

# Méthodes non-linéaires : Polynômes, modèles par morceaux, splines et GAM

**Nicolas Verzelen, Joseph Salmon**

INRA / Université de Montpellier



# Méthodes non-linéaires

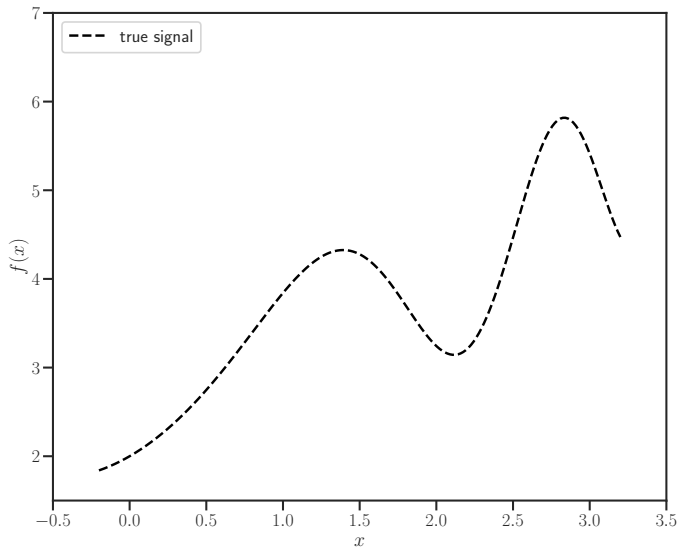
En pratique : modèle linéaire pas toujours bien adapté, mais souvent l'hypothèse linéaire est raisonnable

Alternatives :

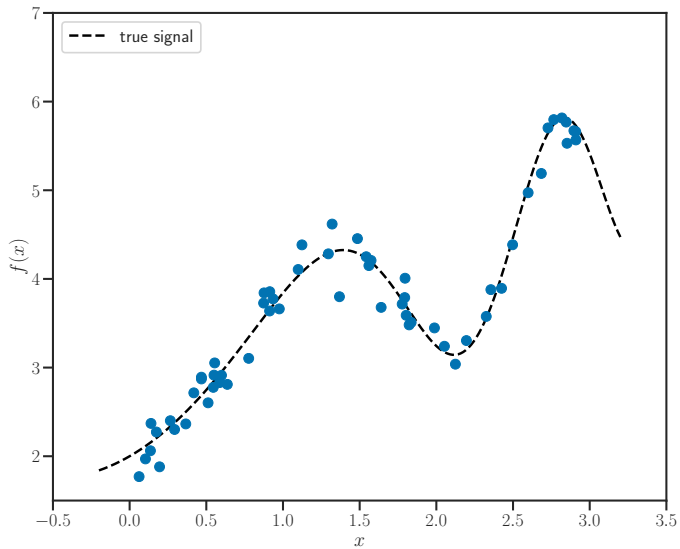
- ▶ polynômes (paramétrique)
- ▶ fonctions en escalier (paramétrique)
- ▶ splines (paramétrique)
- ▶ régression locale (non-paramétrique)
- ▶ modèles additifs généralisés (non-paramétrique)

Flexibilité et interprétabilité !

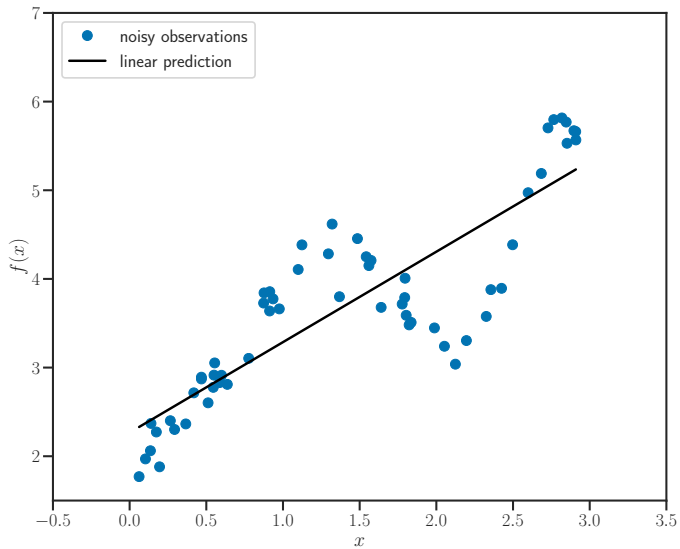
# Limites du modèle linéaire



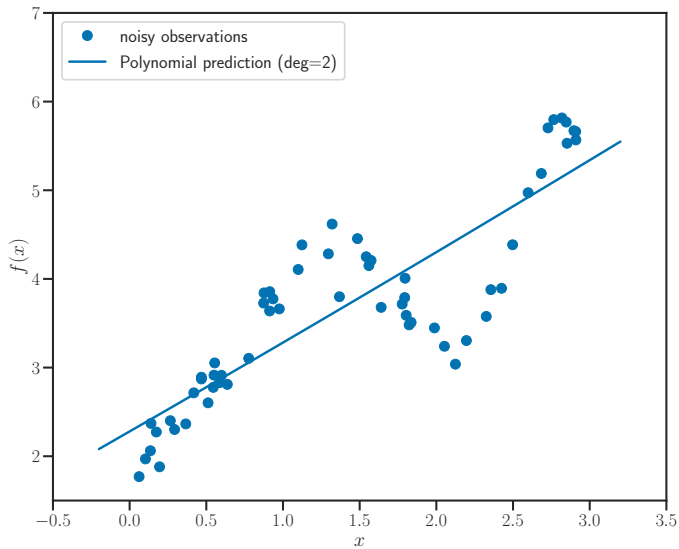
# Limites du modèle linéaire



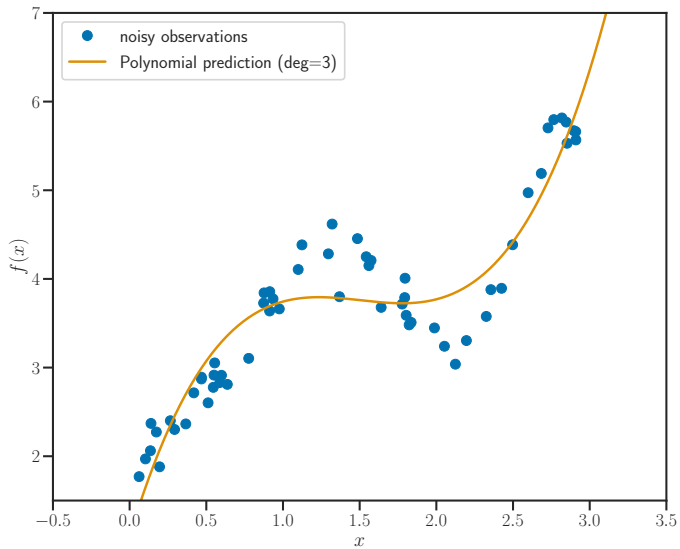
# Limites du modèle linéaire



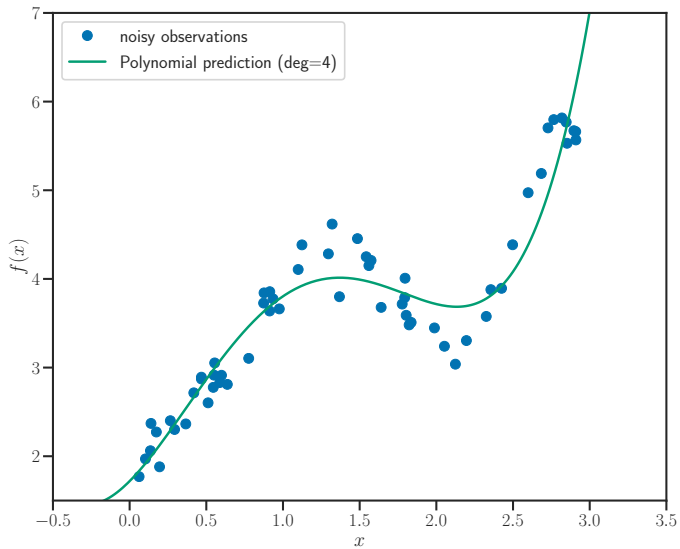
# Limites du modèle linéaire



# Limites du modèle linéaire

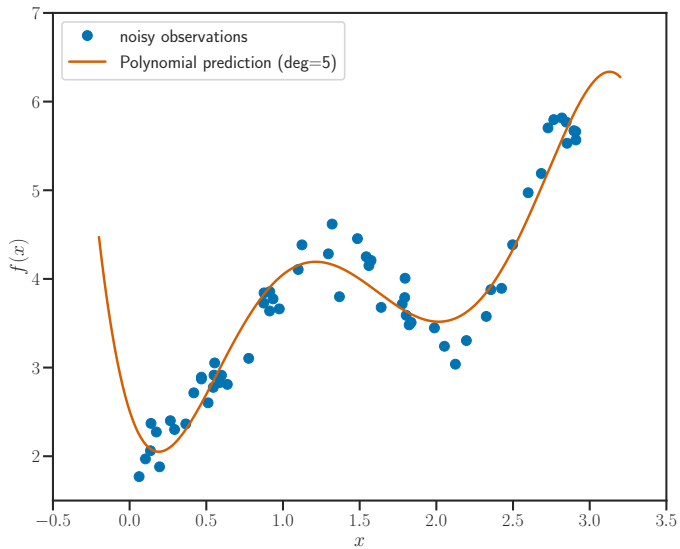


# Limites du modèle linéaire

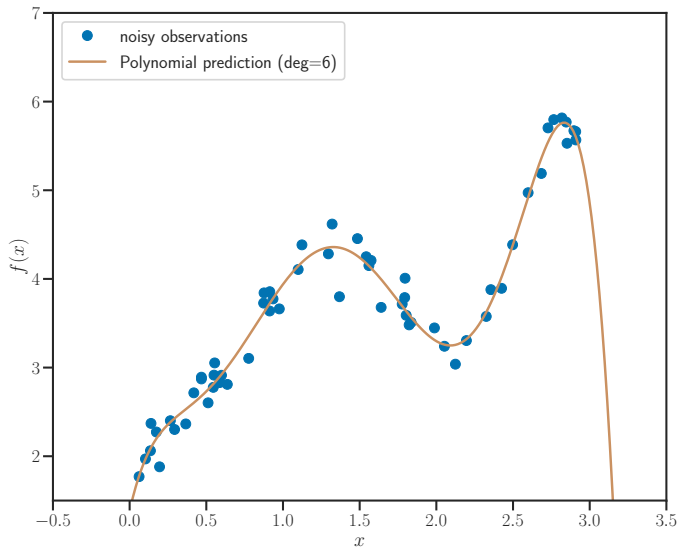




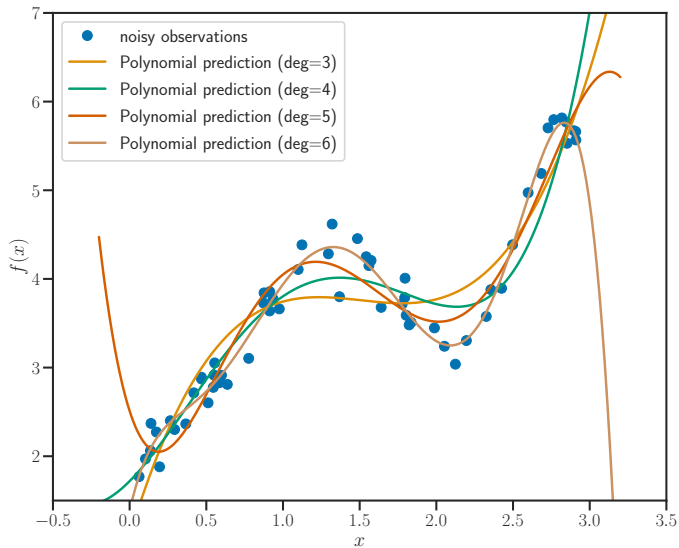
# Limites du modèle linéaire



# Limites du modèle linéaire



# Limites du modèle linéaire



# Modèle polynomial

Observations :  $y_1, \dots, y_n$

Variables explicatives :  $x_1, \dots, x_n$

Modèle polynomiale (degré  $D$ ) :  $y_i \approx \beta_0^* + \sum_{d=1}^D \beta_d^* x_i^d$

Modèle matriciel :  $\mathbf{y} = X\boldsymbol{\beta}^* + \boldsymbol{\varepsilon}$

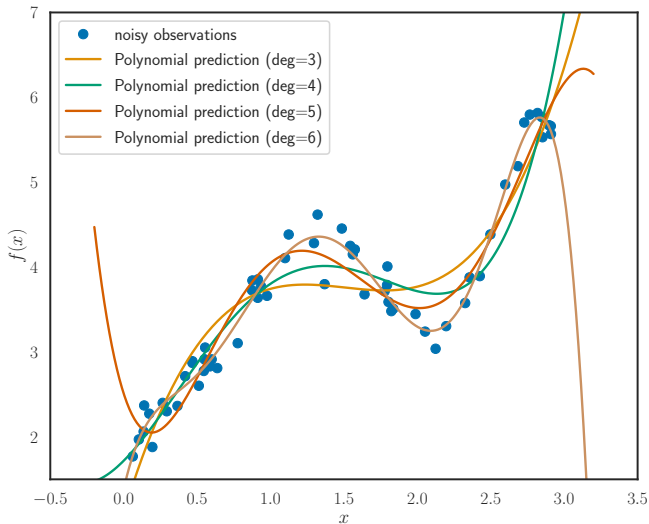
avec  $X = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^D \\ 1 & x_2 & x_2^2 & \dots & x_2^D \\ 1 & x_3 & x_3^2 & \dots & x_3^D \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^D \end{pmatrix}$  : matrice de **Vandermonde**  
(cf. np.vander)

où  $X_{i,j} = x_i^{j-1}$  et  $\boldsymbol{\beta}^* = (\beta_0^*, \dots, \beta_D^*)^\top \in \mathbb{R}^{D+1}$

Rem:  $D + 1$  paramètres pour un polynôme de degré  $D$

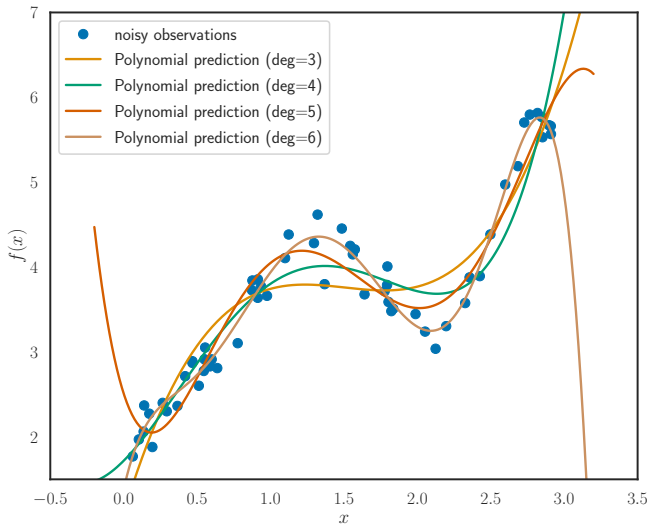
# Choix du degré

On peut utiliser la Validation Croisée (CV) pour choisir le degré



# Choix du degré

On peut utiliser la Validation Croisée (CV) pour choisir le degré

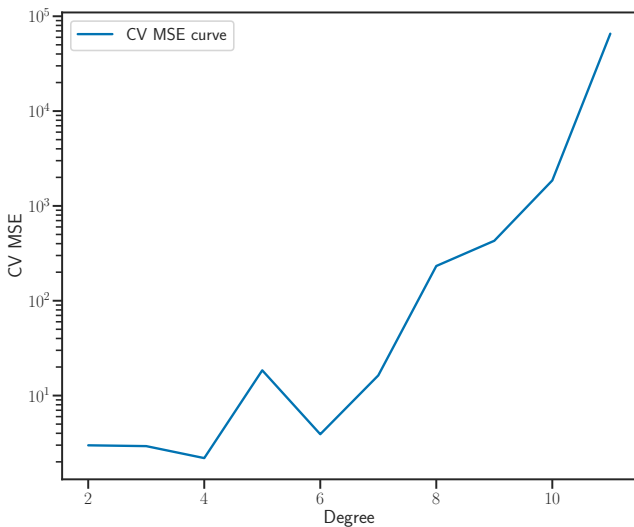


# Régression polynomiale : en pratique

- ▶ créer des variables à partir de  $x$  :  $x^2, x^3, \dots, x^D$
- ▶ utiliser une méthode linéaire (e.g., OLS dans `sklearn`)
- ▶ Incertitude : estimation de la variance / intervalles de confiance identiques au cas du modèle linéaire
- ▶ choix de paramètre par validation croisée

# Choix du degré

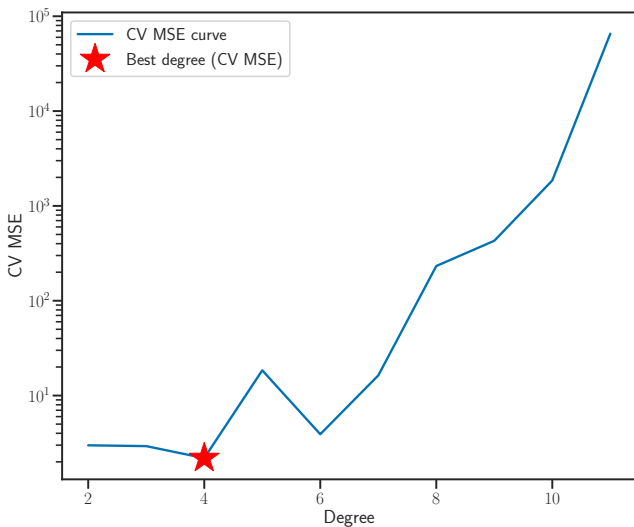
On peut utiliser la Validation Croisée (CV) pour choisir le degré





# Choix du degré

On peut utiliser la Validation Croisée (CV) pour choisir le degré



# Régression polynomiale : + et -

---

---

## Avantages

---

---

- ▶ flexibilité pour de faible degrés
  - ▶ utile en estimation non-paramétrique <sup>(1)</sup> <sup>(2)</sup>
- 
- 

---

---

## Inconvénients

---

---

- ▶ les polynômes sont des fonctions globales (non localisées)
  - ▶ effet de bord et oscillations (mauvaise prédiction en dehors de la gamme de valeurs observées)
  - ▶ le nombre de paramètres à estimer augmente vite avec le degré (et la dimension)
- 
- 

- (1). P. J. GREEN et B. W. SILVERMAN. *Nonparametric regression and generalized linear models*. T. 58. Monographs on Statistics and Applied Probability. A roughness penalty approach. London : Chapman & Hall, 1994, p. xii+182.
- (2). J. FAN et I. GIJBELS. *Local polynomial modelling and its applications*. T. 66. Monographs on Statistics and Applied Probability. London : Chapman & Hall, 1996, p. xvi+341.

## Plusieurs covariables : $p = 2$ et $D = 2$

Considérons le cas  $x_i \in \mathbb{R}^2$  :  $x_i = [a_i, b_i]$ .

Un polynôme d'ordre 2 requiert de fixer :

$$[1, a_i, b_i, a_i^2, a_i b_i, b_i^2]$$

Termes  $a_i b_i$  : **interactions** entre première et seconde variable

Modélisation compacte :

$$y_i = \theta_0 + \theta^\top x_i + \frac{1}{2} x_i^\top \Theta x_i + \varepsilon_i$$

$$y_i = \theta_0 + \sum_{j=1}^p \theta_j x_{i,j} + \frac{1}{2} \sum_{1 \leq j \leq k \leq p} \Theta_{j,k} x_{i,j} x_{i,k} + \varepsilon_i$$

où  $\Theta$  est une matrice (symétrique)  $p \times p$

## Plusieurs covariables : $p = 2$ et $D = 3$

Considérons le cas  $x_i \in \mathbb{R}^2$  :  $x_i = [a_i, b_i]$ .

Un polynôme d'ordre 3 requiert de fixer :

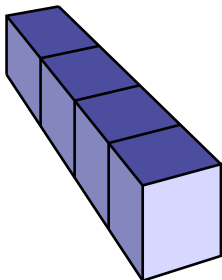
$$[1, a_i, b_i, a_i^2, a_i b_i, b_i^2, a_i^3, a_i^2 b_i, a_i b_i^2, b_i^3]$$

Modélisation compacte :

$$y_i = \theta_0 + \sum_{j=1}^p \theta_j x_{i,j} + \frac{1}{2} \sum_{1 \leq j \leq k \leq p} \Theta_{j,k} \cdot x_{i,j} x_{i,k} \\ + \frac{1}{6} \sum_{1 \leq j \leq k \leq \ell \leq p} \Theta_{j,k,\ell} \cdot x_{i,j} x_{i,k} x_{i,\ell} + \varepsilon_i$$

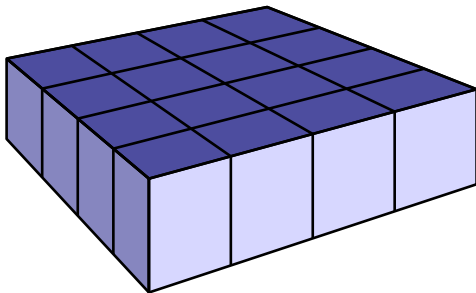
où  $\Theta$  est une matrice (symétrique)  $p \times p$ , et  $\Theta$  est un tenseur (symétrique)  $p \times p \times p$

# Représentation de tenseur 1D



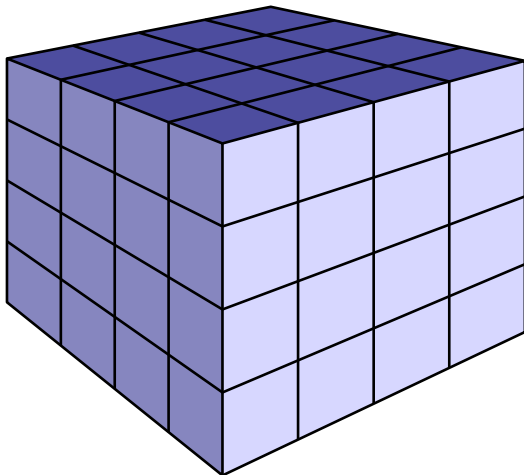
Cas vectoriel

# Représentation de tenseur 2D



Cas matriciel

# Représentation de tenseur 3D



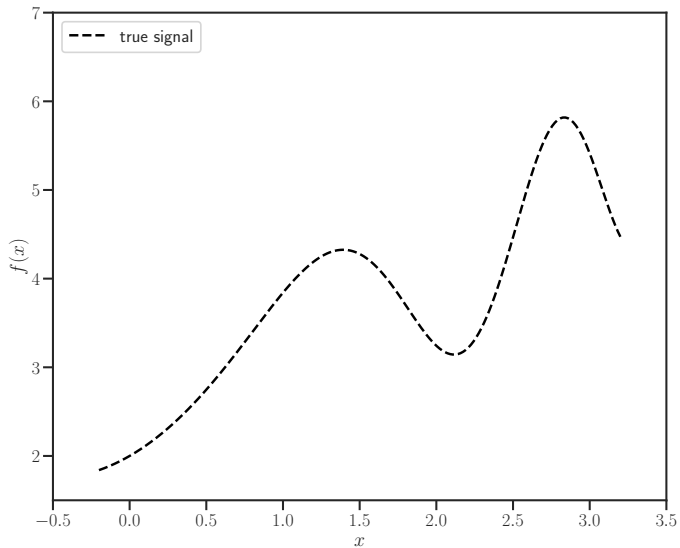
Cas tensoriel

## Autre familles de fonctions utilisée

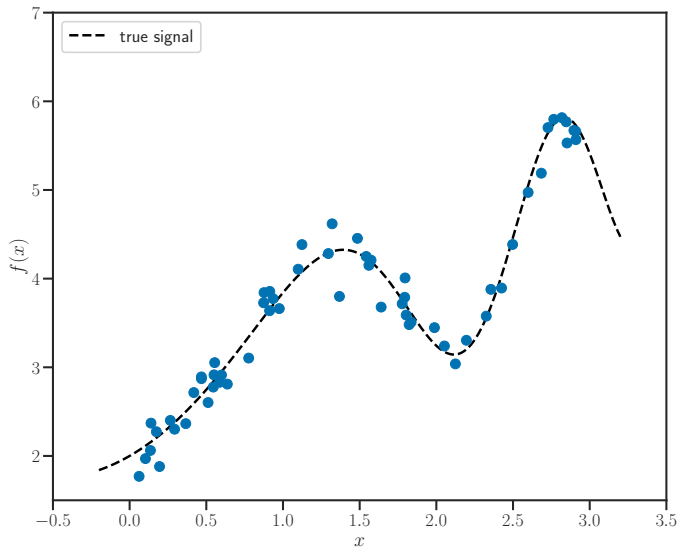
- ▶ Polynômes, dictionnaire de fonction :  $x^1, x^2, \dots, x^D$   
*cf. sklearn.preprocessing.PolynomialFeatures*
- ▶ fonctions en escalier :  $\mathbb{1}_{\{x>t_1\}}, \dots, \mathbb{1}_{\{x>t_K\}}$   
*cf. pandas.cut*



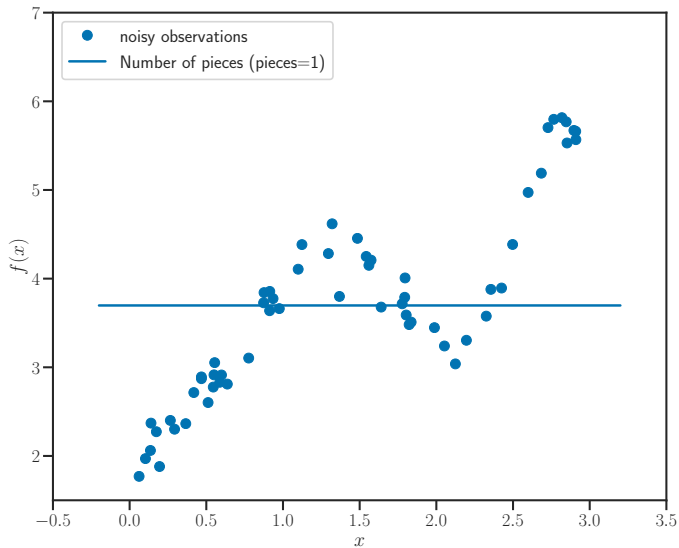
# Fonctions constantes par morceaux



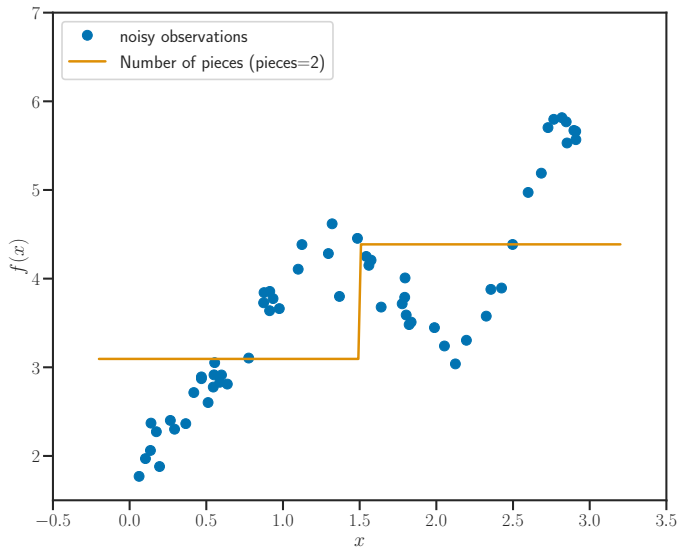
# Fonctions constantes par morceaux



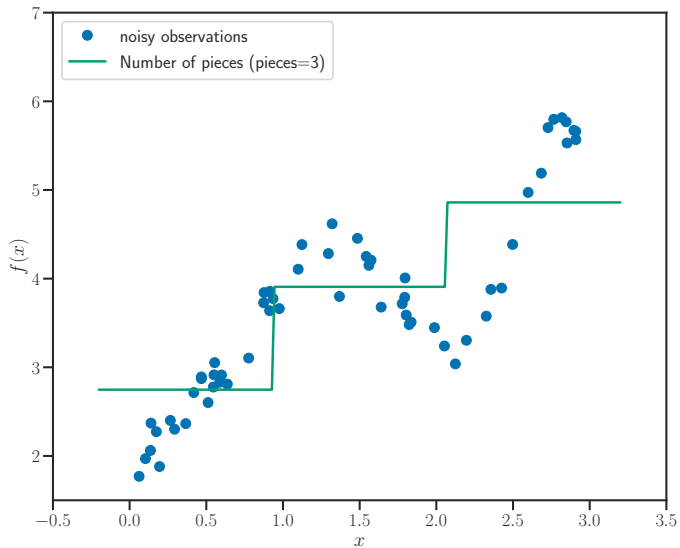
# Fonctions constantes par morceaux



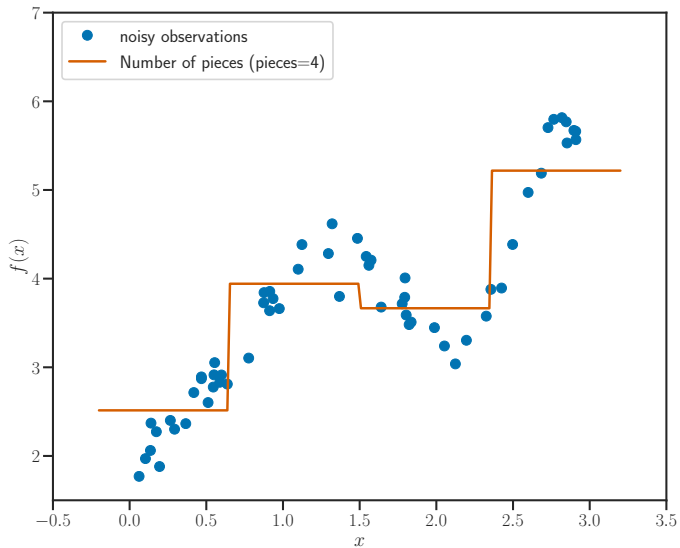
# Fonctions constantes par morceaux



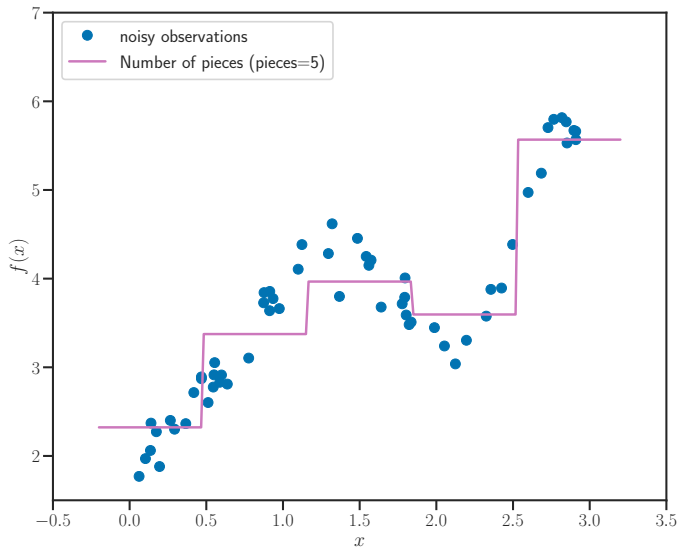
# Fonctions constantes par morceaux



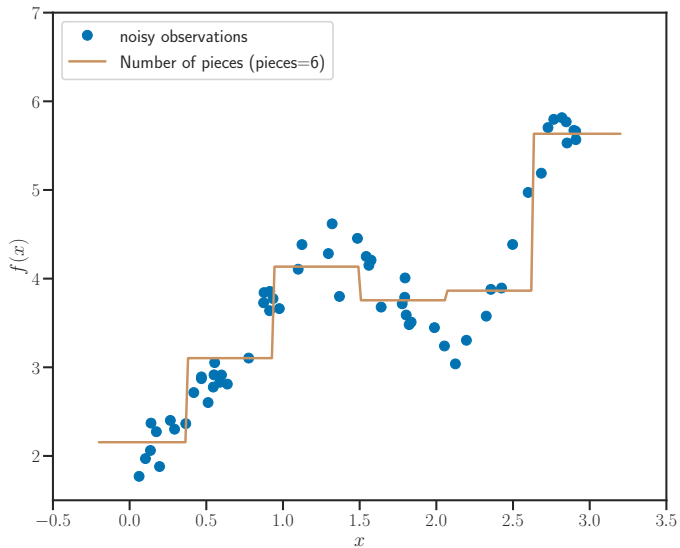
# Fonctions constantes par morceaux



# Fonctions constantes par morceaux

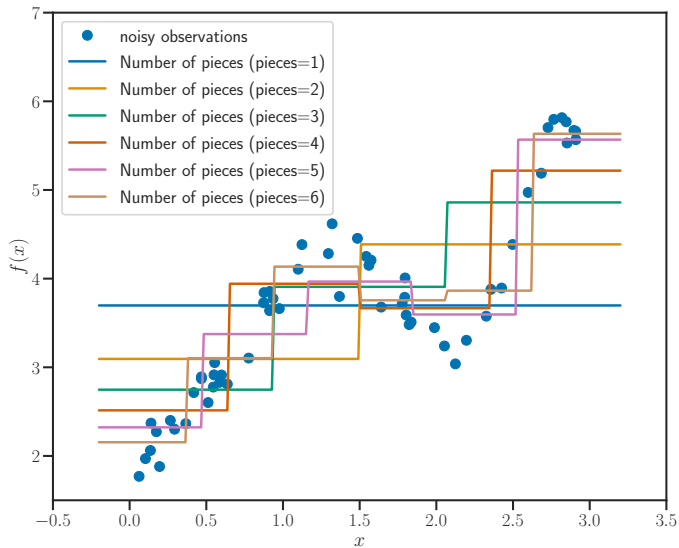


# Fonctions constantes par morceaux

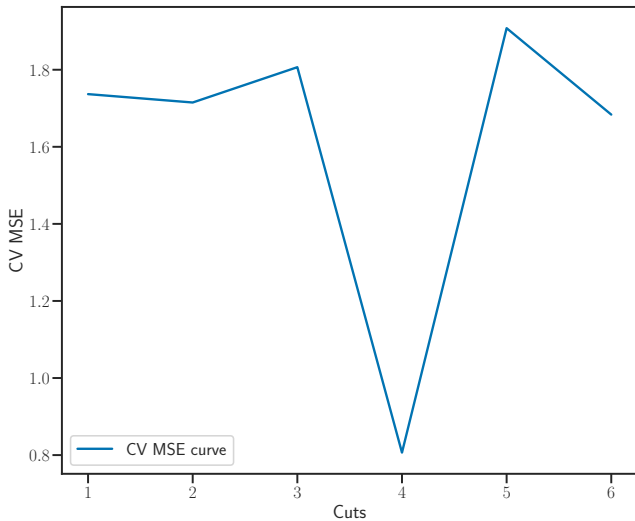




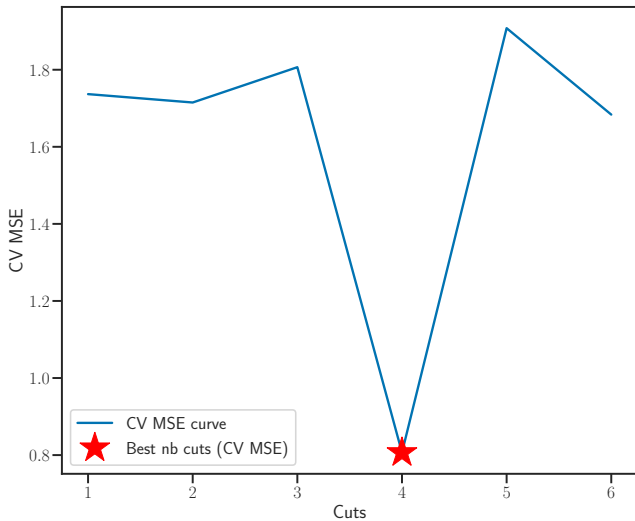
# Fonctions constantes par morceaux



## Choix du nombre de parties



## Choix du nombre de parties



## Fonctions en escalier (suite)

- ▶ Crée une suite de variables binaires représentant chaque groupe
- ▶ Le choix des points de rupture ou des **nœuds** peut être problématique.  
Pour créer des non-linéarité, des alternatives plus lisses sont possibles, par exemple des **splines**

# Splines ( : cerces )

## Définition :

Un **spline**  $f$  est une fonction polynomiale par morceaux sur un intervalle  $[a, b]$ ,  $f : [a, b] \rightarrow \mathbb{R}$ , composé de  $n$  sous-intervalles  $[x_{i-1}, x_i]$  avec  $a = x_0 < x_1 < \dots < x_{n-1} < x_n = b$ . La restriction de  $f$  sur chaque intervalle  $[x_{i-1}, x_i]$  est un polynôme

$P_i : [x_{i-1}, x_i] \rightarrow \mathbb{R}$ , ainsi

$$f(x) = P_1(x), \quad x_0 \leq t < x_1$$

$$f(x) = P_2(x), \quad x_1 \leq t < x_2$$

$\vdots$

$$f(x) = P_i(x), \quad x_{n-1} \leq t \leq x_n.$$

Le plus haut degré des polynômes  $P_i$  est appelé l'**ordre** du spline  $f$ , et les  $x_i$  sont appelé les **nœuds** (  : *knots* )

Rem: les plus populaires sont les splines (cubiques) d'ordre 3

Rem: on privilégie des splines lisses :  $C^0, C^1, C^2$ , etc.

# Utilisation

- ▶ statistiques
- ▶ computer vision, *cf.* courbes de Bézier dans [Inkscape](#) et autres logiciels de dessin vectoriel

# Utilisation

- ▶ statistiques
- ▶ computer vision, *cf.* courbes de Bézier dans [Inkscape](#) et autres logiciels de dessin vectoriel
- ▶ analyse numérique

# Utilisation

- ▶ statistiques
- ▶ computer vision, *cf.* courbes de Bézier dans [Inkscape](#) et autres logiciels de dessin vectoriel
- ▶ analyse numérique
- ▶ etc.



# Utilisation

- ▶ statistiques
- ▶ computer vision, *cf.* courbes de Bézier dans [Inkscape](#) et autres logiciels de dessin vectoriel
- ▶ analyse numérique
- ▶ etc.

# Algorithmes

Approches standards pour ajuster des splines quand on observe des points  $(x_i, y_i)$  pour  $i = 1, \dots, n$  : chercher le spline avec courbure minimum, *i.e.*, résoudre :

$$\hat{f} \triangleq SP_\lambda(\mathbf{y}) \in \arg \min_{f \text{ est un spline}} \left( \sum_{i=1}^n (f(x_i) - y_i)^2 + \lambda \int_a^b |f''(t)|^2 dt \right)$$

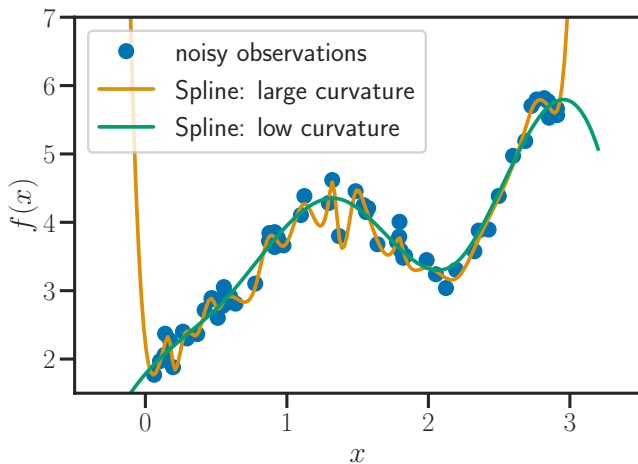
Fait : la solution est atteinte pour un spline cubique, et peut être obtenue par un moindre carré régularisé, avec  $\Omega \in \mathbb{R}^{n \times n}$

$$\arg \min_g \|\mathbf{y} - g\|^2 + \lambda g^\top \Omega g$$

voir détails dans [Ch. 2, Green and Silverman \(1994\)](#)

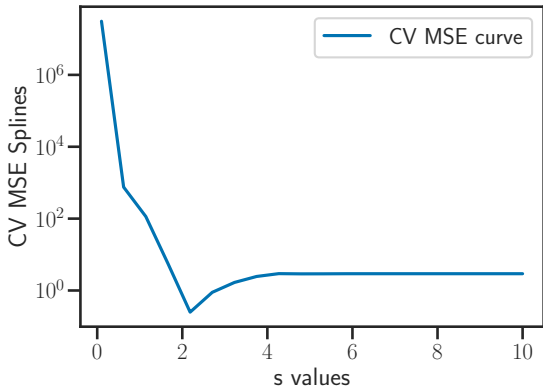
Note : avec cette régularisation les splines ont pour nœuds les  $x_i$

# Visual



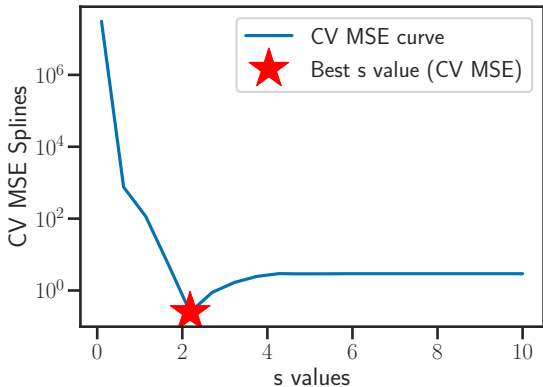
## Choix du paramètre de lissage

On peut utiliser la Validation Croisée (CV) pour choisir le niveau de lissage



## Choix du paramètre de lissage

On peut utiliser la Validation Croisée (CV) pour choisir le niveau de lissage



MSE Spline = 0.2498 vs. MSE Polynomials = 2.1899

# Modèles additifs pour la régression

Avec les des fonctions réelles, *i.e.*,  $f_j : \mathbb{R} \rightarrow \mathbb{R}$ , le modèle s'écrit

$$y_i = \sum_{j=1}^p f_j(x_{i,j}) + \varepsilon_i \quad \text{ou} \quad \mathbf{y} = \sum_{j=1}^p f_j(\mathbf{x}_j) + \varepsilon$$

avec la convention  $f_j(\mathbf{x}_j) = \begin{pmatrix} f_j(x_{1,j}) \\ \vdots \\ f_j(x_{n,j}) \end{pmatrix}$  avec  $\mathbf{x}_j = \begin{pmatrix} x_{1,j} \\ \vdots \\ x_{n,j} \end{pmatrix}$

Rem: potentiellement un des  $f_j$  encode la variable constante

Rem: GAM (Generalized Additive Models) : extension aux modèles linéaires généralisés, *e.g.*, régression logistique,

$g(y_i) = \sum_{j=1}^p f_j(x_{i,j})$ , avec  $g$  une fonction

# Rétro-ajustement (Backfitting)

---

**Algorithm:** Rétro-ajustement d'un modèle additif

---

**Input** :  $X = [\mathbf{x}_1, \dots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$ ,  $\mathbf{y} \in \mathbb{R}^n$

---

# Rétro-ajustement (Backfitting)

---

**Algorithm:** Rétro-ajustement d'un modèle additif

---

**Input** :  $X = [\mathbf{x}_1, \dots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$ ,  $\mathbf{y} \in \mathbb{R}^n$

Initialize :  $f_1 \equiv 0, \dots, f_p \equiv 0$  and  $\mathbf{r} = \mathbf{y}$  (residual)

---



# Rétro-ajustement (Backfitting)

---

**Algorithm:** Rétro-ajustement d'un modèle additif

---

**Input** :  $X = [\mathbf{x}_1, \dots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$ ,  $\mathbf{y} \in \mathbb{R}^n$

Initialize :  $f_1 \equiv 0, \dots, f_p \equiv 0$  and  $\mathbf{r} = \mathbf{y}$  (residual)

**while** *not converged* **do**

|

---

# Rétro-ajustement (Backfitting)

---

**Algorithm:** Rétro-ajustement d'un modèle additif

---

**Input** :  $X = [\mathbf{x}_1, \dots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$ ,  $\mathbf{y} \in \mathbb{R}^n$

Initialize :  $f_1 \equiv 0, \dots, f_p \equiv 0$  and  $\mathbf{r} = \mathbf{y}$  (residual)

**while** *not converged* **do**

**for**  $j = 1, \dots, p$  **do**

        |

    |

# Rétro-ajustement (Backfitting)

---

**Algorithm:** Rétro-ajustement d'un modèle additif

---

**Input** :  $X = [\mathbf{x}_1, \dots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$ ,  $\mathbf{y} \in \mathbb{R}^n$

Initialize :  $f_1 \equiv 0, \dots, f_p \equiv 0$  and  $\mathbf{r} = \mathbf{y}$  (residual)

**while** *not converged* **do**

**for**  $j = 1, \dots, p$  **do**

$\mathbf{r} \leftarrow \mathbf{r} + f_j(\mathbf{x}_j)$

        // Partial residual update

# Rétro-ajustement (Backfitting)

---

**Algorithm:** Rétro-ajustement d'un modèle additif

---

**Input** :  $X = [\mathbf{x}_1, \dots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$ ,  $\mathbf{y} \in \mathbb{R}^n$

Initialize :  $f_1 \equiv 0, \dots, f_p \equiv 0$  and  $\mathbf{r} = \mathbf{y}$  (residual)

**while** *not converged* **do**

**for**  $j = 1, \dots, p$  **do**

$\mathbf{r} \leftarrow \mathbf{r} + f_j(\mathbf{x}_j)$

        // Partial residual update

$f_j \leftarrow SP_{\lambda_j}(\mathbf{r})$

        // update with spline (param.  $\lambda_j$ )



# Rétro-ajustement (Backfitting)

---

**Algorithm:** Rétro-ajustement d'un modèle additif

---

**Input** :  $X = [\mathbf{x}_1, \dots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$ ,  $\mathbf{y} \in \mathbb{R}^n$

Initialize :  $f_1 \equiv 0, \dots, f_p \equiv 0$  and  $\mathbf{r} = \mathbf{y}$  (residual)

**while** *not converged* **do**

**for**  $j = 1, \dots, p$  **do**

$\mathbf{r} \leftarrow \mathbf{r} + f_j(\mathbf{x}_j)$

  // Partial residual update

$f_j \leftarrow SP_{\lambda_j}(\mathbf{r})$

  // update with spline (param.  $\lambda_j$ )

$\mathbf{r} \leftarrow \mathbf{r} - f_j(\mathbf{x}_j)$

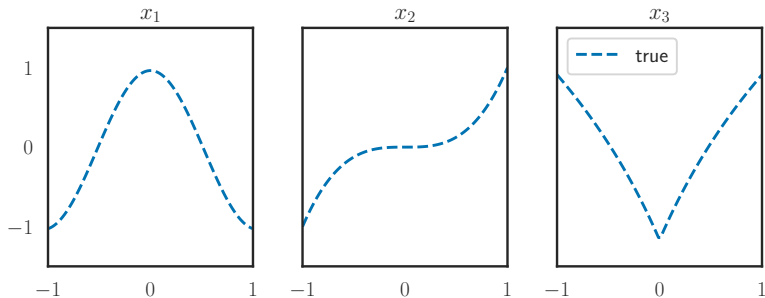
  // Partial residual un-update

**Output:**  $f_1, \dots, f_p$

---

Rem: Rétro-ajustement = descente par coordonnée

# GAM en action



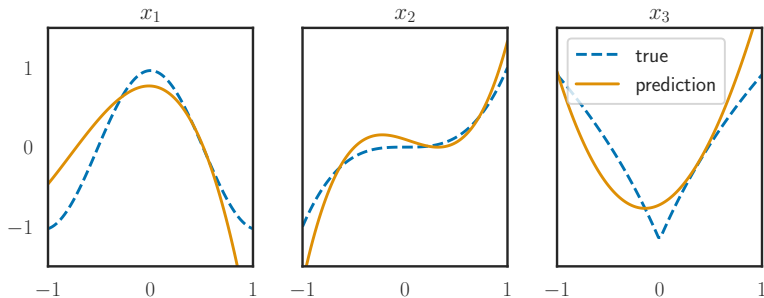
où  $y = f(\mathbf{x}) + \varepsilon$  avec  $f(\mathbf{x}) = f_1(x_1) + f_2(x_2) + f_3(x_3)$  et

$$f_1(x) = \cos(3x)$$

$$f_2(x) = x^3$$

$$f_3(x) = 3 \log(1 + |x|)$$

# GAM en action



où  $y = f(\mathbf{x}) + \varepsilon$  avec  $f(\mathbf{x}) = f_1(x_1) + f_2(x_2) + f_3(x_3)$  et

$$f_1(x) = \cos(3x)$$

$$f_2(x) = x^3$$

$$f_3(x) = 3 \log(1 + |x|)$$



# Avantages et inconvénients des GAM

---

---

## Avantages

- ▶ peut modéliser des effets non-linéaire automatiquement
- ▶ interprétation possible : fonctions 1D (visualisables!)
- ▶ peut s'étendre au cas d'interactions d'ordre plus élevées ( $p$  petit)

---

---

## Inconvénients

- ▶ Critère d'arrêt pas si simple (non-convexe)
  - ▶ Calibrage difficile : au moins un paramètre par variable
- 
-

## Plus de détails sur les GAM :

- ▶ Vidéo : <https://vimeo.com/125940125>
- ▶ Livre : Hastie and Tibshirani (1990)

## Plus de détails sur les GAM :

- ▶ Vidéo : <https://vimeo.com/125940125>
- ▶ Livre : Hastie and Tibshirani (1990)

# References I

- ▶ FAN, J. et I. GIJBELS. *Local polynomial modelling and its applications*. T. 66. Monographs on Statistics and Applied Probability. London : Chapman & Hall, 1996, p. xvi+341.
- ▶ GREEN, P. J. et B. W. SILVERMAN. *Nonparametric regression and generalized linear models*. T. 58. Monographs on Statistics and Applied Probability. A roughness penalty approach. London : Chapman & Hall, 1994, p. xii+182.
- ▶ HASTIE, T. J. et R. J. TIBSHIRANI. *Generalized additive models*. T. 43. CRC press, 1990.