Robust PCA?

Jonash, 黃輔中

2006.7.5

Robust PCA 就是把 Principle Component Analysis 加上一個 Robust Function 去測量在各個 dimension 以內的 outliers, 並且想辦法去做修正的一個方法。以下各段, 我分別會講 Robust Statistic/Function, PCA, 前人的方法, 以及本篇 paper 的作法。

1 Robust Statistics

首先我門要了解什麼是 Robust Statistics. 在做統計時, 我們通常會選用 euclidian distance 來當作原始資料跟做完統計後資料之間的誤差, 更簡單的說, 我們常常會去算一個 $error\_distance = (Data\_ori - Data\_fit)^2$, 我門希望這樣的距離越小越好 (Least Square Fit) 而要去找一個好的 $Data\_fit$, 這樣的距離最常用在 regression 方面。

例如 Linear Regression(2D), 我們通常希望用一條線去 fit一堆資料。Linear Regression 用的$error_distance$就是我們剛剛講的方法, 現在$Data_fit$就是$Data_ori$在那條要找的線上的投影點使得該距離最小。根據這樣的條件, 整個 model的 total error 就是 $\sum_i (Data\_ori_i - Data\_fit_i)^2$ 。

在沒有 outlier 只有 noise 的情況下, 這樣的 model 可以把原始資料 fit 很好, 可是在有 outlier 的情況下, 事情就不是這麼樂觀了。我們再 Figure 1(a) 可以看到最右邊有一個 outlier資料。爲了去 minimize 一條 regression line 使得 total distance 會最小, 我們不得不去遷就那顆 outlier 而整條線會變得比較不 fit 這樣的資料。(至於爲什麼整條線會偏趨得這麼厲害, 我們可以看上面的 error distance, 由於距離是由 data 點到 fit 點之間的距離, 所以對於 fit 好的點, cost會很低。然而

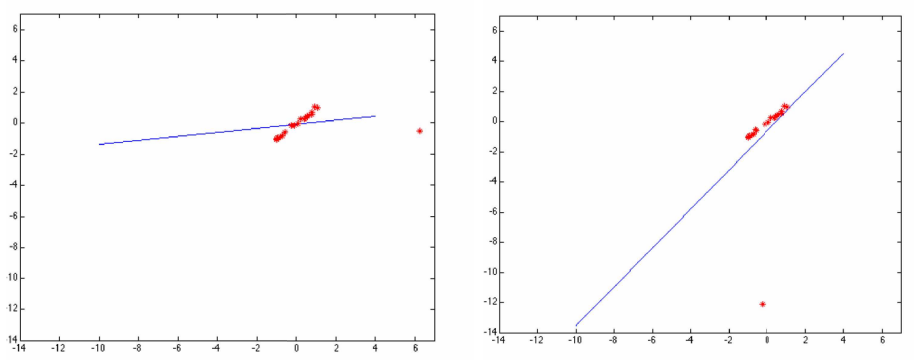

Figure 1: (a) and (b) the original mesh and simplified meshe.

fit 不好的點,cost 相對會高出很多, 因爲這裡是用平方的關係。因此在這樣的 model assumption 之下, 整個 fitting 會偏向 outlier 許多。) 爲了解決這樣的問題, 有一群數學家專門再研究一個領域的學問就叫做"Robust Statistics"。

簡單的來說 Robust Statistics 有兩個目標

1. Recover the BEST fit for the majority of the data
2. Detect and reject outliers

如果 Robust Statistics 做得好, 我們也許會得到像下面這張圖, 即使不能完全消除 outlier 的影響, 明顯的 fitting 情況會好很多。因此目標就是去設計一個 robust function 套用在剛剛我們看到的 error distance 上面。$Robust(error_distance)$這樣的 function 用來達到剛剛我門的兩個目標。首先如果發現 error distance 達到某些條件, 我們把它當成是 outlier, 因此把該 distance 做一個調整, 使得 outlier 影響降低。這裡列舉一些常用的 robust function。

2 Robust PCA 的目標

講完 Robust Statistics 以後我門來看這篇 paper 的主要內容。首先以往在做 PCA 的 paper, 對於 corrupted 的高維資料只能 reject 他或 accept 他。如圖 Figure 2(a) 裡面的鴨子、招財貓、杯子、小豬等。然而如我今天被 corrupted 的是整筆資料, 而是單筆資料裡面的某些維度 Figure 2(b), 例如說一張圖片裡面的某些點爛掉了, 這樣的資料我門要用也不是, 不用也不是。今天如果我們照一般的 PCA 去使用這樣的

資料而做出來的 principle components, 通常結果會很鳥, 也就是 GIGO(garbage in garbage out), 也許不至於到 garbage, 但是我們可以由下圖看到跑出來的 eigen vector 很爛, 很明顯的受到那些 outlier 的影響。注意, 這裡的 outlier 只存在於某些維度, 例如一張 200x200的40000維的圖片, 被 corrupted 的維度可能不到100維。然而我門還是發現這些 courrupted 資料對 fitting 影響很大。

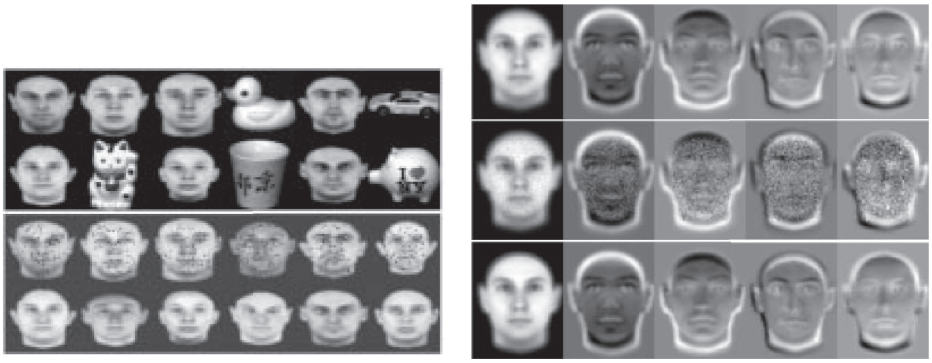

Figure 2: (a) 兩種不同的 outlier, 上半部爲整筆資料是錯誤的, 下半部爲 只有某些 pixel 不正確。(b) 第二排爲直接用普通的 PCA 去做分析, 第 三排爲這篇 paper 的結果。

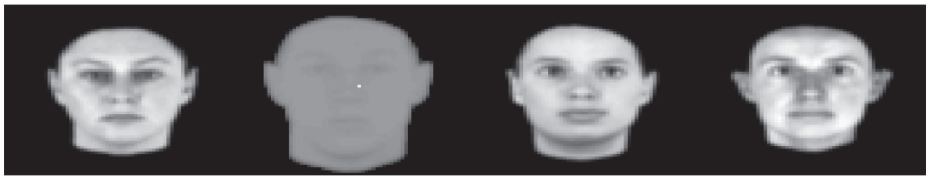

Figure 3: 極端的例子, 其中第二張臉的某個像素被courrupted。

這裡我們可以在看另一個極端的例子,Figure 3。現在有四張臉, 其中第二張臉只有一個 pixel 是 outlier, 可是他的偏離很誇張, 遠超過一般圖片的255色階, 所以這邊第二張圖用 log image 來表現, 其實只是那個點的 luminance 太大了, 是個 outlier。今天我們如果把這四張圖去跑 PCA, 我門會得到 Figure 4 結果,(在第一排) 很明顯的第一個 eigen vector 都跑去描述那個 outlier 了 (因爲 eigen value 超大), 如果我門把該筆資料 reject 掉, 那我們得到的資料就會比較少 (第二排)(很顯然的這樣的作法一點也不實際, 因爲我們不會只有一筆資料有 outlier), 而這篇 paper(第三排) 的目標就是把 robust function 套用在 PCA 的計算當中來去除這樣 outlier 的影響。

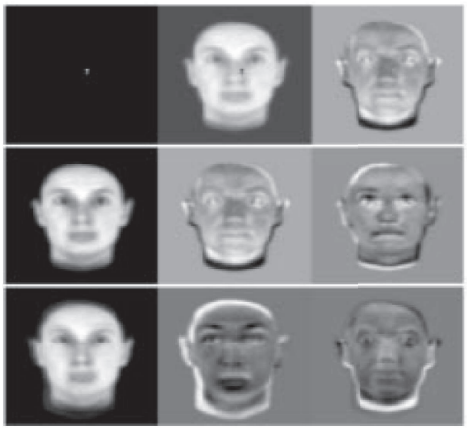

Figure 4: 極端的例子分別用Naive PCA、previous work、這篇 paper 的結果。

3 PCA回顧

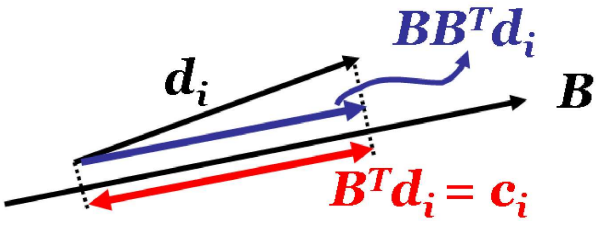

Figure 5: 圖示PCA 的 Dimension Reduction。

在描述完這篇 paper 的目標以後, 我們來看一下 PCA 的重點: Given以 Figure 5, 我門有一筆資料, 用 vector d_i來表示。今天他在平面上有兩個維度, 我們希望減少一個維度, 也就是所謂的"Parsimonious Representation", 節省的表達法。今天我們希望找到一個 vector B, 而我們只要記錄B這個二維的 vector以及資料在B上面的長度就好。如此一來原本紀錄 n 筆二維的資料用掉2*n 的容量, 現在只要用1*n + 2的容量。這就是所謂的 "Parsimonious Representation"。PCA 重點就是在於說找到一個B以及所有資料d_i在上面的投影 (還原過後的資料) 使得誤差會最少。在這裡投影過後 (還原) 的資料是$B * (B^T * d_i)$, 在此$B^T * d_i$是該筆資料在B上面的長度, 或叫c_i(Coefficients), 而在乘一個B是把該長度乘上一個方向還原成向量, 所以誤差就是 $(d_i - B * B^T * d_i)^2$, 而對於所有的資料我門希望 total 誤差最小, 因此在這裡我們又看到了所謂的 Least Square Fit, 且我門把 B稱之爲 Principle Component。

這樣的問題我們可以看成: 有一筆資料D, 我們希望找一組 B(Principle Component), 使的找出來的C(Coefficient) 用來做還原以後的資料($B * C$)會最逼近D。這樣的問題可以寫成數學式 $D = B * C$, 這也是 PCA 的 General Formulation。經過 Dimension Reduction 以後, 我們希望找出一組B'、C' 使得 $D \cong B'C'$, 且 dimension(C') < dimension(D)。在 Probabilistic PCA 的 model 之下, 這樣的問題是寫成 $D = B'C' + \sigma * I$, 也就是把 reduced 的 dimension 當成 error 去算。(註: 這裡的 error model 是 multivariate isotropic error model) 而 Probabilistic PCA 的標準解法是用 EM(Expectation Maximation) 去解。(不懂 EM 的人可以去看 Stanford Machine-Learning course note by Andrew Ng , 寫的還蠻清楚的。)

- E-step: $C = (B^T B)^{-1} B^T D$
- M-step: $B = DC^T (CC^T)^{-1}$

整體得目標就是在經過好幾回的 E-step 和 M-step 可以去 minimize Equation 1這個 error function.

$$\sum_{i=1}^{n} (d_i - Bc_i')^2 \tag{1}$$

4 Previous Work

對於 Robust PCA 來說, 一個 outlier 會對於剛剛的 total error 影響很大, 所以有人 (Xu and Yuille [1995]) 用了個方法去把整筆資料去考慮不是 outlier, 所以提出一種 cost function Equation 2來做 minimization:

$$\sum_{i=1}^{n} V_i (d_i - Bc_i')^2 + \eta(1 - V_i) \tag{2}$$

其中V_i是一個 binary variable, 代表說該筆資料是 (0)or 不是 (1) 一個 outlier。她們的 function 多了一個項$\eta(1 - V_i)$, 這是一個 panelty term 是用來說如果決定某筆資料是 outler, 那就必須付出一些代價, 但這代價應該會比把 outlier 當 inlier 的影響還要小。同時這個η可以用來預防把所有的資料都當成 outlier。(廢話, 如果所

有的資料都是 outlier 的話那就不用算了)。去解這樣的 Equation 2很麻煩, 主要在於說這個 minimization 包含了 contineous variable(B and c) 以及 discrete variable(V_i)。discrete optimation 的問題本來就很難解了, 今天再加上 contineous 的 case, 所以作者提出了一個很複雜的方法來解。不過說眞的, 她們最大的問題還是在 cost function formulation 不好, 使得即使解出來, 效用仍然不是很大。(也就是無法處理只有某些 dimension 是 outlier 得 case, 而這些 case 反而是最常見的)
之後有人提出另一種解法, 假設我們可以對於每一筆資料 i 的每一個維度 p 給予一個權重w_{pi}, 則我們可以仍然沿用最簡單的 Equation 1。同樣的,EM-Algorithm 可以把這個 minimization (Equation 3) 解得很好。

$$\sum_{i=1}^{n}\sum_{p=1}^{d} w_{pi} * (d_{pi} - B * ci)^2 \tag{3}$$

這個式子最大的問題在於他只能用在低維度的 principle component estimation, 主要的困難來自於對於每一筆資料的每一個維度都必須提供一個權重, 對於有上萬維度的影像資料, 這樣的方法不但不切實際, 也不可能。事實上這樣的方法最早被提出來的是在做矩陣運算的期刊上, 通常該領域面對的資料維度都比較小。

5 Robust Principle Component Analysis

雖然 Equation 3的解法不切實際, 不過卻激發了這篇 paper 基本的想法, 用 Robust Function ρ 去取代w_{pi}! 再這裡要注意的一點是, 由於我有些資料的某些 dimension 不準確, 因此我們在算 mean vector 實有可能根本就是錯的, 因此與其把 mean vector 當成已知值, 不如把它當成是一個待估計的量, 希望可以藉由估計 eigen-vector (Principle Component) 時, 同時也把正確的 mean vector 估計出來。

$$\rho(x, \sigma) = \frac{x^2}{x^2 + \sigma^2}$$

$$\sum_{i=1}^{n}\sum_{p=1}^{d} \rho((d_{pi} - \mu_p - b_p * c_i), \sigma_p) \tag{4}$$

在以上的 Equation 4, 我們需要估計的變數分別有μ 、B、C。另外σ_p是一個使用者定義的參數, 用來控制有關判別 outlier程度的參數 (這個參數要描述很複雜, 不過

是作者 heuristically 條出來的, 所以也不是這麼重要, 要了解的再於: 它主要是根據一個 pixel 跟周圍點的 median 差距所求出來的。所以一個點越是 outlier, 那麼他跟周圍的點差距也越大, 則σ也會越大, 則根據 Robust Function ρ, 該點的影響會越小!)。這個式子很明顯的, 需要同時估計三個變數的值, 並且互相是相依的, 因此同時估計其實是不可能的, 因此作者使用了 EM-like 的方法去估計, 也就是三個之中先 fix 兩個, 估計其中一個; 換另外兩個 fix, 估計剩下的 ,不斷的循環, 直到收斂爲止。但另一個問題是已經決定估計某個變數以後, 剩下的就是要如何估計, 在這裡作者對 Equation 4做了另一個假設, 也就是說雖然 Equation 4看起來很複雜, 但我們可以用 quadratic function 去 locally fit 他 (雖然聽起來很爛, 不過也莫可奈何), 所以我們可以對於該變數找偏微分求極值 (Gradient Desent Method), 然後把變數定在該極值, 再去求其他變數的極值。因此這裡我們分別要找三個變數對於 Equation 4做偏微分, 再用 Gauss-Newton 法求極值。(什麼, 不知道高斯牛頓法, 快翻翻微積分課本巴!) 其餘的細節 paper 上都有講, 在這裡我們就講概念而已。

最後這篇 paper 報告他的執行速率, 這裡他用了256張臉的圖片 (120x160), 最後算出 20個 principle components, 使用 Matlab 在 PentinumIII 上跑, 不過卻花了3個小時才算完, 一部份會這麼慢當然是因爲 EM-Algorithm 本身就慢, 不過我想最重要的應該是在於 convergence 的問題, 因爲這邊的解法是假設 function 可以用 locally quadratic 去 fit。因此找一個更好的方法或許會有改善的機會。