[model paradigm]

Using estimator for tensorflow model training, evaluation, and prediction has following parts.

1. define input\_function
   * dataset ETL
2. define model\_function
3. define feature\_column
4. build estimator
5. train/evaluate/predict with estimator

[define input\_function]

The input function is just a function that return tensorflow “dataset”. About how to create “dataset” tutorial is covered in the “dataset” directory.

* example

def input\_fn(<features>, <labels>, <batch\_size>):

dataset = tf.data.Dataset.from\_tensor\_slices((<features, labels>))

dataset = dataset.shuffle(1000).repeat().batch(<batch\_size>)

return dataset

Notice:

the input function basically should not have argument. so don’t forget to use lambda function to transform the input funcition into non-argument version.

* example:

new\_function = lambda: input\_fn(<features>, <labels>, <batch\_size>)

[define model\_function]

the model function parts is covered in “model\_fn.docx” in the same directory.

[build estimator]

<estimator> = tf.estimator.Estimator(model\_fn,

params,

model\_dir)

* **model\_fn**: {function} the funtion defines model and return tf.estimator.EstimatorSpec.
* **params**: {dict} the argument that pass into <model\_fn>. It parametrizes the model function
* **model\_dir**: {str} the directory name of the saving model.

[[train model]](https://www.tensorflow.org/api_docs/python/tf/estimator/Estimator#train)

<estimator>.train(input\_fn,

steps=None)

* **input\_fn**: {function} the funtion defines input pipeline and return tf.data.Dataset.
* **steps**: {int} the training steps number. At each step input\_fn generates a batch of data for training.  
   If None, it will keep calling the input\_fn until it raise tf.errors.OutOfRange.

Notice:

1. if the model\_dir already has checkpoint, it will load it and continuous the training process

[[evaluate model]](https://www.tensorflow.org/api_docs/python/tf/estimator/Estimator#evaluate)

<estimator>.evaluate(input\_fn,

steps)

* **input\_fn**:{function} the funtion defines input pipeline and return tf.data.Dataset.
* **steps**: {int} the evaluation step. At each step input\_fn generates a batch of data for evaluating.  
   If None, it will keep calling the input\_fn until it raise tf.errors.OutOfRange.

Notice:

1. The evaluation result will only be one dot in Tensorboard no matter how many steps you set.  
   (The result is the average accuracy of all steps)

[[predict]](https://www.tensorflow.org/api_docs/python/tf/estimator/Estimator#predict)

<estimator>.predict(input\_fn,

predict\_keys=None)

* **input\_fn**: {function} the funtion defines input pipeline and return tf.data.Dataset.
* **predict\_keys**: {list(str)} the list of prediction key that need to output. If None, it will output all dictionary. The dictionary is actually the <dict> which we define in model function.  
  (if mode == tf.estimator.ModeKeys.PREDICT:

return tf.estimator.EstimatorSpec(mode, <dict>))