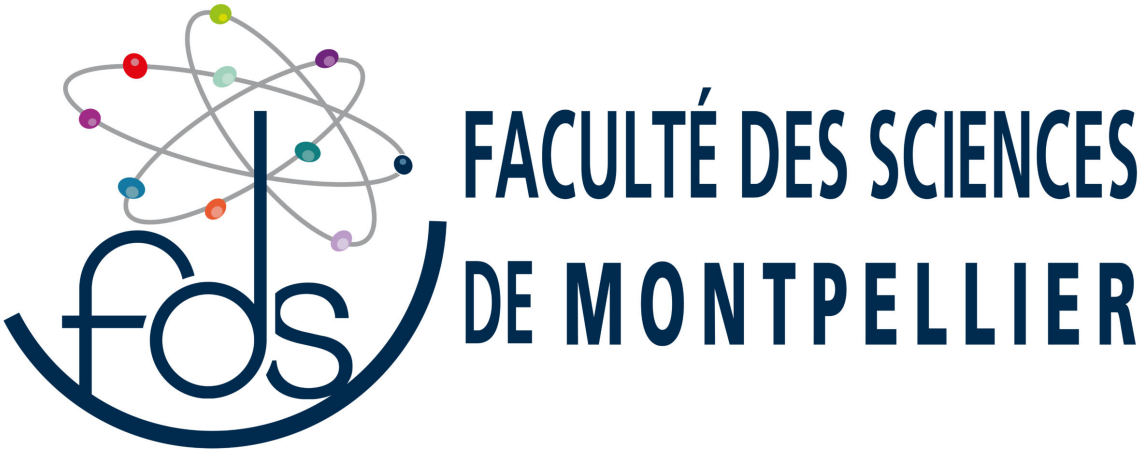
TP : Support Vector Machines

HAX907X - Apprentissage statistique

CLETZ Laura

2025-09-30

Table of contents



1 Introduction

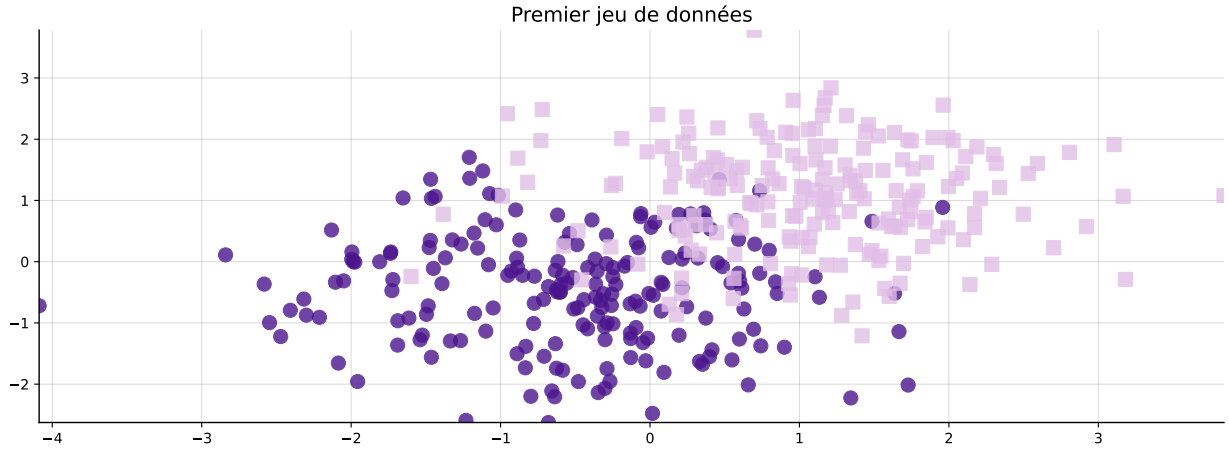
Ce rapport présente la mise en pratique des Support Vector Machines (SVM) comme méthodes de classification. Voici ce que nous traiterons :

- nous générerons tout d'abord notre propre jeu de données pseudo-aléatoires pour prendre en main les fonctions qui nous ont été transmises dans le fichier `python_script/svm_source.py` ;
- nous manipulerons ensuite ces mêmes fonctions en usant du jeu de données "***iris***" ;
- nous jouerons avec les fonctionnalités du SVM GUI ;
- nous verrons ensuite comment les SVM permettent aussi la classification d'images à partir des visages de personnalités politiques ;
- nous observerons sur ces mêmes données la différence dans les performances des SVM avec ou sans facteur de nuisance, avant ou après réduction des dimensions.

L'aide à la stylisation CSS est fournie par GitHub Copilot sous Visual Studio Code… Nous pouvons d'ores et déjà charger les librairies Python nécessaires, en particulier le package `scikit-learn`. Il faudra fixer une graine à chaque cellule, nous optons pour 9204.

2 Prise en main

Nous commençons par générer un jeu de données simple composé de deux gaussiennes bidimensionnelles.

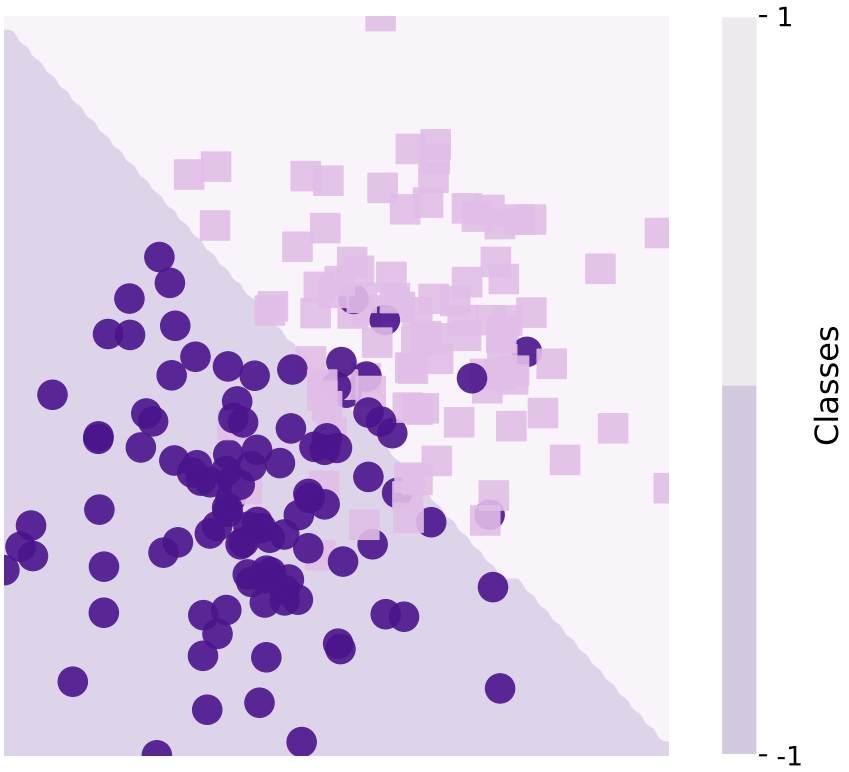


Nous voyons le nuage gaussien formé par des deux plus petits nuages gaussiens de couleurs distinctes, eux-mêmes issus des deux vecteurs gaussiens pseudo-aléatoirement générés.

Nous y avons ajouté notre touche personnelle : des dégradés du violet au lilas, toujours accessibles aux personnes daltoniennes.

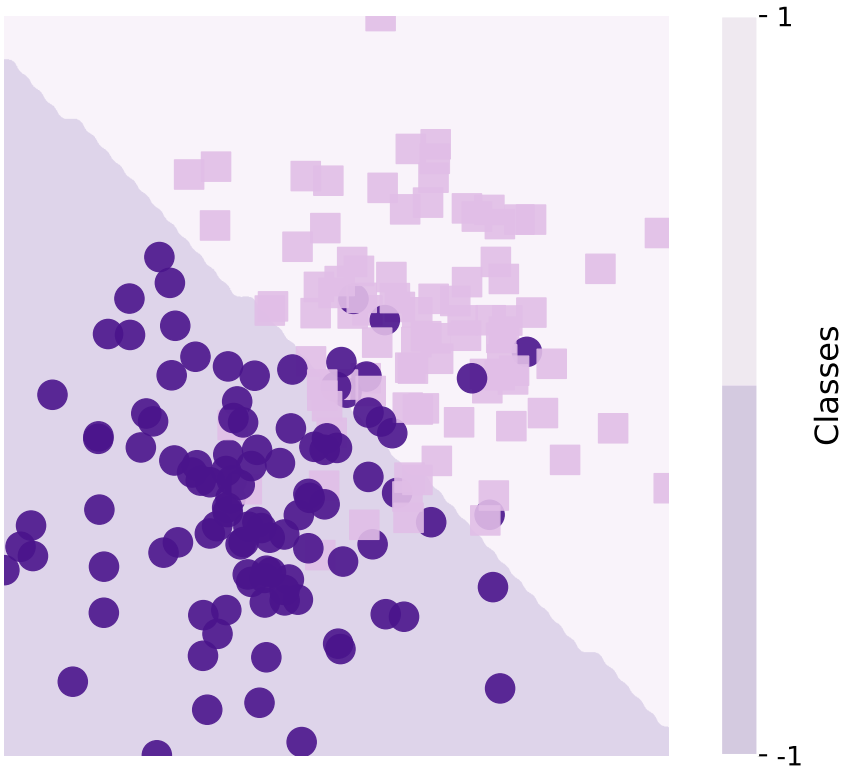
Testons un premier ajustement par SVM sur ces jeux de données de façon à les “séparer”, à les classifier, linéairement pour commencer :

```
Paramètre de régularisation C : 1.0
Paramètre de fonction noyau : linear
Score : 0.9
```



Nous discernons la séparation entre le plus gros de chaque vecteur de données mais le choix d'un noyau linéaire fait qu'il y a tout de même des données de chaque vecteur qui se retrouvent dans la "partie de l'autre". Essayons d'optimiser le choix du paramètre de régularisation C :

```
Meilleur paramètre de régularisation C : 0.15095
Paramètre de fonction noyau choisi : linear
Score : 0.905
```



Il semblerait qu'il y ait plusieurs paramètres C pour lesquels le score est à 0.875 qui est vraisemblablement la valeur optimale. Pourtant, le graphe n'est pas plus convainquant que le précédent.

Un meilleur jeu de données nous permettrait sûrement d'y voir plus clair quant à la performance de classification des SVM.

3 Mise en pratique avec "iris"

Le jeu de données "iris" est disponible pour de nombreux packages sous différents langages de programmation.

Il s'agit d'un échantillon de 150 iris (la plante) équi-réparties dans 3 groupes : l'espèce *setosa*, l'espèce *versicolor* et l'espèce *virginica*.

Séparons ces données aléatoirement en deux jeux de données "train" et "test". X est l'ensemble des facteurs pouvant servir à déterminer l'espèce (longueur et largeur des sépales et des pétales par exemple) et y est l'espèce.

Tentons la classification par SVM comme vue dans la première partie en utilisant le noyau linéaire et une optimisation du paramètre de régularisation C :

```
{'C': 0.8406652885618325, 'kernel': 'linear'}
Score de généralisation pour le noyau linéaire : 0.7466666666666667, 0.68
```

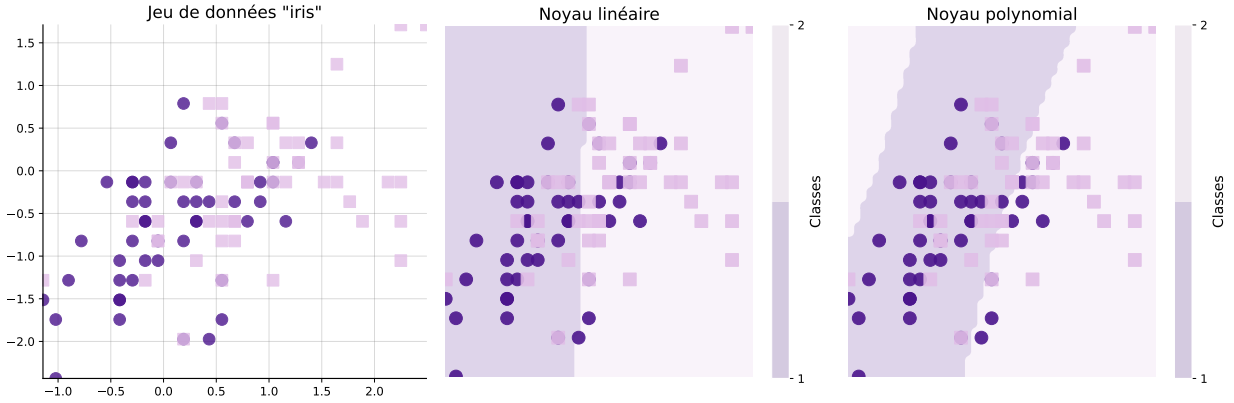
Il est cohérent que le premier score apparaissant soit supérieur au second car ils sont le score de généralisation pour, respectivement, l'ensemble d'apprentissage ("train") et l'ensemble de "test". Grâce au noyau linéaire et le C optimal, nous tournons autour de 70% de précision.

Voyons si le noyau linéaire, donc un polynôme de degré 1, est meilleur qu'un noyau polynomial d'ordre supérieur.

```
Score de généralisation pour le noyau polynomial : 0.6533333333333333, 0.52
```

Les scores obtenus du noyau polynomial d'ordre 2 sont moins satisfaisants que ceux du noyau linéaire. Si nous avions autorisé l'optimisation de l'ordre du polynôme à choisir le degré 1, alors il aurait été choisi et le modèle obtenu aurait été exactement le même que le précédent.

Reportons graphiquement ces résultats :



À l'oeil, ce n'est pas si facile de distinguer la performance de la méthode de noyau linéaire contre celle de noyau polynomial. Il semble, en effet, que la frontière tracée pour la classe 1 (espèce *setosa*) recouvre davantage de points de la classe 2 (espèce *versicolor*) afin de récupérer davantage de points de la classe 1.

4 SVM GUI

Nous allons suivre les indications du site https://scikit-learn.org/1.2/auto_examples/applications/svm_gui.html qui prête le code que nous pouvons retrouver dans `python_script/svm_gui.py`. Ce script est à lancer en console et c'est aussi dans la console qu'apparaitra la précision ("*accuracy*") du modèle. Nous créons 18 points de classe 1 (les noirs ci-dessous) et 4 points de classe 2 (les blancs) et nous essayons d'ajuster une classification en jouant avec trois fonctions à noyau :

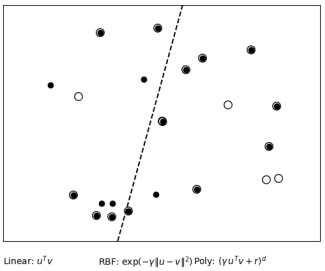
- le noyau linéaire ;
- le noyau polynomial ;
- la fonction RBF (radial basis function) ;

et quatre hyperparamètres :

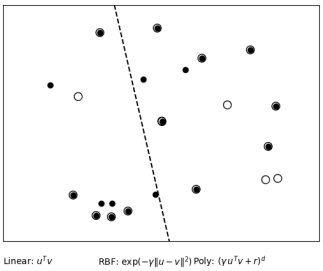
- le paramètre de régularisation C que nous avons déjà vu et qui détermine l'influence des erreurs de classification ;
- le paramètre γ contrôle la distance d'influence à partir d'un unique point ;

- la paramètre de degrés/d'ordre d de la fonction polynomiale ;
- le coefficient à l'origine r d'ordre le paramètre de degré.

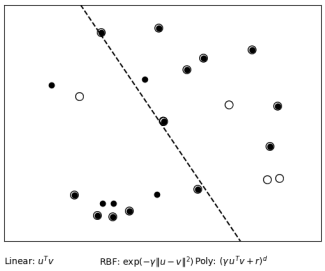
Nous trafiquerons essentiellement les paramètres C et de degré.



(a) Noyau linéaire C=1



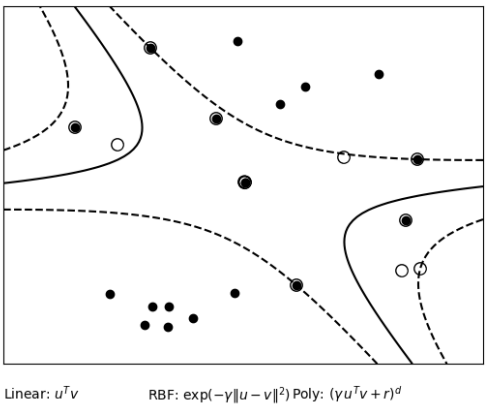
(a) Noyau linéaire C=3



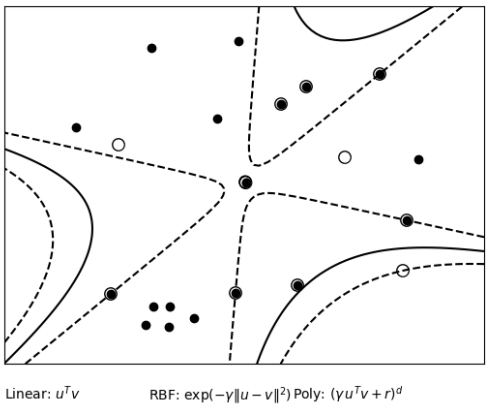
(a) Noyau linéaire C=5

Noyau linéaire et paramètre $C \in \{1, 3, 5\}$

Si l'orientation de la droite change, il n'est pas évident de comprendre le choix ou la conséquence de ces changements. Pour les trois choix de C, la droite sépare pile en deux la classe 1, un point de la classe 2 est d'un côté et les 3 autres de l'autre. Trafiquer le choix du degré pour un noyau polynomial semble avoir plus de sens :



(a) Noyau polynomial d=2



(a) Noyau polynomial d=3

Noyau polynomial et paramètre $d \in \{2, 3\}$

L'allure des deux graphes ci-dessus est nettement différente. La première, de paramètre $d = 2$, regroupe les éléments de la classe 2 dans un sablier qui prend en même temps 4 à 8 éléments de la classe 1 et exclut tous les autres, donc la classification est pertinente et visuelle. Le second, de paramètre $d = 3$, sépare bien moins clairement les classes : il est difficile d'interprêter la délimitation créée par le polynôme de degré 3.

5 Classification des visages

"*Labeled Faces in the Wild (LFW)*" est un jeu de données de reconnaissance et classification des images d'usage courant. Nous allons nous concentrer sur des images du visage de personnalités politiques et, grâce aux SVM, nous allons tenter de remettre correctement le nom de ces personnalités sur leur visage dont en voici quelques noms :

```
['Ariel Sharon',
 'Colin Powell',
 'Donald Rumsfeld',
 'George W Bush',
 'Gerhard Schroeder',
 'Hugo Chavez',
 'Tony Blair']
```

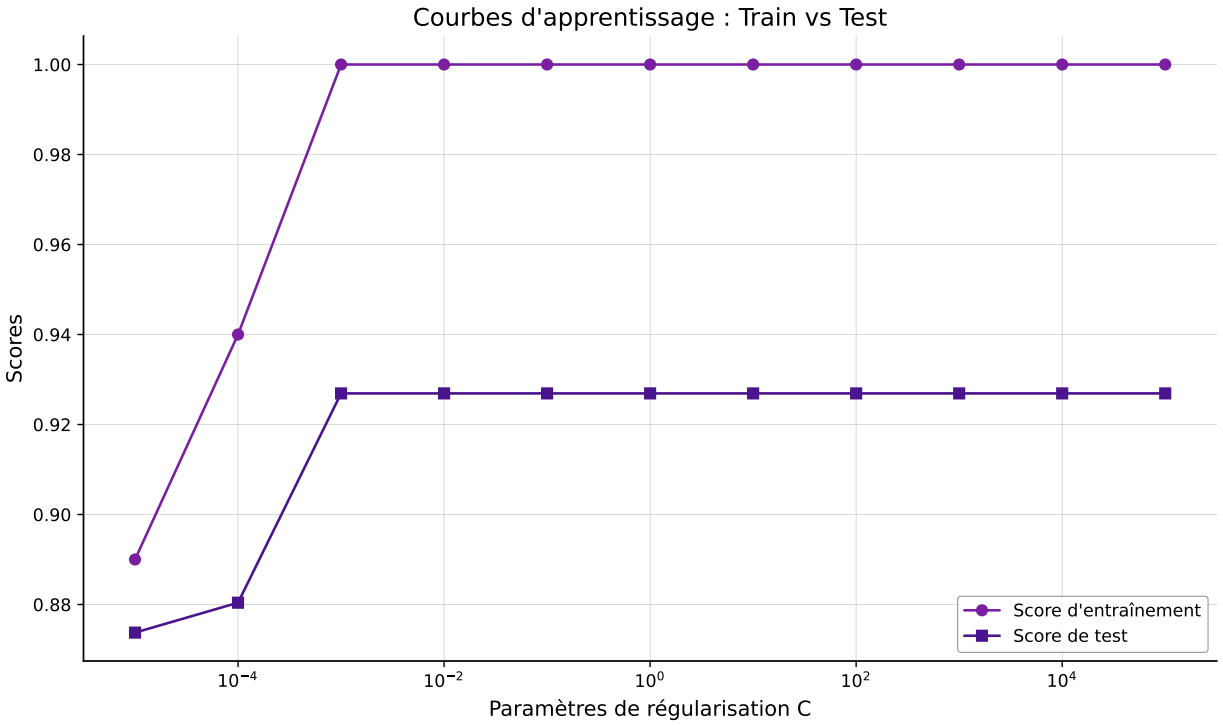
Nous allons sélectionner une paire de ces individus qui seront nos classes 1 et 2 (ou 1 et -1) : George W Bush et Hugo Chavez.

Cette base de données possède bien plus d'images labellisées de George W Bush que d'Hugo Chavez, cela pourra potentiellement freiner les performances des SVM.

```
George W Bush : 530 images
Hugo Chavez : 71 images
```

Comme nous avons fait pour les données gaussiennes et les données d'iris, nous divisons nos données LFW en "train" et "test" et nous essayons d'optimiser pour le noyau linéaire l'hyperparamètre C.

```
Effectué en  6.625s
Meilleur C (train): 0.001
Meilleur C (test): 0.001
```

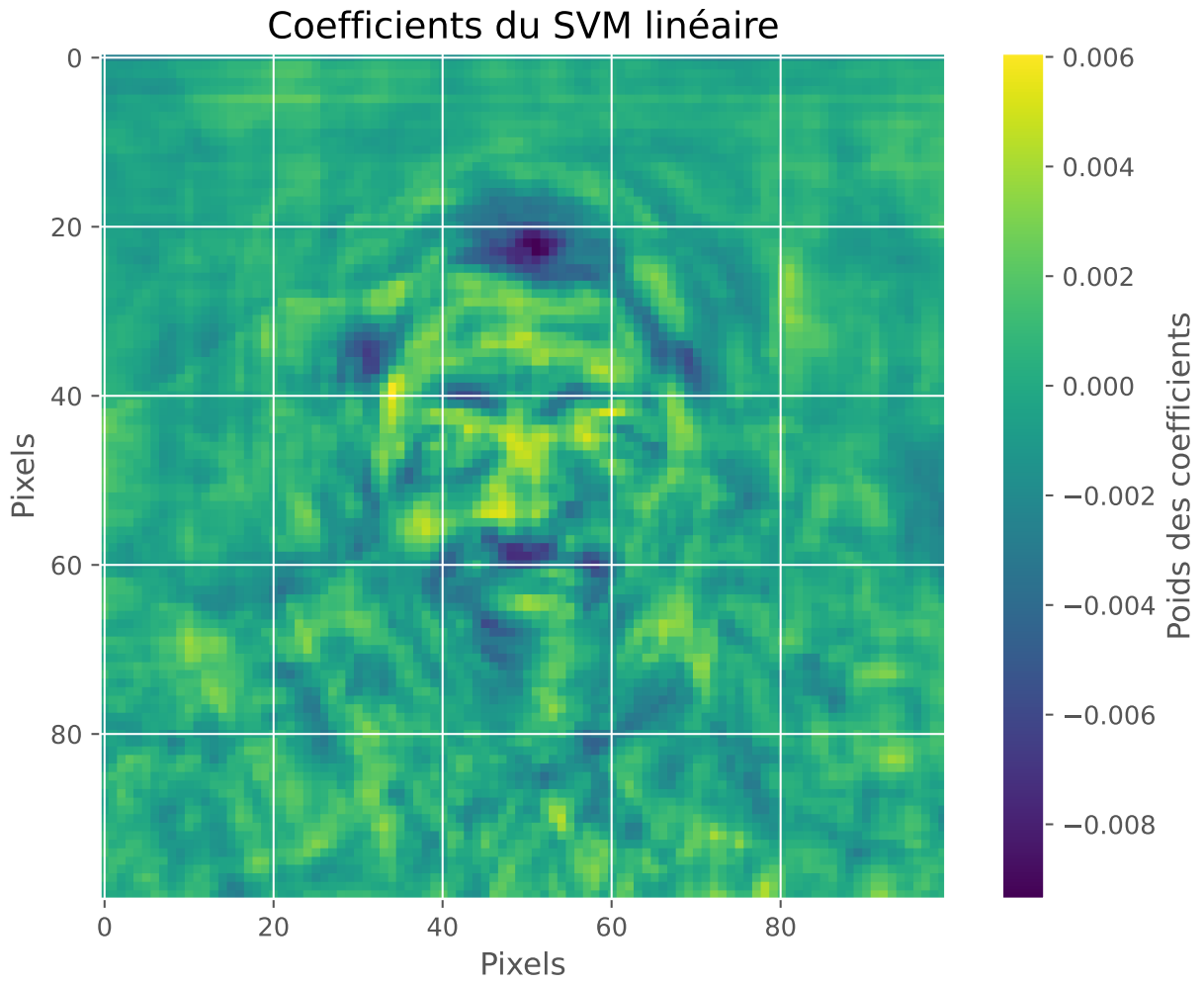



```
Meilleur train score: 1.0000
Meilleur test score: 0.9269
Écart sur-apprentissage au meilleur C (test): 0.0731
```

C'est rassurant : la valeur optimale de C est la même pour l'ensemble d'apprentissage comme pour l'ensemble de test et est égale à 10^{-3}. Un écart existe tout de même entre les scores, ce qui est attendu, évitant ainsi le sur-apprentissage. Le score sur l'ensemble "test" est proche de 95%, donc nous tournons autour d'une erreur de 5% dans la reconnaissance des visages. Les prédictions auront une marge d'erreur similaire. :

```
Prédiction du nom des personnes apparaissant dans les données "test"
Effectué en  0.663s
Niveau de chance : 0.8818635607321131
Précision : 0.9269102990033222
```

La précision obtenue est bien de 95% avec une certitude de précision largement dépassée qui était de 88%. Il est fort probable que lorsque George W Bush est apparu, l'SVM n'a pas eu de mal à le reconnaître mais que lorsque Hugo Chavez est apparu, il a également pensé qu'il s'agissait de Bush ou alors que lorsque Bush n'était pas facile à reconnaître, il l'a classé comme étant Chavez. Voyons ce qu'observent les SVM pour faire de la reconnaissance et classification de visages :



Comme nous classons George W Bush (classe 1) contre Hugo Chavez (classe 2), les zones claires (jaune vif) sont celles où Bush a des caractéristiques distinctives, à savoir, semblerait-il, le nez, les paupières et les lèvres et les zones foncées (bleu nuit) sont celles où Chavez se démarque, donc les cheveux et la pilosité faciale ou la forme de la mâchoire. Tout autour du visage il semble y avoir beaucoup de bruits car des éléments de décor sont parfois très clair ou très foncé mais ces pixels n'aident pas à reconnaître la personne.

Nous notons déjà du bruit, mais observons la différence dans les performances des SVM quand nous ajoutons du bruit en introduisant une variable de nuisance :

```
Score sans variable de nuisance
Score de généralisation pour le noyau linéaire : 1.0, 0.9169435215946844

Score avec variable de nuisance
Score de généralisation pour le noyau linéaire : 0.9966666666666667, 0.8305647840531561
```

Nous perdons 10% de précision en rajoutant du bruit. Nous ne pourrions plus tout à fait nous fier aux résultats de reconnaissance faciale apportée par les SVM. Nous devrions pouvoir améliorer le score sur les données bruitées en réduisant le nombre de dimension par ACP (Analyse des Composantes Principales).

```
Score après réduction de dimension
Score de généralisation pour le noyau linéaire : 0.89, 0.8737541528239202

Effectué en  223.208s
```

L'algorithme est d'office moins certain de son ajustement sur les données d'apprentissage mais nous gagnons une précision de presque 5% en réduisant largement le nombre de dimensions. Nous avons choisi à tâtons le nombre de composantes nous arrangeant et avons remarqué qu'au-delà de 5 composantes nous n'améliorions plus la prédiction.

6 Conclusion

Nous avons pu voir que la performance des SVM, le choix du noyau et des hyperparamètres dépend énormément de la qualité de la base de données. Complexifier un modèle en choisissant un noyau polynomial plutôt qu'un noyau linéaire n'est pas nécessairement meilleur puisque cela va dépendre de la "disposition" des données. Un jeu de données bruité est forcément plus confus et la prédiction en est moins précise mais l'under-fitting (*sous-apprentissage*) peut être évité : nous avions ajouté 300 variables de nuisance qui ont pu être corrigées par réduction de dimensions. Nous avons cependant introduit nous-mêmes un biais dans cette étude en normalisant les données "train" et les données "test" en même temps, ce qui peut produire de l'over-fitting (*sur-apprentissage*) car le modèle a donc travaillé, à une certaine mesure, sur les données "test" servant à la prédiction.