lab1 实验报告

PB22111695 蔡孟辛

1 实验流程

下载了anaconda，配置环境，学习了python的很多函数...

1.1 PART 1

1.1.1 数据预处理

`dataset.to_pandas()` :将Pytorch的dataset对象转换为Pandas的DataFrame对象。

必要的预处理操作：对`Run_time`取log（np.log）

数据集的划分：`dataset.train_test_split()`

1.1.2 定义模型

`__init__()`：

`Parameter()`：自定义类，用于封装模型参数。
`np.random.randn(in_features, out_features)`生成一个形状为`(in_features, out_features)`的随机权重矩阵，元素服从标准正态分布。
`np.zeros(out_features)`生成一个形状为`(out_features,)`的偏置向量，元素全为零。

`predict()`：实现了模型的前向传播（预测）。
`np.dot(x, self.weight.data)`矩阵的点积。

1.1.3 定义MSELoss

均方误差损失：

$$Loss = \frac{1}{|D_{train}|}\sum(y_{pred} - y_{true})^2$$

backward中计算损失相对于模型参数的梯度（分别对loss求导得到）：

$$\nabla weight = \frac{1}{|D_{train}|}x^T * (y_{pred} - y_{true})$$

$$\nabla bias = \frac{2}{|D_{train}|}\sum(y_{pred} - y_{true})$$

1.1.4 调参

`pbar.set_description()`：显示train进度

`list.append()`：在数组最后加入新数据

1.1.5 Train

详细过程略（之前调的没有意识到要截图），写在后面

1.1.6 评估性能

$$relative\_error = \frac{|\mu - \mu_{true}|}{\mu_{true}}$$

1.2 Part 2

感觉跟part1基本一致，但在计算loss使用了二元交叉熵BEC，合并了weight和bias，部分公式略不同。

2 loss曲线与调试超参数的过程

2.1 Part1

之前怎么调参Relative error都大于0.1，后来查看了issue板块6，将

```
np.abs(pred-target).mean()/target.mean()
```

修改为：

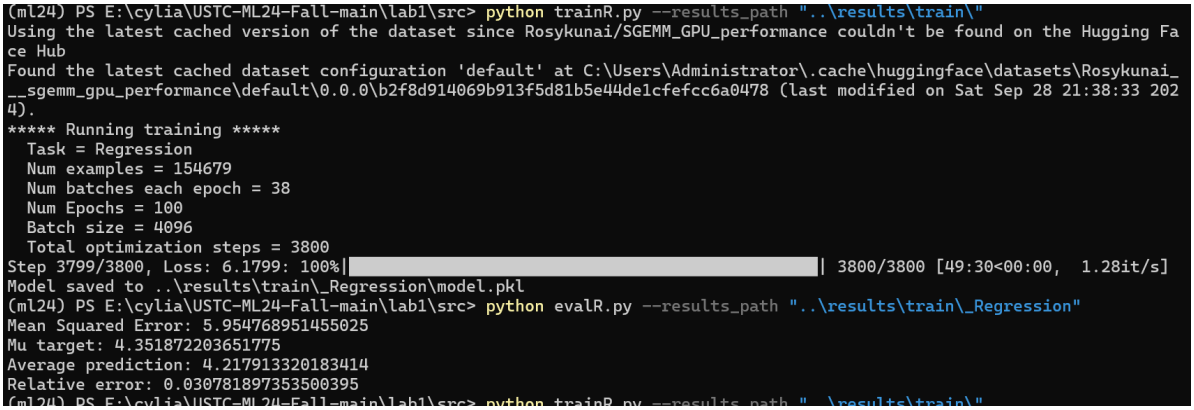
```
np.abs(pred.mean()-target.mean())/target.mean()
```

Relative error结果一下就变好了：)

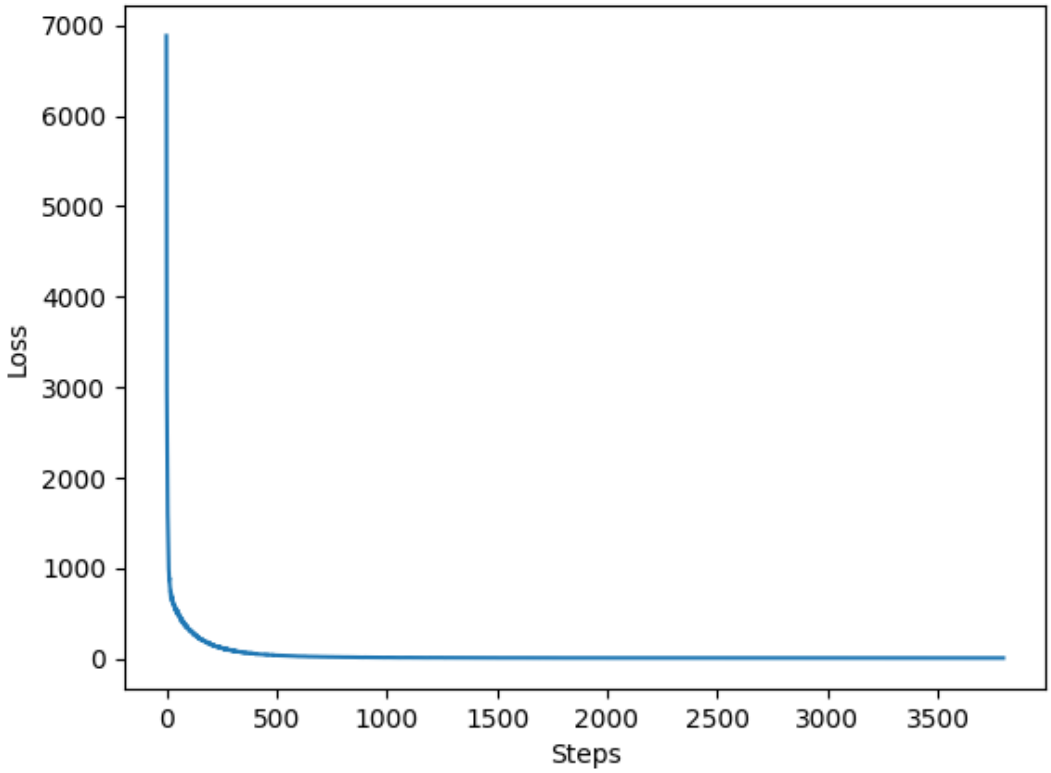
一直调参，直到lr为6e-6时达到临界：

```
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python trainR.py --results_path
"..\results\train\"
Using the latest cached version of the dataset since
Rosykunai/SGEMM_GPU_performance couldn't be found on the Hugging Face Hub
Found the latest cached dataset configuration 'default' at
C:\Users\Administrator\.cache\huggingface\datasets\Rosykunai___sgemm_gpu_perform
ance\default\0.0.0\b2f8d914069b913f5d81b5e44de1cfefcc6a0478 (last modified on Sat
Sep 28 21:38:33 2024).
***** Running training *****
  Task = Regression
  Num examples = 154679
  Num batches each epoch = 38
  Num Epochs = 100
```

```
  Batch size = 4096
  Total optimization steps = 3800
Step 3799/3800, Loss: 6.1799: 100%|
████████████████████████████████████████████████████| 3800/3800 [49:30<00:00,
1.28it/s]
Model saved to ..\results\train\_Regression\model.pkl
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python evalR.py --results_path
"..\results\train\_Regression"
Mean Squared Error: 5.954768951455025
Mu target: 4.351872203651775
Average prediction: 4.217913320183414
Relative error: 0.030781897353500395
```



当时的Loss曲线为（很美丽啊，虽然结果不行）：



从loss曲线看，loss在一开始就急剧下降，然后趋于一个固定的值。

（我最开始的loss超级戏剧，可惜被刷新了）

其实调参的时候也调整了batch_size，但是感觉对结果影响不大，但程序跑的时间更久了。。

对lr_decay和decay_every的调整也对结果影响很小。

无论怎么调整，relative error的临界就是0.03了。。反复阅读实验文档和issue后，发现在对数据进行预处理的时候，可以使用标准化和归一化（但是个人尝试了标准化，后的效果不是很好，在issue上看到助教提到可以对数据（除了run_time取log以外的其他数据）进行归一化处理，于是集中火力对归一化后的参数进行了调参）。

标准化：

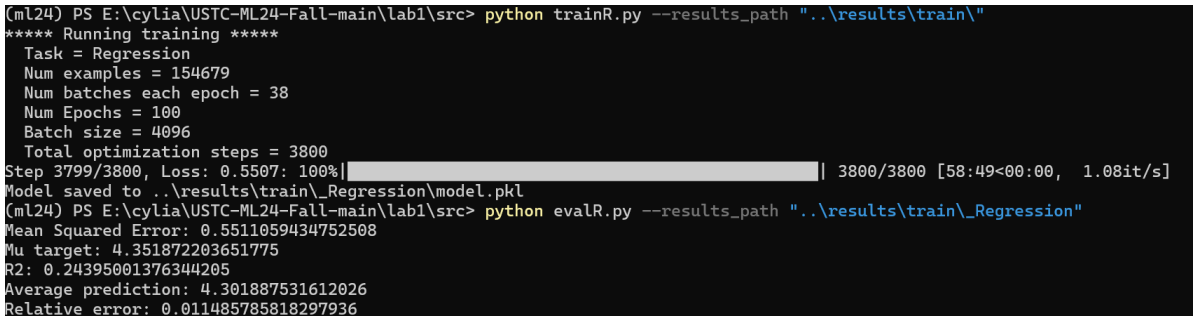
$$z = \frac{x - \mu}{\sigma}$$

归一化：

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

处理数据时进行归一化，调参为1e-2：

```
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python trainR.py --results_path
"..\results\train\"
***** Running training *****
  Task = Regression
  Num examples = 154679
  Num batches each epoch = 38
  Num Epochs = 100
  Batch size = 4096
  Total optimization steps = 3800
Step 3799/3800, Loss: 0.5507: 100%|
████████████████████████████████████████████████████| 3800/3800 [58:49<00:00,
1.08it/s]
Model saved to ..\results\train\_Regression\model.pkl
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python evalR.py --results_path
"..\results\train\_Regression"
Mean Squared Error: 0.5511059434752508
Mu target: 4.351872203651775
R2: 0.24395001376344205
Average prediction: 4.301887531612026
Relative error: 0.011485785818297936
```

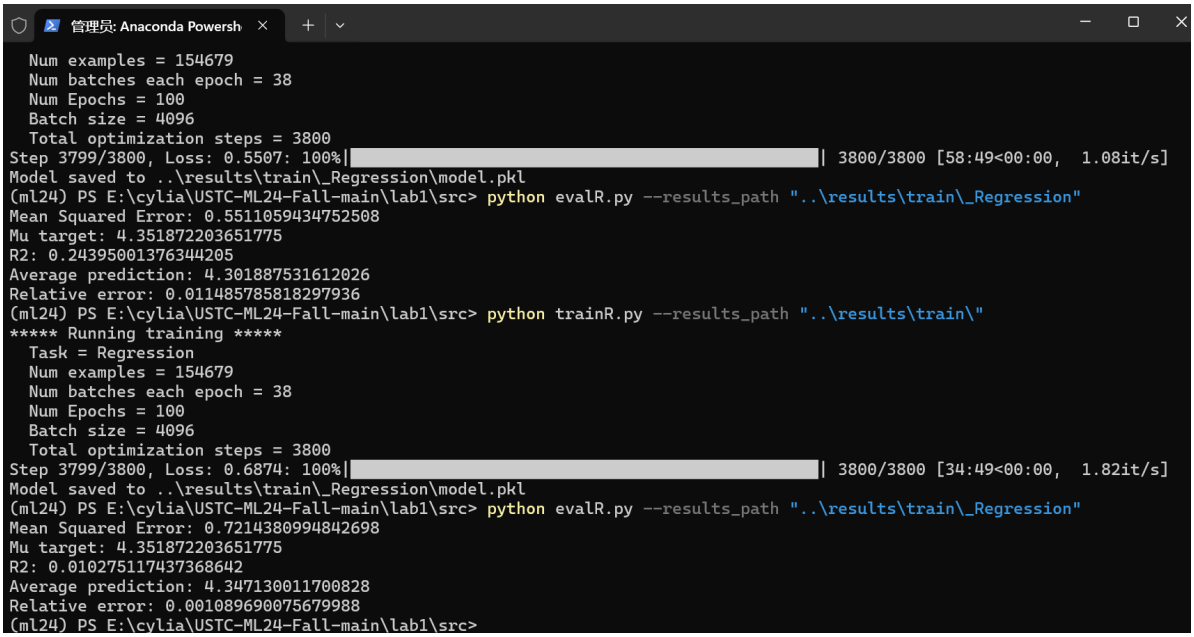


（loss图像被刷新了QAQ）

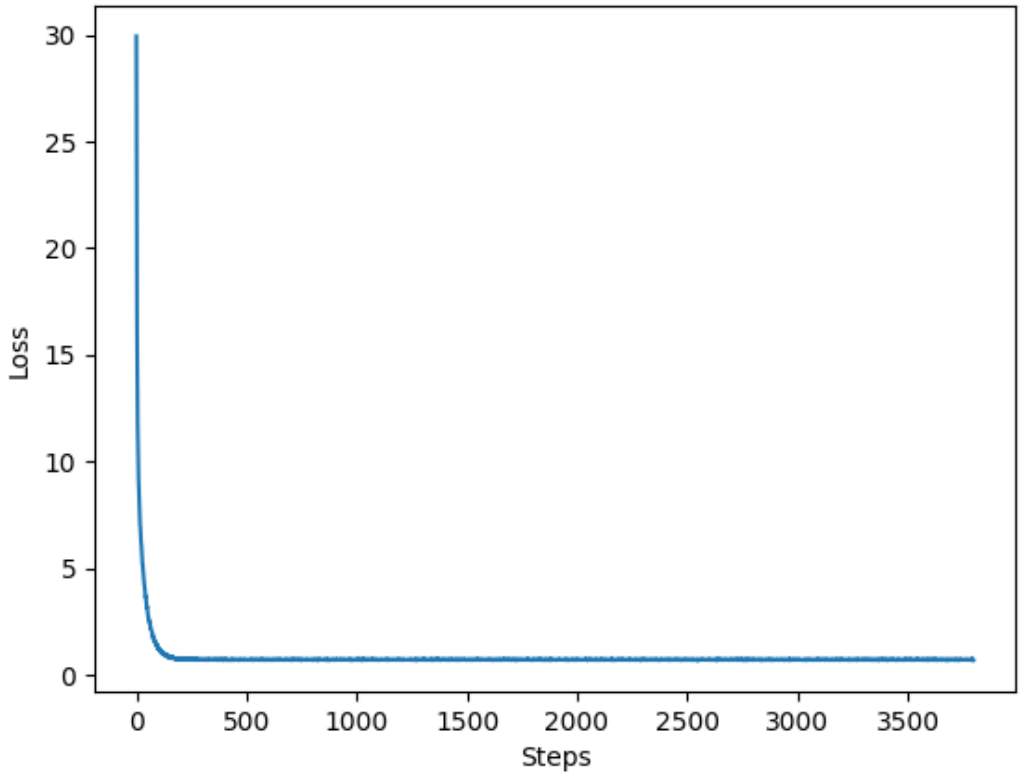
对grad_weight做L2正则化处理（$\lambda = 10$）：

```
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python trainR.py --results_path
"..\results\train\"
```

```
***** Running training *****
  Task = Regression
  Num examples = 154679
  Num batches each epoch = 38
  Num Epochs = 100
  Batch size = 4096
  Total optimization steps = 3800
Step 3799/3800, Loss: 0.6874: 100%|
██████████████████████████████████████████████████| 3800/3800 [34:49<00:00,
1.82it/s]
Model saved to ..\results\train\_Regression\model.pkl
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python evalR.py --results_path
"..\results\train\_Regression"
Mean Squared Error: 0.7214380994842698
Mu target: 4.351872203651775
R2: 0.010275117437368642
Average prediction: 4.347130011700828
Relative error: 0.001089690075679988
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src>
```



Loss图像，美丽：



2.2 Part2

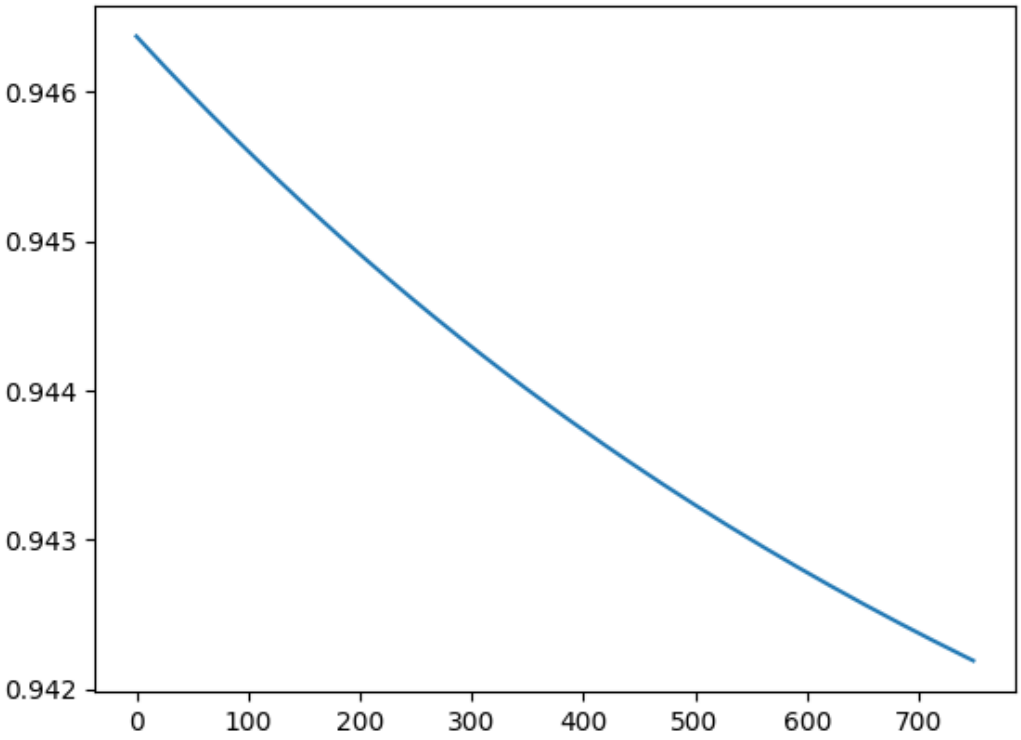
写的时候就进行了归一化。step改为750。

lr=2e-6（初始）

```
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python trainC.py --results_path "..\results\train\"
***** Running training *****
  Task = Classification
  Num examples = 154679
  Total optimization steps = 750
Step 749/750, Loss: 0.9463: 100%|| 750/750 [00:09<00:00, 76.04it/s]
Model saved to ..\results\train\_Classification\model.pkl
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python evalC.py --results_path "..\results\train\_Classification"
Accuracy: 0.5553659167313163
```

lr=1e-4

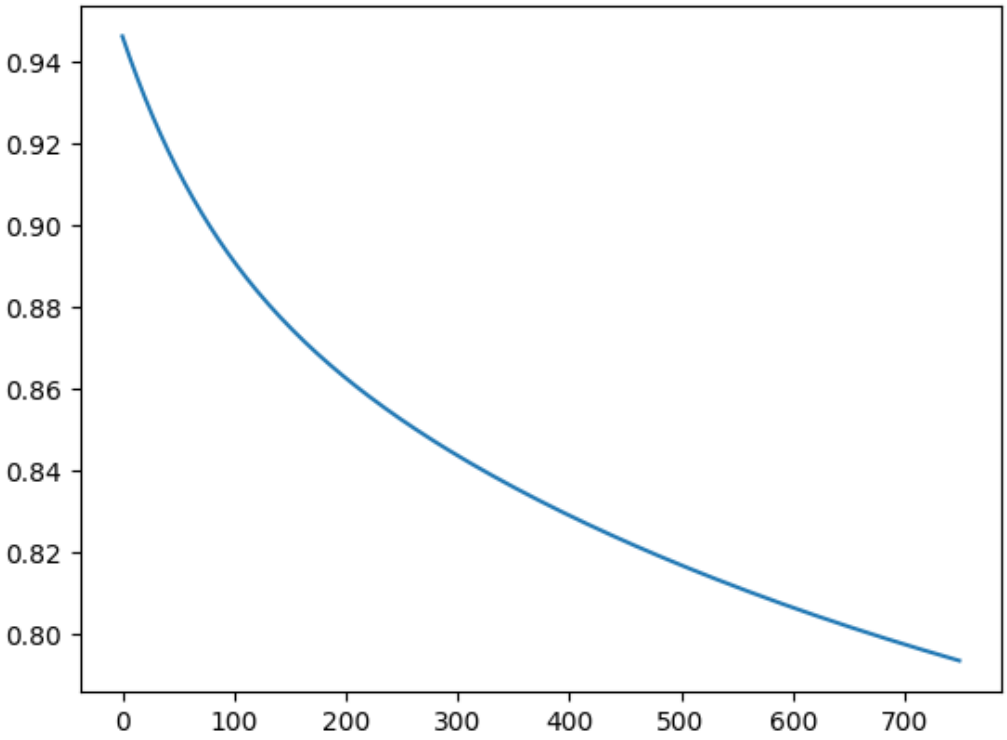
```
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python trainC.py --results_path "..\results\train\"
***** Running training *****
  Task = Classification
  Num examples = 154679
  Total optimization steps = 750
Step 749/750, Loss: 0.9422: 100%|| 750/750 [00:10<00:00, 72.41it/s]
Model saved to ..\results\train\_Classification\model.pkl
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python evalC.py --results_path "..\results\train\_Classification"
Accuracy: 0.5549521592966123
```



loss下降太慢了。

1e-2

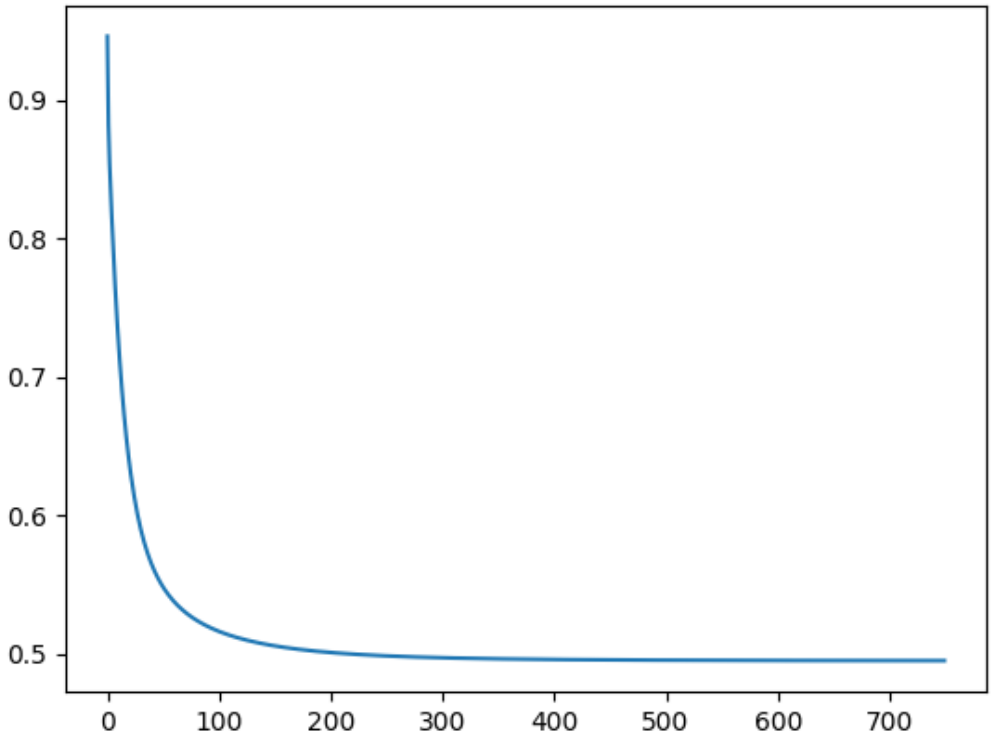
```
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python trainC.py --results_path "..\results\train\"
***** Running training *****
  Task = Classification
  Num examples = 154679
  Total optimization steps = 750
Step 749/750, Loss: 0.7935: 100%|████████████████████████████████████████| 750/750 [00:12<00:00, 58.70it/s]
Model saved to ..\results\train\_Classification\model.pkl
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python evalC.py --results_path "..\results\train\_Classification"
Accuracy: 0.5577450219808637
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src>
```



loss下降依然很慢。

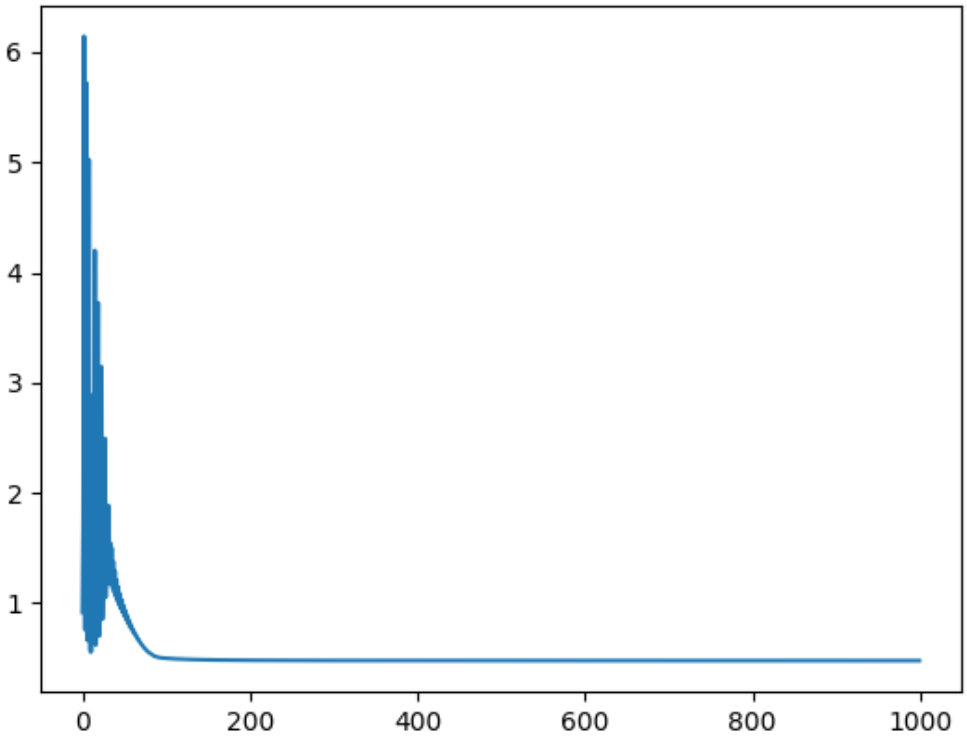
lr=1

```
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python trainC.py --results_path "..\results\train\"
***** Running training *****
  Task = Classification
  Num examples = 154679
  Total optimization steps = 750
Step 749/750, Loss: 0.4954: 100%|████████████████████████████████████████| 750/750 [00:09<00:00, 77.10it/s]
Model saved to ..\results\train\_Classification\model.pkl
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python evalC.py --results_path "..\results\train\_Classification"
Accuracy: 0.8198603568657874
```



lr=5 step=1000

```
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python trainC.py --results_path "..\results\train\"
***** Running training *****
  Task = Classification
  Num examples = 154679
  Total optimization steps = 1000
Step 999/1000, Loss: 0.4783: 100%|████████████████████████████████████████| 1000/1000 [00:13<00:00, 76.42it/s]
Model saved to ..\results\train\_Classification\model.pkl
(ml24) PS E:\cylia\USTC-ML24-Fall-main\lab1\src> python evalC.py --results_path "..\results\train\_Classification"
Accuracy: 0.8409619860356866
```



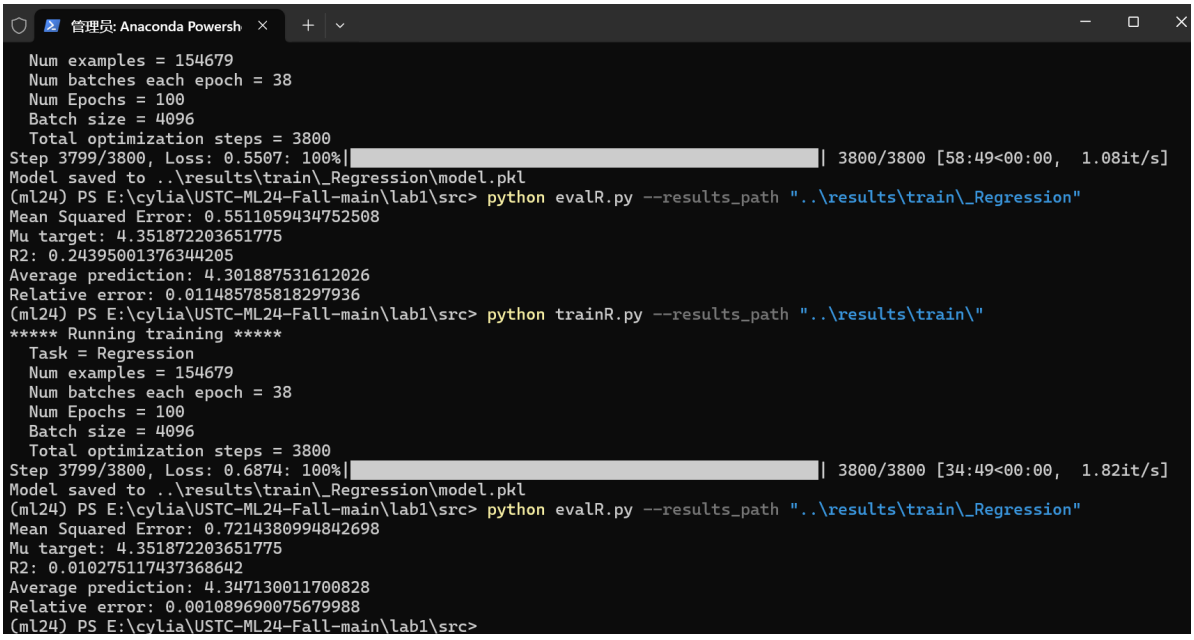
可以看到loss一开始的波动很大，但最后还是趋于平稳的：）

L2正则化调了很多λ,但无法提高accuracy...

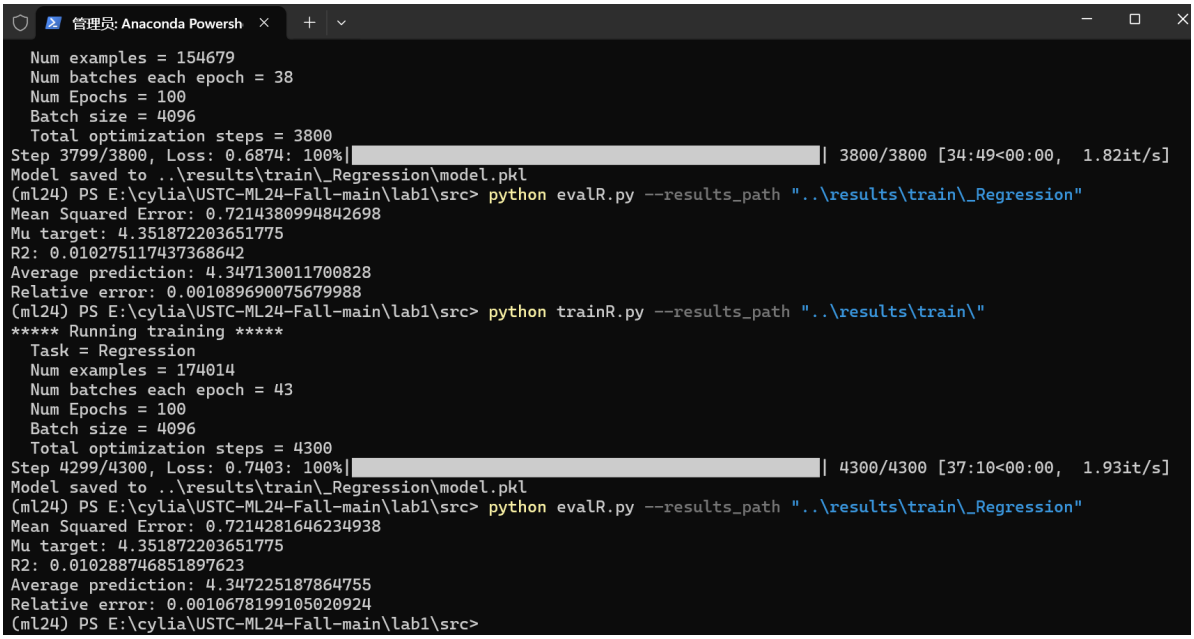
3 在自己划分的数据集上的最好结果

- part1：

8:1:1 relative error= 0.001089690075679988

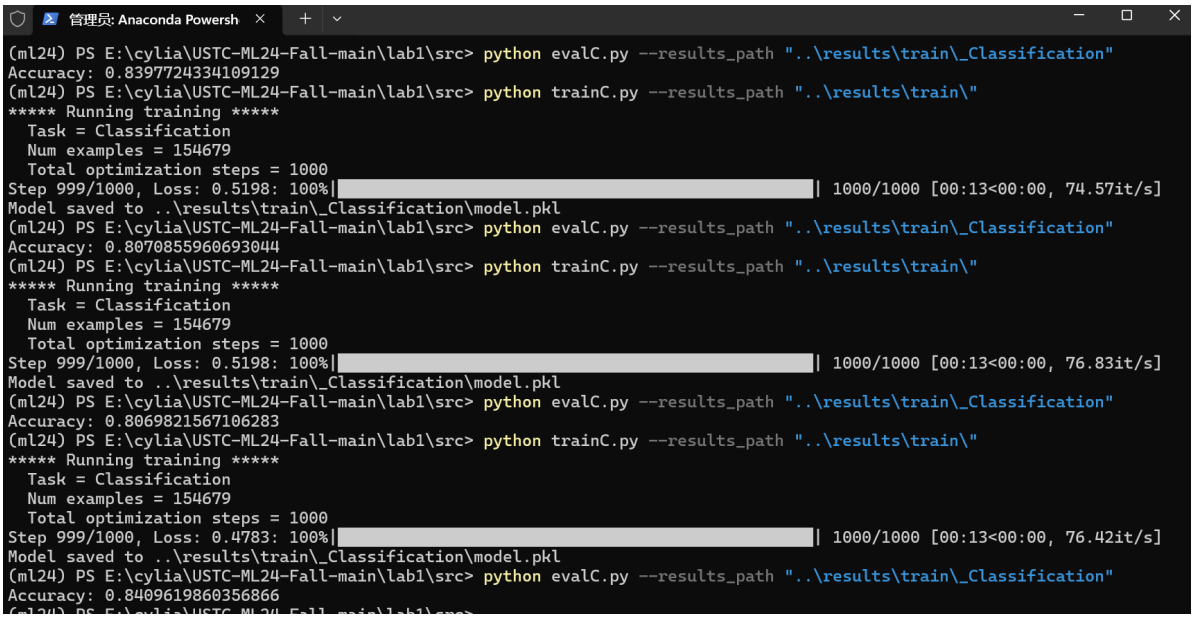


合并train和val：Relative error: 0.0010678199105020924

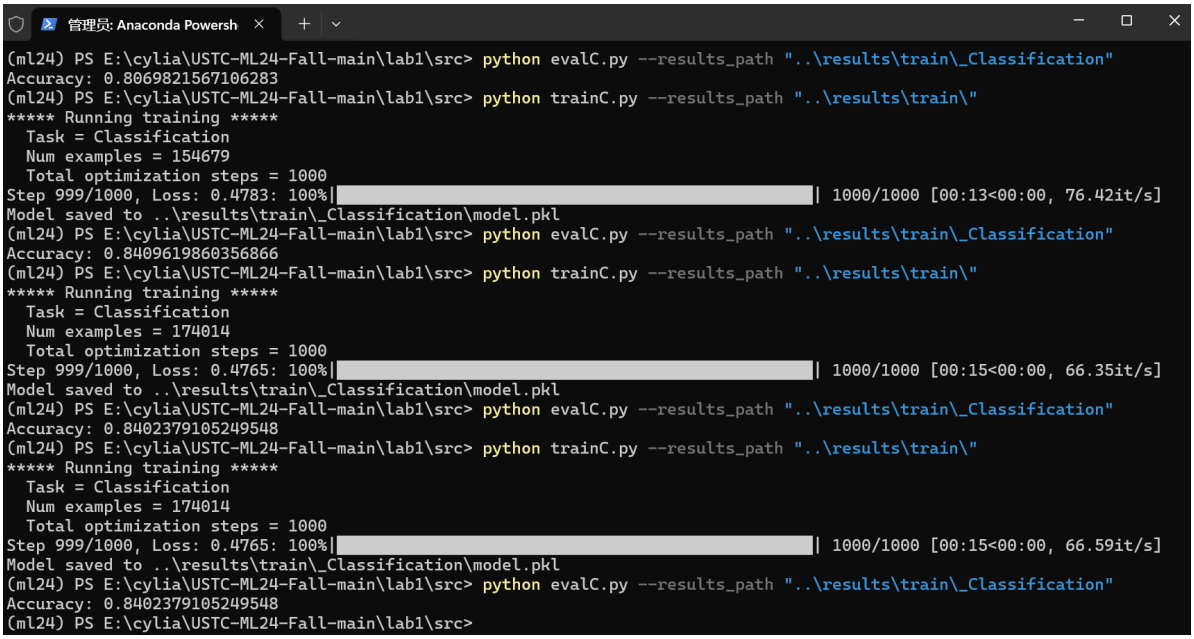


- part2：

8:1:1 Accuracy: 0.8409619860356866



合并train和val：Accuracy: 0.8402379105249548



4 回答问题

- 在对数据进行预处理时,你做了什么额外的操作,为什么?

  对除了Run_time以外的列进行了归一化处理，将数据缩放到特定的范围 [0, 1]内，这样可以提高梯度下降算法的收敛速度，减少训练时间；同时使模型的训练更稳定，减少数值计算的误差，提高模型的性能；同时降低了某些极大极小值对模型训练的影响。

- 在处理分类问题时,使用交叉熵损失的优势是什么?

  交叉熵损失函数能够很好地处理模型输出的概率分布；同时交叉熵损失函数对错误分类的惩罚较大，尤其是当模型对错误类别的预测概率较高时，有助于模型更快地纠正错误；交叉熵损失函数的梯度信息丰富，为模型提供更有效的梯度更新，可以加速模型的训练过程。

- 本次实验中的回归问题参数并不好调,在实验过程中你总结了哪些调参经验?

  学习率对结果的印象最大，是否进行归一化的lr也不同，归一化后lr通常会大很多，lr大于某个值后loss会急剧变大（bug）；

  batch_size影响训练的step，一般来说step越大，loss越能找到平稳值，结果越好；

- 你是否做了正则化,效果如何?为什么?

  作了正则化，效果非常好！！！

  正则化可以防止模型过拟合。（计算loss的权重梯度是使用了L2正则化，惩罚模型中较大的权重，防止模型过拟合训练数据。较大的权重可能会导致模型对训练数据的噪声过于敏感，从而降低模型在测试数据上的泛化能力。）

5 反馈

引言：好难！！！！好难！！！！！！！好难！！！！！！

时间：感觉莫约2周！！！！我国庆都没出去玩QAQ