TP3 N°3 : Support Vector Machine

Abel SILLY

Table of contents

1 Jeu de données Iris

Le but de cette section est de faire une première étude de classification sur le jeu de données iris, on se restreint à la classification entre les espèces *Versicolor* et *Virginica*. On comparera dans cette section les performances entre un noyau linéaire et un noyau polynomial.

1.1 Question 1 : Noyau Linéaire

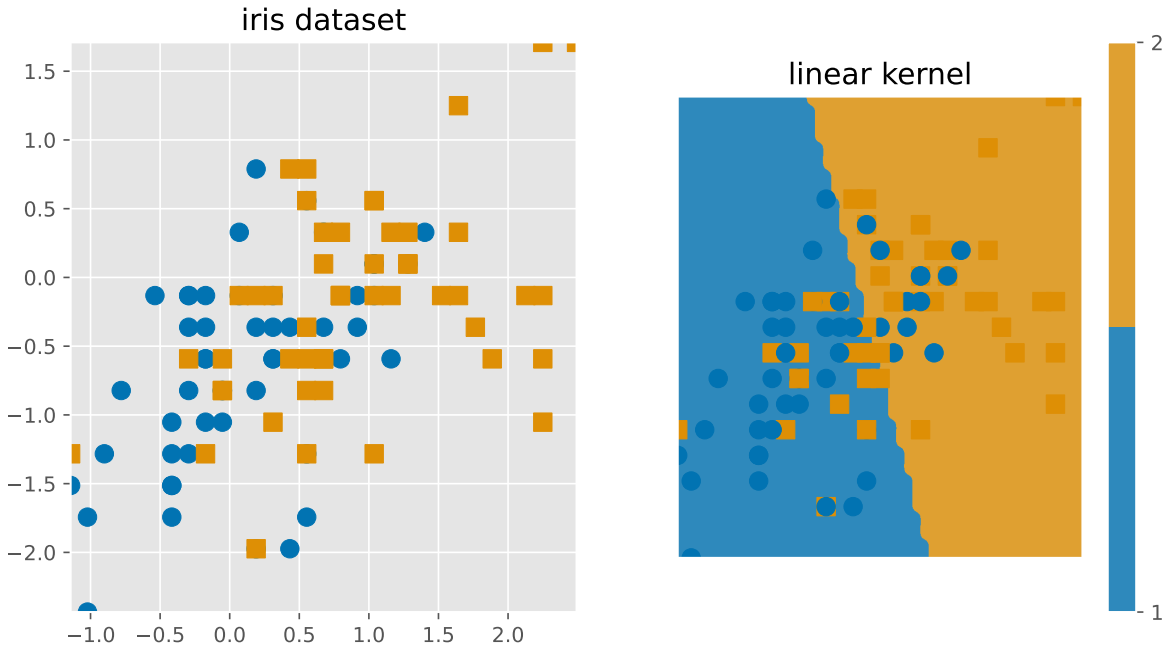
On commence par effectuer une classification des iris en utilisant les deux premières variables et un noyau linéaire.

```
Score : 0.6
Generalization score for linear kernel: 0.7733333333333333, 0.6
```

On obtient alors la partition suivante :

```
Text(0.5, 1.0, 'linear kernel')
```

Représentation de la classification avec un noyau linéaire



1.2 Question 2 : Noyau Polynomial

On s'intéresse maintenant à la même classification mais avec un noyau polynomial.

```
Generalization score for polynomial kernel: 0.6933333333333334, 0.48
```

On observe que sur cet exemple, le noyau linéaire obtient un meilleur score que le noyau polynomial (bien que le noyau polynomial soit plus sofistiqué). A noter que dans ces réalisations nous avons utilisé la valeur par défaut pour l'argument C, c'est-à-dire $C = 1.0$.

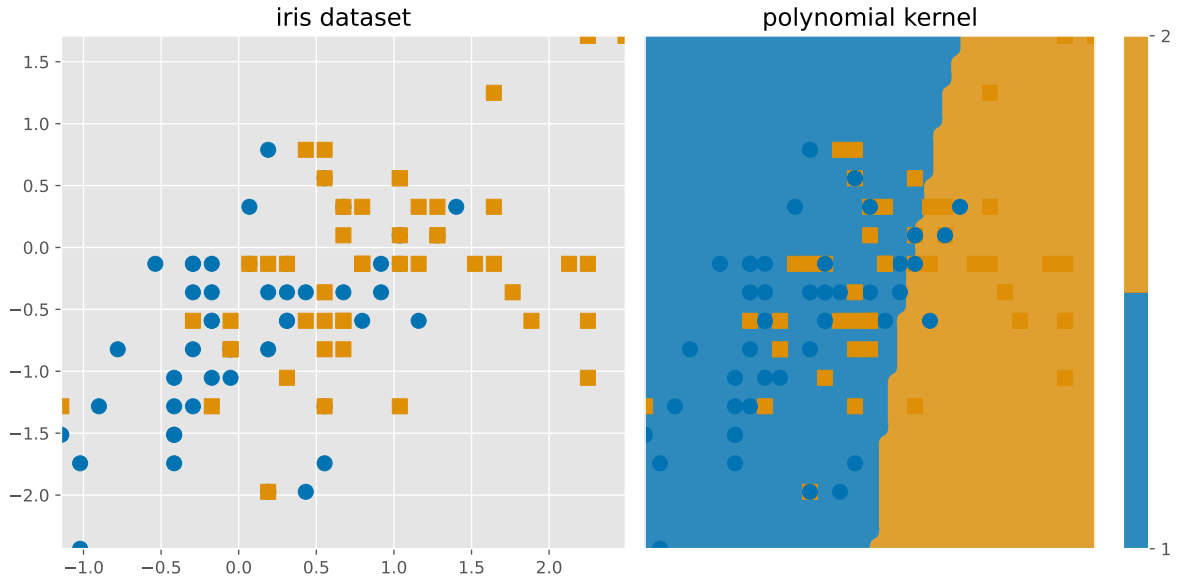


Figure 1: Représentation de la classification avec un noyau polynomial

On voit bien sur ce graphique qu'un ensemble plus important de points oranges sont mal classifiés.

On peut également chercher quels paramètres donneront le meilleur score pour un noyau polynomial et linéaire.

```
{'C': 0.7537688442211055, 'kernel': 'linear'}
Score : 0.6
```

```
{'C': 0.11055276381909529, 'degree': 1, 'gamma': 10.0, 'kernel': 'poly'}
Score : 0.6
```

On observe sur cet exemple que les deux grilles de recherche rendent un score similaire. Le degré optimal du polynome semble être 1, ce qui revient à dire que le noyau est linéaire.

2 SVM GUI

On souhaite maintenant étudier le comportement d'un classifieur linéaire dans un jeu de données déséquilibré en fonction du paramètre C. On affichera la classification pour des valeurs de C égales à 5, 1,0.1 et 0.01.

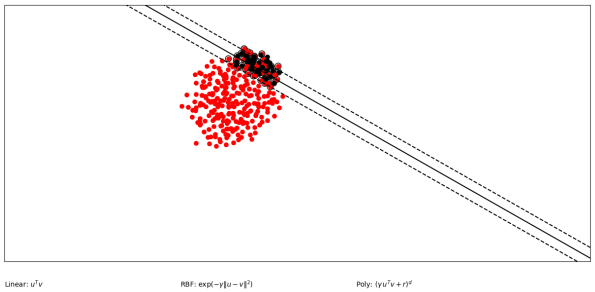


Figure 2: Noyau linéaire et C=1

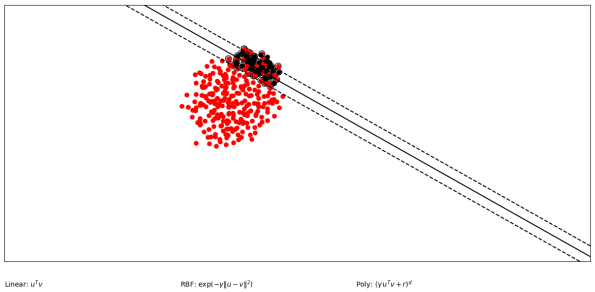


Figure 3: Noyau linéaire et C=0.1

On observe que plus C diminue, moins le classifieur tient compte du groupe de points noir (le classifieur s'écarte petit à petit vers le haut) jusqu'à arriver au cas extrême $C = 0,001$ (cf 5) où tous les points noirs sont mal classifiés. On aimerait donc donner un poids plus important aux erreurs sur la classe de points minoritaires.

3 Classification de visages

Le but de cette partie va être d'effectuer de la classification de visages avec les méthodes SVM sur deux personnes, Tony Blair et Colin Powell. Voici un exemple d'images de Tony Blair issues de notre jeu de données.

Nos prédictions se baseront seulement sur l'illumination (éclairage) des photos.

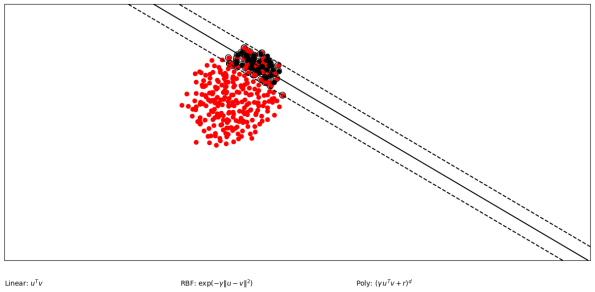


Figure 4: Noyau linéaire et C=0.01

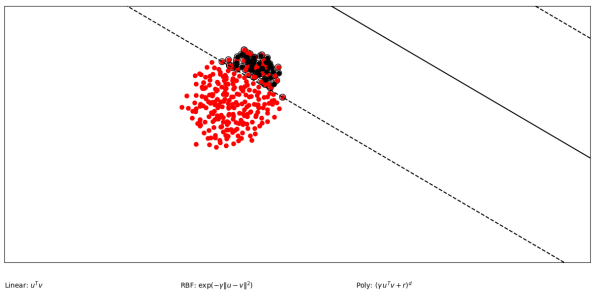


Figure 5: Noyau linéaire et C=0.001

3.1 Question 4 : Influence du paramètre de régularisation

On s'intéresse à l'importance du paramètre de régularisation C, pour cela nous allons représenter l'évolution du score du modèle (à noyau linéaire) en fonction de C où C sera entre 10^{-5} et 10^5 sur une échelle logarithmique.

```
--- Linear kernel ---
Fitting the classifier to the training set
Best C: 0.00042919342601287783
Best score: 0.9315789473684211
done in 13.448s
```

Sur cet exemple on obtient que le meilleur score est d'à peu près 95% et est atteint pour $C \simeq 0.000268$.

On peut maintenant faire une prédiction sur les noms en utilisant le meilleur C, précédement trouvé.

Figure 6: Exemple d'image de Tony Blair issues de la base de données

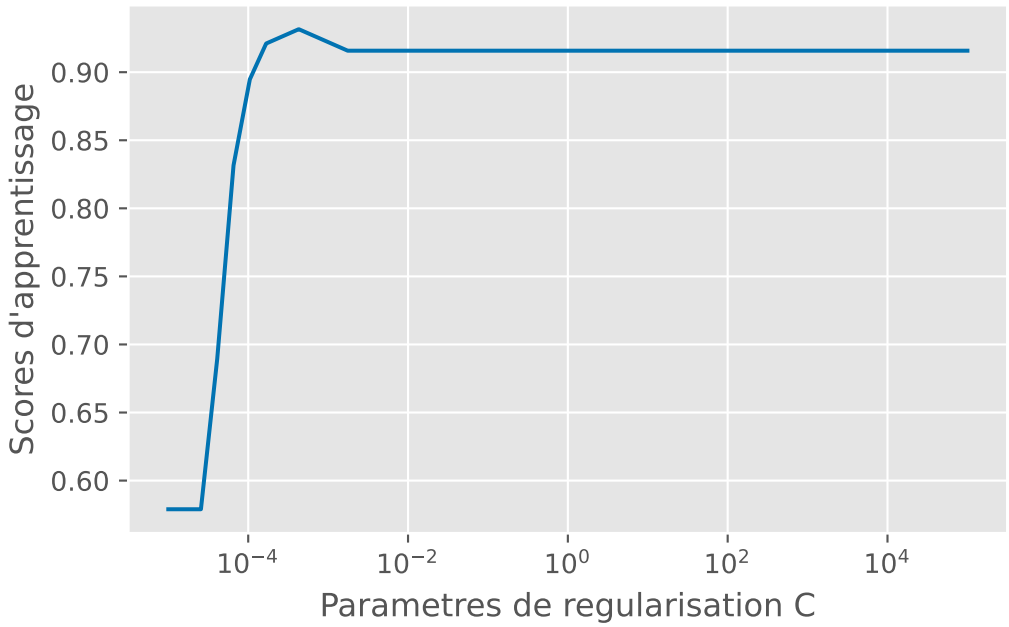


Figure 7: Score obtenu en fonction du paramètre C sur une échelle logarithmique entre 10^{-5} et 10^{5}

```
Predicting the people names on the testing set
Chance level : 0.6210526315789474
Accuracy : 0.9315789473684211
```

On retrouve bien le même score que précédement et on peut le comparer au "niveau de chance" qui représente le score que l'on obtient en prédisant constamment la classe majoritaire. Ce niveau est ici d'à peu près 62%, on gagne donc plus de 30% de précision avec notre méthode.

On peut représenter un exemple d'images et les prédictions associées.

On peut également regarder les coefficients qui importent le plus dans la prédiction, ce qui correspond sur l'image aux zones les plus lumineuses.

3.2 Question 5 : Réduction de performance avec des variables de bruits

On souhaite maintenant observer un effet de sur-paramétrisation en ajoutant des variables de bruit pour la prédiction.

On commence par observer le score obtenu sans variable de bruit.

Figure 8: Exemple d'images et de prédictions associées

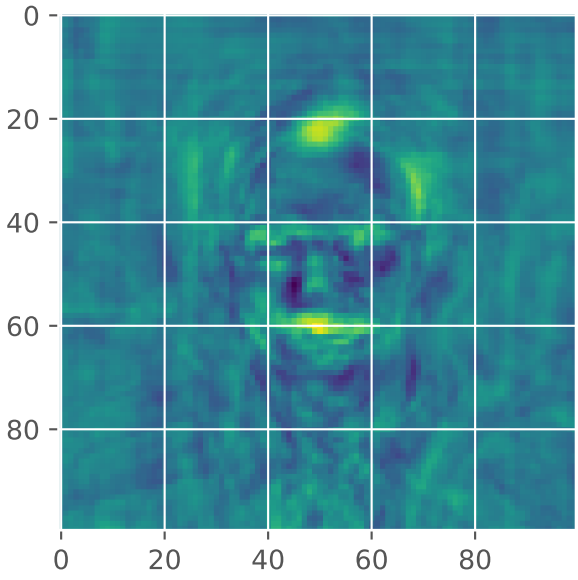


Figure 9: Coefficients les plus informatifs pour la prédiction

```
Score sans variable de nuisance
Generalization score for linear kernel: 1.0, 0.9157894736842105
```

On obtient ici un score de précision de $\approx 92\%$ que nous comparerons avec le score sur les données bruitées.

Nous allons pour commencer rajouter 300 variables gaussiennes centrées réduites.

```
Score avec variable de nuisance
Generalization score for linear kernel: 1.0, 0.9052631578947369
```

Le score obtenu est ici de 90%, ce qui n'est pas si loin du score précédent.

Nous allons donc générer plus de variables gaussiennes avec une variance plus élevée afin de faire baisser le score de façon plus importante.

```
Generalization score for linear kernel: 1.0, 0.7947368421052632
```

On a rajouté ici 5000 variables gaussienne de variance 9, ce qui signifie qu'un tier des variables correspondent à du bruit. On obtient bien cette fois un score de 79% ce qui est bien inférieur au premier score. Avec un tier de variables de bruit supplémentaire on perd à peu près 10% de score.

3.3 Question 6 : Réduction de dimension

On veut maintenant améliorer notre prédiction à l'aide d'une réduction de dimension. Nous allons effectuer une Analyse en Composantes Principales (ACP) sur nos données précemment bruitées, en espérant ainsi éliminer les dimensions de bruit et conserver suffisemment d'information utile pour la prédiction.

Par faciliter l'analyse des résultats on se permet de changer la graine, nous comparerons donc les différents score à une valeur différente des 79% précédement trouvés.

Nous commencerons par une réduction sur 380 et 200 composantes.

```
Score avant réduction :
Generalization score for linear kernel: 1.0, 0.8368421052631579

Score après réduction sur 380 composantes :
Generalization score for linear kernel: 1.0, 0.8842105263157894

Score après réduction sur 200 composantes :
Generalization score for linear kernel: 1.0, 868421052631579
```

Le score avant réduction est de 83%.

On observe donc que le score augmente légèrement après réduction. On gagne un peu plus de 5% de précision avec la réduction sur 380 composantes et un peu plus de 3% avec la réduction sur 200 composantes.

On peut maintenant essayer d'effectuer la réduction sur un autre nombre de composantes.

```
Score après réduction sur 100 composantes
Generalization score for linear kernel: 0.9894736842105263, 0.9210526315789473

Score après réduction sur 50 composantes
Generalization score for linear kernel: 1.0, 0.8315789473684211
```

La réduction sur 100 composantes semble augmenter le score d'une façon assez importante, on gagne en effet 8% de précision sur cette réduction.

Cependant on observe que la réduction sur 50 composantes donne un score similaire à celui de la réduction sur 380 composantes.

Cependant, en éxecutant un certains nombre de fois le code précédent, on observe que les score ont tendance à varier de façon non négligeable. On ne semble donc pas trouver de nombre de dimension significativement meilleur.

On observe tout de même que la réduction donne toujours l'effet attendu, c'est à dire une augmentation du score après réduction.

On arrive donc bien à se séparer du bruit et à garder des dimensions importantes pour la prédiction.