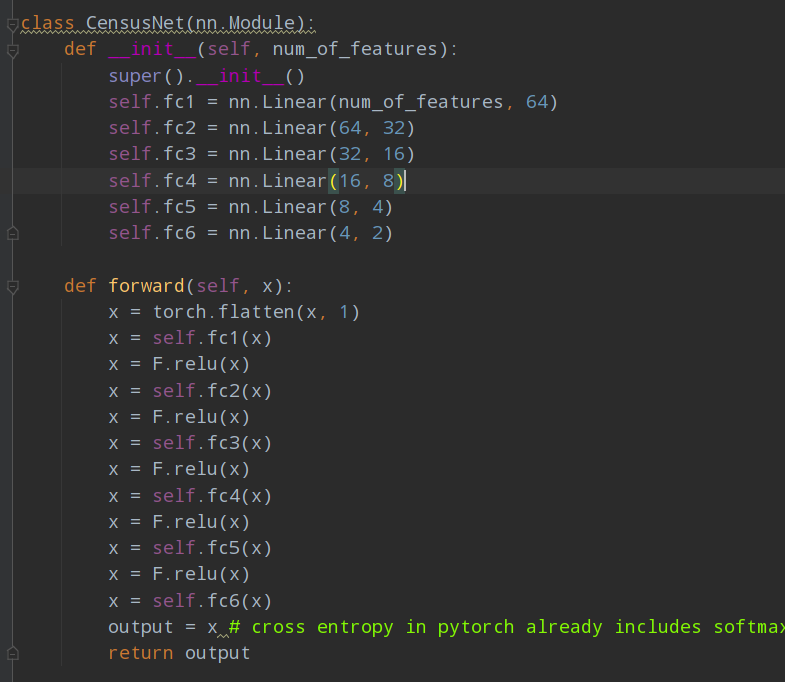
NNlinearize文件是用来进行线性化 并将线性化结果转化为约束的程序

目前NNlinearize是一个demo 完成了 单层的线性化 然后可选择数据集中不同的数据点进行线性化得到的不同方程作为约束

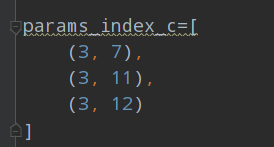
过多的约束会导致问题不可解报错 过少的约束无法提高很多准确性

下面从头到尾来说明：



以上是网络定义

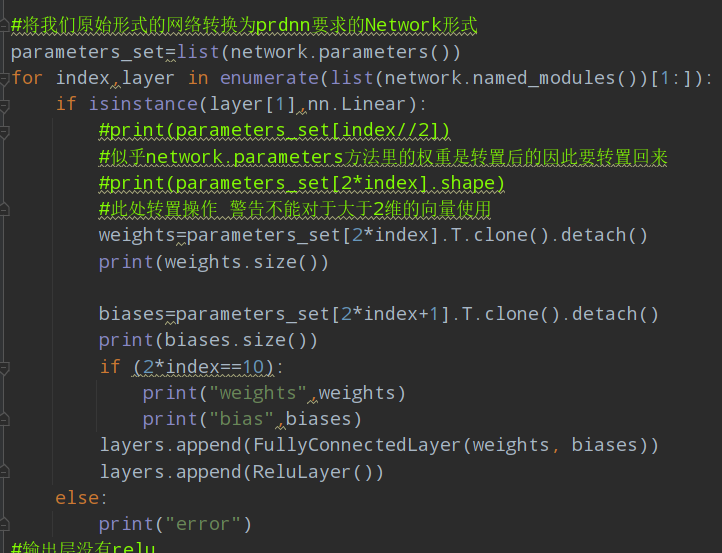
以下是待修复的神经元：



以下为注释掉的一些实验代码 可以用来了解函数的作用和对象的属性



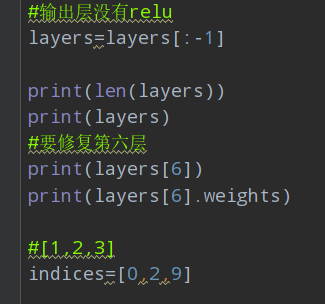
下面是将原始网络转化为prdnn能接受的网络形式 然后调用prdnn的库进行线性化



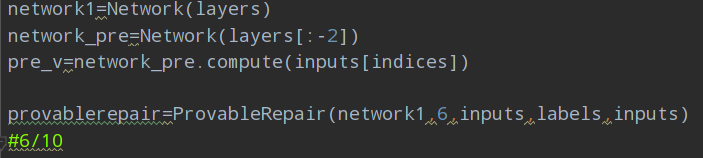
实际过程就是把原网络的权重提取出来 重新赋值给prdnn的网络类的对象

第三层神经元后面的边对应的层次如下:

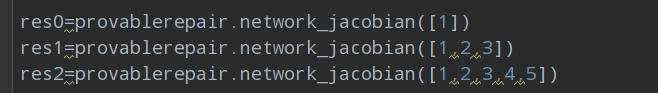




现在如下network1已经成为了用prdnn的形式表示的网络



使用provablerepair类中的函数：



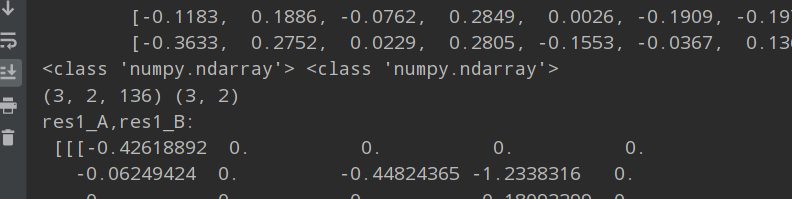
这个函数的作用是返回两个矩阵A\_ B分别代表线性化的结果

具体示例如下：

加入我们选择res1作为说明对象 即为 对索引为1 2 3 的三条数据进行线性化

会得到 3\*2\*136的矩阵A\_(即为代码中的res1\_A)

得到3\*2的矩阵B



对于一条数据而言 例如对于索引为1的数据

A\_[0] B[0] 即为它对应的矩阵A b

这两个矩阵满足 在索引为1的数据点处对网络某一层线性展开时

网络的输出可以被表示为该层权重的线性函数

**例如网络3层 self.fc4中一共有 16\*8个连接权重加上 8个偏置 一共 136个参数**

于是网络输出output=A\*delta\_x+b 其中 delta\_x为第3层的**参数变化量**

Output:2\*1 是一个列向量

A:2\*136

X:136\*1

b:2\*1

对于一条数据 其线性化结果如上所示 由于一次对若干条数据组成的一个batch进行线性化

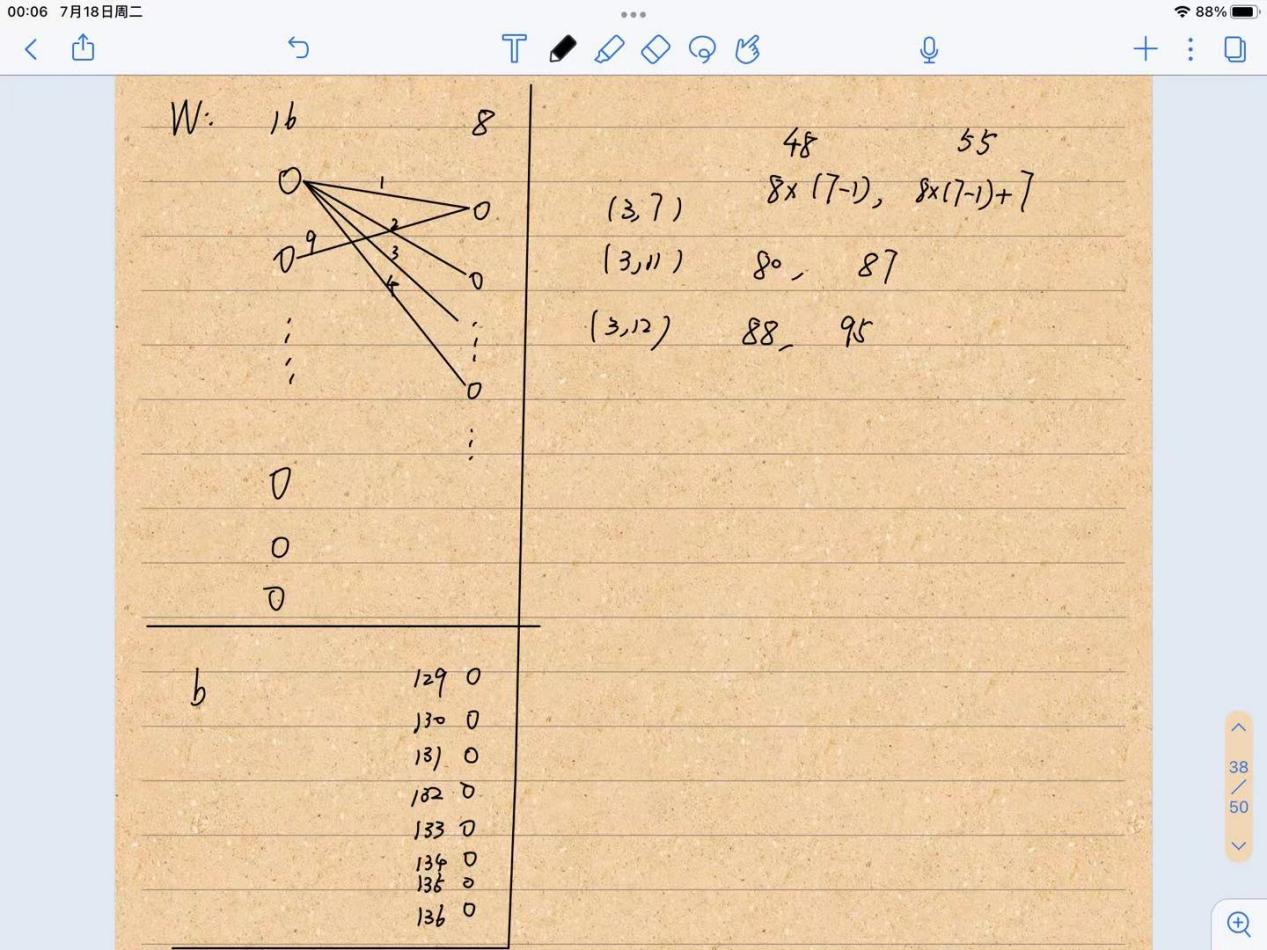
于是若干个A以及b组成了A\_以及B

在这个基础上 还要明确一个关系：A中的元素的含义

例如 我们想找到output的第一个分量 output0 与 第3层 第a个神经元后的第b个边上的权重的关系（因为我们要定位到具体的边）

由于要找参数和输出的第1个分量之间的关系 因此应该找A：2\*136中的第一行的136个权重分别对应那个参数

根据实验观察得出以下结果:



观察得出 W中的参数在A中依次排开 在output=A\*delta\_x+b中

Output[0]=(A\*x+b)[0]

即为output[0]=A的第一行\* delta\_x+ b[0]

delta\_x 中有136个分量 分别代表136个参数的变化量

由于我们只关注一部分参数 于是利用上述观察到的规律把这些参数提取出来

即为图上右侧 A的第一行中 索引48~55,80~87,88~95分别对应（3，7）（3，11)(3,12)这几个神经元后面连接的边 一共是24条

由于不修改别的边 所以别的边在delta\_x里面对应的分量都是0

于是这个公式简化为只跟待修复边有关的

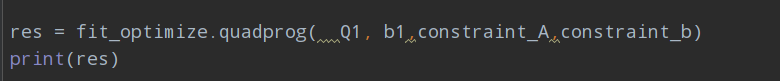
output[0]=A[neuron\_indices]\* delta\_x[neuron\_indices]+ b[0]

然后 如果对于某条数据的正确预测结果是类别0

那就有 output[0]>=output[1] 这就得到了我们需要的约束

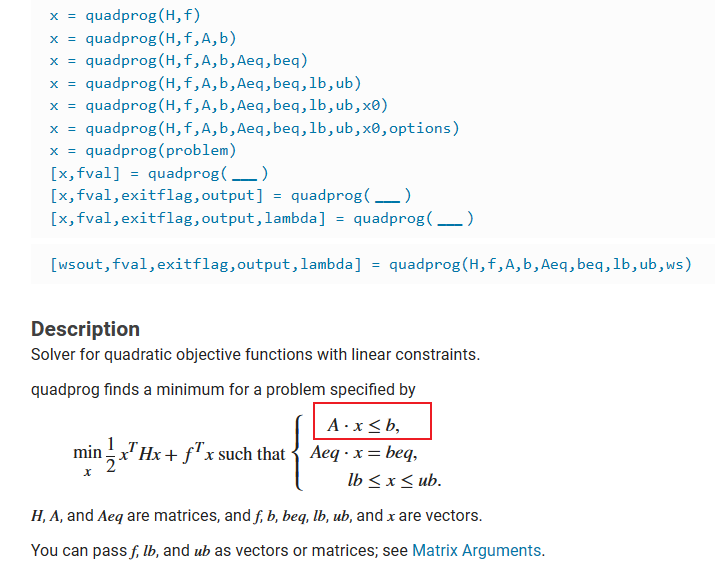
下面就是把这个式子转化为能够使用的形式的过程

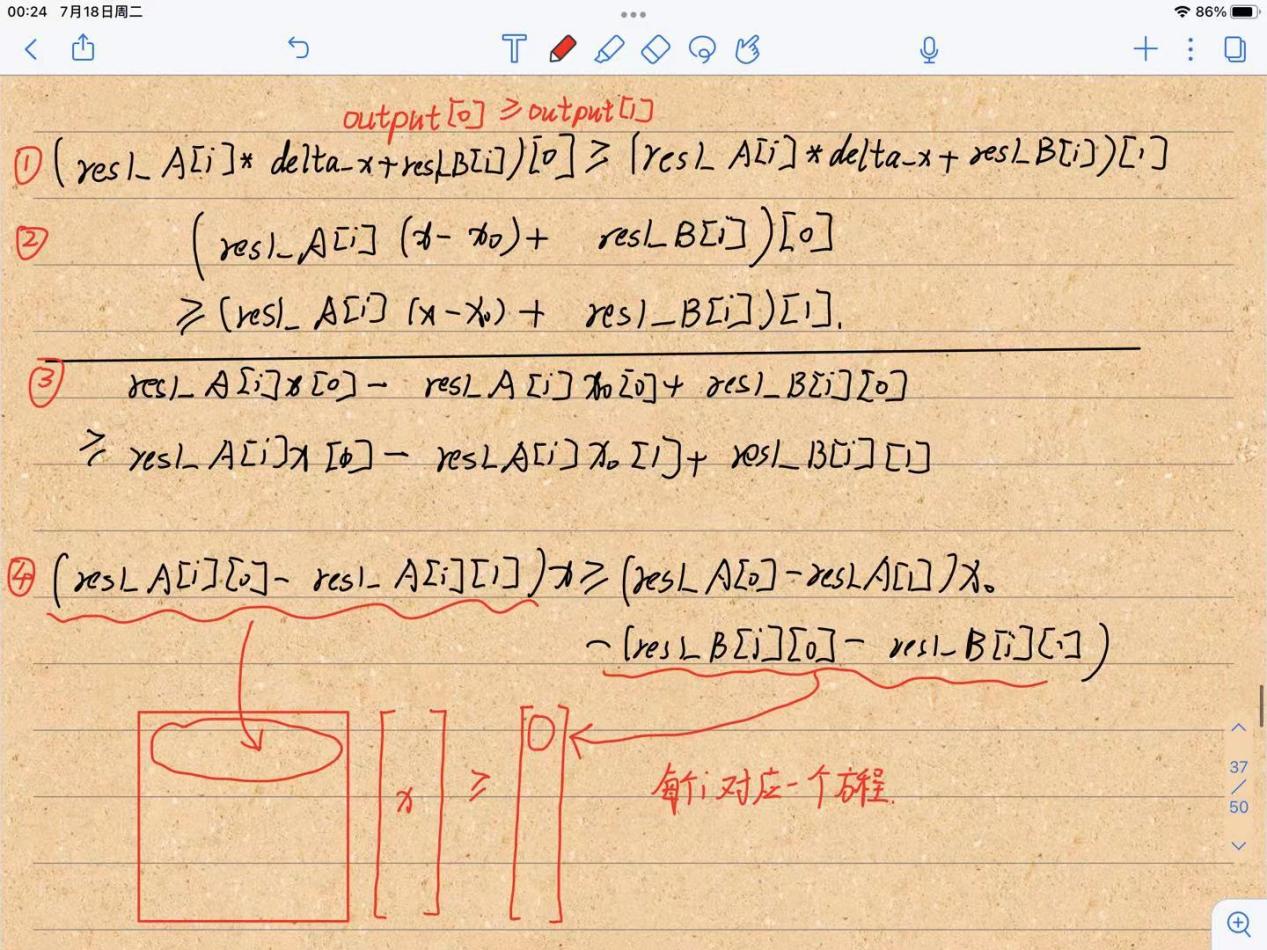
先看我们采用的二次优化函数



这个函数的参数说明如下:

<https://blog.csdn.net/u013421629/article/details/108358409>





对应添加约束的代码如下：

