Apprentissage supervisé

Master parcours SSD - UE Apprentissage Statistique II

Pierre Mahé - bioMérieux & Université de Grenoble-Alpes

Apprentissage statistique ?

Apprentissage statistique = apprentissage automatique

- (statistical learning, machine learning)

Wikipedia : Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed. [...] Machine learning explores the study and construction of algorithms that can learn from and make predictions on data – such algorithms overcome following strictly static program instructions by making data driven predictions or decisions, through building a model from sample inputs.

$\Rightarrow$ Apprentissage à partir d'exemples

- par opposition aux systèmes experts.

Apprentissage statistique

Principe = trouver des régularités dans les données

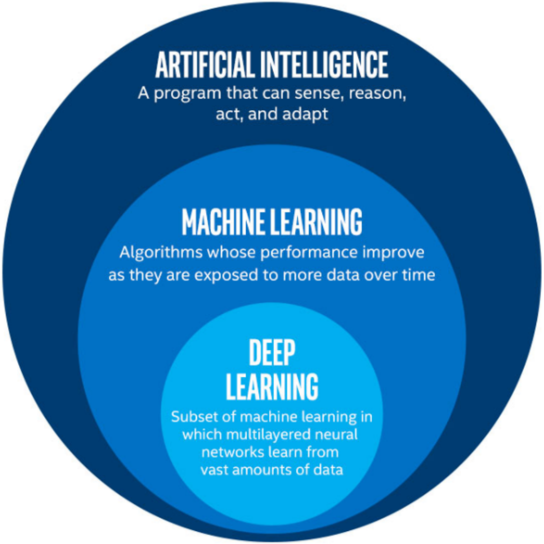
Pourquoi faire ?

- découvrir des structures "cachées" dans les observations
  - apprentissage **non-supervisé**
- **prédire de nouvelles observations**
  - apprentissage **supervisé**

A l'interface de nombreux domaines :

- statistiques
- informatique ("computer science")
- intelligence artificielle
- mathématiques (e.g., optimisation numérique)
- théorie de la décision
- ...

Machine Learning / Intelligence Artificielle / Deep Learning ?



Motivation

Il est très dur de définir ce qui "fait" un 2 :

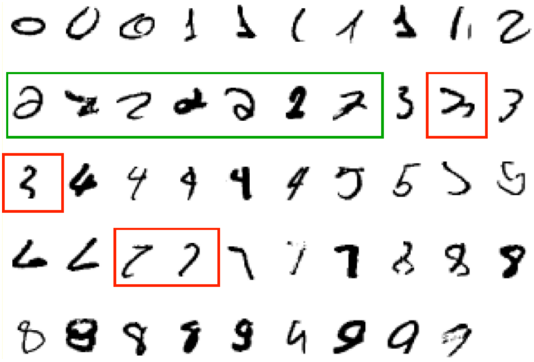
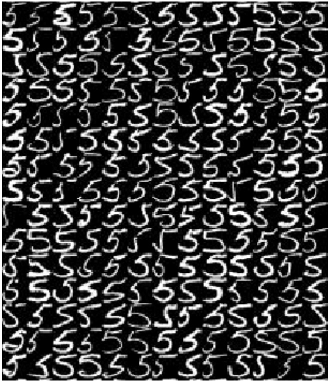


Figure: Exemple tiré d'un cours de G. Hinton

...mais on sait très bien apprendre à les reconnaître.

L'approche Machine Learning [1]



1. USPS dataset : `http://www.cs.nyu.edu/~roweis/data.html`

Intérêt

L'apprentissage statistique prend tout son sens quand :

- l'expertise humaine est absente
  - e.g., analyse de l'ADN
- il est très difficile de l'expliciter
  - e.g., reconnaissance de caractères
- les quantités de données à traiter sont trop importantes
  - e.g., applications web / réseaux sociaux
- les données évoluent dynamiquement
  - e.g., prédire le cours d'actions financières
- ...

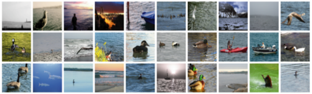
Enormément d'applications dans de nombreux domaines !

Analyse d'image

- Catégorisation

(a) Positive examples of 'whale'

(b) Negative examples of 'whale' obtained by random sampling

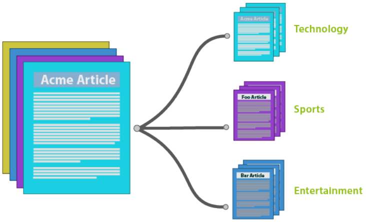


(c) Relevant negatives of 'whale' (this paper)

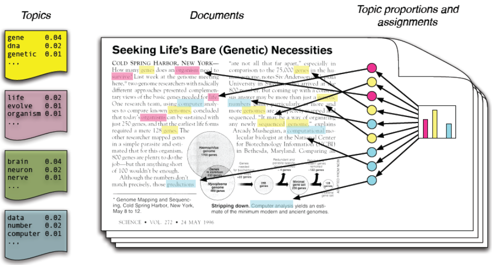
- Interprétation de scènes

Analyse de texte

- Catégorisation automatique



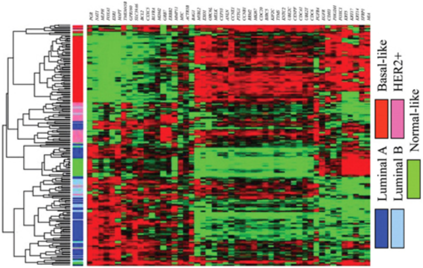
- Détection de thèmes (topics) "cachés"



Biologie / santé

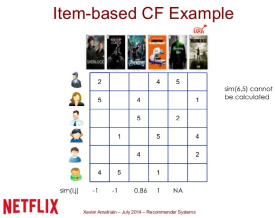
- Prédire la fonction d'une protéine à partir de sa structure

- Diagnostic/prognostic à partir de puces à ADN

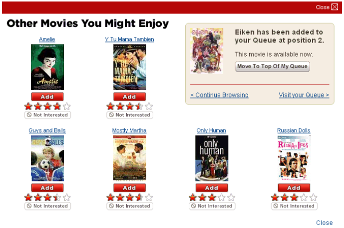


Web / Internet

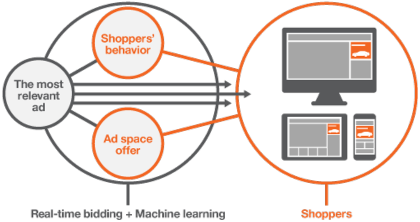
- Recommandation



	2			4	5	
	5		4			1
			5		2	
		1		5		4
			4			2
	4	5		1		
sim(i,j)	-1	-1	0.86	1	NA	

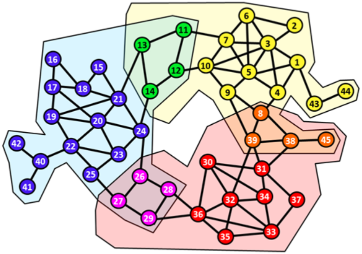


- Publicité ciblée



Réseaux sociaux

- Détection de communautés



Et plein d'autres....

- Audio : reconnaissance de la parole, séparation de sources
- Vidéo : suivi d'objets, surveillance
- Finance, économie
- Sciences de la Vie : climatologie, planétologie
- Génétique
- ...

Défis

- Apprendre des comportements qui soient valables pour d'autres observations
  - notion de **généralisation**
- Faire face à des types de données variés
  - vecteurs, matrices, courbes, séquences, arbres, graphes
- Faire face à des volumes de données conséquents
  - apprentissage : large-scale learning & haute dimension
  - prédiction : problématiques temps-réel

Les deux cadres d'apprentissage principaux[2]

Apprentissage supervisé :

- données : observations (X, Y)
  - descripteurs / variables explicatives + variable d'intérêt
- objectif(s) : prédiction
  - (+ compréhension du lien entre X et Y)

2. mais aussi apprentissage par renforcement, semi-supervisé, transductif, actif, online, ...

Les deux cadres d'apprentissage principaux [2]

Apprentissage supervisé :

- données : observations (X, Y)
  - descripteurs / variables explicatives + variable d'intérêt
- objectif(s) : prédiction
  - (+ compréhension du lien entre X et Y)

Apprentissage non-supervisé :

- données : observations X
  - pas de variable à expliquer
- objectif : identifier des "structures" dans les données
  - moins clairement formalisé que le supervisé

$\Rightarrow$ ce cours = **apprentissage supervisé**

2. mais aussi apprentissage par renforcement, semi-supervisé, transductif, actif, online, ...

Apprentissage supervisé - principe

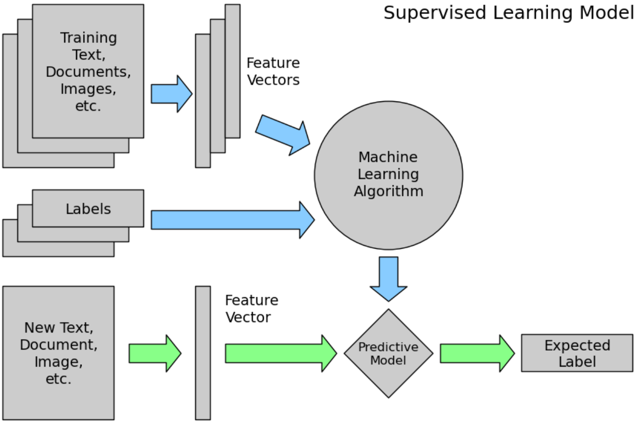


Figure: Image tirée de `http://www.astroml.org/sklearn_tutorial/general_concepts.html`

Apprentissage supervisé - formalisation

On dispose d'un échantillon $\{(x_i, y_i)\}$, $i = 1, ..., n$, :

- des observations $x_i \in \mathcal{X}$,
- des réponses associées $y_i \in \mathcal{Y}$.

$\Rightarrow$ ce sont les données (ou le jeu) d'apprentissage.

Apprentissage supervisé - formalisation

On dispose d'un échantillon $\{(x_i, y_i)\}$, $i = 1, ..., n$, :

- des observations $x_i \in \mathcal{X}$,
- des réponses associées $y_i \in \mathcal{Y}$.

$\Rightarrow$ ce sont les données (ou le jeu) d'apprentissage.

Typiquement :

- $\mathcal{X} = \mathbb{R}^p$: on parle de vecteurs de descripteurs
  - features, attributes, input variables
- Si $\mathcal{Y} = \mathbb{R}$, on parle de régression.
- Si $\mathcal{Y} = \{1, ..., K\}$, on parle de classification
- Si $\mathcal{Y} = \{-1, +1\}$, on parle de classification binaire
  - on note parfois également $\mathcal{Y} = \{0, 1\}$

Apprentissage supervisé - démarche générale

On a un jeu d'apprentissage : $\mathcal{D} = \left\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\right\}_{i=1,\ldots,n}$.

Apprentissage supervisé - démarche générale

On a un jeu d'apprentissage : $\mathcal{D} = \left\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\right\}_{i=1,\ldots,n}$.

$\Rightarrow$ Objectif : apprendre une fonction $f : \mathcal{X} \to \mathcal{Y}$ permettant de prédire la réponse associée à une nouvelle observation.

Apprentissage supervisé - démarche générale

On a un jeu d'apprentissage : $\mathcal{D} = \left\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\right\}_{i=1,\ldots,n}$.

$\Rightarrow$ Objectif : apprendre une fonction $f : \mathcal{X} \to \mathcal{Y}$ permettant de prédire la réponse associée à une nouvelle observation.

Comment faire ?

1. choisir une famille de fonctions candidates
    - e.g., une famille paramétrique de fonctions
2. choisir un membre de la famille grâce à $\mathcal{D}$

Apprentissage supervisé - démarche générale

On a un jeu d'apprentissage : $\mathcal{D} = \left\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\right\}_{i=1,\dots,n}$.

$\Rightarrow$ Objectif : apprendre une fonction $f : \mathcal{X} \to \mathcal{Y}$ permettant de prédire la réponse associée à une nouvelle observation.

Comment faire ?

1. choisir une famille de fonctions candidates
   - e.g., une famille paramétrique de fonctions
2. choisir un membre de la famille grâce à $\mathcal{D}$

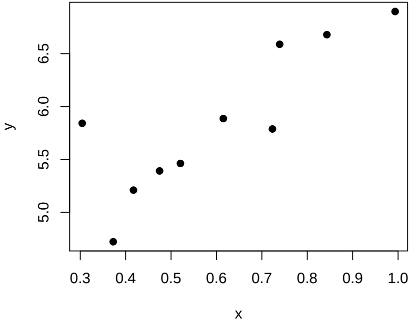
$\Rightarrow$ choisir la famille de fonctions = hypothèse de modélisation

$\Rightarrow$ choisir une fonction = optimiser un critère

Illustration : régressions linéaires & polynomiales

Régression linéaire

Modèle : $\boxed{f(x) = \alpha x + \beta}$



$\Rightarrow$ famille de fonctions : $\left\{(\alpha, \beta) \in \mathbb{R}^2 : f(x) = \alpha x + \beta\right\}$

Régression linéaire

Modèle : $\boxed{f(x) = \alpha x + \beta}$

$\Rightarrow$ comment choisir la "meilleure" fonction (α^*, β^*) ?

Régression linéaire

Modèle : $\boxed{f(x) = \alpha x + \beta}$

$\Rightarrow$ comment choisir la "meilleure" fonction (α^*, β^*) ?

- critère d'erreur des moindres carrés : $\big(y - f(x)\big)^2$
- qu'on évalue sur le jeu d'apprentissage :

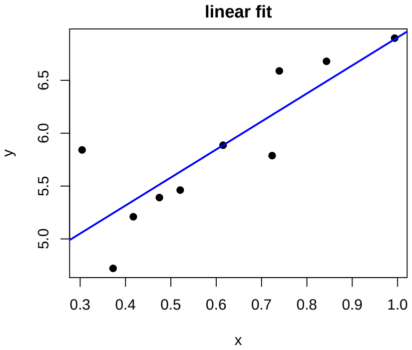
$$J\big((\alpha, \beta)\big) = \sum_{i=1}^{n} \big(f(x_i) - y_i\big)^2$$
$$= \sum_{i=1}^{n} \big(\alpha x_i + \beta - y_i\big)^2$$

- et qu'on minimise : $(\alpha^*, \beta^*) = \arg\min_{\alpha,\beta} J\big((\alpha, \beta)\big)$

$\Rightarrow$ un problème d'optimisation

Régression linéaire

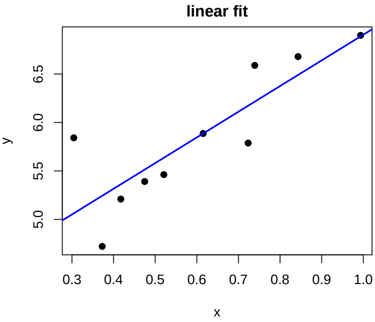
Modèle : $\boxed{f(x) = \alpha x + \beta}$



$\Rightarrow$ solution : $\hat{f}(x) = 2.65x + 4.26$

Régression linéaire

Régression linéaire : modèle parfois trop simple



$\Rightarrow$ pour aller plus loin, deux solutions :

1. considérer des modèles de régression non linéaires
   - e.g., splines, lowess, ...
2. appliquer des transformations aux données d'entrée
   - "basis functions expansion"

Régression polynomiale

Modèle : $\boxed{f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + ... + \beta_d x^d}$

Régression polynomiale

Modèle : $\boxed{f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_d x^d}$

- on a $\mathbf{x} = [x_1, \ldots, x_n]^T$ et $\mathbf{y} = [y_1, \ldots, y_n]^T$
- on définit $\mathbf{b} = [\beta_0, \beta_1, \ldots, \beta_d]^T$
- on construit la matrice Φ :

$$\Phi = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^d \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^d \end{bmatrix}$$

- on minimise la différence entre $\mathbf{y}$ et $\Phi\mathbf{b}$: $||\mathbf{y} - \Phi\mathbf{b}||^2$

Régression polynomiale

Modèle : $\boxed{f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_d x^d}$

- on a $\mathbf{x} = [x_1, \ldots, x_n]^T$ et $\mathbf{y} = [y_1, \ldots, y_n]^T$
- on définit $\mathbf{b} = [\beta_0, \beta_1, \ldots, \beta_d]^T$
- on construit la matrice Φ :

$$\Phi = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^d \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^d \end{bmatrix}$$

- on minimise la différence entre $\mathbf{y}$ et $\Phi\mathbf{b}$: $||\mathbf{y} - \Phi\mathbf{b}||^2$

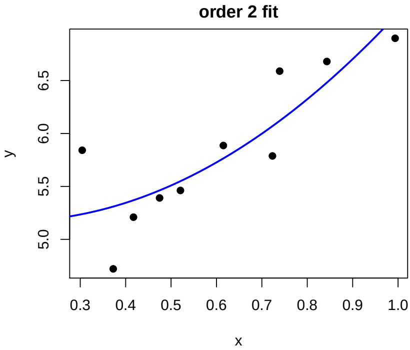
$\Rightarrow$ c'est un problème de régression linéaire multivariée

- modèle linéaire : linéaire selon ses paramètres β_j

$\Rightarrow$ solution $\mathbf{b}^* = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{y}$ (normal equations)

Régression polynomiale

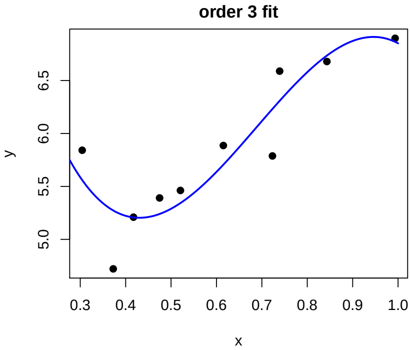
Modèle : $\boxed{f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_d x^d}$



$\Rightarrow$ solution à l'ordre $d = 2$.

Régression polynomiale

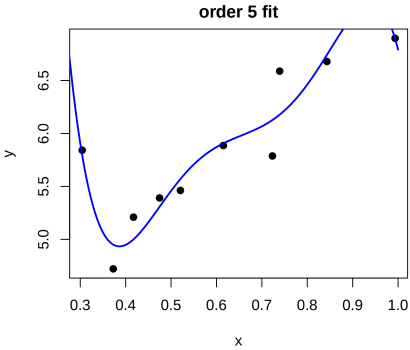
Modèle : $\boxed{f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_d x^d}$



$\Rightarrow$ solution à l'ordre $d = 3$.

Régression polynomiale

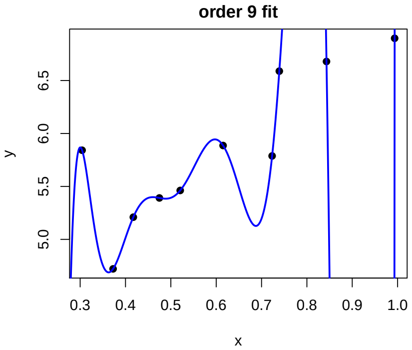
Modèle : $\boxed{f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_d x^d}$



$\Rightarrow$ solution à l'ordre $d = 5$.

Régression polynomiale

Modèle : $\boxed{f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_d x^d}$

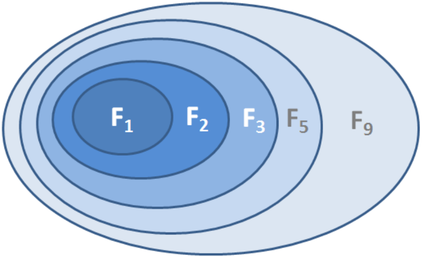


$\Rightarrow$ solution à l'ordre $d = 9$.

Régression polynomiale

Quand d augmente :

- la classe de fonctions est de + en + grande $(F_i \subset F_{i+1})$
- l'erreur d'apprentissage $||\mathbf{y} - \Phi\mathbf{b}||^2$ decroît

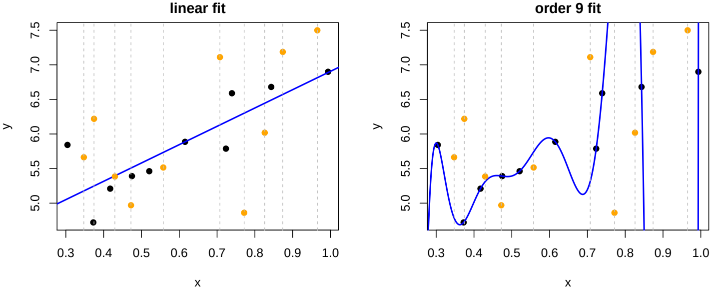


$\Rightarrow$ risque de sur-apprentissage avec les fonctions complexes :

- très (trop) bon "fit" sur les données d'apprentissage
- mauvaise généralisation à de nouvelles données

Régression polynomiale

Sur-apprentissage : illustration



- très (trop) bon "fit" sur les **données d'apprentissage**
- mauvaise généralisation à de nouvelles données

Remarque - "Basis Functions expansions"

Transformations par Basis Functions Expansion :

$$f(x) = \sum_{m=0}^{M} \beta_m h_m(x)$$

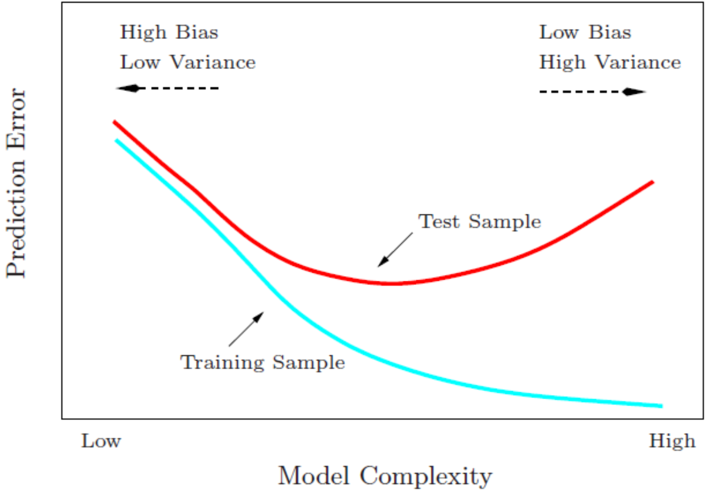
$\Rightarrow$ régression polynomiale : $h_m(x) = x^m$.

Autres exemples :

- fonctions Gaussiennes : $h_m(x) = \exp\left(-\frac{||x-\mu_m||^2}{2\sigma_m^2}\right)$
- autres transformations non-linéaires :
  - $h_m(x) = \log(x)$
  - $h_m(x) = \sqrt{x}$
  - ...
- intéractions $h_m(\mathbf{x}) = \mathbf{x}_i \mathbf{x}_j$, quand $\mathbf{x} \in \mathbb{R}^p, p > 1$.

Compromis biais/variance

Compromis biais/variance [3]



3. Image tirée de Hastie et al. (2001) (Fig.2.11)

Compromis biais/variance

Formalisation :

- On dispose d'un jeu de données $\mathcal{D} = \{(xi, y_i)\}_{i=1,\dots,n}$
- On considère que (x_i, y_i) est tiré selon une loi $P(X, Y)$
- On a obtenu le modèle $\hat{f}_{\mathcal{D}}(x)$ à l'issue de l'apprentissage
- On suppose qu'il existe une vraie fonction $Y = f(X) + \epsilon$, où ϵ est un bruit de moyenne nulle et de variance σ^2.

Compromis biais/variance

Formalisation :

- On dispose d'un jeu de données $\mathcal{D} = \{(xi, y_i)\}_{i=1,\dots,n}$
- On considère que (x_i, y_i) est tiré selon une loi $P(X, Y)$
- On a obtenu le modèle $\hat{f}_{\mathcal{D}}(x)$ à l'issue de l'apprentissage
- On suppose qu'il existe une vraie fonction $Y = f(X) + \epsilon$, où ϵ est un bruit de moyenne nulle et de variance σ^2.

L'erreur faite en $x = x_0$ par le modèle $\hat{f}_{\mathcal{D}}$ est donnée par :

$$E\left[(Y - \hat{f}_{\mathcal{D}}(x_0))^2 | X = x_0\right]$$

Elle dépend :

1. de la variabilité intrinsèque : $Y = f(X) + \epsilon$
2. du fait que $\hat{f}_{\mathcal{D}}$ ait été obtenu sur le jeu (aléatoire) $\mathcal{D}$

Compromis biais variance

L'erreur faite en $x = x_0$ par le modèle $\hat{f}_{\mathcal{D}}$ est donnée par :

$$E\left[(Y - \hat{f}_{\mathcal{D}}(x_0))^2 | X = x_0\right] = E_Y E_{\mathcal{D}}\left[(Y - \hat{f}_{\mathcal{D}}(x_0))^2 | X = x_0\right]$$

$\Rightarrow$ On peut la décomposer comme[4] :

$$\begin{aligned} E\left[(Y - \hat{f}_{\mathcal{D}}(x_0))^2 | X = x_0\right] = & E_Y\left[\left(Y - f(x_0)\right)^2\right] \\ & + \left(f(x_0) - E_{\mathcal{D}}\left[\hat{f}_{\mathcal{D}}(x_0)\right]\right)^2 \\ & + E_{\mathcal{D}}\left[\left(E_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x_0)] - \hat{f}_{\mathcal{D}}(x_0)\right)^2\right] \end{aligned}$$

4. Voir Hastie et al. (2001) + on supprime le conditionnement $X = x_0$.

Compromis biais variance

$$E\left[(Y - \hat{f}_{\mathcal{D}}(x_0))^2 | X = x_0\right] = E_Y\left[\left(Y - f(x_0)\right)^2\right] \quad (1)$$

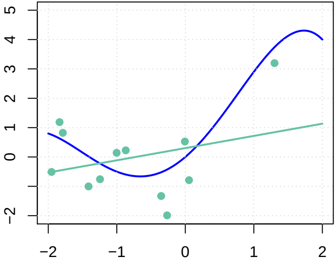
$$+ \left(f(x_0) - E_{\mathcal{D}}\left[\hat{f}_{\mathcal{D}}(x_0)\right]\right)^2 \quad (2)$$

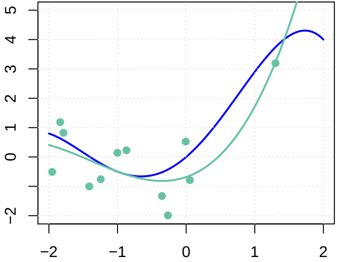
$$+ E_{\mathcal{D}}\left[\left(E_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x_0)] - \hat{f}_{\mathcal{D}}(x_0)\right)^2\right] \quad (3)$$

- (1) est une erreur irréductible liée à la relation non déterministe entre Y et X
  - $P(Y|X = x_0)$: cette erreur est hors de notre contrôle
- (2) est le (carré du) biais du modèle $\hat{f}_{\mathcal{D}}$
  - l'écart entre la vraie fonction et la modèle moyen (selon les données d'apprentissage)
- (3) est la variance du modèle $\hat{f}_{\mathcal{D}}$
  - la variabilité de $\hat{f}_{\mathcal{D}}$ autour du modèle moyen quand le jeu de données change

Compromis biais variance - illustration

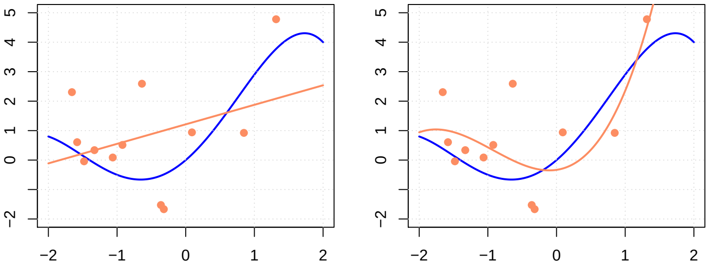
$\Rightarrow$ 8 régressions linéaire et polynomiale ($d = 7$) :





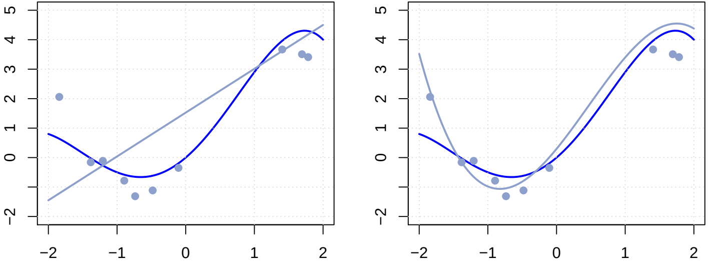
Compromis biais variance - illustration

$\Rightarrow$ 8 régressions linéaire et polynomiale ($d = 7$) :



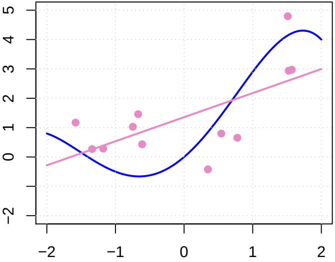
Compromis biais variance - illustration

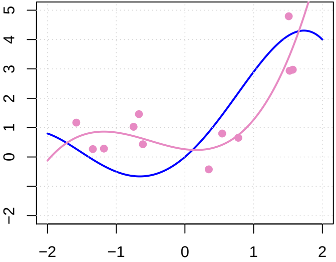
$\Rightarrow$ 8 régressions linéaire et polynomiale ($d = 7$) :



Compromis biais variance - illustration

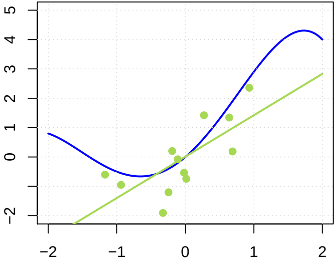
$\Rightarrow$ 8 régressions linéaire et polynomiale ($d = 7$) :

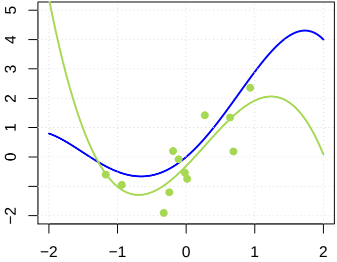




Compromis biais variance - illustration

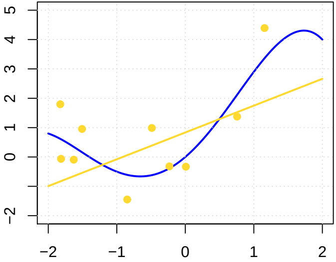
$\Rightarrow$ 8 régressions linéaire et polynomiale ($d = 7$) :

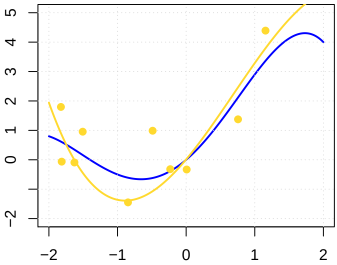




Compromis biais variance - illustration

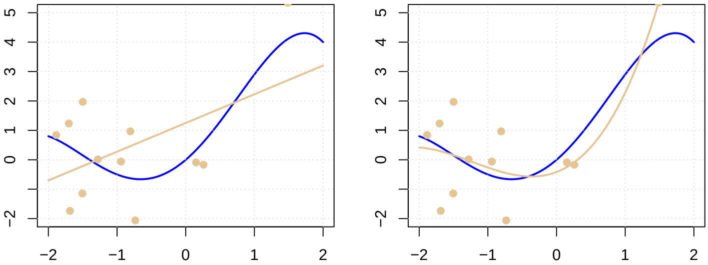
$\Rightarrow$ 8 régressions linéaire et polynomiale ($d = 7$) :





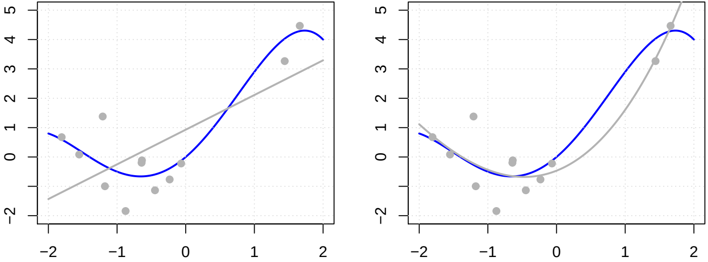
Compromis biais variance - illustration

$\Rightarrow$ 8 régressions linéaire et polynomiale ($d = 7$) :



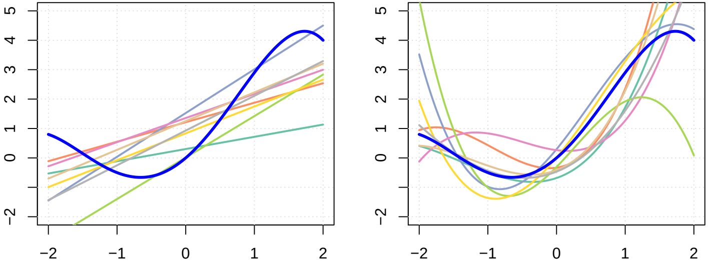
Compromis biais variance - illustration

$\Rightarrow$ 8 régressions linéaire et polynomiale ($d = 7$) :



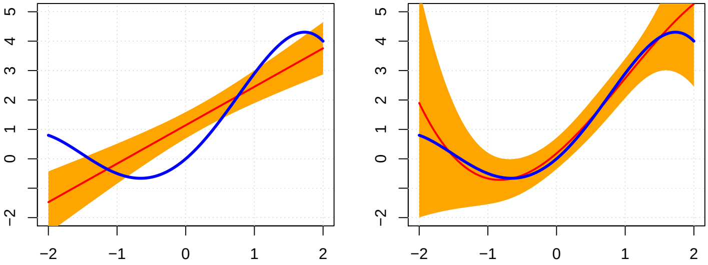
Compromis biais variance - illustration

⇒ résumé :



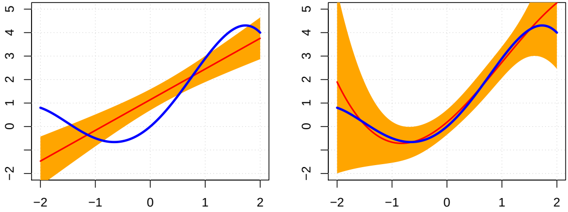
Compromis biais variance - illustration

⇒ modèle moyen et variabilité sur 1000 tirages :



Compromis biais variance

⇒ **Compromis** :



Avec des classes de fonctions simples :

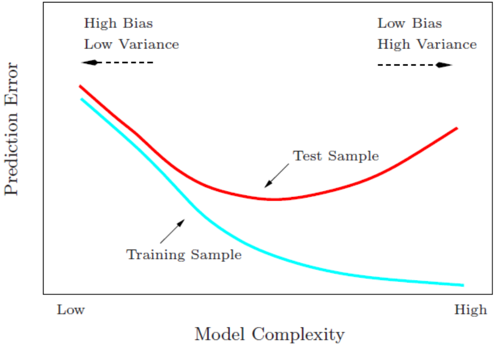
- on approxime en moyenne mal les données : biais élevé
- solution stable selon les échantillons : variance faible

Avec des classes de fonctions complexes :

- on approxime en moyenne bien les données : biais faible
- solution instable selon les échantillons : variance élevée

Compromis biais/variance

Question clé : trouver le bon niveau de complexité pour éviter le sous- et le sur-apprentissage.



⇒ besoin d'un moyen d'estimer l'erreur de généralisation

Estimation de performance et validation croisée

Le paradigme train/test

A partir du jeu de données on doit résoudre deux problèmes :

1. trouver le bon niveau de complexité du modèle
   - compromis biais/variance et sous/sur-apprentissage
2. estimer ses performances de généralisation
   - performances sur de nouvelles données

Le paradigme train/test

A partir du jeu de données on doit résoudre deux problèmes :

1. trouver le bon niveau de complexité du modèle
   - compromis biais/variance et sous/sur-apprentissage
2. estimer ses performances de généralisation
   - performances sur de nouvelles données

Paradigme de l'apprentissage supervisé :

- données d'apprentissage pour construire le modèle
- données de test pour évaluer les performances

Le paradigme train/test

A partir du jeu de données on doit résoudre deux problèmes :

1. trouver le bon niveau de complexité du modèle
   - compromis biais/variance et sous/sur-apprentissage
2. estimer ses performances de généralisation
   - performances sur de nouvelles données

Paradigme de l'apprentissage supervisé :

- données d'apprentissage pour construire le modèle
- données de test pour évaluer les performances

Attention : données de test uniquement utilisées à la toute fin pour évaluer les performances du modèle final

- n'interviennent **jamais** dans la construction du modèle

Le paradigme train/test

Pour optimiser la complexité du modèle :

- besoin d'estimer les performances de généralisation
- mais sans faire appel aux données de test

Pourquoi ?

- les données de test ne permettent que d'estimer l'erreur de généralisation
  - indicateurs de performance + intervalles de confiance
- optimiser le modèle pour maximiser les performances sur **CE jeu de test** serait une forme de sur-apprentissage !
  - et serait donc **optimiste** : la généralisation serait moins bonne

Contrôler la complexité du modèle

Première solution : découpage train / validation / test :

1. train : pour apprendre les différents modèles
2. validation : pour les évaluer et retenir le meilleur
3. test : pour estimer ses performances

$\Rightarrow$ situation optimale - "data rich"

- validation suffisamment grand pour bien estimer l'erreur

Deuxième solution : validation-croisée

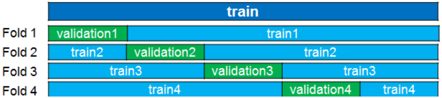
Validation croisée

Si peu de données : délicat de découper en train/validation

- forte incertitude sur l'estimation des performances

Principe de la validation croisée :

- découper le jeu d'apprentissage en K parties - les folds
  - **les données de test sont toujours de côté**
  - si on prend $K = n$ on parle de "leave one out"
- pour $k = 1, ..., K$:
  - fold k = données de validation
  - autres folds = données d'apprentissage



$\Rightarrow$ on évalue les performance sur tout le jeu de données

Validation croisée

Pseudo-code :

1. Définir les K folds de validation croisée
   - en pratique : un vecteur de longueur n avec des valeurs entre 1 et K affectant les n observations aux K folds
2. Pour $k = 1$ à K :
   2.1 mettre de côté la k-ième fold
   2.2 apprendre le modèle sur les $(K - 1)$ folds restantes
   2.3 appliquer le modèle sur les données de la k-ième fold
3. Evaluer les performances du modèle en comparant les valeurs réelles et prédites.
   - estimation globale ou par fold

Validation croisée et sélection de modèle

La validation croisée est notamment utile pour choisir le meilleur modèle entre plusieurs modèles candidats.

- e.g., des modèles + ou - complexes

Pseudo-code :

1. Définir un ensemble de modèles candidats
   - régression polynomiale : différents degrés de polynôme
   - k-PPV : différentes valeurs de k
   - ...
2. Pour chaque modèle :
   2.1 Appliquer la procédure de validation croisée
   2.2 Enregistrer les performances de prédiction
3. Choisir le meilleur modèle.
4. Le construire sur tout le jeu d'apprentissage.
5. L'appliquer sur le jeu de test et estimer sa performance

Théorie de la décision statistique

Apprentissage supervisé - formalisation

Données d'entrée : échantillon $\{(x_i, y_i)\}_{i=1,\dots,n} \in \mathcal{X} \times \mathcal{Y}$.

$\Rightarrow$ Objectif : apprendre une fonction $f : \mathcal{X} \to \mathcal{Y}$ permettant de prédire la réponse associée à une nouvelle observation.

Apprentissage supervisé - formalisation

Données d'entrée : échantillon $\{(x_i, y_i)\}_{i=1,\dots,n} \in \mathcal{X} \times \mathcal{Y}$.

$\Rightarrow$ Objectif : apprendre une fonction $f : \mathcal{X} \to \mathcal{Y}$ permettant de prédire la réponse associée à une nouvelle observation.

Critère : une fonction de perte L (pour "loss") mesurant l'erreur entre y et $f(x)$.

Apprentissage supervisé - formalisation

Données d'entrée : échantillon $\{(x_i, y_i)\}_{i=1,\ldots,n} \in \mathcal{X} \times \mathcal{Y}$.

$\Rightarrow$ Objectif : apprendre une fonction $f : \mathcal{X} \to \mathcal{Y}$ permettant de prédire la réponse associée à une nouvelle observation.

Critère : une fonction de perte L (pour "loss") mesurant l'erreur entre y et $f(x)$.

Typiquement :

- l'erreur quadratique pour la régression :

$$L(y, f(x)) = (y - f(x))^2$$

- le coût 0/1 pour la classification :

$$L(y, f(x)) = \mathbb{1}(y \neq f(x))$$

Apprentissage supervisé - formalisation

Cadre probabiliste : on considère que nos observations (x_i, y_i) sont des variables aléatoires régies par une loi jointe $P(X, Y)$.

Apprentissage supervisé - formalisation

Cadre probabiliste : on considère que nos observations (x_i, y_i) sont des variables aléatoires régies par une loi jointe $P(X, Y)$.

$\Rightarrow$ L'objectif de l'apprentissage supervisé est donc de trouver la fonction f minimisant l'espérance de la fonction de perte :

$$R(f) = E_{X,Y}\big[L(Y, f(X))\big],$$

à partir d'un échantillon $\big\{(x_i, y_i)\big\}$, $i = 1, ..., n$.

Apprentissage supervisé - formalisation

Cadre probabiliste : on considère que nos observations (x_i, y_i) sont des variables aléatoires régies par une loi jointe $P(X, Y)$.

$\Rightarrow$ L'objectif de l'apprentissage supervisé est donc de trouver la fonction f minimisant l'espérance de la fonction de perte :

$$R(f) = E_{X,Y}\big[L(Y, f(X))\big],$$

à partir d'un échantillon $\big\{(x_i, y_i)\big\}$, $i = 1, ..., n$.

$R(f)$ est appelée le risque (ou la perte) de la fonction f.

- $R_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n} L\big(y_i, f(x_i)\big)$ est le **risque empirique**.

Théorie de la décision statistique - régression

La quête du Graal : comment choisir f si on connait $P(X, Y)$

Théorie de la décision statistique - régression

La quête du Graal : comment choisir f si on connait $P(X, Y)$

Cas de la régression et de la perte quadratique :

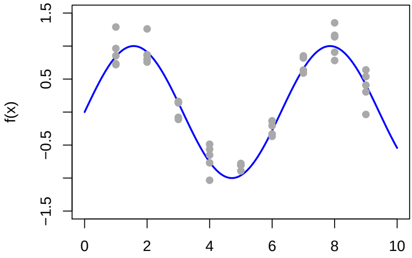
$$
\begin{aligned}
R(f) &= E_{X,Y}\left[L(Y, f(X))\right] \\
&= E_{X,Y}\left[(Y - f(X))^2\right]
\end{aligned}
$$

$\Rightarrow$ meilleure solution : $f(x) = E[Y|X = x]$

- la valeur moyenne que peut prendre Y sachant X
- la "regression function"
- NB : valable pour la perte quadratique
  - $f(x) = \text{median}(Y|X = x)$ si $L(Y, f(X)) = |Y - f(X)|$

Théorie de la décision statistique - régression

Illustration :



- $P(X, Y)$: vraie relation entre X et Y non déterministe
  - ici, $Y = f(X) + \epsilon$, $\epsilon \to \mathcal{N}(0, \sigma^2)$
  - en bleue : vraie fonction, en gris : réalisations bruitées
- On doit prendre une décision
- On minimise la perte quadratique en prenant $E[Y|X]$
  - l'espérance des points gris pour chaque valeur de x

Théorie de la décision statistique - régression

Régression et perte quadratique : démonstration

$$\begin{aligned} R(f) &= E_{X,Y}\left[L(Y, f(X))\right] \\ &= E_{X,Y}\left[(Y - f(X))^2\right] \end{aligned}$$

Théorie de la décision statistique - régression

Régression et perte quadratique : démonstration

$$\begin{aligned} R(f) &= E_{X,Y}\left[L(Y, f(X))\right] \\ &= E_{X,Y}\left[(Y - f(X))^2\right] \end{aligned}$$

$\Rightarrow$ on conditionne sur X : $E_{Y,X}(.) = E_X E_{Y|X}(.)$

$$R(f) = E_X E_{Y|X}\left[(Y - f(X))^2 | X\right]$$

Théorie de la décision statistique - régression

Régression et perte quadratique : démonstration

$$
\begin{aligned}
R(f) &= E_{X,Y}\left[L(Y, f(X))\right] \\
&= E_{X,Y}\left[(Y - f(X))^2\right]
\end{aligned}
$$

$\Rightarrow$ on conditionne sur X : $E_{Y,X}(.) = E_X E_{Y|X}(.)$

$$
R(f) = E_X E_{Y|X}\left[(Y - f(X))^2 | X\right]
$$

$\Rightarrow$ on minimise pour chaque valeur x prise par X :

$$
\begin{aligned}
f(x) &= \arg\min_c E_{Y|X}\left[(Y - c)^2 | X = x\right] \\
&= E[Y|X = x]
\end{aligned}
$$

Théorie de la décision statistique - classification

La quête du Graal : comment choisir f si on connait $P(X, Y)$

Cas de la classification et de la perte 0/1 :

$$\begin{aligned} R(f) &= E_{X,Y}\left[L(Y, f(X))\right] \\ &= E_{X,Y}\left[\mathbb{1}(Y \neq f(X)\right] \end{aligned}$$

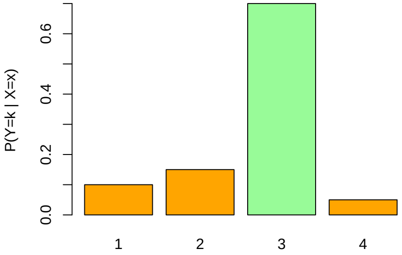
$\Rightarrow$ meilleure solution : $f(x) = \arg \max_{k=1,\dots,K} P(Y = C_k | X = x)$

- la classe la plus vraisemblable sachant X
- le classifieur de Bayes
- NB : valable pour la perte / le coût 0/1
  - se généralise pour des coûts arbitraires $L(C_i, C_j)$

Théorie de la décision statistique - classification

Illustration : $P(Y = k|X = x)$

- pour une valeur x donnée et $K = 4$ catégories



- $P(X, Y)$: vraie relation entre X et Y non déterministe
- On doit prendre une décision
- Erreur : somme des probabilités des choix qu'on rejette
- Minimisée si on choisit la probabilité la plus élevée

Théorie de la décision statistique - classification

Classification et perte 0/1 : démonstration

$$R(f) = E_{X,Y}\left[L(Y, f(X))\right] = E_{X,Y}\left[\mathbb{1}(Y \neq f(X)\right]$$

Théorie de la décision statistique - classification

Classification et perte 0/1 : démonstration

$$R(f) = E_{X,Y}\left[L(Y, f(X))\right] = E_{X,Y}\left[\mathbb{1}(Y \neq f(X))\right]$$

$\Rightarrow$ on conditionne sur X : $E_{Y,X}(.) = E_X E_{Y|X}(.)$

$$\begin{aligned} R(f) &= E_X E_{Y|X}\left[\mathbb{1}(Y \neq f(X)|X\right] \\ &= E_X \sum_{k=1}^{K} \mathbb{1}(C_k \neq f(X)) P(Y = C_k|X) \end{aligned}$$

Théorie de la décision statistique - classification

Classification et perte 0/1 : démonstration

$$R(f) = E_{X,Y}\left[L(Y, f(X))\right] = E_{X,Y}\left[\mathbb{1}(Y \neq f(X)\right]$$

$\Rightarrow$ on conditionne sur X : $E_{Y,X}(.) = E_X E_{Y|X}(.)$

$$\begin{aligned} R(f) &= E_X E_{Y|X}\left[\mathbb{1}(Y \neq f(X)|X\right] \\ &= E_X \sum_{k=1}^{K} \mathbb{1}(C_k \neq f(X))P(Y = C_k|X) \end{aligned}$$

$\Rightarrow$ on minimise pour chaque valeur x prise par X :

$$\begin{aligned} f(x) &= \arg\min_c \sum_{k=1}^{K} \mathbb{1}(C_k \neq c)P(Y = C_k|X = x) \\ &= \arg\max_{k=1,\ldots,K} P(Y = C_k|X = x) \end{aligned}$$

Théorie de la décision statistique

En pratique...

Théorie de la décision statistique

En pratique... on ne connaît pas $P(X, Y)$! !

Théorie de la décision statistique

En pratique... on ne connaît pas $P(X, Y)$! !

Intérêt de cette démarche (théorique) :

- concevoir des algorithmes
  - définition d'estimateurs + stratégies d'estimation
- analyser des algorithmes
  - e.g., se comparer au classifieur de Bayes par simulations
  - étudier performance en fonction de n

L'algorithme des k plus proches voisins

Algorithme des k-PPV

Algorithme des k-plus proches voisins :

1. trouver les k observations x_i les plus proches de l'observation x' à classifier
2. définir $f(x')$ en fonction des réponses y_i des k-PPV
   - régression : valeur moyenne
   - classification : vote majoritaire

$\Rightarrow$ le B-A BA des algorithmes de prédiction

Algorithme des k-PPV

Algorithme des k-plus proches voisins :

1. trouver les k observations x_i les plus proches de l'observation x' à classifier
2. définir $f(x')$ en fonction des réponses y_i des k-PPV
   - régression : valeur moyenne
   - classification : vote majoritaire

$\Rightarrow$ le B-A BA des algorithmes de prédiction

Approche de mémorisation

- + : très simple à mettre en oeuvre
- - : passage à l'échelle

Algorithme des k-PPV

Algorithme des k-plus proches voisins :

1. trouver les k observations x_i les plus proches de l'observation x' à classifier
2. définir $f(x')$ en fonction des réponses y_i des k-PPV
    - régression : valeur moyenne
    - classification : vote majoritaire

$\Rightarrow$ le B-A BA des algorithmes de prédiction

Approche de mémorisation

- + : très simple à mettre en oeuvre
- - : passage à l'échelle

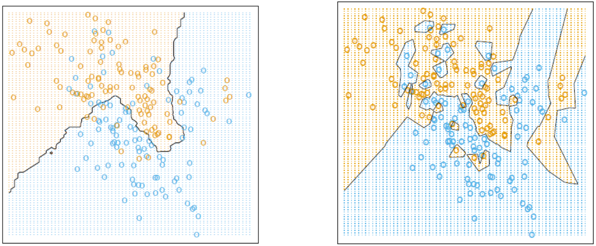
Questions ouvertes :

- choix du critère de distance et de la valeur de k

Algorithme des k-PPV - complexité

Illustration tirée de Hastie et al. (2001) :

- à gauche : $k = 15$; à droite : $k = 1$.



Des petites valeurs de k conduisent à des modèles plus locaux et donc (en général) plus complexes.

k-PPV & théorie de l'apprentissage

k-PPV pour la régression :

$$\hat{f}(x) = \text{Average}(y_i | i \in N_k(x))$$

$\Rightarrow$ approxime directement la regression function $E[Y|X]$

k-PPV & théorie de l'apprentissage

k-PPV pour la régression :

$$\hat{f}(x) = \text{Average}(y_i | i \in N_k(x))$$

$\Rightarrow$ approxime directement la regression function $E[Y|X]$

Nature de l'approximation :

1. espérance $\rightarrow$ moyenne empirique
2. valeur ponctuelle $\rightarrow$ voisinage (dans le conditionnement)

$\Rightarrow$ convergence asymptotique

- $n, k \rightarrow +\infty$; $k/n \rightarrow 0$

k-PPV & théorie de l'apprentissage

k-PPV pour la classification :

$$\hat{f}(x) = \text{Majority}(y_i | i \in N_k(x))$$
$$= \arg\max_{l=1,\ldots,K} \tilde{P}_l = \frac{1}{k} \sum_{i \in N_k(x))} \mathbb{1}(y_i = l)$$

$\Rightarrow$ approxime directement le classifieur de Bayes :

$$\arg\max_{l=1,\ldots,K} P(Y = C_l | X = x)$$

k-PPV & théorie de l'apprentissage

k-PPV pour la classification :

$$\hat{f}(x) = \text{Majority}(y_i | i \in N_k(x))$$
$$= \arg\max_{l=1,\dots,K} \tilde{P}_l = \frac{1}{k} \sum_{i \in N_k(x))} \mathbb{1}(y_i = l)$$

$\Rightarrow$ approxime directement le classifieur de Bayes :

$$\arg\max_{l=1,\dots,K} P(Y = C_l | X = x)$$

Nature de l'approximation :

1. probabilité $\rightarrow$ proportion empirique
2. valeur ponctuelle $\rightarrow$ voisinage (dans le conditionnement)

$\Rightarrow$ convergence asymptotique

- $n, k \rightarrow +\infty\,;\ k/n \rightarrow 0$

k-PPV & théorie de l'apprentissage

En dépit de sa simplicité, l'algorithme des k-PPV :

1. approxime les bonnes fonctions
   - regression function & classifieur de Bayes
2. possède des propriétés de convergence

$\Rightarrow$ pourquoi chercher plus loin ?

k-PPV & théorie de l'apprentissage

En dépit de sa simplicité, l'algorithme des k-PPV :

1. approxime les bonnes fonctions
   - regression function & classifieur de Bayes
2. possède des propriétés de convergence

$\Rightarrow$ pourquoi chercher plus loin ?

Car la convergence est asymptotique :

- $n, k \to +\infty$; $k/n \to 0$

$\Rightarrow$ en pratique on dispose d'un nombre d'observations limité

k-PPV & théorie de l'apprentissage

En dépit de sa simplicité, l'algorithme des k-PPV :

1. approxime les bonnes fonctions
   - regression function & classifieur de Bayes
2. possède des propriétés de convergence

$\Rightarrow$ pourquoi chercher plus loin ?

Car la convergence est asymptotique :

- $n, k \to +\infty$; $k/n \to 0$

$\Rightarrow$ en pratique on dispose d'un nombre d'observations limité

(De plus, ça se complique en haute dimension)

- "fléau de la dimension"

Remarques et conclusions

Conclusion

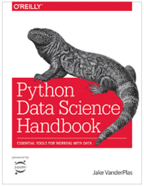
- Introduction aux fondements théoriques de l'apprentissage supervisé
- Complexité des modèles et sur-apprentissage
  - illustration avec régression polynomiale
- Compromis biais-variance
- Erreur de généralisation et validation croisée
- Théorie de la decision statistique
  - regression function et classifieur de Bayes
- Algorithme des k-ppv
  - classification et régression

Mise en oeuvre en Python

Python et Data Science

Packages fondamentaux :

- NumPy : calcul matriciel
- MatPlotLib : visualisation
- Pandas : manipulations de tableaux de données
- Scikit-Learn : machine learning



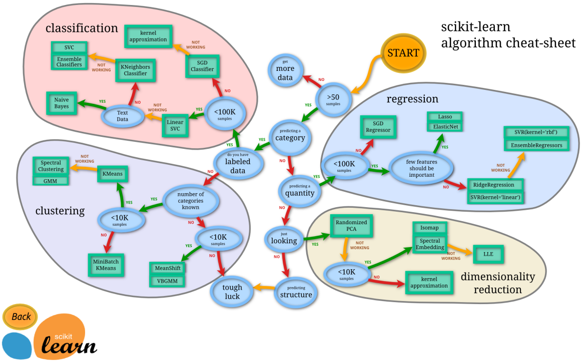
$\Rightarrow$ Bonne introduction : Python Data Science Handbook (VanderPlas, 2017)

$\Rightarrow$ Ce cours : centré sur Scikit-Learn

- pré-requis : les bases de NumPy et MatPlotLib

Scikit-Learn [5]

- très populaire en machine-learning
- très complet
- simple à utiliser (API uniformisée, types standards)



5. http://scikit-learn.org/

Scikit-Learn in a nutshell....

Une collection de modules thématiques

- e.g., le module `linear_model`
- e.g., le module `model_selection`

contenant :

1. des classes définissant des estimateurs
   - conception objet : estimateurs = attributs + méthodes
   - e.g., la classe `LinearRegression` du module `linear_model`
   - e.g., la classe `PCA` du module `decomposition`
2. des fonctions réalisant certains traitements
   - e.g., la fonction `cross_val_score` du module `model_selection`.

Scikit-Learn - estimateurs

API standardisée : les mêmes méthodes clés pour tous les estimateurs :

- `fit` : optimise le modèle à partir des données
- pour les estimateurs "de prédiction" :
  - `predict` : réalise la prédiction sur des données
  - e.g., régression linéaire, k-ppv
- pour les estimateurs "de transformation" :
  - `transform` : applique la transformation aux données (apprentissage ou nouvelles)
  - `fit_transform` : applique `fit` + `transform` aux données d'apprentissage
  - e.g, ACP

Attributs définis lors de l'initialisation par le constructeur.

Scikit-Learn - B-A BA

Pour optimiser et appliquer un modèle :

1. importer la classe correspondante : :
   ```
   from sklearn.linear_model import LinearRegression
   ```
2. instancier le modèle :
   ```
   linReg = LinearRegression()
   ```
   - NB : c'est ici qu'on spécifie des (hyper)paramètres
   - e.g., régression avec ou sans intercept
3. estimer – "fitter" – le modèle sur les données :
   ```
   linReg.fit(X,y)
   ```
4. appliquer le modèle sur des données :
   ```
   preds = linReg.predict(Xtest)
   ```
   - NB : ou la méthode `transform` pour l'ACP par exemple

Scikit-Learn - exemples

Régression linéaire :

```
# load class
from sklearn.linear_model import LinearRegression
# instanciate model
linReg = LinearRegression()
# learn model
linReg.fit(x,y)
# make predictions
lingReg.predict(x_test)
```

ACP :

```
# load class
from sklearn.decomposition import PCA
# instanciate model - consider 2 components
pca = PCA(n_components = 2)
# learn model
pca.fit(X)
# transform (training) data
Xpca = pca.transform(X)
```

Scikit-Learn - TP1

Dans le TP1 nous manipulerons :

1. la classe `PCA`
   - ACP, module `decomposition`
2. la classe `LinearRegression`
   - régression linéaire, module `linear_model`
3. la classe `PolynomialFeatures`
   - expansion polynomiale, module `preprocessing`
4. la classe `KNeighborsClassifier`
   - algorithme des k-ppv, module `neighbors`

Références

T. Hastie, R. Tibshirani, and J.. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

J. VanderPlas. *Python Data Science Handbook*. O'Reilly, 2017.