Daphné BISSONNIER
Constantin JOUNIOT

TP Noté : Vision par ordinateur

Pour ce TP, notre choix d'images se porte sur les 4 classes suivantes : ant, caméra, courgar_face, et Garfield.

1.3)

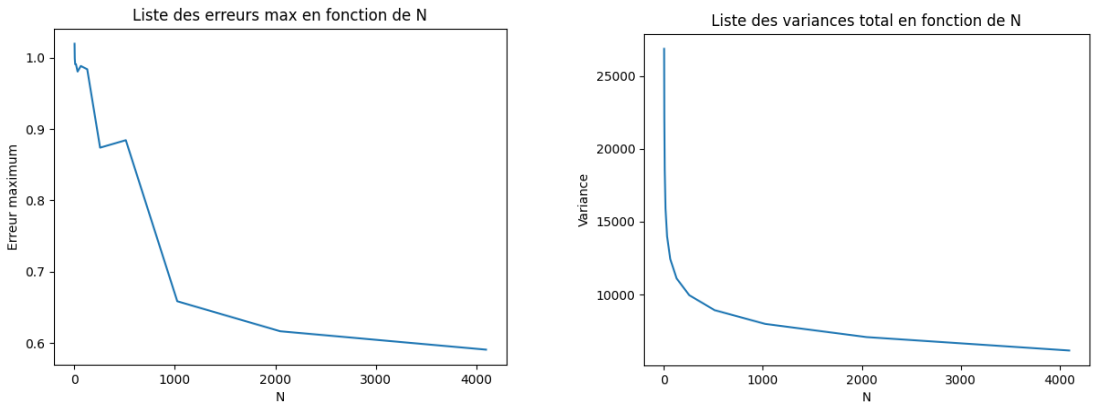
Quelle est la différence entre la plupart des classes de la base Caltech et celles que vous avez utilisées lorsque vous aviez appariés des descripteurs SURF entre eux ? Quelle incidence cela risque t'il d'avoir sur les résultats ?

La principale différence est que la base d'images que nous avions pour les descripteurs SURF était composée de la même figure, comme par exemple l'Arc de Triomphe, prise sous des angles différents de vue. L'algorithme SURF s'occupait donc de trouver les points clés des images et de les comparer à ces points clés dans les images avec les autres angles de vue.
Ensuite la base de données actuelle Caltech est composée de plusieurs images qui représentent soit des dessins soit des textes et autres.
Concernant les résultats, cette base de données risque de rajouter du temps de calcul inutile entre des images qui n'ont rien à voir entre elles.

2.1.2)

Incluez les graphiques obtenus dans votre rapport.
⇒ Inertie et plus grande erreur

Déduisez-en une valeur de N qui vous semble convenable, selon une méthode relative à K-means que vous avez déjà vue dans un cours antérieur.



D'après le graphique ci-dessus ressorti de notre programme, nous pouvons remarquer que la variance cesse d'évoluer significativement à partir de 512 clusters. On peut donc conclure que 512 clusters semble être la valeur N convenable.
Cependant nous avons décidé de passer sur N = 4096 clusters, étant donné que les modèles étaient déjà calculés, et sauvegardés.

A supposer que la méthode de classification que vous utilisez après vectorisation soit la LDA, et que vous n'avez que 2 classes à séparer, quel impact pourrait avoir un vocabulaire visuel qui contiendrait trop de mots ? Pas assez de mots ?

S'il n'y a pas assez de mots, l'erreur sera proche de 1 et sera moins intéressante lors de la prédiction.
S'il y a trop de mots, le temps de calculs sera extrêmement long.

4.1)

Pour cette partie du TP nous allons utiliser les 2 classes suivantes : cougar_face, Garfield.

4.1.1)

A quoi correspondent les valeurs renvoyées par s.transform() ? Quel effet l'augmentation du degré du polynôme a t'elle ?

Les valeurs retournées par la fonction s.transform() sont le degré "d'implication" d'une image dans une classe. Considérons un spectre en deux dimensions où nous avons les classes "cougar_face" et "garfield" aux deux extrémités de ce spectre. Les descripteurs d'images, vont être traités afin d'avoir leurs positions sur leurs spectres.
Lorsque l'on augmente le degré, les extrémités de ce spectre sont de plus en plus éloignées, et donc le positionnement des descripteurs est plus précis sur ce dernier.

Essayez la même approche sur vos données X et y extraites en section 4.1, en faisant varier le degré si nécessaire. Commentez vos résultats : les projections semblent-elles indiquer une bonne séparation des classes ou non ? (la réponse peut être oui comme non, tout dépend des classes que vous avez choisies).

En utilisant nos données, nous pouvons remarquer une bonne différenciation des classes entre les images de tests, chacune ayant des positions proches entre elles lorsqu'elles sont de la même classe.

4.1.2)

Réutilisez vos données X et y extraites en section 4.1 pour séparer les deux classes par un C-SVC polynomial, avec une constante de régularisation et un degré raisonnable. Combien de points supports obtenez-vous ? Aide : sklearn.svm, SVC(), fit().

On doit avoir autant de points de supports que d'images, car les descripteurs seront, comme avec le principe du KDA, répartis sur un spectre et certaines de ce spectre seront associés à des classes. Donc une image, découpée par ses descripteurs, sera associée à un groupe d'image, définissant une classe.

Vectorisez les 4 images des répertoires de test qui n'ont pas été utilisées pour l'instant, et classifiez-les. Reportez vos résultats.

En vectorisant les 4 images de chaques répertoires de test (sans prendre en compte les images que n'avons pas entraîné dans notre modèle), nous arrivons à ce résultat :

```
-- Succès -- Nom fichier : 101_ObjectCategories\cougar_face\test\image_0014.jpg -- Prédiction : cougar_face
-- Succès -- Nom fichier : 101_ObjectCategories\cougar_face\test\image_0039.jpg -- Prédiction : cougar_face
-- Succès -- Nom fichier : 101_ObjectCategories\cougar_face\test\image_0053.jpg -- Prédiction : cougar_face
-- Succès -- Nom fichier : 101_ObjectCategories\cougar_face\test\image_0058.jpg -- Prédiction : cougar_face
-- Succès -- Nom fichier : 101_ObjectCategories\garfield\test\image_0005.jpg -- Prédiction : garfield
-- Echec -- Nom fichier : 101_ObjectCategories\garfield\test\image_0017.jpg -- Prédiction : cougar_face
-- Echec -- Nom fichier : 101_ObjectCategories\garfield\test\image_0018.jpg -- Prédiction : cougar_face
-- Succès -- Nom fichier : 101_ObjectCategories\garfield\test\image_0035.jpg -- Prédiction : garfield
-- Résultat de la prédiction -- Nombre de succès : 6 -- Nombre d'échecs : 2
```

En premier lieu, nous avions des résultats problématiques, quand nous avons mis le degré 2 du polynôme, qui sépare les classes dans le modèle SVC, nous avons eu un résultat où toutes les images étaient prédits comme “cougar_face”. En ajustant ce paramètres, nous avons eu un résultat plus convaincant, soit 2 échecs, et 6 réussites.

4.2)

Vous utiliserez C-SVC, avec le noyau et les paramètres de votre choix. En enchaînant les décisions A-B ou A-C pour aboutir à une classe finale, reportez les classifications obtenues pour chaque image de test non encore vue après les avoir vectorisées.

Nous avons utilisé deux modèles SVC, l’un pour la prédiction lors de la première classification, et un deuxième pour la prédiction lors du deuxième dans une classification.
Pour expliquer le fonctionnement :

Nous avons 4 classes soit : , cougar_face, camera, garfield, et ant
Nous prenons une image “test.jpg” qui est un dessin du personnage “garfield”

Comme classificateur A : nous allons grouper en premier lieu les classes “cougar_face” et “cemara” comme la classe “1”.
Ensuite nous allons regrouper en second lieu les classes “garfield” et “ant” comme la classe “2”.
Nous allons entraîner notre premier modèle SVC, paramétré avec une séparation via un polynôme de degré peu élevé, car nous avons une vue bien trop globale de la séparation des classes. Avec ce modèle, nous allons voir si l’image “test.jpg” appartient à la classe “1” ou “2” précédemment définie.
Naturellement, le modèle va dire que “test.jpg” fait partie du groupe “garfield” et “ant” donc la classe “2”.
Comme classificateur B (classe “1” précédemment) ou C (classe “2” précédemment) : nous allons entraîner un nouveau modèle SVC qui se voudra plus précis, donc aura un degré de polynôme plus élevé, cette fois-ci, en expérimentant plusieurs degrés, nous avons décidé de prendre un degré = 6.
Nous définirons ensuite que la classe “1” est attribuée à “garfield” et la classe “2” est attribuée à “ant”. Nous entraînons notre modèle SVC sur les images d’apprentissage de ces deux classes afin que les prédictions soient plus spécifiques à ces deux classes.
Ensuite nous demandons au modèle de nous prédire dans laquelle de ces deux classes notre image “test.jpg” se situe, normalement la classe “2” qui a été attribuée pour “garfield”.

Ceci est pour l'explication théorique du fonctionnement de notre code. Maintenant nous avons utilisé cette méthode pour chaque image présentes dans nos dossiers de tests pour les 4 classes définies au début de ce TP.
Voici les résultats obtenus :

```
filename : 101_ObjectCategories\ant\test\image_0002.jpg -- x predict : 2 -- y : 2
filename : 101_ObjectCategories\ant\test\image_0023.jpg -- x predict : 2 -- y : 2
filename : 101_ObjectCategories\ant\test\image_0030.jpg -- x predict : 1 -- y : 0
filename : 101_ObjectCategories\ant\test\image_0036.jpg -- x predict : 2 -- y : 2
filename : 101_ObjectCategories\camera\test\image_0005.jpg -- x predict : 1 -- y : 2
filename : 101_ObjectCategories\camera\test\image_0012.jpg -- x predict : 1 -- y : 2
filename : 101_ObjectCategories\camera\test\image_0015.jpg -- x predict : 1 -- y : 2
filename : 101_ObjectCategories\camera\test\image_0032.jpg -- x predict : 1 -- y : 2
filename : 101_ObjectCategories\cougar_face\test\image_0014.jpg -- x predict : 1 -- y : 1
filename : 101_ObjectCategories\cougar_face\test\image_0039.jpg -- x predict : 1 -- y : 1
filename : 101_ObjectCategories\cougar_face\test\image_0053.jpg -- x predict : 1 -- y : 1
filename : 101_ObjectCategories\cougar_face\test\image_0058.jpg -- x predict : 1 -- y : 1
filename : 101_ObjectCategories\garfield\test\image_0005.jpg -- x predict : 2 -- y : 1
filename : 101_ObjectCategories\garfield\test\image_0017.jpg -- x predict : 1 -- y : 0
filename : 101_ObjectCategories\garfield\test\image_0018.jpg -- x predict : 1 -- y : 0
filename : 101_ObjectCategories\garfield\test\image_0035.jpg -- x predict : 2 -- y : 1
-- Résultat de la prédiction -- Nombre de succès : 7 -- Nombre d'échecs : 9
```

La valeur "x predict" étant la valeur retournée par la prédiction du modèle SVC. Et la valeur "y" étant la valeur du vecteur y de notre image.

Dans nos résultats obtenus, les classes où le y est égale à "0" sont des erreurs expliquées par le fait que la prédiction, lors du classificateur A entre les 4 classes, s'est mal passée.
Donc l'image qui était testé lors de la classification B ou C étaient forcément mauvaise, par exemple :
SVC a prédit qu'une image de Garfield faisait partie du groupe "ant", et "cougar_face" alors que non, sont Y sera égale à 0 lorsque l'on crée la valeur y de test, créé en même temps que la matrice X de test.

Les autres erreurs sont celles où le "x predict" est différent du "y", montrant que la prédiction du premier classificateur A s'est bien passée, mais que la prédiction du deuxième classificateur B ou C, s'est mal passée. Pour prendre exemple :
SVC a prédit qu'une image de Garfield faisait partie du groupe "garfield", et "caméra" mais prédit que cette image est de la classe "caméra" entre les deux. Le "y" sera alors égale à "2" et sera différent du "x predict" égale à "1".

Pourquoi avons-nous tant d'erreurs
Afin de répondre à la question, nous avons décidé de refaire entièrement ce TP dans plusieurs combinaisons possibles de données, peut-être que nos données sont difficiles à traiter pour le modèle SVC.
Nous avons créé une version alternative du code, présent dans le dossier **"v2_different_data"** qui génère des combinaisons aléatoires de 4 classes et va calculer

leurs KMeans, leurs erreurs max, variances, ainsi qu'appliquer le principe décrit au-dessus sur l'apprentissage de 4 classes.
Après avoir fait tourner ce programme pendant un assez long moment, nous avons pû voir les résultats des différentes prédictions, présents dans le dossier "big_db", suivi du nom de la combinaison de 4 classes, dans le fichier "Prediction_4class.txt". Dans ces fichiers nous pouvons voir le nombre d'erreurs que le modèle SVC a fait dans ses différentes prédictions. Et nous avons pu voir que, dans la majorité des cas, le modèle SVC fait toujours le même nombre d'erreurs…

En ajustant les paramètres du modèle SVC, tel que le gamma, permettant de rogner un maximum les zones attribuées au classés dans le "spectre" dont on a parlé précédemment, nous n'avons pas trouvé de configuration optimale afin d'avoir une précision supérieure à 50%.
Nous en avons donc conclu que le modèle SVC n'était tout simplement pas assez entraîné pour définir les différentes classes qu'on lui donne. Et que plus de classes sont données, plus de données sont nécessaires pour construires les modèles, étant donné que les prédictions sur 2 classes sont généralement assez bonnes.
Il se peut aussi que nous ayons choisit des classes bien trop compliquées à gérer pour le modèle SVC, peut-être qu'un autre modèle serait plus efficace, tel que le modèle séquentiel de convolution d'un réseau de neurones, soit CNN, mais nous arrivons dès lors dans le domaine de l'IA…

5. BONUS)

Est-il envisageable d'aboutir à une méthode de classification similaire ? Si oui, quelles modifications proposez-vous d'apporter ? Sinon, pourquoi est-il impossible d'intégrer cette nouvelle information dans les décisions ?

Nous pensons qu'il serait intéressant d'utiliser les régions de couleurs en complément aux SURF qui sont extraits des images. Car pour l'instant, nous n'utilisons pas les points clés de nos images, qui sont associés aux descripteurs de ces derniers.
Avec les keypoints, nous pourrions récupérer la zone de couleur courrante de notre descripteurs. Avec ces régions de couleurs, on peut possiblement effectuer le même traitement que l'on a effectuer avec les descripteurs : des listes de points à une classe sur un spectre qui sont ensuite envoyés dans un modèle afin que l'on ai une deuxième version des prédictions, cette fois-ci basé sur les couleurs des points clés de l'image.