Matrix-Free Preconditioning in Online Learning

Ashok Cutkosky, Tamas Sarlos
Google Research

Online Optimization

For $t = 1 \ldots T$, repeat:

1: Learner chooses a point w_t.

2: Environment presents learner with a gradient g_t (think $\mathbb{E}[g_t] = \nabla F(w_t)$).

3: Learner suffers loss $\langle g_t, w_t \rangle$.

The objective is minimize *regret*:

$$R_T(w_\star) = \sum_{t=1}^{T} \underbrace{\langle g_t, w_t \rangle}_{\text{loss suffered}} - \underbrace{\langle g_t, w_\star \rangle}_{\text{benchmark loss}}$$

Online Optimization

For $t = 1 \dots T$, repeat:

1: Learner chooses a point w_t.

2: Environment presents learner with a gradient g_t (think $\mathbb{E}[g_t] = \nabla F(w_t)$).

3: Learner suffers loss $\langle g_t, w_t \rangle$.

The objective is minimize *regret*:

$$R_T(w_\star) = \sum_{t=1}^{T} \underbrace{\langle g_t, w_t \rangle}_{\text{loss suffered}} - \underbrace{\langle g_t, w_\star \rangle}_{\text{benchmark loss}}$$

Running an online algorithm on a stochastic optimization problem guarantees $F(\overline{w}_T) - F(w_\star) \le \frac{R_T(w_\star)}{T}$.

The Classic Algorithm: Gradient Descent

$$w_{t+1} = w_t - \eta_t g_t$$

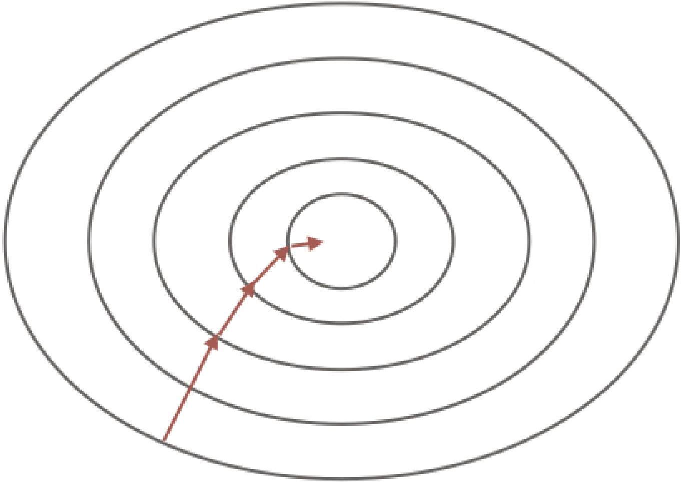
The Classic Algorithm: Gradient Descent

$$w_{t+1} = w_t - \eta_t g_t$$

Gradient descent obtains regret:

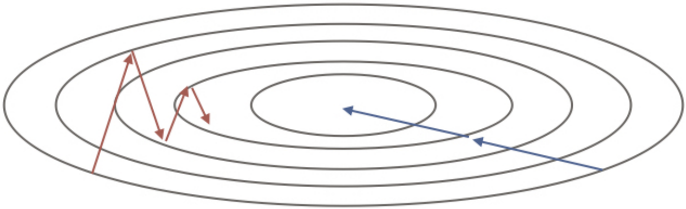
$$R_T(w_\star) \leq \sqrt{\sum_{t=1}^{T} \|w_\star\|^2 \|g_t\|^2}$$

Gradient Descent



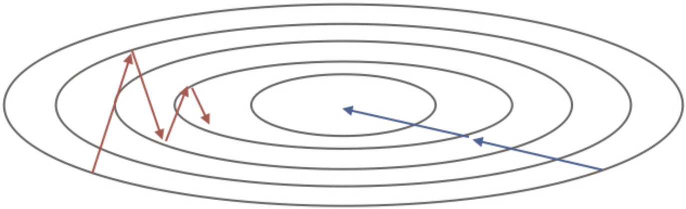
Preconditioning (Deterministic)

- The gradient $\nabla F(w)$ may not point towards the minimum $w_\star$



Preconditioning (Deterministic)

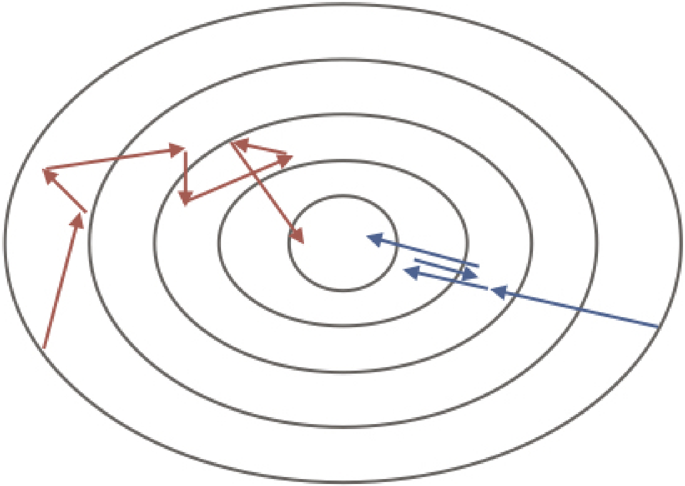
- The gradient $\nabla F(w)$ may not point towards the minimum $w_\star$



Key idea: "Preconditioning" means ignoring irrelevant directions.

Preconditioning (Stochastic)

- Noise can also make g_t not point towards the minimum.



Regret Bounds

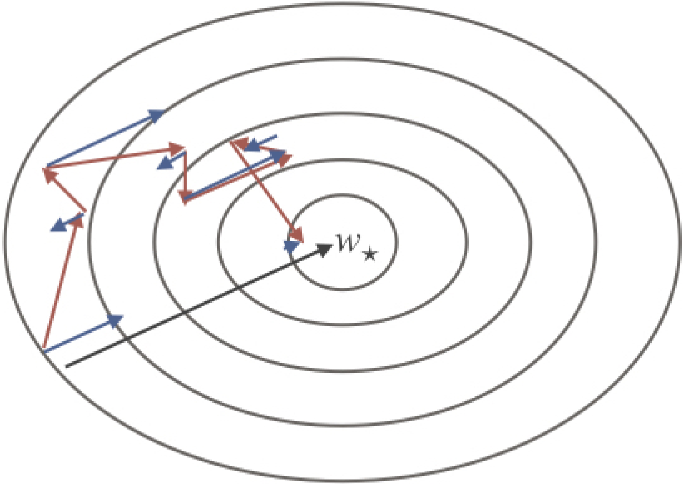
- Regret of un-preconditioned stochastic gradient descent (with the appropriate learning rate) is

$$R_T(w_\star) \leq \sqrt{\sum_{t=1}^{T} \|w_\star\|^2 \|g_t\|^2} = O\left(\sqrt{T}\right)$$

- An ideal preconditioned algorithm should obtain regret

$$R_T(w_\star) \leq \sqrt{\sum_{t=1}^{T} \langle w_\star, g_t \rangle^2} = O\left(\sqrt{T}\right)$$

Regret Bound Picture



Goals

- Want regret bound as good as if we had ignored irrelevant directions (up to constants/logs)

Using the Covariance Matrix

The typical approach to preconditioning maintains the matrix

$$G = \sum_{t=1}^{T} g_t g_t^\top$$

and compute various inverses and square roots of G. This can obtain the guarantee [CO18; KL17]

$$R_T(w_\star) \le \sqrt{d \sum_{t=1}^{T} \langle w_\star, g_t \rangle^2}$$

Issues with Using Covariance Matrix

- d^2 time is too slow - there's a lot of work on compressing the matrix to try to make some tradeoff [Luo+16; GKS18; Aga+18].

Issues with Using Covariance Matrix

- d^2 time is too slow - there's a lot of work on compressing the matrix to try to make some tradeoff [Luo+16; GKS18; Aga+18].
- The regret bound might not even be better!

$$\sqrt{d \sum_{t=1}^{T} \langle w_\star, g_t \rangle^2} \overset{??}{\leq} \sqrt{\|w_\star\|^2 \sum_{t=1}^{T} \|g_t\|^2}$$

Goals

1: Want regret bound as good as if we had ignored irrelevant directions (up to constants/logs).
2: Want an efficient algorithm ($O(d)$ time per update in d-dimensions).
3: Want to never do worse than non-preconditioned algorithms.

Goals

1: Want regret bound as good as if we had ignored irrelevant directions (up to constants/logs).
2: Want an efficient algorithm ($O(d)$ time per update in d-dimensions).
3: Want to never do worse than non-preconditioned algorithms.

- We will achieve 2 and 3, and sometimes 1.

Our Contribution

We provide an online learning algorithm that:

- Runs in $O(d)$ time per-update.
- Always achieves regret:

$$R_T(w_\star) \le \|w_\star\| \sqrt{\sum_{t=1}^{T} \|g_t\|^2}$$

- When $-\langle \sum_{t=1}^{T} g_t, w_\star / \|w_\star\| \rangle \ge \sqrt{\sum_{t=1}^{T} \|g_t\|^2}$, achieves:

$$R_T(w_\star) \le \sqrt{\sum_{t=1}^{T} \langle w_\star, g_t \rangle^2}$$

Unpacking the Condition

- We need $-\langle\sum_{t=1}^T g_t, w_\star/\|w_\star\|\rangle \geq \sqrt{\sum_{t=1}^T \|g_t\|^2}$ for preconditioned regret.
- If g_t are mean-zero independent random variables, then standard concentration results say:

$$-\left\langle \sum_{t=1}^T g_t, w_\star/\|w_\star\| \right\rangle \leq \left\| \sum_{t=1}^T g_t \right\| = \Theta\left(\sqrt{\sum_{t=1}^T \|g_t\|^2}\right)$$

Unpacking the Condition

- We need $-\langle\sum_{t=1}^T g_t, w_\star/\|w_\star\|\rangle \geq \sqrt{\sum_{t=1}^T \|g_t\|^2}$ for preconditioned regret.
- If g_t are mean-zero independent random variables, then standard concentration results say:

$$-\left\langle \sum_{t=1}^T g_t, w_\star/\|w_\star\| \right\rangle \leq \left\| \sum_{t=1}^T g_t \right\| = \Theta\left(\sqrt{\sum_{t=1}^T \|g_t\|^2} \right)$$

We achieve preconditioning whenever there is any “signal” in the gradients.

Coin Betting [OP16]

- Define *wealth*:

$$\text{Wealth}_T = 1 - \sum_{t=1}^{T} \langle g_t, w_t \rangle$$

Coin Betting [OP16]

- Define *wealth*:

$$\text{Wealth}_T = 1 - \sum_{t=1}^{T} \langle g_t, w_t \rangle$$

- High wealth implies low regret:

$$R_T(w_\star) = \underbrace{1 - \sum_{t=1}^{T} \langle g_t, w_\star \rangle}_{\text{out of our control}} - \text{Wealth}_T$$

Coin Betting [OP16]

- Define *wealth*:

$$\text{Wealth}_T = 1 - \sum_{t=1}^{T} \langle g_t, w_t \rangle$$

- High wealth implies low regret:

$$R_T(w_\star) = \underbrace{1 - \sum_{t=1}^{T} \langle g_t, w_\star \rangle}_{\text{out of our control}} - \text{Wealth}_T$$

- At every iteration, choose a *betting fraction* $v_t \in \mathbb{R}^d$ and use

$$w_t = v_t \text{Wealth}_{t-1}$$

Oracle value for v yields good algorithm

Set $v_t = v_\star \approx \frac{w_\star}{\|w_\star\|\sqrt{\sum_{t=1}^T \langle g_t, w_\star\rangle^2}}$. Then

$$R_T(w_\star) \leq \sqrt{\sum_{t=1}^T \langle w_\star, g_t\rangle^2}$$

- There are no matrices here!
- But we don't know this magic value for v.

Online Learning Inside Online Learning [CO18]

- Define $\ell_t(v) = -\log(1 - \langle g_t, v \rangle)$. Then:

$$R_T^v(v_\star) := \sum_{t=1}^{T} \ell_t(v_t) - \ell_t(v_\star)$$

- If $R_T^v(v_\star) = O(\log(T))$, then the final regret $R_T(w_\star)$ is the same as if we'd used the constant $v_t = v_\star$.

Online Learning Inside Online Learning [CO18]

- Define $\ell_t(v) = -\log(1 - \langle g_t, v\rangle)$. Then:

$$R_T^v(v_\star) := \sum_{t=1}^{T} \ell_t(v_t) - \ell_t(v_\star)$$

- If $R_T^v(v_\star) = O(\log(T))$, then the final regret $R_T(w_\star)$ is the same as if we'd used the constant $v_t = v_\star$.
- We can use online learning to choose the v_t!

Overview of Algorithm Strategy

- There exists an unknown $v_\star$ that would give preconditioned regret.
- We can choose v_t using online convex optimization on losses $\ell_t(v) = -\log(1 - \langle g_t, v_t \rangle)$.
- If we get $R_T^v(v_\star) = \sum_{t=1}^T \ell_t(v_t) - \ell_t(v_\star) = O(\log(T))$, then we are as good as picking $v_\star$ from the beginning.
- So how can we obtain logarithmic regret?

How to obtain logarithmic regret?

- Strategy: Remember that the constant $v_\star$ we need to compete with is $v_\star = \frac{w_\star}{\|w_\star\|\sqrt{\sum_{t=1}^{T}\langle g_t, w_\star\rangle^2}}$, so $\|v_\star\| = O(1/\sqrt{T})$ usually.
- This means that we can use a non-preconditioned online learning algorithm to obtain logarithmic regret:

$$R_T^v(v_\star) \leq \|v_\star\|\sqrt{T} = O(1)$$

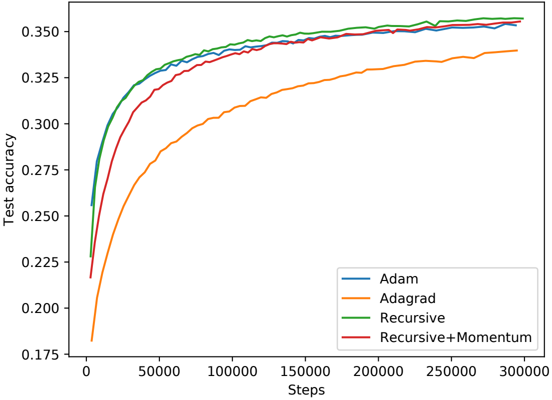
How to obtain logarithmic regret?

- Strategy: Remember that the constant $v_\star$ we need to compete with is $v_\star = \frac{w_\star}{\|w_\star\| \sqrt{\sum_{t=1}^{T} \langle g_t, w_\star \rangle^2}}$, so $\|v_\star\| = O(1/\sqrt{T})$ usually.
- This means that we can use a non-preconditioned online learning algorithm to obtain logarithmic regret:

$$R_T^v(v_\star) \leq \|v_\star\| \sqrt{T} = O(1)$$

- Sometimes the best v is not small - this is why we do not always obtain preconditioned regret.

Experiments



Test accuracy on LM1B dataset with Transformer model

Summary

- When the gradients are "obviously non-random", we obtain preconditioned regret bounds without any bad $\sqrt{d}$ constant factors.
- Otherwise, we decay to the ordinary non-preconditioned regret bounds (actually, we improve log factors).
- The algorithm runs in the same time complexity as ordinary gradient descent.
- The empirical performance is promising.

Summary

- When the gradients are "obviously non-random", we obtain preconditioned regret bounds without any bad $\sqrt{d}$ constant factors.
- Otherwise, we decay to the ordinary non-preconditioned regret bounds (actually, we improve log factors).
- The algorithm runs in the same time complexity as ordinary gradient descent.
- The empirical performance is promising.

Thank you!