**0College of Computer Science and Engineering**

**Department of Computer Science and Artificial Intelligence**

**CCAI-321: Artificial Neural Networks**

**Lab#8 Neural Networks using sickit-learn Python**

Student ID: 1845919

Marks Obtained = / 15 PLO = S1 - AI

**Marks:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Questions** | **1** | **2** | **3** | **4** | **5** | **Total** |
| **Allocated** | **2** | **2** | **2** | **2** | **7** | **15** |
| **Obtained** | **2** | **2** | **2** | **2** | **7** | **15** |
|  |  |  |  |  |  |  |
| **Allocated** |  |  |  |  |  |  |
| **Marks** |  |  |  |  |  |  |

**Weighted Marks:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Allocated** |  |  |  |
| **Obtained** |  |  |  |

# Objectives

• Train and test a neural network with a toy example using sickit-learn in python • Train and test a neural network on a real dataset

# Lab Tool(s)

[Download Python | Python.org](https://www.python.org/downloads/)

[Anaconda | Individual Edition](https://www.anaconda.com/products/individual)

# Lab Deliverables

Submit a pdf document on Blackboard containing your solution to the lab assessment at the end of this document.

# Sickit-Learn Neural Network Models (Supervised)

Multi-layer Perceptron (MLP) is a supervised learning algorithm. Given a set of features p and a target t, it can learn a non-linear function approximator for either classification or regression.

The advantages of Multi-layer Perceptron are:

* Capability to learn non-linear models.
* Capability to learn models in real-time (on-line learning) using partial\_fit.

The disadvantages of Multi-layer Perceptron (MLP) include:

* MLP with hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore different random weight initializations can lead to different validation accuracy.
* MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations.
* MLP is sensitive to feature scaling.

**Note**: The implementation in sickit-learn is not intended for large-scale applications. In particular, scikit-learn offers no GPU support. For much faster, GPU-based implementations, as well as frameworks offering much more flexibility to build deep learning architectures, see Keras and Tensorflow.

**Example 1.** In sicki-tlearn, Class [**MLPClassifier**](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier) implements a multi-layer perceptron (MLP) algorithm that trains using Backpropagation.

MLP trains on two arrays: array X of size (n\_samples, n\_features), which holds the training samples represented as floating point feature vectors; and array y of size (n\_samples,), which holds the target values (class labels) for the training samples.

The example below illustrates how to use the MLPclassifier in a toy example.

|  |
| --- |
| **from** **sklearn.neural\_network** **import** MLPClassifier **from** **sklearn.datasets** **import** make\_classification **from** **sklearn.model\_selection** **import** train\_test\_split    X, y = make\_classification(n\_samples=100)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, stratify=y)    clf = MLPClassifier(max\_iter=300).**fit**(X\_train, y\_train)  clf.**predict\_proba**(X\_test[:1])    clf.**predict**(X\_test[:5, :])    clf.**score**(X\_test, y\_test) |

|  |  |
| --- | --- |
| [**fit**(](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier.fit)X, y) | Fit the model to data matrix X and target(s) y. |
| [**get\_params**(](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier.get_params)[deep]) | Get parameters for this estimator. |
| [**predict**(](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier.predict)X) | Predict using the multi-layer perceptron classifier |
| [**predict\_log\_proba**(](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier.predict_log_proba)X) | Return the log of probability estimates. |
| [**predict\_proba**(](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier.predict_proba)X) | Probability estimates. |

[**score**(](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier.score)X, y[, sample\_weight]) Return the mean accuracy on the given test data and labels.

|  |  |
| --- | --- |
| [**set\_params**(](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier.set_params)\*\*params) | Set the parameters of this estimator |

**Q1. Repeat the example above, with max\_iter = 10, and report the score in the table below. Run the same script again for five times, does the score changes? Explain why.**  **[2 marks]**

|  |  |
| --- | --- |
| **Run** | **Score** |
| **1** | **0.64** |
| **2** | **0.68** |
| **3** | **0.72** |
| **4** | **0.60** |
| **5** | **0.40** |

**Q2. Set the random state for the following functions: make\_classification, train\_test\_split, MLPClassifier. Check the documentation for each of these function to know how to set the random state. Next, repeat the example above, with max\_iter = 10, and report the score in the table below. Run the same script again**

**for five times, does the score changes? Explain why.**  **[2 marks]**

|  |  |
| --- | --- |
| **Run** | **Score** |
| **1** | **0.72** |
| **2** | **0.72** |
| **3** | **0.72** |
| **4** | **0.72** |
| **5** | **0.72** |

**Q3. Repeat the example above, with max\_iter = 50, and report the score in the table below. Run the same script again for max\_iter = 100, 200, and 300. does the score changes? Explain why.**  **[2 marks]**

|  |  |
| --- | --- |
| **Max\_iter** | **Score** |
| **50** | **0.92** |
| **100** | **0.84** |
| **200** | **0.88** |
| **300** | **0.88** |

**Q4. You are given the dataset “dataset\_spine” in csv format. Read the dataset using read\_csv and answer**

**the questions below.**  **[2 marks]**

1. **How many columns (features / attributes) does the dataset has? List the column names**
2. **How many rows (samples / records) does the dataset has?**
3. **What is the data type for each feature? Numerical, categorical, …**
4. **“class\_att” is the target value. Why is the datatype of the target value?**
5. **Is this a regression or a classification problem?**

**Q5. Train an MLPClassifier, follow the instructions below: [1+2+2+2 marks]**

1. **Split the dataset into 70% training and 30% testing**
2. **Initialize an MLPClassifier with 1 hidden layer and 7 neurons**

1. **Initialize the learning rate to 0.1 and the max\_iter to 1000**
2. **Fit the model given the training set**
3. **Predict the output of the testing set**

1. **Compute and report the score**
2. **What is the effect of the number of neurons on the result? To check its effect, repeat the steps from 2 to 6, but change the number of neurons in the hidden layer. How does this affect the score?**
3. **What is the effect of the number of hidden layers on the result? To check its effect, repeat the steps from 2 to 6, but change the number of hidden layers. How does this affect the score?**

1. **Apply gridsearch (GridSearchCV) to find the best hyperparamters for this problem. Use these Hyperparameters: number of hidden layers, number of neurons in each layer, learning rate, max iterations.**