Apprentissage automatique
Chapitre 2 - Apprentissage supervisé et non supervisé

Halim Djerroud

LISV
Laboratoire d'ingénierie
des systèmes de Versailles

UVSQ
université PARIS-SACLAY

révision : 0.1

Plan du cours et déroulement

Plan du cours

1. Introduction.
2. Les Données.
3. **Apprentissage supervisé et non supervisé.**
4. Les réseaux de neurones.

Déroulement

- 18 heures de cours 6 séances de 3 heures.
- Deux contrôles continus (QCM).
- Un projet à faire en binôme.
- Un examen écrit.

Approche du machine learning

Apprentissage supervisé :

- Régression linéaire simple
- Régression linéaire multiple
- Équation normale
- Régression polynomiale
- Modèles linéaires régularisés
- Réseau de neurone
- Machine à vecteurs de support linéaire et non linéaire (SVM)

Apprentissage non supervisé :

- K-NN, K-MEANS
- Arbre de décision

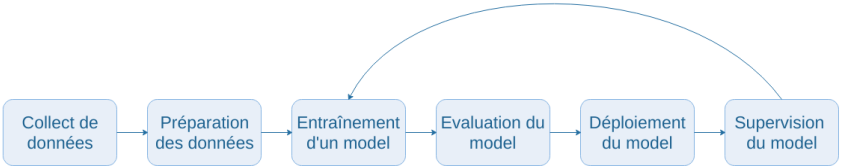
Analyse des séries chronologiques :

- Modèle ARIMA

Apprentissage par renforcement :

- Processus de décision markovien

Quelle est l'approche machine learning ?



1. Analyser les données,
2. Choisir un modèle,
3. Les modèles sont entraînés avec des données (data mining),
4. Estimer l'erreur du modèle,
5. Mettre à jour le modèle.

Contraintes:

- Les données doivent être de très bonne qualité
- Le volume des données est important pour entraîner le modèle

Les types d'apprentissage

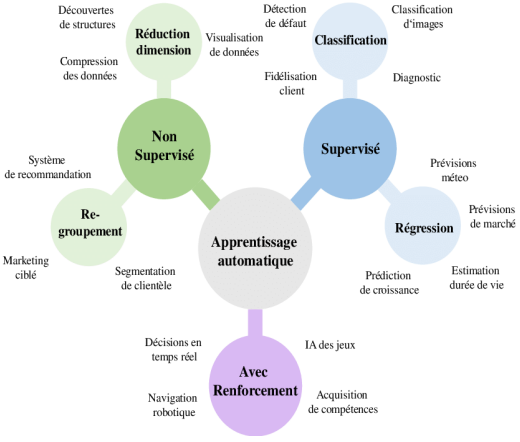


Figure: Source web

Apprentissage supervisé

Apprentissage supervisé : Régression linéaire

Objectifs du chapitre :

1. Régression linéaire

Régression linéaire

Définition

Une régression linéaire est un modèle qui cherche à établir une relation linéaire entre une variable, dite expliquée, et une ou plusieurs variables, dites explicatives.

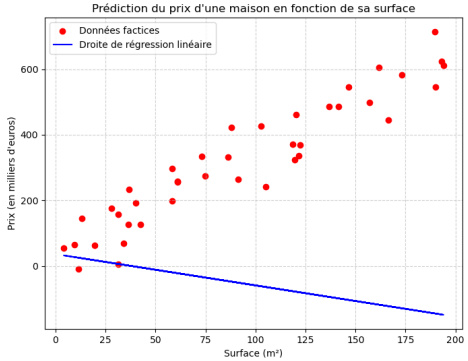
- La régression linéaire est une méthode statistique utilisée pour modéliser la relation entre une variable dépendante (variable cible) et une ou plusieurs variables indépendantes (prédicteurs).
- L'objectif est de trouver la meilleure droite qui prédit la variable dépendante en fonction des variables indépendantes.
- Permet de modéliser la relation entre une variable dépendante Y et une variable indépendante X.
- L'équation du modèle est : $Y = aX + b$
- Les paramètres a (pente) et b (ordonnée à l'origine) sont estimés par *la méthode des moindres carrés*.

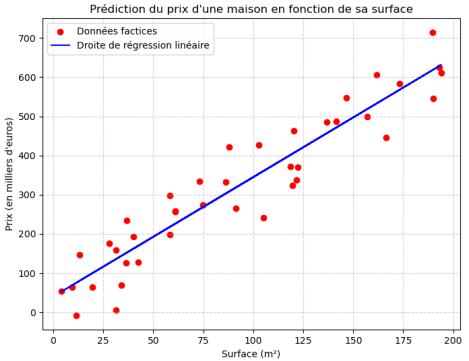
Hypothèses principales

- **Linarité** : La relation entre la variable dépendante et les variables indépendantes est linéaire.
- **Indépendance des erreurs** : Les résidus doivent être indépendants.
- **Homoscédasticité** : La variance des erreurs est constante pour toutes les valeurs des variables indépendantes.
- **Normalité des erreurs** : Les erreurs suivent une distribution normale.
- **Absence de multicollinéarité** : Les variables indépendantes ne doivent pas être trop fortement corrélées entre elles.

Exemple

Prédire le prix d'une maison en fonction de sa surface.





Le but est d'ajuster les coefficients a et b pour aligner la droite le mieux possible.

LISV Laboratoire d'Ingénierie des systèmes de Versailles — UVSQ université PARIS-SACLAY

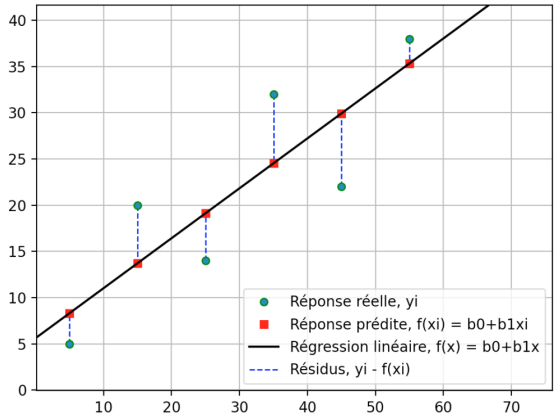
Formulation mathématique

- **Forme générale du modèle** :

$$y = \beta_0 + \beta_1 x + \varepsilon \tag{1}$$

- y : variable dépendante (prédiction)
- x : variable indépendante (prédicteur)
- β_0 : intercept (ordonnée à l'origine)
- β_1 : pente de la droite
- ε : erreur aléatoire

Représentation graphique



- La droite de régression minimise la somme des carrés des résidus.

Méthode des moindres carrés

- La méthode des moindres carrés consiste à minimiser la somme des carrés des résidus $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.
- Les formules pour les coefficients sont :

$$a = \frac{COV(X;Y)}{V(X)} = \frac{n\sum(x_i y_i) - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

$$b = \bar{Y} - a\bar{X} = \frac{\sum y_i - a\sum x_i}{n}$$

$$y = aX + b$$

Problèmes avec la méthode des moindres carrés

- **Sensibilité aux valeurs aberrantes** : Les valeurs extrêmes peuvent avoir un impact significatif sur la droite de régression.
- **Supposition de linéarité** : La méthode suppose que la relation entre les variables est linéaire.
- **Multicolinéarité** : La présence d'une forte corrélation entre les variables indépendantes peut rendre l'estimation des coefficients instable.
- **Hétéroscédasticité** : Lorsque la variance des erreurs n'est pas constante, les coefficients peuvent être biaisés.
- **Non-normalité des résidus** : Si les résidus ne sont pas normalement distribués, les tests statistiques basés sur cette hypothèse peuvent être invalides.
- **Inversion de matrice coûteuse** : La résolution du système des équations normales implique l'inversion d'une matrice, ce qui est coûteux en termes de calcul, notamment pour les grands ensembles de données.

Mesures de performance : il existe plusieurs

- Erreur quadratique moyenne (MSE) :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

où n est le nombre total d'observations.

- Racine de l'erreur quadratique moyenne (RMSE) :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Le RMSE est la racine carrée du MSE et a la même unité que la variable cible y.

- Erreur absolue moyenne (MAE) :

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

MAE mesure la moyenne des écarts absolus entre les valeurs prédites et les valeurs réelles.

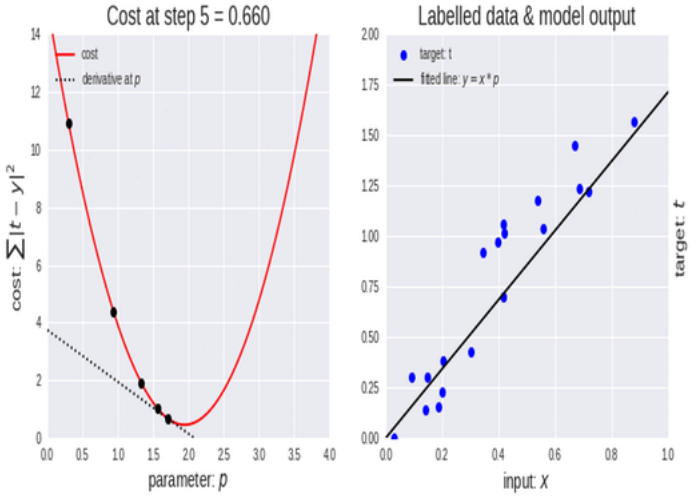
Principe de la descente de gradient

- La descente de gradient est une méthode d'optimisation itérative pour minimiser une fonction coût.
- L'objectif est de mettre à jour les paramètres β_0 et β_1 de manière itérative :

$$\beta_j := \beta_j - a\frac{\partial J}{\partial \beta_j} \tag{2}$$

- a est le taux d'apprentissage, J est la fonction coût.

Algorithme de la descente de gradient



Algorithme de la descente de gradient

1. Initialiser β_0 et β_1 avec des valeurs aléatoires.
2. Répéter jusqu'à convergence :
   - Calculer le gradient ∇J.
   - Mettre à jour les paramètres $\beta_j := \beta_j - a\nabla J$.

Descente de Gradient et RMSE

- La descente de gradient est une méthode itérative pour optimiser la fonction de coût.
- La fonction de coût utilisée pour la régression linéaire est l'erreur quadratique moyenne (MSE) :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

- La racine carrée de cette valeur est le RMSE (Root Mean Squared Error) :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4}$$

- L'objectif de la descente de gradient est de minimiser le RMSE.

Méthodes de validation

- Diviser les données en ensembles d'entraînement et de test.
- Calculer le coefficient de détermination R^2 sur les données de test.
- Analyser les résidus pour évaluer la qualité de la régression.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

où y_i sont les valeurs réelles, $\hat{y}_i$ les valeurs prédites, et $\bar{y}$ la moyenne des y.

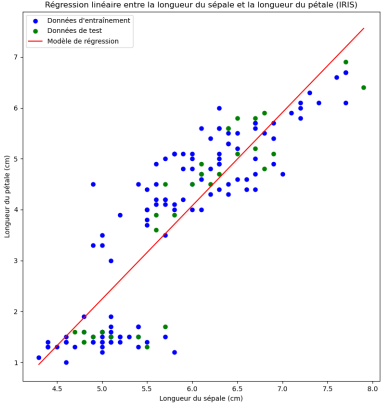
Exemple

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.datasets import load_iris
# Chargement du dataset IRIS
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
# Selection des variables (longueur du sepale et longueur du petale)
X = df[['sepal length (cm)']]
y = df['petal length (cm)']
# Separation des donnees en ensemble d'entrainement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Creation et entrainement du modele de regression lineaire
model = LinearRegression()
model.fit(X_train, y_train)
# Prediction sur les donnees de test
y_pred = model.predict(X_test)
```

Exemple

```
# Calcul des metriques de performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Erreur quadratique moyenne (MSE) : {mse:.2f}')
print(f'Coefficient de determination (R²) : {r2:.2f}')
# Validation croisee
scores = cross_val_score(model, X, y, cv=5, scoring='r2')
print(f'Scores R² de validation croisee : {scores}')
print(f'Moyenne R² : {scores.mean():.2f}')
# Visualisation des resultats
plt.scatter(X_train, y_train, color='blue', label='Donnees d\'entrainement')
plt.scatter(X_test, y_test, color='green', label='Donnees de test')
# Affichage de la droite de regression lisse
X_range = np.linspace(X.min(), X.max(), 100).reshape(-1, 1)
plt.plot(X_range, model.predict(X_range), color='red', label='Modele de regression')
plt.xlabel('Longueur du sepale (cm)')
plt.ylabel('Longueur du petale (cm)')
plt.title('Regression lineaire entre la longueur
          du sepale et la longueur du petale (IRIS)')
plt.legend()
plt.show()
```

Exemple




```
Erreur quadratique moyenne (MSE) : 0.60
Coefficient de determination (R²) : 0.82
Scores R² de validation croisee :
[-51.52454692   0.52866261  -1.85363774
 -0.10127352  -1.87189524]
Moyenne R² : -10.96
```

Exemple avec uniquement "Iris Virginica"

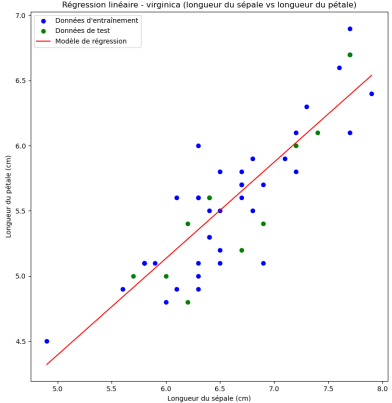
```
...
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['target'] = iris.target  # Ajout de la colonne cible (0, 1 ou 2)

# Filtrer le DataFrame pour ne conserver qu'une seule
# classe (exemple : classe 0 - Setosa)

classe_cible = 2  # Remplacez par 1 (Versicolor)
                  # ou 2 (Virginica) pour changer de classe
df_classe = df[df['target'] == classe_cible]

# Selection des variables (longueur du sepale et longueur du petale)
# pour la classe filtree
X = df_classe[['sepal length (cm)']]
y = df_classe['petal length (cm)']
...
```

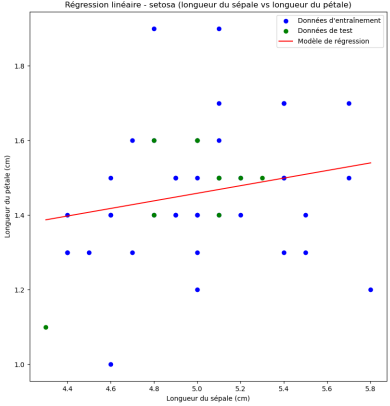
Exemple avec la classe virginica




```
Classe selectionnee : virginica
Erreur quadratique moyenne (MSE) : 0.08
Coefficient de determination (R²) : 0.76
Scores R² de validation croisee : [ 0.64361626
  0.8314034   0.78018601  0.61811866 -0.21852765]
Moyenne R² : 0.53
```

LISV Laboratoire d'Ingénierie des systèmes de Versailles
UVSQ université PARIS-SACLAY

Exemple avec la classe versicolor




```
Classe selectionnee : versicolor
Erreur quadratique moyenne (MSE) : 0.10
Coefficient de determination (R²) : 0.59
Scores R² de validation croisee : [0.79778748
0.4277642  0.30493865 0.28112056 0.54946235]
```

Apprentissage supervisé

Apprentissage supervisé : Régression linéaire multiple

Objectifs du chapitre :

1. Régression linéaire multiple