Bases du data scientist

Data science
Master 2 ISIDIS

Sébastien Verel
verel@lisic.univ-littoral.fr
http://www-lisic.univ-littoral.fr/~verel

Université du Littoral Côte d'Opale
Laboratoire LISIC
Equipe OSMOSE

Bibliographie

Ce cours repose principalement sur ce livre :

Data Science : fondamentaux et études de cas
Machine Learning avec Python et R
Eric Biernat, Michel Lutz,
Eyrolles, 2015.

Que les auteurs en soient remerciés chaleureusement !

Une définition

Data science

"Démarche empirique qui se base sur des données pour apporter une réponse à des problèmes"

Data science : fondamentaux et études de cas, E. Biernat, M. Lutz, Eyrolles, 2015.

Une définition

Data science

"Démarche empirique qui se base sur des données pour apporter une réponse à des problèmes"

Data science : fondamentaux et études de cas, E. Biernat, M. Lutz, Eyrolles, 2015.

Remarque

- Il faut des données !
  Accessibilité juridique, technique, matérielle

Données

Définition

"Le résultat d'une observation faite
sur une population ou sur un échantillon"

Statistique, dictionnaire encyclopédique, Springer (Dodge, 2007)

Une donnée est un **nombre**, ou une **caractéristique** qui apporte une *information* sur un individus, un objet ou une observation

Exemple

Florian : "J'ai 10 ans"

Variable

Lien entre une variable et des données :

Le nombre/caractéristique varie avec les individus/objets

Notations :

- Variable X_j
- pour les individus/objets/observations i : X_{ij}.

Variable X_{age} pour les individus $1, 2, \ldots$: $X_{1age}, X_{2age}, \ldots$

Type de données

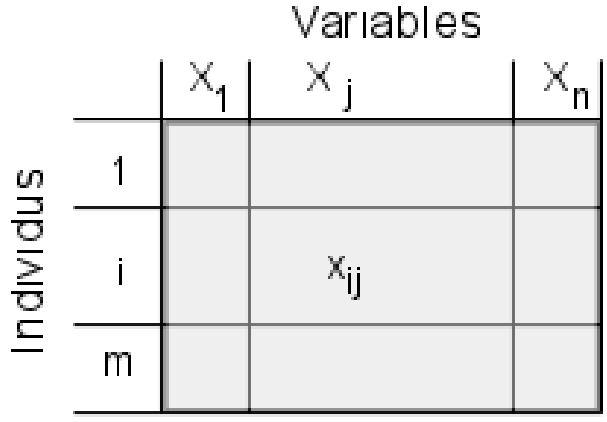
- Donnée **quantitative**
  quantité mesurable, répond au "combien ?"
  calculs possibles (moyenne, etc.),
  comparaisons (égalité, différence, inf/supérieure)
  - Continues : $\in \mathbb{R}$
  - Discrètes : nombre de valeurs "limitées"
- Données **Qualitative**
  qualité ou caractéristiques
  répond à la "catégorie"
  - Nominale (catégorielle)
    couleur des yeux
    comparaison (égalité / différence)
  - Ordinale
    Possède un ordre (degré à un test d'opinion etc.)
    comparaison supérieure / inférieure possible

Représentation matricielle des données

Plusieurs variables $X_1, X_2, \ldots, X_j$ pour j de 1 à n peuvent décrire un même individus/objet/observation.
Grand nombre d'individus i de 1 à m.

La valeur de la variable j sur un individus i se note x_{ij}

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}$$



Apprentissage automatique (Machine Learning)

Définition informelle

Etude et conception de systèmes (méthodes exécutées par une machine) qui sont capables d'apprendre à partir de données.

Exemple

un système qui distinguent les courriels spam et non-spam.

Apprentissage automatique (Machine Learning)

E : l'ensemble de toutes les tâches possibles.
S : un système (une machine)

Définition un peu plus formelle [T.M. Mitchell, 1997]

$T \subset E$: ensemble de taches appelé *training set*
$P : \mathcal{S} \times E \rightarrow \mathbb{R}$: mesure de performance d'un syst. sur des tâches.

Un système S apprend lors d'une expérience Exp si
la performance de S sur les taches T, mesurée par P, s'améliore.
$$P(S_{\text{avant Exp}}, T) \leq P(S_{\text{après Exp}}, T)$$

Exemple

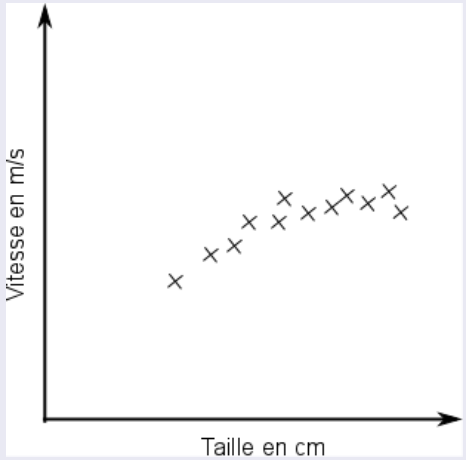
Taches T : Classifier des emails reçus durant une journée
Performance P : Taux de rejet correct des spams par S
Expérience Exp : 1 semaine exposition aux courriels d'un utilisateur

Types d'apprentissage

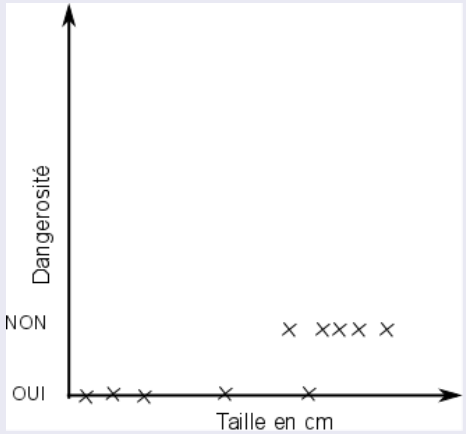
- Apprentissage supervisé :
  Apprentissage sur un ensemble d'exemples étiquetés :
  (x_i, y_i)
- Apprentissage non supervisé :
  Apprentissage sur un ensemble d'exemples non étiquetés (cf. clustering)
  x_i
- Apprentissage semi-supervisé :
  Apprentissage sur un ensemble d'exemples étiquetés / non étiquetés
- Apprentissage par renforcement :
  Apprentissage où les actions sur l'environnement se mesurent par une récompense
- ...

Typologie des problèmes

- Regression : (x_i, y_i) avec $y_i \in \mathbb{R}$



- Classification : (x_i, y_i) avec y_i quelques valeurs possibles



Liste des algorithmes

- Régression linéaire univariée (fait ?)
- Régression linéaire multivariée (fait ?)
- Régression polynomiale
- Régression régularisée
- Naive Bayes
- Régression logistique
- Clustering (fait en partie)
- Arbres de décision (fait)

Régression linéaire univariée

Définition de la fonction hypothèse

$$\text{valeur d'entrée } x \xrightarrow{\text{hypothèse } h} \text{valeur de sortie } y$$

Dans le cas de la régression linéaire univariée :

$$h(X) = \theta_0 + \theta_1 X$$

Trouver le couple (θ_0, θ_1) tel que $h(X)$ soit le plus proche de Y

Erreur d'approximation

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$

Descente de gradient : $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$

Exercice

A partir du jeu de donnée `cars` fournis dans R,

- Réaliser l'analyse de statistique descriptive de la vitesse et la distance d'arrêt (mean, distribution, etc.).
- Calculer le modèle linéaire entre la vitesse et la distance d'arrêt
- Tracer la droite de régression linéaire.

Régression linéaire multivariée

Définition de la fonction hypothèse

$$\text{valeur d'entrée } x \xrightarrow{\text{hypothèse } h} \text{valeur de sortie } y$$

Dans le cas de la régression linéaire multivariée :

$$h(X) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \ldots + \theta_n X_n$$

Erreur d'approximation

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$

Descente de gradient : $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$

Normalisation (scaling) : par centrage réduction, ou entre 0 et 1

Exercice

A partir des jeux de donnée contenu dans `data02.zip`,

- Réaliser l'analyse de statistique descriptive
- Calculer le modèle multilinéaire

Régression polynomiale

Définition de la fonction hypothèse

Dans le cas de la régression linéaire polynomiale :

$$h(X) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \theta_3 X_1^2 + \theta_4 X_2^2$$

Erreur d'approximation

$$J(\Theta) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$

Descente de gradient : $\theta_j := \theta_j - \alpha \frac{\partial J}{\partial \theta_j}(\Theta)$

Exercice

A partir du jeu de donnée `cars` fournis dans R,

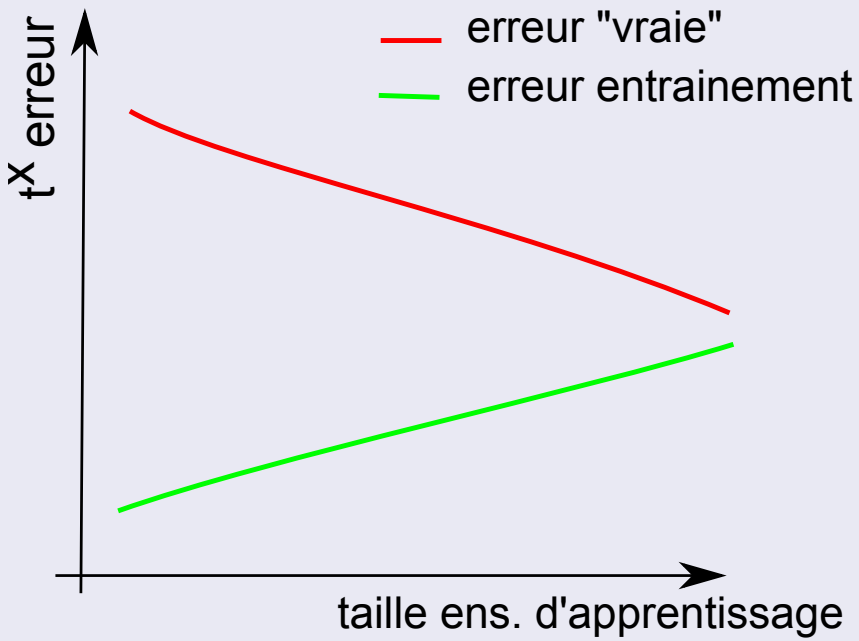
- Calculer une régression polynomiale de degré 2.
- Tracer la droite de régression linéaire.

```
model <- lm(dist ≈ poly(speed, 2, raw = TRUE), data
= cars)
model <- lm(dist ≈ speed + I(speed^2), data = cars)
lines(cars$speed, predict(model, cars))
```

Les erreurs

Relation entre erreurs

- Erreur d'apprentissage : taux d'erreur sur l'ensemble des exemples d'apprentissage
- Erreur "vraie" : erreur sur l'ensemble de tous les exemples possibles



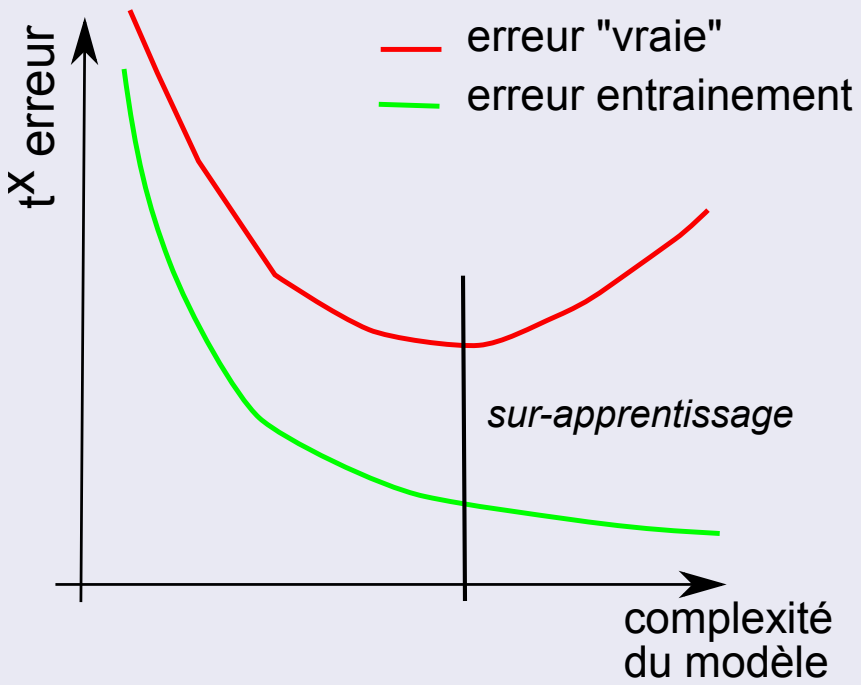
Sur-apprentissage

Exces d'apprentissage

Sur-spécialisation du modèle sur l'ensemble d'entrainement

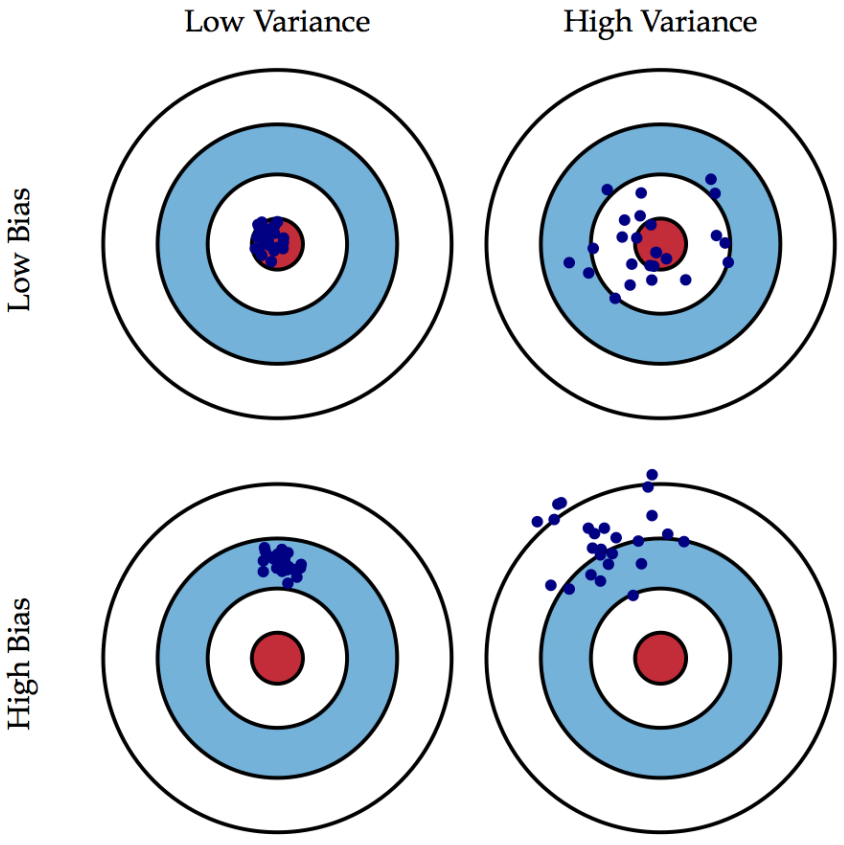
$\Rightarrow$ Perte de capacité de généralisation

$\approx$ Apprentissage "par coeur"



Exemple de mesure de complexité : degré du polynôme

Sur-apprentissage : compromis biais-variance



- Erreur due au **biais** : différence entre la prédiction du modèle et la valeur correcte.
- Erreur due à la **variance** : la variabilité d'une prédiction de modèle pour un point x de donnée

Source Scott Fortmann-Roe : excellent billet
`http://scott.fortmann-roe.com/docs/BiasVariance.html`

Evaluation d'un modèle d'apprentissage

Technique

Partitionner l'ensemble des exemples en :

- un ensemble d'apprentissage ($\approx 70\%$)
- un ensemble *indépendant* de test ($\approx 30\%$)

Le taux d'erreur est estimé (sans biais) sur l'ensemble de test.

Inconvénient

- Requiert un nombre important d'exemples
- Dilemme :
  - Plus on met d'exemples dans le test, plus l'estimation est précise
  - Plus on met d'exemples dans l'apprentissage, meilleur est le modèle (a priori)

Méthode de ré-échantillonnage

Permet d'estimer l'erreur de généralisation.

K-folds cross-validation

Partitionner aléatoirement l'échantillon en K blocs
Pour chaque bloc k,
    Construire le modéle sur les $k-1$ autres blocs
    Calculer l'erreur en test e_k sur le block k
Calculer l'erreur moyenne des erreurs e_k

Autres techniques :

- Leave-one-out ($K = n$)
- Bootstrap, bagging, etc.

Exercice : sur-apprentissage

```
x <- 1:10
y <- x + c(-0.5, 0.5)
plot(x, y)
```

Calculer trois régressions avec des polynômes de degré 1, 3 et 9.
Tracer les régressions obtenues en utilisant en abscisse le vecteur

```
z <- seq(1, 10, length.out = 250)
```

Recommencer les régression avec des données légèrement modifiées :

```
x <- c(1:10, 10:15)
y <- x + c(-0.5, 0.5)
```

Conclure.

Sélection de modèles

Comment sélectionner un modèle parmi un ensemble possible de modèles ?

- Sub-diviser l'ensemble d'apprentissage en ensemble d'apprentissage et de validation
- Entrainer les différents modèles sur le nouvel ensemble d'apprentissage
- Utiliser une mesure de qualité du modèle sur l'ensemble de validation pour sélectionner le modèle.

Régression régularisée

Fonction objectif pour trouver les paramètres Θ du modèle

$$J(\Theta) = L(\Theta) + \Omega(\Theta)$$

- $L(\Theta)$: **erreur d'entrainement**, mesure comment le modèle est proche des données
- $\Omega(\Theta)$: **régularisation**, mesure la complexité du modèle

But

Contrôler la complexité du modèle pour réduire la variance des prédictions, et rendre les prédictions plus stable.

Régression régularisée

Fonction objectif pour trouver les paramètres Θ du modèle

$$J(\Theta) = L(\Theta) + \Omega(\Theta)$$

- $L(\Theta)$: **erreur d'entrainement**, mesure comment le modèle est proche des données
- $\Omega(\Theta)$: **régularisation**, mesure la complexité du modèle

Erreur d'entrainement

Erreur sur les données d'entrainement : $L = \sum_{i=1}^{m} \ell(y_i, h(x_i))$

- Erreur quadratique : $\ell(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$
- Erreur logistique :
  $\ell(y_i, \hat{y}_i) = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})$

Régression régularisée

Fonction objectif pour trouver les paramètres Θ du modèle

$$J(\Theta) = L(\Theta) + \Omega(\Theta)$$

- $L(\Theta)$: **erreur d'entrainement**, mesure comment le modèle est proche des données
- $\Omega(\Theta)$: **régularisation**, mesure la complexité du modèle

Régularisation

Complexité du modèle ?

- Norme L_2 : $\Omega(\omega) = \lambda \|\omega\|^2 = \lambda \sum_{i=1}^{n} \omega_i^2$
- Norme L_1 : $\Omega(\omega) = \lambda \|\omega\|_1 = \lambda \sum_{i=1}^{n} |\omega_i|$

Régression régularisée : cas linéaire

Définition de la fonction hypothèse

Dans le cas de la régression linéaire régularisée :

$$h(X) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \ldots + \theta_n X_n$$

Erreur d'approximation

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2 + \Omega(\Theta)$$

Et toujours une descente de gradient pour minimiser J :
$\theta_j := \theta_j - \alpha \frac{\partial J}{\partial \theta_j}(\Theta)$

Régression régularisée

Régression ridge

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2 + \lambda \sum_{i=1} \theta_j^2$$

Régression Lasso

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2 + \lambda \sum_{i=1} |\theta_j|$$

Régression ElasticNet (= Ridge + Lasso)

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2 + \lambda \sum_{i=1} ((1-\alpha)\theta_j^2 + \alpha|\theta_j|)$$

Classification Naive Bayes

Théorème de Baye

$$Pr(A|B) = \frac{Pr(B|A)Pr(A)}{Pr(B)}$$

Interprétation :

$$posterieure = \frac{vraisemblance \quad anterieure}{evidence}$$

Remarque : indépendance

A et B indépendants ssi

$$Pr(A|B) = Pr(A)$$

A et B indépendants ssi

$$Pr(A, B) = Pr(A)Pr(B)$$

Un (fameux) exemple

Outlook	Temperature	Humidity	Wind	Playball
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

4 variables nominales :

Outlook $\in$ { Sunny , Overcast, Rain } ; Temperature $\in$ { Hot, Mild, Cool } ; Humidity $\in$ { High , Normal } ; Wind $\in$ { Strong , Weak }

1 cible : Playball $\in$ { No, Yes }

14 exemples étiquetés

Exercice

- Calculer les règles bayesiennes pour chacun des attributs pris séparément
- Calculer le modèle de classification bayesien naif qui utilise l'ensemble des attributs (supposés indépendants).

Naive Bayes

Cadre général (décision binaire)

$$A = \prod_{i=1}^{n} \frac{Pr(x_i|d=1)}{Pr(x_i|d=0)} \times \frac{Pr(d=1)}{Pr(d=0)}$$

La décision d prise :

$$d = \begin{cases} 1 & \text{si } A > \alpha \\ 0 & \text{si } A \leq \alpha \end{cases}$$

En général, $\alpha = 1$

Remarque

Calculer $\log(A)$...

Régression logistique

Classifieur linéaire

Fonction hypothèse

$$lr(x) = \begin{cases} 0 & \text{si } h(x) < 0.5 \\ 1 & \text{si } h(x) \geq 0.5 \end{cases}$$

Et interprétation de h comme probabilité :
$0 \leq h(x) \leq 1$ et $h(x) = Pr(y = 1|x, \theta)$.

Fonctions sigmoides

- Courbe de Gompertz
- Tangente hyperbolique
- Fonction de répartition normale centrée réduite
- Fonction logistique

Régression logistique

Cas binaire

Fonction hypothèse

$$h(x_i, \Theta) = g(\theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} \ \ldots + \theta_n x_{in})$$

avec g fonction logistique

Fonction de coût

$$j(h(x), y) == \begin{cases} -\log(h(x)) & \text{si } y = 1 \\ -\log(1 - h(x)) & \text{si } y = 0 \end{cases}$$

$$j(h(x), y) = -y \log(h(x)) - (1 - y) \log(1 - h(x))$$

En sommant :

$$J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} j(h(x_i), y_i)$$

Rem. : en ML, classifier linéaire par rapport aux variables du problème

Régression logistique

Cas multiclasse

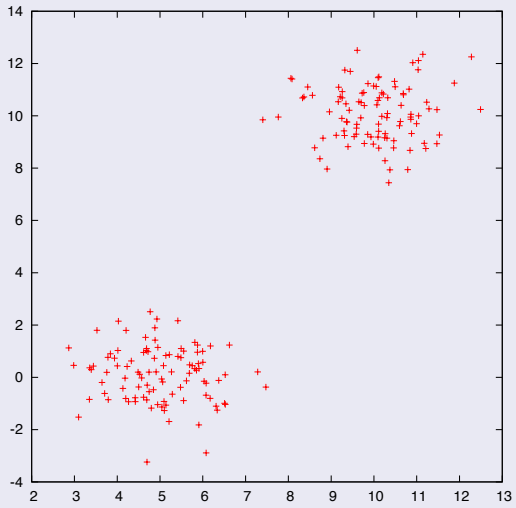
- Pour chaque classe c, on construit un prédicteur $h^{(c)}$ qui donne la probabilité qu'un exemple soit dans la classe c
- La classe x_i est la classe c de plus grande probabilité :

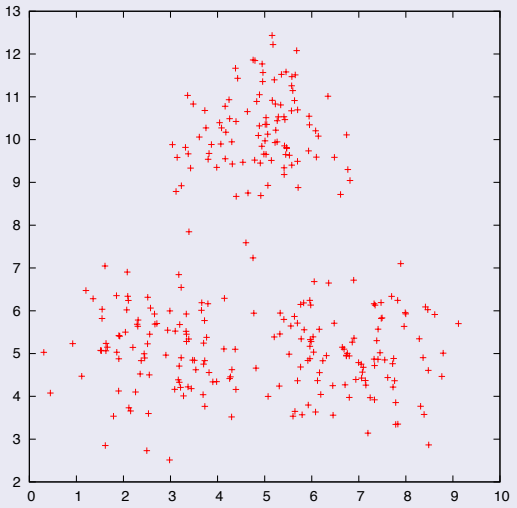
$$c = \text{argmax}_{c \in C} \; h^{(c)}(x_i)$$

Liste des algorithmes

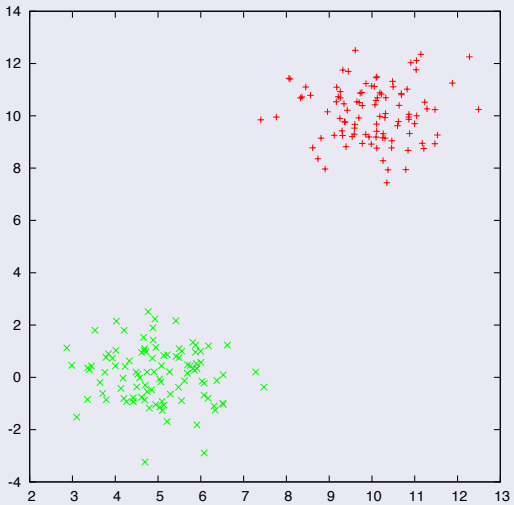
- Régression linéaire univariée
- Régression linéaire multivariée
- Régression polynomiale
- Régression régularisée
- Naive Bayes
- Régression logistique
- Clustering
- Arbres de décision

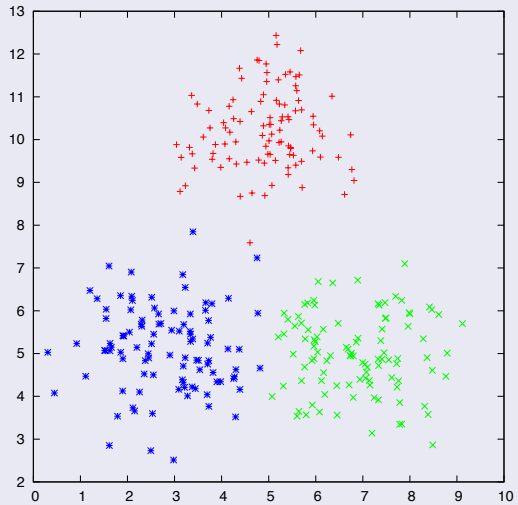
Clustering : Exemples graphiques





Clustering : Exemples graphiques





Clustering : Position du problème

Partitionnement

entrée :

Ensemble de n points / exemples / observations

$$E = \{e_1, e_2, \ldots, e_n\}$$

sortie :

Partition de E

$$P = \{P_1, P_2, \ldots, P_k\}$$

équivalent à une fonction $c : E \to \{1, \ldots, k\}$

Clustering : Position du problème

Partitionnement

entrée :

Ensemble de n points / exemples / observations

$$E = \{e_1, e_2, \ldots, e_n\}$$

sortie :

Partition de E

$$P = \{P_1, P_2, \ldots, P_k\}$$

équivalent à une fonction $c : E \to \{1, \ldots, k\}$

Combien de partitions avec k clusters ?

Clustering : Position du problème

Partitionnement

entrée :

Ensemble de n points / exemples / observations

$$E = \{e_1, e_2, \ldots, e_n\}$$

sortie :

Partition de E

$$P = \{P_1, P_2, \ldots, P_k\}$$

équivalent à une fonction $c : E \to \{1, \ldots, k\}$

Combien de partitions avec k clusters ?

$$k^n/k!$$

Beaucoup même pour $n = 100$ *et* $k = 2$, *comment choisir ?...*

Problème d'optimisation associé

Problème d'optimisation

Critère de la qualité d'une partition :

$$U : \mathcal{P}(E) \to \mathbb{R}$$

Trouver une bonne partition revient à maximiser le critère :

$$\text{argmax}_{P \in \mathcal{P}(E)} U(P)$$

Utilisation de méthodes d'optimisation (locale, greedy, etc.)...

Critère de qualité

Forme

En général, le critère est de la forme :

$$U(P) = \sum_{i \in 1}^{k} w(P_i)$$

avec w une mesure de la qualité d'un cluster.

Exemples

Somme des carrés des distances entre les points du cluster :

$$w(P_i) = \sum_{x \in P_i} \sum_{y \in P_i} d^2(x, y)$$

Probabilité d'observation des points du cluster :

$$w(P_i) = \prod_{x \in P_i} Pr(x|\theta_i)$$

Algorithmes de clustering

Différentes approches :

- Partitionnement hiérarchique :
  Regroupement (ou séparation) de clusters selon un critère
  Dendrogramme
- Partitionnement controïde :
  Utilisation de centres pour paramètrer les clusters
  k-means (cf. plus loin)
- Partitionnement fondé sur des distributions de probabilité
  Un cluster est représenté par une distribution de probabilité dont il faut trouver les paramètres
  Algorithme E-M, Gaussian mixture models
- Partitionnement fondé sur la densité :
  Selon la densité locale de points, croissance du cluster
  DBSCAN

k-means : Pseudo-code

k-means

Choisir (aléatoirement) k centres $\mu_1^{(1)}, \ldots, \mu_k^{(1)}$

repeat

Affecter chaque observations au centre le plus proche :

$$P_i^{(t)} = \{e_j \ : \ d(e_j, \mu_i^{(t)}) \leq d(e_j, \mu_a^{(t)}) \ \forall a = 1..k\}$$

Mettre à jour les centres (moyenne des clusters) :

$$\mu_i^{(t+1)} = \frac{1}{\sharp P_i^{(t)}} \sum_{e_j \in P_i^{(t)}} e_j$$

until plus de changement (convergence)

Le k-means est un algorithme local tel que $U(P^{(t+1)}) < U(P^{(t+1)})$, d'où optima local...

Avantages / Inconvénients

Avantages

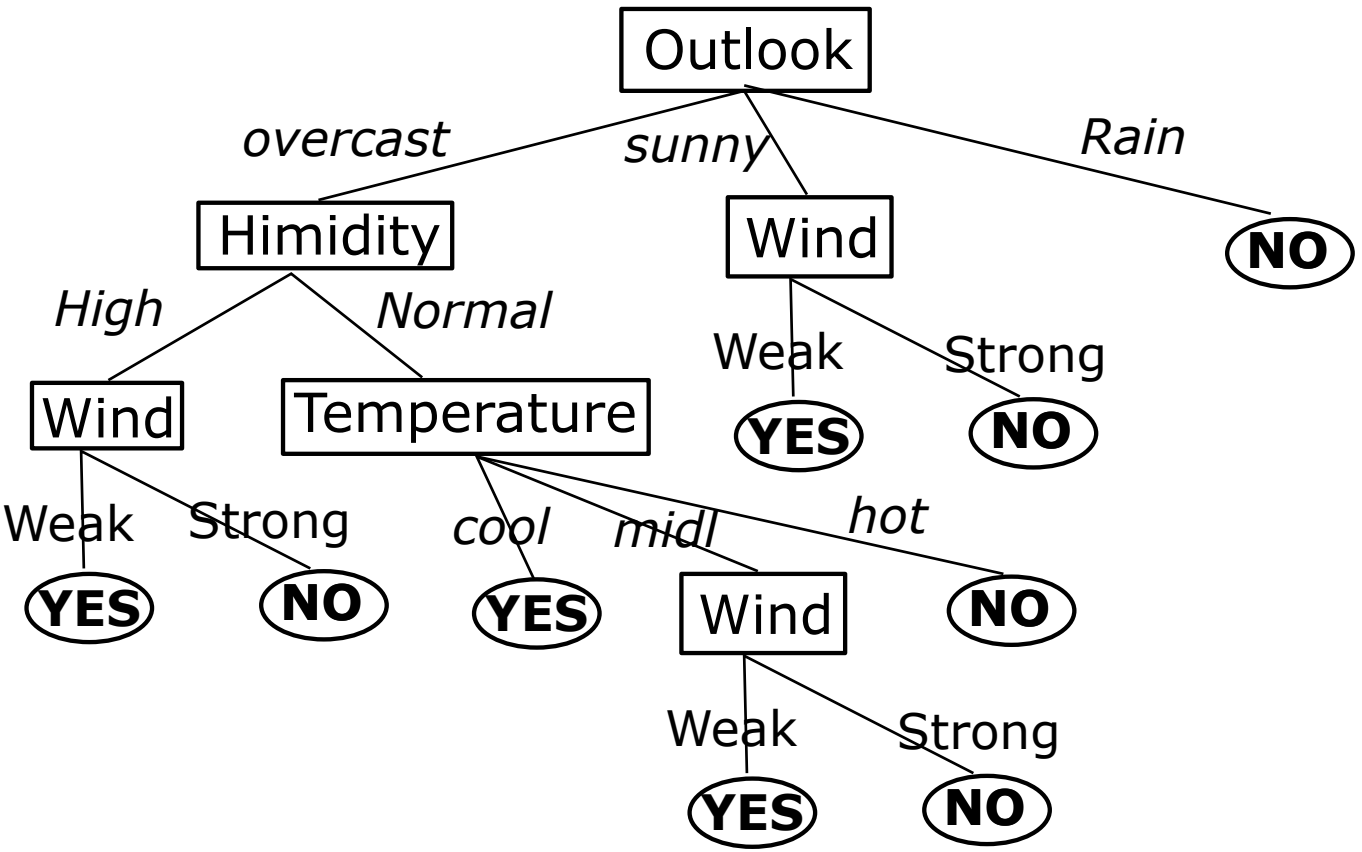
- Facile à interpréter
- Facile à coder
- Complexité polynomiale

Inconvénients

- Fixer le nombre de clusters k (d'où utilisation possible en plus de clustering hiérarchique)
- La forme des clusters est supposée "sphérique" et les clusters séparables

Décision tree

Classification à l'aide d'un arbre



Remarque : un arbre code en fait un ensemble de règles (conjonctions, disjonctions)

Si Outlook = "overcast" et Humidity =... alors playball = Yes

Algorithme d'apprentissage

Apprentissage par arbre de décision

Construction un arbre :

- Noeuds internes : sélectionner d'un attribut comme étiquette, les arcs sont étiquetés par les valeurs de l'attribut
- Feuilles : couper l'arbre avec une valeur de l'attribut cible

On veut en général :

- Un taux d'erreur faible
- Une bonne généralisation
- Un arbre de petite taille compréhensible pour un non expert
- etc.

Nombreux algos : ID3, C4.5, CART, CHAID, algo. évo., etc.

Une classe d'algorithmes d'apprentissage

Algorithme top-down

Procédure : `construireArbre`(X)
if tous les exemple de X appartiennent à la même classe **then**
  Créer une feuille portant le nom de cette classe
else
  Choisir un attribut pour créer un noeud
  Le test associé à ce noeud sépare X en X_d et X_g
  `construireArbre`(X_g)
  `construireArbre`(X_d)
end if

Une classe d'algorithmes d'apprentissage

Algorithmes top-down greedy

Pour chaque noeud interne,

- le "meilleur" attribut est sélectionné selon l'ensemble d'apprentissage
- l'ensemble d'apprentissage est partitionné selon les valeurs possibles de l'attribut du noeud

Le processus est répété en chaque noeud et s'arrête lorsque :

- tous les exemples ont la même valeur d'attribut cible
- un nouveau partitionnement n'augmente pas la qualité de la prédiction

- Top-down : construction à partir de la racine
- Greedy : meilleur choix local, pas de remise en cause
  Les optima locaux guettent ! Optimalité locale *vs.* globale

Critique

Avantages

- Simple à comprendre et à interpréter
- Le modèle est "white-box" (rés. neurones est black-box)
- Peu de préparation des données : pas de normalisation, etc.
- Données numériques et catégorielles possibles
- Robuste aux données aberrantes (outliers)

Inconvénients

- Apprendre un arbre de décision optimal : NP-complet
- Heuristique d'apprentissage greedy : arbre sous optimal
- Création d'arbres trop complexes, sur-spécialisé
- Biais vers certaines formes : attribut avec plus de valeurs, petit arbre, etc.
- Détection difficile des interactions entre attributs
- Certains problèmes sont difficiles à apprendre sous forme d'arbre (xor, parité, multiplexer)

ID3 (Iterative Dichotomiser 3)

Ross Quinlan, 1986

Algorithme top-down greedy
basé sur le gain d'information (information gain)

Principe

1. Calculer l'entropie de tous les attributs en utilisant l'ensemble d'apprentissage S
2. Partitionner l'ensemble S en utilisant l'attribut pour lequel l'entropie est minimum (gain d'information maximum)
3. Construire le noeud de l'arbre avec cet attribut
4. Recommencer récursivement sur chaque sous arbre avec chaque sous-ensemble

Mesure d'entropie

Entropie H

Mesure de la quantité d'incertitude dans un ensemble (dispersion)

$$H(S) = -\sum_{x \in X} p(x) \log_2 p(x)$$

- S : ensemble des données
- D_S : ensemble des classes de S
- $p(s)$: proportion de la classe $s \in D_S$ dans S

Lorsque $H(S) = 0$, S est parfaitement classé.

Gain d'information

Information mutuelle (entropie croisée)

Mesure de l'information conjointe de 2 variables aléatoires (information d'une variable par rapport à l'autre)

$$I(S,T) = -\sum_{s \in D_S, t \in D_T} p(s,t) \log_2 \frac{p(s,t)}{p(s)p(t)}$$

Information gain (- information mutuelle)

Mesure de la différence d'entropie entre avant et après le partitionnement selon un attribut

$$IG(S,T) = -I(S,T) = H(S) - \sum_t p(S_t) H(S_t)$$

- $T = \{S_1, \ldots\}$ partitionnement de $S = \cup_t S_t$
- $p(S_t) = \sharp S_t / \sharp S$
- $H(S)$, $H(S_t)$: entropies de S et de S_t

Pseudo code

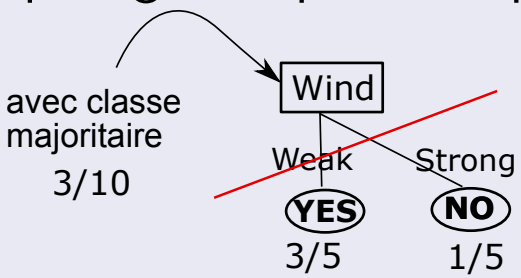
ID3(exemples, cible, attributs) :
- **si** tous les exemples sont positifs (resp. négatifs) **alors**
  - **retourner** une feuille avec l'étiquette positif (resp. négatif)
- **si** attributs est vide **alors**
  - **retourner** une feuille avec l'étiquette la plus fréquente
- **sinon**
  - $A \leftarrow$ attribut de plus grand gain d'information
  - **construire** un noeud avec l'étiquette A
  - **pour** chaque valeurs v_i de A
    - **ajouter** la branche v_i au noeud
    - **si** exemples($A = v_i$) est vide **alors**
      - **ajouter** à la branche la feuille avec l'étiquette la plus fréquente
    - **sinon**
      - **ajouter** à la branche le sous-arbre ID3(exemples($A = v_i$), cible, attributs $-A$)

C4.5

Ross Quinlan, 1993

Amélioration de ID3

- Utilisation du ratio de gain d'information au lieu de IG :
  $IG(S, T)$ biaisé vers attributs ayant un grand nombre de valeurs
  $ratioIG(S, T) = IG(S, T)/H(T)$
- Possibilité de valeur "null" :
  Exemple ignoré lors dans le calcul du noeud
- Prise en compte des attributs à valeur continue :
  Discrétisation par $P(A < a_i)$
  pour toutes les valeurs possibles de A, calcul de IG
- Elagage (pruning) pour réduire la taille de l'arbre :
  Technique bottom-up : branches finales élaguées
  lorsque taux d'erreur plus grand qu'en remplaçant par une feuille



Liste des algorithmes

- Régression linéaire univariée
- Régression linéaire multivariée
- Régression polynomiale
- Régression régularisée
- Naive Bayes
- Régression logistique
- Clustering
- Arbres de décision