作业一实验报告

1120213081 介应奇

一、实验目标

1. 学习搭建基本的机器学习环境
2. 学习使用Pytorch库，熟悉张量的相关运算，自动求导，GPU加速运算
3. 完成时间识别任务，一共有两个事件，分别为：会见接见、考察检查，验证逻辑回归理论的正确性

二、实验内容综述

1. 从语料库中提取出现频率最高的前5000个某种些词性的集合作为词典
2. 根据词典，在训练集、验证集和测试集中提取特征向量
3. 使用从训练集中提取的特征向量，分别对两个事件的发生进行相应的模型训练
4. 训练过程中梯度下降使用四种方式进行尝试对比并其中差异
    a. 批量自动求导梯度下降
    b. 随机自动求导梯度下降
    c. 随机手动求导梯度下降
    d. 批量手动求导梯度下降
5. 查看训练集的损失函数随着迭代次数增加，损失函数值的变化，以及F1-measure、查准率、查全率的变化
6. 查看损失函数的值在验证集中随着迭代增加的变化情况，主要观察是否会出现过拟合现象

三、实验原理

通过使用独热编码的方式对句子进行特征提取，使用提取的特征向量，通过逻辑回归的方式计算出损失函数，利用逻辑回归损失函数是凸函数的性质：局部最优解即使全局最优解，通过迭代计算对目标函数，也就是可以对句子中目标事件是否发生做出判断的函数；随着损失函数的值变小/大（取决于是否给损失函数加负号）；在训练集中，我们的函数对于某件事件发生这件事的拟合程度会越来越高，并逐渐接近于100%，但是如果直接取最终接近100%的这个值，那么这个函数会出现过拟合的现象，也就是泛型会变差，因此我们要引入验证集进行验证，在即将过拟合的点上停止，最终的最优解应该在这个点的附近。

数学基础：概率论、微积分、线性代数

逻辑回归使用原理表达式

$$p(y|x;\theta) = h_\theta(Y=1)^y\left(1-h_\theta(Y=1)\right)^{1-y}$$

$$L(\theta) = \prod_{i=1}^{i=m} p\left(y^{(i)}|x^{(i)};\theta\right)$$

手动求导的迭代表达式

1 批量梯度下降

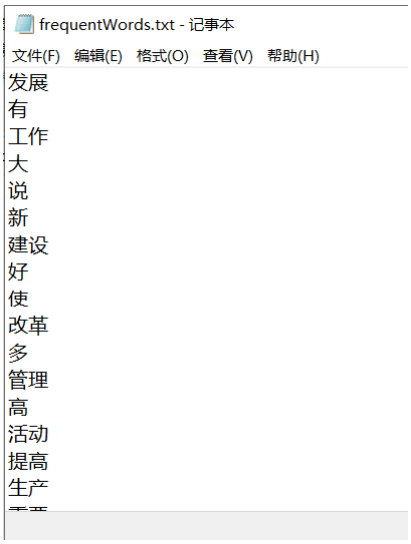
$$\theta := \theta + \alpha\left(\sum_{i=1}^{m}\left[y^{(i)} - g(\theta^T x^{(i)})\right]x^{(i)}\right)$$

2 随机梯度下降

$$\theta := \theta + \alpha\left(\left[y^{(i)} - g(\theta^T x^{(i)})\right]x^{(i)}\right)$$

四、实验过程

1、提取高频词汇



由于检查两种事件是否发生，我对所有的及物动词和形容词做了提取，取这些集合中的前5000个出现频率最高的词作为字典，将他们存放在了`frequentWords.txt`文件中，以每一行一个词的方式进行了存储，提取用的脚本是`get_frequent_verb_words.py`，效果如左图：

2、提取特征向量

完成这一工作的是`extract_feature_vector.py`这个脚本，这个脚本负责提取对应集所对应的句子的特征向量，在文件中选择相应的参数作为输入的文件路径即可，如果想要运行，请将相应的数据文件放在对应路径下，由于数据是顺序排列的，我直接使用一个while循环，同时读取y数据和对应的语料数据，然后在词典中查找这个词的出现情况，出现则这个维度的量记为1，否则为0；特征向量共有5001维，最后一维用来记录其他未出现的词，如果句子中有其他未在词典中出现的词，我们就将这一维记为1，

否则为0，也就是句子中的所有词都在词典中出现的情况，不同集对应的结果记录分别在`train_x.txt、validtion_x.txt、test_x.txt`文件中，记录的格式是，每一行是一个5001维的向量，只记录01序列，用空格隔开。

3、训练模型

这部分的代码全部写在了`cal.py`脚本中，这个脚本中主要实现了训练模型所需要的全部步骤：

1. 导入train_x矩阵和train_y矩阵作为训练数据，导入validation_x矩阵和validation_y矩阵作为验证数据，导入test_x矩阵和test_y矩阵作为测试模型效果的数据，将y数据分为两类，分别对应两个事件的识别，下面是对相应模块的函数的说明文档：

```
# 输入：y数据来源路径
# 输出：分离的y1向量和y2向量，均是在CPU上的向量
def y_transform_to_tensor(path):
    return torch.Tensor(y1_list), torch.Tensor(y2_list)


# 输入：x数据来源路径
# 输出：2维x矩阵的列表
def x_transform_to_tensor(path):
    return x_list_transform
```

2. 对逻辑回归的损失函数对参数θ进行迭代计算，我使用了下面4中方式进行尝试，下面四种方式均使用了GPU加速运算

    a. 批量自动求导

    这部分工能定义在了如下函数中

```
# 输入：x矩阵列表，对应的y值列表，迭代步长，迭代次数，是否调用向量运算进行加速
# 输出：损失函数在迭代过程中的值的序列，迭代的模型序列
def batch_gradient_auto(x, y, step_rate, iterator_times, is_tensor_cal)
    return loss, theta_list
```

    is_tensor_cal参数建议使用True，经实验，如果不打开这个开关，纯使用CPU进行和GPU相同的一次运算，所需的时间我跑了2个小时也没有跑出来，最后结束了进程，而GPU向量加速运算只需要一分钟左右，具体时间见后面表格汇总

每次迭代会将所有的样本进行一遍计算，利用torch提供的自动求导模块，对每次迭代的梯度进行计算，迭代的步长是倍率机制，具体的值是通过随机生成的0-1之间的一个数

b. 批量手动求导

这部分工能定义在了如下函数中

```
# 输入：x矩阵列表，对应的y值列表，迭代步长，迭代次数
# 输出：损失函数在迭代过程中的值的序列，迭代的模型序列
def batch_gradient_hand(x, y, step_rate,
iterator_times)
    return loss, theta_list
```

其他相关的情况同上

c. 随机自动求导

这部分工能定义在了如下函数中

```
# 输入：x矩阵列表，对应的y值列表，迭代步长，迭代次数，每次迭代随机选取的样本数量，是否调用向量运算进行加速
# 输出：损失函数在迭代过程中的值的序列，迭代的模型序列
def random_gradient_auto(x, y, step_rate,
iterator_times, every_num, is_tensor_cal):
    return loss, theta_list
```

其他相关的情况同上

d. 随机手动求导

这部分工能定义在了如下函数中

```
# 输入：x矩阵列表，对应的y值列表，迭代步长，迭代次数，每次迭代随机选取的样本数量
# 输出：损失函数在迭代过程中的值的序列，迭代的模型序列
def random_gradient_hand(x, y, step_rate,
iterator_times, every_num)
    return loss, theta_list
```

其他相关的情况同上

3. 计算迭代过程中参数的变化

利用第二步中数据计算过程返回的theta_list模型参数序列，计算出随着模型迭代，验证集的损失函数的变化，相关实现在下列函数中

```
# 输入: validation x, validation y, iterator model_list
# 输出: 损失序列，最小的损失值，最小损失值出现时的模型参数，最小的参数出现在序列的哪里
def calculate_loss(x, y, model_list)
    return validation_loss, min_loss, min_model, min_i
```

```
# 输入：计算出的模型序列，测试集x矩阵，测试集y向量
# 输出：查全率列表，查准率列表，F1-measure值序列
def f1_measure(model_list, test_x, test_y):
    return recall_list, precision_list, f1m
```

4. 使用3中计算得到的参数进行绘图验证

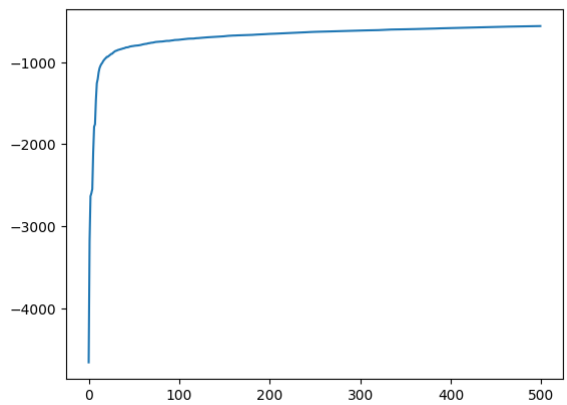
   绘图模块的实现函数说明如下

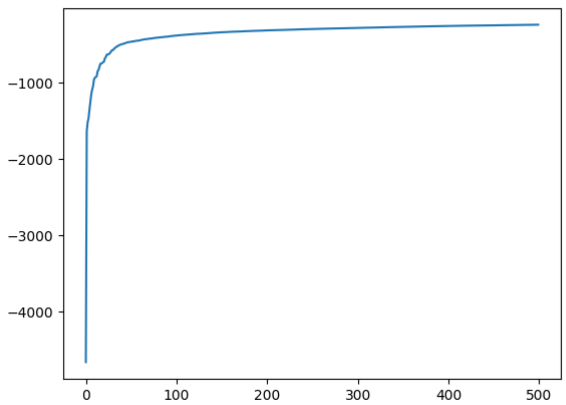
```
# 输入：对应的y值序列名称和列表，x标签，y标签，步长值，最小的位置
# 输出：显示的图片
def draw_single_line(name, y, xlab, ylab, step, min_pos)
def draw_lines(name1, y1, name2, y2, name3, y3, xlab, ylab, step, min_pos)
```

五、实验结论

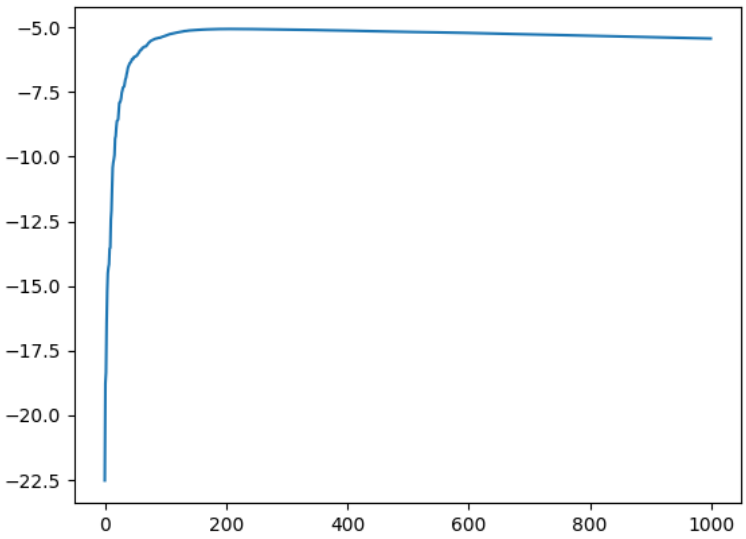
loss函数变化

首先我在使用训练集使用批量求导对模型进行训练过程中，绘制出了y1和y2的loss函数值的曲线，可以看到是，模型的损失函数绝对值快速下降，并逐渐接近0，这与理论预期相符合





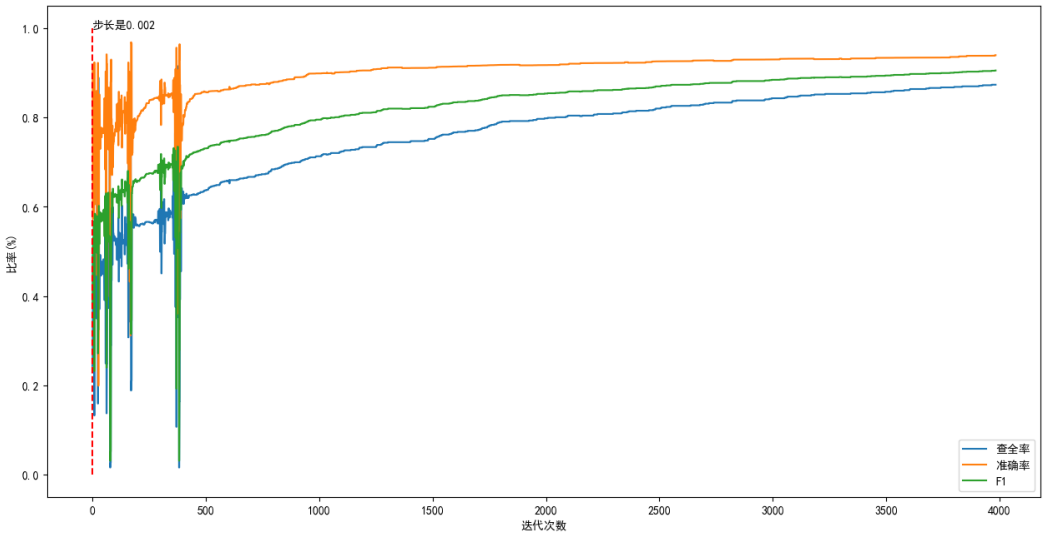
随后，我在验证集中也测试了随着迭代的进行loss函数值的变化如下图



可以看到，在第200次迭代左右的时候loss函数开始出现触底反弹的现象，也就是我们所说的”过拟合现象”，这与我们的理论预期也相符

验证模型的正确性

理论依据：如果模型正确则我们在训练集上的F1值应该随着迭代次数的增加迅速到达接近100%的数，接下来我们将训练集作为`f1_measure`函数的输入进行计算，得出下图



可以看到确实是迅速靠近了100%，说明我们的模型计算没有问题，写的是正确的

观测效果

接下来，我们在测试集中开始计算评估指标，查看我们的模型效果：

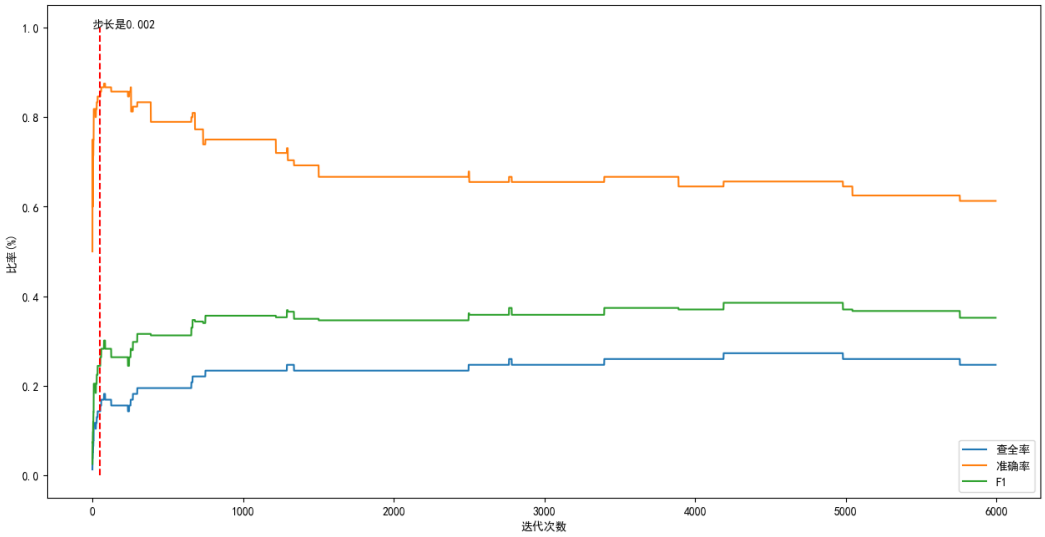
通过使用上述四种实现方式，均得到了收敛的F1-measure值，接下来我将对不同情况的结果进行总结归纳，观察loss值曲线的变化和对应的评测参数的变化：

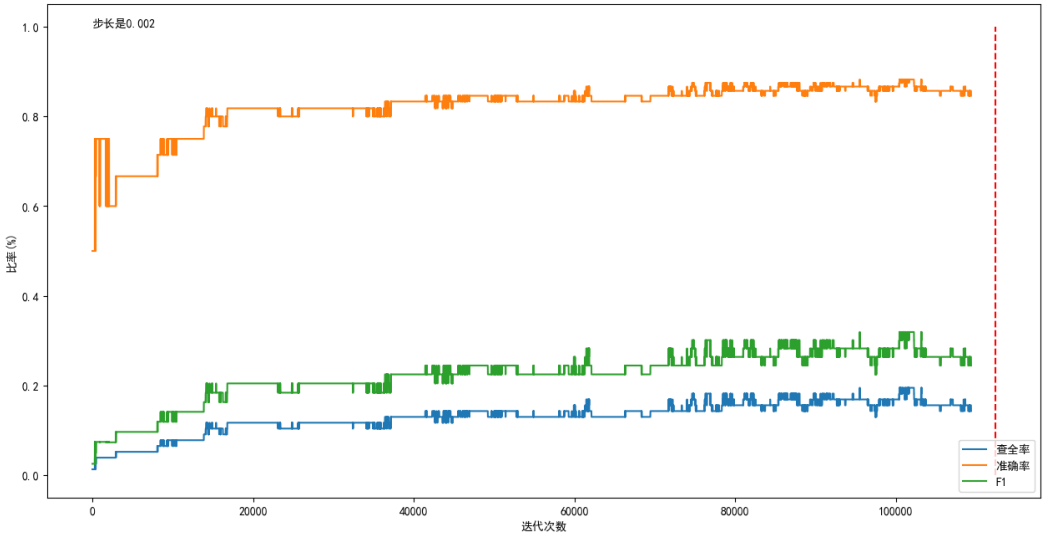
1.批量梯度下降迭代和随机梯度下降迭代方法的对比

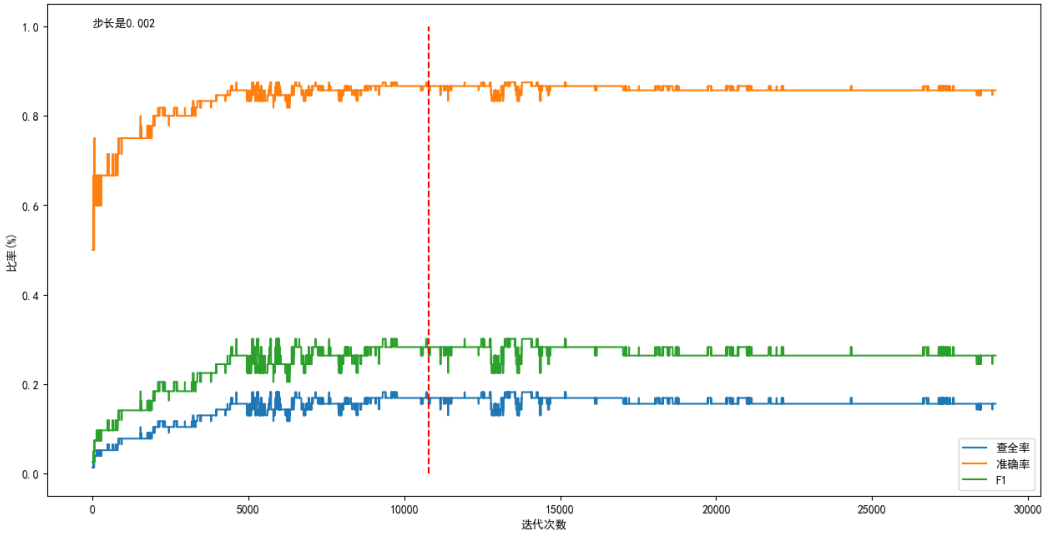
这部分调用的是`random_gradient_auto`和`batch_gradient_auto`这两个函数中的计算方法对y1，也就是接见会见事件的识别结果进行比较

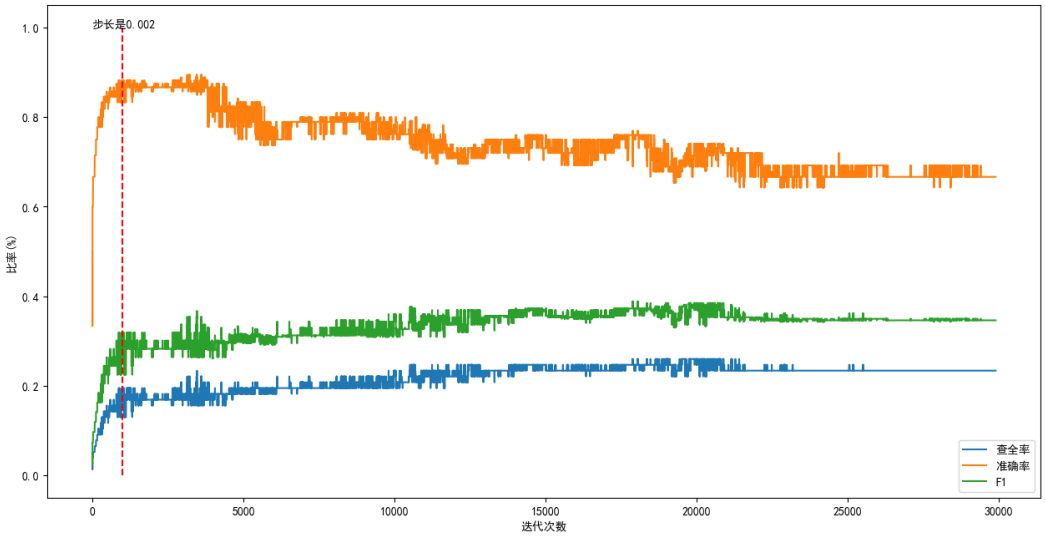
	批量**(15486)**	随机**(10)**	随机**(100)**	随机**(1000)**
迭代次数	6000	120000	30000	30000
步长	0.05*random(0,1)	0.05*random(0,1)	0.05*random(0,1)	0.05*random(0,1)
时间	55120ms	176853ms	41902ms	67830毫秒
每次迭代时间	9.19ms	1.48ms	1.40ms	2.26ms
F1最终收敛值	0.358	0.272	0.263	0.350
查准率最终收敛值	0.655	0.832	0.857	0.667
查全率最终收敛值	0.247	0.143	0.156	0.234
F1最好表现	0.358	0.321	0.301	0.358

运行效果图如下，红色线条标注的是过拟合点的位置，下面四幅图分别按顺序对应了表格中的四次实验


步长是0. 002
比率(%)
迭代次数
查全率
准确率
F1


步长是0. 002
比率(%)
迭代次数
查全率
准确率
F1

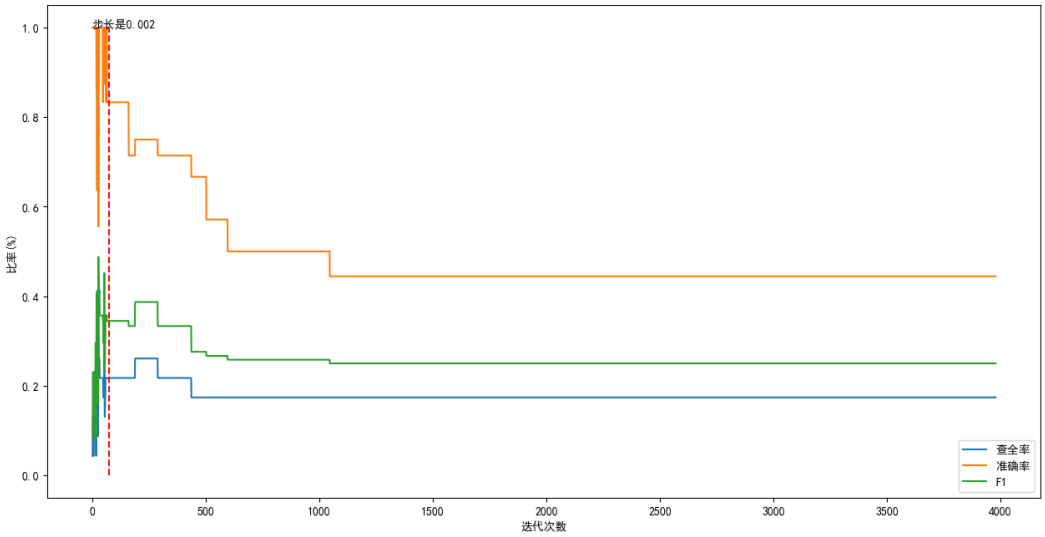

步长是0. 002
比率(%)
迭代次数
查全率
准确率
F1



通过对上面的结果进行观察，我们可以发现以下几点结论：

1. 使用随机迭代的方式可以有效的降低每次迭代所需的时间，这是由于每次的计算量得以减少所导致的，但是在GPU能够加速并行运算的前提下，且计算量没有达到需要GPU分割进行计算的时候，批量迭代的优势是很明显的，并没有在每次迭代的时候运算速度比随机迭代长很多，甚至基本是在同一个数量级上的
2. 随机批量迭代的每批量如果太小，则需要迭代的次数也会需要很多次，并没有快速的达到和批量的同等水平，而如果取到合适的大小，比如1000个样本一批，就比一次迭代15000个样本要快很多，而且下迭代次数5倍的前提下，也可以很快到达过拟合点
3. 过拟合点的位置在测试集上并不是全局F1预测最优的点，这可能是由于测试集的样本量太少，导致了独立同分布的条件的缺失，从而使得在验证集上的loss最低点的时候并不是F1最高，
4. 我的模型在准确率上很高，但是在查全率上表现不好
5. 模型的loss最低点对应了查准率的最高点，这可能说明查准率会对loss函数的数值产生更大的影响，因此两者的关联十分紧密，而随着过了过拟合点后，查准率开始下降，但是查全率却在继续上升，且F1也随着上升
6. 随机采样会使数据的抖动看起来更加明显，而批量则不会，整个变化是比较稳定的

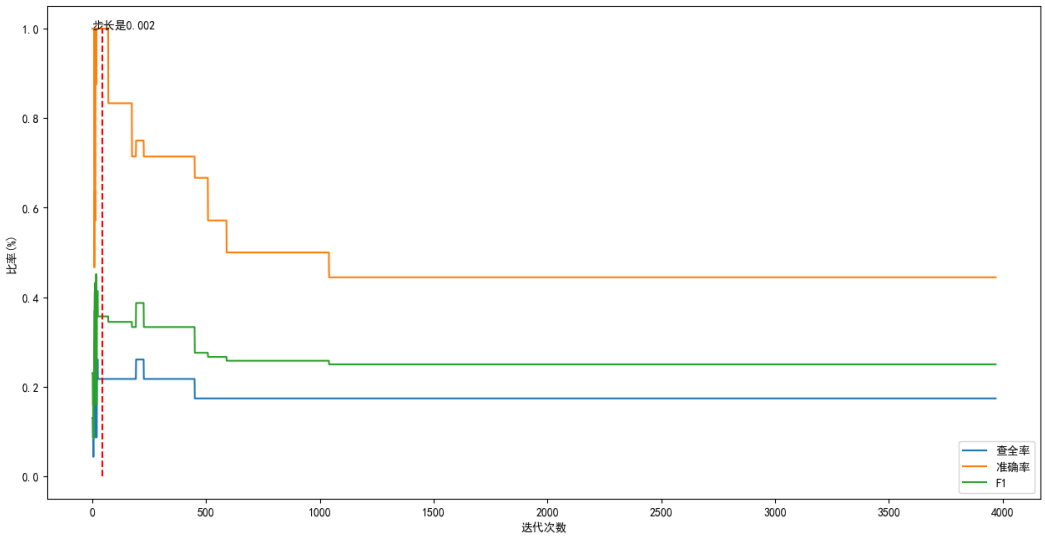
下面我们对第二个事件使用`batch_gradient_auto`进行计算，查看F1的训练效果，如下图



在识别这个事件的时候，我们发现：三个率的最高点都是在过拟合点附近出现的，这与我们的理论符合的非常好，说明事件二在训练集验证集和测试集中都基本符合独立同分布的条件，因此我们的效果才会和理论符合得非常好，同时，最大的F1点也高达50%，精确值甚至到了100%，而且下向后迭代比率出现了整体上的下降，是过拟合现象的映证。

2.手动求导和自动求导的对比

这部分调用的是`batch_gradient_auto`和`batch_gradient_hand`这两个函数中的计算方法对y2，也就是检查事件的识别结果进行比较：



我们可以看到，手动求导的图像和上面的自动求导的图像是一致的（有些许的不同是由于我的步长是倍率乘一个0-1之间的随机步长导致的，但可以看出整体趋势是完全一致的），这说明两个计算途径确实是相同的，这为日后我们使用自动求导奠定了坚实的理论实践基础。

以上的结果是我在取前5000个高频形容词和动词的前提下的效果，已经很不错了，如果我们再对词典作进一步的优化，相信效果一定可以更好

六、参考文献

1. 动手学深度学习 Release 2.0.0 Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola
2. 辛欣老师的2023届知识工程PPT1-2讲

七、鸣谢

讲授理论基础的辛欣老师

讲解环境的搭建和自动求导的语法的助教们

CSDN上在我遇到torch相关语法问题时给出解决方案的人们

回答我问题的同学：范力文、张子翔、马国建

安装GPU时遇到帮我解决问题的同学：马国建

如在复现实验过程中遇到任何问题，请及时联系本人

email：jieyingqi814@qq.com