535520: Optimization Algorithms

Lecture 8 — Variance Reduction and Gradient Methods for Constrained Optimization

Ping-Chun Hsieh (謝秉均)

October 28, 2024

This Lecture

1. Accelerating SGD: Variance Reduction

2. Projected Gradient Descent and Frank-Wolfe

- Reading Material:
  - R. Johnson and T. Zhang, “Accelerating stochastic gradient descent using predictive variance reduction,” NIPS 2013
  - M. Jaggi, “Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization,” ICML 2013
  - A. Defazio, F. Bach, and Simon Lacoste-Julien, “SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives,” NIPS 2017
  - Part of the material is adapted from Prof. Yuxin Chen’s lecture notes

AN ALGORITHM FOR QUADRATIC PROGRAMMING

Marguerite Frank and Philip Wolfe[1]
Princeton University

A finite iteration method for calculating the solution of quadratic programming problems is described. Extensions to more general non-linear problems are suggested.

1. INTRODUCTION

The problem of maximizing a concave quadratic function whose variables are subject to linear inequality constraints has been the subject of several recent studies, from both the computational side and the theoretical (see Bibliography). Our aim here has been to develop a method for solving this non-linear programming problem which should be particularly well adapted to high-speed machine computation.

The quadratic programming problem as such, called PI, is set forth in Section 2.

We find in Section 3 that with the aid of generalized Lagrange multipliers the solutions of PI can be exhibited in a simple way as parts of the solutions of a new quadratic programming problem, called PII, which embraces the multipliers. The maximum sought in PII is known to be zero. A test for the existence of solutions to PI arises from the fact that the boundedness of its objective function is equivalent to the feasibility of the (linear) constraints of PII.

In Section 4 we apply to PII an iterative process in which the principal computation is the simplex method change-of-basis. One step of our "gradient and interpolation" method, given an initial feasible point, selects by the simplex routine a secondary basic feasible point whose projection along the gradient of the objective function at the initial point is sufficiently large. The point at which the objective is maximized for the segment joining the initial and secondary points is then chosen as the initial point for the next step.

(Frank and Wolfe, 1956)

Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization

Martin Jaggi JAGGI@CMAP.POLYTECHNIQUE.FR
CMAP, École Polytechnique, Palaiseau, France

Abstract

We provide stronger and more general primal-dual convergence results for Frank-Wolfe-type algorithms (a.k.a. conditional gradient) for constrained convex optimization, enabled by a simple framework of duality gap certificates. Our analysis also holds if the linear subproblems are only solved approximately (as well as if the gradients are inexact), and is proven to be worst-case optimal in the sparsity of the obtained solutions.

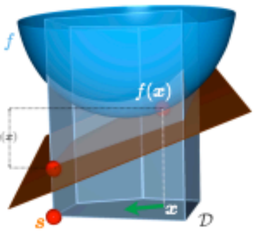
On the application side, this allows us to unify a large variety of existing sparse greedy methods, in particular for optimization over convex hulls of an atomic set, even if those sets can only be approximated, including sparse (or structured sparse) vectors or matrices, low-rank matrices, permutation matrices, or max-norm bounded matrices.

We present a new general framework for convex optimization over *matrix factorizations*, where every Frank-Wolfe iteration will consist of a low-rank update, and discuss the broad application areas of this approach.

Algorithm 1 Frank-Wolfe (1956)

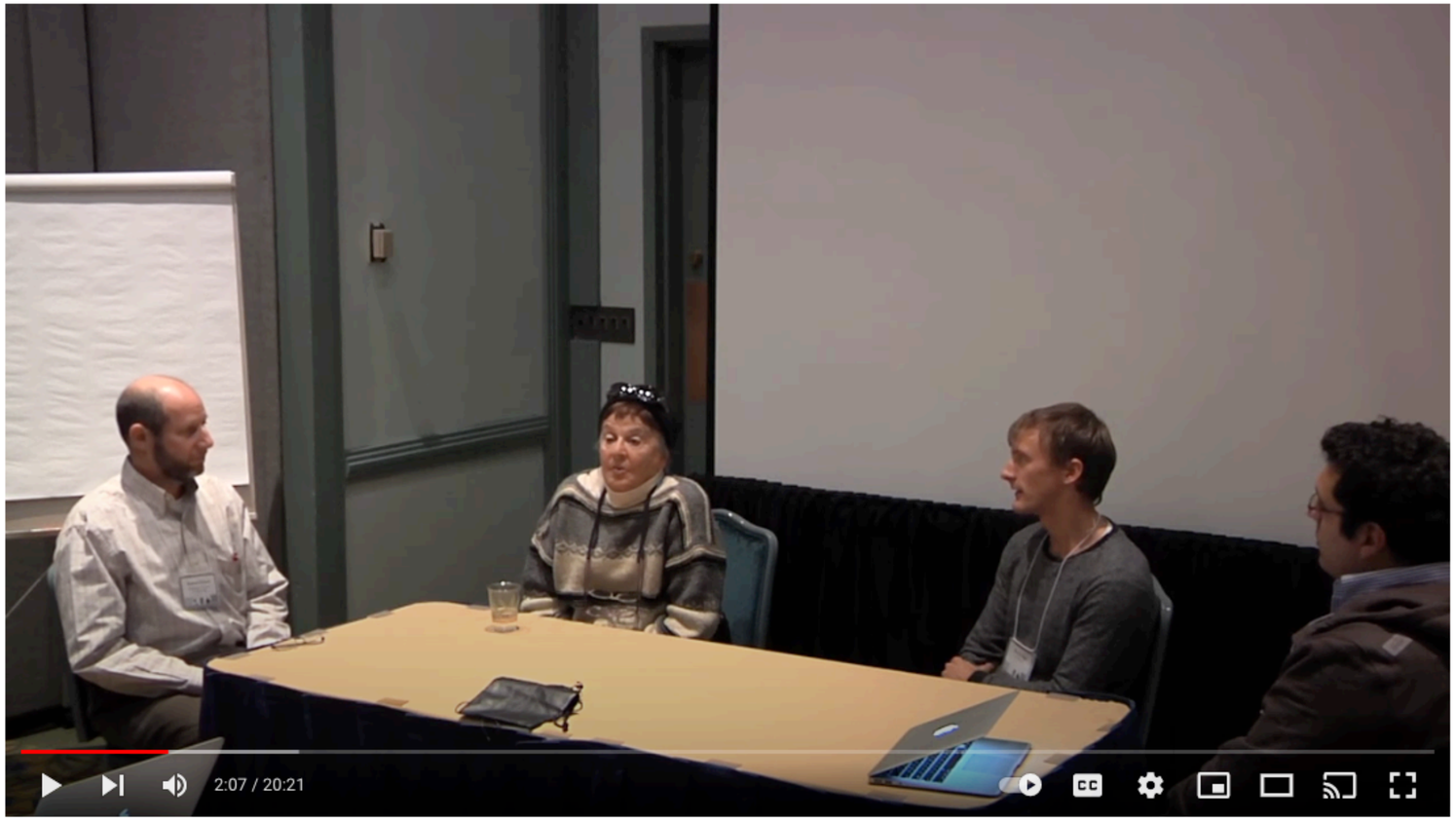
Let $\boldsymbol{x}^{(0)} \in \mathcal{D}$
for $k = 0 \dots K$ **do**
  Compute $\boldsymbol{s} := \arg\min_{\boldsymbol{s} \in \mathcal{D}} \langle \boldsymbol{s}, \nabla f(\boldsymbol{x}^{(k)}) \rangle$
  Update $\boldsymbol{x}^{(k+1)} := (1-\gamma)\boldsymbol{x}^{(k)} + \gamma \boldsymbol{s}$, for $\gamma := \frac{2}{k+2}$
end for

A step of this algorithm is illustrated in the inset figure: At a current position $\boldsymbol{x}$, the algorithm considers the linearization of the objective function, and moves towards a minimizer of this linear function (taken over the same domain).



In terms of convergence, it is known that the iterates of Algorithm 1 satisfy $f(\boldsymbol{x}^{(k)}) - f(\boldsymbol{x}^*) \leq O(\frac{1}{k})$, for $\boldsymbol{x}^*$ being an optimal solution to (1) (Frank & Wolfe, 1956; Dunn & Harshbarger, 1978). In recent years, Frank-Wolfe-type methods have re-gained interest in several areas, fueled by the good scalability, and the crucial property that Algorithm 1 maintains its iterates as a convex

(Jaggi, ICML 2013)



(An interview at NIPS 2013 Workshop)

- Her research was originally on linear algebra (no optimization background)
- Later on, she met Kuhn and Tucker and learned about constrained quadratic programs (QP)
- She accidentally solved QP and Wolfe established the convergence result

https://www.youtube.com/watch?v=24e08AX9Eww

Variance Reduction: Accelerating SGD for Finite-Sum Problems

Review: General Intuition Behind Variance Reduction

- Example: SGD for empirical risk minimization

$$\min_{x\in X} \; F(x) := \frac{1}{n}\sum_{i=1}^{n} f(x; d_i) \qquad (\{d_i\}_{i=1}^{n} \text{ are data samples})$$

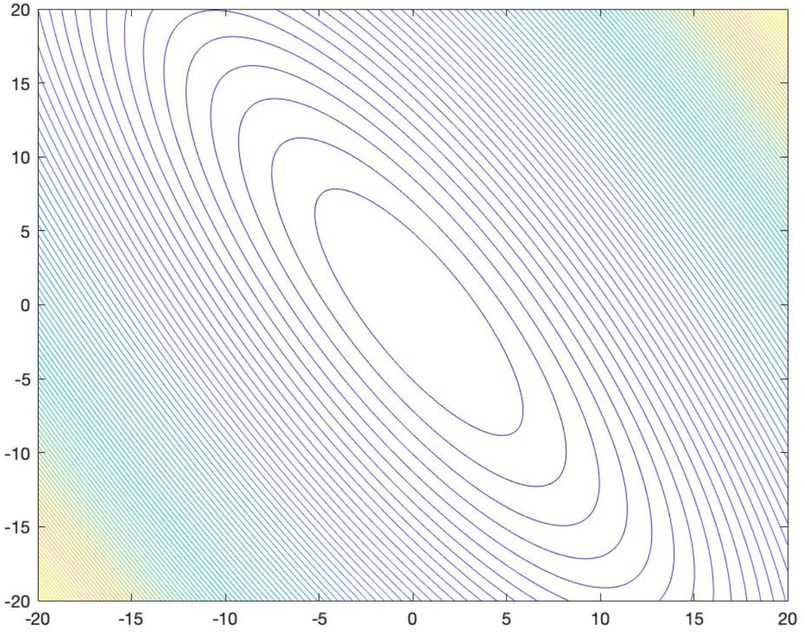
Vanilla SGD:

$$x_{k+1} = x_k - \eta \nabla f(x; d_{I_k}), \quad I_k \sim \text{Unif}(1,n)$$

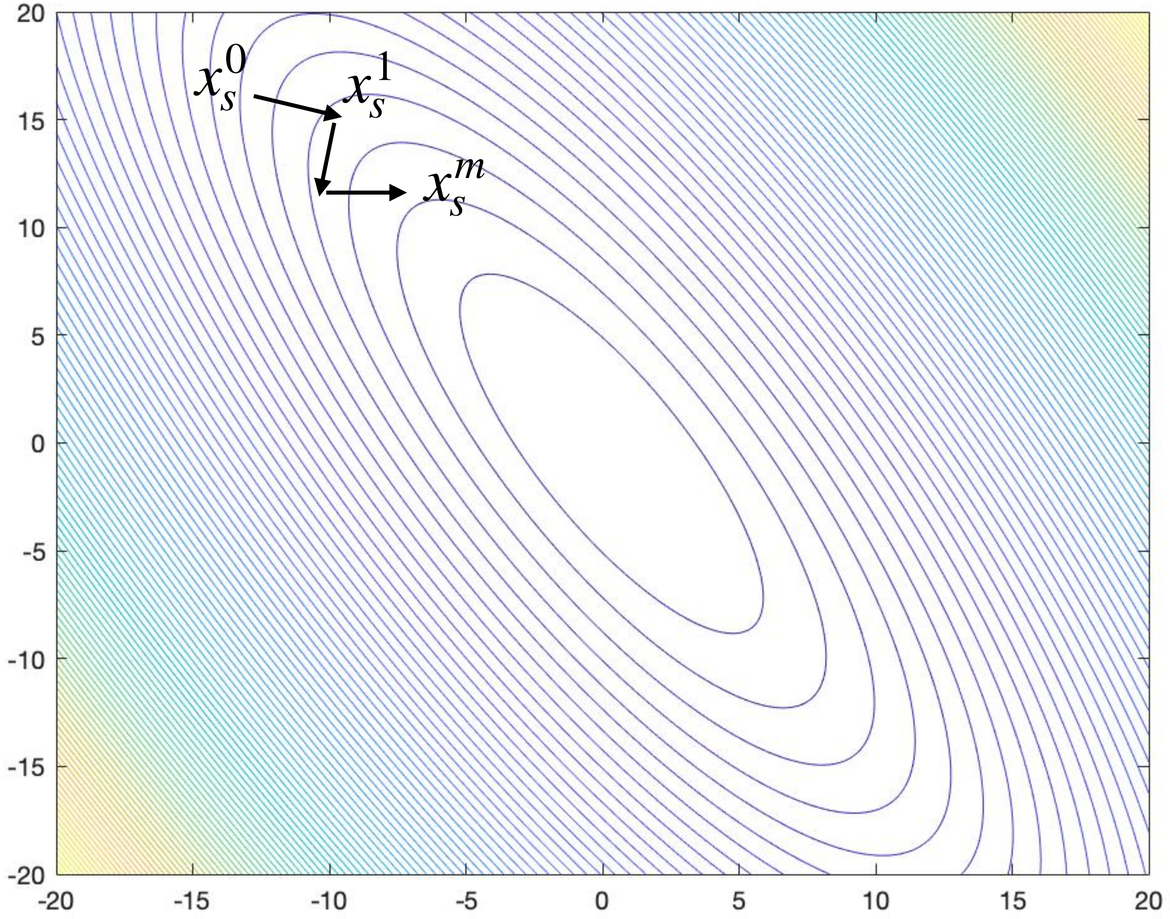
Variance Reduction:

$$x_{k+1} = x_k - \eta\left(\nabla f(x; d_{I_k}) - \nu_k \right), \quad I_k \sim \text{Unif}(1,n)$$

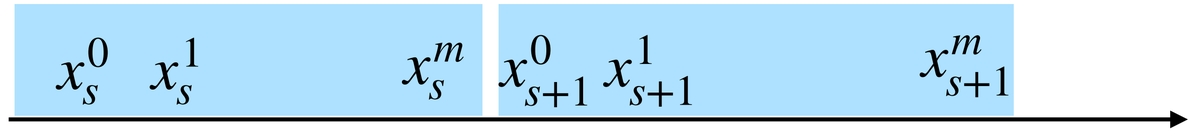
where ν_k and $\nabla f(x; d_{I_k})$ are positively correlated (why?)



Review: Intuition Behind SVRG



Idea: Query "true gradient" only occasionally



In each stochastic update, use

$$x_s^{k+1} = x_s^k - \eta\left(\nabla f(x_s^k; d_{I_k}) - \nu_k\right), \quad I_k \sim \text{Unif}(1,n)$$

$$\nu_k = \nabla f(x_{old}; d_{I_k}) - \nabla F(x_{old})$$

Key observation: If $x_s^k \approx x_{old}$, then $\nabla f(x_s^k; d_{I_k}) - \nu_k \approx \nabla F(x_s^k)$

Review: Pseudo Code of SVRG

Notation: $\nabla f(x; d_i) \equiv \nabla f_i(x)$

Outer loop for snapshots

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

2: $\boldsymbol{x}_s^{\text{old}} \leftarrow x_s^j$ and compute $\underbrace{\nabla F(\boldsymbol{x}_s^{\text{old}})}$ // update snapshot

($j \sim \text{Unif}(0, \cdots, m-1)$) batch gradient

3: initialize $\boldsymbol{x}_s^0 \leftarrow \boldsymbol{x}_s^{\text{old}}$

4: **for** $\underbrace{t = 0, \ldots, m-1}$ **do** Inner loop for iterate updates

each epoch contains m iterations

5: choose i_t uniformly from $\{1, \ldots, n\}$, and

$$\boldsymbol{x}_s^{t+1} = \boldsymbol{x}_s^t - \eta\{\underbrace{\nabla f_{i_t}(\boldsymbol{x}_s^t) - \nabla f_{i_t}(\boldsymbol{x}_s^{\text{old}})}_{\text{stochastic gradient}} + \nabla F(\boldsymbol{x}_s^{\text{old}})\}$$

- Question: How many gradient computations are needed under SVRG in one epoch? How about SGD?

(Slide Credit: Yuxin Chen)

Review: SGD / Batch GD / SVRG

- Consider strongly convex "empirical risk minimization" with n samples

	Iteration Complexity	Per-iteration Cost	Total Computation Cost
Batch GD	$\kappa \log \frac{1}{\epsilon}$	n	$n\kappa \log \frac{1}{\epsilon}$
SGD	$\kappa^2 \frac{1}{\epsilon}$	1	$\kappa^2 \frac{1}{\epsilon}$
SVRG			$(n+\kappa)\log \frac{1}{\epsilon}$

($\kappa := L/\mu$ is the condition number)

Convergence Rate of SVRG for Finite-Sum Problems

Theorem (Convergence of SVRG):

Suppose the following conditions hold.

(1) All $f_i(x)$ are convex and L-smooth

(2) $F(x)$ is μ-strongly convex

(3) The snapshot period m is sufficiently large and $\eta < 1/(4L)$ such that

$$\alpha = \frac{1}{\mu\eta(1-2L\eta)m} + \frac{2L\eta}{1-2L\eta} < 1$$

Then, SVRG achieves $\mathbb{E}[F(x_s)] - F(x^*) \leq \alpha^s \Big(F(x_0) - F(x^*) \Big)$

Two Helper Lemmas

Lemma 1 (PL-like Condition for Sum-Finite Problems):

$$\frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(x) - \nabla f_i(x^*)\|^2 \leq 2L \cdot \Big(F(x) - F(x^*)\Big)$$

- Proof already discussed in Lecture 7

Lemma 2 (Quantifying Variance under SVRG):

$$\mathbb{E}[\|g_s^t\|^2] \leq 4L \cdot \Big(F(x_s^{t-1}) - F(x^*) + F(x_s^{old}) - F(x^*)\Big)$$

- Suppose $x_s^t \approx x_s^{old} \approx x^*$, then what can we say about $\mathbb{E}[\|g_s^t\|^2]$?

Proof of Lemma

Recall that $g_s^t = \nabla f_{i_t}(x_s^{t-1}) - \nabla f_{i_t}(x_s^{old}) + \nabla F(x_s^{old})$

$$E[\|g_s^t\|^2] \le 2 \cdot E\left[\|\nabla f_{i_t}(x_s^{t-1}) - \nabla f_{i_t}(x^*)\|^2\right]$$

$$+ 2 \cdot E\left[\left\|\left(\nabla f_{i_t}(x_s^{old}) - \nabla f_{i_t}(x^*)\right) - \underbrace{\nabla F(x_s^{old})}_{=}\right\|^2\right]$$

$$E\left[\nabla f_{i_t}(x_s^{old}) - \nabla f_{i_t}(x^*)\right]$$

$$\le 2 \cdot E\left[\|\nabla f_{i_t}(x_s^{t-1}) - \nabla f_{i_t}(x^*)\|^2\right] + 2 \cdot E\left[\|\nabla f_{i_t}(x_s^{old}) - \nabla f_{i_t}(x^*)\|^2\right] \text{ (why?)}$$

$$\le 4L \cdot \left(\left(F(x_s^{t-1}) - F(x^*)\right) + \left(F(x_s^{old}) - F(x^*)\right)\right) \text{ (why?)}$$

A Useful Property

For any random vector Z, we always have

$$E\left[\|Z - E[Z]\|^2\right] = E\left[\|Z\|^2\right] - \|E[Z]\|^2 \leq E\left[\|Z\|^2\right].$$

Let’s put everything together and prove the convergence of SVRG

Proof of Convergence of SVRG

Step 1:

$$\mathbb{E}\left[\|x_s^t - x^*\|^2 \,\middle|\, x_s^{t-1}\right] = \mathbb{E}\left[\|x_s^{t-1} - \eta \cdot g_s^t - x^*\|^2 \,\middle|\, x_s^{t-1}\right]$$

$$= \|x_s^{t-1} - x^*\|^2 - 2\cdot\eta\cdot(x_s^{t-1} - x^*)^T \cdot \mathbb{E}\left[g_s^t \,\middle|\, x_s^{t-1}\right] + \eta^2 \cdot \mathbb{E}\left[\|g_s^t\|^2 \,\middle|\, x_s^{t-1}\right]$$

$$\leq \|x_s^{t-1} - x^*\|^2 - 2\cdot\eta\cdot\underbrace{(x_s^{t-1} - x^*)\cdot\nabla F(x_s^{t-1})}_{\geq F(x_s^{t-1}) - F(x^*)} + \eta^2 \cdot 4L\Big(F(x_s^{t-1}) - F(x^*) + F(x_s^{old}) - F(x^*)\Big)$$

$$\leq \|x_s^{t-1} - x^*\|^2 - 2\cdot\eta\cdot\Big(F(x_s^{t-1}) - F(x^*)\Big) + \eta^2 4L\Big(F(x_s^{t-1}) - F(x^*) + F(x_s^{old}) - F(x^*)\Big)$$

$$= \|x_s^{t-1} - x^*\|^2 - 2\cdot\eta\cdot(1 - 2L\eta)\cdot\Big(F(x_s^{t-1}) - F(x^*)\Big) + 4\eta^2 L\cdot\Big(F(x_s^{old}) - F(x^*)\Big) \quad -(*)$$

(Cont.)

Step 2: Consider a fixed epoch s

By taking the sum of (*) over $t=1,\cdots,m$ and taking the total expectation,

$$\mathbb{E}\left[\|X_s^m - x^*\|^2\right] + 2\eta(1-2L\eta)m \cdot \mathbb{E}\left[F(x_{s+1}^{old}) - F(x^*)\right]$$

$$\leq \mathbb{E}\left[\|X_s^0 - x^*\|^2\right] + 4Lm\eta^2 \cdot \mathbb{E}\left[F(x_s^{old}) - F(x^*)\right] \cdots (\quad)$$

$$= \mathbb{E}\left[\|X_s^{old} - x^*\|^2\right] + 4Lm\eta^2 \cdot \mathbb{E}\left[F(x_s^{old}) - F(x^*)\right] \cdots (\quad)$$

$$\leq \frac{2}{\mu}\mathbb{E}\left[F(x_s^{old}) - F(x^*)\right] + 4Lm\eta^2 \cdot \mathbb{E}\left[F(x_s^{old}) - F(x^*)\right] \cdots (\quad)$$

$$= \left(\frac{2}{\mu} + 4Lm\eta^2\right) \cdot \mathbb{E}\left[F(x_s^{old}) - F(x^*)\right]$$

(Cont.).

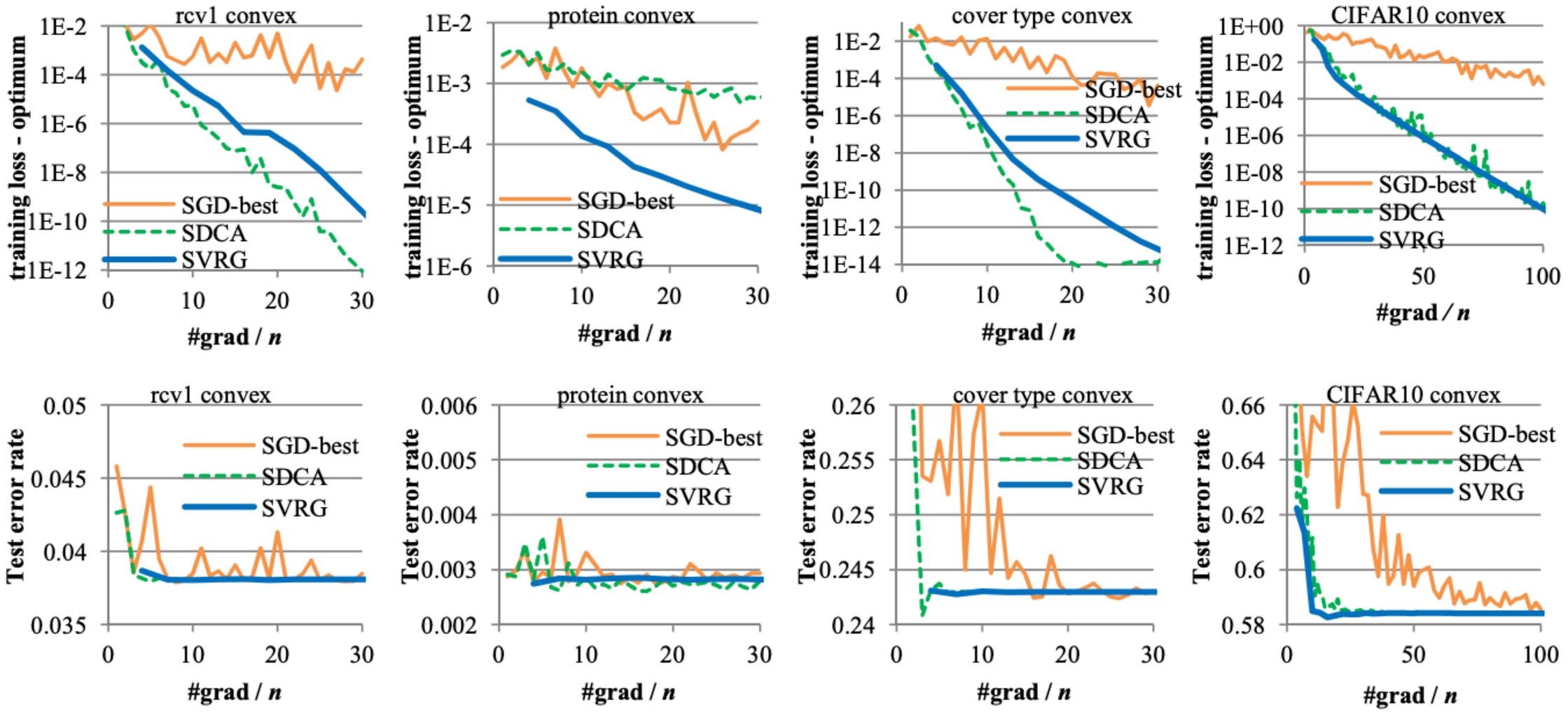
Step3: Consequently, we have

$$\mathbb{E}\left[F(x_{s+1}^{old}) - F(x^*)\right]$$

$$\leq \frac{\frac{2}{\mu} + 4Lm\eta^2}{2\eta(1-2L\eta)m} \cdot \mathbb{E}\left[F(x_s^{old}) - F(x^*)\right] - \underbrace{\frac{1}{2\eta(1-2L\eta)m}\mathbb{E}\left[\|x_s^m - x^*\|^2\right]}_{\geq 0}$$

$$\leq \left(\frac{1}{\mu\eta(1-2L\eta)m} + \frac{2L\eta}{1-2L\eta}\right) \cdot \mathbb{E}\left[F(x_s^{old}) - F(x^*)\right]$$

Numerical Comparison: SGD and SVRG



R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction, NIPS 2013

A General Recipe of Variance Reduction

- Suppose we'd like estimate $E[X]$ (where X is a random variable)
- A general VR approach: If we could compute efficiently $E[Y]$ for another random variable Y that is highly correlated with X, then we could use an estimator as

$$\theta_\alpha = \alpha(X - Y) + E[Y]$$

- Question: Expected value and variance of θ_α?

$$\mathbb{E}[\theta_\alpha] = \alpha\mathbb{E}[X] + (1 - \alpha)\mathbb{E}[Y]$$

$$\mathrm{Var}(\theta_\alpha) = \alpha^2(\mathrm{Var}(X) + \mathrm{Var}(X) - 2\mathrm{Cov}(X, Y))$$

- By varying α from 0 to 1, what do we have regarding variance and bias?
- How to interpret SVRG?

A General Recipe of Variance Reduction: SAGA vs SVRG

SVRG $\quad x_s^{k+1} = x_s^k - \eta\Big(\nabla f(x_s^k; d_{I_k}) - \nabla f(x_{old}; d_{I_k}) - \nabla F(x_{old}) \Big), \quad I_k \sim \text{Unif}(1,n)$

SAGA $\quad x^{k+1} = x^k - \eta\Big(\nabla f(x^k; d_{I_k}) - \nabla f(\phi_{I_k}^k; d_{I_k}) - \frac{1}{n}\sum_{i=1}^{n} \nabla f(\phi_i^k; d_i) \Big), \quad I_k \sim \text{Unif}(1,n)$

where $\nabla f(\phi_i^k; d_i)$ are past stochastic gradients stored in the lookup table

---

General recipe: $\theta_\alpha = \alpha(X - Y) + E[Y]$.

What are X, Y, α in SAGA?

Sample ID	∇f_i
1	
2	
3	
4	

A. Defazio, F. Bach, and Simon Lacoste-Julien, “SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives,” NIPS 2017

Various VR Methods

- SAG (**S**tochastic **A**veraged **G**radient)
- SAGA [Defazio, Bach, and Lacoste-Julien, NIPS 2014]
- SARAH (**S**toch**A**stic **R**ecursive gr**A**dient algorit**H**m) [Nguyen, Liu, Scheinberg, Takac, ICML 2017]
- SDCA (**S**tochastic **D**ual **C**oordinate **A**scent) [Johnson and Zhang, NIPS 2013; Shalev-Shwartz, ICML 2016]
- And more! (See the candidate paper list)

Example: SARAH

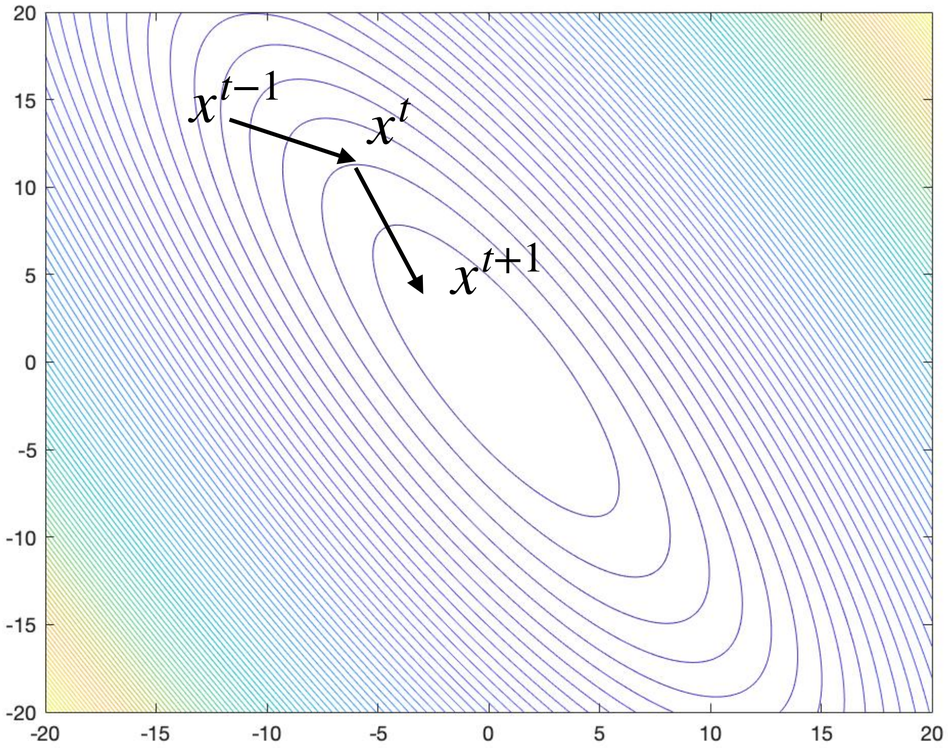
Question: Can we do VR without taking a snapshot?

- **S**toch**A**stic **R**ecursive gr**A**dient algorit**H**m (SARAH)

$$x^{t+1} = x^t - \eta g^t$$

$$g^t = \left(\nabla f_{I_t}(x^t) - \nabla f_{I_t}(x^{t-1})\right) + g^{t-1}$$

Idea: Recursive updates of gradient estimates



- Question: Why is this reasonable? Any issue?

L. Nguyen, J. Liu, K. Scheinberg, M. Takac, “SARAH: A novel method for machine learning problems using stochastic recursive gradient,” ICML 2017

Pseudo Code of SARAH

Notation: $\nabla f(x; d_i) \equiv \nabla f_i(x)$

1: **for** $s = 1, 2, \ldots, S$ **do** — Outer loop for periodic resets

2: $\quad \boldsymbol{x}_s^0 \leftarrow \boldsymbol{x}_{s-1}^{m+1}$, and compute $\underbrace{\boldsymbol{g}_s^0 = \nabla F(\boldsymbol{x}_s^0)}_{\text{batch gradient}}$ // restart $\boldsymbol{g}$ anew

3: $\quad \boldsymbol{x}_s^1 = \boldsymbol{x}_s^0 - \eta \boldsymbol{g}_s^0$

4: $\quad$ **for** $t = 1, \ldots, m$ **do** — Inner loop for recursive updates

5: $\quad\quad$ choose i_t uniformly from $\{1, \ldots, n\}$

6: $\quad\quad \boldsymbol{g}_s^t = \underbrace{\nabla f_{i_t}(\boldsymbol{x}_s^t) - \nabla f_{i_t}(\boldsymbol{x}_s^{t-1})}_{\text{stochastic gradient}} + \boldsymbol{g}_s^{t-1}$

7: $\quad\quad \boldsymbol{x}_s^{t+1} = \boldsymbol{x}_s^t - \eta \boldsymbol{g}_s^t$

- Question: Why do we need to reset g^t periodically?

(Slide Credit: Yuxin Chen)

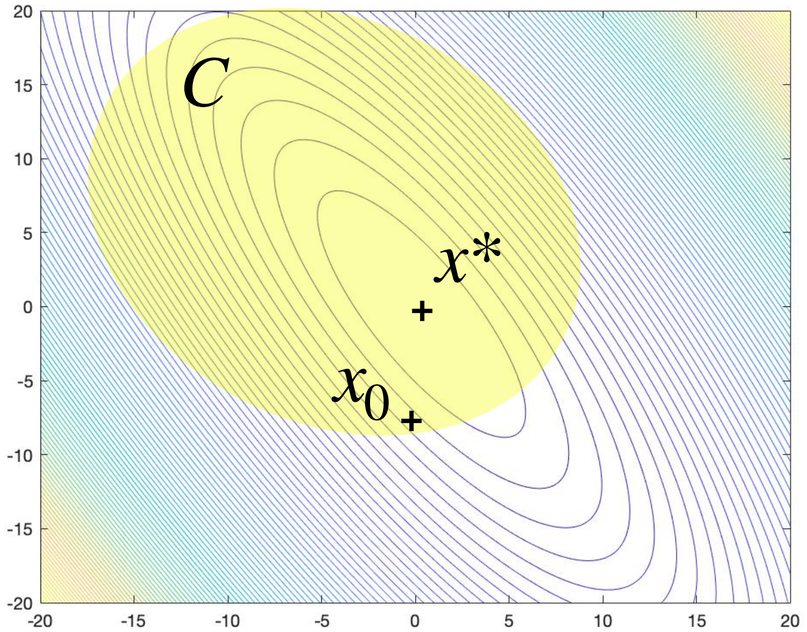
Gradient Methods for Constrained Problems

GD and Constrained Problems

- Can we directly apply GD to constrained problems? Any issues?

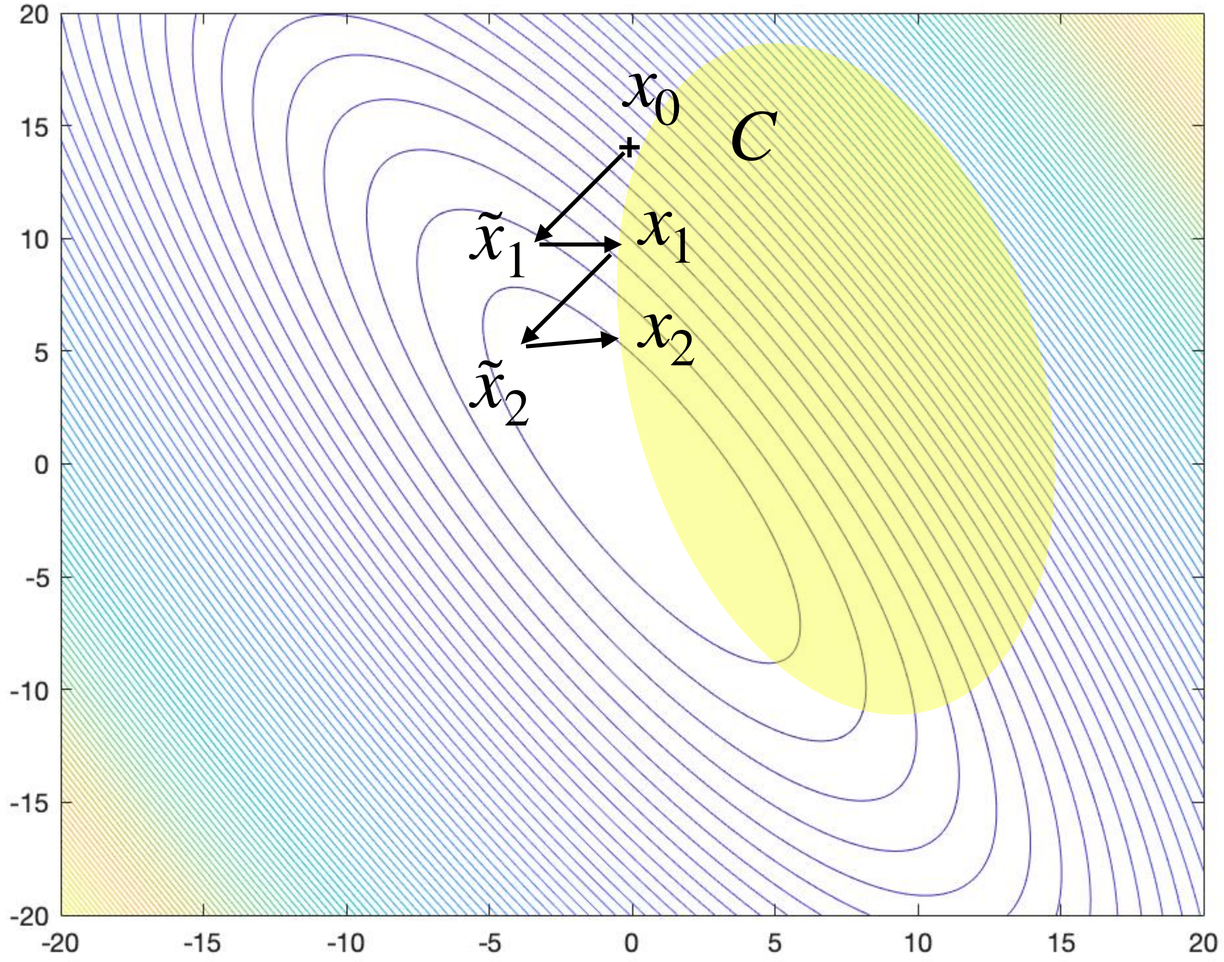
$$\min \quad f(x)$$

subject to $x \in C$



- Challenge: How to enforce constraint satisfaction?
  - 1. Projected GD: Enforce constraints after the GD step
  - 2. Frank-Wolfe (aka Conditional Gradient): Enforce constraints within the GD step

Projected Gradient Descent (PGD)



- Under PGD, the iterates are updated as

$$x_{t+1} = \Pi_C(\underbrace{x_t - \eta_t \nabla f(x_t)}_{=:\tilde{x}_{t+1}})$$

where the projection is defined as

$$\Pi_C(x) := \arg\min_{c \in C} \|x - z\|$$

- Question: If C is a convex set, then is the projection result unique?
- Question: Is PGD efficient and easy to implement?

Depending on the projection

(In other words, projection step needs to be efficient in order for PGD to be practical)

Examples: “Cheap” Projection Steps

- Notably, the “gradient step” takes $O(d)$ time → The projection step is relatively cheap if it only takes comparable time

Example (non-negativity): $x \geq 0$

Example (simplex): $x \geq 0, \sum_i x_i = r$

Example (linear inequality): $Ax \leq b$

Example (ℓ_p-ball): $\|x\|_p \leq c$

Set	Complexity of Projection
Non-negativity	$O(d)$
Simplex	$O(d)$
ℓ_p**-ball**, $p = 1,2,\infty$	$O(d)$
ℓ_p**-ball**, $p \in]1,2[\cup]2,\infty)$	$O(d/\epsilon^2)$

C. W. Combettes and S. Pokutta, “Complexity of Linear Minimization and Projection on Some Sets,” ICML 2017 (available at https://arxiv.org/pdf/2101.10040)

Intuition Behind PGD #1: Stopping at Local Minimizers?

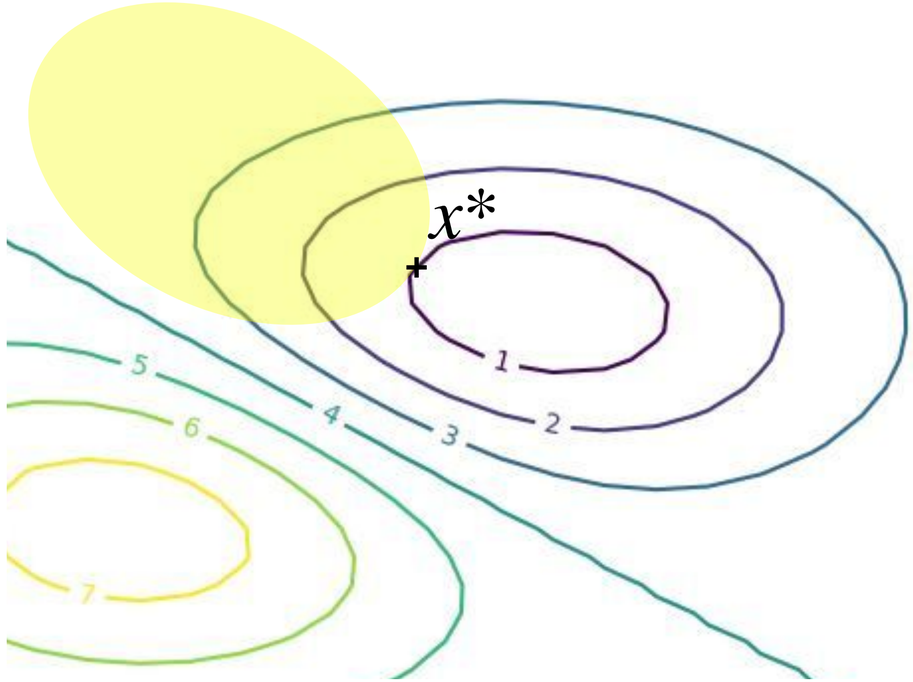
- Recall: FONC-C (Necessary condition for local minimizers of constrained problems)

> *If x^* is a local minimizer of f over a convex feasible set C, then*
>
> $$\nabla f(x^*)^\top (x - x^*) \geq 0, \forall x \in C$$

- The above condition is equivalent to

$$\Big((x^* - \eta \nabla f(x^*)) - x^* \Big)^\top (x - x^*) \leq 0, \forall x \in C$$

This condition holds if and only if x^* is the projection of $x^* - \eta \nabla f(x^*)$ onto C (why?)



Projection Theorem

Theorem (Projection): Let C be a convex set.

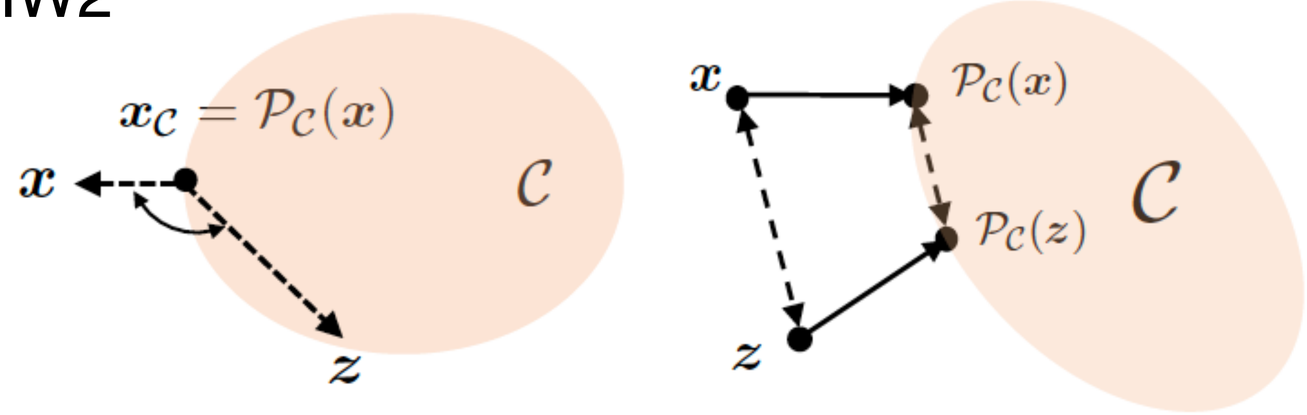
(1) Given some vector $x \in \mathbb{R}^d$, a vector $x_C \in C$ is equal to the projection $\Pi_C(x)$ if and only if

$$(x - x_C)^\top (z - x_C) \leq 0, \quad \forall z \in C$$

(2) The mapping $h : \mathbb{R}^d \to C$ defined by $\Pi_C(x)$ is continuous and non-expansive, i.e.,

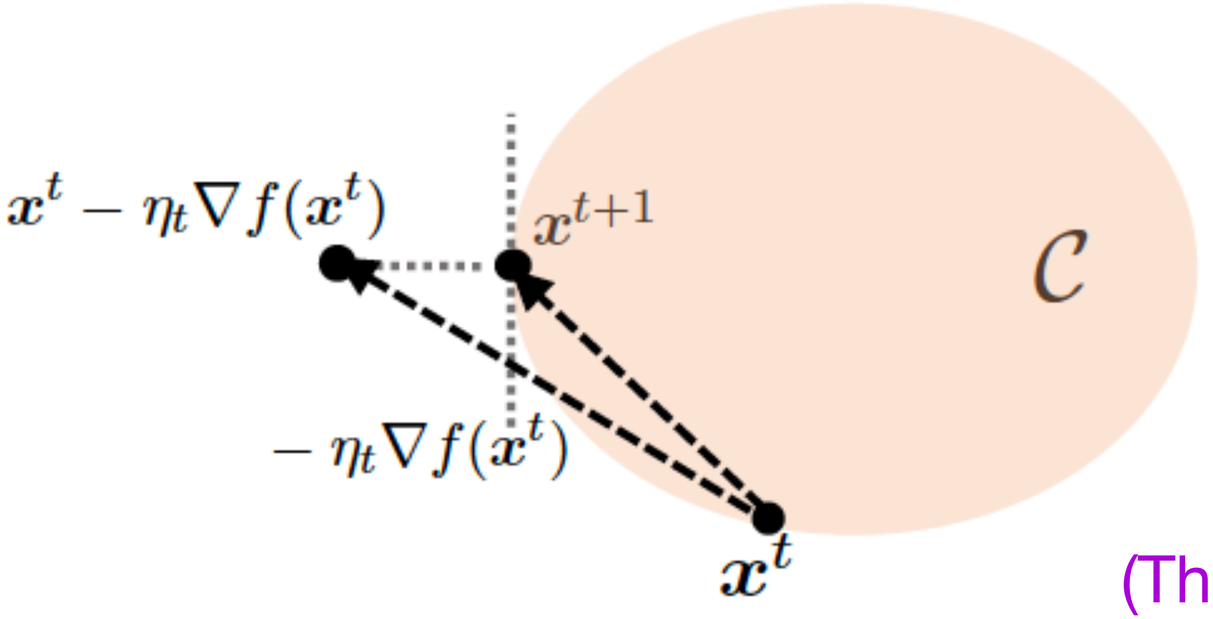
$$\|x_C - z_C\| \leq \|x - z\|, \quad \forall x, z \in \mathbb{R}^d$$

Proof: HW2



(Figure Credit: Yuxin Chen)

Intuition Behind PGD #2: Descent Direction?



From this 2-dimensional toy example, we know

$$-\nabla f(x_t)^\top (x_{t+1} - x_t) \geq 0$$

(That is, $x_{t+1} - x_t$ provides a descent direction)

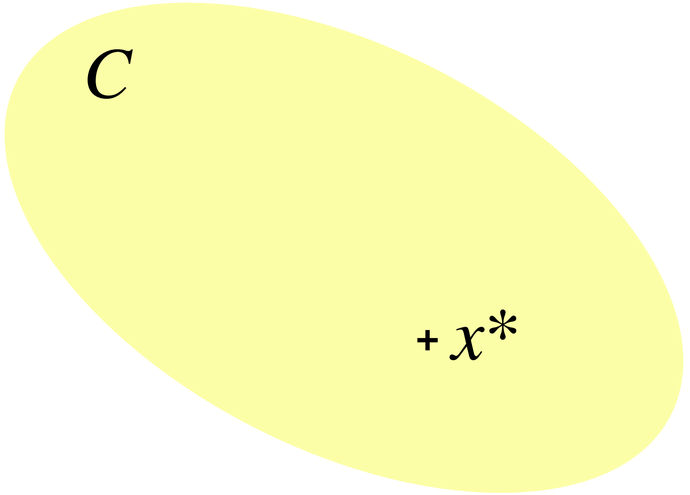
- Can we extend this argument to the general d-dimensional cases? (Projection Theorem!)

(Figure Credit: Yuxin Chen)

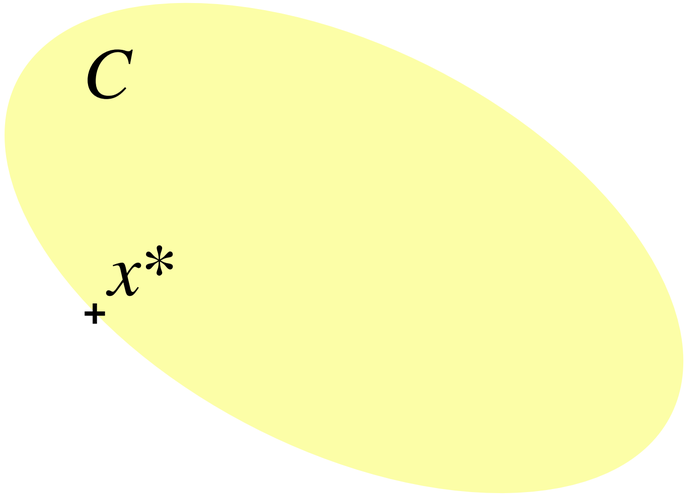
PGD for Strongly-Convex and Smooth Problems

Convergence of PGD: Two Possible Scenarios

- What are the possible scenarios of x^* in constrained problems?
  - Scenario 1: x^* is in the interior of C
    - Is this case challenging for PGD?



  - Scenario 2: x^* is on the boundary of C
    - Why is this case challenging for PGD?



Convergence of PGD: Scenario #1

Theorem (Convergence of PGD):

Suppose x^* is in the interior of C (scenario #1).

Let f be μ-strongly convex and L-smooth. Under PGD with constant step sizes $\eta = 2/(\mu + L)$, we have

$$\|x_t - x^*\| \leqslant \left(\frac{\kappa - 1}{\kappa + 1}\right)^t \cdot \|x_0 - x^*\|$$

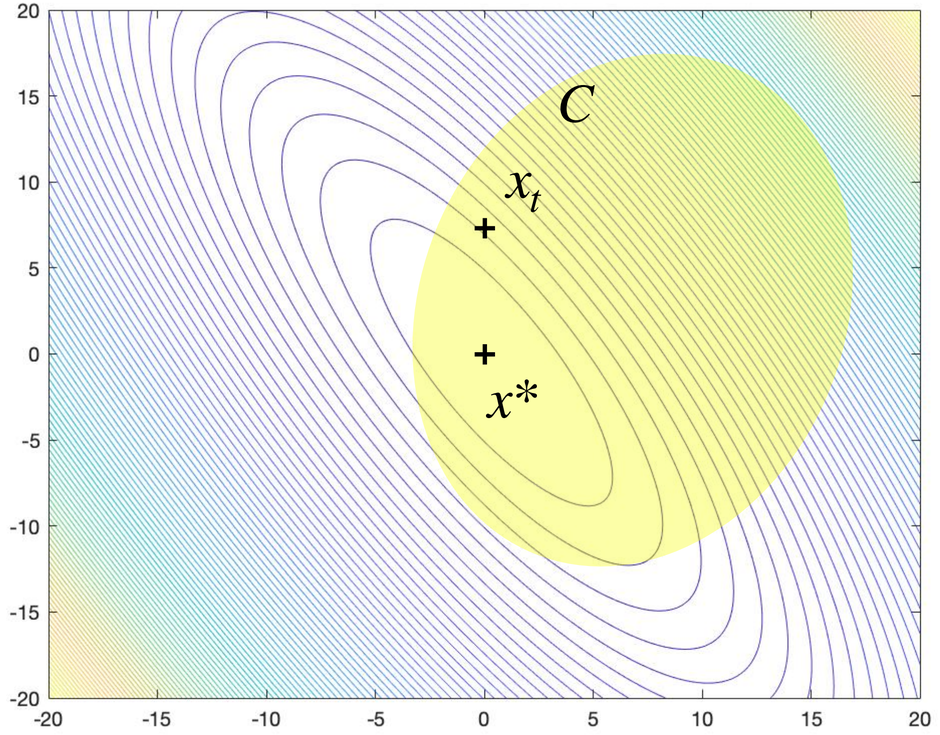
($\kappa := L/\mu$ is called the condition number)

- Question: Is this rate similar to that of GD for unconstrained problems?

Proof of Convergence of PGD: Scenario #1

Step 1: Recall from Lecture 4 that we have shown

$$\left\| \left(x_t - \eta_t \nabla f(x_t) \right) - x^* \right\| \leqslant \left(\frac{\kappa - 1}{\kappa + 1} \right) \cdot \| x_t - x^* \|$$



Step 2: Then, we have

$$\begin{aligned} \left\| x_{t+1} - x^* \right\| &= \left\| \Pi_C \left(x_t - \eta_t \nabla f(x_t) \right) - \Pi_C(x^*) \right\| \\ &\leq \left\| \left(x_t - \eta_t \nabla f(x_t) \right) - x^* \right\| \quad \cdots (\quad) \\ &\leq \left(\frac{\kappa - 1}{\kappa + 1} \right) \cdot \| x_t - x^* \| \end{aligned}$$

Convergence of PGD: Scenarios #1 and #2

Theorem (Convergence of PGD):

Let f be μ-strongly convex and L-smooth. Under PGD with constant step sizes $\eta = 1/L$, we have

$$\|x_t - x^*\|^2 \leqslant \left(1 - \frac{\mu}{L}\right)^t \cdot \|x_0 - x^*\|^2$$

- Question: Comparison with the convergence rate of Scenario #1?

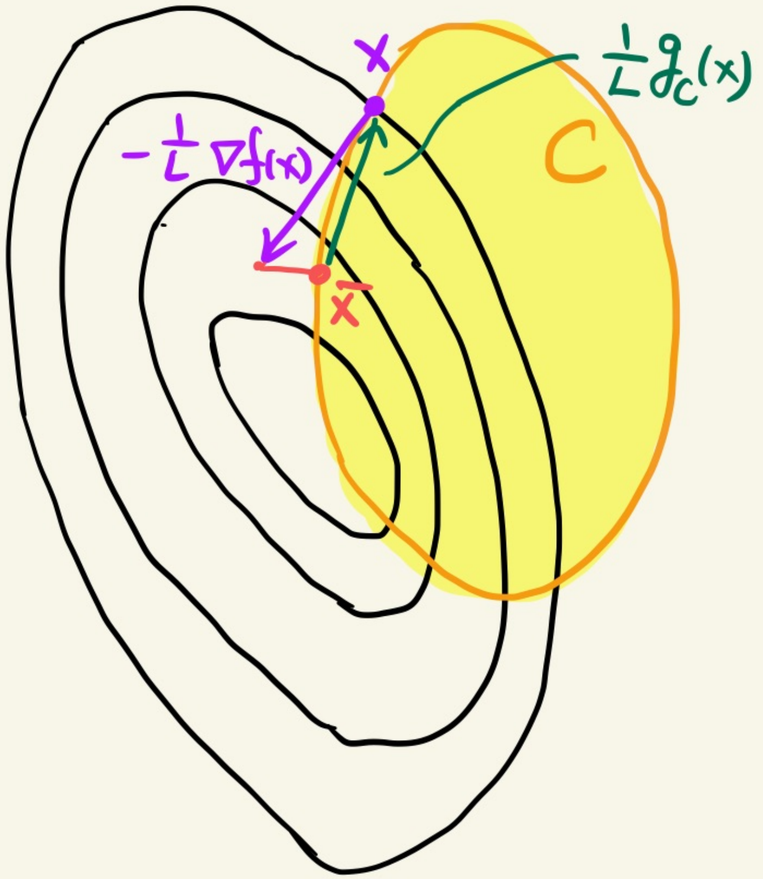
Proof Idea: Rewrite PGD in a form similar to GD.

- Define $\bar{x} := \Pi_C\left(x - \frac{1}{L}\nabla f(x)\right)$

$$g_C(x) := \frac{1}{\eta}(x - \bar{x}) = L(x - \bar{x})$$

- $g_C(x)$ can be viewed as "the generalized $\nabla f(x)$"

  as we know $g_C(x^*) = 0$



Claim: $$g_C(x)^T(x - x^*) \geq \frac{\mu}{2}\|x - x^*\|^2 + \frac{1}{2L}\|g_C(x)\|^2$$ (Intuition?)

(Given this claim, we can reuse the analysis of GD for PGD)

Want: $g_c(x)^T(x-x^*) \geq \frac{\mu}{2}\|x-x^*\|^2 + \frac{1}{2L}\|g_c(x)\|^2$

Proof of Claim:

Step1: $0 \leq f(\bar{x}) - f(x^*)$

$= \left(f(\bar{x}) - f(x)\right) + \left(f(x) - f(x^*)\right)$

why? → $\leq \left(\nabla f(x)^T(\bar{x}-x) + \frac{L}{2}\|\bar{x}-x\|^2\right) + \left(\nabla f(x)^T(x-x^*) - \frac{\mu}{2}\|x-x^*\|^2\right)$

$= \nabla f(x)^T(\bar{x}-x^*) + \frac{1}{2L}\|g_c(x)\|^2 - \frac{\mu}{2}\|x-x^*\|^2$

Step2: We also have $\nabla f(x)^T(\bar{x}-x^*) \leq g_c(x)^T(\bar{x}-x^*)$ since $\left(\bar{x} - \left(x - \frac{1}{L}\nabla f(x)\right)\right)^T(\bar{x}-x^*) \leq 0$

By Combining Step1 and Step2, we can verify the claim □

PGD for Convex and Smooth Problems

Convergence of PGD for Convex and Smooth Problems

Theorem (Convergence of PGD):

Let f be convex and L-smooth. Under PGD with constant step sizes $\eta = 1/L$, we have

$$f(x_t) - f(x^*) \leqslant \frac{3L\|x_0 - x^*\|^2 + \big(f(x_0) - f(x^*)\big)}{t+1}, \quad \forall t \in \mathbb{N}$$

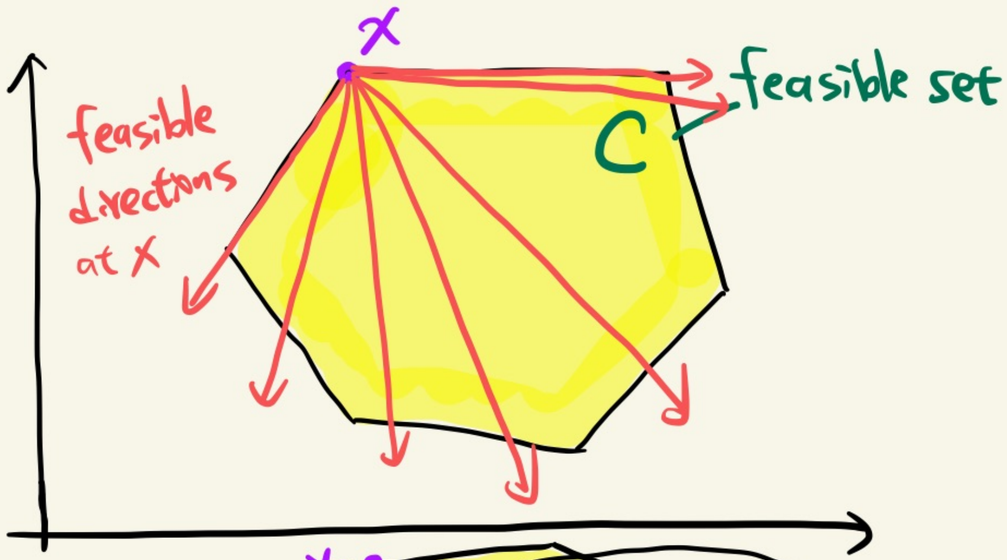
- Remark: PGD achieves $O(1/t)$ rate as GD for unconstrained problems

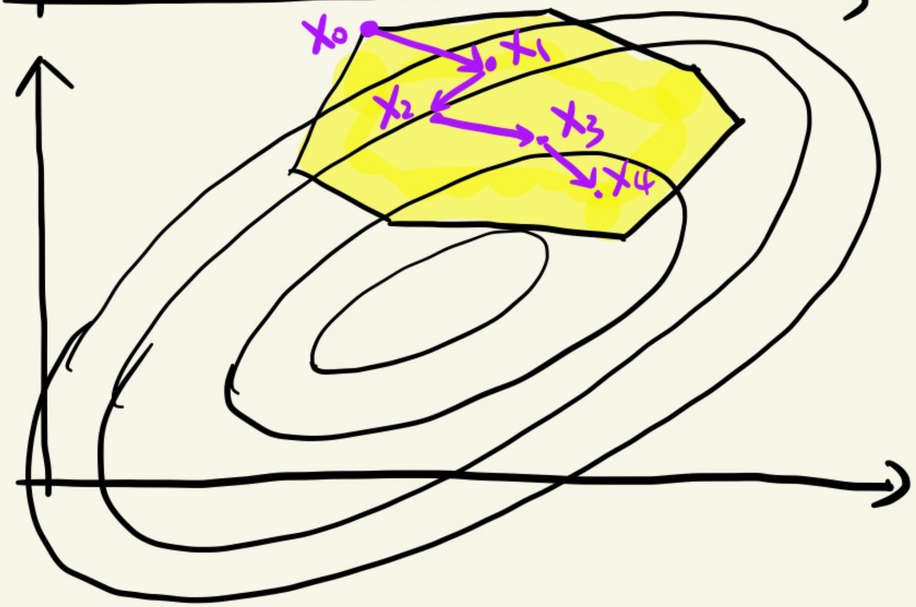
(Proof: HW2 problem for step-by-step analysis)

Can we design a gradient method *without* projection?

Frank-Wolfe Method (or Conditional Gradient)!

Feasible Direction Method





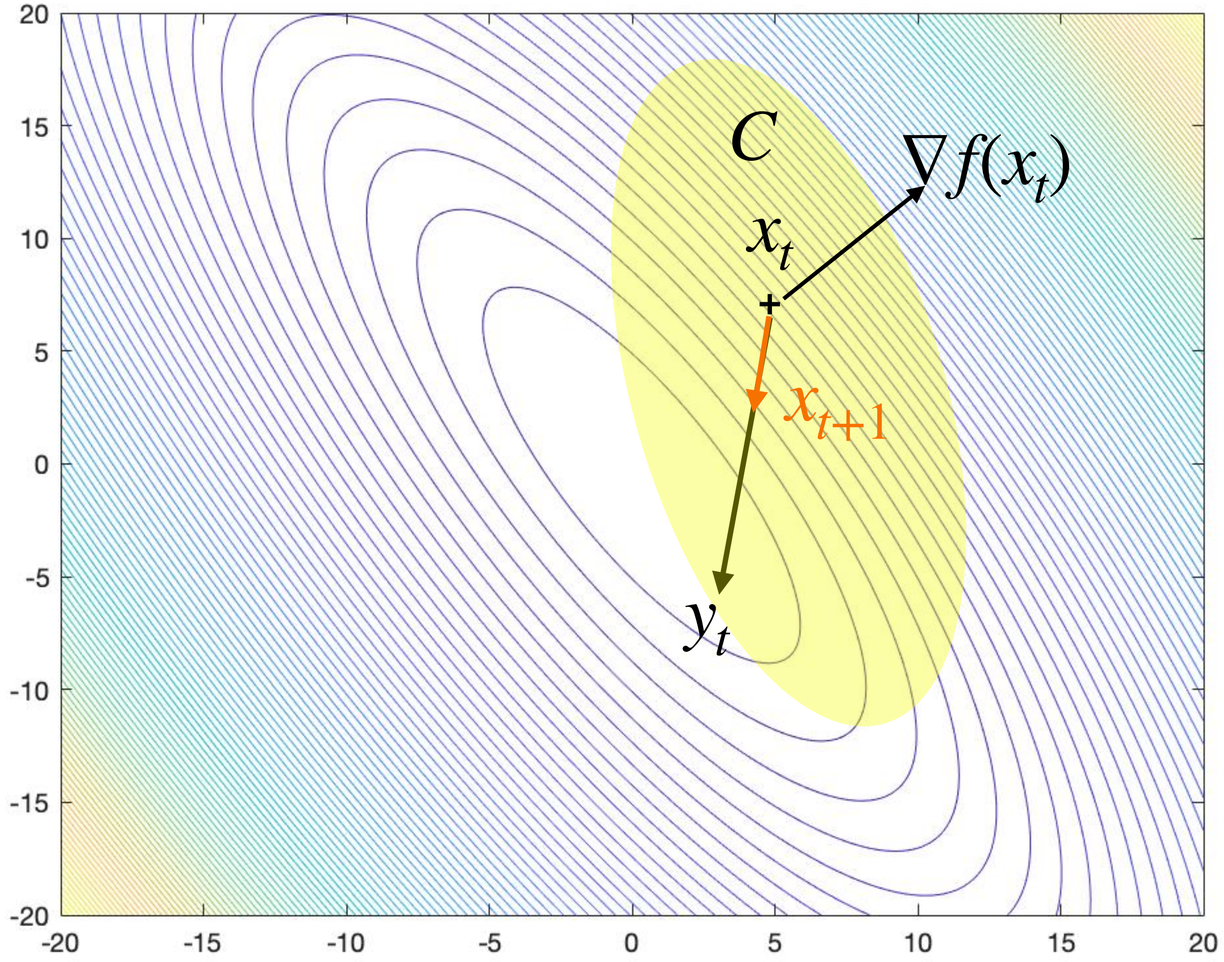
Definition: Given a vector X, we say d is a feasible direction at X if $(X+\alpha d)$ is also feasible for sufficiently small α.

Feasible Direction Method:

$$X_{t+1} = X_t + \alpha_t \cdot d_t,$$

where d_t is a feasible direction with $\nabla f(X_t)^T \cdot d_t < 0$

Frank-Wolfe (FW) Method



- Under FW, the iterates are updated as

$$y_t = \arg\min_{x \in C} \nabla f(x_t)^\top x$$

$$x_{t+1} = (1 - \eta_t) x_t + \eta_t y_t$$

- Question: Is $(y_t - x_t)$ a feasible direction?
- Question: Is FW a feasible direction method?
- Question: Is FW projection-free?

Convergence of Frank-Wolfe for Convex and Smooth Problems

Theorem (Convergence of FW):

Let f be convex and L-smooth. Under FW with step sizes $\eta_t = 2/(t+2)$, we have

$$f(x_t) - f(x^*) \leqslant \frac{2Ld_C^2}{t+2}$$

where $d_C := \sup_{x,y \in C} \|x - y\|$ is the diameter of C

- Remark: FW achieves $O(1/t)$ rate as PGD for convex smooth problems

M. Jaggi, “Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization,” ICML 2013

Proof of Theorem

Step 1: $f(x_{t+1}) - f(x_t)$

$$\leq \underbrace{\nabla f(x_t)^T (x_{t+1} - x_t)}_{} + \underbrace{\frac{L}{2} \| x_{t+1} - x_t \|^2}_{} \quad \cdots (\quad)$$

$$\leq \eta_t \cdot \nabla f(x_t)^T (y_t - x_t) + \frac{L}{2} \eta_t^2 d_C^2 \quad \cdots (\quad)$$

$$\leq \eta_t \cdot \nabla f(x_t)^T (x^* - x_t) + \frac{L}{2} \eta_t^2 d_C^2 \quad \cdots (\quad)$$

$$\leq \eta_t \cdot (f(x^*) - f(x_t)) + \frac{L}{2} \eta_t^2 d_C^2 \quad \cdots (\quad)$$

Step 2: By letting $\Delta_t := f(x_t) - f(x^*)$, we get a recursion

$$\Delta_{t+1} \leq (1 - \eta_t) \Delta_t + \frac{L}{2} \eta_t^2 d_C^2$$

Remark on Diameter and Curvature Constant

- FW has a convergence rate that depends on $L \cdot d_C^2$
- This can be slightly improved by considering the curvature constant (Jaggi, 2013)

Definition: The curvature constant C_f is defined as

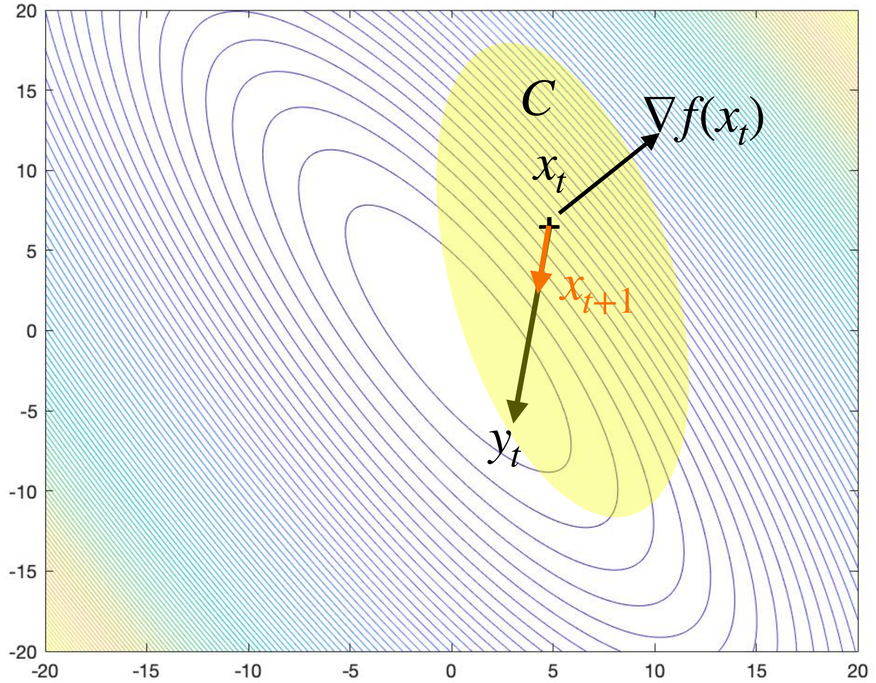
$$C_f := \sup_{x,y \in C, \gamma \in [0,1], z = x + \gamma(y-x)} \frac{2}{\gamma^2} \left(f(z) - f(x) - \nabla f(x)^\top (z - x) \right)$$

- Property: If f is L-smooth, then $C_f \leq L \cdot d_C^2$ (why?)

Remarks on the Step Size of Frank-Wolfe

- Step size $\eta_t = \frac{2}{t+2}$ does not guarantee "descent" (despite that it achieves $O(\frac{1}{t})$ convergence rate)
- To ensure "descent" in each iteration, FW can be combined with exact line search, i.e.,

$$\eta_t \in \arg\min_{\eta \in [0,1]} f(x_t + \eta(y_t - x_t))$$



How to Characterize the Stopping Criterion of Frank-Wolfe?

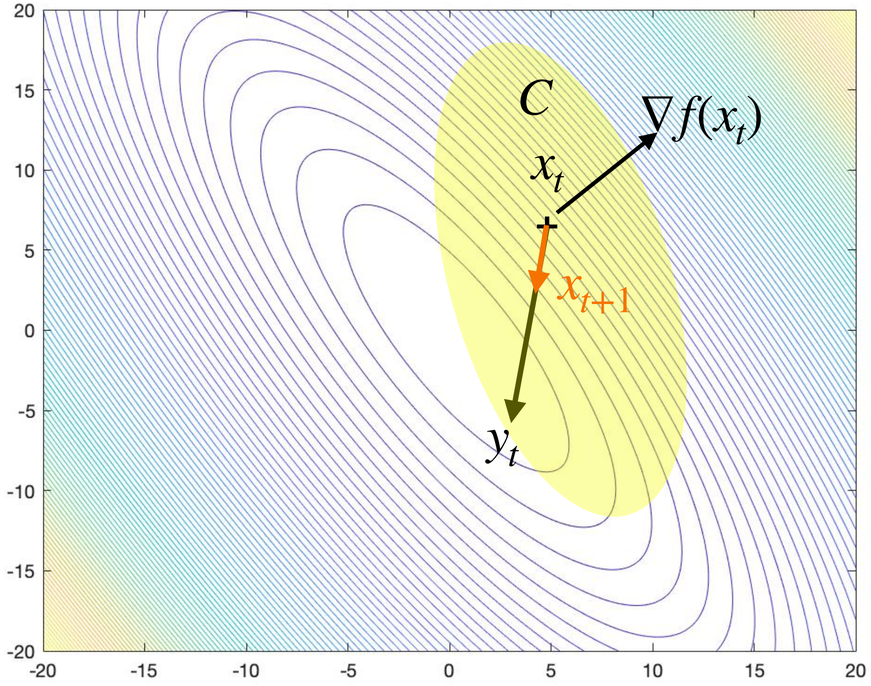
- Recall: Necessary condition for local minimizers of constrained problems

> *If x^* is a local minimizer of f over a convex feasible set C, then*
>
> $$\nabla f(x^*)^\top (x - x^*) \geq 0, \forall x \in C$$

- Idea: Convert the above into Frank-Wolfe gap

$$G_{FW}(x_t) := \max_{x \in C} \nabla f(x_t)^\top (x_t - x) = \nabla f(x_t)^\top (x_t - y_t)$$

  - $G_{FW}(x) \geq 0$, for all $x \in C$
  - $G_{FW}(x) = 0$ if and only if x is a local minimizer
  - If f is convex, then $f(x_t) - f(x^*) \leq G_{FW}(x_t)$

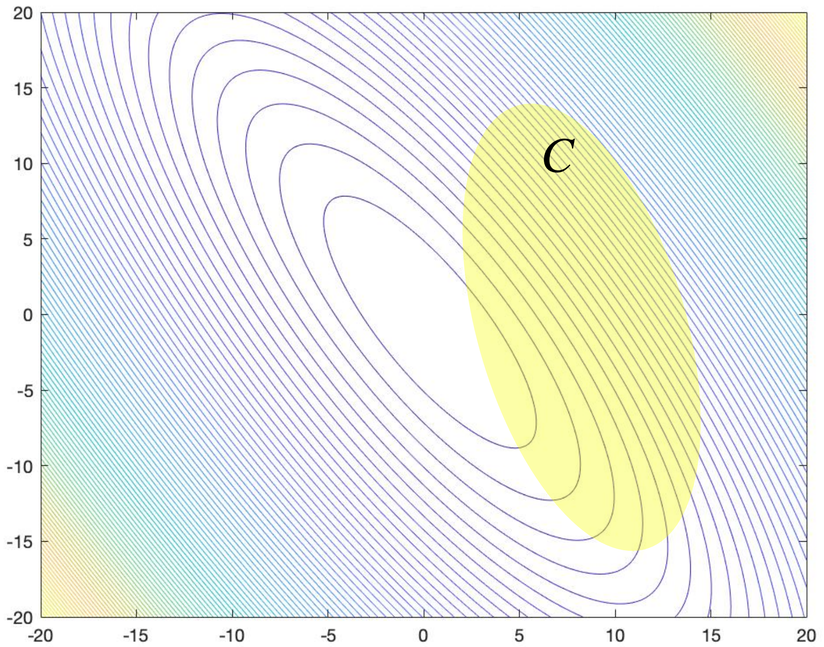


Proof of Stopping Criterion of Frank-Wolfe

$$G_{FW}(x_t) := \max_{x \in C} \nabla f(x_t)^\top (x_t - x) = \nabla f(x_t)^\top (x_t - y_t)$$

- $G_{FW}(x) \geq 0$, for all $x \in C$

- $G_{FW}(x) = 0$ if and only if x is a local minimizer



Proof of Stopping Criterion of Frank-Wolfe (Cont.)

$$G_{FW}(x_t) := \max_{x \in C} \nabla f(x_t)^\top (x_t - x) = \nabla f(x_t)^\top (x_t - y_t)$$

- If f is convex, then $f(x_t) - f(x^*) \leq G_{FW}(x_t)$

Question: Can FW attain a better convergence rate under *strong convexity* (compared to only under convexity)?

Nope in general!

A Counterexample on Impossibility of Faster Convergence Than $O(1/t)$

$$\min_{x} \quad \frac{1}{2} x^\top Q x + b^\top x$$

subject to $x \in \textbf{Conv}\{a_1, \cdots, a_k\} =: C$

- Suppose x^* is on the boundary of C
- Suppose the interior of C is non-empty
- Q is positive definite

- A Classic Result by (Canon & Cullum, 1968): Under FW, there exists an initial point x_0 in the interior of C such that for any $\epsilon > 0$, we have

$$f(x_t) - f(x^*) \geq \frac{1}{t^{1+\epsilon}}, \quad \text{infinitely often}$$

M. Canon and C. Cullum, “A tight upper bound on the rate of convergence of Frank-Wolfe algorithm,” SIAM Journal on Control, 1968.

Another Counterexample on Impossibility of Faster Convergence Than $O(1/t)$

$$\begin{aligned} \min_{x} \quad & \|x\|^2 \\ \textbf{subject to} \quad & x^{(i)} \geq 0, \sum_{i=1}^{d} x^{(i)} = 1 \end{aligned}$$

- A Well-Known Result by (Jaggi, 2013): Under FW and the above quadratic problem with simplex constraint, $f(x) - f(x^*) \leq \epsilon$ requires $\Omega(\min\{n, 1/\epsilon\})$ optimization steps

M. Jaggi. Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization, ICML 2013

Summary: Comparison of PGD and FW

- Consider convex and smooth problems

	Step Size	Convergence Rate	Iteration Complexity
PGD	$\eta = \frac{1}{L}$	$f(x_t) - f(x^*) \leqslant \frac{3L\|x_0 - x^*\|^2 + \left(f(x_0) - f(x^*)\right)}{t+1} = O(\frac{1}{t})$	$O(\frac{1}{\epsilon})$
FW	$\eta_t = \frac{2}{t+2}$	$f(x_t) - f(x^*) \leqslant \frac{2Ld_C^2}{t+2} = O(\frac{1}{t})$	$O(\frac{1}{\epsilon})$

How About Frank-Wolfe for Non-Convex Functions?

Convergence of Frank-Wolfe for Non-Convex Functions

Theorem Let $f(x)$ and C satisfy the following conditions:

(1) f is a continuously differentiable function (possibly non-convex).

(2) C is convex and compact.

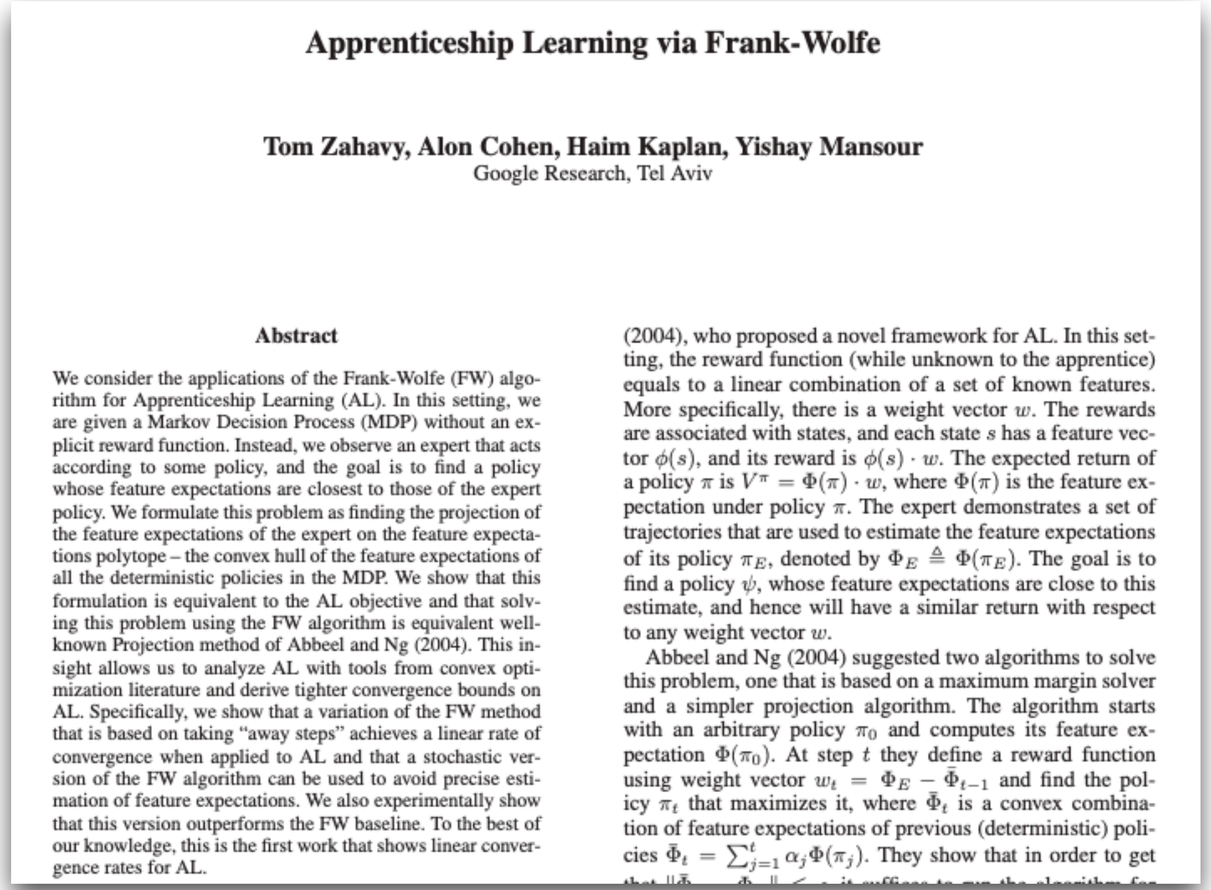
Then, we have $$\min_{0 \le t \le T} G_{FW}(x_t) \le \frac{\max\{2(f(x_0) - f(x^*)), C_f\}}{\sqrt{T+1}}$$

Simon Lacoste-Julien, Convergence Rate of Frank-Wolfe for Non-Convex Objectives, arXiv 2016

Some Applications of Frank-Wolfe Method

- L_p-norm regularization
- Trust-region optimization
- Frank-Wolfe for inverse RL
- Frank-Wolfe for constrained RL

[Zahavy et al., AAAI 2020]



Apprenticeship Learning via Frank-Wolfe

Tom Zahavy, Alon Cohen, Haim Kaplan, Yishay Mansour
Google Research, Tel Aviv

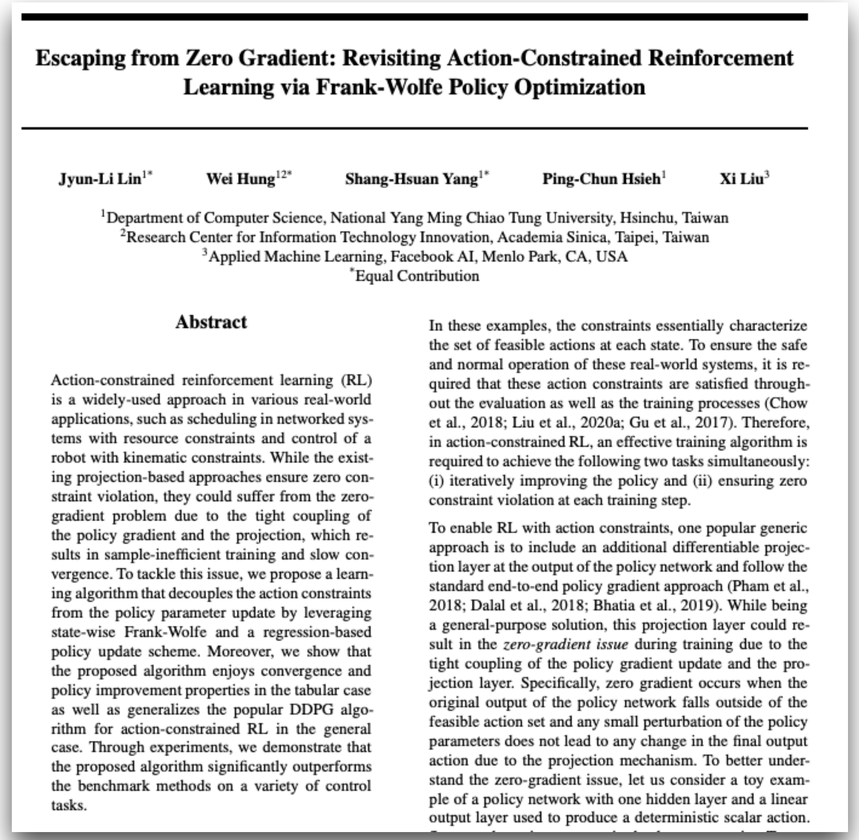
Abstract

We consider the applications of the Frank-Wolfe (FW) algorithm for Apprenticeship Learning (AL). In this setting, we are given a Markov Decision Process (MDP) without an explicit reward function. Instead, we observe an expert that acts according to some policy, and the goal is to find a policy whose feature expectations are closest to those of the expert policy. We formulate this problem as finding the projection of the feature expectations of the expert on the feature expectations polytope – the convex hull of the feature expectations of all the deterministic policies in the MDP. We show that this formulation is equivalent to the AL objective and that solving this problem using the FW algorithm is equivalent well-known Projection method of Abbeel and Ng (2004). This insight allows us to analyze AL with tools from convex optimization literature and derive tighter convergence bounds on AL. Specifically, we show that a variation of the FW method that is based on taking "away steps" achieves a linear rate of convergence when applied to AL and that a stochastic version of the FW algorithm can be used to avoid precise estimation of feature expectations. We also experimentally show that this version outperforms the FW baseline. To the best of our knowledge, this is the first work that shows linear convergence rates for AL.

(2004), who proposed a novel framework for AL. In this setting, the reward function (while unknown to the apprentice) equals to a linear combination of a set of known features. More specifically, there is a weight vector w. The rewards are associated with states, and each state s has a feature vector $\phi(s)$, and its reward is $\phi(s) \cdot w$. The expected return of a policy π is $V^\pi = \Phi(\pi) \cdot w$, where $\Phi(\pi)$ is the feature expectation under policy π. The expert demonstrates a set of trajectories that are used to estimate the feature expectations of its policy π_E, denoted by $\Phi_E \triangleq \Phi(\pi_E)$. The goal is to find a policy ψ, whose feature expectations are close to this estimate, and hence will have a similar return with respect to any weight vector w.

Abbeel and Ng (2004) suggested two algorithms to solve this problem, one that is based on a maximum margin solver and a simpler projection algorithm. The algorithm starts with an arbitrary policy π_0 and computes its feature expectation $\Phi(\pi_0)$. At step t they define a reward function using weight vector $w_t = \Phi_E - \bar{\Phi}_{t-1}$ and find the policy π_t that maximizes it, where $\bar{\Phi}_t$ is a convex combination of feature expectations of previous (deterministic) policies $\bar{\Phi}_t = \sum_{j=1}^{t} \alpha_j \Phi(\pi_j)$. They show that in order to get

[Lin et al., UAI 2021]



Escaping from Zero Gradient: Revisiting Action-Constrained Reinforcement Learning via Frank-Wolfe Policy Optimization

Jyun-Li Lin[1*] **Wei Hung**[12*] **Shang-Hsuan Yang**[1*] **Ping-Chun Hsieh**[1] **Xi Liu**[3]

[1]Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan
[2]Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan
[3]Applied Machine Learning, Facebook AI, Menlo Park, CA, USA
[*]Equal Contribution

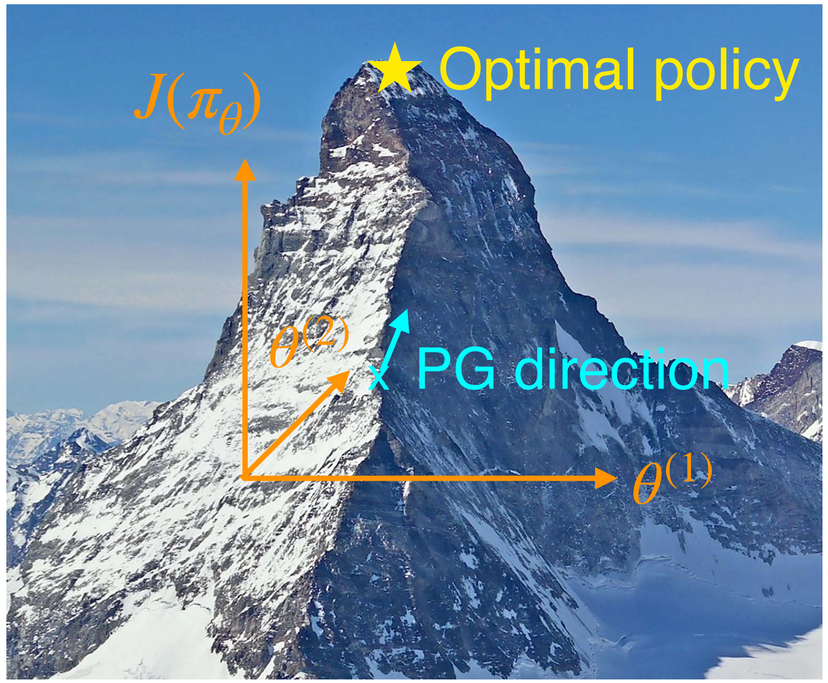
Abstract

Action-constrained reinforcement learning (RL) is a widely-used approach in various real-world applications, such as scheduling in networked systems with resource constraints and control of a robot with kinematic constraints. While the existing projection-based approaches ensure zero constraint violation, they could suffer from the zero-gradient problem due to the tight coupling of the policy gradient and the projection, which results in sample-inefficient training and slow convergence. To tackle this issue, we propose a learning algorithm that decouples the action constraints from the policy parameter update by leveraging state-wise Frank-Wolfe and a regression-based policy update scheme. Moreover, we show that the proposed algorithm enjoys convergence and policy improvement properties in the tabular case as well as generalizes the popular DDPG algorithm for action-constrained RL in the general case. Through experiments, we demonstrate that the proposed algorithm significantly outperforms the benchmark methods on a variety of control tasks.

In these examples, the constraints essentially characterize the set of feasible actions at each state. To ensure the safe and normal operation of these real-world systems, it is required that these action constraints are satisfied throughout the evaluation as well as the training processes (Chow et al., 2018; Liu et al., 2020a; Gu et al., 2017). Therefore, in action-constrained RL, an effective training algorithm is required to achieve the following two tasks simultaneously: (i) iteratively improving the policy and (ii) ensuring zero constraint violation at each training step.

To enable RL with action constraints, one popular generic approach is to include an additional differentiable projection layer at the output of the policy network and follow the standard end-to-end policy gradient approach (Pham et al., 2018; Dalal et al., 2018; Bhatia et al., 2019). While being a general-purpose solution, this projection layer could result in the *zero-gradient issue* during training due to the tight coupling of the policy gradient update and the projection layer. Specifically, zero gradient occurs when the original output of the policy network falls outside of the feasible action set and any small perturbation of the policy parameters does not lead to any change in the final output action due to the projection mechanism. To better understand the zero-gradient issue, let us consider a toy example of a policy network with one hidden layer and a linear output layer used to produce a deterministic scalar action.

A Primer on Policy Gradient



- RL can be cast as an optimization problem

$$\max_\theta J(\pi_\theta) := \mathbb{E}_{s_0 \sim \mu, a_0 \sim \pi_\theta(\cdot|s_0)}[Q^\pi(s_0, a_0)]$$

where $Q^\pi(s,a) := \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \Big| s_0 = s, a_0 = a; \pi\Big]$

(Q-function reflects "how good a policy is")

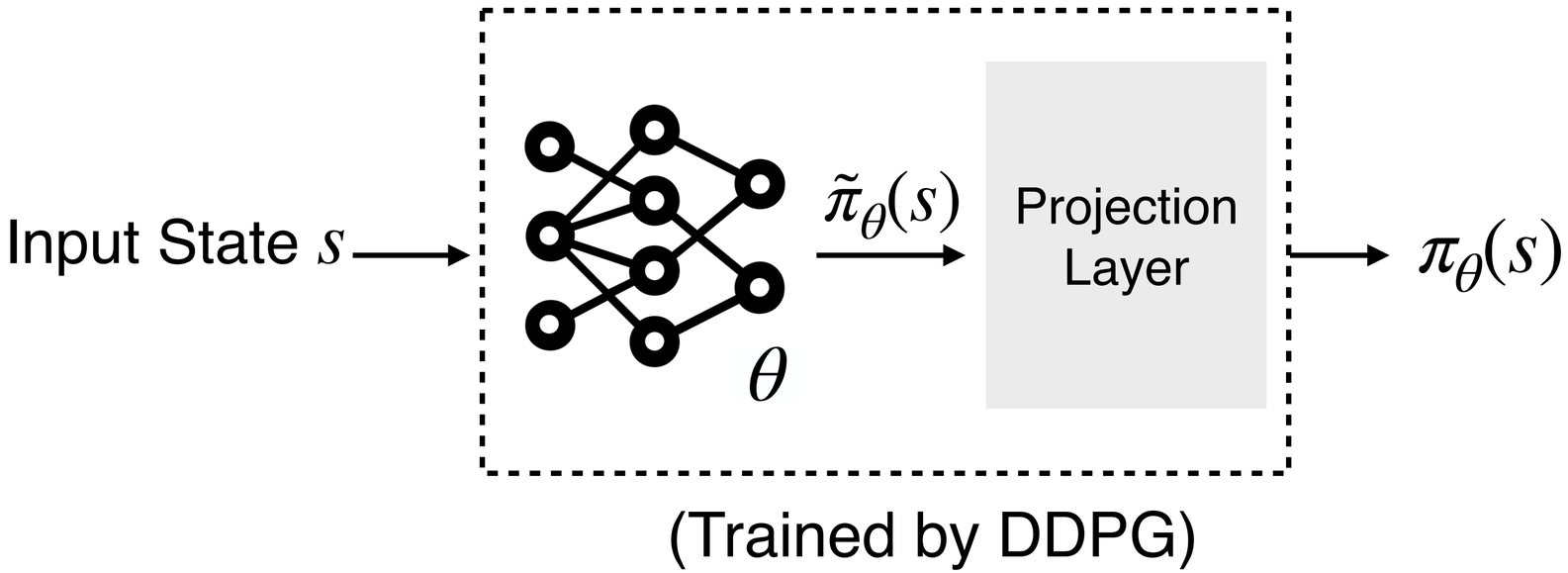
- PG takes "steepest ascent" direction [Silver, 2014]

$$\theta \leftarrow \theta + \alpha \cdot \nabla_\theta J(\pi_\theta)$$

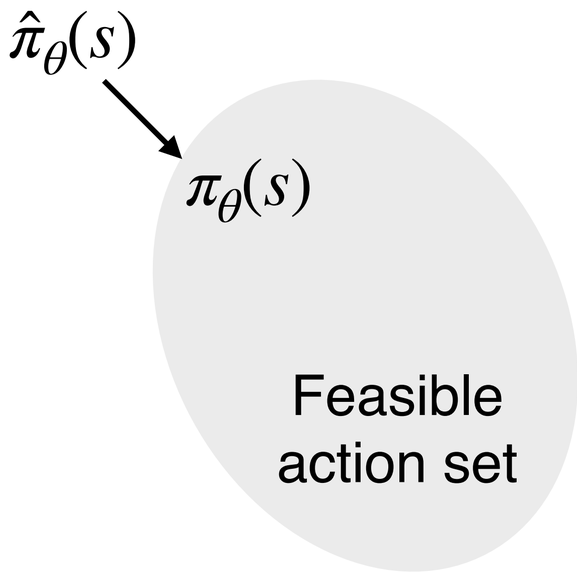
David Silver, "Deterministic Policy Gradient Algorithms," ICML 2014

Existing Solution to Action-Constrained RL: Projection Layer

DDPG-OptLayer [Pham et al., 2018]



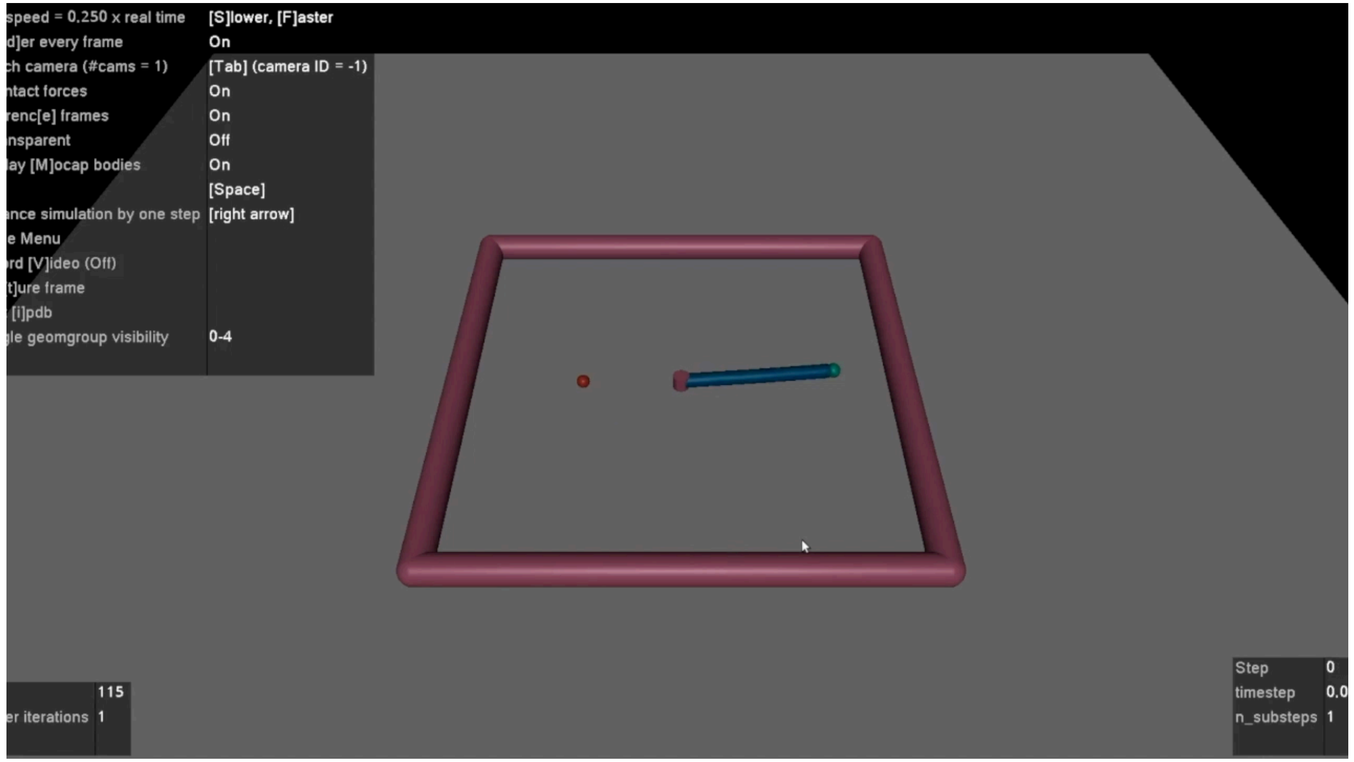
What projection layer does is:



However, this projection layer is problematic

T. Pham, et al., “Optlayer - practical constrained optimization for deep reinforcement learning in the real world,” ICRA 2018

Why is Projection Layer Problematic?
A Motivating Experiment



Zero-gradient issue!

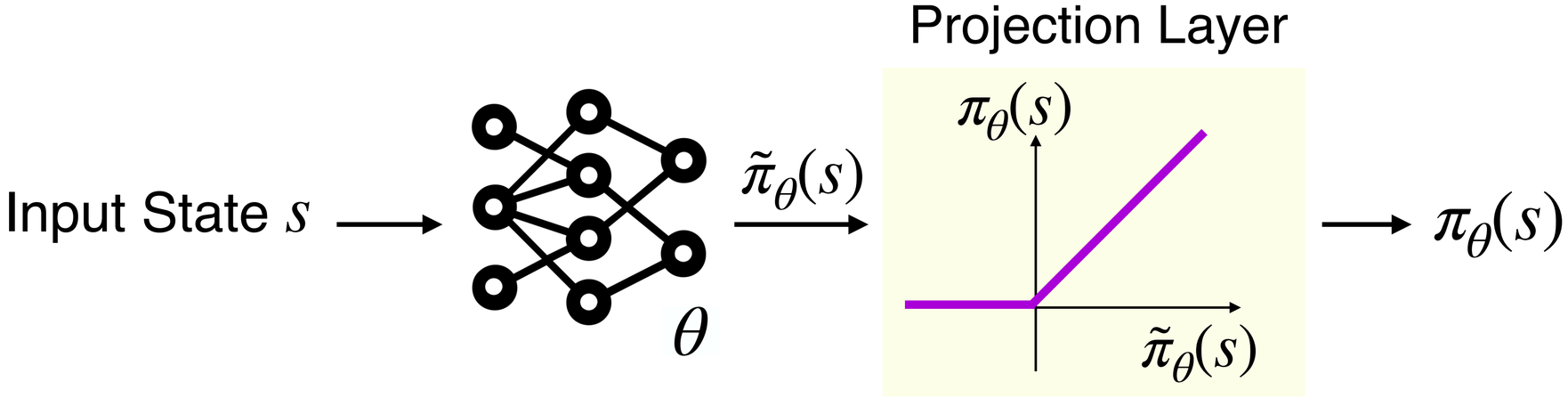
Reacher in MuJoCo

- Two joints with torques $u_1, u_2 \in [-1, 1]$
- Constraints: $|u_1 + u_2| \leq 0.1$, $u_1^2 + u_2^2 \leq 0.02$
- Algorithm: DDPG-OptLayer
- Model trained for 500k steps

What is Zero Gradient in ACRL? A Motivating Example

Suppose output actions $\pi_\theta(s)$ must be nonnegative (i.e., $\pi_\theta(s) \geq 0$)

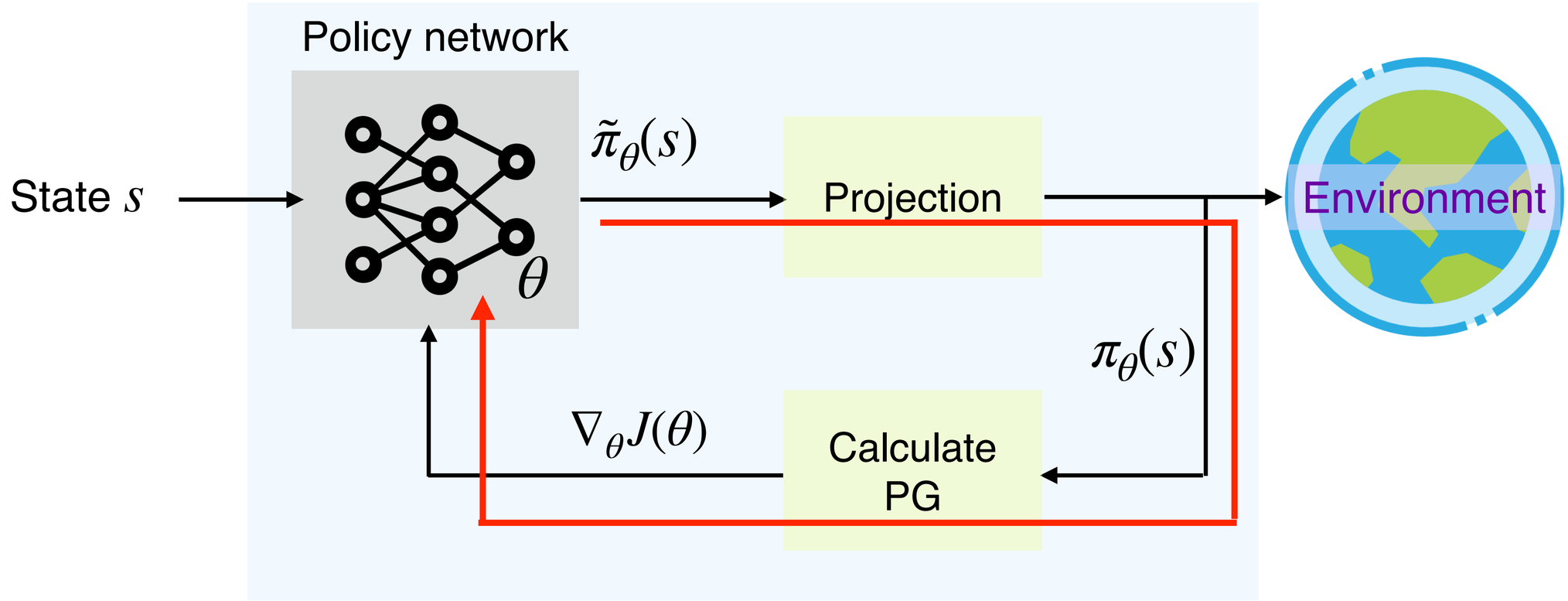
(In this case: Projection = Rectification)



When $\hat{\pi}_\theta(s) < 0$, any small perturbation on θ has no effect!

Zero gradient! (No learning progress at all)

Why DDPG+PGD Fails: Tight Coupling of PG & Projection

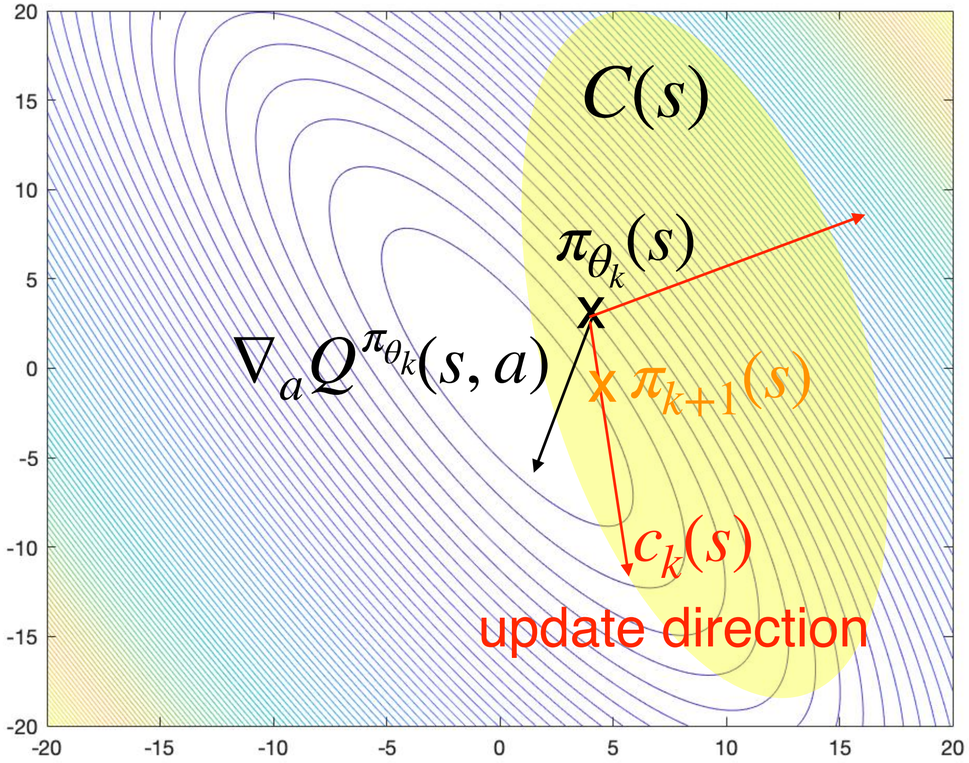


PG and Projection are in the same training loop!

We shall decouple these two components!

Frank-Wolfe Policy Optimization (FWPO)

- FWPO: Search "in the feasible set" for better actions (for each state)
  - Use state-wise Frank-Wolfe for policy improvement



Contour of $Q^{\pi_k}(s,a)$ w.r.t. action a

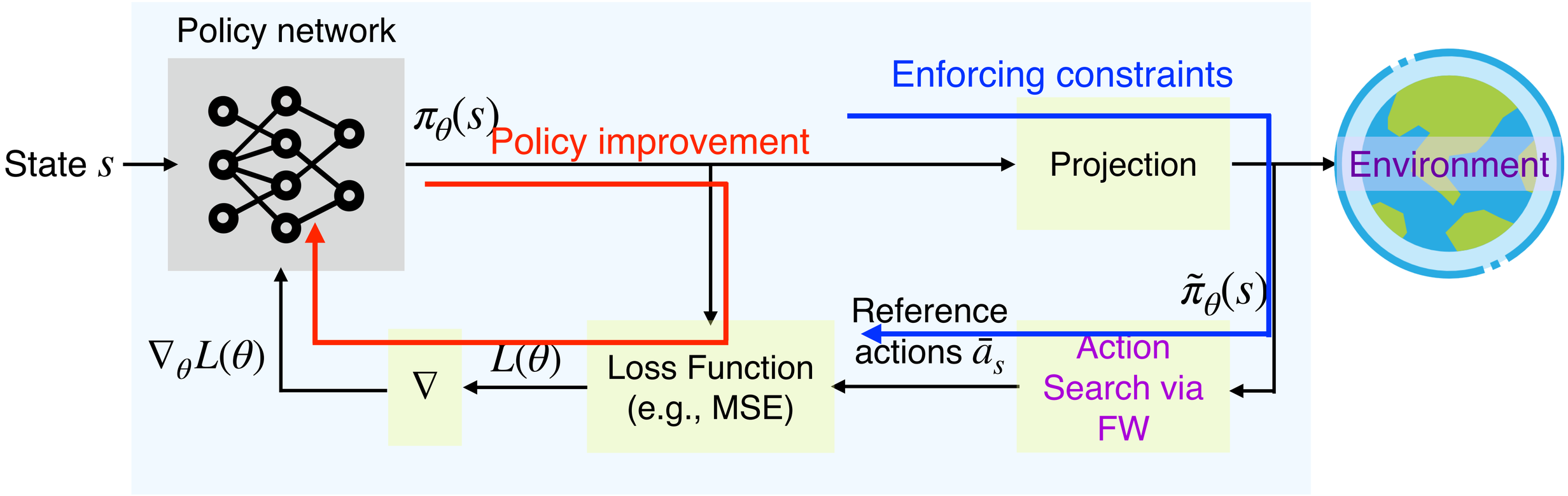
$$c_k(s) = \arg\max_{c \in C(s)} < c - \theta_k(s), \nabla_a Q(s,a;\pi_{\theta_k}) >$$

$$\theta_{k+1}(s) = \theta_k(s) + \alpha_k(s)(c_k(s) - \theta_k(s))$$

Features of FWPO:

1. No constraint violation (always stays in feasible set)
2. No projection layer needed in FWPO
3. No policy gradient in FWPO
4. Provably convergent!

Neural FWPO Framework

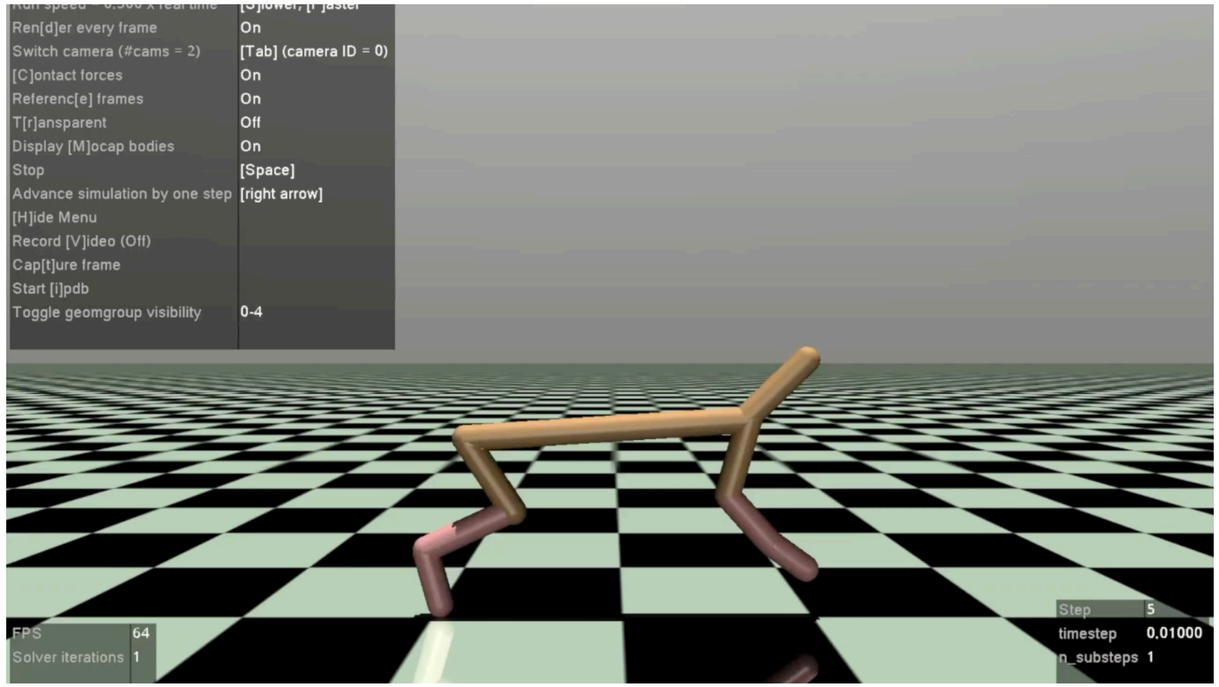


FWPO decouples policy improvement and constraint satisfaction

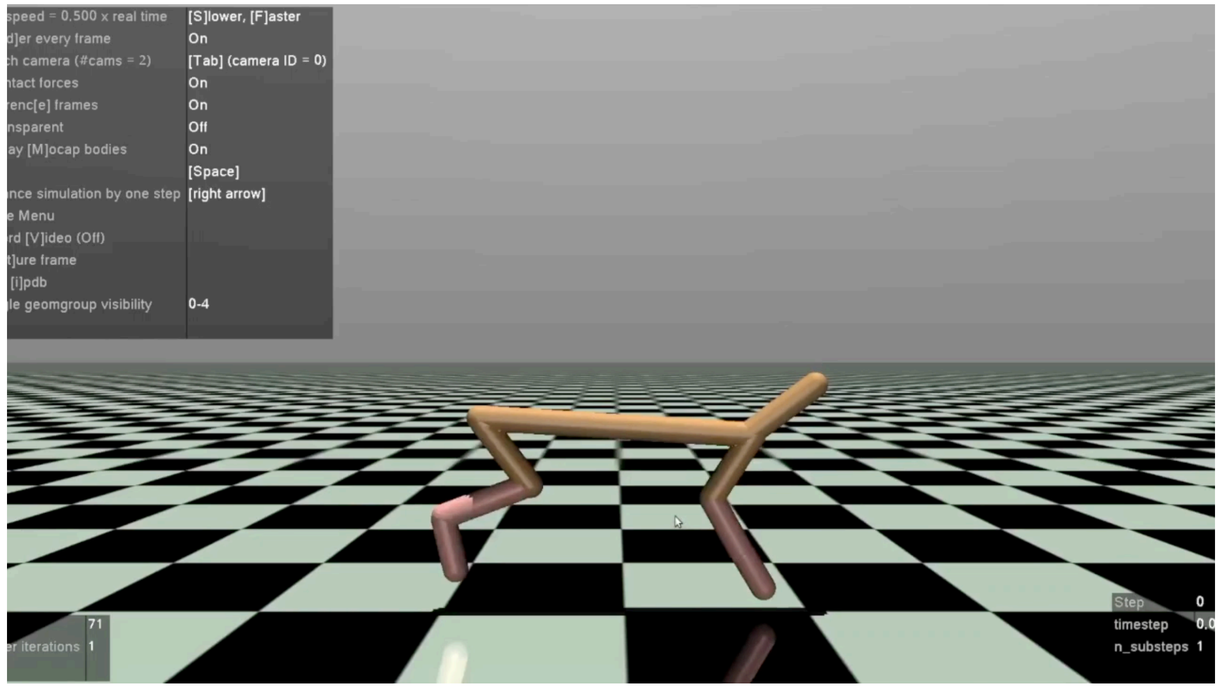
Hence, it completely avoids zero gradient!

Demo: Halfcheetah in MuJoCo

Neural FWPO



DDPG With Projection Layer



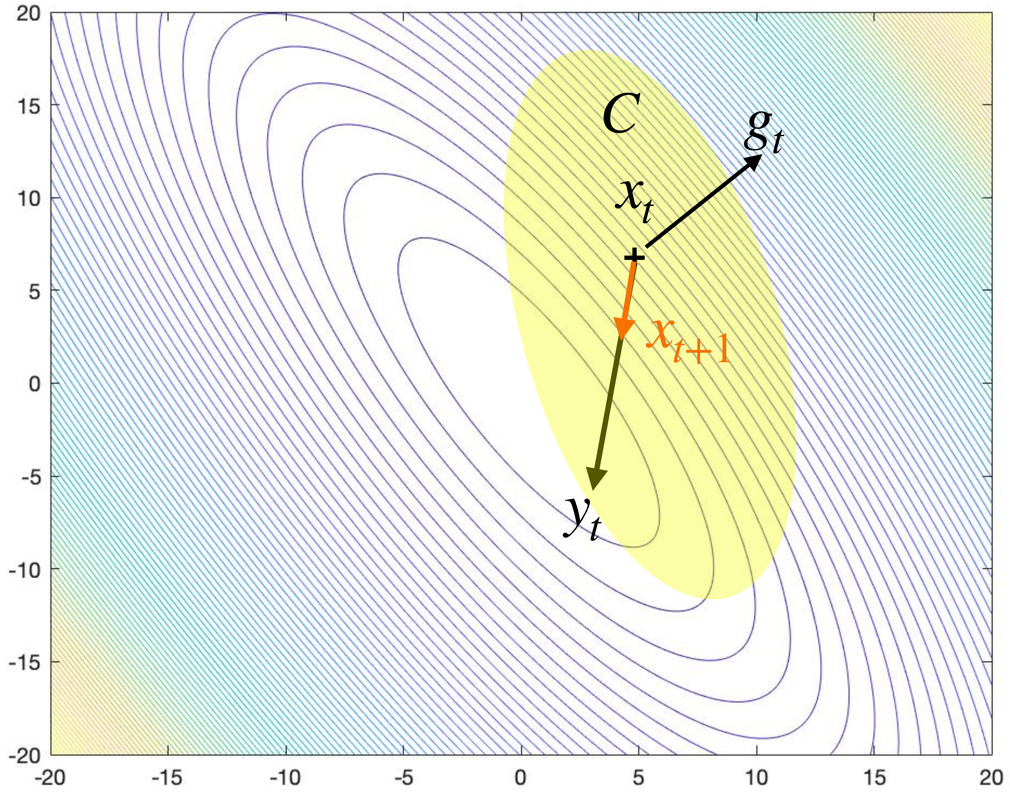
Stochastic Frank-Wolfe

Elad Hazan and Haipeng Luo, “Variance-Reduced and Projection-Free Stochastic Optimization,” ICML 2016

Extending FW to Stochastic FW (SFW)

- Suppose we focus on empirical risk minimization

$$\min_{x\in X} F(x) := \frac{1}{n}\sum_{i=1}^{n} f(x; d_i) \equiv f_i(x) \qquad (\{d_i\}_{i=1}^{n} \text{ are data samples})$$



- Under SFW, the iterates are updated as

$$y_t = \arg\min_{x\in C} g_t^\top x$$

$$x_{t+1} = (1-\eta_t)x_t + \eta_t y_t$$

where $g_t = \frac{1}{|B_t|}\sum_{i\in B_t} \nabla f(x_t; d_i)$ is an unbiased estimate of $\nabla F(x_t)$ based on $|B_t| \equiv m_t$ i.i.d. samples from the dataset

Convergence Rate of Stochastic FW

Theorem Suppose the following conditions hold:

(1) $F(x)$ is L-smooth and convex

(2) Each $f_i(x)$ is G-Lipschitz

(3) Step size $\eta_t = \frac{2}{t+1}$ and $m_t = \left(\frac{G(t+1)}{L \cdot d_C} \right)^2$

Then, we have $E[F(x_t) - F(x^*)] \leq \frac{4 L d_C^2}{t+2}$

Question: Any difference in convergence between SGD and SFW?

Proof of Convergence

Step 1: One-step improvement

$$F(x_{t+1}) \le F(x_t) + \nabla F(x_t)^T (x_{t+1} - x_t) + \frac{L}{2}\|x_{t+1} - x_t\|^2 \cdots (\quad)$$

$$= F(x_{t-1}) + \eta_t \cdot \nabla F(x_{t-1})^T (y_t - x_t) + \frac{L}{2}\eta_t^2 \cdot \|y_t - x_t\|^2 \cdots (\quad)$$

$$\le F(x_{t-1}) + \eta_t \cdot g_t^T (y_t - x_t) + \eta_t \cdot (\nabla F(x_t) - g_t)^T \cdot (y_t - x_t) + \frac{L d_c^2 \eta_t^2}{2} \cdots (\quad)$$

$$\le F(x_{t-1}) + \eta_t \, g_t^T (x^* - x_t) + \eta_t \cdot (\nabla F(x_t) - g_t)^T \cdot (y_t - x_t) + \frac{L d_c^2 \eta_t^2}{2} \cdots (\quad)$$

$$= F(x_{t-1}) + \eta_t \nabla F(x_t)^T (x^* - x_t) + \eta_t \cdot (\nabla F(x_t) - g_t) \cdot (y_t - x^*) + \frac{L \cdot d_c^2 \eta_t^2}{2}$$

$$\le F(x_{t-1}) + \eta_t \left(F(x^*) - F(x_t) \right) + \eta_t \cdot \|\nabla F(x_t) - g_t\| \cdot \|y_t - x^*\| + \frac{L \cdot d_c^2 \cdot \eta_t^2}{2} \cdots (\quad)$$

Step2: By taking the expectation on both sides,

$$E[F(x_{t+1})] \le E[F(x_t)] + \eta_t \cdot E[F(x^*) - F(x_t)]$$

$$+ \eta_t \cdot d_c \cdot \underbrace{E[\| \nabla F(x_t) - g_t \|]}_{\le \sqrt{E[\| \nabla F(x_t) - g_t \|^2]} \le \frac{G}{\sqrt{m_t}} \le \frac{L \cdot d_c \cdot \eta_t}{2}} + \frac{L d_c^2 \eta_t^2}{2}$$

(why?) (why?)

Then, we have $E[F(x_{t+1}) - F(x^*)] \le (1-\eta_t) \cdot E[F(x_t) - F(x^*)] + L \cdot d_c^2 \cdot \eta_t^2$

(Finally, the proof can be completed by an inductive argument)