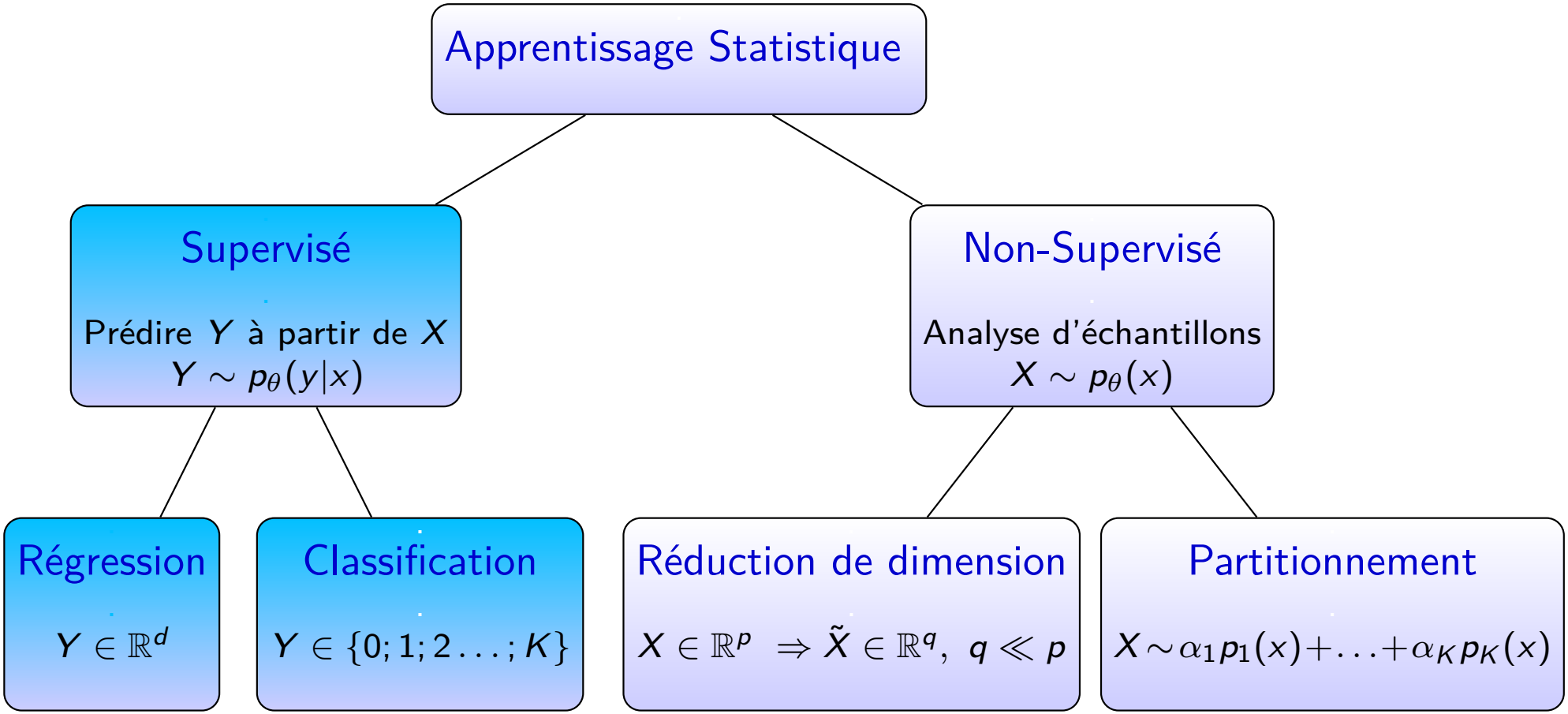
Statistique & Apprentissage

Paul-Henry Cournède

Amphi 8

Introduction à l'Apprentissage Statistique



IV.3 - Modèles à expansion de base

IV.3.a - Généralités

Soit X à valeurs dans $\mathbb{R}^p$, Y à valeurs dans $\mathbb{R}$, telles que le couple (X, Y) admette une densité.

On peut définir la densité conditionnelle $p_{Y|X=x}(y) = p(x, y)/p_X(x)$, $\forall x$ tel que $p_X(x) \neq 0$.

Définition : On appelle modèle linéaire généralisé à expansion de base un modèle tel qu'il existe une fonction de lien $g : \mathbb{R} \to \mathbb{R}$, des coefficients $\beta = (\beta_1, \ldots, \beta_M) \in \mathbb{R}^M$, et des fonctions de base $\psi_m : \mathbb{R}^p \to \mathbb{R}$, vérifiant :

$$g\left(\mathbb{E}(Y|X)\right) = \sum_{m=1}^{M} \beta_m \psi_m(X) \ .$$

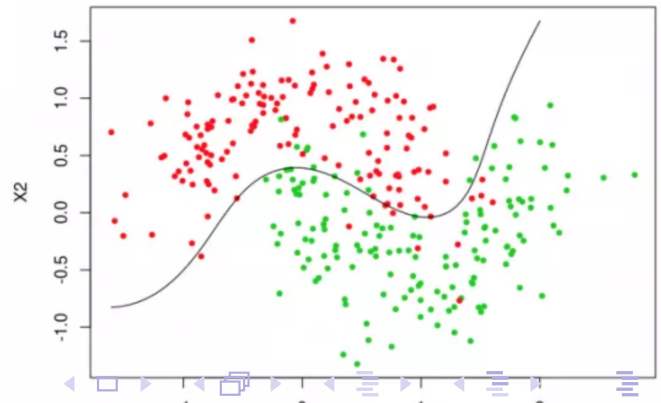
$\Longrightarrow$ Objectif : enrichir l'espace de représentation des données pour mettre en évidence des interactions particulières, modéliser des non-linéarités.

Exemples :

- Régression : $X = (X_1, X_2) \in \mathbb{R}^2$, $Y \in \mathbb{R}$. On prend : $Y = h(X) + \xi$, avec $\xi \sim \mathcal{N}(0; \sigma^2)$ et
$$h(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_{11} X_1^2 + \beta_{12} X_1 X_2 + + \beta_{22} X_2^2$$

- Classification : $X = (X_1, X_2) \in \mathbb{R}^2$, $Y \in \{0; 1\}$.

$$\text{logit}\left(\mathbb{P}(Y = 1|X = x)\right) = \sum_{j,k=0}^{K} \beta_{j,k} X_1^j X_2^k \ .$$



Remarques :

- Si les $\psi_m(X)$ ne font pas intervenir de paramétres inconnus et si on considère le modèle linéaire de régression :

$$Y = \sum_{m=1}^{M} \beta_m \psi_m(X) + \xi, \quad \xi \sim \mathcal{N}(0; \sigma^2).$$

Pour un jeu de données d'apprentissage $(x_i, y_i)_{1 \leq i \leq N}$, notons :

$$A = \begin{pmatrix} \psi_1(x_1) & \psi_2(x_1) & \dots & \psi_M(x_1) \\ \vdots & \vdots & & \vdots \\ \psi_1(x_N) & \psi_2(x_N) & \dots & \psi_M(x_N) \end{pmatrix} \quad , \quad B = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}$$

L'EMV du modéle de régression est donné en minimisant : $R(\beta) = \|A\beta - B\|^2$.

Si A est injective, on a :

$$\hat{\beta} = (A^t A)^{-1} A^t B$$

$$\hat{\sigma}^2 = \frac{1}{N} R(\hat{\beta}) = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \sum_{m=1}^{M} \hat{\beta}_m \psi_m(x_i) \right)^2 .$$

- Expansion de base = Dictionnaire : on peut parfois combiner plusieurs types de décompositions fonctionelles très variées pour notre modèle : potentiellement large.
$\Longrightarrow$ lors de l'apprentissage à partir d'un jeu de données $(x_i, y_i)_{1 \leq i \leq N}$, on peut utiliser le Lasso pour sélectionner les éléments du dictionnaire les plus importants :

$$\hat{\beta} = \arg\min \frac{1}{N} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} \beta_m \psi_m(X)) + \lambda \|\beta\|_1$$

Définition : On appelle modèle additif généralisé un modèle à expansion de base de la forme :

$$h(X) = \beta_0 + \sum_{j=1}^{p} \beta_j \psi_j(X_j) \ ,$$

$\Longrightarrow$ On décompose la contribution de chaque variable de façon additive.

On peut aussi décomposer les ψ_j :

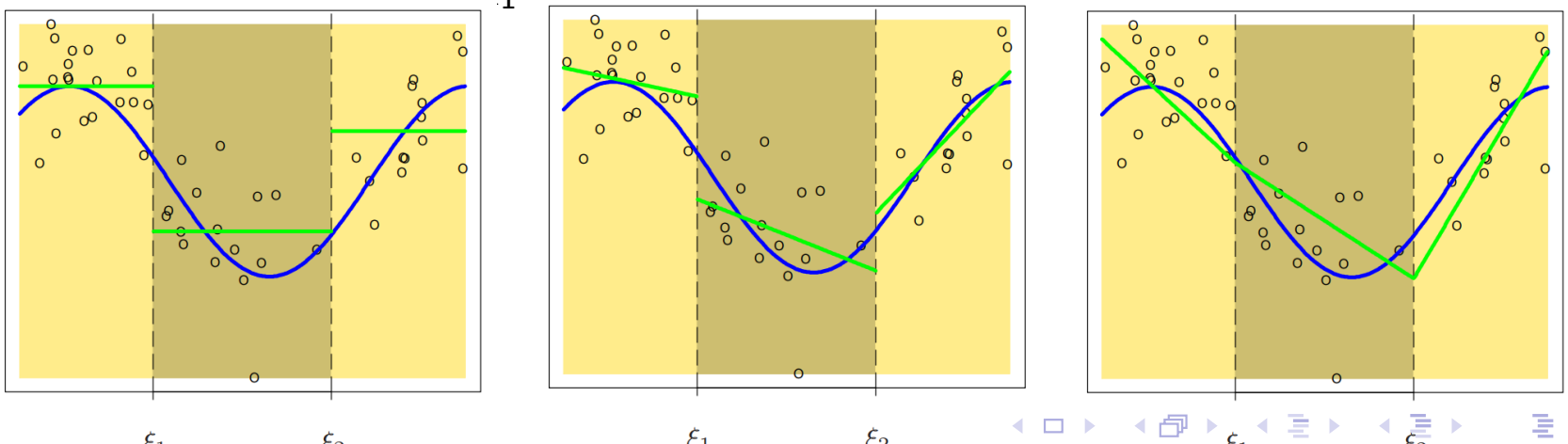
$$h(X) = \beta_0 + \sum_{j=1}^{p} \sum_{m=1}^{M_j} \beta_{jm} \psi_{jm}(X_j)$$

IV.3.b - Expansion de bases polynomiales par morceaux

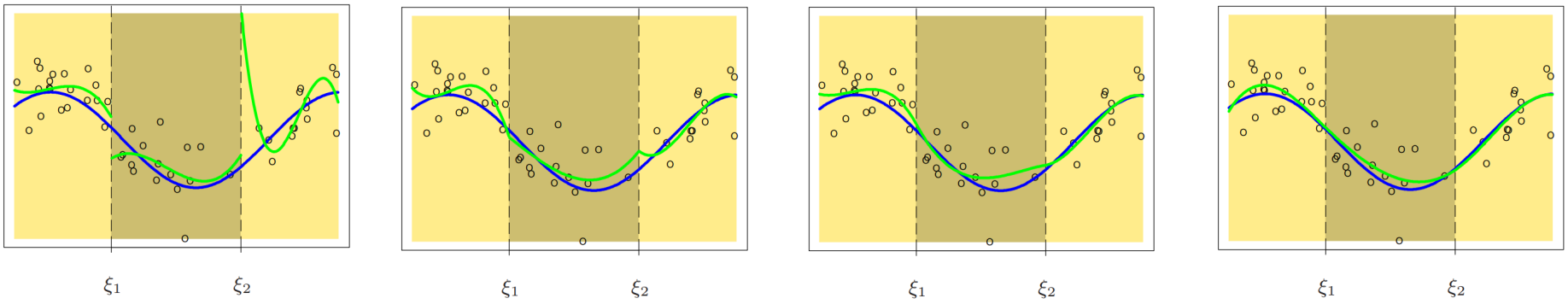
On considère un modèle additif généralisé où les éléments de base sont des fonctions polynomiales par morceaux.

Soit $\mathcal{D}_j$ le domaine de variation de la composante X_j : on divise $\mathcal{D}_j$ en une partition d'intervalles $I_1, \ldots, I_{N_j}$ et on prend :

$$\psi_j(X_j) = \sum_{k=1}^{N_j} \mathbb{I}_{I_k}(X_j) \psi_{jk}(X_j) \quad \Longrightarrow \ \psi_{jk} \text{ polynome sur } I_k$$



Exemple : Polynomes cubiques par morceaux. On n'impose aucune contrainte, une contrainte de continuité, une contrainte d'être $\mathcal{C}^1$, une contrainte d'être $\mathcal{C}^2$:



Définition : On appelle spline cubique un polynome cubique par morceaux qui soit $\mathcal{C}^2$.
$\Longrightarrow$ technique très flexible de modélisation des données.

Proposition : Soit $K+1$ intervalles de $\mathbb{R}$, et $\xi_1, \ldots, \xi_K$ les points frontières. Les splines cubiques sur $\mathbb{R}$ sont générées par la base :

$$h_1(X) = 1, \quad h_2(X) = X, \quad h_3(X) = X^2, \quad h_4(X) = X^3$$
$$h_5(X) = (X - \xi_1)_+^3, \quad \ldots, \quad h_{K+4}(X) = (X - \xi_K)_+^3$$

En effet : $\mathcal{S}$, l'ensemble des splines cubiques sur les $K+1$ intervalles est un espace vectoriel de dimension : $dim = \underbrace{4 * (K+1)}_{\text{dim 4 par intervalle}} - \underbrace{3 * K}_{\text{3 contraintes de continuité par frontière}} = K + 4.$

Les fonctions $\mathcal{P}$ engendrées par la base $\{h_1, \ldots, h_{K+4}\}$ forment un espace vectoriel de dimension $K+4$. Par ailleurs, si $\psi \in \mathcal{P}$, $\psi \in \mathcal{S}$. Donc $\mathcal{P} \subset \mathcal{S}$, et espace vectoriel de même dimension donc $\mathcal{P} = \mathcal{S}$.

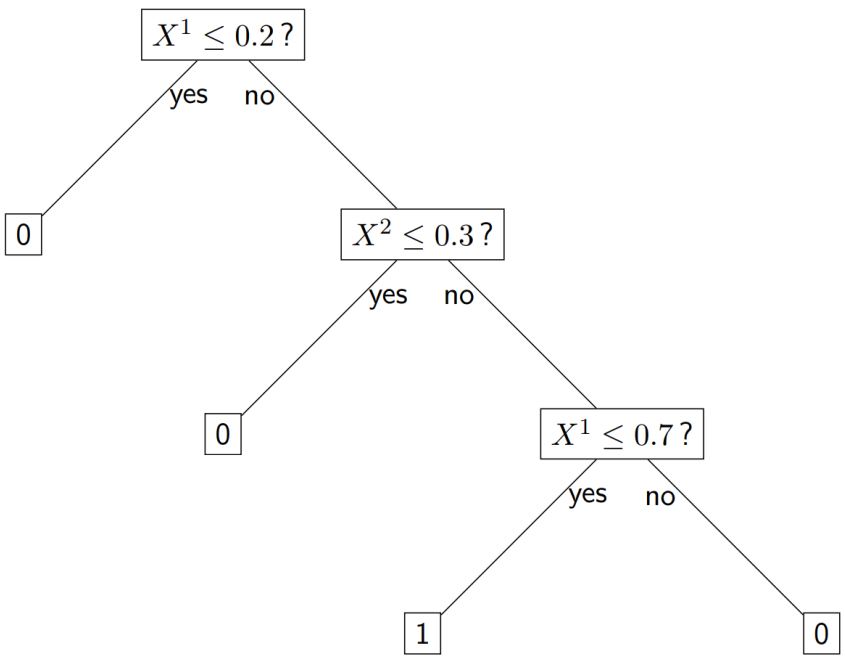
Conséquence : Si les ξ_k sont connus, résolution explicite par les formules de la régression linéaire.

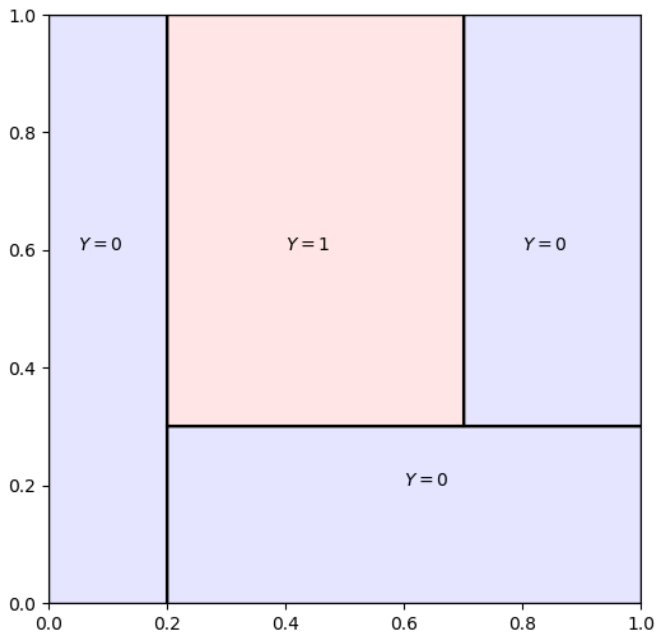
IV.3.c - Arbres de décision

Définition : Soit X à valeurs dans $\mathcal{X} \subset \mathbb{R}^p$ et Y à valeurs dans $\mathcal{Y} \subset \mathbb{R}$. On appelle arbre de décision un modèle tel que

$$\mathbb{E}(Y|X) = \sum_{m=1}^{M} \beta_m \mathbb{I}_{Z_m}(X)$$

où $\{Z_m \subset \mathbb{R}^p, 1 \leq m \leq M\}$ est une partition de $\mathcal{X}$ qui peut être définie à partir d'un arbre binaire, dans lequel chaque noeud partitionne l'espace à partir d'une règle de décision sur l'une des variables explicatives.
Les noeuds terminaux sont les feuilles de l'arbre.





- Modèle pour la classification ou la régression
- Avantage : Interprétabilité du modèle

Procédure d'estimation On construit une suite de partitions $(\mathcal{P}_m)$: $\mathcal{P}_m = \{Z_1, \ldots, Z_m\}$

$\mathcal{P}_{m+1}$ se déduit de $\mathcal{P}_m$ en coupant une des zones en 2 à partir d'une règle décision ($x_j \leq \delta_j$?) sur une des variables.

A chaque itération k, il faut déterminer :

- La région que l'on va partitionner I_k ;
- Sur quelle variable on va partitionner j_k ;
- Quel sera le seuil de partitionnement δ_{j_k}.

On cherche donc à minimiser :

$$J\left((I_k, j_k, \delta_k)_{1\leq k\leq M-1}, \beta\right) = \sum_{m=1}^{M} \sum_{i:x_i\in Z_m} L\left(y_i, \beta_m\right) \ .$$

Pour la régression et la fonction de perte quadratique, on a pour la partition $\mathcal{P}_M$:

$$\sum_{m=1}^{M} \sum_{i:x_i\in Z_m} (y_i - \bar{y}^m)^2 \leq \sum_{m=1}^{M} \sum_{i:x_i\in Z_m} (y_i - \beta_m)^2 \ \ ,\forall \beta_1, \ldots, \beta_M$$

où $\bar{y}^m$ est la valeur moyenne des y_i pour les x_i qui appartiennent à Z_m : $\bar{y}^m = \dfrac{\sum_{i:x_i\in Z_m} y_i}{|\{i : x_i \in Z_m\}|}$,

$\Longrightarrow$ Problème d'optimisation se ramène à un problème d'optimisation sur $(I_k, j_k, \delta_k)_{1\leq k\leq M-1}$.

$\Longrightarrow$ Recherche itérative en commençant par $Z_1 = \mathcal{X}$ puis en testant les différents cas possibles et en résolvant pour chacun d'eux un probème d'optimisation unidimensionnel.

Remarque : Pour la classification, fonction de perte = entropie croisée : $L(y_i, \beta_m) = -y_i \ln(\beta_m)$.

β_m est également donné par la valeur moyenne des y_i pour $x_i \in Z_m$.

$\Longrightarrow$ la procédure cherche à chaque itération à maximiser la pureté de chaque zone.

IV.3.d - Bagging et Random Forest

Soit un ensemble d'apprentissage $\mathcal{T} = (x_i, y_i)_{1<i<N}$.

Bagging : Pour $b = 1, \ldots, B$:

(i) on tire avec remise N échantillons dans $\mathcal{T}$, soit $\mathcal{T}_b$ l'ensemble d'apprentissage obtenu.

(ii) on entraîne un arbre de décision $\hat{h}_b$ sur $\mathcal{T}_b$.

La fonction de prévision est alors obtenue en moyennant les différents arbres appris :

$$\hat{h}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{h}_b(x) \quad \text{ou vote majoritaire pour classification.}$$

Remarques :

- Idée : combiner différents modèles permet de compenser leurs erreurs : méthodes ensemblistes (sagesse des foules).
- Efficace même si chacun des modèles n'est pas très bon
  $\Longrightarrow$ pour les arbres, on peut se contenter d'arbres de faible profondeur.
- L'apprentissage des différents modèles peut se faire en parallèle.
- Les ensembles $\mathcal{T}_b$ sont appelés échantillons bootstraps.

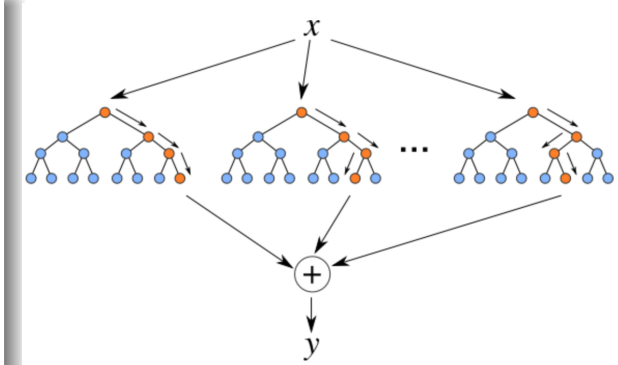
Random Forest (Forêt aléatoire) :

$\Longrightarrow$ tree bagging + méthode du sous-espace aléatoire

À chaque itération de l'algorithme d'apprentissage de l'arbre de décision h_b

$\Longrightarrow$ on sélectionne aléatoirement q features parmi les p, et la variable servant au partitionnement ("j_k") est choisie parmi ces q variables (typiquement $q < \sqrt{p}$).

- Méthode particulièrement simple et robuste.



IV.3.e - Réseaux de Neurones

Définition : Un perceptron à $M+1$ couches est une fonction $\psi : \mathcal{X} \subset \mathbb{R}^p \to \mathcal{Y} \subset \mathbb{R}^K$, telle qu'il existe :

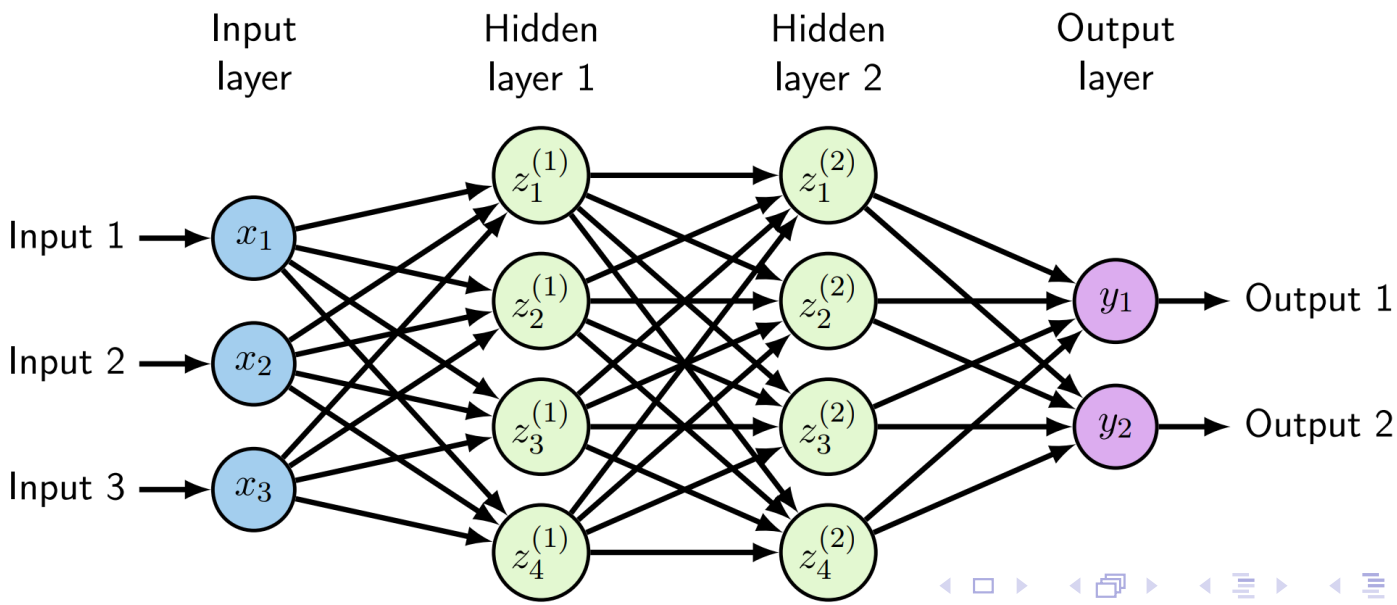
(i) des entiers non nuls $m_0, \ldots, m_M$, avec $m_0 = p$ et $m_M = K$, correspondant au nombre de neurones par couche ;

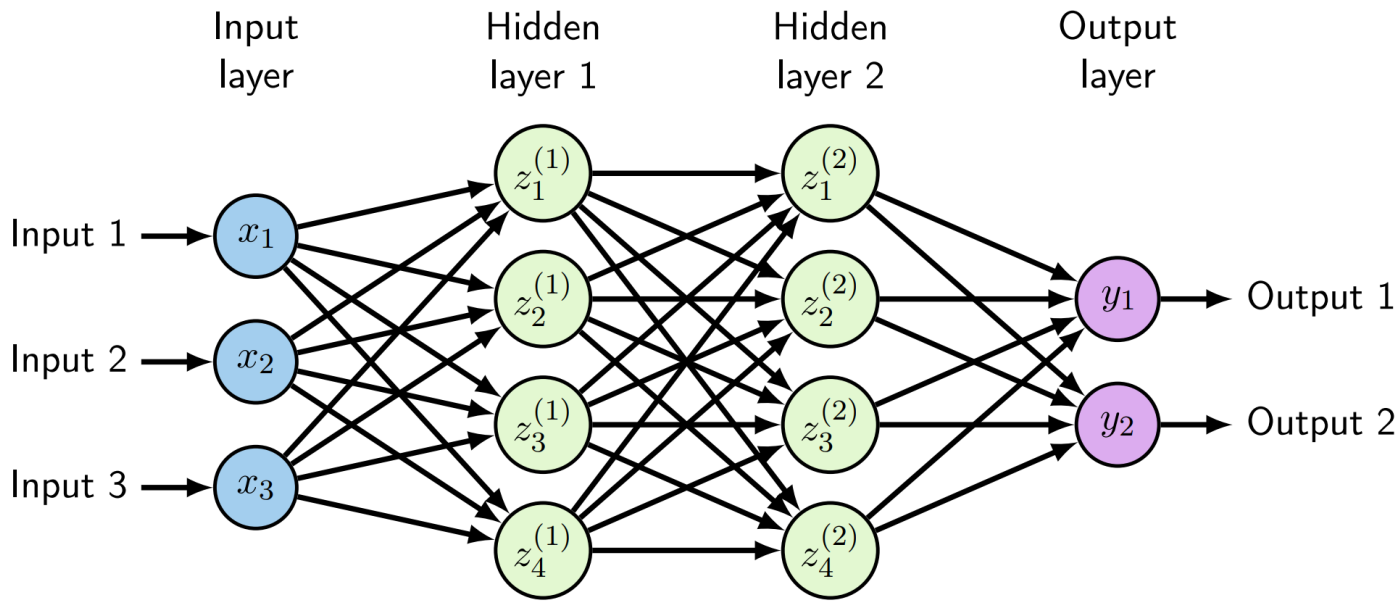
(ii) des transformations affines $g_1, \ldots, g_M$, avec $g_k : \mathbb{R}^{m_{k-1}} \to \mathbb{R}^{m_k}$ et $g_k(z) = W_k z + B_k$ où $W_k \in \mathcal{M}_{m_k, m_{k-1}}(\mathbb{R})$ et $B_k \in \mathbb{R}^{m_k}$;

(iii) des transformations non-linéaires appelées fonctions d'activation $\sigma_1, \ldots, \sigma_M$, avec $\sigma_k : \mathbb{R} \to \mathbb{R}$ et $h_k : \mathbb{R}^{m_k} \to \mathbb{R}^{m_k}$,

$$h_k(z_1, \ldots, z_{m_k}) = \begin{pmatrix} \sigma_k(z_1) \\ \vdots \\ \sigma_k(z_{m_k}) \end{pmatrix}$$

et telle que $\forall x \in \mathcal{X}$: $\psi(x) = h_M \circ g_M \circ \ldots \circ h_1 \circ g_1(x)$.





$$z_l^{(1)} = \sigma_1 \left(\sum_{i=1}^{3} [W_1]_{li} x_i + [B_1]_l \right), \text{ pour } l = 1, \ldots, 4$$

$$z_l^{(2)} = \sigma_2 \left(\sum_{i=1}^{4} [W_2]_{li} z_i^{(1)} + [B_2]_l \right), \text{ pour } l = 1, \ldots, 4$$

$$y_l = \sigma_3 \left(\sum_{i=1}^{4} [W_3]_{li} z_i^{(2)} + [B_3]_l \right), \text{ pour } l = 1, 2$$

$\Longrightarrow$ Pour la régression : les sorties du réseau sont directement les variables à prédire.

MLP pour la Classification à K classes :

$$\psi(x) = \text{softmax} \circ h_M \circ g_M \circ \ldots \circ h_1 \circ g_1(x) \ ,$$

$\Longrightarrow$ on rajoute une transformation softmax telle $\forall k = 1, \ldots, K$, $[\psi(x)]_k = \mathbb{P}(Y = k | X = x)$.

Définition La fonction softmax définie de $\mathbb{R}^K \to \mathbb{R}^K$ est telle que :

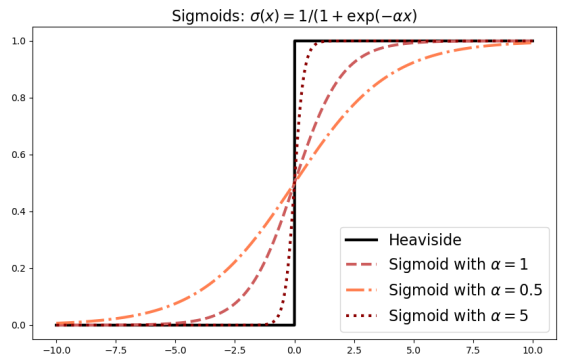
$$\forall z \in \mathbb{R}^K, \text{softmax}(z) = \left(\frac{\exp(z_1)}{\sum_{k=1}^K \exp(z_k)}, \ldots, \frac{\exp(z_K)}{\sum_{k=1}^K \exp(z_k)} \right) \ ,$$

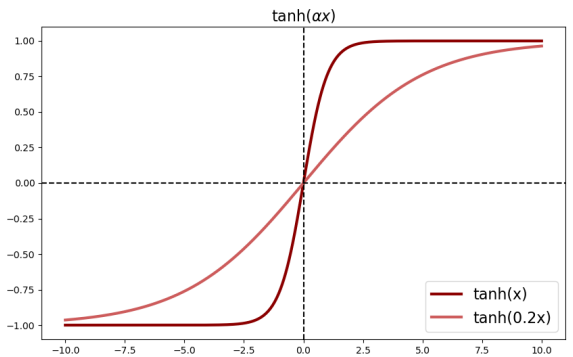
où $(z_1, \ldots, z_K)$ représentent les K composantes de z.

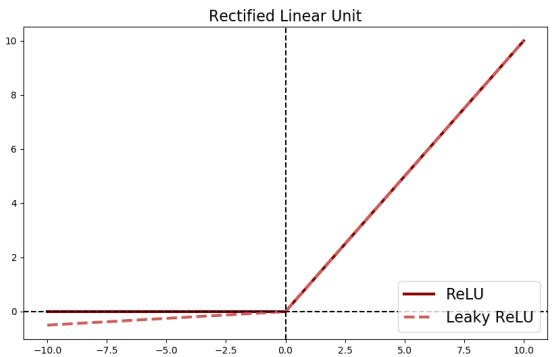
$\Longrightarrow$ Les composantes de softmax(z) somment à 1.

NB : Fonction de décision généralement associée : $\delta(x) = \arg \max_{1 \leq k \leq K} [\psi(x)]_k$.

Fonctions d'activation :







Hyperparamètres et paramètres

- Un vecteur d'hyperparamètres définit une famille de modèles paramètriques $\mathcal{M}_\Theta = \{p_\theta, \theta \in \Theta\}$
- Un vecteur de paramètres caractérise le modèle dans une famille de modèles paramétriques.

$\Longrightarrow$ Les paramètres sont déterminés par inférence statistique ou optimisation : on cherche $\hat{\theta} = \arg\min_{\theta\in\Theta} J\left(\theta; (x_i, y_i)_{1\leq i\leq N}\right)$.

$\Longrightarrow$ Les hyperparamètres sont choisis par sélection de modèles (*eg.* AIC ou validation croisée)

Rappel Définition : Un perceptron à $M+1$ couches est une fonction $\psi : \mathcal{X} \subset \mathbb{R}^p \to \mathcal{Y} \subset \mathbb{R}^K$:

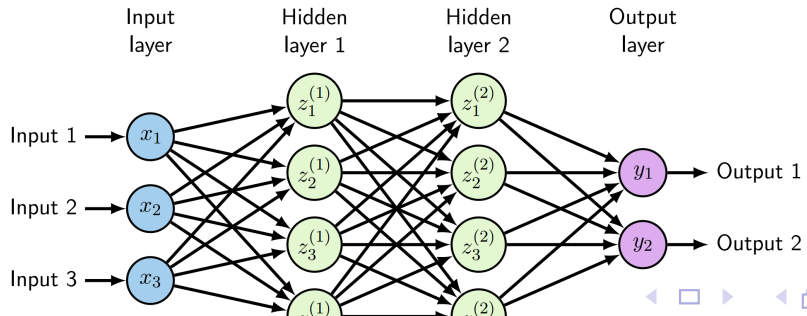
(i) des entiers $m_0, \ldots, m_M$, avec $m_0 = p$ et $m_M = K$: nombre de neurones par couche ;

(ii) des transformations affines $g_1, \ldots, g_M$, avec $g_k : \mathbb{R}^{m_{k-1}} \to \mathbb{R}^{m_k}$ et $g_k(z) = W_k z + B_k$ où $W_k \in \mathcal{M}_{m_k, m_{k-1}}(\mathbb{R})$ et $B_k \in \mathbb{R}^{m_k}$;

(iii) des fonctions d'activation $\sigma_1, \ldots, \sigma_M$, avec $\sigma_k : \mathbb{R} \to \mathbb{R}$ et $h_k : \mathbb{R}^{m_k} \to \mathbb{R}^{m_k}$,

$$h_k(z_1, \ldots, z_{m_k}) = \begin{pmatrix} \sigma_k(z_1) \\ \vdots \\ \sigma_k(z_{m_k}) \end{pmatrix}$$

et telle que $\forall x \in \mathcal{X}$: $\psi(x) = h_M \circ g_M \circ \ldots \circ h_1 \circ g_1(x)$.



Apprentissage : Soit $\mathcal{T} = (x_i, y_i)_{1 \leq i \leq N}$, données d'apprentissage indépendantes.
Pour un jeu d'hyperparamètres donné $\Longrightarrow$ estimation des $\theta = (W_k, B_k)_{1,\dots,M}$ par minimisation :

des moindres carrés $\quad J(\theta; \mathcal{T}) = \dfrac{1}{N} \sum_{i=1}^{N} \|\psi(x_i) - y_i\|^2 \quad$ pour la régression

de l'entropie croisée $\quad J(\theta; \mathcal{T}) = -\dfrac{1}{N} \sum_{i=1}^{N} \sum_{i=1}^{K} y_i \ln(\psi(x_i)) \quad$ pour la classification

Le problème du surapprentissage (overfitting) : Très grand nombre de paramètres : pour une couche à p neurones vers une couche à q neurones, on a $(p+1) \times q$ paramètres.

$\Longrightarrow$ Nécessité de régularisation :

- Introduction de pénalisation $\lambda\Omega(\theta)$: $\Omega(\theta) = \|\theta\|_1$ pour lasso ou $\Omega(\theta) = \|\theta\|_2^2$ pour ridge

Apprentissage : Soit $\mathcal{T} = (x_i, y_i)_{1 \leq i \leq N}$, données d'apprentissage indépendantes.
Pour un jeu d'hyperparamètres donné $\Longrightarrow$ estimation des $\theta = (W_k, B_k)_{1,\dots,M}$ par minimisation :

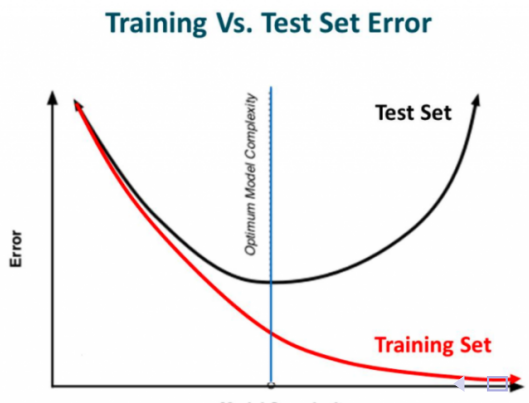
des moindres carrés $\quad J(\theta; \mathcal{T}) = \frac{1}{N} \sum_{i=1}^{N} \|\psi(x_i) - y_i\|^2 + \lambda\Omega(\theta)$ $\quad$ pour la régression

de l'entropie croisée $\quad J(\theta; \mathcal{T}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{i=1}^{K} y_i \ln(\psi(x_i)) + \lambda\Omega(\theta)$ $\quad$ pour la classification

Le problème du surapprentissage (overfitting) Très grand nombre de paramètres : pour une couche à p neurones vers une couche à q neurones, on a $(p+1) \times q$ paramètres.

$\Longrightarrow$ Nécessité de régularisation :

- Introduction de pénalisation $\lambda\Omega(\theta)$: $\Omega(\theta) = \|\theta\|_1$ pour lasso ou $\Omega(\theta) = \|\theta\|_2^2$ pour ridge
- Early stopping : on partage l'ensemble $\mathcal{T}$ en $\mathcal{T}_A$ pour l'apprentissage et $\mathcal{T}_T$ pour le test : on calcule l'erreur sur le test à chaque itération de l'algorithme, on stoppe quand elle ne diminue plus.



Apprentissage : résolution du problème de minimisation

- **Gradient stochastique :**

  Algorithme classique du gradient : $\theta_{n+1} = \theta_n - \rho_n \nabla J(\theta_n)$

  Difficulté : $J(\theta) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \psi(x_i))$ $\Longrightarrow$ souvent grand nombre de gradients à calculer

  Idée : $\mathbb{E}\left(L(Y, \psi(X)\right) \approx \frac{1}{N} \sum_{i=1}^{N} L(y_i, \psi(x_i))$

  $\Longrightarrow$ on peut prendre une approximation empirique de taille plus réduite :

  $$\mathbb{E}\left(L(Y, \psi(X)\right) \approx \frac{1}{Q} \sum_{j=1}^{Q} L(y_{i_j}, \psi(x_{i_j}))$$

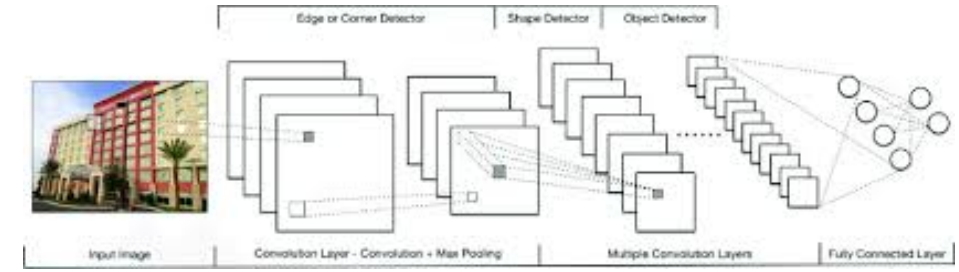
  voire même $\mathbb{E}\left(L(Y, \psi(X)\right) \approx L(y_j, \psi(x_j))$ pour j choisi aléatoirement

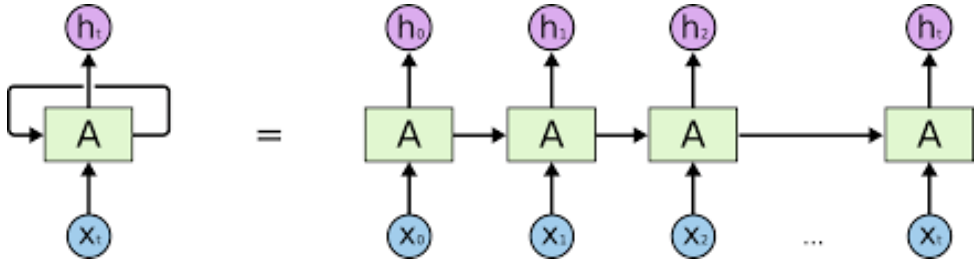
  $\Longrightarrow$ Méthode du gradient stochastique.

- **Rétropropagation du gradient :** utilisaton de la composition des fonctions et du calcul matriciel pour calculer de façon efficace $\frac{\partial L(y_i, \psi(x_i))}{\partial \theta_k}(\theta)$.

Conclusion sur les réseaux de neurones

- Modèle d'une grande richesse : capacité d'approximation de fonction de réponse complexe.
- Des architectures variées





- Des algorithmes d'apprentissage efficaces utilisant les nouveaux moyens de calcul (GPU)
- Capacité à tirer profit des larges bases de données
- Apprentissage de représentations cachées, notamment grâce aux couches profondes.
- Mais... apprentissage pas si simple, vraiment performant pour des larges bases de données d'apprentissage