Optimal Methods for Minimizing Heterogeneously Smooth and Convex Compositions

Aaron Zoll

Department of Applied Math and Statistics
Johns Hopkins University

October 8th 2024

Outline

Motivation

Warm up Problem

Theorem

$$a > 0 \Longrightarrow \boxed{(x^*, y^*) = \left(-\frac{b}{2a}, \frac{4ac - b^2}{4a}\right)}$$

$$a < 0 \Longrightarrow \boxed{(x^*, y^*) = \left(-\frac{b}{2a}, \frac{4ac - b^2}{4a}\right)}$$

$$a = 0 \Longrightarrow \textit{uh oh!}$$

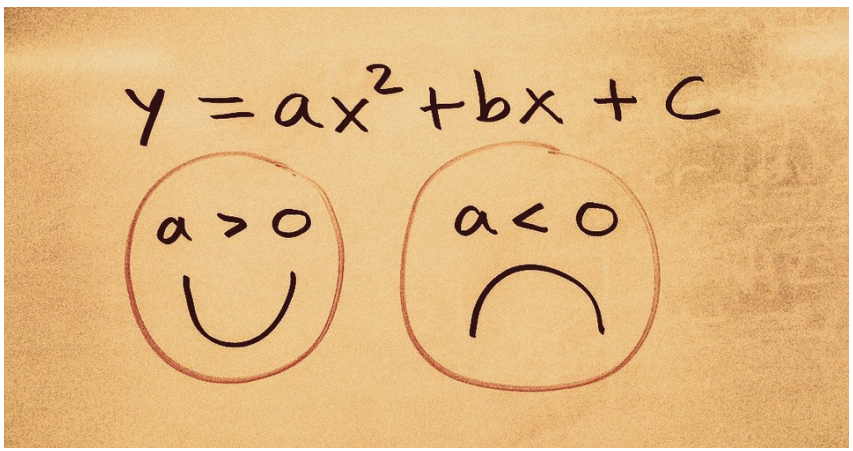


Figure: Happy optimization image

What is Optimization?

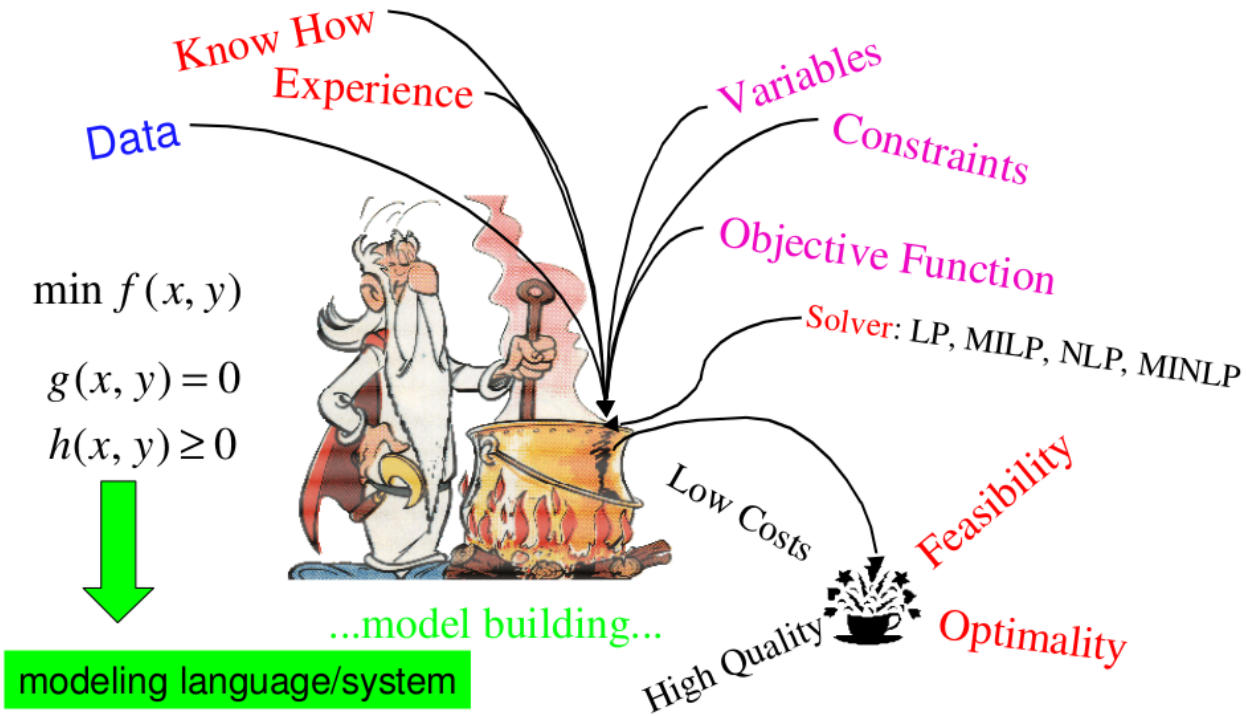
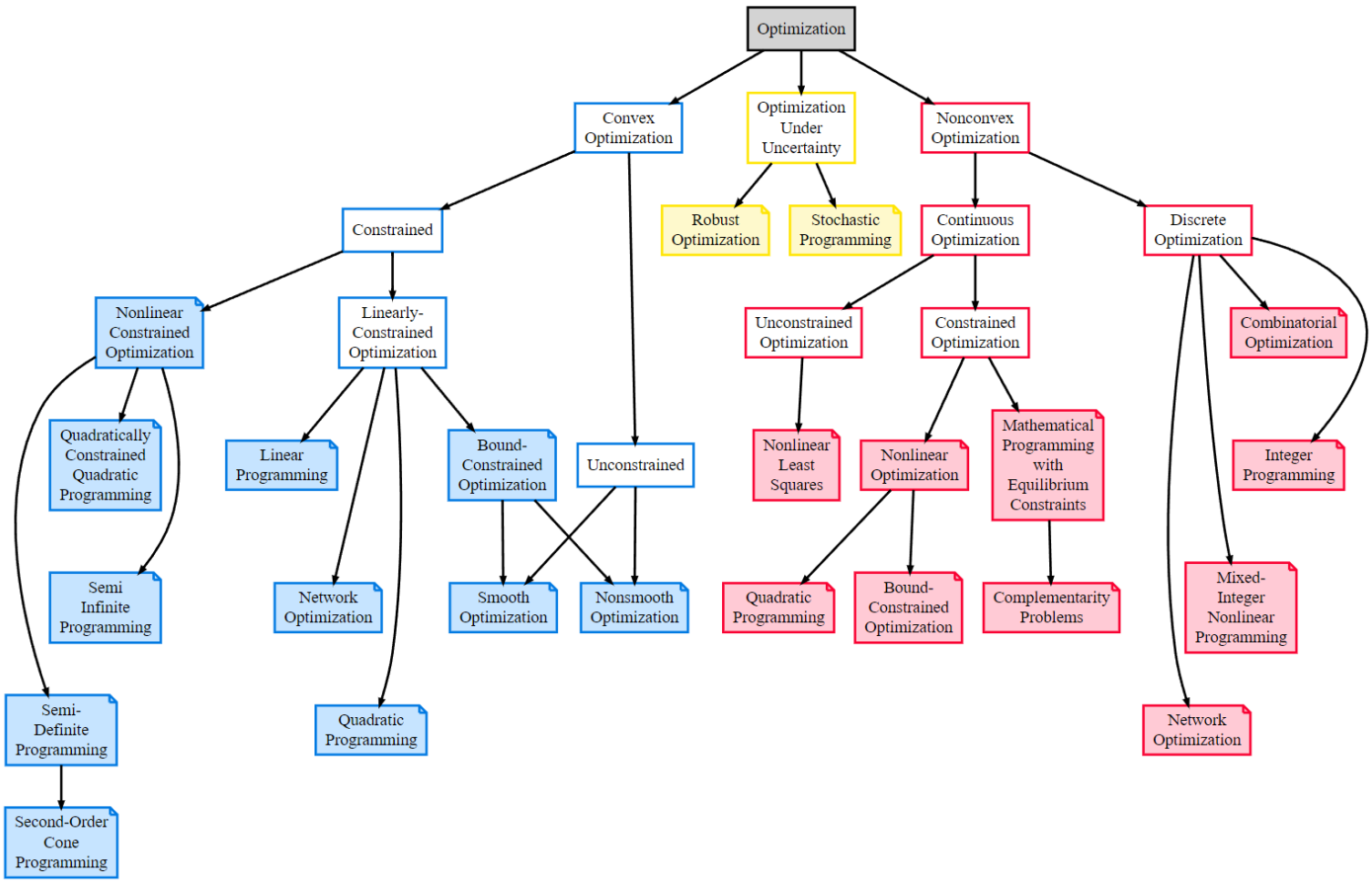


Figure: Optimization Wizard

Different Flavors!



Constrained Optimization [12](motivated this entire project)

Definition

Constrained optimization solves

$$\begin{cases} \min_{x \in \mathcal{X}} & f(x) \\ s.t. & g_i(x) \leq 0 \end{cases}$$

We can rewrite this as

$$\min_{x \in \mathcal{X}} f(x) + \iota_{y \leq 0}(g_1(x), \ldots, g_m(x))$$

where indicator function $\iota_{y \leq 0}(y) = \begin{cases} 0 & y_i \leq 0 \ \forall i = 1, \ldots, m \\ +\infty & o.w. \end{cases}$

Robustness of Composite Optimization

Definition

Composite optimization aims to solve

$$\min_{x \in \mathcal{X}} f(x) + h(g_1(x), \ldots, g_m(x))$$

Remark

Note this looks very similar to Constrained optimization because the latter is a specific case of the former!

Robustness of Composite Optimization

Remark (The Narrative)

We will solve the general composite problem with the following

$$\textit{composition of functions} \rightarrow \textit{sum of functions}$$
$$\textit{sum of functions} \rightarrow \textit{sum of rates}$$
$$\textit{sum of rates} \rightarrow \textit{allows for heterogeneity}$$

Composite Functions	Flavor
$h(z) = \iota_{y \leq 0}(z_1, \ldots, z_m)$	Constrained optimization
$h(z) = a_1 z_1 + \cdots + a_m z_m$	Minimizing sums
$h(z) = \max_i \{z_1, \ldots, z_m\}$	Minimax optimization
$h_\eta(z) = \frac{1}{\eta} \log\left(\sum_{i=1}^m \exp(\eta z_i)\right)$	"Soft-max"
$h_\eta(z) = \frac{1}{\eta} \sum_{i=1}^m z_i \log(z_i)$	Negative Entropy

Heterogeneity

Consider the following problem from statistics:

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{N} \|y - X\beta\|_2^2 \quad \text{s.t.} \quad \|\beta\|_1 \leq t$$

Equivalently, we sometimes see this as

$$\min_{\beta \in \mathbb{R}^p} \underbrace{\|y - X\beta\|_2^2}_{\text{Smooth component}} + \underbrace{\lambda \|\beta\|_1}_{\text{Lipschitz component}}$$

Remark

While each component possess some individual structure, the **objective function** *possess none whatsoever!*

Heterogeneity

Consider the following ML problem of **support vector machines**:

$$\begin{cases} \min_{w,b,\xi} & \|w\|_2^2 + C\sum_{i=1}^n \xi_i \\ s.t. & y_i(w^T x_i - b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{cases}$$

Again, we can reframe this as:

$$\min_{w,b} F(w,b) := \underbrace{\|w\|_2^2}_{\text{Smooth component}} + C\sum_{i=1}^n \underbrace{\max\{0, 1 - y_i(w^T x_i - b)\}}_{\text{Lipschitz component}}$$

Remark

Many nonsmooth functions can be decomposed this way!

Let's talk about parameters

Remark

Heterogeneity $\rightarrow$ *good for expanding function class*

"ugly." (in quotes because no rates are ugly!) for the rates

$$\sum_{j=1}^{m} c_j'' \min\left\{ \frac{2^{\frac{2(q_j-p_j)}{(1+3p_j)(1+q_j)}}}{2^{\frac{2(q_j-p_j)}{(1+3p_j)(1+q_j)}}-1}, \frac{\log_2(R/\tilde{\epsilon})}{2^{\frac{2(q_j-p_j)}{(1+3p_j)(1+q_j)}}} \right\} \left(\frac{L_j^{1+q_j}}{\mu_j^{1+p_j}[\tilde{\epsilon}]^{q_j-p_j}} \right)^{\frac{2}{(1+3p_j)(1+q_j)}}$$

Upshot: we can collapsed all these parameters into two values:

$\underbrace{L_\epsilon^{ADA}}$ handles all the upper curvature

$\underbrace{\mu_\epsilon^{ADA}}$ handles all the lower curvature

Tools

Warm up

- Let's start off with an *easy* question:
- What is $1{,}853{,}020{,}188{,}851{,}841 \times 328{,}256{,}967{,}394{,}537{,}077{,}627$?
- Answer: 608,266,787,713,357,709,119,683,992,618,861,307

- Okay let's try another
- What is $32+43$?
- Answer: 75

- I claim these are both equally easy with the right tools!

What makes a function nice

Remark

"Optimization is hard" - James Schmidt

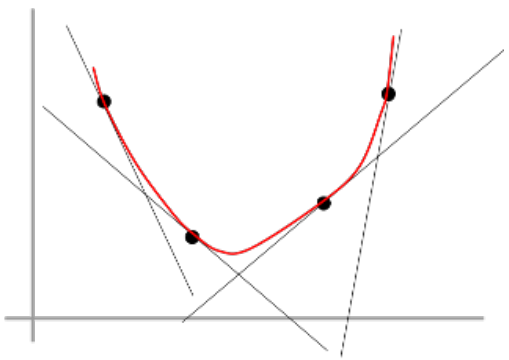
We need to restrict ourselves to nicer classes of functions.

Convexity (recall the quadratic) seems to be a great start as minimizers often

- exist
- correspond to (sub)gradients being zero
- are global [1]

There are many definitions of convexity.

In short, convexity means "curves upwards"



Smooth and Strongly Convex

upper curvature $\rightarrow$ trust gradients as we move away

lower curvature $\rightarrow$ each step forces us much closer

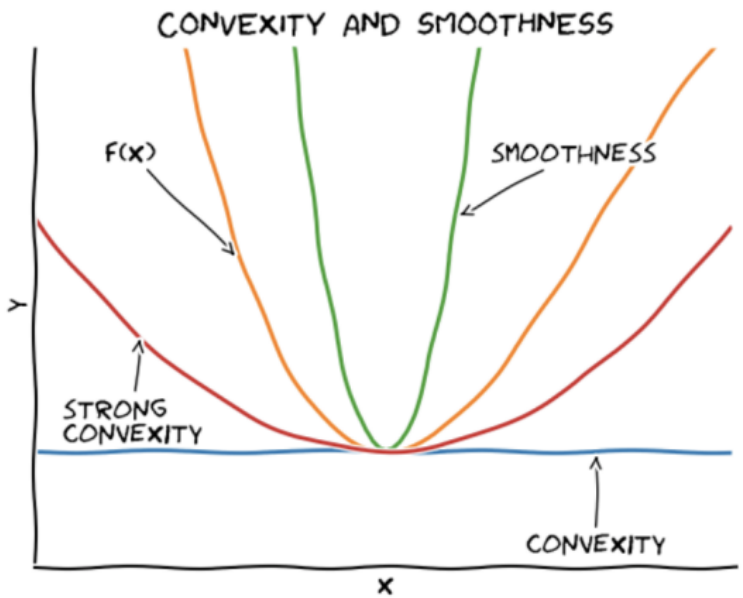


Figure: Zhanyu Wang

Gradient Descent (as discretizied gradient flow)

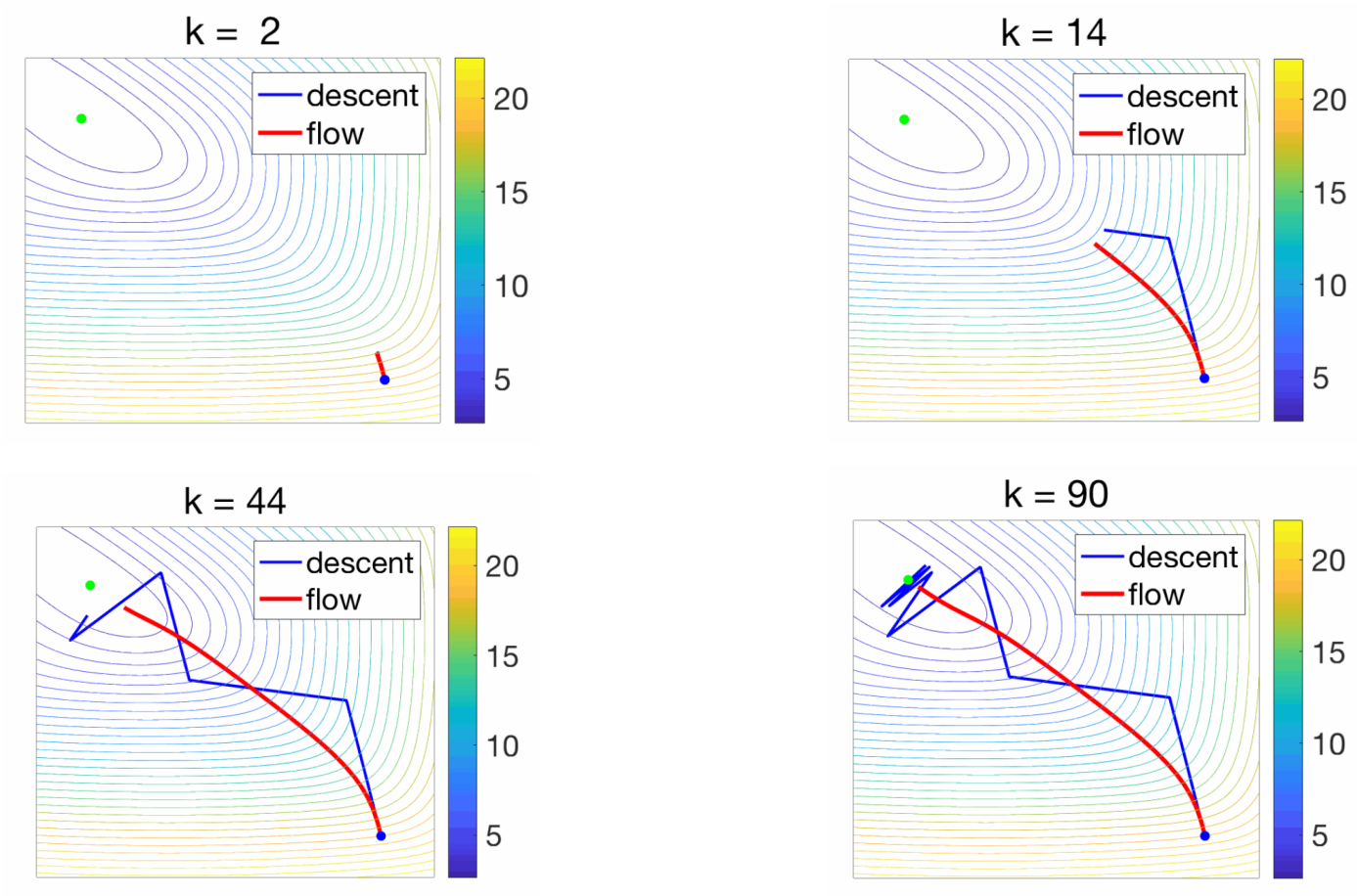


Figure: https://francisbach.com/gradient-flows/

Gradient Descent (as minimizing a majorant)

As $\alpha \to 0$, the quadratic gets real skinny around x_k, and the minimum is at x_k.
As $\alpha \to 2$, we lose the majorant property, but still get descent!

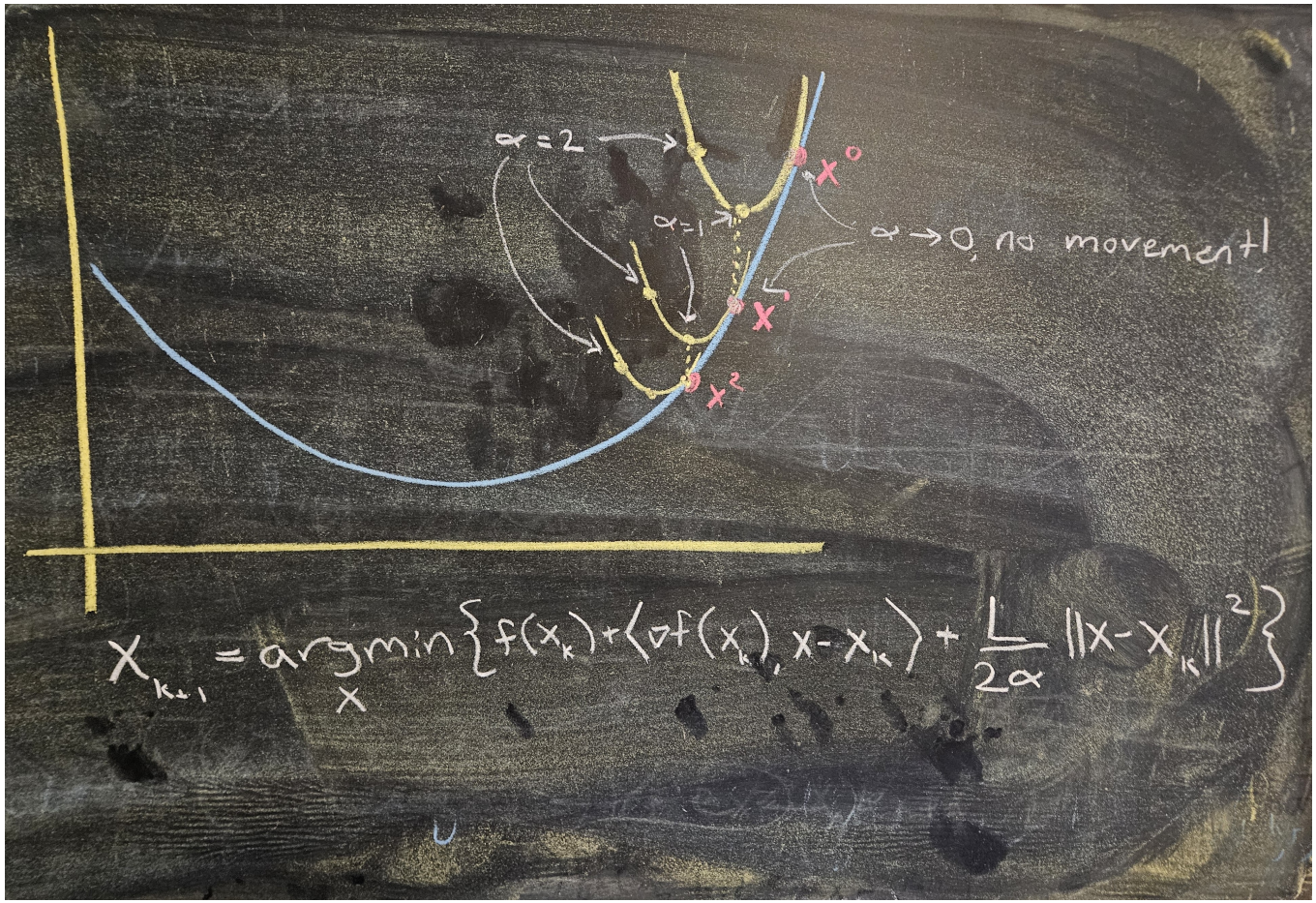


Figure: https://www.desmos.com/calculator/xlyakh5mdj

Gradient Descent (as minimizing a majorant)

Definition

Given stepsize α and "smoothness constant" L. We define the *quadratic approximation* as

$$Q_{f,x_k}(x) := f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{L}{2\alpha} \|x - x_k\|^2$$

We can then explicitly solve for

$$x_{k+1} = \operatorname*{argmin}_x Q_{f,x_k}(x)$$

Since $\nabla Q_{f,x_k}(x) = \nabla f(x_k) + \frac{L}{\alpha}(x - x_k) = 0$ yields

$$x_{k+1} = x_k - \frac{\alpha}{L} \nabla f(x_k)$$

with guaranteed descent for $\alpha \in (0, 2)$. And we get a rate that look like

$$\boxed{f(x^N) - f^* \leq \frac{LD^2}{N+4}}$$

Momentum (magic) [9]

Remark

Momentum is this magical tool that in practice, isn't much harder to run, but in theory is notoriously complicated.

Constant Step Scheme I

0. Choose the point $x_0 \in \mathbb{R}^n$, some $\gamma_0 > 0$, and set $v_0 = x_0$.
1. ***k*th iteration** ($k \geq 0$).

(a) Compute $\alpha_k \in (0, 1)$ from the equation

$$L\alpha_k^2 = (1 - \alpha_k)\gamma_k + \alpha_k\mu.$$

Set $\gamma_{k+1} = (1 - \alpha_k)\gamma_k + \alpha_k\mu$.
(b) Choose $y_k = \frac{1}{\gamma_k + \alpha_k\mu}[\alpha_k\gamma_k v_k + \gamma_{k+1}x_k]$. Compute $f(y_k)$ and $\nabla f(y_k)$.
(c) Set $x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k)$ and

$$v_{k+1} = \frac{1}{\gamma_{k+1}}[(1 - \alpha_k)\gamma_k v_k + \alpha_k\mu y_k - \alpha_k\nabla f(y_k)].$$

Lemma 2.2.3 *Let $\phi_0(x) = \phi_0^* + \frac{\gamma_0}{2}\| x - v_0 \|^2$. Then the process (2.2.4) preserves the canonical form of functions $\{\phi_k(x)\}$:*

$$\phi_k(x) \equiv \phi_k^* + \frac{\gamma_k}{2}\| x - v_k \|^2, \tag{2.2.5}$$

where the sequences $\{\gamma_k\}$, $\{v_k\}$ and $\{\phi_k^*\}$ are defined as follows:

$$\gamma_{k+1} = (1 - \alpha_k)\gamma_k + \alpha_k\mu,$$

$$v_{k+1} = \frac{1}{\gamma_{k+1}}[(1 - \alpha_k)\gamma_k v_k + \alpha_k\mu y_k - \alpha_k\nabla f(y_k)],$$

$$\phi_{k+1}^* = (1 - \alpha_k)\phi_k^* + \alpha_k f(y_k) - \frac{\alpha_k^2}{2\gamma_{k+1}}\| \nabla f(y_k) \|^2 + \frac{\alpha_k(1-\alpha_k)\gamma_k}{\gamma_{k+1}}\left(\frac{\mu}{2}\| y_k - v_k \|^2 + \langle \nabla f(y_k), v_k - y_k \rangle\right).$$

Proof Note that $\nabla^2\phi_0(x) = \gamma_0 I_n$. Let us show that $\nabla^2\phi_k(x) = \gamma_k I_n$ for all $k \geq 0$. Indeed, if it is true for some k, then

$$\nabla^2\phi_{k+1}(x) = (1 - \alpha_k)\nabla^2\phi_k(x) + \alpha_k\mu I_n = ((1 - \alpha_k)\gamma_k + \alpha_k\mu)I_n \equiv \gamma_{k+1}I_n.$$

Figure 4 illustrates the way an estimate sequence aligns with the f function. From definition 1 we can observe that:

$$\forall x \in \mathbb{R}^n, \lim_{k \to \infty} \phi_k(x) \leq f(x)$$

Constant Step scheme III

0. Choose $y_0 = x_0 \in \mathbb{R}^n$.
1. ***k*th iteration** ($k \geq 0$).

$$x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k),$$

$$y_{k+1} = x_{k+1} + \frac{1-\sqrt{q_f}}{1+\sqrt{q_f}}(x_{k+1} - x_k).$$

Figure: Momentum Math

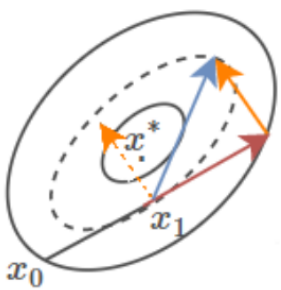
Momentum (magic) [9]

Remark

Key takeaway! Momentum leads to faster convergence. Polyak's fails in some cases. Nesterov's is **provably faster***! (remember this value below)*

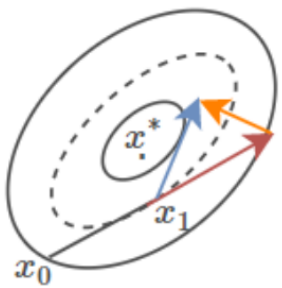
$$f(x^N) - f^* \leq \frac{LD^2}{N^2}$$

Polyak's Momentum



$$x_{t+1} = x_t - \alpha \nabla f(x_t) + \mu(x_t - x_{t-1})$$

Nesterov's Momentum



$$x_{t+1} = x_t + \mu(x_t - x_{t-1}) - \gamma \nabla f(x_t + \mu(x_t - x_{t-1}))$$

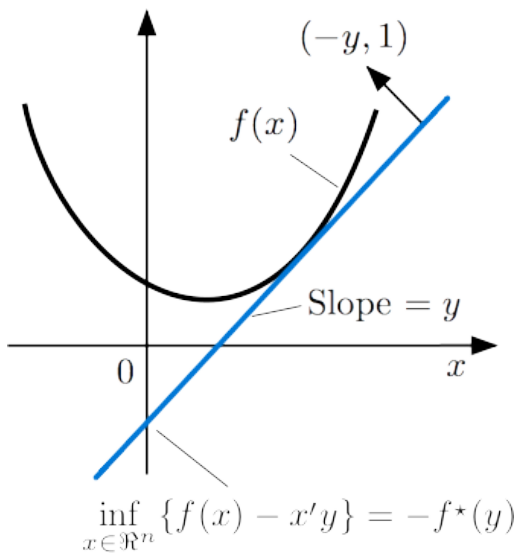
Conjugate Functions [2]

"Duality in mathematics is not a theorem but a principle"-Michael Atiyah

Definition

Given convex function $f : \mathcal{X} \to R$, we define the convex conjugate as

$$f^*(y) := \sup_{x \in X} \{\langle x, y \rangle - f(x)\}$$



convex $f(x)$ $\to$ encodes "primal" objects: points, vectors, resources, cost

conjugate $f^*(y)$ $\to$ encodes "dual" objects: gradients, hyperplanes, prices, profit

Proximal Operators (not as scary as they sound)

Recall back to minimizing that **quadratic upper bound**. We aimed to solve

Formula

$$x_{k+1} = \underset{x}{\operatorname{argmin}} \underbrace{f(x_k) + \langle \nabla f(x_k), x - x_k \rangle}_{\text{call this linearization: } \ell_{f,x_k}} + \frac{L}{2\alpha}\|x - x_k\|^2$$

This motivates

Formula (proximal operator)

$$prox_{g,\tau}(x) := \underset{y}{\operatorname{argmin}}\, g(y) + \frac{\tau}{2}\|y - x\|^2$$

a gradient step is simply $x_{k+1} = \text{prox}_{\ell_{f,x_k},\tau}(x_k)$

Remark

In general, We can replace ℓ_{f,x_k} to any function (some are more computable), to "minimize with a penalty from moving away from x"

Lagrangian

Often to solve constrained optimization , we utilize a *Lagrangian*, which incorporates "Lagrangian-Dual" variables, and instead solve[1]

$$\inf_{x\in\mathcal{X}} \sup_{\lambda\in\mathbb{R}^m_+} \mathcal{L}(x;\lambda) := \underbrace{f(x)}_{\text{(primal) cost function}} + \underbrace{\langle \lambda, g(x)\rangle}_{\text{(dual) penalty for infeasibility}}$$

any $g_i(x)$ infeasible $\Longrightarrow$ inner sup goes to infinity

If all $g_i(x) \leq 0 \Longrightarrow$ supremum attained at $\lambda = 0$

We only look at feasible regions, but in an "unconstrained" way!

Question

How can we utilize this in the composite setting?

[1] $g(x) = (g_1(x), \ldots, g_m(x))$

Lagrangian

Fact

- $(\iota_{y\leq 0})^*(s_1, \ldots, s_m) = \iota_{y\geq 0}(s_1, \ldots, s_m)$. *["conjugate the nonpositive indicator, get the nonnegative indicator"]*
- *If $f : \mathcal{X} \to \mathbb{R}$ is convex, closed, and proper[a], then $(f^*)^* = f$*

[a] not too funky

Motivation for composite Lagrangian

$$\begin{aligned}
\inf_{x\in\mathcal{X}} \sup_{\lambda\in\mathbb{R}^m_+} \mathcal{L}(x;\lambda) &= \inf_{x\in\mathcal{X}} \sup_{\lambda\in\mathbb{R}^m_+} f(x) + \langle \lambda, g(x)\rangle \\
&= \inf_{x\in\mathcal{X}} \sup_{\lambda\in\mathbb{R}^m} f(x) + \langle \lambda, g(x)\rangle - \iota_{y\geq 0}(\lambda) \\
&= \inf_{x\in\mathcal{X}} \sup_{\lambda\in\mathbb{R}^m} f(x) + \langle \lambda, g(x)\rangle - \iota^*_{y\leq 0}(\lambda) \\
&= \inf_{x\in\mathcal{X}} f(x) + \underbrace{\sup_{\lambda\in\mathbb{R}^m} \langle \lambda, g(x)\rangle - \iota^*_{y\leq 0}(\lambda)}_{((\iota_{y\leq 0})^*)^* = \iota_{y\leq 0}} \\
&= \inf_{x\in\mathcal{X}} f(x) + \iota_{y\leq 0}(g(x))
\end{aligned}$$

Composite Lagrangian and Main Story

Therefore, we can turn our primal problem into a primal-dual problem

Formula (simple Lagrangian)

$$\min_{x\in\mathcal{X}} \underbrace{f(x)+h(g(x))}_{F(x)} = \min_{x\in\mathcal{X}} \sup_{\lambda\in\mathbb{R}^m} \underbrace{f(x)+\langle\lambda, g(x)\rangle - h^*(\lambda)}_{\mathcal{L}(x;\lambda)}$$

We now turned a composite problem into an functional sum problem. Our main story will be this:

$$\underbrace{\text{Composition of function}}_{\text{hard}} \rightarrow \underbrace{\text{Sum of functions}}_{\text{less hard}}$$

Composite Function	Flavor	Conjugate Function
$h(z) = \iota_{y\leq 0}(z_1,\ldots,z_m)$	Constrained optimization	$h^*(\lambda) = \iota_{\{y\geq 0\}}(\lambda_1,\ldots,\lambda_m)$
$h(z) = a_1 z_1 + \cdots + a_m z_m$	Minimizing sums	$h^*(\lambda) = \iota_{\{\lambda=(a_1,\ldots,a_m)\}}(\lambda)$
$h(z) = \max_i\{z_1,\ldots,z_m\}$	Minimax optimization	$h^*(\lambda) = \iota_{\{y\in\Delta\}}(\lambda)$
$h_\eta(z) = \frac{1}{\eta}\log\left(\sum_{i=1}^m \exp(\eta z_i)\right)$	"Soft-max"	$h^*(\lambda) = \frac{1}{\eta}\sum_{i=1}^m \lambda_i \log(\lambda_i)$

The "Jimmy and George" Method [12]

Outline

Three main tools:

1. Modified Lagrangian
2. "Q-Analysis"
3. Sliding Technique

Figure: (bear with me through this)

Modified Lagrangian

The standard Lagrangian isn't enough. We want to further "dualize" and unpack the complicated functions.

Formula (modified Lagrangian)

$$\mathcal{L}(x;\lambda,\nu) := f(x) + \langle \lambda, \underbrace{\nu x - g^*(\nu)}_{\text{conjugate of } g} \rangle - h^*(\lambda)$$

We then aim to solve a similar min-max problem

$$\min_{x\in X} f(x) + h(g(x)) = \min_{x\in\mathcal{X}} \max_{\lambda\in\Lambda,\nu\in V} \mathcal{L}(x;\lambda,\nu)$$

Remark

Now we have access to a single function encoding information for points, dual variables, and gradients of our components!

Q-Analysis

We need one more tool...

Definition

If $(x^*; \lambda^*, \nu^*)$ solves above problem, then it is a **saddle point** if for all other $x \in \mathcal{X}$ and $(\lambda, \nu) \in \Lambda \times V$:

$$\mathcal{L}(x^*; \lambda, \nu) \leq \mathcal{L}(x^*; \lambda^*, \nu^*) \leq \mathcal{L}(x; \lambda^*, \nu^*)$$

In particular, for any pair of optimal, sub-optimal points, we have $\mathcal{L}(x; \lambda^*, \nu^*) - \mathcal{L}(x^*; \lambda, \nu) \geq 0$

This inspires the definition of a gap function.

Formula (Gap Function)

$$Q(\hat{z}; z) := \mathcal{L}(\hat{x}; \lambda, \nu) - \mathcal{L}(x; \hat{\lambda}, \hat{\nu})$$

for which the saddle point condition gives

$$\boxed{Q(\hat{z}, z^*) \geq 0}$$

Splitting up Q

Remark

Well unfortunately, we still do not know what to do with this mysterious gap function

$$Q(z^t; z) = Q_\nu(z^t; z) + Q_\lambda(z^t; z) + Q_x(z^t; z)$$

Simply minimize the boxed components (the ones with iterate values ν^t, λ^t, and x^t respectively)

- $Q_\nu(z^t; z) := \mathcal{L}(x^t; \lambda, \nu) - \mathcal{L}(x^t; \lambda, \nu^t) = \langle \lambda, \nu x^t - g^*(\nu) \rangle \boxed{- \langle \lambda, \nu^t x^t - g^*(\nu^t) \rangle}$
- $Q_\lambda(z^t; z) := \mathcal{L}(x^t; \lambda, \nu^t) - \mathcal{L}(x^t; \lambda^t, \nu^t) =$
  $\langle \lambda, \nu^t x^t - g^*(\nu^t) \rangle - h^*(\lambda) \boxed{- \left[\langle \lambda^t, \nu^t x^t - g^*(\nu^t) \rangle - h^*(\lambda^t) \right]}$
- $Q_x(z^t; z) := \mathcal{L}(x^t; \lambda^t, \nu^t) - \mathcal{L}(x; \lambda^t, \nu^t) = \boxed{\langle \sum_{i=1}^m \lambda_i^t \nu_i^t, x^t \rangle + f(x^t)} - \langle \sum_{i=1}^m \lambda_i^t \nu_i^t, x \rangle - f(x)$

Splitting up Q

Remark

We conclude with the following scheme:

minimize Composite $\rightarrow$ *minimax Sum* $\rightarrow$ *minimize Gap* $\rightarrow$ *minimize Linear Functions*[a]

[a] plus perhaps a prox friendly term

We just apply our previous tools (**momentum** to speed up convergence (step 1), **proximal operators** so we don't move too far from our previous iterates, etc.). Step two is just a gradient [11, Lemma 2]

1. $\tilde{x}^t \leftarrow x^{t-1} + \theta_t(x^{t-1} - x^{t-2})$
2. $\nu_i^t \leftarrow \operatorname{argmax}_{\nu_i \in V_i} \langle \nu_i, \tilde{x}^t \rangle - g_i^*(\nu_i) - \tau_t U_{g_i^*}(\nu; \nu^{t-1}) \ \forall i \in [m]$
3. $\lambda^t \leftarrow \operatorname{argmax}_{\lambda \in \Lambda} \langle \lambda, \nu^t \tilde{x}^t - g^*(\nu^t) \rangle - h^*(\lambda) - \frac{\gamma_t}{2} \|\lambda - \lambda^{t-1}\|^2$
4. $x^t \leftarrow \operatorname{argmin}_{x \in \mathcal{X}} \langle \sum_{i=1}^m \lambda_i^t \nu_i^t, x \rangle + f(x) + \frac{\eta_t}{2} \|x - x^{t-1}\|^2$

Sliding Technique (jump scare part 1)

We have now greatly reduced the problem to a **nearly algorithmic method** of computing a momentum term, calling a gradient, and performing two prox steps. We utilize the sliding method [7] to greatly reduce the number of oracle evaluations in return for an inexact solve of some subproblem.

Algorithm 1 Fast Composite Method (FCM) (modified from [27])

Input $x^{-1} = \underline{x}^0 = y_0^{(1)} = x^0 \in \mathcal{X}$, stepsizes $\{\theta_t\}$, $\{\eta_t\}$, $\{\tau_t\}$, and weights $\{\omega_t\}$.

Set $\nu^0 = \nabla g(x^0)$, $\lambda_{-1}^{(1)} = \lambda_0^{(1)} \in \partial h(g(x^0))$

for $t = 1, 2, 3, \ldots, N$ **do**

  Set $\underline{x}^t \leftarrow (\tau_t \underline{x}^{t-1} + \bar{x}^t)/(1+\tau_t)$ where $\bar{x}^t = x^{t-1} + \theta_t(x^{t-1} - x^{t-2})$.

  Set $\nu^t \leftarrow \nabla g(\underline{x}^t)$

  Calculate inner loop iteration limit $\{S_t\}$, stepsizes $\{\beta_s^{(t)}\}$ and $\{\gamma_s^{(t)}\}$, and weights $\{\delta_s^{(t)}\}$.

  for $s = 1, 2, \ldots, S_t$ **do**

    Set $\tilde{h}^{(t),s} = \begin{cases} (\nu^t)^T \lambda_0^{(t)} + \rho_1^{(t)} (\nu^{t-1})^T (\lambda_0^{(t)} - \lambda_{-1}^{(t)}) & if\ s = 1 \\ (\nu^t)^T \lambda_{s-1}^{(t)} + \rho_s^{(t)} (\nu^t)^T (\lambda_{s-1}^{(t)} - \lambda_{s-2}^{(t)}) & o.w. \end{cases}$

    Solve $y_s^{(t)} \leftarrow \underset{y \in \mathcal{X}}{\operatorname{argmin}} \langle \tilde{h}^{(t),s}, y \rangle + f(y) + \frac{\eta_t}{2}\|y - x^{t-1}\|^2 + \frac{\beta_s^{(t)}}{2}\|y - y_{s-1}^{(t)}\|^2$

    Solve $\lambda_s^{(t)} \leftarrow \underset{\lambda \in \Lambda}{\operatorname{argmax}} \langle \lambda, \nu^t (y_s^{(t)} - \underline{x}^t) + g(\underline{x}^t) \rangle - h^*(\lambda) - \frac{\gamma_s^{(t)}}{2}\|\lambda - \lambda_{s-1}^{(t)}\|^2$

  end for

  Set $\lambda_0^{(t+1)} = \lambda_{S_t}^{(t)}$, $\lambda_{-1}^{(t+1)} = \lambda_{S_t - 1}^{(t)}$, $y_0^{(t+1)} = y_{S_t}^{(t)}$

  Set $x^t = \sum_{s=1}^{S_t} \delta_s^{(t)} y_s^{(t)} / \left(\sum_{s=1}^{S_t} \delta_s^{(t)} \right)$ and $\bar{\lambda}^t = \sum_{s=1}^{S_t} \delta_s^{(t)} \lambda_s^{(t)} / \left(\sum_{s=1}^{S_t} \delta_s^{(t)} \right)$

end for

return $\bar{x}^N := \sum_{t=1}^{N} \omega_t x^t / \left(\sum_{t=1}^{N} \omega_t \right)$

Smooth Composite Results

Theorem (jump scare part 2)

Consider the composite setting, and let $L(\Lambda_r)$ be defined above for the Lagrangian and reference set Λ_r. Suppose Algorithm 1 is run with the following stepsizes.

$$\tau_t = \frac{t-1}{2}, \quad \eta_t = \frac{L(\Lambda_r)}{\tau_{t+1}}, \quad \theta_t = \frac{\tau_t}{\tau_{t-1}+1}, \quad \omega_t = \begin{cases} \omega_{t-1}/\theta_t & \text{if } t \geq 2 \\ 1 & \text{if } t = 1 \end{cases} \tag{1}$$

Let $M_t = \|\nu^t\|$, $d(\Lambda_r) = \|\lambda^*\| + r$, and some balancing term $\Delta > 0$, the inner loop stepsizes are calculated as:

$$S_t = \lceil M_t \Delta t \rceil, \quad \tilde{M}_t = \frac{S_t}{\Delta t}, \quad \rho_t = \begin{cases} \tilde{M}_t/\tilde{M}_{t-1} & \text{if } s = 1 \\ 1 & \text{if } s \geq 2 \end{cases}, \quad \beta_t = \frac{\tilde{M}_t d(\Lambda_r)}{\|x^0 - x^*\|}, \quad \gamma_t = \frac{\tilde{M}_t^2}{\beta_t}, \quad \delta_t = 1 \quad \forall s \geq 1 \tag{2}$$

Then we have an ϵ-optimal solution whenever we use at least

$$N \geq \sqrt{\frac{1}{\epsilon}\left(\frac{2d(\Lambda_r)\|x^0 - x^*\|}{\Delta} + L(\Lambda_r)\|x^0 - x^*\|^2\right)}$$

Choosing $\Delta = \frac{d(\Lambda_r)}{\|x^0 - x^*\| L(\Lambda_r)}$ we get that after N steps,

$$\boxed{Q(\bar{z}^N; z) \leq \frac{3L(\Lambda_r)\|x^0 - x^*\|^2}{N^2}}$$

Let's recap (still just in the smooth setting):

- We used *conjugate functions* to remove the composition and encode dual variables and function gradients into a single object to optimize
- We introduced a gap function, one we can split up and minimize sequentially, as a reference for optimality
- Utilizing a sliding technique with cleverly chosen parameters and momentum, we achieve an **optimal rate**.
- That is all to say, that for a given $\epsilon > 0$, we need

$$\boxed{N_\epsilon = \sqrt{\frac{3L(\Lambda_r)\|x^0 - x^*\|^2}{\epsilon}}}$$

steps (expensive gradient calls) to have an ϵ-accurate solution!

- If we want to count total matrix vector multiplications or the prox-steps,

$$\boxed{C_\epsilon = \mathcal{O}\left(\frac{L(\Lambda_r)\|x^0 - x^*\|^2}{\epsilon}\right)}$$

Generalization to Hölder Smoothness and Uniform Convexity

Generalized Curvature

Relaxing our upper and lower curvature conditions

Definition

We say that $f : \mathbb{R}^n \to \mathbb{R}$ is (L, p)-Hölder smooth[a] if $\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\|^p$ or equivalently:

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{p+1}\|y - x\|^{p+1}$$

Conversely, we say f is (μ, q)-Uniformly convex[b] if:

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{q+1}\|y - x\|^{q+1}$$

For $p, q \in [0, 1]$ and $L, \mu \geq 0$

These notions are dual to each other

[a] when $p = 1$, this is the standard smoothness condition

[b] when $q = 1$, this is called "strong convexity"

Hölder Smoothness and L-smoothness

"We're math guys. Of course we tweak hard problems into easy ones and use the same strategy as someone else and call it a new result"

Hölder smoothness is *very* closely related to our standard L-smoothness[2].

Lemma (Nesterov [8])

For $f : \mathbb{R}^d \to \mathbb{R}$, (L, p)-Hölder smooth and $L_\delta \geq \left[\frac{1-p}{1+p}\frac{1}{\delta}\right]^{\frac{1-p}{1+p}} L^{\frac{2}{1+p}}$ we have

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{M}{2}\|y - x\|^2 + \frac{\delta}{2}$$

Lemma (Grimmer [4])

For sum of functions $\bar{g} = \lambda_1 g_1 + \cdots + \lambda_m g_m : \mathbb{R}^d \to \mathbb{R}$, each (L_j, p_j)-Hölder smooth and $L_\delta \geq \sum_{j=1}^m \left[\left[\frac{1-p_j}{1+p_j} \cdot \frac{m}{\delta}\right]^{\frac{1-p_j}{1+p_j}} (\lambda_j L_j)^{\frac{2}{1+p_j}}\right]$ we have

$$\bar{g}(y) \leq \bar{g}(x) + \langle \nabla \bar{g}(x), y - x \rangle + \frac{L_\delta}{2}\|y - x\|^2 + \frac{\delta}{2}$$

[2] https://www.desmos.com/calculator/i93trcyqgq

Choosing the right δ

In the general case, we get a convergence bound like this

$$Q(\bar{z}^N; (x^*; \lambda, \nu)) \leq \frac{1}{2\left(\sum_{t=1}^N \omega_t\right)} \left(\frac{2d(\Lambda_r)\|x^0 - x^*\|}{\Delta} + \frac{L_\delta(\Lambda_r)}{2}\|x^0 - x^*\|^2 + \frac{\delta}{2}\left[\omega_N(\tau_N + 1) + \sum_{t=2}^N \omega_t \tau_t\right]\right)$$

Simplifying (by choosing a nice $\Delta > 0$ and plugging in stepsizes gives)

$$Q(\bar{z}^N; (x^*; \lambda, \nu)) \leq \underbrace{\frac{3L_\delta(\Lambda_r)D^2}{N^2}}_{\text{shrinks with } \delta} + \underbrace{\frac{\delta}{2}N}_{\text{grows with } \delta}$$

Where $L_\delta(\Lambda_r)$ is a function of all the $L_j's$ and p_j'. If we want to make gap function as small as possible, we just need to minimize this one-dimensional problem!

Remark

We will find optimal δ^* (turns out to be ϵ/N), a function of number of iterates N, which is a function of ϵ. $L_\delta(\Lambda_r)$ is a function of δ^*, so this all boils down to one value:

$$Q(\bar{z}^N; (x^*; \lambda, \nu)) \leq \frac{3L_\epsilon^{ADA}\|x^0 - x^*\|^2}{N^2} + \epsilon/2$$

General Hölder Smooth Results

After some heavy algebra we get the following theorem:

Theorem

Taking each g_j being (L_j, p_j)-Hölder smooth, at most $N_\epsilon = N > 0$, that solves[a]

$$\sum_{j=1}^{m} c_j \frac{\hat{L}_j^{\frac{2}{1+p_j}} \|x^0 - x^*\|^2}{\epsilon^{\frac{2}{1+p_j}}} N^{-\frac{1+3p_j}{1+p_j}} = 1$$

[a] Here $\hat{L}_j$ is taken as the maximum over reference set $\lambda_j L_j$, where $c_j = \frac{m^{\frac{2}{1+p_j}}}{(2+2p_j)^{\frac{2}{1+p_j}}}$

Remark

Note: We really just need the sum to be less than 1.

Idea: Make N large enough to make each component small enough (less than 1/m), so the entire sum is less than 1.

General Hölder Smooth Results

Remark

If we want to make sure the total is less than a dollar, we can make sure each stack is less than 20 cents! We will aim to upper bound each summand by 1/m

Figure: AI Coins

General Hölder Smooth Results

This is to say, we can find each N_j by solving

$$c_j \frac{\hat{L}_j^{\frac{2}{1+p_j}} \|x^0 - x^*\|^2}{\epsilon^{\frac{2}{1+p_j}}} N_j^{-\frac{1+3p_j}{1+p_j}} = \frac{1}{m} \implies \boxed{N_j = (mc_j)^{\frac{1+p_j}{1+3p_j}} \frac{\hat{L}_j^{\frac{2}{1+3p_j}} \|x^0 - x^*\|^{2\frac{1+p_j}{1+3p_j}}}{\epsilon^{\frac{2}{1+3p_j}}}}$$

Corollary

In the same setting as above, we can bound the number of iterations to reach an $(\epsilon, \epsilon/r)$-optimal solution can be bounded by any [a]

$$\boxed{N_\epsilon \geq \sum_{j=1}^{m} c_j' \frac{\hat{L}_j^{\frac{2}{1+3p_j}} \|x^0 - x^*\|^{\frac{2+2p_j}{1+3p_j}}}{\epsilon^{\frac{2}{1+3p_j}}}} \tag{3}$$

where $c_j' := \frac{m^{\frac{2}{1+3p_j}}}{(2+2p_j)^{\frac{2}{1+3p_j}}}$

[a] we could instead bound by the max over j

Utilizing Lower Curvature

Let's see how strong-convexity speeds up algorithms. We know from smoothness that after $N = \sqrt{\frac{L\|x^0 - x^*\|^2}{\epsilon}}$ steps:

$$\underbrace{\langle \nabla f(x^*), x^N - x^* \rangle}_{\nabla f(x^*)=0} + \frac{\mu}{2}\|x^N - x^*\|^2 \leq f(x^N) - f(x^*) \leq \epsilon$$

Therefore, we can say

$$\boxed{\|x^N - x^*\|^2 \leq \frac{2\epsilon}{\mu}}$$

If we now **restart** the algorithm initialized at x^N, we can reduce our optimality gap in half after

$$N = N_{\epsilon/2} = \sqrt{\frac{L\|x^N - x^*\|^2}{\epsilon/2}} = 2\sqrt{\frac{L}{\mu}}$$

Utilizing Lower Curvature

Lemma

Given optimality gap $\epsilon_0 = f(x^0) - f(x^*)$, *we can achieve an* $\epsilon_1 = \epsilon_0/2$ *optimal solution in*

$$N = 2\sqrt{L/\mu}$$

iterations for (L, μ) *smooth and strongly convex function.*

Repeating this to reduce our initial gap $f(x^0) - f(x^*)$ to $\tilde{\epsilon}$ takes $\log_2\left(\frac{f(x^0)-f(x^*)}{\tilde{\epsilon}}\right)$ of these restarting runs. In total

$$N_{\tilde{\epsilon}} = 2\log_2\left(\frac{f(x^0) - f(x^*)}{\tilde{\epsilon}}\right)\sqrt{\frac{L}{\mu}}$$

Generalized Lower Curvature

Theorem (jump scare 3)

Suppose each g_j is (L_j, p_j)-Hölder smooth and (μ_j, q_j)-Uniformly convex with $\mu_j > 0$, then for $\tilde{\epsilon} > 0$, restarting achieves an $\tilde{\epsilon}$-optimal solution in at most

$$\sum_{j=1}^{m} c_j'' \min\left\{ \frac{2^{\frac{2(q_j-p_j)}{(1+3p_j)(1+q_j)}}}{2^{\frac{2(q_j-p_j)}{(1+3p_j)(1+q_j)}}-1}, \frac{\log_2(R/\tilde{\epsilon})}{2^{\frac{2(q_j-p_j)}{(1+3p_j)(1+q_j)}}} \right\} \left(\frac{L_j^{1+q_j}}{\mu_j^{1+p_j}[\tilde{\epsilon}]^{q_j-p_j}} \right)^{\frac{2}{(1+3p_j)(1+q_j)}}$$

with $c_j'' := 2^{\frac{2p_j}{1+3q_j}} m^{\frac{1-p_j}{1+3p_j}} (1+q_j)^{\frac{2(1+p_j)}{(1+3p_j)(1+q_j)}} \left[\frac{1}{1+p_j}\right]^{\frac{2}{1+3p_j}}$ *and* $R = Q(z^0; (x^*, \lambda^*, \nu^*))$

Remark

Choosing μ_ϵ^{ADA} correctly, we get that $\mu_\epsilon^{ADA}\|x^N - x^*\|^2 \leq f(x^N) - f(x^*) \leq \epsilon$ after N steps. Therefore we get ϵ-accuracy after

$$N_\epsilon = \mathcal{O}\left(\sqrt{\frac{L_\epsilon^{ADA}}{\mu_\epsilon^{ADA}}} \log(1/\epsilon) \right)$$

Big Picture

Generalized Smoothness and Convexity let's us utilize the same methods for a much wider class of functions. In fact, we can compose the functions in a multitude of ways that may completely remove any structure at all (consider LASSO). Regardless, we really only need access to two constants, and a target accuracy ϵ

	Heterogeneous Formula	"Clean" Formula
Hölder Smooth	$\sum_{j=1}^{m} \frac{\hat{L}_j^{\frac{2}{1+3p_j}} \|x^0 - x^*\|^{\frac{2+2p_j}{1+3p_j}}}{\epsilon^{\frac{2}{1+3p_j}}}$	$\sqrt{\frac{L_\epsilon^{ADA}}{\epsilon}} \|x^0 - x^*\|$
HS & UC	$\sum_{j=1}^{m} \log_2(R/\epsilon) \left(\frac{L_j^{1+q_j}}{\mu_j^{1+p_j} \epsilon^{q_j - p_j}} \right)^{\frac{2}{(1+3p_j)(1+q_j)}}$	$\log(1/\epsilon) \sqrt{\frac{L_\epsilon^{ADA}}{\mu_\epsilon^{ADA}}} Q(z^0, z^*)$

Table: General Order Rates

"Parameter-free"

Universal Methods

As mentioned before, if we consider a general composite problem

$$\min_{x \in \mathcal{X}} f(x) + h(g_1(x), \ldots, g_m(x))$$

$g_j(x)$ both (L_j, p_j)-Hölder smooth and (μ_j, q_j)-Uniformly convex. We have $4m$ different (typically difficult to estimate) functional parameters alone. Universal methods [8, 5, 3, 10, 6] are novel techniques to avoid this exact issue.

Remark

However, this is fixable. We can note that in the end, we just need one upper smoothness constant and one lower smoothness constant, hopefully dependent **only** *on knowledge our target accuracy:* ϵ

Remark

We currently aren't there yet, unfortunately. However, if we assume that our set $\mathcal{X}$ *is bounded, and we have an upper estimate of its diameter, we can make some progress.*

Line Search of L_ϵ^{ADA}

Remark

This technically isn't parameter free, and does require some bounded of $\mathcal{X}$ to be both known and estimated. However, we claim this is much easier to estimate than the composite mess of smoothness parameters.

Algorithm 2 UFCM (Universal Fast Composite Method)

Input $\bar{x} \in \mathcal{X}$, $L_0 > 0$ target accuracy $\epsilon > 0$, and iteration function $N_\epsilon(L)$

Set $\tilde{L} = L_0$
while true **do**
  Set stepsizes to Eq. (3.17) and Eq. (3.18) with $L(\Lambda_r)$ and $d(\Lambda_r)$ replaced by $\tilde{L}$ and $\|x^0 - x^*\|$ replaced by $D_{\mathcal{X}}$
  Run FCM method for $N_\epsilon(\tilde{L})$ iterations
  Set $\bar{x} \leftarrow \bar{x}^{N_\epsilon(\tilde{L})}$ where $\bar{x}^{N_\epsilon(\tilde{L})}$ is the outputed ergodic average
  if $\bar{x}$ satisfies stopping criterion Eq. (3.26) **then**
    break
  else
    $\tilde{L} \leftarrow 2\tilde{L}$
  end if
end while
return $\bar{x}$

Figure: "Parameter Free" Method

Hölder Smooth Results

Remark

Whether we know the value or not, there is some true smoothness constant L_ϵ^{ADA} that tells us how many iterations to run[a]:

$$\sqrt{L_\epsilon^{ADA}/\epsilon}\|x^0 - x^*\|$$

[a] The last run will take by far the longest

Because of clever, adaptive terminating conditions, the final result suffers a small factor $\sqrt{2}$ and only an *additive* log.

$$N_\epsilon^{universal} \leq \max\left\{ \underbrace{\sqrt{12\frac{\max\{L_\epsilon^{ADA}, d(\Lambda_r)\}}{\epsilon}}D_{\mathcal{X}}}_{\text{same rate times } \sqrt{2}} + \underbrace{\left\lceil \log_2\left(\frac{\max\{L(\Lambda_r), d(\Lambda_r)\}}{L_0}\right)\right\rceil}_{\text{negligible term}}, \underbrace{\sqrt{\frac{L_0}{\epsilon}}D_{\mathcal{X}} + 1}_{\text{initial guess too large}} \right\}$$

.

Conclusion

Final Thoughts

- Convex, composite optimization is broad on its own. Adding heterogeneity to the components allows for even more robustness.
- It comes with the drawback of complexity, but only in the work and analysis on our end, not in the actual rates of convergence!
- We can wrap all the underlying constants into just two values, dependent only on the target accuracy.

$$\underbrace{L_\epsilon^{ADA}}_{\text{handles all the upper curvature}} \qquad \underbrace{\mu_\epsilon^{ADA}}_{\text{handles all the lower curvature}}$$

- Simplifying the problem statement with these hyperparameters (and perhaps assuming knowledge of the diameter of $\mathcal{X}$) allows for universal methods to arise.

References

Amitabh Basu.

Convexity and its use in discrete and continuous optimization.
2023.

W. Fenchel.

On conjugate convex functions.
Canadian Journal of Mathematics, 1(1):73–77, 1949.

Nesterov Y.E. Gasnikov, A.V.

Universal method for stochastic composite optimization problems.
Comput. Math. and Math. Phys, pages 48–64, 2018.

Benjamin Grimmer.

On optimal universal first-order methods for minimizing heterogeneous sums.
Optimization Letters, 2023, No 2, p. 427-445, 2023.

Masaru Ito, Zhaosong Lu, and Chuan He.

A parameter-free conditional gradient method for composite minimization under hölder condition.
Journal of Machine Learning Research, 24(166):1–34, 2023.

Ali Kavis, Kfir Y. Levy, Francis Bach, and Volkan Cevher.

Unixgrad: A universal, adaptive algorithm with optimal guarantees for constrained optimization.
33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada., 2019.

Guanghui Lan.

Gradient sliding for composite optimization.
arXiv preprint arXiv:1406.0919, 2014.

References

Yurii Nesterov.

Universal gradient methods for convex optimization problems.
Mathematical Programming, 152(1–2):381–404, May 2014.

Yurii Nesterov.

Lectures on Convex Optimization.
Springer Cham, Switzerland, 2 edition, 2018.

Nuozhou Wang and Shuzhong Zhang.

A gradient complexity analysis for minimizing the sum of strongly convex functions with varying condition numbers.
SIAM Journal on Optimization, 34(2):1374–1401, 2024.

Zhe Zhang and Guanghui Lan.

Optimal algorithms for convex nested stochastic composite optimization.
arVix preprint arXiv:2011.10076, 2022.

Zhe Zhang and Guanghui Lan.

Solving convex smooth function constrained optimization is almost as easy as unconstrained optimization.
arXiv preprint arXiv:2210.05807, 2022.