PRML assignment2 report

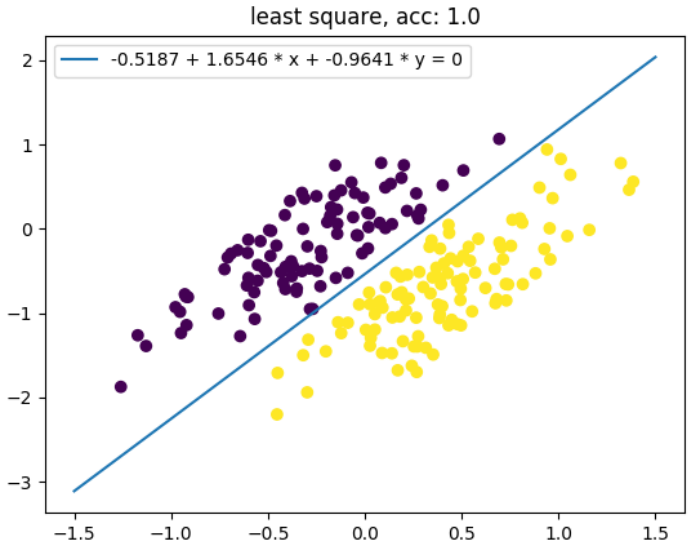
Part1

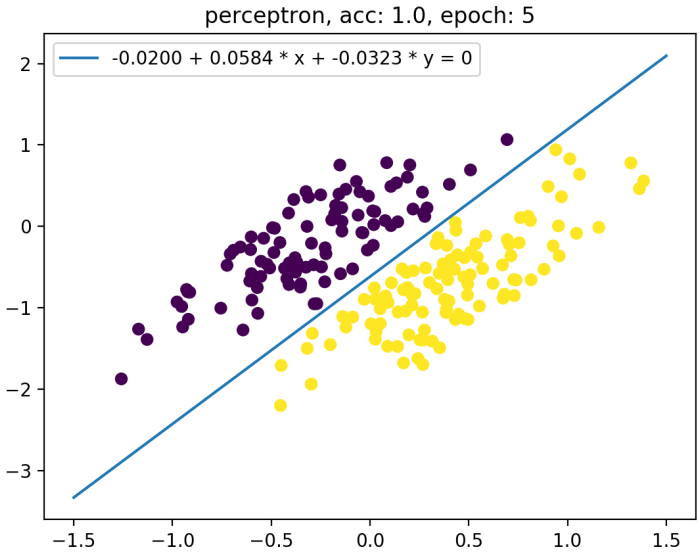
0. 代码组织

Part1 部分的代码位于 part1/main.py 中，有两个函数 least_square() 和 perceptron()，分别实现了 least square model 和 perceptron algorithm

运行代码就能得到 requirements 中的图像，并且计算得到算法的正确率

1. Requirements





如以上两图所示，正确率均为100%

Part2

0. 代码组织

与本部分的相关代码位于 part2 文件夹中，以下为各文件的代码简单介绍，代码的具体细节请参照代码注释。

- **data_utils.py** 实现了一个 data_processor 类，该类提供了两个函数。generate_vocabulary 将接收的 data set 进行处理，生成 vocabulary 并保存，并返回 vocabulary 的长度；process_data 函数将接受的 data set 以及 target set 根据 vocabulary 进行处理，分别返回 multi-hot 和 one-hot 结果
- **layer_utils.py** 中实现了 logistic model 中需要的两个 layer 分别是 fc layer 和 softmax layer。affine 被分为 forward 和 backward 两个函数，因为 softmax 生成了最终结果，所以 softmax_loss 函数同时完成了 forward 和 backward 两个部分的内容。
- **logistic.py** 实现了 logistic_model 类，并拥有一个成员函数 loss。如果 loss 函数被同时传入 data set 和 label，loss 函数会调用 softmax 函数，并得到最终的 loss 数值和参数的 grads；如果传入 data set，则 loss 函数会直接返回 fc 层输出的 scores 结果，用于预测类别。其中可以设定正则化项的系数以及初始化范围。
- **trainer.py** 实现了 trainer 类用于训练 model。其中可以设定学习率，学习率降低率，batch_size，epoch 数以及一些与输出相关的参数。调用 train 函数训练结束后，会返回训练完成的 model 和训练时的相关记录。
- **main.py** 调用以上实现的工具，初始化数据，生成 training set 和 validation set，初始化模型并进行训练，将中间过程可视化，并提供在 test 集上进行测试的实现。main.py 中的函数分别完成不同问题的探究内容，具体见代码。（本报告中大部分图都可以找到相应代码）
- **numerical_gradient.py** 为两个 layer 提供了数值求导的方式，验证其求导计算的正确性。

1. requirement1

multi-hot 实现方法

- 将训练集数据进行清洗。去除 string.puntuation 和替换 string. whitespace
- 统计词频并去除小于 min_count 的词，得到 vocabulary，并以字典形式进行组织和存储（key 为词，value 为 index）
- 接收到需要处理的 data set 时，将数据用同样的方式进行清洗。并扫描 data set 中的每条数据，根据 vocabulary 字典得到相应的 muilt-hot vector

one-hot 实现方法

接收得到 target 向量，并利用 numpy 提供的 list index 直接得到 ont-hot 向量组

```
one_hot = np.zeros((cl, num_classes))
one_hot[np.arange(cl),C] = 1
```

具体代码实现可在 data_utils.py 中找到。

2. requirement2

2.1求解偏导数

偏导数的求解因为篇幅较长，已写在文章最后的**附页**中。

2.2对于 logistic 回归，正则化项是否需要考虑 bias 项

考虑引入正则化项的原因。如果对模型的参数不引入任何的惩罚，即对模型空间（模型复杂度）没有任何限制，那么为了使得模型在训练样本上的拟合结果更优，模型就会选择较大的参数，使得拟合函数的导数值较大，从而局部的函数变化较大，能够兼顾更多训练样本中的细节。这样就会造成模型在训练样本上过拟合，模型泛化能力变差。

正则化的引入就是为了限制模型空间，从而增加模型泛化能力。**而 logistic 中的偏置 b 对模型的曲率没有任何贡献，仅影响模型的位置信息，因此不需要在正则化项中引入 b。**

在 The Elements of Statistical Learning 中也可以找到相应的解释：

Penalization of the intercept would make the procedure depend on the origin chosen for Y *; that is, adding a constant C to each of the targets* y_i *would not simply result in a shift of the predictions by the same amount C.*[1]

2.3如何判断求导是否正确

可以使用数值求导的方式对求导结果进行验证。

$$\frac{\partial \mathrm{f}}{\partial \mathbf{X}_{ij}} \approx \frac{f(\boldsymbol{X}+\boldsymbol{\delta_{ij}})-f(\boldsymbol{X}-\boldsymbol{\delta_{ij}})}{2\varepsilon}$$

其中，$\boldsymbol{\delta_{ij}}$ 给输入 $\mathbf{X}_{ij}$ 引入很小的变化值 ε。

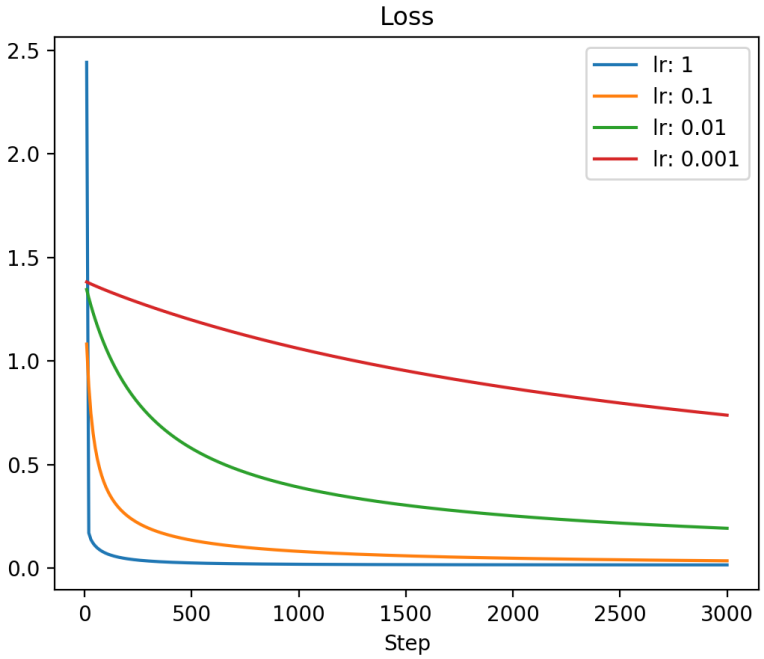
在代码中，numerical_gradient.py 对其进行了实现，随机初始化较小的输入，并且对 fc layer 以及 softmax loss layer 的求导结果进行了比较，结果如下。

```
dw - dw_num:
[[  1.18083321e-11  -6.20214990e-12  -1.17461596e-12   8.80504558e-11]
 [ -3.04312131e-12  -1.75511827e-11   6.27631280e-11   4.55813165e-12]
 [  7.90467691e-12  -5.86726223e-11  -7.29389882e-11  -3.50164342e-12]
 [ -1.41453516e-11   4.41984227e-11   1.03900222e-11  -6.09854389e-11]
 [ -4.47175630e-12  -1.35882416e-11   4.65449901e-12  -2.14566143e-11]]
db - db_num:
[ -5.32307531e-12   1.24980026e-11  -2.43760567e-12  -6.56488197e-11]
dx - dx_num:
[[ -1.38152788e-11  -2.62107003e-12  -4.13612894e-12   2.05724951e-11]
 [ -9.14053555e-12  -7.12627266e-12  -8.88997209e-12   1.40545596e-11]
 [  4.19228541e-12  -3.56407265e-12  -2.58069781e-12   1.30546962e-11]
 [ -1.23837468e-11  -3.03774020e-12   3.40544606e-12  -1.01884334e-11]
 [ -1.11661999e-11   6.75032079e-12  -8.27427363e-12  -9.51430601e-12]
 [  6.49234963e-12   2.78617407e-13  -7.36244399e-12   5.91469582e-13]
 [ -1.21078148e-12  -1.21807460e-11  -2.04023430e-11   4.87151985e-13]
 [  5.54135904e-12  -7.83232507e-12  -8.71458461e-12  -9.66952213e-14]
 [  3.08885903e-12  -1.12075106e-12  -2.24718855e-12   2.79089252e-13]
 [ -5.74122000e-12   1.11123610e-12  -1.09985111e-12  -5.37239697e-12]]
```

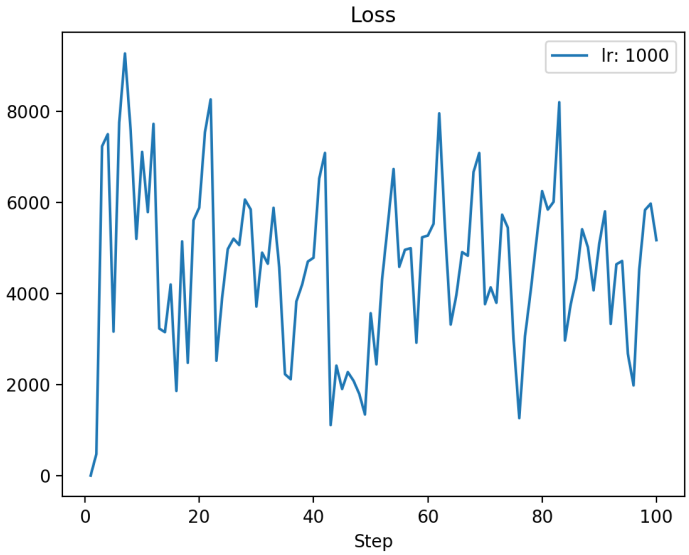
3. requirement3

3.1学习率的选择

为探究学习率对模型生成的影响，保持其他参数不变（使用 full batch gradient descent），使用不同量级的学习率，并将所有的 loss 下降曲线绘制在一张图中，见下图。



从图中可以看出，对于 full batched 条件下，学习率越大，loss 的下降速度越快，能越快的到达最优点，但是并不是学习率越大越好，如果学习率足够大时，因为每次更新的步长过大而导致模型后期（甚至从始自终）无法收敛，影响模型准确性，可以做一个极端的测试，如下图将学习率调的足够大。



因此根据以上结果提出一个大致地学习率选择策略：

- **先在不同量级训练模型并得出一个大致的最优学习率范围（要求收敛速度较快，且不会导致后期无法稳定）**
- **再将选择出的大致范围进行细分，进行训练，选取最优的模型进行使用和预测**

使用该策略时需要的训练次数较多，可以在大致选择最优范围的时候适量减少训练级的数据量来避免时间和硬件的浪费。

3.2何时终止训练

参考资料以及网上的一些教程，找到以下三种方法：

- **给定 loss 的阈值，在 loss 足够小时退出训练循环**
- **当 loss 的变化量小于某个阈值时退出训练循环**
- **给定总 epoch 数量，训练固定量的 epoch**

实际实践与探究

- 实际实现时，因为 loss 能降低到的最小值无法做到精确预估，因此方法一无法设定 loss 的阈值，故没有在代码中实现方法一
- 在探究过程中发现，当 batch 数量较小的时候，loss 变化较为剧烈，所以方法二中阈值的设定也比较麻烦。**代码中实现了方法二的终止方式**，调节阈值时发现，当 batch 过小时，最佳阈值的设定范围极窄。当使用 SGD 时，将阈值设定为 1e-3 时，训练终止过早；当阈值设定为 5e-4 时，代码很长时间都未终止。（因为可能无法终止，所以探究代码没有留下）代码实现时使用了权重和的方式，降低停止的偶然性（如图）。

```
# terminate when change of loss is small
dloss = self.beta * dloss + (1 - self.beta) * abs(loss_pre - loss)
```

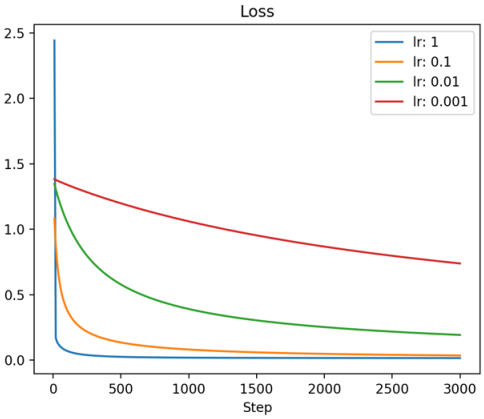
- **为了其他探究的方便，代码中也实现了第三种终止方式**，当第二种终止方式超参未设置时，使用此方法。

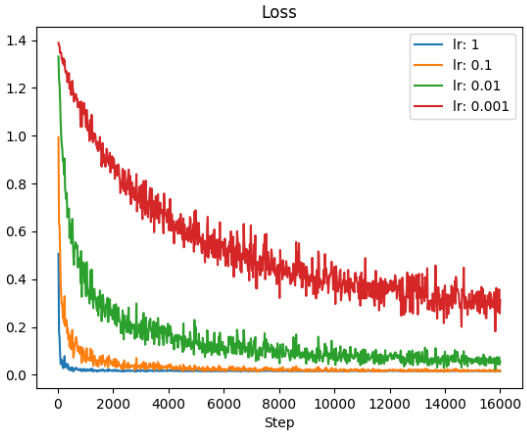
4. requirement4

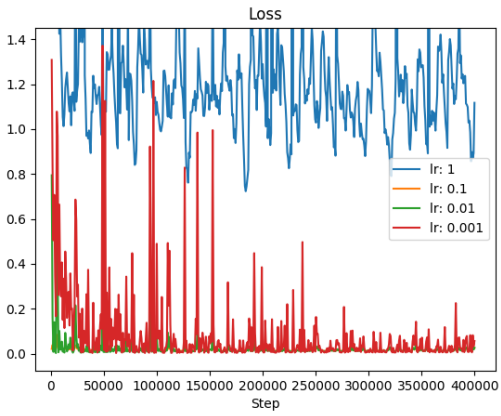
4.1 三种梯度下降方式的探究

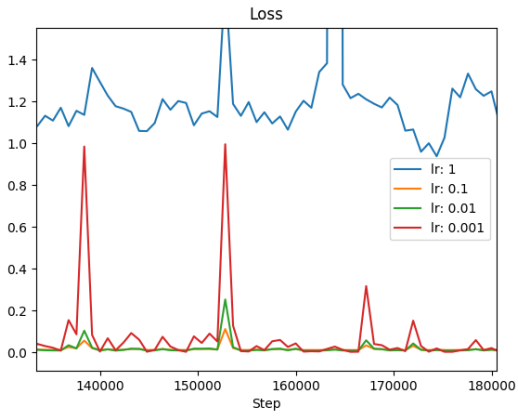
对学习率的敏感度不同

以下四张图，左上为 full batched gradient descent，右上为 batch size 为 50 的 Mini Batch gradient descent，下两张图为 SGD 的 loss 图像（左图为整体，右图为局部）。



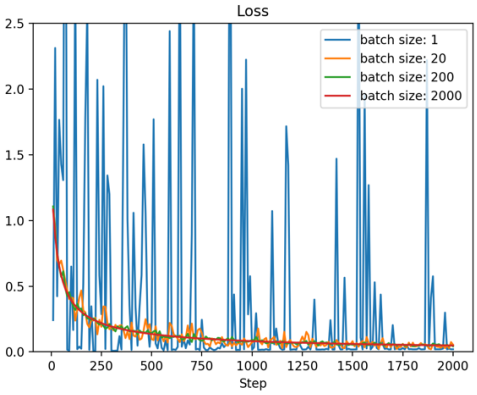






通过比较几个图可以发现，当学习率过小或者过大时，loss 曲线的变化会趋于剧烈。而且当 batch size 越小，不稳定性越大，学习率的有效选取范围会变小。

方向准确性的不同



上图为不同 batch size 下的 loss，横坐标为 step。

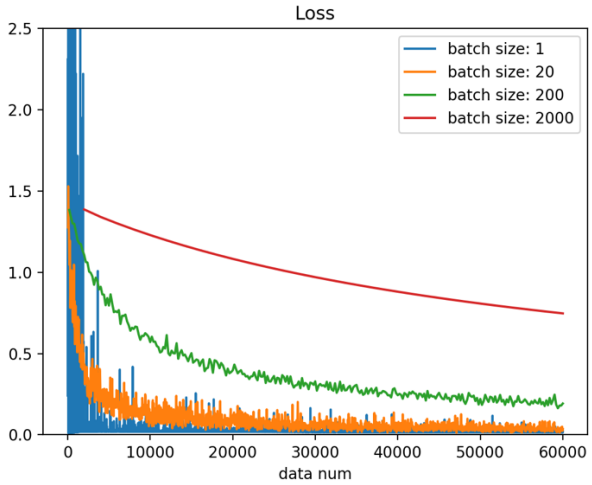
可以得到两个结论：

- 其他条件相同的情况下，batch size 越小，模型的更新方向的不确定性越大，loss 函数的曲折变化程度越大，模型训练稳定性越差
- 其他条件相同、不同的 batch size 条件下，loss 数值下降的大致趋势基本相同，可以初步断定 batch size 越小，模型训练效率越高

可以很直觉性地理解该现象产生的原因。SGD 时，下降方向由一个元素决定，因此随机性最大；而随着 batch size 的增大，非下降方向上的随机性相互抵消，下降方向就趋于稳定；当 full batch 时，得到全局最优的下降方向。

效率的不同

下图为不同 batch size 下的 loss，横坐标为计算所使用的总数据量。



可以得知，batch size 越小，收敛到最优值附近时所需要的数据量越小，可以理解为计算量越少。

但需要注意的一点是，上述计算量的衡量是在使用 CPU 的基础下。使用 GPU 时，在没有超出 GPU 显存的情况下，batch size 增大，计算速度是相对恒定的。再结合稳定性的特点，在 GPU 条件下，选取合适 batch size 的 mini batch 梯度下降方式能够达到很好的计算效率和稳定性。

4.2 下降方式各自优缺点

综上，可以得到：

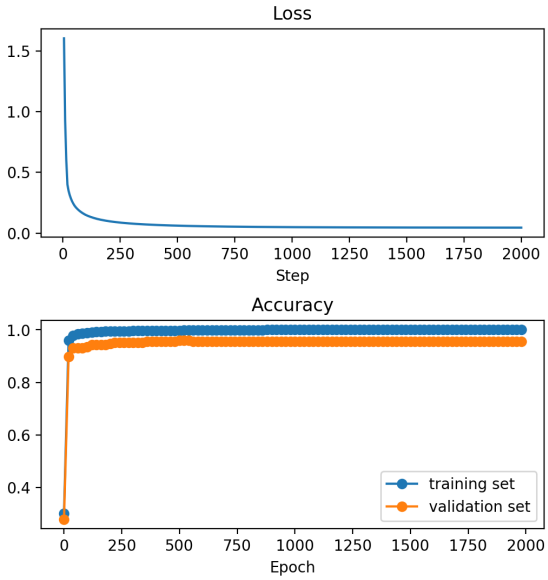
- batch size 越小，对学习率的敏感度增大（即较优 learning rate 选取范围会减小）
- batch size 越小，loss 下降的方向越不稳定
- batch size 越小，数据收敛所需要的数据量越小

5. requirement5

因为代码中实现了validation set和training set，所以调节不同的学习率及其他参数，选取在validation set上获得最优的模型进行验证（代码见part2/main.py）。

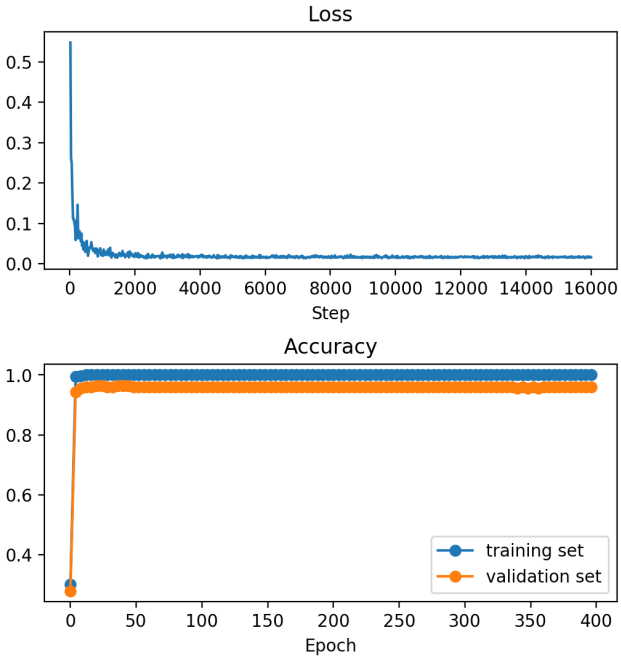
5.1 Full Batched GD（92.25%）

```
best hyparameter:  {'learning_rate': 0.5, 'lr_decay': 0.9, 'lr_decay_freq': 50, 'batch_size': 2000, 'num_epochs': 2000, 'delt
a_loss': 0, 'beta': 0.9, 'save_freq': 5, 'check_freq': 20}
test acc:
0.9244652406417112
```



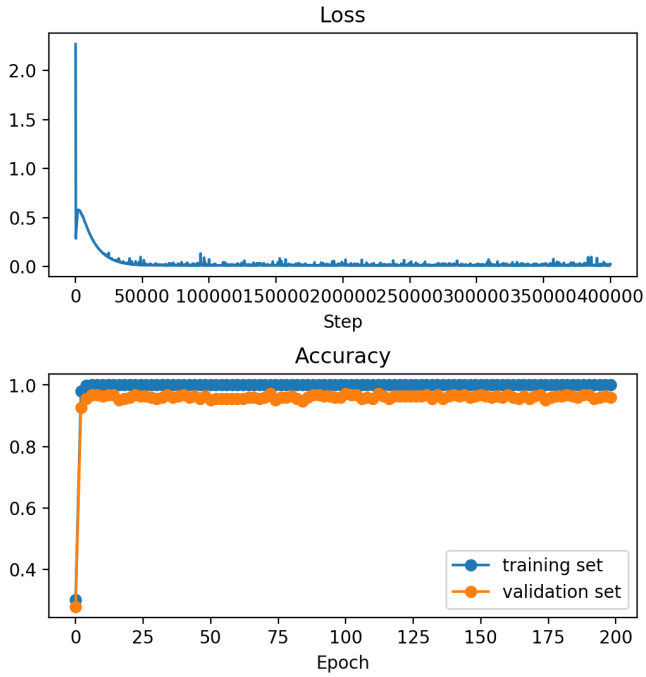
5.2Mini Batch GD（Batch size 为 50）（91.71%）

```
best hyparameter:  {'learning_rate': 0.5, 'lr_decay': 0.9, 'lr_decay_freq': 50, 'batch_size': 50, 'num_epochs': 400, 'delta_l
oss': 0, 'beta': 0.9, 'save_freq': 20, 'check_freq': 4}
test acc:
0.9171122994652406
```



5.3 SGD（92.25%）

```
best hyparameter:  {'learning_rate': 0.5, 'lr_decay': 0.9, 'lr_decay_freq': 50, 'batch_size': 1, 'num_epochs': 200, 'delta_lo
ss': 0, 'beta': 0.9, 'save_freq': 200, 'check_freq': 2}
test acc:
0.9224598930481284
```



6. Supplementary

6.1validation set 设置

因为数据量较小容易造成 overfitting，为了评估 model 训练的效果，所以代码里实现了 validation set，用于模型对评估。

6.2learning rate decay

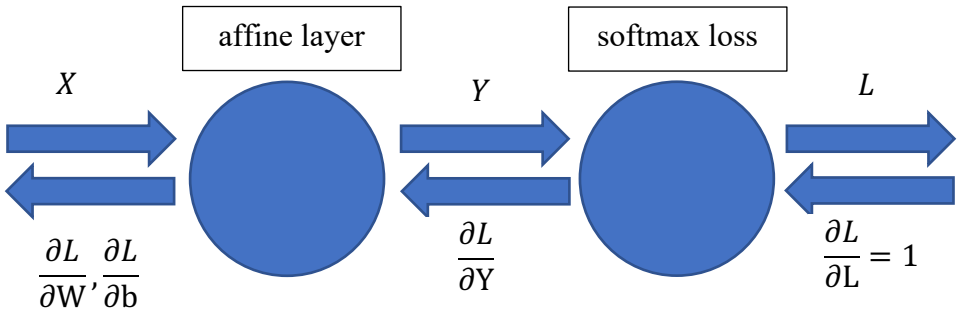
代码中实现了 learning rate decay 的方法，可以设置 learning rate 的减小的比例以及速率，从而减小 batch size 对 learning rate 的敏感度，让模型前期的训练更快。

6.3代码结构组织

代码参照 cs231n[2]中的代码组织，将 trainer、layer、model 和 data processor 分开，虽然代码量较大，但是方便了代码的调试，增加代码的可复用性。

附页

求 loss 函数对 W 和 b 的偏导数



在代码中，求导过程利用了链式法则实现反向传播(如上图所示)，因此以下将对两部分的导数分别进行求解。

Loss 函数为

$$L = -\frac{1}{N}\sum_{n=1}^{N} y_n log\hat{y}_n + \frac{\lambda}{2}\|W\|^2$$

令

$$S = -\frac{1}{N}\sum_{n=1}^{N} y_n log\hat{y}_n$$

则有

$$\frac{\partial \mathrm{L}}{\partial \mathrm{W}} = \frac{\partial \mathrm{S}}{\partial \mathrm{W}} + \lambda W = \frac{\partial \mathrm{S}}{\partial \mathrm{Y}}\frac{\partial \mathrm{Y}}{\partial \mathrm{W}} + \lambda W \tag{1}$$

$$\frac{\partial \mathrm{L}}{\partial \mathrm{b}} = \frac{\partial \mathrm{S}}{\partial \mathrm{b}} = \frac{\partial \mathrm{S}}{\partial \mathrm{Y}}\frac{\partial \mathrm{Y}}{\partial \mathrm{b}} \tag{2}$$

softmax layer

不失一般性，考虑 softmax Loss 函数对输入的第 n 行进行求导，假设第 n 行的类别为c_n，为简化公式以下用 j 表示c_n，总类别数为C。

定义符号，

$\boldsymbol{o_i}$*表示长度为 C，第 i 位为 1 其余为 0 的 one-hot 向量*

$$\boldsymbol{p_{ni}} = \frac{e^{Y_{ni}}}{\sum_{k=1}^{C} e^{Y_{nk}}}$$

则有

$$S = -\frac{1}{N}\sum_{n=1}^{N} log\, p_{nj}$$

以下求解$\frac{\partial \mathrm{S}}{\partial \mathrm{Y}_n}$，

$$\frac{\partial \mathrm{S}}{\partial \mathrm{Y}_n} = \frac{\partial \mathrm{S}}{\partial p_{nj}}\frac{\partial p_{nj}}{\partial \mathrm{Y}_n} \tag{3}$$

其中，

$$\frac{\partial S}{\partial p_{nj}} = -\frac{1}{N}\frac{1}{p_{nj}} \tag{4}$$

计算 $\frac{\partial p_{nj}}{\partial \mathrm{Y}_n}$ 需要分成两种情况

i ≠ j 时，

$$\frac{\partial p_{nj}}{\partial \mathrm{Y}_{ni}} = \frac{-e^{Y_{ni}}e^{Y_{nj}}}{(\sum_{k=1}^{C} e^{Y_{nk}})^2} = -p_{ni}p_{nj} \tag{5}$$

i = j 时，

$$\frac{\partial p_{nj}}{\partial \mathrm{Y}_{nj}} = \frac{e^{Y_{nj}}(\sum_{k=1}^{C} e^{Y_{nk}}) - e^{Y_{nj}}e^{Y_{nj}}}{(\sum_{k=1}^{C} e^{Y_{nk}})^2} = p_{nj}(1 - p_{nj}) \tag{6}$$

结合(3)(4)(5)(6)四式，可得，

$$\frac{\partial S}{\partial \mathrm{Y}_n} = \frac{(\sum_{k=1}^{C} o_k p_{nk}) - o_{c_n}}{N} \tag{7}$$

affine layer

$$Y = WX + b$$

由等式容易得到 Y 对 W 和 b 对偏导数，

$$\frac{\partial \mathrm{Y}}{\partial W} = X \tag{8}$$

$$\frac{\partial \mathrm{Y}}{\partial b} = 1 \tag{9}$$

综合上述(1)(2)(7)(8)(9)，可以得到 $\frac{\partial \mathbf{L}}{\partial \mathbf{W}}$ ，$\frac{\partial \mathbf{L}}{\partial \mathbf{b}}$ ，在代码中已用后向传播的方式实现（代码对应各个部分偏导的结果）。

参考文献和网站

[1] Hastie, T., R. Tibshirani, and J. Friedman (2001). The Elements of Statistical Learning. Springer.
[2] http://cs231n.stanford.edu/2017/syllabus