6 变量的trainable设置观察

tf.Variable OP：特殊之处

```
trainable=False
# 指定某个变量参数不跟着梯度更新参数
```

2.4.3 增加其他功能

- summary: events
- 添加观察损失和参数
  - scalar:添加一个float值的变化(观察损失变化、观察准确率变化)
  - histogram：观察高维tensor分布情况(模型权重、偏置参数的分布)

3 模型的保存与加载

- TensorFlow：checkpoint文件

```
saver.save(sess, '/tmp/ckpt/test/myregression.ckpt')
saver.restore(sess, '/tmp/ckpt/test/myregression.ckpt')
```

命令行参数使用

2.5 TFAPI使用2.0建议

6.6 tf.estimator使用入门

6.6.1 tf.estimator介绍

- **编写一个或多个数据集导入函数。**
  - tf.data
  - 返回值：**一个字典，其中键是特征名称，值是包含相应特征数据的张量（或SparseTensor）****一个包含一个或多个标签的张量**
  - input_fn
- **定义特征列**
  - tf.feature_column：对特征做标识
  - 大多数机器学习问题：连续值和离散特征需要numeric_column
- **实例化相关的预创建的 Estimator**

```
estimator = tf.estimator.LinearClassifier(
    feature_columns=[population, crime_rate, median_education],
    )
```

- **调用训练、评估或推理方法。**例如，所有 Estimator 都提供训练模型的 `train` 方法。
    - train(input_fn=input_fn)
    - evaluate()

6.6.2 案例：使用美国普查数据分类

- 特征值：个人信息情况
    - **类别列和连续列**：
        - 如果某个列的值只能是一个有限集合中的类别之一，则该列称为类别列。例如，婚恋状况（妻子、丈夫、未婚等）或受教育程度（高中、大学等）属于类别列。
        - 如果某个列的值可以是连续范围内的任意数值，则该列称为连续列。例如，一个人的资本收益（如 14084 美元）属于连续列。
- 目标值：收入为高收入1， 低收入0
    - '<=50K': 0
    - '>50K': 1
- 1、读取美国普查收入数据
    - `TextLineDataset`， .txt
    - `tf.decode_csv`
    - 1000—>循环训练—>两边（1000）
    - 批处理大小
    - map, repeat,batch tf.data
- 2、模型选择特征并进行特征工程处理
    - 13列特征必须全部指定一下feature_column
    - 连续型：numeric_column
    - 离散型：字符类型类别
    - `tf.feature_column.categorical_column_with_vocabulary_list`： 指定所有类别字符串，对应到具体数字类别
    - `tf.feature_column.categorical_column_with_hash_bucket`：类别数量过多不确定到底有多少类别，指定一个hash_bucket_size
- 3、模型训练与评估
    - estimator.train或者evaluate
    - **input_fn参数填入的数据获取函数不能有参数**

7.1 神经网络基础

7.1.1 神经网络

- 1、每个连接都有个权值

- 2、同一层神经元之间没有连接
- 3、最后的输出结果对应的层也称之为**全连接层**

- **权重（weight）：就是之前所说的参数，这里被称为一个神经节点的权重。**
- **激活函数（activation function）：激活函数是两层神经元之间的映射函数，是一种输出到输入的转换，一般是非线性的，而且是单调可微函数（因为优化方法是基于梯度的）。常见的激活函数有：sigmoid,tanh**

7.1.1.1 感知机(PLA: Perceptron Learning Algorithm))

- **感知机具有连接的权重和偏置**
- 缺点：对非线性的数据并不能进行有效的分类
  - 感知机模型或者逻辑回归都不能对非线性数据做一个分类
- 神经网络：多个感知机组成(单层)

7.1.4 神经网络发展史

- 1958，感知机
- 1969年Marvin Minsky写了一本叫做《Perceptrons》的书，他提出了著名的两个观点：**1.单层感知机没用，我们需要多层感知机来解决复杂问题 2.没有有效的训练算法**
- 1986年BP算法， 才真正开始流行起来，主要是因为Rumelhart、Hinton、Williams合著的《Learning representations by back-propagating errors》
- 而90年代中期，由Vapnik等人发明的支持向量机（Support Vector Machines，SVM）算法诞生
- 直到2006年，提出了"深度置信网络"概念，2012年，在ImageNet竞赛中，Hinton教授的团队，使用以卷积神经网络

7.2 神经网络多分类原理与反向传播原理

7.2.1 神经网络计算输出

- 权重数量：100个神经元， 每个样本200个特征100 * 200 = 20000
- 输出：一个值：概率 ，逻辑：二分类
- 神金网络：解决多分类问题比较擅长
  - **神经网络解决多分类问题最常用的方法是设置n个输出节点，其中n为类别的个数。**

7.2.3 softmax回归

作用：Softmax回归将神经网络输出转换成概率结果

- softmax特点
  - 每个神经元输出一个概率
  - 所有概率相加为1
- 逻辑回归：sigmoid
  - 二分类问题

- softmax特点

7.2.4 分类损失-交叉熵损失

- loss = y_true * log(y_predict)

逻辑回归：对数似然损失， 梯度下降算法

线性回归：均方误差， 梯度下降算法

神经网络：交叉熵损失， 反向传播算法BP-back-propagation

7.2.6 反向传播算法

- 导数、导数计算图
- 链式法则、逻辑回归的梯度下降优化、向量化编程
- 浅层神经网络的前向计算(传播)与反向计算过程

7.2.6.1 导数

导数也可以理解成某一点处的斜率

- 各点处的导数值一样：当一个点偏移一个不可估量的小的值，所增加的倍数

$$\frac{f(a)}{da} \text{或者} \frac{d}{da} f(a)$$

- 各点的导数值不全一致
  - f(a) = a^2 = 2a
-

函数	导数
$f(a) = a^2$	$2a$
$f(a) = a^3$	$3a^2$
$f(a) = ln(a)$	$\frac{1}{a}$
$f(a) = e^a$	e^a
$\sigma(z) = \frac{1}{1+e^{-z}}$	$\sigma(z)(1-\sigma(z))$
$g(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$1-(tanh(z))^2 = 1-(g(z))^2 1$

7.2.6.2 导数计算图

$J(a,b,c)=3(a+b**c)$

拆解成流程图：

问题：那么现在我们计算J相对于三个变量a,b,c的导数?

- 当损失函数涉及过多的复合函数，求导，链式法则去进行求解

7.2.6.4 逻辑回归的梯度下降

我们使用链式求导法则，反向层层推进，计算出每一层神经节点的偏导数，然后使用梯度下降，不断调整每一个节点的权重（也就是参数theta），从而达到求得全局最小值的目的。

7.2.7 向量化编程

- 批梯度下降：每次计算损失，通过链式法则求出梯度，梯度下降算法更新，用M个样本
  - min-batch：M个选择，32， 64， 128.....
  - W1， 32个梯度值，求平均值
- 随机梯度下降：每次计算损失，通过链式法则求出梯度，梯度下降算法更新，用1个样本

7.2.7.1 向量化优势

最好不要使用for循环去进行计算，因为有Numpy可以进行更加快速的向量化计算。

7.2.7.2 向量化实现伪代码(逻辑回归计算损失，计算梯度，下降过程)

7.2.8 浅层神经网络的前向传播与反向传播

7.2.9 激活函数的选择

激活函数

- tanh 函数（the hyperbolic tangent function，双曲正切函数）
  - 注 :tanh 函数存在和 sigmoid 函数一样的缺点：当 z 趋紧无穷大（或无穷小），导数的梯度（即函数的斜率）就趋紧于 0，这使得梯度算法的速度会减慢。
- 当 z > 0 时，梯度始终为 1，从而提高神经网络基于梯度算法的运算速度，收敛速度远大于 sigmoid 和 tanh。然而当 z < 0 时，梯度一直为 0，但是实际的运用中，该缺陷的影响不是很大。

为什么神金网络需要激活函数?

- 10层，100层没有多大区别

7.3 案例：神经网络分类

7.3.1 美国普查数据神经网络分类

- tf.estimator神经网络，输入列特征，连续型，
  - 连续型特征列不需要处理
  - 离散型特征列：不接受cateogray列特征

- 输入神金网络接口：
  - 离散型特征：1，2，3，4
  - 1、变成低维稠密的实数向量(前者)
  - 2、编程one_hot输入到网络

{'accuracy': 0.8352067, 'accuracy_baseline': 0.76377374, 'auc': 0.8836825, 'auc_precision_recall': 0.6844692, 'average_loss': 0.4832691, 'label/mean': 0.23622628, 'loss': 15.457965, 'precision': 0.66014874, 'prediction/mean': 0.25121066, 'recall': 0.62324494, 'global_step': 1018} accuracy: 0.8352067 accuracy_baseline: 0.76377374 auc: 0.8836825 auc_precision_recall: 0.6844692 average_loss: 0.4832691 global_step: 1018 label/mean: 0.23622628 loss: 15.457965 precision: 0.66014874 prediction/mean: 0.25121066 recall: 0.62324494

{'accuracy': 0.8483508, 'accuracy_baseline': 0.76377374, 'auc': 0.89856523, 'auc_precision_recall': 0.75047415, 'average_loss': 0.3375258, 'label/mean': 0.23622628, 'loss': 10.796185, 'precision': 0.7294235, 'prediction/mean': 0.23496841, 'recall': 0.5691628, 'global_step': 1018} accuracy: 0.8483508 accuracy_baseline: 0.76377374 auc: 0.89856523 auc_precision_recall: 0.75047415 average_loss: 0.3375258 global_step: 1018 label/mean: 0.23622628 loss: 10.796185 precision: 0.7294235 prediction/mean: 0.23496841 recall: 0.5691628

作业：设计一个多层感知机，解决异或问题?