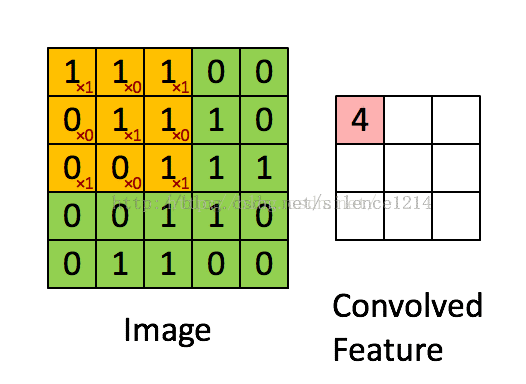
**技术要点一：卷积神经网络**

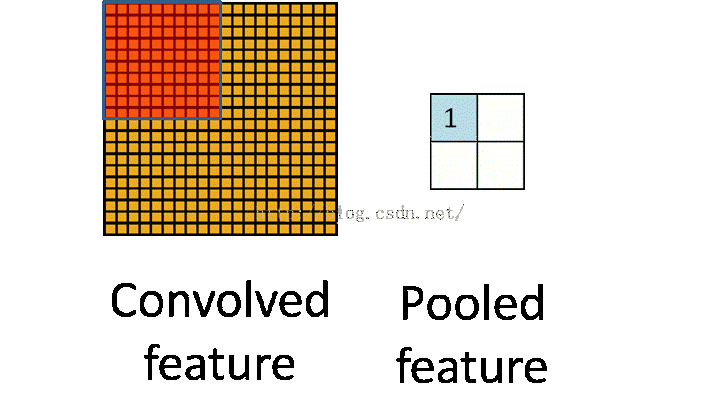
**技术要点二：价值网络**

主要分为三部分步骤，第一步是卷积，第二步为池化，第三步是全连接。



图一、卷积的原理

先介绍卷积原理比较好理解，如图一所示，左侧绿色的部分的5\*5矩阵假设是我们的棋盘信息，然后上面的黄色部分矩阵就是我们的过滤器，用来提取棋盘上的特征，（其实应该叫滤波器或者卷积核），我个人对它的理解是，将其当成是一个棋盘，然后1和0分别代表是否有棋子，让卷积核在输入矩阵上进行从左到右，从上到下滑动，然后每一次滑动，两个矩阵对应位置的元素相乘然后求和，就是右边那个矩阵的一个元素。



图二、池化的原理

再来介绍池化过程，池化分为两种，一种是最大池化，在选中区域中找最大的值作为抽样后的值，另一种是平均值池化，把选中的区域中的平均值作为抽样后的值，这样做的，原因是为了后面全连接的时候减少连接数。

**全连接层**可以用卷积层来代替，全连接层可以转化为卷积核为1x1的卷积。

**卷积的边界处理-padding**

如图一所示，卷积后的矩阵只有3\*3，比原来的图片要小了，因为边界没有了，所以要考虑这个边界的问题，网上说卷积的边界处理有两种方式：

一、丢掉边界，也就是就按右边那个缩小的矩阵来。

二、复制边界，也就是把左边的最外层原封不动地复制过去

但是之前在看matlab代码和tensorflow代码的时候发现并不是那么简单的事情。

matlab中conv2这个“padding”参数可以设为三个值FULL，SAME，VALID

tensorflow中conv2d的"padding"参数可以设为两个值SAME，VALID

它们对边界是这样处理的，对输入的矩阵，包裹n层0，然后再按照上面所说的卷积方法进行卷积，这个n怎么求呢，

FULL: edge\_row = kernel\_row - 1; edge\_cols = kernel\_cols - 1;

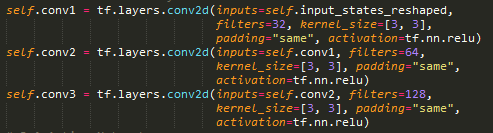
SAME: edge\_row = (kernel\_row - 1) / 2; edge\_cols = (kernel\_cols - 1) / 2;

VALID:edge\_row = edge\_cols = 0;

edge\_row就是边的行数，kernel\_row就是卷积核的行数，所以上面讲的其实就是VALID模式



输入四个棋盘，其中前两个平面分别表示当前player的棋子位置和对手player的棋子位置，有棋子的位置是1，没棋子的位置是0. 然后第三个平面表示对手player最近一步的落子位置，也就是整个平面只有一个位置是1，其余全部是0. 第四个平面，也就是最后一个平面表示的是当前player是不是先手player，如果是先手player则整个平面全部为1，否则全部为0.



这是三个卷积层，卷积核为3\*3的激活函数选用的是RELU，至于padding，采用的是SAME模式的。

然后再分成policy和value两个输出

做个全连接，使用softmax输出棋盘上每个棋子的概率。



然后再用一个卷积核为1\*1的卷积层做全连接，来作为评估网络。

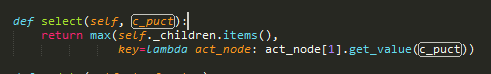


最后再接一个全连接层，使用tanh非线性函数直接输出 [-1-1]之间的局面评分

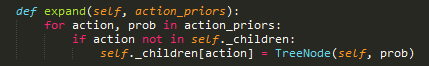
**技术要点三：蒙特卡洛搜索树**

根据PPT上所述，蒙特卡洛树主要作用是通过价值评估函数来对整体盘面进行评估，并且进行剪枝。它会选择评分较大的路径进行搜索，可以很好的减少博弈树的搜索宽度。

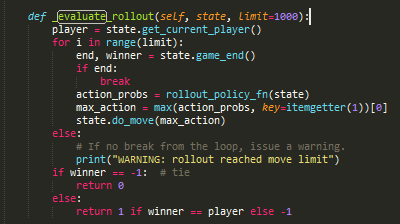
蒙特卡洛搜索树，主要分为选择，扩充，评估和备份四部分。



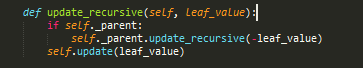
搜索部分主要是比较当前节点以及它的子树的puct值，puct值在先前的训练过程中得到的。



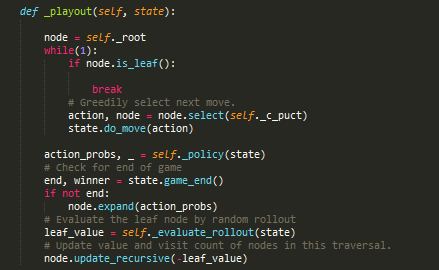
扩充的部分，当前节点的访问节点大于某个阈值的时候，在这个节点的后面插入一个新的节点。其中的先验概率值是来自于策略函数。



Action\_probs是通过策略网络预测得到的概率，然后其中最大的那个就是预测得到的下一步落棋点。



这部分则会更新最新的状态，同时从根部分开始更新。



整体思路就是先通过选择，完成本次走子，然后计算策略，看是否有胜利者，如果没有，就进行扩展，然后再进行评估和更新值。

PS：部分文字摘自我之前写的博客，并没有随便抄袭https://blog.csdn.net/wlmnzf/article/details/50994598