# Tensorflow中的Estimator模块分析

## 1. 简介

Estimator是Tensorflow的高级API，推出的时间不到1年，这个模块有利有弊。

优势：目前TensorFlow正在大力推Estimator；作为高级的API，它的代码架构清晰，sessions、graphs、loops、logging等基本函数已封装，使用是只需要关心导入数据、数据集的处理和构建模型即可，使用较为方便。

弊端：文档和教程仍然偏少，且官方只了封装很少的学习器，真正使用的人不多；此类高级API的封装会使得灵活性降低，不一定适合任务（根据不同问题使用合适的粒度）。

使用后的看法：一般会结合其他的高级API（例如tf.dataset）；代码量很少，能节省很多定义变量，设置网络结构的时间，构建结构很清晰、简洁；环境变量的设置十分方便，直接就可以使用TPU、分布式、eager等。总的来说，Estimator还很不完善，可以尝试着跑跑dome，但不推荐在实际的项目中使用。

## 2. 相关源码

### 2.1 源码学习纲要

Estimator本身的源码路径是'tensorflow/python/estimator'，即tf.estimator。另外，在'tensorflow/contrib/estimator'也有源码但已经弃用。

Estimator的官方实例在'/tensorflow/model/official/mnist/'下，这其中包括了TPU版本的estimator和eager版本的estimator。

### 2.2 Estimator的主要源码

主要的源码在'tensorflow/python/estimator'文件夹下，下面逐文件分析：

（1）estimator.py和estimator\_lib.py：Estimator模块最为核心的文件，封装了estimator类。

（2）exporter.py：用于保存train的模型，主要封装了Exporter类。

（3）model\_fn.py：用于model\_fn（网络结构）的构建，封装了ModeKeys类和EstimatorSpec类。

（4）replicate\_model\_fn.py：用于将model\_fn复写到GPU上。

（5）run\_config.py：estimator的运行环境设置（有host和cluster的概念），封装了RunConfig类。

（6）training.py：封装了有关train和evaluate过程的类和函数。

（7）canned文件夹：“罐头”，里面是官方提供的评估器，包括boosted\_trees 、LinearClassifier等。

（8）export 文件夹：即estimator.export，运行estimator的输出，包括生成的模型、log信息等等。

（9）inputs文件夹：创建了input\_fns，即控制和处理输入给estimator模型的数据，数据有'numpy\_input\_fn'和'pandas\_input\_fn'两种类型可以选择。

## 3. 使用流程

### 3.1 导入数据集

导入数据集是不会调用Estimator模块的。一般是使用tf.data和tf.data.Dataset进行导入数据集。

例如，一般会创建两个函数，一个是导入训练集，另一个是导入测试集，格式如下：

|  |
| --- |
| def input\_fn(dataset):  ... # manipulate dataset, extracting feature names and the label  return feature\_dict, label |

### 3.2 实例化Estimator

Estimator可以翻译为“评估器”，我们可以直接使用官方提供的Estimator，也可以使创建一个自定义的Estimator。

官方文档见“https://tensorflow.google.cn/get\_started/custom\_estimators”。

（1）官方提供的Estimator包括LogisticRegressor、LinearClassifier、LinearRegressor、DNNClassifier等，使用时的格式如下：

|  |
| --- |
| # 对官方预创建的、名为LinearClassifier的Estimator进行示例实例化  # Instantiate an estimator, passing the feature columns.  estimator = tf.estimator.Estimator.LinearClassifier(  feature\_columns=[population, crime\_rate, median\_education],  ) |

（2）如果官方没有提供合适的Estimator，则需要自己自定义一个Estimator，具体需要两步操作。

第一步，先定义网络结构，生成tf.estimator.EstimatorSpec类。更详细地讲，自定义的该网络结构必须含有下列三个部分。

一个输入层：一般是使用tf.feature\_column.input\_layer，以将特征字典和 feature\_columns 转换为模型的输入。

一个或多个隐藏层：一般是使用tf.layers或tf.nn来定义。

一个输出层：输出层的定义一般会根据不同的阶段改变。例如，当train时，可以使用tf.losses.sparse\_softmax\_cross\_entropy，且需要定义optimizer（一般是使用tf.train）；而当predict时，可以使用tf.nn.softmax。

|  |
| --- |
| logits = # ...  loss = tf.losses.softmax\_cross\_entropy(onehot\_labels=labels, logits=logits)  optimizer = tpu\_optimizer.CrossShardOptimizer(  tf.train.GradientDescentOptimizer(learning\_rate=FLAGS.learning\_rate))  train\_op = optimizer.minimize(loss, global\_step=tf.train.get\_global\_step())  return tf.estimator.EstimatorSpec(mode=mode, loss=loss, train\_op=train\_op) |

第二步，生成真正的tf.estimator.Estimator类。所需要的参数包括：model\_fn表示网络结构，是tf.estimator.EstimatorSpec()类；model\_dir表示模型的存储路径；params表示参数，包括data\_format、multi\_gpu等。

|  |
| --- |
| tf.estimator.Estimator(model\_fn=model\_fn, model\_dir="mnist\_model\_cnn", params=params) |

### 3.3 训练、评估或推理

调用训练、评估或推理方法，而所对应的函数分别是train()、evaluate()和predict()。

以evaluate()为例：

|  |
| --- |
| estimator.evaluate(input\_fn=self.\_eval\_spec.input\_fn,  steps=self.\_eval\_spec.steps,  name=self.\_eval\_spec.name,  checkpoint\_path=latest\_ckpt\_path,  hooks=self.\_eval\_spec.hooks) |

最后，可以使用tf.estimator..export\_savedmodel()来保存模型。

## 4. 实例分析

### 4.1 定义网络结构：

|  |
| --- |
| def model\_fn(features, labels, mode, params):  """The model\_fn argument for creating an Estimator."""  model = create\_model(params['data\_format'])  image = features  if isinstance(image, dict):  image = features['image']  if mode == tf.estimator.ModeKeys.PREDICT:  logits = model(image, training=False)  predictions = {  'classes': tf.argmax(logits, axis=1),  'probabilities': tf.nn.softmax(logits),  }  return tf.estimator.EstimatorSpec(  mode=tf.estimator.ModeKeys.PREDICT,  predictions=predictions,  export\_outputs={  'classify': tf.estimator.export.PredictOutput(predictions)  })  if mode == tf.estimator.ModeKeys.TRAIN:  optimizer = tf.train.AdamOptimizer(learning\_rate=LEARNING\_RATE)  # If we are running multi-GPU, we need to wrap the optimizer.  if params.get('multi\_gpu'):  optimizer = tf.contrib.estimator.TowerOptimizer(optimizer)  logits = model(image, training=True)  loss = tf.losses.sparse\_softmax\_cross\_entropy(labels=labels, logits=logits)  accuracy = tf.metrics.accuracy(  labels=labels, predictions=tf.argmax(logits, axis=1))  # Name tensors to be logged with LoggingTensorHook.  tf.identity(LEARNING\_RATE, 'learning\_rate')  tf.identity(loss, 'cross\_entropy')  tf.identity(accuracy[1], name='train\_accuracy')  # Save accuracy scalar to Tensorboard output.  tf.summary.scalar('train\_accuracy', accuracy[1])  return tf.estimator.EstimatorSpec(  mode=tf.estimator.ModeKeys.TRAIN,  loss=loss,  train\_op=optimizer.minimize(loss, tf.train.get\_or\_create\_global\_step()))  if mode == tf.estimator.ModeKeys.EVAL:  logits = model(image, training=False)  loss = tf.losses.sparse\_softmax\_cross\_entropy(labels=labels, logits=logits)  return tf.estimator.EstimatorSpec(  mode=tf.estimator.ModeKeys.EVAL,  loss=loss,  eval\_metric\_ops={  'accuracy':  tf.metrics.accuracy(  labels=labels, predictions=tf.argmax(logits, axis=1)),  }) |

### 4.2 生成estimator

|  |
| --- |
| mnist\_classifier = tf.estimator.Estimator(  model\_fn=model\_fn,  model\_dir=flags.model\_dir,  params={  'data\_format': data\_format,  'multi\_gpu': flags.multi\_gpu  }) |

### 4.3 模型的训练和测试

|  |
| --- |
| # Train and evaluate model.  for \_ in range(flags.train\_epochs // flags.epochs\_between\_evals):  mnist\_classifier.train(input\_fn=train\_input\_fn)  eval\_results = mnist\_classifier.evaluate(input\_fn=eval\_input\_fn) |

### 4.4 保存模型

|  |
| --- |
| # Export the model  if flags.export\_dir is not None:  image = tf.placeholder(tf.float32, [None, 28, 28])  input\_fn = tf.estimator.export.build\_raw\_serving\_input\_receiver\_fn({  'image': image,  })  mnist\_classifier.export\_savedmodel(flags.export\_dir, input\_fn) |

更多的例子，可见https://zhuanlan.zhihu.com/p/33681224和https://blog.csdn.net/crazyman2010/article/details/78921086

## 参考资料

https://zhuanlan.zhihu.com/p/33681224

https://zhuanlan.zhihu.com/p/36976614

https://tensorflow.google.cn/programmers\_guide/estimators

https://zhuanlan.zhihu.com/p/29210791

https://zhuanlan.zhihu.com/p/28634935

https://github.com/tensorflow/models/tree/master/official/mnist