**Yog Chaudhary**

11727095

ADTA 5550: Deep Learning with Big Data Week 3 Assignment

Professor: Dr. Hamidreza Moradi

University of North Texas  **Feb 07, 2024**

**PART I: One-Hot Encoding (20 Points)**

**Question 1.1: Using critical thinking and based on the lectures, is it necessary to perform any kind of coding (integer coding or one-hot coding) on these class values for better performance of the project?**

One-hot encoding is a technique to convert categorical data into a numerical format that machine learning models can understand. It creates new features based on the unique values in the categorical column. For instance, if a column has three unique categories, one-hot encoding will create three new columns, one for each category, and use binary values (0 or 1) to indicate the presence of each category.

While one-hot encoding is used for nominal data (data without an inherent order), ordinal data (with a natural order, like grades A-F) can be converted into numerical values through ordinal encoding. This assigns a unique integer to each category value based on the order. However, it's essential to ensure that this numerical transformation makes sense for the model and the data's context, as it imposes an order that may not naturally exist in nominal data. One-hot encoding is beneficial because it avoids imposing such an order, making it a common choice for preprocessing categorical data before feeding it into a model, especially in regression and deep learning models.

It is necessary to perform encoding for below reasons:

* Numerical Representation-model can comprehend and learn from them as deep learning models frequently work with numerical data.
* Avoiding Misinterpretation
* Neural networks perform better when working with numerical data.
* Compatibility with Libraries

**Question 1.2: If the answer to Question 1.1 is “YES,” what type(s) of encoding needs to be done to process the class values before using the dataset for the deep learning project?**

We should use **one-hot encoding** because this method represents each category as a binary vector. Each category corresponds to a unique combination of 0s and 1s, with a 1 in the position corresponding to the category and 0s in all other positions. This encoding is widely used because it avoids ordinal assumptions and treats each category as distinct.

One of the most urgent issues with deep learning is how difficult it is to function with plain knowledge. It is much better to convert our feedback information to statistics because PCs are so powerful and skilled when it comes to working with numbers.

Factors with name values rather than numeric properties make up all the information.

* The range of prospective traits is usually restricted to a specific number.
* Ostensible elements are commonly mentioned in passing as all-encompassing considerations.

The following situations occur on occasion:

* The "pet" and "feline" characteristics' "canine" and "feline" attributes
* A variable has the colors "red," "green," and "blue" in it.
* The variables "First," "Second," and "Third" with the attributes "first," "Second," and "Third," separately.
* Each number corresponds to a letter from the collection of letters. A few classes, such as common asking, could have a distinguishing relationship.

Simple computations can handle plain facts:

* Undiluted information can be used to progress a decision tree directly without the need for information management (this relies upon the execution).
* However, this is more of a need for the calculations' effective execution than a significant constraint on the actual techniques. Many AI calculations cannot function straightforwardly on name information and require that all information and outcome elements be numeric.
* However, this is more of a need for the calculations' effective execution than a significant constraint on the actual techniques. Many AI calculations cannot function straightforwardly on name information and require that all information and outcome elements be numeric.

**Question 1.3: Based on the answer to Question 1.2, explain the steps of what needs to be done for each type of encoding.**

The conversion is done in two steps:

1. **Th Integer coding.**

Step 1: Create a Mapping: To perform integer encoding, create mapping or dictionary that associates each unique category (class label) with a unique integer value.

For example, if you have class labels 'A', 'B', 'C', 'D', you will create a mapping like this: {'A': 0, 'B': 1, 'C': 2, 'D': 3}.

Step 2: Replace Labels: Apply the mapping to your dataset. Replace the original categorical class labels with their corresponding integer values based on the mapping. This converts the categorical data into numerical format.

1. **One – hot coding.**

Step 1: Create Binary Vectors: One-hot encoding represents each category as a binary vector (array) where each category corresponds to a unique combination of 0s and 1s. The length of the binary vector is equal to the number of unique categories. Each category is represented by a vector with a 1 in the position corresponding to the category and 0s in all other positions.

Step 2: Apply Encoding: we create a binary vector as described in step 1 and replace the original class label with this binary vector. For example:

A black screen with white text

Description automatically generated

**For Example, Second Step: One-Hot Coding**

* Red, Green, and Blue are arranged in a vector (1D cluster) that is transformed into a framework (2D exhibit = vector of vectors) via one-hot coding:



From a collection of all out values, one-hot coding converts a vector (1D exhibit) into a network (2D cluster = vector of vectors) ("red," "green," and "blue").

**Important Point:**

* Whole number coding is often sufficient when a dataset's class attribute only has two distinct values, such as "Yes" and "No," "Supported" and "Dismissed," etc.

**Question 1.4: Based on the answer to Question 1.2, perform the necessary encoding tasks to transform the class values before using the dataset for the deep learning project.**

* Hare Screenshot from jupyter Notebooks

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The number encoding is insufficient for classifying factors without this ordinal connection. In example, allowing the model to anticipate a feature being requested across categories and then encoding it might result in subpar presentation or unexpected results (forecasts somewhere between classes). To encode the full number representation in this case, a one-shot encoding can be used. It is possible to modify the class values before using the dataset by looking at the screenshot up top.

**4.PART II: MLPs (Fully Connected Neural Networks) with Keras (50 Points)**

**First Step, design an MLP neural network (the same as discussed in the lectures)**

* I added the Iris.csv to Data subfolder under JPTR\_NTBK
* First import the Iris.csv dataset and import the necessary libraries such as pandas, NumPy, matplotlib.pyplot as plt, from keras.models import Sequential, from keras.layers import Dense, from keras. utils import plot model.
* Do the integer encoding for the Species column and perform one- hot encoding by using import OneHotEncoder. Now, split the dataset into training and testing sets.

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

* **Define the MLP classifier architecture as follows:**

hidden layer dimensions = (5,) # Hidden unit count in the hidden layer

input dim = 4 # Input feature count

output dim = len(label encoder.classes\_) # Amount of output classes

* **Make the figure and axis objects:**

fig, ax = plt.subplots ()

Display the input Neuron:

ax.scatter(np.zeros(input dim), np.arange(input dim), color='pink,' label='Input Layer,' s=300)

* **Draw the hidden layer layer:**

ax.scatter(np.ones(hidden layer sizes[0]) \* 1, np.arange(hidden layer sizes[0], color='red', label='Hidden Layer', s=300)

* **Display the output layer:**

ax.scatter(np.ones(output dim) \* 2, np.arange(output dim), color='orange', label='Output Layer', s=300)

* **Link the input Neuron to the hidden layer:**

ax.plot([0, 1], I j], color='gray', linewidth=2, linestyle='dashed') for I in range(input dim): for j in range(hidden layer sizes[0]):

* **Link the hidden layer to the output layer as follows:**

for I in range(hidden layer sizes[0]): ax.plot([1, 2], I j], color='gray', linewidth=2, linestyle='dashed')

A screenshot of a computer

Description automatically generated

* Now, we need to add the labels to the layers and set the axis limits and labels for getting the plot.
* Finally we can get the neural network by plt.show()

**The Neural network for the Iris dataset with keras is:**

A diagram of a network

Description automatically generated

**Result from the above Neural network:**

"It's merely a visual representation of a multi-layer perceptron (MLP) architecture featuring a single hidden layer.

The diagram illustrates the construction of an MLP classifier with a solitary hidden layer, consisting of five neurons each. The input layer comprises four neurons, whereas the output layer matches the number of classes in the dataset.

This diagram can aid in grasping the interconnections between the layers and how information flows within the network during both training and prediction. It's evident from the diagram that the input layer connects to the hidden layer, and ultimately, to the output layer.

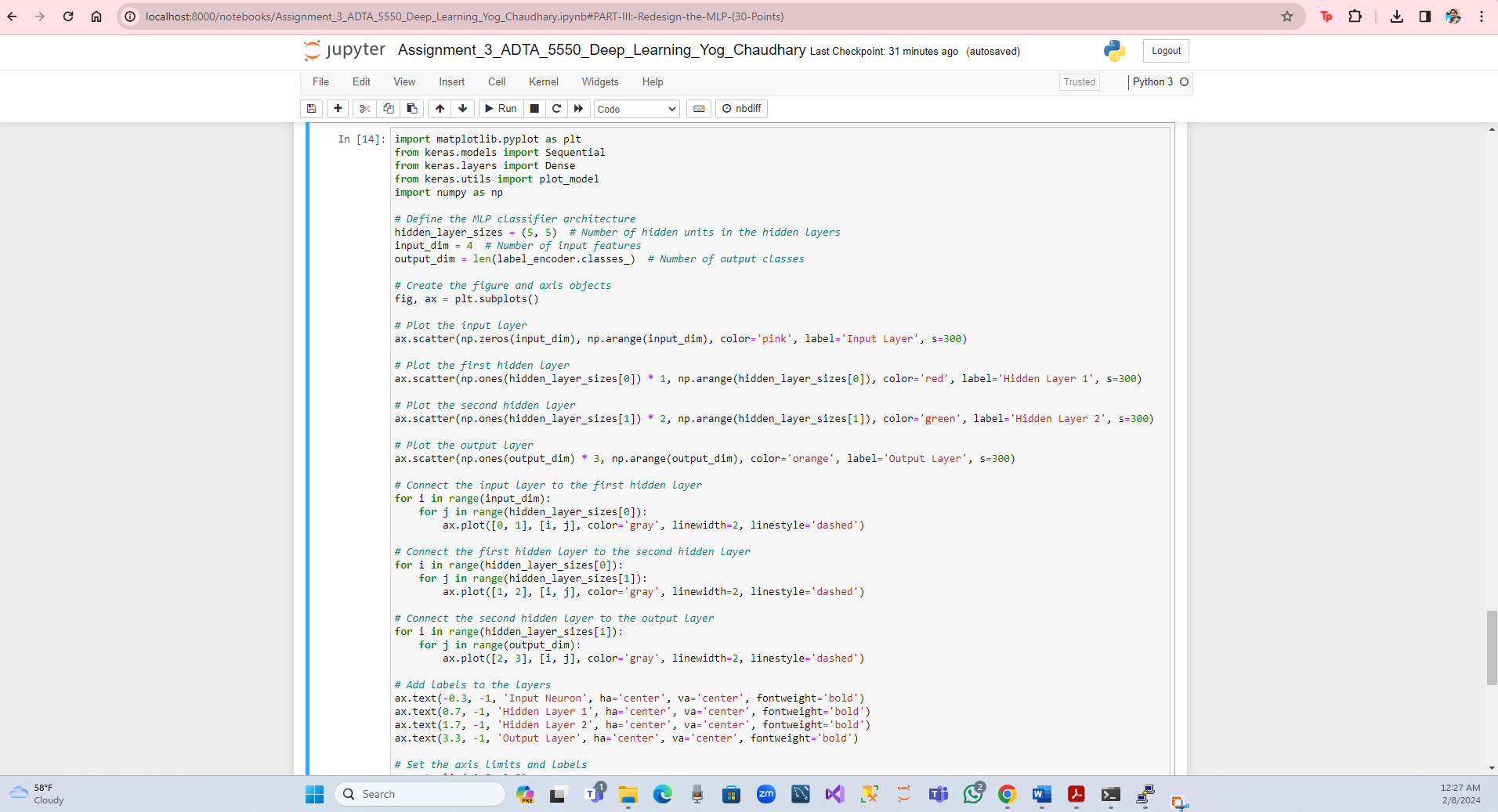
**PART III: Redesign the MLP (30 Points)**

**First Step, redesign the above MLP neural network by adding one hidden dense layer**

By using the libraries, we can plot two hidden layers in the MLP neural network.

The libraries are from keras.models import Sequential, from keras.layers import Dense

From keras.utils import plot\_model



A screenshot of a computer

Description automatically generated

After redesign the above MLP neural network, now we will add another dense hidden layer for the neural network.

MLP is frequently employed in research on computational neuroscience, equally allocated.

registering, and controlling learning challenges. Discourse acknowledgment, picture.

Acknowledgment and machine interpretation are all included in some of the applications.

MLPs and other feedforward networks are like tennis or ping pong. They mostly move in two.

directions when going back and forth. Given that, each estimate is a test of what we presume.

to know, and each reaction is a criticism informing us of how incorrect we are, these.

estimates and reactions might be compared to a game of ping pong as a kind of accelerated.

research. 66.67 percent accuracy is required to evaluate the module using the Cold approval.

Here, we consider the one hot Y test and the bounds X test.

A diagram of a network

Description automatically generated

**Result from the above neural network:**

This is a simple representation of a multi-layer perceptron (MLP) architecture with two hidden layers.

In the diagram, you can observe an MLP classifier with two hidden layers, each containing five neurons. The input layer consists of four neurons, and the output layer has as many neurons as there are classes in the dataset.

By studying this visual, you can gain insights into the interconnectedness of the layers and how data flows through the network during both training and prediction. The plot clearly illustrates the connection from the input layer to the first hidden layer, followed by a link to the second hidden layer, and finally, to the output layer.

Accuracy and Normalized accuracy for the above neural network is 66.67%.

**References:**

By Simplilearn AI & machine Learning, link <https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-keras>

Kassiani Nikolopoulou (Scribbr Team) June 09, 2023, link <https://www.scribbr.com/ai-tools/deep-learning/>

TensorFlow Core link <https://www.tensorflow.org/guide/keras>