CPSC 540: Machine Learning

Stochastic Average Gradient

Mark Schmidt

University of British Columbia

Winter 2018

Last time: Stochastic sub-gradient

- We discussed minimizing finite sums,

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w),$$

  when n is very large.

- For non-smooth f_i, we discussed stochastic subgradient method,

$$w^{k+1} = w^k - \alpha_k g_{i_k},$$

  for some $g_{i_k} \in \partial f_{i_k}(w^k)$ for some random $i_k \in \{1, 2, \ldots, n\}$.
  - May increase f, but moves closer to w^* for small α_k in expectation.

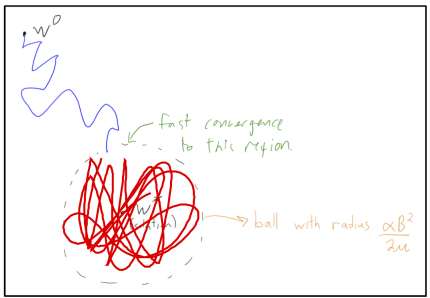
- Same rate as deterministic subgradient method but n times faster.

Stochastic Gradient with Constant Step Size

- Our bound on expected distance with constant step-size:

$$\mathbb{E}[\|w^k - w^*\|^2] \leq (1 - 2\alpha\mu)^k \|w^0 - w^*\|^2 + \frac{\alpha B^2}{2\mu},$$

- First term looks like linear convergence, but second term does not go to zero.



- Justifies the "divide the step-size in half if it looks like it's stalled" heuristic.
  - Dividing α in half divides radius of the ball around w^* in half.

Stochastic Gradient with Decreasing Step Size

- To get convergence, we need a decreasing step size.
  - We need effect of B^2 to go to zero, but we still need to make progress.
  - Classic approach is to choose α_k such that

$$\sum_{k=1}^{\infty} \alpha_k = \infty, \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty,$$

which suggests setting $\alpha_k = O(1/k)$.

  - If $\alpha_k = \frac{1}{\mu k}$ we can show

$$\begin{aligned} \mathbb{E}[f(\bar{w}^k) - f^*] &= O(\log(k)/k) && \text{(non-smooth } f\text{)} \\ &= O(1/k) && \text{(smooth } f\text{)} \end{aligned}$$

for the average iteration $\bar{w}^k = \frac{1}{k}\sum_{k'=1}^{k} w^{k'}$.

Outline

Stochastic Subgradient with Sparse Features

- For many datasets, our feature vectors x^i are very sparse:

"CPSC"	"Expedia"	"vicodin"	<recipient name>	...
1	0	0	0	...
0	1	0	0	...
0	0	1	0	...
0	1	0	1	...
1	0	1	1	...

- Consider case where d is huge but each row x^i has at most z non-zeroes:
  - The $O(d)$ cost of stochastic subgradient might be too high.
  - We can often modify stochastic subgradient to have $O(z)$ cost.

Digression: Operations on Sparse Vectors

- Consider a vector $g \in \mathbb{R}^d$ with at most z non-zeroes:

$$g^T = \begin{bmatrix} 0 & 0 & 0 & 1 & 2 & 0 & -0.5 & 0 & 0 & 0 \end{bmatrix}.$$

- If $z << d$, we can store the vector using $O(z)$ storage instead of $O(d)$:
  - Just store the non-zero values:

$$g_{\text{value}}^T = \begin{bmatrix} 1 & 2 & -0.5 \end{bmatrix}.$$

  - Store index of each non-zero ("pointer"):

$$g_{\text{point}}^T = \begin{bmatrix} 4 & 5 & 7 \end{bmatrix}.$$

- With this representation, we can do standard vector operations in $O(z)$:
  - Compute αg in $O(z)$ by setting $g_{\text{value}} = \alpha g_{\text{value}}$.
  - Compute $w^T g$ in $O(z)$ by multiplying g_{value} by w at positions g_{point}.

Stochastic Subgradient with Sparse Features

- Consider optimizing the hinge-loss,

$$\underset{w\in\mathbb{R}^d}{\text{argmin}} \frac{1}{n}\sum_{i=1}^{n} \max\{0, 1 - y^i(w^T x^i)\},$$

when d is huge but each x^i has at most z non-zeroes.

- A stochastic subgradient method could use

$$w^{k+1} = w^k - \alpha_k g_{i_k}, \text{ where } g_i = \begin{cases} -y^i x^i & \text{if } 1 - y^i(w^T x^i) > 0 \\ 0 & \text{otherwise} \end{cases}$$

- Calculating w^{k+1} is $O(z)$ since these are sparse vector operations.

- So stochastic subgradient is fast if z is small even if d is large.
  - This is how you "train on all e-mails": each e-mail has a limited number of words.

Stochastic Subgradient with Sparse Features

- But consider the L2-regularized hinge-loss in the same setting,

$$\underset{w\in\mathbb{R}^d}{\text{argmin}} \frac{1}{n}\sum_{i=1}^{n}\max\{0, 1 - y_i(w^T x^i)\} + \frac{\lambda}{2}\|w\|^2,$$

  using a stochastic subgradient method,

$$w^{k+1} = w^k - \alpha_k g_{i_k} - \alpha_k \lambda w^k, \text{ where } g_{i_k} \text{ is same as before.}$$

- Problems is that w^k could have d non-zeroes:
  - So adding L2-regularization increases cost from $O(z)$ to $O(d)$?

- There are two standard ways to keep the cost at $O(z)$:
  - L2-regularization: use a $w^k = \beta^k v^k$ (scalar times vector) representation (bonus).
  - "Lazy" updates (which work for many regularizers).

Lazy Updates for Sparse Features with Dense Regularizers

- Consider a feature j that has been zero in the loss for 10 iterations (constant α):

$$\begin{aligned} w_j^k &= w_j^{k-1} - 0 - \alpha\lambda w_j^{k-1} \\ &= (1-\alpha\lambda) w_j^{k-1} \\ &= (1-\alpha\lambda)^2 w_j^{k-2} \\ &\vdots \\ &= (1-\alpha\lambda)^{10} w_j^{k-10}. \end{aligned}$$

- So we can apply 10 regularizer gradient steps in $O(1)$.

- Lazy updates:
  - If j is zero in g_{i_k}, do nothing.
  - If j is non-zero, apply all the old regularizer updates then do the gradient step.
    - Requires keeping a "checkpoint" of the last time each variable was updated.

Lazy Updates for Sparse Features with Dense Regularizers

- Lazy updates that track cumulative effects of simple updates.

- Considern stochastic proximal-gradient for L1-regularization:
  - Soft-threshold operator with constant step-size α applies to each element,

$$w_j^{k+1} = \text{sign}(w_j^k)\max\{0, |w_j^k| - \alpha\lambda\}.$$

  - If all that happens to w_j for 10 iterations is the proximal operator, we can use

$$w_j^{k+10} = \text{sign}(w_j^k)\max\{0, |w_k^k| - 10\alpha\lambda\}.$$

- Digression: stochastic proximal-gradient methods:
  - Same convergence rates as basic stochastic gradient method (doesn't help).
  - Open problem: convergence rate of stochastic proximal-gradient for non-convex.

Stochastic Subgradient Methods in Practice

- We've said that α_k must go to zero for convergence.
- Theory says using $\alpha_k = 1/\mu t$ is close to optimal.
  - Except for some special cases, you should not do this.
  - Usually $\mu = O(1/n)$ or $O(1/\sqrt{n})$ so initial steps are huge.
  - Later steps are tiny: $1/k$ gets small very quickly.
  - Convergence rate slows dramatically if μ isn't accurate.
  - No adaptation to "easier" problems than worst case.
- Decreasing step-sizes are also hard to tune.
- They also make it hard to decide when to stop.

Practical Step-Sizes

- Tricks that can improve theoretical and practical properties:
  1. Use smaller initial step-sizes, that go to zero more slowly:

$$\alpha_k = \gamma/\sqrt{k},$$

     or just use a constant step-size,

$$\alpha_k = \gamma,$$

     which we showed converges linearly to $O(\gamma)$-ball around the solution.
  2. Take a (weighted) average of the iterations or gradients:

$$\bar{w}^k = \sum_{k'=1}^{k} \omega_{k'} w^{k'},$$

     where ω_k is weight of iteration k.
     - Could weight all iterations equally.
     - Could ignore first half of the iterations then weight equally.
     - Could weight proportional to k.

Speeding up Stochastic Subgradient Methods

- Results that support using large steps and averaging:
  - Averaging later iterations achieves $O(1/k)$ in non-smooth case.
  - Gradient averaging improves constants in analysis.
  - $\alpha_k = O(1/k^\beta)$ for $\beta \in (0.5, 1)$ more robust than $\alpha_k = O(1/k)$.
  - Constant step size $(\alpha_k = \alpha)$ achieves linear rate to accuracy $O(\alpha)$.
  - In smooth case, iterate averaging is asymptotically optimal:
    - Achieves same rate as optimal stochastic Newton method.

- These tricks usually help, but tuning is often required:
  - Stochastic subgradient is not a black box.

Stochastic Newton Methods?

- Should we use Nesterov/Newton-like stochastic methods?
  - These do not improve the $O(1/\epsilon)$ convergence rate.
- But some positive results exist.
  - Nesterov/Newton can improve constant factors.
  - Two-phase Newton-like method achieves $O(1/\epsilon)$ without strong-convexity.
  - AdaGrad method,

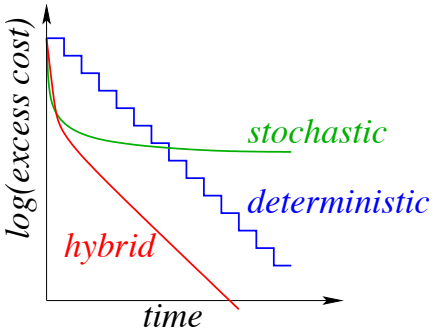
$$w^{k+1} = w^k + \alpha D g_{i_k}, \quad \text{with diagonal } D_{jj} = \sqrt{\sum_{k'=1}^{k} \|\nabla_j f_{i_{k'}}(w^k)\|^2},$$

improves "regret" but not optimization error.

- Popular variations are RMSprop and Adam.
- Recent work argues these may give worse final test error than basic method.

Outline

Better Methods for Smooth Objectives and Finite Datasets?



- Stochastic methods:
  - $O(1/\epsilon)$ iterations but requires 1 gradient per iterations.
  - Rates are unimprovable for general stochastic objectives.
- Deterministic methods:
  - $O(\log(1/\epsilon))$ iterations but requires n gradients per iteration.
  - The faster rate is possible because n is finite.
- For finite n, can we design a better method?

Hybrid Deterministic-Stochastic

- Approach 1: control the sample size.
- Deterministic method uses all n gradients,

$$\nabla f(w^k) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w^k).$$

- Stochastic method approximates it with 1 sample,

$$\nabla f_{i_k}(w^k) \approx \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w^k).$$

- A common variant is to use larger sample $\mathcal{B}^k$

$$\frac{1}{|\mathcal{B}^k|}\sum_{i\in\mathcal{B}^k} \nabla f_i(w^k) \approx \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w^k),$$

particularly useful for vectorization/parallelization.

- For example, with 16 cores set $|\mathcal{B}^k| = 16$ and compute 16 gradients at once.

Approach 1: Batching

- The SG method with a sample $\mathcal{B}^k$ uses iterations

$$w^{k+1} = w^k - \frac{\alpha_k}{|\mathcal{B}^k|} \sum_{i \in \mathcal{B}^k} \nabla f_i(w^k).$$

- Let's view this as a "gradient method with error",

$$w^{k+1} = w^k - \alpha_k(\nabla f(w^k) + e^k),$$

where e^k is the difference between approximate and true gradient.

- If you use $\alpha_k = 1/L$, then using descent lemma this algorithm has

$$f(w^{k+1}) \leq f(w^k) - \underbrace{\frac{1}{2L}\|\nabla f(w^k)\|^2}_{\text{good}} + \underbrace{\frac{1}{2L}\|e^k\|^2}_{\text{bad}},$$

for any error e^k.

Approach 1: Batching

- Our progress bound with $\alpha_k = 1/L$ and error in the gradient of e^k is

$$f(w^{k+1}) \leq f(w^k) - \underbrace{\frac{1}{2L}\|\nabla f(w^k)\|^2}_{\text{good}} + \underbrace{\frac{1}{2L}\|e^k\|^2}_{\text{bad}}.$$

- We can use the batch size $|\mathcal{B}^k|$ control error size e^k.
  - If we sample with replacement we get

$$\mathbb{E}[\|e^k\|^2] = \frac{1}{|\mathcal{B}^k|}\sigma^2,$$

    where σ^2 is the variance of the gradient norms.
    - "Doubling the batch size cuts the error in half".
  - If we sample without replacement from a training set of size n we get

$$\mathbb{E}[\|e^k\|^2] = \frac{n - |\mathcal{B}^k|}{n}\frac{1}{|\mathcal{B}^k|}\sigma^2,$$

    which drives error to zero as batch size approaches n.

Approach 1: Batching

- The SG method with a sample $\mathcal{B}^k$ uses iterations

$$w^{k+1} = w^k - \frac{\alpha_k}{|\mathcal{B}^k|} \sum_{i \in \mathcal{B}^k} \nabla f_i(w^k).$$

- For a fixed sample size $|\mathcal{B}^k|$, the rate is sublinear.

- But we can grow $|\mathcal{B}^k|$ to achieve a linear rate:
  - Early iterations are cheap like SG iterations.
  - Later iterations can use a sophisticated gradient method.
    - No need to set a magical step-size: use a line-search.
    - Can incorporate linear-time approximations to Newton.

- Another approach: at some point switch from stochastic to deterministic:
  - Often after a small number of passes (but hard to know when to switch).

Stochastic Average Gradient

- Growing $|\mathcal{B}^k|$ eventually requires O(n) iteration cost.

- **Can we have 1 gradient per iteration and only $O(\log(1/\epsilon))$ iterations?**
  - YES! First method was the stochastic average gradient (SAG) algorithm in 2012.

- To motivate SAG, let's view gradient descent as performing the iteration

$$w^{k+1} = w^k - \frac{\alpha_k}{n} \sum_{i=1}^{n} v_i^k,$$

  where on each step we set $v_i^k = \nabla f_i(w^k)$ for all i.

- SAG method: only set $v_{i_k}^k = \nabla f_{i_k}(w^k)$ for a randomly-chosen i_k.
  - All other v_i^k are kept at their previous value.

Stochastic Average Gradient

- The SAG iteration is

$$w^{k+1} = w^k - \frac{\alpha_k}{n} \sum_{i=1}^{n} v_i^k,$$

  where on each iteration we set $v_{i_k}^k = \nabla f_{i_k}(w^k)$ for a randomly-chosen i_k.

- Unlike batching, we use a gradient for every example.
  - But the gradients might out of date.

- Stochastic variant of earlier increment aggregated gradient (IAG).
  - Selects i_k cyclically, which destroys performance.

- Key proof idea: $v_i^k \to \nabla f_i(w^*)$ at the same rate that $w^k \to w^*$:
  - So bad term $\|e^k\|^2$ converges linearly to 0.

Convergence Rate of SAG

If each ∇f_i is $L-$continuous and f is strongly-convex, with $\alpha_k = 1/16L$ SAG has

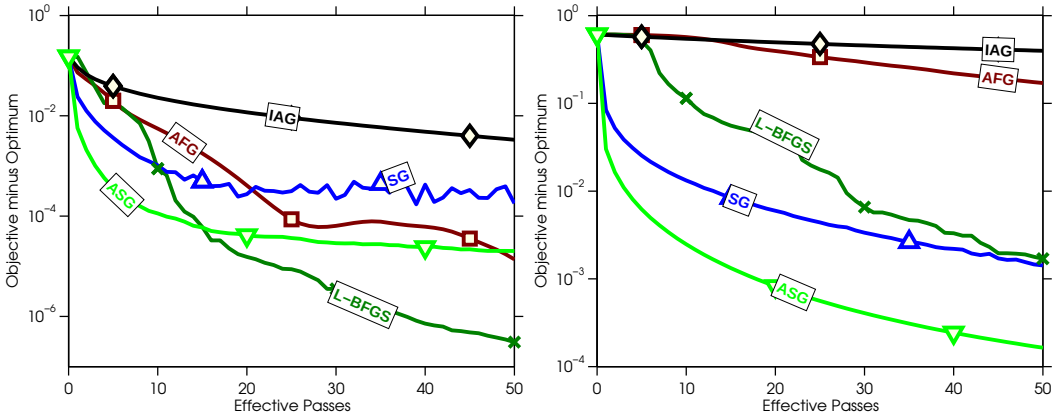
$$\mathbb{E}[f(w^k) - f(w^*)] \leqslant O\left(\left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8n}\right\}\right)^k\right)$$

- Number of ∇f_i evaluations to reach accuracy ϵ:
  - Stochastic: $O(\frac{L}{\mu}(1/\epsilon))$. (Best when n is enormous)
  - Gradient: $O(n\frac{L}{\mu}\log(1/\epsilon))$.
  - Nesterov: $O(n\sqrt{\frac{L}{\mu}}\log(1/\epsilon))$. (Best when n is small and L/μ is big)
  - SAG: $O(\max\{n, \frac{L}{\mu}\}\log(1/\epsilon))$.

- But note that the L values are again different between algorithms.

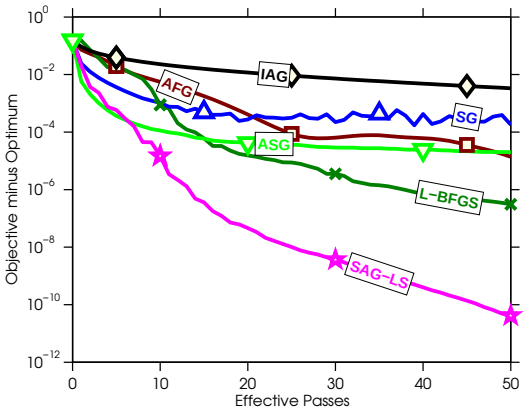
Comparing Deterministic and Stochastic Methods

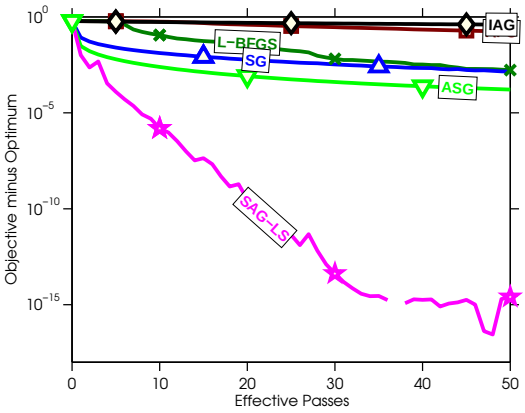
- Two benchmark L2-regularized logistic regression datasets:



SAG Compared to Deterministic/Stochastic Methods

- Two benchmark L2-regularized logistic regression datasets:





Summary

- Practical aspects of stochastic gradient methods:
  - Lazy updates allow regularization with sparse datasets.
  - Different step-size strategies and averaging significantly improve performance.
- Increasing batch sizes:
  - Leads to linear rate in terms of iterations.
  - Makes setting the step-size easier
- Stochastic average gradient: $O(\log(1/\epsilon))$ iterations with 1 gradient per iteration.
- Next time: ways to handle $n = \infty$ and $d = \infty$.

Stochastic Subgradient with Sparse Features

- But consider the L2-regularized hinge-loss in the same setting,

$$\operatorname*{argmin}_{w\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \max\{0, 1 - y_i(w^T x^i)\} + \frac{\lambda}{2}\|w\|^2,$$

  using a stochastic subgradient method,

$$w^{k+1} = w^k - \alpha_k g_{i_k} - \alpha_k \lambda w^k, \text{ where } g_{i_k} \text{ is same as before}$$

- Problems is that w^t could have d non-zeroes:
  - So adding L2-regularization increases cost from $O(k)$ to $O(d)$?
- To use L2-regularization and keep $O(k)$ cost, re-write iteration as

$$\begin{aligned} w^{t+1} &= w^t - \alpha_t g_{i_t} - \alpha_t \lambda w^t \\ &= \underbrace{(1 - \alpha_t \lambda) w^t}_{\text{changes scale of } w^t} - \underbrace{\alpha_t g_{i_t}}_{\text{sparse update}}. \end{aligned}$$

Stochastic Subgradient with Sparse Features

- Let's write the update as two steps

$$w^{t+\frac{1}{2}} = (1 - \alpha_t\lambda)w^t, \quad w^{t+1} = w^{t+\frac{1}{2}} - \alpha_t g_{i_t}.$$

- We can implement both steps in $O(k)$ if we re-parameterize as

$$w^t = \beta^t v^t,$$

for some scalar β^t and vector v^t.

- For the first step we can use

$$\beta^{t+\frac{1}{2}} = (1 - \alpha_t\lambda)\beta^t, \quad v^{t+\frac{1}{2}} = v^t.$$

which costs $O(1)$.

- For the second step we can use

$$\beta^{t+1} = \beta^{t+\frac{1}{2}}, \quad v^{t+1} = v^{t+\frac{1}{2}} - \frac{\alpha_t}{\beta^{t+\frac{1}{2}}} g_{i_t},$$

which costs $O(k)$.