Homework 3 by 161220097 戚赟

Released on 2019 年 4 月 16 日
Due at 20:00, April 17, 2019

1 [15pts] Decision Tree I

(1) [5pts] Assume there is a space contains three binary features X, Y, Z and the objective function is $f(x, y, z) = \neg(x \text{ XOR } y)$. Let H denotes the decision tree constructed by these three features. Please answer the following question: Is function f realizable? If the answer is yes, please draw the decision tree H otherwise please give the reason.
(2) [10pts] Now we have a dataset show by Table.1:

Table 1:example dataset

X	Y	Z	f
1	0	1	1
1	1	0	0
0	0	0	0
0	1	1	1
1	0	1	1
0	0	1	0
0	1	1	1
1	1	1	0

Please use Gini value as partition criterion to draw the decision tree from the dataset. When Gini value is same for two features, please follow the alphabetical order.

Solution:

(1)函数f是可以利用决策树来实现的，实现的决策树H如下：

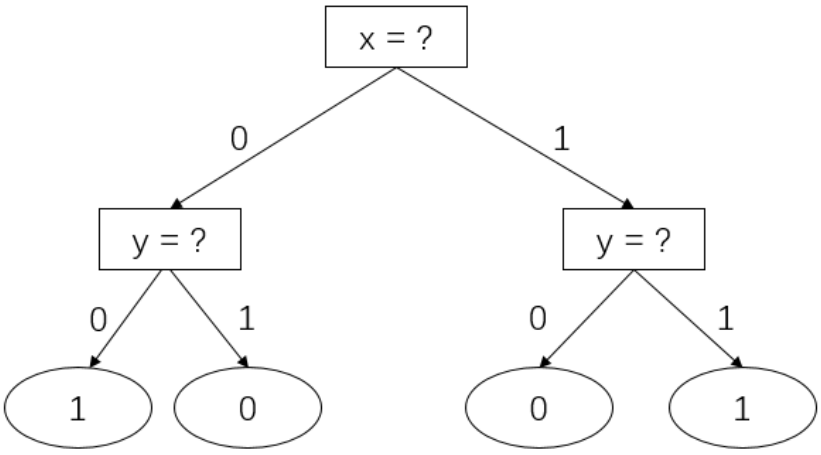

图 1: My Decision Tree.

(2) 首先对于一个特征a的基尼指数的计算公式如下：

$$Gini\_index(D,a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} Gini(D^v) \tag{1.1}$$

计算各项指标的Gini指数，如下：

$$Gini(D,X) = \frac{4}{8} Gini(D\{X=1\}) + \frac{4}{8} Gini(D\{X=0\}) \tag{1.2}$$

$$= \frac{1}{2} \times (1 - (\frac{2}{4})^2 * 2) * 2 = \frac{1}{2} \tag{1.3}$$

$$Gini(D,Y) = \frac{4}{8} Gini(D\{Y=1\}) + \frac{4}{8} Gini(D\{Y=0\}) \tag{1.4}$$

$$= \frac{1}{2} \times (1 - (\frac{2}{4})^2 * 2) * 2 = \frac{1}{2} \tag{1.5}$$

$$Gini(D, Z) = \frac{6}{8}Gini(D\{Z = 1\}) + \frac{2}{8}Gini(D\{Z = 0\}) \tag{1.6}$$

$$= \frac{3}{4} \times (1 - (\frac{2}{3})^2 - (\frac{1}{3})^2) + \frac{1}{2} \times (1 - 1) = \frac{1}{3} \tag{1.7}$$

由于属性Z的Gini指数是最小的，所以选择Z作为根节点进行划分。

当Z=0时，f=0，当z=1时，计算X，Y的Gini指数，经计算，两者相等皆为$\frac{4}{9}$，于是按照字典序进行选择，选择X作为第二划分元素。所以，决策树如下：

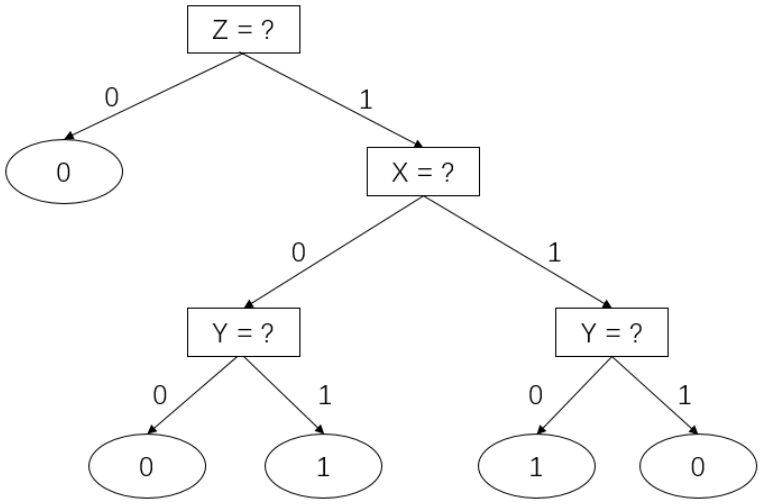

图 2: Decision Tree by GINI

2 [25pts] Decision Tree

Consider the following matrix:

$$\begin{bmatrix} 24 & 53 & 23 & 25 & 32 & 52 & 22 & 43 & 52 & 48 \\ 40 & 52 & 25 & 77 & 48 & 110 & 38 & 44 & 27 & 65 \end{bmatrix}$$

which contains 10 examples and each example contains two features x_1 and x_2. The corresponding label of these 10 examples as follows:

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

In this problem, we want to build a decision tree to do the classification task. (1) [5pts] Calculate the entropy of the root node.
(2) [10pts] Building your decision tree. What is your split rule and the classification error?
(3) [10pts] A multivariate decision tree is a generalization of univariate decision trees, where more than one attribute can be used in the decision for each split. That is, the split need not be orthogonal to a feature's axis.

Building a multivariate decision tree where each decision rule is a linear classifier that makes decisions based on the sign of $\alpha x_1 + \beta x_2 - 1$. What is the depth of your tree, as well as α and β?

Solution:
(1)计算根节点的熵值：

$$Ent(D) = -\sum_{i=1}^{|y|} p_k \log p_k = -\frac{6}{10}\log\frac{6}{10} - \frac{4}{10}\log\frac{4}{10} = -\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5} \tag{2.1}$$

(2)我建立的决策树如下：

分类依据：根据X的值进行判断，直观的来判断来说，当X模10之后，如果余数是3，则label为0，但是对于X=52，label也是0.所以要使用Y进行进一步的判断，如果结果是Y是奇数，label为0，否则label为1。
分类错误：对于给出的10个样本，构造的决策树能够正确的给出label，没有分类错误。

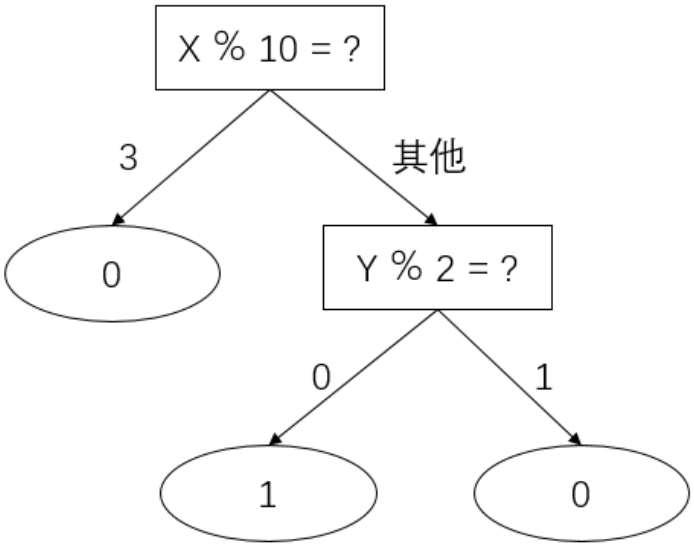

图 3: My Decision Tree

(3)我们需要构建的多变量决策树。对于这10个点进行分类描绘，对应的图如下图4所示. 则我们可以知道，我们可以利用一条直线将两个点进行划分，直线的形式是$\alpha X + \beta Y - 1$,输入大于等于0,label为1，否则为0. 我们只需要确定α和β的值即可.
计算出来可以的一个值为$\alpha = 60$, $beta = 50$，如果sign $\alpha X + \beta Y - 1 > 0$，标签为0，否则为1. 分类情况和多变量决策树如下图5和6所示：

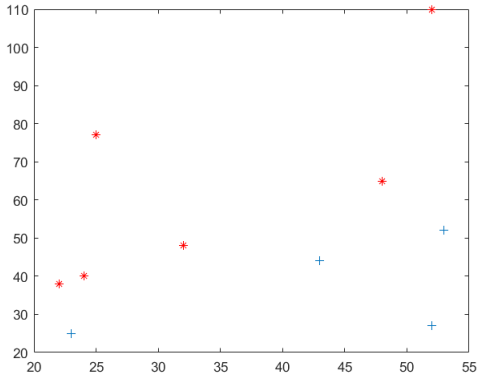

图 4: My Decision Tree

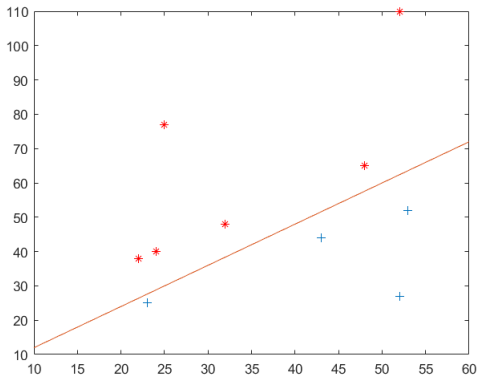

图 5: My Decision Tree

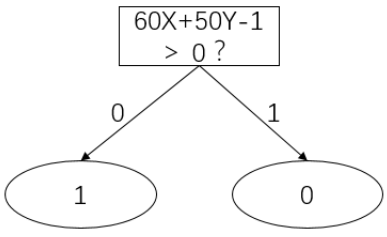

图 6: My Decision Tree

3 [25pts] Convolutional Neural Networks

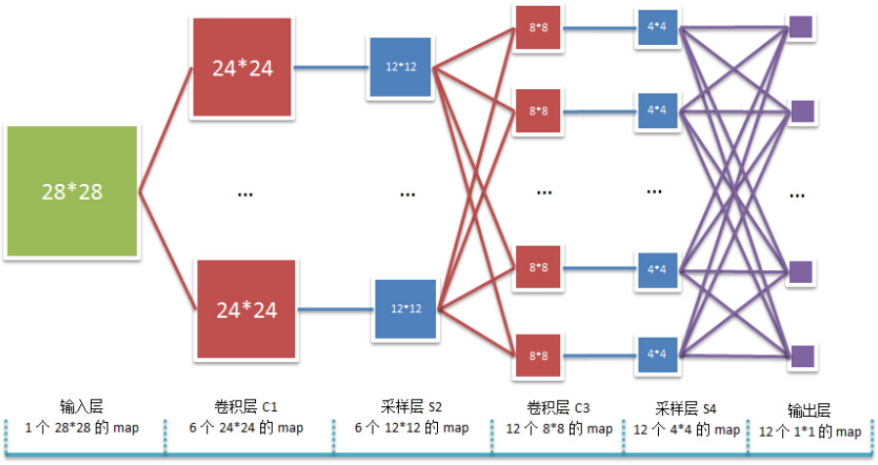

图 7: CNN.

Using Fig. 11 as an example. Assuming that the loss function of the convolutional neural network is cross-entropy:

(1) [5 pts] Briefly describe the forward propagation process;

(2) [5 pts] What is the difference between Relu and Sigmoid activation functions;

(3) [5 pts] Derivation of the fully connected layer;

(4) [5 pts] Derivation of the pooling layer with average pooling;

(5) [5 pts] Derivation of the convolutional layer with Relu;

Solution:
(1)简单的介绍一下，卷积神经网络(CNN)的正向传播过程：
利用Figure1给出的实例来进行解释
1.输入层：输入1个28*28的map
2.输入层到卷积层：假定不进行补充，则使用了6个5*5的过滤器(filter)进行卷积运算，得到了6个24*24的map

3.卷积层到采样层(也叫pooling层)：特征提取为将2*2的方阵利用方法提取成一个数，一般来说是取平均或者取最大值，这一步减少特征值，防止过拟合，得到6个12*12的map
4.采样层到卷积层:假定不进行补充，则使用了12个filter，得到了12个8*8的map
5.卷积层到采样层：与3相同，得到12个4*4的map
6.最后使用了12个4*4的卷积矩阵，对于12个矩阵分布进行卷积运算，每个卷积矩阵得到的结果相加再除以12，最后得到了12个结果，正向过程结束.

(2)说明Relu函数和Sigmod函数的不同之处
两者为

$$Relu : f(x) = max(x, 0) = \begin{cases} 0, x < 0 \\ x, x > 0 \end{cases} \tag{3.1}$$

$$Sigmod : S(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

Relu函数的优势：
第一，防止梯度弥散，sigmod函数只有在0附近有较好的激活性，接近正负饱和区时，变换太缓慢，导数趋于0，造成梯度弥散，而relu函数在大于0的部分梯度为常数,所以不会产生梯度消失的问题。
第二，稀疏性，Relu会使一部分神经元的输出为0，这样就造成网络的稀疏性，缓解了过拟合问题的发生，relu函数在负半区的导数为0,所以一旦神经元激活值进入负半区,那么梯度就会为0,也就是说这个神经元不会经历训练。
第三，加快计算，relu函数的导数计算更快,而sigmoid函数要进行浮点四则运算。

(3)对于全连接层的W, b的计算
全连接层的反向求导是与普通神经网络的反向求导是一致的，因为前导过程没有什么不同，设W_i和b_i为全连接层的参数,a_i为输入，z_i为输出则：

$$\Delta W_i = \frac{\partial J(W,b)}{\partial W_i} = \frac{\partial J(W,b)}{\partial z_i}\frac{\partial z_i}{\partial W_i} = \delta^i (a^{i-1})T \tag{3.3}$$

$$\because J(W,b)\ is\ cross\_entropy \tag{3.4}$$

$$\therefore J(W,b) = -\sum_i z_i \log(p_i) \tag{3.5}$$

$$\tag{3.6}$$

$$\because \frac{\partial J(W,b)}{\partial z_i} = -\sum_k z_k \frac{\partial \log(p_k)}{\partial z_i} \tag{3.7}$$

$$= -\sum_k z_k \frac{\partial \log(p_k)}{\partial p_k}\frac{\partial p_k}{\partial z_i} \tag{3.8}$$

$$= -\sum y_k \frac{1}{p_k} \times \frac{\partial p_k}{\partial z_i} \tag{3.9}$$

$$= -z_i(1-p_i) - \sum_{k\neq i} z_k \frac{-p_i * p_k}{p_l} \tag{3.10}$$

$$= p_i(y_i + \sum_{k\neq 1} y_k) - y_i \tag{3.11}$$

$$= p_i - y_i \tag{3.12}$$

$$\therefore \delta^i = p_i - y_i \tag{3.13}$$

$$\therefore \Delta W_i = (p_i - y_i)(a^{i-1})^T \tag{3.14}$$

$$\therefore \Delta b_i = \frac{\partial J(W,b)}{\partial b_i} = \delta^i = p_i - y_i \tag{3.15}$$

(4)对于池化层计算，如果池化层后面一层为全连接层，那么就直接可以推出。所以我们只需要考虑因此只需讨论下一层l+1为卷积层的情形，上一层l-1也为卷积层，该情形下有：

池化层l的各个神经元的δ只和l+1层的相关神经元有关

池化层l到卷积层l+1做了窄卷积运算，使得矩阵维度减小，因此，δ_i^{l+1}需要与相应的卷积核做宽卷积运算使得矩阵维度扩展回去

池化区域大小是2x2，δ^l的第k个子矩阵为:

$$\begin{bmatrix} 2 & 4 \\ 8 & 6 \end{bmatrix} \tag{3.16}$$

池化区域为4×4，则先做还原得到

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 8 & 6 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.17}$$

因为采用的AVERAGE策略，所以矩阵应该是

$$\begin{bmatrix} 0.5 & 0.5 & 1 & 1 \\ 0.5 & 0.5 & 1 & 1 \\ 2 & 2 & 1.5 & 1.5 \\ 2 & 2 & 1.5 & 1.5 \end{bmatrix} \tag{3.18}$$

因此：

$$\delta_k^{l-1} = \frac{\partial J(W,b)}{\partial a_k^{l-1}} \frac{\partial a_k^{l-1}}{\partial z_k^{l-1}} \tag{3.19}$$

$$\delta^{l-1} = usample(\delta^l) \odot \delta'(z_k^{l-1}) \tag{3.20}$$

其中usample函数完成了池化误差矩阵放大与误差重新分配的逻辑。

(5)卷积神经网络的卷积层的递推的方式与全连接的神经网络的不同之处在于：

1.卷积层是通过张量卷积，或者说若干个矩阵卷积求和而得的当前层的输出，这和DNN不同。这样在卷积层反向传播的时候，上一层的δ^{l-1}递推计算方法要有所改变。

2.对于卷积层，由于W使用的运算是卷积，那么从δ^l推导出该层的所有卷积核的W,b的方式也不同。

所以对于

$$\delta^l = \frac{\partial J(W,b)}{\partial z^l} \tag{3.21}$$

$$= \frac{\partial J(W,b)}{\partial z^{l+1}} \frac{z^{l+1}}{\partial z^l} = \delta^{l+1} \frac{\partial z^{l+1}}{\partial z^l} \tag{3.22}$$

$$\because a^{l-1} * W^l = z^l \tag{3.23}$$

$$\therefore \delta^{l-1} = \delta^l * rot180(W^l) \odot \sigma'(z^{l-1}) \tag{3.24}$$

$$\because function\ is\ Relu \tag{3.25}$$

$$\therefore \delta^{l-1} = \delta^l * rot180(W^l) \odot R(z^{l-1}) \tag{3.26}$$

$$\tag{3.27}$$

其中R()函数表示求导，具体是如果小于0，结果为0，否则结果为1.
对于含有卷积的式子求导时，卷积核被旋转了180度。
然后求W和b，则

$$\frac{\partial J(W,b)}{\partial W^l_{pq}} = \sum_i \sum_j (\sigma^l_{ij} a^{l-1}_{i+p-1,j+q-1}) \tag{3.28}$$

$$\therefore \frac{\partial J(W,b)}{\partial W^l} = a^{l-1}\delta^l \tag{3.29}$$

$$\frac{\partial J(W,b)}{\partial b^l} = \sum_{u,v} (\delta^l)_{u,v} \tag{3.30}$$

其中，δ由于已经知道，则4.5两题中没有给出具体式子.

4 [35 pts] Neural Network in Practice

In this task, you are asked to build a Convolutional Neural Networks (CNNs) from scratch and examine performance of the network you just build on **MNIST** dataset. Fortunately, there are some out-of-the-box deep

learning tools that can help you get started very quickly. For this task, we would like to ask you to work with the **Pytorch** deep learning framework. Additionally, Pytorch comes with a built-in dataset class for MNIST digit classification task in the **torchvision** package, including a training set and a validation set. You may find a pytorch introduction at here. Note that, you can use CPU or GPU for training at your choice.

Please find the detailed requirements below.

(1) [5 pts] You are encouraged to implement the code using *Python3*, implementations in any other programming language will not be judged. Please name the source file (which contains the main function) as *CNN_main.py*. Finally, your code needs to print the performance on the provided validation set once executed.

(2) [10 pts] Use any type of CNNs as you want and draw graphs to show your network architecture in the submitted report. You are encouraged to try more architectures.

(3) [15 pts] During training, you may want to try some different optimization algorithms, such as SGD, Adam. Also, you need to study the effect of learning rate and the number of epoch, on the performance (accuracy).

(4) [5 pts] Plot graphs (learning curves) to demonstrate the change of training loss as well as the validation loss during training.

Solution:

(1)我已经按照要求，利用*Python*3和*pytorch*工具，完成了CNN卷积神经网络的搭建，文件命名为：$CNN\_main.py$.其中具体的函数和参数如下：

全局变量参数：

epochs = 5 #跑的轮数

train_batch_size = 10 #batch的大小

test_batch_size = 1000 #测试的分batch的大小

log_interval = 10 #间隔记录

LR = 0.01 #学习率

SDG_momentum = 0.5 #SGD参数

重要的类:
class Net(nn.Module):继承一个网络模型进行数据的训练
重要的方法:
def train():进行训练
def test():进行测试

代码的执行过程如下:
1.利用torchvision进行数据集的获取
2.构建神经网络
3.构建损失函数
4.进行训练
5.测试并输出结果和图

代码对于测试集合能够输出测试的结果，主要输出准确率.

(2)经过筛选，我构建的神经网络如下：

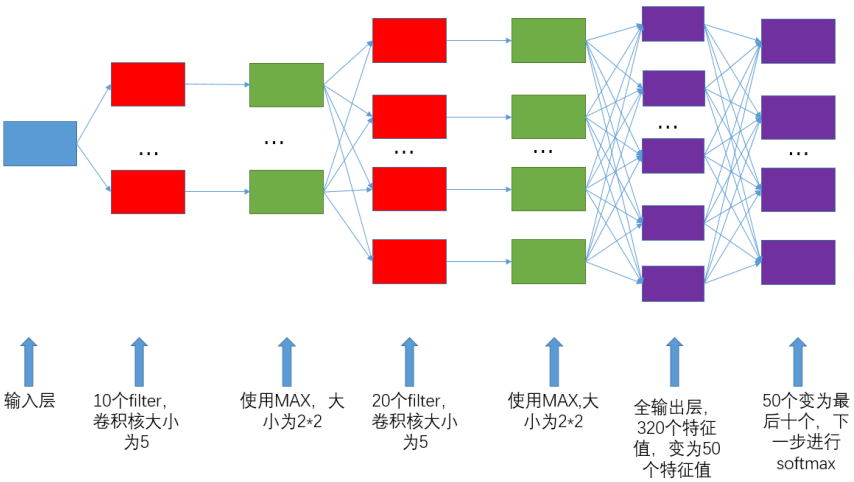

图 8: My CNN

其中我尝试不同的结构，比如pooling层使用AVER函数，或者改变卷积核的大小，效果会不同，但是这样的神经网络是我尝试的效果比较好的之一，下面会给出具体的比较。

(3)为了比较效果，利用控制变量法
1.测试不同结构，对于(1)中的全局变量，不做任何改变，为了效果明显，epoch设置为1，改变pooling层的策略为平均.

accuarcy1	accuracy2
97%	95%

单这次结果来看，利用max比ave效果会好

2.测试不同的optimization，除optimization之外，其余变量相同，结果为：

accuarcy1	accuracy2
97%	90%

单这次结果来看，SGD比Adam算法效果好

3.测试不同的学习率，设置学习率分别为0.01,0.005和0.02,结果为：

accuarcy1	accuracy2	accuracy3
97%	95%	95%

可知在epoch $=$ 1，利用SGD的情况下，学习率0.01会达到最佳效果，但仅仅是针对仅跑一轮的情况下，0.02步长太大达不到最低点，0.005又太小，导致无法达到最低。

4.测试不同的epoch数,设置为1,5,10,结果为：

accuarcy1	accuracy2	accuracy3
97%	98.29%	98.64%

可以看出来，随着多次跑动，其精确度会提升，但是提升的会原来越慢，所以我们需要在时间和准确度综合考量，选择最优的大小

5.测试不同的batch_size的大小，分别为10,50,100:

accuarcy1	accuracy2	accuracy3
96.52 %	91.04%	87.58%

可见，batch_size的数量越多，测试的精度会变低，于是采用大小为10

(5)在经过测试之后，取如下的参数会达到比较好的效果，保证时间和精度两开花

epochs = 10 #跑的轮数

train_batch_size = 10 #batch的大小

log_interval = 10 #间隔记录

LR = 0.01 #学习率

SDG_momentum = 0.5 #SGD参数

采用SDG算法，精度达到98.65%

其中算法执行过程的损失函数图为:

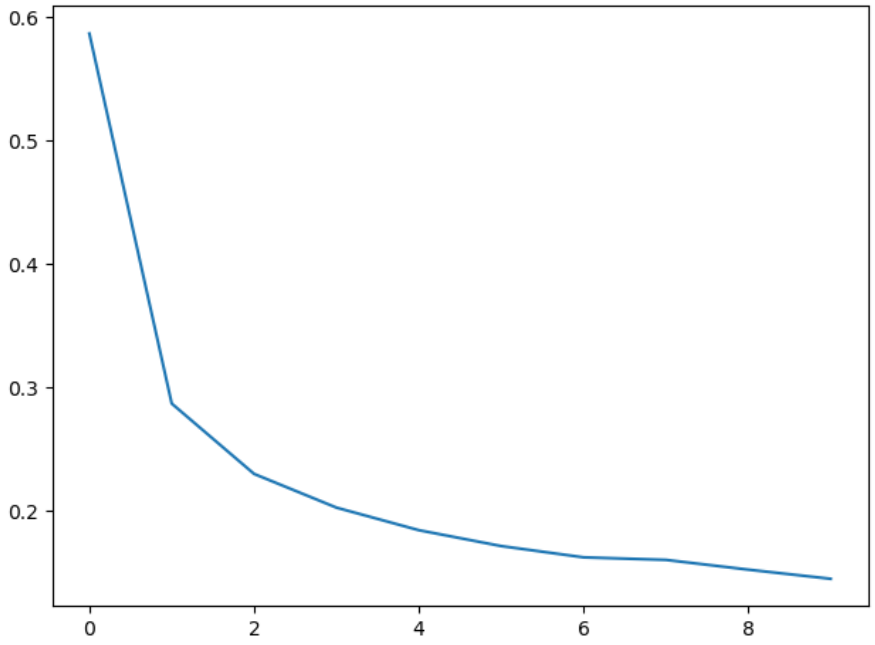

图 9: train loss

算法测试的结果为:

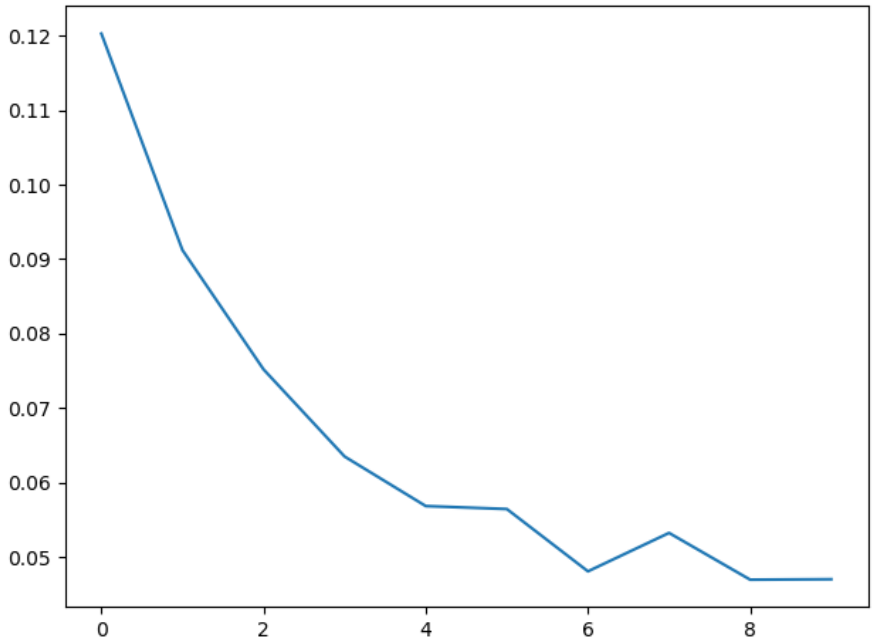

图 10: validation loss

其中取得是每个epoch的平均的loss，绘成的图像，可以看下降趋势都是非线性递减，当epoch设置为60, Train loss会更加贴近一条曲线，如下：

作业心得体会

1.对于第三题的3.4.5小问，不是很理解，网上也没有具体的可理解的讲解，希望助教能够给与帮助和讲解。

2.对于代码题，感谢刘轩同学给予我的帮助，通过这条题目，我学会了利用pytorch这个现成的工具来进行神经网络的搭建和测试，重要的是不断的对于参数的设置的比较，在其中我学会了很多，运用了控制变量的思想，直观的感受了参数变化对于实验结果的影响。最后得出比较好的神经网络，学会了很多关于参数的理解。

3.由于下的是pytorch的无GPU版本，导致我没有利用我的显卡，所以比较慢，以后会尝试利用GPU进行处理

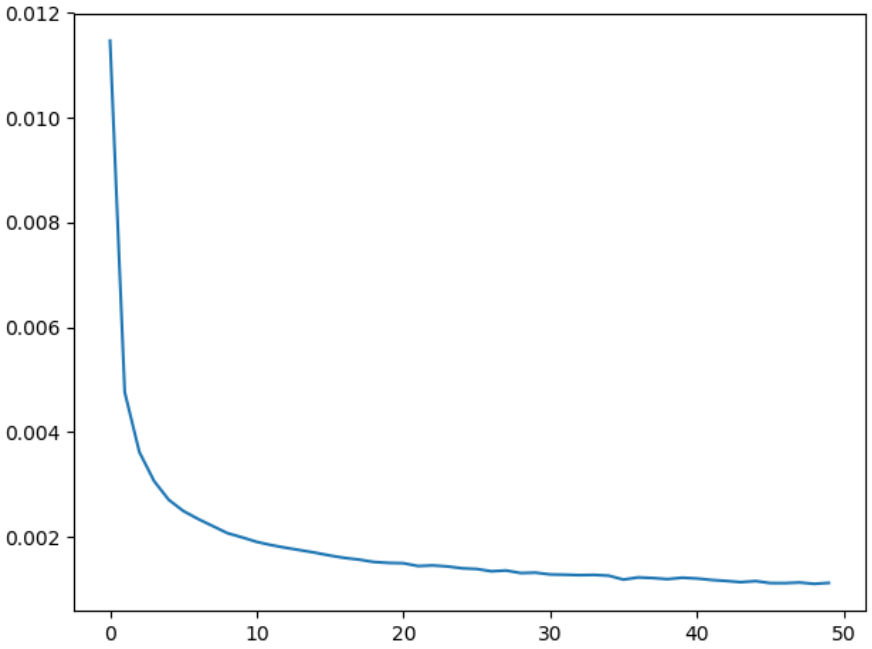

图 11: train loss2

参考：

[1]CNN神经网络推导https://www.cnblogs.com/pinard/p/6494810.html

[2]pytorch安装https://blog.csdn.net/nnUyi/article/details/78471326

[3]pytorch的使用https://blog.csdn.net/Teeyohuang/article/details/79242946