Lecture 19 - SAG

Thursday, March 25, 2021 13:34

today: • finish BCFW
• SAG

application of BCFW to SVMstruct:

• getting s_i is one loss-augmented decoding call for example i

$$\alpha_i^{(t+1)} = \alpha_i^{(t)} + \gamma_t \left(s_i^{(t)} - \alpha_i^{(t)}\right) \quad A_i$$

$\hookrightarrow A_i$

• $w = A\alpha = \sum_i \underbrace{A_i \alpha_i}_{\triangleq w_i}$ $\Rightarrow$ you update

$$w_i^{(t+1)} = w_i^{(t)} + \gamma_t \left(w_s^{(t)} - w_i^{(t)}\right)$$

$$w_s^{(t)} = \frac{1}{\lambda n}\psi_i(\hat{y}_i^{(t)})$$

(worst case) $\to O(nd)$

need to store those in memory

or

store α_i's

$$w_j^{(t+1)} = w_j^{(t)} \quad \forall j \neq i$$

(memory / computation tradeoff)

note: for line search, also need to store $b_i^T \alpha_i \triangleq l_{i,\alpha}^{(t)}$

Convergence constants

for SVMstruct, can show that $C_f^{(i)} \leq \frac{4R_i^2}{\lambda n^2}$

$R_i \triangleq \max_{y \in \mathcal{Y}_i} \|\psi_i(y)\|_2$

$R = \max_i R_i$

Hessian: $(\lambda A^T A)_{(i,y),(j,y')}$

$$= \cancel{\lambda} \frac{1}{\cancel{\lambda}^2 n^2} \langle \psi_i(y), \psi_j(y') \rangle$$

$$d_i^T H_i d_i \leq \frac{1}{\lambda n^2} \underbrace{\max_{y,y'} \langle \psi_i(y), \psi_i(y') \rangle}_{R_i^2} \underbrace{\|d_i\|_2^2}_{\leq 4}$$

use affine invariance carefully

compare $C_f \leq \frac{4R^2}{\lambda \cdot 1}$

$$\Rightarrow C_f^{\otimes} = \sum_{i=1}^n C_f^{(i)} \leq \frac{4R^2}{\lambda n} \approx \frac{C_f}{\boxed{n}}$$

i.e. BCFW is "n times" faster than batch FW for SVMstruct?

importance of affine invariance:

$$C_f \leq L_{\|\cdot\|} \cdot \text{diam}_{\|\cdot\|}(M)^2$$

bad

if you use ℓ_2-norm, we get a bound? $\dim(M) = 2n$

Lipschitz constant in ℓ_2-norm = largest e-value of Hessian

recall Hessian $\lambda A^T A = \frac{1}{\lambda n^2}\left(\langle \psi_i(y), \psi_j(y')\rangle\right)_{(i,y),(j,y')}$

say eg. $\langle \psi_i(y), \psi_i(y')\rangle \approx 1$ for lots of output

$(\mathbb{1}\mathbb{1}^T)\mathbb{1} = \dim \cdot \mathbb{1}$

$\rightarrow$ get largest e-value can scale with dim of matrix $\Rightarrow$ really bad here because dim is exponential

- instead, want to use ℓ_1-norm of $\Delta_{|\mathcal{Y}_i|}$

i.e. ℓ_∞-norm for Lipschitz constant

then get $L_i(\dim(\Delta_{|\mathcal{Y}_i|})) \approx \frac{4R^2}{\lambda}$

on M, use ℓ_1-norm on $\Delta_{|\mathcal{Y}_i|}$ and then max over blocks

i.e. $\alpha = \max_i \|\alpha_i\|_1$ $\quad \ell_1$-ℓ_∞ or ℓ_∞-ℓ_1?

variance reduced SGD

setup: $\min_x \underbrace{\frac{1}{n}\sum_{i=1}^{n} f_i(x)}_{\triangleq f(x)}$ where f is μ-strongly convex, L-smooth

batch gradient method [Cauchy 18th century]

$$x_{t+1} = x_t - \gamma \underbrace{\frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x_t)}_{O(n) \text{ to compute}} \qquad \gamma = \frac{1}{L}$$

$f(x_t) - f(x^*) \leq (1-\rho)^t (f(x_0) - f^*)$ — linear rate

where $\rho \triangleq \frac{\mu}{L} = \frac{1}{\kappa}$

$\kappa \triangleq \frac{L}{\mu}$ "condition #"

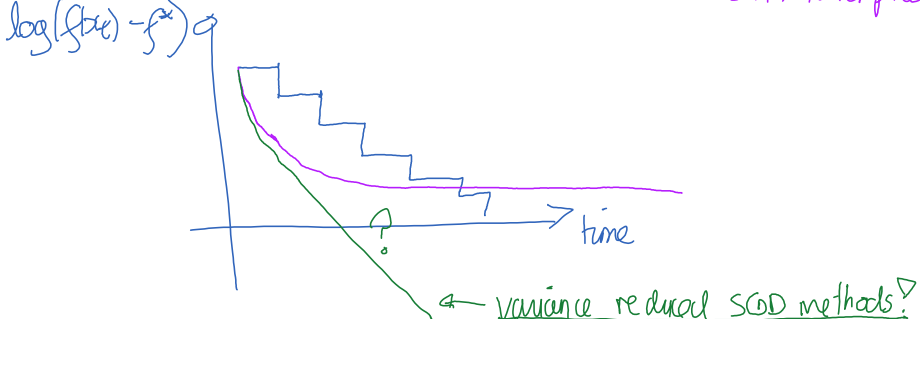
stochastic gradient method
a.k.a. incremental gradient method
[Robbins & Monro 1951]

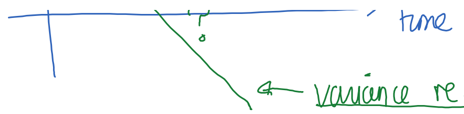
$$x_{t+1} = x_t - \gamma_t \nabla f_{i_t}(x_t)$$

where $i_t \sim \text{Unif}(\{1,\dots,n\})$

$O(1)$ to compute

- $\gamma_t = \text{const.} \cdot \gamma$ $\Rightarrow$ linear rate up to a ball of radius γ
- $\gamma_t \approx \frac{1}{\mu t}$ $\Rightarrow$ $\tilde{O}\left(\frac{1}{\mu t}\right)$ rate i.e. sublinear





← variance reduced SGD methods?

SAG, SAGA, SVRG, SDCA, OEG, etc...

14h22

SAG (stochastic average gradient) [Le Roux, Schmidt & Bach NeurIPS 2012]

(Lagrange opt. prize 2018)

SAG:
- store past gradient for each i (g_i)
- update one at step t

SAG $\Big\{$ pick i_t unif.; update $g_{i_t}^{(t+1)} = \nabla f_{i_t}(x_t)$

$$g_j^{(t+1)} = g_j^{(t)} \text{ for } j \neq i_t$$

$$x_{t+1} = x_t - \gamma \frac{1}{n} \sum_{j=1}^{n} g_j^{(t+1)} \leftarrow \text{stored } \underline{\text{stale}} \text{ gradients}$$

$$\frac{1}{n}\left[\sum_{i=1}^{n} g_i^{(t)} + g_{i_t}^{(t+1)} - g_{i_t}^{(t)}\right]$$

$$\frac{1}{n}\sum_i g_i \approx \text{approx of } \nabla f(x_t)$$

$O(1)$ cost per iteration

[but $O(nd)$ storage cost in worst case]

big surprise: converge linearly and fast

"increment aggregated gradient" (IAG) [Blatt & al. 2007] where you cycle deterministically through $\{1, \dots, n\}$

→ linear rate for quadratic function but $\gamma_{max} \approx O\left(\frac{1}{nL}\right)$ (tiny rate)

big step-size vs.

SAG convergence rate:

thm.: with $\gamma_t = \frac{1}{16L}$ where $L = \max_i (\text{Lipschitz}(\nabla f_i))$

$$\mathbb{E} f(x_t) - f^* \leq \left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8n}\right\}\right)^t C_0$$

C_0 constant

$$\rho_{SAG} = \min\left\{\frac{1}{16\kappa_{SAG}}, \frac{1}{8n}\right\} \quad \text{vs.} \quad \rho_{grad} \approx \frac{1}{\kappa_{grad}}$$

example: ℓ_2-reg. log regression on RCV1

$$n = 700\text{k} \quad L = 0.25 \quad \mu = \frac{1}{n} \qquad \left(\kappa = \frac{n}{4}\right)$$

rate comparison

$n \quad\quad 4)$

rate comparison

gradient method $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.9998$

accelerated gradient (Nesterov) $\left(1-\sqrt{\frac{\mu}{L}}\right) = 0.99761$

Nesterov lower bound $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right) \approx 0.99048$

SAG (n iterations) $(1-\rho_{SAG})^n \approx 0.88250$

practical aspects (see Schmidt & al. Math prog. 2016 paper)

a) storage: if $f_i(w) = h(x_i^T w)$ $\Rightarrow$ $\nabla_w f_i(w) = \underbrace{h'(x_i^T w)}_{\text{scalar}} \overset{\text{input data}}{x_i}$

instead $O(n \cdot d)$ storage $\rightsquigarrow$ $O(n)$ storage

b) initialization of g_i's? best: run SGD for one pass then switch SAG/SAGA

c) step-size?

- $\frac{1}{(4)L}$
- cheap line search heuristic (comes from FISTA)

"wrong inequality"

while $\left\{ f_i\left(w_t - \frac{1}{\tilde{L}_i}\nabla f_i(w_t)\right) \geq f_i(w_t) - \frac{1}{2\tilde{L}_i}\|\nabla f_i(w_t)\|^2 \right.$

set $\tilde{L}_{i,new} = 2\tilde{L}_{i,old}$

else $\tilde{L}_{i,new} = \left(\frac{1}{2}\right)^{1/n} \tilde{L}_{i,old}$

d) non-uniform sampling? sample $i \sim \frac{\tilde{L}_i}{\sum_j \tilde{L}_j}$

e) stopping criterion? you can use $\frac{1}{n}\sum_j g_j^{(t)}$ as approx $\nabla f(x_t) = \frac{1}{n}\sum_{j=1}^{n} \nabla f_j(x_t)$

f) sparse features? $x_{t+1} = x_t - \gamma\left[\underbrace{\nabla f_{i_t}(x_t) - g_i^t}_{\text{sparse}} + P_{S_i}\left[\underbrace{\frac{1}{n}\sum_{j=1}^{n} g_j^t}_{\text{dense}}\right]\right]$ (sparse SAGA)

weighted projection on support of x_i
$S_i = \{u : (x_i)_u \neq 0\}$