Lecture 19 - SAG

Tuesday, March 19, 2019 14:40

today: · continue BCFW on SVMstruct
· SAG / SAGA etc.

extensions of BCFW:

- non-uniform sampling e.g. $\{ C_f^{(i)}, g_i(x) \}$
- using away-step, etc... to "get" linear convergence

} ICML 2016

Application of BCFW on SVMstruct:

- getting s_i is one loss-augmented decoding call for example i
- $w = A\alpha = \sum_i A_i \alpha_i$, $\triangleq w_i$

need to store these in memory or store α_i } (memory / computation tradeoff)

→ you update

$$\alpha_i^{(t+1)} = \alpha_i^{(t)} + \gamma (s_i^{(t)} - \alpha_i^{(t)})$$

$$w_i^{(t+1)} = w_i^{(t)} + \gamma (w_s^{(t)} - w_i^{(t)}) \quad \text{where } w_s^{(t)} \to \frac{1}{\lambda n} A_i(\hat{y}_i^{(t)})$$

} A_i

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

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

Convergence constants:

for SVMstruct, can show that $C_f^{(i)} \leq 4R^2$ $\Rightarrow C_f^{\otimes} \leq 4R^2 \approx C_f$

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

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

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

uses affine invariance crucially

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

if use ℓ_2-norm, get a bad bound ☹ $\text{diam}(M)^2 = 2n$

Lipschitz constant in ℓ_2-norm = largest e-value of 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 e.g. $\langle \psi_i(y), \psi_j(y') \rangle \approx 1$ for lots of outputs

$$\left(11^T\right)1 = 1 \cdot \dim$$

→ get largest e-value can scale with dim. of matrix

(here, dim is exponential → useless bound)

- instead, want to use ℓ_2-norm on $\Delta_{n\leq 1}$ — $\ell_p(v) \geq \ell_q(v)$ when $p \leq q$

  i.e. ℓ_∞-norm for Lipschitz constant...

  then get $L \cdot \text{diam}(M)^2 \approx \frac{4B^2}{1}$

  on M, use ℓ_1-ℓ_∞ norm (ℓ_∞-ℓ_1?)

  i.e. $\|\cdot\| = \max_i \|\cdot_i\|_1$

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 and L-smooth

batch gradient method: [Cauchy 18th century]

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

$\gamma = \frac{1}{L}$ — linear rate: $f(x_t) - f^* \leq (1-\rho)^t (f(x_0) - f^*)$

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

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

stochastic 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

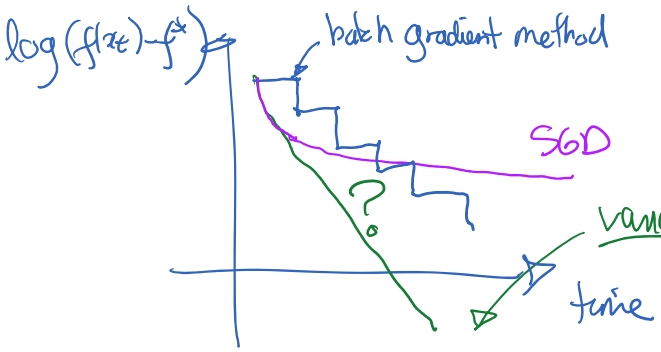
aka. **incremental gradient method**

- $\gamma_t = \text{const.} \gamma$ → linear rate up to a ball of radius γ
- $\gamma_t \approx \frac{1}{\mu t}$ → $\tilde{O}(\frac{1}{\mu t})$ rate

$\ell_{\log}(f(x_t) - f^*) \leq$ batch gradient method

ie. sublinear

deterministic gradient method



$O(1)$ to compute

ie. sublinear

variance reduced SGD methods?

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

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

(15h30)

(Lagrange opt. prize in 2018]

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

SAG
$$\begin{cases} \text{pick } i_t \sim \text{unif}; \quad \text{update } g_{i_t}^{(t+1)} \triangleq \nabla f_{i_t}(x_t) \\ \qquad\qquad g_j^{(t+1)} = g_j^{(t)} \ \forall j \neq i \\ x_{t+1} = x_t - \gamma \frac{1}{n} \sum_{i=1}^{n} g_i^{(t+1)} \end{cases}$$

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

$g_i^{(t+1)}$ ← stale gradient

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

$O(1)$ cost per iteration
[but $O(n)$ storage cost]

big surprise: converge linearly and fast

↳ vs,

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

→ linear rate for quadratic function

"big step size" but $\gamma_{max} \approx O(\frac{1}{n})$ ($\leftarrow$ tiny rate)

SAG convergence rate: thm: with $\gamma_t = \frac{1}{16L}$ where $L = \max_i$ [Lipschitz ∇f_i]

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

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

example: (Greg.) Logistic regression on RCV1

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

rate comparison:

gradient method $\left(\frac{L-\mu}{L+\mu}\right)^2 = 0.99998$

accelerated grad. (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) $(1-D_{SAG})^n = 0.88250$

practical aspects (see Schmidt Math. prog. 2016 paper) big prize?

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 memory best: run SGD for one pass, then start SAG/SAGA

c) step-size?

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

$$\begin{cases} \text{while } f_i\left(w_t - \frac{1}{\tilde{L}_i}\nabla f_i(w_t)\right) \geq f_i(w_t) - \frac{1}{2\tilde{L}_i}\|\nabla f_i(w_t)\|^2 \\ \text{set } \tilde{L}_{i,new} = 2\tilde{L}_{i,old} \\ \text{else } \tilde{L}_{i,new} = \left(\frac{1}{2}\right)^{1/n}\tilde{L}_{i,old} \end{cases}$$

use $\frac{1}{(\max_i \tilde{L}_i)}$ as step-size (see paper)

d) non-uniform sampling? sample $i \sim \frac{L_i}{\sum_j L_j}$

e) Stopping Criterion? you can use $\frac{1}{n}\sum_j g_j^t$ as approx. of $\nabla f(x_t)$

f) sparse features?

$$x_{t+1} = x_t - \gamma\left[\nabla f_{i_t}(x_t) - g_i^t + P_{S_i}\left[\frac{1}{n}\sum_j g_j^t\right]\right] \quad \text{(sparse SAGA)}$$

$\nabla f_{i_t}(x_t)$: sparse

$P_{S_i}[\frac{1}{n}\sum_j g_j^t]$: dense?

[Leblond & al.] 2017

weighted projection on support of z_i

$S_i = \{u : (z_i)_u \neq 0\}$