CPSC 540: Machine Learning

Proximal-Gradient and Stochastic Subgradient

Mark Schmidt

University of British Columbia

Winter 2016

Admin

- Room: We'll be in CHEM B150 starting today.
- Assignment 1:
  - You can use 3 of your 3 late days to hand it in before Thursday.
- Assignment 2:
  - Due in one week.
  - Please look at updated version: some typos fixed and Q4.3 removed.
- Switch to Beamer?
  - Microsoft PowerPoint^{TM} patience is reaching 0.
  - I'll post both versions to Piazza for comment.

Last Time: Regularization Paths

- The regularization path is the set of w values as λ varies,

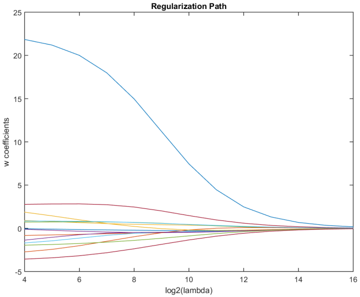
$$w^\lambda = \mathop{\mathrm{argmin}}_{w \in \mathbb{R}^d} f(w) + \lambda r(w),$$

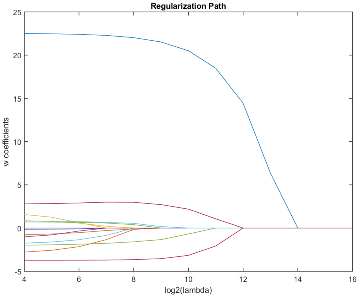
Last Time: Regularization Paths

- The regularization path is the set of w values as λ varies,

$$w^\lambda = \underset{w \in \mathbb{R}^d}{\text{argmin}}\, f(w) + \lambda r(w),$$

- Squared L2-regularization path vs. L1-regularization path:





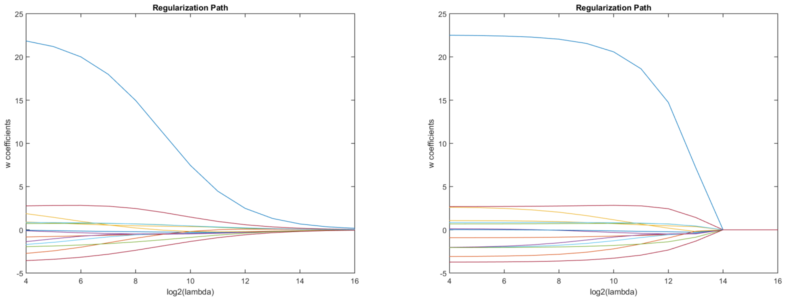
- With $r(w) = \|w\|^2$, w_j get close to 0 but not exactly 0.
- With $r(w) = \|w\|_1$, w_j get set to exactly zero for finite λ.

Last Time: Regularization Paths

- The regularization path is the set of w values as λ varies,

$$w^\lambda = \operatorname*{argmin}_{w \in \mathbb{R}^d} f(w) + \lambda r(w),$$

- Squared L2-regularization path vs. non-squared path:



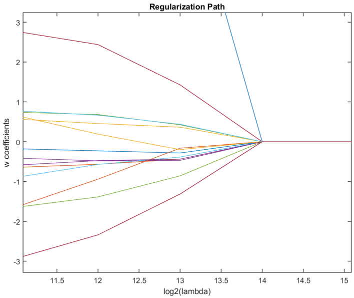
- With $r(w) = \|w\|^2$, w_j get close to 0 but not exactly 0.
- With $r(w) = \|w\|_2$, all w_j get set to exactly zero for finite λ.

Last Time: Regularization Paths

- The regularization path is the set of w values as λ varies,

$$w^\lambda = \mathop{\text{argmin}}_{w \in \mathbb{R}^d} f(w) + \lambda r(w),$$

- Non-squared L2-regularization path:



- You tend to get all or none among regularized variables.

Last Time: Group L1-Regularization

- Last time we discussed group L1-regularization:

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

- 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.

Last Time: Group L1-Regularization

- Last time we discussed group L1-regularization:

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

- 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.
- Why is it called group L1-regularization?
  - If vector v contains the group norms, it's the L1-norm of v:

$$\text{If } v \triangleq \begin{bmatrix} \|x_{12}\|_2 \\ \|x_{34}\|_2 \end{bmatrix} = \begin{bmatrix} \sqrt{x_1^2 + x_2^2} \\ \sqrt{x_3^2 + x_4^2} \end{bmatrix} \text{ then } \sum_{g \in G} \|x_g\|_2 = \|x_{12}\|_2 + \|x_{34}\|_2 = v_1 + v_2 = |v_1| + |v_2| = \|v\|_1.$$

Last Time: Projected-Gradient

- We can convert the non-smooth problem

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

into a smooth problem with simple constraints:

$$\operatorname*{argmin}_{x \in \mathbb{R}^d} f(x) + \lambda \sum_{g \in G} r_g, \text{ subject to } r_g \geq \|x_g\|_2 \text{ for all } g.$$

Last Time: Projected-Gradient

- We can convert the non-smooth problem

$$\mathop{\mathrm{argmin}}_{x\in\mathbb{R}^d} f(x) + \lambda \sum_{g\in G} \|x_g\|_2,$$

into a smooth problem with simple constraints:

$$\mathop{\mathrm{argmin}}_{x\in\mathbb{R}^d} f(x) + \lambda \sum_{g\in G} r_g, \text{ subject to } r_g \geq \|x_g\|_2 \text{ for all } g.$$

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

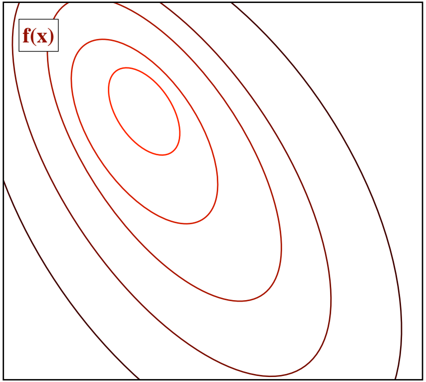
$$x^{t+1} = \mathop{\mathrm{argmin}}_{y\in C} \left\{ f(x^t) + \nabla f(x^t)^T(y - x^t) + \frac{L}{2}\|y - x^t\|^2 \right\},$$

or equivalently projection applied to gradient step:

$$x^{t+1} = \underbrace{\mathop{\mathrm{argmin}}_{y\in C} \left\{ \|y - x_t^{GD}\| \right\}}_{\text{projection of } x_t^{GD} \text{ onto } \mathcal{C}}, \text{ where } x_t^{GD} = \underbrace{x^t - \alpha_t \nabla f(x^t)}_{\text{gradient step}}.$$

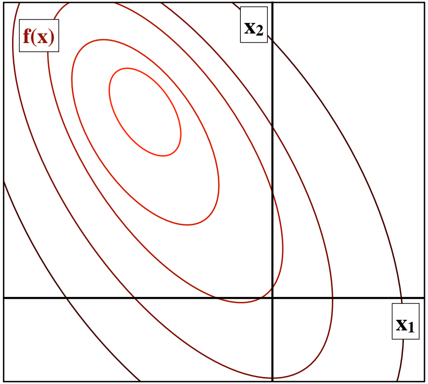
Last Time: Projected-Gradient

$$x^{t+1} = \underbrace{\operatorname*{argmin}_{y \in C} \left\{ \|y - x_t^{GD}\| \right\}}_{\text{projection of } x_t^{GD} \text{ onto } \mathcal{C}}, \text{ where } x_t^{GD} = \underbrace{x^t - \alpha_t \nabla f(x^t)}_{\text{gradient step}}.$$



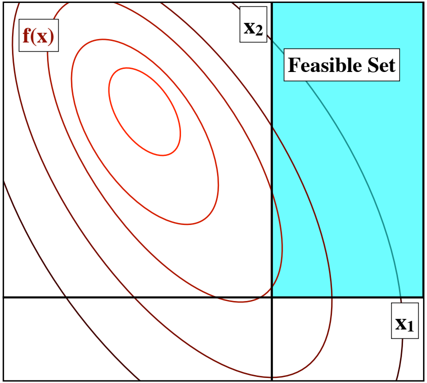
Last Time: Projected-Gradient

$$x^{t+1} = \underbrace{\operatorname*{argmin}_{y \in C} \left\{ \|y - x_t^{GD}\| \right\}}_{\text{projection of } x_t^{GD} \text{ onto } \mathcal{C}}, \text{ where } x_t^{GD} = \underbrace{x^t - \alpha_t \nabla f(x^t)}_{\text{gradient step}}.$$



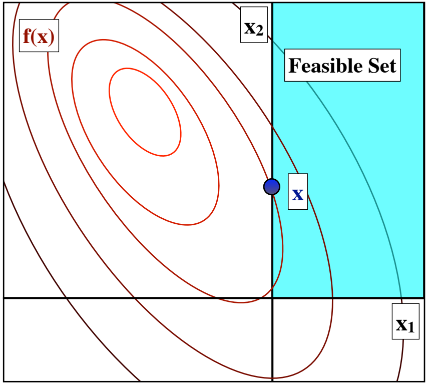
Last Time: Projected-Gradient

$$x^{t+1} = \underbrace{\mathop{\text{argmin}}_{y \in C} \left\{ \|y - x_t^{GD}\| \right\}}_{\text{projection of } x_t^{GD} \text{ onto } \mathcal{C}}, \text{ where } x_t^{GD} = \underbrace{x^t - \alpha_t \nabla f(x^t)}_{\text{gradient step}}.$$



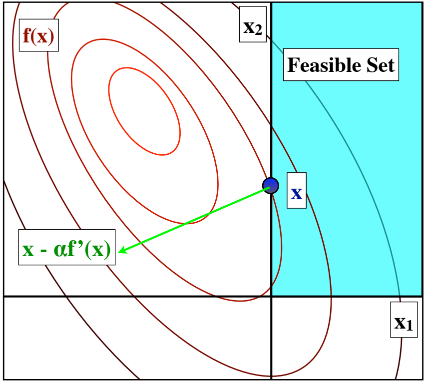
Last Time: Projected-Gradient

$$x^{t+1} = \underbrace{\mathop{\text{argmin}}_{y \in C} \left\{ \|y - x_t^{GD}\| \right\}}_{\text{projection of } x_t^{GD} \text{ onto } \mathcal{C}}, \text{ where } x_t^{GD} = \underbrace{x^t - \alpha_t \nabla f(x^t)}_{\text{gradient step}}.$$



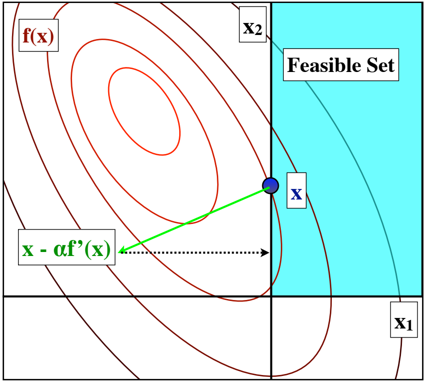
Last Time: Projected-Gradient

$$x^{t+1} = \underbrace{\operatorname*{argmin}_{y \in C} \left\{ \|y - x_t^{GD}\| \right\}}_{\text{projection of } x_t^{GD} \text{ onto } \mathcal{C}}, \quad \text{where } x_t^{GD} = \underbrace{x^t - \alpha_t \nabla f(x^t)}_{\text{gradient step}}.$$



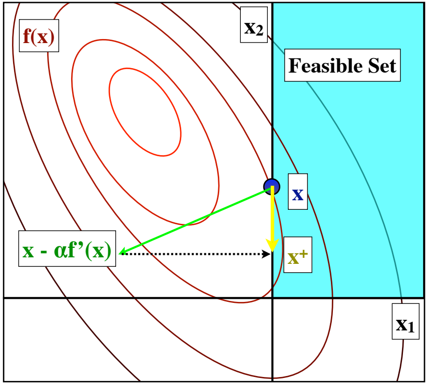
Last Time: Projected-Gradient

$$x^{t+1} = \underbrace{\operatorname*{argmin}_{y \in C} \left\{ \|y - x_t^{GD}\| \right\}}_{\text{projection of } x_t^{GD} \text{ onto } \mathcal{C}}, \text{ where } x_t^{GD} = \underbrace{x^t - \alpha_t \nabla f(x^t)}_{\text{gradient step}}.$$



Last Time: Projected-Gradient

$$x^{t+1} = \underbrace{\operatorname*{argmin}_{y \in C} \{\|y - x_t^{GD}\|\}}_{\text{projection of } x_t^{GD} \text{ onto } \mathcal{C}}, \text{ where } x_t^{GD} = \underbrace{x^t - \alpha_t \nabla f(x^t)}_{\text{gradient step}}.$$



Last Time: Projected-Gradient

- We can convert non-smooth problem into smooth problems with simple constraints:
- But transforming might make problem harder:
  - E.g., transformed problems often lose strong-convexity.
- Can we apply a method like projected-gradient to the original problem?

Gradient Method

- We want to solve a smooth optimization problem:

$$\operatorname*{argmin}_{x \in \mathbb{R}^d} f(x).$$

- Iteration x^t minimizes with quadratic approximation to 'f'':

$$f(y) \approx f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2}\|y - x^t\|^2,$$

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2}\|y - x^t\|^2 \right\}.$$

Gradient Method

- We want to solve a smooth optimization problem:

$$\operatorname*{argmin}_{x \in \mathbb{R}^d} f(x).$$

- Iteration x^t minimizes with quadratic approximation to 'f':

$$f(y) \approx f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2}\|y - x^t\|^2,$$

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2}\|y - x^t\|^2 \right\}.$$

We can equivalently write this as the quadratic optimization:

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathbb{R}^d} \left\{ \frac{1}{2}\|y - (x^t - \alpha_t \nabla f(x^t))\|^2 \right\},$$

and the solution is the gradient algorithm:

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

Proximal-Gradient Method

- We want to solve a smooth plus non-smooth optimization problem:

$$\operatorname*{argmin}_{x\in\mathbb{R}^d} f(x)+r(x).$$

- Iteration x^t minimizes with quadratic approximation to 'f':

$$f(y) \approx f(x^t) + \nabla f(x^t)^T(y - x^t) + \frac{L}{2}\|y - x^t\|^2,$$

$$x^{t+1} = \operatorname*{argmin}_{y\in\mathbb{R}^d}\left\{f(x^t) + \nabla f(x^t)^T(y - x^t) + \frac{L}{2}\|y - x^t\|^2\right\}.$$

We can equivalently write this as the quadratic optimization:

$$x^{t+1} = \operatorname*{argmin}_{y\in\mathbb{R}^d}\left\{\frac{1}{2}\|y - (x^t - \alpha_t\nabla f(x^t))\|^2\right\},$$

and the solution is the gradient algorithm:

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

Proximal-Gradient Method

- We want to solve a smooth plus non-smooth optimization problem:

$$\operatorname*{argmin}_{x \in \mathbb{R}^d} f(x) + r(x).$$

- Iteration x^t minimizes with quadratic approximation to 'f':

$$f(y) + r(y) \approx f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2}\|y - x^t\|^2 + r(y),$$

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)^T (y - x^t) + \frac{L}{2}\|y - x^t\|^2 + r(y) \right\}.$$

We can equivalently write this as the quadratic optimization:

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathbb{R}^d} \left\{ \frac{1}{2}\|y - (x^t - \alpha_t \nabla f(x^t))\|^2 \right\},$$

and the solution is the gradient algorithm:

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

Proximal-Gradient Method

- We want to solve a smooth plus non-smooth optimization problem:

$$\operatorname*{argmin}_{x\in\mathbb{R}^d} f(x)+r(x).$$

- Iteration x^t minimizes with quadratic approximation to 'f':

$$f(y)+r(y) \approx f(x^t) + \nabla f(x^t)^T(y-x^t) + \frac{L}{2}\|y-x^t\|^2+r(y),$$

$$x^{t+1} = \operatorname*{argmin}_{y\in\mathbb{R}^d}\left\{f(x^t) + \nabla f(x^t)^T(y-x^t) + \frac{L}{2}\|y-x^t\|^2+r(y)\right\}.$$

We can equivalently write this as the proximal optimization:

$$x^{t+1} = \operatorname*{argmin}_{y\in\mathbb{R}^d}\left\{\frac{1}{2}\|y-(x^t-\alpha_t\nabla f(x^t))\|^2+\alpha_t r(y)\right\},$$

and the solution is the gradient algorithm:

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

Proximal-Gradient Method

- We want to solve a smooth plus non-smooth optimization problem:

$$\mathop{\mathrm{argmin}}_{x\in\mathbb{R}^d} f(x)+r(x).$$

- Iteration x^t minimizes with quadratic approximation to 'f'':

$$f(y)+r(y) \approx f(x^t) + \nabla f(x^t)^T(y-x^t) + \frac{L}{2}\|y-x^t\|^2+r(y),$$

$$x^{t+1} = \mathop{\mathrm{argmin}}_{y\in\mathbb{R}^d}\left\{f(x^t) + \nabla f(x^t)^T(y-x^t) + \frac{L}{2}\|y-x^t\|^2+r(y)\right\}.$$

We can equivalently write this as the proximal optimization:

$$x^{t+1} = \mathop{\mathrm{argmin}}_{y\in\mathbb{R}^d}\left\{\frac{1}{2}\|y-(x^t-\alpha_t\nabla f(x^t))\|^2+\alpha_t r(y)\right\},$$

and the solution is the proximal-gradient algorithm:

$$x^{t+1} = \mathrm{prox}_{\alpha r}[x^t-\alpha_t\nabla f(x^t)].$$

Proximal-Gradient Method

- So proximal-gradient step takes the form:

$$x_t^{GD} = x^t - \alpha_t \nabla f(x^t),$$
$$x^{t+1} = \operatorname*{argmin}_{y \in \mathbb{R}^d} \left\{ \frac{1}{2}\|y - x_t^{GD}\|^2 + \alpha_t r(y) \right\}.$$

- Second part is called the proximal operator with respect to $\alpha_t r$.
- Convergence rates are still the same as for minimizing f alone:
  - E.g, if ∇f is L-Lipschitz, f is μ-strongly convex.and g is convex, then

$$F(x^t) - F(x^*) \leq \left(1 - \frac{\mu}{L}\right)^t \left[F(x^0) - F(x^*)\right],$$

where $F(x) = f(x) + r(x)$.

Proximal Operator, Iterative Soft Thresholding

- The proximal operator is the solution to

$$\text{prox}_r[x] = \underset{y \in \mathbb{R}^d}{\text{argmin}} \ \frac{1}{2}\|y - x\|^2 + r(y).$$

Proximal Operator, Iterative Soft Thresholding

- The proximal operator is the solution to

$$\text{prox}_r[x] = \underset{y \in \mathbb{R}^d}{\text{argmin}} \; \frac{1}{2}\|y - x\|^2 + r(y).$$

- If $r(y) = \alpha_t \lambda \|y\|_1$, proximal operator is soft-threshold:
  - Apply $x_j = \text{sign}(x_j)\max\{0, |x_j| - \alpha_t\lambda\}$ element-wise.
  - E.g., if $\alpha_t \lambda = 1$:

Input	Threshold	Soft-Threshold
$\begin{bmatrix} 0.6715 \\ -1.2075 \\ 0.7172 \\ 1.6302 \\ 0.4889 \end{bmatrix}$		

Proximal Operator, Iterative Soft Thresholding

- The proximal operator is the solution to

$$\text{prox}_r[x] = \underset{y \in \mathbb{R}^d}{\text{argmin}} \; \frac{1}{2}\|y - x\|^2 + r(y).$$

- If $r(y) = \alpha_t \lambda \|y\|_1$, proximal operator is soft-threshold:
  - Apply $x_j = \text{sign}(x_j) \max\{0, |x_j| - \alpha_t \lambda\}$ element-wise.
  - E.g., if $\alpha_t \lambda = 1$:

Input	Threshold	Soft-Threshold
$\begin{bmatrix} 0.6715 \\ -1.2075 \\ 0.7172 \\ 1.6302 \\ 0.4889 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -1.2075 \\ 0 \\ 1.6302 \\ 0 \end{bmatrix}$	

Proximal Operator, Iterative Soft Thresholding

- The proximal operator is the solution to

$$\text{prox}_r[x] = \underset{y \in \mathbb{R}^d}{\text{argmin}} \ \frac{1}{2}\|y - x\|^2 + r(y).$$

- If $r(y) = \alpha_t \lambda \|y\|_1$, proximal operator is soft-threshold:
  - Apply $x_j = \text{sign}(x_j) \max\{0, |x_j| - \alpha_t \lambda\}$ element-wise.
  - E.g., if $\alpha_t \lambda = 1$:

$$\underset{\text{Input}}{\begin{bmatrix} 0.6715 \\ -1.2075 \\ 0.7172 \\ 1.6302 \\ 0.4889 \end{bmatrix}} \quad \underset{\text{Threshold}}{\begin{bmatrix} 0 \\ -1.2075 \\ 0 \\ 1.6302 \\ 0 \end{bmatrix}} \quad \underset{\text{Soft-Threshold}}{\begin{bmatrix} 0 \\ -0.2075 \\ 0 \\ 0.6302 \\ 0 \end{bmatrix}}$$

Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

$$r(y) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ \infty & \text{if } x \notin \mathcal{C} \end{cases}, \quad (\text{indicator function for convex set } \mathcal{C})$$

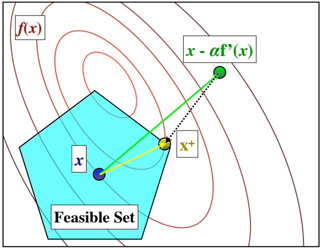
Special case of Projected-Gradient Methods

- Projected-gradient methods are another special case:

$$r(y) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ \infty & \text{if } x \notin \mathcal{C} \end{cases}, \quad (\text{indicator function for convex set } \mathcal{C})$$

gives

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathbb{R}^d} \frac{1}{2}\|y - x\|^2 + r(y) = \operatorname*{argmin}_{y \in \mathcal{C}} \frac{1}{2}\|y - x\|^2 = \operatorname*{argmin}_{y \in \mathcal{C}} \|y - x\|.$$



Proximal-Gradient for Group L1-Regularization

- The proximal operator for L1-regularization,

$$\operatorname*{argmin}_{y \in \mathbb{R}^d} \left\{ \frac{1}{2}\|y - x\|^2 + \alpha_t \lambda \|y\|_1 \right\},$$

applies soft-threshold element-wise,

$$x_j = \frac{x_j}{|x_j|} \max\{0, |x_j| - \alpha_t \lambda\}.$$

Proximal-Gradient for Group L1-Regularization

- The proximal operator for L1-regularization,

$$\underset{y\in\mathbb{R}^d}{\text{argmin}}\left\{\frac{1}{2}\|y-x\|^2+\alpha_t\lambda\|y\|_1\right\},$$

  applies soft-threshold element-wise,

$$x_j=\frac{x_j}{|x_j|}\max\{0,|x_j|-\alpha_t\lambda\}.$$

- The proximal operator for group L1-regularization,

$$\underset{y\in\mathbb{R}^d}{\text{argmin}}\left\{\frac{1}{2}\|y-x\|^2+\alpha_t\lambda\sum_{g\in G}\|y\|_2\right\},$$

  applies a group soft-threshold group-wise,

$$x_g=\frac{x_g}{\|x_g\|_2}\max\{0,\|x_g\|_2-\alpha_t\lambda\}.$$

Exact Proximal-Gradient Methods

- We can efficiently compute the proximity operator for:
  1. L1-Regularization and most separable regularizers.
  2. Group ℓ_1-Regularization (disjoint) and most group-separale regularizers.

Exact Proximal-Gradient Methods

- We can efficiently compute the proximity operator for:
  1. L1-Regularization and most separable regularizers.
  2. Group ℓ_1-Regularization (disjoint) and most group-separale regularizers.
  3. Lower and upper bounds.
  4. Small number of linear constraint.
  5. Probability constraints.
  6. Many norm balls and norm cones.
  7. A few other simple regularizers/constraints.

Exact Proximal-Gradient Methods

- We can efficiently compute the proximity operator for:
  1. L1-Regularization and most separable regularizers.
  2. Group ℓ_1-Regularization (disjoint) and most group-separale regularizers.
  3. Lower and upper bounds.
  4. Small number of linear constraint.
  5. Probability constraints.
  6. Many norm balls and norm cones.
  7. A few other simple regularizers/constraints.
- Can solve these non-smooth problems as fast as smooth problems.
- But what if we can't efficiently compute proximal operator?

Inexact Proximal-Gradient Methods

- We can efficiently approximate the proximal operator for:
  - Overlapping group L1-regularization.
  - Total-variation regularization.
  - Nuclear-norm regularization.
  - Sums of ‘simple’ functions (proximal-Dykstra).

Inexact Proximal-Gradient Methods

- We can efficiently approximate the proximal operator for:
  - Overlapping group L1-regularization.
  - Total-variation regularization.
  - Nuclear-norm regularization.
  - Sums of ‘simple’ functions (proximal-Dykstra).
- 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 $o(\rho^t)$.

Discussion of Proximal-Gradient

- Solution x^* is a fixed-point:

$$x^* = \mathrm{prox}_{\alpha r}[x^* - \alpha f(x^*)], \text{ for any } \alpha.$$

Discussion of Proximal-Gradient

- Solution x^* is a fixed-point:

$$x^* = \text{prox}_{\alpha r}[x^* - \alpha f(x^*)], \text{ for any } \alpha.$$

- With $\alpha_t < 2/L$, guaranteed to decrease objective.
  - Can still use adaptive step-size to estimate 'L'.

Discussion of Proximal-Gradient

- Solution x^* is a fixed-point:

$$x^* = \text{prox}_{\alpha r}[x^* - \alpha f(x^*)], \text{ for any } \alpha.$$

- With $\alpha_t < 2/L$, guaranteed to decrease objective.
  - Can still use adaptive step-size to estimate 'L'.
- With any α_t, proximal–gradient generates a feasible descent direction:
  - If $\bar{x}^t = \text{prox}_{\alpha_t r}[x^t - \alpha_t \nabla f(x^t)]$, then the step

$$x^{t+1} = x^t + \gamma_t(\bar{x}^t - x^t),$$

decreases f and satisfies constraints for γ_t small enough.

Discussion of Proximal-Gradient

- Solution x^* is a fixed-point:

$$x^* = \text{prox}_{\alpha r}[x^* - \alpha f(x^*)], \text{ for any } \alpha.$$

- With $\alpha_t < 2/L$, guaranteed to decrease objective.
  - Can still use adaptive step-size to estimate 'L'.
- With any α_t, proximal–gradient generates a feasible descent direction:
  - If $\bar{x}^t = \text{prox}_{\alpha_t r}[x^t - \alpha_t \nabla f(x^t)]$, then the step

    $$x^{t+1} = x^t + \gamma_t(\bar{x}^t - x^t),$$

    decreases f and satisfies constraints for γ_t small enough.
- If proximal operator is expensive, can do Armijo line-search for γ_t instead of α_t:
  - Fix α_t and decrease γ_t: "backtracking along the feasible direction".
    - Iterations tend to be in interior.
  - Fix γ_t and decrease α_t: "backtracking along the projection arc".
    - Iterations tend to be at boundary.

Faster Proximal-Gradient Methods

- Accelerated proximal-gradient method:

$$x^{t+1} = \text{prox}_{\alpha_t r}[y^t - \alpha_t \nabla f(x^t)]$$
$$y^{t+1} = x^t + \beta_t(x^{t+1} - x^t).$$

  - Convergence properties same as smooth version.

Faster Proximal-Gradient Methods

- Accelerated proximal-gradient method:

$$x^{t+1} = \text{prox}_{\alpha_t r}[y^t - \alpha_t \nabla f(x^t)]$$
$$y^{t+1} = x^t + \beta_t(x^{t+1} - x^t).$$

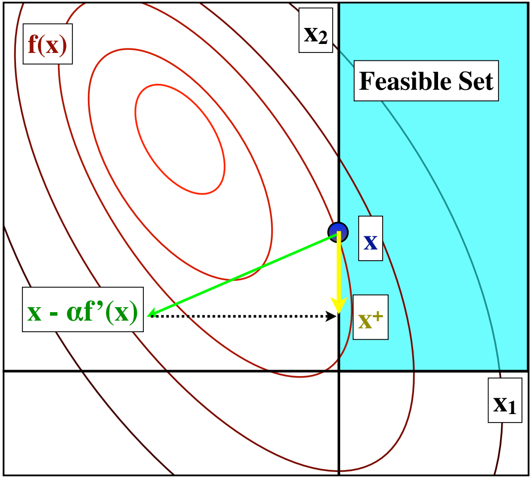
  - Convergence properties same as smooth version.

- The naive Newton-like methods,

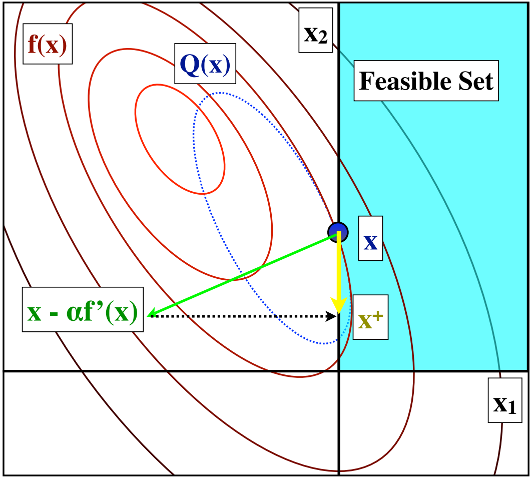
$$x^{t+1} = \text{prox}_{\alpha r}[x^t - \alpha_t d^t], \text{ where } d^t \text{ solves } \nabla^2 f(x^t) d^t = \nabla f(x^t),$$

does NOT work.

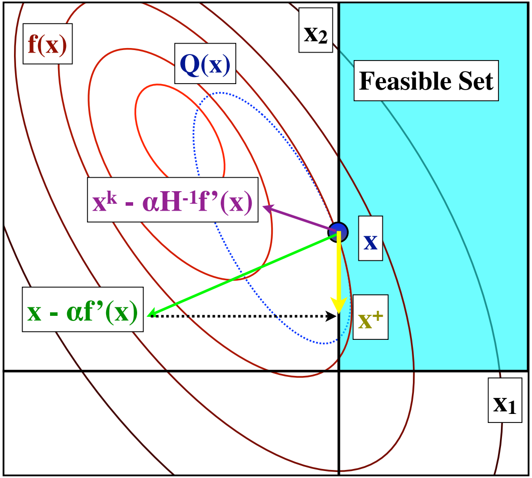
Naive Projected-Newton



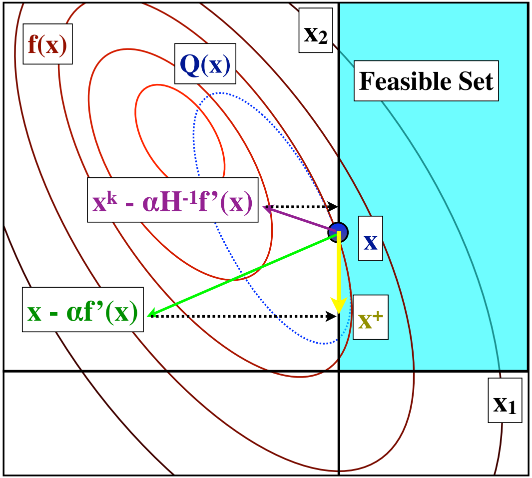
Naive Projected-Newton



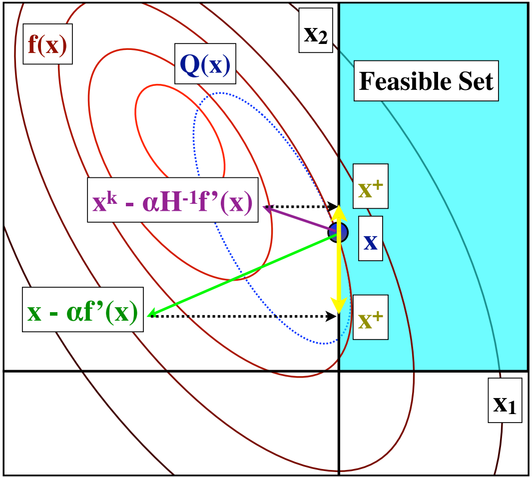
Naive Projected-Newton



Naive Projected-Newton



Naive Projected-Newton



Projected-Newton Method

- Projected-gradient minimizes quadratic approximation,

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

Projected-Newton Method

- Projected-gradient minimizes quadratic approximation,

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

- Newton's method can be viewed as quadratic approximation (wth $H^t \approx \nabla^2 f(x^t)$):

$$x^{t+1} = \underset{y \in \mathbb{R}^d}{\text{argmin}} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t}(y - x^t)H^t(y - x^t) \right\}.$$

Projected-Newton Method

- Projected-gradient minimizes quadratic approximation,

$$x^{t+1} = \operatorname*{argmin}_{y \in C} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t}\|y - x^t\|^2 \right\}.$$

- Newton's method can be viewed as quadratic approximation (wth $H^t \approx \nabla^2 f(x^t)$):

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t}(y - x^t)H^t(y - x^t) \right\}.$$

- Projected Newton minimizes constrained quadratic approximation:

$$x^{t+1} = \operatorname*{argmin}_{y \in C} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t}(y - x^t)H^t(y - x^t) \right\}.$$

Projected-Newton Method

- Projected-gradient minimizes quadratic approximation,

$$x^{t+1} = \operatorname*{argmin}_{y \in C} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t}\|y - x^t\|^2 \right\}.$$

- Newton's method can be viewed as quadratic approximation (wth $H^t \approx \nabla^2 f(x^t)$):

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t}(y - x^t)H^t(y - x^t) \right\}.$$

- Projected Newton minimizes constrained quadratic approximation:

$$x^{t+1} = \operatorname*{argmin}_{y \in C} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{1}{2\alpha_t}(y - x^t)H^t(y - x^t) \right\}.$$

- Equivalently, we project Newton step under different Hessian-defined norm,

$$x^{t+1} = \operatorname*{argmin}_{y \in C} \|y - (x^t - \alpha_t [H^t]^{-1} \nabla f(x^t))\|_{H^t},$$

where general "quadratic norm" is $\|z\|_A = \sqrt{z^T A z}$ for $A \succ 0$.

Discussion of Proximal-Newton

- Proximal-Newton is defined similarly,

$$x^{t+1} = \underset{y \in \mathbb{R}^d}{\text{argmin}} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{L}{2}(y - x^t)H^t(y - x^t) + r(y) \right\}.$$

- But this is expensive even when r is simple.

Discussion of Proximal-Newton

- Proximal-Newton is defined similarly,

$$x^{t+1} = \operatorname*{argmin}_{y \in \mathbb{R}^d} \left\{ f(x^t) + \nabla f(x^t)(y - x^t) + \frac{L}{2}(y - x^t)H^t(y - x^t) + r(y) \right\}.$$

- But this is expensive even when r is simple.
- There are a variety of practical ways to approximate this:
  - Use Barzilai-Borwein or diagonal H^t.
  - Two-metric projection: special method for separable r.
  - Inexact proximal-Newton: solve the above approximately.
    - Useful when f is very expensive but r is simple.
    - "Costly functions with simple regularizers".

(pause)

Big-N Problems

- We can write our standard regularized optimization problem as

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

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

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 more than 14 million annotated images.

Stochastic vs. Deterministic Gradient Methods

- We consider minimizing $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$.

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 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 $\{1, 2, \ldots, n\}$.

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

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.

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 $\{1, 2, \ldots, n\}$.

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

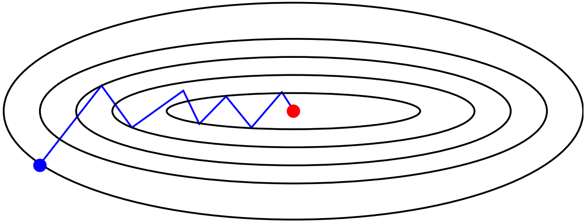
  - Direction is an unbiased estimate of true gradient,

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

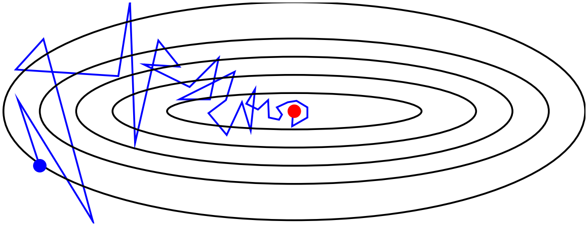
  - Iteration cost is independent of n.
  - Convergence requires $\alpha_t \to 0$.

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?

Stochastic vs. Deterministic Gradient Methods

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

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

Stochastic vs. Deterministic Gradient Methods

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

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 if only unbiased gradient available.

Stochastic vs. Deterministic Gradient Methods

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

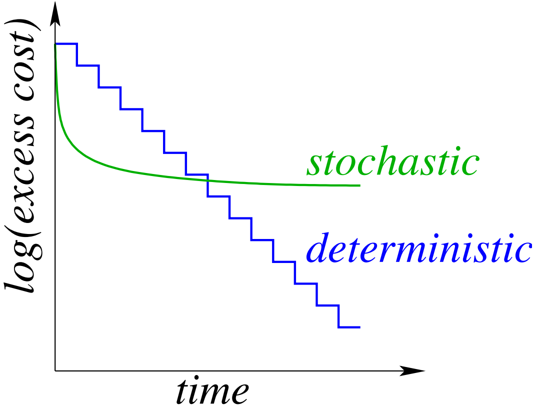
Assumption	Deterministic	Stochastic
Convex	$O(1/\sqrt{\epsilon})$	$O(1/\epsilon^2)$
Strongly	$O(\log(1/\epsilon))$	$O(1/\epsilon)$

- Stochastic has low iteration cost but slow convergence rate.
  - Sublinear rate even in strongly-convex case.
  - Bounds are unimprovable if only unbiased gradient available.
- Nesterov acceleration and momentum do not improve rate:
  - In fact, the momentum must go to zero for convergence.

[Tseng, 1998]

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 objective:

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

Stochastic vs. Deterministic for Non-Smooth

- The story changes for non-smooth problems.
- Consider the binary support vector machine 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) are not faster.

Stochastic vs. Deterministic for Non-Smooth

- The story changes for non-smooth problems.
- Consider the binary support vector machine 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) are not faster.
- 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

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

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

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

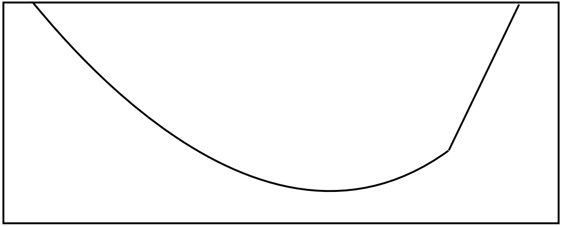
Sub-Gradients and Sub-Differentials

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

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



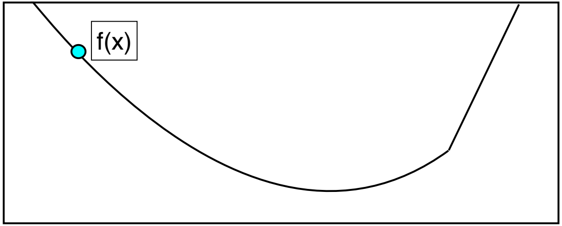
Sub-Gradients and Sub-Differentials

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

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



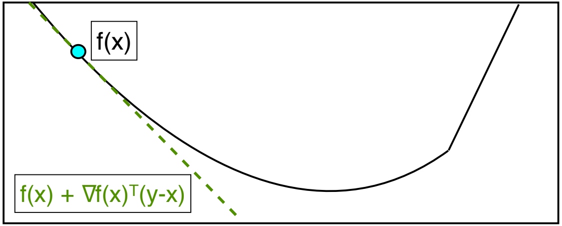
Sub-Gradients and Sub-Differentials

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*

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



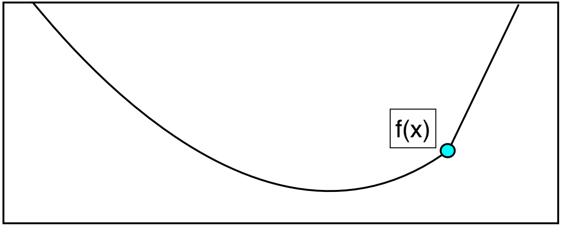
Sub-Gradients and Sub-Differentials

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

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



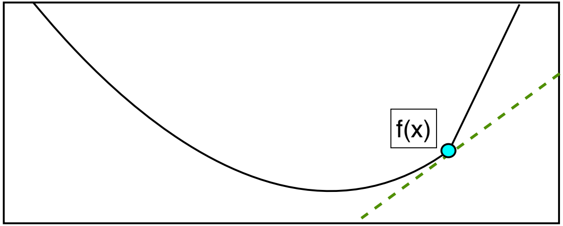
Sub-Gradients and Sub-Differentials

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

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



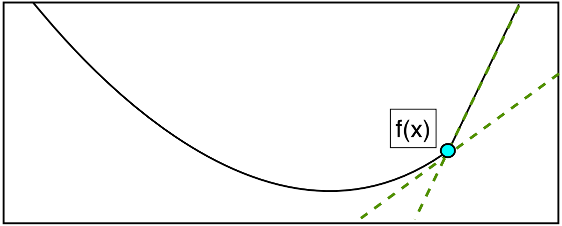
Sub-Gradients and Sub-Differentials

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

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



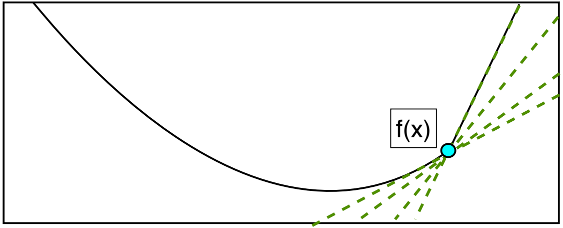
Sub-Gradients and Sub-Differentials

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

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



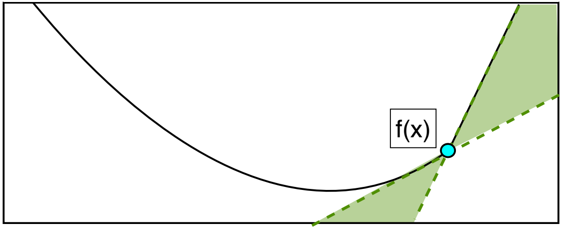
Sub-Gradients and Sub-Differentials

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

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



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 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 Value and Max Functions

- Sub-differential of absolute value function:

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

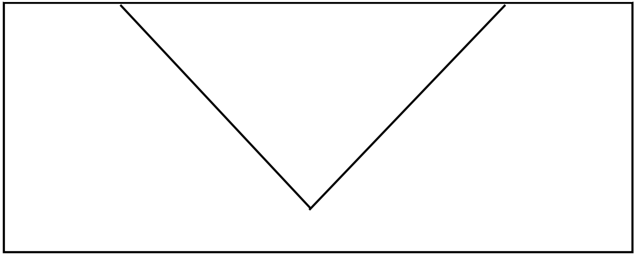
(sign of the variable if non-zero, anything in $[-1, 1]$ at 0)

Sub-Differential of Absolute Value and Max Functions

- 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)

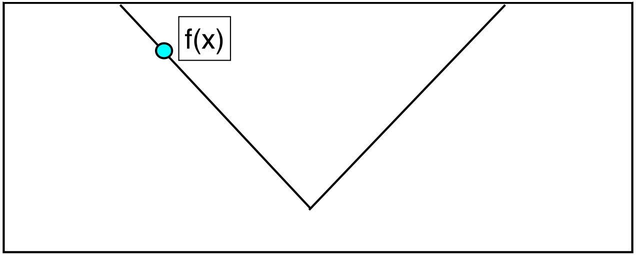


Sub-Differential of Absolute Value and Max Functions

- 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)

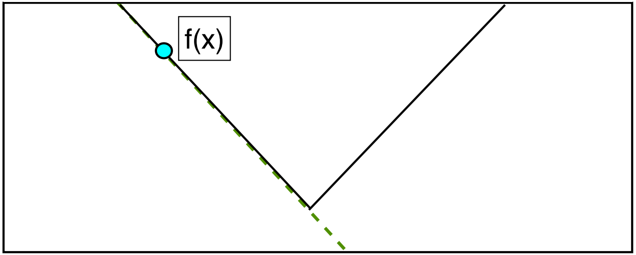


Sub-Differential of Absolute Value and Max Functions

- 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)

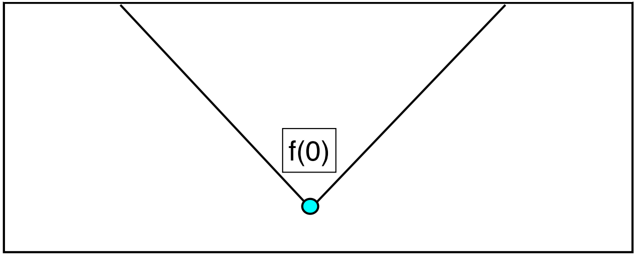


Sub-Differential of Absolute Value and Max Functions

- 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)

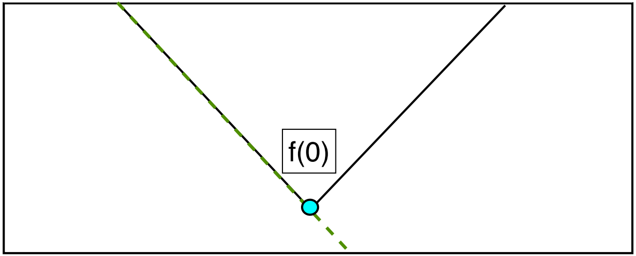


Sub-Differential of Absolute Value and Max Functions

- 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)

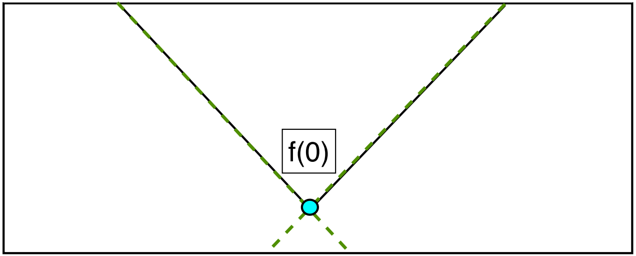


Sub-Differential of Absolute Value and Max Functions

- 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)

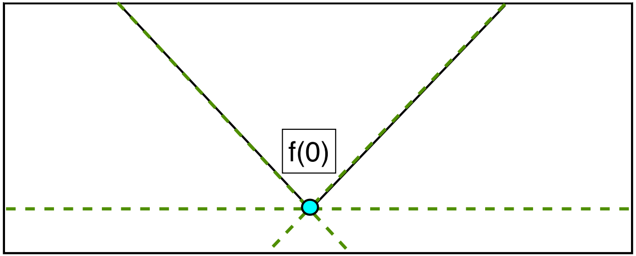


Sub-Differential of Absolute Value and Max Functions

- 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)

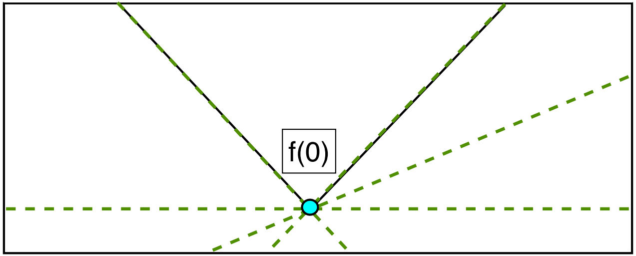


Sub-Differential of Absolute Value and Max Functions

- 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)

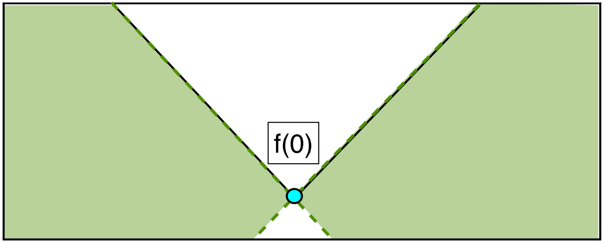


Sub-Differential of Absolute Value and Max Functions

- 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 Value and Max Functions

- 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:

$$\partial(f_1(x) + f_2(x)) = \partial f_1(x) + \partial f_2(x).$$

Sub-Differential of Absolute Value and Max Functions

- 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:

$$\partial(f_1(x) + f_2(x)) = \partial f_1(x) + \partial f_2(x).$$

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

$$\partial \max\{f_1(x), f_2(x)\} =$$

Sub-Differential of Absolute Value and Max Functions

- 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:

$$\partial(f_1(x) + f_2(x)) = \partial f_1(x) + \partial f_2(x).$$

- Sub-differential of max of 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 $g_t \in \partial f(x^t)$.

Subgradient Method

- The basic subgradient method:

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

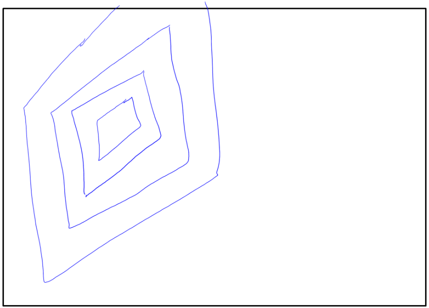
  for some $g_t \in \partial f(x^t)$.
- Unfortunately, may increase the objective even for small α_t.
- But, distance to solution decreases:
  - $\|x^{t+1} - x^*\| < \|x^t - x^*\|$ for small enough α.

Subgradient Method

- The basic subgradient method:

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

  for some $g_t \in \partial f(x^t)$.
- Unfortunately, may increase the objective even for small α_t.
- But, distance to solution decreases:
  - $\|x^{t+1} - x^*\| < \|x^t - x^*\|$ for small enough α.

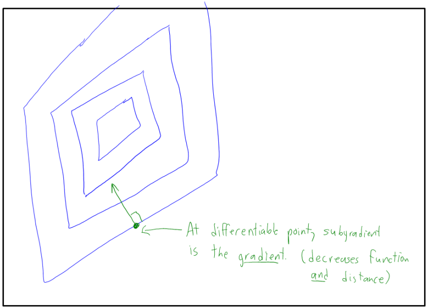


Subgradient Method

- The basic subgradient method:

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

  for some $g_t \in \partial f(x^t)$.
- Unfortunately, may increase the objective even for small α_t.
- But, distance to solution decreases:
  - $\|x^{t+1} - x^*\| < \|x^t - x^*\|$ for small enough α.

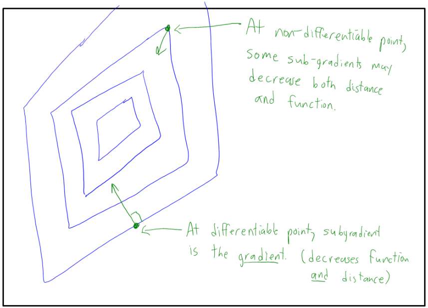


Subgradient Method

- The basic subgradient method:

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

  for some $g_t \in \partial f(x^t)$.
- Unfortunately, may increase the objective even for small α_t.
- But, distance to solution decreases:
  - $\|x^{t+1} - x^*\| < \|x^t - x^*\|$ for small enough α.

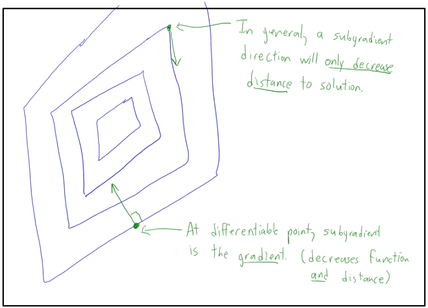


Subgradient Method

- The basic subgradient method:

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

  for some $g_t \in \partial f(x^t)$.
- Unfortunately, may increase the objective even for small α_t.
- But, distance to solution decreases:
  - $\|x^{t+1} - x^*\| < \|x^t - x^*\|$ for small enough α.



Strong-Convexity Inequalities for Non-Differentiable f

- A “first-oder” 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.

Strong-Convexity Inequalities for Non-Differentiable f

- A "first-oder" 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

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

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 $i_t \in \{1, 2, \ldots, n\}$.

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 $i_t \in \{1, 2, \dots, n\}$.
- Stochastic subgradient is n times faster with similar convergence properties.
- We'll conisder it under the standard assumptions that
  - f is μ-strongly-convex:
  - $\mathbb{E}[\|g_t\|^2] \leq B^2$ (finite variance and bounded subgradients).

Convergnece Rate of Stochastic Subgradient

- Since function value may not decrease, we analyze distance to x^*:

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

Convergnece Rate of Stochastic Subgradient

- Since function value may not decrease, we analyze distance to x^*:

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

- Many analyses of distance to x^* start this way.
- First term is we what we want, we need to bound the second/third terms.

Convergnece Rate of Stochastic Subgradient

- Expansion of distance:

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

Convergnece Rate of Stochastic Subgradient

- Expansion of distance:

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

- Take expectation with respect to i_t:

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

Convergnece Rate of Stochastic Subgradient

- Expansion of distance:

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

- Take expectation with respect to i_t:

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

- Using strong-convexity inequality,

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

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.

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.
- Taking full expectation and applying recursively with constant $\alpha_t = \alpha$ gives:

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

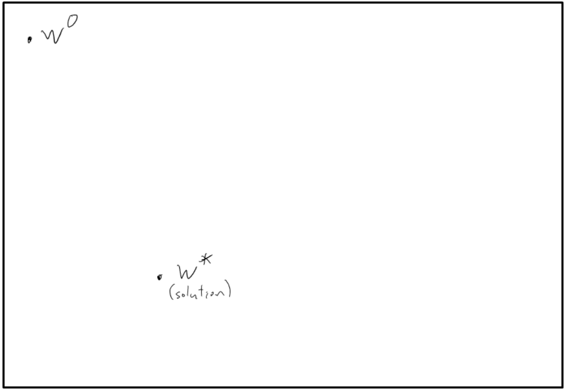
  after some of math (last term comes from bounding a geometric series).
- First term looks like linear convergence, but second term does not go to zero.

Stochastic Gradient with Constant Step Size

- Our bound on expected distance:

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

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

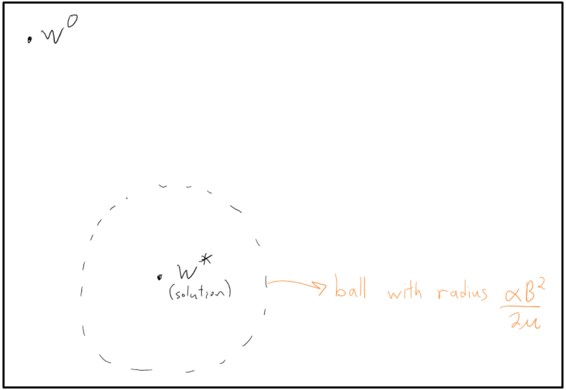


Stochastic Gradient with Constant Step Size

- Our bound on expected distance:

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

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

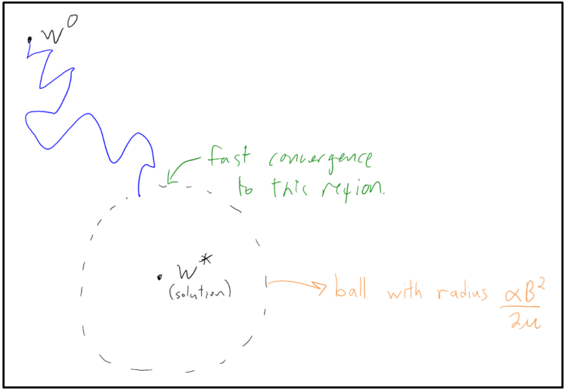


Stochastic Gradient with Constant Step Size

- Our bound on expected distance:

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

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

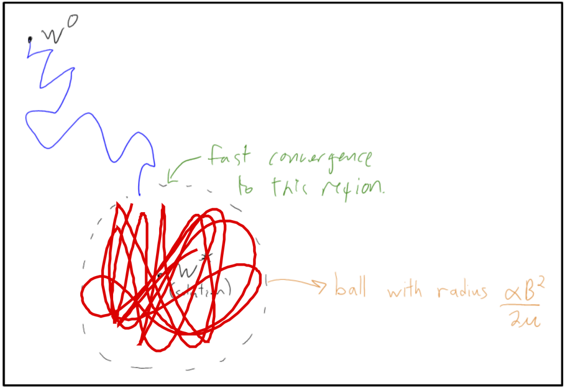


Stochastic Gradient with Constant Step Size

- Our bound on expected distance:

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

- 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.
  - Region that we converge to shrinks over time.
  - But it can't shrink too quickly or we may never reach x^*.

Stochastic Gradient with Decreasing Step Size

- To get convergence, we need a decreasing step size.
  - Region that we converge to shrinks over time.
  - But it can't shrink too quickly or we may never reach x^*.
  - 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 $\alpha_t = O(1/t)$.

Stochastic Gradient with Decreasing Step Size

- To get convergence, we need a decreasing step size.
  - Region that we converge to shrinks over time.
  - But it can't shrink too quickly or we may never reach x^*.
  - 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

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

      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

- Proximal-gradient: linear rates for sum of smooth and non-smooth.
- Proximal-Newton: even faster rates in special cases.

Summary

- Proximal-gradient: linear rates for sum of smooth and non-smooth.
- Proximal-Newton: even faster rates in special cases.
- 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.

Summary

- Proximal-gradient: linear rates for sum of smooth and non-smooth.
- Proximal-Newton: even faster rates in special cases.
- 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: faster stochastic methods, and kernels for exponential/infinite bases.