# Mini-Projet d'Optimisation ML

# Modélisation, SGD et Méthodes Proximales

**Étudiant :** Dia Ibrahima Alassane Alassan

**Matricule :** C30813

**Filière :** M2 Statistique et Science de Données (SSD)

**Faculté :** Sciences et Techniques

**Année universitaire :** 2025-2026

# 1 Phase 1 : Fondements et Gradient Déterministe (Chap. 1 & 2)

## 1.1 Question 1 : Analyse

On considère la fonction objectif définie par

$$F(w) = \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + e^{-y_i x_i^\top w}\right) + \frac{\lambda}{2}\|w\|^2, \quad w \in R^d.$$

### 1.1.1 Régularité

Posons, pour tout $i = 1, \ldots, n$,

$$\ell_i(w) = \log\left(1 + e^{-y_i x_i^\top w}\right).$$

Considérons la fonction scalaire

$$\varphi(t) = \log(1 + e^t), \quad t \in R.$$

On a

$$\varphi'(t) = \frac{e^t}{1+e^t}, \qquad \varphi''(t) = \frac{e^t}{(1+e^t)^2},$$

ce qui montre que $\varphi \in C^2(R)$.

La fonction $w \mapsto -y_i x_i^\top w$ est affine, donc de classe $C^\infty$. Par composition, $\ell_i(w) = \varphi(-y_i x_i^\top w)$ est de classe $C^2$ sur $R^d$. Le terme de régularisation $\frac{\lambda}{2}\|w\|^2$ est un polynôme, donc également de classe $C^2$. La somme et la moyenne de fonctions de classe $C^2$ étant encore de classe $C^2$, on en déduit que

$$F \in C^2(R^d).$$

### 1.1.2 Convexité

La fonction $\varphi$ est convexe puisque $\varphi''(t) \geq 0$ pour tout $t \in R$. La composition d'une fonction convexe avec une application affine conserve la convexité, ce qui implique que $\ell_i$ est convexe sur $R^d$ pour tout $i$. Le terme de régularisation $\frac{\lambda}{2}\|w\|^2$ est également convexe. La somme (et la moyenne) de fonctions convexes étant convexe, la fonction $F$ est convexe.

### 1.1.3 Forte convexité

Pour tout $i$, la Hessienne de $\ell_i$ s'écrit

$$\nabla^2 \ell_i(w) = \varphi''(-y_i x_i^\top w)\, x_i x_i^\top,$$

qui est une matrice semi-définie positive. Ainsi,

$$\nabla^2 \left(\frac{1}{n}\sum_{i=1}^{n} \ell_i(w)\right) \succeq 0.$$

Par ailleurs,

$$\nabla^2 \left( \frac{\lambda}{2} \|w\|^2 \right) = \lambda I.$$

On obtient alors, pour tout $w \in R^d$,

$$\nabla^2 F(w) \succeq \lambda I.$$

Par conséquent, la fonction $F$ est $\lambda$-fortement convexe pour $\lambda > 0$, ce qui garantit l'existence et l'unicité du minimiseur global.

## 1.2 Question 2 : Gradient et Lipschitzianité

On considère la fonction objectif

$$F(w) = \frac{1}{n} \sum_{i=1}^{n} \log \left( 1 + e^{-y_i x_i^\top w} \right) + \frac{\lambda}{2} \|w\|^2, \quad w \in R^d.$$

### 1.2.1 Lipschitzianité du gradient

La fonction $F$ est de classe $C^2$. Pour montrer que son gradient est $L$-Lipschitzien, il suffit de majorer la Hessienne.

La Hessienne de $\ell_i$ est donnée par

$$\nabla^2 \ell_i(w) = \varphi''(-y_i x_i^\top w)\, x_i x_i^\top,$$

où $\varphi(t) = \log(1 + e^t)$ et

$$\varphi''(t) = \frac{e^t}{(1 + e^t)^2}.$$

On a, pour tout $t \in R$,

$$0 \leq \varphi''(t) \leq \frac{1}{4}.$$

Il en résulte que

$$0 \preceq \nabla^2 \ell_i(w) \preceq \frac{1}{4} x_i x_i^\top.$$

En sommant et en moyennant,

$$\nabla^2 \left( \frac{1}{n} \sum_{i=1}^{n} \ell_i(w) \right) \preceq \frac{1}{4n} \sum_{i=1}^{n} x_i x_i^\top.$$

Par ailleurs,

$$\nabla^2 \left( \frac{\lambda}{2} \|w\|^2 \right) = \lambda I.$$

On obtient donc

$$\nabla^2 F(w) \preceq \frac{1}{4n} \sum_{i=1}^{n} x_i x_i^\top + \lambda I.$$

### 1.2.2 Expression de la constante de Lipschitz

Soit $X \in R^{n \times d}$ la matrice des données dont la $i$-ème ligne est $x_i^\top$. On a

$$X^\top X = \sum_{i=1}^{n} x_i x_i^\top.$$

Ainsi,

$$\nabla^2 F(w) \preceq \frac{1}{4n} X^\top X + \lambda I.$$

La plus grande valeur propre de cette matrice est majorée par

$$L = \lambda + \frac{1}{4n} \|X\|_2^2,$$

où $\|X\|_2$ désigne la norme spectrale de $X$. Par conséquent, le gradient de $F$ est $L$-Lipschitzien et vérifie

$$\|\nabla F(u) - \nabla F(v)\| \leq L\|u - v\|, \quad \forall u, v \in R^d.$$

## 1.3 Question 3 : Descente de Gradient à pas fixe

Dans cette partie, nous implémentons l'algorithme de descente de gradient pour résoudre le problème d'optimisation associé à une régression logistique avec régularisation de type Ridge.

### 1.3.1 Principe de la descente de gradient

La méthode de la descente de gradient consiste à construire une suite $(w_k)_{k \geq 0}$ qui converge vers le minimiseur $w$ en suivant itérativement la direction opposée au gradient. À partir d'une initialisation $w_0 \in R^d$, l'algorithme s'écrit :

$$w_{k+1} = w_k - \alpha \nabla F(w_k),$$

où $\alpha > 0$ est un pas d'apprentissage constant.

Intuitivement, le gradient $\nabla F(w_k)$ indique la direction de plus forte augmentation de la fonction $F$, et le déplacement dans la direction opposée permet de faire décroître la valeur de la fonction objectif.

### 1.3.2 Choix du pas d'apprentissage

D'après la Question 2, le gradient de $F$ est $L$-Lipschitzien, avec une constante $L$ donnée par :

$$L = \lambda + \frac{1}{4n} \|X\|_2^2,$$

où $X \in R^{n \times d}$ est la matrice des données et $\|\cdot\|_2$ désigne la norme spectrale.

Lorsque le pas $\alpha$ vérifie

$$0 < \alpha \leq \frac{1}{L},$$

la descente de gradient est garantie convergente. Dans ce cas, la suite $(F(w_k))_{k \geq 0}$ est décroissante et converge vers la valeur minimale $F(w^)$.

### 1.3.3 Convergence

Étant donné que la fonction $F$ est $\lambda$-fortement convexe et que son gradient est $L$-Lipschitzien, la descente de gradient à pas fixe converge linéairement vers l'unique minimiseur $w$. Cette propriété justifie l'utilisation de cette méthode comme algorithme de référence avant de passer à des méthodes plus avancées telles que le gradient stochastique ou les méthodes proximales.

## 1.4 Implémentation numérique et choix du jeu de données

### 1.4.1 Choix du jeu de données

Pour la mise en pratique de l'algorithme de descente de gradient, nous nous sommes appuyés sur le jeu de données *Breast Cancer Wisconsin*, disponible dans la bibliothèque `scikit-learn`. Ce jeu de données est constitué de $n = 569$ observations décrites par $d = 30$ variables numériques, et correspond à un problème de classification binaire.

### 1.4.2 Implémentation de la descente de gradient

La méthode de la descente de gradient à pas fixe a été implémentée en langage Python à l'aide de la bibliothèque `NumPy`. À partir d'une initialisation $w_0 = 0$, les itérations ont été calculées selon la relation :

$$w_{k+1} = w_k - \alpha \nabla F(w_k),$$

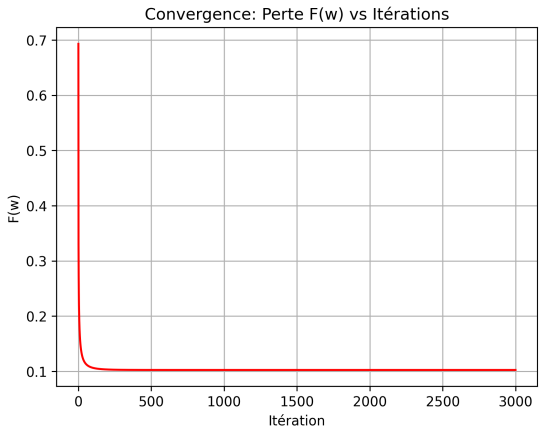
où le gradient $\nabla F(w)$ est calculé explicitement à partir de la formule analytique obtenue à la Question 2.

Le pas d'apprentissage $\alpha$ a été choisi en fonction de la constante de Lipschitz $L$ du gradient, avec $\alpha = 1/L$, ce qui garantit la convergence de l'algorithme. La convergence a été évaluée à l'aide de l'évolution de la valeur de la fonction objectif $F(w_k)$ ainsi que de la norme du gradient au cours des itérations.

### 1.4.3 Objectif de l'expérimentation

Cette implémentation numérique permet de confirmer expérimentalement les résultats théoriques présentés dans les sections précédentes et constitue une base solide avant d'aborder des méthodes plus avancées telles que le gradient stochastique ou les approches proximales.

Figure 1: Descente de gradient déterministe : évolution de la fonction objectif au cours des itérations.




La figure 1 illustre la décroissance progressive de la fonction objectif au cours de la descente de gradient à pas fixe. La diminution régulière de la perte confirme la stabilité de la méthode dans le cas lisse et fortement convexe. (régularisation $\ell_2$). On observe d'abord une phase de décroissance relativement rapide, suivie d'un ralentissement à mesure que l'on s'approche du minimiseur, un comportement caractéristique des méthodes de gradient appliquées à des fonctions convexes.

La combinaison de la diminution de la fonction objectif et de la norme du gradient fournit une validation numérique cohérente des résultats théoriques établis dans la Phase 1. Elle justifie également l'utilisation de ces deux quantités comme critères d'arrêt pertinents pour l'algorithme de descente de gradient.

### 1.4.4 Comparaison avec la méthode du Gradient Conjugué

Afin d'évaluer les performances de la descente de gradient à pas fixe, nous la comparons à la méthode du gradient conjugué appliquée au même problème d'optimisation.

La descente de gradient met à jour la solution en suivant systématiquement la direction opposée au gradient. Cette stratégie peut conduire à des oscillations et à une convergence relativement lente, notamment lorsque la fonction objectif présente des niveaux de courbure très variables.

À l'inverse, la méthode du gradient conjugué construit des directions de recherche conjuguées, ce qui lui permet d'exploiter plus efficacement la structure de courbure du problème. En pratique, cette approche converge généralement en un nombre d'itérations bien inférieur à celui requis par la descente de gradient classique.

Les résultats numériques indiquent que la méthode du gradient conjugué atteint une valeur de la fonction objectif comparable, voire légèrement inférieure, tout en nécessitant un temps d'exécution et un nombre d'itérations nettement

plus faibles. Cette comparaison met clairement en évidence l'intérêt des méthodes de type gradient conjugué pour les problèmes d'optimisation lisses et fortement convexes.

Contrairement à la descente de gradient classique, le gradient conjugué construit des directions de recherche adaptatives et détermine automatiquement le pas optimal à chaque itération. Par conséquent, l'analyse de sa convergence s'appuie principalement sur des indicateurs numériques tels que le nombre d'itérations ou le temps de calcul, plutôt que sur des courbes d'évolution détaillées.

# 2 Phase 2

## 2.1 Gradient Stochastique (SGD)

Lorsque $n$ devient très grand, le calcul du gradient complet

$$\nabla F(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla \ell_i(w) + \lambda w$$

devient coûteux. On utilise alors une méthode stochastique consistant à approximer le gradient à partir d'un seul exemple (ou d'un mini-lot).

On rappelle :

$$F(w) = \frac{1}{n}\sum_{i=1}^{n} \ell_i(w) + \frac{\lambda}{2}\|w\|^2, \qquad \ell_i(w) = \log\left(1 + e^{-y_i x_i^\top w}\right).$$

Le gradient de $\ell_i$ est :

$$\nabla \ell_i(w) = -\frac{y_i x_i}{1 + e^{y_i x_i^\top w}}.$$

À l'itération $k$, on tire un indice $i_k$ uniformément dans $\{1, \ldots, n\}$ et on définit l'estimateur stochastique du gradient :

$$g_{i_k}(w_k) = \nabla \ell_{i_k}(w_k) + \lambda w_k = -\frac{y_{i_k} x_{i_k}}{1 + e^{y_{i_k} x_{i_k}^\top w_k}} + \lambda w_k.$$

L'algorithme de SGD est alors :

$$w_{k+1} = w_k - \alpha_k \, g_{i_k}(w_k),$$

où $(\alpha_k)_{k\geq 0}$ est une suite de pas décroissants. Une règle classique est :

$$\alpha_k = \frac{\alpha_0}{\sqrt{1+k}},$$

qui garantit une décroissance progressive du pas et stabilise l'optimisation au cours des itérations.

### 2.1.1 Implémentation numérique du gradient stochastique

Pour illustrer le passage à l'échelle stochastique, nous avons implémenté l'algorithme du gradient stochastique (SGD) sur le même jeu de données que celui utilisé en Phase 1. Les données avaient déjà été chargées et prétraitées (centrage et réduction) lors de l'étude déterministe, puis réutilisées pour éviter toute redondance inutile.

À chaque itération, une observation est sélectionnée aléatoirement parmi l'ensemble du jeu de données, et un estimateur stochastique du gradient est calculé à partir de ce seul point. Le vecteur des paramètres est ensuite mis à jour à l'aide d'un pas d'apprentissage décroissant, suivant une règle de type

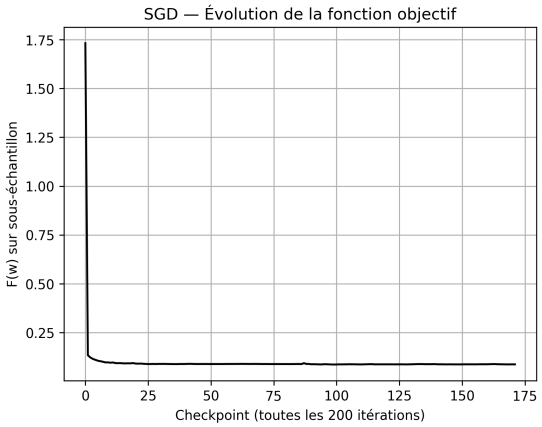
$$\alpha_k = \frac{\alpha_0}{\sqrt{1+k}},$$

permettant de stabiliser l'algorithme au cours des itérations.

Le suivi de la convergence est effectué en évaluant périodiquement la fonction objectif sur un sous-échantillon fixe du jeu de données. Cette approche permet de réduire le coût de calcul tout en fournissant une estimation fiable de l'évolution de la performance du modèle.

Les résultats numériques montrent une décroissance globale de la fonction objectif, bien que celle-ci soit plus irrégulière que dans le cas de la descente de gradient déterministe. Ce comportement est typique des méthodes stochastiques et s'explique par la variance introduite par l'échantillonnage aléatoire. Malgré cette variabilité, l'algorithme converge vers une solution satisfaisante, confirmant l'efficacité du gradient stochastique pour les problèmes de grande dimension.

Figure 2: SGD : évolution de la fonction objectif (évaluation périodique sur un sous-échantillon).




La figure 2 met en évidence le comportement typique du gradient stochastique : la fonction objectif diminue globalement, tout en présentant des fluctuations

plus importantes que dans le cas déterministe. Ces oscillations sont dues à la variance introduite par l'échantillonnage aléatoire des observations, mais la tendance moyenne reste clairement décroissante.

## 2.2 Optimiseurs modernes — Adam vs RMSProp

RMSProp repose sur une moyenne mobile des carrés des gradients, ce qui permet de normaliser les mises à jour et de limiter les effets d'une variance élevée des gradients. Cette approche conduit à des mises à jour plus stables que celles du gradient stochastique standard, en particulier au début de l'optimisation.

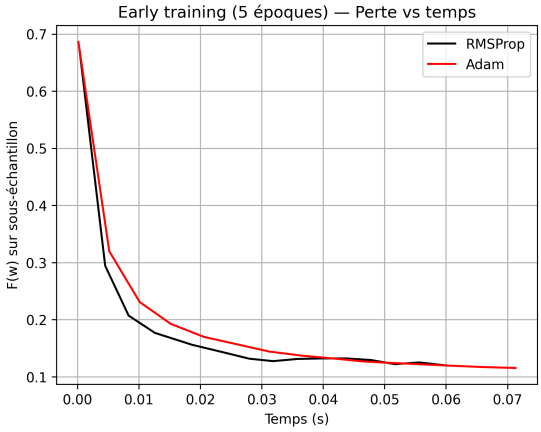
Adam combine le principe de RMSProp avec un terme de momentum, en maintenant simultanément une moyenne mobile des gradients et une moyenne mobile de leurs carrés. Grâce à cette double estimation, Adam bénéficie d'une adaptation plus fine du pas d'apprentissage et d'une correction du biais présent lors des premières itérations.

Les résultats obtenus sur les premières époques d'entraînement montrent que les deux méthodes assurent une décroissance rapide de la fonction objectif, bien que les trajectoires restent plus irrégulières que dans le cas déterministe. Cependant, Adam présente généralement une convergence initiale plus rapide et plus régulière que RMSProp, entraînant une diminution plus marquée de la perte et une amélioration plus rapide des performances de classification. Ce comportement souligne l'intérêt d'Adam pour les phases initiales de l'entraînement, où la vitesse de convergence est un critère déterminant.

En termes de valeur de la fonction objectif, Adam atteint une perte légèrement plus faible que RMSProp, traduisant une convergence initiale plus efficace. En revanche, cette amélioration s'accompagne d'un coût en temps plus élevé, Adam nécessitant un temps d'exécution supérieur sur les premières époques considérées. Ce surcoût s'explique par le calcul supplémentaire des moments d'ordre un et deux à chaque itération.

Ces résultats illustrent un compromis classique entre vitesse de calcul et qualité de la convergence initiale : RMSProp offre un temps de calcul plus réduit, tandis qu'Adam garantit une convergence initiale légèrement plus précise.

Figure 3: Courbes de convergence (Perte vs Temps) pour RMSProp et Adam sur les premières époques d'entraînement.




On constate que les deux optimiseurs permettent une diminution rapide de la fonction objectif dès les premières secondes d'entraînement. Adam atteint une valeur de perte légèrement inférieure à celle de RMSProp, traduisant une convergence initiale plus efficace. Toutefois, cette amélioration s'accompagne d'un temps de calcul plus élevé, en raison du calcul supplémentaire des moments d'ordre un et deux.

Cette comparaison illustre le compromis classique entre vitesse d'exécution et qualité de la convergence : RMSProp offre un temps d'exécution plus réduit, tandis qu'Adam garantit une convergence légèrement plus précise pour un temps comparable.

## 2.3 Impact du momentum

Le momentum introduit une moyenne mobile des gradients, ce qui permet de filtrer les fluctuations aléatoires et de renforcer les directions de descente cohérentes au cours du temps. Ainsi, les mises à jour deviennent plus régulières et moins sensibles aux variations locales du gradient. Ce mécanisme réduit les oscillations et favorise une progression plus stable vers une solution optimale. Dans le cadre du gradient stochastique classique, les mises à jour sont fortement influencées par la variance induite par l'échantillonnage aléatoire. Cette variance peut entraîner des oscillations importantes de la trajectoire d'optimisation, particulièrement dans les directions associées à une forte courbure de la fonction objectif, ce qui nuit à la stabilité des itérations.

Dans le cas de l'optimiseur Adam, le momentum est combiné à une normalisation adaptative du pas d'apprentissage via une estimation de la variance des gradients. Cette combinaison améliore encore la stabilité des itérations, en particulier au début de l'entraînement, où les gradients peuvent être de grande amplitude et fortement variables. Les résultats expérimentaux observés confir-

ment que l'introduction du momentum conduit à une convergence plus régulière et plus robuste que les méthodes stochastiques sans momentum.

En résumé, le momentum joue un rôle clé dans la stabilisation des méthodes d'optimisation stochastiques, en réduisant la variance des mises à jour et en atténuant les oscillations, ce qui se traduit par une convergence plus fiable et plus efficace, notamment lors des premières époques d'entraînement.

# 3 Non-lissé, Parcimonie et Proximal

## 3.1 Problème non lisse et opérateur proximal

On considère le problème d'optimisation suivant :

$$\min_{w \in R^d} \ \Phi(w) = f(w) + \lambda \|w\|_1,$$

où $f(w)$ désigne la perte logistique et $\|w\|_1 = \sum_{j=1}^{d} |w_j|$ est la norme $\ell_1$.

La fonction $f$ est lisse et convexe, cependant la norme $\ell_1$ n'est pas différentiable en zéro. En effet, la fonction valeur absolue $|t|$ présente un point anguleux en $t = 0$, ce qui entraîne l'absence de gradient lorsque certaines composantes de $w$ sont nulles. Par conséquent, la fonction objectif $\Phi$ est convexe mais non lisse.

Cette non-lissité empêche l'application directe des méthodes de gradient classiques et motive l'utilisation de méthodes proximales. Celles-ci reposent sur l'opérateur proximal, défini pour une fonction convexe $g$ par :

$$\text{prox}_{\gamma g}(v) = \arg\min_{u \in R^d} \left\{ \frac{1}{2}\|u - v\|^2 + \gamma g(u) \right\},$$

où $\gamma > 0$ est un paramètre de pas.

Cet opérateur annule exactement les composantes de faible amplitude et réduit les autres, ce qui favorise naturellement la parcimonie du vecteur des paramètres. Il constitue l'élément central des méthodes proximales utilisées pour résoudre des problèmes d'optimisation non lisses.

## 3.2 ISTA

Pour résoudre le problème non lisse

$$\min_{w \in R^d} \ \Phi(w) = f(w) + \lambda \|w\|_1,$$

où $f$ est convexe, différentiable et à gradient $L$-Lipschitzien, on utilise l'algorithme ISTA (*Iterative Soft-Thresholding Algorithm*).

Le principe d'ISTA repose sur une itération de type *gradient proximal*, qui combine un pas de gradient sur la partie lisse $f$ et l'application de l'opérateur proximal associé à la norme $\ell_1$. À l'itération $k$, la mise à jour s'écrit :

$$w_{k+1} = \text{prox}_{\alpha\lambda\|\cdot\|_1} \left( w_k - \alpha \nabla f(w_k) \right),$$

où $\alpha \in (0, 1/L]$ est un pas de descente.

Dans le cas de la régularisation $\ell_1$, l'opérateur proximal est donné par l'opérateur de seuil doux, qui agit composante par composante :

$$\big(\mathrm{prox}_{\alpha\lambda\|\cdot\|_1}(v)\big)_j = \mathrm{sign}(v_j)\ \max\big(|v_j| - \alpha\lambda,\, 0\big).$$

### 3.3 FISTA

Afin d'accélérer ISTA, on utilise FISTA (*Fast Iterative Soft-Thresholding Algorithm*), basé sur l'accélération de Nesterov. Pour le problème

$$\min_{w \in R^d}\ \Phi(w) = f(w) + \lambda\|w\|_1,$$

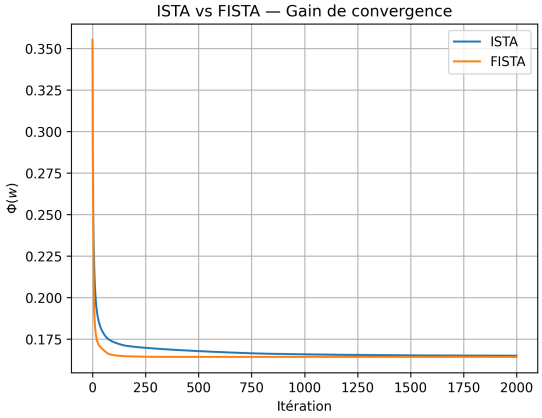
où $f$ est différentiable à gradient $L$-Lipschitzien, FISTA remplace l'itération proximale d'ISTA par une mise à jour effectuée en un point extrapolé $y_k$.

L'algorithme s'écrit :

$$w_{k+1} = \mathrm{prox}_{\alpha\lambda\|\cdot\|_1}\left(y_k - \alpha\nabla f(y_k)\right), \quad t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}, \quad y_{k+1} = w_{k+1} + \frac{t_k - 1}{t_{k+1}}(w_{k+1} - w_k),$$

avec $t_0 = 1$ et $y_0 = w_0$, et un pas $\alpha \in (0, 1/L]$.

Figure 4: Influence de $\lambda$ : évolution de la parcimonie (nombre de coefficients nuls) en fonction de $\lambda$.




La figure 4 illustre l'impact de $\lambda$ sur la parcimonie de la solution. Lorsque $\lambda$ augmente, le seuil doux annule davantage de composantes de $w$, ce qui se traduit par une augmentation du nombre de coefficients nuls. À l'inverse, un $\lambda$ faible réduit la pénalisation et conduit à une solution plus dense.

### 3.4 Influence du paramètre $\lambda$

Le paramètre de régularisation $\lambda$ joue un rôle central dans les problèmes d'optimisation régularisés par la norme $\ell_1$, car il contrôle directement le compromis entre qualité d'ajustement aux données et parcimonie du vecteur des paramètres.

Lorsque $\lambda$ est faible, le terme de régularisation $\lambda\|w\|_1$ a une influence limitée par rapport à la perte logistique. Dans ce cas, l'algorithme cherche principalement à minimiser l'erreur de classification, ce qui conduit à une solution comportant un grand nombre de coefficients non nuls. La parcimonie est alors faible, mais le modèle conserve une grande capacité d'ajustement. Ainsi, l'analyse du nombre de coefficients nuls en fonction de $\lambda$ met en évidence un compromis fondamental : un $\lambda$ trop faible conduit à un modèle dense et potentiellement instable, tandis qu'un $\lambda$ trop élevé impose une parcimonie excessive. À l'inverse, lorsque $\lambda$ augmente, la pénalisation associée à la norme $\ell_1$ devient plus forte. Pour des valeurs très élevées de $\lambda$, la régularisation domine la fonction objectif et peut conduire à une solution quasi nulle, où la majorité des coefficients sont annulés. Bien que cette situation maximise la parcimonie, elle peut dégrader les performances de classification en raison d'un sous-ajustement du modèle.

Le choix de $\lambda$ doit donc être effectué avec soin, afin d'obtenir un équilibre satisfaisant entre interprétabilité du modèle et performance prédictive.