Lecture 20 - variance reduction

Tuesday, March 30, 2021 14:14

today: • variance reduction perspective
• application to CRF

variance reduction idea

X & Y are R.V.'s

goal: estimate $\mathbb{E}X$ using M.C. samples

Suppose: $\mathbb{E}Y$ is cheap to compute and Y is correlated with X

consider estimator

$\alpha \in [0,1]$ $\quad \Theta_\alpha \triangleq \alpha(X-Y)+\mathbb{E}Y$ to approximate $\mathbb{E}X$

α convex combo coef. between $\mathbb{E}X$ & $\mathbb{E}Y$

properties: $\mathbb{E}\Theta_\alpha = \alpha\mathbb{E}X + (1-\alpha)\mathbb{E}Y$ → unbiased (i.e. $\mathbb{E}\Theta_\alpha = \mathbb{E}X$) if $\mathbb{E}X=\mathbb{E}Y$ [not interesting] or $\alpha=1$

variance: $\mathrm{var}(\Theta_\alpha) = \alpha^2[\mathrm{Var}X + \mathrm{Var}Y - 2\mathrm{cov}(X,Y)]$

→ variance reduction?

for $\alpha=1$ (unbiased setting) $\quad \Theta_\alpha = X + (\mathbb{E}Y - Y)$ correction

SGD setting:

X is $\nabla f_i(x_t)$; $\mathbb{E}X$ = batch gradient

SAG/SAGA algorithm: Y is g_i [past stored gradient]

$$\mathbb{E}Y = \frac{1}{n}\sum_{i=1}^{n} g_i$$

SAG alg.: $\alpha = \frac{1}{n}$ (biased)

SAGA alg.: $\alpha = 1$ ⇒ (unbiased)

$\alpha(X-Y)$ $\quad +\mathbb{E}Y$

SAG: $x_{t+1} = x_t - \gamma\left[\frac{1}{n}\left[\nabla f_{i_t}(x_t) - g_{i_t}^{(t)}\right] + \frac{1}{n}\sum_j g_j^{(t)}\right]$ (biased)

SAGA: $x_{t+1} = x_t - \gamma\left[\underbrace{1}_{(q=1)}\left[\nabla f_{i_t}(x_t) - g_{i_t}^{(t)}\right] + \frac{1}{n}\sum_j g_j^{(t)}\right]$ (unbiased)

SVRG, (stochastic variance reduced): $x_{t+1} = x_t - \gamma\left[\nabla f_{i_t}(x_t) - \nabla f_{i_t}(x_{old}) + \frac{1}{n}\sum_j \nabla f_j(x_{old})\right]$ (unbiased)

SVRG: $x_{t+1} = x_t - \gamma \left[\nabla f_{i_t}(x_t) - \nabla f_{i_t}(x_{old}) + \frac{1}{n}\sum_j \nabla f_j(x_{old}) \right]$ (unbiased)

(Stochastic variance reduced gradient)

x_{old} is updated from outer loop

SVRG alg:

for $k = 0, \ldots$ (outer loop)

compute $g_{ref} \triangleq \frac{1}{n}\sum_j \nabla f_j(x^{(k)})$

for $t = 0, \ldots, T_{max}$

sample i_t

$$x_{t+1}^{(k)} = x_t^{(k)} - \gamma \left[\nabla f_{i_t}(x_t^{(k)}) - \nabla f_{i_t}(x^{(k)}) + g_{ref} \right]$$

end

$x^{(k+1)} = x_{T_{max}}^{(k)}$

questions:
- what is T_{max}?
- what is γ?

original SVRG convergence result: need $\gamma \leq \approx \frac{1}{L}$

$T_{max} \geq \approx \frac{L}{\mu} = \kappa$ → to run alg., need to know κ

⇒ not adaptive to local strong convexity

tweak of SVRG (now called "loopless") [Hoffmann & al. NeurIPS 2015] $T_{max} \sim$ Geom(—)

[at inner loop, do a batch gradient comp. with prob $\frac{1}{n}$]

then, get same convergence result as SAGA

↳ comp. cost.: size of inner loop $\mathbb{E}[T_{max}] = n$;

overall cost of SVRG $\approx$ 3 (SGD cost) for n updates

SAG / SAGA / loopless SVRG, convergence for convex fct. ($\mu = 0$) get $\min_{t \leq T} \{\mathbb{E} f(x_t) - f^*\} = O\left(\frac{1}{T}\right)$

[contrast this with $O\left(\frac{1}{\sqrt{T}}\right)$ for SGD]

* Note: interpolation regime: $\|\nabla f_i(x^*)\| = 0 \ \forall i$

vs. just $\left\|\frac{1}{n}\sum_i \nabla f_i(x^*)\right\| = 0$

↳ get similar rate as SVRG / SAGA with SGD

CRF optimization

	primal	dual
SVM struct	$\min_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^n H_i(w)$ $H_i(w) = \max_{\tilde{y}} \ell_i(\tilde{y}) - w^T\psi_i(\tilde{y})$ (margin)	$\max_{\alpha_i \in \Delta_{\|Y_i\|}} -\frac{\lambda}{2}\|w(\alpha)\|^2 + \frac{1}{n}\sum_{i=1}^n \ell_i^T \alpha_i$
CRF	$\min \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^n -\log p(y^{(i)}\|x^{(i)};w)$ $\log\left(\sum_{\tilde{y}} \exp(-w^T\psi_i(\tilde{y}))\right)$	$\max_{\alpha_i \in \Delta_{\|Y_i\|}} -\frac{\lambda}{2}\|w(\alpha)\|^2 + \frac{1}{n}\sum_{i=1}^n H_i(\alpha_i)$ (entropy) $\triangleq -\sum_{\tilde{y}} \alpha_i(\tilde{y}) \log \alpha_i(\tilde{y})$

KKT $\rightarrow$ $w(\alpha) = \frac{1}{\lambda n}\sum_i \sum_{\tilde{y}} \alpha_i(\tilde{y})\psi_i(\tilde{y})$

$= \frac{1}{\lambda n}\sum_i \sum_{c \in C} \mu_{i,c}(\tilde{y}_c)\psi_{i,c}(\tilde{y}_c)$ ← from MRF

$p(y|x;w) \propto \exp(\langle w, \psi(x,y)\rangle)$

(*) at optimality

$\alpha_i^*(y) = p(y|x^{(i)}; w(\alpha^*))$

$\alpha_i^* \in$ interior of $\Delta_{|Y_i|}$ unlike sparse soln in SVM struct

15h27

CRF optimization

- primal is smooth [vs. non-smooth for SVM struct]
  - for a while, batch L-BFGS was method of choice [batch $\Rightarrow$ slow for large n]
  - [Collins & al. JMLR 2008]: online exponentiated gradient (OEG)

block-coordinate method on dual; exponentiated gradient step on block dual obj.:

$\alpha_i(\tilde{y})^{(t+1)} \propto \alpha_i(\tilde{y})^{(t)} \exp\left(-\gamma_t \nabla_{\alpha_i} D(\alpha^{(t+1)})\right)$

EG alg $\rightarrow$ proximal gradient step using $KL(\alpha\|\alpha^{(t)})$ as a Bregman divergence for prox term

$\rightarrow$ get linear convergence rate with cheap $O(r)$ updates (like SGD) [vs. $O(n)$ for batch methods]

[can think of it as a variance reduced method as well?]

- SAGA for CRF [Schmidt & al. AISTATS 2015]

$w^{(t+1)} = (1-\lambda\gamma_t)w^{(t)} - \gamma_t\left[\nabla f_{i_t}(w^{(t)}) - g_i^{(t)} + \frac{1}{n}\sum_j g_j^{(t)}\right]$

- SDCA (stochastic dual coordinate ascent)

$\alpha_{i_t}^{(t+1)}(y) = (1-\gamma_t)\alpha_{i_t}^{(t)}(y) + \gamma_t \hat{s}_i^{(t)}(y)$, $\hat{s}_i^{(t)}(y) \in \mathbb{E}[q_i]$ (stochastic)

$p(y|x_i; w(\alpha^{(t)}))$

as related fixed pt.

coordinate ascent)

↳ SOTA for CRF [Lacoste-Julien & al. UAI 2018] thanks to line search

as related fixed pt. update $\alpha_i^{t+1} = p(y|x_i; w(\alpha^t)) \forall i$

$p(y|x_i; w(\alpha^{(t)})) \triangleq \alpha_i(w)$ related to subgradient primal

[note: BCFW is a special case of SDCA on SVM struct obj??]

proximal gradient method

↳ generalization of projected gradient method to other non-smooth fcts.

composite framework $F(w) \triangleq f(w) + \Omega(w)$ where f is convex & L-smooth, Ω " " but not nec smooth

- constrained opt. $\Omega(w) = \delta_M(w) \triangleq \begin{cases} 0 & \text{if } w \in M \\ +\infty & \text{o.w.} \end{cases}$ ("indicator fct." on M)
- ℓ_1-regularization $\Omega(w) = \|w\|_1$

proximal gradient update:

$$w_{t+1} = \underset{w}{\operatorname{argmin}} \underbrace{f(w_t) + \langle \nabla f(w_t), w - w_t \rangle + \frac{1}{2\gamma_t} \|w - w_t\|_2^2}_{\triangleq B_t(w)} + \Omega(w)$$

- if $\gamma \leq \frac{1}{L}$, then $f(w) \leq B_t(w) \; \forall w$
- we can rewrite $B_t(w) = \frac{1}{2\gamma_t} \|w - [w_t - \gamma_t \nabla f(w_t)]\|_2^2 + \text{cst.}$ (by completing the square)

$\Rightarrow$ if $\Omega(w) = \delta_M(w)$, we get projected grad., obv

$$w_{t+1} = \text{Prox}_{\gamma_t}^{\Omega}(w_t - \gamma_t \nabla f(w_t))$$

↳ "proximal operator"

$$\text{prox}_{\gamma}^{\Omega}(z) \triangleq \underset{w}{\operatorname{argmin}} \left\{ \Omega(w) + \frac{1}{2\gamma} \|w - z\|^2 \right\}$$

could be replaced by Bregman divergence to get other generalizations, e.g. OEG

(*) like projection, prox operator is non-expansive (i.e. 1-Lipschitz) (recall lecture 11 landscape of rates)

i.e. $\|\text{prox}_x^{\Omega}(w) - \text{prox}_x^{\Omega}(w')\|_2 \leq \|w - w'\|_2$

(*) like projection, prox operator is non-expansive (i.e. 1-Lipschitz)

ie. $\|\text{prox}_\gamma^\Omega(w) - \text{prox}_\gamma^\Omega(w')\|_2 \leq \|w - w'\|_2$

(recall lecture 11 landscape of rates)

$\Rightarrow$ convergence rate for prox. gradient method on $F = f + \Omega$ are same as unconstrained gradient descent on f

* to be useful, need $\text{prox}_\gamma^\Omega$ to be efficiently computable

$$\text{prox}_\gamma^{\|\cdot\|_2}(z) = \underset{w}{\text{argmin}} \; \|w\|_2 + \frac{1}{2\gamma}\|w - z\|_2^2$$

"soft-thresholding" (component-wise) $\triangleq \begin{cases} \text{sgn}(z_d)[|z_d| - \gamma] & \text{if } |z_d| \geq \gamma \\ 0 & \text{o.w.} \end{cases}$

used e.g. for lasso: ℓ_1-regularized least-squares

FISTA $\rightarrow$ accelerated prox. gradient method

$\hookrightarrow$ SOTA for batch (prox) lasso

$\min_w \frac{1}{n}\sum_i f_i(w) + \Omega(w)$

* scikit-learn $\rightarrow$ use SAGA for lasso & ℓ_1-reg., log.-reg.

Prox SAGA $\quad w_{t+1} = \text{Prox}_\gamma^\Omega\left(w_t - \gamma\left[\nabla f_{i_t}(w_t) - g_{i_t}^{(t)} + \frac{1}{n}\sum_j g_j^{(t)}\right]\right)$

Could accelerate prox SAGA using "catalyst" (see next class)