机器学习导论

姓名：殷天润　　**学号：**171240565

2019 年 4 月 17 日

请独立完成作业，不得抄袭。
若得到他人帮助, 请致谢。
若参考了其它资料，请给出引用。
鼓励讨论，但需独立书写解题过程。

第一部分　作业

题目 (ML problem 1)

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.

解答：

1. Yes. It is realizable. The decision is as following:

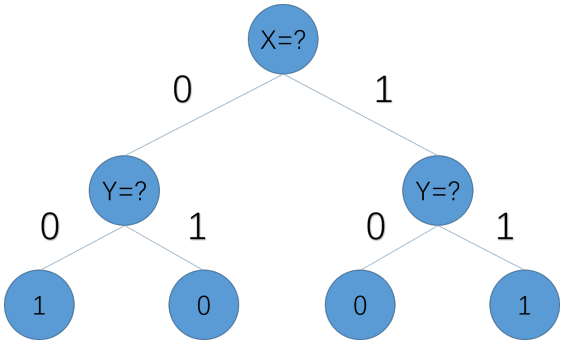


图 1: (Desicion tree of Pro 1.1)

2. 选根节点:

   (a) Gini_index(Z)=$\frac{2}{8}*(1-1)+\frac{6}{8}*(1-(\frac{4}{6})^2-(\frac{2}{6})^2)=\frac{1}{3}$

   (b) Gini_index(Y)=$\frac{1}{2}*(1-0.5^2-0.5^2)*2$=0.5

   (c) Gini_index(X)=$\frac{1}{2}*(1-0.5^2-0.5^2)*2$=0.5

   因此根节点为Z;

   Z=0时,所有的f值都是0不用继续分类;

   Z=1时,

   (a) Gini_index(Y)=$\frac{1}{2}*(1-(\frac{2}{3})^2-(\frac{1}{3})^2)*2$=0.44444

   (b) Gini_index(X)=$\frac{1}{2}*(1-(\frac{2}{3})^2-(\frac{1}{3})^2)*2$=0.44444

   根据字典序以X作为节点;

   然后仅剩下Y作为其余结点;

   决策树如图:

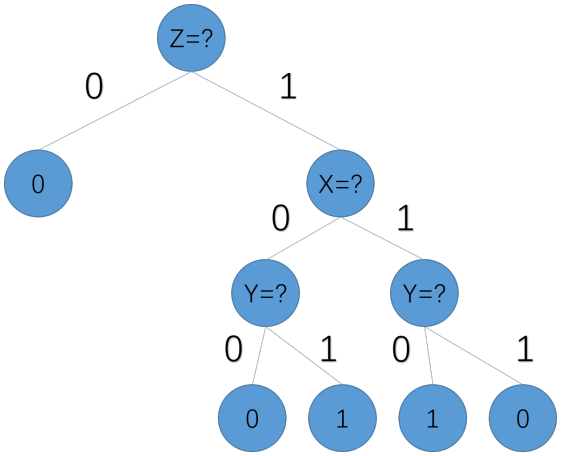


图 2: (Desicion tree of Pro 1.2)

题目 (ML problem 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?

解答：

1. $-(p_1 log_2\, p_1 + p_0 log_2\, p_0) = -(0.6 * log_2 * 0.6 + 0.4 * log_2 * 0.4) = 0.9709505944546686$
2. 我使用C程序计算了每一个以$x_1 \geq x_{1i}$为划分,以及$x_2 \geq x_{2i}$为划分的信息增益(Gain);
   因此我选择$x_2 \geq 38$作为划分;

```
Max Gain :53 0.144484

0:x1[i]: 24 0.00740339
1:x1[i]: 53 0.144484
2:x1[i]: 23 0.0789821
3:x1[i]: 25 0.00580215
4:x1[i]: 32 0.0464393
5:x1[i]: 52 0.0912774
6:x1[i]: 22 0
7:x1[i]: 43 0.124511
8:x1[i]: 52 0.0912774
9:x1[i]: 48 0.0199731
```

图 3: (第一层信息增益($x_1 \geq x_{1i}$))

因为$x_2 < 38$划分出来的两个元素label都是0,因此不需要继续划分;

现在考虑$x_2 \geq 38$划分出来的8个元素,同理:

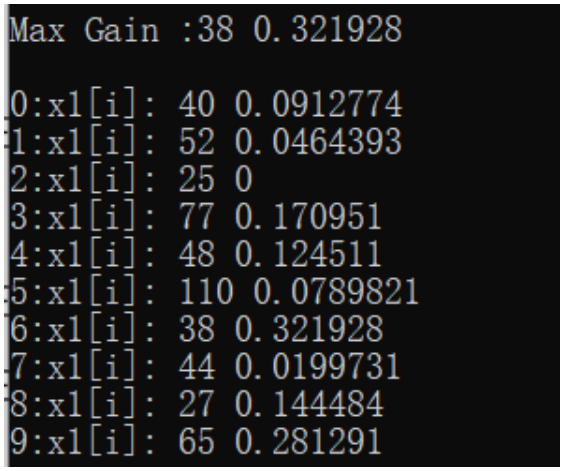


图 4: (第一层信息增益($x_2 \geq x_{2i}$))

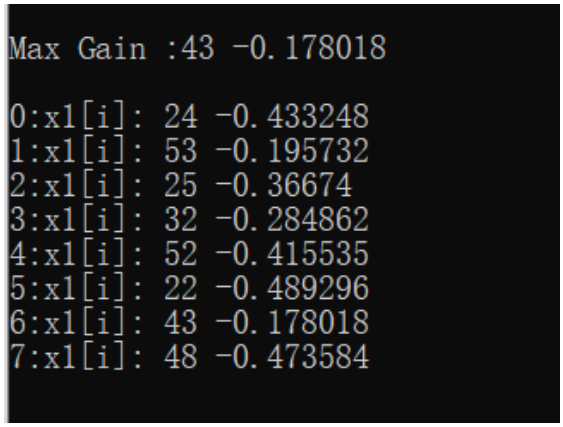


图 5: (第二层信息增益($x_1 \geq x_{1i}$))

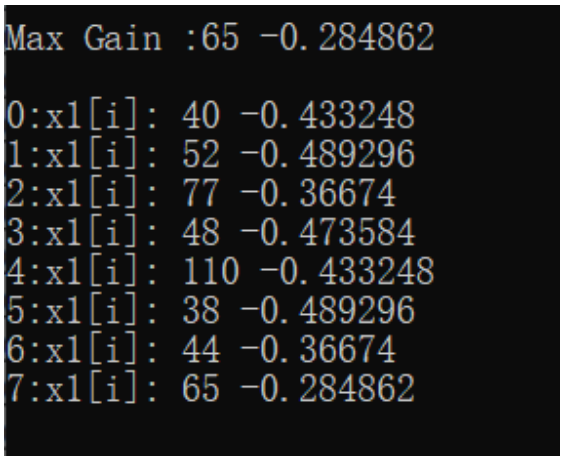


图 6: (第二层信息增益($x_2 \geq x_{2i}$))

因此选择$x_1 \geq 43$作为划分;

此时,$x_1 < 43$的集合都是label为1的,不用继续划分;

仅仅需要划分$x_1 \geq 43$的部分:　此时根据上图划分$x_2 \geq 65$,此时分出来的两类

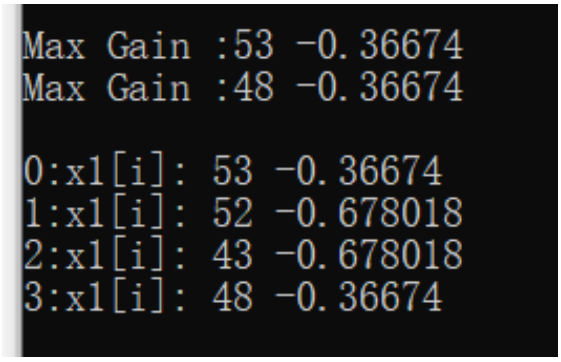


图 7: (第三层信息增益($x_1 \geq x_{1i}$))

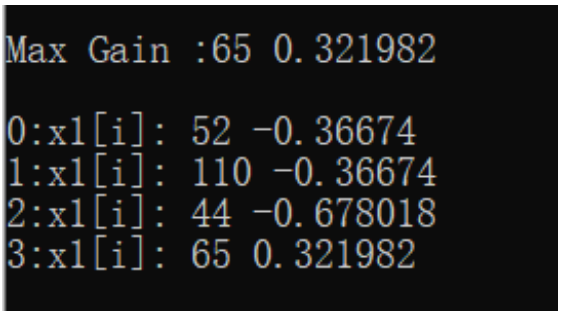


图 8: (第三层信息增益($x_2 \geq x_{2i}$))

的label内部都一样,不必继续划分;

(Hint: C++代码见附件PRO2_1.cpp,因为只是计算用途,所以基本没有可读性)

作决策树如下:

3. 我调用了python的最小二乘法进行了计算,代码如图:

   结果是y=0.68196814-0.02264422*x_1+0.01454232*x_2;我调用了sklearn库的LineadRegression也得到了一样的结果;

   代入进行预测,如果结果大于0.53label为1,小于0.5label为0,表格如下:

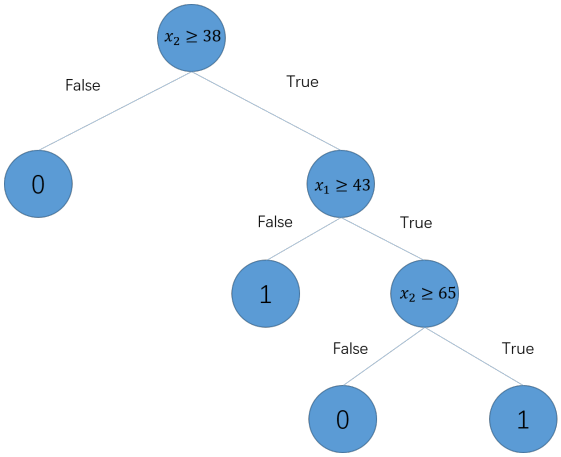


图 9: (Desicion tree of Pro 2.2)

```
from numpy.linalg import inv
from numpy import dot, transpose
from numpy.linalg import lstsq

from sklearn.linear_model import LinearRegression
X = [[1, 24, 40], [1, 53, 52], [1, 23, 25], [1, 25, 77], [1, 32, 48], [1, 52, 110], [1, 22, 38], [1, 43, 44], [1, 52, 27], [1, 48, 65]]
y = [[1], [0], [0], [1], [1], [1], [1], [0], [0], [1]]

print(lstsq(X, y)[0])
print(
    dot(inv(dot(transpose(X), X)), dot(transpose(X), y)))
```

图 10: (Code)

x_1	x_2	label	0.68196814-0.02264422*x_1+0.01454232*x_2
24	40	1	0.7201
53	52	0	0.2380
23	25	0	0.5247
25	77	1	1.2356
32	48	1	0.6553
52	110	1	1.1041
22	38	1	0.7364
43	44	0	0.3481
52	27	0	-0.1028
48	65	1	0.5402

决策树如图:

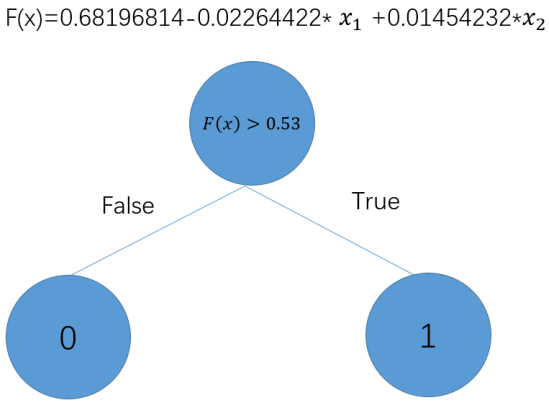
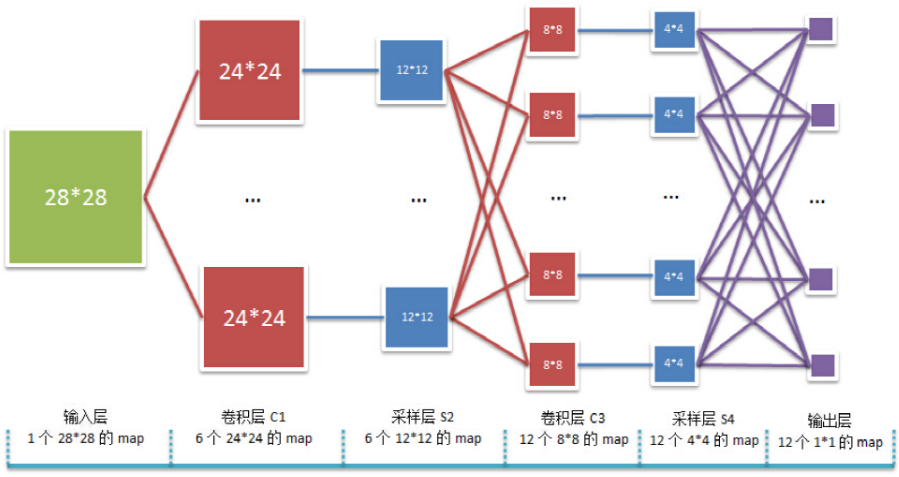


图 11: (Multivariate decision tree)

题目 (ML problem 3)

[25pts] Convolutional Neural Networks



Using Fig. CNN 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;

解答：

1. (a) 首先这是一个没有对原图像进行描边处理的卷积操作由28 × 28的输入变成了6 × 24 × 24(也就是深度为6)的卷积层C_1；

(b) 然后应该是经过了池化层进行了采样,从图片来看,kernel_size应该是2,因此输入的是$6 \times 24 \times 24$输出的是$6 \times 12 \times 12$的S_2

(c) 然后再一次对S_2进行了卷积操作,输入是6×12×12的S_2,输出是12×8×8的C_3,由原先的6层深度变成了现在的12层深度

(d) 相似地,进行了池化层采样,kernel_size应该也是2,变成了$12 \times 4 \times 4$;

(e) 最后应该是经过了全连接层得到了最后的$12 \times 1 \times 1$的输出

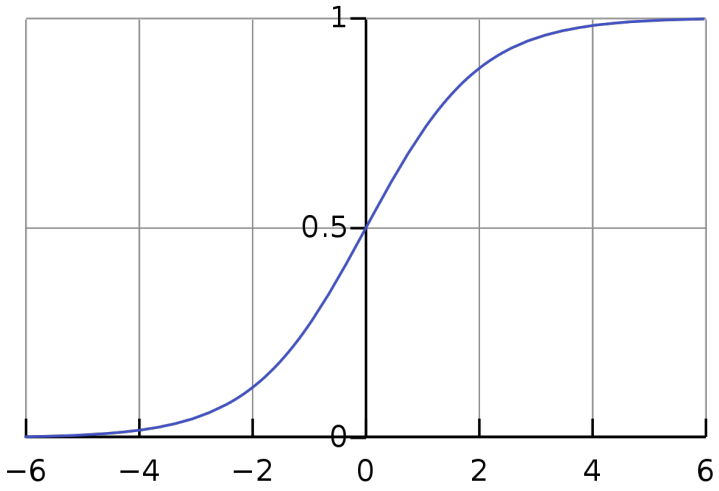


图 12: Sigmod

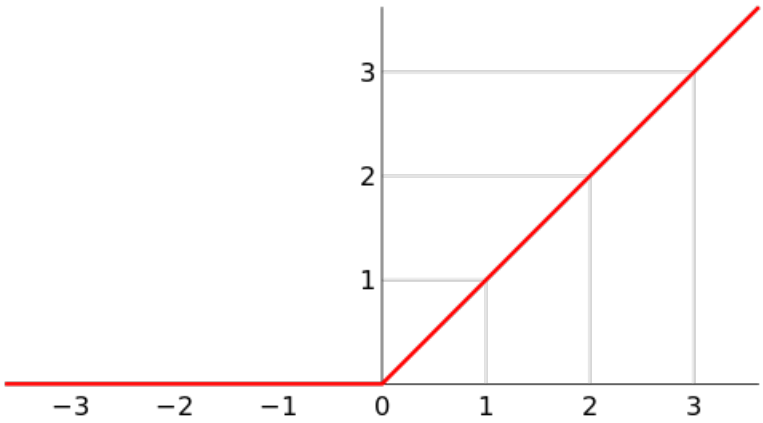


图 13: Relu

2. (a) Sigmod函数和Relu函数都是激活函数;下面是不同点:

(b) Sigmod函数历史比较悠久,,把数压缩到0-1的连续函数之中，但是也有一些缺点,简而言之是Sigmod函数容易饱和,当输入非常大或者非常小的时候，神经元的梯度就接近于0了,并且输出不是0均值的,并且Sigmod函数的计算量会比Relu函数大一点;

(c) Relu函数是一个最近比较受欢迎的激活函数,Krizhevsky et al. 发现使用 ReLU

得到的SGD的收敛速度会比 sigmoid/tanh 快很多(如上图右)。有人说这是因为它是linear，而且梯度不会饱和 ;并且只要计算一个阈值,计算量小;

但是Relu函数也有缺点,它在¡0的时候都是0,容易神经元“坏死”,因为负梯度之后它永远都是0了;

3. 假设有损失函数$L : R^T \rightarrow R$,以及y=Wx+b; 根据链式法则:$\frac{\partial L}{\partial w} = DL(y(W)) * Dy(W)$ 由$y_1 = \sum_j = 1^N W_{1j}x_j + b1$

   最后可以算到$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial y} * x$

4. 池化层:

$$\begin{aligned}\frac{\partial J(W,b;x,y)}{\partial b^{(l+1)}} &= \sum_{i=0}^{\frac{m-n+1}{r}-1}\sum_{j=0}^{\frac{m-n+1}{r}-1}\frac{\partial J(W,b;x,y)}{\partial z_{i,j}^{(l+1)}}\frac{\partial z_{i,j}^{l+1}}{\partial b^{(l+1)}} \\ &= \sum_{i=0}^{\frac{m-n+1}{r}-1}\sum_{j=0}^{\frac{m-n+1}{r}-1}\delta_{i,j}^{l+1}\end{aligned} \tag{1}$$

$$\begin{aligned}\frac{\partial J(W,b;x,y)}{\partial \beta^{(l+1)}} &= \sum_{i=0}^{\frac{m-n+1}{r}-1}\sum_{j=0}^{\frac{m-n+1}{r}-1}\frac{\partial J(W,b;x,y)}{\partial z_{i,j}^{(l+1)}}\frac{\partial z_{i,j}^{l+1}}{\partial \beta^{(l+1)}} \\ &= \sum_{i=0}^{\frac{m-n+1}{r}-1}\sum_{j=0}^{\frac{m-n+1}{r}-1}\delta_{i,j}^{l+1}\sum_{u=ir}^{(i+1)r-1}\sum_{v=jr}^{(j+1)r-1}a_{u,v}^{(l)}\end{aligned} \tag{2}$$

5. 卷积层:

$$\begin{aligned}\frac{\partial J(W,b;x,y)}{\partial b^{(l)}} &= \sum_{u=0}^{m-n}\sum_{v=0}^{m-n}\frac{\partial J(W,b;x,y)}{\partial z_{u,v}^{(l)}}\frac{z_{u,v}^{(l)}}{\partial b^{(l)}} \\ &= \sum_{u=0}^{m-n}\sum_{v=0}^{m-n}\delta_{u,v}^{(l)}\end{aligned} \tag{3}$$

$$\begin{aligned}\frac{\partial J(W,b;x,y)}{\partial k_{rot_{g,h}}^{(l)}} &= \sum_{u=0}^{m-n}\sum_{v=0}^{m-n}\frac{\partial J(W,b;x,y)}{\partial z_{u,v}^{(l)}}\frac{\partial z_{u,v}^{(l)}}{\partial_{g,h}^{(l)}} \\ &= \sum_{u=0}^{m-n}\sum_{v=0}^{m-n}x_{rot_{u+g,v+h}}^{l-1} * \delta_{u,v}^{(l)}\end{aligned} \tag{4}$$

注： 第二题参考了:*https* : *//blog.csdn.net/yangdashi*888/*article/details/*78015448
第三题参考了:*https* : *//zhuanlan.zhihu.com/p/*32836341

题目 (ML problem 4)

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

https : *//pytorch.org/tutorials/beginner/blitz/cifar*10$_t$*utorial.htmlhere*. 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.

解答：

1. 我主要参考了Pytorch以及莫烦Python教程完成了我的代码,我的代码每50次就会输出一下目前的train_loss,validation_loss,最后会绘制四张图片:Train loss and validation loss,Epoch and validation loss,Epoch and accuracy,Training count and accuracy;然后输出最后的accuracy以及其他一些参数;

2. 一开始我参考了莫烦python构建了卷积−− >relu激活−− >池化−− >卷积−− >relu激活−− >池化−− >输出的网络, 在输入EPOCH=15,LR=0.0005,使用Adam方法学

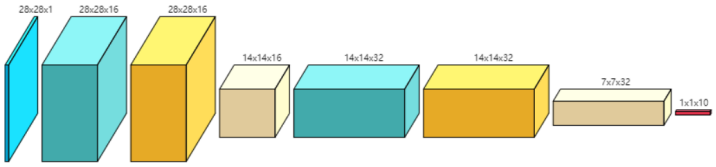


图 14: Net1

习的情况下:结果如下面的图片:

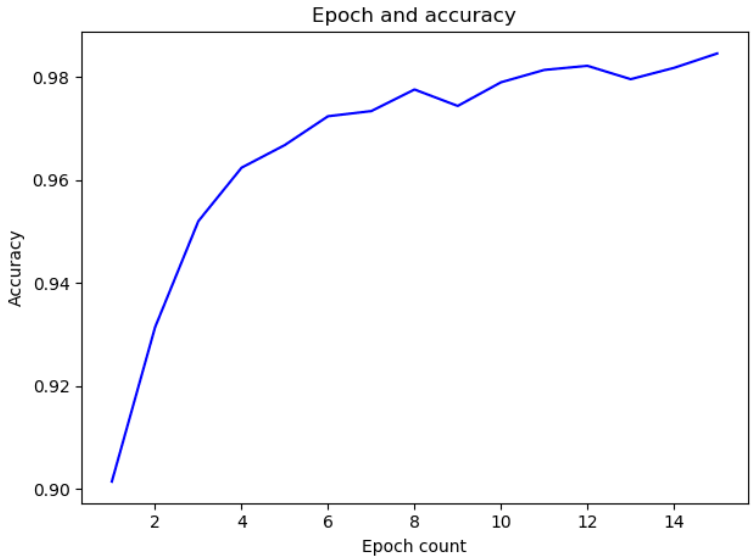


图 15: Epoch and accuracy

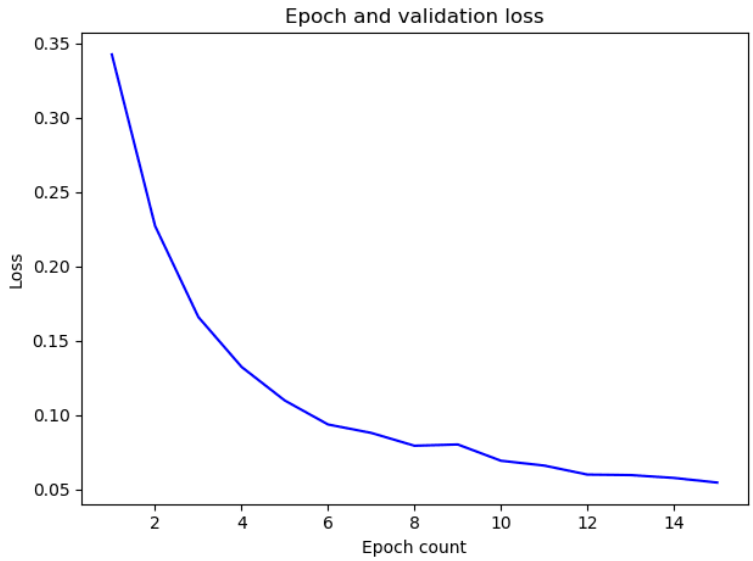


图 16: Epoch and validation loss

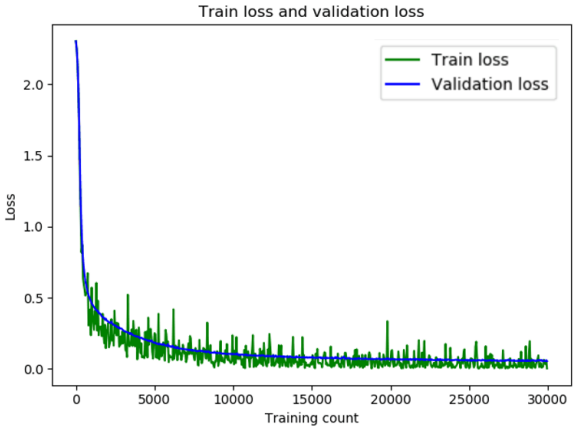


图 17: Train loss and validation loss

然后我结合官网的教程做了一些改进,最后加了三层全连接并且倒数两个用Relu函数做了激活: 此时网络是:卷积−− >relu激活−− >池化−− >卷积−− >relu激活−− >池化−− >全连接−− >relu激活−− >全连接−− >relu激活−− >输出的网络

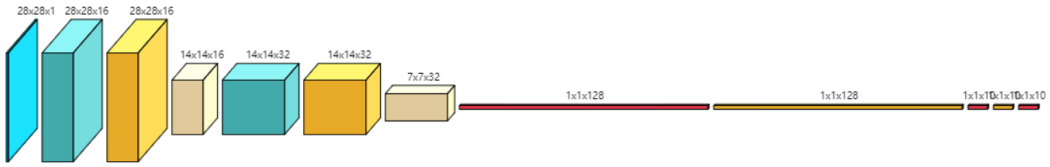


图 18: Net2

结果如下面的图片:

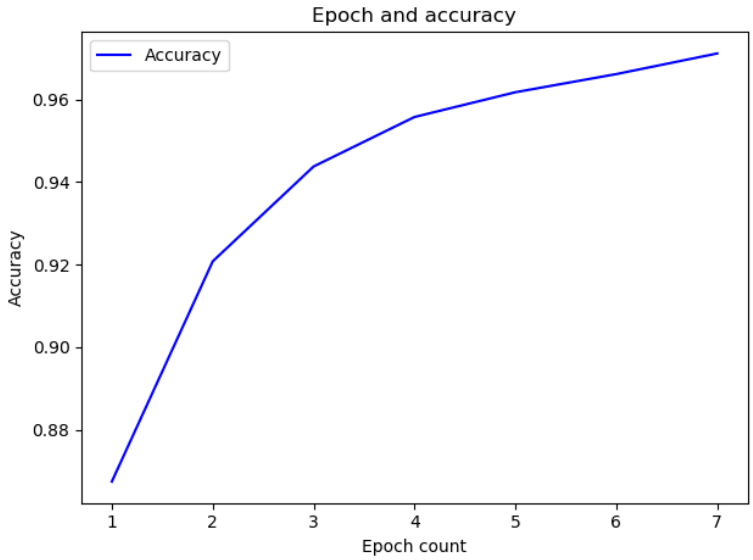


图 19: Epoch and accuracy

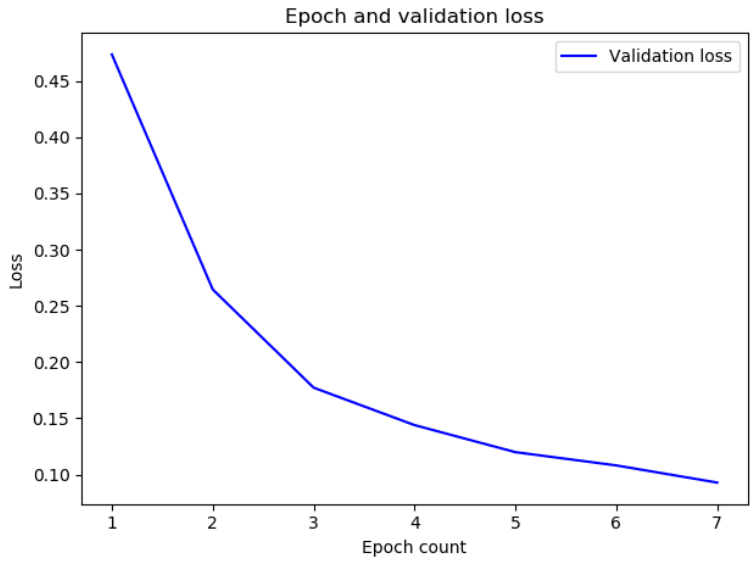


图 20: Epoch and validation loss

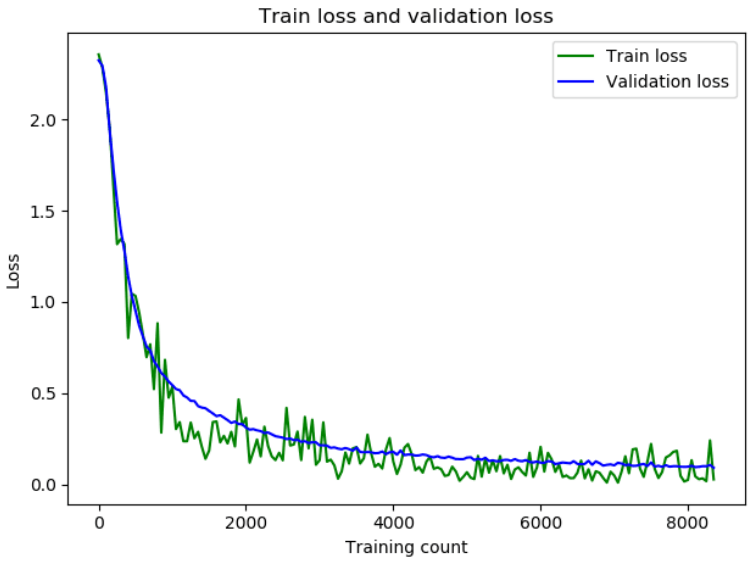


图 21: Train loss and validation loss

3. 我也探索了SGD,相对于Adam方法,SGD要求的学习率要更高一点,我在下图中是设置的0.01

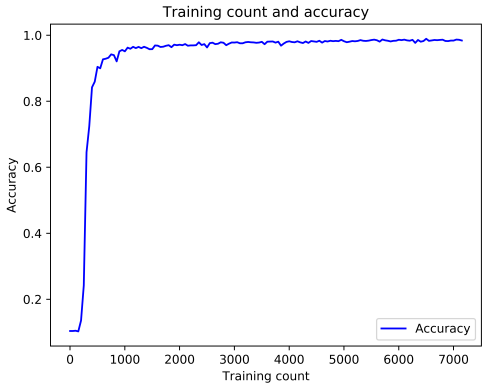


图 22: Training count and accuracy of SGD

对于adam方法,epoh与accuracy的关系我在第二问的图片中有放出,下面的图片是学习率与accuracy的关系,在这些数据中,我控制了Epoch为6:

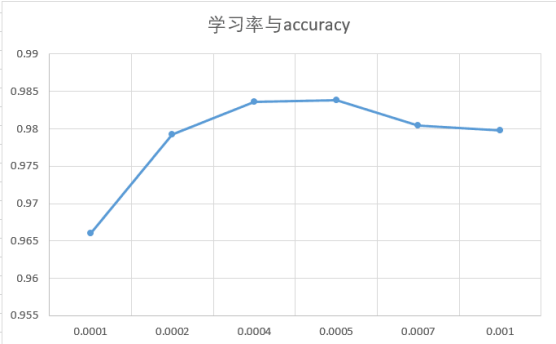


图 23: 学习率与accuracy的关系

4. 这一问的图片上面几问都有相应的放出,总结下来一般情况下train_loss的抖动会更大一点,并且如果过拟合比较严重的话,validation_loss会比train_loss高很多;