

# MAPSI — cours 8 : Régressions

Pierre-Henri Wuillemin & Raphaël Fournier-S'niehotta  
(& Nicolas Thome)

`pierre-henri.wuillemin@lip6.fr`

LIP6 / ISIR – Sorbonne Université, France

- Jusqu'ici, beaucoup de problèmes de **classification**
  - supervisés (chiffres, lettres)
  - non-supervisés (geyser)
- D'autres problèmes existent...
  - suivi de cibles (cf cours 10)
  - modélisation explicative
  - **régression** : modèle expliquant une variable continue
- Sources de données
  - [www.kaggle.com](http://www.kaggle.com)
  - <http://archive.ics.uci.edu/ml/>

- Prédiction des prix des maisons (Boston)
- Prédiction des notes d'un vin
- Prédiction du prix des voitures d'occasion
- Résistance du béton
- Propagation des feux de forêt
- Consommation électrique
- Eruptions solaires
- ...

# Régression simple (1)

- $X$  et  $Y$  jouent des rôles dissymétriques
- $Y$  = variable expliquée = variable endogène
- on veut « expliquer » la valeur de  $Y$  par celle de  $X$
- on veut estimer la valeur de  $Y$  en fonction de celle de  $X$



$X$  = taux d'alcool dans le sang  $\implies Y$  = vitesse



$X$  = surface du logement  $\implies Y$  = prix au  $m^2$



$X$  = quantité d'engrais à l'hectare  $\implies Y$  = rendement

Variable exogène  $X$  peut être aléatoire, mais pas forcément :



⇒ l'expérimentateur peut faire varier comme il veut la quantité d'engrais de parcelle en parcelle

## Hypothèses

- relation imprécise entre  $X$  et  $Y$
- valeur de  $Y$  dépend de  $X$  et d'un facteur aléatoire  $\mathcal{E}$  :  
 $Y = f(X, \mathcal{E})$
- $\mathcal{E}$  = résidu = erreur = bruit

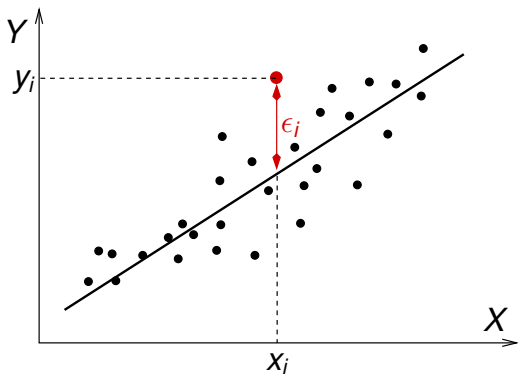
$$Y = f(X, \mathcal{E})$$

- $\mathcal{E}$  variable aléatoire  $\implies Y$  variable aléatoire

## *Modèle linéaire ou régression*

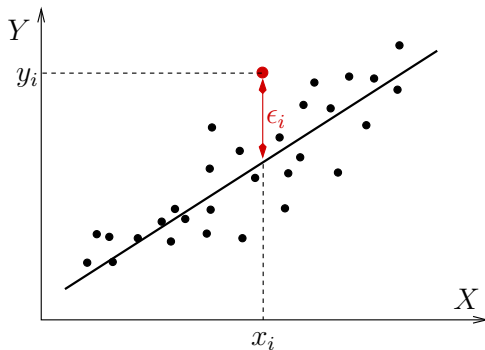
- On dispose de  $n$  observations  $(x_i, y_i)$  du couple  $(X, Y)$
- fonction  $f$  affine :  $Y = \alpha + \beta X + \mathcal{E}$
- $\alpha$  et  $\beta$  : paramètres inconnus
- Les observations vérifient  $y_i = \alpha + \beta x_i + \mathcal{E}_i$
- existence des résidus  $\mathcal{E}_i \sim \mathcal{E}$ 
  - $\implies$  les points  $(x_i, y_i)$  ne sont pas sur une même droite
  - $\implies$  on ne peut déterminer exactement  $\alpha$  et  $\beta$
  - $\implies$  **estimation de  $\alpha$  et  $\beta$**

## Régression simple (4)



$$Y = \alpha + \beta X + \varepsilon$$

- Cas simple : régression linéaire mono-dimensionnelle



Modélisation :  $Y = \alpha + \beta X + \mathcal{E}$

On dispose d'un ensemble d'observations  $(x_i, y_i)$

⇒ trouver  $\alpha^*, \beta^*$  qui minimisent les erreurs  $\mathcal{E}_i$



- Modélisation :  $Y = \alpha + \beta X + \mathcal{E}$
- $\mathcal{E}$  est une variable aléatoire,  $\{\dots, \mathcal{E}_i, \dots\}$  sont des tirages selon cette loi
- Hypothèse (dite du bruit blanc) :  $\mathcal{E} \sim \mathcal{N}(0, \sigma)$
- Notations :
  - $Y_i = \alpha + \beta X_i + \mathcal{E}_i$  et :  $E[Y_i] = \alpha + \beta x_i$ ,  $V[Y_i] = \sigma^2$
  - On note  $Y_i \sim \mathcal{N}(\alpha + \beta x_i, \sigma)$

Comment trouver  $\alpha$  et  $\beta$  ?

$$Y = \alpha + \beta X + \varepsilon$$

$$\Rightarrow \begin{cases} E(Y) = \alpha + \beta E(X) \\ Y - E(Y) = \beta(X - E(X)) + \varepsilon \end{cases}$$

- Multiplication par  $(X - E(X))$  et passage à l'espérance :

$$E[(Y - E(Y))(X - E(X))] = \beta E[(X - E(X))^2] + E[\varepsilon(X - E(X))]$$

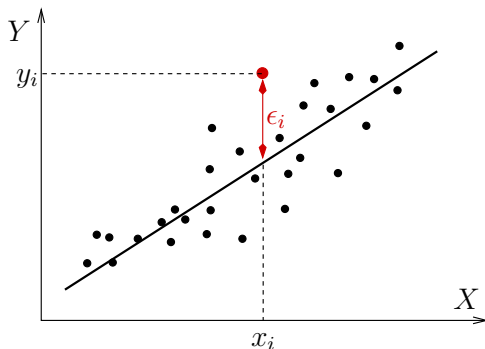
$$\Rightarrow \text{cov}(X, Y) = \beta \sigma_X^2 + \text{cov}(\varepsilon, X)$$

- or  $\text{cov}(\varepsilon, X) = 0$  par hypothèse (bruit)

$$\beta^* = \frac{\text{cov}(X, Y)}{\sigma_X^2} \quad \alpha^* = E(Y) - \frac{\text{cov}(X, Y)}{\sigma_X^2} E(X)$$

# Conclusion

On peut trouver l'équation de la droite qui explique les points (avec des hypothèses sur  $\mathcal{E}$ )



$$\beta^* = \frac{\text{cov}(X, Y)}{\sigma_X^2} \quad \alpha^* = E(Y) - \frac{\text{cov}(X, Y)}{\sigma_X^2} E(X)$$

# Covariance et corrélation

- Covariance : représente la dépendance (linéaire) entre  $X$  et  $Y$

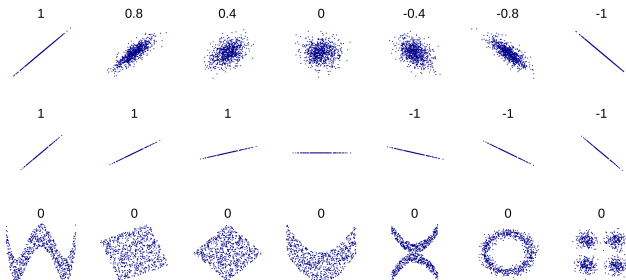
$$\text{cov}(X, Y) = E[(Y - E(Y))(X - E(X))]$$

- Coefficient de corrélation linéaire  $r$

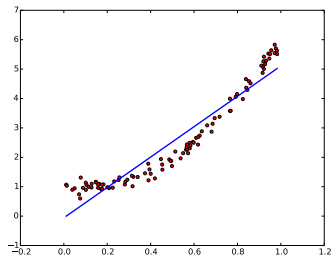
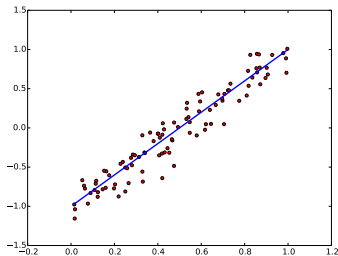
$$r = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$

- Normalisation par les variance  $\sigma_X \sigma_Y$

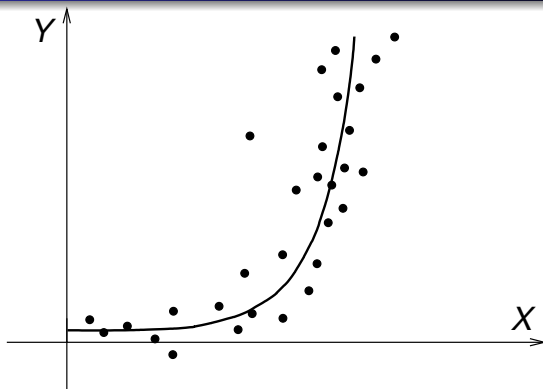
- $r \in [-1; 1]$



- Ca marche bien... sur des données linéaires



# Changement de variable



$$Y = e^{-1+0,5X^2} \Rightarrow \ln Y = -1 + 0,5X^2$$

$\Rightarrow$  changement de variables :  $Y' = \ln Y$  et  $X' = X^2$

$\Rightarrow Y' = -1 + 0,5X'$

**Généralisation au cas multi-dimensionnel plus loin**

- On dispose toujours d'observations iid  $\{(x_i, y_i)\}_{i=1, \dots, N}$  et on fait toujours une hypothèse gaussienne sur le bruit
- Généralisation à n'importe quel modélisation  $Y = f(X)$ ,  
(par exemple)  $Y = \alpha X^2 + \beta X + \gamma + \mathcal{E}$
- Notations :
  - $Y_i \sim \mathcal{N}(f(x_i), \sigma)$
  - Proba. d'observation pour  $(y_i, x_i)$  :

$$p(y_i | x_i, \theta, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2} \|y_i - f(x_i)\|^2\right)$$

- Vraisemblance pour la base :

$$\mathcal{L} = p(\mathbf{y} | \mathbf{x}, \theta, \sigma) = \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2} \|y_i - f(x_i)\|^2\right)$$

- Comment maximiser la vraisemblance ?

$$\mathcal{L} = p(\mathbf{y}|\mathbf{x}, \theta, \sigma) = \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(y_i - f(x_i))^2\right)$$

- On fait souvent l'hypothèse que  $\sigma$  est connu
- Passage au log :

$$\log \mathcal{L} = \sum_i -\log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}(y_i - f(x_i))^2$$

## Approche standard :

- Calcul du gradient (dérivé)
- Annulation du gradient
  - Analytique (si possible)
  - Itérative (sinon)

Définition : gradient = vecteur des dérivées par rapport aux paramètres



- Simplification (si  $\sigma$  est connu), et  $f(x) = \alpha x^2 + \beta x + \gamma$

$$\arg \max_{\alpha, \beta, \gamma} \sum_i -\log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}(y_i - f(x_i))^2 = \arg \max_{\alpha, \beta, \gamma} \sum_i -(y_i - f(x_i))^2$$

- Calcul du gradient ( $\nabla \mathcal{L} = \nabla_{\alpha, \beta, \gamma} \mathcal{L}$ ) :

$$\nabla_{\alpha, \beta, \gamma} \mathcal{L} = \begin{bmatrix} \frac{\partial(\sum_i -(y_i - f(x_i))^2)}{\partial \alpha} \\ \frac{\partial(\sum_i -(y_i - f(x_i))^2)}{\partial \beta} \\ \frac{\partial(\sum_i -(y_i - f(x_i))^2)}{\partial \gamma} \end{bmatrix} = \begin{bmatrix} \sum_i 2x_i^2(y_i - \alpha x_i^2 - \beta x_i - \gamma) \\ \sum_i 2x_i(y_i - \alpha x_i^2 - \beta x_i - \gamma) \\ \sum_i 2(y_i - \alpha x_i^2 - \beta x_i - \gamma) \end{bmatrix}$$

Bonne ou mauvaise nouvelle ?

- **Très bonne nouvelle !** Ces équations forment un système de  $n$  équations linéaires à  $n$  inconnues

$$\nabla_{\alpha, \beta, \gamma} \log \mathcal{L} = 0 \Leftrightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- Résolution par factorisation matricielle (LU, QR, Choleski...)
- En python :

- `numpy.linalg.solve` :

`numpy.linalg.solve(a, b)`

Solve a linear matrix equation, or system of linear scalar equations.

Computes the "exact" solution,  $x$ , of the well-determined, i.e., full rank, linear matrix equation  $ax = b$ .

**Parameters:** `a : (..., M, M) array_like`

Coefficient matrix.

`b : {(..., M), (..., M, K)}, array_like`

Ordinate or "dependent variable" values.

**Returns:** `x : {(..., M), (..., M, K)} ndarray`

Solution to the system  $ax = b$ . Returned shape is identical to  $b$ .

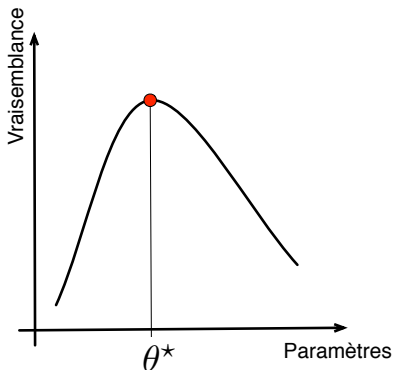
**Raises:** `LinAlgError` :

If  $a$  is singular or not square.

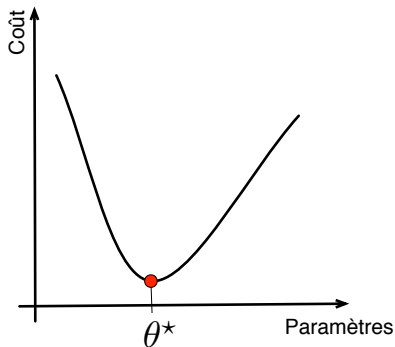
- `sklearn`

## Approches probabilistes :

trouver les paramètres  $\theta^*$  qui maximisent la vraisemblance



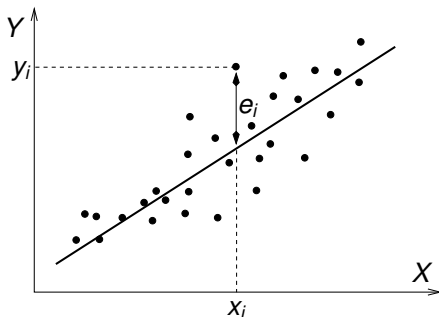
**Approches par coût :** trouver les paramètres  $\theta^*$  qui minimisent un coût défini



# Coût des moindres carrés (1)

observations  $\implies$  couples  $(x_i, y_i) \implies$  en principe  $y_i = a + bx_i$

en pratique :  $e_i = y_i - (a + bx_i) \neq 0$



$\implies$  on cherche la droite  $y = a + bx$  dont les couples sont  
**le plus proche.**

$\implies$  Minimisation de la somme des carrés des distances  
(euclidiennes) verticales entre les points et la droite ( $\min \sum e_i^2$ )

## *Programme d'optimisation*

Trouver  $a^*$  et  $b^*$  pour lesquels on a :  $\min_{a,b} \sum_{i=1}^n e_i^2$

ou encore :  $F(a, b) = \sum_{i=1}^n [y_i - a - bx_i]^2 \implies (a^*, b^*) = \arg \min_{a,b} F(a, b)$

dérivées partielles = 0 (conditions suffisantes d'optimalité) :

$$\frac{\partial F(a, b)}{\partial a} = \sum_{i=1}^n (-2)[y_i - a - bx_i] = 0$$

$$\frac{\partial F(a, b)}{\partial b} = \sum_{i=1}^n (-2)x_i[y_i - a - bx_i] = 0$$

$$\frac{\partial F(a, b)}{\partial a} = \sum_{i=1}^n (-2)[y_i - a - bx_i] = 0 \quad (1)$$

$$\frac{\partial F(a, b)}{\partial b} = \sum_{i=1}^n (-2)x_i[y_i - a - bx_i] = 0 \quad (2)$$

- **Lien avec la version analytique**

$$(1) \iff a = \bar{y} - b\bar{x}$$

$$(2) \iff b \sum_{i=1}^n x_i^2 = \sum_{i=1}^n x_i y_i - a \sum_{i=1}^n x_i \implies b = \frac{\frac{1}{n} \sum_i x_i y_i - \bar{y} \bar{x}}{\frac{1}{n} \sum_i x_i^2 - \bar{x}^2} = \frac{\text{cov}(x, y)}{s_x^2}$$

- Résolution du système d'équations linéaires :

$$\nabla_{a,b} \text{Cost} = 0 \iff \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Avec :

$$\begin{aligned} a_{11} &= n & a_{12} &= \sum_i x_i & b_1 &= \sum_i y_i \\ a_{21} &= \sum_i x_i & a_{22} &= \sum_i x_i^2 & b_2 &= \sum_i x_i y_i \end{aligned},$$

Posons  $\hat{y}_i = a + bx_i$

$s_y^2$  = variance empirique de  $Y$  :

$$\begin{aligned}s_y^2 &= \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2 = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i + e_i - \bar{y})^2 \\ &= \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 + \frac{1}{n} \sum_{i=1}^n (e_i)^2 + 2 \frac{1}{n} \sum_{i=1}^n e_i (\hat{y}_i - \bar{y})\end{aligned}$$

$$\text{Or } \frac{1}{n} \sum_{i=1}^n e_i (\hat{y}_i - \bar{y}) = \text{cov}(e_i, \hat{y}_i) = \text{cov}(e_i, a + bx_i) = b \text{cov}(e_i, x_i) = 0$$

$$\begin{aligned}\text{Donc } s_y^2 &= \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 + \frac{1}{n} \sum_{i=1}^n (e_i)^2 \\ &= \text{variance expliquée} + \text{variance résiduelle}\end{aligned}$$

# Indicateur $R^2$ (1/2)

$s_y^2$  = variance empirique de  $Y$  :

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

= variance expliquée + variance résiduelle

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} = 1 - \frac{\sum_i e_i^2}{\sum_i (y_i - \bar{y})^2} = 1 - \frac{\text{variance résiduelle}}{\text{variance totale}}$$

Le modèle linéaire rend d'autant mieux compte de la liaison entre  $X$  et  $Y$  que  $R^2$  est plus proche de 1

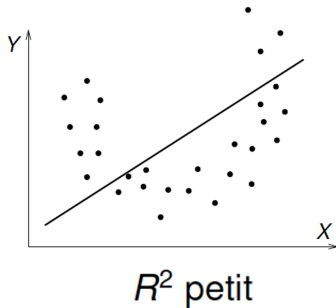
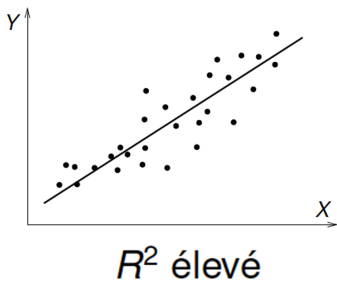
N.B :  $R^2$  et coefficient de corrélation linéaire  $r$

$$r = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y} \Rightarrow R^2 = r^2$$

$$R^2 \in [0, 1], r \in [-1, 1]$$



## l'indicateur $R^2$ (2/2)



- La plupart des données réelles sont multi-dimensionnelles

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ \vdots & & & \\ x_{N1} & x_{N2} & \cdots & x_{Nd} \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}.$$

$x_{ij}$

- $i$  représente un indice d'échantillon
- $j$  un indice de caractéristique.

Notre but : estimer  $E[Y|X_1, X_2, \dots, X_d]$

- L'hypothèse linéaire correspond à :

$$f(\mathbf{x}_i) = \sum_j x_{ij} w_j + b, \quad \mathbf{x}_i \in \mathbb{R}^d$$

- Le problème de minimisation du coût des moindres carrés :

$$\mathbf{w}^*, b^* = \arg \min_{\mathbf{w}, b} \sum_{i=1}^N (f_{\mathbf{w}, b}(\mathbf{x}_i) - y_i)^2$$

- Quand les dimensions augmentent, le modèle linéaire devient complexe

Il est possible d'écrire le problème précédent sous forme matricielle :

- plus simple à écrire + inclusion du biais

$$f(\mathbf{x}_i) = \langle \mathbf{x}_i^\dagger, \mathbf{w}^\dagger \rangle, \quad \text{avec : } \mathbf{x}_i^\dagger = [\mathbf{x}_i, 1] \text{ et } \mathbf{w}^\dagger = [\mathbf{w}, b]$$

- On considère en général  $\mathbf{w}$  comme un vecteur colonne...

$$\mathbf{w}^{\dagger*} = \arg \min_{\mathbf{w}^\dagger} (X^\dagger \mathbf{w}^\dagger - Y)^T (X^\dagger \mathbf{w}^\dagger - Y)$$

- résolution adaptée aux langages de script inaptes aux boucles
- résolution très rapide sur GPU

$$\frac{\partial C}{\partial w_j} = \sum_i 2x_{ij}(f_{\mathbf{w}}(\mathbf{x}_i) - y_i)$$

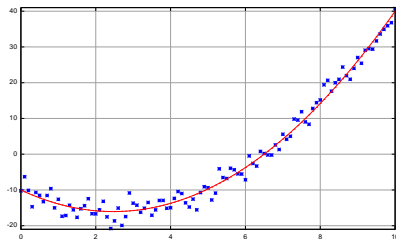
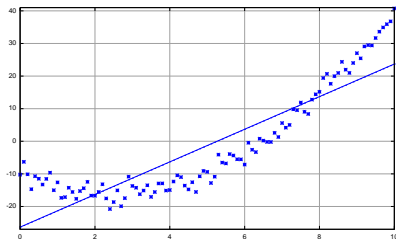
$$\nabla_{\mathbf{w}} C = \begin{bmatrix} \frac{\partial C}{\partial w_1} \\ \vdots \\ \frac{\partial C}{\partial w_d} \end{bmatrix} = 2X^T(X\mathbf{w} - Y) \in \mathbb{R}^d$$

**Résolution :**

$$\nabla_{\mathbf{w}} C = 0 \Leftrightarrow X^T X \mathbf{w} = X^T Y$$

Système d'équations linéaires :  $X^T X \in \mathbb{R}^{d \times d}$ ,  $X^T Y \in \mathbb{R}^{d \times 1}$

# Passage au non-linéaire



Assez trivial : il suffit d'une astuce...

Par exemple, pour un modèle quadratique :

- Concaténation :

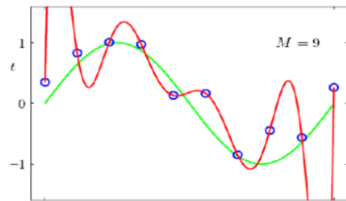
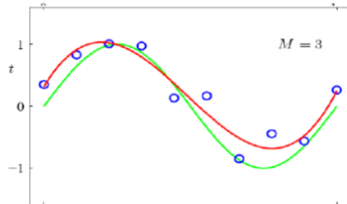
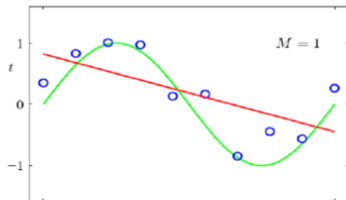
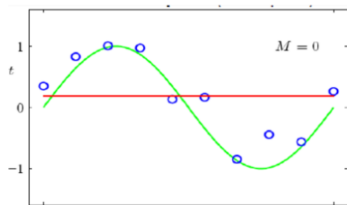
$$X_e = [1, X, X * X]$$

- Puis résolution standard :  $X_e^T X_e \mathbf{w}_e = X_e^T Y$
- Attention à l'inférence sur les nouveaux points et à l'interprétation de  $\mathbf{w}_e$

- **Apprentissage inductif** : bonnes performances sur des données de test  $\neq$  données d'apprentissage  $\Rightarrow$  **Généralisation**
  - **Apprentissage statistique**  $\neq$  optimisation
  - Hypothèse standard  $P_{train}(x, y) = P_{test}(x, y)$
- **Ne pas évaluer les performances sur l'ensemble d'apprentissage !**
- **Expressivité d'un modèle** : capacité à représenter des fonctions complexes ; e.g. nombre de paramètres
  - Plus l'expressivité du modèle augmente, meilleures sont les performances en apprentissage
  - Et les performances en test ?

# Généralisation en régression

- **Ne pas évaluer les performances sur l'ensemble d'apprentissage !**
- **Expressivité d'un modèle** : représenter des fonctions complexes
  - Expressivité  $\uparrow$  : meilleures performances en apprentissage
  - Et les performances en test ?
- Exemple dans le cas de la régression linéaire  $\rightarrow$  polynomiale



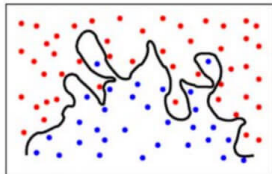
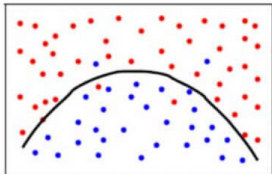
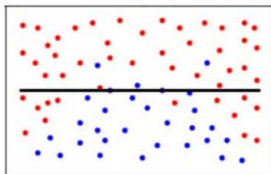


- **Ne pas évaluer les performances sur l'ensemble d'apprentissage !**
- **Expressivité d'un modèle** : représenter des fonctions complexes
  - Expressivité  $\uparrow$  : meilleures performances en apprentissage
  - Et les performances en test ?
- Exemple pour la classification

Underfitting

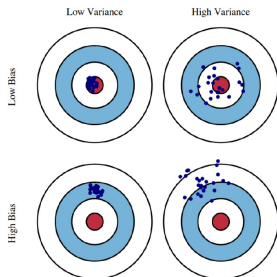


Overfitting



# Dilemne biais-variance

$$\underbrace{E_{\mathbf{x},y,D} \left[ (h_D(\mathbf{x}) - y)^2 \right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D} \left[ (h_D(\mathbf{x}) - \bar{h}(\mathbf{x}))^2 \right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y} \left[ (\bar{y}(\mathbf{x}) - y)^2 \right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}} \left[ (\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}))^2 \right]}_{\text{Bias}^2}$$



- **Expressivité trop faible**

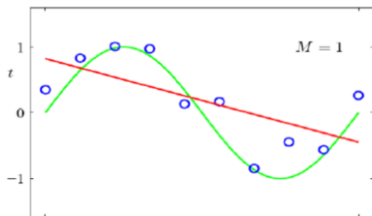
- **Bias fort**
- **Sous-apprentissage**

- **Expressivité trop forte**

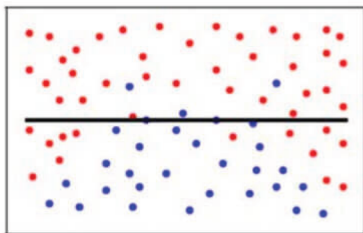
- **Variance forte**, estimateur varie fortement vs l'échantillon d'apprentissage
- **Sur-apprentissage**

- **Modèle de trop faible capacité : incapacité à représenter les observations**

Régression



Classification

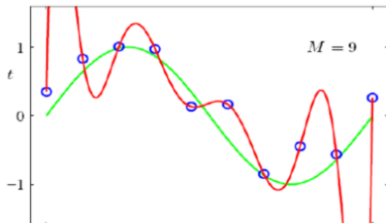


- Chaîne de Markov : nombre d'état trop faible, cf TME 6

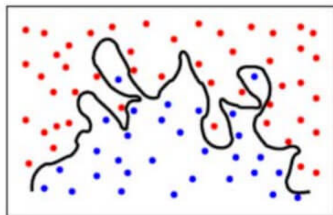
- **Modèle de trop faible capacité : apprentissage par coeur de l'échantillon d'apprentissage  $P_{train}(x, y)$**

- Mais pas de la loi inconnu  $P(x, y)$  !

Régression



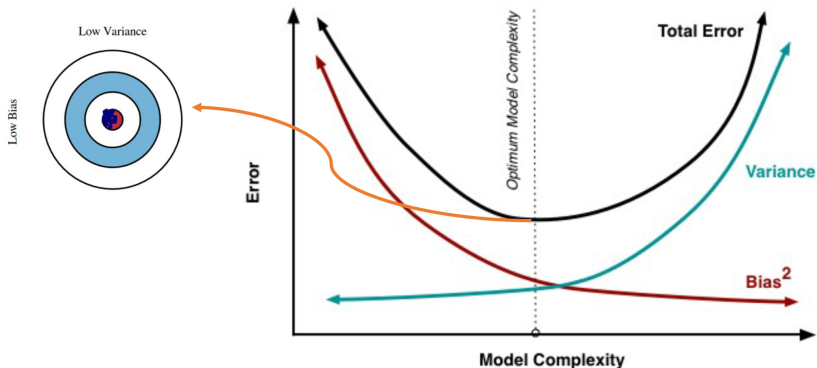
Classification



- **Chaîne de Markov : nombre d'état trop élevé, cf TME 6**

- La matrice de transition contient :  $a_{11} = 0$
- Quelle est la vraisemblance de  $S = \{\dots, q_1, q_1, \dots\}$  ?  
 $\Rightarrow 0$  même si le reste de la séquence ressemble...
- Est-ce le comportement attendu ?  $\Rightarrow$  **Ca dépend !**
- Possibilité : ajouter 1 dans la phase de comptage sur toutes les cases de  $A$  pour que toutes les transitions soient possibles

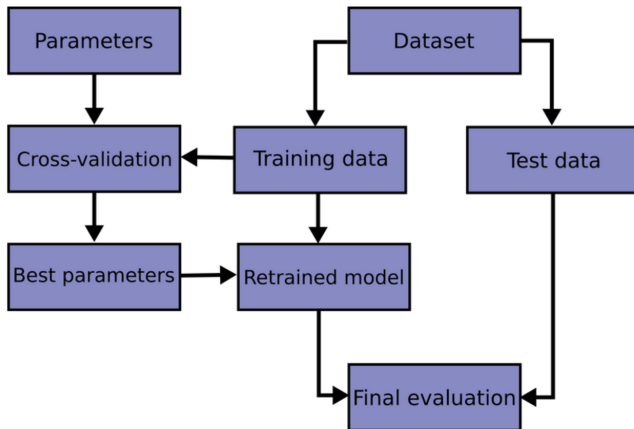
## Comment déterminer le meilleur modèle : sélection de modèle



# Sélection de modèle : validation croisée

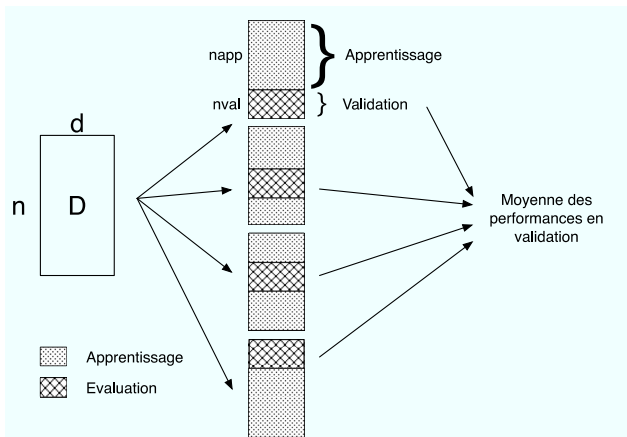
## 3 jeux de données

- **Apprentissage** : optimiser les paramètres du modèles
- **Validation** : sélectionner le meilleur modèle
- **Test** : évaluer les performances finales le meilleur modèle



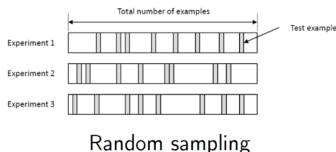
**Après cross-val : ré-entraînement sur l'ensemble du jeu d'apprentissage**

- Apprentissage  $\rightarrow \{Aprentissage', val\}$

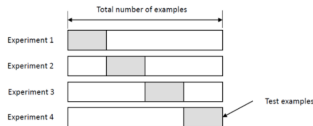


## Comment générer les exemples d'apprentissage et de test ?

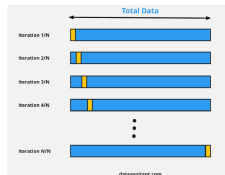
- Cross-validation : partitionnement de train en  $\frac{N}{k}$  sous-ensembles train/val, train  $N - k$ , val  $k$
- Leave-one out :  $k = 1$
- On calcule la moyenne des performances sur les ensembles de val pour sélectionner le modèle
  - Calcul de la variance : permet de mettre en place des tests statistiques
  - $k$  plus petit : variance plus grande



Random sampling



Cross-validation



Leave-one-out



Ce cadre de formalisation est très large et généralisable...

- Données  $\mathbf{x} \in \mathbb{R}^d$ , hypothèse iid : tous les  $\mathbf{x}$  sont indépendants
- **Etiquettes**  $y$  : Classes (discrimination) , Réels (régression)
- **But** : construire une fonction  $f$  telle que  $f(\mathbf{x})$  soit une bonne approximation de  $y$
- **Critères** :
  - Coût  $C$  :

$$\arg \min_{\theta} \sum_{i=1}^N \Delta(f_{\theta}(\mathbf{x}_i), y_i)$$

- Moindres carrés :

$$C = \sum_{i=1}^N \Delta(f_{\theta}(\mathbf{x}_i), y_i) = \sum_{i=1}^N (f_{\theta}(\mathbf{x}_i) - y_i)^2$$

- Coût charnière (codage  $y = \{+1, -1\}$ )

$$C = \sum_{i=1}^N \Delta(f_{\theta}(\mathbf{x}_i), y_i) = \sum_{i=1}^N (-y_i f_{\theta}(\mathbf{x}_i))_+$$

Dans le cas des fonctions de coût exotique (cf coût logistique), il manque parfois une solution analytique

## Algorithme itératif :

- 1 Initialiser  $\mathbf{w}_0$
- 2 En boucle (avec mise à jour du gradient) :

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \epsilon \nabla_{\mathbf{w}} C$$

A condition de choisir  $\epsilon$  suffisamment petit et de faire suffisamment d'itération, nous trouvons  $\mathbf{w}^*$

Le calcul de  $\nabla_{\mathbf{w}} C$  est coûteux... Il est possible de décomposer le problème :

$$C = \sum_{i=1}^N C_i, \quad C_i = (\mathbf{x}_i \mathbf{w} - y_i)^2$$

Algorithme stochastique (Cas MC : ADALINE) :

- ❶ Initialiser  $\mathbf{w}_0$
- ❷ En boucle (avec mise à jour du gradient) :
  - Tirage aléatoire d'un échantillon  $i$
  - Calcul de  $\nabla_{\mathbf{w}} C_i$  (cas MC :  $\nabla_{\mathbf{w}} C_i = 2\mathbf{x}_i^T (\mathbf{x}_i \mathbf{w} - y_i)$ )
  - MAJ :  $\mathbf{w}^{t+1} = \mathbf{w}^t - \epsilon \nabla_{\mathbf{w}} C_i$

## Perceptron

Algorithme de classification binaire des années 60 : toujours très efficace aujourd'hui

$$C = \sum_{i=1}^N (-y_i \mathbf{x}_i \mathbf{w})_+$$

Algorithme stochastique (Cas charnière : Perceptron) :

- ❶ Initialiser  $\mathbf{w}_0$
- ❷ En boucle (avec mise à jour du gradient) :
  - Tirage aléatoire d'un échantillon  $i$
  - Si  $y_i \mathbf{x}_i \mathbf{w} \leq 0$ 
    - Calcul de  $\nabla_{\mathbf{w}} C_i = -y_i \mathbf{x}_i^T$
    - MAJ :  $\mathbf{w}^{t+1} = \mathbf{w}^t - \epsilon \nabla_{\mathbf{w}} C_i$