9. Gradient Based Algorithms

SOCP, SDP and Gradient Based Algorithms

Sun Woo Lim

Feb 20, 2021

Semi-Definite Programming and Linear Matrix Inequality

SDP in inequality form : : $min_x c^T x$ s.t. $F(x) := F_0 + x_1 F_1 + x_m F_m \succeq 0$ for $F_0, F_1, ..., F_m \in \mathbb{S}^n$.
SDP in standard form : : $min_{X \in \mathbb{S}^n} tr(CX)$ s.t. $tr(A_i X) = b_i$, $i = 1, ..., m$ and $X \succeq 0$
where $C, A_1, ..., A_m \in \mathbb{S}^n$, $b_1, ..., b_m \in \mathbb{R}$

Remember that $\mathbb{S}^n$ is a vector space ($\mathbb{S}^n_+$ is not).
Given fixed symmetric matrices $F_0, F_1, ..., F_m \in \mathbb{S}^n$, $\{F(x) := F_0 + x_1 F_1 + x_m F_m | x \in \mathbb{R}^m\}$ is an **affine subspace** of $\mathbb{S}^n$.
The equation $F_0, F_1, ..., F_m \succeq 0$ is called a **Linear Matrix Inequality**.

If the primal problem is expressed in multiple **LMI**'s, can **incorporate** them into a single **LMI** in **block diagonal form**.

Given $F_0, F_1, ..., F_m \in \mathbb{S}^n$, $\{x \in \mathbb{R}^m | F(x) := F_0 + x_1 F_1 + x_m F_m \succeq 0\}$ is called a **spectrahedron.**

Figure: Left : Polyhedron, Right : Spectrahedron

✓ Every spectrahedron is a convex set, while the converse is not true.

SDP duality

✓ Even if **both** primal and dual are **feasible**, strong duality may fail.
✓ If **either** the primal and dual is **strictly feasible**, $p^* = d^*$.
✓ If **both** are strictly feasible, both the
primal and dual optimal sets are nonempty.

SDP duality Example

$p^* = min_X tr(CX)$ s.t. $tr(X) = b$, $X \in \mathbb{S}_+^n$.
Find p^* and express it in terms of b and eigenvalues of C.

Step 1) Slater's Condition. **I can find** $\frac{b}{n} I$ that **satisfies the equality constraint** and **strictly satisfies the inequality constraint**. Then, $p^* = d^*$.

Step 2) $d^* = max_{\nu \in \mathbb{R}} b\nu$ s.t. $C - \nu I \in \mathbb{S}_+^n$.

The inequality constraint means "All eigenvalues of C are bigger than or equal to ν".

Step 3) $\nu^* = \lambda_{min}(C)$ and $p^* = d^* = b\lambda_{min}(C)$.

Descent Algorithms : General Case

We use **algorithms** to **sequentially** approach the **solution** to an optimization problem : finding the **minimum** of a function.

Here, only deal with **unconstrained minimization problem**.

Why **sequential algorithms**? Usually in **high dimensional setting!**

1. Humans : Hard to analytically find the solution! Cannot **see** the optimum point.
2. Computers : Hard to analytically find the solution! Grid Approximation? Too many Grids.

General Descent Algorithm

1. Start at $x_0 \in dom(f)$.
2. Descent Step : $x_{k+1} = x_k + s_k v_k$, s_k is called a **step size** and v_k is called a **descent direction**.

In order for v_k to be a descent direction, only need $f(x_k + s_k v_k) < f(x_k)$ for all small $s_k > 0$.

Generally, doesn't even need f : differentiable.

For differentiable f, **usually** set $v_k = -\nabla f(x_k)$:
"A ftn **locally decreases** in the direction of **negative gradient**.
If **gradient** is zero, I'm at an **extremum (local minimum / maximum / saddle point)**"

: called **Gradient Descent**.

Gradient Descent : Convex Function

If the objective function is **convex and differentiable** and $dom(f_0) = \mathbb{R}^n$, x^* is the global minimizer of f iff $\nabla_x f(x)|x = x^* = 0$

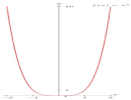
Proof Sketch) $f(y) \geq f(x) + \nabla f(x)^T (y - x), \forall y \in \mathbb{R}^n$: **FOC** of convexity for a differentiable function.

Proof : Section 4.2.3 in Boyd & Vandenburghe

Gradient Descent : Convex Function

However, even for **convex functions**, the claim "the function decreases in the direction of its negative gradient" is only true, **locally**.

(a) $y = x^2$ works for sufficiently small fixed step size



(b) $y = x^4$ may not work even for sufficiently small fixed step size

✓ For some cases, need to decrease the **learning rate** (not necessarily every iteration). ex) $lr_{new} = lr_{old} \times 0.99$

- Advantage : less vulnerable to gradient explosion
- Downfall : ↑ of iteration to converge, more seriously, the learning may stop even if I am not at the optimal point!

Even for a convex functions, selection of **step size** is very important. Discussion about the **step size** keeps going on.

Gradient Descent : Practice Question

$f : \mathbb{R}^n \to \mathbb{R}$, $f(x) = \frac{1}{4}||x||_2^4$. As always, $x^* := argmin_x f(x)$.
Since f is convex w.r.t. x, I use descent algorithms, especially, the gradient descent : $x_{t+1} = x_t - \eta \nabla f(x_t)$ for **fixed** $\eta > 0$.
Find x^* and suppose $||x_0|| = c > 0$. Find the range of η in terms of c s.t. the gradient descent algorithm converge to x^*.

Gradient Descent : Stochastic Gradient Descent

1. Calculating the gradient at each step is the most time consuming part ($\because$ Usually the "objective function" is the average loss over huge dataset).
2. Let B_k be random subset of **indices** then, $w_{k+1} = w_k - \eta(\frac{1}{card(B_k)}) \sum_{i \in B_k} \nabla_w E(w)|_{w=w_k}$
3. This B_k, the sample is called a "batch" or "mini batch".
4. Batch size choice : Speed vs Quality

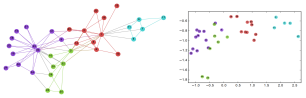
> Of course computing the gradient over the entire dataset is expensive. So now we go to the other extreme. A batch size of just 1 sample. In this case the gradient of that sample may take you completely the wrong direction. But hey, the cost of computing the one gradient was quite trivial. As you take steps with regard to just one sample you "wander" around a bit, but on the average you head towards an equally reasonable local minimum as in full batch gradient descent.

Figure: A good explanation of SGD batch size tradeoff by David Parks, answer from stats.stackexchange

SGD Example : Easy Network Embedding using Matrix Factorization

✓ **Directed Graph** is a set of vertices and a collection of **directed edges** : ex) Graph of likes in SNS
✓ **Adjacency Matrix** is a square matrix representing if pairs of vertices are adjacent in the graph. **symmetric** if **undirected**

What is Network Embedding?



(a) Input: Karate Graph (b) Output: Representation

Figure: Image from Perozzi et al., 2014

$C = \sum_{(u,v \in V \times V} (z_u^T z_v - A_{u,v})^2$: Adjacency based Similarity Learning based on Matrix Factorization

Pseudo Code :

Gradient Based Exact Line Search

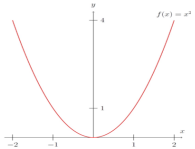
No need of differentiability of a function.

Exact Line Search Steps

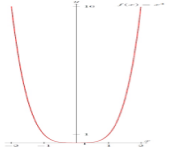
1. Set x_1 and according descent direction v_1 (negative gradient $-\nabla f(x_1)$ or something else)
2. At x_1, choose s_1^* satisfying $f(x_1 + s_1 v_1)$ is minimal.
3. $x_2 = x_1 + s_1^* v_1$ and set v_2 and choose s_2^* satisfying $f(x_2 + s_2 v_2)$ is minimal.
4. $s_k^* := argmin_{s \geq 0} f(x_k + s \times v_k)$. Solve this in every step.

Important) Finding optimal s requires solving a univariate (+ generally non-convex) optimization problem : computationally hard (Section 12.2.1.3 in Calafiore & El Ghaoui) $\rightarrow$ ELS is **not** usually used.

Cases where Gradient Based ELS work



(a) $y = x^2$: ELS works in one step



(b) $y = x^4$: ELS works in one step

ELS : Example

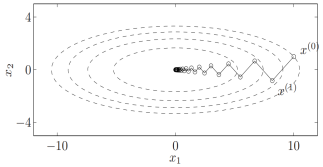


Figure 9.2 Some contour lines of the function $f(x) = (1/2)(x_1^2 + 10x_2^2)$. The condition number of the sublevel sets, which are ellipsoids, is exactly 10. The figure shows the iterates of the gradient method with exact line search, started at $x^{(0)} = (10, 1)$.

Figure: from Boyd & Vandenburghe p469

Let $\gamma \geq 1$ and $f(\vec{x}) := \frac{1}{2}(x_1^2 + \gamma x_2^2) = \frac{1}{2} x^T \begin{bmatrix} 1 & 0 \\ 0 & \gamma \end{bmatrix} x$. Let $\vec{x}_0 = (\gamma, 1)$.

Can show $\vec{x}_k = (\gamma \rho^k, (-1)^k \rho^k)$ where $\rho := \frac{\gamma - 1}{\gamma + 1}$ and thus, $f(\vec{x}_k) = \rho^{2k} f(\vec{x}_0)$

Important) Convergence is slow if $\rho \approx 1 \leftrightarrow \gamma >> 1$. If $\gamma \in (0, 1)$, convergence is slow if $\gamma << 1$: $\kappa(\begin{bmatrix} 1 & 0 \\ 0 & \gamma \end{bmatrix})$ is big.

$\kappa(A) := \frac{\sigma_1}{\sigma_n} = ||A||_2 ||A^{-1}||_2$ derived from $\sigma_{max}(A^{-1}) = \frac{1}{\sigma_{min}(A)}$

Convergence Analysis of Gradient Based ELS in Convex Case

Assumptions

1. Assume $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable
2. Assume $dom(f) = \mathbb{R}^n$
3. Assume $MI \succeq \nabla^2 f(x) \succeq mI$ for some $0 < m < M, \forall x \in \mathbb{R}^n$: function is quadradically upper and lower bounded.

Quadradic Bound of f thanks to the Assumptions

Given $x, y \in \mathbb{R}^n$, I have $f(y) = f(x) + \nabla f(x)^T (y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(z)(y - x)$ for some z between x and y.

Thus, $f(x) + \nabla f(x)^T + \frac{m}{2}||y - x||_2^2 \leq f(y) \leq f(x) + \nabla f(x)^T + \frac{M}{2}||y - x||_2^2$.

: Offers a good lower, upper bound of f. Proof follows in following slides.

The Lower Bound here implies f is bounded below and there's a unique $x^* := argmin_x f(x)$ that **achieves** $p^* = min_x f(x)$.

Math : m-strong convexity and M-Smoothness

m-strongly convexity

Given $m > 0$, $f : \mathbb{R}^n \to \mathbb{R}$ is defined to be **m-strongly convex** if $g(x) := f(x) - \frac{m}{2}||x||_2^2$ is a **convex function**.

If f is twice differentiable, m-strong convexity $\leftrightarrow \nabla^2 f(x) \succeq mI$

12.1.2.1 Quadratic lower bound. We know from the characterization in (8.4) that a differentiable function f is convex if and only if

$$\forall x, y \in \text{dom} f, f(y) \geq f(x) + \nabla f(x)^\top (y - x), \tag{12.8}$$

which means that the linear function $f(x) + \nabla f(x)^\top (y - x)$ is a global lower bound on $f(y)$. Then, applying (12.8) to $f(x) = f_0(x) - \frac{m}{2}\|x\|_2^2$, we have that a differentiable f_0 is strongly convex if and only if

$$\forall x, y \in \text{dom} f_0,\ f_0(y) \geq f_0(x) + \nabla f_0(x)^\top (y - x) + \frac{m}{2}\|y - x\|_2^2, \tag{12.9}$$

which means geometrically that at any $x \in \text{dom} f_0$, there is a convex quadratic function

$$f_{\text{low}}(y) \doteq f_0(x) + \nabla f_0(x)^\top (y - x) + \frac{m}{2}\|y - x\|_2^2$$

Figure: from Calafiore & El Ghaoui Section 12.1.2.1

Math : m-strong convexity and M-Smoothness

M-Smoothness
Given $M > 0$, $f : \mathbb{R}^n \to \mathbb{R}$ is defined to be **M-smooth** if $||\nabla f(y) - \nabla f(x)||_2 \leq M||y - x||_2, \forall x, y \in dom(f)$.

Because M-smoothness implies $\frac{M}{2}||x||_2^2 - f(x)$ is convex,
If f is twice differentiable, M-Smoothness implies $MI \succeq \nabla^2 f(x)$

M - Smoothness means the **gradient is Lipschitz Continuous**.

Convergence Analysis of Gradient Based ELS in Convex Case : Distance to Optimal Point

Convergence Analysis of Gradient Based ELS in Convex Case : Linear Convergence

When the speed of Convergence to the optimality is **exponential (geometric)**, we call **the algorithm linearly converges**.
When the speed of Convergence to the optimality is **doubly exponential**, we call **the algorithm quadradically converges**.

Convergence Analysis of Gradient Based ELS in Convex Case : Linear Convergence

Convergence Analysis of Gradient Based ELS in Convex Case : Linear Convergence

Backtracking Line Search : General Function

BLS attempts to find a good **step size** if the **Armijo Condition** is satisfied.

The **BLS** is another idea to choose the **step size** that makes the function decrease **locally.**
But unlike **ELS**, it **guarantees a suffient rate of decrese in** ϕ without a need to solve an optimization problem of $\phi(s)$.

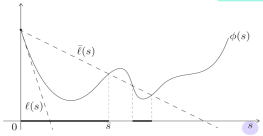
Armijo Condition

Let $\alpha \in (0, 1)$, typically $(0, 0.5)$. Assume I am not at an extremum (local min / max / saddle point).

Suppose the descent direction is v_k (which need not be $-\nabla f(x^{(k)})$

Pick $\beta \in (0, 1)$ and $\alpha \in (0, 0.5)$

The parameter α is typically chosen between 0.01 and 0.3, meaning that we accept a decrease in f between 1% and 30% of the prediction based on the linear extrapolation. The parameter β is often chosen to be between 0.1 (which corresponds to a very crude search) and 0.8 (which corresponds to a less crude search).



$$\phi(s) = f(x^{(k)} + sv_k)$$
$$\bar{l}(s) = f(x^{(k)}) + \alpha s \nabla f(x^{(k)})^T v_k$$
$$l(s) = f(x^{(k)}) + s \nabla f(x^{(k)})^T v_k$$

Again there is some $\bar{s} > 0$ such that

: Armijo Condition

$$\phi(s) < \bar{l}(s), \quad s \in (0, \bar{s}).$$

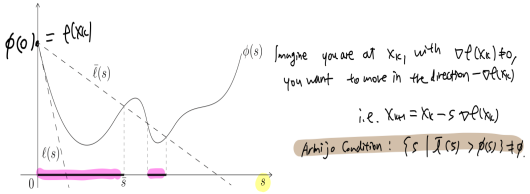
Gradient Based Backtracking Line Search : General Function

The Gradient Based **BLS** attempts to find a good **step size** if the **Armijo Condition** is satisfied.

The **BLS** is another idea to choose the **step size** to **ensure** $f(x_{new}) \leq f(x_{old})$: function is decreasing **locally.**

Armijo Condition

Let $\alpha \in (0,1)$, typically $(0, 0.5)$. Assume $\nabla f(x_k) \neq 0$: I am not at an extremum (local min / max / saddle point).



$$l(s) := f(x^{(k)}) - s\underbrace{\|\nabla f(x^{(k)})\|_2^2}_{= \nabla f(x_k)^T \nabla f(x_k)}$$

$$\bar{l}(s) := f(x^{(k)}) - \alpha s\|\nabla f(x^{(k)})\|_2^2 \quad \text{Slope here} = \alpha \nabla f(x_k)$$

is less steep than the slope of $\nabla f(x_k)$

$$\phi(s) := f(x^{(k)} - s\nabla f(x^{(k)}))$$

Gradient Based Backtracking Line Search : General Function

Pick $\alpha \in (0, 0.5)$ and $\beta \in (0, 1)$

At $x^{(k)}$ to determine $x^{(k+1)}$

- Set $s := 1$

- While

$\phi(s)$

$f(x^{(k)} - s\nabla f(x^{(k)})) >$

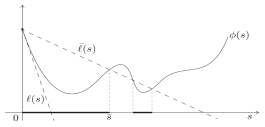
$f(x^{(k)}) - \alpha s \|\nabla f(x^{(k)})\|_2^2$ $\quad l(s)$

(i.e. the Armijo condition fails)

replace s by βs and repeat.

Step by step decrease the step size $s \to \beta s$

- Return $x^{(k)} - s\nabla f(x^{(k)})$ as $x^{(k+1)}$.



The algorithm will terminate in a finite number of iterations when $\nabla f(x^{(k)}) \neq 0$ since

There is an open interval $(0, \bar{s})$ (i.e. $\bar{s} > 0$) such that

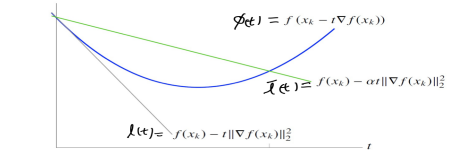
$\phi(s)$

$f(x^{(k)} - s\nabla f(x^{(k)})) <$

$f(x^{(k)}) - \alpha s \|\nabla f(x^{(k)})\|_2^2$ $\quad \bar{l}(s)$

for all $s \in (0, \bar{s})$.

Gradient Based Backtracking Line Search : Convex Function



Exercise) Let $f : \mathbb{R}^2 \to \mathbb{R}$ with $dom(f) = \mathbb{R}^2$, $f(x) = x_1^2 + 3x_2^2$. Let the hyperparameters $\alpha = 0.25, \beta = 0.5$.
Now, $x_k = (2, 2)^T$. Use the Gradient Based **BLS** to find the next position of x : x_{k+1}.

Convergence Analysis of Gradient Based BLS in Convex Case

Same assumption as in ELS case :

1. Assume $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable
2. Assume $dom(f) = \mathbb{R}^n$
3. Assume $MI \succeq \nabla^2 f(x) \succeq mI$ for some $0 < m < M, \forall x \in \mathbb{R}^n$: function is quadradically upper and lower bounded.

Convergence Analysis of Gradient Based BLS in Convex Case