TRƯỜNG ĐẠI HỌC THỦY LỢI

**KHOA CÔNG NGHỆ THÔNG TIN**

A blue and white logo

Description automatically generated with low confidence

**BÀI TẬP LỚN**

HỌC PHẦN: HỌC MÁY

**ĐỀ TÀI: DỰ ĐOÁN THỜI TIẾT**

**Phần 1: Lý thuyết**

* Trình bày lý thuyết của các thuật toán Neural Network.

Neural Network (Mạng nơ-ron nhân tạo) là một trong những phương pháp mạnh mẽ nhất để xử lý các bài toán phân loại và hồi quy. mô phỏng cách mà hệ thống thần kinh của con người hoạt động và học hỏi, giúp máy tính học cách phân loại, dự đoán và ra quyết định dựa trên dữ liệu đầu vào.

1. Input: Là 1 tập DL đã được gán nhãn
2. Output: Là 1 vecto dự đoán
3. Method:

* Cấu trúc mạng nơ-ron:
* Lớp đầu vào: Nhận dữ liệu đầu vào từ các tập dữ liệu (các đặc trưng như temp, humidity, precipitation).
* Lớp ẩn: Xử lý dữ liệu và tạo ra các đặc trưng mới dựa trên trọng số liên kết giữa các nơ-ron. (Có thế có nhiều lớp)
* Lớp đầu ra: Tạo dự đoán.
* Hàm kích hoạt: Mỗi nơ-ron trong mạng sử dụng 1 hàm kích hoạt để quyết định xem có kích hoạt tín hiệu tới nơ-ron tiếp theo hay không. Các hàm phổ biến:
* Sigmoid: Được dùng cho bài toán phân loại nhị phân
* ReLU: Phổ biến trong các mạng sâu
* Softmax: Dùng cho bài toán phân loại đa lớp (có nhiều nhãn như sun, rain, driz, snow, fog, …)
* Hàm mất mát

Để đánh giá độ chính xác của dự đoán, mạng nơ-ron sử dụng một hàm mất mát. Với bài toán phân loại, phổ biến là **Cross-Entropy Loss**

* Thuật toán lan truyền ngược

Mạng nơ-ron được huấn luyện bằng cách điều chỉnh trọng số giữa các nơ-ron thông qua quá trình lan truyền ngược, nhằm giảm thiểu hàm mất mát. Trọng số được cập nhật dựa trên thuật toán tối ưu hóa (**Optimization Algorithm**), phổ biến là **Gradient Descent**

1. Điều kiện dừng

* Số lần huấn luyện đạt đến mức đã định.
* Mạng nơ-ron đạt được độ chính xác thỏa đáng trên tập kiểm tra.
* Tốc độ học quá nhỏ, không thể cập nhật thêm trọng số.

**Phần 2: Ứng dụng trong thực tế**

***1. Mô tả bài toán***

* **Tên bài toán**: Dự đoán thời tiết.
* **Mục đích**: Dự đoán tình trạng thời tiết khi nhập các thông số thời tiết như precipitation, temp, humidity, wind.
* **Input**: Thông số thời tiết (precipitation, temp, humidity, wind).
* **Output**: Thời tiết (weather: sun, rain, driz, snow, fog).

Bài toán này nhằm dự đoán các kiểu thời tiết (sun, rain, driz, snow, fog) dựa trên thông tin về lượng mưa (precipitation), nhiệt độ (temp), độ ẩm (humidity), và sức gió (wind).

***2. Mô tả tập dữ liệu của bài toán***

* Dữ liệu:
* Gồm những chiều thông tin: precipitation, temp, humidity, wind, weather.
* Có 1461 dữ liệu.
* Các nhãn lớp gồm: sun, rain, driz, snow, fog.
* Mô tả ma trận dữ liệu (X), nhãn lớp (Y).
  + Ma trận dữ liệu (X) gồm các cột:
    - Precipitation: Thể hiện lượng mưa.
    - Temp: Thể hiện nhiệt độ.
    - Humidity: Thể hiện độ ẩm.
    - Wind: Thể hiện sức gió.
  + Nhãn lớp (Y) gồm:
    - Weather: Thể hiện tình trạng thời tiết (sun, rain, driz, snow, fog).
* Chia tập dữ liệu thành 2 phần với tỷ lệ 3:7 gồm:
* 439 dòng đầu: kiểm tra mô hình.
* 1022 dòng cuối: huấn luyện mô hình.

***3. Viết ứng dụng***

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import classification\_report, accuracy\_score

# Đọc dữ liệu từ file CSV

df = pd.read\_csv('./seattle-weather.csv')

# Xử lý dữ liệu thiếu (nếu có) bằng cách loại bỏ các hàng có giá trị null

df = df.dropna()

# Mã hóa thuộc tính weather (categorical) thành số

le = LabelEncoder()

df['weather\_encoded'] = le.fit\_transform(df['weather'])

# Sử dụng các cột numerical như precipitation, temp\_max, temp\_min, wind làm dữ liệu đầu vào

X\_data = df[['precipitation', 'temp\_max', 'temp\_min', 'wind']].values

# Chuẩn hóa (scale) các thuộc tính numerical để phù hợp với MLP

scaler = StandardScaler()

X\_data = scaler.fit\_transform(X\_data)

# Nhãn (target)

y\_data = df['weather\_encoded'].values

# Chia tập dữ liệu thành tập huấn luyện và kiểm tra (shuffle=True để trộn ngẫu nhiên dữ liệu)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_data, y\_data, test\_size=0.3, shuffle=True, random\_state=42)

# Xây dựng mô hình MLP (Multilayer Perceptron - Neural Network)

clf = MLPClassifier(hidden\_layer\_sizes=(100, 50), max\_iter=1000, activation='relu', solver='adam', random\_state=42)

# Huấn luyện mô hình

clf.fit(X\_train, y\_train)

# Dự đoán trên tập kiểm tra

y\_pred = clf.predict(X\_test)

# In kết quả dự đoán so với thực tế (in 10 dòng để dễ đọc)

print("Thực tế \t Dự đoán")

for i in range(10):

    print(le.inverse\_transform([y\_test[i]])[0], "\t\t", le.inverse\_transform([y\_pred[i]])[0])

# Đánh giá mô hình và in bảng phân loại chi tiết

print("\nBáo cáo phân loại chi tiết:\n")

print(classification\_report(y\_test, y\_pred, target\_names=le.classes\_))

# In độ chính xác (accuracy)

accuracy = accuracy\_score(y\_test, y\_pred)

print('Tỷ lệ dự đoán đúng (accuracy):', np.around(accuracy \* 100, 2), '%')

***4. Phân tích kết quả của chương trình***

Neural Network

* Tỷ lệ dự đoán đúng trên tập test: 84.97 %
* Tỷ lệ dự đoán sai trên tập test: 15.03%

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated