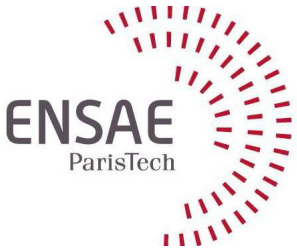
Hicham JANATI



ENSAE ParisTech - 2017
Geometric Methods in Machine Learning

Review

On

Random Features for Polynomial Kernels

Contents

1 Introduction

Machine learning with kernel methods relies on computing and storing the N^2 kernel Gram matrix where N is the number of samples. Thus, algorithms that are based on such methods are not scalable with the size of the data. Especially when the training complexity reaches $O(n^3)$ magnitude. In order to accelerate training, several randomization methods have been developed where kernel approximations are carried out using random samples. Two major categories can be distinguished:

Data dependent procedures (involving subsampled kernel matrix elements; low-rank approximations such as the **Nyström** method [1])

Data independent algorithms that are based on Integral representations of the kernel function (**Random Fourier Features**[2, 3]) or Series expansions (Maclaurin [4]).

In this work, we focus on the latter. We are interested in the Random Fourier Features procedure developed by (Jeffrey Pennington et al., 2015) [3] to approximate Polynomial kernels. To get a good appreciation of the problematic, investigating previous work in the field is a must. Let's recall the findings of (Rahimi and Recht,2007)[2] where, for the first time, an explicit approximation algorithm was introduced based on the Bochner theorem. Because the polynomial kernel violates one of the Bochner theorem assumptions, Random Fourier Features cannot be applied directly. Pennington et al. managed to approximate the polynomial kernel K by a function $\hat{K}$ on which the Bochner theorem can be applied.

Random Maclaurin [4] and Tensor Sketching [5] are two other different approches considered to be most competitive to Spherical Random Features and are used in [3] for experimental comparisons.

All these data independent methods fall in the general framework of low-dimensional embedding constructed by a random map. Thus, in Section 2 we recall the main idea of random features. We focus on the Random Fourier Features method of [2] and briefly perform simulations and real data experiments by approximating radial basis kernels (Gaussian, Laplacian) on which the Bochner theorem applies. In Section 3 before reproducing the experimental results, we investigate the problematic of the polynomial kernel and the solution proposed by [3].

2 Randomized feature map

2.1 Introduction

In standard kernel machines, the data are mapped to a infinite-dimensional space $\mathcal{H}$ (the associated kernel Hilbert space) with a map function $\phi : \mathbb{R}^d \to \mathcal{H}$. If the kernel trick allows us to perform linear calculations in $\mathcal{H}$ using the kernel function: $K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$, it comes however with a large computational cost when the data size explodes. The idea behind random features is to embed the data into a finite-dimensional space – typically euclidean – where the linear dot products approximate K with high probability independently of the data. Such a random map $Z : \mathbb{R}^d \to \mathbb{R}^D$ must verify:

$$K(\mathbf{x}, \mathbf{y}) \approx \langle Z(\mathbf{x}), Z(\mathbf{y}) \rangle$$

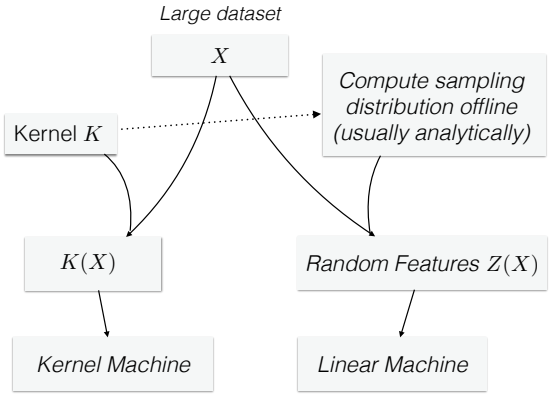


Figure 1: *Scheme putting in contrast exact kernel machine and random features method. The dashed arrow emphasizes that even though the kernel function is used to derive the sampling distribution yet the operation is data independent.*

The randomized feature map scheme is summarized in Figure 1 and put in contrast with an exact kernel machine.

2.2 Integral representation: Fourier features

Kernel Integral representations can be used to produce such random features if the performed transformations yield probability measures [1] from which samples can be drawn in a Monte-Carlo fashion. If the Kernel verifies the assumptions below, the Bochner theorem ensures that the Inverse Fourier transform is an appropriate integral representation.

Under the assumptions of theorem 1, the Bochner theorem implies that K can be seen as a single variable function. Its Fourier transform $\mathcal{F}_K$ is a probability measure:

$$\mathcal{F}_K(\mathbf{w}) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} K(\mathbf{z}) e^{j\langle \mathbf{w}, \mathbf{z} \rangle} d\mathbf{z} = p(\mathbf{w})$$

Therefore, K can be written as the inverse Fourier transform of $\mathcal{F}_K$:

$$\begin{aligned} K(\mathbf{z}) &= \int_{\mathbb{R}^d} p(\mathbf{w}) e^{-j\langle \mathbf{w}, \mathbf{z} \rangle} d\mathbf{w} \\ &= \mathbb{E}_{\mathbf{w}} \left[e^{-j\langle \mathbf{w}, \mathbf{z} \rangle} \right] \\ &= \mathbb{E}_{\mathbf{w}} \left[\cos(\langle \mathbf{w}, \mathbf{x} - \mathbf{y} \rangle) \right] \\ &= 2\mathbb{E}_{\mathbf{w},\mathbf{b}} \left[\cos(\langle \mathbf{w}, \mathbf{x} \rangle + \mathbf{b}) \cos(\langle \mathbf{w}, \mathbf{y} \rangle + \mathbf{b}) \right] (\star) \end{aligned}$$

Where $\mathbf{w}$ is sampled from p and $\mathbf{b}$ follows a uniform distribution over $[0, 2\pi]$. The last two equalities are justified by the fact that $K(\mathbf{z})$ is real and:

$$\begin{aligned} &\mathbb{E}_{\mathbf{b} \sim \mathcal{U}(0,2\pi)} \left[2 \cos(\langle \mathbf{w}, \mathbf{x} \rangle + \mathbf{b}) \cos(\langle \mathbf{w}, \mathbf{y} \rangle + \mathbf{b}) \right] \\ &= \frac{1}{2\pi} \int_0^{2\pi} \left(\cos(\langle \mathbf{w}, \mathbf{x} + \mathbf{y} \rangle + 2\mathbf{b}) + \cos(\langle \mathbf{w}, \mathbf{x} - \mathbf{y} \rangle) \right) d\mathbf{b} \\ &= \frac{1}{2\pi} \int_0^{2\pi} \cos(\langle \mathbf{w}, \mathbf{x} + \mathbf{y} \rangle + 2\mathbf{b}) d\mathbf{b} + \cos(\langle \mathbf{w}, \mathbf{x} - \mathbf{y} \rangle) \\ &= \cos(\langle \mathbf{w}, \mathbf{x} - \mathbf{y} \rangle) \end{aligned}$$

The expectation $(\star)$ can be approximated by a simple Monte Carlo estimator where given the appropriate i.i.d samples, the i-th coordinate of $Z(\mathbf{x})$ is given by $\sqrt{\frac{2}{D}} \cos(\mathbf{w}_i^T \mathbf{x} + \mathbf{b}_i)$.

□

Theorem 1

If :

1. The Kernel K is shift-invariant: $K(\mathbf{x}, \mathbf{y}) = K(\mathbf{z})$ where $\mathbf{x} - \mathbf{y} = \mathbf{z}$
2. The function $K(\mathbf{z})$ is positive definite on $\mathbb{R}^d$ for all d.

Then K is the Fourier transform of a finite-Borel measure p on $\mathbb{R}^d$.
Moreover, if $\mathbf{w}_1, \ldots, \mathbf{w}_D$ and $\mathbf{b}_1, \ldots, \mathbf{b}_D$ are i.i.d samples drawn respectively from p and the Uniform distribution over $[0, 2\pi]$. The feature map is given by:

$$Z(\mathbf{x}) = \sqrt{\frac{2}{D}} \left[\cos {\mathbf{w}_1}^T \mathbf{x} + \mathbf{b}_1), \ldots, \cos({\mathbf{w}_D}^T \mathbf{x} + \mathbf{b}_D) \right]$$

2.3 Experiments on Radial-basis kernels

We implement in Python the algorithm in Theorem 1 and evaluate the Kernel approximation (MSE) before applying to real datasets.

We derive analytically the sampling distribution p for the Gaussian and Laplacian kernels. For the sake of brevity, details are provided in Supplementary Material.

[1] Once the said Borel measures are normalized.

Gaussian kernel

$$K(\mathbf{z}) = e^{-\gamma\|z\|_2^2} \Rightarrow p(\mathbf{w}) = \frac{1}{\sqrt{4\pi\gamma}} e^{-\frac{\mathbf{w}^2}{4\gamma}}$$

We recognize a gaussian distribution $\mathcal{N}(0, \sqrt{2\gamma} I_d)$.

Laplacian kernel

$$K(\mathbf{z}) = e^{-\gamma\|z\|_1} \Rightarrow p(w) = \frac{\gamma^2}{\pi\gamma\left(w^2 + \gamma^2\right)}$$

Where by $p(w)$ we denote the univariate distribution of a single coordinate of $\mathbf{w}$. Each component of $\mathbf{w}$ therefore follows a Cauchy distribution.

Experiments

All experiments relying on randomization are performed 10 times (MSE, SVM accuracy and computation time). The mean is plotted within its +- standard deviation interval.

On *Gisette* dataset we approximate the kernel matrix and plot the mean squared error. For different values of d, we keep the first d features to evaluate the effect of the dimension on the approximation (Figure 2). On average the approximation is better for low dimensions but at the expense of stability. The exact Laplacian kernel seems to be slower than the gaussian, which explains why it benefits more from random features in terms of computation time.

On Figure 3 we compare the exact SVM with the gaussian kernel to the random Fourier features procedure.

The results of the same experiments performed on USPS dataset are provided in Supplementary Figures 6, 7.

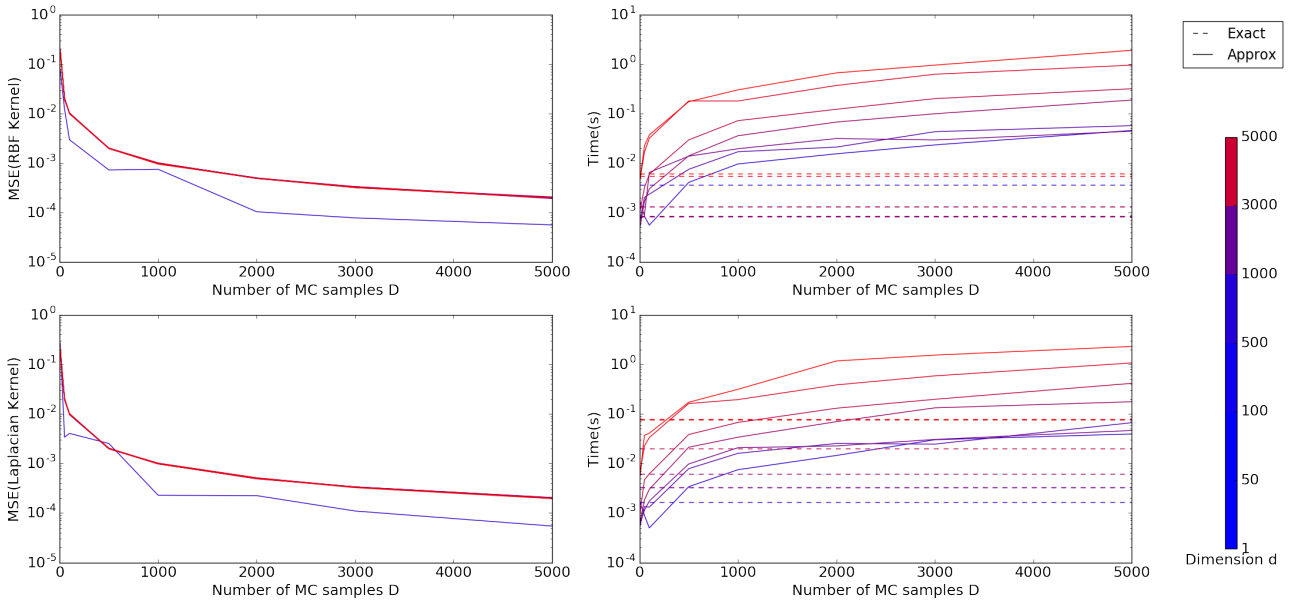


Figure 2: *MSE of kernel approximations on Gisette dataset for different sets of kept features.*

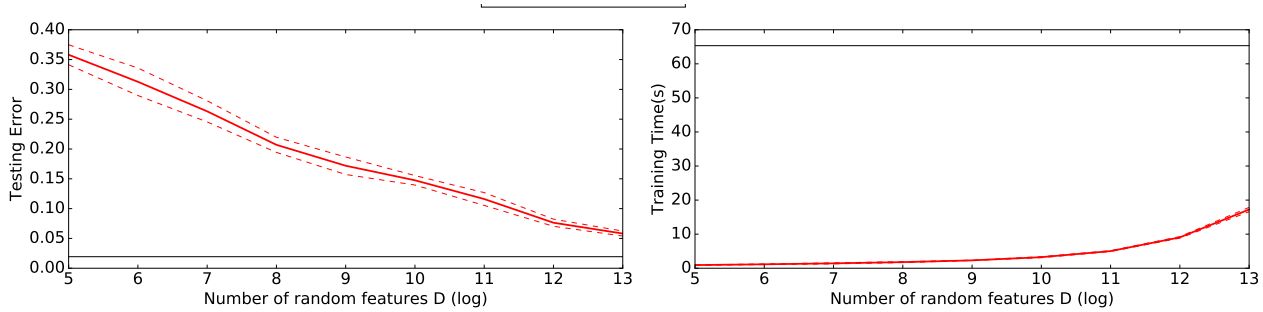


Figure 3: *Testing accuracy of SVM machines. Red: RFF + Linear SVM, dashed: +- std.* ***Back:*** *Exact Kernel SVM*

3 Spherical Random Features for polynomial kernels

3.1 The trick

Given two tuning parameters, Polynomial kernels are defined by their dot product:

$$K(\mathbf{x},\mathbf{y}) = (c + \gamma\langle \mathbf{x},\mathbf{y}\rangle)^p$$

To write it as a shift invariant kernel, data must be normalized. If $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$ then K can be written as:

$$K(\mathbf{x},\mathbf{y}) = \left(1 - \frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{a^2}\right)^p$$

In (Pennington et al, 2015) [3], the authors proved that the Fourier transform of K is not non-negative. Even though $K(z)$ is arbitrarily defined for $z > 2$ (because the data are normalized, we have $\|\mathbf{z}\| < 2 \forall \mathbf{z} \in \mathbb{R}^d$) we cannot construct a Kernel such that it its Fourier transform is a probability distribution.

The solution proposed by [3] is as follows:

1. Approximate the polynomial Kernel by a surrogate function $\hat{K}$ that is shift-invariant and positive definite on $\mathbb{R}^d \forall d$
2. Approximate $\hat{K}$ by applying Theorem 1.

It is important to note that the first approximation procedure is *offline*, i.e data independent. It only requires the knowledge of the dimension d.

Schoberg's theorem characterizes positive-definite radial functions by their integral representation:

$$f(r) = \int_{\mathbb{R}_+} e^{-r^2t^2} d\mu(t)$$

where μ is a finite non-negative Borel measure on $\mathbb{R}_+$

Given that any finite sum of gaussians verifies the Bochner theorem, approximating the polynomial kernel by a sum of gaussians seems a good idea:

$$\hat{K}(z) = \sum_{i=1}^{N} c_i e^{-\sigma_i^2 z^2}$$

Its Fourier transform is a probability distribution if c is positive and is given by:

$$p(w) = \sum_{i=1}^{N} c_i \left(\frac{1}{\sqrt{2}\sigma_i}\right)^d e^{-\frac{w^2}{4\sigma_i^2}}$$

However, for a good approximation, we let $c \in \mathbb{R}^N$ and clip the negative values of p:

$$p(w) = \max\left(0, \sum_{i=1}^{N} c_i \left(\frac{1}{\sqrt{2}\sigma_i}\right)^d e^{-\frac{w^2}{4\sigma_i^2}}\right)$$

Let θ be the 2N-concatenated vector (σ, c). We obtain $\hat{K}_\theta$ by solving the minimization problem:

$$\min_\theta \frac{1}{2} \int_0^2 \left[K(z) - \hat{K}_\theta(z)\right]^2 dz$$

Where: $\hat{K}_\theta(z)$ can be approximated on a grid:

$$\hat{K}_\theta(z) = \int_{\mathbb{R}_+} wp(w) \left(\frac{w}{z}\right)^{d/2-1} \mathcal{J}_{d/2-1}(wz)dw$$

3.2 Implementation

Implementing the formulas above and their gradients is not obvious because of numerical instability (the exponentials explose rapidly). [3] circumvent this problem by introducing a scaled Bessel function and **including an exponential term in w in the Fourier transform**. We express some concern about their implementation in Section 4. We adapted the Matlab functions of [3] [2] in as two classes[3]:

ApproxKernel: Joins matlab functions as class methods to solve the optimization problem (L-BFGS) given the parameters of the kernel and a random initialization. After estimating the optimal parameter θ, an evaluation of the probability distribution p is carried out on a univariate grid.

SRF: Adapted from our RFF class used with Gaussian and Laplacian kernels in order to sample from an empirical distribution.

3.3 Experiments

We reproduce the experiments of [3] on USPS and Gisette datasets. We notice however that even their implementation does not avoid overflows for large values of d (dimensionality). For Gisette dataset, we perform a PCA projection on 2049 components, 2049 being the largest number for which the algorithm is numerically stable.

Mean curves over 10 runs of the algorithm within their ± std intervals are more intuitive than tables: Figures (4, 5) show MSE, SVM accuracy and computation time comparisons with an exact SVM. As mentioned in [3], we can appreciate how the kernel approximation improves with P in terms of both variance and accuracy.

We notice that random features yield a similar (or even higher) accuracy than exact SVMs. Yet we must keep in mind that reducing training time is the main purpose: curves must be analyzed in parallel. It is also important to notice that the gap between time curves in Gisette dataset is larger than that of USPS dataset which must be due to the high dimension of Gisette (since it contains less samples).

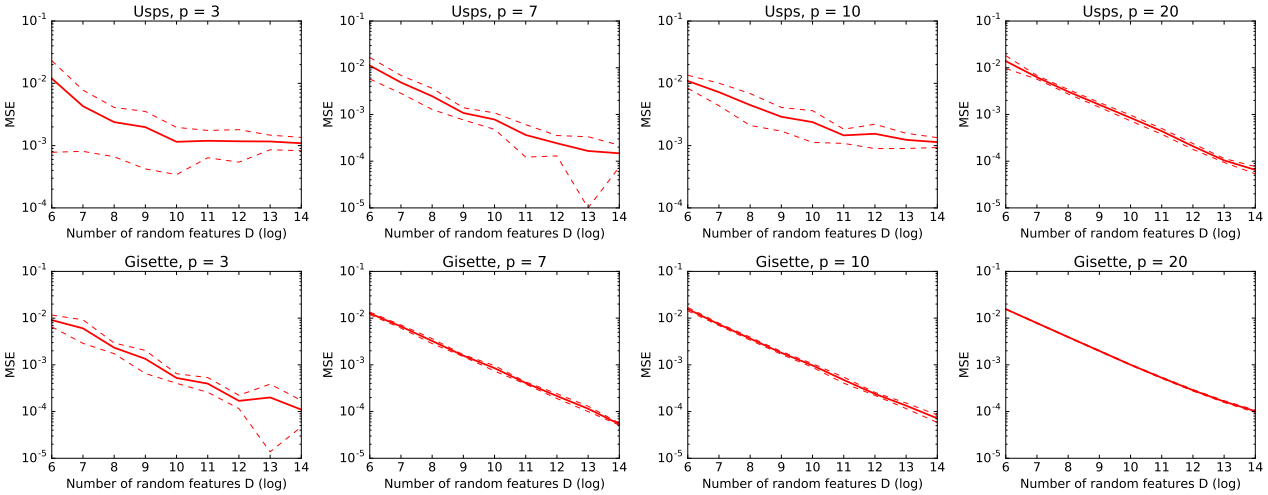


Figure 4: *MSE of kernel approximations on Gisette dataset of a polynomial kernel a = 4 and different degrees P. Dashed curves show the +- std interval computed on 10 runs.*

[2] See github repository of Felix Yu https://github.com/felixyu/SRF

[3] All code and experiments are uploaded, see last Section.

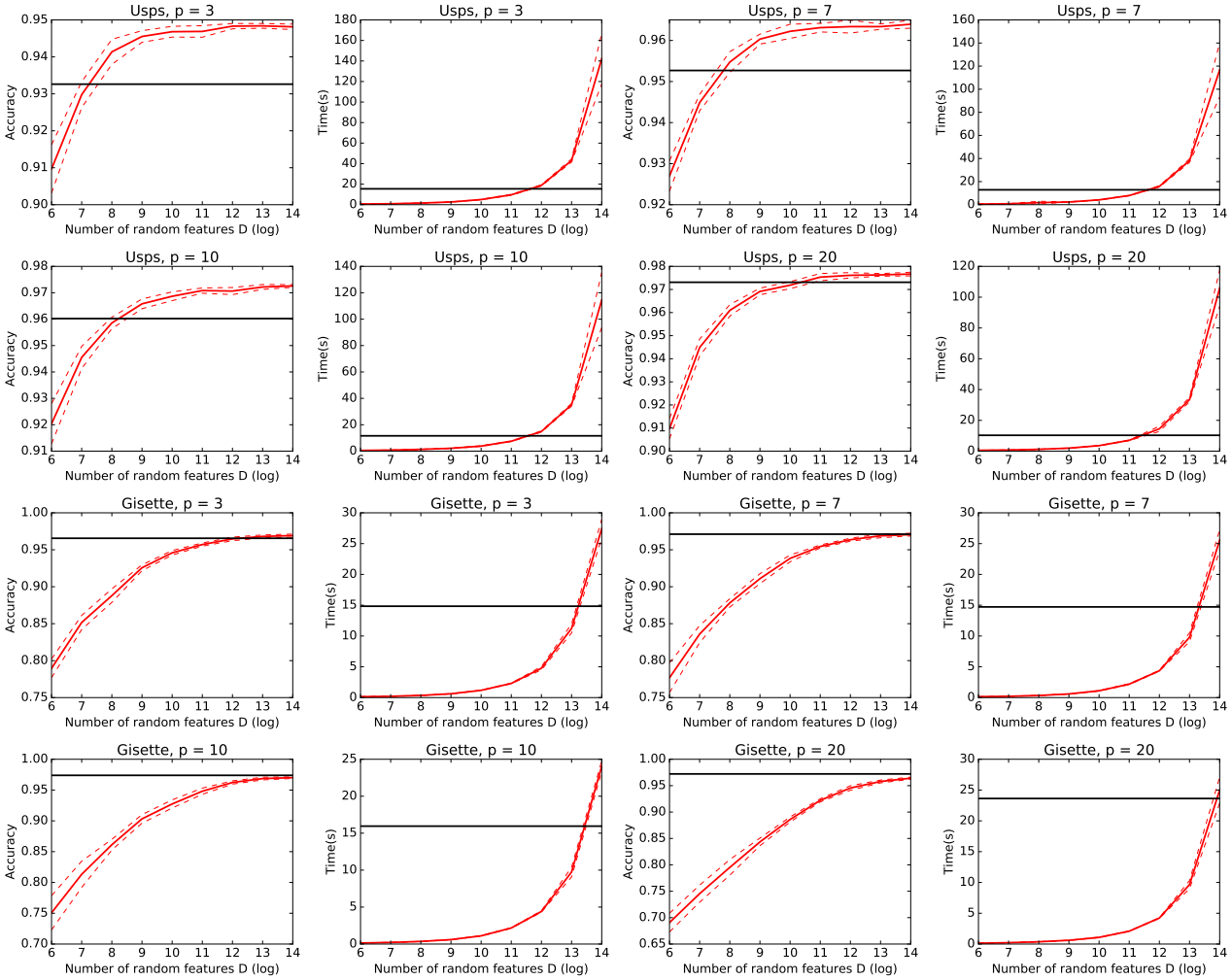


Figure 5: *Testing accuracy and computation time of SVM machines. Red: KernelApprox + SRFF + Linear SVM, dashed: +- std.* ***Back****: Exact Kernel SVM . SVM tuning parameter C = 1. Dashed curves show the +- std interval computed on 10 runs.*

4 Comments and discussion

We included the curves data in Supplementary Figures 1, 2 for the sake of comparison. The numbers are slightly different but we notice similar patterns: better approximation for higher polynomial degrees p and large number of random features D (obviously). The proposed implementation needs further improvement: The proposed code does not follow the formulas presented in the paper [3] and can be very misleading as according to the Matlab code, the approximated PDF yielded by the functions **does not correspond** to that of:

$$p(w) = \max\left(0, \sum_{i=1}^{N} c_i \left(\frac{1}{\sqrt{2}\sigma_i}\right)^d e^{-\frac{w^2}{4\sigma_i^2}}\right)$$

but to a normalized version of:

$$g(w) = \max\left(0, \sum_{i=1}^{N} c_i \frac{1}{\Gamma(d/2)} \left(\frac{w}{2\sigma_i}\right)^{d-1} e^{-\frac{w^2}{4\sigma_i^2}}\right)$$

The added terms are removed from the Bessel function so as to obtain the same objective function. When performing the RFF sampling from the pdf (Let X be such a sample), the authors

draw and i.i.d sample Y from a Gaussian distribution and multiply it elementwise to obtain $W = X \odot Y$.

Moreover, to estimate a cutoff W_{max} after which the integral value is infinitesimal, the authors use again the same pdf vector. The PDF plots in [3] are therefore wrong and do not correspond to their equations.

These differences between the paper and the code are probably a trick to avoid numerical problems. Yet clarification is needed.

Finally, unlike what was observed by the authors of [3], we find the minimization problem to be quite sensitive to initialization.

Code, Experiments

All code and experiments can be found in https://github.com/hichamjanati/srf

Abbreviations

RFF: Random Fourier Features **SRF**: Spherical random features **CDF**: Cumulative distribution function **PCA**: Principal component analysis **PDF:** Probability distribution function **SVM**: Support vector machine

References

[1] Drineas and Mahoney. On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. *Journal of Machine Learning Research*, 2005.

[2] Ali Rahimi and Ben Recht. Random Features for Large-Scale Kernel Machines. *NIPS*, 2007.

[3] Jeffrey Pennington, Felix X. Yu, and Sanjiv Kumar. Spherical Random Features for Polynomial Kernels. *NIPS*, 2015.

[4] Purushottam Kar and Harish Karnick. Random Feature Maps for Dot Product Kernels. *Journal of Machine Learning Research*, 2012.

[5] Pham and Pagh. Fast and Scalable Polynomial Kernels via Explicit Feature Maps. *KDD*, 2013.

Supplementary material

A Supplementary Information

A.1 Sampling distributions

Bochner theorem ensures that in random Fourier features framework the sampling distribution is given by:

$$p(\mathbf{w}) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} K(\mathbf{z}) e^{j\langle \mathbf{w}, \mathbf{z} \rangle} d\mathbf{z}$$

Let's derive that of the Gaussian and Laplacian kernels:

Gaussian Kernel

Let K be a gaussian kernel with a tuning parameter γ and $\mathbf{z} \in \mathbb{R}^d$:

$$K(\mathbf{z}) = \exp\left(-\gamma \|z\|_2^2\right)$$

$$\begin{aligned}
p(\mathbf{w}) &= \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \exp\left(-\gamma \|z\|_2^2 + j\langle \mathbf{w}, \mathbf{z} \rangle\right) d\mathbf{z} \\
&= \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \exp\left(-\gamma \sum_{i=1}^{d} z_i^2 + j w_i z_i\right) d\mathbf{z} \\
&= \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \prod_{i=1}^{d} \exp\left(-\gamma z_i^2 + j w_i z_i\right) d\mathbf{z} \\
&= \prod_{i=1}^{d} \frac{1}{2\pi} \int_{\mathbb{R}} \exp\left(-\gamma z_i^2 + j w_i z_i\right) dz_i
\end{aligned}$$

It boils down to the 1D integral :

$$\begin{aligned}
F(w) &= \frac{1}{2\pi} \int_{\mathbb{R}} \exp\left(-\gamma z^2 + jwz\right) dz \\
&= \frac{1}{2\pi} e^{-\frac{w^2}{4\gamma}} \int_{\mathbb{R}} \exp\left(-(\sqrt{\gamma} z - j \frac{w}{2\sqrt{\gamma}})^2\right) dz \\
&= \frac{1}{2\pi} e^{-\frac{w^2}{4\gamma}} I(w)
\end{aligned}$$

The function $f : (w, z) \mapsto \exp\left(-(\sqrt{\gamma} z - j \frac{w}{2\sqrt{\gamma}})^2\right)$ is in $\mathcal{C}^\infty(\mathbb{R}^2)$ and its partial derivative $\frac{\partial f}{\partial w} = j \frac{1}{2\sqrt{\gamma}} (\sqrt{\gamma} z - j \frac{w}{2\sqrt{\gamma}}) \exp\left(-(\sqrt{\gamma} z - j \frac{w}{2\sqrt{\gamma}})^2\right)$ as a function of z is continuous and integrable on $\mathbb{R}$. Thus, we can derive under the integral and:

$$
\begin{aligned}
I'(w) &= \int_{\mathbb{R}} j\frac{1}{2\sqrt{\gamma}}(\sqrt{\gamma}z - j\frac{w}{2\sqrt{\gamma}})\exp\left(-(\sqrt{\gamma}z - j\frac{w}{2\sqrt{\gamma}})^2\right)dz \\
&= -j\frac{1}{2\gamma}\left[\exp\left(-(\sqrt{\gamma}z - j\frac{w}{2\sqrt{\gamma}})^2\right)\right]_{\mathbb{R}} \\
&= 0
\end{aligned}
$$

And since $I(0) = \int_{\mathbb{R}} \exp\left(-\gamma z^2\right) dz = \sqrt{\frac{\pi}{\gamma}}$, the 1D distribution is given by :

$$F(w) = \frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{2\gamma}}e^{-\frac{w^2}{4\gamma}}$$

we recognize a Gaussian distribution $\mathcal{N}(0, \sqrt{2\gamma})$ □

Laplacian kernel

The integration is performed immediately.

B Supplementary Figures

B.1 Gaussian Kernel, USPS

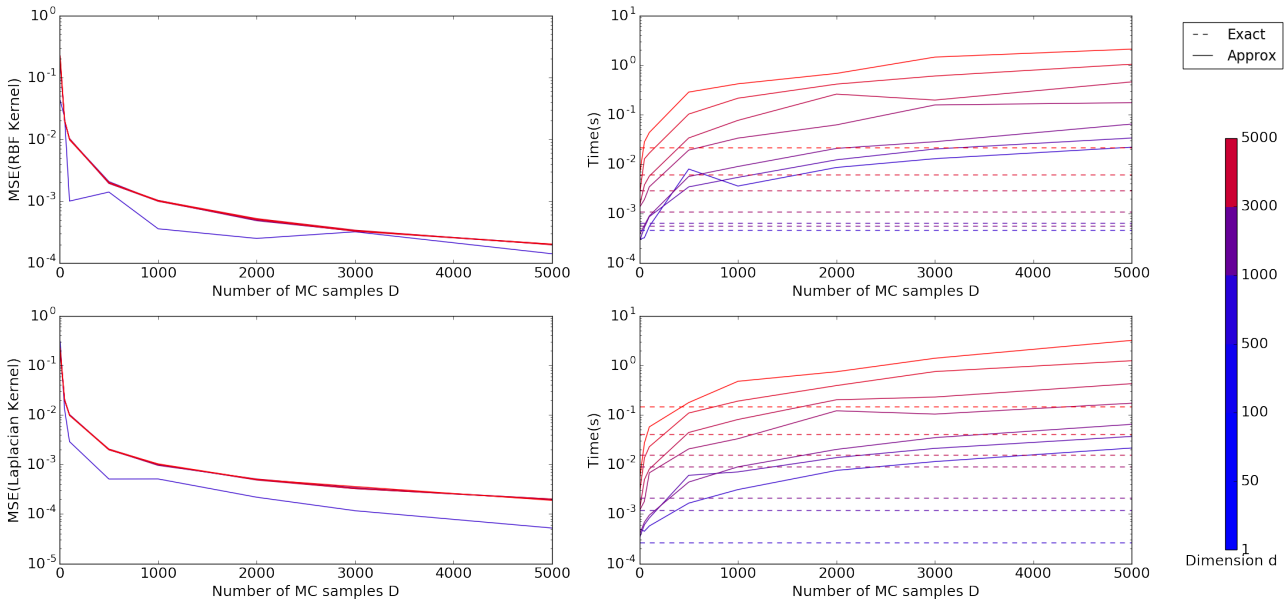


Figure 6: *MSE of kernel approximations on USPS dataset for different sets of kept features.*

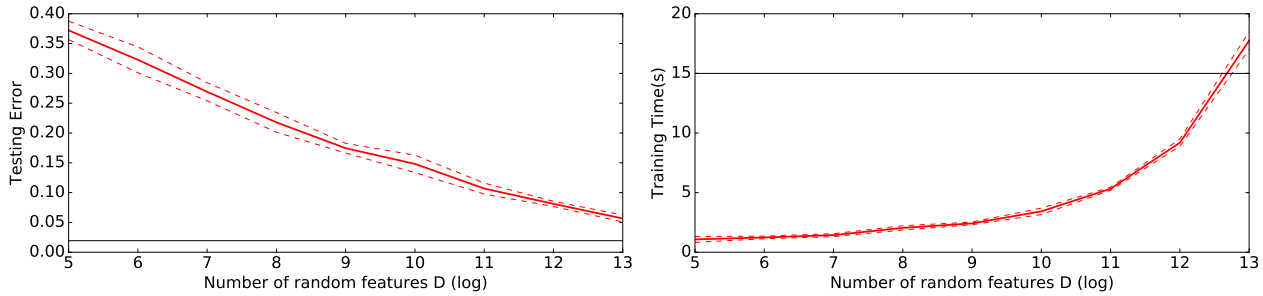


Figure 7: *Testing accuracy of SVM machines. Red: RFF + Linear SVM, dashed: +- std.* ***Back:*** *Exact Kernel SVM on USPS dataset.*

B.2 Polynomial kernels experiments

USPS

P	$D = 2^9$	$D = 2^{10}$	$D = 2^{11}$	$D = 2^{12}$	$D = 2^{13}$	$D = 2^{14}$	Exact
p = 3 Accuracy	92.98%± 0.35	94.14%± 0.34	94.55%± 0.16	94.68%± 0.15	94.69%± 0.16	94.83%± 0.07	93.26
Time in s	92.71s ± 6.51	158.02s ± 11.46	263.47s ± 6.61	500.93s ± 15.5	962.58s ± 47.03	1883.07s ± 59.01	1546.57
p = 7 Accuracy	94.49%± 0.2	95.47%± 0.25	96.03%± 0.12	96.22%± 0.17	96.31%± 0.1	96.34%± 0.16	95.27
Time in s	81.51s ± 11.31	179.49s ± 115.64	234.12s ± 15.07	416.62s ± 16.39	793.27s ± 14.07	1582.71s ± 57.32	1289.43
p = 10 Accuracy	94.55%± 0.42	95.86%± 0.22	96.58%± 0.18	96.87%± 0.17	97.08%± 0.1	97.06%± 0.13	96.02
Time in s	77.91s ± 12.84	134.0s ± 10.96	214.82s ± 19.92	383.55s ± 24.26	751.5s ± 18.58	1488.49s ± 47.46	1160.42
p = 20 Accuracy	94.49%± 0.37	96.09%± 0.25	96.91%± 0.14	97.19%± 0.15	97.53%± 0.16	97.61%± 0.12	97.31
Time in s	72.06s ± 10.72	117.82s ± 9.4	195.59s ± 11.91	356.89s ± 10.97	704.06s ± 18.88	1450.47s ± 171.35	1029.53

Table 1: *SVM Accuracy on testing set of USPS for different values of D and P. Exact column shows the accuracy of the Exact SVM*

Gisette

P	$D = 2^9$	$D = 2^{10}$	$D = 2^{11}$	$D = 2^{12}$	$D = 2^{13}$	$D = 2^{14}$	Exact
p = 3 Accuracy	85.17%± 0.95	88.79%± 0.91	92.57%± 0.4	94.59%± 0.31	95.71%± 0.21	96.47%± 0.25	96.56
Time in s	20.49s ± 0.77	35.4s ± 1.29	61.85s ± 1.68	117.93s ± 3.37	230.04s ± 4.69	476.94s ± 26.91	1483.61
p = 7 Accuracy	83.59%± 1.16	87.81%± 0.56	91.09%± 0.67	93.83%± 0.46	95.44%± 0.2	96.36%± 0.2	97.11
Time in s	20.83s ± 0.76	34.09s ± 1.25	59.19s ± 2.15	110.49s ± 4.24	218.44s ± 7.3	435.85s ± 6.25	1472.3
p = 10 Accuracy	81.32%± 2.13	86.13%± 0.87	90.32%± 0.69	92.74%± 0.66	94.81%± 0.5	96.23%± 0.26	97.39
Time in s	21.72s ± 2.0	34.98s ± 1.0	59.85s ± 1.43	110.8s ± 3.56	217.2s ± 4.1	440.74s ± 10.19	1595.16
p = 20 Accuracy	74.54%± 1.61	79.52%± 1.48	84.34%± 0.72	88.59%± 0.48	92.22%± 0.3	94.56%± 0.43	97.22
Time in s	21.79s ± 0.77	36.62s ± 0.8	60.55s ± 1.87	108.02s ± 3.62	208.81s ± 2.87	422.24s ± 5.97	2362.91

Table 2: *SVM Accuracy on testing set of transformed Gisette (PCA on 2049 components) for different values of D and P. Exact column shows the accuracy of the Exact SVM*