21.3 Lesson Summary - Neural Networks and Deep Learning

Neural Networks classify data by constructing a series of nodes for test data. The nodes and their connections indicate the likelihood of traversal to a different node and the probability of arriving at a given output. These models can be trained to identify the components in an image, handwriting recognition and beyond.

Concept: **Tensor Flow Playground** is a useful data visualization website that allows you to see how a neural network model for different kinds of datasets is fit.

* Suppl link: <http://playground.tensorflow.org/>

Concept: **TensorFlow** is a free open-source machine learning library that can be used to build neural networks. **Keras** is an API built on top of TensorFlow that lets you interact with TensorFlow in a more straight forward, intuitive manner.

Concept: The input for a neural network must be numerical vector data. Nonnumerical or categorical data must be converted into numerical data before it can be input into a neural network. If your data has 3 categories the data must be converted into a 3-number vector with 2 numbers being zero and the vector component corresponding to the category set to one. A vector with all zeros and a single number one is referred to as **one-hot** encoding. To convert categorical or label data into numerical vector data you can use the following code:

*from sklearn.preprocessing import LabelEncoder*

*from keras.utils import to\_categorical*

*label\_encoder = LabelEncoder()*

*label\_encoder.fit(**data\_to\_convert)*

*encoded\_data = label\_encoder.transform(data\_to\_convert)*

*one\_hot\_y = to\_categorical(encoded\_y)*

*one\_hot\_y*

* Activity: 01-Ins\_One\_Hot\_Encoding
* Suppl link: https://en.wikipedia.org/wiki/One-hot

Concept: To create a neural network model for a dataset where you're using 2 features to classify data into 3 different categories using one set of 4 hidden nodes you can use the following code:

*from tensorflow.keras.layers import Dense*

*number\_inputs = 2*

*number\_hidden\_nodes = 4*

*model.add(Dense(units=number\_hidden\_nodes,*

*activation='relu', input\_dim=number\_inputs))*

*number\_classes = 3*

*model.add(Dense(units=number\_classes, activation='softmax'))*

*model.compile(optimizer='adam',*

*loss='categorical\_crossentropy',*

*metrics=['accuracy'])*

Once your model is created, you can train it using the following code:

*model.fit(*

*X\_train\_scaled,*

*y\_train\_categorical,*

*epochs=1000,*

*shuffle=True,*

*verbose=2*

*)*

To validate the trained model, you can use the following code:

*model\_loss, model\_accuracy = model.evaluate(*

*X\_test\_scaled, y\_test\_categorical, verbose=2)*

*print(f"Loss: {model\_loss}, Accuracy: {model\_accuracy}"))*

To make predictions on new data you can use the following code:

*new\_data = np.array([[0.2, 0.3, 0.4]])*

*print(f"Predicted class: {model.predict\_classes(new\_data)}")*

* Activity: 02-Evr\_First\_Neural\_Network, 04-Stu\_Moons\_Neural\_Networks

Concept: **Deep Learning** allows the algorithm to discover patterns in the data itself without a user provided set of classifications. Deep Learning is especially useful with categorical data. To create a Deep Learning model with 2 input dimensions, 2 hidden layers of 5 nodes each to generate 2 categories you could use the following code:

*from tensorflow.keras.models import Sequential*

*deep\_model = Sequential()*

*deep\_model.add(Dense(units=5, activation='relu', input\_dim=2))*

*deep\_model.add(Dense(units=5, activation='relu'))*

*deep\_model.add(Dense(units=2, activation='softmax'))*

*deep\_model.compile(optimizer='adam',*

*loss='categorical\_crossentropy',*

*metrics=['accuracy'])*

Training and testing a deep learning model is the same as with a neural network:

*deep\_model.fit(*

*X\_train\_scaled,*

*y\_train\_categorical,*

*epochs=100,*

*shuffle=True,*

*verbose=2*

*)*

*model\_loss, model\_accuracy = deep\_model.evaluate(*

*X\_test\_scaled, y\_test\_categorical, verbose=2)*

*print(f"Deep Neural Network - Loss: {model\_loss}, Accuracy: {model\_accuracy}")*

* Activity: 03-Evr\_Deep\_Learning, 05-Stu\_Deep\_Voice\_Deep\_Learning

Concept: Fitting models can be time consuming so they can be **saved** for future use with the following code:

*model.save("my\_saved\_model.h5")*

To load a saved model you can use the following code:

*from tensorflow.keras.models import load\_model*

*voice\_model = load\_model("my\_saved\_model.h5")*

* Activity: 06-Ins\_Saving\_Models, 07-Stu\_Smartphone\_SavingModels

Concept: **K-means** is a method of cluster analysis that uses an unsupervised neural network to cluster data into the 'K' number of groups specified in the creation of the model. To create and fit a k-means model into 3 clusters of data you could use the following code:

*from sklearn.cluster import KMeans*

*kmeans = KMeans(n\_clusters=4)*

*kmeans.fit(data)*

To predict the clusters with a fitted k-means model you can use the following code:

*predicted\_clusters = kmeans.predict(data)*

* Activity: 08-Ins\_Kmeans, 09-Stu\_Kmeans