TensorFlow 开发起步

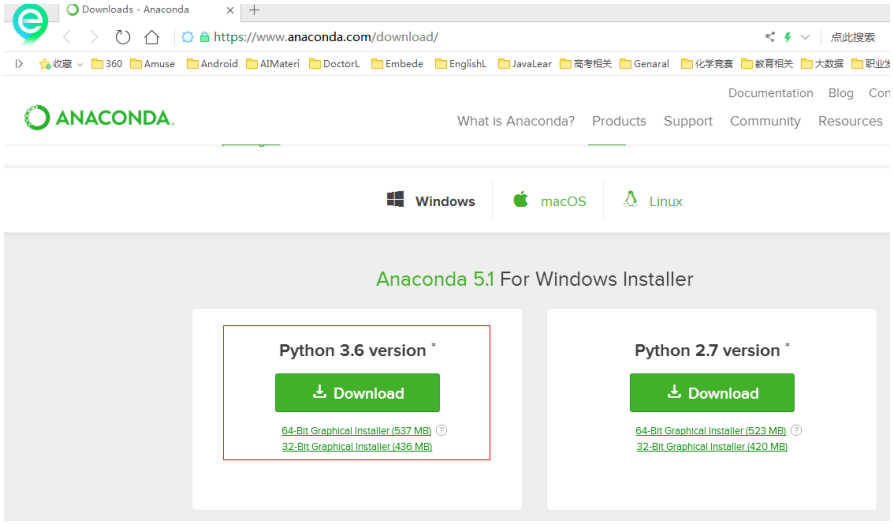
成都职业技术学院　　卓国锋

一、TensorFlow 安装

1、Anaconda 安装

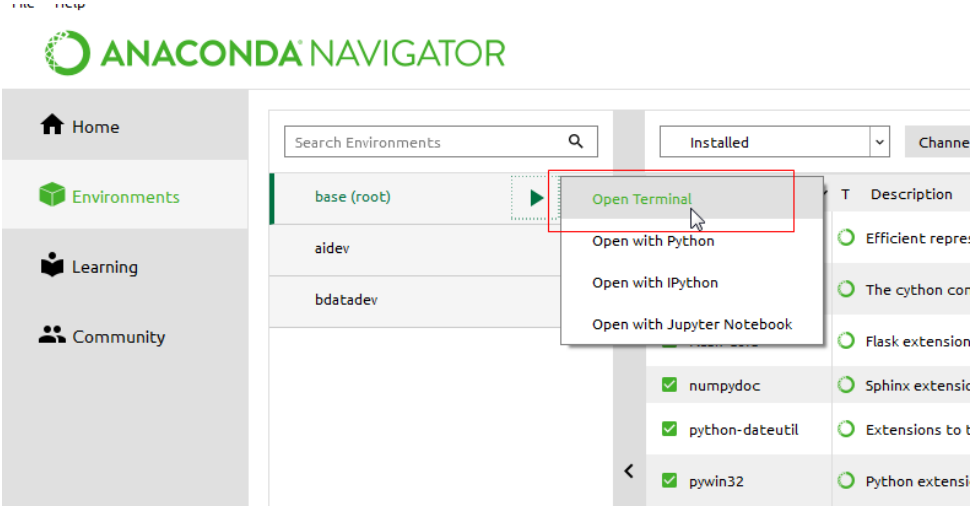
Anaconda3-5.1.0-Windows-x86_64

下载网址：https://www.anaconda.com/download/



2、Tensorflow 安装

1）进入到管理员命令界面



2）tensorflow 安装

```
pip install tensorflow
```

3)tensorflow 更新：

```
pip install --upgrade tensorflow
```

4)tensorflow 卸载：

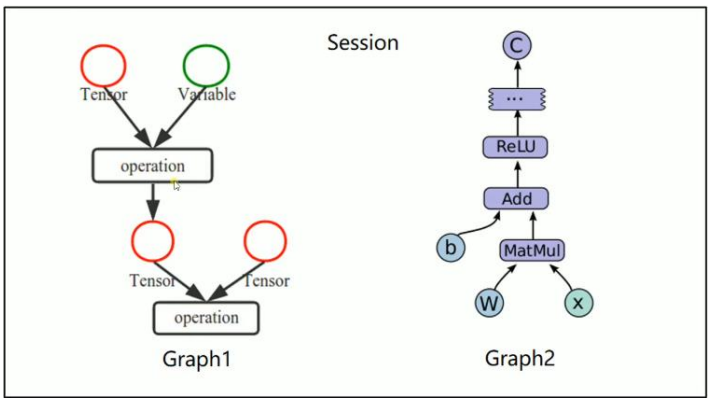
```
Pip uninstall tensorflow
```

三、Tensorflow 简介

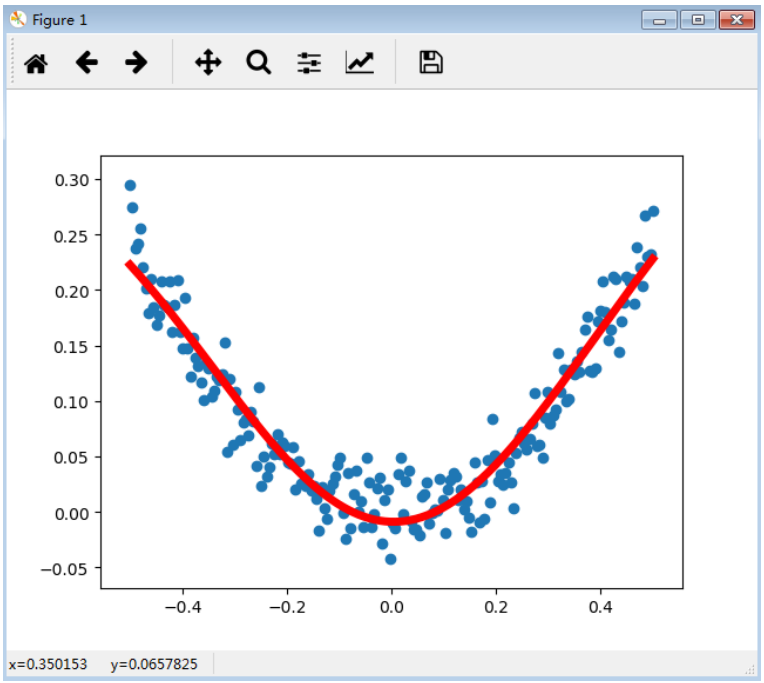
一）Tensorflow 说明：

Tensorflow 是一个编程系统，使用图（graphs）来表示计算任务，图（graphs）中的节点

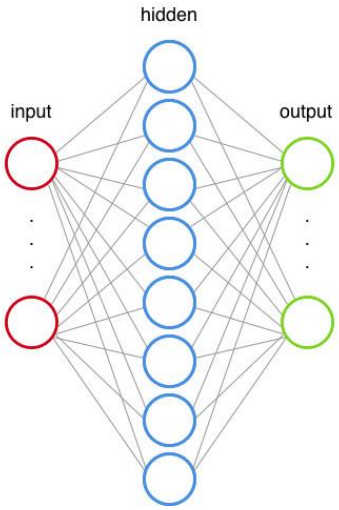
称之为 op（operation），一个 op 获得 0 个或多个 Tensor，执行计算，产生 0 个或多个 Tensor。Tensor 看作是一个 n 维的数组或列表。图必须在会话（Session）里被启动。



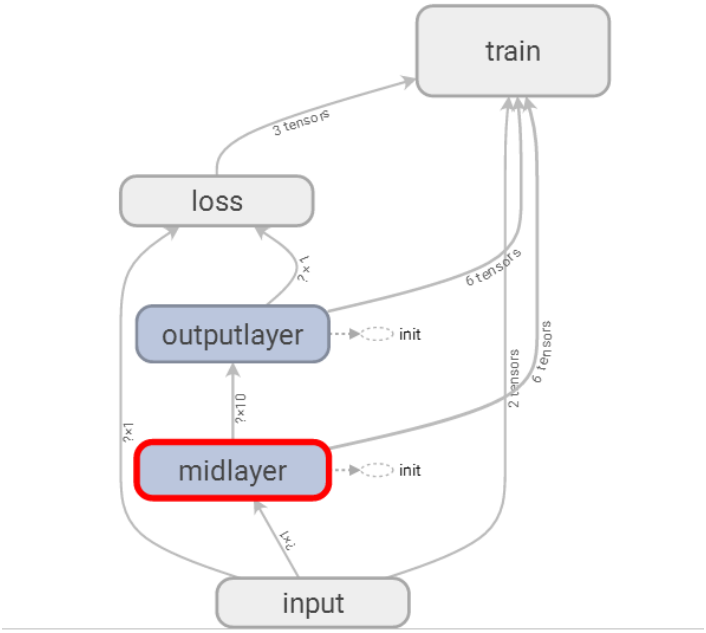
二）非线性回归实现过程说明 TensorFlow：



1、数据流图分析



2、模型结构图：



三）实现过程：

1、导入库

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

2、生成样本

使用 numpy 生成 200 个随机点

```
x_data = np.linspace(-0.5, 0.5, 200)[:, np.newaxis]
noise = np.random.normal(0, 0.02, x_data.shape)
y_data = np.square(x_data) + noise
```

3、定义两个 placeholder（输入值）

```
x = tf.placeholder(tf.float32, [None, 1])
y = tf.placeholder(tf.float32, [None, 1])
```

4、定义神经网络中间层

```
Weights_L1 = tf.Variable(tf.random_normal([1, 10]))
biases_L1 = tf.Variable(tf.zeros([1, 10]))
Wx_plus_b_L1 = tf.matmul(x, Weights_L1) + biases_L1
L1 = tf.nn.tanh(Wx_plus_b_L1)
```

5、定义神经网络输出层

```
    Weights_L2 = tf.Variable(tf.random_normal([10, 1]))
    biases_L2 = tf.Variable(tf.zeros([1, 1]))
    Wx_plus_b_L2 = tf.matmul(L1, Weights_L2) + biases_L2
    prediction = tf.nn.tanh(Wx_plus_b_L2)
```

6、二次代价函数：损失函数

```
        loss = tf.reduce_mean(tf.square(y - prediction))
```

7、调用优化器

使用梯度下降法训练:

```
    train_step = tf.train.GradientDescentOptimizer(0.1).minimize(loss)
```

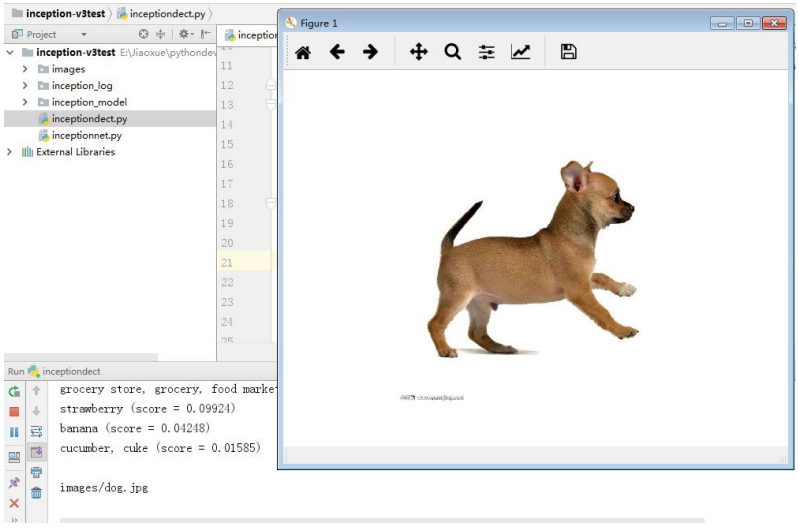
8、创建会话执行运算

```
with tf.Session() as sess:
    # 变量初始化
    sess.run(tf.global_variables_initializer())
    for _ in range(2000):
        sess.run(train_step, feed_dict={x: x_data, y: y_data})
# 获得预测值
    prediction_value = sess.run(prediction, feed_dict={x: x_data})
```

9、绘制结果图

```
plt.figure()
plt.scatter(x_data, y_data)
plt.plot(x_data, prediction_value, 'r-', lw=5)
plt.show()
```

四、图像分类案例实现



一）获取模型

1、导入库

```
    import tensorflow as tf
    import os
    import tarfile
    import requests
```

2、定义模型下载地址与保存位置

```
# inception 模型下载地址
inception_pretrain_model_url                                                    =
'http://download.tensorflow.org/models/image/imagenet/inception-2015-12-05.tgz'
# 模型存放地址
inception_pretrain_model_dir = "inception_model"
if not os.path.exists(inception_pretrain_model_dir):
    os.makedirs(inception_pretrain_model_dir)
# 获取文件名，以及文件路径
filename = inception_pretrain_model_url.split('/')[-1]
```

```
filepath = os.path.join(inception_pretrain_model_dir, filename)
```
3、下载模型
```
    if not os.path.exists(filepath):
        print("download: ", filename)
        r = requests.get(inception_pretrain_model_url, stream=True)
        with open(filepath, 'wb') as f:
            for chunk in r.iter_content(chunk_size=1024):
                if chunk:
                    f.write(chunk)
    print("finish: ", filename)
```
4、解压文件
```
tarfile.open(filepath, 'r:gz').extractall(inception_pretrain_model_dir)
```
5、定义模型结构存放文件
```
log_dir = 'inception_log'
if not os.path.exists(log_dir):
    os.makedirs(log_dir)
# classify_image_graph_def.pb 为 google 训练好的模型
inception_graph_def_file = os.path.join(inception_pretrain_model_dir,
'classify_image_graph_def.pb')
```
6、创建会话保存模型图
```
with tf.Session() as sess:
    # 创建一个图来存放 google 训练好的模型
    with tf.gfile.FastGFile(inception_graph_def_file, 'rb') as f:
        graph_def = tf.GraphDef()
        graph_def.ParseFromString(f.read())
        tf.import_graph_def(graph_def, name='')
    # 保存图的结构
    writer = tf.summary.FileWriter(log_dir, sess.graph)
    writer.close()
```
二）利用模型来分类图片
1、导入库
```
    import tensorflow as tf
    import os
    import numpy as np
    from PIL import Image
    import matplotlib.pyplot as plt
```
2、处理保存分类名称文件
```
class NodeLookup(object):
    def __init__(self):
        label_lookup_path = 'inception_model/imagenet_2012_challenge_label_map_proto.pbtxt'
        uid_lookup_path = 'inception_model/imagenet_synset_to_human_label_map.txt'
        self.node_lookup = self.load(label_lookup_path, uid_lookup_path)
    def load(self, label_lookup_path, uid_lookup_path):
        # 加载分类字符串对应分类名称的文件
```

```
        proto_as_ascii_lines = tf.gfile.GFile(uid_lookup_path).readlines()
        uid_to_human = {}
        #一行一行读取数据
        for line in proto_as_ascii_lines :
            #去掉换行符
            line=line.strip('\n')
            #按照'\t'分割
            parsed_items = line.split('\t')
            #获取分类编号
            uid = parsed_items[0]
            #获取分类名称
            human_string = parsed_items[1]
            #保存编号字符串 n********与分类名称映射关系
            uid_to_human[uid] = human_string
        #  加载分类字符串 n********对应分类编号 1-1000 的文件
        proto_as_ascii = tf.gfile.GFile(label_lookup_path).readlines()
        node_id_to_uid = {}
        for line in proto_as_ascii:
            if line.startswith('  target_class:'):
                #获取分类编号 1-1000
                target_class = int(line.split(': ')[1])
            if line.startswith('  target_class_string:'):
                #获取编号字符串 n********
                target_class_string = line.split(': ')[1]
                #保存分类编号 1-1000 与编号字符串 n********映射关系
                node_id_to_uid[target_class] = target_class_string[1:-2]
        #建立分类编号 1-1000 对应分类名称的映射关系
        node_id_to_name = {}
        for key, val in node_id_to_uid.items():
            #获取分类名称
            name = uid_to_human[val]
            #建立分类编号 1-1000 到分类名称的映射关系
            node_id_to_name[key] = name
        return node_id_to_name
    #传入分类编号 1-1000 返回分类名称
    def id_to_string(self, node_id):
        if node_id not in self.node_lookup:
            return ''
        return self.node_lookup[node_id]
```

3、创建一个图来存放 google 训练好的模型

```
with tf.gfile.FastGFile('inception_model/classify_image_graph_def.pb', 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
    tf.import_graph_def(graph_def, name='')
```

4、创建会话识别分类图片

```
with tf.Session() as sess:
    softmax_tensor = sess.graph.get_tensor_by_name('softmax:0')
    #遍历目录
    for root,dirs,files in os.walk('images/'):
        for file in files:
            #载入图片
            image_data = tf.gfile.FastGFile(os.path.join(root,file), 'rb').read()
            predictions = sess.run(softmax_tensor,{'DecodeJpeg/contents:0': image_data})#图片格式是 jpg 格式
            predictions = np.squeeze(predictions)#把结果转为 1 维数据
            #打印图片路径及名称
            image_path = os.path.join(root,file)
            print(image_path)
            #显示图片
            img=Image.open(image_path)
            plt.imshow(img)
            plt.axis('off')
            plt.show()
            #排序
            top_k = predictions.argsort()[-5:][::-1]
            node_lookup = NodeLookup()
            for node_id in top_k:
                #获取分类名称
                human_string = node_lookup.id_to_string(node_id)
                #获取该分类的置信度
                score = predictions[node_id]
                print('%s (score = %.5f)' % (human_string, score))
            print()
```

五、TensorBoard 的基本使用

1、安装 TensorBoard

```
conda install tensorboard
```

2、命名空间为 TensorBoard 中的节点

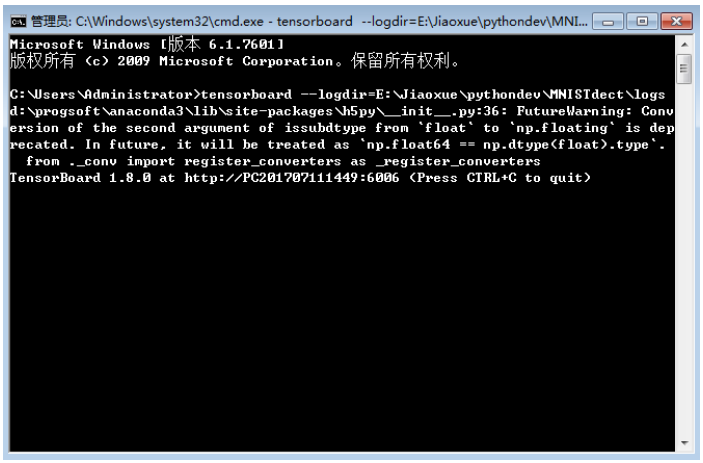
例如：

```
with tf.name_scope('input'):
    x = tf.placeholder(tf.float32,[None,784],name='x-input')
    y = tf.placeholder(tf.float32,[None,10],name='y-input')
with tf.name_scope('layer'):
    # 创建一个简单的神经网络
    with tf.name_scope('wights'):
        W = tf.Variable(tf.zeros([784, 10]), name='W')
    with tf.name_scope('biases'):
        b = tf.Variable(tf.zeros([10]), name='b')
    with tf.name_scope('wx_plus_b'):
```

```
            wx_plus_b = tf.matmul(x, W) + b
        with tf.name_scope('softmax'):
            prediction = tf.nn.softmax(wx_plus_b)
    #保存模型图片
writer = tf.summary.FileWriter('logs/', sess.graph)
```

3、启动 TensorBoard

tensorboard --logdir=E:\Jiaoxue\pythondev\MNISTdect\logs



4、浏览 TensorBoard

http://PC201707111449:6006