Distributed sequential sampling for adaptive kernel DL and online learning

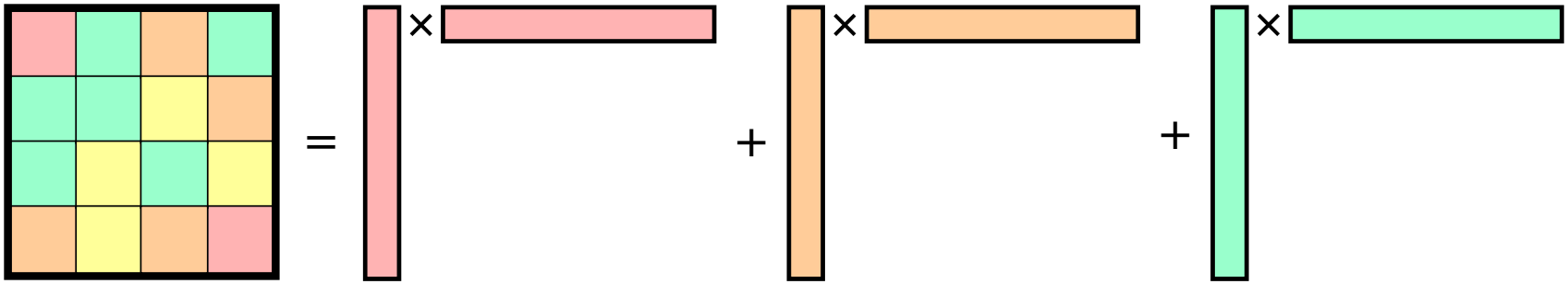
Daniele Calandriello, Alessandro Lazaric, **Michal Valko**
SequeL, Inria Lille – Nord Europe

DeepMind

DeepMind London, September 19, 2017, La France avance!

Distributed sequential sampling for adaptive kernel DL

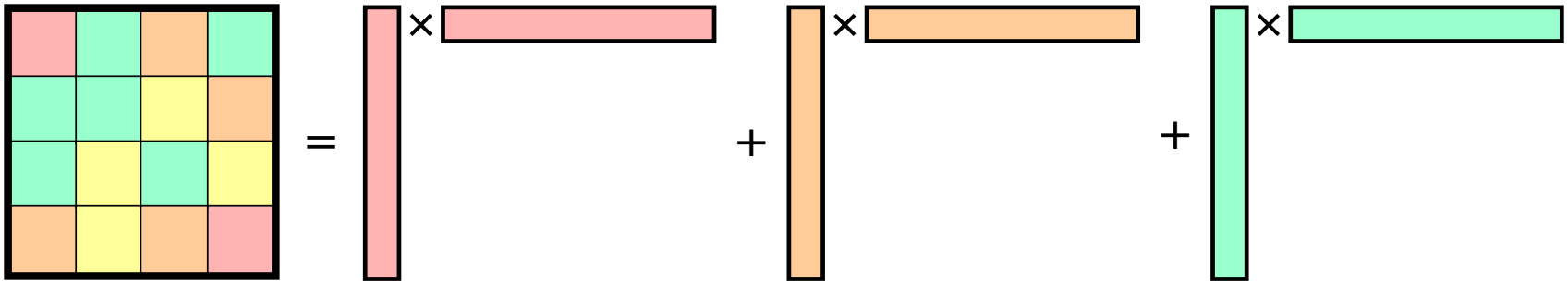
What is Dictionary Learning (DL)?



Finding an accurate representation of the input data as a linear combination of a small set of basic elements (atoms)

Distributed sequential sampling for adaptive kernel DL

What is Dictionary Learning (DL)?

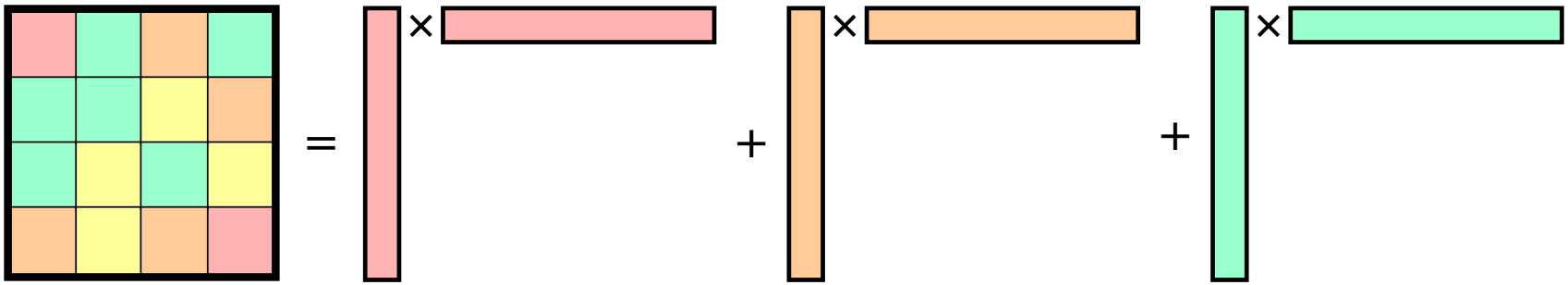


Finding an accurate representation of the input data as a linear combination of a small set of basic elements (atoms)

Representation/Unsupervised learning

Distributed sequential sampling for adaptive kernel DL

What is Dictionary Learning (DL)?



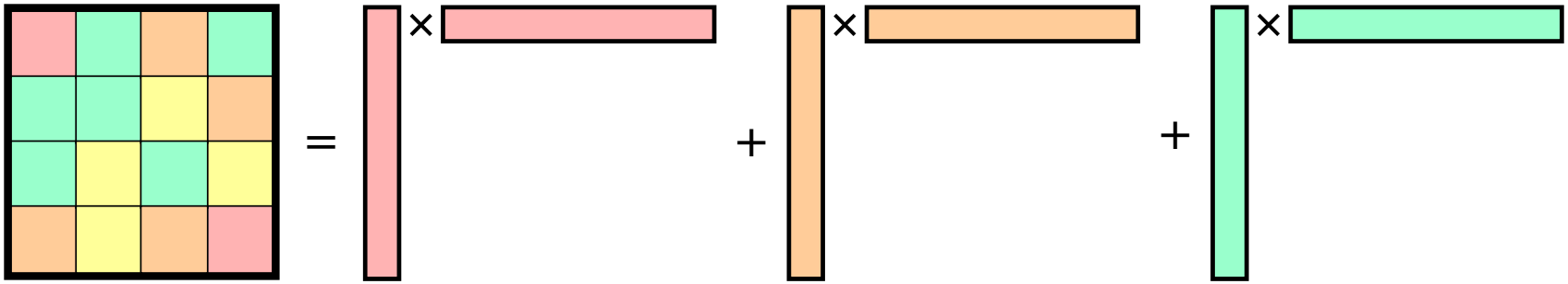
Finding an accurate representation of the input data as a linear combination of a small set of basic elements (atoms)

Representation/Unsupervised learning

> "Most important open problem in ML" **Y. LeCun**, NIPS 2016

Distributed sequential sampling for adaptive kernel DL

What is Dictionary Learning (DL)?



Finding an accurate representation of the input data as a linear combination of a small set of basic elements (atoms)

Representation/Unsupervised learning

"Most important open problem in ML" **Y. LeCun**, NIPS 2016

"Already solved" **J. Schmidhuber**, NIPS 2016

Distributed sequential sampling for adaptive kernel DL

Why DL for kernel problems?

Kernel methods have huge scalability problem

Problem: for a dataset $\mathcal{D}$ with n samples

- $\mathcal{O}(\mathbf{n^2})$ time to construct kernel matrix $\mathbf{K}$
- $\mathcal{O}(\mathbf{n^3})$ time to compute solution
- $\mathcal{O}(\mathbf{n^2})$ space to store it

Distributed sequential sampling for adaptive kernel DL

Why DL for kernel problems?

Kernel methods have huge scalability problem

Problem: for a dataset $\mathcal{D}$ with n samples

- $\mathcal{O}(\mathbf{n^2})$ time to construct kernel matrix $\mathbf{K}$
- $\mathcal{O}(\mathbf{n^3})$ time to compute solution
- $\mathcal{O}(\mathbf{n^2})$ space to store it

Solution:
compute accurate, small dictionary $\mathcal{I}$ to represent $\mathcal{D}$
compute approximate solution on $\mathcal{I}$ efficiently

Distributed sequential sampling for adaptive kernel DL

Why DL for kernel problems?

Problem: Existing DL methods guarantee either scalablility or accuracy

Distributed sequential sampling for adaptive kernel DL

Why DL for kernel problems?

Problem: Existing DL methods guarantee either scalablility or accuracy

we want both

Distributed sequential sampling for adaptive kernel DL

Why DL for kernel problems?

Problem: Existing DL methods guarantee either scalablility or accuracy

we want both

We present SQUEAK , a dictionary learning algorithm that guarantees

Distributed sequential sampling for adaptive kernel DL

Why DL for kernel problems?

Problem: Existing DL methods guarantee either scalablility or accuracy

we want both

We present SQUEAK , a dictionary learning algorithm that guarantees

In all cases accurate reconstruction of the input

Distributed sequential sampling for adaptive kernel DL

Why DL for kernel problems?

Problem: Existing DL methods guarantee either scalablility or accuracy

we want both

We present SQUEAK , a dictionary learning algorithm that guarantees

In all cases accurate reconstruction of the input

Adapts to the data:

on "easy" problems small ($\mathcal{O}(n)$) space/time requirements

on "hard" problems not worse than storing whole input

Distributed sequential sampling for adaptive kernel DL

Why DL for kernel problems?

Problem: Existing DL methods guarantee either scalablility or accuracy

we want both

We present SQUEAK , a dictionary learning algorithm that guarantees

- In all cases accurate reconstruction of the input
- Adapts to the data:
  - on "easy" problems small ($\mathcal{O}(n)$) space/time requirements
  - on "hard" problems not worse than storing whole input
- Only local data access, distributed version with $\mathcal{O}(\log n)$ runtime

Preliminaries: Setting and Kernels

Indexing $[t] = \{1, \dots, t\}$, notation $\mathbf{K}$ matrices, $\mathbf{k}$ vectors, k scalar
Dataset $\mathcal{D}_n = \{\mathbf{x}_i\}_{i=1}^n$, samples $\mathbf{x}_i \in \mathcal{X}$ (e.g., $\mathbb{R}^d$)

Kernel function	$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
Feature map	$\varphi(\mathbf{x}_i) : \mathcal{X} \to \mathcal{H} = \boldsymbol{\phi}_i$
Kernel trick	$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}_i^\mathsf{T} \boldsymbol{\phi}_j$
Feature matrix	$\boldsymbol{\Phi}_t = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \dots, \boldsymbol{\phi}_t] : \mathbb{R}^t \to \mathcal{H}$
Empirical kernel matrix	$\mathbf{K}_t \in \mathbb{R}^{t \times t} = \mathbf{K}_{[t],[t]} = \boldsymbol{\Phi}_t^\mathsf{T} \boldsymbol{\Phi}_t$
New column	$\mathbf{k}_{[t-1],t} \in \mathbb{R}^{t-1} = \boldsymbol{\Phi}_{t-1}^\mathsf{T} \boldsymbol{\phi}_t$
Kernel at a point	$k_{t,t} \in \mathbb{R} = \boldsymbol{\phi}_t^\mathsf{T} \boldsymbol{\phi}_t$

Preliminaries: Setting and Kernels

Indexing $[t] = \{1, \dots, t\}$, notation $\mathbf{K}$ matrices, $\mathbf{k}$ vectors, k scalar
Dataset $\mathcal{D}_n = \{\mathbf{x}_i\}_{i=1}^n$, samples $\mathbf{x}_i \in \mathcal{X}$ (e.g., $\mathbb{R}^d$)

Kernel function	$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
Feature map	$\varphi(\mathbf{x}_i) : \mathcal{X} \to \mathcal{H} = \boldsymbol{\phi}_i$
Kernel trick	$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}_i^\mathsf{T} \boldsymbol{\phi}_j$
Feature matrix	$\boldsymbol{\Phi}_t = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \dots, \boldsymbol{\phi}_t] : \mathbb{R}^t \to \mathcal{H}$
Empirical kernel matrix	$\mathbf{K}_t \in \mathbb{R}^{t \times t} = \mathbf{K}_{[t],[t]} = \boldsymbol{\Phi}_t^\mathsf{T} \boldsymbol{\Phi}_t$
New column	$\mathbf{k}_{[t-1],t} \in \mathbb{R}^{t-1} = \boldsymbol{\Phi}_{t-1}^\mathsf{T} \boldsymbol{\phi}_t$
Kernel at a point	$k_{t,t} \in \mathbb{R} = \boldsymbol{\phi}_t^\mathsf{T} \boldsymbol{\phi}_t$

Find a small dictionary $\mathcal{I} = \{(w_j, \phi_j)\}_{j=1}^m$ such that $\widetilde{\mathbf{K}} = f(\mathcal{I})$ close to $\mathbf{K}$

Example: kernel ridge regression

$$\widehat{w}_n = (\mathbf{K}_n + \gamma \mathbf{I})^{-1} \mathbf{y}_n$$
$$\widehat{y}_n = \mathbf{K}_n \widehat{w}_n = \mathbf{K}_n (\mathbf{K}_n + \gamma \mathbf{I})^{-1} \mathbf{y}_n = \mathbf{P}_n \mathbf{y}_n$$

Example: kernel ridge regression

$$\widehat{w}_n = (\mathbf{K}_n + \gamma \mathbf{I})^{-1} \mathbf{y}_n$$
$$\widehat{y}_n = \mathbf{K}_n \widehat{w}_n = \mathbf{K}_n (\mathbf{K}_n + \gamma \mathbf{I})^{-1} \mathbf{y}_n = \mathbf{P}_n \mathbf{y}_n$$

If we can have accurate low-rank approximations ...

$$\widetilde{\mathbf{K}}_n \preceq \mathbf{K}_n \preceq \widetilde{\mathbf{K}}_n + \frac{\gamma}{1-\varepsilon} \mathbf{I}$$

Example: kernel ridge regression

$$\widehat{w}_n = (\mathbf{K}_n + \gamma \mathbf{I})^{-1} \mathbf{y}_n$$
$$\widehat{y}_n = \mathbf{K}_n \widehat{w}_n = \mathbf{K}_n (\mathbf{K}_n + \gamma \mathbf{I})^{-1} \mathbf{y}_n = \mathbf{P}_n \mathbf{y}_n$$

If we can have accurate low-rank approximations ...

$$\widetilde{\mathbf{K}}_n \preceq \mathbf{K}_n \preceq \widetilde{\mathbf{K}}_n + \frac{\gamma}{1-\varepsilon} \mathbf{I}$$

... then we can used them for to get good approximate solutions:

$$\widetilde{w}_n = (\widetilde{\mathbf{K}}_n + \gamma \mathbf{I})^{-1} \mathbf{y}_n$$
$$R(\widetilde{w}_n) \leq \left(1 + \frac{1}{1-\varepsilon}\right) R(\widehat{w}_n)$$

Example: kernel ridge regression

$$\widehat{w}_n = (\mathbf{K}_n + \gamma \mathbf{I})^{-1} \mathbf{y}_n$$
$$\widehat{y}_n = \mathbf{K}_n \widehat{w}_n = \mathbf{K}_n (\mathbf{K}_n + \gamma \mathbf{I})^{-1} \mathbf{y}_n = \mathbf{P}_n \mathbf{y}_n$$

If we can have accurate low-rank approximations ...

$$\widetilde{\mathbf{K}}_n \preceq \mathbf{K}_n \preceq \widetilde{\mathbf{K}}_n + \frac{\gamma}{1-\varepsilon} \mathbf{I}$$

... then we can used them for to get good approximate solutions:

$$\widetilde{w}_n = (\widetilde{\mathbf{K}}_n + \gamma \mathbf{I})^{-1} \mathbf{y}_n$$
$$R(\widetilde{w}_n) \leq \left(1 + \frac{1}{1-\varepsilon}\right) R(\widehat{w}_n)$$

~~$\mathcal{O}(\mathbf{n^3})$~~ $\Rightarrow \mathcal{O}(\mathbf{nm} + \mathbf{m^3})$ time to compute the approx. solution

~~$\mathcal{O}(n^2)$~~ $\Rightarrow \mathcal{O}(\mathbf{nm})$ space to store dictionary

Distributed sequential sampling for adaptive kernel DL

We consider Positive Semi-Definite matrices

$$\mathbf{A} = \mathbf{A}^{1/2}(\mathbf{A}^{1/2})^\intercal = \sum_{i=1}^{m} \mathbf{a}_i \mathbf{a}_i^\intercal \qquad \widetilde{\mathbf{A}} = \sum_{i=1}^{m} w_i \mathbf{x}_i \mathbf{x}_i^\intercal$$

Method	w_i	$\mathbf{x}_i$	Accuracy	Space	Time
Whole Input	1	$\mathbf{a}_i$	★★★★★		

Distributed sequential sampling for adaptive kernel DL

We consider Positive Semi-Definite matrices

$$\mathbf{A} = \mathbf{A}^{1/2}(\mathbf{A}^{1/2})^\intercal = \sum_{i=1}^{m} \mathbf{a}_i \mathbf{a}_i^\intercal \qquad \widetilde{\mathbf{A}} = \sum_{i=1}^{m} w_i \mathbf{x}_i \mathbf{x}_i^\intercal$$

Method	w_i	$\mathbf{x}_i$	Accuracy	Space	Time
Whole Input	1	$\mathbf{a}_i$	★★★★★		
Empty dictionary	0	$\mathbf{0}$		★★★★★	★★★★★

Distributed sequential sampling for adaptive kernel DL

We consider Positive Semi-Definite matrices

$$\mathbf{A} = \mathbf{A}^{1/2}(\mathbf{A}^{1/2})^\top = \sum_{i=1}^{m} \mathbf{a}_i \mathbf{a}_i^\top \qquad \widetilde{\mathbf{A}} = \sum_{i=1}^{m} w_i \mathbf{x}_i \mathbf{x}_i^\top$$

Method	w_i	$\mathbf{x}_i$	Accuracy	Space	Time
Whole Input	1	$\mathbf{a}_i$	★★★★★		
PCA	λ_i	$\mathbf{u}_i$	★★★★★	★★★★★	★
Empty dictionary	0	$\mathbf{0}$		★★★★★	★★★★★

Distributed sequential sampling for adaptive kernel DL

We consider Positive Semi-Definite matrices

$$\mathbf{A} = \mathbf{A}^{1/2}(\mathbf{A}^{1/2})^\intercal = \sum_{i=1}^{m} \mathbf{a}_i \mathbf{a}_i^\intercal \qquad \widetilde{\mathbf{A}} = \sum_{i=1}^{m} w_i \mathbf{x}_i \mathbf{x}_i^\intercal$$

Method	w_i	$\mathbf{x}_i$	Accuracy	Space	Time
Whole Input	1	$\mathbf{a}_i$	★★★★★		
PCA	λ_i	$\mathbf{u}_i$	★★★★★	★★★★★	★
RLS (this)	$1/\tau_i$	$\mathbf{a}_i$	★★★★	★★★	★★★
Empty dictionary	0	$\mathbf{0}$		★★★★★	★★★★★

Distributed sequential sampling for adaptive kernel DL

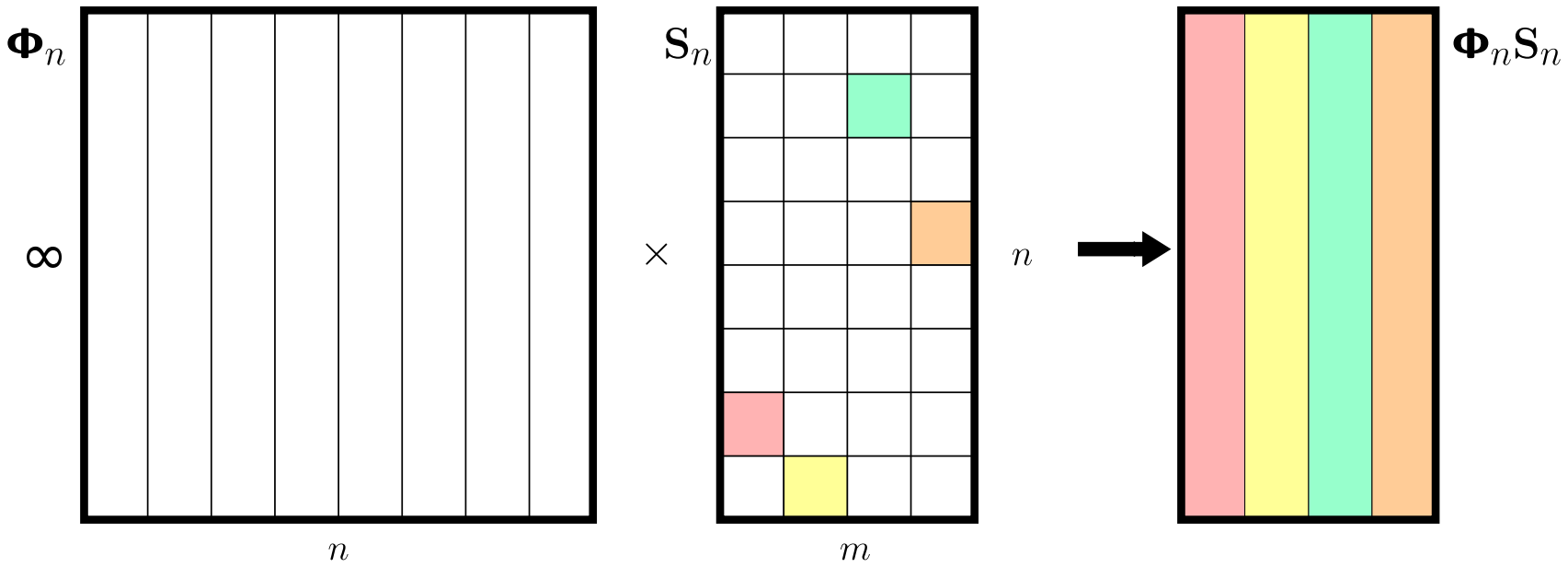
We consider Positive Semi-Definite matrices

$$\mathbf{A} = \mathbf{A}^{1/2}(\mathbf{A}^{1/2})^\intercal = \sum_{i=1}^{m} \mathbf{a}_i \mathbf{a}_i^\intercal \qquad \widetilde{\mathbf{A}} = \sum_{i=1}^{m} w_i \mathbf{x}_i \mathbf{x}_i^\intercal$$

Method	w_i	$\mathbf{x}_i$	Accuracy	Space	Time
Whole Input	1	$\mathbf{a}_i$	★★★★★		
PCA	λ_i	$\mathbf{u}_i$	★★★★★	★★★★★	★
RLS (this)	$1/\tau_i$	$\mathbf{a}_i$	★★★★	★★★	★★★
Uniform	n/m	$\mathbf{a}_i$	★★	★★	★★★★
Empty dictionary	0	$\mathbf{0}$		★★★★★	★★★★★

Reconstruction Guarantees

Given dataset $\mathcal{D}_n$ and dictionary $\mathcal{I}_n$, the selection matrix $\mathbf{S}_n$ is defined as



$$\sum_{i=1}^{m} w_i \phi_i \phi_i^\mathsf{T} = \sum_{i=1}^{m} (\sqrt{w_i}\phi_i)(\sqrt{w_i}\phi_i)^\mathsf{T} = \mathbf{\Phi}_n \mathbf{S}_n \mathbf{S}_n^\mathsf{T} \mathbf{\Phi}_n^\mathsf{T}$$

Reconstruction guarantees

Consider the regularized projection $\mathbf{P}_n$

$$\mathbf{P}_n = \mathbf{K}_n(\mathbf{K}_n + \gamma\mathbf{I})^{-1}$$

$$= (\mathbf{K}_n + \gamma\mathbf{I})^{-1/2}\mathbf{K}_n^{1/2}\mathbf{K}_n^{1/2}(\mathbf{K}_n + \gamma\mathbf{I})^{-1/2} = \sum_{i=1}^{n} \mathbf{v}_i\mathbf{v}_i^\mathsf{T}$$

$$\widetilde{\mathbf{P}}_n = (\mathbf{K}_n + \gamma\mathbf{I})^{-1/2}\mathbf{K}_n^{1/2}\mathbf{S}_n\mathbf{S}_n^\mathsf{T}\mathbf{K}_n^{1/2}(\mathbf{K}_n + \gamma\mathbf{I})^{-1/2} = \sum_{j=1}^{m} w_j\mathbf{v}_j\mathbf{v}_j^\mathsf{T}$$

Reconstruction guarantees

Consider the regularized projection $\mathbf{P}_n$

$$\mathbf{P}_n = \mathbf{K}_n(\mathbf{K}_n + \gamma\mathbf{I})^{-1}$$

$$= (\mathbf{K}_n + \gamma\mathbf{I})^{-1/2}\mathbf{K}_n^{1/2}\mathbf{K}_n^{1/2}(\mathbf{K}_n + \gamma\mathbf{I})^{-1/2} = \sum_{i=1}^{n} \mathbf{v}_i\mathbf{v}_i^\mathsf{T}$$

$$\widetilde{\mathbf{P}}_n = (\mathbf{K}_n + \gamma\mathbf{I})^{-1/2}\mathbf{K}_n^{1/2}\mathbf{S}_n\mathbf{S}_n^\mathsf{T}\mathbf{K}_n^{1/2}(\mathbf{K}_n + \gamma\mathbf{I})^{-1/2} = \sum_{j=1}^{m} w_j\mathbf{v}_j\mathbf{v}_j^\mathsf{T}$$

An accurate dictionary satisfies

$$\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2 \leq \varepsilon$$

Reconstruction guarantees

Why would bounding $\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2$ be useful?

Reconstruction guarantees

Why would bounding $\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2$ be useful?

It appears in many problems e.g.,Kernel Ridge Regression

Reconstruction guarantees

Why would bounding $\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2$ be useful?

It appears in many problems e.g.,Kernel Ridge Regression

$$\widehat{\mathbf{w}}_n = (\mathbf{K}_n + \gamma \mathbf{I})^{-1} \mathbf{y}_n$$
$$\widehat{\mathbf{y}}_n = \mathbf{K}_n \widehat{\mathbf{w}}_n = \mathbf{K}_n (\mathbf{K}_n + \gamma \mathbf{I})^{-1} \mathbf{y}_n = \mathbf{P}_n \mathbf{y}_n$$

Reconstruction guarantees

Why would bounding $\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2$ be useful?

We can compute accurate low rank approximations. Let

$$\widetilde{\mathbf{K}}_n = \mathbf{K}_n\mathbf{S}_n(\mathbf{S}_n\mathbf{K}_n\mathbf{S}_n + \gamma\mathbf{I})^{-1}\mathbf{S}_n\mathbf{K}_n$$

then

$$\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2 \leq \varepsilon \Rightarrow \widetilde{\mathbf{K}}_n \preceq \mathbf{K}_n \preceq \widetilde{\mathbf{K}}_n + \frac{\gamma}{1-\varepsilon}\mathbf{I}$$

See Musco&Musco'16

Reconstruction guarantees

Why would bounding $\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2$ be useful?

We can compute accurate low rank approximations. Let

$$\widetilde{\mathbf{K}}_n = \mathbf{K}_n\mathbf{S}_n(\mathbf{S}_n\mathbf{K}_n\mathbf{S}_n + \gamma\mathbf{I})^{-1}\mathbf{S}_n\mathbf{K}_n$$

then

$$\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2 \leq \varepsilon \Rightarrow \widetilde{\mathbf{K}}_n \preceq \mathbf{K}_n \preceq \widetilde{\mathbf{K}}_n + \frac{\gamma}{1-\varepsilon}\mathbf{I}$$

e.g., Kernel Ridge Regression

$$\widetilde{\mathbf{w}}_n = (\widetilde{\mathbf{K}}_n + \gamma\mathbf{I})^{-1}\mathbf{y}_n$$

$$R(\widetilde{\mathbf{w}}_n) \leq \left(1 + \frac{1}{1-\varepsilon}\right) R(\widehat{\mathbf{w}}_n)$$

~~$\mathcal{O}(\mathbf{n^3})$~~ $\Rightarrow \mathcal{O}(\mathbf{nm} + \mathbf{m^3})$ time to compute the approx. solution

~~$\mathcal{O}(n^2)$~~ $\Rightarrow \mathcal{O}(\mathbf{nm})$ space to store dictionary

Reconstruction guarantees

Why would bounding $\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2$ be useful?

We can compute accurate low rank approximations. Let

$$\widetilde{\mathbf{K}}_n = \mathbf{K}_n\mathbf{S}_n(\mathbf{S}_n\mathbf{K}_n\mathbf{S}_n + \gamma\mathbf{I})^{-1}\mathbf{S}_n\mathbf{K}_n$$

then

$$\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2 \leq \varepsilon \Rightarrow \widetilde{\mathbf{K}}_n \preceq \mathbf{K}_n \preceq \widetilde{\mathbf{K}}_n + \frac{\gamma}{1-\varepsilon}\mathbf{I}$$

e.g., Kernel Ridge Regression* *Gaussian Processes

$$\widetilde{\mathbf{w}}_n = (\widetilde{\mathbf{K}}_n + \gamma\mathbf{I})^{-1}\mathbf{y}_n$$

$$R(\widetilde{\mathbf{w}}_n) \leq \left(1 + \frac{1}{1-\varepsilon}\right) R(\widehat{\mathbf{w}}_n)$$

~~$\mathcal{O}(\mathbf{n^3})$~~ $\Rightarrow \mathcal{O}(\mathbf{nm} + \mathbf{m^3})$ time to compute the approx. solution

~~$\mathcal{O}(n^2)$~~ $\Rightarrow \mathcal{O}(\mathbf{nm})$ space to store dictionary

Reconstruction guarantees

Why would bounding $\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2$ be useful?

We can compute accurate low rank approximations. Let

$$\widetilde{\mathbf{K}}_n = \mathbf{K}_n\mathbf{S}_n(\mathbf{S}_n\mathbf{K}_n\mathbf{S}_n + \gamma\mathbf{I})^{-1}\mathbf{S}_n\mathbf{K}_n$$

then

$$\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2 \leq \varepsilon \Rightarrow \widetilde{\mathbf{K}}_n \preceq \mathbf{K}_n \preceq \widetilde{\mathbf{K}}_n + \frac{\gamma}{1-\varepsilon}\mathbf{I}$$

e.g., Kernel PCA, $\mathbf{K}_n$ and $\widetilde{\mathbf{K}}_n$ have close leading eigenvalues/vectors
e.g., Kernel K-Means can be formulated as a quadratic form

$$\min_{\mathbf{C}} \mathrm{Tr}(\mathbf{K}_n - \mathbf{C}\mathbf{C}^\intercal\mathbf{K}_n\mathbf{C}\mathbf{C}^\intercal) \sim \min_{\widetilde{\mathbf{C}}} \mathrm{Tr}(\widetilde{\mathbf{K}}_n - \widetilde{\mathbf{C}}\widetilde{\mathbf{C}}^\intercal\widetilde{\mathbf{K}}_n\widetilde{\mathbf{C}}\widetilde{\mathbf{C}}^\intercal)$$

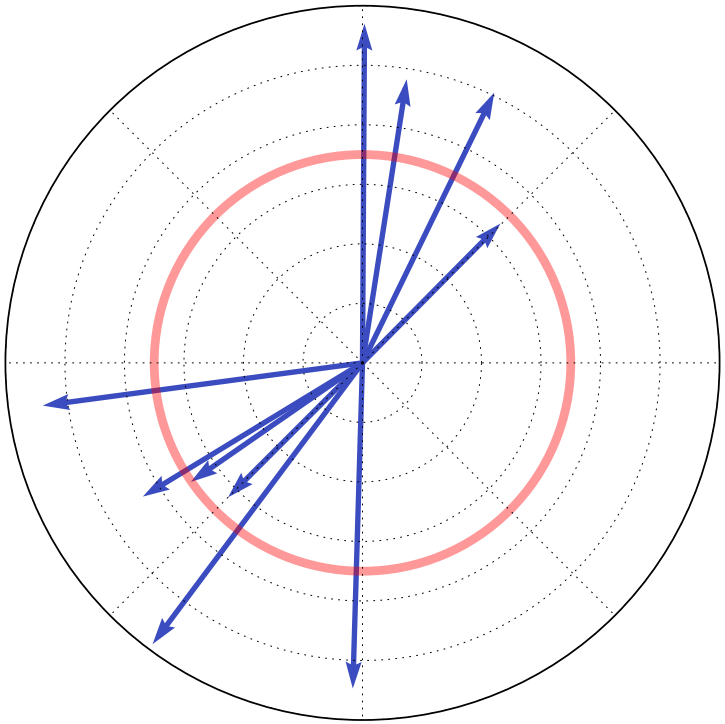
See Musco&Musco'16

Distributed sequential sampling for adaptive kernel DL

How do we compute an accurate ($\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2 \leq \varepsilon$) dictionary?

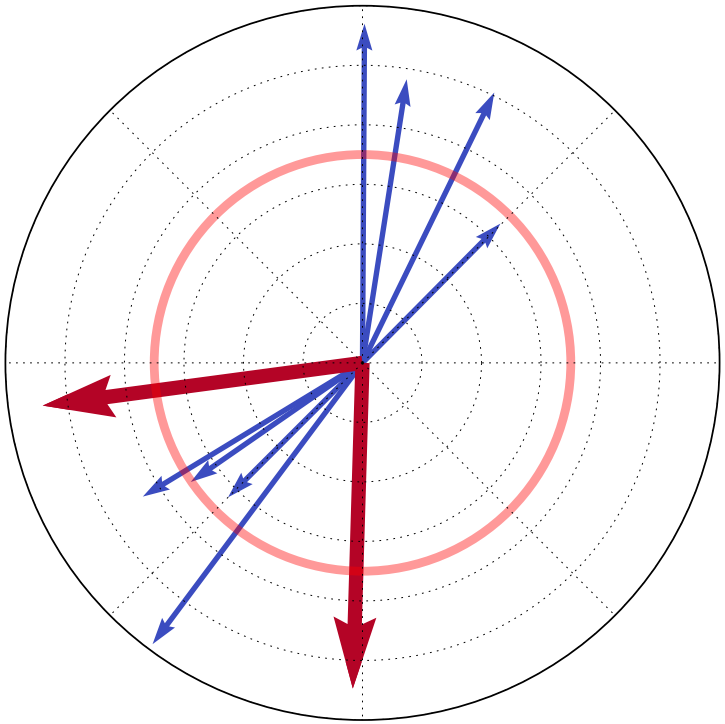
Distributed sequential sampling for adaptive kernel DL

How do we compute an accurate ($\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2 \leq \varepsilon$) dictionary?



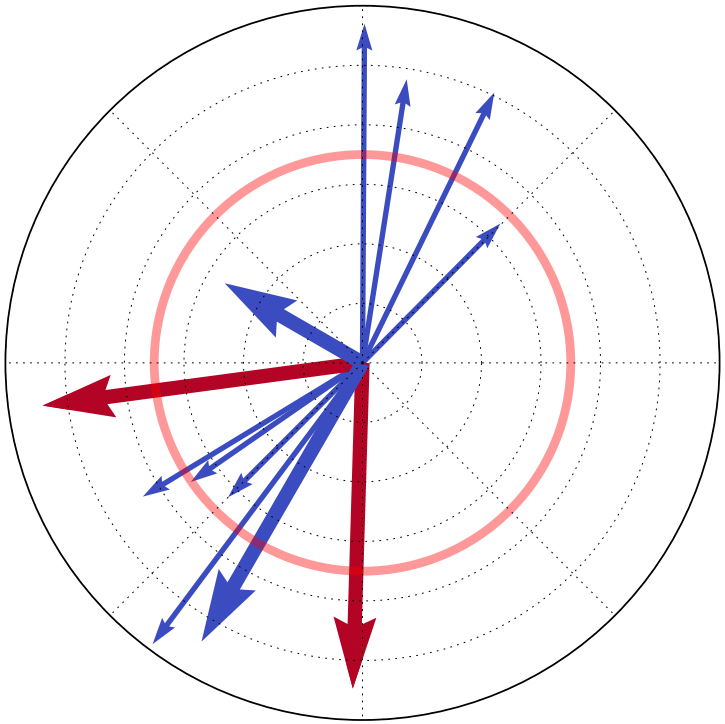
Distributed sequential sampling for adaptive kernel DL

How do we compute an accurate ($\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2 \leq \varepsilon$) dictionary?



Distributed sequential sampling for adaptive kernel DL

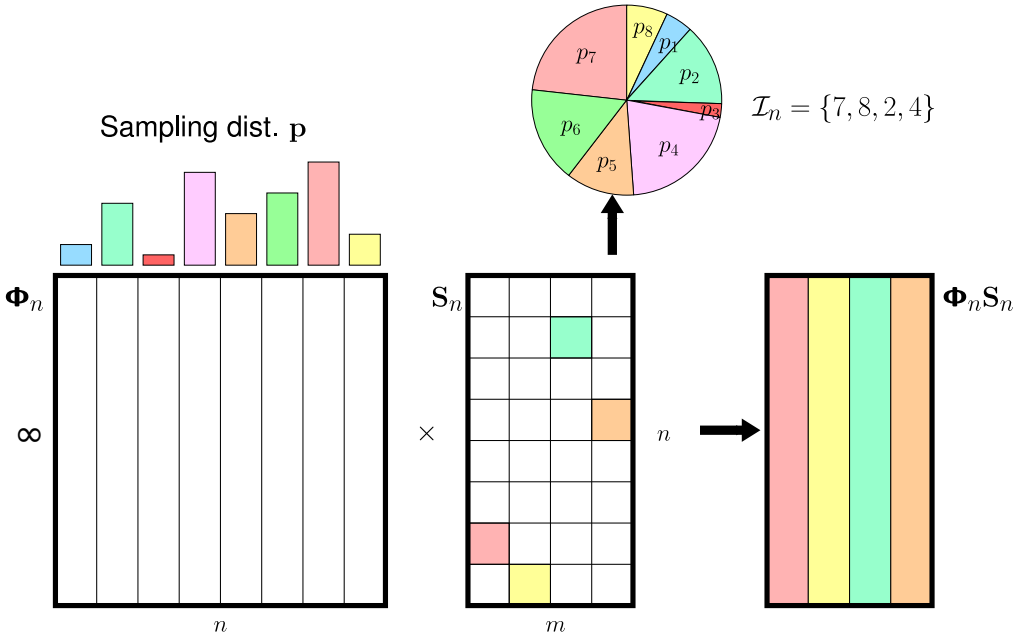
How do we compute an accurate ($\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2 \leq \varepsilon$) dictionary?



Distributed sequential sampling for adaptive kernel DL

How do we compute an accurate ($\|\mathbf{P}_n - \widetilde{\mathbf{P}}_n\|_2 \leq \varepsilon$) dictionary?

Sample m points w.p. $p_{n,i}$, add to $\mathcal{I}$ with weight $1/p_{n,i}$ (unbiased)



? How to choose the sampling distribution?

? How to choose m?

Ridge Leverage Scores and Effective Dimension

Definition

Given a kernel matrix $\mathbf{K}_n \in \mathbb{R}^{n \times n}$, define

γ-RLS
$$\tau_{n,i} = \mathbf{e}_{n,i}\mathbf{K}_n^\top(\mathbf{K}_n + \gamma \mathbf{I}_n)^{-1}\mathbf{e}_{n,i}$$
$$= \phi_i^\top(\mathbf{\Phi}_n\mathbf{\Phi}_n^\top + \gamma \mathbf{I})^{-1}\phi_i \quad (1)$$

effective dim.
$$d_{\text{eff}}\gamma_n = \sum_{i=1}^{n} \tau_{n,i} = \text{Tr}\left(\mathbf{K}_n(\mathbf{K}_n + \gamma \mathbf{I}_n)^{-1}\right) \quad (2)$$

Ridge Leverage Scores

Intuitively, RLS capture orthogonality

$$\tau_{n,i} = \mathbf{e}_{n,i}\mathbf{K}_n^\intercal(\mathbf{K}_n + \gamma\mathbf{I}_n)^{-1}\mathbf{e}_{n,i} = \phi_i^\intercal(\mathbf{\Phi}_n\mathbf{\Phi}_n^\intercal + \gamma\mathbf{I})^{-1}\phi_i$$

If all ϕ_i are orthogonal, we have

$$\tau_{n,i} = \phi_i^\intercal(\mathbf{\Phi}_n\mathbf{\Phi}_n^\intercal + \gamma\mathbf{I})^{-1}\phi_i = \phi_i^\intercal(\phi_i\phi_i^\intercal + \gamma\mathbf{I})^{-1}\phi_i = \frac{\phi_i^\intercal\phi_i}{\phi_i^\intercal\phi_i + \gamma} \sim \mathbf{1}$$

If all ϕ_i are identical (collinear), we have

$$\tau_{n,i} = \phi_i^\intercal(\mathbf{\Phi}_n\mathbf{\Phi}_n^\intercal + \gamma\mathbf{I})^{-1}\phi_i = \phi_i^\intercal(n\phi_i\phi_i^\intercal + \gamma\mathbf{I})^{-1}\phi_i = \frac{\phi_i^\intercal\phi_i}{n\phi_i^\intercal\phi_i + \gamma} \sim \frac{1}{n}$$

Ridge Leverage Scores

Intuitively, RLS capture orthogonality

$$\tau_{n,i} = \mathbf{e}_{n,i}\mathbf{K}_n^\intercal(\mathbf{K}_n + \gamma\mathbf{I}_n)^{-1}\mathbf{e}_{n,i} = \phi_i^\intercal(\mathbf{\Phi}_n\mathbf{\Phi}_n^\intercal + \gamma\mathbf{I})^{-1}\phi_i$$

If all ϕ_i are orthogonal, we have

$$\tau_{n,i} = \phi_i^\intercal(\mathbf{\Phi}_n\mathbf{\Phi}_n^\intercal + \gamma\mathbf{I})^{-1}\phi_i = \phi_i^\intercal(\phi_i\phi_i^\intercal + \gamma\mathbf{I})^{-1}\phi_i = \frac{\phi_i^\intercal\phi_i}{\phi_i^\intercal\phi_i + \gamma} \sim \mathbf{1}$$

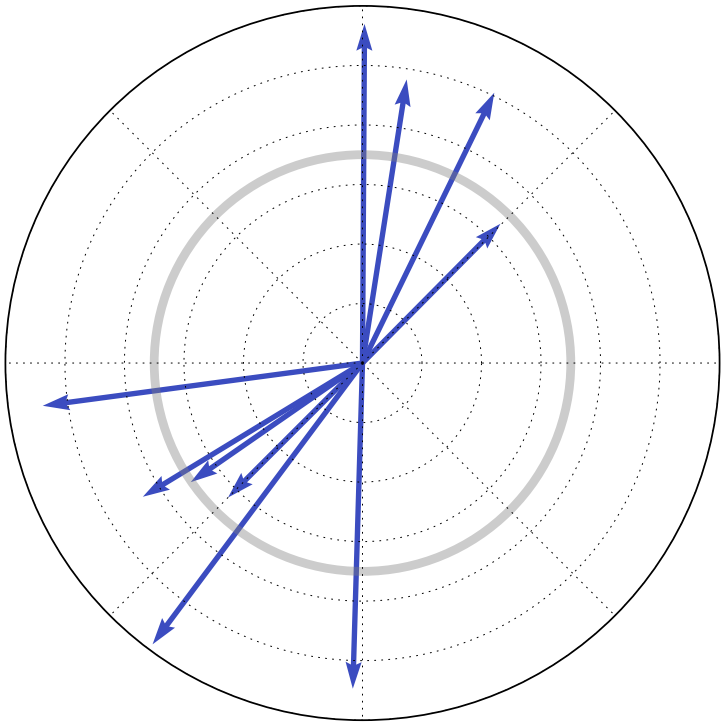
If all ϕ_i are identical (collinear), we have

$$\tau_{n,i} = \phi_i^\intercal(\mathbf{\Phi}_n\mathbf{\Phi}_n^\intercal + \gamma\mathbf{I})^{-1}\phi_i = \phi_i^\intercal(n\phi_i\phi_i^\intercal + \gamma\mathbf{I})^{-1}\phi_i = \frac{\phi_i^\intercal\phi_i}{n\phi_i^\intercal\phi_i + \gamma} \sim \frac{1}{n}$$

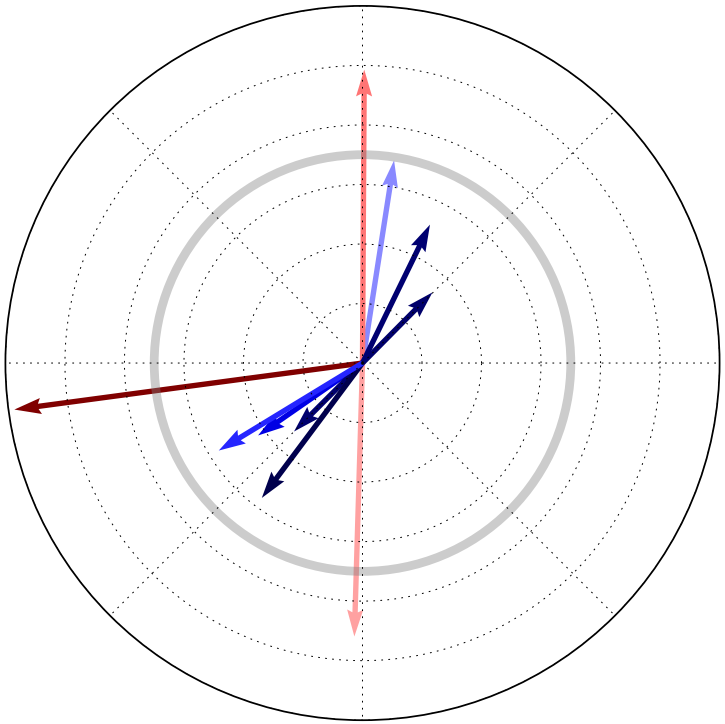
Given $\mathbf{\Phi}_{t-1}$, adding a new column to it can only reduce the RLS of columns already in $\mathbf{\Phi}_{t-1}$

$$\tau_{\mathbf{t},\mathbf{i}} \leq \tau_{\mathbf{t-1},\mathbf{i}}$$

Ridge Leverage Scores

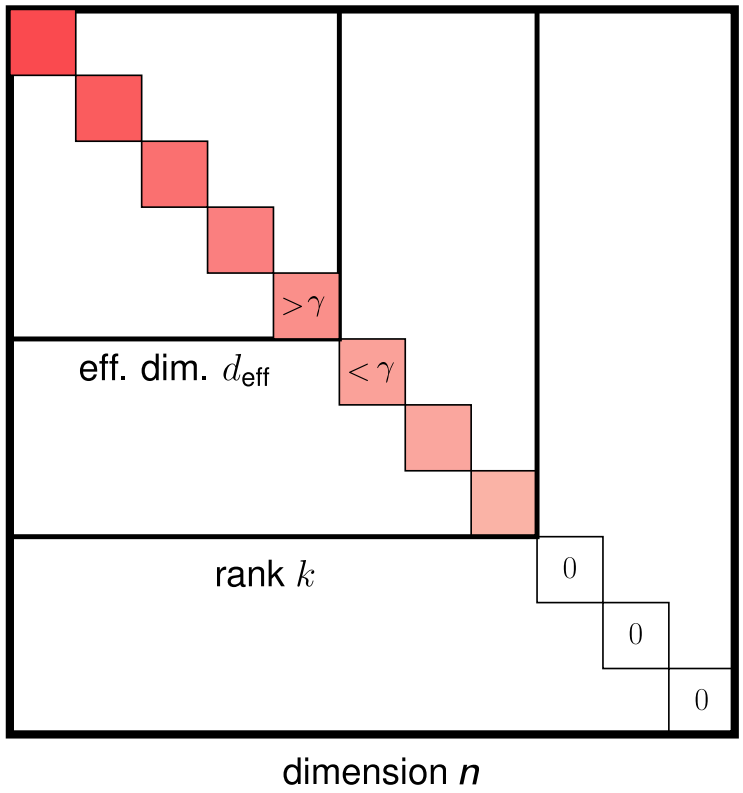


Ridge Leverage Scores



Effective Dimension

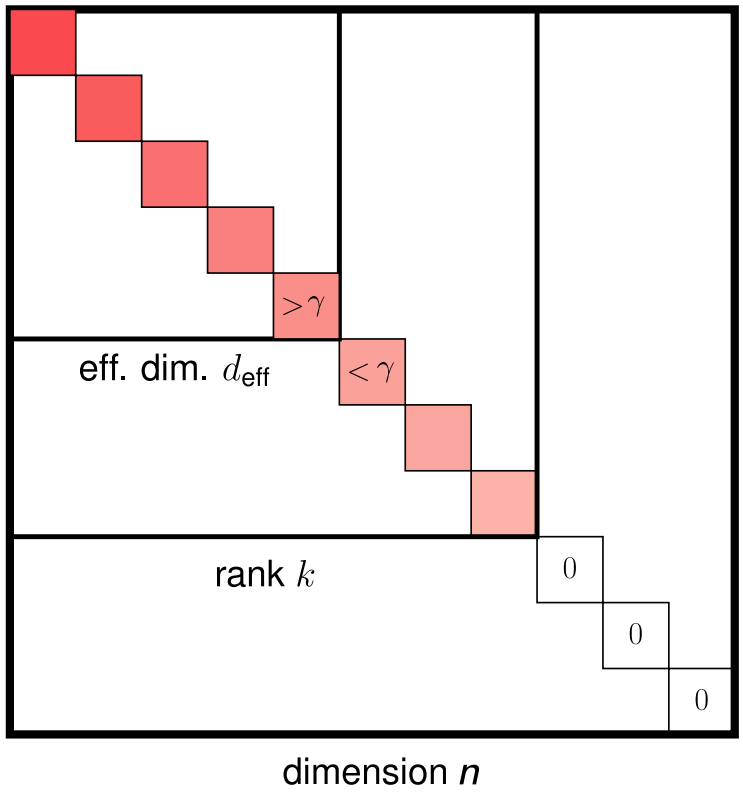
Intuitively, the effective dimension is a soft version of matrix rank



$$d_{\text{eff}}\gamma_t = \sum_{i=1}^{t} \frac{\lambda_i(\mathbf{K}_t)}{\lambda_i(\mathbf{K}_t) + \gamma} \leq \text{Rank}(\mathbf{K}_t)$$

Effective Dimension

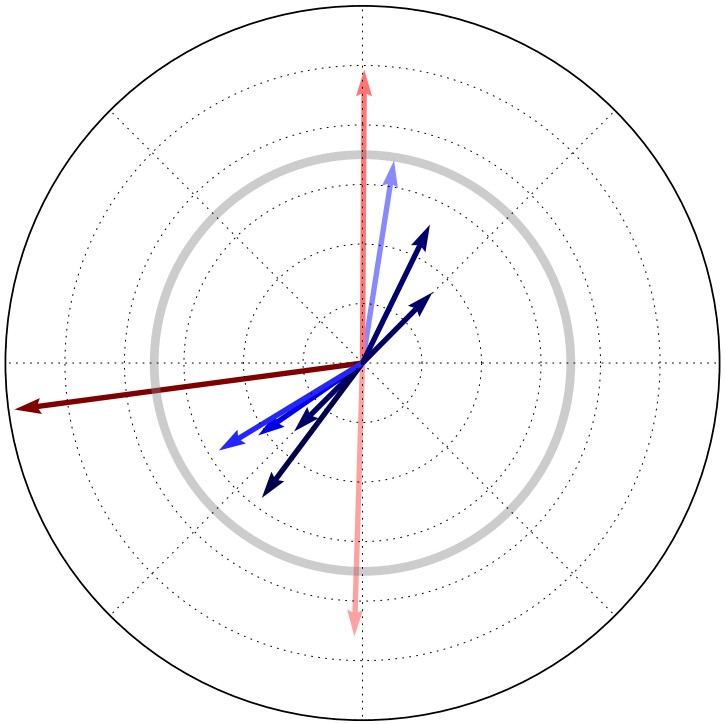
Intuitively, the effective dimension is a soft version of matrix rank



Given $d_{\text{eff}}\gamma_{t-1}$, adding a new column to $\mathbf{\Phi}_{t-1}$ can only increase $d_{\text{eff}}\gamma_t$

$$\mathbf{d_{\text{eff}}\gamma_t \geq d_{\text{eff}}\gamma_{t-1}}$$

Effective Dimension



Nystrom Sampling

Theorem (Alaoui, Mahoney, 2015)

Given γ be the Nystrom regularization, ε the accuracy, δ the confidence.
If the dictionary $\mathcal{I}_n$ is computed using the sampling distribution $p_{n,i} \propto \tau_{n,i}$ and using at least m columns

$$m \geq \left(\frac{2\mathbf{d}_{eff}\gamma_{\mathbf{n}}}{\varepsilon^2} \right) \log \left(\frac{n}{\delta} \right),$$

then with probability $1 - \delta$

$$\|\mathbf{P_n} - \widetilde{\mathbf{P}}_{\mathbf{n}}\|_2 \leq \varepsilon$$

Nystrom Sampling

Theorem (Alaoui, Mahoney, 2015)

Given γ be the Nystrom regularization, ε the accuracy, δ the confidence.
If the dictionary $\mathcal{I}_n$ is computed using the sampling distribution $p_{n,i} \propto \tau_{n,i}$ and using at least m columns

$$m \geq \left(\frac{2\mathbf{d}_{eff}\gamma_{\mathbf{n}}}{\varepsilon^2} \right) \log \left(\frac{n}{\delta} \right),$$

then with probability $1 - \delta$

$$\|\mathbf{P_n} - \widetilde{\mathbf{P}}_{\mathbf{n}}\|_2 \leq \varepsilon$$

Done!

Nystrom Sampling

Theorem (Alaoui, Mahoney, 2015)

Given γ be the Nystrom regularization, ε the accuracy, δ the confidence.
If the dictionary $\mathcal{I}_n$ is computed using the sampling distribution $p_{n,i} \propto \tau_{n,i}$ and using at least m columns

$$m \geq \left(\frac{2\mathbf{d}_{eff}\gamma_{\mathbf{n}}}{\varepsilon^2}\right) \log\left(\frac{n}{\delta}\right),$$

then with probability $1-\delta$

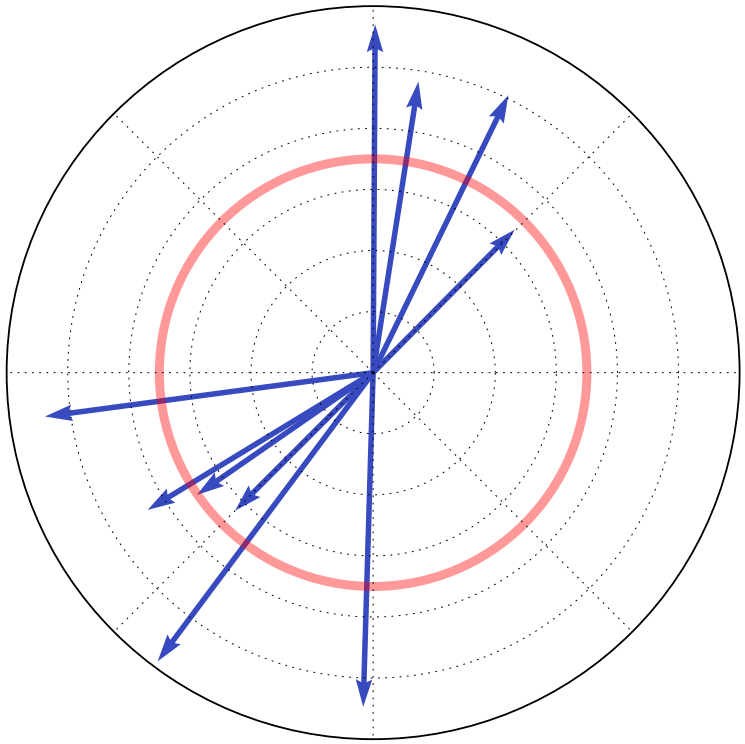
$$\|\mathbf{P_n} - \widetilde{\mathbf{P}}_{\mathbf{n}}\|_2 \leq \varepsilon$$

Done!

If someone gave us the RLS

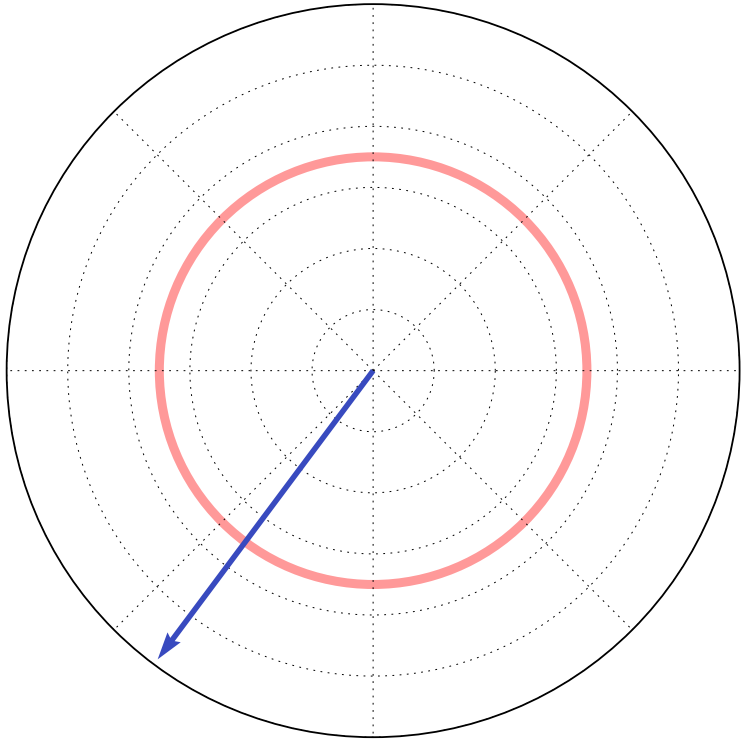
Computing $\tau_{n,i} = \mathbf{e}_{n,i}\mathbf{K}_n^{\mathsf{T}}(\mathbf{K}_n + \gamma\mathbf{I}_n)^{-1}\mathbf{e}_{n,i}$ also requires storing and inverting the full $\mathbf{K}_n$

Distributed sequential sampling for adaptive kernel DL



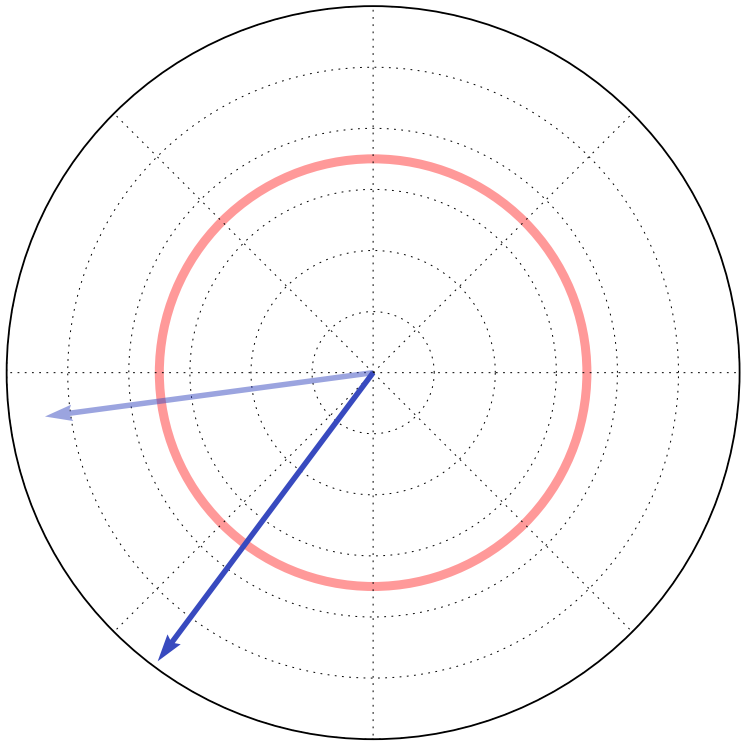
Distributed sequential sampling for adaptive kernel DL

$\widetilde{p}_{1,1} = \widetilde{\tau}_{1,1}$ 



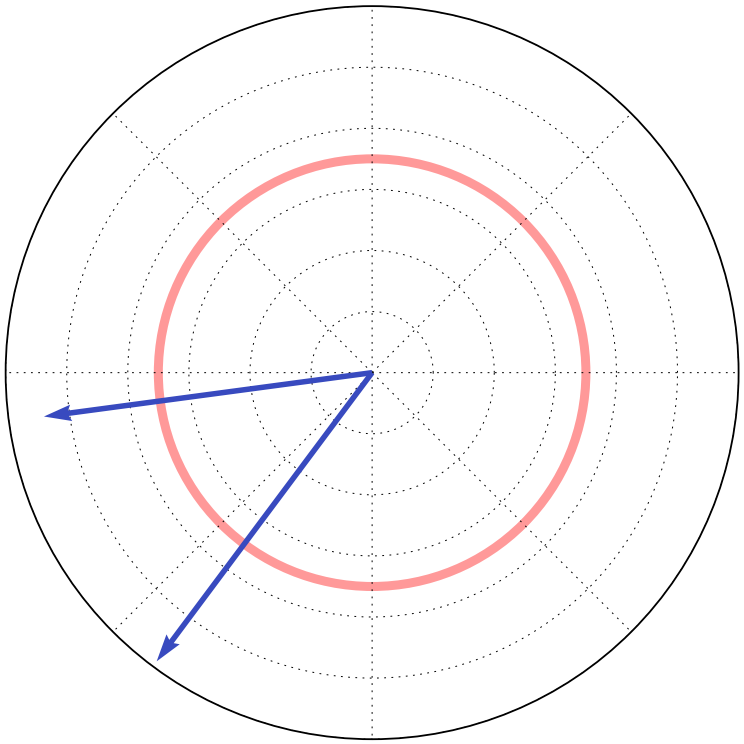
Distributed sequential sampling for adaptive kernel DL

$$\widetilde{p}_{2,1} = \widetilde{\tau}_{2,1}/\widetilde{\tau}_{1,1}$$
$$\widetilde{p}_{2,2} = \widetilde{\tau}_{2,2}$$



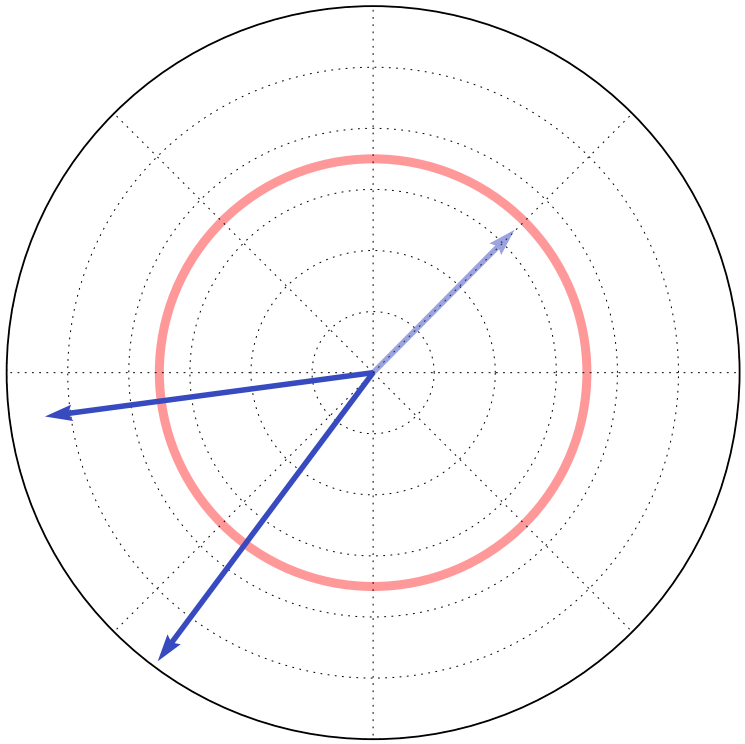
Distributed sequential sampling for adaptive kernel DL

$$\widetilde{p}_{2,1} = \widetilde{\tau}_{2,1}/\widetilde{\tau}_{1,1}$$
$$\widetilde{p}_{2,2} = \widetilde{\tau}_{2,2}$$



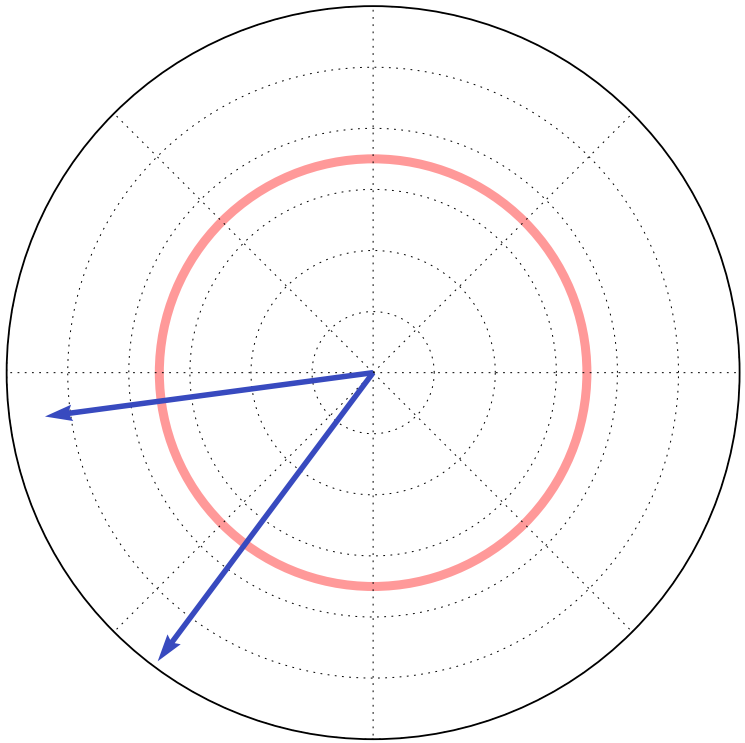
Distributed sequential sampling for adaptive kernel DL

$$\widetilde{p}_{3,1} = \widetilde{\tau}_{3,1}/\widetilde{\tau}_{2,1}$$
$$\widetilde{p}_{3,2} = \widetilde{\tau}_{3,2}/\widetilde{\tau}_{2,2}$$
$$\widetilde{p}_{3,3} = \widetilde{\tau}_{3,3}$$



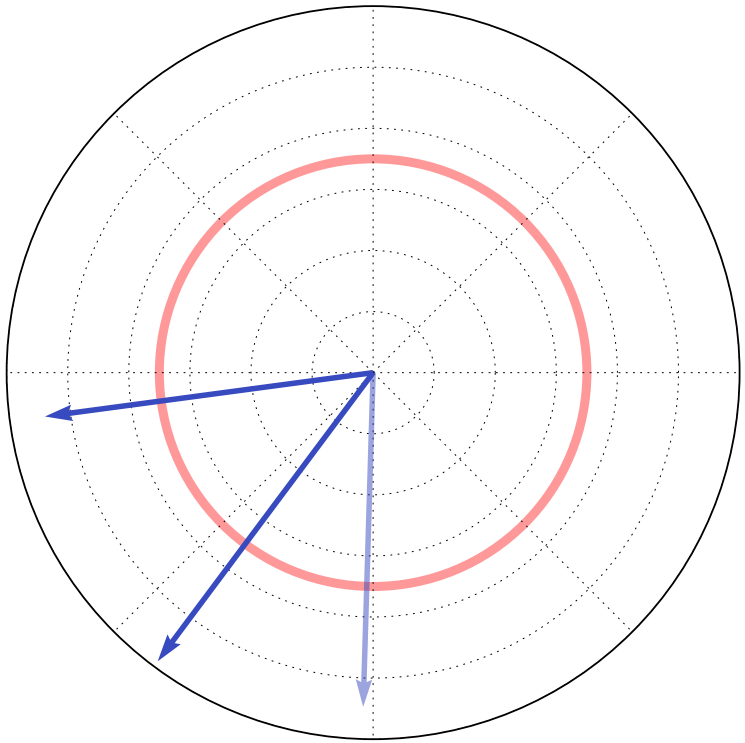
Distributed sequential sampling for adaptive kernel DL

$$\widetilde{p}_{3,1} = \widetilde{\tau}_{3,1}/\widetilde{\tau}_{2,1}$$
$$\widetilde{p}_{3,2} = \widetilde{\tau}_{3,2}/\widetilde{\tau}_{2,2}$$
$$\widetilde{p}_{3,3} = \widetilde{\tau}_{3,3}$$



Distributed sequential sampling for adaptive kernel DL

$\widetilde{p}_{4,1} = \widetilde{\tau}_{4,1}/\widetilde{\tau}_{3,1}$
$\widetilde{p}_{4,2} = \widetilde{\tau}_{4,2}/\widetilde{\tau}_{3,2}$
$\widetilde{p}_{3,3}$
$\widetilde{p}_{4,4} = \widetilde{\tau}_{4,4}$



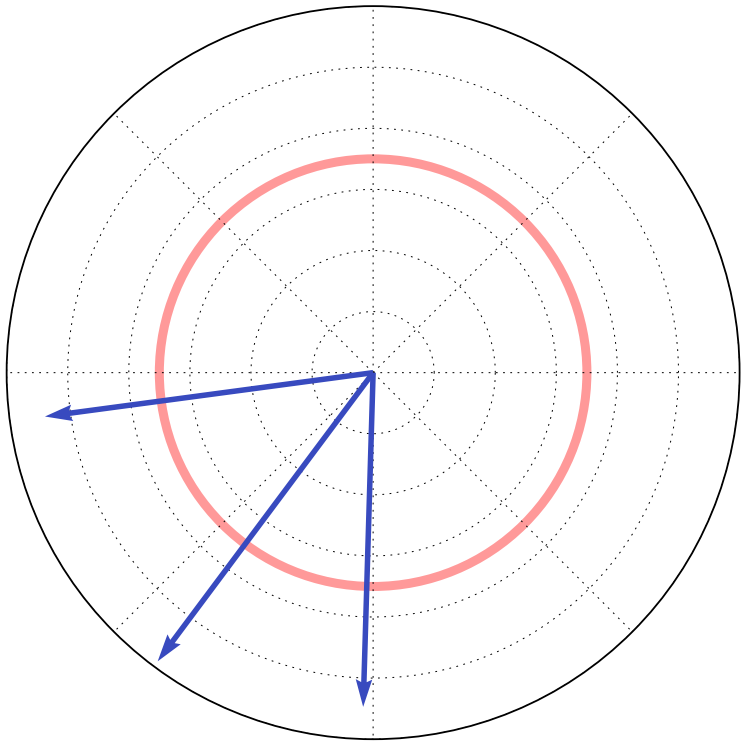
Distributed sequential sampling for adaptive kernel DL

$\widetilde{p}_{4,1} = \widetilde{\tau}_{4,1}/\widetilde{\tau}_{3,1}$

$\widetilde{p}_{4,2} = \widetilde{\tau}_{4,2}/\widetilde{\tau}_{3,2}$

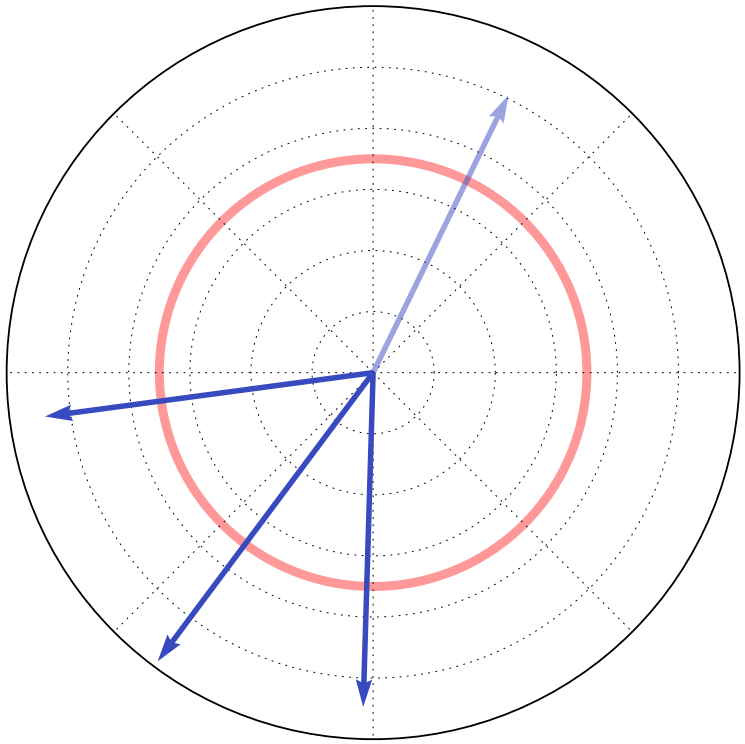
$\widetilde{p}_{3,3}$

$\widetilde{p}_{4,4} = \widetilde{\tau}_{4,4}$



Distributed sequential sampling for adaptive kernel DL

$\widetilde{p}_{5,1} = \widetilde{\tau}_{5,1}/\widetilde{\tau}_{4,1}$
$\widetilde{p}_{5,2} = \widetilde{\tau}_{5,2}/\widetilde{\tau}_{4,2}$
$\widetilde{p}_{3,3}$ 
$\widetilde{p}_{5,4} = \widetilde{\tau}_{5,4}/\widetilde{\tau}_{4,4}$
$\widetilde{p}_{5,5} = \widetilde{\tau}_{5,5}$



Distributed sequential sampling for adaptive kernel DL

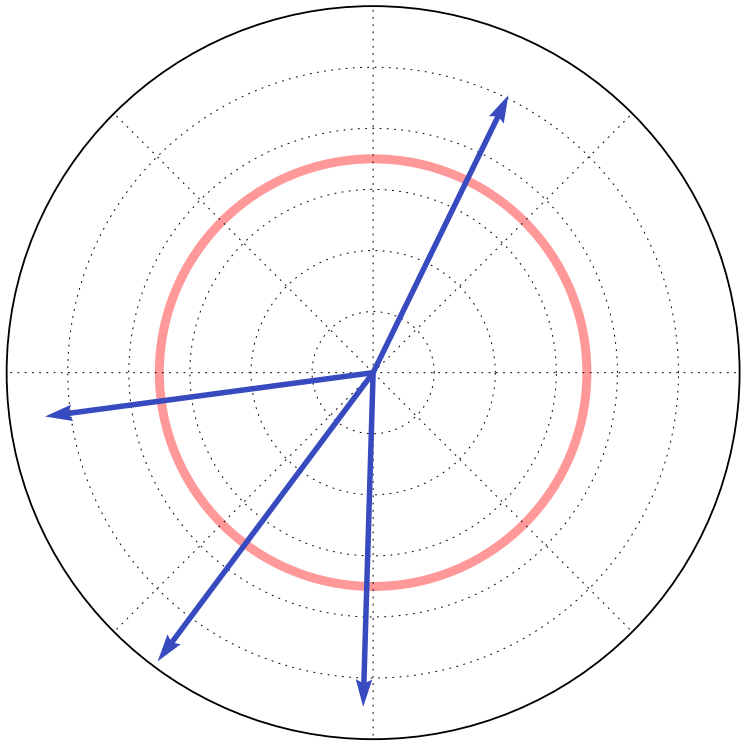
$\widetilde{p}_{5,1} = \widetilde{\tau}_{5,1}/\widetilde{\tau}_{4,1}$

$\widetilde{p}_{5,2} = \widetilde{\tau}_{5,2}/\widetilde{\tau}_{4,2}$

$\widetilde{p}_{3,3}$

$\widetilde{p}_{5,4} = \widetilde{\tau}_{5,4}/\widetilde{\tau}_{4,4}$

$\widetilde{p}_{5,5} = \widetilde{\tau}_{5,5}$



Distributed sequential sampling for adaptive kernel DL

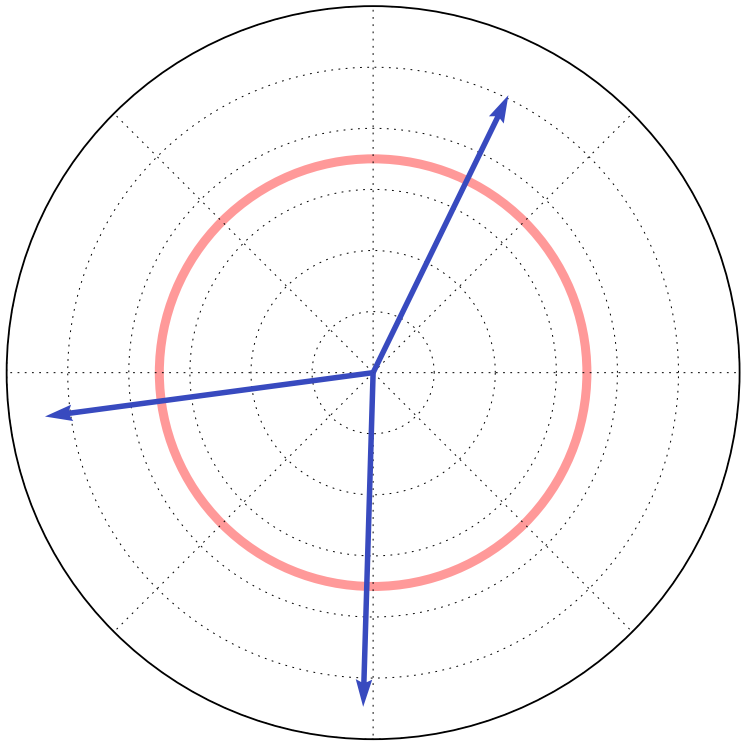
$\widetilde{p}_{5,1} = \widetilde{\tau}_{5,1}/\widetilde{\tau}_{4,1}$

$\widetilde{p}_{5,2} = \widetilde{\tau}_{5,2}/\widetilde{\tau}_{4,2}$

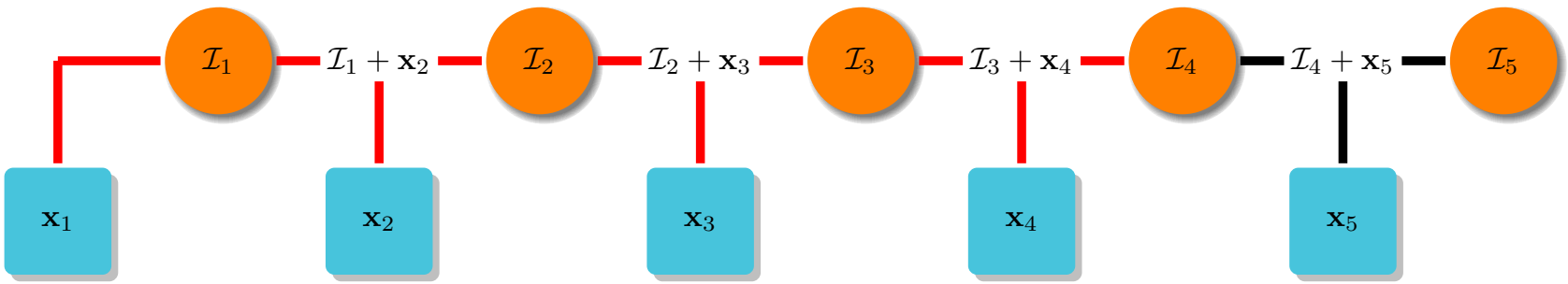
$\widetilde{p}_{3,3}$

$\widetilde{p}_{5,4} = \widetilde{\tau}_{5,4}/\widetilde{\tau}_{4,4}$

$\widetilde{p}_{5,5} = \widetilde{\tau}_{5,5}$



Distributed sequential sampling for adaptive kernel DL



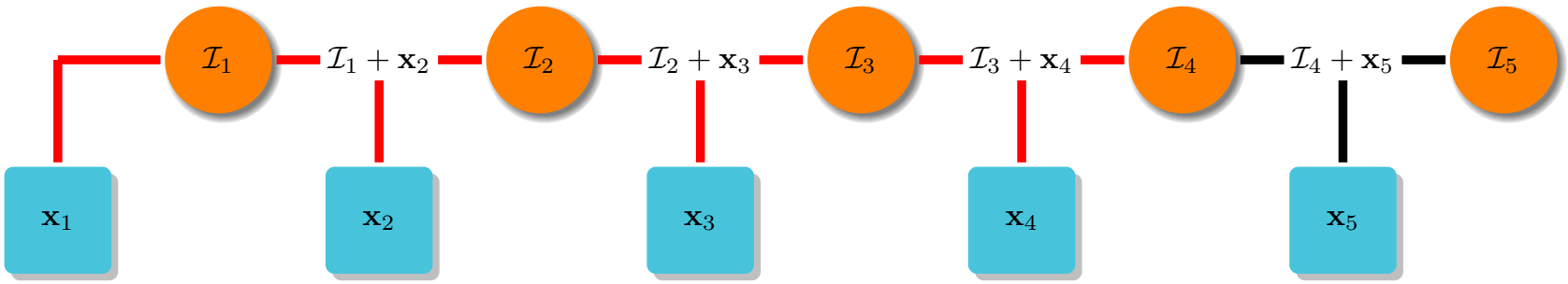
Distributed sequential sampling for adaptive kernel DL

Instead of sampling from multinomial consider the sampling process

$$q_{i,i} \sim \mathcal{B}(\widetilde{p}_{i,i}, \overline{q})$$
$$q_{t,i} \sim \mathcal{B}(\widetilde{p}_{t,i}/\widetilde{p}_{t-1,i}, q_{t-1,i})$$

Similar to importance sampling. If the $\widetilde{p}_{t,i}$ were fixed in advance

$$\begin{aligned}\mathbb{P}(z_{t,i,j} = 1) &= \mathbb{P}(\mathcal{B}(\widetilde{p}_{t,i}/\widetilde{p}_{t-1,i}) = 1) z_{t-1,i,j} \\ &= \mathbb{P}(\mathcal{B}(\widetilde{p}_{t,i}/\widetilde{p}_{t-1,i}) = 1)\mathbb{P}(\mathcal{B}(\widetilde{p}_{t-1,i}/\widetilde{p}_{t-2,i}) = 1) z_{t-2,i,j} \\ &= \frac{\widetilde{p}_{t,i}}{\widetilde{p}_{t-1,i}} \frac{\widetilde{p}_{t-1,i}}{\widetilde{p}_{t-2,i}} \cdots \frac{\widetilde{p}_{i+1,i}}{\widetilde{p}_{i,i}} \frac{\widetilde{p}_{i,i}}{1} = \widetilde{p}_{t,i}\end{aligned}$$



Estimating RLS

$$\widetilde{\tau}_{t,i} = \frac{1+\varepsilon}{\alpha\gamma}\left(k_{i,i} - \mathbf{k_{t,i}}\overline{\mathbf{S}}\left(\overline{\mathbf{S}}^\top \mathbf{K_t}\overline{\mathbf{S}} + \gamma\mathbf{I}\right)^{-1}\overline{\mathbf{S}}^\top \mathbf{k_{t,i}}\right),$$

- $\widetilde{\tau}_{t,i} = \mathbf{e}_i^\top \widetilde{\mathbf{K}}_\mathbf{t}(\widetilde{\mathbf{K}}_\mathbf{t} + \gamma\mathbf{I})^{-1}\mathbf{e}_i$ would fail
- Instead, approximate $\tau_{t,i}$ directly in $\mathcal{H}$, and then reformulate using kernel trick $\widetilde{\tau}_{\mathbf{t},\mathbf{i}} = \phi_\mathbf{i}^\top(\mathbf{\Phi}\overline{\mathbf{S}}\overline{\mathbf{S}}^\top\mathbf{\Phi}^\top + \gamma\mathbf{I})^{-1}\phi_\mathbf{i}$
- $\widetilde{\tau}_{t,i}$ can be computed in $\mathcal{O}(|\mathcal{I}_\mathbf{t}|^2)$ space and $\mathcal{O}(|\mathcal{I}_\mathbf{t}|^3)$ time
  - ↳ independent from t
- $\widetilde{\tau}_{t,i}$ for $i \in \mathcal{I}_t$ can be computed using only samples contained in $\mathcal{I}_t$.

Estimating RLS

Lemma Assume that the dictionary $\mathcal{I}_{t-1}$ is accurate, and let $\overline{\mathbf{S}}_t$ be constructed by adding $(1, \phi_t)$ to $\mathcal{I}_{t-1}$. Then, denoting $\alpha = (1+\varepsilon)/(1-\varepsilon)$, for all i such that $i \in \{\mathcal{I}_{t-1} \cup \{t\}\}$,

$$\widetilde{\tau}_{t,i} = \frac{1+\varepsilon}{\alpha\gamma}\left(k_{i,i} - \mathbf{k}_{t,i}\overline{\mathbf{S}}\left(\overline{\mathbf{S}}^{\mathsf{T}}\mathbf{K}_t\overline{\mathbf{S}} + \gamma\mathbf{I}\right)^{-1}\overline{\mathbf{S}}^{\mathsf{T}}\mathbf{k}_{t,i}\right), \tag{3}$$

is an α-approximation of the RLS $\tau_{t,i}$, that is $\tau_{\mathbf{t},\mathbf{i}}(\gamma)/\alpha \leq \widetilde{\tau}_{\mathbf{t},\mathbf{i}} \leq \tau_{\mathbf{t},\mathbf{i}}(\gamma)$.

SQUEAK https://team.inria.fr/sequel/software/

Dictionary $\mathcal{I}_t = \{(j, \phi_j, q_{t,j}, \widetilde{p}_{t,j})\}$, weights $w_i = \frac{q_{t,j}}{\widetilde{p}_{t,j}\overline{q}}$

Input: $\mathcal{D}$, regularization $\gamma, \overline{q}$, ε, **Output**: $\mathcal{I}_n$

Initialize $\mathcal{I}_0$ as empty, $\widetilde{p}_{1,0} = 1$
for $t = 1, \ldots, n$ **do**
  Receive new sample $\mathbf{x}_t$
  Compute α-app. RLS $\{\widetilde{\tau}_{t,i} : i \in \mathcal{I}_{t-1} \cup \{t\}\}$, using $\mathcal{I}_{t-1}$, $\mathbf{x}$, and Eq. 3
  Set $\widetilde{\mathbf{p}}_{\mathbf{t},\mathbf{i}} = \min\left\{\widetilde{\tau}_{\mathbf{t},\mathbf{i}},\ \widetilde{\mathbf{p}}_{\mathbf{t}-\mathbf{1},\mathbf{i}}\right\}$
  Initialize $\mathcal{I}_t = \emptyset$
  for all $j \in \{1, \ldots, t-1\}$ **do**
    if $q_{t-1,j} \neq 0$ **then**
      $\mathbf{q}_{\mathbf{t},\mathbf{j}} \sim \mathcal{B}(\widetilde{\mathbf{p}}_{\mathbf{t},\mathbf{j}}/\widetilde{\mathbf{p}}_{\mathbf{t}-\mathbf{1},\mathbf{j}}, \mathbf{q}_{\mathbf{t}-\mathbf{1},\mathbf{j}})$
      Add $(j, \phi_j, q_{t,j}, \widetilde{p}_{t,j})$ to $\mathcal{I}_t$.
    end if
  end for
  $\mathbf{q}_{\mathbf{t},\mathbf{t}} \sim \mathcal{B}(\widetilde{\mathbf{p}}_{\mathbf{t},\mathbf{t}}, \overline{\mathbf{q}})$
  Add $q_{t,t}$ copies of $(t, \phi_t, q_{t,t}, \widetilde{p}_{t,t})$ to $\mathcal{I}_t$
end for

SHRINK

EXPAND

DICT-UPDATE

SQUEAK

Theorem

Let $\alpha = (\frac{1+\varepsilon}{1-\varepsilon})$ *and* $\gamma > 1$. *For any* $0 \leq \varepsilon \leq 1$, *and* $0 \leq \delta \leq 1$, *if we run* SQUEAK *with* $\overline{\mathbf{q}} = \mathcal{O}(\frac{\alpha}{\varepsilon^2}\log(\frac{\mathbf{n}}{\delta}))$, *then w.p.* $1-\delta$, *for all* $t \in [n]$

(1) $\|\mathbf{P_t} - \widetilde{\mathbf{P}}_\mathbf{t}\|_2 \leq \varepsilon$.

(2) $|\mathcal{I}_\mathbf{t}| = \sum_i q_{t,i} \leq \mathcal{O}(\overline{q} d_{eff}\gamma_t) \leq \mathcal{O}(\frac{\alpha}{\varepsilon^2}\mathbf{d}_{eff}\gamma_\mathbf{n}\log(\frac{\mathbf{n}}{\delta}))$.

SQUEAK

Theorem

Let $\alpha = (\frac{1+\varepsilon}{1-\varepsilon})$ *and* $\gamma > 1$. *For any* $0 \leq \varepsilon \leq 1$, *and* $0 \leq \delta \leq 1$, *if we run* SQUEAK *with* $\overline{\mathbf{q}} = \mathcal{O}(\frac{\alpha}{\varepsilon^2}\log(\frac{\mathbf{n}}{\delta}))$, *then w.p.* $1-\delta$, *for all* $t \in [n]$

(1) $\|\mathbf{P_t} - \widetilde{\mathbf{P}}_\mathbf{t}\|_2 \leq \varepsilon$.

(2) $|\mathcal{I}_\mathbf{t}| = \sum_i q_{t,i} \leq \mathcal{O}(\overline{q} d_{eff}\gamma_t) \leq \mathcal{O}(\frac{\alpha}{\varepsilon^2}\mathbf{d}_{eff}\gamma_\mathbf{n}\log(\frac{\mathbf{n}}{\delta}))$.

- Accuracy and space/time guarantees
- Anytime risk guarantees
- In worst case, no space gain (stores full $\mathbf{K}_n$)
- In worst case, no space overhead (stores full $\mathbf{K}_n$)
- RLS estimator not incremental, not easy because of changing weights
- Unnormalized $\widetilde{p}_{t,i}$, no need for appr. $d_{\text{eff}}\gamma_t$

SQUEAK

Theorem

Let $\alpha = (\frac{1+\varepsilon}{1-\varepsilon})$ *and* $\gamma > 1$. *For any* $0 \leq \varepsilon \leq 1$, *and* $0 \leq \delta \leq 1$, *if we run* SQUEAK *with* $\overline{\mathbf{q}} = \mathcal{O}(\frac{\alpha}{\varepsilon^2}\log(\frac{\mathbf{n}}{\delta}))$, *then w.p.* $1-\delta$, *for all* $t \in [n]$

(1) $\|\mathbf{P_t} - \widetilde{\mathbf{P}}_\mathbf{t}\|_2 \leq \varepsilon$.

(2) $|\mathcal{I}_\mathbf{t}| = \sum_i q_{t,i} \leq \mathcal{O}(\overline{q} d_{eff} \gamma_t) \leq \mathcal{O}(\frac{\alpha}{\varepsilon^2}\mathbf{d}_{eff}\gamma_\mathbf{n}\log(\frac{\mathbf{n}}{\delta}))$.

- Only need to compute $\widetilde{\tau}_{t,i}$ if $i \in \mathcal{I}_t$, never recompute after dropping
  - Never construct the whole $\mathbf{K}_n$
    - subquadratic runtime ~~$\mathcal{O}(\mathbf{n}^3)$~~ $\Rightarrow \mathcal{O}(\mathbf{n}|\mathcal{I}_\mathbf{n}|^3) \leq \widetilde{\mathcal{O}}(\mathbf{n}\mathbf{d}_{\text{eff}}\gamma_\mathbf{n}^3)$
- Store points directly in the dictionary
  - $\widetilde{\mathcal{O}}(\mathbf{d}_{\text{eff}}\gamma_\mathbf{n}^2 + \mathbf{d}_{\text{eff}}\gamma_\mathbf{n}\mathbf{d})$ space constant in n
    - single pass over the dataset (streaming)

Proof sketch

Need to bound

$$\mathbb{P}\left(\exists t \in \{1, \dots, n\} : \|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\|_2 \geq \varepsilon \cup |\mathcal{I}_t| \geq 3\overline{q} d_{\text{eff}} \gamma_t\right)$$

Proof sketch

Need to bound

$$\mathbb{P}\left(\exists t \in \{1,\dots,n\} : \|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\|_2 \geq \varepsilon \cup |\mathcal{I}_t| \geq 3\overline{q} d_{\text{eff}} \gamma_t\right)$$

After a union bound

$$\sum_{t=1}^{n} \mathbb{P}\left(\|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\|_2 \geq \varepsilon\right)$$
$$+ \sum_{t=1}^{n} \mathbb{P}\left(|\mathcal{I}_t| \geq 3\overline{q} d_{\text{eff}} \gamma_t \cap \left\{\forall t' \in \{1,\dots,t\} : \|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\|_2 \leq \varepsilon\right\}\right)$$

Proof sketch

We start by bounding $\mathbb{P}\left(\|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\|_2 \geq \varepsilon\right)$.

Indicator of keeping copy j of sample i at step s

$$z_{s,i,j} = \mathbb{I}\left\{\text{coin-flip w.p. } \frac{\widetilde{p}_{s,i}}{\widetilde{p}_{s-1,i}}\right\} z_{s-1,i,j}, \qquad \sum_j z_{s,i,j} = q_{s,i}$$

Proof sketch

We start by bounding $\mathbb{P}\left(\|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\|_2 \geq \varepsilon\right)$.

Indicator of keeping copy j of sample i at step s

$$z_{s,i,j} = \mathbb{I}\left\{\text{coin-flip w.p. } \frac{\widetilde{p}_{s,i}}{\widetilde{p}_{s-1,i}}\right\} z_{s-1,i,j}, \qquad \sum_j z_{s,i,j} = q_{s,i}$$

Random object

$$\mathbf{Y}_t = \mathbf{P}_t - \widetilde{\mathbf{P}}_t = \frac{1}{\overline{q}} \sum_{i=1}^{t} \sum_{j=1}^{\overline{q}} \left(1 - \frac{z_{t,i,j}}{\widetilde{p}_{t,i}}\right) \mathbf{v}_i \mathbf{v}_i^\top$$

Proof sketch

We start by bounding $\mathbb{P}\left(\|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\|_2 \geq \varepsilon\right)$.

Indicator of keeping copy j of sample i at step s

$$z_{s,i,j} = \mathbb{I}\left\{\text{coin-flip w.p. } \frac{\widetilde{p}_{s,i}}{\widetilde{p}_{s-1,i}}\right\} z_{s-1,i,j}, \qquad \sum_j z_{s,i,j} = q_{s,i}$$

Random object

$$\mathbf{Y}_t = \mathbf{P}_t - \widetilde{\mathbf{P}}_t = \frac{1}{\overline{q}} \sum_{i=1}^{t} \sum_{j=1}^{\overline{q}} \left(1 - \frac{z_{t,i,j}}{\widetilde{p}_{t,i}}\right) \mathbf{v}_i \mathbf{v}_i^\top$$

Cannot use concentrations for independent r.v., because $z_{t,i,j}$ and $z_{t,i',j'}$ are both dependent on $z_{t-1,i'',j''}$ through the estimates.

Proof sketch

We can use variants of Bernstein's inequality for matrix martingales
↳need deterministic bound on variance

Variance $\mathbf{W}$ scales (roughly) with $\|\mathbf{W}\|_2^2 \sim \max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}} \right\}$

Since $\widetilde{p}_{1,i} \geq \widetilde{p}_{2,i} \geq \cdots \geq \widetilde{p}_{t-1,i} \geq \widetilde{p}_{t,i} \geq p_{t,i}/\alpha \geq 1/(t\alpha)$ we have

$$\max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}} \right\} = \max_{s \in [t-1]: z_{s,i,j}=1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}} \right\} \leq \frac{\alpha}{p_{t,i}} \leq \alpha t$$

This looks too pessimistic. When $\frac{1}{\widetilde{p}_{s,i}}$ is large, $z_{s,i,j}$ should be zero.

Proof sketch

$\max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}^2} \right\}$ complex random variable

↳ maximum of dependent variables

Proof sketch

$\max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}^2} \right\}$ complex random variable

↳ maximum of dependent variables

Moreover $\max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}^2} \right\}$ depends on $\max_{s=0}^{t-1} \left\{ \frac{z_{s,i',j'}}{\widetilde{p}_{s,i'}^2} \right\}$

Proof sketch

$\max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}^2} \right\}$ complex random variable

↳ maximum of dependent variables

Moreover $\max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}^2} \right\}$ depends on $\max_{s=0}^{t-1} \left\{ \frac{z_{s,i',j'}}{\widetilde{p}_{s,i'}^2} \right\}$

We will find another set of dominating r.v. $1/w_{i,j}$, indep. from each other

Proof sketch

$\max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}^2} \right\}$ complex random variable

↳ maximum of dependent variables

Moreover $\max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}^2} \right\}$ depends on $\max_{s=0}^{t-1} \left\{ \frac{z_{s,i',j'}}{\widetilde{p}_{s,i'}^2} \right\}$

We will find another set of dominating r.v. $1/w_{i,j}$, indep. from each other

Random variable A stochastically dominates random variable B, if for all values a the two equivalent conditions are verified

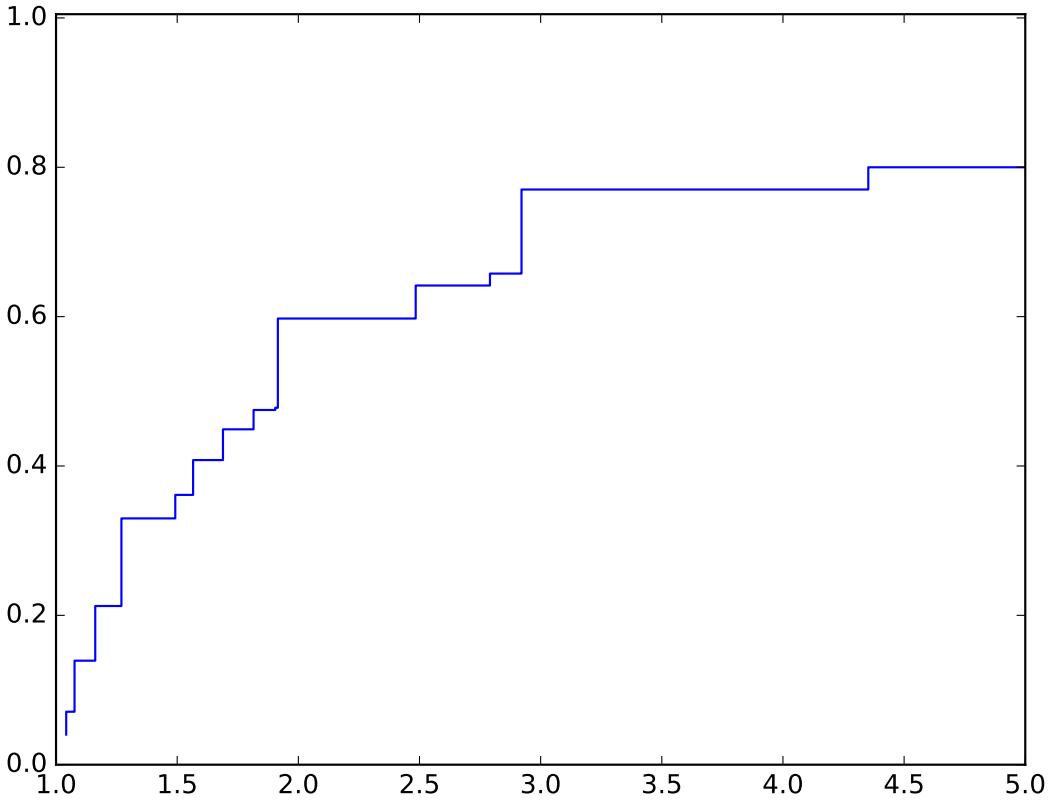
$$\mathbb{P}(A \geq a) \geq \mathbb{P}(B \geq a) \Leftrightarrow \mathbb{P}(A \leq a) \leq \mathbb{P}(B \leq a).$$

Proof sketch

Imagine the sequence $\widetilde{p}_{s,i}$ was fixed in advance. I can compute exactly the distribution of all $z_{s,i,j}$.

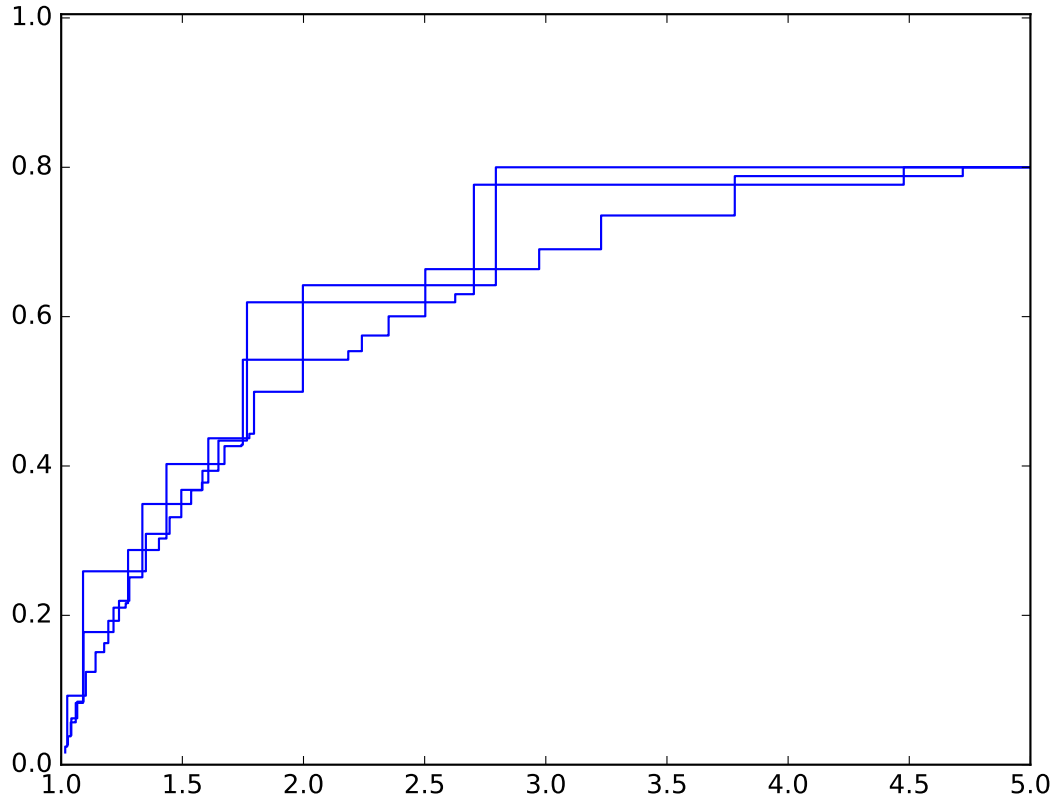
Proof sketch

Imagine the sequence $\widetilde{p}_{s,i}$ was fixed in advance. I can compute exactly the distribution of all $z_{s,i,j}$.



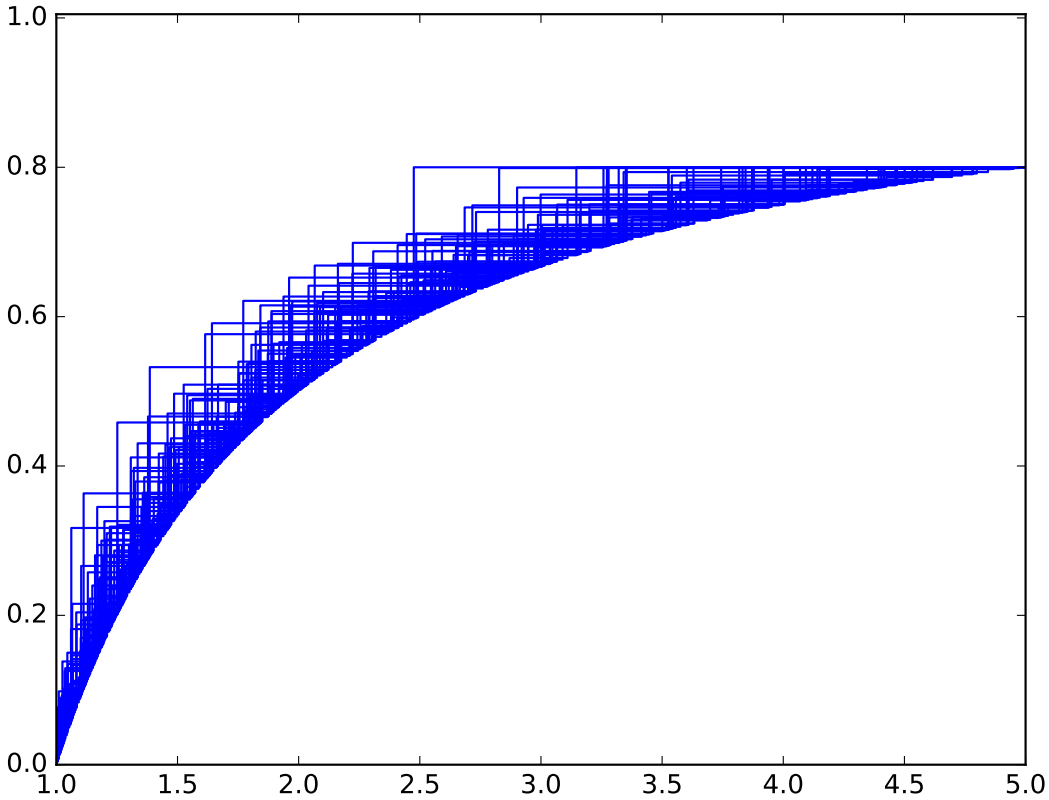
Proof sketch

Imagine the sequence $\widetilde{p}_{s,i}$ was fixed in advance. I can compute exactly the distribution of all $z_{s,i,j}$.



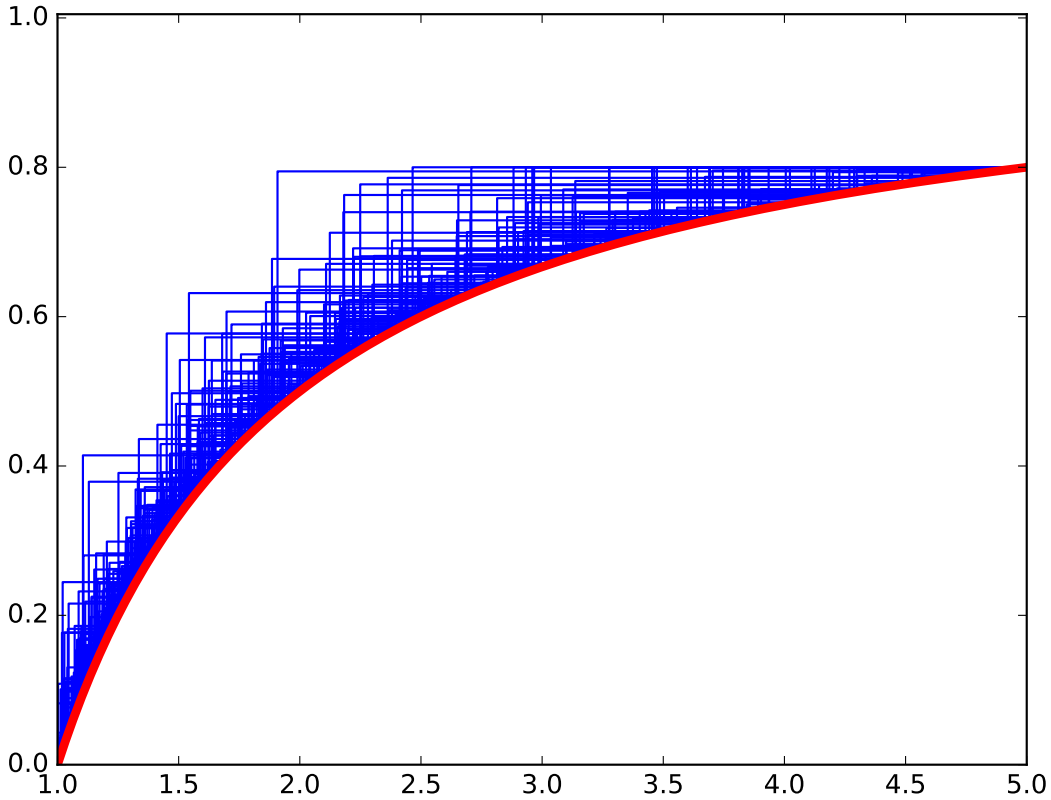
Proof sketch

Imagine the sequence $\widetilde{p}_{s,i}$ was fixed in advance. I can compute exactly the distribution of all $z_{s,i,j}$.



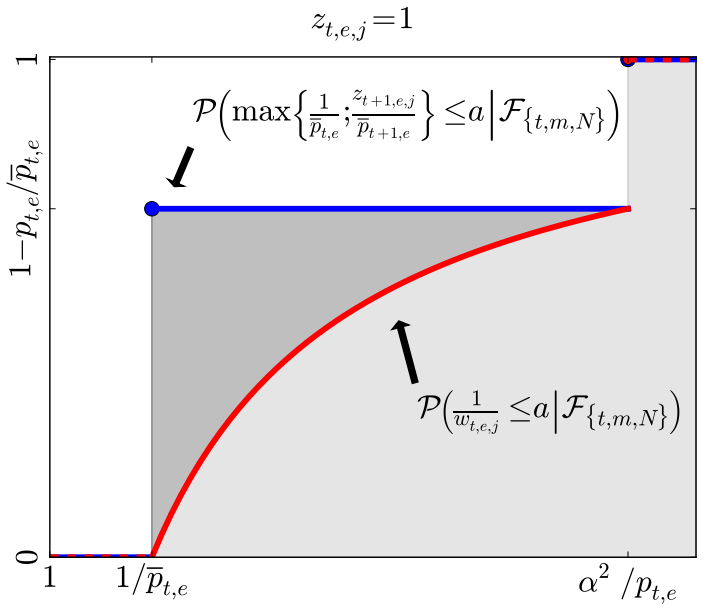
Proof sketch

Imagine the sequence $\widetilde{p}_{s,i}$ was fixed in advance. I can compute exactly the distribution of all $z_{s,i,j}$.



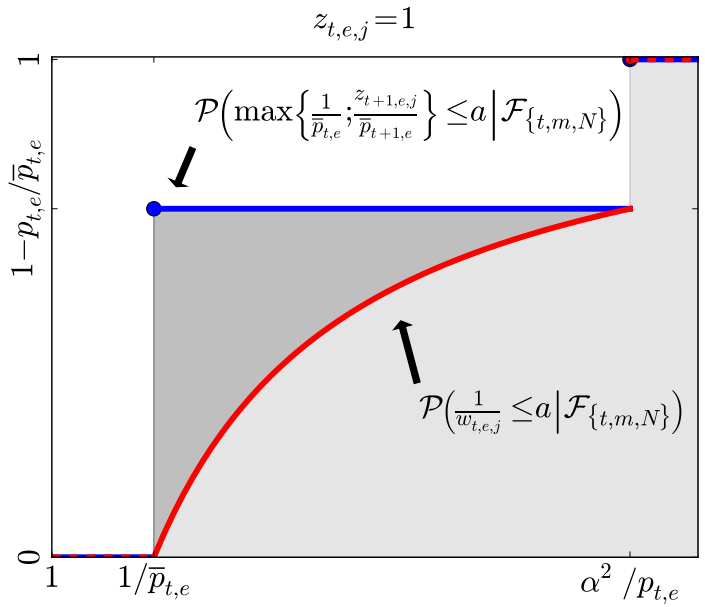
Proof sketch

Imagine the sequence $\widetilde{p}_{s,i}$ was fixed in advance. I can compute exactly the distribution of all $z_{s,i,j}$.



Proof sketch

Imagine the sequence $\widetilde{p}_{s,i}$ was fixed in advance. I can compute exactly the distribution of all $z_{s,i,j}$.



$$\mathbb{P}\left(\frac{1}{w_{0,i,j}}\leq a\right)=\begin{cases}0 & \text{for} \quad a<1\\ 1-\frac{1}{a} & \text{for} \quad 1\leq a<\alpha/p_{t,i}\\ 1 & \text{for} \quad \alpha/p_{t,i}\leq a\end{cases},$$

Proof sketch

We can now unwind the proof

5 dominate $\max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}^2} \right\}$ with $1/w_{i,j}$

Proof sketch

We can now unwind the proof

5 dominate $\max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}^2} \right\}$ with $1/w_{i,j}$

4 apply Bernstein inequality for indep. r.v. to bound $\mathbb{P}(\|\mathbf{W}\| \geq \sigma^2)$

Proof sketch

We can now unwind the proof

5 dominate $\max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}^2} \right\}$ with $1/w_{i,j}$

4 apply Bernstein inequality for indep. r.v. to bound $\mathbb{P}(\|\mathbf{W}\| \geq \sigma^2)$

3 apply Freedman inequality to bound $\mathbb{P}(\|\mathbf{Y}\| \geq \varepsilon \cap \|\mathbf{W}\| \leq \sigma^2)$

Proof sketch

We can now unwind the proof

5 dominate $\max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}^2} \right\}$ with $1/w_{i,j}$

4 apply Bernstein inequality for indep. r.v. to bound $\mathbb{P}(\|\mathbf{W}\| \geq \sigma^2)$

3 apply Freedman inequality to bound $\mathbb{P}(\|\mathbf{Y}\| \geq \varepsilon \cap \|\mathbf{W}\| \leq \sigma^2)$

2 apply another stochastic dominance argument to bound
$\mathbb{P}\left(|\mathcal{I}_t| \geq 3\overline{q} d_{\text{eff}} \gamma_t \cap \left\{ \forall t' \in \{1, \ldots, t\} : \|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\|_2 \leq \varepsilon \right\}\right)$

Proof sketch

We can now unwind the proof

5 dominate $\max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}^2} \right\}$ with $1/w_{i,j}$

4 apply Bernstein inequality for indep. r.v. to bound $\mathbb{P}(\|\mathbf{W}\| \geq \sigma^2)$

3 apply Freedman inequality to bound $\mathbb{P}(\|\mathbf{Y}\| \geq \varepsilon \cap \|\mathbf{W}\| \leq \sigma^2)$

2 apply another stochastic dominance argument to bound $\mathbb{P}\left(|\mathcal{I}_t| \geq 3\overline{q} d_{\text{eff}} \gamma_t \cap \left\{ \forall t' \in \{1, \ldots, t\} : \|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\|_2 \leq \varepsilon \right\}\right)$

1 union bound

Proof sketch

We can now unwind the proof

5 dominate $\max_{s=0}^{t-1} \left\{ \frac{z_{s,i,j}}{\widetilde{p}_{s,i}^2} \right\}$ with $1/w_{i,j}$

4 apply Bernstein inequality for indep. r.v. to bound $\mathbb{P}(\|\mathbf{W}\| \geq \sigma^2)$

3 apply Freedman inequality to bound $\mathbb{P}(\|\mathbf{Y}\| \geq \varepsilon \cap \|\mathbf{W}\| \leq \sigma^2)$

2 apply another stochastic dominance argument to bound
$\mathbb{P}\left(|\mathcal{I}_t| \geq 3\overline{q} d_{\text{eff}} \gamma_t \cap \left\{ \forall t' \in \{1, \ldots, t\} : \|\mathbf{P}_t - \widetilde{\mathbf{P}}_t\|_2 \leq \varepsilon \right\}\right)$

1 union bound

0 Q.E.D.

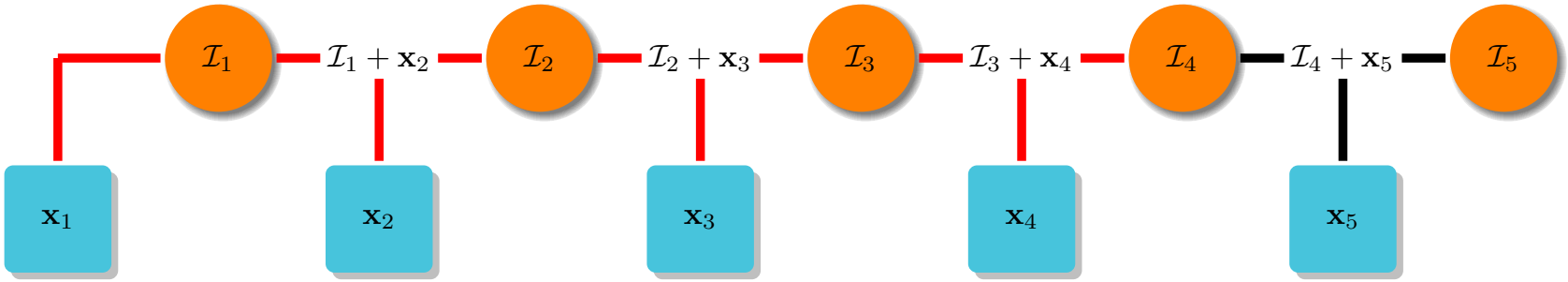
Distributed sequential sampling for adaptive kernel DL

SQUEAK is a strictly sequential algorithm

Distributed sequential sampling for adaptive kernel DL

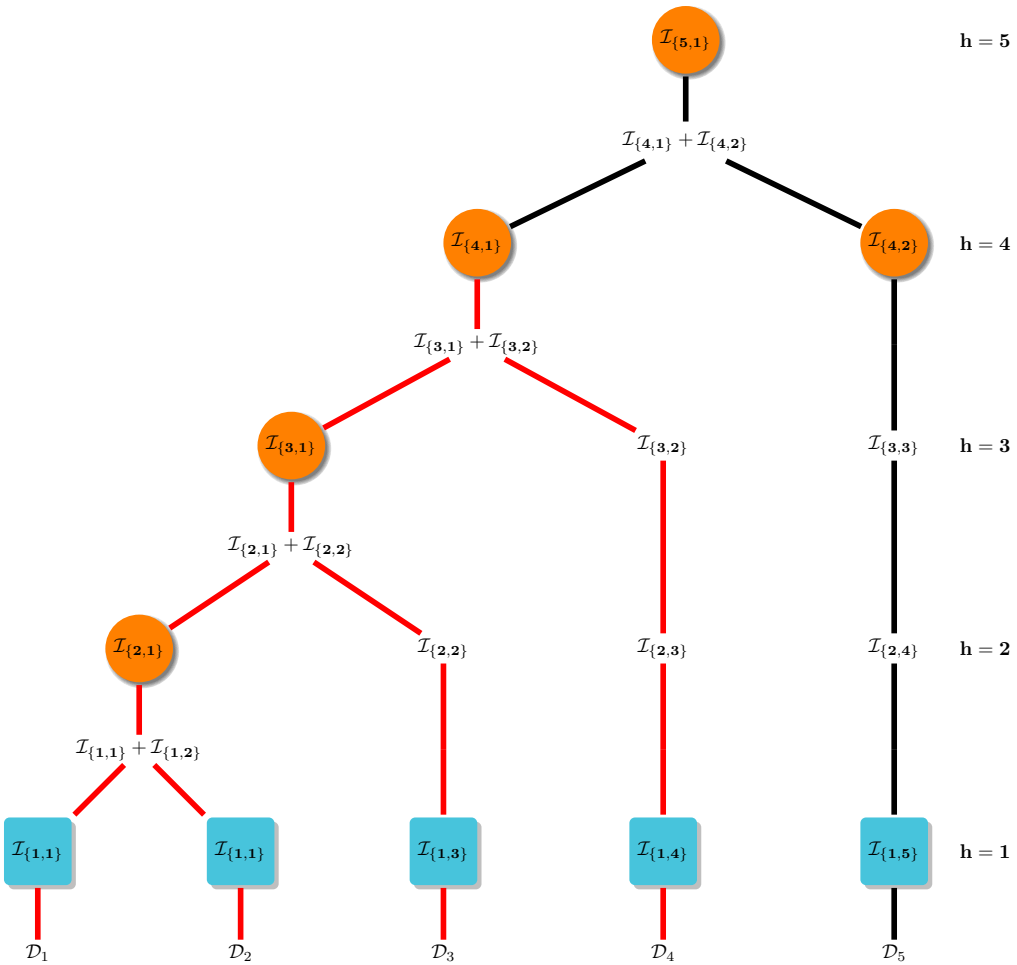
SQUEAK is a strictly sequential algorithm

We just did a sequential analysis



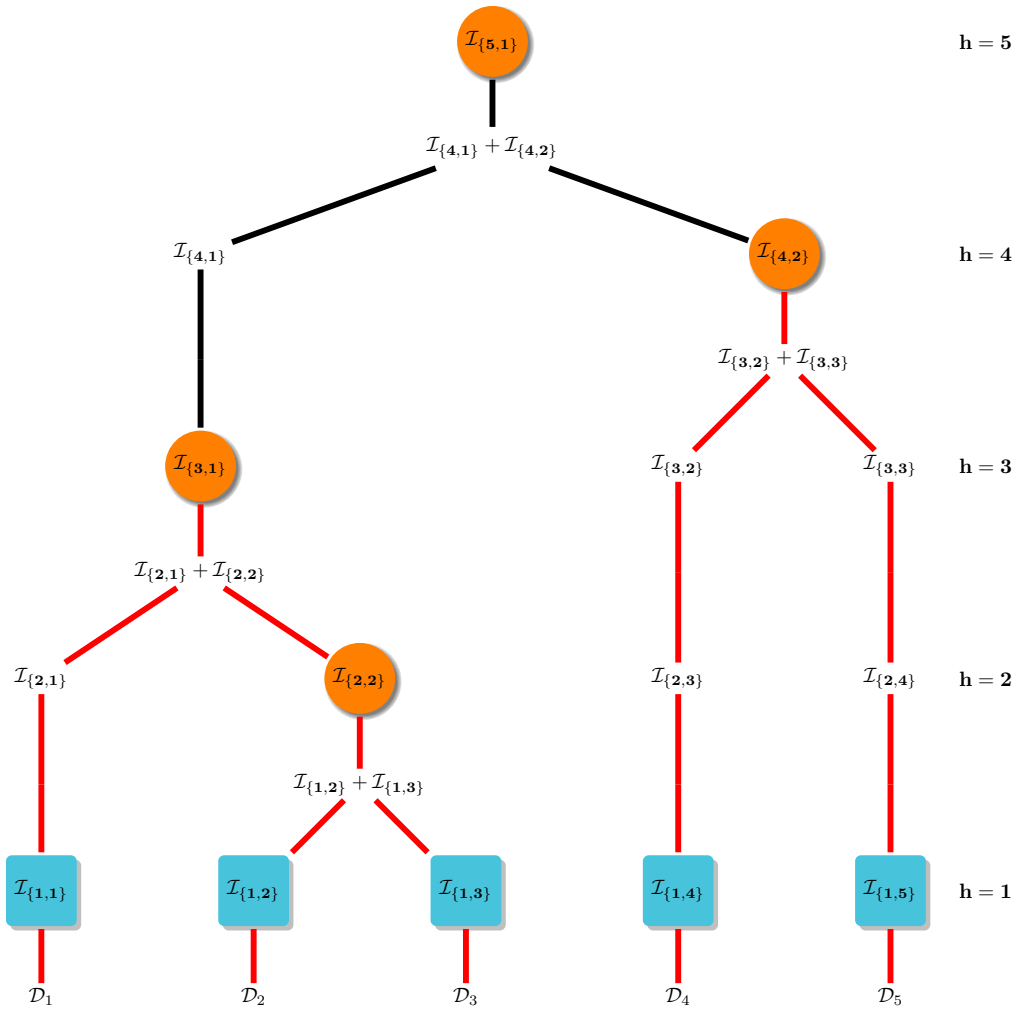
Distributed sequential sampling for adaptive kernel DL

SQUEAK is a strictly sequential algorithm

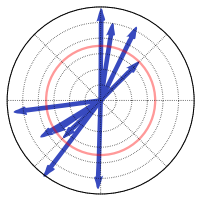


Distributed sequential sampling for adaptive kernel DL

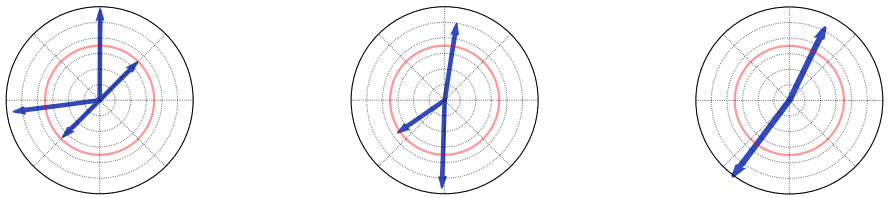
DISQUEAK is the distributed equivalent



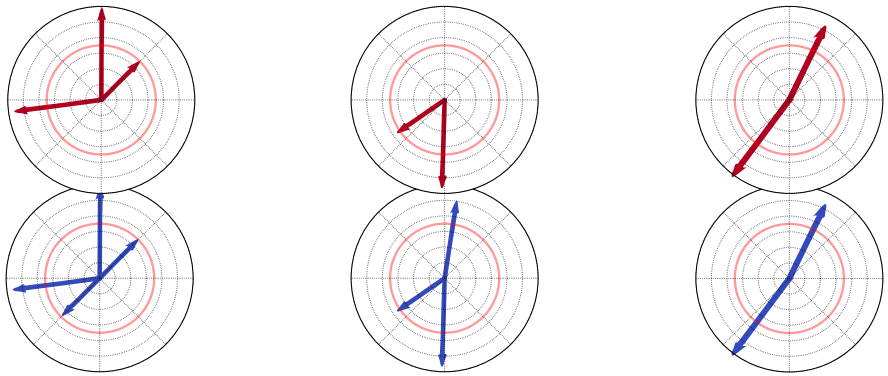
Distributed sequential sampling for adaptive kernel DL



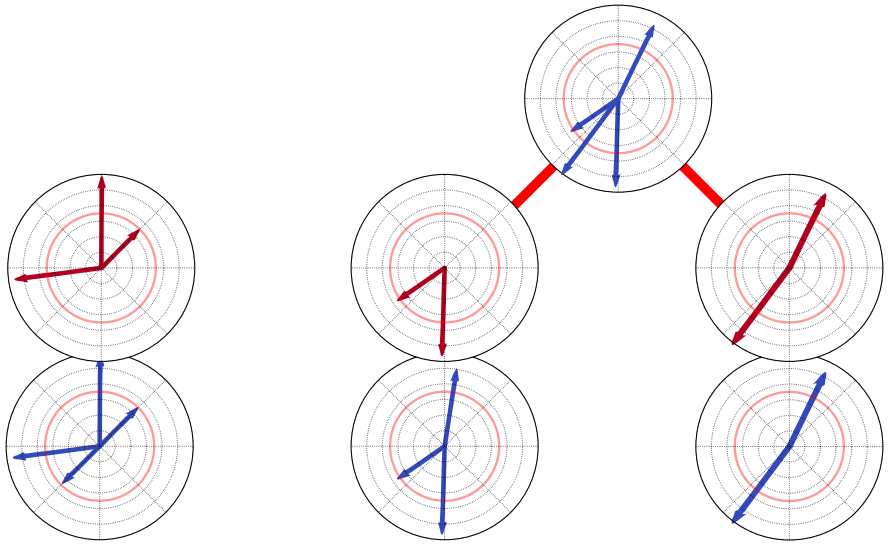
Distributed sequential sampling for adaptive kernel DL



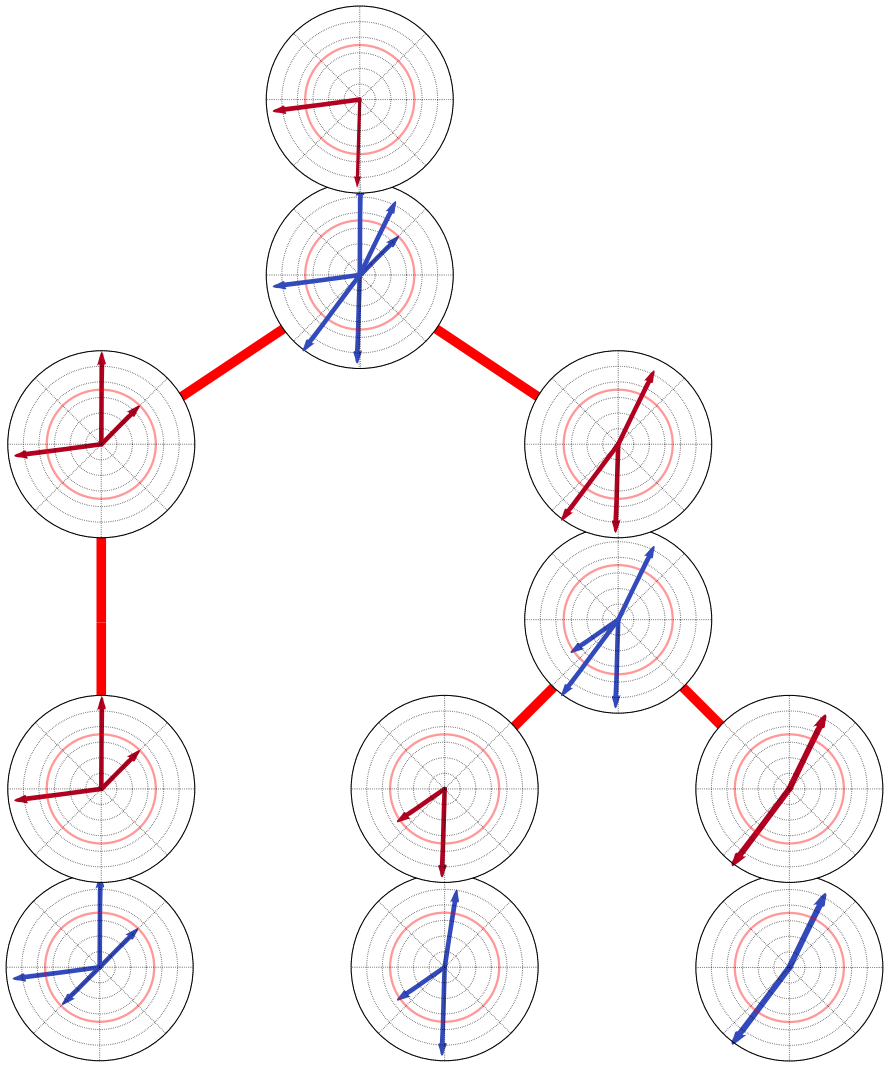
Distributed sequential sampling for adaptive kernel DL



Distributed sequential sampling for adaptive kernel DL



Distributed sequential sampling for adaptive kernel DL



DISQUEAK

Input: Dataset $\mathcal{D}$, regularization $\gamma, \overline{q}, \varepsilon$, **Output**: $\mathcal{I}_{\mathcal{D}}$

1: Partition $\mathcal{D}$ into disjoint sub-datasets $\mathcal{D}_i$
2: Run SQUEAK on each $\mathcal{D}_i$, build set $\mathcal{S}_1 = \{\mathcal{I}_{\mathcal{D}_i}\}_{i=1}^{k}$
3: **for** $h = 1, \ldots, k-1$ **do**
4: **if** $|\mathcal{S}_h| > 1$ **then** ▷DICT-MERGE
5: Pick two dictionaries $\mathcal{I}_{\mathcal{D}}, \mathcal{I}_{\mathcal{D}'}$ from $\mathcal{S}_h$
6: $\overline{\mathcal{I}} = \mathcal{I}_{\mathcal{D}} \cup \mathcal{I}_{\mathcal{D}'}$
7: $\mathcal{I}_{\mathcal{D},\mathcal{D}'} = \text{DICT-UPDATE}(\overline{\mathcal{I}})$ using Eq. (4)
8: Place $\mathcal{I}_{\mathcal{D},\mathcal{D}'}$ back into $\mathcal{S}_{h+1}$
9: **else**
10: $\mathcal{S}_{h+1} = \mathcal{S}_h$
11: **end if**
12: **end for**
13: Return $\mathcal{I}_{\mathcal{D}}$, the last dictionary in $\mathcal{S}_k$

$$\widetilde{\tau}_{\mathcal{D}\cup\mathcal{D}',i} = \frac{1-2\varepsilon}{\gamma}\left(k_{i,i} - \mathbf{k}_i^{\mathsf{T}}\overline{\mathbf{S}}(\overline{\mathbf{S}}^{\mathsf{T}}\mathbf{K}\overline{\mathbf{S}} + \gamma\mathbf{I})^{-1}\overline{\mathbf{S}}^{\mathsf{T}}\mathbf{k}_i\right), \qquad (4)$$

DISQUEAK

Theorem

Let $\alpha = (\frac{1+2\varepsilon}{1-2\varepsilon})$ *and* $\gamma > 1$. *For any* $0 \leq \varepsilon \leq 1$, *and* $0 \leq \delta \leq 1$, *if we run* DISQUEAK *with* $\overline{\mathbf{q}} = \mathcal{O}(\frac{\alpha}{\varepsilon^2}\log(\frac{\mathbf{n}}{\delta}))$, *then w.p.* $1-\delta$, *for all nodes* $\{h, l\}$ *in the merge tree*

(1) $\|\mathbf{P}_{\{\mathbf{h},\mathbf{l}\}} - \widetilde{\mathbf{P}}_{\{\mathbf{h},\mathbf{l}\}}\|_2 \leq \varepsilon$.

(2) $|\mathcal{I}_{\{\mathbf{h},\mathbf{l}\}}| \leq \mathcal{O}(\overline{q} d_{\mathit{eff}} \gamma_{\{h,l\}}) \leq \mathcal{O}(\frac{\alpha}{\varepsilon^2}\mathbf{d}_{\mathit{eff}}\gamma_{\mathbf{n}}\log(\frac{\mathbf{n}}{\delta}))$.

- Same accuracy as SQUEAK but much faster
- Space/accuracy guarantees for all nodes
- Much more space used, but spread across many machines
- Runtime depends on exact merge tree
  - ↳ Fully unbalanced tree: $\mathcal{O}(n|\mathcal{I}_n|^3)$, same as SQUEAK
    - ↳ Fully balanced tree: $\mathcal{O}(\log(n)|\mathcal{I}_n|^3)$ time, $\mathcal{O}(n|\mathcal{I}_n|^3)$ work!

Comparison

	Time	$\lvert\mathcal{I}_n\rvert$ (Total space $= \mathcal{O}(n\lvert\mathcal{I}_n\rvert)$)	Increm.
EXACT	n^3	n	-
Bach'13	$\frac{n d_{\max n}^2}{\varepsilon} + \frac{d_{\max n}^3}{\varepsilon}$	$\frac{d_{\max,n}}{\varepsilon}$	No
A&M'15	$n(\lvert\mathcal{I}_n\rvert)^2$	$\left(\frac{\lambda_{\min} + n\gamma\varepsilon}{\lambda_{\min} - n\gamma\varepsilon}\right) d_{\text{eff}\,n} + \frac{\mathrm{Tr}(\mathbf{K}_n)}{\gamma\varepsilon}$	No
Cal&al'16	$\frac{\lambda_{\max}^2}{\gamma^2}\frac{n^2 d_{\text{eff}\,n}^3}{\varepsilon^2}$	$\frac{\lambda_{\max}}{\gamma}\frac{d_{\text{eff}\,n}}{\varepsilon^2}$	Yes
SQUEAK	$\frac{n d_{\text{eff}\,n}^3}{\varepsilon^2}$	$\frac{d_{\text{eff}\,n}}{\varepsilon^2}$	Yes
RLS-SAMPLING	$\frac{n}{\varepsilon^2}$	$\frac{d_{\text{eff}\,n}}{\varepsilon^2}$	-
M&M'16	$\frac{n d_{\text{eff}\,n}^3}{\varepsilon^2}$	$\frac{d_{\text{eff}\,n}}{\varepsilon^2}$	No

SQUEAK - recap before application

SQUEAK and DISQUEAK

First method (with guarantees) to break $\mathcal{O}(n)$ time barrier using DISQUEAK, with M&M'16 first to break $\mathcal{O}(n^2)$ barrier

Strong reconstruction guarantees, suitable for many downstream kernel (and not) tasks

SQUEAK - recap before application

SQUEAK and DISQUEAK

- First method (with guarantees) to break $\mathcal{O}(n)$ time barrier using DISQUEAK, with M&M'16 first to break $\mathcal{O}(n^2)$ barrier
- Strong reconstruction guarantees, suitable for many downstream kernel (and not) tasks
- Final dictionary can be updated if new samples arrive

SQUEAK - recap before application

SQUEAK and DISQUEAK

- First method (with guarantees) to break $\mathcal{O}(n)$ time barrier using DISQUEAK, with M&M'16 first to break $\mathcal{O}(n^2)$ barrier
- Strong reconstruction guarantees, suitable for many downstream kernel (and not) tasks
- Final dictionary can be updated if new samples arrive
- Novel analysis, potentially useful for general importance sampling

SQUEAK - recap before application

SQUEAK and DISQUEAK

- First method (with guarantees) to break $\mathcal{O}(n)$ time barrier using DISQUEAK, with M&M'16 first to break $\mathcal{O}(n^2)$ barrier
- Strong reconstruction guarantees, suitable for many downstream kernel (and not) tasks
- Final dictionary can be updated if new samples arrive
- Novel analysis, potentially useful for general importance sampling

SQUEAK - recap before application

SQUEAK and DISQUEAK

- First method (with guarantees) to break $\mathcal{O}(n)$ time barrier using DISQUEAK, with M&M'16 first to break $\mathcal{O}(n^2)$ barrier
- Strong reconstruction guarantees, suitable for many downstream kernel (and not) tasks
- Final dictionary can be updated if new samples arrive
- Novel analysis, potentially useful for general importance sampling

Future work

Experiments

↳ Trivial to implement: 328 lines of python, single file, including distributed task queue `http://researchers.lille.inria.fr/~calandri/publications.html`

SQUEAK - recap before application

SQUEAK and DISQUEAK

- First method (with guarantees) to break $\mathcal{O}(n)$ time barrier using DISQUEAK, with M&M'16 first to break $\mathcal{O}(n^2)$ barrier
- Strong reconstruction guarantees, suitable for many downstream kernel (and not) tasks
- Final dictionary can be updated if new samples arrive
- Novel analysis, potentially useful for general importance sampling

Future work

Experiments

↳ Trivial to implement: 328 lines of python, single file, including distributed task queue `http://researchers.lille.inria.fr/~calandri/publications.html`
Preliminary results promising, easily scales to 100k of samples

SQUEAK - recap before application

SQUEAK and DISQUEAK

- First method (with guarantees) to break $\mathcal{O}(n)$ time barrier using DISQUEAK, with M&M'16 first to break $\mathcal{O}(n^2)$ barrier
- Strong reconstruction guarantees, suitable for many downstream kernel (and not) tasks
- Final dictionary can be updated if new samples arrive
- Novel analysis, potentially useful for general importance sampling

Future work

- Experiments
  - ↳ Trivial to implement: 328 lines of python, single file, including distributed task queue `http://researchers.lille.inria.fr/~calandri/publications.html`
    Preliminary results promising, easily scales to 100k of samples
- Beyond passive processing: SQUEAK for active learning

SQUEAK - recap before application

SQUEAK and DISQUEAK

- First method (with guarantees) to break $\mathcal{O}(n)$ time barrier using DISQUEAK, with M&M'16 first to break $\mathcal{O}(n^2)$ barrier
- Strong reconstruction guarantees, suitable for many downstream kernel (and not) tasks
- Final dictionary can be updated if new samples arrive
- Novel analysis, potentially useful for general importance sampling

Future work

- Experiments
  - ↳ Trivial to implement: 328 lines of python, single file, including distributed task queue `http://researchers.lille.inria.fr/~calandri/publications.html`
    Preliminary results promising, easily scales to 100k of samples
- Beyond closed formulas: SQUEAK for gradient based methods (done)

Online Kernel Learning (OKL)

Online game between learner and adversary, at each round $t \in [T]$

1. the adversary reveals a new point $\varphi(\mathbf{x}_t) = \boldsymbol{\phi}_t \in \mathcal{H}$
2. the learner chooses a function $f_{\mathbf{w}_t}$ and predicts $f_{\mathbf{w}_t}(\mathbf{x}_t) = \varphi(\mathbf{x}_t)^\top \mathbf{w}_t$,
3. the adversary reveals the curved loss ℓ_t,
4. the learner suffers $\ell_t(\boldsymbol{\phi}_t^\top \mathbf{w}_t)$ and observes the associated gradient $\mathbf{g}_t$.

Online Kernel Learning (OKL)

Online game between learner and adversary, at each round $t \in [T]$

1. the adversary reveals a new point $\varphi(\mathbf{x}_t) = \phi_t \in \mathcal{H}$
2. the learner chooses a function $f_{\mathbf{w}_t}$ and predicts $f_{\mathbf{w}_t}(\mathbf{x}_t) = \varphi(\mathbf{x}_t)^\top \mathbf{w}_t$,
3. the adversary reveals the curved loss ℓ_t,
4. the learner suffers $\ell_t(\phi_t^\top \mathbf{w}_t)$ and observes the associated gradient $\mathbf{g}_t$.

Kernel

$\varphi(\cdot) : \mathcal{X} \to \mathcal{H}$ is the high-dimensional (possibly infinite) map

$\mathbf{\Phi}_t = [\phi_1, \dots, \phi_t]$, $\mathbf{\Phi}_t^\top \mathbf{\Phi}_t = \mathbf{K}_t$ (kernel trick)

$\mathbf{g}_t = \ell_t'(\phi_t^\top \mathbf{w}_t)\phi_t := \dot{g}_t \phi_t$

Online Kernel Learning (OKL)

Online game between learner and adversary, at each round $t \in [T]$

1. the adversary reveals a new point $\varphi(\mathbf{x}_t) = \phi_t \in \mathcal{H}$
2. the learner chooses a function $f_{\mathbf{w}_t}$ and predicts $f_{\mathbf{w}_t}(\mathbf{x}_t) = \varphi(\mathbf{x}_t)^\top \mathbf{w}_t$,
3. the adversary reveals the curved loss ℓ_t,
4. the learner suffers $\ell_t(\phi_t^\top \mathbf{w}_t)$ and observes the associated gradient $\mathbf{g}_t$.

Kernel

$\varphi(\cdot) : \mathcal{X} \to \mathcal{H}$ is the high-dimensional (possibly infinite) map

$\mathbf{\Phi}_t = [\phi_1, \dots, \phi_t]$, $\mathbf{\Phi}_t^\top \mathbf{\Phi}_t = \mathbf{K}_t$ (kernel trick)

$\mathbf{g}_t = \ell_t'(\phi_t^\top \mathbf{w}_t)\phi_t := \dot{g}_t \phi_t$

Learning to minimize regret $R(\mathbf{w}) = \sum_{t=1}^T \ell_t(\phi_t \mathbf{w}_t) - \ell_t(\phi_t \mathbf{w})$
and compete with best-in-hindsight $\mathbf{w}^* := \arg\min_{\mathbf{w} \in \mathcal{H}} \sum_{t=1}^T \ell_t(\phi_t \mathbf{w})$

Curved losses?

We assume the losses ℓ_t are scalar Lipschitz

$$|\ell_t'(z)| \leq L \text{ whenever } |z| \leq C$$

and curved

$$\ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{w}) \geq \ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{u}) + \nabla\ell_t(\boldsymbol{\phi}_t^\mathsf{T}\mathbf{u})^\mathsf{T}(\mathbf{w}-\mathbf{u}) + \sigma\left(\nabla\ell_\mathbf{t}(\boldsymbol{\phi}_\mathbf{t}^\mathsf{T}\mathbf{u})^\mathsf{T}(\mathbf{w}-\mathbf{u})\right)^\mathbf{2}.$$

Curved losses?

We assume the losses ℓ_t are scalar Lipschitz

$$|\ell_t'(z)| \leq L \text{ whenever } |z| \leq C$$

and curved

$$\ell_t(\phi_t^\mathsf{T}\mathbf{w}) \geq \ell_t(\phi_t^\mathsf{T}\mathbf{u}) + \nabla\ell_t(\phi_t^\mathsf{T}\mathbf{u})^\mathsf{T}(\mathbf{w}-\mathbf{u}) + \sigma\left(\nabla\ell_\mathbf{t}(\phi_\mathbf{t}^\mathsf{T}\mathbf{u})^\mathsf{T}(\mathbf{w}-\mathbf{u})\right)^2.$$

You already use curved losses

weaker than strong convexity

↳ strongly convex only along $\nabla\ell_t(\phi_t^\mathsf{T}\mathbf{u})$

satisfied by exp-concave losses

↳ squared loss, squared hinge-loss

Second-Order OKL (Kernel Online Newton Step)

Second-Order Gradient Descent

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{A}_t^{-1}\mathbf{g}_t, \qquad \mathbf{A}_t = \sum_{s=1}^{t} \sigma \mathbf{g}_t \mathbf{g}_t^\top + \alpha \mathbf{I}$$

If $\varphi(\mathbf{x}) = \mathbf{x} : \mathbb{R}^d \to \mathbb{R}^d$ is the identity, Online Newton Step Hazan et al. 2006

↳ $\mathcal{O}(\mathbf{d^2})$ time/space per-step (Bottleneck: storing and inverting $\mathbf{X}_t^\top \mathbf{X}_t$)

$R(\mathbf{w}^*) \leq \alpha \|\mathbf{w}^* - \mathbf{w}_0\|_2^2 + \mathbf{d} \log(\mathbf{T})$

Sketched-ONS fast approximation, but only for $\varphi(\mathbf{x})$ identity and low-rank $\mathbf{X}_t^\top \mathbf{X}_t$ Luo et al. 2016

If $\varphi(\mathbf{x}) : \mathbb{R}^d \to \mathcal{H}$ is arbitrary, Kernel-ONS Calandriello et al. 2017c

↳ $\mathcal{O}(\mathbf{t^2})$ time/space per-step (Bottleneck: storing and inverting $\mathbf{K}_t$)

$$R(\mathbf{w}^*) \leq \underbrace{\alpha \|\mathbf{w}^* - \mathbf{w_0}\|_2^2}_{a} + \underbrace{\mathbf{d_{\text{eff}}}^T(\alpha/(\mathbf{L}\sigma))\log(\mathbf{T})}_{b}$$

(a) start cost, (b) effective dimension (degrees of freedom) of data

Effective Dimension

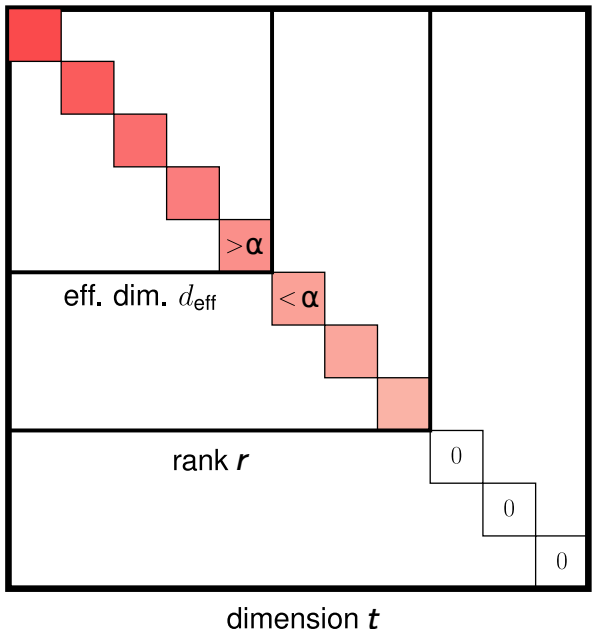
Formally $d_{\text{eff}}^{T}(\frac{\alpha}{L\sigma})$ is an $\frac{\alpha}{L\sigma}$ soft-thresholded version of the rank defined as

$$d_{\text{eff}}^{T}\left(\frac{\alpha}{L\sigma}\right) = \text{Tr}\left(\mathbf{K}_T\left(\mathbf{K}_T + \frac{\alpha}{L\sigma}\mathbf{I}\right)^{-1}\right) = \sum_{t=1}^{T}\frac{\lambda_t}{\lambda_t + \frac{\alpha}{L\sigma}} \leq \text{Rank}(\mathbf{K}_T) = r$$

Effective Dimension

Formally $d_{\text{eff}}^{T}(\frac{\alpha}{L\sigma})$ is an $\frac{\alpha}{L\sigma}$ soft-thresholded version of the rank defined as

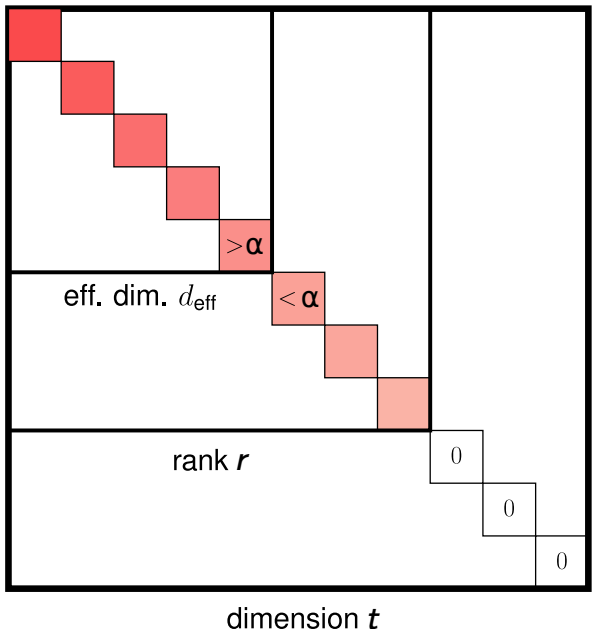
$$d_{\text{eff}}^{T}\left(\frac{\alpha}{L\sigma}\right) = \text{Tr}\left(\mathbf{K}_T\left(\mathbf{K}_T + \frac{\alpha}{L\sigma}\mathbf{I}\right)^{-1}\right) = \sum_{t=1}^{T}\frac{\lambda_t}{\lambda_t + \frac{\alpha}{L\sigma}} \leq \text{Rank}(\mathbf{K}_T) = r$$



Effective Dimension

Formally $d_{\text{eff}}^{T}(\frac{\alpha}{L\sigma})$ is an $\frac{\alpha}{L\sigma}$ soft-thresholded version of the rank defined as

$$d_{\text{eff}}^{T}\left(\frac{\alpha}{L\sigma}\right) = \text{Tr}\left(\mathbf{K}_T\left(\mathbf{K}_T + \frac{\alpha}{L\sigma}\mathbf{I}\right)^{-1}\right) = \sum_{t=1}^{T} \frac{\lambda_t}{\lambda_t + \frac{\alpha}{L\sigma}} \leq \text{Rank}(\mathbf{K}_T) = r$$

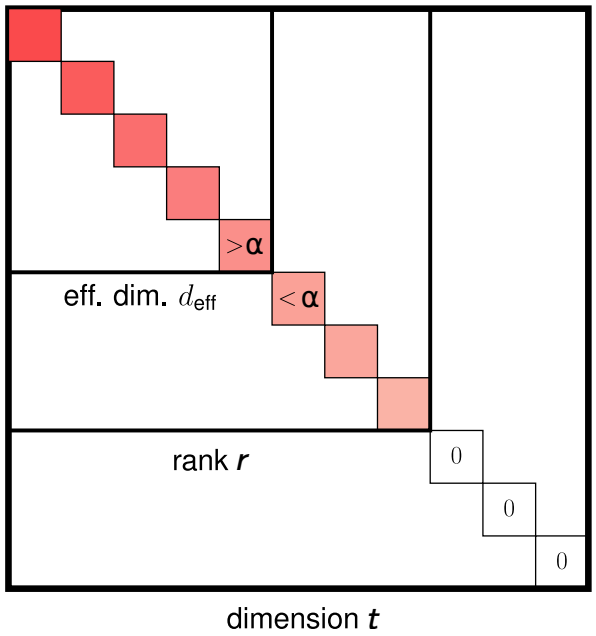


Intuitively, it quantifies the number of relevant orthogonal directions played by the adversary.

Effective Dimension

Formally $d_{\text{eff}}^{T}(\frac{\alpha}{L\sigma})$ is an $\frac{\alpha}{L\sigma}$ soft-thresholded version of the rank defined as

$$d_{\text{eff}}^{T}\left(\frac{\alpha}{L\sigma}\right) = \text{Tr}\left(\mathbf{K}_T\left(\mathbf{K}_T + \frac{\alpha}{L\sigma}\mathbf{I}\right)^{-1}\right) = \sum_{t=1}^{T} \frac{\lambda_t}{\lambda_t + \frac{\alpha}{L\sigma}} \leq \text{Rank}(\mathbf{K}_T) = r$$

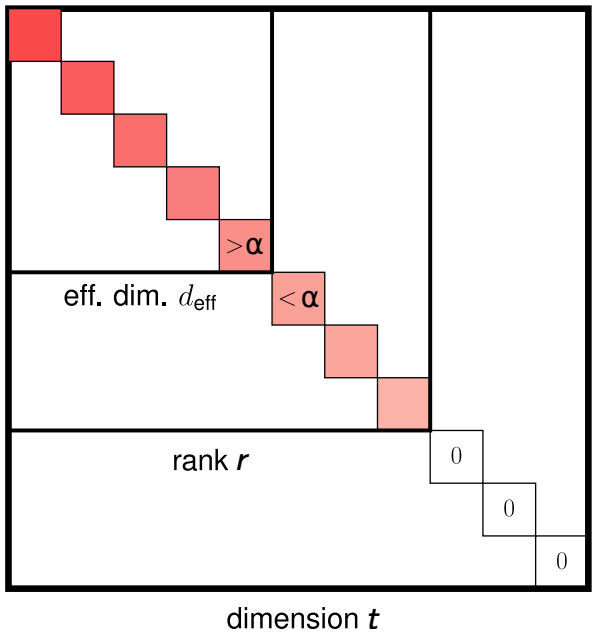


A direction (eigenvector) is relevant if its importance (eigenvalue) is larger than the Lipschitz discounted regularization $\alpha/(L\sigma)$

Effective Dimension

Formally $d_{\text{eff}}^{T}(\frac{\alpha}{L\sigma})$ is an $\frac{\alpha}{L\sigma}$ soft-thresholded version of the rank defined as

$$d_{\text{eff}}^{T}\left(\frac{\alpha}{L\sigma}\right) = \text{Tr}\left(\mathbf{K}_T\left(\mathbf{K}_T + \frac{\alpha}{L\sigma}\mathbf{I}\right)^{-1}\right) = \sum_{t=1}^{T}\frac{\lambda_t}{\lambda_t + \frac{\alpha}{L\sigma}} \leq \text{Rank}(\mathbf{K}_T) = r$$



Assume $\|\phi_t\| = 1$, then if all ϕ_t are orthogonal and $\alpha = \sqrt{T}$ then

$$d_{\text{eff}}^{T}(\sqrt{T}) \sim \sqrt{T}$$

and

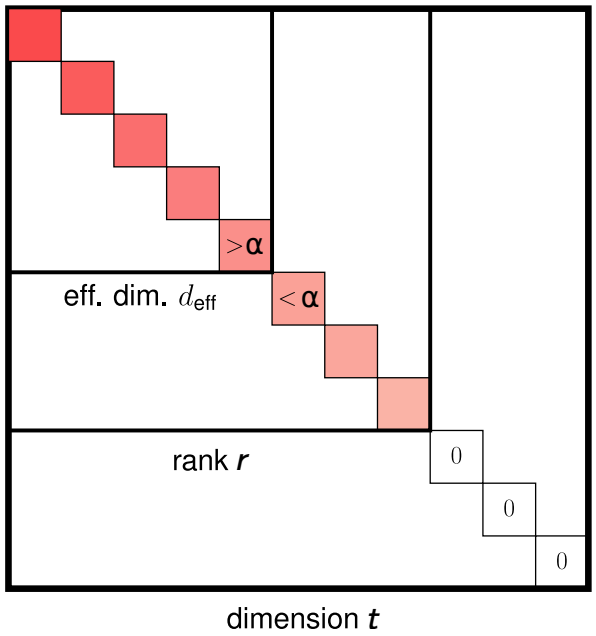
$$R(\mathbf{w}^*) \leq \sqrt{T} + d_{\text{eff}}^{T}(\sqrt{T})\log(T) \sim \sqrt{T}$$

recover first order bound.

Effective Dimension

Formally $d_{\text{eff}}^{T}(\frac{\alpha}{L\sigma})$ is an $\frac{\alpha}{L\sigma}$ soft-thresholded version of the rank defined as

$$d_{\text{eff}}^{T}\left(\frac{\alpha}{L\sigma}\right) = \text{Tr}\left(\mathbf{K}_T\left(\mathbf{K}_T + \frac{\alpha}{L\sigma}\mathbf{I}\right)^{-1}\right) = \sum_{t=1}^{T} \frac{\lambda_t}{\lambda_t + \frac{\alpha}{L\sigma}} \leq \text{Rank}(\mathbf{K}_T) = r$$



If all ϕ_t come from a bounded distribution or a finite set and $\alpha = 1$ then

$$d_{\text{eff}}^{T}(1) \sim \mathcal{O}(1) \leq r$$

is constant in T and

$$R(\mathbf{w}^*) \leq \mathcal{O}(1) + \mathcal{O}(1)\log(T) \sim \log T$$

logarithmic in T.

Ideal method

How to achieve adaptive $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret
but without computational complexity depending on t?

Ideal method

How to achieve adaptive $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret
but without computational complexity depending on t?

Use approximate second order gradient in $\mathcal{H}$ Calandriello et al. 2017c

↳ $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret, but runtime still depends on t

Ideal method

How to achieve adaptive $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret
but without computational complexity depending on t?

Use approximate second order gradient in $\mathcal{H}$ Calandriello et al. 2017c

↳ $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret, but runtime still depends on t

Use exact second order gradient in approximate $\widetilde{\mathcal{H}}$

$$\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \mathbf{w}^*) = \sum_{t=1}^{T} \underbrace{\ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}})}_{a} + \underbrace{\ell_t(\phi_t \overline{\mathbf{w}}) - \ell_t(\phi_t \mathbf{w}^*)}_{b}$$

Ideal method

How to achieve adaptive $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret
but without computational complexity depending on t?

Use approximate second order gradient in $\mathcal{H}$ Calandriello et al. 2017c

↳ $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret, but runtime still depends on t

Use exact second order gradient in approximate $\widetilde{\mathcal{H}}$

$$\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \mathbf{w}^*) = \sum_{t=1}^{T} \underbrace{\ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}})}_{a} + \underbrace{\ell_t(\phi_t \overline{\mathbf{w}}) - \ell_t(\phi_t \mathbf{w}^*)}_{b}$$

(a) error between online and batch in $\widetilde{\mathcal{H}}$:

↳ $d_{\text{eff}}^{T}(\alpha)\log(T)$ bound using KONS analysis

Ideal method

How to achieve adaptive $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret
but without computational complexity depending on t?

Use approximate second order gradient in $\mathcal{H}$ Calandriello et al. 2017c

↳ $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret, but runtime still depends on t

Use exact second order gradient in approximate $\widetilde{\mathcal{H}}$

$$\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \mathbf{w}^*) = \sum_{t=1}^{T} \underbrace{\ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}})}_{a} + \underbrace{\ell_t(\phi_t \overline{\mathbf{w}}) - \ell_t(\phi_t \mathbf{w}^*)}_{b}$$

(a) error between online and batch in $\widetilde{\mathcal{H}}$:

↳ $d_{\text{eff}}^{T}(\alpha)\log(T)$ bound using KONS analysis

(b) error between $\overline{\mathbf{w}}$ best in $\widetilde{\mathcal{H}}$ and $\mathbf{w}^*$ best in $\mathcal{H}$: bound how?

Subspace approximation error

$\widetilde{\mathcal{H}}$ cannot be fixed

↳ the adversary will find orthogonal points and exploit this

Subspace approximation error

$\widetilde{\mathcal{H}}$ cannot be fixed

↳ the adversary will find orthogonal points and exploit this

Use Nyström approximation instead and adapt it online

↳ $\widetilde{\mathcal{H}}_t = \text{Span}(\mathcal{I}_t)$ defined using m_t inducing points $\mathcal{I}_t = \{\phi_s\}_{s=1}^{m_t}$

If the adversary plays a "sufficiently orthogonal" ϕ_t, add it to $\mathcal{I}_{t+1}$

Subspace approximation error

$\widetilde{\mathcal{H}}$ cannot be fixed

↳ the adversary will find orthogonal points and exploit this

Use Nyström approximation instead and adapt it online

↳ $\widetilde{\mathcal{H}}_t = \mathrm{Span}(\mathcal{I}_t)$ defined using m_t inducing points $\mathcal{I}_t = \{\phi_s\}_{s=1}^{m_t}$

If the adversary plays a "sufficiently orthogonal" ϕ_t, add it to $\mathcal{I}_{t+1}$

$\widetilde{\mathcal{H}}_t$ is finite dimensional: runtime independent of t

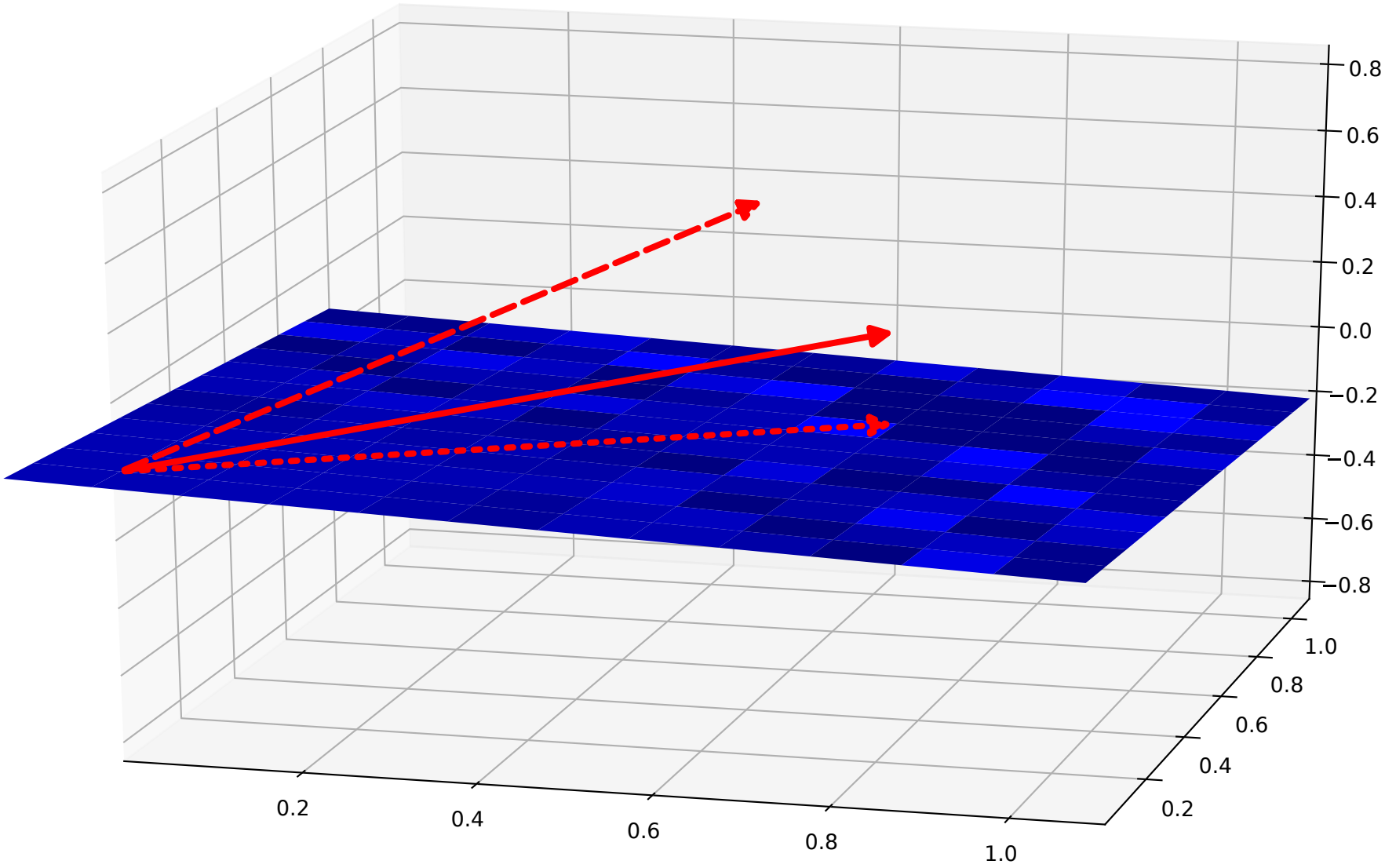
↳ Easy to embed (project) points as

$$\widetilde{\varphi}(\cdot) = \mathbf{\Sigma}^{-1}\mathbf{U}^\top \mathbf{\Phi}_{\mathcal{I}}^\top \varphi(\cdot) : \mathbb{R}^d \to \mathbb{R}^{m_t}$$

with $\mathbf{K}_{\mathcal{I}} = \mathbf{\Phi}_{\mathcal{I}}^\top \mathbf{\Phi}_{\mathcal{I}} = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}\mathbf{U}^\top$

$\mathcal{O}(m_t^2)$ time/space cost to run exact KONS in $\widetilde{\mathcal{H}}_t$

Aproximating what?



Subspace approximation error

"sufficiently orthogonal" is measured using **γ-ridge leverage scores**

$$\mathbb{P}(\text{include } \phi_t \text{ in } \mathcal{I}_t) \sim \phi_t^\mathsf{T}\phi_t - \phi_t^\mathsf{T}(\mathbf{\Phi}_{\mathcal{I}_{t-1}}\mathbf{\Phi}_{\mathcal{I}_{t-1}}^\mathsf{T} + \gamma\mathbf{I})^{-1}\phi_t$$

Also computable in m_t^2 time.

Subspace approximation error

"sufficiently orthogonal" is measured using **γ-ridge leverage scores**

$$\mathbb{P}(\text{include } \phi_t \text{ in } \mathcal{I}_t) \sim \phi_t^\mathsf{T}\phi_t - \phi_t^\mathsf{T}(\mathbf{\Phi}_{\mathcal{I}_{t-1}}\mathbf{\Phi}_{\mathcal{I}_{t-1}}^\mathsf{T} + \gamma\mathbf{I})^{-1}\phi_t$$

Also computable in m_t^2 time.

Guarantees that Calandriello et al. 2017c

$$m_t \leq d_{\text{eff}}^t(\gamma)\log^2(T) \quad \text{(space/time)}, \quad \mathbf{K}_t - \widetilde{\mathbf{K}}_t \preceq \gamma\mathbf{I} \quad \text{(accuracy)}$$

Online/batch error

Use KONS guarantees to bound $\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}})$ separately for each $\widetilde{\mathcal{H}}_t$ and associated $\overline{\mathbf{w}}_j$

$$\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}}) \leq J d_{\text{eff}}^{T}(\alpha/(L\sigma)) \log(T) + \sum_{j=1}^{J} \underbrace{\alpha \|\overline{\mathbf{w}}_j - \mathbf{w}_{t_j}\|_2^2}_{\text{start costs}}$$

Online/batch error

Use KONS guarantees to bound $\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}})$ separately for each $\widetilde{\mathcal{H}}_t$ and associated $\overline{\mathbf{w}}_j$

$$\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}}) \leq J d_{\text{eff}}^{T}(\alpha/(L\sigma)) \log(T) + \sum_{j=1}^{J} \underbrace{\alpha \|\overline{\mathbf{w}}_j - \mathbf{w}_{t_j}\|_2^2}_{\text{start costs}}$$

Every time we change $\widetilde{\mathcal{H}}$ we pay $\alpha \|\overline{\mathbf{w}}_j - \mathbf{w}_{t_j}\|_2^2$

↳ the adversary can influence $\mathbf{w}_{t_j}$ and make it large

Online/batch error

Use KONS guarantees to bound $\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}})$ separately for each $\widetilde{\mathcal{H}}_t$ and associated $\overline{\mathbf{w}}_j$

$$\sum_{t=1}^{T} \ell_t(\phi_t \widetilde{\mathbf{w}}_t) - \ell_t(\phi_t \overline{\mathbf{w}}) \leq J d_{\text{eff}}^{T}(\alpha/(L\sigma)) \log(T) + \sum_{j=1}^{J} \underbrace{\alpha \|\overline{\mathbf{w}}_j - \mathbf{w}_{t_j}\|_2^2}_{\text{start costs}}$$

Every time we change $\widetilde{\mathcal{H}}$ we pay $\alpha \|\overline{\mathbf{w}}_j - \mathbf{w}_{t_j}\|_2^2$

↳ the adversary can influence $\mathbf{w}_{t_j}$ and make it large

Reset $\widetilde{\mathbf{w}}_t$ and $\widetilde{\mathbf{A}}_t$ when $\widetilde{\mathcal{H}}_t$ changes

↳ Wasteful, but not too often. At most $J \leq d_{\text{eff}}^{T}(\gamma)$ times.

Learning is preserved through $\widetilde{\mathcal{H}}_t$ that always improves

Adaptive doubling trick

Final regret guarantees

For any curved loss

$$R(\mathbf{w}^*) \leq J\left(\alpha\|\mathbf{w}^*\|_2^2 + d_{\text{eff}}^T \log(\alpha/(L\sigma))\right) + \frac{\gamma}{\alpha} T$$

Final regret guarantees

For any curved loss

$$R(\mathbf{w}^*) \leq J\left(\alpha\|\mathbf{w}^*\|_2^2 + d_{\text{eff}}^T \log(\alpha/(L\sigma))\right) + \frac{\gamma}{\alpha} T$$

Setting $\gamma = \alpha/T$ removes second term

↳ computational cost is $\mathcal{O}(d_{\text{eff}}^T(1/T)^2)$, still small in many cases

Final regret guarantees

For any curved loss

$$R(\mathbf{w}^*) \leq J\left(\alpha\|\mathbf{w}^*\|_2^2 + d_{\text{eff}}^T \log(\alpha/(L\sigma))\right) + \frac{\gamma}{\alpha} T$$

Setting $\gamma = \alpha/T$ removes second term

↳ computational cost is $\mathcal{O}(d_{\text{eff}}^T (1/T)^2)$, still small in many cases

For squared loss only and $\gamma = \alpha$

$$R(\mathbf{w}^*) \leq J\left(\alpha\|\mathbf{w}^*\|_2^2 + d_{\text{eff}}^T \log(\alpha/(L\sigma))\right) + J\left(\sum_{t=1}^{T} \ell_t(\boldsymbol{\phi}_t \mathbf{w}^*) + \alpha\|\mathbf{w}^*\|_2^2\right)$$

Last term $J(\sum_{t=1}^{T} \ell_t(\boldsymbol{\phi}_t \mathbf{w}^*) + \alpha\|\mathbf{w}^*\|_2^2)$ replaces $\frac{\gamma}{\alpha} T$

↳ regularized cumulative loss of $\mathbf{w}^*$

if $\mathcal{H}$ is good, very small

Conclusions

Algorithm	cadata $n = 20k$, $d = 8$			casp $n = 45k$, $d = 9$		
	Avg. Squared Loss	#SV	Time	Avg. Squared Loss	#SV	Time
FOGD	0.04097 $\pm$ 0.00015	30	—	0.08021 $\pm$ 0.00031	30	—
NOGD	0.03983 $\pm$ 0.00018	30	—	0.07844 $\pm$ 0.00008	30	—
PROS-N-KONS	0.03095 $\pm$ 0.00110	20	18.59	**0.06773** $\pm$ 0.00105	21	40.73
CON-KONS	**0.02850** $\pm$ 0.00174	19	18.45	**0.06832** $\pm$ 0.00315	20	40.91
B-KONS	0.03095 $\pm$ 0.00118	19	18.65	**0.06775** $\pm$ 0.00067	21	41.13
BATCH	0.02202 $\pm$ 0.00002	—	—	0.06100 $\pm$ 0.00003	—	—

Algorithm	slice $n = 53k$, $d = 385$			year $n = 463k$, $d = 90$		
	Avg. Squared Loss	#SV	Time	Avg. Squared Loss	#SV	Time
FOGD	**0.00726** $\pm$ 0.00019	30	—	0.01427 $\pm$ 0.00004	30	—
NOGD	0.02636 $\pm$ 0.00460	30	—	0.01427 $\pm$ 0.00004	30	—
DUAL-SGD	—	—	—	0.01440 $\pm$ 0.00000	100	—
PROS-N-KONS	did not complete	—	—	0.01450 $\pm$ 0.00014	149	884.82
CON-KONS	did not complete	—	—	0.01444 $\pm$ 0.00017	147	889.42
B-KONS	**0.00913** $\pm$ 0.00045	100	60	**0.01302** $\pm$ 0.00006	100	505.36
BATCH	0.00212 $\pm$ 0.00001	—	—	0.01147 $\pm$ 0.00001	—	—

Bibliography


Daniele Calandriello, Alessandro Lazaric, and Michal Valko. "Distributed Sequential Sampling for Kernel Matrix Approximation". In: International Conference on Artificial Intelligence and Statistics. 2017.


Daniele Calandriello, Alessandro Lazaric, and Michal Valko. "Efficient second-order online kernel learning with adaptive embedding". In: Neural Information Processing Systems. 2017.


Daniele Calandriello, Alessandro Lazaric, and Michal Valko. "International Conference on Machine Learning". In: Neural Information Processing Systems. 2017.


Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. "Logarithmic regret algorithms for online convex optimization". In: Conference on Learning Theory. Springer, 2006, pp. 499–513.


Haipeng Luo, Alekh Agarwal, Nicolo Cesa-Bianchi, and John Langford. "Efficient second-order online learning via sketching". In: Neural Information Processing Systems. 2016.