CPSC 540: Machine Learning

Stochastic Subgradient

Mark Schmidt

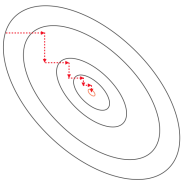
University of British Columbia

Winter 2019

Last Time: Coordinate Optimization

- In coordinate optimization we only update one variable on each iteration.

$$w_{j_k}^{k+1} = w_{j_k}^k - \alpha_k \nabla_k f(w^k),$$



- More efficient than gradient descent if the iterations are d-times cheaper.
  - True for pairwise separable f like label propagation,

$$f(w) = \sum_{i=1}^{d} f_i(w_j) + \sum_{(i,j)\in E} f_{ij}(w_i, w_j).$$

under random choice of j_k.

Analyzing Coordinate Descent

- To analyze coordinate descent, we can write it as

$$w^{k+1} = w^k - \alpha_k e_{j_k} \nabla_{j_k} f(w^k),$$

where "elementary vector" e_j has a zero in every position except j,

$$e_3^\top = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- We usually assume that each $\nabla_j f$ is L-Lipshitz ("coordinate-wise Lipschitz"),

$$|\nabla_j f(w + \gamma e_j) - \nabla_j f(w)| \leq L|\gamma|,$$

which for $\mathcal{C}^2$ functions is equivalent to $|\nabla^2_{jj} f(w)| \leq L$ for all i.

(diagonals of Hessian are bounded)

- This is not a stronger assumption:
  - If the gradient is L-Lipschitz then it's also coordiante-wise L-Lipschitz.

Convergence Rate of Coordinate Optimization

- Coordinate-wise Lipschitz assumption implies a coordinate-wise descent lemma,

$$f(w^{k+1}) \leq f(w^k) + \nabla_j f(w^k)(w^{k+1} - w^k)_j + \frac{L}{2}(w^{k+1} - w^k)_j^2,$$

for any w^{k+1} and w^k that only differ in coordinate j.

- With $\alpha_k = 1/L$ (for simplicity), plugging in $(w^{k+1} - w^k) = -(1/L)e_{j_k}\nabla_{j_k} f(w^k)$ gives

$$f(w^{k+1}) \leq f(w^k) - \frac{1}{2L}|\nabla_{j_k} f(w^k)|^2,$$

a progress bound based on only updating coordinate j_k.

- If we did optimal update (as in label propagation), this bound would still hold.
  - Optimal update decreases f by at least as much as any other update.

Convergence Rate of Randomized Coordinate Optimization

- Our bound for updating coordinate j_k is

$$f(w^{k+1}) \leq f(w^k) - \frac{1}{2L}|\nabla_{j_k} f(w^k)|^2,$$

so progress depends on which j_k that we choose.

- Let's consider expected progress with random selection of j_k,

$$\begin{aligned}
\mathbb{E}[f(w^{k+1})] &\leq \mathbb{E}\left[f(w^k) - \frac{1}{2L}|\nabla_{j_k} f(w^k)|^2\right] && \text{(expectation wrt } j_k \text{ given } w^k\text{)} \\
&= \mathbb{E}[f(w^k)] - \frac{1}{2L}\mathbb{E}[|\nabla_{j_k} f(w^k)|^2] && \text{(linearity of expectation)} \\
&= \underbrace{f(w^k)}_{\text{no } j_k} - \frac{1}{2L}\sum_{j=1}^{d} p(j_k = j)|\nabla_j f(w^k)|^2 && \text{(definition of expectation)}
\end{aligned}$$

Convergence Rate of Randomized Coordinate Optimization

- The bound from the previous slide is

$$E[f(w^{k+1})] \leq f(w^k) - \frac{1}{2L}\sum_{j=1}^{d} p(j_k = j)|\nabla_j f(w^k)|^2.$$

- Let's choose j_k uniformly in this bound, $p(j_k = j) = 1/d$.

$$\begin{aligned}\mathbb{E}[f(w^{k+1})] &\leq f(w^k) - \frac{1}{2L}\sum_{j=1}^{d} \frac{1}{d}|\nabla_j f(w^k)|^2 \\ &= f(w^k) - \frac{1}{2dL}\sum_{j=1}^{d} |\nabla_j f(w^k)|^2 \\ &= f(w^k) - \frac{1}{2dL}\|\nabla f(w^k)\|^2.\end{aligned}$$

Convergence Rate of Randomized Coordinate Optimization

- Our guaranteed progress bound for randomized coordinate optimization,

$$\mathbb{E}[f(w^{k+1}))] \leq f(w^k) - \frac{1}{2dL}\|\nabla f(w^k)\|^2.$$

- If we use strongly convexity or PL and recurse carefully (see bonus) we get

$$\mathbb{E}[f(w^k)] - f^* \leq \left(1 - \frac{\mu}{dL}\right)^k [f(w^0) - f^*].$$

  which means we expect to need $O\left(d\frac{L}{\mu}\log(1/\epsilon)\right)$ iterations.

- For PL functions gradient descent needs $O\left(\frac{L}{\mu}\log(1/\epsilon)\right)$ iterations.

- So coordinate optimization needs d-times as many iterations?

Randomized Coordinate Optimization vs. Gradient Descent

- If coordinate descent step are d-times cheaper then both algorithms need

$$O\left(\frac{L}{\mu}\log(1/\epsilon)\right),$$

in terms of "gradient descent iteration cost".

- So why prefer coordinate optimization?

- The Lipschitz constants L are different.
  - Gradient descent uses L_f and coordinate optimization uses L_c.
  - L_c is maximum gradient changes if you change *one* coordinate.
  - L_f is maximum gradient changes if you change *all* coordinates.

- Since $L_c \leq L_f$, coordinate optimization is faster.
  - The gain is because coordinate descent allows bigger step-sizes.
  - For [non-]convex functions, similar trade-off: $O(L_f/\epsilon)$ vs. $O(dL_c/\epsilon)$ iterations.
  - Comparison is harder with line-search/coordinate-optimization, quasi-Newton, etc.

Lipschitz Sampling

- Can we do better than choosing j_k uniformly at random?
- You can go faster if you have an L_j for each coordinate:

$$|\nabla_j f(w + \gamma e_j) - \nabla_j f(w)| \leq L_j |\gamma|.$$

- Using L_{j_k} as the step-size and sampling j_k proportional to L_j gives

$$\mathbb{E}[f(w^k)] - f^* \leq \left(1 - \frac{\mu}{d\bar{L}}\right)^w [f(w^0) - f^*],$$

  where $\bar{L}$ as the average Lipschitz constant (previously we used the maximum L_j).

- For label propagation, this requires stronger assumptions on the graph structure:
  - We need expected number of edges connected to j_k to be $O(|E|/d)$.
  - This might not be true if the high-degree nodes have the highest L_j values.

Greedy Gauss-Southwell Selection Rule

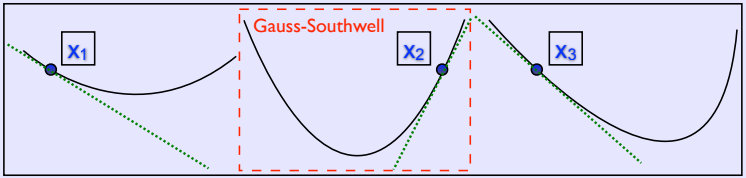
- Our bound on the progress if we choose coordinate j_k is

$$f(w^{k+1}) \leq f(w^k) - \frac{1}{2L}|\nabla_{j_k} f(w^k)|^2.$$

and the "best" j_k according to the bound is

$$j_k \in \underset{j}{\text{argmax}}\{|\nabla_j f(w^k)|\},$$

- This is called greedy selection or the Gauss-Southwell rule.



Greedy Gauss-Southwell Selection Rule

- Our bound on the progress if we choose coordinate j_k is

$$f(w^{k+1}) \leq f(w^k) - \frac{1}{2L}|\nabla_{j_k} f(w^k)|^2.$$

  and the "best" j_k according to the bound is

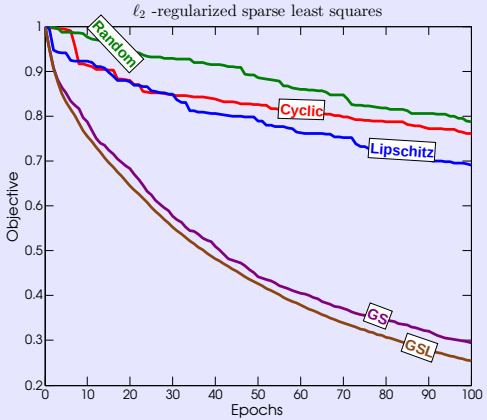
$$j_k \in \underset{j}{\text{argmax}}\{|\nabla_j f(w^k)|\},$$

- This is called greedy selection or the Gauss-Southwell rule.

- Can we ever find max gradient value d-times cheaper than computing gradient?
  - Yes, for pairwise-separable where maximum degree is similar to average degree.
    - Includes lattice-structured graphs, complete graphs, and Facebook graph.
  - You can efficiently track the gradient values and track the max with a max-heap.

Numerical Comparison of Coordinate Selection Rules

Comparison on problems where Gauss-Southwell has similar cost to random:



"Cyclic" goes through the j in order: bad worst-case bounds but often works well

There also exist accelerated coordinate descent methods.

Problems Suitable for Coordinate Optimization

- We now know that many problems satisfy the "d-times faster" condition.
- For example, composition of a smooth function with affine map plus separable

$$F(w) = f(Aw) + \sum_{j=1}^{d} f_j(w_j)$$

  for a matrix A, smooth function f, and potentially non-smooth f_j.
  - Includes L1-regularized least squares, logistic regression, etc.
- Key idea: you can track Aw as you go for a cost $O(n)$ instead of $O(nd)$ (bonus).
- In this setting, we get same rate as if non-smooth f_j were not there.

  (and faster than the sublinear $O(1/k)$ rate for subgradient methods)
- Recent works: coordinate optimization leads to faster PageRank methods.

Outline

Finite-Sum Optimization Problems

- Solving our standard regularized optimization problem

$$\underset{w\in\mathbb{R}^d}{\text{argmin}} \sum_{i=1}^{n} \text{loss}_i(w) \quad + \quad r(w),$$

$$\text{data fitting term} \quad + \quad \text{regularizer}$$

is a special case of solving the generic finite-sum optimization problem

$$\underset{w\in\mathbb{R}^d}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n} f_i(w),$$

where $f_i(w) = \text{loss}_i(w) + \frac{1}{n} r(w)$.

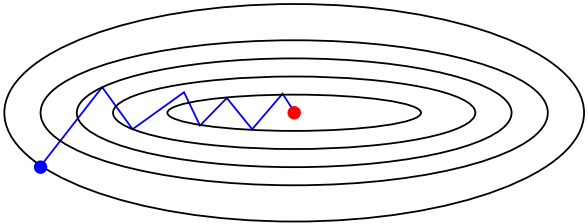
- Gradient methods are effective when d is very large.
- What if number of training examples n is very large?
  - E.g., ImageNet has ≈ 14 million annotated images.

Stochastic vs. Deterministic Gradient Methods

- We consider minimizing $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$.
- Deterministic gradient method [Cauchy, 1847]:

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

  - Iteration cost is linear in n.
  - Convergence with constant α_k or line-search.



Stochastic vs. Deterministic Gradient Methods

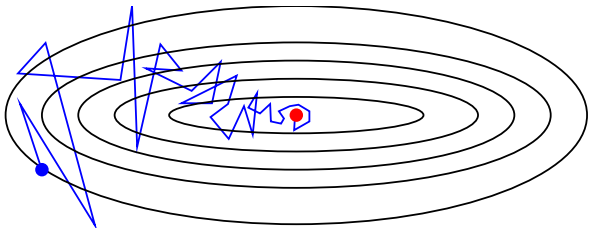
- Stochastic gradient method [Robbins & Monro, 1951]:
  - Random selection of i_k from $\{1, 2, \ldots, n\}$.

$$w^{k+1} = w^k - \alpha_k \nabla f_{i_k}(w^k).$$

  - With $p(i_k = i) = 1/n$, the stochastic gradient is an unbiased estimate of gradient,

$$\mathbb{E}[\nabla f_{i_k}(w)] = \sum_{i=1}^{n} p(i_k = i) \nabla f_i(w) = \sum_{i=1}^{n} \frac{1}{n} \nabla f_i(w) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w) = \nabla f(w).$$

  - Iteration cost is independent of n.
  - Convergence requires $\alpha_k \to 0$.



Stochastic vs. Deterministic Gradient Methods

Stochastic iterations are n times faster, but how many iterations are needed?

- If ∇f is Lipschitz continuous then we have:

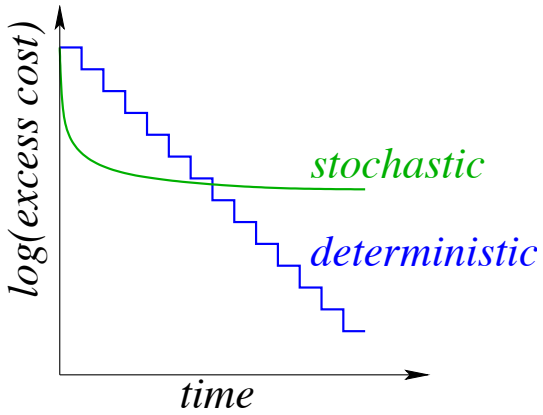
Assumption	Deterministic	Stochastic
Convex	$O(1/\sqrt{\epsilon})$	$O(1/\epsilon^2)$
Strongly	$O(\log(1/\epsilon))$	$O(1/\epsilon)$

- Stochastic has low iteration cost but slow convergence rate.
  - Sublinear rate even in strongly-convex case.
  - Bounds are unimprovable with "unbiased gradient approximation" oracle.
    - Oracle returns a g_k satisfying $\mathbb{E}[g_k] = \nabla f(w^k)$.

- Momentum and Newton-like methods do not improve rates in stochastic case.
  - Can only improve constant factors.

Stochastic vs. Deterministic Convergence Rates

Plot of convergence rates in strongly-convex case:



Stochastic will be superior for low-accuracy/time situations.

Stochastic vs. Deterministic for Non-Smooth

- The story changes for non-smooth problems.
- Consider the binary support vector machine (SVM) objective:

$$f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i(w^\top x_i)\} + \frac{\lambda}{2}\|w\|^2.$$

- Rates for subgradient methods for non-smooth objectives:

Assumption	Deterministic	Stochastic
Convex	$O(1/\epsilon^2)$	$O(1/\epsilon^2)$
Strongly	$O(1/\epsilon)$	$O(1/\epsilon)$

- So for non-smooth problems:
  - Deterministic methods are not faster than stochastic method.
  - So use stochastic subgradient (iterations are n times faster).

Subgradient Method

- The basic subgradient method:

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

  for some $g_k \in \partial f(w^k)$.
- Decreases distance to solution for small enough α_k (for convex f).

- The basic 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 $i_k \in \{1, 2, \dots, n\}$.

- Stochastic subgradient is n times faster with similar convergence properties.
- Decreases expected distance to solution for small enough α_k (for convex f).

Convergence Rate of Stochastic Gradient Method

- We'll first show progress bound for stochastic gradient assuming ∇f is Lipschitz.
  - We'll come back to the non-smooth case.

- Recall the the descent lemma applied to w^{k+1} and w^k,

$$f(w^{k+1}) \leq f(w^k) + \nabla f(w^k)^\top (w^{k+1} - w^k) + \frac{L}{2}\|w^{k+1} - w^k\|^2.$$

- Plugging in stochastic gradient iteration $(w^{k+1} - w^k) = -\alpha_k \nabla f_{i_k}(w^k)$ gives

$$f(w^{k+1}) \leq f(w^k) - \alpha_k \nabla f(w^k)^\top \nabla f_{i_k}(w^k) + \alpha_k^2 \frac{L}{2}\|\nabla f_{i_k}(w^k)\|^2.$$

Convergence Rate of Stochastic Gradient Method

- So far any choice of α_k and i_k we have

$$f(w^{k+1}) \leq f(w^k) - \alpha_k \nabla f(w^k)^\top \nabla f_{i_k}(w^k) + \alpha_k^2 \frac{L}{2} \|\nabla f_{i_k}(w^k)\|^2.$$

- Let's take the expectation with respect to i_k assuming $p(i_k = i) = 1/n$,

$$\begin{aligned}\mathbb{E}[f(w^{k+1})] &\leq \mathbb{E}[f(w^k) - \alpha_k \nabla f(w^k)^\top \nabla f_{i_k}(w^k) + \alpha_k^2 \frac{L}{2} \|\nabla f_{i_k}(w^k)\|^2] \\ &= f(w^k) - \alpha_k \nabla f(w^k)^\top \mathbb{E}[\nabla f_{i_k}(w^k)] + \alpha_k^2 \frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(w^k)\|^2],\end{aligned}$$

  where the second line uses linearity of expectation (and α_k not depending on i_k).
- We know that $\mathbb{E}[\nabla f_{i_k}(w^k)] = \nabla f(w^k)$ (unbiased) so this gives

$$\mathbb{E}[f(w^{k+1})] \leq f(w^k) - \alpha_k \underbrace{\|\nabla f(w^k)\|^2}_{\text{good}} + \alpha_k^2 \underbrace{\frac{L}{2} \mathbb{E}[\|\nabla f_{i_k}(w^k)\|^2]}_{\text{bad}}.$$

Convergence Rate of Stochastic Gradient Method

- So a progress bound for stochastic gradient is

$$\mathbb{E}[f(w^{k+1})] \leq f(w^k) - \alpha_k \underbrace{\|\nabla f(w^k)\|^2}_{\text{good}} + \alpha_k^2 \underbrace{\frac{L}{2}\mathbb{E}[\|\nabla f_{i_k}(w^k)\|^2]}_{\text{bad}}.$$

- "Good" term looks like usual measure of progress: big gradient $\rightarrow$ big progress.
- "Bad" term is the problem: less progress if gradients are very different.
  - And now choosing $\alpha_k = 1/L$ might not be small enough.
  - But we can control badness: if α_k is small then $\alpha_k >> \alpha_k^2$.

- Step-size α_k controls how fast we move towards solution.
- And squared step-size α_k^2 controls how much variance moves us away.
  - This term will destroy linear convergence.

Summary

- Convergence rate of d coordinate descent iterations is faster than gradient descent.
- Better coordinate selection with Lipschitz sampling or Gauss-Southwell.
- $f(Ax) + \sum_j f_j(w_j)$ structure also allows coordinate optimization.
  - Even for non-smooth f_j.
- Stochastic subgadient method: same rate as subgradient but n times cheaper.

- Next time: new stochastic methods with linear convergence rates.

Applying Expected Bound Recursively (Coordinate Optimization)

- Our guaranteed progress bound for randomized coordinate optimization,

$$\mathbb{E}[f(w^{k+1}))] \leq f(w^k) - \frac{1}{2dL}\|\nabla f(w^k)\|^2.$$

- If we subtract f^* and use strong-convexity or PL (as before),

$$\mathbb{E}[f(w^{k+1})] - f^* \leq \left(1 - \frac{\mu}{dL}\right)[f(w^k) - f^*].$$

- By recursing we get linear convergence rate,

$$\begin{aligned}
\mathbb{E}[\mathbb{E}[f(w^{k+1})]] - f^* &\leq \mathbb{E}\left[\left(1 - \frac{\mu}{dL}\right)[f(w^k) - f^*]\right] && \text{(expectation wrt } j_{k-1}\text{)} \\
\mathbb{E}[f(w^{k+1})] - f(w^*) &\leq \left(1 - \frac{\mu}{dL}\right)[\mathbb{E}[f(w^k)] - f^*] && \text{(iterated expectations)} \\
&\leq \left(1 - \frac{\mu}{dL}\right)^2 [f(w^{k-1}) - f^*]
\end{aligned}$$

- You keep alternating between taking an expectation back in time and recursing.

Gauss-Southwell Convergence Rate

- The progress bound under the greedy Gauss-Southwell rule is

$$f(w^{k+1}) \leq f(w^k) - \frac{1}{2L}\|\nabla f(w^k)\|_\infty^2,$$

and this leads to a faster rate of

$$f(w^k) - f^* \leq \left(1 - \frac{\mu_1}{L}\right)^k [f(w^0) - f^*],$$

where μ_1 is the PL constant in the ∞-norm

$$\mu[f(w) - f^*] \leq \frac{1}{2}\|f(w)\|_\infty^2.$$

- This is faster because $\frac{\mu}{n} \leq \mu_1 \leq \mu$ (by norm equivalences).

- If you know the L_j values, a faster rule is "Gauss-Southwell-Lipschitz".

Gauss-Southwell-Lipschitz

- Our bound on the progress with an L_j for each coordinate is

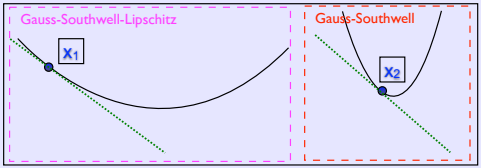
$$f(w^{k+1}) \leq f(w^k) - \frac{1}{2L_{j_k}}|\nabla_{j_k} f(w^k)|^2.$$

- The best coordinate to update according to this bound is

$$j_k \in \underset{j}{\text{argmax}} \frac{|\nabla_j f(w^k)|^2}{L_j}$$

which is called the Gauss-Southwell-Lipschitz rule.

  - "If gradients are similar, pick the one that changes more slowly".



- This is the optimal update for quadratic functions.

Problems Suitable for Coordinate Optimization

- We now know that many problems satisfy the "d-times faster" condition.
- For example, consider composition of a smooth function with affine map,

$$F(w) = f(Aw),$$

for a matrix A and a smooth function g with cost of $O(n)$.

(includes least squares and logistic regression)

- Using f' as the gradient of f, the partial derivatives have the form

$$\nabla_j F(x) = a_j^\top f'(Aw).$$

- If we have Aw, this costs $O(n)$ instead of $O(nd)$ for the full gradient.
- We can track the product Aw^k as we go with $O(n)$ cost,

$$Aw^{k+1} = A(w^k + \gamma_k e_{j_k}) = \underbrace{Aw^k}_{\text{old value}} + \gamma_k \underbrace{Ae_{j_k}}_{O(n)},$$

Coordinate Optimization for Non-Smooth Objectives

- We can apply coordinate optimization for problems of the form

$$F(x) = \underbrace{f(x)}_{\text{smooth}} + \underbrace{\sum_{j=1}^{d} f_j(x_j)}_{\text{separable}},$$

where the f_j can be non-smooth.

  - This includes enforcing non-negative constraints, or using L1-regularization.

- For proximal-PL F, with coordinate-wise proximal-gradient steps we have

$$\mathbb{E}[f(w^k)] - f^* \leq \left(1 - \frac{\mu}{dL}\right)^k [f(w^0) - f^*],$$

the same convergence linear rate as if the non-smooth f_j were not there.

(and faster than the sublinear $O(1/k)$ rate for subgradient methods)

Block Coordinate Descent

- We can't apply coordinate optimization for group L1-regularization.
  - Non-smooth term is non-separable, so coordinate optimization can get stuck.
- Block coordinate optimization and block coordinate descent:
  - Update groups of variables on each iteration.
- If you choose the "blocks" to be the "groups", you can apply to group L1-regularization.
- Many problems have this "block" structure.
  - You might also use blocks to apply Newton's method to the blocks.
  - This is efficient if the block size isn't too big.