Bandit PCA

Gergely Neu
Univ. Pompeu Fabra (Barcelona, Spain)

joint work with Wojciech Kotłowski
Appetizer  |  PCA, bandit PCA, phase retrieval
Principal component analysis (PCA)
Principal component analysis (PCA)

principal component
≈
“direction with minimal total projection loss”
Bandit PCA

Principal Component Analysis with
• sequentially chosen projections (online PCA)
• partial observability (bandit PCA)
Bandit PCA

Principal Component Analysis with
• sequentially chosen projections (online PCA)
• partial observability (bandit PCA)

\[ t = 1 \]
Bandit PCA

Principal Component Analysis with
• sequentially chosen projections (online PCA)
• partial observability (bandit PCA)

\[ t = 1 \text{ environment chooses hidden vector } x_t \]
Bandit PCA

Principal Component Analysis with
• sequentially chosen projections (online PCA)
• partial observability (bandit PCA)

$t = 1$

learner chooses projection $\mathbf{w}_t$

environment chooses hidden vector $x_t$
Bandit PCA

Principal Component Analysis with
• sequentially chosen projections (online PCA)
• partial observability (bandit PCA)

$t = 1$

environment chooses hidden vector $x_t$

Learner incurs and observes projection loss $1 - (w_t^T x_t)^2$

learner chooses projection $w_t$
Bandit PCA

Principal Component Analysis with
• sequentially chosen projections (online PCA)
• partial observability (bandit PCA)
Bandit PCA

Principal Component Analysis with
• sequentially chosen projections (online PCA)
• partial observability (bandit PCA)

\[ t = 1 \quad t = 2 \]

environment chooses hidden vector \( x_t \)
Bandit PCA

Principal Component Analysis with
• sequentially chosen projections (online PCA)
• partial observability (bandit PCA)

\[ t = 1 \quad \text{learner chooses projection } w_t \]
\[ t = 2 \quad \text{environment chooses hidden vector } x_t \]
Bandit PCA

Principal Component Analysis with
• sequentially chosen projections (online PCA)
• partial observability (bandit PCA)

\[ t = 1 \quad t = 2 \]

learner chooses projection \( w_t \)

Learner incurs and observes projection loss \( 1 - (w_t^T x_t)^2 \)

environment chooses hidden vector \( x_t \)
Bandit PCA

Principal Component Analysis with

• sequentially chosen projections (online PCA)
• partial observability (bandit PCA)

$t = 1$  $t = 2$  $t = 3$  $t = 4$  ...
Bandit PCA

Principal Component Analysis with
• sequentially chosen projections (online PCA)
• partial observability (bandit PCA)

\[ t = 1 \quad t = 2 \quad t = 3 \quad t = 4 \]

GOAL:
minimize total projection loss
Bandit PCA

Principal Component Analysis with
• sequentially chosen projections (online PCA)
• partial observability (bandit PCA)

\[
t = 1 \quad t = 2 \quad t = 3 \quad t = 4
\]

Bandit problem:
true \( x_t \) is never observed!

GOAL:
minimize total projection loss
Why is this hard?
Why is this hard?

\( x_t \) is ambiguous!
Why is this hard?

$x_t$ is ambiguous!
Why is this hard?

\[ x_t \] is ambiguous!

even worse in high dim
Why is this hard?

\( x_t \) is ambiguous!
even worse in high dim

**Challenge:**
How do we reconstruct \( x_t \) from a **single** projection?

NB: \( \pm x_t \) are impossible to tell apart with any number of projections
Why is this interesting?
Why is this interesting?

Bandit PCA \approx \text{online phase retrieval}
Why is this interesting?

Bandit PCA $\approx$ online phase retrieval

Phase retrieval:
- $\mathbf{w}_t \sim \mathcal{N}(0, I_{d \times d})$ i.i.d.
- $\mathbf{x}_t = \mathbf{x}$ fixed
- Observations:
  $$|\mathbf{x}^\top \mathbf{w}_t|^2 (+\text{noise})$$

Fienup (1982), Millane (1990)
Why is this interesting?

**Bandit PCA ≈ online phase retrieval**

Phase retrieval:
- \( \mathbf{w}_t \sim \mathcal{N}(0, I_{d \times d}) \) i.i.d.
- \( \mathbf{x}_t = \mathbf{x} \) fixed
- Observations:
  \[ |\mathbf{x}^\top \mathbf{w}_t|^2 (+\text{noise}) \]
- Applications in:
  - **diffractive imaging**
  - **X-ray crystallography**
  - **astronomy**...

Fienup (1982), Millane (1990)
Why is this interesting?

Bandit PCA ≈ online phase retrieval

Phase retrieval:
• $\mathbf{w}_t \sim \mathcal{N}(0, I_{d \times d})$ i.i.d.
• $\mathbf{x}_t = \mathbf{x}$ fixed
• Observations: $|\mathbf{x}^T \mathbf{w}_t|^2$ (+noise)
• Applications in
  • diffractive imaging
  • X-ray crystallography
  • astronomy...

Bandit PCA:
• $\mathbf{w}_t$ chosen adaptively
• $\mathbf{x}_t$ arbitrary
• Observations: $|\mathbf{x}_t^T \mathbf{w}_t|^2$ (+noise)

Fienup (1982), Millane (1990)
Why is this interesting?

Bandit PCA ≈ online phase retrieval

Phase retrieval:
• \( \mathbf{w}_t \sim \mathcal{N}(0, I_{d \times d}) \) i.i.d.
• \( \mathbf{x}_t = \mathbf{x} \) fixed
• Observations:
  \[ |\mathbf{x}^T \mathbf{w}_t|^2 \text{ (+noise)} \]
• Applications in
  • diffractive imaging
  • X-ray crystallography
  • astronomy...

Bandit PCA:
• \( \mathbf{w}_t \) chosen adaptively
• \( \mathbf{x}_t \) arbitrary
• Observations:
  \[ |\mathbf{x}_t^T \mathbf{w}_t|^2 \text{ (+noise)} \]

Applicable in the same settings but with adaptive measurements!

Fienup (1982), Millane (1990)
Let’s get technical

Classic tricks for online PCA
Bandit PCA – general framework

For $t = 1, 2, \ldots, T$

- Environment picks secret loss matrix $L_t$
- Learner picks unit-norm vector $w_t$
- Learner incurs and observes loss $w_t^T L_t w_t$

Generalizes the basic PCA setup with $L_t = x_t x_t^T$
Bandit PCA – general framework

**For** \( t = 1, 2, \ldots, T \)

- Environment picks secret loss matrix \( L_t \)
- Learner picks unit-norm vector \( w_t \)
- Learner incurs and observes loss \( w_t^T L_t w_t \)

Generalizes the basic PCA setup with \( L_t = x_t x_t^T \)

**GOAL:**

minimize total expected regret

\[
\text{regret}_T = \max_{u: \|u\| = 1} \mathbb{E} \left[ \sum_{t=1}^{T} (w_t^T L_t w_t - u^T L_t u) \right]
\]
Bandit PCA – general framework

For $t = 1, 2, \ldots, T$
- Environment picks secret loss matrix $L_t$
- Learner picks unit-norm vector $w_t$
- Learner incurs and observes loss $w_t^T L_t w_t$

**GOAL:**

minimize total expected regret

$$\text{regret}_T = \max_{u: \|u\| = 1} \mathbb{E} \left[ \sum_{t=1}^{T} (w_t^T L_t w_t - u^T L_t u) \right]$$

Generalizes the basic PCA setup with $L_t = x_t x_t^T$

Nonlinear loss!!!!
Linearizing the losses: SDP formulation


Observation #1:

• Loss is linear in matrix variable $w_tw_t^T$:

$$w_t^T L_tw_t = \text{tr}(w_tw_t^T L_t)$$
Linearizing the losses: SDP formulation


**Observation #1:**

- Loss is linear in matrix variable $\mathbf{w}_t \mathbf{w}_t^T$:
  \[
  \mathbf{w}_t^T \mathbf{L}_t \mathbf{w}_t = \text{tr}(\mathbf{w}_t \mathbf{w}_t^T \mathbf{L}_t)
  \]

**Observation #2:**

- The non-convex set of matrices $\mathbf{w} \mathbf{w}^T$ can be convexified through randomization:
  \[
  \mathcal{S} = \text{conv}(\mathbf{w} \mathbf{w}^T: \|\mathbf{w}\| = 1) = \{\mathbf{W}: \mathbf{W} \succeq 0, \text{tr}(\mathbf{W}) = 1\}
  \]
Bandit PCA
= Bandit linear optimization

For $t = 1, 2, \ldots, T$

- Environment picks secret loss matrix $L_t$
- Learner picks density matrix $W_t \in S$
- Learner draws random $w_t$ s.t. $\mathbb{E}[w_t w_t^\top] = W_t$
- Learner incurs and observes loss
  \[ \langle w_t w_t^\top, L_t \rangle \overset{\text{def}}{=} \text{tr}(w_t w_t^\top L_t) \]
Bandit PCA
= Bandit linear optimization

**Idea:**
Apply a generic linear bandit algorithm!

For $t = 1, 2, ..., T$
- Environment picks secret loss matrix $L_t$
- Learner picks density matrix $W_t \in S$
- Learner draws random $w_t$ s.t. $\mathbb{E}[w_tw_t^T] = W_t$
- Learner incurs and observes loss
  \[ \langle w_tw_t^T, L_t \rangle \overset{\text{def}}{=} \text{tr}(w_tw_t^T L_t) \]
Bandit PCA
= Bandit linear optimization

Idea:
Apply a generic linear bandit algorithm!

For \( t = 1, 2, ..., T \)
- Environment picks secret loss matrix \( L_t \)
- Learner picks density matrix \( W_t \in S \)
- Learner draws random \( w_t \) s.t. \( \mathbb{E}[w_tw_t^T] = W_t \)
- Learner incurs and observes loss
  \[ \langle w_t w_t^T, L_t \rangle \overset{\text{def}}{=} \text{tr}(w_t w_t^T L_t) \]

GeometricHedge guarantees

\[ \text{regret}_T = \tilde{O}(d^2 \sqrt{T}) \]

Dani, Hayes, Kakade (2008),
Bubeck and Eldan (2015)
Bandit PCA
= Bandit linear optimization

For $t = 1, 2, ..., T$
- Environment picks secret loss matrix $L_t$
- Learner picks density matrix $W_t \in S$
- Learner draws random $w_t$ s.t. $E[w_t w_t^T] = W_t$
- Learner incurs and observes loss $\langle w_t w_t^T, L_t \rangle \overset{\text{def}}{=} \text{tr}(w_t w_t^T L_t)$

**Idea:**
Apply a generic linear bandit algorithm!

**GeometricHedge guarantees**
regret$_T = \tilde{O}(d^2 \sqrt{T})$


**BUT**
no polytime implementation is known 😞😞😞
Bandit PCA = Bandit linear optimization

Idea:
Apply a generic linear bandit algorithm!

For $t = 1, 2, ..., T$
- Environment picks secret loss matrix $L_t$
- Learner picks density matrix $W_t \in S$
- Learner draws random $w_t$ s.t. $\mathbb{E}[w_tw_t^T] = W_t$

Our contribution:
a fast algorithm with regret $O\left(d^{3/2}\sqrt{T \log T}\right)$

Geometric Hedge guarantees regret $\sum_{t=1}^T L_t \leq \tilde{O}(d^2 T)$

Dani, Hayes, Kakade (2008),
Bubeck and Eldan (2015)

But no polytime implementation is known 😞😞😞
Main course | Algorithm
            | Main results
Online Mirror Descent for bandit PCA

**Idea:** rely on the good old template

$$W_{t+1} = \arg\min_{W \in \mathcal{S}} \{ \eta \langle W, \hat{L}_t \rangle + D(\| W \| \| W_t \|) \}$$
Online Mirror Descent for bandit PCA

Idea: rely on the good old template

\[
W_{t+1} = \arg \min_{W \in \mathcal{S}} \left\{ \eta \langle W, \hat{L}_t \rangle + D(W \| W_t) \right\}
\]

+ Sample \( w_t \) so that

\[
\mathbb{E}[w_tw_t^\top] = W_t
\]
Online Mirror Descent for bandit PCA

\[ W_{t+1} = \arg \min_{W \in S} \left\{ \eta \langle W, \hat{L}_t \rangle + D(W \| W_t) \right\} \]

Idea: rely on the good old template

Sample \( w_t \) so that

\[ \mathbb{E}[w_t w_t^\top] = W_t \]

loss estimate \( \hat{L}_t = ? \)

divergence \( D = ? \)

+ sampling scheme?
loss estimate $\hat{L}_t = ?$

sampling scheme?
Sampling scheme?

First thought:

decompose $W_t = \sum_i \lambda_i u_i u_i^T$

and sample $w_t$ so that $P[w_t = u_i] = \lambda_i$

Recall:

$\sum_i \lambda_i = \text{tr}(W) = 1$

$\lambda_i \geq 0$

Warmuth and Kuzmin (2006)
Sampling scheme?

First thought:

Decompose $W_t = \sum_i \lambda_i u_i u_i^T$

and sample $w_t$ so that $\mathbb{P}[w_t = u_i] = \lambda_i$

Recall: $\sum_i \lambda_i = \text{tr}(W) = 1$

$\lambda_i \geq 0$

Warmuth and Kuzmin (2006)

Unbiased: $\mathbb{E}[w_t w_t^T] = W_t$
Sampling scheme?

First thought:

Decompose \( W_t = \sum_i \lambda_i u_i u_i^\top \)
and sample \( w_t \) so that \( \mathbb{P}[w_t = u_i] = \lambda_i \)

Recall: \( \sum_i \lambda_i = \text{tr}(W) = 1 \)
\( \lambda_i \geq 0 \)

Unbiased: \( \mathbb{E}[w_t w_t^\top] = W_t \)

Only senses “diagonal” elements of \( L_t \) 😞 😞
Sampling scheme?

First thought:

\[ W_t = \sum_i \lambda_i u_i u_i^T \]

and sample \( w_t \) so that \( \mathbb{P}[w_t = u_i] = \lambda_i \)

Unbiased: \( \mathbb{E}[w_t w_t^T] = W_t \)

Recall:

\[ \sum_i \lambda_i = \text{tr}(W) = 1 \]
\[ \lambda_i \geq 0 \]

only senses “diagonal” elements of \( L_t \) 😞😞
Sampling done right

**Sparse sampling**

- sample **two** indices $i, j \sim \lambda$
- if $i = j$, set $w_t = u_i$
- otherwise draw random sign $s \in \{-1, 1\}$ and set $w_t = \frac{1}{\sqrt{2}} (u_i + su_j)$
**Sparse sampling**

- sample **two** indices $i, j \sim \lambda$
- if $i = j$, set $w_t = u_i$
- otherwise draw random sign $s \in \{-1, 1\}$ and set $w_t = \frac{1}{\sqrt{2}}(u_i + su_j)$

**Sampling done right**
Sparse sampling

• sample two indices
  \( i, j \sim \lambda \)
• if \( i = j \), set
  \( w_t = u_i \)
• otherwise draw random sign \( s \in \{-1,1\} \) and set
  \( w_t = \frac{1}{\sqrt{2}} (u_i + su_j) \)
Sparse sampling

- sample two indices $i, j \sim \lambda$
- if $i = j$, set $w_t = u_i$
- otherwise draw random sign $s \in \{-1, 1\}$ and set $w_t = \frac{1}{\sqrt{2}} (u_i + su_j)$

Sampling done right
Sparse sampling

• sample **two** indices
  \[ i, j \sim \lambda \]

• if \( i = j \), set
  \[ w_t = u_i \]

• otherwise draw random sign \( s \in \{-1, 1\} \) and set
  \[ w_t = \frac{1}{\sqrt{2}} \left( u_i + su_j \right) \]

Sampling done right
Sampling and loss estimation done right

**Sparse sampling**
- sample **two** indices $i, j \sim \lambda$
- if $i = j$, set $w_t = u_i$
- otherwise draw random sign $s \in \{-1, 1\}$ and set $w_t = \frac{1}{\sqrt{2}}(u_i + su_j)$

**Loss estimation**
- let $\ell = w_t^T L_t w_t$
- if $i = j$, set $\hat{L}_t = \frac{\ell}{\lambda_i^2} u_i u_i^T$
- otherwise set $\hat{L}_t = \frac{s \ell}{\lambda_i \lambda_j} (u_j u_i^T + u_i u_j^T)$
Sampling and loss estimation done right

Sparse sampling
• sample two indices $i, j \sim \lambda$
• if $i = j$, set $w_t = u_i$
• otherwise draw random sign $s \in \{-1,1\}$ and set $w_t = \frac{1}{2} u_i + s u_j$

Loss estimation
• let $\ell = w_t^T L_t w_t$
• if $i = j$, set $\hat{L}_t = \frac{\ell}{\lambda_i^2} u_i u_i^T$
• otherwise set $\hat{L}_t = \frac{s \ell}{\lambda_i} (u_i u_i^T + u_j u_j^T)$

Lemma:
$\mathbb{E}[w_t w_t^T] = W_t$

Lemma:
$\mathbb{E}[\hat{L}_t] = L_t$
divergence $D = ?$
What divergence?

First thought:
the usual quantum relative entropy
\[ D(W||U) = W \log(WU^{-1}) \]
induced by the quantum entropy \( R(W) = W \log W \)
What divergence?

First thought:
the usual quantum relative entropy

\[ D(W \| U) = W \log(WU^{-1}) \]
induced by the quantum entropy \( R(W) = W \log W \)

a.k.a. “Matrix Hedge”
Warmuth and Kuzmin (2006)
Matrix Hedge for Bandit PCA does not work?

W.K.

June 25, 2018

Consider the adversarial $k = 1$ PCA with bandit feedback. In each trial, the algorithm plays with a rank-one matrix $w_t w_t^\top$ with $w_t \in \mathbb{R}^d$, $\|w_t\| = 1$. Then, nature chooses a symmetric loss matrix $L_t \in \mathbb{R}^{d \times d}$ with eigenvalues bounded in $[0, 1]$, and the algorithm receives and observes loss $\ell_t = \text{tr}(w_t w_t^\top L_t)$.

We start with a standard bound on the Matrix Hedge algorithm: for any loss sequence $\tilde{L}_1, \ldots, \tilde{L}_T$ such that each $\tilde{L}_t$ has eigenvalues in the range $[-a, \infty)$, the sequence of density matrices $W_1, \ldots, W_T$ produced by Matrix Hedge with fixed learning rate $\eta$ has regret against a comparator density matrix $U$ upper-bounded by:

$$\text{regret}_T(U) = \sum_{t=1}^T \text{tr}((W_t - U)\tilde{L}_t) \leq \frac{\ln d}{\eta} + \kappa(\eta a) \eta \sum_{t=1}^T \text{tr}(W_t \tilde{L}_t^2),$$

where $\kappa(x) = \frac{e^x - x - 1}{x^2}$. The trick is now to use this bound in the bandit case as follows: in each trial $t = 1, \ldots, T$, the algorithm probabilistically chooses $w_t w_t^\top$ such that $\mathbb{E}_t[w_t w_t^\top] = W_t$ (where $\mathbb{E}_t[\cdot]$ denotes the conditional expectation with respect the randomness at trial $t$, conditioned on all the past); then, the algorithm observes $\ell_t$ and produced an estimate $\tilde{L}_t$ of the loss matrix $L_t$, with eigenvalues in $[-a, \infty]$, such that $\mathbb{E}_t[\tilde{L}_t] = L_t + c_t I$ (the estimate is allowed to be biased by a multiplicity of identity matrix!). The expected regret of the algorithm is given by:

$$\begin{bmatrix} T \\ \end{bmatrix} \begin{bmatrix} \eta \end{bmatrix}$$
Matrix Hedge for Bandit PCA does not work?

W.K.

June 25, 2018

Doesn’t work indeed 😞😞😞😞

In each trial, the algorithm plays with a symmetric loss matrix $L_t \in \mathbb{S}_{d\times d}$, chooses a symmetric loss matrix $L_t \in \mathbb{S}_{d\times d}$ and observes loss $\ell_t = \text{tr}(w_t w_t^T L_t)$. We start with a standard bound on the Matrix Hedge algorithm: for any loss sequence $\tilde{L}_1, \ldots, \tilde{L}_T$ such that each $\tilde{L}_t$ has eigenvalues in the range $[-a, \infty)$, the sequence of density matrices $W_1, \ldots, W_T$ produced by Matrix Hedge with fixed learning rate $\eta$ has regret against a comparator density matrix $U$ upper-bounded by:

$$
\text{regret}_T(U) = \sum_{t=1}^{T} \text{tr}((W_t - U)\tilde{L}_t) \leq \frac{\ln d}{\eta} + \kappa(\eta a) \eta \sum_{t=1}^{T} \text{tr}(W_t\tilde{L}_t^2),
$$

where $\kappa(x) = \frac{e^x - x - 1}{x^2}$. The trick is now to use this bound in the bandit case as follows: in each trial $t = 1, \ldots, T$, the algorithm probabilistically chooses $w_t w_t^T$ such that $\mathbb{E}_t[w_t w_t^T] = W_t$ (where $\mathbb{E}_t[\cdot]$ denotes the conditional expectation with respect the randomness at trial $t$, conditioned on all the past); then, the algorithm observes $\ell_t$ and produced an estimate $\tilde{L}_t$ of the loss matrix $L_t$, with eigenvalues in $[-a, \infty]$, such that $\mathbb{E}_t[\tilde{L}_t] = L_t + c_t I$ (the estimate is allowed to be biased by a multiplicity of identity matrix!). The expected regret of the algorithm is given by:

$$
\sum_{t=1}^{T} \text{tr}(W_t \tilde{L}_t^2) = \sum_{t=1}^{T} \text{tr}(W_t (L_t + c_t I)^2) 
$$
Matrix Hedge for Bandit PCA does not work?

W.K.

June 25, 2018

Doesn’t work indeed 😞😞😞😞

In each trial, the algorithm plays with a loss function that chooses a symmetric loss matrix $L_t \in \mathbb{R}^{d \times d}$ and observes loss $\ell_t = \text{tr}(w_t w_t^T L_t)$.

We start with a standard bound on the Matrix Hedge algorithm: for any loss sequence $\tilde{L}_1, \ldots, \tilde{L}_T$ such that each $\tilde{L}_t$ has eigenvalues in the range $[-a, \infty)$, the sequence of density matrices $W_1, \ldots, W_T$ produced by Matrix Hedge with fixed learning rate $\eta$ has regret against a comparator density matrix $U$

$$
\text{regret}_T(U) = \sum_{t=1}^{T} \text{tr}((W_t - U)\tilde{L}_t) \leq \frac{\ln d}{\eta} + \kappa(\eta a) \eta \sum_{t=1}^{T} \text{tr}(W_t \tilde{L}_t^2),
$$

where $\kappa(x) = \frac{e^x - x - 1}{x^2}$. The trick is now to use this bound in the bandit case as follows: in each trial $t = 1, \ldots, T$, the algorithm probabilistically chooses the arm $a_t$ such that $E_t[\ell_t a_t^T] = W_t$ (where $E_t[\cdot]$ denotes the conditional expectation with respect to the observed losses up to time $t$); then, the algorithm observes $\ell_t$ and produces a new density matrix $W_t$ with eigenvalues in $[-a, \infty)$, and $L_t + c_t I$ (the expected regret of the algorithm is very high).

This bound is virtually useless (for complicated reasons)
The right divergence

\[ D(W \| U) = \text{tr}(WU^{-1}) - \log \det(WU^{-1}) - d \]

The Bregman divergence induced by

\[ R(W) = -\log \det W \]

a.k.a. Stein’s loss (James and Stein, 1967)
The right divergence

\[ D(W\|U) = \text{tr}(WU^{-1}) - \log \det(WU^{-1}) - d \]

The Bregman divergence induced by

\[ R(W) = -\log \det W \]

a.k.a. Stein’s loss (James and Stein, 1967)

The matrix generalization of the trendy “log-barrier” regularizer – \( \sum_i \log p_i \)

(Foster et al., 2016, Agarwal et al., 2017, Bubeck et al. 2018, Wei and Luo, 2018, Luo et al., 2018, ...)
Online Mirror Descent for bandit PCA

\[
W_{t+1} = \arg \min_{W \in \mathcal{S}} \left\{ \eta \langle W, \hat{L}_t \rangle + D(W \| W_t) \right\}
\]

- loss estimate $\hat{L}_t =$?
- divergence $D =$?
- sampling scheme?

Sample $w_t$ so that
\[
\mathbb{E}[w_tw_t^T] = W_t
\]
Online Mirror Descent for bandit PCA

\[ W_{t+1} = \arg \min_{W \in \mathcal{S}} \{ \eta \langle W, \hat{L}_t \rangle + D(W \| W_t) \} \]

loss estimate \( \hat{L}_t = \) divergence \( D = ? \)

Sample \( w_t \) so that
\[ \mathbb{E}[w_tw_t^T] = W_t \]

+ sampling scheme?
Main course  |  Algorithm
Main results
Main result #1: upper bounds

**Theorem**

\[
\text{regret}_T \leq \frac{d \log T}{\eta} + \eta d \sum_{t=1}^{T} \|L_t\|_F^2
\]

**For rank-1 losses:**

\[
\text{regret}_T = \Theta(d \sqrt{T \log T})
\]

**In general:**

\[
\text{regret}_T = \Theta(d^{3/2} \sqrt{T \log T})
\]
Main result #2: lower bound

Theorem
There is a problem instance on which any algorithm will suffer
\[ \text{regret}_T = \Omega\left( d\sqrt{T/\log T} \right) \]
Dessert | Fast implementation
Implementing the update

\[ W_{t+1} = \arg \min_{W \in S} \{ \eta \langle W, \hat{L}_t \rangle + D(W \parallel W_t) \} \]
Implementing the update

\[
W_{t+1} = \arg\min_{W \in S} \{ \eta \langle W, \hat{L}_t \rangle + D(W\|W_t) \}
\]

= by the classic decomposition

\[
\bar{W}_{t+1} = \arg\min_{W} \{ \eta \langle W, \hat{L}_t \rangle + D(W\|W_t) \}
\]
\[
W_{t+1} = \arg\min_{W \in S} D(W\|\bar{W}_{t+1})
\]
Implementing the update

\[ W_{t+1} = \arg\min_{W \in \mathcal{S}} \{ \eta \langle W, \hat{L}_t \rangle + D(W \| W_t) \} \]

= by the classic decomposition

\[ \widetilde{W}_{t+1} = (W_t^{-1} + \eta \hat{L}_t)^{-1} \]
\[ W_{t+1} = \text{renormalize}(\widetilde{W}_{t+1}) \]
Implementing the update

\[ W_{t+1} = \arg \min_{W \in S} \{ \eta \langle W, \hat{L}_t \rangle + D(W \| W_t) \} \]

= by the classic decomposition

\[ \overline{W}_{t+1} = (W_t^{-1} + \eta \hat{L}_t)^{-1} \]
\[ W_{t+1} = \text{renormalize}(\overline{W}_{t+1}) \]

takes \( O(d^3) \) time in general
Implementing the update

\[ W_{t+1} = \arg \min_{W \in \mathcal{S}} \{ \eta \langle W, \hat{L}_t \rangle + D(W \| W_t) \} \]

= by the classic decomposition

\[ \tilde{W}_{t+1} = (W_t^{-1} + \eta \hat{L}_t)^{-1} \]

\[ W_{t+1} = \text{renormalize}(\tilde{W}_{t+1}) \]

takes \( \mathcal{O}(d^3) \) time in general

BUT ONLY \( \mathcal{O}(d) \) TIME IN OUR CASE!!
Updating in $\mathcal{O}(d)$ time

$W_t =$
Updating in $O(d)$ time

$W_t = \begin{array}{c}
\end{array}$

$\hat{L}_t = \begin{array}{c}
\end{array}$
Updating in $O(d)$ time

\[ W_t = \cdot \]

\[ \hat{W}_{t+1} = \cdot \]

- Computing $\hat{L}: O(1)$ time
- Computing $\hat{W}: O(1)$ time
Updating in $\mathcal{O}(d)$ time

\[
W_t = \ldots
\]

\[
\bar{W}_{t+1} = \ldots
\]

- Computing $\hat{L}$: $\mathcal{O}(1)$ time
- Computing $\bar{W}$: $\mathcal{O}(1)$ time
- Computing new eigenvectors: $\mathcal{O}(d)$ time
Updating in $\mathcal{O}(d)$ time

\[ W_t = \cdot \]

\[ \bar{W}_{t+1} = \cdot \]

- Computing $\hat{L}$: $\mathcal{O}(1)$ time
- Computing $\bar{W}$: $\mathcal{O}(1)$ time
- Computing new eigenvectors: $\mathcal{O}(d)$ time
- Renormalization: $\mathcal{O}(d)$ time
Updating in $\mathcal{O}(d)$ time

$$W_t =$$

**UPDATING TAKES LESS TIME THAN READING THE FULL LOSS MATRIX $L_t$!!!**

$$\vec{W}_{t+1} =$$
## Summary

<table>
<thead>
<tr>
<th></th>
<th>Previous best</th>
<th>Our work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime</td>
<td>no polytime?</td>
<td>$d$</td>
</tr>
<tr>
<td>Upper bound</td>
<td>$d^2 \sqrt{T}$</td>
<td>$d^{3/2} \sqrt{T}$</td>
</tr>
<tr>
<td>Lower bound</td>
<td>$\sqrt{dT}$</td>
<td>$d \sqrt{T}$</td>
</tr>
</tbody>
</table>
## Summary

<table>
<thead>
<tr>
<th></th>
<th>Previous best</th>
<th>Our work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime</td>
<td>no polytime?</td>
<td>$d$</td>
</tr>
<tr>
<td>Upper bound</td>
<td>Still a gap of $\sqrt{d}$</td>
<td>$d^{3/2} \sqrt{T}$</td>
</tr>
<tr>
<td>Lower bound</td>
<td>❄️❄️❄️❄️</td>
<td>$d \sqrt{T}$</td>
</tr>
</tbody>
</table>

Still a gap of $\sqrt{d}$
Open problem: $d$ or $d^{3/2}$?

$d$ looks obvious, right?
Open problem: $d$ or $d^{3/2}$?

$d$ looks obvious, right?

- multi-armed bandits:
  \(d\) parameters to estimate \(\Rightarrow \sqrt{dT}\) regret
- bandit PCA:
  \(d^2\) parameters to estimate \(\Rightarrow d\sqrt{T}\) regret?
Open problem: $d$ or $d^{3/2}$?

$d$ looks obvious, right?

- multi-armed bandits:
  $d$ parameters to estimate $\Rightarrow \sqrt{dT}$ regret
- bandit PCA:
  $d^2$ parameters to estimate $\Rightarrow d\sqrt{T}$ regret?

NO:

Lemma:
For i.i.d. data, every non-adaptive algorithm will have error at least

$$\Omega\left(\frac{d^{3/2}}{\sqrt{T}}\right)$$
Open problem: $d$ or $d^{3/2}$?

**If true dependence is $\Theta(d)$:**
First known case with a gap between non-adaptive and adaptive algorithms!!!

**Lemma:**
For i.i.d. data, every non-adaptive algorithm will have error at least $\Omega\left(\frac{d^{3/2}}{\sqrt{T}}\right)$
Open problem:

tfaster rates for phase retrieval

Our bound for PR: \[ O \left( \frac{d}{\sqrt{T}} \right) \]

SOTA for PR: \[ O \left( \frac{d}{T} \right) \]
Open problem: faster rates for phase retrieval

Our bound for PR: $O\left(\frac{d}{\sqrt{T}}\right)$

SOTA for PR: $O\left(\frac{d}{T}\right)$

Why such a big gap?
Open problem: faster rates for phase retrieval

Our bound for PR: $O\left(\frac{d}{\sqrt{T}}\right)$

SOTA for PR: $O\left(\frac{d}{T}\right)$

Why such a big gap?
- i.i.d. assumption
- spiked covariance model

Can we exploit these to obtain even better rates?
Thanks!