HAX907X Apprentissage statistique

TP : Support Vector Machines (SVM)

Marine GERMAIN
Coralie ROMANI DE VINCI

Table des matières

1 Introduction

Les *Support Vector Machines* (SVM) sont une méthode de classification supervisée qui vise à séparer différentes classes en maximisant la marge entre elles. Le principe repose sur la recherche d'hyperplans séparateurs, ce qui les rend particulièrement efficaces dans des espaces de grande dimension.

Dans ce TP, nous mettons en pratique les SVM à l'aide du package scikit-learn. Nous appliquerons ces méthodes à la fois sur des données simulées et sur des jeux de données réels, en particulier le dataset `iris` et une base de données d'images. Nous analyserons également l'influence des noyaux et des hyperparamètres sur la performance des modèles, afin de mieux comprendre cette approche de classification.

2 Base de données `iris`

Cette section est consacrée à l'étude de la base de données `iris` sur laquelle nous ferons une étude de classification sur les classes 1 et 2. Nous considérerons d'abord le noyau linéaire puis le noyau polynomial et nous les comparerons. Le dataset est divisé en deux parties : un ensemble d'entraînement (75% des données) et un ensemble de test (25%).

2.1 Classification avec noyau linéaire

La classification des deux premières variables avec un noyau linéaire a pour objectif de trouver un hyperplan qui sépare au mieux les deux classes en maximisant la marge entre les points des deux classes. Les scores obtenus pour cette méthode sont les suivants :

```
{'C': np.float64(0.013049019780144023), 'kernel': 'linear'}
Generalization score for linear kernel: 0.6933333333333334, 0.64
```

Figure 1: Score obtenu pour le noyau linéaire

Pour les données d'entraînement, le modèle a classifié correctement 69,3% des données, contre 64% pour les données de test.

2.2 Classification avec noyau polynomial

L'utilisation d'un noyau polynomial sur les deux premières variables permet de trouver une frontière de décision non linéaire capable de séparer au mieux les deux classes. L'objectif est d'évaluer si l'emploi d'un noyau polynomial nous permet d'obtenir un meilleur classifieur que celui obtenu précédemment.
Les scores obtenus pour cette méthode sont les suivants :

```
{'C': np.float64(1.0), 'degree': np.int64(1), 'gamma': np.float64(10.0), 'kernel': 'poly'}
Generalization score for polynomial kernel: 0.68, 0.68
```

Figure 2: Score obtenu pour le noyau polynomial

Pour les données d'entraînement et de test, le modèle a classifié correctement 68% des données.

Comparaison des deux méthodes :

- Nous disposons d'abord des scores obtenus en figure ?? et figure ??. On remarque que le score de test obtenu pour le noyau polynomial est supérieur à celui du noyau linéaire : $0.68 \geq 0.64$. La différence reste toutefois relativement faible, ce qui ne permet pas de conclure à une nette supériorité du modèle polynomial.

- Ensuite, nous avons tracé graphiquement ces résultats à l'aide des frontières :

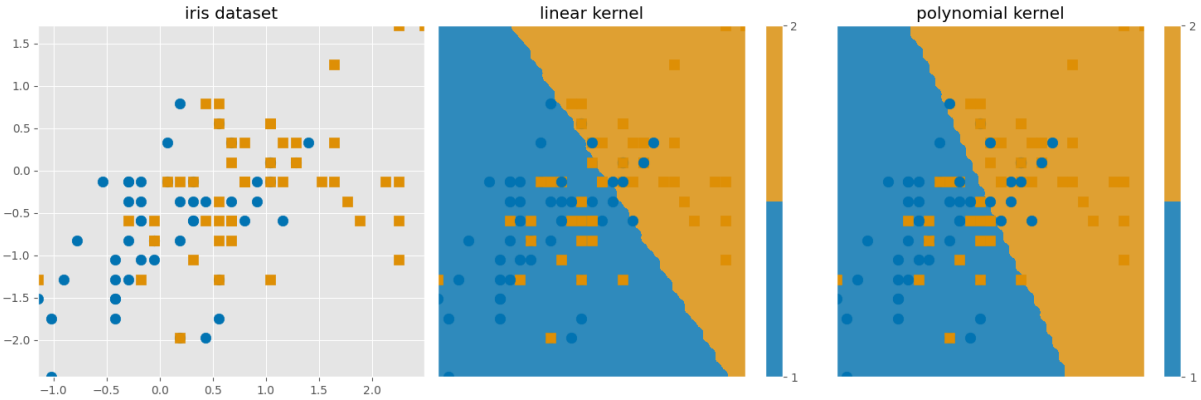
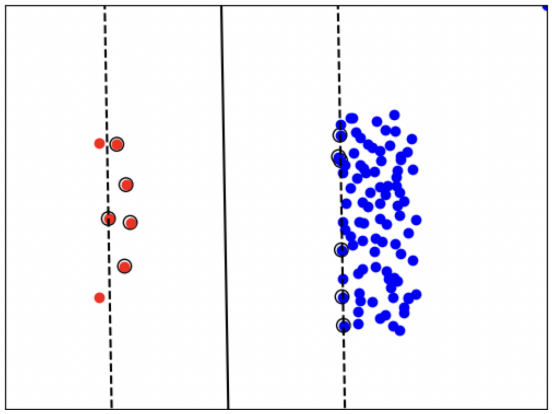


Figure 3: Visualisation de la frontière séparant les classes 1 (en bleu) et 2 (en orange) pour les noyaux linéaire et polynomial

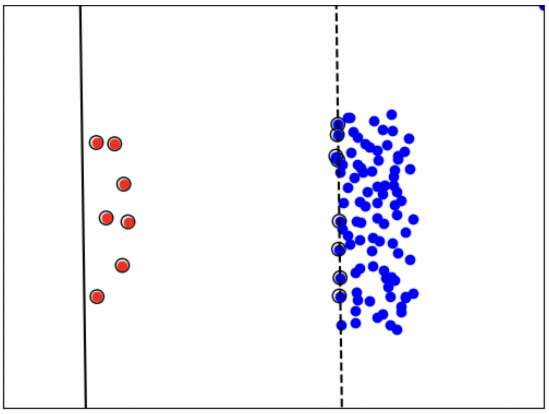
La figure ?? illustre les frontières de décision obtenues par les deux classifieurs appliqués au dataset `iris`. On remarque que la frontière associée au noyau polynomial reste proche d'une séparation linéaire.
En comparant les graphiques, on constate que le classifieur linéaire commet moins d'erreurs sur les données de la classe 1, tandis que le classifieur polynomial en commet moins sur les données de la classe 2.
À ce stade, les performances des deux modèles semblent similaires, sans qu'un noyau ne se démarque nettement de l'autre.

3 SVM GUI

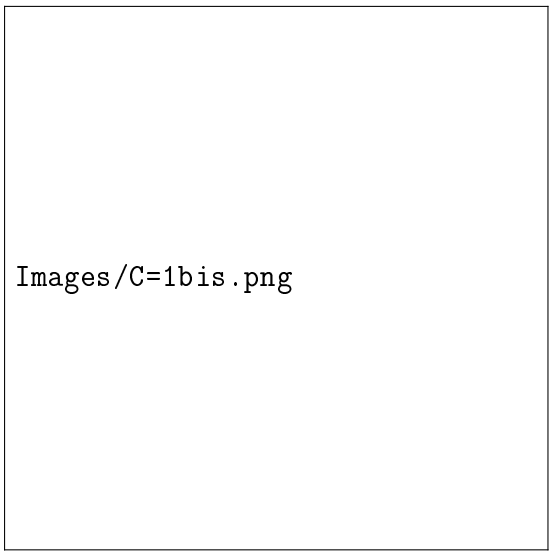
Nous utilisons dans cette section le script `svm_gui.py` permettant de lancer une application qui évalue en temps réel l'impact du choix du noyau et du paramètre de régularisation C. Nous avons généré manuellement une base de données très déséquilibrée (90% vs 10% environ). À l'aide d'un noyau linéaire et en faisant évoluer la valeur de C, nous obtenons la figure **??**.



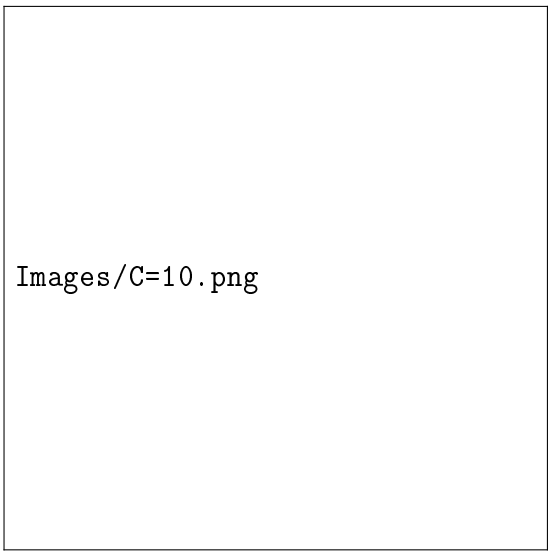
(a) C = 1



(b) C = 0.3



(c) C = 1



(d) C = 10

Figure 4: Visualisation de la région de décision selon différents paramètres de régularisation C

On s'aperçoit ici, de l'influence du paramètre de régularisation C sur la frontière de décision obtenue avec un noyau linéaire. Lorsque C est élevé (figure (a)), le modèle cherche à minimiser les erreurs de classification sur l'ensemble d'entraînement, ce qui conduit à une frontière de décision plus stricte et plus proche des données. À mesure que la valeur de C diminue (figures (b) et (c)), la marge augmente et le modèle accepte davantage de points mal classés afin d'obtenir une séparation plus régulière. On observe ainsi que la diminution de C rend la frontière plus « souple », permettant de mieux généraliser, mais au prix d'un plus grand nombre d'erreurs sur l'échantillon minoritaire. Notons que pour $C = 0.3$ (figure (d)), la frontière se décale complètement et toutes les observations se

retrouvent du même côté. Le modèle adopte alors une solution triviale qui ne sépare plus les deux classes ce qui illustre les limites d'une trop faible valeur de C dans un contexte de données déséquilibrées.

4 Classification des visages

Dans cette section, nous exploitons une base d'images extraites de « Labeled Faces in the Wild ». L'objectif est de classifier, à l'aide d'un SVM à noyau linéaire, deux types d'images : des portraits de Tony Blair (figure **??**) et de Colin Powell.

Figure 5: Base de données de 12 portraits de Tony Blair

4.1 Influence du paramètre de régularisation

Nous souhaitons observer l'influence du paramètre de régularisation C sur la qualité du classifieur. Pour cela, nous estimons le score d'apprentissage en fonction de ce paramètre en figure **??**.

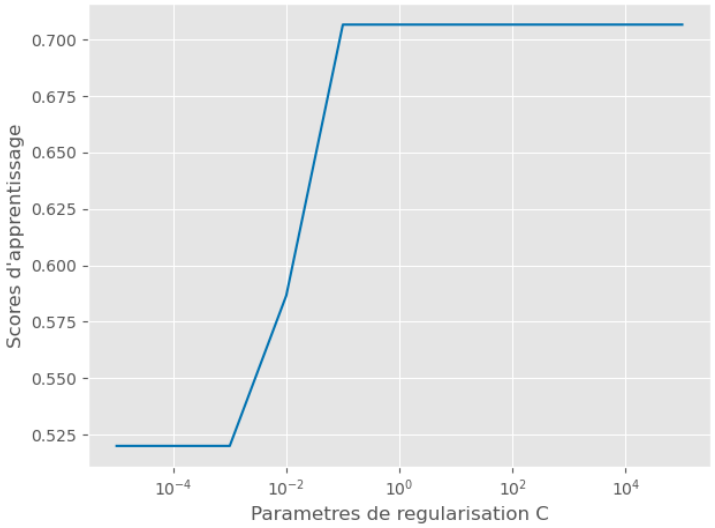


Figure 6: Score d'apprentissage en fonction du paramètres de régularisation C pour des classifieurs SVM à noyau linéaire

En faisant varier les valeurs de C entre 10^{-5} et 10^{5}, on remarque qu'on atteint un score d'apprentissage maximal d'environ 0.725 lorsque $C = 10^{-1}$. Au delà de cette valeur de C, le score reste maximal de façon constante.
Ce résultat est cohérent, car il est attendu qu'un SVM atteigne un score maximal pour une valeur intermédiaire de C. En effet, si C est trop petit il y a un risque un sous-apprentissage et s'il est trop grand, il y a un risque de sur-apprentissage.
Ainsi, le paramètre de régularisation optimal est $C = 0.1$.

```
[0 1 1 1 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1 1 1 1
 0 1 1 0 1 0 0 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 1 0 0 1 1 0 0 1 0
 1 0 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1
 0 0 1 1 0 0 1 0 1 1 0 1 0 1 0 0 0 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0
 1 1 1 1 1]
[0 1 1 1 1 0 1 0 0 1 1 0 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1 1 1 1
 1 0 1 0 1 1 0 1 0 0 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1 0 0 0
 1 0 1 1 0 1 0 0 0 0 0 1 1 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 0 0 1
 0 0 1 1 0 1 1 0 1 1 0 1 0 1 1 0 0 1 1 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 0 1 0
 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0
 1 0 1 1 1]
done in 3.059s
Chance level : 0.6210526315789474
Accuracy : 0.8894736842105263
```

Figure 7: Visualisation de la prédiction, du test et de la précision

La figure **??** illustre les résultats obtenus avec le classifieur SVM linéaire optimal :

- Prédiction : matrice binaires des valeurs prédites pour les images par le classifieur entraîné.
- Test : matrice binaires des valeurs réelle correspondant aux images.
- Chance level : précision attendue si l'on prédisait toujours la classe majoritaire. Cela donne un niveau de référence minimal pour évaluer la performance du classifieur.
- Précision : proportion de prédictions correctes, calculée comme la fraction de valeurs correctement prédites par rapport aux valeurs réelles.

Ainsi, le classifieur linéaire atteint une performance nettement supérieure au niveau de chance, démontrant sa capacité à distinguer correctement les classes sur ce jeu de test.

Figure 8: Prédictions faites à partir du classifieur optimal sur un échantillon d'images

La figure ?? montre un échantillon des résultats obtenues : sur 12 images, 11 ont été correctement prédites.

4.2 Variable de nuisances

Montrons que le score de prédiction est sensible au bruit. Pour cela, nous ajoutons des variables de bruit et recommençons la prédiction. Le bruitage consiste à ajouter des variables de bruit (ici 300), et à les mélanger aux autres.
Nous obtenons alors les valeurs suivantes avec et sans nuisances :

```
Score avec variable de nuisance
Generalization score for linear kernel: 1.0, 0.49473684210526314
```

Figure 9: Score avec variable de nuisance

```
Score sans variable de nuisance
Generalization score for linear kernel: 1.0, 0.9105263157894737
```

Figure 10: Score sans variable de nuisance

Nous constatons que le bruit endommage fortement le classifieur qui voit son score presque réduit de moitié.

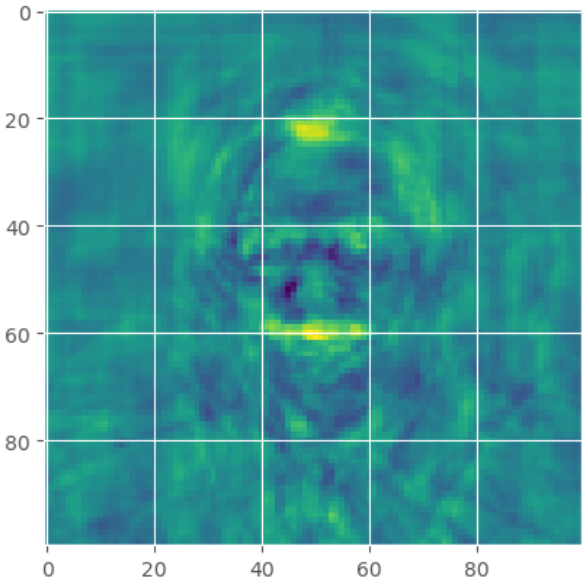


Figure 11: WTF

4.3 Réduction de dimensions

A présent, nous voulons améliorer la prédiction obtenue dans la question précédente à l'aide d'une réduction de dimension.

```
Score apres reduction de dimension
Generalization score for linear kernel: 0.9526315789473684, 0.5421052631578948
```

Figure 12