First-Order Optimization Algorithms for Machine Learning

Randomized Algorithms

Mark Schmidt

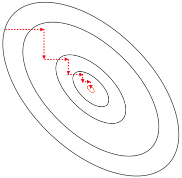
University of British Columbia

Summer 2020

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,

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

  under random choice of j_k.

Analyzing Coordinate Descent

- To analyze coordinate descent, we can write it as

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

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,

$$\mathbb{E}[f(w^{k+1})] \leq \mathbb{E}\left[f(w^k) - \frac{1}{2L}|\nabla_{j_k} f(w^k)|^2\right] \quad \text{(expectation wrt } j_k \text{ given } w^k\text{)}$$

$$= \mathbb{E}[f(w^k)] - \frac{1}{2L}\mathbb{E}[|\nabla_{j_k} f(w^k)|^2] \quad \text{(linearity of expectation)}$$

$$= \underbrace{f(w^k)}_{\text{no } j_k} - \frac{1}{2L}\sum_{j=1}^{d} p(j_k = j)|\nabla_j f(w^k)|^2 \quad \text{(definition of expectation)}$$

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.
  - Let L_c be the maximum gradient changes if you change *one* coordinate.
  - Let L_f be maximum gradient changes if you change *all* coordinates.
  - Gradient descent uses L_f and coordinate optimization uses L_c.

- 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:

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

  - For L2-regularized least squares we would have $L_j = \|x_j\|^2 + \lambda$.
- 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 $\bar{L}$ is 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

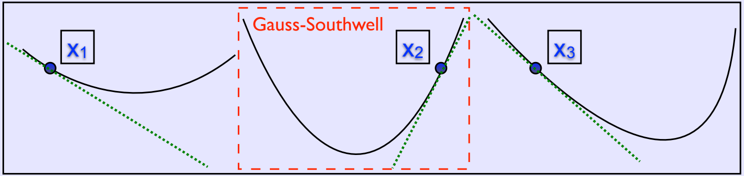
- Our bound on the progress if we choose coordinate j_k is

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

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.

- This can be viewed as "steepest descent in the L1-norm",

$$w^{k+1} \in \underset{v \in \mathbb{R}^d}{\text{argmin}} \left\{ f(w^k) + \nabla f(w^k)^\top (v - w^k) + \frac{L}{2}\|v - w^k\|_1^2 \right\}.$$

Greedy Gauss-Southwell Selection Rule

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

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

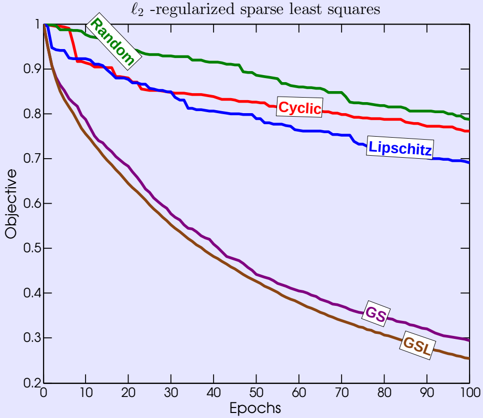
  and the "best" j_k according to the bound is

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

- 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.

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 random coordinate-wise proximal-gradient 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)
- There are 4 different "greedy" rules in this setting (GS-s, GS-r, GS-q, GS-1).

Coordinate Optimization for "Composition with Linear"

- 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).
- Recent works: coordinate optimization leads to faster PageRank methods.

Block Coordinate Descent

- Instead of updating 1 variable, block coordinate descent updates a "block".
- Examples where you might want to do this:
  - Coordinate descent steps converge too slow or don't fully-utilize parallel resources:
    - Better to do a Newton step on 50 variables on each iteration?
  - In multi-class logistic regression,

$$f(W) = \sum_{i=1}^{n} \left[-w_{y^i}^\top x^i + \log\left(\sum_c \exp\left(w_c^\top x^i\right)\right)\right],$$

    the cost of computing/updating 1 partial derivative w_c^j is the same as for w_c.
    - So you could update an entire vector for cost of updating 1 parameter.
  - In group L1-regularization,

$$F(w) = f(w) + \lambda \sum_{g \in \mathcal{G}} \|w_g\|,$$

    since the non-smooth part is only "group separable".
    - Coordinate descent will get stuck, block coordinate descent on groups works.

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

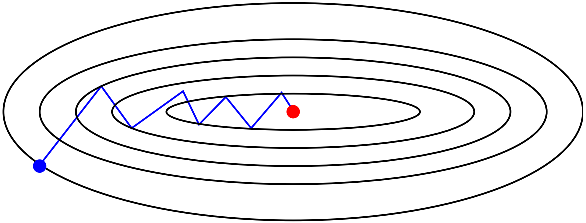
- Gradient methods are effective when d is very large.
- What if number of training examples n is very large?
  - E.g., ImageNet has ≈ 14 million annotated images.

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

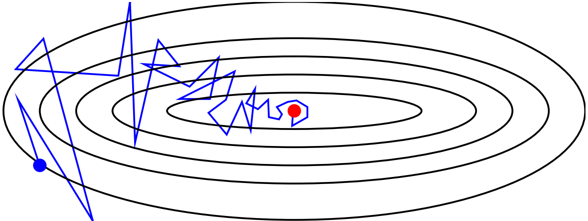
- Stochastic gradient method [Robbins & Monro, 1951]:
  - Random selection of i_k from $\{1, 2, \ldots, n\}$.

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

  - 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:

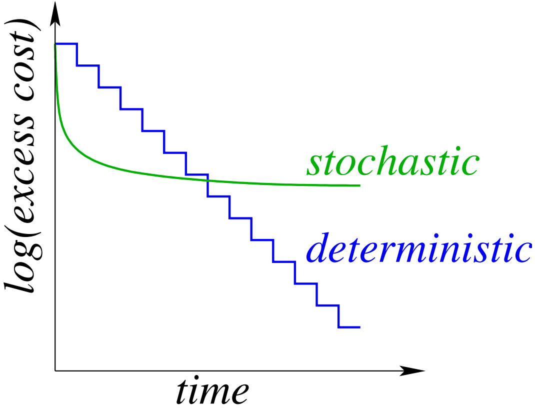
Assumption	Deterministic	Stochastic
Convex	$O(1/\sqrt{\epsilon})$	$O(1/\epsilon^2)$
Strongly	$O(\log(1/\epsilon))$	$O(1/\epsilon)$

- 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 (ϵ's bottleneck is variance, not condition number).

Stochastic vs. Deterministic Convergence Rates

Plot of convergence rates in strongly-convex case:



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

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, and in some cases we may want to update "blocks".
- Stochastic gradient method: n-times cheaper than gradient descent.
  - But much slower convergence rate for smooth functions.

- Next time: SGD theory and practice.

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,

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

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

$$\leq \left(1 - \frac{\mu}{dL}\right)^2[f(w^{k-1}) - f^*]$$

- 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

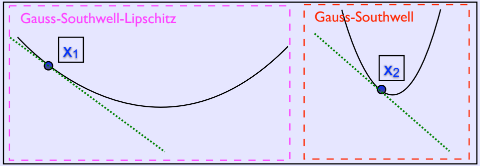
- Our bound on the progress with an L_j for each coordinate is

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

- 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.
(Assuming f' costs $O(n)$)
- We can track the product Aw^k as we go with $O(n)$ cost,
$$Aw^{k+1} = A(w^k + \gamma_k e_{j_k}) = \underbrace{Aw^k}_{\text{old value}} + \gamma_k \underbrace{Ae_{j_k}}_{O(n)},$$