EE-311—Apprentissage et intelligence artificielle

3. Optimisation convexe, regression multi-variables, regression logit

Michael Liebling

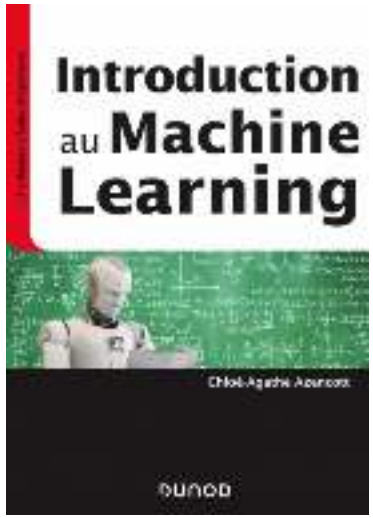
https://moodle.epfl.ch/course/view.php?id=16090

7 mars 2024 (compilé le 6 mars 2025)

idiap RESEARCH INSTITUTE

EPFL

Ouvrage de référence et source



Ces transparents sont basés en grande partie sur le texte de Chloé-Agathe Azencott "Introduction au Machine Learning", Dunod, 2019
ISBN 978-210-080153-4

L'auteure a mis le texte (sans les exercices) à disposition ici : http://cazencott.info/dotclear/public/lectures/IntroML_Azencott.pdf

Avertissement : Bien que ces transparents partagent la notation mathématique, la structure de l'exposition (en partie), et certains exemples avec le livre, ils ne constituent qu'un complément et non un remplacement ou une source unique pour la couverture des matières du cours. À ce titre, ces transparents ne se substituent pas au texte.

Contenu

- Optimisation Convexe
  - Descente de gradient (ne requiert pas de dérivées secondes)
  - Méthode de Newton (requiert des dérivées secondes)
  - Méthode du gradient conjugué
  - Méthode du gradient stochastique
  - Descente de coordonnées
- Régression linéaire multi-variable
- Régression logit
- Régression polynomiale

Optimisation : introduction

Les problèmes d'apprentissage peuvent souvent s'exprimer sous forme d'une fonction coût à minimiser.

Le but de l'optimisation est de minimiser cette fonction coût $f(\vec{x})$.

Lorsque la fonction coût est minimale (pour un argument $\vec{x}^*$), ses dérivées partielles sont nulles :

$$\frac{\partial f}{\partial x_i}(\vec{x}^*) = 0, \quad i = 1, \ldots, n$$

Les fonctions à minimiser sont rarement linéaires, mais on favorisera des fonctions et des formulations de problèmes *convexes*, afin d'assurer qu'il y ait une solution unique (et des algorithmes numériques qui convergent pour la trouver)

Convexité

Définition A.1 (Ensemble convexe) On dit d'un ensemble $\mathcal{S} \subseteq \mathbb{R}^n$ qu'il est *convexe* si est seulement si quels que soient $\vec{u}, \vec{v} \in \mathcal{S}$ et $t \in [0, 1]$,

$$t\vec{u} + (1 - t)\vec{v} \in \mathcal{S}$$

$\Leftrightarrow$ le segment $[\vec{u}, \vec{v}]$ est entièrement contenu dans $\mathcal{S}$.

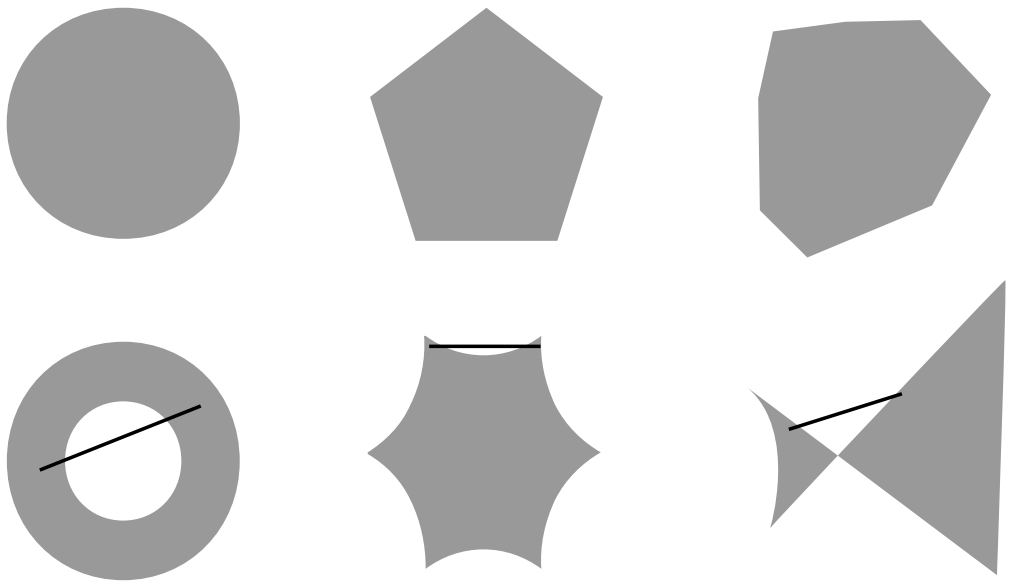


FIGURE A.1 – Les trois ensembles de $\mathbb{R}^2$ présentés sur la rangée du haut sont convexes. Les trois ensembles sur la rangée du bas ne sont pas convexes, et un exemple de segment reliant deux points de l'ensemble mais n'étant pas entièrement inclus dans cet ensemble est présenté pour chacun d'entre eux.

Azencott

Fonction convexe

Définition A.2 (Fonction convexe) Soit $\mathcal{U} \subseteq \mathbb{R}^n$. Une fonction $f : \mathcal{U} \to \mathbb{R}$ est dite *convexe* lorsque

- le domaine de définition $\mathcal{U}$ de f est un ensemble convexe ;
- quels que soient $\vec{u}, \vec{v} \in \mathcal{U}$ et $t \in [0, 1]$,

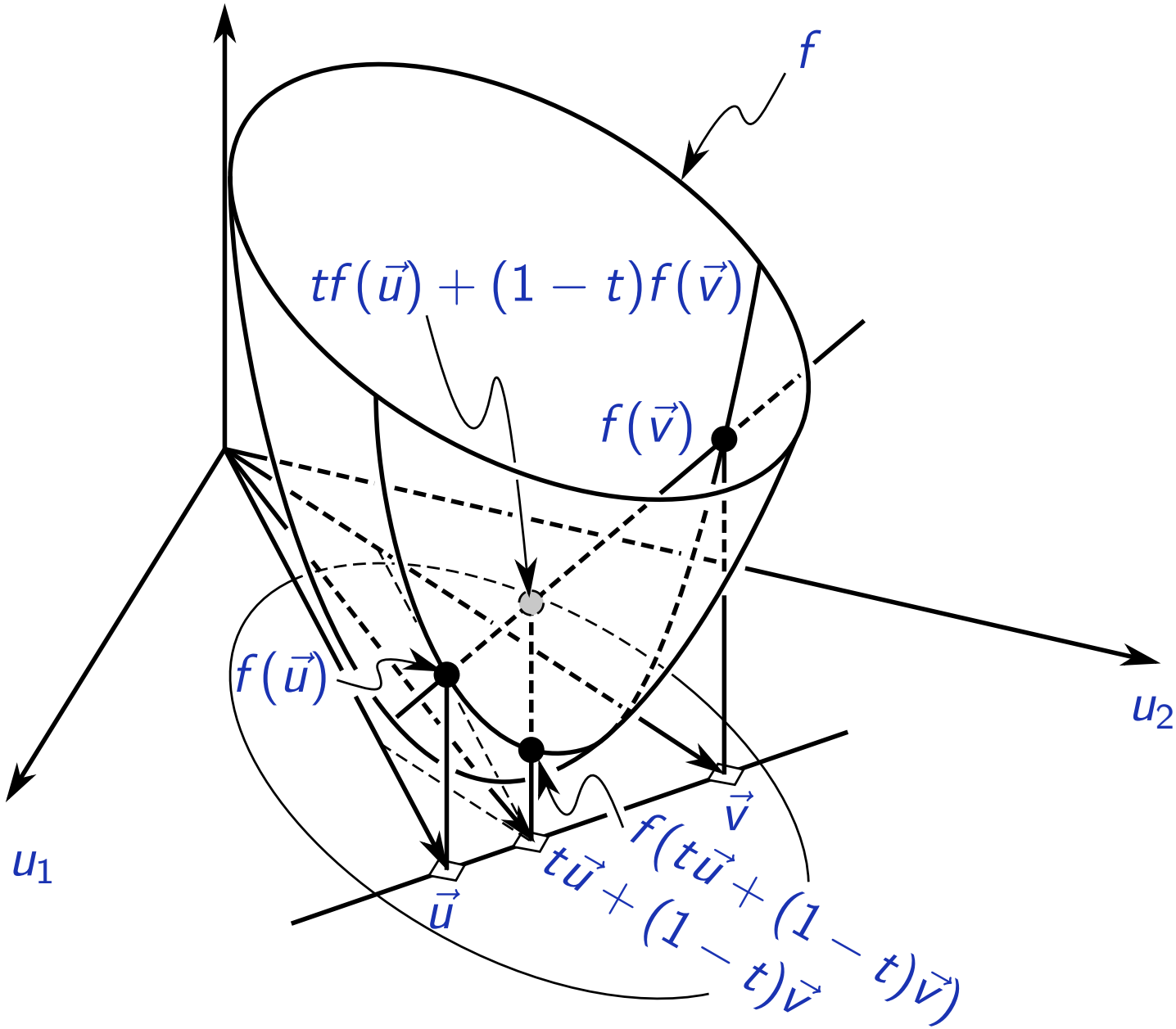
$$f(t\vec{u} + (1-t)\vec{v}) \leq tf(\vec{u}) + (1-t)f(\vec{v}),$$

c'est-à-dire que, sur $[\vec{u}, \vec{v}]$, f se situe au-dessous du segment $[(\vec{u}, f(\vec{u})), (\vec{v}, f(\vec{v}))]$.

Si l'inégalité est stricte pour tout $\vec{u} \neq \vec{v} \in \mathcal{U}$ et $t \in]0, 1[$, on parle alors de fonction *strictement convexe*. Une fonction strictement convexe a une courbure supérieure à celle d'une fonction affine.

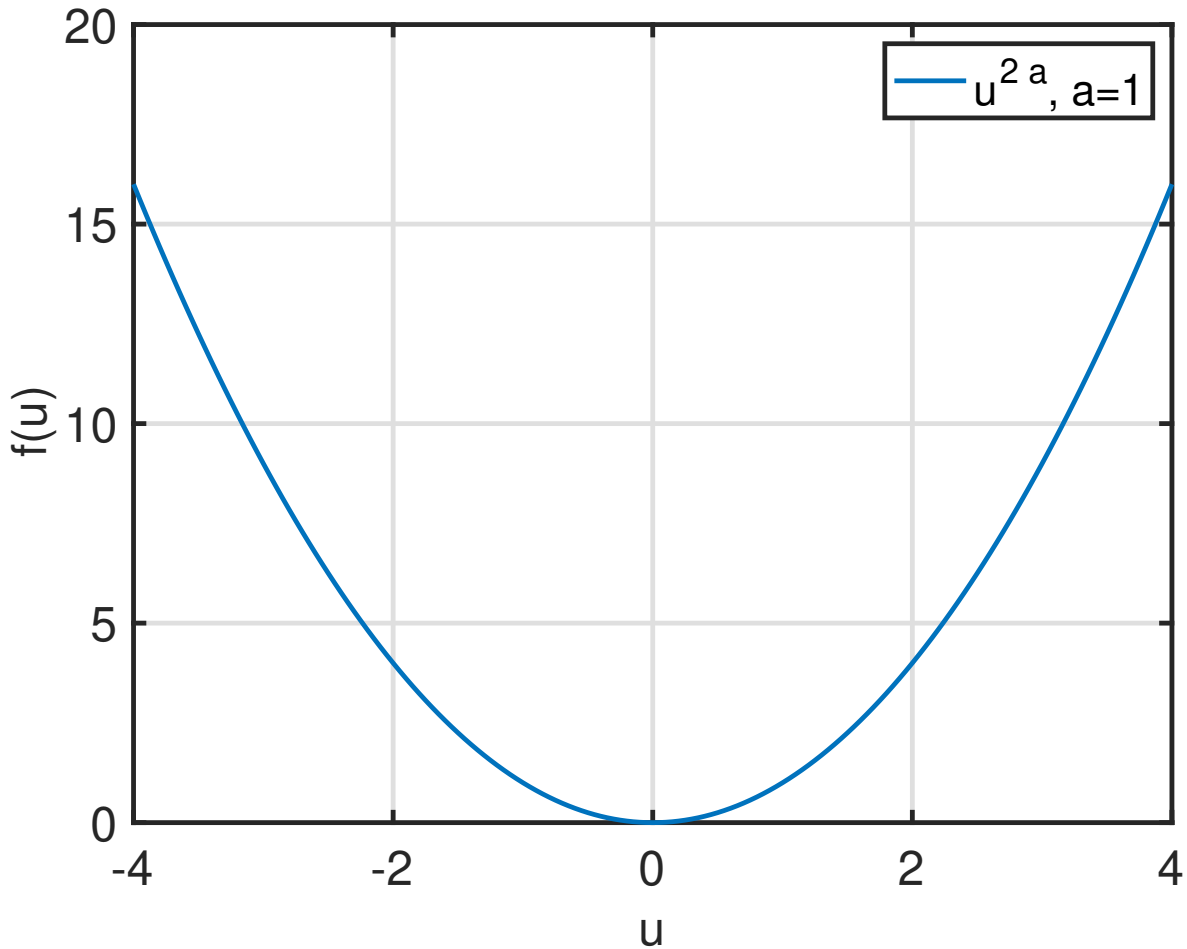
Dans le cas où il existe $k > 0$ tel que $f - \frac{k}{2}\|\vec{u}\|_2^2$ est strictement convexe, f est dite *fortement convexe*.

Illustration : définition d'une fonctions convexe



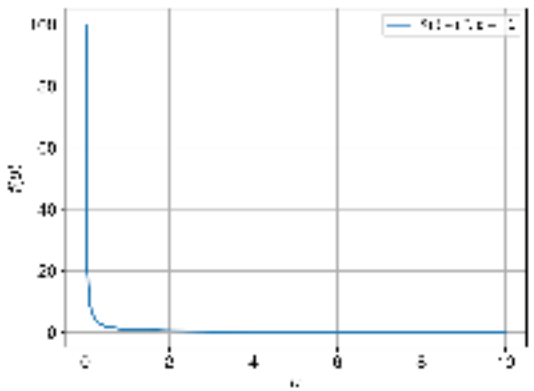
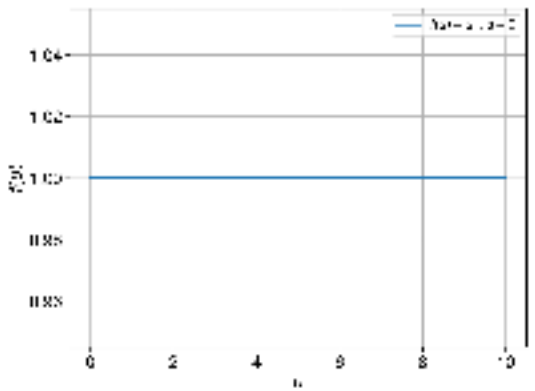
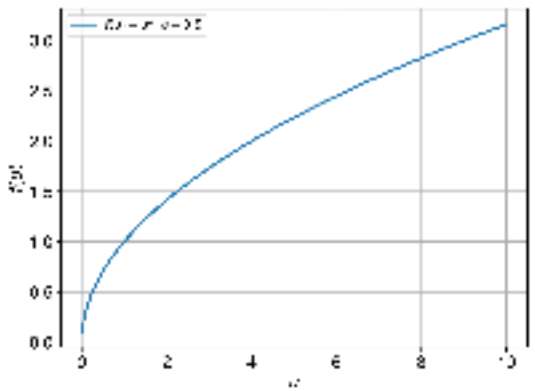
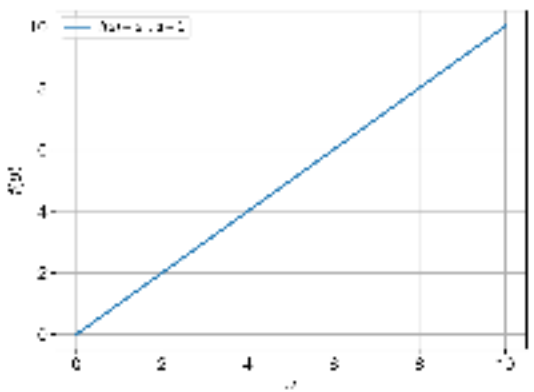
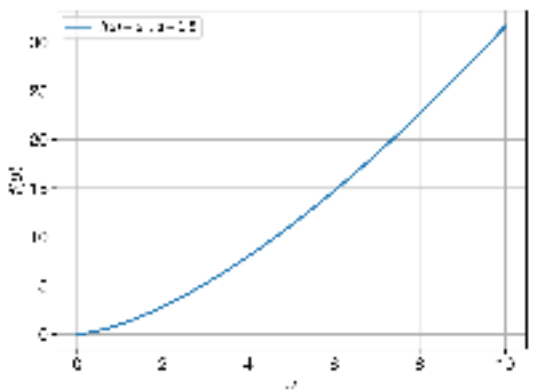
Fonction convexe : puissances paires

$$f : \mathbb{R} \to \mathbb{R}$$
$$u \mapsto u^{2a}, \quad a \in \mathbb{N}$$



Fonction convexe : puissances hors intervalle $]0, 1[$

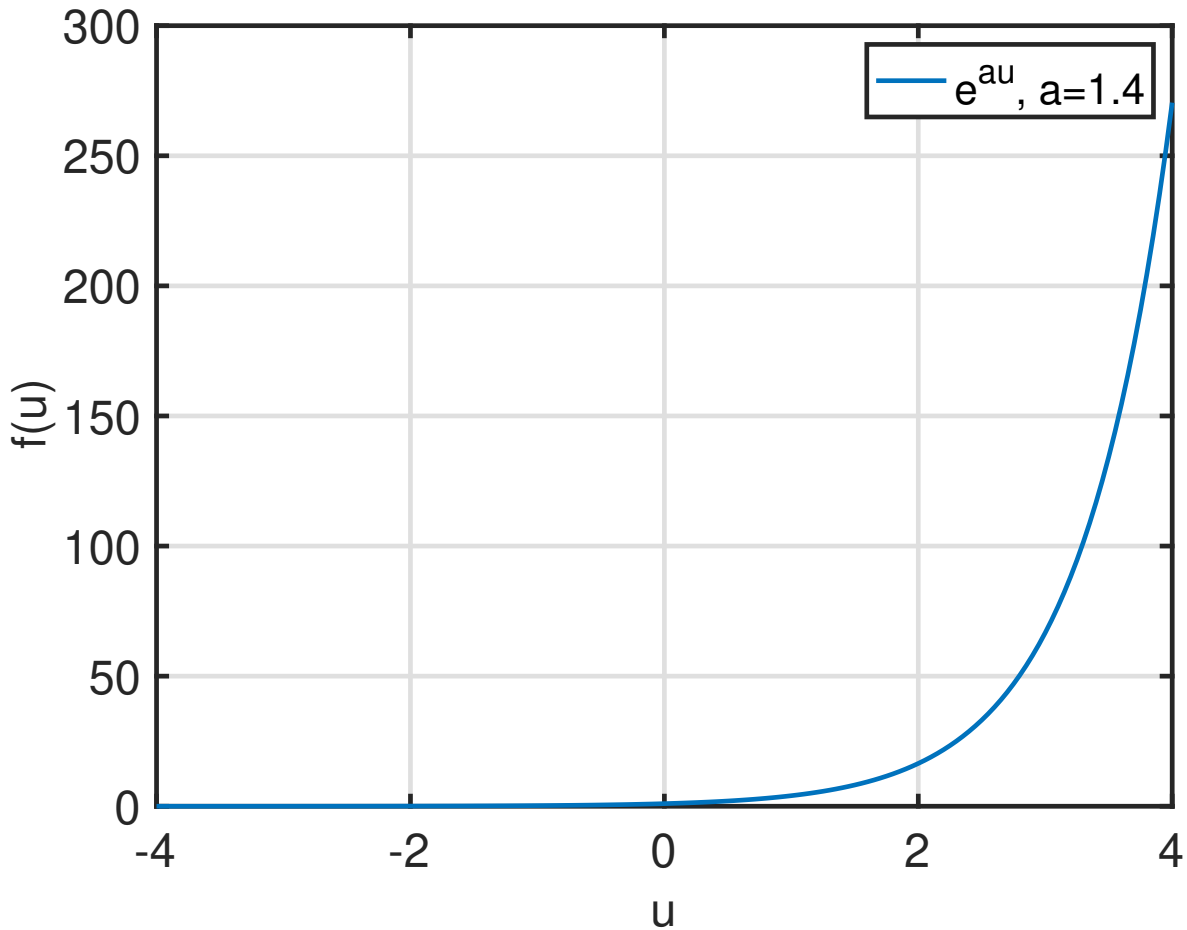
$$f : \mathbb{R}_+^* \to \mathbb{R}$$
$$u \mapsto u^a, \quad a \notin]0, 1[$$

Remarque : convexe si $a \notin]0, 1[\Leftrightarrow$ convexe si $a \in]-\infty, 0] \cup [1, \infty[$

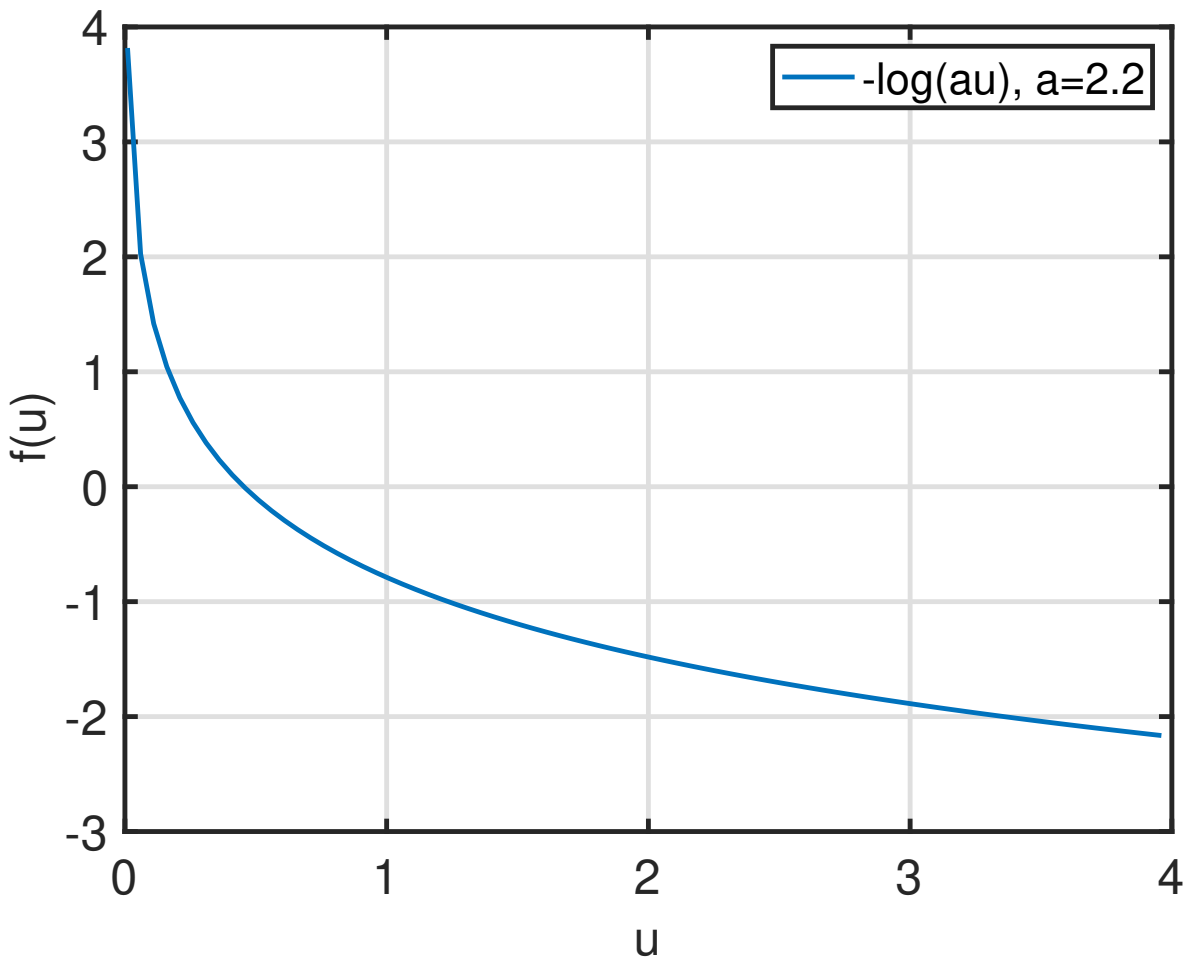
Fonction convexe : exponentielle

$$f : \mathbb{R} \to \mathbb{R}$$
$$u \mapsto e^{au}, \quad a \in \mathbb{R}$$



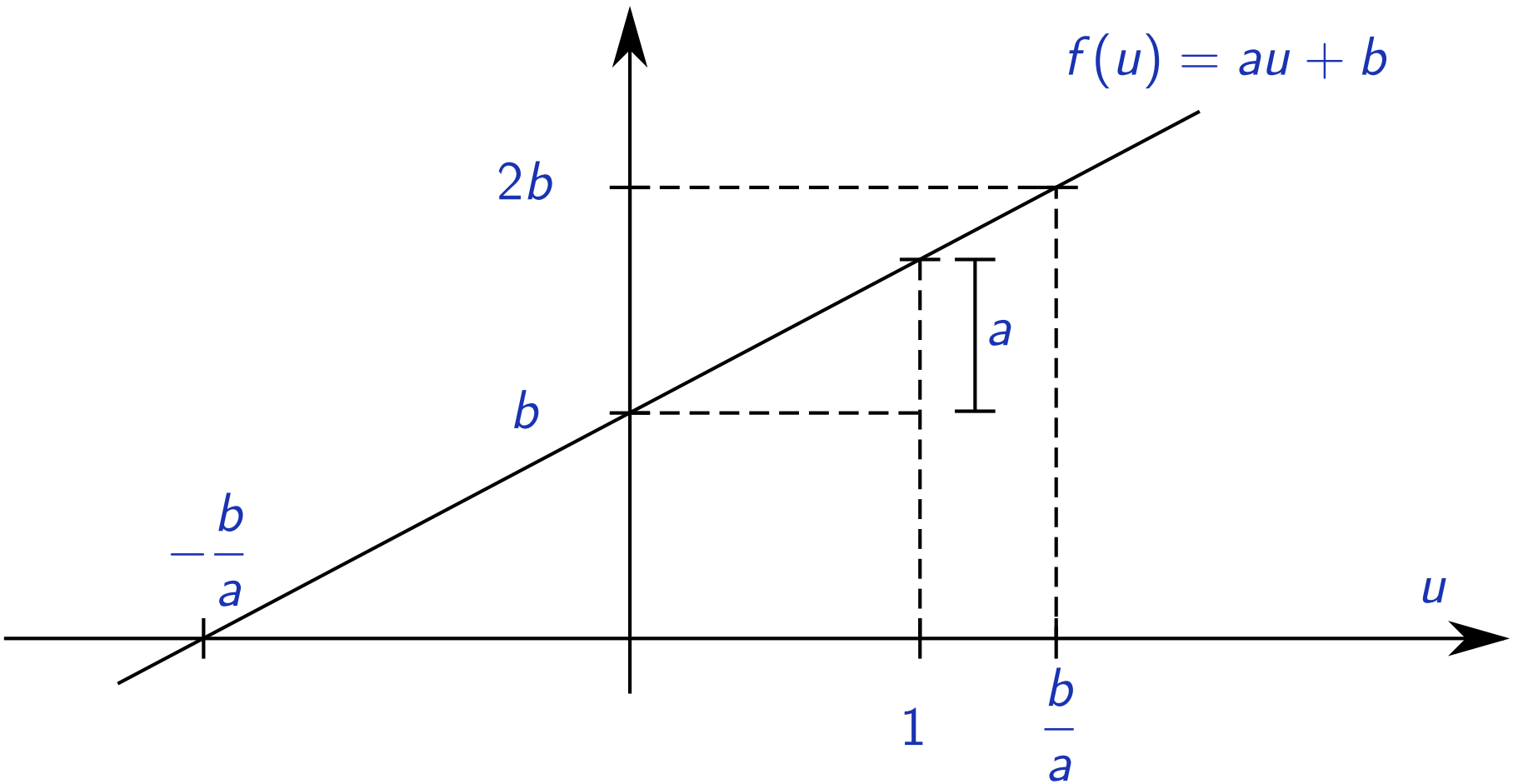
Fonction convexe : logarithme

$$f : \mathbb{R}_+^* \to \mathbb{R}$$
$$u \mapsto -\log(au), \quad a \in \mathbb{R}^+$$



Fonction convexe : fonction linéaire affine

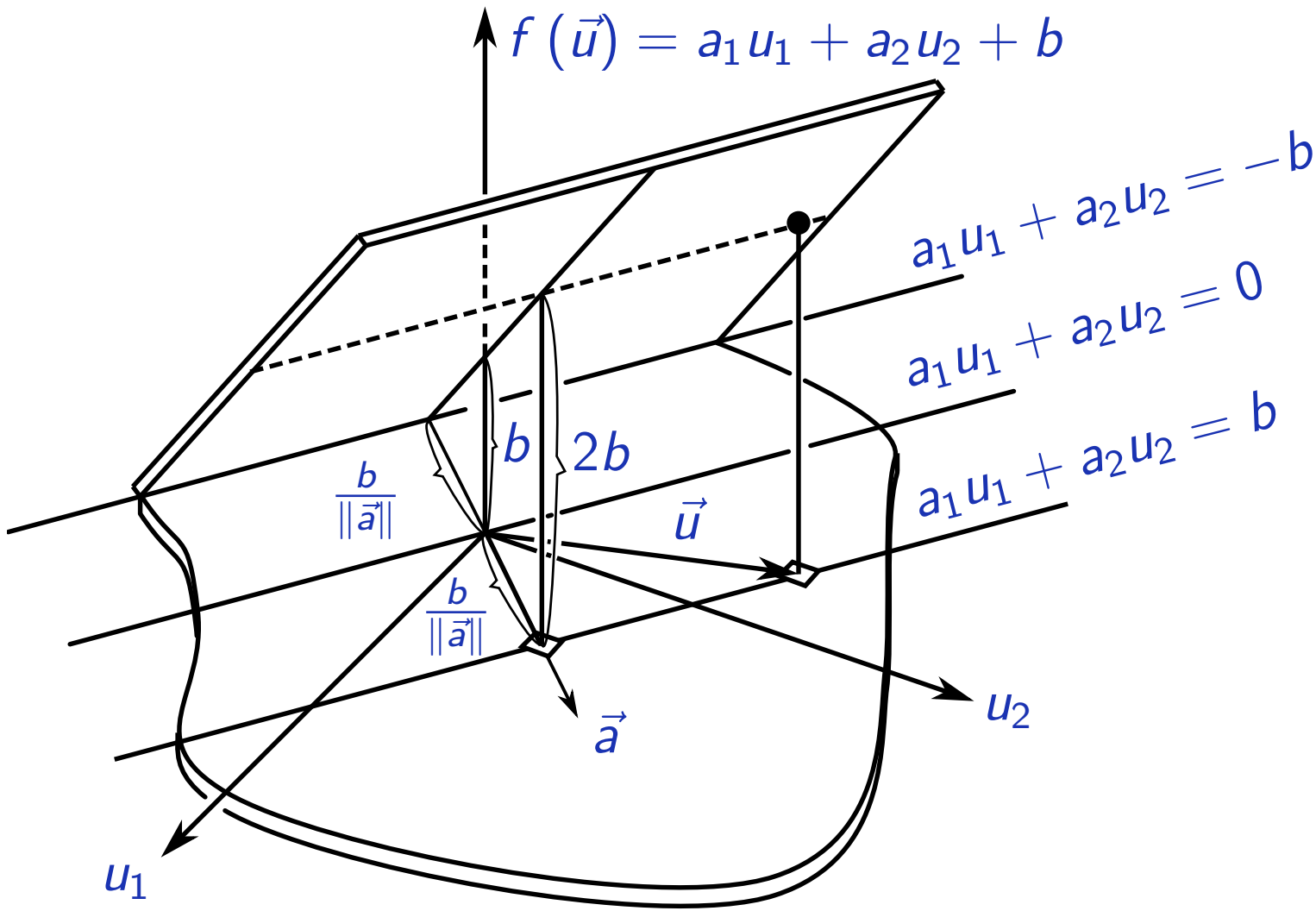
$$f : \mathbb{R} \to \mathbb{R}$$
$$u \mapsto au + b, \quad a \in \mathbb{R}, b \in \mathbb{R}$$



Fonction convexe : fonction linéaire affine ($n > 1$)

$$f : \mathbb{R}^n \to \mathbb{R}$$

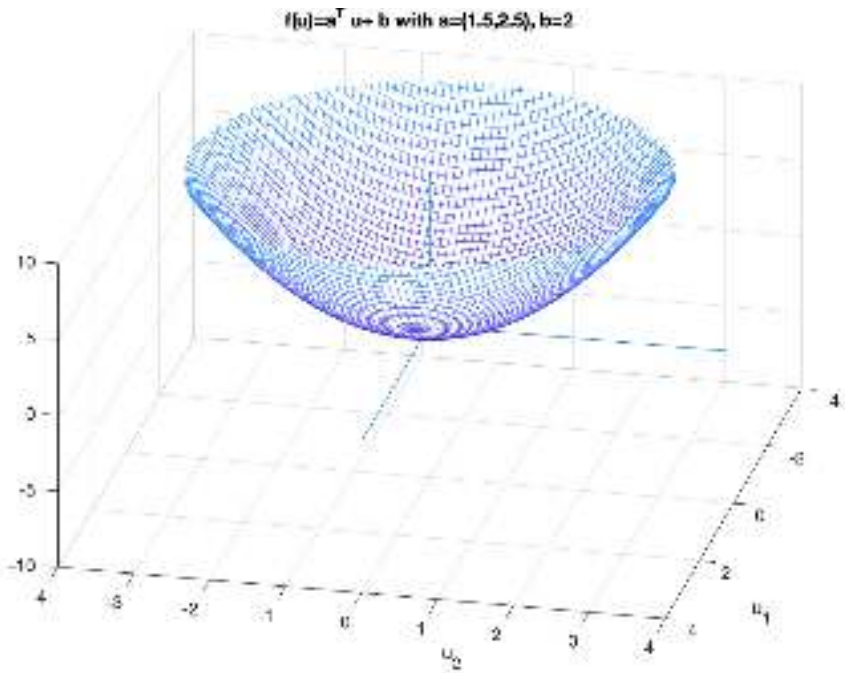
$$\vec{u} \mapsto \vec{a}^\top \vec{u} + b, \quad \vec{a} \in \mathbb{R}^n, b \in \mathbb{R}$$



Fonction convexe : forme quadratique

$$f : \mathbb{R}^n \to \mathbb{R}$$

$$\vec{u} \mapsto \frac{1}{2}\vec{u}^\top Q\vec{u} + \vec{a}^\top \vec{u} + b, \quad \vec{a} \in \mathbb{R}^n, b \in \mathbb{R}, Q \succeq 0$$



Convexe si Q est semi-définie positive
Par exemple :

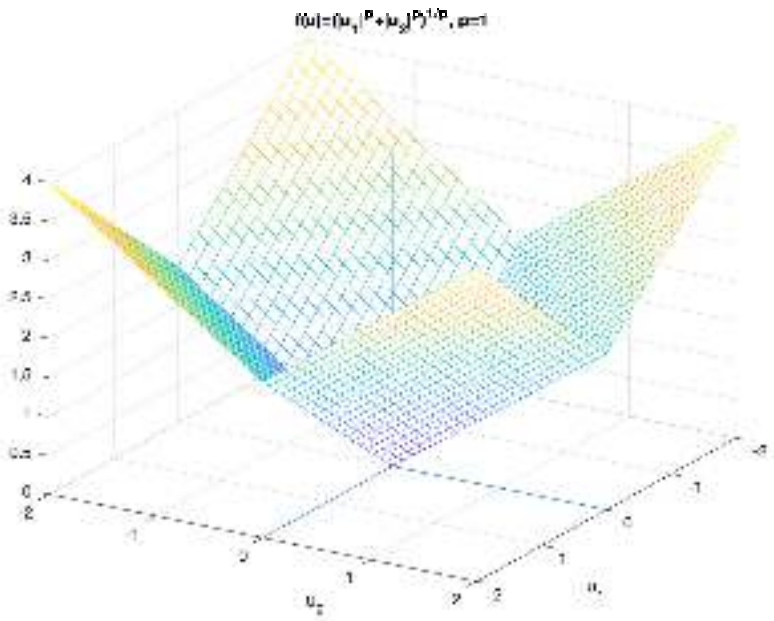
$$Q = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

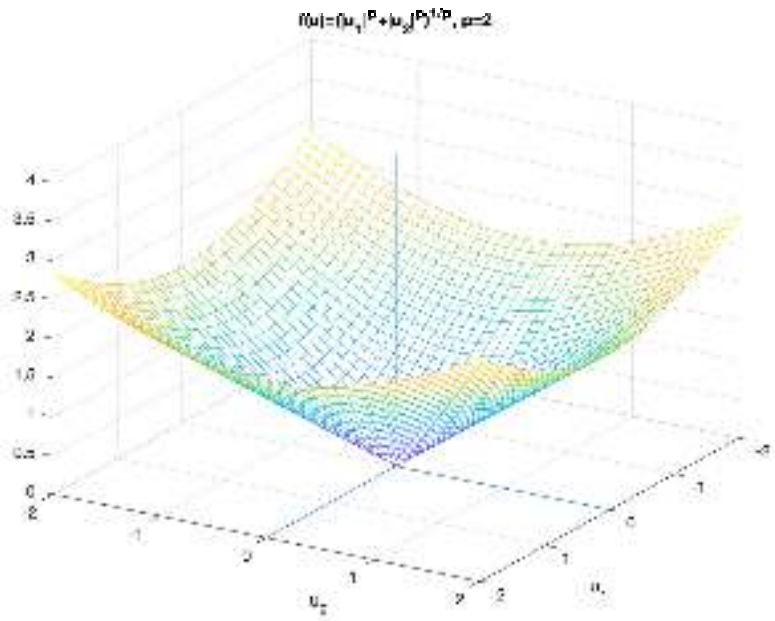
Rappel : une matrice Q de dimension $n \times n$ à valeurs réelles est semi-définie positive si $\vec{u}^\top Q\vec{u} \geq 0 \ \forall \vec{u} \in \mathbb{R}^n \setminus \{\vec{0}\}$.

Fonction convexe : normes

$$f : \mathbb{R}^n \to \mathbb{R}_+$$

$$\vec{u} \mapsto \|\vec{u}\|_p = \left(\sum_{i=1}^{n} |u_i|^p \right)^{1/p} \qquad p \geq 1$$





Optimisation convexe

Optimisation convexe Étant donnés $\mathcal{U} \subseteq \mathbb{R}^n$ (avec n un entier positif) et une fonction $f : \mathcal{U} \to \mathbb{R}$ convexe, on appelle *problème d'optimisation convexe* le problème suivant :

$$\min_{\vec{u} \in \mathcal{U}} f(\vec{u})$$

f : fonction objective/fonction critère

Point minimum de f sur $\mathcal{U}$: un point $\vec{u}^* \in \mathcal{U}$ vérifiant :

$$f(\vec{u}^*) \leq f(\vec{u}) \quad \forall \vec{u} \in \mathcal{U}$$

Minimum de f sur $\mathcal{U}$: la valeur $f(\vec{u}^*)$.

Une fonction convexe est au dessus de sa tangente

Théorème 1.2 Soit $\mathcal{U} \subseteq \mathbb{R}^n$ et une fonction $f : \mathcal{U} \to \mathbb{R}$ de classe $\mathcal{C}^1$. f est convexe si et seulement si :

1. $\mathcal{U}$ est convexe
2. quels que soient $\vec{u}, \vec{v} \in \mathcal{U}$,

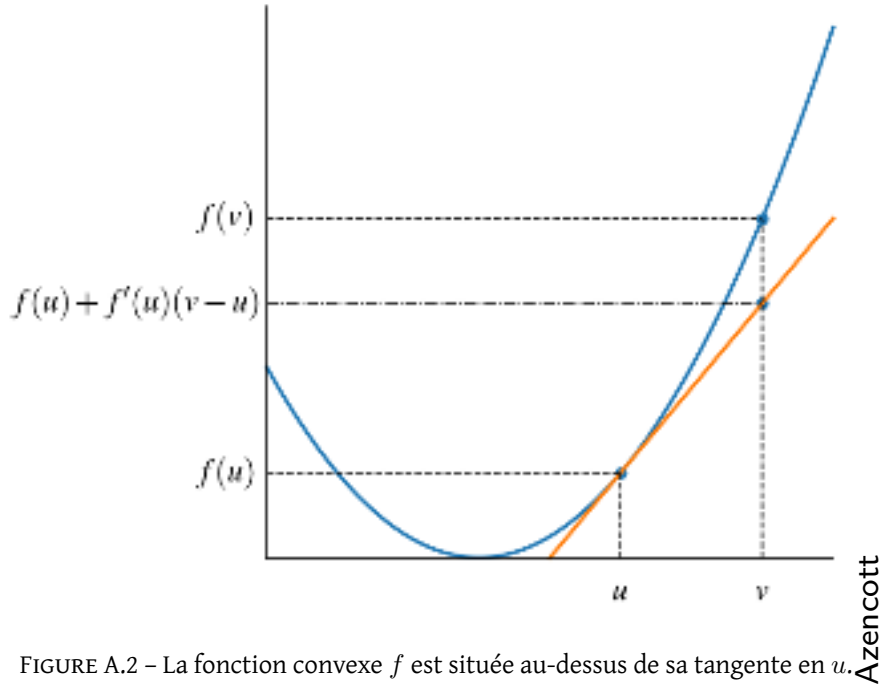


FIGURE A.2 – La fonction convexe f est située au-dessus de sa tangente en u. Àzencott

$$f(\vec{v}) \geq f(\vec{u}) + (\nabla f(\vec{u}))^\top (\vec{v} - \vec{u}) \tag{1}$$

Rappel de notation (vecteur gradient) :

$$\nabla = \begin{pmatrix} \frac{\partial}{\partial u_1} \\ \vdots \\ \frac{\partial}{\partial u_n} \end{pmatrix} \quad \Longrightarrow \quad \nabla f(\vec{u}) = \begin{pmatrix} \frac{\partial f(\vec{u})}{\partial u_1} \\ \vdots \\ \frac{\partial f(\vec{u})}{\partial u_n} \end{pmatrix}$$

Minimum d'une fonction convexe et différentiable

Théorème 1.3 Soit $\mathscr{U} \subseteq \mathbb{R}^n$ et une fonction $f : \mathscr{U} \to \mathbb{R}$ convexe, de classe $\mathscr{C}^1$. Les propositions suivantes sont équivalentes :

1. $\vec{u}^*$ est un point de minimum de f sur $\mathscr{U}$
2. $\nabla f(\vec{u}^*) = \vec{0}_{n\times 1}$.

Algorithme du gradient

Soient $\mathcal{U} \subseteq \mathbb{R}^n$ et $f : \mathcal{U} \to \mathbb{R}$ une fonction de classe $\mathcal{C}^1$. Étant donnés un pas $\alpha > 0$ et une tolérance $\epsilon > 0$ on appelle *algorithme du gradient* l'algorithme suivant :

- Choisir $\vec{u}$ aléatoirement dans $\mathcal{U}$
- Tant que $\|\nabla f(\vec{u})\|_2^2 > \epsilon$, actualiser $\vec{u}$:

$$\vec{u} \leftarrow \vec{u} - \alpha \nabla f(\vec{u}).$$

$\vec{u}$ est alors approximation du point de minimum global de f sur $\mathcal{U}$.

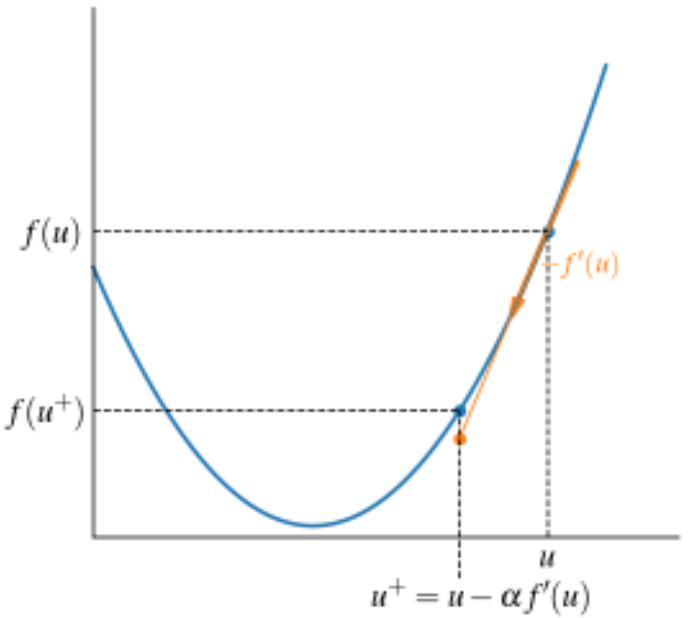
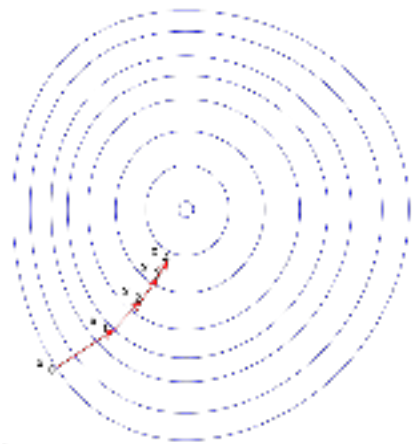


FIGURE A.3 – Une itération de l'algorithme du gradient déplace u de $-\alpha f'(u)$. Azencott



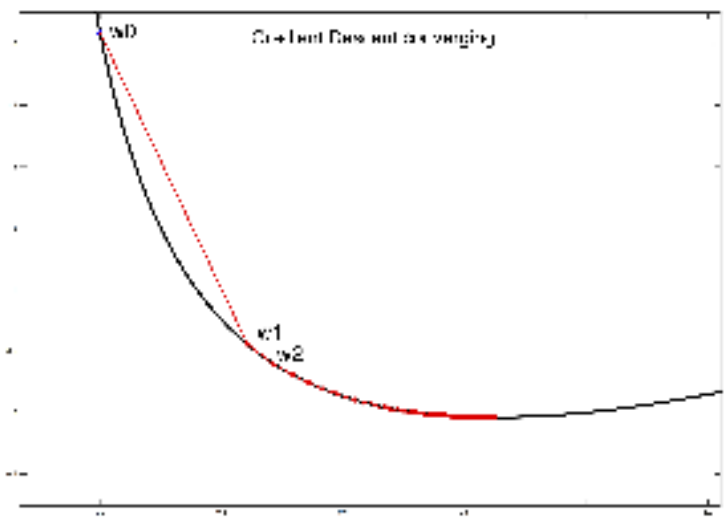
wikipedia.orgGradient_descent

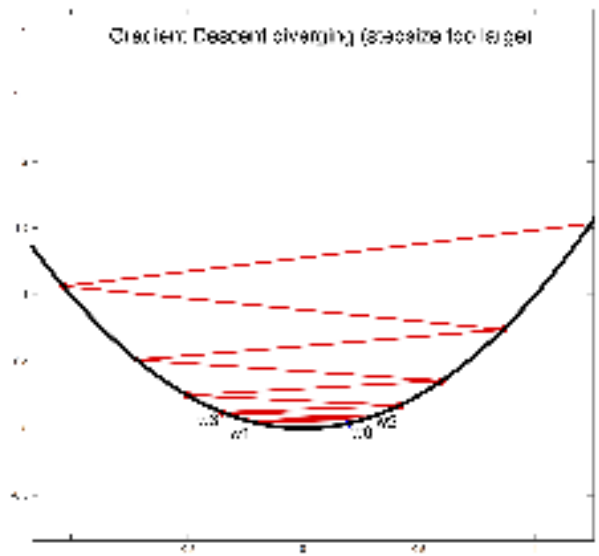
Le vecteur opposé du gradient, $-\nabla f(\vec{u})$, indique la direction de descente la plus raide au point $\vec{u}$.

Algorithme du gradient : remarques

Notes :

- plus la tolérance ϵ est faible : plus le point de minimum trouvé sera proche numériquement du point de minimum global
- si le pas α est faible : convergence lente
- si le pas α est grand : $\vec{u}$ peut osciller autour du minimum global et l'algorithme peut diverger
- on utilise souvent un pas adaptatif : d'abord grand puis qui diminue





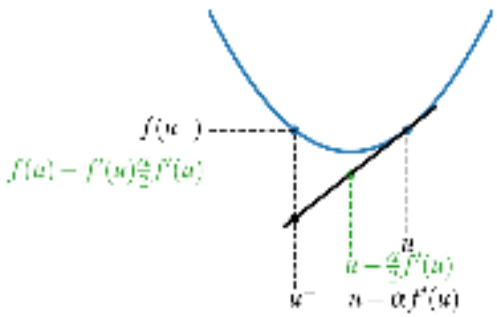
http://www.cs.cornell.edu/courses/cs4780/2017sp/lectures/lecturenote07.html

Recherche linéaire par rebroussement (BLS : Backtracking Line Search) : réduction du pas (rebroussement) s'il induit un coût supérieur à la hauteur de la tangente à un demi-pas

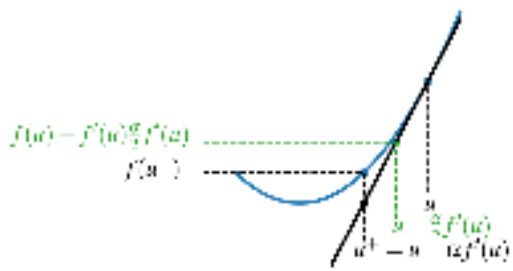
Étant donné un pas initial $\alpha > 0$, un coefficient de réduction $\beta \in]0, 1[$ et une tolérance $\epsilon > 0$:

1. Choisir $\vec{u}$ aléatoirement dans $\mathcal{U}$
2. Tant que $\|\nabla f(\vec{u})\|_2^2 > \epsilon$
   - Si $f(\vec{u} - \alpha \nabla f(\vec{u})) > f(\vec{u}) - \frac{\alpha}{2} \nabla f(\vec{u})^\top \nabla f(\vec{u})$ réduire le pas $\alpha \leftarrow \beta\alpha$ (sinon on garde le même pas)
   - Actualiser $\vec{u}$: $\vec{u} \leftarrow \vec{u} - \alpha \nabla f(\vec{u})$.

$\vec{u}$ est alors approximation du point de minimum global de f sur $\mathcal{U}$.



(A) Quand $f(u - \alpha f'(u)) > f(u) - f'(u)\frac{\alpha}{2}f'(u)$, le pas α est trop élevé et $u - \alpha f'(u)$ va se retrouver de l'autre côté du point de minimum. Il faut donc le réduire.



(B) Quand $f(u - \alpha f'(u)) \leq f(u) - f'(u)\frac{\alpha}{2}f'(u)$, le pas α est suffisamment petit pour que $u - \alpha f'(u)$ soit entre le point de minimum et u.

FIGURE A.4 – Comparer $f(u - \alpha f'(u))$ à $f(u) - f'(u)\frac{\alpha}{2}f'(u)$ permet de déterminer si la valeur de α est trop élevée.

Azencott

Le problème de la descente de gradient

(a)

(b)

Figure 10.6.1. (a) Steepest descent method in a long, narrow "valley." While more efficient than the strategy of Figure 10.5.1, steepest descent is nonetheless an inefficient strategy, taking many steps to reach the valley floor. (b) Magnified view of one step: A step starts off in the local gradient direction, perpendicular to the contour lines, and traverses a straight line until a local minimum is reached, where the traverse is parallel to the local contour lines.

Méthode de Newton (1/2)

Développement de Taylor au deuxième ordre de f en $\vec{u}$.

$$g(\vec{v}) = f(\vec{u}) + \nabla f(\vec{u})^\top (\vec{v} - \vec{u}) + \frac{1}{2}(\vec{v} - \vec{u})^\top \nabla^2 f(\vec{u})(\vec{v} - \vec{u})$$

avec $\nabla^2 f(\vec{u}) = \left[\frac{\partial^2}{\partial u_i \partial u_j} f(\vec{u})\right]_{1 \leq i,j \leq n}$ la matrice hessienne.

On peut minimiser g en annulant son gradient (p.r. à $\vec{v}$) :

$$\nabla g(\vec{v}) = \nabla f(\vec{u}) + \nabla^2 f(\vec{u})(\vec{v} - \vec{u}) = 0$$

pour trouver que g est minimale au point

$$\vec{v}^* = \vec{u} - \left(\nabla^2 f(\vec{u})\right)^{-1} \nabla f(\vec{u})$$

On proposera donc l'actualisation :

$$\vec{u} \leftarrow \vec{u} - \left(\nabla^2 f(\vec{u})\right)^{-1} \nabla f(\vec{u})$$

Par analogie avec la formule de l'actualisation de la méthode du gradient ($\vec{u} \leftarrow \vec{u} - \alpha \nabla f(\vec{u})$), cela correspond à fixer un «pas» :

$\alpha = \left(\nabla^2 f(\vec{u})\right)^{-1}$ $\leftarrow$ mais attention : α est ici bien une matrice, pas un scalaire !

Méthode de Newton (2/2)

Soient $\mathcal{U} \subseteq \mathbb{R}^n$ et $f : \mathcal{U} \to \mathbb{R}$ une fonction de classe $\mathcal{C}^2$. Étant donné une tolérance $\epsilon > 0$, on appelle *méthode de Newton* l'algorithme suivant :

1. Choisir $\vec{u}$ aléatoirement dans $\mathcal{U}$
2. Tant que $\|\nabla f(\vec{u})\|_2^2 > \epsilon$
   - calculer le «pas» $\alpha = \left(\nabla^2 f(\vec{u})\right)^{-1}$
   - Actualiser $\vec{u}$: $\vec{u} \leftarrow \vec{u} - \alpha \nabla f(\vec{u})$.

$\vec{u}$ est alors approximation du point de minimum global de f sur $\mathcal{U}$.

Notes :

• méthode potentiellement gourmande en raison du calcul de l'inverse de la hessienne et de son stockage

• la méthode de Newton produit un chemin différent que la descente de gradient (α est un matrice)

• si f est quadratique : convergence en 1 pas !

Méthode de Newton par gradient conjugué

But : éviter de devoir calculer l'inverse de la hessienne (e.g. quand ses dimensions sont grandes) pour le calcul du pas :

$$\alpha = \left(\nabla^2 f(\vec{u})\right)^{-1}$$

utilisé lors de l'actualisation

$$\vec{u} \leftarrow \vec{u} - \alpha \nabla f(\vec{u}) = \vec{u} - \overbrace{\left(\nabla^2 f(\vec{u})\right)^{-1} \nabla f(\vec{u})}^{\vec{\delta}}$$

En définissant une grandeur $\vec{\delta} = \left(\nabla^2 f(\vec{u})\right)^{-1} \nabla f(\vec{u})$, la règle d'actualisation devient :

$$\vec{u} \leftarrow \vec{u} - \vec{\delta}$$

et on voit aussi que $\vec{\delta}$ est solution d'une équation linéaire (de type $A\vec{x} = \vec{b}$) :

$$\underbrace{\nabla^2 f(\vec{u})}_{A} \underbrace{\vec{\delta}}_{\vec{x}} = \underbrace{\nabla f(\vec{u})}_{\vec{b}}.$$

Comme $A \succeq 0$ (car f convexe) on peut utiliser la méthode du gradient conjugué pour la résoudre (voir page suivante).

Méthode du gradient conjugué (pour résoudre $A\vec{\delta} = \vec{b}$)

Idée : construire une base de $\mathbb{R}^n$ constituée de vecteurs conjugués par rapport à A, i.e. : $\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n\}$ tels que $\vec{v}_i^\top A \vec{v}_j = 0 \quad \forall i \neq j$.

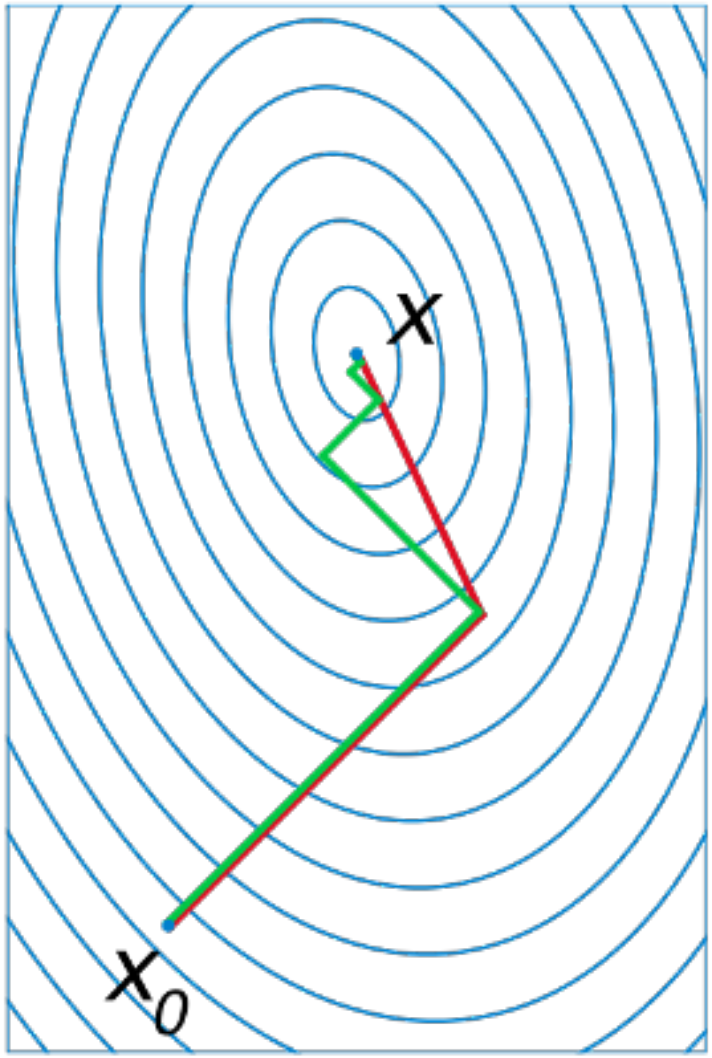
Définition 1.11 (Méthode du gradient conjugué) Étant donnés $A \in \mathbb{R}^{n \times n}$ semi définie positive et $\vec{b} \in \mathbb{R}^n$:

1. Initialisation
   - Choisir aléatoirement $\vec{\delta}^{(0)} \in \mathbb{R}^n$.
   - Initialiser $\vec{r}_0 = \vec{v}_0 = \vec{b} - A\vec{\delta}^{(0)}$
2. Pour $t = 1, \ldots, n$:
   (a) Actualiser $\vec{\delta}^{(t)}$:
   $$\vec{\delta}^{(t)} = \vec{\delta}^{(t-1)} + \frac{\vec{r}_{t-1}^\top \vec{r}_{t-1}}{\vec{v}_{t-1}^\top A \vec{v}_{t-1}} \vec{v}_{t-1}.$$
   (b) Actualiser le résiduel :
   $$\vec{r}_t = \vec{b} - A\vec{\delta}^{(t)}$$
   (c) Actualiser $\vec{v}_t$:
   $$\vec{v}_t = \vec{r}_t + \frac{\vec{r}_t^\top \vec{r}_t}{\vec{r}_{t-1}^\top \vec{r}_{t-1}} \vec{v}_{t-1}.$$

$\vec{\delta}^{(n)}$ est la solution cherchée. (*Note :* résidus perpendiculaires !)

Gradient conjugué : illustration

Minimisation dans le cas d'une fonction quadratique.



wikipedia.org/Conjugate_gradient_method

Comparaison de la convergence de la descente de gradient avec pas optimaux (vert) et gradient conjugué (rouge).
Dans le cas où la fonction est quadratique, le gradient conjugué converge en n pas, au maximum.

Note : ici, la méthode de Newton convergerait en 1 pas seulement, mais celui-ci requiert le calcul (coûteux) de l'inverse de la matrice hessienne.

Algorithme du gradient stochastique (motivation)

Lorsque n est très grand, même le calcul du gradient peut devenir très coûteux. Dans le cas où on a une fonction à minimiser qui peut s'écrire sous la forme :

$$f(\vec{u}) = \sum_{i=1}^{n} f_i(\vec{u}),$$

son gradient se décompose alors aussi :

$$\nabla f(\vec{u}) = \sum_{i=1}^{n} \nabla f_i(\vec{u}),$$

et on peut accélérer les calculs en n'utilisant à chaque itération qu'une seule des fonctions f_i, i.e. on remplace $\sum_{i=1}^{n} \nabla f_i(\vec{u})$ par $\nabla f_k(\vec{u})$.

Algorithme du gradient stochastique

Soient $\mathcal{U} \subseteq \mathbb{R}^n$ et $f : \mathcal{U} \to \mathbb{R}$ une fonction de classe $\mathcal{C}^1$ décomposable sous la forme

$$f(\vec{u}) = \sum_{i=1}^{n} f_i(\vec{u}).$$

Étant donnés un pas $\alpha > 0$ et une tolérance $\epsilon > 0$ on appelle algorithme du gradient stochastique l'algorithme suivant :

1. Choisir $\vec{u}$ aléatoirement dans $\mathcal{U}$
2. Tant que $\|\nabla f(\vec{u})\|_2^2 > \epsilon$:
    - choisir k aléatoirement parmi $\{1, 2, \ldots, n\}$
    - actualiser $\vec{u}$: $\vec{u} \leftarrow \vec{u} - \alpha \nabla f_k(\vec{u})$

$\vec{u}$ est alors une approximation du point de minimum global de f sur $\mathcal{U}$.

Descente de coordonnées

Soient $\mathcal{U} \subseteq \mathbb{R}^n$ et $f : \mathcal{U} \to \mathbb{R}$ une fonction de la forme

$$f : \vec{u} \mapsto g(\vec{u}) + \sum_{i=1}^{n} h_i(u_i).$$

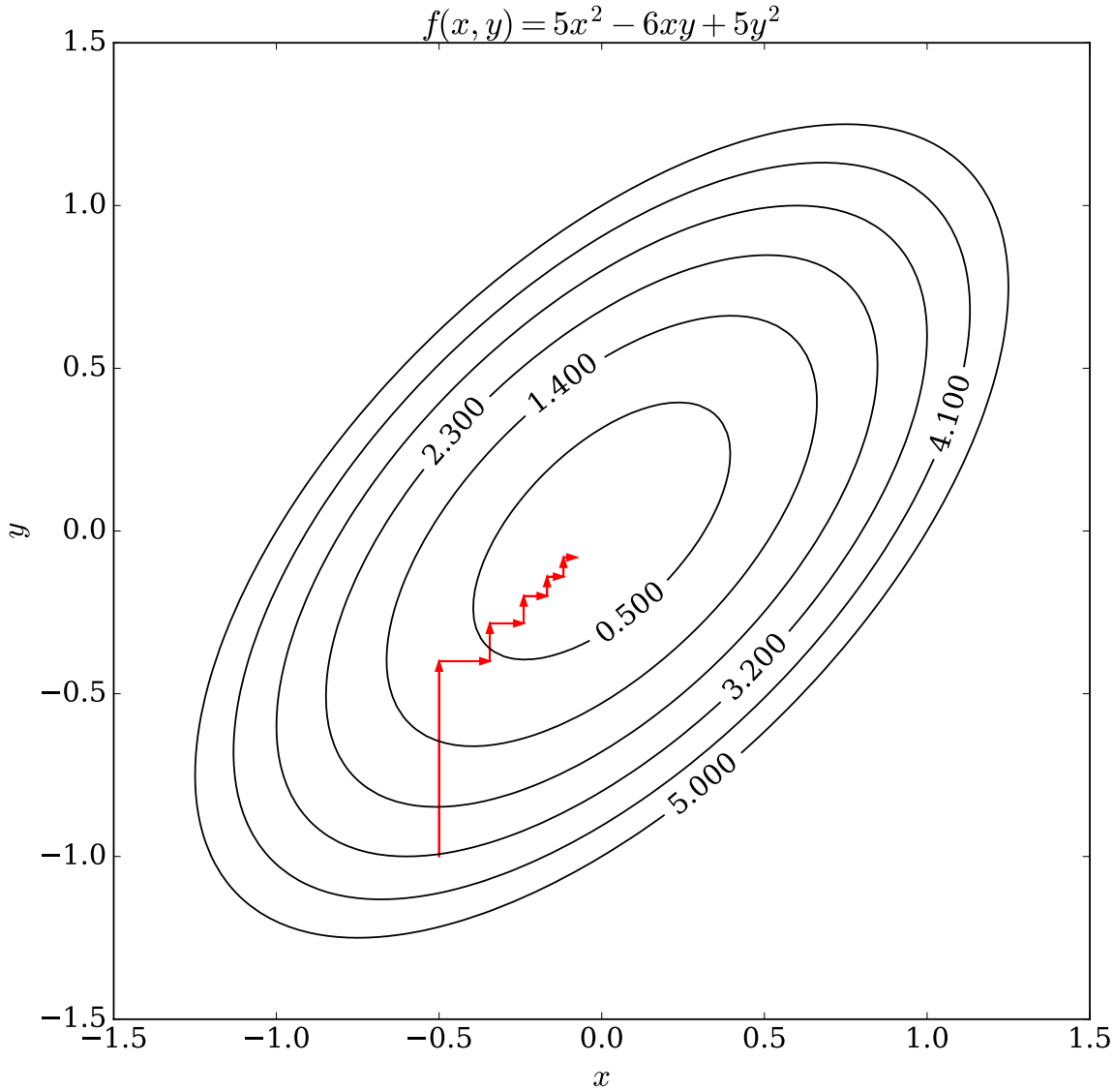
où g est une fonction convexe de classe $\mathcal{C}^1$ et les n fonctions h_i sont convexes.

Algorithme : descente de coordonnées (coordinate descent)

1. Choisir $\vec{u}$ aléatoirement dans $\mathcal{U}$
2. Tant que $\|\nabla f(\vec{u})\|_2^2 > \epsilon$, actualiser $\vec{u}$:
   - $u_1^{(t)}$ point minimal de $u \mapsto f\left(u, u_2^{(t-1)}, \ldots, u_n^{(t-1)}\right)$
   - $u_2^{(t)}$ point minimal de $u \mapsto f\left(u_1^{(t-1)}, u, \ldots, u_n^{(t-1)}\right)$
   - $\ldots$
   - $u_n^{(t)}$ point minimal de $u \mapsto f\left(u_1^{(t-1)}, u_1^{(t-1)}, \ldots, u\right)$

$\vec{u}$ est alors une approximation du point de minimum global de f sur $\mathcal{U}$.

Descente de coordonnées (illustration)



https://en.wikipedia.org/wiki/Coordinate_descent

Régressions paramétriques

Régression paramétrique : la forme analytique de la fonction de décision est connue.
Exemple : Régression linéaire : simplicité, interprétation facile, implémentable même avec jeu de données de taille modeste

Apprentissage supervisé d'un modèle paramétrique Dans le cadre d'un *modèle paramétrique* on utilise un algorithme d'apprentissage pour trouver les valeurs optimales des paramètres d'un modèle dont on a défini la forme analytique en fonction de descripteurs.

Complexité des modèles paramétriques comparée aux modèles non paramétriques :

- paramétrique : complexité grandit avec le nombre de paramètres à apprendre (variables)
- non paramétrique : complexité grandit avec le nombre d'observations

Exemple : Modèle paramétrique comparé à modèle non paramétrique

Exemple 1 : apprentissage des coefficients α, β, γ dans la fonction de décision $f : \vec{x} \mapsto \alpha x_1 + \beta x_2^2 x_4^2 + \gamma e^{x_3 - x_5}$ est paramétrique. Quel que soit le nombre d'observations, ce modèle ne change pas !

Exemple 2 : méthode du plus proche voisin (qui associe à $\vec{x}$ l'étiquette du point du jeu d'entraînement dont il est le plus proche (en distance euclidienne)) apprend un modèle non paramétrique : on ne sait pas écrire la fonction de décision comme une fonction des variables prédictives. Plus il y a d'observations, plus le modèle pourra apprendre une frontière de décision complexe.

Modèle paramétrique et erreurs (utilisation justifiée d'une fonction de perte quadratique)

On se donne un jeu $\mathscr{D} = \left\{\vec{x}^i, y^i\right\}$ de n observations en p dimensions et leur étiquettes réelles.

On suppose que la fonction de décision f est paramétrée par le vecteur $\vec{\beta} \in \mathbb{R}^m$.

Hypothèse : erreurs (différence entre étiquettes réelles et les valeurs correspondantes de f) sont normalement distribuées, centrées en 0 :

$$y = f(\vec{x}|\vec{\beta}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

Avec ce modèle, les observations $\vec{x}$ sont les réalisation de p variables aléatoires $X_1, X_2, \ldots, X_p$ à valeurs réelles qui vérifient :

$$\mathbb{P}(Y = y|X = \vec{x}) \sim \mathcal{N}\left(f\left(\vec{x}|\vec{\beta}\right), \sigma^2\right)$$

avec $\mathbb{P}(X = \vec{x})$ pour $\mathbb{P}(X_1 = x_1, X_2 = x_2, \ldots, X_p = x_p)$.

Modèle et hypothèse d'erreur : illustration

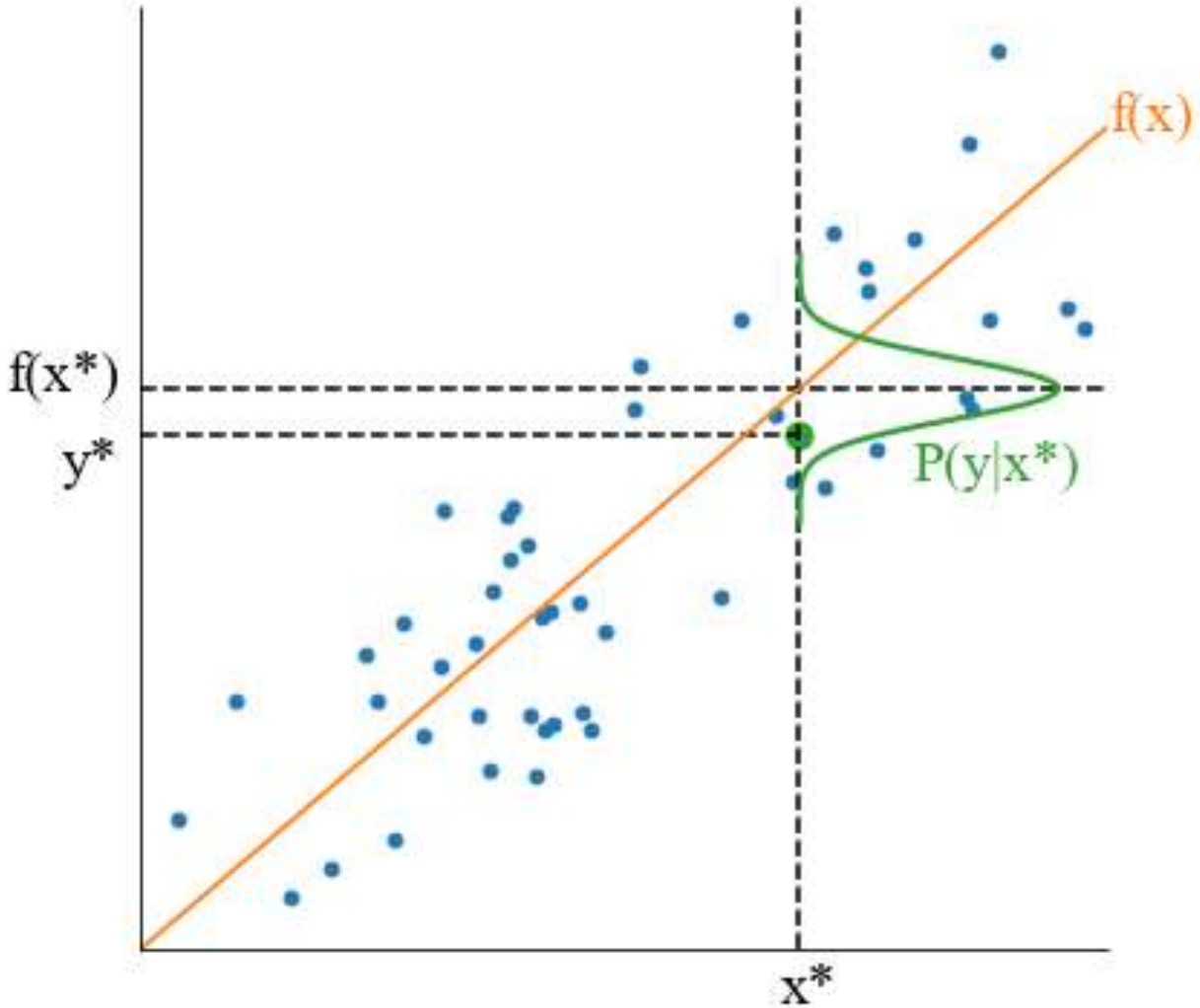


FIGURE 5.1 – Pour une observation x^* donnée (ici en une dimension) , la distribution des valeurs possibles de l'étiquette y^* correspondante est une gaussienne centrée en $f(x^*)$.

Azencott

Estimation par maximum de vraisemblance et méthode des moindres carrés

Sous l'hypothèse

$$\mathbb{P}(Y = y|X = \vec{x}) \sim \mathcal{N}\left(f\left(\vec{x}|\vec{\beta}\right), \sigma^2\right)$$

et en supposant que les n observations sont indépendantes et identiquement distribuées, le log de vraisemblance du paramètre $\vec{\beta}$ vaut :

$$\begin{aligned}
\log \mathbb{P}(\mathscr{D}|\vec{\beta}) &= \log \prod_{i=1}^{n} \mathbb{P}(X = \vec{x}^i|\vec{\beta}) \\
&= \log \prod_{i=1}^{n} \mathbb{P}(y^i|\vec{x}^i) + \log \prod_{i=1}^{n} \mathbb{P}(X = \vec{x}^i) \\
&= -\log \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y^i - f(\vec{x}^i|\vec{\beta})\right)^2 + C
\end{aligned}$$

avec C une constante par rapport à $\vec{\beta}$ (vient du coefficient $\frac{1}{2\sigma^2}$ et des $\mathbb{P}(X = \vec{x}^i)$).

Minimisation des moindres carrés

Maximiser la vraisemblance

$$-\log \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y^i - f(\vec{x}^i | \vec{\beta}) \right)^2 + C$$

revient donc à minimiser

$$\sum_{i=1}^{n} \left(y^i - f(\vec{x}^i | \vec{\beta}) \right)^2$$

i.e. minimisation des moindres carrés (Gauss, Legendre)
$\Rightarrow$ Utiliser une fonction de coût quadratique est par conséquent justifié lorsque l'on sait que les mesures sont sujettes à un bruit additif ϵ dont les valeurs ont moyenne nulle et sont distribuées selon une loi normale, $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

Régression linéaire

Modèles paramétriques modèle paramétrique si le but de l'algorithme d'apprentissage est de trouver les valeurs optimales des paramètres d'un modèle dont on a défini la forme analytique.

Fonction de décision

$$f : \vec{x} \mapsto \beta_0 + \sum_{j=1}^{p} \beta_j x_j = \begin{pmatrix} 1 & x_1 & \dots & x_p \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} = \begin{pmatrix} 1 & \vec{x} \end{pmatrix} \vec{\beta}$$

avec $\vec{\beta} \in \mathbb{R}^m$ et $m = p + 1$. Nombre de variables : p.

Régression linéaire (définition du problème) : c'est le modèle de la forme $f : \vec{x} \mapsto \beta_0 + \sum_{j=1}^{p} \beta_j x_j$ dont les coefficients sont obtenus par :

$$\vec{\beta}^* = \arg \min_{\vec{\beta} \in \mathbb{R}^{p+1}} \sum_{i=1}^{n} \left(y^i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_j^i \right) \right)^2$$

Le problème de régression linéaire a une solution explicite :

Si on ajoute une colonne de 1 à la matrice d'observation :

$$X = \begin{pmatrix} 1 & x_1^1 & \cdots & x_p^1 \\ \vdots & & & \vdots \\ 1 & x_1^n & \cdots & x_p^n \end{pmatrix}$$

On peut réécrire la somme des résidus quadratiques (Residual Sum of Squares) :

$$\text{RSS} = \sum_{i=1}^{n} \left(y^i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_j^i \right) \right)^2 = \left(\vec{y} - X\vec{\beta} \right)^\top \left(\vec{y} - X\vec{\beta} \right)$$

On peut minimiser cette forme quadratique en calculant son gradient :

$$\nabla_{\vec{\beta}} \text{RSS} = -2X^\top \left(\vec{y} - X\vec{\beta} \right)$$ (voir dérivation en annexe)

et en posant $\nabla_{\vec{\beta}} \text{RSS} = \vec{0}_{m \times 1}$ on obtient : $X^\top X \vec{\beta}^* = X^\top \vec{y}$

Si X est de rang égal à son nombre de colonnes (m), on a :

$$\vec{\beta}^* = \left(X^\top X \right)^{-1} X^\top \vec{y}$$ ← solution analytique !

Régression logistique

But : dans un problème de classification binaire, étant donné les variables de $\vec{x}$, trouver la classe correspondante $y \in \{0, 1\}$.

Exemple : x le nombre d'heures passées à travailler pour un examen et y si l'examen a été réussi ou non (pass/fail).

Modèle : on impose un modèle probabiliste qui exprime

$$\mathbb{P}(Y = y \,|X = \vec{x}),$$

probabilité d'appartenance à la classe y connaissant la variable $\vec{x}$.

Régression logistique, suite

Contrainte pour le modèle :

- la probabilité $\mathbb{P}(Y = y \,|X = \vec{x})$ doit être comprise entre 0 et 1
- on veut une fonction de décision $f : \mathbb{R} \to [0, 1]$
- la fonction doit faire intervenir une combinaison linéaire des variables dans $\vec{x}$
- une légère perturbation de $\vec{x}$ lorsque sa probabilité d'appartenance à une classe est proche de 0 ou 1 devrait avoir peu d'influence sur l'estimation de la nouvelle probabilité
- une légère perturbation de $\vec{x}$ lorsque sa probabilité d'appartenance à une classes est incertaine, e.g. 0.5, pourrait affecter l'estimation plus fortement

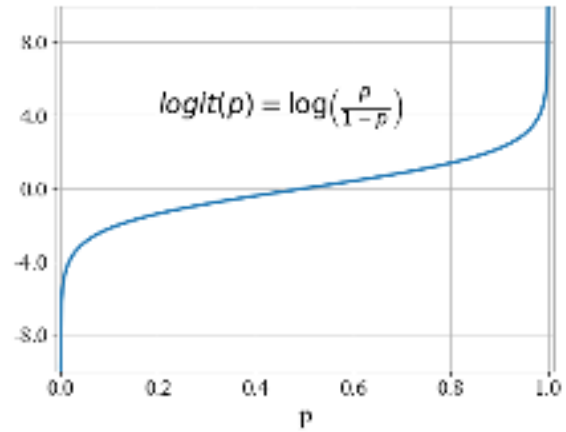
Définition des fonctions logit et logistique

Définition 5.3 (Fonction logit)

On appelle fonction logit la fonction :

$$\text{logit} : [0,1] \to \mathbb{R}$$

$$p \mapsto \log \frac{p}{1-p}$$



(A) Fonction logit.

Azencott

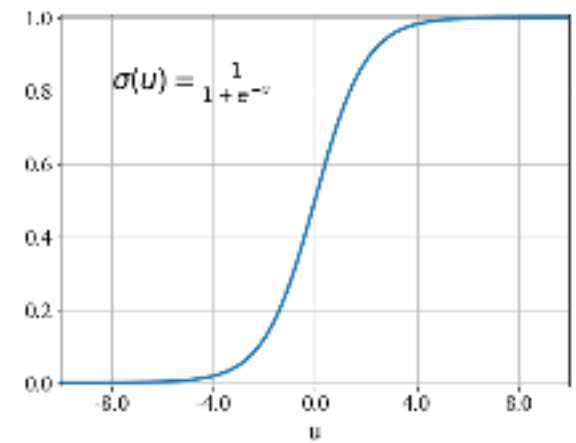
La fonction inverse/réciproque :

Définition 5.4 (Fonction logistique)

On appelle fonction logistique la fonction :

$$\sigma : \mathbb{R} \to [0,1]$$

$$u \mapsto \frac{1}{1+e^{-u}} = \frac{e^u}{1+e^u}$$



(B) Fonction logistique.

Azencott

Fonctions logit et logistique

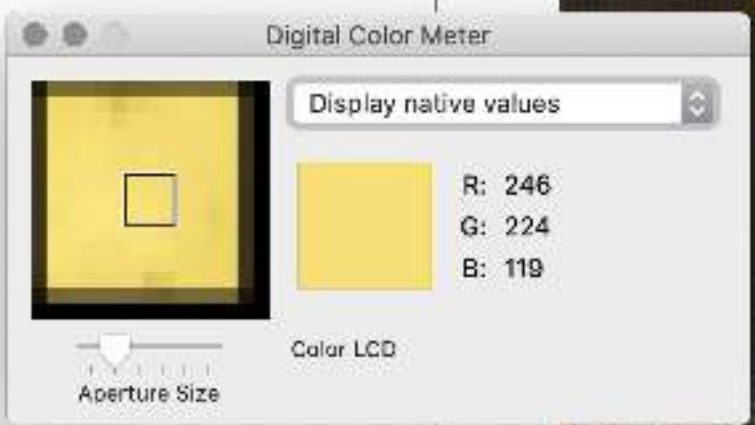
Idée : plutôt que d'essayer de modéliser directement la probabilité par une fonction linéaire, on modélise la *transformation logit* de $\mathbb{P}(Y = 1 \,|\vec{x})$ comme une combinaison linéaire, i.e. une fonction de $(1\ \vec{x})\,\vec{\beta} = \beta_0 + \sum_{j=1}^{p} x_j \beta_j$, soit :

$$\log \frac{\mathbb{P}(Y = 1 \,|\vec{x})}{1 - \mathbb{P}(Y = 1 \,|\vec{x})} = (1\ \vec{x})\,\vec{\beta}$$

De manière équivalente, on peut exprimer la probabilité comme une fonction logistique dont l'argument est une fonction linéaire :

$$\mathbb{P}(Y = 1 \,|\vec{x}) = \sigma\left((1\ \vec{x})\,\vec{\beta}\right)$$

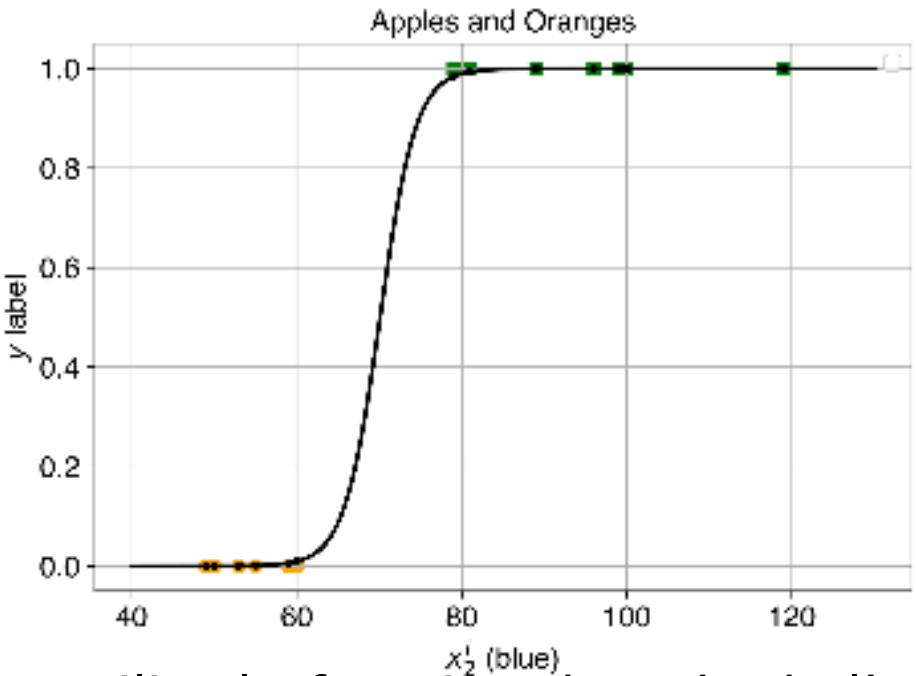
Apples and Oranges : valeurs RGB (classification binaire sur le canal bleu x_2)



On a deux classes, oranges et pommes : $\mathcal{Y} = \{0, 1\}$

On veut une fonction de décision qui indique la probabilité que l'objet soit une pomme ($Y = 1$) étant donné la valeur du canal bleu x_2 dans l'image, et qui prenne la forme :

$$f(\vec{x}) = \mathbb{P}(Y = 1 \,|\vec{x}) = \sigma\left((1\ \vec{x})\,\vec{\beta}\right) = \sigma(\beta_0 + \beta_1 x_2)$$



On cherche un "bon" $\vec{\beta}$:
$f(\vec{x}) =$
$\sigma(-32.426 + 0.464 x_2)$,
en particulier,

on utilise la fonction de coût de l'entropie croisée (cross-entropy) :

$$L_H : \{0, 1\} \times]0, 1[\to \mathbb{R}$$
$$y, f(\vec{x}) \mapsto -y \log f(\vec{x}) - (1 - y) \log(1 - f(\vec{x}))$$

Régression logistique

Définition 5.5 Régression logistique. On appelle régression logistique le modèle $f : x \mapsto \sigma\left((1\ \vec{x})\,\vec{\beta}\right)$ dont les coefficients $\vec{\beta}$ sont obtenus tels qu'ils maximisent la vraissemblance :

$$\arg\max_{\vec{\beta}\in\mathbb{R}^{p+1}} \sum_{i=1}^{n} y^i \log\sigma\left((1\ \vec{x^i})\,\vec{\beta}\right) + (1-y^i)\log\left(1-\sigma\left((1\ \vec{x^i})\,\vec{\beta}\right)\right)$$

Note : Maximiser la vraisemblance de $\vec{\beta}$ sous ce modèle est équivalent à minimiser le risque empirique défini en utilisant la fonction de coût logistique (c.-à-d. l'entropie croisée, comme $\mathcal{Y} = \{0, 1\}$) :

$$\begin{aligned}\mathscr{R}_n(h) &= \frac{1}{n}\sum_{i=1}^{n} L_H(f(\vec{x^i}), y^i) \\ &= \frac{1}{n}\sum_{i=1}^{n} -y^i \log f(\vec{x^i}) - (1-y^i)\log\left(1-f(\vec{x^i})\right)\end{aligned}$$

Solution de la régression logistique

La vraisemblance de la régression logistique est une fonction concave et le gradient en $\vec{\beta}$ de la vraisemblance pour la régression logistique vaut :

$$\sum_{i=1}^{n} \left(y^i - \frac{1}{1 + e^{-(1\ \vec{x}^i)\vec{\beta}}} \right) (1\ \vec{x}^i)$$

- Ce gradient ne peut pas être annulé de manière analytique : pas de solution explicite pour la régression logistique
- solution obtenue par l'algorithme du gradient (ou une de ses variantes)
- convergence vers la solution optimale car la vraisemblance est concave (pas de maximum local)
- De manière équivalente, le coût est convexe et a pour gradient :

$$-\sum_{i=1}^{n} \left(y^i - \frac{1}{1 + e^{-(1\ \vec{x}^i)\vec{\beta}}} \right) (1\ \vec{x}^i)$$

Régression Polynomiale

Dans le cas de la régression polynomiale de degré d, on cherche une fonction de décision de la forme :

$$f : \vec{x} \mapsto \beta_{00} + \sum_{j=1}^{p} \beta_{1j} x_j + \sum_{j=1}^{p} \beta_{2j} (x_j)^2 + \cdots + \sum_{j=1}^{p} \beta_{dj} (x_j)^d$$

Il s'agit en fait d'une régression linéaire sur $p \times d$ variables :

$$\begin{aligned}
&x_1, x_2, \ldots, x_p, \\
&(x_1)^2, (x_2)^2, \ldots, (x_p)^2, \\
&\ldots \\
&(x_1)^d, (x_2)^d, \ldots, (x_p)^d,
\end{aligned}$$

Régression polynomiale (suite)

Fixons $p = 1$, $d = 2$:

$$f : \vec{x} \mapsto \beta_{00} + \beta_{11} x_1 + \beta_{21} (x_1)^2$$

Fixons ensuite $n = 4$, on peut alors construire la matrice :

$$X = \begin{pmatrix} 1 & x_1^1 & (x_1^1)^2 \\ 1 & x_1^2 & (x_1^2)^2 \\ 1 & x_1^3 & (x_1^3)^2 \\ 1 & x_1^4 & (x_1^4)^2 \end{pmatrix}$$

et avec le vecteur d'étiquettes $\vec{y}$ et celui de paramètres $\vec{\beta}$:

$$\vec{y} = \begin{pmatrix} y^1 \\ y^2 \\ y^3 \\ y^4 \end{pmatrix} \quad \vec{\beta} = \begin{pmatrix} \beta_{00} \\ \beta_{11} \\ \beta_{21} \end{pmatrix}$$

on a alors bien une expression **linéaire** pour les résidus, identique à la régression linéaire :

$$\text{RSS} = \left(\vec{y} - X\vec{\beta}\right)^\top \left(\vec{y} - X\vec{\beta}\right)$$

Résumé du cours 3

- méthodes d'optimisation : nécessaires à la résolution numérique de problèmes de minimisation de coût
- convexité : assure convergence vers solution unique
- gradient : suivre la pente la plus forte, avec un pas à déterminer
    - rebroussement
    - Newton (requiert calcul de la matrice hessienne et de son inverse)
    - gradient conjugué (requiert hessienne mais pas son inverse)
    - gradient stochastique : fonction coût sous forme de somme, minimisation partielle
    - descente de coordonnées (fixer toutes les variables sauf une)
- Régressions paramétriques
    - correspondes à maximisation de la vraissemblance des paramètres du modèle étant donné des observations
    - minimisation des moindes carrés (régression linéaire)
    - régression logistique : fonction non-linéaire, modélisation de probabilités
    - régression polynomiale : c'est une régression linéaire !

Guide de lecture pour ce cours

Chloé-Agathe Azencott “Introduction au Machine Learning”,
Dunod, 2019, ISBN 978-210-080153-4
Appendice A : Notions d’optimisation convexe
Chapitre 5 : Régressions paramétriques

- 5.1 Apprentissage supervisé d’un modèle paramétrique
- 5.2 Régression linéaire
- 5.3 Régression logistique
- 5.4 Régression polynomiale

Dérivation solution des moindres carrés pour problème régression linéaire donné sous forme matricielle (new)

On veut trouver le vecteur $\vec{\beta}$ tel qu'il minimise la somme des résidus quadratiques (residual sum of squares) :

$$\text{RSS} = \sum_{i=1}^{n} \left(y^i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_j^i \right) \right)^2 = \left(\vec{y} - X\vec{\beta} \right)^\top \left(\vec{y} - X\vec{\beta} \right)$$

On développe le produit scalaire :

$$\vec{y}^\top \vec{y} - \vec{y}^\top X\vec{\beta} - \underbrace{\vec{\beta}^\top X^\top \vec{y}}_{\vec{y}^\top X\vec{\beta}} + \vec{\beta}^\top X^\top X\vec{\beta} = \underbrace{\vec{y}^\top \vec{y}}_{\text{const.}} - 2\vec{y}^\top X\vec{\beta} + \vec{\beta}^\top X^\top X\vec{\beta}$$

Fixons $n = 2$ et $p = 1$ (et $m = p + 1 = 2$) :

$$X = \begin{pmatrix} 1 & x_1^1 \\ 1 & x_1^2 \end{pmatrix}_{n \times m = 2 \times 2} \qquad \vec{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}_{m \times 1 = 2 \times 1} \qquad \vec{y} = \begin{pmatrix} y_0 \\ y_1 \end{pmatrix}_{n \times 1 = 2 \times 1}$$

Dérivation des moindres carrés sous forme matricielle 2(new)

L'opérateur gradient est :

$$\nabla_{\vec{\beta}} = \begin{pmatrix} \frac{\partial}{\partial \beta_0} \\ \frac{\partial}{\partial \beta_1} \end{pmatrix}$$

Développons le terme croisé :

$$\vec{y}^\top X \vec{\beta} = \begin{pmatrix} y^1 & y^2 \end{pmatrix} \begin{pmatrix} \beta_0 + x_1^1 \beta_1 \\ \beta_0 + x_1^2 \beta_1 \end{pmatrix} = y^1(\beta_0 + x_1^1 \beta_1) + y^2(\beta_0 + x_1^2 \beta_1)$$

et calculons ses dérivées par rapport à β_0 :

$$\frac{\partial}{\partial \beta_0}(\vec{y}^\top X \vec{\beta}) = y^1 + y^2 = \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} y^1 \\ y^2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \end{pmatrix} \vec{y}$$

ainsi que par rapport à β_1 :

$$\frac{\partial}{\partial \beta_1}(\vec{y}^\top X \vec{\beta}) = y^1 x^1 + y^2 x^2 = \begin{pmatrix} x_1^1 & x_1^2 \end{pmatrix} \begin{pmatrix} y^1 \\ y^2 \end{pmatrix} = \begin{pmatrix} x_1^1 & x_1^2 \end{pmatrix} \vec{y}$$

d'où on peut identifier l'expression matricielle :

$$\nabla_{\vec{\beta}}(\vec{y}^\top X \vec{\beta}) = X^\top \vec{y}$$

Dérivation des moindres carrés sous forme matricielle 3(new)

Développons ensuite le terme quadratique :

$$\vec{\beta}^\top \underbrace{X^\top X}_{C} \vec{\beta} = \begin{pmatrix} \beta_0 & \beta_1 \end{pmatrix} \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = c_{11}\beta_0^2 + 2c_{12}\beta_0\beta_1 + c_{22}\beta_1^2$$

(où l'on a utilisé le fait que $c_{12} = c_{21}$) et calculons les dérivées par rapport à β_0 et β_1 :

$$\frac{\partial}{\partial \beta_0}(\vec{\beta}^\top X^\top X \vec{\beta}) = 2c_{11}\beta_0 + 2c_{12}\beta_1 = 2 \begin{pmatrix} c_{11} & c_{12} \end{pmatrix} \vec{\beta}$$

$$\frac{\partial}{\partial \beta_1}(\vec{\beta}^\top X^\top X \vec{\beta}) = 2c_{12}\beta_0 + 2c_{22}\beta_1 + = 2 \begin{pmatrix} c_{12} & c_{22} \end{pmatrix} \vec{\beta}$$

on peut à nouveau mettre ces expressions sous forme matricielle :

$$\nabla_{\vec{\beta}}(\vec{\beta}^\top X^\top X \vec{\beta}) = 2C\vec{\beta} = 2(X^\top X)\vec{\beta}$$

Dérivation des moindres carrés sous forme matricielle 4 (new)

Finalement, on utilise les deux expressions calculées :

$$\nabla_{\vec{\beta}}(\vec{y}^\top X\vec{\beta}) = X^\top \vec{y}$$

$$\nabla_{\vec{\beta}}(\vec{\beta}^\top X^\top X\vec{\beta}) = 2(X^\top X)\vec{\beta}$$

pour simplifier l'expression et obtenir, finalement :

$$\begin{aligned}\nabla_{\vec{\beta}}\text{RSS} &= \nabla_{\vec{\beta}}\left(\underbrace{\vec{y}^\top\vec{y}}_{\text{const.}} - 2\vec{y}^\top X\vec{\beta} + \vec{\beta}^\top X^\top X\vec{\beta}\right) \\ &= -2X^\top\left(\vec{y} - X\vec{\beta}\right)\end{aligned}$$

Pour n et p plus grands, l'expression matricielle reste la même et on pourrait prendre la même approche que dans notre cas particulier pour le montrer.