Introduction aux problèmes d'optimisation différentiable

Guillaume Lecué[1]

Dans cette section, on commence par présenter quelques problèmes d'optimisation (avec ou sans contrainte) issus de problèmes réels rencontrés en statistiques, traitement du signal, machine learning, etc.. Ces problèmes permettent d'inscrire le cours d'optimisation différentiable comme un outil pour les autres cours de l'ENSAE.

On présente ensuite une représentation géométrique des problèmes d'optimisation différentiable qui permettent de soulever les points clefs qui permettront de résoudre ces problèmes d'un point de vue mathématiques. En particulier, on insistera sur l'aspect "local" d'un problème d'optimisation qui demande une description locale d'une fonction et d'un ensemble (la contrainte). Ceci justifie l'adjectif "différentiable" dans le titre du cours car sous l'hypothèse de différentiation, décrire localement une fonction et l'ensemble des contraintes est plus facile.

Les deux points qui consistent à décrire localement une fonction et une contrainte dressent naturellement le plan de la première partie du cours qui concerne la résolution mathématique des problèmes d'optimisation. Dans la deuxième partie du cours, on intéressera à la résolution algorithmique de problèmes d'optimisation.

1 Quelques exemples de problèmes d'optimisation

L'optimisation est une branche des mathématiques cherchant à résoudre des problèmes de la forme

$$\min\left(f(x) : x \in K\right) \tag{1.1}$$

où $f : E \to \mathbb{R}$ et $K \subset E$. Dans ce cours, on prendra $E = \mathbb{R}^d$ ou, plus généralement, un ouvert U de $\mathbb{R}^d$ (car on définit plus facilement la différentiation sur un ouvert). Les questions naturellement associées à (1.1) sont par exemple :

- un minimum existe-t'il ?
- le minimum est-il unique ?
- comment caractériser le minimum par des conditions nécessaires et suffisantes ou seulement nécessaires ?
- est-il possible de calculer ou d'approcher une solution au problème (1.1) de manière efficace à l'aide d'un algorithme ?

Les trois premières questions sont des questions mathématiques. La dernière est une question algorithmique qui soulève elle-même des questions d'ordre mathématiques comme sur la vitesse de convergence d'un algorithme donné et son coût computationel.

L'objectif de ce cours est de donner des éléments de réponse a ces quatre questions dans le cas où f est différentiable et K est une contrainte définie par des fonctions différentiables. Le rôle joué par cette hypothèse de différentiation est de permettent de décrire "simplement" une fonction f et une contrainte K localement, c'est-à-dire en un point.

1. CREST, ENSAE. Bureau 3029, 5 avenue Henry Le Chatelier. 91 120 Palaiseau. Email : guillaume.lecue@ensae.fr.

Remarque 1.1 *Dans la formalisation du problème* (1.1)*, il y a a priori un petit abus de notation car au moment de poser le problème, on ne sait pas a priori s'il a une solution ; on devrait alors parler du problème* $\inf(f(x) : x \in K)$ *plutôt que de poser le problème directement avec un* min*.*

Définition 1.2 *La fonction f qu'on cherche à optimiser dans* (1.1) *est appelée* ***fonction objectif*** *et le sous-ensemble K est appelé la* ***contrainte****. Une* ***solution au problème*** (1.1) *est un élément $x^* \in K$ tel que $f(x^*) = \min(f(x) : x \in K)$. L'ensemble des solutions au problème* (1.1) *est noté* $\operatorname{argmin}_{x\in K} f(x)$ *:*

$$\operatorname*{argmin}_{x\in K} f(x) = \{x^* \in K : f(x^*) = \min(f(x) : x \in K)\}.$$

(C'est un ensemble.) Dire que (1.1) ***admet une solution*** *signifie que* $\operatorname{argmin}_{x\in K} f(x)$ *est non vide. De même,* ***le problème*** (1.1) ***admet une unique solution*** *signifie que* $\operatorname{argmin}_{x\in K} f(x)$ *contient un unique élément.*

Idéalement, on voudrait pouvoir trouver une solution x^* au problème (1.1) et ensuite, si nécessaire, en déduire la valeur $f(x^*)$. Mais pour certains problèmes, on ne sait qu'estimer la valeur minimale de f sur K. Par ailleurs, d'un point de vue purement pratique, on ne sera pas, en général, en mesure de trouver exactement une solution x^* au problème mais on se contentera d'une approximation de x^*, par exemple, par des algorithmes itératifs.

Remarque 1.3 *On ne s'intéresse qu'au problème de minimisation vu qu'en remplaçant f par son opposé $-f$ dans un problème de maximisation on retrouve un problème de minimisation :*

$$\max(f(x) : x \in K) = -\min(-f(x) : x \in K)$$

et les ensembles de solutions sont les même :

$$\operatorname*{argmax}_{x\in K} f(x) = \operatorname*{argmin}_{x\in K}(-f(x)).$$

Il existe plusieurs familles de problèmes d'optimisation. On en donne trois dans la définition suivante. Etant donné un problème d'optimisation, il est toujours bon d'identifier dans quelle famille il se situe.

Définition 1.4 *Soit U un ouvert de $\mathbb{R}^n$ et $K \subset U$. Soit $f : U \to \mathbb{R}$. On considère le problème d'optimisation sous contrainte* $\min_{x\in K} f(x)$ *un fermé de $\mathbb{R}^n$. On dit que ce problème est*

- *un* ***problème d'optimisation différentiable (OD)*** *quand f est différentiable et s'il existe $g_1, \dots, g_r, h_1, \dots, h_l : U \to \mathbb{R}$ différentiables tels que*

$$K = \{x \in U : g_1(x) = \cdots = g_r(x), h_1(x) \le 0, \cdots, h_l(x) \le 0\};$$

- *un* ***problème d'optimisation convexe différentiable (OCD)*** *quand f est convexe différentiable et s'il existe $A \in \mathbb{R}^{r\times n}, b \in \mathbb{R}^r$ et $h_1, \dots, h_l : U \to \mathbb{R}$ convexes et différentiables tels que*

$$K = \{x \in U : Ax = b, h_1(x) \le 0, \cdots, h_l(x) \le 0\}.$$

Il existe d'autres familles de problèmes d'optimisation, comme l'optimisation convexe quand f et K sont convexes, ou l'optimisation discrète quand K est un ensemble discret. Dans ce cours nous n'étudierons que les problèmes (OD) et (OCD).

On présente maintenant quelques problèmes concrets qui s'écrivent sous la forme d'un problème d'optimisation.

1.1 Transport optimal de marchandises

Énoncé du problème : Des magasins (numérotés de 1 à n) ont une demande $d_1, \dots, d_n$ pour un produit. On dispose de plusieurs entrepôts (numérotés de 1 à m) ayant chacun pour stock $s_1, \dots, s_m$ de ce produit. Sachant que le coût unitaire de transport d'un produit de l'entrepôt j vers le magasin i est donné par c_{ij}, quelle stratégie minimise le coût de transport total pour satisfaire la demande de tous les magasins ?

Modélisation mathématique : Pour tout $1 \leq i \leq n, 1 \leq j \leq m$ on note x_{ij} le nombre de produits transportés de l'entrepôt j vers le magasin i. Étant donné une matrice de transport $x = (x_{ij}) \in \mathbb{R}^{n \times m}$, le coût total de transport est donné par $\sum_{\substack{i=1,\dots,n \\ j=1,\dots,m}} x_{ij} c_{ij}$. On définit ainsi la fonction objectif par

$$x \in \mathbb{R}^{n \times m} \to f(x) = \sum_{i,j} x_{ij} c_{ij}. \tag{1.2}$$

L'ensemble K de contraintes est un sous-ensemble de $\mathbb{R}^{n \times m}$ de toutes les matrices $x = (x_{ij})$ vérifiant des contraintes physiques et d'approvisionnement :

- chaque magasin $i = 1, \dots, n$ veut répondre à sa demande : $\sum_{j=1}^{m} x_{ij} \geq d_i$
- chaque entrepôt $j = 1, \dots, m$ ne peut pas fournir plus que son stock : $\sum_{i=1}^{n} x_{ij} \leq s_j$
- il ne peut y avoir qu'un nombre positif de produits déplacés : $x_{ij} \geq 0$. (On pourrait aussi préciser la contrainte avec $x_{ij} \in \mathbb{N}$, mais on verra par la suite pourquoi on ne le fait pas).

On définit alors l'ensemble de contrainte par

$$K = \left\{ x = (x_{ij}) \in \mathbb{R}^{n \times m} : \forall j = 1, \dots, m, i = 1, \dots, n, \sum_{j=1}^{m} x_{ij} \geq d_i; \sum_{i=1}^{n} x_{ij} \leq s_j; x_{ij} \geq 0 \right\}. \tag{1.3}$$

Résoudre le problème d'allocation optimal de produit à partir de m entrepôts vers n magasins connaissant le coût unitaire de transport de chaque entrepôt vers chaque magasin ainsi que la demande de chaque magasin et le stock de chaque entrepôt se résout donc en trouvant une solution au problème d'optimisation de la forme (1.1) où f est donnée en (1.2) et K est donnée en (1.3).

Par ailleurs, ici, la fonction objectif et la contrainte sont définies par des fonctions linéaires. C'est un cas particulièrement favorable pour la résolution numérique du problème. On parle alors de **linear programming**.

Définition 1.5 *On dit que (1.1) est un* ***problème d'optimisation convexe*** *quand la contrainte K est un ensemble convexe et quand la fonction objectif est convexe. On dit que (1.1) est un* ***problème d'optimisation linéaire*** *quand la fonction objectif est une fonction affine et quand la contrainte K est l'intersection d'hyperplans et de demi-plans.*

Remarque 1.6 *On aurait pu ajouter la contrainte $x_{ij} \in \mathbb{N}$ dans K si les produits à transporter sont insécables (ce qui n'est pas le cas si on transporte des liquides comme de l'essence par exemple). Le problème d'optimisation associé est alors beaucoup plus compliqué à résoudre que celui n'imposant que la contrainte $x_{ij} \geq 0$. En effet, la convexité de la contrainte est un critère important pour le résolution pratique du problème. On aura donc tendance à "convexifier" les contraintes non-convexes : ici on passe de $\mathbb{N}$ à $\mathbb{R}_+$ par convexification de $\mathbb{N}$. Le problème résultant de cette convexification est un linear programming, donc beaucoup plus facile à résoudre que dans le cas où on restreints les x_{ij} à être dans $\mathbb{N}$. On donne une solution à coordonnées entière en prenant la partie entière des coordonnées d'une solution au problème de linear programming.*

1.2 Construction d'un filtre anti-spams

Énoncé du problème : On dispose d'un ensemble d'emails qu'on a nous même étiquetés comme *SPAM* ou *EMAIL*. C'est notre base de données d'apprentissage, qu'on note $(X_i, Y_i)_{i=1}^n$, où X_i est le i-ième email et $Y_i \in \{SPAM, EMAIL\}$ est le *label* ou étiquette associé à X_i. On souhaite construire un filtre anti-spam à partir de cette base de données. C'est-à-dire, étant donné un nouvel email, on souhaite être capable de dire automatiquement si c'est une *SPAM* ou un *EMAIL*.

Featuring : La première étape pour résoudre un problème d'apprentissage est de transformer les données d'entrées, les X_i, en un vecteur ; c'est ce qu'on appelle le featuring. Cette transformation doit être pertinente vis-à-vis du problème posé. On peut par exemple, pour chaque email construire des features comme 1) taille de l'email 2) nombre d'images 3) nombre de fautes d'orthographe 4) présence ou non du nom/prénom du destinataire 5) heure d'envoi de l'email 6) nombre de destinataires 7) présence de certains mots etc.. Disons, qu'on a construit p features. On se retrouve alors après le featuring avec n couples $(\tilde{X}_i, Y_i)_{i=1}^n$ où $\tilde{X}_i$ est le vecteur des features de X_i associé à son label Y_i. Aussi pour simplifier, on va prendre $Y_i = 1$ quand c'est un *SPAM* et $Y_i = -1$ quand c'est un *EMAIL*.

Procédure (ou modèle) : A partir de la base de données "featurisées" $(\tilde{X}_i, Y_i)_{i=1}^n$, on va partitionner l'espace $\mathbb{R}^p$ en deux parties disjointes $\mathbb{R}^p = \mathcal{S} \sqcup \mathcal{E}$: une partie telle que si on reçoit un email X tel que $\tilde{X} \in \mathcal{S}$ (où $\tilde{X}$ est le vecteur de features de X) alors on décidera que X est un *SPAM* et une deuxième partie telle que si $\tilde{X} \in \mathcal{E}$ alors on décidera que c'est un *EMAIL*. Une première approche est de chercher un séparateur linéaire, càd de partitionner l'espace des features $\mathbb{R}^p$ en deux demi-espaces affines.

Dans ce cas, on va chercher un vecteur $w \in \mathbb{R}^p$ et $b \in \mathbb{R}$ tels que l'hyperplan séparateur $\{x \in \mathbb{R}^p : \langle x, w \rangle + b = 0\}$ fasse le moins d'erreur possible sur la base de données $(\tilde{X}_i, Y_i)_{i=1}^n$. Une erreur est commise sur la i-ième donnée quand $Y_i \neq \text{sgn}(\langle w, \tilde{X}_i \rangle + b)$. Le nombre total d'erreurs de l'hyper-plan séparateur associé au paramètre $(w, b) \in \mathbb{R}^p \times \mathbb{R}$ sur la base de données est donc donné par $\sum_{i=1}^n I(Y_i \neq \text{sgn}(\langle w, \tilde{X}_i \rangle + b))$ (où on note $I(A) = 1$ quand A est vrai et 0 sinon). Pour minimiser ce nombre d'erreurs total, on va chercher à résoudre le problème d'optimisation

$$\min \left(\sum_{i=1}^n I(Y_i \neq \text{sgn}(\langle w, \tilde{X}_i \rangle + b)) : w \in \mathbb{R}^p, b \in \mathbb{R} \right). \tag{1.4}$$

On retrouve bien un problème de la forme (1.1) où la contrainte K est tout l'espace $\mathbb{R}^p \times \mathbb{R}$ et la fonction objectif est

$$f : \left\{ \begin{array}{ccc} \mathbb{R}^p \times \mathbb{R} & \to & \mathbb{R} \\ (w, b) & \to & \sum_{i=1}^n I(Y_i \neq \text{sgn}(\langle w, \tilde{X}_i \rangle + b)). \end{array} \right. \tag{1.5}$$

Comme la contrainte est tout l'espace des paramètres, on dit que le problème est **sans contrainte**.

On l'a déjà évoqué un peu plus haut, un point clef de la résolution pratique d'un problème d'optimisation par un algorithme 'efficace' (c'est-à-dire retournant une réponse en un temps acceptable) est la convexité (de f et K). Ici, la fonction objectif (1.5) n'est pas convexe et le problème d'optimisation associé est en général très difficile à résoudre algorithmiquement. Une stratégie classique est alors de faire une **relaxation convexe** : on cherche une fonction convexe proche de la fonction objectif. Pour ce faire, on voit que

$$I(Y_i \neq \text{sgn}(\langle w, \tilde{X}_i \rangle + b)) = I(Y_i(\langle w, \tilde{X}_i \rangle + b) \leq 0).$$

On va alors remplacer la fonction $t \in \mathbb{R} \to I(t \leq 0)$ (qui est non convexe) par une fonction convexe qui fait sens pour le problème (càd qui charge plus les nombres négatif – car se sont des

erreurs – que les nombres positifs) ; par exemple, on peut prendre $\phi : t \in \mathbb{R} \to \max(1-t, 0)$. On obtient alors une nouvelle fonction objectif donnée par

$$\tilde{f} : \left\{ \begin{array}{ccc} \mathbb{R}^p \times \mathbb{R} & \to & \mathbb{R} \\ (w, b) & \to & \sum_{i=1}^n \phi(Y_i(\langle w, \tilde{X}_i \rangle + b)). \end{array} \right. \tag{1.6}$$

Le problème d'optimisation associé $\min(\tilde{f}(w, b) : (w, b) \in \mathbb{R}^p \times \mathbb{R})$ est convexe (en tant que somme de composées de fonctions convexes ϕ et de fonctions affines $(w, b) \to \langle w, \tilde{X}_i \rangle + b$) et peut donc se résoudre numériquement par des algorithmes classiques développés en *optimisation convexe* (que nous verrons au dernier chapitre du cours). Néanmoins, il n'est pas différentiable car ϕ n'est pas dérivable en 1. On peut aussi, choisir une autre relaxation convexe comme $\phi(t) = \exp(-t)$ et avoir une problème d'optimisation associé qui est convexe et différentiable.

1.3 Séparateur linéaire à forte marge

Énoncé du problème : A l'image du problème précédent, on dispose d'un ensemble de points étiquetés $\{(X_i, Y_i) : i = 1, \ldots, n\}$ où $X_i \in \mathbb{R}^p$ et $Y_i \in \{-1, 1\}$. Cette fois, on suppose que les données sont linéairement séparables, càd qu'il existe un hyper-plan de $\mathbb{R}^p$ permettant de séparer les données étiquetées $+1$ des données étiquetées -1. Un tel hyper-plan est appelé hyper-plan séparateur. En général (quand p est suffisamment grand) il en existe plusieurs et on va en choisir un tel que les deux nuages de points $\{X_i : Y_i = 1\}$ et $\{X_i : Y_i = -1\}$ soient les plus éloignés possible de l'hyper-plan séparateur comme dans la Figure 1.

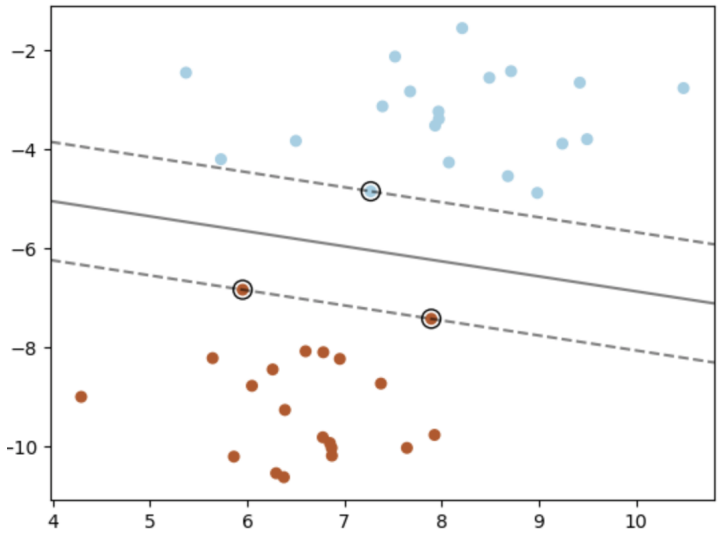

FIGURE 1 – Séparateur linéaire à forte marge.

Procédure : Un hyper-plan peut être paramétré par un vecteur normal $w \in \mathbb{R}^p$ et un intercept $b \in \mathbb{R}$ ainsi $\Delta(w, b) = \{x \in \mathbb{R}^p : \langle w, x \rangle + b = 0\}$. Dans un premier temps, il faut qu'on détermine la distance entre un point $x \in \mathbb{R}^p$ et un hyper-plan $\Delta(w, b)$. On rappelle d'abord, la définition de distance d'un point à un ensemble : si $x \in \mathbb{R}^p$ et $E \subset \mathbb{R}^p$ alors on définit la distance (euclidienne) de x à E par

$$d(x, E) = \inf_{y \in E} \|x - y\|_2 .$$

Quand E est un hyper-plan de la forme $\Delta(w,b)$, on peut expliciter cette distance en fonction de x, w et b. Pour cela, on va montrer le lemme suivant.

Lemme 1.7 *Soit $w, x \in \mathbb{R}^p$ et $b \in \mathbb{R}$. La distance entre x et $\Delta(w,b)$ est donnée par*

$$d(x, \Delta(w,b)) = \frac{|\langle x, w\rangle + b|}{\|w\|_2}.$$

Preuve. On note par $p(x)$ la projection de x sur $\Delta(w,b)$. C'est par définition l'unique point tel que $p(x) \in \operatorname{argmin}_{y\in\Delta(w,b)} \|y - x\|_2$. On a donc $p(x) \in \Delta(w,b)$ et pour tout $y \in \Delta(w,b)$, $\|p(x) - x\|_2^2 \leq \|y - x\|_2^2$. En particulier,

$$\|p(x) - x\|_2^2 \leq \|y - x\|_2^2 = \|y - p(x)\|_2^2 + 2\langle y - p(x), p(x) - x\rangle + \|p(x) - x\|_2^2$$

donc $2\langle y - p(x), x - p(x)\rangle \leq \|y - p(x)\|_2^2$. Ceci étant vrai pour tout $y \in \Delta(w,b)$ et $\{y - p(x) : y \in \Delta(w,b)\} = \operatorname{vect}(w)^\top$, on a pour tout $z \in \operatorname{vect}(w)^\top$, $2\langle z, x - p(x)\rangle \leq \|z\|_2^2$. Or étant donné $z \in \operatorname{vect}(w)^\top$, on a pour tout $\lambda \in \mathbb{R}$, $\lambda z \in \Delta(w,b)$, donc $2\lambda\langle z, x - p(x)\rangle \leq |\lambda|^2 \|z\|_2^2$. Ceci étant vrai pour tout $\lambda \in \mathbb{R}$, on a $\langle z, x - p(x)\rangle = 0$ pour tout $z \in \operatorname{vect}(w)^\top$. Autrement dit $x - p(x) \in \operatorname{vect}(w)$.

On note $\mu \in \mathbb{R}$ tel que $x - p(x) = \mu w$. Comme $p(x) \in \Delta(w,b)$, on a $\langle p(x), w\rangle + b = 0$ et donc $\langle x + \mu w, w\rangle + b = 0$. On obtient alors, $-\mu \|w\|_2^2 = \langle x, w\rangle + b$ et donc

$$p(x) = x - \mu w = x + \frac{\langle x, w\rangle + b}{\|w\|_2^2}.$$

Par ailleurs,

$$d(x, \Delta(w,b)) = \|x - p(x)\|_2 = |\mu| \, \|w\|_2 = \frac{|\langle x, w\rangle + b|}{\|w\|_2}.$$

■

Notre objectif est de trouver un hyper-plan séparateur ayant la plus forte marge. On écrit d'abord les conditions que doivent vérifier un hyper-plan séparateur : $\Delta(w,b)$ sépare les deux nuages de points $\{X_i : Y_i = 1\}$ et $\{X_i : Y_i = -1\}$ quand

$$Y_i(\langle w, X_i\rangle + b) \geq 0, \forall i = 1, \ldots, n.$$

La **marge** est par définition la plus petite distance entre les nuages de points $\{X_i : Y_i = 1\}$ et $\{X_i : Y_i = -1\}$ et l'hyper-plan séparateur $\Delta(w,b)$. Pour un $w \in \mathbb{R}^p$ et $b \in \mathbb{R}$ donnés, elle vaut

$$\min_{i=1,\ldots,n} \frac{|\langle X_i, w\rangle + b|}{\|w\|_2}. \tag{1.7}$$

C'est la fonction objectif de notre problème d'optimisation qu'on cherche à maximiser parmi tous les hyper-plans séparateurs. On aboutit alors au problème d'optimisation suivant :

$$\max_{w\in\mathbb{R}^p, b\in\mathbb{R}} \left(\min_{i=1,\ldots,n} \frac{|\langle X_i, w\rangle + b|}{\|w\|_2} : Y_i(\langle w, X_i\rangle + b) \geq 0, \forall i = 1, \ldots, n \right). \tag{1.8}$$

C'est bien un problème de la forme (1.1) mais la fonction objectif n'est pas convexe. Le problème (1.8) est donc difficile à résoudre numériquement. On va alors le réécrire de manière plus simple. En fait, on va montrer qu'il existe un problème d'optimisation convexe qui permet de résoudre

(1.8). Dans ce cas, on dira que les deux problèmes d'optimisation sont équivalents lorsque les solutions de l'un permettent d'obtenir les solutions de l'autre.

On considère le problème d'optimisation

$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}} \left(\|w\|_2 : Y_i(\langle w, X_i\rangle + b) \geq 1, \forall i = 1, \dots, n \right). \tag{1.9}$$

On montre que les deux problèmes (1.8) et (1.9) sont équivalents dans le lemme suivant.

Lemme 1.8 *Les deux problèmes (1.8) et (1.9) sont équivalents :*

i) Si $(\hat{w}, \hat{b})$ est solution de (1.8) alors $(\hat{w}/\alpha, \hat{b}/\alpha)$ pour $\alpha = \min_{i=1,\dots,n} |\langle X_i, \hat{w}\rangle + \hat{b}|$ est solution de (1.9).

ii) Si $(\hat{w}, \hat{b})$ est solution de (1.9) alors pour tout $\alpha > 0$, $(\hat{w}/\alpha, \hat{b}/\alpha)$ est solution de (1.8).

Preuve. Preuve de *i)*. On note $\alpha = \min_{i=1,\dots,n} |\langle X_i, \hat{w}\rangle + \hat{b}|$. On a alors $(\tilde{w}, \tilde{b}) = (\hat{w}/\alpha, \hat{b}/\alpha)$ est aussi solution de (1.8) vu que si (w, b) est séparateur alors $(w/\alpha, b/\alpha)$ l'est aussi et que la fonction objectif est homogène. Or

$$\min_{i=1,\dots,n} \frac{|\langle X_i, \tilde{w}\rangle + \tilde{b}|}{\|\tilde{w}\|_2} = \frac{1}{\|\tilde{w}\|_2}$$

et $Y_i(\langle \tilde{w}, X_i\rangle + \tilde{b}) \geq 1$ (car $Y_i(\langle \tilde{w}, X_i\rangle + \tilde{b}) > 0$ et $\min_{i=1,\dots,n} |\langle \tilde{w}, X_i\rangle + \tilde{b}| = 1$) donc $(\tilde{w}, \tilde{b}) \in \operatorname{argmin}_{w,b} \left(1/\|w\|_2 : Y_i(\langle w, X_i\rangle + b) \geq 1\right)$. Par ailleurs,

$$\operatorname*{argmax}_{w\in\mathbb{R}^p, b\in\mathbb{R}} \left(\frac{1}{\|w\|_2} : Y_i(\langle w, X_i\rangle + b) \geq 1 \right) = \operatorname*{argmin}_{w\in\mathbb{R}^p, b\in\mathbb{R}} \left(\|w\|_2 : Y_i(\langle w, X_i\rangle + b) \geq 1 \right). \tag{1.10}$$

Donc $(\tilde{w}, \tilde{b})$ est bien solution de (1.9).

Preuve de *ii)*. Si $(\hat{w}, \hat{b})$ est solution de (1.9) alors d'après (1.10), $(\hat{w}, \hat{b})$ est aussi solution de

$$\max_{w\in\mathbb{R}^p, b\in\mathbb{R}} \left(\frac{1}{\|w\|_2} : Y_i(\langle w, X_i\rangle + b) \geq 1 \right).$$

De plus pour tout $\alpha > 0$, $(\tilde{w}, \tilde{b}) = (\hat{w}/\alpha, \hat{b}/\alpha)$ est solution de

$$\max_{w\in\mathbb{R}^p, b\in\mathbb{R}} \left(\frac{\alpha}{\|w\|_2} : Y_i(\langle w, X_i\rangle + b) \geq \alpha \right).$$

En particulier, on a $\min_{i=1,\dots,n} |(\langle \tilde{w}, X_i\rangle + \tilde{b}| = \alpha$ (sinon, on pourrait augmenter la fonction objectif dans le problème d'optimisation précédent). De plus, $Y_i(\langle w, X_i\rangle + b) \geq \min_{i=1,\dots,n} |(\langle w, X_i\rangle + b|$ pour tout $i = 1, \dots, n$ si et seulement si $Y_i(\langle w, X_i\rangle + b) \geq 0$ pour tout $i = 1, \dots, n$ car $|Y_i| = 1$. Donc $(\tilde{w}, \tilde{b})$ est bien solution de (1.8). ∎

D'après le Lemme 1.8, il suffit de résoudre (1.9) pour résoudre (1.8). Or (1.9) est un problème d'optimisation dont la fonction objectif est convexe et les contraintes sont affines. C'est donc un problèmes qu'on pourra résoudre assez facilement en pratique. En fait, on peut même voir que (1.9) entre dans la classe des problèmes quadratiques pour lesquels il existe des algorithmes efficaces.

Définition 1.9 *Un problème d'optimisation est dit* ***quadratique (quadratic programming)*** *quand il peut s'écrire sous la forme :*

$$\min_{x\in\mathbb{R}^p} \left(\frac{1}{2} x^\top Q x + \langle q, x\rangle : Ax \leq b \right)$$

où $Q \in \mathcal{S}_d$ est une matrice symétrique de $\mathbb{R}^{d\times d}$, $q \in \mathbb{R}^d$, $A \in \mathbb{R}^{p\times d}$ et $b \in \mathbb{R}^p$. Par convention, l'inégalité "$Ax \leq b$" signifie que pour tout $j = 1, \dots, p, (Ax)_j \leq b_j$.

Remarque 1.10 *Ce qu'on a fait ici sur l'espace $\mathbb{R}^p$ s'étend facilement à n'importe quel espace de Hilbert, et en particulier, au espace de Hilbert de dimension infinie. Dans ce cas, il existe toujours un hyperplan séparant un nombre fini (ici n) de points. Etant donné des données d'entrées $X_i, i = 1, \cdots, n$ labelisées à valeurs dans un espace quelconque $\mathcal{X}$ (ça peut être par exemple l'ensemble de tous les mots sur le dictionnaire $\{A, T, C, G\}$ pour les problèmes de génétique), on peut les plonger dans un espace de Hilbert de dimension infinie $X_i \in \mathcal{X} \to \tilde{X}_i \in H$ et chercher les séparateurs linéaires à forte marge dans H pour les données $(\tilde{X}_i, Y_i)_{i=1}^n$. Il existe une multitude de plongement de ce type et ces techniques sont appelées* ***les méthodes à noyaux*** *en machine learning car le plongement $\mathcal{X} \to H$ se fait grâce à un noyau $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ par $x \in \mathcal{X} \to K(x, \cdot) \in H$ (à chaque point x de $\mathcal{X}$, on associe une fonction $K(x, \cdot)$). Cette approche marche pour tout type d'espace ; on peut donc plonger un espace ne possédant aucune structure (même pas une addition) dans un espace très structuré comme un espace de Hilbert.*

1.4 Estimateur des moindres carrés

Énoncé et modélisation du problème : On souhaite prédire la consommation de gaz en fonction de la température extérieure. On dispose d'une historique de consommation et de température $(T_i, C_i)_{i=1}^n$ où T_i est la température du jour i et C_i est la température du jour i. On veut pouvoir inférer la consommation C étant donnée une température T.

Procédure statistique : L'approche classique pour résoudre ce problème est de chercher une droite dans $\mathbb{R}^2$ qui minimise la somme des résidus de la droite au nuage de points $\{(T_i, C_i) : i = 1, \ldots, n\}$. On fait donc ici l'hypothèse que la consommation de gaz peut s'expliquer de manière affine en fonction de la température (c'est une hypothèse forte). Une droite de $\mathbb{R}^2$ est habituellement paramétrée par un coefficient directeur $w \in \mathbb{R}$ et une ordonnée à l'origine $b \in \mathbb{R}$. Étant donné une droite $T \to wT + b$, le résidu à la i-ième donnée est $(C_i - (wT_i + b))^2$ (on pourrait aussi prendre une erreur en $|C_i - (wT_i + b)|$; ici on prends le carré car il y a un modèle probabiliste derrière qui le justifie, mais qui n'est pas présenté ici par soucis de concision). En effet, si on avait prédit la consommation du jour i par $wT_i + b$ alors on aurait commis l'erreur $(C_i - (wT_i + b))^2$. On cherche la droite affine faisant le moins d'erreur possible sur la base de données. Ceci nous donne notre fonction objectif : c'est la somme totale des erreurs faites par $T \to wT + b$ sur la base de données,

$$f : \left\{ \begin{array}{ccc} \mathbb{R}^p \times \mathbb{R} & \to & \mathbb{R} \\ (w, b) & \to & \sum_{i=1}^n (Y_i - (wT_i + b))^2. \end{array} \right. \tag{1.11}$$

Le problème d'optimisation associé est $\min(f(w, b) : (w, b) \in \mathbb{R}^2 \times \mathbb{R})$, qui est donc un problème d'optimisation sans contrainte, convexe et différentiable.

Critique du modèle et amélioration : Dans l'approche précédente, on fait l'hypothèse que la consommation s'explique bien par une fonction affine de la température. C'est une hypothèse forte qui n'est en fait pas vérifiée en pratique. Une manière d'affiner l'inférence est d'augmenter le nombre de features. Dans le modèle précédent, il n'y a qu'une seule feature, la température ; on peut en ajouter d'autres par exemple, les températures et consomations des jours précédents, la force du vent, des features calendaires (jour de la semaine, jour férié, jour de vacance, etc.). Disons qu'on aboutit à un nombre p de features. Pour chaque jour i, on a donc un vecteur $X_i \in \mathbb{R}^p$ de features qui, on l'espère, devrait permettre de bien expliquer la consommation C_i. Si on fait l'hypothèse que C_i peut être prédit de manière affine à partir de X_i, on va alors chercher un vecteur $w \in \mathbb{R}^p$ et un **intercept** $b \in \mathbb{R}$ tel que la somme des erreurs soit petite

$$\sum_{i=1}^n (C_i - (\langle X_i, w \rangle + b))^2.$$

On est alors amené à résoudre la problème d'optimisation suivant

$$(\hat{w}, \hat{b}) \in \underset{(w,b)\in\mathbb{R}^p\times\mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} (C_i - (\langle X_i, w\rangle + b))^2$$

qui est aussi sans contrainte, convexe et différentiable. L'estimateur associé de la consommation est la fonction $X \in \mathbb{R}^p \to \langle \hat{w}, X\rangle + \hat{b}$; on l'appelle **l'estimateur des moindres carrés**

1.5 Estimation d'un signal constant par bloc

Énoncé et modélisation du problème : On fait des observations bruitées d'un signal constant par bloc et on cherche à reconstruire ce signal. On observe $y_i = f(i/n) + g_i$ pour $i = 1, \ldots, n$ où $g_1, \ldots, g_n$ sont des Gaussiennes i.i.d. centrées de variance σ^2 et $f : [0, 1] \to \mathbb{R}$ est un fonction constante par morceau. On souhaite reconstruire le vecteur $(f(i/n))_{i=1,\ldots,n}$.

Procédure statistique : Pour résoudre ce genre de problème, on a souvent recours à des procédures qui cherchent à faire un meilleur compromis entre un terme d'adéquation aux données (souvent donné par la vraisemblance du modèle) et l'a priori qu'on a sur le signal à estimer ; ici, le signal est constant par bloc. Le terme d'adéquation aux données est ici le critère des moindres carrés : $x \in \mathbb{R}^n \to (1/n)\sum_{i=1}^n (x_i - y_i)^2$ (comme vu précédemment). Pour induire, une structure "constant par bloc", on introduit ce qui est appelé la "norme en variation totale" : $x \in \mathbb{R}^n \to \|x\|_{TV} = \sum_{i=1}^{n-1} |x_{i+1} - x_i|$. L'idée de cette "norme" est que si $\|x\|_{TV}$ est petit alors les coordonnées successives de x seront proches et donc elle induit une structure "constant par bloc" dans un vecteur – c'est ce qu'on cherche à faire ici. On va alors chercher un signal qui est à la fois proche des données et qui a une petite norme TV, par exemple en cherchant

$$\hat{x}_n \in \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \left(\frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2 + \lambda \|x\|_{TV} \right) \tag{1.12}$$

où λ est un paramètre appelé paramètre de régularisation qui quantifie le trade-off entre le terme d'adéquation aux données et le terme induisant la structure "constant par bloc" de la norme TV.

On retrouve bien un problème de la forme (1.1) où la fonction objectif est $x \in \mathbb{R}^n \to (1/n)\sum_{i=1}^n (x_i - y_i)^2 + \lambda \|x\|_{TV}$ et la contrainte $K = \mathbb{R}^n$ – c'est donc un problème d'optimisation sans contrainte. C'est un problème d'optimisation convexe mais pas différentiable car $x \to \|x\|_{TV}$ n'est pas différentiable.

1.6 Max-cut

Il existe de nombreux problèmes d'optimisation sur les graphes. Ces problèmes ont connus d'importants développements depuis l'émergence d'internet et de réseaux sociaux. On donne ici un exemple de problème d'optimisation sur un graphe connu sous le nom de problème de MAX-CUT (en français : coupure maximale).

On se donne un graphe $G = (V, E)$ où V est l'ensemble des sommets, généralement, on prends $V = \{1, \ldots, n\}$ et E est l'ensemble des arêtes du graphe : c'est un sous-ensemble de $V \times V$ ($(i, j) \in V$ ssi il y a une arête entre le noeud i et le noeud j). On suppose que le graphe est non orienté, ce qui est équivalent à dire que si $(i, j) \in V$ alors $(j, i) \in V$.

Un **CUT** de G est une partition de ces sommets en deux parties : $S \sqcup S^c = V$ (où S^c est le complémentaire de S dans V). MAX-CUT est le nom qu'on donne au problème qui consiste à trouver un CUT de G – càd un sous-ensemble $S \subset V$ – tel que le nombre d'arêtes entre S et son

complémentaire S^c est maximal. On écrit formellement ce problème du MAX-CUT par

$$\max_{S\subset V} \sum_{(i,j)\in V} I(i \in S \text{ et } j \in S^c). \tag{1.13}$$

Il est souvent utile de transformer les problèmes d'optimisation sur les graphes en des problèmes d'optimisation sur les matrices. Pour cela, on introduit la **matrice d'adjacence** d'un graphe $G = (V, E)$ où $V = \{1, \dots, n\}$ par

$$A_{ij} = \begin{cases} 1 & \text{si } (i,j) \in V \\ 0 & \text{sinon.} \end{cases}$$

En utilisant cette notation on voit que pour tout $i, j \in \{1, \dots, n\}$, on a $I(i \in S \text{ et } j \in S^c) = (1 - x_i x_j)/2$ où $x = (x_i)_{i\in V} \in \{-1, 1\}^n$ est tel que $x_i = 1$ si $i \in S$ et $x_i = -1$ si $i \in S^c$ et que sommer sur V est équivalent à sommer sur $\{1, \dots, n\}$ si on multiplie le terme de sommation par A_{ij}. On voit alors que le problème du MAX-CUT peut se réécrire comme le problème d'optimisation suivant :

$$\max_{x\in\{-1,1\}^n} \frac{1}{2} \sum_{i,j=1}^{n} A_{ij}(1 - x_j x_i). \tag{1.14}$$

En effet, si x^* est solution du problème précédent alors en posant $S^* = \{i \in \{1, \dots, n\} : x_i^* = 1\}$, on obtient une solution au problème initial.

On peut aussi réécrire le problème (1.14) sous la forme d'un problème de minimisation d'une fonctionnelle quadratique sous-contraintes :

$$\min_{x\in\mathbb{R}^n} \left(x^\top A x : x_i^2 = 1 \right). \tag{1.15}$$

C'est donc un problème d'optimisation sous contrainte ; la fonction objectif est une fonction quadratique $x \to x^\top A x$ et la contrainte est donnée par n contraintes d'égalité "$x_i^2 = 1$". La difficulté ici est que la contrainte n'est pas un ensemble convexe. Une approche possible est alors de 'convexifier' cet ensemble.

1.7 Calage sur marge dans la Macro Calmar de l'INSEE

Comme indiqué dans la note explicative de l'INSEE https://www.insee.fr/fr/information/2021902 : La macro SAS CALMAR (CALage sur MARges) permet de redresser un échantillon provenant d'une enquête par sondage, par re-pondération des individus sondés, en utilisant une information auxiliaire disponible sur un certain nombre de variables, appelées variables de calage. Le redressement consiste à remplacer les pondérations initiales (ou "poids de sondage") par de nouvelles pondérations telles que :

- pour une variable de calage catégorielle (ou "qualitative"), les effectifs des modalités de la variable estimés dans l'échantillon, après redressement, seront égaux aux effectifs connus sur la population ;
- pour une variable numérique (ou "quantitative"), le total de la variable estimé dans l'échantillon, après redressement, sera égal au total connu sur la population.

Cette méthode de redressement permet de réduire la variance d'échantillonnage, et, dans certains cas, de réduire le biais dû à la non réponse totale.

Formellement, on dispose d'un population totale $(X_i)_{i\in U}$ où pour tout $i \in U$, X_i est un vecteur de $\mathbb{R}^d$ et $|U| = N$. On réalise un sondage de taille n aléatoire de cette population : on choisit un sous-ensemble $S \subset U$ de taille n et on observe $(X_i)_{i\in S}$. On connaît $\pi_i = \mathbb{P}[i \in S]$ pour tout $i \in U$

(par exemple, pour un plan de sondage aléatoire uniforme on a $\pi_i = n/N$) – en fait, pour le calage sur marge, on ne doit connaître les π_i que pour les individus sondés, càd pour les $i \in S$. Les *poids de sondage* sont donnés par $d_i = 1/\pi_i$ pour tout $i \in U$. On dispose d'informations auxiliaires qui sont les effectifs totaux de la population totales sur chacune des d features : on connaît

$$m_j = \sum_{i \in U} X_{ij}, \text{ pour tout } j = 1, \dots, d$$

où on note X_{ij} la j-ième coordonnée de X_i pour tout $i \in U$ et $j = 1, \dots, d$. On souhaite alors trouver de nouveaux poids $(w_k)_{k \in S}$ pour tous les individus sondés tels que $(w_i)_{i \in \mathcal{S}}$ est proche des poids de sondage $(d_k)_{i \in \mathcal{S}}$ (pour une certaine notion de proximité à choisir) et vérifient les équations de calage sur les d marginales (ou features) :

$$\sum_{i \in S} w_i X_{ij} = m_j, \text{ pour tout } j = 1, \dots, d. \tag{1.16}$$

On souhaite donc que les d marginales sur les individus sondés soient égales aux d marginales sur la population totale.

On peut réécrire ce problème sous forme matricielle : on note $\mathbb{X} \in \mathbb{R}^{n \times d}$ la matrice ayant pour vecteurs lignes les n vecteurs $X_i, i \in S$. On voit alors que $w = (w_i)_{i \in S}$ vérifie les d équations de calage sur marge si et seulement si $\mathbb{X}^\top w = m$ où $m = (m_j)_{j=1}^d$. En particulier, les équations de calage admettent au moins une solution si et seulement si m est dans l'image de $\mathbb{X}^\top$. Comme $\mathrm{Im}(\mathbb{X}^\top) = \mathrm{Ker}(\mathbb{X})^\perp$, on voit qu'il est nécessaire et suffisant de vérifier que $m \in \mathrm{Ker}(\mathbb{X})^\perp$. Pour vérifier celà on peut trouver une base du noyau de $\mathbb{X}$ et vérifier que m est orthogonal à chaque élément de cette base. Dans la suite, on ne s'intéresse pas au problème d'existence d'une solution vérifiant exactement les équations de calage mais seulement à s'en approcher en cherchant w qui minimise

$$w = (w_i)_{i \in S} \to \frac{1}{2} \sum_{j=1}^{d} \left(\sum_{i \in S} w_i X_{ij} - m_j \right)^2 = \frac{1}{2} \left\| \mathbb{X}^\top w - m \right\|_2^2.$$

En supposant que le problème de calage sur marge admet bien une solution, on va chercher une solution w la plus proche possible des poids de sondage $(d_i)_{i \in S}$. Pour cela, on se fixe une mesure de similarité entre vecteurs de poids donnée par un noyau K définie sur l'ensemble des poids $\mathbb{R}^n \times \mathbb{R}^n$ (si on ne mets aucune contrainte sur les poids, même pas qu'ils sont positif ou de somme égale à 1). Pour le problème de calage sur marge, on cherche alors à résoudre le problème suivant :

$$\min_{(w_i)_{i \in S}} \left(K((w_i)_{i \in S}, (d_i)_{i \in S}) : \sum_{i \in S} w_i X_{ij} = m_j, j = 1, \dots, d \right). \tag{1.17}$$

Qui est un problème d'optimisation sous contrainte d'égalité linéaire et si K est convexe, c'est un problème d'optimisation convexe. En particulier, si la contrainte est non vide, elle est qualifiée.

Dans la macro CALMAR, les noyaux de similarités ont la forme

$$K((w_i)_{i \in S}, (d_i)_{i \in S}) = \sum_{i \in S} d_i G\left(\frac{w_i}{d_i} \right)$$

où la fonction G est convexe, $G(1) = G'(1) = 0$. On utilise dans la suite la fonction inverse de la dérivée de G : $F(u) = (G')^{(-1)}(u)$. On a par exemple,

$$G(r) = \frac{1}{2}(r-1)^2 \text{ et } F(u) = 1 + u \text{ (méthode linéaire)}$$
$$G(r) = r \log r - r + 1 \text{ et } F(u) = \exp(u), u > 0 \text{ (méthode raking ratio).}$$

Pour ce choix de noyau, le problème de calage sur marge s'écrit sous la forme

$$\min_{(w_i)_{i\in S}} \left(\sum_{i\in S} d_i G\left(\frac{w_i}{d_i}\right) : \sum_{i\in S} w_i X_{ij} = m_j, j = 1, \ldots, d \right) \tag{1.18}$$

et donc par KKT ('KKT' est le nom qu'on donne au théorème de Karush, Kuhn et Tucker ; c'est un des théorèmes classiques en optimisation sous contraintes, on le verra plus tard) si $(w_i)_{i\in S}$ est solution de ce problème alors il existe $(\lambda_i)_{i\in S}$ (qu'on appelle coefficient de Lagrange) tel que

$$G'\left(\frac{w_i}{d_i}\right) = \sum_{j=1}^{d} \lambda_j X_{ij} \text{ pour tout } i \in S$$

c'est-à-dire en utilisant F l'inverse de G' :

$$w_i = d_i F\left(\sum_{j=1}^{d} \lambda_j X_{ij}\right) \text{ pour tout } i \in S.$$

De plus, cette solution doit vérifier les équations de calage $\mathbb{X}^\top w = m$, ce qui permet de déterminer des coefficients de Lagrange $(\lambda_j)_{j=1}^d$: les $(\lambda_j)_{j=1}^d$ doivent vérifier

$$\sum_{i\in S} d_i F\left(\sum_{j'=1}^{d} \lambda_{j'} X_{ij'}\right) X_{ij} = m_j, \text{ pour tout } j = 1, \ldots, d.$$

On utilise une méthode de descente de gradient comme méthode numérique pour déterminer de tels $(\lambda_j)_{j=1}^d$ sur la fonction :

$$(\lambda_j)_{j=1}^d \to \sum_{j=1}^{d} \left(\sum_{i\in S} d_i F\left(\sum_{j'=1}^{d} \lambda_{j'} X_{ij'}\right) X_{ij} - m_j \right)^2.$$

Le critère d'arrêt dans CALMAR étant donné par

$$\max_{i\in S} \left| \frac{w_i^{(k+1)}}{d_i} - \frac{w_i^{(k)}}{d_i} \right| \leq \varepsilon$$

où $(w_i^{(k)})_{i\in S}$ est le vecteur de poids obtenu à partir des coefficients $(\lambda_j^{(k)})_{j=1}^d$ de la k-ième étape de descente de gradient.

Remarque : Les données catégorielles sont transformée en *dummy variables*.

2 Représentation graphique / visualisation géométrique

But : Le but de cette section est de décrire une manière de visualiser une fonction $f : \mathbb{R}^2 \to \mathbb{R}$ qui permet de comprendre le type d'équation que va vérifier une solution au problème (1.1). **L'idée principale est de décrire une fonction à partir de ces lignes de niveaux** à la manière d'une carte IGN, voir Figure 2 .

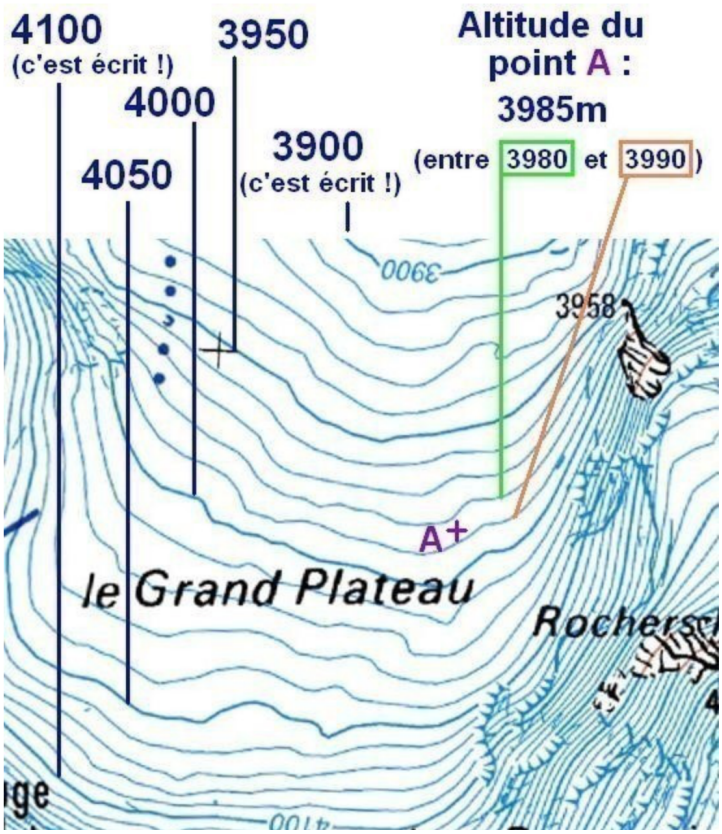

FIGURE 2 – Représentation d'un relief par les lignes de niveaux de la fonction altitude.

Définition 2.1 *Soit $f : \mathbb{R}^d \to \mathbb{R}$. Soit $\alpha \in \mathbb{R}$. La ligne de niveau α de f est*

$$\mathcal{L}_f(\alpha) = \{x \in \mathbb{R}^p : f(x) = \alpha\}.$$

Par exemple, si $f(x, y) = x^2 + y^2$ alors $\mathcal{L}_f(\alpha)$ est le cercle Euclidien centré en 0 et de rayon $\sqrt{\alpha}$ si $\alpha \geq 0$ et sinon, $\mathcal{L}_f(\alpha)$ est vide.

L'approche classique pour la représentation d'une fonction $f : \mathbb{R}^2 \to \mathbb{R}$ est de tracer une surface dans $\mathbb{R}^3$ où $\mathbb{R}^3$ est décomposé en un plan "xy" pour la variable $x \in \mathbb{R}^2$ et une droite "z" où sont représentées les valeurs prises par f (voir Figure 3). Cette manière de visualiser une fonction $f : \mathbb{R}^2 \to \mathbb{R}$ ne permet pas de visualiser facilement les idées clefs derrière la résolution des problèmes du type (1.1). On va plutôt utiliser une visualisation de $f : \mathbb{R}^2 \to \mathbb{R}$ par lignes de niveaux (voir Figure 4). Pour cela, on donne quelques lignes sous `pyhton` permettant de générer les deux types de visualisation (on encourage le lecteur à les reproduire et à en reproduire d'autres).

Exemple : visualisations de la différence de deux densités Gaussiennes.

On génère un réseau de points du plan 'xy' et les valeurs 'z' associées prise par la fonction "différence de deux Gaussiennes" par les lignes de code suivantes :

```
delta = 0.025
x = np.arange(-3.0, 3.0, delta)
y = np.arange(-2.0, 2.0, delta)
X, Y = np.meshgrid(x, y)
Z1 = mlab.bivariate_normal(X, Y, 1.0, 1.0, 0.0, 0.0)
```

```
Z2 = mlab.bivariate_normal(X, Y, 1.5, 0.5, 1, 1)
Z = 10.0 * (Z2 - Z1)
```

En première approche, on peut faire une visualisation 3D de cette surface comme dans la Figure 3. Ici, on représente tout l'espace $\mathbb{R}^3$. On obtient ce type de représentation sous python avec les lignes de code suivantes :

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, color='b')
```

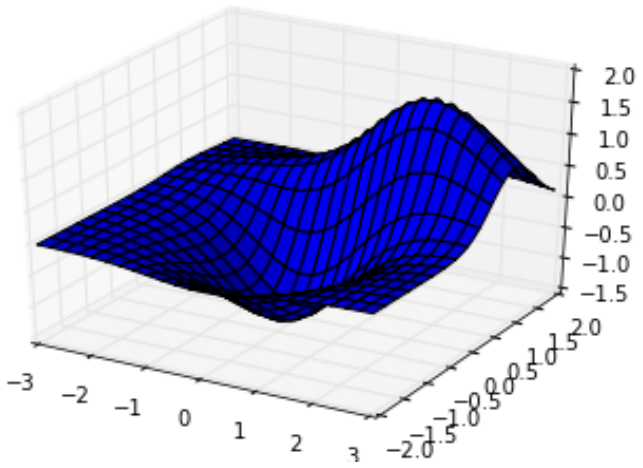

FIGURE 3 – Représentation en 3D d'une fonction $f : \mathbb{R}^2 \to \mathbb{R}$.

L'approche qu'on va utiliser est celle basée sur la visualisation dans $\mathbb{R}^2$ des lignes de niveau de f, comme dans la Figure 4. Ici, on travail dans $\mathbb{R}^2$, l'espace de la variable $x \in \mathbb{R}^2$. On obtient ce type de représentation sous python avec les lignes de code suivantes :

```
plt.figure()
CS = plt.contour(X, Y, Z)
plt.clabel(CS, inline=1, fontsize=10)
```

A retenir : Dans ce cours, on représente les fonctions $f : \mathbb{R}^2 \to \mathbb{R}$ par leurs lignes de niveaux (comme dans la Figure 4) et non par une représentation en 3D (comme dans la Figure 3). C'est en effet la représentation par lignes de niveaux qui permet de bien visualiser le problème d'optimisation (1.1).

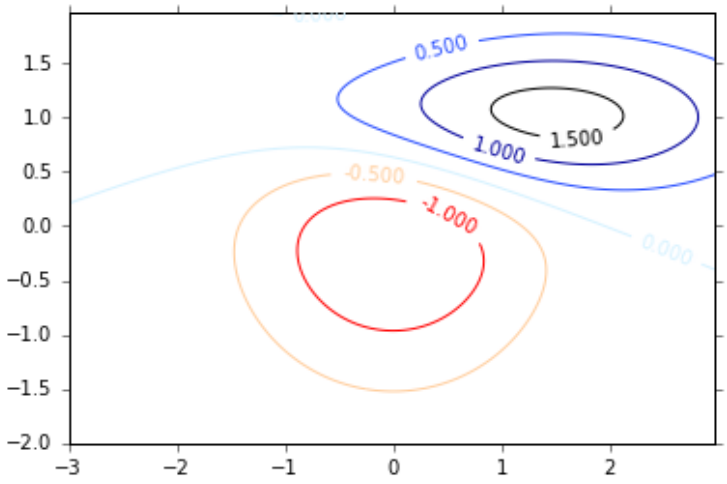

FIGURE 4 – Représentation par lignes de niveaux d'une fonction $f : \mathbb{R}^2 \to \mathbb{R}$.

3 Les problèmes d'optimisation sont des problèmes locaux

But : Dans cette section, on souhaite faire passer l'idée que les problèmes d'optimisation du type (1.1) sont des problèmes "locaux". C'est-à-dire que pour les résoudre, on a besoin de décrire localement la fonction f et la contrainte K. On commence d'abord par dire ce qu'on entend par "décrire localement une fonction $\mathbb{R}^d \to \mathbb{R}$ et un sous-ensemble de $\mathbb{R}^d$". Cette section est informelle et a pour but de faire passer les principales idées du cours et de l'organisation qui en découle. On formalisera ces concepts dans les 5 ou 6 cours suivants.

Décrire localement une fonction $\mathbb{R}^d \to \mathbb{R}$ signifie qu'on regarde à quoi ressemble une fonction autour d'un point donné. C'est ici que l'hypothèse de "différentiation" (inscrite dans le titre du cours) aide beaucoup car quand une fonction est différentiable en un point x^* alors la fonction f ressemble localement à une fonction affine de vecteur normal donné par le **gradient de f en x^***, noté $\nabla f(x^*)$, (on reviendra sur cette notion fondamentale tout au long du cours) et par l'**intercept** $f(x^*)$. En équation, quand f est différentiable en x^*, f est décrite localement en x^* par la fonction affine

$$F_{x^*} : x \in \mathbb{R}^d \to \langle \nabla f(x^*), x - x^* \rangle + f(x^*). \tag{3.1}$$

Cela signifie que la fonction affine ci-dessus est une approximation d'ordre 1 de f en x^*. En, particulier, on représentera la ligne de niveau $\alpha = f(x^*)$ de cette fonction affine. On voit que

$$\mathcal{L}_{F_{x^*}}(f(x^*)) = \{x : \langle \nabla f(x^*), x - x^* \rangle = 0\} = x^* + \text{vect}(\nabla f(x^*))^{\perp}, \tag{3.2}$$

C'est donc l'hyperplan qui passe par x^* et qui est orthogonal à $\nabla f(x^*)$. On verra par la suite que $\nabla f(x^*)$ est aussi un vecteur normal de la ligne de niveau $f(x^*)$ de f en x^* (ce qui est assez intuitif car $\mathcal{L}_{F_{x^*}}(f(x^*))$ est aussi une approximation du premier ordre de $\mathcal{L}_f(f(x^*))$ vu que F_{x^*} est une approximation au premier ordre de f en x^*).

La deuxième étape est maintenant de décrire localement une sous-ensemble de $\mathbb{R}^d$. Décrire localement un ensemble $K \subset \mathbb{R}^d$ est assez compliqué même sous l'hypothèse de différentiation. Cette hypothèse signifie que la contrainte K s'écrit sous la forme

$$K = \left\{ x \in U : \begin{array}{c} g_1(x) = \cdots = g_r(x) = 0 \\ h_1(x) \leq 0, \cdots, h_l(x) \leq 0 \end{array} \right\} \tag{3.3}$$

où $g_1, \dots, g_r : U \to \mathbb{R}$ et $h_1, \dots, h_l : U \to \mathbb{R}$ sont des classes $\mathcal{C}^1$ (et U est un ouvert de $\mathbb{R}^d$). Pour décrire localement en un point x^* un ensemble K, on aura recours à la notion de **cône tangent à K en x^***, noté $T_K(x^*)$, qui est une manière de généraliser la notion d'hyper-plan tangent d'une surface différentiable. Il se trouve que cette notion de cône tangent est elle-même difficile à manipuler pour le problème (1.1) qu'on souhaite résoudre, on aura alors recours à une hypothèse (qui a été beaucoup étudiée en optimisation), c'est **l'hypothèse de qualification**. Cette dernière hypothèse dit littéralement que le cône tangent à K (pour un K de la forme (3.3)) en x^* se décrit facilement par

$$T_K(x^*) = \left\{ v \in \mathbb{R}^d : \begin{array}{c} \langle \nabla g_1(x^*), v \rangle = \cdots = \langle \nabla g_r(x^*), v \rangle = 0 \\ \langle \nabla h_1(x^*), v \rangle \leq 0, \cdots, \langle \nabla h_l(x^*), v \rangle \leq 0 \end{array} \right\} \tag{3.4}$$

Une fois qu'on dispose des outils décrivant localement la fonction f et la contrainte K, on peut se représenter visuellement des conditions nécessaires satisfaites par le ou les points x^* de l'espace $\mathbb{R}^d$ (dans le cas $d = 2$) qui seront solution du problème (1.1) (voir Figure 5) :

- on part du minimum global de f. S'il est dans K alors ce minimum est solution de (1.1), sinon on fait croître les lignes de niveau de f, jusqu'à celle qui est la première à rencontrer la contrainte K. C'est en un tel point x^* que sera atteint le ou la solution au problème (1.1).
- En même temps qu'on fait croître les lignes de niveaux de f, on fait propager en son front, ces descriptions locales, en particulier, leur vecteur normal c'est à dire le gradient de f en chaque point dans le cas différentiable.
- Au point x^* de rencontre de la première ligne de niveau de f avec la contrainte K, on ne regarde plus que la description locale de f donnée par son gradient et celle de la contrainte donnée par $T_K(x^*)$. On voit alors que l'équation

$$-\nabla f(x^*) \in \left\{ z \in \mathbb{R}^d : \langle z, v \rangle \leq 0, \forall v \in T_K(x^*) \right\} := N_K(x^*)$$

  est satisfaite. Cette équation est centrale dans l'étude des problème d'optimisation ; on l'appellera **" condition nécessaire d'Euler/Péano/Kantorovitch"**.

Dans la troisième étape, l'ensemble $N_K(x^*)$ est apparu. On l'appelle le **cône normal à K en x^***. C'est une généralisation du vecteur normal à l'hyper-plan tangent à une fonction en x^* qui est le gradient $\nabla f(x^*)$ dans le cas différentiable. Sous l'hypothèse de qualification, ce cône normale s'écrit facilement :

$$N_K(x^*) = \left\{ \sum_{i=1}^{r} \beta_i \nabla g_i(x^*) + \sum_{j \in J(x^*)} \lambda_j \nabla h_j(x^*) : \beta_i \in \mathbb{R}, \lambda_j \geq 0 \right\} \tag{3.5}$$

et $J(x^*) = \{ j \in \{1, \dots, l\} : h_j(x^*) = 0 \}$. Dans ce cas, la condition $-\nabla f(x^*) \in N_K(x^*)$ est appelée la **condition de Karush-Kuhn-Tucker**, connue sous le nom de **condition KKT**.

Exemple : Trouver géométriquement la solution au problème

$$\min \left(x^2 + y^2 : (x-5)^2 + (y-5)^2 \leq 4 \right).$$

On représente d'abord la contrainte : c'est une boule Euclidienne de centre $(5, 5)$ et de rayon 2. Le minimum global de f sur $\mathbb{R}^2$ est atteint en $(0, 0)$ et la fonction objectif $f(x, y) = x^2 + y^2$ y prends la valeur 0. On fait alors croître les lignes de niveau de f à partir du niveau 0 jusqu'à rencontrer la contrainte. Le premier point de rencontre est en $x^* = (4, 4)$. Si on décrit la contrainte de la

manière canonique (3.3) $K = \{(x, y) \in \mathbb{R}^2 : h_1(x, y) \leq 0\}$ où $h_1(x, y) = (x-5)^2 + (y-5)^5 - 4$, on a bien, en x^*,

$$-\nabla f(x^*) \in \{\lambda \nabla h_1(x^*) : \lambda \geq 0\}$$

où $\{\lambda \nabla h_1(x^*) : \lambda \geq 0\}$ est le cône normal à K en x^* (c'est le cône normal au cône tangent à K en x^*). En effet, $\nabla f(x^*) = (2x, 2y) = (8, 8)$ et $\nabla h_1(x^*) = (2(x-5), 2(y-5)) = -(2, 2)$ pour $x^* = (x, y) = (4, 4)$. ■

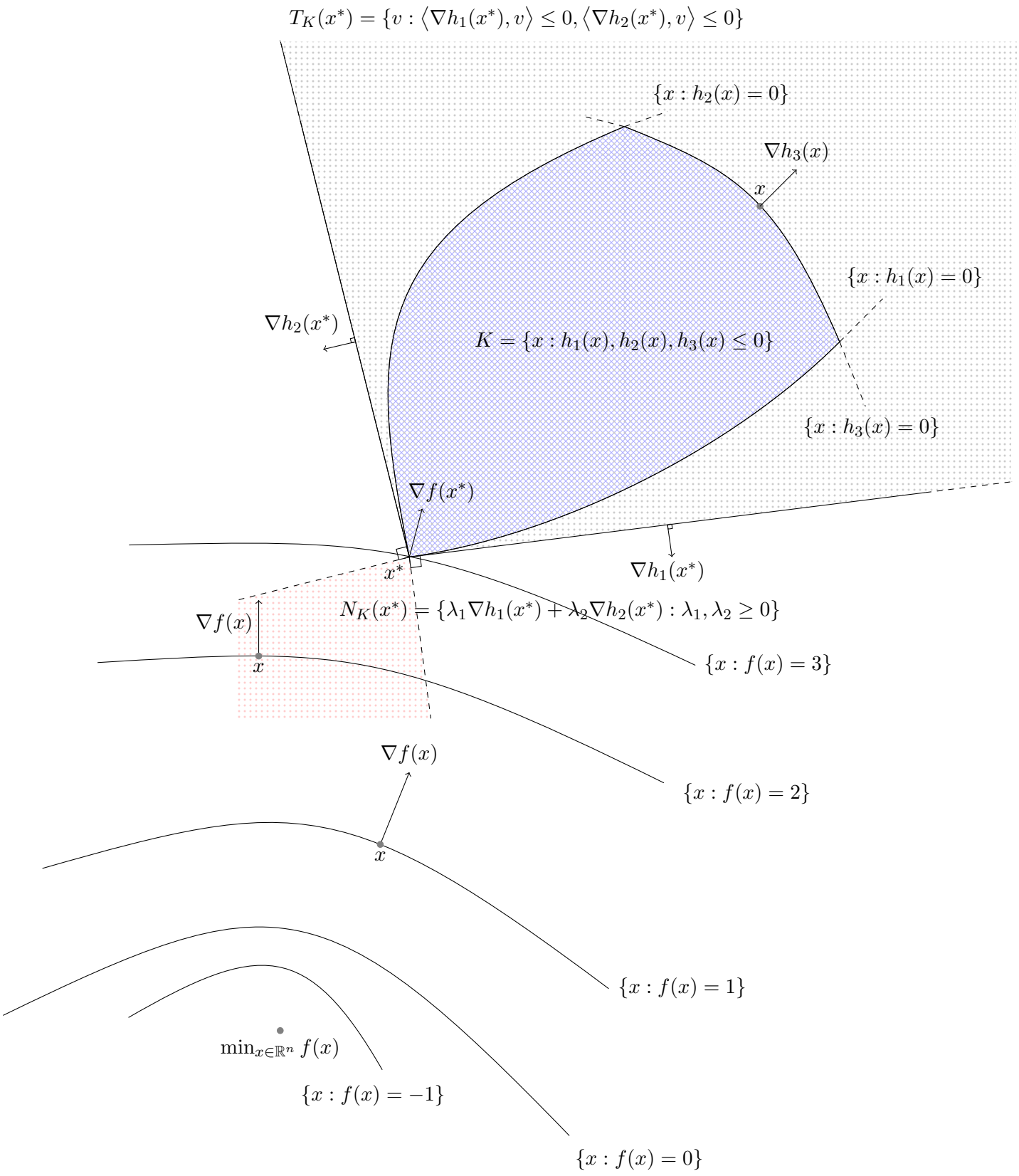

FIGURE 5 – Si x^* est solution du problème d'optimisation (1.1) alors $-\nabla f(x^*) \in N_K(x^*)$. C'est ce que dit le théorème d'Euler/Péano/Kantorovitch. Si de plus la contrainte K est qualifiée en x^* alors le cône normal à K en x^* admet une description simple donnée en (3.5) et on retrouve KKT.

En résumé : Il existe une multitude de problèmes concrets qui s'énoncent sous forme d'un problème d'optimisation comme en (1.1). Pour résoudre, ce problème d'un point de vue mathématiques, on aura besoin d'outils tels que le gradient d'une fonction, le cône tangent et le cône normal à une contrainte K, la condition d'Euler/Péano/Kantorovitch et d'étudier l'hypothèse de qualification qui donne une description simple des cônes tangents et normaux à une contraintes. Dans ce cas, la condition d'Euler/Péano/Kantorovitch est appelée la condition KKT, qui est la condition la plus utilisée en optimisation. On verra aussi que la condition d'Euler/Péano/Kantorovitch qui est une condition sur les cônes normaux de la ligne de niveau $f(x^*)$ de f et de K en x^* peut aussi se retrouver par une autre dualité appelée dualité Lagrangienne. C'est le programme des 5 ou 6 prochains cours.