OPTIMIZATION FOR MACHINE LEARNING

October 5, 2023

Today: Lecture on stochastic gradient
Tutorial (acceleration + stochastic gradient)

Next sessions: November 6: Tutorial + last lecture
November 9: Exam preparation (Tutorial 3h)

- Exam: December 15, 2pm - 4pm

MORE ON STOCHASTIC GRADIENT

Setup: . minimize $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$ $w \in \mathbb{R}^d$

$\forall i = 1..n$, $f_i : \mathbb{R}^d \to \mathbb{R}$ that depends on the i^{th} element in a dataset of n elements

- Every f_i is C^1, and evaluating ∇f_i (or f_i) requires to access the corresponding data point
- $n \gg 1$ so that evaluating $\nabla f = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i$ is expensive

GD: $w_{k+1} = w_k - \alpha_k \nabla f(w_k)$ ($\alpha_k > 0$) : expensive to use in that setting

SG (Stochastic Gradient) $w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k)$ ($\alpha_k > 0$) where $i_k \in \{1, \dots, n\}$ drawn at random

(1) More on convergence rates

↳ SG has worse convergence rates than GD

Ex) If f_i is convex $\forall i$, then

- After K iterations of GD, $f(w_K) - \min_{w \in \mathbb{R}^d} f(w) \leq O\left(\frac{1}{K}\right)$
- After K iterations of SG, $\mathbb{E}\left[f(w_K) - \min_{w \in \mathbb{R}^d} f(w)\right] \leq O\left(\frac{1}{\sqrt{K}}\right)$

⚠ The number of iterations is not a good metric for comparing GD and SG

⇒ We care about the number of accesses to data points

⇒ 1 iteration of GD = 1 ∇f = n accesses to data points

1 iteration of SG = ∇f_{i_k} = 1 access to 1 data point

Def: An **epoch** is a cost unit corresponding to n accesses to data points in a dataset of size n.

↳ 1 iteration of GD costs 1 epoch

1 iteration of SG costs $\frac{1}{n}$ epoch

$\Leftrightarrow$ 1 epoch is the cost of n iterations of SG

↳ If we provide convergence rates in terms of epochs and not in terms of iterations, then SG will have better guarantees for $n \gg 1$

Ex) If f_i is convex $\forall i$, then

- After N_E epochs, GD will have done N_E iterations and so

$$f(w_{N_E}) - \min_{w \in \mathbb{R}^d} f(w) \leq O\left(\frac{1}{N_E}\right) \quad (K = N_E \text{ above})$$

- After N_E epochs, SG will have done $n \times N_E$ iterations and so

$$\mathbb{E}\left[f(w_{nN_E}) - \min_{w \in \mathbb{R}^d} f(w)\right] \leq O\left(\frac{1}{\sqrt{n}\sqrt{N_E}}\right) \quad (K = nN_E \text{ above})$$

For $n \gg N_E$ (i.e. $n \gg 1$), $\frac{1}{\sqrt{n}\sqrt{N_E}} \ll \frac{1}{N_E}$: Better rate for SG !

(2) More on randomness

↳ SG is by design a randomized method

$\Rightarrow$ Several runs of SG can give very different results "Variance of SG"

$\Rightarrow$ But on average the method works (see convergence rates)

↳ Variants of SG have been developed to guarantee that the behavior of any run is closer to the average than for "Vanilla" SG ("Vanilla" = basic)

Batch stochastic gradient

cardinality (number of elements) in S_k

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

average of all ∇f_i's corresponding to indices in S_k

where S_k is a set of indices drawn randomly in $\{1, \dots, n\}$ (with or without replacement)

with replacement: S_k can contain multiple copies of the same index

- SG is a special case of batch SG for which $|S_k| = 1$
- GD is a special case of batch SG $|S_k| = n$ and indices drawn without replacement

$$\Rightarrow S_k = \{1, \dots, n\}$$

- Case of interest: mini-batch SG

$$1 < |S_k| << n \quad \forall k$$

↳ In general, using a batch of more than 1 data point reduces the variance of the algorithm (compared to vanilla SG) and it can even improve the performance for $|S_k| > 1$

↳ But every iteration of batch SGV is more expensive than an iteration of vanilla SG, so finding the right value for the batch size ($|S_k|$) is not straightforward.

NB: There are other ways to reduce the variance by changing the stochastic gradient iteration

(3) Advanced SG methods

Generic iteration: defined coordinate wise

j^{th} coordinate of m_k

$$\forall j = 1..d, \quad [w_{k+1}]_j = [w_k]_j - \alpha_k \frac{[m_k]_j}{[v_k]_j}$$

$\alpha_k > 0$

$\mathbb{R}^d \ni m_k$: direction (depends on a stochastic gradient, or a batch of stochastic gradients)

$\mathbb{R}^d \ni v_k$: vector of normalization for the various coordinates

$$SG : m_k = \nabla f_{i_k}(w_k), \quad v_k = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

a) SG with momentum

$$v_k = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{but} \quad m_k = \beta_1 m_{k-1} + (1-\beta_1) \nabla f_{i_k}(w_k)$$

$\beta_1 \in (0,1)$

→ Combines previous step with current stochastic gradient ($\approx$ Heavy ball)

→ Was used for the groundbreaking results on neural network performance in 2012

b) AdaGrad (~2014)

$$m_k = \nabla f_{i_k}(w_k) \qquad [v_k]_j = \sqrt{\sum_{l=0}^{k} [\nabla f_{i_l}(w_l)]_j^2}$$

→ Normalize every coordinate according to the value of all previous stochastic gradients

→ Useful for problems with sparse gradients (lots of zero coefficients), used in recommender systems

c) Adam (2015)

"Combines the ideas of AdaGrad and SG with momentum"

AdaGrad → change v_k

SG with momentum → change m_k

→ Default training algorithm in many deep learning tasks especially NLP

$$m_k = (1-\beta_1) \sum_{l=0}^{k} \frac{\beta_1^{k-l} \nabla f_{i_l}(w_l)}{1-\beta_1^{k+1}}$$

$$[v_k]_j = \sqrt{\frac{(1-\beta_2) \sum_{l=0}^{k} \beta_2^{k-l} \left[\nabla f_{i_l}(w_l)\right]_j^2}{1-\beta_2^{k+1}}}$$