Optimisation pour l'apprentissage automatique

09/12/2024

Liens utiles

clement.royer@lamsade.dauphine.fr

https://www.lamsade.dauphine.fr/~croyer/coursOAA.html

Aujourd'hui

Introduction à l'optimisation pour l'apprentissage

Convexité

Principes de base de descente de gradient

Logistique

- 3 semaines de cours
  - → Cette semaine : 3 séances, 2 CTD + 1 pratique
    - → SGD
  - → la semaine prochaine : 2 séances, 1 cours + 1 pratique
    - → Adam, Adagrad, etc.
  - → Dernière semaine (22/01 - 23/01) : 2 CTD + 1 pratique
    - → Autodiff, regularisation, L-BFGS, ...
- Intervenants : Clément ROYER (+ Paul CAILLON ?)
- Notation :
  - Examen : 30/01, 10h - 12h, 60 % de la note
  - Projet : Deadline ~ Fin février, prolongement des TP, 40 % de la note
- Emploi du temps :

  9h - 12h15

1) Introduction

→ But du cours : Etudier les algorithmes d'optimisation utilisés en apprentissage (ML, Deep Learning, etc)

→ Problème type : Minimisation du risque empirique (ERM)

Données : $\{(x_i, y_i)\}_{i=1..m}$

x_i : entrées (vecteurs, matrices, ...)

y_i : sorties (vecteurs, réels, classes)

Objectif : Trouver un modèle qui colle aux données, c'est-à-dire qui associe chaque x_i au y_i correspondant.

⇒ Dans ce cours, un modèle sera une fonction $h(\cdot; w) : x \longmapsto h(x; w)$ paramétrée par un vecteur de paramètres $w \in \mathbb{R}^d$

⇒ On cherche le meilleur modèle → Optimisation pour nos données et relativement à une certaine fonction de perte (loss)

Problème d'optimisation

$$\min_{w \in \mathbb{R}^d} f(w) = \frac{1}{m} \sum_{i=1}^{m} \ell\left(h(x_i; w), y_i\right)$$

- $f(w)$: fonction objectif
- ℓ : fonction de perte
- x_i, y_i : Données
- $h(x_i; w)$: Modèle (paramétré par w)
- $\frac{1}{m} \sum_{i=1}^{m}$: Moyenne empirique
- $\min_{w \in \mathbb{R}^d}$: minimiser
- On cherche la plus petite valeur possible de $f(w)$
- $w \in \mathbb{R}^d$: Paramètres / Variables du problème, sur lesquelles on peut agir

Exemples

① Régression linéaire

$x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, $h(x;w) = x^T w = \sum_{j=1}^{d} [x]_j [w]_j$

$\ell(h,y) = \frac{1}{2}(h-y)^2$ (perte aux moindres carrés / ℓ_2)

Problème :

$$\underset{w \in \mathbb{R}^d}{\text{minimiser}} \quad \frac{1}{2n} \sum_{i=1}^{n} (x_i^T w - y_i)^2 = \frac{1}{2n} \|Xw - y\|^2$$

avec $X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \in \mathbb{R}^{n \times d}$, $y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$

et $\|v\|^2 = \sum_{i=1}^{n} [v]_i^2 \quad \forall v \in \mathbb{R}^n$

② SVM Linéaires (SVM = Support Vector Machines)

$$\underset{w \in \mathbb{R}^d}{\text{minimiser}} \quad \frac{1}{n} \sum_{i=1}^{n} \max(1 - y_i x_i^T w, 0)$$

avec $x_i \in \mathbb{R}^d$, $h(x,w) = x^T w$, $y_i \in \{-1, 1\}$

$\ell(h,y) = \max(1 - yh, 0)$ "Hinge loss"

Si $h >> 1$ et $y = +1$, $\ell(h,y) = 0$

Si $h << -1$ et $y = -1$, $\ell(h,y) = 0$

Sinon $\ell(h,y) > 0$

(3) Régression logistique $\quad h(\cdot;w) : x \mapsto x^T w$

$x_i \in \mathbb{R}^d$, $h(x;w) = x^T w$, $y_i \in \{-1, 1\}$

$$\ell(h, y) = \log(1 + \exp(-y h))$$

Problème : $\quad \underset{w \in \mathbb{R}^d}{\text{minimiser}} \; \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i x_i^T w\right)\right)$

Idée : $x_i^T w \gg 1$ et $y_i = +1$

$\log(1 + \exp(-y_i x_i^T w)) \simeq 0$

(4) "0/1 loss"

$y_i \in \mathcal{Y} \quad \forall i = 1 \dots n \qquad h(\cdot; w) : x \mapsto h(x; w) \in \mathcal{Y}$

$$\ell(h, y) = \mathbb{1}(h \neq y) = \begin{cases} 1 & \text{si } h \neq y \\ 0 & \text{si } h = y \end{cases}$$

$$\underset{w \in \mathbb{R}^d}{\text{minimiser}} \; \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(h(x_i; w) \neq y_i)$$

↳ Pour une même tâche d'apprentissage, on peut définir une infinité de problèmes d'optimisation basés sur des pertes et des modèles

⇒ En pratique, on évite de choisir des fonctions discontinues (comme la "0/1 loss") et on préfère des fonctions continues, pour lesquelles on sait quantifier comment la fonction varie quand ses paramètres varient

⇒ On considère dans la grande majorité des cas des pertes convexes

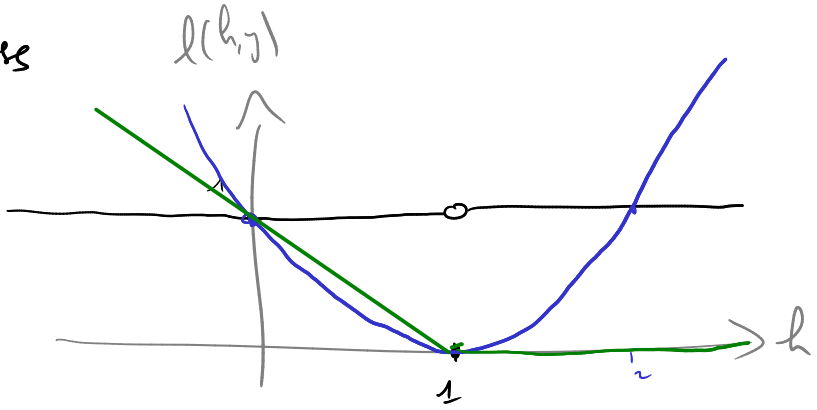
Définition : $\varphi : \mathbb{R}^d \to \mathbb{R}$ est convexe si :

$$\forall (v,w) \in (\mathbb{R}^d)^2,\ \forall \alpha \in [0,1],$$

$$\varphi(\alpha v + (1-\alpha) w) \leq \alpha \varphi(v) + (1-\alpha)\varphi(w)$$

Ex) Perte ℓ_2 convexe, Hinge loss, perte logistique

Contre-ex) 0/1 loss



$y = 1$

— 0/1 loss

— ℓ_2 loss $(h-y)^2$

— Hinge loss $\max(1-yh, 0)$

Caractérisation des fonctions convexes (tiré de "Patterns, predictions and actions", Moritz Hardt, Benjamin Recht, 2022)

"Plus de 50% des fonctions convexes en ML sont obtenues à partir des axiomes suivants"

1. Toutes les normes sont convexes ($\|v\|^2 = \sum_{j=1}^{\hat{}} [v]_j^2$ est une norme, mais n'est pas la seule)

2. Si φ est convexe et que $\lambda \geq 0$, alors $\lambda\varphi$ est convexe

3. Si φ et ψ sont convexes, alors $\varphi + \psi$ est convexe

4. Si φ et ψ sont convexes, alors $\max(\varphi, \psi)$ est convexe

5. Si φ est convexe, alors pour toute matrice A et tout vecteur b de tailles appropriées, $w \mapsto \varphi(Aw+b)$ est convexe

Exercice : En utilisant les axiomes, justifier que la fonction de l'exemple ② (SVM) est convexe

$$w \longmapsto \frac{1}{n} \sum_{i=1}^{n} \max(1 - y_i x_i^T w, 0)$$

- $w \longmapsto 0$ est convexe $\quad 0 \leq 0 + 0$
- $\mathbb{R} \to \mathbb{R}$, $t \mapsto t$ est convexe $\quad \underbrace{\alpha t + (1-\alpha) u}_{\varphi(\alpha t + (1-\alpha)u)} \leq \alpha \underbrace{t}_{\varphi(t)} + (1-\alpha) \underbrace{u}_{\varphi(u)}$

- Par l'axiome 5, $w \longmapsto 1 - y_i x_i^T w$ est convexe $\forall i = 1..n$

$$w \longmapsto \varphi\left(\underbrace{(-y_i x_i^T)}_{A} w + \underbrace{1}_{b} \right) \quad \text{avec } \varphi : t \longmapsto t$$

- Par l'axiome 4, $w \longmapsto \max(1 - y_i x_i^T w, 0)$ est convexe $\forall i = 1..n$
- Par l'axiome 3, $w \longmapsto \sum_{i=1}^{n} \max(1 - y_i x_i^T w, 0)$ est convexe
- Par l'axiome 2, $w \longmapsto \underbrace{\frac{1}{n}}_{\geq 0} \sum_{i=1}^{n} \max(1 - y_i x_i^T w, 0)$ est convexe

2) Descente de gradient

2-a) Descente de gradient dans le cas convexe

Problème : $\min_{w \in \mathbb{R}^d} f(w)$ $\quad f : \mathbb{R}^d \to \mathbb{R}$ de classe C^1

f C^1 $\Rightarrow$ En tout point $w \in \mathbb{R}^d$, il existe un vecteur

appelé le gradient de f en w (et noté $\nabla f(w) \in \mathbb{R}^d$) qui représente la dérivée de f en w et donc la variation de f au voisinage de w

$$\forall v \in \mathbb{R}^d, \quad f(v) \approx f(w) + \nabla f(w)^T (v - w)$$

lorsque $\|v - w\|$ est faible

→ Avec le gradient, on approxime la fonction f par une fonction linéaire

Théorème : Si f est C^1 et convexe, alors

$$\left[w^* \in \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} f(w) \right] \Longleftrightarrow \left[\nabla f(w^*) = 0_{\mathbb{R}^d} \right]$$

"argmin"
Ensemble des solutions / des minima du problème
minimiser $f(w)$
$w \in \mathbb{R}^d$

Condition qui est calculable / vérifiable en temps fini

w^* est une solution ssi : $f(w^*) \leq f(w) \ \forall w \in \mathbb{R}^d$] condition qui n'est pas vérifiable en temps fini

Principe de la descente de gradient (basé sur le théorème)

Etant donné un point $w \in \mathbb{R}^d$ et f C^1 convexe,

1) Soit $\nabla f(w) = 0$ et dans ce cas w est une solution

2) Soit $\|\nabla f(w)\| \neq 0$ ($\Leftrightarrow \nabla f(w) \neq 0_{\mathbb{R}^d}$) et dans ce cas pour un $\alpha > 0$ suffisamment petit

Rappel : $v = 0 \Leftrightarrow \|v\| = 0$

$$f(w - \alpha \nabla f(w)) \approx f(w) + \nabla f(w)^T (w - \alpha \nabla f(w) - w)$$

$$\Leftrightarrow \quad f(w - \alpha \nabla f(w)) \approx f(w) - \alpha \underbrace{\nabla f(w)^T \nabla f(w)}_{= \|\nabla f(w)\|^2}$$

$$\Leftrightarrow \quad f(w - \alpha \nabla f(w)) \approx f(w) - \alpha \underbrace{\|\nabla f(w)\|^2}_{>0} \quad \underbrace{}_{<0} < f(w)$$

Algorithme de descente de gradient

Initialisation : Choisir $w_0 \in \mathbb{R}^d$.

Boucle :

Pour $k = 0, 1, 2, \ldots$

- Calculer $\nabla f(w_k)$. Si $\|\nabla f(w_k)\| = 0$, terminer.
- Poser $w_{k+1} = w_k - \alpha_k \nabla f(w_k)$ avec $\alpha_k > 0$.

→ En pratique :

- On utilise un critère d'arrêt pour l'algorithme, qui est souvent une combinaison d'un critère de convergence et d'un critère de budget
- Critère de convergence : $\|\nabla f(w_k)\| = 0$, $\|\nabla f(w_k)\| \leq \varepsilon$ avec $\varepsilon > 0$

$$\|w_{k+1} - w_k\| = \alpha_k \|\nabla f(w_k)\| \leq \varepsilon \text{ avec } \varepsilon > 0$$

$$f(w_{k+1}) - f(w_k) \leq \varepsilon, \ \varepsilon > 0$$

↑ Requiert de calculer les valeurs de fonction, ce que ne fait pas l'algorithme de base

- Critère de budget : $k = \underbrace{K}_{\substack{\text{nombre} \\ \text{max d'itérations}}}$, temps CPU/GPU = temps_max,

  nombre maximum d'appels à f (dépend de l'implémentation) ou à ∇f,

⇒ La théorie de la descente de gradient permet de lier les 2 critères :

- Si on souhaite $\|\nabla f(w_k)\| \leq \epsilon$, on peut borner le nombre d'itérations nécessaires à la descente de gradient

  complexité (au pire cas)

- Si on fait tourner la descente de gradient pendant k itérations, on peut garantir une précision sur $\|\nabla f(w_k)\|$

  vitesse de convergence

- S'applique à la majorité des paires (critère de convergence / de budget)

→ Choix de α_k (α_k : taille de pas, toujours >0)

- Taille de pas constante : $\alpha_k = \alpha > 0 \quad \forall k \in \mathbb{N}$

  ⊕ Il existe des valeurs constantes pour lesquelles la convergence est garantie sur les formulations classiques (régression linéaire / régression logistique)

  ⊖ Si $\alpha > 0$ trop grande, la méthode diverge ($\|w_k\| \to \infty$)

  ⊖ Si $\alpha > 0$ trop petite, numériquement la méthode stagne

- Taille de pas décroissante (fixée a priori) : $\alpha_k \xrightarrow[k \to \infty]{} 0$

  Ex) $\alpha_k = \frac{\alpha_0}{k+1}$ avec $\alpha_0 > 0$, $\alpha_k = \frac{\alpha_0}{(k+1)^t}$ $t > 0$

  ⊕ Garantit que α_k est une bonne taille de pas pour k suffisamment grand (en particulier, $f(w_{k+1}) < f(w_k)$ pour k suffisamment grand)

⊖ Si α_k décroît trop vite, numériquement la méthode va stagner

- Hybride : ("scheduling")
  - Démarrer avec un pas constant $\alpha > 0$
  - Soit au bout de K itérations, soit lorsqu'on stagne (ex: $f(w_{k+1}) \simeq f(w_k)$)

    $\alpha \leftarrow \alpha/10$ (plus généralement $\alpha \leftarrow \theta\alpha$, $\theta \in (0,1)$)

Partie 1 des TPs (et du projet) : Tester des tailles de pas