Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота *№4*

із дисципліни ***«Методи і технології штучного інтелекту»***

Тема: ***«Моделювання функції двох змінних з двома входами і одним виходом на основі нейронних мереж»***

Виконав:

Студент групи ІА-34

Ястремський Богдан

Перевірив:

старший викладач кафедри ІСТ

Польшакова Ольга Михайлівна

Київ — 2025

Тема: Моделювання функції двох змінних з двома входами і одним виходом на основі нейронних мереж.

Мета: Дослідити структуру та принцип роботи нейронної мережі. За допомогою нейронної мережі змоделювати функцію двох змінних.

Хід роботи:

1. Імпортуємо модулі для роботи з обчисленнями, нейронними мережами, та нормалізацією даних. Обчислюємо y, z згідно варіанту.

```
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential, Model
from keras.layers import Input, Dense, Concatenate, SimpleRNN, Reshape
from sklearn.preprocessing import MinMaxScaler


x_values = np.linspace(-5, 5, 1000)
y_values = 7 * np.sin(2.5 * np.cos(x_values))
z_values = (x_values - 2) ** 2 * (1 - y_values**2)

# Data normalizing for lowering the value of MSE/MAE due to function value scaling
scaler_x = MinMaxScaler()
scaler_y = MinMaxScaler()
scaler_z = MinMaxScaler()

x = scaler_x.fit_transform(x_values.reshape(-1, 1)).flatten()
y = scaler_y.fit_transform(y_values.reshape(-1, 1)).flatten()
z = scaler_z.fit_transform(z_values.reshape(-1, 1)).flatten()
```

2. Оскільки ми будемо тестувати мережі, використовуючи 2 однакові параметри – к-сть шарів та нейронів, то для легкості, створюємо функцію з моделлю як параметром для навчання, тестування та візуалізації результату. Навчаємо модель на 20 епохах з розміром сегменту 100.

```
def model_testing(model):
    model.compile(optimizer='adam', loss='mse')
    model.fit(x, z, epochs=20, batch_size=100)
    z_pred = model.predict(x)

    plt.plot(x, z, label='Actual')
    plt.plot(x, z_pred, label='Predicted')
    plt.legend()
    plt.show()
```

3. Створюємо мережу feed forward backprop, яка містить вхідний шар – перший Dense, інші шари є прихованими. Далі створюємо вихідний шар.

```
def feed_forward_creation(layers, neurons):
    model = Sequential()
    model.add(Dense(neurons, activation='relu', input_shape=(1,)))

    for i in range(layers-1):
        model.add(Dense(neurons, activation='relu'))
    model.add(Dense(1, name='output'))
    return model
```

4. Створюємо cascade forward backprop. Вона містить вхідний шар з одним нейроном, каскадні зв’язки на кожному етапі через метод Concatenate(), приховані шари а також вихідний шар.

```
def cascade_forward_creation(layers, neurons):
    inputLayer = Input(shape=(1,), name='input')
    current = Dense(neurons, activation='relu', input_shape=(1,))(inputLayer)

    for i in range(layers-1):
        concatenatedLayer = Concatenate()([inputLayer, current])
        current = Dense(neurons, activation='relu', input_shape=(1,))(concatenatedLayer)

    outputLayer = Dense(1, name='output')(current)
    model = Model(inputs=inputLayer, outputs=outputLayer)

    return model
```

5. Створюємо Elman backprop. Вона схожа на feedforward, але з додаванням контекстного шару. Контекстний шар отримує дані від прихованого шару і, у свою чергу, передає свій вихід назад у прихований шар.

```
def elman_creation(layers, neurons):
    model = Sequential()
    model.add(Reshape((1, 1), input_shape=(1,), name='input_reshape'))
    model.add(SimpleRNN(neurons, return_sequences=True, activation='relu',
                        input_shape=(1,)))

    for i in range(layers - 1):
        model.add(SimpleRNN(neurons, return_sequences=True, activation='relu'))

    model.add(Dense(1, name='output'))
    model.add(Reshape((1,), input_shape=(1, 1), name='output_reshape'))

    return model
```

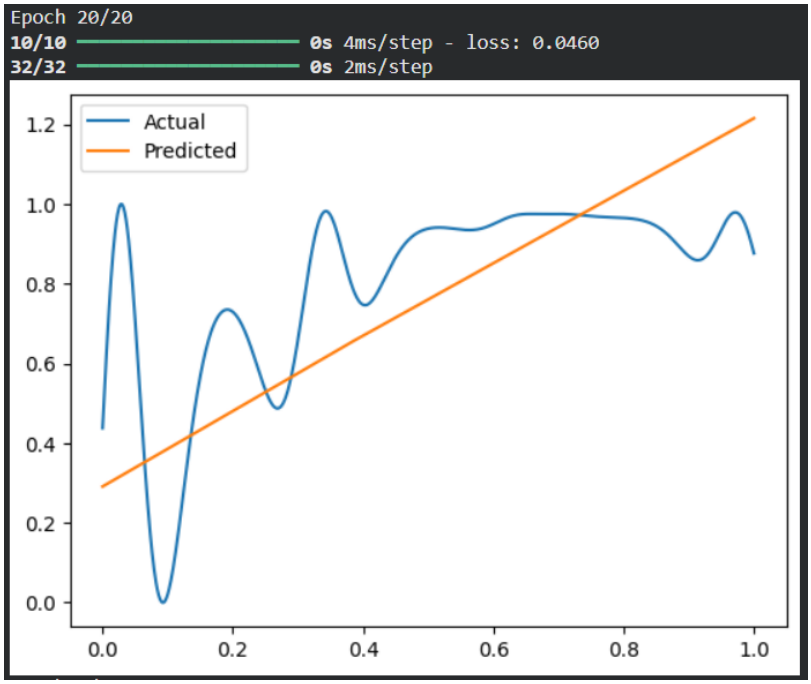
6. Створюємо власні мережі відповідно до завдання.

```
f1 = feed_forward_creation(1, 10)
f2 = feed_forward_creation(1, 20)
c1 = cascade_forward_creation(1, 20)
c2 = cascade_forward_creation(2, 10)
e1 = elman_creation(1, 15)
e2 = elman_creation(3, 5)

model_testing(f1)
model_testing(f2)
model_testing(c1)
model_testing(c2)
model_testing(e1)
model_testing(e2)
```

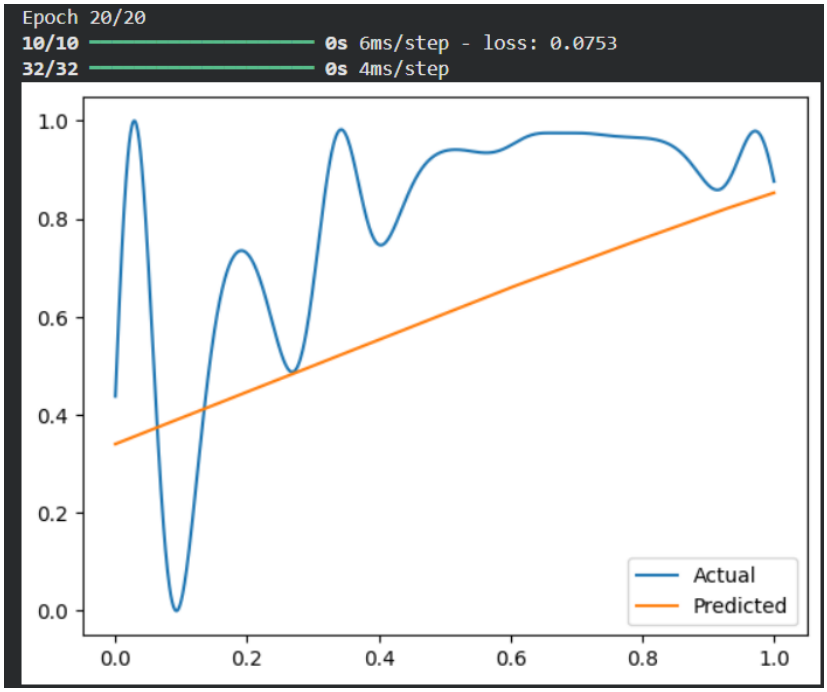
7. Результати навчання:

1) Feed forward NN – 1 шар, 10 нейронів:



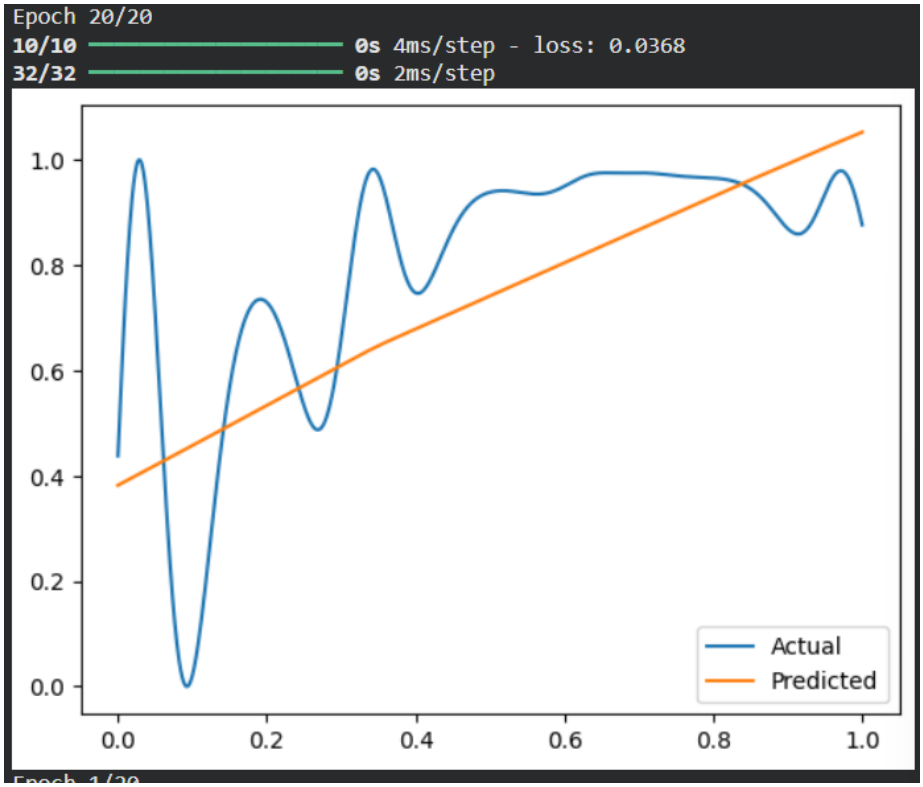
Для такої складної функції з багатьма піками, результат непоганий, враховуючи пропорцію «легкість моделі – помилка».

2) Feed forward NN – 1 шар, 20 нейронів:



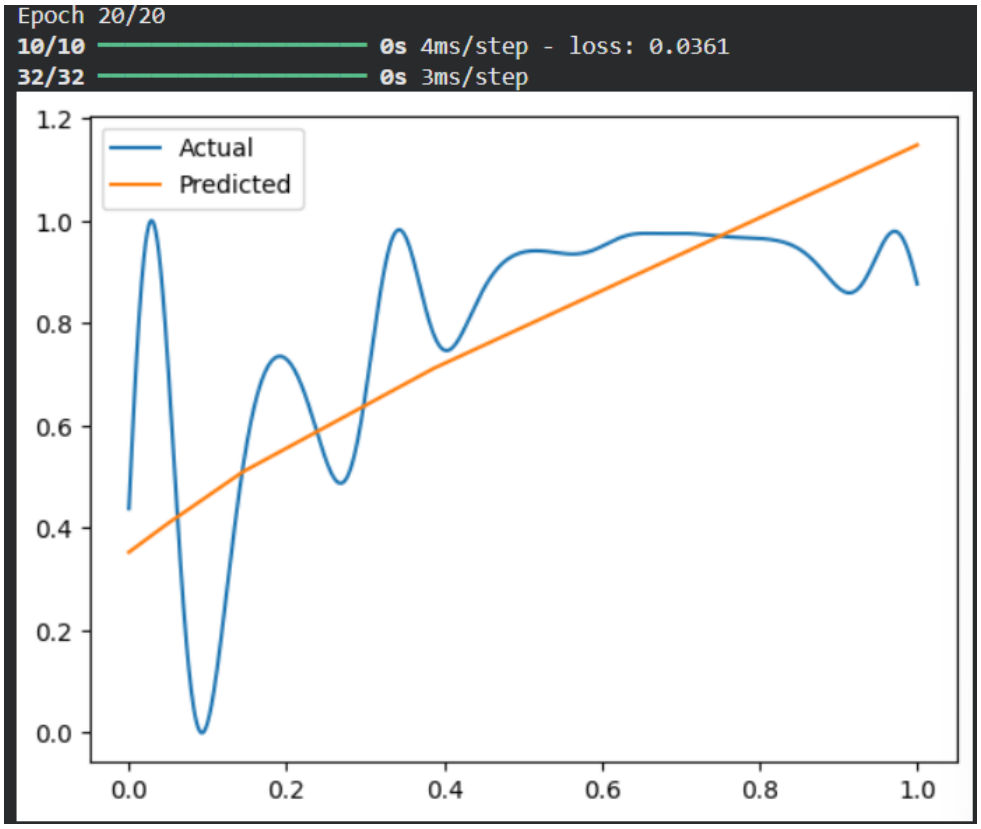
Як бачимо, результат погіршився, тому можемо зробити попередній висновок, що не завжди збільшення кількості шарів призведе до зменшення помилки.

3) Cascade forward NN – 1 шар, 10 нейронів:



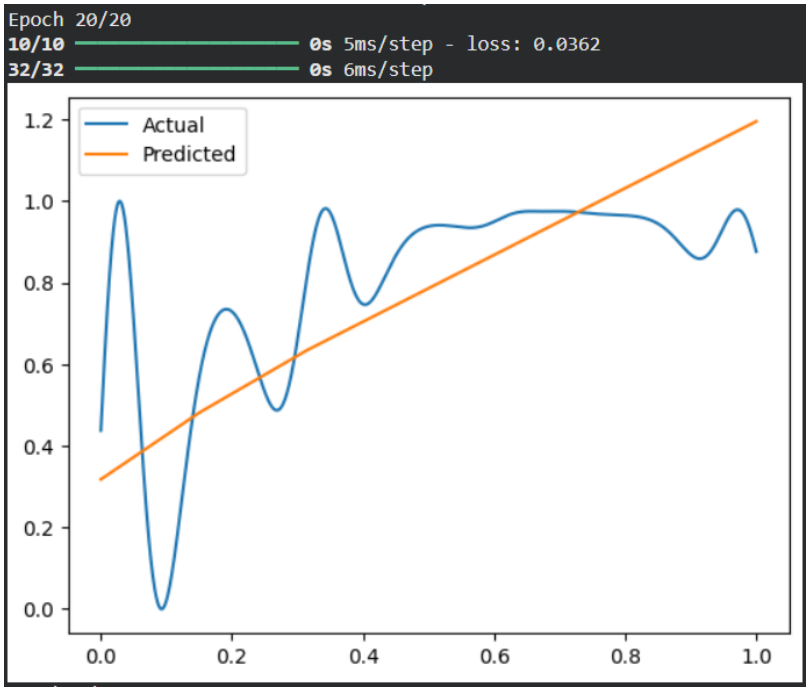
Як бачимо, ускладення типу НМ призвело до зменшення помилки.

4) Cascade forward NN – 1 шар, 20 нейронів:



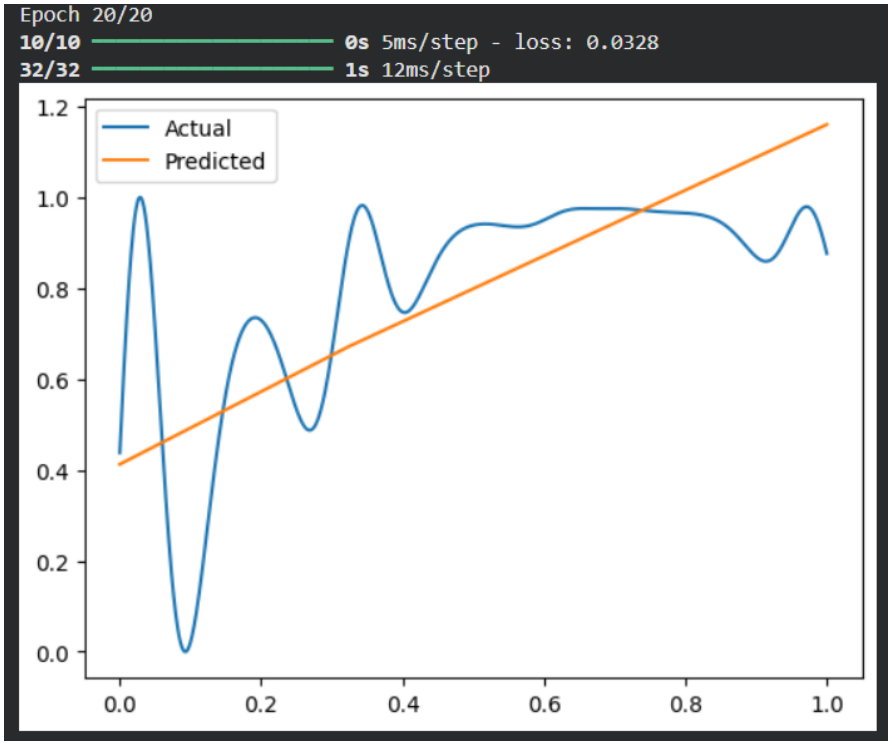
А ось тут, збільшення шарів призвело до зменшення помилки, хоч і незначного.

5) Elman NN – 1 шар, 15 нейронів:



Бачимо, що від попередньої нейронної мережі, в плані помилки не відрізняється, тому ще раз впевнюємося, що складно – не зазвичай швидко.

6) Elman NN – 3 шари, 5 нейронів:



Найкращий результат. Робимо висновок, збільшення кількості шарів посприяло покращенню роботи моделі.

Висновок: на даному лабораторному занятті я познайомився і дослідив структуру та принцип роботи нейронної мережі. Моделював три типи мереж: feed forward, cascade forward, elman. Feed forward в силу простоти показала непогані, але найгірші результати. Cascade forward показала себе трохи краще, проте, в деяких випадках була гіршою за попередньою в плані помилки та швидкодії. Elman показала себе найкраще, і тому може використовуватися в більш складних задачах. Загалом, збільшення кількості шарів чи складності навчання не завжди призводить до кращих результатів, тому важливо аналізувати, які вхідні дані наявні, та що саме треба з ними робити.

Відповіді на контрольні питання:

1. Що таке штучна нейронна мережа і за аналогією з чим вона була створена?

   Штучні нейронні мережі (ШНМ) – математичні моделі, а також їхня програмна та апаратна реалізація, побудовані за принципом функціонування біологічних нейронних мереж – мереж нервових клітин живого організму.

2. З яких основних елементів складається штучний нейрон?

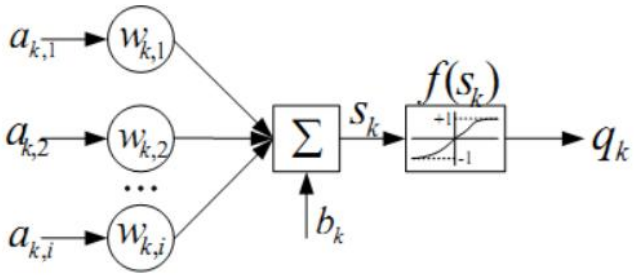


Рисунок 4.1 – Схема штучного нейрону

де q_k – вихідний сигнал k-го нейрона;

f – активаційна функція нейрона;

$a_{k,i}$ – вхідні сигнали k–го нейрона;

w_{ki} – синаптична вага k–го нейрона;

b_k – зміщення k–го нейрона.

3. Що таке архітектура нейронної мережі?

   Архітектура нейронної мережі — це план або структура того, як організовані та з’єднані між собою штучні нейрони (або вузли) у мережі.
4. Які бувають типи нейронних мереж (наприклад, прямого поширення, рекурентні)?
   - Персептрон (Perceptron): Основний блок для багатьох інших нейронних мереж. Використовується для бінарної класифікації.
   - Багатошаровий персептрон (Multilayer Perceptron, MLP): Складається з декількох шарів персептронів і використовується для різноманітних завдань, включаючи класифікацію та регресію.
   - Зворотнє поширення (Backpropagation) нейронна мережа: Використовується для навчання багатьох інших типів нейронних мереж, зокрема MLP.
   - Згорткові нейронні мережі (Convolutional Neural Networks, CNN): Ефективні для обробки зображень і використовують згорткові шари для виявлення патернів у зображеннях.
   - Рекурентні нейронні мережі (Recurrent Neural Networks, RNN): Призначені для обробки послідовних даних, таких як текст або часові ряди.
   - Довга короткострокова пам'ять (Long Short-Term Memory, LSTM): Вид рекурентних нейронних мереж, які здатні зберігати та використовувати інформацію на тривалий термін.
5. Що таке активаційна функція і для чого вона використовується?

   Функція активації оцінює, чи має нейрон бути «запущеним». Це математична функція, яка визначає вихід нейрона на основі його входу. Функції активації потрібні для нейронних мереж, оскільки без них вихід моделі був би просто лінійною функцією входу. Іншими словами, нейронні мережі не зможуть обробляти великі обсяги даних. Основна мета — створити нелінійність моделі, що дасть змогу нейромережі працювати зі складними залежностями й патернами.

6. Які найпоширеніші активаційні функції застосовуються на практиці?

   Східчаста функція — бінарна, одна із найпростіших функцій активації. Нейрон активується, якщо значення Y більше за певне порогове значення, тобто вихід дорівнює одиниці. Якщо Y менший за це порогове значення, то нейрон не буде активовано і його значення дорівнює нулю. Її часто використовують у задачах бінарної класифікації, де метою є класифікація вхідних даних за однією з двох категорій.

   Лінійну функцію активації також називають функцією без активації, оскільки за використання такої функції вихід завжди пропорційний входу. Таким чином, така функція надає зважену суму вхідних даних і повертає значення, подане в мережу, формуючи прямі зв'язки між входом і виходом. Вихід лінійної функції не обмежений кордонами.

   ReLu (Rectified Linear Unit) — одна з найпопулярніших функцій активації в сучасних нейромережах, особливо якщо йдеться про глибокі нейронні мережі (DNNs), які мають велику кількість шарів між входом та виходом. Ця функція замінює всі від'ємні вхідні значення на 0 та не змінює позитивні значення.

   Leaky ReLU (Leaky Rectified Linear Unit) повертає ненульовий результат, а не від'ємні значення, що перетворюються на нуль (як у випадку з ReLU). Це допомагає збільшити діапазон функції ReLU. Зазвичай значення a становить 0,01 або близько того.

   Сигмоїда — поширена нелінійна функція активації, яку переважно використовують у машинному навчанні. Він може прийняти будь-яке дійсне значення та дає вихідні значення в діапазоні між 0 і 1, з більшим вхідним значенням, ближчим до 1, і меншим, ближчим до 0.

   Tanh (гіперболічний тангенс) — це скоригована версія сигмоїди. Однак, на відміну від sigmoid, вихідний сигнал якого становить від 0 до 1, вихід tanh варіюється від -1 до 1. Більший вхідний сигнал ближче до 1, менший — ближче до -1.

Softmax — це нелінійна функція, яку переважно використовують для класифікаційних задач. Вона перетворює набори чисел (логітів) на ймовірності.

7. Що таке навчання нейронної мережі?

   Навчання нейронної мережі — це процес автоматичного коригування її внутрішніх параметрів (званих вагами та зміщеннями) для виконання певного завдання, такого як розпізнавання образів, прогнозування або класифікація.

8. Яка різниця між навчанням з учителем, без учителя та з підкріпленням?

   Навчання з учителем (Supervised Learning)

   - **Дані:** Використовується набір даних із **мітками (labels)**. Кожен вхідний приклад має відповідну правильну відповідь або категорію, надану "вчителем" (людиною або попередньо розміченою базою даних).
   - **Метод зворотного зв'язку:** Мережа отримує пряму вказівку на те, чи була її відповідь правильною, і наскільки вона помилилася.
   - **Мета:** Навчитися передбачати мітку для нових, невидимих раніше даних.
   - **Приклади:** Класифікація електронних листів як "спам" або "не спам", розпізнавання облич на фотографіях, прогнозування цін на житло.

   Навчання без учителя (Unsupervised Learning)

   - **Дані:** Використовується набір даних **без міток**. Мережі надаються лише вхідні дані без вказівки на правильні відповіді.
   - **Метод зворотного зв'язку:** Зворотний зв'язок відсутній. Мережа самостійно шукає приховані структури, закономірності, групи або аномалії в даних.
   - **Мета:** Зрозуміти внутрішню структуру даних, стиснути їх або виявити схожі об'єкти.

- **Приклади:** Кластеризація клієнтів на сегменти за купівельною поведінкою, виявлення аномалій у мережевому трафіку, зменшення розмірності даних.

Навчання з підкріпленням (Reinforcement Learning)

- **Дані:** Немає фіксованого набору даних. Натомість існує **агент** (нейронна мережа), який взаємодіє з **середовищем**.
- **Метод зворотного зв'язку:** Агент отримує **винагороду (reward)** або **штраф (punishment)** за свої дії. Немає прямої вказівки на "правильну" дію, є лише оцінка її наслідків.
- **Мета:** Навчитися послідовності дій (політиці) для максимізації сукупної винагороди з часом.
- **Приклади:** Навчання робота ходити, гра в шахи або Go проти людини, автономне водіння автомобіля, управління виробничими процесами.

9. Що таке помилка (функція втрат) у процесі навчання нейронної мережі?

   Вона виконує роль кількісного вимірювача того, наскільки погано (або добре) мережа виконує своє поточне завдання.

   Головна мета всього процесу навчання нейронної мережі — мінімізувати це значення втрат.

10. Яким чином здійснюється корекція ваг у нейронній мережі?

    Корекція ваг (weight adjustment) у нейронній мережі здійснюється за допомогою комбінації двох ключових алгоритмів: зворотного поширення помилки (backpropagation) та градієнтного спуску (gradient descent) (або його сучасних варіантів).

11. Що таке алгоритм зворотного поширення помилки (backpropagation)?

    Алгоритм зворотного поширення помилки використовує правила диференціального числення (зокрема, ланцюгове правило) для того, щоб "пройтися" назад по мережі — від вихідного шару до вхідного.

Під час цього зворотного проходу обчислюється, наскільки сильно кожен окремий вага або зміщення в мережі вплинув на загальну помилку. Результатом є градієнт — вектор, який вказує "напрямок" і "крутизну" збільшення помилки.

12. Які основні властивості нейронних мереж роблять їх ефективними для розв’язання задач ШІ?
    - Нелінійність: Вони можуть вивчати складні нелінійні взаємозв'язки в даних, які не під силу простим лінійним моделям.
    - Паралельна обробка: Архітектура мережі дозволяє обробляти інформацію паралельно, що прискорює навчання та виконання завдань (особливо на GPU).
    - Адаптивність: Вони можуть навчатися на основі досвіду (даних) та адаптувати свою продуктивність без прямого програмування кожної умови.
13. У яких сферах найчастіше застосовуються нейронні мережі?
    - Комп'ютерний зір: Розпізнавання зображень, виявлення об'єктів, автономне водіння, медична діагностика (аналіз рентгенівських знімків).
    - Обробка природної мови (NLP): Машинний переклад (Google Translate), голосові помічники (Siri, Alexa), аналіз настроїв тексту, чат-боти.
    - Охорона здоров'я: Прогнозування захворювань, розробка ліків, аналіз медичних даних пацієнтів.
    - Фінанси: Виявлення шахрайства, алгоритмічна торгівля на біржі, оцінка кредитних ризиків.
    - Рекомендаційні системи: Персоналізовані рекомендації товарів (Amazon) або фільмів (Netflix).
14. Які переваги нейронних мереж у порівнянні з класичними алгоритмами?
    - Вища точність на складних даних: Перевершують класичні методи в задачах із неструктурованими даними (зображення, звук, текст).

- Масштабованість: Ефективно працюють з дуже великими обсягами даних (Big Data).
- Менша потреба в ручній роботі: Автоматизують процес вилучення ознак, що зменшує потребу в експертних знаннях домену для підготовки даних.

15. Які недоліки та обмеження мають нейронні мережі?
    - Потреба у великих обсягах даних: Для ефективного навчання глибоким мережам зазвичай потрібні величезні набори даних.
    - Висока обчислювальна вартість: Навчання великих моделей вимагає потужного обладнання (GPU/TPU) та часу.