TP 3 : Support Vector Machine (SVM)

Hachem Reda Riwa El Mazzouji Wahel

Introduction

Les machines à vecteurs de support (SVM) constituent une méthode de classification supervisée largement utilisée pour leur capacité à séparer efficacement des classes, même dans des espaces de grande dimension. Leur principe repose sur la recherche d'un hyperplan séparateur maximisant la marge entre les classes, avec la possibilité d'utiliser des fonctions noyau afin de gérer des données non linéairement séparables.

Le but de ce TP est de mettre en pratique les SVM sur des données simulées et des jeux de données réels, à l'aide du package scikit-learn. Nous cherchons notamment à comprendre comment contrôler les paramètres influençant leur flexibilité, tels que les hyperparamètres et le choix du noyau.

Mise en oeuvre sur la base de données Iris

Cette section porte sur la base de données Iris. Les données sont partagées en un ensemble d'apprentissage (75 %) et un ensemble de test (25 %). Nous appliquons un SVM avec noyau linéaire puis polynomial, afin de comparer leurs performances et l'impact du choix du noyau sur la frontière de décision.

Question 1

Le noyau linéaire correspond au cas le plus simple des SVM, il consiste à chercher un hyperplan linéaire qui sépare au mieux les deux classes. Nous avons restreint l'étude aux classes 1 et 2 du jeu de données Iris et entraîné un SVM à noyau linéaire. Le paramètre de régularisation C, qui contrôle le compromis entre largeur de marge et erreurs de classification, a été optimisé par validation croisée au moyen d'une recherche en grille.

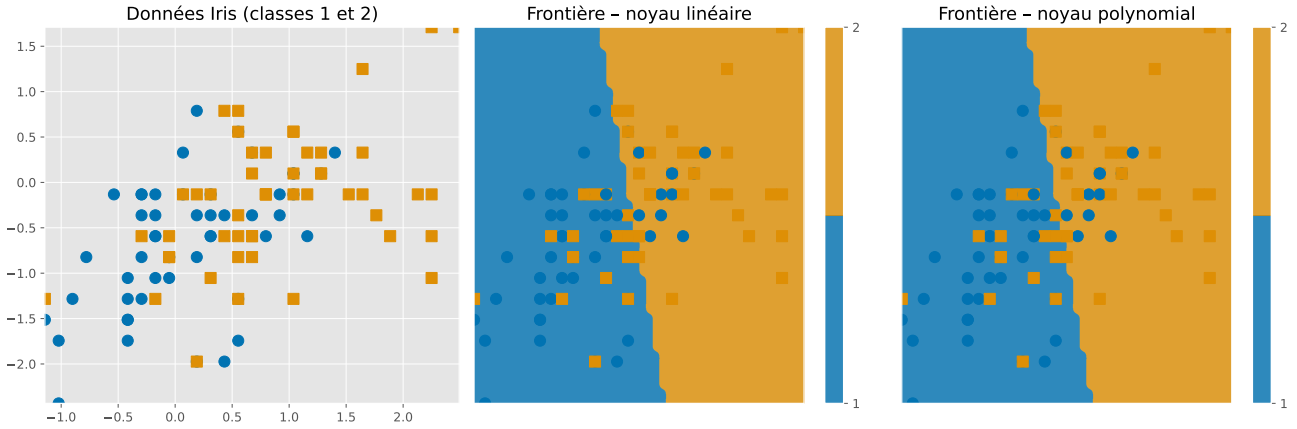
```
Meilleur paramètre C : 8.310 (kernel = linéaire)
Score entraînement : 0.733
Score test : 0.72
```

Le meilleur paramètre trouvé est C $\approx$ 8.31. Avec cette valeur, le modèle atteint une précision de 73% sur l'échantillon d'apprentissage et 72% sur l'échantillon de test.

Question 2

Le noyau polynomial permet de projeter les données dans un espace de dimension supérieure en utilisant des combinaisons polynomiales des variables d'entrée, ce qui rend possible la séparation de classes non linéairement séparables dans l'espace initial.

```
Meilleurs paramètres (poly) : C = 1.000, degré = 1, gamma = 10.0
Score entraînement : 0.720
Score test : 0.720
```



Nous avons testé un SVM avec noyau polynomial en faisant varier les paramètres C, γ et le degré du polynôme. Le meilleur modèle obtenu correspond en réalité à un polynôme de degré 1, équivalent à un noyau linéaire.

Les scores obtenus ($\approx 72\%$) sont très proches de ceux du SVM linéaire, et la frontière de décision est la même. Cela montre que, dans ce cas, l'utilisation d'un noyau polynomial plus complexe n'apporte pas d'amélioration par rapport au noyau linéaire.

SVM GUI

Question 3

Dans cette partie nous avons utilisé le script `svm_gui.py`, pour étudier le paramètre de régularisation C.

On observe que lorsque C est grand, l'algorithme cherche à classer correctement tous les points, quitte à réduire la marge et à ajuster fortement la frontière de décision.

Lorsque C est petit, les erreurs sont davantage tolérées, la marge s'élargit et la frontière se déplace au profit de la classe majoritaire.

Nous avons généré un jeu de données fortement déséquilibré, composé de 90 points appartenant à une classe et seulement 10 à l'autre. Nous avons ensuite étudié l'effet du paramètre C avec un noyau linéaire.

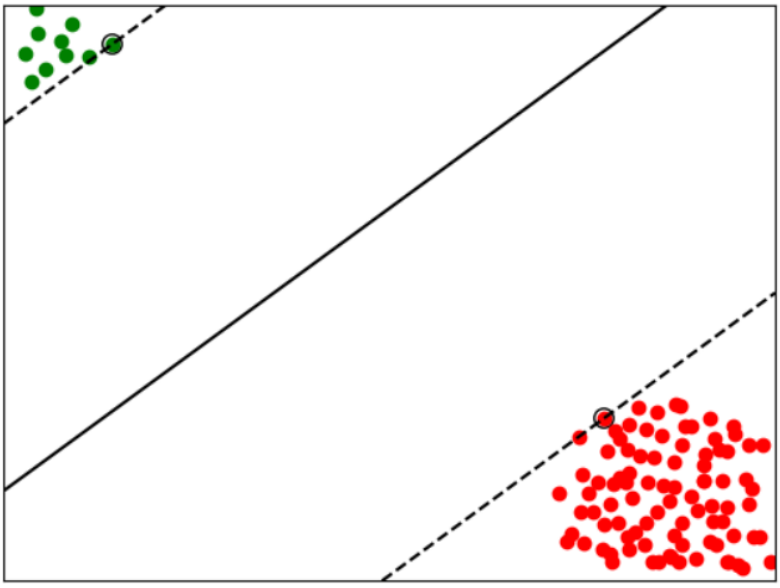


Figure 1 – Visualisation de la frontière de décision pour $C = 1$

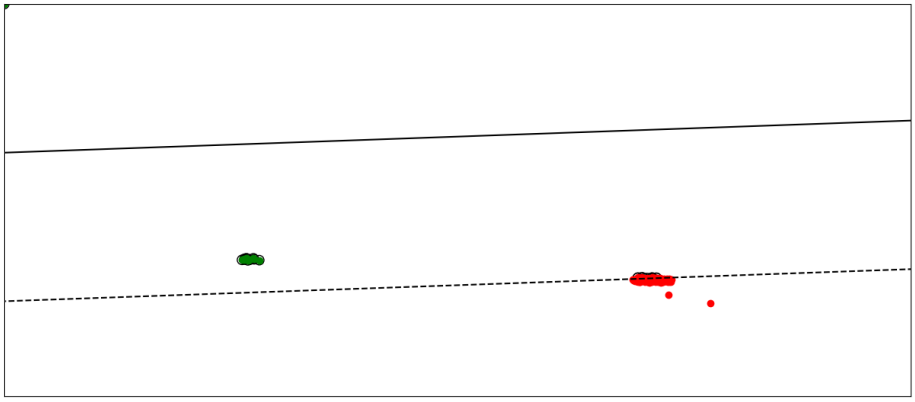


Figure 2 – Visualisation de la frontière de décision pour $C = 0.01$

On observe qu'avec $C = 1$, le SVM maintient une séparation correcte entre les deux classes, y compris pour la minorité, tout en élargissant légèrement la marge.

En revanche, avec $C = 0.01$, certains points verts franchissent l'hyperplan et sont mal classés, la frontière devient moins stricte et privilégie clairement la classe majoritaire.

Ainsi, la diminution de C élargit les marges et accentue le biais en faveur de la classe majoritaire, car le modèle tolère davantage d'erreurs sur la minorité.

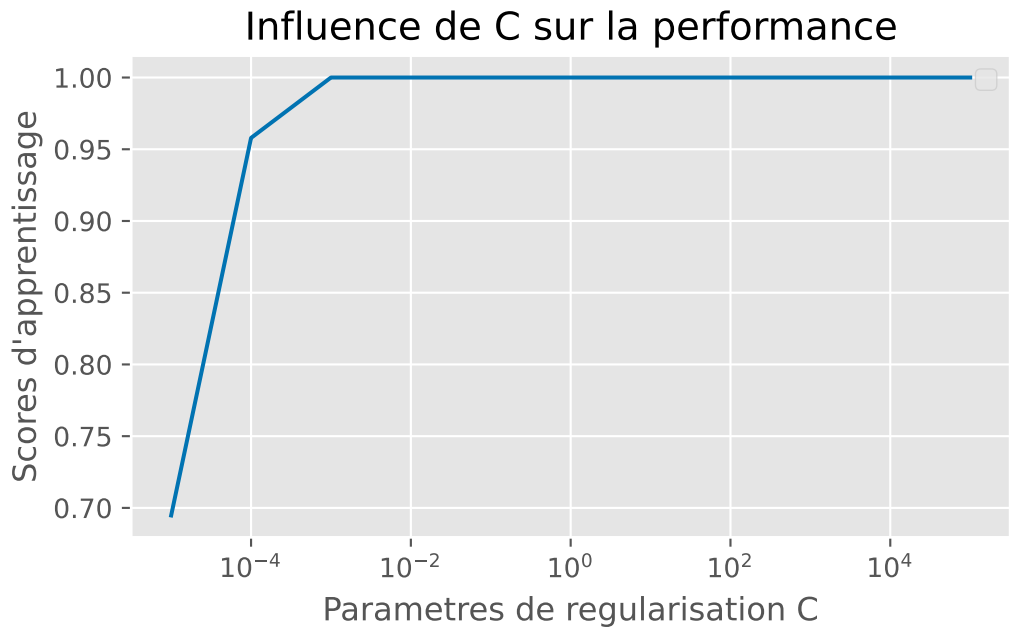
Concrètement, cet effet peut être corrigé en ajustant la pondération des erreurs via le paramètre `class_weight` de `SVC`, afin de donner plus de poids à la classe minoritaire. Une autre approche consiste à utiliser une calibration des probabilités (option `probability=True`), permettant de rééquilibrer la décision du classifieur et d'améliorer la prise en compte des classes rares.

Classification de visages

Dans cette section, nous appliquons les SVM à un problème de reconnaissance faciale en utilisant la base Labeled Faces in the Wild (LFW). L'objectif est de distinguer deux individus à partir d'images, en extrayant des caractéristiques puis en entraînant un classifieur SVM à noyau linéaire.

Question 4

```
=== SVM linéaire : influence de C ===
Meilleur C : 1.0e-03
Score test optimal : 1.000
```

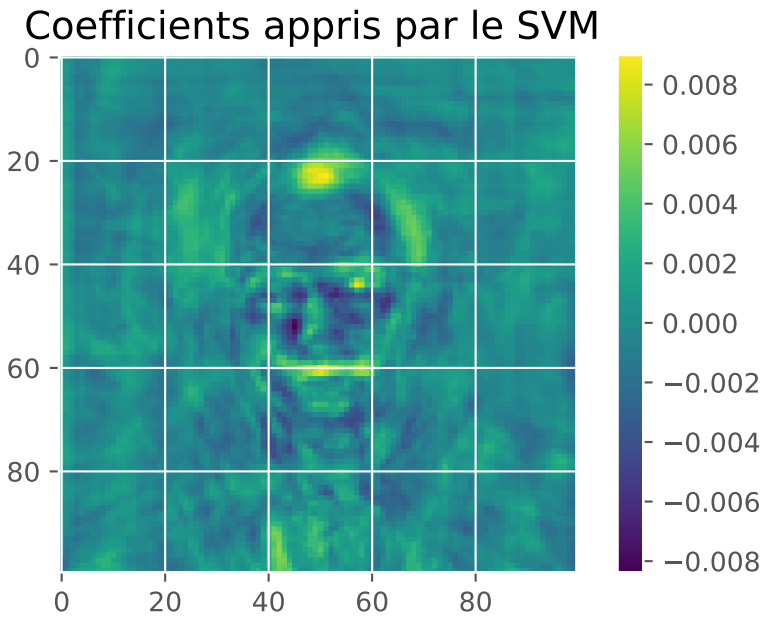



```
Prédiction des noms des personnes sur l'échantillon de test
Réalisé en 0.407s
Niveau de hasard (majorité) : 0.621
Taux de précision final sur le test : 0.900
```

L'analyse de l'impact du paramètre de régularisation C montre que, pour des valeurs très faibles, le modèle est trop contraint et n'arrive pas à séparer correctement les classes, ce qui conduit à un phénomène de sous-apprentissage. Autour de $C \approx 1.0 \times 10^{-3}$, la performance atteint son optimum et se stabilise. En revanche, pour des valeurs trop grandes de C, le modèle tend à sur-apprendre sans gain significatif.

Le meilleur compromis est obtenu pour $C \approx 1.0 \times 10^{-3}$, avec un taux de précision de 91 % sur l'échantillon de test, largement supérieur au niveau de hasard estimé à 62 %.

La prédiction sur l'échantillon de test montre que le modèle distingue globalement bien les deux individus. Sur les images affichées, la majorité est correctement classée, bien que quelques erreurs subsistent, ce qui illustre les limites du classifieur.



La visualisation des coefficients appris met en évidence les zones les plus discriminantes des visages utilisées par le SVM. Cela confirme que le modèle capture bien les traits distinctifs permettant de différencier les individus, et illustre l'efficacité de cette approche pour la classification faciale.

Question 5

Dans cette partie, nous avons évalué la robustesse du SVM en introduisant des variables de nuisance (300 dimensions supplémentaires générées aléatoirement).

```
Score sans variable de nuisance
[Données originales] Score entraînement : 1.000 | Score test : 0.911
Score avec variable de nuisance
[Données bruitées] Score entraînement : 1.000 | Score test : 0.516
```

Sans ajout de variables de nuisance, le SVM linéaire obtient un taux de précision parfait à l'entraînement (1.0) et une excellente performance en test (≈ 0.91).

En revanche, lorsque 300 variables aléatoires bruitées sont ajoutées, la performance en test chute fortement (≈ 0.52), alors que l'entraînement reste à 1.0.

Cela met en évidence que l'ajout de dimensions non informatives détériore la capacité de généralisation : le modèle sur-apprend en exploitant le bruit, ce qui entraîne une baisse nette de la précision sur de nouvelles données.

Question 6

Pour atténuer l'effet des variables de nuisance, nous avons appliqué une réduction de dimension par analyse en composantes principales (ACP) sur les données bruitées, afin de ne conserver que les composantes expliquant le plus de variance.

```
Score apres reduction de dimension
[] Score entraînement : 1.000 | Score test : 0.484
```

En pratique, pour de petites valeurs du nombre de composantes, l'algorithme ne converge pas (temps de calcul très long), ce qui nous a conduit à fixer le nombre de composantes à $n = 200$. Avec 200 composantes principales, le modèle conserve un taux de précision parfait à l'entraînement, mais la performance en test reste limitée (≈ 0.484). Ce résultat montre que, bien que l'ACP réduise l'influence des dimensions inutiles, un trop grand nombre de composantes maintient encore une part importante de bruit. Le choix du nombre optimal de composantes est donc crucial pour améliorer la généralisation du modèle.

Question 7

Dans le script fourni, la normalisation est effectuée avant la séparation entre apprentissage et test, comme l'illustre l'extrait de code ci-dessous.

```python
X = (np.mean(images, axis=3)).reshape(n_samples, -1)

X -= np.mean(X, axis=0)
X /= np.std(X, axis=0)

indices = np.random.permutation(X.shape[0])
train_idx, test_idx = indices[:X.shape[0] // 2], indices[X.shape[0] // 2:]
X_train, X_test = X[train_idx, :], X[test_idx, :]
y_train, y_test = y[train_idx], y[test_idx]
images_train, images_test = images[
    train_idx, :, :, :], images[test_idx, :, :, :]
```

Cette approche engendre une fuite d'information, dans la mesure où les statistiques issues du jeu de test (moyenne et variance) interviennent dans la transformation des données d'apprentissage. Une telle erreur introduit un biais et conduit à une surestimation de la performance du modèle, puisque l'indépendance du jeu de test n'est plus respectée. La procédure correcte consiste à ajuster la normalisation uniquement sur l'échantillon d'apprentissage, puis à appliquer la transformation ainsi obtenue aux données de test, garantissant ainsi une évaluation fidèle de la capacité de généralisation du classifieur.

Conclusion

Ce TP nous a permis de mettre en pratique les machines à vecteurs de support sur différents jeux de données. Nous avons observé l'influence du paramètre de régularisation C, l'impact du choix du noyau, ainsi que les limites liées au bruit et au déséquilibre des classes. Nous avons également vu comment des biais dans le prétraitement peuvent fausser l'évaluation des performances. Enfin, l'étude sur les visages a montré l'efficacité des SVM pour la classification d'images, tout en soulignant la nécessité d'un bon choix de paramètres et d'une réduction de dimension adaptée.