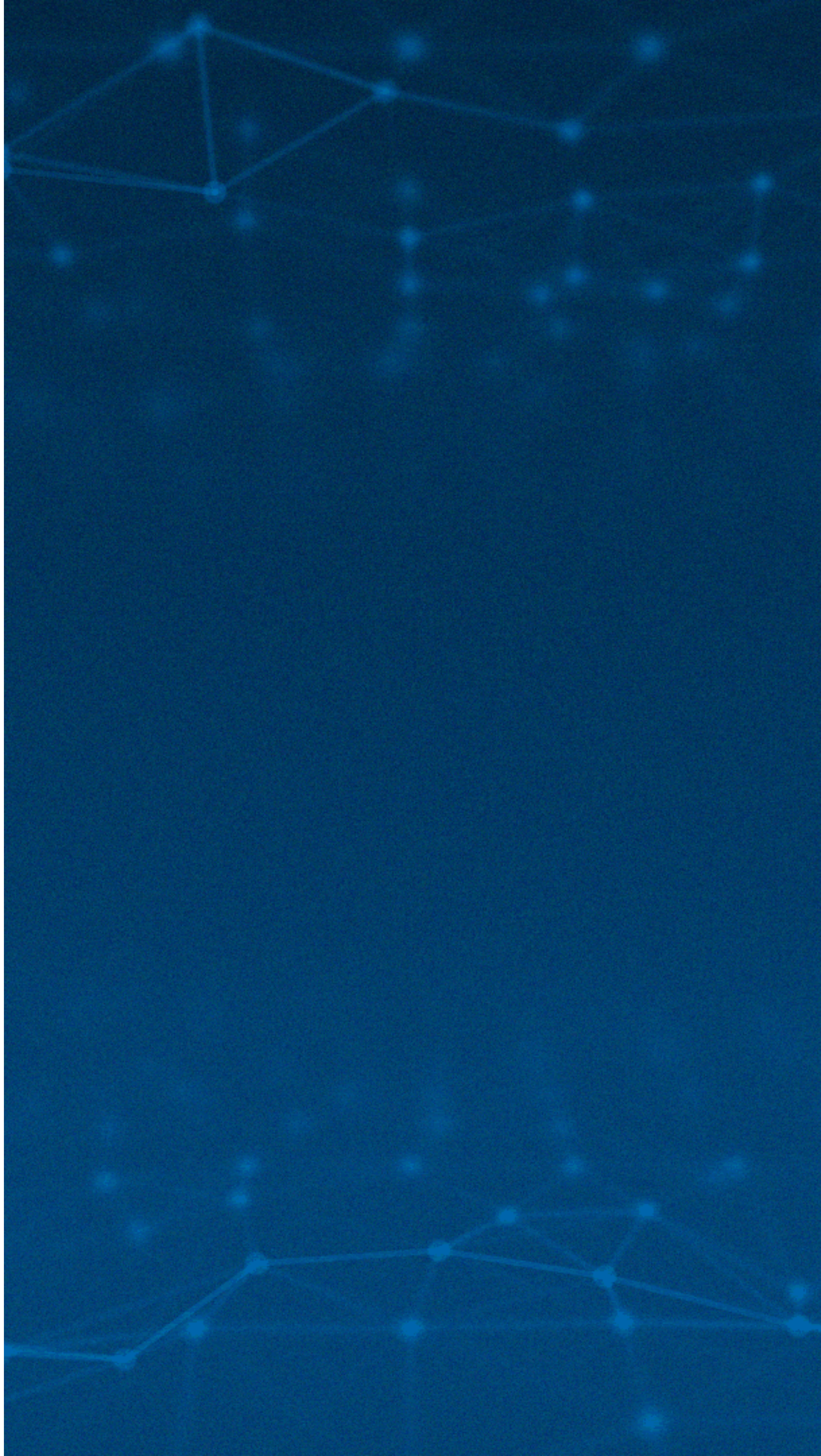
How negative dependence broke the quadratic barrier for learning with graphs and kernels

Michal Valko

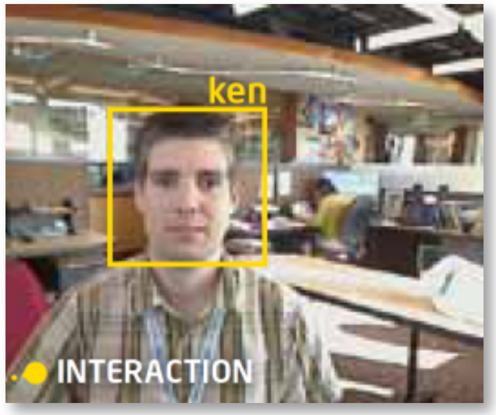
ONLINE LEARNING

when we reason on the fly



IN 2007 IT ALL STARTED WITH AN IDEA...

- Develop **sequential machine learning** recognition system
- System with **minimal feedback**
- 90% accurate over 90% of time
- With **theory** that guarantee's its performance
- **Efficient** (e.g., mobile device)



from B. Kveton

… AND RESULTED IN A REAL SYSTEM IN 2009

- **adaptive graph-based** recognition system
  - highly accurate
  - trained from a **small amount** of labeled data
  - real-time running time
  - robust to outliers
  - theoretical analysis

$$\frac{1}{n}\sum_{t}(\ell_t^{q}[t]-y_t)^2 \leq \frac{1}{n_l}\sum_{i\in l}(l_i^{*}-y_i)^2+\mathrm{O}(n^{-\frac{1}{2}})$$

from B. Kveton

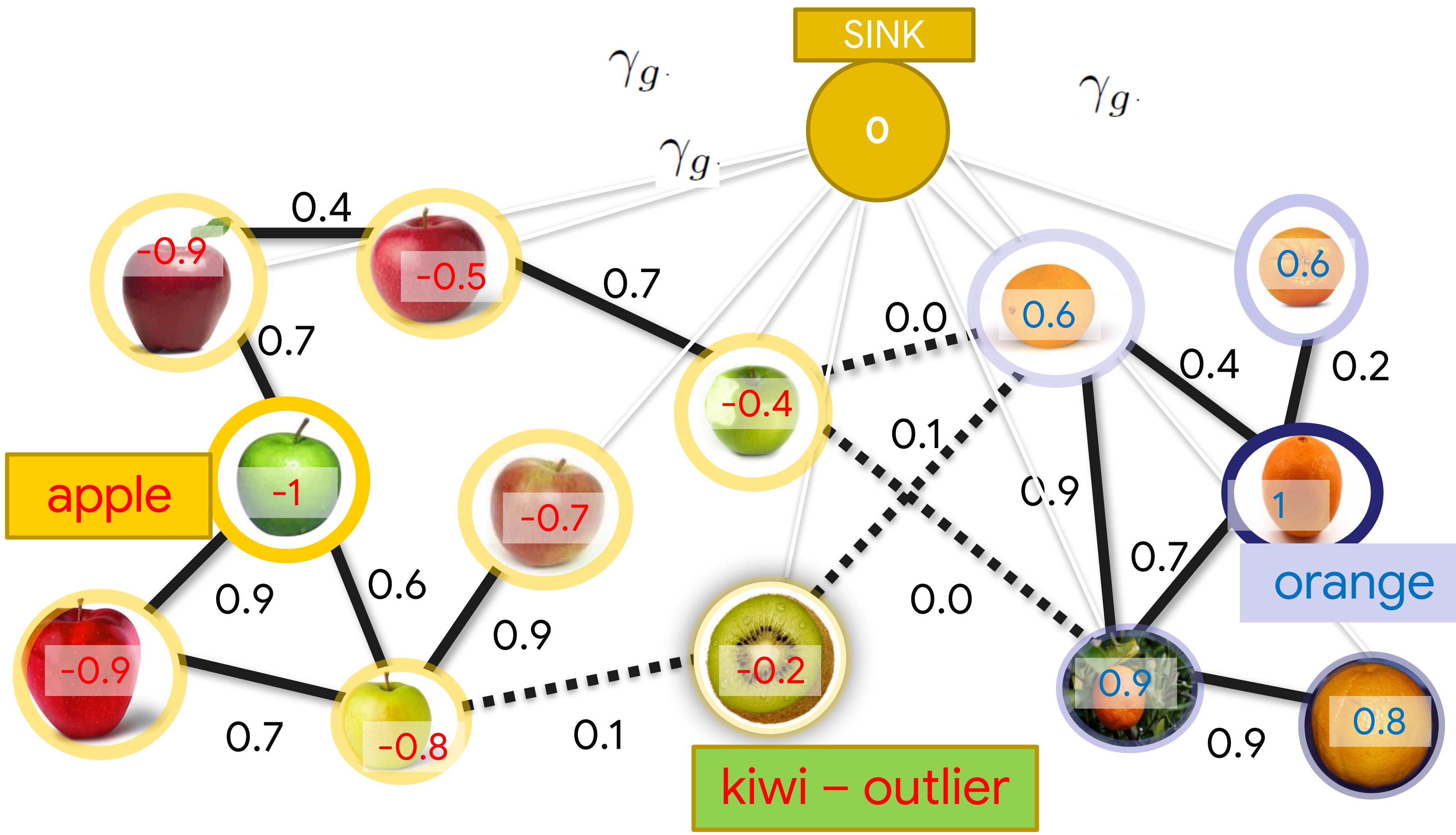
THIS CAN'T SCALE: CONNECTED CAR

Personalization

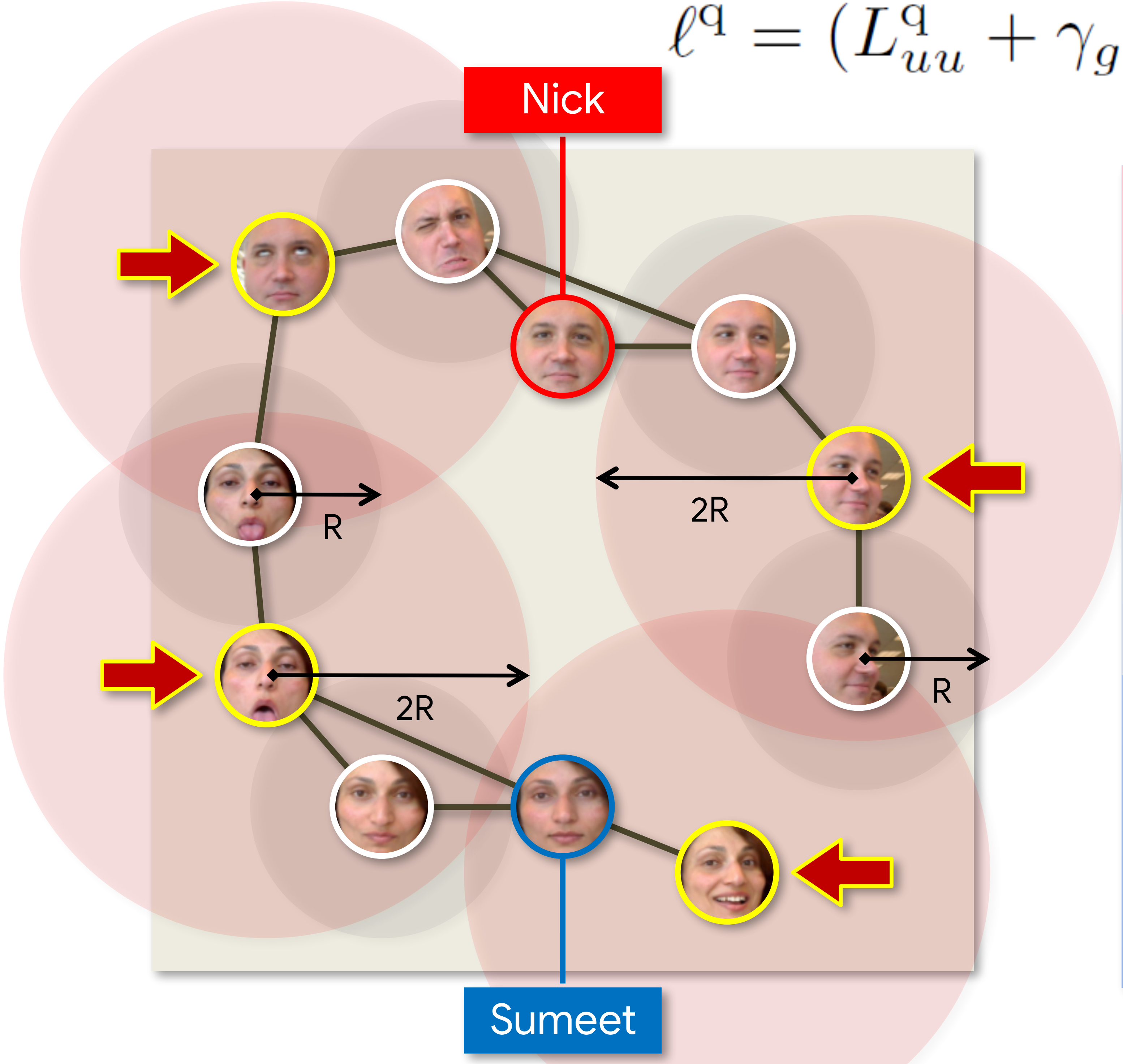
2 BIG REAL-WORLD ISSUES

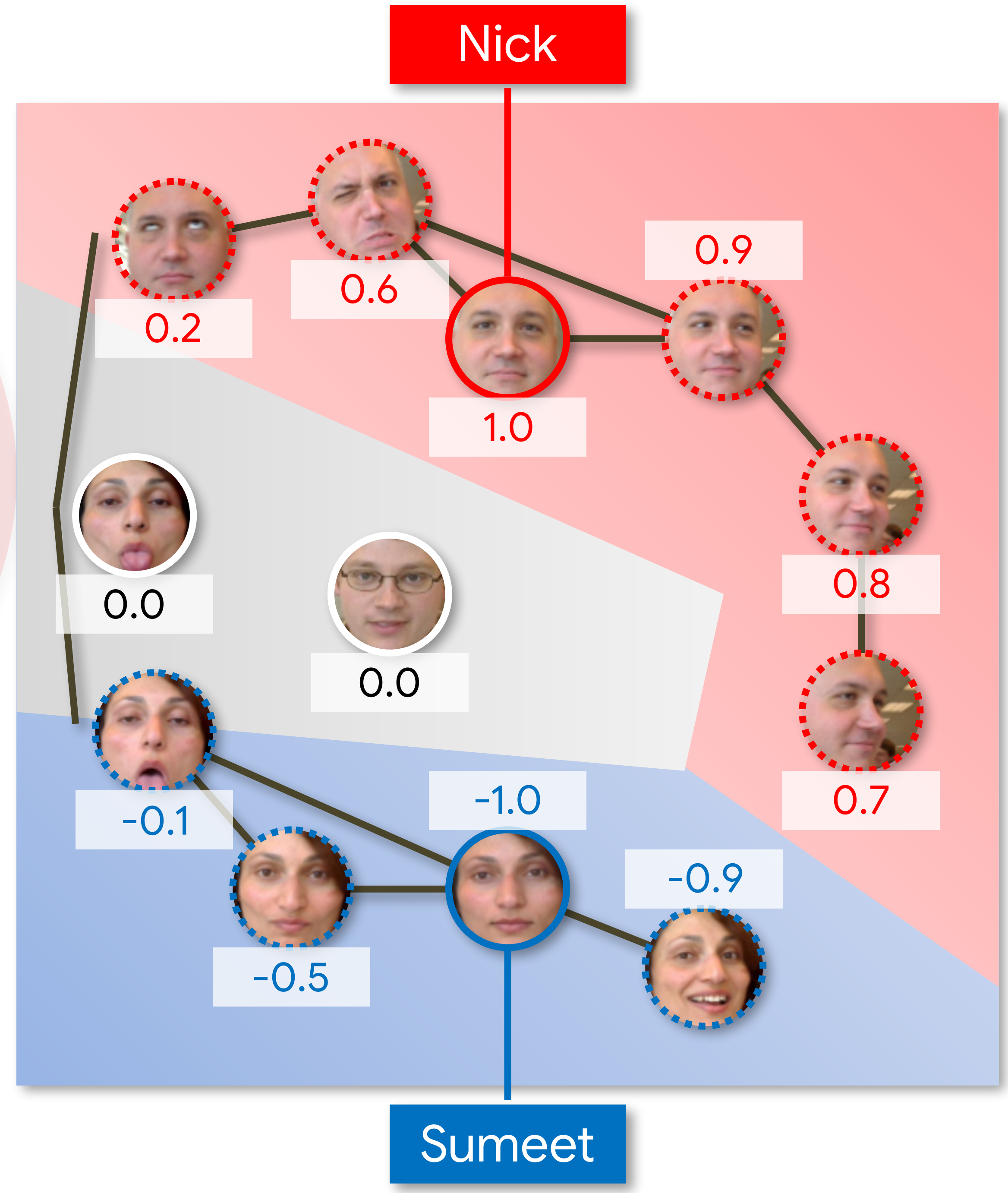
- SIZE and SPEED
- ANOMALIES

$$\mathbf{f}_u = (\mathbf{L}_{uu} + \gamma_g \mathbf{I})^{-1} (\mathbf{W}_{ul} \mathbf{f}_l)$$



$$\ell^{q} = (L^{q}_{uu} + \gamma_g V)^{-1} W^{q}_{ul} \ell_l$$



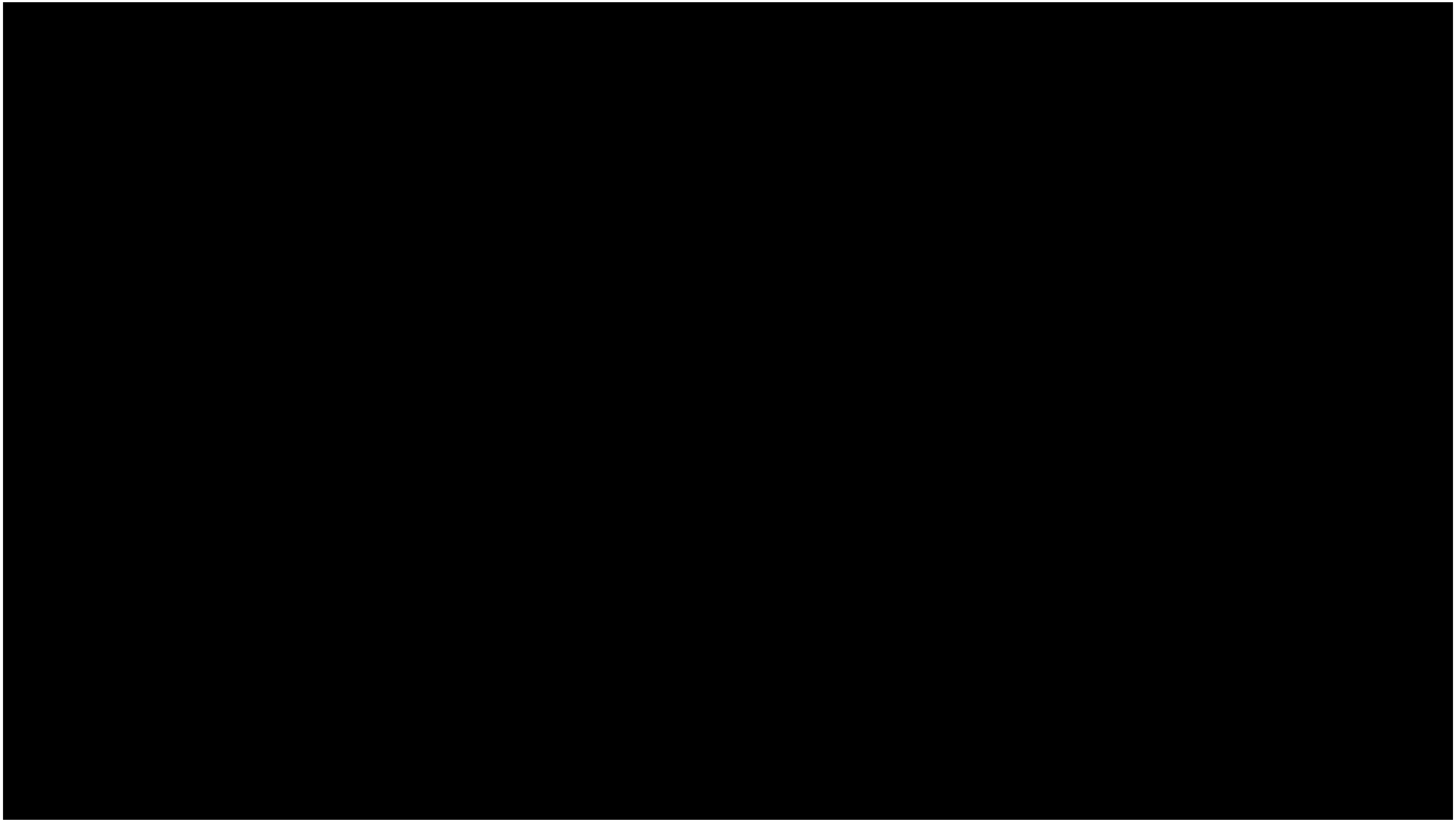


$$\frac{1}{n}\sum_t (\ell^{q}_t[t] - y_t)^2 \le \frac{3}{n}\sum_t (\ell^{*}_t - y_t)^2 + \frac{3}{n}\sum_t (\ell^{o}_t[t] - \ell^{*}_t)^2 + \frac{3}{n}\sum_t (\ell^{q}_t[t] - \ell^{o}_t[t])^2$$

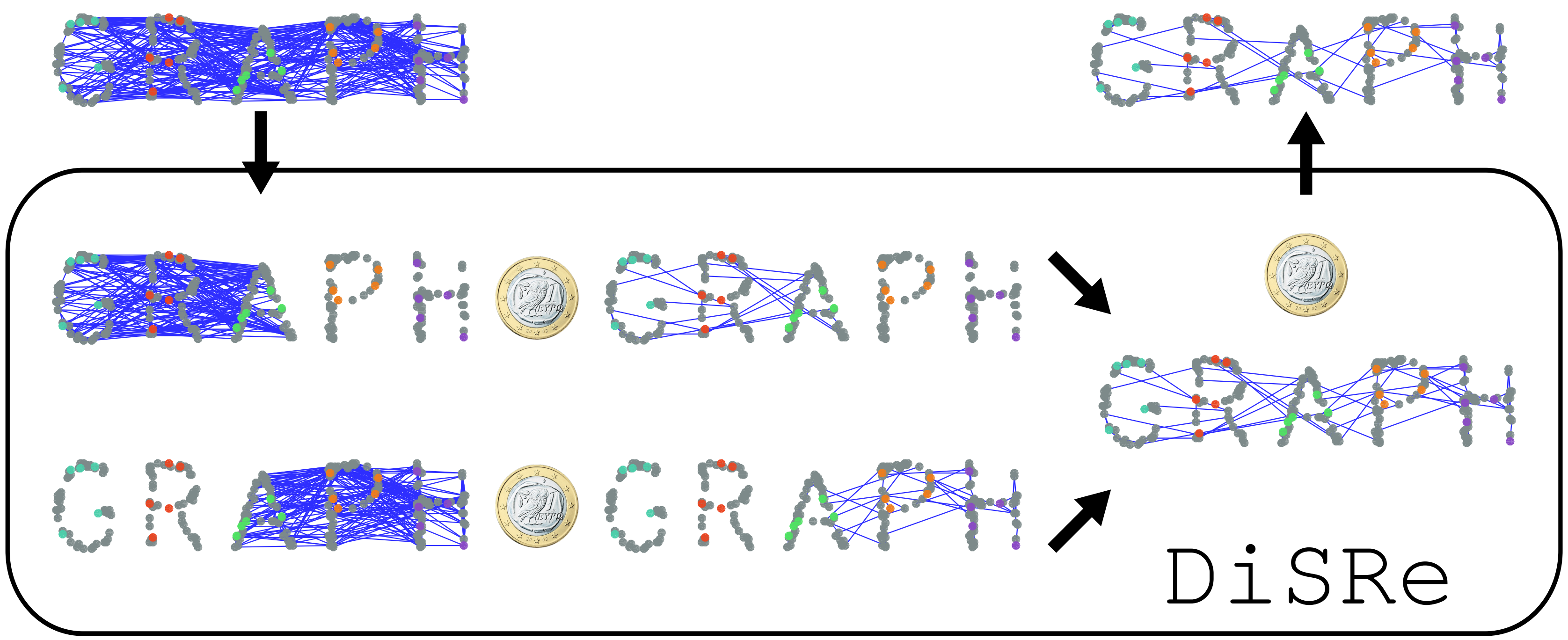
Error of our solution	Offline learning error	Online learning error	Quantization error

Online Semi-Supervised Learning on Quantized Graphs. In Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, California, July 2010.

FACE-RECOGNITION FOR INTEL



SCALE UP!!! 10 YEARS TO BREAK THE N^2



MV, Kveton, Huang, Ting: **Online Semi-Supervised Learning on Quantized Graphs** UAI 2010

Kveton, MV, Rahimi, Huang: **Semi-Supervised Learning with Max-Margin Graph Cuts** AISTATS 2010

Calandriello, Lazaric, MV: **Distributed sequential sampling for kernel matrix approximation** AISTATS **2017**

Calandriello, Lazaric, MV: **Second-order kernel online convex optimization with adaptive sketching**, ICML 2017

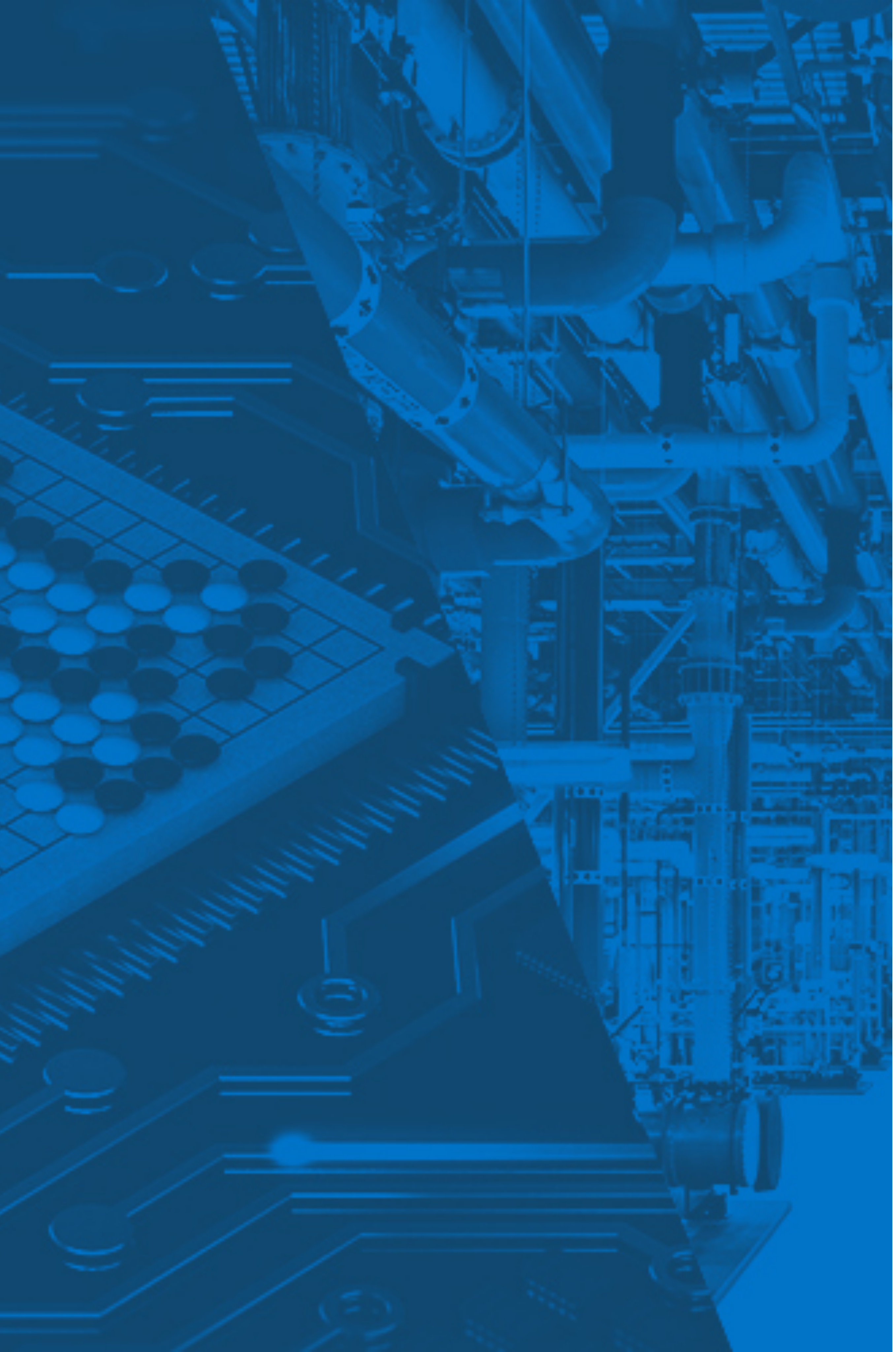
Calandriello, Lazaric, MV: **Efficient second-order online kernel learning with adaptive embedding**, NIPS 2017

Calandriello, Koutis, Lazaric, MV: **Improved large-scale graph learning through ridge spectral sparsification**, ICML 2018

Calandriello, Carratino, Lazaric, MV, Rosasco: **Gaussian process optimization with adaptive sketching: Scalable and no regret**, COLT 2019 and NEGDEP@ICML2019

Dereziński*, Calandriello*, MV: **Exact sampling of determinantal point processes with sublinear time preprocessing**, NEGDEP@ICML2019

code: http://researchers.lille.inria.fr/~valko/hp/publications/squeak.py



COMING UP...

- Sparsification
- Resistance distance
- Leverage scores
- 1-pass is a must
- Online leverage scores
- Negative dependence!
- SQUEAK
- Back to the beginning
  - Spectral sparsifiers
- Back to the future
  - GP-UCB & DPPs

JOINT WORK WITH...

Alessandro Lazaric
FAIR Paris

Ali Rahimi
Google Research

Branislav Kveton
Google Research

Daniel Ting
Tableau Research

Daniele Calandriello
IIT, Genova

Ling Huang
AHI Fintech

Lorenzo Rosasco
IIT, Genova

Luigi Carratino
IIT, Genova

Michał Dereziński
UC Berkeley

Yiannis Koutis
NJIT & CMU

Laplacians and kernels

*Reproducing kernel Hilbert space**

Vector space $\mathcal{H}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$

Feature map $\varphi(\mathbf{x}) : \mathcal{X} \to \mathcal{H}$

Kernel function $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \langle \mathcal{K}(\mathbf{x}, \cdot), \mathcal{K}(\mathbf{x}', \cdot) \rangle_{\mathcal{H}} = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{H}}$

Kernels evaluated at the dataset

Features $\varphi(\mathbf{x}_i) = \phi_i$

Kernel $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle_{\mathcal{H}} = \phi_i^\mathsf{T} \phi_j$

Feature map $\mathbf{\Phi}_n = [\phi_1, \phi_2, \ldots, \phi_n] : \mathbb{R}^n \to \mathcal{H}$

Empirical kernel matrix $\mathbf{K}_n \in \mathbb{R}^{n \times n}$, s.t. $[\mathbf{K}]_{i,j} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$

Column $\mathbf{k}_{[t-1],t} \in \mathbb{R}^{t-1} = \mathbf{\Phi}_{t-1}^\mathsf{T} \phi_t$

Kernel at a point $k_{i,i} \in \mathbb{R} = \phi_t^\mathsf{T} \phi_t$

*Not entering into formal details

Laplacians and kernels

*Reproducing kernel Hilbert space**

Vector space $\mathcal{H}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$

Feature map $\varphi(\mathbf{x}) : \mathcal{X} \to \mathcal{H}$

Kernel function $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \langle \mathcal{K}(\mathbf{x}, \cdot), \mathcal{K}(\mathbf{x}', \cdot) \rangle_{\mathcal{H}} = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{H}}$

Kernels evaluated at the dataset

Features $\varphi(\mathbf{x}_i) = \phi_i$

Kernel $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle_{\mathcal{H}} = \phi_i^{\mathsf{T}} \phi_j$

Feature map $\mathbf{\Phi}_n = [\phi_1, \phi_2, \ldots, \phi_n] : \mathbb{R}^n \to \mathcal{H}$

Empirical kernel matrix $\mathbf{K}_n \in \mathbb{R}^{n \times n}$, s.t. $[\mathbf{K}]_{i,j} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$

Column $\mathbf{k}_{[t-1],t} \in \mathbb{R}^{t-1} = \mathbf{\Phi}_{t-1}^{\mathsf{T}} \phi_t$

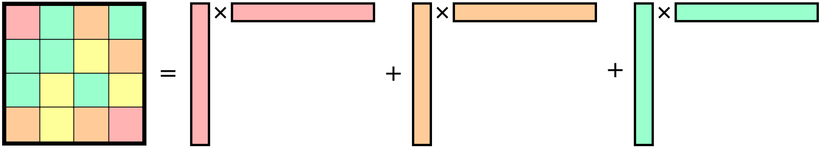
Kernel at a point $k_{i,i} \in \mathbb{R} = \phi_t^{\mathsf{T}} \phi_t$

*Not entering into formal details

Part 1: Kernel Dictionary Learning - The Hammer

Dictionary Learning

Covariance operator: $\mathbf{\Phi}_n\mathbf{\Phi}_n^\mathsf{T} = \sum_{i=1}^n \phi_i\phi_i^\mathsf{T}$



Dictionary learning:* find an accurate representation of the input data as a linear combination of a small set of basic elements (atoms)

*other people may give other definitions...

Singular Value Decomposition – Learning Atoms

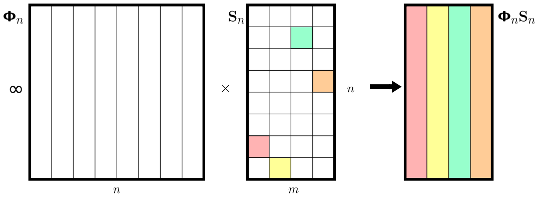
SVD of $\mathbf{\Phi}_n = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^{\mathsf{T}}$ (with rank r)*

$$\mathbf{\Phi}_n\mathbf{\Phi}_n^{\mathsf{T}} = \sum_{i=1}^{n} \phi_i\phi_i^{\mathsf{T}} = \sum_{j=1}^{r} \sigma_i^2 \mathbf{v}_i\mathbf{v}_i^{\mathsf{T}}$$

*With kernels (e.g., Gaussian), r is often as large as n

Dataset Subsampling – Learning Weights

Dictionary $\mathcal{I} = \{(w_j, \phi_j)\}_{j=1}^m$



$$\sum_{j=1}^{m} w_j \phi_j \phi_j^\mathsf{T} = \sum_{j=1}^{m} (\sqrt{w_j}\phi_j)(\sqrt{w_j}\phi_j)^\mathsf{T} = \mathbf{\Phi}_n \mathbf{S}_n \mathbf{S}_n^\mathsf{T} \mathbf{\Phi}_n^\mathsf{T}$$

which points? (ϕ_j) which weights? (w_j)

how many? (m) which guarantees?

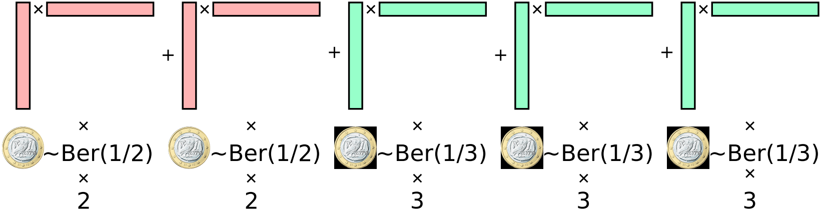
**Remark:* we do not reduce the vectors ϕ_j

Nyström Sampling – Intuition

Sample points $\mathbf{x}_i$ w.p. p_i and add it to $\mathcal{I}$ with weight $\propto 1/p_i$

Nyström Sampling – Intuition

Sample points $\mathbf{x}_i$ w.p. p_i and add it to $\mathcal{I}$ with weight $\propto 1/p_i$



Nyström Sampling – Formally

Input: budget $\overline{q}$, probabilities $\left\{p_i\right\}_i$ *(not necessarily normalized!)*

Init: $\mathcal{I} = \emptyset$

For all $i = 1, \ldots, n$

Draw $q_i \sim \mathcal{B}(p_i, \overline{q})$

Compute weight $w_i = \frac{1}{p_i}\frac{q_i}{\overline{q}}$

Add $(w_i, \mathbf{x}_i)$ to $\mathcal{I}$

Output: $\mathcal{I}$

q_i may be seen as adding q_i copies of $\mathbf{x}_i$ with weight $1/(p_i\overline{q})$

Nyström Sampling – Formally

Lemma

The Nyström estimator ($z_{i,j}$: one out of $\overline{q}$ Bernoulli trials of probability p_i)

$$\mathbf{\Phi}_n \mathbf{S}_n \mathbf{S}_n^\mathsf{T} \mathbf{\Phi}_n^\mathsf{T} = \sum_{i=1}^{n} \sum_{j=1}^{\overline{q}} \frac{1}{p_i} \frac{z_{i,j}}{\overline{q}} \phi_i \phi_i^\mathsf{T}$$

is unbiased

$$\mathbb{E}_{\mathbf{S}_n}\left[\mathbf{\Phi}_n \mathbf{S}_n \mathbf{S}_n^\mathsf{T} \mathbf{\Phi}_n^\mathsf{T}\right] = \mathbf{\Phi}_n \mathbf{\Phi}_n^\mathsf{T}$$

and its dictionary has size

$$\mathbb{P}\left(|\mathcal{I}| \geq 3\overline{q} \sum_{i=1}^{n} p_i\right) \leq \exp\left(-\overline{q} \sum_{i=1}^{n} p_i\right)$$

E.g., uniform sampling $p_i = 1/n$, $|\mathcal{I}| \leq 3\overline{q}$ w.h.p.

Nyström Sampling – Formally

Lemma

The Nyström estimator ($z_{i,j}$: one out of $\overline{q}$ Bernoulli trials of probability p_i)

$$\mathbf{\Phi}_n \mathbf{S}_n \mathbf{S}_n^\mathsf{T} \mathbf{\Phi}_n^\mathsf{T} = \sum_{i=1}^{n} \sum_{j=1}^{\overline{q}} \frac{1}{p_i} \frac{z_{i,j}}{\overline{q}} \phi_i \phi_i^\mathsf{T}$$

is unbiased

$$\mathbb{E}_{\mathbf{S}_n} \left[\mathbf{\Phi}_n \mathbf{S}_n \mathbf{S}_n^\mathsf{T} \mathbf{\Phi}_n^\mathsf{T} \right] = \mathbf{\Phi}_n \mathbf{\Phi}_n^\mathsf{T}$$

and its dictionary has size

$$\mathbb{P}\left(|\mathcal{I}| \geq 3\overline{q} \sum_{i=1}^{n} p_i \right) \leq \exp\left(-\overline{q} \sum_{i=1}^{n} p_i \right)$$

E.g., uniform sampling $p_i = 1/n$, $|\mathcal{I}| \leq 3\overline{q}$ w.h.p.

But is the approximate covariance good?

Reconstruction Guarantees

An (ε, γ)-accurate dictionary $\mathcal{I}$ satisfies*

$$\overbrace{(1-\varepsilon)\boldsymbol{\Phi}_n\boldsymbol{\Phi}_n^\mathsf{T}}^{\text{multiplicative error}} - \overbrace{\varepsilon\gamma\mathbf{I}}^{\text{additive error}} \preceq \boldsymbol{\Phi}_n\mathbf{S}\mathbf{S}^\mathsf{T}\boldsymbol{\Phi}_n^\mathsf{T} \preceq \overbrace{(1+\varepsilon)\boldsymbol{\Phi}_n\boldsymbol{\Phi}_n^\mathsf{T}}^{\text{multiplicative error}} + \overbrace{\varepsilon\gamma\mathbf{I}}^{\text{additive error}}$$

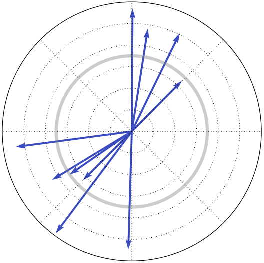
Remarks

- If $\gamma = 0$, all spectrum of $\boldsymbol{\Phi}_n\boldsymbol{\Phi}_n^\mathsf{T}$ is preserved up to $1 \pm \varepsilon$ multiplicative error
- If $\gamma > 0$ only eigenvalues larger than $\varepsilon\gamma$ are preserved

*If a dictionary is accurate in this sense, then it is accurate to build many other things

Nyström Sampling Guarantees – Intuition

Uniform sampling $p_i = 1/n$



$$\mathbf{\Phi}_n\mathbf{\Phi}_n^\mathsf{T} - \mathbf{\Phi}_n\mathbf{S}\mathbf{S}^\mathsf{T}\mathbf{\Phi}_n^\mathsf{T} = \sum_{i=1}^{n}\phi_i\phi_i^\mathsf{T} - \sum_{j=1}^{m} w_j\phi_j\phi_j^\mathsf{T}$$

"Important" directions may have probability too small to be selected
"Redundant" directions may have probability too large to be selected

Ridge Leverage Scores*

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

*leverage scores evaluate the "relevance" of a point in statistics

Ridge Leverage Scores*

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

RLS capture "soft" orthogonality

- If all ϕ_i are *orthogonal*

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

- If all ϕ_i are *collinear*

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

*leverage scores evaluate the "relevance" of a point in statistics

Ridge Leverage Scores*

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

RLS capture "soft" orthogonality

- If all ϕ_i are *orthogonal*

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

- If all ϕ_i are *collinear*

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

Given $\mathbf{\Phi}_{t-1}$, adding columns reduce previous RLS

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

*leverage scores evaluate the "relevance" of a point in statistics

Ridge Leverage Scores*

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

RLS capture "soft" orthogonality

- If all ϕ_i are *orthogonal*

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

- If all ϕ_i are *collinear*

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

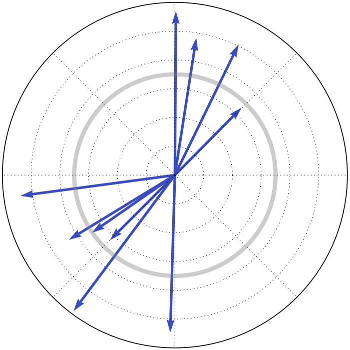
Given $\mathbf{\Phi}_{t-1}$, adding columns reduce previous RLS

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

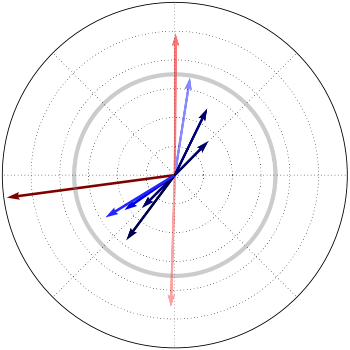
RLS decrease with γ

*leverage scores evaluate the "relevance" of a point in statistics

Ridge Leverage Scores

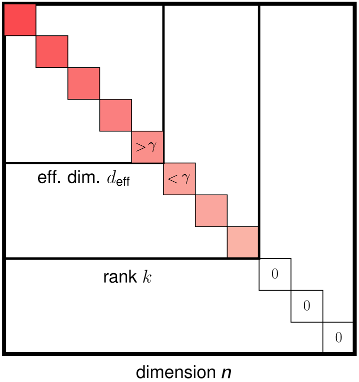


Ridge Leverage Scores



Effective Dimension

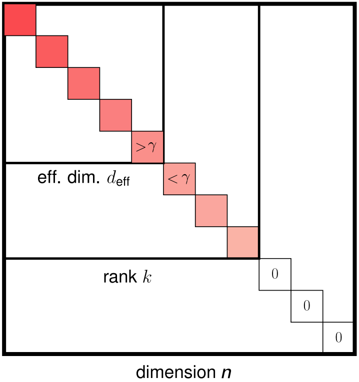
The effective dimension is the number of relevant directions in the data



$$d_{\text{eff}}^{n}(\gamma) = \sum_{i=1}^{n} \tau_{n,i} = \operatorname{Tr}\left(\mathbf{K}_n(\mathbf{K}_n + \gamma \mathbf{I}_n)^{-1}\right) = \sum_{i=1}^{n} \frac{\lambda_i(\mathbf{K}_n)}{\lambda_i(\mathbf{K}_n) + \gamma} \leq \operatorname{Rank}(\mathbf{K}_n)$$

Effective Dimension

The effective dimension is the number of relevant directions in the data



Given $d_{\text{eff}}^{t-1}(\gamma)$, adding a new column to $\mathbf{\Phi}_{t-1}$ may increase $d_{\text{eff}}^{t}(\gamma)$

$$\mathbf{d}_{\text{eff}}^{\mathbf{t}}(\gamma) \geq \mathbf{d}_{\text{eff}}^{\mathbf{t-1}}(\gamma)$$

Nyström Sampling – Formally

Input: budget $\overline{q}$, probabilities $\left\{p_i\right\}_i$ *(not necessarily normalized!)*

Init: $\mathcal{I} = \emptyset$

For all $i = 1, \ldots, n$

Set $p_i = \tau_{n,i}$

Draw $q_i \sim \mathcal{B}(p_i, \overline{q})$

Compute weight $w_i = \frac{1}{p_i}\frac{q_i}{\overline{q}}$

Add $(w_i, \mathbf{x}_i)$ to $\mathcal{I}$

Output: $\mathcal{I}$

q_i may be seen as adding q_i copies of $\mathbf{x}_i$ with weight $1/(p_i\overline{q})$

Oracle RLS Sampling

Theorem (Alaoui and Mahoney, 2014)

Consider the Nyström estimator with oracle RLS sampling $p_i = \tau_{n,i}$. *If*

$$\overline{q} \geq \frac{4\log(n/\delta)}{\varepsilon^2}$$

then $\mathcal{I}$ *is an* (ε, γ)*-accurate dictionary w.p.* $1 - \delta$ *and*

$$|\mathcal{I}| \leq 3\overline{q} d_{\mathit{eff}}^{n}(\gamma)$$

Oracle RLS Sampling

Theorem (Alaoui and Mahoney, 2014)

Consider the Nyström estimator with oracle RLS sampling $p_i = \tau_{n,i}$. *If*

$$\overline{q} \geq \frac{4\log(n/\delta)}{\varepsilon^2}$$

then $\mathcal{I}$ *is an* (ε, γ)*-accurate dictionary w.p.* $1 - \delta$ *and*

$$|\mathcal{I}| \leq 3\overline{q} d_{\text{eff}}^{n}(\gamma)$$

Small and accurate dictionary adapting to the "complexity" of the data

$$d_{\text{eff}}^{n}(\gamma) = \sum_{i=1}^{n} \tau_{i,n} \ll n\tau_{\max}$$

Given the RLS as input

Oracle RLS Sampling

Theorem (Alaoui and Mahoney, 2014)

Consider the Nyström estimator with oracle RLS sampling $p_i = \tau_{n,i}$. *If*

$$\overline{q} \geq \frac{4\log(n/\delta)}{\varepsilon^2}$$

then $\mathcal{I}$ *is an* (ε,γ)*-accurate dictionary w.p.* $1-\delta$ *and*

$$|\mathcal{I}| \leq 3\overline{q}d^n_{\text{eff}}(\gamma)$$

Small and accurate dictionary adapting to the "complexity" of the data

$$d^n_{\text{eff}}(\gamma) = \sum_{i=1}^{n} \tau_{i,n} \ll n\tau_{\max}$$

Given the RLS as input

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

Estimating RLS from a Dictionary

Approximate the kernel matrix directly

$$\tau_{n,i} = \mathbf{e}_{n,i}\mathbf{K}_n^\mathsf{T}(\mathbf{K}_n + \gamma\mathbf{I}_n)^{-1}\mathbf{e}_{n,i}$$
$$\widetilde{\tau}_{n,i} = \mathbf{e}_i^\mathsf{T}\widetilde{\mathbf{K}}_\mathbf{n}(\widetilde{\mathbf{K}}_\mathbf{n} + \gamma\mathbf{I})^{-1}\mathbf{e}_i$$

Estimating RLS from a Dictionary

Approximate the kernel matrix directly

$$\tau_{n,i} = \mathbf{e}_{n,i}\mathbf{K}_n^{\mathsf{T}}(\mathbf{K}_n + \gamma\mathbf{I}_n)^{-1}\mathbf{e}_{n,i}$$
$$\widetilde{\tau}_{n,i} = \mathbf{e}_i^{\mathsf{T}}\widetilde{\mathbf{K}}_{\mathbf{n}}(\widetilde{\mathbf{K}}_{\mathbf{n}} + \gamma\mathbf{I})^{-1}\mathbf{e}_i$$

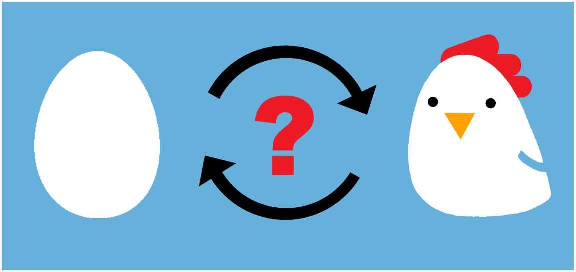
Instead, approximate $\tau_{n,i}$ directly in $\mathcal{H}$

$$\tau_{n,i} = \phi_i^{\mathsf{T}}(\mathbf{\Phi}_n\mathbf{\Phi}_n^{\mathsf{T}} + \gamma\mathbf{I})^{-1}\phi_i$$
$$\widetilde{\tau}_{n,i} = \phi_i^{\mathsf{T}}(\mathbf{\Phi}_n\mathbf{S}_n\mathbf{S}_n^{\mathsf{T}}\mathbf{\Phi}_n^{\mathsf{T}} + \gamma\mathbf{I})^{-1}\phi_i$$

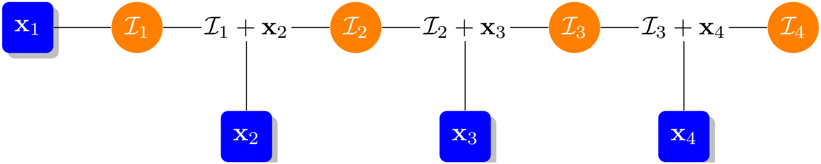
Chicken and Egg problem

Given accurate $\widetilde{\tau}_{n,i}$ $\Rightarrow$ compute accurate dictionary

Given accurate dictionary $\Rightarrow$ compute accurate $\widetilde{\tau}_{n,i}$



Sequential RLS Sampling – Intuition



SQUEAK

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 $\mathcal{I}_{t-1}$, $\mathbf{x}_t$
    Set $\widetilde{\mathbf{p}}_{\mathbf{t,i}} = \min\left\{\widetilde{\tau}_{\mathbf{t,i}},\ \widetilde{\mathbf{p}}_{\mathbf{t-1,i}}\right\}$

DICT-UPDATE:

    Initialize $\mathcal{I}_t = \emptyset$

SHRINK:

    for all $j \in \{1, \ldots, t-1\}$ **do**
        if $q_{t-1,j} \neq 0$ **then**
            $\mathbf{q}_{\mathbf{t,j}} \sim \mathcal{B}(\widetilde{\mathbf{p}}_{\mathbf{t,j}}/\widetilde{\mathbf{p}}_{\mathbf{t-1,j}}, \mathbf{q}_{\mathbf{t-1,j}})$
            Add $(j, \phi_j, q_{t,j}, \widetilde{p}_{t,j})$ to $\mathcal{I}_t$.
        end if
    end for

EXPAND:

    $\mathbf{q}_{\mathbf{t,t}} \sim \mathcal{B}(\widetilde{\mathbf{p}}_{\mathbf{t,t}}, \overline{\mathbf{q}})$
    Add $q_{t,t}$ copies of $(t, \phi_t, q_{t,t}, \widetilde{p}_{t,t})$ to $\mathcal{I}_t$

end for

SQUEAK

Theorem

Consider the Nyström estimator built using SQUEAK *. If*

$$\overline{q} \geq \frac{4\alpha \log(n/\delta)}{\varepsilon^2} \qquad \textit{where } \alpha = \left(\tfrac{1+\varepsilon}{1-\varepsilon}\right),$$

then for all $t = 1, \ldots, n$ $\mathcal{I}_t$ *is an* (ε, γ)*-accurate dictionary w.p.* $1 - \delta$ *and*

$$|\mathcal{I}_t| \leq 3\overline{q} d_{\mathit{eff}}^t(\gamma)$$

SQUEAK

Theorem

Consider the Nyström estimator built using SQUEAK *. If*

$$\overline{q} \geq \frac{4\alpha \log(n/\delta)}{\varepsilon^2} \qquad \text{where } \alpha = \left(\frac{1+\varepsilon}{1-\varepsilon}\right),$$

then for all $t = 1, \ldots, n$ $\mathcal{I}_t$ *is an* (ε, γ)*-accurate dictionary w.p.* $1 - \delta$ *and*

$$|\mathcal{I}_t| \leq 3\overline{q} d_{\mathit{eff}}^t(\gamma)$$

- Accuracy and space/time guarantees
- Anytime 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 $\widetilde{p}_{t,i}$

SQUEAK

Theorem

Consider the Nyström estimator built using SQUEAK *. If*

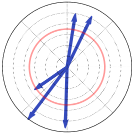
$$\overline{q} \geq \frac{4\alpha \log(n/\delta)}{\varepsilon^2} \qquad \textit{where } \alpha = \left(\frac{1+\varepsilon}{1-\varepsilon}\right),$$

then for all $t = 1, \ldots, n$ $\mathcal{I}_t$ *is an* (ε, γ)*-accurate dictionary w.p.* $1 - \delta$ *and*

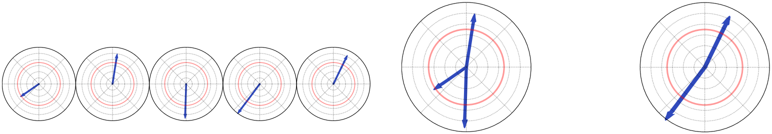
$$|\mathcal{I}_t| \leq 3\overline{q} d^t_{\mathit{eff}}(\gamma)$$

- 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 ~~$\mathcal{O}(\mathbf{n}^3)$~~ $\Rightarrow \mathcal{O}(\mathbf{n}|\mathcal{I}_\mathbf{n}|^3) \leq \widetilde{\mathcal{O}}(\mathbf{n}\mathbf{d}^\mathbf{n}_{\text{eff}}(\gamma)^3)$
- Store points directly in the dictionary
  - ↳ $\widetilde{\mathcal{O}}(\mathbf{d}^\mathbf{n}_{\text{eff}}(\gamma)^2 + \mathbf{d}^\mathbf{n}_{\text{eff}}(\gamma)\mathbf{D})$ space "constant" in n
    - ↳ single pass over the dataset (streaming)

Sequential RLS sampling – Distributed Version

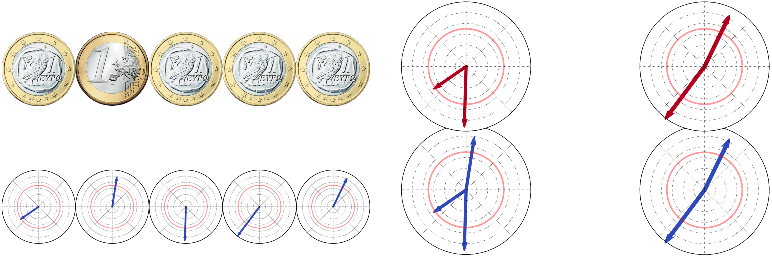


Sequential RLS sampling – Distributed Version



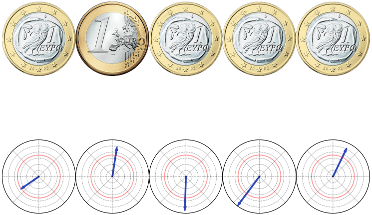
Sequential RLS sampling – Distributed Version

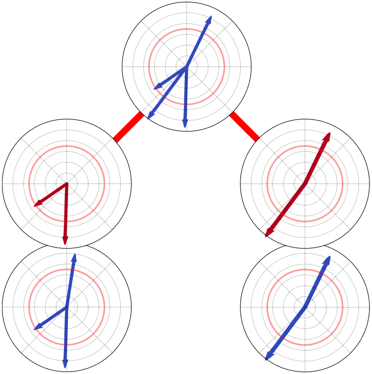
$\widetilde{p}_{1,i} \propto \widetilde{\tau}_{1,i},$
$z_{1,i} = \mathbb{I}\{Ber(\widetilde{p}_{1,i})\}$



Sequential RLS sampling – Distributed Version

$\widetilde{p}_{1,i} \propto \widetilde{\tau}_{1,i}$,
$z_{1,i} = \mathbb{I}\{Ber(\widetilde{p}_{1,i})\}$

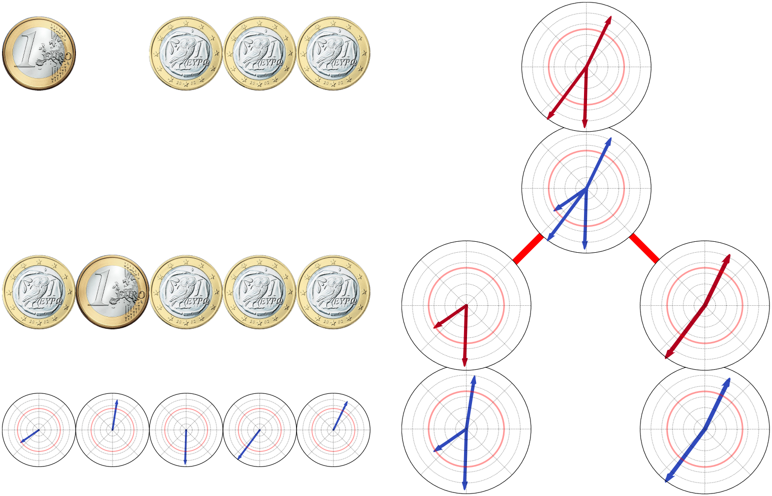




Sequential RLS sampling – Distributed Version

$$\widetilde{p}_{1,i} \propto \widetilde{\tau}_{1,i}, \quad z_{1,i} = \mathbb{I}\{Ber(\widetilde{p}_{1,i})\}$$

$$\widetilde{p}_{2,i} \propto \widetilde{\tau}_{2,i}, \quad z_{2,i} = \mathbb{I}\left\{ Ber\left(\frac{\widetilde{p}_{2,i}}{\widetilde{p}_{1,i}}\right)\right\} z_{1,i}$$

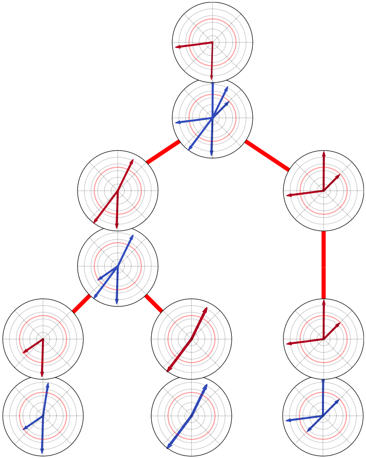


Sequential RLS sampling – Distributed Version

$\widetilde{p}_{1,i} \propto \widetilde{\tau}_{1,i},$
$z_{1,i} = \mathbb{I}\{Ber(\widetilde{p}_{1,i})\}$

$\widetilde{p}_{2,i} \propto \widetilde{\tau}_{2,i}$
$z_{2,i} = \mathbb{I}\left\{Ber\left(\frac{\widetilde{p}_{2,i}}{\widetilde{p}_{1,i}}\right)\right\} z_{1,i}$

$\widetilde{p}_{3,i} \propto \widetilde{\tau}_{3,i}$
$z_{3,i} = \mathbb{I}\left\{Ber\left(\frac{\widetilde{p}_{3,i}}{\widetilde{p}_{2,i}}\right)\right\} z_{2,i}$



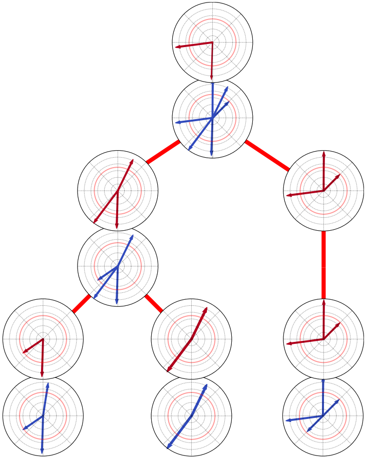
Sequential RLS sampling – Distributed Version

$\widetilde{p}_{1,i} \propto \widetilde{\tau}_{1,i},$
$z_{1,i} = \mathbb{I}\{Ber(\widetilde{p}_{1,i})\}$

$\widetilde{p}_{2,i} \propto \widetilde{\tau}_{2,i}$
$z_{2,i} = \mathbb{I}\left\{Ber\left(\frac{\widetilde{p}_{2,i}}{\widetilde{p}_{1,i}}\right)\right\} z_{1,i}$

$\widetilde{p}_{3,i} \propto \widetilde{\tau}_{3,i}$
$z_{3,i} = \mathbb{I}\left\{Ber\left(\frac{\widetilde{p}_{3,i}}{\widetilde{p}_{2,i}}\right)\right\} z_{2,i}$

- Dataset is distributed over multiple machines



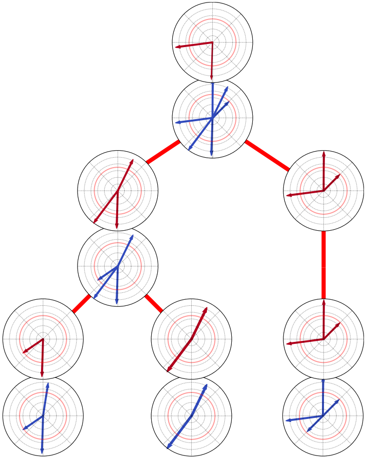
Sequential RLS sampling – Distributed Version

$\widetilde{p}_{1,i} \propto \widetilde{\tau}_{1,i},$
$z_{1,i} = \mathbb{I}\{Ber(\widetilde{p}_{1,i})\}$

$\widetilde{p}_{2,i} \propto \widetilde{\tau}_{2,i}$
$z_{2,i} = \mathbb{I}\left\{Ber\left(\frac{\widetilde{p}_{2,i}}{\widetilde{p}_{1,i}}\right)\right\} z_{1,i}$

$\widetilde{p}_{3,i} \propto \widetilde{\tau}_{3,i}$
$z_{3,i} = \mathbb{I}\left\{Ber\left(\frac{\widetilde{p}_{3,i}}{\widetilde{p}_{2,i}}\right)\right\} z_{2,i}$

- Dataset is distributed over multiple machines
- Communication is limited to samples in the dictionaries



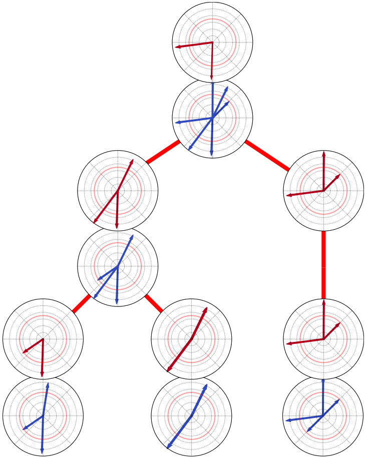
Sequential RLS sampling – Distributed Version

$\widetilde{p}_{1,i} \propto \widetilde{\tau}_{1,i},$
$z_{1,i} = \mathbb{I}\{Ber(\widetilde{p}_{1,i})\}$

$\widetilde{p}_{2,i} \propto \widetilde{\tau}_{2,i}$
$z_{2,i} = \mathbb{I}\left\{Ber\left(\frac{\widetilde{p}_{2,i}}{\widetilde{p}_{1,i}}\right)\right\} z_{1,i}$

$\widetilde{p}_{3,i} \propto \widetilde{\tau}_{3,i}$
$z_{3,i} = \mathbb{I}\left\{Ber\left(\frac{\widetilde{p}_{3,i}}{\widetilde{p}_{2,i}}\right)\right\} z_{2,i}$

- Dataset is distributed over multiple machines
- Communication is limited to samples in the dictionaries
- Runtime depends on merge tree



Comparison

 = oracle, $\mu(\gamma) = \max_i \tau_{n,i}(\gamma) \leq 1/\gamma$ regularized coherence

	$\widetilde{\mathcal{O}}$(Runtime)	$\mathcal{O}(\lvert\mathcal{I}_n\rvert)$	Passes
Bach, 2013 (Uniform)	$n\mu(\gamma)$ + 	$n\mu(\gamma)$	1

Comparison

 = oracle, $\mu(\gamma) = \max_i \tau_{n,i}(\gamma) \leq 1/\gamma$ regularized coherence

	$\widetilde{\mathcal{O}}$(Runtime)	$\mathcal{O}(\vert\mathcal{I}_n\vert)$	Passes
Bach, 2013 (Uniform)	$n\mu(\gamma) +$ 	$n\mu(\gamma)$	1
Oracle RLS sampling	$n +$ 	$d_{\text{eff}}^n(\gamma)\log(n)$	Many

Comparison

= oracle, $\mu(\gamma) = \max_i \tau_{n,i}(\gamma) \leq 1/\gamma$ regularized coherence

	$\widetilde{\mathcal{O}}$(Runtime)	$\mathcal{O}(\lvert\mathcal{I}_n\rvert)$	Passes
Bach, 2013 (Uniform)	$n\mu(\gamma) +$	$n\mu(\gamma)$	1
Oracle RLS sampling	$n +$	$d_{\text{eff}}^n(\gamma)\log(n)$	Many
Exact RLS sampling	n^3	$d_{\text{eff}}^n(\gamma)\log(n)$	Many

Comparison

 = oracle, $\mu(\gamma) = \max_i \tau_{n,i}(\gamma) \leq 1/\gamma$ regularized coherence

	$\widetilde{\mathcal{O}}$(Runtime)	$\mathcal{O}(\vert\mathcal{I}_n\vert)$	Passes
Bach, 2013 (Uniform)	$n\mu(\gamma) +$ 	$n\mu(\gamma)$	1
Oracle RLS sampling	$n +$ 	$d_{\text{eff}}^n(\gamma)\log(n)$	Many
Exact RLS sampling	n^3	$d_{\text{eff}}^n(\gamma)\log(n)$	Many
Alaoui and Mahoney, 2015	$n^3\mu(\gamma)^2$	$n\mu(\gamma) + d_{\text{eff}}^n(\gamma)\log(n)$	3

Comparison

 = oracle, $\mu(\gamma) = \max_i \tau_{n,i}(\gamma) \leq 1/\gamma$ regularized coherence

	$\widetilde{\mathcal{O}}$(Runtime)	$\mathcal{O}(\lvert\mathcal{I}_n\rvert)$	Passes
Bach, 2013 (Uniform)	$n\mu(\gamma)$ + 	$n\mu(\gamma)$	1
Oracle RLS sampling	n + 	$d_{\text{eff}}^n(\gamma)\log(n)$	Many
Exact RLS sampling	n^3	$d_{\text{eff}}^n(\gamma)\log(n)$	Many
Alaoui and Mahoney, 2015	$n^3\mu(\gamma)^2$	$n\mu(\gamma) + d_{\text{eff}}^n(\gamma)\log(n)$	3
SQUEAK Calandriello et al., 2017a	$(n/k)d_{\text{eff}}^n(\gamma)^3$	$d_{\text{eff}}^n(\gamma)\log(n)$	1

Comparison

 = oracle, $\mu(\gamma) = \max_i \tau_{n,i}(\gamma) \leq 1/\gamma$ regularized coherence

	$\widetilde{\mathcal{O}}$(Runtime)	$\mathcal{O}(\lvert\mathcal{I}_n\rvert)$	Passes
Bach, 2013 (Uniform)	$n\mu(\gamma) +$ 	$n\mu(\gamma)$	1
Oracle RLS sampling	$n +$ 	$d_{\text{eff}}^n(\gamma)\log(n)$	Many
Exact RLS sampling	n^3	$d_{\text{eff}}^n(\gamma)\log(n)$	Many
Alaoui and Mahoney, 2015	$n^3\mu(\gamma)^2$	$n\mu(\gamma) + d_{\text{eff}}^n(\gamma)\log(n)$	3
SQUEAK Calandriello et al., 2017a	$(n/k)d_{\text{eff}}^n(\gamma)^3$	$d_{\text{eff}}^n(\gamma)\log(n)$	1
KORS Calandriello et al., 2017b	$nd_{\text{eff}}^n(\gamma)^2$	$d_{\text{eff}}^n(\gamma)\log^2(n)$	1

Comparison

 = oracle, $\mu(\gamma) = \max_i \tau_{n,i}(\gamma) \leq 1/\gamma$ regularized coherence

	$\widetilde{\mathcal{O}}$(Runtime)	$\mathcal{O}(\lvert\mathcal{I}_n\rvert)$	Passes
Bach, 2013 (Uniform)	$n\mu(\gamma) +$ 	$n\mu(\gamma)$	1
Oracle RLS sampling	$n +$ 	$d_{\text{eff}}^n(\gamma)\log(n)$	Many
Exact RLS sampling	n^3	$d_{\text{eff}}^n(\gamma)\log(n)$	Many
Alaoui and Mahoney, 2015	$n^3\mu(\gamma)^2$	$n\mu(\gamma) + d_{\text{eff}}^n(\gamma)\log(n)$	3
SQUEAK Calandriello et al., 2017a	$(n/k)d_{\text{eff}}^n(\gamma)^3$	$d_{\text{eff}}^n(\gamma)\log(n)$	1
KORS Calandriello et al., 2017b	$nd_{\text{eff}}^n(\gamma)^2$	$d_{\text{eff}}^n(\gamma)\log^2(n)$	1
Musco and Musco, 2017	$nd_{\text{eff}}^n(\gamma)^2$	$d_{\text{eff}}^n(\gamma)\log(n)$	$\log(n)$

Recap

Construct a small, provably accurate dictionary in near-linear time

SQUEAK and DISQUEAK

Sub-linear time using multiple machines

Final dictionary can be updated if new samples arrive

Recap

Construct a small, provably accurate dictionary in near-linear time

SQUEAK and DISQUEAK

- Sub-linear time using multiple machines
- Final dictionary can be updated if new samples arrive
- Novel analysis, potentially useful for general importance sampling

Recap

Construct a small, provably accurate dictionary in near-linear time

SQUEAK and DISQUEAK

- Sub-linear time using multiple machines
- Final dictionary can be updated if new samples arrive
- Novel analysis, potentially useful for general importance sampling

Future work

- Experiments
  - ↳ Easy to implement: distributed task queue
    Preliminary results promising, easily scales to 1M+ samples

Recap

Construct a small, provably accurate dictionary in near-linear time

SQUEAK and DISQUEAK

- Sub-linear time using multiple machines
- Final dictionary can be updated if new samples arrive
- Novel analysis, potentially useful for general importance sampling

Future work

- Experiments
  - ↳ Easy to implement: distributed task queue
    Preliminary results promising, easily scales to 1M+ samples
- Beyond passive processing: SQUEAK for active learning

Part 2: Applications - The Nails

Kernel Regression

Kernel ridge regression

$$\widehat{\boldsymbol{\omega}}_n = \arg\min_{\boldsymbol{\omega}} \|\mathbf{y}_n - \mathbf{K}_n\boldsymbol{\omega}\|^2 + \lambda\|\boldsymbol{\omega}\|^2 = (\mathbf{K}_n + \lambda\mathbf{I})^{-1}\mathbf{y}_n$$

Regularized Nyström kernel approximation

$$\widetilde{\boldsymbol{K}}_n = \mathbf{K}_n\mathbf{S}_n(\mathbf{S}_n^\mathsf{T}\mathbf{K}_n\mathbf{S}_n + \gamma\mathbf{I}_{\mathcal{I}_n})^{-1}\mathbf{S}_n^\mathsf{T}\mathbf{K}_n = \boldsymbol{\Phi}_n^\mathsf{T}\boldsymbol{\Phi}_n\mathbf{S}_n(\mathbf{S}_n^\mathsf{T}\mathbf{K}_n\mathbf{S}_n + \gamma\mathbf{I}_{\mathcal{I}_n})^{-1}\mathbf{S}_n^\mathsf{T}\boldsymbol{\Phi}_n^\mathsf{T}\boldsymbol{\Phi}_n$$

$$\begin{aligned}\widetilde{\boldsymbol{\omega}}_n &= (\widetilde{\boldsymbol{K}}_n + \lambda\mathbf{I}_n)^{-1}\mathbf{y}_n \\ &= \frac{1}{\lambda}\left(\mathbf{y}_n - \mathbf{K}_n\mathbf{S}_n(\mathbf{S}_n^\mathsf{T}\mathbf{K}_n\mathbf{S}_n + \lambda(\mathbf{S}_n^\mathsf{T}\mathbf{K}_n\mathbf{S}_n + \gamma\mathbf{I}_{\mathcal{I}_n}))^{-1}\mathbf{S}_n^\mathsf{T}\mathbf{K}_n\mathbf{y}_n\right)\end{aligned}$$

Efficient computation

- Construct the matrix $\mathcal{O}(n|\mathcal{I}_n|^2)$
- Invert the matrix $\mathcal{O}(|\mathcal{I}_n|^3)$
- Time ~~$\mathcal{O}(n^3)$~~ $\Rightarrow \mathcal{O}(n|\mathcal{I}_n|^2 + |\mathcal{I}_n|^3)$
- Space ~~$\mathcal{O}(n^2)$~~ $\Rightarrow \mathcal{O}(n|\mathcal{I}_n|)$

Kernel Regression

Theorem (Alaoui and Mahoney, 2014)

Consider the regularized Nyström kernel approximation generated by an (ε, γ)-accurate dictionary. Then

$$\mathbf{0} \preceq \mathbf{K}_n - \widetilde{\mathbf{K}}_n \preceq \frac{\gamma}{1-\varepsilon}\mathbf{K}_n(\mathbf{K}_n + \gamma \mathbf{I}_n)^{-1} \preceq \frac{\gamma}{1-\varepsilon}\mathbf{I}_n.$$

and

$$\mathcal{R}_{\mathcal{D}}(\widetilde{\boldsymbol{\omega}}) \leq \left(1 + \frac{\gamma}{\lambda}\frac{\varepsilon}{1-\varepsilon}\right)^2 \mathcal{R}_{\mathcal{D}}(\widehat{\boldsymbol{\omega}}),$$

If $\gamma = \lambda$ (i.e., additive error of the same order of the regularization)

- SQUEAK can be used to computed $\widetilde{\boldsymbol{\omega}}$ in $\mathcal{O}(nd_{\text{eff}}^n(\lambda)^2 + d_{\text{eff}}^n(\lambda)^3)$ time
- with a prediction error $1/(1-\varepsilon)^2$ larger than the exact solution

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 $f_{\mathbf{w}_t}(\mathbf{x}_t) = \varphi(\mathbf{x}_t)^\mathsf{T}\mathbf{w}_t$,
3. the adversary reveals the curved loss ℓ_t,
4. the learner suffers $\ell_t(\phi_t^\mathsf{T}\mathbf{w}_t)$ 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 $f_{\mathbf{w}_t}(\mathbf{x}_t) = \varphi(\mathbf{x}_t)^\mathsf{T}\mathbf{w}_t$,
3. the adversary reveals the curved loss ℓ_t,
4. the learner suffers $\ell_t(\phi_t^\mathsf{T}\mathbf{w}_t)$ and observes the associated gradient $\mathbf{g}_t$.

Kernel flexible but curse of kernelization

t parameters $\Rightarrow$ $\mathcal{O}(t)$ per-step prediction cost

$\mathbf{g}_t = \ell_t'(\phi_t^\mathsf{T}\mathbf{w}_t)\phi_t := \dot{g}_t\phi_t$

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 $f_{\mathbf{w}_t}(\mathbf{x}_t) = \varphi(\mathbf{x}_t)^\mathsf{T}\mathbf{w}_t$,
3. the adversary reveals the curved loss ℓ_t,
4. the learner suffers $\ell_t(\phi_t^\mathsf{T}\mathbf{w}_t)$ and observes the associated gradient $\mathbf{g}_t$.

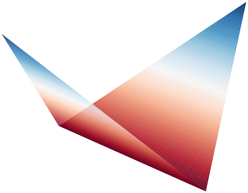
Kernel flexible but curse of kernelization

t parameters $\Rightarrow$ $\mathcal{O}(t)$ per-step prediction cost

$\mathbf{g}_t = \ell_t'(\phi_t^\mathsf{T}\mathbf{w}_t)\phi_t := \dot{g}_t\phi_t$

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

OGD and losses

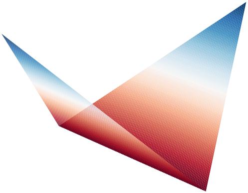


convex

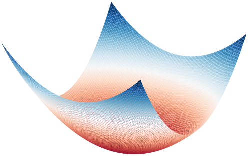
First order (GD) [Kivinen et al., 2004; Zinkevich, 2003]

$\sqrt{T}$ regret, $\mathcal{O}(d)/\mathcal{O}(t)$ time/space per-step

OGD and losses



convex

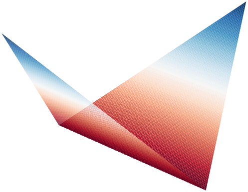


strongly convex

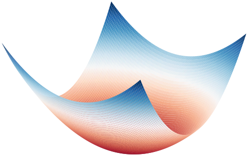
First order (GD) [Kivinen et al., 2004; Zinkevich, 2003]
$\sqrt{T}$ regret, $\mathcal{O}(d)/\mathcal{O}(t)$ time/space per-step

First order (GD) [Hazan et al., 2008]
$\log(T)$ regret,

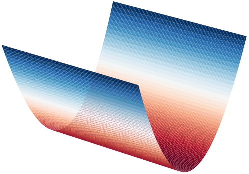
OGD and losses



convex



strongly convex



First order (GD) [Kivinen et al., 2004; Zinkevich, 2003]

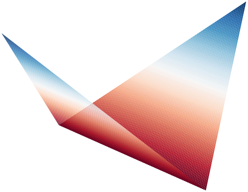
$\sqrt{T}$ regret, $\mathcal{O}(d)/\mathcal{O}(t)$ time/space per-step

First order (GD) [Hazan et al., 2008]

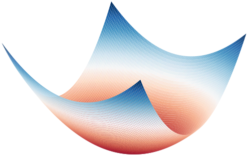
$\log(T)$ regret, but often not satisfied in practice

↳(e.g. $(y_t - \phi_t^\mathsf{T}\mathbf{w}_t)^2$)

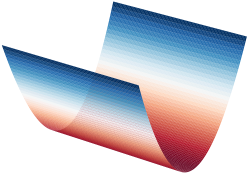
OGD and losses



convex



strongly convex



σ**-curved**

Second order (Newton-like) [Hazan et al., 2006; Zhdanov and Kalnishkan, 2010]
$\log(T)$ regret, $\mathcal{O}(d^2)/\mathcal{O}(t^2)$ time/space per-step

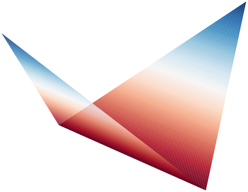
OGD and losses

convex **strongly convex** **σ-curved**

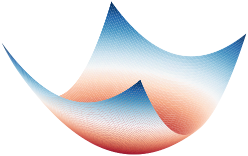
Second order (Newton-like) [Hazan et al., 2006; Zhdanov and Kalnishkan, 2010]
$\log(T)$ regret, $\mathcal{O}(d^2)/\mathcal{O}(t^2)$ time/space per-step

Weaker than strong convexity

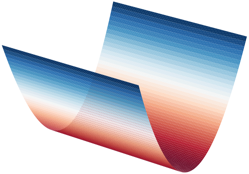
OGD and losses



convex



strongly convex



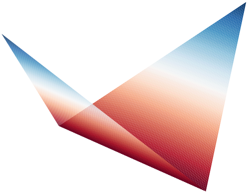
σ**-curved**

Second order (Newton-like) [Hazan et al., 2006; Zhdanov and Kalnishkan, 2010]
$\log(T)$ regret, $\mathcal{O}(d^2)/\mathcal{O}(t^2)$ time/space per-step

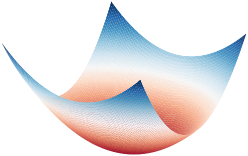
Weaker than strong convexity

Satisfied by exp-concave losses:
↳squared loss, squared hinge-loss, logistic loss

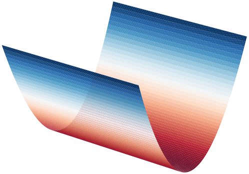
OGD and losses



convex



strongly convex



σ-curved

Second order (Newton-like) [Hazan et al., 2006; Zhdanov and Kalnishkan, 2010]
$\log(T)$ regret, $\mathcal{O}(d^2)/\mathcal{O}(t^2)$ time/space per-step

Weaker than strong convexity

Satisfied by exp-concave losses:
↳squared loss, squared hinge-loss, logistic loss

Assumptions:
ℓ_t are σ-curved and $|\ell_t'(z)| \leq L$ whenever $|z| \leq C$ (scalar Lipschitz)

Second-Order OKL (Kernel Online Newton Step)

Second-Order Gradient Descent

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

Second-Order OKL (Kernel Online Newton Step)

Second-Order Gradient Descent

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

Regret [Hazan et al., 2006; Luo et al., 2016]

$$R(\mathbf{w}^*) \leq \overbrace{\alpha \|\mathbf{w}^* - \mathbf{w_0}\|_2^2}^{\text{initial error}} + \mathcal{O}\left(\sum_{t=1}^{T} \mathbf{g}_t^\mathsf{T} (\mathbf{G}_t \mathbf{G}_t^\mathsf{T} + \alpha \mathbf{I})^{-1} \mathbf{g}_t\right)$$

Second-Order OKL (Kernel Online Newton Step)

Second-Order Gradient Descent

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

Regret [Hazan et al., 2006; Luo et al., 2016]

$$R(\mathbf{w}^*) \leq \overbrace{\alpha \|\mathbf{w}^* - \mathbf{w}_0\|_2^2}^{\text{initial error}} + \mathcal{O}\left(\sum_{t=1}^{T} \mathbf{g}_t^\mathsf{T} (\mathbf{G}_t \mathbf{G}_t^\mathsf{T} + \alpha \mathbf{I})^{-1} \mathbf{g}_t \right)$$

$$\leq \alpha \|\mathbf{w}^* - \mathbf{w}_0\|^2 + \mathcal{O}\left(L \sum_{t=1}^{T} \phi_t^\mathsf{T} (\boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\mathsf{T} + \alpha \mathbf{I})^{-1} \phi_t \right)$$

Second-Order OKL (Kernel Online Newton Step)

Second-Order Gradient Descent

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

Regret [Hazan et al., 2006; Luo et al., 2016]

$$R(\mathbf{w}^*) \leq \overbrace{\alpha \|\mathbf{w}^* - \mathbf{w}_0\|_2^2}^{\text{initial error}} + \mathcal{O}\left(\sum_{t=1}^{T} \mathbf{g}_t^\mathsf{T} (\mathbf{G}_t \mathbf{G}_t^\mathsf{T} + \alpha \mathbf{I})^{-1} \mathbf{g}_t \right)$$

$$\leq \alpha \|\mathbf{w}^* - \mathbf{w}_0\|^2 + \mathcal{O}\left(L \overbrace{\sum_{t=1}^{T} \phi_t^\mathsf{T} (\mathbf{\Phi}_t \mathbf{\Phi}_t^\mathsf{T} + \alpha \mathbf{I})^{-1} \phi_t}^{\text{online effective dimension}} \right)$$

Second-Order OKL (Kernel Online Newton Step)

Second-Order Gradient Descent

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

Regret [Hazan et al., 2006; Luo et al., 2016]

$$R(\mathbf{w}^*) \leq \overbrace{\alpha \|\mathbf{w}^* - \mathbf{w}_0\|_2^2}^{\text{initial error}} + \mathcal{O}\left(\sum_{t=1}^{T} \mathbf{g}_t^\mathsf{T} (\mathbf{G}_t \mathbf{G}_t^\mathsf{T} + \alpha \mathbf{I})^{-1} \mathbf{g}_t\right)$$

$$\leq \alpha \|\mathbf{w}^* - \mathbf{w}_0\|^2 + \mathcal{O}\left(L \overbrace{\sum_{t=1}^{T} \phi_t^\mathsf{T} (\mathbf{\Phi}_t \mathbf{\Phi}_t^\mathsf{T} + \alpha \mathbf{I})^{-1} \phi_t}^{\text{online effective dimension}}\right)$$

$$\leq \alpha \|\mathbf{w}^* - \mathbf{w}_0\|^2 + \mathcal{O}(\log \mathrm{Det}(\mathbf{K}_T / \alpha + \mathbf{I}_n))$$

Second-Order OKL (Kernel Online Newton Step)

Second-Order Gradient Descent

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

Regret [Hazan et al., 2006; Luo et al., 2016]

$$R(\mathbf{w}^*) \leq \overbrace{\alpha \|\mathbf{w}^* - \mathbf{w}_0\|_2^2}^{\text{initial error}} + \mathcal{O}\left(\sum_{t=1}^{T} \mathbf{g}_t^\mathsf{T} (\mathbf{G}_t \mathbf{G}_t^\mathsf{T} + \alpha \mathbf{I})^{-1} \mathbf{g}_t \right)$$

$$\leq \alpha \|\mathbf{w}^* - \mathbf{w}_0\|^2 + \mathcal{O}\left(L \overbrace{\sum_{t=1}^{T} \phi_t^\mathsf{T} (\mathbf{\Phi}_t \mathbf{\Phi}_t^\mathsf{T} + \alpha \mathbf{I})^{-1} \phi_t}^{\text{online effective dimension}} \right)$$

$$\leq \alpha \|\mathbf{w}^* - \mathbf{w}_0\|^2 + \mathcal{O}(\log \operatorname{Det}(\mathbf{K}_T/\alpha + \mathbf{I}_n))$$

$$\leq \alpha \|\mathbf{w}^* - \mathbf{w}_0\|^2 + \mathcal{O}(d_{\text{eff}}^T(\alpha) \log(T))$$ [Calandriello et al., 2017b]

Effective Dimension in online learning

$$R(\mathbf{w}^*) \leq \alpha \|\mathbf{w}^* - \mathbf{w}_0\|^2 + \mathcal{O}(d_{\text{eff}}^T(\alpha) \log(T))$$

$d_{\text{eff}}^T(\alpha)$ number of relevant orthogonal directions played by the adversary.

Every new orthogonal direction causes some regret.
↳if it is played often enough (i.e., $\geq \alpha/(L\sigma)$)

Effective Dimension in online learning

$$R(\mathbf{w}^*) \leq \alpha\|\mathbf{w}^* - \mathbf{w}_0\|^2 + \mathcal{O}(d_{\text{eff}}^T(\alpha)\log(T))$$

$d_{\text{eff}}^T(\alpha)$ number of relevant orthogonal directions played by the adversary.

Every new orthogonal direction causes some regret.
↳if it is played often enough (i.e., $\geq \alpha/(L\sigma)$)

If all ϕ_t are orthogonal

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

and

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

Effective Dimension in online learning

$$R(\mathbf{w}^*) \leq \alpha\|\mathbf{w}^* - \mathbf{w}_0\|^2 + \mathcal{O}(d_{\text{eff}}^T(\alpha)\log(T))$$

$d_{\text{eff}}^T(\alpha)$ number of relevant orthogonal directions played by the adversary.

Every new orthogonal direction causes some regret.
↳if it is played often enough (i.e., $\geq \alpha/(L\sigma)$)

If all ϕ_t are orthogonal

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

and

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

If ϕ_t from finite subspace

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

is constant in T and

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

Approximating KONS

KONS: $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret

↳ large $\mathcal{H} \Rightarrow \mathcal{O}(t)$ prediction $\phi_t^{\mathsf{T}}\mathbf{w}_t$, $\mathcal{O}(t^2)$ updates $\mathbf{g}_t - \mathbf{A}_t^{-1}\mathbf{g}_t$

Approximating KONS

KONS: $d_{\text{eff}}^T(\alpha)\log(T)$ regret

↳ large $\mathcal{H} \Rightarrow \mathcal{O}(t)$ prediction $\phi_t^\mathsf{T}\mathbf{w}_t$, $\mathcal{O}(t^2)$ updates $\mathbf{g}_t - \mathbf{A}_t^{-1}\mathbf{g}_t$

Use approximate second order updates in large $\mathcal{H}$ [Calandriello et al., 2017b]

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

Approximating KONS

KONS: $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret

↳ large $\mathcal{H} \Rightarrow \mathcal{O}(t)$ prediction $\phi_t^{\mathsf{T}}\mathbf{w}_t$, $\mathcal{O}(t^2)$ updates $\mathbf{g}_t - \mathbf{A}_t^{-1}\mathbf{g}_t$

Use approximate second order updates in large $\mathcal{H}$ [Calandriello et al., 2017b]

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

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

↳ replace φ with approximate map $\widetilde{\varphi}$ (random features, embeddings)

Approximating KONS

KONS: $d_{\text{eff}}^T(\alpha)\log(T)$ regret

↳ large $\mathcal{H} \Rightarrow \mathcal{O}(t)$ prediction $\phi_t^\mathsf{T}\mathbf{w}_t$, $\mathcal{O}(t^2)$ updates $\mathbf{g}_t - \mathbf{A}_t^{-1}\mathbf{g}_t$

Use approximate second order updates in large $\mathcal{H}$ [Calandriello et al., 2017b]

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

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

↳ replace φ with approximate map $\widetilde{\varphi}$ (random features, embeddings)

finite $\widetilde{\mathcal{H}} \Rightarrow$ constant per-step prediction/update cost

Approximating KONS

KONS: $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret

↳ large $\mathcal{H} \Rightarrow \mathcal{O}(t)$ prediction $\phi_t^{\mathsf{T}}\mathbf{w}_t$, $\mathcal{O}(t^2)$ updates $\mathbf{g}_t - \mathbf{A}_t^{-1}\mathbf{g}_t$

Use approximate second order updates in large $\mathcal{H}$ [Calandriello et al., 2017b]

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

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

↳ replace φ with approximate map $\widetilde{\varphi}$ (random features, embeddings)

finite $\widetilde{\mathcal{H}} \Rightarrow$ constant per-step prediction/update cost

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

Approximating KONS

KONS: $d_{\text{eff}}^{T}(\alpha)\log(T)$ regret

↳ large $\mathcal{H} \Rightarrow \mathcal{O}(t)$ prediction $\phi_t^{\mathsf{T}}\mathbf{w}_t$, $\mathcal{O}(t^2)$ updates $\mathbf{g}_t - \mathbf{A}_t^{-1}\mathbf{g}_t$

Use approximate second order updates in large $\mathcal{H}$ [Calandriello et al., 2017b]

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

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

↳ replace φ with approximate map $\widetilde{\varphi}$ (random features, embeddings)

finite $\widetilde{\mathcal{H}} \Rightarrow$ constant per-step prediction/update cost

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

(a) Exact KONS in $\widetilde{\mathcal{H}}$: $d_{\text{eff}}^{T}(\alpha)\log(T)$

Approximating KONS

KONS: $d_{\text{eff}}^T(\alpha)\log(T)$ regret

↳ large $\mathcal{H} \Rightarrow \mathcal{O}(t)$ prediction $\phi_t^\mathsf{T}\mathbf{w}_t$, $\mathcal{O}(t^2)$ updates $\mathbf{g}_t - \mathbf{A}_t^{-1}\mathbf{g}_t$

Use approximate second order updates in large $\mathcal{H}$ [Calandriello et al., 2017b]

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

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

↳ replace φ with approximate map $\widetilde{\varphi}$ (random features, embeddings)

finite $\widetilde{\mathcal{H}} \Rightarrow$ constant per-step prediction/update cost

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

(a) Exact KONS in $\widetilde{\mathcal{H}}$: $d_{\text{eff}}^T(\alpha)\log(T)$

(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

same for fixed budget (e.g., k-rank approx [Luo et al., 2016])

Subspace approximation error

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

↳ the adversary will find orthogonal points and exploit this

same for fixed budget (e.g., k-rank approx [Luo et al., 2016])

Use Nyström approximation instead and adapt it online

Subspace approximation error

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

↳ the adversary will find orthogonal points and exploit this

same for fixed budget (e.g., k-rank approx [Luo et al., 2016])

Use Nyström approximation instead and adapt it online

↳ 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

same for fixed budget (e.g., k-rank approx [Luo et al., 2016])

Use Nyström approximation instead and adapt it online

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

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

Subspace approximation error

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

↳ the adversary will find orthogonal points and exploit this

same for fixed budget (e.g., k-rank approx [Luo et al., 2016])

Use Nyström approximation instead and adapt it online

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

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

Use RLS (KORS) to select inducing points

Subspace approximation error

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

↳ the adversary will find orthogonal points and exploit this

same for fixed budget (e.g., k-rank approx [Luo et al., 2016])

Use Nyström approximation instead and adapt it online

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

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

Use RLS (KORS) to select inducing points

↳ SQUEAK without removal ($\mathcal{I}_t \subseteq \mathcal{I}_{t+1}$, $\widetilde{\mathcal{H}}_t \subseteq \widetilde{\mathcal{H}}_{t+1}$)

Subspace approximation error

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

↳ the adversary will find orthogonal points and exploit this

same for fixed budget (e.g., k-rank approx [Luo et al., 2016])

Use Nyström approximation instead and adapt it online

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

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

Use RLS (KORS) to select inducing points

↳ SQUEAK without removal ($\mathcal{I}_t \subseteq \mathcal{I}_{t+1}$, $\widetilde{\mathcal{H}}_t \subseteq \widetilde{\mathcal{H}}_{t+1}$)

w.h.p. accurate and maximum size $|\widetilde{\mathcal{H}}_t| \leq \mathcal{O}(d_{\text{eff}}^T(\gamma) \log^2(T))$

Subspace approximation error

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

↳ the adversary will find orthogonal points and exploit this

same for fixed budget (e.g., k-rank approx [Luo et al., 2016])

Use Nyström approximation instead and adapt it online

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

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

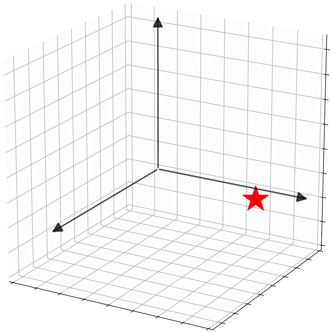
Use RLS (KORS) to select inducing points

↳ SQUEAK without removal ($\mathcal{I}_t \subseteq \mathcal{I}_{t+1}$, $\widetilde{\mathcal{H}}_t \subseteq \widetilde{\mathcal{H}}_{t+1}$)

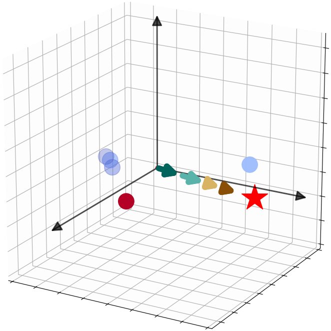
w.h.p. accurate and maximum size $|\widetilde{\mathcal{H}}_t| \leq \mathcal{O}(d_{\text{eff}}^T(\gamma)\log^2(T))$

$\widetilde{\mathcal{O}}(d_{\text{eff}}^T(\gamma)^2)$ time/space cost to run exact KONS in $\widetilde{\mathcal{H}}_t$

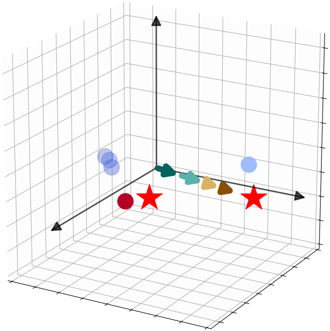
PROS-N-KONS



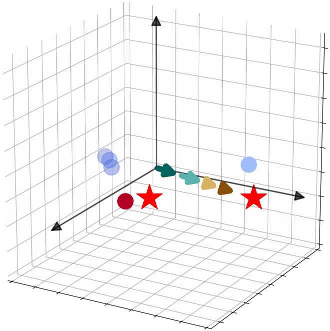
PROS-N-KONS



PROS-N-KONS



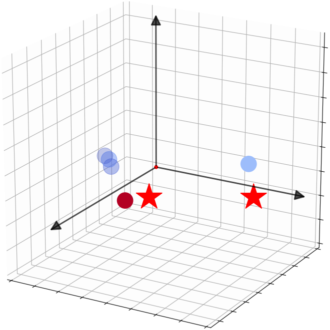
PROS-N-KONS



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

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

PROS-N-KONS



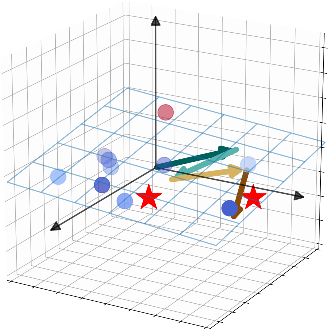
Reset $\widetilde{\mathbf{w}}_t$ and $\widetilde{\mathbf{A}}_t$ when $\widetilde{\mathcal{H}}_t$ changes

↳ 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

PROS-N-KONS



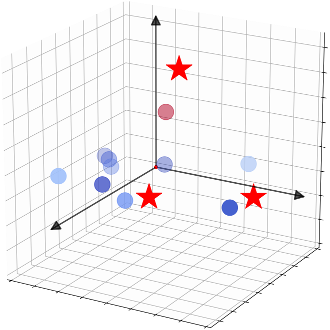
Reset $\widetilde{\mathbf{w}}_t$ and $\widetilde{\mathbf{A}}_t$ when $\widetilde{\mathcal{H}}_t$ changes

↳ 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

PROS-N-KONS



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

Experiments - regression

$\alpha = 1, \gamma = 1$						
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	—	—

Experiments - binary classification

$\alpha = 1, \gamma = 1$						
Algorithm	ijcnn1 $n = 141,691$, $d = 22$			cod-rna $n = 271,617$, $d = 8$		
	accuracy	#SV	time	accuracy	#SV	time
FOGD	9.06 ± 0.05	400	—	10.30 ± 0.10	400	—
NOGD	9.55 ± 0.01	100	—	13.80 ± 2.10	100	—
Dual-SGD	**8.35** ± 0.20	100	—	**4.83** ± 0.21	100	—
PROS-N-KONS	9.70 ± 0.01	100	211.91	13.95 ± 1.19	38	270.81
Con-KONS	9.64 ± 0.01	101	215.71	18.99 ± 9.47	38	271.85
B-KONS	9.70 ± 0.01	98	206.53	13.99 ± 1.16	38	274.94
BATCH	8.33 ± 0.03	—	—	3.781 ± 0.01	—	—

$\alpha = 0.01, \gamma = 0.01$						
Algorithm	ijcnn1 $n = 141,691$, $d = 22$			cod-rna $n = 271,617$, $d = 8$		
	accuracy	#SV	time	accuracy	#SV	time
FOGD	9.06 ± 0.05	400	—	10.30 ± 0.10	400	—
NOGD	9.55 ± 0.01	100	—	13.80 ± 2.10	100	—
Dual-SGD	8.35 ± 0.20	100	—	4.83 ± 0.21	100	—
PROS-N-KONS	10.73 ± 0.12	436	1003.82	4.91 ± 0.04	111	459.28
Con-KONS	6.23 ± 0.18	432	987.33	5.81 ± 1.96	111	458.90
B-KONS	**4.85** ± 0.08	100	147.22	**4.57** ± 0.05	100	333.57
BATCH	5.61 ± 0.01	—	—	3.61 ± 0.01	—	—

PROS-N-KONS - recap

PROS-N-KONS: avoid curse of kernelization, constant per-step cost

PROS-N-KONS - recap

PROS-N-KONS: avoid curse of kernelization, constant per-step cost

First approximate method with logarithmic regret

PROS-N-KONS - recap

PROS-N-KONS: avoid curse of kernelization, constant per-step cost

First approximate method with logarithmic regret

Future work

PROS-N-KONS - recap

PROS-N-KONS: avoid curse of kernelization, constant per-step cost

First approximate method with logarithmic regret

Future work

Restarts really necessary?

PROS-N-KONS - recap

PROS-N-KONS: avoid curse of kernelization, constant per-step cost

First approximate method with logarithmic regret

Future work

Restarts really necessary?

Adaptive α and γ?

PROS-N-KONS - recap

PROS-N-KONS: avoid curse of kernelization, constant per-step cost

First approximate method with logarithmic regret

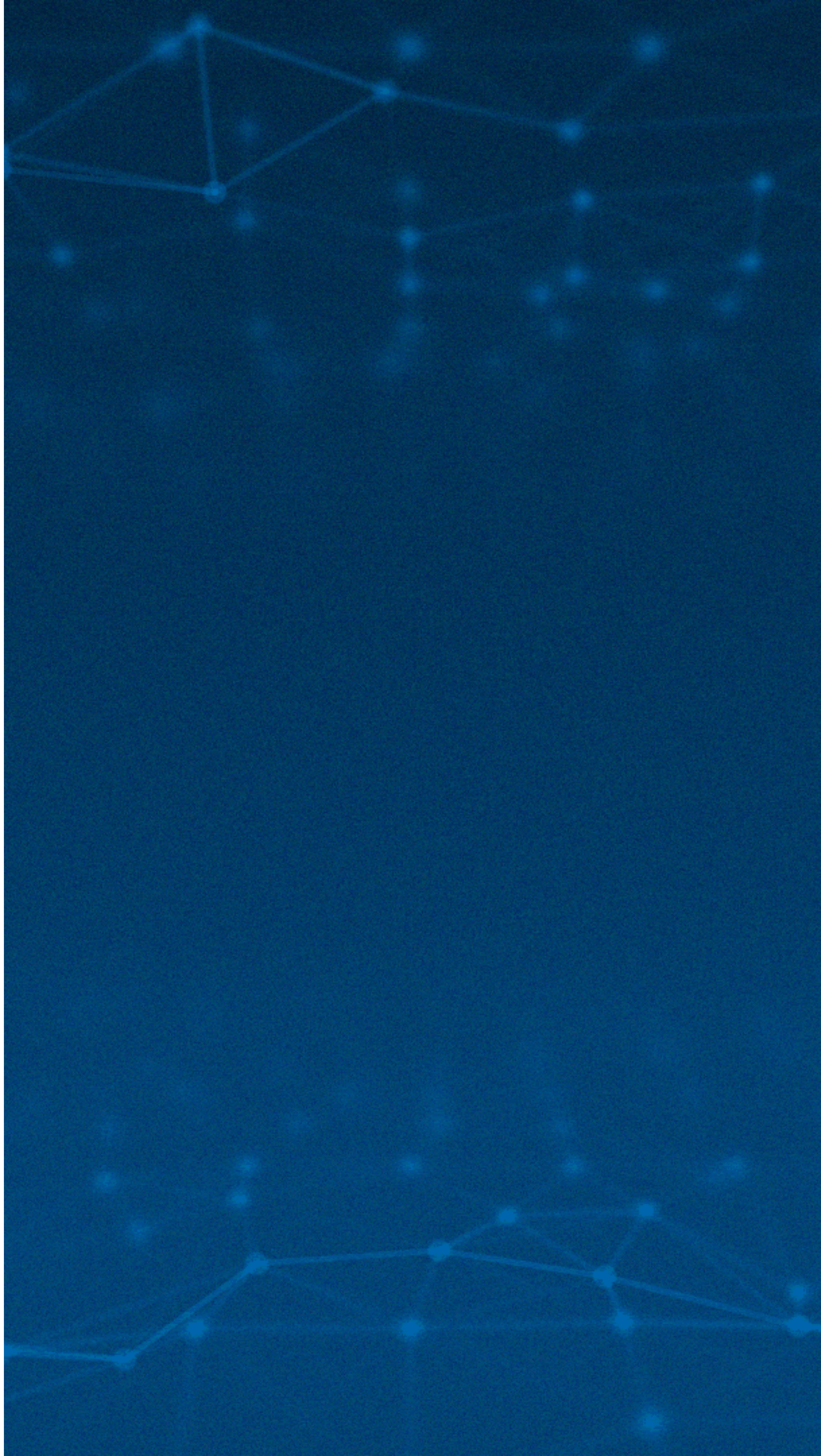
Future work

Restarts really necessary?

Adaptive α and γ?

...and now, back to the beginning!

BACK TO THE BEGINNING: GRAPH SPARSIFICATION



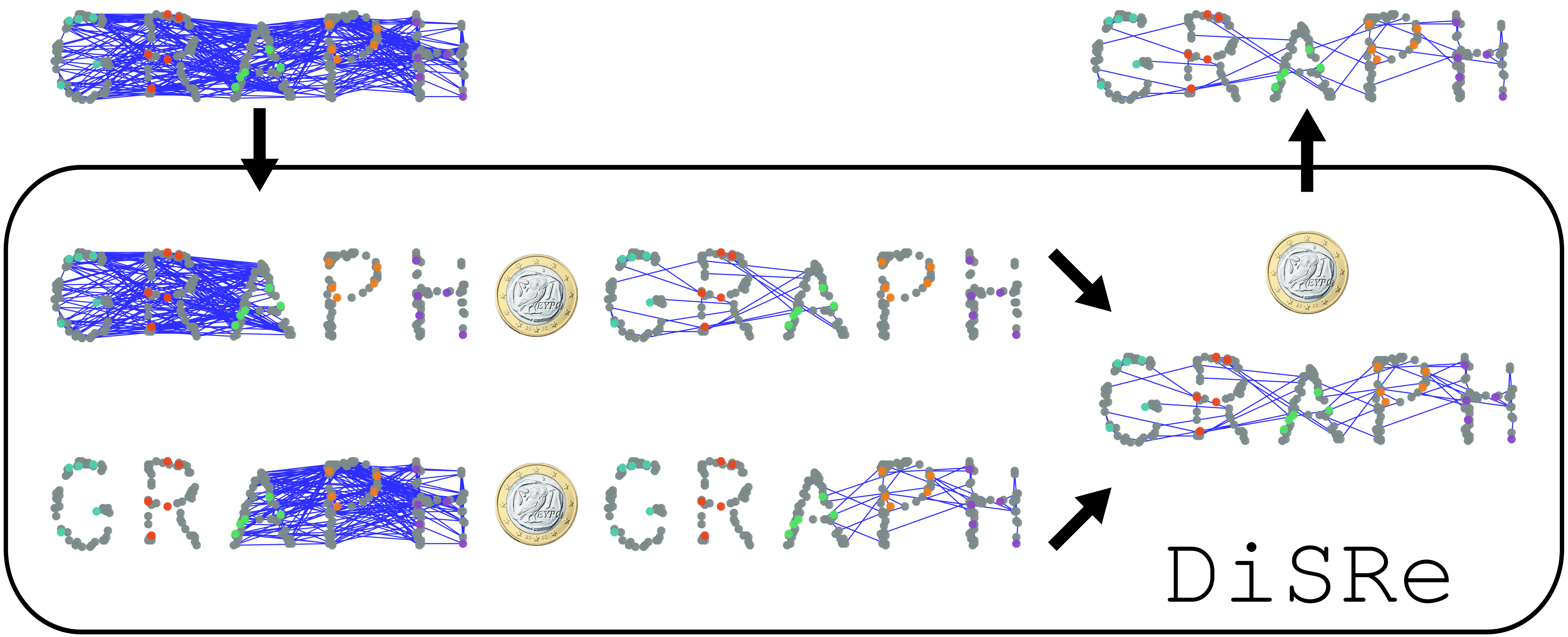
SCALING UP GRAPH LEARNING

- Large graphs do not fit in a single machine memory
- multiple passes slow, distribution has communication costs
- removing edges impacts structure/accuracy
- Make the graph sparse, while preserving its structure for learning

$$(1-\varepsilon)\mathbf{L}_{\mathcal{G}} \preceq \mathbf{L}_{\mathcal{H}} \preceq (1+\varepsilon)\mathbf{L}_{\mathcal{G}}$$

$$(1-\varepsilon)\mathbf{L}_{\mathcal{G}} - \varepsilon\gamma\mathbf{I} \preceq \mathbf{L}_{\mathcal{H}} \preceq (1+\varepsilon)\mathbf{L}_{\mathcal{G}} + \varepsilon\gamma\mathbf{I}$$

DISRE GUARANTEES



Theorem

Given an arbitrary graph $\mathcal{G}$ w.h.p. DISRE satisfies

(1) each sub-graphs is an (ε, γ)-sparsifier

(2) with at most $\mathcal{O}(d_{\text{eff}}(\gamma)\log(n))$ edges.

DISRE EXPERIMENTS

Dataset: Amazon co-purchase graph [Yang and Leskovec 2015]
↳ natural, artificially sparse (true graph known only to Amazon)
↳ we compute 4-step random walk to recover removed co-purchases [Gleich and Mahoney 2015]

Target: eigenvector $\mathbf{v}$ associated with $\lambda_2(\mathbf{L}_\mathcal{G})$ [Sadhanala et al. 2016]

$n = 334,863$ nodes, $m = 98,465,352$ edges (294 avg. degree)

Alg.	*Parameters*	$\|\mathcal{E}\|$ (x10^6)	$\|\|\widetilde{\mathbf{f}} - \mathbf{v}\|\|_2^2$ ($\sigma = 10^{-3}$)	$\|\|\widetilde{\mathbf{f}} - \mathbf{v}\|\|_2^2$ ($\sigma = 10^{-2}$)
EXACT		98.5	0.067 ± 0.0004	0.756 ± 0.006
kN	$k = 60$	15.7	0.172 ± 0.0004	0.822 ± 0.002
DisRe	$\gamma = 0$	22.8	0.068 ± 0.0004	**0.756** ± 0.005
DisRe	$\gamma = 10^2$	11.8	**0.068** ± 0.0002	0.772 ± 0.004

Time: Loading $\mathcal{G}$ from disk 90sec, DisRe 120sec($k = 4 \times 32$ CPU), computing $\widetilde{\mathbf{f}}$ 120sec, computing $\widehat{\mathbf{f}}$ 720sec

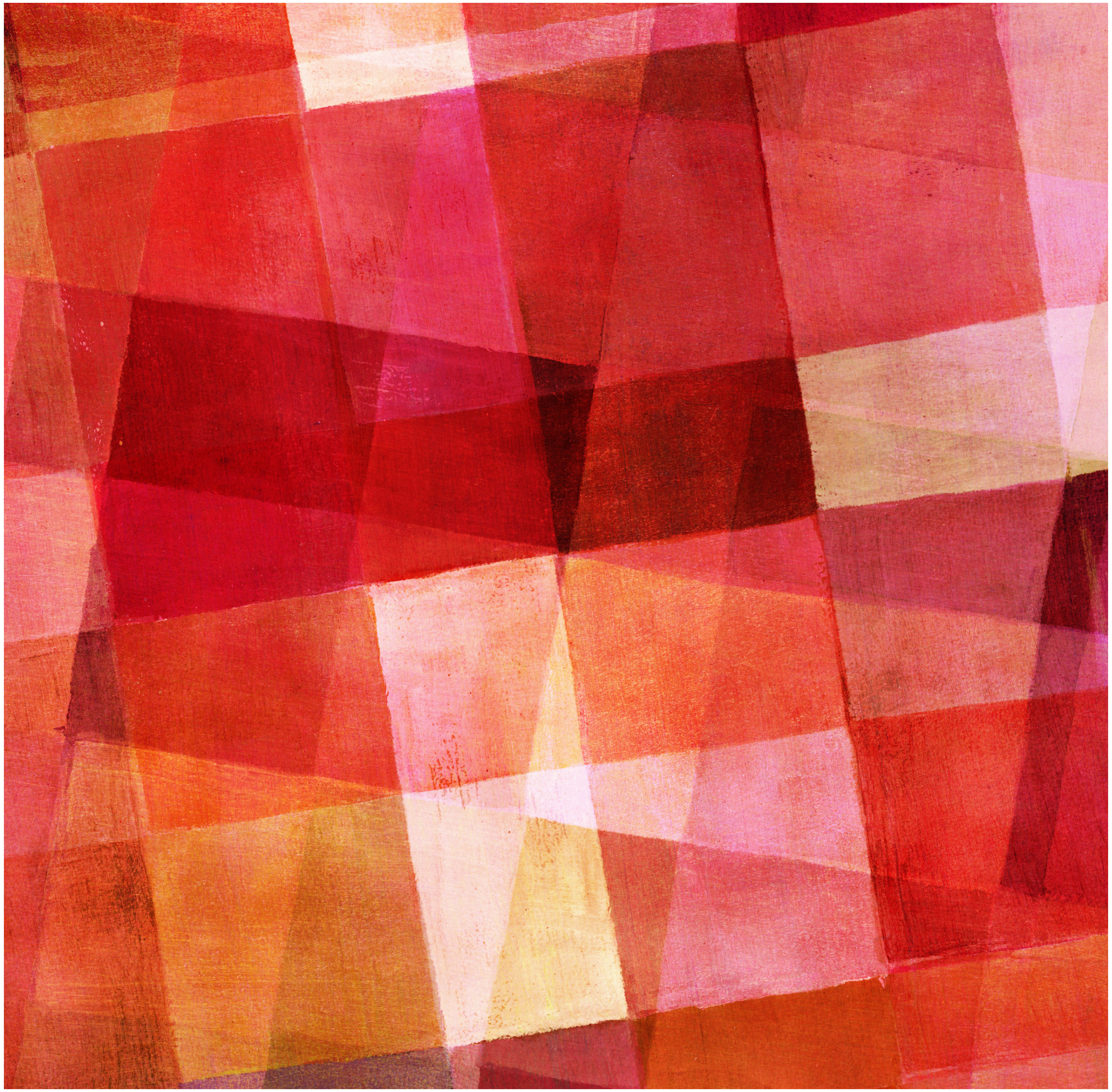
AFTER 12 YEARS?

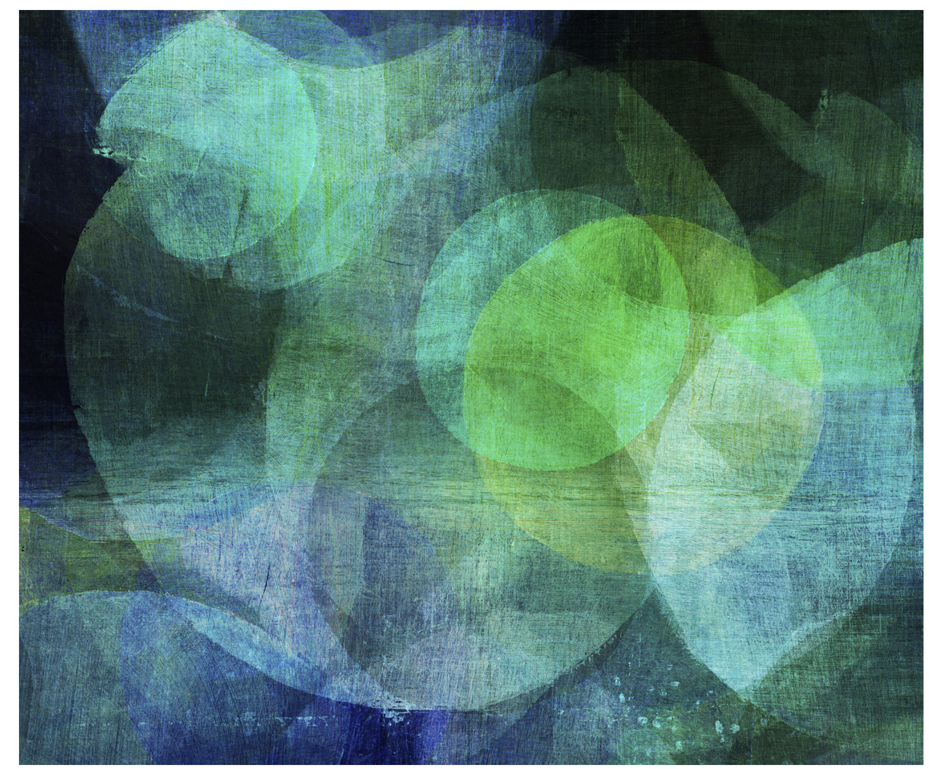
THIS IS JUST THE BEGINNING!

SPARSIFICATION: NEXT

- **SPARSIFYING GP-UCB RIGHT**
  - More than 20 years of heuristics
  - Even 2019 results on sparsifying LinUCB can go wrong
  - BKB - **adaptive** dictionary, guarantees regret and is fast
- **BATCHED GP-UCB SPARSIFICATION** - stay tuned!
- *Negative dependence/online leverages scores/DPPs*
- **FAST SAMPLING OF DPPs** - repulsion for the sets!
  - w/Michał Dereziński and Daniele Calandriello
  - online lev. Scores + R-DPP + downsampling → **perfect**

Michal Valko, valkom@google.com
http://researchers.lille.inria.fr/~valko/hp/





Michal Valko, valkom@google.com
http://researchers.lille.inria.fr/~valko/hp/