



Random Features for Large-Scale Kernel Machines¹

Kailong Wang¹

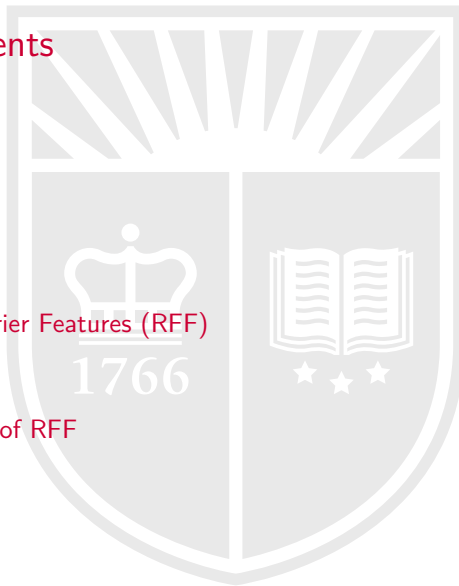
¹Rutgers University

August 8, 2023

¹Ali Rahimi and Benjamin Recht. “Random Features for Large-Scale Kernel Machines”. In: 20 (2007). Ed. by J. Platt et al. URL: https://proceedings.neurips.cc/paper_files/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf.

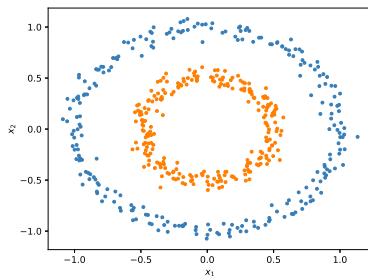
Table of Contents

- 1 Motivation
- 2 Random Fourier Features (RFF)
- 3 Convergence of RFF



Linear Non-separable Problem

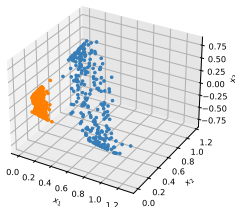
Consider a binary classification problem with non-linear samples.



e.g. For the above dataset $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2]$ where column vector $\mathbf{x}_i \in \mathbb{R}^N$, a linear decision boundary does not exist.

Lifting

One idea is **LIFTING** the samples into a high dimensional space in which the samples are linearly separable.



In this case, the function $\phi(\mathbf{X}) = [\mathbf{x}_1 \circ \mathbf{x}_1, \mathbf{x}_2 \circ \mathbf{x}_2, \sqrt{2}\mathbf{x}_1 \circ \mathbf{x}_2]$, where \circ is the Hadamard product, lifts the samples into \mathbb{R}^3 and the samples are linearly separable.



SVM²

The idea of lifting has been implemented in many classification algorithms such as *support vector machine* (SVM).

Dual Problem of SVM

$$\max_{\alpha} \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_n \alpha_m y_n y_m \langle \mathbf{x}_n, \mathbf{x}_m \rangle$$

Dual Problem with Lifting

$$\max_{\alpha} \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_n \alpha_m y_n y_m \langle \phi(\mathbf{x}_n), \phi(\mathbf{x}_m) \rangle$$

This is the hard-margin SVM. The soft-margin SVM is similar.

²Stephen Boyd and Lieven Vandenberghe. "Convex optimization". In: (2004).



Curse of Dimensionality–Type I

$$\begin{aligned}\langle \phi(\mathbf{x}_n), \phi(\mathbf{x}_m) \rangle &= \begin{bmatrix} x_{n,1}^2 & x_{n,2}^2 & \sqrt{2}x_{n,1}x_{n,2} \end{bmatrix} \begin{bmatrix} x_{m,1}^2 & x_{m,2}^2 & \sqrt{2}x_{m,1}x_{m,2} \end{bmatrix}^\top \\ &= x_{n,1}^2 x_{m,1}^2 + x_{n,2}^2 x_{m,2}^2 + 2x_{n,1}x_{n,2}x_{m,1}x_{m,2}\end{aligned}$$

As shown in the given example, the inner product (a constant value) of the lifted vector has computational complexity depends on lifted dimension. For a function lifts the original vector space to a much higher dimension, such a calculation can be computationally thirsty. Alternatively, this can be done as follows, whose computational complexity only depends on the dimension of the original vector space.

$$\begin{aligned}(\langle \mathbf{x}_n, \mathbf{x}_m \rangle)^2 &= ([x_{n,1} \ x_{n,2}][x_{m,1} \ x_{m,2}]^\top)^2 \\ &= (x_{n,1}x_{m,1} + x_{n,2}x_{m,2})^2 \\ &= x_{n,1}^2 x_{m,1}^2 + x_{n,2}^2 x_{m,2}^2 + 2x_{n,1}x_{n,2}x_{m,1}x_{m,2} \\ &= \langle \phi(\mathbf{x}_n), \phi(\mathbf{x}_m) \rangle\end{aligned}$$



Kernel Trick

The type of function, such as $(\langle \cdot, \cdot \rangle)^2$, that provides a computationally efficient way to compute the inner product in the high dimensional space is called a **Kernel Function**.

$$K(\cdot, \cdot) = \langle \phi(\cdot), \phi(\cdot) \rangle$$

The matrix that is formed by stacking the kernel function for all samples is called the **Kernel Matrix** or **Gram Matrix K**,

$$\mathbf{K}_{nm} \equiv K(\mathbf{x}_n, \mathbf{x}_m).$$

Some kernel functions can lift the original vector space to an infinite dimensional space. The algorithms involve kernel trick is called **Kernel Machines**.



Curse of Dimensionality–Type II

Another famous kernel machine is kernel ridge regression (KRR). With $\mathbf{y} \in \mathbb{R}^N$, $\mathbf{X} \in \mathbb{R}^{N \times d}$, and $\phi_{d \rightarrow k}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^k$, the loss function is

$$\mathcal{L}(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} (\mathbf{y} - \phi(\mathbf{X})\mathbf{w})^\top (\mathbf{y} - \phi(\mathbf{X})\mathbf{w}) + \lambda \mathbf{w}^\top \mathbf{w}.$$

The normal equation of KRR is

$$\begin{aligned} \mathbf{w} &= (\phi(\mathbf{X})^\top \phi(\mathbf{X}) + \lambda \mathbf{I}_k)^{-1} \phi(\mathbf{X})^\top \mathbf{y} \\ &= (\mathbf{K} + \lambda \mathbf{I}_k)^{-1} \phi(\mathbf{X})^\top \mathbf{y}. \end{aligned}$$

Solving this problem requires $\Theta(k^3)$ time and $\Theta(k^2)$ memory.

Motivation

Can we find a way to construct the **Kernel Matrix**, which is equivalent to lifting \mathbf{X} to \mathbb{R}^s with $d < s \ll k$, while not sacrifices model performance?

Some Prerequisites

Definition: Shift Invariant Kernel (Radial Basis Function (RBF))

A kernel function $K(\mathbf{x}_n, \mathbf{x}_m)$ is called **shift invariant** if it can be written as $K(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n - \mathbf{x}_m)$ for some function $k(\cdot)$ (e.g. $K_{Gaussian}(\mathbf{x}_n, \mathbf{x}_m) = \exp(-\gamma \|\mathbf{x}_n - \mathbf{x}_m\|_2^2)$).

Mercer's Theorem

A continuous function $K(\mathbf{x}_n, \mathbf{x}_m)$ is a valid kernel function if and only if the kernel matrix \mathbf{K} is **positive semi-definite**.

Bochner's Theorem

A continuous function $k(\cdot)$ is **positive semi-definite** if and only if it is the Fourier transform of a non-negative measure.



Random Fourier Features

Conclusion

A continuous **shift invariant** kernel $K(\mathbf{x}_n, \mathbf{x}_m)$, which is **positive semi-definite** (Mercer's Theorem), is the Fourier transform of a non-negative measure $p(\cdot)$.

$$\phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m) = K(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n - \mathbf{x}_m) \quad (1)$$

$$= \int_{\mathbb{R}^d} p(\omega) \exp(i\omega^\top (\mathbf{x}_n - \mathbf{x}_m)) d(\mathbf{x}_n - \mathbf{x}_m) \quad (2)$$

$$= \mathbb{E}_\omega [\xi_\omega(\mathbf{x}_n)^\mathsf{H} \xi_\omega(\mathbf{x}_m)] \quad (3)$$

Here $\xi_\omega(\mathbf{x}) = \exp(i\omega^\top (\mathbf{x}))$.



Random Fourier Features

Since both the $p(\cdot)$ and $k(\Delta)$ are real-valued, we can replace $\exp(i\omega^\top(\mathbf{x}))$ with $z_\omega(\mathbf{x}) = \begin{bmatrix} \cos(\omega^\top(\mathbf{x})) \\ \sin(\omega^\top(\mathbf{x})) \end{bmatrix} = \sqrt{2} \cos(\omega^\top(\mathbf{x}) + b)$. Then eq. (3) becomes

$\mathbb{E}_\omega[z_\omega(\mathbf{x}_n)^\top z_\omega(\mathbf{x}_m)]$, which means $z_\omega(\mathbf{x}_n)^\top z_\omega(\mathbf{x}_m)$ is an unbiased estimator of $\phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m)$.

Let ω been drawn from $p(\omega)$ and b been uniformly drawn from $[0, 2\pi]$, the calculation of $\phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m)$ transforms into a sampling problem.

To further reduce the variance of the estimator, we can randomly draw s samples of ω and b , normalize each corresponding $z_\omega(\mathbf{x})$ by \sqrt{s} , and concatenate them into one vector. Then the inner product

$$z(\mathbf{x}_n)^\top z(\mathbf{x}_m) = \frac{1}{s} \sum_{j=1}^s z_{\omega_j}(\mathbf{x}_n)^\top z_{\omega_j}(\mathbf{x}_m)$$



Algorithm

Algorithm Random Fourier Features

Require: A shift invariant kernel $K(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n - \mathbf{x}_m) = k(\Delta)$.

Ensure: A randomized feature map $z(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^s$ so that $z(\mathbf{x}_n)^\top z(\mathbf{x}_m) \approx K(\mathbf{x}_n, \mathbf{x}_m)$.

Compute the Fourier transform $p(\cdot)$ of the kernel K : $p(\omega) = \frac{1}{2\pi} \int \exp(-i\omega^\top \Delta) k(\Delta) d\Delta$

Draw s i.i.d. samples $\omega_1, \omega_2, \dots, \omega_s \in \mathbb{R}^d$ from $p(\omega)$ and s i.i.d. samples $b_1, b_2, \dots, b_s \in [0, 2\pi]$.

Let $z(\mathbf{x}) \equiv \sqrt{\frac{2}{s}} [\cos(\omega_1^\top \mathbf{x} + b_1), \cos(\omega_2^\top \mathbf{x} + b_2), \dots, \cos(\omega_s^\top \mathbf{x} + b_s)]^\top$



Common RFF

| Kernel | $K(\Delta)$ | $p(\omega)$ |
|-----------|----------------------------------|--|
| Gaussian | $\exp(-\gamma \ \Delta\ _2^2)$ | $(2\pi)^{-\frac{s}{2}} \exp(-\gamma \ \omega\ _2^2)$ |
| Laplacian | $\exp(-\ \Delta\ _1)$ | $\prod_d (\pi(1 + \omega_d^2))^{-1}$ |
| Cauchy | $\prod_d 2(1 + \Delta_d^2)^{-1}$ | $\exp(-\ \omega\ _1)$ |



Convergence with Hoeffding's Inequality³

Hoeffding's Inequality

Let X_1, X_2, \dots, X_N be independent random variables. Assume that $X_i \in [m_i, M_i]$ for every i . Then, for any $\epsilon > 0$, we have

$$\mathbb{P}\left(\left|\sum_{i=1}^N (X_i - \mathbb{E}[X_i])\right| \geq \epsilon\right) \leq 2 \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^N (M_i - m_i)^2}\right).$$

Bound for *any* pair of samples \mathbf{x}_n and \mathbf{x}_m

Given z_w is bounded random variable between $[-\sqrt{2/s}, \sqrt{2/s}]$, with Hoeffding's Inequality, we have

$$\mathbb{P}(|z(\mathbf{x}_n)^T z(\mathbf{x}_m) - K(\mathbf{x}_n, \mathbf{x}_m)| \geq \epsilon) \leq 2 \exp\left(-\frac{s\epsilon^2}{4}\right).$$

³**Roman Vershynin**. “High-Dimensional Probability: An Introduction with Applications in Data Science”. In: (2018).

Convergence

Bound for the Kernel Matrix

Let \mathcal{M} be a compact subset of \mathbb{R}^d with diameter $\text{diam}(\mathcal{M})$. Then, for the mapping z defined in Algorithm 1, we have

$$\begin{aligned} & \mathbb{P}\left(\sup_{x, y \in \mathcal{M}} |z(\mathbf{x}_n)^\top z(\mathbf{x}_m) - K(\mathbf{x}_n, \mathbf{x}_m)| \geq \epsilon\right) \\ & \leq 2^8 \left(\frac{\sigma_{p(\cdot)} \text{diam}(\mathcal{M})}{\epsilon}\right)^2 \exp\left(-\frac{s\epsilon^2}{4(d+2)}\right). \end{aligned}$$

The $\sigma_{p(\cdot)}^2 = \mathbb{E}_{p(\cdot)}[\omega^\top \omega]$ is the second moment of the Fourier transform of the $K(\cdot, \cdot)$.

The proof of this bound uses the knowledge of ϵ -net and ϵ -covering number.

Experiment

| Datasets | Fourier + LS | SVM |
|--|----------------------------|----------------------------------|
| CPU regression 6,500 instances; 21 dims | 3.6% 20 secs k = 300 | 5.5% 51 secs |
| Census regression 18,000 instances; 119 dims | 5% 36 secs k = 500 | 8.8% 7.5 mins |
| Adult classification 32,000 instances; 123 dims | 14.9% 9 secs k = 500 | 14.8% 73 mins |
| KDDCUP99 classification 4,900,000 instances; 127 dims | 7.3% 1.5 mins k = 50 | 6.2% (18%) 1.4 secs (20 secs) |

Table: Comparison of testing error and training time between ridge regression with random features and Support Vector Machine. For classification tasks, the percent of testing points incorrectly predicted is reported. For regression tasks, the RMS error normalized by the norm of the ground truth is reported.