Дніпропетровський Національний Університет

кафедра математичного забезпечення ЕОМ

Курсова робота з теми

"МЕТОДИ АДАПТИВНОГО ПРОЕКТУВАННЯ НЕЙРОННИХ МЕРЕЖ ПРЯМОГО РОЗПОВСЮДЖЕННЯ З КАСКАДНОЮ АРХІТЕКТУРОЮ"

Виконав

Студент групи ПЗ-09-м

Додатко О.В

Керівник

доцент кафедри МЗ ЕОМ

Кузнєцов К.А.

Дніпропетровськ 2009 р

Оглавление

Вступ

Дану роботу присвячено розробці інструменту для автоматичного підбору архітектури нейронних мереж за допомогою еволюційних алгоритмів. Розглянуто альтернативні способи представлення архітектури та алгоритми навчання нейронних мереж каскадного типу.

З появою потужних обчислювальних машин, що здатні виконувати мільйони операцій в секунду, з‘явилась можливість втілювати в життя складні алгоритми вирішення тих чи інших задач реального світу. Однак, традиційна обчислювальна техніка не надто добре пристосована до розв’язання таких задач як розпізнавання образів, класифікація, прогнозування, ітд. Тому ведуться розробки альтернативних способів обчислень. Ідеї багатьох із них запозичені у природи : нейронні мережі, генетичні алгоритми, L-системи ітд.

Багато задач можуть бути розв’язані з використанням нейронних мереж. Однією із найскладніших проблем при використанні такого підходу є вибір правильної архітектури мережі – кількості нейронів, їхніх активаційних функцій, наявності зв’язків між ними.

Побудова архітектури мережі вручну є прийнятною для невеликих за обсягом задач або для задач із вже відомим розв’язком. Але цей підхід не є задовільним при побудові великих мереж для розв’язання складних завдань з реального світу. Тому ведуться розробки методів автоматичного підбору архітектури нейронних мереж, що підходила б якнайкраще до конкретної задачі.

У даній роботі розглянуто деякі існуючі методи навчання нейронних мереж із автоматичним підбором архітектури. Також було здійснено спробу створити програмне забезпечення, здатне знаходити оптимальну для конкретної задачі архітектуру нейронної мережі прямого розповсюдження з каскадними елементами. Цього було досягнення в результаті розробки компактної схеми кодування архітектури нейронної мережі та їх поєднання із еволюційними алгоритмами.

Порівняння алгоритмів навчання нейронних мереж

Нейронні мережі з неполіноміальними функціями активації можуть бути використані для апроксимації будь-яких функцій. В роботі [1] це було доведено для обмежених та неперервних функцій, а згодом ці результати були поширені [2] для довільних цільових функцій. Традиційно для розв’язання задач апроксимації використовують мережі прямого розповсюдження (*Feedforward Neural Network*).

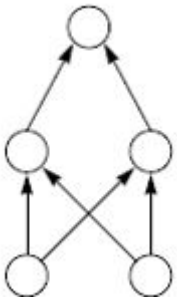


Рисунок 1: Класична архітектура для задачі XOR

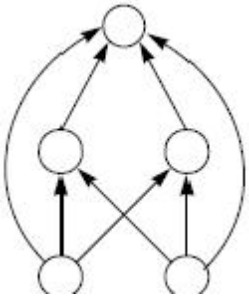


Рисунок 2: Архітектура для задачі XOR з каскадними елементами

В [3] доведено, що проста *FNN* мережа (див. рис. 1), навчається розв‘язувати задачу *XOR* набагато повільніше, ніж мережа з додатковими прямими зв‘язками між вхідними та вихідним нейронами (див. рис 2). Так у деяких випадках може бути дуже важливо мати у мережі такі прямі зв'язки між вхідними та вихідними шарами. У загальному випадку, можна розглядати зв'язки між будь-якими шарами і модулями мережі. Існує гіпотеза, що такі зв’язки усувають плато в поверхні помилки.

Отже, архітектура штучної нейронної мережі може сильно вплинути на її здатність до навчання. При використанні нейронних мереж для розв’язання конкретної задачі дуже важливо знайти її найкращу архітектуру. Для невеликих проблем або проблем з відомими розв’язками архітектуру мережі можна розробити і вручну. Однак, коли мова йде про складну проблему, що не має аналітичного розв‘язку, знайти оптимальну архітектуру вручну стає практично неможливим.

Традиційними методами навчання мережі з автоматичним підбором архітектури є конструктивні та деструктивні алгоритми. Яскравим прикладом таких методів є алгоритм каскадної кореляції [4]. Цей алгоритм має значну перевагу над алгоритмами навчання *FNN* у швидкості та ефективності навчання (не потребує зворотного розповсюдження), а також стабільності (мережа зберігає засвоєну інформацію за будь-яких змін вхідних даних). Також цей алгоритм дає змогу економити людські ресурси, бо мережа автоматично визначає власні розміри і топологію. Однак, цей алгоритм має певні недоліки. Топологія мережі є сталою за структурою та змінюється лише у розмірах в залежності від задачі. Але ця архітектура майже завжди буде неоптимальною. Використання як конструктивних, так і деструктивних методів навчання не достатньо для вирішення складних проблем.

Застосування еволюційних алгоритмів для підбору архітектури нейронних мереж

У подальших дослідженнях були спроби застосувати для автоматичного підбору архітектури еволюційні, генетичні, ітераційні та ін. алгоритми. Загальною рекомендацією є зменшення пошукового простору за допомогою якомога компактнішої схеми кодування архітектури. Тобто представлення графа мережі у вигляді матриці суміжності не є прийнятним. У даній роботі для цього було використано граматику графів, що заснована на L-системах.

Ця граматика базується на граматиці, запропонованій у [3]. Але вона містить додаткові символи для усунення неоднозначностей, властивих попередній граматиці. Цими символами є символи атрибутів вершин.

Відповідності у термінології

У наведеній нижче таблиці подано еквівалентні значення термінів, що було використано для опису підбору архітектури мережі та генетичних алгоритмів.

Генетичні алгоритми	Підбір архітектури мережі
Індивід	L-система
Хромосома	Правило L-системи, Аксіома L-системи
Міра пристосовуваності	Помилка на виході мережі (SSE)
Схрещування (Cross-over operator)	Не застосовано (N/A)
Оператор мутації (Mutation operator)	• Заміна вершини на шар нейронів • бінарне дерево • каскад

Таблиця 1: Відповідність між генетичними алгоритмами та L-системами

Граматика G0L

У граматиці для побудови графів визначаються такі символи :

1. Символи латинського алфавіту – ідентифікатори вершин. Вони не мають обов’язково бути унікальними. Наявність ідентифікатора просто свідчить про наявність вершини.
2. Цілі числа – з’єднання. Позначають відносну адресу вершини, яка буде з’єднана з поточною. Поточна вершина для даного з’єднання – це найближча зліва до даного з’єднання вершина.

3. Квадратні дужки []. Усе, що знаходиться між ними, являється підсистемою, ізольованим графом. Така підсистема розглядається як одна вершина поточної системи. До вхідних з'єднань підсистеми приєднуються зовнішні з'єднання, напрямлені до підсистеми. Вихідні з'єднання приєднуються до зовнішніх вершин, з якими дана підсистема з'єднана.

Мережа з каскадною архітектурою може бути описана такою L-системою :

G = <N, T, P, S>

T = {c, d, b, _, **:**, i, o, [,]}

N = { INPUTS , LAYERS, OUTPUTS, **A**}

S = {_INPUTS:i +1+2 _LAYERS +1 _OUTPUTS:o }

P = {_INPUTS ==> [_c:io_d:io]

_LAYERS ==> [[_A:io]:io+1_b:io]

_A:io ==> _b:io+1[_A:io]:io}

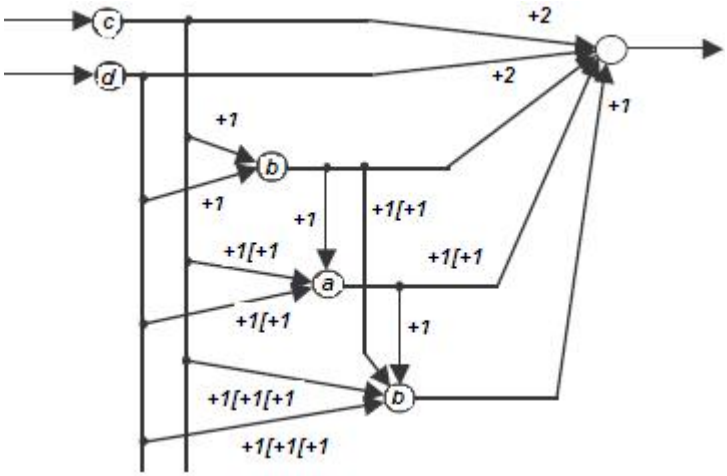


Рисунок 3: Каскадна мережа

Після триразового застосування правил до аксіоми отримуємо ланцюг виводу:

[_c:io_d:io]:i+1+2[[_b:io+1[_A:io]:io]:io+1_b:io]+1_OUTPUTS:o , що описує граф на рис. 3.

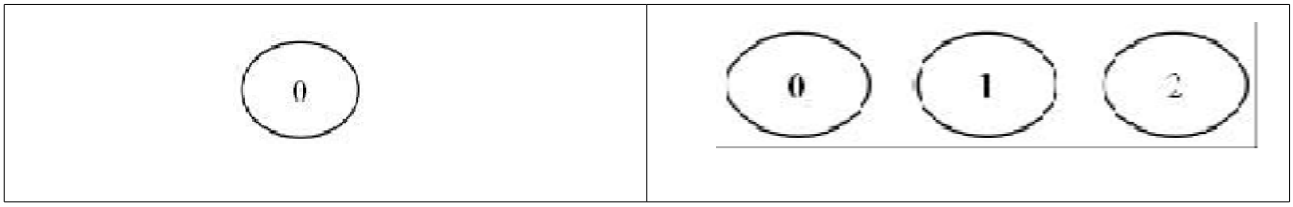
Оператори мутації

Для підбору архітектури було введено такі оператори мутації :

1. Заміна вершини на шар нейронів. До L-системи вводиться правило вигляду:

   Padd = {_b ==> **[_x1:io_x2:io_x3:io]**}

   Кількість нейронів, на які вершину буде замінено, обирається випадково. Для зменшення зони пошуку ця кількість є обмеженою. Найбільша кількість вершин, що може бути введена до системи за одне застосування мутації, становить **5**.



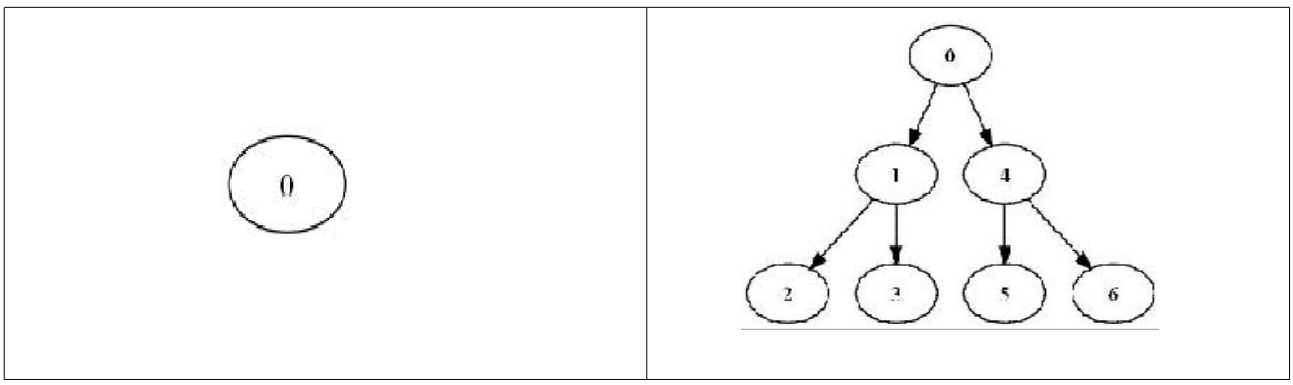
2. Заміна вершини на бінарне дерево. До L-системи вводиться правило виду :

   Padd =

   {_b ==> **[_x1:io+1+2+3…+N**

   _b1:io_b2:io_b3:io…_bN:io]}

   Кількість нейронів, на які вершину буде замінено, обирається випадково. Для зменшення зони пошуку ця кількість є обмеженою. Найбільша розмірність дерева **N**, що може бути введене до системи за одне застосування мутації, становить **3**.



3. Заміна вершини на каскад. До L-системи вводиться правило виду :

   Padd = {**_b** ==> **[[_x:io]:i+1_y:o]** **_x:io** ==> **_y:io+1[_x:io]:io**}

При використанні будь-якого з правил, вершина b для застосування мутації обирається випадково з-поміж вершин, які можуть з'явитися у результаті переписування L-системи.

Навчання мережі методом випадкового пошуку.

Для навчання таких мереж було використано алгоритм випадкового пошуку. Ефективність роботи алгоритму випадкового пошуку сильно залежить від того, як будуть робитися відхилення (тобто, як буде обиратися наступна точка для порівняння). Вибір великого кроку загрожує тим, що ми можемо пропустити розв'язок. Вибір малого кроку призводить до зростання часу, витраченого на знаходження розв'язку.

Зважаючи на це, для розв'язання задачі було обрано метод „From Genetic Algorithms To Efficient Optimization", описаний Deniz Yuret у своїй роботі [5].

У основі метода лежать такі принципи :

1. Для знаходження глобального максимуму функції проводиться кілька спроб знаходжен обирається найкраща. Нова початкова точка для алгоритму локального пошуку обирається таким чином, щоб вона була якомога більше віддалена від раніше знайдених локальних максимумів (рис. 4).

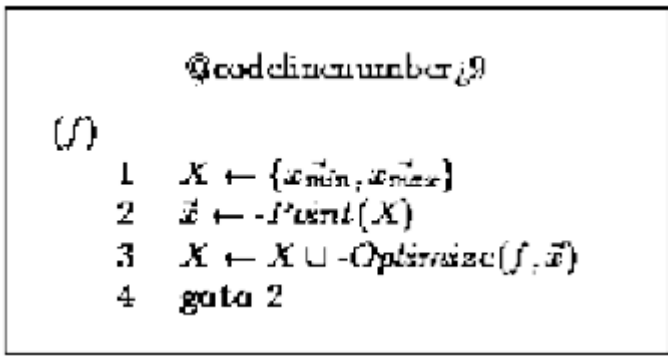


Рисунок 4: Псевдокод алгоритму глобальної оптимізації

   Для пошукових просторів з великою розмірністю прийнятним є випадковий вибір нової початкової точки.

Такий підхід дозволяє виходити з зони локального максимуму та дає шанси знайти більш оптимальний розв'язок.

2. Розмір пробного кроку змінюється у відповідності з локальним характером поверхні помилки. Крок слід змінювати пропорційно до частоти вдалих кроків. (зменшувати при зменшенні частоти успішних кроків та збільшувати при зростанні цієї частоти).

```
@codelinenumber@
-Optimize(f, x)
1   initialize v; u ← 0
2   while |v| ≥ THRESHOLD
3       iter ← 0
4       while f(x + v) < f(x) and iter < MAXITER
5           v ← -Vector(v)
6           iter ← iter + 1
7       if f(x + v) < f(x)
8           v ← v/2
9       else if iter = 0
10          x ← x + v;  u ← v + v;  v ← 2v
11      else if f(x + u + v) > f(x)
12          x ← x + u + v;  u ← u + v;  v ← 2u
13      else
14          x ← x + v;  u ← v;  v ← 2v
15  return x
```

Рисунок 5: Алгоритм локального пошуку

Така стратегія має переваги порівняно із методами, у яких розмір кроку є сталою величиною, оскільки він враховує характер поверхні помилки (рис.).

3. Відслідковувати напрямки недавніх успішних кроків (використовувати статистичний градієнт).

   Використання статистичного градієнту може пришвидшити досягнення оптимального розв’язку завдяки відкидання невдалих напрямків та перегляді вдалих у першу чергу.

Поєднання цих ідей у одному алгоритмі дає достатньо ефективний алгоритм оптимізації, який не висуває жодних вимог до цільової функції. Ефективність цього алгоритму було підтверджено у роботі Deniz Yuret на тесті де-Джонга (De Jong's test-problem suite [10]) та деяких інших задачах.

Його перевагами є те, що розмір пробного кроку змінюється у відповідності з локальним характером поверхні помилки та використання статистичного градієнту. Цей метод дав прийнятні результати на простих задачах (задача XOR) але виявився недостатньо ефективним для розв‘язання задачі “Two Spiral Problem”.

Результати досліджень

Задача «Two Spiral problem»

Задача «Two Spiral problem» полягає у розділенні всіх точок площини на дві спіралі, виходячи з таких даних :

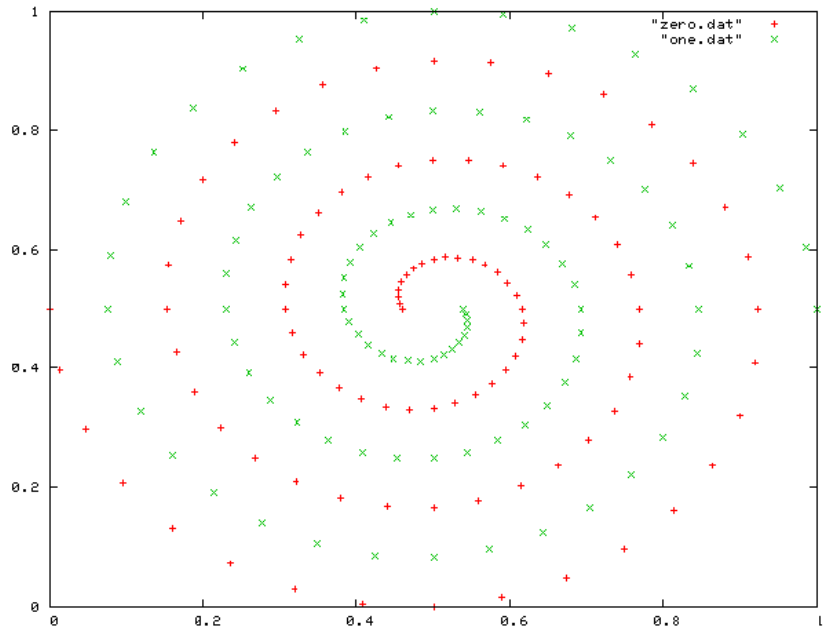


Рисунок 6: Вхідні дані для задачі "Two spiral problem"

Червоний колір відповідає значенню “0”. Зелений колір відповідає значенню “1”. У результаті ми повинні отримати приблизно такий розподіл точок :

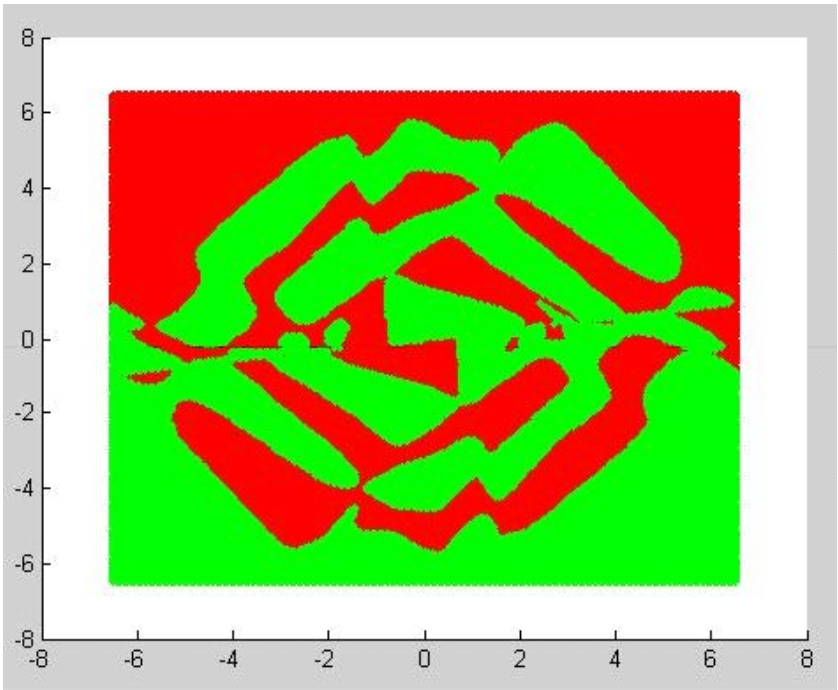


Рисунок 7: Two Spiral problem - розвязок задачі

Метод випадкового пошуку :

Як видно з рис. 8, результати навчання мережі за допомогою метода випадкового пошуку не є задовільними.

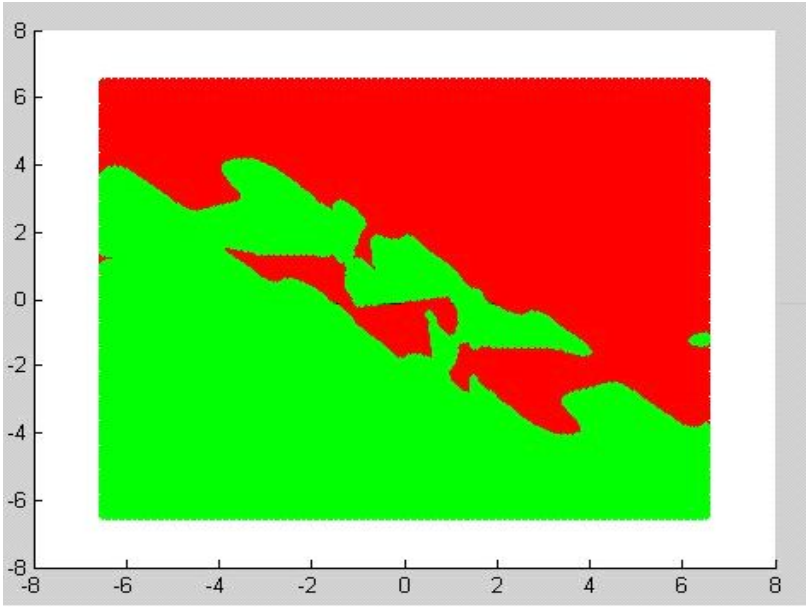


Рисунок 8: Two Spiral problem випадковий пошук

Навчання за допомогою Matlab Neural network toolbox

З моменту отримання попередніх результатів було дещо вдосконалено процедуру візуалізації результатів.

Надалі в ілюстраціях результатів будуть використані такі кольори :

Колір	Значення
Червоний	Значення **0**, отримане на **тестових** даних
Зелений	Значення **1**, отримане на **тестових** даних
Синій	Значення **0**, отримане на **навчальних** даних
Жовтий	Значення **1**, отримане на **навчальних** даних

Наступним кроком стало застосування **Matlab Neural network toolbox** для навчання мереж із нестандартною (нешаруватою архітектурою). **Matlab Neural network toolbox** має можливості роботи з мережами, що мають каскадну архітектуру. Зокрема, функцію створення каскадної мережі *newcf*. А також алгоритм LMA (*Levenberg-Marquardt backpropagation algorithm*) для навчання таких мереж. Проблемою функції newcf є те, що вона створює повністю зв'язану архітектуру. Тобто, до кожного шару мережі приєднані всі попередні шари. Отже, використання цієї функції не є прийнятним у рамках розглянутої задачі. Отже, постає потреба самим конструювати необхідну архітектуру (*custom network*) у термінах обєкту **Matlab Neural Network** (для більш детальної інформації див. [6]).

Особливості Matlab Neural network toolbox

1. Неможливо задати властивості (активаційну функцію, "замороження" ваг, тощо) для одного конкретного нейрона у шарі. Це можна зробити лише для шару в цілому.

   Тому було вирішено обрати представлення мережі "**1 нейрон — 1 шар**".

2. Вибрана нами функція ініціалізації шарів "***initnw***" не працює, якщо мережа має хоча б один шар нейронів із пороговою (***hardlim, hardlims***) функцією актиації. Це може серйозно заплутати людей, що проектують мережу. Навіть при правильно спроектованій архітектурі мережа може погано навчатися при поганому виборі початкових ваг зєднань.

3. Якщо два шари нейронів зєднані, то навчаються усі можливі звязки між їхніми нейронами.

   Мережі, розглянуті у даній роботі, не потребують такої повної звязаності. Навпаки, при їхньому проектуванні ми намагаємося її уникнути.

4. Нейрони вхідного слою (***net.inputs***) не описуються у структурах (**net.layers**), (**net.layerConnect**). Ці нейрони винесені у окремий шар, що має лінійні активаційні функції.

   Таким чином, при проектуванні мережі слід або

   - Розбити матрицю суміжності графа архітектури на дві частини (inputs, layers)
   - Використовувати матрицю суміжності графа архітектури у якості структури ***net.layerConnect***. Встановити у матриці ***net.IW*** значення "**1**". Та "заморозити" ці значення ***net.inputw{i}.learn = 0;***

   Ми обрали другий варіант як більш простий та масштабований.

Результати навчання мереж, сконструйованих вручну

Цю реалізацію було випробувано на задачі “Two spiral problem”. Випробування проводилися як на архітектурах, заданих вручну так і на архітектурах, отриманих у результаті мутації. Нижче наведені іхні результати :

Каскадна мережа Фальмана

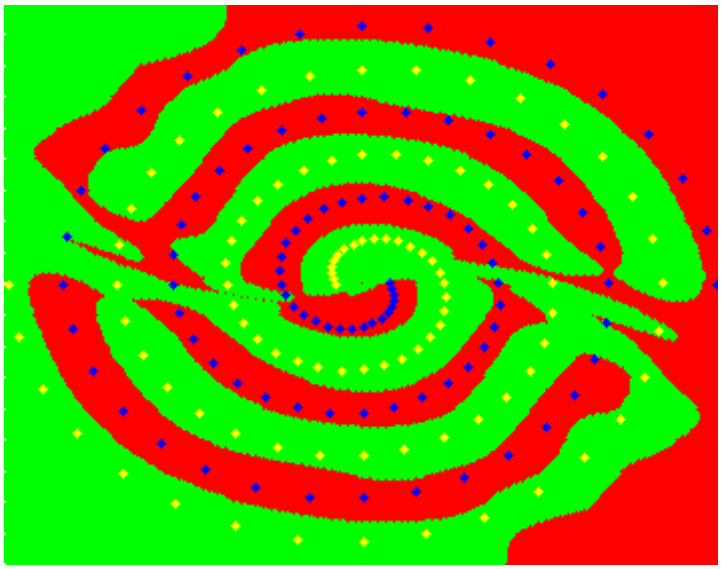


Рисунок 9: Каскад з 16 шарів

Каскадна мережа із шарами різного розміру

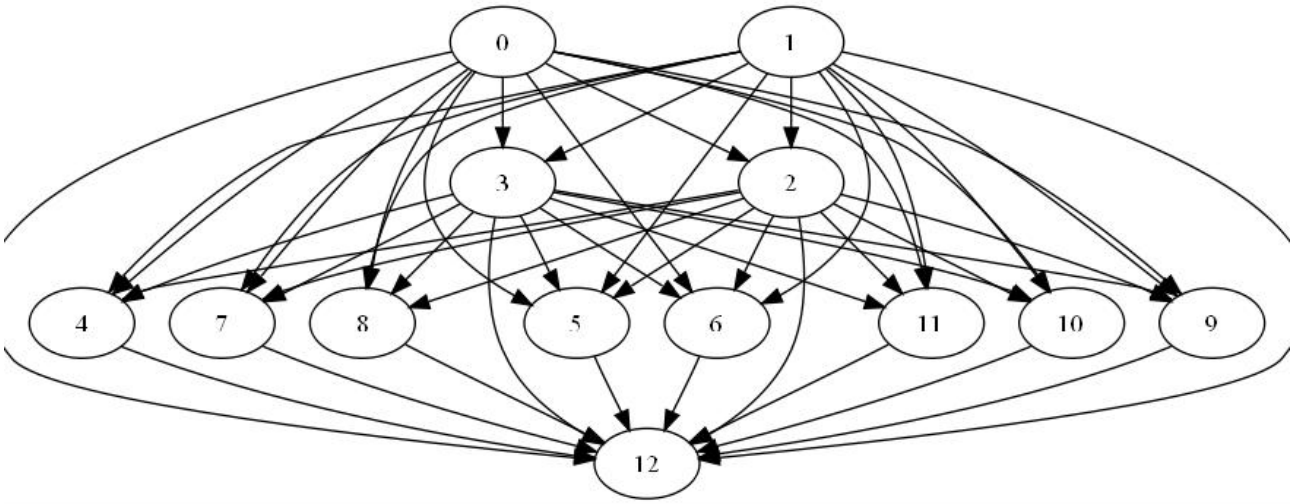


Рисунок 10: Архітектура мережі

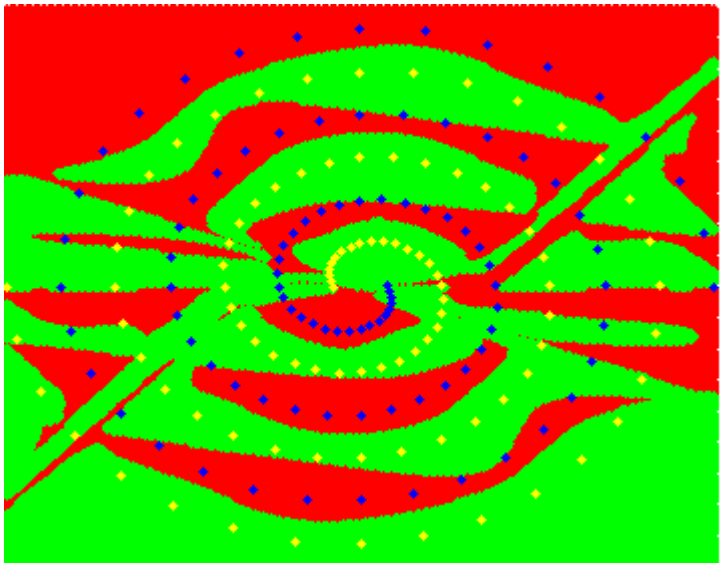


Рисунок 11: Результат навчання

Результати навчання із автоматичним підбором архітектури.

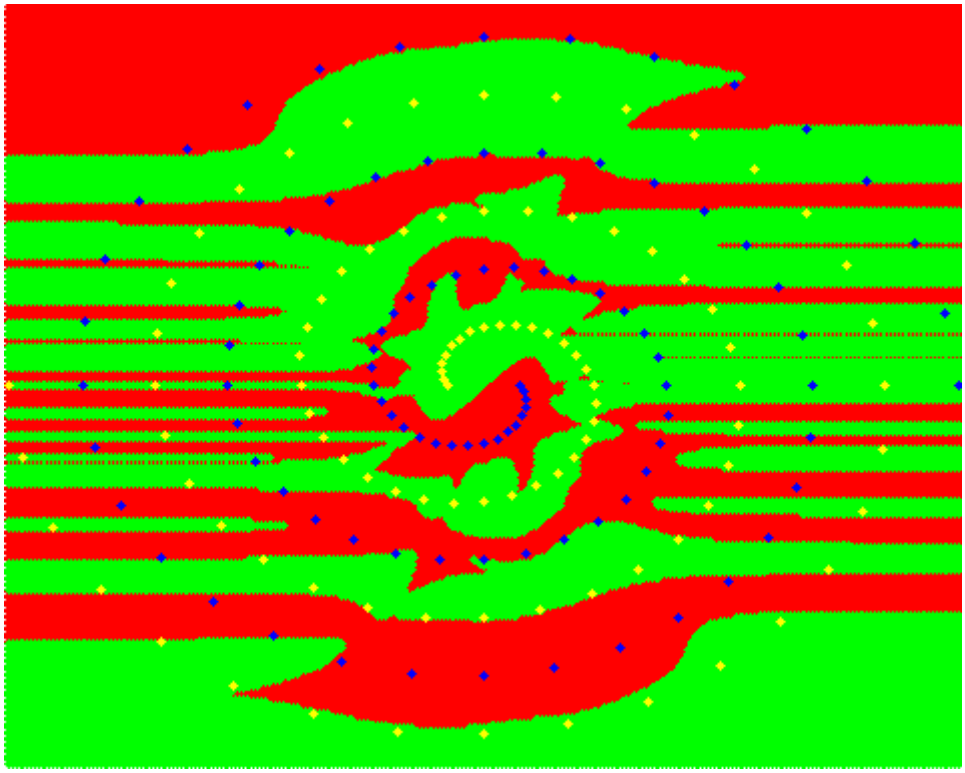


Рисунок 12: Спроба1

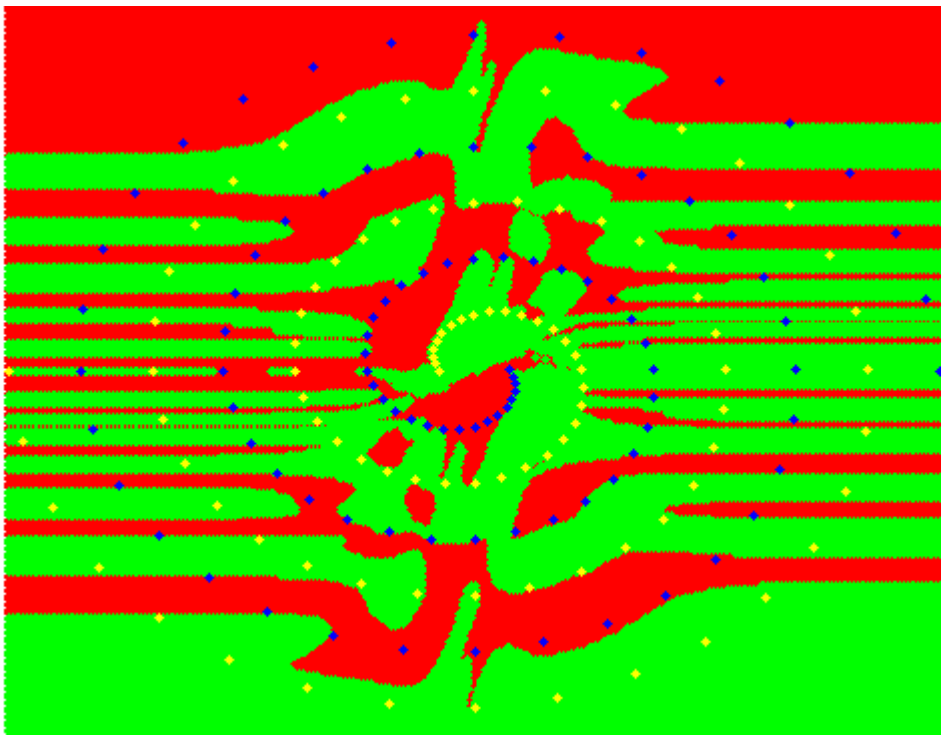


Рисунок 13: Спроба 2

Мережа	Кількість нейронів	Помилка навчання (MSE)	Кількість спроб мутації
Каскадна (рис. 10)	16	0.343073	N/A
Ручна архітектура1 (рис. 11)	13	0.438923	N/A
Автоматична архітектура1 (рис. 12)	89	0.458596	21
Автоматична архітектура 2 (рис. 13)	85	0.492109	51

У якості характеристики придатності архітектури використовується значення помилки на навчальній вибірці. При цьому застосують однакову кількість переписувань для обох архітектур. Результати роботи алгоритму випадкового пошуку архітектури свідчать, що ця схема порівняння є недостатньо ефективною. Наразі ведуться роботи над покращенням схеми порівняння із урахуванням кількості переписувань.

Висновки

Таким чином, в роботі запропоновано засоби для побудови та подальшого навчання нейронних мереж каскадної архітектури. Для них розроблено компактну схему кодування, придатну до подальшого застосування у еволюційних алгоритмах. Було створено бібліотеку для перетворення нейромережі з G0L граматики або матриці суміжності графу архітектури до формату Matlab neural network toolbox.

На архітектурах, заданих вручну, були отримані прийнянті результати. Для архітектур, підібраних автоматично, результати дещо гірші, але вони можуть бути покращені більш вдалим вибором початкової архітектури мережі або використанням більш складних та ефективних еволюційних алгоритмів.

Список літератури

1: Hornik K., Stinchcombe M., White H. Multilayer feedforward networks are universal approximators, NeuralNetworks,, 1989.- 2 358-366
2: Leshno M., Schocken S. Multilayer feedforward networks with non-polynomial activation functions can approximate any function, Neural Networks,, 1993.- 6 861-867
3: Boers E. J. W., Sprinkhuizen-Kuyper I. Combined Biological Metaphors, Advances in the Neural Information Processing Systems,MIT Press, 2001.- 153-183
4: Fahlman S. E., Lebiere C.. The Cascaded-correlation learning architecture, Advances in Neural Information Processing Systems,, 1990.- 2 524-532
5: Yuret D., From Genetic Algorithms To Efficient Optimization, Massachusetts institute of technology.- artificial intelligence laboratory.- Technical Report No. 1569,1994
6: Matlab Neural Network toolbox manual:
http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/