TP3

November 28, 2019

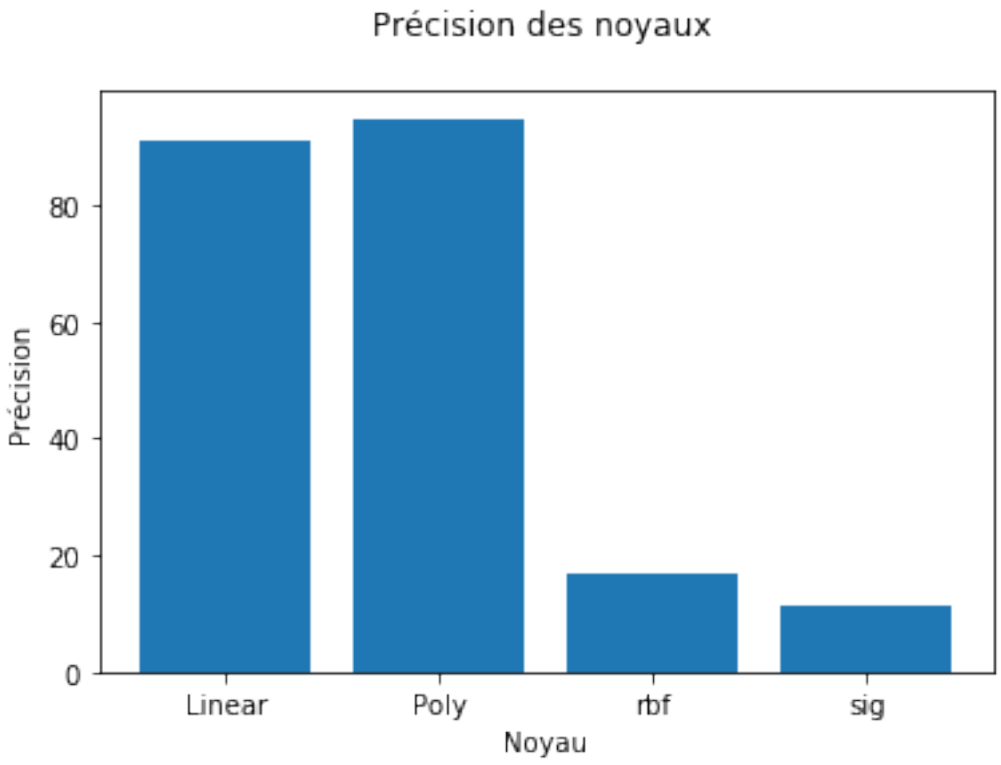
Apprentissage par Machines à Vecteurs de Support

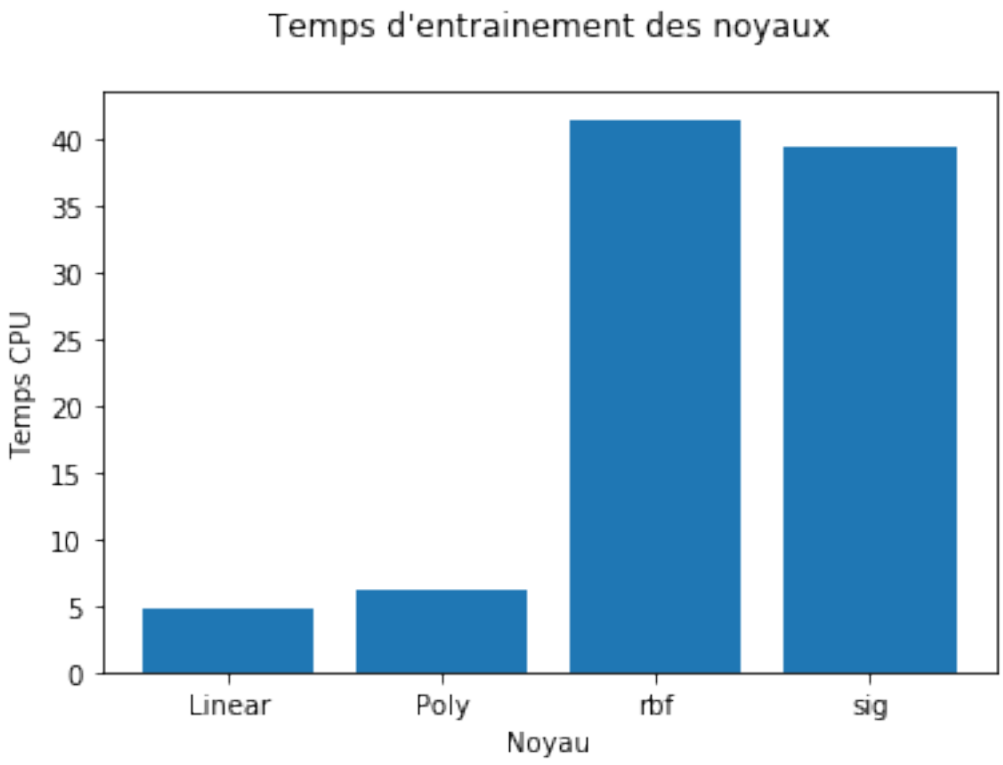
Construire un modèle de classification ayant comme paramètres un noyau linear

```
precision score linear = 90.85714285714286
```

Tentez d'améliorer les résultats en variant la fonction noyau : 'poly', 'rbf', 'sigmoid', 'precomputed'

```
precision score poly = 94.80952380952381
precision score rbf = 16.952380952380953
precision score sigmoid = 11.285714285714285
```





Faites varier le paramètre de tolérance aux erreurs C

```
precision score for c = 0.2 = 94.80952380952381
precision score for c = 0.4 = 94.80952380952381
precision score for c = 0.6 = 94.80952380952381
precision score for c = 0.8 = 94.80952380952381
precision score for c = 1.0 = 94.80952380952381
```

Tracez la courbe d'erreur de classification sur les données d'entrainement et de test en fonction de C En variant la tolérance aux erreurs C, on obtient toujours les mêmes résultats. Ainsi l'erreur de classification sur les données est la même

Construisez la matrice de confusion en utilisant le package metrics

```
CM linéaire :

[[196   0   0   0   0   2   0   1   1   0]
 [  0 233   1   2   0   0   0   0   1   0]
 [  2   4 187   1   3   1   2   1   1   0]
 [  1   1   6 190   0   9   0   2   4   2]
 [  1   2   3   0 179   0   2   2   0   9]
```

```
 [  4   4   2  10   0 156   2   0   2   0]
 [  1   1   6   0   1   2 196   0   1   0]
 [  0   3   3   2   1   0   0 228   0  10]
 [  3   9   4   7   1  11   4   0 168   5]
 [  0   2   2   3  10   1   0   6   2 175]]
```

CM poly :

```
[[195   1   0   0   0   1   1   1   1   0]
 [  0 232   1   1   1   0   0   0   0   2]
 [  2   1 195   3   0   0   1   0   0   0]
 [  2   1   4 200   1   3   0   1   3   0]
 [  0   1   1   0 186   1   1   0   0   8]
 [  2   5   1   2   0 169   1   0   0   0]
 [  0   0   0   0   0   1 205   0   2   0]
 [  0   4   2   1   0   0   0 234   0   6]
 [  1   3   1   4   1   2   2   0 194   4]
 [  0   3   1   2   4   1   0   8   1 181]]
```

CM rbf :

```
[[  8 192   0   0   0   0   0   0   0   0]
 [  0 237   0   0   0   0   0   0   0   0]
 [  0 186  16   0   0   0   0   0   0   0]
 [  0 201   0  14   0   0   0   0   0   0]
 [  0 182   0   0  16   0   0   0   0   0]
 [  0 168   0   0   0  12   0   0   0   0]
 [  0 194   0   0   0   0  14   0   0   0]
 [  0 236   0   0   0   0   0  11   0   0]
 [  0 199   0   0   0   0   0   0  13   0]
 [  0 186   0   0   0   0   0   0   0  15]]
```

CM sig :

```
[[  0 200   0   0   0   0   0   0   0   0]
 [  0 237   0   0   0   0   0   0   0   0]
 [  0 202   0   0   0   0   0   0   0   0]
 [  0 215   0   0   0   0   0   0   0   0]
 [  0 198   0   0   0   0   0   0   0   0]
 [  0 180   0   0   0   0   0   0   0   0]
 [  0 208   0   0   0   0   0   0   0   0]
 [  0 247   0   0   0   0   0   0   0   0]
 [  0 212   0   0   0   0   0   0   0   0]
 [  0 201   0   0   0   0   0   0   0   0]]
```

A votre avis, quels sont les avantages et les inconvénients du SVM ?

Avantages :

- Il existe plusieurs fonctions de noyaux qui peuvent améliorer les résultats. Cela dépend du nombre des classes dans le jeu de données et de sa taille.
- Le noyau linéaire donne un résultat rapide avec une bonne performance.
- Nécessite moins de jeu de données que le réseau de neurones

Inconvénients :

- Les noyaux rbf et sigmoid ont de faibles performances (du moins dans notre cas)
- Ces deux noyaux nécessitent beaucoup de temps de calcul
- Nécessite plus de jeu de données que kNN

A votre avis, quel est le meilleur classifieur et quels sont ses hyperparamètres finaux ? Selon les cas, un algorithme peut être meilleur qu'un autre.

En effet, les réseaux de neurones sont plus performants que d'autres méthodes si, il existe vraiment beaucoup de données et leur structure est complexe. En particulier, si vous souhaitez résoudre un problème de classification, vous avez besoin de nombreux exemples par classe.

En revanche, le classifieur kNN n'a pas besoin de beaucoup d'exemples par classe. Donc, si vous avez un problème de classification avec quelques exemples d'entraînement par classe, kNN aurait probablement une meilleure performance que d'autres méthodes.

Pour obtenir une bonne performance, SVM nécessite beaucoup plus de données d'entraînement que kNN, mais moins que le réseaux de neurones. SVM linéaire peut être entraîné plus rapidement, mais il est moins précis que les approches non linéaires. Son temps d'entraînement est également plus long que celui du cas linéaire.

Cela dit, les algorithmes d'entraînement pour SVM ont de meilleures garanties. Les réseaux de neurones sont plus difficiles et nécessitent parfois un peu d'entretien.