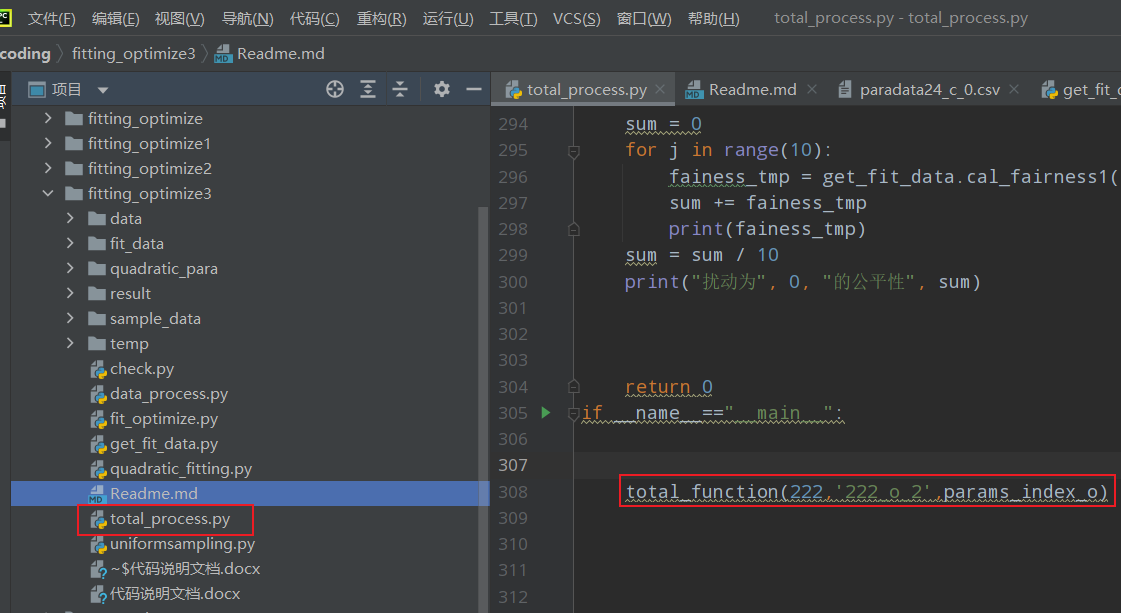
Total\_process文件中的total\_function是将整个流程封装起来的函数:



该函数内部的流程：

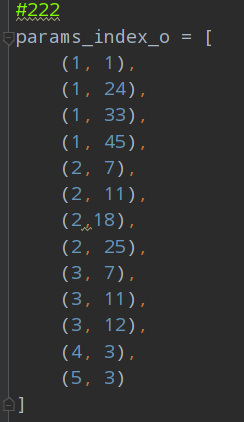
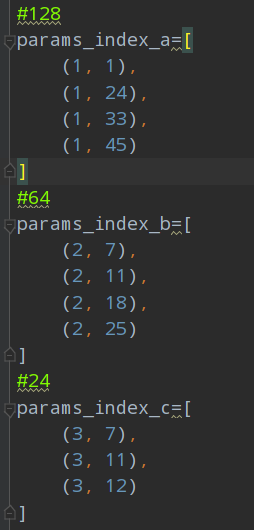
1. 计算得到拟合需要的数据 并保存
2. 进行拟合 得到二次型
3. 根据得到的二次型参数 调用凸优化 得到优化参数取值
4. 根据参数取值修改原网络得到优化后的网络
5. 计算优化前和优化后的网络公平性和准确性 输出

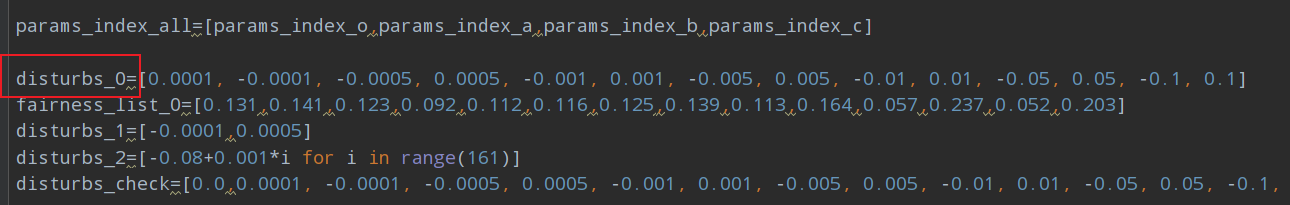
流程图示:



222代表待修复参数个数，”222\_o\_0”代表待修复参数个数222，修复选择的参数是params\_index\_o中的参数 \_0代表选择disturbs\_0中的扰动计算得到拟合所用的数据

下面展现params\_index\_o以及disturbs\_0

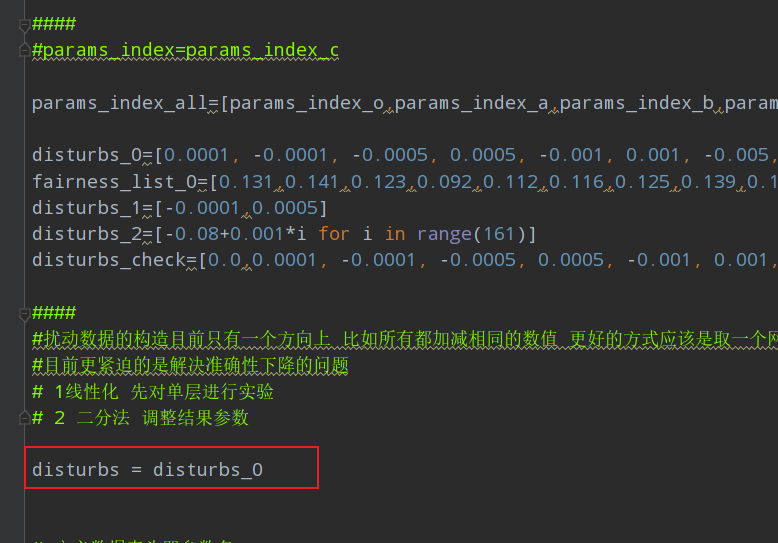




可以看到params\_index\_a/b/c/o选择了不同的待修复神经元 对应了不同的参数个数

Disturbs\_0/1/2代表了不同的扰动方案 用于生成数量不同的拟合数据

需要注意 **其中disturbs的选择需要在外部手动调整**

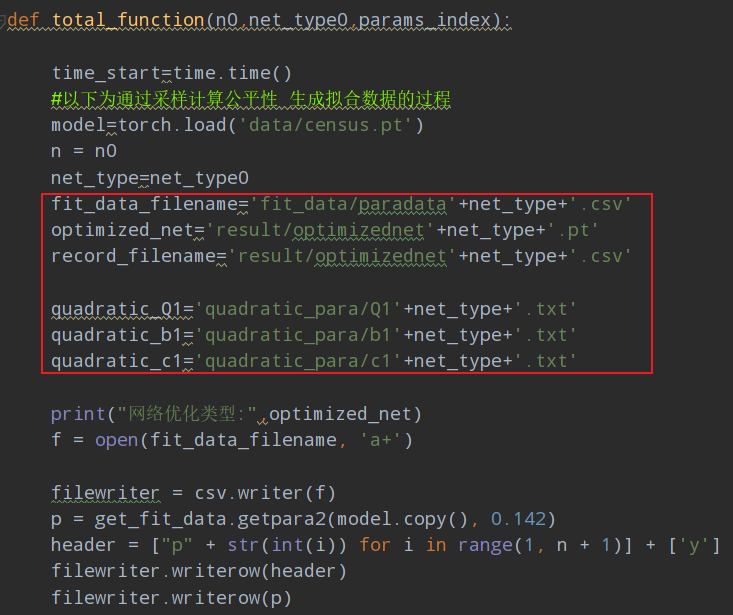


回到total\_function

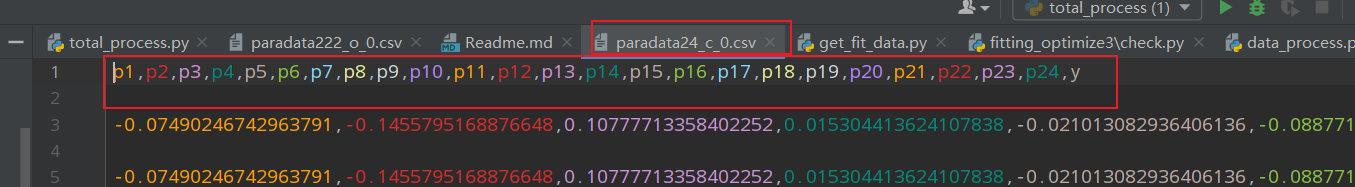


在知道了上述参数后 进入函数内部：

**预先准备：**

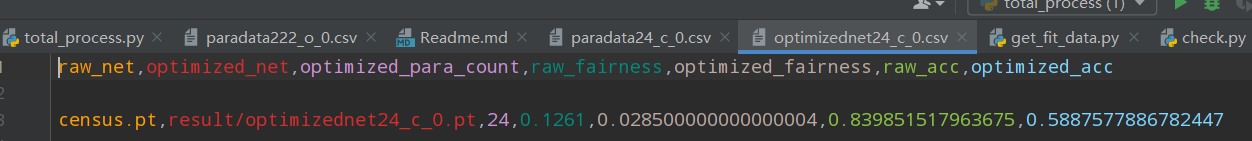


Fit\_data\_filename :用来保存拟合所用的数据：

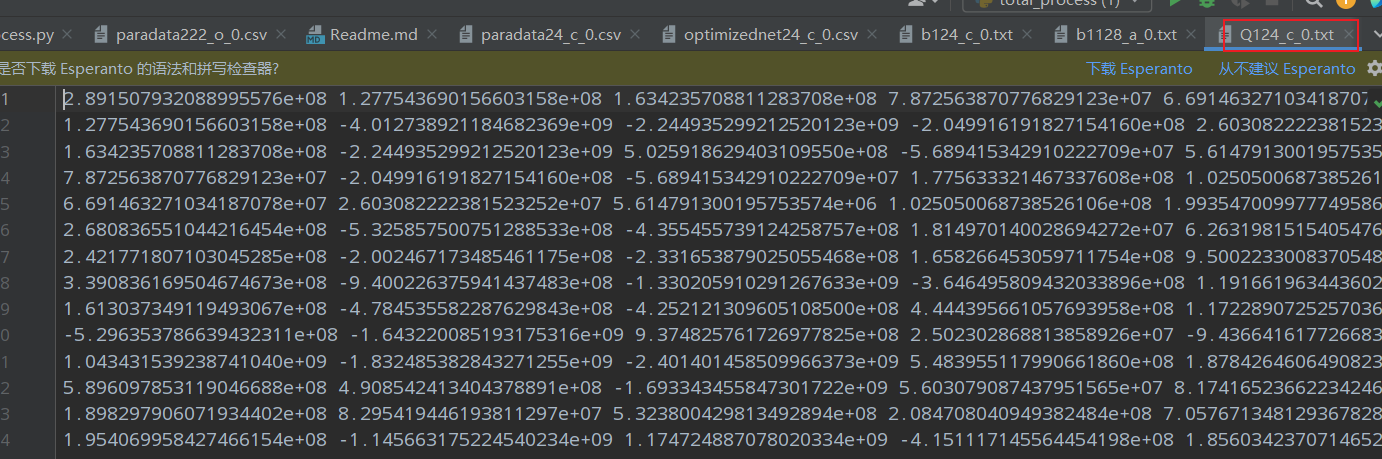


Optimized\_net:用来保存优化后的网络文件

Record\_filename:用来保存优化后的结果：

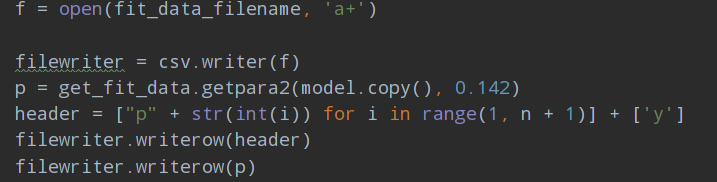


Quardratic\_Q1,b1,c1保存本次修复过程中拟合的二次型的参数



**例如对于Q124\_c\_0**:代表 针对params\_index\_c中神经元相连的24个参数，使用disturbs\_0中的扰动生成拟合数据，拟合得到的二次型的Q的参数 显然，此时应该是24\*24的矩阵

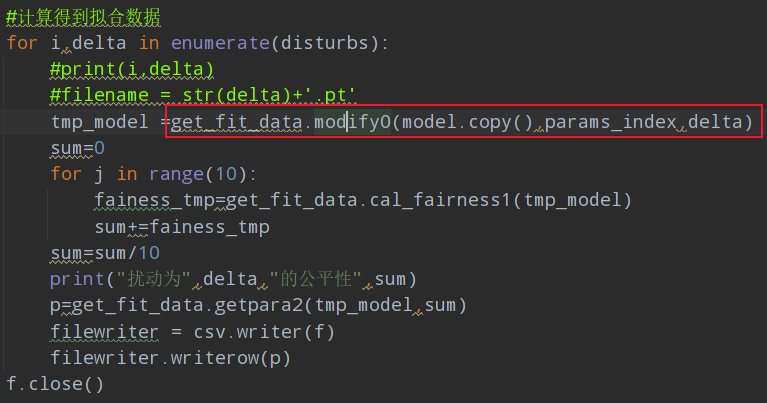
如文件中所示



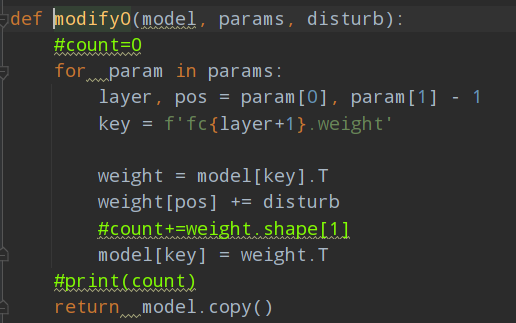
这几行是用来保存拟合数据的 a+代表追加写入

因为要一条条数据进行写入

**生成拟合数据：**



Modify0的作用改变模型计算公平性:



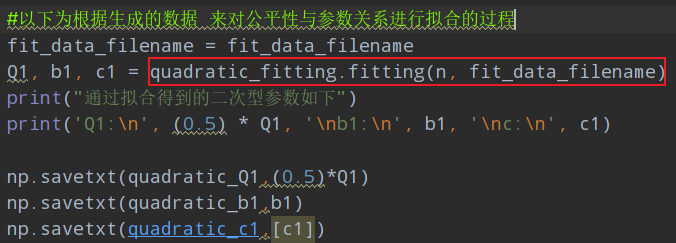
此处的model:

Params:params\_index\_o/a/b/c

disturb是disturs\_0/1/2中的一个元素 是一个具体的数

**Modify0将输入的model在params指定的参数位置进行修改 修改的幅度是disturb**

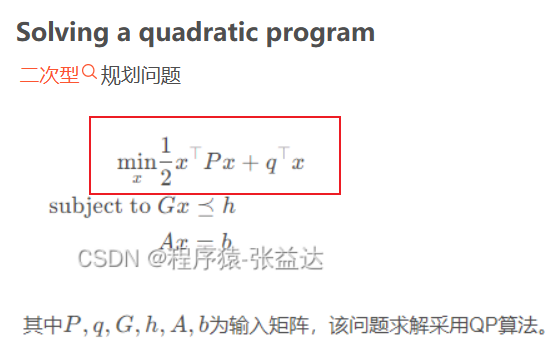
**二次拟合：**



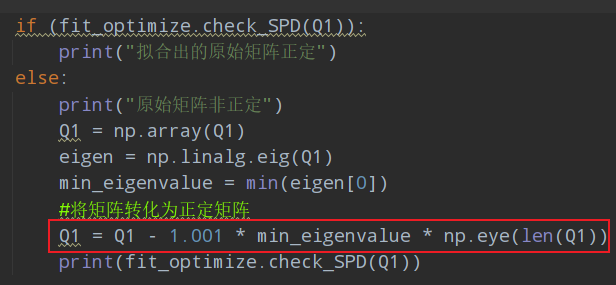
Quadratic\_fitting根据fit\_data\_filename中的拟合数据拟合得到一个n元二次型

随后保存二次型的参数

这里注意，由于在后续调用二次规划库时，库中实现的功能是针对如下形式



因此Q要乘以0.5



当矩阵非正定，通过加上一定倍数的单位矩阵，强行将该矩阵变为正定矩阵

其中min\_eigenvalue是最小的特征值

对修正后的正定矩阵进行二次规划



得到的结果res就是优化的参数取值

例如本次选择了”222\_o\_0”

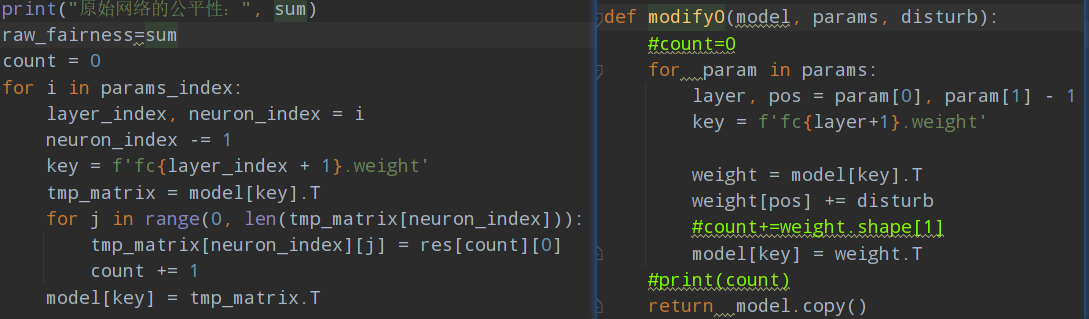
那么res就是一个222长度的向量 其中每一维都代表一个参数的取值

下面将res装进网络里 得到修复后的网络



**注意，根据params\_index\_o选择的神经元提取待修复参数的顺序要与上面装进去的顺序对应，这样才能保证我们把优化后的参数装进了正确的位置**

前面提取参数的代码与该处放进去的代码对比如下(实际上体现在modify0中):



**其中两次采取转置就是因为之前提到的model[key]会返回某一层链接矩阵的转置**

现在就得到了修改后的网络

之后就是调用公平性计算函数以及准确性计算函数的过程 并输出

这就是整个拟合-优化过程