Lecture 19 - SAG

Thursday, March 28, 2024 9:38 AM

today: · finish BCFW
· SAG

application of BCFW to SVMstruct:

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

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

$\alpha_i^{(t)} \in \Delta_{n_i}$

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

$\Rightarrow$ you update

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

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

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

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

need to store these in memory or store α_i's

(memory / computation tradeoff)

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

convergence constants

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

$R = \max_i R_i$

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

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

$$(\lambda n^2) d_i^T H_i d_i = \sum_{y,y'} d_{i,y} d_{i,y'} \langle \psi_i(y), \psi_i(y') \rangle$$

$$\leq \sum_{y,y'} |d_{i,y}| |d_{i,y'}| \underbrace{|\langle \psi_i(y), \psi_i(y') \rangle|}_{\leq R_i^2}$$

$$\leq R_i^2 \left(\sum_y |d_{i,y}| \right) \left(\sum_{y'} |d_{i,y'}| \right)$$

$\|d_i\|_1 \quad \|d_i\|_1$

$$\leq 4R_i^2$$

$\max_{s,x \in \Delta} \|s - x\|_1 = 2$

$$d_i^T H_i d_i \leq \frac{4R_i^2}{\lambda n^2}$$

compare with $C_f \leq \frac{4R^2}{\lambda}$ (1)

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

ie. BCFW is "n times" faster than batch FW for SVM struct

Skipped:

importance of affine invariance:

importance of affine invariance:

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

$f(x_t) - f(x^*) \le L \|x_0 - x^*\|_2^2$

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

recall Hessian $\nabla A^T A = I$ $\left(\langle \psi_i(y), \psi_j(y') \rangle\right)_{(i,y),(j,y')}$

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

$\Delta n^2 \quad (11^T)\Delta = (\dim) \cdot \Delta$

→ 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 $\Delta(x_i)$

  i.e. ℓ_∞-norm for Lipschitz constant

  then get $L_i (\mathrm{diam}(\Delta_{(x_i)}))^2 \approx \frac{4R^2}{n}$

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]

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

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

$\gamma = \frac{1}{L}$

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

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

stochastic gradient method

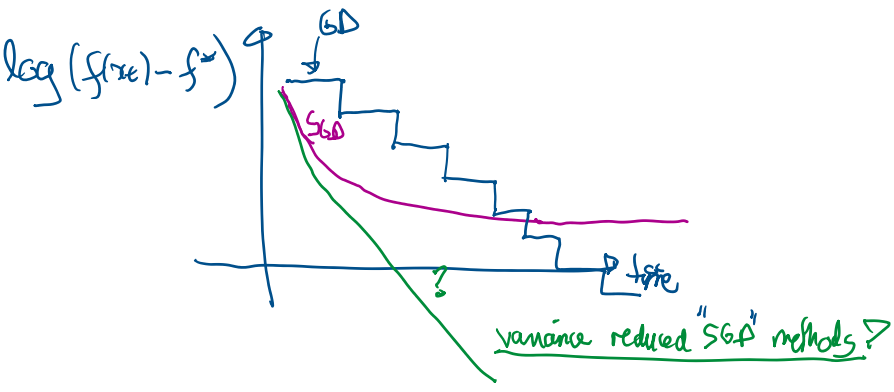
aka. incremental gradient method

[Robbins & Monro 1951]

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

where $i_t \sim \mathrm{unif}(\{1, \dots, n\})$ $O(1)$ to compute

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



10h39

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

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

(Lagrange opt. prize 2018)

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

SAG { pick i_t unif. j update $g_{i_t}^{(t+1)} = \nabla f_{i_t}(x_t)$
$g_j^{(t+1)} = g_j^{(t)}$ for $j \neq i_t$

$$x_{t+1} = x_t - \gamma \frac{1}{n} \sum_{i=1}^{n} g_i^{(t+1)} \leftarrow \text{stored } \underline{\text{state}} \text{ gradients}$$

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

$\frac{1}{n}\sum_i g_i \approx$ 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 functions

but $\gamma_{max} \approx O\left(\frac{1}{n}\right)$ (↯ tiny rate)

→ big step-size vs.

SAG convergence rate

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

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

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

example: ℓ_2-reg. log. regression on RCV1

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

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

accelerated gradient (Nesterov) $\left(1-\sqrt{\frac{\mu}{L}}\right) = 0.9976$

Nesterov lower bound $\left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right) = 0.9905$

SAG (n iterations) $(1-D_{SAG})^n \approx 0.8825$

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

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

instead $O(nd)$ storage $\hookrightarrow O(n)$ storage

b) initialization of g_i^t's? best: run SGD for one pass then switch SAG/SAGA

c) step-size? • $\frac{1}{16L}$

• cheap line search heuristic (comes from FISTA)

while $f_i\left(w_t - \frac{1}{\tilde{L}_i}\nabla f_i(w_t)\right) \geq f_i(w_t) - \frac{1}{2\tilde{L}_i}\|\nabla f_i(w_t)\|_2^2$ 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 $i \sim \frac{\tilde{L}_i}{\sum_j \tilde{L}_j}$

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

f) sparse features? $x_{t+1} = x_t - \gamma\left[\underbrace{\nabla f_{i_t}(x)}_{\text{sparse}} - g_i^t + \underbrace{P_{S_i}}_{\substack{\text{weighted projection on} \\ \text{support of } x_i}}\left[\underbrace{\frac{1}{n}\sum_j g_j^t}_{\text{dense}}\right]\right]$ (sparse SAGA)

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