Faculté des Sciences et Techniques

Département de Mathématiques et Informatique

OPTIMISATION EN MACHINE LEARNING

Modélisation, SGD et Méthodes Proximales

Étudiant : Mohamed El Hafed KHATTRI

Master : Statistiques et Sciences des Données (SSD)

Année universitaire : 2025-2026

Encadré par :
Dr. Mohamed Mahmoud EL BENANY

9 janvier 2026

Table des matières

I Fondements et Gradient Déterministe

I.1 Analyse de la fonction objectif

Considérons la fonction de perte logistique régularisée (Ridge) :

$$F(w) = \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i x_i^T w}) + \frac{\lambda}{2}\|w\|^2 \tag{1}$$

où $w \in \mathbb{R}^d$, $x_i \in \mathbb{R}^d$ et $y_i \in \{-1, 1\}$.

Régularité : Décomposons $F(w) = f(w) + r(w)$ où :

— $f(w) = \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i x_i^T w})$ est la perte logistique moyenne

— $r(w) = \frac{\lambda}{2}\|w\|^2$ est le terme de régularisation

Pour chaque i, la fonction $w \mapsto \log(1 + e^{-y_i x_i^T w})$ est une composition de fonctions $\mathcal{C}^\infty$:
— $w \mapsto x_i^T w$ est linéaire donc $\mathcal{C}^\infty$
— $t \mapsto e^t$ est $\mathcal{C}^\infty$
— $t \mapsto \log(1 + t)$ est $\mathcal{C}^\infty$ sur $(0, +\infty)$

Par composition, chaque terme est $\mathcal{C}^\infty$, donc leur somme f est $\mathcal{C}^\infty$. Le terme quadratique r est également $\mathcal{C}^\infty$. Ainsi $F \in \mathcal{C}^2$.

Convexité : Il suffit de montrer que chaque composante est convexe.
Convexité de f : Pour $t \in \mathbb{R}$, soit $\phi(t) = \log(1 + e^{-t})$. Calculons :

$$\phi'(t) = \frac{-e^{-t}}{1 + e^{-t}} = -\sigma(t)$$
$$\phi''(t) = \frac{e^{-t}}{(1 + e^{-t})^2} = \sigma(t)(1 - \sigma(t)) \geq 0$$

où $\sigma(t) = \frac{1}{1+e^{-t}}$ est la fonction sigmoïde. Donc ϕ est convexe.

Pour chaque i, $w \mapsto \phi(y_i x_i^T w)$ est convexe comme la composition d'une fonction convexe et d'une fonction affine. La somme de fonctions convexes étant convexe, f est convexe.

Convexité de r : $r(w) = \frac{\lambda}{2}\|w\|^2$ est convexe car elle est quadratique avec la Hessienne $\lambda I_d \succeq 0$.

Par conséquent, $F = f + r$ est convexe.

Convexité forte : Une fonction deux fois différentiable est λ-fortement convexe si et seulement si :

$$\nabla^2 F(w) \succeq \lambda I_d \quad \forall w$$

(où $A \succeq B$ signifie que $A - B$ est semi-définie positive)

Calculons la Hessienne de F :

Hessienne f :

$$\nabla^2 f(w) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\sigma_i(w)(1 - \sigma_i(w))}_{\text{scalaire} \geq 0} x_i x_i^T$$

où $\sigma_i(w) = \dfrac{1}{1 + e^{y_i x_i^T w}} \in (0, 1)$.

Cette Hessienne est semi-définie positive car c'est une somme de matrices $x_i x_i^T$ (qui sont SDP) avec des coefficients positifs.

Hessienne de r :

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

Hessienne totale :

$$\nabla^2 F(w) = \nabla^2 f(w) + \nabla^2 r(w) = \underbrace{\nabla^2 f(w)}_{\succeq 0} + \lambda I_d \succeq \lambda I_d$$

I.2 Calcul du gradient

Gradient de f :

$$\begin{aligned}
\nabla_w \log(1 + e^{-y_i x_i^T w}) &= \frac{1}{1 + e^{-y_i x_i^T w}} \cdot e^{-y_i x_i^T w} \cdot (-y_i x_i) \\
&= \frac{-y_i x_i e^{-y_i x_i^T w}}{1 + e^{-y_i x_i^T w}} \\
&= -y_i x_i \cdot \frac{1}{1 + e^{y_i x_i^T w}} \\
&= -y_i x_i \sigma(-y_i x_i^T w)
\end{aligned}$$

Gradient de r :

$$\nabla_w \frac{\lambda}{2} \|w\|^2 = \lambda w$$

En sommant, on obtient le résultat annoncé :

$$\nabla F(w) = -\frac{1}{n} \sum_{i=1}^{n} \frac{y_i x_i}{1 + e^{y_i x_i^T w}} + \lambda w = -\frac{1}{n} \sum_{i=1}^{n} y_i x_i \sigma(-y_i x_i^T w) + \lambda w$$

Lipschitz-continuité du gradient

Le gradient est L-Lipschitzien si et seulement si la Hessienne satisfait $\nabla^2 F(w) \preceq L I_d$. Calculons la Hessienne de F :

$$\nabla^2 F(w) = \frac{1}{n} \sum_{i=1}^{n} \frac{e^{y_i x_i^T w}}{(1 + e^{y_i x_i^T w})^2} x_i x_i^T + \lambda I_d$$

Posons $\sigma_i(w) = \sigma(-y_i x_i^T w) = \frac{1}{1+e^{y_i x_i^T w}}$. Alors :

$$\frac{e^{y_i x_i^T w}}{(1 + e^{y_i x_i^T w})^2} = \sigma_i(w)(1 - \sigma_i(w))$$

Cette quantité est maximale quand $\sigma_i = 1/2$, donnant une valeur de $1/4$. Donc :

$$\nabla^2 F(w) \preceq \frac{1}{4n} \sum_{i=1}^{n} x_i x_i^T + \lambda I_d = \frac{1}{4n} X^T X + \lambda I_d$$

Or $\|X^T X\|_2 = \|X\|_2^2$ (norme spectrale). Donc :

$$\nabla^2 F(w) \preceq \left(\frac{1}{4n} \|X\|_2^2 + \lambda \right) I_d$$

D'où $L = \frac{1}{4n} \|X\|_2^2 + \lambda$.

I.3 Implémentation des algorithmes

Cette section présente les performances de l'algorithme de Descente de Gradient (GD) à pas fixe comparativement à la méthode du Gradient Conjugué (CG) pour la minimisation de la perte logistique régularisée par la norme L_2.

L'implémentation de la descente de gradient a été réalisée avec un pas fixe η choisi de manière à assurer la stabilité du schéma itératif. Nous avons comparé cette approche avec l'algorithme du Gradient Conjugué implémenté via `scipy.optimize`.

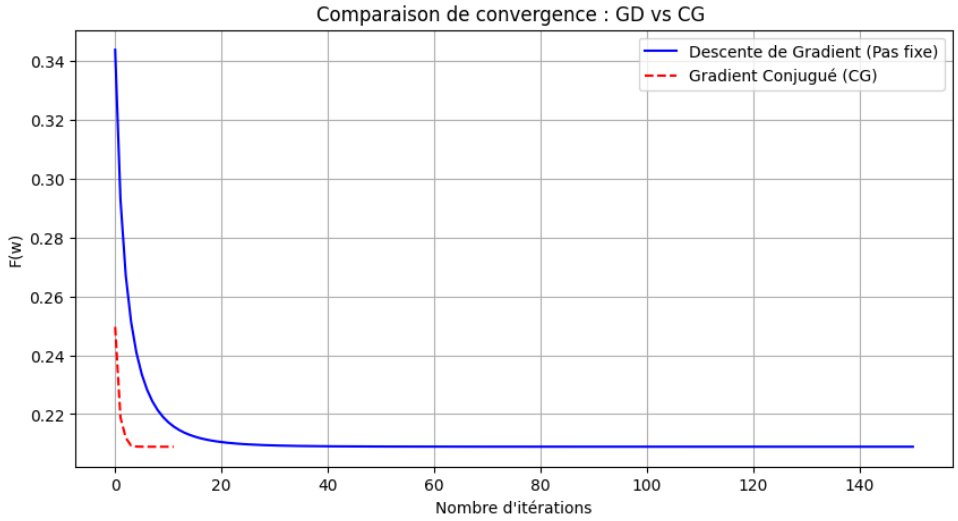

FIGURE 1 – Évolution de la fonction de perte $F(w)$ en fonction du nombre d'itérations.

L'observation des courbes de convergence permet de tirer les conclusions suivantes :

- **Vitesse de convergence :** Le Gradient Conjugué (CG) converge vers le minimum global en nettement moins d'itérations que la descente de gradient classique. Cela s'explique par le fait que le CG génère des directions de descente mutuellement conjuguées, évitant ainsi les oscillations inhérentes à la GD à pas fixe dans les zones de forte courbure.
- **Précision numérique :** Les deux méthodes convergent vers la même valeur finale de la fonction objective, ce qui confirme empiriquement la forte convexité de F démontrée précédemment.
- **Sensibilité au pas :** Alors que la performance de la GD est extrêmement sensible au choix du paramètre η (un pas trop grand provoque la divergence), le CG utilise une recherche de pas optimale (line search) à chaque itération, garantissant une décroissance plus robuste de la perte.
- **Complexité et Limites :** Bien que ces méthodes déterministes soient efficaces pour des jeux de données de taille modérée, le coût de calcul du gradient complet $\nabla F(w)$ devient énorme lorsque n est très grand, justifiant ainsi le passage aux méthodes stochastiques dans la suite de l'étude.

II Passage à l'Échelle Stochastique

II.1 Gradient Stochastique (SGD)

Lorsque n est très grand, calculer $\nabla F(w)$ nécessite $\mathcal{O}(n)$ opérations, ce qui devient très coûteux. Contrairement à la descente de gradient classique qui utilise tous les points pour chaque mise à jour, le SGD n'utilise qu'un seul échantillon tiré aléatoirement. A

chaque itération k, on tire un indice i_k uniformément dans $\{1, \ldots, n\}$ et on effectue :

$$w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k)$$

où $f_i(w) = \log(1 + e^{-y_i x_i^T w}) + \frac{\lambda}{2}\|w\|^2$ est la perte sur l'exemple i.

Le gradient stochastique est :

$$\nabla f_i(w) = -y_i x_i \sigma(-y_i x_i^T w) + \lambda w$$

Algorithm 1 Gradient Stochastique (SGD)

Entrée : $w_0 \in \mathbb{R}^d$, nombre d'époques T, pas initial α_0
for $t = 1$ to T **do**
    Permuter aléatoirement les indices $\{1, \ldots, n\}$
    for $i = 1$ to n **do**
        $k \leftarrow (t-1)n + i$
        $\alpha_k \leftarrow \frac{\alpha_0}{1+\alpha_0 \lambda k}$ (décroissance)
        $w_{k+1} \leftarrow w_k - \alpha_k \nabla f_i(w_k)$
    end for
end for
Retourner w_k

Pour assurer la convergence du SGD, le pas α_k doit satisfaire une règle de décroissance. Une règle classique est celle de Robbins-Monro : $\alpha_k = \frac{\alpha_0}{1+\alpha_0 \lambda k}$ où α_0 est le pas initial et γ le paramètre de décroissance.

La figure 2 compare la convergence de la descente de gradient à pas fixe et du SGD pour $\alpha_0 = 0.2$ et $\gamma = 0.001$.

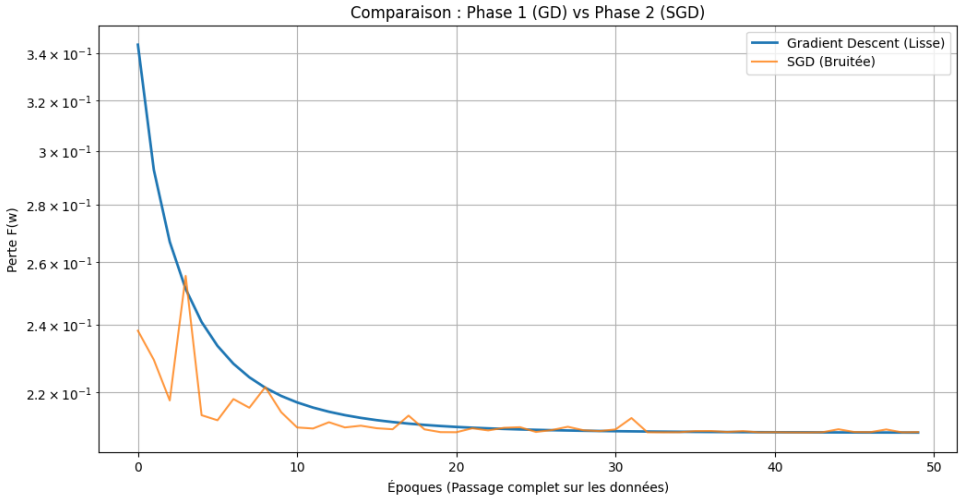

FIGURE 2 – Évolution de la perte $F(w)$: Comparaison entre Gradient Déterministe et SGD.

— **Vitesse par époque :** On observe que le SGD atteint des valeurs de perte très basses dès les premières époques. Puisque le SGD effectue n mises à jour par époque (une par échantillon), il explore l'espace des paramètres beaucoup plus activement que le GD qui n'effectue qu'une seule mise à jour après avoir parcouru l'intégralité du dataset.
— **Bruit de gradient :** La trajectoire du SGD est moins "lisse" que celle du GD.

— **Stabilité :** La règle de décroissance du pas α_k s'avère indispensable. Sans elle, le poids w ne convergerait pas mais oscillerait indéfiniment dans une région autour de l'optimum proportionnelle à la taille du pas fixe.

II.2 Optimiseurs Modernes

RMSProp

RMSProp adapte le pas d'apprentissage pour chaque dimension en utilisant une moyenne mobile des gradients au carré. Il divise le pas par la racine carrée de la moyenne des carrés des gradients. Cela permet de réduire les oscillations sur les variables qui bougent trop vite.

L'avantage de RMSProp est qu'il normalise les gradients, ce qui stabilise l'apprentissage pour des features de différentes échelles.

Adam

Adam (Adaptive Moment Estimation) combine RMSProp avec le momentum. Il garde une trace de la direction moyenne du gradient (le moment) pour accélérer la convergence dans les zones de faible pente.

Nous avons comparé le SGD (avec règle de décroissance), RMSProp et Adam sur les 30 premières époques. Les résultats sont synthétisés dans la figure suivante.

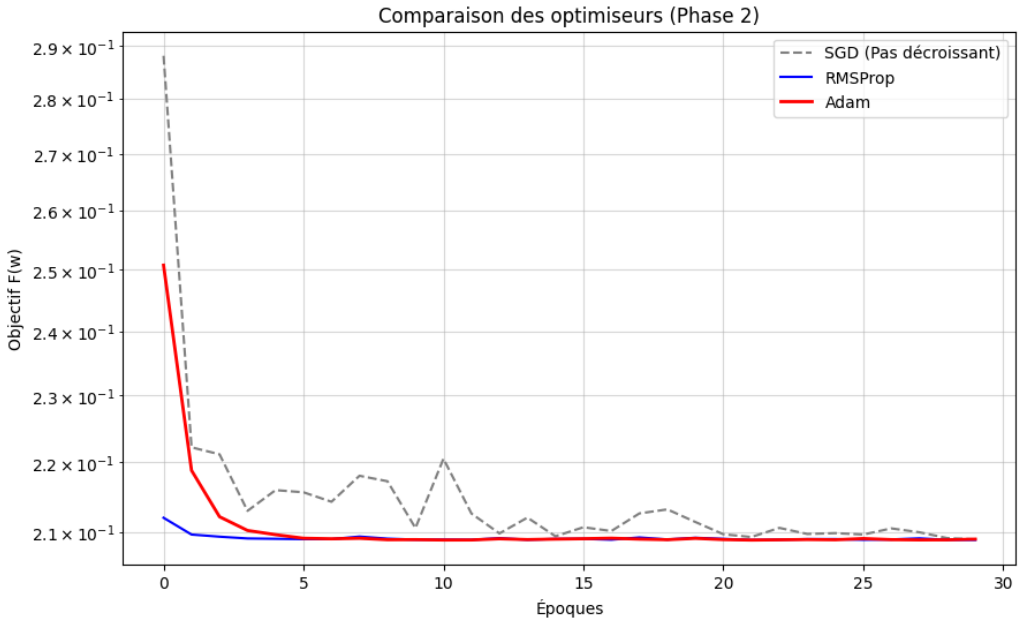

FIGURE 3 – Évolution de l'objectif $F(w)$ pour les différents optimiseurs stochastiques.

L'analyse des courbes de convergence permet de tirer plusieurs remarques :

— **Stabilité :** Alors que le SGD présente une trajectoire "hachée" due au bruit des échantillons individuels, **Adam** produit une courbe nettement plus lisse. Le mécanisme de correction du biais (bias correction) dans Adam est crucial pour stabiliser les premières itérations.

— **Vitesse de convergence :** Adam atteint un plateau de perte plus rapidement que RMSProp. L'adaptation individuelle des pas permet de s'affranchir d'un réglage fastidieux du *learning rate* global, rendant l'optimisation plus robuste aux données de grande dimension (comme dans le cas du jeu de données Breast Cancer).

Les méthodes adaptatives, et particulièrement Adam, s'imposent comme le meilleur compromis entre la vitesse de convergence et la stabilité numérique. Elles constituent une base solide avant d'aborder des problèmes plus complexes où la fonction objective n'est plus dérivable partout.

II.3 Impact du Momentum

Le momentum classique s'écrit :

$$v_{k+1} = \mu v_k - \alpha \nabla f_{i_k}(w_k), \quad w_{k+1} = w_k + v_{k+1}$$

En accumulant les directions passées, le momentum permet à l'algorithme de maintenir une vitesse constante dans les directions pertinentes tout en annulant les composantes de bruit aléatoires. Cela évite les "pics" de perte que l'on observe parfois avec le SGD classique.

III Non-Lissé, Parcimonie et Proximal

III.1 Régularisation L1 et parcimonie

Considérons maintenant le problème avec régularisation L1 :

$$\min_{w \in \mathbb{R}^d} \Phi(w) = f(w) + \lambda \|w\|_1 \tag{2}$$

où $f(w) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + e^{-y_i x_i^T w})$ est la perte logistique.

La norme L1 est définie par $\|w\|_1 = \sum_{j=1}^{d} |w_j|$. La fonction valeur absolue $t \mapsto |t|$ n'est pas différentiable en $t = 0$ car :

$$\lim_{h \to 0^+} \frac{|h| - |0|}{h} = 1 \neq -1 = \lim_{h \to 0^-} \frac{|h| - |0|}{h}$$

Par conséquent, $\|w\|_1$ n'est pas différentiable aux points où certaines coordonnées $w_j = 0$. La fonction objectif Φ hérite de cette non-différentiabilité, bien que f soit lisse.

La régularisation L1 favorise les solutions parcimonieuses (avec de nombreux coefficients nuls). Elle effectue une sélection de variables automatique, utile en haute dimension quand on suppose que seules quelques variables sont pertinentes.

Opérateur proximal

Définition 1. *Pour une fonction convexe $g : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ et $\alpha > 0$, l'opérateur proximal est :*

$$prox_{\alpha g}(v) = \arg\min_{w \in \mathbb{R}^d} \left\{ g(w) + \frac{1}{2\alpha} \|w - v\|^2 \right\}$$

III.2 Algorithmes proximaux

ISTA (Iterative Soft-Thresholding Algorithm)

ISTA applique itérativement un pas de gradient sur f suivi d'un opérateur proximal :

Algorithm 2 ISTA

Entrée : $w_0 \in \mathbb{R}^d$, pas $\alpha \leq 1/L_f$, tolérance ϵ
$k \leftarrow 0$
while critère de convergence **do**
    $v_k \leftarrow w_k - \alpha \nabla f(w_k)$
    $w_{k+1} \leftarrow \text{prox}_{\alpha\lambda\|\cdot\|_1}(v_k)$ (seuil doux)
    $k \leftarrow k + 1$
end while
Retourner w_k

Convergence : ISTA converge avec un taux $\mathcal{O}(1/k)$ pour une fonction objectif convexe.

FISTA (Fast ISTA)

FISTA accélère ISTA via une technique d'extrapolation de Nesterov :

Algorithm 3 FISTA

Entrée : $w_0 = z_0 \in \mathbb{R}^d$, pas $\alpha \leq 1/L_f$, $t_0 = 1$
$k \leftarrow 0$
while critère de convergence **do**
    $v_k \leftarrow z_k - \alpha \nabla f(z_k)$
    $w_{k+1} \leftarrow \text{prox}_{\alpha\lambda\|\cdot\|_1}(v_k)$
    $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$
    $z_{k+1} \leftarrow w_{k+1} + \frac{t_k - 1}{t_{k+1}}(w_{k+1} - w_k)$ (extrapolation)
    $k \leftarrow k + 1$
end while
Retourner w_k

Nous avons implémenté l'algorithme **ISTA** (Iterative Soft-Thresholding Algorithm) et sa version accélérée **FISTA**. La Figure 4 illustre le gain de performance obtenu grâce à l'accélération de Nesterov.

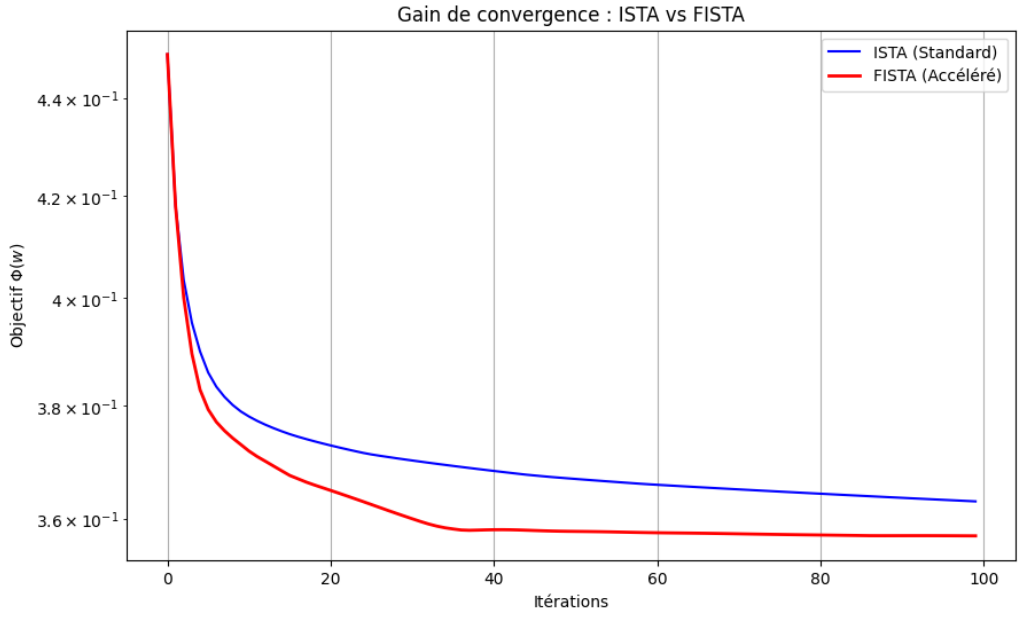

FIGURE 4 – Comparaison de la vitesse de convergence entre ISTA et FISTA.

L'analyse des résultats met en évidence deux points majeurs :

— **Gain de convergence :** FISTA atteint une précision optimale en nettement moins d'itérations qu'ISTA. L'utilisation d'un point intermédiaire de gradient (combinaison linéaire des deux itérés précédents) permet de compenser la lenteur du gradient à proximité du minimum.

— **Sélection de variables :** À l'issue de l'optimisation, nous observons que de nombreux coefficients du vecteur w sont **exactement nuls**. Ce phénomène de parcimonie simplifie l'interprétation du modèle en identifiant les caractéristiques biologiques (radius, texture, etc.) les plus prédictives du caractère malin des tumeurs.

III.3 Analyse de la parcimonie

L'un des objectifs majeurs de l'utilisation de la norme L_1 est la réduction de la complexité du modèle. La Figure 5 illustre la relation entre l'intensité de la régularisation λ et le nombre de variables conservées par l'algorithme FISTA.

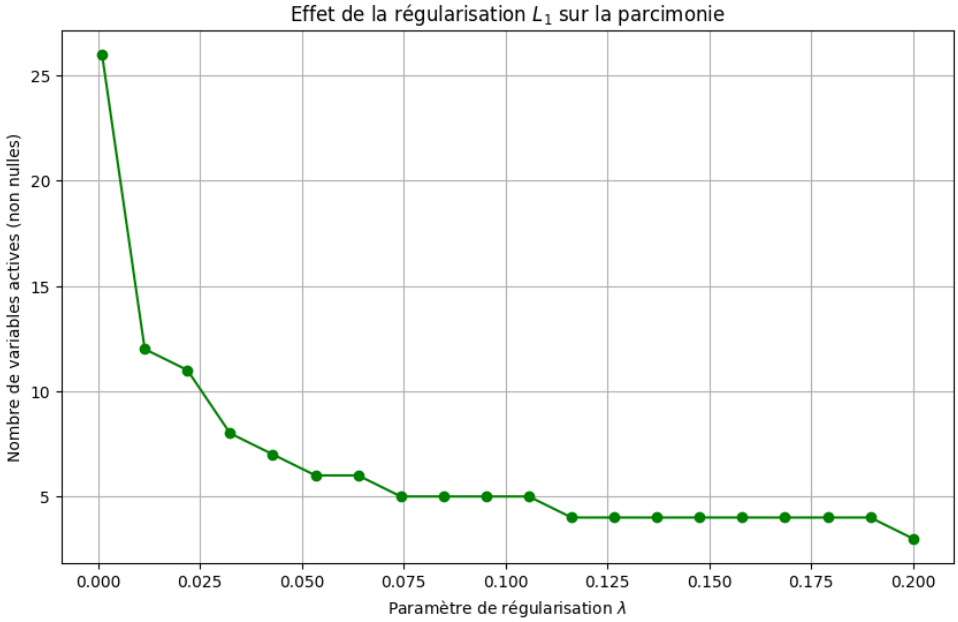

FIGURE 5 – Nombre de coefficients non nuls en fonction de λ.

On observe qu'à mesure que λ augmente, le nombre de variables actives diminue drastiquement. Pour une valeur de $\lambda \approx 0.1$, le modèle ne conserve qu'une fraction des caractéristiques initiales (environ 10 sur 30). Cela démontre que seules quelques mesures (comme le *radius worst* ou les *concave points*) sont réellement déterminantes pour diagnostiquer une tumeur. Cette propriété de parcimonie est essentielle en milieu médical : elle permet aux praticiens de se concentrer sur un nombre restreint de biomarqueurs critiques, réduisant ainsi le coût et la complexité des examens tout en maintenant une précision élevée.

Conclusion

Ce projet nous a permis d'explorer l'ensemble du spectre de l'optimisation numérique appliquée à l'apprentissage automatique, depuis les fondements déterministes jusqu'aux méthodes stochastiques et proximales de pointe.

L'étude a démontré que le choix de l'optimiseur dépend étroitement de la structure du problème et de la taille des données :

- Les méthodes de **Gradient Conjugué** (Phase 1) offrent une précision inégalée pour des jeux de données de taille modérée mais deviennent inefficaces face à la montée en charge.
- Les approches stochastiques comme **Adam** (Phase 2) se révèlent indispensables pour le traitement de gros volumes de données, offrant un excellent compromis entre stabilité et rapidité grâce à l'adaptation dynamique du pas d'apprentissage.
- Enfin, les méthodes proximales comme **FISTA** (Phase 3) ont montré leur supériorité pour traiter les fonctions non-lisses, permettant d'allier performance prédictive et parcimonie du modèle.