Outils d'optimisation pour les sciences des données et de la décision

15 novembre 2024

Programme des séances restantes

- Aujourd'hui : Cours (gradient stochastique)
- 22/11 : TD (gradient stochastique)
  ⚠ Exercices + Notebook Python
- 29/11 : Séance-bilan

Projet : . Sujet bientôt disponible !
. Date rendu (flexible) : 31 janvier 2025

MÉTHODES DE GRADIENT STOCHASTIQUE

① Motivation

↳ Jusqu'à maintenant, on a considéré un problème général : $\min_{w \in \mathbb{R}^d} f(w)$ avec $f : \mathbb{R}^d \to \mathbb{R}$ de classe C^1

$\Rightarrow$ En exploitant la structure de f, on peut développer des algorithmes d'optimisation plus adaptés au problème que la descente de gradient

Exemples de problèmes structurés en sciences des données

1) Régression linéaire

$\{(x_i, y_i)\}_{i=1..m}$ jeu de données avec m | éléments / points / échantillons

$\forall i=1..m$,

$x_i \in \mathbb{R}^d$: vecteur d'attributs (features)

$y_i \in \mathbb{R}$: label

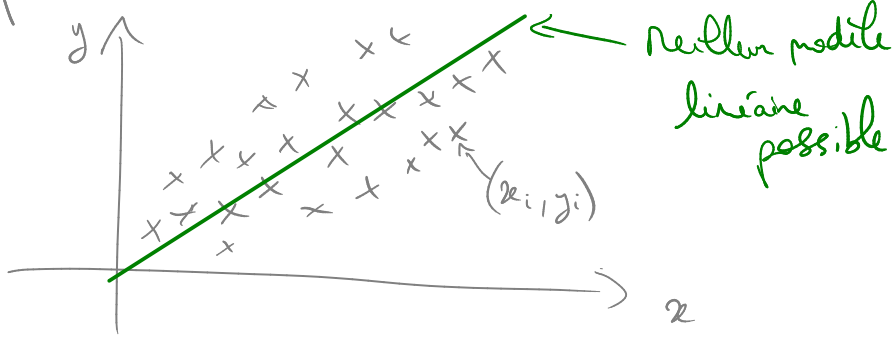
But : Trouver un modèle linéaire, c'est-à-dire une fonction $x \in \mathbb{R}^d \longmapsto x^T w = \sum_{j=1}^{d} [x]_j [w]_j$,

qui colle aux données du mieux possible, c'est-à-dire que $x_i^T w - y_i \approx 0 \quad \forall i=1..m$

↑ "à peu près" = le plus proche possible de 0

⚠ Pas toujours possible de coller exactement aux données

$d = 1$



Meilleur modèle linéaire possible

→ Pas de modèle linéaire parfait (qui passerait par tous les points)

→ On peut quand même trouver le meilleur modèle possible étant données les points

Formalisation via un problème d'optimisation

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \left(x_i^T w - y_i \right)^2$$

- Somme sur tous les points
- Valeur du modèle linéaire en x_i
- Erreur du modèle : écart entre $x_i^T w$ et y_i (au carré $\times \frac{1}{2}$ par convention)

- Problème convexe voire fortement convexe
- Fonction objectif de classe C^1

⇒ On peut utiliser la descente de gradient pour résoudre le problème

- Si $n \gg 1$, peut être coûteux de calculer la fonction et/ou son gradient
- Si de plus les données proviennent d'une même distribution, il n'est pas forcément nécessaire de considérer tous les points du jeu de données pour déterminer si la valeur de la fonction peut être améliorée

→ Propriétés structurelles du problème

2) Réseaux de neurones

→ Modèle simple de réseau "feed-forward" pour la régression

→ Idée : Aller au-delà des modèles linéaires

Données : $\{(x_i, y_i)\}_{i=1..m}$ $\quad x_i \in \mathbb{R}^{d_0}$ $d_0 \geq 1$; $\quad y_i \in \mathbb{R}^{d_L}$ $d_L \geq 1$

Réseau de neurones à L couches

Suite de fonctions qui produisent une suite de vecteurs

$x^{(0)} \in \mathbb{R}^{d_0} \longmapsto x^{(1)} \longmapsto x^{(2)} \longmapsto \dots \longmapsto x^{(L)} \in \mathbb{R}^{d_L}$

$x^{(0)}$: entrée du réseau ; $x^{(1)}$: 1re couche ; $x^{(L)}$: dernière (L-ième) couche, sortie

$$\forall l = 1..L, \quad x^{(l)} = \sigma\left(W^{(l)} x^{(l-1)} + b^{(l)}\right)$$

$W^{(l)} x^{(l-1)} + b^{(l)}$: transformation linéaire de $x^{(l-1)}$

où $W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ matrice de poids (weights)

$b^{(l)} \in \mathbb{R}^{d_l}$ vecteur de biais

Paramètres de la couche l → $W^{(l)}, b^{(l)}$

σ fonction d'activation non linéaire qui est appliquée à chaque coordonnée de son vecteur argument

$$\forall v \in \mathbb{R}^{d_l}, \quad \sigma(v) = \begin{bmatrix} \sigma([v]_1) \\ \vdots \\ \sigma([v]_{d_l}) \end{bmatrix} \in \mathbb{R}^{d_l}$$

(Ex) $\sigma(t) = \max(t, 0)$: Activation ReLU (la plus fréquente en pratique)

$\sigma(t) = \tanh(t)$ (populaire en traitement naturel des langues)

But : • Prédire y_i via $x_i^{(L)}$, où $x_i^{(L)}$ est la sortie du réseau quand x_i est donné en entrée

$\Rightarrow$ On souhaite que $\|x_i^{(L)} - y_i\|^2$ soit la plus faible possible ($y_i \in \mathbb{R}^{d_L}$ $\Rightarrow$ on utilise une norme pour mesurer un écart)

• Variables du problème : poids $\{W^{(\ell)}\}_{\ell=1..L}$ et $\{b^{(\ell)}\}_{\ell=1..L}$

$\Rightarrow$ On concatène toutes ces variables sous la forme d'un vecteur $w \in \mathbb{R}^d$

$$w = \begin{bmatrix} W_{11}^{(1)} \\ \vdots \\ W_{d_1 d_0}^{(1)} \\ b_1^{(1)} \\ \vdots \end{bmatrix} \qquad d = \underbrace{d_1 d_0}_{W^{(1)}} + \underbrace{d_1}_{b^{(1)}} + \underbrace{d_2 d_1}_{W^{(2)}} + \underbrace{d_2}_{b^{(2)}} + \dots + \underbrace{d_L d_{L-1}}_{W^{(L)}} + \underbrace{d_L}_{b^{(L)}}$$

$\Rightarrow$ On note $x_i^{(L)}(w)$ la sortie du réseau de mêmes poids avec x_i en entrée et w en paramètres

Problème d'optimisation

prédiction à partir de x_i

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

- Si $L > 1$, problème non convexe
- Si $\sigma = \max(\cdot, 0)$, fonction objectif non lisse
- Si $L \gg 1$ et/ou $d \gg 1$, très coûteux de calculer la fonction objectif ou son gradient
- Si $n \gg 1$, coût supplémentaire qui n'est pas forcément nécessaire si les données proviennent d'une même distribution

↳ Dans les deux exemples, la fonction objectif du problème d'optimisation possède une structure de somme finie ($\frac{1}{n}\sum_{i=1}^{n} \dots$), où chacun des termes de la somme dépend d'un point distinct du jeu de données.

Q) Si l'on dispose d'un algorithme (de type descente de gradient) pour résoudre le problème, comment l'adapter pour prendre en compte la structure en somme finie et la distribution sous-jacente des données ?

② Algorithme du gradient stochastique

→ Développé dans les années 1940-1950 par des statisticiens (Robbins & Monro)

→ Regain d'intérêt à la fin des années 2000 avec l'utilisation de méthodes de gradient stochastique pour entraîner les réseaux de neurones (+ lien avec l'algorithme du perceptron)

⇒ Success story en 2012 : Performances de réseaux de neurones en classification d'images

Problème

(P) minimiser $_{w \in \mathbb{R}^d}$ $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$

avec $f_i : \mathbb{R}^d \to \mathbb{R}$, de classe C^1 et qui dépend du $i$ème point d'un jeu de données à n éléments

$f_i \in C^1 \ \forall i \quad \Rightarrow$ f est C^1, et on peut donc appliquer la descente de gradient au problème (P)

Itération de la descente de gradient

$\nabla f(w_k)$: gradient de f en w_k (w_k n'est pas minimum $\Leftrightarrow \|\nabla f(w_k)\| \neq 0$)

$$w_{k+1} = w_k - \alpha_k \nabla f(w_k)$$

avec $\alpha_k > 0$

Pour (P),

$$\nabla f(w_k) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_k)$$

$\Rightarrow$ Un calcul de ∇f nécessite d'accéder à tous les points du jeu de données (coûteux !)

Itération du gradient stochastique

$\nabla f_i(w)$: gradient de f_i évalué en w

$$w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k)$$

avec $\alpha_k > 0$

et i_k tiré aléatoirement entre 1 et n

$i_k \in \{1, \dots, n\}$

(+) Coût : 1 seul accès à un point du jeu de données par itération

(−) Approximation du gradient de f par celui de f_{i_k} $\Rightarrow$ risque d'utiliser une direction qui n'améliore pas la fonction

(?) Aléatoire : comment tirer les i_k ? Est-ce que l'aléatoire est bénéfique ?

(2) Théorie du gradient stochastique

(P) minimiser $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$
$w \in \mathbb{R}^d$

Hypothèses :

- $f(w) \geq \underline{f}$, $\underline{f} \in \mathbb{R}$
- $f_i \in C^1 \quad \forall i \in \{1, \dots, n\}$
- $f \in C^{1,1}_L$, càd que

$$\forall (v, w) \in (\mathbb{R}^d)^2, \quad \|\nabla f(v) - \nabla f(w)\| \leq L \|v - w\|$$

avec $L > 0$

$$\Rightarrow \forall (v, w) \in (\mathbb{R}^d)^2,$$

$$f(v) \leq f(w) + \nabla f(w)^T (v - w) + \frac{L}{2}\|v - w\|^2$$

Inégalité fondamentale pour la théorie de la descente de gradient

Pour la descente de gradient

$$w_{k+1} = w_k - \alpha_k \nabla f(w_k) \quad , \alpha_k > 0$$

et donc par l'inégalité fondamentale

$$f(w_{k+1}) \leq f(w_k) - \alpha_k \|\nabla f(w_k)\|^2 + \frac{L}{2}\alpha_k^2 \|\nabla f(w_k)\|^2$$

$$< f(w_k) \quad \text{si } \alpha_k < \frac{2}{L}$$

- Permet de définir des valeurs pour α_k qui garantissent une décroissance de f à chaque itération
- Permet aussi de montrer des garanties de complexité pour la descente de gradient

Pour $\{\alpha_k\}$ bien choisie (ex : $\alpha_k = \frac{1}{L}\ \forall k$), la descente de gradient calcule w_k tel que $\|\nabla f(w_k)\| \leq \varepsilon$ en au plus

$O(\varepsilon^{-2})$ itérations

$\| $

$C\varepsilon^{-2}$, où $C>0$ ne dépend pas de ε

Pour le gradient stochastique

$$w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k) \qquad \alpha_k > 0$$

i_k ↑ indice aléatoire entre 1 et m

Inégalité fondamentale appliquée à (w_{k+1}, w_k)

$$f(w_{k+1}) \leq f(w_k) - \alpha_k \nabla f(w_k)^T \nabla f_{i_k}(w_k) + \frac{L}{2}\alpha_k^2 \|\nabla f_{i_k}(w_k)\|^2$$

vecteurs choisis aléatoirement

→ L'inégalité ne permet plus de garantir $f(w_{k+1}) < f(w_k)$ à cause de l'aléatoire sur i_k.

→ On peut cependant garantir une décroissance [en moyenne / en espérance] sous les bonnes hypothèses

Hypothèses sur les i_k

A l'itération k, on suppose que

1) i_k est tiré indépendamment de $i_0, \dots, i_{k-1}$

2) $\mathbb{E}_{i_k}[\nabla f_{i_k}(w_k)] = \nabla f(w_k)$

3) $\mathbb{E}_{i_k}\left[\|\nabla f_{i_k}(w_k)\|^2\right] \leq \sigma^2$ avec $\sigma^2 > 0$

Interprétation

1) A chaque itération, on a la même probabilité sur chaque valeur entre $\{1, \dots, n\}$

2) En moyenne, le gradient stochastique $\nabla f_{i_k}(w_k)$ équivaut à $\nabla f(w_k)$

$\forall$ variable A qui dépend de i_k, $\mathbb{E}_{i_k}[A(i_k)] = \sum_{i=1}^{n} \mathbb{P}(i_k = i)\, A(i)$

$$\mathbb{E}_{i_k}\left[\nabla f_{i_k}(w_k)\right] = \sum_{i=1}^{n} \mathbb{P}(i_k = i)\, \nabla f_i(w_k)$$

3) La norme des gradients stochastiques ne varie pas trop

Exemple classique (typique de l'implémentation)

Tirer i_k selon une loi uniforme

$$\mathbb{P}(i_k = i) = \frac{1}{n}$$

$\Rightarrow$ 1), 2) (+ 3) en général)

$$\mathbb{E}_{i_k}\left[\nabla f_{i_k}(w_k)\right] = \sum_{i=1}^{n} \mathbb{P}(i_k = i)\, \nabla f_i(w_k)$$

$$= \sum_{i=1}^{n} \frac{1}{n} \nabla f_i(w_k) = \underbrace{\frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_k)}_{\text{définition de } \nabla f(w_k)} = \nabla f(w_k)$$

↳ Sous ces hypothèses sur i_k, on montre qu'à chaque itération

$$\mathbb{E}_{i_k}\left[f(w_{k+1})\right] \leq f(w_k) - \alpha_k \|\nabla f(w_k)\|^2 + \frac{L\sigma^2\alpha_k^2}{2}$$

↑ Ne dépend plus de i_k

→ Permet de garantir $\mathbb{E}_{i_k}\left[f(w_{k+1})\right] < f(w_k)$

"Décroissance en moyenne"

pour des bons choix de α_k

(ex) $\{\alpha_k\}_k \to 0$ (par décroissant)

$0 < \alpha_k \leq \frac{1}{L}\frac{\|\nabla f(x_k)\|^2}{\sigma^2}$ → par définition par l'itération k

⇒ Définir une stratégie de choix de pas convenable pour la théorie

⇒ En pratique, on choisit un pas constant (ex : Dans PyTorch, $\alpha_k = 0.01$)

NB : Les stratégies de recherche linéaire ne sont pas utilisées pour les méthodes de gradient stochastique parce qu'elles nécessitent d'évaluer f, donc d'accéder à tous les points du jeu de données.

Théorème Sous les hypothèses précédentes, supposons que l'on choisit $\alpha_k = \frac{\varepsilon^2}{L\sigma^2}$ $\forall k$ avec $\varepsilon > 0$.

Alors, la méthode de gradient stochastique

calcule en moyenne un point w_k tel que

$\|\nabla f(w_k)\| \leq \varepsilon$ en au plus $O(\varepsilon^{-4})$ itérations.

Formellement, après $O(\varepsilon^{-4})$ itérations, $\mathbb{E}\left[\min_{0\leq l\leq k} \|\nabla f(x_l)\|\right] \leq \varepsilon$

Comparaison avec la descente de gradient

Descente de gradient:	$\alpha_k = \frac{1}{L}$ $\forall k$	$\min_{0\leq l\leq k} \|\nabla f(x_l)\| \leq \varepsilon$	$O(\varepsilon^{-2})$ itérations
Gradient stochastique	$\alpha_k = \frac{\varepsilon^2}{L\sigma^2}$ $\forall k$	$\mathbb{E}\left[\min_{0\leq l\leq k} \|\nabla f(x_l)\|\right] \leq \varepsilon$	$O(\varepsilon^{-4})$ itérations
	→ DG: pas indépendant de ε → GS: pas fonction de ε, qui doit être redéfini si on change la valeur de ε	→ DG: convergence déterministe → GS: convergence en moyenne	ε^{-2} augmente moins vite que ε^{-4} quand ε diminue $\varepsilon \to \varepsilon/10$ $\varepsilon^{-2} = 100\,\varepsilon^{-2}$ $\varepsilon^{-4} = 10000\,\varepsilon^{-4}$

⇒ Si l'on regarde uniquement ces résultats, la descente de gradient semble bien meilleure que le gradient stochastique

⇒ Mais ces résultats ne tiennent pas compte du coût de calcul des gradients, qui était la raison pour laquelle on a considéré le gradient stochastique

1 itération de descente de gradient = 1 calcul de $\nabla f(w_k)$
= n accès à des points du jeu de données

1 itération de gradient stochastique = 1 calcul de $\nabla f_i(x_t)$

= 1 accès à un point du jeu de données

$\Rightarrow$ En terme d'accès aux données, 1 itération de gradient stochastique est n fois moins coûteuse qu'1 itération de descente de gradient

$\Rightarrow$ Pour pouvoir comparer ces deux algorithmes de manière équitable, il faut utiliser une autre quantité que le nombre d'itérations

Définition : Une époque (epoch en anglais) est définie comme le coût d'accès à n points d'un jeu de données à n éléments

NB : Si on accède n fois au même point, on compte n accès

1 itération de descente de gradient a un coût de 1 époque

1 itération de gradient stochastique a un coût de $\frac{1}{n}$ époque

Conséquences sur la complexité

Descente de gradient	$O(\varepsilon^{-2})$ itérations	$O(\varepsilon^{-2})$ époques
Gradient stochastique	$O(\varepsilon^{-4})$ itérations	$O(\frac{1}{n}\varepsilon^{-4})$ époques

Lorsque $n \gg 1$, on peut avoir $n \gg \varepsilon^{-2}$ et alors la complexité du gradient stochastique est meilleure

$\Rightarrow$ Explique en partie pourquoi une méthode de gradient stochastique converge plus vite qu'une descente de gradient sur les premières époques.

(3) Variantes de la méthode du gradient stochastique

- Utiliser plus qu'un accès au jeu de données par itération

DG : n accès / itération GS : 1 accès / itération

$1 < n_b < n$ accès par itération ?

Méthode de gradient stochastique par fournées (batch stochastic gradient)

$$w_{k+1} = w_k - \alpha_k \left(\frac{1}{|S_k|} \sum_{i \in S_k} \nabla f_i(w_k) \right)$$

$$w_{k+1} = w_k - \frac{\alpha_k}{|S_k|} \sum_{i \in S_k} \nabla f_i(w_k)$$

$|S_k|$: taille de l'ensemble S_k

avec $\alpha_k > 0$ et S_k un ensemble d'indices tirés aléatoirement entre 1 et n, avec ou sans remise

$\rightarrow$ Généralise à la fois le gradient stochastique ($S_k = \{i_k\}$, $|S_k| = 1$) et la descente de gradient ($S_k = \{1, \dots, n\}$, $|S_k| = n$)

tirage sans remise

$\rightarrow$ Permet notamment de prendre en compte les possibilités de calcul parallèle des $\nabla f_i(w_k)$

$\Rightarrow$ Si on dispose de r processeurs qui peuvent calculer des gradients en parallèle, on posera $|S_k| = r$

(en général $r \ll n$)

$\longrightarrow$ Deux catégories de méthodes par fournées $(|S_k| = m_b \ \forall k)$ $m_b \in \{1, \dots, n\}$

- Grande fournée : $m_b \gg 1$, $m_b < n$
  - $\oplus$ Peu de variabilité entre les exécutions
  - $\ominus$ Coût comparable à celui de la descente de gradient
- Petite/Mini-fournée (Mini-batch) : $m_b > 1$, $m_b \ll n$
  - $\oplus$ Coût bien plus faible que celui de la descente de gradient
  - $\oplus$ Réduit la variance des gradients stochastiques (mathématiquement, $\mathbb{E}\left[\|\nabla f_{i_k}(w_k)\|^2\right] \leq \sigma^2$ $\longrightarrow$ $\mathbb{E}\left[\left\|\frac{1}{|S_k|}\sum_{i \in S_k} \nabla f_i(w_k)\right\|^2\right] \leq \frac{\sigma^2}{m_b}$)

    Réduit aussi l'écart entre les exécutions de la méthode
  - $\ominus$ Plus cher (à chaque itération) que le gradient stochastique, et il peut arriver que le gradient stochastique classique soit quand même meilleur en pratique
  - $\ominus$ Choisir la bonne taille de fournée est un problème en soi, qui dépend beaucoup des données considérées

Autres variantes

- $\longrightarrow$ Pour f non lisse, on peut utiliser des sous-gradients à la place des gradients, ou des variantes du gradient proximal
- $\longrightarrow$ On peut considérer des problèmes régularisés (gradient proximal stochastique)

→ On peut faire des méthodes d'ordre 2 (cf projet)

→ On peut incorporer du momentum dans l'itération du gradient stochastique

$$w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k) + \beta_{k+1}(w_k - w_{k-1})$$

↑ Terme de momentum

⊕ Marche bien en pratique, base de la méthode d'entraînement des réseaux de neurones modernes (Adam) 2015

⊖ Pas de théorie complète de convergence