Today's variance reduction perspective

Variance reduction idea

\[ X, Y \text{ be R.V.} \]

Goal: estimate \( EX \) using M.C. samples

Suppose: \( EY \) is cheap to compute and \( Y \) is correlated with \( X \)

Consider estimator \( \Theta_\alpha = \alpha (X-Y) + EY \) to approximate \( EX \)

\( \alpha \in [0, 1] \)

Properties:

\[ \mathbb{E}\Theta_\alpha = \alpha EX + (1-\alpha) EY \]

\( \sim \) unbiased (i.e., \( \mathbb{E}\Theta_\alpha = EX \))

if \( EY = EX \) (not interesting)

\[ \text{variance: } \text{Var}(\Theta_\alpha) = \alpha^2 \left[ \text{Var}(X) + \text{Var}(Y) - 2 \text{cov}(X,Y) \right] \]

for \( \alpha = 1 \) (unbiased setting)

\( \text{S(0) setting: } \Theta_\alpha = X + (\text{EY-Y}) \)

\( \text{variance reduction} \)
S(6)D setting:

\[ X \text{ is } \nabla f_i(x_t); \quad EX = \text{batch gradient} \]

SA6/SA6A algorithm: \( Y \) is \( g_{ti} \) [past shared gradient]

\[ EY = \frac{1}{n} \sum g_i \]

SA6 algorithm: \( \alpha = \frac{1}{n} \) (biased)

SA6A: \( \alpha = 1 \) (unbiased)

SA6:

\[ x_{t+1} = x_t - \alpha \left[ \nabla f_i_t(x_t) - g_{ti} \right] + \frac{1}{n} \sum g_{ji} \]  

(biased)

SA6A:

\[ x_{t+1} = x_t - \alpha \left[ \nabla f_i_t(x_t) - g_{ti} + \frac{1}{n} \sum g_{ji} \right] \]  

(unbiased)

SVRG:

\[ x_{t+1} = x_t - \alpha \left[ \nabla f_i_t(x_t) - \nabla f_i_t(x_{t-1}) + \frac{1}{n} \sum_{j=1}^{n} \nabla f_i_t(z_{j}) \right] \]  

(stochastic variance reduced gradient)

SVRG algorithm:

\[ x_{t+1} \text{ is updated from outer loop} \]
SVRG algorithm:

\[
\begin{align*}
\text{for } k = 0, \ldots, & \quad \text{(outer loop)} \\
& \quad \text{compute } g_{k} = \frac{1}{n} \sum_{j=1}^{n} \nabla f_{j}(x^{(k)}) \\
& \quad \text{for } t = 0, \ldots, T_{\max} \\
& \quad \quad \text{sample } i_t \\
& \quad \quad x^{(k)}_{t+1} = x^{(k)}_{t} - \gamma \left[ \nabla f_{i_t}(x^{(k)}_{t}) - \nabla f_{i_t}(x^{(k)}) + g_{k} \right] \\
& \quad \text{end} \\
& \quad x^{(k+1)} = x^{(k)} \\
& \text{end}
\end{align*}
\]

Questions:
- What is \( T_{\max} \)?
- What is \( \gamma \)?

Original SVRG convergence result:

\[
\text{need } \gamma \leq \frac{1}{L}
\]

"\( T_{\max} > \frac{1}{\mu} \text{ or } k \approx \frac{L}{\mu} \) \Rightarrow \text{is run alg., need to know } k_{\ast} \Rightarrow \text{not adaptive, no local strong convexity}
\]

break on SVRG:
- \( [\text{Hoffmann et al. NIPS 2015}] \)

\( T_{\max} \sim \text{Geom}(\cdot) \)

[after inner loop iteration, do batch gradient step]
CRF Optimization

- Primal objective is smooth & strongly convex [vs non-smooth for SVM

- For a while, batch L-BFGS was method of choice [batch => slow for large n]

- [Collins et al. JMLR 2005]: Online exponentiated gradient (OEG)

  \[ \alpha_i(t)_{(t+1)} = \alpha_i(t)_{(t)} \exp \left( -\eta \beta \nabla_i \ell(x(t)) \right) \]

  EOG alg \rightarrow primal gradient step using \( KL(\alpha | 1_{m}) \) as Bregman divergence

  \( \rightarrow \) get linear convergence rate with cheap \( C(1) \) updates (like SGD)

  \[ \text{Live } \alpha(n) \text{ for batch method} \]

  I can think of it as variance reduced SGD as well.

SAOA for CRF:

\[ \omega^{(t+1)} = (1 - \alpha \beta) \omega^{(t)} - \alpha \beta \left[ \nabla_i \ell_i(x(t)) - g_i^{(t)} \right] \]

Schmidt et al.
Proximal gradient method

A generalization of projected gradient method for non-smooth functions.

**Composite framework:** \( F(w) = f(w) + \Omega(w) \) where \( f \) is convex & \( L \)-smooth.

- **Constrained opt.** \( \Omega(w) = \Sigma_M(w) \equiv \left\{ \begin{array}{ll} 0 & \text{if} \ w \in M \ 
\infty & \text{else} \end{array} \right. \)

  - "Indicator of \( M \)

- \( L_1 \)-regularization \( \Omega(w) = \|w\|_1 \)

**Proximal gradient update:**

\[
\omega_{t+1} = \text{prox}_{\Theta}(f(w_t) + \langle \nabla f(w_t), w_t - w \rangle + \frac{1}{2} \| w_t - w \|^2 + \Theta(w))
\]

[Note: BCFW is a special case of SDCA on SVSHeard data.]
\[
\omega_{t+1} = \arg\min_{\omega} \frac{1}{2} \langle \nabla f(\omega_t) , \omega - \omega_t \rangle + \frac{1}{2\delta_t} \| \omega - \omega_t \|^2 + R(\omega)
\]

\[\cong B_{\delta_t}(\omega)\]

If \( \delta_t \leq \frac{1}{L} \), then \( f(\omega) \leq B_{\delta_t}(\omega) + R(\omega) \)

we can rewrite \( B_{\delta_t}(\omega) = \frac{1}{2\delta_t} \| \omega - \frac{1}{\delta_t} \nabla f(\omega_t) \|^2 + c(\omega) \)

then completing the square

we get the projected gradient alg.

\[\omega_{t+1} = \text{prox}_{\frac{\delta_t}{2}} \left( \omega_t - \frac{\delta_t}{\delta_t} \nabla f(\omega_t) \right)\]

\[\Rightarrow \text{"proximal operator" } \text{prox}_{\frac{\delta_t}{2}}(z) \cong \arg\min_{\omega} \frac{1}{2} \langle \omega - \omega_t \rangle + \frac{1}{2\delta_t} \| \omega - \omega_t \|^2 + R(\omega)\]

like projection, prox operator is non-expensive (i.e. 1-lipschitz)

i.e. \[\| \text{prox}_{\frac{\delta_t}{2}}(w) - \text{prox}_{\frac{\delta_t}{2}}(z) \| \leq \frac{\| w - z \|}{2\delta_t}\]

\[\Rightarrow \text{convergence rate of prox. gradient method}\]

\[\text{one same as unconstrained gradient descent}\]

\[\text{could replace with proximal algorithm to get then generalization (e.g. O(\delta^2))}\]