УДК 681.5.004.8
DOI 10.36910/10.36910/6775-2313-5352-2023-22-20
Смолянкін О.О., Смолянкін О.О., Сацик О.В., Сацик В.О., Решетило О.М.
Луцький національний технічний університет, м. Луцьк, Україна

АНАЛІЗ АЛГОРИТМІВ МАШИНОЇ ЛОГІКИ ТА ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ПОБУДОВИ СИСТЕМ ПРИЙНЯТТЯ РІШЕНЬ

Аналіз алгоритмів машинної логіки та штучного інтелекту для побудови систем прийняття рішень є важливим етапом досліджень у цій області. У процесі аналізу вивчаються різні алгоритми та методи, які використовуються для розробки систем прийняття рішень, зокрема базовані на машинній логіці та штучному інтелекті. Одним із прикладів штучного інтелекту для розв'язання логічних завдань є використання нейронних мереж. Нейронні мережі є потужним інструментом для моделювання та вирішення різноманітних задач. Застосування нейронних мереж для вирішення логічних прикладів може включати створення моделей, які навчаються на основі вхідних даних і здатні зробити висновки на основі цих даних. Наприклад, можна побудувати систему штучного інтелекту, яка використовує нейронну мережу для вирішення логічних прикладів на мові JavaScript. У такій системі можуть бути визначені правила та умови, за якими нейронна мережа буде приймати рішення. Навчання нейронної мережі здійснюється на основі тренувальних даних, які включають вхідні параметри та очікувані результати. Після тренування мережа зможе вирішувати логічні приклади на основі свого навчання. Одним із викликів при розробці систем прийняття рішень є забезпечення точності та надійності результатів. Потрібно враховувати можливі помилки алгоритмів, недостатню або надлишкову гнучкість системи, а також вплив невизначеності і неповноти вхідних даних. Важливим аспектом є проведення експериментів та оцінка ефективності різних алгоритмів для визначення їхньої придатності до конкретної задачі прийняття рішень.

Ключові слова: штучний інтелект, нейронні мережі, алгоритми, мова JS.

Постановка проблеми. Штучний інтелект (ШІ) це швидкозростаюча галузь в інформатиці, яка фокусується на побудові систем інтелекту, які можуть вчитися і робити висновки як люди. ШІ має довгу історію, починаючи з 1950-их років, і суттєво еволюціонувався протягом останніх років. Наразі, ШІ має широке коло застосувань, від розпізнавання голосу чи зображень, до автономних автомобілів і чат ботів.

Зачатки ШІ можуть бути прослідковані до конференції в Дартмуті 1956 року, коли група дослідників зібралась обговорити можливість побудови машин, які можуть думати як люди. Конференція позначила початок досліджень в галузі ШІ і вела до розробки ранніх систем ШІ, таких як експертні системи, або системи на основі правил.

Пізніше з'явився символьний ШІ, тобто методи побудови штучного інтелекту, який міг оперувати високорівневим символьним описом проблеми(таким який могла прочитати людина), логіки чи пошуку. Символьний ШІ використовував такі інструменти, як логічне програмування, семантичні сітки і фрейми(структури даних, які використовуються для поділу знань на підструктури відображаючи стереотипні ситуації). Застосування символьного ШІ було широким, його використовували в символьній математиці, систем базованих на знаннях, семантичної павутини, онтології, і систем для автоматичного доведення теорем. Символьний штучний інтелект був домінуючою парадигмою в дослідженні ШІ протягом довгих років аж до 1990-их [1]. Хоча штучний інтелект (ШІ) розроблявся протягом багатьох років і досяг значних досягнень, його розвиток стикається з новими викликами і проблемами. Деякі з цих проблем включають розуміння контексту, етичні питання, відсутність здатності до абстрагування, відсутність свідомості, безпека і кіберзлочинність.

Аналіз останніх досліджень і публікацій. Розвиток і дослідження штучного інтелекту йшли по синусоїді, спочатку попит і інтерес до ШІ зростали, компанії виділяли фінансування на дослідження. Потім же через брак швидких результатів, чи із-за обмежених обчислювальних потужностей тодішніх комп'ютерів і браку даних, наставав занепад. Так відомі 2 “зими штучного інтелекту” 1974-1980, і 1987-1993 [2].

Мета роботи. Дослідження можливостей застосування простого штучного інтелекту (ШІ) з використанням нейронної мережі для вирішення логічних прикладів. Стаття спрямована на вивчення та опис потенційних переваг використання простого штучного інтелекту в

поєднанні з нейронною мережею для вирішення логічних прикладів. У статті буде розглянуто різні підходи до розв'язання логічних завдань за допомогою нейронних мереж, з фокусом на простому штучному інтелекті.

Викладення основного матеріалу. Сучасний ШІ має широке коло застосувань, від персональних помічників і домашніх розумних пристроїв, до охорони здоров'я і фінансів. Системи з розпізнавання зображення використовуються в розпізнаванню облич, системах безпеки і автономних транспортних засобах. Системи з розпізнавання голосу використовуються як віртуальні помічники, кол центри, або перекладачі на іншу мову [3].

Інших популярний застосунок ШІ це обробка натуральної мови, або по англійськи natural language processing (NLP), яке спонукає машини розуміти людську мову і генерувати відповідь людською мовою. NLP використовуються в чат ботах, віртуальних помічниках і сервісах перекладу. NLP системи можуть бути натреновані щоб розуміти нюанси людської мови, включаючи сарказм, іронію, гумор, і давати відповідну відповідь [4].

В охороні здоров'я штучний інтелект може допомогти розпізнавати захворювання, аналізувати медичні зображення, передбачати стан пацієнта, розробляти персоналізований план одужання. В фінансах ШІ може допомогти в виявленні шахрайства, управлінні портфоліо, торгівлі на біржових ринках. У військовій справі, ШІ може бути натренований допомагати вести військові дії, наприклад, знаходячи ворога, постійно моніторячи зображення з неба (ворожі літаки, ракети чи дрони) чи землі (ворожа піхота і техніка), робити системи наведення і автономних захисних систем, повз які не пройде ніякий ворог. На глобальному рівні, ШІ може допомогти розрахувати можливі дії противника і прогнозувати втрати при тих чи інших діях зі строни військового командування.

Також ШІ може зменшити проблему "вогонь по своїм" завдяки кращому розпізнаванню "свій-чужий".

Для побудови ШІ існують наступні алгоритми створення машинної логіки:

1. Лінійна регресія. Це простий алгоритм який використовується для передбачення числових значень базуючись на множині вхідних параметрів. Він працює на підгонці даних під лінійне рівняння. Самими простими прикладами такого ШІ може бути інтерполяція і екстраполяція. Наприклад, прогнозування цін на акції в майбутньому по математичній формулі базуючись на відомих цінах минулого.

2. Логічна регресія. Тип регресії який використовується для передбачення бінарних висновків(наприклад, "так" чи "ні", "вірно" чи "невірно"). Такий ШІ працює на підгонці даних під логічні функції.

3. Дерева рішень. Алгоритм який використовує деревовидну модель для відображення рішень з можливих наслідків. Часто цей алгоритм використовується в класифікації задач і може обробляти як числові дані, так і дані категорій.

4. Нейронні мережі: група алгоритмів натхненних структурою і функціональністю людського мозку. Можуть використовуватися для багатьох задач, таких як розпізнавання зображення, голосу, так і NLP.

5. Глибинне навчання. Підмножина нейронних мереж які характеризуються тим що мають багато прошарків. Вони часто використовуються в таких складних задачах як розпізнавання голосу чи зображення.

6. Генетичні алгоритми. Тип оптимізаційного алгоритму, натхненний процесом природної селекції. Вони можуть використовуватися для оптимізації широкого кола проблем, включаючи тонке налаштування параметрів для алгоритмів машинного навчання.

Найчастіше використовуються нейронні мережі і глибинне навчання для побудови ШІ. В наступному прикладі, показано як можна побудувати простий ШІ для вирішення логічних прикладів, використовуючи нейронну мережу. Для цього побудуємо простий перцептрон, який буде складатися з 2 входів (2 параметри для нашої функції), 3 прихованих нейронів, і 1 вихідного (який буде видавати результат функції, рис. 1).

Для побудови і тренування мережі нам потрібно виконати певні кроки.

1. Ініціалізувати початкові значення ваги для зав'язків входів і прихованих нейронів, ваги для зав'язків прихованих нейронів і виходу, а також зміщення прихованого шару і зміщення для виходу.

2. Визначити активаційну функцію. В нашому випадку будемо використовувати функцію сигмоїд.

3. Робимо один крок прямого розповсюдження. Для цього перемножаємо вхідні дані на ваги зв'язків входу і прихованого шару, додаємо до результатів масив зміщення, і отримуємо

значення прихованих нейронів. Тоді ці значення перемножаємо на ваги зв'язків прихованого шару і виходів і добавляємо зміщення виходу.

4. Вичисляємо похибку між отриманим значенням виходу нейронної мережі, і значенням, яке ми очікували отримати. Також на цьому кроці ми оновлюємо ваги зв'язків між нейронами і зміщення.

5. Повторюємо кроки 3 і 4 певну кількість разів. В нашому результаті, виконується 1 мільйон кроків, перш ніж ми вважаємо нейронну мережу натренованою.

6. На цьому кроці ми можемо використовувати мережу, щоб передбачати результати параметрів, з яким нейронна мережа ще не працювала.

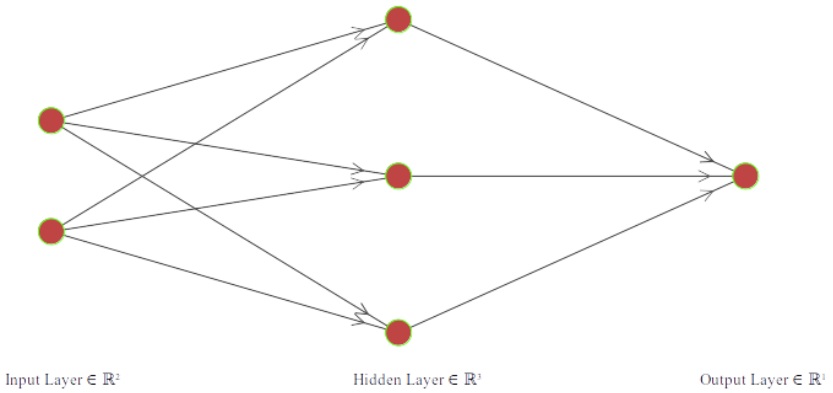

Рисунок 1– Схема перцептрона для нейронної мережі.

```
// Define the activation function (sigmoid)
function sigmoid(x) {
 return 1 / (1 + Math.exp(-x));
}
function sigmoidDerivative(x) {
 x = sigmoid(x);
 return x * (1 - x);
}
function NeuralNetwork(inputSize, hiddenSize, outputSize) {
 this.inputSize = inputSize;
 this.hiddenSize = hiddenSize;
 this.outputSize = outputSize;
 // Initialization hiddenLayer
 this.hiddenLayer = Array(this.hiddenSize).fill(0);
 // Init hiddenWeights
 this.hiddenWeight = Array(this.inputSize)
   .fill()
   .map(() => Array(this.hiddenSize)
     .fill()
     .map(() => Math.random() * 2 - 1));
 // Init hiddenBiases
 this.hiddenBiases = Array(this.hiddenSize)
   .fill()
   .map(() => Math.random() * 2 - 1);
 // Init outputWeight
 this.outputWeight = Array(this.hiddenSize)
   .fill()
   .map(() => Math.random() * 2 - 1);
 // Init outputBiase
 this.outputBiase = Math.random();
 this.forward = function(input) {
   for (let i = 0; i < this.hiddenSize; i++) {
     for (let inputIndex = 0; inputIndex < inputSize; inputIndex++) {
       this.hiddenLayer[i] += input[inputIndex] * this.hiddenWeight[inputIndex][i];
```

```
  }
    this.hiddenLayer[i] += this.hiddenBiases[i];
    this.hiddenLayer[i] = sigmoid(this.hiddenLayer[i]);
  }
  let output = 0;
  for (let i = 0; i < this.hiddenSize; i++) {
    output += this.hiddenLayer[i] * this.outputWeight[i];
  }
  output += this.outputBiase;
  output = sigmoid(output);
  return output;
 }
this.train = function(learningRate) {
  // Feedforward
  const inputs = [
    Math.round(Math.random()),
    Math.round(Math.random())
  ];
  const expectedResult = inputs[0] ^ inputs[1];
  const actualResult = this.forward(inputs);
  // Backpropagation
  const error = expectedResult - actualResult;
  const resultDelta = error * sigmoidDerivative(actualResult);
  let hiddenErrors = [];
  let hiddenDeltas = [];
  for (let i = 0; i < this.hiddenSize; i++) {
    hiddenErrors[i] = resultDelta * this.outputWeight[i];
    hiddenDeltas[i] = hiddenErrors[i] * sigmoidDerivative(this.hiddenLayer[i]);
  }
  // Update weights and biases
  for (let i = 0; i < this.hiddenSize; i++) {
    for (let inputIndex = 0; inputIndex < this.inputSize; inputIndex++) {
      this.hiddenWeight[inputIndex][i] += learningRate * hiddenDeltas[i] * inputs[inputIndex];
  }
    this.hiddenBiases[i] += learningRate * hiddenDeltas[i];
  }
  for (let i = 0; i < this.hiddenSize; i++) {
    this.outputWeight[i] += learningRate * resultDelta * this.hiddenLayer[i];
  }
  this.outputBiase += learningRate * resultDelta;
 }
this.output = function (parameters) {
  const result = this.forward(parameters);
  console.log(`For parameters ${parameters[0]} and ${parameters[1]} result is = ${result}`);
 }
 // Define the neural network architecture
 const inputSize = 2;
 const hiddenSize = 3;
 const nn = new NeuralNetwork(inputSize, hiddenSize);
 const epochs = 1000_000;
 const learningRate = 0.1;
 for (let i = 0; i < epochs; i++) {
  nn.train(learningRate);
 }
 // Test the neural network
 nn.output([0, 0]); // should output close to 0
 nn.output([0, 1]); // should output close to 1
 nn.output([1, 0]); // should output close to 1
```

```
nn.output([1, 1]); // should output close to 0
```

В результаті, ми отримали значення дуже близькі до тих, які очікуємо.

```
// For parameters 0 and 0 result is = 0.0005674384536595668
// For parameters 0 and 1 result is = 0.9955431106150707
// For parameters 1 and 0 result is = 0.9986396330129992
// For parameters 1 and 1 result is = 0.005870125295067755
```

Висновки. Нейронна мережа, як підвид штучного інтелекту, представляє собою не якийсь конкретний алгоритм, а всього лише набір даних - перцептронів, в яких є значення нейронів та вхідні і вихідні ваги. Труднощами з розробки нейронних мереж є підбір початкових значень, тривалість навчання, підбір активаційної функції. Простий штучний інтелект разом з нейронною мережею може бути потужним інструментом для вирішення логічних завдань. Він дозволяє автоматизувати процес розв'язання логічних прикладів, що спрощує роботу користувача та забезпечує швидкі й точні результати. Нейронні мережі можуть ефективно моделювати логічне мислення та вирішувати складні логічні завдання. Вони здатні до виявлення складних залежностей між вхідними та вихідними даними, що дозволяє їм ефективно розв'язувати логічні приклади різного рівня складності. Простий штучний інтелект з використанням нейронних мереж має свої переваги порівняно з традиційними методами вирішення логічних завдань, такими як експертні системи або правила. Він може автоматично вивчати залежності з великої кількості даних і пристосовуватися до нових ситуацій. Однак, використання простого ШІ з використанням нейронної мережі також має свої обмеження. Нейронні мережі можуть бути складними у тренуванні, вимагати значних обчислювальних ресурсів та велику кількість даних для досягнення задовільних результатів.

Інформаційні джерела

1. Francis Bach, Richard S. Sutton, Andrew G. Barto. Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning series) second edition Edition. Видавництво: A Bradford Book, 2018. 505 с. ISBN 966518153X.

2. Методи штучного інтелекту: навч. посіб. В.Б.Гітіс, К.Ю. Гудкова. Краматорськ: ДДМА, 2018. 136 с. ISBN 966-379-823-3.

3. Методи та системи штучного інтелекту: теорія та практика. Булгакова О.С., Зосімов В.В., Поздєєв В.О. Видавництво: Гельветика, 2020. 356 с.

4. Системи штучного інтелекту. Н.Б. Шаховська, Р.М. Камінський, О.Б. Вовк. Львів: Львівська політехніка, 2018. 392 с. ISBN 978-617-57-40-11-4.

Smoliankin O., Smoliankin O., Satsyk O., Satsyk V., Reshetylo O.
Lutsk National Technical University, Lutsk, Ukraine

THE ANALYSIS OF MACHINE LOGIC AND ARTIFICIAL INTELLIGENCE ALGORITHMS FOR DECISION-MAKING SYSTEMS IS AN IMPORTANT STAGE OF RESEARCH IN THIS FIELD

Various algorithms and methods used in the development of decision-making systems, particularly those based on machine logic and artificial intelligence, are studied during the analysis. One example of artificial intelligence for solving logical tasks is the use of neural networks. Neural networks are powerful tools for modeling and solving diverse problems. Applying neural networks to solve logical examples may involve creating models that learn from input data and can draw conclusions based on that data. For instance, it is possible to build an artificial intelligence system that utilizes a neural network to solve logical examples using JavaScript. In such a system, rules and conditions can be defined for the neural network to make decisions. The network is trained using training data that includes input parameters and expected results. After training, the network can solve logical examples based on its learning.

One of the challenges in developing decision-making systems is ensuring the accuracy and reliability of the results. Potential algorithmic errors, insufficient or excessive flexibility of the system, as well as the influence of input data uncertainty and incompleteness, need to be considered. Conducting experiments and evaluating the effectiveness of different algorithms are important aspects to determine their suitability for a specific decision-making task.

Keywords: *artificial intelligence, neural network, algorithms, JS language.*