Optimisation à grande échelle

Joon Kwon

Master 2 — MathSV

jeudi 6 octobre 2022

Motivation

Soient deux entiers potentiellement très grands :

- $d \geqslant 1$ représentant la dimension
- $n \geqslant 1$ représentant le nombre de données

On s'intéresse typiquement à la minimisation sur $\mathbb{R}^d$ de fonctions objectifs de la forme :

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x),$$

où les $f_i : \mathbb{R}^d \to \mathbb{R}$ sont différentiables.

Le calcul d'un gradient $\nabla f(x)$ correspond en fait au calcul de n gradients $\nabla f_i(x)$.

Exemples :

- Minimisation du risque empirique en apprentissage statistique
- Estimation par maximum de vraisemblance

Méthode du gradient stochastique

Données massives et mini-batchs

Réduction de variance

Algorithmes à mise à l'échelle diagonale

Méthode du gradient stochastique

Définition

- $f : \mathbb{R}^d \to \mathbb{R}$ différentiable
- $x^{(1)} \in \mathbb{R}^d$ quelconque
- $(\gamma^{(t)})_{t \geqslant 1}$ une suite de pas strictement positifs

$$x^{(t+1)} = x^{(t)} - \gamma^{(t)} \hat{g}^{(t)}, \qquad t \geqslant 1,$$

où $\hat{g}^{(t)}$ est un estimateur non biaisé de $\nabla f(x^{(t)})$, autrement dit une variable aléatoire telle que :

$$\mathbb{E}\left[\hat{g}^{(t)} \mid x^{(t)}\right] = \nabla f(x^{(t)}).$$

- Les itérées sont aléatoires.
- On perd en précision mais $\hat{g}^{(t)}$ peut être beaucoup moins coûteux à obtenir que $\nabla f(x^{(t)})$.

Hypothèse : second moment borné

On suppose que les estimateurs du gradient $\hat{g}^{(t)}$ ont un second moment borné

Soit $\sigma > 0$ tel que :

$$\mathbb{E}\left[\left\|\hat{g}^{(t)}\right\|^2\right] \leqslant \sigma^2, \qquad t \geqslant 1.$$

Garantie dans le cas régulière et non-convexe

Théorème

Soit $T \geqslant 1$ et $L > 0$. On suppose que f est L-régulière. La méthode du gradient stochastique avec garantit avec $\gamma^{(t)} = \sqrt{2/(L\sigma^2 t)}$ (pour $t \geqslant 1$) :

$$\mathbb{E}\left[\frac{\sum_{t=1}^{T} \gamma^{(t)} \left\|\nabla f(x^{(t)})\right\|^2}{\sum_{t=1}^{T} \gamma^{(t)}}\right] \leqslant \sigma\sqrt{\frac{L}{2}} \times \frac{f(x^{(1)}) - f(x^*) + 1 + \log T}{\sqrt{T+1}}.$$

Cela garantit un point stationnaire approché.

Garantie dans le cas convexe

Théorème

Soit $T \geqslant 1$. On suppose que f est convexe. La méthode du gradient stochastique avec garantit avec $\gamma^{(t)} = \sqrt{1/(\sigma^2 t)}$ (pour $t \geqslant 1$) :

$$\mathbb{E}\left[f\left(\frac{\sum_{t=1}^{T}\gamma^{(t)}x^{(t)}}{\sum_{t=1}^{T}\gamma^{(t)}}\right)\right] - f(x^*) \leqslant \frac{\sigma}{2} \times \frac{\left\|x^{(1)} - x^*\right\|^2 + 1 + \log T}{\sqrt{T+1}}.$$

Cela garantit un minimiseur approché.

Garantie dans le cas fortement convexe et régulière

Théorème

Soit $T \geqslant 1$, $K, L > 0$. On suppose que f est K-fortement convexe et L-regulière. La méthode du gradient stochastique avec garantit avec $\gamma^{(t)} = \sqrt{1/(Kt)}$ (pour $t \geqslant 1$) :

$$\mathbb{E}\left[f\left(x^{(T)}\right)\right] - f(x^*) \leqslant \frac{2L\sigma^2}{K^2 T}.$$

Cela garantit un minimiseur approché.

Données massives et mini-batchs

Mini-batchs

On considère des fonctions objectifs de la forme :

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x),$$

- **Idée :** construire des estimateurs du gradient en ne tirant aléatoirement qu'un petit nombre de données à chaque itération.
- Soit $k \geqslant 1$ un entier.
- À chaque étape $t \geqslant 1$, on tire uniformément un sous-ensemble :

$$B^{(t)} \subset \{1, \ldots, n\} \qquad \text{tel que} \qquad \left| B^{(t)} \right| = k.$$

- $B^{(t)}$ est appelé un mini-batch.
- On construit :

$$\hat{g}^{(t)} = \frac{1}{k} \sum_{i \in B^{(t)}} \nabla f_i(x^{(t)}).$$

- On a bien :

$$\mathbb{E}\left[\hat{g}^{(t)} \mid x^{(t)} \right] = \nabla f(x^{(t)}).$$

Remarques sur les mini-batchs

- Dans la pratique, on renumérote une fois aléatoirement les i, puis on choisit les mini-batchs :

$$B^{(1)} = \{1, \dots, k\}, \qquad B^{(2)} = \{k+1, \dots, 2k\}, \dots$$

- Quand $n \gg 1$, on prend $k \ll n$. Alors, calculer un estimateur du gradient $\hat{g}^{(t)}$ sur un mini-batch est beaucoup moins coûteux que calculer le gradient exact $\nabla f(x^{(t)})$.
- En revanche, on perd en vitesse de convergence :

	Gradient exact	Gradient stochastique
Régulière	$\frac{1}{T}$	$\frac{1}{\sqrt{T}}$
Fortement convexe et régulière	$\left(1 - \frac{K}{L}\right)^T$	$\frac{1}{T}$

- Reste très avantageux pour les premières itérations, pour lesquelles la précision des gradients est moins importante.
- **Question :** peut-on combiner
  - l'avantage computationel des mini-batchs, et
  - des vitesses de convergence rapides ?
- **Réponse :** oui, grace aux techniques de réduction de variance.

Réduction de variance

Principe

Idée de base : un estimateur peut être amélioré si on connaît une deuxième variable aléatoire positivement corrélée à la première et dont l'espérance est connue.

- Soit X, Y deux variables aléatoires réelles
- On suppose que $\mathbb{E}[Y]$ est connu.
- But : estimer $\mathbb{E}[X]$.

Alors,

- X est un estimateur de $\mathbb{E}[X]$.
- $Z = X - (Y - \mathbb{E}[Y])$ est aussi un estimateur de $\mathbb{E}[X]$. De plus,

$$\operatorname{Var} Z = \operatorname{Var} X + \operatorname{Var} Y - 2 \operatorname{Cov}(X, Y).$$

- Si $\operatorname{Cov}(X, Y) > \frac{1}{2} \operatorname{Var} Y$, on a $\operatorname{Var} Z < \operatorname{Var} X$ et alors Z est un "meilleur" estimateur que X.

SAGA (Defazio, Bach & Lacoste-Julien, 2014)

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x).$$

- $x^{(1)} \in \mathbb{R}^d$ quelconque. Calculer et stocker le gradient de chaque fonction f_i en $x^{(1)}$:

$$g^{[i]} \leftarrow \nabla f_i(x^{(1)}), \qquad 1 \leqslant i \leqslant n.$$

- Pour $t \geqslant 1$,
  - On tire $i^{(t)} \sim \mathcal{U}(\{1, \dots, n\})$
  - On calcule l'estimateur $\hat{g}^{(t)} = \nabla f_{i^{(t)}}(x^{(t)}) - g^{[i^{(t)}]} + \frac{1}{n}\sum_{i=1}^{n} g^{[i]}$
  - On itère $x^{(t+1)} = x^{(t)} - \gamma^{(t)}\hat{g}^{(t)}$
  - On met à jour $g^{[i^{(t)}]} \leftarrow \nabla f_{i^{(t)}}(x^{(t)})$

On calcule un seul gradient par itération.
Il est nécessaire de stocker n gradients.

Dans la pratique, on utilise des minibatchs qui ne se chevauchent pas.

$$B^{(1)} = \{1, \dots, k\}, \qquad B^{(2)} = \{k+1, \dots, 2k\}, \dots$$

On ne doit alors stocker qu'un nombre de gradients correspondant au nombre de mini-batchs différents utilisés ($\simeq n/k$).

SVRG (Johnson & Zhang, 2013)

- $\tau \geqslant 1$ un paramètre entier
- $x^{(1)} \in \mathbb{R}^d$ quelconque

Pour $t \geqslant 1$,

- On calcule $\mu^{(t)} = \dfrac{1}{n} \displaystyle\sum_{i=1}^{n} \nabla f_i(x^{(t)})$
- On pose $x^{(t,1)} = x^{(t)}$.
- Pour $1 \leqslant s \leqslant \tau$,
  - On tire $i^{(t,s)} \sim \mathcal{U}(\{1, \dots, n\})$.
  - On calcule l'estimateur $\hat{g}^{(t,s)} = \nabla f_{i^{(t,s)}}(x^{(t,s)}) - \nabla f_{i^{(t,s)}}(x^{(t)}) + \mu^{(t)}$
  - On itère $x^{(t,s+1)} = x^{(t,s)} - \gamma^{(t,s)} \hat{g}^{(t,s)}$
- On pose $x^{(t+1)} = x^{(t,\tau)}$

On calcule n gradients tous les τ itérations, et 2 gradients à chaque itération.
Il est nécessaire de stocker 1 gradient.

Vitesses de convergence

SAGA et SVRG garantissent

	Vitesse de convergence
Régulière	$\frac{1}{T}$
Fortement convexe et régulière	α^T

pour un certain α dépendant des propriétés des fonctions f_i (et moins avantageux que $(1 - K/L)$)

Algorithmes à mise à l'échelle diagonale

Rappel : Méthodes de Newton et quasi-Newton

- Descente de gradient :

$$x^{(t+1)} = x^{(t)} - \gamma^{(t)} \nabla f(x^{(t)})$$
$$= \arg\min_{x \in \mathbb{R}^d} \left\{ f(x^{(t)}) + \nabla f(x^{(t)})^\top (x - x^{(t)}) + \frac{1}{2\gamma^{(t)}} \left\| x - x^{(t)} \right\|^2 \right\}$$

- Méthode de Newton :

$$x^{(t+1)} = x^{(t)} - \left(\nabla^2 f(x^{(t)}) \right)^{-1} \nabla f(x^{(t)})$$
$$= \arg\min_{x \in \mathbb{R}^d} \left\{ f(x^{(t)}) + \nabla f(x^{(t)})^\top (x - x^{(t)}) + \frac{1}{2} (x - x^{(t)})^\top \nabla^2 f(x^{(t)}) (x - x^{(t)}) \right\}$$

- Méthode quasi-Newton :

$$x^{(t+1)} = x^{(t)} - \left(B^{(t)} \right)^{-1} \nabla f(x^{(t)})$$
$$= \arg\min_{x \in \mathbb{R}^d} \left\{ f(x^{(t)}) + \nabla f(x^{(t)})^\top (x - x^{(t)}) + \frac{1}{2} (x - x^{(t)})^\top B^{(t)} (x - x^{(t)}) \right\}$$

  - $B^{(t)}$ est une approximation de $\nabla^2 f(x^{(t)})$ construite à partir des seuls gradients,
  - $B^{(t)}$ est une matrice de taille $d \times d$, dont l'utilisation est coûteuse quand $d \gg 1$ (à l'exception de L-BFGS).

AdaGrad et RMSProp

Idée : se restreindre pour $B^{(t)}$ aux matrices diagonales pour avoir un coût computationel par étape de $O(d)$.

Cela revient à appliquer un pas (step-size) différent à chaque composante. Si on écrit $B^{(t)} = \mathrm{diag}(b_1^{(t)}, \ldots, b_d^{(t)})$,

$$x^{(t+1)} = x^{(t)} - \left(B^{(t)}\right)^{-1} \nabla f(x^{(t)}) = x^{(t)} - \begin{pmatrix} (b_1^{(t)})^{-1} \frac{\partial f}{\partial x_1}(x^{(t)}) \\ \vdots \\ (b_d^{(t)})^{-1} \frac{\partial f}{\partial x_1}(x^{(t)}) \end{pmatrix}$$

- AdaGrad correspond à :

$$b_i^{(t)} = \frac{1}{\gamma} \sqrt{\sum_{s=1}^{t} \left(\frac{\partial f}{\partial x_i}(x^{(s)})\right)^2}, \qquad 1 \leqslant i \leqslant d.$$

- RMSProp correspond à :

$$b_i^{(t)} = \frac{1}{\gamma} \sqrt{\sum_{s=1}^{t} \beta^{t-s} \left(\frac{\partial f}{\partial x_i}(x^{(s)})\right)^2}, \qquad 1 \leqslant i \leqslant d.$$

où $\beta = 0.99$ en pratique.

Adam

Adam est une sophistication de RMSProp :

- les matrices $B^{(t)}$ sont les mêmes
- le gradient courant $\nabla f(x^{(t)})$ est remplacé par une somme des gradients précédemment observés, pondérés par récence.

$$b_i^{(t)} = \frac{1}{\gamma}\sqrt{\sum_{s=1}^{t} \beta_2^{t-s} \left(\frac{\partial f}{\partial x_i}(x^{(s)})\right)^2}, \qquad 1 \leqslant i \leqslant d$$

$$x^{(t+1)} = x^{(t)} - (B^{(t)})^{-1} \left(\sum_{s=1}^{t} \beta_1^{t-s} \nabla f(x^{(s)})\right)$$

où $\beta_1 = 0.9$ et $\beta_2 = 0.999$ en pratique.

Remarques

- On peut remplacer dans ces algorithmes $\nabla f(x^{(t)})$ par un estimateur $\hat{g}^{(t)}$.
- RMSProp et Adam font partie des algorithes les plus performants pour, entre autres, l'entraînement des réseaux de neurones.