Practical-2 FNN explaination

---

1. Importing Libraries
```python
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
%matplotlib inline
```

- **TensorFlow and Keras**: Used for building and training the neural network.
- **Pandas & Numpy**: For data manipulation and handling.
- **Matplotlib**: For visualizing data, accuracy, and loss plots.
- **%matplotlib inline**: Ensures plots appear inline in Jupyter.

---

2. Loading the Dataset
```python
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

- **MNIST dataset**: Automatically loaded by TensorFlow, split into `x_train`, `y_train` (training data) and `x_test`, `y_test` (testing data).

---

3. Checking Dataset Size
```python
len(x_train)   # Output: 60000
len(x_test)    # Output: 10000
```

- Verifies that there are 60,000 training samples and 10,000 test samples in the MNIST dataset.

---

4. Checking Image Dimensions and Content
```python
x_train.shape
```

```
x_train[0]   # Display the first image matrix
```

- **Image Shape**: Each image is 28x28 pixels, totaling 784 pixels per image.
- **First Image Data**: Shows raw pixel values (0 to 255) representing grayscale intensity.

---

5. Visualizing the First Image
```python
plt.matshow(x_train[0])
plt.imshow(-x_train[0], cmap="gray")
```

- **Visualization**: Displays the first image, showing how it would look in grayscale.

---

6. Normalizing Data
```python
x_train, x_test = x_train / 255, x_test / 255
```

- **Normalization**: Scales pixel values to a range between 0 and 1, helping the neural network process data efficiently.

---

7. Building the Model
```python
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation="relu"),
    keras.layers.Dense(10, activation="softmax")
])
model.summary()
```

- **Sequential Model**: Defines a linear stack of layers.
- **Flatten Layer**: Reshapes each 28x28 image into a 1D array of 784 values.
- **Dense Layer (128 units, ReLU)**: First hidden layer with 128 neurons using ReLU activation.
- **Output Layer (10 units, Softmax)**: Final layer with 10 neurons (one per class), using Softmax to produce probabilities for each digit.

---

8. Compiling the Model
```python
model.compile(optimizer="sgd", loss="sparse_categorical_crossentropy",
metrics=['accuracy'])
```

- **Optimizer**: Stochastic Gradient Descent (SGD) for weight updates.
- **Loss**: `sparse_categorical_crossentropy` for multi-class classification.
- **Metrics**: Tracks accuracy during training.

---

9. Training the Model
```python
history = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=10)
```

- **Training**: Runs for 10 epochs, with both training and validation data (test data) evaluated at each epoch.
- **History Object**: Stores accuracy and loss metrics for each epoch.

---

10. Evaluating the Model
```python
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Loss=%.3f" %test_loss)
print("Accuracy=%.3f" %test_acc)
```

- **Evaluation**: Calculates final loss and accuracy on test data to see how well the model performs on unseen data.

---

11. Visualizing a Random Image from Test Set
```python
n = random.randint(0,9999)
plt.imshow(x_test[n])
plt.show()
```

- **Random Test Image**: Selects a random test image and displays it for visualization.

---

12. Predicting a Handwritten Digit
```python
predicted_value = model.predict(x_test)
print("Handwritten number in the image is= %d"
%np.argmax(predicted_value[n]))
```

- **Prediction**: Uses the model to predict the digit in the randomly selected test image.
- **Argmax**: Finds the digit with the highest predicted probability.

---

13. Plotting Model Accuracy
```python
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

- **Accuracy Plot**: Shows training and validation accuracy over epochs to visualize model performance.

---

14. Plotting Model Loss
```python
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

- **Loss Plot**: Plots training and validation loss over epochs, useful for analyzing model learning and overfitting.

---

15. Combined Plot for Training Loss and Accuracy
```python
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Training Loss and accuracy')
plt.ylabel('accuracy/Loss')
plt.xlabel('epoch')
plt.legend(['accuracy', 'val_accuracy','loss','val_loss'])
plt.show()
```

- **Combined Plot**: Displays accuracy and loss trends for both training and validation data in a single plot, providing an overview of model learning.

Questions answers
————————————————————————————————————

Here are important questions for Assignment 2, which covers implementing Feedforward Neural Networks with Keras and TensorFlow:

Important Questions and Answers

1. **What is a Feedforward Neural Network (FNN)?**
   - A Feedforward Neural Network is a type of neural network where connections between nodes do not form a cycle, allowing information to move in one direction—from input to output.

2. **How does a Feedforward Neural Network work?**
   - Data flows through input layers, passes through hidden layers where computations occur, and finally reaches the output layer for predictions.

3. **List three real-world applications of Feedforward Neural Networks.**
   - Image classification, sentiment analysis, and stock price prediction.

4. **What are the key components of a Feedforward Neural Network?**
   - **Input Layer**: Accepts input data.
   - **Hidden Layers**: Performs computations on the data.
   - **Output Layer**: Provides the final prediction or classification.

5. **What is a cost function in a Feedforward Neural Network?**
   - It’s a function used to evaluate the error of a network's prediction, guiding the optimization process to minimize this error.

6. **Explain the mean square error (MSE) cost function.**
   - MSE measures the average of the squares of errors (the difference between actual and predicted values), commonly used in regression tasks.

7. **What is a loss function in neural networks?**
   - A loss function calculates the difference between predicted and actual values; it’s minimized during training to improve model accuracy.

8. **Differentiate between the Sigmoid and Softmax activation functions.**
   - **Sigmoid**: Used in binary classification, outputs probabilities between 0 and 1.
   - **Softmax**: Used in multi-class classification, outputs probabilities that sum up to 1.

9. **What is the purpose of flattening a dataset?**
   - Flattening converts multi-dimensional data into a one-dimensional array to prepare it for input into a dense layer in a neural network.

10. **Describe the MNIST and CIFAR-10 datasets.**
    - **MNIST**: A dataset of handwritten digits (0-9) with 60,000 training and 10,000 test images.
    - **CIFAR-10**: A dataset of 60,000 32x32 color images across 10 categories.

11. **What is the role of an optimizer in model training?**
    - An optimizer updates model weights to minimize the loss function, improving accuracy. Common optimizers include SGD and Adam.

12. **What are epochs in neural network training?**
    - An epoch is one complete pass of the training dataset through the network.

13. **Explain the purpose of LabelBinarizer and classification_report in sklearn.**
    - **LabelBinarizer**: Converts categorical labels to binary form for compatibility with neural networks.
    - **classification_report**: Provides a summary of model performance metrics such as precision, recall, and F1-score.

14. **What is the significance of the fit() function in model training?**
    - The `fit()` function trains the model on the data by iterating over the data and updating weights based on the loss function.