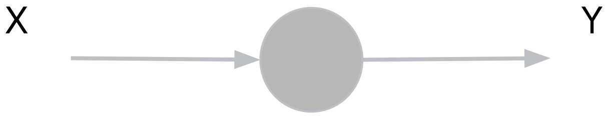
Understanding Neurons

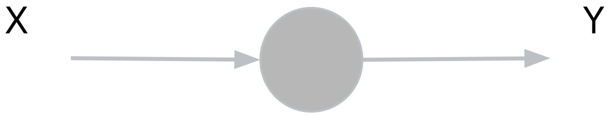
The Building Blocks of Deep Learning

Laurence Moroney, Google

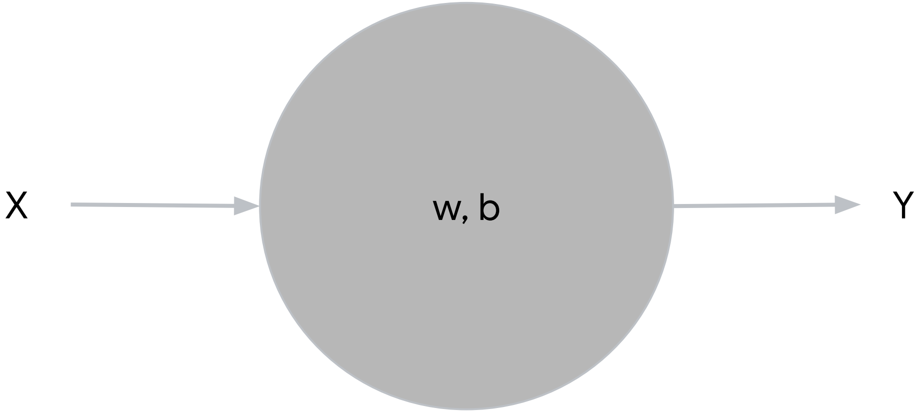
```python
model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

print(model.predict([10.0]))
```


X
Y

```python
model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

print(model.predict([10.0]))
```



$$y = f(x) = wx+b$$


X
w, b
Y

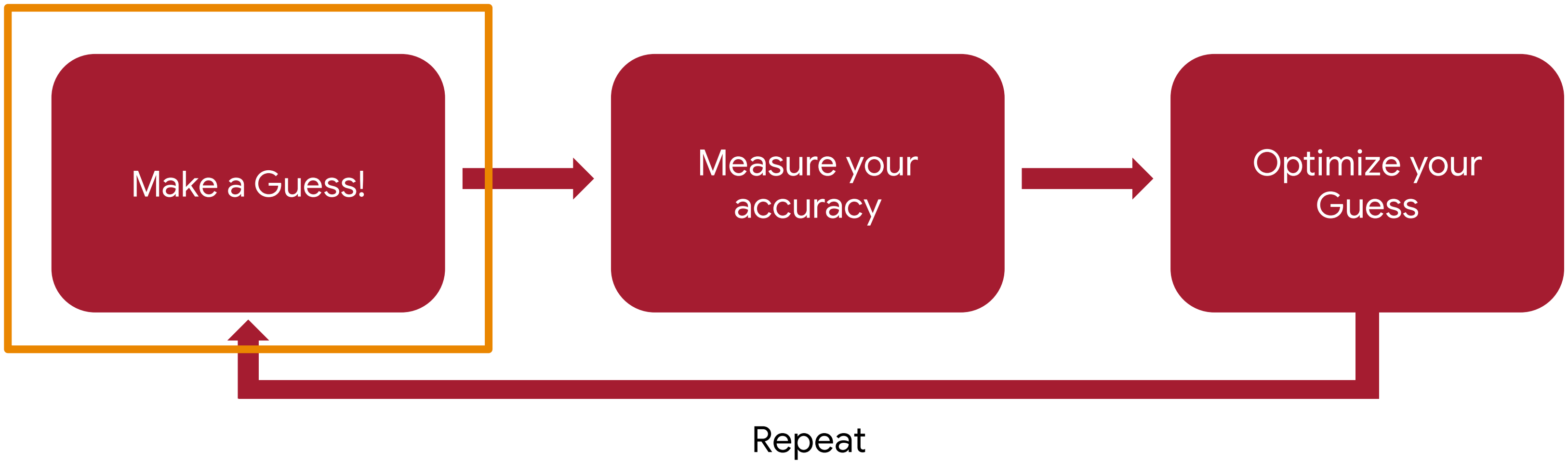
```python
class Model(object):
  def __init__(self):
    self.w = tf.Variable(10.0)
    self.b = tf.Variable(10.0)

  def __call__(self, x):
    return self.w * x + self.b
```

```python
class Model(object):
  def __init__(self):
    self.w = tf.Variable(10.0)
    self.b = tf.Variable(10.0)

  def __call__(self, x):
    return self.w * x + self.b
```

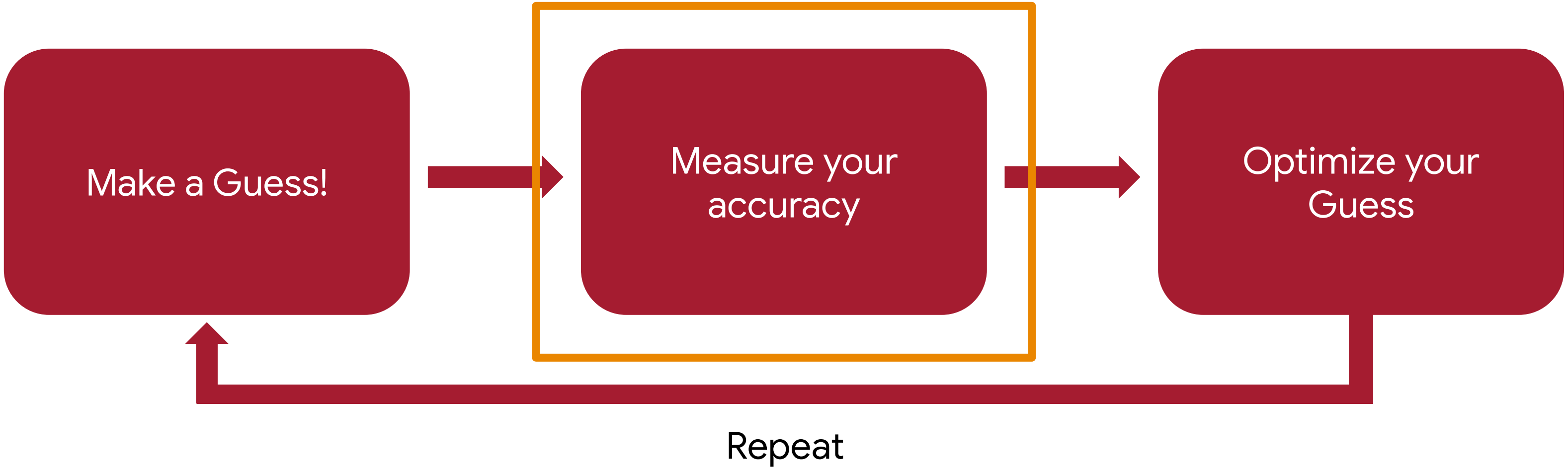
```
model = Model()
xs = [-1.0, 0.0, 1.0, 2.0, 3.0, 4.0]
ys = [-3.0, -1.0, 1.0, 3.0, 5.0, 7.0]
print(model(xs))
```

```
[ 0. 10. 20. 30. 40. 50.]
```

The Machine Learning Paradigm



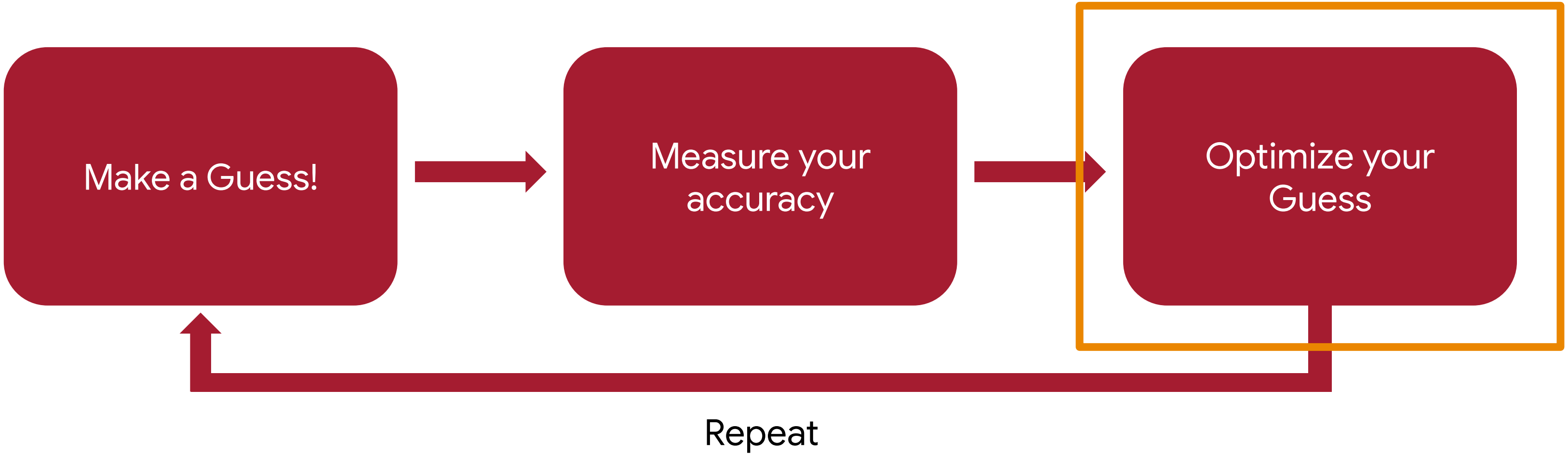
The Machine Learning Paradigm



```python
def loss(predicted_y, target_y):
  return tf.reduce_mean(tf.square(predicted_y - target_y))
```

```python
def train(model, xs, ys, learning_rate):
  with tf.GradientTape() as t:
    current_loss = loss(model(xs), ys)

  dw, db = t.gradient(current_loss, [model.w, model.b])
  model.w.assign_sub(learning_rate * dw)
  model.b.assign_sub(learning_rate * db)
  return current_loss
```

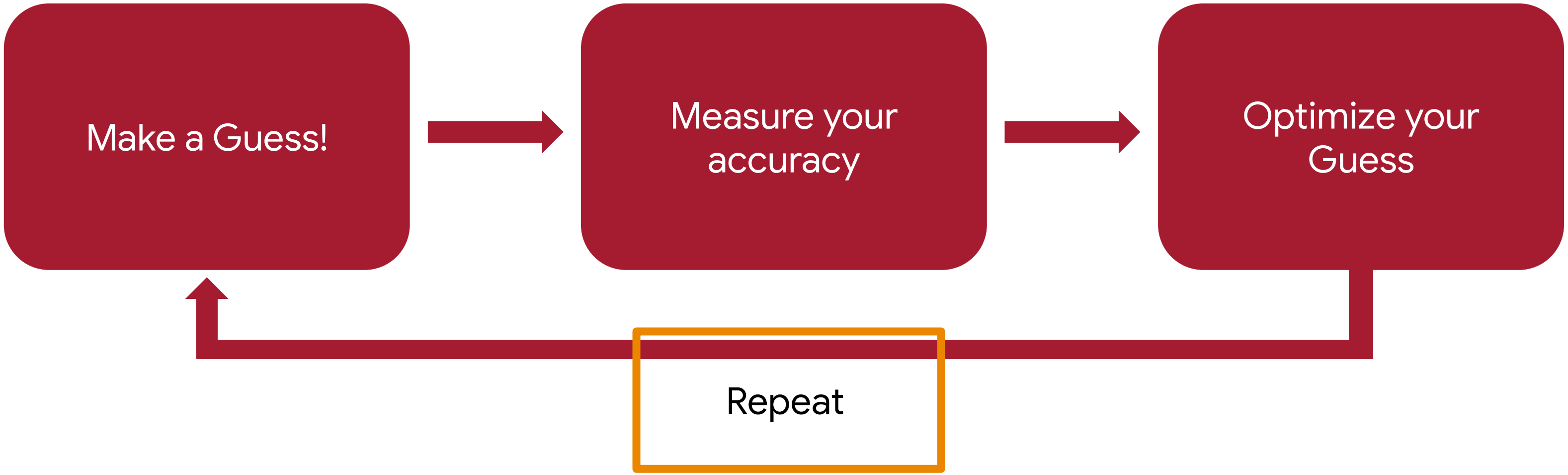
The Machine Learning Paradigm



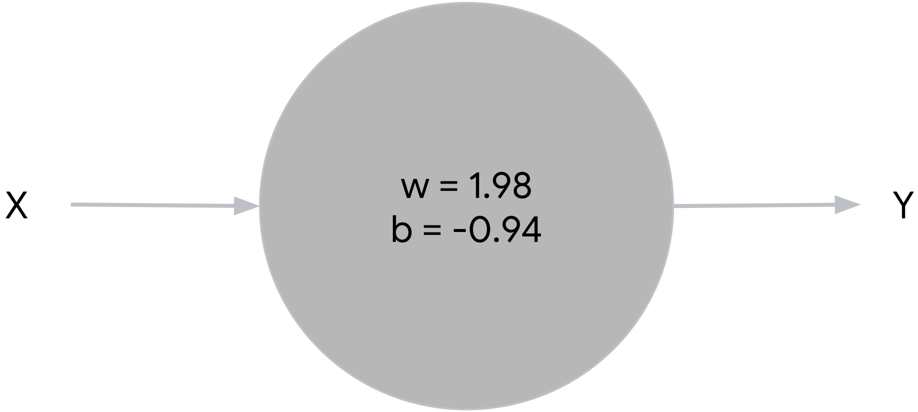
```python
def train(model, xs, ys, learning_rate):
  with tf.GradientTape() as t:
    current_loss = loss(model(xs), ys)

  dw, db = t.gradient(current_loss, [model.w, model.b])
  model.w.assign_sub(learning_rate * dw)
  model.b.assign_sub(learning_rate * db)
  return current_loss
```

```
def train(model, xs, ys, learning_rate):
  with tf.GradientTape() as t:
    current_loss = loss(model(xs), ys)

  dw, db = t.gradient(current_loss, [model.w, model.b])
  model.w.assign_sub(learning_rate * dw)
  model.b.assign_sub(learning_rate * db)
  return current_loss
```

The Machine Learning Paradigm



```python
for epoch in range(50):
  current_loss = train(model, xs, ys, learning_rate=0.1)
```



$$y = 1.98x - 0.94$$

Your turn!