4. Deep Learning – Multilayer Perceptrons Models in Keras

Contents

4. Deep Learning – Multilayer Perceptrons Models in Keras

Steps to create MLP model in keras

- ✓ Model creation (Neural Network Model) by using Keras
- ✓ Compile the model
- ✓ Model training
- ✓ Model prediction

Behind the steps

- ✓ Model creation
  - Model Layers
    - Weights initialization
    - Activation function

- ✓ Compile the model.
  - Optimization
  - loss function
  - metrics

- ✓ Model training
  - Epochs
  - Batch size

- ✓ Model predictions

1. Create model (Neural Network Model) by using Keras

- ✓ Sequential is a predefined class in keras package
- ✓ By using this we can create a model.

```
from tensorflow.keras.models import Sequential

model = Sequential()
```

1.1. Model layers

- ✓ Once model created then we need to add layers to model.
  - Creating layers means, create object to Dense class
- ✓ We need to specify number of features to input layer.
  - Below example, 8 means its features

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


model = Sequential()

layers_hidden_input = Dense(16, input_shape = (8, ))

model.add(layers_hidden_input)
```

1.2. Important properties to layers

- ✓ Main properties,
  - Weight initialization.
  - Activation functions.

1.2.1. Weight Initialization

- ✓ By using kernel_initializer
  - random_uniform, random_normal, zero

Example

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


model = Sequential()

layers_hidden_input = Dense(16, input_shape = (8, ),
kernel_initializer = "random_uniform")

model.add(layers_hidden_input)
```

1.2.2. Activation Function

- ✓ We need to use activation functions like,
  - softmax
  - rectified linear (ReLU),
  - tanh,
  - sigmoid

Example

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


model = Sequential()

layers_hidden_input = Dense(16, input_shape = (8, ),
kernel_initializer = "random_uniform", activation = "relu")

model.add(layers_hidden_input)
```

2. Model Compilation

- ✓ Once model created then we need to compile the model (Neural Network Model).
- ✓ During this step TensorFlow converts the model into a graph so the training can be carried out efficiently.
- ✓ We can compile by calling compile() method
- ✓ Important attributes in compile() method,
  - Model optimizer
  - Loss function
  - Metrics

Example

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


model = Sequential()

layers_hidden_input = Dense(16, input_shape = (8, ),
kernel_initializer = "random_uniform", activation = "relu")

model.add(layers_hidden_input)

model.compile(optimizer = ..., loss = ..., metrics = ...)
```

3. Model Training

- ✓ Once model created and compiled then next step is to train the model.
- ✓ We can train the model by using fit(….) method

Example

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


model = Sequential()

layers_hidden_input = Dense(16, input_shape = (8, ),
kernel_initializer = "random_uniform", activation = "relu")

model.add(layers_hidden_input)

model.compile(optimizer = ..., loss = ..., metrics = ...)

model.fit(X, y, epochs = ..., batch_size =...)
```

4. Model Prediction

- ✓ We model training is done then we can predict with new data.
  - model.predict(X): To generate network output for the input data
  - model.evaluate(X, y): To calculate the loss values for the input data

Example

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


model = Sequential()

layers_hidden_input = Dense(16, input_shape = (8, ),
kernel_initializer = "random_uniform", activation = "relu")

model.add(layers_hidden_input)

model.compile(optimizer = ..., loss = ..., metrics = ...)

model.fit(X, y, epochs = ..., batch_size =...)

model.predict(X)
```