COP528 Applied Machine Learning (LABs)

LAB Day 07. Neural Network

Introduction

The aim of this session is to understand Neural Network. The experiment includes **1** tutorial example and **2** tasks. Please follow the tutorial to learn the basic functions and then complete the tasks.

Tutorial 01. Simple neural network example

Review the example of backpropagation in the lecture slides so that you have a better understanding of Backpropagation algorithm. Then run the following code to understand how to create a simple Neural Network with sklearn.

(1) Loading dataset

```
# Use the iris dataset as an example
import numpy as np # for numpy
# Now import a multiclassification model from a neural network for training multiclassification data
from sklearn.neural_network import MLPClassifier
# Now import the libraries in sklearn that are used to evaluate the prediction metrics, such as confusion matrices
and classification reports
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
iris_data = load_iris()
X = iris_data.data
y = iris_data.target
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.3, random_state=0)
```

(2) Calling the model in sklearn

```
# """
# Main parameters.
# hidden_layer_sizes: number of hidden layer cells (tuples), e.g. (100,100,100,50)
# activation : activation function, {'identity', 'logistic', 'tanh', 'relu'}, default 'relu'; [f(x) = x, 1/(1+exp(-x)), tanh(x), max(0, x)]
# solver : solver, {'lbfgs', 'sgd', 'adam'}, default 'adam '; [newton method, stochastic gradient descent, adaptive momemtum]
# alpha : L2 regularization parameter, float, optional, default 0.0001
# batch_size : batch size, optional, not applicable to 'lbfgs', default 'auto', in this case, batch_size=min(200, n_samples)`
# learning_rate : learning rate, {'constant', 'invscaling', 'adaptive'}, default 'constant', only for gradient descent sgd
# learning_rate_init : initial value of the learning rate, optional, default 0.001, only for sgd or adam
# power_t : descent exponent, optional, default 0.5, applies to 'invscaling',learning_rate_init/pow(t,power_t), sgd only
# max_iter : maximum number of iterations, optional, default 200, number of iterator convergence iterations, for sgd/adam, represents the number of epochs, not the number of descent steps
# shuffle : per iteration, whether to shuffle or not, optional, default True, only for sgd or adam
# random_state: default None; if int, random number generator seed, if RandomStates instance, random number generator, if None, np.random
# tol : tolerance, optional, default le-4, stop iterating if loss does not reach this value in two consecutive iterations unless set to 'adaptive', otherwise
# beta_1 : adam exponent decay parameter 1, optional, default 0.9
# beta_2 : adam exponential decay parameter 2, optional, default 0.999
# epsilon : adam numerical stability value, optional, default 1e-8
# """

# Multiple parameters in parentheses, if not written, default values are used, we generally want to build the hidden layer structure, debug the regularization parameters and set the maximum number of iterations
mlp = MLPClassifier(hidden_layer_sizes=(20),alpha=0.01,max_iter=300)
# The model can be trained by calling the fit function. The general method of training by calling the model function is fit()
mlp.fit(X_train,y_train) # Here the y value needs to be reduced to a one-dimensional array
# This is how the model is trained, and we can then call a variety of functions to get the trained parameters
# For example, to get the accuracy rate
print('The accuracy of the test set is:',mlp.score(X_test,y_test))
# e.g. output the current Current value of the loss function
print('The Current value of the loss function of the training set is:',mlp.loss_)
# For example, output the weights of each theta
print('The weight value :',mlp.coefs_)
```

(3) Confusion matrix and classification reports

```
# The confusion matrix and classification report is an indicator of the predicted and true values
# The confusion matrix provides a visual indication of the number of correct and incorrect classifications, and the category to which the correct samples were incorrectly classified
matrix_test = confusion_matrix(y_test,mlp.predict(X_test))
print('The confusion matrix for the training set is:\n',matrix_test)
# There are multiple metrics in the classification report to evaluate how good the prediction is.
# '''
# TP: Prediction is 1 (Positive), actual is also 1 (Truth-prediction is correct)
# TN: predicted 0 (Negative), also 0 (Truth - predicted correctly)
# FP: predicted 1 (Positive), actually 0 (False - wrong prediction)
# FN: Prediction is 0 (Negative), actual is 1 (False-prediction is wrong)
# '''
report_test = classification_report(y_test,mlp.predict(X_test))
print(f'The classification report for the training set is: \n {report_test}')
```

Task 01. Neural network classification experiments

Use sklearn to define a MLP Neural Network and use it to train a model for seeds quality classification. Analyze the MLP with different numbers of layers and different numbers of neurons on each layer.

The Wheat Seeds Dataset involves the prediction of species given measurements of seeds from different varieties of wheat.

It is a binary (2-class) classification problem. The number of observations for each class is balanced. There are 210 observations with 7 input variables and 1 output variable. The variable names are as follows:

Area.

Perimeter.

Compactness

Length of kernel.

Width of kernel.

Asymmetry coefficient.

Length of kernel groove.

Class (1, 2, 3).

(1) Use the following code to load the data.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
df = pd.read_csv("https://raw.githubusercontent.com/lborohfang/COP528AML_Files/main/seeds_dataset.csv")
print(df.head(10))
seeds_data = np.asarray(df.iloc[:,:-1])
seeds_label= np.asarray(df.iloc[:,-1])
X_train, X_test, y_train, y_test = train_test_split(
   seeds_data, seeds_label, test_size=0.3, random_state=0)
```

Task 02. Neural network classification and visualisation

In Day 06 Tutorial 01, you have created a mesh grid to visualise the classification boundary of SVM. Please use the same idea and data to visualize the boundaries of MLP neural networks with 1 and 2 hidden layers.

Task 03. glass classification

The glass classification dataset, from the American Forensic Science Service, defines six types of glass based on their oxide content (i.e. Na, Fe, K, etc.). Please use this dataset for the classification task using MLP NN, SVM and kNN, compare the classification accuracy and adjust the MLP NN to improve its results.

Loading data

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
data=pd.read_csv('glass.csv')
# print(data.head(5))
X=data.iloc[:,:-1]
y=data.iloc[:,-1]
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.3, random_state=0)
```