DU Python Big Data

Machine Learning

Partie 3

gilles.michel@sudintralog.com
@unimes.fr

Principe

Mettre en œuvre des algorithmes améliorant **automatiquement** les performances d'un système

Plan

I. Régressions
II. Arbres de décision
III. Forêt aléatoire et apprentissage d'ensemble
IV. SVM (Support Vector Machine)

II. Arbres de décision pour régression

Un arbre de décision peut être utilisé pour effectuer une régression (non linéaire) en escalier (méthode CART)

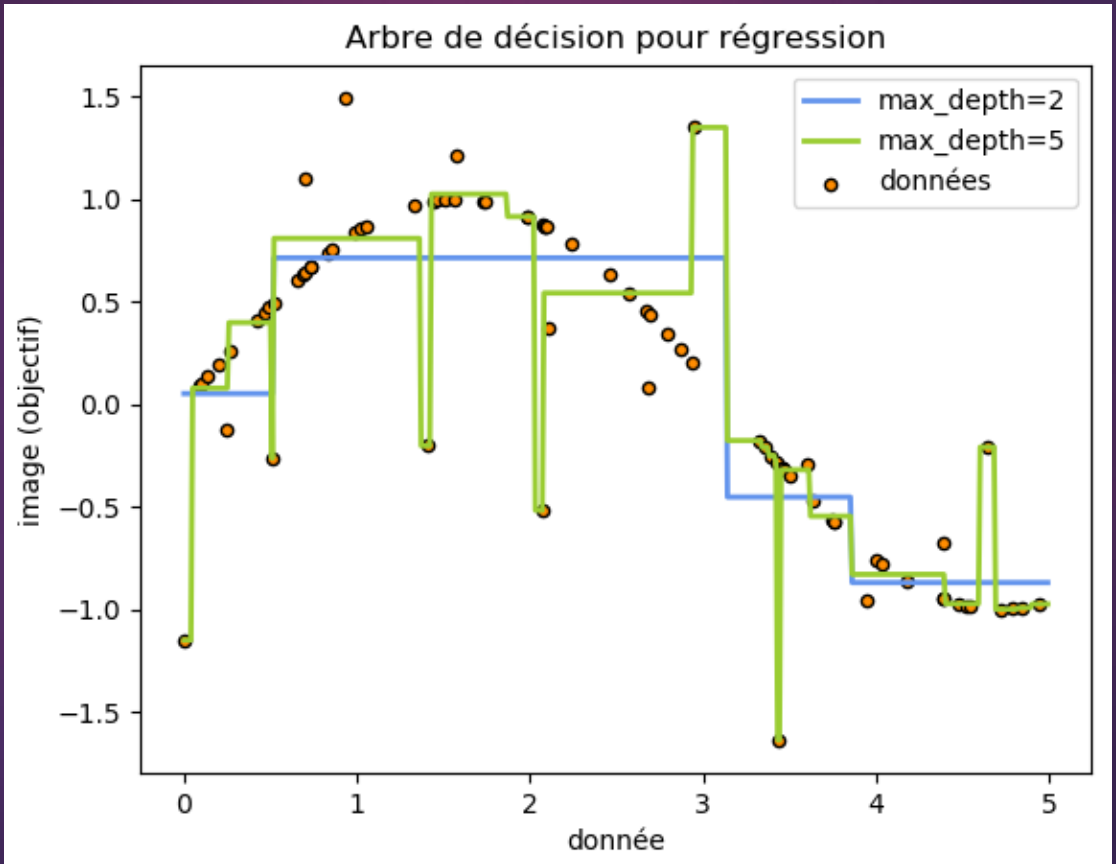
➡ **classe tree.DecisionTreeRegressor(max_depth=…)**

L'arbre détermine les classes où la sortie a une faible variance et donne comme valeur de classe la moyenne des sorties des individus de la classe considérée

Il faut que les données de sortie fournies soient des float (avec une partie décimale, même nulle)

Régression par arbre de décision

En sortie d'une régression par arbre de décision, on obtient une fonction en escalier (la profondeur max_depth correspond à la précision de la régression)



Exemple de code minimal pour régression

from sklearn import tree

X = [[0, 0], [1, 1]] → BDD d'apprentissage

Y = [0.5 , 3.0] → image de chaque individu

clf = tree.DecisionTreeRegressor()

clf = clf.fit(X, Y) → démarre l'apprentissage

clf.predict([[2., 2.]]) → image d'un nouvel individu

tree.plot_tree(clf.fit(X, Y)) → représente l'arbre

Plan

I. Régressions
II. Arbres de décision
III. Forêt aléatoire et apprentissage d'ensemble
IV. SVM (Support Vector Machine)

Forêt aléatoire (random forest)

- Algorithme proposé en 1995 par Ho et détaillé en 2001 par Breiman et Cutler

Constat :

- les arbres de décision sont sensibles à l'ordre des prédicteurs (variables dans les tests)

➔ On calcule différents arbres basés uniquement sur une partie des variables (en général moins que $\sqrt{nb\_variables}$)

➔ Algorithme très efficace quand *nb_variables* est grand

Principe de bagging

Bagging = agrégation de modèles

- Les forêts aléatoires fournissent plusieurs arbres de décision ➔ plusieurs prédictions différentes pour chaque individu
- Nécessité de regrouper toutes ces prédictions en une seul ➔ agrégation (bagging)

Méthodes de bagging

Comment obtenir la prédiction finale pour un individu à partir de plusieurs arbres de décision ?

- Pour une classification : on choisit la catégorie la plus fréquente
- Pour une régression : on fait la moyenne des valeurs prédites

Forêts aléatoires avec Scikit-Learn

- sklearn.ensemble.RandomForestClassifier
- sklearn.ensemble.RandomForestRegressor

Plan

I. Régressions
II. Arbres de décision
III. Forêt aléatoire et apprentissage d'ensemble
IV. SVM (Support Vector Machine)

IV. SVM

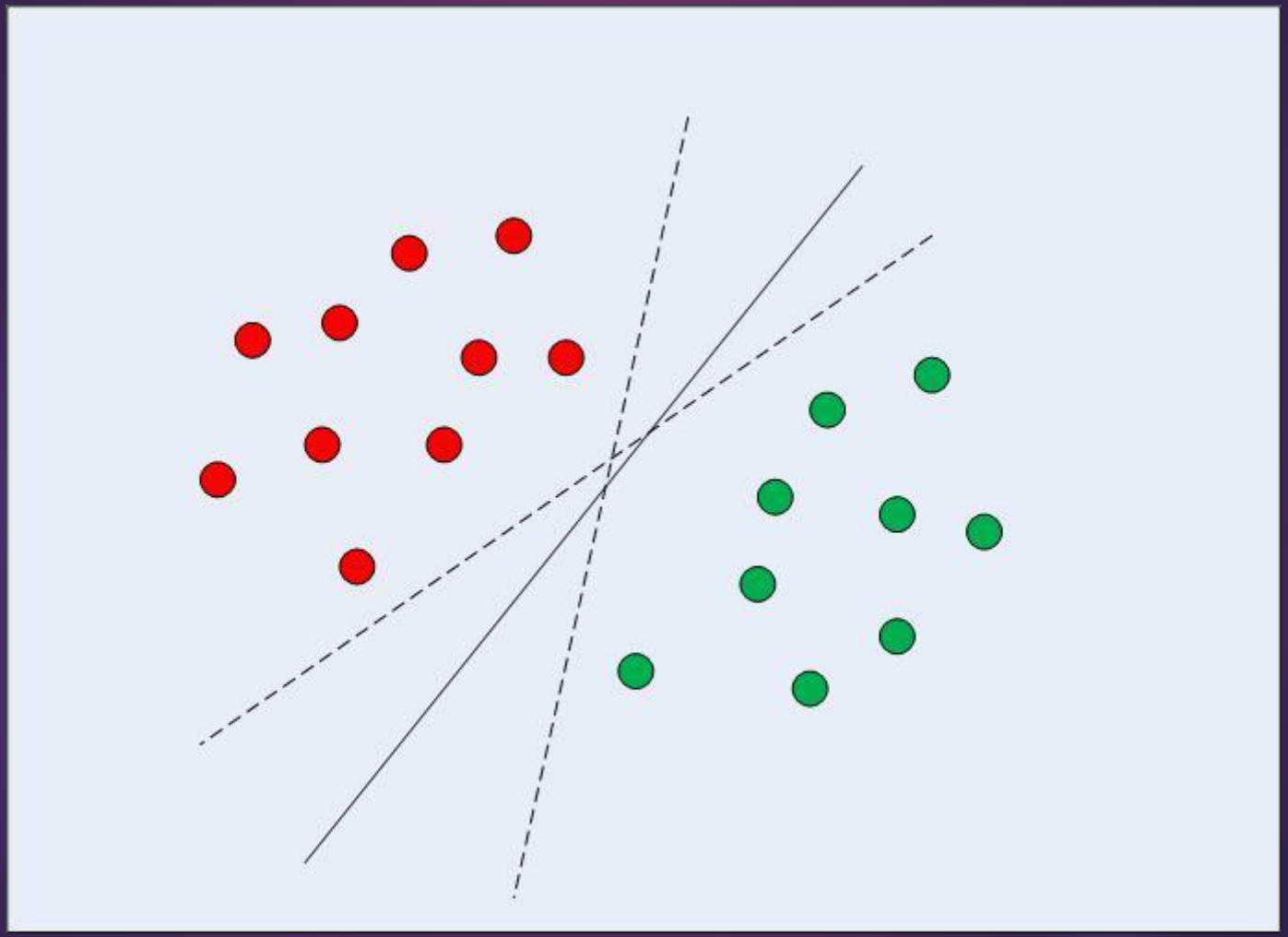
- Support Vector Machine
  (Machine à vecteurs de support (de la marge))

  ou

- Séparateur à Vaste Marge

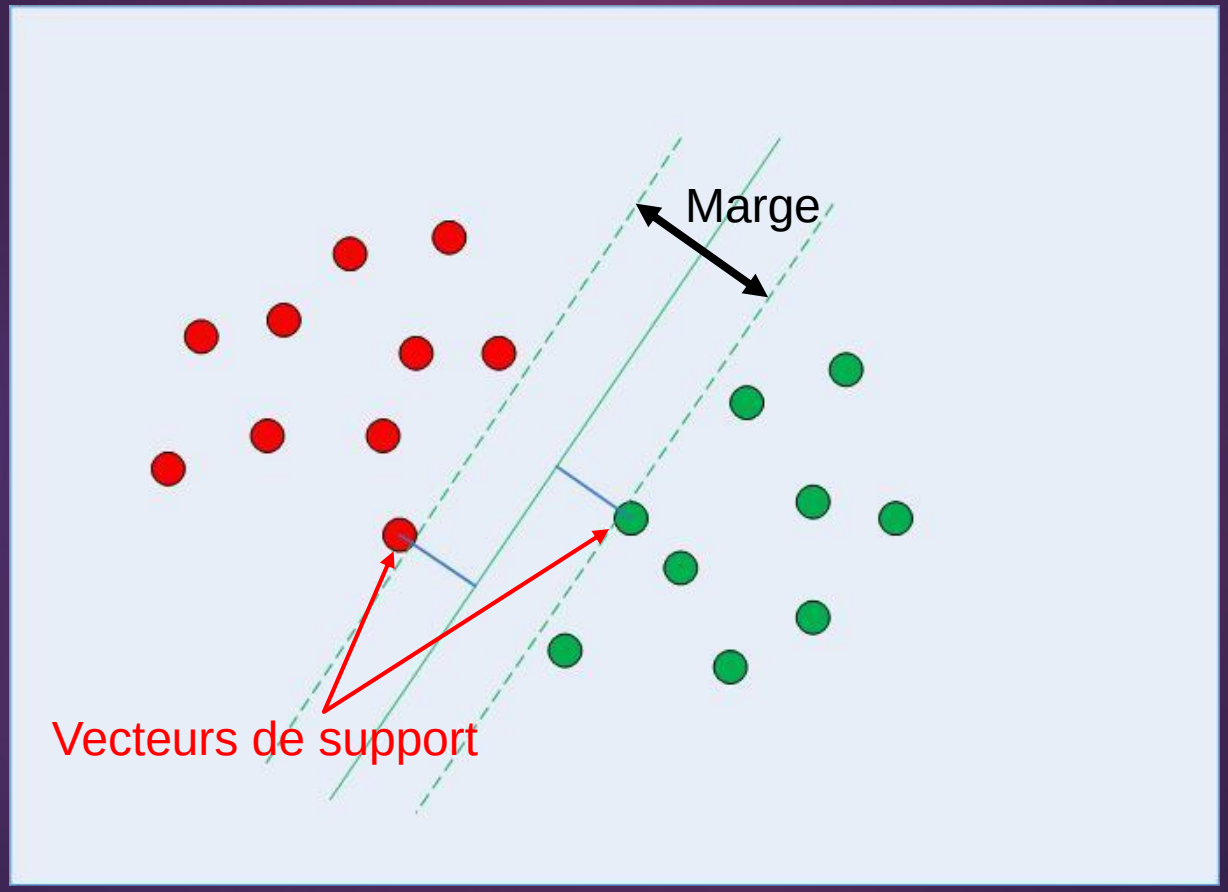
- Algorithme d'apprentissage supervisé

Plusieurs séparateurs



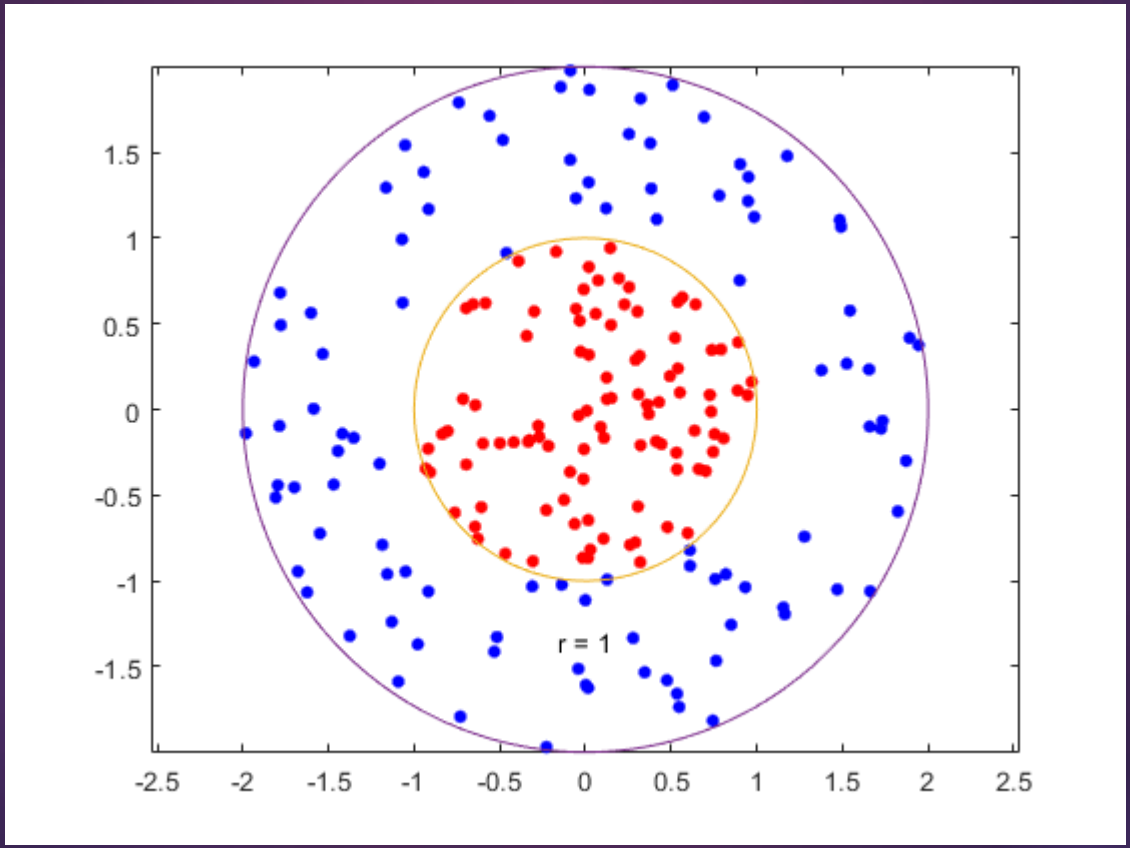
Séparateur optimal

- Maximiser la marge (distance orthogonale entre 2 individus de classes distinctes = les vecteurs de support)



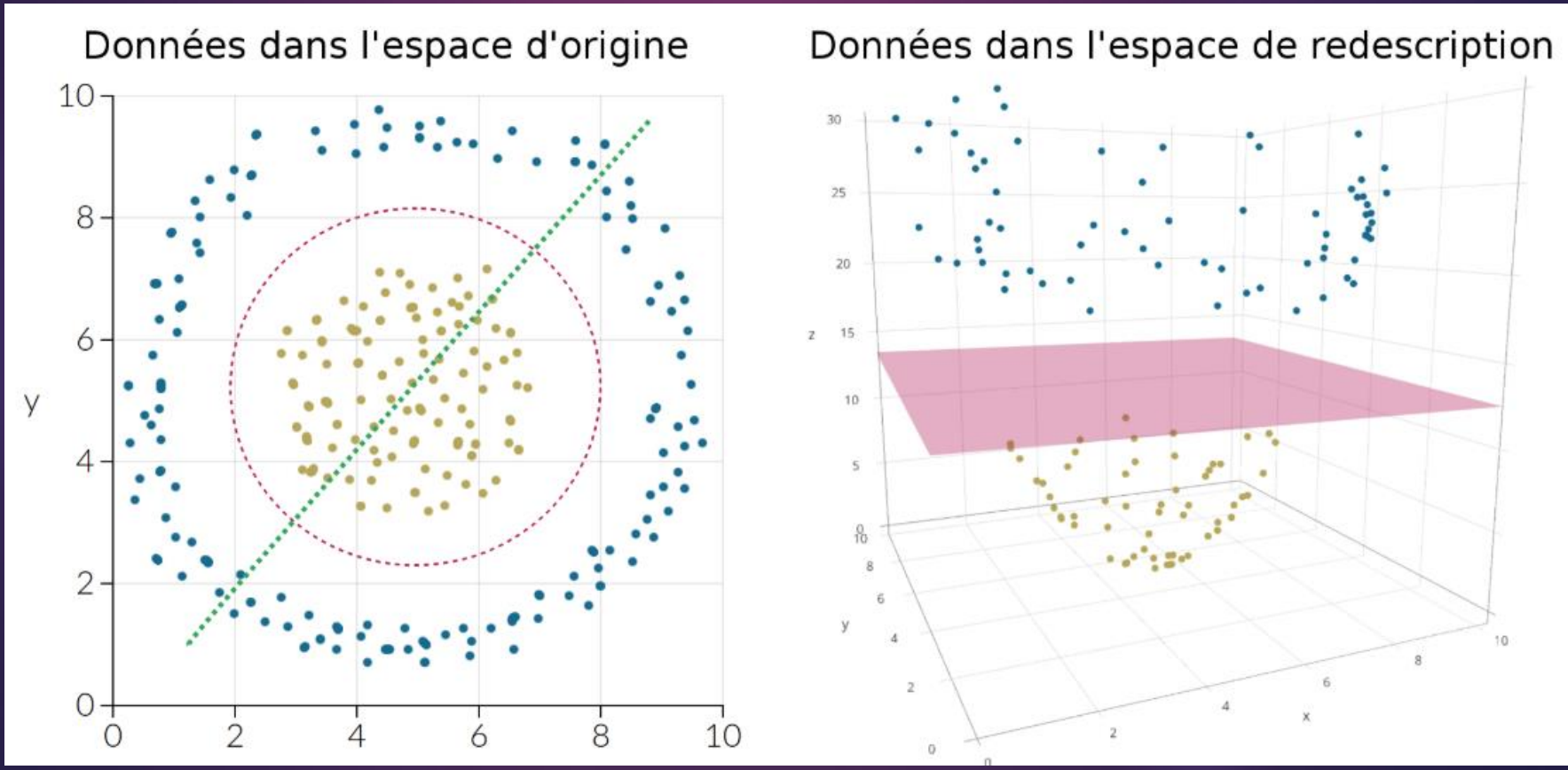
Discrimination non linéaire

- Utilisation d'une fonction noyau qui « redresse » un problème quadratique en un problème linéaire (souvent dans un espace de dimension supérieure)



Utilisation d'une fonction noyau

Exemple par transformation polynomiale du type $P(x, y) = x^2 + y^2$ (fonction distance²)



SVM sous Python

- Classe svm de Scikit-Learn
- Sous-classes svm.SVC pour une classification et svm.SVR pour une régression

Exemple de code


```
from sklearn.svm import SVC # classification
from sklearn.svm import SVR  # régression

classifier = SVC(kernel='linear')
# ou (kernel='poly', degree=5)
# ou (kernel='rbf') pour un noyau gaussien
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
# idem pour SVR
```

Evaluation du modèle SVM

- Classe metrics de sklearn

from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))

```
[[152    0]
 [  1  122]]
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       152
           1       1.00      0.99      1.00       123

avg / total       1.00      1.00      1.00       275
```