UAP 3 - PCA

1 – Analyse en composantes principales

PCA est une méthode utilisée pour l'identification et la reconnaissance des formes. Elle permet d'exprimer les données tout en distinguant entre leurs similarités et leurs différences. Etant donné que le problème de la reconnaissance des formes peut devenir de plus en plus difficile notamment lorsque les données (images) sont de très grandes dimensions, PCA peut être vu comme un outil très puissant pour analyser les données du fait qu'elle opère en réduisant leurs dimensions de façon considérable. L'autre utilisation avantageuse de PCA est qu'en réduisant leurs dimensions, les données peuvent être compressés sans perdre beaucoup d'information utile. L'algorithme PCA fut utilisé pour la première fois dans le domaine de la reconnaissance du visage par M. Turk et A. Pentland en 1991 au MIT MediaLabs. Il est également connu sous le nom d'eigenfaces, nommé d'après l'utilisation des valeurs propres et des vecteurs propres (respectivement eigenvalues et eigenvectors en anglais). Aujourd'hui, près de deux décennies après son invention et en dépit de ses problèmes, PCA reste certainement l'algorithme de reconnaissance le plus connu et le plus couramment évoqué lors des comparaisons entre les algorithmes.

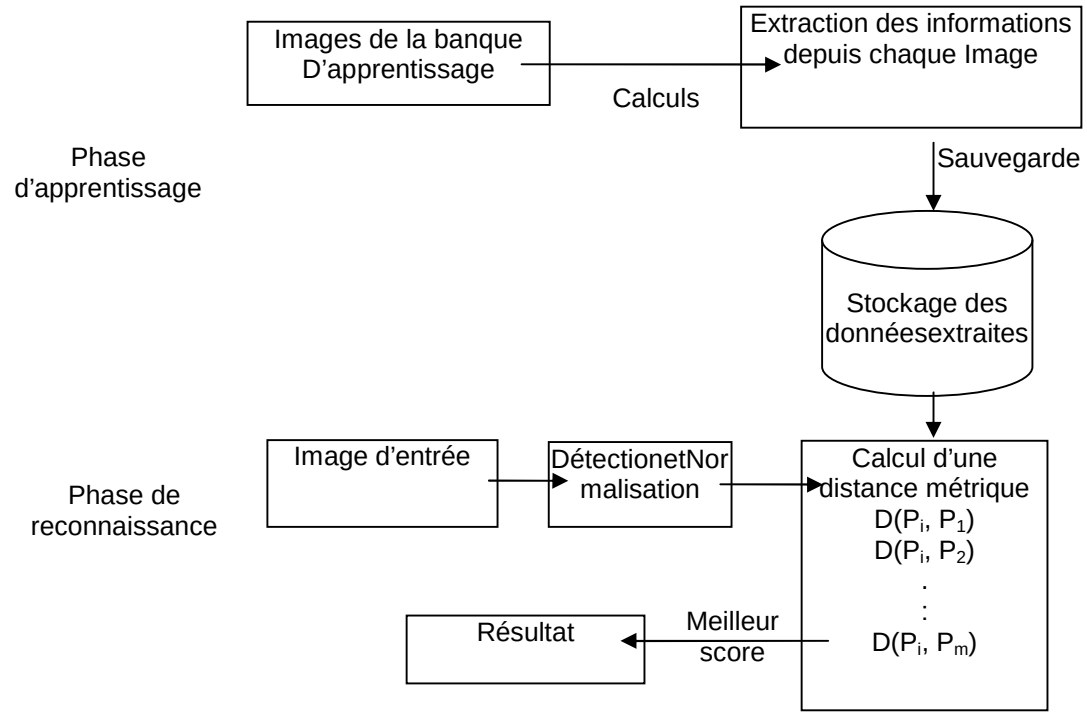

FIG.3.5- déroulement d'un algorithme de reconnaissance

L'idée principale de PCA est d'extraire de l'information depuis une image d'un visage, l'encoder aussi efficacement que possible et la comparer àdes images traitées préalablement de la même façon et stockées dans une base de données. Comme tout

algorithme de reconnaissance, le processus de PCA est divisé en deux phases distinctes : phase d'apprentissage et phase de reconnaissance.

1 – 1 Apprentissage

Dans cette phase, M images (pour M individus) de largeur w et de hauteur h seront considérés comme des vecteurs de taille w x h et non plus comme des matrices. Les images étant en niveaux de gris, chaque pixel peut prendre une valeur entre 0 et 255. Ces vecteurs Γ_i ($1 \leq i \leq M$) sont concaténés pour en faire une seule matrice Γ(w x h, M) de w x h lignes et M colonnes (figure.3.6).

$$\Gamma = \begin{bmatrix} a_{1,1} & b_{1,1} & \cdots & z_{1,1} \\ \vdots & & \ddots & \vdots \\ a_{1,w} & & & \vdots \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ a_{h,1} & & \cdots & \vdots \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ a_{h,w} & & & z_{h,w} \end{bmatrix}$$

Fig.3.6- Matrice globale contenant l'ensemble des images pour l'entraînement

Soustraction de la moyenne

L'image moyenne Ψ (i.e. vecteur moyen) est calculée puis soustraite de toutes les images :

$$\Psi = \frac{1}{M}\sum_{i=1}^{M} \Gamma_i \qquad (2.1)$$

$$\Phi_i = \Gamma_i - \Psi \ (i = 1,2,..M) \qquad (2.2)$$

FIG.3.7- Image Moyenne

Calcul de la matrice de covariance

Dans cette étape la matrice de covariance du jeu de données est calculée selon la formule suivante:

$$C = \frac{1}{M}\sum_{i=1}^{M}\Phi_i\Phi_i^T = AxA^T \qquad (2.3)$$

$$\Phi = [\Phi_1\Phi_2 ....\Phi_M] \qquad (2.4)$$

Calcul des vecteurs propres et valeurs propres

Dans cette étape les vecteurs u_i et les valeurs propres associées v_i de la matrice *C*sont calculés. Sur le plan pratique, ce calcul peut parfois s'avérer très long et très gourmand en mémoire et en temps processeur. La matrice *C* est en effet une matrice de taille (w x h, w x h) (de l'ordre de la résolution de l'image). Avec un nombre M d'images inférieur à la résolution (w x h), il y aura seulement *M* vecteurs propres qui contiennent de l'information (les autres vecteurs propres auront des valeurs propres associées nulles).

Pour résoudre ce problème on utilise la solution proposée par M. Turk et A. Pentland. Pour un vecteur propre v_i associé à une valeur propre λ_i nous avons :

$$Cv_i = \lambda_i v_i \qquad (2.5)$$

La matrice C est de la forme AA^T. Considérons la matrice $L = A^TA$ ayant les vecteurs propres $u_i$associés aux valeurs propres e_i:

$$Lu_i = e_iu_i$$

Soit : $A^TA\,u_i = e_iu_i$

En multipliant à gauche par A les deux cotés de l'égalité, nous obtenons :

$$AA^TAu_i = Ae_iu_i \qquad (2.6)$$

Et puisque $C = A\,A^T$ nous pouvons simplifier (2.6):

$$CAu_i = Ae_iu_i$$

$$C(Au_i) = e_i(Au_i)$$

D'après la définition des vecteurs et valeurs propres de la matrice C

$$v_i = Au_i \quad , \quad \lambda_i = e_i$$

La matrice L est une matrice M x M et le calcul de ses vecteurs et valeurs propres est beaucoup plus aisé qu'avec la matrice C. Nous passons en effet d'une complexité de l'ordre de la résolution de l'image à une complexité de l'ordre du nombre d'images. Le gain de ressources est certainement non négligeable.

Ensuite il y a l'étape de sélection des vecteurs propres où une grande partie de l'efficacité de PCA consiste à n'en sélectionner que les *M' (M' < M)* associés aux valeurs propres les plus grandes (ceux associés aux valeurs propres les plus petites ne contiennent que très peu d'information utile).

A partir de là, un nouvel espace vectoriel $E_{v,}$ appelé espace de visage (facespace), est engendré par les M' vecteurs propres retenus.

La représentation des vecteurs propres rappelle des images fantômes chacune mettant en avant une partie du visage [31], on les appelle eigenfaces (figure 3.3).

Dans la dernière étape de la phase d'apprentissage, les images de départ sont projetées sur l'espace des visages; chaque image Γ_i est alors transformée en ses composantes eigenfacesw_k par une simple opération de projection vectorielle:

$$W_k = e_k^T (\Gamma_i - \Psi) \quad (1 \le k \le M') \quad (2.7)$$

Les$w_k$sont appelés poids (weights) et forment un vecteur Ω^T :

$$\Omega^T = [w_1, w_2, w_3, ..., w_{M'}] \quad (2.8)$$

Les vecteurs Ω^T sont conservés afin d'être utilisés pour classifier une nouvelle image lors de la phase de reconnaissance.

1 – 2 Reconnaissance

Le processus d'affectation d'une nouvelle image Γ_i à une classe issue de la phase d'entraînement procède en deux étapes: en premier, l'image Γ_i est transformée en ses composantes eigenfaces selon la formule (2.7).$\Omega^T_{new} = [w_1, w_2, w_3, ..., w_{M'}]$

FIG.3.8- Image moyenne et 24 eigenfaces

Ensuite, la classe de visage fournissant la meilleure description de $\Gamma_{new,}$ est déterminée en calculant la distance minimale entre le vecteur Ω^T_{new} nouvellement créé et ceux stockés dans la base de données. La métrique la plus souvent utilisée est la distance euclidienne donnée par:

$$d(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (2.9)$$

Dans la pratique, l'espace de visages *E* est transformé en un espace de Mahalanobis en divisant chaque vecteur propre sur la racine carrée de la valeur propre qui lui est associée [31]. Cette méthode offre des taux de reconnaissance nettement supérieurs. Un visage Γ_{new} appartient à une classe *k* quand le minimum de la distance entre Ω_{new} et Ω_k ($1 \leq k \leq M$) est en dessous d'un certains seuil (threshold) θ, Dans le cas contraire le visage est considéré comme inconnu et peut éventuellement être utilisé pour créer une nouvelle classe.

1 – 3 Enrôlement d'une nouvelle personne

L'ajout d'une nouvelle personne à la base d'apprentissage peut s'effectuer par une simple projection de la nouvelle image sur l'espace de visage E_v cette approche peut être retenue lorsque la banque d'apprentissage est relativement grande et que les visages s'y trouvant sont représentatifs.

Cependant, il serait intéressant à long terme (après plusieurs ajouts selon la méthode qu'on vient de décrire) de réeffectuer la phase d'apprentissage complète et générer un nouvel espace de visage en prenant en compte les nouvelles images, cette méthode plus coûteuse permet d'obtenir des visages propres plus représentatifs de la base.

1 – 4 Ressources requises

On a vu que lors de la phase de reconnaissance, une nouvelle image est projetée sur l'espace de visages et son vecteur de poids Ω^T est comparé à d'autres vecteurs de la base de données. Afin d'accélérer le processus de calcul, les vecteurs propres et les vecteurs de poids Ω^T doivent être présents en permanence en mémoire centrale.

Pour une banque d'apprentissage contenant 1000 images de 128 x 96 pixels, la conservation des 200 premiers vecteurs propres nécessite 200 x 128 x 96 x 4 = 9,375 Mo d'espace mémoire (pour des réels à simple précision), une représentation d'une image par son vecteur Ω^T requiert quant à elle un vecteur de 200 éléments (soit 1000 x 200 x 4 ≈ 780 Ko).

1 – 5 Discussions

PCA est un algorithme incontournable du monde de la reconnaissance du visage. Il est souvent étudié en premier pour acquérir des bases solides dans le domaine. D'un point de vue technique, PCA utilise des pixels d'une image en niveaux de gris mais on verra par la suite qu'on peut se servir des coefficients DCT au lieu des pixels. Le principe selon lequel on peut construire un sous espace vectoriel en ne retenant que les meilleurs vecteurs propres, tout en conservant beaucoup d'informations utiles, fait de PCA un des algorithmes les moins exigeants en ressources et il est couramment utilisé en réduction de dimensionnalité où il peut être utilisé en amont avec d'autres algorithmes (i.e. LDA, HMM). Toutefois sa simplicité de mise en œuvre contraste souvent avec une forte sensibilité aux changements d'éclairement, de pose, et d'expression faciale.