

MACHINE LEARNING

(APPRENTISSAGE AUTOMATIQUE)

par Andrew Ng
coursera.com



ORGANISATION DU MOOC

- Cours vidéo (3h/semaine) en 11 semaines
- Vidéos en anglais
- Andrew Ng, professeur associé en informatique à Stanford
- Quizz et TP à rendre chaque semaine (3h/semaine)
- Langage de programmation utilisé : Octave
- Apprentissage pratique, peu de calculs théoriques
- 28 septembre - 17 janvier

QU'EST CE QUE LE MACHINE LEARNING ?

→ donner la capacité à l'ordinateur
d'apprendre sans programmation explicite

Apprentissage supervisé

(connaissance des classes l'input/output,
modèle donné)



Régression linéaire

(valeur continues,
gros spectre de
données)



Régression logistique

(classification, valeur
discrètes)

Apprentissage non-supervisé

(le système n'a que les données, pas
leurs types)



Clustering

(regroupement
des données)

Première partie : Apprentissage supervisé

I. Régression linéaire

- A. Généralités
- B. Descente de gradient
- C. Equation normale

III. Réseaux de neurones

- A. Généralités
- B. Classification

II. Régression logistique

- A. Classification binaire
- B. Classification multi-classe
- C. Régularisation
- D. Support Vector Machines
 - 1. Généralités
 - 2. Frontière de décision
 - 3. Kernel

Deuxième partie : Apprentissage non supervisé

I. K-Means Algorithme

- A. Généralités
- B. Algorithme itératif
- C. Optimisation

II. Analyse en composantes principales

- A. Généralités
- B. Préprocessing et Algorithme

PREMIÈRE PARTIE : APPRENTISSAGE SUPERVISÉ

I. RÉGRESSION LINÉAIRE (UNE OU PLUSIEURS VARIABLES)

A. Généralités

Notation :

n : nombre de paramètre
m : nombre d'exemples
 $x^{(i)}$: i^{eme} exemple
 $x^{(i)}_{(j)}$: valeur du param. j
h : fonction d'hypothèse de l'output (par l'algorithme)

Données d'entraînement :

$x^{(i)}$: *input*
 $y^{(i)}$: *output*

Ensemble des données :

$((x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)}))$

PREMIÈRE PARTIE : APPRENTISSAGE SUPERVISÉ

I. RÉGRESSION LINÉAIRE (UNE OU PLUSIEURS VARIABLES)

A. Généralités

But :

Minimiser l'erreur entre la prédiction et la valeur réelle

Paramètre d'apprentissage

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \dots \\ \theta_n \end{bmatrix}$$

Cas linéaire :

$$h(\theta) = \theta_0 + \sum_{i=0}^n \theta_i * x_i$$

$$h(\theta) = \langle \theta \mid x \rangle = \theta^T x$$

Erreur quantifiée par la fonction de coût J :

$$J(\theta) = \frac{1}{2m} \sum_{i=0}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

B. Descente de Gradient

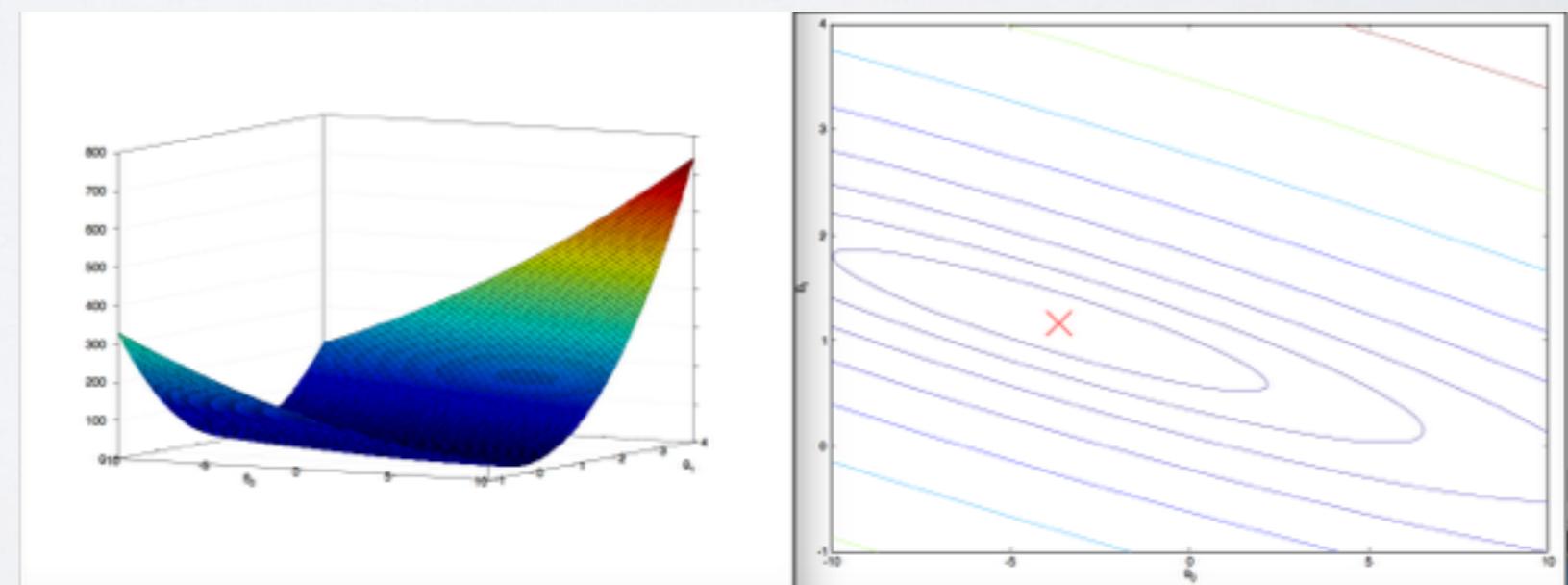
But :

Minimiser l'erreur entre la prédiction et la valeur réelle

Algorithme : Descente de Gradient

$$\begin{aligned} & \text{Repeat until } CV\{ \\ & \forall j \quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \\ & \Leftrightarrow \forall j \quad \theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=0}^m (h_\theta(x^{(i)}) - y^{(i)}) \times x_j^i \} \end{aligned}$$

Graphiques de J pour deux paramètres



Améliorer l'algorithme

- Trouver le pas optimal  α trop grand : saute le min
 α trop petit : algorithme trop lent
- Ajuster les paramètres : converge plus rapide

$$x_i \rightarrow \frac{x_i}{s_i} \in [-1; 1]$$

où $s_i = \max x_i - \min x_i$

- Normalisation par la moyenne

$$x_i \rightarrow \frac{x_i - \mu_i}{s_i}$$

où μ_i : valeur moyenne des x_i

C. Equation normale

→ méthode analytique, sans itération (méthode des moindres carrés)

$$X = \begin{bmatrix} x_1^1 & \dots & x_n^1 \\ \vdots & & \vdots \\ \vdots & & \vdots \\ x_1^m & \dots & x_n^m \end{bmatrix} \in \mathbb{R}^{n \times m}$$

matrice des exemples

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

vecteur des outputs réels

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

En partant de J et en dérivant, on obtient : $\theta = (X^t X)^{-1} X^t y$

Descente de Gradient

Choisir manuellement α
Beaucoup d'itérations
Optimal pour n grand (10^4)

Equation normale

Pas d'itérations
Calcul d'un inverse $O(n^3)$
Optimal pour n petit ($< 10^3$)

II. RÉGRESSION LOGISTIQUE

A. CLASSIFICATION BINAIRE

fonction logistique : $h_{\theta}(x) = \frac{1}{1+e^{(-\theta^t x)}} = \mathbb{P}(y = 1 \mid x; \theta) = p_{\theta}(y = 1 \mid x)$

Données → Ajustement de θ → Frontière de décision

h permet de prédire l'output et de définir la frontière de décision

$$y = 1 \text{ si } h_{\theta}(x) \geq 0,5 \Leftrightarrow y = 1 \text{ si } \theta^t x = 0$$

Fonction coût :

$$J(\theta) = \frac{1}{m} \sum_{i=0}^m \text{cost}(h_{\theta}(x), y)$$
$$\text{cost}(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{si } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{si } y = 0 \end{cases}$$

$$\Rightarrow J(\theta) = \frac{1}{m} \sum_{i=0}^m [y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))]$$

Descente de gradient

Repeat until CV

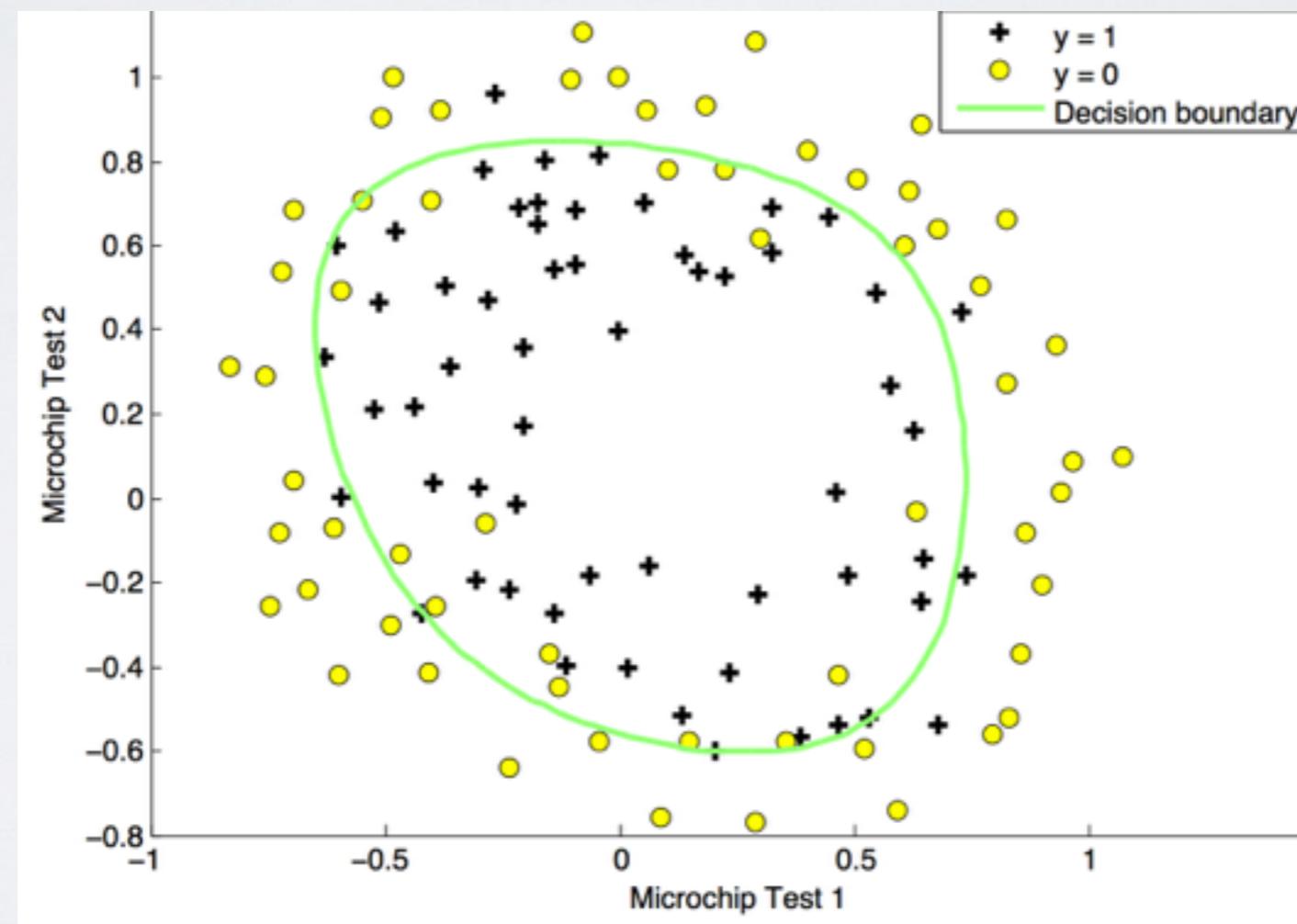
{

$$\forall j \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$\Leftrightarrow \forall j \theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=0}^m (h_\theta(x^{(i)}) - y^{(i)}) \times x_j^i$$

}

Autres algorithmes : Gradient Conjugué, BFGS (rapide mais plus complexe)

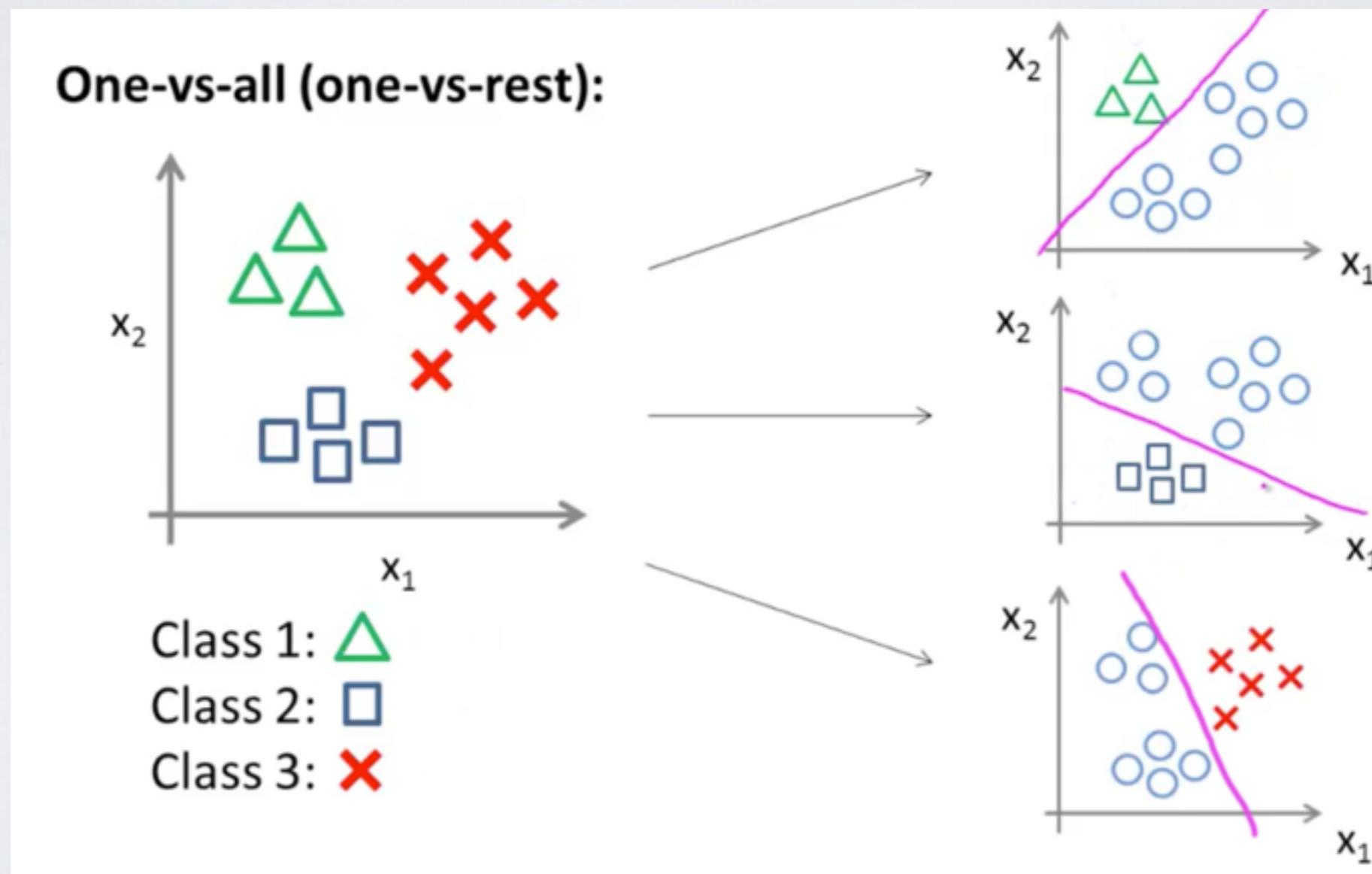


B. CLASSIFICATION MULTICLASSE

exemple : reconnaissance de chiffres écrits à la main (10 classes)

→ Algorithme One vs All

→ Choix de la classe i telle que $\max_i(h_{\theta}^{(i)}(x))$ tq $h_{\theta}^{(i)}(x) = P(y = i | x; \theta) \quad \forall i$



C. RÉGULARISATION

→ Optimisation de l'algorithme

Sur-apprentissage

Variance élevée
Polynôme trop haut (Lagrange)
 J quasiment nul
Impossible de généraliser

Sous-apprentissage

Estimateur biaisé
Trop peu de donnée
 J élevé

$$J(\theta) = \frac{1}{2m} \sum_{i=0}^m (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2$$

λ paramètre de régularisation, petites valeurs pour θ_j

λ trop petit

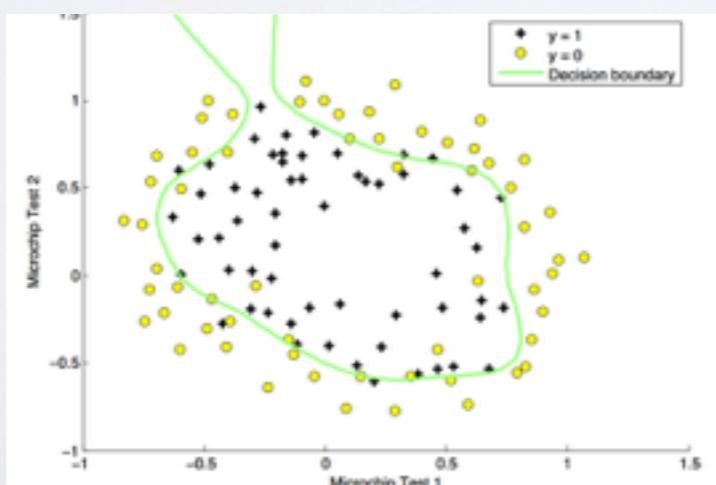


Figure 5: No regularization (Overfitting) ($\lambda = 0$)

λ trop grand

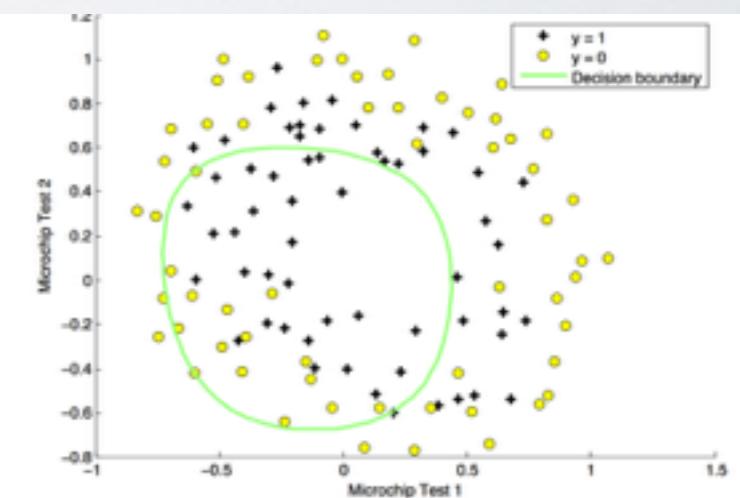


Figure 6: Too much regularization (Underfitting) ($\lambda = 100$)

C. SUPPORT VECTOR MACHINES

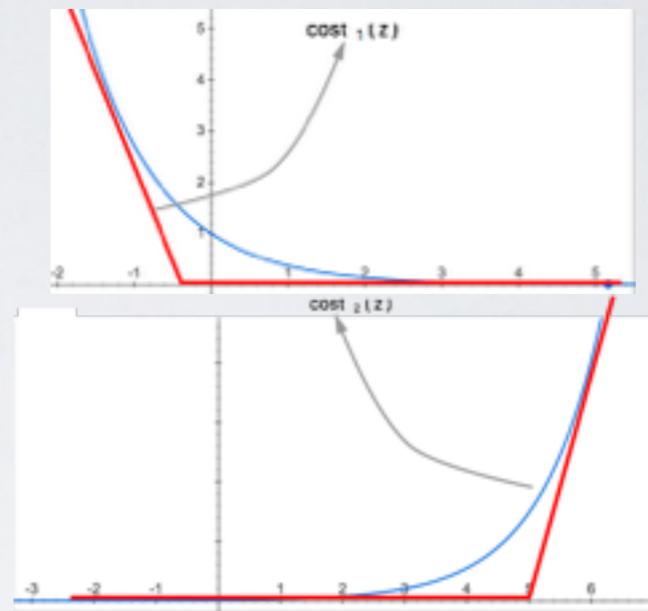
→ Séparer les exemples avec des marges aussi grosses que possibles

I. Généralités

$$J(\theta) = -y \log(h_\theta(x)) - (1-y) \log(1-h_\theta(x))$$

$$y = 1 \Rightarrow J = -\log \frac{1}{1 + e^{-\theta^t x}} \sim e^{-\theta^t x}$$

$$y = 0 \Rightarrow J = -\log(1 - \frac{1}{1 + e^{-\theta^t x}}) \sim \frac{1}{1 + e^{-\theta^t x}}$$



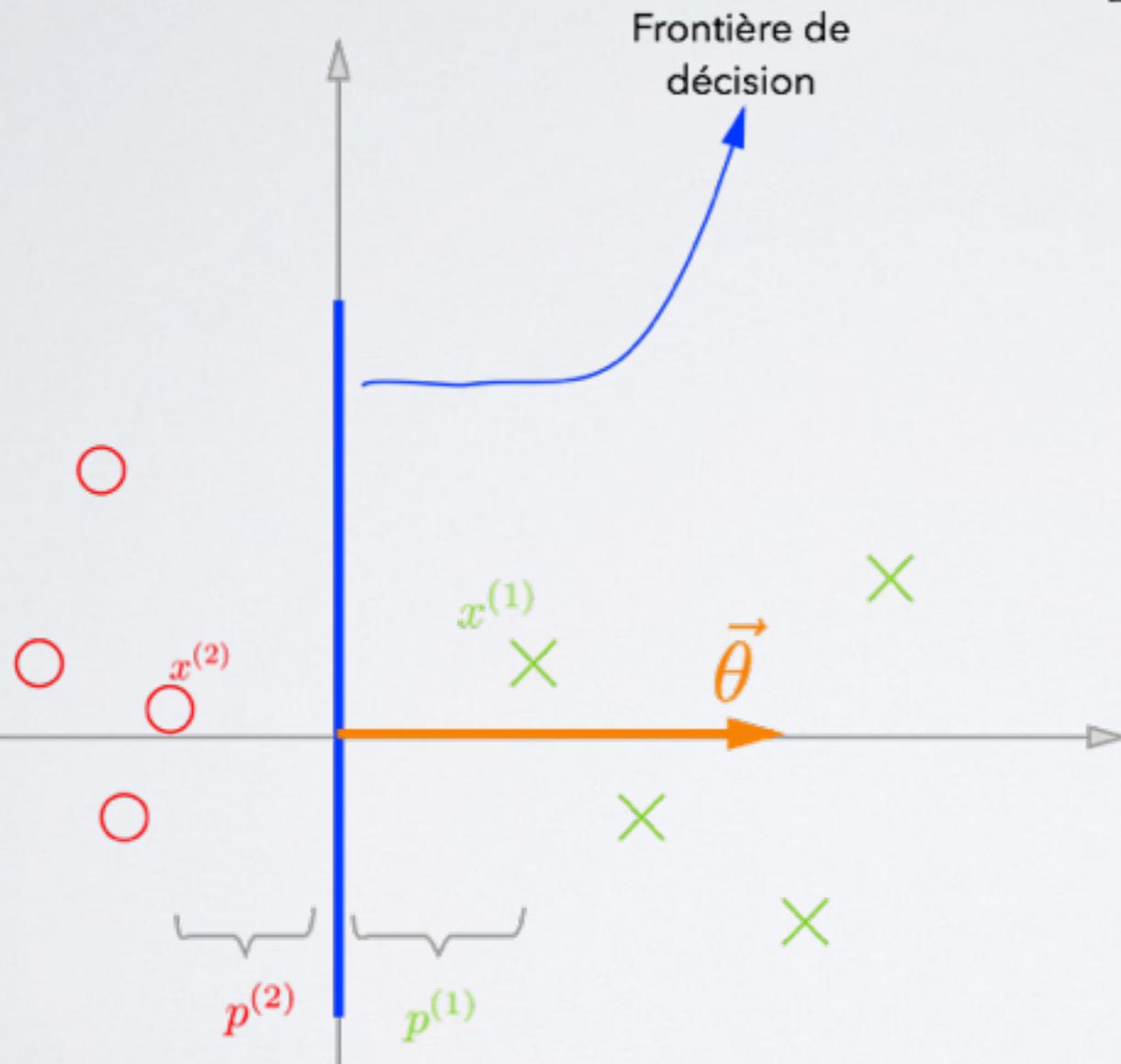
Hypothèse : $h_\theta(x) = \begin{cases} 1 & \text{si } \theta^t x \geq 1 \\ 0 & \text{si } \theta^t x \leq -1 \end{cases}$

But : $\min C \sum_{i=1}^m [y^{(i)} \text{cost}_1(\theta^t x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^t x^{(i)})] + \frac{1}{2} \sum_{j=1}^m \theta_j^2$

avec $C \sim \frac{1}{m}$ (régularisation)

2. Frontière de décision

$$\Rightarrow \min_{\theta} \frac{1}{2} \sum_{j=1}^n \theta_j^2 \Rightarrow \min_{\theta} \frac{1}{2} \| \theta \|^2$$



$p^{(i)}$ projection de $x^{(i)}$ sur $\vec{\theta}$

$$\begin{cases} p^{(i)} \parallel \theta \parallel \geq 1 \text{ si } y^{(i)} = 1 \\ p^{(i)} \parallel \theta \parallel \leq -1 \text{ si } y^{(i)} = 0 \end{cases}$$

$$\begin{array}{ll} p^{(i)} \nearrow & \text{si } y = 1 \\ p^{(i)} \searrow & \text{si } y = 0 \end{array}$$

effet de grande marge

3. Kernel

$l^{(i)}$: repère qui définit un paramètre

$$f_i = \text{similarity}(x, l^{(i)}) = \exp\left(-\frac{\|x - l^{(i)}\|}{2\sigma^2}\right)$$

$$l^{(i)} = x^{(i)}$$

$$x^{(i)} \rightarrow \begin{bmatrix} f_1^{(i)} = \text{sim}(x^{(i)}, l^{(1)}) \\ f_2^{(i)} = \text{sim}(x^{(i)}, l^{(2)}) \\ \dots \\ f_m^{(i)} = \text{sim}(x^{(i)}, l^{(m)}) \end{bmatrix}$$



Prédire des frontières non linéaires

$$f^{(i)} \in \mathbb{R}^{m+1}$$

$$SVM \ et \ Kernel : y = 1 \ si \ \theta^t f \geq 0$$

C

σ^2

Grand \Rightarrow peu biaise, haute variance (petit λ)
 \Rightarrow surapprentissage

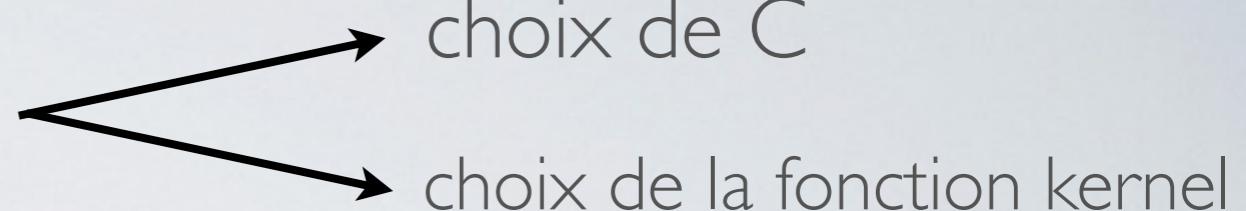
Grand \Rightarrow f_i varie lentement
 \Rightarrow fortement biaise petite variance

Petit \Rightarrow fortement biaise, petite variance (grand λ)
 \Rightarrow sousapprentissage

Petit \Rightarrow f_i varie rapidement
 \Rightarrow peu biaise haute variance

En pratique,

→ utilisation de package SVM



n : paramètres m : nombre d'exemples

- n est grand, n petit

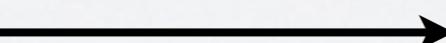
$$10^4 \sim n \gg m \sim 10^3$$

régression logistique

SVM sans kernel ($y = 1$ si $\theta^t x \geq 0$)

- n est petit, m est moyen

$$n \leq 10^3 \quad m \sim 10^4$$



SVM kernel gaussienne
($y = 1$ si $\theta^t f \geq 0$)

- n est petit, m est grand

$$n \ll m$$



ajouter des paramètres



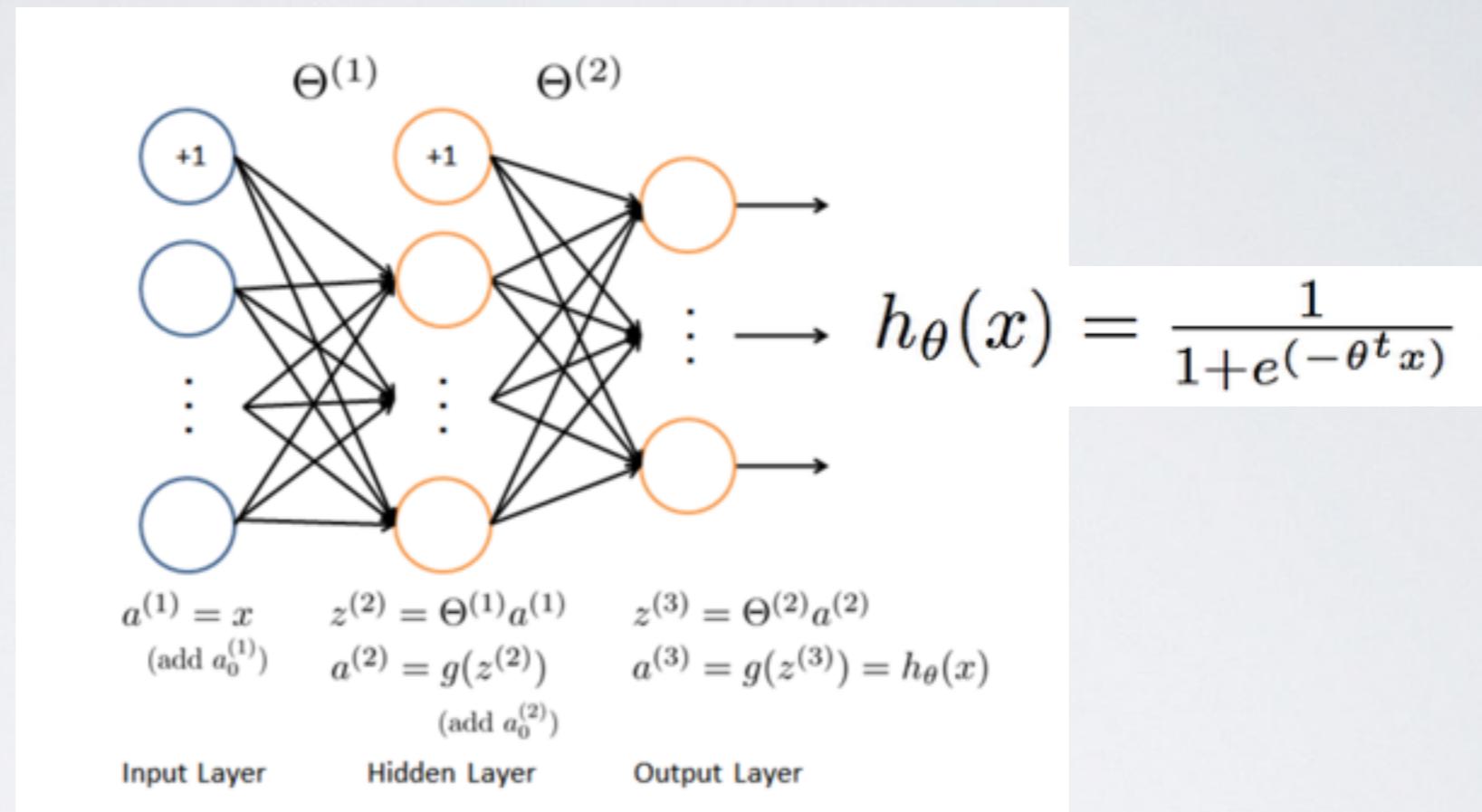
régression logistique

SVM sans kernel

III. RÉSEAUX DE NEURONES

→ pour des n grands → **exemple** : vision par ordinateur

A. GÉNÉRALITÉS



$a_i^{(j)}$: activation de l'unité i dans la couche j

$\theta^{(j)}$: matrice des poids contrôlant les fonctions de combinaison couche j à couche $j+1$

$$a_i^{(j+1)} = g\left(\sum_{i=0}^n \theta_{1,i}^{(j)} a_i^{(j)}\right)$$

B. CLASSIFICATION

L : nombre de couches

s_j : nombre d'unités dans la couche j

s_L = 1 → 1 unité output → classification binaire

s_L = K → K unité output → classification multi-classe

Coût :

$$J(\theta) = \frac{-1}{m} \left[\sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(h_\theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - h_\theta(x^{(i)}))_k \right] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\theta_{j,i}^{(l)})^2$$

Algorithme de rétropropagation (pour minimiser J):

- calcul des activations par une propagation directe avec les données
- calcul des erreurs pour chaque neurone i dans la couche l

Algorithme de rétropropagation

Données : $(x^{(1)}, y^{(1)}) \dots (x^{(m)}, y^{(m)})$

$$\Delta_{i,j}^{(l)} = 0 \quad \forall i, j, l$$
$$\forall i \in [|1; m|]$$

{

Propagation du signal : $a^{(1)} = x^{(i)}$

$$a^{(l+1)} = g(\theta^{(l)} a^{(l)})$$

Pour le neurone de sortie : $\delta^{(L)} = a^{(L)} - y^{(i)}$

$$\delta^{(i)} = (\Theta^{(i)})^t \delta^{(i+1)} * g'(\theta^{(i)} a^{(i)}) \quad \forall i \in [|2; L-1|]$$

$$\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^t$$

}

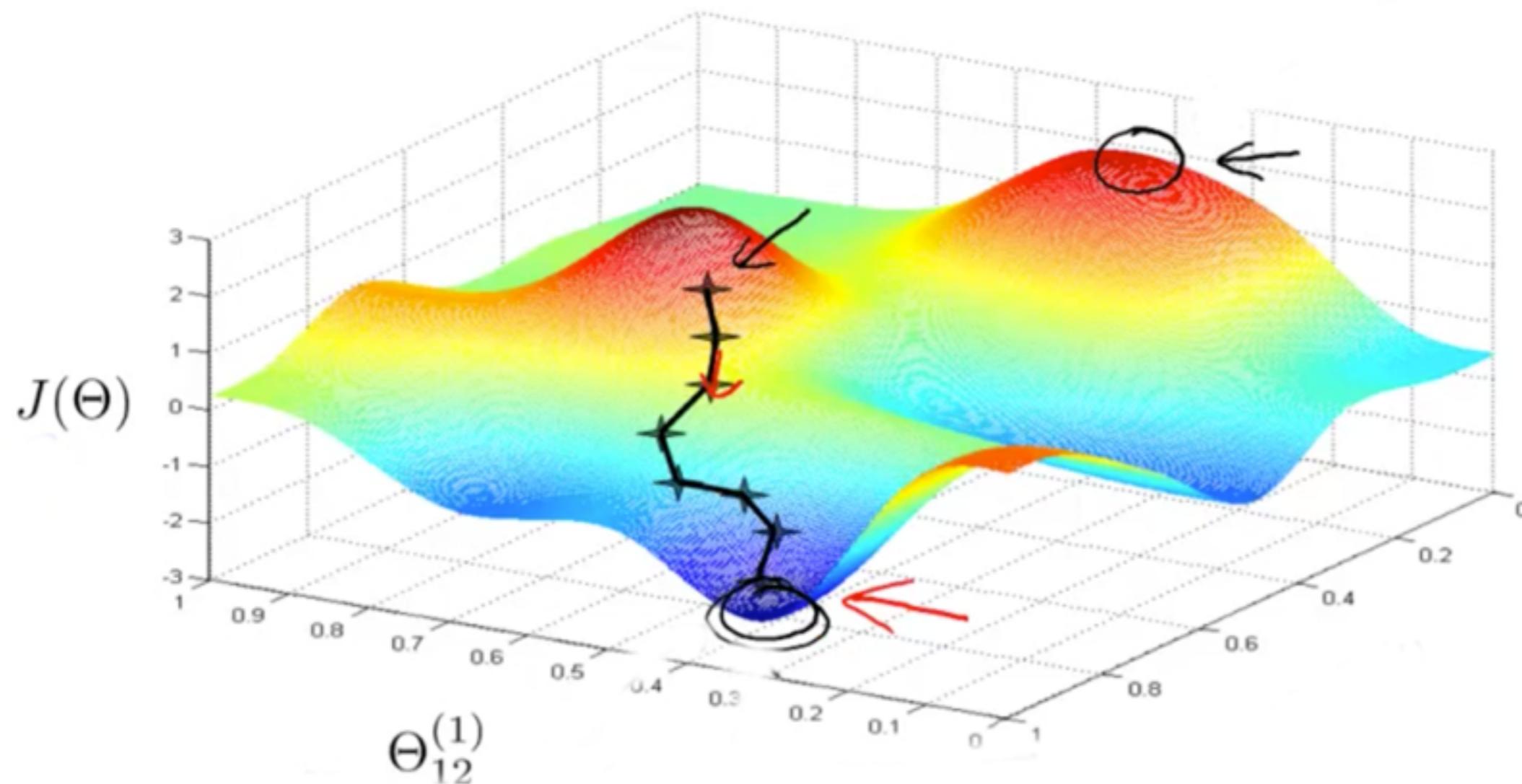
$$D_{i,j}^{(l)} := \frac{1}{m} \Delta_{i,j}^{(l)} + \lambda \Theta_{i,j}^{(l)} \text{ si } j = 0$$

$$D_{i,j}^{(l)} := \frac{1}{m} \Delta_{i,j}^{(l)} \text{ si } j \neq 0$$

$$\text{Finalement : } \underbrace{\frac{\partial}{\partial \Theta_{i,j}^{(l)}}}_{J(\Theta)} = D_{i,j}^{(l)}$$

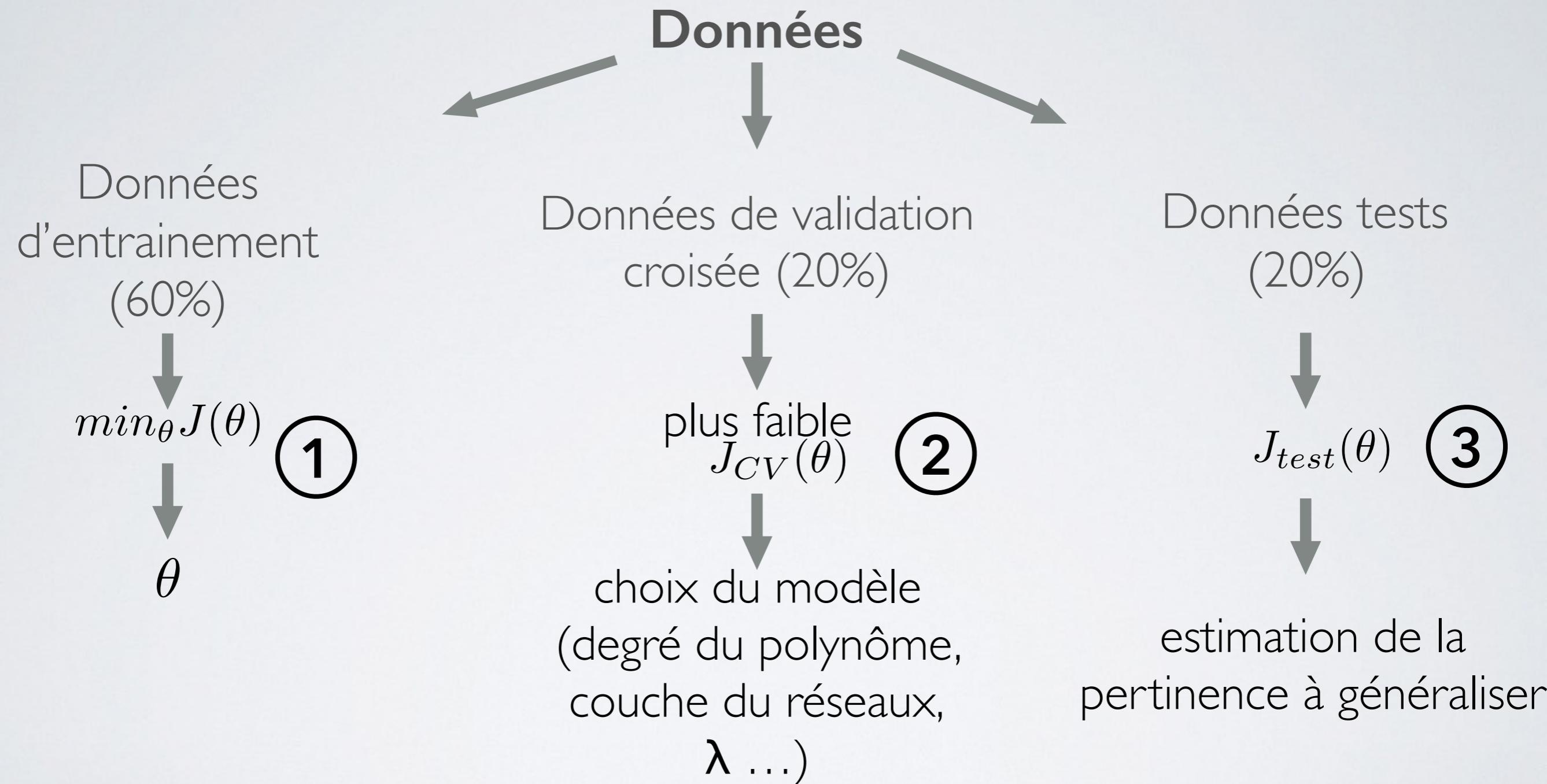
possibilité d'utiliser la descente de gradient ou autre algorithme pour minimiser J

Algorithme de rétropropagation



APPLIQUER LE MACHINE LEARNING

Faire un diagnostic pour debugger l'algorithme



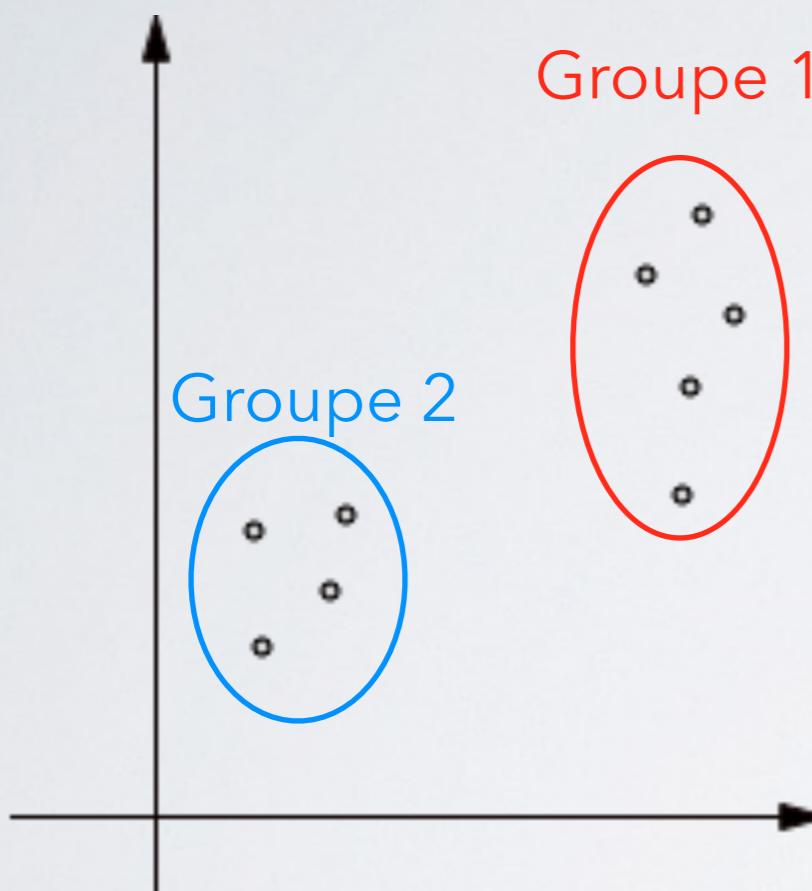
cf ex 6 !!!!

cf quiz semaine 7

DEUXIÈME PARTIE : APPRENTISSAGE NON-SUPERVISÉ

I. K-MEANS ALGORITHME

A. Généralités



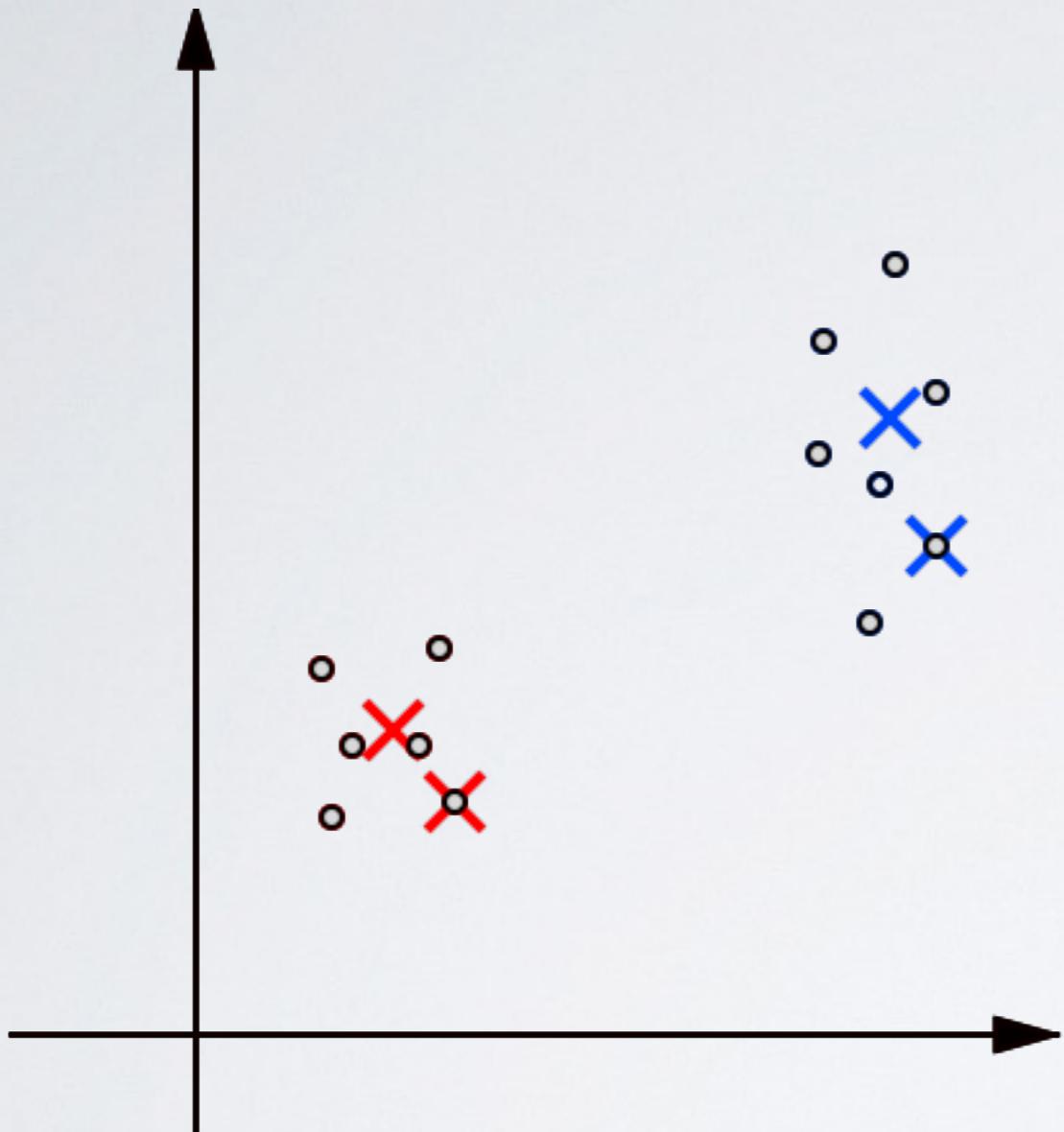
- données non étiquetés
- algorithme qui trouve la structure
- clustering

segmentation
du marché

analyse de
réseaux sociaux

→ K-Means algorithme

B. Algorithme itératif



✖ : centre du groupe

$$x^{(i)} \in \mathbb{R}^n$$

$$x_0 = 1$$

Input :

- K, nombre de groupes
- Données $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

Initialisation du centre

- I. Attribution de points au groupe
- II. Déplacement du centre à la moyenne des points

Initialisation aleatoire des K centres $\mu_1, \dots, \mu_K \in \mathbb{R}^n$

Répéter {

for $i = 1$ to m

*① $c^{(i)} := \text{index (de } 1 \dots K \text{) du centre le plus proche de } x^{(i)}$
i.e $x^{(i)} := \min_k \|x^{(i)} - \mu_k\|$*

for $k = 1$ to K

② $\mu_k := \text{moyenne des points attribués au groupe } k$

C. Optimisation

$\mu_{c^{(i)}}$: centre du groupe auquel l'exemple $x^{(i)}$ a été attribué

Distorsion : $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$

→ à minimiser selon $c^{(1)}, \dots, c^{(m)}$ ① et selon μ_1, \dots, μ_K ②

Initialisation : Réaliser plusieurs K-Means algorithmes avec différents premiers centres (égaux à des exemples)

→ choisir celui avec la plus petite distorsion

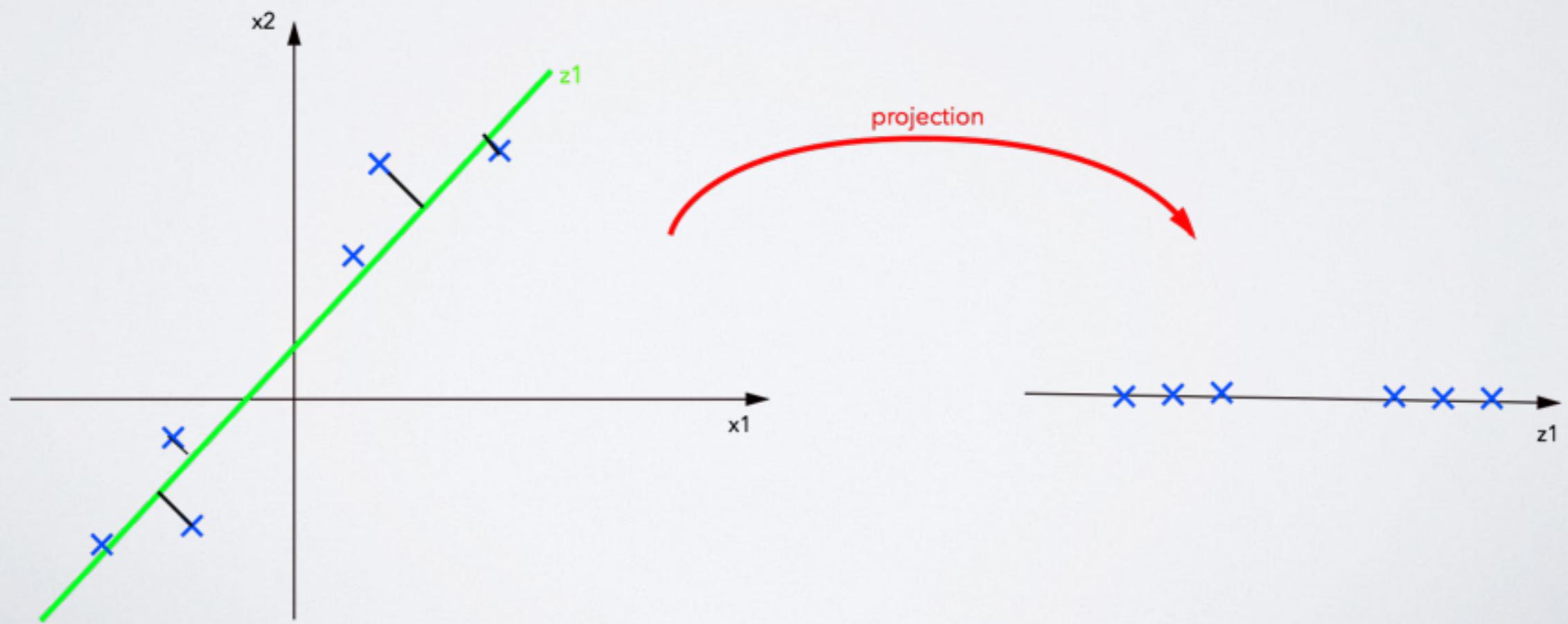
Nombre de groupes : Elbow Méthode ou à la main

II. ANALYSE EN COMPOSANTES PRINCIPALES

A. Généralités

réduction de dimension de n à k (compression de données)

- réduire l'erreur de projection
- trouver les k directions pour projeter les données
- ajuster les paramètres au préalable



B. Preprocessing et Algorithme

$$\mu_j = \frac{1}{m} \sum_{j=0}^m x_j$$

$$x_j \leftarrow x_j - \mu_j$$

ou

$$x_j \leftarrow \frac{x_j - \mu_j}{s_j}$$

Calcul de la matrice de covariance :

$$\Sigma = \frac{1}{m} \sum_{i=1}^m x^{(i)} (x^{(i)})^t \in S_n^+$$

$$(\Sigma = \frac{1}{m} X^t X)$$

Calcul des vecteurs propres de Σ :

Decomposition en valeurs singulières

$$[U, \quad S, \quad V] = svd(\Sigma);$$

$$U = \begin{bmatrix} \cdot & \cdot & \cdots & \cdot \\ u^{(1)} & u^{(2)} & \cdots & u^{(n)} \\ \cdot & \cdot & \cdots & \cdot \end{bmatrix} \in \mathbb{R}^{n \times n}$$


$$k \text{ directions} \Rightarrow U_{reduc} \in \mathbb{R}^{n \times k}$$

$$\Rightarrow X_{approx} = U_{reduc} Z$$

Décomposition en valeurs singulières

Variation totale :

$$\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2$$

Erreur de projection moyenne :

$$\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{approx}^{(i)}\|^2$$

$$S = \begin{bmatrix} \cdot & & 0 \\ & \ddots & \\ 0 & & \ddots \end{bmatrix} \in M_n \Rightarrow variance = \sum_{i=1}^n S_{ii}$$

Pour retenir 99% de la variance :

$$\frac{\frac{1}{m} \sum_{i=1}^k S_{ii}}{\frac{1}{m} \sum_{i=1}^n S_{ii}} \geq 0,99 \Leftrightarrow \frac{\frac{1}{m} \sum_{i=1}^n \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m} \sum_{i=1}^n \|x^{(i)}\|^2} \leq 0,01$$

Détection d'anomalies

$$(x^{(1)}, x^{(1)}, \dots, x^{(m)}) \rightarrow p(x)$$

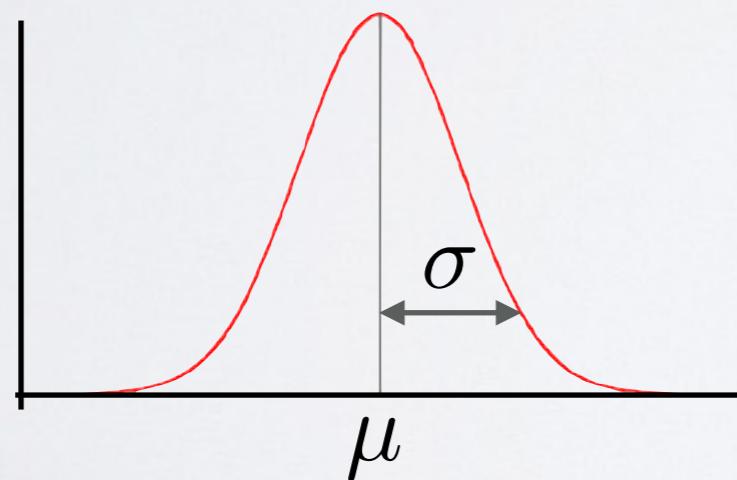
Est ce que x_{test} est une anomalie ?

Anomalie si $p(x_{test}) < \epsilon$

Modèle

$$X \sim N(\mu, \sigma^2)$$

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$



$$\begin{aligned} EMV_m : \\ \hat{\mu} &= \frac{1}{m} \sum_{i=0}^m x^{(i)} \\ \hat{\sigma} &= \sqrt{\frac{1}{m} \sum_{i=0}^m (x^{(i)} - \hat{\mu})^2} \end{aligned}$$

$$\text{estimateur de densité : } p(x) = \prod_{i=1}^m p(x_i; \mu_i; \sigma_i^2)$$

Algorithme de détection d'anomalies

1. Choix des $x^{(i)}$ (indicateurs d'anomalies)
2. Ajustement des paramètres

$$\mu_j = \frac{1}{m} \sum_{i=0}^m x_j^{(i)}$$

$$\sigma_j = \sqrt{\frac{1}{m} \sum_{i=0}^m (x_j^{(i)} - \mu_j)^2}$$

3. Calcul x_{test}

$$p(x) = \frac{1}{(2\pi)^{n/2} \prod_{i=1}^n \sigma_i} \exp\left(-\sum_{j=1}^m \frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

\Rightarrow Anomalie si $p(x) < \epsilon$

Différentes détection d'anomalies et apprentissage supervisé

Détection d'anomalies

peu d'anomalies ($y=1$)
beaucoup de données normales
plusieurs types d'anomalies
exemple :

détection de fraudes
industries
supervision de machines

Apprentissage supervisé

autant de vraies ($y=1$) données
que de fausses ($y=0$)
assez de données pour prédire
les données vraies

exemple :

classificateur de spam
prédiction de météo
classificateur médicale

Systèmes de recommandations

- apprentissage automatique des paramètres importants
- prédiction de notation par régression linéaire

$n_u = \text{nombre d' users}$

$n_m = \text{nombre de films}$

$m^{(j)} = \text{nombre de films notes par } j$

$r^{(i,j)} = 1 \text{ si } j \text{ a note } i$

$y^{(i,j)} = \text{note par } j \text{ sur } i$

$x_p^{(i)} = \text{note du film } i \text{ pour le parametre } p$

Prédiction des données

Devinez $\theta \rightarrow \text{apprentissage de } x \rightarrow \text{amelioration de } \theta \rightarrow \text{amelioration de } x \rightarrow \dots$

- deux régressions linéaires séparées

Algorithme de recommandation

1. *Initialisation* $x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}$

2. *Minimisation par descente de gradient*

$\forall i \forall j$

$$x_k^{(i)} := x_k^{(i)} - \alpha [\sum_{j:r(i,j)=1} (\theta^{(j)T} x^{(i)} - y^{(i,j)}) \theta_k^{(j)} + \lambda x_k^{(i)}]$$

$$\theta_k^{(i)} := \theta_k^{(i)} - \alpha [\sum_{i:r(i,j)=1} (\theta^{(j)T} x^{(i)} - y^{(i,j)}) x_k^{(j)} + \lambda \theta_k^{(i)}]$$

3. *Prediction de note* $\theta^T x$

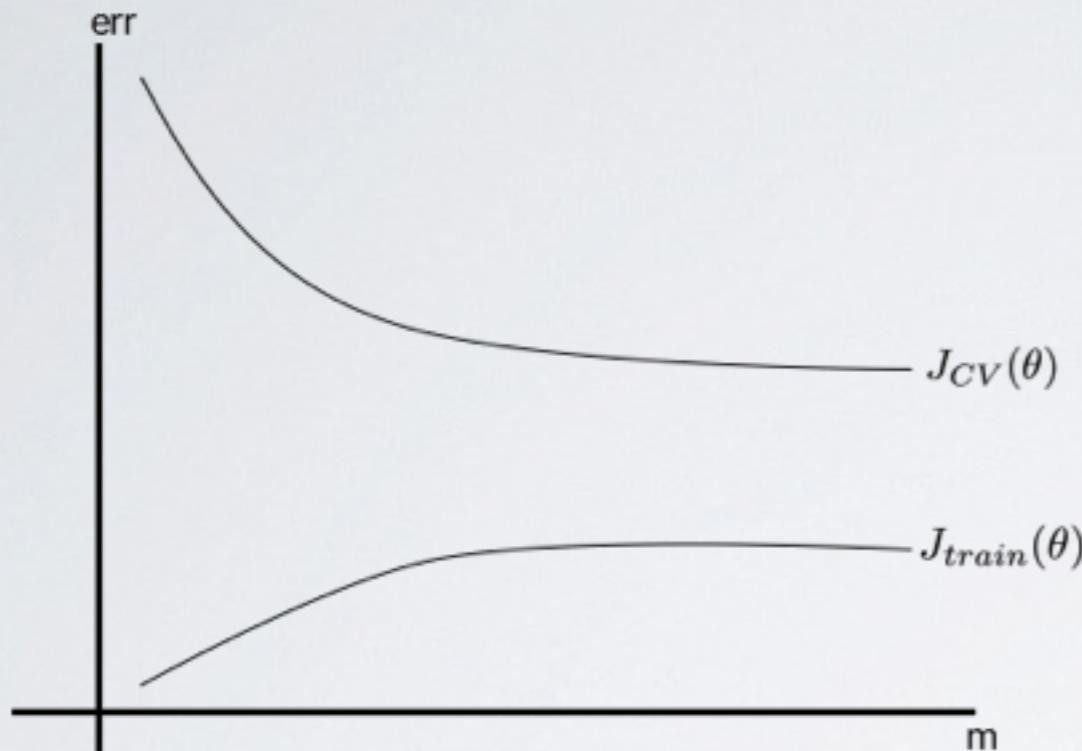
Trouver les films similaires

1. $\forall movie i \rightarrow apprendre x^{(i)} \in \mathbb{R}^n$

2. $\min_i \|x^{(j)} - x^{(i)}\|$

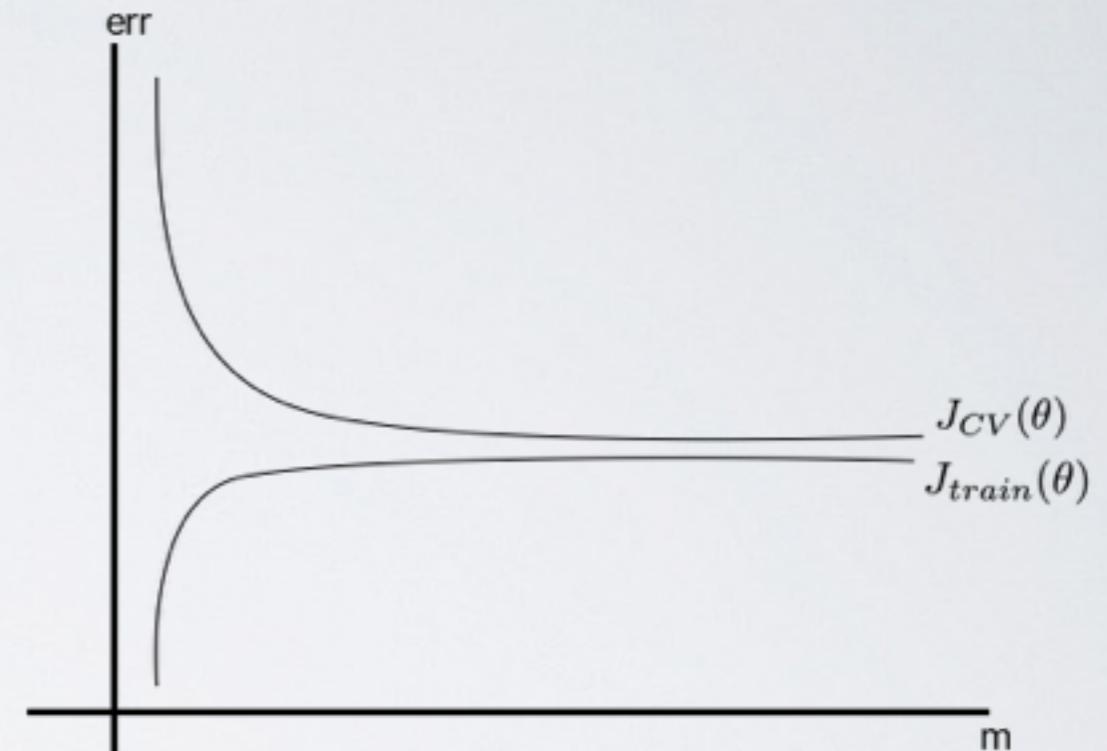
Machine Learning sur de grosses bases de données

→ problème de calculs liés au nombre de données $m \approx 10^8$



Variance élevée

→ besoin d'un grand m



Biais élevé

- pas besoin d'un grand m
- besoin de paramètres supplémentaires

→ Gradient stochastique pour remplacer le Batch gradient
utilisation du gradient sur chaque exemple (pas de somme sur les m données)

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_{\theta}(x^{(i)}) - y^{(i)})^2$$
$$J_{train}(\theta) = \frac{1}{m} \sum_{i=1}^m cost(\theta, (x^{(i)}, y^{(i)}))$$

Algorithme

1. *Initialisation aleatoire des donnees d'entrainement*

2. *Repeter {*

{

$\forall i \in [1, m]$

$\forall j \in [1, n] \quad \theta_j := \theta_j - \alpha(h_{\theta}(x^{(i)}) - y^{(i)})x_j^{(i)}$ }

}

→ approche autour du minimum global, pas exact

→ On répète moins de 10 fois l'étape 2

À chaque itération, on utilise :

Batch gradient : m exemples

Stochastique gradient : un exemple

$b \sim 2 - 100$

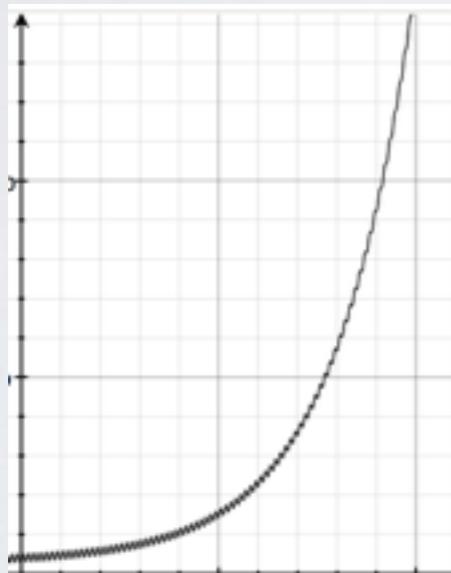
Mini-Batch gradient : b exemples

Vérification de la convergence

Batch gradient : graphe de J_{train} en fonction des itérations

Stochastique gradient :

- pendant l'apprentissage, calcul de $\text{cost}(\theta, (x^{(i)}, y^{(i)}))$ avant de modifier θ
- Toutes les 1000 itérations, on trace la moyenne $\text{cost}(\theta, (x^{(i)}, y^{(i)}))$ sur ces itérations



Divergence → utiliser plus petit α