

## 说明

### 1 . 运行方法

#### 1 . 1 运行环境架设

- 1 ) 先安装 winpython for python 2.7 的科学计算环境。由于该环境是绿色安装的，所以相当于解压。
- 2 ) 在安装好 winpython 后，进入 winpython 目录下，点击里面的类似命令窗口的图标，进入进入到 `python` 环境。
- 3 ) 输入 `pip install theano`. 安装 `theano` 库（需要有网络）
- 4 ) 输入 `pip install keras`, 安装 `keras` 库（需要有网络）

#### 1 . 2 运行程序

##### 1 ) 训练模型

还是前面说的，进入 winpython 的命令窗口，切换当前目录到我的代码目录下。  
然后输入指令：

[例如训练一个一步预测模型]：

```
python main.py --cmd train --predictStep 1 --modelfile1 model1.h5 --datapath ../data --echo 30
```

这句话意思是

`--cmd train` 表示为训练模型

`--predictStep 1` 表示该模型是用来 1 步预测

`--modelfile1 model1.h5` 表示该模型训练后保存存在 `model.h5` 这个文件下

`--datapath ../data` 表示数据所在的目录

`--echo 30` 表示训练 30 个轮回

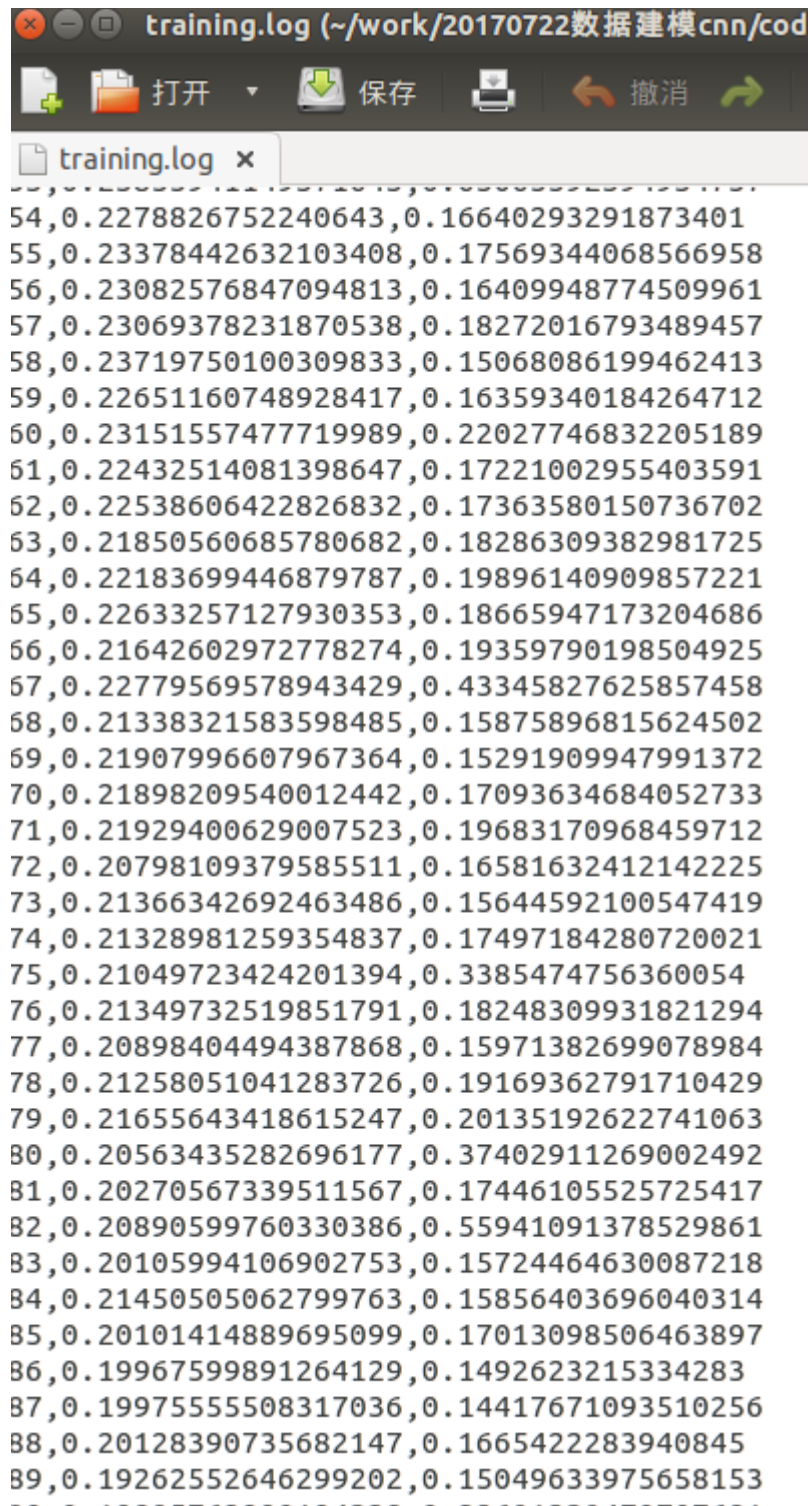
同理，训练一个 2 步预测模型，指令如下

```
python script_sigleFileMultiCol.py --cmd train --predictStep 2 --modelfile2 model2.h5  
--datapath ../data --echo 30
```

训练过程略微耗时，在 `cpu` 上跑，需要十几分钟，在 `gpu` 上跑需要 1 到 2 分钟。在所给的代码中，包含了两个训练好的模型文件，分别是 `model1.h5`, `model2.h5`。

```
jingxia@jingxia: ~/work/20170722数据建模cnn/code
Epoch 49/95
103680/103680 [=====] - 1s - loss: 0.2503 - val_loss: 0.2069
Epoch 50/95
103680/103680 [=====] - 1s - loss: 0.2325 - val_loss: 0.4384
Epoch 51/95
103680/103680 [=====] - 1s - loss: 0.2326 - val_loss: 0.1857
Epoch 52/95
103680/103680 [=====] - 1s - loss: 0.2340 - val_loss: 0.2067
Epoch 53/95
103680/103680 [=====] - 1s - loss: 0.2411 - val_loss: 0.1880
Epoch 54/95
103680/103680 [=====] - 1s - loss: 0.2386 - val_loss: 0.6507
Epoch 55/95
103680/103680 [=====] - 1s - loss: 0.2279 - val_loss: 0.1664
Epoch 56/95
103680/103680 [=====] - 1s - loss: 0.2338 - val_loss: 0.1757
Epoch 57/95
103680/103680 [=====] - 1s - loss: 0.2308 - val_loss: 0.1641
Epoch 58/95
103680/103680 [=====] - 1s - loss: 0.2307 - val_loss: 0.1827
Epoch 59/95
103680/103680 [=====] - 1s - loss: 0.2372 - val_loss: 0.1507
Epoch 60/95
103680/103680 [=====] - 1s - loss: 0.2265 - val_loss: 0.1636
Epoch 61/95
103680/103680 [=====] - 1s - loss: 0.2315 - val_loss: 0.2203
Epoch 62/95
103680/103680 [=====] - 1s - loss: 0.2243 - val_loss: 0.1722
Epoch 63/95
103680/103680 [=====] - 1s - loss: 0.2254 - val_loss: 0.1736
Epoch 64/95
103680/103680 [=====] - 1s - loss: 0.2185 - val_loss: 0.1829
Epoch 65/95
103680/103680 [=====] - 1s - loss: 0.2218 - val_loss: 0.1990
Epoch 66/95
103680/103680 [=====] - 1s - loss: 0.2263 - val_loss: 0.1867
Epoch 67/95
103680/103680 [=====] - 1s - loss: 0.2164 - val_loss: 0.1936
Epoch 68/95
```

训练过程大约如此：



```
54,0.2278826752240643,0.16640293291873401
55,0.23378442632103408,0.17569344068566958
56,0.23082576847094813,0.16409948774509961
57,0.23069378231870538,0.18272016793489457
58,0.23719750100309833,0.15068086199462413
59,0.22651160748928417,0.16359340184264712
60,0.23151557477719989,0.22027746832205189
61,0.22432514081398647,0.17221002955403591
62,0.22538606422826832,0.17363580150736702
63,0.21850560685780682,0.18286309382981725
64,0.22183699446879787,0.19896140909857221
65,0.22633257127930353,0.18665947173204686
66,0.21642602972778274,0.19359790198504925
67,0.22779569578943429,0.43345827625857458
68,0.21338321583598485,0.15875896815624502
69,0.21907996607967364,0.15291909947991372
70,0.21898209540012442,0.17093634684052733
71,0.21929400629007523,0.19683170968459712
72,0.20798109379585511,0.16581632412142225
73,0.21366342692463486,0.15644592100547419
74,0.21328981259354837,0.17497184280720021
75,0.21049723424201394,0.3385474756360054
76,0.21349732519851791,0.18248309931821294
77,0.20898404494387868,0.15971382699078984
78,0.21258051041283726,0.19169362791710429
79,0.21655643418615247,0.20135192622741063
80,0.20563435282696177,0.37402911269002492
81,0.20270567339511567,0.17446105525725417
82,0.20890599760330386,0.55941091378529861
83,0.20105994106902753,0.15724464630087218
84,0.21450505062799763,0.15856403696040314
85,0.20101414889695099,0.17013098506463897
86,0.19967599891264129,0.1492623215334283
87,0.19975555508317036,0.14417671093510256
88,0.20128390735682147,0.1665422283940845
89,0.19262552646299202,0.15049633975658153
```

并同时把这个过程也保存在同一目录下的 training.log 文件里，可以用记事本打开看：

训练的模型，1 步模型 mse 大概在 0.15 左右，2 步模型 mse 大概在 0.3 左右。原始数据的取值范围在 0 到 1 2 左右，也即预测相对偏差为大约在 0.01 级别。

## 2 ) 预测数据

指令：

```
python main.py --cmd test --modelfile1 model1.h5 --modelfile2 model2.h5 --datapath ../data --
```

output result.csv

这句话意思是

--cmd test 表示为测试模型

--modelfile1 model1.h5 表示加载 model 1 .h5 这个模型文件作为 1 步预测模型

--modelfile2 model2.h5 表示加载 model 2 .h5 这个模型文件作为 2 步预测模型

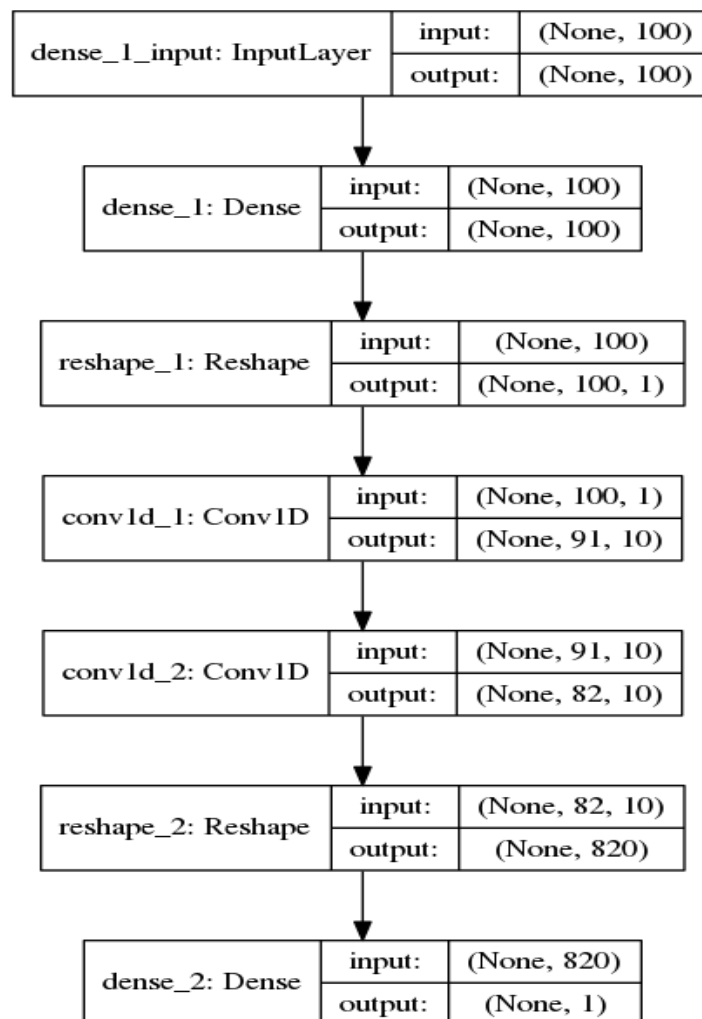
--datapath ../data 表示数据所在的目录

--output result.csv 表示把计算的结果保存在 result.csv

那么运行完后，会自动保存结果到 result.csv。 该文件可以用 e x c e l 打开，格式符合需求里说到的格式。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	lib002.xls	7.59	7.663	7.59	7.642	7.7009372711	7.8276119232	7.8072013855	7.87489748	7.6734366417	7.7789216042	7.7765946388	7.8571848869	
2	lib006.xls	13.493	13.549	13.337	13.483	13.653011322	13.7090978622	13.7633590698	13.8929767609	13.4679946899	13.564207077	13.6699142456	13.7282400131	
3	lib012.xls	6.589	6.688	6.572	6.613	6.6871614456	6.7035393715	6.7811560631	6.7881503105	6.6559104919	6.6641893387	6.7217617035	6.7330255508	
4	lib011.xls	8.193	8.209	8.167	8.209	8.2965965271	8.2802762985	8.3216247559	8.2949094772	8.2638816834	8.2415513992	8.308883667	8.2746524811	
5	lib010.xls	6.522	6.565	6.478	6.53	6.6396179199	6.5161614418	6.6799316406	6.6104545593	6.5778942108	6.4787101746	6.6398429871	6.5392808914	
6	lib003.xls	5.377	5.698	5.377	5.625	5.4897332192	5.4136381149	5.7320580483	5.636488724	5.4403452873	5.3648376465	5.6416010857	5.5449290276	
7	lib013.xls	8.181	8.505	8.116	8.478	8.2932376862	8.2395544052	8.5808944702	8.4446401596	8.2117366791	8.1717224121	8.5309410095	8.4051561356	
8	lib008.xls	12.902	12.945	12.842	12.851	12.9164524078	12.8614053726	13.0059337616	12.9092540741	12.8719387054	12.7837753296	12.909702301	12.8250627518	
9	lib014.xls	10.45	10.928	10.415	10.595	10.4282627106	10.2836370468	10.8258552551	10.6275463104	10.3904705048	10.2120790482	10.6031646729	10.4406280518	
10	lib005.xls	15.812	16.205	15.791	16.128	15.9588661194	15.8663511276	16.2414169312	16.0825119019	15.8482093811	15.7033224106	16.0723228455	15.9374055862	
11	lib001.xls	9.668	9.88	9.658	9.809	9.711271286	9.6076526642	9.8897399902	9.7380962372	9.6848926544	9.575047493	9.8349533081	9.6923780441	
12	lib004.xls	50.297	53.044	50.297	51.449	50.3299026489	51.6318435669	52.5711555481	53.5995483398	49.9157409668	50.7711410522	50.8753662109	51.7447128296	

### 3)展示网络结果图



## 2. 模型和训练方法简要说明

模型如模型图所示，由于是序列数据，所以采用 2 个堆叠的 1 维的卷积层作为模型的核心层。然后输出层采用全连接层。

训练过程并没有像需求那么说的，每个文件，每一列训练一个模型。主要是由于这样的话，数据量不够，且我发现这几个文件，每一列具备一定的相同的特征。所以我把所有文件的所有列的数据都放在一起训练。（当然，也切分了 20 % 的数据作为检验的。因此只用到所有数据量的 80 %）

另外，llb007, llb009 数据由于太少，不足 100 个. 所以抛弃这两个文件。

训练的 epoch 无需太大，大概有 30 到 50 左右就够了，太多了反而不好。程序中加入了训练的技巧，前 5 次中自动以小批量数据训练，剩下的以大批量的数据训练。（对于 gpu 而言，只要能放入内存，批量越大，速度越快）