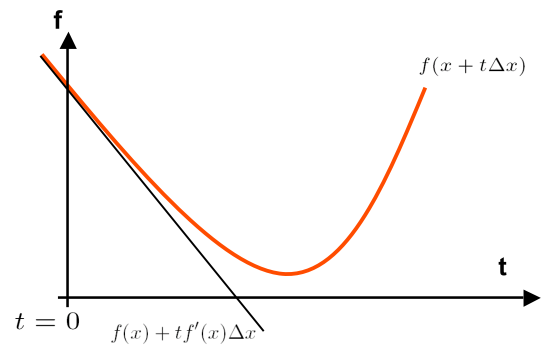
Lecture 12: convergence

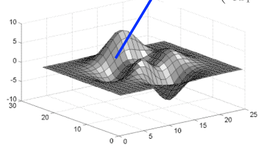
- More about multivariable calculus
- Descent methods
- Backtracking line search
- More about convexity (first and second order)
- Newton step
- Example 1: linear programming (one var., one constr.)
- Example 2: linear programming (one var., two constr.)
- Example 3: linear programming (two var., one constr.)
- Example 4: linear programming (N var., M constr.)

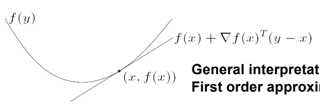
Derivative (one variable)



Derivative, i.e. gradient (multiple variables)

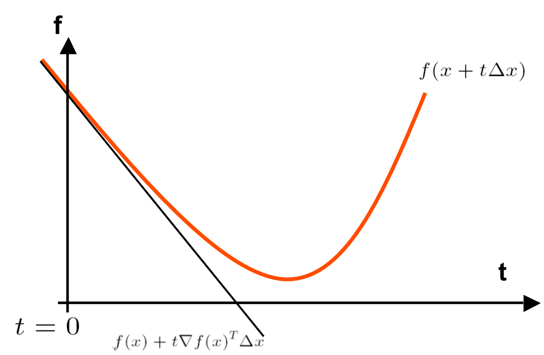
$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n}\right)$$





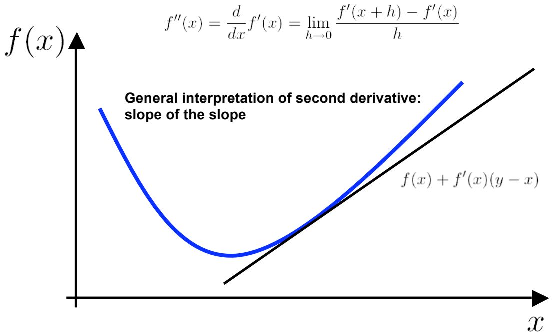
General interpretation of derivative (gradient): First order approximation of the function (affine)

Derivative



Second derivative

$$f''(x) = \frac{d}{dx} f'(x) = \lim_{h \to 0} \frac{f'(x+h) - f'(x)}{h}$$



General interpretation of second derivative: slope of the slope

Hessian matrix (multiple variables)

What if function has more than one variable?

f is **twice differentiable** if $\mathbf{dom}\, f$ is open and the Hessian $\nabla^2 f(x) \in \mathbf{S}^n$,

$$\nabla^2 f(x)_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}, \quad i, j = 1, \ldots, n,$$

Example:

$f(x, y) = x^2 + xy + y^2$

$$\nabla^2 f(x, y) = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

$\frac{\partial f}{\partial x} = 2x + y$ $\qquad$ $\frac{\partial f}{\partial y} = 2y + x$

$\frac{\partial^2 f}{\partial x^2} = 2$ $\quad$ $\frac{\partial^2 f}{\partial y \partial x} = 1$ $\quad$ $\frac{\partial^2 f}{\partial y^2} = 2$ $\quad$ $\frac{\partial^2 f}{\partial x \partial y} = 1$

Descent methods, convex functions (reminder)

$$x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)} \quad \text{with } f(x^{(k+1)}) < f(x^{(k)})$$

- other notations: $x^+ = x + t\Delta x$, $x := x + t\Delta x$
- Δx is the *step*, or *search direction*; t is the *step size*, or *step length*
- from convexity, $f(x^+) < f(x)$ implies $\nabla f(x)^T \Delta x < 0$ ($i.e.$, Δx is a *descent direction*)

General descent method.

given a starting point $x \in \textbf{dom}\, f$.
repeat
1. Determine a descent direction Δx.
2. *Line search.* Choose a step size $t > 0$.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

[S. Boyd, L. Vandenberghe, Convex Convex Optimization lect. Notes, Stanford Univ. 2004]

Backtracking methods

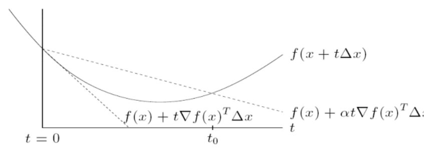
exact line search: $t = \text{argmin}_{t>0} f(x + t\Delta x)$

backtracking line search (with parameters $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$)

- starting at $t = 1$, repeat $t := \beta t$ until

$$f(x + t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$$

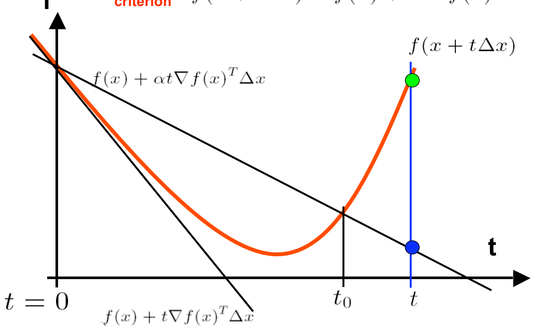
- graphical interpretation: backtrack until $t \le t_0$



[S. Boyd, L. Vandenberghe, Convex Convex Optimization lect. Notes, Stanford Univ. 2004]

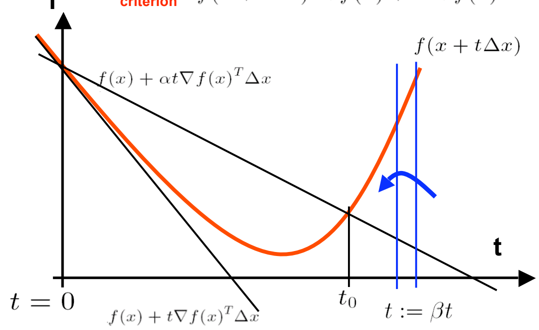
Backtracking methods

Stopping criterion $f(x + t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$



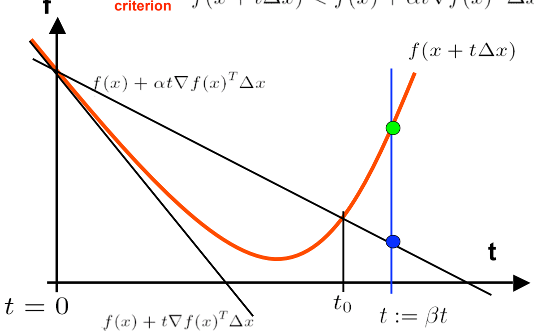
Backtracking methods

Stopping criterion $f(x + t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$



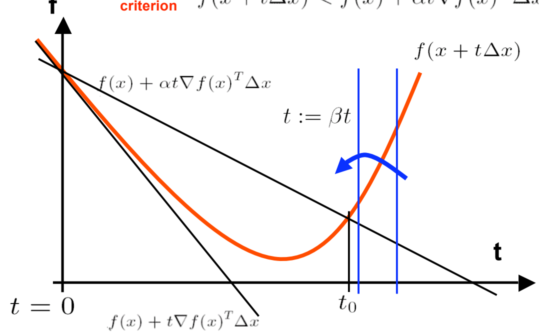
Backtracking methods

Stopping criterion $f(x + t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$



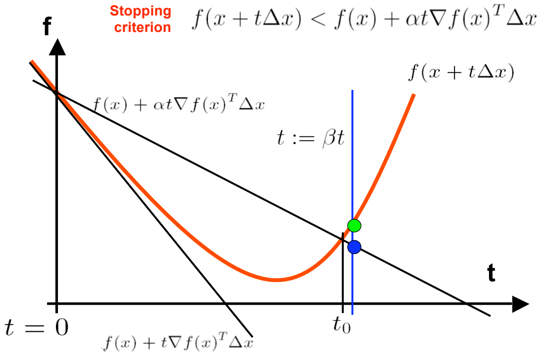
Backtracking methods

Stopping criterion $f(x + t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$



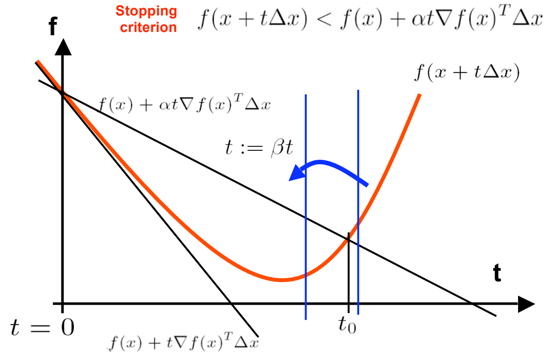
Backtracking methods

Stopping criterion $f(x+t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$



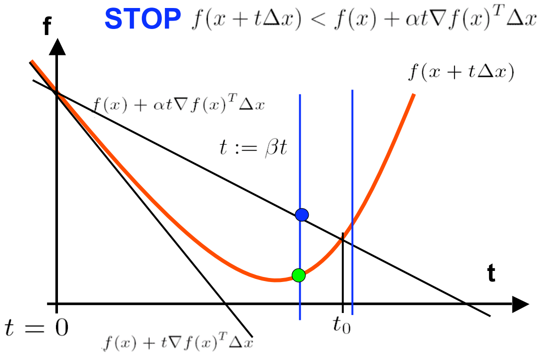
Backtracking methods

Stopping criterion $f(x+t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$



Backtracking methods

STOP $f(x+t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$



Backtracking methods

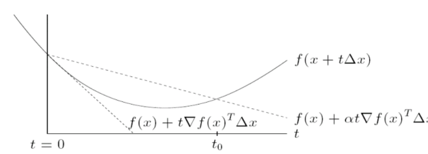
exact line search: $t = \operatorname{argmin}_{t>0} f(x + t\Delta x)$

backtracking line search (with parameters $\alpha \in (0, 1/2)$, $\beta \in (0,1)$)

- starting at $t = 1$, repeat $t := \beta t$ until

$$f(x+t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$$

- graphical interpretation: backtrack until $t \le t_0$



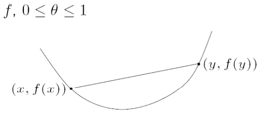
[S. Boyd, L. Vandenberghe, Convex Convex Optimization lect. Notes, Stanford Univ. 2004]

Convex functions: reminder

$f : \mathbf{R}^n \to \mathbf{R}$ is convex if $\mathbf{dom}\, f$ is a convex set and

$$f(\theta x + (1-\theta)y) \le \theta f(x) + (1-\theta) f(y)$$

for all $x, y \in \mathbf{dom}\, f$, $0 \le \theta \le 1$



- f is concave if $-f$ is convex
- f is strictly convex if $\mathbf{dom}\, f$ is convex and

$$f(\theta x + (1-\theta)y) < \theta f(x) + (1-\theta) f(y)$$

for $x, y \in \mathbf{dom}\, f$, $x \ne y$, $0 < \theta < 1$

[S. Boyd, L. Vandenberghe, Convex Convex Optimization lect. Notes, Stanford Univ. 2004]

First order conditions

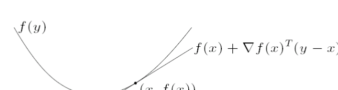
f is **differentiable** if $\mathbf{dom}\, f$ is open and the gradient

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n} \right)$$

exists at each $x \in \mathbf{dom}\, f$

1st-order condition: differentiable f with convex domain is convex iff

$$f(y) \ge f(x) + \nabla f(x)^T (y - x) \quad \text{for all } x, y \in \mathbf{dom}\, f$$



first-order approximation of f is global underestimator

[S. Boyd, L. Vandenberghe, Convex Convex Optimization lect. Notes, Stanford Univ. 2004]

Second order conditions

f is **twice differentiable** if $\mathbf{dom}\, f$ is open and the Hessian $\nabla^2 f(x) \in \mathbf{S}^n$,

$$\nabla^2 f(x)_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}, \quad i, j = 1, \ldots, n,$$

exists at each $x \in \mathbf{dom}\, f$

2nd-order conditions: for twice differentiable f with convex domain

- f is convex if and only if

$$\nabla^2 f(x) \succeq 0 \quad \text{for all } x \in \mathbf{dom}\, f$$

- if $\nabla^2 f(x) \succ 0$ for all $x \in \mathbf{dom}\, f$, then f is strictly convex

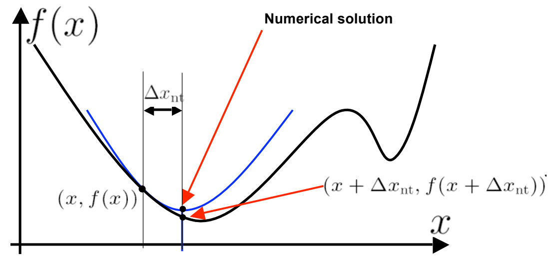
[S. Boyd, L. Vandenberghe, Convex Convex Optimization lect. Notes, Stanford Univ. 2004]

Newton step

Quadratic approximation of a function

$$\hat{f}(x+v) = f(x) + f'(x)v + \tfrac{1}{2}f''(x)v^2$$

Graphical interpretation

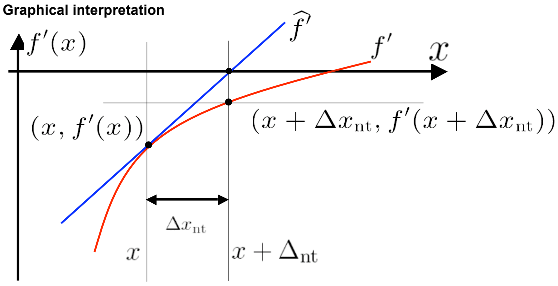


Newton step

Quadratic approximation of a function

$$\hat{f}(x+v) = f(x) + f'(x)v + \tfrac{1}{2}f''(x)v^2$$

Graphical interpretation



Newton step

Quadratic approximation of a function

$$\hat{f}(x+v) = f(x) + f'(x)v + \tfrac{1}{2}f''(x)v^2$$

Find the minimum of $\hat{f}(x+v)$ **with respect to** v

$$\hat{f}'(x+v) = f'(x) + v f''(x)$$

$$\hat{f}'(x+v) = 0 \Leftrightarrow v = -\frac{f'(x)}{f''(x)}$$

Newton step: $\Delta x_{nt} = -\dfrac{f'(x)}{f''(x)}$

Newton step (more than one dimension)

$$\Delta x_{\rm nt} = -\nabla^2 f(x)^{-1} \nabla f(x)$$

interpretations

- $x + \Delta x_{\rm nt}$ minimizes second order approximation

$$\hat{f}(x+v) = f(x) + \nabla f(x)^T v + \frac{1}{2} v^T \nabla^2 f(x) v$$

- $x + \Delta x_{\rm nt}$ solves linearized optimality condition

$$\nabla f(x+v) \approx \nabla \hat{f}(x+v) = \nabla f(x) + \nabla^2 f(x) v = 0$$

[S. Boyd, L. Vandenberghe, Convex Convex Optimization lect. Notes, Stanford Univ. 2004]

Newton step descent algorithm

General algorithm:

given a starting point $x \in \mathbf{dom}\, f$, tolerance $\epsilon > 0$.
repeat

1. *Compute the Newton step and decrement.*
   $\Delta x_{\rm nt} := -\nabla^2 f(x)^{-1} \nabla f(x); \quad \lambda^2 := \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)$
2. *Stopping criterion.* **quit** if $\lambda^2/2 \leq \epsilon$.
3. *Line search.* Choose step size t by backtracking line search.
4. *Update.* $x := x + t\Delta x_{\rm nt}$.

Application: linear programming

Back to linear programming: how would you solve a linear program with interior point methods?

$$\begin{array}{ll}\textbf{min:} & \mathbf{c}^{\mathbf{T}} \cdot \mathbf{x} \\ \textbf{s.t.} & \mathbf{A}\ \mathbf{x} \leq \mathbf{b}\end{array}$$

Instantiation of all the constraints:

$$\begin{array}{lll}\textbf{min:} & c_1x_1 + c_2x_2 + \cdots + c_Nc_N & \\ \textbf{s.t.:} & a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,j}x_j \cdots + a_{1,N}x_N & \leq b_1 \\ & a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,j}x_j \cdots + a_{2,N}x_N & \leq b_2 \\ & \vdots & \vdots \\ & a_{M,1}x_1 + a_{M,2}x_2 + \cdots + a_{M,j}x_j \cdots + a_{M,N}x_N & \leq b_M\end{array}$$

Application: linear programming

Back to linear programming: how would you solve a linear program with interior point methods?

$$\begin{array}{ll}\textbf{min:} & \mathbf{c}^{\mathbf{T}} \cdot \mathbf{x} \\ \textbf{s.t.} & \mathbf{A}\ \mathbf{x} \leq \mathbf{b}\end{array}$$

Instantiation of all the constraints:

$$\begin{array}{lll}\textbf{min:} & c_1x_1 + c_2x_2 + \cdots + c_Nc_N & \\ \textbf{s.t.:} & b_1 - (a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,j}x_j \cdots + a_{1,N}x_N) & \geq 0 \\ & b_2 - (a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,j}x_j \cdots + a_{2,N}x_N) & \geq 0 \\ & \vdots & \vdots \\ & b_M - (a_{M,1}x_1 + a_{M,2}x_2 + \cdots + a_{M,j}x_j \cdots + a_{M,N}x_N) & \geq 0\end{array}$$

Linear programming (one variable, one constraint)

Example: one constraint $\begin{array}{ll}\textbf{min:} & c \cdot x \\ \textbf{s.t.} & ax \leq b\end{array}$

Rewrite the constraint $\begin{array}{ll}\textbf{min:} & c \cdot x \\ \textbf{s.t.} & b - ax \geq 0\end{array}$

Add logarithmic barrier $\begin{array}{ll}\textbf{min:} & c \cdot x - \varepsilon \log(b - ax) \\ \textbf{s.t.} & \text{no constraints}\end{array}$

Solve the unconstrained control problem:

$$f'(x) = c + \frac{a}{b - ax}$$

Linear programming (one variable, two constraints)

Example: two constraints $\begin{array}{ll}\textbf{min:} & c \cdot x \\ \textbf{s.t.} & ax \leq b \\ & dx \leq e\end{array}$

Rewrite the constraints $\begin{array}{ll}\textbf{min:} & c \cdot x \\ \textbf{s.t.} & e - dx \geq 0 \\ & b - ax \geq 0\end{array}$

Add logarithmic barrier

$$\begin{array}{ll}\textbf{min:} & c \cdot x - \varepsilon \log(b - ax) - \varepsilon \log(e - dx) \\ \textbf{s.t.} & \text{no constraints}\end{array}$$

Solve the unconstrained control problem:

$$f'(x) = c + \varepsilon \frac{a}{b - ax} + \varepsilon \frac{d}{e - dx}$$

Linear programming (two variables, two constraints)

Example: two variables /two constraints $\begin{array}{ll}\textbf{min:} & \alpha x + \beta y \\ \textbf{s.t.} & \gamma x \leq \delta \\ & \zeta y \leq \xi\end{array}$

Rewrite the constraints $\begin{array}{ll}\textbf{min:} & \alpha x + \beta y \\ \textbf{s.t.} & \delta - \gamma x \geq 0 \\ & \xi - \zeta y \geq 0\end{array}$

Add logarithmic barrier

$$\begin{array}{ll}\textbf{min:} & \alpha x + \beta y - \varepsilon \log(\delta - \gamma x) - \varepsilon \log(\xi - \zeta y) \\ \textbf{s.t.} & \text{no constraints}\end{array}$$

Solve the unconstrained control problem:

$$\nabla f(x, y) = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + \varepsilon \begin{pmatrix} \frac{\gamma}{\delta - \gamma x} \\ \frac{\zeta}{\xi - \zeta y} \end{pmatrix}$$

Linear programming (two variables, one constraints)

Example: two variables /two constraints $\begin{array}{ll}\textbf{min:} & \alpha x + \beta y \\ \textbf{s.t.} & \gamma x + \delta y \leq \mu\end{array}$

Add logarithmic barrier

$$\begin{array}{ll}\textbf{min:} & \alpha x + \beta y - \varepsilon \log(\mu - (\gamma x + \delta y)) \\ \textbf{s.t.} & \text{no constraints}\end{array}$$

Solve the unconstrained control problem:

$$\nabla f(x, y) = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + \varepsilon \begin{pmatrix} \frac{\gamma}{\mu - (\gamma x + \delta y)} \\ \frac{\delta}{\mu - (\gamma x + \delta y)} \end{pmatrix}$$

Linear programming (N variables, M constraints)

Example: two variables /two constraints

$$\begin{array}{lll} \textbf{min:} & c_1x_1 + c_2x_2 + \cdots + c_Nc_N & \\ \textbf{s.t.:} & a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,j}x_j \cdots + a_{1,N}x_N & \leq b_1 \\ & a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,j}x_j \cdots + a_{2,N}x_N & \leq b_2 \\ & \vdots & \vdots \\ & a_{M,1}x_1 + a_{M,2}x_2 + \cdots + a_{M,j}x_j \cdots + a_{M,N}x_N & \leq b_M \end{array}$$

Rewrite the constraints

$$\begin{array}{lll} \textbf{min:} & c_1x_1 + c_2x_2 + \cdots + c_Nc_N & \\ \textbf{s.t.:} & b_1 - (a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,j}x_j \cdots + a_{1,N}x_N) & \geq 0 \\ & b_2 - (a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,j}x_j \cdots + a_{2,N}x_N) & \geq 0 \\ & \vdots & \vdots \\ & b_M - (a_{M,1}x_1 + a_{M,2}x_2 + \cdots + a_{M,j}x_j \cdots + a_{M,N}x_N) & \geq 0 \end{array}$$

Linear programming (N variables, M constraints)

Rewrite the constraints

$$\begin{array}{lll} \textbf{min:} & c_1x_1 + c_2x_2 + \cdots + c_Nc_N & \\ \textbf{s.t.:} & b_1 - (a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,j}x_j \cdots + a_{1,N}x_N) & \geq 0 \\ & b_2 - (a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,j}x_j \cdots + a_{2,N}x_N) & \geq 0 \\ & \vdots & \vdots \\ & b_M - (a_{M,1}x_1 + a_{M,2}x_2 + \cdots + a_{M,j}x_j \cdots + a_{M,N}x_N) & \geq 0 \end{array}$$

Add logarithmic barrier:

$$\begin{array}{lll} \textbf{min:} & c_1x_1 + c_2x_2 + \cdots + c_Nc_N & \\ & +\varepsilon \log(b_1 - (a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,j}x_j \cdots + a_{1,N}x_N)) & \geq 0 \\ & +\varepsilon \log(b_2 - (a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,j}x_j \cdots + a_{2,N}x_N)) & \geq 0 \\ & \vdots & \vdots \\ & +\varepsilon \log(b_M - (a_{M,1}x_1 + a_{M,2}x_2 + \cdots + a_{M,j}x_j \cdots + a_{M,N}x_N)) & \geq 0 \\ \textbf{s.t.} & \text{no constraints} & \end{array}$$

Linear programming (N variables, M constraints)

Add logarithmic barrier:

$$\begin{array}{lll} \textbf{min:} & c_1x_1 + c_2x_2 + \cdots + c_Nc_N & \\ & +\varepsilon \log(b_1 - (a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,j}x_j \cdots + a_{1,N}x_N)) & \geq 0 \\ & +\varepsilon \log(b_2 - (a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,j}x_j \cdots + a_{2,N}x_N)) & \geq 0 \\ & \vdots & \vdots \\ & +\varepsilon \log(b_M - (a_{M,1}x_1 + a_{M,2}x_2 + \cdots + a_{M,j}x_j \cdots + a_{M,N}x_N)) & \geq 0 \\ \textbf{s.t.} & \text{no constraints} & \end{array}$$

Gradient:

$$\nabla f(x_1, x_2, \cdots, x_N) = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix}$$

Linear programming (N variables, M constraints)

Gradient:

$$\nabla f(x_1, x_2, \cdots, x_N) = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix}$$

Components of the gradient:

$$\begin{array}{rl} v_1 = & \varepsilon \cdot \frac{a_{1,1}}{b_1 - (a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,j}x_j \cdots + a_{1,N}x_N)} \\ + & \varepsilon \cdot \frac{a_{2,1}}{b_2 - (a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,j}x_j \cdots + a_{2,N}x_N)} \\ + & \vdots \\ + & \varepsilon \cdot \frac{a_{M,1}}{b_M - (a_{M,1}x_1 + a_{M,2}x_2 + \cdots + a_{M,j}x_j \cdots + a_{M,N}x_N)} \end{array}$$

Linear programming (N variables, M constraints)

Gradient:

$$\nabla f(x_1, x_2, \cdots, x_N) = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix}$$

Components of the gradient:

$$\begin{array}{rl} v_2 = & \varepsilon \cdot \frac{a_{1,2}}{b_1 - (a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,j}x_j \cdots + a_{1,N}x_N)} \\ + & \varepsilon \cdot \frac{a_{2,2}}{b_2 - (a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,j}x_j \cdots + a_{2,N}x_N)} \\ + & \vdots \\ + & \varepsilon \cdot \frac{a_{M,2}}{b_M - (a_{M,1}x_1 + a_{M,2}x_2 + \cdots + a_{M,j}x_j \cdots + a_{M,N}x_N)} \end{array}$$

Linear programming (N variables, M constraints)

Gradient:

$$\nabla f(x_1, x_2, \cdots, x_N) = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix}$$

Components of the gradient:

$$\begin{array}{rl} v_i = & \varepsilon \cdot \frac{a_{1,i}}{b_1 - (a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,j}x_j \cdots + a_{1,N}x_N)} \\ + & \varepsilon \cdot \frac{a_{2,i}}{b_2 - (a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,j}x_j \cdots + a_{2,N}x_N)} \\ + & \vdots \\ + & \varepsilon \cdot \frac{a_{M,i}}{b_M - (a_{M,1}x_1 + a_{M,2}x_2 + \cdots + a_{M,j}x_j \cdots + a_{M,N}x_N)} \end{array}$$