人工智能实践：Tensorflow笔记

曹 健

北京大学

软件与微电子学院

本讲目标：学会神经网络计算过程，使用基于**TF2**原生代码搭建你的第一个的神经网络训练模型

- 当今人工智能主流方向——连接主义
- 前向传播
- 损失函数（初体会）
- 梯度下降（初体会）
- 学习率（初体会）
- 反向传播更新参数
- **Tensorflow 2** 常用函数

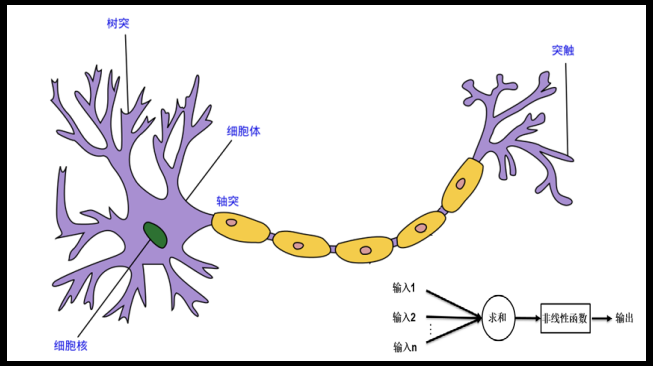
人工智能：让机器具备人的思维和意识。

人工智能三学派：

✓ **行为主义**：基于控制论，构建感知-动作控制系统。（控制论，如平衡、行走、避障等自适应控制系统）

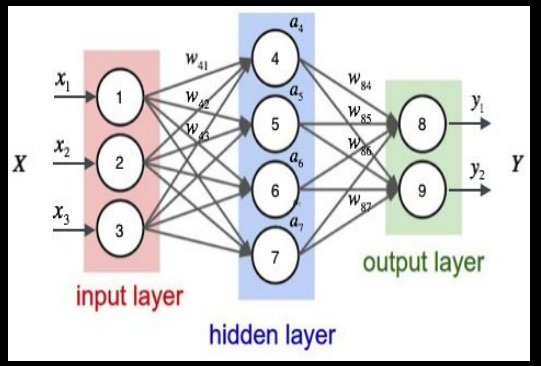
✓ **符号主义**：基于算数逻辑表达式，求解问题时先把问题描述为表达式，再求解表达式。（可用公式描述、实现理性思维，如专家系统）

✓ **连接主义**：仿生学，模仿神经元连接关系。（仿脑神经元连接，实现感性思维，如神经网络）



单个神经元

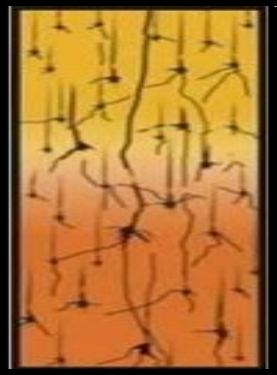
神经元连接成网络



人工神经网络

理解：基于连结主义的神经网络设计过程

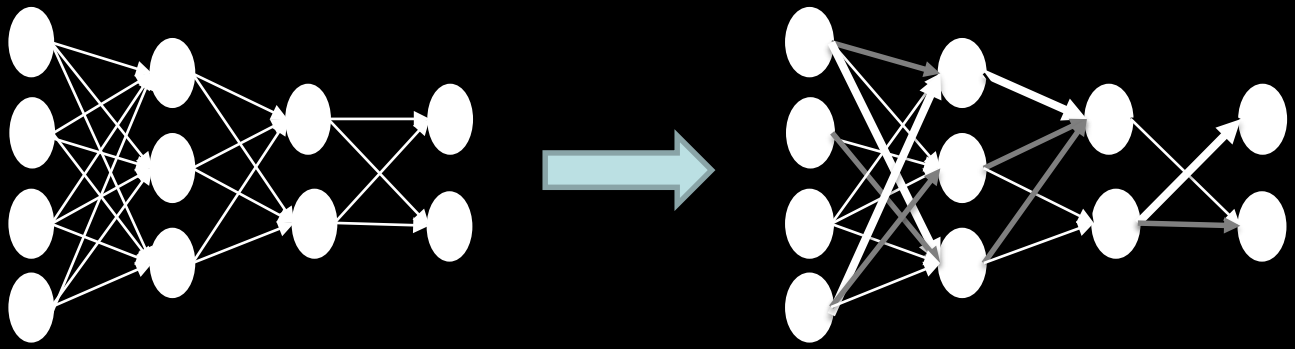

新生儿


1个月


9个月


2年


成年

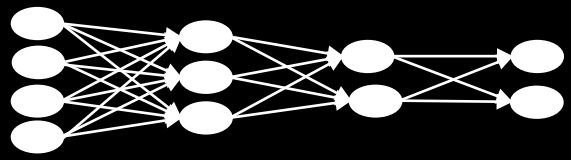


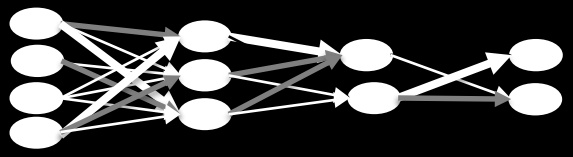
用计算机仿出神经网络连接关系，让计算机具备感性思维。

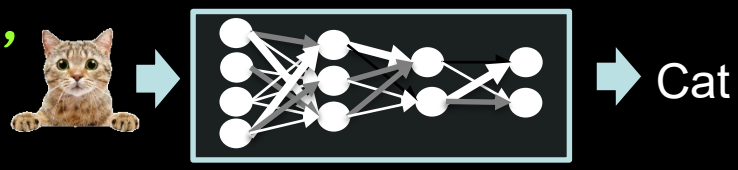
Cat

（特 征，　　标 签）

- ✓ 准备数据：采集大量“特征/标签”数据
- ✓ 搭建网络：搭建神经网络结构
- ✓ 优化参数：训练网络获取最佳参数（反传）
- ✓ 应用网络：将网络保存为模型，输入新数据，输出分类或预测结果（前传）







给鸢尾花分类（Iris）



0狗尾草鸢尾

1杂色鸢尾

2弗吉尼亚鸢尾

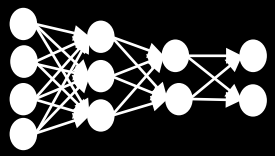
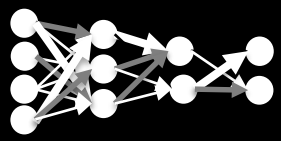
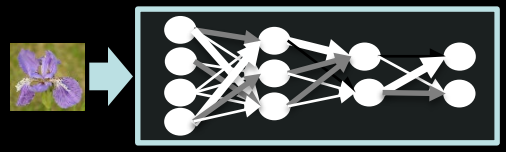
这是哪类鸢尾花？

人们通过经验总结出了规律：通过测量花的花萼长、花萼宽、花瓣长、花瓣宽，可以得出鸢尾花的类别。
（如：花萼长>花萼宽 且 花瓣长/花瓣宽>2 则为 1杂色鸢尾）

if语句 case语句 —— 专家系统 把专家的经验告知计算机，计算机执行逻辑判别（理性计算），给出分类。

神经网络：采集大量（花萼长、花萼宽、花瓣长、花瓣宽 ， 对应的类别）数据对构成数据集
输入特征 标签 （需人工标定）

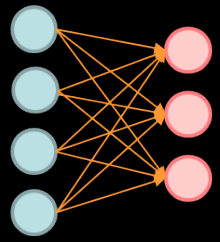
把数据集喂入搭建好的神经网络结构， 网络优化参数得到模型， 模型读入新输入特征， 输出识别结果。

1杂色鸢尾

用神经网络实现鸢尾花分类

[花萼长、花萼宽、花瓣长、花瓣宽]



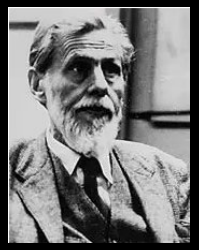
输入层　输出层

0狗尾草鸢尾

1杂色鸢尾

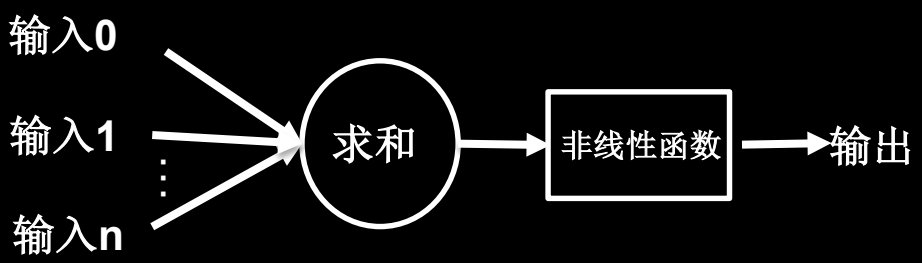
2弗吉尼亚鸢尾

输出是每个种类的可能性大小

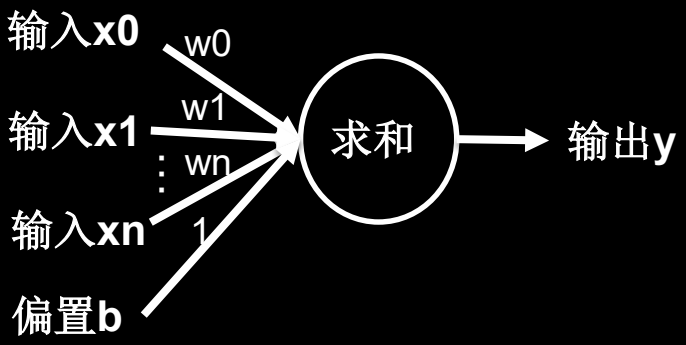


McCulloch
麦卡洛克

Pitts
皮茨



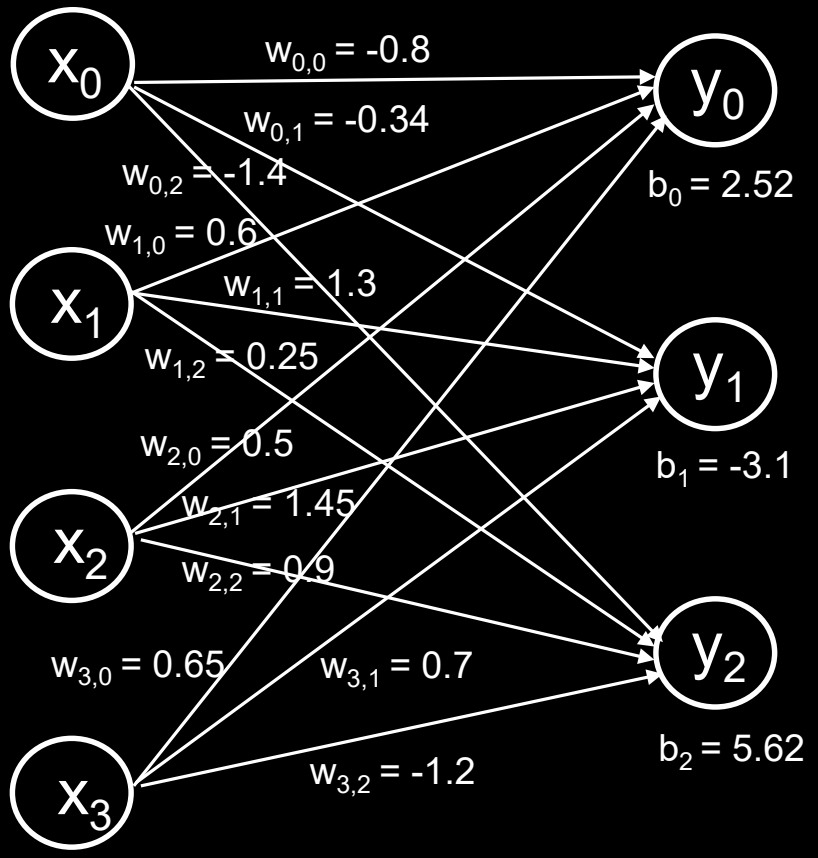
MP模型



$$y = x * w + b$$

（1，3）　（1，4）（4，3）（3，）

用神经网络实现鸢尾花分类：搭建网络



搭建网络时随机初始化了所有参数w和b：

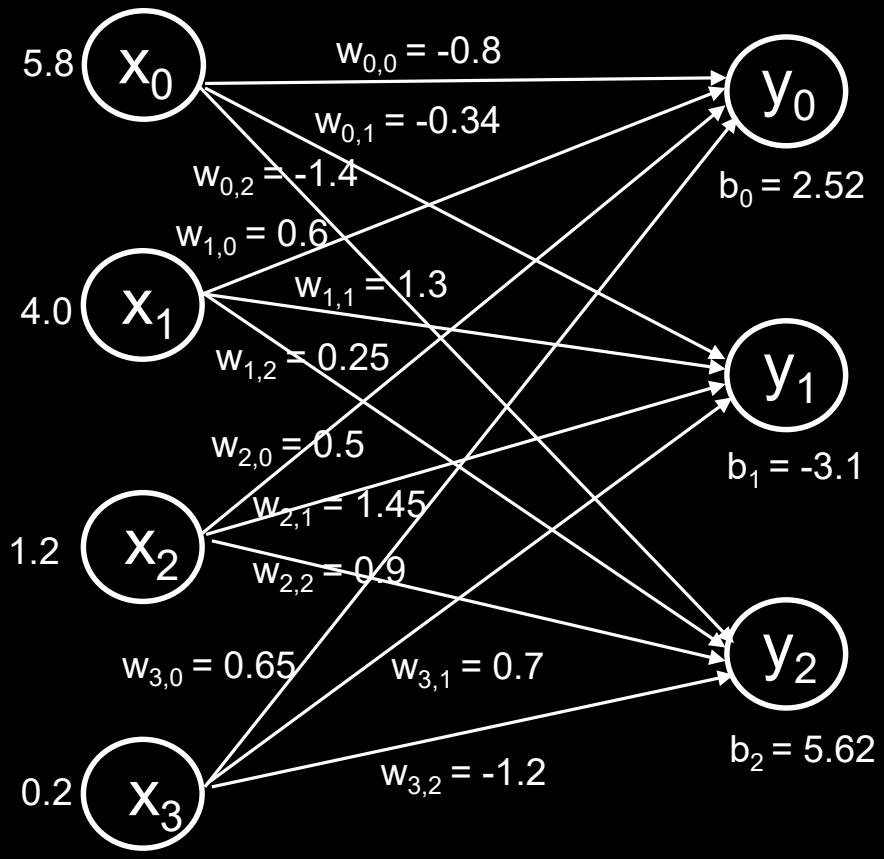
-0.8	-0.34	-1.4
0.6	1.3	0.25
0.5	1.45	0.9
0.65	0.7	-1.2

随机初始化w

2.52	-3.1	5.62

随机初始化b

用神经网络实现鸢尾花分类：喂入数据



搭建网络时随机初始化了所有参数w和b：

-0.8	-0.34	-1.4
0.6	1.3	0.25
0.5	1.45	0.9
0.65	0.7	-1.2

随机初始化w

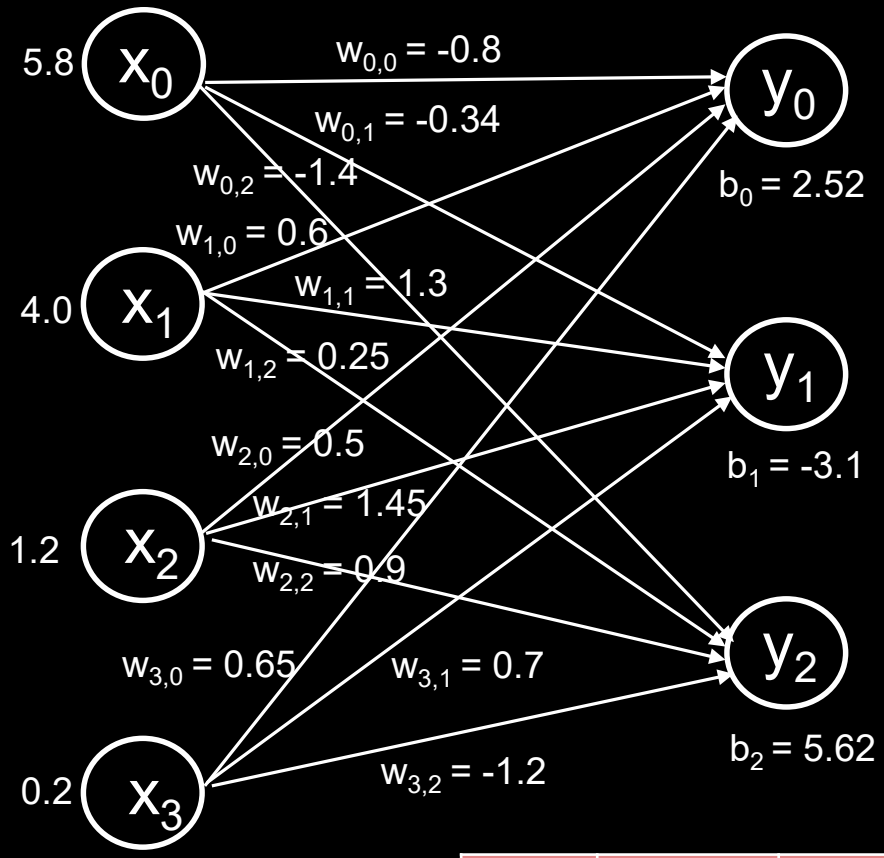
2.52	-3.1	5.62

随机初始化b

标签：0狗尾草鸢尾

用神经网络实现鸢尾花分类：前向传播

源码：p38_forward.py



搭建网络时随机初始化了所有参数w和b：

-0.8	-0.34	-1.4
0.6	1.3	0.25
0.5	1.45	0.9
0.65	0.7	-1.2

随机初始化w

2.52	-3.1	5.62

随机初始化b

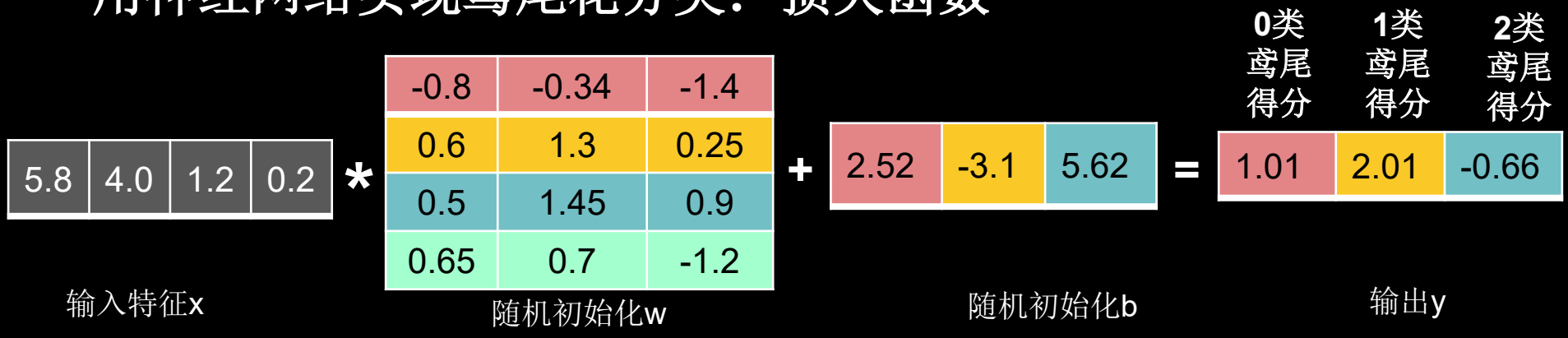
y = x * w + b ：

5.8	4.0	1.2	0.2	*	-0.8	-0.34	-1.4	+	2.52	-3.1	5.62	=	1.01	2.01	-0.66
					0.6	1.3	0.25								
					0.5	1.45	0.9								
					0.65	0.7	-1.2								

输入特征x　　随机初始化w　　随机初始化b　　输出y

0类鸢尾得分　1类鸢尾得分　2类鸢尾得分

用神经网络实现鸢尾花分类：损失函数



标签：0狗尾草鸢尾

为什么0类鸢尾的得分不是最高？

损失函数（loss function）：预测值（y）与标准答案（y_）的差距。

损失函数可以定量判断W、b的优劣，当损失函数输出最小时，参数W、b会出现最优值。

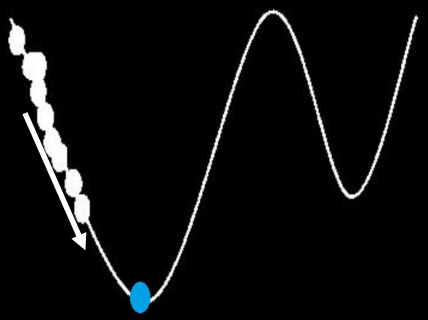
均方误差：　$MSE(y, y\_) = \frac{\sum_{k=0}^{n}(y-y\_)^2}{n}$

用神经网络实现鸢尾花分类：梯度下降

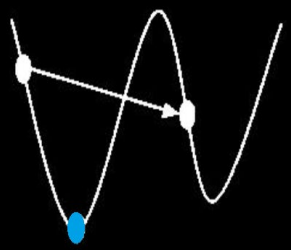
目的：想找到一组参数w和b，使得损失函数最小。

梯度： 函数对各参数求偏导后的向量。 函数梯度下降方向是函数减小方向。

梯度下降法：沿损失函数梯度下降的方向，寻找损失函数的最小值，得到最优参数的方法。



学习率过小



学习率过大

$$w_{t+1} = w_t - lr * \frac{\partial loss}{\partial w_t}$$

$$b_{t+1} = b - lr * \frac{\partial loss}{\partial b_t}$$

$$w_{t+1} * x + b_{t+1} \rightarrow y$$

学习率（learning rate，lr）：当学习率设置的过小时，收敛过程将变得十分缓慢。而当学习率设置的过大时，梯度可能会在最小值附近来回震荡，甚至可能无法收敛。

用神经网络实现鸢尾花分类：反向传播

$$w_{t+1} = w_t - lr * \frac{\partial loss}{\partial w_t}$$

✓ **反向传播：** 从后向前，逐层求损失函数对每层神经元参数的偏导数，迭代更新所有参数。

eg：损失函数 loss = (w + 1)²　　$\frac{\partial loss}{\partial w} = 2w + 2$

参数w初始化为5，学习率为0.2 则

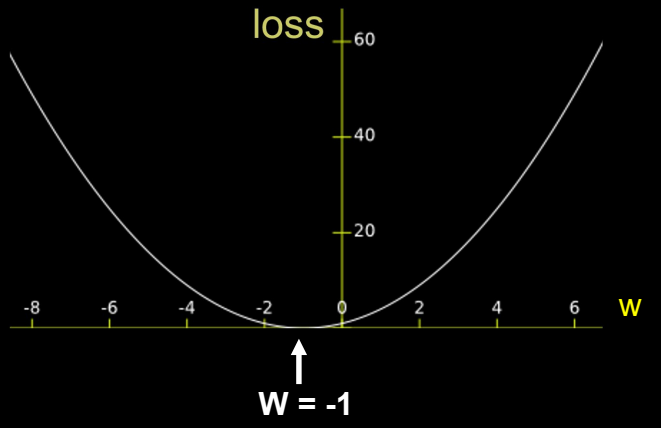
1次　参数w：5　　5 - 0.2 * (2 * 5 + 2) = 2.6

2次　参数w：2.6　　2.6 - 0.2 * (2 * 2.6 + 2) = 1.16

3次　参数w：1.16　　1.16 – 0.2 * (2 * 1.16 + 2) = 0.296

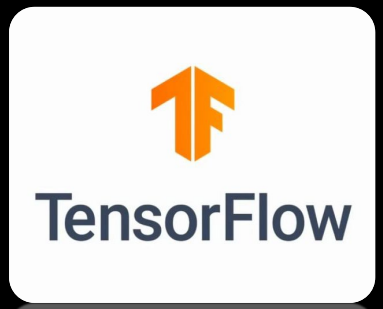
4次　参数w：0.296

……



源码：p13_backpropagation.py

TensorFlow 2



2019年 3 月　Tensorflow 2.0 测试版发布

2019年10月　Tensorflow 2.0 正式版发布

2020年 1 月　Tensorflow 2.1 发布

√ 张量（Tensor）：多维数组（列表） 阶：张量的维数

维数	阶	名字	例子
0-D	0	标量 scalar	s=1 2 3
1-D	1	向量 vector	v=[1, 2, 3]
2-D	2	矩阵 matrix	m=[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
n-D	n	张量 tensor	t=[[[

n个

张量可以表示0阶到n阶数组（列表）

数据类型

- ✓ **tf.int，tf.float ……**

  tf.int 32，tf.float 32，tf.float 64

  默认

- ✓ **tf.bool**

  tf.constant([True, False])

- ✓ **tf.string**

  tf.constant(“Hello, world!”)

创建一个张量：

`tf.constant(张量内容,dtype=数据类型(可选))`

如何创建一个Tensor

✓创建一个张量

tf.constant(张量内容，dtype=数据类型(可选))

```
import tensorflow as tf
a=tf.constant([1,5],dtype=tf.int64)
print(a)
print(a.dtype)
print(a.shape)
```

运行结果：

```
<tf.Tensor([1,5], shape=(2 , ) , dtype=int64)
<dtype: 'int64'>
(2,)
```

源码：p17_constant.py

如何创建一个Tensor

✓将numpy的数据类型转换为Tensor数据类型

tf. convert_to_tensor(数据名，dtype=数据类型(可选))

```
import tensorflow as tf
import numpy as np
a = np.arange(0, 5)
b = tf.convert_to_tensor( a,  dtype=tf.int64 )
print(a)
print(b)
```

运行结果：

```
[0 1 2 3 4]
tf.Tensor([0 1 2 3 4], shape=( 5 , ), dtype=int64)
```

源码：p18_convert_to_tensor.py

如何创建一个Tensor

✓创建全为0的张量
tf. zeros(维度)

✓创建全为1的张量
tf. ones(维度)

✓创建全为指定值的张量
tf. fill(维度，指定值)

维度：
一维 直接写个数
二维 用 [行，列]
多维 用 [n,m,j,k……]

```
a = tf.zeros([2, 3])
b = tf.ones(4)
c = tf.fill([2, 2], 9)
print(a)
print(b)
print(c)
```

运行结果：

```
tf.Tensor([[0. 0. 0.] [0. 0. 0.]], shape=(2, 3), dtype=float32)
tf.Tensor([1. 1. 1. 1.], shape=(4, ), dtype=float32)
tf.Tensor([[9 9] [9 9]], shape=(2, 2), dtype=int32)
```

源码：p19_zeros_ones_fill.py

如何创建一个Tensor

✓生成正态分布的随机数，默认均值为0，标准差为1
tf. random.normal (维度，mean=均值，stddev=标准差)

✓生成截断式正态分布的随机数
tf. random.truncated_normal (维度，mean=均值，stddev=标准差)

在tf.truncated_normal中如果随机生成数据的取值在（μ-2σ，μ+2σ）之外则重新进行生成，保证了生成值在均值附近。

μ：均值，　σ：标准差

标准差计算公式　$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}$

如何创建一个Tensor

```
d = tf.random.normal ([2, 2], mean=0.5, stddev=1)
print(d)

e = tf.random.truncated_normal ([2, 2], mean=0.5, stddev=1)
print(e)
```

运行结果：

```
tf.Tensor(
[[0.7925745 0.643315 ]
 [1.4752257 0.2533372]], shape=(2, 2), dtype=float32)

tf.Tensor(
[[ 1.3688478   1.0125661 ]
 [ 0.17475659 -0.02224463]], shape=(2, 2), dtype=float32)
```


源码：p21_random.normal.py

如何创建一个Tensor

✓生成均匀分布随机数　[minval, maxval)

tf. random. uniform(维度，minval=最小值，maxval=最大值)

```
f = tf.random.uniform([2, 2], minval=0, maxval=1)
print(f)
```

运行结果：

```
tf.Tensor(
[[0.28219545 0.15581512]
 [0.77972126 0.47817433]], shape=(2, 2), dtype=float32)
```

源码：p22_random.uniform.py

常用函数

✓强制tensor转换为该数据类型

tf.cast (张量名，dtype=数据类型)

✓计算张量维度上元素的最小值

tf.reduce_min (张量名)

✓计算张量维度上元素的最大值

tf.reduce_max (张量名)

```
x1 = tf.constant ([1., 2., 3.],
dtype=tf.float64)
print(x1)

x2 = tf.cast (x1, tf.int32)
print(x2)

print (tf.reduce_min(x2),
tf.reduce_max(x2))
```

运行结果：

```
tf.Tensor([1. 2. 3.], shape=(3,), dtype=float64)

tf.Tensor([1 2 3], shape=(3,), dtype=int32)

tf.Tensor(1, shape=(), dtype=int32)

tf.Tensor(3, shape=(), dtype=intt32)
```

源码：p23_cast_reduce_minmax.py

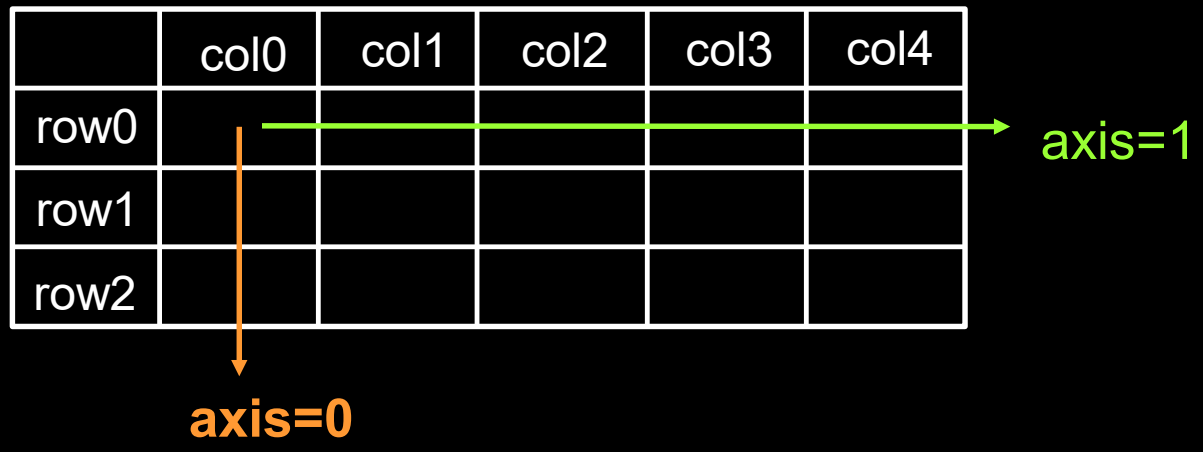
常用函数

理解axis

在一个二维张量或数组中，可以通过调整 axis 等于0或1 控制执行维度。

✓ axis=0代表跨行（经度，down)，而axis=1代表跨列（纬度，across)

✓ 如果不指定axis，则所有元素参与计算。



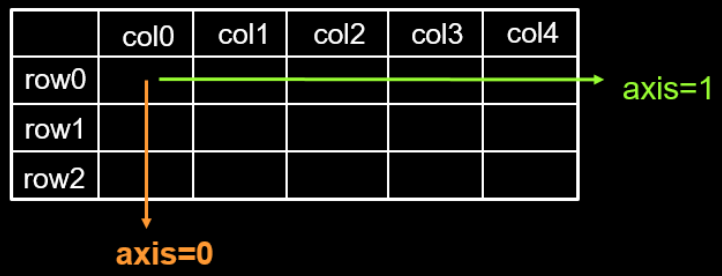
常用函数

✓ 计算张量沿着指定维度的平均值

tf.reduce_mean (张量名，axis=操作轴)

✓ 计算张量沿着指定维度的和

tf.reduce_sum (张量名，axis=操作轴)




```
x=tf.constant( [ [ 1, 2,  3],
                 [  2, 2, 3] ] )

print(x)

print(tf.reduce_mean( x ))

print(tf.reduce_sum( x, axis=1 ))
```

运行结果：

```
tf.Tensor(
[[1 2 3]
 [2 2 3]], shape=(2, 3), dtype=int32)

tf.Tensor(2, shape=(), dtype=int32)
```

因为要取整，所以为2

```
tf.Tensor([6 7], shape=(2,), dtype=int32)
```

源码：p25_reduce_meansum.py

常用函数　　tf.Variable

- ✓ **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。

tf.Variable(初始值)

w = tf.Variable(tf.random.normal([2, 2], mean=0, stddev=1))

tf.random.normal(维度,mean=均值,stddev=标准差)
生成正态分布的随机数，默认均值为0，标准差为1

常用函数　TensorFlow中的数学运算

加　减　乘　除

- ✓ 对应元素的四则运算：**tf.add，tf.subtract，tf.multiply，tf.divide**
- ✓ 平方、次方与开方： **tf.square，tf.pow，tf.sqrt**

计算某个张量的n次方：tf.pow(张量名, n次方数)

- ✓ 矩阵乘：**tf.matmul**

常用函数　对应元素的四则运算

✓ 实现两个张量的对应元素相加
tf.add (张量1，张量2)

✓ 实现两个张量的对应元素相减
tf.subtract (张量1，张量2)

✓ 实现两个张量的对应元素相乘
tf.multiply (张量1，张量2)

✓ 实现两个张量的对应元素相除
tf.divide (张量1，张量2)

只有维度相同的张量才可以做四则运算

常用函数　对应元素的四则运算

```
a = tf.ones([1, 3])

b = tf.fill([1, 3], 3.)

print(a)

print(b)

print(tf.add(a,b))

print(tf.subtract(a,b))

print(tf.multiply(a,b))

print(tf.divide(b,a))
```

运行结果：

```
tf.Tensor([[1. 1. 1.]], shape=(1, 3), dtype=float32)

tf.Tensor([[3. 3. 3.]], shape=(1, 3), dtype=float32

tf.Tensor([[4. 4. 4.]], shape=(1, 3), dtype=float32)

tf.Tensor([[-2. -2. -2.]], shape=(1, 3), dtype=float32)

tf.Tensor([[3. 3. 3.]], shape=(1, 3), dtype=float32)

tf.Tensor([[3. 3. 3.]], shape=(1, 3), dtype=float32)
```

源码：p29_add_subtract_multiply_divide.py

常用函数　平方、次方与开方

✓计算某个张量的平方
tf.square (张量名)
✓计算某个张量的n次方
tf.pow (张量名，n次方数)
✓计算某个张量的开方
tf.sqrt (张量名）

```
a = tf.fill([1, 2], 3.)

print(a)

print(tf.pow(a, 3))

print(tf.square(a))

print(tf.sqrt(a))
```

运行结果：

```
tf.Tensor([[3. 3.]], shape=(1, 2), dtype=float32)

tf.Tensor([[27. 27.]], shape=(1, 2), dtype=float32)

tf.Tensor([[9. 9.]], shape=(1, 2), dtype=float32)

tf.Tensor([[1.7320508 1.7320508]], shape=(1, 2), dtype=float32)
```

源码：p30_square_pow_sqrt.py

常用函数　矩阵乘 tf.matmul

✓实现两个矩阵的相乘
tf.matmul(矩阵1，矩阵2)

```
a = tf.ones([3, 2])
b = tf.fill([2, 3], 3.)
print(tf.matmul(a, b))
```

运行结果：

```
tf.Tensor(
[[6. 6. 6.]
 [6. 6. 6.]
 [6. 6. 6.]], shape=(3, 3), dtype=float32)
```

源码：p31_matmul.py

常用函数　　tf.data.Dataset.from_tensor_slices

✓切分传入张量的第一维度，生成输入特征/标签对，构建数据集

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

（Numpy和Tensor格式都可用该语句读入数据）

```
features = tf.constant([12,23,10,17])
labels = tf.constant([0, 1, 1, 0])
dataset = tf.data.Dataset.from_tensor_slices((features, labels))
print(dataset)
for element in dataset:
    print(element)
```

运行结果：

<TensorSliceDataset shapes: ((),()), types: (tf.int32, tf.int32))> （特征，标签）配对

(<tf.Tensor: id=9, shape=(), dtype=int32, numpy=12>, <tf.Tensor: id=10, shape=(), dtype=int32, numpy=0>)

(<tf.Tensor: id=11, shape=(), dtype=int32, numpy=23>, <tf.Tensor: id=12, shape=(), dtype=int32, numpy=1>)

(<tf.Tensor: id=13, shape=(), dtype=int32, numpy=10>, <tf.Tensor: id=14, shape=(), dtype=int32, numpy=1>)

(<tf.Tensor: id=15, shape=(), dtype=int32, numpy=17>, <tf.Tensor: id=16, shape=(), dtype=int32, numpy=0>)

源码：p33_from_tensor_slices.py

常用函数　　tf.GradientTape

✓with结构记录计算过程，gradient求出张量的梯度

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

```
with tf.GradientTape( ) as tape:
    w = tf.Variable(tf.constant(3.0))
    loss = tf.pow(w,2)
grad = tape.gradient(loss,w)
print(grad)
```

$$\frac{\partial w^2}{\partial w} = 2w = 2*3.0 = 6.0$$

运行结果：

```
tf.Tensor(6.0, shape=(), dtype=float32)
```

源码：p34_GradientTape.py

常用函数　　enumerate

- ✓ enumerate是python的内建函数，它可遍历每个元素(如列表、元组或字符串)，组合为：索引 元素，常在for循环中使用。

enumerate(列表名)

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

运行结果：

```
0 one
1 two
2 three
```

源码：p35_enumerate.py

常用函数　　tf.one_hot

- ✓ 独热编码（one-hot encoding）：在分类问题中，常用独热码做标签，标记类别：1表示是，0表示非。

（ 0狗尾草鸢尾　1杂色鸢尾　2弗吉尼亚鸢尾 ）

标　签：　1

独热码：　（0.　　1.　　0.）

常用函数　　tf.one_hot

✓ tf.one_hot()函数将待转换数据，转换为one-hot形式的数据输出。

tf.one_hot (待转换数据, depth=几分类)

```
classes = 3
labels = tf.constant([1,0,2])  # 输入的元素值最小为0，最大为2
output = tf.one_hot( labels, depth=classes )
print(output)
```

运行结果：

```
[[0. 1. 0.]
 [1. 0. 0.]
 [0. 0. 1.]], shape=(3, 3), dtype=float32)
```

源码：p37_one_hot.py

常用函数 tf.nn.softmax

5.8	4.0	1.2	0.2

*

-0.8	-0.34	-1.4
0.6	1.3	0.25
0.5	1.45	0.9
0.65	0.7	-1.2

\+

2.52	-3.1	5.62

=

0类鸢尾得分	1类鸢尾得分	2类鸢尾得分
1.01	2.01	-0.66

输出y

$$y \begin{cases} 1.01 \\ 2.01 \\ -0.66 \end{cases} \quad \frac{e^{y_i}}{\sum_{j=0}^{n} e^{y_i}}$$

$$\frac{e^{y_0}}{e^{y_0}+e^{y_1}+e^{y_2}} = \frac{2.75}{10.73} = 0.256$$

$$\frac{e^{y_1}}{e^{y_0}+e^{y_1}+e^{y_2}} = \frac{7.46}{10.73} = 0.695$$

$$\frac{e^{y_2}}{e^{y_0}+e^{y_1}+e^{y_2}} = \frac{0.52}{10.73} = 0.048$$

$e^{y_0} = e^{1.01} = 2.75$

$e^{y_1} = e^{2.01} = 7.46$

$e^{y_2} = e^{-0.66} = 0.52$

$e^{y_0} + e^{y_1} + e^{y_2} = 10.73$

$$\text{Softmax}(y_i) = \frac{e^{y_i}}{\sum_{j=0}^{n} e^{y_i}}$$

tf.nn.softmax(x) 使输出符合概率分布

源码：p38_forward.py

常用函数　　tf.nn.softmax

✓ 当n分类的n个输出（y_0，y_1, …… y_{n-1}）通过softmax( ) 函数，便符合概率分布了。

$$\forall x \ P(X=x) \in [0,1] \text{ 且 } \sum_x P(X=x) = 1$$

```
y = tf.constant ( [1.01, 2.01, -0.66] )
y_pro = tf.nn.softmax(y)
print("After softmax, y_pro is:", y_pro)
```

输出结果：

```
After softmax, y_pro is: tf.Tensor([0.25598174  0.69583046
0.0481878], shape=(3,), dtype=float32)
```

源码：p39_softmax.py

常用函数　　assign_sub

- ✓ 赋值操作，更新参数的值并返回。
- ✓ 调用assign_sub前，先用 tf.Variable 定义变量 w 为可训练（可自更新）。

w.assign_sub (w要自减的内容)

```
w = tf.Variable(4)
w.assign_sub(1)
print(w)
```

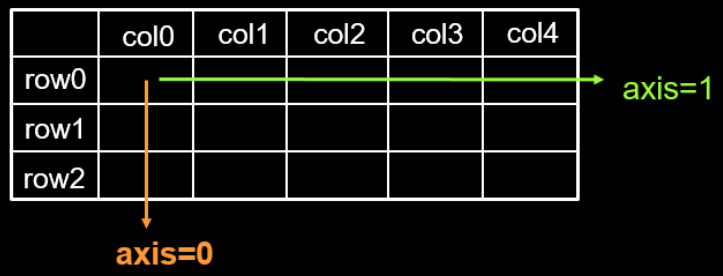
w -= 1 即　w = w - 1

运行结果：

```
<tf.Variable 'Variable:0' shape=() dtype=int32, numpy=3>
```

源码：p40_assign_sub.py

常用函数　　tf.argmax



✓ 返回张量沿指定维度最大值的索引

tf.argmax (张量名,axis=操作轴)

```
import numpy as np
test = np.array([[1, 2, 3], [2, 3, 4], [5, 4, 3], [8, 7, 2]])
print(test)
print( tf.argmax (test, axis=0))  # 返回每一列（经度）最大值的索引
print( tf.argmax (test, axis=1))  # 返回每一行（纬度）最大值的索引
```

运行结果：

```
[[1  2  3]
 [2  3  4]
 [5  4  3]
 [8  7  2]]

tf.Tensor([3 3 1], shape=(3,), dtype=int64)
tf.Tensor([2 2 0 0], shape=(4,), dtype=int64)
```

源码：p41_argmax.py

鸢尾花数据集（Iris）

✓数据集介绍

共有数据150组，每组包括花萼长、花萼宽、花瓣长、花瓣宽4个输入特征。同时给出了，这一组特征对应的鸢尾花类别。类别包括Setosa Iris（狗尾草鸢尾），Versicolour Iris（杂色鸢尾），Virginica Iris（弗吉尼亚鸢尾）三类，分别用数字0，1，2表示。

0狗尾草鸢尾

1杂色鸢尾

2弗吉尼亚鸢尾

鸢尾花数据集（Iris）

从sklearn包 datasets 读入数据集，语法为：

```
from sklearn.datasets import load_iris
x_data = datasets.load_iris().data    返回iris数据集所有输入特征
y_data = datasets.load_iris().target  返回iris数据集所有标签
```

源码： p42_datasets.load_iris.py

神经网络实现鸢尾花分类

✓ 准备数据
- 数据集读入
- 数据集乱序
- 生成训练集和测试集（即 x_train / y_train）
- 配成 （输入特征，标签） 对，每次读入一小撮（batch）

✓ 搭建网络
- 定义神经网路中所有可训练参数

✓ 参数优化
- 嵌套循环迭代，with结构更新参数，显示当前loss

✓ 测试效果
- 计算当前参数前向传播后的准确率，显示当前acc

✓ acc / loss可视化

神经网络实现鸢尾花分类

✓ 数据集读入

从sklearn包datasets 读入数据集：

```
from sklearn.datasets import datasets
x_data = datasets.load_iris().data    返回iris数据集所有输入特征
y_data = datasets.load_iris().target  返回iris数据集所有标签
```

✓ 数据集乱序

```
np.random.seed(116)  # 使用相同的seed，使输入特征/标签一一对应
np.random.shuffle(x_data)
np.random.seed(116)
np.random.shuffle(y_data)
tf.random.set_seed(116)
```

✓ 数据集分出永不相见的训练集和测试集

```
x_train = x_data[:-30]
y_train = y_data[:-30]
x_test = x_data[-30:]
y_test = y_data[-30:]
```

✓ 配成[输入特征，标签]对，每次喂入一小撮（batch）

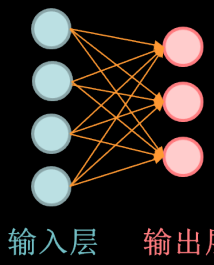
```
train_db = tf.data.Dataset.from_tensor_slices((x_train, y_train)).batch(32)
test_db = tf.data.Dataset.from_tensor_slices((x_test, y_test)).batch(32)
```


源码：p45_iris.py

神经网络实现鸢尾花分类

✓ 定义神经网路中所有可训练参数

```
w1 = tf.Variable(tf.random.truncated_normal([ 4,  3 ], stddev=0.1, seed=1))
b1 = tf.Variable(tf.random.truncated_normal([ 3 ], stddev=0.1, seed=1))
```



✓ 嵌套循环迭代，with结构更新参数，显示当前loss

```
for epoch in range(epoch):  #数据集级别迭代
    for step, (x_train, y_train) in enumerate(train_db):  #batch级别迭代
        with tf.GradientTape() as tape:  # 记录梯度信息
            前向传播过程计算y
            计算总loss
        grads = tape.gradient(loss, [ w1, b1 ])
        w1.assign_sub(lr * grads[0])  #参数自更新
        b1.assign_sub(lr * grads[1])
    print("Epoch {}, loss: {}".format(epoch, loss_all/4))
```

神经网络实现鸢尾花分类

✓ 计算当前参数前向传播后的准确率，显示当前acc

```
for x_test,  y_test in  test_db:
        y = tf.matmul(h, w) + b  # y为预测结果
        y = tf.nn.softmax(y)       # y符合概率分布
        pred = tf.argmax(y, axis=1)  # 返回y中最大值的索引，即预测的分类
        pred = tf.cast(pred, dtype=y_test.dtype) #调整数据类型与标签一致
        correct = tf.cast(tf.equal(pred, y_test), dtype=tf.int32)
        correct = tf.reduce_sum (correct) # 将每个batch的correct数加起来
        total_correct += int (correct) # 将所有batch中的correct数加起来
        total_number += x_test.shape [0]
acc = total_correct / total_number
print("test_acc:", acc)
```

✓ acc / loss可视化

```
plt.title('Acc Curve')  # 图片标题
plt.xlabel('Epoch')  # x轴名称
plt.ylabel('Acc')  # y轴名称
plt.plot(test_acc, label="$Accuracy$")  # 逐点画出test_acc值并连线
plt.legend()
plt.show()
```