Fisher Face

人臉辨識的研究中最有名的做法就是使用 *PCA*，利用 *PCA* 找出最能代表不同人臉的特徵，也就是所謂的 *Eigen face*。雖然使用 *PCA* 來做辨識，有著不錯的辨識率，但因爲 *PCA* 並不單獨考慮每個人自己的人臉特性以及不同人的人臉差異，而是用全部的人臉一起考慮來找出轉換矩陣(transformation matrix)，此做法可能會導致不同的人臉影像降維後，分布會混在一起而不易辨識 (i.e. 在未降維前同一個人的人臉影像應該都會集中分布在一起)。爲了改進這個缺點，有人想到利用 *LDA* 取代 *PCA* 來作辯識，*LDA* 主要概念就是要找到一個 subspace，在這個 subspace 裡不同 class (i.e. 不同人) 的 instance (i.e. 人臉影像)在降維後彼此的距離愈大愈好，而同一個 class 的 instance 距離則是愈小愈好，所以直覺上使用 *LDA* 會比 *PCA* 有著更好的效果。

可是事情並沒有想像中簡單，使用 *LDA* 會有一些限制，首先我們先回憶一下解決 *LDA* 等價於解決下面式子所描述的 eigenvalue problem:

$$S_B w_k = \lambda_k S_w w_k$$

這裡有一個關鍵性的問題也就是 S_w^{-1} 是不是存在? 如果存在就可以將上面的式子轉換如下:

$$S_w^{-1} S_B w_k = \lambda w_k$$

也就能可用傳統的 eigenvalue 解法解決這個問題，但如果 S_w^{-1} 不存在則可能會導致無解。

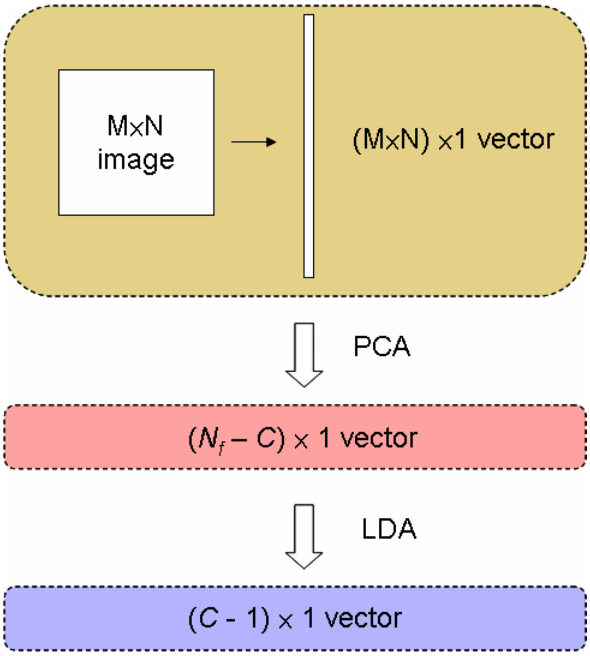
假設總共有 N_f 人臉影像，這些影像分屬於 c 個不同的 class，μ_i 爲第 i 個 class 的 mean，則依照之前 LDA tutorial 的定義，within-class matrix ($S_w \in \mathbb{R}^{d\times d}$) 計算如下:

$$S_w = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T$$

因爲 $(x_k - \mu_i)(x_k - \mu_i)^T$ 爲向量的外積，故 rank 爲 1，第 i 個 class 所貢獻的 rank 總共是 N_{fi} -1 (i.e.因爲最多只有 N_{fi} -1 獨立的 term)，所以 S_w 的 rank 爲 $N_f - c$。

$$(N_{f1} - 1) + (N_{f2} - 1) + \cdots + (N_{fc} - 1) = N_f - c$$

在實際的例子中，S_w 的 dimension 幾乎都遠大於 instance 的個數 (i.e.例如 320×240 解析度的人臉影像就需要超過 7 萬 6 千多張影像，現實生活中是不存在影像個數如此大的 dataset)，所以很遺憾地 S_w^{-1} 是不存在的，不過聰明的人又想到，那就先利用 *PCA* 降低人臉影像的維度低於 $N_f - c$ 再使用 *LDA*。所以應用 *LDA* 於 face recognition 的流程如下圖:



圖一. Fisher face recognition 流程圖

而最佳的轉換矩陣則透過解決下面的問題就可以得到

$$W_{opt}^T = W_{fld}^T W_{pca}^T,$$

where

$$W_{pca} = \arg\max_W |W^T S_T W|$$

$$W_{fld} = \arg\max_W \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|}$$

W_{pca} 和 W_{fld} 都可以用求解 eigenvalue 的演算法來得到，而 W_{opt}^T 的每個 row 也就是所謂的 Fisher face。

2DLDA (Two Dimensional Linear Discriminant Analysis)

Introduction

傳統的 *LDA* 只能針對一維向量去做降維，如果想要將 *LDA* 應用在二維的影像上，必須先將二維的影像轉換成一維的向量，但這樣的做法會使得影像中的結構資訊遺失(例如人臉影像會因此被切成許多不連續的片斷)，同時因爲一維向量的維度爲($M\times N$) × 1 所以在計算 *LDA* 的 *within-class matrix* 和 *between-class matrix* 過程中會產生($M\times N$) × ($M\times N$)的超高維矩陣，要對維度如此巨大的矩陣去求解 eigenvector 和 eigenvalue，勢必需要可觀的時間與空間，這也是傳統的 *LDA* 不適用於較大解析影像的原因。*2DLDA* 直接對二維影像做降維，而不需要轉換成向量，同時能改善前述傳統 *LDA* 應用於影像的缺點:

1. 不會破壞影像的結構而且更能利用影像裡的資訊
2. 能夠大量減少計算的時間和空間複雜度

Main Idea

傳統的 *LDA* 的目標就是找到一個轉換矩陣(transformation matrix) W 將高維向量投影至較低維度的樣本空間(feature space)，而 *2DLDA* 的對象是一個二維的影像(可視爲一個二維矩陣)，所以，*2DLDA* 的目標就是將高維的矩陣轉換成一個較低維的矩陣。

Formulation

爲了達到上述的目標，*2DLDA* 引進了 tensor product，其概念爲假設有一個($\ell_1\times\ell_2$)維的向量空間 $\mathcal{L}\otimes\mathcal{R}$，它是$\mathcal{L}$和$\mathcal{R}$的這兩個向量空間的 tensor product，其中，其中$\mathcal{L}$的基底爲$\{u_i\}_{i=1}^{\ell_1}$而$\mathcal{R}$的基底爲$\{v_i\}_{i=1}^{\ell_2}$，這時我們可以定義兩個矩陣如下

$$L=[u_1,\ldots,u_{\ell_1}]\in\mathbb{R}^{r\times\ell_1}$$

$$R=[v_1,\ldots,v_{\ell_2}]\in\mathbb{R}^{c\times\ell_2}$$

矩陣 L 和 R 分別由向量空間$\mathcal{L}$和$\mathcal{R}$的基底所構成。如果現在有一個 $X\in\mathbb{R}^{r\times c}$ 的二維矩陣，它在 $\mathcal{L}\otimes\mathcal{R}$ 的投影會是 $L^TXR\in\mathbb{R}^{\ell_1\times\ell_2}$ 。

利用這個概念，我們主要的問題就是去找兩個轉換矩陣 L 和 R 分別對目標矩陣

(我們輸入的影像)的行和列做降維。目標問題描述如下，假設有 n 張影像 $A_i \in \mathbb{R}^{r\times c}, i=1,...,n$ ，我們想要找到 $L \in \mathbb{R}^{r\times \ell_1}$ 和 $R \in \mathbb{R}^{c\times \ell_2}$ 使得 $B_i=L^T A_i R$ ，B_i 爲降維後的影像，其維度爲 $B_i \in \mathbb{R}^{\ell_1\times \ell_2}$ 。

Solution

如同傳統的 *LDA*，我們以經知道每張影像所屬的 class 爲何(例如我們有一張人臉影像，我們已經知道這張影像中的人臉是屬於那個人)，假設有 n 張影像 $A_i \in \mathbb{R}^{r\times c}, i=1,...,n$ ，k 個 class 其符號爲 $\prod_1,...,\prod_k$ ，每個 class 的平均爲 $M_i = \frac{1}{n_i}\sum_{X\in\prod_i} X$ ，所有影像的平均 $M = \frac{1}{n}\sum_{i=1}^{k}\sum_{X\in\prod_i} X$ ，之後仿照傳統 LDA 我們可以定義 *within-class distance* 爲

$$D_w = \sum_{i=1}^{k}\sum_{X\in\prod_i} \| X - M_i \|_F^2$$

between-class distance 爲

$$D_b = \sum_{i=1}^{k} n_i \| M_i - M \|_F^2$$

這邊所採用的 matrix norm 爲 *Frobenius norm*，其算式如下

$$\| A \|_F = \sqrt{\sum_{i=1}^{r}\sum_{j=1}^{c} a_{ij}^2}$$

因爲 $trace(MM^T) = \| M \|_F^2$ 所以我們可以改寫 *within-class distance* 和 *between-class distance* 如下:

$$D_w = trace(\sum_{i=1}^{k}\sum_{X\in\prod_i} (X - M_i)(X - M_i)^T)$$

$$D_b = trace(\sum_{i=1}^{k} n_i (M_i - M)(M_i - M)^T)$$

同理假設已知 L 和 R 我們可以算出降維後的 *within-class distance* 和 *between-class distance* 如下

$$\widetilde{D}_w = trace(\sum_{i=1}^{k}\sum_{X\in\prod_i} L^T (X - M_i) RR^T (X - M_i)^T L)$$

$$\widetilde{D}_b = trace(\sum_{i=1}^{k} n_i L^T (M_i - M) RR^T (M_i - M^T) L)$$

如果我們想要達到與 *LDA* 同樣的效果，最大化每個 class 之間的距離最小化同一個 class 中每張影像的距離，那我們必須去最佳化 *L* 和 *R*，但是這兩個轉換矩陣都是未知，所以不可能同時最佳化 ，所以折衷的作法是每次固定一個變數如 *L* 或 *R* 然後去對另一個變數做最佳化，接著反覆這個步驟直到得到滿意的結果。因爲 2DLDA 的做法是 iterative optimization，所以除非迴圈數無限大否則 *2DLDA* 的解沒有 close form。

(1) 計算 *L*:

首先固定 *R*，這邊我們再定義兩個矩陣

$$S_w^R = \sum_{i=1}^{k} \sum_{X \in \Pi_i} (X - M_i) R R^T (X - M_i)^T$$

$$S_b^R = \sum_{i=1}^{k} n_i (M_i - M) R R^T (M_i - M)^T$$

所以 $\widetilde{D}_w$ 和 $\widetilde{D}_b$ 可以簡化爲

$$\widetilde{D}_w = trace(L^T S_w^R L)$$
$$\widetilde{D}_b = trace(L^T S_b^R L)$$

爲了讓 $\widetilde{D}_w$ 和 $\widetilde{D}_b$ 最佳化，我們必須去求解 $\max_L trace((L^T S_w^R L)^{-1}(L^T S_b^R L))$ ，而這個式子等同於去求解一個 generalized eigenvalue problem $S_w^R x = \lambda S_b^R x$，得到 eigenvector 先對 eigenvalue 做排序越大得排越前面，之後取前 ℓ_1 個 eigenvector 當作 *L*。

(2) 計算 *R*:

我們先固定 *L*，這裡我們也定義兩個矩陣

$$S_w^L = \sum_{i=1}^{k} \sum_{X \in \Pi_i} (X - M_i)^T L L^T (X - M_i)$$

$$S_b^L = \sum_{i=1}^{k} n_i (M_i - M)^T L L^T (M_i - M)$$

因爲 trace(*AB*) = trace(*BA*)，所以 $\widetilde{D}_w$ 和 $\widetilde{D}_b$ 可以化簡成

$$\widetilde{D}_w = trace(R^T S_w^L R)$$
$$\widetilde{D}_b = trace(R^T S_b^L R)$$

同理我們可以跟計算 *L* 的時候求解類似的 generalized eigenvalue problem，這裡我

們取前ℓ_2個 eigenvector 當作 R。

這邊要特別注意在計算 L 和 R 的過程中，*within-class matrix* (S_w^L和S_w^R)和 *between-class matrix* (S_b^L和S_b^R)其維度最多爲$r\times r$ 或$c\times c$已經遠比傳統 *LDA* 的 $(r\times c)\times(r\times c)$小了許多，這也是 *2DLDA* 在計算上優於 *LDA* 的地方

Algorithm

Algorithm 2DLDA$(A_1,\cdots,A_n,\ell_1,\ell_2)$
Input: $A_1,\cdots,A_n,\ell_1,\ell_2$
Output: $L,R,B_1,\cdots,B_n$

1. Compute the mean M_i of ith class for each i as $M_i=\frac{1}{n_i}\sum_{X\in\Pi_i}X$;
2. Compute the global mean $M=\frac{1}{n}\sum_{i=1}^{k}\sum_{X\in\Pi_i}X$;
3. $R_0\leftarrow(I_{\ell_2},0)^T$;
4. For j from 1 to I
5. $S_w^R\leftarrow\sum_{i=1}^{k}\sum_{X\in\Pi_i}(X-M_i)R_{j-1}R_{j-1}^T(X-M_i)^T$,
   $S_b^R\leftarrow\sum_{i=1}^{k}n_i(M_i-M)R_{j-1}R_{j-1}^T(M_i-M)^T$;
6. Compute the first ℓ_1 eigenvectors $\{\phi_\ell^L\}_{\ell=1}^{\ell_1}$ of $\left(S_w^R\right)^{-1}S_b^R$;
7. $L_j\leftarrow\left[\phi_1^L,\cdots,\phi_{\ell_1}^L\right]$
8. $S_w^L\leftarrow\sum_{i=1}^{k}\sum_{X\in\Pi_i}(X-M_i)^TL_jL_j^T(X-M_i)$,
   $S_b^L\leftarrow\sum_{i=1}^{k}n_i(M_i-M)^TL_jL_j^T(M_i-M)$;
9. Compute the first ℓ_2 eigenvectors $\{\phi_\ell^R\}_{\ell=1}^{\ell_2}$ of $\left(S_w^L\right)^{-1}S_b^L$;
10. $R_j\leftarrow\left[\phi_1^R,\cdots,\phi_{\ell_2}^R\right]$;
11. EndFor
12. $L\leftarrow L_I$, $R\leftarrow R_I$;
13. $B_\ell\leftarrow L^TA_\ell R$, for $\ell=1,\cdots,n$;
14. return$(L,R,B_1,\cdots,B_n)$.

特別注意，演算法中的第三步I_{ℓ_2}爲($\ell_2\times\ell_2$)維的單位矩陣，而第四步的 I 爲預先設定好的 iteration 數可以根，而其他步驟前面都有說明過，所以這邊不再贅述。

Conclusion

2DLDA 是一個快速而且省空間的降維方法，比傳統 *LDA* 更適合處理二維以及大量的資料。