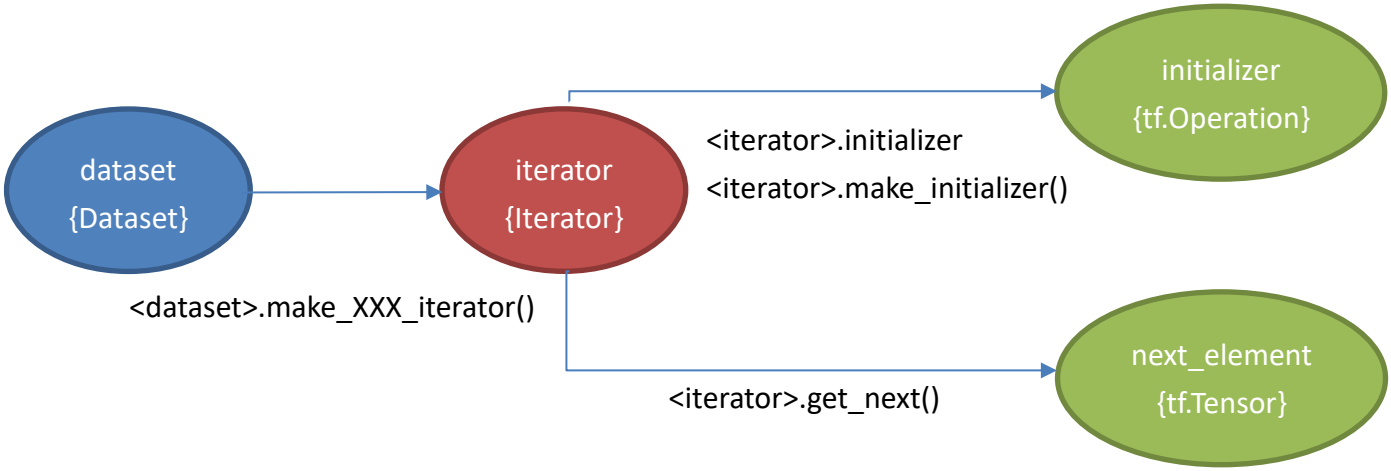
[concept]

Using feed_dict for every training step is too time consuming. The best practice for building tensorflow data input pipeline is using their built-in API, Dataset.




[build dataset]

- from numpy

dataset = tf.data.Dataset.from_tensor_slices(<ndarray>)

Notice:

1. The ndarray will be embedded as tf.constant and consuming graph memory. So this practice is not recommended for large dataset.
2. Usually we will pass features and labels together in this way:
   dataset = tf.data.Dataset.from_tensor_slices((<feature>, <label>))

- from tensor

[create dataset iterator]

1. the dataset will not be shown in tensorboard graph until <dataset>.make…Iterator() function been called.
2. the same dataset can have mulitiple independent Iterator and they don't affect each other.
3. There are generally four types of iterators: one-shot, initializable, reinitializable, feedable

- one-shot iterator

✓ The most common iterator when dataset is derived from local memory.
✓ Non-parametrized.
✓ Note: Currently, one-shot iterators are the only type that is easily usable with an Estimator.

<iterator> = <dataset>.make_one_shot_iterator()
<next_element> = <iterator>.get_next()

```
dataset = tf.data.Dataset.range(100)
iterator = dataset.make_one_shot_iterator()
next_element = iterator.get_next()

for i in range(100):
  value = sess.run(next_element)
  assert i == value
```

- initializable iterator

✓ The most common iterator when dataset is derived from tensor.
✓ It can be initialized and reinitialized by feed_dict argument.
✓ Notice that the iterator need to be initalized before being used.

<dataset>.make_initializable_iterator()
<next_element> = <iterator>.get_next()
<initialization_operation> = iterator.initializer

```
max_value = tf.placeholder(tf.int64, shape=[])
dataset = tf.data.Dataset.range(max_value)
iterator = dataset.make_initializable_iterator()
next_element = iterator.get_next()

# Initialize an iterator over a dataset with 10 elements.
sess.run(iterator.initializer, feed_dict={max_value: 10})
for i in range(10):
  value = sess.run(next_element)
  assert i == value

# Initialize the same iterator over a dataset with 100 elements.
sess.run(iterator.initializer, feed_dict={max_value: 100})
for i in range(100):
  value = sess.run(next_element)
  assert i == value
```

- reinitialzable iterator

✓ The iterator can change different source dataset
✓ The iterator is first defined by dataset's shape an type

- ✓ Before using it, we need to initialize its sources and, sure it can be reinitialized.

```
<iterator> = tf.data.Iterator.from_structure(<training_dataset>.output_types,
                                             <training_dataset>.output_shapes)

<next_element> = <iterator>.get_next()
<initializer_op1> = <iterator>.make_initializer(<dataset1>)
<initializer_op2> = <iterator>.make_initializer(<dataset2>)
```