Today's: variance reduced SGD
  - prox method, catalyst, etc...

Variance reduction idea:

\[ \nabla \mathbb{E} f = \mathbb{E} [\nabla f] = \mathbb{E} [\nabla f] \]

SGD: approximate \( \mathbb{E} f \) with just \( x \) (i.e. \( n \hat{x} \))

General idea:

Goal: estimate \( \mathbb{E} f \) using M.C. samples

Suppose: \( \mathbb{E} f \) is cheap to compute and \( y \) is correlated with \( x \)

Consider estimator \( \hat{g} = \alpha (x-y) + \mathbb{E} x \) to approximate \( \mathbb{E} x \)

(Here \( x \) and \( y \) are i.i.d.)

\[ \mathbb{E} \hat{g} = \alpha \mathbb{E} x + (1-\alpha) \mathbb{E} y \]

\( \Rightarrow \) unbiased if \( \alpha = \frac{1}{2} \)

\( \Rightarrow \) \( \alpha = 1 \)

Variance:

\[ \text{Var}(\hat{g}) = \alpha^2 \left[ \text{Var}(x) + \text{Var}(y) - 2 \text{Cov}(x,y) \right] \]

\( \Rightarrow \) variance reduction aspect

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

\[ \hat{g} = x + [y - \mathbb{E} y] \]

For SGD softmax.
SAG/SAGA algorithm: $Y = \sum_{i=1}^{n} \frac{g_i}{\epsilon}$ [post stored gradient]

SAG algorithm: $\alpha = \eta$

Standard SAG: $w_{k+1} = w_k - \eta \sum_{i=1}^{k} g_i$

$$\sum a^t + \nabla f_i(w_t) - a^t$$
\[ \text{SAG:} \quad w_{t+1} = w_t - \alpha \left[ \nabla E_i(w_t) - \frac{\partial E}{\partial w_t} \right] \quad (\text{direct}) \]

\[ \text{SAGA:} \quad w_{t+1} = w_t - \alpha \left[ \nabla E_i(w_t) - \frac{\partial E}{\partial w_t} + \frac{1}{\eta} \frac{\partial E}{\partial w_t} \right] \quad (\text{unbiased, i.e. } \mathbb{E}[w_t] = \nabla E_i(w_t) \text{ for fixed } w_t \text{ from outer loop}) \]

\[ \text{SVRG: } \quad w_{t+1} = w_t - \alpha \left[ \nabla E_i(w_t) - \frac{\partial E}{\partial w_t} + \frac{1}{n} \sum_{i=1}^{n} \nabla E_i(w_{t\text{avg}}) \right] \quad (\text{for fixed } w_t \text{ from outer loop}) \]

\[ \text{SVRG: } \quad k = 0, \ldots, \quad (\text{outer loop}) \]
- compute \( g_{\text{avg}} = \frac{1}{n} \sum_{i=1}^{n} \nabla E_i(w_{t\text{avg}}) \)
- \( w_{t\text{avg}} = w_t \)
- for \( t = 0, \ldots, t_{\text{max}} \)
  - sample \( i_t \)
  - \( w_{t+1} = w_t - \alpha \left[ \nabla E_i(w_{t\text{avg}}) - \frac{\partial E}{\partial w_t} + g_{\text{avg}} \right] \)
- end
- \( w_{t_{\text{max}}} = w_{t_{\text{max}}} \)

Questions:
- What is \( t_{\text{max}} \)?
- What is \( \alpha \)?

\[ \text{SAG: } \quad \text{need to store } g_t \quad \text{\( \mathcal{O}(n) \) } \quad \text{no } t_{\text{max}} \text{ to tune} \]

\[ \text{SAGA: } \quad \text{no } t_{\text{max}} \text{ to tune} \quad \text{no weight 2 loops} \]

\[ \text{SVRG: } \quad \text{track } t_{\text{max}} \text{, use } 2 \text{ loops} \quad \text{only need to store } w_{t\text{avg}} \]

SVRG convergence results:
- \( n \leq 1 \)
- \( t_{\text{max}} \geq \frac{\alpha}{\eta} \)
- \( \Rightarrow \text{in practice, } t_{\text{max}} \approx \max \frac{\alpha}{\eta}, n, k \)

Important remark: "adaptively, to stream correctly"
important concept: "adaptivity for strong convexity"

SA6 result \( \lambda \geq \frac{1}{L} \Rightarrow \text{algorithm parameters do not depend on } \mu \) (strongly)

SVRG \( \lambda \geq \frac{1}{L} \text{ but } t_{\max} = \max \{ n, \lambda \} \gtrsim \) depends on \( \mu \)

for just convex SDN (\( \mu = 0 \)), get \( \min \epsilon \in [\frac{L}{2}, \epsilon^*] = O(\frac{1}{\sqrt{\epsilon}}) \)

[contrast with \( \frac{\sqrt{\epsilon}}{\epsilon} \) for SGD]

SA6A "simple" convergence result:

if you use \( \lambda = \frac{a}{n} \) for SA6A for \( \frac{\delta}{\lambda} \) i.e. \( \mathbb{E}(\epsilon(x^*) - f^*) \leq (1 - p) \epsilon C_0 \)

then rate \( p \gtrsim \frac{1}{n} \min \frac{\delta \lambda}{\epsilon}, \frac{1}{a} \)

two effects:

- bigger \( \lambda \) size \( \Rightarrow \) bigger variance

- smaller \( \lambda \) \( \Rightarrow \) slower gradient descent rate (determined by \( \frac{1}{\lambda} \))

is \( K < n \), then as long as \( \frac{\epsilon \lambda}{n} \geq \frac{1}{\lambda} \text{ i.e. } A_{k \lambda} \geq \frac{\epsilon \lambda}{n} \), we get same rate (roughly)

\( \Rightarrow \) "niceness" to step size when \( K = n \)

SA6A complexity to reach \( \epsilon \) error \( O((n + k) \log \frac{1}{\epsilon}) \) "SGD steps"

Practical aspects of SA6/SA6A:

- storage: if \( S_i(\omega) = h(x_i^,\omega) \), \( \Delta S_i(\omega) = \frac{\partial h}{\partial \omega} x_i \)
b) initialization? hack which is to use \[ \frac{1}{\| \tilde{v} \|} \leq g_i^2 \] where \( S_j = \sum_i \tilde{v}_i^2 \tilde{v}_i \) has been visited before \( t_j \).

c) step-size? \[ \frac{1}{L} \]

- cheap line search heuristic

\[
\text{while } \frac{\sum_i (\tilde{v}_i - \frac{1}{L} \nabla f_i(t_i)) + \frac{1}{L} \| \nabla f_i(t_i) \|_2^2}{L} \geq \frac{\sum_i f_i(t_i) - \frac{1}{L} \| \nabla f_i(t_i) \|_2^2}{L} \]

set \( L = L - 4 \) and end.

(come from FISTA)

\[ L \] is smoothening term

use stepsize \( \frac{1}{L} \).

a) non-uniform sampling? sample \( i \sim \frac{1}{\tilde{v}_i^2} \) then get rate with \( \frac{1}{\tilde{v}_i^2} \) instead of max \( \frac{1}{\tilde{v}_i^2} \).

e) stopping criterion? you can use \( \frac{1}{\tilde{v}_i^2} \geq g_i^2 \) as approximate \( \nabla f_i(x_t) \).

f) sparse features? two tricks:

1) "hard thresholding" \( \tilde{w}_k = \tilde{w}_k^d - \langle \tilde{w}_k^d \rangle \tilde{g}_i^2 \) [see note on top]

2) \( f_i(x) = \frac{1}{2} \| x \|_2^2 + h(x) \)

\[ \tilde{w}_{k+1} = (1-\lambda) \tilde{w}_k - \lambda \left( \sum_{i \in \text{support of } x^*} \tilde{g}_i^2 \right) \tilde{w}_k \]

\[ S_i = \tilde{w}_k : (x) \cdot \tilde{g}_i \]
Pointers

- Variance reduction perspective on SAG / SAGA / SVRG -- in SAGA paper:
  - A. Defazio, F. Bach and S. Lacoste-Julien
    SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives
    NIPS 2014
  - Other pointers:
    - SVRG paper:
      Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. NIPS 2013
    - Note that a variant of SVRG that is *adaptive to local strong convexity* is given in the following paper (where the end of the inner loop is decided randomly: at every inner loop iteration, with probability 1/n, you end the inner loop):
      T. Hofmann, A. Lucchi, S. Lacoste-Julien, and Brian McWilliams
      Variance Reduced Stochastic Gradient Descent with Neighbors, NIPS 2015
  - The practical aspects of SAG are described in the massive journal version:
      Minimizing Finite Sums with the Stochastic Average Gradient
      Mathematical Programming, 162:83-162, 2017 | arxiv
    - An alternative to the complicated "lagged updates" when you have sparse features is the Sparse SAGA algorithm; see Section 2 of:
      ASAGA: Asynchronous Parallel SAGA,
      R. Leblond, F. Pedregosa and S. Lacoste-Julien
      AISTATS 2017