人工智能基础LAB2实验要求

DDL：2022.6.28 23:59:59

文档修改时间：2022.6.4

修改内容：

1. SVM算法中的`cvxpy`库版本要求：1.1.13
2. 添加SVM算法数据集样本特征数据类型描述
3. 修复了反向传播链式法则公式中的下标错误

1. 实验内容与提示

本次实验包含传统机器学习与深度学习两部分。

实验部分需要使用python=3.6，建议使用anaconda管理python环境，深度学习部分要求使用pytorch=1.8.1，torchvision=0.9.1完成（安装说明见https://pytorch.org，学习教程可以参考https://pytorch123.com），实验部分使用CPU足够训练，如果想体验GPU的速度可以使用colab。
安装部分见附录，本次实验pytorch所需api见**pytorch使用帮助.pdf**

评分标准

项目内容	分值
决策树	2
SVM算法	2
手写感知机模型并进行反向传播	2
实现一个卷积神经网络	2
实验报告	2
迟交或者不符合格式	倒扣分

2. 传统机器学习

说明：机器学习部分禁止直接调用任何机器学习库（如sklearn），需手动实现算法部分。

本实验中我们不以准确率等评价指标作为本次实验的评价标准，提供的评价指标是为了帮助同学们判断实现的模型是否存在较大的问题，我们更关注代码的完整程度和独立完成情况，禁止抄袭代码。

实验环境及配置

python=3.6(或更高版本)

numpy=1.21(或更高版本)

2.1决策树

原数据集地址

在决策树部分，你需要通过病人的生理状况、临床观察症状等来推断病人是否癌症复发。

每个样本由9个**离散属性**组成，属性包括肿瘤大小，年龄，是否接触放射性物质等，每个属性的可能取值数在2-12不等。标签包括癌症复发或不复发，取值为{0, 1}，是一个二分类任务。助教已经将数据预处理成便于使用的形式，数据文件放置在`src1/dataset/dt`下。

我们在`DecisionTree`类中提供两个接口：

- `fit(train_features, train_labels)`，在这个函数中你需要根据训练数据生成一棵决策树，`train_features`和`train_labels`在`main.py`中加载。
- `predict(test_features)`，在这个函数中你需要根据已经生成的决策树来预测标签，`test_features`在`main.py`中加载。这个函数需要返回一个维度为`(测试样本数, )`的numpy数组，代表你对测试标签的预测值，这个预测值会和实际的测试标签`test_labels`进行比较。

在`DecisionTree`类中，上述两个函数接口不允许修改，并且`predict`函数的返回值必须满足上述要求。但是，我们允许你在实现时对`DecisionTree`类添加自定义的类方法或属性，以便于实现递归形式的决策树算法。

决策树的生成算法可以参考：

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
　　属性集 $A = \{a_1, a_2, \ldots, a_d\}$.
过程： 函数 TreeGenerate(D, A)
1: 生成结点 node;
2: **if** D 中样本全属于同一类别 C **then**
3: 　将 node 标记为 C 类叶结点; **return**
4: **end if**
5: **if** $A = \varnothing$ **OR** D 中样本在 A 上取值相同 **then**
6: 　将 node 标记为叶结点, 其类别标记为 D 中样本数最多的类; **return**
7: **end if**
8: 从 A 中选择最优划分属性 a_*;
9: **for** a_* 的每一个值 a_*^v **do**
10: 　为 node 生成一个分支; 令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集;
11: 　**if** D_v 为空 **then**
12: 　　将分支结点标记为叶结点, 其类别标记为 D 中样本最多的类; **return**
13: 　**else**
14: 　　以 TreeGenerate(D_v, $A \setminus \{a_*\}$)为分支结点
15: 　**end if**
16: **end for**
输出： 以 node 为根结点的一棵决策树

图 4.2　决策树学习基本算法

你需要在实验报告中贴出程序打印出的准确率，并且结合代码解释你是如何选择最优划分属性，如何生成叶子结点和中间结点的。

2.2支持向量机

原数据集地址

在支持向量机部分，你需要根据病人的生理指标预测在给定的时间中病人是否会死亡。

为处理这一问题，我们将其抽象成一个二分类问题：输入生理指标与给定的时间，模型将其分类为死亡或未死亡。

每个样本具有7个**数值型**属性，包括年龄，射血分数，血小板数以及给定的时间等等。由于助教已经帮你进行了数据标准化，避免了量纲差异，所以所有数据类型**均为浮点数**，且数值在-10到10之间。标签即为患者在给定的时间中是否死亡，1表示已死亡，**-1表示未死亡。**使用{-1,1}作为标签集便于SVM的处理。

支持向量机的算法用一个类来实现，这个类的实例化需要提供3个超参数：

- `C`：软间隔参数，默认为1。
- `kernel`：核函数，可以在`Linear`，`Poly`，`Gauss`三个选项中选择，代表线性核，多项式核，高斯核。
- `epsilon`：由于凸优化问题使用相应的凸优化问题求解库，求解完成后可能有一些变量非常小，这时候我们将其视为0，即优化问题求解完成后小于`epsilon`的变量均视为0。

SVM类有如下接口必须完成实现：

- `fit(train_features, train_labels)`，在这个函数中你需要根据训练数据训练SVM分类器，`train_features`和`train_labels`在`main.py`中加载。你需要以类属性的形式记录训练后的模型参数供标签预测使用。
- `predict(test_features)`，在这个函数中你需要根据训练好的支持向量机来预测标签，`test_features`在`main.py`中加载。这个函数需要返回一个维度为`(测试样本数,)`的numpy数组，代表你对测试标签的预测值，这个预测值会和实际的测试标签`test_labels`进行比较。

其余成员变量以及类方法可以按需要添加。

助教已经在代码当中为你实现了`Linear`，`Poly`，`Gauss`核函数的计算供你调用，其接口为`SupportVectorMachine.KERNEL(x, y, [d=2], [sigma=1])`，`x`和`y`为两个输入向量，`d`和`sigma`为可选参数，使用多项式核的时候会使用到`d`参数，使用高斯核的时候会使用到`sigma`参数，实验过程中你不需要修改这两个可选参数，使用助教给出的默认值即可。

你需要在报告中贴出三种核函数的试验结果。并体会不同核函数的选用对于分类效果的影响。

SVM的训练过程中将涉及凸优化问题的求解，这里助教推荐使用`cvxpy`库进行求解。它是一个自动求解库，用户只需要设置优化变量以及约束，求解库将自动将其转化成标准形式并使用相应的求解算法。简单的使用方法可以参考这篇博客以及官方文档。

具体而言，使用`cvxpy`库求解凸优化问题的步骤为：创建变量，建立约束方程，建立目标函数，构造问题，求解问题，使用求解结果，一个简单的示例如下所示：

```
from cvxpy import *
# Create two scalar optimization variables.
x = Variable()
y = Variable()
# Create two constraints.
constraints = [x + y == 1, x - y >= 1]
obj = Minimize(square(x - y))
prob = Problem(obj, constraints)
prob.solve()  # Returns the optimal value.
print("status:", prob.status)
print("optimal value ", prob.value)
print("optimal var", x.value, y.value)

#output
# status: optimal
# optimal value 0.999999999761
# optimal var 1.00000000001 -1.19961841702e-11
```

作为提示，`cvxpy`库可以使用`@`表示矩阵乘法，同时提供一些便利的计算函数，如`cvxpy.sum()`表示求和，`cvxpy.quad_form()`表示二次型。同时，调用`solve()`时可以提供`verbose=True`参数来打印问题求解的详细过程，便于在问题求解失败时进行debug；还可以提供`solver=<solver>`参数来指定求解器。

本次实验推荐使用`1.1.13`版本的`cvxpy`库，使用高版本可能导致目标函数报错为非DCP形式。

```
    raise DCPError(
cvxpy.error.DCPError: Problem does not follow DCP rules. Specifically:
The objective is not DCP. Its following subexpressions are not:
QuadForm(var1, [[ 9.78858442e+01  9.64877589e+00  7.86485043e+01 ... -5.15200467e+01
  -2.38080223e+01 -1.51178300e+01]
```

文件说明

传统机器学习部分的文件说明：

1. `DecisionTree.py`：在这个文件中你需要实现决策树算法。
2. `SVM.py`：在这个文件中你需要补全软间隔支持向量机的实现。
3. `main.py`：`test_decisiontree`与`test_svm`函数将自动加载数据集，运行你实现的决策树或svm算法，并且对你预测的标签和真实标签进行比较，计算并打印准确率。`main.py`无需进行修改。
4. `utils.py`:一些其他用到的函数，无需进行修改。

以下是数据集相关文件介绍，由于我们已经帮你完成了数据加载，所以这部分信息对于实验而言并非必要，仅写出来共大家参考。

5. `dataset/dt/dt_train.data`：决策树训练数据。总共有221行，即221个训练样本。每行有10个整型数值，其中第1个数值是标签，其余9个数值是属性取值。
6. `dataset/dt/dt_test.data`：决策树测试数据。共有56个测试样本，每行有10个整型数值，第1个数值是标签值，其余9个数值是属性取值。**我们保证在训练数据中没有出现的属性取值不会在测试数据中出现。**
7. `dataset/svm/svm_train_data.csv`：SVM使用的训练用样本，共269个，第一行为表头，此后每一行有7个`float64`类型数值，用“,”进行分割。
8. `dataset/svm/svm_test_data.csv`：SVM使用的测试用样本，共30个样本，格式同训练数据。
9. `dataset/svm/svm_train_label.csv`：SVM使用的训练用样本的标签，第一行为表头。
10. `dataset/svm/svm_test_label.csv`：SVM使用的测试用样本的标签，第一行为表头。

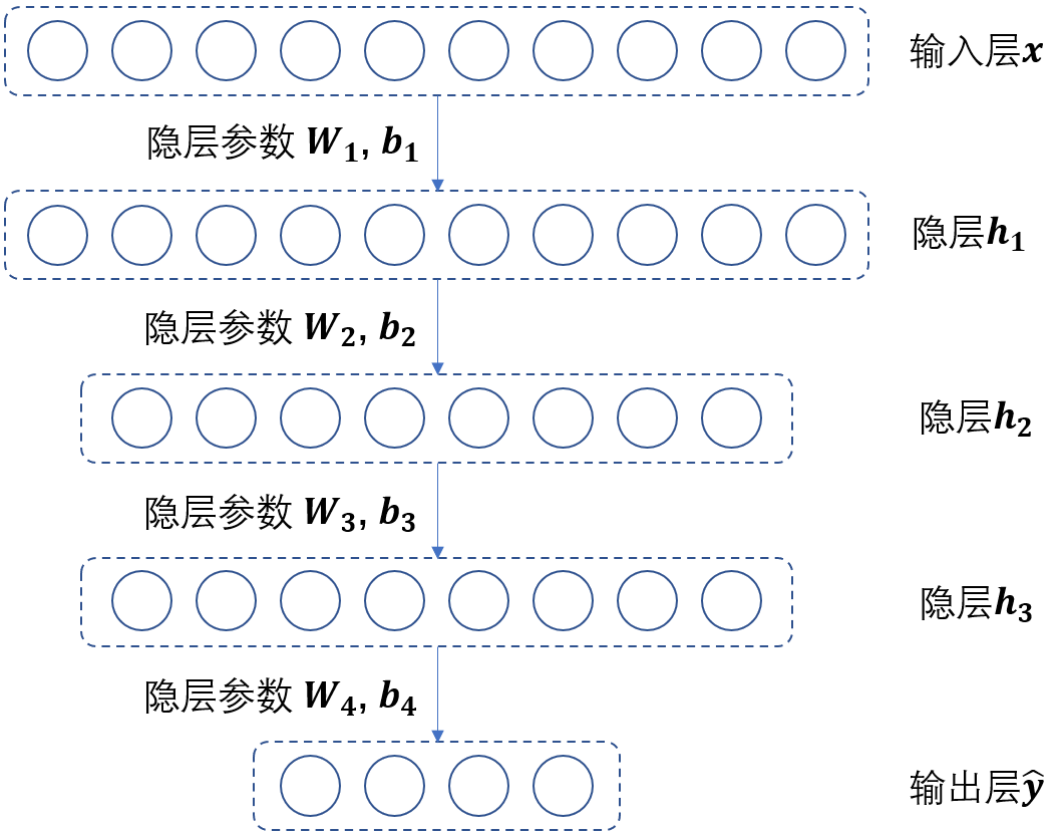
3. 深度学习

3.1 手写感知机模型并进行反向传播

实验目的：考察同学们对矩阵链式求导的掌握

实验内容：实现一个5层的感知机模型（**输入层为10，隐层神经元设置为10，8，8，输出层为4，**即输入的特征个数为10，输出的类别个数为4，激活函数设置为**tanh**）；实现mlp前向传播算法（1分）；实现反向传播和梯度下降算法（1分）。

感知机模型：



前向传播

$$
\begin{aligned}
\mathbf{h_1} &= s_1(\mathbf{W_1 x} + \mathbf{b_1}) \\
\mathbf{h_2} &= s_2(\mathbf{W_2 h_1} + \mathbf{b_2}) \\
\mathbf{h_3} &= s_3(\mathbf{W_3 h_2} + \mathbf{b_3}) \\
\hat{\mathbf{y}} &= s_4(\mathbf{W_4 h_3} + \mathbf{b_4}) \\
l(\hat{\mathbf{y}}, \mathbf{y}) &= CrossEntropyLoss(\hat{\mathbf{y}}, \mathbf{y}) = -\log(\hat{y}_t)
\end{aligned}
$$

其中，$\mathbf{y}$是样本类别的one-hot向量表示，t是样本所处的类别，$\hat{y}_t$是$\hat{\mathbf{y}}$的第t个分量。

反向传播

链式法则的展开形式：

$$
\begin{aligned}
\frac{\partial L}{\partial \mathbf{W_4}} &= (l' {s_4}') \mathbf{h_3^T}, \frac{\partial L}{\partial \mathbf{b_4}} = l' {s_4}' \\
\frac{\partial L}{\partial \mathbf{W_3}} &= (\mathbf{W_4^T}(l' {s_4}') \odot {s_3}') \mathbf{h_2^T}, \frac{\partial L}{\partial \mathbf{b_3}} = \mathbf{W_4^T}(l' {s_4}') \odot {s_3}' \\
\frac{\partial L}{\partial \mathbf{W_2}} &= (\mathbf{W_3^T}(\mathbf{W_4^T}(l' {s_4}') \odot {s_3}') \odot {s_2}') \mathbf{h_1^T}, \frac{\partial L}{\partial \mathbf{b_2}} = \mathbf{W_3^T}(\mathbf{W_4^T}(l' {s_4}') \odot {s_3}') \odot {s_2}' \\
\frac{\partial L}{\partial \mathbf{W_1}} &= (\mathbf{W_2^T}(\mathbf{W_3^T}(\mathbf{W_4^T}(l' {s_4}') \odot {s_3}') \odot {s_2}') \odot {s_1}') \mathbf{x^T}, \frac{\partial L}{\partial \mathbf{b_1}} = \mathbf{W_2^T}(\mathbf{W_3^T}(\mathbf{W_4^T}(l' {s_4}') \odot {s_3}') \odot {s_2}') \odot {s_1}'
\end{aligned}
$$

其中⊙表示按位乘，并且：

$$
\begin{aligned}
s_4(x_1, x_2, x_3, x_4) = Softmax(x_1, x_2, x_3, x_4) &= \frac{1}{e^{x_1} + e^{x_2} + e^{x_3} + e^{x_4}}(e^{x_1}, e^{x_2}, e^{x_3}, e^{x_4}), \\
s_1 = s_2 = s_3 &= \tanh(\cdot) \\
{s_1}' = {s_2}' = {s_3}' &= 1 - \tanh^2 \\
(l' {s_4}')_i &= \begin{cases} \hat{y}_i - 1 & i = t \\ \hat{y}_i & i \neq t \end{cases}
\end{aligned}
$$

使用反向传播从输出层到输入层计算上述梯度，可以参考https://blog.csdn.net/xholes/article/details/78461164。

梯度下降：

$$\mathbf{W}_i = \mathbf{W}_i - \eta \frac{\partial L}{\partial \mathbf{W}_i}$$
$$\mathbf{b}_i = \mathbf{b}_i - \eta \frac{\partial L}{\partial \mathbf{b}_i}$$

实验要求：通过矩阵运算实现模型；实现各参数的梯度计算，给出各参数矩阵的梯度，并与pytorch自动计算的梯度进行对比；实现梯度下降算法优化参数矩阵，给出loss的训练曲线。

禁止直接调包，例如自动求导，torch.nn，torch.optim模块等都禁止使用。模型的输入使用随机生成的100个样本，每个样本的特征是10维。即输入的数据维度为(100,10)，采用one-hot编码的label维度为(100,4)，loss使用交叉熵损失。文件命名为MLP_manual.py(提供了基础的代码框架，也可按照自己需求更改)。报告中需给出loss训练曲线和与pytorch自动求导的对比，以及W, b的最终结果。

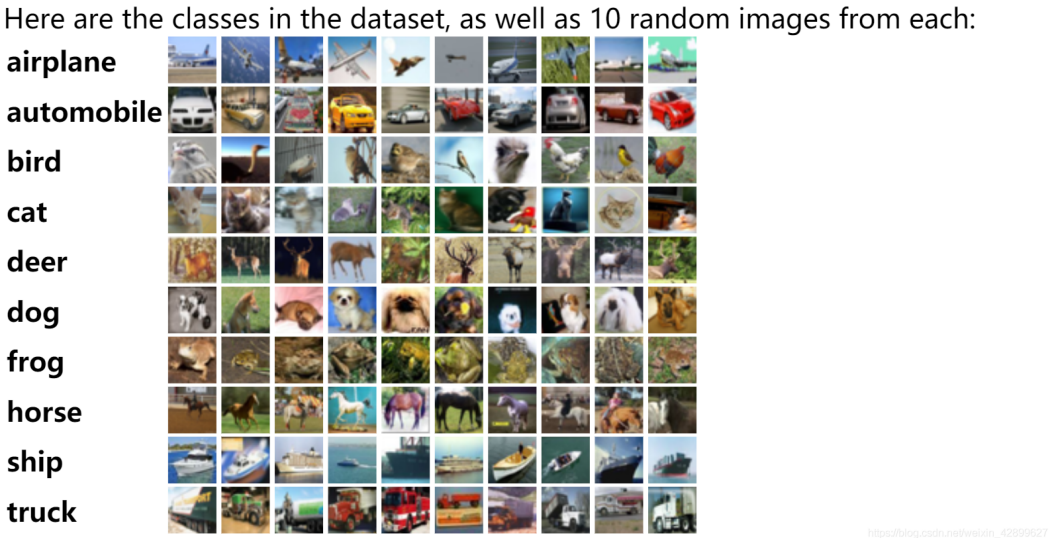
3.2 实现一个卷积神经网络

实验目的：对卷积神经网络的初步掌握，实现图像分类。
数据集介绍：CIFAR10数据集。由于pytorch提供的下载接口不一定稳定，同学们可以自行下载数据集，数据集下载地址：

链接：https://rec.ustc.edu.cn/share/6e2aadd0-da71-11ec-a601-7b38acc41387 密码：lab2

下载完后将解压后的`data/`目录放置在`src2/`目录下即可。

有稳定网络连接的同学也可以直接在代码中将`download`参数改为`True`自动下载数据集。



实验内容：构建一个给定架构的卷积神经网络模型，每个人根据学号选择需要实现的模型，并在CIFAR10数据集上进行测试。请根据自己学号的最后两位相加然后模6再加1计算得出自己的模型，例如：PB******12，那么(1+2)%6+1=4，即选择列表中第四个模型。请认真阅读表格说明，**注意并非所有网络结构参数都已给出，但均可根据已有参数推出。**

编号	layer1	layer2	layer3	layer4	layer5	layer6	layer7	layer8	激活函数
--	2d卷积	池化	2d卷积	池化	卷积	线性层	线性层	线性层	--
1	16,5	最大池化	32,5	最大池化	-	120	84	10	tanh
2	24,9	最大池化	32,3	最大池化	-	108	72	10	relu
3	16,5	平均池化	32,5	平均池化	-	120	84	10	relu
4	24,9	平均池化	32,3	平均池化	-	108	72	10	tanh
5	12,5	最大池化	24,3	最大池化	32,2	108	84	10	relu
6	10,5	平均池化	20,3	平均池化	32,3	120	72	10	tanh

表格说明：

1. 2d卷积（a,b) a:kernel个数，b:kernel size为（b,b)
2. 池化采取两种方式，默认池化大小为2
3. 线性层 b:output channel的大小，**需要自己求出input channel的大小**
4. 激活函数 在每个卷积和线性层后都加入激活函数，池化层无需添加
5. -表示没有

实验要求：仅可以在注释的方框中书写你的代码，**不能修改其他代码，不能超出方框外书写。**报告中需要贴上终端输出的截图。

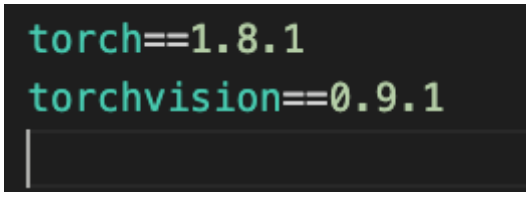
4. 提交内容与方式

DDL：2022年6月28日 23:59:59。逾期扣分。

提交方式：压缩文件，命名为 LAB2_PBxxxxxxxx_王二.zip，上传到bb系统作业区。

提交内容与格式：

- 在实验报告中总结以上的实验结果，并对实验结果进行分析。如果实验中存在一些细节在实验要求中未提及，请在报告中说明你的处理方法及原因。如果你训练出的分类器效果不好，想办法改进你的分类器，例如改善特征选取方法。比较改进前后模型的变化并对你的改进做出分析。要求pdf格式。请注意实验报告中不仅要有结果截图，要有适当的文字说明！
- 提供一个描述所有所需依赖包的 requirements.txt，手动列入代码中用到的所有非标准库及版本(推荐) 或者使用命令 `pip freeze > requirements.txt`，**删除掉没用到的包**；例如下面这张图片。



- 你的提交文件应按如下结构：

```
-src1

    --DecisionTree.py

    --SVM.py

-src2

    --MLP_manual.py

    --MyNet.py

-实验报告.pdf
```

附录

Python 安装

已经安装并使用过 Python 的同学可以跳过这一节。对于没有用过 Python 的同学，助教列出了三个**可选的**方式。当然只要你的程序能在合适的版本跑通，我们并不关心你是如何安装的。

1. 安装 Python 官方版本:

   Linux 用户可以通过包管理器直接安装。
   其他系统的用户可以参考官方的 tutorials
   ，当然网上有很多安装教程，可以自己找一篇博客看一看。

2. 通过 Anaconda:

   Anaconda 清华镜像安装包：https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/ 在这个链接中选择自己的系统/架构安装。

Anaconda3-2022.05-Windows-x86.exe	487.8 MiB	2022-05-11 02:36
Anaconda3-2022.05-Windows-x86_64.exe	593.9 MiB	2022-05-11 02:36
Anaconda3-2022.05-MacOSX-x86_64.sh	584.0 MiB	2022-05-11 02:36
Anaconda3-2022.05-MacOSX-arm64.pkg	428.3 MiB	2022-05-11 02:36
Anaconda3-2022.05-MacOSX-x86_64.pkg	591.0 MiB	2022-05-11 02:36
Anaconda3-2022.05-MacOSX-arm64.sh	420.0 MiB	2022-05-11 02:36
Anaconda3-2022.05-Linux-x86_64.sh	658.8 MiB	2022-05-11 02:35
Anaconda3-2022.05-Linux-s390x.sh	279.8 MiB	2022-05-11 02:35
Anaconda3-2022.05-Linux-ppc64le.sh	367.3 MiB	2022-05-11 02:35
Anaconda3-2022.05-Linux-aarch64.sh	567.6 MiB	2022-05-11 02:35

   安装过程也有很多博客教程，这里贴一篇 Windows 系统的:https://blog.csdn.net/qq_45344586/article/details/124028689
   Linux/MacOS 用户按照安装脚本提示操作即可。

3. 直接使用 JetBrains 的 Pycharm IDE: https://www.jetbrains.com/zh-cn/pycharm/

numpy 初步使用

助教在这里列出一些 numpy 数组的索引方法。本实验不要求很高的 numpy 技巧，要使用的 API 一般都可以通过搜索引擎和官方的 API Reference找到: https://numpy.org/doc/stable/reference/index.html#

假设你初始化了一个 numpy 二维数组`x = np.array([[1,2],[3,4],[1,4]])`

- 索引数组的第N行: `x[N-1]`。如`x[0]`.返回`array([1, 2])`
- 索引数组的第N列: `x[:,N-1]`。如`x[:,0]`返回`array([1, 3, 1])`

- 按条件索引：如返回第二列是元素是4的所有行：`x[x[:,1] == 4]`，返回`array([[3, 4], [1, 4]])`

pytorch 安装

pytorch官网

pip 安装

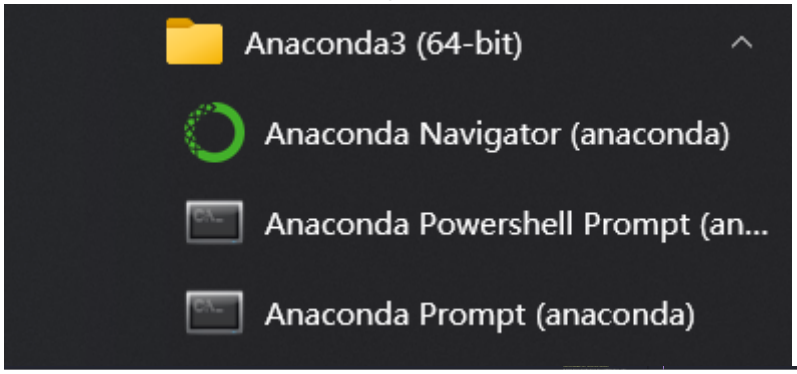
不使用Anaconda的同学可以直接使用pip安装。终端内输入命令`pip3 install torch==1.8.2+cpu torchvision==0.9.2+cpu torchaudio===0.8.2 -f https://download.pytorch.org/whl/lts/1.8/torch_lts.html`

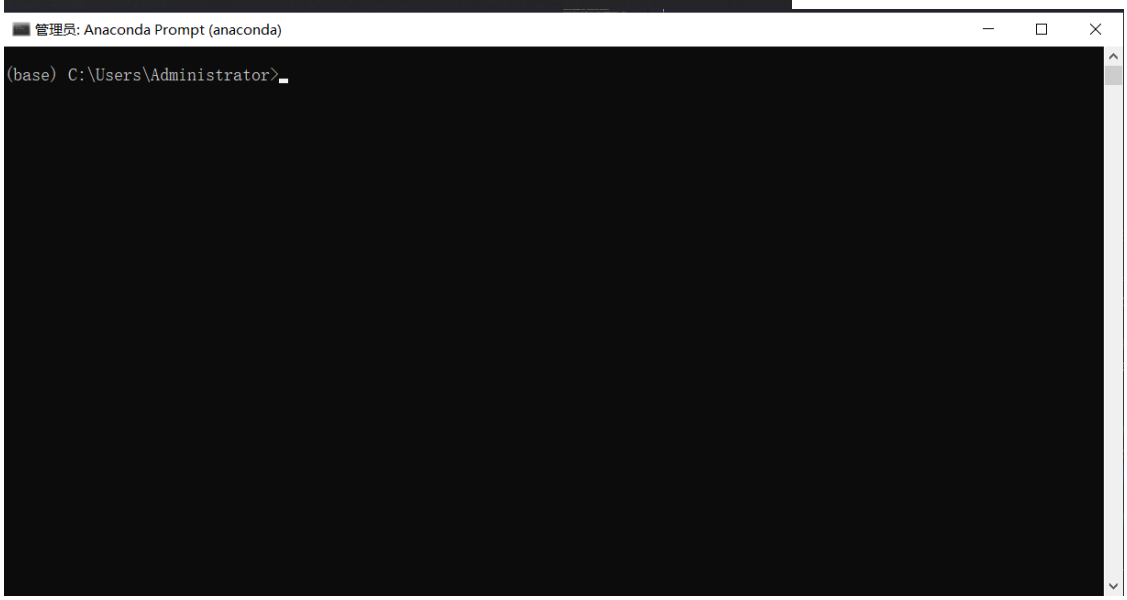
Anaconda 安装

Windows 版 Anaconda 安装及环境变量配置
参考博客https://blog.csdn.net/fan18317517352/article/details/123035625

虚拟环境创建

1. 在开始菜单找到 Anaconda Prompt (anaconda) 命令行工具，并打开。



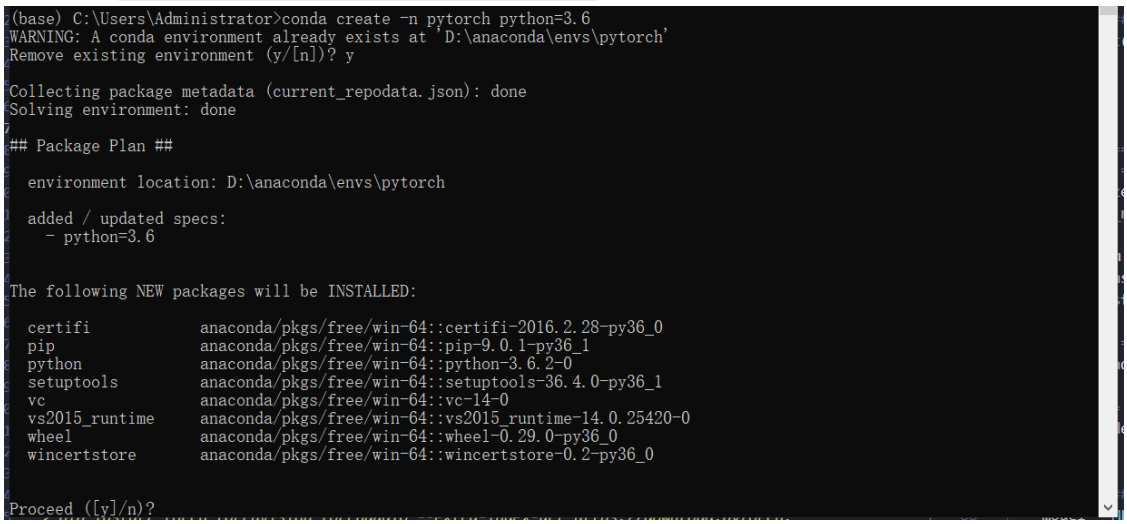


2. 输入命令 `.condarc` 用记事本打开 .condarc 文件，修改文件内容，更改为清华源：

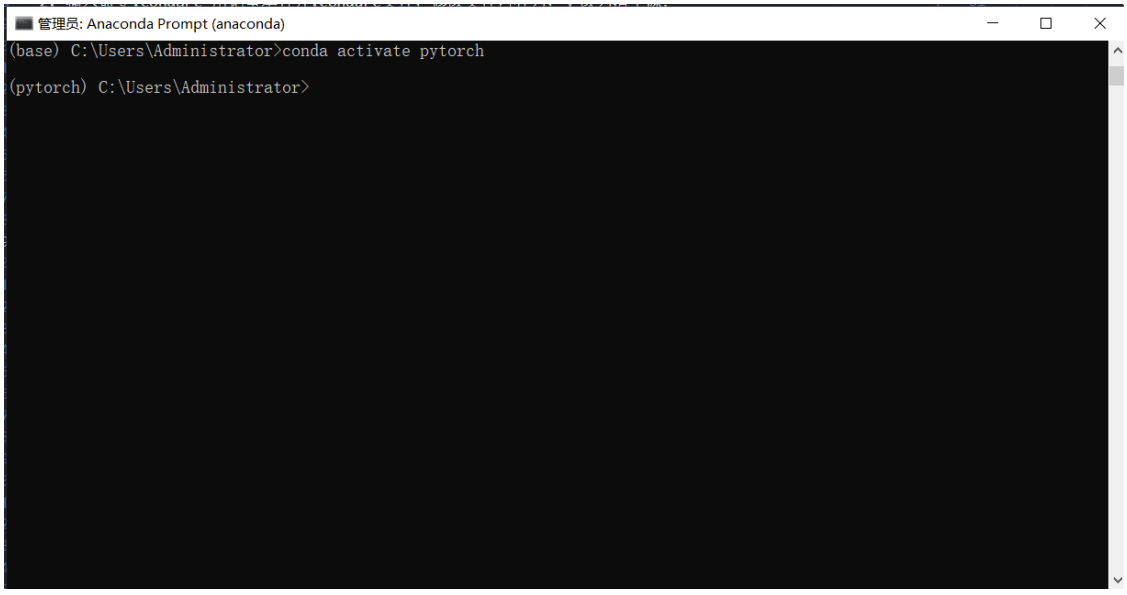
```
ssl_verify: true
show_channel_urls: true

channels:
  -http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/win-64/
  -http://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/win-64/
```

3. 输入命令 `conda create -n pytorch python=3.6` 创建虚拟环境。



4. 输入命令 `conda activate pytorch` 激活虚拟环境。



5. 输入命令 `conda install pytorch\==1.8.1 torchvision\==0.9.1 torchaudio==0.8.1 cpuonly -c pytorch` 安装cpu版本 pytorch1.8.1 torchvision 0.9.1 torchaudio 0.8.1