Optimization for Statistics and Machine Learning

Eric Chi

August 11, 2016

Basic Inequality

recall basic inequality for convex differentiable f:

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

- first-order approximation of f at x is a global lower bound

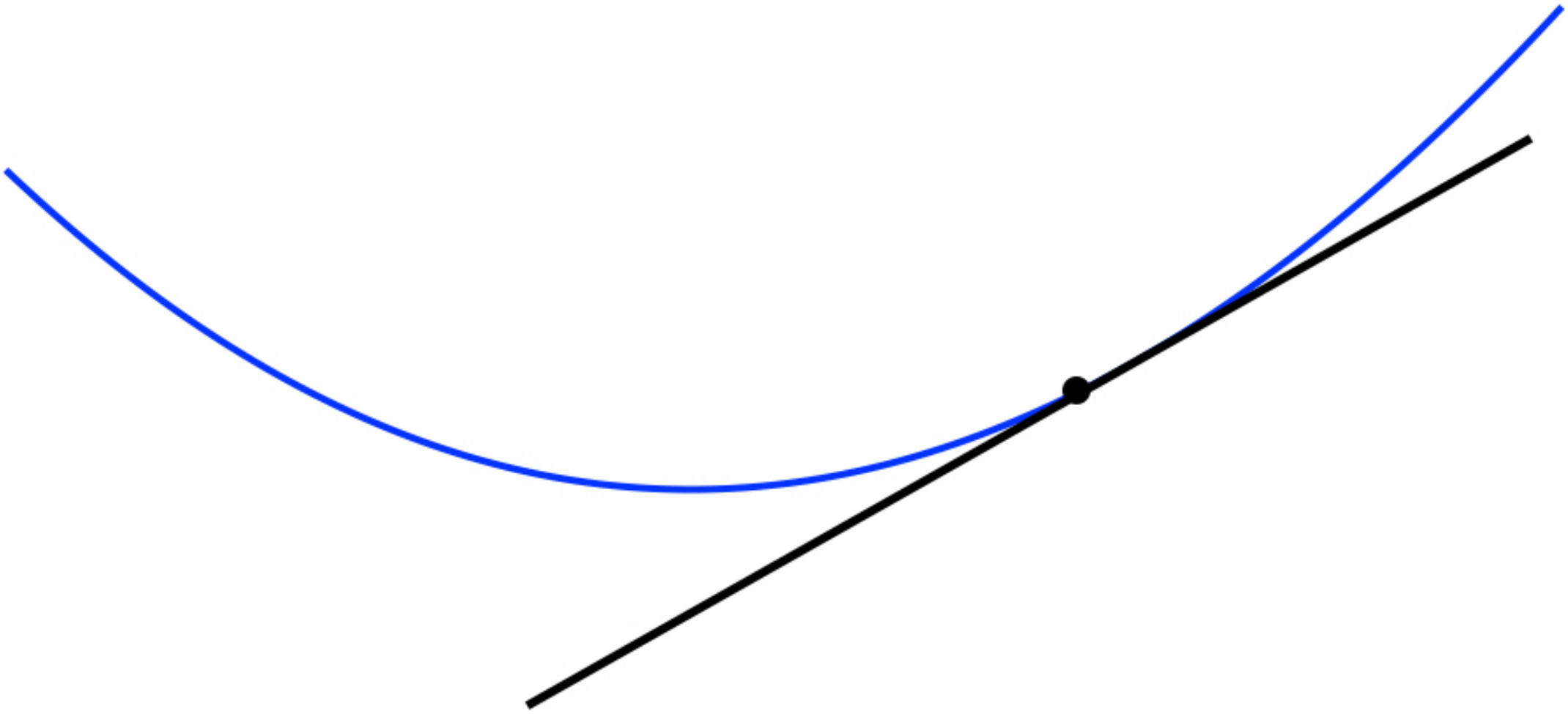


Figure 1: Best linear approximation

Subgradient

Definition: g is a subgradient of a convex function f at $x \in \mathrm{dom} f$ if

$$f(y) \geq f(x) + g^T(y - x), \ \forall y \in \mathrm{dom} f$$

Properties:

- $f(x) + g^T(y - x)$ is a global lower bound on $f(y)$
- if f is convex and differentiable, then $\nabla f(x)$ is a subgradient of f at x

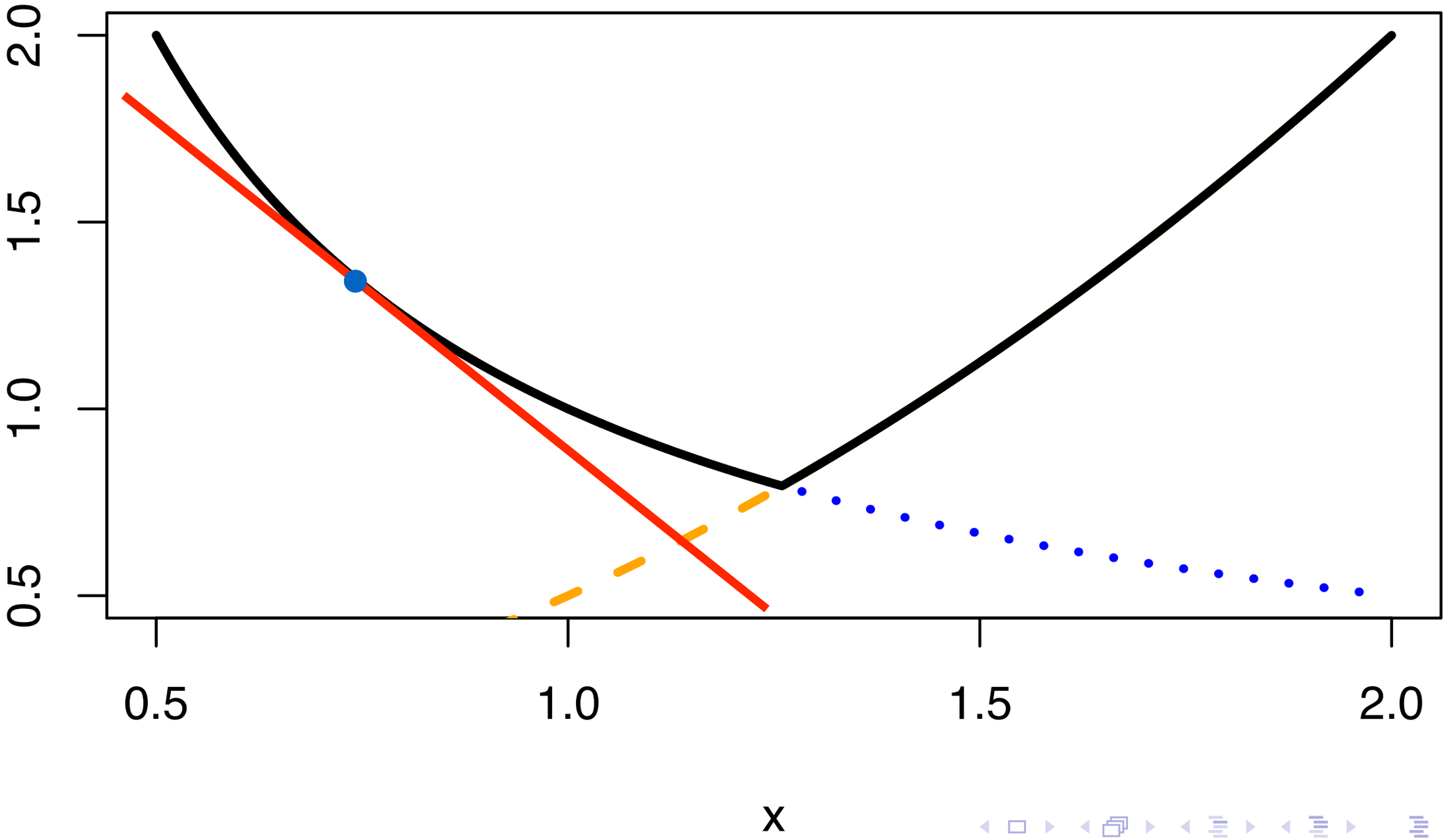
Applications:

- algorithms for nondifferentiable convex optimization
- optimality conditions, duality for nondifferentiable problems

Example

$f(x) = \max\{f_1(x), f_2(x)\}, \quad f_1, f_2$ convex and differentiable

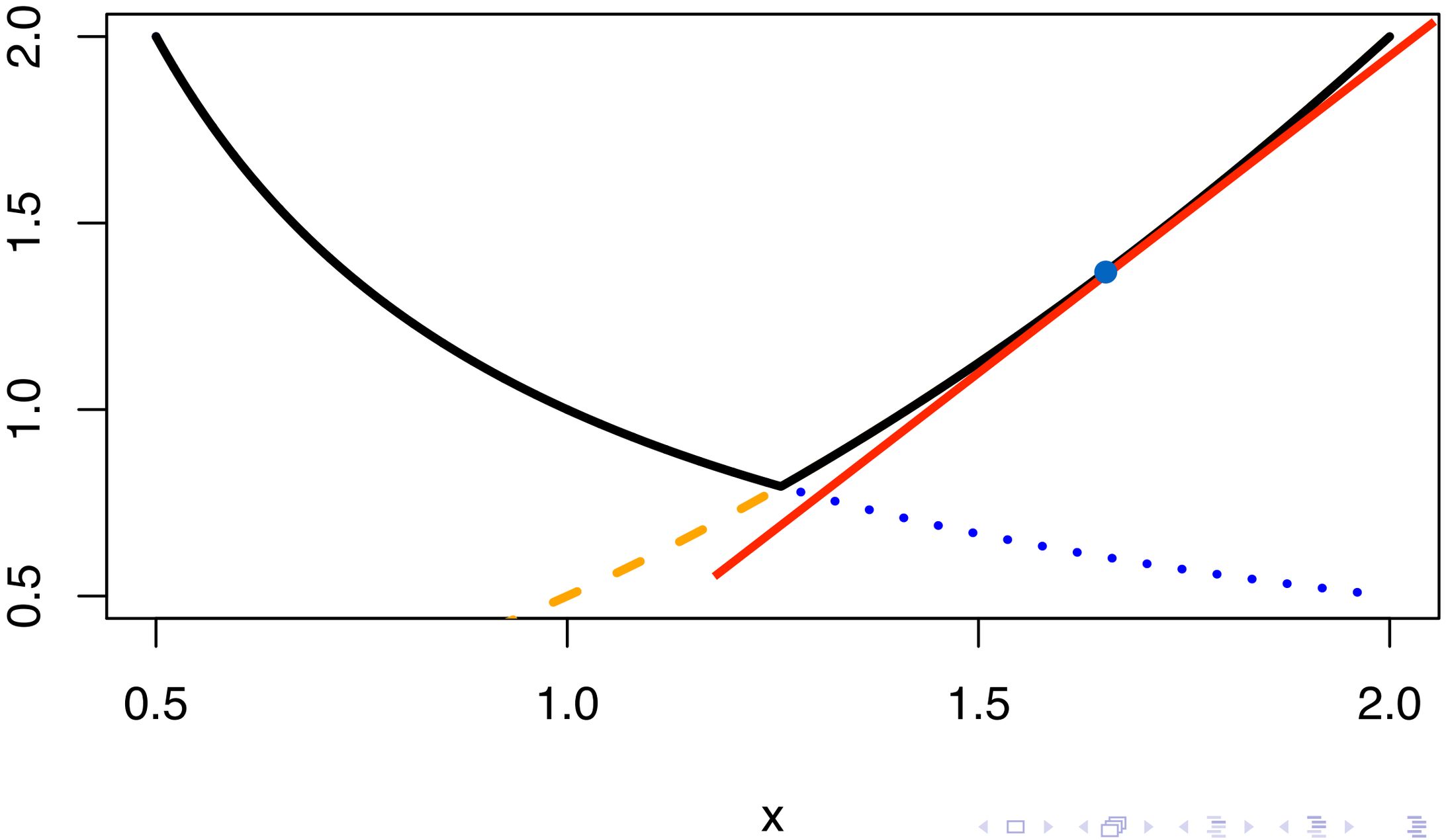
- subgradients at x_0 form line segment $[\nabla f_1(x_0), \nabla f_2(x_0)]$
- if $f_1(\hat{x}) > f_2(\hat{x})$, subgradient of f at $\hat{x}$ is $\nabla f_1(\hat{x})$
- if $f_1(\hat{x}) < f_2(\hat{x})$, subgradient of f at $\hat{x}$ is $\nabla f_2(\hat{x})$



Example

$f(x) = \max\{f_1(x), f_2(x)\}, \quad f_1, f_2$ convex and differentiable

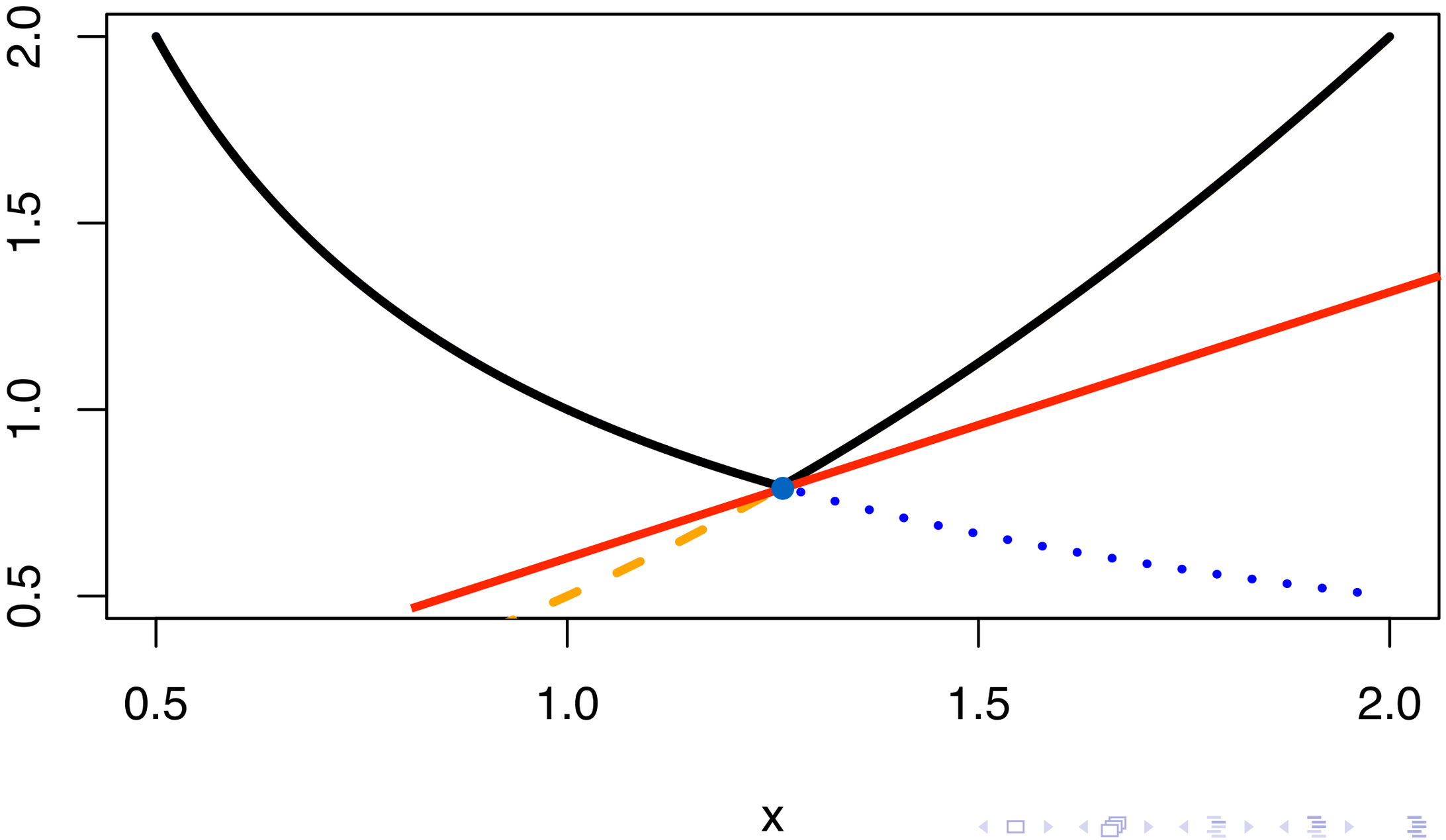
- subgradients at x_0 form line segment $[\nabla f_1(x_0), \nabla f_2(x_0)]$
- if $f_1(\hat{x}) > f_2(\hat{x})$, subgradient of f at $\hat{x}$ is $\nabla f_1(\hat{x})$
- if $f_1(\hat{x}) < f_2(\hat{x})$, subgradient of f at $\hat{x}$ is $\nabla f_2(\hat{x})$



Example

$f(x) = \max\{f_1(x), f_2(x)\}, \quad f_1, f_2$ convex and differentiable

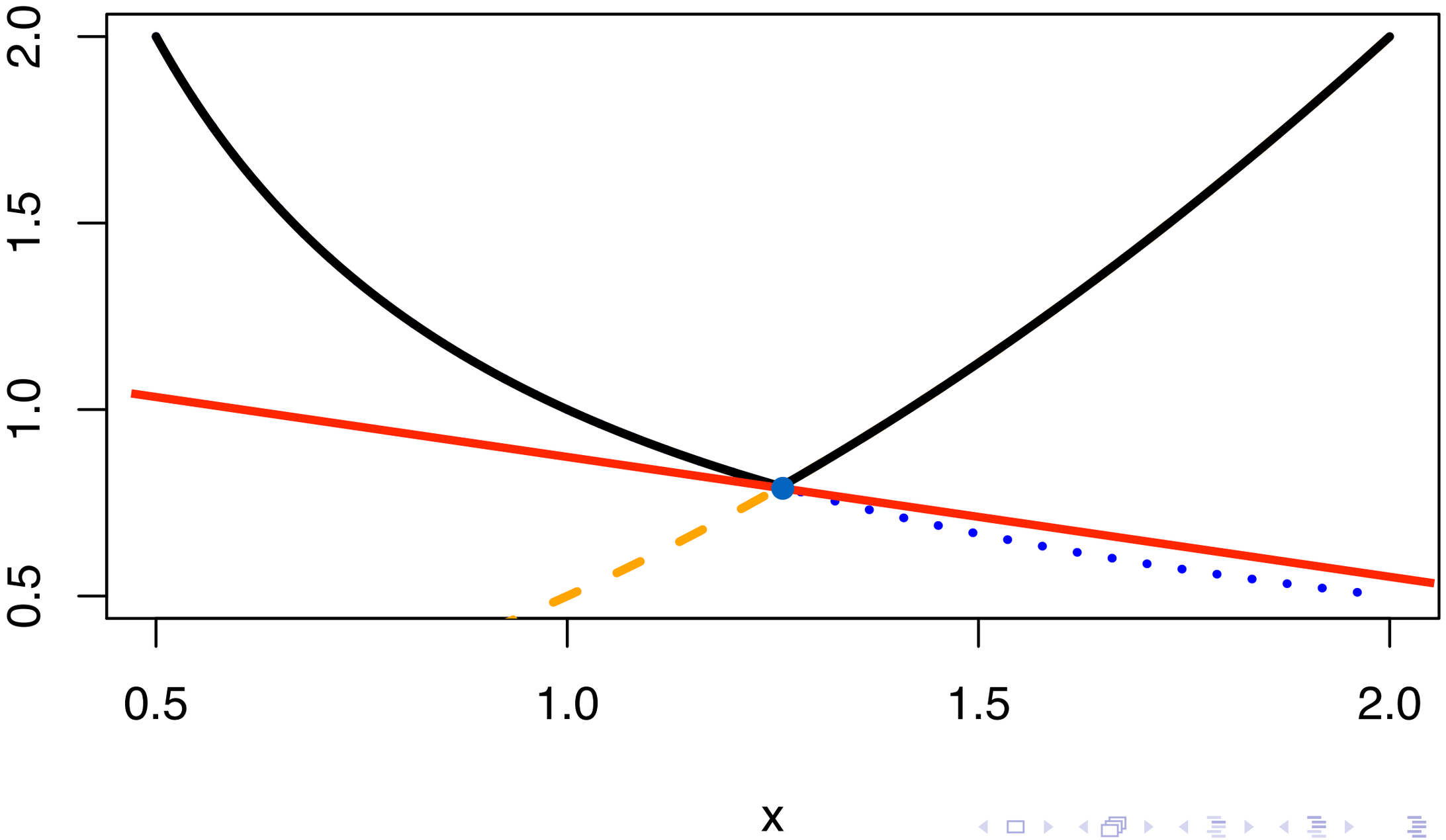
- subgradients at x_0 form line segment $[\nabla f_1(x_0), \nabla f_2(x_0)]$
- if $f_1(\hat{x}) > f_2(\hat{x})$, subgradient of f at $\hat{x}$ is $\nabla f_1(\hat{x})$
- if $f_1(\hat{x}) < f_2(\hat{x})$, subgradient of f at $\hat{x}$ is $\nabla f_2(\hat{x})$



Example

$f(x) = \max\{f_1(x), f_2(x)\}, \quad f_1, f_2$ convex and differentiable

- subgradients at x_0 form line segment $[\nabla f_1(x_0), \nabla f_2(x_0)]$
- if $f_1(\hat{x}) > f_2(\hat{x})$, subgradient of f at $\hat{x}$ is $\nabla f_1(\hat{x})$
- if $f_1(\hat{x}) < f_2(\hat{x})$, subgradient of f at $\hat{x}$ is $\nabla f_2(\hat{x})$



Subdifferential

the subdifferential $\partial f(x)$ of f at x is the set of all subgradients:

$$\partial f(x) = \{g : g^T(y - x) \leq f(y) - f(x), \forall y \in \text{dom } f\}$$

properties

- $\partial f(x)$ is a closed convex set (possibly empty)
- if $x \in \text{int dom } f$ then $\partial f(x)$ is nonempty and bounded

Some basic rules

differentiable functions: $\partial f(x) = \{\nabla f(x)\}$ if f is differentiable at x

nonnegative combination:
if $h(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x)$ with $\alpha_1, \alpha_2 \geq 0$, then

$$\partial h(x) = \alpha_1 \partial f_1(x) + \alpha_2 \partial f_2(x)$$

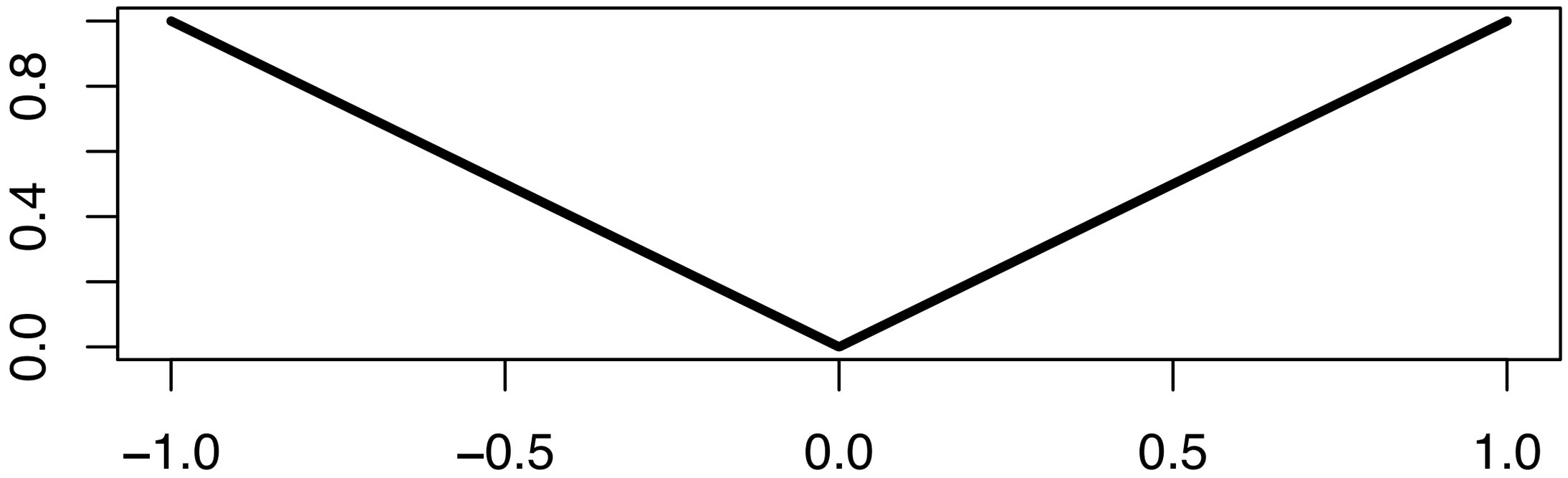
affine transformation of variables: if h(x) = f (Ax + b), then

$$\partial h(x) = A^T \partial f(Ax + b)$$

Examples

absolute value $f(x) = |x|$

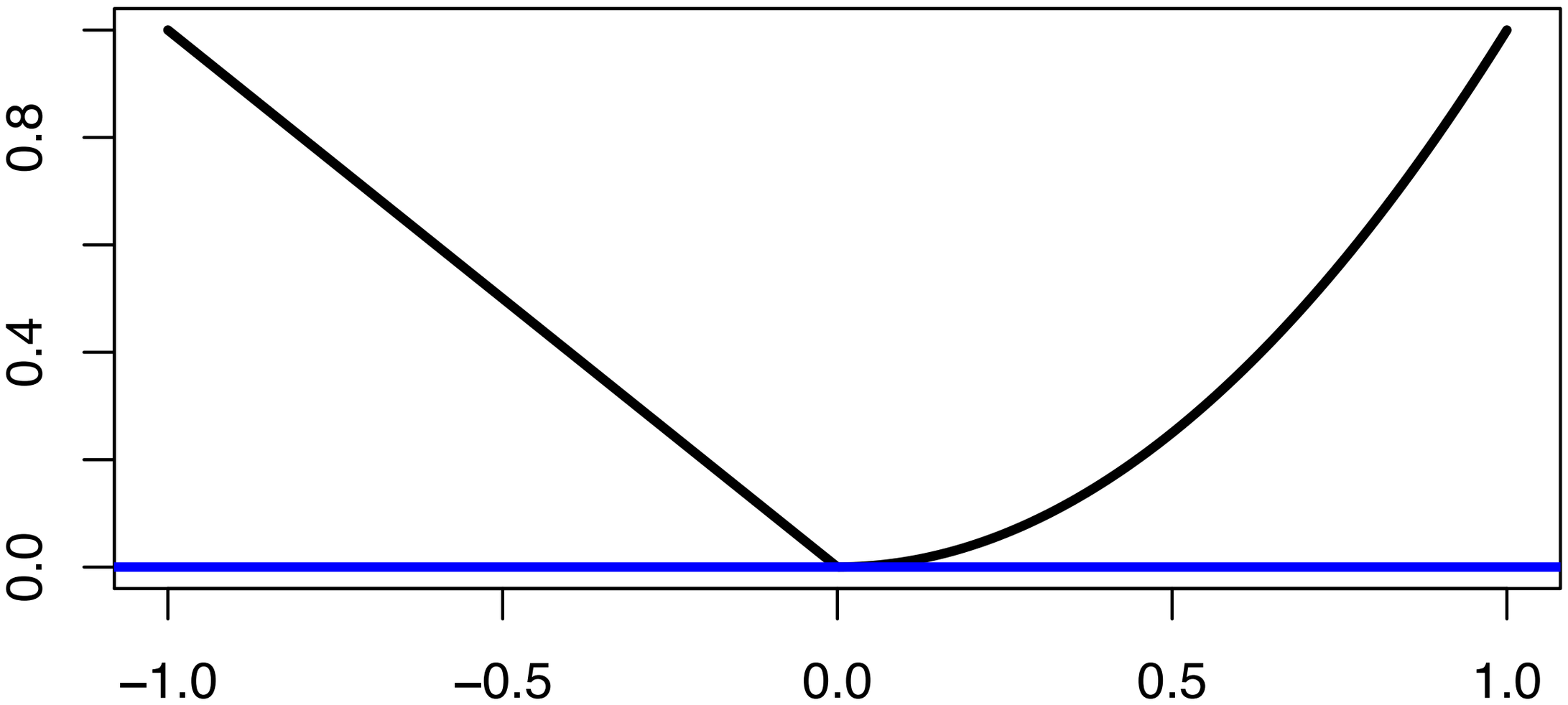
$$f(x) = \max\{-x, x\}$$



$$\partial f(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ [-1, 1] & \text{if } x = 0 \end{cases}$$

Optimality conditions — unconstrained

Property: $x^\star$ minimizes $f(x)$ if and only if $0 \in \partial f(x^\star)$



proof: For all x

$$0 \in \partial f(x^\star) \iff f(x) \geq f(x^\star) + 0^T(x - x^\star)$$

Univariate Lasso

$$\text{minimize } f(x) = \frac{1}{2}(x-a)^2 + \lambda|x|$$

Q: How do we know f attains a global minimum $x^\star$?

Stationarity condition

$$a - x \in \lambda\partial|x|$$

Case 1: $x^\star > 0 \implies \lambda\partial|x^\star| = \{\lambda\}$

$$a - x = \lambda$$

$$x = a - \lambda > 0$$

Case 2: $x^\star < 0 \implies \lambda\partial|x^\star| = \{-\lambda\}$

$$a - x = -\lambda$$

$$x = a + \lambda < 0$$

Case 3: $x^\star = 0 \implies \lambda\partial|x^\star| = [\lambda, \lambda]$

$$a - x \in [-\lambda, \lambda]$$

Univariate Lasso

$$\text{minimize } f(x) = \frac{1}{2}(x-a)^2 + \lambda|x|$$

$$x^\star = \begin{cases} a-\lambda & a > \lambda \\ a+\lambda & a < -\lambda \\ 0 & |a| \leq \lambda \end{cases}$$

The mapping is called the **softthreshold** operator

An alternative and more compact representation

$$x^\star = S(a,\lambda) = \text{sgn}(a)[|a| - \lambda]_+$$

This is a very useful operation. We will use it to solve

- The multivariate lasso regression problem
- Low rank matrix completion

Multivariate Lasso

Q: How does the R package glmnet solve the multivariate lasso regression?
A: Coordinate Descent
The objective function

$$f(b) = \frac{1}{2n}\|y - Xb\|_2^2 + \lambda\|b\|_1$$
$$= \frac{1}{2n}\|y - \sum_{j=1}^{p} b_j x_j\|_2^2 + \lambda\|b\|_1$$

Hold all b_j fixed except one and optimize over it.

Multivariate Lasso

$$r_k = y - \sum_{j \neq k} b_j x_j$$

$$\begin{aligned} f(b) &= \frac{1}{2n}\|r_k - b_k x_k\|_2^2 + \lambda\|b\|_1 \\ &= \frac{1}{2n} b_k^2 \|x_k\|_2^2 - \frac{1}{n} b_k r_k^T x_k + \lambda |b_k| + c \\ &= \frac{1}{n}\|x_k\|_2^2 \left[\frac{1}{2}\left(b_k - \frac{r_k^T x_k}{\|x_k\|_2^2} \right)^2 + \frac{\lambda n}{\|x_k\|_2^2} |b_k| \right] + c' \end{aligned}$$

where c, c' are constants that do not depend on b_k.

Update rule:

$$\begin{aligned} b_k &= S\left(\frac{r_k^T x_k}{\|x_k\|_2^2}, \frac{\lambda n}{\|x_k^2\|_2^2} \right) \\ &= \frac{1}{\|x_k\|_2^2} S\left(r_k^T x_k, \lambda n \right) \end{aligned}$$

Coordinate Descent

Start with initial guess $x^{(0)}$ and repeat until convergence for $k = 1, 2, 3, \ldots$

$$x_1^{(k)} = \arg\min_{x_1} f(x_1, x_2^{(k-1)}, x_3^{(k-1)}, \ldots, x_n^{(k-1)})$$

$$x_2^{(k)} = \arg\min_{x_2} f(x_1^{(k)}, x_2, x_3^{(k-1)}, \ldots, x_n^{(k-1)})$$

$$\vdots$$

$$x_n^{(k)} = \arg\min_{x_n} f(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, \ldots, x_n)$$

Coordinate Descent

Tseng (2001) proves that for $f = g + \sum_i h_i$ with f continuous on a compact set $\{x : f(x) \leq f(x^{(0)})\}$ and f attains its minimum, any limit point of the sequence $x^{(k)}$ is a minimizer of f

- Okay to cycle through coordinates in any order
- Okay to update over disjoint blocks of coordinates

Pathwise coordinate descent for lasso

Friedman et al. (2007), Friedman et al. (2009)
Outer loop (pathwise):

- Compute the solution at sequence $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r$
- **Homotopy / Warm-start:** For λ_k, initialize algorithm at the solution for λ_{k-1}

Inner loop (active set) **VERY IMPORTANT**

- Perform one coordinate cycle (or small number of cycles), and record active set S of coefficients that are nonzero
- Cycle over coefficients in S until convergence
- Check KKT conditions over all coefficients; if not all satisfied, add coefficients in violation of KKT to S, and repeat

References

surveys of fast gradient methods

- Nesterov, Introductory Lectures on Convex Optimization. A Basic Course (2004)
- Tseng, On accelerated proximal gradient methods for convex-concave optimization (2008)

FISTA

- Beck and Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems
- Su, Boyd, and Candés, A Differential Equation for Modeling Nesterov's Accelerated Gradient Method: Theory and Insights

Pathwise optimization

- Friedman, Hastie, Hofling, and Tibshirani, Pathwise coordinate optimization
- Xiao, Wu, and Zhou, ConvexLAR: An Extension of Least Angle Regression

(Noisy) Matrix Completion

$$\text{minimize} \frac{1}{2}\|P_\Omega(X) - P_\Omega(Z)\|_{\text{F}}^2 + \lambda\|Z\|_*,$$

where Ω is a set of observed indices and

$$[P_\Omega(Z)]_{ij} = \begin{cases} z_{ij} & \text{for } (i,j) \in \Omega \\ 0 & \text{o.w.} \end{cases}$$

An MM algorithm for matrix completion

Suppose we already have the nth iterate $Z^{(n)}$. Observe that the following quadratic function of Z is always nonnegative

$$0 \leq \sum_{(i,j)\in\Omega^c} (z_{ij} - z_{ij}^{(n)})^2,$$

and the inequality becomes equality when $Z = Z^{(n)}$. Adding the above inequality to the equality above it produces the desired quadratic majorization anchored at the nth iterate $Z^{(n)}$.

An MM algorithm for matrix completion

$$\frac{1}{2}\|P_\Omega(X) - P_\Omega(Z)\|_{\mathrm{F}}^2 \leq \frac{1}{2}\|Y - Z\|_{\mathrm{F}}^2$$

where

$$y_{ij} = \begin{cases} x_{ij} & \text{for } (i,j) \in \Omega \\ z_{ij}^{(n)} & \text{o.w.} \end{cases}$$

Just need to be able to solve

$$\text{minimize } \frac{1}{2}\|Y - Z\|_{\mathrm{F}}^2 + \lambda\|Z\|_*$$

An MM algorithm for matrix completion

$$\begin{aligned}
\frac{1}{2}\|Y - Z\|_{\mathsf{F}}^2 + \lambda\|Z\|_* &= \frac{1}{2}\|Y\|_{\mathsf{F}}^2 - \mathrm{tr}(Y^T Z) + \frac{1}{2}\|Z\|_{\mathsf{F}}^2 + \lambda\|Z\|_* \\
&= \frac{1}{2}\sum_{i=1}^{n}\sigma_i^2 - \mathrm{tr}(Y^T Z) + \frac{1}{2}\sum_{i=1}^{n} d_i^2 + \lambda\sum_{i=1}^{n} d_i \\
&\geq \frac{1}{2}\sum_{i=1}^{n}\sigma_i^2 - \sum_{i=1}^{n}\sigma_i d_i + \frac{1}{2}\sum_{i=1}^{n} d_i^2 + \lambda\sum_{i=1}^{n} d_i
\end{aligned}$$

Lower bound is due to Fan's inequality. Equality is attained if and only if Y and Z share the same left and right singular vectors U and V.

An MM algorithm for matrix completion

Optimal Z has singular value decomposition $Z = UDV^T$ and the singular values d_i satisfy

$$(d_1, \ldots, d_n) = \arg\min \sum_{i=1}^{n} \left[\frac{1}{2}\sigma_i^2 - \sigma_i d_i + d_i^2 + \lambda |d_i| \right]$$

$$\begin{aligned} d_i &= \arg\min \frac{1}{2}[\sigma_i - d_i]^2 + \lambda |d_i| \\ &= S(\sigma_i, \lambda) \end{aligned}$$

An MM algorithm for matrix completion

Given a data matrix X, a set of observed indices Ω, and an initial model matrix $Z^{(0)}$, repeat the following four steps until convergence.

1. $y_{ij}^{(n+1)} = x_{ij}$ if $(i,j) \in \Omega$ and $z_{ij}^{(n)}$ if not.
2. $(U, D, V) = \text{SVD}(Y^{(n+1)})$
3. $\tilde{D} = S(D, \lambda)$
4. $Z^{(n+1)} = U\tilde{D}V^T$

This is the softimpute algorithm (Mazumder et al. 2010)

References

Coordinate Descent & Lasso

- Friedman, Hastie, Hofling, and Tibshirani, Pathwise coordinate optimization
- Wu and Lange, Coordinate descent algorithms for lasso penalized regression

Matrix Completion with MM algorithm

- Mazumder, Hastie, and Tibshirani, Spectral Regularization Algorithms for Learning Large Incomplete Matrices
- Chi, Zhou, Chen, Ortega-Del Vecchyo, and Lange, Genotype imputation via matrix completion

What is a Projection?

Given: A point z in R^n and a non-empty set C.
Goal: Find a point in C that is "closest" to z.
Properties we'd like to have:

1. Such a point always exists.
2. Such a point is always unique.

A sufficient solution:

1. C is closed and convex.
2. "close" is measured in Euclidean distance.

Why are computing projections relevant?

All kinds of regression can use projections as an algorithmic primitive.

- Least squares linear regression
- Shape restricted regression
  - Isotonic regression
  - Convex regression
- Penalized regression
  - Lasso, generalized lasso, group lasso, etc.

Methods for solving the feasibility problem can use projections as an algorithmic primitive.

The Projection Theorem

Let C be a non-empty closed convex set in R^n.

1. For every $z \in R^n$, there exists a unique minimizer of

$$f(x) = \|x - z\|_2^2$$

over all $x \in C$

   - Call it the projection of z onto C
   - Denote it by $P_C(z) = \arg\min_{x \in C} f(x)$.

2. $z^\star = P_C(z)$ if and only if

$$(x - z^\star)^t (z - z^\star) \leq 0, \quad \forall x \in C.$$

The Projection Theorem Proof

Let C be a non-empty closed convex set in R^n.

1. For every $z \in R^n$, there exists a unique minimizer of

$$f(x) = \|x - z\|_2^2$$

over all $x \in C$.

Proof: Since f has compact level sets $\Longrightarrow$, it has at least one minimizer. Since f is strictly convex, a global minimizer is unique.

The Projection Theorem Proof

Let C be a non-empty closed convex set in R^n.

2. $z^\star = P_C(z)$ if and only if

$$(x - z^\star)^t (z - z^\star) \leq 0, \quad \forall x \in C.$$

Proof: $z^\star$ is the global minimizer of f if and only if

$$\nabla f(z^\star)^t (x - z^\star) \geq 0, \quad \forall x \in C.$$

But

$$\nabla f(z^\star) = 2(z^\star - z).$$

Example: The Projection onto a line

Let $C = \text{span}\{a\}$, where $a \in R^n$.

$$P_C(x) = \frac{\langle a, x \rangle}{\|a\|_2^2} a = \langle u, x \rangle u,$$

where $u = \frac{a}{\|a\|_2}$.

Work: $\mathcal{O}(n)$

Example: The Projection onto a subspace

Let $C = \text{span}\{a_1, \ldots, a_p\}$, where $a \in R^n$.

$$P_C(x) = A(A^tA)^{-1}A^tx$$

Work: $\mathcal{O}(np^2)$

$$\begin{aligned} P_C(y) &= \arg\min_{\hat{y} \in C} \|\hat{y} - y\|_2 \\ &= X\left[\arg\min_b \|Xb - y\|_2\right] \\ \hat{y} &= X(X^tX)^{-1}X^ty \end{aligned}$$

Least squares prediction is the projection of the observed response y onto the column space of the design X.

Example: Projection onto a 2-norm ball

Let $C = \{y \in R^n : \|y - z\|_2 \leq r\}$.

$$P_C(x) = \begin{cases} z + r\frac{x-z}{\|x-z\|_2} & x \notin C \\ x & x \in C. \end{cases}$$

Work: $\mathcal{O}(n)$

Projection onto Closed Convex Set

Indicator function of a closed convex set C

$$\delta_C(x) = \begin{cases} 0 & x \in C \\ \infty & x \notin C \end{cases}$$

$$P_C(x) = \arg\min_{u \in C} \|u - x\|_2^2 = \arg\min_u \|u - x\|_2^2 + \delta_C(x)$$

sudifferential of δ_C is the **normal cone** to C at x denoted $N_C(x)$

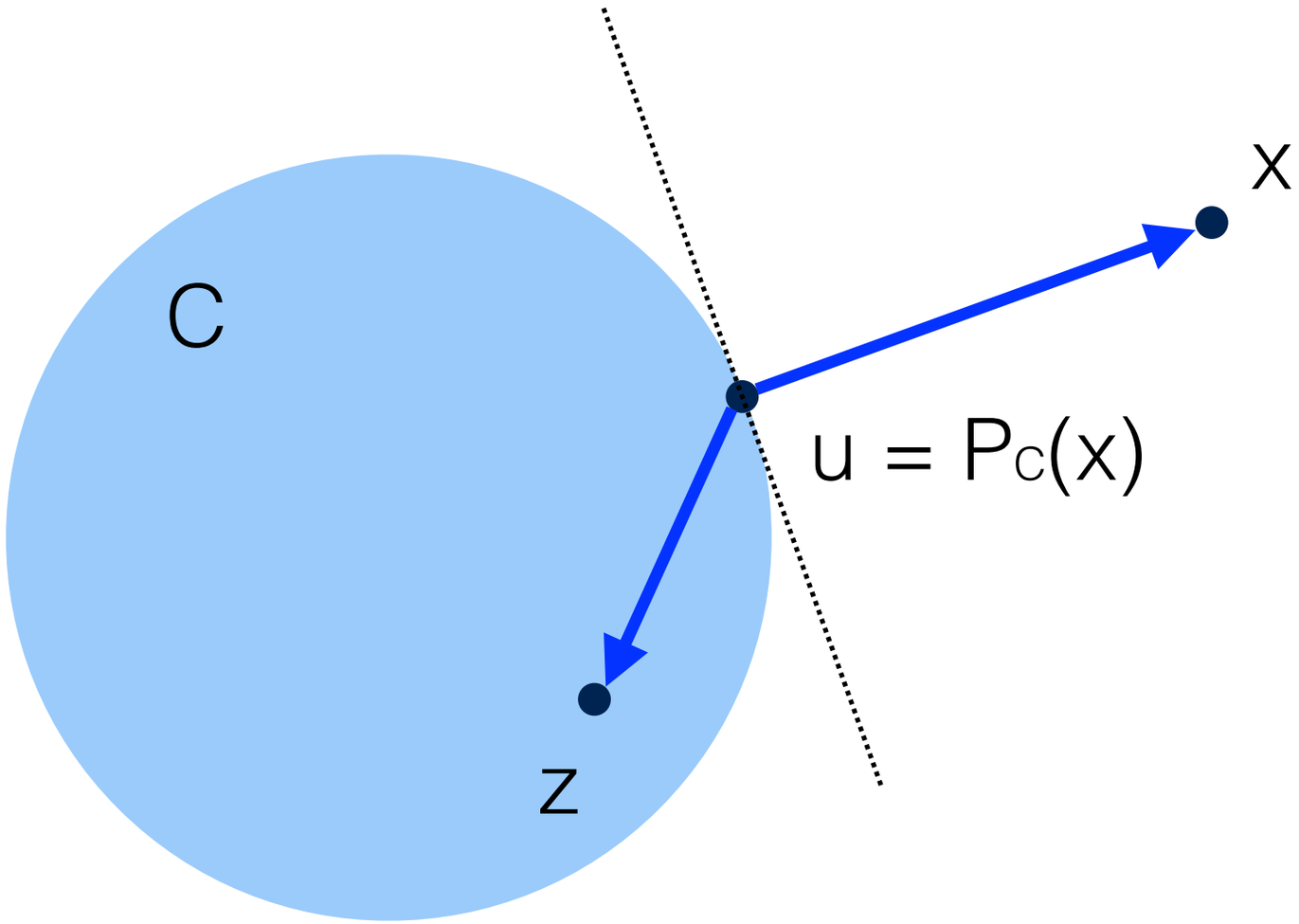
$$\partial \delta_C = N_C(x) = \{s : s^T(y - x) \leq 0, \forall y \in C\}$$

subgradient characterization

$$u = P_C(x) \iff (x - u)^T(z - u) \leq 0, \ \forall z \in C$$

Subgradient Characterization

$$u = P_C(x) \iff (x-u)^T(z-u) \leq 0,\ \forall z \in C$$



Proximal Mapping

The proximal mapping (prox-operator) of a convex function h is

$$\text{prox}_h(x) = \arg\min_u \left(h(u) + \frac{1}{2}\|u - x\|_2^2 \right)$$

- Proximal mapping is unique if it exists
- If h is closed, the proximal mapping exists.

subgradient characterization

from optimality conditions of minimization in the definition:

$$\begin{aligned} u = \text{prox}_h(x) &\iff x - u \in \partial h(u) \\ &\iff h(z) \geq h(u) + (x - u)^T (z - u) \quad \forall z \end{aligned}$$

Examples of Proximal Mappings

The proximal mapping (prox-operator) of a convex function h is

$$\text{prox}_h(x) = \arg\min_u \left(h(u) + \frac{1}{2}\|u - x\|_2^2 \right)$$

- $h(x) = 0$: $\text{prox}_h(x) = x$
- $h(x) = \delta_C(x)$ (indicator function of C): prox_h is projection onto C

$$\text{prox}_h(x) = \arg\min_{u \in C} \|u - x\|_2^2 = P_C(x)$$

- $h(x) = t\|x\|_1$: prox_h is the 'soft-threshold' (shrinkage) operation

$$[\text{prox}_{th}(x)]_i = x_i(1 - \min\{1, t/|x_i|\}), \quad i = 1, \ldots, n$$

Nonexpansiveness

if $u = \text{prox}_h(x), v = \text{prox}_h(y)$, then

$$(u - v)^T (x - y) \geq \|u - v\|_2^2$$

prox_h is firmly nonexpansive, or co-coercive with constant 1

- follows from subgradient characterization of proximal map and monotonicity of subgradient

$$x - u \in \partial h(u), y - v \in \partial h(v) \implies (x - u - y + v)^T (u - v) \geq 0$$

- implies

$$\|\text{prox}_h(x) - \text{prox}_h(y)\|_2 \leq \|x - y\|_2$$

prox_h is **nonexpansive** or **Lipschitz continuous** with constant 1

Proximal Gradient Method

unconstrained problem with cost function split in two components

$$\text{minimize } f(x) = g(x) + h(x)$$

- g convex, differentiable, with dom; $g = \mathbb{R}^n$
- h closed, convex, possibly nondifferentiable, $\text{prox}_h(x)$ is inexpensive

proximal gradient algorithm

$$x^{(k)} = \text{prox}_{t_k h}(x^{(k-1)} - t_k \nabla g(x^{(k-1)}))$$

$t_k > 0$ is step size, constant or determined by line search
Also known as forward-backward splitting

Interpretation

$$x^+ = \text{prox}_{th}(x - t\nabla g(x))$$

from definition of proximal operator:

$$x^+ = \arg\min_u \left(h(u) + \frac{1}{2t}\|u - x + t\nabla g(x)\|_2^2 \right)$$

$$= \arg\min_u \left(h(u) + g(x) + \nabla g(x)^T(u - x) + \frac{1}{2t}\|u - x\|_2^2 \right)$$

x^+ minimizes $h(u)$ plus a simple quadratic local model of $g(u)$ around x

Examples

$$\text{minimize } g(x) + h(x)$$

gradient descent: $h(x) = 0$, i.e. minimize $g(x)$

$$x^{(k)} = x^{(k-1)} - t_k \nabla g(x^{(k-1)})$$

projected gradient descent: $h(x) = \delta_C(x)$, i.e. minimize $g(x)$ over C

$$x^{(k)} = P_C\left(x^{(k-1)} - t_k \nabla g(x^{(k-1)})\right)$$

Nonnegative Least Squares: Projected Gradient

optimization problem:

$$\min_{b \in \mathbb{R}^p_+} f(b) := \frac{1}{2} \sum_{i=1}^{n} (y_i - x_i^T b)^2$$

projection onto nonnegative orthant: Let $C = \mathbb{R}_+$.

$$[P_C(b)]_i = \max\{b_i, 0\} = \begin{cases} b_i & b_i \geq 0 \\ 0 & \text{o.w.} \end{cases}$$

projected gradient descent:

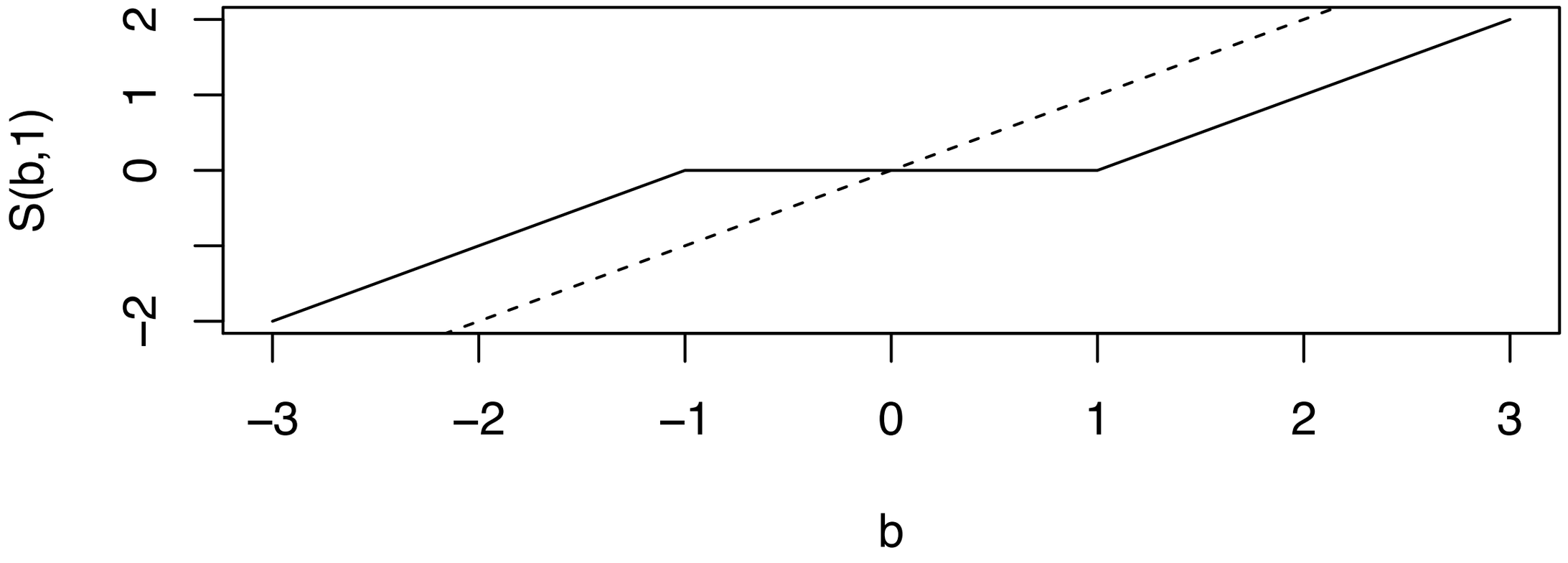
$$b^{(k)} = P_C \left(b^{(k-1)} - t_k X^T (X b^{(k-1)} - y) \right)$$

Proximal Mapping of 1-norm

$$h(b) = \|b\|_1$$

soft-thresholding function

$$[\text{prox}_{th}(b)]_i = S(b_i, \lambda) = \begin{cases} u_i - \lambda & u_i > \lambda \\ 0 & |u_i| \leq \lambda \\ u_i + \lambda & u_i < -\lambda \end{cases}$$



Lasso: Proximal Gradient

optimization problem:

$$\min_b f(b) := \frac{1}{2}\|y - Xb\|_2^2 + \lambda\|b\|_1$$

proximal mapping of 1-norm: soft-thresholding function

$$[\text{prox}_{th}(b)]_i = S(b_i, \lambda) = \begin{cases} u_i - \lambda & u_i > \lambda \\ 0 & |u_i| \leq \lambda \\ u_i + \lambda & u_i < -\lambda \end{cases}$$

proximal gradient descent: iterative soft-thresholding algorithm (ISTA)

$$b^{(k)} = S\left(b^{(k-1)} - t_k X^T(Xb^{(k-1)} - y), \lambda\right)$$

Proximal Mapping of nuclear-norm

$$\text{prox}_h(X) = \arg\min_U \left(h(U) + \frac{1}{2}\|U - X\|_F^2 \right)$$

$$h(X) = \|X\|_* = \sum_{i=1}^{r} \sigma_i$$

where σ_i are singular values of X

From Lecture 17

$$\text{minimize } \frac{1}{2}\|Y - Z\|_F^2 + \lambda\|Z\|_*$$

Solution is

$$Z = U\tilde{D}V^T$$

where $(U, D, V) = \text{SVD}(Y)$ and $\tilde{D} = S(D, \lambda)$

Singular Value Thresholding: Proximal Gradient

optimization problem:

$$\min_X f(X) := \frac{1}{2}\|P_\Omega(Y) - P_\Omega(X)\|_{\mathsf{F}}^2 + \lambda\|X\|_*$$

gradient:

$$\nabla\|P_\Omega(Y) - P_\Omega(X)\|_{\mathsf{F}}^2 = [P_\Omega(X) - P_\Omega(Y)]$$

proximal gradient descent: Singular Value Thresholding (SVT)

$$\begin{aligned} X^{(k+1/2)} &= X^{(k)} - t_k[P_\Omega(X^{(k)}) - P_\Omega(Y)] \\ (U, D, V) &= \mathsf{SVD}(X^{(k+1/2)}) \\ X^{(k+1)} &= US\,(D, \lambda)\,V^T \end{aligned}$$

Reduced Rank Regression: Proximal Gradient

optimization problem:

$$\min_X f(X) := \frac{1}{2}\|Y - XB\|_{\mathsf{F}}^2 + \lambda\|X\|_*$$

gradient:

$$\nabla\|Y - XB\|_{\mathsf{F}}^2 = X^T(XB - Y)$$

proximal gradient descent: Singular Value Thresholding (SVT)

$$\begin{aligned} B^{(k+1/2)} &= B^{(k)} - t_k X^T(XB^{(k)} - Y) \\ (U, D, V) &= \mathrm{SVD}(B^{(k+1/2)}) \\ B^{(k+1)} &= US\,(D, \lambda)\,V^T \end{aligned}$$

Convergence of proximal gradient method

to minimize $g + h$, choose $x^{(0)}$ and repeat

$$x^{(k)} = \text{prox}_{t_k h}(x^{(k-1)} - t_k \nabla g(x^{(k-1)})), \quad k \geq 1$$

assumptions

- g convex with dom $g = \mathbb{R}^n$; ∇g Lipschitz continuous with constant L:

$$\|\nabla g(x) - \nabla g(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y$$

- h is closed and convex (so that $\text{prox}_{th}(x)$ is well defined)
- optimal value $f^\star$ is finite and attained at some $x^\star$ (not necessarily unique)

convergence result: $1/k$ rate convergence with fixed step size $t_k = 1/L$

Fast (proximal) gradient methods

- Nesterov (1983, 1988, 2005): three gradient projection methods with $\mathcal{O}(1/k^2)$ convergence rate
- Beck & Teboulle (2008): FISTA, a proximal gradient version of Nesterov's 1983 method
- Nesterov (2004 book), Tseng (2008): overview and unified analysis of fast gradient methods
- several recent variations and extensions

this lecture: FISTA and Nesterov's 2nd method (1988) as presented by Tseng

Why should you care: Potentially lots of speed up for little additional code

FISTA (basic version)

$$\text{minimize } f(x) = g(x) + h(x)$$

- g is convex, differentiable, with dom $g = \mathbb{R}^n$
- h is closed, convex, with inexpensive prox_{th} operator

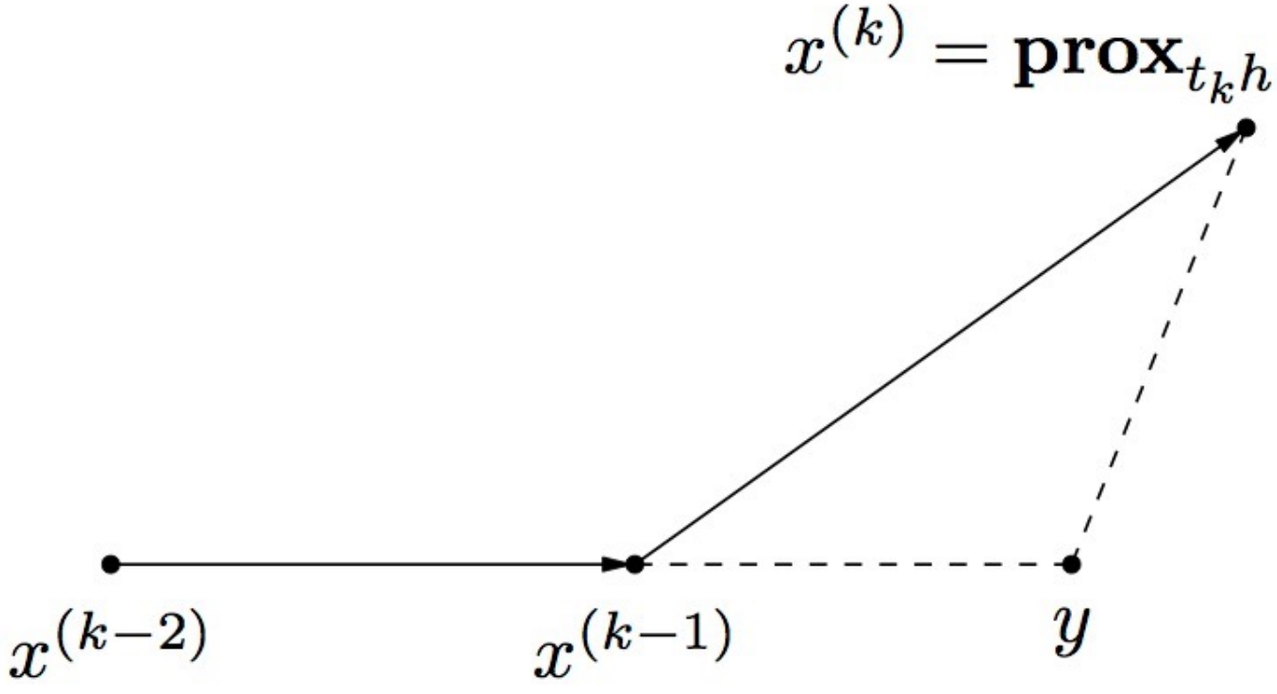
algorithm: choose any $x^{(0)} = x^{(-1)}$; for $k \geq 1$, repeat the steps

$$y = x^{(k-1)} + \frac{k-2}{k+1}\left[x^{(k-1)} - x^{(k-2)}\right]$$
$$x^{(k)} = \text{prox}_{t_k h}(y - t_k \nabla g(y))$$

- step size t_k fixed or determined by line search - acronym stands for 'Fast Iterative Shrinkage-Thresholding Algorithm'

Interpretation

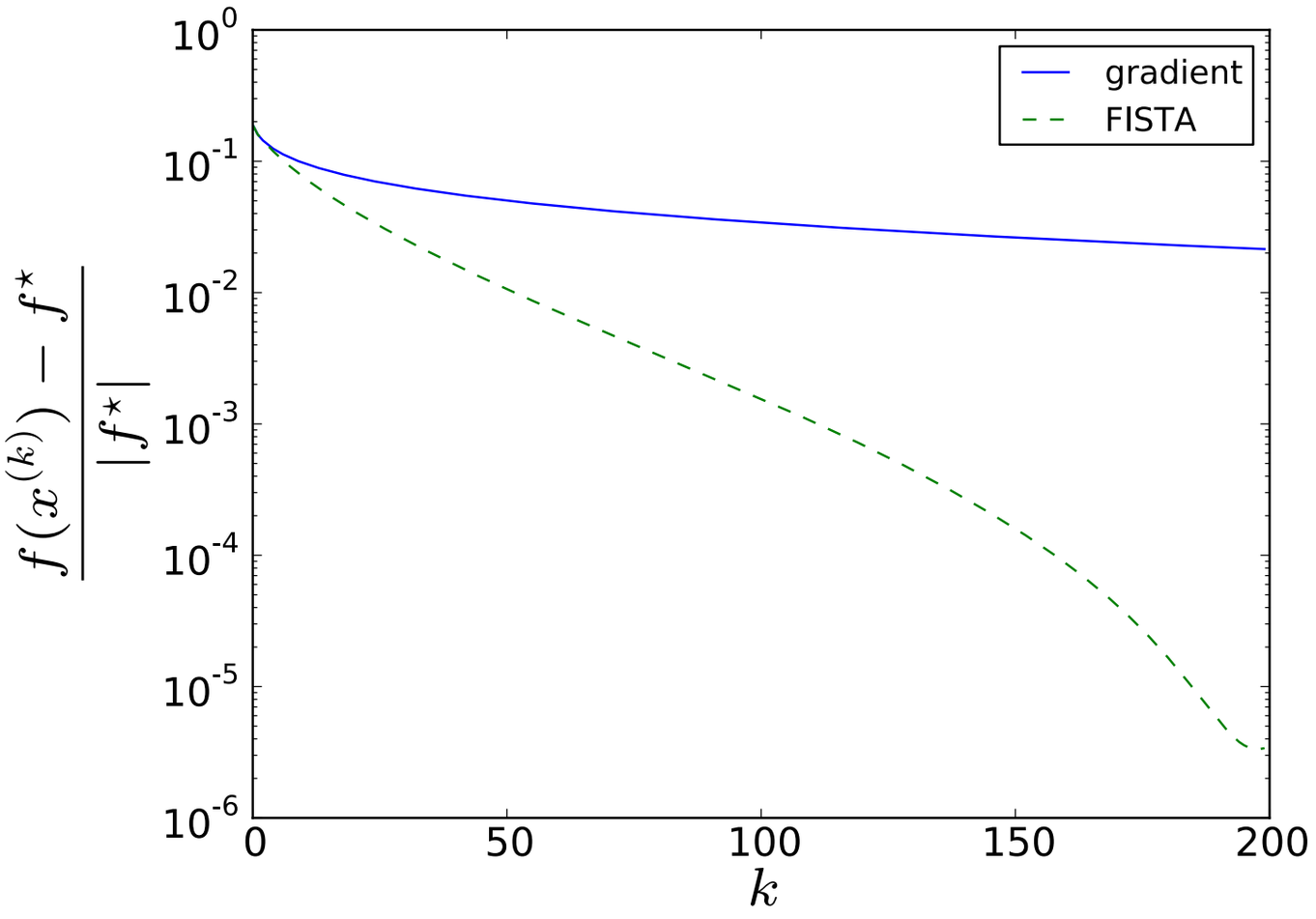
- first iteration ($k = 1$) is a proximal gradient step at $y = x^{(0)}$
- next iterations are proximal gradient steps at extrapolated points y



note: $x^{(k)}$ is feasible (in dom h); y may be outside dom h

Example

$$\text{minimize} \ \log\left(\sum_{i=1}^{n} \exp[x_i{}^T b - y_i]\right)$$

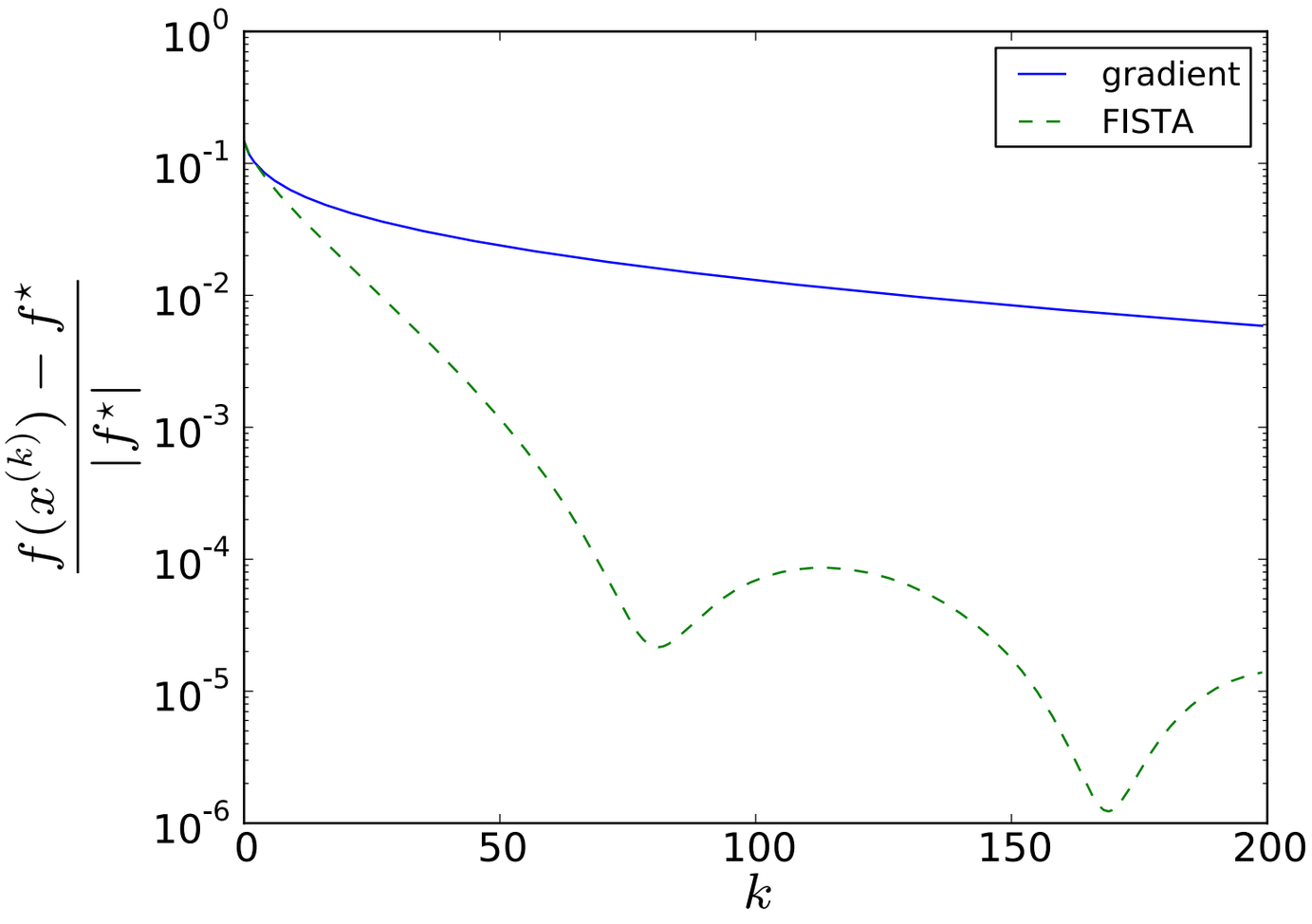


randomly generated data with $n = 2000, p = 1000$, same fixed step size

Example

$$\text{minimize } \log\left(\sum_{i=1}^{n} \exp[x_i{}^T b - y_i]\right)$$

Another instance



FISTA is **not** a descent method

Convergence of FISTA

assumptions

- g convex with $\operatorname{dom} g = \mathbb{R}^n$; ∇g is Lipschitz continuous with constant L:

$$\|\nabla g(x) - \nabla g(y)\|_2 \leq L\|x - y\|_2 \quad \forall x, y$$

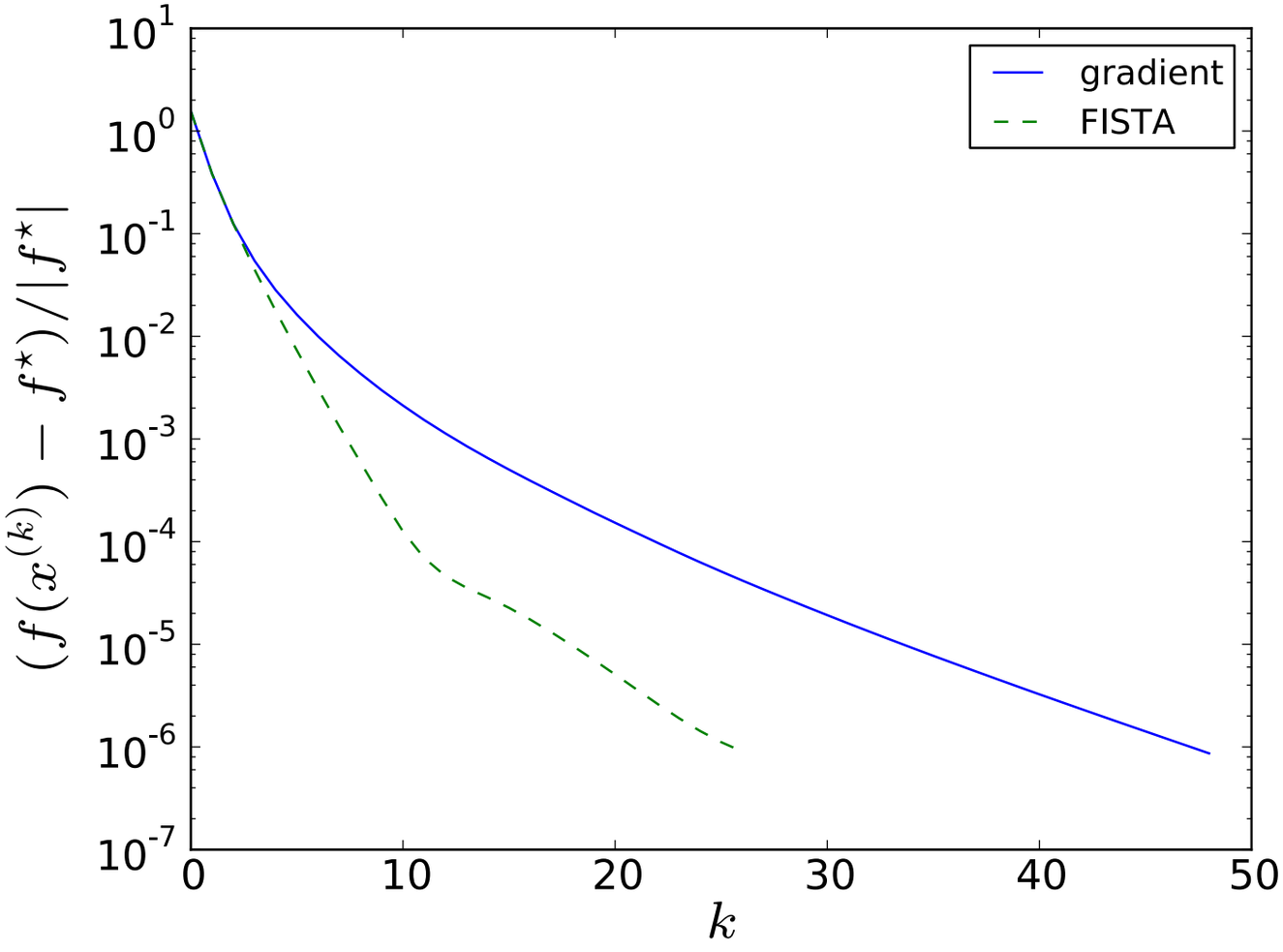
- h is closed and convex (so that $\operatorname{prox}_{th}(u)$ is well defined)
- optimal value $f^\star$ is finite and attained at $x^\star$ (not necessarily unique)

convergence result: $f(x^{(k)}) - f^\star$ decreases at least as fast as $1/k^2$

- with fixed step size $t_k = 1/L$
- with suitable line search

Example: quadratic program with box constraints

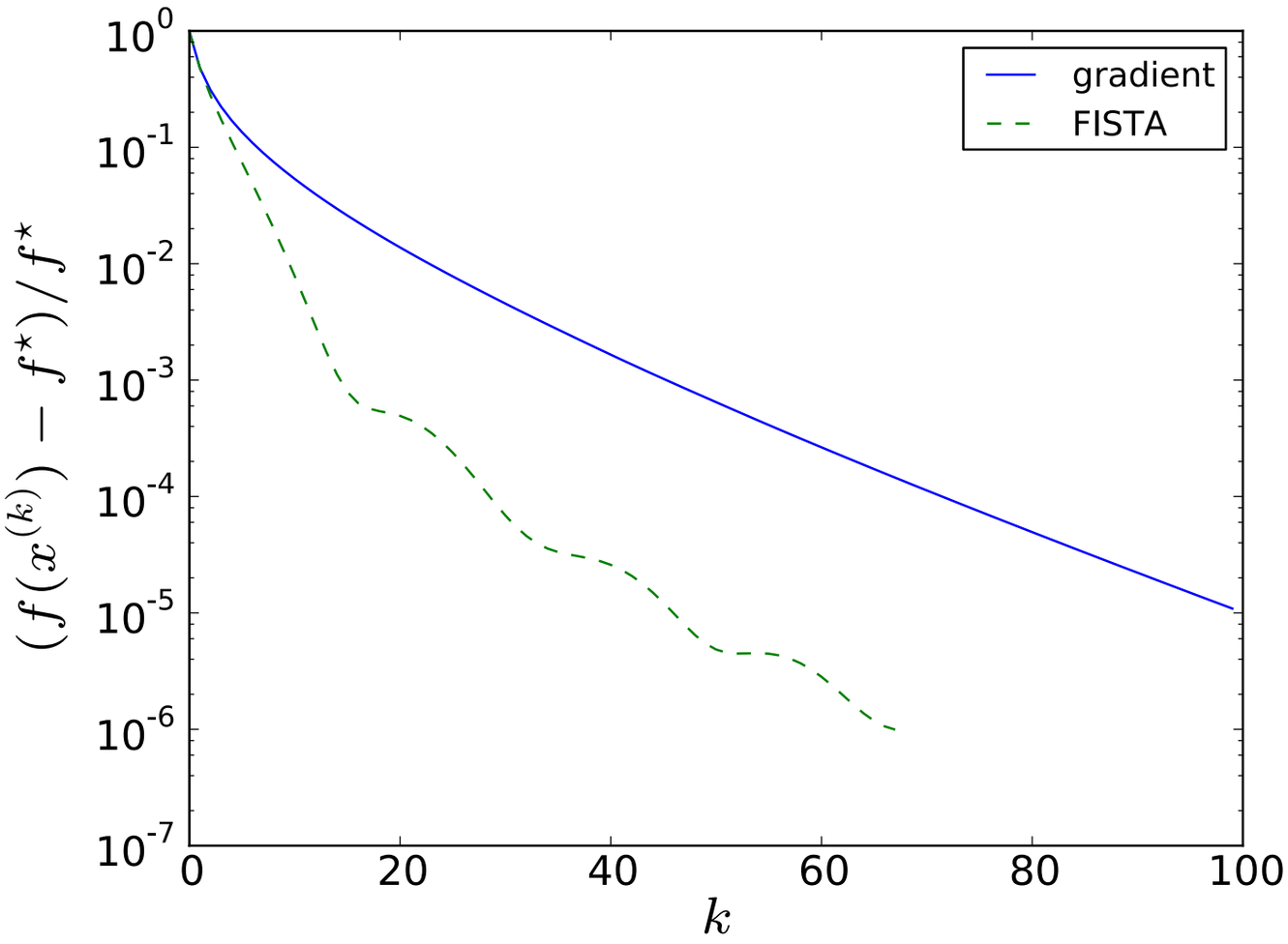
$$\text{minimize } (1/2)x^T Ax + b^T x$$
$$\text{subject to } 0 \leq x \leq 1$$



$n = 3000$; fixed step size $t = 1/\lambda_{\max}(A)$

Example: lasso

$$\text{minimize } \frac{1}{2}\|y - Xb\|_2^2 + \lambda\|b\|_1$$



randomly generated $X \in \mathbb{R}^{2000\times1000}$; step $t_k = 1/L$ with $L = \lambda_{\max}(A^T A)$

Why is ADMM useful?

- Easy to implement
- Easy to design distributed / parallel algorithms
- Get modestly accurate solutions in relatively few iterations (10-100s)
- Convergence with weak regularity conditions

Often a first algorithm when introducing new estimation procedures

- Robust PCA (Candes et al. 2009)
- Sparse PCA via Fantope projection (Vu et al. 2013)
- Lasso for hierarchical interactions (Bien et al. 2013)
- Joint graphical lasso (Danaher et al. 2014)
- Multivariate convex regression (Mazumder et al. 2015)
- etc.

Optimization Problem

$$\begin{aligned} &\text{minimize } f(x) + g(z) \\ &\text{subject to } Ax + Bz = c \end{aligned}$$

Some examples we care about:

Lasso:

$$\begin{aligned} &\text{minimize } \frac{1}{2}\|y - Xb\|_2^2 + \|z\|_1 \\ &\text{subject to } b - z = 0 \end{aligned}$$

Trend Filtering:

$$\begin{aligned} &\text{minimize } \frac{1}{2}\|y - b\|_2^2 + \|\theta\|_1 \\ &\text{subject to } Db - \theta = 0 \end{aligned}$$

Reformulating unconstrained problems this way is called **variable splitting**

Solve an Equivalent Optimization Problem

Original

$$\begin{aligned} &\text{minimize } f(x) + g(z) \\ &\text{subject to } Ax + Bz = c \end{aligned}$$

Equivalent

$$\begin{aligned} &\text{minimize } f(x) + g(z) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 \\ &\text{subject to } Ax + Bz = c \end{aligned}$$

Augmented Lagrangian Method (ALM)

$$\text{minimize } f(x) + g(z) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2$$

$$\text{subject to } Ax + Bz = c$$

Augmented Lagrangian

$$\mathcal{L}_\rho(x, z, y) = f(x) + g(z) + \langle y, Ax + Bz - c\rangle + \frac{\rho}{2}\|Ax + Bz - c\|_2^2$$

- $\mathcal{L}_0(x, z, y)$ is the Lagrangian for the original problem.

ALM

$$(x^{(m+1)}, z^{(m+1)}) = \arg\min_{x,z} \mathcal{L}_\rho(x, z, y^{(m)})$$

$$y^{(m+1)} = y^{(m)} + \rho(Ax^{(m+1)} + Bz^{(m+1)} - c)$$

- Also called **Method of Multipliers** or **Bregman iteration**

Augmented Lagrangian Method (ALM)

Augmented Lagrangian

$$\mathcal{L}_\rho(x, z, y) = f(x) + g(z) + \langle y, Ax + Bz - c\rangle + \frac{\rho}{2}\|Ax + Bz - c\|_2^2$$

Dual Function

$$\psi_D(y) = \inf_{x,z} \mathcal{L}_\rho(x, z, y)$$

Gradient of Dual

$$(x^\star, z^\star) = \arg\min_{x,z} \mathcal{L}_\rho(x, z, \nu)$$

$$\nabla\psi_D(y) = Ax^\star + Bz^\star - c$$

ALM = Gradient Ascent with fixed step size

$$(x^{(m+1)}, z^{(m+1)}) = \arg\min_{x,z} \mathcal{L}_\rho(x, z, y^{(m)})$$

$$y^{(m+1)} = y^{(m)} + \rho(Ax^{(m+1)} + Bz^{(m+1)} - c)$$

Augmented Lagrangian Method (ALM)

- More accurately, ALM is proximal point applied to the dual
- Alternative interpretation: gradient ascent with fixed step size on a (Moreau-Yosida) smoothed dual.

Issues with ALM and a Solution

Primal variable update can be hard

$$
\begin{aligned}
(x^{(m+1)}, z^{(m+1)}) &= \arg\min_{x,z} \mathcal{L}_\rho(x, z, y^{(m)}) \\
&= \arg\min_{x,z} f(x) + g(z) + \langle \nu, Ax + Bz - c \rangle \\
&\quad + \frac{\rho}{2} \|Ax + Bz - c\|_2^2
\end{aligned}
$$

Solution: Block coordinate descent on (x, z)

$$
\begin{aligned}
x^{(m+1)} &= \arg\min_{x} \mathcal{L}_\rho(x, z^{(m)}, y^{(m)}) \\
z^{(m+1)} &= \arg\min_{z} \mathcal{L}_\rho(x^{(m+1)}, z, y^{(m)}) \\
y^{(m+1)} &= y^{(m)} + \rho(Ax^{(m+1)} + Bz^{(m+1)} - c)
\end{aligned}
$$

This is the Alternating Direction Method of Multipliers (ADMM)

Scaled Form

residual: $r = Ax + Bz - c$

$$\begin{aligned} y^T r + \frac{\rho}{2}\|r\|_2^2 &= \frac{\rho}{2}\|r + \rho^{-1}y\|_2^2 - \frac{\rho}{2}\|\rho^{-1}y\|_2^2 \\ &= \frac{\rho}{2}\|r + u\|_2^2 - \frac{\rho}{2}\|u\|_2^2, \end{aligned}$$

scaled dual variable: $u = \rho^{-1}y$

ADMM using u

$$\begin{aligned} x^{(m+1)} &= \arg\min_x \left(f(x) + \frac{\rho}{2}\|Ax + Bz^{(m)} - c + u^{(m)}\|_2^2 \right) \\ z^{(m+1)} &= \arg\min_z \left(g(z) + \frac{\rho}{2}\|Ax^{(m+1)} + Bz - c + u^{(m)}\|_2^2 \right) \\ u^{(m+1)} &= u^{(m)} + (Ax^{(m+1)} + Bz^{(m+1)} - c) \end{aligned}$$

Convex Feasibility: Alternating Projections

Find a point in $C \cap D$

$$\text{minimize } \delta_C(x) + \delta_D(x)$$

Write a scaled form ADMM algorithm

$$x^{(k+1)} =$$

$$z^{(k+1)} =$$

$$u^{(k+1)} =$$

Convex Feasibility: Parallel Projections

Find a point in $\cap_{i=1}^{N} A_i$

$$C = A_1 \times \cdots \times A_N$$
$$D = \{(x_1, \ldots, x_N) : x_1 = x_2 = \cdots = x_N\}$$

$$\text{minimize } \delta_C(x) + \delta_D(x)$$

Write a scaled form ADMM algorithm

$$x^{(k+1)} =$$
$$z^{(k+1)} =$$
$$u^{(k+1)} =$$

Least Absolute Deviations

$$\begin{aligned} &\text{minimize } \|z\|_1 \\ &\text{subject to } Ax - z = b \end{aligned}$$

Write a scaled form ADMM algorithm

$$x^{(k+1)} =$$

$$z^{(k+1)} =$$

$$u^{(k+1)} =$$

Q: What is the computational complexity of a complete round of updates?

Basis Pursuit

$$\text{minimize } \delta_C(x) + \|z\|_1$$
$$\text{subject to } x - z = 0$$

$$C = \{x : Ax = b\}$$

$$P_C(x) = [I - A^T(AA^T)^{-1}A]x + A^T(AA^T)^{-1}b$$

Write a scaled form ADMM algorithm

$$x^{(k+1)} =$$
$$z^{(k+1)} =$$
$$u^{(k+1)} =$$

Q: What is the computational complexity of a complete round of updates?

Lasso

$$\begin{aligned} \text{minimize } & \frac{1}{2}\|b - Ax\|_2^2 + \lambda\|z\|_1 \\ \text{subject to } & x - z = 0 \end{aligned}$$

Write a scaled form ADMM algorithm

$$x^{(k+1)} =$$

$$z^{(k+1)} =$$

$$u^{(k+1)} =$$

Q: What is the computational complexity of a complete round of updates?

Generalized Lasso

$$\begin{aligned} \text{minimize } & \frac{1}{2}\|b - Ax\|_2^2 + \lambda\|z\|_1 \\ \text{subject to } & Dx - z = 0 \end{aligned}$$

Write a scaled form ADMM algorithm

$$x^{(k+1)} =$$

$$z^{(k+1)} =$$

$$u^{(k+1)} =$$

Q: What is the computational complexity of a complete round of updates?

Proximal Mapping of $-\log\det X$

Let $X \in S^n_{++}$.

$$h(X) = -\log\det X$$

$$[\text{prox}_{th}(W)]_i = \frac{\lambda_i + \sqrt{\lambda_i^2 + 4t}}{2} \qquad \text{for } i = 1, \ldots, n$$

where W has eigendecomposition $Q\Lambda Q^T$ and λ_i is the ith eigenvalue.

Sparse Inverse Covariance

$$\begin{aligned} &\text{minimize } \operatorname{tr}(SX) - \log\det X + \lambda\|Z\|_1 \\ &\text{subject to } X - Z = 0 \end{aligned}$$

$\|Z\|_1 = \sum_{i \neq j} |z_{ij}|$

Write a scaled form ADMM algorithm

$$X^{(k+1)} =$$

$$Z^{(k+1)} =$$

$$U^{(k+1)} =$$

Q: What is the computational complexity of a complete round of updates?

Distributed Computing

Use ADMM to break a big data problem into smaller data problems

- Consensus
- Sharing

See “Distributed Optimization and Statistical learning via the ADMM" for tutorial on how to implment ADMM using

- MPI
- Hadoop / MapReduce

Boyd’s website also has sample code.

Convergence of ADMM

Optimization Problem

$$\begin{aligned} &\text{minimize } f(x) + g(z) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 \\ &\text{subject to } Ax + Bz = c \end{aligned}$$

Augmented Lagrangian

$$\mathcal{L}_\rho(x, z, y) = f(x) + g(z) + \langle y, Ax + Bz - c\rangle + \frac{\rho}{2}\|Ax + Bz - c\|_2^2$$

Assumption 1: The (extended-real-valued) functions $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ are closed, proper, and convex.

Convergence of ADMM

Optimization Problem

$$\begin{aligned} &\text{minimize } f(x) + g(z) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 \\ &\text{subject to } Ax + Bz = c \end{aligned}$$

Augmented Lagrangian

$$\mathcal{L}_\rho(x, z, y) = f(x) + g(z) + \langle y, Ax + Bz - c\rangle + \frac{\rho}{2}\|Ax + Bz - c\|_2^2$$

Assumption 2: The unaugmented Lagrangian $\mathcal{L}_0$ has a saddle point, namely there exists $(x^\star, z^\star, y^\star)$, non necessarily unique, such that

$$\mathcal{L}_0(x^\star, z^\star, y) \leq \mathcal{L}_0(x^\star, z^\star, y^\star) \leq \mathcal{L}_0(x, z, y^\star) \quad \forall x, z, y.$$

Convergence of ADMM

Proposition: Under assumptions 1 and 2, the ADMM iterates satisfy

- **Residual convergence.** $r^k = Ax^{(k)} + Bz(k) - c \to 0$ as $k \to \infty$
- **Objective convergence.** $f(x^{(k)}) + g(z^{(k)}) \to p^\star$ as $k \to \infty$
- **Dual variable convergence.** $y^{(k)} \to y^\star$ as $k \to \infty$.

Proof: See "Distributed Optimization and Statistical learning via the ADMM"

Note: Neither $x^{(k)}$ nor $z^{(k)}$ need converge to optimal values. Additional assumptions are needed to guarantee this.

Other Useful References

Wotao Yin @ UCLA Math

- Self Equivalence of the Alternating Direction Method of Multipliers
- On the linear convergence of the ADMM in decentralized consensus optimization
- Faster convergence rates of relaxed Peaceman-Rachford and ADMM under regularity assumptions
- Parallel Multi-Block ADMM with $o(1/k)$ Convergence
- etc.