Apprentissage Statistique

Régression

Guillaume Wisniewski
guillaume.wisniewski@limsi.fr

d'après la présentation de L. Bottou

Université Paris Sud — LIMSI

janvier 2016

Problématique

Question
Trouver y en fonction de x

Réponse

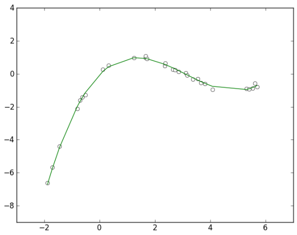
x	y
0.78	0.70
-1.99	-0.41
3.62	-0.88
1.12	0.43
2.78	-0.93
1.45	0.11
0.23	0.97
2.48	-0.79
⋮	⋮

Problématique

Question
Trouver y en fonction de x

x	y
0.78	0.70
-1.99	-0.41
3.62	-0.88
1.12	0.43
2.78	-0.93
1.45	0.11
0.23	0.97
2.48	-0.79
⋮	⋮

Réponse
Un peu de logique...



Il suffit de relier les points...

Généralisation : la tâche de régression

Régression

- $\mathbf{x} \in \mathbb{R}^n$ = vecteur de caractéristiques décrivant un exemple
- $y \in \mathbb{R}$ = réponse / étiquette
- Trouver f tel que $\forall \mathbf{x}, f(\mathbf{x}) = y$

Exemple : cancer de la prostate

- exemple = un homme représenté par : le poids de la prostate, l'âge, la quantité de *benign prostatic hyperplasia*, le *Gleason score*, ...
- extraction manuelle des caractéristiques pertinentes
- étiquette : volume du cancer

$\Rightarrow f$ ne peut pas être déterminée manuellement

Solution : apprentissage automatique

Déterminer f à partir des données $\Rightarrow$ paradigme de l'apprentissage statistique

1. (extraction des caractéristiques)
2. choix d'une classe de fonction
3. choix d'un critère d'apprentissage
4. déterminer f
5. estimer les performances de f

Modèle linéaire

1re étape : choix d'une classe de fonctions

- pour le moment : fonctions linéaires :

$$f_w : \mathbb{R}^n \mapsto \mathbb{R}$$
$$\mathbf{x} \rightarrow \sum_{i=1}^{n} x_i \times w_i + w_0 = \mathbf{x}^\top \mathbf{w} + w_0$$

- fonctions paramétrées par $(\mathbf{w} \in \mathbb{R}^n, w_0 \in \mathbb{R})$
- classe de fonction : ensemble des fonctions que l'on obtient en considérant tous les paramètres
- objectif : trouver la « meilleure » valeur du paramètre

Application du modèle linéaire (1)

Prédiction

- on suppose que le régresseur a pour paramètres $\mathbf{w} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$
  et $w_0 = 4$
- On considère les exemples $\mathbf{x}^1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ et $\mathbf{x}^2 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$.
  Quelle est la valeur prédite ?
- le terme w_0 est appelé biais. Pourquoi ?
- On décrit souvent un régresseur linéaire comme une somme pondérée. Pourquoi ?

Application du modèle linéaire (2)

Représentation

- On considère les paires (exemple, étiquette) suivante : $(1, 1.1), (2, 1.9), (2.9, 3.2)$.
- Représentez les points et la meilleure régression linéaire correspondante.

Moindres carrés (régression linéaire)

- en entrée : $\mathbf{x}^{(i)} \in \mathbb{R}^n$ = un exemple
- sortie : $w^\top \mathbf{x}^{(i)}$ = prédiction
- sortie attendue : $y^{(i)}$

Principe : l'erreur peut être mesurée par :

$$y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)}$$

Critère d'apprentissage :

$$\min_w \underbrace{\sum_{i=1}^{n} \left(y^{(i)} - w^\top x^{(i)}\right)^2}_{C(w)}$$

Résolution (1)

À l'optimum :

$$\frac{\mathrm{d}C}{\mathrm{d}w} = \sum_{i=1}^{n} 2 \times \left(y^{(i)} - w^\top x^{(i)}\right)^\top x^{(i)} = 0$$

soit :

$$\left[\sum_{i=1}^{n} {x^{(i)}}^\top x^{(i)}\right] \times w = \sum_{i=1}^{n} y^{(i)} \cdot x^{(i)}$$

ou de manière compacte :

$$\left(X^\top X\right) w = \left(X^\top Y\right)$$

où X = matrice résultant de la concaténation des vecteurs $x^{(i)}$

Résolution (2)

En première approximation :

$$w = \left(X^\top X\right)^{-1} \left(X^\top Y\right)$$

- problème dès que $X^\top X$ n'est pas inversible
- problème d'efficacité
- en pratique : fonctions qui le font correctement (en python : `numpy.linalg.lstsq`)

Notation matricielle I

- Prédiciton de la valeur associée à $x^{(i)}$:

$$\begin{aligned} \hat{y^{(i)}} &= {x^{(i)}}^\top w + w_0 \\ &= x_1^{(i)} \cdot w_1 + x_2^{(i)} \cdot w_2 + \cdots + x_n^{(i)} \cdot w_n + w_0 \end{aligned}$$

- En considérant tout le corpus (N examples) :

$$\begin{aligned} \hat{y^{(1)}} &= x_1^{(1)} \cdot w_1 + x_2^{(1)} \cdot w_2 + \cdots + x_n^{(1)} \cdot w_n + w_0 \\ \hat{y^{(2)}} &= x_1^{(2)} \cdot w_1 + x_2^{(2)} \cdot w_2 + \cdots + x_n^{(2)} \cdot w_n + w_0 \\ &\vdots \\ \hat{y^{(N)}} &= x_1^{(N)} \cdot w_1 + x_2^{(N)} \cdot w_2 + \cdots + x_n^{(N)} \cdot w_n + w_0 \end{aligned}$$

Notation matricielle II

- en passant aux matrices :

$$\underbrace{\begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(N)} \end{bmatrix}}_{N\times 1} = \underbrace{\begin{bmatrix} x_1^{(1)}\cdot w_1 + x_2^{(1)}\cdot w_2 + \cdots + x_n^{(1)}\cdot w_n \\ x_1^{(2)}\cdot w_1 + x_2^{(2)}\cdot w_2 + \cdots + x_n^{(2)}\cdot w_n \\ \vdots \\ x_1^{(N)}\cdot w_1 + x_2^{(N)}\cdot w_2 + \cdots + x_n^{(N)}\cdot w_n \end{bmatrix}}_{N\times 1} + \underbrace{\begin{bmatrix} w_0 \\ w_0 \\ \vdots \\ w_0 \end{bmatrix}}_{N\times 1}$$

- 1^re^ astuce : multiplication de matrices

$$\begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(N)} \end{bmatrix} = \underbrace{\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & & & \\ x_1^{(N)} & x_2^{(N)} & \cdots & x_n^{(N)} \end{bmatrix}}_{N\times n} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}}_{n\times 1} + \begin{bmatrix} w_0 \\ w_0 \\ \vdots \\ w_0 \end{bmatrix}$$

Notation matricielle III

- 2^e^ astuce : le « coup du 1 »

$$\begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(N)} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & & & & \\ 1 & x_1^{(N)} & x_2^{(N)} & \cdots & x_n^{(N)} \end{bmatrix}}_{N\times(n+1)} \underbrace{\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}}_{(n+1)\times 1}$$

En pratique, le biais est rarement distingué et l'on considère toujours qu'il y a une caractéristique « constante »

Exemple n° 1

Génération de données artificielles

```
import numpy as np
from numpy.random import uniform, normal, seed

seed(1024)

p = np.poly1d([0.087, -0.81, 1.69, -0.03])

n = 50                  # number of points
t_low, t_high = -2, 6   # domain
noise_scale = 1

# sort x to be sure the points can be plotted
x = sorted(uniform(low=t_low, high=t_high, size=n))
x = np.array(x)
y = p(x) + normal(loc=0, scale=noise_scale, size=n)
```

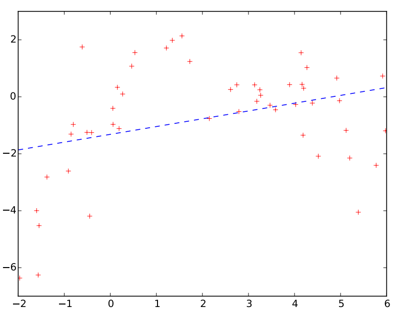
Régression linéaire

```
X = np.vstack([np.ones(n), x])

m, c = np.linalg.lstsq(X.T, y)[0]

plt.plot(x, m * x + c, "r+")
plt.plot(x, p(x), "-x")
```

Résultat



Régression polynomiale

- $x^{(i)} \in \mathbb{R}^n$ = caractéristiques décrivants le i^e^ exemple
- ...les caractéristiques peuvent être décrites à la main
- ...ou automatiquement : $x_2 = x_1^2$, $x_3 = x_1^3$, ...

$\Rightarrow$ expansion de la base

- on peut considérer des expansions de degré arbitraire
- on parle alors de régression polynomiale. Pourquoi ?

En pratique

$$V = \begin{pmatrix} 1 & \alpha_1 & {\alpha_1}^2 & \dots & {\alpha_1}^{n-1} \\ 1 & \alpha_2 & {\alpha_2}^2 & \dots & {\alpha_2}^{n-1} \\ 1 & \alpha_3 & {\alpha_3}^2 & \dots & {\alpha_3}^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha_m & {\alpha_m}^2 & \dots & {\alpha_m}^{n-1} \end{pmatrix}$$

Matrice de Vandermonde (`np.vander`)

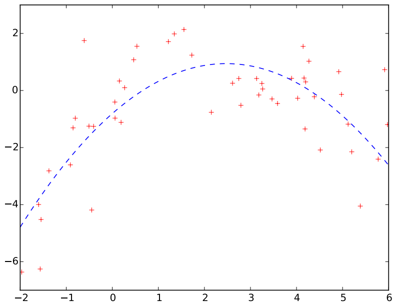
Plus directement

```
inter_poly = np.poly1d(np.polyfit(x, y, i))
xp = np.linspace(-2, 6, 100)
plt.plot(x, y, "r+", xp, inter_poly(xp), "--")
```

- préférable d'utiliser `polyfit` → évite des problèmes d'instabilité numérique

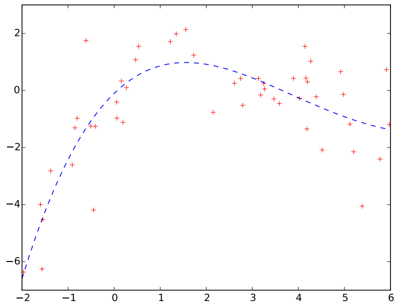
Exemple n° 2 : régression polynomiale

n = 2



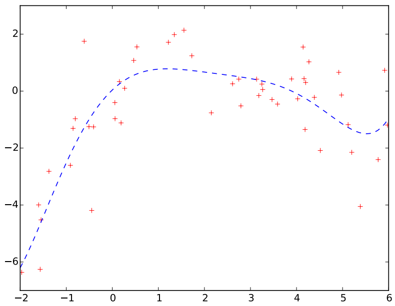
Exemple n° 2 : régression polynomiale

n = 4



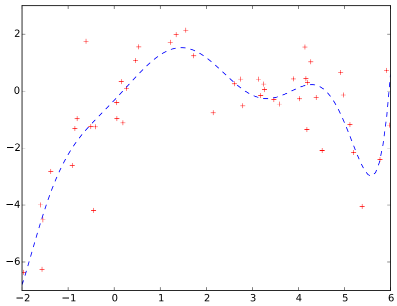
Exemple n° 2 : régression polynomiale

n = 6



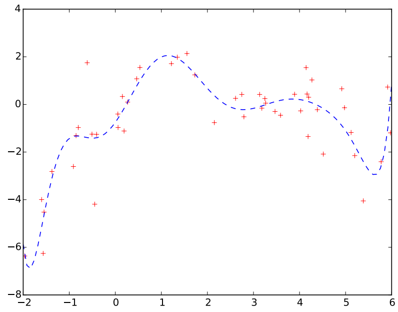
Exemple n° 2 : régression polynomiale

n = 8



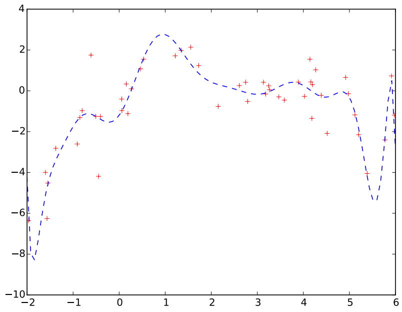
Exemple n° 2 : régression polynomiale

n = 10



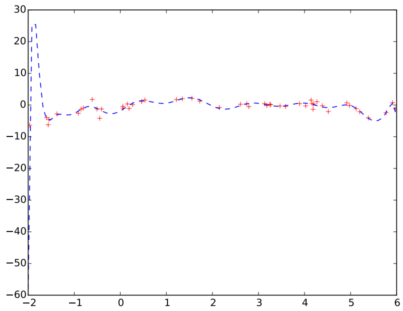
Exemple n° 2 : régression polynomiale

n = 15



Exemple n° 2 : régression polynomiale

n = 20



Comment évaluer les performances

Erreur en apprentissage :

$$\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{f}(x^{(i)})\right)^2$$

- erreur moyenne / moyenne des erreurs
- mesure à quel point le modèle appris « explique » les données

Les différentes erreurs

Erreur en apprentissage

L'erreur en apprentissage est calculée :

```
sum((inter_poly(xx) - yy) ** 2 for xx, yy in zip(x, y)) / le
```

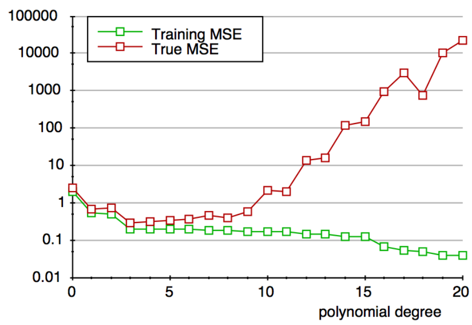
Erreur sur tous les points de l'ensemble d'apprentissage

Erreur « générale »

$$\frac{1}{x_b - x_a}\int_{x_a}^{x_b}\left(f_{\text{true}}(x) - \hat{f}(x)\right)^2 \mathrm{d}x$$

$\Rightarrow$ erreur sur tout le domaine de définition

Résultat



Conclusions (locales)

adapter la capacité du modèle aux données

Problème essentiel : validation

- erreur en apprentissage n'est pas un critère suffisant
- comment estimer la qualité d'un régresseur ?

Principe de l'évaluation

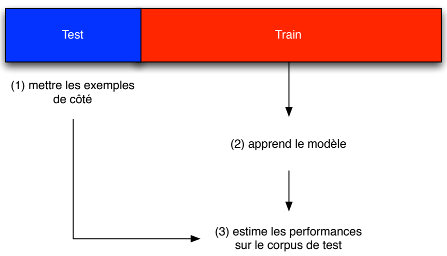
Le problème de l'erreur en apprentissage

- apprentissage par cœur
- pas (toujours) corrélée à l'erreur réelle (obtenue sur de nouvelles données)

Corpus de test

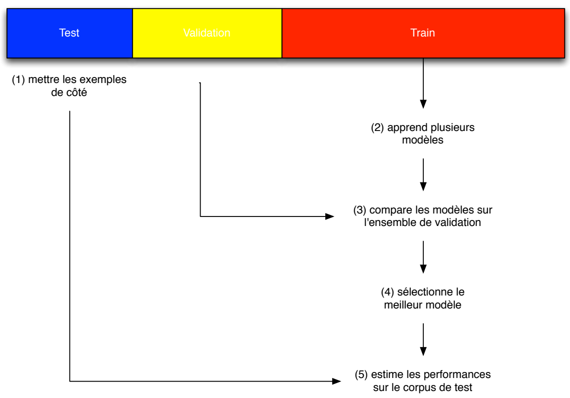
- = données utilisées uniquement pour l'évaluation
- si suffisamment nombreuses : l'erreur en test est un bon estimateur de l'erreur réelle

Séparation en train/test



- le corpus de test ne doit (devrait) être utilisé qu'une fois
- problème : exemples sacrifiés pour estimer les performances

Encore mieux !



Le mot de la fin

Exemple d'une tâche réelle :
http://archive.ics.uci.edu/ml/datasets/Forest+Fires