Lecture 19 - SAG

Tuesday, March 28, 2023 2:32 PM

today: • finish BCFW
• SAG

application of BCFW to SVMstruct:

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

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

• $w = A\alpha = \sum_i A_i \alpha_i$, $\triangleq w_i$

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

need to store these in memory

$$\begin{cases} w_i^{(t+1)} = w_i^{(t)} + \gamma_t \left(w_s^{(t)} - w_i^{(t)} \right) \\ \qquad \hookrightarrow \frac{1}{\lambda n} \psi_i(\hat{y}_i^{(t)}) \\ w_j^{(t+1)} = w_j^{(t)} \quad \forall j \neq i \end{cases}$$

or store α_i's

(memory / computation tradeoff)

note: for line search, also need to store $b_i^T g_i \triangleq \ell_{i,s}^{(t)}$

convergence constants:

for SVMstruct, can show that $C_f^{(i)} \leq \frac{4R_i^2}{\lambda n^2}$ $\qquad R_i \triangleq \max_{y \in \mathcal{Y}_i} \|\psi_i(y)\|_2$

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

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

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

$\leq R_i^2$ $\qquad \leq 2$

$R \triangleq \max_i R_i$

compare with $C_f \leq \frac{4R^2}{\lambda n}$

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

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

importance of affine invariance:

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

If you use ℓ_2-norm, we get a bad bound? $\text{diam}(\mathcal{U}) \approx 2n$

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

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

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

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

→ get largest e-value can scale with dim of a matrix
⇒ really bad here because dim is exponentially

- instead, want to use ℓ_1-norm of $\Delta(y, y_i)$
  i.e. ℓ_∞-norm for Lipschitz constant
  then get $L_i \approx (\text{diam}(\Delta(y, y_i)))^2 \approx \frac{4R^2}{\Delta}$

variance reduced SGD

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

$f(x_t) - f^* \le (1-\rho)^t (f(x_0) - f^*)$ ← linear rate

batch gradient method [Cauchy 18th century]

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

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

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

stochastic gradient method
aka. incremental gradient method
[Robbins & Monro 1951]

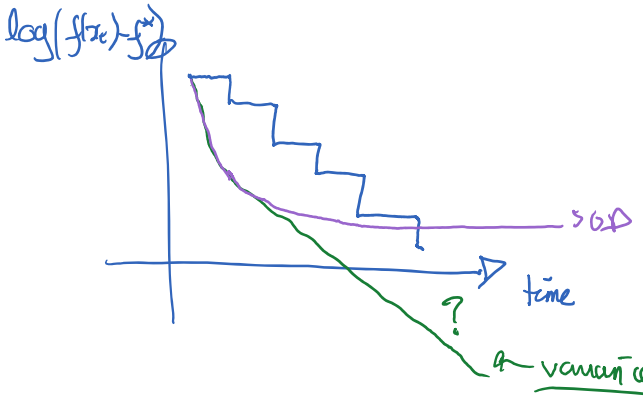
$x_{t+1} = x_t - \gamma_t \nabla f_{i_t}(x_t)$

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

$O(1)$ to compute

$\gamma_t = \text{const. } \gamma$ ⇒ linear rate up to a ball of radius

$\gamma_t = \frac{1}{\mu t}$ ⇒ $\tilde{O}(\frac{1}{t})$ rate
i.e. sublinear



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

15h33

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

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

$$x_{t+1} = x_t - \gamma \frac{1}{n} \sum_{i=1}^{n} g_i^{(t+1)}$$ ← stored stale gradients

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

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

- $O(1)$ cost per iteration; big surprise: converge linearly and fast

[but $O(nd)$ storage cost in worst case] "incremental aggregated gradient" (IAG) [Blatt' & al. 2007] where you cycle deterministically through $\{1,\dots,n\}$

→ linear rate for quadratic fct.

big step-size vs but $\gamma_{max} \approx O(\frac{1}{n})$ (sting rate)

SAG convergence rate

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

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

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

example: ℓ_2-reg. log-regression on RCV1

$n = 700K$ $L = 0.25$ $\mu = \frac{1}{n}$ $(\kappa = \frac{n}{4})$

rate comparison

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

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

SAG (in iterations) $(1-\rho_{SAG})^n \approx 0.88825$

practical aspects [see Schmidt & al. Math Prog. 2016 paper]

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

instead of $O(n \cdot d)$ 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 $f_i\left(w_t - \frac{1}{\tilde{L}_i}\nabla f_i(w_t)\right) \geq f_i(w_t) - \frac{1}{2\tilde{L}_i}\|\nabla f_i(w_t)\|^2$
  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 $i \propto \tilde{L}_i$, i.e. $p_i = \frac{\tilde{L}_i}{\sum_j \tilde{L}_j}$

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

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

sparse; dense

weighted projection on support of x_{i_t}

$S_{i_t} \triangleq \{u : (x_{i_t})_u \neq 0\}$