Notes sur la régularisation

Maximilien

July 2020

Notions MIASHS fonctions, normes, espaces vectoriels, dérivation, sous-dérivée, différentielle, sous-différentielle, optimisation sous contrainte, Lagrangien.

1 Introduction

L'apprentissage automatique cherche à trouver des fonctions $h : \mathcal{X} \mapsto \mathcal{Y}$ où $\mathcal{X}$ serait par exemple l'ensemble des photos de chiens et de chats et $\mathcal{Y}$ les deux étiquettes "chiens" et "chats". Trouver de telles fonctions se fait via une procédure d'apprentissage.

Il n'est pas possible de construire de manière absolue ce qu'est "un chien" ou "un chat" et il convient de sélectionner notre fonction h en nous appuyant sur des données observées. Cependant, la fonction h doit conserver certaines propriétés (e.g. peu d'erreurs de classification) sur des données futures. Cette dichotomie est la source de tous les problèmes en *machine learning* : on cherche h sur des données *passées* pour prédire le *future*. Seulement, une bonne fonction pourrait tout simplement "apprendre par cœur" les données passées pour être considérées comme une bonne fonction.

Intuitivement. Comme cela a été vu dans les notes sur le sur-apprentissage ou la dimension de Vapnik Chervonenski, plus on considèrera un grand ensemble de fonctions, plus le risque qu'il existe une fonction capable de mémoriser, au moins une partie du jeu de données. Une stratégie permettant de limiter le sur-apprentissage consiste donc à considérer "moins de fonctions". Malheureusement, cela peut être contraignant. Une autre stratégie s'appuie sur la notion de régularisation. Ici, on considère l'ensemble de fonctions qui nous intéresse, mais on pénalise chacune d'entre elle par une mesure de "régularité" et le résultat de la procédure d'optimisation sera le meilleur compromis entre "erreur" sur le jeu d'apprentissage et "régularité" (e.g. dérivés majorées par une constante K).

La régression polynomiale (i.e. linéaire) permet de visualiser ces effets. En effet, on peut voir dans la figure **??** différentes interpolations polynomiales, régularisées ainsi que le développement de Taylor de la fonction. On voit bien qu'en

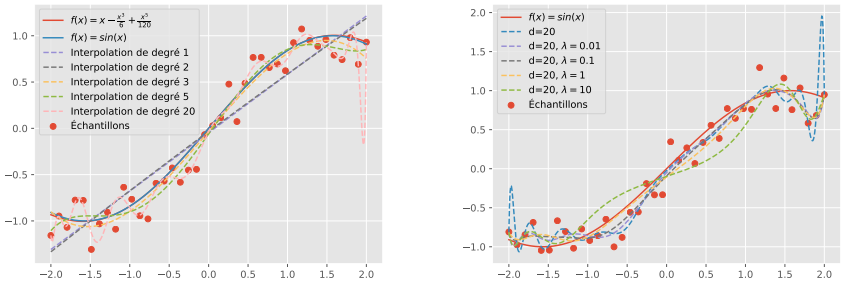

FIGURE 1 – Interpolation polynomiale, série de Taylor, et régularisation.

connaissant les "bon coefficients" du polynôme, on arrive à une assez bonne approximation de la vraie fonction. Les autres scénarios s'adaptent plus ou moins au bruit des données et fournissent des résultats plus ou moins caricaturaux. La régularisation permet de limiter ces effets tout en considérant le plus grand ensemble de fonctions.

2 Formalisation de l'apprentissage automatique

Soit deux variables aléatoires $X \in \mathcal{X}$ et $Y \in \mathcal{Y}$ où on pourrait par exemple voir $\mathcal{X}$ comme un ensemble de photos et $\mathcal{Y}$ un ensemble d'étiquettes sur ces photos (e.g. chien et chat).

On supposera $X \times Y \sim p$, où p ne serait pas connue. Notre objectif est de trouver une fonction f telle qu'on arrive prédire Y à partir de X en minimisant une certaine notion d'erreur.

Définition de l'erreur. La première étape consiste à définir l'erreur pour une réalisation. Soit $X \times Y \sim p$, une erreur élémentaire peut-être vue de deux manières différentes :

— $r : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$
— $r : \mathcal{Y} \times \mathbb{P}_{\mathcal{Y}} \mapsto \mathbb{R}^+$

Dans le premier cas, on peut voir l'erreur comme une "distance" entre deux éléments de $\mathcal{Y}$ et dans le second comme une "mesure" de l'adéquation entre une mesure de probabilité sur $\mathcal{Y}$ et un élément de $\mathcal{Y}$. Les moindres carrés sont une forme d'erreur où l'écart entre deux éléments de $\mathcal{Y} = \mathbb{R}$ est calculé comme l'écart quadratique de leur valeur :

$$\begin{aligned} r : \mathcal{Y} \times \mathcal{Y} &\mapsto \mathbb{R}^+ \\ \hat{y}, y &\to (\hat{y} - y)^2 \end{aligned}$$

De manière similaire et tout aussi standard, dans le cadre d'un problème de

classification, nous avons l'entropie relative. Ici, $\mathcal{Y} = \{i\}_{i \leq K}$.

$$\begin{aligned} r : \mathbb{P}_{\mathcal{Y}} \times \mathcal{Y} &\mapsto \mathbb{R}^+ \\ \hat{y}, y &\to -\sum_k \delta_{k=y} \ln\big(\hat{y_k}\big), \end{aligned}$$

où $\hat{y}$ est associé à un point du K-simplexe interprété de la manière suivante : $\mathbb{P}(y = k|x) = \hat{y}(x)_k$.

L'objectif du *machine learning* est d'avoir les meilleures performances en espérance relativement à ces erreurs. On parlera de risque de généralisation lorsqu'on fera référence à l'erreur en espérance pour une fonction $h : \mathcal{X} \mapsto \mathcal{Y}/\mathbb{P}_{\mathcal{Y}}$:

$$R(h) = \mathbb{E}_{X \times Y \sim p}\big[r(h(X), Y)\big].$$

Soit $\mathcal{H}$ un ensemble de fonctions, l'objective du *machine learning* est de trouver la fonction f de $\mathcal{H}$ suivante :

$$f = \underset{h \in \mathcal{H}}{\operatorname{argmin}} R(h).$$

Ce problème n'est malheureusement pas résoluble, ne serait-ce que parce que nous ne connaissons pas la loi p.

La stratégie mise en place en pratique et appellée *minimisation du risque empirique* (ERM) consiste à échantillonner selon p (i.e. collecter des données) $\mathcal{D} = \{(x_i, y_i)\}_{i \leq N} \sim p^N$ (chaque couple est i.i.d.). Ainsi, le risque empirique suivant est un estimateur sans biais du risque de généralisation :

$$Re(h) = \frac{1}{N} \sum_{i=1}^{N} r(h(x_i), y_i),$$

et l'objectif se reformule de la manière suivante :

$$\hat{f} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} Re(h).$$

La quantité $Re(\hat{f})$ n'est quant à elle plus sans biais. On sait qu'on va favoriser les fonctions particulièrement adaptées aux fluctuations aléatoires de notre jeu de données. La question soulevée ici est celle du *gap de généralisation*, c'est-à -dire :

$$\Big|Re(\hat{f}) - R(\hat{f})\Big|.$$

Il semble intuitif que plus l'ensemble de fonctions considéré $\mathcal{H}$ sera grand, plus on risque de trouver une fonction arbitrairement "forte" sur $\mathcal{D}$ mais mauvaise en espérance. L'exemple paradigmatique de cette idée est la fonction mémoire :

$$f(x) = \begin{cases} y, & \text{si } (x, y) \in \mathcal{D} \\ \text{random}(\mathcal{Y}), & \text{sinon} \end{cases}$$

où random($\mathcal{Y}$) retourne un élément choisi aléatoirement de $\mathcal{Y}$. On voit bien que la fonction f ainsi construite ne nous intéresse pas.

La suite de ces notes se concentrera sur les cas où $|\mathcal{H}| = \infty$. On essaiera en particulier d'établir quelques majorants de généralisation relativement à la taille de l'ensemble de fonctions et à celle du jeu de données.

3 Régularisation

3.1 Introduction

On constate assez intuitivement qu'une fonction capable de mémoriser tout un jeu de données oscillera énormément. Une stratégie consiste ainsi à contraindre cette oscillation au travers de la norme du gradient $\left\|\nabla \hat{f}(x)\right\|$. Dans le cadre d'un modèle linéaire

$$f(x) = g(\langle \beta, x \rangle),$$

cette norme est directement liée à la norme du vecteur de paramètres $\|\beta\|$. Dit autrement, on souhaite résoudre le problème d'optimisation suivant :

$$\hat{f} = \underset{\beta \in \mathbb{R}^d}{\operatorname{argmin}} R(\beta), \ s.t. \ \kappa(\beta) \leq C$$

pour une constante C fixée et où κ est une fonction quelconque, e.g. $\kappa(\beta) = \|\beta\|_1$. Le Lagrangien associé à ce problème est donc :

$$\mathcal{L}(\beta, \lambda) = R(\beta) + \lambda(\kappa(\beta) - C).$$

En pratique, nous ne savons pas fixer C. Pour cela, la stratégie est la suivante. Le jeu de données $\mathcal{D}$ est découpé en une partie dite "d'apprentissage" et une autre de "validation". L'ensemble de validation sera "quasiment" sans biais et permettra d'évaluer la qualité de notre choix pour C. On définit donc *a priori* une liste de valeurs possibles à tester pour C (ce problème étant éventuellement convexe, un approche ressemblant à une descente de gradient peut être adoptée), et on regarde les performances de notre modèle sur l'ensemble de validation. Enfin, il suffit de sélectionner la valeur ayant obtenu le risque le plus faible (ici le risque est quasiment sans biais [1]) et de réoptimiser un modèle sur toutes les données cette fois-ci pour la valeur de C choisie.

On remarque que quitte à choisir C empiriquement, les paramètres λ et C deviennent redondants et il est possible de reformuler le problème de la manière suivante :

$$\hat{f} = \underset{\beta \in \mathbb{R}^d}{\operatorname{argmin}} R(\beta) + \lambda\kappa(\beta).$$

C'est ce qu'on appelle un problème d'apprentissage régularisé. On retombe sur la formulation non régularisée lorsque $\lambda = 0$.

1. On peut étudier théoriquement le biais d'un tel risque en considérant l'ensemble des fonctions associées à une valeur de C comme un ensemble fini de modèles.

Notons que la régularisation est rigoureusement fondée sur deux principes que nous n'approfondirons pas ici. Tout d'abord, on peut démontrer que le gap de généralisation sera petit avec forte probabilité si $\left\|\beta_0 - \hat{\beta}\right\|$ où β_0 est choisi arbitrairement. Autrement dit, si le résultat de l'optimisation n'est pas "très loin" d'un point de départ arbitraire (qui ne dépend pas des données), alors le gap de généralisation sera faible. Ensuite, les principes de parcimonie nous disent que plus la théorie sera simple, plus celle-ci aura de bonnes performances prédictives. Ainsi, on fixe $\beta_0 = 0$ et on retrouve nos normalisations ℓ_1 et ℓ_2.

3.2 Les normes de régularisation et leur effet

Des choix courants sont $\kappa(\beta) = \|\beta\|_1$ et $\kappa(\beta) = \|\beta\|_2^2$. On remarque cependant que la norme ℓ_1 entraà®ne des problèmes de différentiabilité.

Bien qu'en dimension finie toutes les normes soient équivalentes (i.e. même topologie induite) [2], leur "géométrie" n'est pas strictement équivalente et cela a un effet majeur en *machine learning*.

Reprenons notamment les normes suivantes :

$$\ell_1(x) = \sum_i |x_i|$$

$$\ell_2(x) = \sqrt{\sum x_i^2}.$$

et étudions la forme que revêt notre contrainte selon la norme choisie.

La figure **??** illustre l'effet de la norme. Il s'agit d'un problème d'optimisation quadratique dont le minimum est atteint en $\beta = [0.25, 1.5]^T$. Lorsqu'une contrainte est ajouté, la solution choisie est telle que la ligne de niveau correspondant est tangente à la ligne de contrainte. On remarque que pour la norme ℓ_1, il s'agit d'une solution *sparse* correspondant au paramètre $\beta = [0, 1]^T$. À l'inverse, lorsque la norme est ℓ_2, le paramètre obtenu est $\beta = [0.24, 1]^T$. En pratique, d'un point de vue des performances prédictives, la norme ℓ_2 fonctionne mieux. D'un point de vue de l'interprétation des résultats, la norme ℓ_1 est préférable.

Notons également que plus la dimension est grande, plus l'hypercube définit pas notre contrainte de régularisation ℓ_1 possédera d'angle et permettra donc à la solution d'être *sparse*.

3.3 Zoom sur le cas ℓ_2

La norme ℓ_2 permet d'accroà®tre les performances prédictives de notre modèle. Elle n'offre de plus aucune difficulté calculatoire supplémentaire. Dans le cadre des régressions linéaires ou logistique, la fonction devient même strictement convexe et coercive !

2. Pour peu qu'on soit sur $\mathbb{R}$ ou $\mathbb{C}$. La démonstration s'appuyant sur un argument de compacité ne s'applique pas à un ev sur $\mathbb{Q}$. Bref, hors sujet.

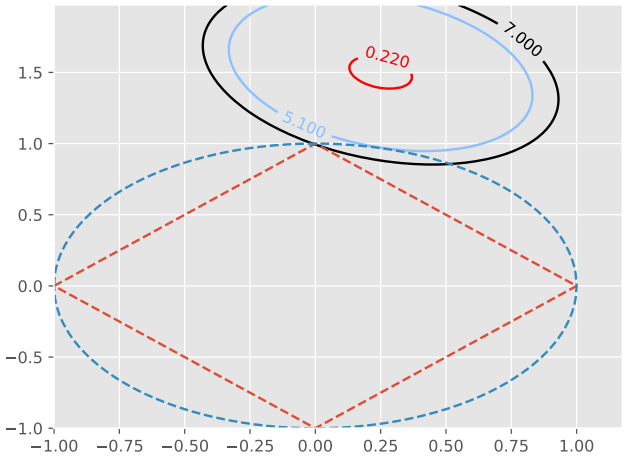

FIGURE 2 – Un problème d'optimisation quadratique, et les contraintes $\|\beta\|_1 = 1$ ou $\|\beta\|_2 = 1$.

Prenons le cas de la régression linéaire avec pénalité ℓ_2 (appelée Ridge). On cherche à optimiser :

$$\hat{f} = \underset{\beta \in \mathbb{R}^d}{\operatorname{argmin}} \, Re(\beta) + \lambda \|\beta\|_2^2, \ \lambda \geq 0$$

En annulant le gradient, on retrouve ce qui s'apparente aux équations normales :

$$X^T X\beta + \lambda\beta = X^T Y$$
$$\Leftrightarrow (X^T X + \lambda Id)\beta = X^T Y$$

où Id est la matrice identité. $X^T X$ est au moins semi-définie positive et λId est définie positive si $\lambda > 0$. Ainsi $(X^T X + \lambda Id)$ est définie positive à tous les coups si $\lambda > 0$. On peut donc l'inverser :

$$\beta = (X^T X + \lambda Id)^{-1} X^T Y.$$

3.4 Zoom sur le cas ℓ_1

La norme ℓ_1 offre un avantage certain pour ce qui est de l'interprétation des résultats. Cependant, autant la norme ℓ_2 n'offre pas de soucis particulier à optimiser autant la ℓ_1 est plus complexe. En effet, la fonction $\|\beta\|_1$ n'est pas différentiable à partir du moment où une unique dimension du vecteur β est à 0. Heureusement, il existe des stratégies permettant de contourner ce genre de problème.

Un premier outil est celui des sous-différentielles.

Définition 1 (Sous-dérivée)**.** *Soit un intervalle $I \subset \mathbb{R}$ et une fonction $f : I \mapsto \mathbb{R}$, convexe sur I. On appelle une sous-dérivée de f en un point x_0 un réel $c \in \mathbb{R}$ tel que $\forall x \in I$, on a :*

$$f(x) - f(x_0) \geq c(x - x_0).$$

Autrement dit, la fonction f est au-dessus de la droite $c(x - x_0) + f(x_0)$.

Proposition 1. *Soit les limites :*

$$a = \lim_{x \to x_0^+} \frac{f(x) - f(x_0)}{x - x_0}$$

$$b = \lim_{x \to x_0^-} \frac{f(x) - f(x_0)}{x - x_0}$$

L'ensemble de toutes les sous-dérivées est compris dans l'intervalle fermé $[a, b]$. Si cet ensemble se résume à un unique élément (i.e. $a = b$), alors il s'agit de la dérivée usuelle.

Définition 2 (sous-différentielle)**.** *On appelle sous-différentielle d'une fonction convexe $f : I \mapsto \mathbb{R}$ au point x_0 l'ensemble de tous les réels compris dans l'intervalle $[a, b]$. On note $\partial f(x_0)$ cet ensemble.*

Ainsi, si l'ensemble contient un unique élément, il s'agit de la dérivée, et on a donc : $\partial f(x_0) = \frac{\partial f}{\partial x}(x_0)$.

La norme ℓ_1 est définie à partir de la notion de valeur absolue. Illustrons les notions précedentes pour la fonction $g(x) = |x|$. On remarque que cette fonction est dérivable presque partout. En fait, le seul endroit où elle ne l'est pas est $x = 0$. La fonction g est convexe sur $\mathbb{R}$ et on observe que lorsque $x = 0$, il existe plusieurs valeurs de sous-dérivée possibles. En particulier, nous avons :

$$\partial g(x) = \begin{cases} -1 & \text{si } x < 0 \\ [-1, +1] & \text{si } x = 0 \\ +1 & \text{si } x > 0 \end{cases}$$

Proposition 2. *Soit une fonction convexe sur I de $f : I \mapsto \mathbb{R}$ et soit $\partial f(x_0)$ sa sous-différentielle en x_0. x_0 est un minimum sur I de la fonction si et seulement si $0 \in \partial f(x_0)$.*

Démonstration. ($\Rightarrow$) Si x_0 est un minimum sur I, alors $\forall y \in I$:

$$f(y) \geq f(x_0) \Leftrightarrow f(y) - f(x_0) \geq 0$$

Si $y < x$ (resp. $y > x$), alors :

$$\frac{f(y) - f(x)}{y - x} \leq 0 \text{ (resp. } \geq 0)$$

$\forall y \in I$ et donc, a fortiori, pour $y \to x_0^{+/-}$. Ainsi, $\partial f(x) = [a, b]$ avec $a \leq 0$ et $b \geq 0 \Leftrightarrow 0 \in \partial f(x)$.

($\Leftarrow$)Si $0 \in \partial f(x_0)$, alors $f(y) - f(x_0) \geq 0 \; \forall y \in I$ et x_0 est donc un minimum sur I. □

De manière similaire, la notion s'étend à toute fonction définie sur un espace Euclidien ($\mathbb{R}^n$).

Définition 3 (Sous-gradient)**.** *Soit U une partie convexe de $\mathbb{R}^n$ et $f : U \mapsto \mathbb{R}$ une fonction convexe sur U.*

$v \in \mathbb{R}^n$ est un sous gradient en $x_0 \in U$ si on a $\forall x \in U$:

$$f(x) - f(x_0) \geq \langle v, x - x_0 \rangle.$$

De la même manière que précédemment, on peut définir la sous différentielle de notre fonction en un point x_0.

Proposition 3. *Soit U une partie convexe de $\mathbb{R}^n$ et f, g deux fonctions convexes de U dans $\mathbb{R}$. On a la propriété suivante :*

$$\partial(g + f) = \partial g + \partial f$$

où la somme à droite est celle de Minkowski[3]*. Cette propriété est connue sous le nom du théorème de Moreau-Rockafellar.*

Proposition 4. *Soit U une partie convexe de $\mathbb{R}^n$, f une fonction convexe de U dans $\mathbb{R}$ et $\lambda \in \mathbb{R}$. On a :*

$$\partial \lambda f = \lambda \partial f.$$

Reprenons notre problème initial. Dans le cadre d'une régression linéaire avec pénalité lasso (ℓ_1), l'objectif est de minimiser la fonction suivante :

$$\|X\beta - y\|_2^2 + \lambda \|\beta\|_1 .$$

Ainsi, comme indiqué précédemment, β est un minimum de cette fonction si et seulement si :

$$0 \in \partial(\|X\beta - Y\|_2^2 + \lambda \|\beta\|_1) = 2X^T X\beta - 2X^T Y + \lambda \partial \|\beta\|_1$$

Cela revient à dire qu'il existe $v \in \partial \|\beta\|_1$ tel qu'on ait :

$$2X^T X\beta - 2X^T Y = -\lambda v$$

Comme indiqué précédemment, pour $v \in \partial \|\beta\|_1$, on a :

$$v_i \in \begin{cases} \{-1\} & \text{si } \beta_i < 0 \\ [-1, +1] & \text{si } \beta_i = 0 \\ \{+1\} & \text{si } \beta_i > 0 \end{cases}$$

En notant X_i la i^{eme} colonne de X, on peut ainsi constater la propriété suivante :

$$\begin{cases} 2X_i^T X\beta - 2X_i^T Y = -\lambda \text{sign}(\beta_i) & \text{si } \beta_i \neq 0 \\ \left|2X_i^T X\beta - 2X_i^T Y\right| \leq \lambda & \text{si } \beta_i = 0 \end{cases}$$

3. Si A et B sont deux ensembles, $A + B = \{a + b | a \in A, b \in B\}$.

Cette dernière doit être satisfaite par une solution du problème mais ne permet pas de déduire une solution en forme close.

L'algorithme de descente de coordonnées permet de définir une suite d'opérations où l'optimisation est faite paramètre par paramètre. Reprenons tout d'abord le Lagrangien :

$$\mathcal{L}(\beta, \lambda) = R(\beta) + P(\lambda, \beta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \beta^T x_i)^2 + \lambda \sum_j |\beta_j|$$

où le $1/2$ a été ajouté afin de simplifier les notations après différentiation mais n'influe pas le résultat de l'optimisation. Concentrons-nous sur la première partie $R(\beta)$:

$$\begin{aligned}\frac{\partial R}{\partial \beta_j} &= -\sum_i (y_i - \beta^T x_i) x_{ij} \\ &= -\sum_i \left[x_{ij}(y_i - \sum_{k \neq j} \beta_k x_{ik})\right] + \beta_j \sum_i x_{ij}^2 \\ &\triangleq -\rho_j + \beta_j z_j\end{aligned}$$

Pour ce qui est de la régularisation, nous avons :

$$P(\beta, \lambda) = \lambda \sum_{k \neq j} |\beta_j| + \lambda |\beta_j|$$

Calculons sa sous-différentielle :

$$\partial_j P(\lambda, \beta) = \partial \lambda |\beta_j| = \lambda \partial |\beta_j| \begin{cases} \{-\lambda\} & \text{si } \beta_j < 0 \\ [-\lambda, +\lambda] & \text{si } \beta_j = 0 \\ \{+\lambda\} & \text{si } \beta_i > 0 \end{cases}$$

Enfin, cherchons un minimum pour β_j :

$$0 \in \partial_j \mathcal{L}(\beta, \lambda) \Leftrightarrow 0 \in \begin{cases} \{-\rho_j + \beta_j z_j - \lambda\} & \text{si } \beta_j < 0 \\ [-\rho_j - \lambda, -\rho_j + \lambda] & \text{si } \beta_j = 0 \\ \{-\rho_j + \beta_j z_j + \lambda\} & \text{si } \beta_i > 0 \end{cases}$$

On remarque dans la seconde condition que la seule manière qu'ait l'intervalle de contenir 0 est d'avoir :

$$-\lambda \leq \rho_j \leq \lambda.$$

Concernant la première condition (à un signe près pour la troisième) nous avons :

$$-\rho_j + \beta_j z_j = \lambda \Leftrightarrow \beta_j = \frac{\lambda + \rho_j}{z_j}.$$

Dit autrement, le minimum de β_j est atteint de la manière suivante :

$$\beta_j = \begin{cases} \frac{\lambda + \rho_j}{z_j} & \text{si } \rho_j < -\lambda \\ 0 & \text{si } -\lambda \leq \rho_j \leq \lambda \\ \frac{-\lambda + \rho_j}{z_j} & \text{si } \rho_j > \lambda \end{cases}$$

où on reconnaà®t $\frac{\text{sign}(\rho_j)}{z_j}(|\rho_j| - \lambda)_+$ [4].

L'algorithme suivant illustre la descende de coordonnées pour Lasso.

Algorithm 1: Descente de coordonnées pour Lasso

```
input  : X, λ > 0, β^(0)
output: β̂
t ← 0 ;
while not stop_condition do
    t ← t + 1 ;
    for j = 1 to d do
        compute z_j ;
        compute ρ_j ;
        update β̂_j ;
    end
end
```

En fonction des valeurs de λ, plus ou moins de dimensions de β seront à 0. Plus λ sera grand, plus le nombre de dimensions à 0 le sera également. La figure **??** illustre les "chemins" suivis par chaque dimension en fonction d'une valeur de λ (α dans `scikit-learn`).

4 Interprétation Bayésienne de la régularisation

Cette section concernant autant la pénalité ℓ_1 que ℓ_2 mais est plus simple d'un point de vue calculatoire pour la seconde. Cette dernière sera donc l'objet de notre attention.

Soit un ensemble d'observations $\{(x_i, y_i)\}_{i \leq n}$ telles que $\forall i \leq n, x_i \in \mathbb{R}^d, y_i \in \mathbb{R}$ et étant donnée un vecteur $\beta \in \mathbb{R}^d$:

$$y_i = \langle \beta, x_i \rangle + \epsilon_i,$$

soit un vecteur i.i.d. avec $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

Le cadre Bayésien suppose que l'espace des paramètres est lui-même probabilisé et muni d'une mesure de probabilité dite *a priori* :

$$\beta \sim \mathcal{N}(\mathbf{0}, \Lambda^{-1}),$$

où Λ est la matrice de précision. Cette loi indique notre confiance dans les paramètres possibles avant d'avoir vu des données. L'objectif du framework Bayésien va être de mettre à jour cette loi après avoir observé des données. Plus la quantité des données sera importante, plus la loi sera proche de ces dernières. À l'inverse, plus elle sera faible, plus ce sera notre a priori qui gouvernera les prédictions. On a choisi de mettre l'espérance des paramètres à 0 pour les raisons précitées dans ce document. Le choix d'un *a priori* normal permet de

4. $(\cdot)_+ = \max(0; \cdot)$.

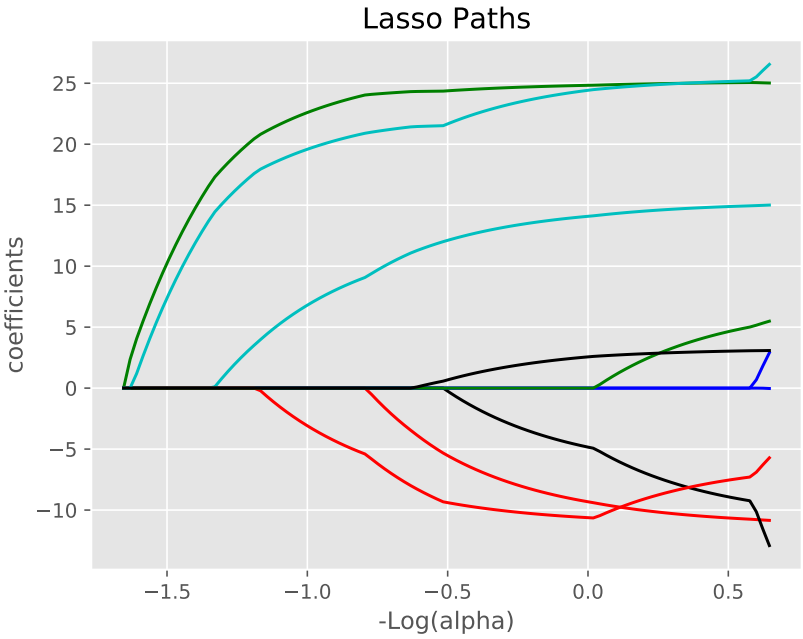

FIGURE 3 – Les abscisses correspondent à une valeur du paramètre de régularisation (valeur élevée à gauche, et proche de 0 à droite). Les ordonnées indiquent la valeur des différentes dimension de notre vecteur de paramètre pour un niveau de régularisation. Le fait que beaucoup de valeurs soient à 0 illustrent de l'effet sparse de Lasso.

retomber sur la régularisation ℓ_2. Le choix d'une loi de Laplace aurait donné le cas ℓ_1.

Nous avons de plus que la loi de $y_i|\beta, x_i$ est donnée de la manière suivante :

$$y_i|\beta, x_i \sim \mathcal{N}(\beta^T x_i, \sigma^2)$$

Nous aurions pu traiter σ^2 de la même manière que β et lui associer une loi *a priori*.

Nous souhaitons à partir d'une nouvelle donnée x prédire la valeur y qui lui serait associée. Pour cela, nous devons dans un premier temps déterminer le vecteur β.

La "loi de Baye" nous donne la solution suivante :

$$p(\beta|X, Y; \Lambda, \sigma^2) \propto \prod_i \Big[p(y_i|x_i, \beta; \sigma^2)\Big] p(\beta; \Lambda)$$

Sans rentrer dans les détails du calcul, cela nous permet maintenant de

formuler une prédiction pour une nouvelle observation :

$$p(y|x, X, Y; \sigma^2, \Lambda) = \int p(y|\beta)p(\beta|X, Y; \Lambda, \sigma^2)d\beta.$$

On obtient une prédiction sur y en marginalisant.

Par chance, cette intégrale est calculable à la main est nous donne :

$$y|x, X, Y \sim \mathcal{N}(\hat{\beta}^T x, \sigma^2 + x^T \Sigma x)$$

avec :

$$\hat{\beta} = (X^T X + \Lambda)^{-1} X^T Y \text{ et } \Sigma = \sigma^2 (X^T X + \Lambda)^{-1}.$$

En fixant $\Lambda = \lambda Id$, on constate que l'espérance de $y|x, X, Y$ est exactement la solution analytique que nous avions déduit de la régression Ridge (i.e. avec pénalité ℓ_2). On dit que Ridge est le maximum a posteriori de cette formulation Bayésienne.