Apprentissage Automatique

Rapport de projet

Choix du dataset

Plusieurs pistes de réflexions se sont initialement offertes à nous. Nous voulions expérimenter la réalisation d'un classifieur. Nous avons songé à réaliser un classifieur permettant de différencier :

- Une mouette, d'un goéland
- Un corbeau, d'une corneille

Finalement, nous sommes partis sur la différenciation entre abeilles avec pollen et sans pollen.

Exemple de données du dataset

Nous avons souhaité, au vu du temps disponible rester sur un dataset assez propre à l'image du dataset Fashion MNIST vu en cours (images de montres, chaussures, sacs, pulls).

Informations sur le dataset

Lien	Description	Taille	Format des données
https://www.kaggle.com/ivanfel/honey-bee-pollen	2 classes : Abeilles avec pollen et sans pollen sur les pattes	714 photos	Images .jpg 300x180

Identification de la tâche, de la classe d'algos, des structures de données features

a) Objectif global

La finalité de notre projet est de pouvoir déterminer la santé d'une ruche. En effet, si une abeille a du pollen sur les pattes, cela indique qu'elle amène de la nourriture au couvain (larve d'abeilles). Ce qui nous permet d'avoir une idée de la santé globale d'une ruche.

b) Type de tâche à accomplir

Nous avons deux classes à déterminer :

- Une abeille sans pollen
- Une abeille avec du pollen

Le modèle est ici supervisé, nous cherchons à déterminer si un élément appartient à l'une ou l'autre des classes, nous utilisons donc un algorithme de classification (sous-catégorie de modèle supervisé).

c) Structure des données et aspects du formatage

Nous n'avons pas de données catégorielles, en effet malgré le fichier .csv fourni, il ne contient pas d'autres informations que le nom de l'image et sa catégorie associée.

	filename	pollen_carrying
0	P10057-125r.jpg	1
1	P57380-120r.jpg	1
2	NP63671-221r.jpg	0
3	NP53776-47r.jpg	0

Nous n'utilisons pas ce fichier car, le préfixe du nom des images définit leur classe. Par exemple : **NP**1268-15r.jpg pour une abeille **n**on **p**ollinisatrice et **P**7797-103r.jpg pour une abeille **p**ollinisatrice.

Seul point de vigilance relevé par le créateur du dataset : les numéros correspondent respectivement au numéro de l'image et de "la frame" et ne sont donc pas numérotés de manière séquentielle. Pas de problème dans notre cas nous ne les exploitons pas (aucune valeur).

Nous nous sommes orientés vers un pré-processing PCA (Principal Component Analysis) pour optimiser les performances.

d) Quel algorithme(s) semble(nt) bien adapté(s) ?

Nous avons choisi de nous orienter vers un bayésien naïf gaussien qui nous permettra d'agir en tant que classifieur supervisé permettant de mettre en relief une variation de forme.

Une évolution possible aurait été l'utilisation d'un algorithme bayésien « non naïf ». Pour pousser l'exploration encore plus loin, un SVC aurait aussi pu être envisagé étant également un algorithme de classification.

e) Points de vigilance particuliers (choix des hyperparamètres, mesure des performances, ensemble de données non équilibrées, etc).

Ce dataset a la particularité d'avoir des images au format .jpg 180x300 pixel ce qui est une résolution plutôt importante comparé au MNIST Number (28×28 pixel) par exemple..

Il a donc fallu faire attention aux dimensions de la structure des données et tester plusieurs "sous-résolutions" afin d'avoir un plus gros grain (compromis à faire malgré la perte de structuration lié à la PCA).

Large grain

In [4]:
```
# size = (300, 180) # default resolution
# size = (150, 90)
# size = (100, 60)
# size = (30, 18)
# size = (25, 15)
size = (20, 12)   # preferred resolution
# size = (15, 9)    # too small
nbClasses = 2
```

En effet, l'utilisation de la PCA nous a amené des problématiques liées aux nombres de composantes à tester. Nous nous sommes par exemple limités à un maximum de 500 composantes (grâce à la réduction de résolution, nous en étions encore loin) pour des raisons de puissance de calcul.

Démarche / Méthodologie

Travail réalisé

1. *Rendre le dataset exploitable*

La première phase du projet a consisté à extraire les photos de leur dossier et les stocker sous forme de np.array à l'aide de la librairie skimage. Nous avons eu besoin de se débarrasser des composantes couleur (greyscale) afin de réduire le nombre de dimension des images et car des composants RGB ne nous apportait que peu d'informations.

2. *"Mélanger" le dataset et le décomposer en sous-ensembles*

Par défaut, le chargement des données se faisait par ordre alphabétique donc il y a eu besoin de mélanger la liste des fichiers pour pas avoir tous les P à la suite puis les NP.

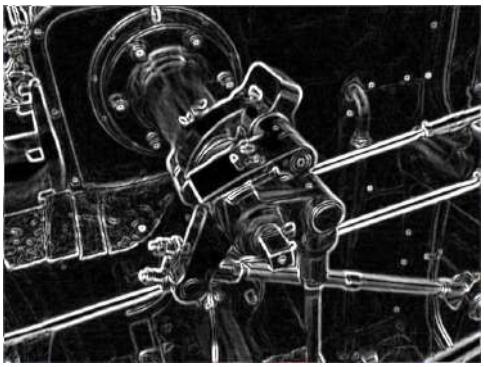
3. *Premier essai Bayésien Naif Gaussien*

La classe GaussianNB utilisée provient du module *naive_bayes* de la librairie sklearn. Cela nous a permis d'avoir des premiers résultats aux alentours de 20 et 25%.

Nous avons également testé les classes ComplementNB, BernoulliNB, CategoricalNB mais GaussianNB s'en est très souvent mieux sorti.

4. *Pré traitement : filtre sobel*

Après avoir aplanit les images du dataset, nous leur avons appliqué un filtre sobel utilisé dans le but de mieux déduire "les coutours" des éléments de l'image (≈25% → ≈33%).



Exemple d'utilisation d'un filtre sobel

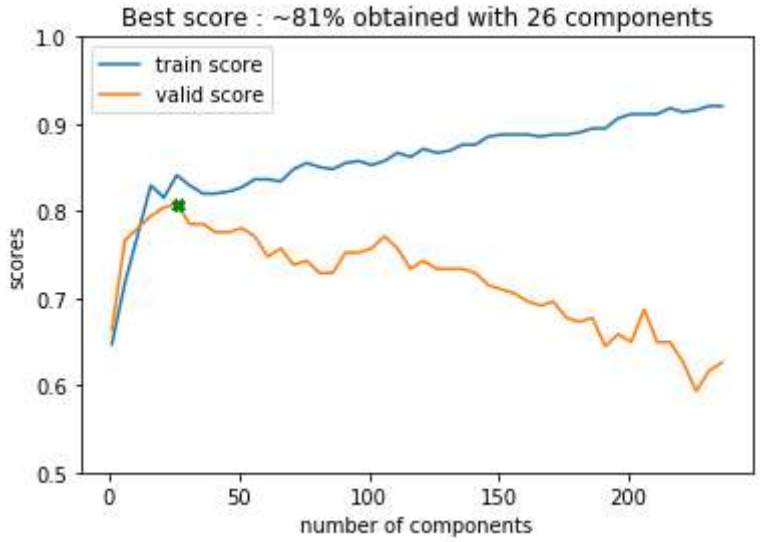
5. *Premiers essais de la PCA*

La classe PCA utilisée provient du module *decomposition* de la librairie sklearn, premiers essais avec n_components=95% fournissant une explication de la variance.

6. *Boucles variation paramètres, graphes et analyse*

Dans un second temps et afin d'optimiser les performances, nous avons fait varier le nombre de composantes de la PCA, ce qui a conduit aux observations suivantes

Les meilleurs résultats sont donc obtenus avec le modèle GaussianNB couplé à une PCA ayant un nombre de composantes compris entre 20 et 30.



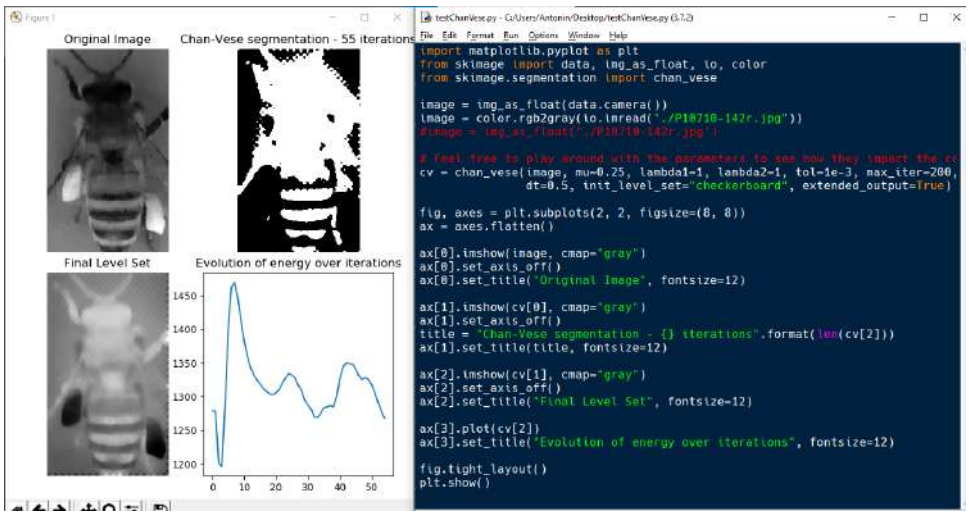
Ces scores sont obtenus pour un ratio de 60% de données d'entrainement et 30% de données de validation, ratio que nous trouvons le plus optimal et le moins biaisé. Le temps d'exécution (chargement du dataset + PCA + Bayésien Naif) est plus que raisonnable et ne dépasse rarement plus qu'une minute.

La majorité du temps, on observe un phénomène d'underfitting se produire.

Travail en cours / non réalisé

7. Essai d'autres filtres que sobel (gaussian blur, skimage seed mask, canny edge detector..)

D'autres filtres de la librairie skimage, notamment Chan-Vese Segmentation, auraient pu donner des résultats très propres et peut-être plus exploitables :



Dans les faits, la segmentation Chan-Vese est moins intéressante (≈70% d'efficacité) et beaucoup plus longue (10min si l'on redimensionne les images qu'après traitement). Sans redimensionnement, elle aurait peut-être pu faire la différence car les zones de pollen se détachent vraiment de l'image (voir ci-dessus).

Au final, pas mal d'autres filtres existent et auraient pu correspondre à nos besoins mais leur prise en main et l'optimisation de leurs paramètres aurait été un peu trop chronophage.

Conclusion

Il est pour nous difficile d'optimiser encore plus les résultats obtenus avec le Bayésien Naif car les abeilles avec/sans pollen se ressemblent plus qu'un 5 et un 1 dans MNIST par exemple : gros grain amplifie surement un peu.

Nous avons peu nous rendre compte que le pré-traitement influence beaucoup les résultats, même si le filtre sobel n'est peut-être pas le plus optimisé, la marge d'amélioration est quantifiable bien que la moyenne (mean) observée ressemble plus à une bouillie de pixels que sans ce filtre.

Suite à ce travail, il reste donc pour nous une part de mystère car, à l'œil nu, nous étions loin de nous douter de l'efficacité de cette solution !