Apprentissage statistique : introduction

Gilles Gasso

INSA Rouen - Département ASI
Laboratoire LITIS

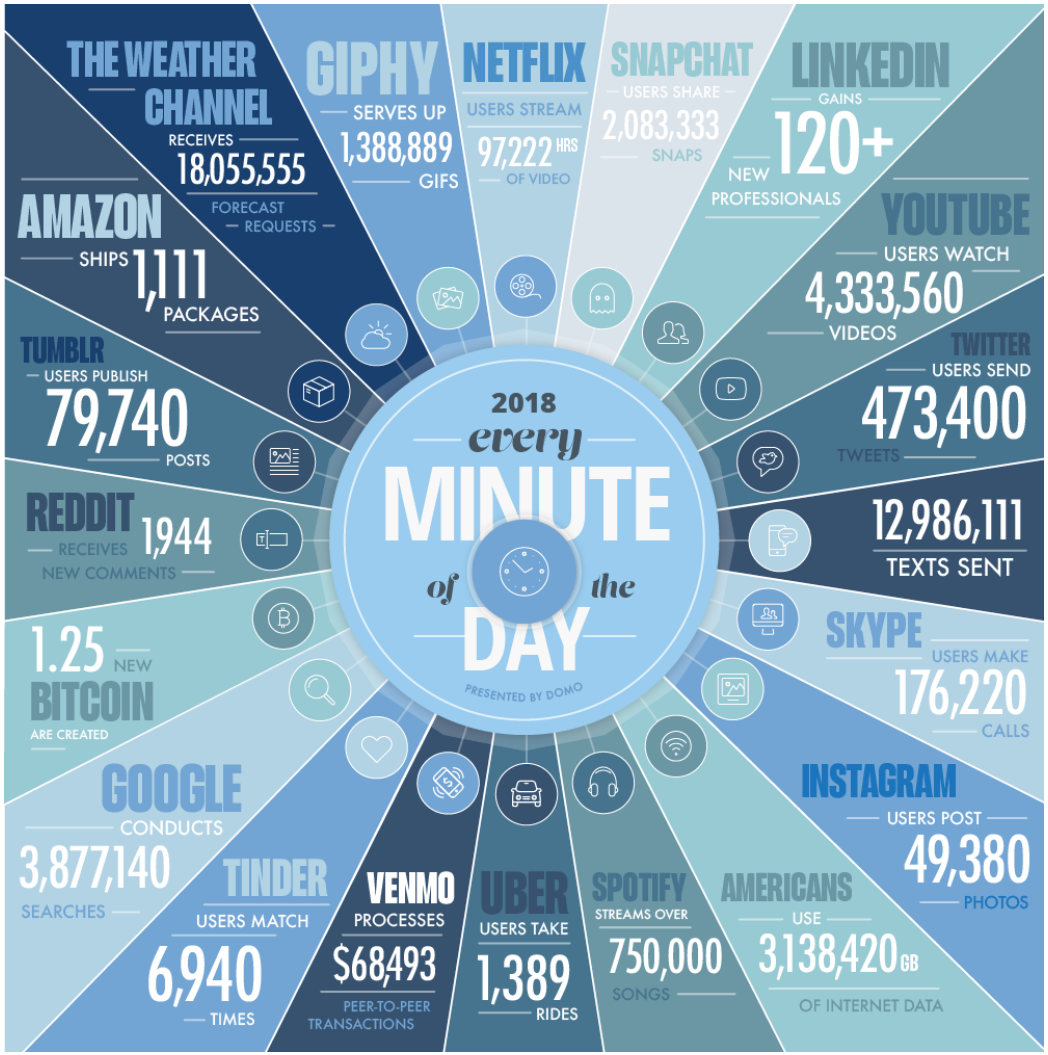
CIMPA Research School 2019

Antananarivo, Madagascar

10 août 2022

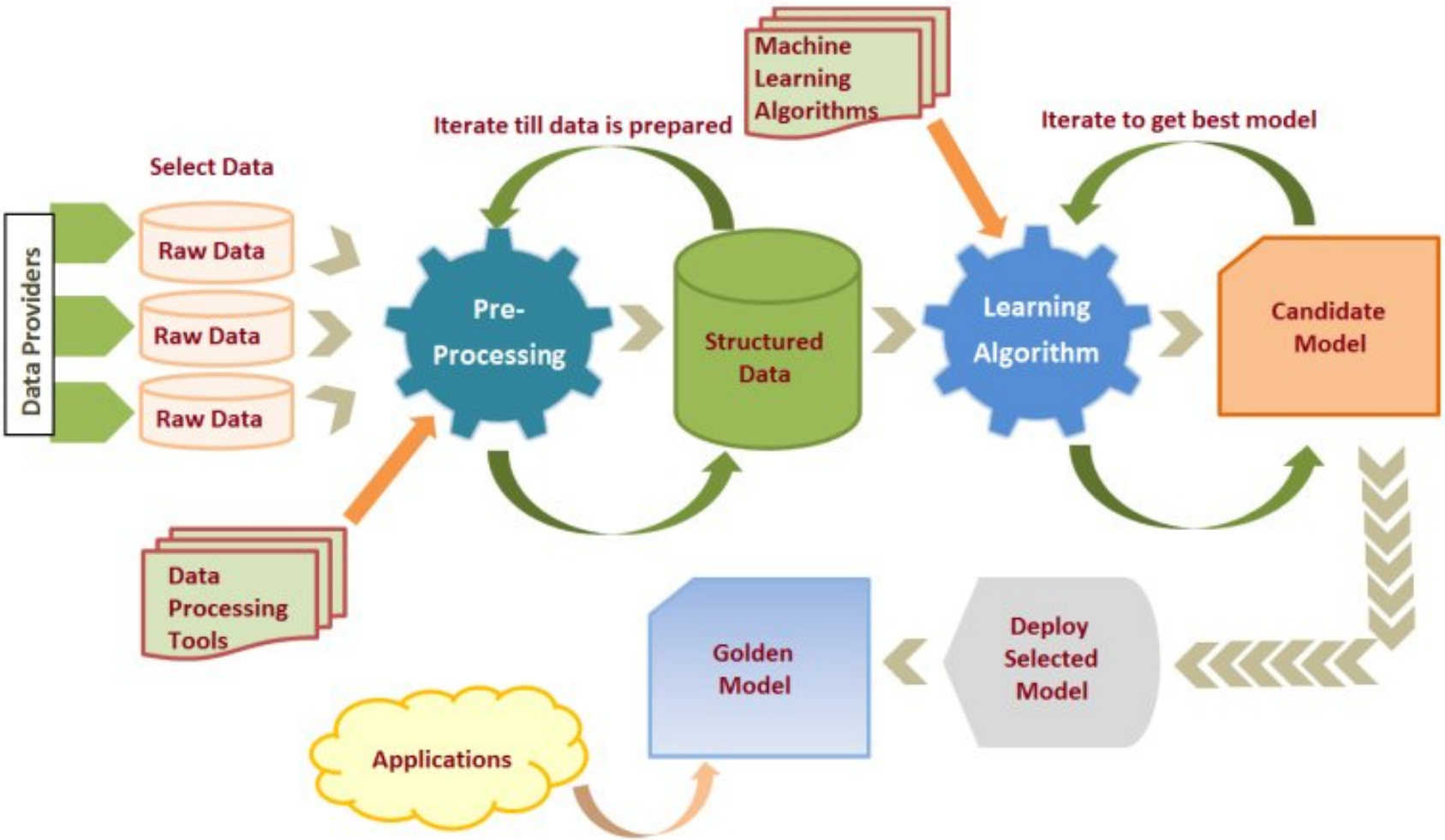
Plan

Des données ...

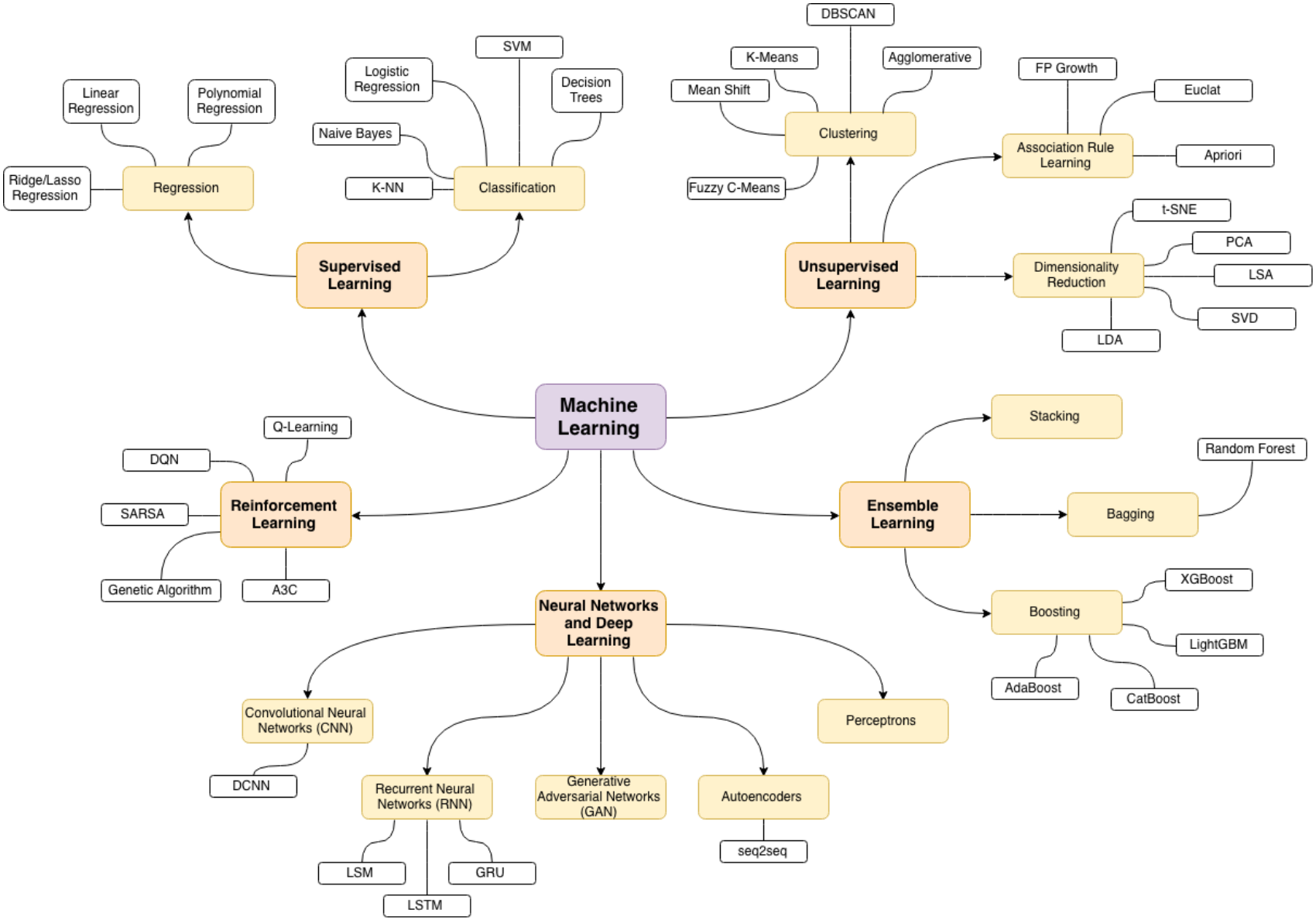


- Texte
- Audio
- Images
- Vidéos
- Graphes ...

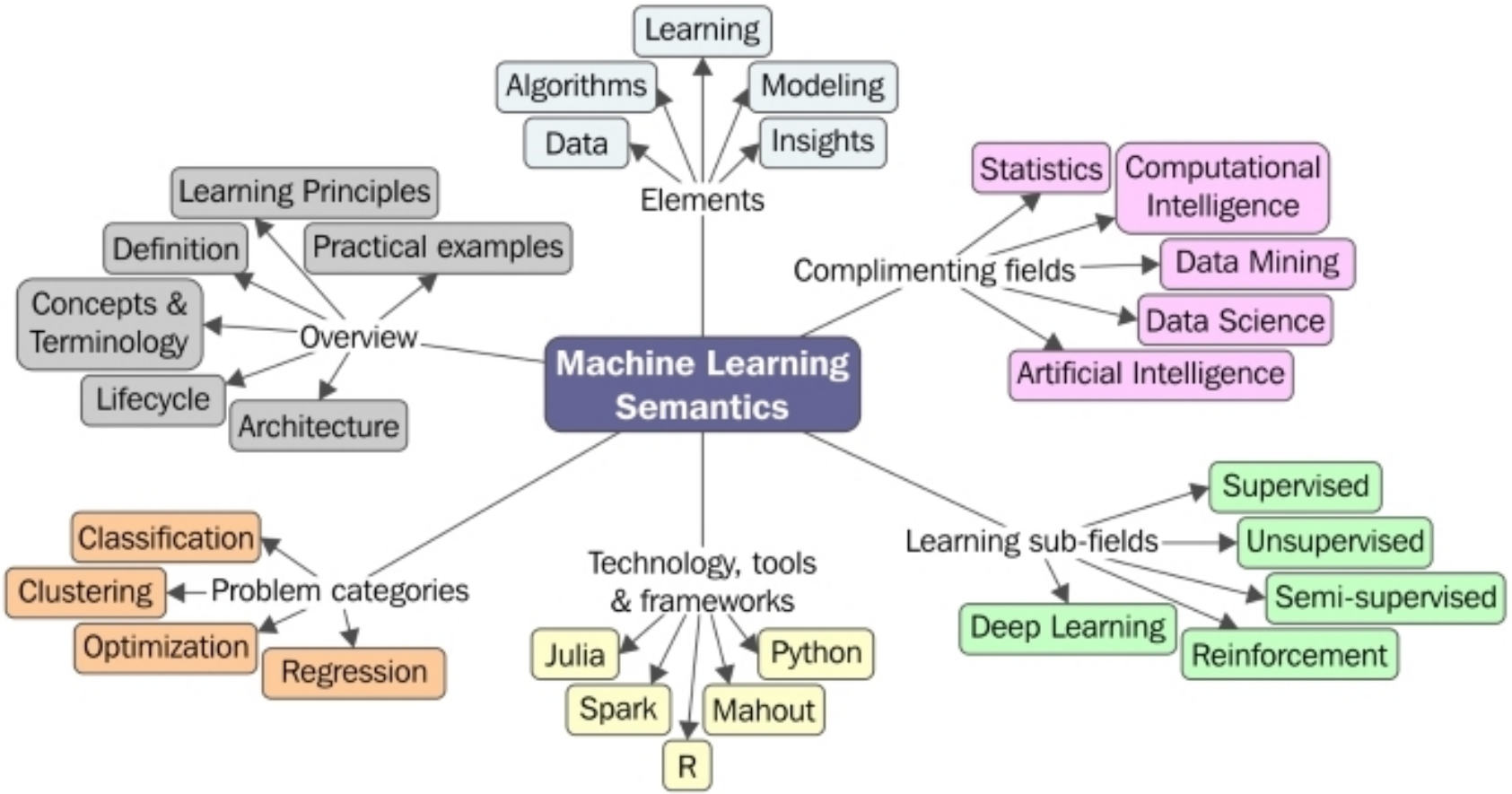
à leur traitement...



par des algorithmes (taxonomie)



par des algorithmes (taxonomie)

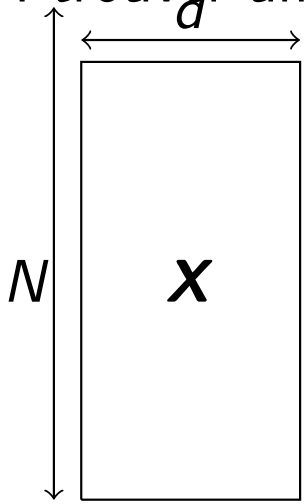


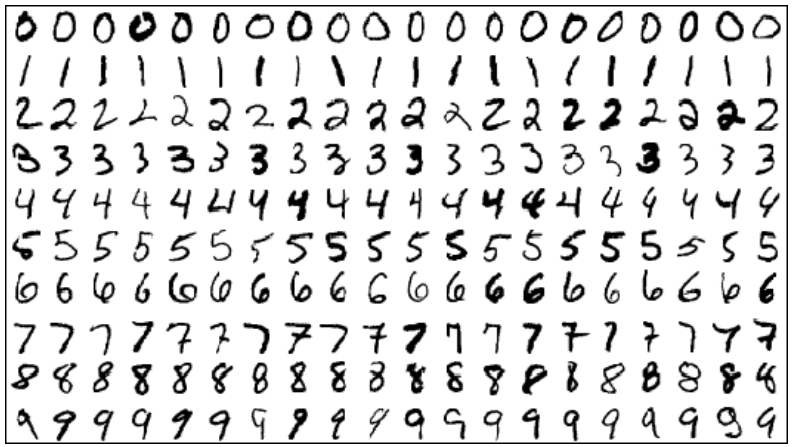
Objectifs du cours

Aperçu de quelques méthodes de machine learning

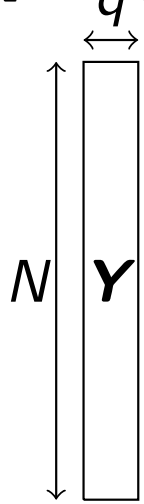
Réduction de dimension

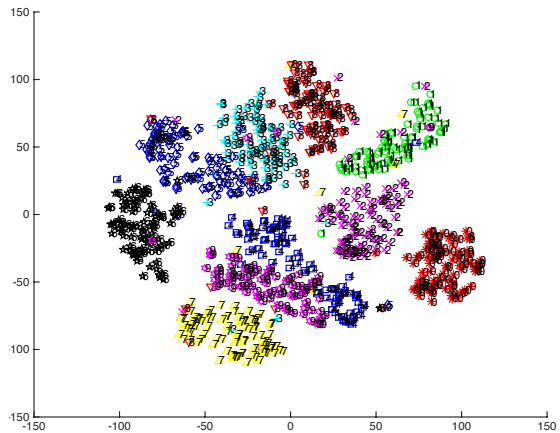
- Données : $\boldsymbol{X} \in \mathbb{R}^{N \times d}$
- Objectif : trouver une projection des données $\boldsymbol{Y} \in \mathbb{R}^{N \times q}$ avec $q < d$





$d = 784$





$q = 2$

Réduction de dimension : quelle utilité ?

- Visualiser ($q = 2$ ou 3)
  - valider le codage des données
  - identifier les données atypiques
  - visualiser les classes (données étiquetées)

- Représentation ($q < d$)
  - Réduire le bruit
  - pré-traitement : efficacité calculatoire
  - hypothèse des variables cachées (variété de petite dimension)

Opérations de Codage/Décodage

$$\begin{array}{rccccc} cod: & \mathbb{R}^d & \longrightarrow & \mathbb{R}^q & , & \boldsymbol{x} \longmapsto \boldsymbol{y} = cod(\boldsymbol{x}) \\ dec: & \mathbb{R}^q & \longrightarrow & \mathbb{R}^d & , & \boldsymbol{y} \longmapsto \boldsymbol{x} = dec(\boldsymbol{y}) \end{array}$$

Quel est le critère à optimiser ?

Malédiction de la dimensionalité

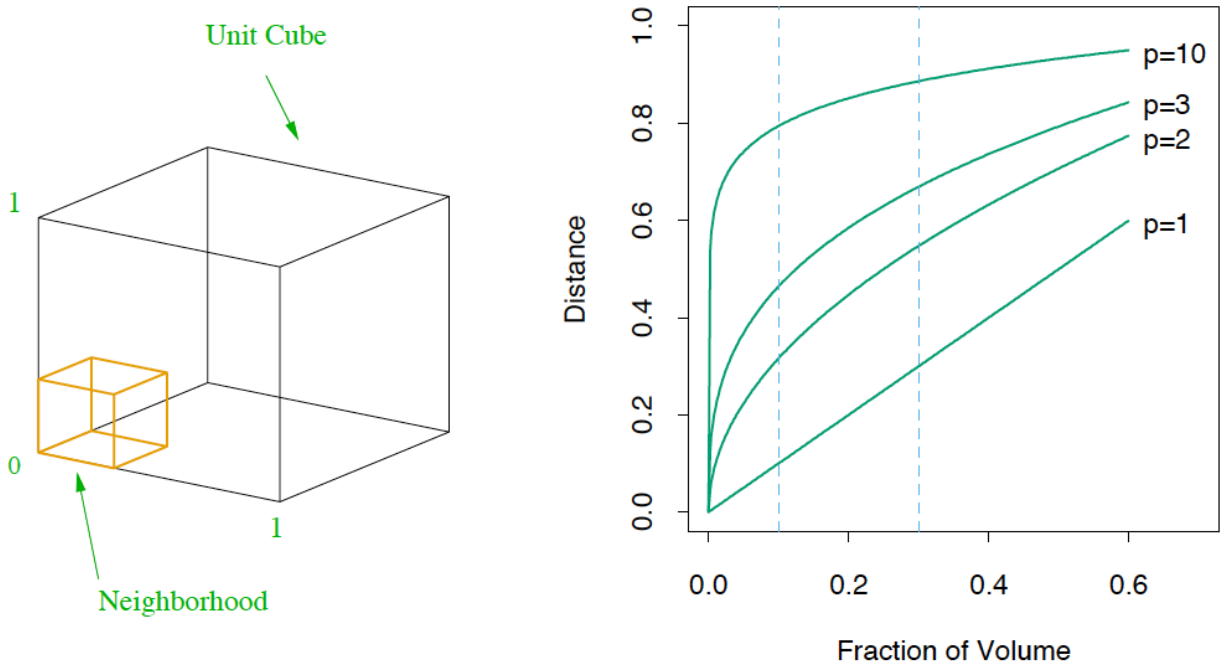


FIGURE 2.6. *The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction r of the volume of the data, for different dimensions p. In ten dimensions we need to cover 80% of the range of each coordinate to capture 10% of the data.*

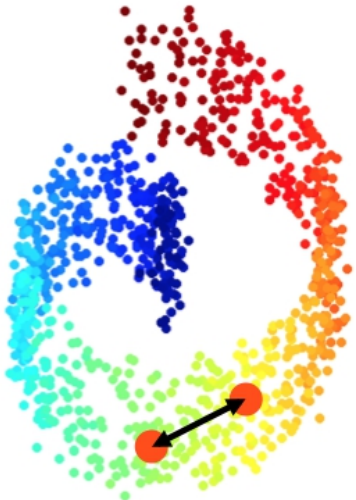
En grande dimension, les intuitions sur les distances ne sont pas les mêmes qu'en dimension 2 ou 3

image extraite the Elements of Statistical Learning, chapitre 2 (pp 23) :

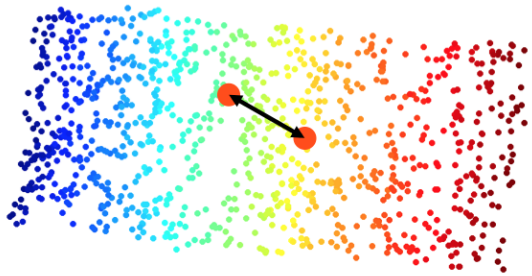
Réduction de dimension : principe

- Projeter les $\{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1,\cdots,N}$ dans une espace de dimension $q < d$ en préservant la structure des données
  - préservation de la distance
  - préservation du voisinage
  - ...

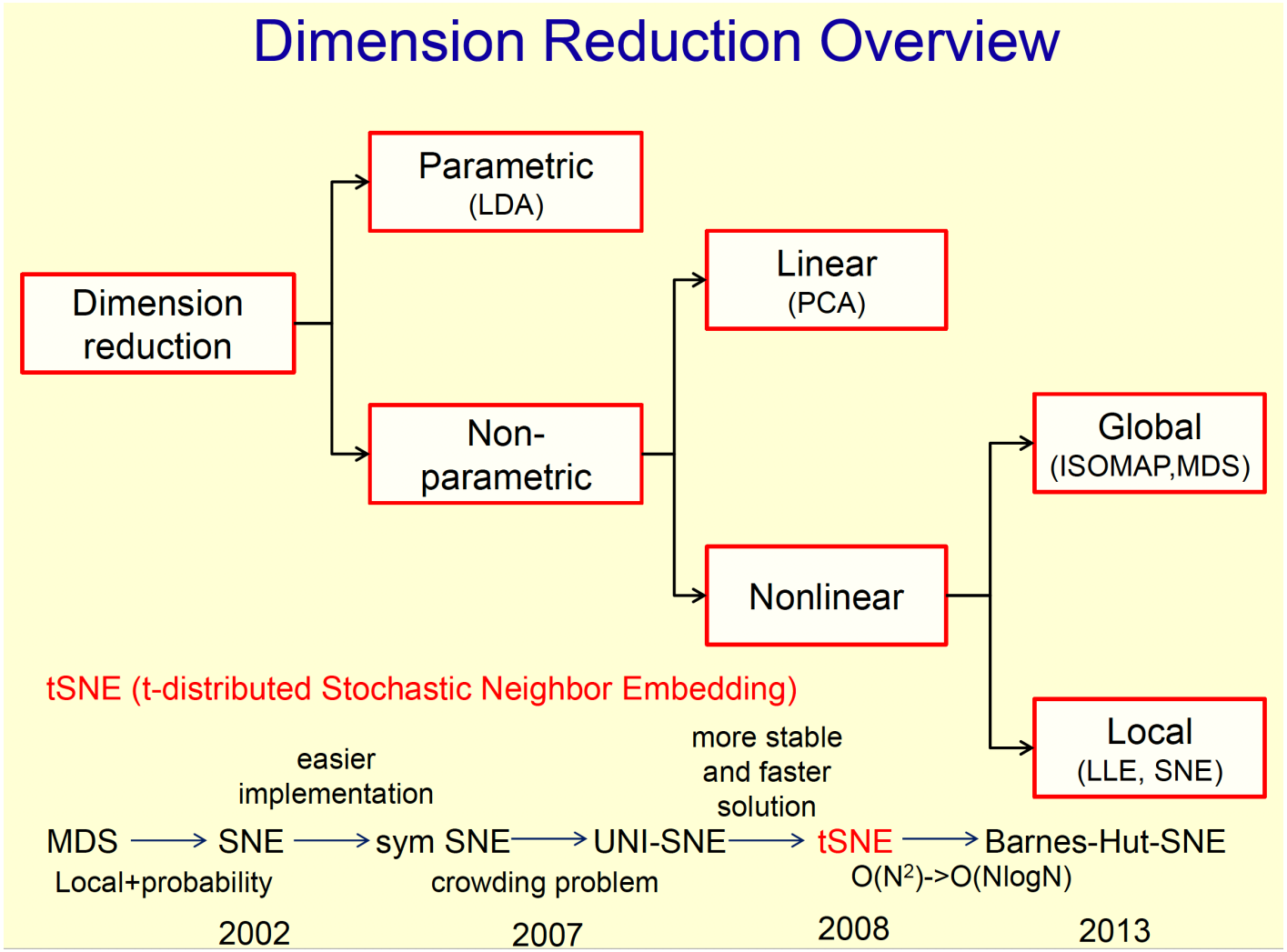
$\boldsymbol{x}_i$ en 3D



réduction $\boldsymbol{y}_i$ en 2D :
préservation de la distance



Méthodes de réduction de dimension



ACP : Analyse en composantes principales

Modèle : données = information + bruit

$$\boldsymbol{X} = \boldsymbol{Y}\boldsymbol{V}^\top + \boldsymbol{B}$$

Projection linéaire :

$$\begin{array}{lccccccc} cod: & \mathbb{R}^d & \longrightarrow & \mathbb{R}^q & , & \boldsymbol{X} & \longmapsto & \boldsymbol{Y} = \boldsymbol{X}\boldsymbol{V} \\ dec: & \mathbb{R}^q & \longrightarrow & \mathbb{R}^d & , & \boldsymbol{Y} & \longmapsto & \boldsymbol{Y}\boldsymbol{V}^\top \end{array}$$

Les dimensions : $\boldsymbol{X} \in \mathbb{R}^{N\times d}, \quad \boldsymbol{Y} \in \mathbb{R}^{N\times q}, \quad \boldsymbol{V} \in \mathbb{R}^{q\times d}$

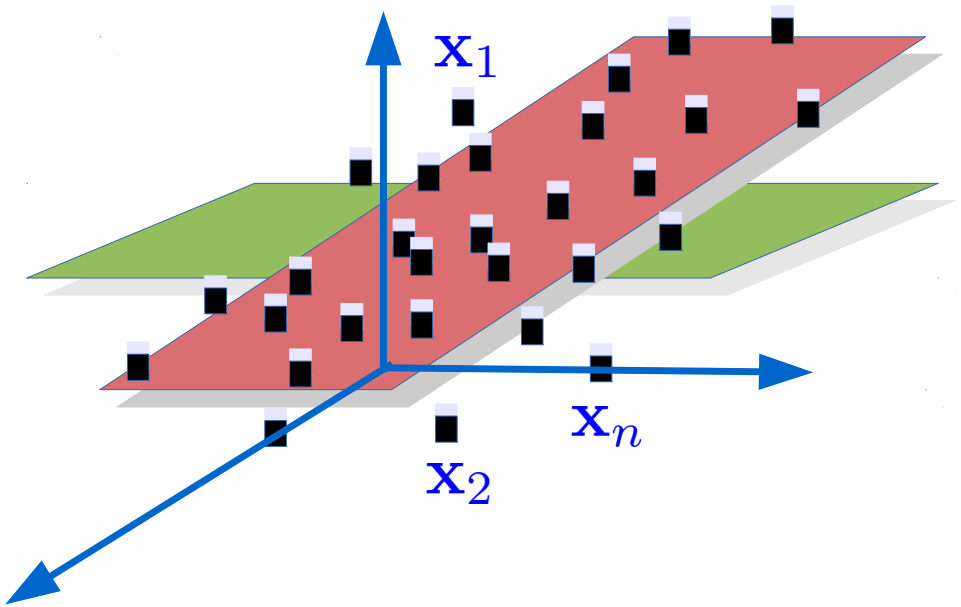
Objectif : minimiser l'erreur de reconstruction entre $\boldsymbol{X}$ et $dec(cod(\boldsymbol{X}))$

$$\min_{\boldsymbol{Y}\in\mathbb{R}^{N\times q},\boldsymbol{V}} \|\boldsymbol{X} - \boldsymbol{Y}\boldsymbol{V}^\top\|_F^2$$

$\|\boldsymbol{B}\|_F^2 = \sum_{i=1}^N \sum_{j=1}^q B_{ij}^2$ pour une matrice $\boldsymbol{B} \in \mathbb{R}^{N\times q}$

ACP vue autrement

Projection linéaire des $\{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^N$ dans un sous-espace de dimension $q\,(q < d)$ telle que "la variance des projections $\{\boldsymbol{y}_i \in \mathbb{R}^q\}_{i=1}^N$ soit maximale"



Maximisation de la variance (cas $q = 1$)

$$\max_{\boldsymbol{v} \in \mathbb{R}^q} \|\boldsymbol{X}\boldsymbol{v}\|_2^2 \quad \text{avec } \|\boldsymbol{v}\|_2^2 = 1 \text{ et } \boldsymbol{y} = \boldsymbol{X}\boldsymbol{v}$$

Calcul pratique de l'ACP

Théorème (Eckart & Young, 1936)

L'unique solution du problème d'optimisation

$$\min_{\boldsymbol{y}\in\mathbb{R}^N,\boldsymbol{v}\in\mathbb{R}^d} J(\boldsymbol{y},\boldsymbol{v}) \;=\; \|\boldsymbol{X}-\boldsymbol{y}\boldsymbol{v}^\top\|_F^2 \quad \text{avec} \quad \|\boldsymbol{v}\|=1,$$

est $\boldsymbol{v}^\star$ et $\boldsymbol{y}^\star = X\,\boldsymbol{v}^\star$; $\boldsymbol{v}^\star$ est le vecteur propre associé à λ la plus grande valeur propre de $\boldsymbol{X}^\top\boldsymbol{X}$. De plus on a $\|\boldsymbol{y}^\star\| = \sqrt{\lambda}$.

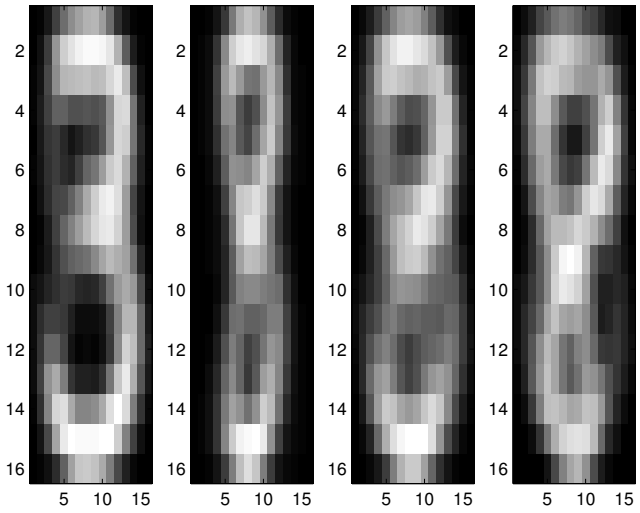
Le théorème de Courant-Fisher [a] permet de déduire que $\boldsymbol{V}\in\mathbb{R}^{q\times d}$ est formé par les q premiers vecteurs propres de $\boldsymbol{X}^\top\boldsymbol{X}$

a. `https://en.wikipedia.org/wiki/Min-max_theorem`

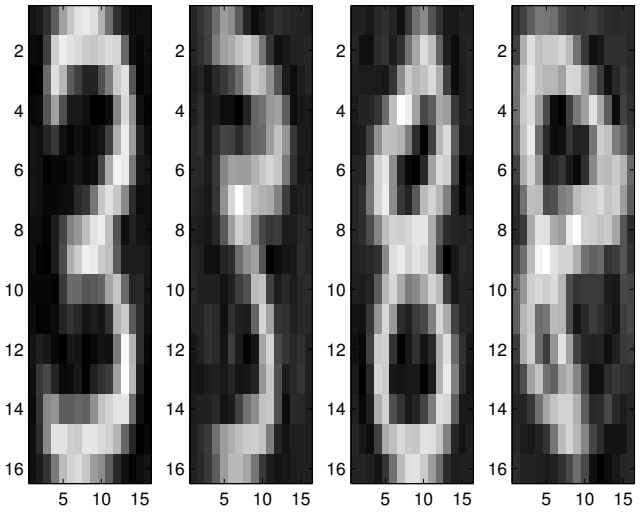
Trois manières différentes de calculer $\boldsymbol{Y}$

$$svd(\boldsymbol{X}),\; eig(\boldsymbol{X}^\top\boldsymbol{X}),\; eig(\boldsymbol{X}\boldsymbol{X}^\top)$$

ACP sur MNIST : réduction de dimension



$dec(cod(\boldsymbol{X}))$ avec $q = 2$

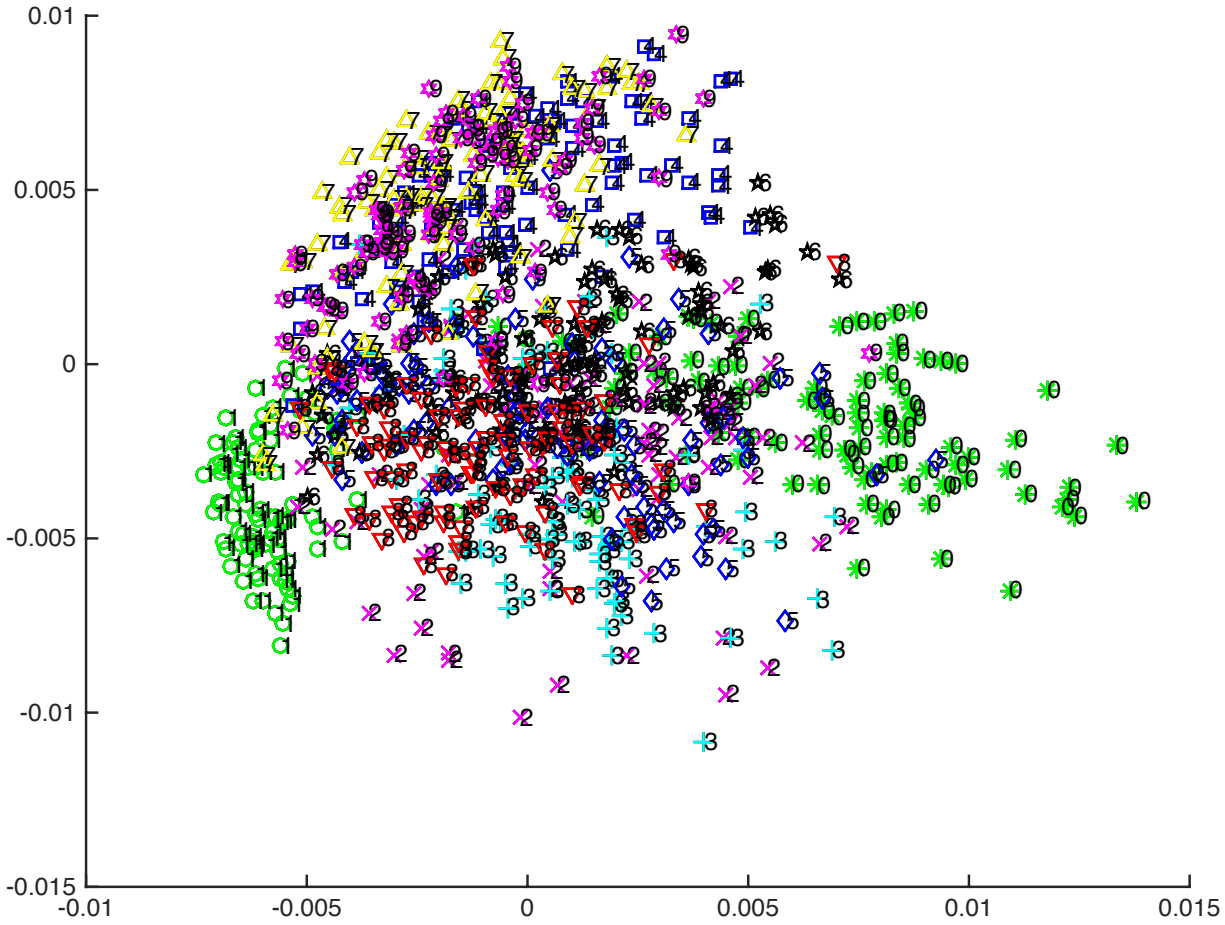


$dec(cod(\boldsymbol{X}))$ avec $q = 50$

ACP sur les données MNIST : visualisation

$d = 784$ $q = 2$

$Y =$



Limites de l'ACP

- ACP : méthode de projection linéaire
- Hypothèse implicite : distribution gaussienne des données
- Information portée par les statistiques d'ordre 2 (dépendance linéaire ou non)

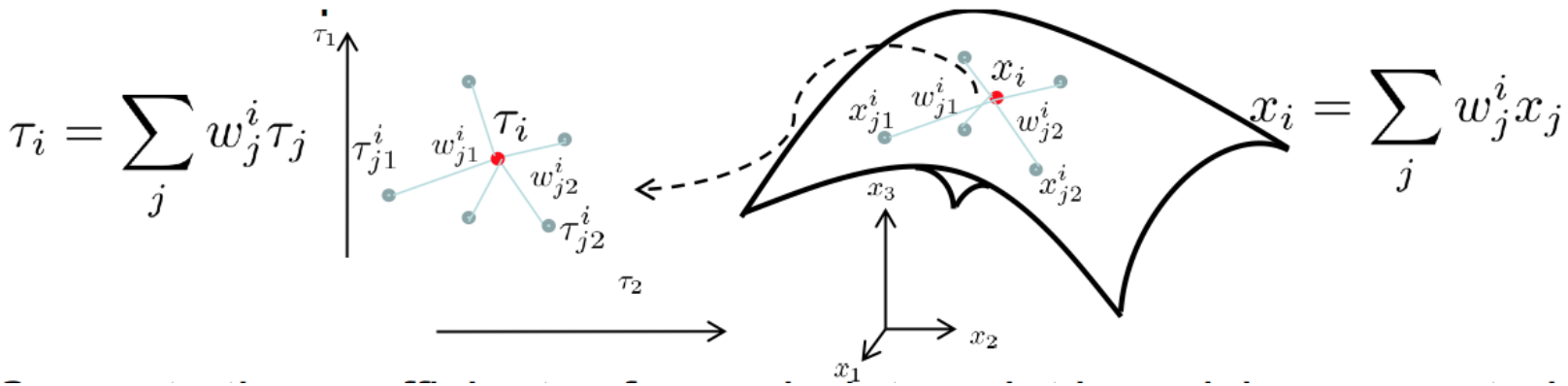
Approches non-linéaires de projection des données

- ACP non-linéaire (via méthodes à noyaux)
- ISOMAP, LLE, MVU, SNE, t-SNE ...
- Réseaux de neurones
  - auto-encodeurs
  - embeddings de données : word2vec, doc2vec (texte), signal2vec (séries temporelles)

LLE (Local Linear Embedding)

L'idée

- Représenter localement chaque point $\boldsymbol{x}_i = \sum_j w_j^i \boldsymbol{x}_j$ comme une combinaison linéaire de ses voisins
- Trouver la projection des $\boldsymbol{x}_i$, connaissant les coeff w_j^i de façon à préserver le voisinage des points

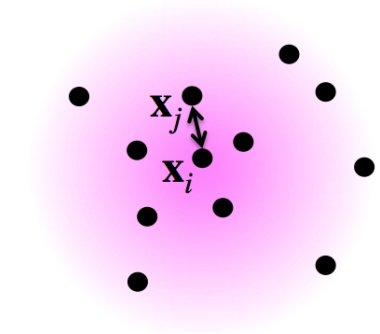


La non-linéarité de la méthode est l'approximation locale de chaque $\boldsymbol{x}_i$

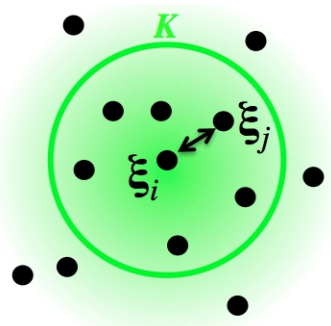
SNE (Stochastic Neighbor Embedding) et t-SNE

Le principe

- Convertir les distances $dist(\boldsymbol{x}_i, \boldsymbol{x}_j)$ entre paires de points en probabilités $\mathbb{P}_X(\boldsymbol{x}_i|\boldsymbol{x}_j)$ (qu'ils soient proches)
  - distances faibles $\rightarrow$ probabilités fortes
- Idem pour les distances entre points projetés $dist(\boldsymbol{y}_i, \boldsymbol{y}_j) \rightarrow \mathbb{P}_Y(\boldsymbol{y}_i|\boldsymbol{y}_j)$
- Calcul des $\{\boldsymbol{y}_i\}$: minimiser la distance entre les distributions p_X et p_Y



$dist(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2$



$\delta_{ij} = \|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\|_2$

SNE : mise en oeuvre

pour les $\{\boldsymbol{x}_i\}_{i=1}^n$

- définir la proba que $\boldsymbol{x}_j$ soit voisin de $\boldsymbol{x}_i$

$$\mathbb{P}_X(\boldsymbol{x}_i|\boldsymbol{x}_j) = \frac{\exp^{-d_{ij}^2}}{\sum_{k=1,k\neq i}^N \exp^{-d_{ik}^2}}$$

avec $$d_{ij} = \frac{dist(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}{2\sigma_i^2}$$

Le paramètre σ_i définit le nombre de voisins de $\boldsymbol{x}_i$. Il est choisi tq

$$\log(K) = -\sum_{j=1}^N \mathbb{P}_X(\boldsymbol{x}_i|\boldsymbol{x}_j) \log \mathbb{P}_X(\boldsymbol{x}_i|\boldsymbol{x}_j$$

pour les $\{\boldsymbol{y}_i\}_{i=1}^n$ (ce sont les inconnues)

- la proba que $\boldsymbol{y}_j$ soit voisin de $\boldsymbol{y}_i$

$$\mathbb{P}_Y(\boldsymbol{y}_i|\boldsymbol{y}_j) = \frac{\exp^{-\delta_{ij}^2}}{\sum_{k=1,k\neq i}^N \exp^{-\delta_{ik}^2}}$$

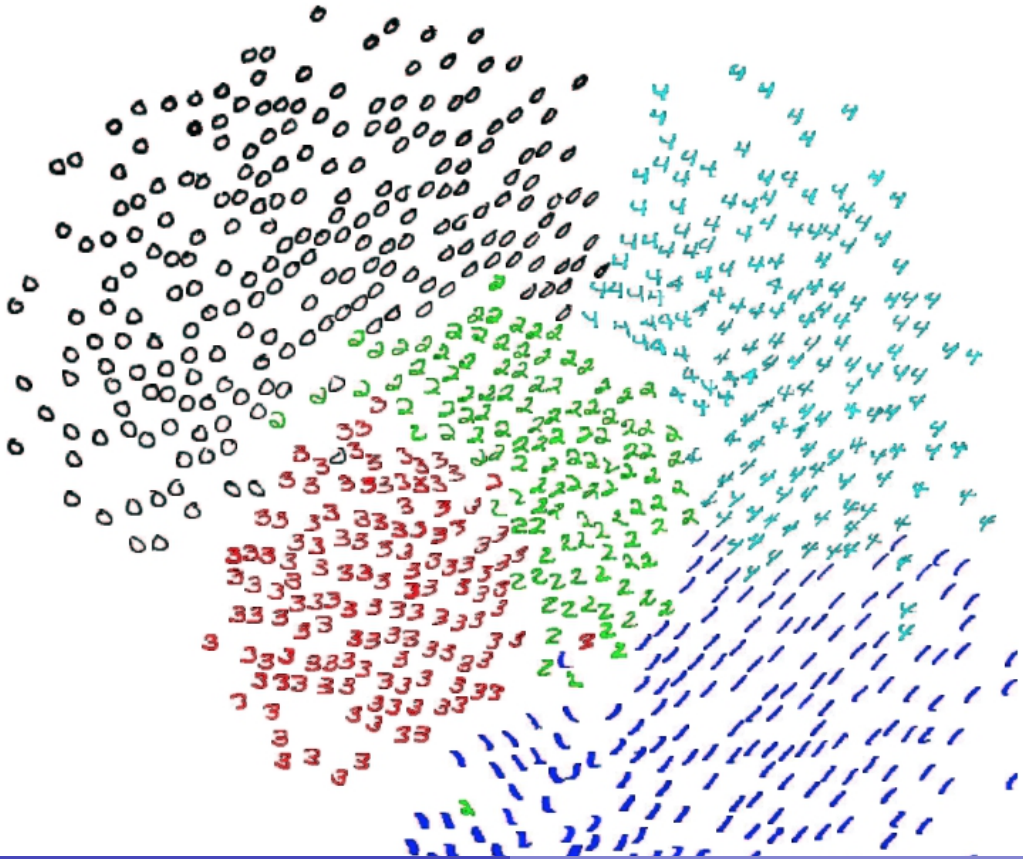
avec $$\delta_{ij} = \|\boldsymbol{y}_i - \boldsymbol{y}_j\|$$

Caclcul des $\{\boldsymbol{y}_i\}_{i=1}^n$

- Minimiser la divergence de Kullback-Leibler entre $\mathbb{P}_X$ et $\mathbb{P}_Y$
- $\min_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_N} \sum_{i,j=1}^N \mathbb{P}_X(\boldsymbol{x}_i|\boldsymbol{x}_j) \log \frac{\mathbb{P}_X(\boldsymbol{x}_i|\boldsymbol{x}_j)}{\mathbb{P}_Y(\boldsymbol{y}_i|\boldsymbol{y}_j)}$
- Optimisation par une méthode de type descente de gradient

SNE en marche

$X =$ 

$Y =$ 

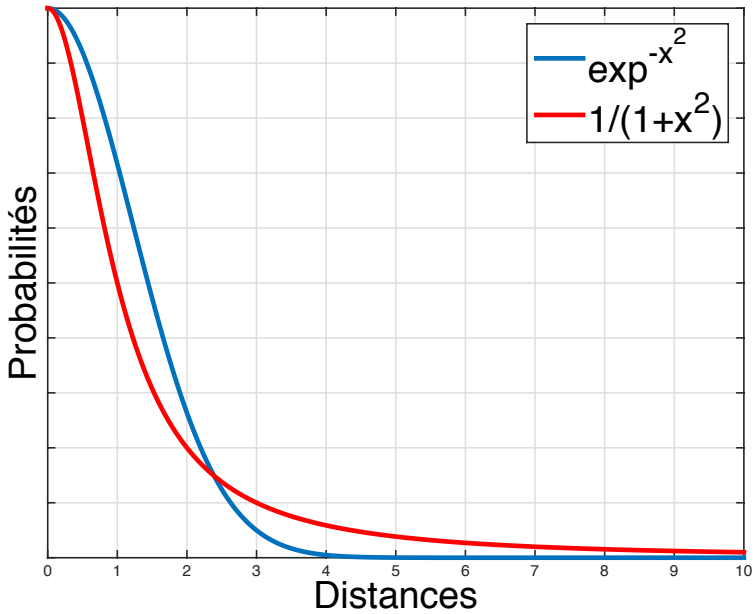
Du SNE au t-SNE

SNE

Proba $\boldsymbol{y}_j$ soit voisin de $\boldsymbol{y}_i$

$$\mathbb{P}_Y(\boldsymbol{y}_i|\boldsymbol{y}_j) = \frac{\exp^{-\delta_{ij}^2}}{\sum_{k=1,k\neq i}^N \exp^{-\delta_{ik}^2}}$$

avec $\delta_{ij} = \|\boldsymbol{y}_i - \boldsymbol{y}_j\|$



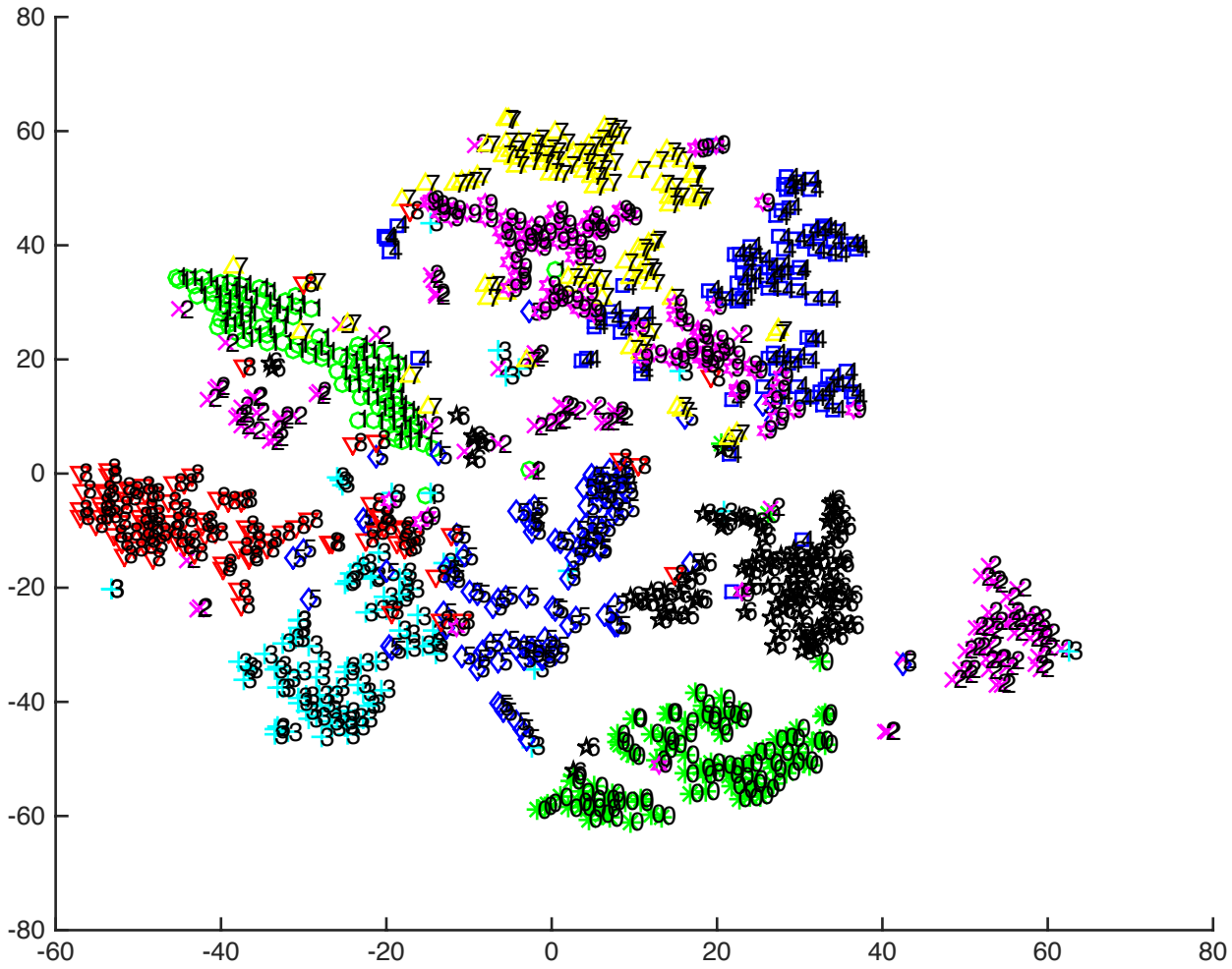
t-SNE

Proba $\boldsymbol{t}_j$ soit voisin de $\boldsymbol{t}_i$

$$\mathbb{P}_Y(\boldsymbol{y}_i|\boldsymbol{y}_j) = \frac{\left(1+\delta_{ij}^2\right)^{-1}}{\sum_{k=1,k\neq i}^N \left(1+\delta_{ik}^2\right)^{-1}}$$

- $\mathbb{P}_x = \mathbb{P}_Y$ large $\Rightarrow \delta_Y < d_X$ (attraction)
- $\mathbb{P}_x = \mathbb{P}_Y$ petite $\Rightarrow \delta_Y > d_X$ (répulsion)

Illustration du t-SNE



https://lvdmaaten.github.io/tsne/

Uniform Manifold Approximation & Projection UMAP

UMAP = Uniform Manifold Approximation & Projection

$$d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \qquad \delta_{ij} = \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2$$

$$v_i(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp^{\frac{-d_{ij}+m_i}{\sigma_i}} \qquad w_i(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp^{\frac{-\delta_{ij}+m_i}{\sigma_i}}$$

1. estimer m_i et σ_i via les K voisins du point $\boldsymbol{x}_i$
2. symétrisation des w

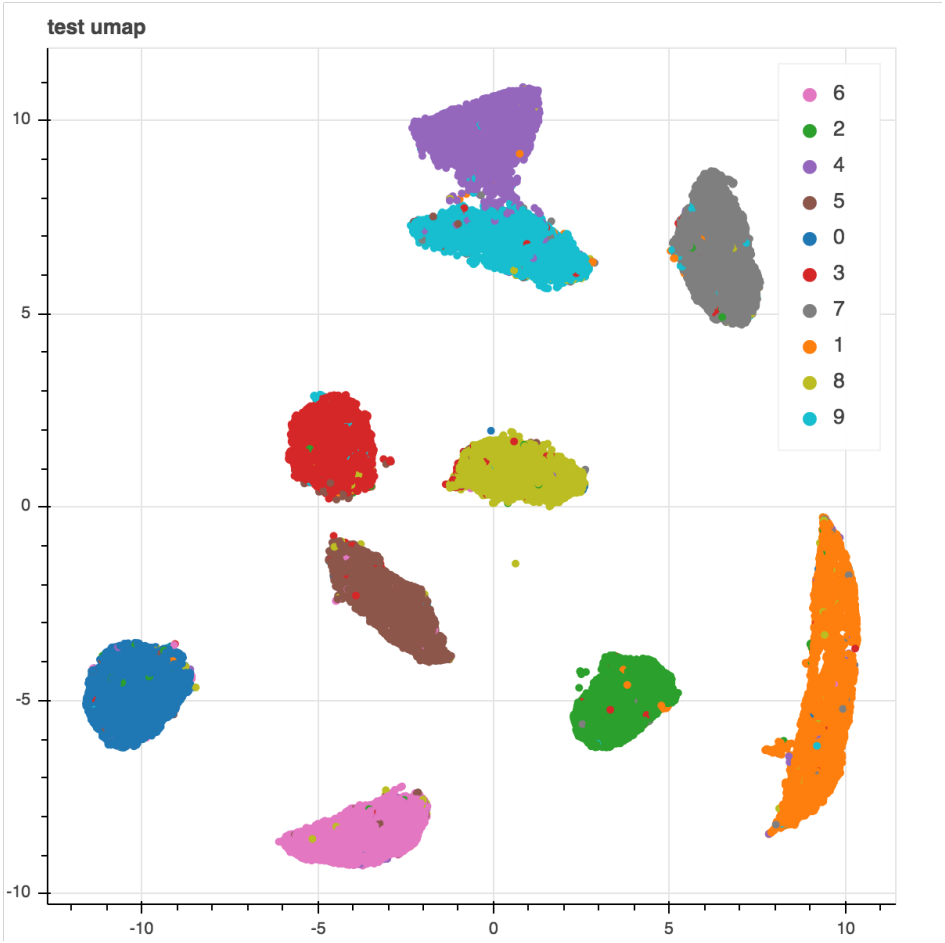
$$w(i,j) = v_i(\boldsymbol{x}_i, \boldsymbol{x}_j) + v_j(\boldsymbol{x}_i, \boldsymbol{x}_j) - v_i(\boldsymbol{x}_i, \boldsymbol{x}_j)v_j(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

3. minimiser l'entropie croisée de v et w par gradient stochastique

$$\min_Y \sum_{i=1}^{n}\sum_{j=1}^{n} w(i,j)\log\frac{w(i,j)}{v(i,j)} + (1-w(i,j))\log\frac{1-w(i,j)}{1-v(i,j)}$$

https://www.math.upenn.edu/~jhansen/2018/05/04/UMAP/ https://umap-learn.readthedocs.io/en/latest/how_umap_works.html

Uniform manifold approx. & projection (UMAP)



UMAP : Uniform Manifold Approximation and Projection for Dimension Reduction Leland McInnes, John Healy (Submitted on 9 Feb 2018)

https://github.com/lmcinnes/umap

Conclusion : réduction de dimension/visualisation

- Plusieurs méthodes (basées graphes, probabilistes, non-probabilistes ...)
- Mise en oeuvre plus délicate que l'ACP
- Utiles pour la visualisation des données en 2D ou 3D
- difficile de dire quelle méthode choisir en pratique
- Quelques toolboxes
  - Matlab : `https://lvdmaaten.github.io/drtoolbox/`
  - Python : `http://scikit-learn.org/stable/modules/manifold.html#manifold`
  - Outils graphiques : `http://divvy.ucsd.edu/`

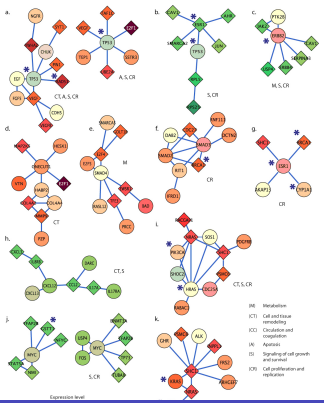
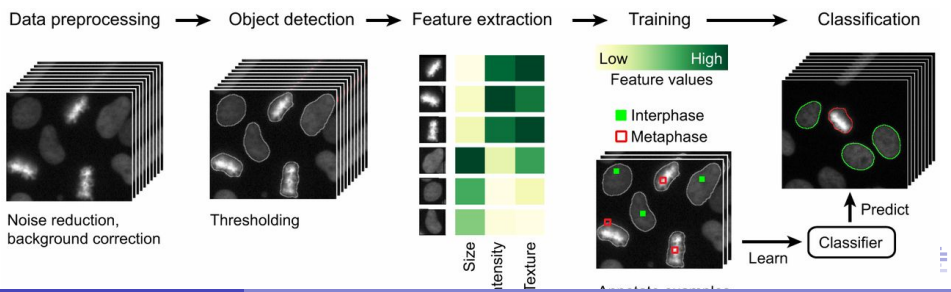
Problèmes de discrimination

Plusieurs applications réelles

- Localisation d'objets
- Biomédical : classification et prédiction de protéines, Imagerie
- Détection d'intrusion, détection d'anomalies, détection de fraudes
- Analyse de sentiments . . .





Problèmes de discrimination : qu'est-ce ? (1)

- Classiquement dénommés problèmes de classification
- Données : $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$
- $\boldsymbol{x}$: point appartenant à un espace $\mathcal{X}$ (exemple : $\mathcal{X} = \mathbb{R}^d$)
- $y \in \mathcal{Y}$: étiquette associée. $\mathcal{Y}$: ensemble fini

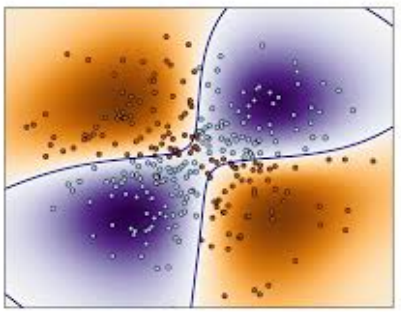
Différents types de problème de classification

- Binaire : $\mathcal{Y} = \{-1, 1\}$ ou $\mathcal{Y} = \{0, 1\}$

  Détection de fraudes, détection d'anomalies ...
- Multiclasse : $\mathcal{Y} = \{1, 2, \cdots, K\}$

  reconnaissance d'objets, de locuteurs ...
- Multi-étiquette : $\mathcal{Y} = 2^{\{1,2,\cdots,K\}}$

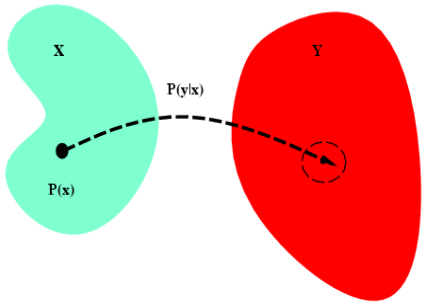
  Reconnaissance du topic de documents ...



Problèmes de discrimination : qu'est-ce ? (2)

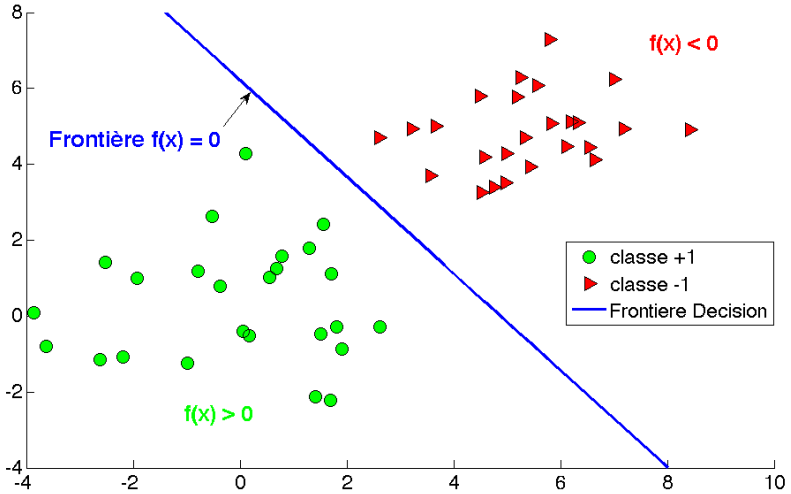
Comment on fait ?

- Apprendre une fonction de classification $f : \mathcal{X} \to \mathcal{Y}$ permettant de prédire le label de $\boldsymbol{x}$



Exemple : cas d'un problème binaire

- $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \{-1, +1\}\}_{i=1}^{N}$
- Modèle : $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$
- $\boldsymbol{x}$ est assigné à la classe
  - $+1$ si $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b \geq 0$
  - -1 autrement



Problèmes de discrimination : qu'est-ce ? (3)

Différents approches et algorithmes

- approche bayésienne de la décision
- régression logistique
- SVM, K plus proches voisins,
- approches ensemblistes : forêts aléatoires, bagging, boosting . . .

Classification binaire

Exemple : classification d'athlètes à partir de leurs paramètres biologiques

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \{0, 1\}\}_{i=1}^{N}$$

rcc	wcc	hc	hg	ferr	bmi	ssf	pcBfat	lbm	ht	wt	sex
4.82	7.6	43.2	14.4	58	22.37	50	11.64	53.11	163.9	60.1	f
4.32	6.8	40.6	13.7	46	17.54	54.6	12.16	46.12	173	52.5	f
5.16	7.2	44.3	14.5	88	18.29	61.9	12.92	48.76	175	56	f
4.53	5	40.7	14	41	17.79	56.8	12.55	38.3	156.9	43.8	f
4.42	6.4	42.8	14.5	63	20.31	58.9	13.46	39.03	149	45.1	f
4.93	7.3	46.2	15.1	41	21.12	34	6.59	67	184.4	71.8	m
5.21	7.5	47.5	16.5	20	21.89	46.7	9.5	70	187.3	76.8	m
5.09	8.9	46.3	15.4	44	29.97	71.1	13.97	88	185.1	102.7	m
4.94	6.3	45.7	15.5	50	23.11	34.3	6.43	74	184.9	79	m
4.86	3.9	44.9	15.4	73	22.83	34.5	6.56	70	181	74.8	m
4.51	4.4	41.6	12.7	44	19.44	65.1	15.07	53.42	179.9	62.9	f
4.62	7.3	43.8	14.7	26	21.2	76.8	18.08	61.85	188.7	75.5	f

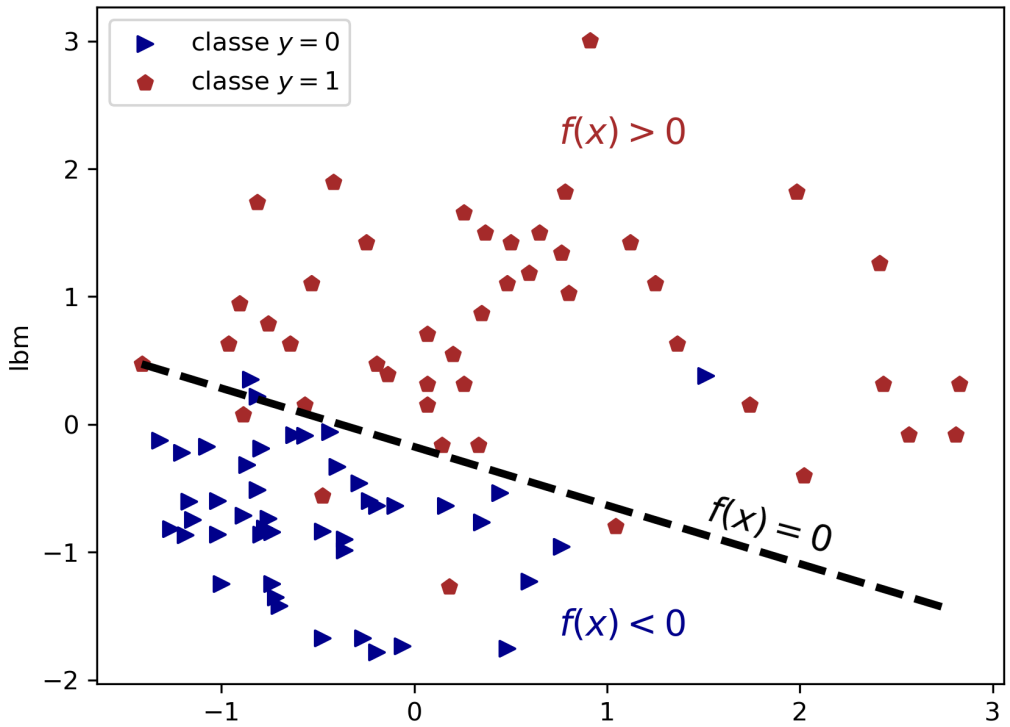
Entrées $\boldsymbol{X}$

Etiquettes $\boldsymbol{y}$

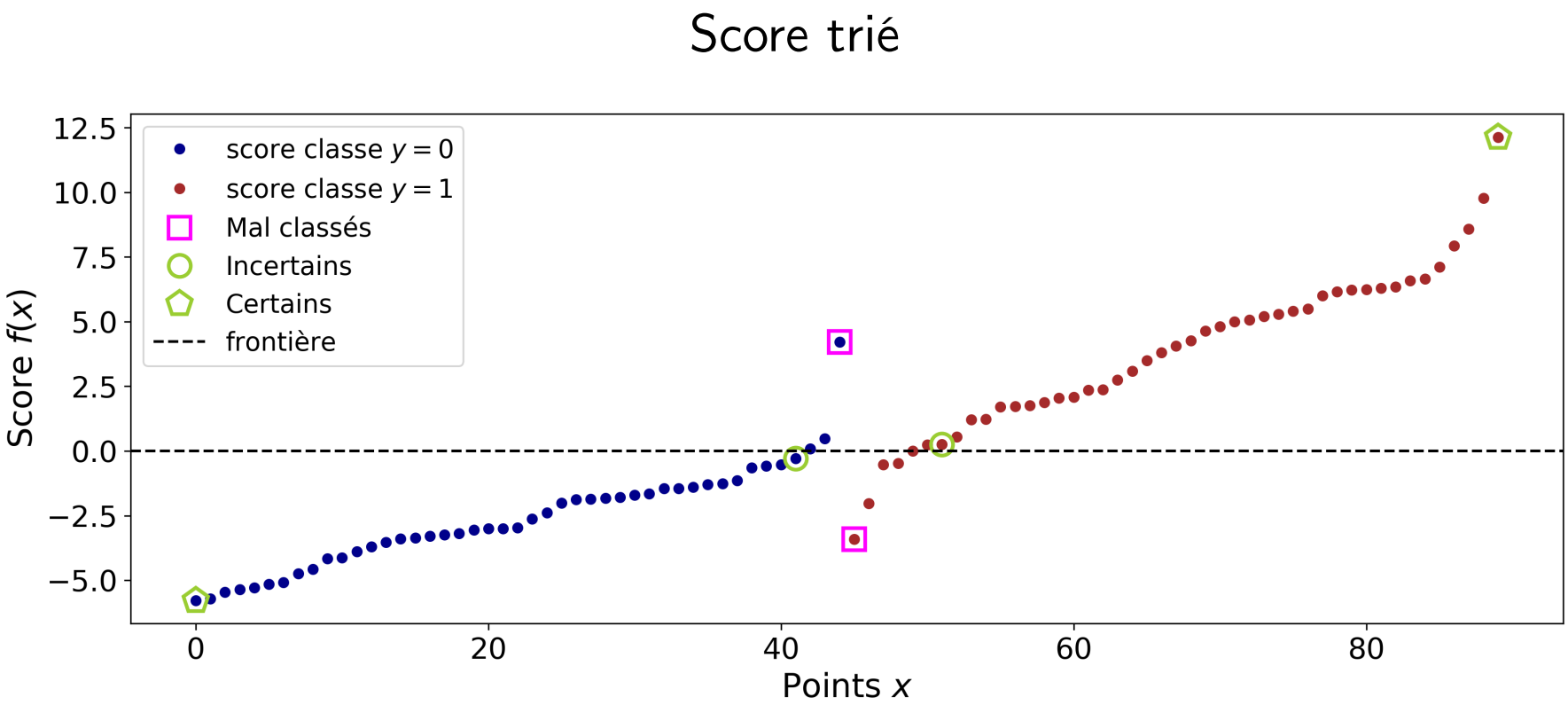
Fonction de scoring

- Modèle : $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$
- Décision : $\boldsymbol{x}$ est assigné à la classe $\hat{y} = \begin{cases} 1 & \text{si} \quad f(\boldsymbol{x}) \geq 0 \\ 0 & \text{si} \quad f(\boldsymbol{x}) < 0 \end{cases}$

Classification à partir de deux variables (*ferr*, *lbm*)



Confiance dans la décision



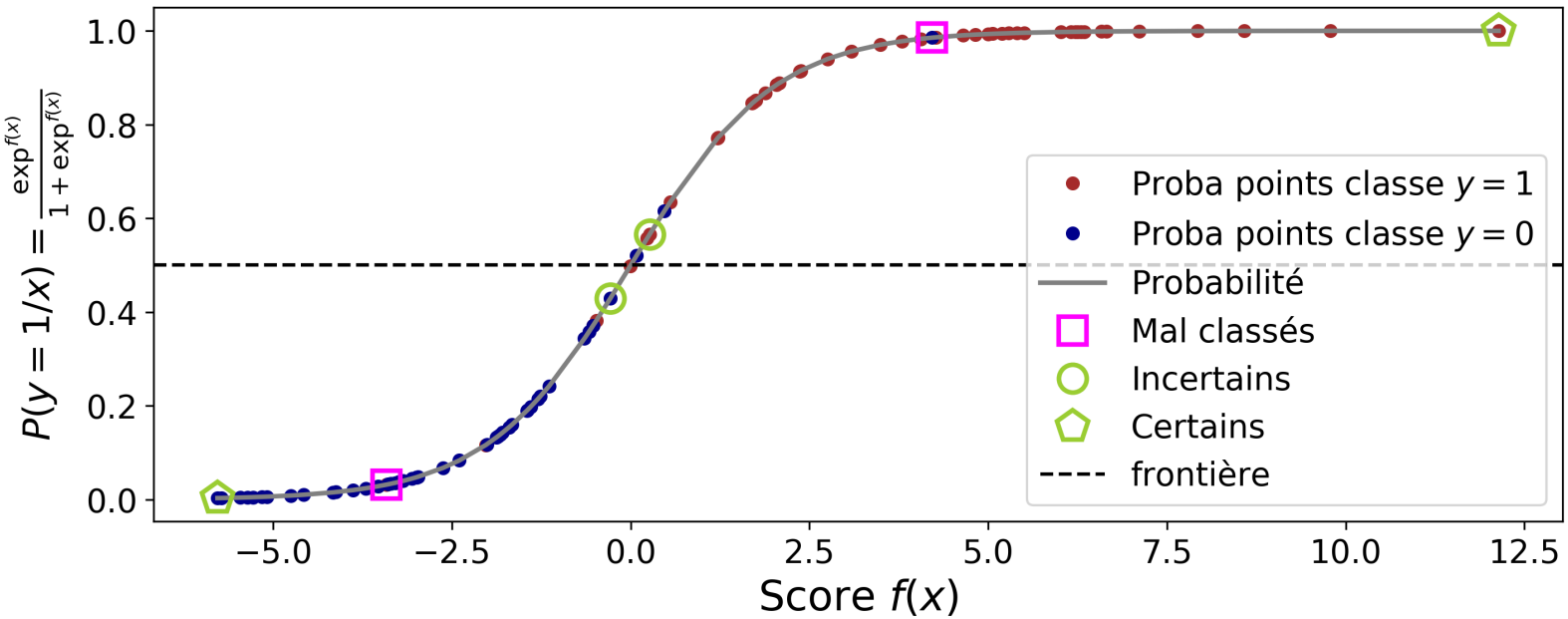
- Confiant : $f(\boldsymbol{x}) \to \infty$ et $y = 1$ ou $f(\boldsymbol{x}) \to -\infty$ et $y = 0$
- Incertain : $f(\boldsymbol{x}) \to 0$

Quantifier la confiance : du score à la probabilité a posteriori

- Fonction sigmoide, monotone croissante $\mathbb{R} \to [0, 1]$

$$\mathbb{P}(y = 1|\boldsymbol{x}) = \frac{\exp^{f(\boldsymbol{x})}}{1 + \exp^{f(\boldsymbol{x})}} \to \mathbb{P}(y = 0|\boldsymbol{x}) = 1 - \mathbb{P}(y = 1|\boldsymbol{x}) = \frac{1}{1 + \exp^{f(\boldsymbol{x})}}$$

- la fonction de décision devient $\hat{y} = \begin{cases} 1 & \text{si} \quad \mathbb{P}(y = 1/\text{x}) > 0.5 \\ 0 & \text{si} \quad \mathbb{P}(y = 1/\text{x}) < 0.5 \end{cases}$



Estimation de la fonction de score

- Pour une bonne classification, on cherche la fonction de score f tq

$$\mathbb{P}(y=1|\boldsymbol{x}) = \frac{\exp^{f(\boldsymbol{x})}}{1+\exp^{f(\boldsymbol{x})}} \rightarrow \left\{ \begin{array}{lll} 1 & \text{si} & y=1 \\ 0 & \text{si} & y=0 \end{array} \right.$$

- Maximisation de la log-vraisemblance conditionnelle

$$\begin{aligned} \mathcal{L}(\{y_i\}_{i=1}^N/\{\boldsymbol{x}_i\}_{i=1}^N; f) &= \log \prod_{i=1}^N \left[\mathbb{P}(y_i=1|\boldsymbol{x}_i)^{y_i}(1-\mathbb{P}(y_i=1|\boldsymbol{x}_i))^{1-y_i}\right] \\ &= \sum_{i=1}^N y_i \log(\mathbb{P}(y_i=1|\boldsymbol{x}_i)) + (1-y_i)\log(1-\mathbb{P}(y_i=1|\boldsymbol{x}_i)) \end{aligned}$$

- Problème d'optimisation

$$\max_f \ \mathcal{L}(\{y_i\}_{i=1}^N/\{\boldsymbol{x}_i\}_{i=1}^N; f) \Leftrightarrow \min_f J(f)$$

avec $J(f) = -\sum_{i=1}^N [y_i \log(\mathbb{P}(y_i=1|\boldsymbol{x}_i)) + (1-y_i)\log(1-\mathbb{P}(y_i=1|\boldsymbol{x}_i))]$

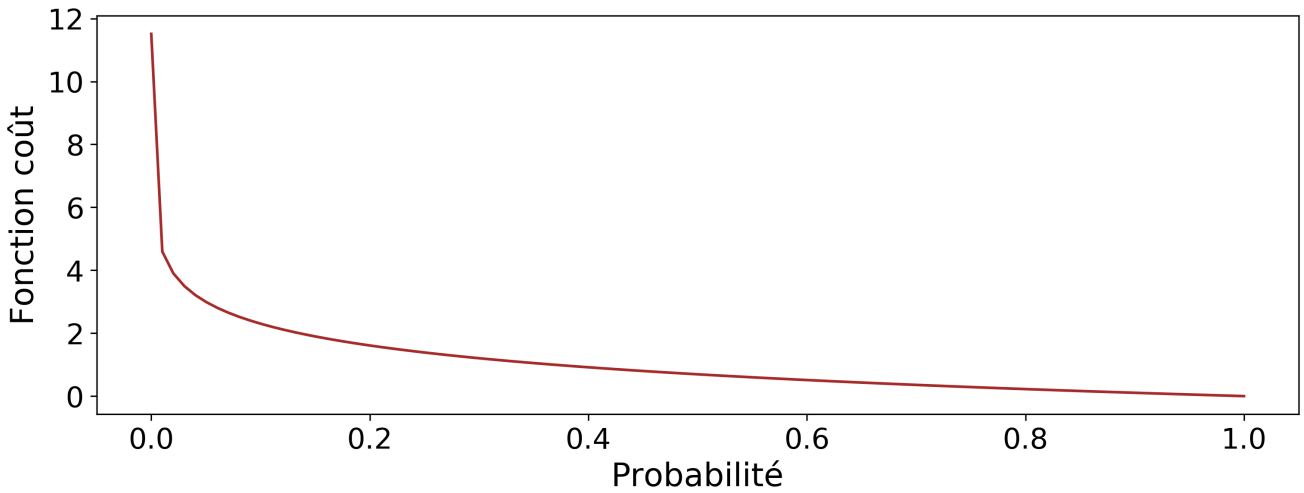
Critère à optimiser

- Ré-écriture de la fonction objectif

$$J(f) = \sum_{i=1}^{N} \ell(y_i, p_i)$$

avec $\ell(y_i, p_i) = -y_i \log p_i - (1 - y_i) \log(1 - p_i)$ et $p_i = \mathbb{P}(y_i = 1 | \boldsymbol{x}_i)$

- $\ell(y, p)$: fonction de coût connu sous le nom de **binary cross entropy**



Détails de mise en oeuvre

- Fonction de score : $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b == \boldsymbol{\varphi}^\top \boldsymbol{\theta}$

  avec $\boldsymbol{\varphi} = \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \in \mathbb{R}^{d+1}$ et $\boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{w} \\ b \end{pmatrix}$

- Proba a posteriori : $\mathbb{P}(y = 1|\boldsymbol{x}) = p = \frac{\exp^{f(\boldsymbol{x})}}{1+\exp^{f(\boldsymbol{x})}} = \frac{\exp^{\boldsymbol{\varphi}^\top \boldsymbol{\theta}}}{1+\exp^{\boldsymbol{\varphi}^\top \boldsymbol{\theta}}}$
- On en déduit

$$\min_f J(f) = \sum_{i=1}^{N} -y_i \log p_i - (1 - y_i) \log(1 - p_i)$$

$$\Leftrightarrow \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{i=1}^{N} \left[- y_i \boldsymbol{\varphi}_i^\top \boldsymbol{\theta} + \log(1 + \exp^{\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}) \right]$$

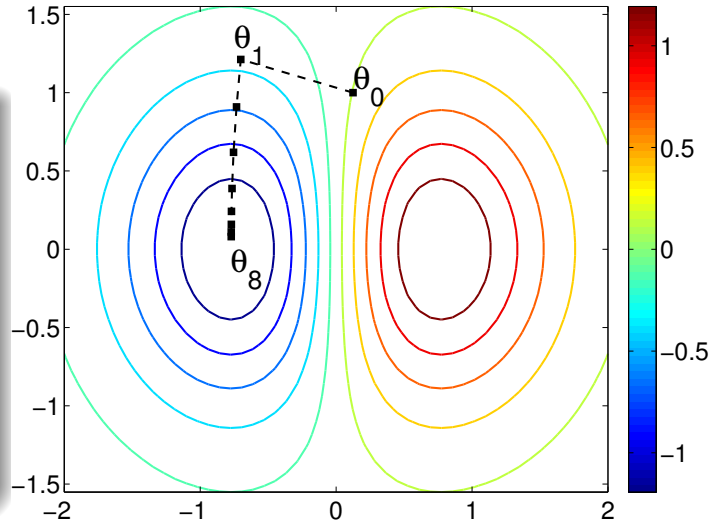
Résolution

Utiliser un algorithme de descente

Détails de mise en oeuvre (2)

Principe des méthodes de descente pour résoudre $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

- Partir d'un point initial $\boldsymbol{\theta}_0$
- Construire une séquence $\{\boldsymbol{\theta}_k\}$ avec $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha_k \boldsymbol{h}_k$
- S'assurer que la séquence $\{\boldsymbol{\theta}_k\}$ converge vers un point stationnaire $\hat{\boldsymbol{\theta}}$



- $\boldsymbol{h}_k$: direction de descente tq $J(\boldsymbol{\theta}_k) < J(\boldsymbol{\theta}_{k+1})$, α_k est le pas de descente

Deux méthodes pratiques

- Méthode du gradient : $\boldsymbol{h} = -\nabla J(\boldsymbol{\theta})$
- Newton : $\boldsymbol{h} = -\boldsymbol{H}^{-1} \nabla J(\boldsymbol{\theta})$ avec $\boldsymbol{H}$: matrice hessienne

Détails de mise en oeuvre (3)

Méthode de Newton appliquée à la régression logistique

- A chaque itération, calculer $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha_k \boldsymbol{H}^{-1} \nabla J(\boldsymbol{\theta})$
- Gradient $g = \nabla_\theta J(\theta)$

$$\nabla J(\boldsymbol{\theta}) \quad = \quad -\sum_{i=1}^{N} (y_i - p_i)\, \boldsymbol{\varphi}_i \quad \text{avec} \quad p_i = \frac{\exp^{\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}{1 + \exp^{\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}$$

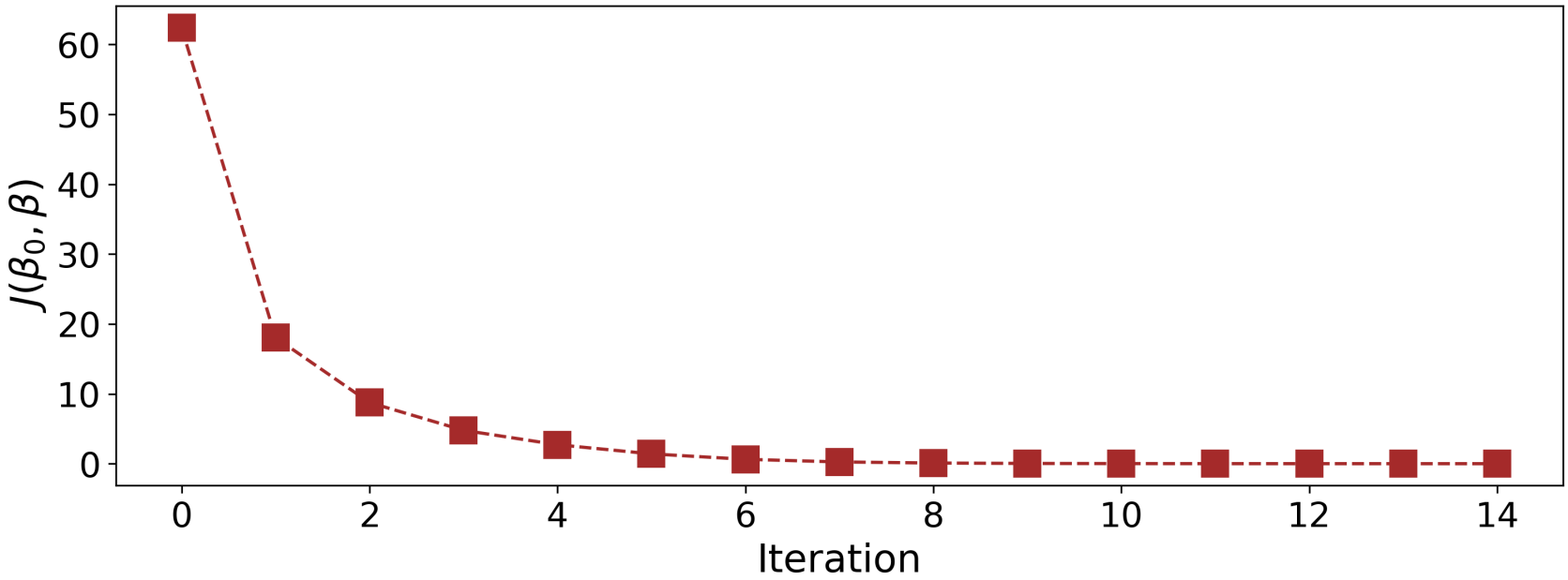
- Matrice hessienne $\boldsymbol{H} = \frac{\partial^2 J(\theta)}{\partial \theta\ \partial \theta^\top}$

$$H \quad = \quad \sum_{i=1}^{N} p_i\,(1 - p_i)\, \boldsymbol{\varphi}_i \boldsymbol{\varphi}_i^\top$$

Itérations

$$\longrightarrow \quad \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \left(\boldsymbol{\Phi}^\top \mathrm{W}_k \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^\top (\mathrm{y} - \mathrm{p}_k) \quad \text{avec} \quad \mathrm{W}_k = \mathrm{diag}(p_i 1 - p_i))$$

Illustration



Remarque

- En pratique on résout le problème de régression logistique régularisé

$$\min_{\boldsymbol{\theta}} C \; J(\boldsymbol{\theta}) + \Omega(\boldsymbol{\theta})$$

  - $C > 0$: paramètre de régularisation fixé par l'utilisateur !
  - Choix courant de régularisation : $\Omega(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2 = \sum_{j=1}^{d+1} \theta_j^2$
  - Sélection des variables utiles : $\Omega(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1 = \sum_{j=1}^{d+1} |\theta_j|$ (voir cours apprentissage de représentation parcimonieuse)

Exploitation du modèle

Classification d'un nouveau point $\boldsymbol{x}_j$

- On a l'estimation des paramètres $\hat{\boldsymbol{\theta}}$
- Calculer les probabilités a posteriori

$$\mathbb{P}(y=1|\boldsymbol{x}_j) = \frac{\exp^{\boldsymbol{\varphi}_j^\top \hat{\boldsymbol{\theta}}}}{1+\exp^{\boldsymbol{\varphi}_j^\top \hat{\boldsymbol{\theta}}}} \quad \text{et} \quad \mathbb{P}(y=0|\boldsymbol{x}_j) = \frac{1}{1+\exp^{\boldsymbol{\varphi}_j^\top \hat{\boldsymbol{\theta}}}}$$

avec $\boldsymbol{\varphi}_j = \begin{pmatrix} \boldsymbol{x}_j \\ 1 \end{pmatrix}$

- Prédire le label $\hat{y}_j = 1$ si $\mathbb{P}(y=1|\boldsymbol{x}_j) \geq 1/2$ ou $\hat{y}_j = 0$ autrement

Extension au cas multi-classe

On a K classes numérotées $0, \cdots, K-1$

- On a $K-1$ modèles de scores à estimer
- Les probabilités a posteriori se définissent comme

$$\mathbb{P}(y=k|\boldsymbol{x}) = \frac{\exp^{\boldsymbol{\varphi}^\top \boldsymbol{\theta}_k}}{1+\sum_{k=1}^{K-1}\exp^{\boldsymbol{\varphi}^\top \boldsymbol{\theta}_k}} \quad \forall k=1,\cdots,K-1$$

$$\mathbb{P}(y=0|\boldsymbol{x}) = \frac{1}{1+\sum_{k=1}^{K-1}\exp^{\boldsymbol{\varphi}^\top \boldsymbol{\theta}_k}}$$

- Règle de décision :

  prédire la classe de plus forte probabilité a posteriori i.e.

$$\hat{y} = \text{argmax}_{k\in\{0,\cdots,K-1\}}\ \mathbb{P}(y=k|\boldsymbol{x})$$

Estimation des paramètres : cas multi-classe

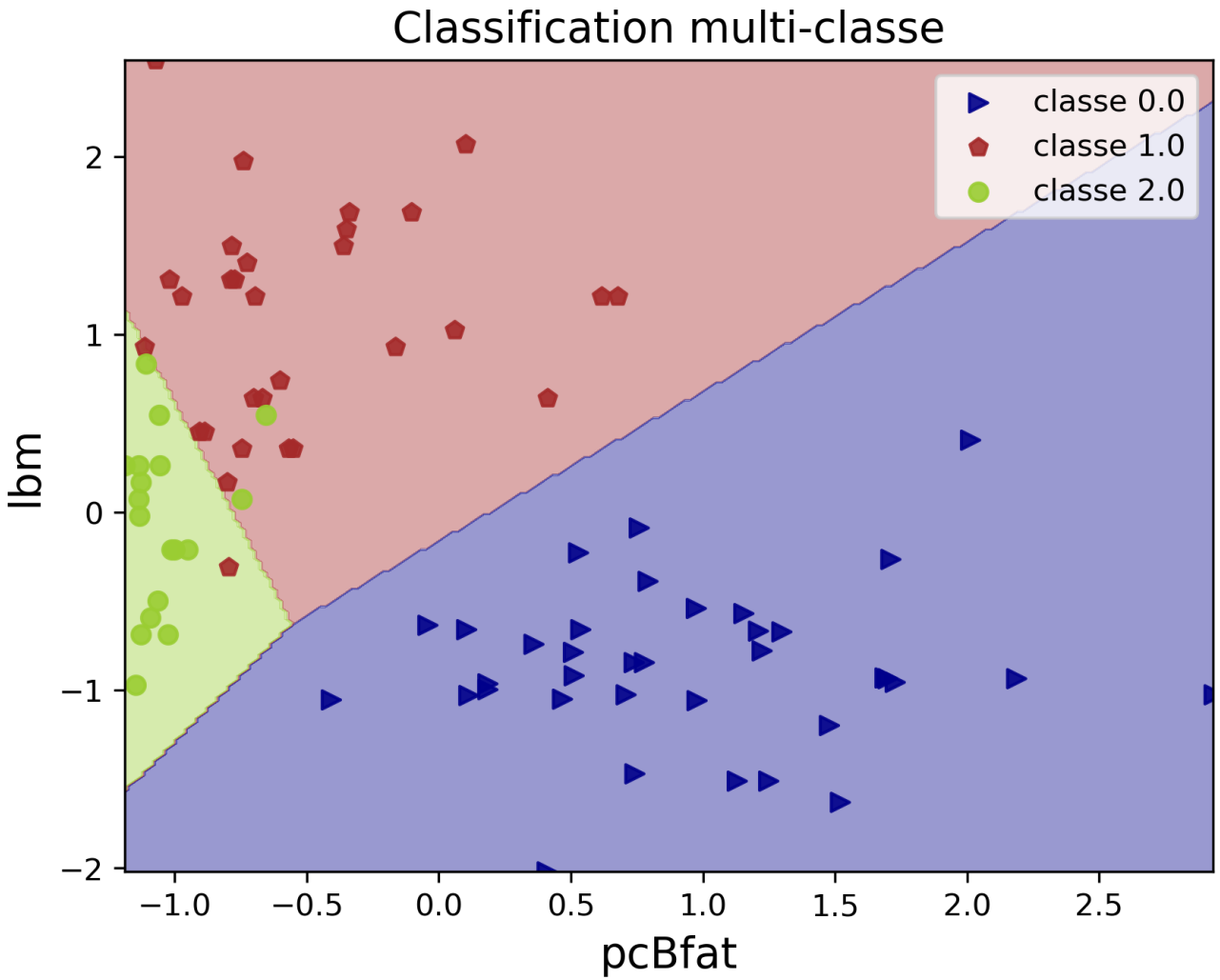
- Données $\{(\boldsymbol{x}_i, y_i)\}_{i=1,\cdots,N}$ avec $y_i \in \{0, \cdots, K-1\}$
- On définit le vecteur $\mathrm{z}_i \in \mathbb{R}^K$ tel que $z_{ik} = 1$ si $y_i = k$ et $t_{ik} = 0$ autrement
- Exemple : si $K = 3$ et $y_i = 1$ on a $\mathrm{z}_i = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^\top$
- Log vraisemblance conditionnelle (distribution multinomiale)

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} \log \mathbb{P}(y = k|\boldsymbol{x}_i) \quad \text{avec} \quad \mathbb{P}(y = k|\boldsymbol{x}) = \frac{\exp^{\boldsymbol{\varphi}^\top \boldsymbol{\theta}_k}}{1 + \sum_{k=1}^{K-1} \exp^{\boldsymbol{\varphi}^\top \boldsymbol{\theta}_k}}$$

Estimation des paramètres

Maximisation de la log-vraisemblance par rapport à $K-1$ vecteur $\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_{K-1}$

Illustration



Boîtes à outils

- Python : `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`
- R : `https://cran.r-project.org/web/packages/HSAUR/`
- Multi-langages : LIBLINEAR (dans le cas multiclasse) `https://www.csie.ntu.edu.tw/~cjlin/liblinear/`

Séparateur linéaire

But

- $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \{-1, 1\}\}_{i=1\dots N}$: ensemble de points étiquetés
- Construire à partir de $\mathcal{D}$ une fonction $f : \mathcal{X} \to \{-1, 1\}$ ou $f : \mathcal{X} \to \mathbb{R}$ qui permet de prédire la classe -1 ou 1 d'un point $\boldsymbol{x} \in \mathcal{X}$

Fonction de décision

- On suppose l'espace des entrées $\mathcal{X} = \mathbb{R}^d$
- Fonction de décision : $f : \mathbb{R}^d \to \mathbb{R}$ telle que si

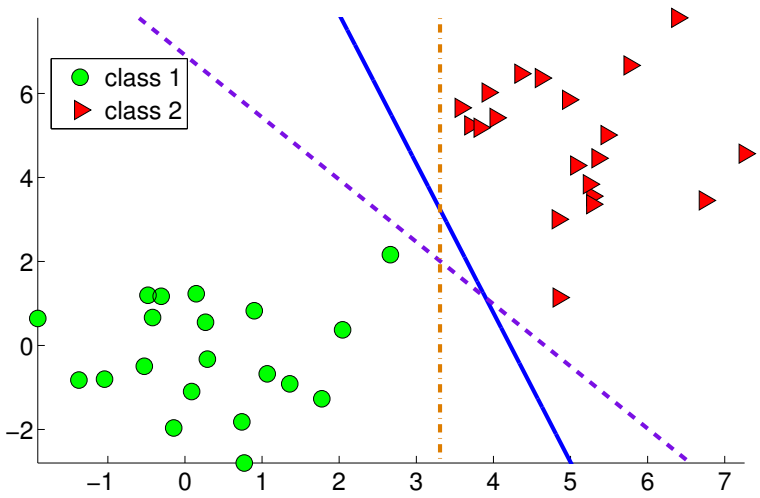
$$\begin{array}{ll} f(\boldsymbol{x}) < 0 & \text{affecter } \boldsymbol{x} \text{ à la classe } -1 \\ f(\boldsymbol{x}) > 0 & \text{affecter } \boldsymbol{x} \text{ à la classe } 1 \end{array}$$

- Fonction de décision linéaire :

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b, \qquad \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}$$

Discrimination linéaire en 2D

Trouver une fonction linéaire séparant les points des classes 1 et 2



- Frontière de décision : $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$
- Plusieurs frontières possibles
- Toutes les fonctions de décision se valent-elles ?

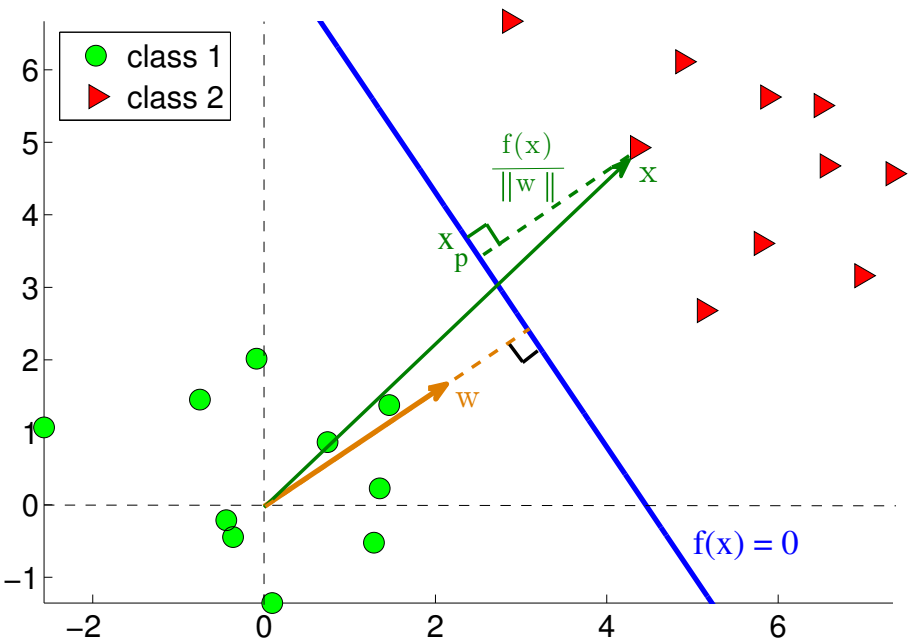
Quelle solution choisir ?

Choisir celle qui maximise la marge entre les points des classes

Notion de géométrie

Distance d'un point à la frontière de décision

Soit un hyperplan H défini par $\{f(\boldsymbol{z}) = \boldsymbol{w}^\top \boldsymbol{z} + b = 0\}$. La distance du point $x \in \mathbb{R}^d$ à H est $d(\boldsymbol{x}, H) = \frac{|\boldsymbol{w}^\top \boldsymbol{x}+b|}{\|\boldsymbol{w}\|} = \frac{|f(\boldsymbol{x})|}{\|\boldsymbol{w}\|}$



Soit $\boldsymbol{x}_p$ la projection orthogonale de $\boldsymbol{x}$ sur H.

On a $\boldsymbol{x} = \boldsymbol{x}_p + a\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \rightarrow a\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} = \boldsymbol{x} - \boldsymbol{x}_p$.

En prenant le produit scalaire avec $\boldsymbol{w}$ on a $a\boldsymbol{w}^\top \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} = \boldsymbol{w}^\top \boldsymbol{x} - \boldsymbol{w}^\top \boldsymbol{x}_p$.

On en déduit $a\frac{\|\boldsymbol{w}\|^2}{\|\boldsymbol{w}\|} = \boldsymbol{w}^\top \boldsymbol{x} + b - \underbrace{(\boldsymbol{w}^\top \boldsymbol{x}_p + b)}_{=0}$

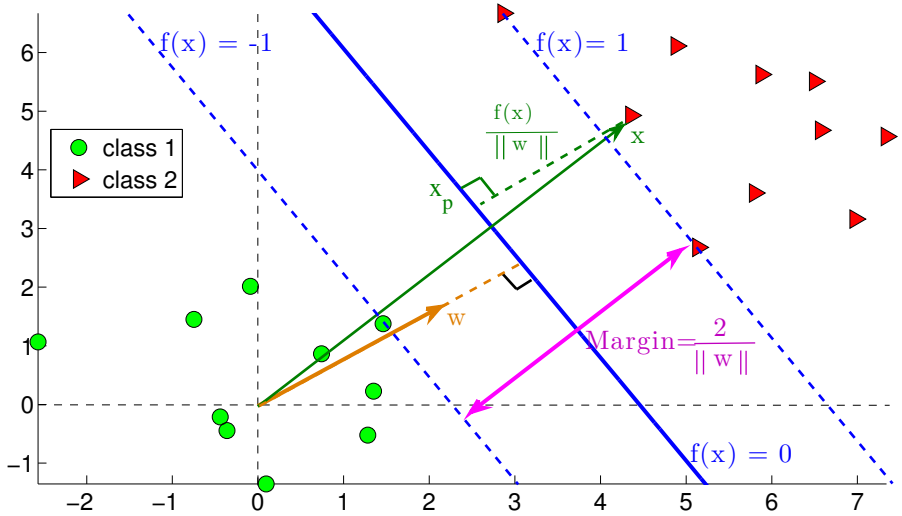
et donc $a = \frac{\boldsymbol{w}^\top \boldsymbol{x}+b}{\|\boldsymbol{w}\|}$

Séparation et marge

Hyperplan canonique

- Un hyperplan est dit canonique par rapport aux données $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ si $\min_{\boldsymbol{x}_i} |\boldsymbol{w}^\top \boldsymbol{x}_i + b| = 1$

$$\Rightarrow \text{On veut un hyperplan tq } \begin{cases} \min_{\boldsymbol{x}_i} \boldsymbol{w}^\top \boldsymbol{x}_i + b = 1 & \forall i, y_i = 1 \\ \max_{\boldsymbol{x}_i} \boldsymbol{w}^\top \boldsymbol{x}_i + b = -1 & \forall i, y_i = -1 \end{cases}$$



Fonction de décision à vaste marge

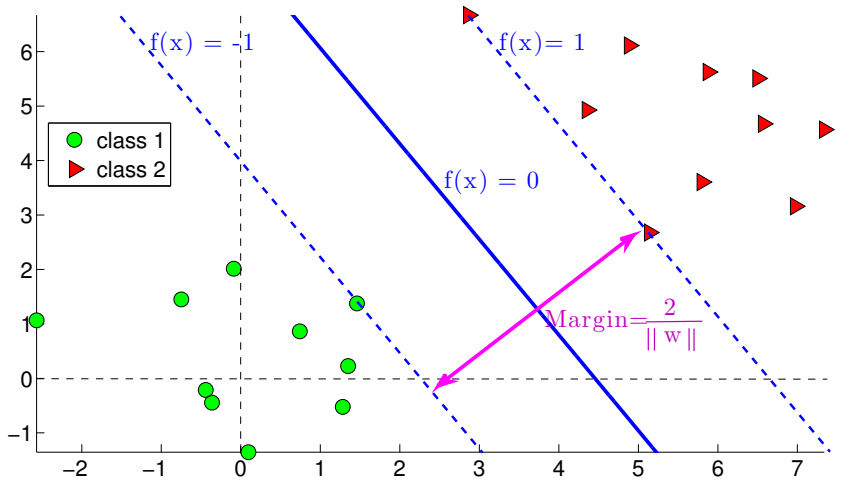
- La marge est $M = \frac{2}{\|\boldsymbol{w}\|}$
- Maximiser la marge
- Classer correctement chaque point i.e. $\forall i, \quad y_i f(\boldsymbol{x}_i) > 1$

Formulation du problème de maximisation de marge

Séparateur à vaste marge (SVM) : formulation

- $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}\}_{i=1}^N$: points linéairement séparables
- Objectif : trouver une fonction $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$ qui maximise la marge et discrimine correctement les points de $\mathcal{D}$

$$\begin{array}{lll} \min_{\boldsymbol{w},b} & \frac{1}{2}\|\boldsymbol{w}\|^2 & \text{maximisation de la marge} \\ \text{s.c.} & y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 \quad \forall i = 1, \cdots, N & \text{tous les points bien classés} \end{array}$$



Résolution

- Solution $\boldsymbol{w}$ (d'après la théorie de l'optimisation sous contraintes)

$$w = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$$

- Les coefficients α_i sont solution du problème dual
- Problème dual : problème de programmation quadratique

$$\begin{aligned} \max_{\{\alpha_i,\}_{i=1}^N} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j \\ \text{s.c.} \quad & \alpha_i \geq 0, \quad \forall\, i = 1, \cdots, N \\ & \sum_{i=1}^{N} \alpha_i y_i = 0 \end{aligned}$$

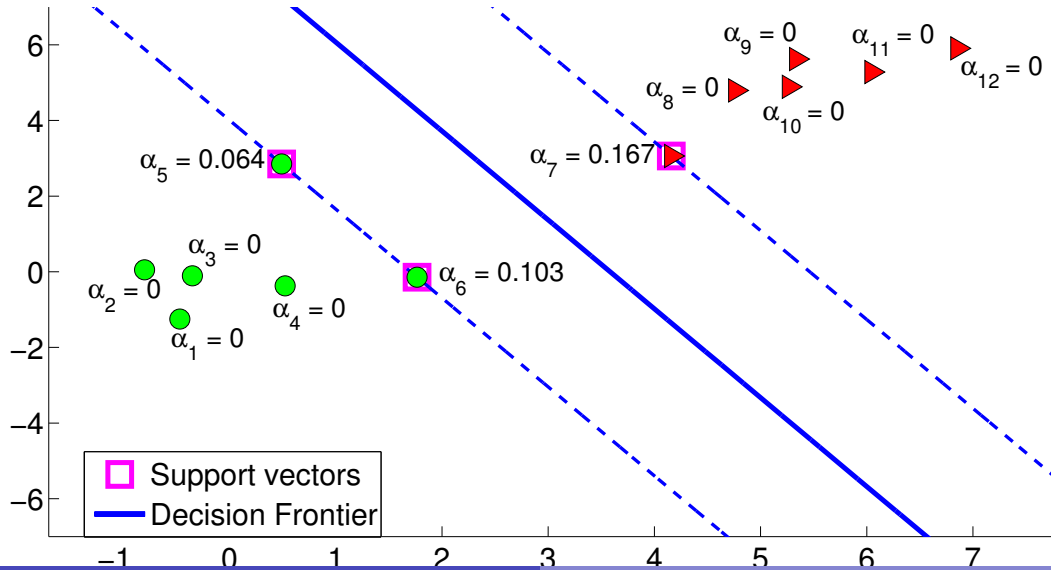
Forme matricielle

$$\begin{aligned} \max_{\boldsymbol{\alpha} \in \mathbb{R}^N} \quad & -\frac{1}{2} \boldsymbol{\alpha}^\top \boldsymbol{H} \boldsymbol{\alpha} + \mathbb{1}^\top \boldsymbol{\alpha} \\ \text{s.c.} \quad & 0 \leq \boldsymbol{\alpha} \quad \text{et} \quad \boldsymbol{\alpha}^\top \boldsymbol{y} = 0 \end{aligned}$$

avec $\boldsymbol{H} \in \mathbb{R}^{N \times N}$ une matrice tq $\boldsymbol{H}_{ij} = y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j$

Les vecteurs supports

- Après résolution du problème dual on obtient deux types de coefficients α_i
  - Pour un point $\boldsymbol{x}_j$, si $y_j(\boldsymbol{w}^\top \boldsymbol{x}_j + b) > 1$ alors $\alpha_j = 0$
  - Pour un point $\boldsymbol{x}_i$, si $y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) = 1$ alors $\alpha_i \geq 0$
- Solution : $\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$. $\boldsymbol{w}$ n'est défini que par les points tels que $y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) = 1$. On les appelle **vecteurs supports**



En pratique

Calcul du SVM linéairement séparable

- Modèle : $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$
- Il existe des toolboxes de résolution qui
  - utilisent les données $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ pour résoudre le dual
  - en déduisent la solution $\boldsymbol{w} = \sum_{i=1}^n \alpha_i y_i \boldsymbol{x}_i$ et b

Classification d'un nouveau point $\boldsymbol{x}_\ell$

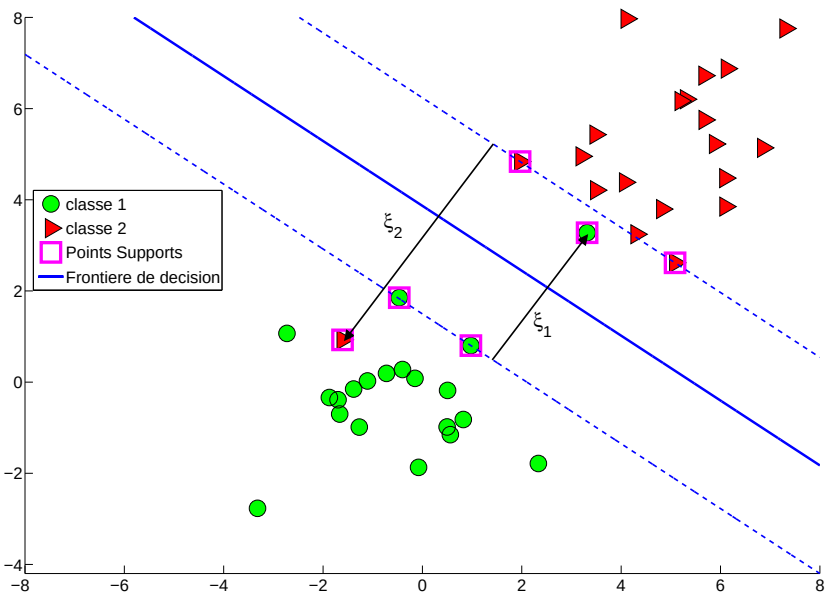
- Calcul de la fonction de décision
  $f(\boldsymbol{x}_\ell) = \boldsymbol{w}^\top \boldsymbol{x}_\ell + b = \sum_{i=1}^N \alpha_i y_i \boldsymbol{x}_i^\top \boldsymbol{x}_\ell + b$
- Affecter $\boldsymbol{x}_\ell$ à la classe $+1$ si $f(\boldsymbol{x}_\ell) > 0$ et à la classe -1 autrement

Cas non séparable

Que se passe-t-il si les données ne sont pas linéairement séparable ?

Relacher les contraintes

- Relâcher $y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1$
- Accepter $y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i$ avec $\xi_i \geq 0$ le terme "d'erreur"
- Inclure la somme des "erreurs" $\sum_{i=1}^{N} \xi_i$ dans le problème de SVM

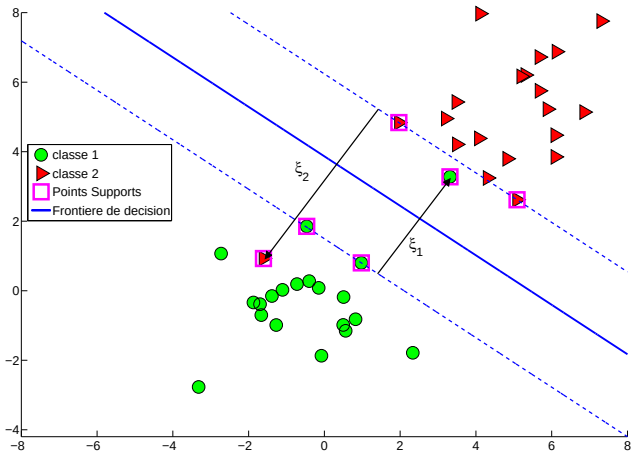


Cas non séparable : formulation

SVM : cas non-séparable

$$\begin{array}{ll} \min_{\boldsymbol{w},b,\{\xi_i\}_{i=1}^N} & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^N \xi_i \\ \text{s.c.} & y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i \quad \forall i = 1, \cdots, N \\ & \xi_i \geq 0 \quad \forall i = 1, \ldots, N \end{array}$$

- $C > 0$: paramètre de régularisation (compromis entre erreur et marge)
- C est à fixer par l'utilisateur !



Cas non séparable : la solution

Le problème dual

$$\begin{aligned} \max_{\{\alpha_i\}} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j \\ \text{s.c.} \quad & 0 \leq \alpha_i \leq C, \quad \forall\, i = 1, \cdots, N \\ & \sum_{i=1}^{N} \alpha_i y_i = 0 \end{aligned}$$

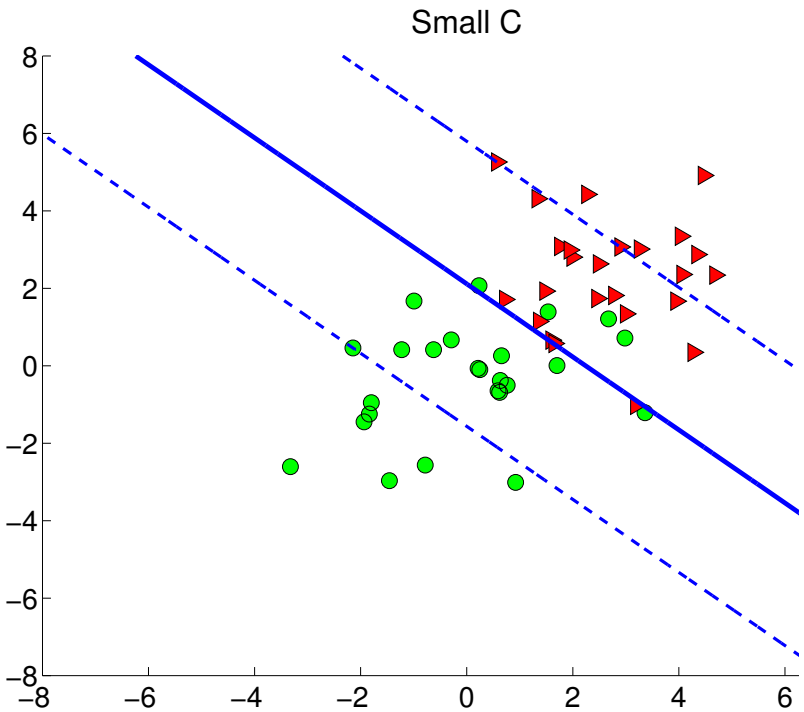
Théorème [Solution d'un SVM linéaire : cas non séparable]

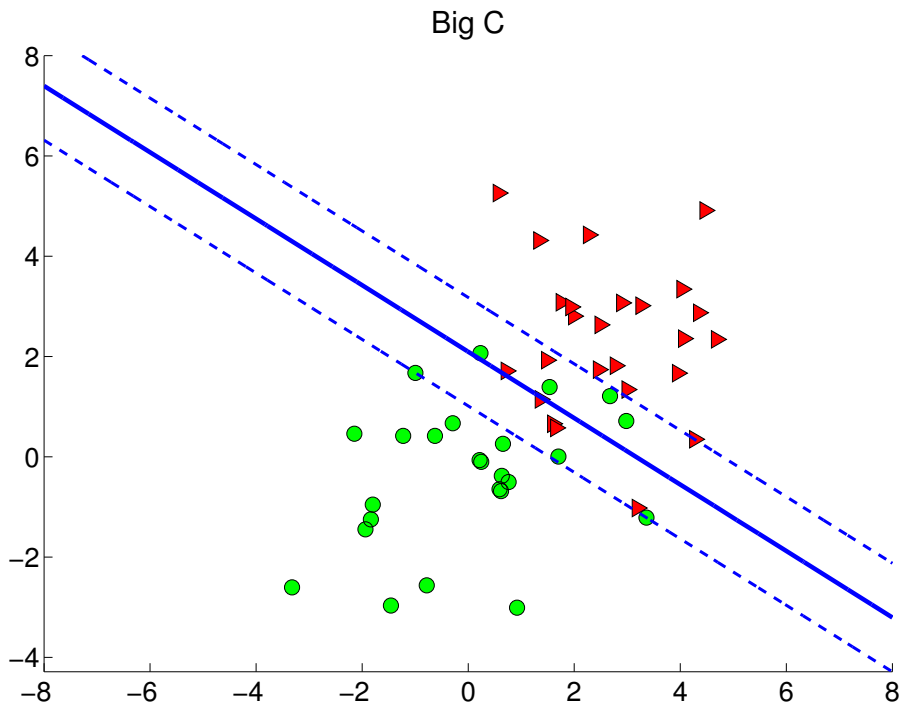
Soit un problème de SVM non-séparable de fonction de décision $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$. *Le vecteur* $\boldsymbol{w}$ *est défini par* $\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$ *où les coefficients* α_i *sont solution du problem dual ci-dessus.*

Qu'est-ce qui a changé ? Rien sauf les contraintes sur α_i qui sont maintenant $0 \leq \alpha_i \leq C$.

Illustrations

Résolution d'un SVM pour $C = 0.01$ petit et $C = 1000$ grand





Le choix de C influence la solution : C petit $\rightarrow$ marge grande ; C grand $\rightarrow$ marge petite

En pratique

Elements d'entrée

Données étiquetées : $\{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}\}_{i=1}^N$

Méthodologie

1. Centrer les données : $\{\boldsymbol{x}_i\}_{i=1}^N \longrightarrow \{\boldsymbol{x}_i = \boldsymbol{x}_i - \bar{\boldsymbol{x}}\}_{i=1}^N$
2. Fixer le paramètre $C > 0$ du SVM
3. Utiliser un solveur pour résoudre le problème et obtenir les $\alpha_i \neq 0$, les points supports $\boldsymbol{x}_i$ correspondants et le biais b
4. En déduire la fonction de décision : $f(\boldsymbol{x}) = \sum_{i \in SV} \alpha_i y_i \boldsymbol{x}_i^\top \boldsymbol{x} + b$
5. Evaluer l'erreur de généralisation du SVM obtenu (validation croisée ...). Recommencer à partir de l'étape 2 si elle n'est pa satisfaisante.

Cas multi-classe

On a K classes $\mathcal{C}_1, \cdots, \mathcal{C}_K$

Le problème multi-classe est plus compliqué à résoudre pour un SVM. Différentes approches existent. Les plus communes :

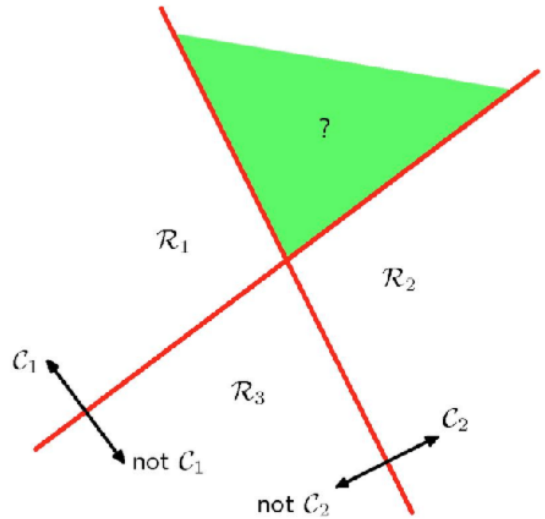
- Stratégie de "un contre tous"
  - Apprendre K modèles SVM (une classe contre les autres)
  - Classifier chaque point selon le principe "winner takes all"
- Stratégie de "un contre un"
  - Apprendre $K(K-1)/2$ modèles SVM (une classe contre une autre)
  - Classifier chaque point selon le principe du vote majoritaire
  - ou Estimer des probabilités a posteriori (principe du pairwise coupling) ; classifier selon le maximum de probabilité a posteriori

Cas multi-classe : un contre tous

Données étiquetées : $\{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \{\mathcal{C}_1, \cdots, \mathcal{C}_K\}\}_{i=1}^N$

Principe

- Pour chaque classe $\mathcal{C}_k$
  - Apprendre un SVM $f_k(\boldsymbol{x}) = \boldsymbol{w}_k^\top \boldsymbol{x} + b_k$ avec les données $\{(\boldsymbol{x}_i, z_i) \in \mathbb{R}^d \times \{-1, 1\}\}$
  - $z_i = 1$ si $y_i = \mathcal{C}_k$ et $z_i = -1$ autrement



Classification d'un point $\boldsymbol{x}_\ell$

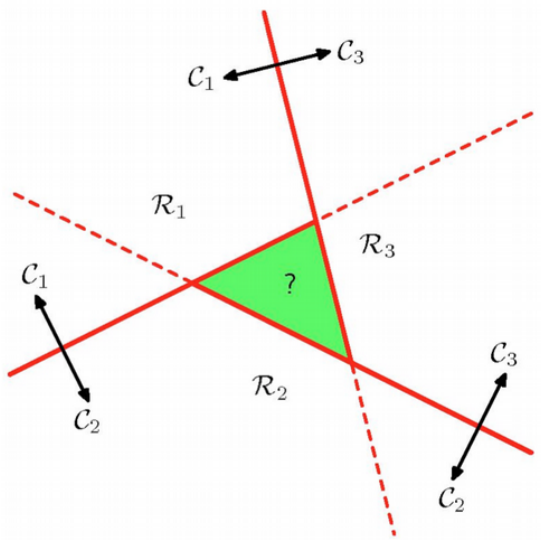
- Stratégie du winner takes all
- $D(\boldsymbol{x}_\ell) = \text{argmax}_{k=1,\cdots,K}\{\boldsymbol{w}_1^\top \boldsymbol{x}_\ell + b_1 \cdots, \boldsymbol{w}_k^\top \boldsymbol{x}_\ell + b_k, \cdots, \boldsymbol{w}_K^\top \boldsymbol{x}_\ell + b_K\}$

Cas multi-classe : un contre un

Données étiquetées : $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \{\mathcal{C}_1, \cdots, \mathcal{C}_K\}\}_{i=1}^N$

Principe

- Pour chaque paire de classes $(\mathcal{C}_j, \mathcal{C}_k)$
  - Extraire de $\mathcal{D}$ les points tq $y_i = \mathcal{C}_j$ ou $\mathcal{C}_k$
  - Apprendre un SVM $f_{jk}(\boldsymbol{x}) = \boldsymbol{w}_{jk}^\top \boldsymbol{x} + b_{jk}$ avec des données $\{(\boldsymbol{x}_i, z_i) \in \mathbb{R}^d \times \{-1, 1\}\}$
  - $z_i = 1$ si $y_i = \mathcal{C}_j$ et $z_i = -1$ si $y_i = \mathcal{C}_k$

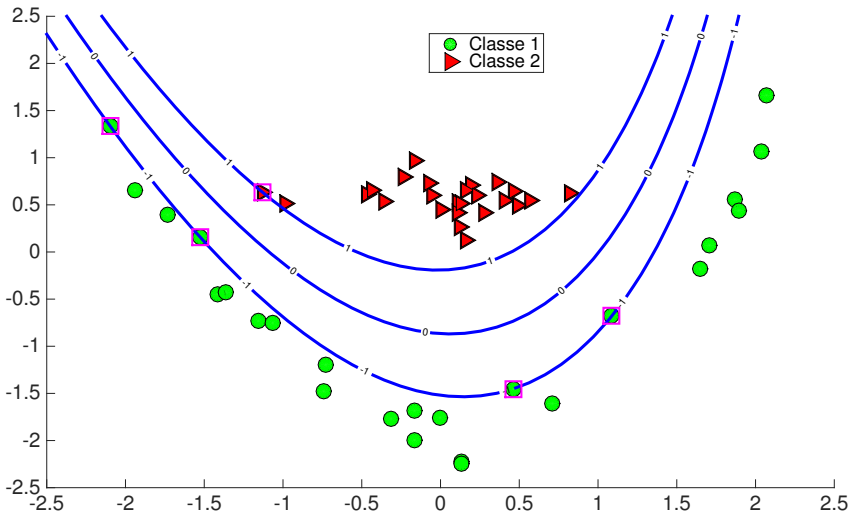


Classification d'un point $\boldsymbol{x}_\ell$: vote majoritaire

- Pour chaque modèle f_{jk}
  - si $f_{jk}(\boldsymbol{x}_\ell) > 0$ incrémenter de 1 le nombre de votes pour la classe $\mathcal{C}_j$ sinon incrémenter celui de $\mathcal{C}_k$
- Affecter $\boldsymbol{x}_\ell$ à la classe ayant eu le maximum de votes

Cas non-linéaire

Exemple : problème 2D non-linéairement séparable



- Projection non-linéaire de $\boldsymbol{x}$

$$\begin{array}{ccc} \mathbb{R}^2 & \to & \mathcal{H} \\ \boldsymbol{x} & \mapsto & \Phi(\boldsymbol{x}) = \begin{pmatrix} \sqrt{2}x_1 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \\ \sqrt{2}x_2 \end{pmatrix} \end{array}$$

- Apprendre un SVM linéaire avec les points $\{\Phi(\boldsymbol{x}_i), y_i\}$

Le modèle SVM

$$f(\boldsymbol{x}) = b + \sum_{i \in SV} \alpha_i y_i \Phi(\boldsymbol{x}_i)^\top \Phi(\boldsymbol{x})$$

Cas non-linéaire : l'astuce du noyau

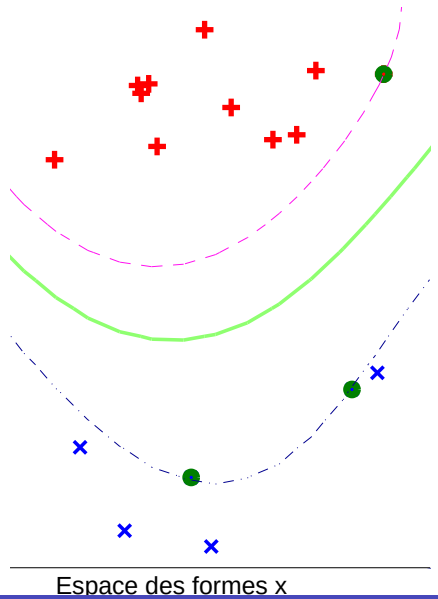
Fonction de décision

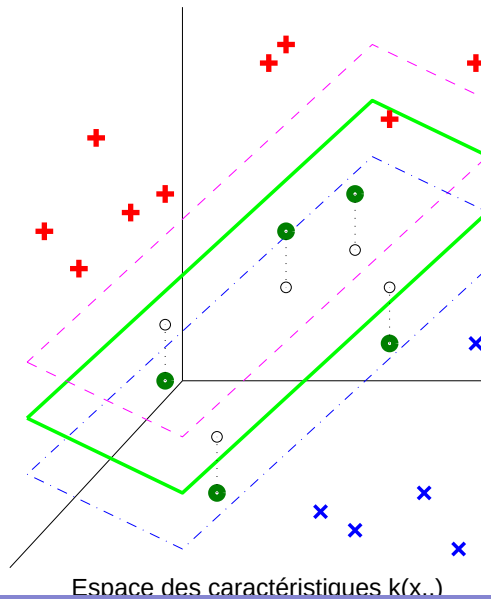
$$f(\boldsymbol{x}) = b + \sum_{i \in SV} \alpha_i y_i \underbrace{\Phi(\boldsymbol{x}_i)^\top \Phi(\boldsymbol{x})}_{\text{Noyau } k(\boldsymbol{x}_i, \boldsymbol{x})}$$

Le noyau

- Connaissance explicite de $\Phi(\boldsymbol{x})$ non-nécessaire
- Il suffit de définir une fonction $k(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

Problème linéairement non-séparable dans $\mathcal{X}$ $\longrightarrow$ Problème séparable dans l'espace $\mathcal{H}$ induit par le noyau





Notion de fonction noyau

Definition (Noyau)

Soit $\mathcal{X}$ l'espace des points $\boldsymbol{x}$. Un *noyau* est une fonction k de $\mathcal{X} \times \mathcal{X}$ dans $\mathbb{R}$ définie par

$$\begin{array}{rccc} k: & \mathcal{X} \times \mathcal{X} & \longrightarrow & \mathbb{R} \\ & \boldsymbol{x}, \boldsymbol{z} & \longmapsto & k(\boldsymbol{x}, \boldsymbol{z}) \end{array}$$

Definition (Noyau défini positif)

Un noyau $k(\boldsymbol{x}, \boldsymbol{z})$ est défini positif si :

- il est symétrique : $k(\boldsymbol{x}, \boldsymbol{z}) = k(\boldsymbol{z}, \boldsymbol{x})$
- pour tout entier non nul n :

$$\forall \{\alpha_i \in \mathbb{R}\}_{i=1,n}, \forall \{\boldsymbol{x}_i \in \mathcal{X}\}_{i=1,n}, \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$$

Quels noyaux pour les SVM ? $\longrightarrow$ Noyaux définis positifs

Exemples de noyaux positifs

Noyau linéaire $\boldsymbol{s}, \boldsymbol{t} \in \mathbb{R}^d, \quad k(\boldsymbol{s}, \boldsymbol{t}) = \boldsymbol{s}^\top \boldsymbol{t}$

symétrique : $\boldsymbol{s}^\top \boldsymbol{t} = \boldsymbol{t}^\top \boldsymbol{s}$

positif : $$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \boldsymbol{x}_i^\top \boldsymbol{x}_j$$
$$= \left(\sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i\right)^\top \left(\sum_{j=1}^{n} \alpha_j \boldsymbol{x}_j\right) = \left\|\sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i\right\|^2$$

Noyau produit : $k(\boldsymbol{s}, \boldsymbol{t}) = g(\boldsymbol{s})g(\boldsymbol{t})$ pour $g : \mathbb{R}^d \to \mathbb{R}$,

symetrique par construction

positif : $$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j g(\boldsymbol{x}_i) g(\boldsymbol{x}_j)$$
$$= \left(\sum_{i=1}^{n} \alpha_i g(\boldsymbol{x}_i)\right)\left(\sum_{j=1}^{n} \alpha_j g(\boldsymbol{x}_j)\right) = \left(\sum_{i=1}^{n} \alpha_i g(\boldsymbol{x}_i)\right)^2$$

k est défini positive $\Leftrightarrow$ (sa racine carrée existe) $\Leftrightarrow k(\boldsymbol{s}, \boldsymbol{t}) = \langle \phi_{\boldsymbol{s}}, \phi_{\boldsymbol{t}} \rangle$

Exemple

Soit $\{\phi_j, j = 1, p\}$ un ensemble fini de fonctions $\mathcal{X}$ to $\mathbb{R}$ (polynômes, filtres...)

feature map et noyau linéaire

feature map : $\begin{array}{rccc} \boldsymbol{\Phi}: & \mathcal{X} & \to & \mathbb{R}^p \\ & \boldsymbol{s} & \mapsto & \boldsymbol{\Phi} = \big(\phi_1(\boldsymbol{s}), ..., \phi_p(\boldsymbol{s})\big) \end{array}$

Noyau linéaire dans l'espace $\mathbb{R}^p$

$$k(\boldsymbol{s}, \boldsymbol{t}) = \big(\phi_1(\boldsymbol{s}), ..., \phi_p(\boldsymbol{s})\big)^\top \big(\phi_1(\boldsymbol{t}), ..., \phi_p(\boldsymbol{t})\big)$$

Ex : noyau quadratique : $\boldsymbol{s}, \boldsymbol{t} \in \mathbb{R}^d, \quad k(\boldsymbol{s}, \boldsymbol{t}) = \big(\boldsymbol{s}^\top \boldsymbol{t} + 1\big)^2$

feature map :

$$\begin{array}{rccc} \boldsymbol{\Phi}: & \mathbb{R}^d & \to & \mathbb{R}^{p=1+d+\frac{d(d+1)}{2}} \\ & \boldsymbol{s} & \mapsto & \boldsymbol{\Phi} = \big(1, \sqrt{2}s_1, ..., \sqrt{2}s_j, ..., \sqrt{2}s_d, s_1^2, ..., s_j^2, ..., s_d^2, ..., \sqrt{2}s_i s_j, ...\big) \end{array}$$

Algèbre de noyaux définis positifs

si $k_1(\boldsymbol{s},\boldsymbol{t})$ et $k_2(\boldsymbol{s},\boldsymbol{t})$ sont des noyaux définis positifs, alors sont définis positifs :

- $\forall a_1, a_2 \in \mathbb{R}^+ \quad a_1 k_1(\boldsymbol{s},\boldsymbol{t}) + a_2 k_2(\boldsymbol{s},\boldsymbol{t})$
- Noyau produit $k_1(\boldsymbol{s},\boldsymbol{t})k_2(\boldsymbol{s},\boldsymbol{t})$

preuve

- par linéarité :

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j(a_1 k_1(\boldsymbol{x}_i,\boldsymbol{x}_j)+k_2(\boldsymbol{x}_i,\boldsymbol{x}_j))= a_1\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j k_1(\boldsymbol{x}_i,\boldsymbol{x}_j)+\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j k_2(\boldsymbol{x}_i,\boldsymbol{x}_j)$$

- $\exists\psi_\ell$ s.t. $k_1(\boldsymbol{s},\boldsymbol{t}) = \sum_\ell \psi_\ell(\boldsymbol{s})\psi_\ell(\boldsymbol{t})$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j\ k_1(\boldsymbol{x}_i,\boldsymbol{x}_j)k_2(\boldsymbol{x}_i,\boldsymbol{x}_j) = \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j\Big(\sum_\ell \psi_\ell(\boldsymbol{x}_i)\psi_\ell(\boldsymbol{x}_j)k_2(\boldsymbol{x}_i,\boldsymbol{x}_j)\Big)$$

$$= \sum_\ell\sum_{i=1}^{n}\sum_{j=1}^{n}\big(\alpha_i\psi_\ell(\boldsymbol{x}_i)\big)\ \big(\alpha_j\psi_\ell(\boldsymbol{x}_j)\big)\ k_2(\boldsymbol{x}_i,\boldsymbol{x}_j)$$

N. Cristianini and J. Shawe Taylor, kernel methods for pattern analysis, 2004

Exemples de noyaux définis positifs

Type	Nom	$k(\boldsymbol{x}, \boldsymbol{z})$
radial	Gaussien	$\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{z}\|^2}{\sigma}\right)$
radial	Laplacien	$\exp(-\|\boldsymbol{x}-\boldsymbol{z}\|/\sigma)$
non stat.	χ^2	$\exp(-r/\sigma),\ r=\sum_k \frac{(\boldsymbol{x}_k-\boldsymbol{z}_k)^2}{\boldsymbol{x}_k+\boldsymbol{z}_k}$
projectif	polynomial	$(s^\top t+\sigma)^p$
projectif	cosinus	$\boldsymbol{x}^\top \boldsymbol{z}/\|\boldsymbol{x}\|\|\boldsymbol{z}\|$
projectif	corrélation	$\exp\left(\frac{\boldsymbol{x}^\top \boldsymbol{z}}{\|\boldsymbol{x}\|\|\boldsymbol{z}\|}-\sigma\right)$

- Les fonctions noyaux dépendent souvent de paramètres (ordre : p et/ou largeur de bande σ)
- La valeur optimale de ces paramètres est à fixer par l'utilisateur

Noyaux sur données structurées

Noyaux sur des histogrammes et des distributions,

Noyaux sur des séquences

- spectral string kernel $k(\boldsymbol{s}, \boldsymbol{t}) = \sum_u \phi_u(\boldsymbol{s})\phi_u(\boldsymbol{t})$
- similarité par alignements $k(\boldsymbol{s}, \boldsymbol{t}) = \sum_\pi \exp(\beta(\boldsymbol{s}, \boldsymbol{t}, \pi))$

Noyaux sur des graphes,

- noyaux de diffusion
- convolution sur des sous graphes (marches aléatoires)

noyaux sur des automates, des systèmes dynamiques...

Shawe-Taylor & Cristianini's Book, 2004 ; JP Vert, 2006

Combinaison de noyaux

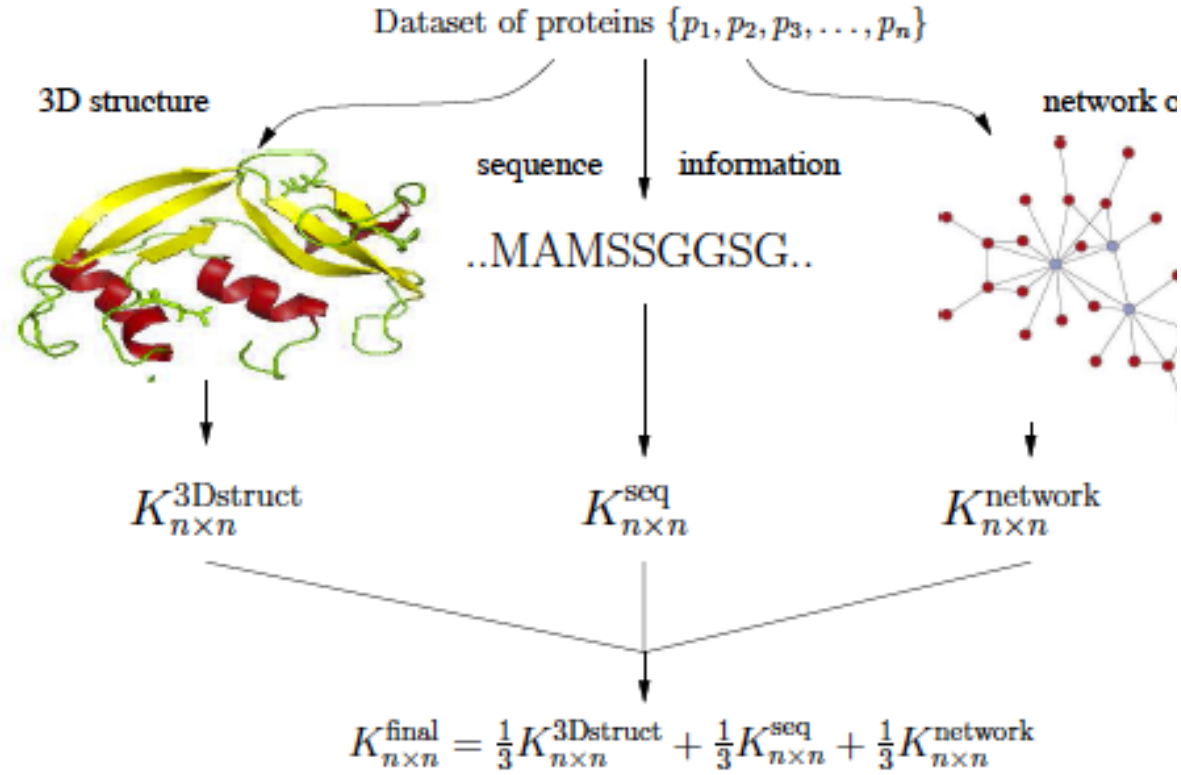


Figure 2: A dataset of proteins can be regarded in (at least) three different wa of 3D structures, a dataset of sequences and a set of nodes in a network which in other. A different kernel matrix can be extracted from each datatype, using know shapes, strings and graphs. The resulting kernels can then be combined togethe weights, as is the case above where a simple average is considered, or estimated the subject of Section 5.2

RKHS

Definition (Espace de Hilbert à noyau reproduisant (RKHS))

Un espace de Hilbert $\mathcal{H}$ équipé du produit scalaire $\langle \bullet, \bullet \rangle_{\mathcal{H}}$ est à noyau reproduisant s'il existe un noyau déf. positif k tq

$$\begin{aligned} &\forall \boldsymbol{s} \in \mathcal{X}, \quad k(\bullet, \boldsymbol{s}) \in \mathcal{H} \\ &\forall g \in \mathcal{H}, \quad g(\boldsymbol{s}) = \langle g(\bullet), k(\boldsymbol{s}, \bullet) \rangle_{\mathcal{H}} \end{aligned}$$

Caution : $g = g(\bullet)$ est une fonction et $g(\boldsymbol{s})$ est la valeur prise par g au point $\boldsymbol{s}$

Noyau défini positif $\Leftrightarrow$ RKHS

- definit le produit scalaire dans $\mathcal{H}$
- définit la régularité dans $\mathcal{H}$
- Il existe une fonction de "mapping" $\Phi : \mathcal{X} \to \mathcal{H}$ telle que $\forall \boldsymbol{s}, \boldsymbol{t} \in \mathcal{X}$ le produit scalaire $\langle \Phi(\boldsymbol{s}), \Phi(\boldsymbol{t}) \rangle_{\mathcal{H}} = k(\boldsymbol{s}, \boldsymbol{t})$

SVM non-linéaire : formulation

- Soit $k(\cdot,\cdot)$ un noyau défini positif qui induit un espace $\mathcal{H}$
- Il existe une fonction de "mapping" $\Phi : \mathcal{X} \to \mathcal{H}$ telle que $\forall \boldsymbol{x}, \boldsymbol{z} \in \mathcal{X}$ le produit scalaire $\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle_{\mathcal{H}} = k(\boldsymbol{x}, \boldsymbol{z})$

Formulation non-linéaire du SVM

$$
\begin{array}{lll}
\min_{\boldsymbol{w} \in \mathcal{H}, b, \{\xi_i\}_{i=1}^N} & \frac{1}{2}\|\boldsymbol{w}\|_{\mathcal{H}}^2 + C \sum_{i=1}^N \xi_i & \\
\text{s.c.} & y_i(\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i) \rangle_{\mathcal{H}} + b) \geq 1 - \xi_i & \forall i = 1, \cdots, N \\
& \xi_i \geq 0 & \forall i = 1, \ldots, N
\end{array}
$$

SVM non-linéaire : solution

Le problème dual

$$\max_{\{\alpha_i\}} \quad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \underbrace{\langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}}}_{k(\boldsymbol{x}_i, \boldsymbol{x}_j)}$$

$$\text{s.c.} \quad 0 \leq \alpha_i \leq C, \quad \forall\, i = 1, \cdots, N$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

Forme matricielle

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^N} \quad -\frac{1}{2} \boldsymbol{\alpha}^\top \boldsymbol{H} \boldsymbol{\alpha} + \mathbb{1}^\top \boldsymbol{\alpha}$$

$$\text{s.c.} \quad 0 \leq \boldsymbol{\alpha} \leq C\, \mathbb{1}^\top, \boldsymbol{\alpha}^\top \boldsymbol{y} = 0$$

avec $\boldsymbol{H} \in \mathbb{R}^{N \times N}$ une matrice tq $\boldsymbol{H}_{ij} = y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$

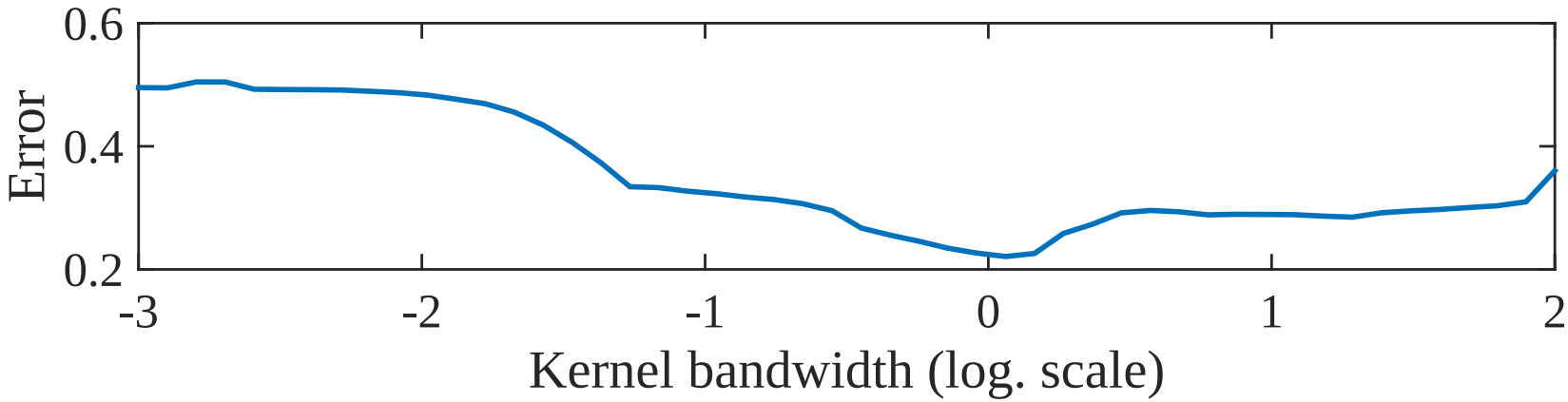
Fonction de décision

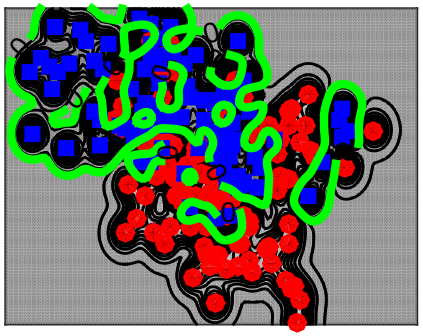
$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

- $f(\boldsymbol{x})$ s'exprime directement en fonction du noyau k
- SVM linéaire = SVM avec un noyau linéaire $k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x}^\top \boldsymbol{z}$

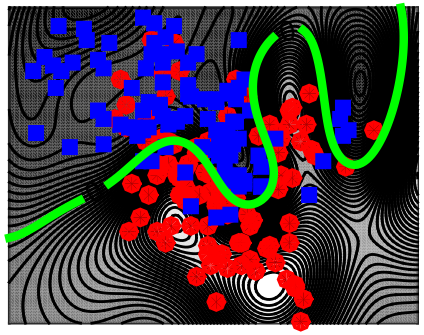
Influence de l'hyper-paramètre du noyau

Noyau gaussien : $\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{z}\|^2}{\sigma}\right)$

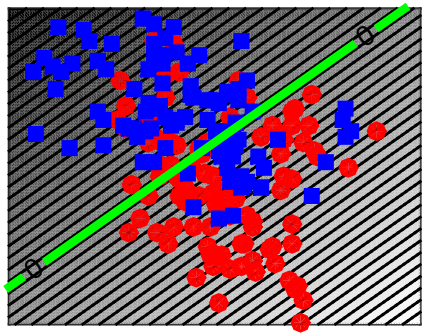




σ too small



nice σ



σ too large

Solveurs de SVM

- LibSVM (Multi-langage) :
  `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`
- ScikitLearn (Python)
  `http://scikit-learn.org/stable/modules/svm.html`
- PyML `http://pyml.sourceforge.net/#`

...

Introduction

But

- $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1\cdots N}$: ensemble de points étiquetés
- Objectif : apprendre à partir de $\mathcal{D}$ une fonction mathématique

$$\begin{array}{rccc} f: & \mathcal{X} & \longrightarrow & \mathcal{Y} \\ & \boldsymbol{x} & \longmapsto & \hat{y} = f(x) \end{array}$$

qui prédit la sortie $\hat{y}$ associée à tout point $\boldsymbol{x} \in \mathcal{X}$

Propriétés de l'apprentissage

- $\forall\ (\boldsymbol{x}_i, y_i) \in \mathcal{D}$, on veut que $f(\boldsymbol{x}_i)$ prédise la bonne valeur de y_i
- f doit pouvoir prédire les bonnes sorties pour des exemples futurs $\boldsymbol{x}_j$

Introduction

Exemple : classification d'objets dans des images





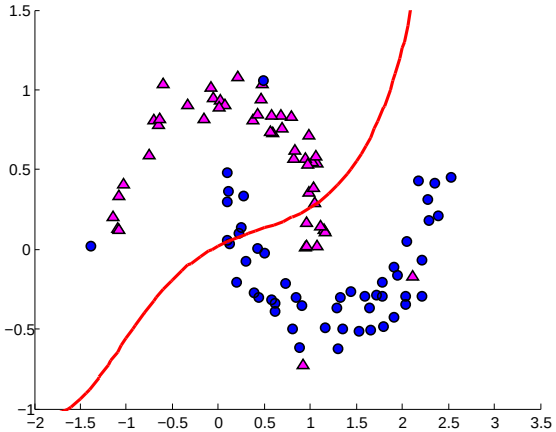
Modèles possibles pour ce problème et hyper-paramètres à régler

- Régression logistique : choix du modèle (linéaire ou polynomial), choix du paramètre de régularisation C
- SVM : choix de l'hyper-paramètre C, du noyau k et son paramètre
- ...

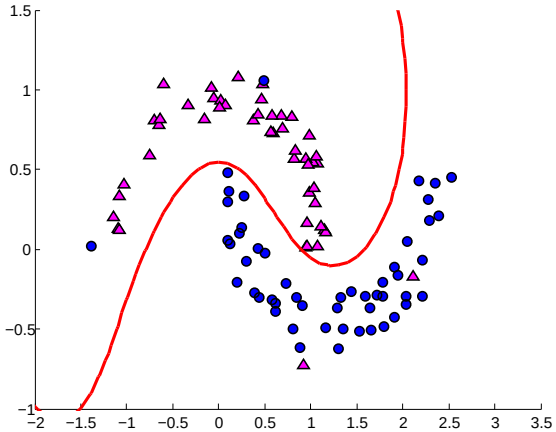
Sélection de modèle

Influence du choix des hyper-paramètres du modèle : exemple

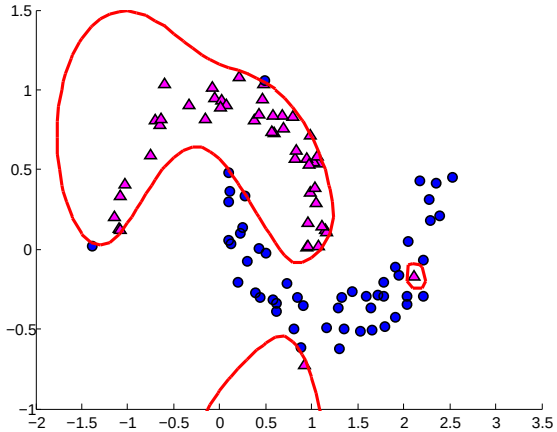
- Modèle SVM polynomial. Influence du choix du paramètre C



C trop petit
(sous-apprentissage)



C Correct



C trop grand
(sur-apprentissage)

- Choix de l'hyper-paramètre influe sur la solution

$\rightarrow$ Nécessité de sélectionner le "bon modèle"

Motivation

Objectifs

- Evaluation du modèle : mesure de performance
- Estimation de la capacité de généralisation du modèle
- Procédures pratiques de sélection de modèle

Remarques

- On se limitera aux problèmes de classification binaire
- Les points abordés s'appliquent à d'autres problèmes d'apprentissage

Matrice de confusion

Matrice de confusion pour la classification binaire

Prédiction $\hat{y}$/Vérite y	Positifs	Négatifs
Positifs	TP	FP
Négatifs	FN	TN
Total	N_+	N_-

- TP : nombre de positifs classés positifs (bonnes prédictions)
- FP : nombre de négatifs classés positifs (erreurs)
- FN : nombre de positifs classés négatifs (erreurs)
- TN : nombre de négatifs classés négatifs (bonnes prédictions)

Plusieurs critères peuvent être construits à partir de cette matrice

Mesures de performances pour la classification (1)

Critères classiques

- Taux d'erreur $= \frac{FP + FN}{TP + TN + FP + FN}$ $(\searrow\searrow)$
- Taux de bonne classification $= \frac{TP + TN}{TP + TN + FP + FN}$ $(\nearrow\nearrow)$ [a]

a. $(\nearrow\nearrow)$ signifie plus grand est le critère meilleur est le modèle

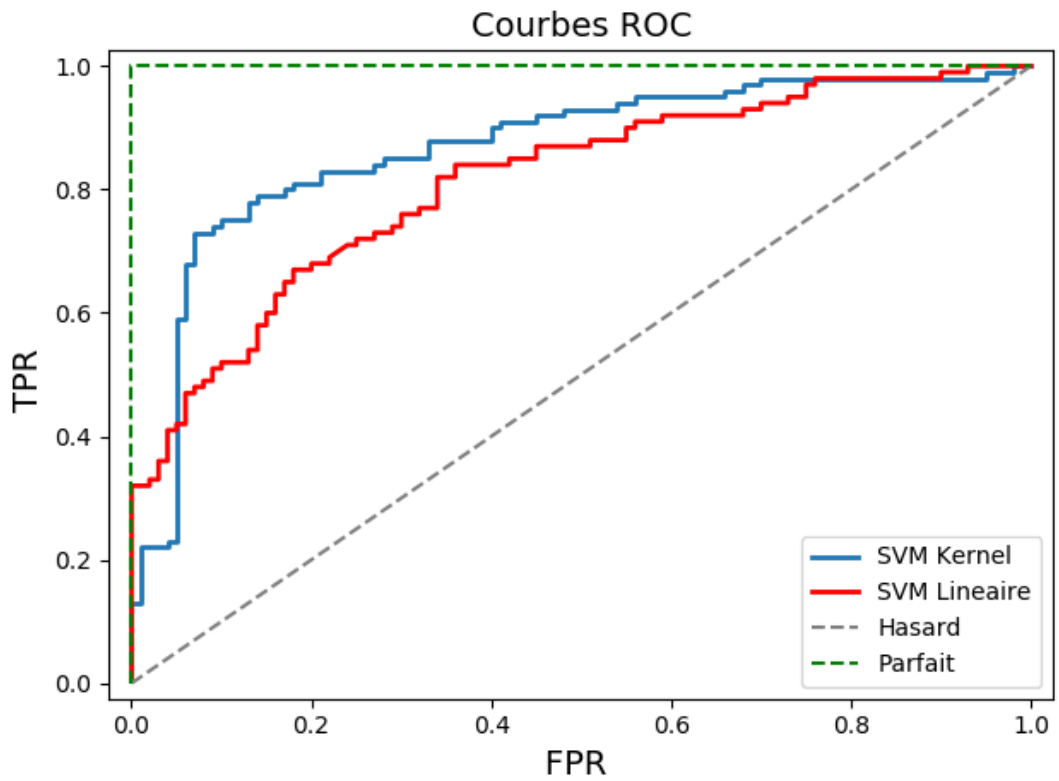
Critères utilisés en recherche d'information

- Precision $= \frac{TP}{TP + FP}$
- Rappel $=$ Taux de vrais positifs $= \frac{TP}{TP + FN}$
- F-Mesure $= 2\frac{\text{Precision} \times \text{Rappel}}{\text{Precision} + \text{Rappel}}$ $(\nearrow\nearrow)$

Mesures de performances pour la classification (2)

Courbe ROC

- C'est la courbe TPR = fonction(FPR)
- Permet de comparer différents modèles entre eux



Mesures de performances pour la classification (3)

Area Under the ROC Curve (AUC)

- Soit $\mathcal{D} = \{(\boldsymbol{x}_i, y_i = 1)\}_{i=1}^{n_+} \cup \{(\boldsymbol{x}_j, y_j = -1)\}_{i=1}^{n_-}$ et soit f la fonction de décision. L'AUC est définie par

$$\text{AUC} = \sum_{i=1}^{n_+} \sum_{j=1}^{n_-} \frac{\mathbb{I}\left[f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j)\right] + 0.5\,\mathbb{I}\left[f(\boldsymbol{x}_i) = f(\boldsymbol{x}_j)\right]}{n_+ n_-}$$

  avec $\mathbb{I}$ la fonction indicatrice

- L'AUC est comprise entre 0 et 1 (↗↗)
- Privilégie la fonction de décision telle que $f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j)$ $\forall\ (y_i = 1, y_j = -1)$

Fonction de coût

Intérêt

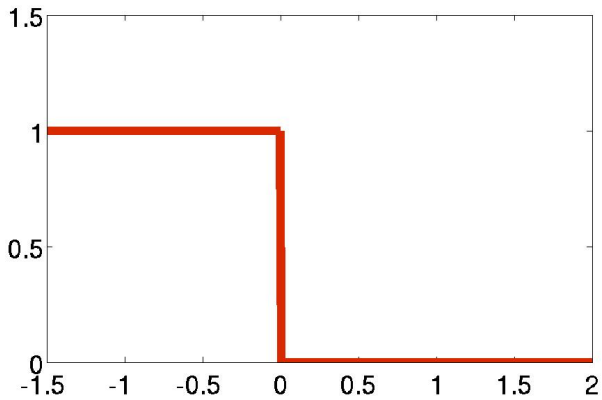
- Pénaliser f lorsque la prédiction $\hat{y} = f(x)$ est différente de y
- Le coût d'une bonne prédiction est plus faible que celui d'une erreur

Exemple : classification binaire

- On suppose $\mathcal{Y} = \{-1, 1\}$
- Coût 0 - 1

$$\ell(y, f(\boldsymbol{x})) = \mathbb{I}_{yf(\boldsymbol{x}) \leq 0} = \begin{cases} 0 & \text{si} \quad yf(\boldsymbol{x}) > 0 \\ 1 & \text{si} \quad yf(\boldsymbol{x}) \leq 0 \end{cases}$$

permet de mesurer le nombre d'erreurs de classification



Notion de fonction risque et apprentissage

Fonction risque

$$
\begin{aligned}
R(f) &= \Pr(\mathbb{I}_{Yf(X)\leq 0}) \\
&= \mathrm{Esp}_{X,Y}\ell(Y, f(X)) \\
R(f) &= \int_{\mathcal{X},\mathcal{Y}} \ell(y, f(\boldsymbol{x}))\mathrm{p}(\boldsymbol{x}, y)dxdy
\end{aligned}
$$

- C'est la probabilité d'erreur
- Mesure la capacité de généralisation de f (bien prédire sur des données futures)

Le problème d'apprentissage statistique

Trouver la fonction f^* qui minimise $R(f)$

$$f^* = \mathrm{argmin}_f \mathrm{Esp}_{X,Y}\ell(Y, f(X))$$

mais la solution f^* non atteignable car $\Pr(X, Y)$ n'est pas connue

Risque empirique

On a que les données $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1\cdots N}$. On définit le risque empirique

$$R_{\text{emp}}(f) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(\boldsymbol{x}_i))$$

Minimisation du risque empirique

- On cherche alors une fonction de décision

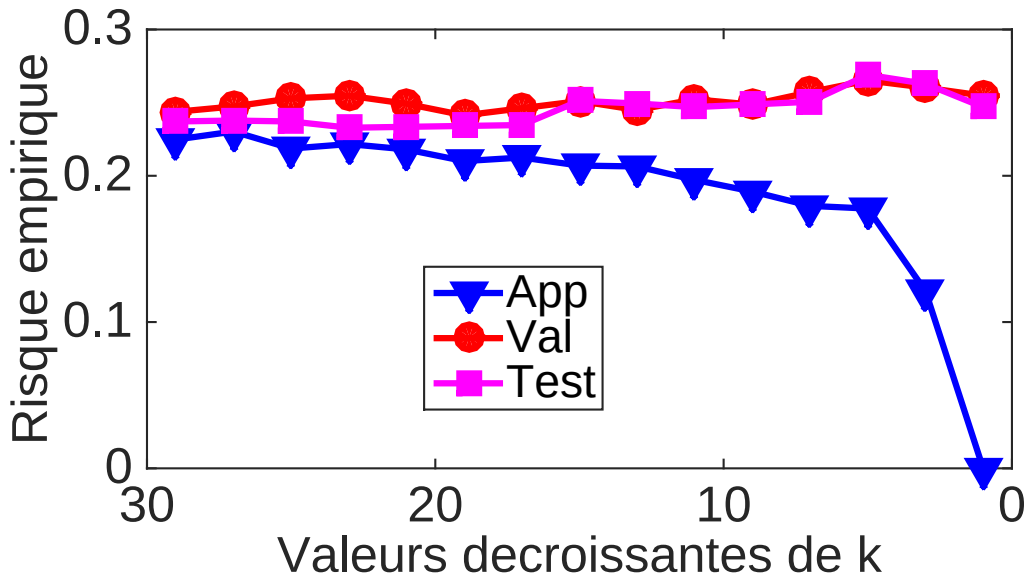
$$f_n \quad = \quad \text{argmin}_f R_{\text{emp}}(f)$$

- $R_{\text{emp}}(f_N)$ est le risque empirique correspondant à f_N. C'est une approximation du vrai risque $R(f_N) = \text{Esp}_{X,Y} \ell(Y, f_N(X))$

Risque empirique et sur-apprentissage

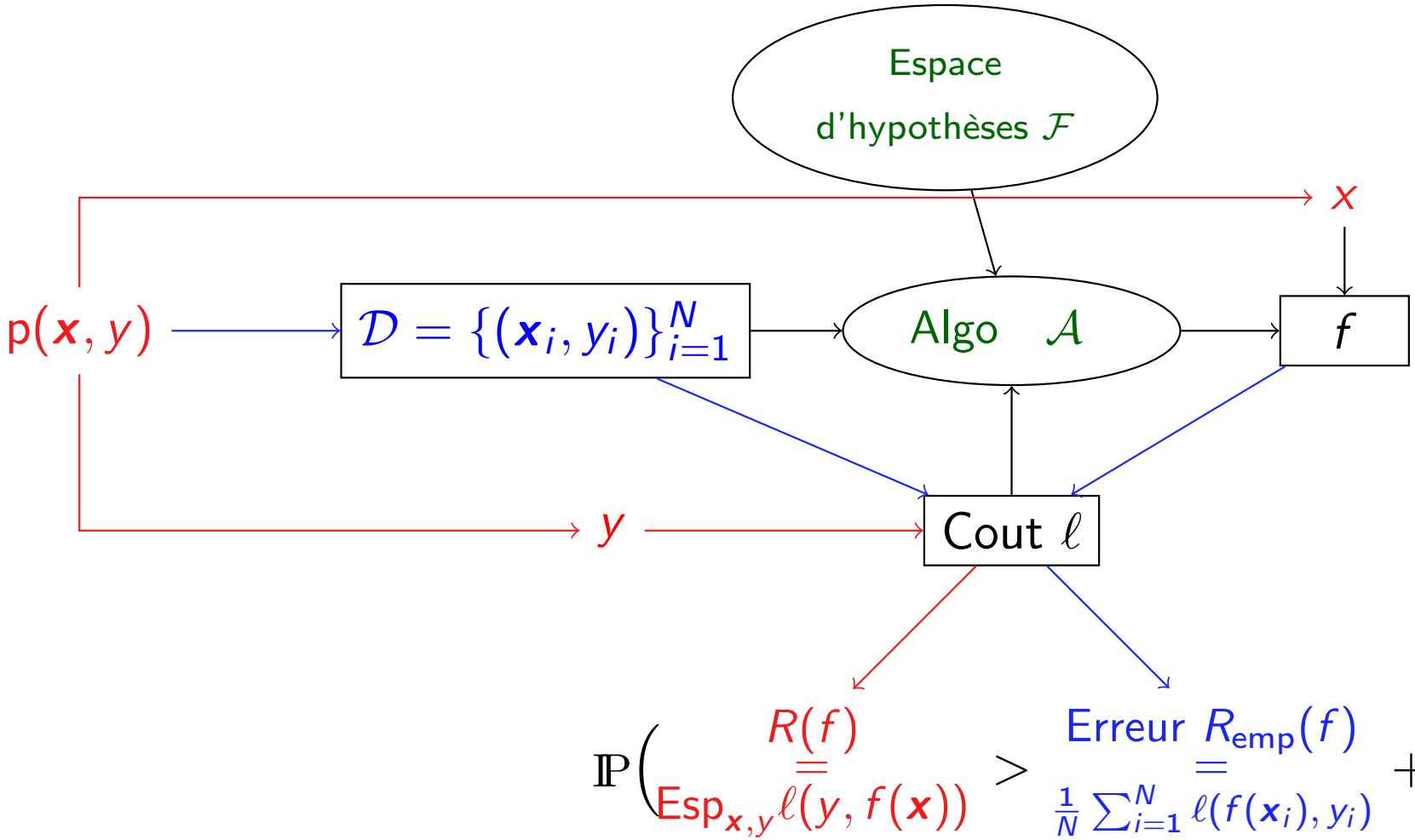
- Doit-on sélectionner f en se basant sur $R_{\text{emp}}(f_N)$?
- En général non !
- car on peut trouver une fonction de décision f_n suffisamment complexe telle que $R_{\text{emp}}(f_N) = 0$ mais qui donne un risque $R(f_N)$ élevé

Illustration sur une fonction de classification de type K-PPV



$\Longrightarrow$ Contrôler la complexité de la fonction f

Le paradigme de l'apprentissage statistique



Étant donné $\mathcal{D}$, trouver un modèle f dans une famille $\mathcal{F}$ (ex : fonctions linéaires, quadratiques ...) avec de bonnes propriétés de généralisation

Pourquoi l'apprentissage est possible

Borne sur l'erreur de généralisation

Soit les données $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1\cdots N}$. Soit $\mathcal{F}$ un espace de fonctions. Pour tout $f \in \mathcal{F}$, avec une probabilité $1 - \delta$ on a

$$R(f) \leq R_{\text{emp}}(f) + \mathcal{O}\left(\sqrt{\frac{h}{N}\log\frac{2eN}{h} + \frac{\log 2/\delta}{N}}\right)$$

$h > 0$ mesure la "complexité" de la classe de fonctions $\mathcal{F}$

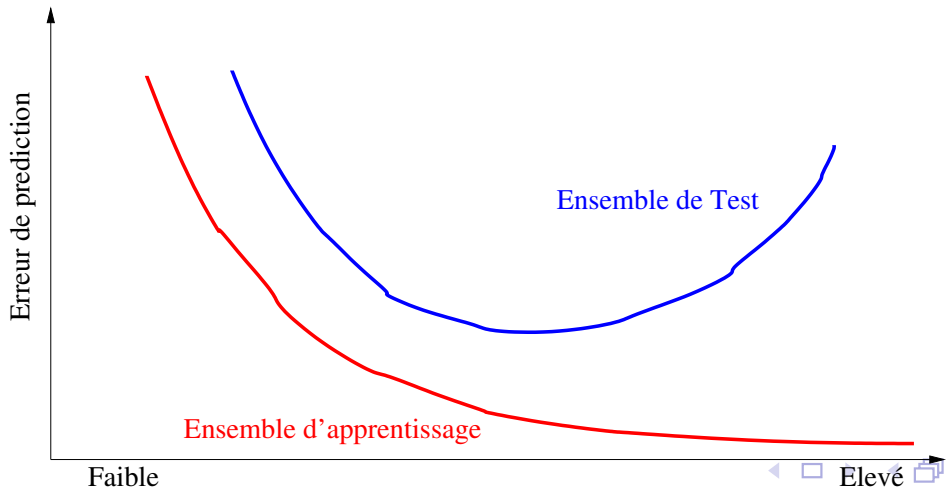
- la généralisation est possible si $h < \infty$
- plus $N >> h$ mieux c'est (le nombre de données croît avec la complexité du modèle)
- modèle linéaire $f(x) = \boldsymbol{w}^\top \boldsymbol{x} + b$ avec $\boldsymbol{w} \in \mathbb{R}^d$, $h = d + 1$

Illustration

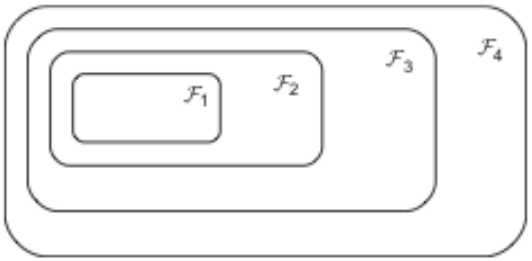
Généralisation / sur-apprentissage

$$R(f) \leq \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i), y_i) + \text{terme}(N, h(\mathcal{F}))$$

- Généralisation : capacité de f (élaboré sur $\mathcal{D}$) à donner de bonnes performances lorsqu'il est testé sur des données autres
- $R_{\text{emp}}(f)$ n'est pas un bon estimateur de la capacité de généralisation
- Le sur-apprentissage apparaît avec la complexité croissante de f



Contrôle de la complexité : régularisation



Soit $k_1 < k_2 < k_3 < \cdots$

On définit $\mathcal{F}_j = \{f : \Omega(f) \leq k_j\}$

$\Omega(f)$: fonction de régularisation

Exemple : $\Omega(f) = \|f\|^2$

Minimisation du risque empirique régularisé

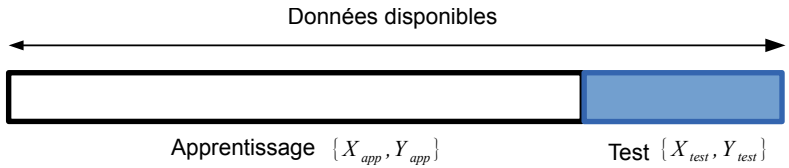
$$\min_f \frac{1}{n} \sum_{i=1}^{n} \ell(f(\boldsymbol{x}_i), y_i) + \lambda \, \Omega(f)$$

- $\lambda > 0$: hyper-paramètre de régularisation
- $\lambda >> 1 \rightarrow$ on privilégie un f de faible complexité

Exemple : SVM $\min_f \frac{1}{N} \sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i), y_i) + \lambda \|f\|^2$ avec la fonction coût $\ell(y, f(\boldsymbol{x})) = \max(0, 1 - yf(\boldsymbol{x}))$ et $\lambda = 1/C$

Paradigme ensemble test/ensemble d'apprentissage

Découper aléatoirement $\mathcal{D}_N$ en deux ensembles disjoints $\mathcal{D}_{app}$ et $\mathcal{D}_{test}$



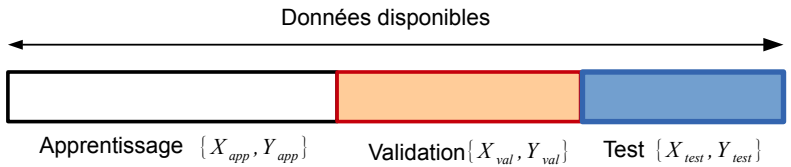
- $\mathcal{D}_{app} = \{(\boldsymbol{x}_i, y_i)\}_{i=1\cdots n_{app}}$: données servant à l'apprentissage de f
- $\mathcal{D}_{test} = \{(\boldsymbol{x}_i, y_i)\}_{i=1\cdots n_{test}}$: données servant à évaluer la capacité de généralisation (mesure de la performance) du modèle f

Remarques

- Plus n_{app} est grand mieux est l'apprentissage
- Plus n_{test} est grand meilleure est l'estimation de la performance en généralisation de f
- $\mathcal{D}_{test}$ n'est utilisé qu'une seule fois !

Ensemble de validation

Comment sélectionner le "bon" modèle sans toucher à $\mathcal{D}_{test}$?



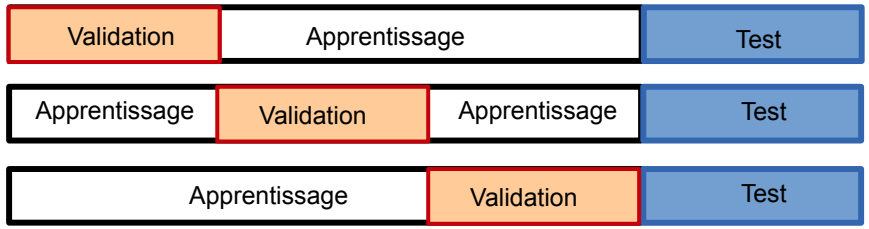
1. Découper aléatoirement $\mathcal{D}_N = \mathcal{D}_{app} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$
2. Apprendre chaque modèle possible sur $\mathcal{D}_{app}$
3. Evaluer sa performance en généralisation sur $\mathcal{D}_{val}$
4. Sélectionner le modèle qui donne la meilleure performance sur $\mathcal{D}_{val}$
5. Tester le modèle retenu sur $\mathcal{D}_{test}$

Remarque

- $\mathcal{D}_{test}$ n'est utilisé qu'une seule fois !

Validation croisée (K-fold validation)

Si le nombre de points de $\mathcal{D}_N$ est modeste ?



1. Découper aléatoirement $\mathcal{D}_n = \mathcal{D}_{app} \cup \mathcal{D}_{test}$
2. Découper ensuite aléatoirement $\mathcal{D}_{app} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_K$ en K ensembles
3. Pour $k = 1$ à K
   1. Mettre de côté $\mathcal{D}_k$
   2. Apprendre le modèle f sur les $K - 1$ ensembles restants
   3. Evaluer sa performance R_k en généralisation sur $\mathcal{D}_k$
4. Moyenner les K mesures de performance R_k

Sélection de modèle

Problématique

- Etant donné un ensemble de modèles $\mathcal{F} = \{f_1, f_2, \cdots\}$, choisir la fonction de décision qui donnerait les meilleures performances sur de futures données

Exemples de choix en fonction de la technique de classification

- Régression logisitique polynomiale : choix de l'ordre du polynôme
- SVM : choix de l'hyper-paramètre C
- Réseaux de neurones : nombre couches cachées, neurones par couche
- ...

Procédure pratique

Méthodologie générale

Entrées : famille de hyper-paramètres $\mathcal{F} = \{p_1, p_2, \cdots\}$ et $\mathcal{D}_N = \{(\boldsymbol{x}_i, y_i)\}_{i=1\cdots N}$

1. Découper les données $(\mathcal{D}_{appval}, \mathcal{D}_{test}) \leftarrow \texttt{SplitData}(\mathcal{D}, \texttt{options})$
2. Sélectionner le meilleur modèle : $f^* \leftarrow \texttt{Selection}(\mathcal{D}_{app}, \mathcal{F})$
3. $Perf \leftarrow \texttt{EvaluerPerf}(\mathcal{D}_{test}, f^*)$

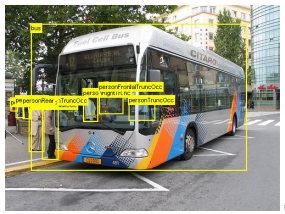
Procédure pratique (suite)

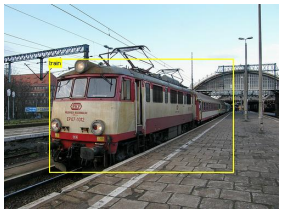
function $f^* \leftarrow \texttt{Selection}\,(\mathcal{D}_{app}, \mathcal{F})$

1. Redécouper les données $(\mathcal{D}_{app}, \mathcal{D}_{val}) \leftarrow \texttt{SplitData}\,(\mathcal{D}_{\texttt{appval}}, \texttt{options})$
2. Pour $p_i \in \mathcal{F}$
   1. Apprendre le modèle : $f_i \leftarrow \texttt{ApprendreModele}\,(\mathcal{D}_{app}, \texttt{p}_i)$
   2. $Perf\,(i) \leftarrow \texttt{EvaluerPerf}\,(\mathcal{D}_{val}, f_i)$
3. Sélectionner le meilleur hyper-paramètre : $p^* \leftarrow$ argmin $Perf$
4. $f^* \leftarrow \texttt{ApprendreModele}\,(\mathcal{D}_{appval}, \texttt{p}^*)$

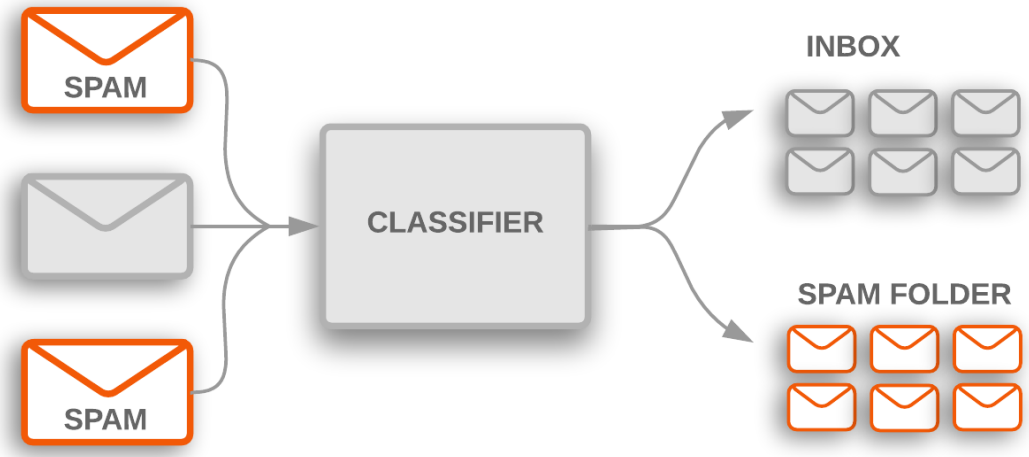
Traitement de données complexes

Classification d'images : bus $y = 1$ vs train $y = 0$

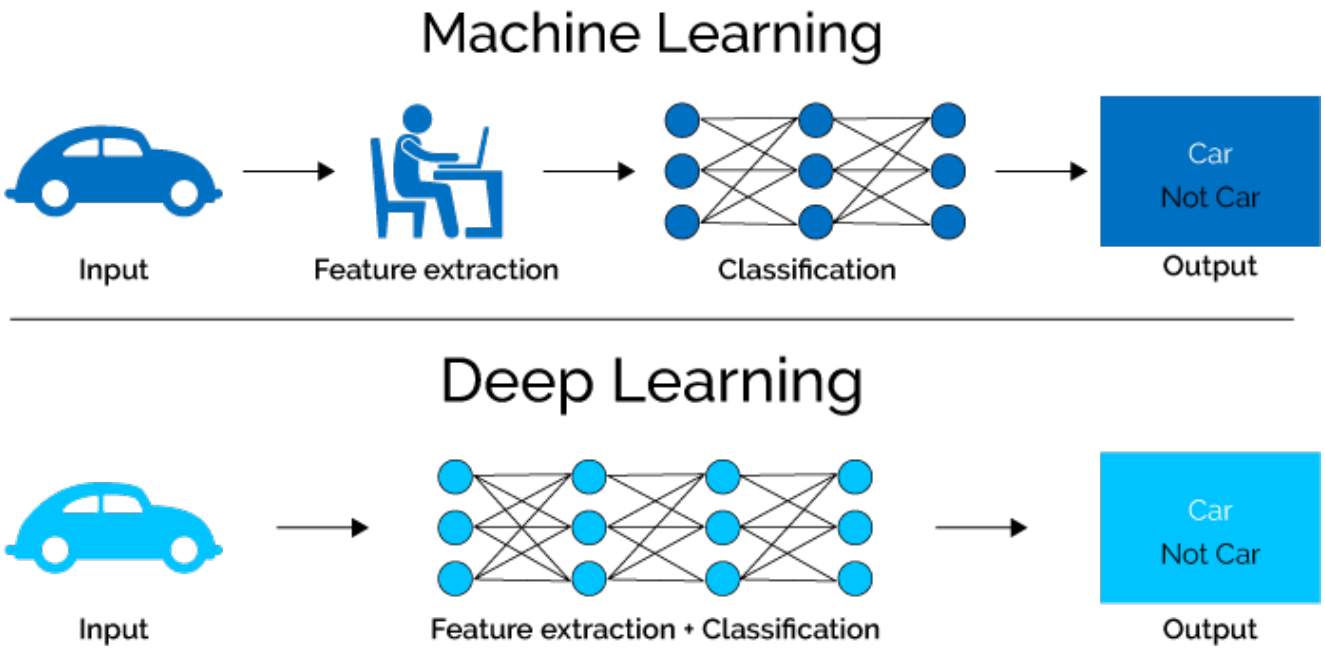



Classification de textes : spam vs non-spam

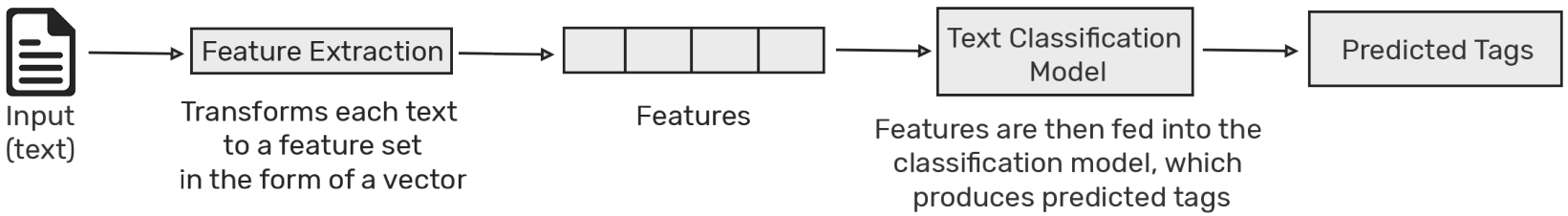


Démarche usuelle d'apprentissage

Image



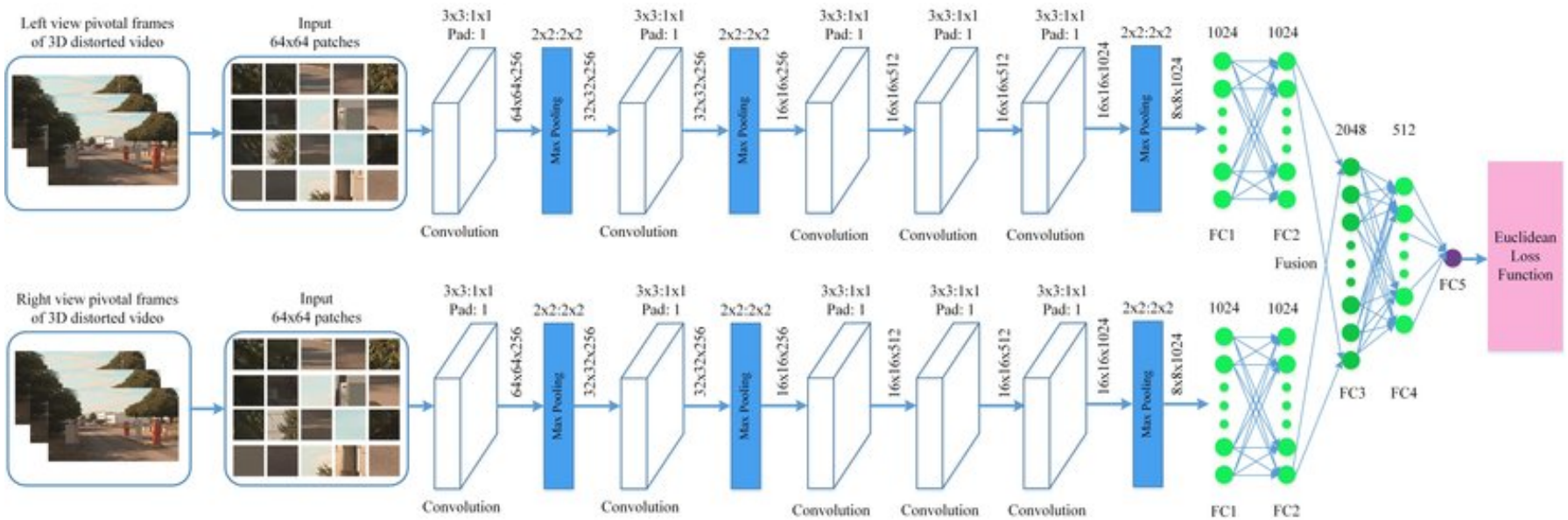
Texte



Deep Learning

End-to-end learning

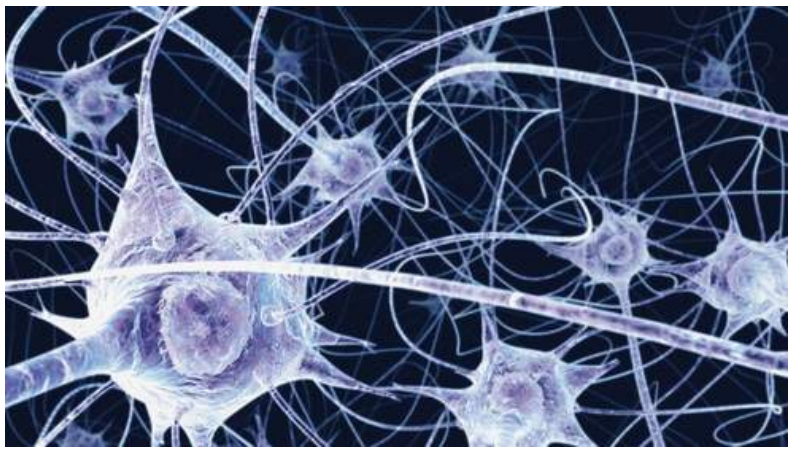
- Peut-on apprendre automatiquement et simultanément les caractéristiques $\Phi(\boldsymbol{x})$ et la fonction de classification f ?

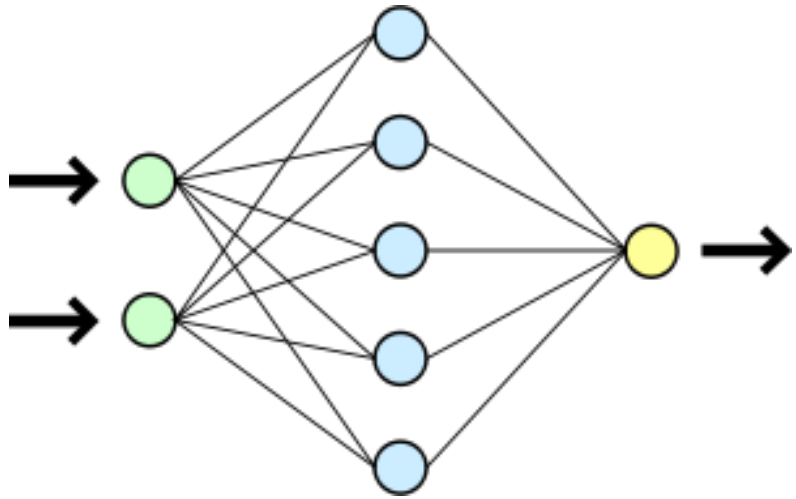


Réseaux de neurones

Principe général

- Composition de nombreuses fonctions élémentaires $f = f_1 \circ f_2 \circ \cdots \circ f_m$
- Apprendre à partir de données $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^N$
- Différents types de réseaux de neurones : perceptron multicouches, réseaux convolutionnels, réseaux récurrents ...





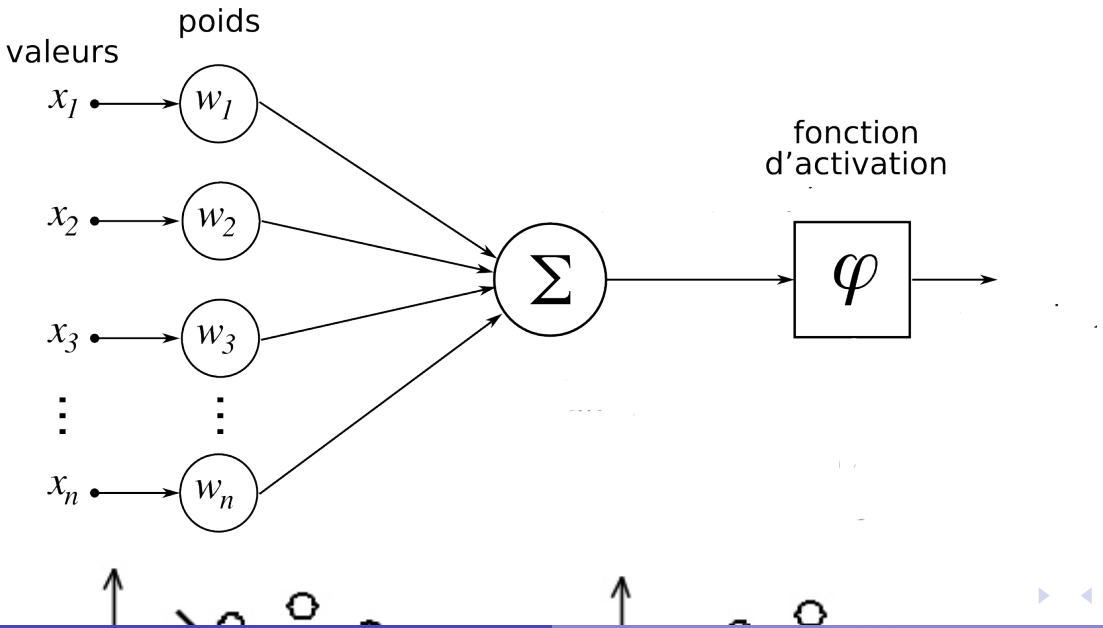
Le neurone formel [McCulloch et Pitts, 1943]

Unité élémentaire : neurone formel

- Entrée : $\boldsymbol{x}$, sortie : y
- Fonction de transfert entrée-sortie

$$y = \varphi\left(\boldsymbol{w}^\top \boldsymbol{x} + b\right)$$

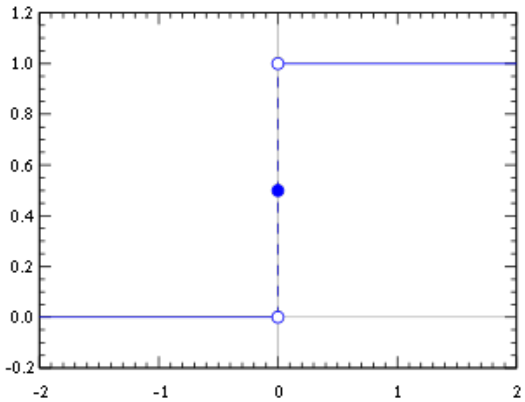
φ fonction d'activation non-linéaire ou linéaire



Le neurone formel [McCulloch et Pitts, 1943]

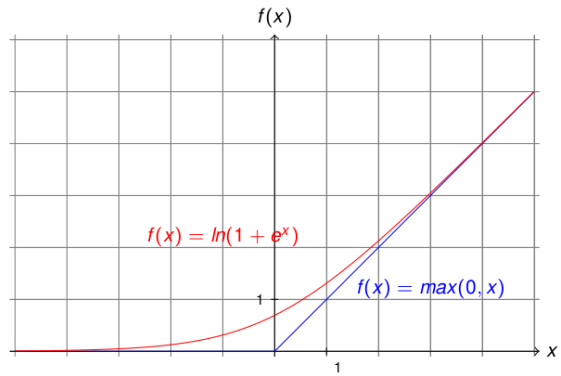
Fonctions d'activation

- fonction identité
- heaviside : $\varphi(\alpha) = 0$ si $\alpha < 0$, 1 sinon
- sigmoide : $\varphi(\alpha) = \frac{1}{1+e^{-\alpha}}$
- tanh : $\varphi(\alpha) = \frac{e^{\alpha}-e^{-\alpha}}{e^{x}+e^{-\alpha}} = \frac{e^{2\alpha}-1}{e^{2x}+1}$
- ReLU : $\varphi(\alpha) = \max(0, \alpha)$



heaviside

tanh

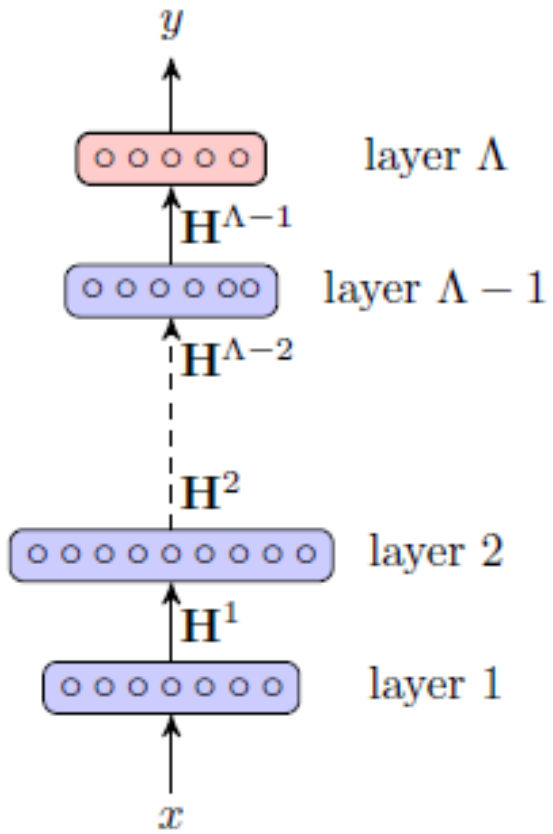


ReLU

Réseaux en couches

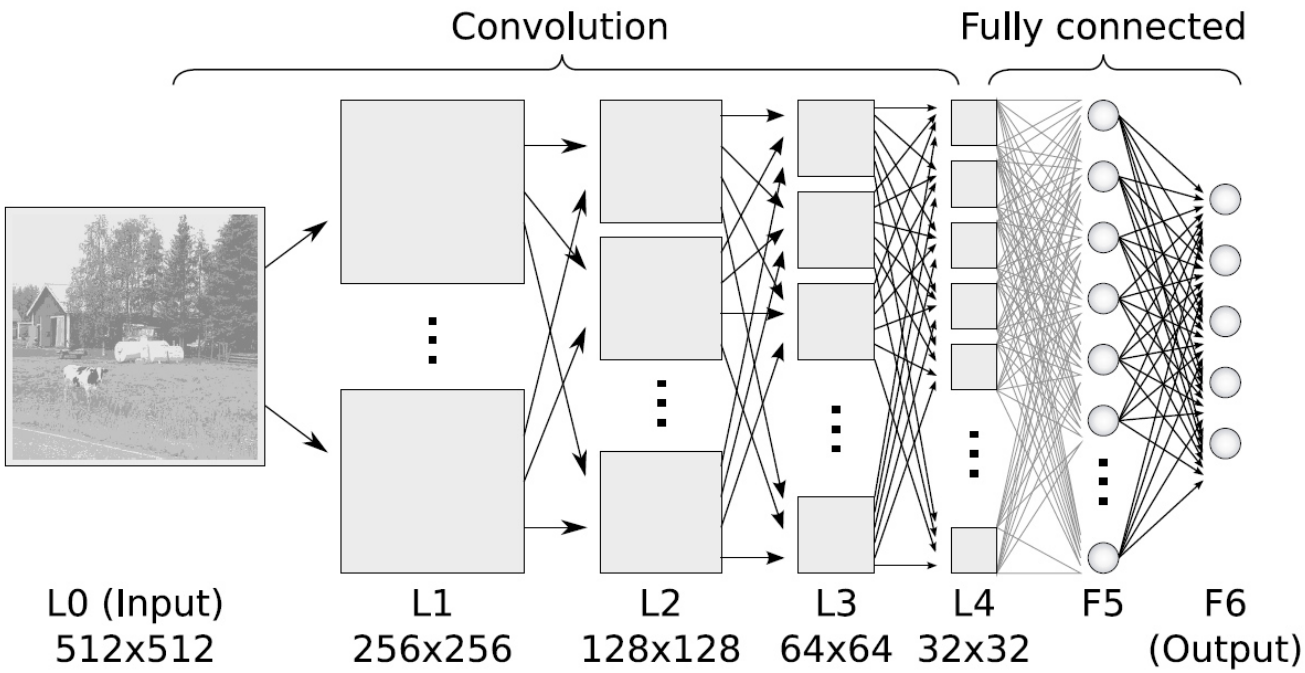
Profondeur du réseau

- Si plus d'une couche : couches dites « cachées », perceptron multicouches
- Si beaucoup de couches : architectures profondes



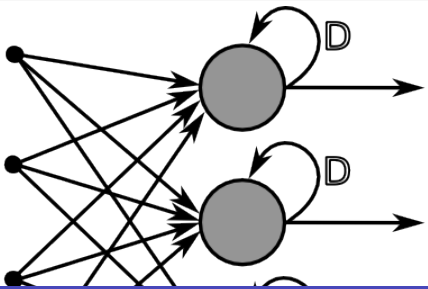
Réseaux convolutionnels

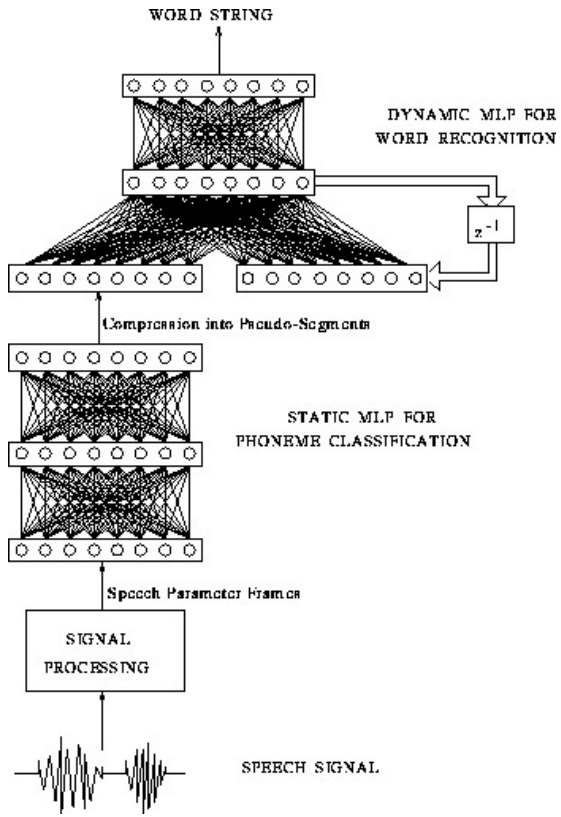
- Poids partagés, connexions locales
- Apprentissage de configurations particulières



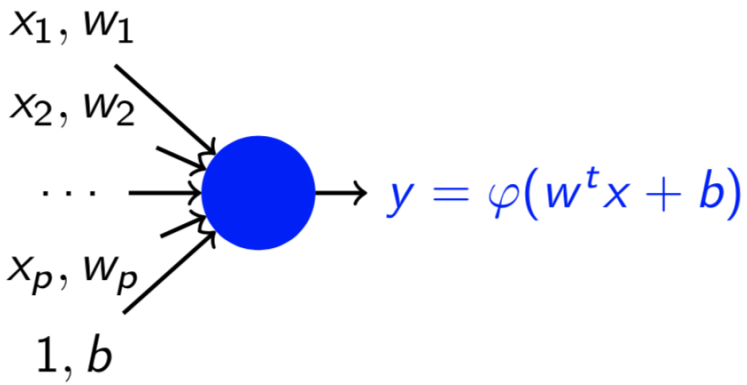
Réseaux récurrents

- Adapté aux séquences (textes, parole ...)
- Permet de prendre en compte le contexte
- On calcule $y(n)$ à partir de :
  - $x(n)$ les entrées courantes
  - $y(n-1)$ les sorties de l'exemple précédent (provenant d'une même séquence)
- Hypothèse $\simeq$ Markovienne





Réseau non linéaire à une couche



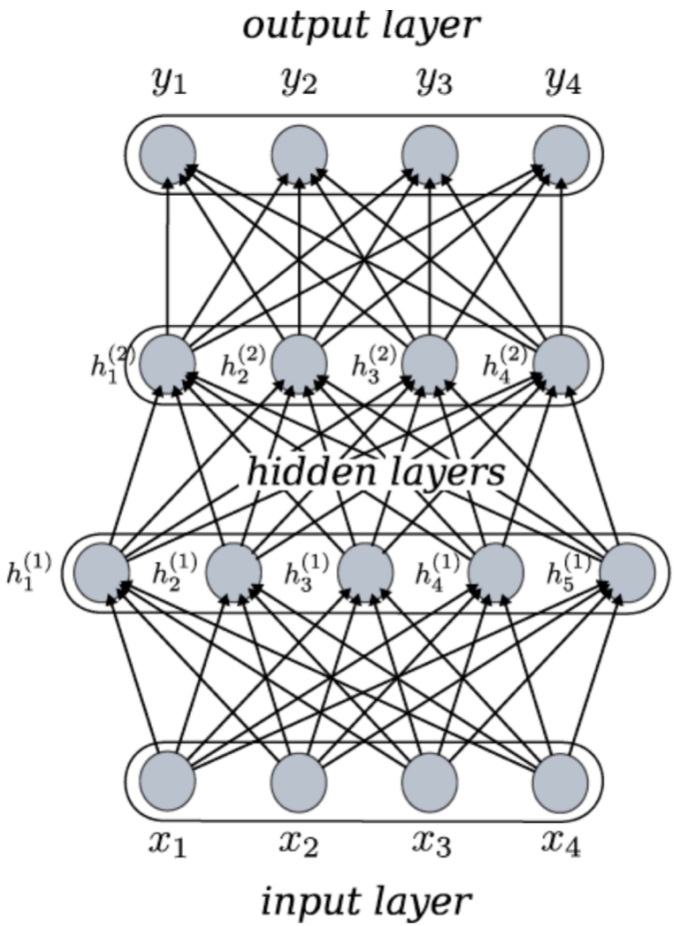
Calcul des paramètres

- Critère : $J(\boldsymbol{w}) = \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \varphi\left(\boldsymbol{w}^\top \boldsymbol{x}_i + b\right)\right)^2$
- Descente de gradient : $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k - \eta \nabla J(\boldsymbol{w}^k)$ avec

$$\nabla J(\boldsymbol{w}_t) \quad = \quad -\sum_{n=1}^{N} \left(y_i - \varphi(\boldsymbol{w}^\top + b)\right) \times \varphi'\left(\boldsymbol{w}^\top \boldsymbol{x}_i + b\right) \boldsymbol{x}_i$$

- Idem pour le paramètre b

Réseau à deux couches cachées



$$y = \varphi(W_3 h^{(2)})$$

$$h^{(2)} = \varphi(W_2 h^{(1)})$$

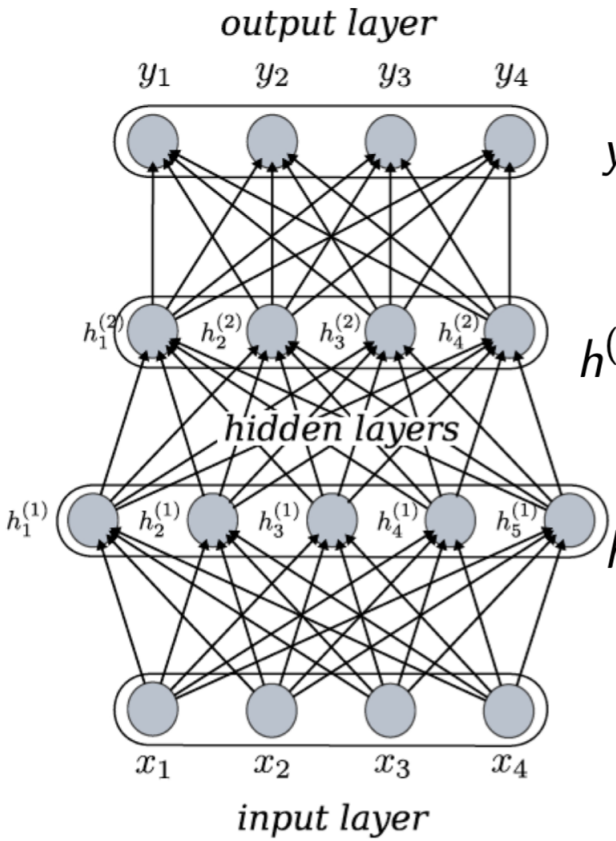
$$h^{(1)} = \varphi(W_1 \mathbf{x})$$

$$\mathbf{x}$$

Perceptron multi-couches

Utiliser la propagation arrière du gradient pour calculer les poids $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$

Réseau à deux couches : propagation arrière du gradient



$$y = \varphi(W_3 h^{(2)})$$
$$\uparrow$$
$$h^{(2)} = \varphi(W_2 h^{(1)})$$
$$\uparrow$$
$$h^{(1)} = \varphi(W_1 \mathbf{x})$$
$$\uparrow$$
$$\mathbf{x}$$

$$\nabla_{W_3} J = (y - y_a)\varphi'(W_3 h^{(2)})h^{(2}$$
$$\downarrow$$
$$\nabla_{W_2} J = \nabla_{h^{(2)}} J \varphi'(W_2 h^{(1)})h^{(1)}$$
$$\downarrow$$
$$\nabla_{W_1} J =$$

Mise à jour des paramètres

$$\mathbf{W}_\ell^{k+1} \leftarrow \mathbf{W}_\ell^k - \eta \nabla_{\mathbf{W}_\ell^k} J(\mathbf{W}_\ell^k) \quad \text{pour} \quad \ell = 1, 2, 3$$

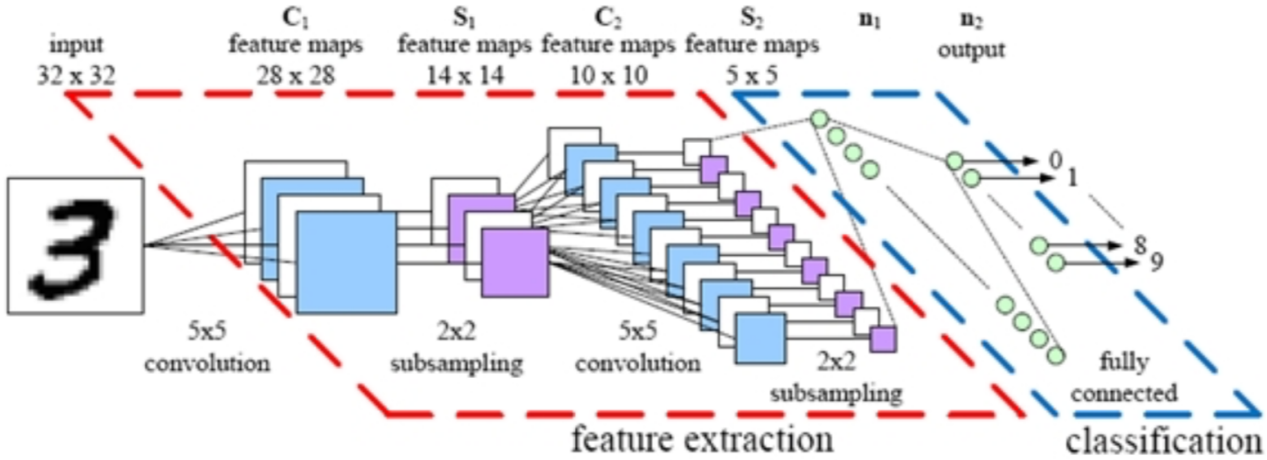
Estimation des poids : approche stochastique

En pratique, les poids sont optimisés par gradient stochastique

Algorithm 1 Backpropagation algorithm

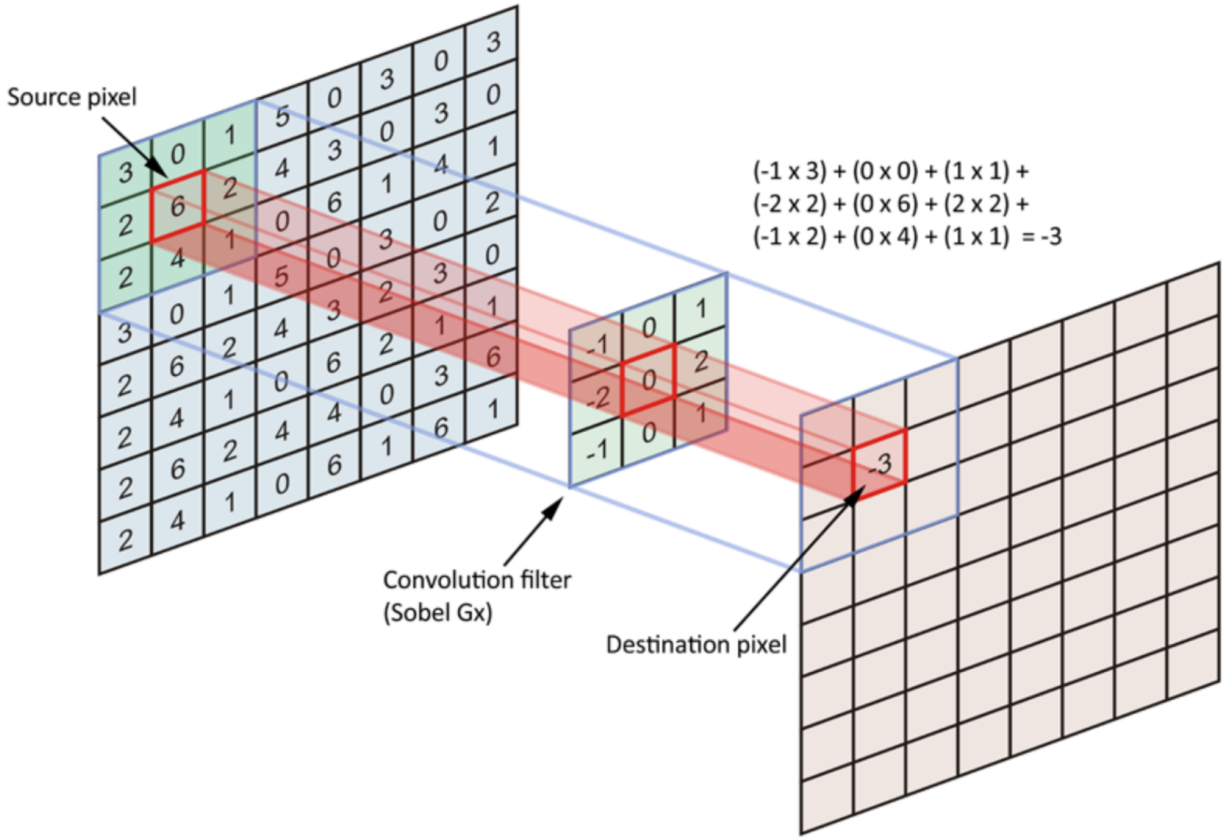
Initialiser η, les poids $\mathbf{W}_\ell$, fixer la taille L du lot (batch)
while non convergence **do**
  for $t = 1 \rightarrow \text{round}(N/L)$ **do**
    Tirer aléatoirement un lot de L points $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{L}$
    Calculer les gradients $\nabla_{\mathbf{W}_\ell^k} J(\mathbf{W}_\ell^k) \quad \forall\, \ell$ sur le batch
    Mettre à jour les poids $\mathbf{W}_\ell \leftarrow \mathbf{W}_\ell - \eta \nabla_{\mathbf{W}_\ell} J(\mathbf{W}_\ell^k) \quad \forall\, \ell$
  end for
end while

Et sur des images ?



Utiliser des réseaux convolutionnels (Convolutional Neural Network)

C'est quoi les CNN ?

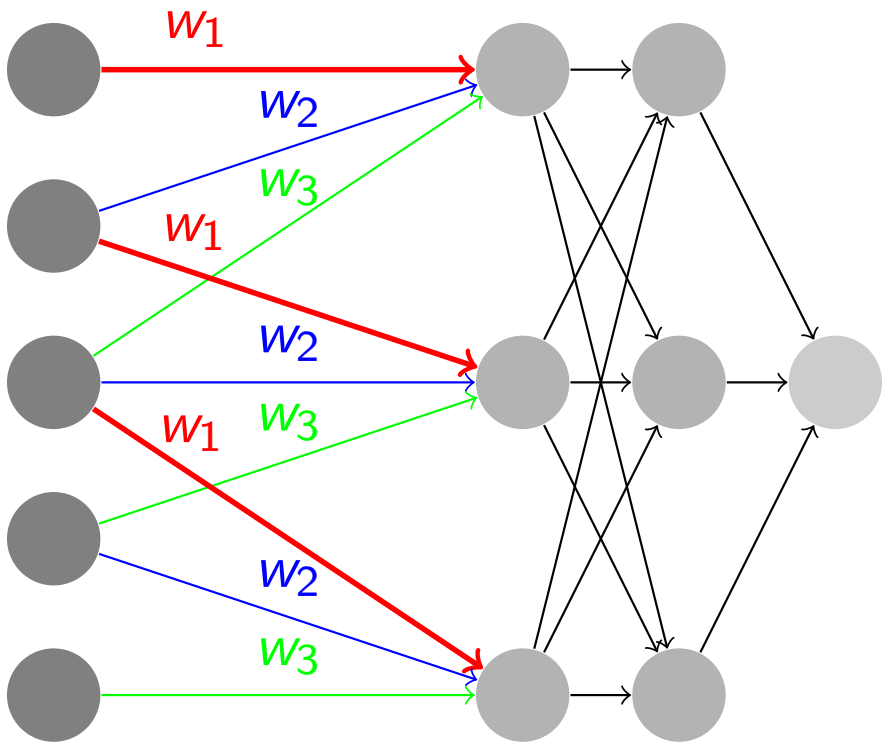


La convolution réduit le nombre de paramètres du réseau

C'est quoi les CNN (2) ?

Réseaux de neurones convolutionnels

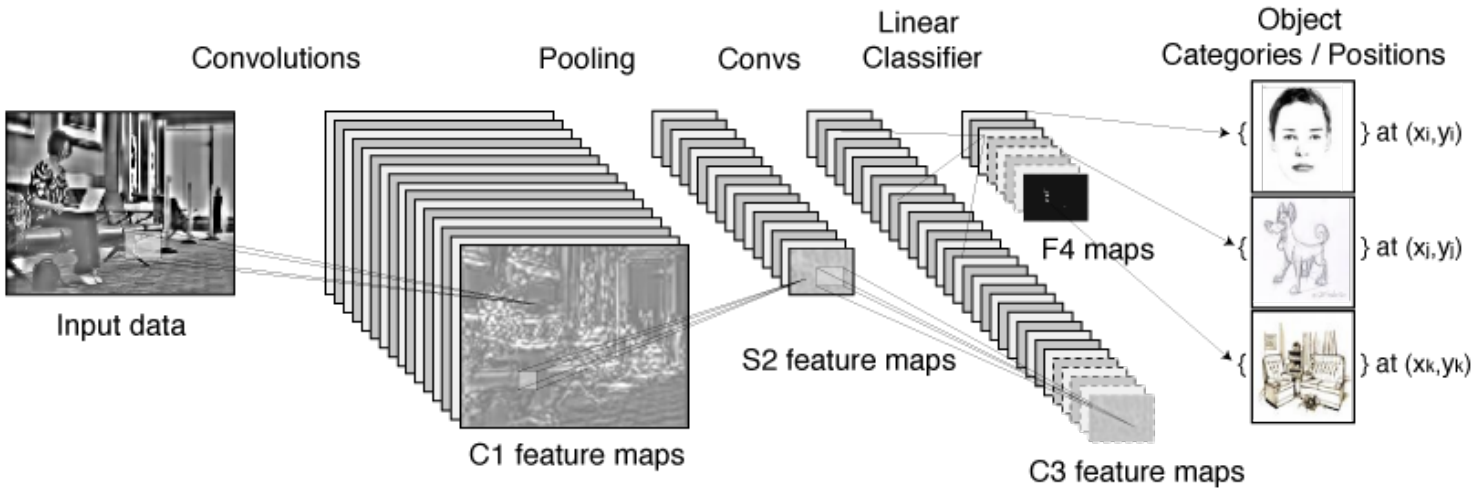
- Les poids sont partagés entre les neurones, **sauf les biais**



C'est quoi les CNN (3) ?

Couches convolutionnelles :

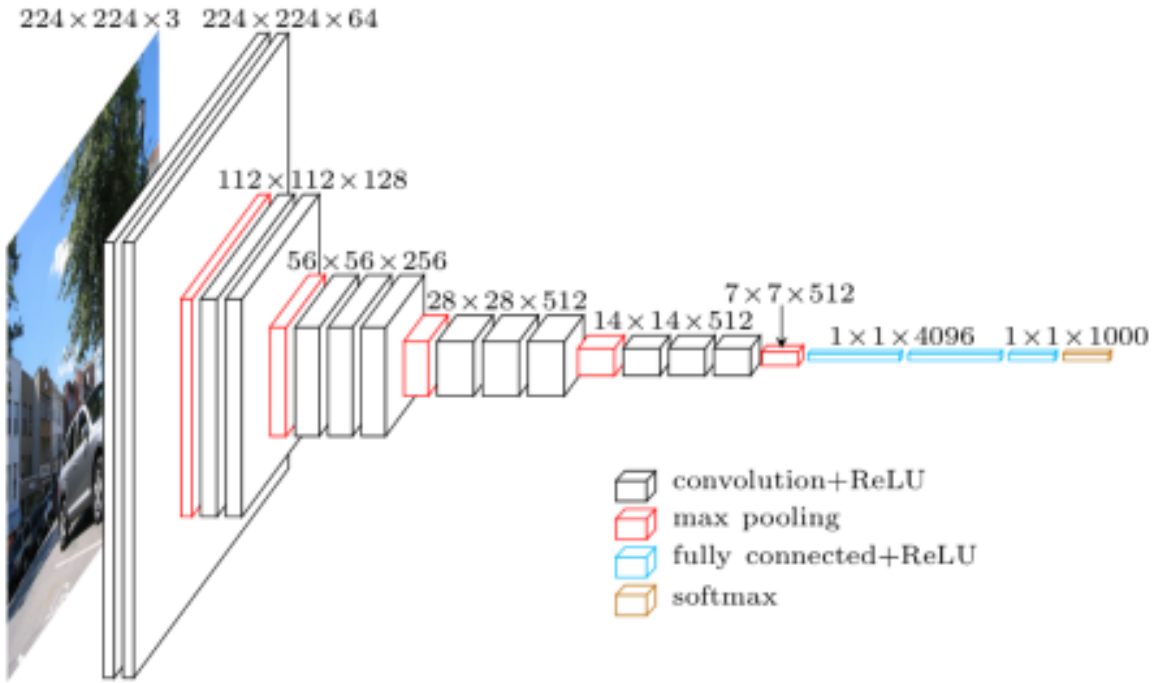
- Généralement plusieurs convolutions à chaque étage
- Alternance des couches de Conv et de Pooling pour concentrer l'information spatialement
- Couches hautes denses (ou Full Connex.) pour établir la classification



animation : `http://cs231n.github.io/convolutional-networks/`

C'est quoi les CNN (4) ?

Exemple de CNN : VGG16



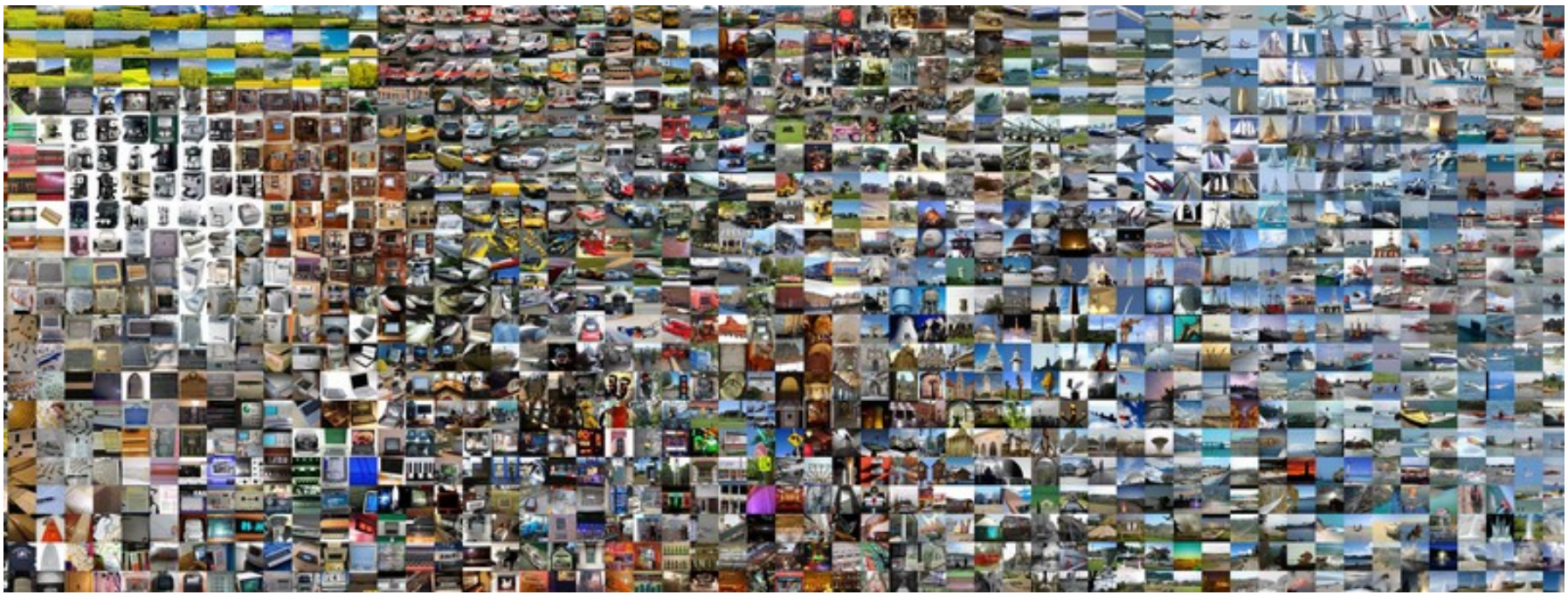
Caractéristiques

- 16 couches : 13 couches de conv. + pooling, 3 denses
- 138M de paramètres dont 124M (90%) pour les 3 couches denses

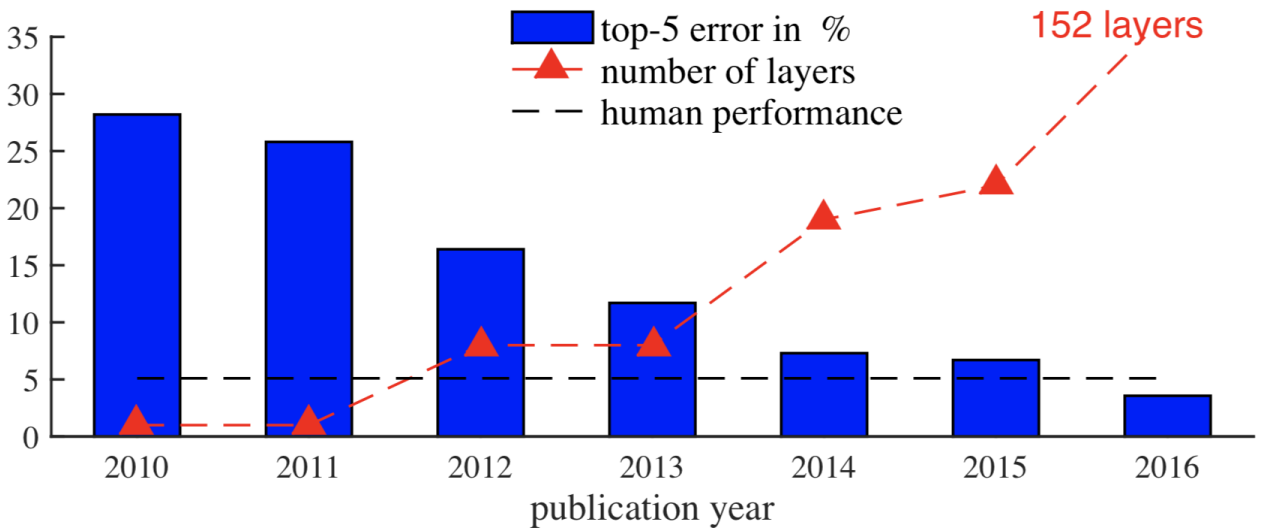
Les raisons du succès

ImageNet

- $>$ 14M d'images, 1000+ classes (objets, animaux, scènes, etc.)
- Images couleur $512 * 512$



Résultats sur Imagenet

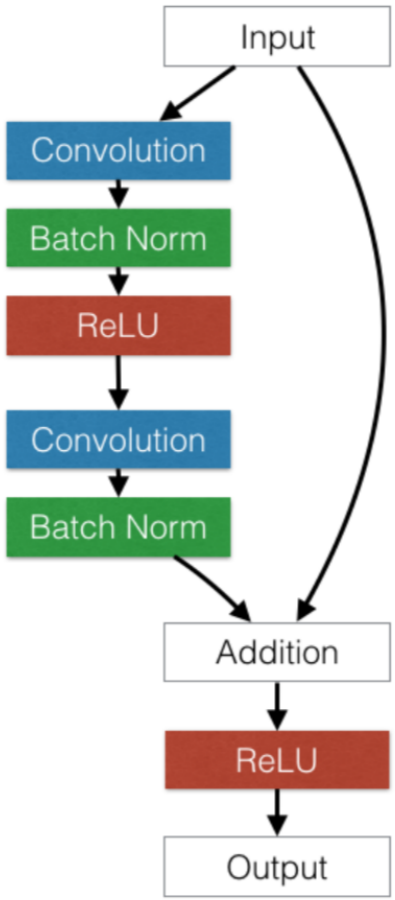


2012 AlexNet

2014 VGG

2016 Resnet (Residual Network)

Resnet

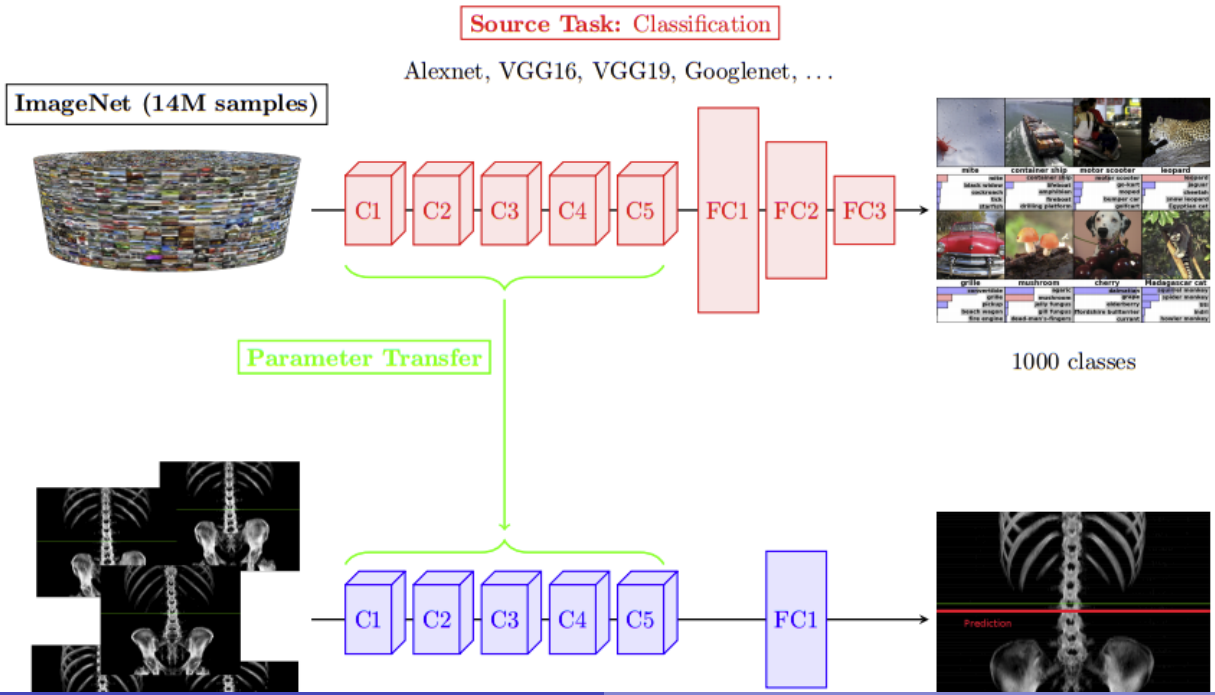

Input
Convolution
Batch Norm
ReLU
Convolution
Batch Norm
Addition
ReLU
Output


VGG-19
34-layer plain
34-layer residual

Transfer learning

Comment faire quand on a peu de données ?

- Utiliser un réseau pré-appris (AlexNet, VGG16, etc.) sur une très grosse base (ImageNet)
- Adapter les couches d'entrées et de sorties
- Réapprentissage sur le nouveau jeu de données



Mise en œuvre

∃ de nombreuses librairies

- TensorFlow (Google, python) `https://www.tensorflow.org`
- Keras (Google, python) `https://keras.io/` (+ Theano ou TF)
- pytorch (Facebook, python) `http://pybrain.org/` (basé sur torch)
- CNTK (Microsoft, Python, C++, C#)
  `https://github.com/microsoft/CNTK`

Exemple de code Keras pour vgg16 : `code/vgg16.py`