灰度共生矩阵函数与参数

如下图为一个简单的计算实例。图 a 为原图像，最大灰度级为 16。为表示方便，这里将灰度级数减小到了 4 级，图 a 变为图 b 的形式。这样，(f1,f2)取值范围便为[0,3]。取不同的间隔，将(f1,f2)各种组合出现的次数排列起来，就可得到图 e~g 所示的灰度共生矩阵。e 表示图 b 中(x,y)与偏离它的(x+1,y+0)构成点对时，(f1,f2)取值的情况(填充黄色部分为 f1 取 0,f2 取 1 时的情况，由图 b 填充易知共 10 种)。同理，f,g 分别表示图 c,d 中(x,y)分别与点(x+1,y+1),(x+2,y+0)构成点对时点对(f1,f2)出现的情况（图 c 填充黄色部分表示 f1 取 0,f2 取 0 时，对角线点对(0,0)出现的情况，共 8 种；图 d 填充黄色部分表示 f1 取 0,f2 取 2 时水平点对(0,2)出现的情况，共 9 种）。例如，对于 a=1,b=0,点对中(0,1)的组合共出现了 10 次。对比可以看出，(0,1),(1,2),(2,3)和(3,0)均有较高的出现频数。图 b 表明，图像中存在明显的左上右下方向的纹理。

a

2	6	10	14	2	6	10
6	10	14	2	6	10	14
10	14	2	6	10	14	2
14	2	6	10	14	2	6
2	6	10	14	2	6	10
6	10	14	2	6	10	14
10	14	2	6	10	14	2

b

0	1	2	3	0	1	2
1	2	3	0	1	2	3
2	3	0	1	2	3	0
3	0	1	2	3	0	1
0	1	2	3	0	1	2
1	2	3	0	1	2	3
2	3	0	1	2	3	0

c

0	1	2	3	0	1	2
1	2	3	0	1	2	3
2	3	0	1	2	3	0
3	0	1	2	3	0	1
0	1	2	3	0	1	2
1	2	3	0	1	2	3
2	3	0	1	2	3	0

d

0	1	2	3	0	1	2
1	2	3	0	1	2	3
2	3	0	1	2	3	0
3	0	1	2	3	0	1
0	1	2	3	0	1	2
1	2	3	0	1	2	3
2	3	0	1	2	3	0

e

f1 \ f2	0	1	2	3
0	0	10	0	0
1	0	0	11	0
2	0	0	0	11
3	10	0	0	0

(a=1,b=0)

f

f1 \ f2	0	1	2	3
0	8	0	0	0
1	0	9	0	0
2	0	0	10	0
3	0	0	0	9

(a=1,b=1)

g

f1 \ f2	0	1	2	3
0	0	0	9	0
1	0	0	0	9
2	9	0	0	0
3	0	8	0	0

(a=2,b=0)

换言之，先读取数据，然后设定灰度值，进而匹配所有可能的灰度相邻对，按照 0,45,90,135 的四个方向去匹配就可以生成矩阵。

上述已经在算法上实现，但是需要将导入的数据变为标准的 DataFrame 格式，写了四个函数，可以计算四个角度上运算的灰度共生矩阵：

```
def glcm_90(k,start = 1):
    a = []
    for i in range(start,len(k)+1):
        for j in range(start,len(k)+1):
            a.append((i,j))##配对所有可能的灰度组合
    col = []
```

```
    ind = []
    for i in range(start,len(k)+1):
        col.append(str(i))
        ind.append(str(i))
    d = DataFrame(columns= col ,index = ind)##生成空的灰度矩阵
    for double in a:
        count = 0
        for i in range(start,len(k)):
            for j in range(start,len(k)+1):
                    if k.ix[str(i)][str(j)] ==double[0] and k.ix[str(i+1)][str(j)] ==double[1]:
                        count += 1
        d.ix[str(double[0])][str(double[1])] = count
    return d/d.sum().sum()

def glcm_0(k,start = 1):
    a = []
    for i in range(start,len(k)+1):
        for j in range(start,len(k)+1):
            a.append((i,j))##配对所有可能的灰度组合
    col = []
    ind = []
    for i in range(start,len(k)+1):
        col.append(str(i))
        ind.append(str(i))
    d = DataFrame(columns= col ,index = ind)##生成空的灰度矩阵
    for double in a:
        count = 0
        for i in range(start,len(k)+1):
            for j in range(start,len(k)):
                if k.ix[str(i)][str(j)] ==double[0] and k.ix[str(i)][str(j + 1)] ==double[1]:
                    count += 1
        d.ix[str(double[0])][str(double[1])] = count
    return d/d.sum().sum()

def glcm_45(k,start = 1):
    a = []
    for i in range(start,len(k)+1):
        for j in range(start,len(k)+1):
            a.append((i,j))##配对所有可能的灰度组合
    col = []
    ind = []
    for i in range(start,len(k)+1):
        col.append(str(i))
        ind.append(str(i))
```

```
    d = DataFrame(columns= col ,index = ind)##生成空的灰度矩阵
    for double in a:
        count = 0
        for i in range(start,len(k)):
            for j in range(start,len(k)):
                if k.ix[str(i)][str(j)] ==double[0] and k.ix[str(i+1)][str(j+1)] ==double[1]:
                    count += 1
        d.ix[str(double[0])][str(double[1])] = count
    return d/d.sum().sum()

def glcm_135(k,start = 1):
    a = []
    for i in range(start,len(k)+1):
        for j in range(start,len(k)+1):
            a.append((i,j))##配对所有可能的灰度组合
    col = []
    ind = []
    for i in range(start,len(k)+1):
        col.append(str(i))
        ind.append(str(i))
    d = DataFrame(columns= col ,index = ind)##生成空的灰度矩阵
    for double in a:
        count = 0
        p = reversed(range(start+1,len(k)+1))
        q = []
        for c in p:
            q.append(c)
        for i in q:
            for j in range(start,len(k)) :
                if k.ix[str(i)][str(j)] ==double[0] and k.ix[str(i-1)][str(j+1)] ==double[1]:
                    count += 1
        d.ix[str(double[0])][str(double[1])] = count
    return d/d.sum().sum()
```

上述是个函数分别是竖直平移，水平平移，对角线移动的算法。可以生成以 DataFrame 格式保存的灰度共生矩阵。

已简易的灰度信息矩阵为例：

	1	2	3	4
1	1	2	4	3
2	2	3	2	2
3	3	4	1	4
4	4	1	3	1

利用上述四个函数可以生成如下矩阵：

0 度（水平移动）：

	1	2	3	4
1	0	0.0833333	0.0833333	0.0833333
2	0	0.0833333	0.0833333	0.0833333
3	0.0833333	0.0833333	0	0.0833333
4	0.166667	0	0.0833333	0

45 度（对角移动）：

	1	2	3	4
1	0.111111	0	0.111111	0
2	0	0.111111	0	0.222222
3	0.222222	0	0	0
4	0	0.111111	0.111111	0

90 度（垂直移动）：

	1	2	3	4
1	0	0.0833333	0.0833333	0
2	0.0833333	0	0.166667	0.0833333
3	0	0.0833333	0	0.166667
4	0.166667	0.0833333	0	0

135 度（对角移动）：

	1	2	3	4
1	0.111111	0.111111	0	0
2	0	0.111111	0.111111	0
3	0	0	0.111111	0.222222
4	0	0.111111	0	0.111111

ASM 能量（angular second moment)

$$ASM=\sum_{i=1}^{k}\sum_{j=1}^{k}(G(i,j))^2$$

也即每个矩阵元素的平方和。

如果灰度共生矩阵中的值集中在某一块（比如对连续灰度值图像，值集中在对角线；对结构化的图像，值集中在偏离对角线的位置），则 ASM 有较大值，若 G 中的值分布较均匀（如噪声严重的图像），则 ASM 有较小的值。

能量是灰度共生矩阵元素值的平方和，所以也称能量，反映了图像灰度分布均匀程度和纹理粗细度。如果共生矩阵的所有值均相等，则 ASM 值小；相反，如果其中一些值大而其它值小，则 ASM 值大。当共生矩阵中元素集中分布时，此时 ASM 值大。ASM 值大表明一种较均一和规则变化的纹理模式。

对比度（contrast）

$$CON=\sum_{n=0}^{k-1}n^2\{\sum_{|i-j|=n}G(i,j)\}$$

直接反映了某个像素值及其领域像素值的亮度的对比情况。如果偏离对角线的元素有较大值，即图像亮度值变化很快，则 CON 会有较大取值，这也符合对比度的定义。其中 。反映了图像的清晰度和纹理沟纹深浅的程度。纹理沟纹越深，其对比度越大，视觉效果越清晰；反之，对比度小，则沟纹浅，效果模糊。灰度差即对比度大的象素对越多，这个值越大。灰

度公生矩阵中远离对角线的元素值越大，CON 越大。

逆差矩（inverse different moment）

$$IDM=\sum_{i=1}^{k}\sum_{j=1}^{k}\frac{G(i,j)}{1+(i-j)^2}$$

如果灰度共生矩阵对角元素有较大值，IDM 就会取较大的值。因此连续灰度的图像会有较大 IDM 值。

逆差矩： 反映图像纹理的同质性，度量图像纹理局部变化的多少。其值大则说明图像纹理的不同区域间缺少变化，局部非常均匀。

熵（entropy）

$$ENT=-\sum_{i=1}^{k}\sum_{j=1}^{k}G(i,j)\log G(i,j)$$

若灰度共生矩阵值分布均匀，也即图像近于随机或噪声很大，熵会有较大值。

熵是图像所具有的信息量的度量，纹理信息也属于图像的信息，是一个随机性的度量，当共生矩阵中所有元素有最大的随机性、空间共生矩阵中所有值几乎相等时，共生矩阵中元素分散分布时，熵较大。它表示了图像中纹理的非均匀程度或复杂程度。

自相关（correlation）

$$COR=\sum_{i=1}^{k}\sum_{j=1}^{k}\frac{(ij)G(i,j)-u_i u_j}{s_i s_j}$$

其中

$$u_i=\sum_{i=1}^{k}\sum_{j=1}^{k}i\cdot G(i,j)$$

$$u_j=\sum_{i=1}^{k}\sum_{j=1}^{k}j\cdot G(i,j)$$

$$s_i^2=\sum_{i=1}^{k}\sum_{j=1}^{k}G(i,j)(i-u_i)^2$$

$$s_j^2=\sum_{i=1}^{k}\sum_{j=1}^{k}G(i,j)(j-u_j)^2$$

自相关反应了图像纹理的一致性。如果图像中有水平方向纹理，则水平方向矩阵的 COR 大于其余矩阵的 COR 值。它度量空间灰度共生矩阵元素在行或列方向上的相似程度，因此，相关值大小反映了图像中局部灰度相关性。当矩阵元素值均匀相等时，相关值就大;相反，如果矩阵像元值相差很大则相关值小。

上述五种矩阵属性，已经实现：

```
def ASM(m):
    q = m**2
    return q.sum().sum()

def CON(m):
    con = 0
    for i in range(1,len(m)+1):
        for j in range(i+1,len(m)+1):
            a = m.ix[str(j)][str(j-i)]
```

```
                con += i**2*a
        for c in range(1,len(m)+1):
            for u in range(c+1,len(m)+1):
                e = m.ix[str(u-c)][str(u)]
                con += c**2*e
        return con

def IDM(m):
        idm = 0
        for i in range(1,len(m)+1):
            for j in range(1,len(m)+1):
                idm +=(m.ix[str(i)][str(j)]/(1+(i-j)**2))
        return idm

def ENT(m):
        ent = 0
        for i in range(1,len(m)+1):
            for j in range(1,len(m)+1):
                g = m.ix[str(i)][str(j)]
                if g != 0:
                    ent +=(-(g*np.log2(g)))
        return ent

def COR(m):
        cor = 0
        u = []
        v = []
        sj = []
        si = []
        for i in range(1,len(m)+1):
            u.append(i*m.ix[str(i)].sum())
        for j in range(1,len(m)+1):
            v.append(j*m.ix[:][str(j)].sum())
        u1 = sum(u)
        v1 = sum(v)
        for i in range(1,len(m)+1):
            si.append(((i-u1)**2)*m.ix[str(i)].sum())
        for j in range(1,len(m)+1):
            sj.append(((j-v1)**2)*m.ix[:][str(j)].sum())
        s1 = sum(si)
        s2 = sum(sj)
        for i in range(1,len(m)+1):
            for j in range(1,len(m)+1):
                g = m.ix[str(i)][str(j)]
```

```
                cor += ((g*j*i-u1*v1)/math.sqrt(s1*s2))
    return cor
```

上述四个方向的灰度共生矩阵的对应参数如下：

0 度：

(0.09722222222222221,

3.6666666666666674,

0.3666666666666667,

3.4182958340544896,

-75.466666666666683)

45 度：

(0.16049382716049382，

2.7777777777777777,

0.4111111111111111,

2.725480556997868,

-69.643216010117484)

90 度：

(0.125,

3.0833333333333335,

0.35833333333333334,

3.0849625007211561,

-83.839754986098498)

135 度：

(0.13580246913580246,

0.888888888888889,

0.6888888888888889,

2.9477027792200898,

-96.346886026046946)

考虑分析一下该矩阵的行列均值和行列标准差：

```
def STA(m):
    cor = 0
    u = []
    v = []
    sj = []
    si = []
    for i in range(1,len(m)+1):
        u.append(i*m.ix[str(i)].sum())
    for j in range(1,len(m)+1):
        v.append(j*m.ix[:][str(j)].sum())
    u1 = sum(u)/len(m)
    v1 = sum(v)/len(m)
    for i in range(1,len(m)+1):
        si.append(((i-u1)**2)*m.ix[str(i)].sum())
    for j in range(1,len(m)+1):
```

```
            sj.append(((j-v1)**2)*m.ix[:][str(j)].sum())
        s1 = sum(si)/len(m)
        s2 = sum(sj)/len(m)
        return [u1,v1,np.sqrt(s1),np.sqrt(s2)]
```

汇总全部的信息与函数，可以生成一个列表反应四个方向的五个信息，直接传入灰度信息矩阵即可。

```
def glcm_beta(k):
        d =
        DataFrame(columns=['0','45','90','135'],index=['ASM','CON','IDM','ENT','COR','RANGE','IMEAN','CMEAN','ISTD','CSTD'])
        m1 = glcm_0(k)
        m2 = glcm_45(k)
        m3 = glcm_90(k)
        m4 = glcm_135(k)
        d.ix[:]['0']=
        [ASM(m1),CON(m1),IDM(m1),ENT(m1),COR(m1),m1.max().max()-m1.min().min(),STA(m1)[0],STA(m1)[1],STA(m1)[2],STA(m1)[3]]
        d.ix[:]['45'] =
        [ASM(m2),CON(m2),IDM(m2),ENT(m2),COR(m2),m2.max().max()-m2.min().min(),STA(m2)[0],STA(m2)[1],STA(m2)[2],STA(m2)[3]]
        d.ix[:]['90'] =
        [ASM(m3),CON(m3),IDM(m3),ENT(m3),COR(m3),m3.max().max()-m3.min().min(),STA(m3)[0],STA(m3)[1],STA(m3)[2],STA(m3)[3]]
        d.ix[:]['135'] =
        [ASM(m4),CON(m4),IDM(m4),ENT(m4),COR(m4),m4.max().max()-m4.min().min(),STA(m1)[0],STA(m4)[1],STA(m4)[2],STA(m4)[3]]
        return d
```

输出：

	0	**45**	**90**	**135**
ASM	0.097222	0.160494	0.125000	0.135802
CON	3.666667	2.777778	3.083333	0.888889
IDM	0.366667	0.411111	0.358333	0.688889
ENT	3.418296	2.725481	3.084963	2.947703
COR	-75.466667	-69.643216	-83.839755	-96.346886
RANGE	0.166667	0.222222	0.166667	0.222222
IMEAN	0.625000	0.611111	0.645833	0.625000
CMEAN	0.625000	0.583333	0.625000	0.694444
ISTD	1.091516	1.060296	1.098897	1.096519
CSTD	1.091516	1.048312	1.091516	1.162111

可以考虑，将已有 CT 图像已知的结节位置截取图像，然后转化灰度信息表，然后套用上述函数，来看结节图像的特点，然后训练。可以考虑利用上面提取的信息，用 ANN 网络。最后生成一个分类器，然后将 test 的图像按一定原则切分，判断区域内是否存在结节。