

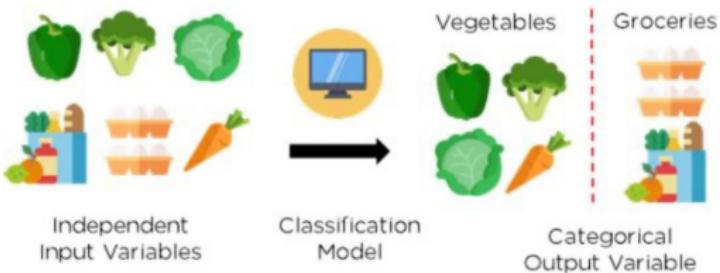
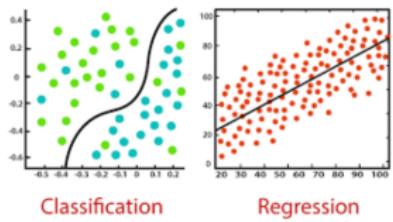
MAPSI — cours 9 : machine learning & classification

Nicolas Thome
nicolas.thome@isir.upmc.fr

LIP6 / ISIR – Sorbonne Université, France

Classification

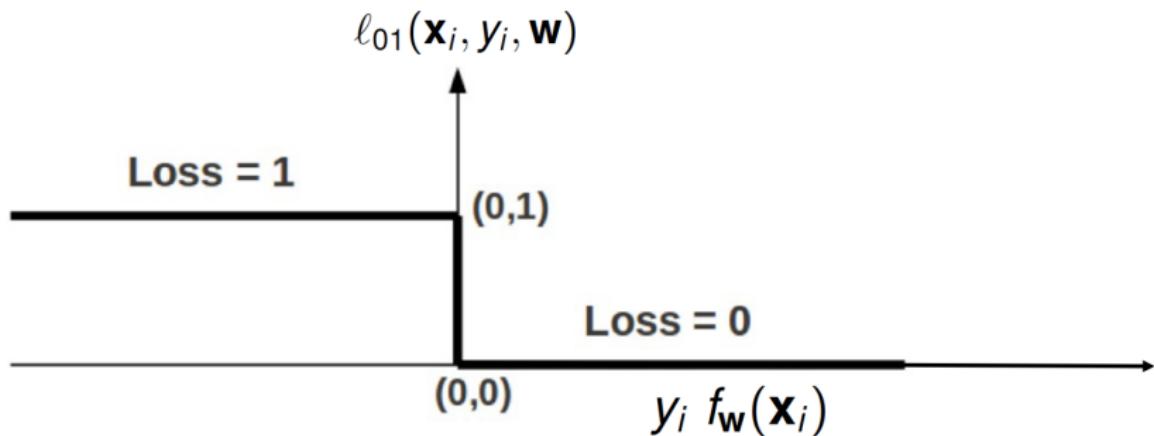
- **Régression** : prédire une sortie continue
- **Classification** : prédire un sortie discrète catégorielle,
 $y \in \{1; K\}$



- 1 Méthodes par coût
- 2 Méthodes probabilistes
- 3 MAPSI et le machine learning

Classification binaire

- **Classification binaire** : 2 classes $\Rightarrow y \in \{-1; +1\}$
- Exemple d'entrée $\mathbf{x}_i \in \mathbb{R}^d$
- Modèle de prédiction : $f_{\mathbf{w}}(\mathbf{x}_i) \in \mathbb{R}$, fonction de score
 - Peut être convertie en probabilité $\in [0, 1]$
- Fonction de perte "0/1 loss" : $\ell_{01}(\mathbf{x}_i, y_i, \mathbf{w}) = 1 - \text{sign}[y_i \cdot f_{\mathbf{w}}(\mathbf{x}_i)]$
 - $\ell_{01}(\mathbf{x}_i, y_i, \mathbf{w}) = 0$ si \mathbf{x}_i bien classé ($f_{\mathbf{w}}(\mathbf{x}_i)$ même signe que y_i)
 - $\ell_{01}(\mathbf{x}_i, y_i, \mathbf{w}) = 1$ si \mathbf{x}_i mal classé ($f_{\mathbf{w}}(\mathbf{x}_i)$ signe opposé à y_i)

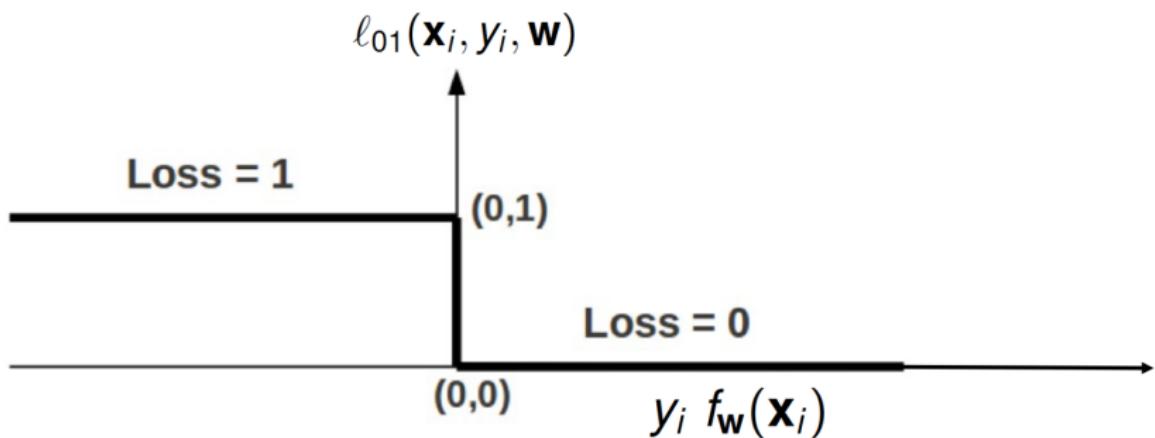


Classification & apprentissage

- Apprentissage : N couples i.i.d $(\mathbf{X}, \mathbf{Y}) := (\mathbf{x}_i, y_i)_{i \in \{1;N\}}$
- Fonction de perte en apprentissage :

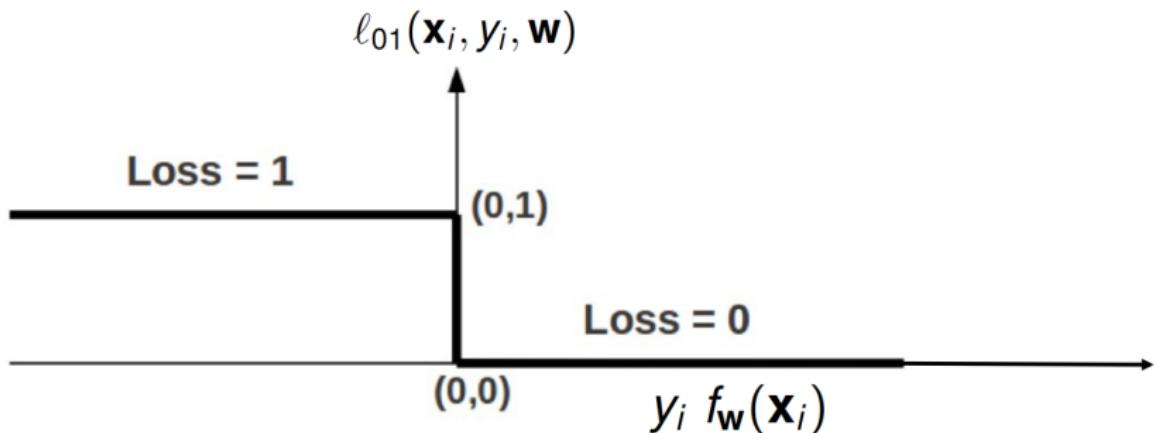
$$\mathcal{L}_{01}(\mathbf{w}, X, Y) = \frac{1}{N} \sum_{i=1}^N 1 - \text{sign}[y_i \cdot f_{\mathbf{w}}(\mathbf{x}_i)]$$

- But : trouver \mathbf{w}^* tq : $\mathbf{w}^* = \arg \min_{\mathbf{w}} \mathcal{L}_{01}(\mathbf{w}, X, Y)$
- Problème : comment minimiser \mathcal{L}_{01} ? Fonction non dérivable



Classification & apprentissage

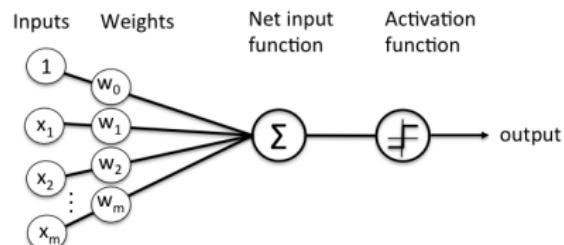
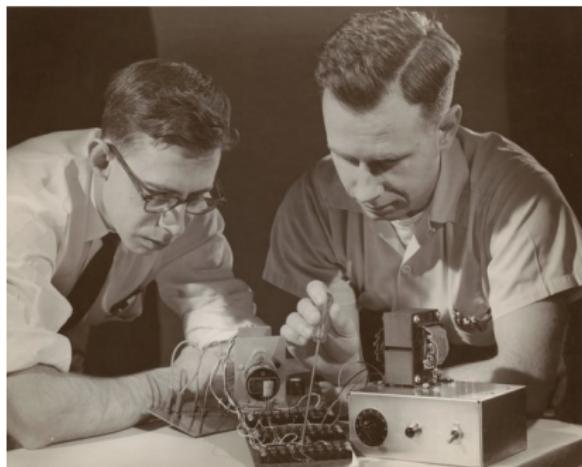
$$\mathcal{L}_{01}(\mathbf{w}, X, Y) = \frac{1}{N} \sum_{i=1}^N 1 - \text{sign}[y_i \cdot f_{\mathbf{w}}(\mathbf{x}_i)]$$



- Problème : comment minimiser \mathcal{L}_{01} ? fonction non dérivable
- Loss de régression $(y_i - f_{\mathbf{w}}(\mathbf{x}_i))^2$ non adaptée : pourquoi ?

Perceptron

Historique (1957, 10 ans avant la disquette ou le PC !!)



Schematic of Rosenblatt's perceptron.

- Fonction de score linéaire : $f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i = \sum_j x_{ij} w_j$
 - Avec bias ou données $\mathbf{x}_e := [\mathbf{x} \ 1]$ (voir cours régression)
- Fonction de perte : $C(\mathbf{w}) = \sum_{i=1}^N (-y_i \cdot \mathbf{w}^T \mathbf{x}_i)_+$, $(x)_+ = \max(0, x)$
 - $C(\mathbf{w})$: hinge loss, perte charnière
 - $C(\mathbf{w}) = 0$ si bien classé, coût $(-y_i \cdot \mathbf{w}^T \mathbf{x}_i)$ linéaire / score sinon

- Moindres carrés :
 - pas bien adapté à la classification
- Hinge (= charnière)
 - Limité au cas binaire, mais mieux adapté à la classification

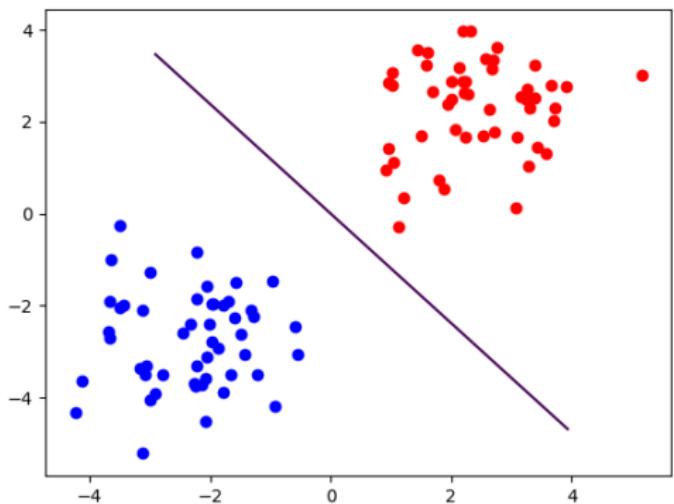
Descente de gradient

- ❶ Initialiser \mathbf{w}_0
- ❷ Bouclage jusqu'à cvg
 - Calcul de
$$\nabla_{\mathbf{w}} C = \sum_{i|y_i f(\mathbf{x}_i) \leq 0} -y_i \mathbf{x}_i$$
 - MAJ : $\mathbf{w}^{t+1} = \mathbf{w}^t - \epsilon \nabla_{\mathbf{w}} C$

Algorithme stochastique

- ❶ Initialiser \mathbf{w}_0
- ❷ Bouclage jusqu'à cvg
 - Tirage aléatoire d'un échantillon i
 - Si $y_i \mathbf{x}_i \mathbf{w} \leq 0$
 - Calcul de $\nabla_{\mathbf{w}} C_i = -y_i \mathbf{x}_i$
 - MAJ : $\mathbf{w}^{t+1} = \mathbf{w}^t - \epsilon \nabla_{\mathbf{w}} C_i$

Hinge/charnière : attention aux différentes fonctions optimales



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

- Super frontière... Optimale ($C(\mathbf{w}) = 0$)...
- Mais est-ce la seule frontière optimale ?

2 modifications principales / perceptron

- ➊ **Marge** : renforcer la classification des points limites :

$$f(\mathbf{x}_i) = \sum_j x_{ij} w_j, \quad C(\mathbf{w}) = \sum_{i=1}^N (\textcolor{red}{1} - y_i \cdot \mathbf{w}^T \mathbf{x}_i)_+$$

2 modifications principales / perceptron

- ➊ **Marge :** renforcer la classification des points limites :

$$f(\mathbf{x}_i) = \sum_j x_{ij} w_j, \quad C(\mathbf{w}) = \sum_{i=1}^N (\textcolor{red}{1} - y_i \cdot \mathbf{w}^T \mathbf{x}_i)_+$$

⇒ aucun impact si $\mathbf{x}_i \mathbf{w}$ n'est pas normalisé, la fonction de score est définie à un facteur près.

Vers les Supports Vectors Machines (SVM)

2 modifications principales / perceptron

- ➊ Marge : renforcer la classification des points limites :

$$f(\mathbf{x}_i) = \sum_j x_{ij} w_j, \quad C(\mathbf{w}) = \sum_{i=1}^N (\textcolor{red}{1} - y_i \cdot \mathbf{w}^T \mathbf{x}_i)_+$$

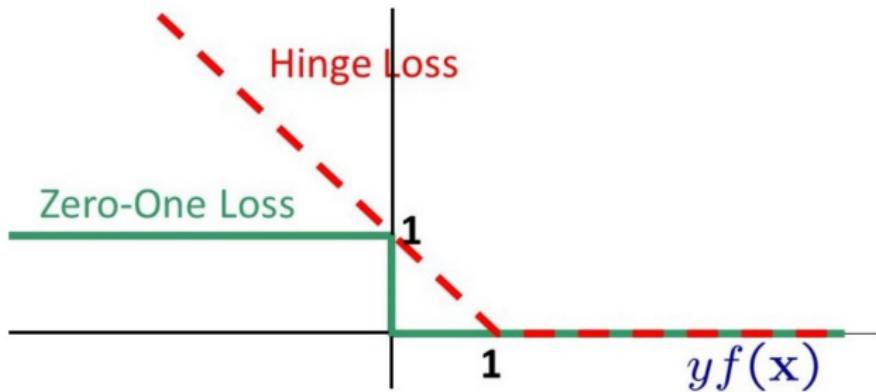
⇒ aucun impact si $\mathbf{x}_i \mathbf{w}$ n'est pas normalisé, la fonction de score est définie à un facteur près.

- ➋ Imposer la marge + contraindre la norme de fonction de score :

$$f(\mathbf{x}_i) = \sum_j x_{ij} w_j, \quad C(\mathbf{w}) = \sum_{i=1}^N (\textcolor{red}{1} - y_i \cdot \mathbf{w}^T \mathbf{x}_i)_+ + \lambda \|\mathbf{w}\|^2$$

SVM : hinge loss

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

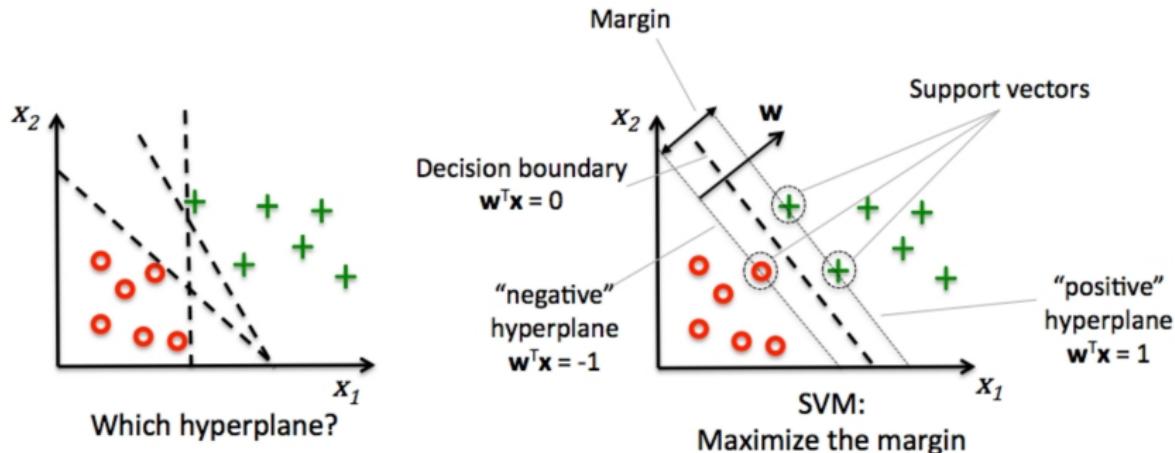


Bonnes propriétés :

- **Borne supérieure à \mathcal{L}_{01}**
- **Fonction convexe** \Rightarrow minimiseur global par descente de gradient (pas bien choisi)

SVM : maximisation de la marge

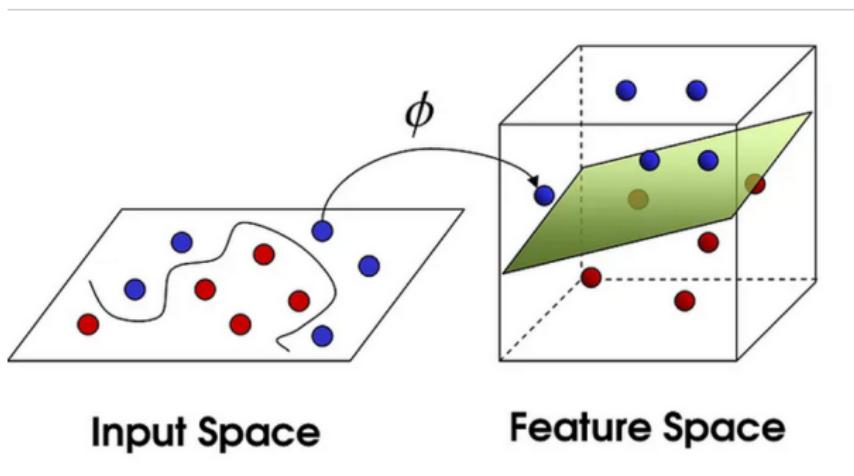
$$C(\mathbf{w}) = \sum_{i=1}^N (1 - y_i \cdot \mathbf{w}^T \mathbf{x}_i)_+ + \lambda \|\mathbf{w}\|^2$$



- Min $\|\mathbf{w}\|^2 \Leftrightarrow$ **maximisation de la marge géométrique** $\frac{1}{\|\mathbf{w}\|^2}$ (\oplus/\ominus)
- Solution unique**
- Borne sur l'erreur de généralisation**

Au-delà des modèles linéaires

- Modèles linéaires : limités à des hyperplans séparateurs
- Méthodes à noyau "kernel methods") : projeter les données dans un nouvel espace et appliquer le modèle linéaire dans l'espace transformé
 - $f_w(\mathbf{x}_i) = \mathbf{w}^T \phi(\mathbf{x}_i)$
 - Cf régression polynomiale cours 8 + kernel trick

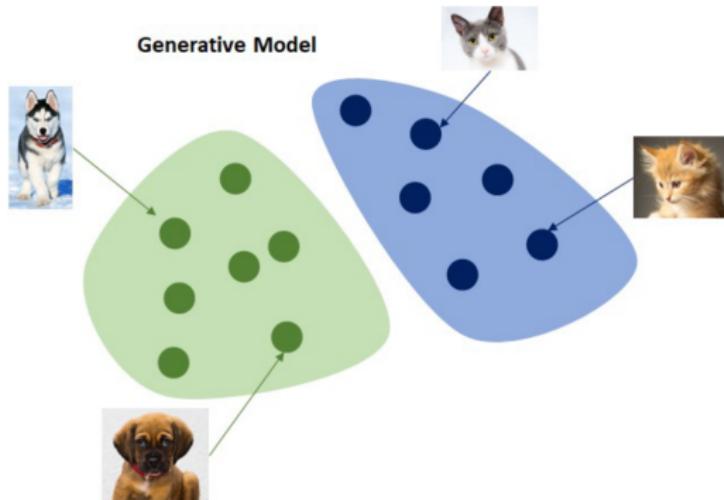


- k-NN, arbres de décision, réseaux de neurones profonds, etc

- 1 Méthodes par coût
- 2 Méthodes probabilistes
- 3 MAPSI et le machine learning

Rappel sur les modèles génératifs

- N couples i.i.d $(\mathbf{X}, \mathbf{Y}) := (\mathbf{x}_i, y_i)_{i \in \{1; N\}}$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \llbracket 1; K \rrbracket$
- On modélise $p(\mathbf{x}_i, y_i, \theta) = p(\mathbf{x}_i | y_i, \theta)p(y_i, \theta)$



- **Apprentissage : apprendre θ_k pour chaque classe indépendamment**

(regrouper \mathbf{x}_i tq $y_i = k$) : $p(\mathbf{X}, \mathbf{Y} = k, \theta_k) = \prod_{i=1}^{N_k} p(\mathbf{x}_i | \theta_k, y_i = k)p(\theta_k, y_i = k)$

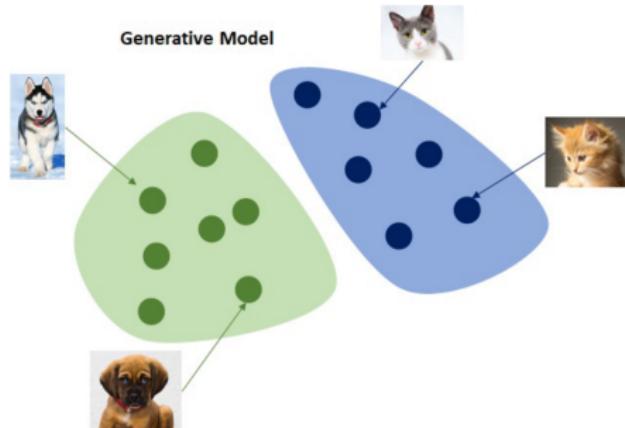
- Ex TME 3 : $p(\theta_k, y_i = k) = p(\theta_k)p(y_i = k)$ et $p(y_i = k) = p_k = \frac{N_k}{N}$
- N.B : un prior sur y_k et sur θ_k

Rappel sur les modèles génératifs (2)

Apprentissage : θ_k seulement pour la classe k

- $p(\mathbf{X}, \theta_k) = \prod_{i=1}^{N_k} p(\mathbf{x}_i | \theta_k) p(\theta_k) p_k$

- **Estimation de densité**

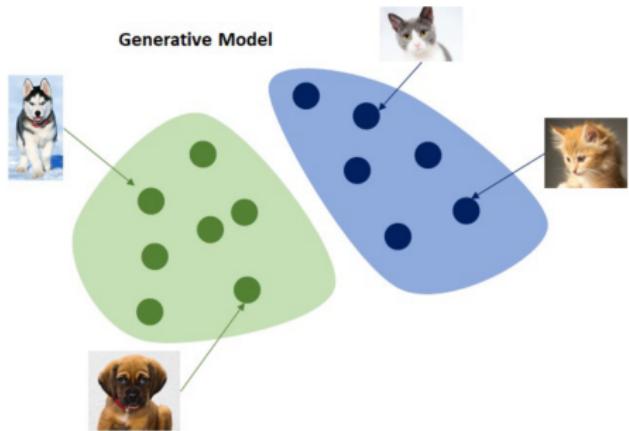


- MV : $\theta_k^* = \arg \max_{\theta_k} p(\mathbf{X} | \theta_k) = \arg \max_{\theta_k} \prod_{i=1}^{N_k} p(\mathbf{x}_i | \theta_k)$
- MAP $\theta_k^* = \arg \max_{\theta_k} p(\theta_k | \mathbf{X}) = \arg \max_{\theta_k} \prod_{i=1}^{N_k} p(\mathbf{x}_i | \theta_k) p(\theta_k)$
 - Choix d'un modèle pour $p(\mathbf{x}_i | \theta_k)$ (e.g. Gaussienne) et $p(\theta_k)$
 - Résolution : analytique : $\frac{\partial \mathcal{L}(\mathbf{X}, \theta)}{\partial \theta} = 0$ ou approchée : EM, gradient...

Rappel sur les modèles génératifs (3)

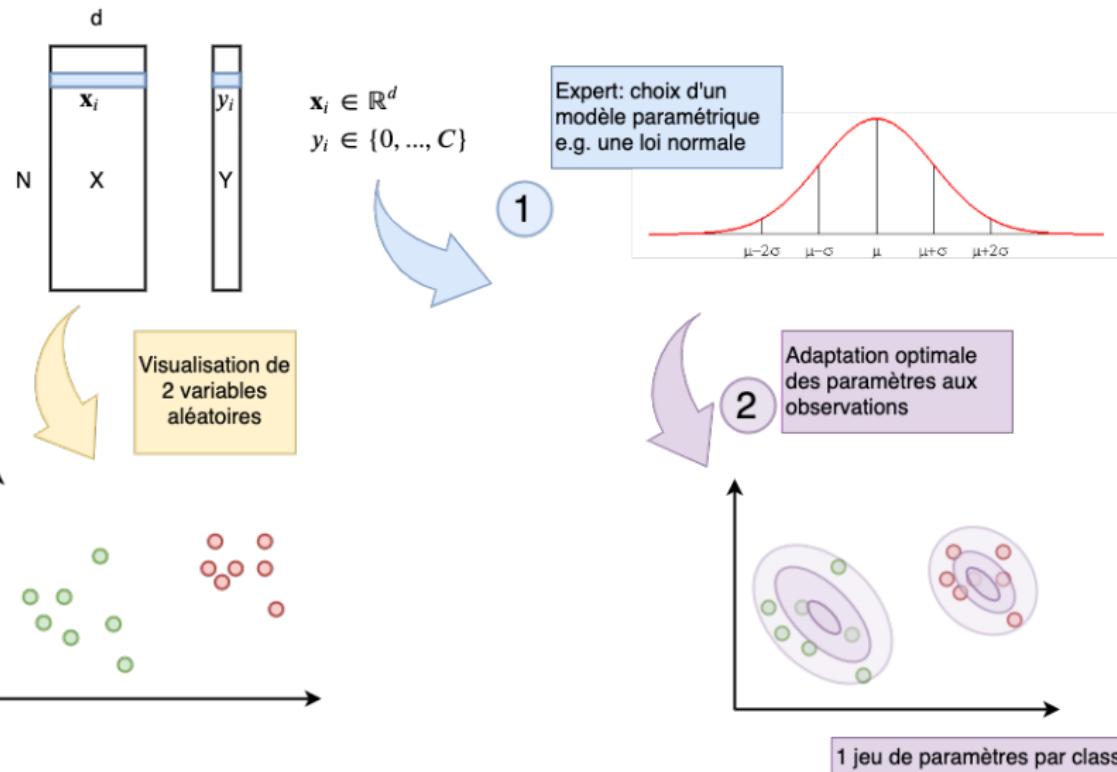
Inférence : $p(\mathbf{x}_i, y_k, \theta_k) = p(\mathbf{x}_i|y_k, \theta_k)p(y_k, \theta_k)$

- $p(y_k, \theta_k|\mathbf{x}_i) = \frac{p(\mathbf{x}_i|\theta_k)p(y_k, \theta_k)}{p(\mathbf{x}_i)}$
- $\hat{y} = \arg \max_k p(\mathbf{x}_i|\theta_k)p(y_k, \theta_k)$
- Ex : $\hat{y} = \arg \max_k p(\mathbf{x}_i|\theta_k)p_k$

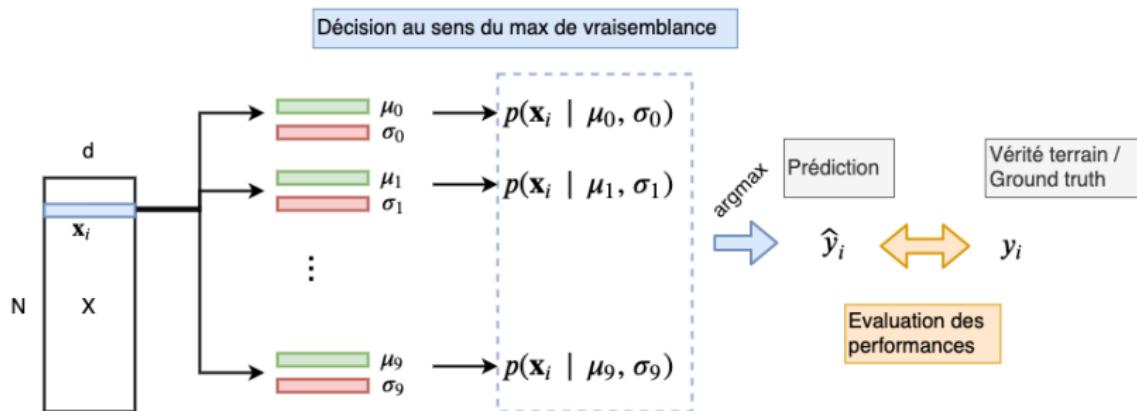


- Application bis en génération : $\tilde{x} \sim p_k p(\mathbf{x}|\theta_k) p(\theta_k)$

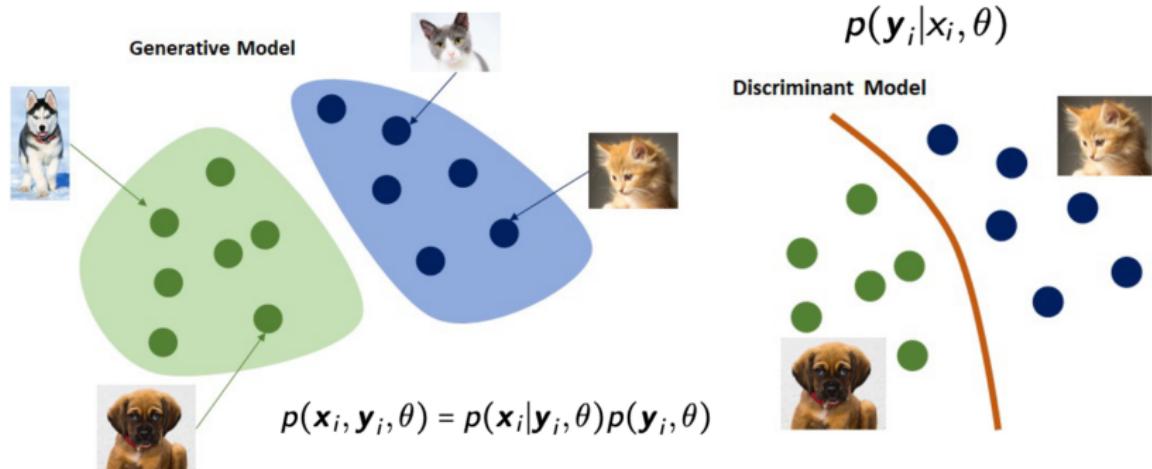
Modèles génératifs



Modèles génératifs

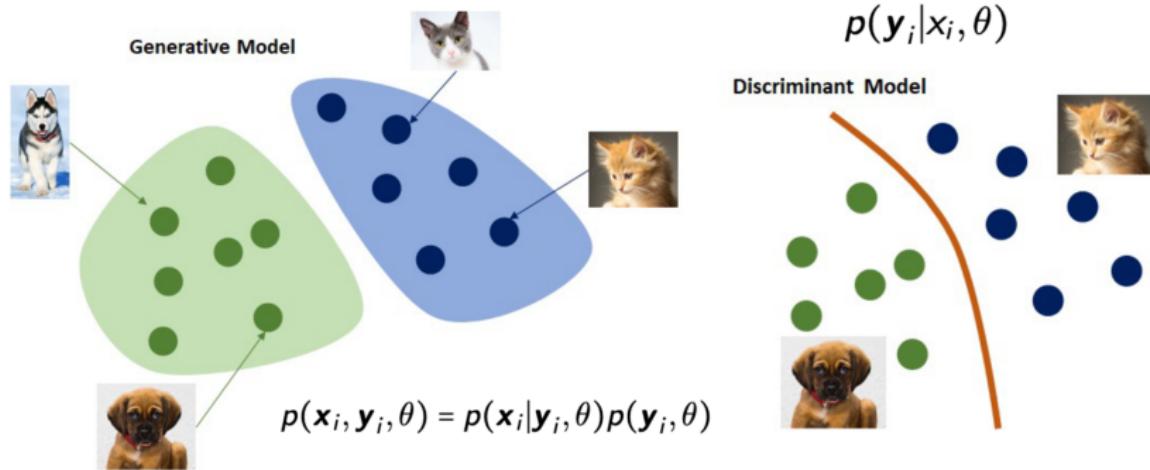


Modèle génératifs vs discriminatifs



- Approche **générative** : par classe
 $p(\mathbf{x}_i, y_k, \theta_k) = [p(\mathbf{x}_i | y_k, \theta_k)] p(y_k, \theta_k)$
- Quel modèle colle le mieux à mon observation ?
- Approche **discriminative** :
 $p(\mathbf{x}_i, y_k, \theta_k) = [p(y_k | \mathbf{x}_i, \theta_k)] p(\mathbf{x}_i, \theta_k)$
- Classe k $p(y_k | \mathbf{x}_i, \theta_k)$ vs classe $k l$ $p(y_l | \mathbf{x}_i, \theta_l)$: qu'est ce qui distingue la classe k de la classe l ?

Limite du modèle génératif pour la classification



- En **génératif** : $p(\mathbf{x}_i | \theta_k)$, \mathbf{x}_i avec y_k
- Moins lié à la classification, plus complexe, hypothèses sur $p(\mathbf{x}_i | \theta_k)$
- Robustesse aux outliers
- En **discriminatif** : $p(y_k | \mathbf{x}_i, \theta_k) \forall k$
- Directement lié à la tâche de classification finale, plus simple
- Meilleurs performances en général

⇒ Modèle discriminatif le plus connu : **Régression logistique**

Régression logistique

- En fait : $\mathcal{D} := (\mathbf{X}, \mathbf{Y}) = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$
- Régression logistique binaire : 2 classes, $Y = 0$ ou $Y = 1$.
- Comment modéliser $p(\mathbf{Y}|\mathbf{X}, \theta)$?

Régression logistique

- En fait : $\mathcal{D} := (\mathbf{X}, \mathbf{Y}) = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$
- Régression logistique binaire : 2 classes, $Y = 0$ ou $Y = 1$.
- Comment modéliser $p(\mathbf{Y}|\mathbf{X}, \theta)$?
 - On repère une variable de Bernoulli
 $p(Y = 1|X = \mathbf{x}, \theta) = 1 - p(Y = 0|X = \mathbf{x}, \theta)$
 - On choisit de modéliser *arbitrairement* avec $\theta := (\mathbf{w}, b)$:

$$p(Y = 1|X = \mathbf{x}, \mathbf{w}, b) =$$

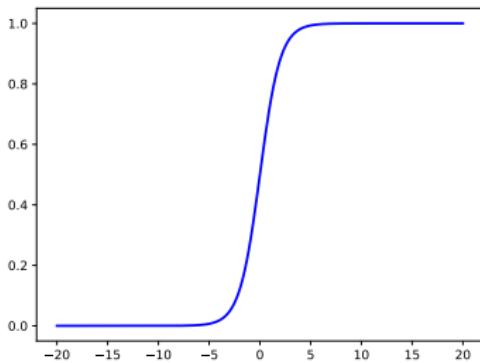
$$f_{\mathbf{w}, b}(\mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{w}^T \mathbf{x} + b))}$$

Régression logistique

- En fait : $\mathcal{D} := (\mathbf{X}, \mathbf{Y}) = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$
- Régression logistique binaire : 2 classes, $Y = 0$ ou $Y = 1$.
- Comment modéliser $p(\mathbf{Y}|\mathbf{X}, \theta)$?
 - On repère une variable de Bernoulli
 $p(Y = 1|X = \mathbf{x}, \theta) = 1 - p(Y = 0|X = \mathbf{x}, \theta)$
 - On choisit de modéliser *arbitrairement* avec $\theta := (\mathbf{w}, b)$:

$$p(Y = 1|X = \mathbf{x}, \mathbf{w}, b) =$$

$$f_{\mathbf{w}, b}(\mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{w}^T \mathbf{x} + b))}$$



- Données : $\mathcal{D} = (\mathbf{X}, \mathbf{Y})$
- Modèle : $p(Y = 1 | X = \mathbf{x}, \mathbf{w}, b) = f_{\mathbf{w}, b}(\mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{w}^T \mathbf{x} + b))}$
- Bornes du modèle :
 - $\lim_{\mathbf{w}^T \mathbf{x} + b \rightarrow -\infty} f(\mathbf{x}) = 0$
 - $\lim_{\mathbf{w}^T \mathbf{x} + b \rightarrow \infty} f(\mathbf{x}) = 1$

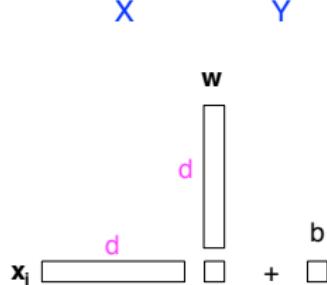
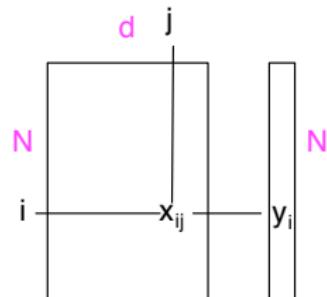
Dimension des éléments en présence

- Données : $D = (\mathbf{X}, \mathbf{Y})$

- Modèle : $p(Y=1|X=\mathbf{x}, \mathbf{w}, b) = f_{\mathbf{w}, b}(\mathbf{x}) = \frac{1}{1+\exp(-(\mathbf{w}^T \mathbf{x} + b))}$

- Bornes du modèle :

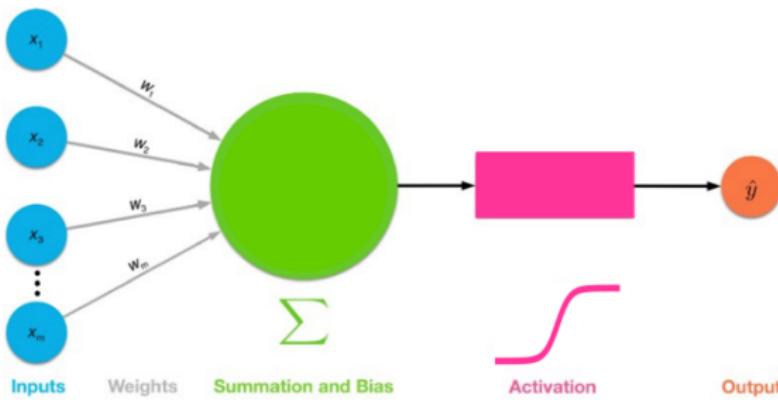
- $\lim_{\mathbf{w}^T \mathbf{x} + b \rightarrow -\infty} f(\mathbf{x}) = 0$
- $\lim_{\mathbf{w}^T \mathbf{x} + b \rightarrow \infty} f(\mathbf{x}) = 1$
- Si : $\mathbf{w}^T \mathbf{x} + b = 0 \Rightarrow f(\mathbf{x}) = 0.5$



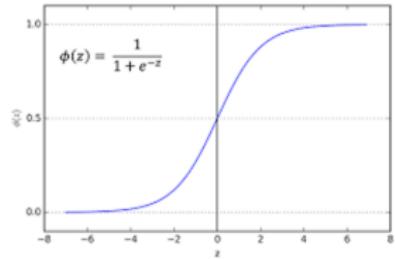
Régression logistique & neurone formel

Régression logistique = neurone formel

- ① Projection linéaire : $s = \mathbf{w}^T \mathbf{x} + b$ comme dans perceptron, SVM
 - Entraînement différent (voir ci-après)
- ② Fonction non-linéaire : sigmoïde : $\hat{y} = \sigma(s) = \frac{1}{1+e^{-as}}$
 - Interprétation probabiliste : $\hat{y} \in [0, 1]$ et estime $p(Y = 1 | X = \mathbf{x})$



$$\hat{y} = \frac{1}{1+e^{-a(\mathbf{w}^T \mathbf{x} + b)}}$$



Comment trouver les w^* ?

Comment trouver les \mathbf{w}^* ?

⇒ Par maximum de vraisemblance (conditionnelle) !

Vraisemblance (conditionnelle) -indépendance entre échantillons-

$$L = \prod_{i=1}^N p(Y = y_i | X = \mathbf{x}_i)$$

- Truc de Bernoulli :

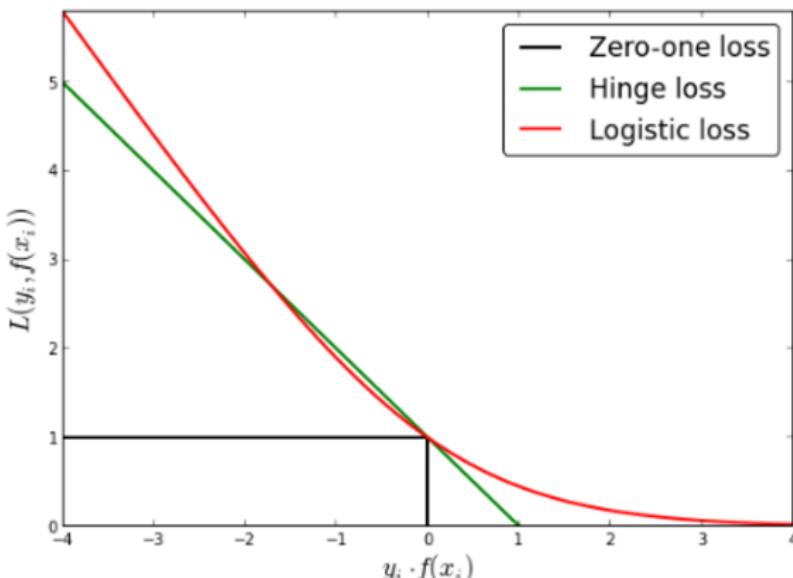
$$p(Y = y_i | X = \mathbf{x}_i) = p(Y = 1 | X = \mathbf{x}_i)^{y_i} (1 - p(Y = 1 | X = \mathbf{x}_i))^{1-y_i}$$

- Passage au log
- Remplacement des $p(Y = 1 | X = \mathbf{x})$ par des $f(\mathbf{x})$
- Nouvelle formulation :

$$\mathbf{w}^*, b^* = \arg \max_{\mathbf{w}, b} \sum_{i=1}^N [y_i \log(f(\mathbf{x}_i)) + (1 - y_i) \log(1 - f(\mathbf{x}_i))]$$

Régression logistique et 0/1 loss

- Régression logistique $\hat{y} = \sigma(s) = \frac{1}{1+e^{-(\mathbf{w}^T \mathbf{x} + b)}}$
- $y \in \{-1; +1\}$ et $p(Y = y | \mathbf{x})$ estimé par $\frac{1}{1+e^{-y(\mathbf{w}^T \mathbf{x} + b)}}$ ($\sigma(-x) = 1 - \sigma(x)$)
- **Logistic loss :** $-\log [p(Y = y | \mathbf{x})] = \log \left[1 + e^{-y(\mathbf{w}^T \mathbf{x} + b)} \right]$
 - Borne supérieure convexe à la 0/1 loss



Données : $D = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$

$$\mathbf{w}^*, b^* = \arg \max_{\mathbf{w}, b} \sum_{i=1}^N [y_i \log(f(\mathbf{x}_i)) + (1 - y_i) \log(1 - f(\mathbf{x}_i))]$$

On remplace $f(x)$ par sa valeur et on développe le coût...

... [quelques lignes de calcul] ...

Résolution

Données : $D = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$

$$\mathbf{w}^*, b^* = \arg \max_{\mathbf{w}, b} \sum_{i=1}^N [y_i \log(f(\mathbf{x}_i)) + (1 - y_i) \log(1 - f(\mathbf{x}_i))]$$

On remplace $f(x)$ par sa valeur et on développe le coût...

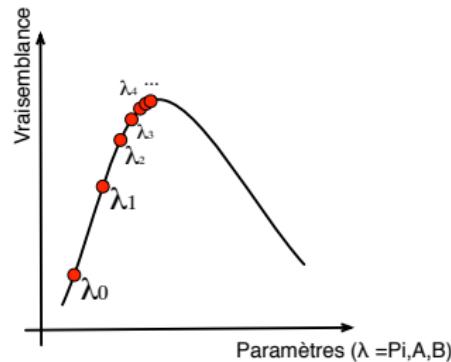
... [quelques lignes de calcul] ...

$$\frac{\partial}{\partial w_j} L_{\text{log}} = \sum_{i=1}^N x_{ij} \left(y_i - \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x}_i + b))} \right)$$

$$\frac{\partial}{\partial b} L_{\text{log}} = \sum_{i=1}^N \left(y_i - \frac{1}{1 + \exp(-(\mathbf{w} \cdot \mathbf{x}_i + b))} \right)$$

- Annulation directe du gradient **impossible**
- ⇒ Algorithme itératif :
 - Init des paramètres w_0, b_0
 - Tant que convergence non atteinte
 - Calcul des : $\frac{\partial L_{\log}}{\partial b}, \frac{\partial L_{\log}}{\partial w_j}$
 - Mise à jour (montée de gradient) : $\theta^t = \theta^{t-1} + \epsilon \frac{\partial L_{\log}}{\partial \theta}$

Cas convexe :



USPS : génératif VS discriminant

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

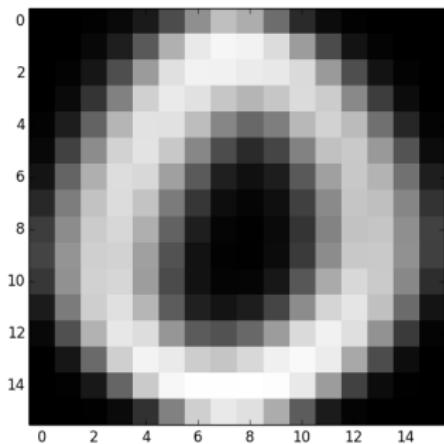
0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

USPS : génératif VS discriminant

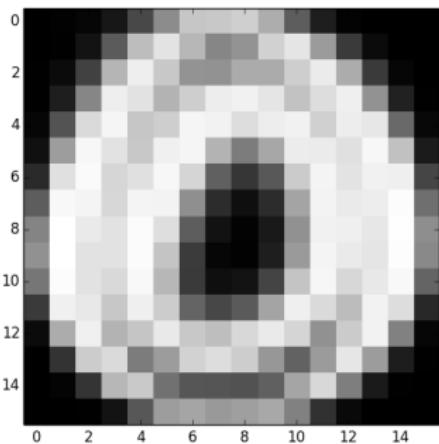
Modèle génératif gaussien : **classe 0**

Visualisation de la moyenne de la classe :



$\mu + \text{reshape}$

Visualisation de la variance de la classe :

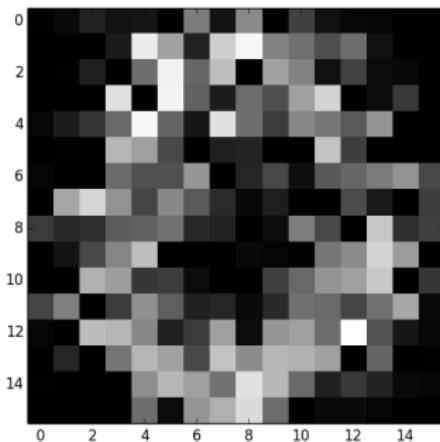


$\sigma + \text{reshape}$

USPS : génératif VS discriminant

Possibilité de générer un échantillon :

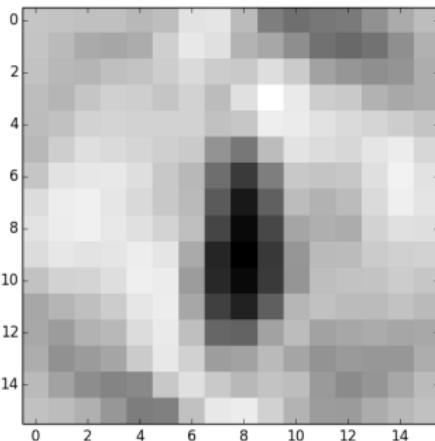
Tirage d'une valeur gaussienne pour chaque pixel...



Mais hypothèse d'indépendance des pixels \Rightarrow qualité BOF

USPS : génératif VS discriminant

En régression logistique : classe 0 VS toutes les autres



Mise en évidence des zones qui ne sont utilisées **que** par les 0

- Apprentissage : 6229 images
- Test : 3069 images
- Naive Bayes (modèle de pixel Bernouilli, cf TME 3)
 - Taux bonne classif. en test : 85.3%
- Regression logistique
 - Taux bonne classif. en test : 93.6%

Régression logistique Bayésienne

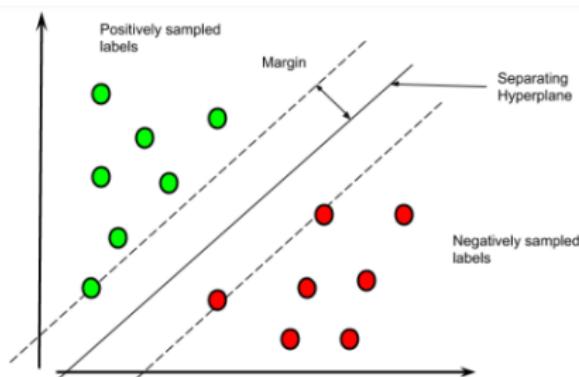
$$p(\mathbf{w}|\mathbf{X}, \mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})$$

- **Prior Gaussien :** $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \mathbf{0}, \sigma_0^2 \mathbf{I}) \propto e^{-\frac{\|\mathbf{w}^2\|}{2\sigma_0^2}}$

$$-\log [p(\mathbf{w}|\mathbf{X}, \mathbf{Y})] = -\sum_{i=1}^N [y_i \log(f(\mathbf{x}_i)) + (1 - y_i) \log(1 - f(\mathbf{x}_i))] + \frac{\|\mathbf{w}^2\|}{2\sigma_0^2}$$

- **Prior Gaussien \leftrightarrow régularisation ℓ_2 sur \mathbf{w} ~ SVM (max margin)**

- $\lambda = \frac{1}{2\sigma_0^2}$ hyper-paramètre de régularisation



Régression logistique Bayésienne

$$p(\mathbf{w}|\mathbf{X}, \mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})$$

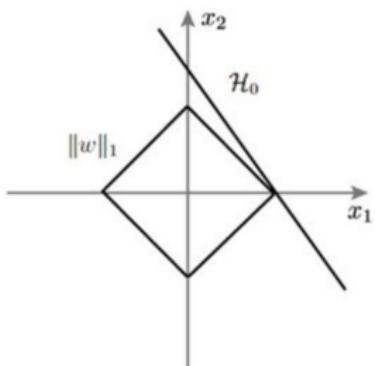
- **Prior Laplace :** $p(\mathbf{w}) = \propto e^{-\frac{\|\mathbf{w}\|_1}{b}}$, avec $\|\mathbf{w}\|_1 = \sum_{i=1}^d |w_i|$

$$-\log [p(\mathbf{w}|\mathbf{X}, \mathbf{Y})] = -\sum_{i=1}^N [y_i \log(f(\mathbf{x}_i)) + (1 - y_i) \log(1 - f(\mathbf{x}_i))] + \frac{\|\mathbf{w}\|_1}{b}$$

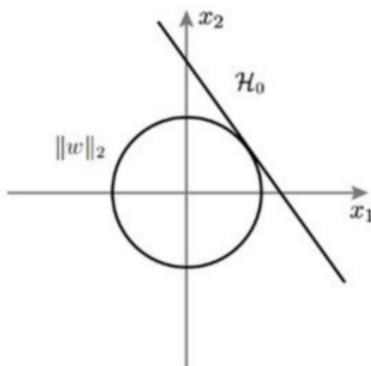
- **Prior Laplace \leftrightarrow régularisation ℓ_1 sur \mathbf{w}** \Rightarrow solution sparse

- $\lambda = \frac{1}{b}$ hyper-paramètre de régularisation

A L1 regularization

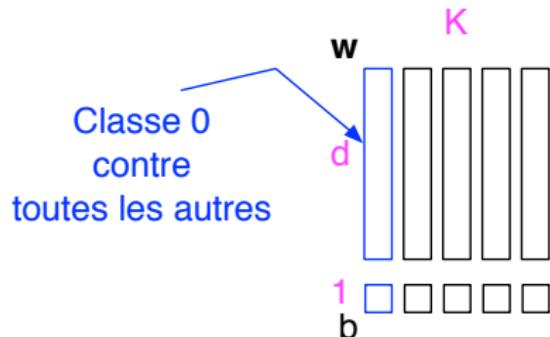


B L2 regularization



Comment passer au multi-classes ?

un contre tous (*one against all*) :
 K classes $\Rightarrow K$ classifieurs appris
séparément sur toutes les données



- $f(\mathbf{x}) \Rightarrow f_k(\mathbf{x})$ et critère de décision :

$$k^* = \arg \max_k f_k(\mathbf{x})$$

Quelle classe veut **le plus** l'échantillon \mathbf{x} ?

- Critères de rejet :
 - pas de $f_k(\mathbf{x}) > 0.5$
 - plusieurs $f_k(\mathbf{x}) > 0.5$

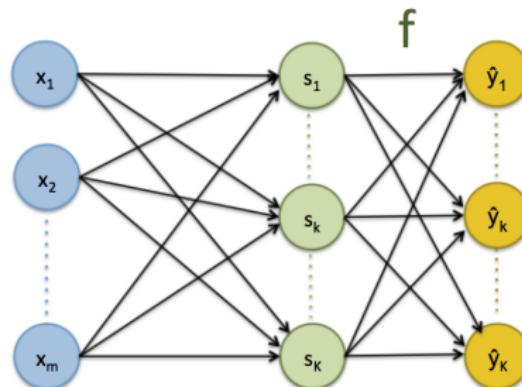
Régression logistique multiclasse

- Passage au "vrai" multiclasse $y \in \llbracket 1; K \rrbracket$
- Régression logistique multi-classe : projection linéaire $\rightarrow K$ + softmax

$$\hat{y}_k = \frac{e^{\mathbf{w}_k^T \mathbf{x} + b_k}}{\sum_{k'=1}^K e^{\mathbf{w}_{k'}^T \mathbf{x} + b_{k'}}}$$

- Interprétation probabiliste : \hat{y}_k estime $P(y_k | \mathbf{x}, \mathbf{w}_k, b_k)$

- $\hat{y}_k \in [0, 1]$ et $\sum_{i=1}^K \hat{y}_k = 1$



Régression logistique multiclasse

- Apprentissage :

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = -\log [P(y_k | \mathbf{x}_i, \mathbf{w}_k, b_k)] = \mathbf{w}_k^T \mathbf{x}_i + b_k - \log \left[\sum_{k'=1}^K e^{\mathbf{w}_{k'}^T \mathbf{x}_i + b_{k'}} \right]$$

- $\mathcal{L}(\mathbf{W}, \mathbf{b})$ convexe, borne sup à 0/1 loss
- On peut montrer que :

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \mathbf{x}_i^T (\hat{y}_k - y_k)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = (\hat{y}_k - y_k)$$

On considère un examen, pour lequel un étudiant s peut répondre à la question q correctement, ce qui est noté $x_{qs} = 1$ ou incorrectement, $x_{qs} = 0$. On suppose que la chance de succès dépend de la capacité de l'étudiant α_s et de la difficulté de la question δ_q . On postule le modèle de réponse suivant :

$$p(x_{qs} = 1 | \alpha_s, \delta_q) = \sigma(\alpha_s - \delta_q), \quad \text{avec : } \sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$p(x_{qs} = 1 | \alpha_s, \delta_q) = \sigma(\alpha_s - \delta_q), \quad \text{avec : } \sigma(x) = \frac{1}{1 + \exp(-x)}$$

- ➊ Etudier rapidement la fonction σ et conclure qu'elle permet effectivement de modéliser une probabilité.
- ➋ A quelle condition un étudiant s a-t-il autant de chance répondre correctement que de se tromper à une question q ?
- ➌ Quelle loi permet de modéliser la variable x_{qs} ? Montrer que :

$$p(x_{qs} | \alpha_s, \delta_q) = \sigma(\alpha_s - \delta_q)^{x_{qs}} (1 - \sigma(\alpha_s - \delta_q))^{(1-x_{qs})}$$

où x_{qs} peut prendre les valeurs 0 ou 1.

- Limites du modèle proposé :
un exercice est simple ou dur dans l'absolu, un étudiant est fort ou faible toujours en absolu...
- Trouver une approche plus riche :

- Limites du modèle proposé :
un exercice est simple ou dur dans l'absolu, un étudiant est fort ou faible toujours en absolu...
- Trouver une approche plus riche :

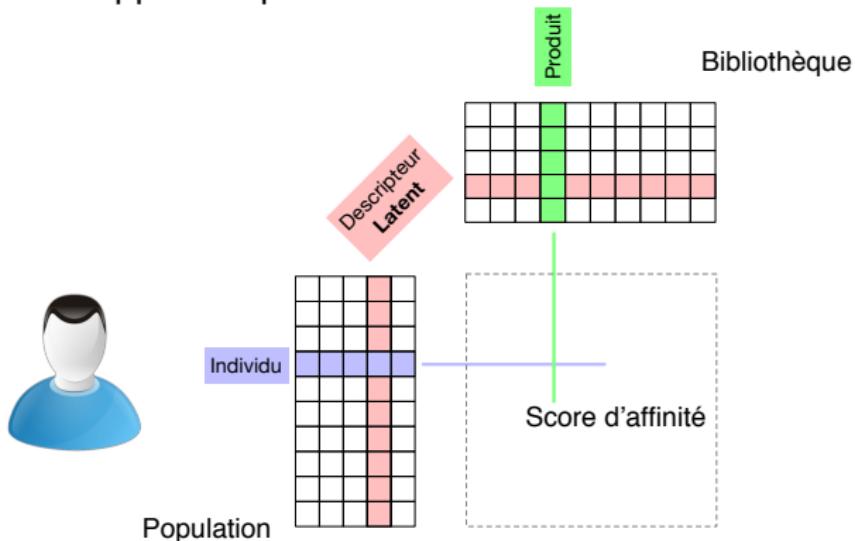
$$\mathbf{s}, \mathbf{q} \in \mathbb{R}^d, \quad p(x_{qs} = 1 | \mathbf{s}, \mathbf{q}) = \frac{1}{1 + \exp(-\mathbf{s} \cdot \mathbf{q})}$$

Travail dans un espace vectoriel :

- un match sur une dimension $\Rightarrow \mathbf{s} \cdot \mathbf{q} \nearrow$
- vecteurs orthogonaux $\mathbf{s} = [0, 1, 0], \mathbf{q} = [1, 0, 1] \Rightarrow \mathbf{s} \cdot \mathbf{q} = 0$

Du cas jouet au cas réel

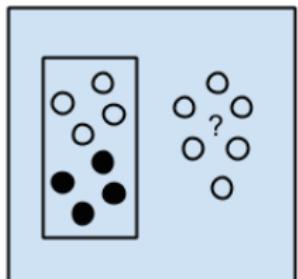
- Limites du modèle proposé :
un exercice est simple ou dur dans l'absolu, un étudiant est fort ou faible toujours en absolu...
- Trouver une approche plus riche :



- 1 Méthodes par coût
- 2 Méthodes probabilistes
- 3 MAPSI et le machine learning

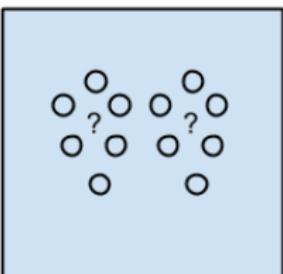
Différents cadres de machine learning

Supervisé



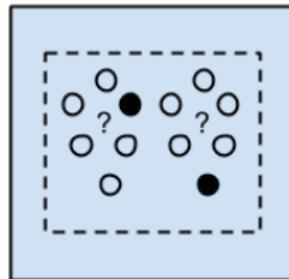
Supervised Learning
Algorithms

Non-supervisé



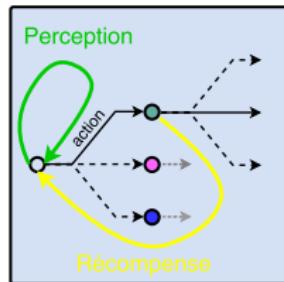
Unsupervised Learning
Algorithms

Semi-supervisé



Semi-supervised
Learning Algorithms

Renforcement

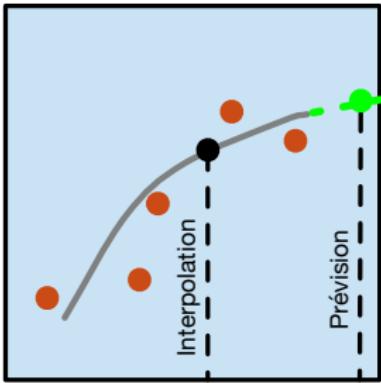


Jason Brownlee

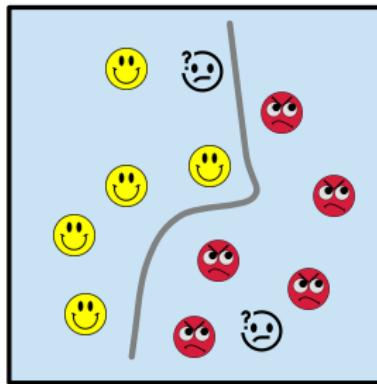
- Différents algorithmes...
- ... et différentes évaluations
- Différentes **données**, différents **coûts**...
Et une nouvelle donne avec *Amazon Mechanical Turk*

Grande familles de problématiques

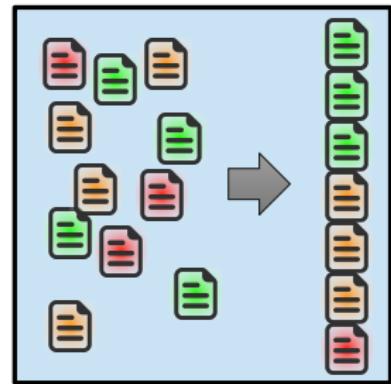
Régression



Classification



Ordonnancement



Les familles d'algorithmes (1/3)

Modèles génératifs

apprentissage Bayesien

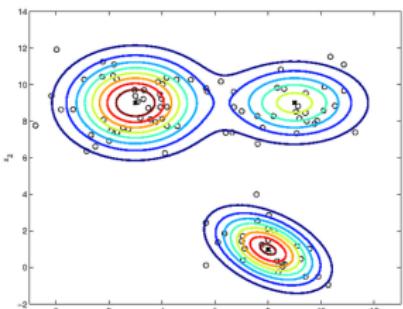
- 1 Choix d'un modèle paramétrique

e.g. mixture de 3 Gaussiennes

- 2 Apprentissage des paramètres

maximum de vraisemblance

- 3 Exploitation = inférence



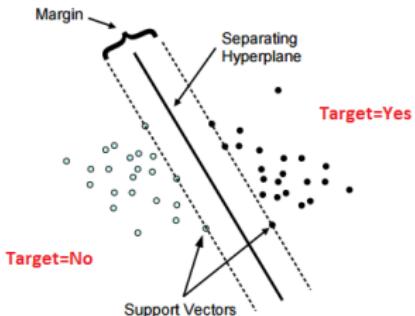
e.g. Mixtures, Naïve Bayes, MM, HMM...
Usages : extraction thématique, classification de spam

Modélisation discriminante (classification)

- 1 Apprentissage d'une frontière

i.e. chercher les différences dans les caractéristiques

- 2 Classification de nouveaux points

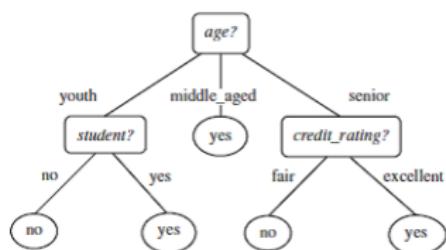


Support Vector Machine
e.g. Perceptron, SVM, Régression logistique,...
Usages : classification de signaux, de textes, ...

Les familles d'algorithmes (2/3)

Arbre de décision

- 1 Sélectionner un caractéristique
- 2 Couper
- 3 Décider

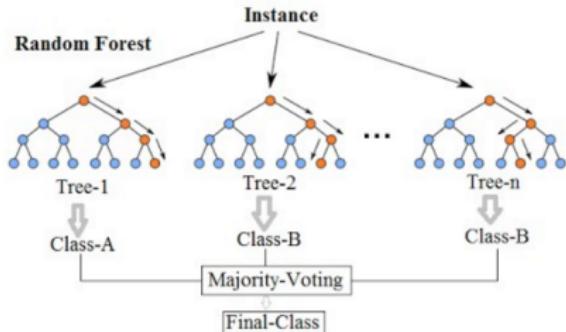


e.g. C4.5

Usages : besoin d'une décision expliquée

Approches ensemblistes

- 1 Multiplier les classifieurs
- 2 Fusionner les décisions



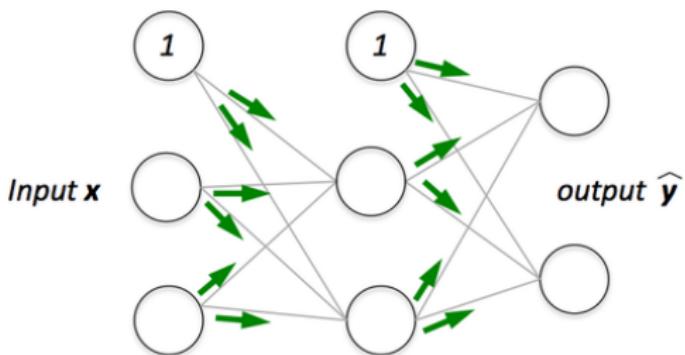
e.g. Random Forest, Boosting, Bagging

Usages : classification robuste, parallélisable

Les familles d'algorithmes (3/3)

Les réseaux de neurones

- 1 Au départ, un **opérateur paramétrique complexe...**
Et un algorithme d'apprentissage efficace.
- 2 une chaîne de traitements capable
d'extraire des caractéristiques pertinentes automatiquement

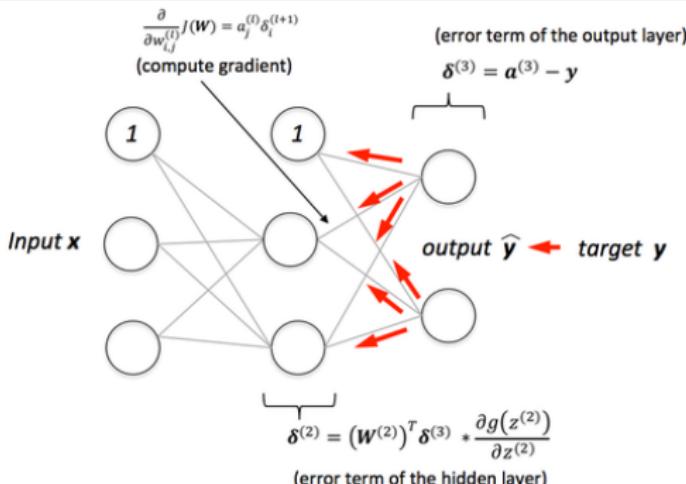


Sebastian Raschka

Les familles d'algorithmes (3/3)

Les réseaux de neurones

- 1 Au départ, un **opérateur paramétrique complexe...**
Et un algorithme d'apprentissage efficace.
- 2 une chaîne de traitements capable
d'extraire des caractéristiques pertinentes automatiquement

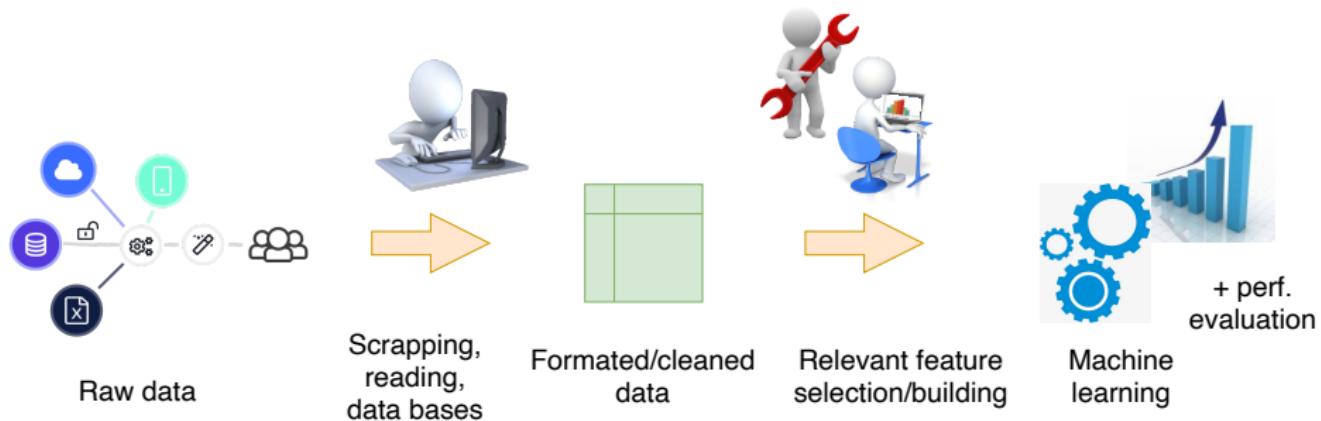


Sebastian Raschka

Chaine de traitements

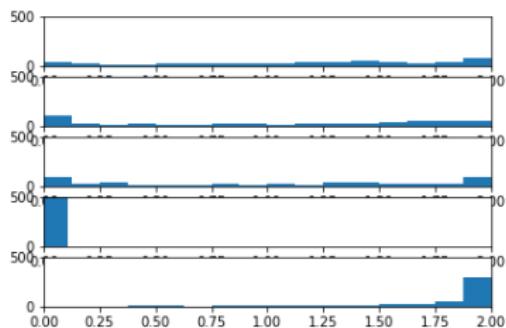
Toutes les étapes comptent...

Surtout les premières pour les performances !
Choisir & optimiser un modèle n'est qu'une partie du travail...



⇒ Valoriser vos compétences d'informaticien !

- Histogramme de répartition des classes
 - Critique pour la définition des *a priori*
- Histogramme de d'analyse des valeurs d'une variable
 - Comprendre les données
 - Choisir un modèle pour coller à cette variable en fonction de l'Histogramme
 - Faire marcher les arbres de décision –plus tard :)-



Distribution de différents pixels dans une base USPS

- Supervisé/non-supervisé
- Complétion de données manquante
- Classification / régression
- Compléter des données manquantes
(est-ce une classe de problème ?)

Exemple :

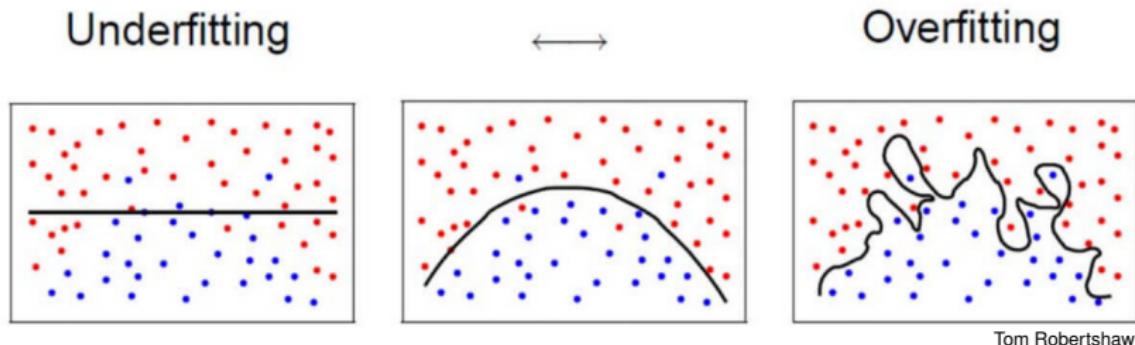
- Prédire les ventes de parfum lors des prochaines soldes
- Reconnaître qu'une image contient un chat

- Choisir une ou plusieurs loi de proba. pour les données
- Formuler la vraisemblance
- Optimiser la vraisemblance
 - Trouver les paramètres optimaux des lois usuelles sur wikipedia
 - Sinon, annuler la dérivée de la vraisemblance
 - Sinon, maximiser itérativement la vraisemblance
- Traiter de nouvelles données en inférence (les faire passer dans le modèle optimisé)

[MAPSI] Optimiser un modèle basé sur une fonction de coût

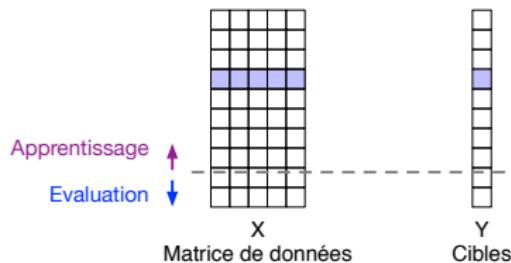
- Choisir une représentation des données, un modèle
- Choisir une fonction de cout
- Optimiser le cout
 - Annuler la dérivée du cout
 - Sinon, minimiser itérativement le cout
- Traiter de nouvelles données en inférence (les faire passer dans le modèle optimisé)

- **Apprendre** un modèle est aussi important que de **l'évaluer**
- **Apprendre et évaluer** sur les mêmes données est aberrant



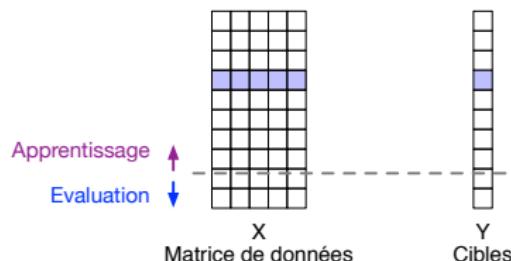
- **Apprendre** un modèle est aussi important que de **l'évaluer**
- **Apprendre et évaluer** sur les mêmes données est aberrant

Dilemme de répartition des données :

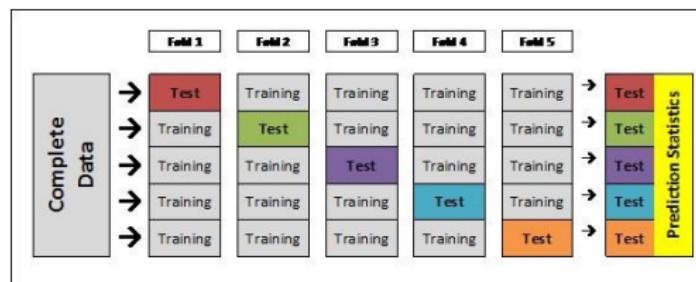


- **Apprendre** un modèle est aussi important que de **l'évaluer**
- **Apprendre et évaluer** sur les mêmes données est aberrant

Dilemme de répartition des données :



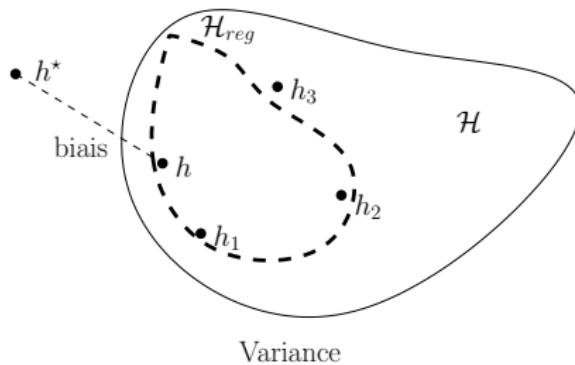
Solution = validation croisée



Golden Helix

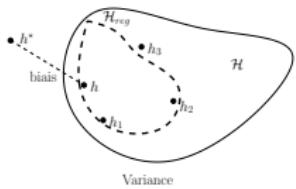
Mais pourquoi aller plus loin ? ??

Une proposition d'analyse avec le dilemme biais-variance



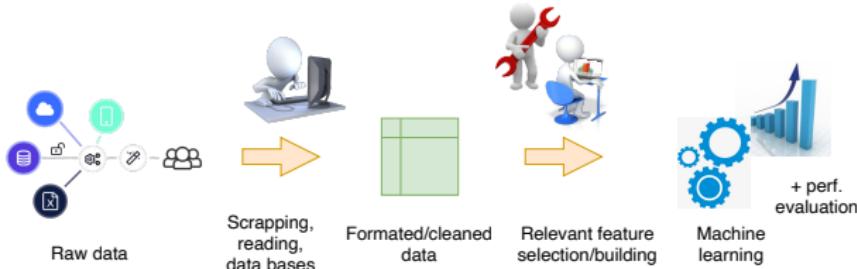
- Variance = taille de l'espace de recherche du modèle
 - Représentation des données + complexité du modèle (paramètres & hyper-paramètres)
- Biais = distance entre le meilleur modèle et le modèle retenu

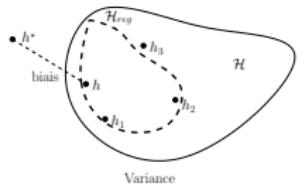
Différents paradigmes au fil des époques (1/4)



Exploiter seulement des informations pertinentes proposées/construites par un expert

- Efficace très vite
 - Peu de bruit = les modèles simples marchent bien
- ⇒ Approches très rentables... Que vous êtes déjà en mesure d'implémenter !





Générer [automatiquement] plein de caractéristique... Puis sélectionner celles qui sont utiles

A priori Critères de sélection/transformation de variables

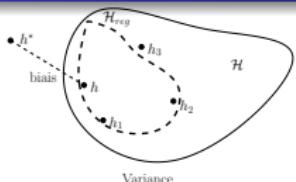
- ACP
- Sélection de variables sur critère de séparabilité des données

On Line Apprendre ce qui est utile ou pas = **régularisation** Soit des données $X \in \mathbb{R}^{N \times d}$ avec d très grand et une fonction de décision/régression linéaire $f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{w}$

$$\arg \min_{\mathbf{w}} \mathcal{C} + \lambda \Omega(\mathbf{w}), \Omega(\mathbf{w}) = \begin{cases} \sum_j w_j^2 & \text{régul. } L_2 \\ \sum_j |w_j| & \text{régul. } L_1 \end{cases}$$

$\Rightarrow \lambda$ devient un hyper-paramètre critique !

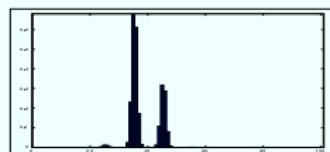
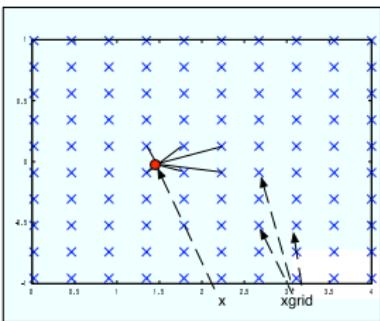
Différents paradigmes au fil des époques (3/4)



Les méthodes à noyaux et les espaces de représentation universel

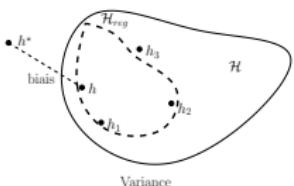
- Moins travailler sur les descripteurs = moins besoin d'expertise externe.
- Avoir des fonctions à la fois simple à apprendre [formulation convexe, régularisation] mais capable de traiter des cas complexe.

$$f(\mathbf{x}) = \sum_i w_i k(\mathbf{x}, \mathbf{x}_i), \text{ où } k \text{ est une fonction de similarité}$$



Représentation en histogramme des similarités gaussiennes entre un point (rond rouge) et tous les points de la grille

$$\phi(\mathbf{x})_j = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_{grid,j}\|^2}{2\sigma^2}\right)$$

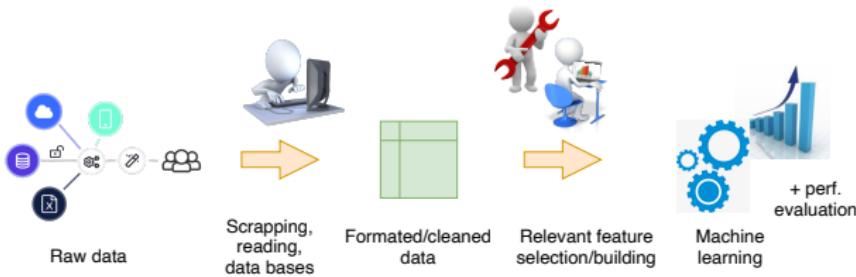


Créer des opérateurs super-complexes... Mais efficaces

● Réseaux de neurones

- La promesse d'une extraction de caractéristiques automatiques
- L'apprentissage de représentation et la sémantique

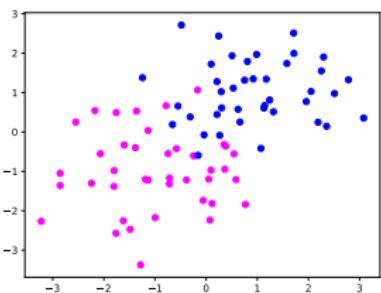
● XGBoost



Curse of dimensionality

A classical toy example to illustrate the curse of dimensionality :

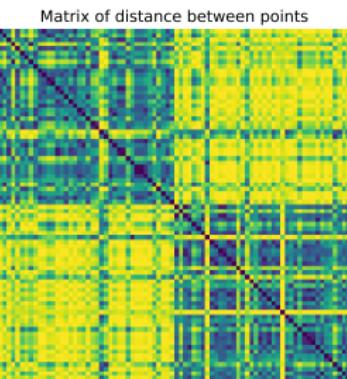
Original dataset :



Matrix view
Matrix of raw points



Distance matrix

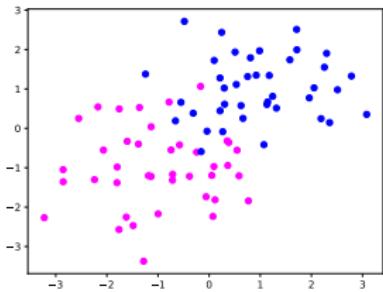


Easy problem / classes are clearly separated

Curse of dimensionality

A classical toy example to illustrate the curse of dimensionality :

Original dataset :



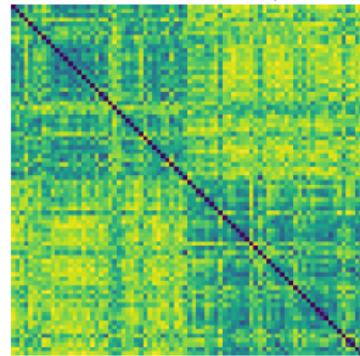
Matrix view

Matrix of raw points



Distance matrix

Matrix of distance between points

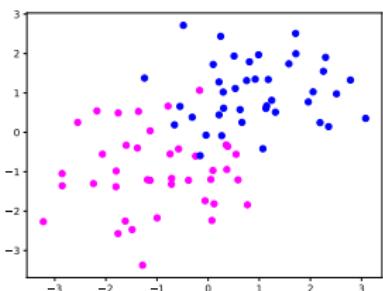


Adding some noisy dimensions in the dataset

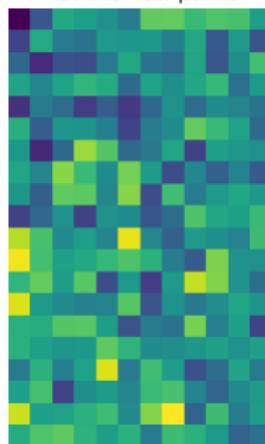
Curse of dimensionality

A classical toy example to illustrate the curse of dimensionality :

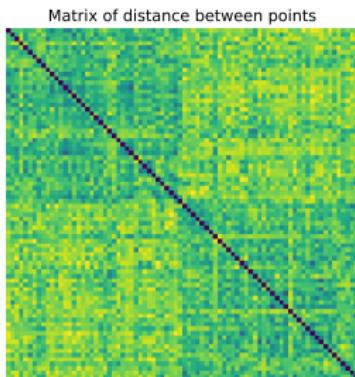
Original dataset :



Matrix view
Matrix of raw points



Distance matrix



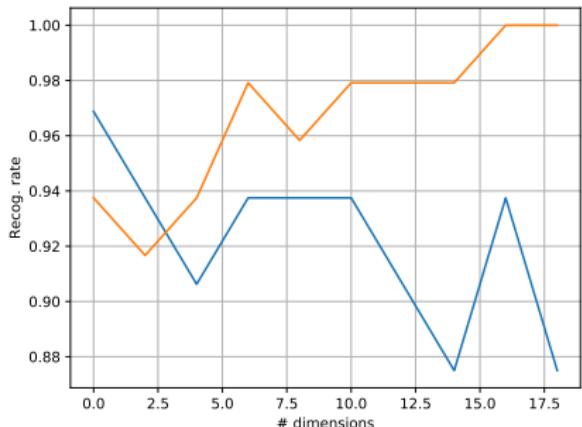
Adding **more** noisy dimensions in the dataset

⇒ Euclidian distance is very sensitive to the dimensionality issue

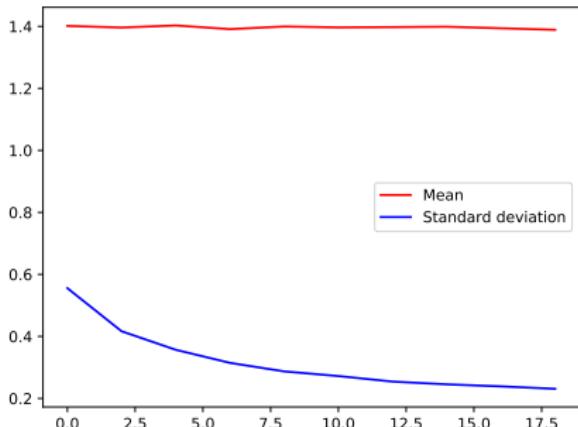
Curse of dimensionality

A classical toy example to illustrate the curse of dimensionality :

Basic classifier on those datasets



Distances between points in the dataset



⇒ Learn accuracy ↗, test accuracy ↘ = overfitting

⇒ All points tend to lay on an hypersphere (they become equidistant)