计算机科学与技术学院神经网络与深度学习课程实验报告

实验题目：神经网络实现		学号：201600301148
日期：3.18	班级：人工智能	姓名：周阳
Email：862077860@qq.com		

实验目的：

1. 实现三层 mlp 的前项运算，反向传播算法，并调参得到最佳模型。
2. 实现 softmax 线性分类器，并进行测试（cifar-10 数据集）
3. 可视化 训练参数

实验软件和硬件环境：

CPU：英特尔至强 E5

GPU：NVIDIA GeForce 1060 6G

内存：16G

Pycharm

Python 3.6

实验原理和方法：

1，softmax

①score 的计算

$$y = Wx$$

②softmax 概率（prob）

$$S_i = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

为了防止溢出使用以下形式计算 softmax（不影响求导）

$$\frac{e^{(x_i - M)}}{\sum_j^n e^{(x_j - M)}}$$

③cross entropy（loss 的定义式）

$$-\sum_i \text{label}_i \log S_i$$

④cross entropy（loss 的计算式）

$$loss = -\log S_i = -y_i + \log \sum_j e^{\hat{y}_j}$$

由于 label 是 one-hot 的形式。

⑤对 W 进行求导的结果

$$\begin{cases} dW = (-1 + \dfrac{e^{\hat{y}_j}}{\sum_j e^{\hat{y}_j}})x_i & j = i \\ dW = \dfrac{e^{\hat{y}_j}}{\sum_i e^{\hat{y}_j}} x_i & j \neq i \end{cases}$$

2，三层神经网络

①前项计算

如下图所示进行 forward compute

$$h^1 = W^1 x + b^1$$

$$a^1 = relu\left(h^1\right)$$

$$h^2 = W^2 a^1 + b^2$$

$$a^2 = relu\left(h^2\right)$$

$$h^3 = W^3 a^2 + b^3$$

②反向传播

定义中间结果

$$\delta^l = \frac{\partial loss}{\partial h^l}$$

（该层损失的定义式）

$$\delta^l = \left(\left(W^{l+1}\right)^T \delta^{l+1}\right) \odot relu'\left(h^l\right)$$

（该层损失的递推式）

根据中间结果损失，再得每一层 w 和 b 的梯度结果

$$\frac{\partial loss}{\partial W_{ij}^{l}} = \frac{\partial loss}{\partial h_i^{l}} \frac{\partial h_i^{l}}{\partial W_j^{l}} = \delta_i^{l} a_j^{l-1}$$

$$\frac{\partial loss}{\partial b_i^{l}} = \frac{\partial loss}{\partial h_i^{l}} \frac{\partial h_i^{l}}{\partial b_i^{l}} = \delta_i^{l}$$

实验步骤：（不要求罗列完整源代码）

①使用循环计算 softmax 的前项运算与 BP 结果对比不使用循环 BP 结果

使用循环：

```
batch_size = X.shape[0] ##样本数目
C = W.shape[1] ##类别数目

score = X.dot(W)## 得分函数值
score = score.T

for i in range(batch_size):
  f = score[:, i] ## 得分函数值一行
  f -= np.max(f) ## 归一化到0以下，这段的ex平稳,这种技术可以使数据溢出问
  prob = np.exp(f) / np.sum(np.exp(f), axis=0) ## 通过softmax 将得

  loss += - f[y[i]] + np.log(np.sum(np.exp(f), axis=0)) ## 交叉熵l
  for j in range(C):
    dW[:,j] += prob[j] * X[i]
  dW[:,y[i]] -= X[i]

  # dW的求解： 使用链式法则
  # dW   = (p[j] - 1) * X[:, i]     , if j == i
  # dW   =  p[j]      * X[:, j]     , otherwise

dW /= batch_size
loss /= batch_size

# 正则化
loss += reg * np.sum(W ** 2)/2.0 ## 正则化项  使用l2正则化
dW += reg * W
```

直接使用矩阵运算

```
batch_size = X.shape[0]  ##样本数目


f = X.dot(W).T

f -= np.amax(f, axis=0)

prob = np.exp(f) / np.sum(np.exp(f), axis=0)
prob[y, range(batch_size)] -= 1

loss = np.sum(- f[y, range(batch_size)] + np.log(np.sum(np.exp(f), axis=0))) / batch_size

dW = prob.dot(X) / batch_size

# 正则化
loss += 0.5 * reg * np.sum(W ** 2)
dW += reg * W.T

dW = dW.T
```

二者结果相同，效率差 38 倍！（在我的硬件条件下）

```
naive loss: 2.440237e+00 computed in 0.151816s
vectorized loss: 2.440237e+00 computed in 0.003907s
```

②三层神经网络前项运算

```
##  前向传播  ##

score1 = np.dot(X, W1) + b1
a1 = np.maximum(0, score1) ## set relu as activate function
score2 = np.dot(a1, W2) + b2
a2 = np.maximum(0, score2) ## set relu as activate function
scores = np.dot(a2, W3) + b3
```

③三层神经网络 BP 反向传播

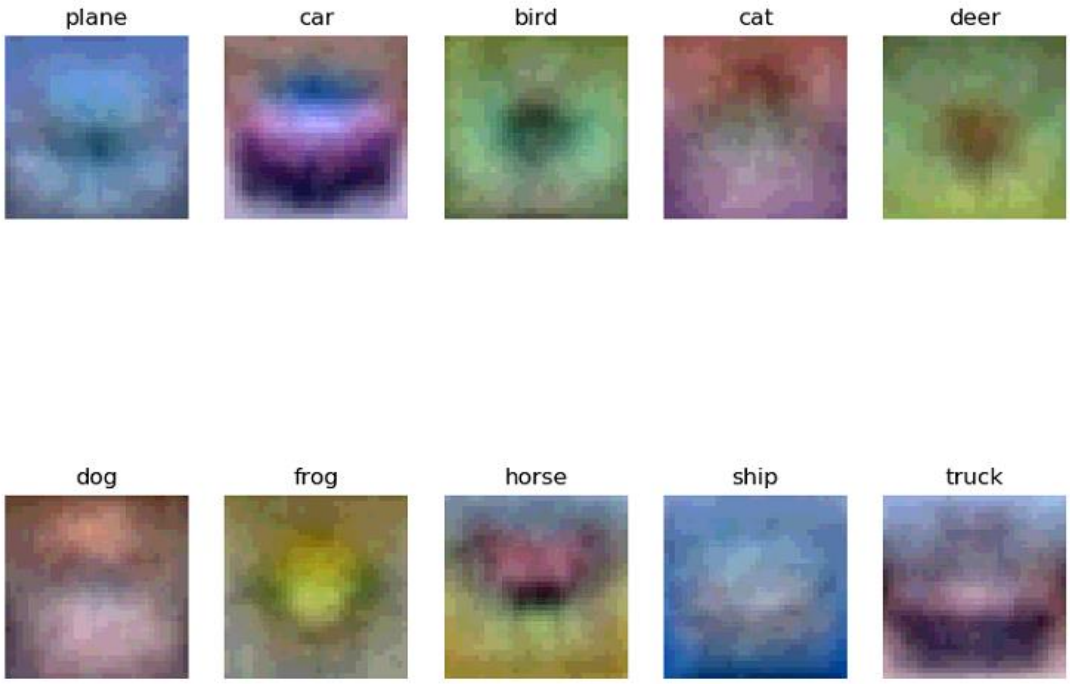
```
dscore = np.exp(f) / np.exp(f).sum(axis = 1, keepdims = True)
dscore[range(N), y] -= 1
dscore /= N
## 类似softmax的反向传播

grads['W3'] = np.dot(a2.T, dscore) + reg * W3
grads['b3'] = np.sum(dscore, axis = 0)

dhidden2 = np.dot(dscore, W3.T)
dhidden2[a2 <= 0.00001] = 0

grads['W2'] = np.dot(a1.T, dhidden2) + reg * W2
grads['b2'] = np.sum(dhidden2, axis = 0)

dhidden1 = np.dot(dhidden2, W2.T)
dhidden1[a2 <= 0.00001] = 0

grads['W1'] = np.dot(X.T, dhidden1) + reg * W1
grads['b1'] = np.sum(dhidden1, axis = 0)
```

④结果与可视化

Softmax 准确率：

```
best validation accuracy achieved during cross-validation: 0.362000
softmax on raw pixels final test set accuracy: 0.358000
```

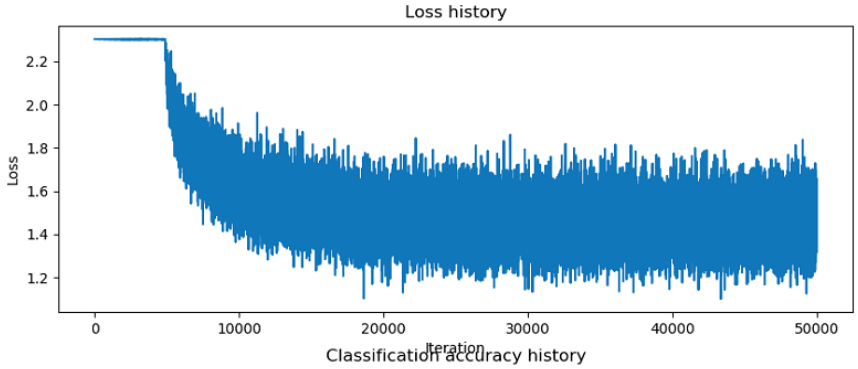
Softmax 可视化：

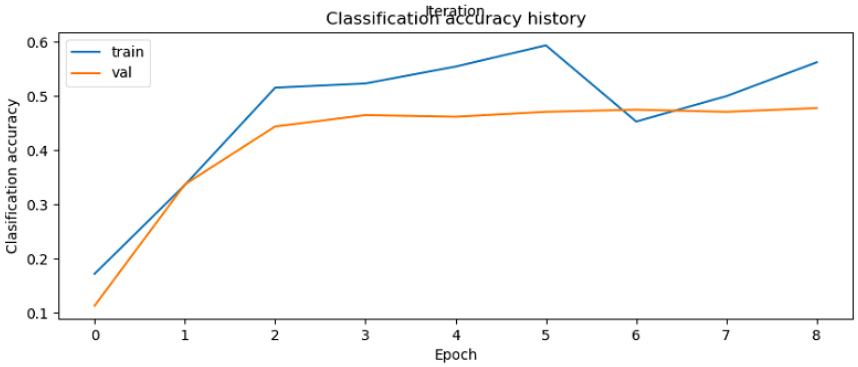


神经网络准确率：

使用超参数：

```
hidden_size = 50
num_classes = 10
batch_size = 128
lr = 5e-3
reg = 0.01
num_batch = 50000
learning_rate_decay = 0.99
```

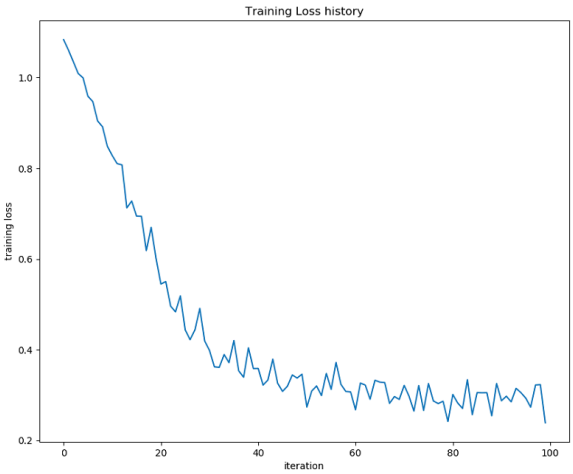




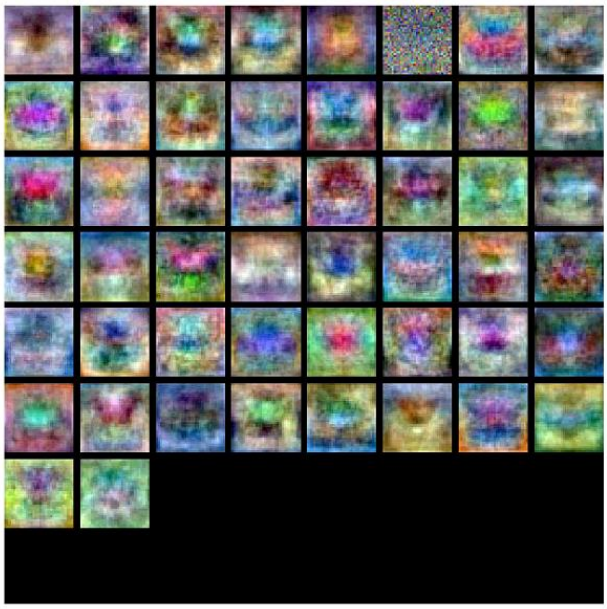
最终 Valid 准确率：0.481

神经网络可视化：

Toy model：



参数矩阵可视化：



结论分析与体会：

本次试验旨在让我们对，模型训练前向计算，Loss 计算，反向传播有所概念，最终得到如此结果。

本次实验对于我而言最大的难点可能在于理论知识的不完备，对于 softmax 的求导，和各种知识都是经过了大量的查阅之后才有所了解的。

就实验过程中遇到和出现的问题，你是如何解决和处理的，自拟 1－3 道问答题：

1，那部分内容最困难？

答：对于本次试验，涉及大量的推导和大量的矩阵操作，推导公式与课上学习的内容有些出入，需要我一点一点地推导完毕，这部分对于我来说还是十分困难的。

2，什么地方认为非常重要？

答：本次实验让我对调参，参数初始化有了更深的理解，在超参数的认知和选择上都有了新的见解。