Lecture 12

Lecture 12: Feasible direction methods

Kin Cheong Sou
December 2, 2013

- Consider the problem to find

$$f^* = \text{infimum } f(x), \tag{1a}$$
$$\text{subject to } x \in X, \tag{1b}$$

$X \subseteq \mathbb{R}^n$ nonempty, closed and convex; $f : \mathbb{R}^n \to \mathbb{R}$ is C^1 on X

- A natural idea is to mimic the line search methods for unconstrained problems.

- However, most methods for (1) manipulate (that is, relax) the constraints defining X; in some cases even such that the sequence $\{x_k\}$ is infeasible until convergence. Why?

- Consider a constraint "$g_i(x) \leq b_i$," where g_i is nonlinear
- Checking whether p is a feasible direction at x, or what the maximum feasible step from x in the direction of p is, is very difficult
- For which step length $\alpha > 0$ is $g_i(x + \alpha p) = b_i$? This is a nonlinear equation in α!
- Assuming that X is polyhedral, these problems are not present
- Note: KKT always necessary for a local min for polyhedral sets; methods will find such points

Step 0. Determine a *starting point* $x_0 \in \mathbb{R}^n$ such that $x_0 \in X$. Set $k := 0$

Step 1. Determine a *search direction* $p_k \in \mathbb{R}^n$ such that p_k is a feasible descent direction

Step 2. Determine a *step length* $\alpha_k > 0$ such that $f(x_k + \alpha_k p_k) < f(x_k)$ and $x_k + \alpha_k p_k \in X$

Step 3. Let $x_{k+1} := x_k + \alpha_k p_k$

Step 4. If a *termination criterion* is fulfilled, then stop! Otherwise, let $k := k + 1$ and go to Step 1

- Similar form as the general method for unconstrained optimization
- Just as *local* as methods for unconstrained optimization
- Search directions typically based on the approximation of f—a "relaxation"
- Search direction often of the form $p_k = y_k - x_k$, where $y_k \in X$ solves an approximate problem
- Line searches similar; note the maximum step
- Termination criteria and descent based on first-order optimality and/or fixed-point theory ($p_k \approx \mathbf{0}^n$)

- The Frank–Wolfe method is based on a first-order approximation of f around the iterate x_k. This means that the relaxed problems are LPs, which can then be solved by using the Simplex method
- Remember the first-order optimality condition: *If $x^* \in X$ is a local minimum of f on X then*

$$\nabla f(x^*)^{\mathrm{T}}(x - x^*) \geq 0, \qquad x \in X,$$

holds

- Remember also the following equivalent statement:

$$\underset{\mathbf{x} \in X}{\text{minimum}}\ \nabla f(x^*)^{\mathrm{T}}(x - x^*) = 0$$

- Follows that if, given an iterate $x_k \in X$,

$$\underset{\mathbf{y} \in X}{\text{minimum}}\ \nabla f(x_k)^{\mathrm{T}}(y - x_k) < 0,$$

  and y_k is an optimal solution to this LP problem, then the direction of $p_k := y_k - x_k$ is a feasible descent direction with respect to f at x

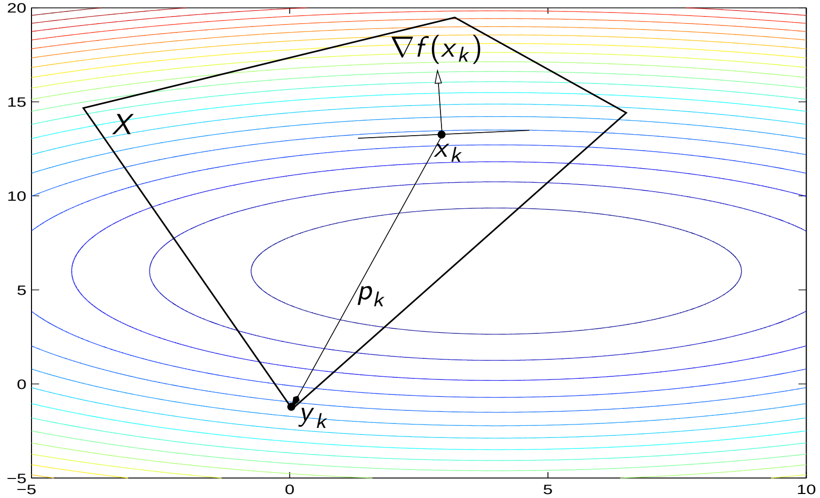
- Search direction towards an extreme point of X [one that is optimal in the LP over X with costs $c = \nabla f(x_k)$]

- This is the basis of the *Frank–Wolfe algorithm*

- We assume that X is bounded in order to ensure that the LP always has a finite optimal solution. The algorithm can be extended to work for unbounded polyhedra

- The search directions then are either towards an extreme point (finite optimal solution to LP) or in the direction of an extreme ray of X (unbounded solution to LP)

- Both cases identified in the Simplex method



$\nabla f(x_k)$
X
x_k
p_k
y_k
20
15
10
5
0
−5
−5
0
5
10

Step 0. Find $x_0 \in X$ (for example any extreme point in X). Set $k := 0$

Step 1. Find an optimal solution y_k to the problem to

$$\operatorname*{minimize}_{\mathbf{y} \in X} z_k(y) := \nabla f(x_k)^{\mathrm{T}}(y - x_k) \qquad (2)$$

Let $p_k := y_k - x_k$ be the search direction

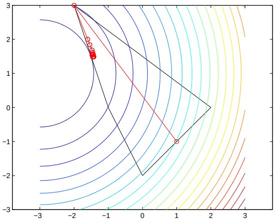
Step 2. Approximately solve the problem to minimize $f(x_k + \alpha p_k)$ over $\alpha \in [0, 1]$. Let α_k be the step length

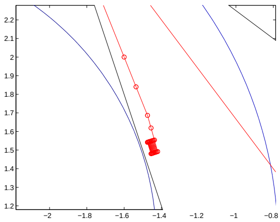
Step 3. Let $x_{k+1} := x_k + \alpha_k p_k$

Step 4. If, for example, $z_k(y_k)$ or α_k is close to zero, then terminate! Otherwise, let $k := k + 1$ and go to Step 1

- *Suppose* $X \subset \mathbb{R}^n$ *nonempty polytope;* f *in* C^1 *on* X
- *In Step 2 of the Frank–Wolfe algorithm, we either use an exact line search or the Armijo step length rule*
- *Then: the sequence* $\{x_k\}$ *is bounded and every limit point (at least one exists) is stationary;*
- $\{f(x_k)\}$ *is descending, and therefore has a limit;*
- $z_k(y_k) \to 0$ $(\nabla f(x_k)^{\mathrm{T}} p_k \to 0)$
- *If* f *is convex on* X*, then every limit point is globally optimal*

Franke-Wolfe convergence





- Remember the following characterization of convex functions in C^1 on X: *f is convex on X* $\Longleftrightarrow$

$$f(y) \geq f(x) + \nabla f(x)^{\mathrm{T}}(y - x), \quad x, y \in X$$

- Suppose f is convex on X. Then, $f(x_k) + z_k(y_k) \leq f^*$ (lower bound, LBD), and $f(x_k) + z_k(y_k) = f(x_k)$ if and only if x_k is globally optimal. *A relaxation—cf. the Relaxation Theorem!*

- Utilize the lower bound as follows: we know that $f^* \in [f(x_k) + z_k(y_k), f(x_k)]$. Store the best LBD, and check in Step 4 whether $[f(x_k) - \mathrm{LBD}]/|\mathrm{LBD}|$ is small, and if so terminate

- Frank–Wolfe uses linear approximations—works best for almost linear problems
- For highly nonlinear problems, the approximation is bad—the optimal solution may be far from an extreme point
- In order to find a near-optimum requires many iterations—the algorithm is slow
- Another reason is that the information generated (the extreme points) is forgotten. If we keep the linear subproblem, we can do much better by storing and utilizing this information

- Remember the Representation Theorem (special case for polytopes): *Let* $P = \{\, x \in \mathbb{R}^n \mid Ax = b;\; x \geq \mathbf{0}^n \}$, *be nonempty and bounded, and* $V = \{v^1, \dots, v^K\}$ *be the set of extreme points of* P. *Every* $x \in P$ *can be represented as a convex combination of the points in* V, *that is,*

$$x = \sum_{i=1}^{K} \alpha_i v^i,$$

for some $\alpha_1, \dots, \alpha_k \geq 0$ *such that* $\sum_{i=1}^{K} \alpha_i = 1$

- The idea behind the Simplicial decomposition method is to generate the extreme points v^i which can be used to describe an optimal solution x^*, that is, the vectors v^i with positive weights α_i in

$$x^* = \sum_{i=1}^{K} \alpha_i v^i$$

- The process is still iterative: we generate a "working set" $\mathcal{P}_k$ of indices i, optimize the function f over the convex hull of the known points, and check for stationarity and/or generate a new extreme point

Step 0. Find $x_0 \in X$, for example any extreme point in X. Set $k := 0$. Let $\mathcal{P}_0 := \emptyset$

Step 1. Let y^k be an optimal solution to the LP problem

$$\operatorname*{minimize}_{\mathbf{y} \in X} \; z_k(y) := \nabla f(x_k)^{\mathrm{T}}(y - x_k)$$

Let $\mathcal{P}_{k+1} := \mathcal{P}_k \cup \{k\}$

Step 2. Let $(\mu_k, \boldsymbol{\nu}_{k+1})$ be an approximate solution to the *restricted master problem* (RMP) to

$$\begin{aligned}
\underset{(\mu,\boldsymbol{\nu})}{\text{minimize}} \quad & f\left(\mu x_k + \sum_{i \in \mathcal{P}_{k+1}} \nu_i y^i\right), && \text{(3a)} \\
\text{subject to} \quad & \mu + \sum_{i \in \mathcal{P}_{k+1}} \nu_i = 1, && \text{(3b)} \\
& \mu, \nu_i \geq 0, \qquad i \in \mathcal{P}_{k+1} && \text{(3c)}
\end{aligned}$$

Step 3. Let $x_{k+1} := \mu_{k+1} x_k + \sum_{i \in \mathcal{P}_{k+1}} (\boldsymbol{\nu}_{k+1})_i y^i$

Step 4. If, for example, $z_k(y^k)$ is close to zero, or if $\mathcal{P}_{k+1} = \mathcal{P}_k$, then terminate! Otherwise, let $k := k+1$ and go to Step 1

- This basic algorithm keeps all information generated, and adds one new extreme point in every iteration
- An alternative is to drop columns (vectors y^i) that have received a zero (or, low) weight, or to keep only a maximum number of vectors
- Special case: maximum number of vectors kept $= 1 \Longrightarrow$ the Frank–Wolfe algorithm!
- We obviously improve the Frank–Wolfe algorithm by utilizing more information
- Unfortunately, we cannot do line search!

- In theory, SD will converge after a finite number of iterations, as there are finite many extreme points.
- However, the restricted master problem is harder to solve when the set $\mathcal{P}_k$ is large. Extreme cases: $|\mathcal{P}_k| = 1$, Frank-Wolfe and line search, easy! If $\mathcal{P}_k$ contains *all* extreme points, the restricted is just the original problem in disguise.
- We fix this by in each iteration also removing some extreme points from $\mathcal{P}$. Practical rules.
  - Drop y^i if $\nu_i = 0$.
  - Limit the size of $|\mathcal{P}_k| = r$. (Again, $r = 1$ is Frank-Wolfe.)

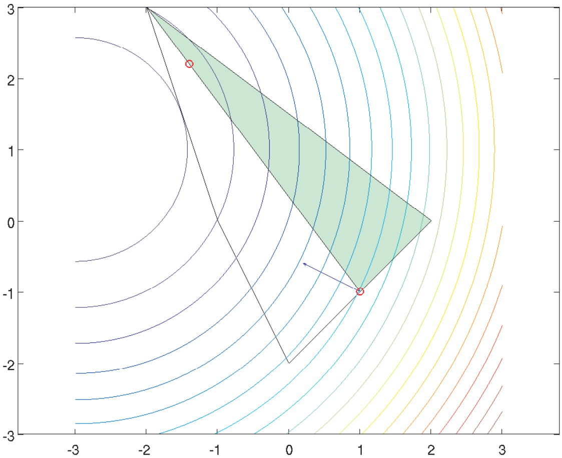


Figure : Example implementation of SD. Starting at $x_0 = (1, -1)^{\mathrm{T}}$, and with $\mathcal{P}_0$ as the extreme points at $(2, 0)^{\mathrm{T}}$, $|\mathcal{P}_k| \leq 2$.

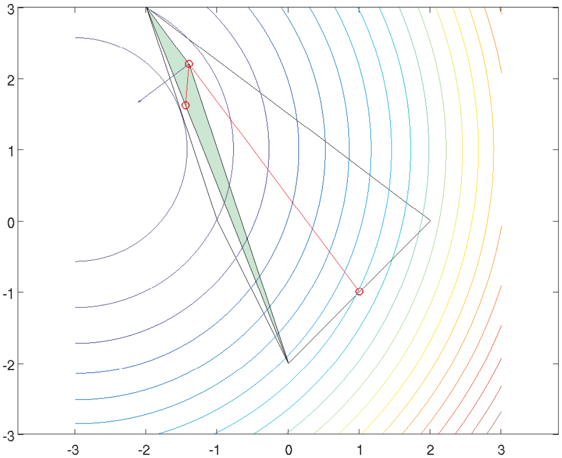


Figure : Example implementation of SD. Starting at $x_0 = (1, -1)^{\mathrm{T}}$, and with $\mathcal{P}_0$ as the extreme points at $(2, 0)^{\mathrm{T}}$, $|\mathcal{P}_k| \leq 2$.

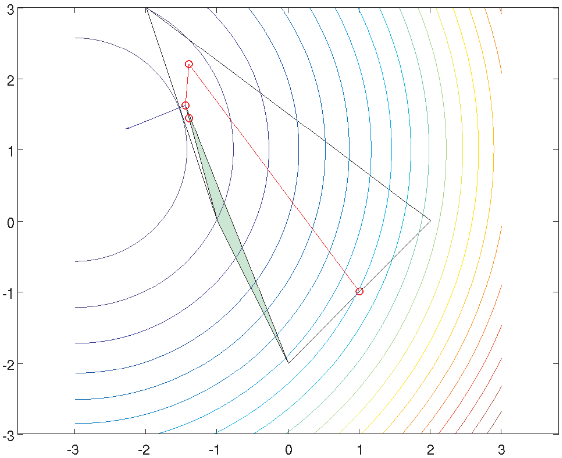


Figure : Example implementation of SD. Starting at $x_0 = (1, -1)^{\mathrm{T}}$, and with $\mathcal{P}_0$ as the extreme points at $(2, 0)^{\mathrm{T}}$, $|\mathcal{P}_k| \leq 2$.

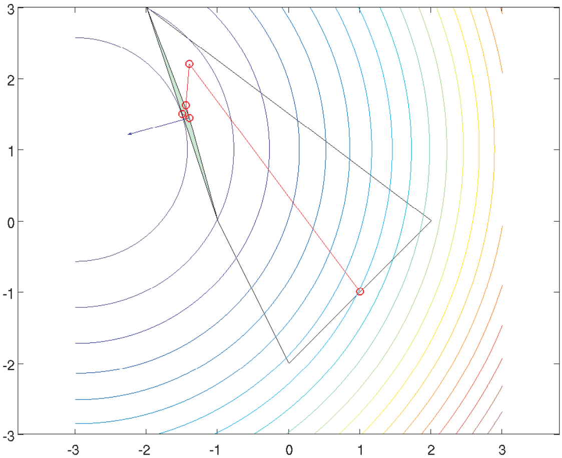


Figure : Example implementation of SD. Starting at $x_0 = (1, -1)^{\mathrm{T}}$, and with $\mathcal{P}_0$ as the extreme points at $(2, 0)^{\mathrm{T}}$, $|\mathcal{P}_k| \leq 2$.

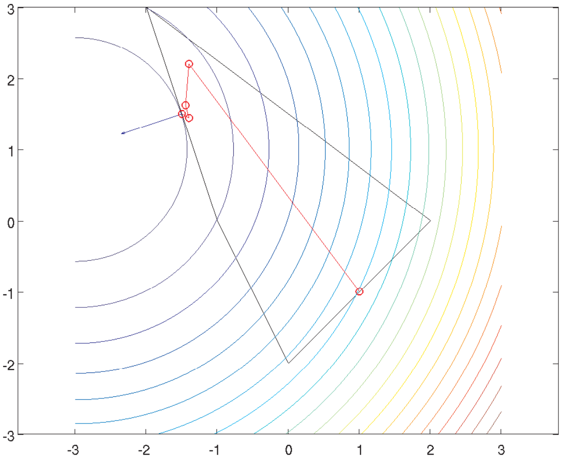


Figure : Example implementation of SD. Starting at $x_0 = (1, -1)^{\mathrm{T}}$, and with $\mathcal{P}_0$ as the extreme points at $(2, 0)^{\mathrm{T}}$, $|\mathcal{P}_k| \leq 2$.

- It does at least as well as the Frank–Wolfe algorithm: line segment $[x_k, y^k]$ feasible in RMP

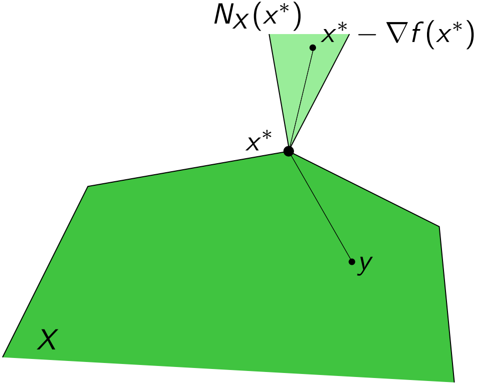
- If x^* unique then convergence is finite if the RMPs are solved exactly, and the maximum number of vectors kept is $\geq$ the number needed to span x^*

- Much more efficient than the Frank–Wolfe algorithm in practice (consider the above FW example!)

- We can solve the RMPs efficiently, since the constraints are simple

- The gradient projection algorithm is based on the projection characterization of a stationary point: $x^* \in X$ *is a stationary point if and only if, for any* $\alpha > 0$,

$$x^* = \mathrm{Proj}_X[x^* - \alpha\nabla f(x^*)]$$



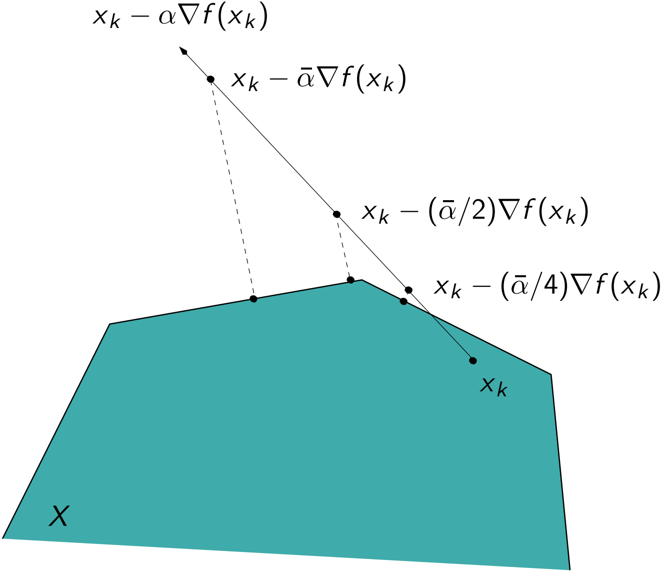
- Let $p := \mathrm{Proj}_X[x - \alpha\nabla f(x)] - x$, for any $\alpha > 0$. Then, if and only if x is non-stationary, p is a feasible descent direction of f at x

- The gradient projection algorithm is normally stated such that the line search is done over the *projection arc*, that is, we find a step length α_k for which

$$x_{k+1} := \mathrm{Proj}_X[x_k - \alpha_k\nabla f(x_k)], \qquad k = 1, \dots \tag{4}$$

has a good objective value. Use the Armijo rule to determine α_k

- Note: gradient projection becomes steepest descent with Armijo line search when $X = \mathbb{R}^n$!


$x_k - \alpha\nabla f(x_k)$
$x_k - \bar{\alpha}\nabla f(x_k)$
$x_k - (\bar{\alpha}/2)\nabla f(x_k)$
$x_k - (\bar{\alpha}/4)\nabla f(x_k)$
x_k
X

- Bottleneck: how can we compute projections?
- In general, we study the KKT conditions of the system and apply a simplex-like method.
- If we have a specially structured feasible polyhedron, projections may be easier to compute.
- Particular case: the unit simplex (the feasible set of the SD subproblems).

- Example: the feasible set is $S = \{x \in \mathbb{R}^n \mid 0 \leq x_i \leq 1, i = 1, \ldots, n\}$.
- Then $\mathrm{Proj}_S(x) = z$, where

$$z_i = \begin{cases} 0, & x_i < 0, \\ x_i, & 0 \leq x_i \leq 1 \\ 1, & 1 < x_i, \end{cases}$$

  for $i = 1, \ldots, n$.
- Exercise: prove this by applying the varitional inequality (or KKT conditions) to the problem

$$\min_{z \in S} \frac{1}{2} \|x - z\|^2$$

.

- *$X \subseteq \mathbb{R}^n$ nonempty, closed, convex; $f \in C^1$ on X;*
- *for the starting point $x_0 \in X$ it holds that the level set $\mathrm{lev}_f\,(f(x_0))$ intersected with X is bounded*
- *In the algorithm (5), the step length α_k is given by the Armijo step length rule along the projection arc*
- *Then: the sequence $\{x_k\}$ is bounded;*
- *every limit point of $\{x_k\}$ is stationary;*
- *$\{f(x_k)\}$ descending, lower bounded, hence convergent*
- Convergence arguments similar to steepest descent one

- *Assume:* $X \subseteq \mathbb{R}^n$ *nonempty, closed, convex;*
- $f \in C^1$ *on* X*;* f *convex;*
- *an optimal solution* x^* *exists*
- *In the algorithm* (5)*, the step length* α_k *is given by the Armijo step length rule along the projection arc*
- *Then: the sequence* $\{x_k\}$ *converges to an optimal solution*
- Note: with $X = \mathbb{R}^n \Longrightarrow$ convergence of steepest descent for convex problems with optimal solutions!

- A large-scale nonlinear network flow problem which is used to estimate traffic flows in cities
- Model over the small city of Sioux Falls in North Dakota, USA; 24 nodes, 76 links, and 528 pairs of origin and destination
- Three algorithms for the RMPs were tested—a Newton method and two gradient projection methods. MATLAB implementation.
- Remarkable difference—The Frank–Wolfe method suffers from very small steps being taken. Why? Many extreme points active = many routes used

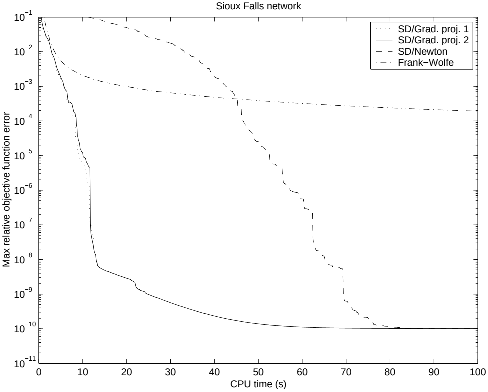


Figure : The performance of SD vs. FW on the Sioux Falls network