第四章　利用SNP预测表型

作业布置 & 参考资料

- 作业：
  - 1. 实践作业：完成SNP_MLP 和 SNP_CNN模型训练，可视化训练过程的损失曲线与Pearson相关系数变化。选择至少一个其他表型（如TKW），报告测试集的相关系数并分析模型表现。
  - 2. 思考题：比较MLP和CNN的结构，参数量和训练效果；比较同一个模型不同学习率对训练效果的影响？SNP_CNN中就是卷积+MLP，前面的卷积有什么作用?
- 参考资料：
  - 1. MLP: https://blog.csdn.net/u011734144/article/details/80924207
  - 2. CNN: https://blog.csdn.net/IronmanJay/article/details/128689946
  - 3. 可视化库：Seaborn/Matplotlib（用于绘制loss曲线，相关系数曲线）

目标

- 使用 小麦snp数据集 构建 表型预测模型
  - 1. 使用lightningModule包装模型 (给MLP和CNN模型的具体实现，然后大家用LightningModel包装起来)
  - 2. 使用pearson相关系数作为评估指标
  - 3. 对训练过程的评估指标进行可视化
  - 4. 对结果进行分析总结(讲啥都行)
  - 5. 对于有能力且有兴趣尝试其他模型的同学，可以自己设计模型训练。

数据集基本介绍

- df_snp

	snp_0	snp_1	snp_2	snp_3	snp_4	snp_5	snp_6	snp_7	...	snp_33705	snp_33706	snp_33707	snp_33708
0	0	1	1	0	1	0	1	0	...	1	1	1	0
1	1	0	0	1	0	1	1	0	...	1	1	1	1
2	0	1	1	0	1	0	1	0	...	0	0	1	0
3	1	0	0	1	0	1	1	0	...	1	0	1	0
4	1	0	0	1	0	1	0	1	...	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1995	0	1	1	0	1	1	0	1	...	1	0	0	0
1996	0	1	0	1	1	1	1	0	...	0	1	1	0
1997	0	1	1	0	1	0	0	1	...	1	0	1	1
1998	1	0	1	0	1	0	0	1	...	1	0	1	1
1999	0	1	0	1	1	0	1	0	...	1	0	1	0

每一行是一个小麦样本(2000个)

每一列是一个snp位点(33709个)

df_snp中的每一个值是 0或1

0代表这个snp位点和参考基因组相同
1代表这个snp位点和参考基因组不同

数据集基本介绍

- df_pheno

	TKW	TW	GL	GW	GH	GP	SDS	PHT
0	1.620228	1.942195	0.592252	0.591566	-0.482315	-1.627567	-2.646675	1.355421
1	-0.749077	-0.199763	-0.000361	-0.945802	1.462503	0.128817	1.072801	0.824866
2	-1.733724	-2.681714	0.695316	-1.765731	0.879058	1.053230	0.924022	-0.236244
3	-0.010593	1.058212	-0.051892	-0.382100	1.462503	-1.072920	0.924022	0.196060
4	0.266339	0.752218	0.798379	-1.048293	2.045949	0.036376	0.328906	-0.609598
...	...	...	...	...	...	...	...	...
1995	1.158675	-0.743752	0.051171	2.180180	-0.871278	0.128817	-0.266210	1.630524
1996	-0.287524	-1.661734	0.411892	0.130356	1.073540	-1.072920	-0.563768	0.196060
1997	0.235569	-0.335760	-0.051892	1.104022	0.879058	-0.425831	-0.414989	0.294311
1998	-0.764463	1.806197	-2.448110	0.335338	1.268021	-1.812450	0.031348	-0.609598
1999	-1.610643	1.398205	-2.370813	-0.638328	-0.287833	-1.720009	-0.563768	0.235360

2000个小麦样本的 8种表型数据

已经经过 标准化

trait	中文含义
GH	谷粒硬度
GL	谷粒长度
GP	谷物蛋白
GW	谷粒宽度
PHT	植物高度
SDS	十二烷基硫酸钠-沉降
TKW	千粒重
TW	测试重量

关于数据集的不同文件格式

```
./
├── [   56]  csv
│   ├── [ 284K]  df_pheno.csv
│   └── [ 129M]  df_snp.csv
└── [   64]  parquet
    ├── [  30K]  df_pheno.parquet
    └── [  27M]  df_snp.parquet
```

csv 和 parquet 文件中保存的都是相同的内容

区别在于:
parquet 存储消耗的空间更小，读取的速度更快

如果数据集比较小，那么用简单的csv就很合适，
人很容易就能看懂

如果数据集很大，那就用parquet，否则读取很慢，
在处理dataset对象的时候，要等很久

大家可以
df_snp_parquet = pd.read_parquet("./df_snp.parquet")
df_snp_csv = pd.read_csv("./df_snp.csv")
简单比较一下 不同文件格式的读取速度

```
1  df_snp = pd.read_parquet("./df_snp.parquet")
✓ 2.4s
```

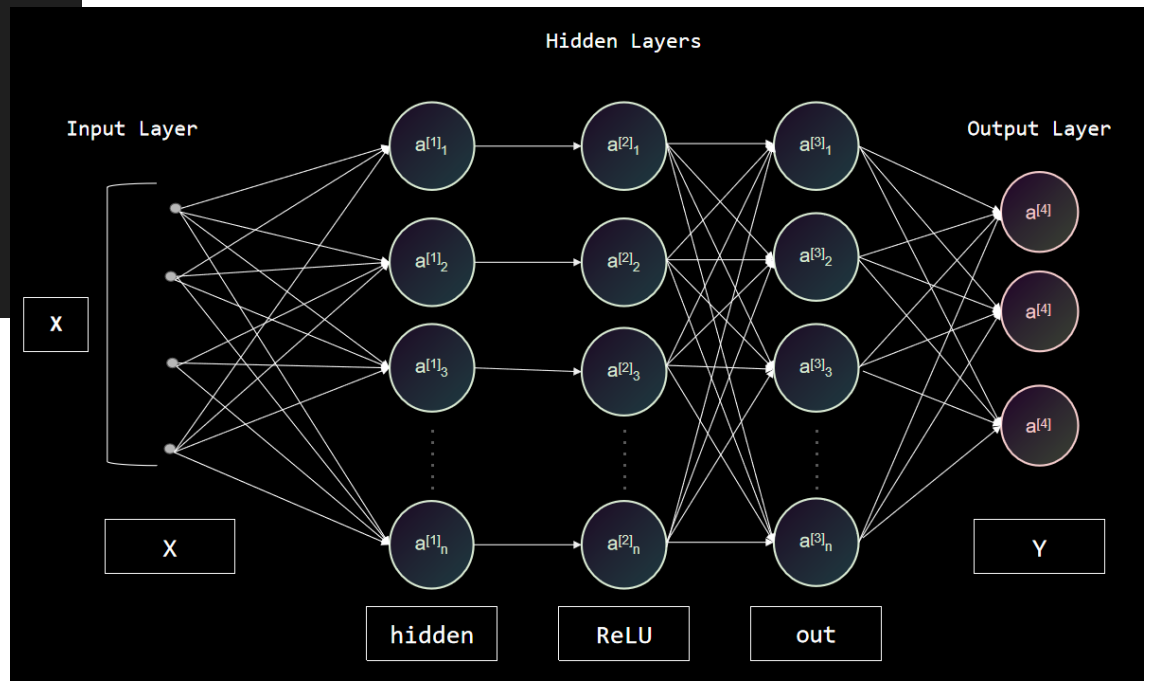
```
1  df_snp = pd.read_csv("./df_snp.csv")
✓ 22.6s
```

模型结构 SNP_MLP 和 SNP_CNN

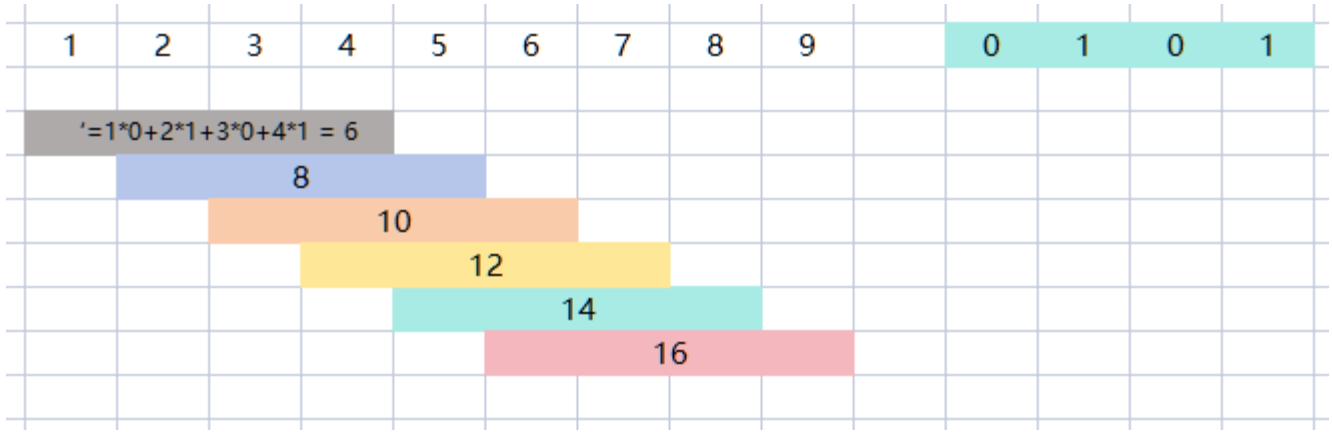
- SNP_MLP 和 SNP_CNN
  - 三个参数
    - snp_feature_count：输入X一共有多少个特征
    - out_feature_count：输出Y一共有多少个特征
    - **dropout_rate**：随机将神经元置为0的比例，用于避免过拟合
  - 模型前向传播的输入X维度
    - 【batch_size, snp_feature_count】
  - 模型的输出Y的维度
    - 【batch_size】 只预测一个表型，有能力的同学可以自己改成同时预测多个表型

MLP

```
LitModel(
  (model): SNP_MLP(
    (mlp): Sequential(
      (0): Linear(in_features=33709, out_features=1024, bias=True)
      (1): LeakyReLU(negative_slope=0.01)
      (2): Dropout(p=0.3, inplace=False)
      (3): Linear(in_features=1024, out_features=256, bias=True)
      (4): LeakyReLU(negative_slope=0.01)
      (5): Dropout(p=0.3, inplace=False)
      (6): Linear(in_features=256, out_features=1, bias=True)
    )
  )
  (loss_func): MSELoss()
)
```

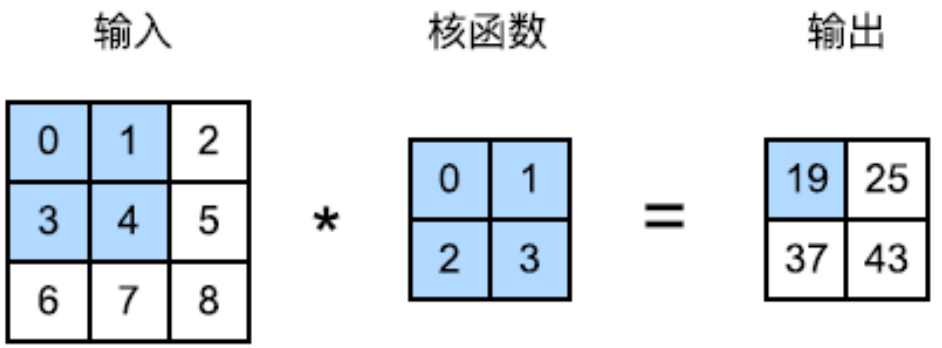


卷积

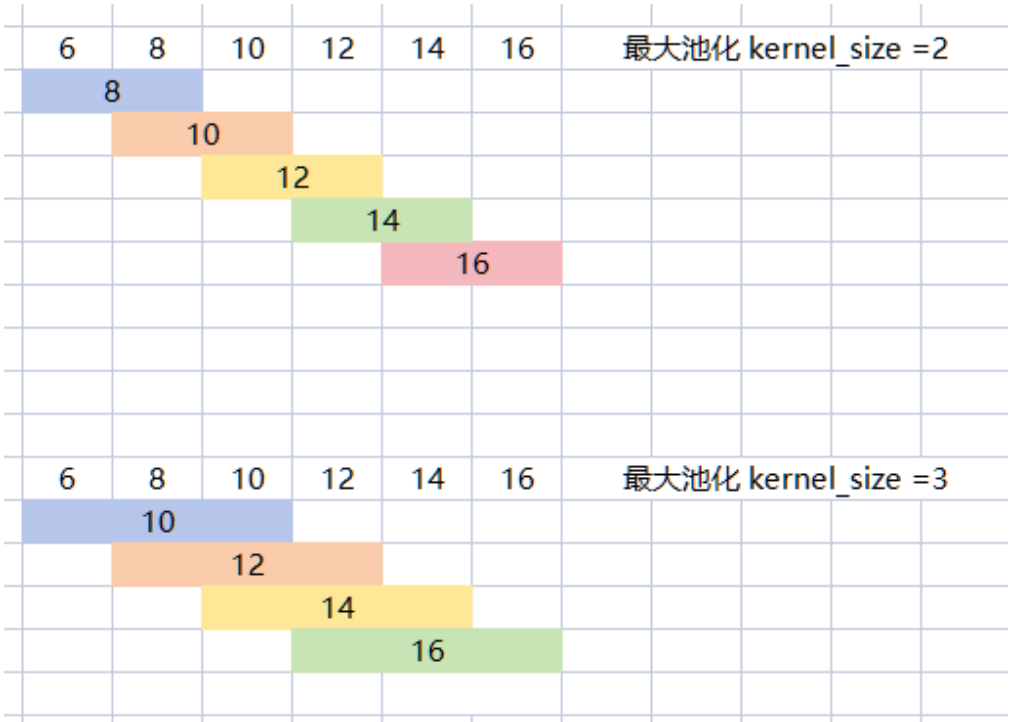

1 2 3 4 5 6 7 8 9
0 1 0 1
'=1*0+2*1+3*0+4*1 = 6
8
10
12
14
16

一维卷积

二维卷积

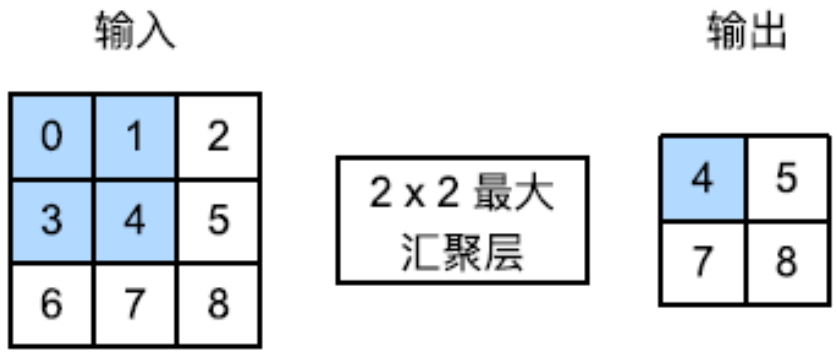

输入
核函数
输出
0 1 2
3 4 5
6 7 8
*
0 1
2 3
=
19 25
37 43

池化



平均值池化?

二维池化



SNP_CNN的实现

```
(model): SNP_CNN(
  (net): Sequential(
    (0): Conv1d(1, 64, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv1d(64, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (4): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv1d(128, 256, kernel_size=(3,), stride=(1,), padding=(1,))
    (7): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (9): Conv1d(256, 1, kernel_size=(1,), stride=(1,), padding=(1,))
    (10): Flatten(start_dim=1, end_dim=-1)
    (11): Linear(in_features=4215, out_features=1024, bias=True)
    (12): Linear_(
      (linear): Sequential(
        (0): Dropout(p=0.3, inplace=False)
        (1): ReLU()
        (2): Linear(in_features=1024, out_features=512, bias=False)
      )
    )
    (13): Linear_(
      (linear): Sequential(
        (0): Dropout(p=0.3, inplace=False)
        (1): ReLU()
        (2): Linear(in_features=512, out_features=1, bias=False)
      )
    )
  )
```

具体步骤

- 1. 对df_snp和df_pheno 分出训练集，验证集和测试集 (2000个样本，按照1500:250:250)
- 2. 自己写好Dataset类，然后定义dataset对象和dataloader对象
- 3. 使用LightningModule将模型SNP_MLP和SNP_CNN训练逻辑包装起来
- 4. 保存好训练过程的评估指标，包括train_loss, valid_loss, train_pearson, valid_pearson
- 5. 保存 valid_pearson 最高的模型
- 6. 对 train_loss, valid_loss, train_pearson, valid_pearson 进行可视化
- 7. 使用最优模型，对测试集的结果进行预测评估

- **鼓励随意发挥**

最终希望看到

- 对于一个表型 TKW(千粒重)，希望看到最优模型的训练集，验证集和测试集的pearson相关系数
- 然后将 训练过程的各种指标进行可视化，看看训练是不是平缓了

```
result_pearson
```

```
{'train': 0.9140078540911868,
 'valid': 0.5831217157731843,
 'test': 0.6264795251753515}
```