Міністерство освіти і науки України
Західноукраїнський національний університет

ЗВІТ
З МОДУЛЬНОЇ РОБОТИ №2

із дисципліни «Обчислювальний інтелект»

на тему «Дослідження класифікатора на основі нейронних мереж прямого поширення (FeedFoward Neural Networks)»

Виконав:
Студент: Рябий В.В.
Групи: КНм-11

Тернопіль 2024

Зміст

Вступ

Обчислюваний інтелект стає все більш потужним і важливим інструментом у різних сферах нашого життя. Одним із ключових напрямків цієї області є розробка та вдосконалення нейронних мереж - комп'ютерних систем, які намагаються моделювати роботу людського мозку для вирішення різноманітних завдань. У цьому модулі ми зосередимося на одному з основних типів нейронних мереж - нейронних мережах прямого поширення, або FeedForward Neural Networks (FFNN). FFNN є одним із найпоширеніших класифікаторів у сфері машинного навчання та глибокого навчання. Вони використовуються для різноманітних завдань, від розпізнавання образів до прогнозування часових рядів. Мета цього модулю – дослідження можливостей роботи нейронних мереж прямого поширення для класифікації зображень. Ми детально розглянемо теорію роботи FFNN, їх структуру та принципи навчання. Після цього ми перейдемо до практичної частини, де виконаємо дослідження класифікації даних за допомогою FFNN на реальних наборах даних. Цей модуль дозволить зрозуміти основні концепції нейронних мереж прямого поширення, їхні можливості та обмеження, а також навчити їх практично використовувати цей тип нейронних мереж.

Мета роботи: Дослідження можливостей роботи нейронних мереж прямого поширення для класифікації зображень.

Мій порядковий номер у журналі відповідає числу 22, я виконуватиму завдання варіанта №1, оскільки завдань всього 21 – таблиця 1.

Таблиця 1 –Завдання до модульної роботи.

Вар.	Data Set	1-Layer			2-Layer			3-Layer		
		solver	activations	max_iter	solver	activations	max_iter	solver	activations	max_iter
1	Handwritten Digits	'lbfgs'	'identity'	200	'lbfgs'	('logistic', 'tanh')	200	'lbfgs'	('logistic', 'tanh', 'relu')	200

Хід роботи

Крок 1 - Завантаження бібліотек та даних

Імпортуємо необхідні бібліотеки scipy.io, numpy та підмодуль MLPClassifier з бібліотеки sklearn.neural_network.

Завантажуємо дані з файлів train_digits.mat та test_digits.mat за допомогою scipy.io.loadmat

Крок 2 - Підготовка даних

Розділяємо дані на навчальні та тестові вибірки (X_train, y_train, X_test, y_test).

Нормалізуємо дані, розділяючи кожне значення на 255, щоб отримати значення від 0 до 1.

Крок 3 - Створення та навчання моделей.

Створюємо три різні моделі нейронних мереж з різними архітектурами за допомогою MLPClassifier. Кожна модель має свої параметри, такі як розміри шарів, метод оптимізації, функції активації тощо. Навчаємо кожну модель на навчальних даних (X_train, y_train) за допомогою методу fit.

Крок 4 - Оцінка моделей

Оцінюємо точність кожної моделі на навчальних та тестових даних. Для цього використовуємо функцію accuracy_score з sklearn.metrics. Також навчаємо та оцінюємо модель методу опорних векторів (SVM) на тестових даних.

Крок 5 - Візуалізація результатів.

Виводимо діаграму, на якому показані точності кожної моделі (1-шарової, 2-шарової, 3-шарової нейронної мережі та SVM). Для цього використовуємо бібліотеку matplotlib – рисунок 1.

```
import scipy.io
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Завантаження даних
train_data = scipy.io.loadmat('train_digits.mat')
test_data = scipy.io.loadmat('test_digits.mat')

X_train = train_data['X']
y_train = train_data['y'].ravel()
X_test = test_data['X']
y_test = test_data['y'].ravel()

# Нормалізація даних
X_train /= 255.0
X_test /= 255.0

# Розділення на навчальну та тестову вибірки
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2,
random_state=42)

# Моделі для 2-шарової нейронної мережі з різними значеннями активації
models_2_layer = [
    MLPClassifier(hidden_layer_sizes=(3, 3), solver='lbfgs', activation='logistic',
max_iter=200, random_state=42),
    MLPClassifier(hidden_layer_sizes=(3, 3), solver='lbfgs', activation='tanh',
max_iter=200, random_state=42)
]


# Моделі для 3-шарової нейронної мережі з різними значеннями активації
models_3_layer = [
    MLPClassifier(hidden_layer_sizes=(20, 7, 10), solver='lbfgs',
activation='logistic', max_iter=200, random_state=42),
    MLPClassifier(hidden_layer_sizes=(20, 7, 10), solver='lbfgs',
activation='tanh', max_iter=200, random_state=42),
    MLPClassifier(hidden_layer_sizes=(20, 7, 10), solver='lbfgs',
activation='relu', max_iter=200, random_state=42)
]

# Навчання та оцінка моделей для 2-шарової нейронної мережі
for model in models_2_layer:
```

```
    model.fit(X_train, y_train)
    accuracy = accuracy_score(y_test, model.predict(X_test))
    print("Accuracy for 2-layer model with activation", model.activation, ":",
accuracy)

# Навчання та оцінка моделей для 3-шарової нейронної мережі
for model in models_3_layer:
    model.fit(X_train, y_train)
    accuracy = accuracy_score(y_test, model.predict(X_test))
    print("Accuracy for 3-layer model with activation", model.activation, ":",
accuracy)

# SVM
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)
svm_accuracy = accuracy_score(y_test, svm_model.predict(X_test))

# Візуалізація результатів
labels = ['SVM', '2-Layer (logistic, logistic)', '2-Layer (tanh, tanh)', '3-Layer
(logistic, tanh, logistic)', '3-Layer (logistic, tanh, tanh)', '3-Layer (logistic,
tanh, relu)']
accuracies = [svm_accuracy] + [accuracy_score(y_test, model.predict(X_test)) for
model in models_2_layer + models_3_layer]
plt.bar(labels, accuracies)
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.title('Accuracy of Different Models')
plt.xticks(rotation=45, ha='right')
plt.show()
```

Також у ході розробки та написання коду було розділено значення параметру activation для моделей 2-шарової та 3-шарової нейронної мережі. Параметр activation має бути рядком, яка представляє одне з можливих значень: 'logistic', 'relu', 'identity', 'tanh'. Для цього потрібно створити додаткові обробки для поступово конання параметрів:

```
# Моделі для 2-шарової нейронної мережі з різними значеннями активації
models_2_layer = [
    MLPClassifier(hidden_layer_sizes=(3, 3), solver='lbfgs', activation='logistic',
max_iter=200, random_state=42),
    MLPClassifier(hidden_layer_sizes=(3, 3), solver='lbfgs', activation='tanh',
max_iter=200, random_state=42)
]


# Моделі для 3-шарової нейронної мережі з різними значеннями активації
models_3_layer = [
    MLPClassifier(hidden_layer_sizes=(20, 7, 10), solver='lbfgs',
activation='logistic', max_iter=200, random_state=42),
```

```
    MLPClassifier(hidden_layer_sizes=(20, 7, 10), solver='lbfgs',
activation='tanh', max_iter=200, random_state=42),
    MLPClassifier(hidden_layer_sizes=(20, 7, 10), solver='lbfgs',
activation='relu', max_iter=200, random_state=42)
]
```

Підсумовуючи цей код навчає та оцінює моделі з різними значеннями активації для 1-, 2- та 3-шарової нейронної мережі окремо і виводить точність кожної моделі.

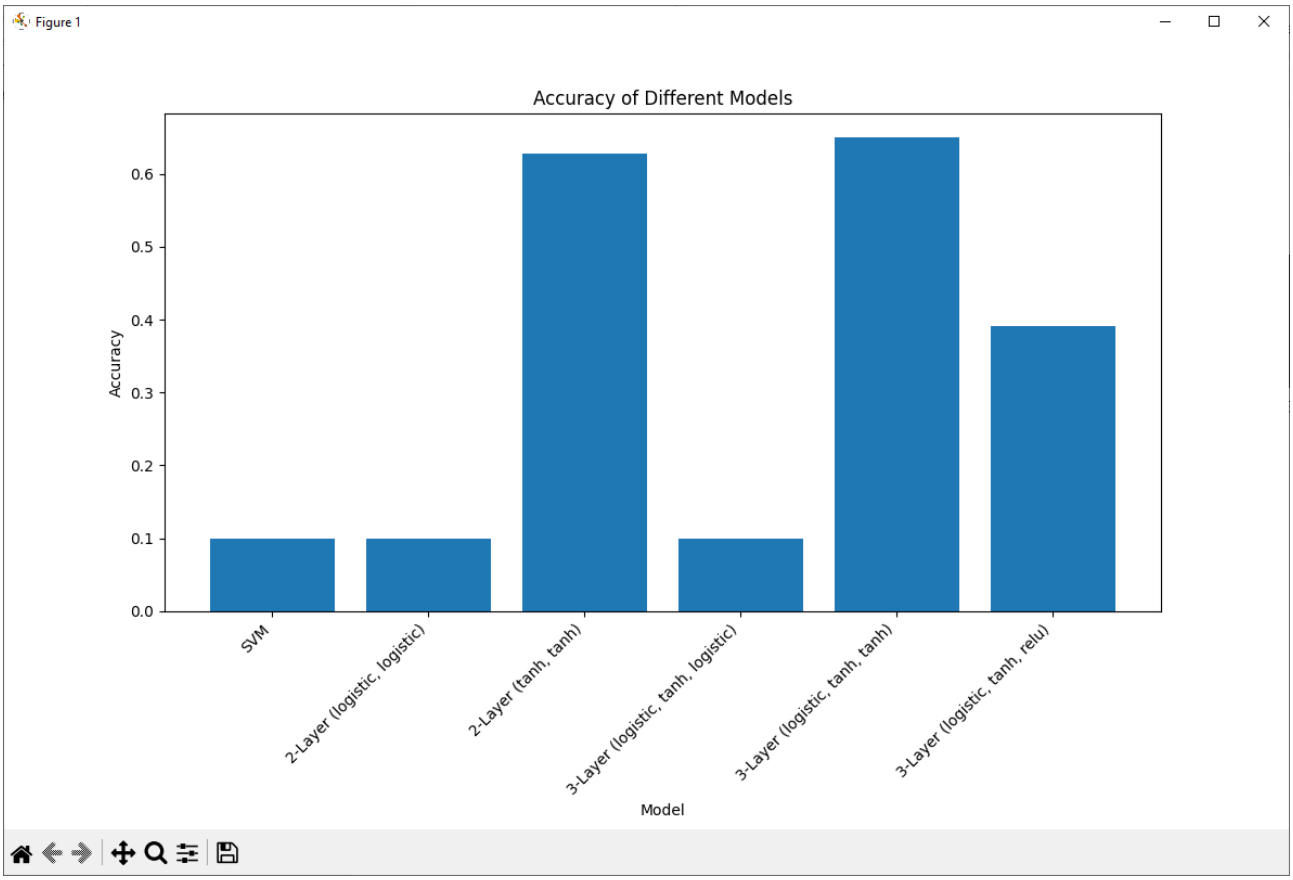

Рисунок 1 – Порівняльна діаграма точності

Висновок

У цій модульній роботі ми зосередились на одному з основних типів нейронних мереж - нейронних мережах прямого поширення, або FeedForward Neural Networks (FFNN). Дослідили можливості роботи нейронних мереж прямого поширення для класифікації зображень. Ми детально розглянули теорію роботи FFNN, їх структуру та принципи навчання. Після цього ми на практиці виконали дослідження класифікації даних за допомогою FFNN на реальних наборах даних. Цей модуль дозволив зрозуміти основні концепції нейронних мереж прямого поширення, їхні можливості та обмеження, а також навчились практично використовувати цей тип нейронних мереж.

Додаток А

На moodle відправлено звіт по роботі у форматі pdf.

У коментарі із завданням додано посилання на github із програмою: