}x_n = b_1, \\ x_2 + a_{2m+1}x_{m+1} + ... + a_{2n}x_n = b_2, \\ \text{..... ....} \\ x_m + a_{mm+1}x_{m+1} + ... + a_{mn}x_n = b_m, \end{cases}$$

$$x_j \ge 0, j = 1, ..., n.$$

Тут a_{ij}, b_i та c_j (i = 1, ..., m; j = 1, ..., n) - задані постійні числа (m < n та $b_i > 0$).

Векторна форма даної задачі має наступний вигляд. Знайти максимум функції

$$F = \sum_{j=1}^{n} c_j x_j \qquad (1.9)$$

при умовах $x_1p_1 + x_2p_2 + ... + x_mp_m + ... + x_np_n = p_0,$ (1.10)

$$x_j \geq 0, j = 1, ..., n, \qquad (1.11)$$

де $p_1 = \begin{bmatrix} 1 \\ 0 \\ ... \\ 0 \end{bmatrix}$; $p_2 = \begin{bmatrix} 0 \\ 1 \\ ... \\ 0 \end{bmatrix}$; ...; $p_m = \begin{bmatrix} 0 \\ 0 \\ ... \\ 1 \end{bmatrix}$; $p_{m+1} = \begin{bmatrix} a_{1m+1} \\ a_{2m+1} \\ ... \\ a_{mm+1} \end{bmatrix}$; ...; $p_n = \begin{bmatrix} a_{1n} \\ a_{2n} \\ ... \\ a_{mn} \end{bmatrix}$; $p_0 = \begin{bmatrix} b_1 \\ b_2 \\ ... \\ b_m \end{bmatrix}$.

Тому що $b_1p_1 + b_2p_2 + ... + b_mp_m = p_0$, то за визначенням опорного плану x = (b_1; b_2; ...; b_m; 0; ...; 0) є опорним планом даної задачі (останні n-m компонентів вектора x дорівнюють нулю). Цей план визначається системою одиничних векторів p_1, p_2, ..., p_m, які утворюють базис m-вимірного простору. Тому кожний з векторів p_1, p_2, ..., p_m, ..., p_n, а також вектор p_0 можуть бути представлені у вигляді лінійної комбінації векторів даного базису.

Нехай $p_j = \sum_{i=1}^{m} x_{ij}p_i$, j = 0, 1, ..., n. Покладемо:

$$z_j = \sum_{i=1}^{m} c_i x_{ij} , j = 1, 2, ..., n,$$

$$\Delta j = z_j - c_j , j = 1, 2, ..., n.$$

Оскільки вектори $p_1, p_2, ..., p_m$ - одиничні, то $x_{ij} = a_{ij}$ та

$$z_j = \sum_{i=1}^{m} c_i a_{ij} ,$$

$$\Delta j = \sum_{i=1}^{m} c_i a_{ij} - c_j .$$

Опорний план $x' = (x_1', x_2', ..., x_m', 0, 0, ..., 0)$ задачі (1.9)-(1.11) є оптимальним, якщо $\Delta j \geq 0$ для будь-якого j, $j = 1, 2, ..., n$).

Якщо $\Delta k < 0$ для деякого $j = k$ та серед чисел a_{ik}, $i = 1, 2, ..., m$, немає позитивних ($a_{ik} \leq 0$),то цільова функція (1.9) задачі (1.9)-(1.11) не обмежена на множині її планів.

Якщо опорний план x задачі (1.9)-(1.11) невироджений та $\Delta k < 0$, але серед чисел a_{ik} є позитивні (не всі $a_{ik} \leq 0$), то існує опорний план x' такий, що $F(x') > F(x)$.

Перераховані властивості дозволяють перевірити, чи є знайдений опорний план оптимальним, і виявити доцільність переходу до нового опорного плану.

Дослідження опорного плану на оптимальність, а також подальший обчислювальний процес зручніше вести, якщо умови задачі і початкові дані, отримані після визначення вихідного опорного плану, записати, як показано в табл. 1.1.

У стовпці $c_б$ цієї таблиці записують коефіцієнти при невідомих цільової функції, що мають ті ж індекси, що і вектори даного базису.

У стовпці p_0 записують позитивні компоненти вихідного опорного плану, у ньому ж результати обчислень одержують позитивні компоненти опорного плану. Стовпці векторів p_j представляють собою коефіцієнти розкладання цих векторів по векторам даного базису.

У табл. 1.1 перші m рядків визначаються вихідними даними задачі, а показники $(m+1)$-го рядка обчислюють.

У цьому рядку в стовпці вектора p_0 записують значення цільової функції, яке вона приймає при даному опорному плані, а в стовпці вектора p_j - значення $\Delta j = z_j - c_j$.

Таблиця 1.1 - Вихідний опорний план

I	Базис	$c_б$	p_0	c_1	c_2	...	c_r	...	c_m	c_{m+1}	...	c_k	...	c_n
				p_1	p_2	...	p_r	...	p_m	p_{m+1}	...	p_k	...	p_n
1	p_1	c_1	b_1	1	0	...	0	...	0	$a_{1(m+1)}$	...	a_{1k}	...	a_{1n}
2	p_2	c_2	b_2	0	1	...	0	...	0	$a_{2(m+1)}$	...	a_{2k}	...	a_{2n}
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
R	p_r	c_r	b_r	0	1	...	0	...	0	$a_{r(m+1)}$	...	a_{rk}	...	a_{rn}
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
M	p_m	c_m	b_m	0	0	...	1	...	1	$a_{m(m+1)}$	...	a_{mn}	...	a_{mn}
M+1			F_0	0	0	...	0	...	0	$\Delta m+1$	...	Δk	...	Δn

Значення z_j знаходиться як скалярний добуток вектора p_j, $j = 1, 2, ..., n$, та вектора $c_б = (c_1, c_2, ..., c_m)$:

$$z_j = \sum_{i=1}^{m} c_i a_{ij} \quad , j = 1, 2, ..., n.$$

Значення F_0 дорівнює скалярному добутку вектора p_0 та вектора $c_б$:

$$F_0 = \sum_{i=1}^{m} c_i b_i \quad .$$

Після заповнення табл. 1.1 вихідний опорний план перевіряють на оптимальність. Для цього продивляються елементи (m+1)-го рядка таблиці. В результаті може мати місце один з наступних трьох випадків:

1) $\Delta j \geq 0$ для $j = m+1, m+2, ..., n$ (при $j = 1, ..., m$, $z_j = c_j$ та, відповідно, $\Delta j = 0$). Тому в даному випадку числа $\Delta j \geq 0$ для всіх j від 1 до n;

2) $\Delta j < 0$ для деякого j та усі відповідні цьому індексу величини $a_{ij} \leq 0$ $(i = 1, 2, ..., m)$;

3) $\Delta j < 0$ для деяких індексів j та для кожного такого j у крайньому разі одне з чисел a_{ij} позитивне.

У першому випадку на підставі ознаки оптимальності вихідний опорний план є оптимальним. В другому випадку цільова функція не обмежена зверху на множині планів, а в третьому випадку можна перейти від вихідного плану до нового опорного плану, при якому значення цільової функції збільшиться. Цей перехід від одного опорного плану до іншого здійснюється виключенням з вихідного базису якого-небудь з векторів і введенням у нього нового вектора. У якості вектора, що вводиться в базис, можна взяти кожен з векторів p_j, що має індекс j, для якого $\Delta j<0$. Нехай, наприклад, $\Delta k<0$ і вирішено ввести в базис вектор p_k. Для визначення вектора, що підлягає виключенню з базису, знаходять $\min (b_i/a_{ik})$ для всіх $a_{ik}>0$. Нехай цей мінімум досягається при $i = r$. Тоді з базису виключають вектор p_r, а число a_{rk} називають **розрішаючим елементом**. Стовпець і рядок, на перетині яких знаходиться розрішаючий елемент, називають **направляючим**.

Після виділення направляючого рядка і направляючого стовпця знаходять новий опорний план x і коефіцієнти розкладання векторів p_j через вектори нового базису, що відповідають новому опорному плану. Позитивні компоненти нового опорного плану обчислюють за формулою:

$$b_i' = \begin{cases} b_i - \dfrac{b_r}{a_{rk}} a_{ik}, \text{ при } i \neq r, \\ \dfrac{b_r}{a_{rk}}, \text{ при } i = r, \end{cases} \qquad (1.12)$$

а коефіцієнти розкладання векторів p_j через вектори нового базису, що відповідають новому опорному плану, за формулою:

$$a_{ij}' = \begin{cases} a_{ij} - \dfrac{a_{rj}}{a_{rk}} a_{ik}, \text{ при } i \neq r, \\ \dfrac{a_{rj}}{a_{rk}}, \text{ при } i = r. \end{cases} \qquad (1.13)$$

Після обчислення b_i' та a_{ij}' згідно (1.12) та (1.13) їхні значення заносять у табл. 1.2.

Елементи (m+1)-го рядка цієї таблиці можуть бути обчислені або за формулами:

$$F_0' = F_0 - \Delta k(b_r/a_{rk}), \tag{1.14}$$

$$\Delta j' = \Delta j - \Delta k(a_{rj}/a_{rk}), \tag{1.15}$$

або на підставі їх визначення.

Таблиця 1.2 – Опорний план

i	Базис	$c_б$	p_0	c_1	c_2	...	c_r	...	c_m	c_{m+1}	...	c_k	...	c_n
				p_1	p_2	...	p_r	...	p_m	p_{m+1}	...	p_k	...	p_n
1	p_1	c_1	b'_1	1	0	...	a'_{1r}	...	0	$a'_{1(m+1)}$	...	0	...	a'_{1n}
2	p_2	c_2	b'_2	0	1	...	a'_{2r}	...	0	$a'_{2(m+1)}$	...	0	...	a'_{2n}
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
r	p_k	c_k	b'_r	0	0	...	a'_{rr}	...	0	$a'_{r(m+1)}$	...	1	...	a'_{rn}
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
m	p_m	c_m	b'_m	0	0	...	a'_{mr}	...	1	$a'_{m(m+1)}$	...	0	...	a'_{mn}
m+1			F'_0	0	0	...	z'_r-c_r	...	0	z'_{m+1}-c_m	...	0	...	z'_n-c_n

Наявність двох способів знаходження елементів (m+1)-го рядка дозволяє здійснювати контроль правильності обчислень, що проводяться.

З (1.14) випливає, що при переході від одного опорного плану до іншого найбільш доцільно ввести в базис вектор p_j, що має індекс j, при якому максимальним за абсолютною величиною є число $\Delta j(b_r/a_{rj})$, $\Delta j>0$, $a_{rj}>0$. Однак з метою спрощення обчислювального процесу надалі будемо вектор, що вводиться в базис, визначати, виходячи з максимальної абсолютної величини від'ємних чисел Δj. Якщо ж таких чисел декілька, то в базис вводиться вектор, що має такий же індекс, як і максимальне з чисел c_j, що визначаються даними числами Δj ($\Delta j<0$).

Отже, перехід від одного опорного плану до іншого зводиться до переходу від однієї симплекс-таблиці до іншої. Елементи нової

симплекс-таблиці можна обчислити як за допомогою формул (1.12)-(1.15), так і за правилами, що безпосередньо витікають з них. Ці правила полягають у наступному.

У стовпцях векторів, що входять у базис, на перетині рядків і стовпців однойменних векторів проставляють одиниці, а всі інші елементи даних стовпців встановлюють рівними нулю.

Елементи векторів p_0 та p_j у рядку нової симплекс-таблиці, у якій записаний вектор, що вводиться в базис, одержують з елементів цього ж рядка вихідної таблиці розподілом їх на величину розрішаючого елемента. У стовпці $c_б$ у рядку вектора, що вводиться проставляють величину c_k, де k - індекс вектора, що вводиться .

Інші елементи стовпців векторів p_0 та p_j нової симплекс-таблиці, обчислюють за правилом трикутника. Для обчислення кожного з цих елементів знаходять три числа.

1) Число, що знаходиться у вихідній симплекс-таблиці на місці шуканого елемента нової симплекс-таблиці.

2) Число, що знаходиться у вихідній симплекс-таблиці на перетині рядка, у якому знаходиться шуканий елемент нової симплекс-таблиці, і стовпця, відповідного вектору, що вводиться в базис.

3) Число, що знаходиться в новій симплекс-таблиці на перетині стовпця, у якому знаходиться шуканий елемент, і рядки вводяться знову в базис вектора. Як відзначено раніше, цей рядок виходить з рядка вихідної симплекс-таблиці діленням її елемента на вирішуючий елемент.

Ці три числа утворюють своєрідний трикутник, дві вершини якого відповідають числам, що знаходяться у вихідній симплекс-таблиці, а третя - числу, що знаходиться в новій симплекс-таблиці. Для визначення шуканого елемента нової симплекс-таблиці з першого числа віднімають добуток другого і третього.

Після заповнення нової симплекс-таблиці продивляються елементи (m+1)-го рядка. Якщо всі $z'_j-c_j \geq 0$, то новий опорний план є оптимальним. Якщо ж серед зазначених чисел є від'ємні, то, використовуючи описану вище послідовність дій, знаходять новий опорний план. Цей процес продовжується доти, доки або не одержують оптимальний план задачі, або не встановлюють її невирішуваність.

При знаходженні вирішення задачі ЛП припускалося, що ця задача має опорні плани і кожний такий план є невиродженим.

Якщо ж задача має вироджені опорні плани, то на одній з ітерацій одна або декілька змінних опорного плану можуть виявитися рівними нулю. Таким чином, при переході від одного опорного плану до іншого значення функції може залишитися попереднім.

Більш того, можливий випадок, коли функція зберігає своє значення протягом декількох ітерацій, а також можливе повернення до початкового базису. У цьому випадку говорять, що виникло **зациклення**. Однак на практиці цей випадок зустрічається дуже рідко, тому на ньому зупинятися не будемо.

РОЗДІЛ 2. НЕЛІНІЙНЕ ПРОГРАМУВАННЯ

2.1 Загальна задача нелінійного програмування

Формально **загальна задача нелінійного програмування** (нелінійної оптимізації) може бути сформульована таким чином.

Мінімізувати **цільову функцію** $f(w) = f(w_1,w_2,...,w_N)$, де $w - \{w_1,w_2,...,w_N\}$ – набір змінних параметрів, від яких залежить значення цільової функції при p лінійних та / або нелінійних обмеженнях:

у вигляді рівностей:
$$\begin{cases} c_1(w_1, \dots, w_N) = 0, \\ \dots\dots\dots\dots\dots\dots \\ c_m(w_1, \dots, w_N) = 0; \end{cases}$$

та у вигляді нерівностей:
$$\begin{cases} c_{m+1}(w_1, \dots, w_N) \geq 0, \\ \dots\dots\dots\dots\dots\dots \\ c_p(w_1, \dots, w_N) \geq 0. \end{cases}$$

У **векторній формі загальна задача нелінійного програмування** запишеться таким чином:

$$f(w) \rightarrow \min;$$

$$c_j(w) = 0;\ j = 1, ..., m;$$

$$c_k(w) \geq 0;\ k = m+1, ..., p.$$

У більшості практичних випадків від обмежень типу рівність можна позбутися без порушення загального формулювання задачі.

Будемо розглядати вектор варійованих змінних w як множину. Нехай множина змінних w розбита на дві підмножини:

$$w = u \cup v,$$

де $u = \{w_1, \dots, w_m\} = \{u_1, \dots, u_m\}$, $v = \{w_{m+1}, \dots, w_p\} = \{v_1, \dots, v_p\}$.

Тоді систему обмежень у вигляді рівностей можна представити таким чином:

$$\begin{cases} c_1(u, v) = 0, \\ \dots\dots\dots\dots \\ c_m(u, v) = 0. \end{cases}$$

При цьому система обмежень у вигляді рівностей є замкнутою щодо змінної u.

Нехай A - алгоритм, що вирішує систему обмежень у вигляді рівностей щодо змінної u (при заданих величинах v): $u = A(v)$.

Тепер функції в рівняннях залежать тільки від v:

$$f(w) = f(u,v) = f(A(v),v) = f'(v),$$

$$c_j(w) = c_j(u,v) = c_j(A(v),v) = c_j'(v);\ j = 1, \dots, m;$$

$$c_k(w) = c_k(u,v) = c_k(A(v),v) = c_k'(v) \geq 0;\ k = m+1, \dots, p.$$

а загальна задача нелінійного програмування перетворюється в **задачу нелінійного програмування без обмежень типу рівність**:

$$f'(v) \to \min,$$

$$c_k'(v) \geq 0,\ k = m+1, \dots, p;$$

$$u = A(v),\ w = u \cup v.$$

Частковим випадком загальної задачі нелінійної оптимізації є **задача класичної оптимізації**, що представляє собою загальну задачу нелінійного програмування, у якій система обмежень відсутня:

$$f(w) \rightarrow \min.$$

2.2 Класифікація методів нелінійного програмування

Задача нелінійної оптимізації представляє собою задачу пошуку екстремуму деякої функції. **Екстремум** – це крайнє значення – мінімуму або максимуму функції. Розрізняють екстремуми: **локальний** – екстремум у деякій довільно малій околиці даної точки, і **глобальний** – екстремум у всій розглянутій області значень змінних.

Якщо глобальний екстремум функції шукають на всьому просторі незалежних змінних, то така задача оптимізації називається **задачею на безумовний екстремум**, а методи її вирішення – **методами безумовної оптимізації**. Якщо ж екстремум шукають при деяких обмеженнях на незалежні змінні, то така задача оптимізації називається **задачею на умовний екстремум**, а методи її вирішення – **методами оптимізації з обмеженнями.**

У залежності від кількості екстремумів на інтервалі пошуку цільова функція може бути **унімодальною** (один екстремум) або **полімодальною** (декілька екстремумів**).**

На практиці більшість функцій – полімодальні, тобто містять декілька екстремумів. Пошук глобального екстремуму полімодальної функції вкрай складна задача, при вирішенні якої виникає **проблема яровості**: збіжність методу оптимізації погіршується при наближенні до точного рішення оптимізаційної задачі через те, що він блукає по локальних мінімумах.

Переважна більшість методів чисельного рішення задач нелінійного програмування є **ітераційними**. У цих методах спочатку задається w_0 – початкове наближення до точки екстремуму, після чого виконуються ітерації за правилом: $w_{k+1} = A(w_k)$, де A – алгоритм, що відповідає методу, що застосовується, w_k та w_{k+1} – значення наборів керованих змінних на k-ій та (k+1)-ій ітераціях, відповідно, k = 1, ..., K, де K – максимально припустима кількість ітерацій (задається користувачем).

Проекційними (неітеративними) методами оптимізації називають методи, що дозволяють оптимізувати функцію за один прохід без ітеративного горегування значень керованих змінних.

Задачі класичної оптимізації по кількості змінних у цільовій функції поділяються на **одновимірні** (одна змінна) і **багатовимірні** (декілька змінних).

Усі методи рішення задач нелінійного програмування за способом використання градієнта цільової функції поділяються на **градієнтні**, що вимагають обчислення похідних, і **безградієнтні**, що замість похідних використовують значення цільової функції в групі з декількох точок.

Для програмної реалізації градієнтних методів необхідно вміти розраховувати значення похідної функції, заданої таблично, у деякій точці.

Для оцінки значення похідної використовують **метод кінцевих різниць**: нехай h – крок диференціювання, тоді похідна n-го порядку визначається за формулою:

$$y^{(n)} = \frac{\Delta^n y}{h^n},$$

де $\Delta^n y$ - кінцева різниця n-го порядку.

Центральна кінцева різниця першого порядку визначається за формулою:

$$\Delta^1 y = \frac{y^{i+1} - y^{i-1}}{2},$$

де y^s – значення функції y в s-ій точці.

Для визначення **кінцевої різниці n-го порядку** використовують ітеративну формулу:

$$\Delta^n y = \Delta(\Delta^{n-1} y).$$

У залежності від рівня математичного обґрунтування чисельні методи нелінійної оптимізації поділяють на **математичні**, що мають строге математичне обґрунтування, та **евристичні,** засновані переважно на досвіді та розуміннях розробника (ці методи не мають строгого математичного обґрунтування).

Чисельні методи оптимізації характеризуються наступними критеріями:

- **швидкість збіжності** методу – визначає наскільки швидко зменшується (збільшується) значення цільової функції за одиницю часу;
- **кількість ітерацій** методу для знаходження екстремуму з заданою точністю;
- **точність** методу, що досягається, при заданій кількості ітерацій;
- **трудомісткість програмної реалізації**;
- **вимоги методу до ресурсів ЕОМ** при програмній реалізації (необхідний обсяг пам'яті й обсяг обчислень).

РОЗДІЛ 3. ОДНОВИМІРНІ МЕТОДИ ОПТИМІЗАЦІЇ

Задачу одновимірної оптимізації можна сформулювати таким чином: знайти мінімум функції f(w) на відрізку w ∈ [a,b], де w – змінна від якої залежить функція f, а та b – деякі задані константи, такі, що: a<b.

Функція f(w) називається **унімодальною** на відрізку [a,b], якщо вона монотонна по обидві сторони від єдиної на розглянутій ділянці оптимальної точки w*∈ [a,b].

Функція f(w) називається **монотонною,** якщо для двох довільних точок w_1 та w_2, таких, що $w_1 \leq w_2$ виконується одна з нерівностей: $f(w_1) \leq f(w_2)$ або $f(w_1) \geq f(w_2)$.

Функція f(w) називається **безперервною** в деякій області, якщо вона безперервна в кожній точці цієї області, тобто якщо в кожній точці цієї області функція має певне значення F і межа рівна F; у протилежному випадку функція називається **розривною.**

3.1 Методи прямого пошуку

Застосування методів прямого пошуку (методів виключення інтервалів) накладає єдину вимогу на досліджувану функцію: вона повинна бути унімодальною. Отже, зазначені методи можна використовувати для аналізу як **безперервних**, так і **розривних** функцій, а також у випадках, коли змінні приймають значення з дискретної множини.

Логічна структура пошуку за допомогою методів виключення інтервалів заснована на простому порівнянні значень функції в двох пробних точках. Крім того, при такому порівнянні в розрахунок приймається тільки відношення порядку на множині значень функції і не враховується величина різниці між значеннями функції.

3.1.1 Метод розподілу інтервалу навпіл

Даний метод дозволяє виключити в точності половину інтервалу на кожній ітерації. Цей метод також називають **трьохточковим пошуком** на рівних інтервалах, оскільки його реалізація заснована

на виборі трьох пробних точок, рівномірно розташованих в інтервалі пошуку:

а) Покласти $w_m=(a+b)/2$; $L=b-a$. Обчислити $f(w_m)$.

б) Покласти $w_1=a+L/4$; $w_2=b-L/4$, точки w_1, w_2, w_m – поділяють інтервал (a,b) на 4 рівні частини. Обчислити $f(w_1)$ та $f(w_2)$.

в) Порівняти $f(w_1)$ та $f(w_m)$.

Якщо $f(w_1) < f(w_m)$ – виключити інтервал (w_m, b), покласти $b = w_m$. Середньою точкою нового інтервалу стає точка w_1, отже, необхідно покласти $w_m=w_1$. Перейти до кроку д).

Якщо $f(w_1) > f(w_m)$, перейти до кроку г).

г) Порівняти $f(w_2)$ та $f(w_m)$.

Якщо $f(w_2) < f(w_m)$, виключити інтервал (a,w_m), $a=w_m$, тому що середньою точкою нового інтервалу стає x_2, покласти $w_m=w_2$. Перейти до кроку д).

Якщо $f(w_2) \geq f(w_m)$, виключити інтервали (a,w_1), (w_2,b). Покласти $a=w_1$, $b=w_2$, w_m – не змінюється, тобто продовжує залишатися середньою точкою нового інтервалу. Перейти до кроку д).

д) Обчислити $L=b-a$, якщо величина $|L|$ є малою, закінчити пошук. У протилежному випадку повернутися до кроку а).

3.1.2 Метод Фібоначчі

Існує інтервал невизначеності (w_1,w_3) і відоме значення функції $f(w_2)$ усередині цього інтервалу, необхідно знайти таку точку w_4, щоб одержати найменший інтервал невизначеності.

Покладемо $w_2 - w_1 = L$ та $w_3 – w_2 = R$, причому $L > R$ і ці значення будуть фіксовані, якщо відомі w_1, w_2, w_3. Якщо w_4 знаходиться в інтервалі (w_1,w_2), то:

а) якщо $f(w_4) < f(w_2)$, то новим інтервалом невизначеності буде (w_1,w_2) довжиною $w_2 - w_1 = L$;

б) якщо $f(w_4) > f(w_2)$, то новим інтервалом невизначеності буде (w_4,w_3) довжиною $w_3 – w_4$.

Оскільки не відомо, яка з цих ситуацій буде мати місце, виберемо w_4, таким чином, щоб мінімізувати найбільшу з довжин $w_3 – w_4$ та $w_2 –w_1$. Це можливо, якщо довжини $w_3 – w_4$ та $w_2 –w_1$ рівні, тобто w_4 знаходиться усередині інтервалу симетрично щодо точки w_2, що вже

лежить усередині інтервалу. Будь-яке інше положення точки w_4 може привести до того, що отриманий інтервал буде більше L.

На n-му обчисленні n-ту точку варто помістити симетрично відносно (n-1)-ої точки. Положення цієї останньої точки в принципі залежить від нас. Для того, щоб одержати найбільше зменшення інтервалу на даному етапі, варто розділити навпіл попередній інтервал. Тоді точка w_n буде збігатися з точкою w_{n-1}. Звичайно точки w_n та w_{n-1} відстоять одна від одної на достатній відстані, щоб визначити, в якій половині лівій або правій знаходиться інтервал невизначеності. Вони містяться на відстані ε/2 по обидві сторони від середини відрізка L_{n-1} (ε – вибирається самостійно або вибирається рівній мінімально можливій відстані між двома точками).

Інтервал невизначеності буде мати довжину L_n, отже, $L_{n-1} = 2L_n - \varepsilon$.

На попередньому етапі точки w_{n-1} та w_{n-2} повинні бути розміщенні симетрично усередині інтервалу L_{n-2} на відстані L_{n-1} від кінців цього інтервалу. Отже, $L_{n-2} = L_{n-1} + L_n$

На передостанньому етапі w_{n-2} залишається як внутрішня точка.

Аналогічно: $L_{n-3} = L_{n-2} + L_{n-1}$

У загальному випадку: $L_{j-1} = L_j + L_{j+1}$ при $1 < j < n$.

Таким чином,

$$L_{n-1} = 2L_n - \varepsilon,$$

$$L_{n-2} = L_{n-1} + L_n = 3L_n - \varepsilon,$$

$$L_{n-3} = L_{n-2} + L_{n-1} = 5L_n - 2\varepsilon,$$

$$L_{n-4} = L_{n-3} + L_{n-2} = 8L_n - 3\varepsilon \text{ і т.д.}$$

Якщо визначити послідовність чисел Фібоначчі наступним чином:

$$F_0 = 1,\ F_1 = 1,\ F_k = F_{k-1} + F_{k-2} \text{ для } k = 2,3,\ldots,$$

тоді

$$L_{j-1} = F_{j+1}L_n - F_{j-1}\varepsilon\ ,\ \ j = 1,2,\ldots,n\text{-}1...$$

Якщо початковий інтервал (a,b) має довжину L= b - a, тоді

$$L_1 = F_n L_n - \varepsilon F_{n-2},$$

тобто $$L_n = L_1/F_n + \varepsilon F_{n-2}/F_n .$$

Отже, провівши n обчислень функції, початковий інтервал невизначеності зменшується в $1/F_n$ раз у порівнянні з його початковою довжиною (нехтуючи ε).

Якщо пошук почато, то його нескладно продовжити, використовуючи описане вище правило симетрії. Отже, необхідно знайти положення першої точки, що міститься на відстані L_n від одного з кінців початкового інтервалу, причому неважливо, від якого кінця, оскільки інша точка міститься відповідно до правила симетрії на відстані L_2 від іншого кінця інтервалу:

$$L_2 = F_{n-1}L_n - \varepsilon F_{n-3} = F_{n-1}\frac{L_1}{F_n} + \varepsilon\frac{(F_{n-1}F_{n-2} - F_n F_{n-3})}{F_n} = \frac{F_{n-1}}{F_n}L_1 + \frac{(-1)^n\varepsilon}{F_n}.$$

Після того, як знайдено положення першої точки, числа Фібоначчі більше не потрібні. Значення ε, що використовується, може визначатися з практичних розумінь. Воно повинно бути менше L_1/F_{n+1}, у протилежному випадку буде дарма витрачатися час на обчислення функції.

У процесі пошуку інтервалу (w_1,w_2) із точкою w_2, що вже лежить у цьому інтервалі, наступна точка w_4 завжди вибирається такою, що $w_3 - w_4 = w_2 - w_1$ або $w_4 - w_1 = w_3 - w_2$, тобто $w_4 = w_1 + w_3 - w_2$. Нехай $f(w_2) = f_2$ і $f(w_4) = f_4$, розглянемо чотири випадки, якщо:

а) $w_4 < w_2$, $f_4 < f_2$ – новий інтервал (w_1,w_2), містить точку w_4;

б) $w_4 > w_2$, $f_4 < f_2$ – новий інтервал (w_2,w_3), містить точку w_4;

в) $w_4 < w_2$, $f_4 > f_2$ – новий інтервал (w_4,w_3), містить точку w_2;

г) $w_4 > w_2$, $f_4 < f_2$ – новий інтервал (w_1,w_4), містить точку w_2.

3.1.3 Пошук методом “золотого перетину”

Стратегія даного методу така: пробні точки відстоять від граничних точок інтервалу на відстані τ. При такому симетричному

розташуванні точок довжина інтервалу, що залишається після виключення, завжди дорівнює τ, незалежно від того, яке зі значень функції в пробних точках виявляється меншим.

Припустимо, що виключається правий підінтервал. Підінтервал, що залишився довжиною τ містить одну пробну точку, розташовану на (1- τ) від лівої граничної точки. Щоб симетрія пошукового зразка зберігалася, відстань (1- τ) повинна складати τ – ту частину довжини інтервалу, що дорівнює τ:

$$(1 - \tau)/\tau = \tau.$$

При такому виборі τ випливає, що пробна точка розміщується на відстані рівній τ – ій частині довжини інтервалу від правої граничної точки інтервалу.

Звідси випливає, що при виборі τ відповідно до умови $(1 - \tau) = \tau^2$ симетричний пошук зразка, зберігається при переході до зменшеного інтервалу. Вирішуючи квадратне рівняння, одержуємо $\tau = (-1 \pm \sqrt{5})/2 \approx 0{,}61803$.

3.1.4 Метод рівномірного пошуку

Спочатку задається відрізок [a,b] та вибирається число рівних підінтервалів n, на які розбивається відрізок [a,b].

Потім знаходять довжину підінтервалу: h = (b-a)/n.

Обчислюють значення функції f(w) на множині точок $w_i = a + kh$, k = 0, ..., n, і знаходять її мінімум шляхом простого перебору значень функції $f(w_i)$ у точках w_i. Нехай точка мінімуму буде w_m.

Якщо умова унімодальності не виконана (тобто потрібно знайти глобальний мінімум), тоді w_m приймається як наближене рішення задачі. Точність результату, що одержана, залежить від вибору параметра n.

Якщо функція f(w) є унімодальною, тоді точне рішення:

$$w \in [w_{m-1}, w_{m+1}].$$

Якщо задана точність не досягнута, тобто $2h > \varepsilon$, де ε - задана константа, створюється новий інтервал $[a,b] = [w_{m-1}, w_{m+1}]$ та

повторюється все спочатку. Мінімальне значення n у цьому випадку дорівнює 3.

3.2 Методи пошуку з використанням поліноміальної апроксимації

Розглянемо методи пошуку, що дозволяють врахувати відносні зміни значень функції і, як наслідок, у ряді випадків виявляються більш ефективними, ніж методи виключення інтервалів. Однак виграш в ефективності досягається ціною введення додаткової вимоги, відповідно до якої функції, що досліджуються, повинні бути досить гладкими. Основна ідея розглянутих методів зв'язана з можливістю апроксимації гладкої функції поліномом і наступного використання апроксимуючого полінома для оцінювання координати точки оптимуму. Необхідними умовами ефективної реалізації такого підходу є унімодальність і безперервність досліджуваної функції. Відповідно до **теореми Вейєрштраса про апроксимацію**, якщо функція безперервна в деякому інтервалі, то її з будь-яким ступенем точності можна апроксимувати поліномом досить високого порядку. Отже, якщо функція унімодальна і знайдений поліном, що досить точно її апроксимує, то координату точки оптимуму функції можна оцінити шляхом обчислення координати точки оптимуму полінома. Відповідно до теореми Вейєрштраса, якість оцінок координати точки оптимуму, одержуваних за допомогою апроксимуючого полінома, можна підвищити двома способами: використанням полінома більш високого порядку і зменшенням інтервалу апроксимації. Другий спосіб, взагалі є більш кращим, оскільки побудова апроксимуючого полінома порядку вище третього стає дуже складною процедурою, тоді як зменшення інтервалу в умовах, коли виконується припущення про унімодальність функції, особливої складності не представляє.

3.2.1 Метод оцінювання з використанням квадратичної апроксимації

Найпростішим варіантом поліноміальної інтерполяції є **квадратична апроксимація**, заснована на тому факті, що функція,

яка приймає мінімальне значення у внутрішній точці інтервалу, повинна бути принаймні квадратичною. Якщо ж функція лінійна, то її оптимальне значення може досягатися тільки в одній із двох граничних точок інтервалу.

Таким чином, при реалізації методу оцінювання з використанням квадратичної апроксимації передбачається, що в обмеженому інтервалі можна апроксимувати функцію квадратичним поліномом, а потім використовувати побудовану апроксимаційну схему для оцінювання координати точки істини мінімуму функції.

Якщо задана послідовність точок w_1, w_2, w_3 і відомі відповідні цим точкам значення функції f_1, f_2 та f_3, то можна визначити постійні величини a_0, a_1 та a_2 таким чином, що значення квадратичної функції $q(w) = a_0+a_1(w - w_1)+a_2(w - w_1)(w - w_2)$ збігаються зі значеннями $f(w)$ у трьох зазначених точках. Перейдемо до обчислення $q(w)$ у кожній із трьох заданих точок.

Насамперед, тому що $f1 = f(w_1) = q(w_1) = a_0$, маємо $a_0 = f_1$.

Далі, оскільки $f_2 = f(w_2) = q(w_2) = f_1 + a_1(w_2 - w_1)$, одержуємо $a_1 = (f_2 - f_1)/(w_2 - w_1)$.

Нарешті, при $x = w_3$:

$f_3 = f(w_3) = q(w_3) = f_1+(f_2-f_1)/(w_2-w_1)(w_3-w_1)+a_2(w_3-w_1)(w_3-w_2)$.

Вирішуючи останнє рівняння відносно a_2, одержуємо:

$$a_2 = \frac{1}{w_3 - w_2}\left(\frac{f_3 - f_1}{w_3 - w_1} - \frac{f_2 - f_1}{w_2 - w_1}\right).$$

Якщо точність апроксимації досліджуваної функції в інтервалі від w_1 до w_3 за допомогою квадратичного полінома виявляється досить високою, то відповідно до запропонованої стратегії пошуку побудований поліном можна використовувати для оцінювання координати точки оптимуму.

Нагадаємо, що стаціонарні точки функції одної змінної визначаються шляхом прирівнювання до нуля її першої похідної і наступного знаходження коренів, отриманого таким чином рівняння.

У даному випадку з рівняння

$$\frac{dq}{dw} = a_1 + a_2 + (w - w_2) + a_2(w - w_1) = 0$$

можна одержати: $\overline{w} = \frac{w_2 + w_1}{2} - \frac{a_1}{2a_2}$.

Оскільки функція f(w) на розглянутому інтервалі має властивість унімодальності, а апроксимуючий квадратичний поліном також є унімодальною функцією, то можна очікувати, що величина $\overline{w}$ виявиться прийнятною оцінкою координати точки істиного оптимуму w^*.

3.2.2 Метод послідовного оцінювання з використанням квадратичної апроксимації (метод Пауелла)

Метод Пауелла заснований на послідовному застосуванні процедури оцінювання з використанням квадратичної апроксимації. Схему алгоритму можна описати таким чином. Нехай w_1 - початкова точка, Δw - обрана величина кроку по осі w.

Крок 1. Обчислити $w_2 = w_1 + \Delta w$.

Крок 2. Обчислити $f(w_1)$ та $f(w_2)$.

Крок 3. Якщо $f(w_1) > f(w_2)$,

тоді покласти: $w_3 = w_1 + 2\Delta w$,

інакше покласти: $w_3 = w_1 - \Delta w$.

Крок 4. Обчислити $f(w_3)$ і знайти $F_{min} = \min \{f_1, f_2, f_3\}$, w_{min} = точка w_i, що відповідає F_{min}.

Крок 5. По трьох точках w_1, w_2, w_3 обчислити $\bar{x}$, використовуючи формулу для оцінювання за допомогою квадратичної апроксимації.

Крок 6. Перевірка на закінчення пошуку.

(а) Чи є різниця $F_{min} - f(\overline{w})$ достатньо малою?

(б) Чи є різниця $w_{min} - \overline{w}$ достатньо малою?

Якщо обидві умови виконуються, закінчити пошук.

У протилежному випадку перейти до кроку 7.

Крок 7. Вибрати «найкращу» точку (w_{min} або $\overline{w}$) і дві точки по обидві сторони від неї. Позначити ці точки в природному порядку і перейти до кроку 4.

Зазначимо, що при першій реалізації кроку 5 межі інтервалу, що містить точку мінімуму, не обов'язково виявляються встановленими. При цьому отримана точка $\overline{w}$ може знаходитися за точкою w_3. Для

того, щоб виключити можливість занадто великого екстраполяційного переміщення, варто провести після кроку 5 додаткову перевірку й у випадку, коли точка x знаходиться занадто далеко від w_3, замінити $\overline{w}$ точкою, координата якої обчислюється з розрахунком заздалегідь установленої довжини кроку.

3.3 Методи пошуку з використанням похідних

Методи прямого пошуку і точкового оцінювання ґрунтуються на припущеннях про унімодальність й у ряді випадків про безперервність досліджуваної цільової функції. Доцільно припустити, що якщо на додаток до умови безперервності ввести вимогу диференціювання функції, то ефективність пошукових процедур можна істотно підвищити.

Необхідною умовою існування локального мінімуму функції в деякій точці z є обертання в нуль першої похідної функції в точці z, тобто $f'(z) = df/dw|_{w=z} = 0$.

Якщо функція f(w) містить члени, що включають w у третьому і більш високих ступенях, то безпосереднє одержання аналітичного рішення рівняння f'(w) = 0 може виявитися складним. У таких випадках використовуються наближені методи послідовного пошуку стаціонарної точки функції f.

3.3.1 Метод Ньютона-Рафсона

Для функцій однієї змінної класичний підхід при пошуку значень w у точках перегину функції f(w) полягає у вирішенні рівняння f'(w) = 0.

Вирішити таке рівняння не завжди просто. Тому коротко розглянемо числовий метод його рішення. Якщо можна знайти два значення a та b, таких, що f'(a) та f'(b) мають протилежні знаки, тоді, у силу очевидних припущень про безперервність, буде існувати корінь цього рівняння n, причому $a < n < b$.

Метод Ньютона дозволяє поліпшити відносно грубу апроксимацію, щоб одержати корінь рівняння f'(w) = 0.

У загальному випадку поліпшене значення для

$$w_{r+1} = w_r - \frac{f'(w_r)}{f''(w_r)}, \; r = 0, 1, ...$$

Ітерації можуть бути продовжені доти, доки для двох наступних апроксимацій не буде досягнута необхідна точність.

3.3.2 Метод середньої точки (метод Больцано)

Якщо функція f(w) унімодальна в заданому інтервалі пошуку, то точкою оптимуму є точка, у якій f'(w) = 0. Якщо при цьому є можливість обчислювати як значення функції, так і її похідної, то для знаходження кореня рівняння f'(w) = 0 можна скористатися ефективним алгоритмом виключення інтервалів, на кожній ітерації якого розглядається лише одна пробна точка. Наприклад, якщо в точці z виконується нерівність f'(z) < 0, то з урахуванням припущення про унімодальність природно стверджувати, що точка мінімуму не може знаходитися лівіше точки z. Іншими словами, інтервал $w \le z$ підлягає виключенню. З іншого боку, якщо f'(z)>0, то точка мінімуму не може знаходитися правіше z та інтервал $w \le z$ можна виключити.

Визначимо дві точки L та R таким чином, що f'(L)<0 та f'(R)>0. Стаціонарна точка розташована між L та R. Обчислимо значення похідної функції в середній точці розглянутого інтервалу z = (L+R)/2. Якщо f'(z)>0, то інтервал (z, R) можна виключити з інтервалу пошуку. З іншого боку, якщо f'(z)<0, то можна виключити інтервал (L, z). Нижче дається формалізований опис кроків алгоритму.

Нехай є обмежений інтервал $a \le w \le b$ та заданий параметр збіжності ε.

Крок 1. Покласти R = b, L = a; при цьому f'(a)<0 та f'(b)>0.

Крок 2. Обчислити z = (R+L)/2 та f'(z).

Крок 3. Якщо $|f'(z)|<\varepsilon$, закінчити пошук. У протилежному випадку, якщо f'(z)<0, покласти L = z та перейти до кроку 2. Якщо

f'(z)>0, покласти R=z та перейти до кроку 2.

Слід зазначити, що логічна структура пошуку відповідно до викладеного методу виключення інтервалів заснована лише на дослідженні знака похідної незалежно від значень, які ця похідна приймає.

3.3.3 Метод січних (метод хорд)

Метод січних, що є комбінацією методу Ньютона і загальної схеми виключення інтервалів, орієнтований на знаходження кореня рівняння f'(w) = 0 в інтервалі (a, b), якщо, зрозуміло, такий корінь існує.

Припустимо, що в процесі пошуку стаціонарної точки функції f'(x) в інтервалі (a, b) виявлені дві точки L та R, у яких знаки похідної різні. У цьому випадку алгоритм методу січних дозволяє апроксимувати функцію f'(w) «січною прямою» (прямою лінією, що з'єднує дві точки) і знайти точку, у якій січна графіка f'(w) перетинає вісь абсцис. Таким чином випливає, що наближення до стаціонарної точки x^* визначається за формулою:

$$z = R - \frac{f'(R)(R - L)}{f'(R) - f'(L)} .$$

Якщо $|f'(z)| \le \varepsilon$, пошук варто закінчити. У протилежному випадку необхідно вибрати одну з точок L або R таким чином, щоб знаки похідної в цій точці і точці 2 були різні, а потім повторити основний крок алгоритму.

Легко бачити, що на відміну від методу середньої точки метод січних заснований на дослідженні не тільки знака, але і значень похідної в пробних точках і тому в ряді випадків дозволяє виключити більше половини інтервалу пошуку.

3.3.4 Метод пошуку з використанням кубічної апроксимації

Відповідно до методу, що розглядається, функція f, що підлягає мінімізації, апроксимується поліномом третього порядку. Логічна схема методу аналогічна схемі методів з використанням

квадратичної апроксимації. Однак у даному випадку побудова апроксимуючого полінома проводиться на основі меншого числа точок, оскільки в кожній точці можна обчислювати значення як функції, так і її похідної.

Робота алгоритму починається в довільно обраній точці w_1: знаходиться інша точка w_2, така, що похідні $f'(w_1)$ та $f'(w_2)$ мають різні знаки. Іншими словами, необхідно покласти стаціонарну точку w^*, у якій $f'(w) = 0$, в інтервал між w_1 та w_2.

Апроксимуюча кубічна функція записується в наступному вигляді:

$$f^*(w_1) = a_0+a_1(w - w_1)+a_2(w - w_1)(w – w_2) + a_3(w – w_1)^2(w – w_2).$$

Параметри цього рівняння підбираються таким чином, щоб значення $f^*(w)$ і її похідної в точках w_1 та w_2 збігалися зі значеннями $f(w)$ і $f'(w)$ у цих точках.

Приведемо формалізований опис алгоритму. Нехай задані початкова точка w_0, позитивна величина кроку Δ та параметри збіжності ε_1 та ε_2 .

Крок 1. Обчислити $f'(w_0)$.

Якщо $f'(w_0)<0$, обчислити $w_{k+1}= w_k+2^k\Delta$ для значень $k = 0, 1, ...$

Якщо $f'(w_0)>0$, обчислити $x_{k+1}= w_k - 2^k\Delta$ для значень $k = 0, 1, ...$

Крок 2. Обчислити значення $f'(w)$ у точках w_{k+1} при $k= 0, 1, 2, ...$ аж до точки w, у якій $f'(w_{M-1})/f'(w)<0$. Потім покласти $w_1 = w_{M-1}$, $w_2 = w$. Обчислити значення $f(w_1)$, $f(w_2)$, $f'(w_1)$ та $f'(w_2)$.

Крок 3. Знайти стаціонарну точку w^* апроксимуючого кубічного полінома:

$$w^* = \begin{cases} w_2, \text{якщо } \mu < 0, \\ w_2 - \mu(w_2 - w_1), \text{якщо } 0 \le \mu \le 1, \\ w_1, \text{якщо } \mu > 1, \end{cases}$$

де

$$\mu = \frac{f'(w_2)+\beta - z}{f'(w_2)-f'(w_1)+2\beta},$$

$$z = \frac{3(f'(w_1)-f'(w_2))}{w_2 - w_1}+f'(w_1)+f'(w_2),$$

$$\beta = \begin{cases} \left(z^2 - f'(w_1)f'(w_2)\right)^{\frac{1}{2}}, \text{ якщо } w_1 < w_2, \\ -\left(z^2 - f'(w_1)f'(w_2)\right)^{\frac{1}{2}}, \text{ якщо } w_1 > w_2. \end{cases}$$

Формула для β забезпечує належний вибір одного з двох коренів квадратного рівняння; для значень μ, покладених в інтервалі від 0 до 1, формула гарантує, що одержувана точка w розташована між w_1 та w_2.

Крок 4. Якщо $f(w^*)<f(w_1)$, перейти до кроку 5. У протилежному випадку обчислювати w^* по формулі $w^* = w^* + 0{,}5(w^* - w_1)$ доти, доки не буде виконуватися нерівність $f(w^*) \le f(w_1)$.

Крок 5. Перевірка на закінчення пошуку.

Якщо $|f'(w^*)| \le \varepsilon_1$ та $|f'(w^*)| \le \varepsilon_2$, пошук закінчити.

У протилежному випадку покласти або

(а) $w_2=w_1$ і $w_1=w^*$, якщо $f'(w^*)f'(w_1)<0$,

або

(б) $w_1=w^*$, якщо $f'(w^*)f'(w_2)<0$.

Потім перейти до кроку 3.

Зазначимо, що кроки 1 та 2 реалізують процедуру пошуку меж інтервалу евристичним методом, причому зміна знака похідної використовується як критерій переходу через точку оптимуму. На кроці 3 проводяться обчислення координати точки оптимуму апроксимуючого полінома. Крок 4 асоційований з перевіркою того факту, що отримана оцінка дійсно є поліпшеним наближенням до точки оптимуму.

РОЗДІЛ 4. БАГАТОВИМІРНІ БЕЗГРАДІЄНТНІ МЕТОДИ НЕЛІНІЙНОЇ ОПТИМІЗАЦІЇ

4.1 Метод Нелдера - Міда

Метод Нелдера-Міда **(пошук по симплексу, пошук по деформованому багатограннику, S^2-метод)** є розвитком **симплексного методу Спендлі, Хекста і Хімсворта**. Множина (n+1)-ої рівновіддаленої точки в n-мірному просторі називається **регулярним симплексом**. Ця конфігурація розглядається в методі Спендлі, Хекста і Хімсворта. Отже, у двовимірному просторі симплексом є рівносторонній трикутник, а в тривимірному просторі — правильний тетраедр.

Ідея методу полягає в порівнянні значень функції в (n+1) вершинах симплексу і переміщенні симплексу в напрямку оптимальної точки за допомогою ітераційної процедури. У **симплексному методі**, запропонованому спочатку, регулярний симплекс використовувався на кожному етапі. Нелдер і Мід запропонували кілька модифікацій цього методу, що допускають, щоб симплекси були неправильними. У результаті вийшов дуже надійний метод прямого пошуку, що є одним з найефективніших, якщо $n \le 6$.

У методі Спендлі, Хекста і Хімсворта симплекс переміщується за допомогою трьох основних операцій: **відображення**, **розтягання** і **стиску**. Зміст цих операцій стане зрозумілим при розгляді кроків процедури Нелдера-Міда.

Крок 1. Знайдемо значення функції $f_1 = f(w_1)$, $f_2 = f(w_2)$, ..., $f_{n+1} = f(w_{n+1})$ у вершинах симплексу.

Крок 2. Знайдемо найбільше значення функції f_h, що випливає з найбільшого значення функції f_g, найменше значення функції f_l та відповідні їм точки w_h, w_g та w_l.

Крок 3. Знайдемо центр ваги всіх точок, за винятком точки w_h. Нехай центром ваги буде $w_0 = \frac{1}{n}\sum_{i \ne h} w_i$. Обчислимо $f(w_0) = f_0$.

Крок 4. Зручніше за все почати переміщення від точки w_h. Відобразивши точку w_h щодо точки w_0, одержимо точку w_r та

знайдемо $f(w_r) = f_r$.

Операція **відображення** ілюструється рис. 4.1.

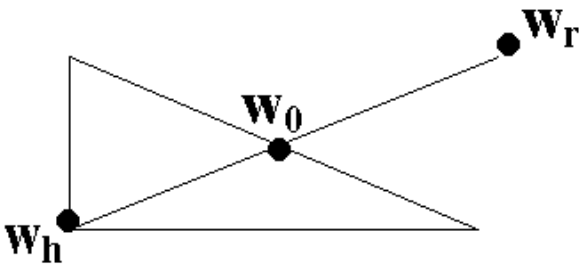


Рис 4.1 - Операція відображення симплексу

Якщо $a > 0$ - коефіцієнт відображення, то положення точки w_r визначається таким чином:

$$w_r - w_0 = a(w_0 - w_h),$$

тобто $$w_r = (1+a)w_0 - aw_h,$$

де $a = |w_r - w_0| / |w_0 - w_h|$.

Крок 5. Порівняємо значення функцій f_r та f_l.

1. Якщо $f_r < f_l$, то ми одержали найменше значення функції. Напрямок із точки w_0 у точку w_r найбільш зручний для переміщення. Таким чином, ми робимо **розтягання** в цьому напрямку і знаходимо точку w_e і значення функції $f_e = f(w_e)$.

Рис. 4.2 ілюструє операцію **розтягання симплексу**.

Коефіцієнт розтягання $v > 1$ можна знайти з виразу:

$$v = |x_e - x_0| / |x_r - x_0|.$$

а) Якщо $f_e < f_l$, то заміняємо точку w_h на точку w_e і перевіряємо (n+1)-у точку симплексу на збіжність до мінімуму (див. крок 8). Якщо збіжність досягнута, то процес зупиняється; у протилежному випадку повертаємося на крок 2.

б) Якщо $f_e \geq f_l$, то відкидаємо точку w_e. Очевидно, ми

перемістилися занадто далеко від точки w_0 до точки w_r. Тому варто замінити точку w_h, на точку w_r, у якій було отримано поліпшення (крок 5.1), перевірити збіжність і, якщо вона не досягнута, повернутися на крок 3.

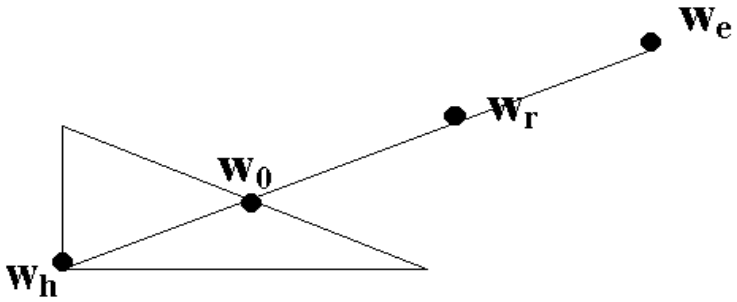


Рис 4.2 - Операція розтягання симплексу

2. Якщо $f_r > f_l$, але $f_r \le f_g$, то w_r є кращою точкою в порівнянні з іншими двома точками симплексу і ми заміняємо точку w_h на точку w_r і, якщо збіжність не досягнута, повертаємося на крок 2, тобто виконуємо пункт 5.1 б), описаний вище.

3. Якщо $f_r > f_l$ і $f_r > f_g$, то перейдемо на крок 6.

Крок 6. Порівняємо значення функцій f_r і f_h .

1. Якщо $f_r > f_h$, то переходимо безпосередньо до кроку стиску 6.2; у протилежному випадку, якщо $f_r < f_h$, то заміняємо точку x_h на точку x_r і значення функції f_h на значення функції f_r, запам'ятовуємо значення $f_r > f_g$ із кроку 5.2, приведеного вище, потім переходимо на крок 6.2.

2. У цьому випадку $f_r > f_h$, тому ясно, що ми перемістилися занадто далеко від точки x_h до точки w_0. Спробуємо виправити це, знайшовши точку w_c (а потім f_c) за допомогою кроку **стиску**, показаного на рис. 4.3.

Якщо $f_r > f_h$, то відразу переходимо до кроку стиску і знаходимо точку w_c зі співвідношення $w_c - w_0 = b(w_h - w_0)$, де b $(0 < b < 1)$ - коефіцієнт стиску. Тоді $w_c = bw_h + (1\text{-}b)w_0$.

Якщо $f_r < f_h$, то спочатку замінимо точку w_h на точку w_r, а потім зробимо стиск. Тоді точку w_c (див. рис. 4.4) знайдемо зі співвідношення: $w_c = bw_r + (1\text{-}b)w_0$.

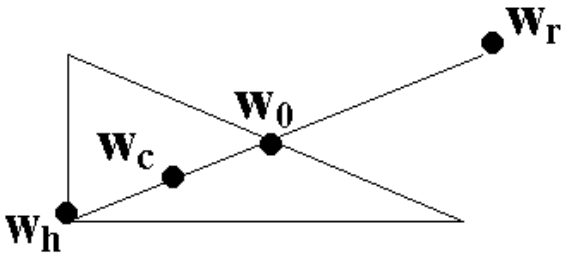


Рис. 4.3 - Крок стиску симплексу

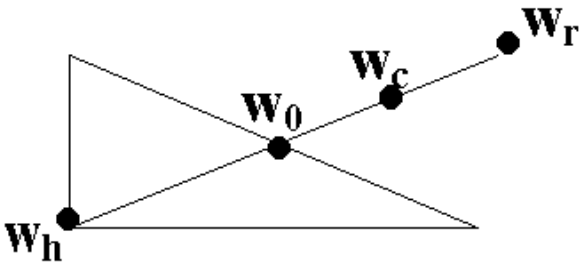


Рис. 4.4 – Стиск симплексу

Крок 7. Порівняємо значення функцій f_c та f_h.

1. Якщо $f_c < f_h$, то заміняємо точку w_h на точку w_c, і якщо збіжність не досягнута, то повертаємося на крок 2.

2. Якщо $f_c > f_h$, то очевидно, що всі наші спроби знайти значення менше f_h закінчилися невдачею, тому ми переходимо на крок 8.

Крок 8. На цьому кроці ми зменшуємо розмірність симплексу розподілом навпіл відстані від кожної точки симплексу до w_l — точки, що визначає найменше значення функції. Таким чином, точка w_i заміняється на точку $w_i + 0{,}5(w_i - w_l)$.

Потім обчислюємо f_i, для i = 1, 2,..., (n +1), перевіряємо збіжність і, якщо вона не досягнута, повертаємося на крок 3.

Крок 9. Перевірка збіжності заснована на тому, що стандартне відхилення (n+1)-го значення функції було менше деякого заданого малого значення ε.

У цьому випадку обчислюється

$$\sigma^2 = \sum_{i=1}^{n+1} \frac{\left(f_i - \bar{f}\right)^2}{n+1}, \text{ де } f = \sum_{i=1}^{n+1} f_i \Big/ n+1.$$

Якщо $\sigma < \varepsilon$, то всі значення функції дуже близькі один до одного, і тому вони, можливо, лежать поблизу точки мінімуму функції x_l.

Коефіцієнти a, b та v у вищенаведеній процедурі є відповідно коефіцієнтами відображення, стиску і розтягання. Нелдер і Мід рекомендують брати a = 1, b = 0,5 та v = 2. Рекомендація заснована на результатах експериментів з різними комбінаціями значень. Ці значення параметрів дозволяють методу бути ефективним у різних складних ситуаціях.

4.2 Метод Хука-Дживса

В основу **методу Хука – Дживса** покладено ту обставину, що пошук, який періодично проводиться у напрямку $d^{(i)} = w^{(i)} - w^{(i-1)}$ дозволяє істотно прискорити збіжність. Метод Хука-Дживса є одним з перших алгоритмів, у яких при визначенні нового напрямку пошуку враховується інформація, отримана на попередніх ітераціях.

Власне кажучи, процедура Хука - Дживса представляє собою комбінацію "**досліджуючого**" **пошуку** з циклічною зміною змінних і прискорюючого **пошуку за зразком** з використанням визначених евристичних правил. Досліджуваний пошук орієнтований на виявлення характеру локального поводження цільової функції і визначення напрямків уздовж «ярів». Отримана в результаті досліджуваного пошуку інформація потім використовується в процесі пошуку за зразком при русі по «ярах».

Для проведення **досліджуючого пошуку** необхідно задати величину кроку, що може бути різною для різних координатних напрямків і змінюватися в процесі пошуку. Досліджуючий пошук починається в деякій вихідній точці. Якщо значення цільової функції в пробній точці не перевищує значення функції у вихідній точці, то крок пошуку розглядається як успішний. У протилежному випадку необхідно повернутися в попередню точку і зробити крок у

протилежному напрямку з наступною перевіркою значення цільової функції. Після перебору всіх N координат досліджуючий пошук завершується. Отриману в результаті точку називають **базовою точкою.**

Пошук за зразком полягає в реалізації єдиного кроку з отриманої базової точки уздовж прямої, що з'єднує цю точку з попередньою базовою точкою. Нова точка зразка визначається відповідно до формули:

$$w^{(k+1)}_p = w^{(k)}+(w^{(k)}-w^{(k-1)}).$$

Як тільки рух за зразком не приводить до зменшення цільової функції, точка $w^{(k+1)}_p$ фіксується в якості тимчасової базової точки і знову проводиться досліджуючий пошук. Якщо в результаті виходить точка з меншим значенням цільової функції, ніж у точці $w^{(k)}$, то вона розглядається як нова базова точка $w^{(k+1)}$. З іншого боку, якщо досліджуваний пошук невдалий, необхідно повернутися в точку $w^{(k)}$ і провести досліджуючий пошук з метою виявлення нового напрямку мінімізації. В результаті виникає ситуація, коли такий пошук не приводить до успіху. У цьому випадку потрібно зменшити величину кроку шляхом введення деякого множника і відновити пошук. Пошук завершується, коли величина кроку стає досить малою.

Метод пошуку Хука — Дживса

Крок 1. Визначити: початкову точку w(0), збільшення Δ_i, i = 1, 2, ..., N, коефіцієнт зменшення кроку a > 1, параметр закінчення пошуку $\varepsilon > 0$.

Крок 2. Провести досліджуючий пошук.

Крок 3. Чи був досліджуваний пошук вдалим (чи знайдена точка з меншим значенням цільової функції)?

Так: перейти до кроку 5.

Ні: перейти до кроку 4.

Крок 4. Перевірка на закінчення пошуку. Чи виконується нерівність $\|\Delta w\| < \varepsilon$?

Так: припинити пошук; поточна точка апроксимує точку оптимуму w^*.

Ні: зменшити збільшення по формулі:

$$\Delta_i = \Delta_i / a,\ i=1, 2, \ldots, N.$$

Перейти до кроку 2.

Крок 5. Провести пошук за зразком:

$$w^{(k+1)}{}_p = w^{(k)} + (w^{(k)} - w^{(k-1)})$$

Крок 6. Провести пошук, що досліджується, використовуючи $w^{(k+1)}{}_p$ як базову точку; нехай $w^{(k+1)}$ — отримана в результаті точка.

Крок 7. Чи виконується нерівність $f(w^{(k+1)}) < f(w^{(k)})$?

Так: покласти $w^{(k-1)} = w^{(k)}$, $w^{(k)} = w^{(k+1)}$. Перейти до кроку 5.

Ні: перейти до кроку 4.

4.3 Метод спряжених напрямків Пауелла

Найбільш ефективним з алгоритмів прямого пошуку є метод, розроблений Пауеллом. При роботі цього алгоритму інформація, отримана на попередніх ітераціях, використовується для побудови векторів напрямків пошуку, а також для усунення зациклення послідовності координатних пошуків. Метод орієнтований на вирішення задач із квадратичними цільовими функціями і ґрунтується на фундаментальних теоретичних результатах.

Задачі з квадратичними цільовими функціями займають важливе місце в теорії оптимізації за двома причинами.

1. **Квадратична функція** являє собою найпростіший тип нелінійних функцій, для яких може бути сформульована задача безумовної оптимізації (лінійні функції не мають внутрішні оптимуми). Отже, якщо за допомогою того або іншого методу успішно вирішуються задачі оптимізації з цільовими функціями загального виду, то такий метод повинний виявитися ефективним при вирішенні задач із квадратичними функціями.

2. В околиці точки оптимуму будь-яку нелінійну функцію можна апроксимувати квадратичною функцією (оскільки лінійний член розкладання Тейлора обертається в нуль). Отже, робота алгоритму при вирішенні задач із квадратичними функціями дозволяє одержати визначене уявлення про збіжність алгоритму у випадку, коли мінімізується функція загального вигляду.

Основна ідея алгоритму полягає в тому, що якщо квадратична

функція N змінних приведена до вигляду суми повних квадратів, то її оптимум може бути знайдений у результаті реалізації N одновимірних пошуків по перетворених координатних напрямках.

Процедура перетворення квадратичної функції $q(w) = a+b^Tw+0,5Cw$ до вигляду суми повних квадратів еквівалентна знаходженню такої матриці перетворення T, що приводить матрицю квадратичної форми до діагонального вигляду. Таким чином, задана квадратична форма $Q(w) = w^TCw$ шляхом перетворення $w = Tz$ приводиться до вигляду $Q(x)=z^TT^TCTz=Z^TDz$, де D — діагональна матриця, тобто елементи D відмінні від нуля тільки при i=j.

Нехай t_j — j-й стовпець матриці T. Тоді перетворення дозволяє записати кожний вектор w у вигляді лінійної комбінації векторів-стовпців t_j:

$$w = Tz = t_1z_1 + t_2z_2 + ... + t_Nz_N.$$

Іншими словами, замість координат вектора w у стандартній координатній системі, зумовленій множиною векторів $e^{(i)}$, використовуються координати вектора в новій координатній системі, заданій векторами t_j. Крім того, система векторів t_j відповідає головним осям розглянутої квадратичної форми, оскільки матриця квадратичної форми приводиться до діагонального вигляду.

Отже, за допомогою перетворення змінних квадратичної функції будується нова система координат, що збігається з головними осями квадратичної функції. Отже, одномірний пошук точки оптимуму в просторі перетворених змінних z еквівалентний пошуку уздовж кожної з головних осей квадратичної функції. Так як напрямок головних осей визначаються векторами t_j, то одномірний пошук проводиться в напрямках, заданих цими векторами.

Нехай C - симетрична матриця порядку NxN, напрямки $s^{(1)}$, $s^{(2)}$, ..., $s^{(r)}$, $r \le N$, називаються C-спряженими, якщо ці напрямки лінійно незалежні і $s^{(i)}TCs^{(j)} = 0$ для всіх $i \ne j$.

При проведенні розрахунків переважніше будувати систему спряжених напрямків, виходячи з однієї початкової точки, що легко здійснити за допомогою одиничних координатних векторів $e^{(1)}$, $e^{(2)}$, ..., $e^{(N)}$.

Сформулюємо **узагальнену властивість рівнобіжного підпростору**.

Якщо точка $y^{(1)}$ знайдена в результаті пошуку з точки $w^{(1)}$ уздовж кожного з M (M<N) спряжених напрямків, а точка $y^{(2)}$ отримана в результаті пошуку з точки $w^{(2)}$ уздовж кожного з тих же M спряжених напрямків $s^{(1)}$, $s^{(2)}$, ..., $s^{(M)}$, то вектор $(y^{(2)} - y^{(1)})$ задає напрямок, спряжений з усіма обраними M напрямками.

Алгоритм спряжених напрямків Пауелла запишемо в наступному вигляді.

Крок 1. Задати початкову точку $w^{(0)}$ та систему N лінійно незалежних напрямків; можливий випадок, коли $s^{(i)} = e^{(i)}$, i = l, 2, ..., N.

Крок 2. Мінімізувати f(w) при послідовному русі по (N+1) напрямках; при цьому отримана раніше точка мінімуму береться в якості вихідної, а напрямок $s^{(N)}$ використовується як при першому, так і останньому пошуку.

Крок 3. Визначити новий спряжений напрямок за допомогою узагальненої властивості рівнобіжного підпростору.

Крок 4. Замінити $s^{(1)}$ на $s^{(2)}$ і т.д. Замінити $s^{(N)}$ спряженим напрямком.

Перейти до кроку 2.

Для того, щоб застосувати викладений метод на практиці, його необхідно доповнити процедурами перевірки збіжності і лінійної незалежності системи напрямків. Перевірка лінійної незалежності особливо важлива в тих випадках, коли функція f(w) не є квадратичною.

Зі способу побудови алгоритму випливає, що у випадку, коли цільова функція квадратична і має мінімум, точка мінімуму знаходиться в результаті реалізації N циклів, що включають кроки 2, 3 і 4, де N — кількість змінних. Якщо ж функція не є квадратичною, то потрібно більш ніж N циклів. Разом з тим можна дати строгий доказ того, що при деякому припущенні метод Пауелла збігається до точки локального мінімуму із суперлінійною швидкістю.

Розглянутий метод дозволяє побудувати послідовність точок $w^{(k)}$, що збігається до рішення w*. Метод збігається, якщо нерівність

$$\frac{\left\|\varepsilon^{(k+1)}\right\|}{\left\|\varepsilon^{(k)}\right\|} \le 1, \text{ де } \varepsilon^{(k)} = w^{(k)} - w^{*}, \qquad (4.1)$$

виконується на кожній ітерації. Оскільки при розрахунках звичайно оперують кінцевими десятковими дробами, навіть найефективніший алгоритм вимагає проведення нескінченної послідовності ітерацій. Тому в першу чергу представляють інтерес асимптотичні властивості збіжності методів, що досліджуються. Будемо говорити, що алгоритм має збіжність порядку r, якщо

$$\lim_{k\to\infty} \frac{\left\|\varepsilon^{(k+1)}\right\|}{\left\|\varepsilon^{(k)}\right\|^{r}} = C ,$$

де C — постійна величина.

З формули (4.1) випливає, що при r = 1 має місце нерівність C ≤ 1. Якщо r = 1 або r = 2, то алгоритм характеризується лінійною або квадратичною швидкістю збіжності, відповідно. При r = 1 та C = 0 алгоритм характеризується суперлінійною швидкістю збіжності.

РОЗДІЛ 5. БАГАТОВИМІРНІ ГРАДІЄНТНІ МЕТОДИ НЕЛІНІЙНОЇ ОПТИМІЗАЦІЇ

Нехай задано вигляд (правило обчислення) деякої функції y = f(w, x), де w та x – складені вектори-стовпці змінних і відомі реалізації функції y при відомих значеннях x. Потрібно оцінити невідомі параметри w, таким чином, щоб значення функції y при цих параметрах незначно відрізнялися від відомих реалізацій y* при однакових заданих значеннях x, тобто необхідно мінімізувати деяку цільову функцію F(w), де w-складений вектор-стовпець змінних, що керуються, за кінцеву кількість ітерацій.

Представимо функцію F(w) у вигляді інтегрального критерію якості

$$F(w)=\frac{1}{k}\sum_{m=1}^{k}Q(\varepsilon(w,m)),$$

де k – номер поточної ітерації, m=1, 2,..., k – номера попередніх ітерацій, а Q(ε (w,k)) – деякий миттєвий критерій якості, що залежить від вектора помилки Q(ε (w,m)):

$$\varepsilon(w,m) = y(m)-y^*(m)$$

і який має вигляд квадратичної форми:

$$Q(\varepsilon(w,m)) = \varepsilon^T(w,m)R\varepsilon(w,m),$$

де R - позитивно визначена матриця.

Градієнтні методи мінімізації інтегрального критерію якості засновані на використанні градієнта цільової функції F(w). Ці методи носять ітеративний характер, тому що компоненти градієнта виявляються нелінійними функціями.

Усі далі розглянуті методи засновані на ітераційній процедурі, що реалізована відповідно до формули:

$$w_{k+1}=w_k+\alpha_k s(w_k),$$

де w_k, w_{k+1} – поточне і нове наближення значень керувальних змінних до оптимального рішення, відповідно, α_k – крок збіжності, $s(w_k)$-напрямок пошуку в N-вимірному просторі змінних, що керуються. Спосіб визначення $s(w_k)$ і α_k на кожній ітерації залежить від особливостей конкретного методу.

5.1 Метод Коші

Метод Коші (**метод найшвидшого спуску,** у навчанні нейронних мереж відомий, як **алгоритм зворотного поширення помилки першого порядку, Backpropagation**) полягає в реалізації правила:

$$w_k = w_{k-1} - \gamma \nabla_w Q(\varepsilon(w_{k-1}, k)) = w_{k-1} - \gamma \frac{\partial Q(\varepsilon(w_{k-1}, k))}{\partial w} .$$

Позначивши поточний градієнт $g = \frac{\partial Q}{\partial w}$, одержимо:

$$w_{k+1} = w_k - \gamma_k g_k,$$

де γ_k - швидкість навчання.

Величина γ або покладається постійною, і тоді звичайно послідовність w_k сходиться в околицю оптимального значення w, або вона є спадаючою функцією часу так, як це робиться в стохастичних алгоритмах оптимізації й адаптації.

Дана процедура може виконуватися доти, доки значення керувальних змінних не стабілізуються, або доки помилка не зменшиться до прийнятного значення.

Слід зазначити, що дану процедуру характеризує повільна швидкість збіжності і можливість влучення в локальні мінімуми функціонала.

5.2 Метод Ньютона

Метод Ньютона або алгоритм зворотного поширення помилки другого порядку відносно до задачі мінімізації цільової

функції (1) має вигляд:

$$w_k = w_{k-1} - \left(\sum_{m=1}^{k} \frac{\partial}{\partial w} \left(\frac{\partial Q(m)}{\partial w} \right)^T \right)^{-1} \gamma \nabla_w Q(\varepsilon(w_{k-1}, k)) \ .$$

Обчислення градієнта $\nabla_w Q(\varepsilon(w_{k-1},k))$ може бути виконане за допомогою методу Коші.

У результаті задача зводиться до обчислення матриці других похідних $\sum_{m=1}^{k} \frac{\partial}{\partial w} \left(\frac{\partial Q(m)}{\partial w} \right)^T$ мінімізуючого функціонала.

Метод Ньютона за певних умов має істотно більшу швидкість збіжності, ніж градієнтний метод.

5.3 Алгоритми спряжених градієнтів

Алгоритми спряжених градієнтів представляють собою підклас методів, що збігаються квадратично. Алгоритми спряжених градієнтів використовують метод Коші для обчислення похідних цільової функції щодо керувальних змінних.

Всі алгоритми спряжених градієнтів на першій ітерації починають роботу з пошуку у найкрутішому напрямку спуску - протилежному градієнту цільової функції:

$$p_0 = -g_0.$$

Це напрямок, у якому цільова функція зменшується найбільш швидко. Однак зазначимо, що це не обов'язково забезпечує найшвидшу збіжність.

Після цього виконується лінійний пошук, щоб визначити оптимальну відстань для руху у поточному напрямку пошуку:

$$w_{k+1} = w_k + \alpha_k p_k.$$

Наступний напрямок пошуку визначається так, щоб він був спряженим з попередніми напрямками. Пошук відбувається за спряженим напрямком градієнта, щоб визначити розмір кроку, що мінімізує цільову функцію. Загальна процедура для визначення

нового напрямку пошуку поєднує новий найкрутіший напрямок спуску з попереднім напрямком пошуку: $p_{k+1} = -g_k + \beta_k p_{k-1}$.

У більшості алгоритмів спряжених градієнтів розмір кроку корегується при кожній ітерації, на відміну від інших алгоритмів, де швидкість навчання використовується для визначення розміру кроку.

Різні версії алгоритмів спряжених градієнтів відрізняються способом, за яким обчислюється константа β.

Для **алгоритму Флетчера-Рівса** правило обчислення константи β має вигляд:

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}},$$

де β - відношення квадрата норми поточного градієнта до квадрата норми попереднього градієнта.

Для **алгоритму Полака-Ріб'єра** правило обчислення константи β має вигляд:

$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{g_{k-1}^T g_{k-1}},$$

де β - внутрішній добуток попередньої зміни в градієнті і поточного градієнта, розділений на квадрат норми попереднього градієнта.

5.4 Партан – метод

Мінімізація квадратичної цільової функції $F(w) = -\frac{1}{2}(w, Aw) + bw$, де A – матриця, що задає позитивно визначену квадратичну форму (w,Aw), b - деякий вектор. Відповідно до партан-методу алгоритм має наступний вигляд.

А. Початковий крок.

1) У довільній точці w_0 знаходиться градієнт g_1 функції F(w);
2) на прямій $w = w_0 + \lambda g_1$ знаходиться точка w_1, що доставляє максимум функції F(w).

Б. Загальний крок. Нехай вже знайдені точки w_0, w_1, ...,w_{k-1}, w_k.

1) у точці w_k знаходиться градієнт g_{k+1};
2) на прямій $w = w_k + \lambda\, g_{k+1}$ визначається точка w'_{k+1}, у якій функція F(w) досягає умовного екстремуму за формулою:

$$w'_{k+1} = \lambda'_k g_{k+1} + w_k\,, \text{ де } \lambda'_k = \frac{g^2_{k+1}}{(g_{k+1}, Ag_{k+1})};$$

3) на прямій $w = w_{k-1} - \lambda\,(w'_{k+1} - w_{k-1})$ знаходиться точка w_{k+1}, що доставляє умовний екстремум F(w), за формулою:

$$w'_{k+1} = w_{k-1} + \lambda''_k (w'_{k+1} - w_{k-1})\,,$$

де $$\lambda''_k = \frac{((w'_{k+1} - w_{k-1}), g_k) g^2_{k+1}}{((w'_{k+1} - w_{k-1}), A(w'_{k+1} - w_{k-1}))}.$$

В. Зупинення алгоритму. Процес припиняється, коли на черговому кроці виявиться, що $g_{m+1} = 0$. Тоді w_m - точка максимуму F(w).

5.5 Метод Заутендайка

Метод Заутендайка (**метод проекцій**) реалізується у відповідності з наступними кроками.

Крок 1. Почати в w_0 і покласти проектуючу матрицю P_0=I.

Крок 2. Для k-го кроку проектуюча матриця обчислюється таким чином:

$$P_k = P_{k-1} - P_{k-1}\Delta g_k \left[\Delta g_k{}^T P_{k-1} \Delta g_k\right]^{-1} \Delta g_k P_{k-1}\,.$$

Крок 3. Якщо $P_k g_k \neq 0$, покласти s_k = - $P_k g_k$ і мінімізувати функцію F(w) у напрямку s_k; точка мінімуму w_{k+1}. Повторити крок 2. Після того, як пройдено n напрямків пошуку, де n - кількість керувальних змінних, почати знову з кроку 1 при $x_0 = x_n$.

Крок 4. Якщо $P_k g_k = 0$ і $g_k = 0$, закінчити пошук.

Крок 5. Якщо $P_k g_k = 0$ і $g_k \neq 0$, знову почати з кроку 1, поклавши $x_0 = x_k$.

5.6 Багатопараметричний пошук

Метод Міля-Кентрелла полягає в реалізації послідовності кроків.

На першому кроці $\Delta w_{k-1} = 0$, а w_0 повинно бути задано.

На k-му кроці:

1. Обчислюються w_k, g_k та Δw_{k-1}.

2. Користуючись одним з ефективних способів двовимірного пошуку, знаходять з необхідною точністю λ_k^1 та λ_k^2.

3. Обчислюють x_{k+1}: $x_{k+1}=x_k - \lambda_k^1 g_k + \lambda_k^2 \Delta w_{k-1}$ та переходять до пункту 1.

4. Кожний (n+1)-й крок, де n - кількість керувальних змінних, починається з $\Delta w_{k-1} = 0$.

5. Процес закінчується, коли $|\Delta F(w)| < \varepsilon$.

У **методі Крегга-Леві** на кожному кроці здійснюється багатовимірний пошук за більшим числом параметрів, ніж у методі Міля-Кентрелла. Кожний наступний вектор w знаходиться за формулою:

$$w_{k+1} = w_k - \lambda_k^0 g_k + \sum_{i=1}^{m} \lambda_k^i \Delta w^{i-1}, \text{ при } m \leq n\text{-}1,$$

де w^i – і-та керувальна змінна.

На початкових кроках алгоритму Δw^i можна покласти рівними нулю.

5.7 Квазіньютонівські методи

Ці методи подібні методам спряжених градієнтів, оскільки також засновані на властивостях квадратичних функцій. Відповідно до викладених вище методів пошуку рішення здійснюється за системою спряжених напрямків, тоді як квазіньютонівські методи

мають позитивні риси методу Ньютона, однак використовують тільки перші похідні. В усіх методах зазначеного класу побудова векторів напрямків пошуку здійснюється за допомогою формули:

$$w_{k+1}=w_k - \alpha_k A_k g_k,$$

де A_k - матриця порядку N x N, що зветься **метрикою**. Методи пошуку уздовж напрямків, обумовлених цією формулою, називаються **методами змінної метрики**, оскільки матриця A змінюється на кожній ітерації.

Градієнт g_k може бути знайдений за допомогою методу Коші і, отже, квазіньютонівські методи полягають у знаходженні матриці A_k.

Метод Давідона-Флетчера-Пауелла для знаходження A_k використовує наступну формулу:

$$\Delta w_k = w_{k+1} - w_k ,$$

$$\Delta g_k = g_{k+1} - g_k ,$$

$$A_k = A_{k-1} + \frac{\Delta w_{k-1}\Delta w_{k-1}^T}{\Delta g_{k-1}^T \Delta g_{k-1}} - \frac{A_{k-1}\Delta g_{k-1}\Delta g_{k-1}^T A_{k-1}}{\Delta g_{k-1}^T A_{k-1}\Delta g_{k-1}},$$

причому звичайно зручно вибирати A_0=I, де I – одинична матриця.

Метод Бройдена-Флетчера-Шенно (ранг 2) реалізується відповідно до рекурентної формули:

$$A_{k+1} = \left[I - \frac{\Delta w_k \Delta g_k^T}{\Delta w_k^T \Delta g_k} \right] A_k \left[I - \frac{\Delta w_k \Delta g_k^T}{\Delta w_k^T \Delta g_k} \right] + \frac{\Delta w_k \Delta w_k^T}{\Delta w_k^T \Delta g_k} .$$

До числа головних переваг цього методу варто віднести не завжди обов'язкову необхідність повернення до початкової ітерації алгоритму і відносно слабку залежність від точності обчислень при проведенні одновимірного пошуку.

Метод Бройдена (ранг 1) реалізується у відповідності з формулою:

$$A_{k+1} = A_k + \frac{[\Delta w_k - A_k \Delta g_k][\Delta w_k - A_k \Delta g_k]^T}{[\Delta w_k - A_k \Delta g_k]^T \Delta g_k} .$$

Метод Пірсона № 2 для знаходження матриці A_{k+1} використовує рекурентний вираз:

$$A_{k+1} = A_k + \frac{[\Delta w_k - A_k \Delta g_k]\Delta w_k^T}{\Delta w_k^T \Delta g_k} , A_0 = R_0,$$

де R_0 – довільна позитивно визначена симетрична матриця.

Метод Пірсона № 3 реалізується у відповідності з формулою:

$$A_{k+1} = A_k + \frac{[\Delta w_k - A_k \Delta g_k][A_k \Delta g_k]^T}{\Delta g_k^T A_k \Delta g_k} , A_0 = R_0,$$

де R_0 – довільна позитивно визначена симетрична матриця.

Проективний метод Ньютона-Рафсона реалізується у відповідності з формулою:

$$A_{k+1} = A_k - \frac{(A_k \Delta g_k)(A_k \Delta g_k)^T}{\Delta g_k^T A_k \Delta g_k} , A_0 = R_0,$$

де R_0 – довільна позитивно визначена симетрична матриця.

5.8 Алгоритм Левенберга-Марквардта

Алгоритм Левенберга-Марквардта вимагає наявності інформації про значення других похідних цільової функції.

В алгоритмі Левенберга-Марквардта використовується метод Коші, щоб обчислити якобіан J цільової функції щодо керувальних змінних. Керувальні змінні змінюються відповідно до коригувального правила, що у матричній формі має вигляд:

$$H=J^TJ,\ g=J^Te,\ w_{k+1} = w_k - [H_k+\eta I]^{-1}g_k,$$

де J - якобіан, e - вектор помилок, η - скаляр, I – одинична матриця.

Адаптивне значення η збільшується в η^+ раз доти, доки значення цільової функції не зменшиться. Після чого зміни вносяться в мережу і η зменшується в η^- раз.

Алгоритм Левенберга-Марквардта має вигляд:

Крок 1. Ініціалізація: задаються початкові значення керувальних змінних, а також граничні значення параметрів закінчення роботи алгоритму.

Лічильник числа циклів навчання k=1.

Крок 2. Перевірка збіжності та умов закінчення роботи. Якщо збіжність досягнута, k>Epochs або робота повинна бути припинена – закінчення роботи.

Крок 3. Якщо $\eta \leq \eta_{max}$, то перехід до кроку 4, інакше до кроку 7.

Крок 4. Обчислити значення помилки і скорегувати відповідним чином керувальні змінні:

$$H=J^TJ,\ g=J^Te,$$

$$w_{k+1} = w_k - [H_k+\eta I]^{-1}g_k.$$

Крок 5.Обчислити нове значення цільової функції. Якщо воно менше поточного, то повернути попередні значення керувальним змінним, змінити η: $\eta = \eta\eta^-$ і вийти з циклу, інакше зафіксувати значення керувальних змінних і змінити η: $\eta = \eta\eta^+$.

Крок 6. Перехід до кроку 4.

Крок 7. k=k+1.

Крок 8. Перехід до кроку 2.

Навчання зупиняється, якщо виконується хоча б одна з умов:

1) досягнуто максимально припустиме число циклів навчання Epochs;

2) цільова функція мінімізована (ціль досягнута);

3) градієнт цільової функції менше мінімального припустимого значення;

4) η перевищує максимальне припустиме значення η_{max};

5) цільова функція збільшилась більше ніж у MAX_FAIL разів, починаючи з останнього разу зменшення.

5.9 Узагальнений градієнтний алгоритм

Подібність розглянутих градієнтних методів дає підставу для розробки узагальненого градієнтного алгоритму, що має наступний вигляд.

Крок 1. Ініціалізація. Задати початкові значення керувальних змінних w, максимально припустиме число циклів навчання Epochs, параметр збіжності алгоритму ε_1, параметр збіжності уздовж прямої ε_2 (для простоти можна покласти $\varepsilon_2 = \varepsilon_1$).

Крок 2. Покласти лічильник ітерацій k=0.

Крок 3. Обчислити компоненти $\frac{\partial Q(\varepsilon(w_{k-1}, k))}{\partial w}$.

Крок 4. Чи виконується рівність $\left\| \frac{\partial Q(\varepsilon(w_{k-1}, k))}{\partial w} \right\| \le \varepsilon_1$?

Так: Збіжність досягнута. Перехід до кроку 13.

Ні: перейти до кроку 5.

Крок 5. Чи виконується нерівність k>Epochs?

Так: Досягнуто максимальне число циклів навчання, збіжність не досягнута.

Перехід до кроку 13.

Ні : перейти до кроку 6.

Крок 6. Обчислити $s(w_k)$.

Крок 7. Чи виконується нерівність $\frac{\partial Q(\varepsilon(w_{k-1}, k))^T}{\partial w} s(w_k) < 0$?

Так: перейти до кроку 9.

Ні : покласти $s(w_k) = -\frac{\partial Q(\varepsilon(w_{k-1}, k))}{\partial w}$. Перейти до кроку 9.

Крок 8. Знайти таке значення α_k, при якому $F(w_k + \alpha_k s(w_k)) \to \min$, використовуючи параметр ε_2.

Крок 9. Покласти $w_{k+1} = w_k + \alpha_k s(w_k)$.

Крок 10. Чи виконується нерівність $F(w_{k+1}) < F(w_k)$?

Так: перейти до кроку 11.

Ні : Закінчення пошуку: немає зменшення функції. Перехід до кроку 13.

Крок 11. Чи виконується нерівність $\frac{\|\Delta w\|}{\|w_k\|} \le \varepsilon_1$?

Так: Закінчення пошуку: немає просування до рішення. Перехід до кроку 13.

Ні : перейти до кроку 12.

Крок 12. Покласти k=k+1. Перейти до кроку 3.

Крок 13. Зупинення.

Необхідно зазначити, що в процесі одновимірного пошуку необхідно по можливості уникати точних обчислень, тому що на виконання операцій пошуку уздовж прямої витрачається дуже значна частина загального часу обчислень.

5.10 Окремі випадки узагальненого градієнтного алгоритму

У вищеописаному узагальненому градієнтному алгоритмі можна використовувати різні градієнтні методи шляхом визначення відповідних напрямків пошуку на кроці 6. Визначаючи відповідним чином $s(w_k)$, можна трансформувати розглянутий узагальнений градієнтний алгоритм практично в будь-який градієнтний алгоритм навчання багатошарових нейромереж, що дозволяє істотно спростити розробку нейромережевих систем без жорсткої прив'язки до визначеного градієнтного алгоритму.

Для **методу Ньютона** або алгоритму зворотного поширення помилки другого порядку крок 6 узагальненого градієнтного алгоритму має вигляд:

$$s(w_k)=-\left(\sum_{m=1}^{k}\frac{\partial}{\partial w}\left(\frac{\partial Q(m)}{\partial w}\right)^T\right)^{-1}\nabla_w Q(\varepsilon(w_{k-1},k)) .$$

Для **алгоритмів спряжених градієнтів** крок 6 узагальненого градієнтного алгоритму має вигляд:

$$s(w_k)=-g_k+\beta_k\, s(w_{k-1}).$$

Різні версії алгоритмів спряжених градієнтів відрізняються способом, за яким обчислюється константа β.

Для **алгоритму Левенберга-Марквардта** крок 6 узагальненого градієнтного алгоритму має вигляд:

$$s(w_k)=-[H_k+\eta I]^{-1}g_k.$$

5.11 Інтегральний метод оптимізації

При експериментальному дослідженні об'єктів часто зустрічаються задачі пошуку екстремальних значень цільових функцій, у якості яких вибираються критерії ефективності і якості досліджуваного об'єкта, а також різні критерії оптимальності в задачах статистичного оцінювання і планування експериментів. Для вирішення цих задач може бути з успіхом застосований інтегральний метод оптимізації.

Нехай y = f(x) є цільова функція, де залежна змінна y представляє вихідний параметр якості й ефективності досліджуваного об'єкта, а x - векторна вхідна величина об'єкту. Величина y спостерігається в умовах випадкового шуму. Міра кожної точки x, наприклад її ефективність чи якість, визначається значенням цільової функції, тобто $d\mu_x = f(x)dx$. Потрібно знайти таку оцінку x^* значення вхідної величини x, що забезпечує мінімум функції ризику:

$$R(x^*) = \min_{x^*} \int_{\Omega x} W(x^*, x)d\mu_x ,$$

де $W(x^*, x)$ - функція втрат, що зростає зі збільшенням відхилення точки x^* від точки x в області вхідних змінних Ωx. Отже, задача зводиться до мінімізації інтеграла від функції втрат $W(x^*, x)$ з урахуванням міри μ_x простору вхідних змінних, обумовлених цільовою функцією f(x).

У залежності від вибору функції втрат $W(x^*,x)$ можна розрізняти, наприклад, наступні класи задач оптимізації.

1). Класична задача оптимізації.

Нехай $W(x^*,x) = 1 - \delta(x^* - x)$, тоді

$$\min_{x^*} \int_{\Omega x} (1 - \delta(x^* - x))f(x)dx = \Phi - \max_{x^*} f(x) , \text{ де } \Phi = \int_{\Omega x} f(x)dx .$$

Це відома класична задача максимізації за вхідними змінними x цільової функції f(x). Недолік класичної постановки задачі

оптимізації полягає в порівняно високій чутливості рішення до рівня випадкових шумів, що має місце при практичному вирішенні задачі в умовах дії випадкових помилок спостереження, наприклад за умови, що:

$$y = f(x) + \xi .$$

У цьому випадку при високому рівні випадкових перешкод ξ ускладнене вирішення з достатньою точністю задачі пошуку максимуму по x цільової функції f(x), тому що значення, за яким спостерігають, може випадково виявитися максимальним у точці x, віддаленій на значну відстань від точки максимуму x_{max} цільової функції f(x). Тому класична постановка задачі оптимізації знайшла широке застосування лише при детерміністському підході до задач оптимізації (при відсутності випадкових шумів), наприклад при вирішенні математичних задач, при моделюванні об'єктів, а також у випадку невисокого рівня випадкових перешкод, які можна не враховувати, що мають місце при проведенні лабораторних досліджень у спеціальних умовах. В умовах же промислового виробництва з великим рівнем шумів потрібно застосовувати спеціальні міри зниження рівня випадкових помилок спостереження, що приводить до збільшення витрат на реалізацію пошукового методу максимізації, які можна врахувати введенням деякої вагової функції c(x). Тоді одержуємо, що функція ризику

$$R(x^*) = \int_{\Omega x} c(x)(1 - \delta(x^* - x))f(x)dx = \Phi^* - c(x^*)f(x^*)$$

і знаходження її мінімуму ускладнюється, якщо вагова функція c(x) не дорівнює деякій константі c_0.

2). Задача оптимізації за середньоквадратичним критерієм.

Нехай

$$W(x^*, x) = (x^* - x)^T B (x^* - x),$$

де B - матриця вагових коефіцієнтів. Тоді функція ризику $R(x^*)$ має мінімум у точці

$$x^* = \frac{\int_{\Omega x} x f(x) dx}{\int_{\Omega x} f(x) dx},$$

яка є «центром ваги» цільової функції f(x). Середньоквадратична оцінка збігається з точкою максимуму лише для симетричних цільових функцій. У загальному випадку це різні точки. Середньоквадратична оцінка є точкою, мінімально віддаленою в розумінні міри $d\mu_x = f(x)dx$ від усіх точок області Ωx, тобто з урахуванням вагової функції g(x) = f(x) вона має більше притягання до точок з найбільшими значеннями цільової функції. Ці властивості середньоквадратичної оцінки дозволяють використовувати її як наближену оцінку максимуму цільової функції. Основною перевагою середньоквадратичної оцінки є її низька чутливість до випадкових адитивних, некорегованих із входом помилок спостереження ξ з нульовим математичним сподіванням.

Середньоквадратична оцінка має визначені переваги і при вирішенні задачі максимізації багатоекстремальних цільових функцій, представлених у вигляді:

$$f(x) = f_1(x) + f_2(x),$$

де $f_1(x)$ - унімодальна функція, $f_2(x)$ - осцилююча функція типу випадкового поля або, що являє собою «рельєфний» (просторовий) шум. У цьому випадку при інтегруванні просторовий шум відфільтровується, а одержувана оцінка x^* виявляється порівняно близькою до точки глобального максимуму.

Інтегральний метод доцільно застосовувати при вирішенні задачі оптимізації в умовах масового виробництва виробів або масового експерименту, коли необхідно, наприклад, знайти таке налагодження параметрів устаткування, що було б оптимальним в розумінні мінімуму середньоквадратичного відхилення параметрів устаткування від параметрів виробів або досліджуваних зразків, причому при цьому враховується функція вартості виробів або зразків.

Інтегральний метод оптимізації може також застосовуватися і для

вирішення детермінованих задач максимізації цільової функції f(x), якщо застосувати монотонне перетворення $\Psi\{f(x)\}$, що не змінюючи положення екстремальної точки цільової функції, звужує функцію та робить її більш дельтоподібною. У результаті такого перетворення середньоквадратична оцінка наближується до точки максимуму цільової функції. Але при цьому виникає обчислювальна проблема, пов'язана з чисельним інтегруванням дельтоподібних функцій. Прості процедури Монте-Карло виявляються малоефективними, і в цьому випадку доцільно застосувати, наприклад, процедуру **методу суттєвої вибірки**, ідея якого полягає в тому, щоб вибирати випадкові вузли інтегрування з щільністю імовірності, пропорційної значенню функції, що інтегрується, тобто кількість випадкових точок повинна бути більше там, де функція, що інтегрується, має найбільші значення. Реалізація методу суттєвої вибірки на ЕОМ пов'язана з необхідністю генерування випадкових чисел, що мають складні функції щільності імовірності та мають визначені обчислювальні труднощі і значно знижують ефективність методу. Як просту реалізацію методу суттєвої вибірки можна запропонувати процедуру, засновану на поетапному застосуванні найпростішого **методу Монте-Карло** з рівномірною щільністю імовірності випадкових чисел у прямокутній області. На першому етапі прямокутна область генерування випадкових чисел вибирається такою, щоб вона включала область інтегрування Ωx .

На кожному наступному етапі обсяг прямокутної області зменшується в k^n раз, де n - розмірність простору вхідних змінних E_x, $k>1$, а центр області співпадає з точкою x^*_j, що відповідає середньому значенню підінтегральної функції, що отримане на попередньому етапі.

Нехай, наприклад, потрібно обчислити значення інтеграла:

$$\int_{\Omega x} f(x)dx .$$

Загальне число статистичних іспитів за методом Монте-Карло:

$$N = \sum_{j=1}^{s} N_j \ ,$$

де N_j - число іспитів на j-му етапі, s - загальне число етапів процедури. Значення оцінки інтеграла на кожному етапі обчислюється за формулою:

$$I = \frac{V_j}{N_j} \sum_{i=1}^{N_j} f\left(x_{ij}\right) \ ,$$

де V_j – об‘єм області генерування випадкових чисел x_{ij}. Дисперсія цієї оцінки інтеграла обчислюється за формулою:

$$D\{I_j\} = V_j^2 \frac{\sum_{i=1}^{N_j} \left(f\left(x_{ij}\right) - I_j\right)^2}{N_j\left(N_j - 1\right)} \ .$$

Загальна оцінка інтеграла може бути обчислена на підставі результатів інтегрування на кожному етапі за наступною формулою усереднення з вагою:

$$I = \sum_{j=1}^{s} c_j I_j \ ,$$

де значення вагового коефіцієнта знаходяться за формулою:

$$c_j = \frac{D^{-1}\{I_j\}}{\sum_{j=1}^{s} D^{-1}\{I_j\}} \ .$$

Перевагою розглянутої процедури методу суттєвої вибірки є простота реалізації, недоліком - можливість «усікання» підінтегральної функції на останніх етапах інтегрування, що приводить до зміщених оцінок інтеграла.

РОЗДІЛ 6. МЕТОДИ БАГАТОВИМІРНОЇ ОПТИМІЗАЦІЇ З ОБМЕЖЕННЯМИ

6.1 Метод штрафних функцій

Основна ідея **методу штрафної функції** полягає в перетворенні задачі мінімізації функції z = f(w) з відповідними обмеженнями, накладеними на w, у задачу пошуку мінімуму без обмежень функції z = f(w) + p(w).

Функція p(w) є **штрафною**. Необхідно, щоб при порушенні обмежень вона "штрафувала" функцію z, тобто збільшувала її значення. У цьому випадку мінімум z буде знаходитися усередині області обмежень. Функція p(w), що задовольняє цій умові, може бути не єдиною.

Задачу мінімізації можна сформулювати таким чином:
мінімізувати функцію z = f(w) при обмеженнях $c_j(w) > 0$, j = 1, 2, ..., m.

Функцію p(w) зручно записати таким чином:

$$p(w) = r\sum_{j=1}^{m}\frac{1}{c_j(w)}\ , \qquad (6.1)$$

де r - позитивна величина.

Тоді функція z = φ (w, r) приймає вигляд

$$z = \varphi(w, r) = f(w) + r\sum_{j=1}^{m}\frac{1}{c_j(w)}\ . \qquad (6.2)$$

Якщо w приймає допустимі значення, тобто значення, для яких $c_j(w) > 0$, то z приймає значення, що більше відповідних значень f(w) (істинної цільової функції даної задачі), і різницю можна зменшити за рахунок того, що r може бути дуже малою величиною.

Але якщо w приймає значення, які хоча і є припустимими, але близькі до границі області обмежень, і принаймні одна з функцій

$c_j(w)$ близька до нуля, тоді значення функції p(w) і, отже, значення функції z стануть дуже великими. Таким чином, вплив функції p(w) полягає в створенні "гребеня з крутими краями" уздовж кожної границі області обмежень. Отже, якщо пошук починається з припустимої точки і здійснюється пошук мінімуму функції $\varphi(w, r)$ без обмежень, то мінімум, звичайно, буде досягатися усередині припустимої області для задачі з обмеженнями. Припускаючи r досить малою величиною для того, щоб вплив p(w) був малим у точці мінімуму, ми можемо зробити точку мінімуму функції $\varphi(w, r)$ без обмежень такою, що збігається з точкою мінімуму функції f(w) з обмеженнями.

У загальному випадку неможливо аналітично визначити положення мінімуму функції $\varphi(w, r)$, розглядаючи її як звичайну функцію від r. Для його визначення необхідно звернутися до числових методів.

Слід зазначити, що якщо цільова функція f(w) опукла, а функція $c_j(w)$ ввігнута, то функція $\varphi(w, r)$, задана рівнянням (6.2), також є опуклою функцією в області обмежень, що сама є опуклою. Отже, $\varphi(w,r)$ має для даного значення r єдиний мінімум.

Якщо w_1 та w_2 - точки, що належать допустимій області, тобто $c_j(w_1) \geq 0$ та $c_j(w_2) \geq 0$ для j = 1, 2, .. ., m, то при $0 < \theta < 1$ справедлива нерівність

$$c_j\left(\theta w_2 + (1-\theta)w_1\right) \geq \theta c_j(w_2) + (1-\theta)c_j(w_1) \geq 0 ,$$

тому що функція $c_j(w)$ опукла. Отже, припустима область є опуклою.

Таким чином, точка $w_2+(1-\theta)w_1$ при $0 < \theta < 1$ також є припустимою.

Крім того, функція $1/c_j(w)$ є опуклою для всіх w, що задовольняють нерівності $c_j(w) \geq 0$. Якщо $h(w) = 1/c_j(w)$, то

$$\nabla h(w) = \frac{-\nabla c_j(w)}{\left[c_j(w)\right]^2} .$$

Отже, гессіан функції h(w) має вигляд:

$$H(w) = -\frac{C(w)}{[c_j(w)]^2} + \frac{2\nabla c(w)\nabla c(w)^T}{[c_j(w)]^3},$$

де $C(w)_{ik} = \partial^2 c_j(w) / \partial w_i \partial w_k$ є гессіан функції $c_j(w)$. Тоді, якщо p - довільний вектор, то справедлива рівність:

$$p^T H(w) p = -\frac{p^T C(w) p}{[c_j(w)]^2} + \frac{2[p^T \nabla c_j(w)]^2}{[c_j(w)]^3},$$

де завжди $p^T H(w) p > 0$, тому що c(w) - негативно визначена матриця через те, що $c_j(w)$ - опукла функція і $c_j(w) \geq 0$. Тоді матриця H(w) позитивно визначена і $1/c_j(w)$ опукла у всій області.

Якщо $r > 0$, то функція p(w), задана рівнянням (6.1), і функція $\varphi(w,r)$, задана рівнянням (6.2), також опукла.

Припустимо, що $w_1^*, w_2^*, ..., w_k^*$ - мінімальні точки функції $\varphi(w,r)$ для спадаючої послідовності значень $r_1, r_2, ..., r_k, ...$, яка прагне до нуля. Тоді послідовність точок $w_1^*, w_2^*, ..., w_k^*$ збігається до оптимального вирішення задачі з обмеженнями: мінімізувати функцію $z = f(w)$ при обмеженнях $c_j(w) > 0$, $j = 1, 2, ..., m$ і $r_k \to 0$

Отже, $\lim w_k^* = w^*$ та $\lim_{r_k \to 0}[\min \varphi(w, r_k)] = f(w^*)$,

де w^* - мінімальна точка функції f(w) при наявності обмежень.

6.2 Метод SUMT Фіакко-Маккорміка

Результати попереднього розділу показують, що можна вирішити задачу мінімізації з обмеженнями (мінімізувати функцію f(w) при обмеженнях $c_j(w) \geq 0$), вирішуючи для послідовності значень r, що прагне до нуля, задачі без обмежень наступного вигляду: мінімізувати функцію $\varphi(w,r) = f(w) + r\sum_{j=1}^{m} \frac{1}{c_j(w)}$.

Метод SUMT (sequential unconstrained minimisation technique) був уперше запропонований Керролом у 1961 році. Його ідеї були досліджені Фіакко і Маккорміком, що не тільки розглянули

теоретичні питання і збіжність методу, але й створили практичну систему для його реалізації.

Для того, щоб можна було застосувати дійсний метод на практиці, необхідно побудувати обчислювальний метод, що використовує теоретичну властивість збіжності, розглянутої у попередньому розділі. Теоретично тут не виникає труднощів. Для заданих функцією f(w) обмеженнях $c_j(w) \geq 0$, j = 1, ..., m, необхідно вибрати початкове значення $r = r_0$, щоб сформувати функцію $\varphi(w, r_0)$, що мінімізується без обмежень методом Давідона-Флетчера-Пауелла. Знайшовши мінімум функції $\varphi(w, r_0)$, необхідно зменшити значення r. Це можна зробити ефективно і просто, якщо знайти $r_1 = r_0/c$, де константа $c>1$.

Потім необхідно мінімізувати функцію $\varphi(w, r_1)$, знову використовуючи метод Давідона-Флетчера-Пауелла. Таким чином, буде розроблена ітераційна процедура. На k-му кроці мінімізується функція $\varphi(w, r_k)$, мінімум якої знаходиться в точці w_k^*. Важливо, що її можна використовувати надалі як першу точку в ітераційній процедурі мінімізації функції $\varphi(w, r_{k-1})$, де $r_{k+1} = r_k/c$. Тепер ясно, що послідовність r_k спадає і прагне до нуля, отже, послідовність точок мінімумів буде сходитися до рішення задачі з обмеженнями.

Нижче приведена блок-схема (рис. 6.1) методу SUMT. Передбачається, що на початку процедури існує припустима точка. Важливо, щоб у процесі наступних обчислень точки, що одержуються належали припустимій області. Метод Давідона-Флетчера-Пауелла є градієнтним методом мінімізації, що використовує при одновимірному пошуку кубічну інтерполяцію. Тоді, у міру наближення точки w до границі усередині припустимої області $\varphi(w, r) \to \infty$, а в міру наближення точки w до границі зовні припустимої області $\varphi(w, r) \to -\infty$.

Таким чином, якщо пошук здійснюється уздовж прямої, що з'єднує дві точки, одна з яких лежить усередині, а інша поза областю обмежень, то кубічна інтерполяція виявляється неприйнятною, оскільки функція розривна уздовж даної прямої.

Дійсно, якщо мінімум буде знайдений поза припустимою областю, то метод Давідона-Флетчера-Пауелла не дозволить знову ввійти в область обмежень. Необхідно ретельно досліджувати такі питання при використанні методу Давідона-Флетчера-Пауелла в даній задачі.

Вибір початкового значення r може виявитися важливим з погляду скорочення числа ітерацій при мінімізації функції $\varphi(w,r)$.

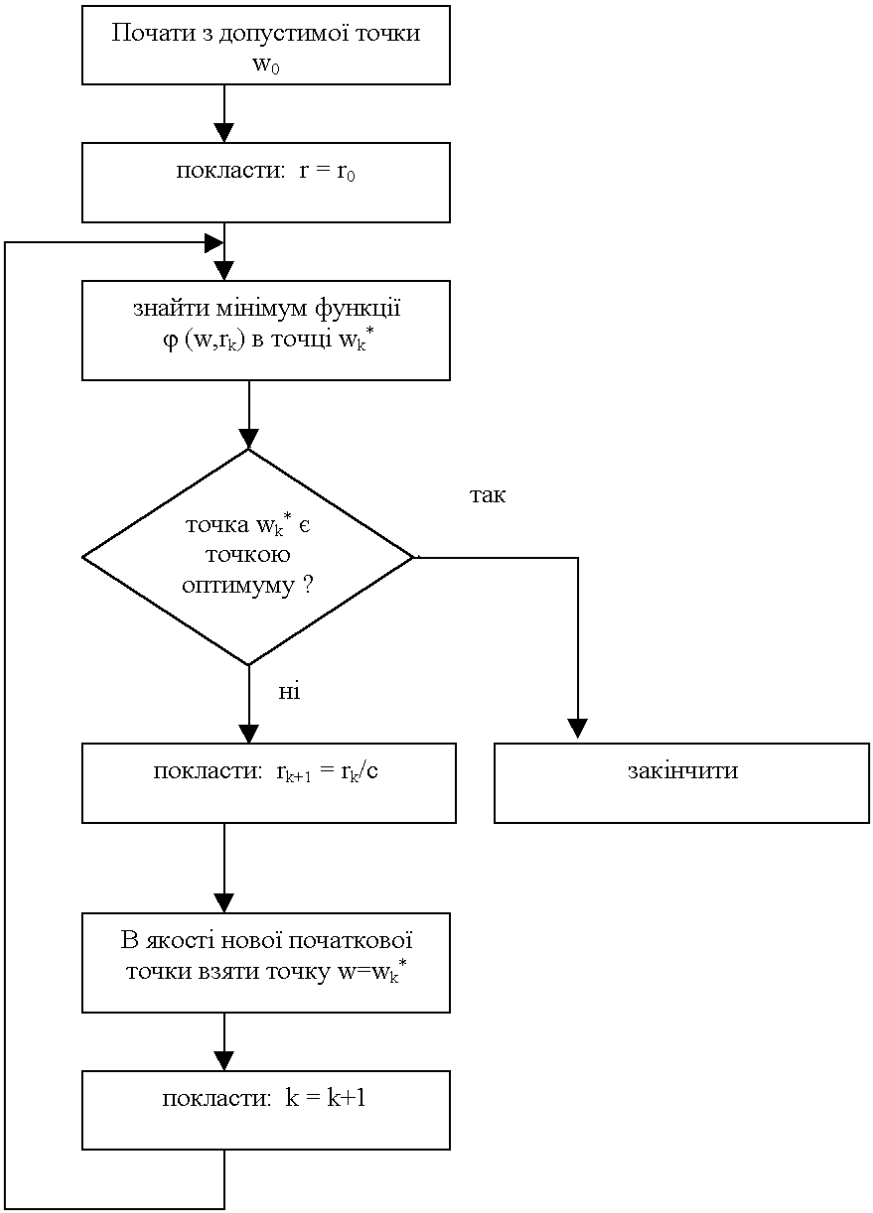


Рис 6.1 - Схема методу SUMT

Якщо спочатку r обрано дуже малим, для того щоб функція $\varphi(w, r)$ мало відрізнялася від функції f(w), то метод буде сходитися дуже швидко. Однак такий вибір може привести до серйозних ускладнень при обчисленнях. Для малих r функція $\varphi(w, r)$ буде швидко змінюватися в околиці мінімуму, що може викликати утруднення при використанні градієнтного методу. Занадто ж велике значення r може привести до того, що штрафна функція p(w) у рівнянні z=f(w)+p(w) стане домінуючою. Тому "розумний" вибір початкової точки дуже важливий.

Для багатьох задач "розумним" значенням для початкової точки є значення $r_0 = 1$. Більш раціональний підхід полягає в тому, щоб зрозуміти, що якщо початкова точка w буде лежати поблизу мінімуму функції

$$\varphi(w,r) = f(w) + r\sum_{j=1}^{m}\frac{1}{c_j(w)} = f(w) + p(w),$$

то градієнт функції $\varphi(w, r)$ буде малий:

$$\nabla\varphi(w,r) = \nabla f(w) + r\nabla p(w).$$

Квадрат норми цього вектора

$$\nabla f(w)^T\nabla f(w) + 2r\nabla f(w)^T\nabla p(w) + r^2\nabla p(w)^T\nabla p(w)$$

та мінімум буде досягнутий при $r = \dfrac{-\nabla f(w)^T\nabla p(w)}{\nabla p(w)^T\nabla p(w)}$.

Це початкове значення r, як припускають Фіакко і Маккормік, повинно давати гарні результати в загальному випадку. Зменшити значення r дуже просто: $r_{k+1} = r_k/c$, де c = 10. Для мінімізації функції $\varphi(w, r_{k+1})$ використовується метод Давідона-Флетчера-Пауелла. Як початкову точку використовують оптимальну точку функції $\varphi(w, r_k)$, і це виявляється дуже ефективним.

РОЗДІЛ 7. МЕТОДИ ОПТИМІЗАЦІЇ В ЗАДАЧАХ АПРОКСИМАЦІЇ І НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ

При побудові адаптивних систем діагностики і прогнозування перспективним є використання **нейронних мереж** (НМ), що мають такі властивості, як здатність до навчання, універсальність і здатність апроксимувати будь-які обчислювані функції. Це дозволяє використовувати їх для класифікації, оцінки значень параметрів і побудови математичних моделей складних процесів і об'єктів навіть у тих випадках, коли іншими способами це зробити складно.

Крім перерахованих вище переваг НМ характеризуються високою надійністю і стійкістю до негативних зовнішніх впливів (високими робастними властивостями), а також мають здібності самостійно вилучати знання з даних у процесі навчання.

7.1 Формальний нейрон. Одношаровий персептрон

Нервова система людини складається з кліток, названих **нейронами**, і має приголомшуючу складність: близько 10^{11} нейронів беруть участь у близько 10^{15} передавальних зв'язках, що мають довжину метр і більше. Кожний нейрон має багато якостей, спільних з іншими клітинами, але його унікальною здатністю є прийом, обробка і передача електрохімічних сигналів по нервових шляхах, що утворюють комунікаційну систему мозку.

На рис. 7.1 показана структура **біологічного нейрона**. Дендрити йдуть від тіла нервової клітки до інших нейронів, де вони приймають сигнали в точках з'єднання, названих **синапсами**.

Прийняті синапсом вхідні сигнали підводяться до тіла нейрона. Тут вони сумуються, причому одні входи прагнуть збудити нейрон, інші – перешкодити його збудженню. Коли сумарне збудження в тілі нейрона перевищує деякий **поріг**, нейрон збуджується, посилаючи по аксону сигнал іншим нейронам. У цієї основної функціональної схеми багато ускладнень і виключень, проте більшість штучних нейронних мереж моделюють лише ці прості властивості.

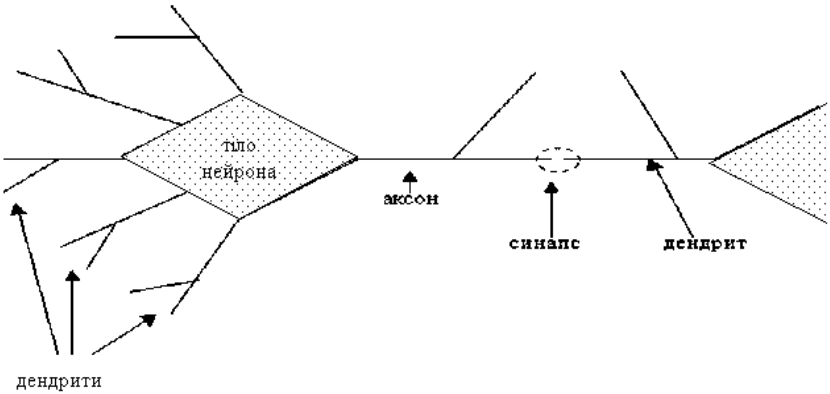


Рис. 7.1 - Структура біологічного нейрона

Отже, **нейрон (формальний нейрон, нейроподібний елемент)**, що являє собою примітивний обчислювальний пристрій, який має кілька входів і один вихід, є основним обчислювальним елементом НМ.

Одношаровий персептрон є одним з найпростіших варіантів НМ (рис. 7.2) і містить всього один нейрон.

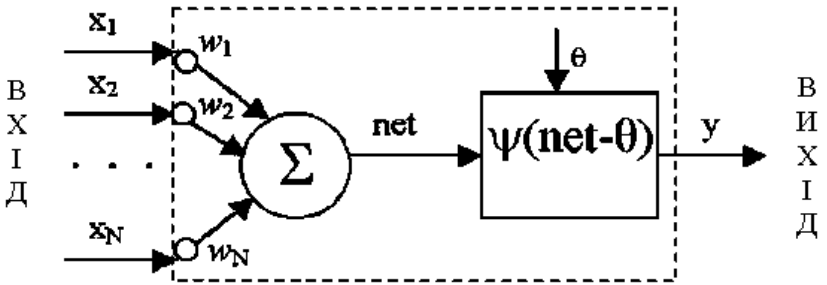


Рис. 7.2 - Одношаровий персептрон (формальний нейрон)

На вхід нейроподібного елемента (формального нейрона) надходить набір вхідних сигналів x_1, x_2, ..., x_N або вхідний вектор x. Кожний вхідний сигнал множиться на відповідну **вагу** зв'язку w_1, w_2, ..., w_N - аналог ефективності синапсу (міжнейронного контакту).

Вага зв'язку є скалярною величиною, позитивною для збуджуючих і негативною для гальмуючих зв'язків. Зважені вагами зв'язків вхідні сигнали надходять на блок сумації, що відповідає тілу клітки, де здійснюється їх алгебраїчна сумація і визначається рівень збудження нейроподібного елемента net:

$$net = \sum_{i=1}^{N} w_i x_i .$$

Вихідний сигнал нейрона y визначається шляхом пропущення рівня збудження net через нелінійну **функцію активації** ψ:

$$y = \psi(net-\theta),$$

де θ - деякий постійний зсув (аналог порога нейрона).

Звичайно використовуються найпростіші нелінійні функції: бінарна (порогова):

$$\psi(x) = \begin{cases} 1, \text{ при } x > 0, \\ 0, \text{ при } x \le 0, \end{cases}$$

або сигмоїдальна: $\psi(x) = 1/(1+e^{-x})$.

У залежності від типу функції активації розрізняють дискретні персептрони, що використовують порогову функцію активації, і реальні – що використовують реальні функції активації, наприклад сигмоїдальну функцію.

Кожний нейрон має невелику пам'ять, реалізовану ваговими коефіцієнтами вхідних синапсів (міжнейронних контактів) і порогом нейрона. Тому нейрони можна розглядати як запам'ятовуючі пристрої. У той же час нейрони можуть розглядатися як примітивні процесори, що здійснюють обчислення значення функції активації на основі різниці зваженої суми вхідних сигналів і порогу.

Алгоритм навчання одношарового дискретного персептрона має вигляд.

Крок 1. Вагам $w_i(0)$ (i=1, ... ,N) і порогові $\theta(0)$ привласнюються випадкові значення (через $w_i(t)$ позначений ваговий коефіцієнт і-го

входу персептрона в момент часу t, через $\theta(t)$ позначена величина зсуву (порога) нейрона в момент часу t).

Крок 2. Пред'являється черговий вхідний вектор $x=\{x_1,...,x_N\}^T$ з навчальної множини і бажаний вихід $y^*(t)$ ($y^*(t) = 1$, якщо $x(t)$ відноситься до класу A, $y^*(t) = 0$, якщо $x(t)$ відноситься до класу B).

Крок 3. Обчислюється реальне значення на виході персептрона за формулами:

$$net = \sum_{i=1}^{N} w_i(t)x_i(t),$$

$$y(t) = \psi(net - \theta(t)).$$

Крок 4. Корегуються ваги відповідно до рівностей:

$$w_i(t+1) = w_i(t) + \eta(y^*(t) - y(t))x_i(t),\ i = 1,2,...,N,$$

$$\theta(t+1) = \theta(t) + \eta(y^*(t) - y(t)),$$

де η - позитивний коригувальний приріст.

Крок 5. Якщо досягнута збіжність, то процедура навчання закінчується; у протилежному випадку - перехід до кроку 2.

Відповідно до даного алгоритму спочатку відбувається ініціалізація параметрів персептрона випадковими значеннями. Потім по черзі пред'являються образи з відомою класифікацією, обрані з навчальної множини, і корегуються ваги відповідно до формул кроків 3 і 4. Величина корекції визначається позитивним коригувальним приростом η, конкретне значення якого вибирається досить великим, щоб швидше здійснювалася корекція ваг, і в той же час досить малим, щоб не допустити надмірного зростання значень ваг.

Процедура навчання продовжується доти, поки не буде досягнута збіжність, тобто поки не будуть отримані ваги, що забезпечують правильну класифікацію для всіх образів з навчальної множини.

У тому випадку, коли навчальні вибірки розділити гіперплощиною неможливо, для навчання персептрона можна використовувати **алгоритм Уідроу-Хоффа**, що мінімізує

середньоквадратичну помилку між бажаними і реальними виходами мережі для навчальних даних. Цей алгоритм також можна застосовувати для навчання одношарового реального персептрона. Алгоритм Уідроу-Хоффа можна записати в тому ж вигляді, що і вищеописаний алгоритм, припускаючи, що у вузлах персептрона нелінійні елементи відсутні, а коригувальне збільшення η у процесі ітерацій поступово зменшується до нуля.

Якщо для вирішення задачі розпізнавання образів використовується дискретний персептрон, рішаюче правило відносить вхідний образ до класу А, якщо вихід персептрона дорівнює 1, і до класу В – у протилежному випадку.

У випадку, якщо для вирішення задач розпізнавання образів використовується реальний персептрон, рішаюче правило відносить вхідний образ до класу А, якщо вихід мережі більше 0.5, і до класу В в протилежному випадку.

Усі моделі НМ представляють собою множину нейронів, зв'язаних певним способом між собою. Основними відмінностями моделей НМ є способи зв'язку нейронів між собою, механізми і напрямки поширення сигналів по мережі, а також обмеження на функції активації, що використовуються.

Одна з найважливіших властивостей НМ - здатність до самоорганізації і самоадаптації з метою поліпшення якості функціонування. Це досягається навчанням НМ, алгоритм якого задається набором навчальних правил. Навчальні правила визначають, яким чином змінюються зв'язки у відповідь на вхідний вплив. **Навчання** засноване на збільшенні сили зв'язку (ваги синапсу) між одночасно активними нейронами. Таким чином, зв'язки, що часто використовуються, підсилюються, що пояснює феномен навчання шляхом повторення і звикання.

На цей час не існує єдиної стандартної класифікації НМ, оскільки нейроінформатика є новою областю науки і термінологія тут ще не встановилася. Тому розглянемо класифікацію НМ тільки за деякими базовим характеристикам.

У залежності від типу функції активації НМ підрозділяють на дискретні, реальні (безперервні) і дискретно-безперервні.

У залежності від напрямку поширення сигналів НМ підрозділяють на мережі прямого поширення, мережі зворотного поширення і двонаправлені НМ.

У залежності від кількості і структури зв'язків НМ підрозділяють на повнозв‘язні (усі нейрони зв'язані з усіма) і неповнозв‘язні.

У залежності від кількості шарів нейронів НМ підрозділяють на одношарові і багатошарові. Іноді особливо виділяють двошарові і тришарові НМ.

7.2 Багатошаровий персептрон

Основним обчислювальним елементом **багатошарового персептрона** або **багатошарової нейронної мережі** (БНМ) є формальний нейрон. Він виконує параметричне нелінійне перетворення вхідного вектора x в скалярну величину y. Нейрони утворюють мережу, що характеризується наступними параметрами і властивостями: M - число шарів мережі, N_μ - число нейронів μ-го шару, при чому зв'язки між нейронами в шарі відсутні.

Виходи нейронів μ-го шару, μ = 1,2,...,M-1 надходять на входи нейронів тільки наступного μ+1-го шару. Зовнішній векторний сигнал x надходить на входи нейронів тільки першого шару, виходи нейронів останнього M-го шару утворюють вектор виходів мережі $y^{(M)}$.

Структура мережі показана на рис. 7.3.

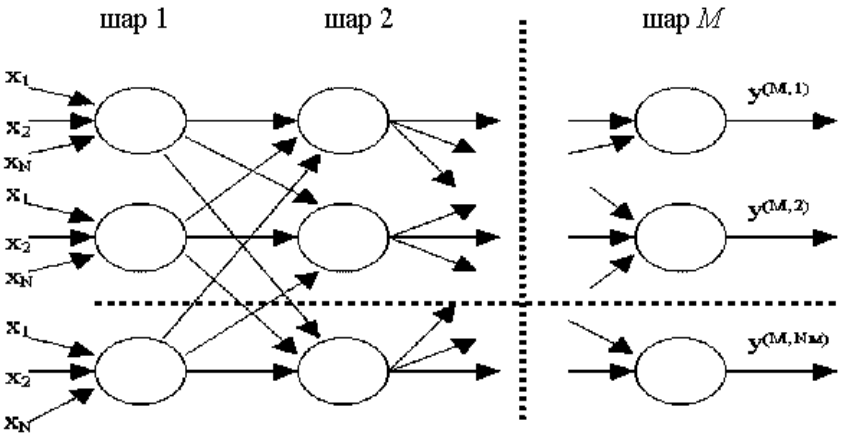


Рис.7.3 - Структура багатошарової нейронної мережі

Кожний і-й нейрон μ-го шару (μі-й нейрон) перетворює вхідний вектор $x^{(\mu,i)}$ у вихідну скалярну величину $y^{(\mu,i)}$. Це перетворення складається з двох етапів: спочатку обчислюється дискримінантна функція $net^{(\mu,i)}$, що далі перетворюється у вихідну величину $y^{(\mu,i)}$.

Дискримінантна функція являє собою відрізок багатовимірного ряду Тейлора. Коефіцієнти розкладання відрізка багатовимірного ряду Тейлора утворюють вектор вагових коефіцієнтів $w^{(\mu,i)}$, або пам'ять нейрона. Дискримінантна функція нейрона має вигляд:

$$net^{(\mu,i)} = w_0^{(\mu,i)} + \sum_{j=1}^{N} w_j^{(\mu,i)} x_j^{(\mu,i)} \ ,$$

де $w^{(\mu,i)}=(w_0^{(\mu,i)}, w_1^{(\mu,i)}, \ldots , w_N^{(\mu,i)})^T$ - вектор вагових коефіцієнтів нейрона; $x_j^{(\mu,i)}$ - j-а компонента N-вимірного вхідного вектора $x^{(\mu,i)}$.

Нелінійне перетворення $y^{(\mu,i)} = \psi(net^{(\mu,i)})$ задається функцією активації, що є монотонною й обмеженою. Зокрема, при негативних виходах нейрона такою функцією може бути сигмоїдальна функція $\psi(x) = 1/(1+e^{-x})$.

Позначимо через $y^{(\mu)} = (y^{(\mu,1)}, y^{(\mu,2)}, \ldots , y^{(\mu,N\mu)})^T$ вектор виходу нейронів μ-го шару.

Процес навчання мережі здійснюється в результаті мінімізації цільової функції - певного критерію якості F(w), що характеризує інтегральну міру близькості виходів мережі $y^{(M)}(k)$ і вказівок учителя y*(k):

$$F(w)=\frac{1}{k}\sum_{m=1}^{k} Q(\varepsilon(w,m)) \ ,$$

де k – номер поточного циклу навчання НМ; m=1, 2,..., k – номера попередніх циклів навчання НМ; w - складений вектор-стовпець вагових коефіцієнтів мережі, що складається з векторів-стовпців $w^{(\mu)} =(w^{(\mu,1)T}, w^{(\mu,2)T}, \ldots , w_N^{(N\mu)T})^T$, μ=M,M-1,...,1 кожного шару.

Для навчання БНМ на практиці використовують, як правило, градієнтні алгоритми.

7.3 Порівняльна характеристика методів оптимізації при навчанні багатошарових нейронних мереж

Інтерес для дослідження становить, який окремий випадок узагальненого градієнтного алгоритму (тобто який градієнтний алгоритм навчання БНМ) є кращим.

З аналізу розглянутих градієнтних алгоритмів навчання БНМ випливає, що їхня збіжність залежить, крім навчальних даних і початкових ваг, ще й від параметрів, що задаються: максимальної припустимої загальної помилки навчання (мета навчання) і кількості припустимих циклів навчання (критерій швидкості навчання) для усієї вибірки. Ці два критерії доцільно використовувати для **порівняння методів навчання** БНМ. Зафіксувавши по черзі кожний з них, можна досліджувати, як для даного алгоритму навчання змінюється значення іншого критерію при однакових наборах різних навчальних вибірок і однакових початкових вагах.

Результати експериментів варто оцінювати за такими критеріями, як час роботи алгоритму навчання і витрачена кількість циклів навчання.

Проводилися експерименти по навчанню тришарових персептронів діагностиці (класифікації на придатні і непридатні) виробів (у різних серіях експериментів варіювалося також число входів НМ).

У якості вихідних даних для навчання НМ використовувалися як навчальні вибірки реальних виробів, так і вибірки з псевдовипадкових чисел.

Фрагменти результатів експериментів приведені в таблицях 7.1 і 7.2, а також на рис. 7.4.

Як видно з рис. 7.4 і табл. 7.1 та 7.2, найкращим серед усіх методів навчання БНМ виявився алгоритм Левенберга-Марквардта. Він є найбільш швидким серед всіх алгоритмів і менш складним з обчислювальної точки зору, ніж метод Ньютона.

Метод Ньютона є найбільш складним з обчислювальної точки зору і найвимогливішим до умов застосування, але, разом з тим, він перевершує алгоритми спряжених градієнтів як по швидкості навчання, так і по точності.

Алгоритми спряжених градієнтів Полака-Ріб’єра і Флетчера-Рівса є конкуруючими й поступаються як по швидкості, так і по точності методу Ньютона й алгоритму Левенберга-Марквардта.

Таблиця 7.1 - Порівняльна характеристика алгоритмів навчання БНМ при фіксованій максимальній точності

Алгоритм	Фіксована максимальна точність (помилка)					
	10^{-1}		10^{-3}		10^{-6}	
	Час навчання, с	Число циклів навчання	Час навчання, с	Число циклів навчання	Час навчання, с	Число циклів навчання
Ньютона	2.03	3	8.68	26	9.78	45
Флетчера-Рівса	1.87	2	10.55	40	27.9	172
Полака-Ріб'єра	1.87	2	10.93	41	26.04	157
Левенберга – Марквардта	1.91	3	5.01	13	5.91	14

Таблиця 7.2 - Порівняльна характеристика алгоритмів навчання БНМ при фіксованому максимальному числі циклів навчання

Алгоритм	Фісоване максимальне число циклів навчання					
	10			100		
	Час навчання, с	Досягнута точність	Затрачене число циклів навчання	Час навчання, с	Досягнута точність	Затрачене число циклів навчання
Ньютона	3.29	0.007	10	20.49	$7.81 \cdot 10^{-8}$	100
Флетчера-Рівса	3.3	0.01	10	17.63	$1.11 \cdot 10^{-4}$	100
Полака-Ріб'єра	2.9	0.01	10	17.31	$3.79 \cdot 10^{-5}$	100
Левенберга – Марквардта	2.63	$3.01 \cdot 10^{-8}$	10	4.66	$2.25 \cdot 10^{-11}$	16

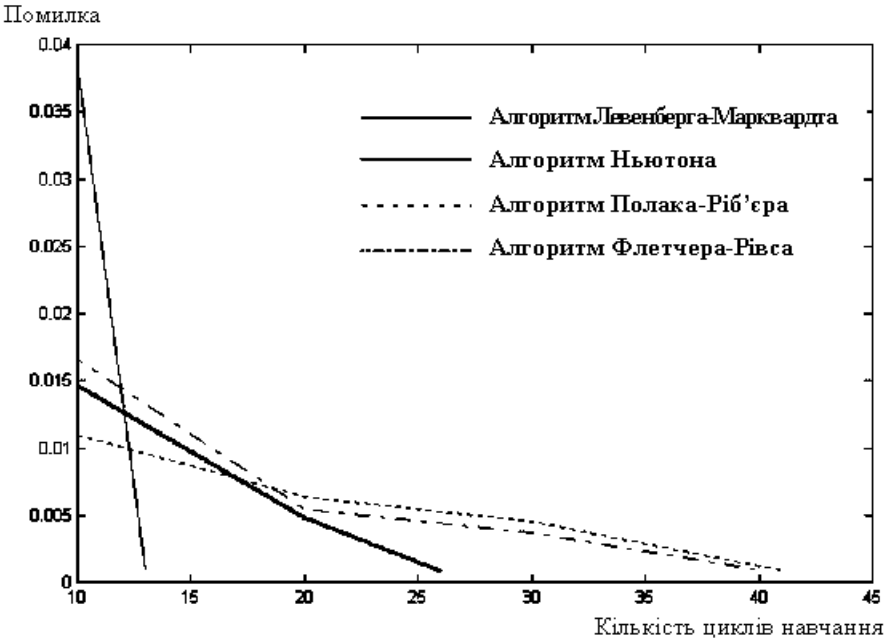


Рис. 7.4 - Графік збіжності алгоритмів
навчання багатошарових нейронних мереж

Результати експериментів свідчать про працездатність узагальненого градієнтного алгоритму, про його великі адаптивні можливості і дозволяють рекомендувати узагальнений градієнтний алгоритм навчання БНМ для впровадження і використання на практиці.

7.4 Спостерігаючий алгоритм навчання

У тому випадку, коли цільова функція навчання БНМ є досить гладкою, не містить складних вигинів і великої кількості локальних мінімумів, традиційно застосовувані градієнтні алгоритми навчання НМ дозволяють досягти мети навчання, забезпечуючи досить швидку збіжність.

Коли ж цільова функція є істотно нелінійною і містить велику кількість локальних мінімумів, а обсяг навчальних даних великий, градієнтні алгоритми забезпечують швидку збіжність тільки на початку навчання, після чого в багатьох випадках збіжність навчання істотно зменшується, що можна пояснити влученням мережі в локальні мінімуми і недостатньо великим коригувальним приростом, щоб швидко вибратися з локального мінімуму.

Тому необхідно використовувати алгоритм, що дозволяє прискорити збіжність градієнтних алгоритмів навчання БНМ у випадку влучення мережі в локальні мінімуми.

Для прискорення роботи градієнтних алгоритмів при влученні НМ у локальний мінімум необхідно вирішити дві задачі:

1) встановити, що мережа потрапила в локальний мінімум і швидкість збіжності є низькою;

2) вивести мережу з локального мінімуму.

Для вирішення першої задачі пропонується спостерігати за процесом навчання БНМ, запам'ятовуючи значення цільової функції на кожному циклі навчання мережі, і, у випадку, якщо протягом заданої кількості циклів цільова функція зменшилася менш ніж на задану величину, виконувати процедуру виводу мережі з локального мінімуму.

Вивід мережі з локального мінімуму пропонується здійснювати, змінюючи значення ваг і порогів мережі на деякі відносно невеликі величини таким чином, щоб мережа зберегла весь попередній позитивний досвід і в той же час могла змінити своє положення в багатовимірному просторі ваг і порогів.

Зазначимо, що корекція ваг мережі навіть на досить малу величину може приводити до істотних змін, причому, не тільки позитивних, але і до негативних, коли спостерігаючий алгоритм погіршує збіжність, досягнуту градієнтним алгоритмом. Тому результати змін ваг БНМ на основі спостерігаючого алгоритму повинні прийматися тільки в тому випадку, коли вони поліпшують збіжність, у протилежному випадку повинні відновлюватися значення ваг, отримані після застосування коригувального правила градієнтного алгоритму.

Узагальнюючи вищесказане, запишемо **спостерігаючий алгоритм.**

Крок 1. Ініціалізація параметрів градієнтного алгоритму навчання БНМ. Ініціалізація параметрів спостерігаючого алгоритму: кроку α, розміру вікна спостереження Δt (у циклах), критерію доцільності застосування алгоритму спостереження на даному етапі ξ та покажчика комірки вікна спостереження pt: pt =1. Резервування пам'яті для вікна спостереження Err(Δt), тут Err(Δt) – масив Δt елементів.

Крок 2. Установити лічильник циклів навчання epoch=0.

Крок 3. Якщо epoch>Epochs, де Epochs – задана максимальна припустима кількість циклів навчання НМ, тоді перейти на крок 11, у протилежному випадку – перейти на крок 4.

Крок 4. Обчислити значення perf - цільової функції навчання БНМ.

Крок 5. Якщо номер комірки пам'яті $p_t \le \Delta t$, тоді прийняти $Err(p_t)$ = perf, $p_t = p_t+1$; у протилежному випадку, прийняти $p_t=1$, $Err(p_t)$ = perf.

Крок 6. Перевірити критерій зупинення для градієнтного алгоритму. Якщо градієнтний алгоритм повинен припинити роботу, то перейти до кроку 11, у протилежному випадку - виконати корекцію ваг для даного циклу навчання на основі градієнтного алгоритму.

Крок 7. Якщо epoch> Δt та $|perf - Err(mod(epoch, \Delta t) + 1)| / \Delta t < \xi$, де mod(a,b) – залишок від цілочислового розподілу a на b, тоді перейти на крок 8, у протилежному випадку – перейти на крок 10.

Крок 8. Прийняти: $w^*=R(w)$, де w та w^* - набори значень ваг і порогів БНМ, відповідно, до і після застосування коригувального правила ваг НМ спостерігаючого алгоритму R(w), і обчислити значення цільової функції $perf^*$ після застосування правила R(w).

Крок 9. Якщо $perf^* <$ perf, де perf – значення цільової функції до застосування коригувального правила ваг спостерігаючого алгоритму, тоді установити: $w = w^*$.

Крок 10. Установити: epoch=epoch+1. Перейти на крок 3.

Крок 11. Зупинення.

У якості правила R(w) пропонується використовувати наступні вирази:

$$R(w) = \alpha w, \qquad (7.1)$$

$$R(w) = w + \alpha w; \tag{7.2}$$

$$R(w) = w + \alpha \, rand, \tag{7.3}$$

де rand – випадкове число в діапазоні [0,1].

Для порівняння розробленого спостерігаючого алгоритму, і найбільш швидкого градієнтного алгоритму Левенберга-Марквардта на основі БНМ зважувалися задачі класифікації об'єктів і прогнозування значень параметрів складних виробів за реальними даними. Результати навчання представлені на рис. 7.5 та рис. 7.6.

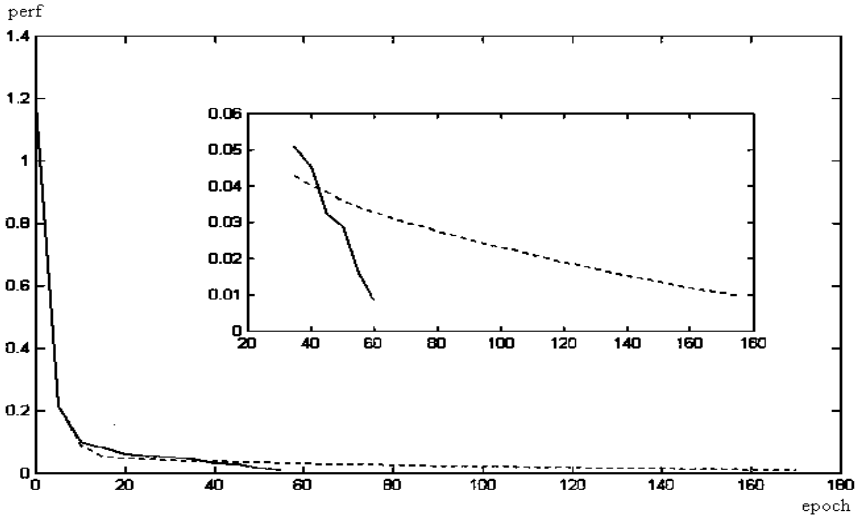
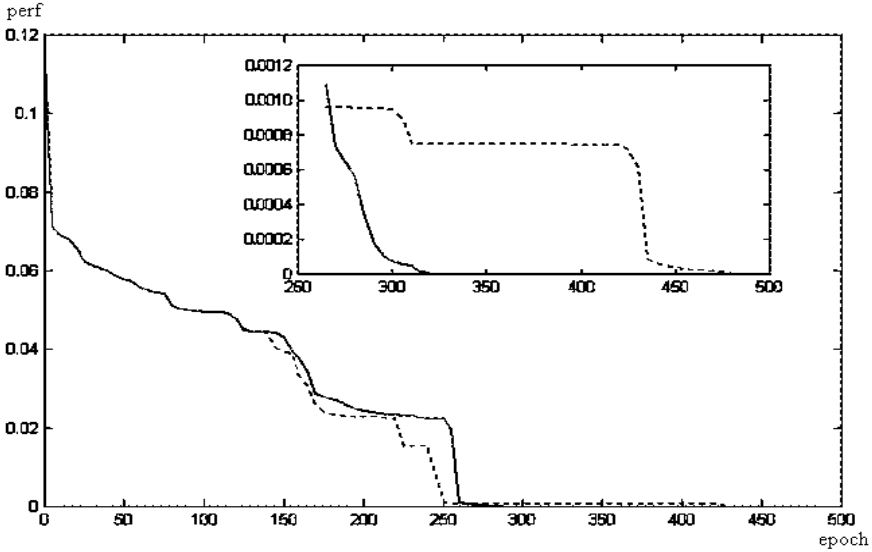


Рис. 7.5 - Графіки зменшення помилки perf при навчанні НМ класифікації (пунктирною лінією виділений графік помилки алгоритму Левенберга-Марквардта, суцільною лінією – спостерігаючого алгоритму)

Як видно з рис. 7.5 і рис.7.6, запропонований алгоритм дозволяє істотно скоротити час навчання БНМ і збільшити швидкість збіжності градієнтного алгоритму. У різних експериментах час навчання БНМ із використанням спостерігаючого алгоритму, скорочувалося на 20 - 60 %, що є дуже гарним результатом.

Аналізуючи графіки збіжності градієнтного і спостерігаючого алгоритмів можна зробити наступні висновки:

1) на початку обидва алгоритми забезпечують однакову збіжність, що пояснюється двома причинами: спостерігаючий алгоритм включається тільки після виконання Δt перших циклів



навчання і збіжність, що забезпечується градієнтним алгоритмом, на початку навчання досить висока і не вимагає включення спостерігаючого алгоритму;

Рис. 7.6 - Графіки зменшення помилки perf при навчанні НМ чисельної апроксимації функцій (пунктирною лінією виділений графік помилки алгоритму Левенберга-Марквардта, суцільною лінією –спостерігаючий алгоритм)

2) після проходження початкової фази навчання градієнтний алгоритм протягом деякого часу забезпечує більш швидку збіжність, ніж спостерігаючий алгоритм.

Це, очевидно, пояснюється тим, що навіть зміни, що зменшують значення цільової функції, внесені спостерігаючим алгоритмом можуть іноді негативно впливати на загальний процес збіжності, тобто заважають роботі градієнтного алгоритму на даному етапі.

3) на завершальній стадії навчання, коли градієнтний алгоритм навчання НМ починає блукати по локальних мінімумах цільової функції і забезпечує дуже повільну збіжність, спостерігаючий алгоритм забезпечує істотно більш швидку збіжність і перевищує за швидкістю градієнтний алгоритм.

Слід зазначити, що застосування спостерігаючого алгоритму для вирішення задач класифікації легко розділимих образів може не давати виграшу у швидкості навчання, оскільки через високу швидкість збіжності градієнтного алгоритму спостерігаючий алгоритм у роботу не буде вступати.

Експерименти по навчанню БНМ вирішенню практичних задач на основі спостерігаючого алгоритму дозволяють рекомендувати правила (7.1) і (7.2) для навчання НМ класифікації, а правило (7.3) – для навчання апроксимації. Застосування правил (7.1) і (7.2) для навчання апроксимації недоцільно. Це пояснюється тим, що дані правила здатні вносити досить сильні збурювання в набір ваг мережі.

З опису спостерігаючого алгоритму, випливає, що, крім вибору відповідного правила R(w), необхідно задавати значення визначених параметрів. Для того, щоб з'ясувати, які значення найбільш прийнятні для відповідних параметрів спостерігаючого алгоритму проводилися експерименти, фрагменти результатів яких представлені на рис. 7.7 та 7.8.

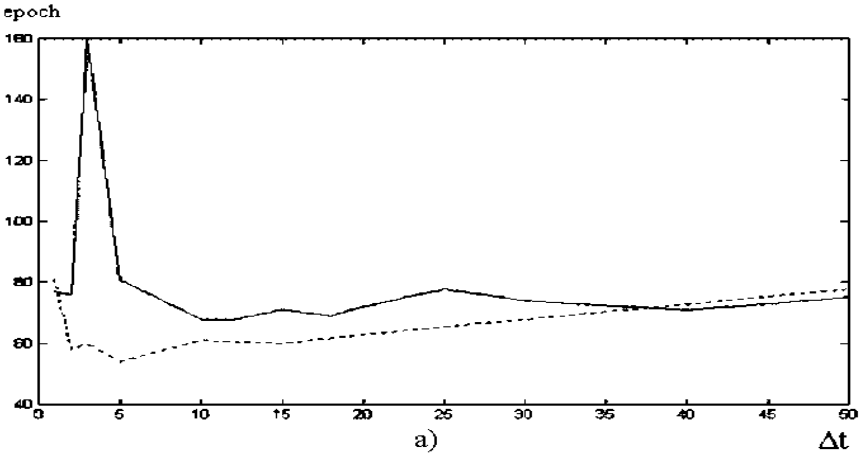
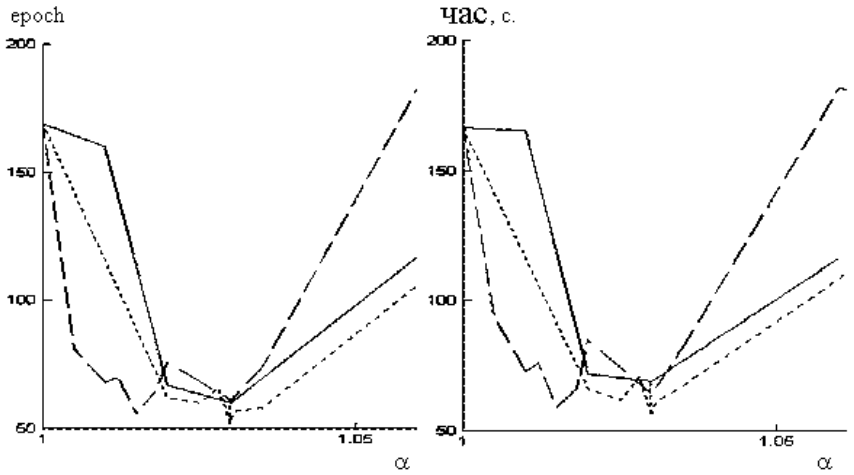


Рис. 7.7 - Графіки залежностей кількості циклів (a) і часу (b) навчання БНМ від параметра Δt для спостерігаючого алгоритму при фіксованому параметрі α (суцільною лінією виділені графіки при α=1.01, пунктирною лінією – при α=1.03)



a) b)

Рис.7.8 – Графіки залежностей кількості циклів (a) і часу (b) навчання БНМ від параметра α для спостерігаючого алгоритму при фіксованому параметрі Δt (суцільною лінією виділені графіки при Δt =3, штриховою - при Δt =5, пунктирною - при Δt = 10)

З рис. 7.7 та 7.8 легко побачити, що прийнятними для більшості випадків можуть бути наступні значення параметрів спостерігаючого алгоритму: α = 1.03, Δt = 6 - 20. Значення параметра ξ варто прийняти рівним 0.1.

Інтерес для дослідження становить також, як впливає розмір вікна спостереження Δt на кількість успішних виконань кроку 9 спостерігаючого алгоритму. Результати експериментів представлені на рис. 7.9

Як видно з рис. 7.9, зі зменшенням розміру вікна спостереження спостерігаючий алгоритм застосовується частіше, однак, як випливає з рис. 7.6, при цьому швидкість збіжності знижується.

З іншого боку, на інтервалі значень $\Delta t \in [6, 20]$ кількість успішних виконань кроку 9 спостерігаючого алгоритму є середнім і забезпечує досить швидку збіжність. Це може служити додатковим підтвердженням обґрунтованості вибору вищенаведених значень параметрів спостерігаючого алгоритму.

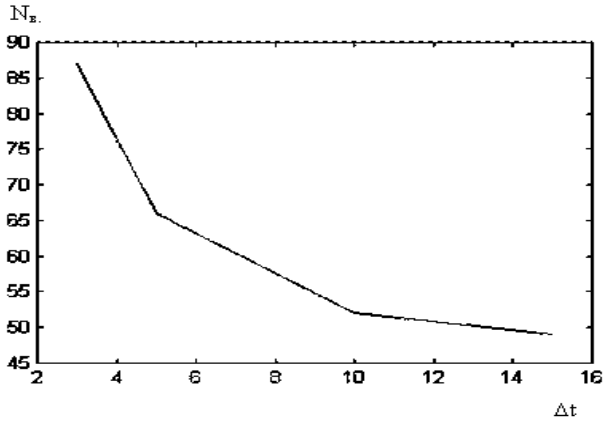


Рис.7.9 - Графік залежності кількості успішних виконань кроку 9 $N_{в.}$ спостерігаючого алгоритму від розміру вікна спостереження при фіксованому кроці α =1.01

7.5 Радіально-базисні нейронні мережі

Радіально-базисна НМ (РБНМ) складається з двох шарів. Сполучні вагові вектори шарів будемо позначати $w^{(\mu,j)}$, де μ-номер шару (μ=1,2), j – номер нейрона (вузла) у шарі. Базисні (чи ядерні) функції в першому шарі здійснюють локалізовану реакцію на вхідний стимул. Вихідні вузли мережі формують зважену лінійну комбінацію з базисних функцій, обчислених вузлами першого шару.

Вихідні вузли відповідають вихідним класам, у той час, як вузли першого шару представляють собою кластери (кількість кластерів m задається користувачем), на які розбивається вхідний простір. Позначимо $x = (x_1, ..., x_i, ..., x_N)$ і $y = (y_1, ..., y_i, ..., y_K)$ - вхід і вихід мережі, відповідно. Тут N – кількість ознак, а K-число класів.

Вихід u_j j-го вузла першого шару, використовуючи ядерну функцію Гауссіан як базисну, визначається за формулою:

$$u_j = \exp\left[-\frac{\left(x - w^{(1,j)}\right)^T\left(x - w^{(1,j)}\right)}{2\sigma_j^2}\right], \quad j=1,2,...,m,$$

де x - вхідний образ (екземпляр), $w^{(1,j)}$ - його вхідний ваговий вектор (тобто центр Гауссіана для вузла j) і σ^2_j - параметр нормалізації j-го вузла, такий що $0 < u_j < 1$ (чим ближче вхід до центру Гауссіана , тим сильніше реакція вузла).

Вихід y_j j-го вузла другого шару визначається з виразу:

$$y_j = w^{(2,j)T}u,\ j = 1, 2, ..., K,$$

де $w^{(2,j)}$ – ваговий вектор для j-го вузла другого шару і u - вектор виходів першого шару.

Мережа виконує лінійну комбінацію нелінійних базисних функцій. Задача навчання мережі полягає в мінімізації помилки

$$E = \frac{1}{2}\sum_{s=1}^{S}\sum_{j=1}^{K}\left(y_j^s - y_j^{s*}\right)^2 ,$$

де y^{s*}_j і y^s_j - бажане і розраховане значення виходу j-го вузла вихідного шару для s-го екземпляра, S - розмір набору даних (кількість екземплярів), і K - число вихідних вузлів (число класів). Далі для наочності верхній індекс s опущений.

Навчання РБНМ може виконуватися двома різними способами.

Перший спосіб полягає в тому, що алгоритмом кластеризації формується фіксована множина центрів кластерів. Потім мінімізацією квадратичної помилки, тобто мінімізацією E, одержують асоціації центрів кластерів з виходом.

Другий спосіб полягає в тому, що центри кластерів можуть бути також навчені поряд з вагами від першого шару до вихідного шару методом градієнтного спуску. Однак, навчання центрів поряд з вагами може привести до влучення мережі в локальні мінімуми.

Нехай фіксована множина центрів кластерів сформована на основі першого способу, а центри кластерів будуть позначені $w^{(1,j)}$, j = 1, ..., m. Параметр нормалізації σj являє міру розподілу даних, що асоційовані з кожним вузлом.

Навчання у вихідному шарі виконується після того, як визначені параметри базисних функцій.

Ваги звичайно навчають, використовуючи алгоритм середньоквадратичних відхилень:

$$\Delta w^{(\mu,j)} = -\eta e_j u \ ,$$

де $e_j = y_j - y^*_j$ та η - коефіцієнт швидкості навчання.

7.6 Методи оптимізації в задачах апроксимації

Якщо задана функція y(x), то це означає, що будь-якому припустимому значенню x співставлено значення y. Але нерідко виявляється, що знаходження цього значення дуже трудомістке. Наприклад, y(x) може бути визначене як вирішення складної задачі, у якій x відіграє роль параметра або y(x) виміряється в дорогому експерименті. При цьому можна обчислити невелику таблицю значень функції, але пряме знаходження функції при великому числі значень аргументу буде практично неможливим. Функція y(x) може брати участь у певних фізико-технічних або чисто математичних розрахунках, де її приходиться багаторазово обчислювати. У цьому випадку вигідно замінити функцію y(x) наближеною формулою, тобто підібрати деяку функцію $\varphi(x)$, що, у загальному випадку, не буде проходити через усі точки експериментальної вибірки, але буде до них досить близькою і просто обчислюватиметься. Потім при всіх значеннях аргументу приймають $y(x) \approx \varphi(x)$. Така заміна називається **апроксимацією** (від лат. approxima - наближення).

Якщо ж необхідно замінити y(x) функцією $\varphi(x)$, що проходить через усі точки експериментальної вибірки, то застосовують **інтерполяцію** (від лат. inter - між, polo - пригладжую).

Велика частина класичного числового аналізу ґрунтується на наближенні багаточленами, тому що з ними легко працювати. Однак також використовуються й інші класи функцій.

Вибравши вузлові точки і клас функцій, що наближають, необхідно також вибрати ще одну визначену функцію з цього класу за допомогою певного **критерію** - деякої міри наближення чи **"погодження"**. Перш ніж почати обчислення, потрібно вирішити, яку точність необхідно мати у відповіді і який критерій обрати для виміру цієї точності.

Існують 3 класи чи групи функцій, що широко застосовуються у числовому аналізі. Перша група містить у собі лінійні комбінації функцій $1, x, x^2, \ldots, x^n$, що збігається з класом усіх багаточленів

ступеня n (чи менше). Другий клас утворюють функції cos $a_i x$, sin $a_i x$. Цей клас має відношення до рядів Фур'є й інтегралу Фур'є. Третя група утворюється функціями e^{-az}.

Що стосується критерію погодження, то класичним критерієм погодження є “точний збіг у вузлових точках”. Цей критерій має перевагу простоти теорії і виконання обчислень, але разом з тим незручність через ігнорування шуму (погрішності, що виникає при вимірі чи обчисленні значень у вузлових точках). Інший відносно кращий критерій — це “найменші квадрати”. Він означає, що сума квадратів відхилень у вузлових точках повинна бути найменшою, іншими словами, мінімізованою. Цей критерій використовує інформацію про помилку, щоб одержати деяке згладжування шуму. Очевидно, можливі й інші критерії.

Нехай деяка функція y = y(x) задана n парами значень абсцис x_i і ординат y_i. **Задача лінійного регресійного аналізу** полягає в знаходженні такої лінії регресії, щоб сума квадратів відхилень уздовж осі ординат була мінімальною.

Для **рівняння регресії** $y = b_0\varphi_0(x) + b_1\varphi_1(x)$, де $\varphi_0(x)$ и $\varphi_1(x)$ - довільні функції, вимога мінімальності виразу $\sum_{i=1}^{n}[y_i - b_0\varphi_0(x_i) - b_1\varphi_1(x_i)]^2$ приводить до наступних значень для коефіцієнтів b_0 та b_1:

$$b_0 = \frac{\overline{y\varphi_0}\,\overline{\varphi_1^2} - \overline{y\varphi_1}\,\overline{\varphi_0\varphi_1}}{\overline{\varphi_0^2}\,\overline{\varphi_1^2} - \left(\overline{\varphi_0\varphi_1}\right)^2}\ ,\ b_1 = \frac{\overline{y\varphi_0^2}\,\overline{\varphi_1^2} - \overline{\varphi_0\varphi_1}\,\overline{y\varphi_0}}{\overline{\varphi_0^2}\,\overline{\varphi_1^2} - \left(\overline{\varphi_0\varphi_1}\right)^2},$$

де $\overline{y\varphi_0} = \frac{\sum_{i=1}^{n} y_i\varphi_0(x_i)}{n}$, $\overline{y\varphi_1} = \frac{\sum_{i=1}^{n} y_i\varphi_1(x_i)}{n}$, $\overline{\varphi_0\varphi_1} = \frac{\sum_{i=1}^{n} \varphi_0\varphi_1(x_i)}{n}$,

$$\overline{\varphi_0^{\,2}} = \frac{\sum_{i=1}^{n} \varphi_0^{\,2}(x_i)}{n}\ ,\ \overline{\varphi_1^{\,2}} = \frac{\sum_{i=1}^{n} \varphi_1^{\,2}(x_i)}{n}.$$

Розглянемо багатовимірний випадок лінійної регресійної моделі.

Нехай $x = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & \dots & \vdots \\ x_{p1} & x_{p2} & \dots & x_{pm} \end{bmatrix}$ – матриця p значень, що спостерігаються, m змінних $x_1, x_2, \dots, x_m$, $y = (y_1, y_2, \dots, y_p)^T$ - вектор значень, що спостерігаються, пояснювальної змінної, де ^{T} - символ транспонування, тоді **загальна лінійна регресійна модель** може бути представлена в стандартному вигляді так:

$$y_i = \sum_{j=1}^{m} b_j x_{ij} + u_i \ , \ i = 1,2,\dots, p \quad \text{або} \quad y = xb + u,$$

де $b = (b_1, b_2, \dots, b_m)^T$ та $u = (u_1, u_2, \dots, u_p)^T$ - вектор констант та вектор членів помилки (залишків), відповідно.

Оцінкою найменших квадратів є такий вектор параметрів b, що мінімізує суму квадратів регресійних залишків E:

$$E = u^T u = (y-xb)^T (y-xb).$$

Виходячи з умови мінімізації суми квадратів регресійних залишків:

$$\frac{\partial E}{\partial b_i} = 0 \ ,$$

де i = 1,2,...,m, одержуємо систему нормальних рівнянь:

$$\frac{\partial E}{\partial b} = -2\left(x^T y - x^T x b\right) = 0 .$$

Звідки $b = (x^T x)^{-1} x^T y$, де $(x^T x)^{-1}$ – матриця, зворотна матриці $x^T x$, що по припущенню існує.

Для знаходження **зворотної матриці** рекомендується використовувати **узагальнений метод виключення з вибором**

ведучого елемента матриці, заснований на використанні **ітеративної формули Бен-Ізраеля.**

Узагальненою зворотною матрицею A_+ для довільної матриці A розміру (m x n) називається матриця, що задовольняє наступним умовам:

$$AA_+A=A,\ A_+AA_+=A_+,\ AA_+=(AA_+)^T,\ A_+A=(A_+A)^T.$$

Теорема Пенроуза говорить, що для будь-якої матриці існує матриця, що задовольняє всім цим умовам і притому єдина.

Нехай λ^* - максимальне характеристичне значення матриці AA^T. Тоді покладемо $\alpha = \frac{2}{|\lambda^*|}0.8$. Якщо позначити через $A_+^{(t)}$ t-е наближення, найближче до узагальненої зворотної матриці $\mathbf{A}_+$ для матриці A, то ми зможемо одержати A_+ з рекурентного співвідношення:

$$A_+^{(0)} = \alpha A;\ A_+^{(t+1)} = A_+^{(t)}\left(2I - AA_+^{(t)}\right).$$

Таким чином, $\lim\limits_{t\to\infty} A_+^{(t)} = A_+$.

Максимальне характеристичне значення λ^* можна знайти за допомогою наступного алгоритму.

Крок 1. Ініціалізація. Установити: k=1, z_i=1, i=1,2,...,m. Задати значення критерію збіжності ξ.

Крок 2. Обчислити $y_i = \sum_{j=1}^{m}\sum_{d=1}^{n} A_{ij}A_{jd}$, i=1,2,...,m.

Крок 3. Якщо y_k=0, то перейти на крок 4, інакше перейти на крок 6.

Крок 4. Якщо k<m, то k=k+1, інакше k=1.

Крок 5. Перейти на крок 3.

Крок 6. Обчислити $z_i = \frac{y_i}{y_k}$, i=1,2,...,m, та установити $\lambda^* = y_k$.

Крок 7. Перевірка збіжності: Якщо $\left|\frac{\lambda^*}{y_k} - 1\right| > \xi$, то перейти на крок 2, інакше – зупинення.

У тих випадках, коли **рівняння регресії** є **нелінійним** щодо оцінюваних параметрів, використовується **нелінійний метод найменших квадратів.**

Нехай y – змінна, що пояснюється; y_1, y_2 ,...,y_m – набір її спостережень; x_1, x_2, ..., x_k – змінні, що пояснюють, і-те спостереження за якими являє собою вектор: $(x_{i1},x_{i2},...,x_{ik})^T$.

Необхідно змінну, що пояснюється, y виразити через x_1, x_2, ..., x_k за допомогою функції f, вигляд якої відомий, однак невідомі деякі її параметри w_1, w_2, ..., w_n:

$$y_i = f_i(w_1,...,w_n; x_{i1},..., x_{ik}) + F_i, \ i = 1,2,...,m,$$

де F_i-відхилення. Попередній вираз можна записати більш компактно, як $y_i = f_i(w_1,...,w_n) + F_i$, i=1,2,...,m, залишивши лише ті параметри, що будемо шукати за методом найменших квадратів, тобто мінімізуючи

$$E = \sum_{i=1}^{m} F_i^2 \ .$$

Введемо вектори:

$$w = (w_1, w_2, ..., w_n)^T, \ f = (f_1, f_2, ..., f_m)^T,$$

$$F = (F_1, F_2, ..., F_m)^T, \ \ y = (y_1, y_2, ..., y_m)^T.$$

Тепер сформулюємо **задачу нелінійного регресійного аналізу:** знайти w*, таке, що при F = y-f цільова функція (сума квадратів залишків) $E=F^TF$ мінімізується.

Наближене значення w_t, що одержується на t-ому кроці ітеративного процесу і наступне наближене значення w_{t+1} зв'язані між собою вектором поправки Δw : $w_{t+1} = w_t + \Delta w$. Формула вектора поправки Δw відповідно до умови мінімізації виводиться з рішення системи лінійних рівнянь:

$$(A^TA)\,\Delta w \ = (-A^TF),$$

звідки:

$$\Delta w = -(A^T A)^{-1} A^T F.$$

Тут A – Якобіан F (матриця перших часткових похідних):

$$A = \begin{bmatrix} \frac{\partial F_1}{\partial w_1} & \cdots & \frac{\partial F_1}{\partial w_n} \\ \vdots & \frac{\partial F_i}{\partial w_j} & \vdots \\ \frac{\partial F_m}{\partial w_1} & \cdots & \frac{\partial F_m}{\partial w_n} \end{bmatrix}_{w = w_t}.$$

Мінімізація цільової функції E здійснюється за допомогою градієнтних методів оптимізації.

7.7 Алгоритм кластер-регресійної апроксимації та його нейромережева інтерпретація

Нехай задана навчальна вибірка екземплярів $x = \{x^s_j\}$, де x^s_j - значення j-ої ознаки s-го екземпляру, s=1,2,...,S; S - кількість екземплярів у навчальній вибірці, та кожному екземпляру співставлене значення прогнозованого параметру y^s. Тоді залежність y від x можна записати в вигляді: y = f(w,x)+Err, де f(w,x) - деяка функція, вид якої задається алгоритмом апроксимації або користувачем, w - множина параметрів, що дозволяють налагодити (навчити) функцію на вирішення певної апроксимаційної задачі, Err - деяка помилка, що виникає завдяки неповної відповідності виду апроксимуючої функції виду реально існуючої залежності та похибками у визначенні значень параметрів апроксимуючої функції.

Для знаходження параметрів апроксимуючої функції можна використовувати засоби регресійного аналізу. При використанні одновимірних лінійних регресійних моделей для апроксимації багатовимірних нелінійних залежностей, як правило, не вдається забезпечити бажану точність прогнозування, але параметри апроксимуючої функції можуть бути розраховані у неітеративному режимі. З іншого боку, у певних областях у просторі ознак значення

прогнозованого параметру може визначатися в основному якоюсь однією ознакою, що є найбільш сильно корельованою з прогнозованим параметром у цій області.

Тому, узагальнюючи сказане вище, можна заключити, що кусочно-лінійна регресія для декількох ознак може забезпечити кращу апроксимацію, ніж лінійна регресійна модель для якоїсь однієї ознаки. Побудова кусочно-регресійної моделі вимагає виділення областей компактно згрупованих екземплярів у просторі ознак з близькими значеннями прогнозованого параметру - кластерів.

Алгоритм побудови кластер-регресійної апроксимації запишемо у наступному виді.

Крок 1. Ініціалізація: задати навчальну вибірку екземплярів x та набір співставлених їм значень цільового параметру y.

Крок 2. На основі екземплярів навчальної вибірки сформувати кластери C^q, q = 1,2,...,Q, де Q - кількість сформованих кластерів. Для виділення кластерів пропонується використовувати описаний нижче алгоритм формування кластерів.

Крок 3. Розділити екземпляри навчальної вибірки на групи, що відповідають кожному кластеру за мінімумом відстані від екземпляру до центру кластера у просторі ознак.

Крок 4. Для екземплярів кожної групи знайти значення коефіцієнтів кореляції їхніх ознак та значень прогнозованого параметру:

$$r^q(x_j, y) = \frac{\sum_{s=1}^{S}\left(x_j^s - \frac{1}{S^q}\sum_{s=1}^{S} x_j^s\right)\left(y^s - \frac{1}{S^q}\sum_{s=1}^{S} y^s\right)}{\sqrt{\sum_{s=1}^{S}\left(x_j^s - \frac{1}{S^q}\sum_{s=1}^{S} x_j^s\right)^2 \sum_{s=1}^{S}\left(y^s - \frac{1}{S^q}\sum_{s=1}^{S} y^s\right)^2}}, x^s \in C^q,$$

$$j=1,2,...,N,$$

де N - кількість ознак, S^q - кількість екземплярів, що відносяться до q-го кластеру.

Крок 5. Для кожної групи екземплярів знайти номер ознаки, модуль коефіцієнту кореляції котрого з прогнозованим параметром є найбільшим:

$$g^q = \arg\max_j \left| r^q(x_j, y) \right|, \; q = 1,2,...,Q; \; j=1,2,...,N.$$

Крок 6. Для кожної групи екземплярів оцінити значення параметрів рівняння лінійної регресії для прогнозованого параметру за ознакою x_{g^q}.

Якщо рівняння лінійної регресії прогнозованого параметру за ознакою g^q для екземплярів q-го кластеру має вид: $y_q = b_{q0} x_{g^q} + b_{q1}$, тоді значення коефіцієнтів b_{q0} та b_{q1} можуть бути знайдені за допомогою формул:

$$b_{q0} = (a^q_1 - a^q_2 a^q_3)/(a^q_4 - a^q_3 a^q_3),$$

$$b_{q1} = (a^q_4 a^q_2 - a^q_3 a^q_1)/(a^q_4 - a^q_3 a^q_3),$$

$$a^q_1 = \frac{1}{S^q}\sum_{s=1}^{S} y^s x^s_{g^q}, x^s \in C^q,$$

$$a^q_2 = \frac{1}{S^q}\sum_{s=1}^{S} y^s, x^s \in C^q,$$

$$a^q_3 = \frac{1}{S^q}\sum_{s=1}^{S} x^s_{g^q}, x^s \in C^q,$$

$$a^q_4 = \frac{1}{S^q}\sum_{s=1}^{S} \left(x^s_{g^q}\right)^2, x^s \in C^q.$$

Алгоритм розрахунку значення прогнозованого параметру на основі кластер-регресійної апроксимації запишемо у наступному вигляді.

Крок 1. Ініціалізація: Задати вибірку екземплярів для котрих здійснюється прогнозування $x=\{x^s_j\}$. Задати кількість кластерів Q та координати їхніх центрів $C=\{C^q_j\}$. Для кожного кластеру задати номер ознаки g^q та значення параметрів рівняння регресії b_{q0} та b_{q1}.

Крок 2. Для кожного екземпляру визначити відстані від екземпляру до центрів усіх кластерів $R(x^s,C^p)$, s=1,...,S; p=1,...,Q.

Крок 3. Для кожного екземпляру знайти:

$$q = \arg\min_{p} R(x^s, C^p),\ s=1,...,S;\ p=1,...,Q.$$

Крок 4. Значення прогнозованого параметру для кожного s-го екземпляру визначити за формулою:

$$y^s = y^s_q = b_{q0} x^s_{g^q} + b_{q1},\ s=1,...,S.$$

Для **формування кластерів** (компактних областей групування екземплярів у просторі ознак) пропонується використовувати наступний алгоритм.

Крок 1. Ініціалізація параметрів алгоритму побудови навчальної вибірки. Задати вихідну вибірку екземплярів $x_{вих.}$ та співставлені їм номери класів або значення прогнозованого параметру $y_{вих.}$, а також L - кількість разбиттів вихідної вибірки. Занести у змінну N кількість ознак, що характеризують екземпляри, а у змінну S – кількість екземплярі вихідної вибірки. Прийняти ширину допустимого інтервалу варіації прогнозованого параметру:

$$dy = |\max(y_{вих.}) - \min(y_{вих.})| / L,$$

де min(a) та max(a) – мінімальне та максимальне значення вектору a, відповідно. Установити лічильник newind=1.

Крок 2. Обчислити відстані між екземплярами вихідної вибірки:

$$R(p,q) = \begin{cases} \sqrt{\sum_{j=1}^{N} \left(x_j^p - x_j^q\right)^2},\ p \neq q; & p=1,...,S,\ q=1,..,S, \\ RealMax,\ p = q, \end{cases}$$

де RealMax - максимально число, що може бути представлене у ЕОМ, x^p_j- значення j-ої ознаки p-го екземпляру.

Крок 3. Знайти у матриці відстаней R мінімальний елемент minx та його індекси q та p, а також максимальний елемент maxx, за умови, що при знаходженні мінімуму й максимуму тут і далі ігноруються елементи, які дорівнюють RealMax.

Крок 4. Прийняти a=| maxx-minx | / (2L).

Крок 5. Якщо minx<Realmax, тоді перейти на крок 6, у противному випадку – перейти на крок 13.

Крок 6 Прийняти: $C^{(newind)} = x_{вих.}(q)$, y*(newind) = $y_{вих.}(q)$, де C та y* – масиви еталонних екземплярів кластерів та співставлених ним прогнозованих значень, відповідно. Установити: newind=newind+1, значення текучого мінімального елементу у рядку yteck=R(q,p). Зайти мінімальний елемент mminx та його індекси mqmin та mpmin серед елементів q-го рядку матриці R.

Крок 7. Якщо mminx < Realmax, тоді перейти на крок 8, у протилежному випадку – перейти на крок 11.

Крок 8. Установити значення покажчика видаленого екземпляру із стовпця deleted = 0 (у матриці R нумерація рядків та стовпців повинна начинатися з 1).

Крок 9. Якщо | yteck - R(q,mpmin) | ≤ a, тоді перейти на крок 10, інакше прийняти: deleted = mpmin, R(q,mpmin) = Realmax, R(mpmin,q) = Realmax та перейти на крок 11.

Крок 10. Якщо | ($y_{вих.}$(mpmin)-$y_{вих.}$(q)| ≤ dy, тоді прийняти:

R(v, mpmin) = realmax, R(mpmin, v) = realmax, v=1,..,M, у противному випадку – прийняти:

$$C^{(newind)}=x_{вих.}(mpmin),\ y^*(newind) = y_{вих.}(mpmin),$$

$$newind = newind+1,\ R(v, mpmin) = Realmax,$$

$$R(mpmin,v)=Realmax,\ v=1,...,S.$$

Крок 11. Знайти мінімальний елемент mminx та його індекси mqmin та mpmin серед елементів q-го рядку матриці R.

Крок 12. Перейти на крок 7.

Крок 13. Прийняти: R(v, q) = Realmax, R(q, v) = Realmax, v = 1,...,S, покажчик видаленого екземпляру із рядку dstr = q. Знайти у матриці відстаней R мінімальний елемент minx та його індекси q та p.

Крок 14. Перейти на крок 5.

Крок 15. Якщо (deleted ≠ dstr) та (deleted>0), тоді прийняти:

$C^{(newind)} = x_{вих.}$(deleted), y*(deleted) = $y_{вих.}$(deleted).

Крок 16. Зупинення.

У результаті виконання цього алгоритму для вихідної вибірки $x_{вих.}$ та співставленого їй набору значень $y_{вих.}$ ми отримаємо множину еталонних екземплярів кожного кластеру C та співставлений ним набір значень y*. Після цього необхідно здійснити розділення на кластери для всіх екземплярів вихідної вибірки за мінімумом близькості до еталонних екземплярів кластеру, а потім для кожного кластеру прийняти координати центру кластеру рівними:

$$C_j^q = \frac{1}{S^q}\sum_{s=1}^{S} x_j^s,\ x^s \in C^q\ ,\ q = 1,..., Q;\ j = 1,...,N;$$

де S^q - кількість екземплярів вихідної вибірки, що належать до q-го кластеру, Q - кількість сформованих кластерів.

Після формування кластерів алгоритм кластер-регресійної апроксимації можна використовувати для синтезу та налагоджування вагових коефіцієнтів багатошарової НМ.

Якщо сформовано Q кластерів, тоді НМ буде складатися з 5 шарів нейронів, дискримінантні функції яких будуть задаватися формулами:

$$\varphi^{(1,i)}(x,w) = \sum_{j=1}^{N} w_j x_j + w_0\ ,\ i=1,2,...,Q;$$

$$\varphi^{(1,i)}(x,w) = \sum_{j=1}^{N} (x_j - w_j)^2\ ,\ i = Q+1,\ Q+2,...,2Q;$$

$$\varphi^{(2,i)}(x,w) = \sum_{j=1}^{2} w_j x_j + w_0\ ,\ i=1,2,...,Q^2;$$

$$\varphi^{(3,i)}(x,w) = \sum_{j=1}^{Q} w_j x_j + w_0\ ,\ i=1,2,...,Q;$$

$$\varphi^{(4,i)}(x,w) = \prod_{j=1}^{2} w_j x_j\ ,\ i=1,2,...,Q;$$

$$\varphi^{(5,i)}(x,w) = \sum_{j=1}^{Q} w_j x_j + w_0\ ,\ i=1;$$

де $\varphi^{(\mu,i)}$ - дискримінантна функція і-го нейрону μ-го шару.

Функції активації нейронів будуть визначатися виразами:

$$\psi^{(\mu)}(\alpha)=\begin{cases}0, \alpha \le 0, \\ 1, \alpha > 0;\end{cases} \quad \mu=2,3;$$

$$\psi^{(\mu)}(\alpha) = \alpha, \quad \mu=1,4,5.$$

Схема НМ, синтезованої та навченої на основі запропонованого алгоритму, зображена на рис. 7.10.

Ваговий коефіцієнт j-го входу i-го нейрону μ-го шару мережі буде розраховуватися за формулою:

$$w_j^{(\mu,i)}=\begin{cases}0, \mu=5, i=1, j=0; \\ 1, \mu=5, i=1, j>0; \\ 1, \mu=4, \forall i, j>0; \\ 1-Q, \mu=3, i=1, j=0; \\ 1, \mu=3, i=1, j=0; \\ 0, \mu=2, \forall i, j=0; \\ 1, \mu=2, \forall i, j=2; \\ -1, \mu=2, \forall i, j=1; \\ b_{q0}, \mu=1, i=q, j=g^q, q=1,...,Q; \\ b_{q1}, \mu=1, i=q, j=0, q=1,...,Q; \\ 0, \mu=1, i=q, 0<j\neq g^q, q=1,...,Q; \\ C_j^q, \mu=1, i=Q+1, Q+2,...,2Q, j=1,2,...,N.\end{cases}$$

Розглянутий алгоритм дозволяє досягнути суттєво більшої точності, ніж метод лінійної одновимірної регресії, проте він має таку ж простоту та ефективність обчислювальних процедур.

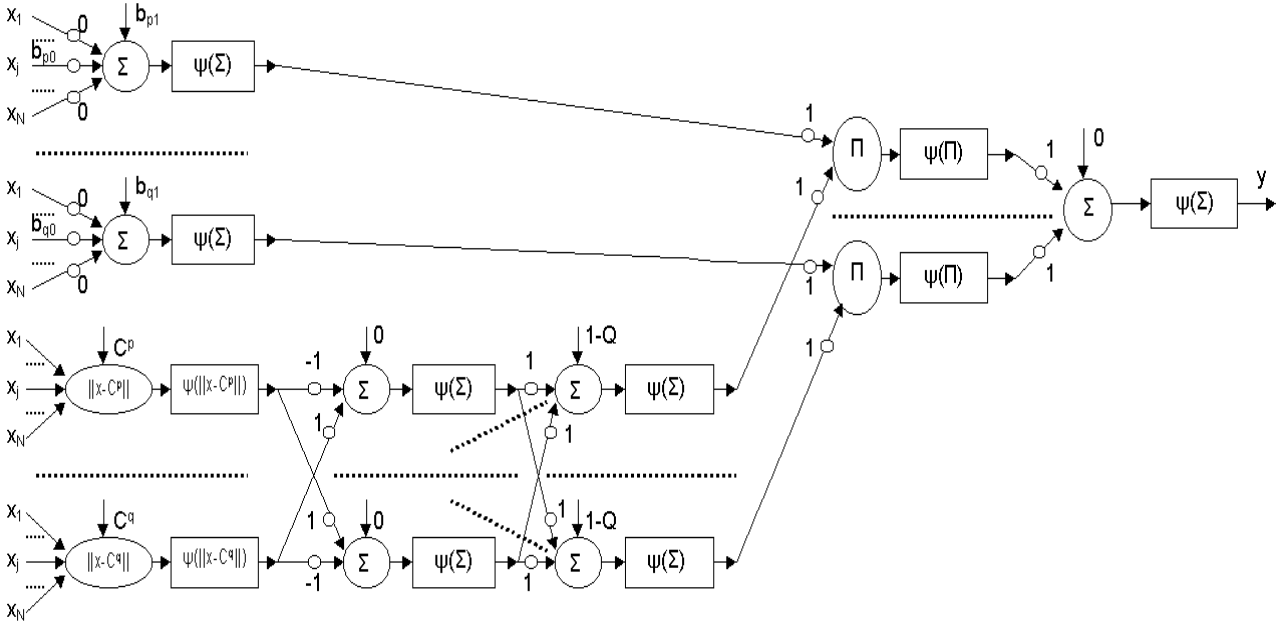


Рис. 7.10 - Схема НМ, синтезованої на основі алгоритму кластер-регресійної апроксимації (випадок лінійної одновимірної регресії)

НМ, що отримується у результаті синтезу й навчання на основі розглянутого алгоритму, є логічно прозорою та легко перетворюється у дерево рішаючих правил "Якщо - то", що дозволяє використовувати її для видобування знань з даних. Точність класифікації та швидкість навчання НМ, сформованої на основі запропонованого алгоритму, є достатніми для багатьох прикладних задач діагностики та прогнозування.

У випадку, коли запропонований алгоритм не дозволяє отримати модель залежності прогнозованого параметру від набору ознак із заданою точністю, тоді кроки 5 та 6 алгоритму побудови кластер-регресійної апроксимації можна замінити на наступні.

Крок 5. Для кожного q-го кластеру усі ознаки екземплярів, для яких $|r(x_j,y)| < P$, де P - заданий порог, $0 < P < 1$, вилучити із набору ознак екземплярів, що належать до q-го кластеру.

Крок 6. Для кожного q-го кластеру оцінити параметри багатовимірної лінійної регресії прогнозованого параметру (див. розд. 7.6):

$$y^s = \sum_{j=1}^{m_q} b_{qj} x_j^s, x_j^s \in C^q ,$$

$$b = (x^T x)^{-1} x^T y,$$

де m_q - кількість ознак екземплярів q-го кластеру, для яких:

$$|r(x_j,y)| < P.$$

На основі розрахованих b_{qj} визначити коефіцієнти для усього вихідного набору ознак екземплярів q-го кластеру:

$$\beta_{qj} = \begin{cases} 0, \text{якщо } \left|r\left(x_j, y\right)\right| < P; \\ b_{qk}, \text{у протилежному випадку;} \end{cases}$$

де k - номер ознаки з набору ознак, що враховувалися при визначенні параметрів багатовимірної регресії для q-го кластеру, яка відповідає j-й ознаці із вихідного набору ознак.

Алгоритм розрахунку значення прогнозованого параметру на основі кластер-регресійної апроксимації для випадку багатовимірної лінійної регресії запишемо у наступному вигляді.

Крок 1. Ініціалізація: Задати вибірку екземплярів для котрих здійснюється прогнозування $x=\{x^s_j\}$. Задати кількість кластерів Q та координати їхніх центрів $C=\{C^q_j\}$. Для кожного кластеру задати значення параметрів багатовимірної лінійної регресії $\{\beta_{qj}\}$.

Крок 2. Для кожного екземпляру визначити відстані від екземпляру до центрів усіх кластерів $R(x^s,C^p)$, s=1,...,S; p=1,...,Q.

Крок 3. Для кожного екземпляру знайти:

$$q = \arg\min_p R(x^s, C^p),\ s=1,...,S;\ p=1,...,Q.$$

Крок 4. Значення прогнозованого параметру для кожного s-го екземпляру визначити за формулою:

$$y^s = \sum_{j=1}^{N} \beta_{qj} x^s_j,\ x^s_j \in C^q,\ s = 1, ..., S.$$

Розглянутий алгоритм кластер-регресійної апроксимації для випадку багатовимірної лінійної регресії можна використовувати для синтезу та налагоджування вагових коефіцієнтів багатошарової НМ.

Якщо сформовано Q кластерів, тоді НМ буде складатися з 5 шарів нейронів, дискримінантні функції яких будуть задаватися формулами:

$$\varphi^{(1,i)}(x,w) = \sum_{j=1}^{N} w_j x_j + w_0,\ i=1,2,...,Q;$$

$$\varphi^{(1,i)}(x,w) = \sum_{j=1}^{N} (x_j - w_j)^2,\ i = Q+1, Q+2,...,2Q;$$

$$\varphi^{(2,i)}(x,w) = \sum_{j=1}^{2} w_j x_j + w_0,\ i=1,2,...,Q^2;$$

$$\varphi^{(3,i)}(x,w) = \sum_{j=1}^{Q} w_j x_j + w_0,\ i=1,2,...,Q;$$

$$\varphi^{(4,i)}(x,w)=\prod_{j=1}^{2} w_j x_j \text{ , i=1,2,...,Q;}$$

$$\varphi^{(5,i)}(x,w)=\sum_{j=1}^{Q} w_j x_j + w_0 \text{ , i=1;}$$

де $\varphi^{(\mu,i)}$ - дискримінантна функція i-го нейрону μ-го шару.

Функції активації нейронів будуть визначатися виразами:

$$\psi^{(\mu)}(\alpha)=\begin{cases}0, \alpha \le 0,\\ 1, \alpha > 0;\end{cases} \quad \mu=2,3;$$

$$\psi^{(\mu)}(\alpha)=\alpha, \quad \mu=1,4,5.$$

Ваговий коефіцієнт j-го входу i-го нейрону μ-го шару мережі буде розраховуватися за формулою:

$$w_j^{(\mu,i)}=\begin{cases}0, \mu=5, i=1, j=0;\\ 1, \mu=5, i=1, j>0;\\ 1, \mu=4, \forall i, j>0;\\ 1-Q, \mu=3, i=1, j=0;\\ 1, \mu=3, i=1, j=0;\\ 0, \mu=2, \forall i, j=0;\\ 1, \mu=2, \forall i, j=2;\\ -1, \mu=2, \forall i, j=1;\\ \beta_{ij}, \mu=1, i=1,...,Q, j=1,...,N;\\ 0, \mu=1, i=1,...,Q, j=0,;\\ C_j^q, \mu=1, i=Q+1, Q+2,...,2Q, j=1,...,N.\end{cases}$$

Схема НМ, синтезованої та навченої на основі запропонованого алгоритму, зображена на рис. 7.11.

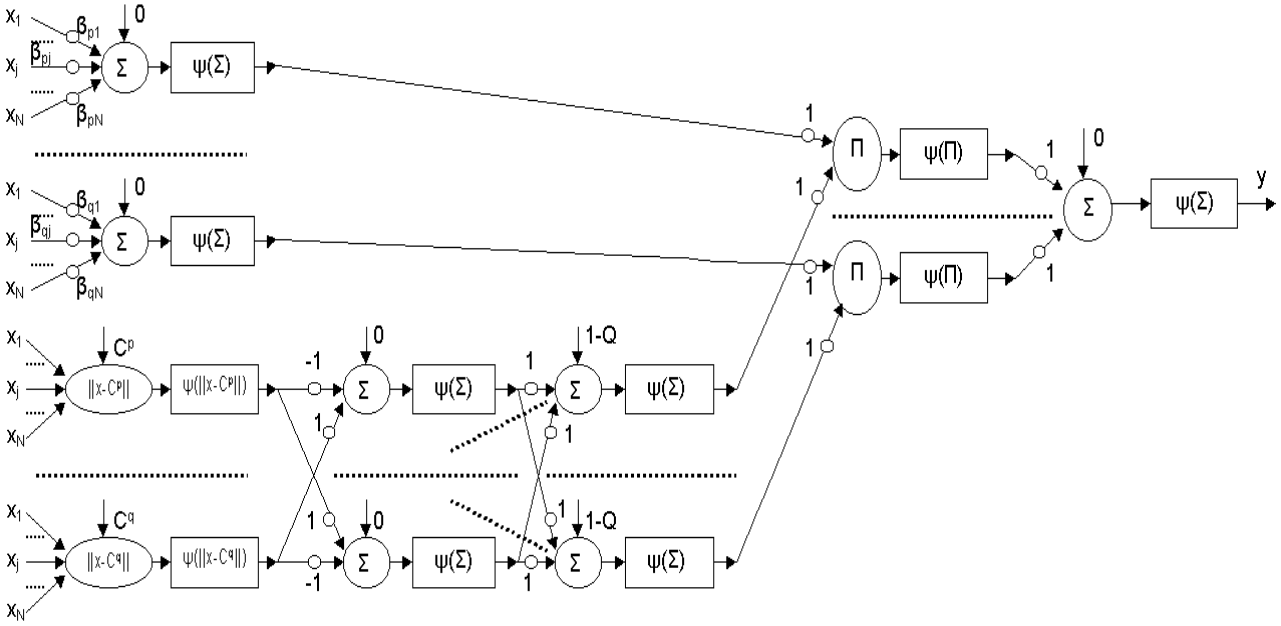


Рис. 7.11 - Схема НМ, синтезованої на основі алгоритму кластер-регресійної апроксимації (випадок лінійної багатовимірної регресії)

Запропонований алгоритм дозволяє досягнути більшої точності, ніж методи лінійної регресії, проте він має достатньо високу ефективність та простоту обчислювальних процедур.

У випадку, коли алгоритм кластер-регресійної апроксимації для випадку багатовимірної лінійної регресії не дозволяє отримати модель залежності прогнозованого параметру від набору ознак із заданою точністю, тоді кроки 5 та 6 алгоритму побудови кластер-регресійної апроксимації можна замінити на наступний крок.Крок 5. Для кожного q-го кластеру оцінити параметри багатовимірної нелінійної регресії прогнозованого параметру (див. розд. 7.6):

$$y^s = \frac{1}{1+\exp\left(-\sum_{j=1}^{N}\beta_{qj}x_j^s\right)}, x_j^s \in C^q ,$$

де β_{qj} - ваговий коефіцієнт j-ої ознаки для багатовимірної нелінійної регресії прогнозованого параметру у q-му кластері.

Алгоритм розрахунку значення прогнозованого параметру на основі кластер-регресійної апроксимації для випадку багатовимірної нелінійної регресії запишемо у наступному виді.

Крок 1. Ініціалізація: Задати вибірку екземплярів для котрих здійснюється прогнозування $x=\{x_j^s\}$. Задати кількість кластерів Q та координати їхніх центрів $C=\{C_j^q\}$. Для кожного кластеру задати значення параметрів багатовимірної нелінійної регресії $\{\beta_{qj}\}$.

Крок 2. Для кожного екземпляру визначити відстані від екземпляру до центрів усіх кластерів $R(x^s,C^p)$, s=1,...,S; p=1,...,Q.

Крок 3. Для кожного екземпляру знайти:

$$q = \arg\min_p R(x^s, C^p), \ s=1,...,S; \ p=1,...,Q.$$

Крок 4. Значення прогнозованого параметру для кожного s-го екземпляру визначити за формулою:

$$y^s = \frac{1}{1+\exp\left(-\sum_{j=1}^{N}\beta_{qj}x_j^s\right)}, x_j^s \in C^q, \ s = 1, ..., S.$$

Розглянутий алгоритм кластер-регресійної апроксимації для випадку багатовимірної лінійної регресії можна використовувати для синтезу та налагоджування вагових коефіцієнтів багатошарової НМ.

Якщо сформовано Q кластерів, тоді НМ буде складатися з 5 шарів нейронів, дискримінантні функції яких будуть задаватися формулами:

$$\varphi^{(1,i)}(x,w)=\sum_{j=1}^{N} w_j x_j + w_0 \text{ , } i=1,2,...,Q;$$

$$\varphi^{(1,i)}(x,w)=\sum_{j=1}^{N} (x_j - w_j)^2 \text{ , } i = Q+1, Q+2,...,2Q;$$

$$\varphi^{(2,i)}(x,w)=\sum_{j=1}^{2} w_j x_j + w_0 \text{ , } i=1,2,...,Q^2;$$

$$\varphi^{(3,i)}(x,w)=\sum_{j=1}^{Q} w_j x_j + w_0 \text{ , } i=1,2,...,Q;$$

$$\varphi^{(4,i)}(x,w)=\prod_{j=1}^{2} w_j x_j \text{ , } i=1,2,...,Q;$$

$$\varphi^{(5,i)}(x,w)=\sum_{j=1}^{Q} w_j x_j + w_0 \text{ , } i=1;$$

де $\varphi^{(\mu,i)}$ - дискримінантна функція і-го нейрону μ-го шару.

Функції активації нейронів будуть визначатися виразами:

$$\psi^{(\mu,i)}(\alpha)=\frac{1}{1+e^{-\alpha}} \text{ , } \mu = 1;\ i = 1,...,Q\ ;$$

$$\psi^{(\mu,i)}(\alpha) = \alpha,\ \mu = 1;\ i = Q+1,...,2Q;$$

$$\psi^{(\mu)}(\alpha)=\begin{cases}0, \alpha \le 0, \\ 1, \alpha > 0;\end{cases} \quad \mu=2,3;$$

$$\psi^{(\mu)}(\alpha) = \alpha, \ \mu = 4,5.$$

Ваговий коефіцієнт j-го входу i-го нейрону μ-го шару мережі буде розраховуватися за формулою:

$$w_j^{(\mu,i)} = \begin{cases}0, \mu = 5, i = 1, j = 0; \\ 1, \mu = 5, i = 1, j > 0; \\ 1, \mu = 4, \forall i, j > 0; \\ 1 - Q, \mu = 3, i = 1, j = 0; \\ 1, \mu = 3, i = 1, j = 0; \\ 0, \mu = 2, \forall i, j = 0; \\ 1, \mu = 2, \forall i, j = 2; \\ -1, \mu = 2, \forall i, j = 1; \\ \beta_{ij}, \mu = 1, i = 1,..., Q, j = 1,..., N; \\ 0, \mu = 1, i = 1,..., Q, j = 0; \\ C_j^q, \mu = 1, i = Q+1, Q+2,..., 2Q, j = 1,..., N.\end{cases}$$

Схема НМ, синтезованої та навченої на основі запропонованого алгоритму, буде виглядати також, як і зображена на рис. 7.11.

Запропонований алгоритм дозволяє досягати ще більшої точності у порівнянні із лінійною регресією, але необхідність розв'язання задачі багатовимірної нелінійної оптимізації для кожного кластеру призведе до суттєво більших витрат часу у процесі навчання мережі.

РОЗДІЛ 8. ПРОГРАМНІ ЗАСОБИ ЧИСЕЛЬНОЇ ОПТИМІЗАЦІЇ

8.1 Оптимізація і нейронні мережі в пакеті Matlab

MATLAB (MATrix LABoratory – МАТрична ЛАБораторія) представляє собою математичний пакет, призначений для вирішення задач обчислювальної математики, математичної фізики і побудови чисельних моделей складних об'єктів і процесів.

Пакет MATLAB складається з інтерпретатора - модельного середовища, що має термінальний інтерфейс, ядра (набору найпростіших стандартних операцій, функцій і процедур для обчислень), а також бібліотек функцій (Toolbox).

Перевагами пакету є: багаті графічні можливості, великий набір математичних функцій, простота вбудованої мови MATLAB, можливість автоматичного перетворення текстів програм мовою MATLAB у тексти програм мовою Сі, а також те, що тексти бібліотечних функцій постачаються у вихідному вигляді.

До недоліків пакету варто віднести відсутність дружнього інтерфейсу користувача і низьку швидкість роботи.

Для вирішення **оптимізаційних задач** у середовищі MATLAB необхідно встановити і використовувати бібліотеку **Optimization Toolbox**. Розглянемо деякі найбільш важливі функції бібліотеки Optimization Toolbox.

Задача **лінійного програмування** $F = c^T x \rightarrow \text{Min}$ при обмеженнях $Ax \leq b$, де c та b - вектори, A – матриця, вирішується за допомогою функції lp.

Формат виклику	Дія
x = lp(c,A,b)	повертає вектор x, що мінімізує рівняння $c^T x$ при обмеженнях $Ax \leq b$.
x = lp(c,A,b,vlb,vub)	встановлює нижню vlb і верхню vub границі для x, що обмежують рішення в діапазоні $vlb \leq x \leq vub$.
x = lp(c,A,b,vlb,vub,x0)	установлює початкову точку в x0.
x = lp(c,A,b,vlb,vub,x0,e)	вказує, що перші e обмежень – рівності.

Одновимірний пошук мінімуму функції f(w) на фіксованому інтервалі $w_1 < w < w_2$, де w_1 і w_2 – скаляри, а f(w) – функція, що повертає скаляр, здійснюється за допомогою функції fmin:

w = fmin('f', w1,w2) повертає значення w, що є локальним мінімумом функції f(w) на інтервалі w1 < w < w2.

Багатовимірний безградієнтний пошук мінімуму функції f(w) без обмежень, де w – вектор керувальних змінних, а f(w) – функція, що повертає скаляр, здійснюється на основі функції fmins:

w = fmins('f',w0) повертає вектор w, що є локальним мінімумом функції f(w) поблизу початкового вектора w0.

Багатовимірний градієнтний безумовний пошук мінімуму функції f(w), де w – вектор керувальних змінних, а f(w) – функція, що повертає скаляр, здійснюється за допомогою функції fminu:

w = fminu('f',w0) починаючи з точки w0 шукає мінімум функції f.

Для моделювання **нейронних мереж** за допомогою пакета MATLAB необхідно встановити і використовувати бібліотеку **Neural Network Toolbox**. Розглянемо приклади побудови моделей і навчання НМ мовою пакета MATLAB.

Моделювання і навчання **радіально-базисних НМ**

x = [1 2 3];	Задаємо значення ознак екземплярів навчальної вибірки: 3 екземпляри (стовпці), 1 ознака (рядки).
y = [2.0 4.1 5.9];	Задаємо значення параметра, що прогнозується, для 3 екземплярів навчальної вибірки.
net = newrb(x,y);	Створюємо і навчаємо радіально-базисну НМ
a = sim(net,x)	Обчислюємо по навченій мережі net значення параметра, що прогнозується, для екземплярів, які характеризуються набором значень ознак x.

Моделювання та навчання **персептрона**

x = [0.1 0.5 0.2 0.4 0.3 0.9; 0.9 0.5 0.8 0.6 0.7 0.1; 0.3 0.0 0.6 0.1 0.2 0.9];	Задаємо значення ознак екземплярів навчальної вибірки: 6 екземплярів (стовпці), 3 ознаки (рядки).
y = [1 0 1 0 1 0];	Задаємо номери класів для 6 екземплярів навчальної вибірки.
net=newff(repmat([0 1], 3,1), [2,1],{'logsig', 'logsig'}, 'trainlm');	Створюємо нейронну мережу net і визначаємо її топологію: діапазон зміни значень ознак [0 1], кількість ознак – 3, кількість вихідних змінних – 1, на першому шарі – 2 нейрони, на другому шарі – 1 нейрон, нейрони 1 та 2 шарів мають сигмоїдальні функції активації (logsig), для навчання мережі використовується метод Левенберга-Марквардта (trainlm).
net.trainparam. show = 25;	Задаємо період відображення інформації про процес навчання на екрані в циклах (епохах) навчання.
net.trainparam.lr = 0.01;	Задаємо крок навчання.
net.trainparam. epochs = 500;	Задаємо максимально припустиму кількість циклів навчання (епох).
net.trainparam. goal = 0.01;	Задаємо максимально припустиме значення критерію навчання (помилки навчання).
ct = cputime;	Визначаємо і запам'ятовуємо поточне значення лічильника часу в змінній ct.
net = train(net, x,y);	Навчаємо нейронну мережу net на основі навчальної вибірки, що представлена набором значень ознак екземплярів x та набором значень відповідних їм номерів класів y.
ct = cputime-ct	Визначаємо поточне значення лічильника часу, віднімаємо з нього значення змінної ct – визначаємо час навчання НМ, що заносимо в змінну ct й видаємо на екран (ознака друку на екрані – відсутність символу “;” наприкінці оператора).
a = round(sim(net, x));	Обчислюємо по навченій мережі net номери класів для екземплярів, що характеризуються набором значень ознак x.

Окрім розглянутих засобів моделювання НМ пакет MATLAB, починаючи з версії 6.0, містить візуальний інтерфейсний модуль **nntool**, який входить до бібліотеки Neural Network Toolbox.

Використання nntool дозволяє більш зручними засобами, ніж написання програми вручну, будувати нейромережеві моделі об'єктів та процесів. Розглянемо деякі основні можливості та прийоми роботи із засобом nntool.

Після запуску Matlab.exe в командному вікні для початку роботи із nntool треба ввести: nntool. Після цього завантажиться засіб nntool (рис. 8.1).

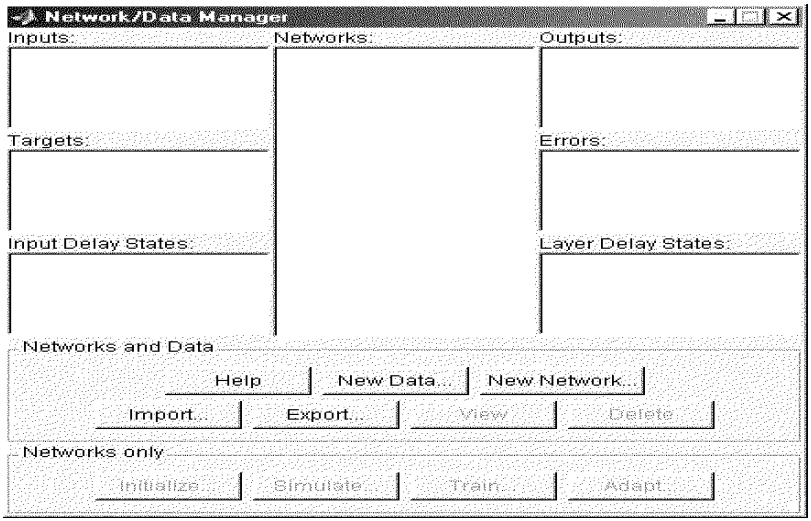


Рис. 8.1 - Головна діалогова форма засобу nntool

На панелі Network and Data ("Нейромережі та дані") користувач має натиснути кнопки для задання вихідних даних для побудови нейромережевої моделі.

Кнопка New Data ("Нові дані") викликає редактор для створення нових даних (рис. 8.2).

Поле Name ("Ім'я") задає ім'я нової змінної середовища MATLAB, до якої зберігається масив нових даних, що вводяться у полі Value ("Значення"). Панель Data Type ("Тип даних") визначає призначення введених даних: Inputs - входи мережі, Targets - цільові

значення виходів мережі, Input Delay States та Layer Delay States - описи затримок на входах та у шарах мережі, Outputs - фактичні значення на виходах мережі, Errors - помилки мережі.

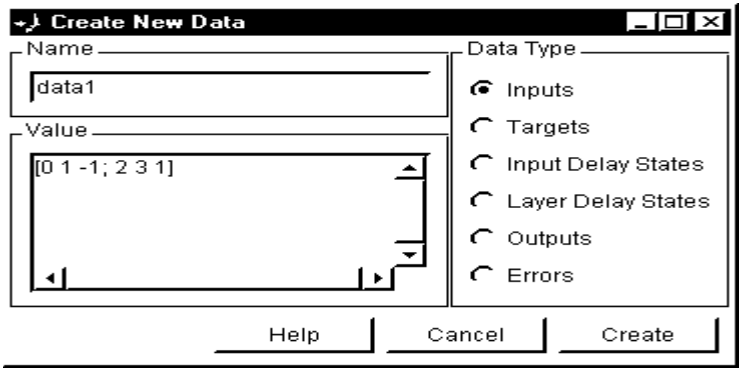


Рис. 8.2 - Редактор даних засобу nntool

Якщо дані уже існують у вигляді зовнішніх файлів або містяться у середовищі MATLAB у вигляді змінних, вони можуть бути імпортовані за допомогою кнопки Import ("Імпорт"). При цьому з'являється діалогова форма (рис. 8.3).

Рис. 8.3 - Діалогова форма імпорту даних

Поле Source ("Джерело") дозволяє обрати джерело введення даних: Import from Matlab workspace (імпорт даних із середовища MATLAB) або Load from disk file (завантаження даних із файлу на диску). Кнопка Browse дозволяє обрати необхідний файл.

Поле Select a Variable ("Вибір змінної") дозволяє вказати засобу nntool, яку змінну треба використовувати для імпорту даних.

Панель Destination ("Приймач") дозволяє задати змінну для прийому даних, що імпортуються. Її ім'я вказується у полі Name ("Ім'я"), а призначення (Import as) обирається із наведеного меню.

Кнопка Export ("Експорт") головної діалогової форми дозволяє зберегти дані із середовища nntool у файлі на диску, або передати їх до середовища MATLAB.

Кнопка "New Network" ("Нова мережа") викликає діалогову форму для конструювання нейромережі та визначення її параметрів (рис. 8.4).

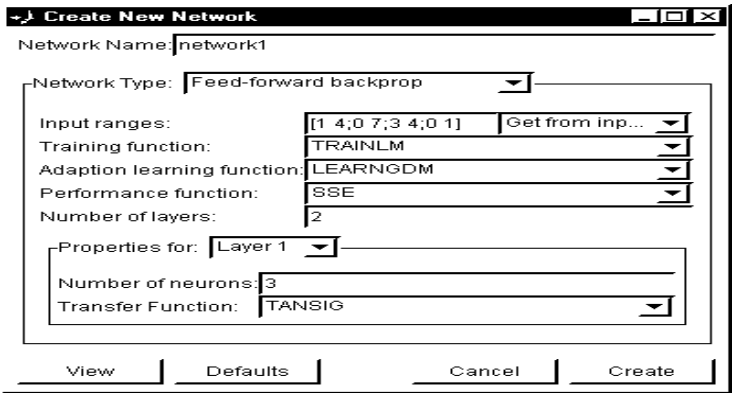


Рис. 8.4 - Форма конструювання нейромережі

Поле Network Name ("Ім'я мережі") визначає ім'я змінної, де зберігається мережа. Список вибору Network Type ("Тип мережі") дозволяє обрати тип архітектури мережі (наприклад, Feed-forward backprop - багатошарова нейромережа прямого поширення), поля Training Function, Adaptation Learning Function, Performance Function та Number of Layers визначають, відповідно, тип алгоритму

навчання мережі, тип алгоритму адаптації ваг мережі, цільову функцію та кількість шарів мережі.

Панель Properties for Layer K дозволяє задати властивості для нейронів K-го шару мережі. У полі Number of Neurons вказують кількість нейронів для поточного шару, а у полі Transfer Function - тип функції активації нейронів поточного шару мережі.

Кнопка "View" дозволяє отримати графічне зображення схеми побудованої нейромережі (рис. 8.5).

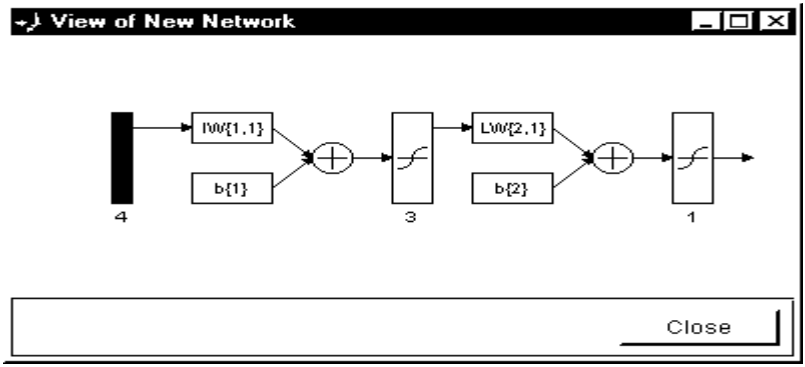


Рис. 8.5 - Приклад зображення форми нейромержі, побудованої за допомогою засобу nntool

Кнопка "Delete" головної форми дозволяє видалити непотрібний елемент даних (змінну або мережу).

Кнопка "Help" дозволяє викликати довідкову службу MATLAB з описом необхідних компонентів та поясненнями щодо їхнього використання.

Після побудови нейромережі у нижній частині головної діалогової форми nntool стає доступною панель Networks only, що призначена для роботи із побудованою мережею (рис. 8.6).

При натисненні будь-якої з кнопок цієї панелі викликається діалогова форма Network ("Мережа"), яка містить набір закладок-панелей для роботи з мережею (рис. 8.7-8.10).

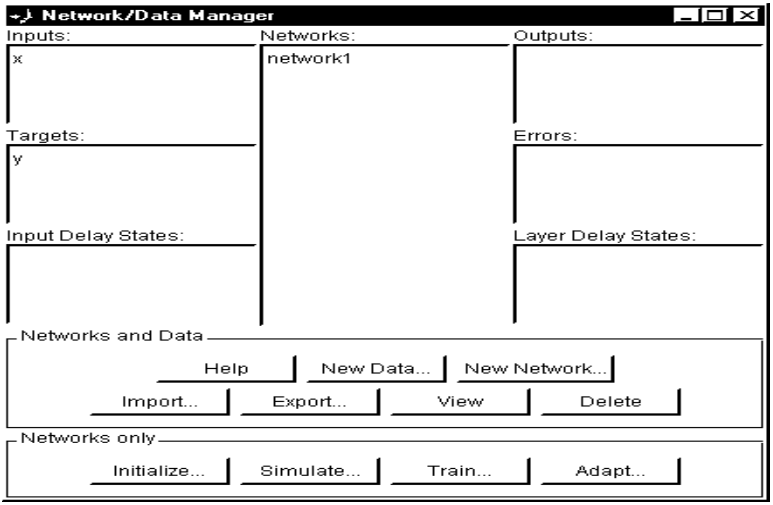


Рис 8.6 - Головна діалогова форма nntool після побудови мережі

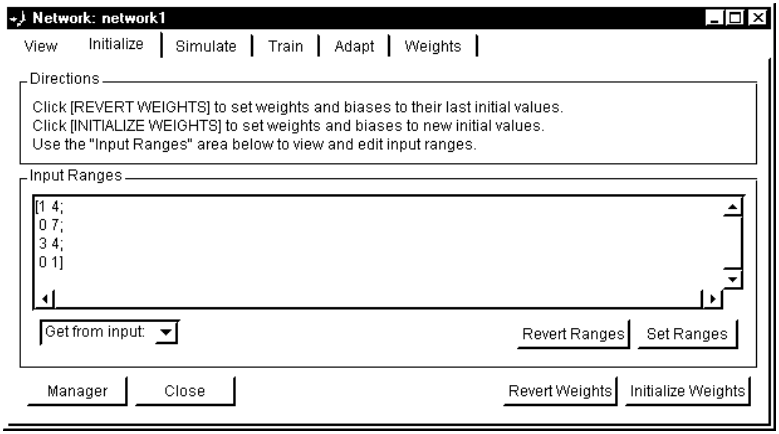


Рис. 8.7 - Форма Network: закладка Initialize

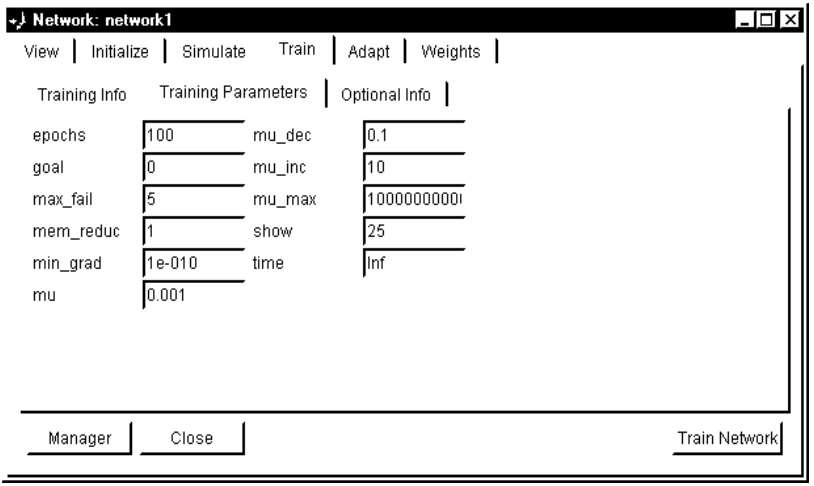


Рис. 8.8 - Форма Network: закладка Train

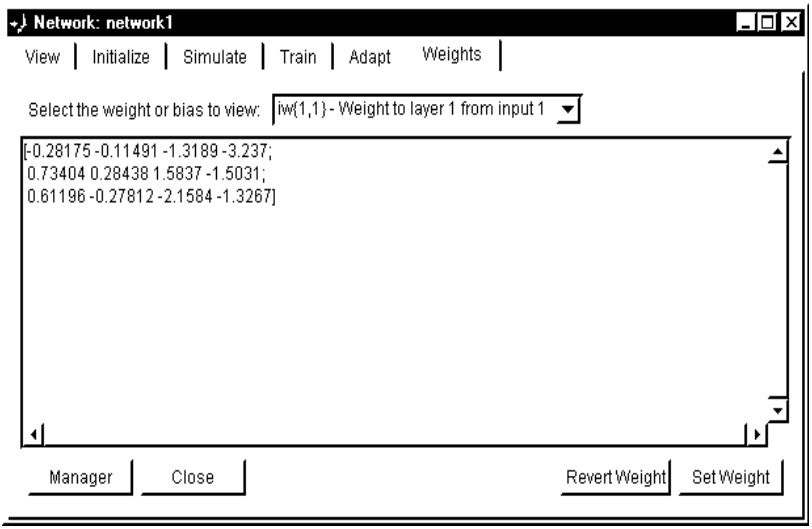


Рис. 8.9 - Форма Network: закладка Weights

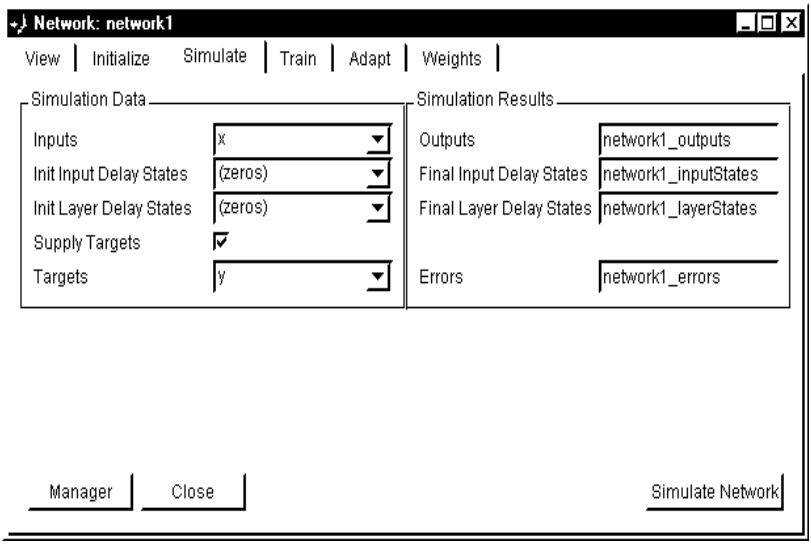


Рис. 8.10 - Форма Network: закладка Simulate

Закладка Initialize ("Ініціалізація") дозволяє задати межі, в яких змінюються вхідні дані та розрахувати на їхній основі початкові значення ваг мережі.

Ініціалізована мережа може бути навчена за допомогою закладки Train ("Тренування, навчання"). Серед параметрів навчання, доступних на цій закладці обов'язково необхідно задати: goal - максимально припустиме значення цільової функції, epochs - максимальна припустима кількість циклів навчання мережі, show - шаг виводу на екран інформації про навчання мережі, задається в циклах навчання. В процесі навчання середовище MATLAB будує графік зміни значення цільової функції по епохах - циклах навчання (рис. 8.11).

Ваги мережі, що була навчена, можна переглянути, використовуючи закладку Weights ("Ваги").

Після того, як мережа навчилася, її можна використовувати для розпізнавання за допомогою закладки Simulate ("Моделювання").

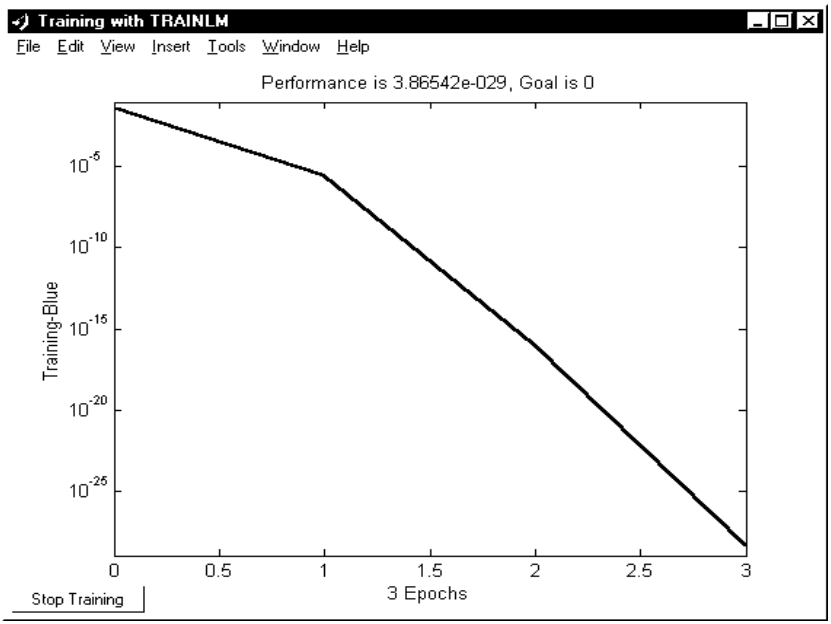


Рис. 8.11 - Графік зміни значення
цільової функції в процесі навчання

8.2 Оптимізація і нейронні мережі в автоматизованій системі «Діагностика»

Автоматизована система (АС) **«Діагностика»** розроблена з метою автоматизації процесу технічної і медичної діагностики і представляє собою програмний комплекс із відкритою архітектурою.

Процес діагностики в АС «Діагностика» розбитий на окремі етапи: планування експерименту, відбір інформативних ознак, факторний аналіз, побудова рішаючого (чи апроксимуючого) правила, оптимізація рішаючого (апроксимуючого) правила.

Для підвищення ефективності комплексу кожний етап діагностики можна виконати різними методами, при розумному

підборі яких можна досягти високої точності діагностики та (або) мінімальних часових витрат.

Особливістю комплексу є його модульність і відкритість, що дозволяє додавати в нього нові методи діагностики.

Вхідні дані програми (навчальні вибірки й інформація про об'єкти, що аналізуються) зберігаються у файлах різних форматів (текстовому, таблиць Excel, баз даних Database, Paradox, Microsoft Access і ін.). Це дозволяє зняти обмеження розміру файлів даних, стандартизувати процес введення даних, дозволяє методами SQL запитів проводити вибір даних за заданими параметрами, а також дає можливість використовувати, при необхідності, дані, підготовлені в інших програмах.

Підсистема оптимізації АС «Діагностика» представляє собою програмний модуль, що використовується підсистемами, які входять до складу АС. Модуль оптимізації реалізований мовою Inprise (Borland) Delphi і містить процедури, що реалізують наступні алгоритми:

- лінійного програмування;
- одновимірної оптимізації:
    - розподілу інтервалу навпіл;
    - Ньютона-Рафсона;
    - Фібоначчі;
    - золотого перетину;
    - квадратичної апроксимації;
    - кубічної апроксимації з використанням похідних;
    - середньої точки;
- багатовимірної оптимізації:
    - градієнтний метод Коші;
    - Ньютона;
    - спряжених градієнтів Флетчера-Рівса;
    - спряжених градієнтів Полака-Ріб‘єра;
    - партан-метод;
    - Левенберга-Марквардта;
    - Давідона-Флетчера-Пауелла;
    - узагальнений градієнтний ;
    - Хука-Дживса;
    - Нелдера-Міда;

- штрафних функцій.

Процедури, що входять до складу підсистеми оптимізації можуть бути викликані користувачем за допомогою модуля інтерфейсу.

У процесі роботи процедур оптимізації генерується звіт, що зберігається у внутрішній змінній модуля оптимізації Report. Цей звіт доступний як іншим підсистемам, що використовують модуль оптимізації, так і користувачу при роботі через модуль інтерфейсу.

Підсистема нейромережевої діагностики АС «Діагностика» дозволяє вирішувати задачі діагностики і прогнозування на основі НМ.

Підсистема нейромережової діагностики складається з трьох Windows-додатків. Такий поділ на програми є обґрунтованим, оскільки кожна програма може бути використана окремо, що дозволяє істотно знизити вимоги програмного комплексу до ресурсів ЕОМ.

До складу нейромережевої підсистеми входять програми:

- візуальний конструктор НМ – редактор програм вбудованою мовою макросів;
- емулятор-відлагоджувач програм вбудованою мовою макросів, що дозволяє в режимі інтерпретатора по кроках переглянути результати виконання команд;
- програма побудови і друку звітів, графіків, створення, перетворення і візуалізації даних.

У підсистемі нейромережевої діагностики реалізовані наступні алгоритми:

- алгоритм навчання одношарового персептрона Уідроу-Хоффа;
- градієнтні алгоритми навчання багатошарових нейронних мереж (алгоритм зворотного поширення помилки першого порядку, алгоритм Ньютона, алгоритми спряжених градієнтів Флетчера-Рівса і Полака-Ріб‘єра, алгоритм Левенберга-Марквардта);
- алгоритми оцінки інформативності і відбору інформативних ознак для персептронів;
- алгоритми навчання радіально-базисних НМ;
- алгоритми навчання НМ Хопфілда;
- алгоритми навчання машини Больцмана;

- алгоритми формування карти ознак самоорганізації Кохонена (КОСК), планування експериментів на основі КОСК, а також навчання системи КОСК-АЗП;
- алгоритми навчання НМ LVQ.

Підсистема нейромережевої діагностики має відкриту архітектуру і дозволяє користувачу не тільки вибирати запрограмовані функції і параметри, але і самостійно задавати їх. Наприклад, користувач має можливість задавати власні функції активації формальних нейронів.

Програма «емулятор-відлагоджувач» функціонує під управлінням програм на вбудованій мові макросів. Вбудована мова макросів призначена для створення макросів, що дозволяють зберігати інформацію про структуру розроблених користувачем моделей НМ, а також для управління процесом роботи нейромережевої підсистеми діагностичного програмного комплексу.

Синтаксис цієї мови достатньо простий та схожий на синтаксис мови Бейсик, що робить його доступним для користувачів. Алфавіт мови включає у себе літери латинського алфавіту (заголовні і рядкові), цифри 0-9, а також знаки “,”, “.”, “%”, “;”, “+”, “_”, “-”. Мова містить відносно невелику кількість макросів.

Опис основних макросів наведено у табл. 8.1. Параметри макросів, що задаються користувачем, набрано курсивом.

Завдяки модулю інтерфейсу АС «Діагностика» підсистема нейромережевої діагностики може спільно працювати з програмним комплексом MATLAB 5.2 фірми MathWorks, при цьому досягається сумісність за даними, а також можливо здійснювати MATLAB-вставки у середину програми мовою макросів підсистеми нейромережевої діагностики. Для виконання MATLAB-вставок у програмах вбудованою мовою макросів необхідна наявність на ЕОМ установленої програми MATLAB 5.2.

Таблиця 8.1 – Основні макроси підсистеми нейромережевої діагностики

Формат макросу	**Призначення**
SET_DATA_SOURCE *путь та ім'я файлу*	Установити джерело даних
SET_DATA_DESTINATION *путь та ім'я файлу*	Установити приймач даних
SET_*внутрішня змінна значення*	Присвоїти внутрішній змінній значення
MATLAB_EXEC *путь та ім'я MATLAB-файлу*	Запустити на виконання MATLAB-файл
WIN_EXEC *путь та ім'я Windows-додатку*	Запустити на виконання Windows-додаток
; або %	коментар (діє у межах даної строки)
STOP або END	Зупинення - закінчення програми
*модель*_CREATE	Створити нейромережу
модель _DATA_RANDOM	Створити навчальну вибірку із псевдовипадкових чисел
модель _DATA_LOAD	Завантажити навчальну вибірку
модель _TRAIN	Навчити нейромережу
модель _SIM	Емулювати роботу нейромережі
модель _FREE	Знищити нейромережу

ВИСНОВКИ

Методи й алгоритми оптимізації, розглянуті в цій книзі, складають теоретичний базис для розробки програмного забезпечення в області математичного моделювання і його застосування в інженерно-технічних задачах.

Наведемо основні рекомендації з використання розглянутих методів.

Лінійне програмування доцільно застосовувати в задачах планування економіки та техніки.

Одновимірні методи нелінійної оптимізації самі по собі мають дуже вузьку сферу застосування, однак як складова частина систем діагностики і прогнозування вони знаходять широке застосування при пошуку оптимальних значень параметрів алгоритмів побудови математичних моделей.

Багатовимірні безградієнтні методи оптимізації не вимагають обчислення похідних і не накладають вимог безперервності на похідну, однак при цьому характеризуються дуже повільною швидкістю збіжності. Тому дані методи варто застосовувати при вирішенні тих задач, де обчислення похідних ускладнене або неможливе.

Багатовимірні градієнтні методи оптимізації характеризуються більш високою швидкістю збіжності, ніж безградієнтні методи, однак вимагають безперервності функції і її похідної. Ці методи доцільно застосовувати при вирішенні задач багатовимірної нелінійної апроксимації залежностей, заданих набором точок, а також у задачах навчання НМ.

У цілому, застосування методів числової оптимізації при побудові математичних моделей складних об'єктів і процесів є необхідною умовою для забезпечення і підвищення їхньої надійності, стійкості та працездатності.

ЛІТЕРАТУРА

1. Boseniuk T., van der Meer M., Poschel T. A Multiprocessor system for high speed simulation of neural networks // Journal of New Generation Computer Systems, 1990, № 3, P. 65-71.
2. Dubrovin V., Morshchavka S., Piza D., Subbotin S. Plant recognition by genetic algorithm based back-propagation // Proceedings, Remote Sensing 2000: from spectroscopy to remotely sensed spectra. Soil Science Society of America, Bouyocos Conference, Corpus Christi, Texas, October 22-25, 2000.-P. 47- 54.
3. Dubrovin V., Subbotin S. The Quick Method of Neural Network Training // Proceedings of International Conference on Modern Problems of Telecommunications, Computer Science and Engineers Training TCSET'2002.-Lviv-Slavsko: NU"Lvivska Politechnica", P. 266-267.
4. Dubrovin V., Subbotin S., Morshchavka S., Piza D. The plant recognition on remote sensing results by the feed-forward neural networks // Smart Engineering System Design, 2001, № 3, P. 251-256.
5. Dubrovin V.I., Subbotin S.A., Morshchavka S.V., Piza D.M. The plant recognition on remote sensing results by the feed-forward neural networks // Smart Engeneering Systems Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Data Mining, and Complex Systems, ANNIE 2000: the 10-th Anniversary edition / ed. C. H. Dagli et al.-Missouri-Rolla:ASME Press, 2000, vol.10, P. 697-701.
6. Neural Network Toolbox User Guide / Beale M., Demuth H. - Natick: Mathworks, 1997. - 700 p.
7. Абу-Мустафа Я.С., Псалтис Д. Оптические нейронно-сетевые компьютеры // В мире науки, 1987, № 5, С. 42-50.
8. Аведьян Э.Д. Алгоритмы настройки многослойных нейронных сетей // Автоматика и телемеханика. – 1995. - № 4. - С. 106-118.
9. Адаменко В.А., Басов Ю.Ф., Дубровин В.И., Субботин С.А. Нейросетевая обработка сигналов в задачах диагностики газотурбинных авиадвигателей // Цифровая обработка сигналов и ее

применение: 3-я Международная конференция и выставка.-М.:РНТОРЭС им. А.С. Попова, 2000.-С. 40-45.

10. Адаменко В.А., Дубровин В.И., Жеманюк П.Д., Субботин С.А. Диагностика лопаток авиадвигателей по спектрам свободных затухающих колебаний после ударного возбуждения // Автоматика-2000. Міжнародна конференція з автоматичного управління, Львів, 11-15 вересня 2000 / Праці у 7-ми томах.-Т. 5.-Львів: Державний НДІ інформаційної інфраструктури, 2000.- С. 7-13.
11. Адаменко В.А., Дубровин В.И., Жеманюк П.Д., Субботин С.А. Диагностика усталостных трещин в деталях газотурбинных авиадвигателей // Надійність машин та прогнозування їх ресурсу / Доповіді міжнародної науково-технічної конференції.-В 2-х томах. Том 1.-Івано-Франківськ: ІФДТУНГ-Факел, 2000.-С. 151 – 158.
12. Адаменко В.А., Дубровин В.И., Субботин С.А. Диагностика лопаток авиадвигателей по спектрам затухающих колебаний после ударного возбуждения на основе нейронных сетей прямого распространения // Нові матеріали і технології в металургії та машинобудуванні, 2000, № 1, С. 91-96.
13. Адаменко В.А., Дубровин В.И., Субботин С.А. Нейросетевая диагностика деталей энергетических установок, работающих при циклических нагрузках // Новые технологии, методы обработки и упрочнения деталей энергетических установок: Тез. докл. Международной конференции “Новые технологии, методы обработки и упрочнения деталей энергетических установок” / Отв. ред. В.К. Яценко.-Запорожье: ЗГТУ, 2000.-С. 4-6.
14. Банди Б. Методы оптимизации. Вводный курс: Пер с англ..-М.: Радио и связь, 1988.-128 с.
15. Бовель Е.И., Паршин В.В. Нейронные сети в системах автоматического распознавания речи // Зарубежная радиоэлектроника. - 1998. - №4. - С. 50-57.
16. Богуслаев А.В., Дубровин В.И., Субботин С.А., Яценко В.К. Моделирование коэффициента ультразвукового упрочнения деталей авиадвигателей // Нові матеріали і технології в металургії та машинобудуванні, 2001, №2, С. 87-90.
17. Богуслаев А.В., Дубровин В.И., Субботин С.А., Яценко В.К. Модель коэффициента упрочнения деталей ГТД // Технологические системы, 2001, № 3.-С.42-45.

18. Бондаров П.А., Проскурин Р.А. Обучение нейросети на базе шарового метода оптимизации Ньютона // Информационные технологии,1998, № 7
19. Борисов А.Н.,Павлов В.А. Прогнозирование непрерывных функций с использованием нейронных сетей // Автоматика и ВТ, 1995, № 5 С.39-50.
20. Ван Кэмп Д. Нейроны для компьютеров //В мире науки, 1992, № 11-12.
21. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация. М.: Мир, 1985.-509 с.
22. Горбань А.Н. Обучение нейронных сетей. М.: изд. СССР-США СП "ParaGraph", 1990.-160 с.
23. Горбань А.Н., Россиев Д.А. Нейронные сети на персональном компьютере // Новосибирск: Наука, 1996.- 276 с.
24. Демиденко Е.З. Линейная и нелинейная регрессия.- М.: Финансы и статистика, 1981.- 302 с.
25. Деннис Дж. мл., Шнабель Р. Численные методы безусловной оптимизации и решения нелинейных уравнений. М.: Мир, 1988.- 440 с.
26. Дубровин В.И. Идентификация и оптимизация сложных технических процессов и объектов.-Запорожье: ЗГТУ, 1997.- 92 с.
27. Дубровин В.И. Приведение задачи минимизации коэффициента детонации ЛПМ к задаче линейного программирования // Сучасні проблеми автоматизованої розробки і виробництва радіоелектронних засобів та підготовки інженерних кадрів / Матеріали міжнародної науково-технічної конференції, присвяченої 150-річчю Державного університету «Львівська політехніка», 1 частина - Львів: Державний університет «Львівська політехніка», 1994, С.44-46.
28. Дубровин В.И., Колесник Н.В., Субботин С.А. Нейросетевая модель зависимости спортивного потенциала от данных биохимического контроля тренировочных нагрузок // Вестник новых медицинских технологий, 2000, № 3-4, С. 39.
29. Дубровин В.И., Колесник Н.В., Субботин С.А. Нейросетевое прогнозирование состояния функционально-метаболических систем у спортсменов высшей квалификации // Моделирование неравновесных систем-2000: Материалы III Всероссийского се-

минара / Под .ред. А.Н. Горбаня.-Красноярск: ИПЦ КГТУ, 2000.-С. 82-83.

30. Дубровин В.И., Колесник Н.В., Субботин С.А. Прогнозирование состояния спортивной формы на основе нейросетевой классификации данных биохимического контроля тренировочных нагрузок // Физика и радиоэлектроника в медицине и экологии: Труды 4-й межд. науч.-тех. конф.- Владимир: Изд-во Института оценки природных ресурсов, 2000 / В 2-х частях. Часть 2.-С. 35-40.
31. Дубровин В.И., Лютый А.В., Пономарчук В.И. Моделирование и оптимизация лентопротяжных механизмов (анализ возмущающих факторов и постановка задачи) // Досвід розробки та застосування приладо-технологічних САПР мікроелектроніки / Тези доповідей 3-ої науково-технічної конференції, частина 2,- Львів: ВО «Полярон», 1995, С.148-149.
32. Дубровин В.И., Лютый А.В., Пономарчук В.И. Решение задачи минимизации коэффициента детонации ЛПМ с использованием регрессионных моделей // Проблемы и перспективы международного сотрудничества в системе высшего образования и науки / Материалы международной научно-технической конференции.-Владимир: ВлГТУ, 1996, С.105-106.
33. Дубровин В.И., Морщавка С.В., Пиза Д. М., Субботин С.А. Нейросетевая идентификация объектов по спектрам // Труды международной конференции “Идентификация систем и задачи управления” SICPRO’ 2000.-М.: ИПУ РАН, 2000.-С. 1190-1204 (CD-ROM).
34. Дубровин В.И., Морщавка С.В., Пиза Д. М., Субботин С.А. Применение радиально-базисных нейронных сетей для обработки данных дистанционного зондирования растений // Цифровая обработка сигналов и ее применение: 3-я Международная конференция и выставка.-М.:РНТОРЭС им. А.С. Попова, 2000.-С.48-53.
35. Дубровин В.И., Морщавка С.В., Пиза Д. М., Субботин С.А. Распознавание растений по результатам дистанционного зондирования на основе многослойных нейронных сетей // Математичні машини і системи, 2000, № 2-3, С. 113-119.
36. Дубровин В.И., Морщавка С.В., Пиза Д.М., Субботин С.А. Нейросетевая классификация растений по коэффициентам спектральной яркости // Нейроинформатика и ее приложения // Ма-

териалы VIII Всероссийского семинара 6-8 октября 2000 года, Красноярск / Под общей ред. А.Н. Горбаня; Отв. за вып. Г.М. Цыбульский.-Красноярск:ИПЦ КГТУ, 2000.- С. 61-62.

37. Дубровин В.И., Морщавка С.В., Субботин С.А., Пиза Д.М. Нейросетевая идентификация растений // Современные проблемы радиоэлектроники / Сборник научных трудов / Под ред. А.В. Сарафанова.-Красноярск: ИПЦ КГТУ, 2000.-С. 198.
38. Дубровин В.И., Пономарчук В.И. Исследование лентопротяжных механизмов с использованием регрессионных моделей // Измерительная и вычислительная техника в технологических процессах и конверсии производства / Тезисы докладов Второй научно-технической конференции.- Хмельницкий: Издательство «Поділля», 1993, С.93.
39. Дубровин В.И., Субботин С.А. Алгоритм ускорения процесса обучения нейронных сетей // Научно-технический калейдоскоп. Серия "Приборостроение, радиотехника и информационные технологии" / Под. ред. Л.И. Волгина. - Ульяновск: Научно-производственный журнал, 2001, № 2.-С. 49-55.
40. Дубровин В.И., Субботин С.А. Алгоритм ускоренного обучения нейросетей // Нейроинформатика и ее приложения / Материалы IX Всероссийского семинара, 5-7 октября 2001 г. / Под ред. А.Н.Горбаня. Отв. за выпуск Г.М.Цибульский.- Красноярск:КГТУ, 2001.-С. 63-64.
41. Дубровин В.И., Субботин С.А. Диагностика на основе эвристических алгоритмов в условиях ограниченного объема обучающей выборки // Proceedings of International conference “Soft computing and measurement” SCM-2000, 27-30 June 2000.-Saint-Petersburg: Saint-Petersburg State Electrotechnical University (LETI), 2000.-CD-ROM
42. Дубровин В.И., Субботин С.А. Индивидуальное прогнозирование надежности изделий электронной техники на основе нейронных сетей // Труды VII Всероссийской конференции “Нейрокомпьютеры и их применение” НКП-2001 с международным участием , Москва, 14-16 февраля, 2001 г.- М.: ИПУ РАН, С. 228-231.
43. Дубровин В.И., Субботин С.А. Интегрированные многоклассификаторные нейросетевые системы диагностики // Електротехніка та електроенергетика, 2001, № 1, С. 38-43.

44. Дубровин В.И., Субботин С.А. Моделирование акустических характеристик магнитных головок // Труды III Всероссийской научно-технической конференции "Нейроинформатика-2001".-М.: МИФИ, 2001.-Ч.2, С.89-96.
45. Дубровин В.И., Субботин С.А. Моделирование лентопротяжного механизма на основе многослойной нейронной сети // Математическое образование на Алтае / Материалы первой краевой конференции по математическому образованию на Алтае. – Барнаул, БГПУ, 2000, С.35-36.
46. Дубровин В.И., Субботин С.А. Моделирование лентопротяжного механизма на основе многослойной нейронной сети прямого распространения // Современные проблемы радиоэлектроники / Сборник научных трудов / Под ред. А.В. Сарафанова.-Красноярск: ИПЦ КГТУ, 2000.-С. 257-258.
47. Дубровин В.И., Субботин С.А. Нейросетевая диагностика в управлении качеством // Управление в технических системах – XXI век: сборник научных трудов III Международной научно-технической конференции.-Ковров: КГТА, 2000.-С. 136-138.
48. Дубровин В.И., Субботин С.А. Нейросетевая диагностика газотурбинных лопаток // Оптические, радиоволновые и тепловые методы и средства контроля качества материалов, промышленных изделий и окружающей среды / Тезисы докладов VIII международной научно-технической конференции.-Ульяновск: УлГТУ, 2000.-С.121-124.
49. Дубровин В.И., Субботин С.А. Нейросетевая диагностика лопаток энергетических установок // Датчики и преобразователи информации систем измерения, контроля и управления / Сборник материалов XII научно-технической конференции с участием зарубежных специалистов. Под ред. проф. В.Н. Азарова. М.: МГИЭМ, 2000.-С. 240-242.
50. Дубровин В.И., Субботин С.А. Нейросетевая интерпретация алгоритма многомерной классификации // Труды III Всероссийской научно-технической конференции "Нейроинформатика-2001".-М.: МИФИ, 2001.-Ч.1 С.38-46.
51. Дубровин В.И., Субботин С.А. Нейросетевая модель зависимости акустических характеристик магнитных головок от параметров сендастового сердечника // Вестник Хакасского государственного университета им. Н.Ф. Катанова. Вып. 4. Серия 1: Ин-

форматика.-Абакан: Издательство ХГУ им. Н.Ф. Катанова, 2001.-С. 25-28.

52. Дубровин В.И., Субботин С.А. Нейросетевая подсистема диагностического программного комплекса // Нейрокомпьютеры: разработка и применение, 2001, №2, С. 55-62.
53. Дубровин В.И., Субботин С.А. Нейросетевое моделирование и оценка параметров нелинейных регрессий // Нейрокомпьютеры и их применение / Сборник докладов 6-ой Всероссийской конференции, Москва 16-18 февраля 2000.-М.:Издательское предприятие журнала “Радиотехника”, 2000.- С. 118-120.
54. Дубровин В.И., Субботин С.А. Нейросетевое моделирование лентопротяжного механизма // Труды VII Всероссийской конференции “Нейрокомпьютеры и их применение” НКП-2001 с международным участием, Москва, 14-16 февраля, 2001 г.- М.: ИПУ РАН, С. 153-156.
55. Дубровин В.И., Субботин С.А. Обобщенный градиентный алгоритм обучения многослойных нейронных сетей //Електротехніка та електроенергетика, 2000, № 1, С. 17-22.
56. Дубровин В.И., Субботин С.А. Онлайновые методы управления качеством: гибридная диагностика на основе нейронных сетей // Радіоелектроніка. Інформатика. Управління, 2001, № 1, С. 158 –163.
57. Дубровин В.И., Субботин С.А. Подсистема нейросетевой диагностики // Нейроинформатика и ее приложения: Материалы VIII Всероссийского семинара 6-8 октября 2000 года, Красноярск / Под общей ред. А.Н. Горбаня; Отв. за вып. Г.М. Цыбульский.-Красноярск:ИПЦ КГТУ, 2000.- С. 63-64.
58. Дубровин В.И., Субботин С.А. Прогнозирование отказов деталй ГТД в процессе эксплуатации // Моделирование неравновесных систем-2000: Материалы III Всероссийского семинара / Под .ред. А.Н. Горбаня.-Красноярск: ИПЦ КГТУ, 2000.-С. 84.
59. Дубровин В.И., Субботин С.А. Программный комплекс нейросетевой диагностики // Программные продукты и системы, 2000, № 3, С. 21-23
60. Дубровин В.И., Субботин С.А. Следящий алгоритм обучения нейронных сетей // Вимірювальна та обчислювальна техніка в технологічних процесах: Збірник наукових праць.-Хмельницький: ТУП, 2001.-С. 88-91.

61. Дубровин В.И., Субботин С.А., Адаменко В.А., Басов Ю.Ф. Диагностика авиадвигателей на основе нейросетевого гибридного классификатора // Труды Международной конференции "Параллельные вычисления и задачи управления" (PACO'2001).-М.: ИПУ РАН, 2001.-Т.5, С. 53-73 (CD-ROM, ISBN 5-201-09559-3).
62. Дубровин В.И., Субботин С.А., Адаменко В.А., Басов Ю.Ф. Построение гибридных систем диагностики деталей энергетических установок на основе нейронных сетей // Modelling and Analysis of Safety, Risk and Quality in Computer Systems / Proceedings of the International Scientific School MA SRQ - 2001.-СПб.: ООО НПО "Омега", 2001.-С. 236-239.
63. Дубровин В.И., Субботин С.А., Согорин А.А. Радиальнобазисные нейронные сети в задачах технической диагностики // Інтернет, освіта, наука, друга міжнарожна конференція ІОН-2000, 10-12 жовтня. Збірник матеріалів конференції.-Вінниця:Універсум-Вінниця, 2000.-С. 303-306.
64. Дубровин В.И., Субботин С.А., Яценко В.К. Методика оценки коэффициента упрочнения деталей газотурбинных авиадвигателей // Техническая диагностика и неразрушающий контроль, 2001, № 3.-С. 42-45.
65. Дубровин В.И., Субботин С.А., Яценко В.К. Нейросетевая методика расчета коэффициента упрочнения деталей авиадвигателей // Техника машиностроения, 2001, № 4, С. 46-52.
66. Дубровин В.И., Субботин С.А., Яценко В.К. Построение нейросетевой модели коэффициента упрочнения при обкатке деталей энергетических установок // Електротехніка та електроенергетика, 2001, № 2, С. 38-42.
67. Дубровин В.И., Субботин С.А., Яценко В.К. Прогнозирование запаса прочности деталей авиадвигателей // Моделирование неравновесных систем - 2001 / Материалы IV Всероссийского семинара - Красноярск: ИВМ СО РАН, 2001.-С. 44-45.
68. Дубровін В.І., Субботін С.О. Вибір функцій активації формального нейрона та дослідження їх впливу на якість навчання нейронних мереж // Вісник Національного університету “Львівська політехніка” “Комп’ютерні системи проектування. Теорія і практика”, № 398, 2000.-С.12-17.

69. Итоги науки и техники. Сер. "Физ. и Матем. модели нейронных сетей" /Под ред. А.А.Веденова. - М.: Изд-во ВИНИТИ, 1990-92 - Т. 1-5.
70. Кабанов В.А., Хробостов Д.А. Нейроподобные и клеточные структуры в системах управления // Автоматика и вычислительная техника.-1996, № 5.
71. Картавцев В.В. Нейронная сеть предсказывает курс доллара // Компьютеры + программы - 1993 - N 6(7) - С. 10-13.
72. Корнеев В.В. Параллельные вычислительные системы .-М.: Нолидж, 1999.-320 с.
73. Кривенко В.И., Евченко Л.Н, Субботин С.А. Нейросетевое моделирование суммарного показателя качества жизни больных хроническим обструктивным бронхитом в ассоциации с клиническими особенностями течения заболевания // Вестник новых медицинских технологий, 2001, Т. VIII, № 4, С. 7-10.
74. Кривенко В.И., Евченко Л.Н., Субботин С.А. Нейросетевое моделирование показателя качества жизни для диспансерного учета пациентов // Моделирование неравновесных систем - 2001 / Материалы IV Всероссийского семинара - Красноярск: ИВМ СО РАН, 2001.-С. 79-80.
75. Круг Г.К., Кабанов В.А., Фомин Г.А., Фомина Е.С. Планирование эксперимента в задачах нелинейного оценивания и распознавания образов.-М.: Наука, 1981.-172 с.
76. Круглов В.В., Борисов В.В. Искусственные нейронные сети.-М.: Горячая линия - Телеком, 2001.-382 с.
77. Морщавка С.В., Субботин С.А., Дубровин В.И., Пиза Д.М. Нейросетевая классификация растений по результатам дистанционного зондирования // 5-й Международный молодежный форум “Радиоэлектроника и молодежь в XXI веке”. Сб. научных трудов. Ч. 2.- Харьков: ХТУРЭ, 2001.-С. 324-325.
78. Нейроинформатика / А.Н.Горбань, В.Л.Дунин-Барковский, А.Н.Кирдин, Е.М.Миркес, А.Ю.Новоходько, Д.А.Россиев, С.А.Терехов, М.Ю.Сенашова, В.Г.Царегородцев. Новосибирск: Наука, Сибирская издательская фирма РАН, 1998.- 296 с.
79. Нейрокомпьютеры и интеллектуальные роботы / Амосов Н.М., Байдык Т.Н., Гольцев А.Д. и др.; под ред. Амосова Н.М. - Киев: Наукова думка, 1991. - 272 с.

80. Пшеничный Б.Н., Данилин Ю.М. Численные методы в экстремальных задачах. М.:Наука, 1975.- 319 с.
81. Реклейтис Г., Рейвиндран А., Рэгсдел К. Оптимизация в технике: В 2-х кн. Пер с англ.– М.:Мир, 1986.-Кн. 1: 349 с., Кн. 2: 320 с.
82. Субботін С.О. Нейронні мережі керують якістю // Пульсар, 1999, № 12, С. 8-10.
83. Терехов В.А., Ефимов Д.В., Тюкин И.Ю., Антонов В.Н. Нейросетевые системы управления.-СПб.: Изд-во С.-Петербургского ун-та, 1999.-265 с.
84. Трикоз Д.В. Нейронные сети: как это делается?// Компьютеры + программы, 1993, № 4, С. 14-20.
85. Тэнк Д.У., Хопфилд Д.Д. Коллективные вычисления в нейроноподобных электронных схемах //В мире науки, 1988, № 2, С.44-53.
86. Уоссермен Ф. Нейрокомпьютерная техника.- М.: Мир, 1992.
87. Химмельблау Д. Прикладное нелинейное программирование. М.: Мир, 1975.-534 с.
88. Хинтон Дж. Е. Как обучаются нейронные сети // В мире науки, 1992, № 11-12, С. 103-107.
89. Хинтон Дж.Е. Обучение в параллельных сетях / Реальность и прогнозы искусственного интеллекта.- М.: Мир, 1987.- С. 124-136.
90. Цыпкин Я.З. Основы теории обучающихся систем. М.: Наука, 1970.-252 с.
91. Дубровин В.И., Субботин С.А., Богуслаев А.В., Яценко В.К. Интеллектуальные средства диагностики и прогнозирования надежности авиадвигателей.-Запорожье: ОАО "Мотор Сич", 2003.- 279 с.

ДОДАТОК. ПУТІВНИК ЗА СПИСКОМ ЛІТЕРАТУРИ

тема	**номер літературного джерела**
Лінійне програмування	**21, 26, 27, 81**
Одновимірні методи нелінійної оптимізації	**14, 21, 81, 87**
Багатовимірні безградієнтні методи нелінійної оптимізації	**14, 21, 24, 25, 81, 87**
Багатовимірні градієнтні методи нелінійної оптимізації	**1-6, 8-15, 18, 21, 22, 25, 33-37, 41, 44, 55, 75, 80, 81, 90, 91**
Нейронні мережі	**1-13, 15-20, 22, 23, 33, 37, 39-44, 48-50, 53, 56, 60-63, 68-72, 78, 79, 83, 84-86, 88, 89, 91**
Апроксимація	**24, 25, 32, 38, 53, 75, 90, 91**
Програмні засоби чисельної оптимізації	**52, 57, 59**
Застосування методів оптимізації і нейронних мереж у науці та техніці	**2, 4, 5, 9-13, 15-17, 26-31, 33, 37, 42, 44-49, 51, 54, 58, 64-67, 71-74, 77, 82, 91**

Електронні версії певних робіт, що наведені у списку літератури, розміщені на веб-сайтах в Інтернет за адресами: http://csit.narod.ru/people/Subbotin.htm та http://www.zntu.edu.ua/RIC.

Valeriy Dubrovin and Sergey Subbotin

Methods of Optimization and TheirApplications in Neural Networks Learning

SUMMARY

The methods of numerical optimization are widely used in practice and examples of application of optimization methods at problem solving of artificial neural networks learning, and also at problem solving of many-dimensional non-linear approximation are considered in the book.

The book is intended for the students of speciality "Software of Automated Systems" and also can be used by the engineers, post-graduate students and students of various specialities for acquisition of practical skills of the solution of optimization problems.

Introduction contains definition of a sphere of optimization methods problems and review of areas of their application.

Chapter 1. Linear programming contains a statement of a problem of linear programming, description of it's properties, and also contains a simplex method for solution of a linear programming problem.

Chapter 2. Non-linear programming contains a statement of a common problem of non-linear programming and description of a classification of methods of non-linear programming.

Chapter 3. One-dimensional methods of non-linear optimization contains review of methods of direct searching (a division of interval on halves method, a Fibonacci method, a method of «golden section», a method of even searching), methods of searching with use of polynomial approximation (method of estimation with use of quadratic approximation and method of a series estimation with use of quadratic approximation), and also methods of searching with use of derivatives (a Newton - Rafson method, a method of mean point, a method of

intersecting (method of chords) and a method of searching with use of cubic approximation).

Chapter 4. Many-dimensional methods of non-linear optimization without gradients contains description of a Nelder-Meed method, a Hook-Jeeves method, and also a conjugate gradient method of Powell.

Chapter 5. Many-dimensional gradient methods of non-linear optimization describes a Cauchy's method, a Newton's method, algorithms of conjugate gradients of Fletcher-Reeves and Polack-Ribiere, a Partan-method, a Zautendaick's method, a method of multiparameter searching, quasi-Newton methods, a Levenberg-Marquardt algorithm, a generalized gradient algorithm, and also an integral method of optimization.

Chapter 6. Methods of many-dimensional optimization with limitations contains description of a penal functions method and a method SUMT.

Chapter 7. Methods of optimization in approximation and neural networks learning contains review of basic concepts of the theory of artificial neural networks and formulation of problems of neural networks learning as optimization problems. In this chapter the problems of application of optimization methods for solving of approximation problem are also considered. The algorithm of cluster-regression approximation developed by the authors is proposed for construction and training of neural nretworks.

Chapter 8. Software of numerical optimization is contained a review of main means for solving of optimization problems and modelling of neural networks on the basis of Matlab software package and automated system "Diagnostica".

Conclusions contains recommendations for application of the considered methods of optimization at the solution of practical problems.

Дубровін Валерій Іванович

Субботін Сергій Олександрович

Методи оптимізації та їх застосування в задачах навчання нейронних мереж

Навчальний посібник

Рецензенти: **В.М. Порохня**, доктор технічних наук, професор, завідувач кафедрою математичних методів та інформаційних технологій в економіці Запорізької державної інженерної академії

Л.О. Галкін, доктор технічних наук, професор кафедри твердотільної електроніки та мікроелектроніки Запорізького державного університету

Підписано до друку 8.05.2003 р. Формат 60x84$^{1}/_{16}$. Гарнітура Times New Roman.
Обл.-вид. арк. 3,78, ум. друк. арк. 8.2 Тираж 300 екз. Зам. № 788
Україна, 69063, Запоріжжя, вул. Жуковського, 64, ЗНТУ, Друкарня