Supplementary Materials for the Implementation of Adaptive Resampling for Random Fourier Feature Model

Xin Huang

June 20, 2025

1 Derivation of the Gradient and Hessian of the Loss Function with regularization

We consider the loss function:

$$L(\hat{\beta}) = \underbrace{\frac{1}{J}\sum_{j=1}^{J}|\beta(x_j) - y_j|^2}_{=:L_1(\hat{\beta})} + \underbrace{\lambda_1 \sum_{k=1}^{K}|\hat{\beta}_k|^2}_{=:L_2(\hat{\beta})} + \underbrace{\lambda_2(\sum_{k=1}^{K}|\hat{\beta}_k|^2)^2}_{=:L_3(\hat{\beta})},$$

where

$$\beta(x) = \sum_{k=1}^{K}\hat{\beta}_k\phi_k(x) = \hat{\beta}^T\phi(x),$$

denotes the model function, with

$$\hat{\beta} = (\hat{\beta}_1, \ldots, \hat{\beta}_K)^T \in \mathbb{C}^K$$

representing the amplitude coefficients to be optimized, and

$$\phi(x) = (\phi_1(x), \ldots, \phi_K(x))^T \in \mathbb{C}^K$$

stands for the basis functions corresponding to the random Fourier feature frequencies $\omega = (\omega_1, \ldots, \omega_K)$ by

$$\phi_k(x) = \mathrm{e}^{\mathrm{i}\omega_k \cdot x} = \cos(\omega_k \cdot x) + \mathrm{i}\sin(\omega_k \cdot x),$$

using the training data set $\{x_j, y_j\}_{j=1}^{J}$ and regularization parameters λ_1, λ_2.

We can write the real and imaginary parts of the model function separately as

$$\hat{\beta} = a + \mathrm{i}b, \quad \text{with } a, b \in \mathbb{R}^K.$$

and

$$\phi(x) = u(x) + \mathrm{i}v(x), \quad \text{with } u, v \in \mathbb{R}^K\,,$$

where

$$u(x) = (u_1(x), \ldots, u_K(x))^T \quad \text{and} \quad v(x) = (v_1(x), \ldots, v_K(x))^T\,,$$

with

$$u_k(x) = \cos(\omega_k \cdot x) \quad \text{and} \quad v_k(x) = \sin(\omega_k \cdot x)$$

represents the real and imaginary parts of $\phi(x)$, respectively.

Then we can rewrite the model function as

$$\beta(x) = \hat{\beta}^T \phi(x) = (a^T u(x) - b^T v(x)) + \mathrm{i}(a^T v(x) + b^T u(x))\,.$$

Given the real-valued label data $y = (y_1, \ldots, y_J) \in \mathbb{R}^J$, we can write the first term of the loss function as

$$L_1(\hat{\beta}) = \frac{1}{J}\sum_{j=1}^{J} |\hat{\beta}^T\phi(x) - y_j|^2 = \frac{1}{J}\sum_{j=1}^{J} |(a^T u(x_j) - b^T v(x_j)) - y_j|^2 + |(a^T v(x_j) + b^T u(x_j))|^2\,.$$

By introducing the short-hand notations for the stacked training data and sampled frequencies as:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_J \end{bmatrix} \in \mathbb{R}^{J\times d}, \quad \Omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_K \end{bmatrix} \in \mathbb{R}^{K\times d}\,,$$

and

$$U = \cos(\Omega X^T), \quad V = \sin(\Omega X^T), \quad \text{with } U, V \in \mathbb{R}^{K\times J}\,,$$

we can then compute the gradient of the loss function L_1 with respect to the real and imaginary part of $\hat{\beta}$ respectively as:

$$\frac{\partial L_1}{\partial a_k} = \frac{1}{J}\sum_{j=1}^{J} \big(2(a^T u(x_j) - b^T v(x_j) - y_j)u_k(x_j) + 2(a^T v(x_j) + b^T u(x_j))v_k(x_j)\big), \quad \text{for } k = 1, \ldots, K\,,$$

$$\Longrightarrow \frac{\partial L_1}{\partial a} = \frac{1}{J}\big(2U(U^T a - V^T b - y) + 2V(V^T a + U^T b)\big)\,,$$

and

$$\frac{\partial L_1}{\partial b_k} = \frac{1}{J}\sum_{j=1}^{J} \big(-2(a^T u(x_j) - b^T v(x_j) - y_j)v_k(x_j) + 2(a^T v(x_j) + b^T u(x_j))u_k(x_j)\big), \quad \text{for } k = 1, \ldots, K\,,$$

$$\Longrightarrow \frac{\partial L_1}{\partial b} = \frac{1}{J}\big(-2V(U^T a - V^T b - y) + 2U(V^T a + U^T b)\big)\,.$$

Furthermore, we can compute the Hessian of the loss function L_1 with respect to the real and imaginary part of $\hat{\beta}$ respectively as:

$$\frac{\partial^2 L_1}{\partial a_\ell \partial b_k} = \frac{1}{J}\sum_{j=1}^{J}\big(-2u_\ell(x_j)v_k(x_j) + 2v_\ell(x_j)u_k(x_j)\big), \quad \text{for } k,\ell = 1,\dots,K\,,$$

$$\Longrightarrow \frac{\partial^2 L_1}{\partial a \partial b^T} = \frac{1}{J}\big(-2UV^T + 2VU^T\big)\,,$$

$$\frac{\partial^2 L_1}{\partial b_\ell \partial a_k} = \frac{1}{J}\sum_{j=1}^{J}\big(-2v_\ell(x_j)u_k(x_j) + 2u_\ell(x_j)v_k(x_j)\big), \quad \text{for } k,\ell = 1,\dots,K\,,$$

$$\Longrightarrow \frac{\partial^2 L_1}{\partial b \partial a^T} = \frac{1}{J}\big(-2VU^T + 2UV^T\big)\,,$$

$$\frac{\partial^2 L_1}{\partial a_\ell \partial a_k} = \frac{1}{J}\sum_{j=1}^{J}\big(2u_\ell(x_j)u_k(x_j) + 2v_\ell(x_j)v_k(x_j)\big), \quad \text{for } k,\ell = 1,\dots,K\,,$$

$$\Longrightarrow \frac{\partial^2 L_1}{\partial a^2} = \frac{1}{J}\big(2UU^T + 2VV^T\big)\,,$$

and,

$$\frac{\partial^2 L_1}{\partial b_\ell \partial b_k} = \frac{1}{J}\sum_{j=1}^{J}\big(2v_\ell(x_j)v_k(x_j) + 2u_\ell(x_j)u_k(x_j)\big), \quad \text{for } k,\ell = 1,\dots,K\,,$$

$$\Longrightarrow \frac{\partial^2 L_1}{\partial b^2} = \frac{1}{J}\big(2VV^T + 2UU^T\big)\,.$$

The derivative and Hessian of the regularization terms

$$L_2(a,b) = \lambda_1(a^T a + b^T b)$$

and

$$L_3(a,b) = \lambda_2(a^T a + b^T b)^2 = \lambda_2(a^T a)^2 + 2\lambda_2(a^T a)(b^T b) + \lambda_2(b^T b)^2$$

can be computed similarly as follows:

$$\frac{\partial L_2}{\partial a} = 2\lambda_1 a\,, \quad \frac{\partial L_2}{\partial b} = 2\lambda_1 b\,,$$
$$\frac{\partial^2 L_2}{\partial a^2} = 2\lambda_1 I_K\,, \quad \frac{\partial^2 L_2}{\partial a \partial b^T} = \frac{\partial^2 L_2}{\partial b \partial a^T} = 0\,, \quad \frac{\partial^2 L_2}{\partial b^2} = 2\lambda_1 I_K\,,$$

and

$$\frac{\partial L_3}{\partial a} = 4\lambda_2(a^T a + b^T b)a\,, \quad \frac{\partial L_3}{\partial b} = 4\lambda_2(a^T a + b^T b)b\,,$$
$$\frac{\partial^2 L_3}{\partial a^2} = 4\lambda_2\big(2aa^T + (a^T a + b^T b)I_K\big)\,, \quad \frac{\partial^2 L_3}{\partial b^2} = 4\lambda_2\big(2bb^T + (a^T a + b^T b)I_K\big)\,,$$
$$\frac{\partial^2 L_3}{\partial b \partial a^T} = 8\lambda_2 b a^T\,, \quad \frac{\partial^2 L_3}{\partial a \partial b^T} = 8\lambda_2 a b^T\,.$$

The total gradient ∇L and the Hessian H of the loss function $L(\hat{\beta})$ with respect to the real and imaginary parts of $\hat{\beta}$ can then be computed as:

$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial a} \\ \frac{\partial L}{\partial b} \end{bmatrix} \in \mathbb{R}^{2K} ,$$

and

$$H = \begin{bmatrix} H_{aa} & H_{ab} \\ H_{ba} & H_{bb} \end{bmatrix} \in \mathbb{R}^{2K \times 2K} ,$$

where

$$\begin{aligned} H_{aa} &= \frac{\partial^2 L}{\partial a^2} = \frac{\partial^2 L_1}{\partial a^2} + \frac{\partial^2 L_2}{\partial a^2} + \frac{\partial^2 L_3}{\partial a^2} , \\ H_{ab} &= \frac{\partial^2 L}{\partial a \partial b^T} = \frac{\partial^2 L_1}{\partial a \partial b^T} + \frac{\partial^2 L_2}{\partial a \partial b^T} + \frac{\partial^2 L_3}{\partial a \partial b^T} , \\ H_{ba} &= \frac{\partial^2 L}{\partial b \partial a^T} = \frac{\partial^2 L_1}{\partial b \partial a^T} + \frac{\partial^2 L_2}{\partial b \partial a^T} + \frac{\partial^2 L_3}{\partial b \partial a^T} , \\ H_{bb} &= \frac{\partial^2 L}{\partial b^2} = \frac{\partial^2 L_1}{\partial b^2} + \frac{\partial^2 L_2}{\partial b^2} + \frac{\partial^2 L_3}{\partial b^2} , \end{aligned}$$

with the element in the i-th row and j-th column of matrix H_{ab} satisfying

$$(H_{ab})_{ij} = \frac{\partial^2 L}{\partial a_i \partial b_j} , \quad \text{for } i, j = 1, \ldots, K .$$

2 Additional numerial results for implementation of Algorithm 1 and Algorithm 3 of the manuscript

We tested further in the manuscript two variants of the algorithm presented in Section 4 of this code repository, namely Algorithm 1 and Algorithm 3, using simple random walk in $\mathbb{R}^d$ or applying the empirical covariance matrix for the sampling of frequencies, respectively. The corresponding numerical results for the training and validation error, together with the distribution of sampled frequencies, are shown in the followoing Figures 1 to 4.

These results show similarity to those of Algorithm 2—all of the three algorithms efficiently reduce the validation error through resampling itterations, and the distribution of final sampled frequencies shows good agreement with the analitical optimal density. Specifically for Algorithm 1, we introduced Gaussian noise in the training data y_j by:

$$y_j = f(x_j) + \xi_j , \quad \text{with } \xi_j \sim \mathcal{N}(0, s^2) ,$$

with noise level parameter $s = 0.025$, leading to noise-to-signal ratio

$$\text{NSR} = \frac{\sum_{j=1}^{J} |y_j - f(x_j)|^2}{\sum_{j=1}^{J} |y_j|^2} = \frac{\sum_{j=1}^{J} |\xi_j|^2}{\sum_{j=1}^{J} |y_j|^2} \approx 2.75 \times 10^{-2} .$$

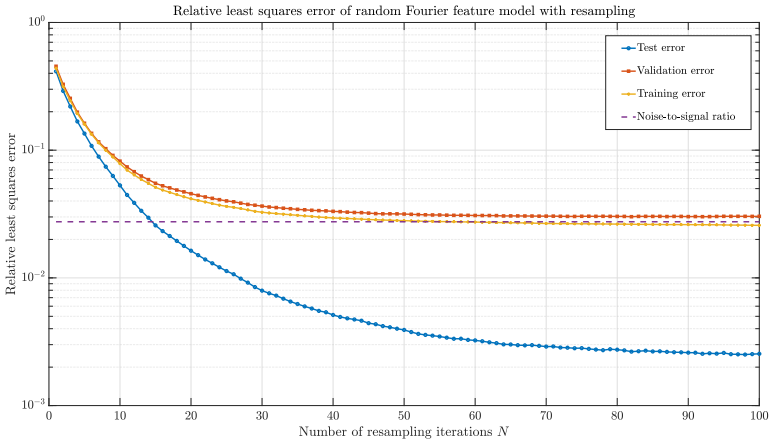
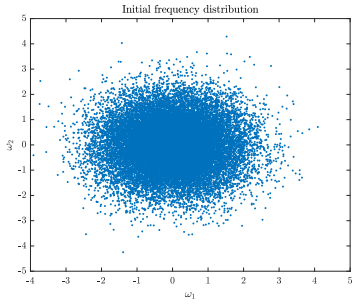
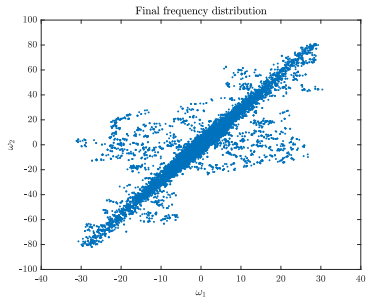


Figure 1: Relative least squares error of the random Fourier feature model $\beta(x)$ trained with resampling Algorithm 1.

The results of Figure 1 show that the resampling Algorithm 1 achieves efficient denoising of the training data, providing a test error approximately one order of magnitude smaller than the noise-to-signal ratio.

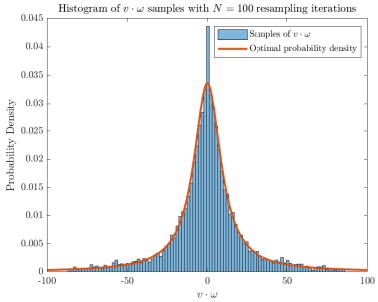


(a) Initial distribution of frequencies

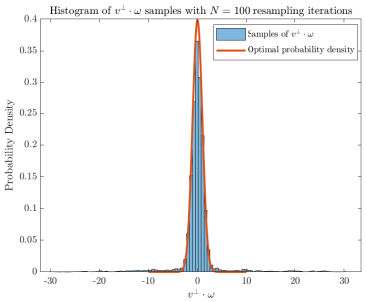


(b) final distribution of frequencies

Top row: The resampling Algorithm 1 rotates the distribution of frequency samples from initial Gaussian distribution.



(c) Histogram of rotated frequency samples $v \cdot \omega$



(d) Histogram of rotated frequency samples $v^{\perp} \cdot \omega$

Bottom row: The resampling Algorithm 1 achieves frequency distribution close to optimal distribution.

Figure 2: The frequency distribution by implementation of resampling Algorithm 1.

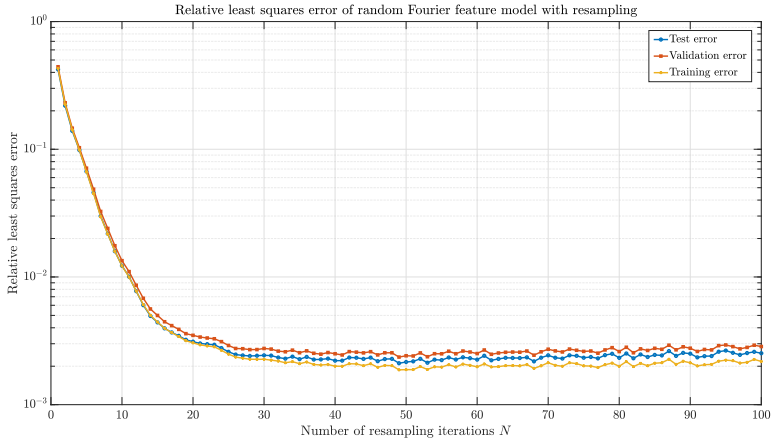
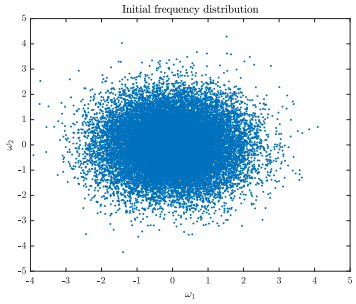
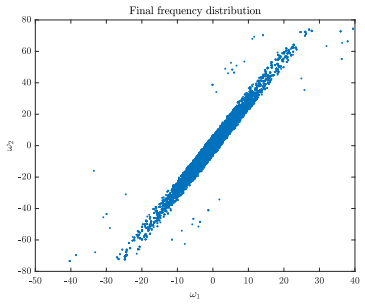


Figure 3: Relative least squares error of the random Fourier feature model $\beta(x)$ trained with resampling Algorithm 3.

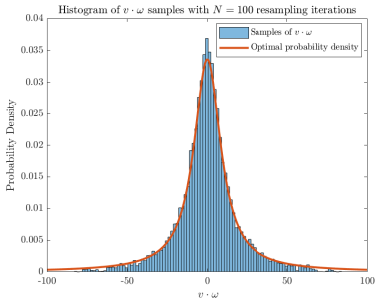


(a) Initial distribution of frequencies

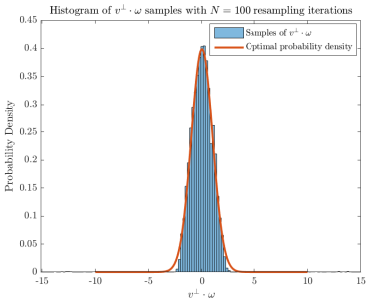


(b) final distribution of frequencies

Top row: The resampling Algorithm 3 rotates the distribution of frequency samples from initial Gaussian distribution.



(c) Histogram of rotated frequency samples $v \cdot \omega$



(d) Histogram of rotated frequency samples $v^{\perp} \cdot \omega$

Bottom row: The resampling Algorithm 3 achieves frequency distribution close to optimal distribution.

Figure 4: The frequency distribution by implementation of resampling Algorithm 3.