Universal Conditional Gradient Sliding

Yuyuan Ouyang and Trevor Squires[1]

CORI Seminar 2021

[1] This research is partially supported by the Office of Navel Research grant N00014- 20-1-2089.

Problem Setting

Convex Optimization

Our problem of interest is computing an ε-solution $\tilde{x} \in X$ to

$$f^* := \min_{x \in X} f(x) \tag{CO}$$

such that $f(\tilde{x}) - f^* < \varepsilon$.

Problem Setting

Convex Optimization

Our problem of interest is computing an ε-solution $\tilde{x} \in X$ to

$$f^* := \min_{x \in X} f(x) \tag{CO}$$

such that $f(\tilde{x}) - f^* < \varepsilon$.

Here,

- f is a real-valued, convex function
- $\mathbb{R}^n$ is a high dimensional space
- $X \subseteq \mathbb{R}^n$ closed, bounded, and convex

Problem Setting

Convex Optimization

Our problem of interest is computing an ε-solution $\tilde{x} \in X$ to

$$f^* := \min_{x \in X} f(x) \tag{CO}$$

such that $f(\tilde{x}) - f^* < \varepsilon$.

There are numerous additional properties that can be leveraged to more easily achieve our task.

Problem Setting

Convex Optimization

Our problem of interest is computing an ε-solution $\tilde{x} \in X$ to

$$f^* := \min_{x \in X} f(x) \tag{CO}$$

such that $f(\tilde{x}) - f^* < \varepsilon$.

There are numerous additional properties that can be leveraged to more easily achieve our task.

- projection onto X is easy
- differentiability of f
- smoothness of f

Problem Setting

Convex Optimization

Our problem of interest is computing an ε-solution $\tilde{x} \in X$ to

$$f^* := \min_{x \in X} f(x) \tag{CO}$$

such that $f(\tilde{x}) - f^* < \varepsilon$.

There are numerous additional properties that can be leveraged to more easily achieve our task.

- projection onto X is easy
- differentiability of f
- smoothness of f
- Lipschitz continuity of f
- strong convexity of f

Problem Setting

Convex Optimization

Our problem of interest is computing an ε-solution $\tilde{x} \in X$ to

$$f^* := \min_{x \in X} f(x) \qquad \text{(CO)}$$

such that $f(\tilde{x}) - f^* < \varepsilon$.

There are numerous additional properties that can be leveraged to more easily achieve our task.

- projection onto X is easy
- differentiability of f
- smoothness of f
- Lipschitz continuity of f
- strong convexity of f

For now, let us assume ∇f exists and is Lipschitz continuous with Lipschitz constant L, i.e. f is L-smooth, and that a projection onto X is computationally feasible.

Solving (CO)

Convex Optimization

Our problem of interest is computing an ε-solution $\tilde{x} \in X$ to

$$f^* := \min_{x \in X} f(x) \tag{CO}$$

such that $f(\tilde{x}) - f^* < \varepsilon$.

How fast is fast and how do we measure this?

Solving (CO)

Convex Optimization

Our problem of interest is computing an ε-solution $\tilde{x} \in X$ to

$$f^* := \min_{x \in X} f(x) \tag{CO}$$

such that $f(\tilde{x}) - f^* < \varepsilon$.

How fast is fast and how do we measure this?

$\rightarrow$ count the number of expensive operations

Solving (CO)

Convex Optimization

Our problem of interest is computing an ε-solution $\tilde{x} \in X$ to

$$f^* := \min_{x \in X} f(x) \tag{CO}$$

such that $f(\tilde{x}) - f^* < \varepsilon$.

How fast is fast and how do we measure this?

$\rightarrow$ count the number of expensive operations

Example: Gradient Descent (GD)

$$\begin{aligned} x_k &= \operatorname*{argmin}_{u \in X} \left|\left| u - \left(x_{k-1} - \frac{1}{\eta_k} \nabla f(x_{k-1}) \right) \right|\right|^2 \\ &= \operatorname*{argmin}_{u \in X} \langle \nabla f(x_{k-1}), u \rangle + \frac{\eta_k}{2} \left|\left| u - x_{k-1} \right|\right|^2 \end{aligned}$$

- for properly chosen η_k, GD achieves an ε-solution in $\mathcal{O}(1/\varepsilon)$ iterations
- only expensive operation is gradient evaluation, and GD uses 1 per iteration

Algorithm 1 Nesterov's accelerated gradient descent (NAGD)

Start: Choose $x_0 \in X$. Set $y_0 := x_0$

for $k = 1, \ldots, N$ **do**

$$z_k = (1 - \gamma_k) y_{k-1} + \gamma_k x_{k-1},$$

$$x_k = \underset{u \in X}{\operatorname{argmin}} \langle \nabla f(z_k), u \rangle + \frac{\eta_k}{2} \|u - x_{k-1}\|^2,$$

$$y_k = (1 - \gamma_k) y_{k-1} + \gamma_k x_k.$$

end for

Output y_N.

Solving (CO)

Algorithm 1 Nesterov's accelerated gradient descent (NAGD)

Start: Choose $x_0 \in X$. Set $y_0 := x_0$

for $k = 1, \ldots, N$ **do**

$$
\begin{aligned}
z_k &= (1-\gamma_k)y_{k-1} + \gamma_k x_{k-1}, \\
x_k &= \operatorname*{argmin}_{u \in X} \langle \nabla f(z_k), u \rangle + \frac{\eta_k}{2}\|u - x_{k-1}\|^2, \\
y_k &= (1-\gamma_k)y_{k-1} + \gamma_k x_k.
\end{aligned}
$$

end for

Output y_N.

- minimizes linear approximation proximal problem
- subproblem is still a projection
- reduces to gradient descent when $\gamma_k \equiv 1$

Algorithm 1 Nesterov's accelerated gradient descent (NAGD)

Start: Choose $x_0 \in X$. Set $y_0 := x_0$
for $k = 1, \ldots, N$ **do**

$$\begin{aligned} z_k =& (1-\gamma_k) y_{k-1} + \gamma_k x_{k-1}, \\ x_k =& \underset{u \in X}{\operatorname{argmin}} \langle \nabla f(z_k), u \rangle + \frac{\eta_k}{2} \|u - x_{k-1}\|^2, \\ y_k =& (1-\gamma_k) y_{k-1} + \gamma_k x_k. \end{aligned}$$

end for
Output y_N.

- minimizes linear approximation proximal problem
- subproblem is still a projection
- reduces to gradient descent when $\gamma_k \equiv 1$
- computes ε-solution in only $\mathcal{O}(1/\sqrt{\varepsilon})$ iterations
- requires knowledge of L to set η_k appropriately
- is optimal for solving problems such as (CO) under first order oracle ([1])

Solving (CO)

But what if the projection is not so easy?

Solving (CO)

But what if the projection is not so easy?

- certain sets can be as difficult to project to as the underlying problem is to solve
  - $X = \text{conv}(v_1, \ldots, v_p)$
  - $X = \{Y \in \mathbb{R}^{n \times n} : \text{tr}(Y) = 1, Y \succeq 0\}$
- NAGD is of no use when projection is more difficult than (CO)
- want to design algorithms that do not require difficult optimizations over X, i.e. projection free methods

Algorithm 2 Conditional Gradient (CG)

Start: Choose $y_0 \in X$.
for $k = 1, \ldots, N$ **do**

$$x_k = \underset{x \in X}{\operatorname{argmin}} \langle \nabla f(y_{k-1}), x \rangle$$
$$y_k = (1 - \alpha_k) y_{k-1} + \alpha_k x_k$$

end for
Output y_N.

Algorithm 2 Conditional Gradient (CG)

Start: Choose $y_0 \in X$.
for $k = 1, \dots, N$ **do**

$$x_k = \underset{x \in X}{\operatorname{argmin}} \langle \nabla f(y_{k-1}), x \rangle$$
$$y_k = (1 - \alpha_k) y_{k-1} + \alpha_k x_k$$

end for
Output y_N.

- solves a linear optimization (LO) rather than a projection
  - > when X is convex hull, the LO is a linear program
  - > when X is standard spectrahedron, the LO is a smallest eigenvalue problem
- requires $\mathcal{O}(1/\varepsilon)$ number of iterations to obtain ε-solution [2]
- more gradient evaluations and the addition of linear optimizations, but no projections at all
- optimal number of linear optimizations

Projection Free Methods for Solving (CO)

Question

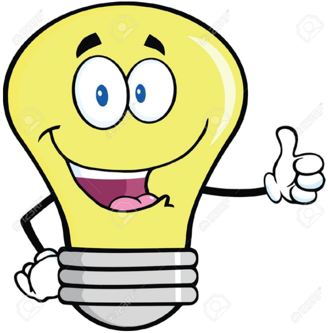
Comparing CG to NAGD, we increase in the complexity of gradient evaluations necessary. Is it possible to keep the gradient evaluations unchanged while being projection free?

Projection Free Methods for Solving (CO)

Question

Comparing CG to NAGD, we increase in the complexity of gradient evaluations necessary. Is it possible to keep the gradient evaluations unchanged while being projection free?

The answer is yes! Simply solve x_k subproblem with a projection free algorithm.



Algorithm 3 Conditional Gradient Sliding (CGS)

Start: Choose $x_0 \in X$. Set $y_0 := x_0$

for $k = 1, \ldots, N$ **do**

$$\begin{aligned} z_k =& (1-\gamma_k) y_{k-1} + \gamma_k x_{k-1}, \\ x_k =& CG(\nabla f(\underline{x}_k), x_{k-1}, \eta_k, \varepsilon_k) \\ y_k =& (1-\gamma_k) y_{k-1} + \gamma_k x_k. \end{aligned}$$

end for

Output y_N.

Algorithm 3 Conditional Gradient Sliding (CGS)

Start: Choose $x_0 \in X$. Set $y_0 := x_0$

for $k = 1, \ldots, N$ **do**

$$\begin{aligned} z_k &= (1-\gamma_k) y_{k-1} + \gamma_k x_{k-1}, \\ x_k &= CG(\nabla f(\underline{x}_k), x_{k-1}, \eta_k, \varepsilon_k) \\ y_k &= (1-\gamma_k) y_{k-1} + \gamma_k x_k. \end{aligned}$$

end for

Output y_N.

- solves x_k subproblem approximately with linear optimizations only
- if parameters are chosen properly, CGS computes ε-solution in $\mathcal{O}(1/\sqrt{\varepsilon})$ gradient evaluations and $\mathcal{O}(1/\varepsilon)$ linear optimizations [3]
- requires L for setting of η_k

Smoothness

A key feature of all the above algorithms is the assumption that the gradient of f is Lipschitz with constant L, i.e,

$$f(x) \leq f(u) + \langle \nabla f(u), x - u \rangle + \frac{L}{2} \|x - u\|^2.$$

Smoothness

A key feature of all the above algorithms is the assumption that the gradient of f is Lipschitz with constant L, i.e,

$$f(x) \leq f(u) + \langle \nabla f(u), x - u \rangle + \frac{L}{2} \|x - u\|^2.$$

This combined with convexity can be leveraged to design efficient optimization methods. However, we may not always have such luxury.

- $f(x) = \lambda \, ||x||$
- $f(x) = \max_{y \in \Delta_m} \langle x, Ay \rangle$

Smoothness Relaxation

Relaxed Assumption - Hölder Smooth

Assume that there exists a Hölder exponent $\nu \in [0, 1]$ and constant $M_\nu > 0$ such that

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{M_\nu}{1+\nu} \|x - y\|^{1+\nu}, \ \forall x, y \in X.$$

Smoothness Relaxation

Relaxed Assumption - Hölder Smooth

Assume that there exists a Hölder exponent $\nu \in [0,1]$ and constant $M_\nu > 0$ such that

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{M_\nu}{1+\nu} \|x - y\|^{1+\nu}, \ \forall x, y \in X.$$

This is a generalization of Lipschitz continuous gradient. In particular,

- any convex smooth f with Lipschitz continuous gradient M_1 is Hölder smooth with $\nu = 1$
- any convex nonsmooth Lipschitz continuous f with is Hölder smooth with $\nu = 0$
- any convex smooth f satisfying

$$\|\nabla f(y) - \nabla f(x)\| \leq M_\nu \|y - x\|^\nu, \ \forall x, y \in X$$

is Hölder smooth with $\nu \in (0,1)$

Algorithm 4 Fast Gradient Method (FGM)

Start: Choose $x_0 \in X$ and $\varepsilon > 0$. Set $y_0 = x_0$

for $k = 1, \ldots, N$ **do**

Decide $L_k > 0$ satisfying

$$f(y_k) \leq f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{L_k}{2} \|y_k - z_k\|^2 + \frac{\varepsilon}{2}\gamma_k$$

where

$$\begin{aligned} z_k =& (1-\gamma_k)y_{k-1} + \gamma_k x_{k-1}, \\ x_k =& \operatorname*{argmin}_{u \in X} \langle \nabla f(z_k), u \rangle + \frac{\eta_k}{2} \|u - x_{k-1}\|^2, \\ y_k =& (1-\gamma_k)y_{k-1} + \gamma_k x_k. \end{aligned}$$

end for

Output y_N.

Algorithm 4 Fast Gradient Method (FGM)

Start: Choose $x_0 \in X$ and $\varepsilon > 0$. Set $y_0 = x_0$
for $k = 1, \ldots, N$ **do**
Decide $L_k > 0$ satisfying

$$f(y_k) \leq f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{L_k}{2} \|y_k - z_k\|^2 + \frac{\varepsilon}{2}\gamma_k$$

where

$$\begin{aligned} z_k =& (1-\gamma_k) y_{k-1} + \gamma_k x_{k-1}, \\ x_k =& \operatorname*{argmin}_{u \in X} \langle \nabla f(z_k), u \rangle + \frac{\eta_k}{2} \|u - x_{k-1}\|^2, \\ y_k =& (1-\gamma_k) y_{k-1} + \gamma_k x_k. \end{aligned}$$

end for
Output y_N.

- achieves ε-solution in $\mathcal{O}((1/\varepsilon)^{\frac{2}{1+3\nu}})$ iterations
- also optimal

Comparison of Algorithms

We repeat the improvements

- rather than minimizing f with FGM, we can apply CG to minimize using linear optimizations instead of projections
- CG for functions with Hölder continuous gradients requires $\mathcal{O}((1/\varepsilon)^{\nu})$ iterations [4]
- we can preserve the $\mathcal{O}((1/\varepsilon)^{\frac{2}{1+3\nu}})$ gradient evaluations by approximately solving the x_k subproblem in FGM using CG

Algorithm 5 Universal Conditional Gradient Sliding (UCGS)

Start: Choose $x_0 \in X$ and $\varepsilon > 0$. Set $y_0 = x_0$
for $k = 1, \dots, N$ **do**
Decide $L_k > 0$ satisfying

$$f(y_k) \leq f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{L_k}{2} ||y_k - z_k||^2 + \frac{\varepsilon}{2}\gamma_k$$

where

$$\begin{aligned} z_k =& (1 - \gamma_k) y_{k-1} + \gamma_k x_{k-1}, \\ x_k =& ACG(\nabla f(z_k), x_{k-1}, \eta_k, \varepsilon_k, \delta_k) \\ y_k =& (1 - \gamma_k) y_{k-1} + \gamma_k x_k. \end{aligned}$$

Terminate if

$$\max_{x \in X} f(y_k) - \ell_k(x) \leq \varepsilon$$

where

$$\ell_k(x) = \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i}{\Gamma_i} \left(f(z_i) + \langle \nabla f(z_i, x - z_i \rangle \right)$$

end for
Output y_N.

Comparison of Algorithms

Properties of UCGS

- contains a stopping condition

Properties of UCGS

- contains a stopping condition
- does not require knowledge of (ν, M_ν) for setting of parameters

Comparison of Algorithms

Properties of UCGS

- contains a stopping condition
- does not require knowledge of (ν, M_ν) for setting of parameters
- maintains the $\mathcal{O}((1/\varepsilon)^{\frac{2}{1+3\nu}})$ gradient evaluations established in FGM for an ε-solution

Comparison of Algorithms

Properties of UCGS

- contains a stopping condition
- does not require knowledge of (ν, M_ν) for setting of parameters
- maintains the $\mathcal{O}((1/\varepsilon)^{\frac{2}{1+3\nu}})$ gradient evaluations established in FGM for an ε-solution
- **improves** the number of linear optimizations required of CG to $\mathcal{O}((1/\varepsilon)^{\frac{4}{1+3\nu}})$

Comparison of Algorithms

Properties of UCGS

- contains a stopping condition
- does not require knowledge of (ν, M_ν) for setting of parameters
- maintains the $\mathcal{O}((1/\varepsilon)^{\frac{2}{1+3\nu}})$ gradient evaluations established in FGM for an ε-solution
- **improves** the number of linear optimizations required of CG to $\mathcal{O}((1/\varepsilon)^{\frac{4}{1+3\nu}})$
- allows linear optimization problems to be solved approximately

Comparison of Algorithms

Properties of UCGS

- contains a stopping condition
- does not require knowledge of (ν, M_ν) for setting of parameters
- maintains the $\mathcal{O}((1/\varepsilon)^{\frac{2}{1+3\nu}})$ gradient evaluations established in FGM for an ε-solution
- **improves** the number of linear optimizations required of CG to $\mathcal{O}((1/\varepsilon)^{\frac{4}{1+3\nu}})$
- allows linear optimization problems to be solved approximately
- achievable by novel parameter choice

Summary of Advantages

Advantages over FGM

- removes need for projections

Summary of Advantages

Advantages over FGM

- removes need for projections

Advantages over Hölder CG

- reduces gradient evaluations required
- reduces linear optimizations required
- provides support for $\nu = 0$

Summary of Advantages

Advantages over FGM

- removes need for projections

Advantages over Hölder CG

- reduces gradient evaluations required
- reduces linear optimizations required
- provides support for $\nu = 0$

Advantages over CGS

- no longer requires knowledge of L in the smooth case
- provides additional application for $\nu \in [0, 1)$
- allows usage of inexact linear optimization solvers
- allows for possibility of early termination with exit criterion

Numerical Experiments - Convex Hull

We consider the problem

$$\min_{x \in \mathrm{conv}(V)} f(x) := ||Ax - b||_2$$

with $V = \{v_1, \ldots, v_p\} \subseteq \mathbb{R}^n$, $\mathrm{conv}(V) := \{x \in \mathbb{R}^n : \exists \lambda \in \Delta_p \text{ s.t. } x = \sum_{j=1}^p \lambda_i v_i\}$, and $\Delta_p := \{\lambda \in \mathbb{R}^p : \sum_{i=1}^p \lambda_i = 1, \lambda_i \geq 0\}$ is the standard simplex.

Numerical Experiments - Convex Hull

We consider the problem

$$\min_{x \in \text{conv}(V)} f(x) := ||Ax - b||_2$$

with $V = \{v_1, \dots, v_p\} \subseteq \mathbb{R}^n$, $\text{conv}(V) := \{x \in \mathbb{R}^n : \exists \lambda \in \Delta_p \text{ s.t. } x = \sum_{j=1}^p \lambda_i v_i\}$, and $\Delta_p := \{\lambda \in \mathbb{R}^p : \sum_{i=1}^p \lambda_i = 1, \lambda_i \geq 0\}$ is the standard simplex.

		UCGS				CG		
n	d	**GE**	**LO**	**Time**	**Error**	**Iter**	**Time**	**Error**
2500	0.2	66	2690	6.71	$9.945e-4$	572	13.42	$9.7086e1$
2500	0.4	60	3679	9.08	$9.976e-4$	524	18.17	$1.404e2$
2500	0.6	62	245	2.64	$9.678e-4$	146	5.29	$5.598e2$
2500	0.8	57	3176	8.45	$9.768e-4$	399	16.93	$2.400e2$
5000	0.2	71	286	7.13	$9.882e-4$	178	14.32	$6.037e2$
5000	0.4	42	52	4.89	$9.585e-4$	84	9.81	$1.689e3$
5000	0.6	68	4564	36.14	$9.727e-4$	483	72.40	$3.527e2$
5000	0.8	67	419	12.91	$9.815e-4$	161	25.94	$1.165e3$
10000	0.2	85	12269	150.51	$9.96e-4$	915	301.21	$2.449e2$
10000	0.4	69	12614	157.39	$9.916e-4$	636	315.27	$4.734e2$
10000	0.6	70	16063	205.87	$9.821e-4$	653	412.14	$5.423e2$
10000	0.8	69	12707	180.65	$9.862e-4$	473	361.73	$8.162e2$

Numerical Experiments - Spectrahedron

For our second experiment, we solve the problem

$$\min_{X \in \mathbf{Spe}_n} f(X) := \sum_{i=1}^{m} ||X - A_i||_2$$

where $\text{Spe}_n := \{X \in \mathbb{R}^{n \times n} : \text{tr}(X) = 1, X \succeq 0\}$ and $A_i \in \text{Spe}_n$ for each $i = 1, \ldots, m$.

Numerical Experiments - Spectrahedron

For our second experiment, we solve the problem

$$\min_{X \in \mathsf{Spe}_n} f(X) := \sum_{i=1}^{m} ||X - A_i||_2$$

where $\mathrm{Spe}_n := \{X \in \mathbb{R}^{n\times n} : \mathrm{tr}(X) = 1, X \succeq 0\}$ and $A_i \in \mathrm{Spe}_n$ for each $i = 1, \ldots, m$.

		UCGS				CG		
n	m	GE	LO	Time	Error	Iter	Time	Error
50	50	1354	8493	9.87	$9.992e-4$	6908	19.74	$6.073e-3$
50	100	1767	11138	13.09	$9.994e-4$	7038	26.19	$1.172e-2$
50	200	2425	15173	25.39	$9.995e-4$	8273	50.79	$2.271e-2$
100	50	1836	13056	159.61	$9.980e-4$	11648	319.25	$3.225e-3$
100	100	2347	16816	216.59	$9.990e-4$	13372	433.20	$5.634e-3$
100	200	3296	23836	310.16	$9.984e-4$	16053	620.36	$9.892e-3$
200	50	1722	33673	470.71	$9.989e-4$	15966	941.43	$3.308e-3$
200	100	2314	46323	730.69	$9.994e-4$	17033	1461.42	$6.870e-3$
200	200	3154	64511	1086.42	$9.992e-4$	19762	2172.85	$1.015e-2$

References

A. Nemirovski and D. Yudin.
Problem complexity and method efficiency in optimization.
Wiley-Interscience Series in Discrete Mathematics. John Wiley, XV, 1983.

Martin Jaggi.
Revisiting Frank-Wolfe: Projection-free sparse convex optimization.
In *ICML (1)*, pages 427–435, 2013.

Guanghui Lan and Yi Zhou.
Conditional gradient sliding for convex optimization.
SIAM Journal on Optimization, 26(2):1379–1409, 2016.

Yu Nesterov.
Complexity bounds for primal-dual methods minimizing the model of objective function.
Mathematical Programming, 171(1):311–330, 2018.