ЛЕКЦІЯ 10.
Мурашині алгоритми та інші види еволюційних алгоритмів

ПЛАН

1. Мурашині алгоритми
2. Еволюційні алгоритми
3. Еволюційні алгоритми в нейронних мережах (на самостійне вивчення)
4. Еволюційні моделі

ЛІТЕРАТУРА

1. Кононюк А.Ю. Нейроні мережі і генетичні алгоритми – К.:«Корнійчук», 2008. – 446 с. SBN 978-966-7599-50

2. Ясницкий Л.Н. Введення в штучний інтелект. — 1-е. — Издательский центр "Академия", 2005. — С. 176. — ISBN 5-7695-1958-4

3. Г. К. Вороновский, К. В. Махотило, С. Н. Петрашев, С. А. Сергєєв. Генетичні алгоритми, штучні нейроні мережі і проблеми віртуальної реальності. — Замовне. — Х.: ОСНОВА, 1997. — С. 112. — ISBN 5-7768-0293-8

4. Д. Рутковская, М. Пилиньский, Л. Рутковский. Нейронные сети, генетические алгоритмы и нечеткие системы —1-е. — М.: Гарячая линия – Телеком, 2007. — С. 384. — ISBN 5-93517-103-1

ВСТУП

Згадаємо з попередньої лекції, що для створення інтелектуальних систем часто застосовують **еволюційний підхід**, що умовно можна поділити на (рис.1):

• **Еволюційні алгоритми** (моделювання загальних закономірностей еволюції). Це системи, які використовують ***еволюційні принципи розвитку популяції***. Вони успішно використовуються для завдань функціональної оптимізації і можуть легко бути описані математичною мовою.

• **Еволюційні моделі**. Це системи, які відтворюють біологічні популяції чи системи і поки що не є корисними в прикладному сенсі. Еволюційні моделі більше схожі на біологічні системи, мають складну поведінку, мало спрямовані на вирішення технічних завдань. До цих систем відносять так зване «штучне життя».

На попередньому ми вивчили генетичні алгоритми. На сьогоднішньому занятті залишилося розглянути: Мурашині алгоритми, Еволюційні стратегії, Еволюційне програмування, Генетичне програмування та окремий вид Еволюційні моделі

Еволюційні алгоритми - термін, що часто використовується для загального опису алгоритмів пошуку, оптимізації або навчання, що засновані на формалізованих принципах природного еволюційного процесу.

Еволюційні методи призначені для пошуку бажаних рішень і засновані на статистичному підході до дослідження ситуацій та ітераційному наближенні системи до шуканого стану. ***На відміну від точних методів математичного***

програмування *еволюційні методи дозволяють знаходити рішення, близькі до оптимальних, за прийнятний час, а* ***на відміну від відомих евристичних методів оптимізації*** *характеризуються істотно меншою залежністю від особливостей додатку (більш універсальні) і в багатьох випадках забезпечують кращу ступінь наближення до оптимального рішення.*

Рис.1.

1. Мурашині алгоритми

http://habrahabr.ru/post/105302/

Мураху не можна назвати кмітливою. Окрема мураха не в змозі прийняти жодного рішення. Вона влаштована вкрай примітивно: всі її дії зводяться до елементарних реакцій на навколишнє оточення і своїх побратимів. Мураха не здатна аналізувати, робити висновки і шукати рішення.

Ці факти, однак, ніяк не узгоджуються з успішністю мурах як виду. Вони існують на планеті понад 100 мільйонів років, будують величезні споруди, забезпечують їх всім необхідним і навіть ведуть справжні війни. У порівнянні з повною безпорадністю окремих особин, досягнення мурах здаються немислимими.

Домогтися таких успіхів мурахи здатні завдяки своїй соціальності. Вони живуть тільки в колективах - колоніях. Всі мурахи колонії формують так званий ройовий інтелект. Особини, що складають колонію, не повинні бути розумними: вони повинні лише взаємодіяти за певними простими правилами і тоді колонія цілком буде ефективною.

В колонії немає домінуючих особин, немає начальників і підлеглих, немає лідерів, що роздають вказівки і координують дії. Колонія є повністю самоорганізованою. Кожна з мурах володіє інформацією лише про локальну

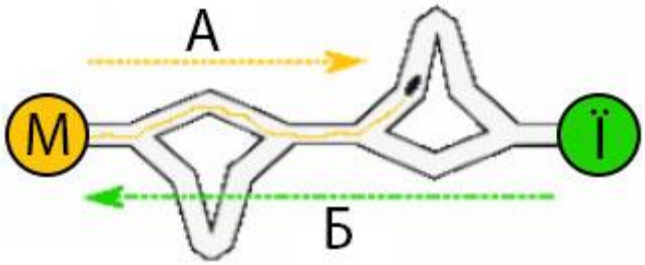
обстановку, жодна з них не має уявлення про всю ситуації в цілому - тільки про те, що дізналася сама або від своїх родичів, явно чи неявно. На неявних взаємодіях мурах засновано механізми пошуку найкоротшого шляху від мурашника до джерела їжі.

Кожен раз проходячи від мурашника до їжі і назад, мурахи залишають за собою доріжку феромонів. Інші мурахи, відчувши такі сліди на землі, будуть інстинктивно спрямовуватися до нього. Оскільки ці мурахи теж залишають за собою доріжки феромонів, то чим більше мурах проходить певним шляхом, тим більш привабливим він стає для їх родичів. При цьому, чим коротше шлях до джерела їжі, тим менше часу потрібно мурашкам на нього - а отже, тим швидше залишені на ньому сліди стають помітними.

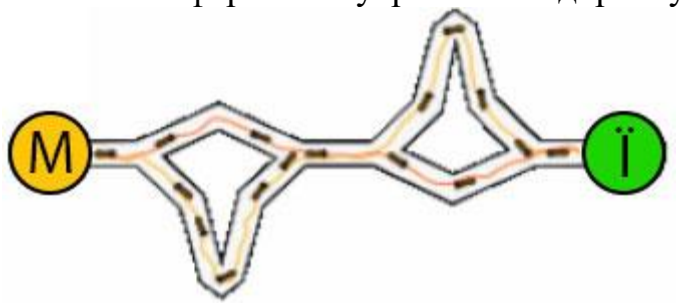
У 1992 році у своїй дисертації Марко Дориго запропонував запозичити описаний природний механізм для вирішення завдань оптимізації. Імітуючи поведінку колонії мурах в природі, мурашині алгоритми використовують багатоагентні системи, агенти яких функціонують по вкрай простим правилам. Мурашині алгоритми є ефективними при вирішенні складних комбінаторних завдань та задач оптимізації.

Ідея алгоритму

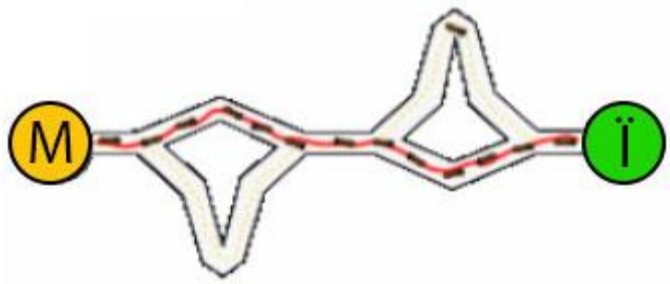
Мураха прямує від мурашника (М) у випадковому напрямку. Якщо вона знаходить їжу (Ї), то повертається до мурашника і залишає по собі слід з **феромону**.



Ці феромони приваблюють інших мурах, що знаходяться поруч, і скоріш за все вони рушать цим маршрутом. Повертаючись до мурашника, вони також залишають свій феромон і укріплюють доріжку.



Якщо існує два маршрути, то коротким за цей час встигне пройти більше мурах ніж по довгому і короткий маршрут стане більш вигідним. Довгші доріжки повільно зникають внаслідок випаровування феромонів.



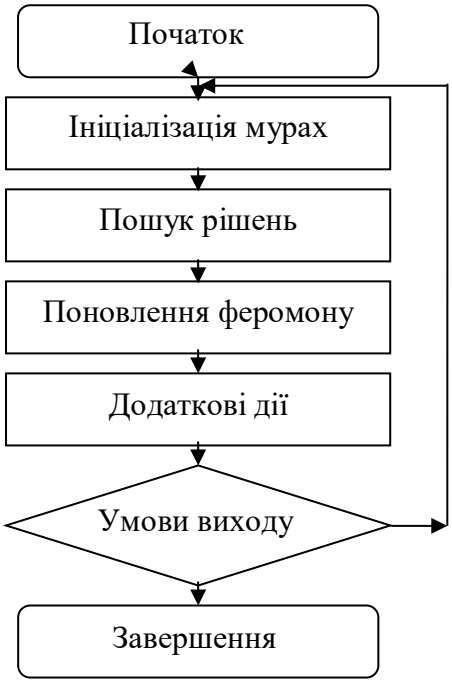
Базова ідея алгоритму мурахи полягає в оптимізації шляхом непрямого зв'язку між автономними агентами.

Мурашиний алгоритм моделює багатоагентну систему. Її агентів надалі будемо називати мурахами. Як і справжні мурахи, вони досить просто влаштовані: для виконання своїх обов'язків вони вимагають невелику кількість пам'яті, а на кожному кроці роботи виконують нескладні обчислення.

Кожна мураха зберігає в пам'яті список пройдених нею вузлів. Цей список називають **списком заборон (tabu list)** або просто пам'яттю мурашки. Вибираючи вузол для наступного кроку, мураха «пам'ятає» про вже пройдені вузли і не розглядає їх як можливих для переходу. На кожному кроці список заборон поповнюється новим вузлом, а перед новою ітерацією алгоритму - тобто перед тим, як мурашка знову проходить шлях - він очищується.

Окрім списку заборон, при виборі вузла для переходу мураха керується «**привабливістю**» ребер, які вона може пройти. *Привабливість залежить, по-перше, від відстані між вузлами (тобто від ваги ребра), а по-друге, від слідів феромонів, залишених на ребрі мурахами, що пройшли по ньому раніше.* Природно, що на відміну від ваг ребер, які є константними, сліди феромонів оновлюються на кожній ітерації алгоритму: як і в природі, з часом сліди випаровуються, а мурахи, що проходять, навпаки, посилюють їх.

Узагальнено, базовий мурашиний алгоритм, незалежно від модифікацій, можна представити у вигляді схеми



Покроковий опис загальної схеми

Припустимо, що навколишнє середовище для мурах представляє повнозв’язний неорієнтований граф. Кожне ребро має вагу, яка позначається як відстань між двома вершинами, що ним з’єднується. Граф є двохскерованим, тому мураха може подорожувати по грані в будь-якому напрямку.

Ймовірність включення ребра в маршрут окремої мурахи пропорційна до кількості феромонів на цьому ребрі, а кількість відкладеного феромону пропорційна до довжини маршруту. Чим коротший маршрут, тим більше феромону буде відкладено на його ребрах, отже, більша кількість мурах буде включати його в синтез власних маршрутів. Моделювання такого підходу, що використовує тільки додатній зворотний зв'язок, призводить до передчасної збіжності - більшість мурашок рухається по локально-оптимальному маршруту.

Уникнути цього можна моделюючи від’ємний зворотний зв'язок у вигляді випаровування феромону. Причому, якщо феромон випаровується швидко, то це призводить до втрати пам'яті колонії і забування хороших рішень, з іншого боку, збільшення часу випаровування може призвести до отримання стійкого локального оптимального рішення.

Мураха

Мураха - це програмний агент, який є членом великої колонії і використовується для вирішення певної проблеми. Мураха забезпечується набором простих правил, які дозволяють їй вибирати шлях у графі. Вона підтримує список вузлів, які вже відвідала і може пройти через кожен вузол лише один раз. Шлях між двома вузлами графа, за яким мураха відвідала кожен вузол, називається шляхом Гамільтона.

Вузли в списку "поточної подорожі" розташовуються в тому порядку, в якому їх відвідала мураха. Пізніше список використовується для визначення протяжності шляху між вузлами. Справжня мураха під час переміщення по шляху буде залишати за собою феромони. В мурашиному алгоритмі агент залишає феромони на ребрах графа після завершення подорожі.

Стартова точка, куди поміщається мураха, залежить від обмежень, накладених умовами завдання, оскільки для кожного завдання спосіб розміщення мурашок *є визначальним*. Або всі вони поміщаються в одну точку, або в різні з повтореннями, або без повторень.

На цьому ж етапі задається початковий рівень феромону. Він ініціалізується невеликим додатнім числом для того, щоб на початковому кроці ймовірності переходу в наступну вершину були нульовими.

Рух мурашки

Рух мурашки ґрунтується на простому ймовірнісному рівнянні. Якщо мураха ще не закінчила шлях, тобто не відвідала усі вузли мережі, для визначення наступного ребра шляху використовується рівняння

 (1)

Тут $\tau (r,u)$ - інтенсивність ферменту на ребрі між вузлами r і u, $\eta (r, u)$ - функція, яка представляє вимір зворотньої відстані для грані, α - вага ферменту, а β - коефіцієнт евристики. Параметри α і β визначають відносну значимість двох параметрів, а також їх вплив на рівняння. Оскільки мураха подорожує тільки по вузлах, які ще не були відвідані (як зазначено списком табу), ймовірність обчислюється лише для ребер, які ведуть до ще не відвіданих вузлів. Ці ребра представляє змінна k.

Подорож мурахи

Пройдений мурахою шлях відображається, коли мураха відвідає всі вузли графа. Цикли заборонено, оскільки в алгоритм включено список табу. Після завершення довжина шляху може бути підрахована - вона дорівнює сумі довжин всіх ребер, якими подорожувала мураха. Рівняння (2) показує кількість феромону, який був залишений на кожному ребрі шляху для мурашки k. Змінна Q є константою.

 (2)

Результат рівняння *є засобом вимірювання шляху*, - короткий шлях характеризується високою концентрацією феромонів, а більш довгий шлях - більш низькою. Далі, отриманий результат використовується в рівнянні (3), щоб збільшити кількість феромону вздовж кожного ребра пройденого мурахою шляху.

 (3)

Важливо, що дане рівняння застосовується до всього шляху, при цьому кожне ребро позначається феромоном пропорційно до довжини шляху. Тому слід дочекатися, поки мураха закінчить подорож і лише потім оновити рівні феромону, в іншому випадку справжня довжина шляху залишиться невідомою. Константа p - значення між 0 і 1.

Випаровування феромонів

На початку шляху у кожного ребра є шанс бути обраним. Щоб поступово видалити ребра, які входять в гірші шляхи графа, до всіх ребер застосовується процедура випаровування феромону. Використовуючи константу p з рівняння (3), отримуємо рівняння (4):

 (4)

Для випаровування феромону використовується зворотний коефіцієнт оновлення шляху.

Повторний запуск

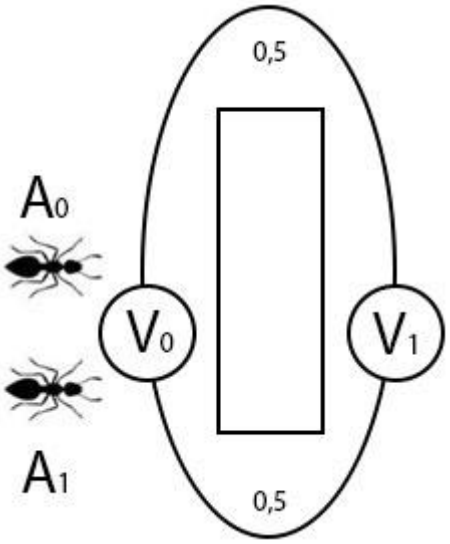
Після того, як шлях мурашки завершено, ребра оновлено відповідно до довжини шляху і сталося випаровування феромону на всіх ребрах, алгоритм запускається повторно. Список табу очищується, і довжина шляху обнулюється. Мурахам дозволяється переміщатися по графу, засновуючи вибір ребра на

рівнянні (1). Цей процес може виконуватися для постійної кількості шляхів або до моменту, коли протягом кількох запусків не було відзначено повторних змін. Потім визначається кращий шлях, який і є рішенням.

Демонстраційний приклад

Щоб побачити, як працюють рівняння, розберемо функціонування розглянутого вище алгоритму на простому прикладі. Візьмемо простий сценарій з двома мурашками з прикладу яке розглянуто вище.

На рисунку показано приклад з двома ребрами між двома вузлами (V_0 і V_1). Кожне ребро ініціалізується і має однакові шанси на те, щоб бути обраним.

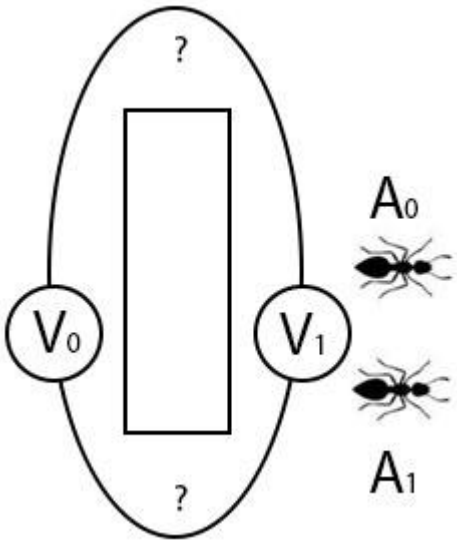


Два мурахи, що знаходяться у вузлі V_0 позначаються як A_0 і A_1. Оскільки ймовірність вибору будь-якого шляху однакова, в цьому циклі проігноруємо рівняння вибору шляху.

Дані для завдання:

- Число пройдених кроків: для A_0 - 20, для A_1 - 10
- Рівень феромону (Q / пройдена відстань): для A_0 - 0.5, A_1 - 1.0
- $\rho = 0.6$
- $\alpha = 0.3$
- $\beta = 1.0$

На наступному рисунку показано, що кожна мураха вибирає свій шлях (мураха A_0 йде по верхньому шляху, а мураха A_1 - по нижньому). Мураха A_0 зробила 20 кроків, а мураха A_1, - тільки 10. За рівнянням (2) розраховуємо кількість феромонів, яке має бути "нанесено".



Примітка: Роботу алгоритму можна змінити, якщо перевизначити його параметри (наприклад, α, β або ρ), наприклад надати більшу вагу феромонам або відстані між вузлами.

Далі за рівнянням (3) обчислюється кількість феромону, яка буде застосовано.

Для мурахи A_0 результат становить: 0,1+(0,5*0,6)=0,4

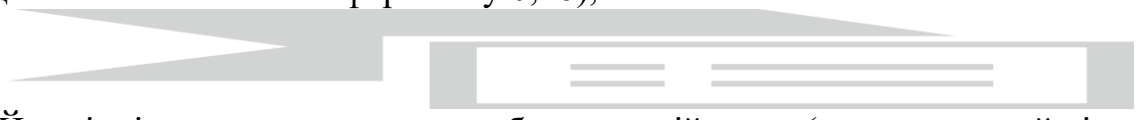
Для мурахи A_1 результат становить: 0,1+(0,5*0,6)=0,4

Далі за допомогою рівняння (4) визначається, яка частина феромонів випарується і, відповідно, скільки залишиться. Результати (для кожного шляху відповідно) становлять:

0,4*(1,0-0,6)=0,16

0,7*(1,0-0,6)=0,28

Ці значення представляють нову кількість феромонів для кожного шляху (верхнього і нижнього, відповідно). Тепер перемістимо мурах назад у вузол V_0 і скористаємося імовірнісним рівнянням вибору шляху (1), щоб визначити, який шлях повинні вибрати мурахи. Ймовірність того, що мураха вибере верхній шлях (представлений кількістю феромону 0,16), становить:



Ймовірність того, що мураха вибере нижній шлях (представлений кількістю феромону 0,28) становить:

При зіставленні двох ймовірностей обидві мурахи виберуть нижній шлях, який є найбільш оптимальним.

Характерні особливості

Для алгоритму мурашиної колонії необхідно вказати:

- Закон виділення феромону.
- Закон випаровування феромону.
- Кількість агентів.
- Місця розміщення.

Ці характеристики вибираються з врахуванням особливостей завдання на основі експериментальних досліджень (евристики).

Алгоритм:

- Реалізує пошук наближених рішень.
- Має поліноміальну складність.
- Є одним з видів імовірнісних алгоритмів (закони виділення випаровування - імовірнісні закони).

Області застосування

Алгоритм оптимізації мурашиної колонії може бути успішно застосований для вирішення складних комплексних завдань оптимізації. Мета вирішення складних комплексних завдань оптимізації - пошук і визначення найбільш відповідного рішення для оптимізації (знаходження мінімуму або максимуму) цільової функції (ціни, точності, часу, відстані тощо) з дискретної множини можливих рішень.

Типовими прикладами вирішення такого завдання є *задача календарного планування, завдання маршрутизації транспорту, різних мереж (GPRS, телефонні, комп'ютерні тощо), розподіл ресурсів та робіт.* Ці задачі виникають у бізнесі, інженерії, виробництві та багатьох інших областях. Дослідження показали, що метод мурашиних колоній може давати результати, навіть кращі ніж при використанні генетичних алгоритмів і нейронних мереж.

Модифікації класичного алгоритму

Результати перших експериментів із застосуванням мурашиного алгоритму для вирішення завдання комівояжера були багатообіцяючими, проте далеко не кращими порівняно з вже існуючими методами. Однак, простота класичного мурашиного алгоритму («мурашиної системи») залишала можливості для доробок

- і саме алгоритмічні вдосконалення стали предметом подальших досліджень фахівців у галузі комбінаторної оптимізації. В основному, ці вдосконалення пов'язано з великим використанням історії пошуку та більш ретельним дослідженням областей навколо вже знайдених вдалих рішень.

Elitist Ant System

Введення в алгоритм так званих «елітних мурах». Досвід показує, що проходячи ребра, що входять в короткі шляхи, мурахи з більшою ймовірністю будуть знаходити коротші шляхи. Ефективною стратегією є штучне збільшення рівня феромонів на найвдаліших маршрутах. Для цього на кожній ітерації алгоритму кожна з елітних мурах проходить шлях, який є найкоротшим із знайдених на даний момент.

Експерименти показують, що, до певного рівня, збільшення числа елітних мурах є досить ефективним, дозволяючи значно скоротити число ітерацій алгоритму. Однак, якщо число елітних мурах занадто велике, то алгоритм досить швидко знаходить субоптимальне рішення і застряє в ньому. Як і інші змінні параметри, оптимальне число елітних мурах слід визначати дослідним шляхом.

Ant-Q

Мурашиний алгоритм, який отримав свою назву за аналогією з методом машинного навчання Q-learning. В основі алгоритму лежить ідея про те, що мурашину систему можна інтерпретувати як систему навчання з підкріпленням. Ant-Q підсилює цю аналогію, запозичуючи багато ідей з Q-навчання.

Алгоритм зберігає Q-таблицю, що співставляє кожному з ребер величину, яка визначає «корисність» переходу по цьому ребру. Q-таблиця змінюється в процесі роботи алгоритму - відбувається навчання системи. Значення корисності переходу по ребру обчислюється виходячи із значень корисності переходу за наступними ребрам в результаті попереднього визначення можливих наступних станів. Після кожної ітерації корисності оновлюються виходячи з довжин шляхів, до складу яких було включено відповідні ребра.

Ant Colony System

Для підвищення ефективності в порівнянні з класичним алгоритмом введено три основних зміни.

По-перше, рівень феромонів на ребрах оновлюється не лише в кінці чергової ітерації, але і при кожному переході мурах з вузла у вузол. По-друге, наприкінці ітерації рівень феромонів підвищується тільки на найкоротшому із знайдених шляхів. По-третє, алгоритм використовує змінене правило переходу: або, з певною часткою ймовірності, мураха безумовно вибирає краще ребро у відповідності до довжини і рівня феромонів, або робить вибір так само, як і в класичному алгоритмі.

Max-min Ant System

Мурашиний алгоритм, в якому підвищення концентрації феромонів відбувається тільки на кращих шляхах з пройдених мурахами. Така велика увага до локальних оптимумів компенсується введенням обмежень на максимальну і мінімальну концентрацію феромонів на ребрах, які вкрай ефективно захищають алгоритм від передчасної збіжності до субоптимальних рішень.

На етапі ініціалізації, концентрація феромонів на всіх ребрах встановлюється рівною максимальній. Після кожної ітерації алгоритму тільки одна мураха залишає за собою слід - або найбільш успішна на даній ітерації, або, аналогічно до алгоритму з елітизмом, елітна. Цим досягається, з одного боку, більш ретельне дослідження області пошуку, з іншого - його прискорення.

ASrank

Модифікація класичного мурашиного алгоритму, в якому в кінці кожної ітерації мурахи ранжуються у відповідно до довжин пройдених ними шляхів. Кількість феромонів, що залишається мурахою на ребрах, таким чином, призначається пропорційно до її позиції. Для ретельного дослідження околу вже знайдених вдалих рішень, алгоритм використовує елітних мурах.

Висновок

Ефективність мурашиних алгоритмів порівнянна з ефективністю загальних метаевристичних методів, а в ряді випадків - і з проблемно-орієнтованими методами. Найкращі результати мурашині алгоритми показують для задач з великою розмірністю областей пошуку і можуть бути успішно застосовані для вирішення складних комбінаторних задач оптимізації. Мурашині алгоритми добре підходять для застосування разом з процедурами локального пошуку, дозволяючи швидко знаходити початкові точки для них.

Найбільш перспективними напрямками подальших досліджень у даному напрямку слід вважати аналіз способу вибору параметрів, що налаштовуються в алгоритмах. В останні роки пропонуються різні способи адаптації параметрів алгоритмів «на льоту». Оскільки від вибору параметрів сильно залежить поведінка мурашиних алгоритмів, саме до цієї проблеми звернено найбільшу увагу дослідників на даний момент.

2. Еволюційні алгоритми

2.1. Еволюційна стратегія

Еволюційна стратегія - евристичний метод оптимізації в розділі еволюційних алгоритмів, який засновано на *адаптації* та *еволюції*.

Еволюційна стратегія схожа з генетичним алгоритмом, але існує декілька суттєвих відмінностей.

Сконцентруємо увагу на найважливіших подібностях і розходженнях між еволюційними стратегіями і генетичними алгоритмами. Очевидно, що **головна подібність** полягає в тому, що обидва методи використовують популяції потенційних рішень і реалізують принцип селекції і перетворення найбільш пристосованих особин. Однак обговорювані підходи сильно відрізняються один від одного.

Перше розходження *полягає в способі представлення особин*. ***Еволюційні стратегії оперують векторами дійсних чисел, тоді як генетичні алгоритми - двійковими векторами***.

Друге розходження між еволюційними стратегіями і генетичними алгоритмами криється *в організації процесу селекції*. При реалізації еволюційної стратегії формується проміжна популяція, що складається з усіх батьків і деякої

кількості нащадків, створених у результаті застосування генетичних операторів. За допомогою селекції розмір цієї проміжної популяції зменшується до величини батьківської популяції за рахунок виключення найменш пристосованих особин. Сформована в такий спосіб популяція утворює чергове покоління. Напроти, у генетичних алгоритмах передбачається, що в результаті селекції з популяції батьків відбирається кількість особин, рівна розмірності вихідної популяції, при цьому деякі (найбільш пристосовані) особи можуть відбиратися багаторазово. У той же час, менш пристосовані особи також мають можливість з'явитися в новій популяції. Однак шанси їхнього відбору пропорційні величині пристосованості особин. Незалежно від застосовуваного в генетичному алгоритмі методу селекції (наприклад, рулетки або рангового) більш пристосовані особи можуть відбиратися багаторазово. *При реалізації еволюційних стратегій особи відбираються без повторень*. В еволюційних стратегіях застосовується детермінована процедура селекції, тоді як у генетичних алгоритмах вона має випадковий характер.

Третє розходження між еволюційними стратегіями і генетичними алгоритмами стосується послідовності виконання процедур селекції і рекомбінації (тобто зміни генів у результаті застосування генетичних операторів). При реалізації еволюційних стратегій *спочатку виконується рекомбінація, а потім селекція*. У випадку виконання генетичних алгоритмів ця послідовність інвертується. При здійсненні застосування еволюційних стратегій нащадок утворюється в результаті схрещування двох батьків і мутації. Формована в такий спосіб проміжна популяція, що складається з усіх батьків і отриманих від них нащадків, надалі піддається селекції, що зменшує розмір цієї популяції до розміру вихідної популяції. При виконанні генетичних алгоритмів спочатку виконується селекція, що приводить до утворення перехідної популяції, після чого генетичні оператори схрещування і мутації застосовуються до особин (обираних з ймовірністю схрещування) і до генів (обираних з ймовірністю мутації).

Наступне розходження між еволюційними стратегіями і генетичними алгоритмами полягає в тому, що параметри генетичних алгоритмів (такі, як ймовірності схрещування і мутації) залишаються сталими протягом усього процесу еволюції, тоді як при реалізації еволюційних стратегій ці параметри піддаються безперервним змінам (так звана самоадаптація параметрів).

Ще одне розходження між еволюційними стратегіями і генетичними алгоритмами стосується трактування обмежень, що накладаються на розв'язувані оптимізаційні задачі. Якщо при реалізації еволюційних стратегій на деякій ітерації нащадок не задовольняє всім обмеженням, то він відкидається і включається в нову популяцію. Якщо таких нащадків виявляється багато, то еволюційна стратегія запускає процес адаптації параметрів, наприклад, шляхом збільшення ймовірності схрещування. У генетичних алгоритмах такі параметри не змінюються. У них може застосовуватися штрафна функція для тих особин, що не задовольняють накладеним обмеженням, однак ця технологія має багато недоліків.

В міру розвитку еволюційних стратегій і генетичних алгоритмів протягом останніх років істотні розходження між ними поступово

зменшуються. Наприклад, при реалізації генетичних алгоритмів для рішення оптимізаційних задач усе частіше застосовується представлення хромосом дійсними числами і різні модифікації «генетичних» операторів, що має на меті підвищити ефективність цих алгоритмів. Подібні методи, що значно відрізняються від класичного генетичного алгоритму, за традицією зберігають колишню назву, хоча більш коректно було б називати їх «еволюційними алгоритмами».

2.2. Еволюційне програмування

Еволюційне програмування було запропоновано доктором Лоуренсом Дж. Фогелем в 1960 році. У той час штучний інтелект було обмежено двома основними напрямками досліджень: моделюванням людського мозку (нейронні мережі) і моделюванням поведінки людини (евристичне програмування). Альтернативний варіант Фогеля відкидав моделювання кінцевого продукту еволюції, і був спрямований на моделювання процесу еволюції, як засобу отримання розумної поведінки. Фогель розглядає інтелект як складову частину здатності робити передбачення зовнішнього середовища у поєднанні з переведенням кожного прогнозу у доцільну відповідь згідно заданої мети (наприклад, для максимізації функції виграшу). *Таким чином, на його думку прогнозування є необхідною умовою для розумної поведінки.*

Гіпотези про вид залежності цільової змінної від інших змінних формулюються системою **у вигляді програм на деякій внутрішній мові програмування.** Якщо це універсальна мова, то теоретично на ній можна виразити залежність будь-якого виду. Процес побудови таких програм будується як еволюція в популяції програм. Якщо система знаходить програму, яка точно відтворює шукану залежність, вона починає вносити до неї невеликі модифікації і відбирає серед дочірніх програм лише ті, які підвищують точність.

Система "вирощує" кілька генетичних ліній програм, що конкурують між собою в точності знаходження шуканої залежності. Спеціальний транслюючий модуль перекладає знайдені залежності з внутрішньої мови системи на зрозумілу користувачеві мову (математичні формули, таблиці тощо), роблячи їх легкодоступними. Для того, щоб зробити отримані результати більш зрозумілими для користувача-нематематика, існує великий арсенал різноманітних засобів візуалізації виявлених залежностей.

Пошук залежності цільових змінних від інших факторів проводиться у формі функцій певного виду. Наприклад, в одному з найбільш вдалих алгоритмів цього типу - **методі групового урахування аргументів (МГУА)** залежність шукають у формі поліномів. Причому складні поліноми заміняються кількома простими, враховують лише деякі ознаки (групи аргументів). Отримані формули залежностей надаються до аналізу та інтерпретації.

Відродження еволюційного програмування було продовжено в 1980-х. розробки стосувалися довільного представлення даних і узагальненої проблеми оптимізації.

Еволюційне програмування було застосоване до різних інженерних завдань:

- Системи управління, системи ідентифікації.
- Маршрутизація трафіку.
- Військове планування.
- Обробка сигналів.
- Ігрові та навчальні програми.

Познайомимося зі стандартним алгоритмом еволюційного програмування.

Вихідна популяція рішень вибирається випадковим чином. У задачах оптимізації значень дійсних чисел (прикладом яких може служити навчання нейронних мереж) особа (хромосома) представляється ланцюгом значень дійсних чисел. Ця популяція оцінюється щодо заданої функції (функції пристосованості). Нащадки утворюються від вхідних у цю популяцію батьків у результаті випадкової мутації. Селекція заснована на ймовірнісному виборі (турнірний метод), при якому кожне рішення змагається з хромосомами, випадковим чином обираними з популяції. Рішення-переможці (які є найкращими) стають батьками для наступного покоління. Описана процедура повторюється так довго, поки не буде знайдено шукане рішення або не буде вичерпаний ліміт машинного часу.

Еволюційне програмування застосовується для оптимізації функціонування нейронних мереж. Так, як і інші еволюційні методи, воно не вимагає градієнтної інформації і тому може використовуватися для рішення задач, у яких ця інформація недоступна, або для її одержання потрібні значні обсяги обчислень. Одними з перших додатків еволюційного програмування вважаються задачі теорії штучного інтелекту, а самі ранні роботи стосувалися теорії кінцевих автоматів.

Спостерігається велика подібність між еволюційними стратегіями й еволюційним програмуванням у їхніх додатках до задач оптимізації безперервних функцій з дійсними значеннями. Деякі дослідники стверджують, що ці процедури, по суті, однакові, хоча вони і розвивалися незалежно одна від одної. Дійсно, обидва методи схожі на генетичні алгоритми. ***Принципове розходження між ними полягає в тому, що еволюційне програмування не зв'язане з конкретною формою представлення особин, оскільки оператор мутації не вимагає застосування якого-небудь спеціального способу кодування***.

2.3. Генетичне програмування

Генетичне програмування – *це по суті методика машинного навчання*, прототипом якої є біологічна еволюція. У загальному випадку обчислення починаються з великого набору програм (популяції), згенерованих випадковим чином або написаних вручну, про які відомо, що це досить хороші рішення. Потім ці програми конкурують між собою в спробі вирішити деяке поставлене користувачем завдання. Це може бути гра, в якій програми змагаються між собою

безпосередньо, або спеціальний тест, покликаний визначити, яка програма краще. По завершенні змагання складається ранжований список програм - від найкращої до найгіршої.

Далі вступає в справу еволюція - найкращі програми копіюються і модифікуються одним із двох способів. Найпростіший називається *мутацією*; в цьому випадку деякі частини програми випадковим чином і дуже незначно змінюються в надії, що від цього рішення стане краще.

Інший спосіб називається *схрещуванням* (або кросовером) - частина однієї з відібраних програм переміщається в іншу. В результаті процедур копіювання та модифікації створюється багато нових програм, які засновані на найкращих особинах попередньої популяції, але не збігаються з ними. На кожному етапі якість програм обчислюється за допомогою функції виживання (fitness function). Оскільки розмір популяції не змінюється, програми, які виявилися гіршими, видаляються з популяції, звільняючи місце для нових.

Створюється нова популяція, яка називається «наступним поколінням», і весь процес повторюється. Оскільки зберігаються і змінюються самі кращі програми, то є надія, що з кожним поколінням вони будуть вдосконалюватися, як діти, які можуть перевершити своїх батьків. Нові покоління створюються доти, поки не буде виконана умова завершення, яке в залежності від завдання може формулюватися одним із таких способів:

- Знайдено ідеальне рішення.
- Знайдено достатньо хороше рішення.
- Рішення не вдається поліпшити протягом кількох поколінь.
- Кількість поколінь досягла заданої межі.

Для таких завдань, як визначення математичної функції, що відображає один набір значень на інший, можна знайти ідеальне рішення. Для інших, наприклад коли йдеться про настільні ігри, знайдене рішення може бути не ідеальним, оскільки його якість залежить від стратегії супротивника.

Блок-схема, зображена на рисунку, надає уявлення про процес генетичного програмування.

Розглянемо *узагальнений приклад еволюційної програми*. Припустимо, що шукається граф, що задовольняє певним обмеженням (наприклад, виконується пошук топології комунікаційної мережі, оптимальної за конкретними критеріями, наприклад, по вартості передачі і т.п.). Кожна особа в еволюційній програмі представляє одне з потенційних рішень, тобто в даному випадку деякий граф. Вихідна популяція графів P(0), сформована випадковим чином або створювана при реалізації якого-небудь евристичного процесу, вважається відправною точкою (k=0) еволюційної програми. Функція пристосованості, що зазвичай задається, зв'язана із системою обмежень розв'язуваної задачі. Ця функція визначає «пристосованість» кожного графа шляхом виявлення «кращих» і «гірших» особин. Можна запропонувати кілька різних операторів мутації, призначених для трансформації окремих графів, і трохи операторів схрещування, що будуть створювати новий граф у результаті рекомбінації структур двох або більш графів. Дуже часто такі оператори обумовлюються характером

розв'язуваної задачі. Наприклад, якщо шукається граф типу «дерево», то можна запропонувати оператор мутації, що видаляє галузь з одного графа і додає нову галузь, що поєднує два окремих підграфи. Інші можливості полягають у проектуванні мутації незалежно від семантики задачі, але з включенням у функцію пристосованості додаткових обмежень - «штрафів» для тих графів, що не є деревами.

Принципову різницю між класичним генетичним алгоритмом і еволюційною програмою, тобто еволюційним алгоритмом у широкому змісті, ілюструють рис. 3.82 і 3.83.

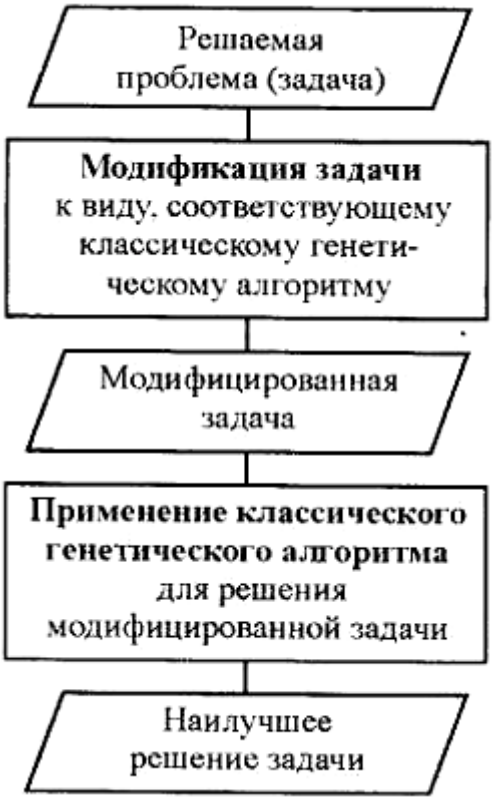


Рис. 3.82. Рішення задачі за допомогою класичного генетичного алгоритму

Класичний генетичний алгоритм, що оперує двійковими послідовностями, вимагає представити розв'язувану задачу в строго визначеному виді (відповідність між потенційними рішеннями і двійковими кодами, декодування і т.п.). Зробити це не завжди просто.

Рис. 3.83. Рішення задачі за допомогою еволюційного алгоритму (еволюційної програми)

Еволюційні програми можуть залишити постановку задачі в незмінному виді за рахунок модифікації хромосом, що представляють потенційні рішення (з використанням «природних» структур даних), і застосування відповідних «генетичних» операторів. Іншими словами, для рішення нетривіальної задачі можна або перетворити її до виду, необхідному для використання генетичного алгоритму (рис. 3.82), або модифікувати генетичний алгоритм так, щоб він задовольняв задачі (рис. 3.83).

При реалізації першого підходу застосовується класичний генетичний алгоритм, а при реалізації другого підходу - еволюційна програма Таким чином, модифіковані генетичні алгоритми можна в загальному випадку називати еволюційними програмами. Однак найчастіше зустрічається термін еволюційні алгоритми. Еволюційні програми також можуть розглядатися як еволюційні алгоритми, підготовлені програмістом для виконання на комп'ютері. Основна задача програміста полягає при цьому у виборі відповідних структур даних і «генетичних» операторів. Саме таке трактування поняття еволюційна програма представляється найбільш обґрунтованою.

Усі поняття, застосовувані в цьому розділі і стосовні головним чином до методів, заснованим на еволюційному підході, можна зіставити головному напрямкові досліджень - комп'ютерному моделюванню еволюційних процесів. Ця область інформатики називається Evolutionary Computation, що можна перевести як еволюційні обчислення.

До еволюційних алгоритмів також застосовується поняття технологія еволюційних обчислень. Можна додати, що назва генетичні алгоритми використовується як у вузькому змісті, тобто для позначення класичних

генетичних алгоритмів і їхніх несуттєвих модифікацій, так і в широкому змісті - припускаючи будь-які еволюційні алгоритми, що значно відрізняються від «класики».

Останнім часом з'явилося багато нових методів, заснованих на еволюційному моделюванні, але базові технології, що використовуються - головним чином класичний генетичний алгоритм, еволюційні стратегії й еволюційне програмування.

3. Еволюційні алгоритми в нейронних мережах

Об'єднання генетичних алгоритмів і нейронних мереж відомо в літературі під абревіатурою COGANN (Combinations of Genetic Algorithms and Neural Networks). Це об'єднання може бути допоміжним (supportive) або рівноправним (collaborative). Допоміжне об'єднання двох методів означає, що вони застосовуються послідовно один за одним, причому один з них служить для підготовки даних, використовуваних при реалізації другого методу. При рівноправному об'єднанні обидва методи застосовуються одночасно. Класифікація цих типів об'єднань генетичних алгоритмів і нейронних мереж представлена в табл. 3.6.

Необхідно відзначити, що відповідно до зауважень, приведених в п. 7, термін «генетичні алгоритми» застосовується тут у більш широкому змісті, чим класичний генетичний алгоритм.

Таблиця 3.6. Об'єднання генетичних алгоритмів і нейронних мереж

Вид объединения	Характеристика объединения	Примеры использования
	Генетические алгоритмы и нейронные сети независимо применяются для решения одной и той же задачи	Однонаправленные нейронные сети, сети Кохонена с самоорганизацией и генетические алгоритмы в задачах классификации
Вспомогательное	Нейронные сети для обеспечения генетических алгоритмов	Формирование исходной популяции для генетического алгоритма
	Генетические алгоритмы для обеспечения нейронных сетей	Анализ нейронных сетей
		Подбор параметров либо преобразование пространства параметров
		Подбор параметров либо правила обучения (эволюция правил обучения)
Равноправное	Генетические алгоритмы для обучения нейронных сетей	Эволюционное обучение сети (эволюция весов связей)
	Генетические алгоритмы для выбора топологии нейронной сети	Эволюционный подбор топологии сети (эволюция сетевой архитектуры)
	Системы, объединяющие адаптивные стратегии генетических алгоритмов и нейронных сетей	Нейронные сети для решения оптимизационных задач с применением генетического алгоритма для подбора весов сети
		Реализация генетического алгоритма с помощью нейронной сети
		Применение нейронной сети для реализации оператора скрещивания в генетическом алгоритме

Незалежне застосування генетичних алгоритмів і нейронних мереж

Генетичні алгоритми і нейронні мережі можуть незалежно застосовуватися для рішення однієї і тієї ж задачі. Цей підхід ілюструється на рис. 3.12.

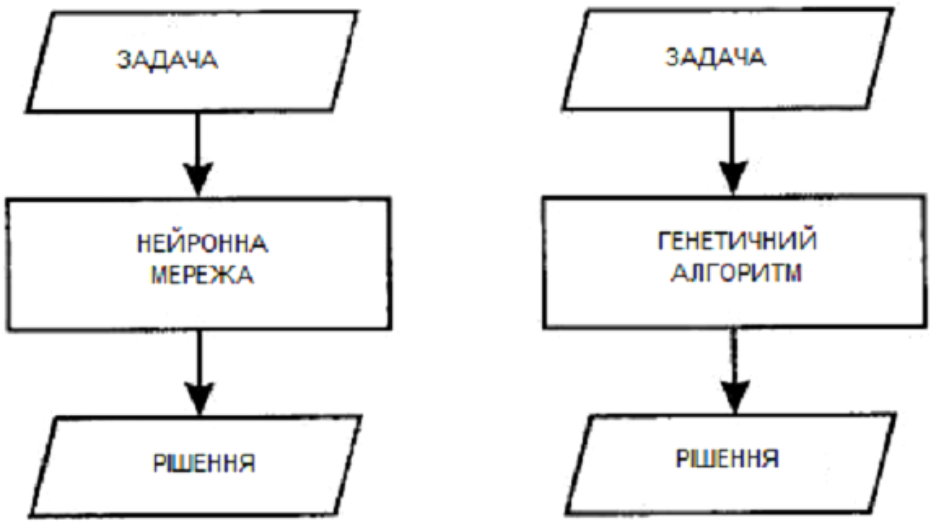


Рис. 3.12. Генетичний алгоритм і нейронна мережа незалежно застосовуються для рішення однієї і тієї ж задачі.

Наприклад, описані незалежні застосування нейронних мереж, генетичних алгоритмів і алгоритму KNN «найближчий сусід» (K - means nearest neighbour) для рішення задач класифікації. У літературі приводяться порівняння тришарової односпрямованої нейронної мережі з навчанням по методу зворотного поширення похибки (навчання з учителем), мережі Кохонена із самоорганізацією (навчання без учителя), системи класифікації, заснованої на генетичному алгоритмі, а також алгоритму KNN «найближчий сусід». Автори ряду робіт вважають незалежне застосування цих методів для рішення задачі автоматичної класифікації результатів ЭМГ (электроміографія - реєстрація електричної активності м'язів) допоміжним об'єднанням. Відомі й інші роботи, у яких порівнюються можливості застосування різних методів (зокрема, генетичних алгоритмів і нейронних мереж) для рішення тих самих задач. Прикладом задачі, яку можна вирішити за допомогою як нейронної мережі, так і генетичного алгоритму, може служити задача про комівояжера.

Нейронні мережі для підтримки генетичних алгоритмів

Більшість дослідників вивчали можливості застосування генетичних алгоритмів для забезпечення роботи нейронних мереж. До нечисленних зворотних випадків відноситься гібридна система, призначена для рішення задачі трасування, що класифікується як приклад допоміжного об'єднання нейронних мереж і генетичних алгоритмів. У цій системі генетичний алгоритм використовується в якості оптимізаційної процедури, призначеної для знаходження найкоротшого шляху. Нейронна мережа застосовується при формуванні вихідної популяції для генетичного алгоритму. Цей підхід схематично ілюструється на рис. 3.133.

Рис. 3.133. Допоміжне об'єднання нейронної мережі з генетичним алгоритмом.

Генетичні алгоритми для підтримки нейронних мереж

Підхід, заснований на використанні генетичного алгоритму для забезпечення роботи нейронної мережі, схематично представлено на рис. 3.13.

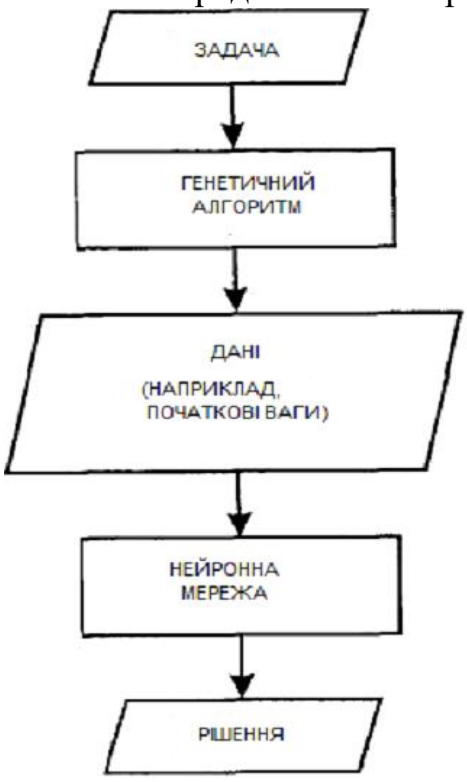


Рис. 3.13. Допоміжне об'єднання генетичного алгоритму з нейронною мережею.

Відомо багато робіт, присвячених подібному об'єднанню методів. *Можна виділити три області проблем:*

- застосування генетичного алгоритму для підбору параметрів або перетворення простору параметрів, використовуваних нейронною мережею для класифікації;
- застосування генетичного алгоритму для підбору правила навчання або параметрів, керуючих навчанням нейронною мережею;
- застосування генетичного алгоритму для аналізу нейронної мережі.

Дві перші області додатка генетичних алгоритмів у нейронних мережах, узагалі говорячи, дозволяють поліпшувати функціонування мереж (тобто вирішують проблему синтезу), тоді як третя служить для аналізу їхнього функціонування. Почнемо обговорення з останньої позиції.

Аналіз нейронних мереж. Деякі дослідники застосовували генетичні алгоритми як допоміжний інструмент для з'ясування закономірностей функціонування нейронних мереж або аналізу ефективності їхньої роботи. Генетичний алгоритм використовувався для побудови «інструментальної системи», що полегшує розуміння функціонування мережі - попросту говорячи, для з'ясування, що і чому робить мережа. Таке розуміння необхідне для того, щоб нейромережевий класифікатор не сприймався в якості «чорної шухляди», що формує відповідь якимось таємничим чином, і щоб рішення по класифікації об'єктів були поясненими. Подібний «інструментарій» (explanation facilities) використовується в більшості експертних систем. Побудова цих інструментів для їхнього застосування в нейронних мережах вважається більш масштабною проблемою, що відноситься до аналізу мереж. Генетичний алгоритм застосовувався для побудови так званих кодових векторів (codebook vectors), що представляють собою вхідні сигнали, при яких функція активації конкретного вихідного нейрона мережі приймає максимальне або близьке до нього значення. Вхідні вектори представлялися в хромосомах множиною дійсних чисел від 0,0 до 1,0. Аналізувалася нейронна мережа, яка призначена для рішення задачі класифікації. Аналогічний підхід застосовувався для мережі ART1 (часткового випадку ART із двійковими вхідними сигналами). За допомогою генетичного алгоритму також проводився аналіз нейронної мережі, використовуваної як модель асоціативного запам'ятовуючого пристрою. Приведені приклади характеризують допоміжне об'єднання генетичних алгоритмів і нейронних мереж, хоча і не можуть вважатися типовими стосовно схеми, представленої на рис. 3.13.

Підбір параметрів або перетворення простору параметрів.

Генетичний алгоритм використовується при підготовці даних для нейронної мережі, яка грає роль класифікатора. Ця підготовка може виконуватися шляхом перетворення простору параметрів або виділенням деякого підпростору, що містить необхідні параметри.

Перший з цих методів, так називане перетворення простору параметрів, застосовується найчастіше в алгоритмах типу «найближчий сусід», хоча відомі

також його доданки в нейромережних класифікаторах. Другий підхід полягає у виділенні підмножини параметрів, що враховуються. Виявляється, що обмеження множини параметрів часто поліпшує функціонування нейронної мережі як класифікатора і, до того ж, скорочує обсяги обчислень. Подібне обмеження множини що враховуються нейронною мережею параметрів застосовувалося, зокрема, для контролю сценаріїв подій на ядерних об'єктах, а також для розпізнавання китайських ієрогліфів. Відомі й інші приклади підготовки даних для нейронних мереж за допомогою генетичних алгоритмів.

Підбір параметрів і правил навчання. Генетичний алгоритм також застосовується для підбора параметрів навчання - найчастіше швидкості навчання (learning rate) і так називаного моменту для алгоритму зворотного поширення похибки. Таке адаптивне уточнення параметрів алгоритму зворотного поширення (вони кодуються в хромосомах) у результаті еволюції може розглядатися як перша спроба еволюційної модифікації правил навчання. Замість безпосереднього застосування генетичного алгоритму для підбору параметрів навчання розвивається еволюційний підхід, спрямований на побудову оптимального правила (алгоритму) навчання.

Помітимо, що еволюційна концепція вже може розглядатися як перехід від допоміжного до рівноправного об'єднання генетичного алгоритму і нейронних мереж.

Застосування генетичних алгоритмів для навчання нейронних мереж

Думка про те, що нейронні мережі можуть навчатися за допомогою генетичного алгоритму, висловлювалася різними дослідниками. Перші роботи на цю тему стосувалися застосування генетичного алгоритму як метод навчання невеликих односпрямованих нейронних мереж, але в наступному було реалізовано застосування цього алгоритму для мереж з більшою розмірністю.

Як правило, задача полягає в оптимізації ваг нейронної мережі, що має апріорі задану топологію. Ваги кодуються у виді двійкових послідовностей (хромосом). Кожна особа популяції характеризується повною множиною ваг нейронної мережі. Оцінка пристосованості особин визначається функцією пристосованості, що задається у виді суми квадратів похибок, тобто різностей між очікуваними (еталонними) і фактично одержуваними значеннями на виході мережі для різних вхідних даних.

Приведемо два найважливіших аргументи на користь застосування генетичних алгоритмів для оптимізації ваг нейронної мережі. Насамперед, генетичні алгоритми забезпечують глобальний перегляд простору ваг і дозволяють уникати локальні мінімуми. Крім того, вони можуть використовуватися в задачах, для яких інформацію про градієнти одержати дуже складно або вона виявляється занадто дорогою.

Генетичні алгоритми для вибору топології нейронних мереж

Як найбільш очевидний спосіб об'єднання генетичного алгоритму з нейронною мережею інтуїтивно сприймається спроба закодувати в генотипі топологію об'єкта (у розглянутому випадку - нейронної мережі) із указівкою кількості нейронів і зв'язків між ними при наступному визначенні ваг мережі за допомогою будь-якого відомого методу.

Проектування оптимальної топології нейронної мережі може бути представлене у виді пошуку такої архітектури, яка забезпечує найкраще (щодо обраного критерію) рішення конкретної задачі. Такий підхід припускає перебір простору архітектури, складеного з усіх можливих варіантів, і вибір точки цього простору, найкращої щодо заданого критерію оптимальності.

З урахуванням достоїнств еволюційного проектування архітектури в останні роки була виконана велика кількість досліджень, у яких основна увага приділялася еволюції з'єднань нейронної мережі, тобто кількості нейронів і топології зв'язків між ними. Лише в деяких роботах розглядалася еволюція функцій переходів, хоча ці функції вважаються важливим елементом архітектури і впливають на функціонування нейронної мережі.

Також, як і у випадку еволюційного навчання, перший крок еволюційного проектування архітектури полягає у формуванні вихідної множини розглянутих варіантів.

Адаптивні взаємодіючі системи

До рівноправного об'єднання генетичних алгоритмів і нейронних мереж варто віднести комбінацію адаптивних стратегій обох методів, що складає єдину адаптивну систему. Можна привести три приклади систем такого типу.

Перший з них - це нейронна мережа для оптимізаційної задачі з генетичним алгоритмом для визначення ваг мережі.

Другий приклад відноситься до реалізації генетичного алгоритму за допомогою нейронної мережі. У цьому випадку нейронні підсистеми застосовуються для виконання генетичних операцій репродукції і схрещування.

У третьому прикладі, трохи схожому на попередній, нейронна мережа також застосовується як оператор схрещування в генетичному алгоритмі, призначеному для

рішення оптимізаційних задач.

Представлені приклади стосуються такого рівноправного об'єднання генетичних алгоритмів і нейронних мереж, що у результаті дозволяє одержати більш ефективний алгоритм, який поєднує кращі якості обох методів.

Типовий цикл еволюції

Як тільки визначений вид еволюції вводиться в штучну нейронну мережу, відразу виникає потреба у відповідній йому схемі хромосомного представлення даних, тобто повинний бути створений спосіб генетичного кодування особин популяції. Розробка способу кодування вважається першим етапом такого

еволюційного підходу, поряд з яким типовий процес еволюції включає наступні кроки:

- декодування;
- навчання;
- оцінювання пристосованості;
- репродукція;
- формування нового покоління.

Приведена на рис. 3.12 блок-схема зберігає свою актуальність, оскільки (як уже згадувалося в п. 3.18) вона відображає і класичний генетичний алгоритм, і так називані еволюційні програми, що засновані на генетичному підході й узагальнюють його принципи. Отже, цій універсальній блок-схемі відповідають різні еволюційні алгоритми, і в кожнім з них у першу чергу повинна бути згенерована вихідна популяція хромосом. За аналогією з класичним генетичним алгоритмом ініціалізація (тобто формування цієї вихідної популяції) полягає у випадковому виборі необхідної кількості хромосом, які включаються в неї, що припускає відповідне генетичне кодування кожної особи. У класичному генетичному алгоритмі хромосоми представляються тільки двійковими послідовностями. При еволюційному підході вибір способу кодування являє собою важливу й актуальну задачу.

Далі відповідно до типового циклу еволюції випливає необхідність декодування кожної особи (хромосому) вихідної або поточної популяції для того, щоб одержати множину рішень (фенотипів) даної задачі. У випадку еволюції ваг, архітектур і/або правил навчання фенотипи представляють відповідно множини ваг, архітектур і правил навчання.

Згодом відповідно до генетичного алгоритму розраховуються значення функції пристосованості особин вихідної (або поточної) популяції. При нейромережному підході після декодування хромосом виходить множина нейронних мереж, для яких степінь пристосованості визначається за результатами навчання цих мереж.

При реалізації типового циклу еволюції необхідно сконструювати множину відповідних нейронних мереж (фенотипів):

- мережі з фіксованою архітектурою і множиною закодованих хромосомами ваг - у випадку еволюції ваг;
- мережі з закодованою хромосомами архітектурою - у випадку еволюції архітектури;
- мережі з випадково згенерованими архитектурами і початковими вагами - у випадку еволюції правил навчання.

Після навчання оцінюється пристосованість кожної особи, що входить у поточну популяцію. Помітимо, що також як і в прикладі максимізації функції (приклад 3.5), для оцінювання пристосованості хромосом необхідно їх спочатку декодувати і лише потім розрахувати значення функції пристосованості особин по їхніх фенотипах.

Наступний крок генетичного алгоритму - це селекція хромосом. Вибираються хромосоми, що підлягають репродукції, тобто формується батьківський пул, особи якого в результаті застосування генетичних операторів

сформують популяцію нащадків. Селекція може бути заснована на методі рулетки або будь-якому іншому, наприклад, по алгоритму Уітлі (Whitley). Згідно з цими методами селекція виконується з ймовірністю, пропорційною пристосованості хромосом, або відповідно до їх рангу (при використанні рангового методу). Під репродукцією в даному випадку розуміється процес відбору (селекції) і копіювання (розмноження) хромосом для формування з них перехідної популяції (батьківського пула), особи якої будуть піддаватися впливові генетичних операторів схрещування, мутації і, можливо, інверсії.

Застосування генетичних операторів за обраним методом селекції хромосом відбувається аналогічно класичному генетичному алгоритмові, причому ці оператори можуть відрізнятися від схрещування і мутації базового алгоритму. Як відзначалося в п. 3.18, для конкретної задачі генетичні оператори можуть визначатися в індивідуальному порядку.

Також як і в класичному генетичному алгоритмі, у результаті застосування генетичних операторів з обраним методом селекції хромосом формується нова популяція особин (нащадків). Наступні кроки алгоритму повторюються для чергової популяції аж до виконання умови завершення генетичного алгоритму. На кожній ітерації формується нове покоління нащадків.

Найкраща особа з останнього покоління вважається шуканим рішенням даної задачі. У такий спосіб виходить найкраща множина ваг, найкраща архітектура або найкраще правило навчання.

4. Еволюційні моделі

Оскільки еволюція є основним механізмом обробки інформації в природних системах, **дослідники прагнуть побудувати теоретичні та комп'ютерні моделі, що реально пояснюють принципи роботи цього механізму**. Для досліджень цього напрямку характерне розуміння, що моделі повинні відтворювати структуру популяції, народження і зникнення особин, а також події між ними.

Найчастіше залучаються наступні концепції:

Ройовий інтелект. Описує колективну поведінку децентралізованої самоорганізованої системи. Розглядається в теорії штучного інтелекту як метод оптимізації. Системи ройового інтелекту, як правило, складаються з множини агентів (багатоагентна система), що локально взаємодіють між собою і з навколишнім середовищем. Самі агенти, зазвичай, є достатньо простими, але всі разом, локально взаємодіючи, створюють так званий ройовий інтелект. Прикладом в природі може служити колонії мурашок, рій бджіл, зграя птахів, риб.

Соціологічний напрямок. Людське суспільство представляє реальний інструмент обробки інформації, який добре піддається спостереженню і задокументований (на відміну від людського мозку). Якщо ройовий інтелект орієнтовано на отримання складної поведінки в системі з простих елементів, цей підхід, навпаки, досліджує побудову простих і спеціальних структур на базі

складних і універсальних об'єктів: «спільнота є менш розумна, ніж більшість її членів».

Для цього напрямку характерне прагнення дати соціологічним поняттям визначення з області інформатики. Наприклад, еліта визначається як носій певної приватної моделі реального світу, а базис (тобто народ) грає роль арбітра між елітами. Еволюційний процес полягає в народженні і загибелі еліт. Базис не в змозі розібратися в суті ідей і моделей, що представляються елітами, і не ставить перед собою такого завдання. Однак, в силу своєї незалученості зберігає здатність до ясної емоційної оцінки, що дозволяє йому легко відрізняти харизматичні еліти від загниваючих, що намагаються зберегти свої привілеї, розуміючи, що їхня ідея або модель не підтвердилася.

Колективний інтелект. Термін з'явився в середині 1980-х років в соціології при вивченні процесу колективного прийняття рішень. Дослідники визначили колективний інтелект як здатність групи знаходити вирішення завдань більш ефективним, ніж найкраще індивідуальне рішення в цій групі.

Досягнення людства народжені колективним розумом. *Люди - нейрони цивілізації.* Роблячи кожен свою справу, удосконалюючись в ньому і обмінюючись результатами своєї праці, ми навчилися створювати речі, які навіть не розуміємо. У своєму есе «Я, Олівець» економіст Леонард Рід зауважує, що жодна людина на світі не знає, як виготовити олівець, - це знання розподілено між тисячами шахтарів, дроворубів, дизайнерів і фабричних робітників.

Великі проекти не робляться поодинці. З часів будівництва єгипетських пірамід результат досягався умінням організувати спільну роботу великої кількості людей. *У війні та на виробництві добре зарекомендували себе ієрархічні схеми, а інформаційні технології за останню чверть століття довели їх практично до досконалості. Однак у сфері інновацій все залежить від особистого досвіду і творчого інтелекту окремої людини.* Звідси і ризики, що пов'язані з астрономічними гонорарами зіркових топ-менеджерів, які своєю інтуїцією повинні компенсувати відсутність технологій для роботи з прихованим знанням персоналу.

Діяльність колективного інтелекту, організована за аналогією з нейронною мережею, обіцяє досягнення якісно нових результатів.

Але як ефективно задіяти інтелект великого числа людей? Ради та комітети не бувають розумнішими і оригінальніше за своїх креативних і компетентних членів. Великі експертні групи обережні, повільні і неоригінальні. Виходить, інтелект, який розуміють як здатність до пошуку нестандартних рішень, неадитивний?

Так вважалося до недавнього часу.

Кожен з нас - багаторівнева структура з безлічі біологічних елементів, у тому числі відповідальних за інформаційні потоки. Кожна клітина, що бере участь в інформаційному потоці, може мати багато інформаційних контактів. Кількість інформаційних зв'язків обумовлюють інтегровану функцію розуму порівнюють з кількістю піщинок в Сахарі, а кількість можливих комбінацій інформаційних зв'язків всередині цього людського колективного розуму - з кількістю атомів у Всесвіті.

Чи можливе щось подібне, в сенсі колективного розуму, для спільноти особин? З чисто теоретичної точки зору це можливо. Потрібно лише поширити міжклітинні інформаційні зв'язки на весь колектив особин. А от чи є в природі передавач інформації, здатний виконати подібну функцію, - це вже інше питання. Телепатію, якщо вона є, якось сумнівно навіть гіпотетично прив'язати до контактів між окремими клітинами, щоб забезпечити функціонування деякого єдиного колективного суперінтелекту з єдиною функцією свідомості.

Надію на прорив дає аналогія з мозком. Якщо взаємозв'язок нейронів породжує індивідуальний людський інтелект, то, ймовірно, і людей можна організувати в колективний інтелект, який буде сильніше за кожний індивідуальний. Треба тільки знайти правильну архітектуру, що дозволяє задіяти для розвитку бізнесу приховане знання, яке не можна перенести з голів людей в корпоративні бази даних. Розробка таких соціальних бізнес-додатків визначатиме розвиток інформаційних технологій на найближче десятиліття.

У пошуках архітектури колективного інтелекту сформувалися два напрямки, які можна умовно назвати **соціальним і семантичним**.

Соціальний напрямок. Увага концентрується на людях, об'єднаних в соціальну мережу, яка в ході комунікації виявляє ідейних лідерів і допомагає вдосконалювати їх пропозиції. На цьому шляху з'явилися методи мозкового штурму, форсайта, організаційних ігор, проте вони нестабільні за продуктивністю і не масштабуються за кількістю учасників.

Семантичний напрямок. Ставить в центр повідомлення, що несе ідею. По зв'язках між повідомленнями вибудовується семантична мережа, аналіз якої виявляє ключові ідеї. Так будується система наукових публікацій, індекси пошукових систем, рейтинги повідомлень в блогах. Але семантичні мережі схильні до різних типів нестійкості - накрутки, флейм, флешмоб.

Стабілізація і масштабування досягаються за рахунок об'єднання соціальних та семантичних мереж. Вже за наявності найпростіших технічних засобів їх інтеграції (скажімо, «френди + коменти») виникають колективні протоінтелектуальні явища, такі як медіавіруси і «розумні натовпи». В їх основі лежить поширення інформації на хвилі сильних емоцій, і це вже знайшло застосування в комерції та політиці.

Дещо ближче до колективного інтелекту знаходяться приклади **краудсорсингу**, коли компанії залучають допомогу тисяч людей в обмін на призи або участь в доходах. Ще складніше колективна інтелектуальна діяльність по створенню вільного ПЗ та інструментів агрегації інформації, наприклад, сайтів або ринків передбачень. Тут скромні вклади учасників обмінюються на повагу спільноти чи інший виграш.

Краудсорсінг в даний час активно розвивається в якості моделі для вирішення будь-якого виду проблем і завдань, що стоять як перед бізнесом, так і перед державою і суспільством в цілому. В рамках парадигми краудсорсінгу рішення задачі передається розподіленій і дуже численній групи людей, за рахунок чого вартість і час досягнення результату радикально знижуються.

Приклади краудсорсінгових проектів:

Вікіпедія - електронна енциклопедія, що створюється переважно силами волонтерів.

InnoCentive - компанія, що запрошує вчених за конкурсну винагороду від $ 10 тис. до $ 100 тис. вирішувати завдання, які ставлять такі компанії, як Procter & Gamble, DuPont і BASF.

Threadless - компанія з виробництва футболок з Чикаго, процес розробки дизайну складається виключно з проведення онлайн-конкурсу, переможці щотижневого конкурсу отримують $ 2 тис. І їх робота запускається у виробництво.

Muji - японська меблева компанія, через свій корпоративний сайт збирає ідеї для своїх виробів і приймає рішення про запуск у виробництво за результатами конкурсу.

eBird - проект, який використовує ресурси любителів для спостереження за птахами.

NASA Clickworkers - проект NASA, створений з метою аналізу масиву знімків марсіанської поверхні силами астрономів-аматорів.

Peer-to-Patent - американський проект, заснований на принципі спільної роботи: державне патентне бюро на постійній основі працює з відкритою Інтернет-спільнотою, в розгляді заявок на патенти бере участь мережа волонтерів (вчені, технічні фахівці, люди, чия кваліфікація дозволяє брати участь у процесі патентування).