TP n°3 : Support Vector Machine (SVM)

Lucine Bonnefont

Objectif

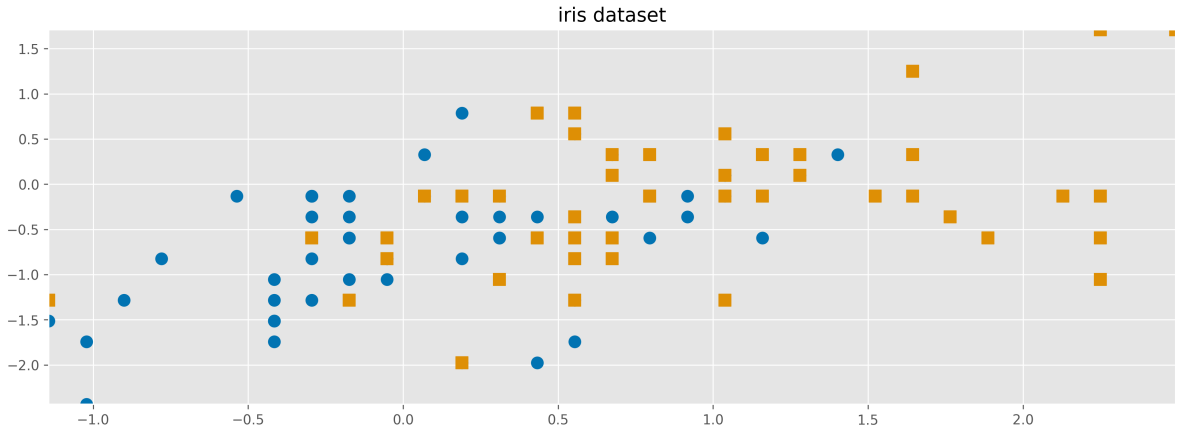
L'objectif de ce TP est de mettre en œuvre la méthode de classification SVM, aussi bien sur des données réelles que simulées, à l'aide de la bibliothèque scikit-learn. Il s'agit également d'apprendre à ajuster ses paramètres (hyperparamètres et choix du noyau) afin d'en maîtriser la flexibilité.

Mise en oeuvre

Dataset "Iris"

Dans cette expérience, l'objectif était de tester un classifieur à vecteurs de support (SVM) linéaire sur le jeu de données Iris, en se concentrant uniquement sur les classes 1 et 2, c'est-à-dire versicolor et virginica. Le dataset Iris contient en réalité trois classes, mais on a volontairement écarté la classe 0 (setosa) car elle est trop facilement séparée. De plus, on n'a retenu que les deux premières variables explicatives, la longueur et la largeur des sépales, afin de travailler dans un espace bidimensionnel et de visualiser plus clairement les limites du classifieur linéaire.

```
Text(0.5, 1.0, 'iris dataset')
```



Pour ajuster le paramètre de régularisation C, qui contrôle l'équilibre entre la largeur de la marge et le respect strict des données d'apprentissage, une recherche sur une grille de valeurs a été menée grâce à la fonction *GridSearchCV*.

```
{'C': np.float64(0.8406652885618325), 'kernel': 'linear'}
Generalization score for linear kernel: 0.7466666666666667, 0.68
```

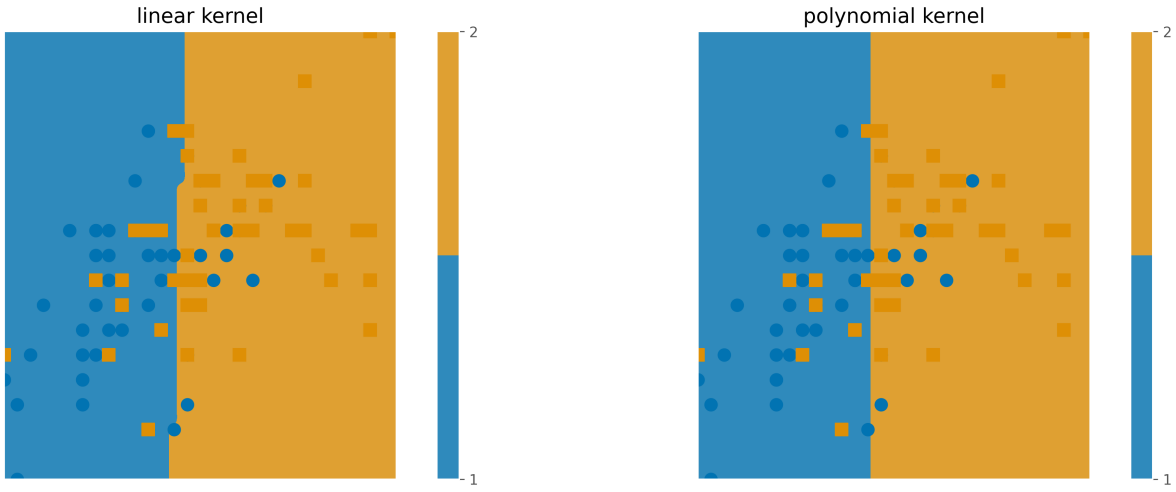
Après entraînement, le meilleur paramètre trouvé est $C = 0,84$. Avec cette valeur, le score obtenu sur l'ensemble d'apprentissage est d'environ 0,75 et de 0,68 pour l'ensemble de test.

Ces résultats signifient que le modèle parvient à faire mieux que le hasard (qui donnerait un taux proche de 0,5) mais reste assez limité. La raison principale tient au fait que les classes versicolor et virginica ne sont pas linéairement séparables. Les nuages de points se chevauchent fortement, si bien qu'aucun hyperplan ne peut tracer une frontière claire entre les deux espèces.

En espérant améliorer la performence du modèle nous allons utiliser un noyau polynomial cette fois ci.

```
{'C': np.float64(0.03162277660168379), 'degree': np.int64(1), 'gamma': np.float64(10.0), 'ker
Generalization score for polynomial kernel: 0.7466666666666667, 0.68
```

Nous retrouvons la même performence que lors de l'utilisation de noyau linéaire. Cette égalité des performances s'explique par le fait qu'un noyau polynomial de degré 1 est en réalité équivalent à un noyau linéaire. Ainsi, pour nos données, la frontière de séparation linéaire semble suffisante et l'usage d'un noyau polynomial plus complexe n'apporte aucun avantage.



SVM GUI

Cette application permet d'évaluer l'impact du choix du paramètre de régularisation C, qui contrôle le compromis entre la maximisation de la marge et la minimisation de l'erreur de classification, appliqué ici à un jeu de données déséquilibré.

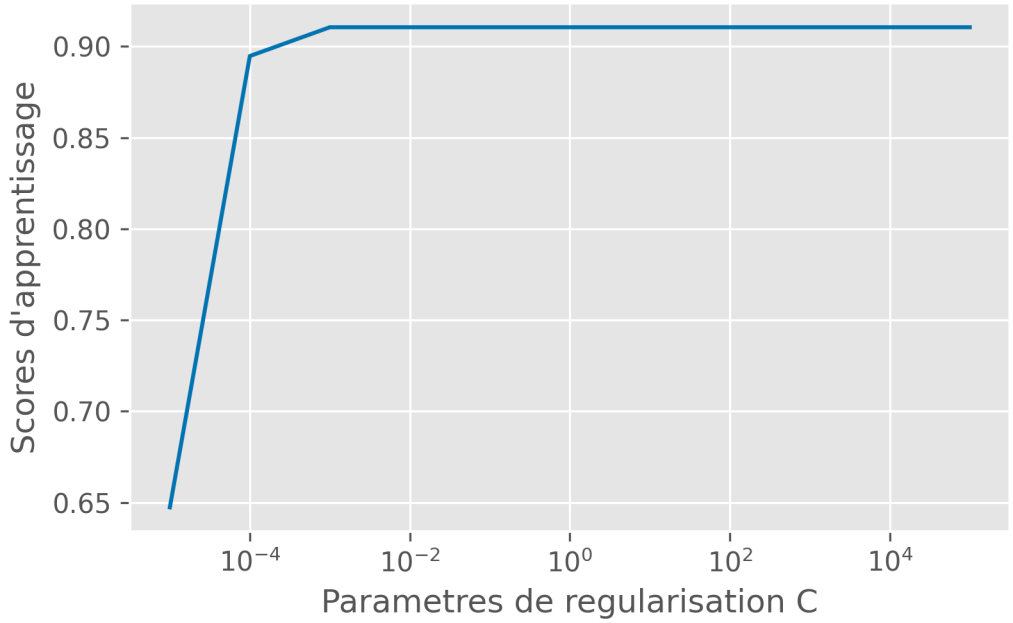
Lorsque C est grand, le SVM cherche à classer correctement tous les points, y compris ceux de la classe minoritaire, quitte à réduire la largeur de la marge. Ce choix peut conduire à une frontière de décision complexe et à un risque de surapprentissage.

À l'inverse, lorsque C est faible, le SVM accepte davantage d'erreurs afin de maximiser la largeur de la marge. Dans un contexte déséquilibré, cela se traduit souvent par une frontière qui favorise la classe majoritaire, car les points minoritaires sont davantage ignorés. Ce modèle est toutefois considéré comme plus robuste au bruit.

Classification de visages

- Lors de l'expérimentation avec un noyau linéaire, nous avons étudié l'influence du paramètre de régularisation C sur la classification des visages de Tony Blair et de Colin Powell. Après apprentissage, le meilleur compromis est obtenu pour une valeur de C égale à 0,001, avec un score de 0,91 sur l'ensemble de test. Ce résultat est largement supérieur au niveau de hasard estimé à 0,62, ce qui confirme la capacité du modèle à apprendre une séparation pertinente entre les deux classes. L'exécution du modèle s'est révélée rapide, avec un temps de calcul inférieur à une seconde, et la précision obtenue illustre l'importance du réglage du paramètre C pour atteindre de bonnes performances tout en évitant le surapprentissage.

```
--- Linear kernel ---
Fitting the classifier to the training set
Best C: 0.001
Best score: 0.9105263157894737
Predicting the people names on the testing set
done in 0.196s
Chance level : 0.6210526315789474
Accuracy : 0.9105263157894737
```



Lorsque le paramètre de régularisation C est très faible, par exemple de l'ordre de 10^{-5}, le SVM privilégie une marge très large et tolère de nombreuses erreurs de classification, ce qui entraîne une frontière de décision trop éloignée des points et une faible précision. Dès que C augmente légèrement, le modèle devient plus strict dans la classification des points d'entraînement, ce qui améliore rapidement la séparation entre les deux classes et stabilise la performance à un niveau élevé. Ainsi, la valeur de C contrôle bien l'équilibre entre la largeur de la marge et la fidélité aux données, et un réglage trop faible empêche le modèle de capturer correctement les distinctions entre les visages.

- Nous avons ensuite étudié l'impact de l'ajout de variables de nuisance sur la performance d'un SVM linéaire. Sans variables de nuisance, le modèle atteint un score parfait sur l'ensemble d'entraînement et une précision de 0,91 sur l'ensemble de test, ce qui montre que le SVM est capable de séparer efficacement les visages de Tony Blair et Colin Powell.
- En ajoutant 300 variables aléatoires supplémentaires, indépendantes de la cible, la performance sur l'ensemble de test chute drastiquement à 0,53, tandis que le score sur

l'ensemble d'entraînement reste à 1,0. Cela illustre que l'introduction de variables non informatives augmente la complexité du modèle et provoque un surapprentissage, réduisant sa capacité de généralisation. L'expérience met donc en évidence la sensibilité des SVM linéaires à la présence de variables inutiles lorsque le nombre de caractéristiques augmente par rapport au nombre d'exemples.

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

Score avec variable de nuisance
Generalization score for linear kernel: 1.0, 0.531578947368421
```

- Après application d'une réduction de dimension par PCA, la performance du SVM linéaire sur les données bruitées est légèrement diminuée sur l'ensemble d'entraînement et reste faible sur l'ensemble de test, même si elle semble améliorer légèrement la performence. Cette expérience devrait montrer que la PCA permet de compresser l'information et de réduire la complexité liée aux variables de nuisance en conservant les composantes principales les plus informatives. Dans notre cas, nous ne pouvons pas réduire conséquemment le nombre de composantes principales par problème de coût de calcul, ce qui explique ce score très proche de celui avec l'ajout de bruit, encore beaucoup de composantes conservées ne sont que du bruit.

```
Score après réduction de dimension
Generalization score for linear kernel: 0.8789473684210526, 0.5684210526315789
```

- Lors du prétraitrement des données on fait la moyenne sur les canaux de couleur(RGB) ce qui entraine une réduction de l'image à une seule intensité par pixel, en supprimant toute information relative à la couleur. Certaines différences pouvant être discriminantes entre les visages (peau, cheveux, vêtements) sont perdues. Cela peut biaiser le modèle vers les traits de luminosité uniquement.