Estimator API

Quiz, 9 questions

9/9 points (100%)

✓ Congratulations! You passed!

Next Item

✓ 1/1 point

1. What are some of the key goals of the estimator API?

- ○ Create production-ready machine learning models using an API
- ○ Train on large datasets that do not fit in memory
- ○ Quickly monitor your training metrics in Tensorboard
- [✓ All of the above]

Correct

✓ 1/1 point

2. What is one of the largest benefits of the estimator API?

- ○ It automatically tunes your ML model hyperparameters for you
- ○ It requires you to specify which hardware you will run on for the best performance
- [✓ It abstracts away boilerplate code which saves you time]

Correct

✓ 1/1 point

3. What is the right way to call a linear regression model with tf.estimator?

- ○ tf.estimator.line_model
- ○ tf.estimator.regression
- [✓ tf.estimator.LinearRegressor]

Correct

- ○ tf.estimator.LinearClassifier

✓ 1/1 point

4. Inputs to the estimator model are in the form of:

- ○ BigQuery datasets
- ○ scalars
- ○ hyperparameters
- [✓ feature columns]

Correct

✓ 1/1 point

5. Numeric inputs can be passed to a linear regressor as-is, but categorical columns are often:

- ○ Not used, only numeric values can be passed
- ○ Uniformly distributed and aggregated first
- [✓ One-hot encoded]

Correct

- ○ Cleansed because of duplicate records

✓

6. What is the size of the training dataset (features + labels) in this example?

1 / 1 point

```
def train_input_fn():
   features = {"sq_footage": [ 1000,    2000,    3000,    1000,  2000,  3000],
               "type":       ["house", "house", "house", "apt", "apt", "apt"]}
                               # prices in thousands
   labels =                  [ 500,     1000,    1500,    700,   1300,  1900]
   return features, labels
```

( ) 6 rows, 2 columns

( ) 7 rows, 4 columns

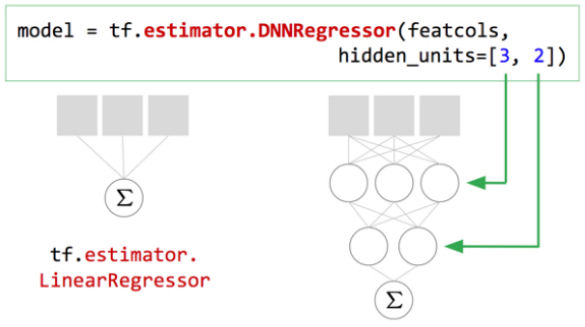
[✓ 6 rows, 3 columns]

Correct

( ) 6 rows, 4 columns

✓ 7. In this example, what extra parameters does the DNNRegressor take that the LinearRegressor doesn't?

1 / 1 point



```
model = tf.estimator.DNNRegressor(featcols,
                                  hidden_units=[3, 2])
```

[✓ hidden_units]

Correct

( ) featcols

( ) regression

( ) neurons

✓ 8. In what situation do you have to delete the model directory before starting training?

1 / 1 point

[✓ If you have changed the model structure from the previous time, for example, you used a DNNRegressor with [64,32] last time and now you are using [32, 16]]

Correct
The old checkpoints are no longer valid for your new model structure. So, you have to start afresh

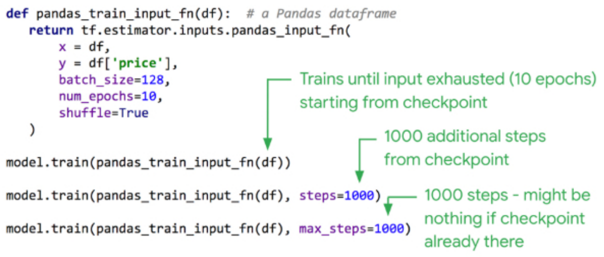
( ) If your model is not performing well enough and you need to train for more epochs or with additional examples

( ) If you want to automatically checkpoint from an earlier saved model

---

✓ 9. What is the difference between steps and max_steps?

1 / 1 point



```
def pandas_train_input_fn(df):  # a Pandas dataframe
   return tf.estimator.inputs.pandas_input_fn(
       x = df,
       y = df['price'],
       batch_size=128,
       num_epochs=10,
       shuffle=True
   )

model.train(pandas_train_input_fn(df))

model.train(pandas_train_input_fn(df), steps=1000)

model.train(pandas_train_input_fn(df), max_steps=1000)
```

Trains until input exhausted (10 epochs) starting from checkpoint
1000 additional steps from checkpoint
1000 steps - might be nothing if checkpoint already there

[✓ Steps means "train these many additional steps". max_steps means "train up to these many steps total, starting from how many ever steps have been completed so far"]

Correct

There is no difference

Steps means "train this many steps total". max_steps means "train these many additional steps"