Projet – Authentification faciale

UV ODATA – IMT Lille Douai

1 Contexte et objectifs

L'authentifaction [1] (*authentication*) consiste à déterminer si une personne (ou une autre entité, par exemple un système informatique) a les privilèges requis pour accéder à un service donné : il peut s'agir par exemple d'autoriser l'accès à un système informatique (site web, serveur de données, smartphone...) ou à un bâtiment ou une partie d'un bâtiment (zone "privée" d'un espace public, zone sécurité défense, accès à un spectacle...). Un système d'authentification fonctionne typiquement de la manière suivante :

1. Dans un premier temps, les utilisateurs sont enregistrés dans le système, qui conserve une trace spécifique à un utilisateur ou un groupe d'utilisateurs (par exemple, un mot de passe).
2. Quand un utilisateur se présente pour accéder à la ressource à accès restreint, il présente une preuve de son identité (le mot de passe).
3. Le système vérifie si la preuve d'identité présentée correspond à celle d'un utilisateur (ou groupe d'utilisateurs) enregistré. Si c'est le cas, l'accès à la ressource est autorisée ; dans le cas contraire cet accès est refusé.

On parle d'authentification biométrique quand la preuve d'identité présentée par l'utilisateur est une de ses caractéristique physique (en principe, unique à l'individu) : visage, iris de l'œil, empreinte palmaire ou digitale...

Dans ce projet, on s'intéresse à l'authentification d'un utilisateur à partir de son visage. L'objectif est de mettre en place et d'évaluer un système d'authentification automatique dans lequel les utilisateurs sont identifiés par une photo de leur visage. Le système contiendra un ensemble de photographies de visages correspondant aux utilisateurs enregistrés (une ou plusieurs photographies par utilisateur). L'authentification consistera à vérifier si le visage de l'utilisateur demandant l'accès correspond bien au visage d'un utilisateur enregistré dans la base de référence.

2 Formalisation du problème

Le problème consiste à déterminer, étant donné un ensemble d'images de visages de référence (appelé *gallery*), si une image requête (appelée *probe*) n'appartenant pas à cette base correspond à une personne représentée dans la base de référence [2]. On cherche donc à établir une fonction d'authentification aut(.,.) définie comme suit :

$$\begin{aligned} \text{aut} : \mathbb{I} \times \mathbb{I}^n &\to \mathbb{B} \\ (I_q, \mathcal{I}) &\mapsto \begin{cases} \text{Vrai si l'identité de } I \text{ est présente dans } \mathcal{I} \\ \text{Faux sinon} \end{cases} \end{aligned}$$

1. `https://fr.wikipedia.org/wiki/Authentification`
2. Ceci est une variante de la vérification de visages (*face verification*), qui consiste à identifier si deux images de visages correspondent à la même personne.

où $\mathbb{I}$ est l'ensemble de définition des images, I_q est l'image requête testée et $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ est l'ensemble des n images de référence enregistrées dans le système.

Pour définir cette fonction, deux hypothèses seront émises :

1. Une image de hauteur h et de largeur w peut se représenter sous forme d'un vecteur de $\mathbb{R}^d$, avec $d = h.w$, soit $\mathbb{I} = \mathbb{R}^d$ (voir Section 8.1).
2. Deux images I et $I' \in \mathbb{R}^d$ représentent une même personne si la distance qui les sépare est inférieure à un seuil r donné : $\delta(I, I') \leq r$.

Dans ce cas, le problème de l'authentification se ramène à une recherche dans $\mathcal{I}$ des voisins de l'image requête I_q dans un rayon r (*radius search*). Si de tels voisins existent, alors la personne est enregistrée dans la base et l'accès autorisé, sinon l'accès est refusé. La fonction d'authentification devient :

$$\begin{aligned} \text{aut} : \mathbb{R}^d \times \mathbb{R}^{n \times d} &\to \mathbb{B} \\ (I_q, \mathcal{I}) &\mapsto |\text{radius\_search}(I_q, \mathcal{I}, r)| > 0 \end{aligned}$$

avec radius_search(.,.,.) la fonction classique de recherche dans un rayon r :

$$\begin{aligned} \text{radius\_search} : \mathbb{R}^d \times \mathbb{R}^{n \times d} \times \mathbb{R} &\to \mathbb{R}^{m \times d} \\ (I_q, \mathcal{I}, r) &\mapsto \{I \in \mathcal{I} \mid \delta(I_q, I) \leq r\} \end{aligned}$$

Ici, r est un paramètre de la fonction aut dont il conviendra de déterminer la valeur pour le jeu de données considéré. $\delta(.,.)$ est une distance entre vecteurs de $\mathbb{R}^d$; on prendra ici la distance euclidienne.

3 Réduction de la dimension des données

L'algorithme de recherche en force brute a cet inconvénient que l'on connaît déjà : son temps d'exécution est trop important quand le nombre n de données dans $\mathcal{D}$ ou leur dimension d est trop élevé(e). Dans l'application présente, n peut être très variable (nombre d'images enregistrées dans la base) mais d est nécessairement très grand (le nombre de pixels dans une image). Il n'est pas envisageable d'avoir un temps d'attente élevé avant d'autoriser un usager (par exemple, à l'entrée d'un bâtiment), il faut donc mettre en œuvre une méthode pour accélérer la recherche des voisins proches. Les méthodes d'indexation vues en cours permettent de limiter la complexité de la recherche relativement à n mais pas à d. Un approche alternative pour traiter les grandes dimensions consiste à effectuer une réduction de la dimension des données. L'analyse en composante principale (ACP) fournit un outil efficace pour réaliser cela. Dans le cas des images de visages, cette approche se nomme *Eigenfaces*.

La méthode Eigenfaces a été introduite (entre autres) par Turk et Pentland (MIT Medialab) en 1991 [3]. Elle permet (parmi d'autres applications) de réduire la dimension d'images à analyser en y appliquant une ACP, selon les étapes suivantes :

1. Linéariser les n images en vecteurs de $\mathbb{R}^d$ et les regrouper dans une matrice de dimension $n \times d$.
2. Centrer les données.
3. Effectuer une ACP sur la matrice des données centrées pour obtenir des composantes principales (vecteurs propres de la matrice de covariance). Ces composantes représentent les constituants "fondamentaux" des visages et sont nommées *Eigenfaces* (de *eigenvectors* – les vecteurs propres – et *faces* – les visages).

3. *Eigenfaces for Recognition.* M. Turk and A. Pentland. Journal of Cognitive Neuroscience, Vol. 3, Num. 1, 1991.

4. Projeter les images sur les k composantes principales les plus significatives pour obtenir une représentation en dimension $k < d$ des images.

Cette représentation dans $\mathbb{R}^k$ peut ensuite être utilisée à la place des images brutes dans le système précédent, qui peut donc traiter des données de dimension moindre. Le nombre de dimensions k à considérer dépend du compromis vitesse / performance souhaité. L'analyse des valeurs propres associées aux composantes principales permet aussi d'éclairer ce choix.

4 Évaluation du système

Le système retourne `Vrai` ou `Faux` selon que l'accès pour une requête donnée est autorisé ou non. Cependant, il n'existe pas de garantie que la réponse du système soit correcte : l'accès peut être autorisé à une personne non présente dans la base de donnée (si l'image fournie est suffisamment proche de visages enregistrés dans le système), ou refusé à une personne enregistrée dans le système (si la photo courante est trop différente des photos enregistrées). Ces erreurs peuvent provenir des données elles-mêmes ou du paramétrage du système (valeurs du rayon r et du nombre k de composantes principales retenues). Il est donc nécessaire d'évaluer les performances effectives du système.

Conditions de l'évaluation Le système doit gérer deux cas de figure : autoriser les utilisateurs enregistrés et refuser les utilisateurs non enregistrés. Il est donc nécessaire de posséder les données suivantes pour en effectuer l'évaluation :

— un ensemble d'images qui constitueront les images enregistrées dans le système (*gallery*).
— un ensemble d'images qui constitueront les requêtes de test (*probes*) soumises au système. Pour tester les deux cas de figure soumis au système, certaines de ces requêtes doivent correspondre à des personnes représentées dans la *gallery* ; les autres doivent correspondre à des personnes absentes de la *gallery*, auquel l'accès devrait être refusé.
— une vérité-terrain, qui indique pour chaque requête si l'accès doit lui être autorisé ou non ; les réponses retournées par le système seront comparées à cette vérité-terrain pour évaluer la qualité du système.

On remarquera ici deux points importants :

— les images requêtes doivent être différentes des images dans la *gallery*, sinon le système fait face au problème (beaucoup plus simple) de retrouver une copie d'un fichier ;
— quand le système utilise la méthode des *Eigenfaces*, les composantes principales des images doivent être calculées uniquement sur la *gallery*, car, en pratique, les *probes* sont inconnues lors de la constitution de la base d'utilisateurs enregistrés [4].

Mesures d'évaluation Lorsque le système fournit un résultat, il peut se trouver dans un des quatre cas suivants :

1. **Vrai positif** : le système autorise l'accès à un utilisateur auquel l'accès doit être autorisé ;
2. **Faux positif** : le système autorise l'accès à un utilisateur auquel l'accès doit être refusé ;
3. **Vrai négatif** : le système refuse l'accès à un utilisateur auquel l'accès doit être refusé ;
4. **Faux négatif** : le système refuse l'accès à un utilisateur auquel l'accès doit être autorisé.

Les vrais positifs et vrais négatifs correspondent à des réponses correctes du système ; les autres correspondent à des réponses incorrectes. Dans la suite, on notera respectivement TP, FP, TN

4. On fait ici l'hypothèse que l'ACP sera recalculée si l'on enregistre de nouveau utilisateurs. Si l'on souhaitait tester la capacité du système à enregistrer de nouveaux utilisateurs sans refaire ce calcul, il serait nécessaire d'utiliser un troisième jeu de données, différent de la *gallery* et des *probes*, sur lequel on calculerait les *Eigenfaces*. Ce cas n'entre pas dans le cadre de ce projet.

et FN le nombre de vrais positifs, faux positifs, vrais négatifs et faux négatifs fournis par le système sur l'ensemble des *probes*.

À partir des valeurs de TP, FP, TN et FN, on peut définir les mesures de performances suivantes :

- **Exactitude** : $A = \frac{TP+TN}{TP+FP+TN+FN}$ – Mesure le taux de bonnes réponses produites par le système.
- **Précision** : $P = \frac{TP}{TP+FP}$ – Mesure le taux de personnes correctement autorisées parmi l'ensemble des personnes autorisées par le système.
- **Sensibilité** (ou **rappel**) : $\text{Sen} = \frac{TP}{TP+FN}$ – Mesure le taux de personnes correctement autorisées parmi l'ensemble des personnes qui devraient être autorisées par le système.
- **Spécificité** : $\text{Spe} = \frac{TN}{TN+FP}$ – Mesure le taux de personnes correctement refusées parmi l'ensemble des personnes refusées par le système.

En faisant varier les paramètres du système, il est possible d'établir des courbes de performance illustrant le comportement du système dans différentes situations (par exemple, des courbes ROC). De nombreuses autres mesures complémentaires sont disponibles pour évaluer des systèmes de ce type. Ces mesures doivent être interprétées en fonction du contexte d'utilisation du système : par exemple, faut-il absolument refuser toutes les personnes non-autorisées, au risque de refuser aussi des personnes autorisées ?

Remarque 1 On retrouve ici des mesures d'évaluation classiques utilisées en classification supervisée (rappel et précision), ou en statistiques (sensibilité et spécificité).

Remarque 2 Contrairement à une simple de recherche de voisins proches, on ne cherche pas ici simplement à approcher une propriété mathématique sur les données (leur distance), mais à approcher une information sur le sens des données (l'identité de la personne sur la photographie). Il n'est donc pas possible ici d'établir une vérité-terrain automatiquement : il faut disposer des données adéquates indiquant l'identité associée à chaque image.

5 Données

Deux ensembles de données sont fournis pour le projet :

- le premier contient 7.462 images de visages de 373 personnes différentes ;
- le second contient 6.059 images de visages de 1.203 personnes différentes.

Dans les deux ensembles de données, les images sont des images en niveaux de gris de taille 150×150 pixels. L'identité de la personne associée à chaque image est indiquée. Les images ont été prétraitées pour le projet ; aucun processus de traitement d'images (autre que l'application de la méthode Eigenfaces) n'est attendu.

Les deux jeux de données sont disponibles dans l'archive `ODATA_project_authentication_data.zip` présente sur MyLearningSpace. Chaque jeu contient :

- une répertoire contenant les images ;
- un fichier `README.md` contenant les informations utiles sur les données.

6 Travail à effectuer

Prétraitement des données Pour chaque ensemble de données fourni :

1. Constituer un ensemble d'images de référence (*gallery*) et un ensemble de 200 requêtes de test (*probes*) tels que :
   - 100 des requêtes de test ont des images de la même identité dans la *gallery*, correspondant ainsi à des tests sur des utilisateurs enregistrés dans le système,

— 100 des requêtes de test n'ont aucune image de la même identité dans la *gallery*, correspondant ainsi à des tests sur des utilisateurs non enregistrés dans le système,
— toutes les images encore disponibles de l'ensemble de données sont utilisées pour constituer la *gallery*.

2. Constituer la vérité-terrain associée aux requêtes.

Système d'authentification par force brute Développer un système d'authentification reposant sur une recherche de voisins proches (*radius search*) par force brute.

Système d'authentification avec Eigenfaces Développer un système d'authentification reposant sur une recherche de voisins proches (*radius search*) par force brute sur des données dont la dimension a été réduite à l'aide de la méthode des Eigenfaces.

Expérimentations Mettre en place des expérimentations sur les deux jeux de données fournis pour :

1. Analyser numériquement l'impact des paramètres de chaque système sur la qualité de ses résultats.
2. Visualiser et analyser qualitativement les composantes principales obtenues par l'approche Eigenfaces.
3. Analyser et comparer les performances (qualité des résultats et vitesse d'exécution) des deux systèmes étudiés.

Le choix des mesures d'évaluation est libre ; il doit être fait en fonction des résultats qui seront mis en avant dans l'analyse.

7 Livrables

Les livrables à remettre à l'issue du projet sont les suivants :

1. Un rapport **au format PDF** contenant :
   — un rappel succint des objectifs du projet ;
   — une description succinte des systèmes mis en œuvre ;
   — une description du protocole expérimental mis en place (objectifs expérimentaux, données, mesures d'évaluation...) ;
   — une description des résultats quantitatifs et qualitatifs obtenus ;
   — une analyse des résultats.
2. Le code permettant de reproduire les expérimentations, suffisamment documenté.

8 Trucs et astuces

8.1 Images

Représentation des images Une image en couleurs est représentée informatiquement comme une matrice de pixels (pour *picture elements*), chaque pixel étant un triplet `(r, g, b)` dont les composantes indiquent la proportion de rouge, vert et bleu dans la couleur du pixel, respectivement [5] Selon la représentation considérée, `r`, `g` et `b` peuvent être des entiers à valeur dans $[0, 255]$ (entiers non signés sur 8 bits) ou des réels à valeur dans $[0, 1]$ (réels sur 32 ou 64 bits).

5. En supposant l'utilisation de l'espace de couleurs RGB pour représenter les couleurs. Cet espace est le plus couramment utilisé, mais il en existe beaucoup d'autres.

Images en niveau de gris Une image en niveaux de gris est représentée par une matrice dont les éléments contiennent une unique valeur (entier dans $[0, 255]$ ou réel dans $[0, 1]$) correspondant à l'intensité du pixel correspondant. Une image RGB est en pratique en niveaux de gris si les composantes `r`, `g` et `b` sont les mêmes pour tous les pixels.

Linéarisation des images Une image de hauteur h et de largeur w en niveaux de gris peut être représentée sous forme d'un vecteur de $\mathbb{N}^d$ ou $\mathbb{R}^d$ (avec $d = h.w$) en concaténant les colonnes (ou les lignes) de la matrice correspondante.

Manipulation des images en Python La bibliothèque `matplotlib` (entre autres) contient des fonctions de base pour lire, écrire et afficher des images. Ces fonctions sont suffisantes pour réaliser le projet. Elles manipulent les images sous forme de tableaux Numpy de dimensions (h, w, c), avec h la hauteur de l'image, w sa largeur et c le nombre de canaux ($c = 1$ pour les images en niveaux de gris, $c = 3$ pour les images RGB).

8.2 ACP efficace

Problématique Effectuer une ACP nécessite d'effectuer de calculer les vecteurs propres et valeurs propres de la matrice de covariance des d variables des données. Le calcul des valeurs et vecteurs propres d'une matrice carrée de taille $d \times d$ a une complexité en temps en $O(d^3)$. La matrice de covariance et celle des vecteurs propres ont une complexité en espace en $O(d^2)$. Dans le problème traité ici, d est très grand (nombre de pixels dans les images), ce qui rend le calcul des valeurs/vecteurs propres, et surtout le stockage en RAM des matrices associées, difficile, même sur une machine moderne. Pour réduire ces complexités, Turk et Pentland, dans leur article introduisant les Eigenfaces, proposent une méthode de calcul des valeurs propres pour le cas où le nombre d'individus (ici, d'images) et inférieur au nombre de variables (ici, le nombre de pixels par image).

Méthode de calcul efficace La méthode pour calculer efficacement les composantes principales d'une matrice de données est la suivante. Soit $\mathbf{D} \in \mathcal{M}(n, d)$ la matrice des données centrées (n individus, d variables) sur lesquelles on souhaite effectuer l'ACP.

1. Calculer la matrice de covariance de $\mathbf{D}^T$: $\text{Cov}(\mathbf{D}^T)$.
2. Calculer les vecteurs propres $\mathbf{v}_i$ de $\text{Cov}(\mathbf{D}^T)$.
3. Calculer composantes principales $\mathbf{w}_i$ de $\mathbf{D}$ (autrement dit, les vecteurs propres de $\text{Cov}(\mathbf{D})$) comme suit : $\mathbf{w}_i = \mathbf{D}^T.\mathbf{v}_i$.
4. Normaliser les $\mathbf{w}_i$ pour obtenir des vecteurs unitaires.

Justification de la méthode Les composantes principales de $\mathbf{D}$ sont les vecteurs propres de la matrice de covariance des données. Les données étant supposées centrées, la matrice de covariance de $\mathbf{D}$ est calculée comme suit :

$$\text{Cov}(\mathbf{D}) = \frac{1}{n-1}\mathbf{D}^T.\mathbf{D}$$

et a une taille de $d \times d$. Au lieu d'effectuer le calcul des valeurs propres et vecteurs propres sur cette matrice, on considère à la place la matrice de covariance de $\mathbf{D}^T$, qui vaut :

$$\text{Cov}(\mathbf{D}^T) = \frac{1}{d-1}\mathbf{D}.\mathbf{D}^T,$$

ce qui revient à inverser individus et variables. $\text{Cov}(\mathbf{D}^T)$ a une taille de $n \times n$. Les valeurs propres λ_i et vecteurs propres $\mathbf{v}_i$ de $\text{Cov}(\mathbf{D}^T)$ vérifient :

$$\text{Cov}(\mathbf{D}^T).\mathbf{v}_i = \lambda_i.\mathbf{v}_i$$

En multipliant par $\mathbf{D}^T$, on obtient :

$$\begin{aligned}
\mathbf{D}^T.\text{Cov}(\mathbf{D}^T).\mathbf{v}_i &= \mathbf{D}^T.\lambda_i.\mathbf{v}_i \\
\frac{1}{d-1}\mathbf{D}^T.\mathbf{D}.\mathbf{D}^T.\mathbf{v}_i &= \lambda_i.\mathbf{D}^T.\mathbf{v}_i \\
\frac{n-1}{d-1}\text{Cov}(\mathbf{D}).\mathbf{D}^T.\mathbf{v}_i &= \lambda_i.\mathbf{D}^T.\mathbf{v}_i \\
\text{Cov}(\mathbf{D}).\mathbf{D}^T.\mathbf{v}_i &= \frac{d-1}{n-1}\lambda_i.\mathbf{D}^T.\mathbf{v}_i
\end{aligned}$$

En notant $\mathbf{w}_i = \mathbf{D}^T.\mathbf{v}_i$ et $\mu_i = \frac{d-1}{n-1}\lambda_i$, on obtient :

$$\text{Cov}(\mathbf{D}).\mathbf{w}_i = \mu_i.\mathbf{w}_i$$

Autrement dit, les μ_i et $\mathbf{w}_i$ sont respectivement les valeurs propres et vecteurs propres de $\text{Cov}(\mathbf{D})$, qui est ce que l'on souhaitait obtenir.

Remarque Cette méthode produit n vecteurs propres au lieu des d vecteurs attendus. Cependant, puisque $n < d$, le rang de $\mathbf{D}$, et donc de $\text{Cov}(\mathbf{D})$, est (au plus) n, et donc $\text{Cov}(\mathbf{D})$ admet (au plus) n valeurs propres non-nulles. Les $(d-n)$ vecteurs propres "manquants" sont les vecteurs propres (non-significatifs) correspondant aux valeurs propres nulles de $\text{Cov}(\mathbf{D})$.