- [Get Started with Eager Execution](#)（入门）
  - [在线运行](#)
  - 鸢尾花分类教程，使用eager execution完成：（1）构建模型；（2）训练模型；（3）利用模型预测未知数据。
  - TensorFlow编程
  - TensorFlow高级概念：（1）启动一个 eager execution开发环境；（2）使用Datasets API导入数据；（3）使用Keras API构建模型
  - 教程涉及APIs：（1）导入和解析数据集；（2）选择模型类型；（3）训练模型；（4）评估模型有效性；（5）使用训练好的模型进行预测
  - 机器学习基础：[机器学习速成课程](#)
  - 运行notebook：（1）Connect to runtime；（2）run
  - 安装tensorflow最新版本（需要版本1.7.0，注意notebook需要重新启动代码执行程序）
  - 一旦启动了 eager execution，将不能在同一程序中禁用。详见[eager execution guide](#)。

```python
import os
import matplotlib.pyplot as plt

import tensorflow as tf
import tensorflow.contrib.eager as tfe

tf.enable_eager_execution()

print("TensorFlow version: {}".format(tf.VERSION))
print("Eager execution: {}".format(tf.executing_eagerly()))

# result
# TensorFlow version: 1.7.0
# Eager execution: True
```

  - 鸢尾花分类问题：根据萼片和花瓣的长度和宽度来对鸢尾花进行分类
  - 鸢尾花三种类别：（1）Iris setosa（2）Iris virginica（3）Iris versicolor
  - [鸢尾花数据集](#)：120条数据
  - 导入和解析训练数据集
  - 下载数据

```python
train_dataset_url = "http://download.tensorflow.org/data/iris_training.csv"

train_dataset_fp = tf.keras.utils.get_file(fname=os.path.basename(train_dataset_url),
                                           origin=train_dataset_url)

print("Local copy of the dataset file: {}".format(train_dataset_fp))
```

  - 检查数据

```python
def load_iris_data():
    csv_path = "data/iris_training.csv"
    data = pd.read_csv(csv_path)
    return data

data = load_iris_data()
print(data.head()) # 查看前五行
```

  - 前五行内容：

	120	4	setosa	versicolor	virginica
0	6.4	2.8	5.6	2.2	2
1	5.0	2.3	3.3	1.0	1
2	4.9	2.5	4.5	1.7	2
3	4.9	3.1	1.5	0.1	0
4	5.7	3.8	1.7	0.3	0

  - 类别标签编码：0—setosa, 1—versicolor，2—virginica
  - 特征与标签更多学习：[ML Terminology section of the Machine Learning Crash Course](#)
  - 解析数据集：将文本数据转为特征和标签张量

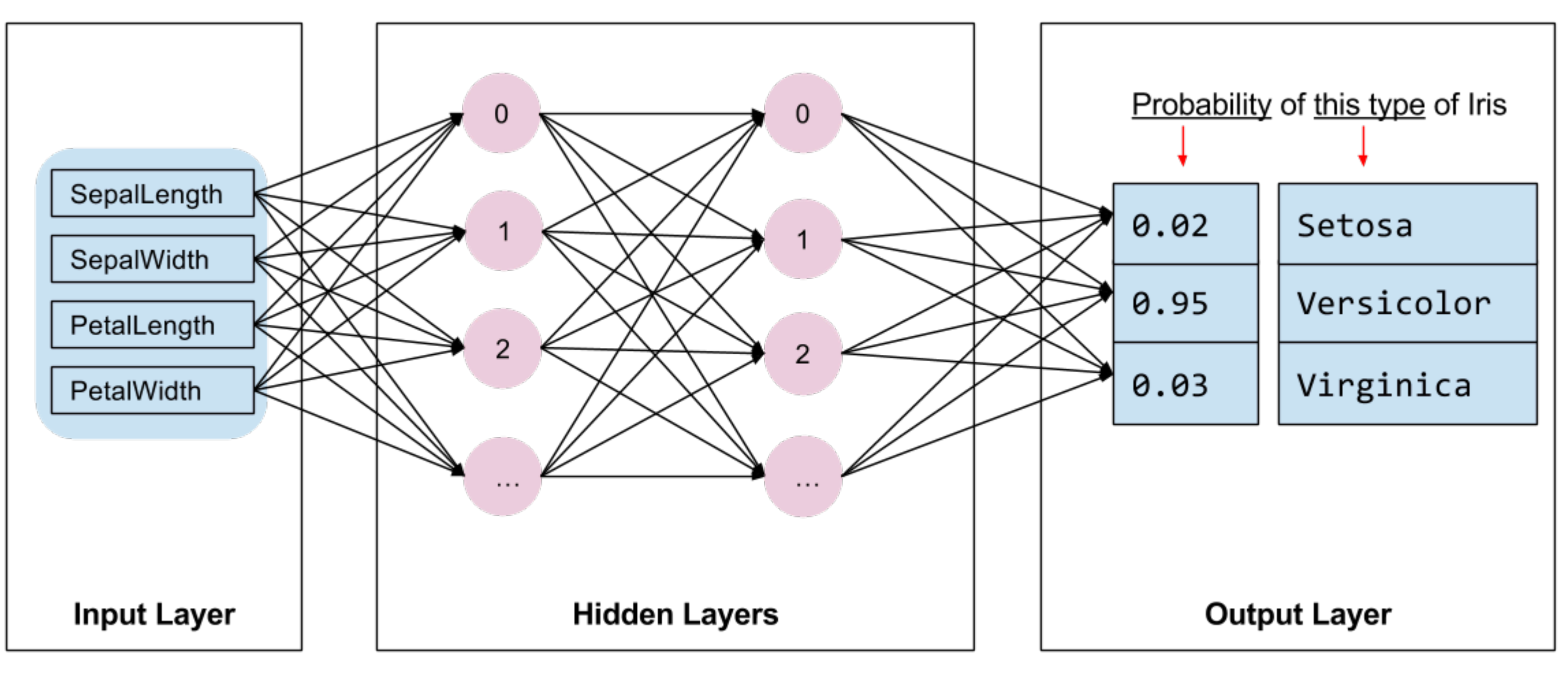
```python
def parse_csv(line):
    example_defaults = [[0.], [0.], [0.], [0.], [0]] # sets field types
    parsed_line = tf.decode_csv(line, example_defaults)
    features = tf.reshape(parsed_line[:-1], shape=(4,)) # 抽取前四列特征
    label = tf.reshape(parsed_line[-1], shape=()) # 抽取最后一列标签
    return features, label
```

  - 创建训练集：[Dataset API](#)，更多参见[Datasets Quick Start guide](#)
  - 训练数据最好是随机顺序的，利用【tf.data.Dataset.shuffle】随机化。
  - 设置buffer_size > 数据个数（120），设置batch_size=32

```python
data_path = "data/iris_training.csv"
train_dataset = tf.data.TextLineDataset(data_path)
train_dataset = train_dataset.skip(1)  # skip the first header row
train_dataset = train_dataset.map(parse_csv)  # parse each row
train_dataset = train_dataset.shuffle(buffer_size=1000)  # randomize
train_dataset = train_dataset.batch(32)

# View a single example entry from a batch
features, label = tfe.Iterator(train_dataset).next()
print("example features:", features[0])
print("example label:", label[0])
```

  - 选择模型的类型
  - 模型是特征和标签的关系
  - 神经网络模型，全连接神经网络（dense, or fully-connected neural network）



  - 使用Keras构建模型：[tf.keras API](#)

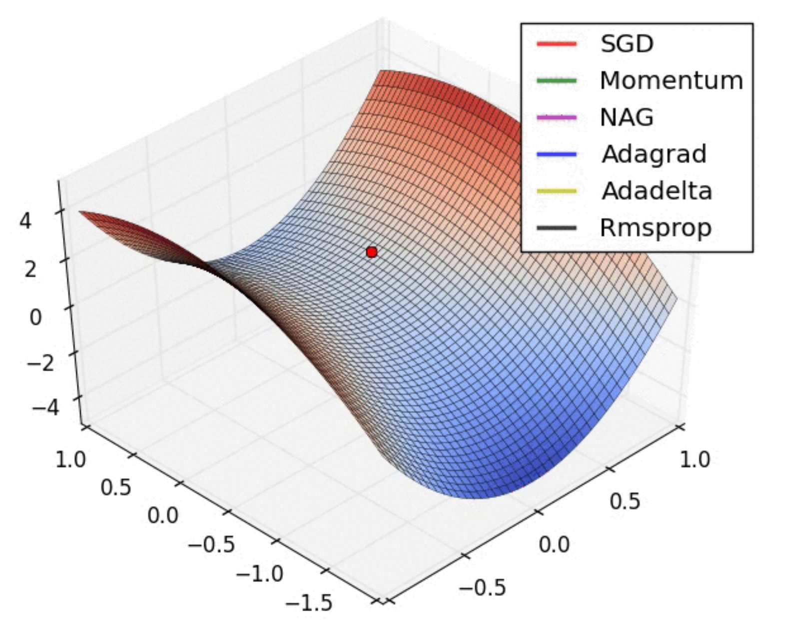
```python
model = tf.keras.Sequential([
    tf.keras.layers.Dense(10, activation="relu", input_shape=(4,)),  # input shape required
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(3)
])
```

  - 训练模型
  - 过拟合问题（overfitting）
  - [监督学习](#)，[非监督学习](#)
  - 定义损失（loss）和梯度函数
  - tf.losses.sparse_softmax_cross_entropy

```python
def loss(model, x, y):
    y_ = model(x)
    return tf.losses.sparse_softmax_cross_entropy(labels=y, logits=y_)


def grad(model, inputs, targets):
    with tfe.GradientTape() as tape:
        loss_value = loss(model, inputs, targets)
    return tape.gradient(loss_value, model.variables)
```

  - 创建优化器（optimizer）：优化算法，最小化损失函数，找到最低点



  - 学习率（超参数）
  - 随机梯度下降优化算法：tf.train.GradientDescentOptimizer

```python
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
```

  - 循环训练
  - num_epochs（轮次，超参数）

```python
# keep results for plotting
train_loss_results = []
train_accuracy_results = []

num_epochs = 201

for epoch in range(num_epochs):
    epoch_loss_avg = tfe.metrics.Mean()
    epoch_accuracy = tfe.metrics.Accuracy()

    # Training loop - using batches of 32
    for x, y in tfe.Iterator(train_dataset):
        # Optimize the model
        grads = grad(model, x, y)
        optimizer.apply_gradients(zip(grads, model.variables),
                                  global_step=tf.train.get_or_create_global_step())

        # Track progress
        epoch_loss_avg(loss(model, x, y))  # add current batch loss
        # compare predicted label to actual label
        epoch_accuracy(tf.argmax(model(x), axis=1, output_type=tf.int32), y)

    # end epoch
    train_loss_results.append(epoch_loss_avg.result())
    train_accuracy_results.append(epoch_accuracy.result())

    if epoch % 50 == 0:
        print("Epoch {:03d}: Loss: {:.3f}, Accuracy: {:.3%}".format(epoch,
                                                                    epoch_loss_avg.result(),
                                                                    epoch_accuracy.result()))
```

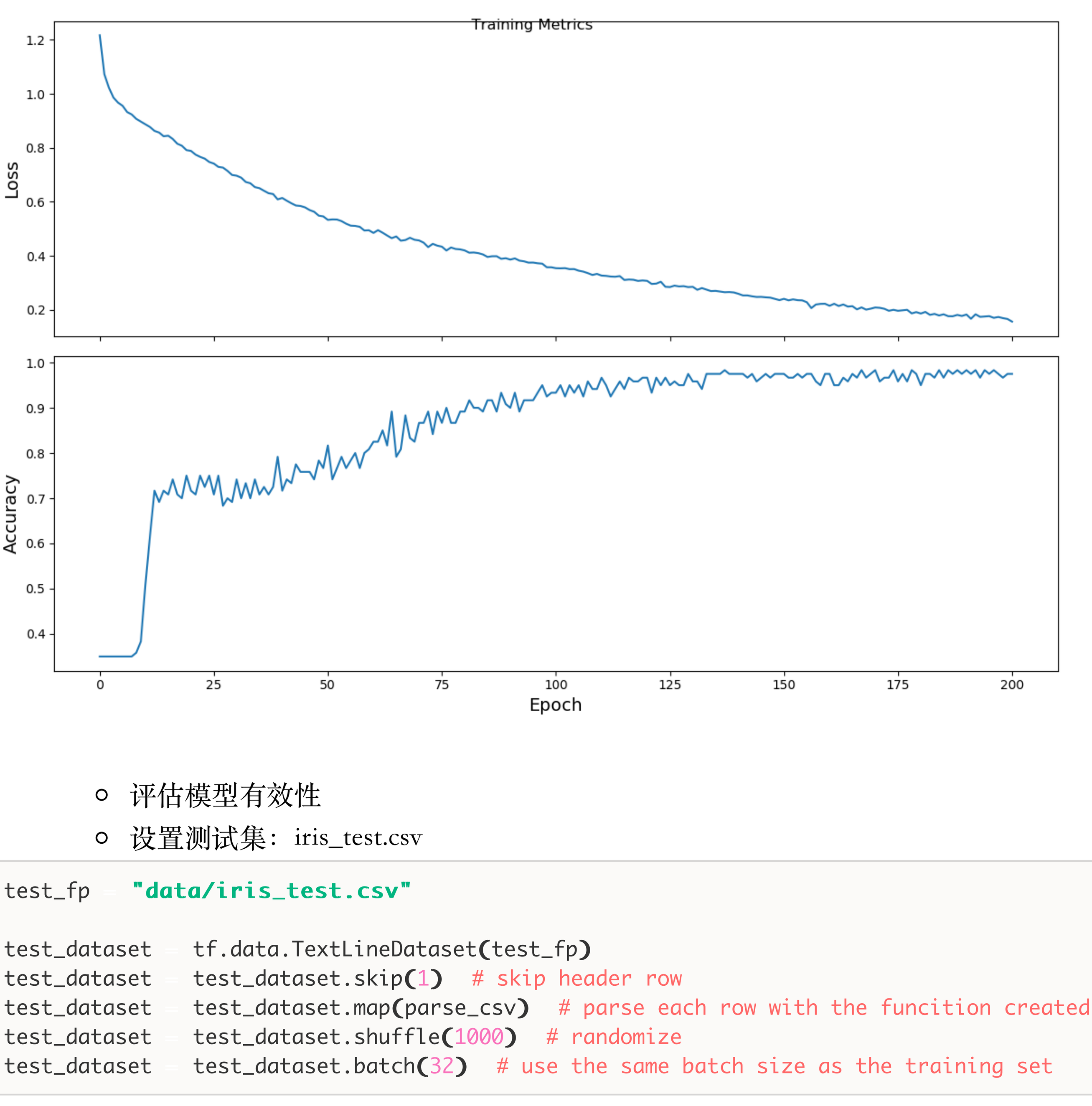
  - 可视化损失函数：TensorBoard，mathplotlib

```python
fig, axes = plt.subplots(2, sharex=True, figsize=(12, 8))
fig.suptitle('Training Metrics')

axes[0].set_ylabel("Loss", fontsize=14)
axes[0].plot(train_loss_results)

axes[1].set_ylabel("Accuracy", fontsize=14)
axes[1].set_xlabel("Epoch", fontsize=14)
axes[1].plot(train_accuracy_results)

plt.show()
```



  - 评估模型有效性
  - 设置测试集：iris_test.csv

```python
test_fp = "data/iris_test.csv"

test_dataset = tf.data.TextLineDataset(test_fp)
test_dataset = test_dataset.skip(1)  # skip header row
test_dataset = test_dataset.map(parse_csv)  # parse each row with the funcition created earlier
test_dataset = test_dataset.shuffle(1000)  # randomize
test_dataset = test_dataset.batch(32)  # use the same batch size as the training set
```

  - 在测试集上评估模型：Test set accuracy: 96.667%

```python
test_accuracy = tfe.metrics.Accuracy()

for (x, y) in tfe.Iterator(test_dataset):
    prediction = tf.argmax(model(x), axis=1, output_type=tf.int32)
    test_accuracy(prediction, y)

print("Test set accuracy: {:.3%}".format(test_accuracy.result()))
```

  - 使用训练好的模型预测未知数据

```python
print("Test set accuracy: {:.3%}".format(test_accuracy.result()))

class_ids = ["Iris setosa", "Iris versicolor", "Iris virginica"]

predict_dataset = tf.convert_to_tensor([
    [5.1, 3.3, 1.7, 0.5,],
    [5.9, 3.0, 4.2, 1.5,],
    [6.9, 3.1, 5.4, 2.1]
])

predictions = model(predict_dataset)

for i, logits in enumerate(predictions):
    class_idx = tf.argmax(logits).numpy()
    name = class_ids[class_idx]
    print("Example {} prediction: {}".format(i, name))
```

更新时间：2018年04月19日17:03:19