2 维核密度估计实验

June 10, 2014

1 问题

讲解 2 维的核密度估计实验，以及需要注意的地方。文章基于 scipy 包的文档[1]上的例子写就。

2 解答

2.1 准备数据

- 样本值：分别从 2 个一维高斯分布中抽样，经过加减运算，再当作二维空间的 X 与 Y。
- 位置点：从二维平面一个方形区域等间隔采点，作为核密度估计的位置点。由 mgrid 函数完成。

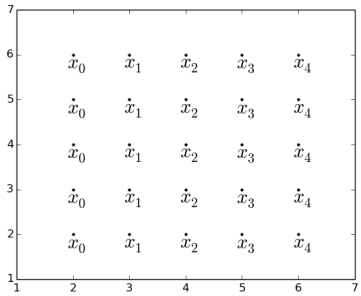
```
r1 = np.random.normal(size=1000)
r2 = np.random.normal(scal=0.5,size=n)
m1 = r1 - r2
m2 = r1 + r2

#样本值
samples = np.vstack([m1,m2])

xmin = m1.min()
xmax = m1.max()
ymin = m2.min()
ymax = m2.max()
```

[1] http://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gaussian_kde.html

```
X,Y = np.mgrid[xmin:xmax:100j,ymin:ymax:100j]
#坐标值
postions = np.vstack([X.ravel(),Y.ravel()])
```

2.1.1 理解 postions

postions 宽为 (xmax - xmin)，高为 (ymax - ymin) 的方形区域。在实际图像中看起来是这样的



写成矩阵形式是

$$\begin{matrix} x_0 & x_1 & x_2 & x_3 & x_4 \\ x_0 & x_1 & x_2 & x_3 & x_4 \\ x_0 & x_1 & x_2 & x_3 & x_4 \\ x_0 & x_1 & x_2 & x_3 & x_4 \\ x_0 & x_1 & x_2 & x_3 & x_4 \end{matrix} \tag{1}$$

但经 np.mgrid 生成的 X 却为：

$$\begin{matrix} x_0 & x_0 & x_0 & x_0 & x_0 \\ x_1 & x_1 & x_1 & x_1 & x_1 \\ x_2 & x_2 & x_2 & x_2 & x_2 \\ x_3 & x_3 & x_3 & x_3 & x_3 \\ x_4 & x_4 & x_4 & x_4 & x_4 \end{matrix} \tag{2}$$

经 X.ravel() 后，排列顺序为：

$$x_0 \quad x_0 \quad \cdots \quad x_1 \quad x_1 \quad \cdots \tag{3}$$

注意 (1) 与 (2) 的旋转关系。这是为什么最终代码里，显示结果时，要调用 np.rot90() 函数。

2.2 求方差

这是二维样本与一维样本区别较大的地方。一维空间的方差，在二维空间里，变成了协方差矩阵。如何理解协方差矩阵？这个“协”就是相关性的意思。N个样本点可以表示成 (2,N) 的形式。第一行全是 X, 第二行全是 Y。

$$X = \begin{matrix} x_0 & x_1 & x_2 & x_3 & \dots & x_{996} & x_{997} & x_{998} & x_{999} \\ y_0 & y_1 & y_2 & y_3 & \dots & y_{996} & y_{997} & y_{998} & y_{999} \end{matrix} \tag{4}$$

对于 X，方差是 sigma_x

对于 Y，方差是 sigma_y

但现在 (X,Y)，不能简单割裂开来，计算 X 的方差时，要考虑 Y 的影响；计算 Y 的方差时，要考虑 X 的影响。所以：

$$\begin{matrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{matrix} \tag{5}$$

如何计算，才能得到这个协方差呢？就要将 X 与 Y 当成上下两行，分别减去其期望：

$$\hat{X} = \begin{matrix} x_0 - \bar{x}, & x_1 - \bar{x}, & \cdots & x_{998} - \bar{x}, & x_{999} - \bar{x} \\ y_0 - \bar{y}, & y_1 - \bar{y}, & \cdots & y_{998} - \bar{y}, & y_{999} - \bar{y} \end{matrix} \tag{6}$$

上面是 (2,1000) 的向量，减去其期望 ($\bar{x}$,$\bar{y}$) 后得到的 (2,1000) 的向量。上面一行用 x_i 表示，下面一行用 y_i 表示

$$\sigma_{xx} = \frac{\sum_{i=0}^{N}(x_i - \bar{x}) * (x_i - \bar{x})}{N-1} \tag{7}$$

$$\sigma_{xy} = \frac{\sum_{i=0}^{N}(x_i - \bar{x}) * (y_i - \bar{y})}{N-1} \tag{8}$$

$$\sigma_{yx} = \frac{\sum_{i=0}^{N}(y_i - \bar{y}) * (x_i - \bar{x})}{N-1} \tag{9}$$

$$\sigma_{yy} = \frac{\sum_{i=0}^{N}(y_i - \bar{y}) * (y_i - \bar{y})}{N-1} \tag{10}$$

注意 σ_{xx} 和 σ_{yy}，二者就是常规的计算方差过程，自身与自身相乘。σ_{xy} 与 σ_{yx} 则是自身与另一向量中对应点相乘，——这样才能体现相关性嘛。

公式有点唬人，转换成代码非常简洁，尤其是 python 中。源码一看便知。

2.3 计算带宽

这个与 1 维的完全一样, 仍然选用 scotts_factor。

$$h = N^{-\frac{1}{d+4}} \tag{11}$$

2.4 二维高斯公式

上面该准备的变量准备好，现在给出基于这些变量的高斯公式。

$$G(x,y) = \frac{1}{\sqrt{2\pi * det(A)} * h} * e^{-\frac{1}{2} * \sum_{i=0}^{N} \frac{(Xpos - X_i) * (Xpos - X_i)}{A^{-1} * h^2}} \tag{12}$$

$$A = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{bmatrix} \tag{13}$$

其中 A 是协方差矩阵，det(A) 表示求行列式的值。X 是二维的，如公式 (1) 所示。h 是带宽。A^{-1} 表示矩阵的逆。

2.4.1 对应代码的解释

这一计算过程由以下代码完成：

```
diff = values[:,i,newaxis] - positions
tdiff = dot(tdiff_factor, diff)
#NOTE: the meaning of sum the diff*tdiff
energy = sum(diff*tdiff,axis=0) / 2.0
```

代码与公式的对应关系：

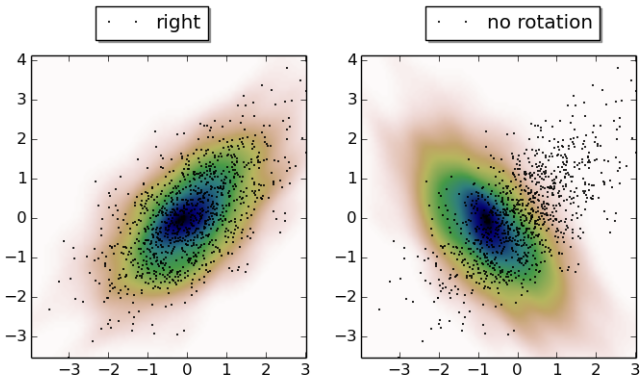
- diff: 公式中绿色的部分
- tdiff: 公式中红色的部分
- tdiff_factor: 对应公式中的 $A^{-1} * h^2$
- dot() 函数：矩阵乘法，符合矩阵乘法规律，(2,2)*(2,1000)→(2,1000)

- diff*tdiff，向量对应点相乘。X 行乘以 X 行，Y 行乘以 Y 行。(2,1000)*(2,1000)→(2,1000)
- sum() 函数：上下两列相加， 2 维高斯是由 1 维高斯相乘而来，在指数上表现为 X 与 Y 的相加。

2.5 代码及实验结果

完整的源码在：https://github.com/xueyayang/v4l2_demo/blob/master/kernel-density-estimation/gaussian_2d_kde.py

实验结果如下：



注意其中的 no rotaion。如果对结果不旋转的话，画出的图像与样本点的分布是垂直的。原因是因为生成的结果，顺序与输入的 samples 的顺序一致。而输入的 samples(公式 2) 与位置点 (公式 1) 的顺序，存在着旋转关系。

3 总结

- 理解协方差很重要
- 理解一维到二维时，高斯公式是相乘的，但指数是相加的。

- 最终的结果要旋转。