二次型能量函数的求解

徐 辉

2016 年 3 月 11 日

1 什么是二次型能量函数

在计算机视觉，图像处理和机器学习等很多领域，经常会遇到各种能量函数。有人统计，在计算机视觉领域，有 30% 左右的论文，它们的模式都是这样的：作者首先提出一个问题，然后分析这个问题，给出这个问题的约束条件，接着定义一个能量函数，最后求解这个能量函数，就号称解决了这个问题。这种思路有章可循，百试不爽。

但是，如何去定义和求解一个能量函数，可不是一件简单的事情。在这里，我们试图去解决一类简单类型的能量函数，称其为“二次型”能量函数。为什么这么称呼，因为它们都可以转化成二次型的形式求解。而且这类能量函数在图像处理论文中经常出现。

比如在图像去噪时，有一幅噪声图像 Y, 我们想要得到的理想图像设为 X。另外，我们还要给出理想图像的一个约束：图像看起来比较光滑。至于光滑怎么去定义，仁者见仁。这里我们给出一个最简单的假设：每个像素点与它邻域中像素点的灰度相差不能太大。这样的话，我们可以很顺利地写出这个能量函数：

$$\min_X \sum_i (X_i - Y_i)^2 + \alpha \sum_i \sum_{j \in N(i)} (X_i - X_j)^2$$

这个能量函数很容易理解，而且这也是一个最简单的二次型能量函数。

我们先来观察这个能量函数，第一项称为保真项，用来约束我们想要求得的理想图像 X 与实际观测图像 Y 的差异，即求解出来的图像不要太离谱，不能与原来的图像相差十万八千里。

第二项可以称为邻域项。因为图像存在这样一种非常经典的网状结构，如下图 1 所示。而且图像自身也有这种天然的局部性质，即两个相邻的

像素点的有关属性往往比较一致。我们通过约束每一对相邻像素点的性质(灰度差值不能太大)，去约束整体图像的性质。

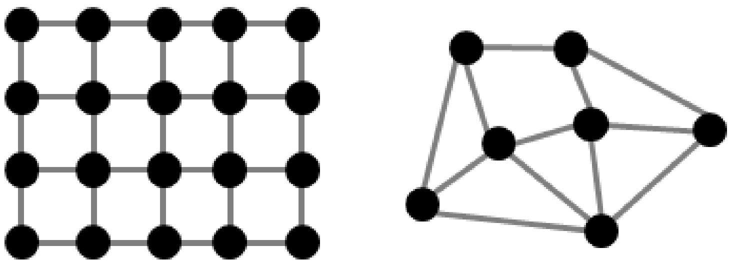

图 1: 左图是图像像素的连接结构，右图是超像素的连接结构

这里千万不要忽略参数 α 的作用。在实验调参的时候，就靠它了。因为，我们想要加在理想图像上的约束条件往往不能同时达到，有时甚至是相互矛盾的。这时参数 α 就起到一种协调和妥协的作用，它会使求解出来的目标图像既不会过分光滑偏离原图像 (保真项约束)，也不会噪声太大 (邻域项约束)，如下图 2 所示。

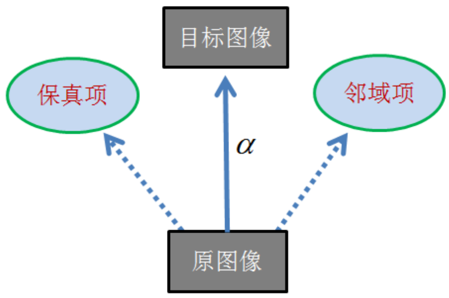

图 2: 参数 α 的作用

2 怎么求解

2.1 简单的规则

如果以前没有求解能量函数的经验的话，可能在面对一个能量函数时，知道其表达什么意思，但不知如何求解。

这里，有一个简单的规则：我们一般把能量函数中的未知量都看成列向量。为什么？

首先，数学中，一般不用行向量。行向量是横着写的，就是一行，大小为 $1 \times N$，像这样 $v = (a_1, a_2, ..., a_N)$。而列向量是竖着写，大小为 $N \times 1$，比较浪费书写空间。即使这样，数学家也只用列向量。为了省空间，一般数学教材上都这样写列向量 $v^T = (a_1, a_2, ..., a_N)$，通过转置，放在一行。

还有，有人会问，很多情况下的未知量，比如像这里的图像 X，将其看成矩阵去求解，不是更符合直觉吗？这其实是一个误解，可以这样说，几乎没有人去求解一个未知的矩阵，即使想要去求矩阵，我们一般也会将其转化成列向量去求解。

最后，几乎所有优化理论教材中讨论的都是未知量为列向量的情形 (未知量为实数已被包含其中)。线性代数中的线性方程组 $Ax = b$，其实也就是列向量的方程组。几乎很少见到矩阵方程组 $AX = C$，如果有，请转化成线性方程组求解。

好吧，这么多毫无逻辑的理由，我能想到的就这么多了。反正大家一定要记住这个简单的规则！

2.2 保真项的改写

好了，现在我们把 X, Y 都看成列向量，大小都是 $N \times 1$，N 是图像上像素点的总个数。很简单，保真项

$$\sum_i (X_i - Y_i)^2$$

可以写为

$$(X - Y)^T I (X - Y)$$

这里的 I 是一个大小为 $N \times N$ 的单位矩阵。大家可以试着去将这个二次型展开，看看是不是正好等于保真项。

$$\begin{pmatrix} x_1 - y_1, & \cdots, & x_i - y_i, & \cdots, & x_N - y_N, \end{pmatrix} \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix} \begin{pmatrix} x_1 - y_1 \\ \vdots \\ x_i - y_i \\ \vdots \\ x_N - y_N \end{pmatrix}$$

有人会问，这里为什么要加上一个单位阵。主要是因为如果存在像素缺少的情况下，可以将单位阵对角线上相应的位置置 0。

2.3 二次型的回忆

那么如何变换邻域项的形式，这可不像上面的保真项一样，能够一眼看出来。如果大家对线性代数中的二次型很熟的话，可能觉得这不是二次型吗?

我们首先来回顾一下二次型的概念。我们称 X^TAX 为一个二次型，其中，X 为一个 $1\times N$ 的列向量，A 为二次型的矩阵，一般为对称矩阵，二次型的结果应该是一个实数值。

$$X=\begin{pmatrix} x_1\\ x_2\\ x_3\end{pmatrix}\quad A=\begin{pmatrix} a_1 & a_2 & a_3\\ a_4 & a_5 & a_6\\ a_7 & a_8 & a_9\end{pmatrix} \tag{1}$$

其中，A 为对称矩阵，$a_2=a_4,a_3=a_7,a_6=a_8$，则有:

$$\begin{aligned} X^TAX&=\begin{pmatrix} x_1 & x_2 & x_3\end{pmatrix}\begin{pmatrix} a_1 & a_2 & a_3\\ a_4 & a_5 & a_6\\ a_7 & a_8 & a_9\end{pmatrix}\begin{pmatrix} x_1\\ x_2\\ x_3\end{pmatrix}\\ &=a_1x_1^2+a_5x_2^2+a_9x_3^2+(a_2+a_4)x_1x_2+(a_3+a_7)x_1x_3+(a_6+a_8)x_2x_3\\ &=a_1x_1^2+a_5x_2^2+a_9x_3^2+2a_2x_1x_2+2a_3x_1x_3+2a_6x_2x_3 \end{aligned} \tag{2}$$

从上面的二次型的展开式可以看出，每一个小式子都是二次的，对角线上的元素 a_1,a_5,a_9 分别与纯种的二次式 x_1^2,x_2^2,x_3^2 相乘，而其他的都是与杂种的二次式 x_1x_2,x_1x_3,x_2x_3 相乘。

2.4 邻域项的转换

上面简单复习了二次型的形式，下面的讨论就简单了。我们现在来观察邻域项，

$$\sum_i\sum_{j\in N(i)}(X_i-X_j)^2 \tag{3}$$

重点考察第 m 个像素点 x_m 和它的 4 个邻域点 $x_{n_1},x_{n_2},x_{n_3},x_{n_4}$，有

$$
\begin{aligned}
&\sum_{n \in n_1,n_2,n_3,n_4} (x_m - x_n)^2 \\
&= 4x_m^2 + x_{n_1}^2 + x_{n_2}^2 + x_{n_3}^2 + x_{n_4}^2 - 2x_m x_{n_1} - 2x_m x_{n_2} \text{-} 2x_m x_{n_3} - 2x_m x_{n_4}
\end{aligned} \tag{4}
$$

哈哈，多么熟悉的二次型的形式！稍安勿躁，我们先来陈述一下定义。

现在，我们可以将上式 (3) 改写为 $X^T H X$，其中 X 为 $N \times 1$ 的列向量，H 为一个 $N \times N$ 的对称矩阵，N 是图像上像素点的总个数。

那么矩阵 H 的每一个元素到底是多少呢?

根据上面的分析，对于第 m 个像素点来说，如式 (4) 所示，它给矩阵 H 带来的是对角线上的第 m 个位置为 4，第 n_1, n_2, n_3, n_4 个位置为 1，另外还在一些对称的位置上，有 8 个 -1。思考一下，第 m 个像素带来的子矩阵 H_m 是不是下面这个样子：

$$
H_m = \begin{pmatrix}
\ddots & & & & & & & & \\
& 1 & & & -1 & & & & \\
& & \ddots & & \vdots & & & & \\
& & & 1 & -1 & & & & \\
& -1 & \cdots & -1 & 4 & -1 & \cdots & -1 & \\
& & & & -1 & 1 & & & \\
& & & & \vdots & & \ddots & & \\
& & & & -1 & & & 1 & \\
& & & & & & & & \ddots
\end{pmatrix} \tag{5}
$$

如果大家可能看得不清楚，可以自己再认真地想一想，画一画。

现在好办了，H_m 是第 m 个像素点的子矩阵，那么总的矩阵为：

$$
H = H_1 + H_2 + ... + H_N,
$$

大家可以算一下，是不是下面的样子：

$$
H = (H_{ij})_{N \times N}
$$

H_{ij} 是矩阵 H 的第 i 行第 j 列的元素。

$$H_{ij} = \begin{cases} 2n & if \quad i = j \\ -2 & if \quad i \neq j, \quad i \in N(j) \\ 0 & else \end{cases}$$

其中，n 为邻域中像素点的个数，这里为 4 。到此为止，我们就给出了邻域项的形式。

2.5 最后的解决

上面我们分别把保真项和邻域项改写成了二次型的形式，即

$$\min_X (X-Y)^T(X-Y) + \alpha X^T H X$$

对于求解这个二次型形式的能量函数，可以运用线性代数和矩阵理论中的知识，对 X 求导，并导数为 0 .

$$2(X-Y) + 2\alpha HX = 0$$

所以 $X = (\alpha H + I)^{-1} Y$. 这其实就是求一个线性方程组的过程。

这里有人会问，这个矩阵 H 的大小太大，怎么去求解方程组？比如，一幅图像，500×500，总的像素点个数 25 万。也就是说，X 是一个 25 万个元素的列向量，而 H 更是不可思议的大，25 万 $\times$25 万，有 625 亿个元素。

其实，这里我们注意到矩阵 H 是一个非常稀疏的矩阵，矩阵的每一行虽说有 25 万个元素，但只有 5 个元素非零。也就是说，矩阵 H 中，只有 5 万分之一的元素非零，也就 125 万个。在数值计算中，稀疏方程组的求解是一个非常重要的问题。现在，有很多软件包，matlab 等都可以去求解稀疏方程组，而且速度很快。这里的意思是，计算并不是问题。

3 真实的例子

上面的那个能量函数，是为了说明问题，我编造的一个简单的例子。这里介绍一个真实的例子，这是王晓洁跑来问我的，王文平老师在 CAGD 上的一篇论文《Denoising point sets via L_0 minimization》。这篇论文是对三维点云进行去噪。这里，我们只讨论文章中的第一个问题，法向估计。

一般来说，我们是通过 PCA 对每个点进行法向估计的。这篇文章似乎不满足这种法向估计的精度，只是将这个 PCA 估计的法向 u 当做观察到的粗糙的法向，将理想的法向记做 v。

这篇文章还对 k-邻域点中的 k 个点的法向进行了约束，认为它们大部分应该相同。这样说可能不清楚，下面详细来介绍。

假设点云中一共有 N 个点，$P_i, i = 1, 2..., N$。对于每个点 P_i，首先寻找它的 k 个最近的点 $P_{i_1}, ..., P_{i_j}, ..., P_{i_k}, j = 1, ..., k$.

我们把所有的点的 PCA 法向排成一列，记做 U, 所有点的理想法向排成一列，记做 V，U 和 V 都是有 N 行。

我们把所有点与它的邻域中 k 个点的法向差也排成一列，记做 D, 一共有 kN 行, 第 1 行到第 k 行分别是第一个点与其邻域中 k 个点的法向之差，第 $k+1$ 行到第 $2k$ 行分别是第二个点与其邻域中 k 个点的法向之差, 依次类推。即

$$D_{ik+j} = V_i - V_{i_j} \qquad i = 1, ..., N \quad j = 1, ..., k$$

在估计理想的法向 V 时，这篇文章给出的能量函数如下:

$$\min_{V} \quad (V-U)^2 + \alpha |D|_0$$

这个能量函数的定义也很好理解。第一项，我们也可以看做法向的保真项，第二项的 0-范数是想使邻近点之间的法向差出现更多的 0, 即邻近点法向相同。

怎么去解呢? 似乎不容易，这里有个 0-范数，通常的解法是分离变量法, 引入一个辅助变量 t, 把 t 看成 D 的双胞胎，用 t 替换 D, 有

$$\min_{V,t} \quad (V-U)^2 + \alpha |t|_0 + \beta (t-D)^2$$

为什么可以这样替换? 我们观察上式，如果 β 非常大，最小化这个能量函数，其实就是使 t 慢慢逼近 D, 这样的话，t 的 0-范数和 D 的 0-范数就相差不大，可以相互代替了。

是不是很有意思? 一开始只有 1 个变量 V，现在硬生生地添加一个变量 t，结果反而变得更容易解了。

对于多个未知变量的能量函数的求解，我们一般使用交替迭代法，即每次只求一个未知变量的值，固定其他所有的变量。

首先，固定变量 V, 去求 t。第一项保真项为定值，V 已知，那么 D 可以算出来，于是问题变为：

$$\min_t \quad \beta(t-D)^2 + \alpha|t|_0$$

如果对稀疏编码有点了解的话，可以知道这个式子，可以通过简单的硬阈值收缩求解。

然后，固定 t, 去求 V。这才是我们今天要讲的主题，前面介绍的分离变量法，交替迭代法，以后有时间再详细介绍。如果 t 固定，稀疏项就是定值，可以去掉，就剩下两个二次项。

$$\min_V \quad (V-U)^2 + \beta(D-t)^2$$

这样看，好像和我们前面说的二次型不太像，我们再展开一下看看

$$\min_V \quad (V-U)^2 + \beta\sum_i \sum_{j\in N(i)} (V_i - V_j - t_{ij})^2$$

这里的 t_{ij} 表示 t 的第 $ik+j$ 行，因为 t 的大小与 D 相同。

似乎一切都变得熟悉起来。这里还有几个小问题。

第一个，因为这里求的是法向，有 3 个分量，幸好这 3 个分量互不影响，我们可以分 3 次，分别来求。

第二个，这里的括号中多了一个变量 t_{ij}，如何处理。有人很快看出来，在二次型后面加个“一次型”就行，转换成 $V^THV + K^TV$ 这样的形式。对，的确是这样。

我们简单展开一下，大家就能看明白。假设对于第 i 个点，它只有 2 个邻近点 j_1, j_2，那么，

$$\begin{aligned}&(V_i - V_{j_1} - t_{ij_1})^2 + (V_i - V_{j_2} - t_{ij_2})^2 \\ &= 2V_i^2 + V_{j_1}^2 + V_{j_2}^2 - 2V_iV_{j_1} - 2V_iV_{j_2} + 2t_{ij_1}(V_{j_1} - V_i) + 2t_{ij_2}(V_{j_2} - V_i)\end{aligned}$$

前面的 5 个二次项前面已经讲过了，可以转换成二次型。后面的一次

项可以写成

$$
\begin{aligned}
&2t_{ij_1}(V_{j_1}-V_i)+2t_{ij_2}(V_{j_2}-V_i)\\
&=2\begin{pmatrix}\cdots, & t_{ij_1}, & \cdots, & -t_{j_1}-t_{j_2}, & \cdots, & t_{ij_2}, & \cdots\end{pmatrix}\begin{pmatrix}\vdots\\ V_{j_1}\\ \vdots\\ V_i\\ \vdots\\ V_{j_2}\\ \vdots\end{pmatrix}
\end{aligned} \tag{6}
$$

这样，一次项可以写出 $2K^TV$ 的形式, 第二个子问题的能量函数如下：

$$\min_V \quad (V-U)^T(V-U)+\beta VH^TV+2\beta K^TV$$

接下来的求导，解线性方程组和前面类似，这里就不说了。

这里，还有一个关于编程序时设置矩阵 H,K 的小技巧，

Algorithm 1 Compute H,K

```
initialize H ← 0, K ← 0
for each point P_i do
    find k nearest points
    update some elements in H, K
end for
```

4 最后的总结

能量函数在研究中广泛使用。很多人只能看得懂，但不知道任何去使用。我觉得如果想要能灵活地运用能量函数，必须首先得学会简单的能量函数的解法。因为，如果自己去定义一个能量函数，都不知道这个能量函数可不可解，如何去解，显然不行。另外，如果我们知道一类能量函数的解法，可以反过来指导我们去定义能量函数：我们可以尽量将我们的约束条件转化为我们可解的能量函数形式。

对于本文中所提的“二次型”能量函数，其实比较常见。比如在图像处理中，像素点之间有着天然的整齐的邻域结构，而在超像素和点云中，它们

的的结构虽然不整齐，但是也存在相互的邻接关系，这些情况下，都适合定义二次型”能量函数。

5 简单的练习

大家如果回忆以前看过的论文，可能见到很多“二次型”能量函数，可以试着去解一解。这里给出两个简单的练习：

$$\min_X \quad \sum_i (X_i - Y_i)^2 + \alpha \sum_i \sum_{j \in N(i)} w_{ij} (X_i - X_j)^2$$

$$\min_X \quad \sum_i (X_i - Y_i)^2 + \alpha \sum_i (X_i - \sum_{j \in N(i)} w_{ij} X_j)^2$$

在我的毕业论文中，还有两个复杂点的二次型能量函数，大家可以试着去解一下。