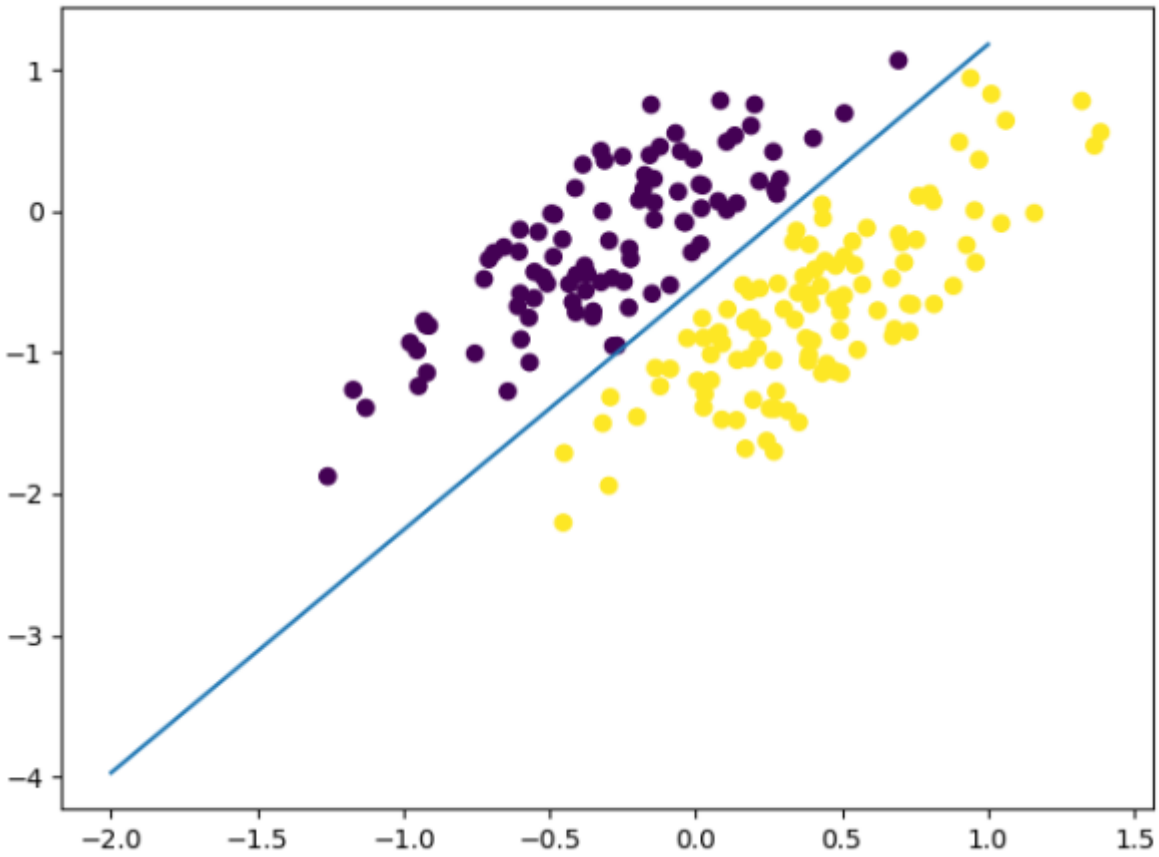
Assignment 2 of PRML

Part 1

- **Least Square Model（最小平方法）**

  最小平方法，即为使平方误差和函数$E_D(\omega)=\frac{1}{2}\sum_{n=1}^{N}(t_n-\omega^T\phi(x_n))$最小的情况下，所得参数矩阵 ω，求导可得$\omega=(\Phi^T\Phi)^{-1}\Phi^Tt$。对于本题目，数据是二维的，即结果为$w_0+w_1x+w_2y$。首先将 x 增广为第一行为1的矩阵（确保计算偏置量），之后按照LSM的求导最终公式，代入计算即可。调用dataset的内置acc函数评估，所得w_0,w_1,w_2以及准确度为：

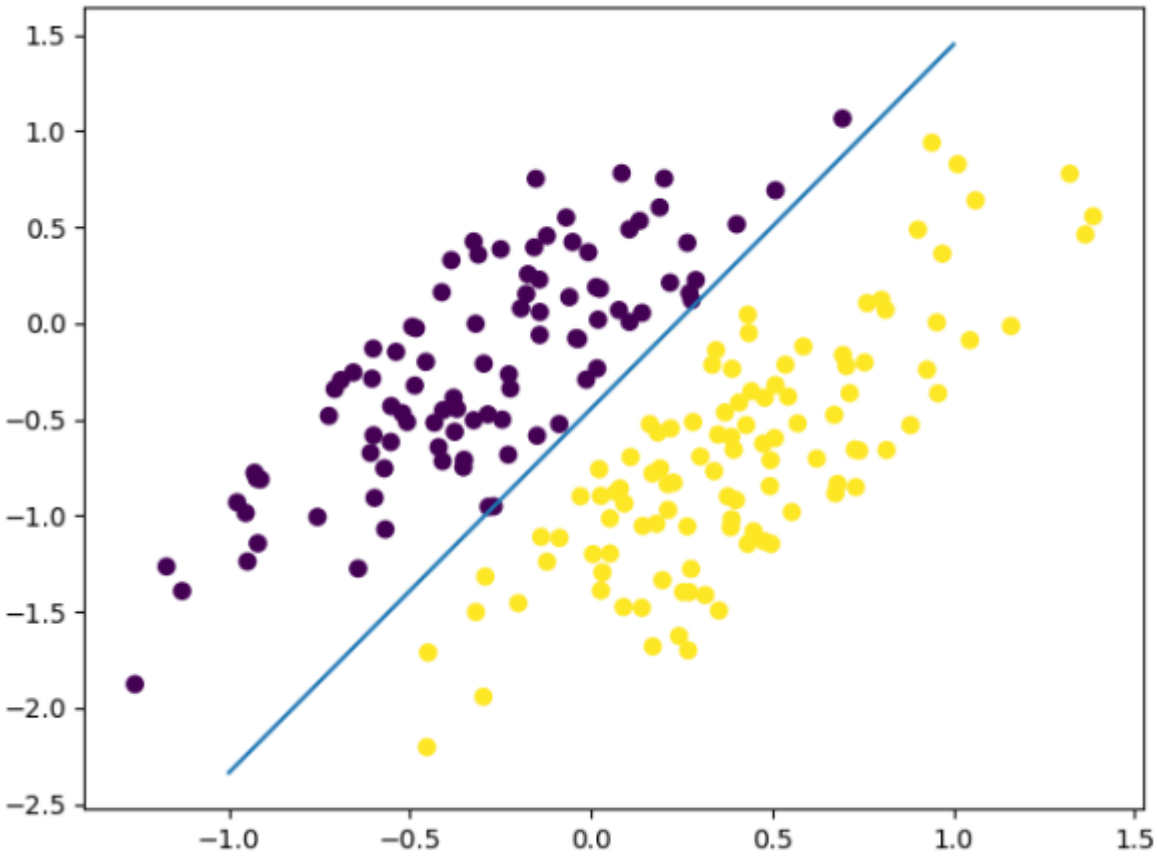
  $w_0=-0.518673, w_1=1.654558, w_2=-0.964092, accuracy=1.0$



- **Perceptron Algorithm（感知器方法）**

  感知器模型模拟视神经控制系统进行识别，在本问题中同LSM一样，仍是获得一个w 参数矩阵，得到$w_0 + w_1 x + w_2 y$ 的分界线。随机选择w_0, w_1, w_2 作为初始参数矩阵，在每次的迭代中采用随机梯度下降法进行参数修正，如果存在错分数据（即$t(w_0 + w_1 x + w_2 y) < 0$）,则根据学习率修正参数。但由于本次任务中数据量过小，在学习率为0.1的情况下，也能在3—5次的迭代后达到正确率1.0结果。而学习率设为0.01时，迭代次数需要达到10000左右。最终所得w_0, w_1, w_2以及准确度为：

  $w_0 = -0.191480, w_1 = 0.819121, w_2 = -0.432681, accuracy = 1.0$



Part 2

Ⅰ. 实现预处理

- **获取字典**

  预处理过程的第一步是获取字典。在本次任务中，将训练集和测试集分开，**仅从训练集中**获得字典。具体处理方法包括替换`string.whitespace`与`string.punctuation`中的字符为空格，然后将它们按空格分割。除此之外，观察到字典中含有'a14'，'16m1'等字样，认定它们对于内容分析没有意义，于是将其全部忽略。但对于仅由阿拉伯数字组成的字符串予以保留。最后按照mincount = 10的标准过滤出最终的字典。

- **得到multi-hot向量**

  将dataset中的训练集和测试集数据取出，把列表转化为矩阵即可。

- **得到one-hot向量**

  将dataset中的训练集和测试集的target项取出，0、1、2、3分别对应[0 0 0 1]、[0 0 1 0]、[0 1 0 0]、[1 0 0 0]，把列表转换为矩阵即可。

Ⅱ. 求解梯度变化

- **计算** $\frac{\partial L}{\partial W_{i,j}}$ **，** $\frac{\partial L}{\partial b_i}$

  2. 现在求dL， 其中：

  $d(ln(1^T exp(\hat{y}))) = \frac{1}{1^T exp(\hat{y})} \odot d(1^T exp(\hat{y}))$

  $d(1^T exp(\hat{y})) = 1^T d(exp(\hat{y})) = 1^T(exp(\hat{y}) \odot d\hat{y})$

  故得到：

  $dL = \frac{1^T(exp(\hat{y}) \odot d\hat{y})}{1^T exp(\hat{y})} - y^T d\hat{y}$

  3. 由矩阵迹的性质$tr(A^T(B \odot C)) = tr((A \odot B)^T C) = \sum_{i,j} A_{ij}B_{ij}C_{ij}$

  可得到：

  $dL = tr(\frac{1^T(exp(\hat{y}) \odot d\hat{y})}{1^T exp(\hat{y})}) - tr(y^T d\hat{y})$

  $= tr((\hat{y} - y)d\hat{y})$

  $= tr((\frac{\partial L}{\partial \hat{y}})^T d\hat{y})$

  4. 由 $d\hat{y} = d(W^T x) = (dW^T)x + W^T dx = (dW^T)x$与矩阵迹恒等式：

  $dL = tr((\frac{\partial L}{\partial \hat{y}})^T (dW^T)x)$

  $= tr(x(\frac{\partial L}{\partial \hat{y}})^T dW^T)$

  $= tr((\frac{\partial L}{\partial W^T})^T dW^T)$

  也即 $\frac{\partial L}{\partial W^T} = \frac{\partial L}{\partial \hat{y}} x^T = (\hat{y} - y)x^T$

  由矩阵运算性质得到**结果**：

  $\frac{\partial L}{\partial W} = -x(y - \hat{y})^T$

  5. 同理，计算$\frac{\partial L}{\partial b}$ 只是缺少一个系数，或者说系数为1^T,故有以下**结果**：

  $\frac{\partial L}{\partial b} = -(y - \hat{y})^T$

- **计算正则化项**$\frac{\partial \lambda ||W||^2}{\partial W}$

正则化项的求导结果是显然的，即 $2\lambda W$

- **（1）关于是否对bias进行正则化**

  首先讨论正则化项存在的意义。为了防止出现过拟合现象（即训练集数据准确度良好，但测试集准确度较差），我们需要加入惩罚项来平衡这种情况，使得最终得到的推断式能够普适于所有数据集。在实际过程中，我们通常采用较小的权值来构造模型，因而使用了L2正则化来获得小的参数，避免过拟合现象。

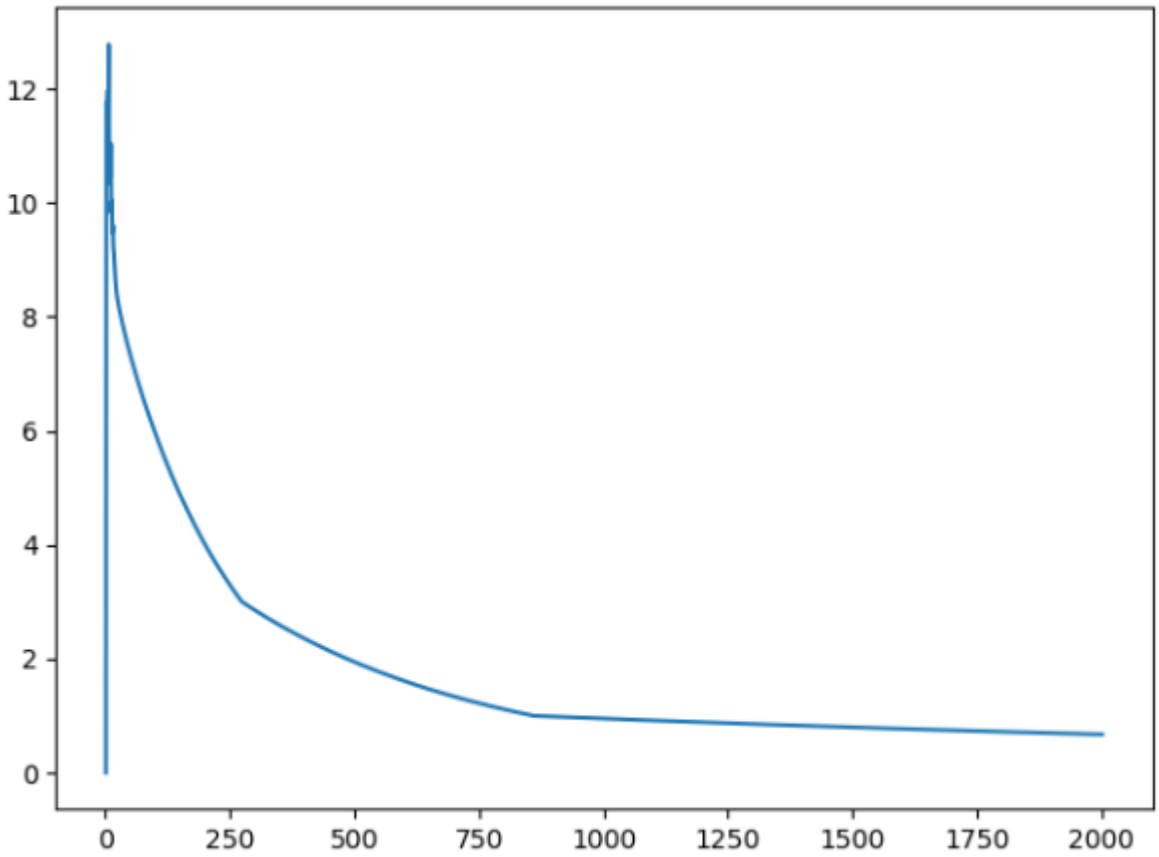
  而**bias不需要正则化**。可以知道，在对参数矩阵 W 进行正则化的前提下，数据x发生微小变化主要受乘法项系数W的影响，即使对bias进行L2正则化，也不会在数据偏移的情况下对结果产生较大的影响，因此没有必要对bias进行正则化。

- **（2）关于解析梯度的确定**

  对于梯度的计算，根据上述证明中，可在每次迭代中计算得到相应的解，但因为反向传播时存在误差，每次得到的更新后参数并不能达到很好的拟合效果。如果使用梯度检验，即每次根据公式得到梯度后，和反向传播的结果比较，在相差很小的情况下，可以认为梯度是正确的。

Ⅲ. 学习率与终止条件

- 使用Full Batch Gradient Descent方法，得到下图**Loss Curve**：



参数：learn_rate = 0.1 learn_time = 2000 $\lambda = 0.01$

输出：

```
1998 ********loss =  0.6724042852424426
1999 ********loss =  0.6721782512842126
2000 ********loss =  0.6719523081075041
The training accuracy is:  1.0
The test accuracy is:  0.9251336898395722
```

> 注：为便于参照，图像中loss起始设为 0。后不再赘述

- **（1）关于确定学习率**

  以使用FBGD方法的过程为例，确定学习率的过程如下：

  - 第一次实验，全过程采用统一学习率 0.1，发现loss的下降速度过慢，因此得知在不同过程需要采用不同的学习率，即在训练初期使用较大的学习率，快速下降至理想范围；在训练后期使用较小的学习率，避免因为数据波动而产生较大的偏移，保证能够稳定地收敛；
  - 第二次实验，观察loss随着迭代过程的变化，大致确定**loss在3—10的范围内学习率设为1.0；loss在1—3的范围内学习率设为0.5；loss在0—1的范围内，使用较低的学习率，例如0.1或0.01（由传入参数决定）**；
  - 第三次实验及以后，进一步观察loss的变化，并开始作图，当图像能够快速逼近收敛范围，并能在收敛过程中保持稳定时，可以基本确定学习率和相应的变化规律。

- **（2）关于何时终止训练**

  以使用FBGD方法过程为例，讨论如何确定终止条件。

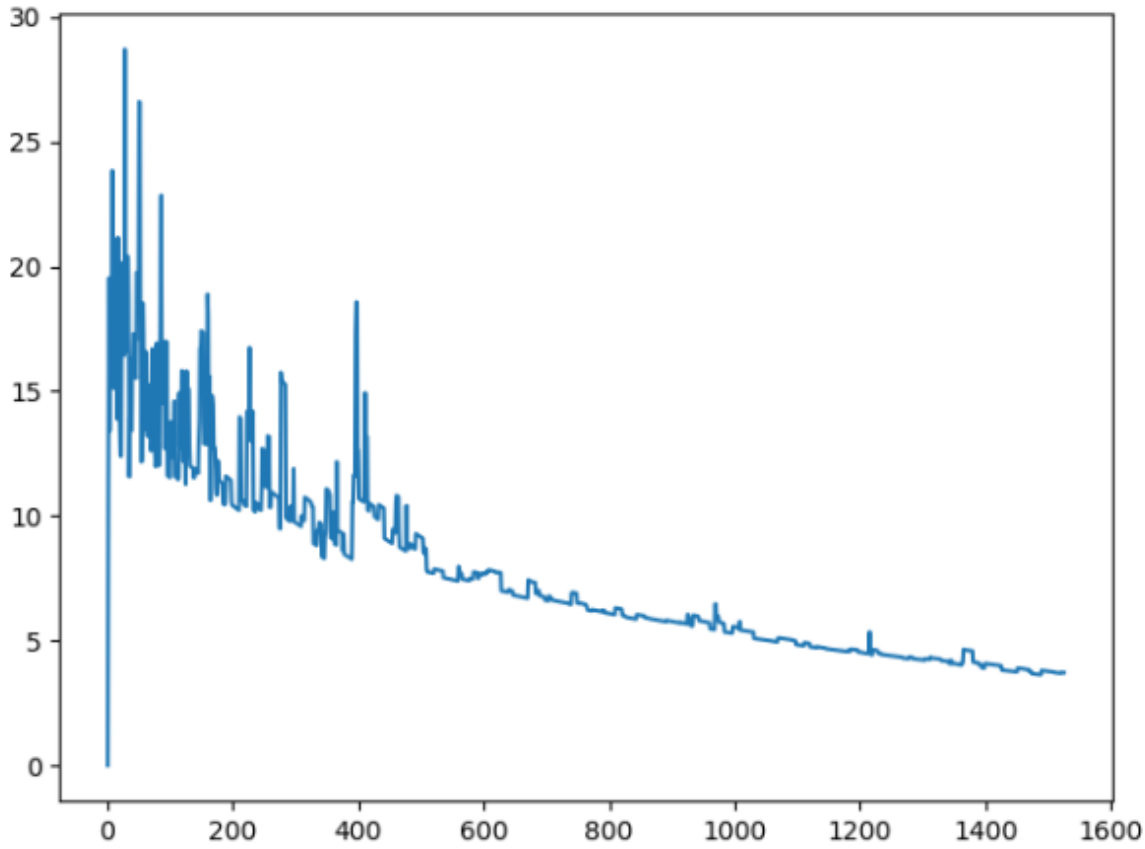
  - **根据迭代次数**设立终止条件。实际操作中，应先进行实验观察loss的变化，大致确定一个迭代次数的范围，使得训练能够达到较优的收敛条件。对于FBGD法，本次实验确定其迭代次数为2000；
  - **根据**$\Delta loss$ 设立终止条件。$\Delta loss$ 即后一次交叉熵与前一次交叉熵的差值，可以作为判断是否收敛的典型参数。在本次任务中，设置差值epsilon为10^{-5} 。
  - **根据收敛点的个数**设立终止条件。 即考虑满足$\Delta loss < 10^{-5}$ 的收敛点有多少个，作为判断收敛的标准。

  本次实验中，结合前两种方法设定终止条件。以FBGD方法为例，即：**当迭代次数超过2000或者**$\Delta loss < 10^{-5}$**时终止训练。**

Ⅳ. 对比：FBGD, SGD, BGD

- **FBGD**：见Ⅲ中的曲线图。
- **SGD**： 由于SGD是每次单独选取一个点进行迭代，可以预见到每次迭代结果的波动会很剧烈。
  - 第一次实验.

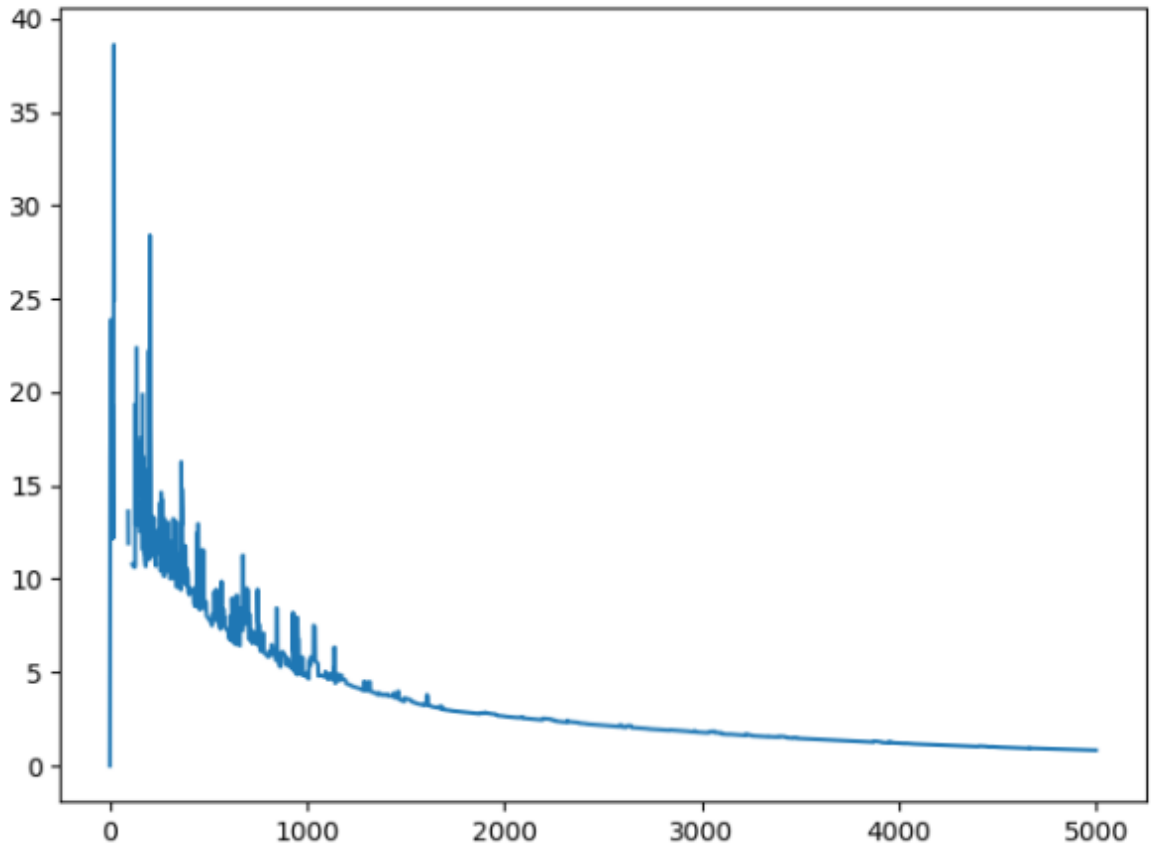
    学习率梯度：$loss \in (3, 30) : rate = 0.5,$
    $loss \in (0, 3) : rate = 0.1(\text{由参数决定})$



可以看到，图像波动剧烈，而且收敛时交叉熵仍是一个很大的数值，显然不可取，故进一步做一些调整。

- 第二次实验.

  学习率梯度：$loss \in (8, 30) : rate = 0.5$
  $loss \in (3, 8) : rate = 0.2$
  $loss \in (0, 3) : rate = 0.1$(由参数决定)

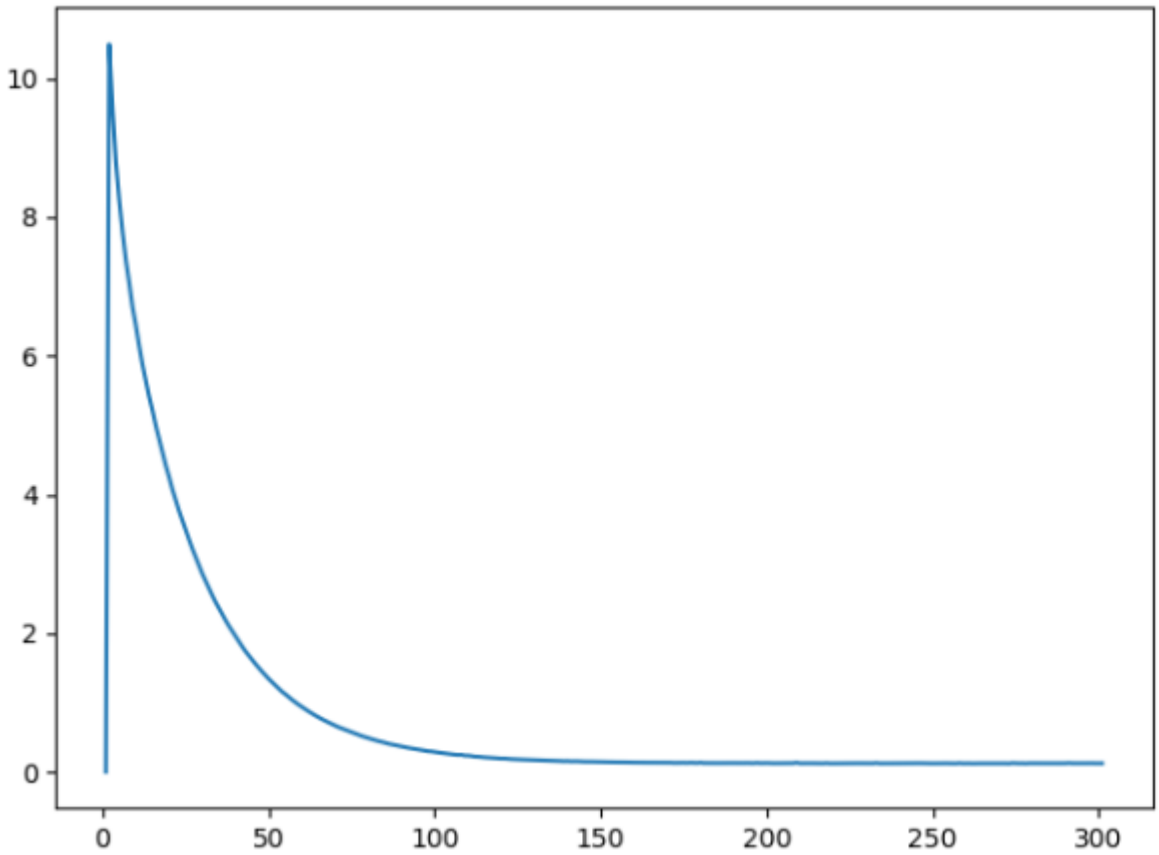



```
4998 ********loss =  0.9477738932146422
4999 ********loss =  0.9474024707179027
5000 ********loss =  0.9470423281393737
The training accuracy is:  0.9924343569203382
The test accuracy is:  0.911096256684492
```

可以看到，调整学习率梯度后，效果明显变好，也能够达到不错的准确率。与FBGD相比，图像抖动更为剧烈，但整体趋势是收敛的。

- **BGD：**BGD则是前两种方法的折衷策略，选取一定的batch进行训练，甚至没有规划学习率梯度，便得到了如下的结果：

  其中 **batch = 100**




```
299 ********loss =  0.12442466292859763
300 ********loss =  0.12309292037632028
The training accuracy is:  0.9933244325767691
The test accuracy is:  0.9258021390374331
```

  可以看出，BGD方法仅用300次迭代就达到了良好的准确率，且速度比FBGD快，收敛较于SGD稳定，同时吸收了前两个方法的优点。但是batch的选取又是一个重要考虑的内容。

- **总结三种方法的优缺点**

  FBGD

  - 优点：使用所有数据，避免局部最优的情况，迭代次数较少；
  - 缺点：当样本数据量很多时训练速度缓慢；

  SGD

  - 优点：随机选取单个数据，训练速度非常快；
  - 缺点：受噪声影响较大，容易达到局部最优解，迭代次数多，且不利于并行实现；

  BGD

  - 优点：综合了前两者的优点，既能加快迭代速度，又能避免噪声影响，保证能够快速地获得良好训练结果；
  - 缺点：挑选batch参数成为新的问题，需要多次实验选取合适的batch值

V. 最终结果

- **FBGD**

  learn_rate： $loss \in (3,30) : rate = 1$
  $loss \in (1,3) : rate = 0.5$
  $loss \in (0,1) : rate = 0.1$

  $\lambda = 0.001$

  迭代2000次后，准确率为**92.51%**

```
1998 ********loss =  0.6724042852424426
1999 ********loss =  0.6721782512842126
2000 ********loss =  0.6719523081075041
The training accuracy is:  1.0
The test accuracy is:  0.9251336898395722
```

- **SGD**

  learn_rate: $loss \in (8,30) : rate = 0.5$
  $loss \in (3,8) : rate = 0.2$
  $loss \in (0,3) : rate = 0.1$

  $\lambda = 0.001$

  迭代5000次后，准确率为**91.11%**

```
4998 ********loss =  0.9477738932146422
4999 ********loss =  0.9474024707179027
5000 ********loss =  0.9470423281393737
The training accuracy is:  0.9924343569203382
The test accuracy is:  0.9110962566844492
```

- **BGD**

  learn_rate = 0.1

  $\lambda = 0.001$

  迭代300次后，准确率为**92.58%**

```
299 ********loss =  0.12442466292859763
300 ********loss =  0.12309292037632028
The training accuracy is:  0.9933244325767691
The test accuracy is:  0.9258021390374331
```

  r