CPSC 540: Machine Learning

Structured Sparsity, Stochatic Subgradient

Mark Schmidt

University of British Columbia

Winter 2017

Admin

- Assignment 1:
  - 2 late days to hand it in today.
- Assignment 2:
  - Due February 6.

Last Time: Group L1-Regularization

- Last time we discussed group L1-regularization:

$$\operatorname*{argmin}_{x \in \mathbb{R}^d} f(x) + \lambda \sum_{g \in G} \|x_g\|_2.$$

- Encourages sparsity in terms of groups g.
  - For example, if $G = \{\{1, 2\}, \{3, 4\}\}$ then we have:

$$\sum_{g \in G} \|x_g\|_2 = \sqrt{x_1^2 + x_2^2} + \sqrt{x_3^2 + x_4^2}.$$

Variables x_1 and x_2 will either be both zero or both non-zero.
Variables x_3 and x_4 will either be both zero or both non-zero.

- Relevant for feature selection when each feature affects multiple parameters.

Last Time: Projected-Gradient

- We discussed minimizing smooth functions with simple constraints,

$$\operatorname*{argmin}_{x \in \mathcal{C}} f(x).$$

- With simple constraints, we can use projected-gradient:

$$x^{t+\frac{1}{2}} = x^t - \alpha_t \nabla f(x^t) \qquad \text{(gradient step)}$$

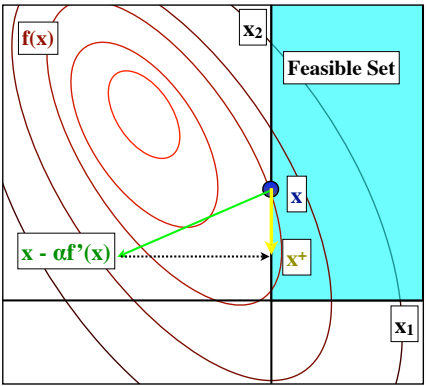
$$x^{t+1} = \operatorname*{argmin}_{y \in \mathcal{C}} \|y - x^{t+\frac{1}{2}}\| \qquad \text{(projection)}$$

- Examples of simple sets include:
  - Upper and lower bounds.
  - Small number of linear equalities or inequalities.
  - Discrete probability distributions.
  - Norm-balls or norm-cones for the standard norms.

Last Time: Projected-Gradient

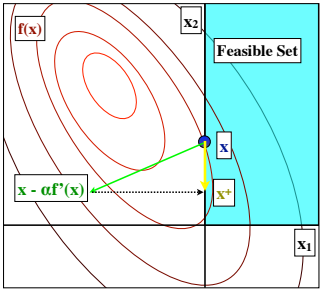
$$x^{t+\frac{1}{2}} = x^t - \alpha_t \nabla f(x^t) \qquad \text{(gradient step)}$$

$$x^{t+1} = \underset{y \in \mathcal{C}}{\text{argmin}} \, \|y - x^{t+\frac{1}{2}}\| \qquad \text{(projection)}$$



Line-Search for Projected Gradient

- There are two ways to do line-search for this algorithm:
  - Backtrack along the line between x^+ and x (search interior).
    - "Backtracking along the feasible direction", costs 1 projection per iteration.



  - Backtrack by decreasing α and re-projecting (search boundary).
    - "Backtracking along the projection arc", costs 1 projection per backtrack.

Last Time: Projected-Newton

- We discussed how the naive projected-Newton method,

$$x^{t+\frac{1}{2}} = x^t - \alpha_t [H_t]^{-1} \nabla f(x^t) \qquad \text{(Newton-like step)}$$
$$x^{t+1} = \underset{y \in \mathcal{C}}{\operatorname{argmin}} \|y - x^{t+\frac{1}{2}}\| \qquad \text{(projection)}$$

  will not work.

- The correct projected-Newton method uses

$$x^{t+\frac{1}{2}} = x^t - \alpha_t [H_t]^{-1} \nabla f(x^t) \qquad \text{(Newton-like step)}$$
$$x^{t+1} = \underset{y \in \mathcal{C}}{\operatorname{argmin}} \|y - x^{t+\frac{1}{2}}\|_{H_t} \qquad \text{(projection under Hessian metric)}$$

- This is expensive even if $\mathcal{C}$ is simple.
  - Practical methods use diagonal H^t, two-metric projection, and inexact projection.

Last Time: Proximal-Gradient

- We discussed proximal-gradient methods for problems of the form

$$\underset{w\in\mathbb{R}^d}{\text{argmin}} \underbrace{g(w)}_{\text{smooth}} + \underbrace{r(w)}_{\text{simple}}.$$

- These methods use the iteration

$$x^{t+\frac{1}{2}} = x^t - \alpha_t \nabla f(x^t) \qquad \text{(gradient step)}$$

$$x^{t+1} = \underset{y\in\mathbb{R}^d}{\text{argmin}} \left\{ \frac{1}{2}\|y - x^{t+\frac{1}{2}}\|^2 + \alpha_t r(y) \right\} \qquad \text{(proximal step)}$$

- Examples of simple functions include:
  - L1-regularization.
  - Group L1-regularization.

Proximal operators for these cases are soft-thresholds: sets variables/groups to 0.

Proximal-Newton

- We can define proximal-Newton methods using

$$x^{t+\frac{1}{2}} = x^t - \alpha_t [H_t]^{-1} \nabla f(x^t) \qquad \text{(gradient step)}$$

$$x^{t+1} = \underset{y \in \mathbb{R}^d}{\text{argmin}} \left\{ \frac{1}{2} \|y - x^{t+\frac{1}{2}}\|_{H_t}^2 + \alpha_t r(y) \right\} \qquad \text{(proximal step)}$$

- This is expensive even for simple r like L1-regularization.
- But there are analogous tricks to projected-Newton methods:
  - Diagonal or Barzilai-Borwein Hessian approximation.
  - "Orthant-wise" methods are analogues of two-metric projection.
  - Inexact methods use approximate proximal operator.

Properties of Proximal-Gradient

- Two convenient properties of proximal-gradient:
  - Proximal operators are non-expansive,

$$\|\text{prox}_r(x) - \text{prox}_r(y)\| \leq \|x - y\|,$$

    it only moves points closer together.

    (including x^k and x^*)

  - For f, only fixed points are global optima,

$$x^* = \text{prox}_r(x^* - \alpha \nabla f(x^*)),$$

    for any $\alpha > 0$.

    (can test $\|x^t - \text{prox}_r(x^t - \nabla f(x^t))\|$ for convergence)

- Proximal gradient/Newton has two line-searches (generalized projected variants):
  - Fix α_t and search along direction to x^{t+1} (1 proximal operator, non-sparse iterates).
  - Vary α_t values (multiple proximal operators per iteration, gives sparse iterations).

Proximal-Gradient Line-Search and Convergence Rate

- Simplest linear convergence proofs are based on the proximal-PL inequality,

$$\frac{1}{2}\mathcal{D}_r(x, L) \geq \mu(F(x) - F^*),$$

where compared to PL inequality we've replaced $\|\nabla f(x)\|^2$ with

$$\mathcal{D}_r(x, \alpha) = -2\alpha \min_y \left[\nabla g(x)^T(y - x) + \frac{\alpha}{2}\|y - x\|^2 + r(y) - r(x)\right],$$

and recall that $F(x) = g(x) + r(x)$ (proof under proximal-PL in bonus slide).

- This non-intuitive property holds for many important problems:
  - g strongly-convex, $g + r$ satisfy PL, L1-regularized least squares, dual SVM problem.
- Can also be used to analyze of coordinate optimization for non-smooth h_j.

- But it's painful to show that functions satisfy this property.

Outline

Structured Sparsity

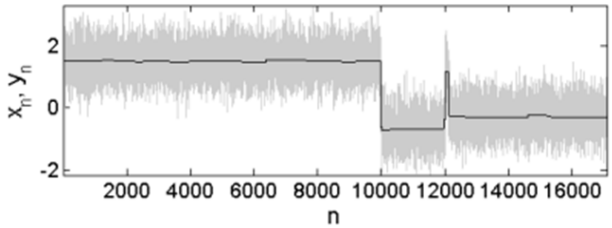
- There are many other patterns that regularization can encourage.
  - We call this structured sparsity.
- The three most common cases:
  - Total-variation regularization encourages slow/sparse changes in w.
  - Nuclear-norm regularization encourages sparsity in rank of matrices.
  - Overlapping group L1-regularization encourages sparsity in variable patterns.

Total-Variation Regularization

- 1D total-variation regularization ("fused LASSO") takes the form

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} g(w) + \lambda \sum_{j=1}^{d-1} |w_j - w_{j+1}|.$$

- Encourages consecutive parameters to have same value.
- Often used for time-series data.



http://statweb.stanford.edu/~bjk/regreg/examples/fusedlassoapprox.html

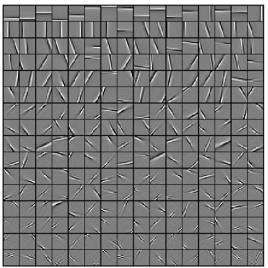
Here x^i is the time and y^i is noisy signal value, while w_i is mean at time i.

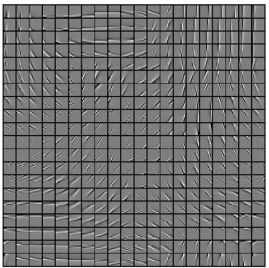
Total-Variation Regularization

- We can also define a 2D version when we have matrix parameters,

$$\underset{W \in \mathbb{R}^{d \times k}}{\operatorname{argmin}} \; g(W) + \lambda \sum_{i=1}^{d-1} \sum_{j=1}^{k-1} |w_{ij} - w_{i+1,j+1}|,$$

  and this is popular for image denoising.
- We could penalize differences on general graph between variables.
- Comparison of latent-factors discovered with/without TV regularization:





Nuclear Norm Regularization

- With matrix parameters an alternative is nuclear norm regularization,

$$\underset{W \in \mathbb{R}^{d \times k}}{\text{argmin}} \; g(W) + \lambda \|W\|_*,$$

  where $\|W\|_*$ is the sum of singular values.

- It's "L1-regularization of the singular values":
  - Encourages parameter matrix to have low-rank: can write $W = UV^T$.

- Consider a multi-class logistic regression with a huge number of features/labels,

$$W = \begin{bmatrix} | & | & & | \\ w_1 & w_2 & \cdots & w_k \\ | & | & & | \end{bmatrix} = UV^T, \quad \text{with} \quad U = \begin{bmatrix} | & | \\ u_1 & u_2 \\ | & | \end{bmatrix}, V = \begin{bmatrix} | & | \\ v_1 & v_2 \\ | & | \end{bmatrix},$$

  U and V can be much smaller, and $XW = (XU)V^T$ can be computed faster:
  - $O(ndr + nrk)$ for rank r instad of $O(ndk)$, which is faster if $r < d$ and $r < k$.

Overlapping Group L1-Regularization

- Overlapping group L1-regularization is exactly what it sounds like,

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} g(w) + \sum_{g \in \mathcal{G}} \lambda_g \|w_g\|_p,$$

  where now the groups g can overlap.

- Why is this interesting?
  - Consider the case of two groups, $\{1\}$ and $\{1, 2\}$,

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} g(w) + \lambda_1 |w_1| + \lambda_2 \sqrt{w_1^2 + w_2^2}.$$

  - The third term encourages both w_1 and w_2 to be zero.
  - But if $w_2 \neq 0$, we still pay a λ_1 penalty for making w_1 non-zero.
  - But if $w_1 \neq 0$, the third term is smooth and doesn't encourage w_2 to be zero.

  - So there are only 3 possible non-zero patterns: $\{\}$, $\{w_2\}$, $\{w_1, w_2\}$.
    - We've won't have $w_1 \neq 0$ and $w_2 = 0$.

Overlapping Group L1-Regularization

- Consider a problem with matrix parameters W.
- We want W to be "band-limited":
  - Non-zeroes only on the main diagonals.

- We can enforce this with overlapping group L1-regularization:
  - Only allow non-zeroes on ± 1 diagonal if you are non-zero on main diagonal.
  - Only allow non-zeroes on ± 2 diagonal if you are non-zero on ± 1 diagonal.
  - Only allow non-zeroes on ± 3 diagonal if you are non-zero on ± 2 diagonal.

Overlapping Group L1-Regularization

- Consider a linear model with higher-order terms,

$$\hat{y}^i = w_0 + w_1\hat{x}_1^i + w_2\hat{x}_2^i + w_3\hat{x}_3^i + w_{12}\hat{x}_1^i\hat{x}_2^i + w_{13}\hat{x}_1^i\hat{x}_3^i + w_{23}\hat{x}_2^i\hat{x}_3^i + w_{123}\hat{x}_1^i\hat{x}_2^i\hat{x}_3^i.$$

  - If d is non-trivial, then the number of higher-order terms is too large.

- We can use overlapping group L1-regularization to enforce a hierarchy.
  - We only allow $w_{12} \neq 0$ if $w_1 \neq 0$ and $w_2 \neq 0$.
    - Enforce this using the groups $\{\{w_1, w_{12}\}, \{w_2, w_{12}\}, \{w_{12}\}\}$.
  - We only allow $w_{123} \neq 0$ if $w_{12} \neq 0$, $w_{13} \neq 0$, and $w_{23} \neq 0$.

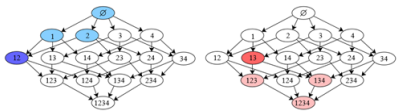


Fig 9: Power set of the set $\{1, \ldots, 4\}$: in blue, an authorized set of selected subsets. In red, an example of a group used within the norm (a subset and all of its descendants in the DAG).

http://arxiv.org/pdf/1109.2397v2.pdf

  - For certain bases, you can work with the full hierarchy in polynomial time.

Overlapping Group L1-Regularization

- Overlapping group-L1 can encourage any intersection-closed sparsity pattern.
  - Set formed from taking $\cap_{g \in \mathcal{G}'} g$ for any $\mathcal{G}' \subset \mathcal{G}$.
- Example is enforcing convex non-zero patterns:

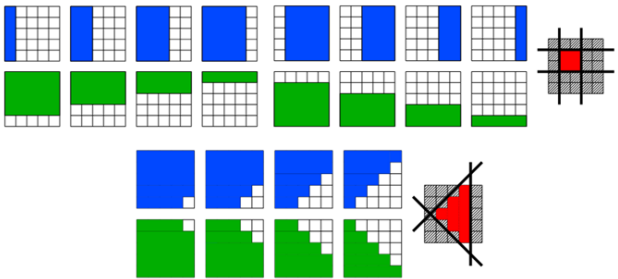
Fig 3: (Left) The set of blue groups to penalize in order to select contiguous patterns in a sequence. (Right) In red, an example of such a nonzero pattern with its corresponding zero pattern (hatched area).

https://arxiv.org/pdf/1109.2397v2.pdf

- There is also a variant ("over-LASSO") that considers unions of groups.

Overlapping Group L1-Regularization

- Overlapping group-L1 can encourage any intersection-closed sparsity pattern.
  - Set formed from taking $\cap_{g \in \mathcal{G}'} g$ for any $\mathcal{G}' \subset \mathcal{G}$.
- Example is enforcing convex non-zero patterns:



https://arxiv.org/pdf/1109.2397v2.pdf

- There is also a variant ("over-LASSO") that considers unions of groups.

Overlapping Group L1-Regularization

- Overlapping group-L1 can encourage any intersection-closed sparsity pattern.
  - Set formed from taking $\cap_{g \in \mathcal{G}'} g$ for any $\mathcal{G}' \subset \mathcal{G}$.
- Example is enforcing convex non-zero patterns:

https://arxiv.org/pdf/1109.2397v2.pdf

- There is also a variant ("over-LASSO") that considers unions of groups.

Structured Sparsity

- There are many other patterns that regularization can encourage.
  - We call this structured sparsity.
- The three most common cases:
  - Total-variation regularization encourages slow/sparse changes in w.
  - Nuclear-norm regularization encourages sparsity in rank of matrices.
  - Overlapping group L1-regularization encourages sparsity in variable patterns.
- Unfortunately, these regularizers are not "simple".

Inexact Proximal-Gradient Methods

- We can efficiently approximate the proximal operator for:
  - Total-variation regularization.
  - Nuclear-norm regularization.
  - Overlapping group L1-regularization.

- For total-variation and overlapping group-L1, we use Dykstra's algorithm
  - Iterative method that computes proximal operator for sum of "simple" functions.

- For nuclear-norm regularization, many method approximate top singular vectors.

- Inexact proximal-gradient methods:
  - Use an approximation to the proximal operator.
  - If approximation error decreases fast enough, same convergence rate:
    - To get $O(\rho^t)$ rate, error must be in $o(\rho^t)$.

Alternating Direction Method of Multipliers

- For total-variation and overlapping group-L1, ADMM is also popular.

- Alternating direction method of multipliers (ADMM) solves:

$$\min_{Ax+By=c} f(x) + r(y).$$

- Alternates between prox-like operators with respect to f and r.
- Can introduce constraints to convert to this form:

$$\min_w \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_1 \quad \Leftrightarrow \quad \min_{v=Xw} \frac{1}{2}\|v - y\|^2 + \lambda\|w\|_1.$$

$$\min_x f(x) + \|Ax\|_1 \quad \Leftrightarrow \quad \min_{v=Ax} f(x) + \|v\|_1.$$

- If prox can not be computed exactly: linearized ADMM.
  - But ADMM rate depends on tuning parameter(s) and iterations aren't sparse.

Frank-Wolfe Method

- In some cases the projected gradient step

$$x^{t+1} = \underset{y \in \mathcal{C}}{\operatorname{argmin}} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{1}{2\alpha_t} \|y - x^t\|^2 \right\},$$

  may be hard to compute.

- Frank-Wolfe step is sometimes cheaper:

$$\bar{x^t} = \underset{y \in \mathcal{C}}{\operatorname{argmin}} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) \right\},$$

  requires compact $\mathcal{C}$, algorithm takes convex combination of x^t and $\bar{x}^t$.

https://www.youtube.com/watch?v=24e08AX9Eww

- $O(1/t)$ rate for smooth convex objectives, some linear convergence results for smooth and strongly-convex.

UV^T Parameterization for Matrix Problems

- Nuclear norm regularization problems,

$$\underset{W \in \mathbb{R}^{d \times k}}{\text{argmin}} \; f(W) + \lambda \|W\|_*,$$

  have solution that with low rank representation $W = UV^T$.
- But standard algorithms are too costly in many applications.
  - Sometimes we can't even store W.

- Many recent approaches directly minimize under UV^T parameterization,

$$\underset{U \in \mathbb{R}^{d \times R}, V \in \mathbb{R}^{k \times R}}{\text{argmin}} \; f(UV^T) + \lambda_U \|U\|_F^2 + \lambda_V \|V\|_F^2,$$

  and just regularize U and V (here we're using the Frobenius matrix norm).

UV^T Parameterization for Matrix Problems

- Many recent approaches directly minimize under UV^T parameterization,

$$\underset{U \in \mathbb{R}^{d \times R}, V \in \mathbb{R}^{k \times R}}{\operatorname{argmin}} f(UV^T) + \lambda_U \|U\|_F^2 + \lambda_V \|V\|_F^2,$$

  and just regularize U and V (here we're using the Frobenius matrix norm).

- We used this approach in 340 for latent-factor models,

$$f(W, Z) = \frac{1}{2}\|ZW - X\|_F^2 + \frac{\lambda_1}{2}\|Z\|_F^2 + \frac{\lambda_2}{2}\|W\|_F^2.$$

- We can sometimes prove this non-convex gives global solution.
  - Including PCA.
- In other cases, people are working hard on finding assumptions where this is true.
  - It works well enough in practice that practitioners don't seem to care.

Outline

Big-N Problems

- We can write our standard regularized optimization problem as

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(x) \quad + \quad r(x)$$

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

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

$$x^{t+1} = x^t - \alpha_t \nabla f(x^t) = x^t - \frac{\alpha_t}{n}\sum_{i=1}^{n} \nabla f_i(x^t).$$

  - Iteration cost is linear in n.
  - Convergence with constant α_t or line-search.
- Stochastic gradient method [Robbins & Monro, 1951]:
  - Random selection of i_t from $\{1, 2, \ldots, n\}$.

$$x^{t+1} = x^t - \alpha_t \nabla f_{i_t}(x^t).$$

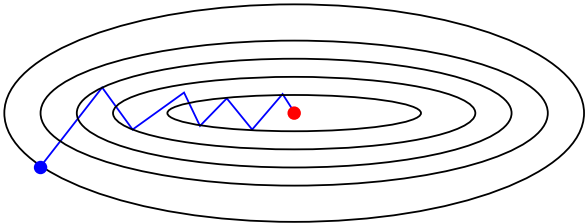
  - Direction is an unbiased estimate of true gradient,

$$\mathbb{E}[f'_{i_t}(x)] = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x) = \nabla f(x).$$

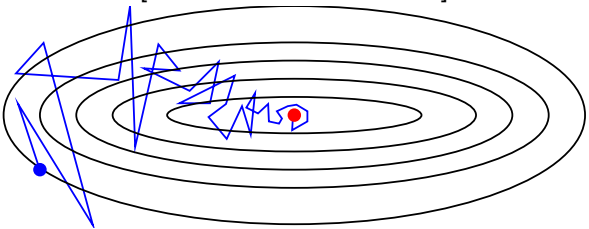
  - Iteration cost is independent of n.
  - Convergence requires $\alpha_t \to 0$.

Stochastic vs. Deterministic Gradient Methods

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



- Stochastic gradient method [Robbins & Monro, 1951]:



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:

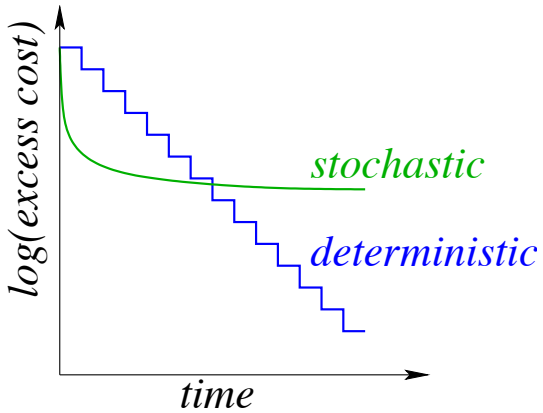
Assumption	Deterministic	Stochastic
Convex	$O(1/\sqrt{\epsilon})$	$O(1/\epsilon^2)$
Strongly	$O(\log(1/\epsilon))$	$O(1/\epsilon)$

- 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_t satisfying $\mathbb{E}[g_t] = \nabla f(x^t)$.

- Nesterov and Newton-like methods do not improve rates in stochastic case.

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:

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

- Rates for subgradient methods for non-smooth objectives:

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

- Other black-box methods (cutting plane, bundle methods) are not faster.
  - In "high-dimensional" setting.

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

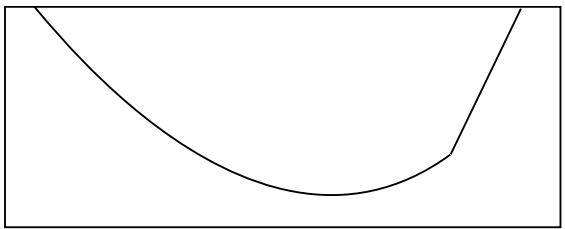
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y.$$

A vector d is a subgradient of a convex function f at x if

$$f(y) \geq f(x) + d^T (y - x), \forall y.$$



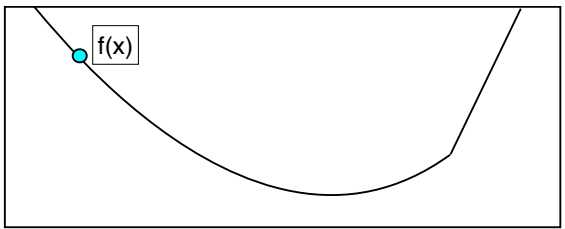
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y.$$

A vector d is a subgradient of a convex function f at x if

$$f(y) \geq f(x) + d^T (y - x), \forall y.$$



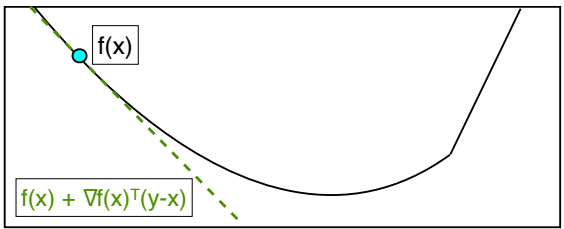
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y.$$

A vector d is a subgradient of a convex function f at x if

$$f(y) \geq f(x) + d^T (y - x), \forall y.$$



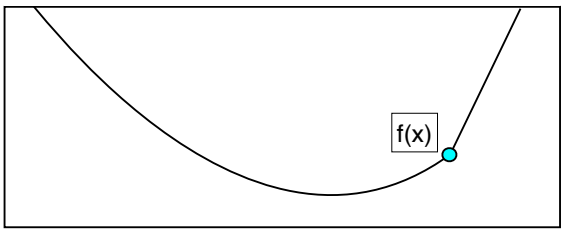
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y.$$

A vector d is a subgradient of a convex function f at x if

$$f(y) \geq f(x) + d^T (y - x), \forall y.$$



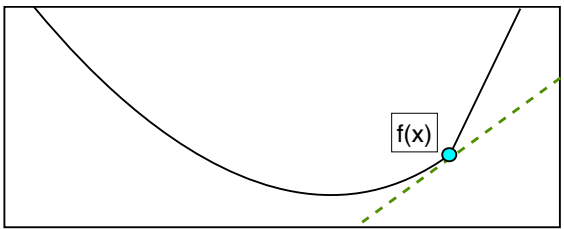
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y.$$

A vector d is a subgradient of a convex function f at x if

$$f(y) \geq f(x) + d^T (y - x), \forall y.$$



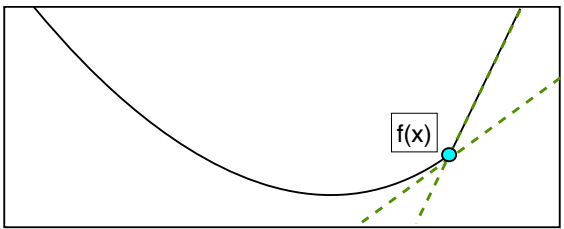
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y.$$

A vector d is a subgradient of a convex function f at x if

$$f(y) \geq f(x) + d^T (y - x), \forall y.$$



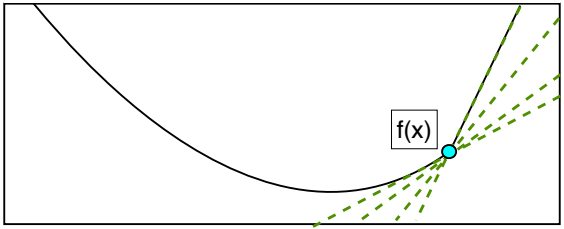
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y.$$

A vector d is a subgradient of a convex function f at x if

$$f(y) \geq f(x) + d^T (y - x), \forall y.$$



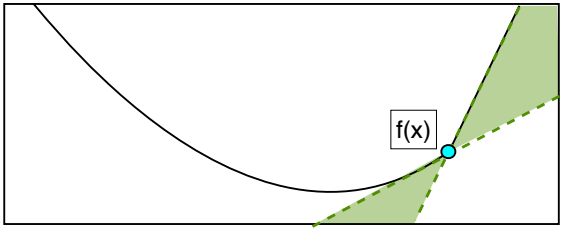
Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y.$$

A vector d is a subgradient of a convex function f at x if

$$f(y) \geq f(x) + d^T (y - x), \forall y.$$



Sub-Gradients and Sub-Differentials

Recall that for *differentiable* convex functions we have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall x, y.$$

A vector d is a subgradient of a convex function f at x if

$$f(y) \geq f(x) + d^T (y - x), \forall y.$$

- At differentiable x:
  - Only subgradient is $\nabla f(x)$.
- At non-differentiable x:
  - We can have a set of subgradients called the sub-differential, $\partial f(x)$.
  - Sub-differential is always non-empty for (almost) all convex functions.
- Note that $0 \in \partial f(x)$ iff x is a global minimum (generalizes $\nabla f(x) = 0$).

Sub-Differential of Absolute Function

- Sub-differential of absolute value function:

$$\partial |x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$

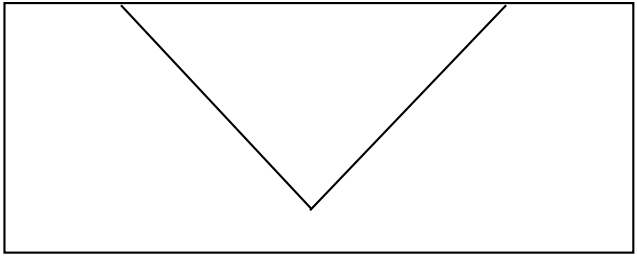
(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)

Sub-Differential of Absolute Function

- Sub-differential of absolute value function:

$$\partial |x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$

(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)

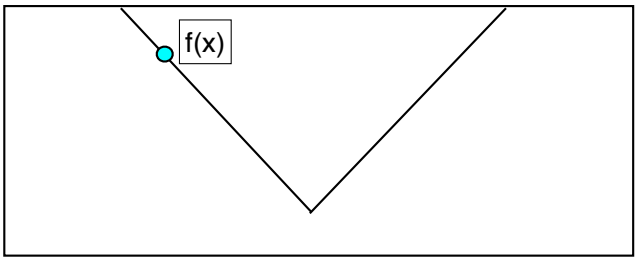


Sub-Differential of Absolute Function

- Sub-differential of absolute value function:

$$\partial |x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$

(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)

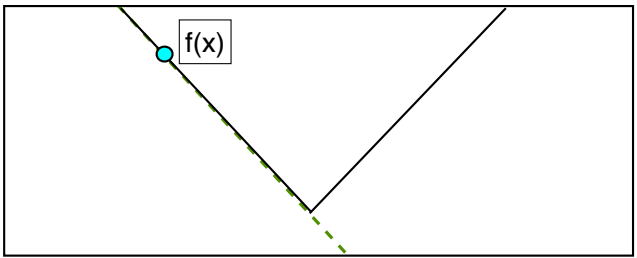


Sub-Differential of Absolute Function

- Sub-differential of absolute value function:

$$\partial |x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$

(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)

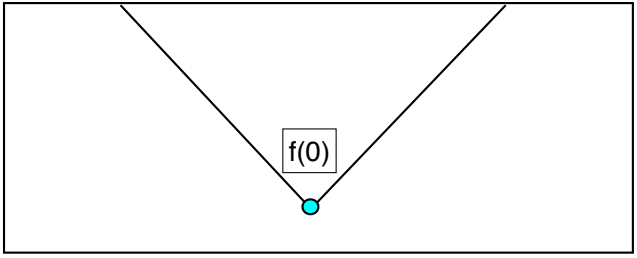


Sub-Differential of Absolute Function

- Sub-differential of absolute value function:

$$\partial |x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$

(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)

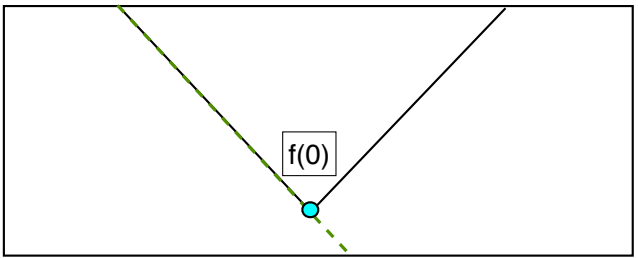


Sub-Differential of Absolute Function

- Sub-differential of absolute value function:

$$\partial |x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$

(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)

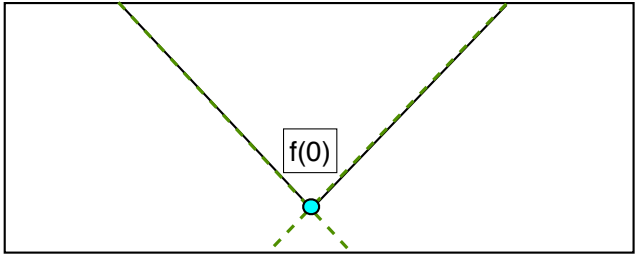


Sub-Differential of Absolute Function

- Sub-differential of absolute value function:

$$\partial |x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$

(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)

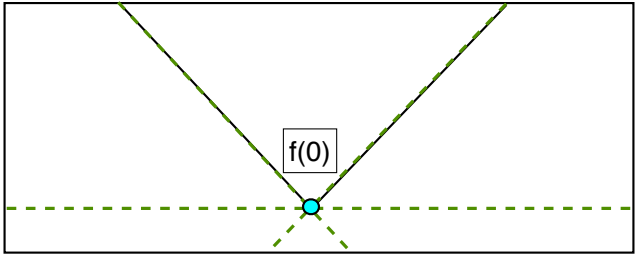


Sub-Differential of Absolute Function

- Sub-differential of absolute value function:

$$\partial |x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$

(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)

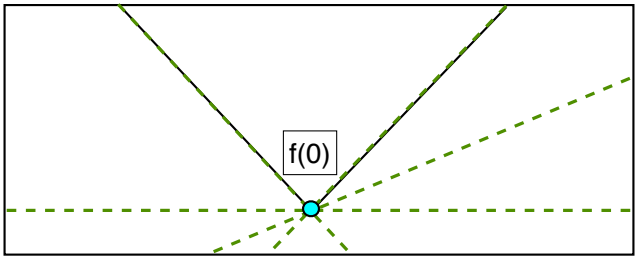


Sub-Differential of Absolute Function

- Sub-differential of absolute value function:

$$\partial |x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$

(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)

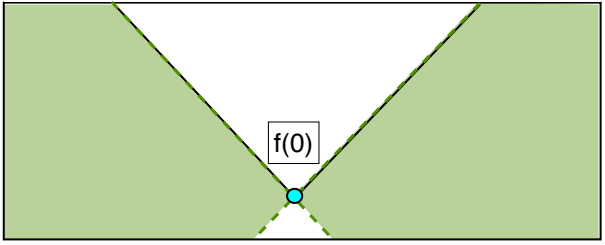


Sub-Differential of Absolute Function

- Sub-differential of absolute value function:

$$\partial |x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$

(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)



Sub-Differential of Common Operations

- Sub-differential of absolute value function:

$$\partial|x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$

(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)

- Sub-differential of sum of convex f_1 and f_2:

$$\partial(f_1(x) + f_2(x)) = d_1 + d_2 \quad \text{for any} \quad d_1 \in \partial f_1(x), d_2 \in \partial f_2(x).$$

- Sub-differential of max of differentiable convex f_1 and f_2:

$$\partial \max\{f_1(x), f_2(x)\} = \begin{cases} \nabla f_1(x) & f_1(x) > f_2(x) \\ \nabla f_2(x) & f_2(x) > f_1(x) \\ \theta \nabla f_1(x) + (1 - \theta)\nabla f_2(x) & f_1(x) = f_2(x) \end{cases}$$

(any "convex combination" of the gradients of the argmax)

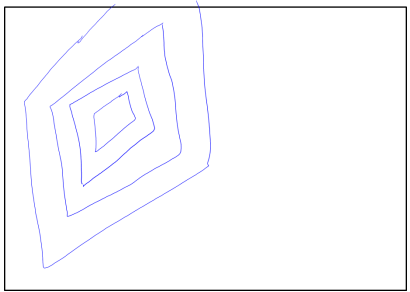
Subgradient Method

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha_t g_t,$$

for some $g_t \in \partial f(x^t)$.

- This can increase the objective even for small α_t.
- But, distance to solution decreases:
  - $\|x^{t+1} - x^*\| < \|x^t - x^*\|$ for small enough α_t.



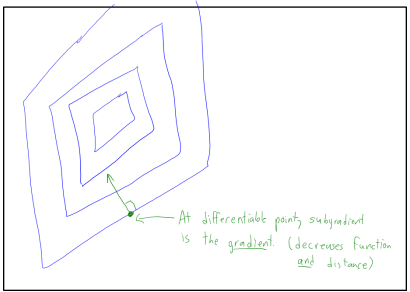
Subgradient Method

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha_t g_t,$$

for some $g_t \in \partial f(x^t)$.

- This can increase the objective even for small α_t.
- But, distance to solution decreases:
  - $\|x^{t+1} - x^*\| < \|x^t - x^*\|$ for small enough α_t.



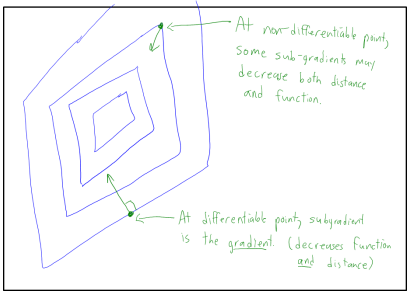
Subgradient Method

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha_t g_t,$$

for some $g_t \in \partial f(x^t)$.

- This can increase the objective even for small α_t.
- But, distance to solution decreases:
  - $\|x^{t+1} - x^*\| < \|x^t - x^*\|$ for small enough α_t.



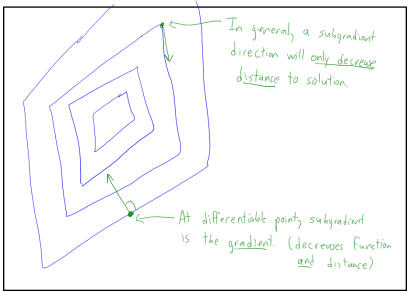
Subgradient Method

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha_t g_t,$$

for some $g_t \in \partial f(x^t)$.

- This can increase the objective even for small α_t.
- But, distance to solution decreases:
  - $\|x^{t+1} - x^*\| < \|x^t - x^*\|$ for small enough α_t.



Subgradient Method

- The basic subgradient method:

$$x^{t+1} = x^t - \alpha_t g_t,$$

  for some $g_t \in \partial f(x^t)$.
- Decreases distance to solution for small enough α_t.

- The basic stochastic subgradient method:

$$x^{t+1} = x^t - \alpha g_{i_t},$$

  for some $g_{i_t} \in \partial f_{i_t}(x^t)$ for some random $i_t \in \{1, 2, \dots, n\}$.

- Stochastic subgradient is n times faster with similar convergence properties.
- Decreases expected distance to solution for small enough α_t.

Outline

Convergence Rate of Stochastic Subgradient Method

- The basic stochastic subgradient method:

$$x^{t+1} = x^t - \alpha g_{i_t},$$

  for some $g_{i_t} \in \partial f_{i_t}(x^t)$ for some random $i_t \in \{1, 2, \ldots, n\}$.
- Since function value may not decrease, we analyze distance to x^*:

$$\begin{aligned}\|x^t - x^*\|^2 &= \|(x^{t-1} - \alpha_t g_{i_t}) - x^*\|^2 \\ &= \|(x^{t-1} - x^*) - \alpha_t g_{i_t}\|^2 \\ &= \|x^{t-1} - x^*\|^2 - 2\alpha_t g_{i_t}^T(x^{t-1} - x^*) + \alpha_t^2 \|g_{i_t}\|^2.\end{aligned}$$

- Take expectation with respect to i_t:

$$\begin{aligned}\mathbb{E}[\|x^t - x^*\|^2] &= \mathbb{E}[\|x^{t-1} - x^*\|] - 2\alpha_t \mathbb{E}[g_{i_t}^T(x^{t-1} - x^*)] + \alpha_t^2 \mathbb{E}[\|g_{i_t}\|^2] \\ &= \underbrace{\|x^{t-1} - x^*\|^2}_{\text{old distance}} - 2\alpha_t \underbrace{g_t^T(x^{t-1} - x^*)}_{\text{expected progress}} + \alpha_t^2 \underbrace{\mathbb{E}[\|g_{i_t}\|^2]}_{\text{“variance”}}.\end{aligned}$$

Convergence Rate of Stochastic Subgradient

- Our expected distance given x^{t-1} is

$$\mathbb{E}[\|x^t - x^*\|^2] = \underbrace{\|x^{t-1} - x^*\|^2}_{\text{old distance}} - 2\alpha_t \underbrace{g_t^T(x^{t-1} - x^*)}_{\text{expected progress}} + \alpha_t^2 \underbrace{\mathbb{E}[\|g_{i_t}\|^2]}_{\text{"variance"}}.$$

- Step-size α_t controls how fast we move towards solution.
- But squared step-size α_t^2 controls how much variance moves us away.

- Standard assumption is that the variance is bounded by constant B^2.
- It follows from strong-convexity that (bonus slide),

$$g_t^T(x^{t-1} - x^*) \geq \mu\|x^{t-1} - x^*\|^2,$$

which gives

$$\begin{aligned}\mathbb{E}[\|x^t - x^*\|^2] &\leq \|x^{t-1} - x^*\|^2 - 2\alpha_t\mu\|x^{t-1} - x^*\|^2 + \alpha_t^2 B^2 \\ &= (1 - 2\alpha_t\mu)\|x^{t-1} - x^*\|^2 + \alpha_t^2 B^2.\end{aligned}$$

Stochastic Gradient with Constant Step Size

- Our bound on expected distance:

$$\mathbb{E}[\|x^t - x^*\|^2] \leq (1 - 2\alpha_t\mu)\|x^{t-1} - x^*\|^2 + \alpha_t^2 B^2.$$

- If α_t is *small* enough, shows distance to solution decreases.

- With constant $\alpha_t = \alpha$ and applying recursively we get

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

  after some of math (last term comes from bounding a geometric series).

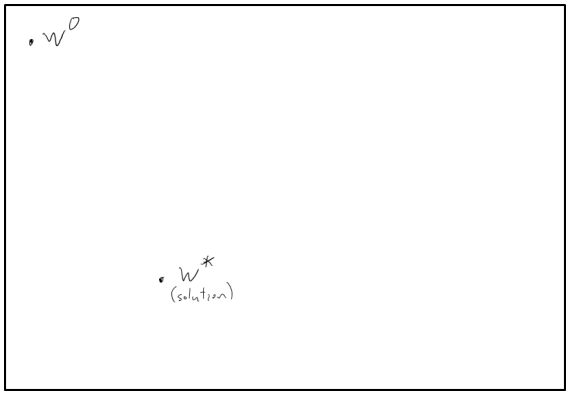
- First term looks like linear convergence, but second term does not go to zero.

Stochastic Gradient with Constant Step Size

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

$$\mathbb{E}[\|x^t - x^*\|^2] \leq (1 - 2\alpha\mu)^t \|x^0 - x^*\|^2 + \frac{\alpha B^2}{2\mu}.$$

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

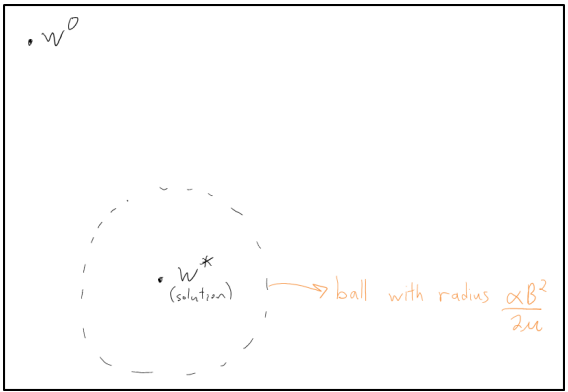


Stochastic Gradient with Constant Step Size

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

$$\mathbb{E}[\|x^t - x^*\|^2] \leq (1 - 2\alpha\mu)^t \|x^0 - x^*\|^2 + \frac{\alpha B^2}{2\mu}.$$

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

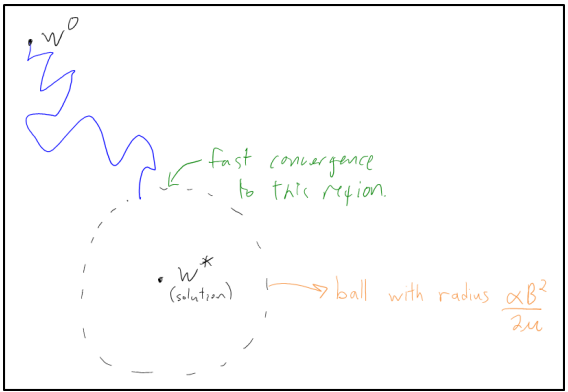


Stochastic Gradient with Constant Step Size

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

$$\mathbb{E}[\|x^t - x^*\|^2] \leq (1 - 2\alpha\mu)^t \|x^0 - x^*\|^2 + \frac{\alpha B^2}{2\mu}.$$

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

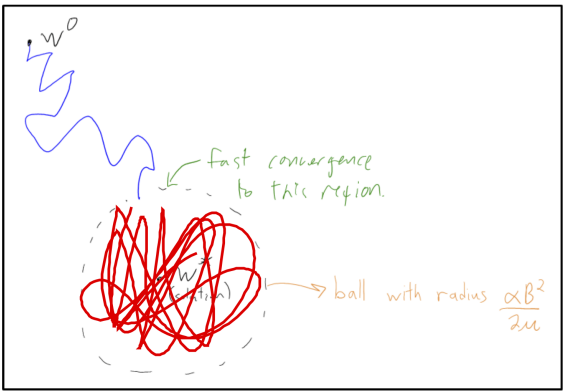


Stochastic Gradient with Constant Step Size

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

$$\mathbb{E}[\|x^t - x^*\|^2] \leq (1 - 2\alpha\mu)^t \|x^0 - x^*\|^2 + \frac{\alpha B^2}{2\mu}.$$

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



Stochastic Gradient with Decreasing Step Size

- To get convergence, we need a decreasing step size.
  - We need effect of variance to go to 0, but we still need to make progress.
  - Classic approach is to choose α_t such that

$$\sum_{t=1}^{\infty} \alpha_t = \infty, \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty,$$

    which suggests setting $\alpha_t = O(1/t)$.
  - We can obtain convergence rates with decreasing steps:
    - If $\alpha_t = \frac{1}{\mu t}$ we can show

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

      for the average iteration $\bar{x}^t = \frac{1}{k}\sum_{k=1}^{T} x_{k-1}$.
    - Note that $O(1/t)$ error implies $O(1/\epsilon)$ iterations required.

Summary

- Structured sparsity encourages more-general patterns in variables.
- Subgradients: generalize gradients for non-smooth convex functions.
- Subgradient method: optimal but very-slow general non-smooth method.
- Stochastic subgadient method: same rate but n times cheaper.
  - Constant step-size: subgradient quickly converges to approximate solution.
  - Decreasing step-size: subgradient slowly converges to exact solution.

- Next time: what if $n = \infty$?

Bonus Slide: Proximal-Gradient Convergence under Proximal-PL

- By Lipschitz continuity of g we have

$$\begin{aligned} F(x_{k+1}) &= g(x_{k+1}) + r(x_k) + r(x_{k+1}) - r(x_k) \\ &\leq F(x_k) + \langle \nabla g(x_k), x_{k+1} - x_k \rangle + \frac{L}{2}||x_{k+1} - x_k||^2 + r(x_{k+1}) - r(x_k) \\ &\leq F(x_k) - \frac{1}{2L}\mathcal{D}_r(x_k, L) \\ &\leq F(x_k) - \frac{\mu}{L}[F(x_k) - F^*], \end{aligned}$$

and then we can take our usual steps.

Bonus Slide: Strong-Convexity Inequalities for Non-Differentiable f

- A "first-order" relationship between subgradient and strong-convexity:
  - If f is μ-strongly convex then for all x and y we have

$$f(y) \geq f(x) + f'(y)^T(y-x) + \frac{\mu}{2}\|y-x\|^2,$$

    for $f'(y) \in \partial f(x)$.
  - The first-order definition of strong-convexity, but with subgradient replacing gradient.
  - Reversing y and x we can write

$$f(x) \geq f(y) + f'(x)^T(x-y) + \frac{\mu}{2}\|x-y\|^2,$$

    for $f'(x) \in \partial f(x)$.
  - Adding the above together gives

$$(f'(y) - f'(x)^)T(y-x)) \geq \mu\|y-x\|^2.$$

  - Applying this with $y = x^{t-1}$ and subgradient g_t and $x = x^*$ (which has $f'(x^*) = 0$ for some subgradient) gives

$$(g_t - 0)^T(x^{t-1} - x^*) \geq \mu\|x^{t-1} - x^*\|^2.$$

Bonus Slide: Faster Rate for Proximal-Gradient

- It's possible to show a slightly faster rate for proximal-gradient using $\alpha_t = 2/(\mu + L)$.
- See `http://www.cs.ubc.ca/~schmidtm/Documents/2014_Notes_ProximalGradient.pdf`