Université de Montpellier
Département de Mathématiques

Optimisation et apprentissage mathématique

Livret questions de cours et exercices

Pr. Bijan Mohammadi

Ce cours porte sur les problèmes d'optimisation, avec et sans contrainte, linéaires et nonlinéaires, avec ou sans gradient, très présents en industries et services. On s'intéresse, par ailleurs, à l'utilisation des techniques d'optimisation en apprentissage mathématique (machine learning). On abordera enfin les questions autour de l'évaluation du gradient en continu et en discret par différentiation automatique en modes direct et inverse.

Les éléments théoriques du cours, ainsi que des exemples de programmes, sont disponibles sur mon site.

1 Espace de fonctions, linéarité, continuité, norme

Soit $E = C([0,1], \mathbb{R})$ muni de la norme $\|.\|_\infty$ et $T : E \to E$ telle que :

$$\forall f \in E, \quad \forall x \in [0,1], \quad (T(f))(x) = \int_0^x f(t)dt.$$

Vérifier que T est bien définie sur E et est à valeur dans E.
Montrer que T est linéaire continue et que

$$|||T||| = \sup_{f \in E, f \neq 0_E} \frac{\|T(f)\|_\infty}{\|f\|_\infty} = 1.$$

Démarche : d'abord majorer $||T(f)||_\infty$, puis trouver une fonction (e.g. $f = 1$) pour laquelle la majoration est optimale, d'où l'égalité.

2 Espace de fonctions, linéarité, continuité, norme

Soit $E = C([0,1], \mathbb{R})$ muni de la norme $\|.\|_\infty$ et $T : E \to \mathbb{R}$ telle que :

$$\forall f \in E, \quad (T(f)) = f(1) - 2f(0).$$

Vérifier que T est bien définie sur E et est à valeur dans $\mathbb{R}$.
Montrer que T est linéaire continue et que $|||T||| = 3$.
Démarche : d'abord majorer $||T(f)||_\infty$, puis trouver une fonction (e.g. $f = 2x-1$) pour laquelle la majoration est optimale, d'où l'égalité.

3 Conditions d'optimalité

On considère l'espace euclidien $E = \mathbb{R}^n$ muni du produit scalaire canonique. Soit $y \in E$ un vecteur fixé non nul et f la fonction de E dans $\mathbb{R}$ définie par :

$$\forall x \in E, \quad f(x) = < x, y > \exp(-\|x\|^2).$$

1. Calculer les dérivées partielles de f et en déduire son gradient.
2. Montrer que si x est un point critique de f alors x est colinéaire à y, puis déterminer les points critiques de f.
3. Tracer la fonctionnelle en 2 dimension.
4. Etudier les conditions d'optimalité d'ordre 2 et montrer qu'il y a 2 points critiques ($+/- y/(\sqrt{2}\|y\|)$) de nature différente.

4 Conditions d'optimalité

Soit $f : \mathbb{R}^2 \to \mathbb{R}$ définie par $f(x,y) = (1-x)(1-y)(x+y-1)$.
1. Déterminer les points critiques de f.
2. Donner la matrice hessienne H_f.
3. Analyser les points critiques de f.
4. Quels sont les extremums globaux de f sur $\mathbb{R}^2$?

5 Conditions d'optimalité

Soit $f : \mathbb{R}^2 \to \mathbb{R}$ définie par $f(x, y) = 2x^3 + 6xy - 3y^2 + 2$.
Mêmes questions que (4).

6 Minimisation et barycentre

Soient $p_1, ..., p_n$, n points du plan $\mathbb{R}^2$. On définit la fonction $f : \mathbb{R}^2 \to \mathbb{R}$ par :

$$\forall x \in \mathbb{R}^2, \quad f(x) = \sum_{i=1}^{n} \|x - p_i\|^2,$$

où $\|.\|$ est la norme euclidienne usuelle. Montrer que f admet un minimum unique sur $\mathbb{R}^2$ en un point que l'on déterminera. Interpréter le résultat en termes géométriques.

7 Ensembles convexes

Soit l'ensemble $C = \{x \in \mathbb{R}^3; x_1 \geqslant 0, x_2 \geqslant 0, x_3 \geqslant 0, x_1 + x_2 + x_3 = 1\}$.
Dessiner C et montrer que C est convexe (utiliser les propriétés des ensembles convexes).
Commencez par étudier le cas 2D avec $C = \{x \in \mathbb{R}^2; x_1 \geqslant 0, x_2 \geqslant 0, x_1 + x_2 = 1\}$.

8 Minimisation sur un ensemble

Soit $C = \{(x, y) \in \mathbb{R}^2 : (x^2 - 1)^2 + y^2 \leqslant 4\} \subset \mathbb{R}^2$.
a. Ecrire un programme pour tracer la frontière de C.
b. S'agit-il d'un ensemble compact ?
c. S'agit-il d'un ensemble convexe ?
d. Considérer $J(x, y) = xy$.
Est-ce que J admet un minimum, un maximum sur C ?
e. Calculer $\inf\{J(x, y) : (x, y) \in C\}$.

9 Minimisation sur un ensemble

On considère $f(x, y) = -(\log(x) + \log(y) + \log(1 - x) + \log(1 - y))$ pour $(x, y) \in \Omega \subset \mathbb{R}^2$.
1. Déterminer et représenter Ω.
2. Calculer le gradient et le hessien de f.
3. Pourquoi f admet un minimum local strict ? Déterminer le.
On définit g sur Ω par $g(x, y) = \exp(-f(x, y))$. Justifier l'existence d'un maximum pour g et déterminer sa valeur.

10 Minimisation sous contraintes d'égalité, Lagrangien

Minimiser $J(x_1, x_2) = x_1^3 x_2$, sous la contrainte $g(x_1, x_2) = x_1^3 - x_2 - 1 = 0$.
D'abord, en se ramenant à la minimisation sans contrainte d'une fonction d'une variable, puis, par la méthode des multiplicateurs de Lagrange. Quelle est la sensibilité de J par rapport aux petites variations de la contrainte g à l'optimum.

11 Minimisation sous contraintes d'égalité, Lagrangien

Résoudre

$$\min_{x\in\mathbb{R}^3} J(x) = -x_1x_2 - x_1x_3 - x_2x_3, \quad \text{avec} \quad h(x) = x_1 + x_2 + x_3 - 3 = 0.$$

Quelle est la sensibilité de J par rapport aux petites variations de la contrainte h à l'optimum. Quel est le lien avec le multiplicateur de Lagrange ?

12 Minimisation sous contraintes d'égalité, Lagrangien

Résoudre

$$\min_{x\in\mathbb{R}^3} J(x) = x_1^2 - 4x_1 - x_1x_2 + x_1x_3 + x_2x_3, \quad \text{avec} \quad h(x) = x_1 + x_2 + x_3 - 1 = 0.$$

Que peut-on dire sur la contrainte à l'optimum ?

13 Projection sur un convexe fermé

Soit E un espace vectoriel. On appelle cône un sous-ensemble C de E tel que :

$$\forall x \in C, \quad \forall \lambda \geqslant 0, \quad \lambda x \in C.$$

1. Donner deux exemples de cônes dans $E = \mathbb{R}^2$, tels que l'un soit convexe et l'autre non.

2. On suppose que E est un espace de Hilbert. Soit C un cône convexe fermé de E et $y = \pi_C(x)$ la projection de x sur C. Montrer que $< y, y - x >= 0$ (utiliser la caractérisation de la projection sur un convexe).

14 Minimisation et projection

Soit $K = \{y = (y_1, y_2) \in \mathbb{R}^2; \alpha \leqslant y_1 \leqslant \beta, \quad \delta \leqslant y_2 \leqslant \gamma\}$, avec $\alpha < \beta$ et $\delta < \gamma$ des réels. On fixe $x = (x_1, x_2) \in \mathbb{R}^2$.

1. Quel est le nombre de solutions de $\min_{y\in K} \|x - y\|_2$?

Construire $y^* = P_k(x)$ vérifiant $\|x - P_K(x)\|_2 = min_{y\in K}\|x - y\|_2$. Exprimer y^* en fonction de x.

Faire un schéma pour $\alpha = \delta = 1$, $\beta = 3$ et $\gamma = 2$, $x = (4, 3)$.

2. Que peut-on dire du nombre de solutions de $\min_{y\in K} \|x - y\|_\infty$ dans le cas $\alpha = \delta = -1$, $\beta = \gamma = 1$, $x = (0, 2)$? (attention : $(\mathbb{R}^2, \|.\|_\infty)$ n'est pas un espace de Hilbert).

15 Minimisation et projection

On cherche à calculer la projection d'un point $y \in \mathbb{R}^2$ sur $C = \{x \in \mathbb{R}^2; x_1^2 \leqslant x_2\}$ en utilisant le théorème de projection sur un convexe.

1. Représenter graphiquement C.

2. Soit $g : E \to \mathbb{R}$ une fonction, où E est une espace vectoriel, et $C_g = \{x \in E; g(x) \leqslant 0\}$. Montrer que si g est convexe alors C_g est convexe. En déduire que C est convexe.

3. Montrer que $\forall y \in \mathbb{R}^2$, le problème de projection sur C admet une solution unique y^* et écrire ce problème sous la forme d'un problème d'optimisation sous contrainte inégalité d'une fonctionnelle J.

16 Minimisation sous contraintes d'égalité

Pour $x = (x_1, x_2) \in \mathbb{R}^2$, on pose $q(x) = x_1^2 + 2x_1x_2 + px_2^2$, $p \in \mathbb{R}$, $J(x) = x_1 + x_2$ et on considère l'ensemble $S = \{x \in \mathbb{R}^2; q(x) = 1\}$.

1. Montrer que q est une forme quadratique (expliciter la forme bilinéaire b telle que $b(x, x) = q(x)$). Déterminer la matrice Φ de b dans la base canonique de $\mathbb{R}^2$. Sans calculer ses valeurs propres, montrer que q est définie positive ssi $p > 1$.

Pour la suite, on prend $p = 2$.

2. Déterminer le produit scalaire $[.|.]$ associé à q.
3. Déterminer $\beta \in \mathbb{R}^2$ tel que $J(x) = [x, \beta]$ pour tout $x \in \mathbb{R}^2$.
4. Minimiser et maximiser J sur S.

17 Minimisation sous contraintes d'égalité

Pour $x = (x_1, x_2, x_3) \in \mathbb{R}^3$, on pose $q(x) = x_1^2 + x_2^2 + x_3^2 + x_1x_2$, $J(x) = x_1 + x_2 + x_3$ et on considère l'ensemble $S = \{x \in \mathbb{R}^3; q(x) = 1\}$.

1. Montrer que q est une forme quadratique définie positive.
2. Déterminer le produit scalaire $[.|.]$ associé et écrire la matrice associée dans la base canonique.
3. Déterminer $\alpha \in \mathbb{R}^3$ tel que $J(x) = [\alpha, x]$.
4. Est-ce que J atteint ses bornes sur S ?
5. Déterminer $\min_{x \in S} J(x)$ et $\max_{x \in S} J(x)$ et les points où les extremums sont atteints.

18 Projection et représentation

Sur l'espace $\mathbb{R}_3[X]$ des polynômes à coefficients réels de degré ingérieur ou égal à 3, on considère la forme bilinéaire définie par :

$$< P, Q >= \frac{1}{2} \int_{-1}^{1} P(t)Q(t)dt.$$

1. Montrer que $< P, Q >$ définit un produit scalaire.
2. Donner la dimension de $\mathbb{R}_3[X]$.
3. Déterminer des réels a, b, c, d, e, f tels que $Q_0(X) = a, Q_1(X) = bX + c, Q_2(X) = dX^2 + eX + f$ forment une base orthonormée de $\mathbb{R}_2[X]$ muni du produit scalaire précédemment défini.

19 Projection

Soit $(E, <, >)$ un espace euclidien et soit $\|.\|$ la norme associée au produit scalaire.

1. Soient u et v deux vecteurs orthogonaux de E, montrer que $\|u + v\|^2 = \|u\|^2 + \|v\|^2$.
2. Soit F un sev de E et p l'application linéaire de E sur F définie par :

$$\forall u \in E, p(u) \in F, \qquad et \qquad \forall v \in F, < u - p(u), v >= 0.$$

a. Vérifier que $\forall u \in E$ et $\forall v \in F$, $u - p(u)$ et $v - p(u)$ sont orthogonaux.

b. En utilisant 1., montrer que $\forall u \in E$ et $\forall v \in F$, $\|u - v\|^2 \geqslant \|u - p(u)\|^2$.

c. Soit $u \in E$, on appelle distance de u à F la quantité $dist(u, F) = \inf_{v \in F} \|u - v\|$. Montrer que $dist(u, F) = \|u - p(u)\|$.

20 Projection

Nous allons appliquer les résultats de 19 pour $E = \mathbb{R}_3[X]$ et $F = \mathbb{R}_2[X]$ définis en 18.

La projection orthogonale $p(Q)$ de Q sur le sous-espace $\mathbb{R}_2[X]$ est définie par : si (Q_0, Q_1, Q_2) est une base orthonormée de $\mathbb{R}_2[X]$, alors $p(Q) = \sum_{i=0}^{2} < Q, Q_i > Q_i$.

1. Calculer $< X^3, Q_0 >$, $< X^3, Q_1 >$ et $< X^3, Q_2 >$, où $Q_0(X) = 1$, $Q_1(X) = \sqrt{3}X$ et $Q_2(X) = -\frac{3\sqrt{5}}{2}X^2 + \frac{\sqrt{5}}{2}$. En déduire $p(X^3)$.

2. Calculer $\|X^3 - p(X^3)\|^2$.

3. En déduire que $\text{dist}(X^3, F) = \frac{2}{5\sqrt{7}}$.

21 Réduction dimensionnelle et minimisation sur un convexe

Lorsque la dimension du problème est grand, et en particulier en présence de contraintes industrielles et lors de phases initiales de conception (avant-projet), il est intéressant de pouvoir travailler, non pas sur l'espace entier des paramètres, mais sur un sous-ensemble convexe (simplexe) défini par des configurations déjà connues et réalisables. Considérons la minimisation d'une fonctionnelle $J(x)$ avec la variable d'optimisation x appartenant à $\mathcal{O}$ un ensemble convexe de $R^N, N >> 1$ qu'on appelle l'ensemble admissible :

$$\min_{x \in \mathcal{O} \subset \mathbb{R}^N} J(x).$$

J représente par exemple la consommation d'un avion ou sa traînée et x sa forme décrite par $N = 1000$ paramètres.

Introduisons des configurations connues $X_{\{i=1,...,n\}}$ avec $n << N$. Dans notre exemple, chaque X_i est une forme admissible de $\mathcal{O} \subset \mathbb{R}^N$. Les X_i permettent aussi d'introduire l'expertise métier lors de la conception. Introduisons le convexe $\mathcal{S}$:

$$\mathcal{S} = \left\{ \sum_{i=1}^{n} \lambda_i X_i \;\middle|\; \sum_{i=1}^{n} \lambda_i = 1 \text{ et } \lambda_i \geqslant 0 \;\; \forall i \right\} \subset \mathcal{O}.$$

Les $\lambda_i \in [0, 1]$ sont les coordonnées barycentriques et les X_i les sommets du simplexe $\mathcal{S}$. Ainsi, remplacer le problème de minimisation en x par celui en $\Lambda \in \mathcal{L} = \{(\lambda_1, ..., \lambda_n) \mid \sum_{i=0}^{n} \lambda_i = 1 \text{ et } \lambda_i \geqslant 0 \;\; \forall i\}$, permet de fortement réduire la dimension du problème :

$$\min_{\Lambda \in \mathcal{L}} J(x(\Lambda)).$$

A l'optimum, la solution $x(\Lambda^*) \in \mathcal{S} \subset \mathcal{O}$ sera donc nécessairement admissible et sera la projection de x^*, la solution du problème initiale, sur le convexe $\mathcal{S}$. Et, nous avons d'après le théorème de caractérisation de la projection sur les convexes :

$$< x^* - p(x^*), y - p(x^*) > = < x^* - x(\Lambda^*), y - x(\Lambda^*) > \leqslant 0, \quad \forall y \in \mathcal{S}.$$

Cette reformulation du problème a aussi d'autres avantages pratiques du fait de la faible dimension n par rapport à N :

- le calcul de gradient de la fonctionnelle peut se faire par différences finies,
- on peut facilement analyser la sensibilité aux perturbations des paramètres et estimer la robustesse de la conception (on parle de quantification des incertitudes).

Pour optimiser la complexité calculatoire, nous avons intérêt de commencer par n le plus petit possible (càd $n = 2$) et enrichir les configurations 'métier' X_i de façon itérative si la convergence

n'est pas satisfaisante. Il faut s'assurer cependant que la nouvelle configuration $X_{n+1} \notin \mathcal{S}$. Par exemple, en vérifiant que :

$$< x^* - x(\Lambda^*), X_{n+1} - x(\Lambda^*) > > 0.$$

En pratique, bien sur, x^* est inconnu. Une approximation serait de le remplacer par $x(\Lambda^*) + n(x(\Lambda^*))$ avec $n(x(\Lambda^*))$ la normale externe à $\mathcal{S}$ en $x(\Lambda^*)$. Faire un dessin pour illustrer cette construction. Nous verrons une implémentation de cette idée.

22 Extrémum sur un ensemble

Soit $f(x_1, x_2) = x_2(3x_1^2 - x_2^2 + 1)$.

1. Calculer le gradient et le hessien de f.

2. Déterminer les points critiques de f et analyser les extremums de f.

3. Tracer $B = \{x \in \mathbb{R}^2 / \|x\|_\infty \leqslant 1\}$ et $S = \{x \in \mathbb{R}^2 / \|x\|_\infty = 1\}$.

4. Montrer que la restriction de f à B atteint un maximum en au moins un point $x^* \in B$. Montrer que $x^* \in S$ et calculer x^*.

23 Forme quadratique

Pour $x = (x_1, x_2, x_3) \in \mathbb{R}^3$, on pose $q(x) = x_1^2 + 4x_1x_2 + x_2^2 + x_3^2$.

1. Montrer que q est une forme quadratique (expliciter la forme bilinéaire b telle que $b(x, x) = q(x)$). Déterminer la matrice Φ de b dans la base canonique de $\mathbb{R}^3$.

2. Déterminer les valeurs propres de Φ et une base de vecteurs propres orthonormée pour le produit scalaire euclidien.

3. Montrer que q n'est pas bornée sur $\mathbb{R}^3$.

4. Justifier l'existence d'extrema de q sur la sphère euclidienne $S = \{x \in \mathbb{R}^3; \|x\|^2 = 1\}$ et les déterminer.

24 Problème de Kepler

Inscrire dans l'ellipsoïde $E = \{(x, y, z) \in \mathbb{R}^3 : x^2/a^2 + y^2/b^2 + z^2/c^2 = 1\}$ le parralépipède de volume maximal dont les arêtes sont parallèles aux axes.

25 Problème de Tartaglia

Décomposer 8 en deux parties positives x_1, x_2 telles que le produit de leur produit par leur différence soit maximal :

$$\max_{x_1 \geqslant 0, x_2 \geqslant 0} x_1x_2(x_2 - x_1), \quad x_1 + x_2 = 8.$$

26 Problème de Dido ou iso-périmétrique du calcul des variations

D'après la légende, Dido a fondé Carthage (Tunisie). Lorsqu'elle arrive en 814 avant JC sur la côte tunisienne, elle demande un terrain. Sa demande est acceptée, mais le terrain doit tenir dans une peau de boeuf. Elle découpe la peau en question en une longue bande mince et l'utilise pour encercler un terrain qui devindra Carthage.

Il s'agit de résoudre le problème d'optimisation suivant :

Quelle est la courbe fermée de longueur L donnée qui maximise la surface délimitée A ?

Considérons la courbe paramétrée fermée $C = (x(t), y(t))$ et la vitesse le long de la courbe $(\dot{x}(t), \dot{y}(t))$. Nous allons calculer sa longueur et la surface qu'elle délimite.

La longueur provient de l'intégration de l'élément de longueur infinitésimal $ds^2 = dx^2 + dy^2$ qui permet d'écrire $ds = \sqrt{\dot{x}^2 + \dot{y}^2}dt$. Ce qui donne $L = \oint \sqrt{\dot{x}^2 + \dot{y}^2}dt$.

Pour le calcul de la surface, nous allons utiliser la formule de Green dans le plan sur un domaine D de frontière ∂D :

$\int_{\partial D} f dx + g dy = \int_D \left(\frac{\partial g}{\partial x} - \frac{\partial f}{\partial y}\right) dxdy.$

En prenant $f = -\frac{y}{2}$ et $g = \frac{x}{2}$, on a $\frac{1}{2}\int_{\partial D} xdy - ydx = \int_D dxdy = A$.

Ainsi, la surface A est donnée par :

$A = \frac{1}{2}\oint (x\dot{y} - y\dot{x})dt.$

Il s'agit donc de résoudre un problème d'optimisation sous contrainte d'égalité :

$$\max_C A(C), \quad \text{sous la contrainte} \quad L(C) = p.$$

Nous allons utiliser le résultat suivant.

Considérons le problème : Maximiser $\oint F(t, z, \dot{z})dt$ sous la contrainte $\oint G(t, z, \dot{z})dt = p$.

Introduisons $\Lambda(t, z, \dot{z}) = F(t, z, \dot{z}) + \lambda G(t, z, \dot{z})$.

La solution du problème vérifie l'équation d'Euler-Lagrange :

$$\frac{d}{dt}\frac{\partial \Lambda}{\partial \dot{z}} - \frac{\partial \Lambda}{\partial z} = 0.$$

Appliquer au problème de Dido la démarche donne (avec $z = (x, y)$) :

$\Lambda = \frac{1}{2}(x\dot{y} - y\dot{x}) + \lambda\sqrt{\dot{x}^2 + \dot{y}^2}$

Nous avons deux équations d'Euler-Lagrange pour x et y :

$$\frac{d}{dt}\frac{\partial \Lambda}{\partial \dot{x}} - \frac{\partial \Lambda}{\partial x} = 0 \quad \text{et} \quad \frac{d}{dt}\frac{\partial \Lambda}{\partial \dot{y}} - \frac{\partial \Lambda}{\partial y} = 0.$$

En dérivant en $(x, \dot{x}, y, \dot{y})$, on a :

$$\frac{d}{dt}\left(-\frac{1}{2}y + \frac{\lambda\dot{x}}{\sqrt{\dot{x}^2 + \dot{y}^2}}\right) - \frac{1}{2}\dot{y} = 0 \quad \text{et} \quad \frac{d}{dt}\left(\frac{1}{2}x + \frac{\lambda\dot{y}}{\sqrt{\dot{x}^2 + \dot{y}^2}}\right) + \frac{1}{2}\dot{x} = 0.$$

En intégrant en t, on obtient :

$$y - \frac{\lambda\dot{x}}{\sqrt{\dot{x}^2 + \dot{y}^2}} = b \quad \text{et} \quad x + \frac{\lambda\dot{y}}{\sqrt{\dot{x}^2 + \dot{y}^2}} = a.$$

Ou encore,

$$y - b = \frac{\lambda\dot{x}}{\sqrt{\dot{x}^2 + \dot{y}^2}} \quad \text{et} \quad x - a = -\frac{\lambda\dot{y}}{\sqrt{\dot{x}^2 + \dot{y}^2}}.$$

Au carré et après sommation, on a :

$(x - a)^2 + (y - b)^2 = \lambda^2$.

C'est le cercle de centre (a, b) et de rayon λ. Le périmètre étant p, on déduit $\lambda = \frac{p}{2\pi}$. La courbe optimale est le cercle de rayon $\frac{p}{2\pi}$ et de surface interne $\frac{p^2}{4\pi}$.

Cette approche permet le calcul des trajectoires optimaux. Un exemple classique sans contrainte est le problème de brachistochrone où on cherche à trouver la courbe $y = f(x)$ reliant un point $A = (x_A, y_A)$ à un point $B = (x_B, y_B)$, situé à une altitude plus faible ($y_B < y_A$), tel qu'un point matériel partant du point A sans vitesse initiale et glissant sans frottement sur la courbe rejoigne le plus rapidement possible le point B, càd qui présente un temps de parcours minimal entre deux points donnés. Aussi, on retrouve ainsi le principe de Fermat de l'optique géométrique.

En général, on ne connaît pas de solution exacte et il faut exhiber une solution numérique.

Comment procéder pour écrire un programme informatique pour résoudre ce type de problèmes, notamment en présence de contraintes supplémentaires ?

27 Minimisation et analyse spectrale

Soit A une matrice symétrique réelle. Montrer que le problème :

$$\max_{\|x\|^2 = x^t x \leqslant 1} x^t A x,$$

admet une solution et interprêter cette solution pour la matrice A ?

Programmer les méthodes de pénalisation, de gradient projeté et d'Uzawa pour ce problème. Partir du programme développé dans l'exercice 35. Commencer par un exemple en une seule variable : $\min_{x \in \mathbb{R}} -ax^2, \quad |x| \leqslant 1$.

28 Une autre caractérisation des extremums

Soient $A \subset \mathbb{R}^n$, $a \in A$ et $f \in C^1(\mathbb{R}^n, \mathbb{R})$. On suppose que a est un extrémum local de f sur A.

Soit $I \subset \mathbb{R}$, $0 \in I$. On considère une courbe paramétrée $X : I \to A$ vérifiant $X \in C^1(I, A)$ et $X(0) = a$.

1. Montrer que $\nabla f(a)$ est perpendiculaire au vecteur tangent $X'(0)$.

Soit Q une matrice d'ordre n, symétrique définie positive, $b \in \mathbb{R}^n$, $c \in \mathbb{R}$ et $f(x) = \frac{1}{2} x^t Q x + b^t x + c$.

2. Montrer que $\lim_{\|x\| \to +\infty} f(x) = +\infty$.
3. Montrer que f admet au moins un minimum sur $\mathbb{R}^n$.

Posons $n = 2, c = 0, b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $Q = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ et $A = \{x = (x_1, x_2); x_1^2 + x_2^2 = 1\}$.

4. Ecrire les équations vérifiées par un extrémum local $a \in A$ de f sur A.
5. Montrer que f admet au moins un minimum sur A. Trouver les minimas.

29 Multi-critère et géométrie

Soit f, h_1 et h_2, 3 fonctions de $\mathbb{R}^3$ dans $\mathbb{R}$ définies par $(\forall x = (x_1, x_2, x_3) \in \mathbb{R}^3)$:

$$f(x) = x_1 - x_2 x_3, \quad h_1(x) = x_1^2 + x_2^2 - 1, \quad h_2(x) = x_1^2 + x_3^2 - 1,$$

et les ensembles

$$C_1 = \{x \in \mathbb{R}^3, h_1(x) = 0\}, \quad C_2 = \{x \in \mathbb{R}^3, h_2(x) = 0\}, \quad C = C_1 \cap C_2.$$

On souhaite résoudre

$$(P) : \min_{x \in C} f(x).$$

1. Quelles sont les formes géométriques de C_1 et C_2 ? Dessiner ces deux ensembles sur un même graphique. Montrer que C est fermé borné.

2. Montrer que f admet un minimum sur C.

3. Ecrire les conditions d'optimalité pour la minimisation de f sur C et en déduire la ou les solutions de (P).

30 Méthode de la plus forte pente

Montrer que dans une méthode de la plus forte pente à pas optimal, deux directions consécutives sont orthogonales.

31 Exemple fondamental

Soit A une matrice symétrique positive de rang n, montrer qu'il y a équivalence entre les trois problèmes suivants :

1. Trouver $x \in \mathbb{R}^n$ tel que $Ax = b$.
2. Trouver $x \in \mathbb{R}^n$ tel que $(Ax, y) = (B, y), \quad \forall y \in \mathbb{R}^n$.
3. Trouver $x \in \mathbb{R}^n$ tel que $J(x) = \frac{1}{2}(Ax, x) - (b, x)$ soit minimal.

32 Orthogonalité des descentes et pas optimal

Montrer qu'appliquée à la solution de (31), à chaque itération, le pas de la méthode de la plus forte pente à pas optimal est donné par :

$$\rho_k = \frac{(\nabla J(x_k), \nabla J(x_k))}{(A\nabla J(x_k), \nabla J(x_k))}$$

33 Recherche linéaire du pas optimal

Pour les fonctionnelles quelconques, on ne connaît pas d'expression comme celle de (32) pour le pas optimal. On cherche le pas numériquement comme solution d'un problème d'optimisation, appelé recherche linéaire car le long de la direction de descente (par exemple $d_k = -\nabla J(x_k)$) :

$$\rho_k = \underset{\rho > 0}{\operatorname{argmin}}\ J(x_k + \rho d_k).$$

Ainsi, on doit résoudre un problème d'optimisation supplémentaire à chaque itération de la méthode de descente considérée.

On utilise le plus souvent des algorithmes de recherche locale (dichotomie, section dorée, secante, et surtout backtraking, etc) qui peuvent alourdir le calcul.

Nous verrons des exemples d'heuristiques pour réduire le coût calculatoire de ces opérations en minimisant en recherchant une décroissance locale suffisante et non pas optimale.

Nous verrons aussi les critères de Wolfe, qui donnent un cadre pour le choix d'un pas ρ_k sous-optimal :

$$J(x_k + \rho_k d_k) \leqslant J(x_k) + c_1 \rho_k\ (d_k, \nabla J(x_k)) \quad \text{condition d'Armijo}$$

$$|(d_k, \nabla J(x_k + \rho_k d_k))| \leqslant c_2 \, |(d_k, \nabla J(x_k))| \quad \text{condition de courbure}$$

avec $0 < c_1 < c_2 < 1$ dont le choix dépend de la méthode de minimisation choisie (par ex. on prend $c_1 = 10^{-4}$ et $c_2 = 0.1$ pour un gradient conjugué).

Ces critères assurent la convergence vers un point critique ($\nabla J(x) = 0$) si la direction de descente d_k vérifie le critère d'angle :

$$0 < |\cos\theta_k| = \frac{|(\nabla J(x_k), d_k)|}{\|\nabla J(x_k)\|\|d_k\|}$$

En plus de la difficulté de choix des constantes c_1 et c_2, les critères de Wolfe restent coûteux car demandent l'évaluation de $\nabla J(x_k + \rho_k d_k)$. Nous verrons des heuristiques permettant d'éviter ces évaluations.

Vérifier les conditions de Wolfe dans le cas des fonctionnelles de (32) pour la méthode de gradient.

La recherche du pas optimal et les régions de confiance sont deux méthodes classiques garantissant la convergence des algorithmes de minimisation. Nous verrons le concept des régions de confiance en (58) pour la mise en oeuvre de minimisation d'une fonctionnelle sans en connaître le gradient.

En machine learning, le pas de descente s'appelle le taux d'apprentissage. Aucun cadre théorique n'existe vraiment pour les problèmes d'apprentissage profond car les fonctionnelles ne vérifient pas les hypothèses telles que (stricte)-convexité. Le choix du taux d'apprentissage reste alors essentiellement empirique (voir (38) pour la méthode du gradient stochastique).

34 Gradient conjugué

Soit $\{d_1, ..., d_n\}$ une famille génératrice de $\mathbb{R}^n$ (base) de vecteurs 2 à 2 A-conjugués ($d_i A d_j = \delta_{ij}$, càd les d_i sont orthogonales pour le produit scalaire $(.,.)_A$).

- Montrer que la fonctionnelle du (31) s'écrit :

$$J(x) = \sum_{k=1}^{n} \frac{1}{2}\rho_k^2 d_k^t A d_k - \rho_k d_k^t b, \quad \rho_k \in \mathbb{R}.$$

Que peut-on dire de cette expression et comment choisir ρ_k pour minimiser J ?

- En raisonnant de façon constructive : $x_{k-1} = \sum_{i=1}^{k-1} \rho_i d_i$, montrer que $d_k^t r_{k-1} = d_k^t b$ où $r_{k-1} = b - A x_{k-1}$ est le résidu à l'étape $k-1$ (aussi le gradient de J ici) et que $r_k = r_{k-1} + \rho_k A d_k$.

Si on construit la nouvelle direction d_{k+1} sous la forme $d_{k+1} = r_k - \beta_k d_k$ (combinaison du gradient et de la précédente direction), $\beta_k = d_k^t A r_k / (d_k^t A d_k)$ permet d'assurer l'A-orthogonalité : $d_k A d_{k+1} = 0$.

Ces deux points constituent les deux idées fondamentales de l'algorithme du gradient conjugué que nous allons utilisé dans l'exemple (35).

35 Résolution numérique de l'exemple fondamental

Nous allons utiliser l'équivalence des propositions dans (31) pour résoudre numériquement le système linéaire $AX = b$ avec A la matrice de Hilbert ($A_{ij} = 1/(i + j - 1)$) et $b_i = \sum_{j=1}^{n} A_{ij}$, $i = 1, ..., n$ pour $n = 100, 1000, 10000, 100000$.

Sans calcul : la solution de ce système est évidemment $x = (1, ..., 1)$.

Pour retrouver cette solution numériquement, nous utiliserons :

- une méthode directe (e.g. LU),

- la méthode de gradient avec le pas optimal calculé dans (32). Vérifier numériquement la propriété démontrée en (30).

- la méthode de gradient conjugué de (34). Vérifier numériquement la propriété d'A-orthogonalité pour les directions d_k.

- la méthode de Newton (que peut-on dire dans ce cas ?)

Que peut-on dire sur les temps de calcul, la vitesse de convergence ?

36 Sous-espace de Krylov

Dans l'exercice (35), expliciter les sous-espace de Krylov $\mathcal{U}_0$, $\mathcal{U}_1$, $\mathcal{U}_2$, $\mathcal{U}_3$ pour $n = 2$ pour une initialisation avec $x_0 = (0, 0)$. Quelle conclusion en tirer ?

37 Régression linéaire et moindres carrés

Utiliser l'équivalence des propositions dans (31) pour montrer que résoudre $Ax = b$ surdéterminé (i.e. plus d'équations que d'inconnus) par moindres carrés revient à résoudre le système des équations normales : $A^tAx = A^tb$.

Expliciter cette construction dans le cas de la droite des moindres carrés.

Programmer un exemple (et comparer le résultat avec `R`).

38 Apprentissage mathématique, fonctionnelles séparables, gradient stochastique, algorithme ADAM, Hadoop

Une fonctionnelle est séparable si le problème d'optimisation associé peut être traité pour chaque dimension de façon indépendante :

$$Argmin_{x_1,...,x_n} J(x_1, ..., x_n) \in \mathbb{R}^n = (Argmin_{x_1} J(x_1, ...), ..., Argmin_{x_n} J(..., x_n)) \in \mathbb{R}^n$$

Les fonctionnelles additives sont des exemples importants de fonctionnelles séparables :

$$J(x_1, ..., x_n) = \sum_{i=1}^{n} j_i(x_i).$$

C'est le cas de la fonctionnelle de Rastrigin, souvent utilisée pour évaluer les algorithmes d'optimisation globale :

$$J(x_1, ..., x_n) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)).$$

Ce cadre est très utile car on est ramené pour chaque problème à une optimisation en une dimension et ces optimisations peuvent être traîtées en parallèle.

Donner d'autres exemples de fonctionnelles séparables et adapter les programmes du cours pour les résoudre tirant avantage de cette séparabilité.

Une autre situation où l'additivité dans la fonctionnelle peut être mise à profit est lorsque la fonctionnelle est une somme d'erreurs entre N mesures $(t_j, y_j)_{j=1,...,N}$ et résultats d'un modèle $(t_j, f(t_j, x))_{j=1,...,N}$. Cette situation est courante en apprentissage mathématique (aussi appelé automatique ou statistique) :

$$J(x) = \sum_{j=1}^{N} (f(t_j, x) - y_j)^2$$

Considérons le problème de régression linéaire par une droite de moindres carrés. La fonctionnelle est additive :

$$J(x_1, x_2) = \sum_{j=1}^{N} ((x_2 t_j + x_1) - y_j)^2$$

La méthode de gradient s'écrit :

$$x_1 \leftarrow x_1 - \rho\ 2 \sum_{j=1}^{N} (x_1 + x_2 t_j - y_j), \tag{1a}$$

$$x_2 \leftarrow x_2 - \rho\ 2\ \sum_{j=1}^{N}\ t_j (x_1 + x_2 t_j - y_j). \tag{1b}$$

Lorsque la quantité de données est grande ou bien si le modèle n'est pas aussi simple à évaluer, une alternative est de considérer des sommes partielles sur des sous-ensembles des indices choisis aléatoirement (`mini-batch`). Ce traitement partiel des données est, particulièrement, indispensable dans le cas d'un stockage distribué de la base de données. En effet, le déplacement des données est bien plus coûteux que le calcul et le stockage de l'ensemble des données en un seul lieu n'est souvent ni possible, ni souhaitable. `Hadoop/MapReduce` de Google est une proposition pour la manipulation de grandes bases de données. Cette méthode de gradient partiel est appelée gradient stochastique. L'incertitude est introduite par le traitement aléatoire et partiel des données. Le pas de descente ρ est appelé taux apprentissage.

Modifier votre code d'optimisation par gradient pour résoudre ce problème de regression linéaire et comparer la convergence du gradient stochastique avec celle de la plus grande pente.

Pour améliorer la convergence de l'algorithme du gradient stochastique, l'algorithme ADAM utilise, comme le gradient conjugué, des informations des itérées précédentes. Il s'adresse, en particulier, aux fonctionnelles avec du bruit et dont le gradient est donc peu fiable. Il est très utilisé en propagation retrograde ou 'backpropagation' de l'erreur entre modèle et données en apprentissage mathématique avec les réseaux de neurones. Nous allons implémenter l'algorithme ADAM, donné par le code `Python` suivant, et comparer sa convergence avec la méthode de gradient. ADAM est considéré en 2019 comme le plus performant pour l'apprentissage des réseaux de neurones.

```
#np.power(vector, pow) = puissance pow élément par élément de vector
rho=0.001,  beta1=0.9, beta2=0.999, epsilon=1.e-8
initialisation: w=w0, m=0, v=0
for t in range(num_iterations):
    g = compute_gradient(x, y)
    m = beta1 * m + (1 - beta1) * g
    v = beta2 * v + (1 - beta2) * np.power(g, 2)
    mhat = m / (1 - np.power(beta1, t))
    vhat = v / (1 - np.power(beta2, t))
    w = w - rho * mhat / (np.power(vhat, 0.5) + epsilon)
```

Le rapport entre la moyenne et l'ecart-type $m/\sqrt{v}$ peut être vu comme une estimation du rapport signal/bruit ou SNR (signal to noise ratio). Dans notre cas, un SNR faible indique la présence de bruit important où la proximité d'un point singulier. Dans les deux cas, l'algorithme réduit le pas de descente. Cela peut être une faiblesse de l'algorithme qui peut converger plus facilement vers des minima locaux qu'une méthode de gradient à pas constant. L'introduction d'une dynamique de second ordre permet d'améliorer la capacité de recherche globale des algorithmes en renforçant l'aspect inertiel des algorithmes. Cet aspect existe déjà dans ADAM qui peut être considéré comme une combinaison de RMSprop et de descente de gradient stochastique (SGD) avec effet mémoire. Il utilise le carré des gradients pour mettre à l'échelle le taux d'apprentissage (le pas de descente) comme RMSprop et utilise l'historique en utilisant la moyenne mobile du gradient au lieu du gradient lui-même comme SGD avec effet mémoire.

38.1 Optimisation globale et inértie

La méthode de descente peut être vue comme une discrétisation par Euler explicit du problème de Cauchy suivant :

$$\dot{x} = -\nabla J(x), \quad x(0) = x_0$$

qui à convergence (i.e. $\dot{x} \sim 0$) atteint le voisinage d'un point critique. On a vu que dans le cadre convexe, ce point critique est l'optimum globale du problème et qu'il y a unicité si la convexité est stricte.

Or, les problèmes d'optimisation peuvent avoir des minima locaux. On a vu la fonctionnelle de Rastrigin plus haut.

En apprentissage statistique en particulier, les problèmes d'optimisation nécessaires à l'identification des paramètres des réseaux de neurones convolutionels comportent beaucoup de minima locaux.

Une idée provenant de la physique et très largement utilisée est d'introduire la notion d'inertie (comme pour une boule pesante) et une vitesse initiale :

$$\eta\ddot{x} + \dot{x} = -\nabla J(x), \quad x(0) = x_0, \quad \dot{x}(0) = x_0'.$$

Les choix de η comme de ρ sont essentiellement empiriques. Cette construction aboutit à la méthode de moment.

Modifier votre code en ce sens et voir l'impact de ce changement sur la minimisation de la fonctionnelle de Rastrigin en dimension 2. Comparer à la méthode de descente à pas fixe.

39 Apprentissage mathématique et décomposition en valeurs singulières

La décomposition en valeurs singulières (Singular Value Decomposition (SVD)) est centrale en apprentissage mathématique. Elle peut recevoir trois interprétation :

-comme une méthode de transformation des variables corrélées en des variables indépendantes,

-comme une méthode d'ordonnancement dimensionnelle mettant en avant en premier les directions où les données sont les plus sensibles,

-comme une méthode de réduction de dimension et de compression d'information.

Singular Value Decomposition et Principal Components Analysis (Analyse en Composantes Principales) sont liées.

Considérons une matrice A_{mn} de n images de taille m. Historiquement, dans les applications, m était souvent bien plus grand que n (par exemple une base de données de 10^3 images d'1 méga pixels RGB donnant $n = 3\ 10^6$. Avec le big data, les choses ont changé et on a des bases de

données de plusieurs millions d'images désormais, voire plus. Il n'est donc pas possible d'assembler la matrice A car 1000 images d'1 méga pixels représentent déjà 12 Giga Octets (1 réel = 4 octets). Les implémentations devront donc être itératives et ne faire appel qu'à des multiplications matrice-vecteur et aux produits scalaire.

La SVD est basée sur la décomposition d'une matrice rectangulaire A sous la forme :

$$A_{mn} = U_{mm} S_{mn} V^t_{nn},$$

où U et V sont des matrices orthogonales et S est rectangulaire avec uniquement des éléments diagonaux non nuls. Cette décomposition existe toujours, même pour les matrices rectangulaires. Dans le cas des matrices carrées, la SVD existe, même si la matrice n'est pas diagonalisable.

On constate que $AA^t = USS^tU^t$. L'orthogonalité de U implique que $AA^tU = USS^t$. Les colonnes de U sont les vecteurs propres de AA^t et les valeurs propres de AA^t sont les éléments diagonaux de la matrice SS^t.

On rappelle que les valeurs propres des matrices symétriques sont réelles et les vecteurs propres orthogonaux.

De même, on constate que $A^tA = VS^tSV^t$ et $A^tAV = VS^tS$. Les colonnes de V sont vecteurs propres de A^tA et les valeurs propres de A^tA sont les éléments diagonaux de la matrice S^tS :

$$A^tAV_i = (S^tS)_{ii} V_i, \quad i = 1, ..., n$$

Les valeurs singulières sont positives et racines carrées des éléments diagonaux de S^tS.

Une autre relation d'importance provient de l'utilisation de l'orthogonalité de V qui permet d'ecrire : $AV = US$. Ceci est intéressant car avec V en main, on obtient $U = AVS^{-1}$ avec S^{-1} définie par $S^{-1}_{ii} = 1/S_{ii}$ si $S_{ii} \neq 0$ et $S^{-1}_{ii} = 0$ sinon.

Ainsi, pour établir la décomposition SVD d'une matrice, nous devons calculer les valeurs et vecteurs propres d'une matrice symétrique, de normaliser ces derniers et de construire les colonnes de U.

En utilisant un logiciel, vérifier la décomposition $A = USV^t$ suivante :

$$A = \begin{pmatrix} 5 & 5 \\ -1 & 7 \end{pmatrix} = \begin{pmatrix} 0 & 1/\sqrt{10} \\ -1 & 2/\sqrt{10} \end{pmatrix} \begin{pmatrix} \sqrt{10} & 0 \\ 0 & 3\sqrt{10} \end{pmatrix} \begin{pmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}$$

- Vérifier que les éléments diagonaux de S sont racines carrées des valeurs propres de A^tA.
- Vérifier que les colonnes de V sont vecteurs propres unitaires de A^tA.
- Vérifier que U et V sont des matrices orthogonales.

Les calculs sont fastidieux à la main. Des logiciels permettent d'accéder à ces décompositions par des méthodes itératives pour de très grands systèmes. En général, on ne calcule que les premières valeurs singulières. Si on n'arrive pas à reconstruire A de façon satisfaisante avec peu de valeurs singulières, les méthodes d'apprentissage pour la classification des images de A ne seront sans doute pas performantes. De même, la compression de données ne sera pas efficace. Cette décomposition est donc centrale dans l'analyse de données.

39.1 Pseudo-inverse

Un autre intérêt de la SVD est son utilisation dans l'inversion des matrices tirant avantage de l'orthogonalité des matrices U et V. On parle de pseudo-inverse : $B = VS^{-1}U^t$.

Le pseudo-inverse de Moore-Penrose $A^+ = (A^tA)^{-1}A^t$ permet de trouver la solution au sens des moindres carrés de l'équation $Ax = b$ quand A a plus de lignes que de colonnes : le système est sur-déterminé (voir aussi 37).

La pseudo-inverse donne un sens à l'inversion grâce aux identités suivantes :

$$ABA = A, \quad BAB = B.$$

En effet, si A est carré et inversible, l'inverse et la pseudo-inverse coincide : $A^{-1} = B$. Mais si A est rectangulaire ou n'est pas inversible, l'inverse n'existe pas, alors que sa pseudo-inverse existe toujours.

Comme exercice, calculer la pseudo-inverse de la matrice A donnée plus haut et la comparer à son inverse.

Maintenant, considérons la matrice :

$$A = \begin{pmatrix} 1 & 2 \end{pmatrix}$$

Montrer que la A se décompose en :

$$A = \begin{pmatrix} 1 \end{pmatrix} \begin{pmatrix} \sqrt{5} & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{5} & -2/\sqrt{5} \\ 2/\sqrt{5} & 1/\sqrt{5} \end{pmatrix}^t$$

Que valent BA et AB ?

40 Apprentissage mathématique et modèles linéaires

Les éléments de ce cours sont à la base des méthodes d'apprentissage mathématique.

Nous allons utiliser les ingrédients précédents pour mettre au point des modèles linéaires explicatifs liant des paramètres $x \in \mathbb{R}^n$ à des données d'observation $y \in \mathbb{R}^m$. Les p scenarii disponibles (différentes observations) de relations entre x et y sont regroupés au sein de bases de données qui peuvent être vues comme deux matrices de variables explicatives $X \in \mathbb{R}^{p \times n}$ et d'observations $Y \in \mathbb{R}^{p \times m}$.

On cherche une relation explicative de la forme :

$$Y = XL + E, \quad L \in \mathbb{R}^{n \times m}.$$

où L (matrice des coefficients du modèle) et E (intercept) sont à estimer. E mesure le degré d'affinité du modèle. Il contient aussi les erreurs ou le bruit. On suppose $E = 0$ pour commencer.

Jusqu'ici, nous connaissions la matrice et le second membre et nous cherchions à trouver la solution d'un système linéaire ($Ax = b$). Ici, on cherche la matrice du système linéaire.

On cherche la solution par moindres carrés (projection L^2) car le problème est, en général, de grande taille et surtout pas carré (souvent nous avons plus d'observations que d'inconnues : $p >> n$). On minimise donc :

$$J(L) = \|Y - XL\|^2 = (Y - XL, Y - XL) = (Y, Y) - 2(X^tY, L) + (X^tXL, L).$$

Si la matrice X est de rang n, X^tX est définie positive. On peut alors utiliser notre résultat fondamental (thm. 14.3.1 du Poly) et retrouver le système des équations normales minimisant $J(L)$:

$$\nabla_L J(L^*) = 0 \Leftrightarrow XX^tL^* = X^tY.$$

La littérature autour de cette méthode est très vaste et elle est très utilisée. L'utilisation de ces modèles est justifiée dans le cadre Gaussien, mais en pratique ils sont utilisés même en dehors de ce cadre.

Optimisation sous contrainte

Pour éviter les problèmes numériques, trouver des solutions robustes et identifier les variables explicatives les plus importantes, on résoud ce problème en ajoutant une contrainte sur L :

$$J(L) = \|Y - XL\|^2 + \alpha\|L\|^q, \quad \alpha > 0.$$

L'expert en analyse de données doit choisir α et q. Les choix les plus classiques pour q sont $q = 1$ (Lasso ou pénalisation L^1), $q = 2$ (Ridge ou pénalisation L^2). Il n'y a pas de choix automatique pour α. Cette approche permet aussi de mettre en évidence les observations les plus significatives. Nous verrons des exemples lors des réalisations informatiques en `Python` en utilisant la bibliothèque `scikit-learn`.

Base de données

Nous allons travailler avec deux bases de données $(X - Y)$ au format `csv` que vous allez lire, analyser et visualiser. Il s'agit notamment de trouver les dimensions n, m des espaces des variables d'entrée x et de sortie ou observation y. Indiquer dans chaque cas le nombre p de scénarios d'apprentissage disponibles. Vérifier que $Rang(X) = n$. Ce cadre est assez générique en analyse de données.

Apprentissage : identifier L

Modifier votre script d'optimisation pour résoudre. Vérifier la sensibilité de votre modèle par rapport aux choix de q et α. Nous verrons aussi des exemples de programmes `python` utilisant la librairie `scikit-learn`.

Incertitude : identifier $E = \mathcal{N}(\mu, \sigma^2), \mu, \sigma \in \mathbb{R}^m$

Dans le développement ci-dessus, nous avons supposé que $E = 0$. Cependant, nous pouvons l'identifier a posteriori (après apprentissage) grace à l'erreur du modèle sur la base de données d'apprentissage. Ceci nous permet de donner une estimation de l'incertitude de notre modèle.

Test : vérifier le comportement du modèle

Considérer des jeux de paramètres explicatifs X_T non inclus dans X. Comparer $Y_{ML} = X_T\ L + E$ et la cible Y_T pour les paramètres X_T. Identifier le meilleur choix pour q.

Comparer votre modèle aux résultats des programmes `python` du cours utilisant la librairie `scikit-learn` :

`https://scikit-learn.org/stable/modules/linear_model.html`

Nous profiterons de ces exercices sur l'apprentissage par machine pour découvrir aussi les algorithmes des k plus proches voisins `K-NN`*, des Séparateurs à Vaste Marge (SVM : exercice 51), de régression logistique (exercice 41) et des forêts aléatoires (Random Forest* `RF`*). Nous les comparerons sur des problèmes de classification et de régression.*

Vous pouvez aussi utiliser `R` si vous préférez.

41 Apprentissage mathématique et Régression Logistique

Beaucoup de problèmes de classification s'intéressent à la question de la probabilité d'un événement $Y(x)$ d'être 1 ou 0 (e.g. appartenant à une certaine classe ou pas). La régression logistique répond naturellement à cette question en modélisant non pas la relation entre les entrées x et la sortie Y, comme dans la régression linéaire, mais directement entre les entrées et cette probabilité.

Considérons un modèle avec deux prédicteurs (variables d'entrée), x_1 et x_2, et une variable de réponse (sortie) binaire (Bernoulli) qui correspond à la probabilité de la sortie Y de valoir 1 : $p = P(Y = 1)$. On considère la relation linéaire suivante [1] :

$$\log_b\left(\frac{P(Y=1)}{P(Y=0)}\right) = \log_b\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \log_b\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

la base b du logarithme est un paramètre libre à optimiser en pratique (dans la suite on prend $b = e$ pour simplification). Comme dans les modèles linéaires, β_0 est l'intercept.

On a donc :

$$\frac{p}{1-p} = e^{\beta_0+\beta_1 x_1+\beta_2 x_2},$$

et ainsi, la probabilité d'avoir $Y = 1$ est :

$$p = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{e^{\beta_0+\beta_1 x_1+\beta_2 x_2}+1} = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2)}}.$$

Ainsi, une fois les β_i identifiés, nous pouvons estimer la probabilité p pour une nouvelle entrée de produire 1. Cette construction s'étend naturellement aux dimensions supérieures. Les β_i sont trouvés en résolvant un problème d'optimisation de moindres carrés issu d'une maximisation de vraisemblance de cette régression (voir exercice 42).

42 Comment éviter le calcul du Hessien ?

Une difficulté avec la méthode de Newton est le calcul du hessien de la fonctionnelle. La méthode de Gauss-Newton permet une alternative pour la résolution de problème où la fonctionnelle est de la forme :

$$j(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x) = \frac{1}{2}\|r(x)\|^2,$$

où l'on suppose que $r(x) : \mathbb{R}^n \to \mathbb{R}^m$ est de classe C^2.

1. Si $J(x) = Dr(x)$ est le jacobien de r, donner le gradient et le hessien de j.

2. En négligeant les termes d'ordre 2 du hessien, montrer que l'itération k de la méthode de Newton implique la résolution de :

$$J(x_k)^t J(x_k) d_k = -J(x_k)^t r(x_k),$$

pour l'obtention de la direction de descente d_k.

3. Quelle condition doit vérifier J pour que $J^t J$ soit inversible ? Quelle conséquence pour m ?

4. Montrer que si $J^t J$ est inversible, alors d_k est une direction de descente.

La méthode de Levenberg-Marquart est une extension de cette méthode et est utilisée lorsque l'on n'est pas certain que $J^t J$ soit inversible (voir `matlab` pour des exemples d'utilisation) :

$$(J(x_k)^t J(x_k) + \lambda_k I) d_k = -J(x_k)^t r(x_k), \quad \lambda_k > 0.$$

1. Le logarithme du rapport des probabilités s'appelle l'Evidence. Pourquoi modéliser par une relation linéaire ce logarithme plutôt que le ratio directement est un choix a priori (Bayesien) et tente de modéliser le fait qu'une croyance/probabilité évolue lentement, puis plus rapidement, et non pas de façon linéaire. Cette évolution ressemble à une fonction Erf ou $Arctg$.

Un tel problème de moindres carrés nonlinéaire apparaît, par exemple, dans un problème de maximisation de vraisemblance :

Dans un dispositif expérimental on mesure aux instants t_i les données $y_i, i = 1, ..., m$. Le phénomène est modélisé par $\phi(t, x \in \mathbb{R}^n)$.

On note $r_i = y_i - \phi(t_i, x)$, la différence entre le modèle et les observations et on suppose que ces erreurs sont indépendantes et identiquement distribués (i.i.d.) de loi $\mathcal{N}(0, \sigma^2)$ et de densité $g_\sigma(\varepsilon) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{\varepsilon^2}{2\sigma^2})$. La vraisemblance des observations $y \in \mathbb{R}^m$ est donnée par :

$$V(\sigma, y, x) = \Pi_{i=1}^m g_\sigma(r_i) = (2\pi\sigma^2)^{-m/2} \exp\left(-\frac{1}{2}\sum_{i=1}^m \frac{r_i^2}{\sigma^2}\right).$$

Pour obtenir le maximum de vraisemblance, à σ fixé, il faut donc résoudre un problème de moindres carrés nonlinéaire : $\min_x \frac{1}{2}\sum_{i=1}^m r_i^2$.

Expliciter les étapes de l'algorithme de Gauss-Newton pour $\phi = at^2 + bt + c$ et $x = (a, b, c)$. Peut-on aborder la question (37) de cette manière ?

43 Exemple fondamental et contraintes d'égalité

On étend la question (31) à une situation sous contrainte d'égalité. Il s'agit de trouver $x \in \mathbb{R}^n$ tel que $J(x) = \frac{1}{2}(Ax, x) - (b, x)$ soit minimal et que la solution vérifie $Bx = c$. A est SDP $n \times n$ et B est $m \times n$, de rang m avec $m \leqslant n$.

1. Montrer l'existence et l'unicité de la solution.

2. Supposons que $\overline{x}$ réalise le minimum de J dans $\mathbb{R}^n$. Montrer que minimiser J sur l'ensemble $K = \{x \in \mathbb{R}^n; Bx = c\}$ équivaut à trouver $x^* \in K$ minimisant $(A(x - \overline{x}), x - \overline{x})$. Donner une interprétation géométrique.

3. Montrer que $x \in K \Leftrightarrow x^* - x \in Ker(B)$.

4. En utilisant le résultat de l'exercice (19) et le fait que $Ker(B)^\perp = Im(B^t)$, montrer que $A(x^* - \overline{x})$ est orthogonal à $(x^* - x), \forall x \in K$ et qu'il existe donc m scalaires $p_1, ..., p_m$ tels que :

$$Ax^* - b + \sum_{i=1}^m p_i B_i^t = 0.$$

Que représente cette égalité par rapport au Lagrangien du problème ?

44 Problèmes primal et dual

Considérons le Lagrangien du problème donné en exercice (43).

Ecrire les conditions d'optimalité de Lagrange et donner sa forme matricielle.

Dire pourquoi B doit être de rang m.

Montrer que le problème dual (équation en $p = (p_1, ..., p_m)$) et primal s'écrivent :

$$BA^{-1}B^t p = BA^{-1}b - c, \quad Ax = b - B^t p.$$

Expliciter ces deux problèmes pour $J(x, y) = x^2 + 2y^2$ et $x + y = 1$.

45 Algorithme d'Uzawa

Pour le problème de la question (43), l'algorithme d'Uzawa construit l'itération :

$$Ax^k = b - B^t p^k, \quad p^{k+1} = p^k + \rho(Bx^k - c), \quad \rho > 0.$$

Montrer que la suite (x^k, p^k) converge vers le point-selle du Lagrangien si :

$$\rho < \frac{\lambda_{min}(A)}{\lambda_{max}(BB^t)}.$$

Que peut-on dire sur la complexité de l'algorithme d'Uzawa par rapport à la méthode de l'exercice 44 ?

46 Optimisation d'un portefeuille

Le modèle de Markovitz permet d'obtenir un portefeuille assurant un revenu r donné pour un risque minimal. Le risque ou la volatilité du portefeuille est modélisé par :

$$J(x) = \frac{1}{2}(Ax, x),$$

où A, SDP, est la matrice de variance-covariance. Le portefeuille est constitué de N actifs. La proportion de capital investie dans l'actif i est notée x_i :

$$\sum_{i=1}^{N} x_i = 1, \quad \sum_{i=1}^{N} r_i x_i = r.$$

Ecrire les conditions d'optimalité de Lagrange et en déduire que les multiplicateurs de Lagrange sont solutions d'un système 2×2 SDP (utiliser l'exercice (44)). Puis, en déduire le portefeuille optimal.

Faites évoluer le programme de l'exercice (35) pour résoudre numériquement ce problème.

On peut supposer qu'un revenu supérieur à r ne serait pas une difficulté. Considérer le même problème en replaçant la contrainte d'égalité par $\sum_{i=1}^{N} r_i x_i \geqslant r$. Ecrire les conditions de KKT et appliquer la méthode des contraintes actives pour résoudre le problème.

Peut-on obtenir le même rendement avec des niveaux de risques moindre en modifiant A (on appelle cela la diversification) ?

47 KKT et contraintes actives

En utilisant les conditions de KKT et la méthode des contraintes actives, résoudre :

$$\min_{x_1,x_2} J(x_1, x_2) = \frac{1}{2}(x_1 - 1)^2 + \frac{1}{2}(x_2 - 2)^2,$$

sous l'ensemble des contraintes :

$$x_1 - x_2 = 1, \quad x_1 + x_2 \leqslant 2, \quad -x_1 \leqslant 0, \quad -x_2 \leqslant 0.$$

Même question pour :

$$\min_{x_1,x_2,x_3} J(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2,$$

sous l'ensemble des contraintes :

$$x_1 + x_2 + x_3 = 3, \quad 2x_1 - x_2 + x_3 \leqslant 5.$$

48 Minimisation sous contraintes

On souhaite résoudre :

$$(P) : \min J(x, y) = x + y,$$

sous les contraintes :

$$g_1(x, y) = (x - 1)^2 + y^2 - 1 \leqslant 0, \quad g_2(x, y) = (x + 4)^2 + (y + 3)^2 - 25 \leqslant 0.$$

1. Représenter graphiquement l'ensemble des contraintes C et montrer qu'il est convexe.

2. Ecrire les conditions de KKT et donner la ou les solutions de (P). Détailler les situations de points non-réguliers.

49 Uzawa et gradient projeté

On étend la question (31) à une situation sous contrainte d'inégalité.

Il s'agit de trouver $x \in \mathbb{R}^n$ tel que $J(x) = \frac{1}{2}(Ax, x) - (b, x)$ soit minimal et que la solution vérifie $Bx \leqslant c$. A est SDP $n \times n$ et B est $m \times n$, de rang m avec $m \leqslant n$.

- Adapter l'algorithme d'Uzawa de l'exercice (45) dans le cadre des contraintes actives en utilisant la méthode de gradient projeté sur les multiplicateurs de Lagrange positifs ou nuls.

- Pourquoi la condition donnée dans l'exercice (45) reste valable pour assurer la convergence vers le point-selle du Lagrangien généralisé ?

- Que peut-on dire de l'opérateur de projection nécessaire dans l'algorithme d'Uzawa par rapport à celui du gradient projeté ?

50 Uzawa, gradient projeté et pénalisation

Décrire la résolution numérique du problème de projection de $a = (a_1, ..., a_n) \in \mathbb{R}^n$ sur $E = \{x = (x_1, ..., x_n) \in \mathbb{R}^n; x_i \leqslant x_{i+1}, \forall i = 1, ..., n - 1\}$: $\min_{x \in E} \|x - a\|$, par l'algorithme d'Uzawa.

Décrire cette résolution par un algorithme de gradient projeté, après une reformulation du problème :

Trouver $(x_1, y_2, ..., y_n)$ avec $x_1 \in \mathbb{R}$, $y_i \geqslant 0$ et $x_i = x_1 + \sum_{j=2}^{i} y_j$ solution de :

$$\min_{x_1 \in \mathbb{R}, y_i \geqslant 0} \sum_{i=1}^{n} (x_1 + (\sum_{j=2}^{i} y_j) - a_i)^2.$$

Comparer les résultats à celle de la méthode de pénalisation pour résoudre :

$$\min_x \frac{1}{2}\|x - a\|^2 + \frac{1}{\varepsilon} \sum_{i=1}^{n-1} (x_i - x_{i+1})_+^2.$$

Programmer les trois approches en dimension n quelconque.

51 Apprentissage mathématique, Séparateurs à Vaste Marge (SVM)

Les Séparateurs à Vaste Marge ou Machines à Vecteurs de Support (Support Vector Machine, SVM), introduits par Vladimir Vapnik dans les années 1990, sont un ensemble de techniques

d'apprentissage supervisé destinées à résoudre des problèmes de discrimination et de régression. Les SVM sont une généralisation des classifieurs linéaires. Les SVM ont sont populaires pour leur capacité à travailler avec des données de grandes dimensions, le faible nombre d'hyperparamètres, leurs garanties théoriques, et leurs bons résultats en pratique.

Les séparateurs à vastes marges utilisent les notions de marge maximale et de fonction noyau. Dans la suite nous considérons uniquement les noyaux linéaires (les données sont séparables par un hyperplan). Si les données ne sont pas linéairement séparables, il faut essayer des noyaux non-linéaires et trouver celui qui convient le mieux au problème. Les noyaux les plus courants sont le linéaire (notre cas ici), polynomial et gaussien.

La marge est la distance entre la frontière de séparation et les échantillons les plus proches. Ces derniers sont appelés vecteurs supports. Dans les SVM, la frontière de séparation est choisie comme celle qui maximise la marge.

Illustrer ces deux idées sur un exemple en 2 dimension.

Dans la suite, nous allons formuler cette idée comme un problème d'optimisation.

Considérons un ensemble d'apprentissage, où les points appartiennent à deux classes :

$$\{(x_1, l_1), (x_2, l_2), \ldots, (x_k, l_k), \ldots, (x_p, l_p)\} \subset \mathbb{R}^N \times \{-1, 1\}$$

où les l_k sont les labels, p est la taille de l'ensemble d'apprentissage et N la dimension des vecteurs d'entrée x.

On suppose que ces données sont séparables par un hyperplan $h(x) = w^T x + b$, avec $x, w \in \mathbb{R}^N$ et $b \in \mathbb{R}$. w est le vecteur des poids et b le biais. x est de classe 1 si $h(x) \geqslant 0$ et de classe -1 sinon.

La séparabilité linéaire s'écrit :

$$l_k h(x_k) = l_k(w^T x_k + b) \geqslant 0 \quad 1 \leqslant k \leqslant p.$$

La frontière de décision $h(x) = 0$ est l'hyperplan séparateur. Le but d'un algorithme d'apprentissage supervisé est d'apprendre la fonction $h(x)$ sur l'ensemble d'apprentissage. La classe d'un nouveau point x, n'appartenant pas à la base d'apprentissage, sera donnée par le signe de $h(x)$.

Pour un ensemble de points, il existe une infinité d'hyperplans séparateurs dont les performances en apprentissage sont identiques.

Illustrer ceci sur votre exemple en 2 dimension.

Cependant, pour ces hyperplans les performances en inférence peuvent être très différentes. On choisit l'hyperplan qui maximise la marge entre les échantillons et l'hyperplan séparateur.

La marge est la distance entre l'hyperplan et les échantillons les plus proches. Ces derniers sont appelés vecteurs supports. L'hyperplan qui maximise la marge est solution du problème d'optimisation :

$$\arg\max_{w,b} \min_{k=1,\ldots,p} \{||x - x_k|| : x \in \mathbb{R}^N, w^T x + b = 0\}$$

On rappelle que la distance entre un point (x_0, y_0) et une droite $ax + by + c = 0$ est $|ax_0 + by_0 + c|/\sqrt{a^2 + b^2}$. La distance entre $x_k \in \mathbb{R}^N$ et l'hyperplan $H = \{h(x) = 0\}$ est $|h(x_k)|/\|w\|$ [2].

2. La direction normale à H est donnée par $\nabla h = w$. La normale unitaire est $n = w/\|w\|$.

Si $P \in H$, on a $h(P) = w^T P + b = 0$. Pour un point $x \in \mathbb{R}^N$, la distance entre x et sa projection orthogonale π sur H est

$$\|x - \pi\| = \| \left(n^t(x - \pi)\right) n\| = \| \left(\frac{w^t}{\|w\|}(x - \pi)\right) \frac{w}{\|w\|}\| = \frac{|w^t x + b - w^t \pi - b|}{\|w\|} = \frac{|h(x) - h(\pi)|}{\|w\|} = \frac{|h(x)|}{\|w\|}$$

$\pi = x_k - (|h(x_k)|/\|w\|)n = x_k - w(|h(x_k)|/\|w\|^2)$ est donc la projection orthogonale de x_k sur H.

On peut vérifier la caractérisation de la projection orthogonale sur le convexe H (c'est un s.e.v) :

$$< x_k - \pi, x >=< x_k - x_k + (|h(x_k)|/\|w\|)n, x >= (h(x_k)/\|w\|) < n, x >= 0, \quad \forall x \in H$$

Considérons un point $x_0 \in \mathbb{R}^N$ appartenant à l'hyperplan $h(x) = 0$. Sa distance aux hyperplans $h(x) = \pm 1$ est de $1/\|w\|$.

Pour résoudre notre problème d'optimisation, il suffit donc de minimiser $\|w\|$ sous une contrainte d'admissibilité. On considère le problème d'optimisation quadratique sous contrainte que nous savons résoudre :

$$\text{Minimiser} \quad \frac{1}{2}||w||^2 \quad \text{sous les contraintes} \quad l_k(w^T x_k + b) \geqslant 0, \quad k = 1, ..., p$$

Résoudre par pénalisation en minimisant : $J(w, b) = \frac{1}{2}||w||^2 + \alpha \sum_{k=1}^{p} \max(0, -l_k(w^T x_k + b))$.

Ecrire le Lagrangien du problème et utiliser l'algorithme d'Uzawa projeté.

Tirer aléatoirement 100 points dans $\mathbb{R}^2$. Les points au-dessus de l'axe des x seront label 1 et les autres label -1.

Considérer les 10 premiers points pour apprentissage. Que valent w et b ?

Inférence : tester ensuite votre classifieur sur les autres 90 points. Visualiser le résultat.

Pour aller plus loin, consulter le site `scikit-learn`.

52 Apprentissage mathématique, Réseaux de Neurones Artificiels (ANN)

Nous allons décrire les différents étapes et ingrédients mathématiques présents dans les réseaux de neurones artificiels (Artificial Neural Networks-ANN) et ceci sans faire appel à un vocabulaire 'neuronal'.

Les réseaux peuvent être classés dans deux grandes catégories : les réseaux de neurones récurrents (RNN) et les réseaux de neurones à convolution (CNN). Les premiers sont, en général, destiner au traîtement de données ayant une causalité intrinsèque, comme les séries temporelles, et les seconds sont plutôt performants pour les traitement d'images. La différence principale entre les deux catégories est la l'absence ou la présence de boucles entre les niveaux dans le réseau. Les CNN sont 'directs' (Feedforward) alors que les RNN font appels à des aller-retours entre les différentes couches du réseau.

ChatGPT fait appel à des réseaux appelés 'Transformer' qui remplacent désormais les RNN pour le traitement de langage ou texte. La différence principale est que les Transformers traitent l'ensemble de l'information d'entrée en même temps et non pas séquentiellement comme dans les RNN.

Les deux catégories de réseaux CNN et RNN (et les Transformers aussi) font essentiellement appel à quatre types de couches :

- C_1 : couches faisant intervenir des modèles linéaires,
- C_2 : couches faisant intervenir des fonctions d'activation pour mettre les valeurs faibles à zéro (les couches Dropout sont un cas particulier de fonctions d'activation),
- C_3 : couches à convolution pour effectuer des convolutions locales qui sont aussi des couches linéaires, mais dont la matrice a une propriété de somme locale donnée,
- C_4 : couches 'maxpool' pour extraire le maximum parmi un ensemble de valeurs.

L'introduction de fonctions nonlinéaires est indispensable car un réseau basé uniqmenent sur des produits de fonctions linéaires sera une fonction linéaire.

Ainsi, un réseau R, liant des entrées x appartenant à une base de données X aux sorties y appartenant à une base Y, comportera une succession de couches C_1, C_2, C_3 et C_4 (pas nécessai-

car les vecteurs de H sont perpendiculaires à n.

rement dans cet ordre) :

$$R : x(\in X) \to ... \to C_1^n \to C_2^n \to C_3^n \to C_4^n \to ... \to y(\in Y)$$

où n indique le nième macro-couche du réseau : chaque macro-couche est formée de 4 couches locales.

Les réseaux peuvent être totalement connectés (fully connected) ou pas. Dans le premier cas, toutes les variables de deux couches successives sont connectées. On parle aussi de couche dense.

Les opérateurs de convolution peuvent être 1D pour les séries temporelles, 2D pour les images ou 3D pour les films. Les supports des convolutions sont des paramètres a priori. Les supports sont différents d'une couche de convolution à une autre. Ces couches sont linéaires avec une contrainte sur la somme locale des variables (support de la convolution).

Les fonctions d'activation sont nonlinéaires. Les plus utilisées sont la fonction `Relu, Sigmoid, Tanh, Softmax, Swish`.

Il n'y a pas de méthodologie a priori pour définir l'architecture du réseau : nombre de couches, l'alternance entre les différents types de couches, nombre de variables (neurones) par couche, etc. Cette définition se fait de façon empirique.

Une fois le réseau défini, les seules variables qui doivent être identifiées sont les n matrices des couches linéaires. Les coefficients de ces matrices sont appelés coefficients synaptiques et sont solutions d'un problème d'optimisation sous contrainte, en général résolu par méthode de descente. Le gradient stochastique et ADAM sont les méthodes les plus utilisées. On cherche aussi à identifier les seuils d'activation des couches d'activation.

Comme dans toutes les méthodes d'apprentissage, la base de données globale disponible est partagée en deux : $(X, Y) = (X_L, Y_L) \cup (X_V, Y_V)$. La première partie est réservée à l'apprentissage (en général 80% des données) et le reste pour la validation (mesurer le sur-apprentissage qu'il faut éviter : contrainte).

Il s'agit, alors, de minimiser l'erreur d'apprentissage J_L ainsi que celle de validation J_V :

$$J_L(A_{i=1,...,n}(X_L, Y_L)) = \|Y_L - < R(X_L, Y_L), X_L > \|^2,$$

$$J_V(A_{i=1,...,n}(X_L, Y_L)) = \|Y_V - < R(X_L, Y_L), X_V > \|^2.$$

Les fonctionnelles sont des sommes de carrés et sont donc séparables. Les volumes de données sont souvent importants. En conséquence, les méthodes de descente partielles, comme celles décrites dans 38, sont utilisées. En générale, les deux fonctionnelles commencent à décroitre et la minimisation s'arrête lorsque l'erreur de validation se met à croître.

On insiste sur le fait que les données de validation (X_V, Y_V) n'interviennent pas dans l'apprentissage. Elles servent uniquement à mesurer le sur-apprentissage à travers la mesure J_V et décider de l'arrêt de l'apprentissage. C'est ce qu'indique la notation $A_{i=1,...,n}(X_L, Y_L)$ et $R(X_L, Y_L)$. La notation $< R(X_L, Y_L), X_L >$ (resp. $< R(X_L, Y_L), X_L >$) exprime l'application du réseau aux données de la base de données d'apprentissage (resp. de validation).

Le théorème, dit d'approximation universelle, montre que toute fonction régulière peut être approchée uniformément, avec une précision arbitraire et dans un domaine fini de l'espace de ses variables par un réseau de neurones élémentaire à une seule couche cachée, avec les neurones possédant tous la même fonction d'activation, et un neurone de sortie linéaire. Ce résultat théorique doit être vu comme un résultat d'existence. Ce n'est pas un résultat d'unicité et ne propose pas non plus une méthode constructive. Il n'adresse pas les difficultés d'apprentissage et de stabilité en grande dimension. Il ne met donc pas en cause les structures d'apprentissage profond.

Nous verrons des exemples de définition et de manipulation de réseaux en séances basés sur des Multi-Layer Perceptrons qui sont la brique de base des réseaux Transformers de ChatGPT.

53 Programmation linéaire

Beaucoup d'applications (transport, logistique, planification, etc) nécessitent la résolution de problèmes de la forme (appelé forme standard) :

$$(\text{PL}) : \min_{z\in\mathbb{R}^m} c^t z, \quad Az = b \in \mathbb{R}^p, \quad z \geqslant 0.$$

Le coût et les contraintes sont linéaires. Le problème est souvent sous déterminé avec plus d'inconnues que d'équations, A a plus de colonnes que de lignes ($p < m$) et A est de rang p, sinon certaines contraintes sont combinaisons linéaires d'autres et peuvent donc être éliminées.

Le problème (PL) est équivalent à sa forme canonique faisant intervenir uniquement des inégalités :

$$(\text{PLC}) : \min_{z\in\mathbb{R}^m} c^t z, \quad Az \leqslant b, \quad -Az \leqslant -b, \quad -z \leqslant 0.$$

Lorsqu'un problème est formulé en terme de (PLC), on se ramène à (PL) par l'introduction de variables d'écart et d'éxcés.

La méthode la plus répandue pour la résolution des problèmes de programmation linéaire de petite taille est la méthode de simplex (G. Dantzig, 1947) qui peut être vue comme une méthode de Gauss avec pivot (variable sortant de la base).

Détailler une itération de l'algorithme de simplexe pour le problème suivant :

$$\max_{x_1,x_2\in\mathbb{R}^+} 10x_1 + 12x_2,$$

$$10x_1 + 5x_2 \leqslant 200, \quad 2x_1 + 3x_2 \leqslant 60, \quad x_1 \leqslant 34, \quad x_2 \leqslant 14.$$

Donner une représentation graphique et vérifier que la solution optimale se trouve à un noeud du simplexe.

54 Programmation linéaire, méthode duale

Considérons le problème (PL) donné en (53).
- Ecrire le Lagrangien de ce problème.
- Ecrire les conditions de KKT.
- Montrer que le problème dual de (PL) est :

$$\max_{\lambda\in\mathbb{R}^p} b^t\lambda, \quad A^t\lambda \leqslant c.$$

- Montrer que le saut de dualité est toujours positif :

$$\delta(x,\lambda) = c^t x - b^t\lambda \geqslant 0,$$

et s'annule à l'optimum.

Si on dispose d'un point intérieur au convexe définit par les contraintes (l'intérieur du simplex), on peut résoudre les conditions de KKT par une méthode de Newton avec projection. Ceci est la base des méthodes de points intérieurs (voir cours).

55 Programmation linéaire, exemple

Considérons

$$\min_{(x_1,x_2,x_3)\in\mathbb{R}^3} J(x_1,x_2,x_3) = -5x_1 - 4x_2 - 6x_3,$$

sous les contraintes :

$$x_1 - x_2 + x_3 \leqslant 20, \quad 3x_1 + 2x_2 + 4x_3 \leqslant 42, \quad 3x_1 + 2x_2 \leqslant 30,$$

$$x_1 \geqslant 0, \quad x_2 \geqslant 0, \quad x_3 \geqslant 0.$$

Résoudre ce problème en utilisant la méthode de simplex avec `matlab` ou `python`.

Quelles sont les contraintes actives à l'optimum (où le multiplicateur de Lagrange associé est strictement positif) ?

56 Un exemple de contrôle

Dans ce problème on utilisera ce qu'on a appris dans le cours de modélisation pour la solution d'une EDP linéaire.

On sait calculer la répartition de la température dans un domaine $\Omega = (0, L)$ (voir cours modélisation) où la température est connue et fixe aux deux extrémités : $u(0) = u(L) = u_b$. La présence de N sources de chaleur est modélisée par le terme source. Pour conserver la notation introduite jusqu'ici, les paramètres de contrôle sont $x = (f_1, ..., f_N)^t \in (\mathbb{R}_*^+)^N$.

- Ecrire le modèle (appelé le problème direct).

On souhaite contrôler la température dans une région donnée $\omega \subset \Omega$. Plus précisément, nous voulons déterminer, par une méthode de moindres carrés, les valeurs à donner aux puissances f de N sources de chaleur pour obtenir la température la plus proche possible d'une température donnée $u_{des} = 2u_b$ sur ω. En d'autres termes, on veut résoudre :

$$\min_x J(u(x)), \quad J(u(x)) = \frac{1}{2}\int_\omega (u(x,s) - u_{des}(s))^2 ds. \tag{2}$$

- Montrer (en utilisant le principe de superposition), que l'on peut décomposer la solution inconnue en la combinaison de $N + 1$ solutions :

$$u = u_0 + \sum_{i=1}^{N} f_i u_i = u_0 + x^t U,$$

où la solution de base u_0 s'obtient par résolution du problème direct sans terme source, en prenant en compte les conditions aux limites non-homogènes. Les N solutions élémentaires u_i, $i = 1, ..., N$ s'obtiennent, respectivement, en mettant à un la puissance associée et en annulant les $N - 1$ autres, ainsi que les conditions aux limites (uniquement des conditions homogènes).

Ainsi, une fois déterminées la solution de base u_0 et les solutions élémentaires u_i, $i = 1, ..., N$, les coefficients $f_i, i = 1, ..., N$, de la solution optimale sont solutions d'un problème de moindres carrés.

- Montrer qu'on obtient les coefficients $f_i, i = 1, ..., N$ en résolvant un système linéaire de N équations à N inconnues : $Ax = B$ avec

$$A_{i,j} = \int_\omega u_i(s)\, u_j(s) ds, \quad \text{et} \quad B_i = \int_\omega (u_0(s) - u_{des}(s))\, u_i(s) ds, \qquad i, j = 1, ..., N.$$

56.1 Avec une contrainte

Il est donc possible de trouver x^* tel que $\nabla_x J(u(x^*)) = 0$.

En pratique, nous devons souvent réaliser cette tâche en présence d'une contrainte telle que :

$$h(x) = \|x\|_1 - M \leqslant 0.$$

Adapter la formulation précédente pour la prise en compte de cette contrainte en considérant la fonctionnelle : $\tilde{J}(u(x),\alpha) = J(u(x)) + \alpha h(x)_+^2$.

Mettre en évidence le système linéaire $\tilde{A}(\alpha)\tilde{x}_\alpha = \tilde{b}$ tel que $\nabla_x \tilde{J}(u(\tilde{x_\alpha})) = 0$.

Tracer l'évolution de $h(x_\alpha)$ pour α croissant. $\alpha = 0$ correspond au cas sans contrainte.

Montrer que α tel que $h(\tilde{x}_\alpha) = 0$ réalise les conditions de KKT.

56.2 Solution par une méthode de descente

Utiliser une méthode de gradient pour résoudre les deux problèmes précédents.

56.3 Optimisation robuste

Jusqu'ici nous avons toujours considéré des problèmes d'optimisation mono-point où l'optimum est recherché en un des points de fonctionnement d'un système. On considère maintenant la solution u comme fonction de la variable x mais aussi d'autres paramètres de 'fonctionnement' qui ne sont pas parmi des variables d'optimisation : $u(x,p)$. Le cas le plus simple est lorsque p est défini sur un intervalle de 'fonctionnement' : $p \in [p_{min}, p_{max}]$. Nous verrons comment étudier la robustesse de notre 'design' par rapport à la variabilité de ces paramètres. Nous verrons aussi comment rechercher un optimum robuste en introduisant une optimisation multi-point basée sur une pondération de fonctionnelles mono-point :

$$j(x) = \sum_{i=1}^{N} \omega_i J(u(x,p_i)), \quad \sum_{i=1}^{N} \omega_i = 1.$$

Nous rechercherons alors à réaliser la condition d'optimalité :

$$\nabla_x j(x) = \sum_{i=1}^{N} \omega_i \nabla_x J(u(x,p_i)) = 0.$$

Or, nous avons vu que :

$$\nabla_x J(u(x,p_i)) = A(p_i)x - B(p_i) = 0.$$

Et, agglomérant les contributions, nous aboutissons à un système $\mathcal{A}x = \mathcal{B}$ que nous exhiberons.

Cet exemple est assez générique, penser à d'autres problèmes de même nature :

- Placement optimal des charges minimisant les déformations d'un plateau (lien avec le problème de la membrane en cours de modélisation),
- dépollution optimale (lien avec l'équation d'advection-diffusion-réaction en cours de modélisation),
- contrôle acoustique d'une salle de concert (lien avec l'équation des ondes et Helmholtz du cours d'optimisation),
- etc.

57 Evaluation du gradient

Nous avons vu que le gradient de la fonctionnelle jouait un rôle important dans les problèmes d'optimisation. Nous nous intéressons ici à l'évaluation de ce gradient et nous essayons de mettre en évidence la complexité de cette opération. Cette évaluation est loin d'être aisée, en particulier en présence d'équation d'état dont la solution est obtenue par l'utilisation de code en boîte-noire.

Considérons une boucle de simulation liant une paramétrisation x à une fonctionnelle J, mesurant un coût ou une contre-performance à minimiser, via la solution d'une équation d'état :

$$J(x) : x \to q(x) \to U(q(x)) \to J(x, q(x), U(q(x))),$$

où q et U sont des variables intermédiaires où dépendantes. L'état U est solution de l'équation d'état $(E(x, q(x), U(q(x))) = 0)$ et représente, en général, la quantité la plus complexe à évaluer. Nous avons vu des exemples d'équations d'état dans les cours de modélisation, d'équations différentielles et d'équations aux dérivées partielles.

Le gradient de J contient diverses contributions avec des complexités d'évaluation variables :

$$\frac{dJ}{dx} = \frac{\partial J}{\partial x} + \frac{\partial J}{\partial q}\frac{\partial q}{\partial x} + \frac{\partial J}{\partial U}\frac{\partial U}{\partial x}. \tag{3}$$

La dernière contribution est la plus coûteuse à évaluer car elle exige la linéarisation de l'équation d'état pour le calcul de $\frac{\partial U}{\partial x}$. Une façon simple de se rendre compte de la complexité consiste à utiliser les différences finies et à exprimer la complexité des évaluations à travers le nombre de fois où l'état doit être réévalué. Pour un calcul de la fonctionnelle, pour un point donné dans l'espace des paramètres d'optimisation, une évaluation de l'état U est nécessaire, tandis que le calcul de $\frac{\partial U}{\partial x}$ demandera n (si $x \in \mathbb{R}^n$) calculs supplémentaires de U pour une approximation seulement précise à l'ordre un. Ceci explique pourquoi la recherche d'une paramétrisation de faible dimension est très importante.

Les difficultés d'une évaluation efficace et précise du gradient suggèrent que le hessien sera encore moins accessibles. C'est pourquoi la méthode de Newton est très rarement utilisée en optimisation. Elle est remplacée par les méthodes quasi-Newton de type BFGS ou bien de Gauss-Newton vu en 42.

57.1 Différences finies

Les différences finies sont la méthode la plus utilisée car elle ne nécessite pas la connaissance explicite des opérateurs et convient donc aux outils 'boîte-noire'. Une approximation à l'ordre un est obtenue par des perturbations le long de chaque dimension :

$$\frac{dJ}{dx_i} \approx \frac{1}{\epsilon}[J(\vec{x} + \epsilon\vec{e_i}, U(\vec{x} + \epsilon\vec{e_i})) - J(\vec{x}, U(\vec{x}))].$$

Lors de l'utilisation de cette technique, les difficultés sont :

1. choix difficile pour l'incrément ϵ, surtout si les paramètres ont des dimensions différentes, ce qui implique souvent des increments différents pour chaque variable.
2. erreurs dans la soustraction de nombres réels voisins lors de la manipulation de nombres réels en discret.
3. complexité calculatoire proportionnelle à la dimension de x.

Les deux premières difficultés peuvent être réduites en utilisant des différences finies centrées, mais en doublant la complexité calculatoire :

$$\frac{dJ}{dx_i} \approx \frac{1}{2\epsilon}[J(\vec{x} + \epsilon\vec{e_i}, U(\vec{x} + \epsilon\vec{e_i})) - J(\vec{x} - \epsilon\vec{e_i}, U(\vec{x} - \epsilon\vec{e_i}))].$$

En pratique, on utilisera dans la mesure du possible cette approche pour identifier le bon incrément ϵ, puis on utilisera la première formule.

57.2 Travailler en variables complexes

Pour réduire encore l'importance du choix de l'incrément ainsi que l'erreur due à la soustraction de nombres réels voisins, on peut travailler en variables complexes. En effet, si J est une fonctionnelle réelle, on a :

$$J(x_i + i\epsilon, U(x_i + i\epsilon)) = J(x, U(x)) + i\epsilon J'_x - \frac{\epsilon^2}{2} J''_{xx} - i\frac{\epsilon^3}{6} J'''_{xxx} + o(\epsilon^3),$$

où $x_i + i\epsilon$ représente l'ajout au i_{eme} composante de x l'increment $\epsilon\sqrt{-1}$. D'où

$$\frac{dJ}{dx_i} \approx \frac{Im(J(x_i + i\epsilon, U(x_i + i\epsilon)))}{\epsilon}.$$

La difficulté ici est de passer en complexe et pour cela il faut disposer du programme source. Ce qui est rarement le cas en pratique. On peut cependant faire cela de façon partielle, notamment sur des routines utilisateurs que l'on est souvent amené à fournir aux codes industriels.

Exerice : Programmer les deux formules de différences finies à l'ordre en et deux et comparer avec le calcul de gradient par variable complexe pour la fonctionnelle :

$$J(x) = \sum_{i=1}^{10} \frac{1}{i^4}(x_i - i)^2,$$

au point $x = (2, ..., 2)$. Calculer l'angle entre les gradients approchées et le gradient exact et l'erreur commise sur son amplitude. Tracer ces quantités en fonction de ϵ.

57.3 Linéarisation directe

Si on dispose du programme source de l'équation d'état, pour réduire l'influence de ϵ, on peut utiliser le calcul des variations. Notons $\delta x = \epsilon\vec{e^i}$ et δU la variation de U (i.e. $\delta U = \delta x_i \partial_{x_i} U$). En linéarisant $E(x, U(x)) = 0$, nous avons

$$\frac{\partial E}{\partial U}\delta U = -\frac{\partial E}{\partial x}\delta x \approx -\frac{1}{\epsilon}[E(x + \epsilon\vec{e_i}, U(x)) - E(x, U(x))]. \quad (4)$$

En utilisant une méthode quasi-Newton pour la solution de l'équation d'état, nous avons :

$$\frac{\partial E(U^k)}{\partial U}\ (U^{k+1} - U^k) = -E(x, U^k(x)). \quad (4)$$

Ainsi, on peut utiliser la même itération, mais en remplaçant n fois le second membre avec les colonnes de $\frac{\partial E}{\partial x}$. On obtient ainsi $\frac{\partial U}{\partial x}$. Les autres contributions au gradient sont, en général, facilement calculables.

57.4 Méthode de Lagrange/Adjoint

Considérons le Lagrangien $L = J + p^T E$ (car $E(x) = 0$ est une contrainte d'égalité) et écrivons l'indépendance du Lagrangien par rapport à la variable dépendante U donnant l'expression de p :

$$\frac{\partial L}{\partial U} = \frac{\partial J}{\partial U} + p^T \frac{\partial E}{\partial U} = 0, \quad \Leftrightarrow \quad p^T = -\frac{\partial J}{\partial U}(\frac{\partial E}{\partial U})^{-1}. \quad (4)$$

Après substitution, on a

$$\frac{dJ}{dx} = \frac{\partial L}{\partial x} = \frac{\partial J}{\partial x} + p^T \frac{\partial E}{\partial x}.$$

Ainsi, on obtient le gradient par une seule résolution de l'équation 'adjointe' qui a la même complexité que (4). Ce qui veut dire que le coût de cette évaluation est indépendant de n.

On peut aussi voir la méthode de l'adjoint comme un regroupement différent des termes du gradient aboutissant à une complexité minimale. Considérons à nouveau (3) :

$$\frac{dJ}{dx} = \frac{\partial J}{\partial x} + \frac{\partial J}{\partial q}\frac{\partial q}{\partial x} + \frac{\partial J}{\partial U}\frac{\partial U}{\partial x}.$$

Nous avons vu que la majeure partie du coût dans cette évaluation provient du terme $\partial U/\partial x$, qu'il faut de plus stocker avant d'en faire le produit avec $\partial J/\partial U$. On peut cependant réécrire formellement le dernier terme comme :

$$\frac{\partial J}{\partial U}\frac{\partial U}{\partial x} = \frac{\partial J}{\partial U}((\frac{\partial E}{\partial U})^{-1}\frac{\partial E}{\partial x}) \quad \text{car} \quad \frac{\partial E}{\partial U}\frac{\partial U}{\partial x} + \frac{\partial E}{\partial x} = 0,$$

Mais on peut regrouper différemment les termes, en déplaçant les parenthèses :

$$\frac{\partial J}{\partial U}\frac{\partial U}{\partial x} = -(\frac{\partial J}{\partial U}(\frac{\partial E}{\partial U})^{-1})\frac{\partial E}{\partial x}.$$

On voit réapparaître la variable intermédiaire p solution de

$$p^T(\frac{\partial E}{\partial U}) = -\frac{\partial J}{\partial U},$$

qui a la complexité d'une évaluation de l'état et qui a la même taille que celui-ci.

57.5 Différentiation par programme ou automatique

Soit f une fonction composée de la forme :

$$x \in R^p \to y = h(x) \in R^n \to z = g(y) \in R^n \to u = f(z) \in R^q.$$

$$u' = f'(z)g'(y)h'(x), \tag{4}$$

où $f' \in R^{q\times n}$, $g' \in R^{n\times n}$ et $h' \in R^{n\times p}$. Pour faire cette évaluation, il faut stocker $M = g'(y)h'(x) \in R^{p\times n}$ puis calculer $u' = f'(z)M$. Après transposition, on a

$$u'^T = h'^T(x)g'^T(y)f'^T(z). \tag{4}$$

Le stockage nécessaire est maintenant $M = g'^T(y)f'^T(z) \in R^{n\times q}$. La complexité dépend alors de la dimension des espaces de départ et d'arrivée. Ces choix s'appellent modes direct et inverse de la différentiation. Ceci est aussi une autre façon de présenter les complexités des deux approches par linéarisation directe et méthode adjointe. D'un point de vue pratique, ils existent deux approches majeures pour la mise en oeuvre de la différentiation automatique : la pré-compilation et la surcharge d'opérateurs. Dans le premier cas, l'outil de différentiation automatique produit un code pour la dérivée en partant du code direct. Dans le second cas, par contre, un nouveau code n'est pas généré pour la dérivée. On introduit plutôt de nouvelles classes de variables, incluant la classe initiale, mais aussi ses dérivées. On redéfinit les opérations natives pour prendre en compte les manipulations des dérivées. L'exécution du code direct avec les variables dans ces nouvelles classes permettra alors d'accéder aux dérivées directement. Cette approche nécessite donc un langage objet de type C++, tandis que la pré-compilation est applicable aux langages de bas-niveaux.

57.6 Un exemple $R \to R^2 \to R$

Soit $f = x^2 + 3x$ ($f' = 2x + 3$),

```
y_1=x
y_2=x**2+2*y_1
f  =y_1+y_2
```

calculons df/dx.
Mode direct Une dérivation ligne à ligne en fonction de x nous donne :

$$\frac{dy_1}{dx} = 1, \quad \frac{dy_2}{dx} = 2x + 2\frac{dy_1}{dx},$$

$$\frac{df}{dx} = \frac{dy_1}{dx} + \frac{dy_2}{dx} = 1 + 2x + 2.$$

On doit stocker tous les calculs intermédiaires.
Mode inverse Considérons le Lagrangien du programme

$$L = y_1 + y_2 + p_1(y_1 - x) + p_2(y_2 - x^2 - 2y_1).$$

$$\frac{df}{dx} = \frac{\partial L}{\partial x} = -p_1 - 2p_2 x,$$

$$\frac{\partial L}{\partial y_1} = 1 + p_1 - 2p_2 = 0,$$

$$\frac{\partial L}{\partial y_2} = 1 + p_2 = 0.$$

Ce système triangulaire est résolu du bas vers le haut (remontée). Donc x n'intervient qu'à la dernière opération. Ce qui est intéressant si x est un vecteur de grande taille.

57.7 Illustration de la différentiation

Pour illustrer notre propos, nous donnons un exemple de calcul de dérivée en utilisant la différentiation automatique en mode directe, les différences finies et la méthodes des variables complexes pour la fonction $f(x) = sin(x^2 + 3x)$ au voisinage de $x = 1$.

```
Function test_gradient

Real x,y,f;
Complex xc,yc,fc,epsc;

!
!  calcul du gradient analytique
!  pour f=sin(x**2+3*x) au voisinage de x=1
!
!  definition de la fonctionnelle
!
x=1;
y=x**2+3*x;
f=sin(y);
!
```

```
!  calcul du gradient par differentiation automatique
!                  en mode direct
dy=2*x+3;
df=cos(y)*dy;
!
!  calcul du gradient par differences finies d'ordre 1
!
eps=0.0001;
x1=1+eps;
y1=x1**2+3*x1;
f1=sin(y1);
dffd=(f1-f)/eps;
!
!  calcul du gradient par differences finies d'ordre 2
!
x2=1.-eps;
y2=x2**2+3.*x2;
f2=sin(y2);
dffd2=(f1-f2)/(2.*eps);
!
!  calcul du gradient par variables complexes
!
epsc=cmplx(0.,eps) ! calcul CVM
xc=(1.,0)+epsc;
yc=xc**2+3.*xc;
fc=sin(yc);
fcvm=real(fc);
dfcvm=imag(fc)/eps;
!
End
```

57.8 DA en apprentissage mathématique

La minimisation d'erreur entre modèle et observation est au coeur de l'apprentissage mathématique. Les méthodes de gradient sont les plus utilisées. On parle de propagation rétrograde ou 'backpropagation' de l'erreur entre modèle et observations.

La différentiation automatique est très utile, notamment pour les réseaux de neurones, car le réseau peut être différent d'un problème à l'autre.

Plusieurs outils de différentiation automatique existent en précompilation ou en surcharge d'opérateurs pour manipuler les codes quelconques.

En machine learning, en en particulier les réseaux de neurones, il s'agit de différencier des opérations linéaires. Ce qui est intéressant car il n'est pas nécessaire de stocker les variables des niveaux intermédiaires pour la propagation rétrograde (voir le problème de stockage en adjoint en nonlinéaire).

En pratique, plusieurs environnements intégrés existent en `Python`.

57.9 Gradient d'une fonctionnelle en présence d'une équation d'état

Nous allons calculer des gradients de fonctionnelles dépendantes de variables solutions d'une équation d'état physique. C'est une situation très commune en optimisation industrielle. Cette

situation apparaît, par exemple, dans les problèmes d'optimisation de forme.

Dans la suite, x indique le paramètre d'optimisation. Considérons la fonctionnelle $J(x, u(x)) = x^n u_s(x)$ où u est solution de l'équation de Poisson (ou équation de membrane) :

$$-u_{ss} = 1, \quad \text{sur} \quad]x, 1[, \quad u(x) = 0, \quad u(1) = 0.$$

Ecrire un code (par exemple utilisant la méthode des différences finies avec marche en temps) pour résoudre ce problème comme solution stationnaire du problème instationnaire. Comparer la solution de votre code avec la solution exacte du problème (à expliciter) : $u(s) = -s^2/2 + (x + 1)\ s/2 - x/2$.

Montrer que $J_{,x}(x)$ le gradient de J par rapport à la frontière x est donné par :

$$J_{,x}(x) = x^{n-1}(nu_s(x) + xu_{sx}(x)) = \frac{x^{n-1}}{2}(-n(x-1) - x).$$

Propager le gradient par différentiation automatique et comparer avec un calcul par différence finie du gradient et la méthode des variables complexes.

Faisons la même analyse avec une équation d'état d'advection-diffusion impliquant le nombre de Peclet Pe (ce nombre sans dimension et positif mesure le rapport entre force d'inertie et de viscosité : un écoulement visqueux a un faible Peclet) :

$$u_s - Pe^{-1}\ \ u_{ss} = 0, \quad \text{sur} \quad]x, 1[, \quad u(x) = 0, \quad u(1) = 1.$$

Montrer que la solution de cette équation est :

$$u(s) \ = \ \frac{\exp(Pe\ x) - \exp(Pe\ s)}{\exp(Pe\ x) - \exp(Pe)}. \tag{4}$$

Faites évoluer votre code différences finies et comparer votre solution numérique à la solution exacte. Introduire une viscosité numérique en cas d'instabilité.

Nons pouvons calculer le gradient de la solution :

$$\begin{aligned}
&u_s(s) = \frac{-Pe\ \exp(Pe\ s)}{\exp(Pe\ x) - \exp(Pe)}, \\
&(u_s)_{,x}(s = x) = \frac{(Pe\ \exp(Pe\ x))^2}{(\exp(Pe\ x) - \exp(Pe))^2} = u_s^2(s = x). \\
&J_{,x}(s = x) = x^{n-1}u_s(x)(n + xu_s(x)).
\end{aligned}$$

Propager le gradient par différentiation automatique et comparer avec un calcul par différence finie du gradient et la méthode des variables complexes.

L'équation de membrane permet aussi de modéliser l'écoulement de Poiseuille dans un canal soumis à un gradient de pression p_s. Les parois sont à $y = \pm x$. La vitesse horizontale de l'écoulement (le long de s) est solution de :

$$u_{yy} = \frac{p_s}{\nu}, \quad u(-x) = u(x) = 0. \tag{3}$$

la solution de ce problème est donnée par : $u(x, y) = \frac{p_s}{2\nu}(y^2 - x^2)$. Le débit du canal est donné par $J(x) = \int_{-x}^{x} u(x, y)dy$. On s'intéresse au gradient du débit par rapport à la largeur du canal. Montrer que :

$$\frac{dJ}{dx} = \int_{-x}^{x} \partial_x u(x, y)dy = \frac{-2x^2 p_s}{\nu}.$$

Adapter le code développé pour la membrane et vérifier la pertinence des résultats numériques pour la solution et le gradient de la fonctionnelle.

58 Optimisation sans dérivée

Dans ce cours, nous avons vu des méthodes de minimisation utilisant le gradient et, eventuellement, le hessien de la fonctionnelle. Ces méthodes sont basées sur l'approximation locale de la fonctionnelle par un modèle m linéaire ou quadratique :

$$f(x^k + s) \sim m_k(s) = f(x^k) + s^T \ \nabla f(x^k) + \frac{1}{2} s^T \ \nabla^2 f(x^k) \ s$$

Or, en pratique, on ne dispose pas toujours du gradient de la fonctionnelle, et encore moins de son hessien. Les différences finies permettent une approximation du gradient par perturbation locale et le hessien peut être approché comme dans les méthodes quasi-Newton de type BFGS.

Mais, il arrive parfois que la fonctionnelle soit bruitée ou son évaluation entachée d'erreur. Ce qui fait que ces approximations ne sont plus pertinentes.

En effet, considérons $f(x) = F(x) + \psi(x)$ avec $\psi(x)$ une composante aléatoire modèlisant l'erreur d'évaluation de la fonctionnelle F qui n'est connue qu'à travers f.

Si $\nabla_h f(x)$ est l'approximation par différences finies centrées du gradient, l'erreur sur cette approximation aura une composante quadratique classique, mais aussi une contribution due à l'incertitude :

$$\|\nabla_h f(x) - \nabla F(x)\| \leqslant L_F h^2 + \frac{\eta(x,h)}{h}$$

où $\eta(x,h) = \sup_{\|z-x\|_\infty} |\psi(z)|$ et L_F est la constante de Lipschitz du hessien (e.g. la plus grande valeur propre dans le cas d'une fonctionnelle quadratique avec A SDP). On constate, en particulier, que l'erreur explose si on essaie d'augmenter la précision de l'estimation du gradient. Une méthode de descente n'est plus exploitable.

Expérimenter cette difficulté en considérant votre programme de minimisation pour une fonctionnelle quadratique.

En conséquence, il faut parfois pouvoir optimiser un système en connaissant uniquement la fonctionnelle f et ne jamais utiliser directement une estimation de son gradient. Les méthodes d'optimisation sans gradient introduisent un modèle linéaire ou quadratique local valable sur une 'région de confiance' :

$$m_k(s) = c + s^T \ g + \frac{1}{2} s^T \ H \ s,$$

où $c \in \mathbb{R}$, $g \in \mathbb{R}^n$ et $H \in \mathbb{R}^{n \times n}$ doivent vérifier certaines conditions de compatibilité ou d'interpolation. c, g et H peuvent aussi être 'fités' ou calés par moindres carrées pour que $m_k(s)$ passe au plus proche d'un nuage de q points $\mathcal{Y} = \{(y_1, f(y_1)), ..., (y_q, f(y_q))\}$. Cette approximation ou apprentissage sera valable uniquement localement. On cherchera le minimum de ce modèle, qui n'a plus de composante aléatoire. Si x^k est le minimum de cette fonctionnelle, les conditions d'interpolation locales sont de la forme :

$$m_k(y_i - x^k) = f(y_i), \quad \forall i \in \{1, ..., q\}.$$

Cette procédure est itérative et la nouvelle information $(x^k, f(x^k))$ peut être integrée à la base d'apprentissage : $\mathcal{Y} \cup \{(x^k, f(x^k))\}$. Il est possible aussi de retenir uniquement q points en substituant ce nouveau couple au couple le moins bon (i.e. avec la plus grande valeur de la fonctionnelle). Bien qu'on puisse travailler avec $q < n$, la complexité calculatoire est souvent comparable aux différences finies : $O(n)$ évaluations de f pour construire le modèle linéaire local (avec $H = 0$) et $O(n^2/2)$ pour le modèle quadratique (symétrie du hessien). Plusieurs programmes très aboutis existent pour l'optimisation sans dérivée. Ils font souvent appel à des heuristiques.

59 Optimisation multi-critère et fronts de Pareto

Les applications comportent souvent des problèmes d'optimisation multi-critère. On est dans le cadre d'optimisation d'un vecteur de q fonctionnelle $\tilde{\mathbf{J}} = \{J_i(x), i = 1, .., q\}$. Les fonctionnelles en jeu sont, en général, en conflit : on veut optimiser la performance d'un système dont on souhaite réduire le coût. Une première notion d'équilibre est celle de Pareto entre points non-dominés : x *is dominé par un point* y *si et seulement si* $J_i(y) \leqslant J_i(x)$*, avec inégalité stricte pour au moins une des fonctionnelles.*

En ces points de l'espace admissible il n'est pas possible d'obtenir une réduction pour toutes les fonctionnelles. La méthode la plus répandue est la minimisation de moyennes pondérées :

$$J(x) = \Sigma\alpha_i J_i(x), \quad \Sigma\alpha_i = 1, \quad \alpha_i \in [0, 1]$$

En variant la pondération, on atteint différents points du front en utilisant une méthode de descente. Cette approche ne peut atteindre les points situés sur les Pareto non-convexes. En effet, si la tangente au front est parallèle à un des axes, ceci indique que la fonctionnelle correspondante a un plateau et une méthode de descente ne pourra pas aller au-delà. Si le front est non-convexe mais la tangente n'est jamais parallèle à un axe, on peut rendre le front convexe en considérant :

$$\tilde{J}(x) = \Sigma\alpha_i J_i^n(x), \quad \Sigma\alpha_i = 1, \quad \alpha_i \in [0, 1]$$

On cherchera la puissance n la plus faible permettant une convexification du front. En effet, le conditionnement des problèmes d'optimisation associés est dégradé avec n croissant.

Considérer $J_1 = x_1 + x_2$ et $J_2 = (1/x_1 + \|x\|^2) + \beta(exp(-100(x_1 - 0.3)^2) + exp(-100(x_1 - 0.6)^2))$ pour $x \in [0.1, 1]^2$. Illustrer J_1 et J_2 pour $\beta = 0$ et 3. Echantillonner l'espace des paramètres et illustrer le front de Pareto pour $\beta = 0$ et $\beta = 3$. Utiliser votre programme de minimisation pour trouver 10 points sur ces fronts en variant la pondération en partant du point $(1, 1)$. Considérer comme direction de descente $d = -(\alpha\nabla J_1 + (1 - \alpha)\nabla J_2)$.

L'argument de la tangente proposé plus haut suggére qu'une méthode d'optimisation globale, permettant de franchir les plateaux, pourra atteindre les points dans les régions non-convexes des Paretos. Utiliser l'algorithme développé en 38.1 avec $\eta \neq 0$ et une vitesse initiale, partant toujours de $(1, 1)$.