TensorFlow 2.0学习-第一课

课程名称：人工智能实践：Tensorflow笔记

Created by：Henry Huang

0.本次笔记的函数总结：

(1)创建张量

(1-1)已知值所有内容的创建张量

```
tf.constant(张量内容，dtype=数据类型(可选参数))
```

(1-2)创建全为0的张量--维度书写[i,j,k,l,...],只有一维不需要用中括号

```
tf.zeros(维度)
```

(1-3)创建全为1的张量

```
tf.ones(维度)
```

(1-4)创建全为指定值的张量

```
tf.fill(维度,指定值)
```

(1-5)随机张量创建

```
tf.random.normal(维度，mean=均值，stddev=标准差)#标准的正态分布
tf.random.truncated_normal (维度，mean=均值，stddev=标准差)#截断的正态分布，超过两倍标准差会重新随机
tf.random.uniform(维度，minval=最小值，maxval=最大值)#均匀分布
```

(2)将numpy数据转化为张量

```
tf.convert_to_tensor(numpy的数据名，dtype=数据类型(可选参数))
```

(3)tensor转换类型

```
tf.cast（张量名，dtype=数据类型）
```

(4)计算最大、最小、均值、方差
** axis**参数：为0代表向下方向运算，即跨行操作；为1代表向右方向运算，即跨列操作。

```
tf.reduce_min（张量名）
tf.reduce_max（张量名）
tf.reduce_mean（张量名，axis=操作轴）#计算张量沿着指定维度的平均值
tf.reduce_sum（张量名，axis=操作轴）#计算张量沿着指定维度的和
```

(5)可训练变量

```
tf.Variable(初始值)#tf.Variable () 将变量标记为“可训练”，被标记的变量会在反向传播
中记录梯度信息。神经网络训练中，常用该函数标记待训练参数。
```

(6)四则运算：

(7)其他数学运算

(8)打标签函数

```
tf.data.Dataset.from_tensor_slices((输入特征，标签))
```

(9)求梯度函数

```
with tf.GradientTape( ) as tape:
若干个计算过程
grad=tape.gradient(函数，对谁求导)
```

(10)遍历器enumerate
用于遍历列表中的每个元素。
前面是索引，后面是具体的元素。

```
seq = ['one', 'two', 'three']
for i, element in enumerate(seq):
print(i, element)
```

(11)独热编码器

```
tf.one_hot(需要编码的变量名字，depth=到底是几个分类需要进行独热编码)
```

(12)softmax函数

可以使得神经网络的输出符合概率分布。

(13)自减函数
w必须是可训练的变量

```
w.assign_sub（w要自减的内容）
```

(14)返回张量沿着指定维度的最大值索引

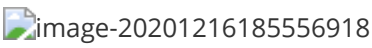
```
tf.argmax（张量名,axis=操作轴）
```

1.初识神经网络

1.1.神经网络三个学派

- 行为主义：基于控制论，“感知-动作”
- 符号主义：基于算数逻辑表达式，求解时可以把问题描述为表达式（专家系统）
- 连接主义：仿生学（神经网络）

1.2.神经网络的设计

 image-20201216185556918

如上图所示，其实每个神经元的运行机制就是：输出=非线性函数（每个输入*w即对应权重+偏置b）。

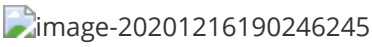
因此训练神经网络的步骤就是：

1. 先随机初始化w和b。然后输入数据，然后使用loss函数计算现有输出和标签的误差。
2. 然后使用梯度下降法，求出loss对每个参数的导数，然后反向传导更新参数
3. 最终的目的就是，使得训练之后，该神经网络对输入的数据集的loss最小

1.2.1 loss函数初识

就是用来衡量预测值与真实值的差距。可以使用诸如MSE均方误差之类的。

1.2.2反向传播更新参数

 image-20201216190246245

其实就是w每次更新为：原来的参数-学习率*loss对该参数的偏导数

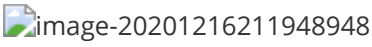
lr为学习率，其实就是梯度下降的步长。

学习率太低，训练太慢。学习率高，容易错过loss最小的地方，导致训练不收敛。

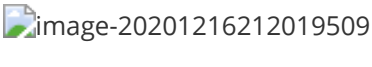
2.Tensorflow基础

2.1 张量Tensor

张量:其实就是一个多维的数组或者列表。

 image-20201216211948948

支持的数据类型如下：

 image-20201216212019509

创建一个张量:

（1）单独创建一个张量

```
tf.constant(张量内容，dtype=数据类型(可选参数))
```

（2）使用numpy中的数据转化而来

```
tf.convert_to_tensor(numpy的数据名，dtype=数据类型(可选参数))
```

3. 其他额外小知识

3.1with 语句有啥用?

可以见链接：https://blog.csdn.net/youzhouliu/article/details/80976004
就是访问资源使用的。

```
#例如：
#(1)紧跟with后面的语句被求值后，返回对象的“__ enter__()”方法被调用，这个方法的返回值将被赋值给as后面的变量；
#(2)当with后面的代码块全部被执行完之后，将调用前面返回对象的“__exit__()”方法。
class Sample:
    def __enter__(self):
        print "in __enter__"
        return "Foo"
    def __exit__(self, exc_type, exc_val, exc_tb):
        print "in __exit__"
def get_sample():
    return Sample()
with get_sample() as sample:
    print "Sample: ", sample

#结果会如下：
in __enter__
Sample:  Foo
in __exit__
```

3.2 batch\epoch是啥?

我们可以一次性将整个数据集喂给神经网络，让神经网络利用全部样本来计算迭代时的梯度（即传统的梯度下降法），也可以一次只喂一个样本（即随机梯度下降法，也称在线梯度下降法），也可以取个折中的方案，即每次喂一部分样本让其完成本轮迭代（即batch梯度下降法）。

第一种是将参数一次性更新500个样本的量，第二种是迭代的更新500次参数。当然是不一样的啦。

知乎高赞

https://blog.csdn.net/L_0000/article/details/85067467?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-3.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-3.control

https://blog.csdn.net/weixin_43202635/article/details/84204180