PDPM
IIITDM
Jabalpur

Winter FDP on Deep Learning & Applications
09-13 Dec. 2019

Introduction to Keras

Keras

- Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
- Here we are going to use Keras on the top of TensorFlow.

- **Keras Model creation-** The core data structure of Keras is a model, a way to organize layers. The simplest type of model is the Sequential model, a linear stack of layers.

```
from keras.models import Sequential

model = Sequential()
```

Keras cont...

- Adding layers to model - Stacking layers is as easy as .add()

```
from keras.layers import Dense

model.add(Dense(units=64, activation='relu', input_dim=100))

model.add(Dense(units=10, activation='softmax'))
```

- Model Compilation - Configure model learning process with .compile()

```
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
```

Keras cont...

- Model Training - To iterate on training data in batches.

  model.fit(x_train, y_train, epochs=5, batch_size=32)

- Evaluate model performance -

  model.evaluate(x_test, y_test, batch_size=128)

- Predict output on new data -

  classes = model.predict(x_test, batch_size=128)

Practice Problem

We are going to use House value prediction dataset https://www.kaggle.com/c/zillow-prize-1/data and our objective is to determine whether the house price is above or below the median.

Practice Problem

- Reading the data file using panda library -

```
import pandas as pd

df = pd.read_csv('housepricedata.csv')

df.head()
```

- Data Separation into input and target –

```
dataset= df.values

X = dataset[:,0:10]

Y = dataset[:,10]
```

Practice Problem

- Data Normalization in the range of 0 to 1 -

```
from sklearn import preprocessing

min_max_scaler = preprocessing.MinMaxScaler()

X_scale = min_max_scaler.fit_transform(X)

X_scale
```

Practice Problem

- Splitting the data into training, testing and validation - Here we are splitting the data 80% for training, 10 % for testing and 10% for validation.

```
from sklearn.model_selection import train_test_split

X_train, X_val_and_test, Y_train, Y_val_and_test = train_test_split(X_scale, Y, test_size=0.2)

X_val, X_test, Y_val, Y_test = train_test_split(X_val_and_test, Y_val_and_test, test_size=0.5)
```

Practice Problem

- Model Building with the help of Dense Layers -

```
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(10,)))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

Practice Problem

- Model Compilation with SGD optimizer to evaluate the performance with loss and accuracy metrics.

```
model.compile(optimizer='sgd',
          loss='binary_crossentropy',
          metrics=['accuracy'])
```

Practice Problem

- Model Training in batch size of 32 for 30 iterations -

```
hist = model.fit(X_train, Y_train,
          batch_size=32, epochs=30,
          validation_data=(X_val, Y_val))
```

Practice Problem

- Model Testing to check the performance of trained model -

```
test_evaluate =model.evaluate(X_test, Y_test)

print("Testing Accuracy :"+ str(test_evaluate[1]*100))

print("Testing Loss :"+ str(test_evaluate[0]))
```

Practice Problem

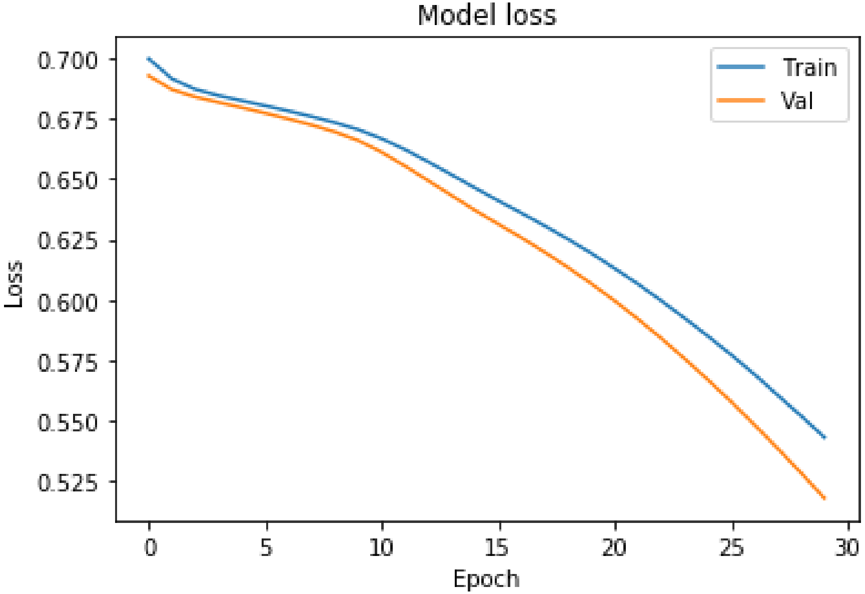
- Saving the weights and model for future use -

  ```
  from keras.models import load_model

  model.save('my_model.h5')

  model.save_weights('my_model_weights.h5')
  ```

Practice Problem

- Plot the training and validation of Loss -

```
import matplotlib.pyplot as plt

plt.plot(hist.history['loss'])

plt.plot(hist.history['val_loss'])

plt.title('Model loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Val'], loc='upper right')

plt.show()
```



Practice Problem

- Plotting the accuracy graph -

plt.plot(hist.history['acc'])

plt.plot(hist.history['val_acc'])

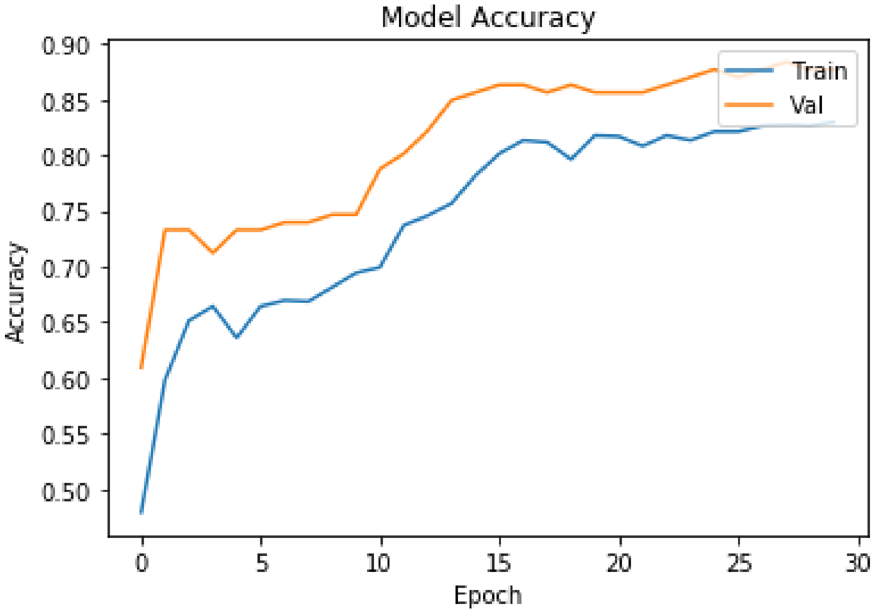
plt.title('Model Accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Val'], loc='upper right')

plt.show()



Exercise

You have to load the weights of the saved model and then add one more layer Dense layer with 8 neuron then check the performance of the model.

References

- https://keras.io/

Thank You