

Super Pense-bête VIP : Machine Learning

Afshine AMIDI et Shervine AMIDI

6 octobre 2018

Table des matières

1 Apprentissage supervisé	2
1.1 Introduction à l'apprentissage supervisé	2
1.2 Notations et concepts généraux	2
1.2.1 Régression linéaire	3
1.2.2 Classification et régression logistique	3
1.2.3 Modèles linéaires généralisés	3
1.3 Support Vector Machines	3
1.4 Apprentissage génératif	4
1.4.1 Gaussian Discriminant Analysis	4
1.4.2 Naive Bayes	4
1.5 Méthode à base d'arbres et d'ensembles	4
1.6 Autres approches non-paramétriques	5
1.7 Théorie d'apprentissage	5
2 Apprentissage non-supervisé	6
2.1 Introduction à l'apprentissage non-supervisé	6
2.2 Partitionnement	6
2.2.1 Espérance-Maximisation	6
2.2.2 Partitionnement k -means	6
2.2.3 Regroupement hiérarchique	6
2.2.4 Indicateurs d'évaluation de clustering	6
2.3 Réduction de dimension	7
2.3.1 Analyse des composantes principales	7
2.3.2 Analyse en composantes indépendantes	7
3 Apprentissage profond	8
3.1 Réseau de neurones	8
3.2 Réseaux de neurones convolutionnels	8
3.3 Réseaux de neurones récurrents	8
3.4 Renforcement Learning	9

4 Astuces de Machine Learning	10
4.1 Indicateurs dans le contexte de la classification	10
4.2 Indicateurs dans le contexte de la régression	10
4.3 Sélection de modèle	10
4.4 Diagnostics	11
5 Rappels	12
5.1 Probabilités et Statistiques	12
5.1.1 Introduction aux probabilités à l'analyse combinatoire	12
5.1.2 Probabilité conditionnelle	12
5.1.3 Variable aléatoires	13
5.1.4 Variables aléatoires conjointement distribuées	13
5.1.5 Estimation des paramètres	14
5.2 Algèbre linéaire et Analyse	14
5.2.1 Notations générales	14
5.2.2 Opérations matricielles	15
5.2.3 Propriétés matricielles	15
5.2.4 Analyse matricielle	16

1 Apprentissage supervisé

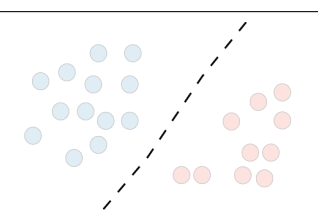
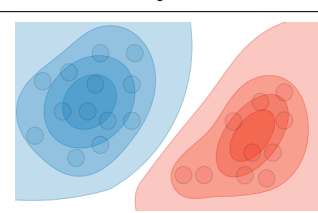
1.1 Introduction à l'apprentissage supervisé

Étant donné un ensemble de points $\{x^{(1)}, \dots, x^{(m)}\}$ associés à un ensemble d'issues $\{y^{(1)}, \dots, y^{(m)}\}$, on veut construire un classifieur qui apprend à prédire y depuis x .

□ **Type de prédiction** – Les différents types de modèle prédictifs sont résumés dans le tableau ci-dessous :

	Régression	Classifieur
Issue	Continu	Classe
Exemples	Régression linéaire	Régression logistique, SVM, Naive Bayes

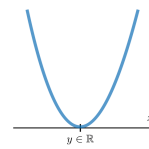
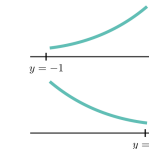
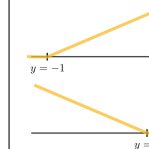
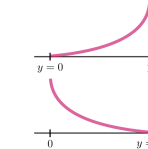
□ **Type de modèle** – Les différents modèles sont présentés dans le tableau ci-dessous :

	Modèle discriminatif	Modèle génératif
But	Estimer directement $P(y x)$	Estimer $P(x y)$ puis déduire $P(y x)$
Ce qui est appris	Frontière de décision	Distribution de proba des données
Illustration		
Exemples	Régressions, SVMs	GDA, Naive Bayes

1.2 Notations et concepts généraux

□ **Hypothèse** – Une hypothèse est notée h_θ et est le modèle que l'on choisit. Pour une entrée donnée $x^{(i)}$, la prédiction donnée par le modèle est $h_\theta(x^{(i)})$.

□ **Fonction de loss** – Une fonction de loss est une fonction $L : (z, y) \in \mathbb{R} \times Y \mapsto L(z, y) \in \mathbb{R}$ prenant comme entrée une valeur prédite z correspondant à une valeur réelle y , et nous renseigne sur la ressemblance de ces deux valeurs. Les fonctions de loss courantes sont récapitulées dans le tableau ci-dessous :

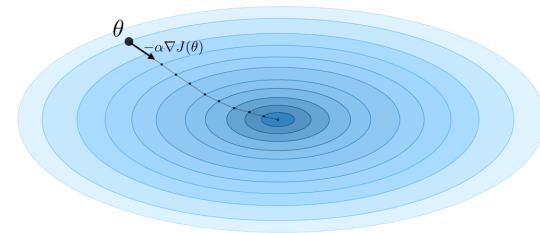
Moindres carrés	Logistique	Hinge loss	Cross-entropie
$\frac{1}{2}(y - z)^2$	$\log(1 + \exp(-yz))$	$\max(0, 1 - yz)$	$-[y \log(z) + (1 - y) \log(1 - z)]$
			
Régression linéaire	Régression logistique	SVM	Réseau de neurones

□ **Fonction de coût** – La fonction de coût J est communément utilisée pour évaluer la performance d'un modèle, et est définie avec la fonction de loss L par :

$$J(\theta) = \sum_{i=1}^m L(h_\theta(x^{(i)}), y^{(i)})$$

□ **Algorithme du gradient** – En notant $\alpha \in \mathbb{R}$ le taux d'apprentissage (en anglais *learning rate*), la règle de mise à jour de l'algorithme est exprimée en fonction du taux d'apprentissage et de la fonction de cost J de la manière suivante :

$$\theta \leftarrow \theta - \alpha \nabla J(\theta)$$



Remarque : L'algorithme du gradient stochastique (en anglais SGD - Stochastic Gradient Descent) met à jour le paramètre à partir de chaque élément du jeu d'entraînement, tandis que l'algorithme du gradient de batch le fait sur chaque lot d'exemples.

□ **Vraisemblance** – La vraisemblance d'un modèle $L(\theta)$ de paramètre θ est utilisée pour trouver le paramètre optimal θ par le biais du maximum de vraisemblance. En pratique, on utilise la log vraisemblance $\ell(\theta) = \log(L(\theta))$ qui est plus facile à optimiser. On a :

$$\theta^{\text{opt}} = \arg \max_{\theta} L(\theta)$$

□ **Algorithme de Newton** – L'algorithme de Newton est une méthode numérique qui trouve θ tel que $\ell'(\theta) = 0$. La règle de mise à jour est :

$$\theta \leftarrow \theta - \frac{\ell'(\theta)}{\ell''(\theta)}$$

Remarque : la généralisation multidimensionnelle, aussi connue sous le nom de la méthode de Newton-Raphson, a la règle de mise à jour suivante :

$$\theta \leftarrow \theta - \left(\nabla_{\theta}^2 \ell(\theta)\right)^{-1} \nabla_{\theta} \ell(\theta)$$

1.2.1 Régression linéaire

On suppose ici que $y|x; \theta \sim \mathcal{N}(\mu, \sigma^2)$

□ **Équations normales** – En notant X la matrice de design, la valeur de θ qui minimise la fonction de cost a une solution de forme fermée tel que :

$$\theta = (X^T X)^{-1} X^T y$$

□ **Algorithme LMS** – En notant α le taux d'apprentissage, la règle de mise à jour d'algorithme des moindres carrés (LMS) pour un jeu de données d'entraînement de m points, aussi connu sous le nom de règle de Widrow-Hoff, est donné par :

$$\forall j, \quad \theta_j \leftarrow \theta_j + \alpha \sum_{i=1}^m [y^{(i)} - h_{\theta}(x^{(i)})] x_j^{(i)}$$

Remarque : la règle de mise à jour est un cas particulier de l'algorithme du gradient.

□ **LWR** – Locally Weighted Regression, souvent noté LWR, est une variante de la régression linéaire appliquant un coefficient à chaque exemple dans sa fonction de coût via $w^{(i)}(x)$, qui est défini avec un paramètre $\tau \in \mathbb{R}$ de la manière suivante :

$$w^{(i)}(x) = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)$$

1.2.2 Classification et régression logistique

□ **Sigmoïde** – La sigmoïde g , aussi connue sous le nom de fonction logistique, est définie par :

$$\forall z \in \mathbb{R}, \quad g(z) = \frac{1}{1 + e^{-z}} \in]0,1[$$

□ **Régression logistique** – On suppose ici que $y|x; \theta \sim \text{Bernoulli}(\phi)$. On a la forme suivante :

$$\phi = p(y = 1|x; \theta) = \frac{1}{1 + \exp(-\theta^T x)} = g(\theta^T x)$$

Remarque : il n'y a pas de solution fermée dans le cas de la régression logistique.

□ **Régression softmax** – Une régression softmax, aussi appelée un régression logistique multi-classe, est utilisée pour généraliser la régression logistique lorsqu'il y a plus de 2 classes à prédire. Par convention, on fixe $\theta_K = 0$, ce qui oblige le paramètre de Bernoulli ϕ_i de chaque classe i à être égal à :

$$\phi_i = \frac{\exp(\theta_i^T x)}{\sum_{j=1}^K \exp(\theta_j^T x)}$$

1.2.3 Modèles linéaires généralisés

□ **Famille exponentielle** – Une classe de distributions est issue de la famille exponentielle lorsqu'elle peut être écrite en termes d'un paramètre naturel, aussi appelé paramètre canonique ou fonction de lien η , d'une statistique suffisante $T(y)$ et d'une fonction de log-partition $a(\eta)$ de la manière suivante :

$$p(y; \eta) = b(y) \exp(\eta T(y) - a(\eta))$$

Remarque : on aura souvent $T(y) = y$. Aussi, $\exp(-a(\eta))$ peut être vu comme un paramètre de normalisation s'assurant que les probabilités somment à un.

Les distributions exponentielles les plus communément rencontrées sont récapitulées dans le tableau ci-dessous :

Distribution	η	$T(y)$	$a(\eta)$	$b(y)$
Bernoulli	$\log\left(\frac{\phi}{1-\phi}\right)$	y	$\log(1 + \exp(\eta))$	1
Gaussian	μ	y	$\frac{\eta^2}{2}$	$\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{y^2}{2}\right)$
Poisson	$\log(\lambda)$	y	e^{η}	$\frac{1}{y!}$
Geometric	$\log(1 - \phi)$	y	$\log\left(\frac{e^{\eta}}{1 - e^{\eta}}\right)$	1

□ **Hypothèses pour les GLMs** – Les modèles linéaires généralisés (GLM) ont pour but de prédire une variable aléatoire y comme une fonction de $x \in \mathbb{R}^{n+1}$ et reposent sur les 3 hypothèses suivantes :

$$(1) \quad y|x; \theta \sim \text{ExpFamily}(\eta) \quad (2) \quad h_{\theta}(x) = E[y|x; \theta] \quad (3) \quad \eta = \theta^T x$$

Remarque : la méthode des moindres carrés ordinaires et la régression logistique sont des cas spéciaux des modèles linéaires généralisés.

1.3 Support Vector Machines

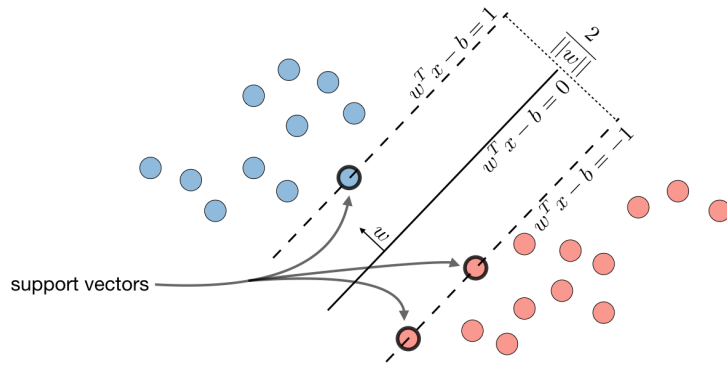
Le but des support vector machines est de trouver la ligne qui maximise la distance minimum à la ligne.

□ **Classifieur à marges optimales** – Le classifieur à marges optimales h est tel que :

$$h(x) = \text{sign}(w^T x - b)$$

où $(w, b) \in \mathbb{R}^n \times \mathbb{R}$ est une solution du problème d'optimisation suivant :

$$\min \frac{1}{2} \|w\|^2 \quad \text{tel que} \quad y^{(i)} (w^T x^{(i)} - b) \geq 1$$



Remarque : la ligne est définie par $w^T x - b = 0$.

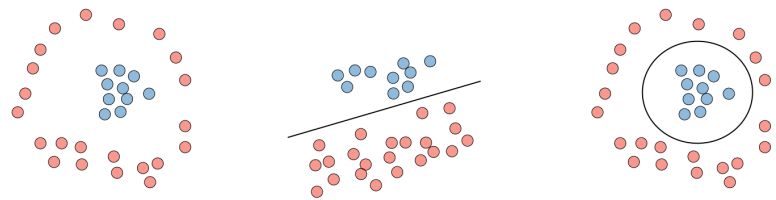
□ **Hinge loss** – Le hinge loss est utilisé dans le cadre des SVMs et est défini de la manière suivante :

$$L(z, y) = [1 - yz]_+ = \max(0, 1 - yz)$$

□ **Noyau** – Étant donné un feature mapping ϕ , on définit le noyau K par :

$$K(x, z) = \phi(x)^T \phi(z)$$

En pratique, le noyau K défini par $K(x, z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$ est nommé noyau gaussien et est communément utilisé.



Séparation non linéaire → Mapping de noyaux ϕ → Ligne de décision dans l'espace initial

Remarque : on dit que l'on utilise "l'astuce du noyau" (en anglais *kernel trick*) pour calculer la fonction de coût en utilisant le noyau parce qu'il se trouve que l'on n'a pas besoin de trouver le mapping explicite, qui est souvent compliqué. Il suffit de connaître les valeurs de $K(x, z)$.

□ **Lagrangien** – On définit le lagrangien $L(w, b)$ par :

$$\mathcal{L}(w, b) = f(w) + \sum_{i=1}^l \beta_i h_i(w)$$

Remarque : les coefficients β_i sont appelés les multiplicateurs de Lagrange.

1.4 Apprentissage génératif

Un modèle génératif essaie d'abord d'apprendre comment les données sont générées en estimant $P(x|y)$, nous permettant ensuite d'estimer $P(y|x)$ par le biais du théorème de Bayes.

1.4.1 Gaussian Discriminant Analysis

□ **Cadre** – Le Gaussian Discriminant Analysis suppose que y et $x|y = 0$ et $x|y = 1$ sont tels que :

$$y \sim \text{Bernoulli}(\phi)$$

$$x|y = 0 \sim \mathcal{N}(\mu_0, \Sigma) \quad \text{et} \quad x|y = 1 \sim \mathcal{N}(\mu_1, \Sigma)$$

□ **Estimation** – Le tableau suivant récapitule les estimations que l'on a trouvées lors de la maximisation de la vraisemblance :

$\hat{\phi}$	$\hat{\mu}_j \quad (j = 0, 1)$	$\hat{\Sigma}$
$\frac{1}{m} \sum_{i=1}^m 1_{\{y^{(i)}=1\}}$	$\frac{\sum_{i=1}^m 1_{\{y^{(i)}=j\}} x^{(i)}}{\sum_{i=1}^m 1_{\{y^{(i)}=j\}}}$	$\frac{1}{m} \sum_{i=1}^m (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T$

1.4.2 Naive Bayes

□ **Hypothèse** – Le modèle de Naive Bayes suppose que les caractéristiques de chaque point sont toutes indépendantes :

$$P(x|y) = P(x_1, x_2, \dots | y) = P(x_1|y)P(x_2|y) \dots = \prod_{i=1}^n P(x_i|y)$$

□ **Solutions** – Maximiser la log vraisemblance donne les solutions suivantes, où $k \in \{0, 1\}, l \in [1, L]$

$$P(y = k) = \frac{1}{m} \times \#\{j|y^{(j)} = k\} \quad \text{et} \quad P(x_i = l|y = k) = \frac{\#\{j|y^{(j)} = k \text{ et } x_i^{(j)} = l\}}{\#\{j|y^{(j)} = k\}}$$

Remarque : Naive Bayes est couramment utilisé pour la classification de texte et pour la détection de spams.

1.5 Méthode à base d'arbres et d'ensembles

Ces méthodes peuvent être utilisées pour des problèmes de régression et de classification.

□ **CART** – Les arbres de classification et de régression (en anglais *CART - Classification And Regression Trees*), aussi connus sous le nom d'arbres de décision, peuvent être représentés sous la forme d'arbres binaires. Ils ont l'avantage d'être très interprétables.

□ **Random forest** – C’est une technique à base d’arbres qui utilise un très grand nombre d’arbres de décisions construits à partir d’ensembles de caractéristiques aléatoirement sélectionnés. Contrairement à un simple arbre de décision, il n’est pas interprétable du tout mais le fait qu’il ait une bonne performance en fait un algorithme populaire.

Remarque : les random forests sont un type de méthode ensembliste.

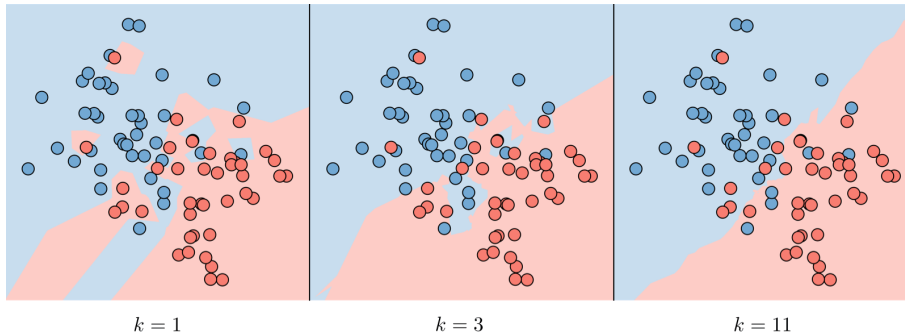
□ **Boosting** – L’idée des méthodes de boosting est de combiner plusieurs modèles faibles pour former un modèle meilleur. Les principales méthodes de boosting sont récapitulées dans le tableau ci-dessous :

Boosting adaptatif	Boosting par gradient
- De grands coefficients sont mis sur les erreurs pour s’améliorer à la prochaine étape de boosting - Connus sous le nom d’Adaboost	- Les modèles faibles sont entraînés sur les erreurs résiduelles

1.6 Autres approches non-paramétriques

□ **k-nearest neighbors** – L’algorithme des k plus proches voisins (en anglais *k-nearest neighbors*), aussi connu sous le nom de k-NN, est une approche non-paramétrique où la réponse d’un point est déterminée par la nature de ses k voisins du jeu de données d’entraînement. Il peut être utilisé dans des cadres de classification et de régression.

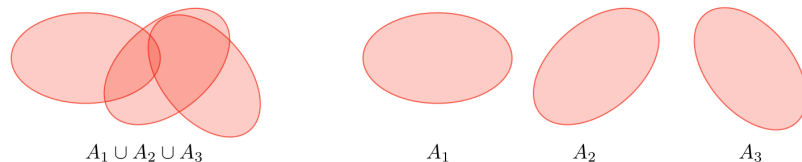
Remarque : Plus le paramètre k est élevé, plus le biais est élevé, et plus le paramètre k est faible, plus la variance est élevée.



1.7 Théorie d’apprentissage

□ **Inégalité de Boole** – Soit A_1, \dots, A_k k événements. On a :

$$P(A_1 \cup \dots \cup A_k) \leq P(A_1) + \dots + P(A_k)$$



□ **Inégalité d’Hoeffding** – Soit Z_1, \dots, Z_m m variables iid tirées d’une distribution de Bernoulli de paramètre ϕ . Soit $\hat{\phi}$ leur moyenne empirique et $\gamma > 0$ fixé. On a :

$$P(|\phi - \hat{\phi}| > \gamma) \leq 2 \exp(-2\gamma^2 m)$$

Remarque : cette inégalité est aussi connue sous le nom de borne de Chernoff.

□ **Erreur de training** – Pour un classifieur donné h, on définit l’erreur d’entraînement $\hat{\epsilon}(h)$, aussi connu sous le nom de risque empirique ou d’erreur empirique, par :

$$\hat{\epsilon}(h) = \frac{1}{m} \sum_{i=1}^m 1_{\{h(x^{(i)}) \neq y^{(i)}\}}$$

□ **Probablement Approximativement Correct (PAC)** – PAC est un cadre dans lequel de nombreux résultats d’apprentissages ont été prouvés, et contient l’ensemble d’hypothèses suivant :

- les jeux d’entraînement et de test suivent la même distribution
- les exemples du jeu d’entraînement sont tirés indépendamment

□ **Éclatement** – Étant donné un ensemble $S = \{x^{(1)}, \dots, x^{(d)}\}$, et un ensemble de classifieurs \mathcal{H} , on dit que \mathcal{H} brise S si pour tout ensemble de labels $\{y^{(1)}, \dots, y^{(d)}\}$, on a :

$$\exists h \in \mathcal{H}, \quad \forall i \in \llbracket 1, d \rrbracket, \quad h(x^{(i)}) = y^{(i)}$$

□ **Théorème de la borne supérieure** – Soit \mathcal{H} une hypothèse finie de classe telle que $|\mathcal{H}| = k$, soit δ , et soit m la taille fixée d’un échantillon. Alors, avec une probabilité d’au moins $1 - \delta$, on a :

$$\epsilon(\hat{h}) \leq \left(\min_{h \in \mathcal{H}} \epsilon(h) \right) + 2 \sqrt{\frac{1}{2m} \log \left(\frac{2k}{\delta} \right)}$$

□ **Dimension VC** – La dimension de Vapnik-Chervonenkis (VC) d’une classe d’hypothèses de classes infinies donnée \mathcal{H} , que l’on note $VC(\mathcal{H})$, est la taille de l’ensemble le plus grand qui est brisé par \mathcal{H} .

Remarque : la dimension VC de $\mathcal{H} = \{\text{set of linear classifiers in 2 dimensions}\}$ est égale à 3.



□ **Théorème (Vapnik)** – Soit \mathcal{H} donné, avec $VC(\mathcal{H}) = d$ avec m le nombre d’exemples d’entraînement. Avec une probabilité d’au moins $1 - \delta$, on a :

$$\epsilon(\hat{h}) \leq \left(\min_{h \in \mathcal{H}} \epsilon(h) \right) + O \left(\sqrt{\frac{d}{m} \log \left(\frac{m}{d} \right)} + \frac{1}{m} \log \left(\frac{1}{\delta} \right) \right)$$

2 Apprentissage non-supervisé

2.1 Introduction à l'apprentissage non-supervisé

□ **Motivation** – Le but de l'apprentissage non-supervisé est de trouver des formes cachées dans un jeu de données non-labelées $\{x^{(1)}, \dots, x^{(m)}\}$.

□ **Inégalité de Jensen** – Soit f une fonction convexe et X une variable aléatoire. On a l'inégalité suivante :

$$E[f(X)] \geq f(E[X])$$

2.2 Partitionnement

2.2.1 Espérance-Maximisation

□ **Variables latentes** – Les variables latentes sont des variables cachées/non-observées qui posent des difficultés aux problèmes d'estimation, et sont souvent notées z . Voici les cadres dans lesquelles les variables latentes sont le plus fréquemment utilisées :

Cadre	Variance latente z	$x z$	Commentaires
Mixture de k gaussiennes	Multinomial(ϕ)	$\mathcal{N}(\mu_j, \Sigma_j)$	$\mu_j \in \mathbb{R}^n, \phi \in \mathbb{R}^k$
Analyse factorielle	$\mathcal{N}(0, I)$	$\mathcal{N}(\mu + \Lambda z, \psi)$	$\mu_j \in \mathbb{R}^n$

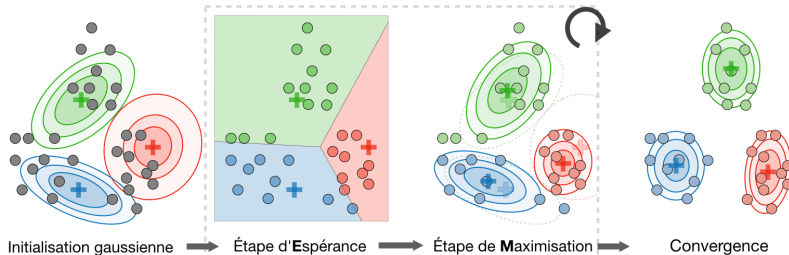
□ **Algorithme** – L'algorithme d'espérance-maximisation (EM) est une méthode efficace pour estimer le paramètre θ . Elle passe par le maximum de vraisemblance en construisant un borne inférieure sur la vraisemblance (E-step) et optimisant cette borne inférieure (M-step) de manière successive :

– **E-step** : Évaluer la probabilité postérieure $Q_i(z^{(i)})$ que chaque point $x^{(i)}$ provienne d'une partition particulière $z^{(i)}$ de la manière suivante :

$$Q_i(z^{(i)}) = P(z^{(i)} | x^{(i)}; \theta)$$

– **M-step** : Utiliser les probabilités postérieures $Q_i(z^{(i)})$ en tant que coefficients propres aux partitions sur les points $x^{(i)}$ pour ré-estimer séparément chaque modèle de partition de la manière suivante :

$$\theta_i = \operatorname{argmax}_{\theta} \sum_i \int_{z^{(i)}} Q_i(z^{(i)}) \log \left(\frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})} \right) dz^{(i)}$$

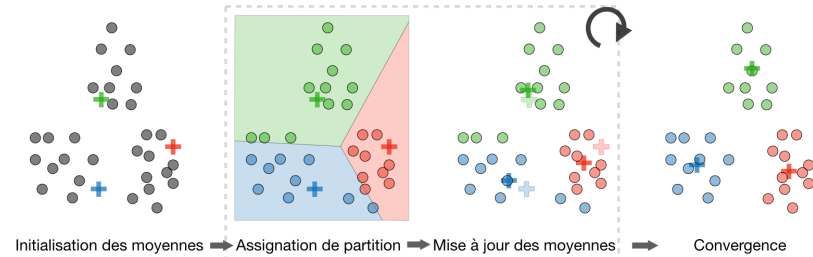


2.2.2 Partitionnement k -means

On note $c^{(i)}$ la partition du point i et μ_j le centre de la partition j .

□ **Algorithme** – Après avoir aléatoirement initialisé les centroïdes de partitions $\mu_1, \mu_2, \dots, \mu_k \in \mathbb{R}^n$, l'algorithme k -means répète l'étape suivante jusqu'à convergence :

$$c^{(i)} = \operatorname{arg min}_j \|x^{(i)} - \mu_j\|^2 \quad \text{et} \quad \mu_j = \frac{\sum_{i=1}^m 1_{\{c^{(i)}=j\}} x^{(i)}}{\sum_{i=1}^m 1_{\{c^{(i)}=j\}}}$$



□ **Fonction de distorsion** – Pour voir si l'algorithme converge, on regarde la fonction de distorsion définie de la manière suivante :

$$J(c, \mu) = \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

2.2.3 Regroupement hiérarchique

□ **Algorithme** – C'est un algorithme de partitionnement avec une approche hiérarchique qui construit des partitions intriquées de manière successive.

□ **Types** – Il y a différents types d'algorithme de regroupement hiérarchique qui ont pour but d'optimiser différents fonctions objectif, récapitulés dans le tableau ci-dessous :

Ward linkage	Average linkage	Complete linkage
Minimize within cluster distance	Minimize average distance between cluster pairs	Minimize maximum distance of between cluster pairs

2.2.4 Indicateurs d'évaluation de clustering

Dans le cadre de l'apprentissage non-supervisé, il est souvent difficile d'évaluer la performance d'un modèle vu que les vrais labels ne sont pas connus (contrairement à l'apprentissage supervisé).

□ **Coefficient silhouette** – En notant a et b la distance moyenne entre un échantillon et tous les autres points d'une même classe, et entre un échantillon et tous les autres points de la prochaine partition la plus proche, le coefficient silhouette s d'un échantillon donné est défini de la manière suivante :

$$s = \frac{b - a}{\max(a, b)}$$

□ **Index de Calinski-Harabaz** – En notant k le nombre de partitions, B_k et W_k les matrices de dispersion entre-partitions et au sein d’une même partition sont définis respectivement par :

$$B_k = \sum_{j=1}^k n_{c^{(j)}} (\mu_{c^{(j)}} - \mu)(\mu_{c^{(j)}} - \mu)^T, \quad W_k = \sum_{i=1}^m (x^{(i)} - \mu_{c^{(i)}})(x^{(i)} - \mu_{c^{(i)}})^T$$

l’index de Calinski-Harabaz $s(k)$ renseigne sur la qualité des partitions, de sorte à ce qu’un score plus élevé indique des partitions plus denses et mieux séparées entre elles. Il est défini par :

$$s(k) = \frac{\text{Tr}(B_k)}{\text{Tr}(W_k)} \times \frac{N - k}{k - 1}$$

2.3 Réduction de dimension

2.3.1 Analyse des composantes principales

C’est une technique de réduction de dimension qui trouve les directions maximisant la variance, vers lesquelles les données sont projetées.

□ **Valeur propre, vecteur propre** – Soit une matrice $A \in \mathbb{R}^{n \times n}$, λ est dit être une valeur propre de A s’il existe un vecteur $z \in \mathbb{R}^n \setminus \{0\}$, appelé vecteur propre, tel que l’on a :

$$Az = \lambda z$$

□ **Théorème spectral** – Soit $A \in \mathbb{R}^{n \times n}$. Si A est symétrique, alors A est diagonalisable par une matrice réelle orthogonale $U \in \mathbb{R}^{n \times n}$. En notant $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$, on a :

$$\exists \Lambda \text{ diagonal, } A = U\Lambda U^T$$

Remarque : le vecteur propre associé à la plus grande valeur propre est appelé le vecteur propre principal de la matrice A .

□ **Algorithme** – La procédure d’analyse des composantes principales (en anglais *PCA - Principal Component Analysis*) est une technique de réduction de dimension qui projette les données sur k dimensions en maximisant la variance des données de la manière suivante :

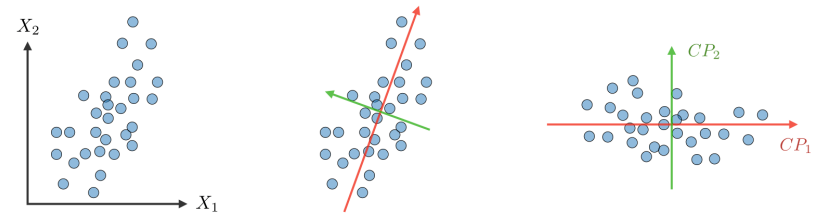
– Étape 1 : Normaliser les données pour avoir une moyenne de 0 et un écart-type de 1.

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{\sigma_j} \quad \text{où} \quad \mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)} \quad \text{et} \quad \sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2$$

– Étape 2 : Calculer $\Sigma = \frac{1}{m} \sum_{i=1}^m x^{(i)} x^{(i)T} \in \mathbb{R}^{n \times n}$, qui est symétrique et aux valeurs propres réelles.

– Étape 3 : Calculer $u_1, \dots, u_k \in \mathbb{R}^n$ les k valeurs propres principales orthogonales de Σ , i.e. les vecteurs propres orthogonaux des k valeurs propres les plus grandes.

– Étape 4 : Projeter les données sur $\text{span}_{\mathbb{R}}(u_1, \dots, u_k)$. Cette procédure maximise la variance sur tous les espaces à k dimensions.



Données dans l'espace initial → Trouve les composantes principales → Données dans l'espace des CP

2.3.2 Analyse en composantes indépendantes

C’est une technique qui vise à trouver les sources génératrices sous-jacentes.

□ **Hypothèses** – On suppose que nos données x ont été générées par un vecteur source à n dimensions $s = (s_1, \dots, s_n)$, où les s_i sont des variables aléatoires indépendantes, par le biais d’une matrice de mélange et inversible A de la manière suivante :

$$x = As$$

Le but est de trouver la matrice de démixage $W = A^{-1}$.

□ **Algorithme d’ICA de Bell and Sejnowski** – Cet algorithme trouve la matrice de démixage W en suivant les étapes ci-dessous :

– Écrire la probabilité de $x = As = W^{-1}s$ par :

$$p(x) = \prod_{i=1}^n p_s(w_i^T x) \cdot |W|$$

– Écrire la log vraisemblance de notre jeu de données d’entraînement $\{x^{(i)}, i \in \llbracket 1, m \rrbracket\}$ et en notant g la fonction sigmoïde par :

$$l(W) = \sum_{i=1}^m \left(\sum_{j=1}^n \log \left(g'(w_j^T x^{(i)}) \right) + \log |W| \right)$$

Par conséquent, l’algorithme du gradient stochastique est tel que pour chaque exemple du jeu d’entraînement $x^{(i)}$, on met à jour W de la manière suivante :

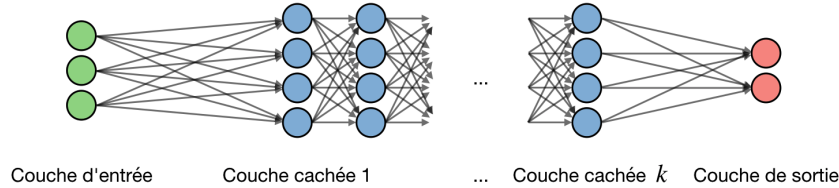
$$W \leftarrow W + \alpha \left(\begin{pmatrix} 1 - 2g(w_1^T x^{(i)}) \\ 1 - 2g(w_2^T x^{(i)}) \\ \vdots \\ 1 - 2g(w_n^T x^{(i)}) \end{pmatrix} x^{(i)T} + (W^T)^{-1} \right)$$

3 Apprentissage profond

3.1 Réseau de neurones

Les réseaux de neurones (en anglais *neural networks*) sont une classe de modèles qui sont construits à l'aide de couches de neurones. Les réseaux de neurones convolutionnels (en anglais *convolutional neural networks*) ainsi que les réseaux de neurones récurrents (en anglais *recurrent neural networks*) font parti des principaux types de réseaux de neurones.

□ **Architecture** – Le vocabulaire autour des architectures des réseaux de neurones est décrit dans la figure ci-dessous :



En notant i la $i^{\text{ème}}$ couche du réseau et j la $j^{\text{ième}}$ unité de couche cachée, on a :

$$z_j^{[i]} = w_j^{[i]T} x + b_j^{[i]}$$

où l'on note w, b, z le coefficient, le biais ainsi que la variable sortie respectivement.

□ **Fonction d'activation** – Les fonctions d'activation sont utilisées à la fin d'une unité de couche cachée pour introduire des complexités non linéaires au modèle. En voici les plus fréquentes :

Sigmoïde	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$

□ **Cross-entropy loss** – Dans le contexte des réseaux de neurones, la fonction objectif de cross-entropie $L(z, y)$ est communément utilisée et est définie de la manière suivante :

$$L(z, y) = - \left[y \log(z) + (1 - y) \log(1 - z) \right]$$

□ **Taux d'apprentissage** – Le taux d'apprentissage (appelé en anglais *learning rate*), souvent noté α ou parfois η , indique la vitesse à laquelle les coefficients évoluent. Cette quantité peut être fixe ou variable. L'une des méthodes les plus populaires à l'heure actuelle s'appelle Adam, qui a un taux d'apprentissage qui s'adapte au fil du temps.

□ **Rétropropagation du gradient** – La rétropropagation du gradient (en anglais *backpropagation*) est une méthode destinée à mettre à jour les coefficients d'un réseau de neurones en comparant la sortie obtenue et la sortie désirée. La dérivée par rapport au coefficient w est calculée à l'aide du théorème de dérivation des fonctions composées, et s'écrit de la manière suivante :

$$\frac{\partial L(z, y)}{\partial w} = \frac{\partial L(z, y)}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w}$$

Ainsi, le coefficient est actualisé de la manière suivante :

$$w \leftarrow w - \eta \frac{\partial L(z, y)}{\partial w}$$

□ **Actualiser les coefficients** – Dans un réseau de neurones, les coefficients sont actualisés comme suit :

- Étape 1 : Prendre un groupe d'observations appartenant aux données du training set.
- Étape 2 : Réaliser la propagation avant pour obtenir le loss correspondant.
- Étape 3 : Effectuer une rétropropagation du loss pour obtenir les gradients.
- Étape 4 : Utiliser les gradients pour actualiser les coefficients du réseau.

□ **Dropout** – Le dropout est une technique qui est destinée à empêcher le sur-ajustement sur les données de training en abandonnant des unités dans un réseau de neurones. En pratique, les neurones sont soit abandonnés avec une probabilité p ou gardés avec une probabilité $1 - p$.

3.2 Réseaux de neurones convolutionnels

□ **Pré-requis de la couche convolutionnelle** – Si l'on note W la taille du volume d'entrée, F la taille de la couche de neurones convolutionnelle, P la quantité de zero padding, alors le nombre de neurones N qui tient dans un volume donné est tel que :

$$N = \frac{W - F + 2P}{S} + 1$$

□ **Normalisation de batch** – C'est une étape possédant les paramètres γ, β qui normalise le batch $\{x_i\}$. En notant μ_B, σ_B^2 la moyenne et la variance de ce que l'on veut corriger au batch, ceci est fait de la manière suivante :

$$x_i \leftarrow \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$$

Cela est normalement effectué après une couche fully-connected/couche convolutionnelle et avant une couche de non-linéarité et a pour but de permettre un taux d'apprentissage plus grand et de réduire une dépendance trop forte à l'initialisation.

3.3 Réseaux de neurones récurrents

□ **Types de porte** – Voici les différents types de porte que l'on rencontre dans un réseau de neurones récurrent type :

Porte d'entrée	Porte d'oubli	Porte de sortie	Porte
Écrire ?	Supprimer ?	A quel point révéler ?	Combien écrire ?

□ **LSTM** – Un réseau de long court terme (en anglais *long short-term memory*, LSTM) est un type de modèle RNN qui empêche le phénomène de *vanishing gradient* en ajoutant des portes d'oubli.

3.4 Reinforcement Learning

Le but du reinforcement learning est pour un agent d'apprendre comment évoluer dans un environnement.

□ **Processus de décision markovien** – Un processus de décision markovien (MDP) est décrite par 5 quantités $(S, \mathcal{A}, \{P_{sa}\}, \gamma, R)$, où :

- S est l'ensemble des états
- \mathcal{A} est l'ensemble des actions
- $\{P_{sa}\}$ sont les probabilités d'états de transition pour $s \in S$ et $a \in \mathcal{A}$
- $\gamma \in [0, 1[$ est le taux d'actualisation (en anglais *discount factor*)
- $R : S \times \mathcal{A} \rightarrow \mathbb{R}$ ou $R : S \rightarrow \mathbb{R}$ est la fonction de récompense que l'algorithme veut maximiser

□ **Politique** – Une politique π est une fonction $\pi : S \rightarrow \mathcal{A}$ qui lie les états aux actions.

Remarque : on dit que l'on effectue une politique donnée π si étant donné un état s , on prend l'action $a = \pi(s)$.

□ **Fonction de valeurs** – Pour une politique donnée π et un état donné s , on définit la fonction de valeurs V^π comme suit :

$$V^\pi(s) = E \left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots | s_0 = s, \pi \right]$$

□ **Équation de Bellman** – Les équations de Bellman optimales caractérisent la fonction de valeurs V^{π^*} de la politique optimale π^* :

$$V^{\pi^*}(s) = R(s) + \max_{a \in \mathcal{A}} \gamma \sum_{s' \in S} P_{sa}(s') V^{\pi^*}(s')$$

Remarque : on note que la politique optimale π^ pour un état donné s est tel que :*

$$\pi^*(s) = \operatorname{argmax}_{a \in \mathcal{A}} \sum_{s' \in S} P_{sa}(s') V^*(s')$$

□ **Algorithme d'itération sur la valeur** – L'algorithme d'itération sur la valeur est faite de deux étapes :

- On initialise la valeur :

$$V_0(s) = 0$$

- On itère la valeur en se basant sur les valeurs précédentes :

$$V_{i+1}(s) = R(s) + \max_{a \in \mathcal{A}} \left[\sum_{s' \in S} \gamma P_{sa}(s') V_i(s') \right]$$

□ **Maximum de vraisemblance** – Les estimations du maximum de vraisemblance pour les transitions de probabilité d'état sont comme suit :

$$P_{sa}(s') = \frac{\text{\#fois où l'action } a \text{ dans l'état } s \text{ est prise pour arriver à l'état } s'}{\text{\#fois où l'action } a \text{ dans l'état } s \text{ est prise}}$$

□ **Q-learning** – Le Q-learning est une estimation non-paramétrique de Q, qui est faite de la manière suivante :

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[R(s, a, s') + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

4 Astuces de Machine Learning

4.1 Indicateurs dans le contexte de la classification

Dans le contexte de la classification binaire, voici les principaux indicateurs à surveiller pour évaluer la performance d'un modèle.

□ **Matrice de confusion** – Une matrice de confusion est utilisée pour avoir une image complète de la performance d'un modèle. Elle est définie de la manière suivante :

		Classe prédite	
		+	-
Classe vraie	+	TP True Positives	FN False Negatives Type II error
	-	FP False Positives Type I error	TN True Negatives

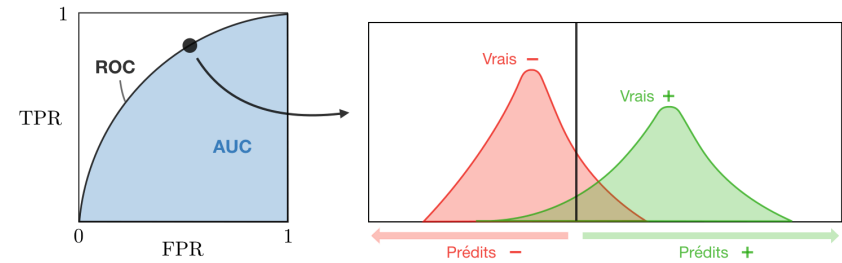
□ **Indicateurs principaux** – Les indicateurs suivants sont communément utilisés pour évaluer la performance des modèles de classification :

Indicateur	Formule	Interprétation
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Performance globale du modèle
Precision	$\frac{TP}{TP + FP}$	À quel point les prédictions positives sont précises
Recall Sensitivity	$\frac{TP}{TP + FN}$	Couverture des observations vraiment positives
Specificity	$\frac{TN}{TN + FP}$	Couverture des observations vraiment négatives
F1 score	$\frac{2TP}{2TP + FP + FN}$	Indicateur hybride pour les classes non-balancées

□ **Courbe ROC** – La fonction d'efficacité du récepteur, plus fréquemment appelée courbe ROC (de l'anglais *Receiver Operating Curve*), est une courbe représentant le taux de *True Positives* en fonction de taux de *False Positives* et obtenue en faisant varier le seuil. Ces indicateurs sont résumés dans le tableau suivant :

Indicateur	Formule	Equivalent
True Positive Rate TPR	$\frac{TP}{TP + FN}$	Recall, sensitivity
False Positive Rate FPR	$\frac{FP}{TN + FP}$	1-specificity

□ **AUC** – L'aire sous la courbe ROC, aussi notée AUC (de l'anglais *Area Under the Curve*) ou AUROC (de l'anglais *Area Under the ROC*), est l'aire sous la courbe ROC comme le montre la figure suivante :



4.2 Indicateurs dans le contexte de la régression

□ **Indicateurs de base** – Étant donné un modèle de régression f , les indicateurs suivants sont communément utilisés pour évaluer la performance d'un modèle :

Somme des carrés totale	Somme des carrés expliquée	Somme des carrés résiduelle
$SS_{tot} = \sum_{i=1}^m (y_i - \bar{y})^2$	$SS_{reg} = \sum_{i=1}^m (f(x_i) - \bar{y})^2$	$SS_{res} = \sum_{i=1}^m (y_i - f(x_i))^2$

□ **Coefficient de détermination** – Le coefficient de détermination, souvent noté R^2 ou r^2 , donne une mesure sur la qualité du modèle et est tel que :

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

□ **Indicateurs principaux** – Les indicateurs suivants sont communément utilisés pour évaluer la performance des modèles de régression, en prenant en compte le nombre de variables n qu'ils prennent en considération :

Cp de Mallow	AIC	BIC	R^2 ajusté
$\frac{SS_{res} + 2(n+1)\hat{\sigma}^2}{m}$	$2[(n+2) - \log(L)]$	$\log(m)(n+2) - 2\log(L)$	$1 - \frac{(1-R^2)(m-1)}{m-n-1}$

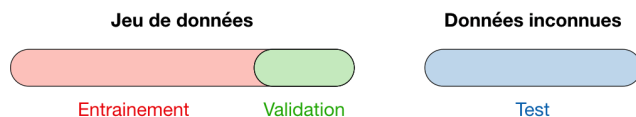
où L est la vraisemblance et $\hat{\sigma}^2$ est une estimation de la variance associée à chaque réponse.

4.3 Sélection de modèle

□ **Vocabulaire** – Lors de la sélection d'un modèle, on divise les données en 3 différentes parties comme suit :

Training set	Validation set	Testing set
<ul style="list-style-type: none"> - Modèle est entraîné - Normalement 80% du dataset 	<ul style="list-style-type: none"> - Modèle est évalué - Normalement 20% du dataset - Aussi appelé hold-out ou development set 	<ul style="list-style-type: none"> - Modèle donne des prédictions - Données jamais vues

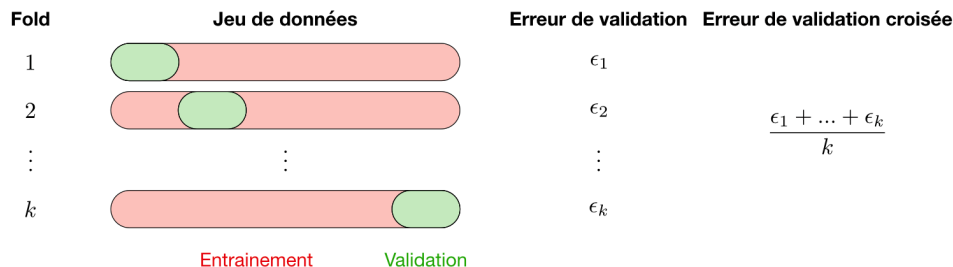
Une fois que le modèle a été choisi, il est entraîné sur le jeu de données entier et testé sur test set (qui n'a jamais été vu). Ces derniers sont représentés dans la figure ci-dessous :



□ **Validation croisée** – La validation croisée, aussi notée CV (de l'anglais *Cross-Validation*), est une méthode qui est utilisée pour sélectionner un modèle qui ne s'appuie pas trop sur le training set de départ. Les différents types de validation croisée rencontrés sont résumés dans le tableau ci-dessous :

<i>k</i> -fold	Leave- <i>p</i> -out
<ul style="list-style-type: none"> - Entraînement sur $k - 1$ folds et évaluation sur le fold restant - Généralement $k = 5$ ou 10 	<ul style="list-style-type: none"> - Entraînement sur $n - p$ observations et évaluation sur les p restantes - Cas $p = 1$ est appelé <i>leave-one-out</i>

La méthode la plus utilisée est appelée validation croisée *k*-fold et partage le jeu de données d'entraînement en *k* folds, de manière à valider le modèle sur un fold tout en entraînant le modèle sur les $k - 1$ autres folds, tout ceci *k* fois. L'erreur est alors moyennée sur *k* folds et est appelée erreur de validation croisée.



□ **Régularisation** – La procédure de régularisation a pour but d'éviter que le modèle ne surapprenne (en anglais *overfit*) les données et ainsi vise à régler les problèmes de grande variance. Le tableau suivant récapitule les différentes techniques de régularisation communément utilisées.

LASSO	Ridge	Elastic Net
<ul style="list-style-type: none"> - Réduit les coefficients à 0 - Bon pour la sélection de variables 	Rend les coefficients plus petits	Compromis entre la sélection de variables et la réduction de coefficients
$\dots + \lambda \ \theta\ _1$ $\lambda \in \mathbb{R}$	$\dots + \lambda \ \theta\ _2^2$ $\lambda \in \mathbb{R}$	$\dots + \lambda \left[(1 - \alpha) \ \theta\ _1 + \alpha \ \theta\ _2^2 \right]$ $\lambda \in \mathbb{R}, \alpha \in [0, 1]$

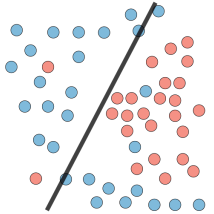
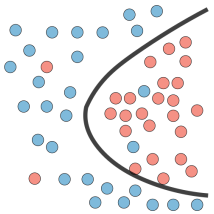
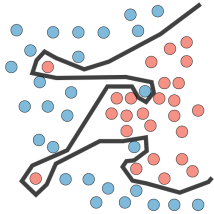
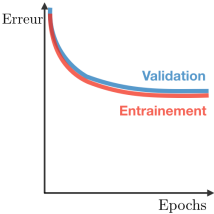
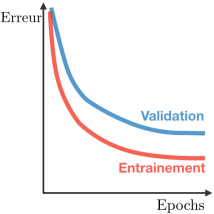
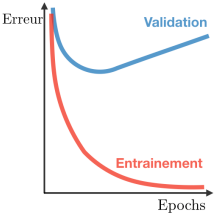
4.4 Diagnostics

□ **Biais** – Le biais d'un modèle est la différence entre l'espérance de la prédiction et du modèle correct pour lequel on essaie de prédire pour des observations données.

□ **Variance** – La variance d'un modèle est la variabilité des prédictions d'un modèle pour des observations données.

□ **Compromis biais/variance** – Plus le modèle est simple, plus le biais est grand et plus le modèle est complexe, plus la variance est grande.

	Underfitting	Just right	Overfitting
Symptômes	<ul style="list-style-type: none"> - Erreur de training élevé - Erreur de training proche de l'erreur de test - Biais élevé 	<ul style="list-style-type: none"> - Erreur de training légèrement inférieure à l'erreur de test 	<ul style="list-style-type: none"> - Erreur de training très faible - Erreur de training beaucoup plus faible que l'erreur de test - Variance élevée
Régression			

<p>Classification</p>			
<p>Deep Learning</p>			
<p>Remèdes</p>	<ul style="list-style-type: none"> - Complexifier le modèle - Ajouter plus de variables - Laisser le training pendant plus de temps 		<ul style="list-style-type: none"> - Effectuer une régularisation - Avoir plus de données

□ **Analyse de l'erreur** – L'analyse de l'erreur consiste à analyser la cause première de la différence en performance entre le modèle actuel et le modèle parfait.

□ **Analyse ablative** – L'analyse ablative consiste à analyser la cause première de la différence en performance entre le modèle actuel et le modèle de base.

5 Rappels

5.1 Probabilités et Statistiques

5.1.1 Introduction aux probabilités à l'analyse combinatoire

□ **Univers de probabilités** – L'ensemble de toutes les issues possibles d'une expérience aléatoire est appelé l'univers de probabilités d'une expérience aléatoire et est noté S .

□ **Évènement** – Toute partie E d'un univers est appelé un évènement. Ainsi, un évènement est un ensemble d'issues possibles d'une expérience aléatoire. Si l'issue de l'expérience aléatoire est contenue dans E , alors on dit que E s'est produit.

□ **Axiomes de probabilités** – Pour chaque évènement E , on note $P(E)$ la probabilité que l'évènement E se produise.

$$(1) \quad 0 \leq P(E) \leq 1 \quad (2) \quad P(S) = 1 \quad (3) \quad P\left(\bigcup_{i=1}^n E_i\right) = \sum_{i=1}^n P(E_i)$$

□ **Permutation** – Une permutation est un arrangement de r objets parmi n objets, dans un ordre donné. Le nombre de tels arrangements est donné par $P(n,r)$, défini par :

$$P(n,r) = \frac{n!}{(n-r)!}$$

□ **Combinaison** – Une combinaison est un arrangement de r objets parmi n objets, où l'ordre ne compte pas. Le nombre de tels arrangements est donné par $C(n,r)$, défini par :

$$C(n,r) = \frac{P(n,r)}{r!} = \frac{n!}{r!(n-r)!}$$

Remarque : on note que pour $0 \leq r \leq n$, on a $P(n,r) \geq C(n,r)$.

5.1.2 Probabilité conditionnelle

□ **Théorème de Bayes** – Pour des évènements A et B tels que $P(B) > 0$, on a :

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Remarque : on a $P(A \cap B) = P(A)P(B|A) = P(A|B)P(B)$.

□ **Partition** – Soit $\{A_i, i \in [1,n]\}$ tel que pour tout i , $A_i \neq \emptyset$. On dit que $\{A_i\}$ est une partition si l'on a :

$$\forall i \neq j, A_i \cap A_j = \emptyset \quad \text{et} \quad \bigcup_{i=1}^n A_i = S$$

Remarque : pour tout évènement B dans l'univers de probabilités, on a $P(B) = \sum_{i=1}^n P(B|A_i)P(A_i)$.

□ **Formule étendue du théorème de Bayes** – Soit $\{A_i, i \in [1, n]\}$ une partition de l'univers de probabilités. On a :

$$P(A_k|B) = \frac{P(B|A_k)P(A_k)}{\sum_{i=1}^n P(B|A_i)P(A_i)}$$

□ **Indépendance** – Deux évènements A et B sont dits indépendants si et seulement si on a :

$$P(A \cap B) = P(A)P(B)$$

5.1.3 Variable aléatoires

□ **Variable aléatoire** – Une variable aléatoire, souvent notée X , est une fonction qui associe chaque élément de l'univers de probabilité à la droite des réels.

□ **Fonction de répartition** – La fonction de répartition F (en anglais *CDF - Cumulative distribution function*), qui est croissante monotone et telle que

$$\lim_{x \rightarrow -\infty} F(x) = 0 \quad \text{et} \quad \lim_{x \rightarrow +\infty} F(x) = 1$$

est définie de la manière suivante :

$$F(x) = P(X \leq x)$$

Remarque : on a $P(a < X \leq B) = F(b) - F(a)$.

□ **Densité de probabilité** – La densité de probabilité f (en anglais *PDF - Probability density function*) est la probabilité que X prenne des valeurs entre deux réalisations adjacentes d'une variable aléatoire.

□ **Relations vérifiées par les PDF et CDF** – Voici les propriétés importantes à savoir dans les cas discret (D) et continu (C).

Case	CDF F	PDF f	Propriétés du PDF
(D)	$F(x) = \sum_{x_i \leq x} P(X = x_i)$	$f(x_j) = P(X = x_j)$	$0 \leq f(x_j) \leq 1$ and $\sum_j f(x_j) = 1$
(C)	$F(x) = \int_{-\infty}^x f(y)dy$	$f(x) = \frac{dF}{dx}$	$f(x) \geq 0$ and $\int_{-\infty}^{+\infty} f(x)dx = 1$

□ **Variance** – La variance d'une variable aléatoire, souvent notée $\text{Var}(X)$ ou σ^2 , est une mesure de la dispersion de ses fonctions de distribution. Elle est déterminée de la manière suivante :

$$\text{Var}(X) = E[(X - E[X])^2] = E[X^2] - E[X]^2$$

□ **Écart-type** – L'écart-type d'une variable aléatoire, souvent notée σ , est une mesure de la dispersion de sa fonction de distribution, exprimée avec les même unités que la variable aléatoire. Il est déterminé de la manière suivante :

$$\sigma = \sqrt{\text{Var}(X)}$$

□ **Espérance et moments de la distribution** – Voici les expressions de l'espérance $E[X]$, l'espérance généralisée $E[g(X)]$, $k^{\text{ième}}$ moment $E[X^k]$ et fonction caractéristique $\psi(\omega)$ dans les cas discret et continu.

Case	$E[X]$	$E[g(X)]$	$E[X^k]$	$\psi(\omega)$
(D)	$\sum_{i=1}^n x_i f(x_i)$	$\sum_{i=1}^n g(x_i) f(x_i)$	$\sum_{i=1}^n x_i^k f(x_i)$	$\sum_{i=1}^n f(x_i) e^{i\omega x_i}$
(C)	$\int_{-\infty}^{+\infty} x f(x) dx$	$\int_{-\infty}^{+\infty} g(x) f(x) dx$	$\int_{-\infty}^{+\infty} x^k f(x) dx$	$\int_{-\infty}^{+\infty} f(x) e^{i\omega x} dx$

Remarque : on a $e^{i\omega x} = \cos(\omega x) + i \sin(\omega x)$.

□ **Transformation de variables aléatoires** – Soit X, Y des variables liées par une certaine fonction. En notant f_X et f_Y les fonctions de distribution de X et Y respectivement, on a :

$$f_Y(y) = f_X(x) \left| \frac{dx}{dy} \right|$$

□ **Loi d'intégration de Leibniz** – Soit g une fonction de x et potentiellement c , et a, b , les limites de l'intervalle qui peuvent dépendre de c . On a :

$$\frac{\partial}{\partial c} \left(\int_a^b g(x) dx \right) = \frac{\partial b}{\partial c} \cdot g(b) - \frac{\partial a}{\partial c} \cdot g(a) + \int_a^b \frac{\partial g}{\partial c}(x) dx$$

□ **Inégalité de Tchebychev** – Soit X une variable aléatoire de moyenne μ . Pour $k, \sigma > 0$, on a l'inégalité suivante :

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

5.1.4 Variables aléatoires conjointement distribuées

□ **Densité conditionnelle** – La densité conditionnelle de X par rapport à Y , souvent notée $f_{X|Y}$, est définie de la manière suivante :

$$f_{X|Y}(x) = \frac{f_{XY}(x, y)}{f_Y(y)}$$

□ **Indépendance** – Deux variables aléatoires X et Y sont dits indépendantes si l'on a :

$$f_{XY}(x, y) = f_X(x) f_Y(y)$$

□ **Densité marginale et fonction de répartition** – À partir de la densité de probabilité f_{XY} , on a :

Cas	Densité marginale	Fonction de répartition
(D)	$f_X(x_i) = \sum_j f_{XY}(x_i, y_j)$	$F_{XY}(x, y) = \sum_{x_i \leq x} \sum_{y_j \leq y} f_{XY}(x_i, y_j)$
(C)	$f_X(x) = \int_{-\infty}^{+\infty} f_{XY}(x, y) dy$	$F_{XY}(x, y) = \int_{-\infty}^x \int_{-\infty}^y f_{XY}(x', y') dx' dy'$

□ **Covariance** – On définit la covariance de deux variables aléatoires X et Y , que l'on note σ_{XY}^2 ou plus souvent $\text{Cov}(X, Y)$, de la manière suivante :

$$\text{Cov}(X, Y) \triangleq \sigma_{XY}^2 = E[(X - \mu_X)(Y - \mu_Y)] = E[XY] - \mu_X \mu_Y$$

□ **Corrélation** – En notant σ_X, σ_Y les écart-types de X et Y , on définit la corrélation entre les variables aléatoires X et Y , que l'on note ρ_{XY} , de la manière suivante :

$$\rho_{XY} = \frac{\sigma_{XY}^2}{\sigma_X \sigma_Y}$$

Remarques : on note que pour toute variable aléatoire X, Y , on a $\rho_{XY} \in [-1, 1]$. Si X et Y sont indépendants, alors $\rho_{XY} = 0$.

□ **Distributions importantes** – Voici les distributions importantes à savoir :

Type	Distribution	PDF	$\psi(\omega)$	$E[X]$	$\text{Var}(X)$
(D)	$X \sim \mathcal{B}(n, p)$ Binomial	$P(X = x) = \binom{n}{x} p^x q^{n-x}$ $x \in \llbracket 0, n \rrbracket$	$(pe^{i\omega} + q)^n$	np	npq
	$X \sim \text{Po}(\mu)$ Poisson	$P(X = x) = \frac{\mu^x}{x!} e^{-\mu}$ $x \in \mathbb{N}$	$e^{\mu(e^{i\omega} - 1)}$	μ	μ
(C)	$X \sim \mathcal{U}(a, b)$ Uniforme	$f(x) = \frac{1}{b-a}$ $x \in [a, b]$	$\frac{e^{i\omega b} - e^{i\omega a}}{(b-a)i\omega}$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$
	$X \sim \mathcal{N}(\mu, \sigma)$ Gaussien	$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ $x \in \mathbb{R}$	$e^{i\omega\mu - \frac{1}{2}\omega^2\sigma^2}$	μ	σ^2
	$X \sim \text{Exp}(\lambda)$ Exponentiel	$f(x) = \lambda e^{-\lambda x}$ $x \in \mathbb{R}_+$	$\frac{1}{1 - i\omega/\lambda}$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$

5.1.5 Estimation des paramètres

□ **Échantillon aléatoire** – Un échantillon aléatoire est une collection de n variables aléatoires X_1, \dots, X_n qui sont indépendantes et identiquement distribuées avec X .

□ **Estimateur** – Un estimateur est une fonction des données qui est utilisée pour trouver la valeur d'un paramètre inconnu dans un modèle statistique.

□ **Biais** – Le biais d'un estimateur $\hat{\theta}$ est défini comme étant la différence entre l'espérance de la distribution de $\hat{\theta}$ et de la valeur vraie, i.e. :

$$\text{Bias}(\hat{\theta}) = E[\hat{\theta}] - \theta$$

Remarque : un estimateur est dit non biaisé lorsque l'on a $E[\hat{\theta}] = \theta$.

□ **Moyenne empirique et variance empirique** – La moyenne empirique et la variance empirique d'un échantillon aléatoire sont utilisées pour estimer la valeur vraie μ et la variance vraie σ^2 d'une distribution, notés \bar{X} et s^2 et sont définies de la manière suivante :

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i \quad \text{and} \quad s^2 = \hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2$$

□ **Théorème de la limite centrale** – Soit un échantillon aléatoire X_1, \dots, X_n suivant une distribution donnée de moyenne μ et de variance σ^2 , alors on a :

$$\bar{X} \underset{n \rightarrow +\infty}{\sim} \mathcal{N}\left(\mu, \frac{\sigma}{\sqrt{n}}\right)$$

5.2 Algèbre linéaire et Analyse

5.2.1 Notations générales

□ **Vecteur** – On note $x \in \mathbb{R}^n$ un vecteur à n entrées, où $x_i \in \mathbb{R}$ est la $i^{\text{ème}}$ entrée :

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n$$

□ **Matrice** – On note $A \in \mathbb{R}^{m \times n}$ une matrice à m lignes et n colonnes, où $A_{i,j} \in \mathbb{R}$ est l'entrée située à la $i^{\text{ème}}$ ligne et $j^{\text{ème}}$ colonne :

$$A = \begin{pmatrix} A_{1,1} & \cdots & A_{1,n} \\ \vdots & & \vdots \\ A_{m,1} & \cdots & A_{m,n} \end{pmatrix} \in \mathbb{R}^{m \times n}$$

Remarque : le vecteur x défini ci-dessus peut être vu comme une matrice $n \times 1$ et est aussi appelé vecteur colonne.

□ **Matrice identité** – La matrice identité $I \in \mathbb{R}^{n \times n}$ est une matrice carrée avec des 1 sur sa diagonale et des 0 partout ailleurs :

$$I = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

Remarque : pour toute matrice $A \in \mathbb{R}^{n \times n}$, on a $A \times I = I \times A = A$.

□ **Matrice diagonale** – Une matrice diagonale $D \in \mathbb{R}^{n \times n}$ est une matrice carrée avec des valeurs non nulles sur sa diagonale et des zéros partout ailleurs.

$$D = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & d_n \end{pmatrix}$$

Remarque : on note aussi $D = \text{diag}(d_1, \dots, d_n)$.

5.2.2 Opérations matricielles

□ **Vecteur-vecteur** – Il y a deux types de multiplication vecteur-vecteur :

– Produit scalaire : pour $x, y \in \mathbb{R}^n$, on a :

$$x^T y = \sum_{i=1}^n x_i y_i \in \mathbb{R}$$

– Produit dyadique : pour $x \in \mathbb{R}^m, y \in \mathbb{R}^n$, on a :

$$xy^T = \begin{pmatrix} x_1 y_1 & \cdots & x_1 y_n \\ \vdots & & \vdots \\ x_m y_1 & \cdots & x_m y_n \end{pmatrix} \in \mathbb{R}^{m \times n}$$

□ **Matrice-vecteur** – Le produit de la matrice $A \in \mathbb{R}^{m \times n}$ et du vecteur $x \in \mathbb{R}^n$ est un vecteur de taille \mathbb{R}^m , tel que :

$$Ax = \begin{pmatrix} a_{r,1}^T x \\ \vdots \\ a_{r,m}^T x \end{pmatrix} = \sum_{i=1}^n a_{c,i} x_i \in \mathbb{R}^m$$

où $a_{r,i}^T$ sont les vecteurs-ligne et $a_{c,j}$ sont les vecteurs-colonne de A et x_i sont les entrées de x .

□ **Matrice-matrice** – Le produit des matrices $A \in \mathbb{R}^{m \times n}$ et $B \in \mathbb{R}^{n \times p}$ est une matrice de taille $\mathbb{R}^{m \times p}$, tel que :

$$AB = \begin{pmatrix} a_{r,1}^T b_{c,1} & \cdots & a_{r,1}^T b_{c,p} \\ \vdots & & \vdots \\ a_{r,m}^T b_{c,1} & \cdots & a_{r,m}^T b_{c,p} \end{pmatrix} = \sum_{i=1}^n a_{c,i} b_{r,i}^T \in \mathbb{R}^{m \times p}$$

où $a_{r,i}^T, b_{r,i}^T$ sont des vecteurs-ligne et $a_{c,j}, b_{c,j}$ sont des vecteurs-colonne de A et B respectivement.

□ **Transposée** – La transposée est une matrice $A \in \mathbb{R}^{m \times n}$, notée A^T , qui est telle que ses entrées sont renversées.

$$\forall i, j, \quad A_{i,j}^T = A_{j,i}$$

Remarque : pour des matrices A, B , on a $(AB)^T = B^T A^T$.

□ **Inverse** – L'inverse d'une matrice carrée inversible A est notée A^{-1} et est l'unique matrice telle que :

$$AA^{-1} = A^{-1}A = I$$

Remarque : toutes les matrices carrées ne sont pas inversibles. Aussi, pour des matrices A, B , on a $(AB)^{-1} = B^{-1}A^{-1}$.

□ **Trace** – La trace d'une matrice carrée A , notée $\text{tr}(A)$, est la somme de ses entrées diagonales :

$$\text{tr}(A) = \sum_{i=1}^n A_{i,i}$$

Remarque : pour toutes matrices A, B , on a $\text{tr}(A^T) = \text{tr}(A)$ et $\text{tr}(AB) = \text{tr}(BA)$.

□ **Déterminant** – Le déterminant d'une matrice carrée $A \in \mathbb{R}^{n \times n}$ notée $|A|$ ou $\det(A)$ est exprimée récursivement en termes de $A_{\setminus i, \setminus j}$, qui est la matrice A sans sa $i^{\text{ème}}$ ligne et $j^{\text{ème}}$ colonne, de la manière suivante :

$$\det(A) = |A| = \sum_{j=1}^n (-1)^{i+j} A_{i,j} |A_{\setminus i, \setminus j}|$$

Remarque : A est inversible si et seulement si $|A| \neq 0$. Aussi, $|AB| = |A||B|$ et $|A^T| = |A|$.

5.2.3 Propriétés matricielles

□ **Décomposition symétrique** – Une matrice donnée A peut être exprimée en termes de ses parties symétrique et antisymétrique de la manière suivante :

$$A = \underbrace{\frac{A + A^T}{2}}_{\text{Symétrique}} + \underbrace{\frac{A - A^T}{2}}_{\text{Antisymétrique}}$$

□ **Norme** – Une norme est une fonction $N : V \rightarrow [0, +\infty[$ où V est un espace vectoriel, et tel que pour tous $x, y \in V$, on a :

- $N(x + y) \leq N(x) + N(y)$
- $N(ax) = |a|N(x)$ pour a scalaire
- si $N(x) = 0$, alors $x = 0$

Pour $x \in V$, les normes les plus utilisées sont récapitulées dans le tableau ci-dessous :

Norme	Notation	Définition	Cas
Manhattan, L^1	$\ x\ _1$	$\sum_{i=1}^n x_i $	LASSO
Euclidien, L^2	$\ x\ _2$	$\sqrt{\sum_{i=1}^n x_i^2}$	Ridge
p -norme, L^p	$\ x\ _p$	$\left(\sum_{i=1}^n x_i^p\right)^{\frac{1}{p}}$	Inégalité de Hölder
Infini, L^∞	$\ x\ _\infty$	$\max_i x_i $	Convergence uniforme

□ **Dépendance linéaire** – Un ensemble de vecteurs est considéré comme étant linéairement dépendant si un des vecteurs de cet ensemble peut être défini comme une combinaison des autres.

Remarque : si aucun vecteur ne peut être noté de cette manière, alors les vecteurs sont dits linéairement indépendants.

□ **Rang d'une matrice** – Le rang d'une matrice donnée A est notée $\text{rang}(A)$ et est la dimension de l'espace vectoriel généré par ses colonnes. Ceci est équivalent au nombre maximum de colonnes indépendantes de A .

□ **Matrice semi-définie positive** – Une matrice $A \in \mathbb{R}^{n \times n}$ est semi-définie positive et est notée $A \succeq 0$ si l'on a :

$$A = A^T \quad \text{et} \quad \forall x \in \mathbb{R}^n, \quad x^T A x \geq 0$$

Remarque : de manière similaire, une matrice A est dite définie positive et est notée $A \succ 0$ si elle est semi-définie positive et que pour tout vector x non-nul, on a $x^T A x > 0$.

□ **Valeur propre, vecteur propre** – Étant donné une matrice $A \in \mathbb{R}^{n \times n}$, λ est une valeur propre de A s'il existe un vecteur $z \in \mathbb{R}^n \setminus \{0\}$, appelé vecteur propre, tel que :

$$Az = \lambda z$$

□ **Théorème spectral** – Soit $A \in \mathbb{R}^{n \times n}$. Si A est symétrique, alors A est diagonalisable par une matrice orthogonale réelle $U \in \mathbb{R}^{n \times n}$. En notant $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$, on a :

$$\exists \Lambda \text{ diagonal, } A = U \Lambda U^T$$

□ **Décomposition en valeurs singulières** – Pour une matrice A de dimensions $m \times n$, la décomposition en valeurs singulières est une technique de factorisation qui garantit l'existence d'une matrice unitaire U $m \times m$, d'une matrice diagonale Σ $m \times n$ et d'une matrice unitaire V $n \times n$, tel que :

$$A = U \Sigma V^T$$

5.2.4 Analyse matricielle

□ **Gradient** – Soit $f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ une fonction et $A \in \mathbb{R}^{m \times n}$ une matrice. Le gradient de f par rapport à A est une matrice de taille $m \times n$, notée $\nabla_A f(A)$, telle que :

$$\left(\nabla_A f(A)\right)_{i,j} = \frac{\partial f(A)}{\partial A_{i,j}}$$

Remarque : le gradient de f est seulement défini lorsque f est une fonction donnant un scalaire.

□ **Hessienne** – Soit $f : \mathbb{R}^n \rightarrow \mathbb{R}$ une fonction et $x \in \mathbb{R}^n$ un vecteur. La hessienne de f par rapport à x est une matrice symétrique $n \times n$, notée $\nabla_x^2 f(x)$, telle que :

$$\left(\nabla_x^2 f(x)\right)_{i,j} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$$

Remarque : la hessienne de f est seulement définie lorsque f est une fonction qui donne un scalaire.

□ **Opérations de gradient** – Pour des matrices A, B, C , les propriétés de gradient suivants sont bons à savoir :

$$\nabla_A \text{tr}(AB) = B^T \quad \nabla_{A^T} f(A) = (\nabla_A f(A))^T$$

$$\nabla_A \text{tr}(ABA^T C) = CAB + C^T A B^T \quad \nabla_A |A| = |A| (A^{-1})^T$$