# Efficient online learning with Kernels for adversarial large scale problems

`Pierre Gaillard` (INRIA, Ecole Normale Supérieure, Paris)

Joint work with Rémi Jézéquel (ENS), Alessandro Rudi (INRIA, ENS)

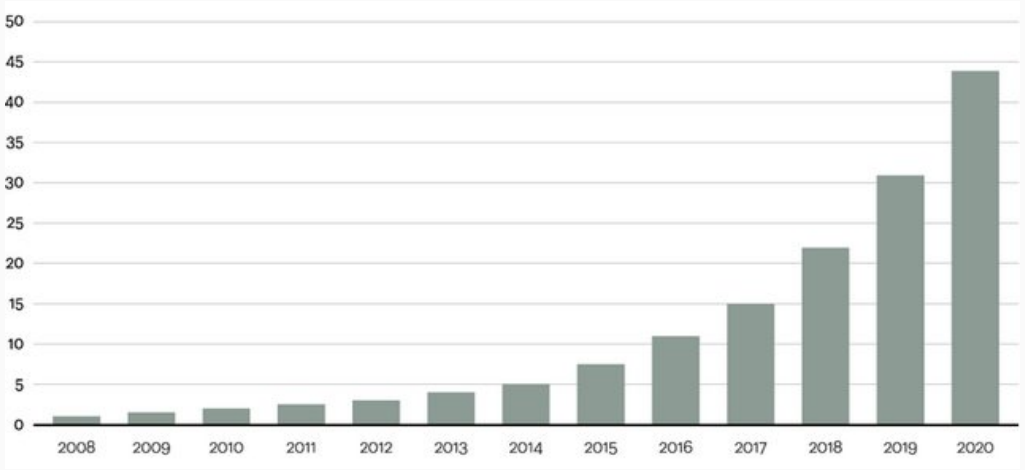
September, 2019

CWI-INRIA Workshop 2019

# Motivation

Nowadays,

- **volume and velocity of data flows** are sharply increasing.
  → need of online methods to treat and adapt to data on the fly
- **very large datasets** are better handled by non-parametric methods
- **data is getting more complicated** and simple stochastic assumptions such as i.i.d. data are often not satisfied
  → need of robust adversarial guarantees




**Goal**: combine these different aspects due to large scale and arbitrary data

# Online learning

**Online learning**: subfield of machine learning where some learner sequentially interacts with an environment and tries to learn and adapt on the fly to the observed data as one goes along.

At each iteration $t \geq 1$,

- the learner receives some input $x_t \in \mathcal{X}$;
- the learner makes a prediction $\hat{y}_t \in \mathbb{R}$
- the environment reveals the output $y_t \in \mathbb{R}$.

Remember the Tim's talk yesterday with football prediction

France winner of the football world cup 2018

# Online learning

**Online learning**: subfield of machine learning where some learner sequentially interacts with an environment and tries to learn and adapt on the fly to the observed data as one goes along.

At each iteration $t \geq 1$,

- the learner receives some input $x_t \in \mathcal{X}$;
- the learner makes a prediction $\hat{y}_t \in \mathbb{R}$
- the environment reveals the output $y_t \in \mathbb{R}$.

**Learner's goal**: minimize his cumulative regret

$$\text{Regret}_n(f) := \sum_{t=1}^{n} (y_t - \hat{y}_t)^2 - \sum_{t=1}^{n} \big(y_t - f(x_t)\big)^2$$

over all functions $f$ in a space of functions $\mathcal{H}$. $\leftarrow$ infinite dimensional parameter space

# Online learning

**Online learning**: subfield of machine learning where some learner sequentially interacts with an environment and tries to learn and adapt on the fly to the observed data as one goes along.

At each iteration $t \geq 1$,
- the learner receives some input $x_t \in \mathcal{X}$;
- the learner makes a prediction $\hat{y}_t \in \mathbb{R}$
- the environment reveals the output $y_t \in \mathbb{R}$.

**Learner's goal**: minimize his cumulative regret

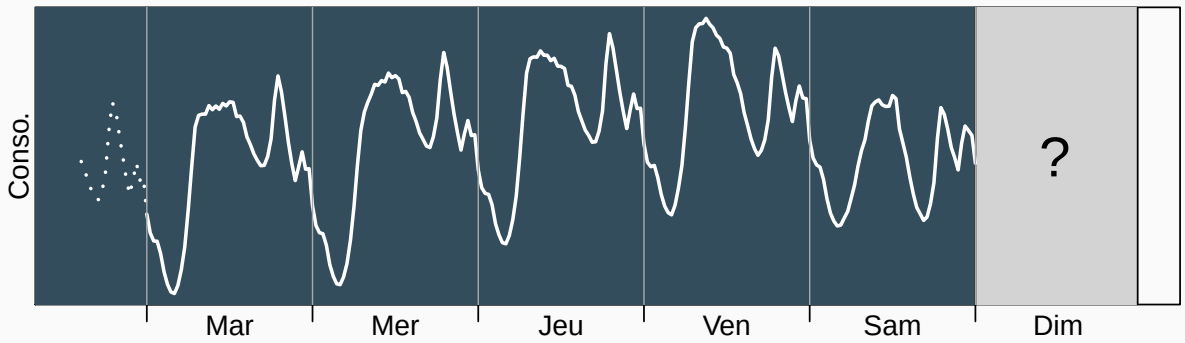
$$\text{Regret}_n(f) := \sum_{t=1}^{n} (y_t - \hat{y}_t)^2 - \sum_{t=1}^{n} \big(y_t - f(x_t)\big)^2$$

over all functions $f$ in a space of functions $\mathcal{H}$. ← infinite dimensional parameter space

The inputs $x_t$ and the outputs $y_t$ are sequentially chosen by the environment and can be arbitrary.

# Example of application

Prediction of the electricity consumption.




The latter is stochastic but non-i.i.d. nor stationary.

Each day,

- the learner receives some features $x_t$ (forecast of the temperature, cloud coverage, calendar information, ...)
- makes a prediction of the electricity load of the following day.

Potentially large scale with new smart-meters that measure the individual consumptions of each household.

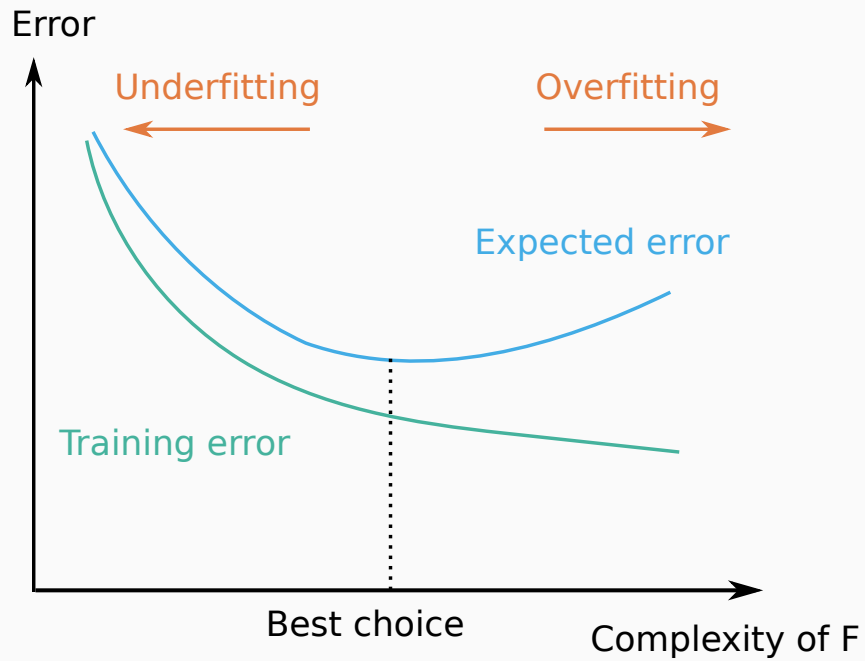
# What class of functions $\mathcal{H}$ do we consider?

**Goal:** minimize the cumulative regret

$$\text{Regret}_n(f) := \sum_{t=1}^{n}(y_t - \hat{y}_t)^2 - \sum_{t=1}^{n}\left(y_t - f(x_t)\right)^2$$

over all functions $f$ in a space of functions $\mathcal{H}$.

**Choice of $\mathcal{H}$:** approximation-estimation trade-off.

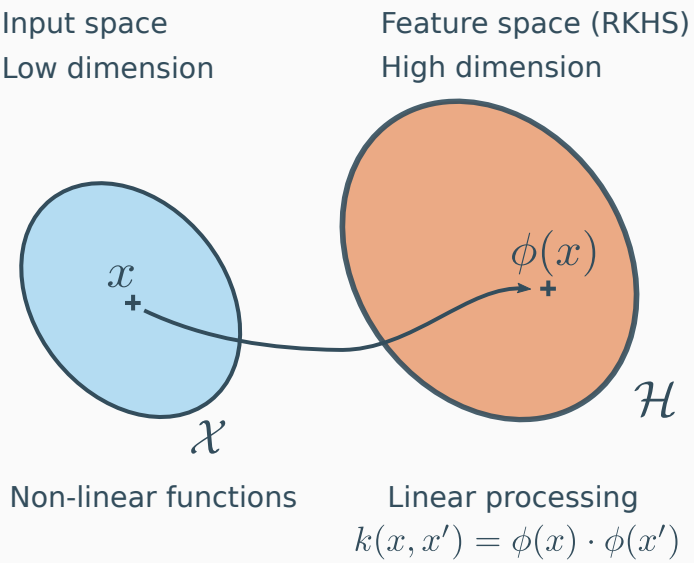


# Reproducing kernel hilbert spaces (RKHS)

We consider Reproducing Kernel Hilbert Space (RKHS) associated with a *kernel function*

$$k(x, x') = \phi(x) \cdot \phi(x').$$

Kernel methods embed finite dimensional data into infinite dimensional feature spaces.




**Pros of RKHS**:

- Many function spaces can be represented as RKHS: polynomials of arbitrary degree, band-limited functions, analytic functions with given decay at infinity, Sobolev spaces,...
- The kernel representation makes the computation “feasible”.

# Intrinsic complexity of the RKHS: the effective dimension

Although the feature space can be very large, the complexity of the RKHS depends on the decay of eigenvalues in the principle component analysis of the kernel matrix.

Effective dimension $\approx$ How many components are needed to approximate $K_{nn}$ at scale $\lambda$?

The complexity of the RKHS is measured by its effective dimension: for all scale $\lambda > 0$

$$d_{\text{eff}}(\lambda) := \mathrm{Tr}(K_{nn}(K_{nn} + \lambda I_n)^{-1})$$

where $K_{nn} := \left[k(x_i, x_j)\right]_{1 \le i,j \le n} \in \mathbb{R}^{n \times n}$ denotes the *kernel matrix* at time $n$.

# Example of upper-bounds on the effective dimension

**In the worst case**, we have

$$d_{\text{eff}}(\lambda) \lesssim \frac{n}{\lambda}\,.$$

**The Gaussian Kernel** satisfies

$$d_{\text{eff}}(\lambda) \lesssim \left(\log \frac{n}{\lambda}\right)^d$$

**The Sobolev space** $W_2^\beta(\mathbb{R}^d)$ (functions from $\mathbb{R}^d \to \mathbb{R}$ whose derivatives up to order $\beta$ are in $L_2(\mathbb{R}^d)$) with $\beta > d/2$ is a RKHS and

$$d_{\text{eff}}(\lambda) \lesssim \left(\frac{n}{\lambda}\right)^{\frac{d}{2\beta}}\,.$$

**Capacity condition**

$$d_{\text{eff}}(\lambda) \lesssim \left(\frac{n}{\lambda}\right)^{\gamma}, \qquad 0 \leq \gamma \leq 1, \lambda > 0$$

# Algorithm based on non-linear ridge forecaster of [1] [2]

## Nonlinear Ridge Forecaster (Probably an instance of Exponential Weights)

Regularization parameter: $\lambda > 0$. At round $t \geq 1$, forecast function

$$\hat{f}_t \in \operatorname*{arg\,min}_{f \in \mathcal{H}} \left\{ \sum_{s=1}^{t-1} \left(y_s - f(x_s)\right)^2 + \lambda \|f\|^2 + f(x_t)^2 \right\}.$$

Optimal regret (up to log factors)

$$\text{Regret}_n(f) \lesssim \lambda \|f\|^2 + d_{\text{eff}}(\lambda)$$

[1] Azoury, K. S. and Warmuth, M. K. 2001.

[2] Vovk, V. 2001.

# Algorithm based on non-linear ridge forecaster of [1] [2]

## Nonlinear Ridge Forecaster (Probably an instance of Exponential Weights)

Regularization parameter: $\lambda > 0$. At round $t \geq 1$, forecast function

$$\hat{f}_t \in \operatorname*{arg\,min}_{f \in \mathcal{H}} \left\{ \sum_{s=1}^{t-1} \left(y_s - f(x_s)\right)^2 + \lambda \|f\|^2 + f(x_t)^2 \right\}.$$

- Optimal regret (up to log factors)

$$\text{Regret}_n(f) \lesssim \lambda \|f\|^2 + d_{\text{eff}}(\lambda)$$

- Closed form solution by solving a $t \times t$ linear system:

$$\hat{f}_t(x) = \sum_{i=1}^{t} k(x, x_i) c_i \quad \text{with} \quad (K_{tt} + \lambda I)^{-1} c = y_{1:t}$$

[1] Azoury, K. S. and Warmuth, M. K. 2001.

[2] Vovk, V. 2001.

# Better complexity?

The issue of the previous algorithm is that it needs to compute and inverse the Kernel matrix:

$$K_{nn} = \left[k(x_i, x_j)\right]_{1 \leq i,j \leq n}$$

of size $n \times n$.

👎 Per-round (time and space) complexity:

$$O(n^2) \quad \leftarrow$$

Prohibitive for large datasets.
Can we improve it?

💡 Hopefully, the eigenvalues decrease rapidly and $K_{nn}$ could be approximated with $d_{\text{eff}}(\lambda)$ principal components.

# Nyström

[3]

The solution of the Nonlinear Ridge forecaster equals

$$\hat{f}_t^{\text{Ridge}}(x) = \sum_{i=1}^{t} k(x_i, x) c_i \quad \text{with} \quad (K_{tt} + \lambda I)^{-1} c = y_{1:t}$$

It belongs to $\text{Span}\big(k(x_1, \cdot), \ldots, k(x_t, \cdot)\big)$.

**Nyström + Nonlinear Ridge forecaster**

1. Sequentially update a dictionary $\mathcal{I}_t \subset \{x_1, \ldots, x_t\}$ of size $m_t \ll t$
2. Solve the problem on $\hat{\mathcal{H}}_t = \text{Span}\Big\{k(x, \cdot),\ x \in \mathcal{I}_t\Big\}$.

$$\hat{f}_t \in \underset{f \in \tilde{\mathcal{H}}_t}{\arg\min} \left\{ \sum_{s=1}^{t-1} \big(y_s - f(x_s)\big)^2 + \lambda \|f\|^2 + f(x_t)^2 \right\}.$$

That is keep only the $m$ columns in $\mathcal{I}_t$ before solving the linear system:

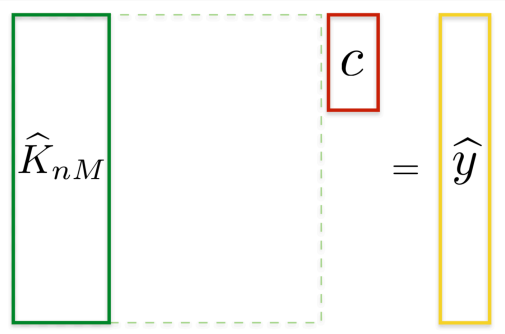
$$\hat{f}_t(x) = \sum_{i : x_i \in \mathcal{I}_t} k(x_i, x) c_i \quad \text{with} \quad (\hat{K}_{tm_t}^\top K_{tm_t} + \lambda K_{m_t m_t})^{-1} c = K_{tm_t}^\top y_{1:t}$$

[3] Smola, A., Schölkopf, B., and Langley, P. 2000.

# Complexity

To compute our prediction we need to solve the linear system:

$$\hat{f}_t(x) = \sum_{i:x_i \in \mathcal{I}_t} k(x_i, x) c_i \quad \text{with} \quad (\hat{K}_{tm_t}^\top K_{tm_t} + \lambda K_{m_t m_t})^{-1} c = K_{tm_t}^\top y_{1:t}$$




**Per-round space and time complexity:** ~~$O(t^2)$~~ $\rightarrow$ $O(m_t^2)$

How to build the dictionary and what size should it be?

# How to build the dictionary?

The inputs $x_t$ might be included into the dictionary independently and uniformly at random.

Calandriello, Lazaric, and Valko 2017 propose the KORS($\mu$) algorithm that evaluate the importance of including $x_t$ to obtain an accurate approximation based on the leverage score.

**Theorem**

Let $\mu, \lambda > 0$. The final dictionary is of size $m = d_{\text{eff}}(\mu)$ and

$$\text{Regret}_n(f) \lesssim \lambda \|f\|^2 + d_{\text{eff}}(\lambda) + \frac{mn\mu}{\lambda}\,.$$

The algorithm runs in $O(m^2)$ time per-iteration.

# Regret for Gaussian kernel

For parameters $\mu, \lambda > 0$,

$$\mathrm{Regret}_n(f) \lesssim \lambda \|f\|^2 + d_{\mathrm{eff}}(\lambda) + \frac{mn\mu}{\lambda}$$

**Corollary (Gaussian kernel)**

For the Gaussian kernel, for the choices $\lambda = 1$ and $\mu = n^{-2}$, we get

$$\mathrm{Regret}_n(f) \lesssim \|f\|^2 + \left(\log n\right)^{d+1}.$$

with a per-round complexity $O(\log(n)^{2d})$.

This is known to be optimal for the Gaussian kernel.

# Explicit rate under capacity condition

For parameters $\mu, \lambda > 0$,

$$\mathrm{Regret}_n(f) \lesssim \lambda \|f\|^2 + d_{\mathrm{eff}}(\lambda) + \frac{mn\mu}{\lambda}$$

## Corollary (Capacity condition)

Let $n \geq 1$ and $m \geq 1$, $\gamma > 0$. Assume that $d_{\mathrm{eff}}(\lambda') \leq (n/\lambda')^{\gamma}$ for all $\lambda' > 0$. Then, with $\mu = nm^{-1/\gamma}$ the dictionary is of size $|\mathcal{I}_n| \lesssim m$ then w.h.p.

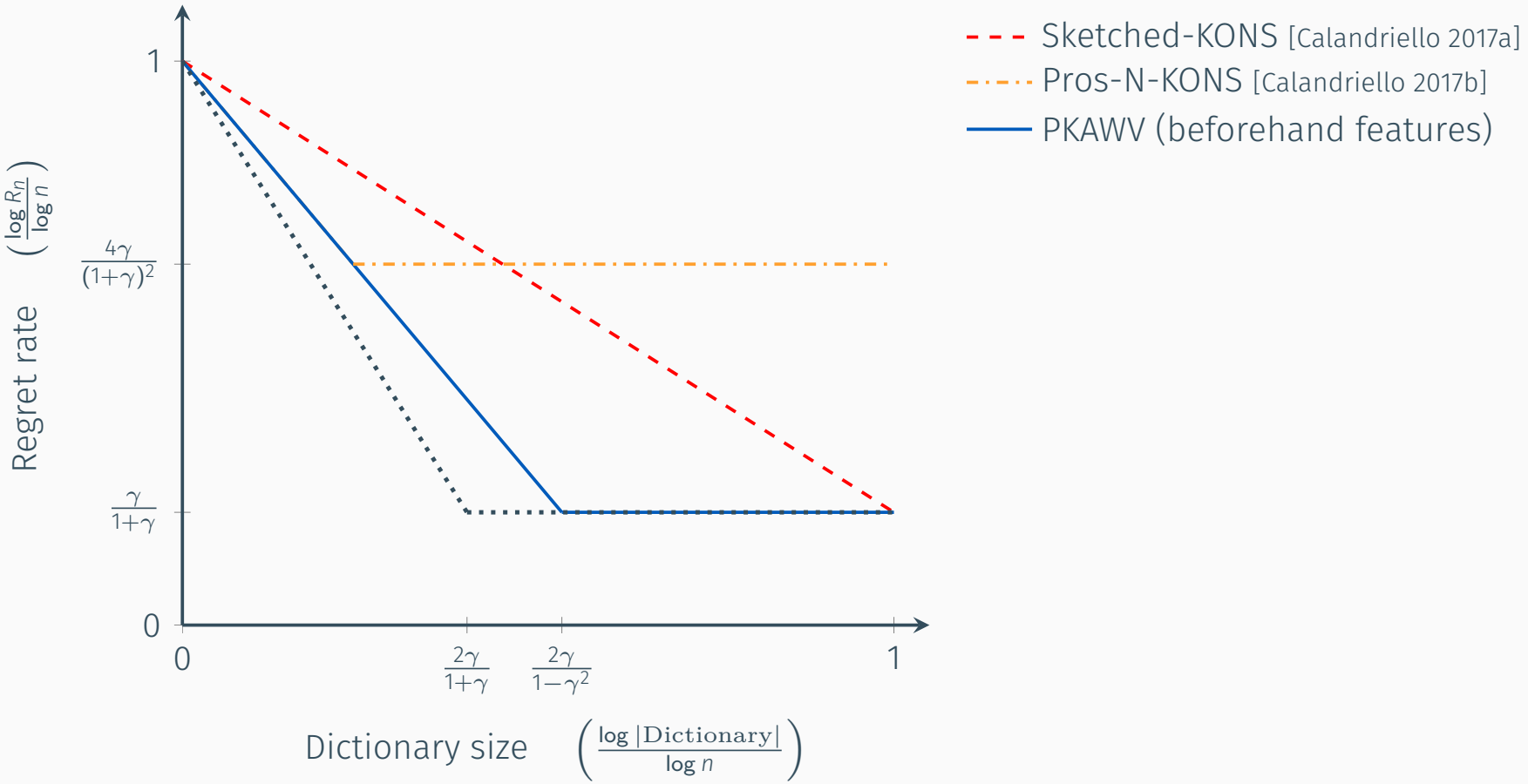
$$R_n \lesssim \begin{cases} n^{\frac{\gamma}{1+\gamma}} & \text{if } m \geq n^{\frac{2\gamma}{1-\gamma^2}} \quad \text{for } \lambda = n^{\frac{\gamma}{1+\gamma}} \\ nm^{\frac{1}{2}-\frac{1}{2\gamma}} & \text{otherwise} \quad \text{for } \lambda = nm^{\frac{1}{2}-\frac{1}{2\gamma}} \end{cases} .$$

The per-round space and time complexity of the algorithm is $O(m^2)$ per iteration.

The algorithm recovers the optimal regret $O(n^{\frac{\gamma}{1+\gamma}})$ with a dictionary of size $m \ll n$ for $\gamma < 1/2$.

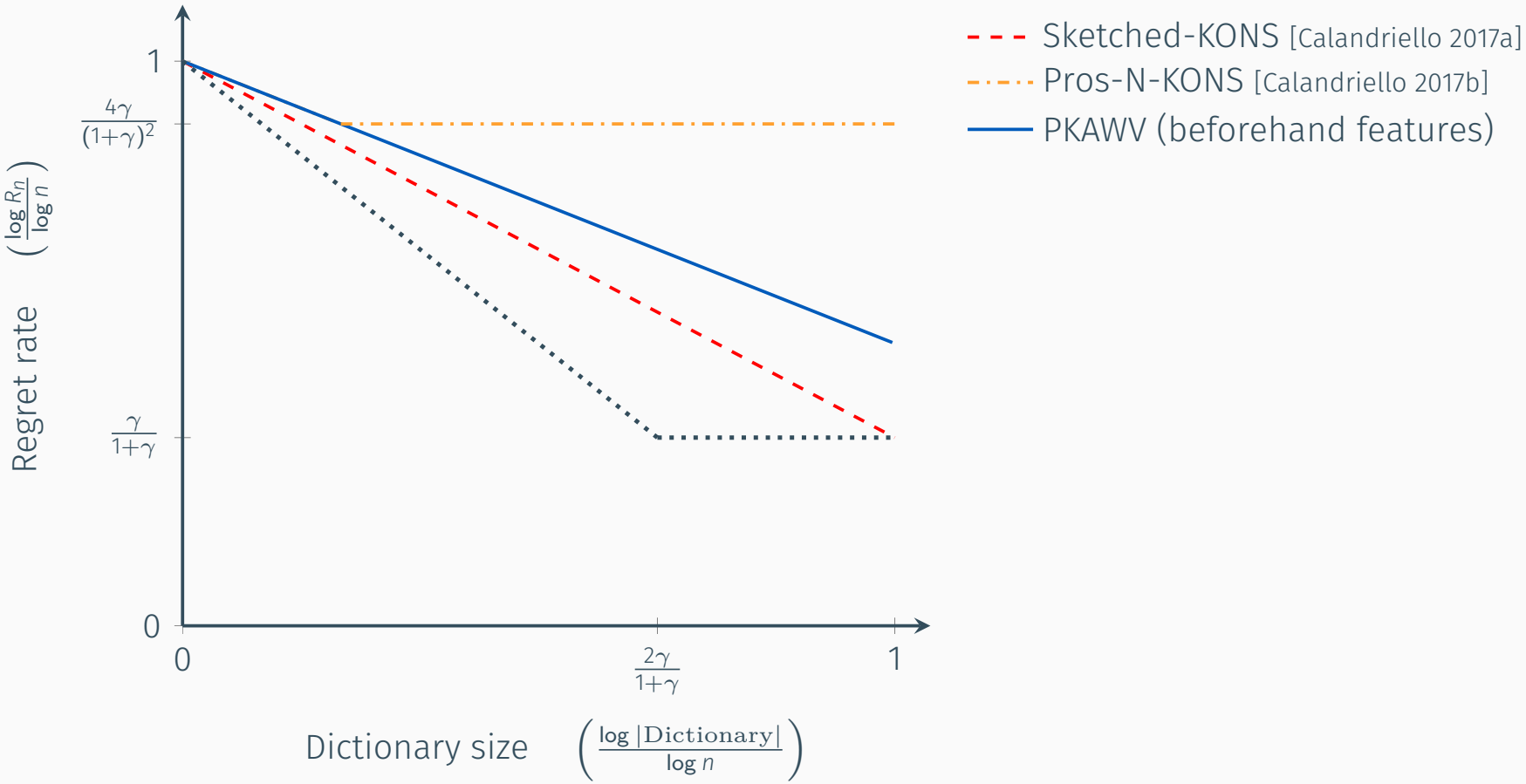
For smaller dictionaries, there is a computational-regret trade-off.

# Regret rate according to the dictionary size ($d_{\mathbf{eff}}(\lambda) \leq (n/\lambda)^{0.25}$)




Comparison of the theoretical regret rate $\text{Regret}_n = n^{\square}$ according to the size of the dictionary $n^{\square}$ when $d_{\text{eff}}(\lambda) \leq (n/\lambda)^{\gamma}$ with $\gamma = 0.25$.

# Regret rate according to the dictionary size ($d_{\mathbf{eff}}(\lambda) \leq (n/\lambda)^{0.5}$)




Comparison of the theoretical regret rate $\text{Regret}_n = n^{\square}$ according to the size of the dictionary $n^{\square}$ when $d_{\text{eff}}(\lambda) \leq (n/\lambda)^{\gamma}$ with $\gamma = 0.5$.

# Beforehand known features

Sequence of feature vectors $x_t$ is given in advance to the learner while only the outputs yt are sequentially revealed.

The dictionary $|\mathcal{I}_n|$ may be computed beforehand.

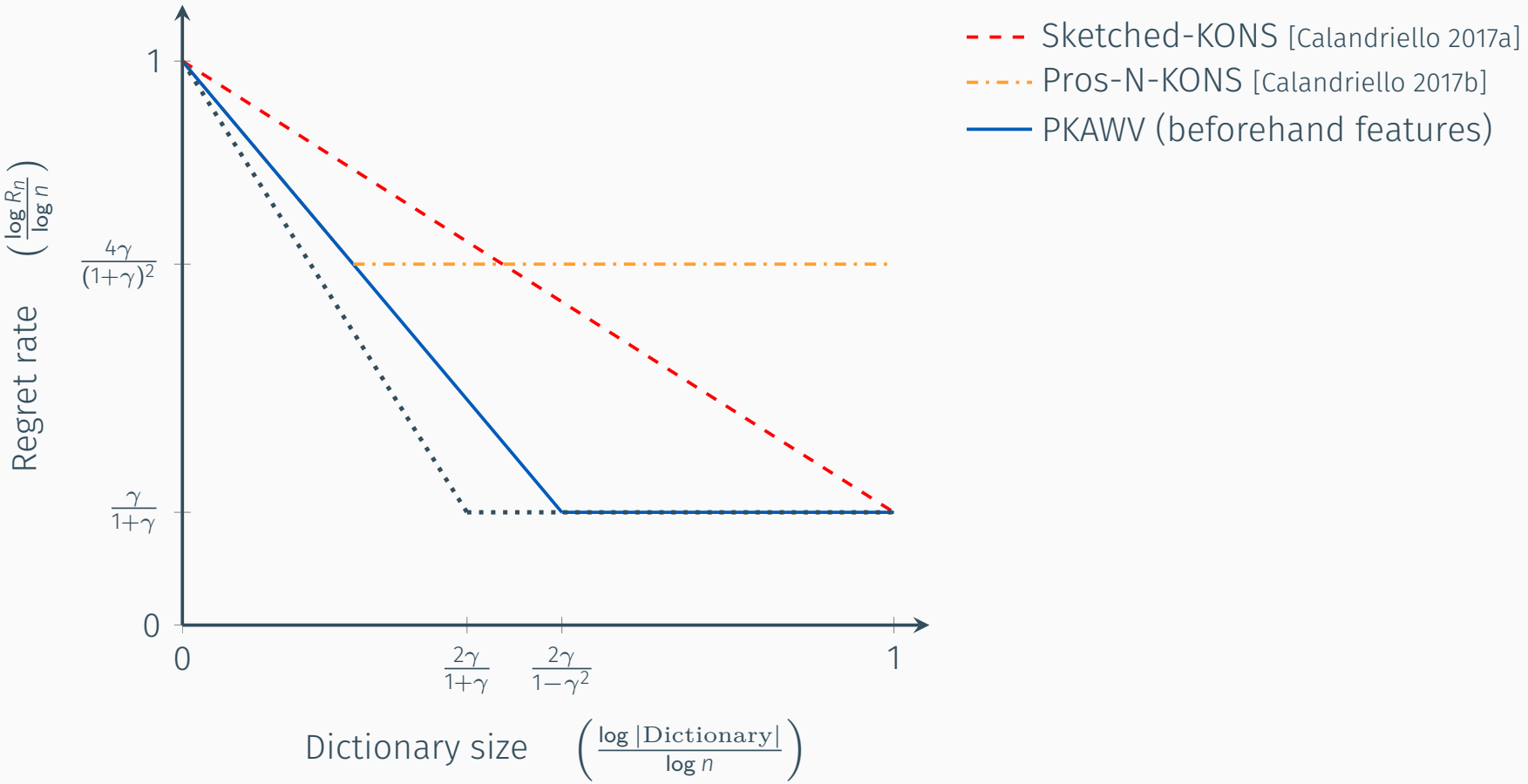
**Theorem**

*Using the parameter $\mu = nm^{-1/\gamma}$, the non-linear Ridge forecaster with dictionary $|\mathcal{I}_n|$ achieves w.h.p*

$$R_n \lesssim \begin{cases} n^{\frac{\gamma}{1+\gamma}} & \text{if } m \geq n^{\frac{2\gamma}{1+\gamma}} \quad \text{for } \lambda = n^{\frac{\gamma}{1+\gamma}} \\ nm^{-\frac{1}{2\gamma}} & \text{otherwise} \quad \text{for } \lambda = nm^{-\frac{1}{2\gamma}} \end{cases} .$$

*Furthermore, w.h.p. the dictionary is of size $|\mathcal{I}_n| \lesssim m$ leading to a per-round space and time complexity $O(m^2)$.*

# Regret rate according to the dictionary size ($d_{\mathbf{eff}}(\lambda) \leq (n/\lambda)^{0.25}$)

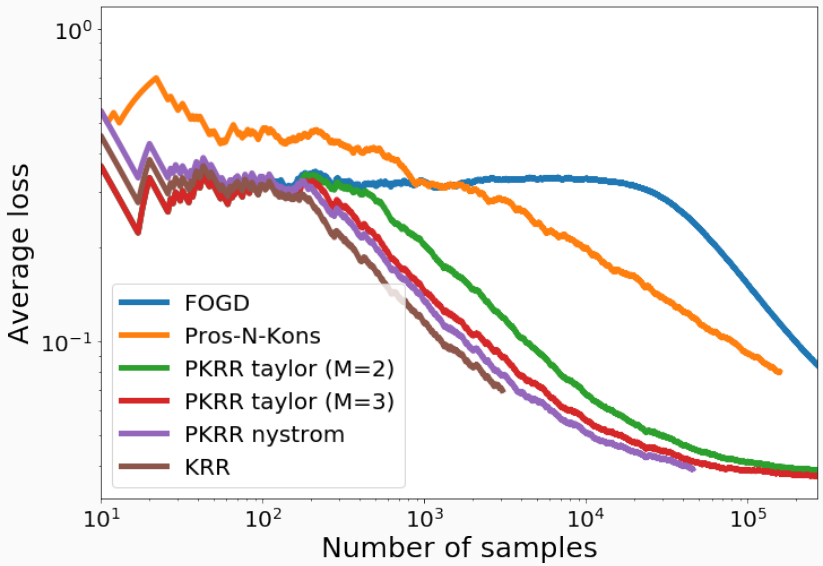



Comparison of the theoretical regret rate $\text{Regret}_n = n^{\square}$ according to the size of the dictionary $n^{\square}$ when $d_{\text{eff}}(\lambda) \leq (n/\lambda)^{\gamma}$ with $\gamma = 0.25$.

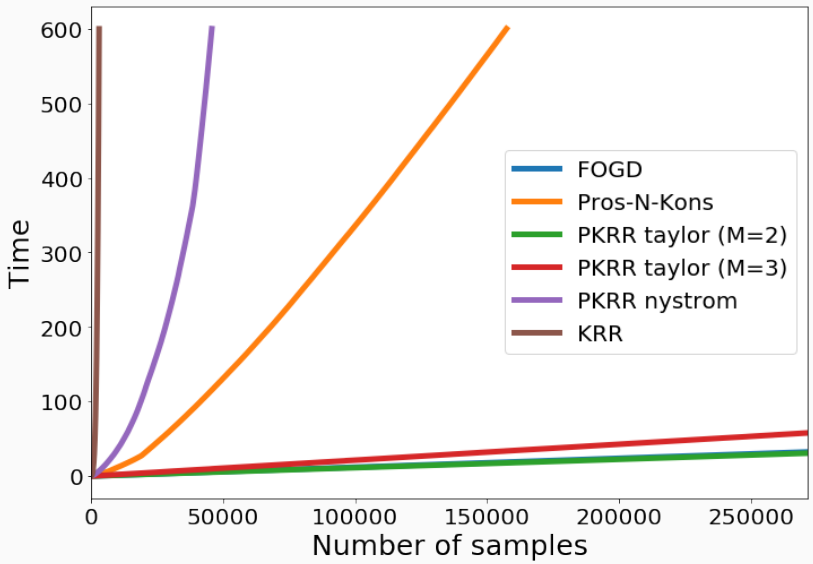
# Experiment on classification dataset

Classification dataset cod-rna: Detection of non-coding RNAs

$$n = 2.7 \cdot 10^5, d = 8$$




Average loss

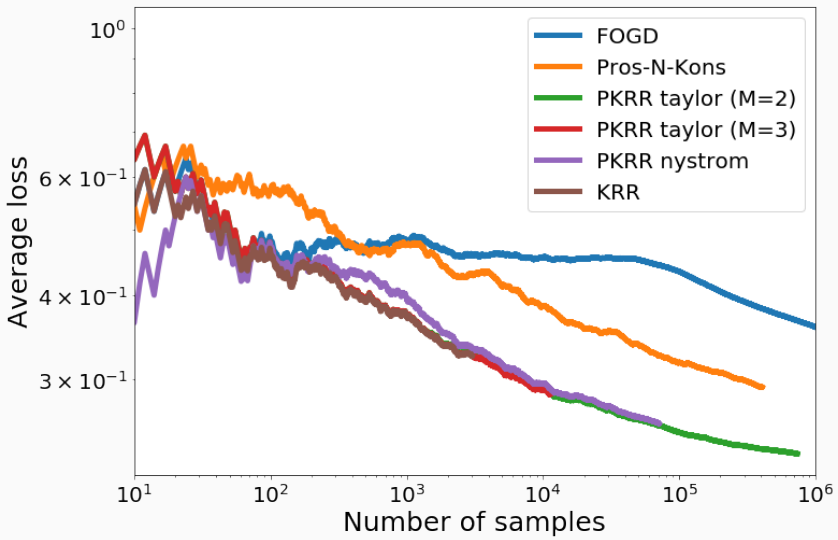



Running time

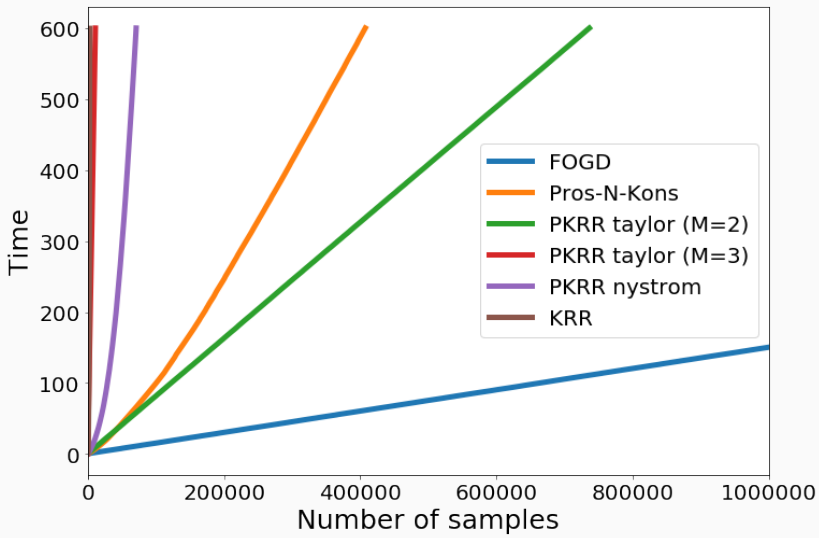
# Experiment on a classification dataset

Classification dataset SUSY: distinguish between a signal process which produces supersymmetric particles and a background process which does not.

$$n = 6 \cdot 10^6, d = 22$$




Average loss




Running time

# Summary

Online learning methods are important for very large datasets $n \gg 1$.

We proposed a method to perform Kernel Online Learning more efficiently while keeping optimal regret.

Better projections can be obtained in specific cases (Gaussian kernel)

# References

Thank you!


- Azoury, K. S. and M. K. Warmuth (2001). “Relative loss bounds for on-line density estimation with the exponential family of distributions”. *Machine Learning*.
- Calandriello, D., A. Lazaric, and M. Valko ([2017]). “Efficient second-order online kernel learning with adaptive embedding”. Neural Information Processing Systems.
- Jezequel, R., P. Gaillard, and A. Rudi (2019). “Efficient online learning with Kernels for adversarial large scale problems”.
- Smola, A., B. Schölkopf, and P. Langley ([2000]). “Sparse greedy matrix approximation for machine learning.”. 17th International Conference on Machine Learning, Stanford, 2000.
- Vovk, V. (2001). “Competitive on-line statistics”. *International Statistical Review*.