计算机视觉　softmax实验

小组成员：潘子晴、王永炜、彭宇

内容简介

1. Softmax Classifier 梯度推导和检验

2. batch size、learning rate、epoch 参数分析

3. L1、L2正则化方法使用

4. 数据预处理 和 实验结果

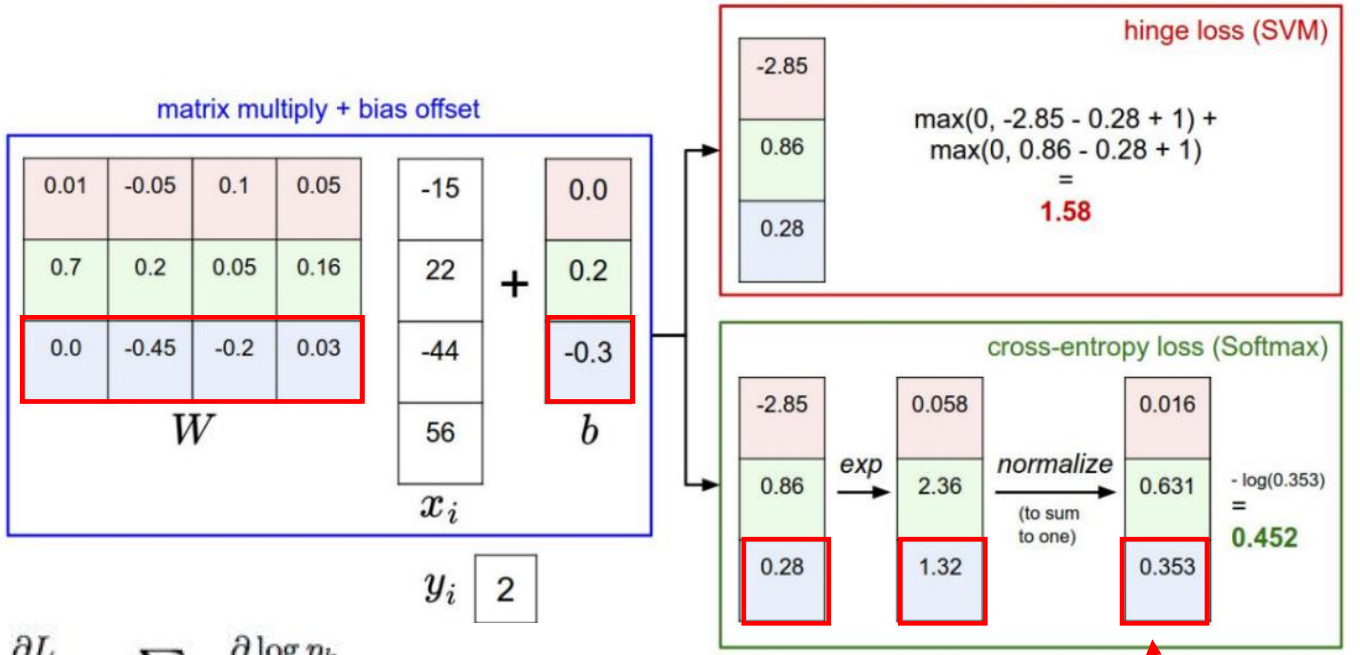
1. Softmax Classifier 梯度推导和检验

Analytic gradient

根据损失函数，对 W 矩阵 中第 yi 行和

除 yi 行外的其他行 分别求导

```
def eval_analytic_grad(W, xi, yi):
    scores = xi.dot(W.T)
    exp_scores = np.exp(scores).reshape(W.shape[0],)
    pro_scores = exp_scores / np.sum(exp_scores)
    grad = np.zeros(W.shape)
    for i in range(W.shape[0]):
        if(i==yi):
            grad[i] = (pro_scores[i] - 1) * xi
        else:
            grad[i] = pro_scores[i] * xi
    return grad
```



$$\frac{\partial L}{\partial o_i} = -\sum_k y_k \frac{\partial \log p_k}{\partial o_i}$$

$$= -\sum_k y_k \frac{1}{p_k} \frac{\partial p_k}{\partial o_i}$$

$$= -y_i(1-p_i) - \sum_{k \neq i} y_k \frac{1}{p_k}(-p_k p_i)$$

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

$$= -y_i + y_i p_i + \sum_{k \neq i} y_k (p_i)$$

$$= p_i \left(\sum_k y_k \right) - y_i$$

$$= p_i - y_i$$

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{sj}}\right)$$

Numerical gradient

```
def eval_numerical_grad(W, xi, yi):
    h = 1e-5
    grad = np.zeros(W.shape)
    for i in range(W.shape[0]):
        for j in range(W.shape[1]):
            fx = loss(W, xi, yi)
            W[i][j] =  W[i][j] + h
            fx_h = loss(W, xi, yi)
            grad[i][j] = (fx_h - fx) / h
    return grad
```

结果比较：

```
Analytic gradient:
[0.03771116 0.05656673 0.01885558 0.09427789]
[-0.15482867 -0.232243   -0.07741433 -0.38707166]
[0.11711751 0.17567627 0.05855876 0.29279378]

Numerical gradient:
[0.03771119 0.05656689 0.01885566 0.09427854]
[-0.15482872 -0.23224296 -0.07741428 -0.38707116]
[0.11711702 0.17567572 0.05855862 0.29279347]
```

相似度：矩阵对应值相减的绝对值之和为

3.053808551485570 6e-06

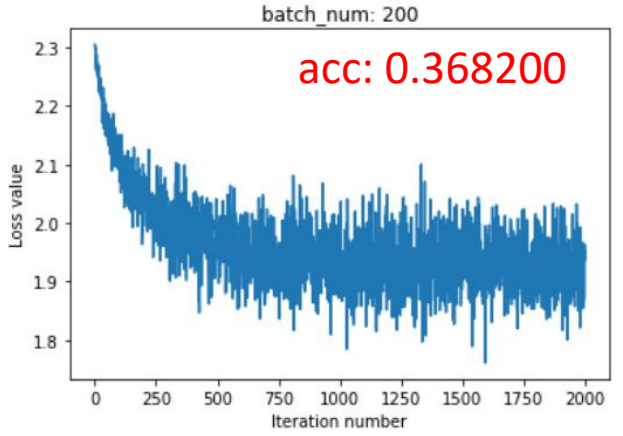
2. batch size、learning rate、epoch 参数分析

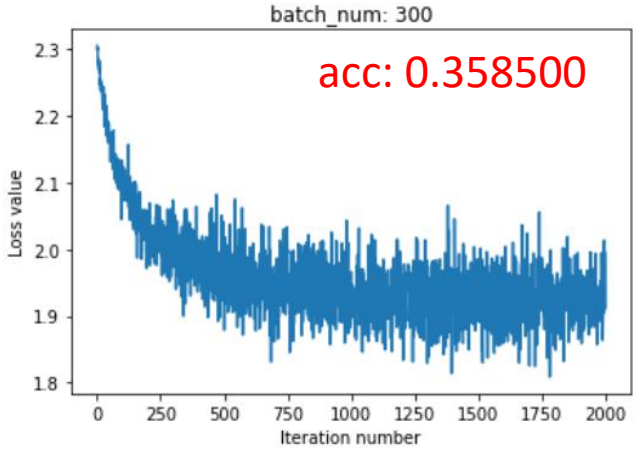
Batch size

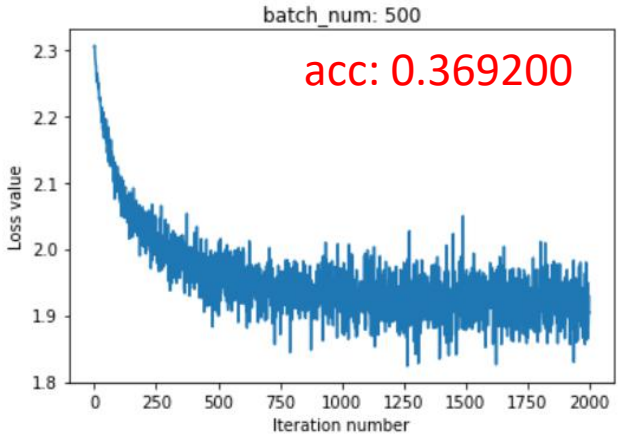
（1）batch_num为每次从总训练集中随机抽取的样本数量，将这部分数据计算得到的梯度 作为 W的更新方向；

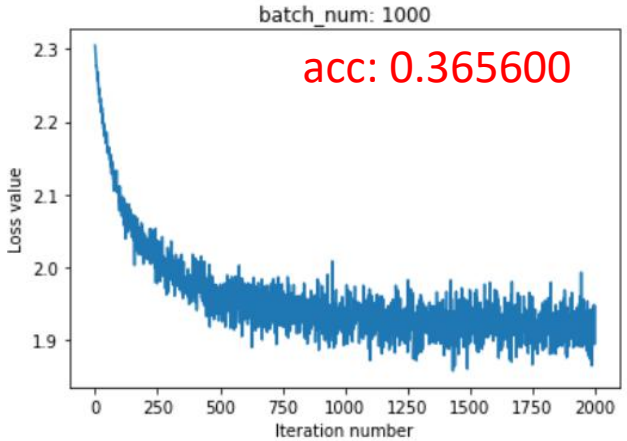
（2）随着batch num 的增大，迭代计算出的梯度值更加稳定，曲线波动减小；但计算量也增大；

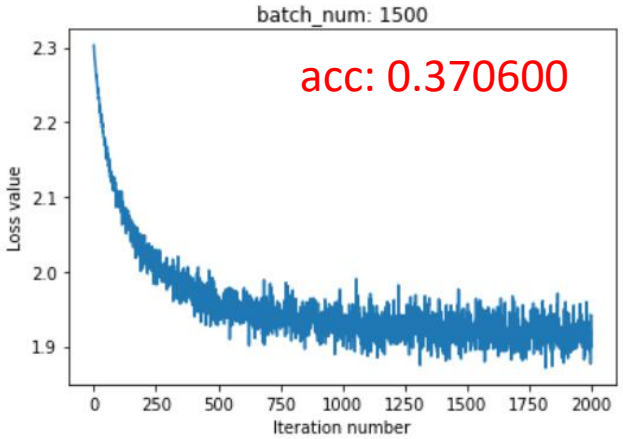
（3）batch_num>=500 时准确率差别不大，说明随机抽取的一定数量的样本数据大致能代表整个训练集的梯度方向。从而我们能靠此方法加快训练速度。









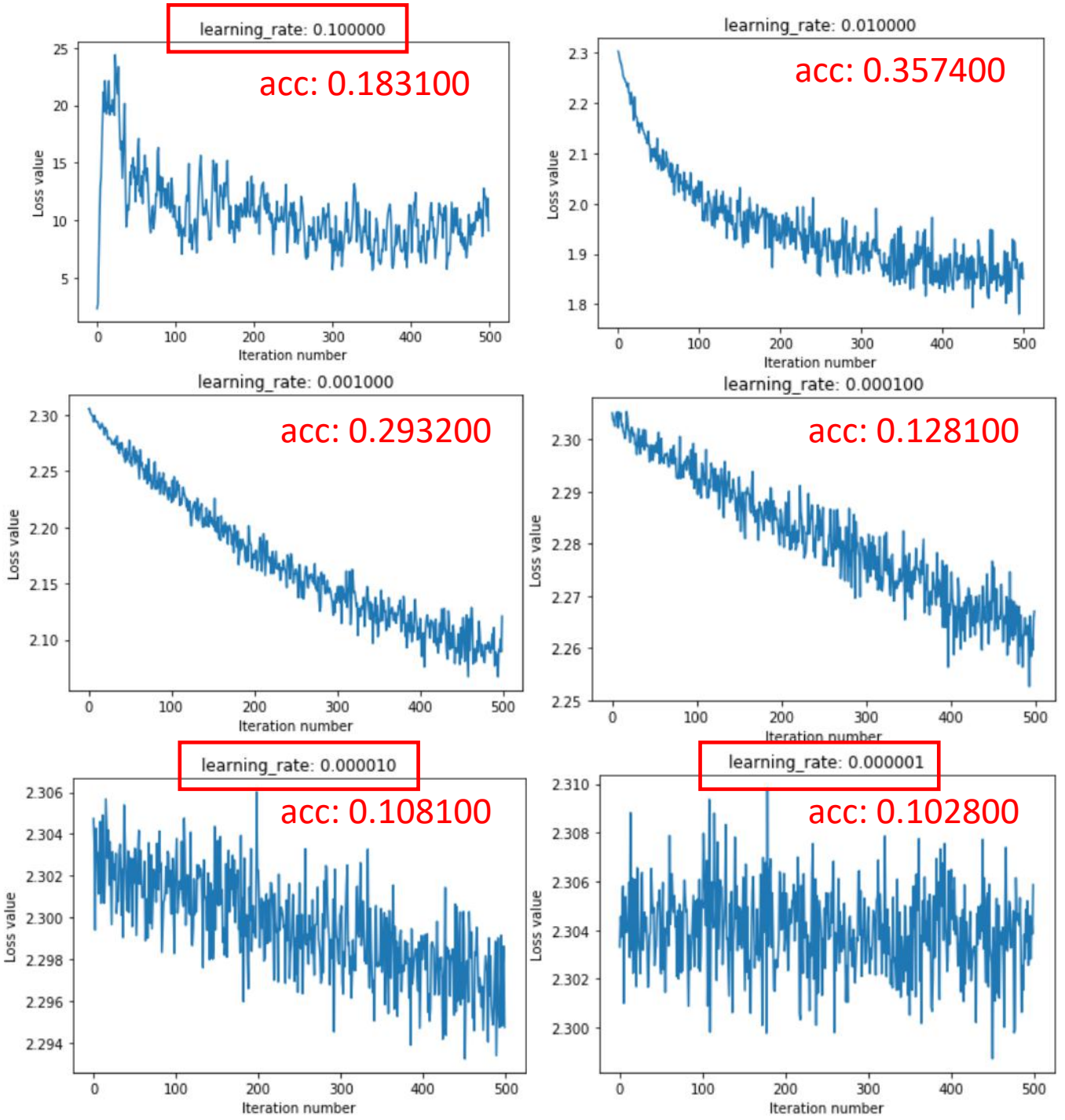


Learning rate

（1）learning rate 决定了每次在梯度方向移动的步长；设置较大的学习率可以加快收敛速度，但最终可能到达不了最小的损失点，甚至有可能模型不收敛；

（2）学习率为 0.1时，迭代一定次数后损失值不再下降，模型不收敛；学习率小于 0.0001时因收敛速度太慢，没有明显效果；

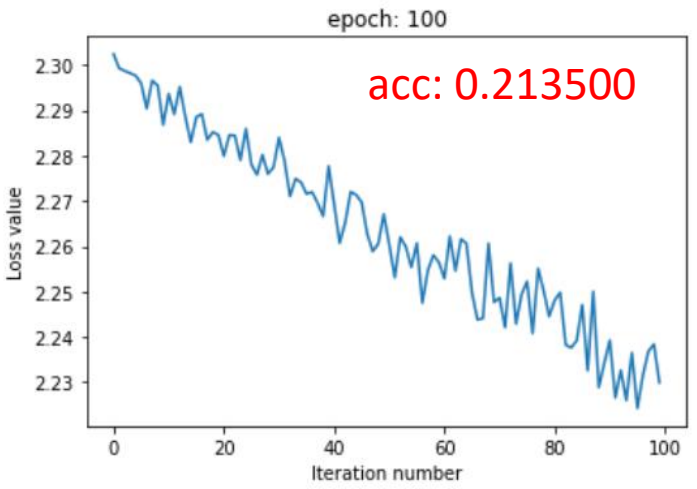
（3）动态更新 learning rate 值：如每迭代一定次数将learning rate值减半。

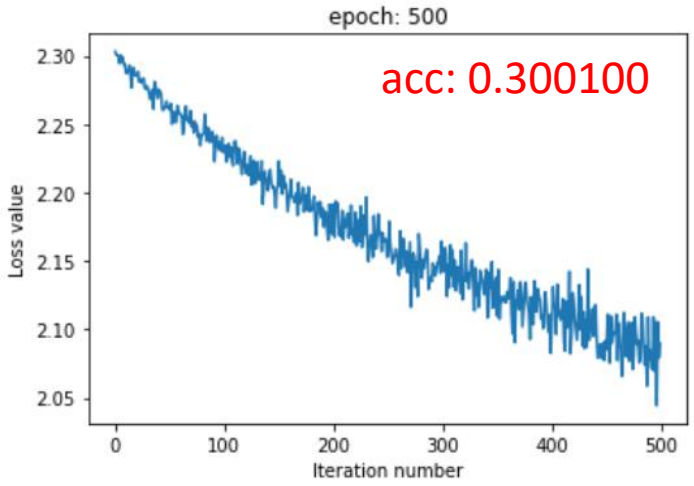


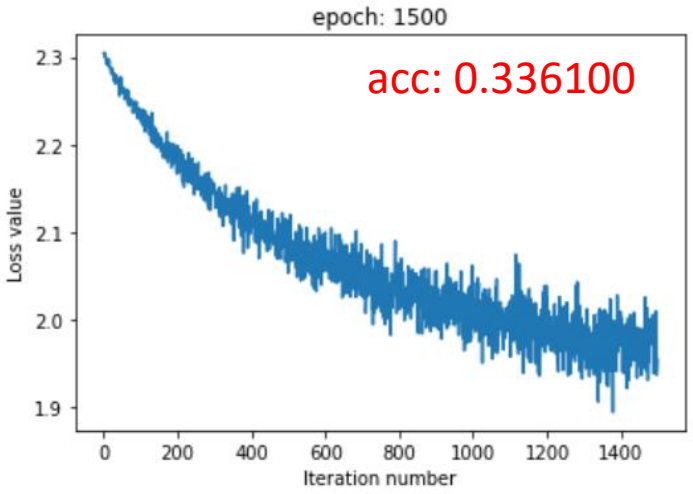
Epoch

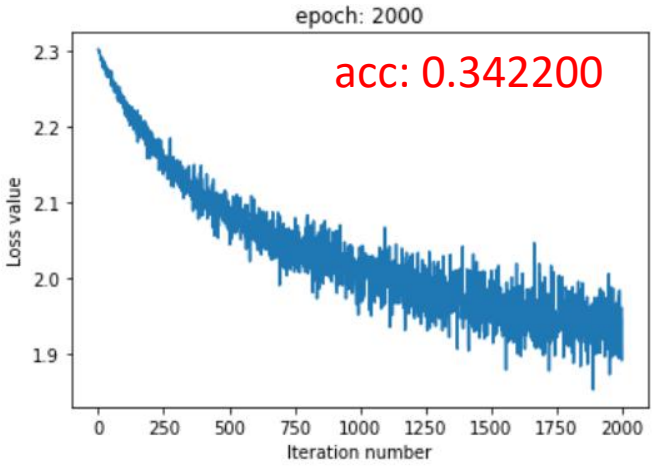
（1）epoch是更新 W 值的迭代次数，原则上迭代次数越多,训练效果越好

（2）迭代次数超过一定值后模型收敛，loss 趋于稳定







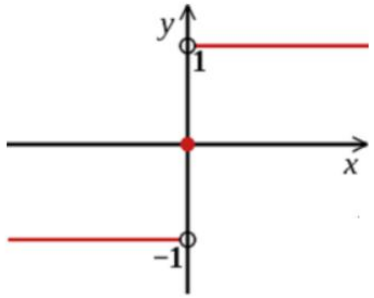


3. L1、L2正则化方法使用

L1正则化方法

（1）L1正则化公式 $$\tilde{J}(w) = J(w) + \lambda \|w\|_1$$

（2）求导公式 $$\nabla_w \tilde{J} = \nabla_w J + \lambda \text{sign}(w)$$



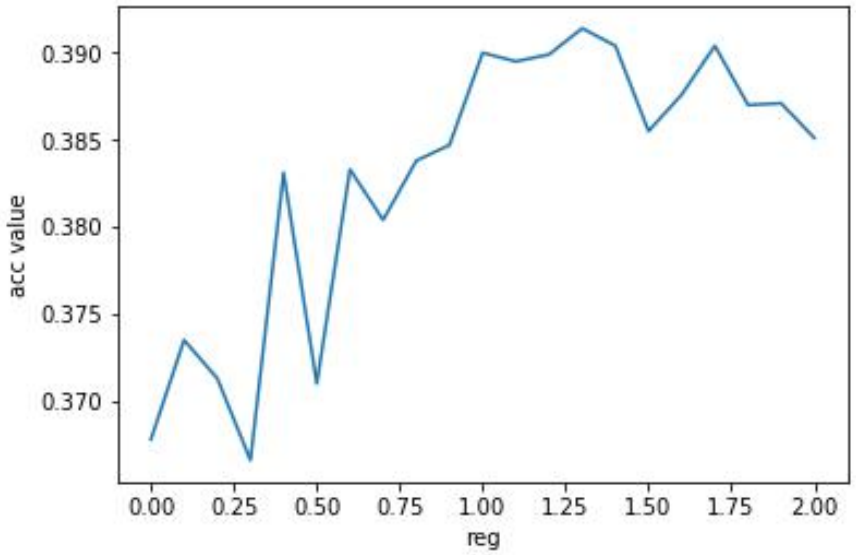
（3）问题：由于绝对值的导数sign(x)不会随着梯度的反方向而下降，实际测试中，直接使用加入L1正则化的导数公式会出现 loss值过大甚至负数的情况

解决方法：近端梯度下降法PGD（未实现）

L2正则化方法

（1）L2正则化公式 $\tilde{J}(w) = J(w) + \frac{\lambda}{2} w^T w$

（2）求导公式 $\nabla_w \tilde{J} = \nabla_w J + \lambda w$



（3）L2正则化的权重 λ 探究：

batch_size: 500

learning_rate: 0.01

epoch: 2000

权重 λ 为1.2 时效果最好

4. 数据预处理 和 实验结果

数据预处理与实验结果

（1）像素归一化到 0 ~ 1

```
def min_max_scaler(X):
    min1 = 0
    max1 = 255
    return (X-min1)/(max1-min1)
```

（2）提取 hog 特征

```
pixels_per_cell=(4, 4),cells_per_block=(2, 2)
```

（3）PCA 降维

保留 95% 信息量

实验结果：训练集50000，测试集10000，迭代50000次，最终测试集上准确率为0.441900

Thanks