**基于Linux环境的简单python语法以及tensorflow机器学习**

**学习笔记**

一、环境安装

首先在系统中搭建虚拟机，至于Linux这里使用Ubantu18.0系统。

1.系统搭建

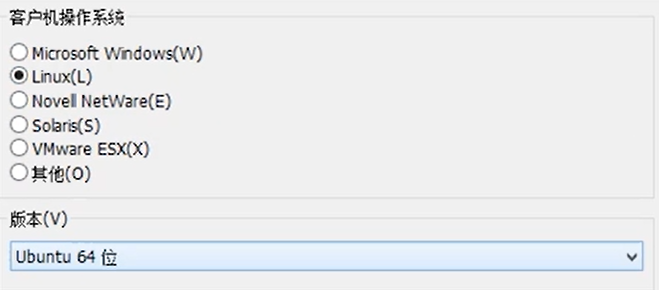
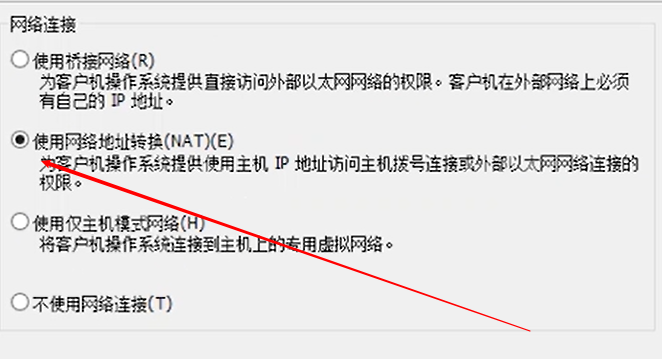
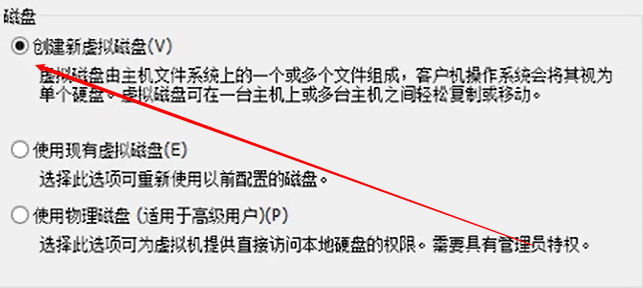
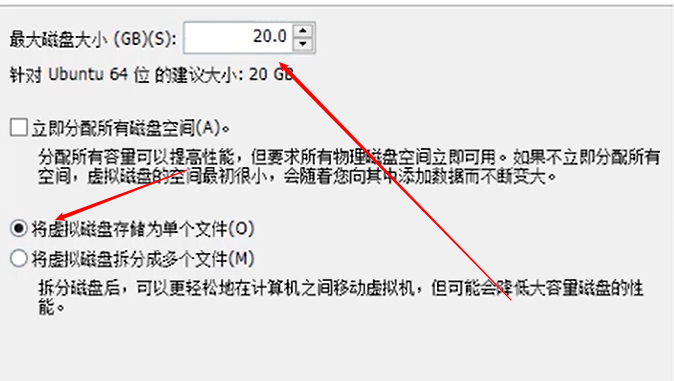
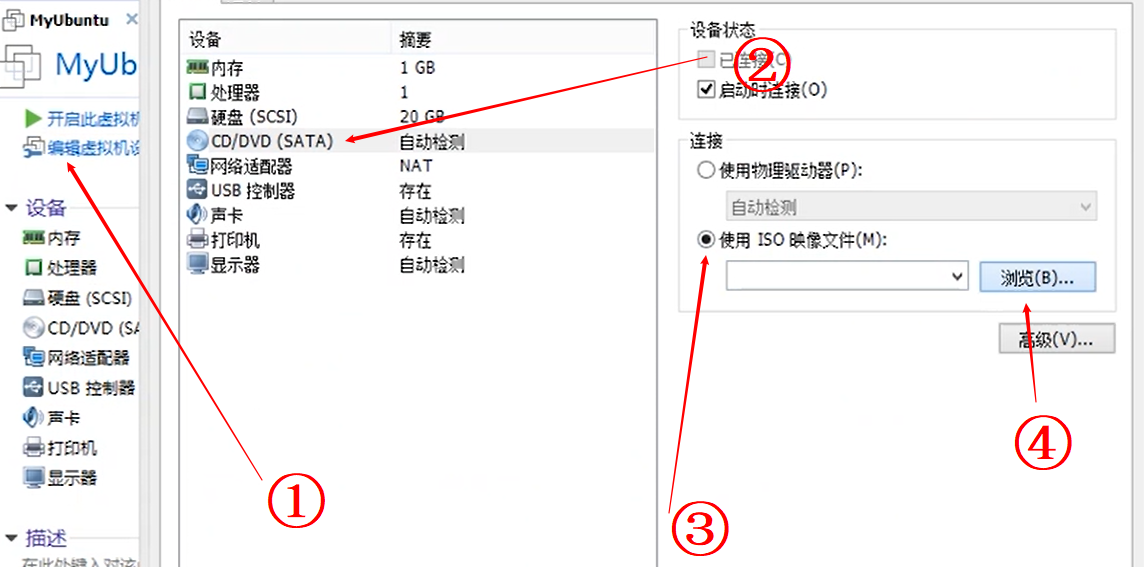
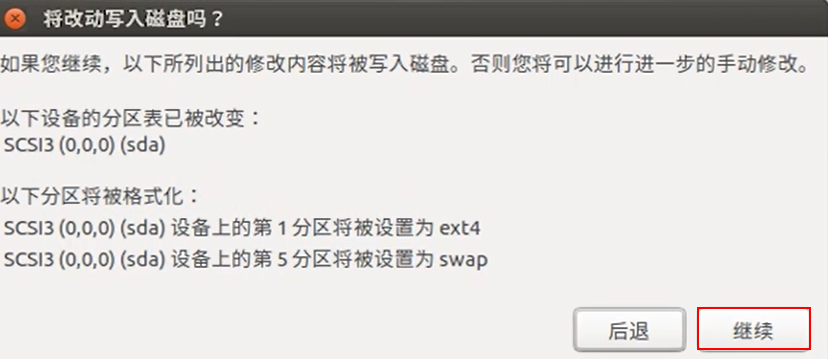
虚拟机与Linux镜像下载地址：

https://mp.weixin.qq.com/s/ti\_E6lAiljmVYM4\_9TUKwg （百度网盘）

虚拟机的具体安装介绍详见网页内容。但镜像安装请不要按照他的方式，因为他的方法虚拟机没办法联网，后面的pip就没法用了。

操作如下：



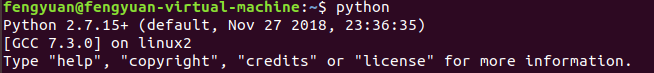
1.环境搭建

首先先在终端中输入python并回车查看python的版本号（最好是python2.7，因为这个版本的支持tensorflow）如果系统没有下载python，在终端中输入

**sudo apt-get install python**

即可。

完毕后记得输入python检测是否安装成功，成功会显示版本号



接下来安装pip，如果电脑里是python3直接执行

**sudo python3 get-pip.py**

等待一下就好了。

最后安装tensorflow。由于之前直接使用pip install tensorflow网速太慢，而且安装总提示错误（我也不懂是什么错），于是我搞了个镜像网站下载，我们只需要在终端输入

**pip install** [**https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.12.0-cp27-none-linux\_x86\_64.whl**](https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.12.0-cp27-none-linux_x86_64.whl)

等待一下就安装好了。

最后检查一下安装是否成功

在pyton命令行下，输入

**import tensorflow as tf**

**tf.\_\_version\_\_**

如果显示版本号证明安装成功。

若不成功自行百度8（笑）

二、系统简单命令

下面列出一些常用的linux指令。但学习python不需要全部掌握。序号被标红色的要求必须掌握。

系统中可能比较频繁用到的文件夹有bin,dev,etc,home,mnt,proc,root,sbin,tmp,usr,var等。

**1.关机**

1.**shutdown -h now**（正常关机）

2.**halt**（关闭内存）

3.**init 0**

**su<pr>**  获取本机root，然后输入密码，指令行有#出现即成功。

**2. 列出所有目录和文件夹的名称**

ls 路径

① ~~路径：

1.相对路径：./表示当前目录（和不写意思一样）,../（上一级目录下）

2.绝对路径 直接用/

例：**ls /root**（这里没有参照物）

② **ls -l** 路径 以详细列表的形式进行展示

③ **ls -la** 路径 隐藏文件/文件夹也会展示, 在linux中隐藏文档一般以.开头

ls列出的颜色结果说明

1.蓝色表示文件夹

2.黑色表示文件

3.绿色表示其权限为拥有所有权限

**3.打印当前工作目录绝对路径**

**pwd**  全称（print working directory）

**4.用于切换当前的工作目录**

**cd**  全称（change directory）

用法：cd+路径

~用于表示当前用户的家目录/root

**cd ~**

**5.创建目录**

**mkdir**  全称（make directory）

① 在指定路径下创建一个文件夹x

**mkdir/root/x**

② 一步到位：-p

**mkdir -p /root/x/a/b/c/d**

如果不加-p会报错

③ **mkdir a b c** 一次性创建多个目录

**6.创建文件**

**touch**

1. **touch+文件路径**

例：**touch Linux.txt**

② 也可以同时创建多个文件（同mkdir的③）

③ **touch /home/xxx/linux.txt**  //在/home/xxx下创建linux.txt文件

../等语法也同样适用

注：不可以在不存在的目录下创建文件

**7. 复制文件或文件夹到指定的位置**

**cp**

用法：cp+被复制的文档路径+文档被保存/复制的路径

①复制文件

linux在复制的过程中是可以改名的

例：**cp linux.txt /home/xxx/linux.txt**

将当前目录下的linux.txt复制到/home/xxx下

**cp linux.txt /home/xxx/linux1.txt**

将当前目录下的linux.txt复制到/home/xxx下并更名为linux1,txt

②复制文件夹

当使用cp命令进行文件夹复制的时候要添加选项-r，-r表示递归的意思

**cp -r /linux/ /home/xxx/linux** 复制当前目录下的linux文件夹到/home/xxx中

**8.移动文档到新的位置**

**mv**

用法：mv+需要移动的文档路径+需要保存的位置路径

1. 移动文件

**mv linux1.txt /linux.txt** 移动当前目录下的linux.txt到根目录中，也可以通过这个操作改名。后面不加文件名也可以，默认为不改

例 ：

直接  **mv/home/linux123/xxx/ /**

移动/home/linux123下的xxx文件夹及其内的东西到/root下

1. 移动文件夹
2. 重命名

**mv /xxx /xxxx**  //将根目录下的xxx文件改名为xxxx

**9. 移除**

**rm**  全称remove

1. rm+选项+需要移除文档路径

**rm/xxx**  后 输入y或yes 删除；输入n或no 不删除；

若不想频繁的确认 可以在指令中添加-f（force）强制直接删除，rm -f

② 删除文件夹 在指令中直接加入-rf选项（其中-r表示递归）

1. **rm -rf xxx（空格）xxxx**  可以批量删除多个
2. 若删除一个目录下有公共特性的文档，用

**rm -f xxx\***  删除当前目录下以xxx开头的所有文件，其中\*称为通配符

**10.文本编辑器**

**vim**

作用：使用vim来打开文件

用法：vim+文件的路径

进入后按**i**进行文本编辑，退出时

**ESC+：+q**  退出

**ESC+：+wq**  保存并退出

**ESC+：+！q**  不保存退出

注：如果计算机里没有vim要手动下载，这里用

**sudo apt install vim**

**11.输出重定向**

解释：一般的命令输出都会显示在终端中，有时候需要经一些命令的执行结果保存到文件中进行后续的分析统计，这时候需要用输出重定向技术。

>，覆盖输出 覆盖掉原来的内容

>>，追加输出，在原始内容末未继续添加

正常执行的指令 >/>> 文件的路径

注意，文件可以不存在，不存在自动新建

例 **ls -la >ls.txt**

**12.直接打开文件**

**cat**

作用1：cat用于直接打开文件

用法：cat+文件的路径（不用以命令形式退出）

作用2：cat用于对文件进行合并

用法：cat+合并路径1+合并路径2 …+文件路径n> 合并之后的文件路径

例

**cat /home/linux123/linux.txt /home/linux123/linux1.txt /home/linux123/linux10.txt > hebing.txt**

合并linux.txt和linux1.txt和linux10.txt

在cat hebing.txt这里要配合输出重定向。

**13.查看磁盘容量**

**df**

作用 查看磁盘空间

**df -h**  -h以较高可读性输出

在输出中，

Avail 可用内存 Used%使用百分比 Mounted on 挂载点（盘挂在在哪里）

**14.查看内存使用情况**

**free**

**free -m**  以M（兆）为单位显示 默认是为K

windows中除C盘以外磁盘可以充当临时内存使用（但速度较慢） 名为Ready Boost

这里用swap表示。

free的输出第二行才为真正的剩余空间。

**15.查看一个文件的前n行**

**head**

**head -n** 文件路径，n表示数字

如果不指定n，则默认显示前10行

**16. 查看一个文件的后n行**

**tail**

1. **tail -n** 文件路径 查看一个文件的后n行
2. **tail -f** 文件路径 通过tail查看一个文件的动态变化

注意：查看的动态内容不能是手动更改的

该命令一般用来查看系统日志较多

退出时按q

**17. 查看文件**

**less**

这里less是以较少的内容进行输出，搭配辅助功能键可以查看更多。

用法：less+需要查看的文件的路径

辅助键有数字+回车，空格（翻页），方向键

**18. 统计文件内容信息 （行数，单词数，字节数）**

**wc**

**wc -lwc** 需要统计的文件路径

-l(lines) 行数

-w(words)单词数

-c(bytes) 字节数

**19. 操作时间和日期（读取，设置）**

**date**

1. **date**  输出形式：年月日，星期，时分秒，是CST当地时间
2. **date +%F**  输出形式 ：年月日，等价于**date -%Y-%m-%d** ，小y是两位年
3. **date "+%F +%T"**  F和+之间有空格，引号让年月日时分秒成为一个不可分割的整体，等价于  **date "+%Y-%m-%d %H:%M:%S"**
4. 获取之前或之后的某个时间（备份）

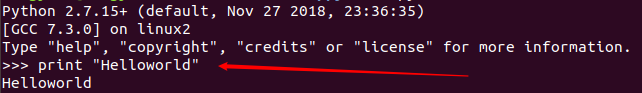
**……**

语法过多，能力有限，先列举到这里，有时间我会补上。

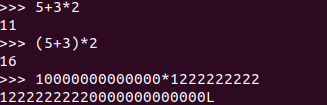
三、python简单语法

这里只介绍简单的语法，默认有其他语言的编程基础，简单的语法一笔带过了。

1. 终端输出



1. 简单计算



1. 输出变量

这里用%s表示占位符，随后会在后面声明占位的变量名称。

例：



1. 列表，元组，字典
2. 列表：**a=[1,2,3,4,5]，b=[“aaa”,“bbb”]，c=[1,2,“3”,a]**等都是列表

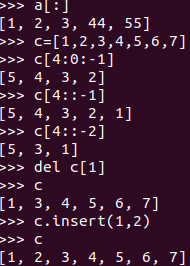
列表索引：“列表名[起:止]”表示从列表中切出相应的元素，注意是前闭后开，也可用“列表名[起:止:切片]”表示从起到止只展示切片位处的元素。不表示起则默认从头开始，不表示止同理，不表示切片时若起在止前则为1，反之为-1。此外还有insert，del等操作，具体是

**列表名.insert(位置，插入元素)**

**del 列表名[位置]**

如下是几个例子。





1. 元组

特性：一旦定义以后就不能再改变

例

f = (1,2,3)

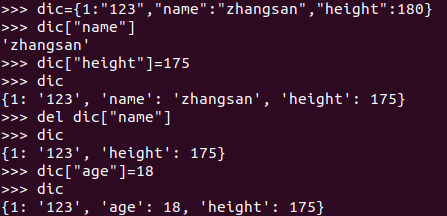
1. 字典

用大括号括起来的n个键值对

例dic={1:”zz”,”abc”:100,200:10}

其中1和zz就是一个键值对，1是键，zz是值。

三种操作：添加，删除，修改



5. 条件语句

这个时候要用vim编写几行代码，直接在终端编写很麻烦，这个时候就要用到vim。

这个时候按ctrl+D可以先退出python编译环境，也可以在终端输入**exit()**。在系统中新建一个.py文件，具体操作

**ctrl+D**

**mkdir python**

**cd python**

**pwd**

**vim a.py**（这里默认文件名是a了，可自行更改）

进入以后按i就可以编辑内容了

现在说条件语句的三中表示形式，这里注意行间的缩进以及英文冒号

**①**

**if 条件成立:**

**任务**

**②**

**if 条件1成立:**

**任务1**

**else:**

**任务2**

**③**

**if 条件1成立:**

**任务1**

**elif 条件2成立:**

**任务2**

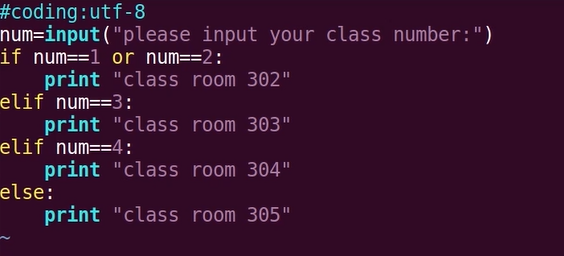
**…………**

**else:**

**任务n**

这里有个问题必须要说一下，如果代码中使用了中文，要在开头处加上**#coding:utf-8**否则会报错。

例子：



这段代码第二行的意思是，在程序运行之后要在终端输入一个数字，这个数字会作为num的值，和C语言里scanf，C++里的cin道理是一样的。后面双引号里的东西则作为提示 编译后在终端显示。

按ESC+：+wq保存后回到python文件夹下，输入

**python a.py**

即可进行编译了。



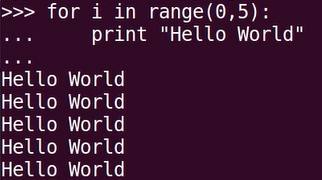
6. 循环语句

格式：

① **for i in range(起:止):**

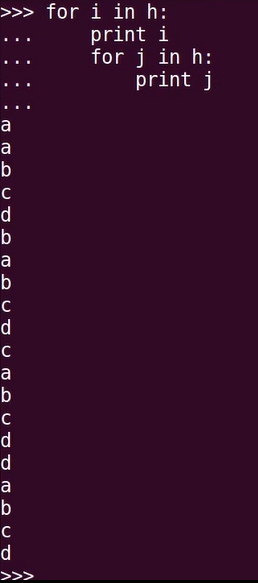
循环体内的任务

例：

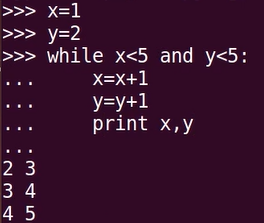


可以看到5次循环，Hello World被输出了5次。

② range也可以换成列表名使用。



③ while也可以用于循环，退出时使用break。



这里没用到break，我们说break一般用在条件语句之后，在满足某一条件之后break退出循环。

7.turtle模块

①环境安装

同样，如果电脑里没有turtle模块依然需要手动下载，在终端输入

**pip install turtle**

先进行安装。我在实际操作时还遇到了需要强制安装python-tk package的情况，于是再输入

**sudo apt-get install python-tk**

进行安装。

完成之后在python环境下输入

**import turtle**

如果没有报错就说明安装成功了。

②基本语句

**import turtle #导入模块**

**t=turtle.Pen() #使用模块中的Pen类实例化处一个t的对象，注意这里的P是大写**

**t.forward(像素点) #前进**

**t.backward(像素点) #后退**

**t.left(角度) #左转角度**

**t.right(角度) #右转角度**

**t.reset() #复位**

注意可以搭配循环画图，可以节省很多行代码。

8.函数，模块，包，变量

①函数

定义函数：

def+函数名+(参数表):

函数体

使用函数：

函数名(参数表)

例：

1.不含返回值

定义：

**def hello(yourname):**

**print “hello %s” %yourname**

使用：

**hello(“goudan”)**

输出：

**hello goudan**

2.含返回值

定义：

**def add(a,b):**

**return a+b**

使用：

**c=add(5,6)**

**c**

输出

**11**

3.内建函数：表示编译器自带的函数

例：

**abs(-10)**

输出：

**10**

②模块

模块是函数的集合，先导入，再使用

例：

**import time**

**time.asctime()**

输出：当前时间

③包

包含有多个模块

例：

**from PIL import Image**

④变量

全局变量与局部变量

全局变量作用于整个程序，局部变量只作用于当前函数，函数执行完毕后释放内存，该局部变量不复存在。

**9.面向对象的编程**

这部分很难，建议自己去网上搜一些东西或者看书理解，我先空着这部分，等有时间再把这个坑填了。

10.对文件的读写操作

使用pickle模块

**import pickle**

此时假设有一个名为game\_data的字典；我们要把它的内容保存到文件中

**game\_data={“position”:”N2 E3”,”pocket”:[“key”,”knife”],”money”:160}**

存：

**save\_file = open(“save.dat”,”wb”)**

**pickle.dump(game\_data,save\_file)**

**save\_file.close()**

**exit()**



注：这里保存的date.save用vim打开不具有可读性

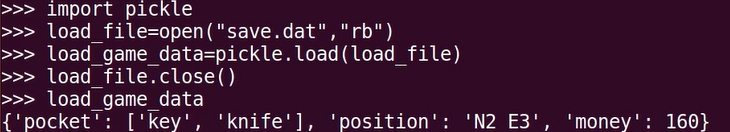
写：

**load\_file = (“save.dat”,”rb”)**

**load\_game\_data = pickle.load(load\_file)**

**load\_file.close()**

**load\_gale\_data**



到这里python一些基本的东西就算完成了。再深入一些的东西建议去kaggle网站学习，那儿有全英版的python基础教程。此外，github上也有python100天学习交流的笔记（包括机器学习），总之学习的途径有很多，这里也只是点到为止，一些比较难的东西压根都没写出来。所以……就先这样吧，接下来写一些机器学习的东西。

kaggle的python学习网址：<https://www.kaggle.com/learn/python>

四、Tensorflow机器学习

基于Tensorflow的NN：用张量表示数据，用计算图搭建神经网络，用绘画执行计算图，优化线上的权重（参数），得到模型。

0维对应标量scalar，1维对应向量vector，2维对应矩阵matrix，……，n维对应张量tensor。

张量可以表示0到n阶数组。

为了方便我们要先更改vim的配置。

终端中输入：

**vim ~/.vimrc**

**i**

**set ts=4**

**set nu**

更改完毕。

tensorflow里的数据类型tf.float32，tf.int32…

下面举个例子：

**cd ~**

**pwd**

**mkdir tf**

**cd tf**

**vim tf3\_1.py**

**import tensorflow as tf**

**a=tf.constant([1.0,2.0])**

**b=tf.constant([3.0,4.0])**

**result = a+b**

**print result**

**ESC+:+wq**

**python tf3\_1.py**



格式解释：

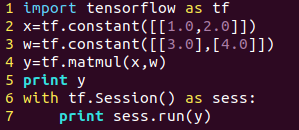
括号里第一部分是张量名，第二部分是维度，第三部分是数据类型。整个过程就是一个计算图，它是搭建神经网络的计算过程，只搭建不计算。如果想要计算，则需要用到Session（会话）执行计算图中的节点运算。

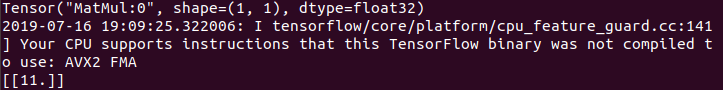
一般用法：

**with tf.Session() as sess:**

**print sess.run(y)**

下面举一个用矩阵乘法进行计算的例子：

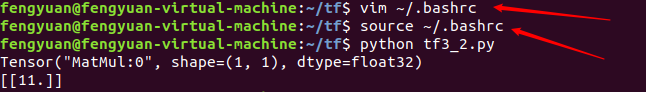




最下面一行是运算结果。中间则是警告，原因是可以进行一些省时的过程但是我们没有省，这个警告可以通过下面的代码关闭，在终端输入：

**vim ~/.bashrc**

**source ~/.bashrc**



相当于进行了矩阵乘法。

1.正向传播

下面进行随机参数生成的语句书写。

w = tf.Variable(tf.random\_normal([2,3],stddev = 2,mean = 0,seed = 1))

tf.random\_normal表示正态分布

[2,3]表示产生2×3的矩阵

stddev = 2表示标准差为2

mean=0表示均值为0

seed=1表示随机种子

tf.truncated\_normal()表示去掉最大偏离点的正态分布，也就是说随机出来的数据如果超过标准差，这个数据就重新生成。

tf.random\_uniform() 平均分布

tf.zeros全0数组 tf.zeros([3,2],int32)

tf.ones全1数组

tf.fill全定值数组 tf.fill([3,2],6)

tf.constant直接给值 tf.constant([3,2,1])

神经网络的实现：

1.准备喂给神经网络的数据集

2.先搭建计算图，再用会话执行（NN前向传播算法->计算输出）

3.迭代优化NN参数（NN反向传播算法->优化参数训练模型）

4.使用训练好的模型预测和分类

变量初始化，计算图节点运算使用会话（with结构）实现

with tf.Session() as sess:

sess.run()

变量初始化：在sess.run函数中用tf.global\_variables\_initializer()

init\_op = tf.global\_variables\_initializer()

sess.run(init\_op)

计算图节点运算：在sess.run函数中写入待运算的节点

sess.run(y)

用tf.placeholder占位，在sess.run函数中用fedd\_dict喂数据

喂一组数据：

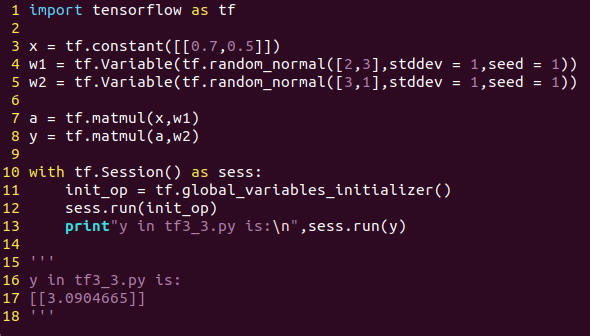
x= tf.placeholder(tf.float32,shape=(1,2))

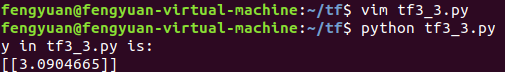
sess.run(y,feed\_dict={x:[[0.5,0.6]]})

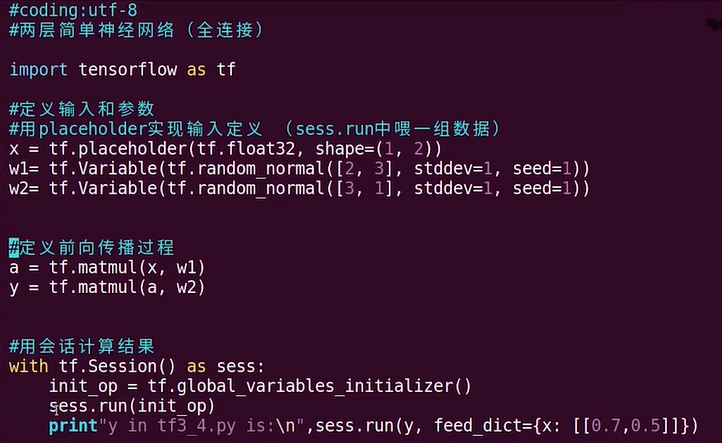
喂多组数据：

x=tf.placeholder(tf.float32,shape=(None,2))

sess.run(y,feed\_dict={x:[[0.1,0.2],[0.2,0.3],[0.3,0.4],[0.4,0.5]]})

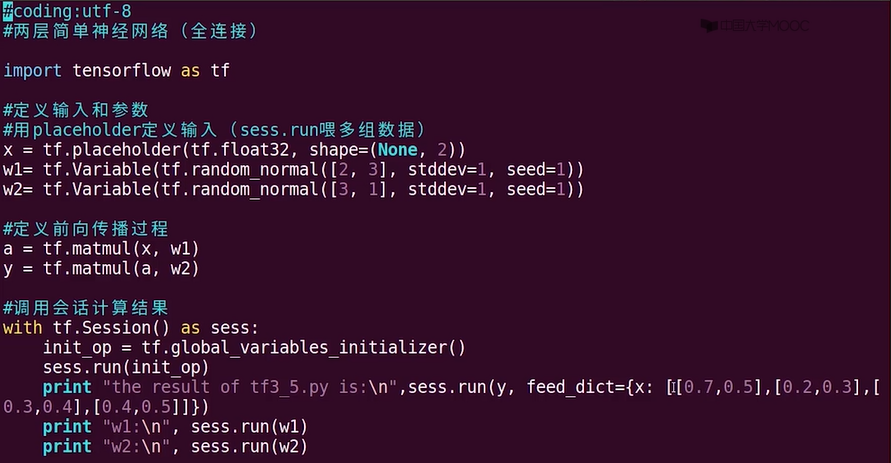






输出结果同上。

传入多组数据的代码：



运行结果，会给出四组值。通过print w1和w2可观察到计算机随机给出的两组权重的值。

2.反向传播

训练模型参数，在所有参数上使用梯度下降，使NN模型在训练数据上的损失函数最小。

损失函数（loss）：预测值y与已知答案y\_的差距

均方误差（最小二乘）

loss = tf.reduce\_mean(tf.square(y\_-y))

反向传播训练方法：以减小loss值为优化目标

train\_step = tf.train.GradientDescentOptimizer(learing\_rate).minimize(loss)

train\_step = tf.train.MomentumOptimizer(learning\_rate,momentum).minimize(loss)

train\_step = tf.train.AdamOptimizer(learning\_rate).minimize(loss)

学习率（learning\_rate）：表示每次参数跟新的幅度，建议使用一个较小的值，比如0.001。

#coding:utf-8

import tensorflow as tf

import numpy as np #numpy模块，是python的科学计算模块

BATCH\_SIZE = 8 #表示一次吃入神经网络多少数据

seed = 23455 #随机的seed值

rng = np.random.RandomState(seed)

X = rng.rand(32,2) #随机数返回32行2列的矩阵，表示32组体积和重量，作为输入数据集

Y = [[int(x0+x1<1)] for (x0,x1) in X]

print “X:\n”,X

print”Y:\n”,Y

x=tf.placeholder(tf.float32,shape=(None,2))

y\_=tf.placeholder(tf.float32,shape=(None,1))

w1 = tf.Variable(tf.random\_normal([2,3],stddev = 1,seed = 1))

w2 = tf.Variable(tf.random\_normal([3,1],stddev = 1,seed = 1))

a = tf.matmul(x,w1)

y = tf.matmul(a,w2)

loss = tf.reduce\_mean(tf.square(y-y\_))

train\_step = tf.train.GradientDescentOptimizer(0.001).minimize(loss)

#或train\_step = tf.train.MomentumOptimizer(0.001,0.9).minimize(loss)

#或train\_step = tf.train.AdamOptimizer(0.001).minimize(loss)

with tf.Session() as sess:

init\_op = tf.global\_variables\_initializer()

sess.run(init\_op)

print ”w1:\n”,sess.run(w1)

print ”w2:\n”,sess.run(w2)

print ”\n”

#训练模型

STEPS = 3000

for i in range(STEPS):

start = (i\*BATCH\_SIZE) % 32

end = start + BATCH\_SIZE

sess.run(train\_step,feed\_dict = {x:X[start:end],y\_:Y[start:end]})

if i % 500 == 0:

total\_loss = sess.run(loss,feed\_dict = {x:X,y\_:Y})

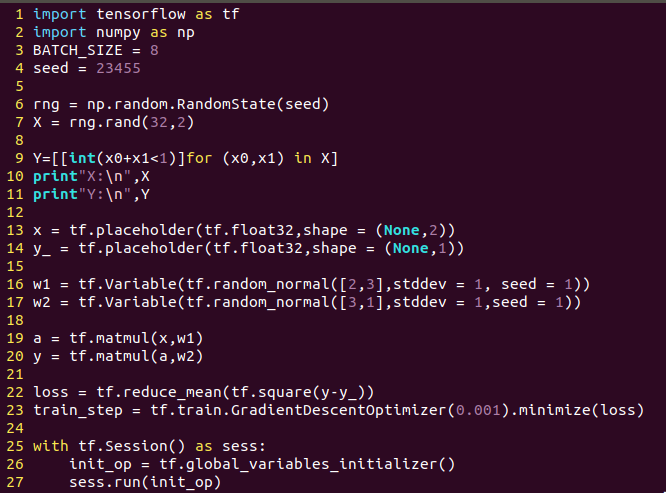
print(“After %d training step(s), loss on all data is %g” %(I,total\_loss))

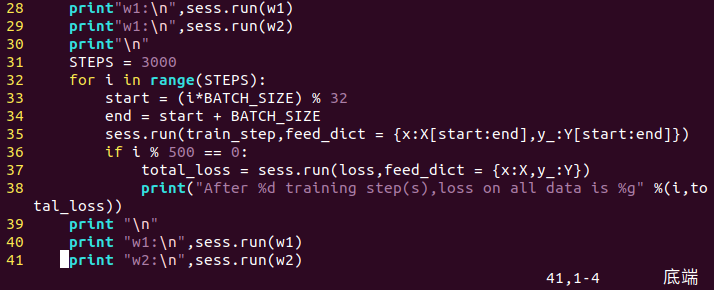
#输出训练后的参数值

print “\n”

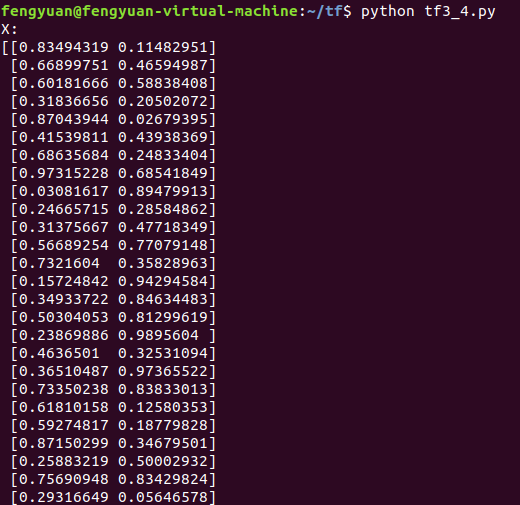
print “w1:\n”,sess.run(w1)

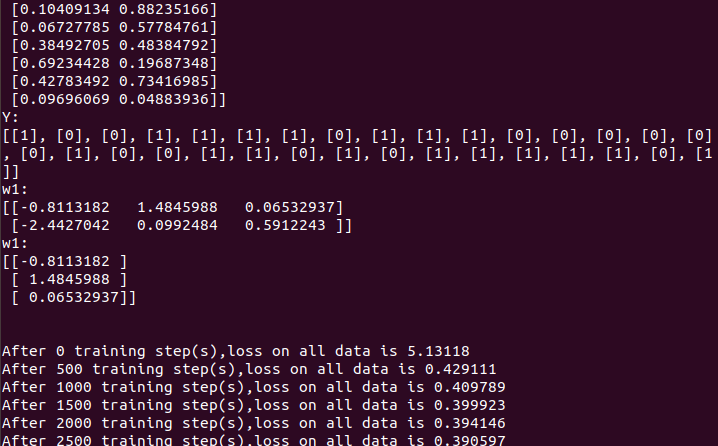
print “w2:\n”,sess.run(w2)

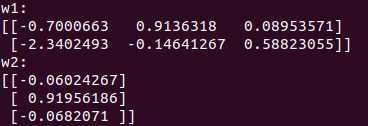




运行结果：

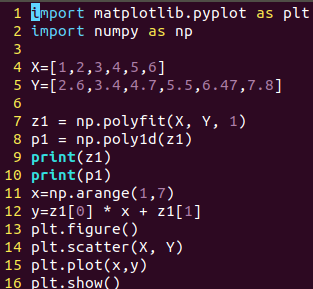






这里放松一下，插入一个一元一次方程的拟合代码加绘图，很简单= =

也便于理解一下numpy模块和matplotlib.pyplot模块

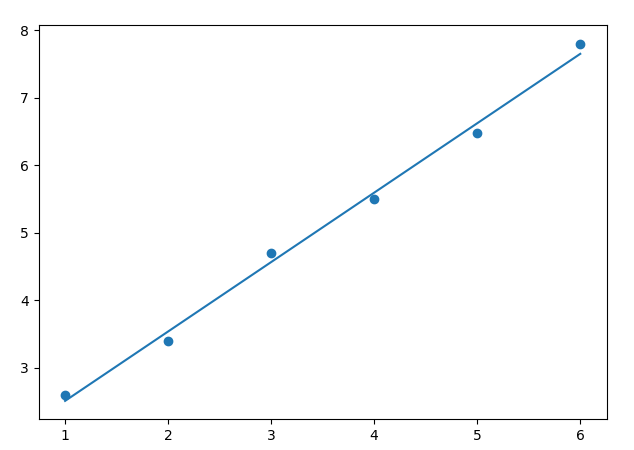


运行结果：

终端：



图片：



搭建神经网络的八股：

1.准备

import

常量定义

生成数据集

2.前传

定义输入，参数和输出

x=

y\_=

w1=

w2=

a=

y=

3.反传

定义损失函数，反向传播方法

loss=

train\_step=

4.迭代

生成会话，训练STEPS轮

最好把上面的代码背下来！

总结：前向传播搭建网络结构，反向传播训练网络参数

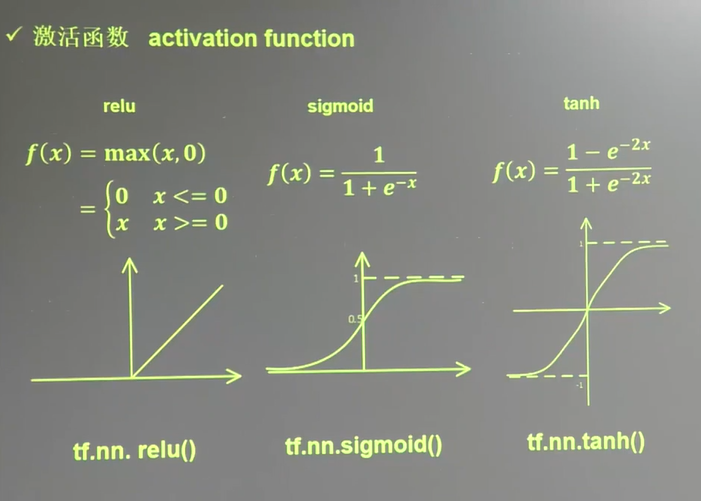
1943年McCulloch Pitts的神经元模型：

其中f为激活函数activation function，b为偏置项bias。

激活函数：

作用：引入非线性，提高模型的表达力，使模型具有更好的区分度

常用激活函数：



神经网络的层数 = 隐藏层的个数+1个输出层 ，这个过程是不计算输入层的个数的。

总参数 = 总W+总b

总W = 输入层×隐藏层+隐藏层， 总b = 隐藏层×输出层+输出层

神经网络的优化：

从四个方面进行优化，分别是损失函数loss，学习率learning\_rate，滑动平均ema，正则化regularization

损失函数loss

有三种方式使loss最小，分别是mse（均方误差），自定义与ce（交叉熵）。

1.均方误差：

MSE(y\_,y) =

代码：

lose\_mse = tf.reuce\_mean(tf.squaer(y\_-y))

2.自定义损失函数：f(y\_,y)，视具体情况定义。

例：简单的一个分段函数：

loss = tf.reduce.sum(tf.where(tf.greater(y,y\_),COST\*(y-y\_),PROFIT\*(y\_-y)))

上面这段代码表示先判断y>y\_？如果大于执行COST\*(y-y\_)，小于则执行PROFIT\*(y\_-y)。

3.交叉熵

代码：

ce = -tf.reduce\_mean(y\_\*tf.log(tf.clip\_by\_value(y,1e-12,1.0)))

当n分类的n个输出（y1，y2，y3，……，yn）通过softmax()函数，是满足概率分布要求（对于任意x都有Px∈[0,1]且所有Px的和为1）

代码：

ce = tf.nn.sparse\_softmax\_cross\_entrophy\_with\_logits(logits = y,labels = tf.argmax(y\_,1))

cem = tf.reduce\_mean(ce)

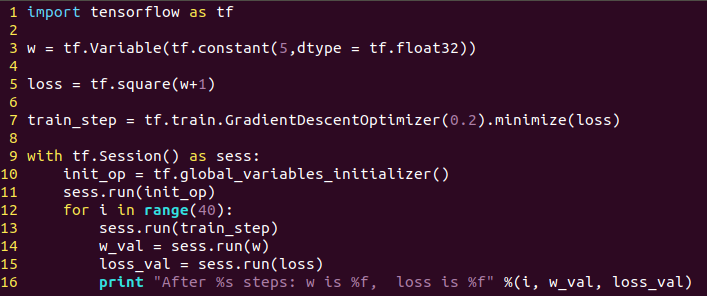
学习率learning\_rate

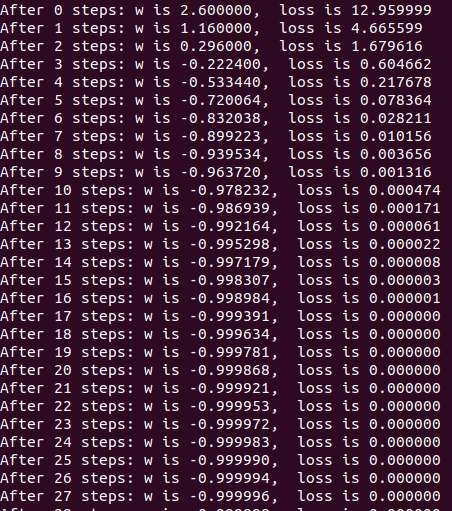
学习率（learning\_rate）：

其中是指损失函数loss对w的导数。

通过多次运算，寻找合适的w使loss函数的值最小（loss 的导数为0）

以lose=(w+1)^2为例，有以下代码：



运行结果：  


可见最后w收敛于-1，达到预期效果。

但是学习率不能过大或过小。学习率过大震荡不收敛，学习率过小收敛速度慢。

指数衰减学习率：

learning\_rate=LEARINING\_RATE\_BASE\*LEARNING\_RATE\_DECAY^(gloabal\_step/LEARNING\_RATE\_STEP)

其中LEARINING\_STEP = 总样本数/BATCH\_SIZE

我们可以通过以下代码动态更新学习率:

global\_step = tf.Variable(0,trainable = False)

learning\_rate=tf.train.exponential\_decay(LEARNING\_RATE\_BASE,global\_step,LEARNING\_RATE\_STEP,LEARINING\_RATE\_DECAY,staircase = True)

到这里再从头到尾捋一下全部思路：

①

import

import tensorflow as tf

②

定义最初学习率LEARNING\_RATE\_BASE

定义学习率的衰减率 LEARNING\_RATE\_DECAY

定义更新学习率的轮数LEARNING\_RATE\_STEP（一般为总样本数/BATCH\_SIZE）,这里再提醒一下，BATCH\_SIZE是每次喂给计算机的数据个数。

③

定义运行几轮BATCH\_SIZE的计数器global\_step，初值设为0，设为不被训练。

global\_step = tf.Variable(0.trainable = False)

定义指数下降学习率

learning\_rate=tf.train.exponential\_decay(LEARNING\_RATE\_BASE,global\_step,LEARNING\_RATE\_STEP,LEARINING\_RATE\_DECAY,staircase = True)

参数的顺序为：1最初学习率，2运行轮数，3更新学习率的轮数，4学习率的衰减率，5梯度下降为真

④

设定初值w

w=tf.Variable(tf.constant(5, dtype = tf.float32))

定义损失函数loss

loss = tf.square(w+1)

定义反向传播方法（3种选一种）

train\_step = tf.train.GradianDescentOptimizer(learning\_rate).minimize(loss,glbal\_step = global\_step)

⑤

生成会话

with tf.Session() as sess:

init\_op = tf.global\_variables\_initializer()

sess.run(init\_op)

for i in range(40):

sess.run(train\_step)

learning\_rate\_val = sess.run(learning\_rate)

global\_step\_val = sess.run(global\_step)

w\_val = sess.run(w)

loss\_val = sess.run(loss)

print “After %s steps: global\_step is %f, w is %f, learning rate is %f, loss is %f” % (i,global\_step\_val,w\_val,learning\_rate\_val,loss\_val)

滑动平均ema

也叫影子值，记录了每个参数一段时间内过往值的平均，增加了模型的泛化性。

影子 = 衰减率\*影子+（1-衰减率）\*参数

衰减率 = min{MOVING\_AVERAGE\_DEACAY,1+轮数/10+轮数}

其中MOVING\_AVERAGE\_DEACAY为超参数，一般是个比较大的值，例0.99。

代码：

ema = tf.train.ExponentialMovingAverage(MOVING\_AVERAGE\_DECAY,global\_step)

ema\_op = ema.apply(tf.trainable\_variables())

其中ema.apply()定义了对括号里的参数求滑动平均；而tf.trainable\_variables()可以自动把待训练的参数汇总成列表。

计算滑动平均和训练过程绑定到一起，有：

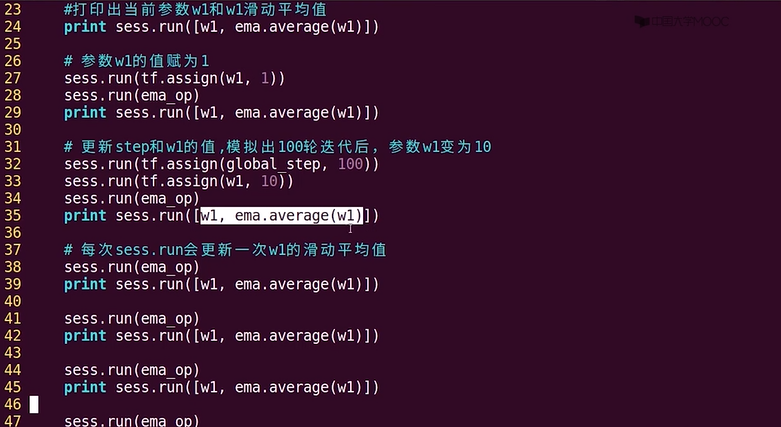
with tf.control\_dependencies([train\_step,ema\_op]):

train\_op = tf.no\_op(name = ‘train’)

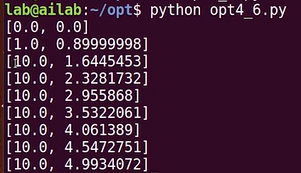
ema.average(参数名)：可以返回参数的滑动平均值

下面给一个滑动平均的例子：





输出：

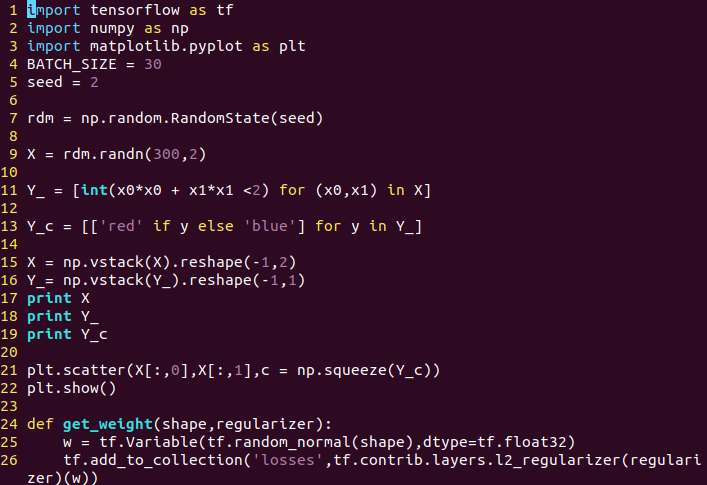


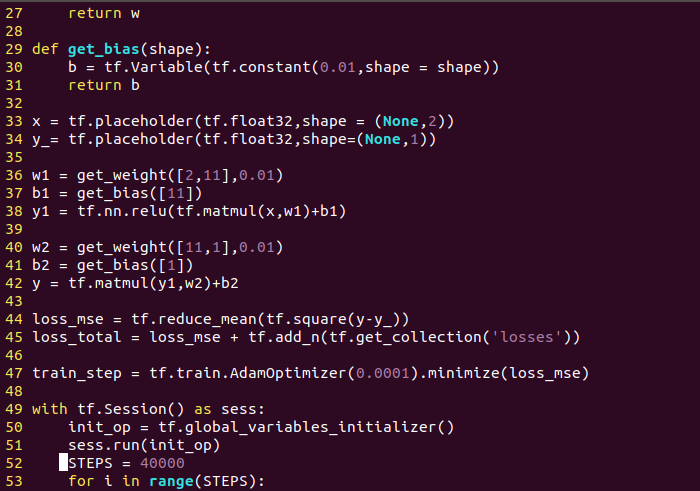
可以看到，每执行一次，后面的滑动平均都在逼近w1。

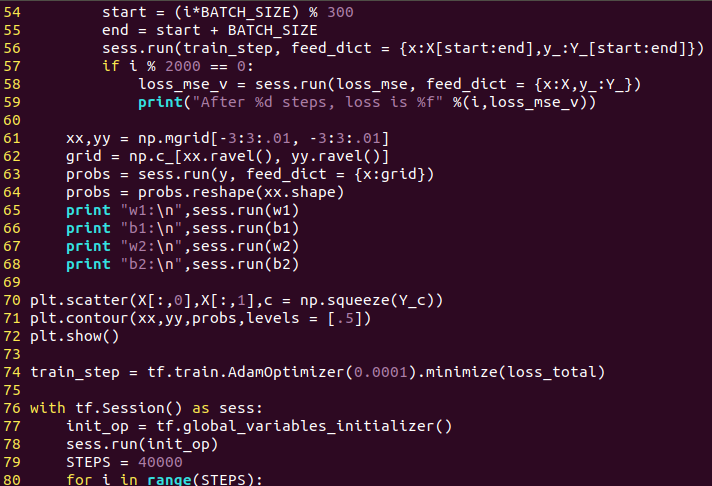
正则化

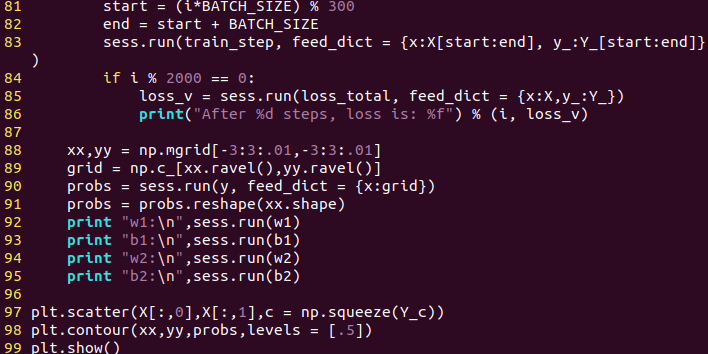
正则化是指在损失函数中引入模型复杂度指标，利用给W加权值，弱化了训练数据的噪声（一般只对w进行正则化）

代码：

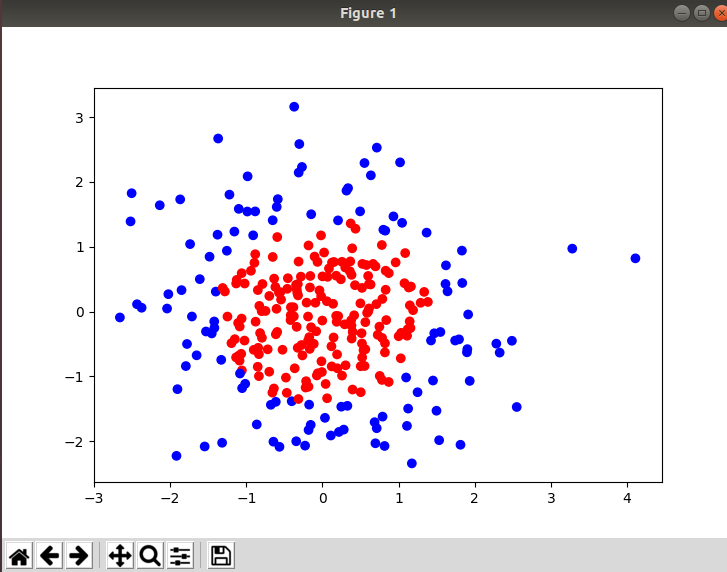




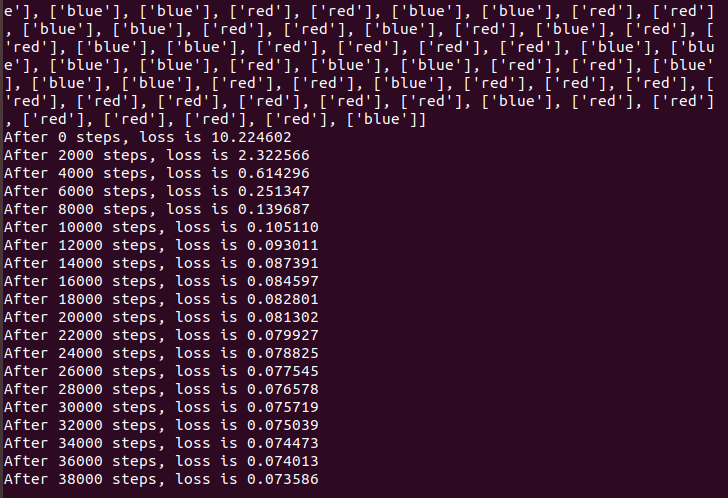




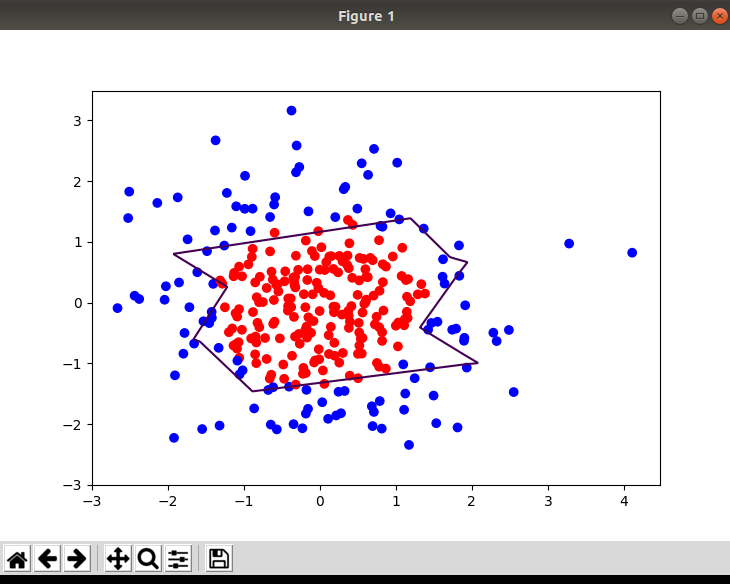
运行结果：



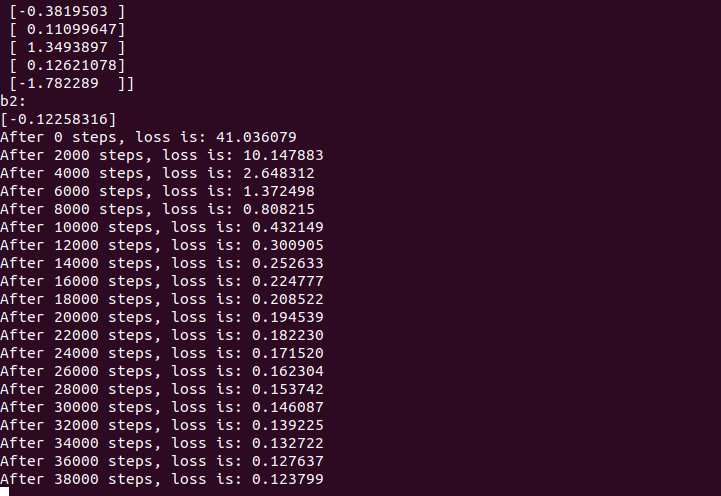
第二轮学习：



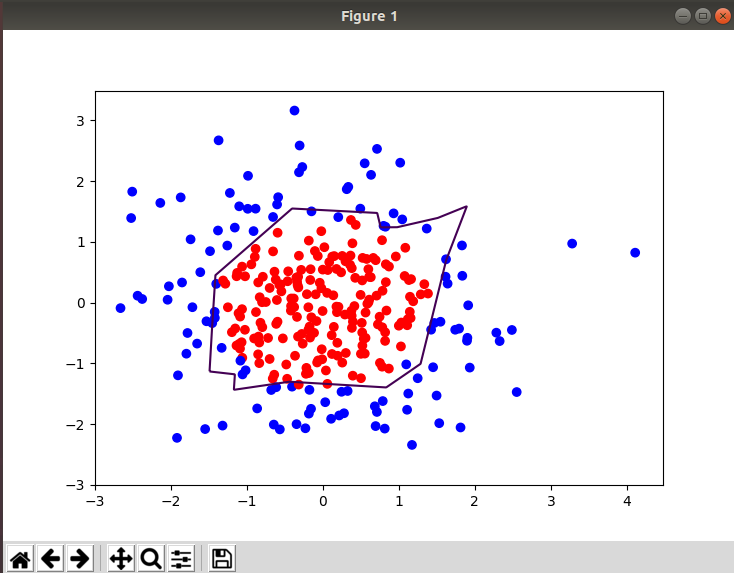
…………



第三轮学习：



…………



对比两幅图发现，第三幅更能准确描述红色点的范围，也就是说正则化可以有效提高学习效率。

新坑：python爬虫

1.简单的一个爬虫程序

1.找到需要爬虫的网站

2.用requests模块爬到想要下载的URL（统一资源定位符，简称网址）

3.写脚本

（注意先安装第三方库（模块），requests等，在终端（cmd）里用pip下载就可以）

# !/user/bin/env python

# -\*- coding:utf-8 -\*-

import requests

import re #（正则模块）

#下载网页  
url = ‘……’

#模拟浏览器发送http请求（对于chrome可以在页面处按F12查看网页源代码，在network中观察请求方式是get还是post）

#这里举一个get文本的例子，再把响应的请求用response表示出来

#注意如果是中文要更改response的属性encoding编码方式utf-8

response.encoding = ‘utf-8’

response = requests.get(url)

print(response.text) #注意这里text是文本，若是图片视频等用其他方式搞，这里我暂时只知道文本

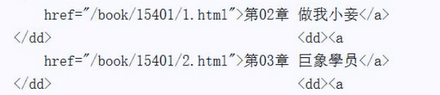
html = response.text #这步操作是用来获得目标小说主页的网页源码

print(html)

#接下来首先获取存放所有超链接的标题头，这里以<dl id=”list”>为例

dl = re.findall(r’<dl id = ‘list’>.\*?</dl>’,html,re.S)[0] # \*?表示非菜单匹配，后面的re.S用来匹配不可见字符，如果不加则显示不出非可见字符，后面的[0]用于把列表剥离出来

print(dl) #用于测试是否成功剥离



#以上图为例，接下来捕获章节标题和子网址

chapter\_info\_list = re.findall(r’href=”(.\*?)”>(.\*?)<’,dl)

#文本大标题，这里同样也是findall大法，这里只是简单的例子模板

title = re.findall(r’<meta property = “og : title” content=”(.\*?)”/>’, html)[0]

#创建一个txt文件，用来保存文字内容

fb = open(‘%s.txt’ % title, ‘w’, encoding = ‘utf-8’)

for chapter\_info in chapter\_info\_list:

chapter\_url, chapter\_title = chapter\_info

chapter\_url = “……%s” % chapter\_url #解释一下，这里由于子网址可能是不完整的，要在之前的网页后面加上/路径。

print(chapter\_url,chapter\_title) #打印，检测

#接下来下载内容

chapter\_response = requests.get(chapter\_url)

chapter\_response.encoding = ‘utf-8’

chapter\_html = chapter\_response.text

chapter\_content=re.findall(r’<script>a1\(\);</script>(.\*?)<script>a2\(\);</script>’,chapter\_html,re.S)[0] #这里假定了内容含在两个,script之间，注意括号转义的问题，视具体情况操作而定（虽然大多情况下都是这样的）

#清洗数据

chapter\_content = chapter\_content.replace(‘ ’,’’)

chapter\_content = chapter\_content.replace(‘&nbsp’,’’)

chapter\_content = chapter\_content.replace(‘<br/>’,’’)

print(chapter\_content) #测试

#持久化

fb.write(chapter\_title)

fb.write(chapter\_content)

fb.write(‘\n’)

print(chapter\_url)

2.爬虫知识点讲解