Keras - Model Compilation, Evaluation, Prediction

Tushar B. Kute,
http://tusharkute.com

Compilation

- The compilation is the final step in creating a model.
- Once the compilation is done, we can move on to training phase.

Loss

- In machine learning, Loss function is used to find error or deviation in the learning process. Keras requires loss function during model compilation process.
- Keras provides quite a few loss function in the losses module and they are as follows –

  mean_squared_error

  mean_absolute_error

  mean_absolute_percentage_error

  mean_squared_logarithmic_error

  squared_hinge

  hinge

  categorical_hinge

Loss

- logcosh
- huber_loss
- categorical_crossentropy
- sparse_categorical_crossentropy
- binary_crossentropy
- kullback_leibler_divergence
- poisson
- cosine_proximity
- is_categorical_crossentropy

Loss

- All above loss function accepts two arguments –
  - y_true – true labels as tensors
  - y_pred – prediction with same shape as y_true

Optimizers

- In machine learning, Optimization is an important process which optimize the input weights by comparing the prediction and the loss function.
- Keras provides quite a few optimizer as a module, optimizers and they are as follows:
- SGD – Stochastic gradient descent optimizer.

  keras.optimizers.SGD(learning_rate = 0.01, momentum = 0.0, nesterov = False)

Optimizers

- RMSprop – RMSProp optimizer.
  - keras.optimizers.RMSprop(learning_rate = 0.001, rho = 0.9)
- Adagrad – Adagrad optimizer.
  - keras.optimizers.Adagrad(learning_rate = 0.01)
- Adadelta – Adadelta optimizer.
  - keras.optimizers.Adadelta(learning_rate = 1.0, rho = 0.95)

Optimizers

- Adam – Adam optimizer.
  - keras.optimizers.Adam(learning_rate = 0.001, beta_1 = 0.9, beta_2 = 0.999, amsgrad = False)
- Adamax – Adamax optimizer from Adam.
  - keras.optimizers.Adamax(learning_rate = 0.002, beta_1 = 0.9, beta_2 = 0.999)
- Nadam – Nesterov Adam optimizer.
  - keras.optimizers.Nadam(learning_rate = 0.002, beta_1 = 0.9, beta_2 = 0.999)

Metrics

- In machine learning, Metrics is used to evaluate the performance of your model. It is similar to loss function, but not used in training process. Keras provides quite a few metrics as a module, metrics and they are as follows

  accuracy

  binary_accuracy

  categorical_accuracy

  sparse_categorical_accuracy

  top_k_categorical_accuracy

  sparse_top_k_categorical_accuracy

  cosine_proximity

  clone_metric

Metrics

- Similar to loss function, metrics also accepts below two arguments –

  y_true – true labels as tensors

  y_pred – prediction with same shape as y_true

Compile the Model

- Keras model provides a method, compile() to compile the model. The argument and default value of the compile() method is as follows

```
compile(
  optimizer,
  loss = None,
  metrics = None,
  loss_weights = None,
  sample_weight_mode = None,
  weighted_metrics = None,
  target_tensors = None
)
```

Compile the Model

- The important arguments are as follows –

    loss function

    Optimizer

    metrics

- A sample code to compile the mode is as follows –

```
from keras import losses
from keras import optimizers
from keras import metrics
model.compile(loss = 'mean_squared_error',
   optimizer = 'sgd', metrics = [metrics.categorical_accuracy])
```

Model Training

- Models are trained by NumPy arrays using fit(). The main purpose of this fit function is used to evaluate your model on training.
- This can be also used for graphing model performance. It has the following syntax –

  model.fit(X, y, epochs = , batch_size =)
- Here,
  - X, y – It is a tuple to evaluate your data.
  - epochs – no of times the model is needed to be evaluated during training.
  - batch_size – training instances.

Model Evaluation

- Evaluation is a process during development of the model to check whether the model is best fit for the given problem and corresponding data.
- Keras model provides a function, evaluate which does the evaluation of the model.
- It has three main arguments,

  Test data

  Test data label

  verbose - true or false

Model Evaluation

- The evaluate function is used to evaluate the test data.

```
score = model.evaluate(x_test, y_test, verbose = 0)

print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Model Prediction

- Prediction is the final step and our expected outcome of the model generation. Keras provides a method, predict to get the prediction of the trained model. The signature of the predict method is as follows,

```
predict(
   x,
   batch_size = None,    verbose = 0,
   steps = None,    callbacks = None,
   max_queue_size = 10,    workers = 1,
   use_multiprocessing = False
)
```

Example:

- Classification Example

Thank you

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com