Keras - Basics

 keras center third  footer_logo

Goal

- Use the keras library to train a neural network

Program

- Keras overview
- Steps for training a Keras model (Sequential API)
- A basic Keras example for solving XOR problem

Keras overview

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Steps for training a Keras model (Sequential API)

- Step 1: Create a model
- Step 2: Define a model
- Step 3: Compile for training
- step 4: Train the model
- Step 5: Evaluate the model

Step 0 - Generate some data:

In [2]:
```
from tensorflow import keras
import numpy as np

x_train = np.random.random((1000, 20))
y_train = keras.utils.to_categorical(
    np.random.randint(10, size=(1000, 1)), num_classes=10
)
x_test = np.random.random((100, 20))
y_test = keras.utils.to_categorical(
    np.random.randint(10, size=(100, 1)), num_classes=10
)
```

In [ ]:
```
x_train[100]
```

In [ ]:
```
y_train[100]
```

Step 1 - Create a model

Sequential API:

In [ ]:
```
from tensorflow.keras.models import Sequential

model = Sequential()
```

`model` is the container for your network architecture.

Step 2 - Define the model

Add three layers:

In [ ]:
```
from tensorflow.keras.layers import Dense, Dropout

#input_layer_neurons=20, hidden_layer_neurons=64
model.add(Dense(units=64, activation="relu", input_dim=20))
model.add(Dropout(rate=0.5))
model.add(Dense(units=10, activation="softmax"))
```

Layers can take parameters like activation functions, layer size, input size, etc.

Step 3 - Compile for training

Compile and pass training settings:

In [ ]:
```
model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
```

For instance:

- loss to minimize
- optimization algorithms
- metrics to report

Step 4 - Train the model

Fit the model by passing it the training data:

In [ ]:
```
model.fit(x_train, y_train, epochs=20, batch_size=128)
```

Step 5 - Evaluate the model

Supply the test data and get the model score:

In [ ]:
```
score = model.evaluate(x_test, y_test, batch_size=128, verbose=False)
score
```

A basic example - summary

```
model = Sequential()
model.add(Dense(64, activation="relu", input_dim=20))
model.add(Dropout(0.5))
model.add(Dense(10, activation="softmax"))

model.compile(loss="categorical_crossentropy",
              optimizer="adam", metrics=["accuracy"])

model.fit(x_train, y_train, epochs=20, batch_size=128)
score = model.evaluate(x_test, y_test, batch_size=128)
```

XOR Keras MLP example

In [ ]:

```
# Step 0. Encode your data
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]]) # A and B column
y = np.array([[0], [1], [1], [0]]) # A XOR B column

print('X is: \n', X)
print('y is: \n', y)
```

In [ ]:

```
seed= np.random.seed(0)

from tensorflow.keras.optimizers import SGD
opt = SGD(learning_rate=0.1)

from tensorflow.keras.initializers import RandomUniform
weights = RandomUniform(minval=0.0, maxval=1.0, seed=seed)


model = Sequential()
model.add(Dense(3, activation="sigmoid", input_dim=2, use_bias=False, kernel_initializer=weights))
model.add(Dense(1, activation="sigmoid", use_bias=False, kernel_initializer=weights))


model.compile(loss="mse", optimizer=opt)

model.fit(X, y, epochs=10000, batch_size=1, verbose=0)

print(model.predict(X))
```

In [ ]:

```
hidden_weights= model.layers[0].get_weights()
hidden_weights
```

In [ ]:

```
output_weights= model.layers[1].get_weights()
output_weights
```

Summary

In this notebook we covered,

- Keras overview
- Steps for training a Keras model (Sequential API)

- A basic Keras example for solving XOR problem

Exercise Keras

Exercise: Keras basics