从本篇文章开始，作者正式开始研究Python深度学习、神经网络及人工智能相关知识。前一篇文章讲解了TensorFlow的安装过程和神经网络基础概念。这篇文章将分享TensorFlow基础并介绍一元直线预测的案例，主要结合作者之前的博客和"莫烦大神"的视频介绍，后面随着深入会讲解具体的项目及应用。

基础性文章，希望对您有所帮助，如果文章中存在错误或不足之处，还请海涵~同时自己也是人工智能的菜鸟，希望大家能与我在这一笔一划的博客中成长起来。

文章目录

同时推荐前面作者另外三个Python系列文章。从2014年开始，作者主要写了三个Python系列文章，分别是基础知识、网络爬虫和数据分析。2018年陆续增加了Python图像识别和Python人工智能专栏。

- Python基础知识系列：Pythonj基础知识学习与提升
- Python网络爬虫系列：Python爬虫之Selenium+Phantomjs+CasperJS
- Python数据分析系列：知识图谱、web数据挖掘及NLP
- Python图像识别系列：Python图像处理及图像识别
- Python人工智能系列：Python人工智能及知识图谱实战

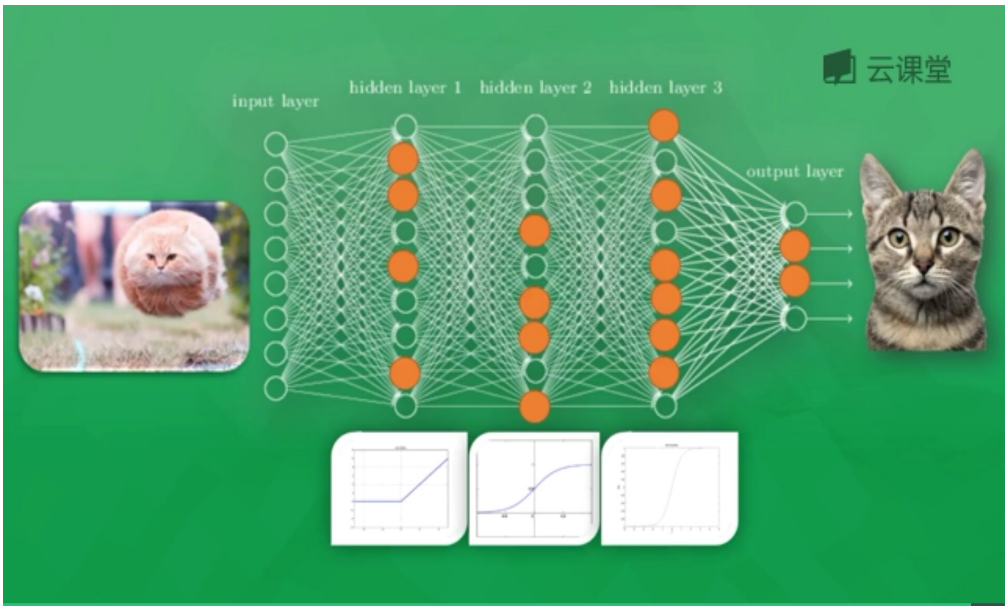
前文：
[Python人工智能] 一.TensorFlow2.0环境搭建及神经网络入门

代码下载地址：https://github.com/eastmountyxz/AI-for-TensorFlow

一.神经网络前言

如下图所示，通过该神经网络识别动物猫或狗，共包括输入层（Input Layer）、隐藏层3层（Hidden Layer）和输出层（Output Layer）。其中每一个神经元都有一个激励函

数，被激励的神经元传递的信息最有价值，它也决定最后的输出结果，经过海量数据的训练，最终神经网络将可以用于识别猫或狗。

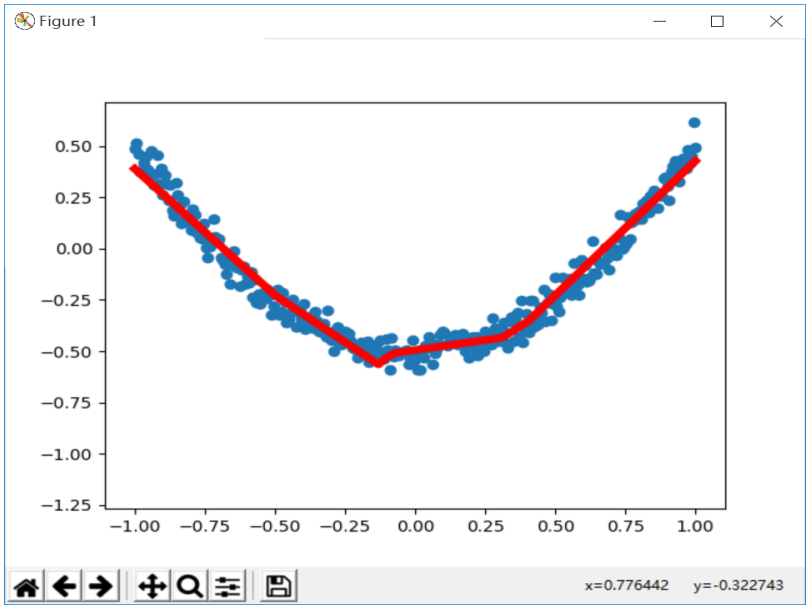


激励函数相当于一个过滤器或激励器，它把特有的信息或特征激活，常见的激活函数包括softplus、sigmoid、relu、softmax、elu、tanh等。对于隐藏层，我们可以使用relu、tanh、softplus等非线性关系；对于分类问题，我们可以使用sigmoid（值越小越接近于0，值越大越接近于1）、softmax函数，对每个类求概率，最后以最大的概率作为结果；对于回归问题，可以使用线性函数（linear function）来实验。

神经网络中的神经元中都有激励函数（activation function），常见的激励函数参考维基百科：https://en.wikipedia.org/wiki/Activation_function

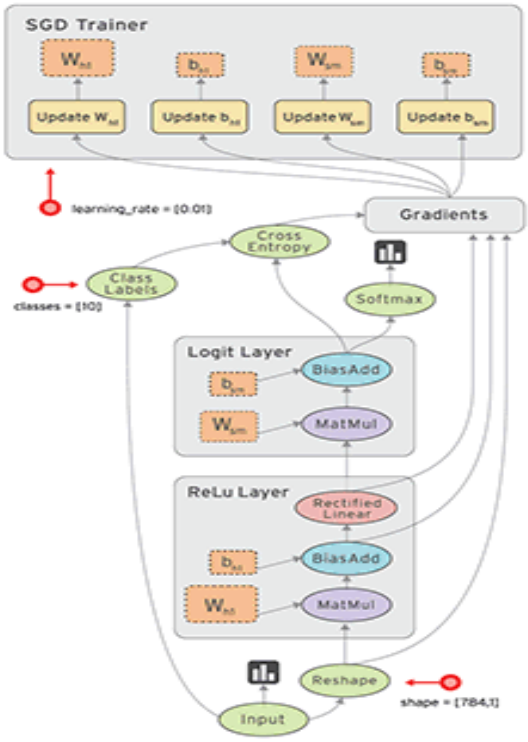
Name	Plot	Equation	Derivative (with respect to *x*)
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ [1]	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Softsign [7][8]		$f(x) = \frac{x}{1+\lvert x\rvert}$	$f'(x) = \frac{1}{(1+\lvert x\rvert)^2}$
Inverse square root unit (ISRU)[9]		$f(x) = \frac{x}{\sqrt{1+\alpha x^2}}$	$f'(x) = \left(\frac{1}{\sqrt{1+\alpha x^2}}\right)^3$

现假设输入x代表一串数字，输出y代表一个二次函数，蓝色的点表示数据集，红线表示神经网络所学习到的一条曲线，并代表我们的数据。最初这条曲线可能是弯弯曲曲的，随着不断地训练和学习，这条曲线会逐渐拟合到我的数据上来，该红线越符合数据的趋势，就表示神经网络学习到的东西越多，亦越能去预测其趋势。



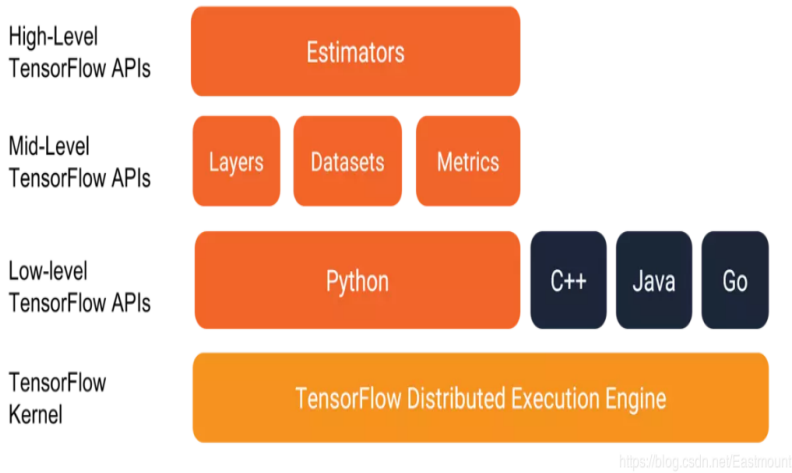
二.TensorFlow结构及工作原理

Tensorflow是一个使用数据流图（data flow graphs）技术来进行数值计算的开源软件库。数据流图是是一个有向图，使用节点（一般用圆形或方形描述，表示一个数学操作或数据输入的起点和数据输出的终点）和线（表示数字、矩阵或Tensor张量）来描述数学计算。数据流图可以方便的将各个节点分配到不同的计算设备上完成异步并行计算，非常适合大规模的机器学习应用[7]。如下图所示，通过Gradients不断学习改进我们的权重W和偏置b，从而提升准确度。

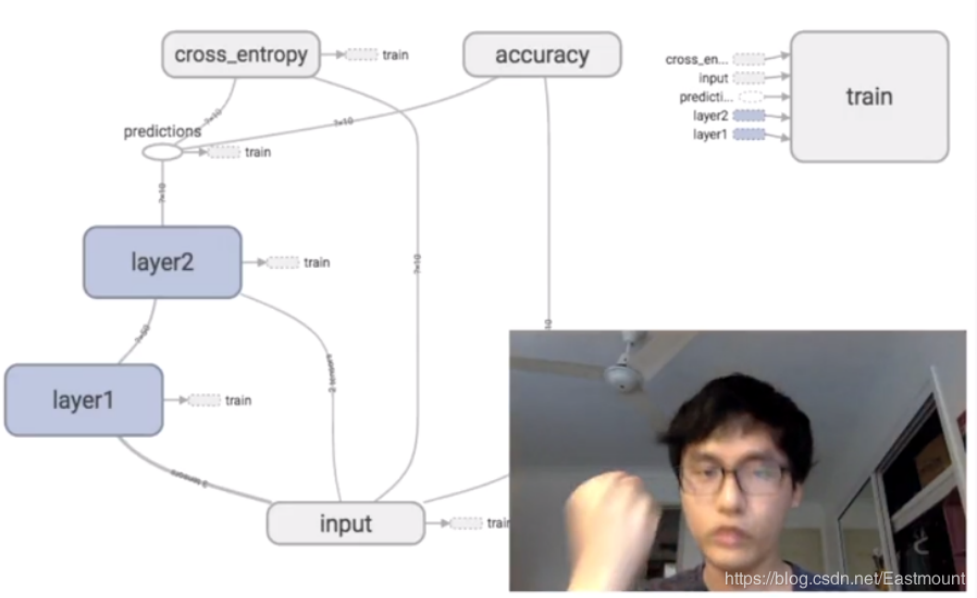


TensorFlow支持各种异构的平台，支持多CPU/GPU、服务器、移动设备，具有良好的跨平台的特性；TensorFlow架构灵活，能够支持各种网络模型，具有良好的通用性。此

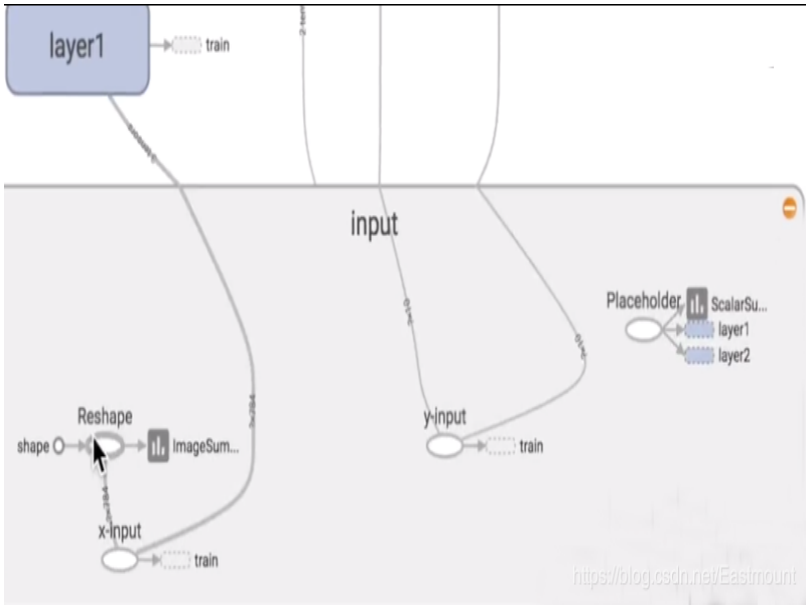
外，TensorFlow内核采用C/C++开发，并提供了C++、Python、Java、Go语言的Client API。其架构灵活，能够支持各种网络模型，具有良好的通用性和可扩展性。tensorflow.js支持在web端使用webGL运行GPU训练深度学习模型，支持在IOS、Android系统中加载运行机器学习模型。其基本结构如下图所示：

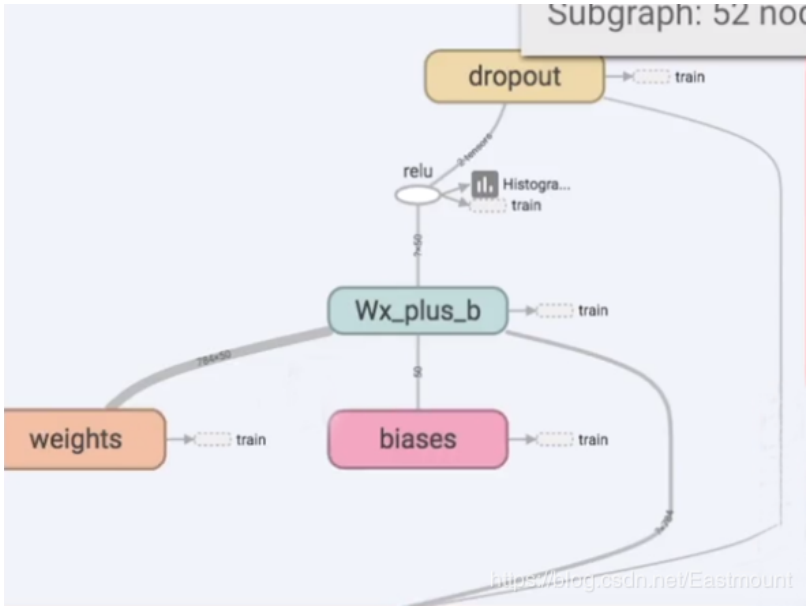


TensorFlow怎么处理数据呢？接下来分享莫烦老师解析TensorFlow的工作原理。
下图是一个包含输入层（input layer）、隐藏层（hidden layer）和输出层（output layer）的典型神经网络结构。



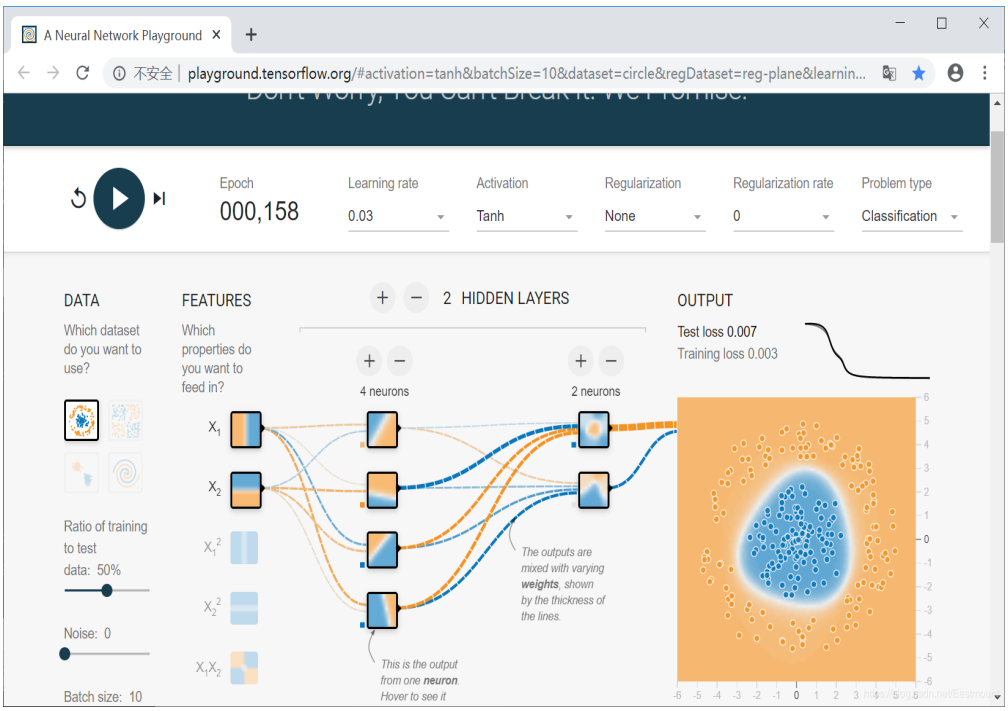
点开输入层input，发现它包含 x-input 和 y-input；然后隐藏层包含了权重（weights）和偏置（biases），以及一个激励函数relu，后面会深入分享。





TensorFlow的第一步是做什么呢？它就是需要建立这样一个结构，然后把数据放到这个结构中，让TensorFlow自己运行，通过不断地训练和学习，修改参数来输出准确的预测结果。TensorFlow中文翻译是“向量在这个结构中飞”，这也是TensorFlow的基本含义。

最后补充一个在线的神经网络模拟器，大家可以试着去运行，看看神经网络的工作原理及参数调整。地址为：http://playground.tensorflow.org/



三.TensorFlow实现一元直线预测

接着补充一个TensorFlow预测线性直线（y = 0.1 * x + 0.3）的案例。

1.首先定义随机生成的100个数字和线性直线

TensorFlow中比较常见的格式是float32，这里也进行相关的设置。同时，预测的权重weight为0.1，偏置biases为0.3

```
x_data = np.random.rand(100).astype(np.float32)
print(x_data)
y_data = x_data * 0.1 + 0.3
print(y_data)
```

输出结果为：

```
[0.42128456 0.3590797  0.31952253 0.54071575 0.4098252  0.87536865
 0.58871204 0.37161928 0.30826327 0.94290715 0.40412208 0.47935355
 ...
 0.00465394 0.0929272  0.18561055 0.8577836  0.6737635  0.04606532
 0.8738483  0.9900948  0.13116711 0.299359  ]
[0.34212846 0.335908   0.33195227 0.3540716  0.34098253 0.38753688
 0.35887122 0.33716193 0.33082634 0.39429075 0.34041223 0.34793538
...
 0.3004654  0.30929273 0.31856108 0.38577837 0.36737636 0.30460656
 0.38738483 0.3990095  0.31311673 0.3299359 ]
```

2.创建TensorFlow结构，定义权重Weights和偏置biases

权重为tf.Variable参数变量，随机生成的一维结构数列，其范围是-1.0到1.0；偏置biases初始值为0。TensorFlow学习的目的是从初始值不断提升到目标0.1（Weights）0和0.3（biases）。

```
Weights = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
print(Weights)
biases = tf.Variable(tf.zeros([1]))
print(biases)
```

输出结果为：

```
<tf.Variable 'Variable_28:0' shape=(1,) dtype=float32_ref>
<tf.Variable 'Variable_29:0' shape=(1,) dtype=float32_ref>
```

3.定义预测值y和损失函数

损失函数为预测的y和实际值y_data的最小方差。

```
y = Weights * x_data + biases
loss = tf.reduce_mean(tf.square(y-y_data))
```

4. 构建神经网络优化器（梯度下降优化函数）

这里的优化器为GradientDescentOptimizer，通过优化器减少误差，每一步训练减少误差并提升参数准确度。学习效率设置为0.5，一般是一个小于1的数字。

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.5)
train = optimizer.minimize(loss)
```

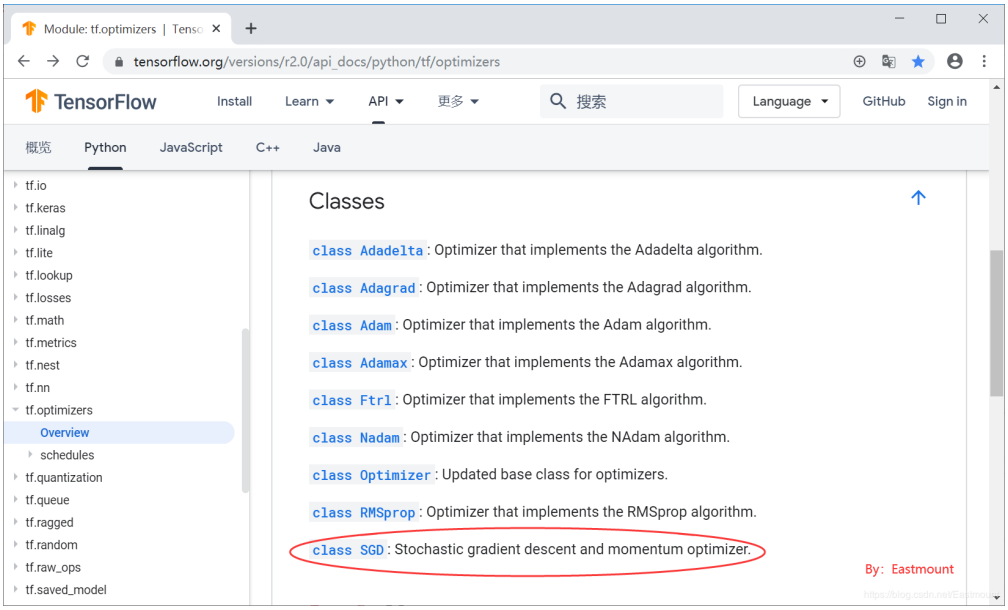
注意，TensorFlow2.0和TensorFlow1.0的一些方法都有改动。举个例子，在预测加利福尼亚的房价案例中，tf.train.GradientDescentOptimizer这个类已经不能使用，是train这个模块被2.0整体移除，对比如下。

- TensorFlow1.0：optimizer = tf.train.GradientDescentOptimizer(0.5)
- TensorFlow2.0：optimizer = tf.optimizers.SGD(learning_rate=0.5)

更多方法请查看官方文档，作者本想以TensorFlow2.0讲解，但发现太多改动，还是准备先以TensorFlow1.0作为入门，后续随着自己学习和理解深入，再分享TensorFlow2.0的代码，还请原谅。

TF2.0官方：https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/optimizers

> In TensorFlow 2.0, Keras became the default high-level API, and optimizer functions migrated from tf.keras.optimizers into separate API called tf.optimizers. They inherit from Keras class Optimizer. Relevant functions from tf.train aren't included into TF 2.0. So to access GradientDescentOptimizer, call tf.optimizers.SGD



5.初始化变量

> init = tf.global_variables_initializer()

6.定义Session并训练，每隔20次输出结果

```
# 定义Session
sess = tf.Session()
# 运行时Session就像一个指针 指向要处理的位置并激活
sess.run(init)t
# 训练201次
for n in range(201):
    # 训练并且每隔20次输出结果
    sess.run(train)
    if n % 20 == 0:
```

```
        # 最佳拟合结果 W: [0.100], b: [0.300]
        print(n, sess.run(Weights), sess.run(biases))
```

输出结果如下图所示，每隔20次训练，就输出一次它的参数——weight权重和biases偏置。我们想要做的是预测线性直线，最初的时候，计算值和与测试值会有很大的差别，神经网络需要做的就是不断缩小预测值y和实际值y_data的差别。最后经过200次训练后，已经非常接近这些散点了。

```
0 [-0.17100081] [0.60644317]
20 [0.00904198] [0.34798568]
40 [0.07555631] [0.31289548]
60 [0.09343112] [0.3034655]
80 [0.09823472] [0.3009313]
100 [0.09952562] [0.3002503]
120 [0.09987252] [0.30006728]
140 [0.09996576] [0.30001807]
160 [0.09999081] [0.30000487]
180 [0.09999753] [0.30000132]
200 [0.09999932] [0.30000037]
```

7.可视化显示

> pre = x_data * sess.run(Weights) + sess.run(biases)
> plt.scatter(x_data, y_data)
> plt.plot(x_data, pre, ‘r-’)
> plt.show()

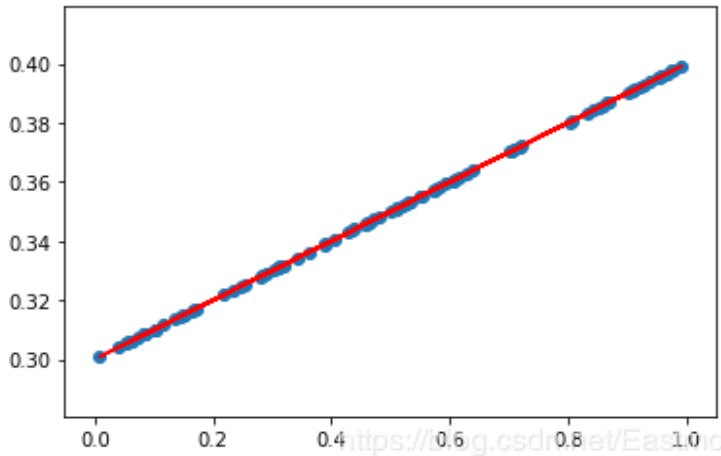
完整代码如下所示（精简版）：

```
# -*- coding: utf-8 -*-
"""
Created on 2019-11-30 下午6点 写于武汉大学

@author: Eastmount CSDN YXZ
"""
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import time

x_data = np.random.rand(100).astype(np.float32)
```

```
y_data = x_data * 0.1 + 0.3 #权重0.1 偏置0.3

#------------------开始创建tensorflow结构------------------
# 权重和偏置
Weights = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
biases = tf.Variable(tf.zeros([1]))

# 预测值y
y = Weights * x_data + biases

# 损失函数
loss = tf.reduce_mean(tf.square(y-y_data))

# 建立神经网络优化器
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.5) #学习效率
train = optimizer.minimize(loss)

# 初始化变量
init = tf.global_variables_initializer()
#------------------结束创建tensorflow结构------------------

# 定义Session
sess = tf.Session()

# 运行时Session就像一个指针 指向要处理的位置并激活
sess.run(init)

# 训练并且每隔20次输出结果
for n in range(201):
    sess.run(train)
    if n % 20 == 0:
        print(n, sess.run(Weights), sess.run(biases))
        pre = x_data * sess.run(Weights) + sess.run(biases)

# 可视化分析
plt.scatter(x_data, y_data)
plt.plot(x_data, pre, 'r-')
plt.show()
```

输出结果如下图所示，权重从最初的随机数0.3913201，逐渐优化成目标0.1；偏置从最初的随机数0.19479582，逐步优化到0.3，通过不断地学习和训练。

```
0 [0.3913201] [0.19479582]
20 [0.16674897] [0.26473704]
40 [0.11642814] [0.29132116]
60 [0.10404326] [0.297864]
80 [0.10099513] [0.2994743]
100 [0.10024493] [0.2998706]
120 [0.10006028] [0.29996818]
140 [0.10001485] [0.29999217]
160 [0.10000364] [0.29999807]
180 [0.1000009] [0.29999954]
200 [0.10000023] [0.2999999]
```

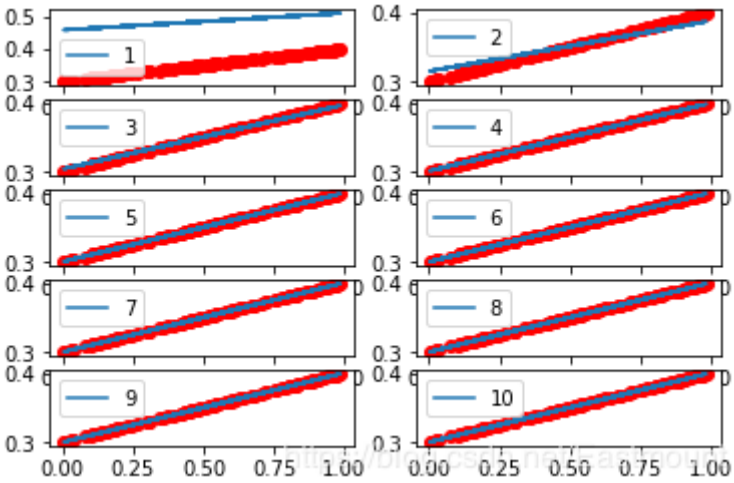


8.可视化对比多张效果图的完整代码

```
# -*- coding: utf-8 -*-
"""
Created on Sun Dec  1 20:00:14 2019

@author: xiuzhang
"""


# -*- coding: utf-8 -*-
"""
Created on 2019-11-30 下午6点 写于武汉大学

@author: Eastmount CSDN YXZ
"""
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

x_data = np.random.rand(100).astype(np.float32)
y_data = x_data * 0.1 + 0.3 #权重0.1 偏置0.3
```

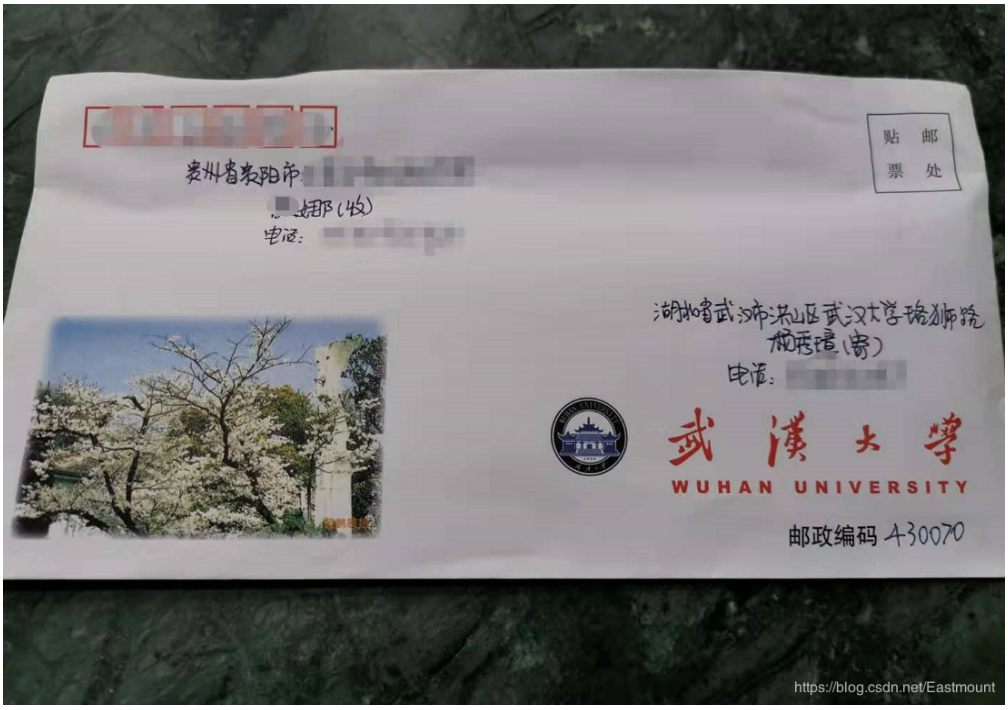
```
#------------------开始创建tensorflow结构------------------
# 权重和偏置
Weights = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
biases = tf.Variable(tf.zeros([1]))

# 预测值y
y = Weights * x_data + biases

# 损失函数
loss = tf.reduce_mean(tf.square(y-y_data))

# 建立神经网络优化器
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.5) #学习效率
train = optimizer.minimize(loss)

# 初始化变量
init = tf.global_variables_initializer()
#------------------结束创建tensorflow结构------------------

# 可视化
plt.plot(x_data, y_data,'ro', marker='^', c='blue',label='original_data')
plt.legend()
plt.show()


# 定义Session
sess = tf.Session()

# 运行时Session就像一个指针 指向要处理的位置并激活
sess.run(init)

# 训练并且每隔20次输出结果
k = 0
for n in range(200):
    sess.run(train)
    if n % 20 == 0:
        print(n, sess.run(Weights), sess.run(biases))
        pre = x_data * sess.run(Weights) + sess.run(biases)

        # 可视化分析
        k = k + 1
        plt.subplot(5, 2, k)
        plt.plot(x_data, y_data, 'ro')
        plt.plot(x_data, pre, label=k)
        plt.legend()

plt.show()
```

输出结果如下图所示：



四.总结

写到这里，这篇基础性的TensorFlow文章就讲述完毕。它通过不断地训练和学习，将预测结果与实际曲线y=0.1*x+0.3向匹配，这是非常基础的一篇深度学习文章，同时文章中存在错误或不足之处，还请海涵~同时，作为人工智能的菜鸟，我希望自己能不断进步并深入，后续将它应用于图像识别、网络安全、对抗样本等领域，一起加油！女神晚上收到了又一封家书，道不尽的思恋，读博不易，写文也不易，且行且珍惜。



(By:Eastmount 2019-11-30 下午6点 写于珞珈山 http://blog.csdn.net/eastmount/)

作者theano人工智能系列：

[Python人工智能] 一.神经网络入门及theano基础代码讲解

[Python人工智能] 二.theano实现回归神经网络分析

[Python人工智能] 三.theano实现分类神经网络及机器学习基础

[Python人工智能] 四.神经网络和深度学习入门知识

[Python人工智能] 五.theano实现神经网络正规化Regularization处理

[Python人工智能] 六.神经网络的评价指标、特征标准化和特征选择

[Python人工智能] 七.加速神经网络、激励函数和过拟合

参考文献：

[1] 神经网络和机器学习基础入门分享 - 作者的文章

[2] 斯坦福机器学习视频NG教授： https://class.coursera.org/ml/class/index

[3] 书籍《游戏开发中的人工智能》、《游戏编程中的人工智能技术》

[4] 网易云莫烦老师视频（强推）：https://study.163.com/course/courseLearn.htm?courseId=1003209007

[5] 神经网络激励函数 - deeplearning

[6] tensorflow架构 - NoMorningstar

[7] 《TensorFlow2.0》低阶 api 入门 - GumKey

[8] https://tensorflow.google.cn/overview/

[9] http://playground.tensorflow.org/

[10] https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/optimizers