Lecture 19 - SAG

Tuesday, March 24, 2020 14:13

today : • finish BCFW
• SAG

convergence result for BCFW

$C_f^{(i)} \leq L_i \, \mathrm{diam}(M_i)^2$ $\qquad C_f^{\otimes} \triangleq \sum_{i=1}^{n} C_f^{(i)}$

$$\mathbb{E}[\underbrace{f(x^{(k)}) - f^*}_{\triangleq \varepsilon_k}] \leq \frac{2n}{k+2n}\left[C_f^{\otimes} + f(x^{(0)}) - f^*\right] \quad \text{for } \gamma_k = \frac{2n}{k+2n}$$

if you use line search : $\varepsilon_k \leq \frac{2n C_f^{\otimes}}{k - k_0 + 2n}$ for $k \geq k_0$

$k_0 \leq n \log_2 \frac{f(x^{(0)}) - f^*}{C_f^{\otimes}}$ ← time to ensure that $\varepsilon_k \leq C_f^{\otimes}$

batch FW $\varepsilon_k \leq \frac{2C_f}{k+2}$ one can show that $C_f^{\otimes} \leq C_f$ for quadratic functions

BCFW $\varepsilon_k \leq \frac{2n C_f^{\otimes}}{k - k_0 + 2n}$ $\qquad C_f$ vs. $n C_f^{\otimes}$

but BCFW is often n times cheaper than batch FW

complexity (# of oracle calls)
FW $\mathcal{O}\left(\frac{nC_f}{\varepsilon}\right)$
BCFW $\mathcal{O}\left(\frac{nC_f^{\otimes}}{\varepsilon}\right)$ (when not using parallelization)

$\Rightarrow$ BCFW is "never" slower than batch FW

extensions to BCFW :
- non-uniform sampling i.e. $\begin{cases} C_f^{(i)} \\ C_{g_i^*}(x) \end{cases}$ } ICML 2015
- using away-step, etc. to "get" linear convergence

Application of BCFW on SVM struct :

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

- $w = A\alpha = \sum_i A_i \alpha_i$ ($\triangleq w_i$)

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

→ you update $w_i^{(k+1)} = w_i^{(k)} + \gamma\left(w_s^{(k)} - w_i^{(k)}\right)$ ($w_s = \frac{1}{\lambda n}\psi_i(y_i^{(k)})$)

$w_j^{(k+1)} = w_j^{(k)}$

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

in memory
or store α_i (memory / computation tradeoff)

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

convergence constants

for SVMstruct, can show that $C_f^{(i)} \leq \frac{4R_i^2}{\lambda n^2}$ $\quad R_i \triangleq \max_{\tilde{y} \neq y_i} \|\psi_i(\tilde{y})\|$ $\quad R \triangleq \max_i R_i$

$\Rightarrow C_f^{(\alpha)} = \sum_i C_f^{(i)} \leq \frac{4R^2}{\lambda n} \simeq \frac{C_f}{n}$

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

use affine invariance crucially

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

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

if use ℓ_2-norm, we get a bad bound? $\quad \operatorname{diam}(M)^2 = 2n$

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

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

$\rightarrow$ get largest evalue can scale with dim. of matrix

* instead, want to use ℓ_1-norm of $\Delta\psi_{y_i}$

ie. ℓ_∞-norm for Lipschitz constant.

then get $L_i \operatorname{diam}(\Delta\psi_{y_i}) \leq \frac{4R^2}{\lambda}$

on M, use ℓ_1-norm on $\Delta\psi_{y_i}$ and then max over blocks

ie. $\|\alpha\| = \max_i \|\alpha_i\|_2$ $\quad \ell_1$-ℓ_∞ or ℓ_∞-ℓ_1?

15h35

variance reduced SGD

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

L-smooth

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

$\triangleq f(x)$ L-smooth

batch gradient method [Cauchy 18th century]

$x_{t+1} = x_t - \gamma \frac{1}{n}\sum_{i=1}^n \nabla f_i(x_t)$

$\nabla f(x_t)$ $O(n)$ to compute

linear rate

$\gamma = \frac{1}{L}$ $f(x_t) - f^* \le (1-\rho)^t (f(x_0) - f^*)$

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

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

Stochastic gradient method

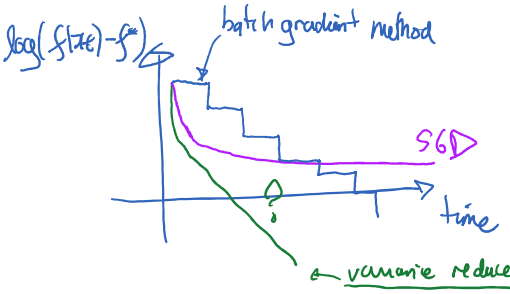
aka. incremental gradient method

[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

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



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

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

(Lagrange opt. prize in 2018)

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

SAG { pick $i_t \sim$ Unif ; update $g_{i_t}^{(t+1)} \triangleq \nabla f_{i_t}(x_t)$

$g_j^{(t+1)} = g_j^{(t)}$ 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

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

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

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

big surprise : converge linearly and fast

"increment aggregated gradient" (IAG) [Blatt et al. 2007]
where you cycle deterministically through $\{1,\dots,n\}$
→ linear rate for quadratic function
but $\gamma_{max} \approx O(\frac{1}{nL})$ ({ tiny rate)

"big step size" vs.

"big step size" vs. but $\gamma_{max} \approx O(\frac{1}{nL})$ (→ 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

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

Example: l_2-reg log. regression on RCV1

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

rate comparisons:

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 $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right) = 0.99048$

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

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

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

instead of $O(d \cdot n)$ storage → $O(n)$ storage

b) initialization of g_i's? 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

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

d) non-uniform sampling? sample in L_i

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

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

[Leblond & al. 2017]

$\nabla f_{i_t}(x_t) - g_i^t$: sparse

$\frac{1}{n_j}\sum_j g_j^t$: dense

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