**Лабораторна робота №1.**

**Реалізація та дослідження нейронних мереж, що навчаються алгоритмом зворотного поширення похибки**

**Завдання**.

1. Виконати програмну реалізацію нейронної мережі прямозв’язаної нейронної мережі та алгоритму навчання зворотного поширення похибки для трьохшарової нейронної мережі із трьома нейронами першого шару, трьома нейронами - другого і двома нейронами - третього або скористатись прикладними пакетами.

2. Використовуючи дані таблиці початкових даних, написати програмний фрагмент для обчислення 27 значень функції по заданих значеннях аргументів та їх варіаціях з кроком ±1 і знайти їх середнє значення. Розділити вибірку на навчальну (20 прикладів) та тестову (7 прикладів).

3. Подати задані значення прикладів з навчальної вибірки на вхід нейронної мережі. Виходом ** вважати обчислене значення функції за варіантом; виходом  - логічну функцію, що дорівнює “1”, якщо  більше середнього значення, та “0”, якщо менше.

4. Використовуючи контрольні приклади з тестової вибірки оцінити точність алгоритму. Оцінити кількість епох навчання, необхідних для досягнення максимум точності.

5. Дослідити нейронну мережі прямого поширення, побудувавши графіки залежностей:

- величини середньої помилки навченої мережі із заданою кількістю прикладів (20+7) від кількості нейронів у прихованому шарі (в межах від 1 до 10);

- величини середньої помилки навченої мережі від кількості прикладів, що використовуються для навчання) зменшувати від 20 до тих пір, поки помилка не сягне заданого рівня, наприклад 0,5;

6. Використовуючи значення зсуву (bias) порогу спрацювання нейронів прихованого шару, підвищити точність апроксимації, якщо це можливо.

В ході виконання роботи припускається використовувати інструментарій макетів програмування Python або MATLAB. В такому випадку студент повинен пояснити роботу всіх використаних функцій і модулів.

**Початкові дані** знаходяться в наступній таблиці. Номер варіанту відповідає останній цифрі порядкового номеру студента в списку групи.

Вхідні змінні приймають значення в діапазоні , де  - базове значення тої змінної. Наприклад, для варіанту 7 змінна  має базове значення 2. Це означає, що навчальна і тестова вибірки для навчання нейронної мережі за цим варіантом мають містити комбінації, в яких  приймає значення 1, 2 або 3.

Таблиця 1.1 – Початкові дані

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Варіант | Вхідні дані | | | Функція для моделювання |
| x1 | x2 | x3 |
| 1 | 1 | 2 | 3 | d1 = x1^2-x2^2+x3^2 |
| 2 | 5 | 4 | 3 | d1 = sin(x1)+sin(x2)-sin(x3) |
| 3 | 9 | 8 | 7 | d1 = tq(x1)+sin(x2)-sin(x3) |
| 4 | 8 | 5 | 3 | d1 = sin(x1)+sin(x2)-cos(x3) |
| 5 | 5 | 6 | 7 | d1 = tg(x1)+sin(x2)-tg(x3) |
| 6 | 1 | 8 | 7 | d1 = sin(x1)+tg(x2)-tg(x3) |
| 7 | 2 | 5 | 4 | d1 = cos(x1)+cos(x2)-sin(x3) |
| 8 | 5 | 7 | 8 | d1 = ln(|cos(x1)|)+tg(x2)-ctg(x3) |
| 9 | 2 | 3 | 6 | d1 = 2^x1+cos(x2)-sin(x3) |
| 0 | 7 | 4 | 5 | d1 = sin(x2^2)+x1^2-tg(x3) |

**Методичні вказівки**.

Алгоритм зворотного поширення похибки (АЗПП) відноситься до класичних алгоритмів навчання прямо зв’язаної нейронної мережі з учителем. В його основі лежить багаторазовий показ певного набору прикладів, для яких відомі наперед еталонні (визначені експериментально) відповіді, або ж бажані реакції нейронної мережі. Для кожного шару, починаючи з останнього, вихідного, у ході порівняння реакцій мережі  та  - еталонного значення ( номер нейрона у останньому шарі, загалом  шарів).

Похибка довільного нейрона довільного шару обчислюється за виразом

, (1.1)

де  - номер шару;  - активація нейрона того шару;  - кількість нейронів у  шарі.

Для вихідного шару розрахункова формула має вигляд

, (1.2)

де  - похідна від гладкої безперервної функції активації (частіше за все, простий сигмоїд або гіперболічний тангенс).

Величина корекції вагових коефіцієнтів того шару обчислюється за формулою

 (1.3)

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

Наведемо *алгоритм зворотного поширення помилки* для мережі у загальному вигляді, а потім розглянемо його на прикладі.

*Крок* 1. Ініціалізація вагових коефіцієнтів випадковими значеннями. Нормування вхідних значень у відповідності до обраної активаційної функції.

*Крок* 2. Подача на вхід навчального образу і розрахунок виходу мережі.

*Крок* 3. Розрахунок помилки вихідного шару.

*Крок* 4. Розрахунок помилки попереднього шару – повторюється для усіх шарів, аж до першого.

*Крок 5*. Корекція вагових коефіцієнтів останнього шару за формулою

, (1.4)

де  - обчислено за (1.3).

*Крок 6*. Корекція вагових коефіцієнтів усіх попередніх шарів аж до першого.

*Крок* 7. Якщо подані всі вхідні образи, то перехід на крок 8, інакше перехід на крок 2 (подається наступний образ).

*Крок* 8. Якщопомилка апроксимації  менша за наперед визначену величину , то перехід на крок 9, в іншому випадку – рандомізація образів й перехід на крок 2.

*Крок 9*. Подаємо на вхід навченої мережі контрольний вектор ** і знаходимо *прогноз *.

*Крок 10.* Закінчення алгоритму.

*Зауваження*. Образи (приклади) бажано подавати у випадковому порядку, який змінюється при кожному поверненні на початок циклу навчання. Це необхідно аби сам порядок їх надходження не став ще одним віртуальним входом мережі і, відповідно, не вніс зміщення в результат.

**Приклад** Для мережі, що використовує у якості активаційної функції простий сигмоїд ** та складається з трьох нейронів вхідного шару, двох нейронів прихованого шару та двох виходів, розрахункові формули набувають вигляду:

- помилки вихідного шару (для кожного нейрону окремо для кожного образу):

 - перший нейрон,

 - другий нейрон.

- помилки прихованого шару (нейронів три, приклад той самий, що і для шару 2)

 - перший нейрон,

- другий нейрон,

- третій нейрон.

- корекції вагових коефіцієнтів нейронів другого шару (розмір матриці 3\*2)

;

;

;

;

;

.

- корекції вагових коефіцієнтів першого шару (розмір матиці 3\*3)

;

;

;

;

;

;

;

;

.

*Зауваження*. Не існує універсальної поради щодо вибору значень  та , які б забезпечували максимально точне й швидке навчання нейронної мережі. Ці значення обираються емпірично в діапазоні .

Зазвичай хороший результат забезпечує значення коефіцієнту навчання  та коефіцієнту інертності . Вони дозволяють зменшити різкі зміни вагових коефіцієнтів при надходженні кожного нового прикладу.

**Лабораторна робота № 2**

**Реалізація та дослідження радіально-базисних нейронних мереж**

**Завдання**.

1. Здійснити програмну реалізацію радіально-базисної нейронної мережі. При реалізації забезпечити можливість динамічної зміни кількості нейронів прихованого шару та «ширини» активаційних вікон, а також виведення інформації у файл та у вигляді графіків.

2. Використовуючи дані таблиці початкових даних, написати програмний фрагмент, що поступово збільшує кількість опорних функцій (величину прихованого шару мережі) до досягнення оптимальної точності. Кожен новий навчальний приклад, що входить у мережу в якості опорної функції, повинен мати максимальну відмінність від усіх вже використаних.

3. Для кожної кількості нейронів у прихованому шарі створеної мережі підібрати такі радіуси опорних функцій, при яких апроксимація буде найкращою, про що пересвідчитись на контрольних прикладах (решта вибірки).

4. Дослідити можливість встановлення індивідуального радіусу для кожної з радіально-базисних функцій. Порівняти результати такого підходу з кращим рішенням.

В ході виконання роботи припускається використовувати інструментарій макетів програмування Python або MATLAB. В такому випадку студент повинен пояснити роботу всіх використаних функцій і модулів.

**Початкові дані** знаходяться в файлі *lab\_ccc\_2.xls* в робочому каталозі дисципліни на сервері локальної комп’ютерної мережі кафедри. Вхідні змінні *Х1* – *Х3* – спільні для всіх. Вихідна змінна *Yi* обирається за наступним принципом: номер варіанту відповідає останній цифрі порядкового номеру студента в списку групи.

**Методичні рекомендації**

Мережа RBF (radial-based function), як і більшість інших нейромереж, призначена для апроксимації функцій, які задані в неявному вигляді набором шаблонів (навчальних образів). Теорема Ковера (Cover) говорить про те, що для набору образів у лінійному просторі роздільна здатність апроксиматора підвищується з підвищенням ступеню функції апроксиматора.

Доведено, що ймовірність  того, що певна випадково сформована множина з рівно ймовірним представництвом елементів двох класів кількістю  абсолютно розділяється гіперповерхнею того порядку, дорівнює

, (2.1)

де  - біноміальні коефіцієнти.

З (2.19) витікають дві основні особливості мереж RBF у порівнянні з багатошаровим персептроном:

1. прихований шар має бути значно потужнішим за вхідний та вихідний;
2. активаційна функція прихованого шару має бути нелінійною по всіх вимірах з однаковим радіусом .

# **Остання властивість й обумовила назву таких мереж.**

Нейромережа характеризується такими особливостями: має єдиний прихований шар, нейрони прихованого шару мають нелінійну активаційну функцію, синаптичні ваги всіх нейронів прихованого шару дорівнюють одиниці. Розглянемо наступні позначення (рис. 2.4):

 - вектор координат центрів активаційних функцій нейронів прихованого шару;

 - ширина вікна активаційної функції *j* -го нейрона прихованого шару;

 - радіально-симетрична активаційна функція нейрона прихованого шару (див. рис. 2.1);

 - вага зв’язку між тим нейроном прихованого шару та тим нейроном вихідного шару.

Рис. 2.1. Радіально-базисна нейронна мережа.























Відмінність радіально-базисних мереж від розглянутого раніше багатошарового персептрона полягає в наявності лише одного прихованого шару, з яким всі нейрони вхідного шару пов’язані рівними одиничними зв’язками. Якщо в персептроні як прихований шар (яких може бути кілька), так і вихідний є обчислюючими, то в радіально-базисній мережі обчислює лише один прихований шар, вихідний натомість забезпечує лінійну згортку нелінійних функцій. Найголовніша ж відмінність – це аргумент функції активації: у МПР він є скалярним добутком вхідного вектора та вектора вагів, у RBF – відстань від вхідним вектором та центром даного нейрона.

Існує кілька методів навчання радіально-базисної мережі – як ітераційних з рекурсією, так і прямих. Вони можуть використовувати самоорганізацію або вчителя. Далі розглянемо один з методів навчання RBF-мережі, що не містить рекурсії*.* Алгоритм цього методу, що має назву випадкового вибору фіксованих центрів, є наступним:

*Крок* 1. Обрати розмір прихованого шару *H* рівним кількості тренувальних шаблонів *Q* . Синаптичні ваги нейронів прихованого шару прийняти рівними 1.

*Крок 2.* Розмістити центри активаційних функцій нейронів прихованого шару в точках простору вхідних сигналів мережі, які входять до набіру навчальних образів , для всіх .

*Крок* 3. Обрати ширини вікон активаційних функцій нейронів прихованого шару  для всіх  достатньо великими, але так, щоб перетин поверхонь, що визначаються активаційним функціями, був мінімальним в просторі вхідних образів.

*Крок* 4. Визначити ваги нейронів вихідного шару нейронної мережі  для усіх , *.* Для цього пред'явити мережі весь набір навчальних образів. Вихід того нейрона прихованого шару для того образу буде таким:

*.*

Для відповідності виходу мережі еталонному зразку потрібно, аби .

Прирівнявши внутрішній вираз з його ідеальним значенням, можемо отримати систему лінійних алгебраїчних рівнянь, яку зручно записати в матричній формі

**, (2.2)

у якому  - інтерполяційна матриця відстаней того зразку від того центру;  - матриця початкових синаптичних вагових коефіцієнтів;  - матриця заданих образів.

Рішення системи (2.20) досить просто знаходиться на вихідному просторі матричним методом

 (2.3)

Таким чином, ми одержимо шукані синаптичні вагові коефіцієнти, що забезпечують найкраще проходження інтерполяційної поверхні через навчальні образи в просторі вхідних образів.

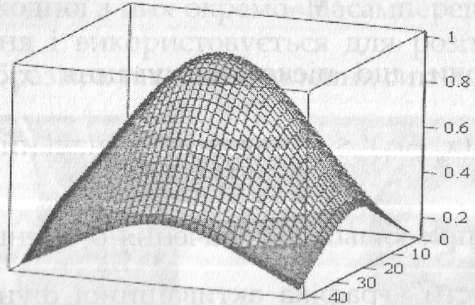


Рис 2.2. Активаційна функція

За відносною простотою побудови й навчання радіально-базисної нейронної мережі ховається цілий ряд її недоліків, які обумовлюють не надто часте застосування цього інструменту в задачах аналізу.

Насамперед, мережа RBF є вкрай чутливою до величини ширини вікон активаційної функції . Двомірний аналог активаційної функції зображено на рис. 2.2. Мало того, що для кожного нейрона  має бути своя, вона ще й має забезпечувати не конфліктність нейронів між собою та, водночас, рівномірність охоплення всього діапазону визначення вхідного масиву даних. Для того аби уникнути «паралічу» мережі через неповну визначеність певних областей вхідних даних і не було великих помилок при апроксимації, вимагаємо виконання наступної умови:



(вибір границь інтервалу достатньо довільний, але його раціональність підтверджена експериментами) або, обмежившись вибором додатного 

.

Враховуючи, що після нормування , одержимо  для всіх навчальних образів. Тоді .

Можна задатися великими значеннями радіусів, але при їх збільшенні зменшується ексцес і ростуть «хвости» графіків активаційної функції, що знову-таки призводить до «паралічу» мережі, але вже в інший бік – неможливості розрізнити образи між собою. Прийнятні результати були отримані в реальних задачах при

 (2.4)

Ще одним недоліком застосування мереж RBF є необхідність виконання певних підготовчих операцій для адекватного навчання і використання навченої мережі RBF. По-перше, треба максимізувати спільну ентропію початкових образів, наприклад, за допомогою відомих методів головних компонент або „вибілювання" входів (про ці методи – в подальших лекціях). Перетворені дані забезпечать якісне та швидке навчання на множині даних мінімальної потужності. Оскільки вказані методи достатньо трудомісткі, необхідно із всієї множини вхідних образів обирати ті, які мають максимальну сумарну попарну евклідову відстань. Наступним кроком має стати нормування.

Неітераційний метод навчання RBF-мережі не завжди є оптимальним методом навчання. Зокрема, коли кількість вхідних образів є великою, застосування градієнтних методів навчання дозволяє зменшити кількість нейронів прихованого шару.

Нарешті, сама мережа RBF має істотне обмеження: якщо багатошарові персептрони забезпечують глобальну апроксимацію нелінійного відображення, радіально-базисні мережі – лише локальну.

Процес функціонування мережі RBF має ще багато особливостей. Але одну варто згадати: за допомогою такої мережі можна розв'язувати лише задачу інтерполяції.

**Лабораторна робота № 3**

**Реалізація та дослідження нейронних мереж з асоціативною пам’яттю**

**Завдання**.

1. Підготувати навчальну вибірку для 5 «чистих класів» об’єктів, що класифікуються. Для цього

- вибрати з трьох слів (Прізвище, Ім’я, по батькові студента) найдовше;

- вибрати з нього 5 різних літер;

- закодувати літери на двійковій матриці 7\*5 (парній варіанти) або 8\*6 (непарні варіанти): темні клітини як 1, світлі – як -1;

- представити кодування у вигляді векторів, зчитуючи матриці по стовпчиках.

2. Підготувати вибірки з зашумленням типу «salt and pepper» (рівноймовіна заміна чорних клітин на білі й навпаки) з ступенем зашумлення від 10 до 60%.

3. Здійснити навчання мережі Хопфілда (варіанти 1-10) або Хеммінга (варіанти 11-20) на початкових даних. Обґрунтувати вибір числа нейронів мережі та алгоритму навчання мережі.

4. Подаючи на вхід випадкові послідовності з зашумлених символів, визначити, при якому ступені зашумлення мережа починає давати більше ніж (10 + 2\*n)% помилкових розпізнавань, де n – номер у списку групи.

5. ДОДАТКОВЕ. Сформувати вхідне зображення, рівновіддалене за відстанню Хеммінга від двох еталонних зображень. Яка реакція при пред'явленні цього зображення?

**Початкові дані** слід згенерувати самостійно використовуючи наступний приклад, що ілюструється рисунком 3.1. Для літери «А» на сітці 8\*5 затемнені і незатемнені клітинки (один з варіантів написання) позначено на фігурі а). щоб утворити з неї навчальний вектор довжиною 8\*5 = 40 знаків, необхідно, пронумерувавши клітинки за принципом рис. 3.1 б), виписати їх по стовпчиках (рис. 3.1 в)).



Рис. 3.1 – Кодування літера для мережа з асоціативної пам’яттю:

а) – відповідність залитих точок і знаків; б) нумерація клітин по стовпцях;

в) вигляд вектору А для навчання мережі.

**Методичні рекомендації**

Мережа Хопфілда. Джон Хопфілд вперше представив свою асоціативну мережу у 1982 р. у Національній Академії Наук. На честь Хопфілда та нового підходу до моделювання, ця мережна парадигма згадується як мережа Хопфілда. Мережа базується на аналогії фізики динамічних систем. Початкові застосування для цього виду мережі включали асоціативну, або адресовану за змістом пам'ять та вирішували задачі оптимізації.

Мережа Хопфілда використовує три прошарки: вхідний, прошарок Хопфілда та вихідний прошарок (рисунок 5.1). Кожен прошарок має однакову кількість нейронів. Входи прошарку Хопфілда під'єднані до виходів відповідних нейронів вхідного прошарку через змінні ваги з'єднань. Виходи прошарку Хопфілда під'єднуються до входів всіх нейронів прошарку Хопфілда, за винятком самого себе, а також до відповідних елементів у вихідному прошарку. В режимі функціонування, мережа скеровує дані з вхідного прошарку через фіксовані ваги з'єднань до прошарку Хопфілда. Прошарок Хопфілда коливається, поки не буде завершена певна кількість циклів, і змінний стан прошарку передається на вихідний прошарок. Цей стан відповідає образу, вже запрограмованому у мережу.

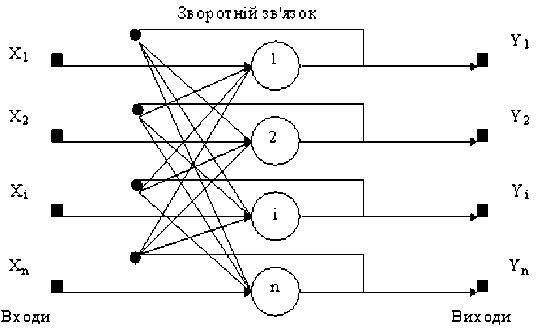


Рис. 5.1 – Структурна схема мережі Хопфілда

Навчання мережі Хопфілда вимагає, щоб навчальний образ був представлений на вхідному та вихідному прошарках одночасно. Рекурсивний характер прошарку Хопфілда забезпечує засоби корекції всіх вагів з'єднань. Недвійкова реалізація мережі повинна мати пороговий механізм у передатній функції. Для правильного навчання мережі відповідні пари "вхід-вихід" мають відрізнятися між собою.

Якщо мережа Хопфілда використовується як пам'ять, що адресується за змістом вона має два головних обмеження. По-перше, число образів, що можуть бути збережені та точно відтворені є строго обмеженим. Якщо зберігається занадто багато параметрів, мережа може збігатись до нового неіснуючого образу, відмінному від всіх запрограмованих образів, або не збігатись взагалі. Межа ємності пам'яті для мережі приблизно 15% від числа нейронів у прошарку Хопфілда. Другим обмеженням парадигми є те, що прошарок Хопфілда може стати нестабільним, якщо навчальні приклади є занадто подібними. Зразок образу вважається нестабільним, якщо він застосовується за нульовий час і мережа збігається до деякого іншого образу з навчальної множини. Ця проблема може бути вирішена вибором навчальних прикладів більш ортогональних між собою.

Задача, розв'язувана даною мережею в якості асоціативної пам'яті, як правило, формулюється таким чином. Відомий деякий набір двійкових сигналів (зображень, звукових оцифровок, інших даних, що описують якийсь об'єкти або характеристики процесів), вважають зразковим. Мережа повинна вміти з зашумленого сигналу, поданого на її вхід, виділити («пригадати» по частковій інформації) відповідний зразок або «дати висновок» про те, що вхідні дані не відповідають жодному із зразків. У загальному випадку, будь-який сигнал може бути описаний вектором , де п - число нейронів у мережі і величина вхідних і вихідних векторів. Кожний елемент  дорівнює або +1, або -1. Позначимо вектор, що описує к - тий зразок, через , а його компоненти, відповідно, - , де т - число зразків. Якщо мережа розпізнає (або «пригадує») якийсь зразок на основі пред'явлених їй даних, її виходи будуть містити саме його, тобто , де  - вектор вихідних значень мережі: . У противному випадку, вихідний вектор не співпаде з жодний зразковим.

Якщо, наприклад, сигнали являють собою якесь зображення, то, відобразивши у графічному виді дані з виходу мережі, можна буде побачити картинку, що цілком збігається з однієї зі зразкових (у випадку успіху) або ж «вільну імпровізацію» мережі (у випадку невдачі).

Алгоритм функціонування мережі

1. На стадії ініціалізації мережі вагові коефіцієнти прихованого шару встановлюються таким чином:

, (5.1)

де  та  – індекси нейронів відповідно, вихідного і прихованого шарів;  та  –  - тий та  - тий елементи вектора *к* - того зразка.

2. На входи мережі подається невідомий сигнал (*t* - номер ітерації). Його поширення безпосередньо встановлює значення виходів:

, (5.2)

тому позначення на схемі мережі вхідних сигналів у явному виді носить чисто умовний характер. Нуль у дужці справа від означає нульову ітерацію в циклі роботи мережі.

3. Розраховується новий стан нейронів

 (5.3)

і нові значення виходів

 (5.4)

де f – порогова передатна функція, яка може мати вигляд однієї з представлених на рисунку 5.2.

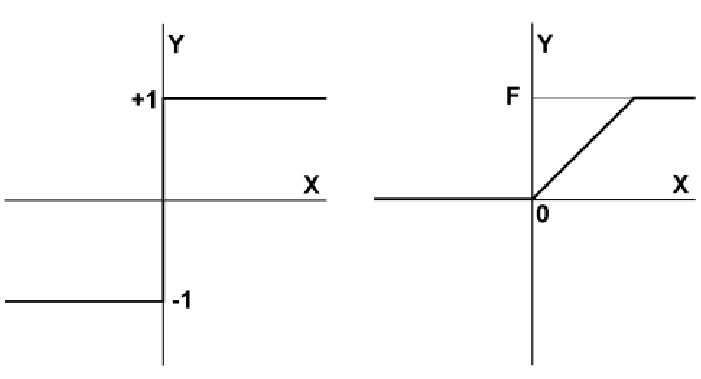


Рис. 5.2 – Передатні функції прихованих нейронів мережі Хопфілда

4. Перевіряється, чи змінилися вихідні значення виходів за останню ітерацію. Якщо так - перехід до пункту 2, інакше (якщо виходи стабілізувались) - кінець. При цьому вихідний вектор являє собою зразок, що найкраще відповідає вхідним даним.

**Мережа Хеммінга** є розширенням мережі Хопфілда. Ця мережа була розроблена Річардом Ліппманом у середині 80-х рр. Мережа Хемінга реалізує класифікатор, що базується на найменшій похибці для векторів двійкових входів, де похибка визначається відстанню Хемінга.

Відстань Хемінга визначається як число бітів, які відрізняються між двома відповідними вхідними векторами фіксованої довжини. Один вхідний вектор є незашумленим прикладом образу, інший є спотвореним образом. Вектор виходів навчальної множини є вектором класів, до яких належать образи. У режимі навчання вхідні вектори розподіляються до категорії, для яких відстань між зразковим вхідним вектором та поточним вхідним вектором є мінімальною.

Мережа Хеммінга має три прошарки:

- вхідний шар з кількістю вузлів, що дорівнює кількості окремих двійкових ознак;

- шар категорій (шар Хопфілда), з кількістю вузлів, що дорівнює кількості категорій або класів;

- вихідний шар, потужність якого відповідає числу вузлів у шарі категорій.

Мережа є простою архітектурою прямого поширення з вхідним шаром, повністю під'єднаним до шару категорій – рисунок 5.3. Кожен елемент у шарі категорій є зворотно під'єднаним до кожного нейрона у тому ж самому шарі й прямо під'єднаним до вихідного нейрону. Вихід з шару категорій до вихідного шару формується через конкуренцію.

Навчання мережі Хеммінга є подібним до методології Хопфілда. На вхідний шар надходить бажаний навчальний образ, а на виході вихідного шару надходить значення бажаного класу, до якого належить вектор. Вихід містить лише значення класу до якого належить вхідний вектор. Рекурсивний характер прошарку Хопфілда забезпечує збіжність корекції всіх вагових коефіцієнтів.

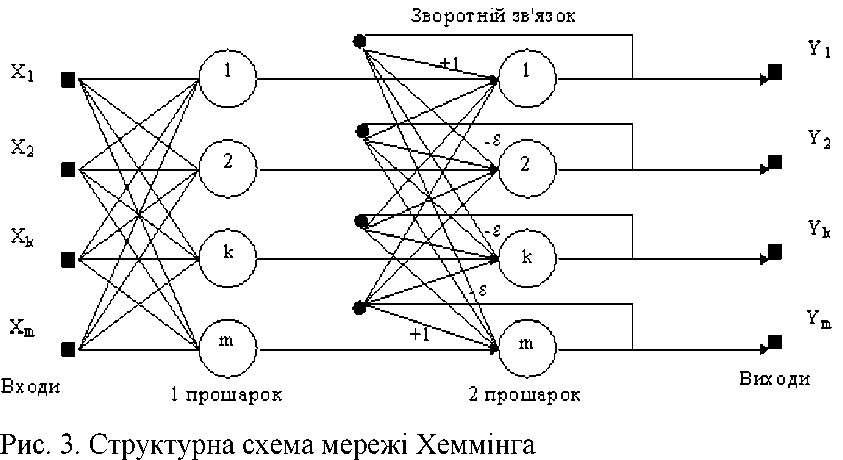


Рис. 5.3 – Структурна схема мережі Хеммінга

Алгоритм функціонування мережі Хеммінга

Крок 1. На стадії ініціалізації ваговим коефіцієнтам першого прошарку і порогу передатної функції присвоюються такі значення:

, , ;

**, .

Тут  – i-тий елемент k-того зразка.

Вагові коефіцієнти гальмуючих синапсів у другому шарі беруть рівними деякій величині  з від’ємним знаком. Синапс нейрона, пов'язаний із його ж виходом має вагу +1.

Крок 2. На входи мережі подається невідомий вектор . Розраховуються стани нейронів першого прошарку (верхній індекс у дужках вказує номер прошарку):

 (5.5)

Після цього отримані значення ініціалізують значення виходів другого шару:

Крок 3. Обчислюються нові стани нейронів другого прошарку:

 (5.7)

і значення їх виходів:

 (5.8)

Передатна функція  має вид порога, причому величина ** повинна бути достатньо великою, щоб будь-які можливі значення аргументу не призводили до насичення.

Крок 4. Перевіряється, чи змінилися виходи нейронів другого шару за останню ітерацію. Якщо так - перейти до кроку 3. Інакше - кінець.

Роль першого шару є умовною: скориставшись один раз на першому кроці значеннями його вагових коефіцієнтів, мережа більше не повертається до нього, тому перший шар може бути взагалі виключений із мережі.

Мережа Хеммінга має ряд переваг над мережею Хопфілда. Вона реалізує оптимальний класифікатор за критерієм мінімуму похибки, якщо похибки вхідних бітів є випадковими та незалежними. Для функціонування мережі Хеммінга потрібна менша кількість нейронів, оскільки середній прошарок вимагає лише один нейрон на клас, замість нейрону на кожен вхідний вузол. І, нарешті, мережа Хеммінга не страждає від неправильних класифікацій, які можуть трапитись у мережі Хопфілда. В цілому, мережа Хеммінга є як швидшою, так і точнішою за мережу Хопфілда.

**Практична реалізація нейронних мереж з авто асоціативною пам’яттю**

Функція створення мережі Хопфілда в MATLAB має вигляд **net = newhop(T)**, де Т – масив елементів розміром RxQ, що є навчальною вибіркою (еталонними відповідями) з Q цільових векторів довжиною R, кожен елемент яких дорівнює +1 або –1.

Перевірка роботи мережі Хопфілда для нового образу А з фіксацією помилки і вихідного значення виконується командою [**Y,Pf,Af] = sim(net,{1 5},{},A)**, де А – зашумлений вхідний образ. Y – значення виходів мережі на кожній ітерації, Pf – значення активаційної функції на кожній ітерації, Af – значення помилки на кожній ітерації.

Приклад:

>> T = [–1 –1 1; 1 –1 1]'

>> net = newhop(T);

>> A = T;

>> Y = sim(net,2,[], A)

Y =

–1 1

–1 –1

1 1

>> A = {[–0.6; –0.6; 0.6]};

>> [Y,Pf,Af] = sim(net,{1 5},{},A)

>> [Y{1} Y{2} Y{3} Y{4} Y{5}]

ans =

–0.6971 –0.8099 –0.9410 –1.0000 –1.0000

–0.9884 –1.0000 –1.0000 –1.0000 –1.0000

0.9884 1.0000 1.0000 1.0000 1.0000

Для мережі Хеммінга не існує стандартної функції в пакеті MATLAB, але вона реалізована у бібліотеці **Neurolab** системи програмування Python. Докладніше з роботою бібліотеки та використанням її функцій можна познайомитись за посиланням <https://code.google.com/archive/p/neurolab/> .

Лабораторна робота № 4.

Реалізація та дослідження багаторядного МГВА

**Завдання**

Необхідно одержати математичну модель певної залежності методом МГВА, здійснити прогнозування та порівняти результат МГВА із результатом, що одержаний будь-яким іншим методом, використовуючи контрольні точки відомої функції. Для цього виконати такі кроки:

1. Розробити програмний фрагмент, що реалізує метод найменших квадратів для лінійної функції від  змінних.

2. У відповідності до критерію селекції (див. початкові дані), розділити набір даних на навчальну та контрольну послідовності.

3. Обрати у відповідності до варіанту опорну функцію.

4. Розрахувати за допомогою MHK коефіцієнти опорних функцій.

5. За відповідними критеріями селекції залишити певний відсоток таких функцій для подальших ітерацій.

6. При збільшенні значення критерію ітерації припинити і визначити прогнозне значення функції, порівняти його із істинним.

7\* Вивести поліном в аналітичному вигляді на друк.

**Початкові дані** знаходяться у файлі *lab4\_vars.xls* в робочому каталогу навчальної дисципліни на сервері кафедри. Набір даних складається з трьох вхідних змінних і однієї цільової, що обирається за номером варіанту. Останній співпадає з останньою цифрою студента в списку групи. Також за варіантом обирається критерій селекції моделей у МГВА та опорна функція, вибір яких регулює таблиця 4.1.

Таблиця 4.1 – Вибір критерія селекції та опорної функції для індивідуального варіанту завдання

|  |  |  |
| --- | --- | --- |
| № вар | Критерій селекції | Опорна функція |
| 1 | Балансу змінних | (4.3) |
| 2 | Регулярності за похибкою | (4.3) |
| 3 | Регулярності за кореляцією | (4.4) |
| 4 | Незміщеності 1 | (4.4) |
| 5 | Балансу змінних | (4.4) |
| 6 | Незміщеності 2 | (4.5) |
| 7 | Регулярності за кореляцією | (4.5) |
| 8 | Регуларності за похибкою | (4.5) |
| 9 | Незміщеності 1 | (4.6) |
| 0 | Незміщеності 2 | (4.6) |

Для перевірки результатів слід скористатися готовою програмою <http://www.machinelearning.ru/wiki/index.php?title=GMDH_Shell>, що виконує

**Методичні вказівки**

Нехай початкові дані зосереджені в матриці *,* де **,длявсіх ** та ** – вектори-стовпчики розмірністю *,* серед яких ** – вхідні фактори,а *Y –* вихідна характеристика. Задача полягає в ідентифікації залежності

 (4.1)

поліномом Колмогорова-Габора у загальному вигляді

** (4.2)

Відомо, що при збільшенні порядку полінома точність наближення ним функції *F(x)* зростає, а потім спадає. Інакше кажучи, нарощуючи максимальний ступінь полінома , маємо наступний вигляд від цього ступеню середньоквадратичної похибки апроксимації (рис. 6.1). Якщо точність є максимальною, то цей процес закінчується.

***1***

***2***

***3***

***4***

***5***

***6***

***S***

******

***- ***

***- ***

***7***

Рис. 4.1 – Зміна точності різних моделей МГВА при збільшенні складності

Особливістю МГВА є те, що він може бути застосований у випадку малої кількості точок експериментів, навіть значно меншої, ніж кількість членів полінома. Це пояснюється тим, що на кожному етапі моделювання апроксимація виконується не за допомогою повного поліному поточної складності, а за допомогою елементарної опорної функції.

Опорна функція обирається на першому етапі реалізації МГВА. Найчастіше в якості опорних використовуються наступні залежності:

1) мультиплікативна

; (4.3)

2) адитивна

; (4.4)

3) повна першого порядку

; (4.5)

4) повна другого порядку

. (4.6)

Для першої функції необхідні початкові дані хоча б трьох експериментів, для другої - 4, для третьої - 5, для четвертої - 7. Це пояснюється тим, що для визначення коефіцієнтів буде використано метод найменших квадратів, в якому необхідно мати хоча б одну ступінь свободи, аби отримана залежність мала ступінь довіри, відмінний від 0.

Позначимо **, де ** - одна із залежностей (4.3)-(4.6) або, можливо, подібна. На наступному кроці за допомогою MHK визначають коефіцієнти  рівнянь, де(без урахування повторів та діагональних елементів) .

Після того, як усі залежності*,*  ідентифіковані за МНК, до справи вступає зовнішній критерій, за яким із усієї множини відбирають найкращі моделі.

Для забезпечення роботи зовнішнього критерію початкову вибірку слід поділити на навчальну та перевірочну. В залежності від обраного критерію якості, навчальна вибірка має містити 50-70% рядків початкової таблиці даних, але мінімум на один рядок більше, ніж порядок опорної функції.

Аби одразу відсіяти “неблагонадійні” моделі, у вигляді обмеження застосовується допоміжний критерій точності, відомий з дисперсійного аналізу

, (4.7)

де  - табличне значення шуканої залежності в тому спостереженні;  - прогнозоване значення в тому ж рядку таблиці за тою моделлю;  - середнє значення шуканої величини.

Якщо за (4.7) отримано значення , кажуть, що модель перспективна й може застосовуватися далі, якщо  - модель може використовуватись, але обережно, при  використання тої моделі небажано, а при  модель не може застосовуватися, оскільки вносить дезінформацію.

Серед моделей, що задовольняють обмеженню за (4.7), визначають  кращих за критеріями регулярності та незміщеності (про них – далі в лекції).  може бути константою (деякі дослідники встановлюють наперед , аби не мати проблем зі зміною розмірності задачі), зростати чи зменшуватися у функції етапу .

Ті моделі залежності, які залишилися, перенумеровують і одержують нову таблицю зі значень . На цьому перший крок селекції закінчено, якщо розрахувати та запам’ятати ефективність найкращої з моделей. На наступному кроці все викладене повторюється: за допомогою MHK визначають коефіцієнти таких самих опорних функцій, але вже від нових змінних

; ; ...

; ; ... (4.8)



Зрозуміло, що процес має циклічний характер і повторюється, допоки значення зовнішнього критерію покращуються. Селекція припиняється, коли її подальше ускладнення неможливе за (4.7) або погіршує значення зовнішнього критерію. Тоді говорять, що оптимальна за складністю модель знайдена й здійснюють зворотній процес розкриття отриманого виразу.

Умови закінчення ітерацій не канонізовані і можуть бути, наприклад, такими:

* середнє значення помилки для наступного ряду селекції є більшим ніж найбільше (середнє) значення помилки для попереднього ряду;
* мінімальне значення помилки наступного ряду більше мінімального значення помилки попереднього ряду;
* максимальне значення помилки наступного ряду більше максимального значення помилки попереднього раду;
* модуль відхилення помилок наступного і попереднього ряду менше деякого числа.

**Критерії регулярності**

Опишемо зовнішні критерії, використання яких базується на принципі зовнішнього доповнення. В залежності від типу задачі О.Г. Івахненко запропонував розглядати такі критерії: регулярності, незміщеності та балансу змінних. Відомі два критерії регулярності:

* мінімум середньоквадратичної помилки на нових точках окремої контрольної послідовності;

; (4.9)

* максимум коефіцієнта кореляції на тих же точках

. (4.10)

Розглянемо процедуру їх застосування. Початкові дані знаходяться у таблиці. Розділимо її рядки на дві частини (приблизно 60% на 40%), тоді *,* де  - кількість точок експерименту в першій (навчальній) вибірці,  - у другій (контрольній). Значення , нагадаємо, повинне бути більшим від числа доданків в опорній функції *.*

Використовуючи елементи навчальної вибірки, визначаємо коефіцієнти тієї з залежностей (6.3)-(6.6), яка прийнята за опорну функцію для усіх  сполучень стовпців, перевіряючи кожну модель за критерієм точності (4.7).

Далі розраховуємо значення критерію регулярності (4.9) чи (4.10) на точках контрольної вибірки, після чого впорядковуємо моделі від кращого значення критерію до гіршого і залишаємо з них певну кількість  кращих. Після перенумерації вони складуть множину функцій наступного ряду селекції.

*Перевагою* критеріїв регулярності є плавність зміни їх значення при збільшенні складності моделі. Недоліком їх використання є низька точність при розв'язанні екстраполяційних задач. Тому критерії регулярності раціонально застосовувати для ідентифікації, інтерполяції та короткострокового прогнозу.

**Критерії незміщеності**

Відомі три види критерію незміщеності, що може обчислюватись на базі аналізу розв'язків, аналізу коефіцієнтів та як „критерій відносної незміщеності".

*Критерій незміщеності, що базується на аналізі розв'язків* (KH1). Для розрахунку KHl необхідно ранжувати всі точки експериментів за збільшенням або зменшенням значення дисперсії. Процедура ранжування описана нижче. Після ранжування точки експериментів нумерують і ділять на дві послідовності:

- до першої відносять точки з парними номерами, їх кількість *,*

- до другої – точки з непарними, їх кількість **. Логічно, що **.

На першому ряді селекції перша послідовність є навчальною, друга - контрольною. Отримані на навчальній послідовності ** рівняння регресії позначимо *.*

Далі першу послідовність вважають контрольною, другу - навчальною. На навчальній послідовності знаходять рівняння регресії *.* Кількість рівнянь ** та ** повинна співпадати, випадок невиконання цієї умови не розглядаємо. Для кожного **того рівняння розраховують середньоквадратичне відхилення

, (4.11)

де  – номер рядка у таблиці значень тої моделі. Моделі з найменшим значенням критерію вважаються найточнішими.

Як і за критерієм регулярності, за критерієм КН1 обирають  найкращих рівнянь, що відповідають меншій оцінці критерію.

Для визначення фактора зупинки алгоритму визначають середнє зважене значення критерію незміщеності моделей даного покоління

 (4.12)

На другому і подальших рядах селекції  процедура залишається тією самою. Селекція продовжується доти, доки середнє значення критерію незміщеності зменшується ().

*Критерій незміщеності, що базується на аналізі коефіцієнтів* (KH2). Точки експериментів ранжуються за величиною дисперсії і поділяються навпіл на навчальну та контрольну послідовності. Точки з більшим значенням дисперсії потрапляють до навчальної послідовності, з меншим - до контрольної. Особливість критерію полягає в тому, що на кожному ряді селекції ранжування і розділення точок експерименту виконується наново. Крім того, зменшується свобода вибору згідно з формулою

**, (4.13)

де * -* число моделей, що переходять у нове покоління на даному ряді селекції; *S -* номер ряду; ** - кількість вхідних змінних;  (чим більше незалежних змінних, тим менше); . Формула (4.13) та пов'язана з нею процедура дають можливість швидкого розв'язання задач великої розмірності.

Значення критерію незміщеності оцінок коефіцієнтів моделі розраховується за формулою

 (4.14)

де *p -* загальне число моделей, ** - коефіцієнти тої моделі, отримані до зміни послідовностей;  - коефіцієнти тієї ж моделі, отримані після зміни;  - відносна доля співпадінь.

До наступного ряду селекції (в наступне покоління) пропускають  за (4.13) моделей, що мають більше значення .

Як і для , розрахунки середнього значення залишених моделей для формування умови загальної зупинки алгоритму справедливі. Селекція продовжується доти, доки середнє значення  у ряді селекції збільшується.

Критерій незміщеності  необхідно застосовувати або разом з критерієм регулярності, або в алгоритмах з повним перебором моделей. При відтинанні частини моделей він швидко сходиться до локального оптимуму – найближчої задовільної моделі.

*Критерій відносної незміщеності.* У цьому випадку використовують лише лінійні часткові описи (наприклад, ). Але щоб уникнути втрати точності, простір початкових аргументів включає також коваріації (наприклад, , , ). Присвоюючи значення коваріацій новим змінним, отримаємо узагальнені аргументи , кількість яких може значно переважати кількість стовпців у початковій таблиці.

Оскільки частковий опис на другому ряді має вигляд , то ортогоналізований частковий опис матиме вигляд

 (4.15)

де * –* вектор, ортогоналізований по відношенню до *.*

Якщо у часткових описах значення змінних центровані та нормовані за середнім значенням, то в ортогоналізованих часткових описах вільний член а0 = 0, а інші коефіцієнти мають такі значення:

 (4.16)

Ортогоналізація – це перетворення ** по відношенню до базового , в результаті якого одержуємо . Для цього достатньо знайти такі модельні значення *,* де *A* визначається за (4.16).

Таким чином,алгоритм МГВА з використанням критерію відносної незміщеності має такі базові кроки:

*Крок* 1. Початкові дані ділимо на дві частини (описано раніше).

*Крок* 2. На першій послідовності визначаємо значення коефіцієнтів  в рівнянні регресії, на іншій - коефіцієнти . Кращими вважаємо ті описи, у яких

 (4.17)

Ряди селекції при цьому матимуть вигляд: ; ;  і так далі до зупинки за глобальним критерієм.

**Критерій балансу змінних**

Критерій балансу змінних є найефективнішим при екстраполяції, тобто застосуванні МГВА у середньостроковому та довгостроковому прогнозуванні. Його визначення може виконуватись як емпірично, так і штучно.

При емпіричному визначенні критерію балансу змінних із подальшого розгляду на даному ряді селекції виключаються ті моделі, які дають заздалегідь невірні результати, що лежать за межами можливих значень прогнозованої залежності. Крім того, на значення прогностичних моделей можуть накладатися додаткові умови, наприклад,  (модельне значення не має перевищувати відомого з таблиці) або  (модельне значення не має виходити за певний діапазон).

Штучні умови балансу не є наслідком принципу фізичної реалізації моделі і визначаються дослідником. Найчастіше – це функції, що є комбінацією сум чи різниць вхідних факторів. При цьому, до переліку вхідних факторів  додаються незалежні змінні , які зазвичай не мають фізичного сенсу.

Наприклад, для задачі побудови екстраполяційної моделі за трьома факторами ,  та  можуть бути введені наступні змінні:

; ; ...  (4.18)

Ці змінні разом з початковими змінними ,  та  утворюють вектор вхідних факторів. Розраховані  заносяться у таблицю початкових даних та надалі використовуються разом з іншими змінними.

За описаною вище методикою із застосуванням МНК намагаються отримати модель оптимальної складності . При цьому, на кожному кроці моделі оцінюють не за звичними критеріями незміщенності чи регулярності, а наступним чином.

Припустимо, що на певному ряду отримані сімейство моделей вигляду

, . (4.19)

Найкращою з усіх  моделей буде вважатися така, яка матиме мінімальне розсіювання на інтервалі спостереження

 (4.20)

Таким чином, за всіма табличними значеннями відомих  ми обчислюємо найкращу модель для прогнозування , обчислюючи на тому ж інтервалі за допоміжними змінними якість моделі.

Вигляд критерію (4.20) та необхідність створення й обчислення додаткових векторів даних вказують на очевидну трудомісткість застосування даного критерію на практиці. Тому при його використанні необхідно застосовувати процедури зменшення кількості комбінацій перебору, у тому числі й враховуючи принципи фізичної реалізації та здорового глузду. Кількість додаткових змінних обирається дослідником і має знаходитися в межах здорового глузду та можливостей обчислювальної машини.

Слід пам’ятати, що критерій балансу змінних за математичною сутністю дуже близький (а в певних задачах ідентичний) критерію мінімуму незміщеності.

**Алгоритм поділу початкової вибірки даних**

Реалізація МГВА, як ми показали раніше, пов'язана із необхідністю поділу генеральної сукупності даних на дві вибірки - навчальну та контрольну. Найбільш поширеним, проте не єдиним, є підхід, за яким до навчальної послідовності обирають точки експериментів з більшим значенням дисперсії, а до контрольної - з меншим. Це пояснюється тим, що область навчання повинна бути якнайширшою, а контрольні точки, в більшості своїй, знаходитися всередині неї.

Для практичної реалізації пропонується наступний варіант алгоритму:

*Крок* 1. Визначити відсоткове співвідношення між кількістю елементів у навчальній і контрольній послідовності (від 70% / 30% до 50% / 50%, пов’язано з критерієм ефективності, що буде застосовуватись).

*Крок* 2. Для кожного стовпчика*,*  розрахувати середнє значення його елементів

, (4.21)

та отримати середнє значення множини образів ().

*Крок* 3. Знайти вибіркові дисперсії для кожного рядка таблиці за формулою

,  (4.22)

*Крок* 4. Для впорядковування таблиці переставити рядки таким чином, щоб першим був рядок з найбільшим значенням дисперсії, а останнім - з найменшим.

*Крок* 5. У відповідності до результату кроку 4 розділити дані в таблиці на навчальну та контрольну послідовності.

Якщо розв'язується задача короткострокового прогнозу або інтерполяція всередині інтервалу наявних даних, тоді додатково шукають оптимальне співвідношення кількості образів у навчальній та контрольній послідовностях з метою отримання найпростішої та достовірної моделі.

Лабораторна робота № 5.

Глибоке навчання загорткових нейронних мереж

**Завдання**

Скориставшись засобами Deep Learning Toolbox, створити і навчити глибоку нейронну мережу. Для цього виконати:

1. Завантаження набору даних MNIST рукописних цифр.

2. Створення згорткової нейронної мережі.

3. Навчання нейронної мережі на отриманому наборі даних та тестування на відповідній тестовій послідовності.

4. Знаходження такої структури мережі та параметрів навчання, які дозволять забезпечити точність розпізнавання не менше 99%.

За підсумками роботи скласти звіт, де пояснити отримані результати, налаштування та параметри навчання.

**Початкові дані**. В якості початкових даних в даній роботі використовуються дані з набору MNIST, який містить 60 000 зображень рукописних цифр 0-9 та 10 000 тестових зображень.

**Методичні вказівки**

На теперішній час термін «глибоке навчання» застосовується до нейронних мереж прямого з’єднання, що мають понад три шари. Особливим різновидом таких мереж є згорткові нейронні мережі (CNN).

Існує три підходи до глибокого навчання таких нейронних мереж:

1) Навчання мережі з нуля – найдовший, але найкраще пристосований для певної задачі.

2) Використання трансферного навчання для навчання існуючої мережі – вже навчену мережу, що розпізнає певні класи доучують на прикладах нових класів, які вона розпізнає.

3) Навчання існуючої навченої мережі для виконання сегментації зображень.

В даній лабораторній роботі все буде присвячено зображенням, як найбільш типовому об’єкту загорткових нейромереж. Але глибоке навчання стає все більш популярними і для інших задач: обробки звуків, зокрема мови, генерації текстів чи зображень і навіть відео. Принципи глибоко навчання також можуть бути застосовані до будь-яких сигналів даних у задачах регресії, прогнозування, пошуку асоціацій та інших.

Щоб знати більше, можна подивитися пояснювальні відео:

1) Що таке глибоке навчання? <https://www.mathworks.com/videos/introduction-to-deep-learning-what-is-deep-learning--1489502328819.html>

2) Глибоке навчання vs. Машинне навчання <https://www.mathworks.com/videos/introduction-to-deep-learning-machine-learning-vs-deep-learning-1489503513018.html>

В навчальному прикладі мова піде про розпізнавання рукописних цифр. Ми будемо використовувати дані набір даних MNIST, який містить 60 000 зображень рукописних цифр 0-9. Сам набір даних та опис кращих досягнень при його аналізі наведені за посиланням <http://yann.lecun.com/exdb/mnist/> . Аби завантажити цей набір даних, треба виконати команду

[imgDataTrain, labelsTrain, imgDataTest, labelsTest] = prepareData;

Preparing MNIST data...  
MNIST data preparation complete.

Код функції prepareData наведений у додатку 1 до даної лабораторної роботи. Аби зрозуміти, що представляє собою набір даних. Виведемо на екран 25 випадкових зображень з нього:

warning off images:imshow:magnificationMustBeFitForDockedFigure

perm = randperm(numel(labelsTrain), 25);

subset = imgDataTrain(:,:,1,perm);

montage(subset)

warning off images:imshow:magnificationMustBeFitForDockedFigure

perm = randperm(numel(labelsTrain), 25);

subset = imgDataTrain(:,:,1,perm);

montage(subset)

Результат виконання наведеного коду ілюструє рисунок 5.1



Рис. 5.1 – Приклад рукописних цифр у наборі даних MNIST

Набір MNIST обраний нами для ілюстрації, адже він містить чорно-білі зображення маленького розміру (28\*28 пікселів). Відтак, для його обробки достатньо використовувати звичайний персональний комп’ютер, без потужних GPU та звернення до хмарних обчислень.

Далі проілюструємо всі кроки – від створення мережі до підвищення її ефективності до рівня 98-99%.

Ми можемо перевірити розмір та клас даних, отриманих в ході виконання функції prepareData, ввівши наступний запит:

>> whois imgDataTrain

Name Size Byte Class

imgDataTrain 28x28x1x60000 47040000 uint8

Для розпізнавання рукописних літер вже існує стандартна згорткова мережа, що входить до складу Deep Learning Toolbox в пакет прикладних програм MATLAB починаючи з версії 2018а. оцінити її роботу дозволяє наступний набір команд, який завантажує мережу, вибирає випадкове значення однієї 10 000 тестових рукописних цифр і вгадує її. Також надано ілюстрацію того зображення, яке було обрано.

load MNISTModel

% Predict the class of an image

randIndx = randi(numel(labelsTest));

img = imgDataTest(:,:,1,randIndx);

actualLabel = labelsTest(randIndx);

predictedLabel = net.classify(img);

imshow(img);

title(['Predicted: ' char(predictedLabel) ', Actual: ' char(actualLabel)])

Звичайно, що при кожному запуску саме зображення, цифра, яку воно позначає та передбачення мережі будуть різні. Вбудована мережа має точність вгадування 99,53%.

Ми ж будуватимемо власну CNN, її найпоширеніший вид – мережу глибокого навчання з шарами згортки, фільтрації, лінійними активаційними функціями (ReLU) та повнозв’язаними шарами, що вирівнюють зображення та розподіляють їх на класи. Докладніше про структуру та функції кожного з шарів можна подивитись у відео «Що таке конволюційна нейронна мережа?» <https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html>

Будуючи мережу з нуля, добре почати з простого поєднання часто використовуваних шарів. Недостатня складність зробить процес налагодження набагато простішим і швидшим, але, ймовірно, нам потрібно буде додати ще кілька шарів, щоб досягти точності, до якої ми прагнемо.

Отже, для створення мережі нам будуть необхідні:

- шар згортки – вводить вхідні зображення через набір згорткових фільтрів, кожен з яких активує певні функції із зображень;

- лінійна активаційна функція (ReLU) – дозволяє прискорити і зробити ефективнішим навчання шляхом відображення негативних значень в нуль та масштабування позитивних значень;

- фільтрація (pooling) – спрощує виведення, виконуючи нелінійне зменшення простору, зменшуючи кількість параметрів, які потрібно ідентифікувати мережі;

- повнозв’язані шари – "вирівнюють" двовимірні просторові функції мережі в вектор ознак для цілей класифікації;

- шар Softmax розподіляє ймовірності для кожної категорії в наборі даних.

Відтак створюємо мережу з наступними шарами:

layers = [ imageInputLayer([28 28 1])

convolution2dLayer(5,20)

reluLayer

maxPooling2dLayer(2, 'Stride', 2)

fullyConnectedLayer(10)

softmaxLayer

classificationLayer() ]

layers =

7x1 Layer array with layers:  
1 '' Image Input 28x28x1 images with 'zerocenter' normalization  
2 '' Convolution 20 5x5 convolutions with stride [1 1] and padding [0 0 0 0]  
3 '' ReLU ReLU  
4 '' Max Pooling 2x2 max pooling with stride [2 2] and padding [0 0 0 0]  
5 '' Fully Connected 10 fully connected layer  
6 '' Softmax softmax  
7 '' Classification Output crossentropyex

Перед навчанням мережі необхідно задати певні опції цього процесу. Насправді їх достатньо багато, але в таблиці 5.1 наведені ключові, які суттєво впливають на результат і швидкість навчання.

Таблиця 5.1 – основні параметри глибокого навчання згорткової мережі

|  |  |  |
| --- | --- | --- |
| **Параметр навчання** | **Що визначає параметр** | **Порада щодо налаштування** |
| Plot of training progress | Визначає, виводити чи ні залежність точності навчання і кількості помилок від номеру епохи. Графічне вікно також містить кнопку зупинки процесу навчання, який можна перервати, якщо результати довго не змінюються | Якщо апаратне забезпечення дозволяє, краще відображати процес навчання наглядно, для цього слід вказати (**'Plots','training-progress'**) |
| Max epochs | Епоха – це пред’явлення всіх прикладів навчальної вибірки мережі по 1 разу. Параметр визначає, скільки епох користувач відводить для навчання мережі | За умовчанням параметр дорівнює 20. Можна збільшувати, але враховувати, що головним чином навчання відбувається в перші 10-20 епох. ( **‘MaxEpoch’ , 20** ) |
| Minibatch size | Мінібатч – порція, яка передається процесору на одночасну обробку. Чим потужніший процесор, тим більшими порціями можна передавати приклади | Чим більше порція, тим швидше навчання. Але слід розуміти, що сили процесора не безмежні. Якщо виникає помилка пам’яті, слід зменшити розмір порцій. За умовчанням (**‘MiniBatchSize’, 64**) |
| Learning rate | Швидкість навчання – головний параметр глибокого навчання, число на яке після кожного показаного прикладу максимально може змінитися ваговий коефіцієнт повнозв’язаного шару | Більш низький рівень навчання забезпечує більш точний результат, але час навчання зростає пропорційно. Зазвичай швидкість навчання обирають близькою до відношення (кількість епох) / (кількість навчальних прикладів) |

Для початку ми вкажемо лише два параметри: виводити хід навчання у графічне вікно та покласти розмір порції рівним 8192 приклади (використано восьми ядерний процесор і7 сьомого покоління з 8 ГБ оперативної пам’яті):

**miniBatchSize = 8192;**

**options = trainingOptions ( 'sgdm' , ...**

**‘MiniBatchSize’ , miniBatchSize, ...**

**‘Plots’, ‘training-progress’ );**

**net = trainNetwork (imgDataTrain, labelsTrain, layers, options);**

Запускаємо глибоке навчання мережі й стежимо за її прогресом. Зупинити навчання та повернути поточний стан мережі можна, натискаючи кнопку зупинки у верхньому правому куті екрана. Як тільки навчання зупиняється, потрібно перезапустити його з самого початку - не можна відновитись з того місця, де він зупинився.

Хід навчання ілюструє рисунок 5.2.

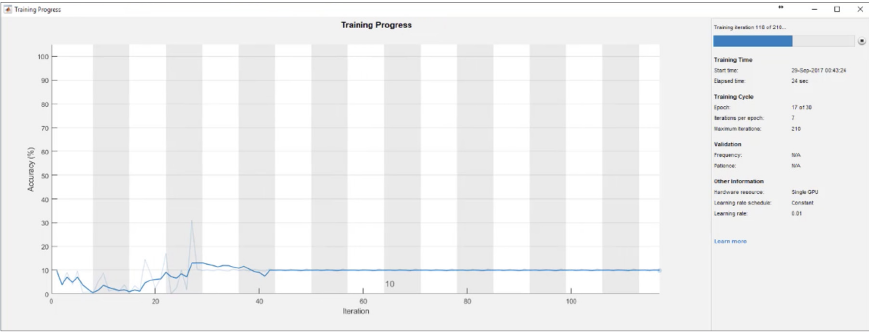


Рис. 5.2 – Хід навчання глибокої нейромережі з 7 шарів

Як видно, з таким набором параметрів навчання точність перестала покращуватись після 28-ї ітерації та стала постійною на рівні близькому до 10%. Зупинити процес варто як тільки крива середньої точності стає нагадувати плато.

Існує багато способів регулювання точності мережі. Серед них збільшення кількості навчальних прикладів, підвищення якості навчальних прикладів, зміна алгоритму навчання та зміна конфігурації мережі. деякі впливають істотно, деякі менш істотно. Наприклад, якщо ми не змінюючи архітектуру мережі замість значення швидкості навчання за умовчанням (0,01) поставимо зменшене в 100 разів значення, включивши до списку параметрів **(…,'InitialLearnRate', 0.0001, …)**.

Запустивши навчання з оновленими параметрами, маємо зовсім інший перебіг навчання, що ілюструється кривою на рисунку 5.3.

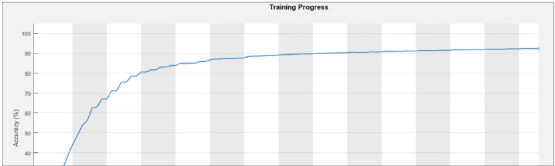


Рис. 5.3 – Хід навчання глибокої нейромережі зі зменшеною швидкістю

Лише одна проста опція дозволила збільшити точність до 91%!

Для деяких задач такий результат може бути задовільним, але ж ви пам’ятаєте, що стандартна мережа забезпечує точність вище 99%! Насправді шлях від 91% до 99% точності – вкрай непростий. Для його подолання можна застосовувати декілька шляхів, наприклад, Байєсівську регуляризацію (запуск навчання кілька разів з підбором найкращих параметрів навчання).

Але очевидно, що необхідно нарощувати кількість шарів і комбінувати їх різним чином, аби точність ставала кращою. Наприклад, якщо замість початкової мережі з 7 різних шарів створити наступну конфігурацію, точність навчання зросте майже до бажаної межі:

layers = [

imageInputLayer([28 28 1])

convolution2dLayer(3,16,'Padding',1)

batchNormalizationLayer

reluLayer

maxPooling2dLayer(2,'Stride',2)

convolution2dLayer(3,32,'Padding',1)

batchNormalizationLayer

reluLayer

maxPooling2dLayer(2,'Stride',2)

convolution2dLayer(3,64,'Padding',1)

batchNormalizationLayer

reluLayer

fullyConnectedLayer(10)

softmaxLayer

classi⿔cationLayer];

Цього разу ми змінили структуру мережі, але залиште параметри навчання такими, як були раніше. Аби перевірити, який відсоток прикладів з тестової вибірки вірно ідентифікує навчена мережа, слід скористатись командами:

>> predLabelsTest = net.classify(imgDataTest);

>> accuracy = sum(predLabelsTest == labelsTest) / numel(labelsTest)

accuracy = 0.9880

З 10 000 тестових зображень наша мережа помилилася лише в 120 випадках! Таку мережу можна використовувати для ідентифікації рукописних цифр в онлайн-зображеннях або навіть у он-лайнових відеопотоках.

Таким чином, підвищуючи складність мережі, змінюючи характеристики певних її шарів та параметри навчання, можна досягати максимального припустимого значення точності розпізнавання. На наборі MNIST на сьогодні рекордні значення демонструє мережа з 35 шарів.

Але слід пам’ятати, що тренувальні й тестові зображення, розглянуті нами в даній роботі, є дуже простими: 28 на 28 пікселів і 1 колір. Коли йде мова про реальні зображення, їх класифікацію і сегментацію, потрібна більші кількість шарів, більше вікно вхідного зображення (зазвичай від 32\*32 до 256\*256 пікселів на кожен з кольорів), більше згорткових шарів, більше вікно усереднення.

Також слід враховувати, що описаний інструмент є не єдиним можливим для реалізації згорткових нейронних мереж. Всі опереації можуть також бути реалізовані з середовищі Statistica Neural Network або мовою Python з використанням бібліотеки моделювання глибоких нейронних мереж Keras.

Додаток 1

Код функції, що завантажує набір даних MNIST

function [imgDataTrain, labelsTrain, imgDataTest, labelsTest] = prepareData

% Copyright 2018 The MathWorks, Inc.

%% Check for the existence of the MNIST files and download them if necessary

filePrefix = 'data';

files = { "train-images-idx3-ubyte",...

"train-labels-idx1-ubyte",...

"t10k-images-idx3-ubyte",...

"t10k-labels-idx1-ubyte" };

% boolean for testing if the files exist

% basically, check for existence of "data" directory

download = exist(fullfile(pwd, 'data'), 'dir') ~= 7;

if download

disp('Downloading files...')

mkdir data

webPrefix = "http://yann.lecun.com/exdb/mnist/";

webSuffix = ".gz";

filenames = files + webSuffix;

for ii = 1:numel(files)

websave(fullfile('data', filenames{ii}),...

char(webPrefix + filenames(ii)));

end

disp('Download complete.')

% unzip the files

cd data

gunzip \*.gz

% return to main directory

cd ..

end

%% Extract the MNIST images into arrays

disp('Preparing MNIST data...');

% Read headers for training set image file

fid = fopen(fullfile(filePrefix, char(files{1})), 'r', 'b');

magicNum = fread(fid, 1, 'uint32');

numImgs = fread(fid, 1, 'uint32');

numRows = fread(fid, 1, 'uint32');

numCols = fread(fid, 1, 'uint32');

% Read the data part

rawImgDataTrain = uint8(fread(fid, numImgs \* numRows \* numCols, 'uint8'));

fclose(fid);

% Reshape the data part into a 4D array

rawImgDataTrain = reshape(rawImgDataTrain, [numRows, numCols, numImgs]);

rawImgDataTrain = permute(rawImgDataTrain, [2,1,3]);

imgDataTrain(:,:,1,:) = uint8(rawImgDataTrain(:,:,:));

% Read headers for training set label file

fid = fopen(fullfile(filePrefix, char(files{2})), 'r', 'b');

magicNum = fread(fid, 1, 'uint32');

numLabels = fread(fid, 1, 'uint32');

% Read the data for the labels

labelsTrain = fread(fid, numLabels, 'uint8');

fclose(fid);

% Process the labels

labelsTrain = categorical(labelsTrain);

% Read headers for test set image file

fid = fopen(fullfile(filePrefix, char(files{3})), 'r', 'b');

magicNum = fread(fid, 1, 'uint32');

numImgs = fread(fid, 1, 'uint32');

numRows = fread(fid, 1, 'uint32');

numCols = fread(fid, 1, 'uint32');

% Read the data part

rawImgDataTest = uint8(fread(fid, numImgs \* numRows \* numCols, 'uint8'));

fclose(fid);

% Reprocess the data part into a 4D array

rawImgDataTest = reshape(rawImgDataTest, [numRows, numCols, numImgs]);

rawImgDataTest = permute(rawImgDataTest, [2,1,3]);

imgDataTest = uint8(zeros(numRows, numCols, 1, numImgs));

imgDataTest(:,:,1,:) = uint8(rawImgDataTest(:,:,:));

% Read headers for test set label file

fid = fopen(fullfile(filePrefix, char(files{4})), 'r', 'b');

magicNum = fread(fid, 1, 'uint32');

numLabels = fread(fid, 1, 'uint32');

% Read the data for the labels

labelsTest = fread(fid, numLabels, 'uint8');

fclose(fid);

% Process the labels

labelsTest = categorical(labelsTest);

disp('MNIST data preparation complete.');

% img = readMNISTImage(imgDataTrain, 3);

% figure, imshow(img);