Low-Precision Random Fourier Features (LP-RFF)

Avner May, Jian Zhang

April 22, 2018

1 Introduction

The goal of this work is to understand whether we can train better kernel approximation models, at lower cost, by quantizing the random Fourier features [1]. We will analyze, empirically and theoretically, the performance of low-precision random Fourier features (LP-RFF), relative to the full-precision features. Specifically, we will consider quantizing the features, which take the form $z_i(x) = \sqrt{\frac{2}{m}}\cos(w_i^T x + b_i)$ into b-bits, through some form of quantization scheme (simplest: random rounding). When using b-bits to represent values in the interval $[-\sqrt{2/m}, \sqrt{2/m}]$, we will simply divide this interval evenly into $2^b - 1$ sub-intervals. Letting $\delta = \frac{2\sqrt{2/m}}{2^b-1}$, this allows us to represent the values $\{-\sqrt{2/m}, -\sqrt{2/m} + \delta, \ldots, \sqrt{2/m} - \delta, \sqrt{2/m}\}$.

Outline:

- Section 2: Analysis of kernel approximation error.
- Section 3: Analysis of generalization performance.
- Section 4: Training in low-precision.
- Section 5: Experiments.
- Section 6: Future Directions.

Main questions:

- What can we say about the variance of these low-precision features, in their approximation of the kernel? Concentration inequalities?
- Can we analyze the generalization performance of these features?

- Is there an optimal level of precision, with regard to achieving the best kernel approximation performance, or generalization performance, under a fixed memory budget?
- How should models be trained on top of these low-precision features? Can we train in low-precision, and if so, what would the effects of such a training scheme be on the performance of the model?
- Can we achieve better performance per bit, if we use *variable precision* to store the random Fourier features (by storing directions of highest variance with more bits)?

2 Kernel Approximation Error for LP-RFF

2.1 Kernel Approximation Variance Analysis

Definition: For $z \in [a, c]$, let $X_z^{a,c}$ be the random variable which with probability $\frac{z-a}{c-a}$ equals $c - z$, and with probability $\frac{c-z}{c-a}$ equals $a - z$.

Lemma 2.1. $\mathbb{E}[X_z^{a,c}] = 0$, *and* $\mathbb{VAR}[X_z^{a,c}] = (c-z)(z-a) \leq \frac{(c-a)^2}{4}$.

Proof.

$$\begin{aligned}
\mathbb{E}[X_z^{a,c}] &= (c-z)\cdot\frac{z-a}{c-a} + (a-z)\cdot\frac{c-z}{c-a} \\
&= 0. \\
\mathbb{VAR}[X_z^{a,c}] &= (c-z)^2\cdot\frac{z-a}{c-a} + (a-z)^2\cdot\frac{c-z}{c-a} \\
&= \frac{(c-z)(z-a)((c-z+z-a))}{c-a} \\
&= (c-z)(z-a) \\
\frac{d}{dz}[\mathbb{VAR}[X_z^{a,c}]] &= \frac{d}{dz}[-z^2 + (c+a)z - ac] \\
&= -2z + c + a.
\end{aligned}$$

Now, setting the derivative to 0 gives $z^* = \frac{c+a}{2}$. Thus, $\arg\max_{z\in[a,c]}(c-z)(z-a) = \frac{c+a}{2}$, and $\max_{z\in[a,c]}(c-z)(z-a) = (c-\frac{c+a}{2})(\frac{c+a}{2}-a) = \frac{(c-a)^2}{4}$. □

Lemma 2.2. *For $x, y \in \mathcal{X}$, assume we have random variables Z_x, Z_y satisfying $\mathbb{E}[Z_x Z_y] = k(x,y)$, and $\mathbb{VAR}[Z_x] = \mathbb{VAR}[Z_y] = \sigma^2$, and that $k(x,x) = k(y,y) = 1$.*[1] *For any unbiased random quantization function Q with bounded variance $\mathbb{VAR}[Q(z)] \leq \tilde{\sigma}^2$ for any z, it follows that $\mathbb{E}[Q(Z_x)Q(Z_y)] = k(x,y)$, and that $\mathbb{VAR}[Q(Z_x)Q(Z_y)] \leq 2\tilde{\sigma}^2 + \tilde{\sigma}^4 + \sigma^2$.*

[1]For example, one specific instance of the random variables Z_x, Z_y is given by random Fourier features, where $z_x = \sqrt{2}\cos(w^T x + b)$, $z_y = \sqrt{2}\cos(w^T y + b)$, $z_x, z_y \in [-\sqrt{2}, \sqrt{2}]$, for random w, b.

Proof. Let $Q(Z_x) = Z_x + \epsilon_x$, and $Q(Z_y) = Z_y + \epsilon_y$, where $\mathbb{E}[\epsilon_x] = \mathbb{E}[\epsilon_y] = 0$ and $\mathbb{E}\left[\epsilon_x^2\right] \leq \tilde{\sigma}^2$, $\mathbb{E}\left[\epsilon_y^2\right] \leq \tilde{\sigma}^2$.

$$
\begin{aligned}
\mathbb{E}[Q(Z_x)Q(Z_y)] &= \mathbb{E}[(Z_x + \epsilon_x)(Z_y + \epsilon_y)] \\
&= \mathbb{E}[Z_x Z_y + Z_y \epsilon_x + Z_x \epsilon_y + \epsilon_x \epsilon_y] \\
&= \mathbb{E}[Z_x Z_y] \\
&= k(x, y). \\
\mathbb{VAR}[Q(Z_x)Q(Z_y)] &= \mathbb{E}\left[\Big(Q(Z_x)Q(Z_y) - k(x,y)\Big)^2\right] \\
&= \mathbb{E}\left[\Big((Z_x + \epsilon_x)(Z_y + \epsilon_y) - k(x,y)\Big)^2\right] \\
&= \mathbb{E}\left[\Big(Z_y\epsilon_x + Z_x\epsilon_y + \epsilon_x\epsilon_y + Z_xZ_y - k(x,y)\Big)^2\right] \\
&= \mathbb{E}\left[\Big(Z_y\epsilon_x + Z_x\epsilon_y + \epsilon_x\epsilon_y\Big)^2\right] + \mathbb{E}\left[\Big(Z_xZ_y - k(x,y)\Big)^2\right] \\
&= \mathbb{E}\left[Z_y^2\epsilon_x^2 + Z_x^2\epsilon_y^2 + \epsilon_x^2\epsilon_y^2\right] + \sigma^2 \\
&\leq k(y,y)\tilde{\sigma}^2 + k(x,x)\tilde{\sigma}^2 + \tilde{\sigma}^4 + \sigma^2 \\
&= 2\tilde{\sigma}^2 + \tilde{\sigma}^4 + \sigma^2.
\end{aligned}
$$

□

Theorem 2.3. *For $x, y \in \mathcal{X}$, assume we have random variables Z_x, Z_y satisfying $\mathbb{E}[Z_xZ_y] = k(x,y)$, and $\mathbb{VAR}[Z_x] = \mathbb{VAR}[Z_y] = \sigma^2$, and that $k(x,x) = k(y,y) = 1$. Let Q be any unbiased quantization function with bounded variance ($\mathbb{E}[Q(z)] = z$, $\mathbb{VAR}[Q(z)] \leq \tilde{\sigma}^2$ for any z). Let $S = Z_xZ_y$, $T = Q(Z_x)Q(Z_y)$, and $(S_1, \ldots, S_n)$, $(T_1, \ldots, T_n)$ be a random sequence of i.i.d. draws from S and T respectively. Define $\bar{S}_n = \frac{1}{n}\sum_{i=1}^n S_i$, and $\bar{T}_n = \frac{1}{n}\sum_{i=1}^n T_i$, to be the empirical mean of these draws. It follows that $\mathbb{E}\left[\bar{S}_n\right] = \mathbb{E}\left[\bar{T}_n\right] = k(x,y)$, and that $\mathbb{VAR}\left[\bar{S}_n\right] = \frac{\sigma^2}{n}$, and $\mathbb{VAR}\left[\bar{T}_n\right] \leq \frac{2\tilde{\sigma}^2 + \tilde{\sigma}^4 + \sigma^2}{n}$.*

Proof.

$$
\begin{aligned}
\mathbb{VAR}\left[\bar{S}_n\right] &= \mathbb{VAR}\left[\frac{1}{n}\sum_{i=1}^n S_i\right] \\
&= \sum_{i=1}^n \mathbb{VAR}\left[\frac{1}{n}S_i\right] \\
&= n \cdot \frac{\sigma^2}{n^2} \\
&= \frac{\sigma^2}{n}
\end{aligned}
$$

The result for $\mathbb{VAR}\left[\bar{T}_n\right]$ follows in the same way, using the result from Lemma 2.2. □

I will now discuss a concentration bound for these low-precision random features:

Theorem 2.4. *For a fixed $x, y \in \mathcal{X}$, let T be any random variable with the property that $\mathbb{E}[T] = k(x, y)$, and $T \in [-2, 2]$.*[2] *Let $(T_1, \ldots, T_n)$, be a random sequence of i.i.d. draws from T, and let $\bar{T}_n = \frac{1}{n}\sum_{i=1}^{n} T_i$ be the empirical mean of this sequence. Then, for any $\epsilon > 0$,*

$$\mathbb{P}\big[|\bar{T}_n - k(x, y)| \geq \epsilon\big] \leq 2\exp(-n\epsilon^2/8).$$

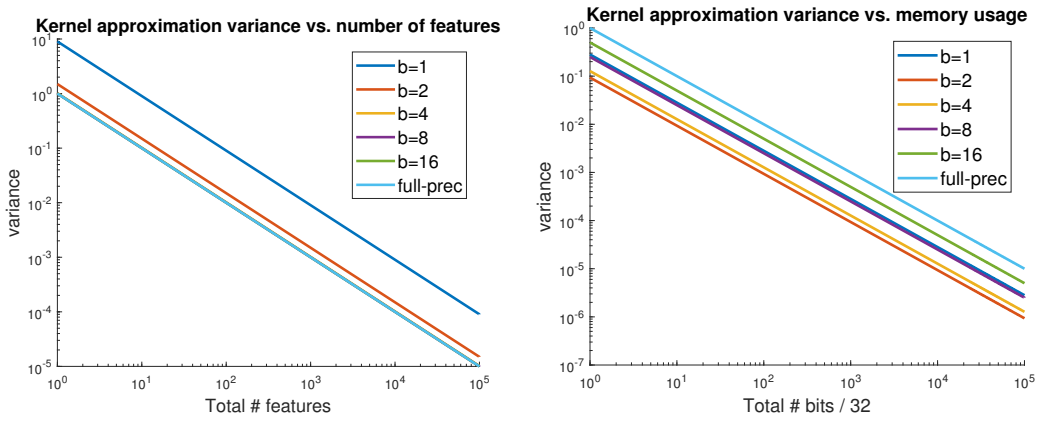
Proof. This follows directly from Hoeffding's inequality. □

Open Question: Can Claim 1 (uniform convergence) from Rahimi and Recht [5] be adapted to this low-precision setting?

2.1.1 Further analysis of variance

In this section, we dig deeper into the bounds proven in Theorem 2.3, to understand whether under a fixed memory budget, we can attain lower kernel approximation error by using LP-RFF. To do so, we will analyze the variance of using $\bar{S}_n$ to approximate $k(x, y)$, relative to $\bar{T}_{n*(32/b)}$. This corresponds to comparing the variance of using n "full precision" features (which we will assume are 32-bit), relative to using $n * (32/b)$ b-bit low-precision features. Note that both of these feature representations use the same total number of bits. We will in this case assume we are using random Fourier features, and thus that $z_x, z_y \in [-\sqrt{2}, \sqrt{2}]$. We will upper bound σ^2 in this context by 1, given the results in [3] for the RBF kernel. Furthermore, it is important to note that the quantization noise $\tilde{\sigma}^2$ is very much tied to the number of bits used to quantize the features z_x. We will thus use $\tilde{\sigma}_b^2$ to denote the variance introduced by quantizing into b-bits. In this case, we divide the interval $[-\sqrt{2}, \sqrt{2}]$ into $2^b - 1$ sub-intervals of equal size, and quantization is performed within each of these intervals; each interval is of size $\frac{2\sqrt{2}}{2^b-1}$. Thus, from Lemma 2.1 the quantization error $\tilde{\sigma}_b^2 \leq \frac{1}{4}\left(\frac{2\sqrt{2}}{2^b-1}\right)^2 = \frac{2}{(2^b-1)^2}$. This allows us to perform the comparison discussed above:

[2] Note that this holds for quantized random Fourier features, where $T = Q(Z_x)Q(Z_y)$, and $Z_x = \sqrt{2}\cos(w^T x + b)$.



(a) (b)

Figure 1: Kernel Approximation Variance, as a function of the number of features used (a), and the amount of memory used (b). Note that in the left plot, the 4-bit, 8-bit, 16-bit, and full-precision lines all overlap.

$$
\begin{aligned}
\mathbb{VAR}\left[\bar{S}_n\right] &= \frac{\sigma^2}{n} \\
&\leq \frac{1}{n}. \\
\mathbb{VAR}\left[\bar{T}_{n*(32/b)}\right] &\leq \frac{2\tilde{\sigma}_b^2 + \tilde{\sigma}_b^4 + \sigma^2}{n*(32/b)} \\
&\leq \frac{2\frac{2}{(2^b-1)^2} + \left(\frac{2}{(2^b-1)^2}\right)^2 + 1}{n*(32/b)} \\
&\leq \frac{\frac{4}{(2^b-1)^2} + \frac{4}{(2^b-1)^4} + 1}{n*(32/b)}
\end{aligned}
$$

In Figure 1, we plot our upper bounds for the "variance per feature" (a), and "variance per bit" (b), on log-log plots, for various values of b and n. In the left plot, we surprisingly see that for 4-bit precision and above, the "variance per feature" closely matches the full precision features. In the right plot, we see that lowering the precision helps reduce the "variance per bit" by approximately a full order of magnitude, with $b = 2$ giving the lowest variance, and full-precision giving the highest variance. These results are validated empirically in Figure 2 of Section 5.

2.2 Spectrum of Low-Precision RFF

In this section, we analyze the effect that using low-precision has on the spectrum of the approximated kernel matrix. We will be quantifying the size of the gap between the low-precision spectrum, and the full-precision spectrum. We will now show that randomly quantizing an RFF

matrix Z with zero-mean noise "elevates" the spectrum of the quantized matrix (relative to the full precision matrix):

Proposition 2.5. *Let Z be an n by d RFF matrix whose i^{th} row is $z(x_i)$, and let C denote the zero mean random quantization noise which is added to Z. If λ denotes the spectrum of ZZ^T, and $\tilde{\lambda}$ denotes the spectrum of $(Z+C)(Z+C)^T$, it follows that $\mathbb{E}\left[\sum_i \lambda_i\right] = n$, while $\mathbb{E}\left[\sum_i \tilde{\lambda}_i\right] \leq n + \frac{2n}{(2^b-1)^2}$.*

Proof.

$$\begin{aligned}
\mathbb{E}\left[\sum_i \lambda_i\right] &= \mathbb{E}\left[trace[ZZ^T]\right] \\
&= \mathbb{E}\left[\sum_i z(x_i)^T z(x_i)\right] \\
&= \sum_i k(x_i, x_i) \\
&= n \\
\mathbb{E}\left[\sum_i \lambda_i\right] &= \mathbb{E}\left[trace((Z+C)(Z+C)^T)\right] \\
&= \mathbb{E}\left[trace(ZZ^T + CZ^T + ZC^T + CC^T)\right] \\
&= \mathbb{E}\left[trace(ZZ^T)\right] + \mathbb{E}\left[trace(CC^T)\right] \\
&= n + \sum_i \mathbb{E}\left[c_i^T c_i\right], \text{ where } c_i \text{ is the } i^{th} \text{ row of } C \\
&= n + \sum_{i=1}^{n}\sum_{j=1}^{d} \mathbb{E}\left[c_{ij}^2\right] \\
&\leq n + nd \cdot \frac{2}{d(2^b-1)^2} \\
&= n + \frac{2n}{(2^b-1)^2}
\end{aligned}$$

Above, we use the fact that $\mathbb{E}\left[c_{ij}^2\right] \leq \frac{1}{4}\left(\frac{2\sqrt{2/d}}{2^b-1}\right)^2 = \frac{2}{d(2^b-1)^2}$. □

Notice that the spectrum $\tilde{\lambda}$ of $(Z+C)(Z+C)^T$ relates to the spectrum $\tilde{\sigma}$ of $(Z+C)$ via $\tilde{\lambda}_i = \tilde{\sigma}_i^2$. Thus, letting σ denote the spectrum of Z, we have shown that $\mathbb{E}\left[\sum_i \tilde{\sigma}_i^2\right] > \sum_i \sigma_i^2$. In other words, adding zero mean noise of any kind to a matrix elevates the ℓ_2 norm of its spectrum. This is intuitive because the noise will always increase the total weight on the diagonal of ZZ^T.

More interestingly, however, we can also show that if we take two random draws C and D of the quantization noise, the spectrum of $(Z+C)(Z+D)^T$ is also elevated relative to ZZ^T, even though the expected value of the entries of the diagonal of this matrix are *equal* to the entries of the ZZ^T diagonal. This is because we can write $(Z+C)(Z+D)^T = ZZ^T + CZ^T + ZD^T + CD^T = ZZ^T + N$, where $N = CZ^T + ZD^T + CD^T$ is zero mean noise. Thus, just like the spectrum of $Z+C$ was higher than that of Z because we added zero mean noise C to it, the spectrum of $(Z+C)(Z+D)^T$ will

be higher than that of ZZ^T because we are adding zero mean noise N to it. We show this in more detail in the proposition below:

Proposition 2.6. *Let Z be an n by d RFF matrix whose i^{th} row is $z(x_i)$, and let C, D denote independent draws of zero mean random quantization noise of the same dimension as Z. If λ denotes the spectrum of ZZ^T, and $\tilde{\lambda}$ denotes the spectrum of $(Z+C)(Z+D)^T$, it follows that $\mathbb{E}\left[\sum_i \lambda_i^2\right] = n + \frac{n^2-n}{d}$, while $\mathbb{E}\left[\sum_i \tilde{\lambda}_i^2\right] \leq n + \frac{n^2-n}{d} + \frac{4n^2}{d}\left(\frac{1}{(2^b-1)^2} + \frac{1}{(2^b-1)^4}\right)$.*

Proof. Letting $A = ZZ^T$ and $\tilde{A} = (Z+C)(Z+D)^T$, we derive the following:

$$\begin{aligned}
\mathbb{E}\left[\sum_i \lambda_i^2\right] &= \mathbb{E}\left[trace(A^T A)\right] \\
&= \mathbb{E}\left[trace(ZZ^T ZZ^T)\right]. \\
\mathbb{E}\left[\sum_i \tilde{\lambda}_i^2\right] &= \mathbb{E}\left[trace(\tilde{A}^T \tilde{A})\right] \\
&= \mathbb{E}\left[trace((Z+D)(Z+C)^T(Z+C)(Z+D)^T)\right] \\
&= \mathbb{E}\left[trace(ZZ^T ZZ^T + ZC^T CZ^T + DZ^T ZD^T + DC^T CD^T + \text{zero-mean})\right] \\
&= \mathbb{E}\left[trace(ZZ^T ZZ^T)\right] + \mathbb{E}\left[trace(ZC^T CZ^T)\right] + \mathbb{E}\left[trace(DZ^T ZD^T)\right] + \mathbb{E}\left[trace(DC^T CD^T)\right] \\
&= \mathbb{E}\left[trace(ZZ^T ZZ^T)\right] + \mathbb{E}\left[trace(ZC^T CZ^T)\right] + \mathbb{E}\left[trace(ZD^T DZ^T)\right] + \mathbb{E}\left[trace(DC^T CD^T)\right] \\
&= \mathbb{E}\left[trace(ZZ^T ZZ^T)\right] + 2\mathbb{E}\left[trace(ZC^T CZ^T)\right] + \mathbb{E}\left[trace(DC^T CD^T)\right].
\end{aligned}$$

Now, we need to compute more precise values for the three terms in this expression. For this purpose, we will repeatedly use the following fact, which applies to any matrices A, B of equal dimensions:

$$trace[AB^T BA^T] = \sum_{i,j=1}^{n} \sum_{k,\hat{k}=1}^{d} a_{jk} a_{j\hat{k}} b_{ik} b_{i\hat{k}}.$$

We will proceed one expression at a time:

- Case 1: $A = B = Z$.

$$\begin{aligned}
\mathbb{E}\left[trace[ZZ^T ZZ^T]\right] &= \sum_{i,j=1}^{n} \sum_{k,\hat{k}=1}^{d} \mathbb{E}\left[z_{jk} z_{j\hat{k}} z_{ik} z_{i\hat{k}}\right] \\
&= \sum_{i=1}^{n} \sum_{k,\hat{k}=1}^{d} \mathbb{E}\left[z_{ik}^2\right] \mathbb{E}\left[z_{i\hat{k}}^2\right] + \sum_{i \neq j} \sum_{k=1}^{d} \mathbb{E}\left[z_{jk}^2\right] \mathbb{E}\left[z_{ik}^2\right] \\
&= \sum_{i=1}^{n} \sum_{k,\hat{k}=1}^{d} \frac{1}{d^2} + \sum_{i \neq j} \sum_{k=1}^{d} \frac{1}{d^2} \\
&= nd^2 \frac{1}{d^2} + (n^2 - n) d \frac{1}{d^2} \\
&= n + \frac{n^2 - n}{d}.
\end{aligned}$$

Above, we used the fact that $\mathbb{E}\left[z_{ik}^2\right] = \mathbb{E}_{w_k,b_k}\left[\frac{2}{d}\cos(w_k^T x_i + b_k)^2\right] = \frac{k(x_i,x_i)}{d} = \frac{1}{d}$.

- Case 2: $A = Z$, $B = C$.

$$\begin{aligned}
\mathbb{E}\left[trace[ZC^TCZ^T]\right] &= \sum_{i,j=1}^{n}\sum_{k,\hat{k}=1}^{d}\mathbb{E}\left[z_{jk}z_{j\hat{k}}c_{ik}c_{i\hat{k}}\right] \\
&= \sum_{i,j=1}^{n}\sum_{k=1}^{d}\mathbb{E}\left[z_{jk}^2\right]\mathbb{E}\left[c_{ik}^2\right] \\
&\leq \sum_{i,j=1}^{n}\sum_{k=1}^{d}\frac{1}{d}\cdot\frac{2}{d(2^b-1)^2} \\
&= n^2 d\cdot\frac{1}{d}\cdot\frac{2}{d(2^b-1)^2} \\
&= \frac{2n^2}{d(2^b-1)^2}.
\end{aligned}$$

Above, we used the fact that $\mathbb{E}\left[c_{ik}^2\right] \leq \frac{1}{4}\left(\frac{2\sqrt{2/d}}{2^b-1}\right)^2 = \frac{2}{d(2^b-1)^2}$, and that $\mathbb{E}\left[z_{ik}^2\right] = \frac{1}{d}$.

- Case 3: $A = C$, $B = D$.

$$\begin{aligned}
\mathbb{E}\left[trace[CD^TDC^T]\right] &= \sum_{i,j=1}^{n}\sum_{k,\hat{k}=1}^{d}\mathbb{E}\left[c_{jk}c_{j\hat{k}}d_{ik}d_{i\hat{k}}\right] \\
&= \sum_{i,j=1}^{n}\sum_{k=1}^{d}\mathbb{E}\left[c_{jk}^2\right]\mathbb{E}\left[d_{ik}^2\right] \\
&\leq \sum_{i,j=1}^{n}\sum_{k=1}^{d}\left(\frac{2}{d(2^b-1)^2}\right)^2 \\
&= n^2 d\cdot\frac{4}{d^2(2^b-1)^4} \\
&= \frac{4n^2}{d(2^b-1)^4}.
\end{aligned}$$

Combining these three results concludes the proof of the proposition. □

In Figures 3 and 4 in Section 5 we validate these results empirically. We observe that the spectra of our kernel approximation matrices are much higher than the actual spectrum of the exact kernel matrix, in the case where we use 1, 2, or 4 bits to quantize our features. We also observe that using $(Z+C)(Z+D)^T$ instead of $(Z+C)(Z+C)^T$ gives a smaller increase to the spectrum.

3 Generalization Performance of Low-Precision RFF

Theorem 3.1. *(extending Theorem 1 of [4] to LP-RFF):*

Let $\phi : \mathcal{X} \times \Omega \to \mathbb{R}$ be a set of feature functions such that $\sup_{x,w} |\phi(x;w)| \leq 1$. Assume there exists a family of deterministic quantization functions $\{Q_\theta : \mathbb{R} \to \mathbb{R} \mid \theta \in \Theta\}$, parameterized by some parameter $\theta \in \Theta$. Let p_Θ be a probability distribution over Θ, p_Ω be a probability distribution over Ω, and define $p(w,\theta) = p_\Omega(w)p_\Theta(\theta)$. Define the quantized feature functions as $\phi'(x;w,\theta) = Q_\theta(\phi(x;w))$, and define

$$\mathcal{F}_p = \left\{ f(x) = \int_{\Omega \times \Theta} \alpha(w,\theta)\phi'(x;w,\theta)dwd\theta \;\middle|\; |\alpha(w,\theta)| \leq Cp(w,\theta) \right\}.$$

Suppose the cost function $c(y,y') = c(yy')$, with $c(yy')$ L-Lipschitz. Then for any $\delta > 0$, if the training data $\{x_i, y_i\}_{i=1}^n$ are drawn iid from some distribution P, Algorithm 1 returns a function $\hat{f}$ that satisfies:

$$R[\hat{f}] - \min_{f \in \mathcal{F}_p} R[f] \leq O\left(\left(\frac{1}{\sqrt{n}} + \frac{1}{\sqrt{m}} \right) LC\sqrt{\log \frac{1}{\delta}} \right)$$

with probability at least $1 - 2\delta$ over the training dataset and the choice of the parameters w_i, θ_i. (Here, $R[F] = \mathbb{E}_{(x,y)\sim P}[c(f(x),y)]$)

Algorithm 1 Low-Precision Weighted Sum of Random Kitchen Sinks Training (adapted from [4])

Input A dataset $\{x_i, y_i\}_{i=1}^n$ of n points, a family of bounded feature functions $|\phi(x;w)| \leq 1$ parameterized by $w \in \Omega$, a family of bounded quantization functions $|Q(z;\theta)| \leq 1$ parameterized by $\theta \in \Theta$, an integer m, a scalar C, probability distributions p_Ω and p_Θ over Ω and Θ.

Output A function $\hat{f}(x) = \sum_{i=1}^m Q(\phi(x;w_i);\theta_i)\alpha_i^*$.

1: Draw $w_1, \ldots, w_m$ from p_Ω, and $\theta_1, \ldots, \theta_m$ from p_Θ.

2: Featurize the input: $z_i \leftarrow [Q(\phi(x_i;w_1);\theta_1), \ldots, Q(\phi(x_i;w_m);\theta_m)]^T$.

3: With the w_i and θ_i fixed, solve the empirical risk minimization problem:

$$\alpha* = \arg\min_{\alpha \in \mathbb{R}^m} \frac{1}{n} \sum_{i=1}^n c(\alpha^T z_i, y_i)$$
$$s.t. \|\alpha\|_\infty < C/m$$

4: **return** $\hat{f}(x) = \sum_{i=1}^m Q(\phi(x;w_i);\theta_i)\alpha_i^*$.

Open Question: How does the family of functions $\mathcal{F}_p$ defined in Proposition 3 compare to the one in [4] (Rahimi and Recht, 2008)?

3.1 "Deterministic" Quantization Functions

In Theorem 3.1, we defined a set of deterministic quantization functions $\{Q_\theta : \mathbb{R} \to \mathbb{R} \mid \theta \in \Theta\}$, parameterized by some parameter $\theta \in \Theta$. We now discuss how to construct these. The idea is

simple: For every quantization interval $[a, c]$, we choose a threshold value $t \in [a, c]$ uniformly at random. Then if a certain unquantized feature z falls in this interval, we will set $Q_t(z) = a$ if $z \leq t$, and $Q_t(z) = c$ if $z > t$. It is easy to show that $\mathbb{E}_t[Q_t(z)] = z$:

$$\begin{aligned}
\mathbb{E}_t[Q_t(z)] &= a \cdot \mathbb{P}[z \leq t] + c \cdot \mathbb{P}[z > t] \\
&= a \cdot \frac{c - z}{c - a} + c \cdot \frac{z - a}{c - a} \\
&= \frac{ac - az + cz - ac}{c - a} \\
&= z.
\end{aligned}$$

Thus, if for every quantization interval we draw a random threshold t_i as described, and concatenate these thresholds as $\theta = (t_1, \ldots, t_{2^b - 1})$, we have successfully constructed a set of unbiased and deterministic quantization functions, as desired.

3.2 Effect of quantization noise at test time

In this section, we show that the quantization noise at test time results in larger expected mean-squared error, in the context of kernel ridge regression.

Theorem 3.2. *Let $Q : \mathbb{R}^m \to \mathbb{R}^m$ be an unbiased (element-wise) random quantization function with bounded variance ($\mathbb{VAR}[Q(z)_i] \leq \tilde{\sigma}^2 \ \forall z, i$), and let $f_w(z) = w^T z$ be a fixed regression model. It follows that the expected mean-squared error, using quantized features $Q(z(x))$, satisfies*

$$\mathbb{E}\left[\Big(f_w(Q(z(x))) - y\Big)^2\right] \leq \mathbb{E}\left[\Big(f_w(z(x)) - y\Big)^2\right] + \|w\|^2 \tilde{\sigma}^2.$$

Proof. Let $z(x)$ denote the full precision RFF representation for x, and $Q(z(x)) = z(x) + \epsilon_x$ represents the randomly quantized representation.

$$\begin{aligned}
\mathbb{E}\left[\Big(f_w(Q(z(x))) - y\Big)^2\right] &= \mathbb{E}\left[(w^T(z(x) + \epsilon_x) - y)^2\right] \\
&= \mathbb{E}\left[(w^T z(x) + w^T \epsilon_x - y)^2\right] \\
&= \mathbb{E}\left[(w^T z(x) - y)^2)\right] + \mathbb{E}\left[(w^T \epsilon_x)^2\right] \\
&= \mathbb{E}\left[\Big(f_w(z(x)) - y\Big)^2\right] + \mathbb{E}\left[\Big(\sum_i w_i \epsilon_{x,i}\Big)^2\right] \\
&= \mathbb{E}\left[\Big(f_w(z(x)) - y\Big)^2\right] + \sum_i w_i^2 \mathbb{E}\left[\epsilon_{x,i}^2\right] \\
&= \mathbb{E}\left[\Big(f_w(z(x)) - y\Big)^2\right] + \|w\|^2 \mathbb{E}\left[\epsilon_{x,1}^2\right] \\
&\leq \mathbb{E}\left[\Big(f_w(z(x)) - y\Big)^2\right] + \|w\|^2 \tilde{\sigma}^2.
\end{aligned}$$

□

This theorem shows that our expected error on a test point (x, y), if we use quantized features at test time, is in expectation $\|w\|^2\tilde{\sigma}^2$ larger than the expected error if we used the full precision features at test time. In the case where Q quantizes each feature into b bits, $\tilde{\sigma}^2 = \frac{2}{m(2^b-1)^2}$. This suggests that it could be desirable, in certain contexts, to train using low precision (to speed up training), but use full precision features at test time.

3.3 Fixed Design Kernel Ridge Regression

In this section, we will show, in the context of "fixed design" linear regression, that the generalization performance of the model depends deeply on the spectrum of the kernel matrix (or kernel approximation matrix) used to train the model.

"Fixed design" linear regression is a particularly straightforward to analyze form of linear regression, in which the test points are the same as the training points; importantly, however, the training labels y_i, and test labels y_i', are noisy, corresponding to zero-mean noise added to the unobserved "true labels" y_i^*. The noise values added to the labels are assumed to be uncorrelated with common variance $\sigma^2 > 0$. In the case of kernel ridge regression, the learned classifier predicts regression values of $\hat{y} = K(K + \lambda\mathbb{1})^{-1}y = Ay$ on the training set, for $A = K(K + \lambda\mathbb{1})^{-1}$. Letting $K = U\Sigma U^T$, where σ_i is the i^{th} largest eigenvalue of K, and U_i is the corresponding eigenvector, we have that $A = U\Gamma U^T$, with $\gamma_i = \frac{\sigma_i}{\sigma_i+\lambda}$.

In his course notes for Advanced Methods for Data Analysis [6], Ryan Tibshirani shows that the expected test error can be precisely calculated as a function of the training error, the variance σ^2 of the label noise, and the "degrees of freedom" of the classifier (which corresponds to the trace of A, when the classifier predicts $\hat{y} = Ay$ on the training set). The precise relation is as follows:

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i')^2\right] \quad = \quad \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2\right] + \frac{2\sigma^2}{n}df(\hat{y})$$

We now show that in the context of kernel ridge regression, the right hand side of this expression can be simplified to only depend on the eigenvalues and eigenvectors of K.

Proposition 3.3. *Letting $A = K(K + \lambda\mathbb{1})^{-1}$, where $K = U\Sigma U^T$ is a kernel matrix with eigenvectors U_i and eigenvalues σ_i, it holds that the expected test error in the context of fixed design linear regression can be expressed as follows:*

$$\mathbb{E}_{\epsilon,\epsilon'}\left[\frac{1}{n}\|A(y^* + \epsilon) - (y^* + \epsilon')\|^2\right] \quad = \quad \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\sigma_i}{\sigma_i + \lambda} - 1\right)^2 (U_i^T y^*)^2 + \frac{\sigma^2}{n}\sum_{i=1}^{n}\left(\frac{\sigma_i}{\sigma_i + \lambda}\right)^2 + \sigma^2,$$

Proof. We simplify the expression from Ryan Tibshirani's course notes, using the facts that $\hat{y} = A(y^* + \epsilon)$, $y' = y + \epsilon'$, $A = U\Gamma U^T$ with $\gamma_i = \frac{\sigma_i}{\sigma_i + \lambda}$, and $df(\hat{y}) = trace(A) = \sum_i \gamma_i = \sum_i \frac{\sigma_i}{\sigma_i + \lambda}$.

$$
\begin{aligned}
\mathbb{E}_{\epsilon,\epsilon'}\left[\frac{1}{n}\|A(y^*+\epsilon) - (y^*+\epsilon')\|^2\right] &= \mathbb{E}_{\epsilon}\left[\frac{1}{n}\|A(y^*+\epsilon) - (y^*+\epsilon)\|^2\right] + \frac{2\sigma^2}{n} df(\hat{y}) \\
&= \frac{1}{n}\mathbb{E}_{\epsilon}\left[\|(A - \mathbb{1})(y^*+\epsilon)\|^2\right] + \frac{2\sigma^2}{n} Tr(A) \\
&= \frac{1}{n}\mathbb{E}_{\epsilon}\left[\|U(\Gamma - \mathbb{1})U^T(y^*+\epsilon)\|^2\right] + \frac{2\sigma^2}{n} Tr(A) \\
&= \frac{1}{n}\mathbb{E}_{\epsilon}\left[(y^*+\epsilon)^T U(\Gamma - \mathbb{1})U^T U(\Gamma - \mathbb{1})U^T(y^*+\epsilon)\right] + \frac{2\sigma^2}{n} Tr(A) \\
&= \frac{1}{n}\mathbb{E}_{\epsilon}\left[Tr\left((y^*+\epsilon)^T U(\Gamma - \mathbb{1})^2 U^T(y^*+\epsilon)\right)\right] + \frac{2\sigma^2}{n} Tr(A) \\
&= \frac{1}{n}\mathbb{E}_{\epsilon}\left[Tr\left(U(\Gamma - \mathbb{1})^2 U^T(y^*+\epsilon)(y^*+\epsilon)^T\right)\right] + \frac{2\sigma^2}{n} Tr(A) \\
&= \frac{1}{n}\mathbb{E}_{\epsilon}\left[Tr\left(U(\Gamma - \mathbb{1})^2 U^T(y^*y^{*T} + \epsilon y^{*T} + y^*\epsilon^T + \epsilon\epsilon^T\right)\right] + \frac{2\sigma^2}{n} Tr(A) \\
&= \frac{1}{n} Tr\left(U(\Gamma - \mathbb{1})^2 U^T\left(y^*y^{*T} + \mathbb{E}_{\epsilon}\left[\epsilon\epsilon^T\right]\right)\right) + \frac{2\sigma^2}{n} Tr(A) \\
&= \frac{1}{n} Tr\left(U(\Gamma - \mathbb{1})^2 U^T\left(y^*y^{*T} + \sigma^2\mathbb{1}\right)\right) + \frac{2\sigma^2}{n} Tr(A) \\
&= \frac{1}{n} Tr\left((\Gamma - \mathbb{1})^2 U^T y^*(U^T y^*)^T\right) + \frac{\sigma^2}{n} Tr\left((\Gamma - \mathbb{1})^2\right) + \frac{2\sigma^2}{n} Tr(U\Gamma U^T) \\
&= \frac{1}{n}\sum_{i=1}^{n}(\gamma_i - 1)^2 (U_i^T y^*)^2 + \frac{\sigma^2}{n}\sum_{i=1}^{n}(\gamma_i - 1)^2 + \frac{2\sigma^2}{n}\sum_{i=1}^{n}\gamma_i \\
&= \frac{1}{n}\sum_{i=1}^{n}(\gamma_i - 1)^2 (U_i^T y^*)^2 + \frac{\sigma^2}{n}\sum_{i=1}^{n}\left(\gamma_i^2 - 2\gamma_i + 1 + 2\gamma_i\right) \\
&= \frac{1}{n}\sum_{i=1}^{n}(\gamma_i - 1)^2 (U_i^T y^*)^2 + \frac{\sigma^2}{n}\sum_{i=1}^{n}\gamma_i^2 + \sigma^2
\end{aligned}
$$

□

This result shows the fundamental relationship between the spectrum of the kernel matrix, and the generalization performance of the regression model.

3.4 When is it safe to use low precision?

Basic idea: If the optimal choice of λ is much larger than the average amount of noise added to each element of the diagonal when using b bits of precision, then it is safe to use b bits. In particular, if we assume that $c = \frac{2n}{d(2^b-1)^2}$ (see Proposition 2.5) is added to each of the first d eigenvalues of the kernel approximation matrix when using b bits, we get the following bound on the loss:

$$L(\tilde{\gamma}) \leq L(\gamma) + \frac{\|y\|^2}{n}\left(\frac{c}{c+\lambda}\right)^2 + 3\sigma^2\frac{c}{c+\lambda}.$$

Thus, if $\lambda >> c$, then the expected generalization performance of the model trained on full-precision RFFs will be similar to one trained on b bit RFFs. If this is true at λ^* (best regularizer for full-precision features), then we know that the best performing full precision model won't be much better than the low-precision model using the same regularization constant.

Thus, we turn our attention to understanding in what cases $\lambda^* >> c = \frac{2n}{d(2^b-1)^2}$. This boils down to studying factors result in high values of λ^*, and which result in low-values.

We are interested in the value of λ that minimizes the expression:

$$L(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\lambda}{\sigma_i+\lambda}\right)^2(U_i^T y^*)^2 + \frac{\sigma^2}{n}\sum_{i=1}^{n}\left(\frac{\sigma_i}{\sigma_i+\lambda}\right)^2.$$

We consider several simplified scenarios:

1. **Case 1**: High noise regime ($\sigma^2 >> \|y^*\|^2$). In this case, the second term is the only one that matters, and thus $\lambda \to \infty$ is optimal.

2. **Case 2**: Low noise regime ($\sigma^2 >> \|y^*\|^2$). In this case, the first term is the only one that matters, and $\lambda \to 0$ is optimal.

3. **Case 3**: $\sigma_1 \neq 0, \sigma_j = 0 \ \forall j \neq 1$. In this case, $\lambda^* = \frac{\sigma_1\sigma^2}{(U_1^T y^*)^2}$. Thus, the "signal to noise ratio" $\frac{\sigma^2}{(U_1^T y^*)^2}$ is central to determining the optimal λ.

4. **Case 4**: $(U_i^T y^*)^2 = \frac{1}{n}\|y^*\|^2 \ \ \forall i$. In this case, $\gamma_i^* = \frac{(U_i^T y^*)^2}{(U_i^T y^*)^2+\sigma^2} = \frac{\|y^*\|^2/n}{\|y^*\|^2/n+\sigma^2} = \frac{\sigma_i}{\sigma_i+\lambda^*}$. So if all the σ_i are equal, this says that $\lambda^* \propto \sigma^2$.

5. **Case 5**: Flat spectrum ($\sigma_i = c \ \forall i$). In this case, $\lambda^* = \frac{n\sigma^2 c}{\|y\|^2}$. We derive this because the gradient of the loss with respect to λ is:

$$\frac{dL}{d\lambda} = 2\sum_{i=1}^{n}\frac{(U_i^T y^*)^2\sigma_i\lambda - \sigma^2\sigma_i}{(\sigma_i+\lambda)^3}.$$

   Letting $\sigma_i = c$, setting this derivate to 0, and solving, gives the desired result.

4 Training a model using low-precision random features

In this section, we present several possible ways of training a model on top of low-precision random Fourier features. Importantly, any of the options outlined below which uses low-precision for both train/test would be inherently limited in terms of how close to the global optimum they could get,

as discussed in the recent High-Accuracy Low-Precision Training paper [7]. Perhaps, however, the error introduced by this quantization could be more than made-up for by the ability to learn a model in a higher-dimensional, more expressive, feature space.

- **LM-HALM**: 'LM-HALP" is the version of HALP designed for learning linear models, presented as Algorithm 4 in [7]. One potential down-side to this approach is that it would mean that the learned model would be full-precision, which may or may not be desirable (for example, computing the full-precision gradient over the entire dataset could be prohibitively expensive, and perhaps even impossible in the case where the full-precision model wouldn't fit in memory).

- **Perceptron updates**: We could use the perceptron algorithm for model updates, which would ensure that all operations could be done in integer arithmetic.

- **Quantized SGD updates ("LP-SGD")**: Consider the full-precision SGD update of the form $w_{t+1} = w_t + g_t z_t$, where $z_t = z(x_t) \in \mathbb{R}^m$ is the random feature representation corresponding to the randomly chosen training point x_t at time t. We could replace this update by updates of the form $w_{t+1} = w_t + \tilde{g}_t \tilde{z}_t$, where $\mathbb{E}\left[\tilde{g}_t\right] = g_t$, and $\tilde{g}_t$ is stored in low-precision format. For example, in the case of logistic regression, $g_t = \eta\big(y_t - p_t\big)$, where $y_t \in \{0, 1\}$ is the label for x_t, $\eta \in \mathbb{R}$ is the learning rate, and $p_t = p(Y_t = 1 | z_t, w_t) = (1 + \exp(-w_t^T z_t))^{-1}$. If $\tilde{g}_t$ is in (δ, b) low-precision format, and $\tilde{z}_t$ is in (δ', b') format, then w_t would be in $(\delta\delta', b + b')$ format.

- **LP-SVRG**: We could directly use LP-SVRG.

- **Update dual variables instead of primal**: In terms of the dual variables α_i, the model becomes $f(x) = \sum_{i=1}^n \alpha_i z(x_i)^T z(x)$. Letting $Z \in \mathbb{R}^{n \times m}$ be the matrix whose i^{th} row is $z(x_i)$, this can be rewritten as $\alpha^T (Z z(x))$. If the random features are binary, $Z z(x)$ can be implemented very efficiently, and its output is an integer vector (note that for huge Z, this operation can be distributed across a cluster of machines). If α is stored in low-precision format (as it would be if updates of the form described above for perceptron or LP-SGD are used), this dot-product could be performed using low-precision integer arithmetic, which can also be implemented fast. As far as updating the values of the dual parameters α_i during training, we can simply simulate the primal updates of the form $w_{t+1} = w_t + \tilde{g}_t z_t$ by using $\alpha_t = \alpha_t + \tilde{g}_t$, where here I am using α_t to denote the dual parameter corresponding to the random point x_t chosen at time t.

- **Distributed optimization**: We could also consider large-scale distribution optimization algorithms (e.g., [1, 2]) in order to speed up training.

4.1 Computing the low-precision random features

In this entire document so far, we have considered the setting where we compute quantized features by first computing the full precision features, and then performing random quantization. Another option is to first quantize x, W, and b, as $\tilde{x}$, $\tilde{W}$, $\tilde{b}$. Then, compute $\cos(\tilde{W}^T\tilde{x} + \tilde{b})$, and quantize the output. One important thing to note is that quantizing x,W, and b in such a way that $\mathbb{E}_{\tilde{x},\tilde{W},\tilde{b}}\left[\tilde{W}^T\tilde{x} + \tilde{b}\right] = W^Tx + b$ does *not* mean that $\mathbb{E}_{\tilde{x},\tilde{W},\tilde{b}}\left[\cos(\tilde{W}^T\tilde{x} + \tilde{b})\right] = \cos(W^Tx + b)$. Thus, features generated in this way would not necessarily produce unbiased estimates of the kernel function, which satisfy $\mathbb{E}\left[z(x)^Tz(y)\right] = k(x, y)$.

Open Question: Is there anything we can prove about computing low-precision features in the above manner? Would they perform well empirically?

5 Experiments

5.1 Kernel Ridge Regression

We use UCI Census dataset for the experiments on kernel ridge regression. This dataset contains 16k training samples with 119 original features. Besides from reporting the kernel approximation errors, we also report the validation L2 loss in the regression problem. The reported L2 loss for each feature precision is from a grid search over $\{1e^{-6}, 1e^{-5}, ..., 1e^{-1}, 1, 10, 100\}$ for L2 regularizer strength λ. To report statistically meaningful results, we average the performance metrics from 5 random seeds for random Fourier feature generation and stochastic quantization. We uniformly use Gaussian kernels with $\sigma = 30.0$ for our experiments on kernel ridge regression.

5.1.1 Kernel approximation error and validation L2 loss

In this section, we evaluate low precision random Fourier features in comparison to full precision features. We report the kernel approximation error and regression performance (the validation L2 loss on the heldout dataset) when using (1) the same number of random Fourier features and (2) using the same memory budget for random Fourier features. As shown in Figure 2 (a) and (c), when using the same number of random Fourier features, we observe the kernel approximation error only degrades when using 1 or 2 bits; the low precision representation in other precision level is on par with the full precision features. Regarding the regression performance, i.e. the validation l2 loss, we only observe degradation for 1,2 and 4 bits representation. Furthermore, in 2 (b) and (d), we compare the performance of feature representation in different precision *under the same memory bits budget for each data sample*. In this setting, we observe 2 or 4 bit features presents the best kernel approximation error, while 8 bits representation gives the best validation L2 loss.

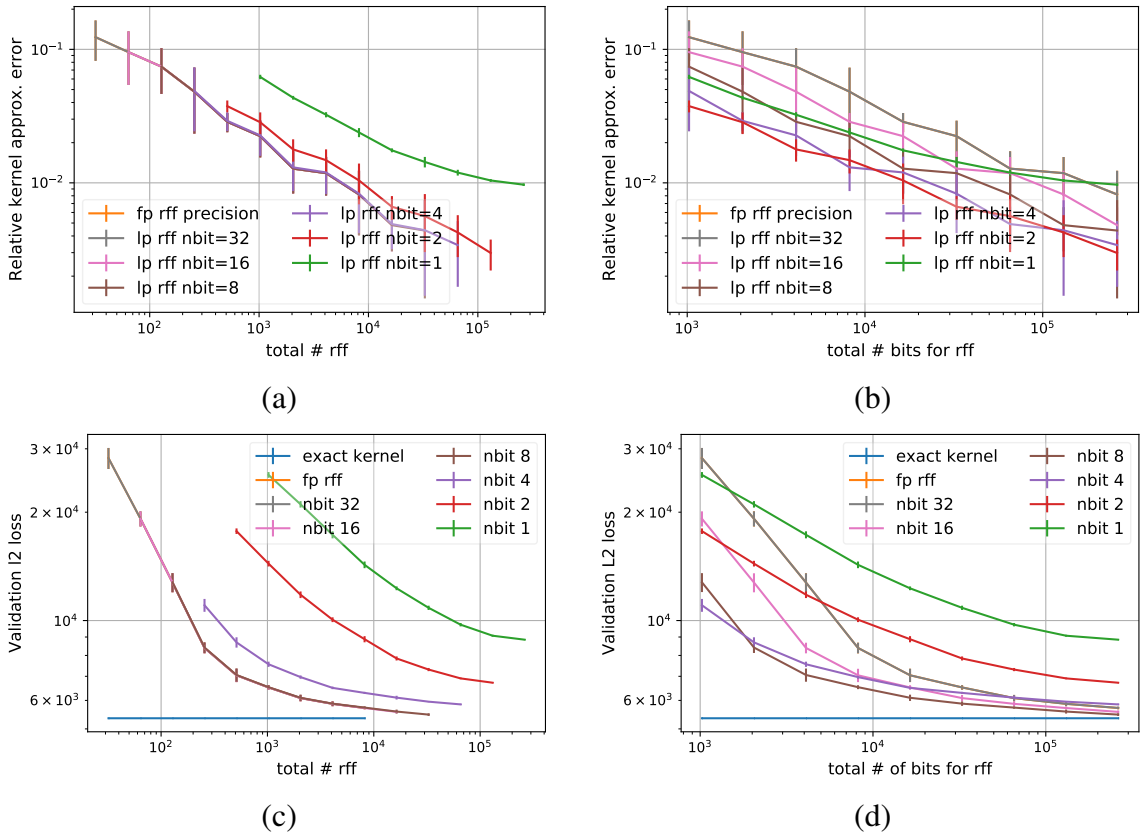
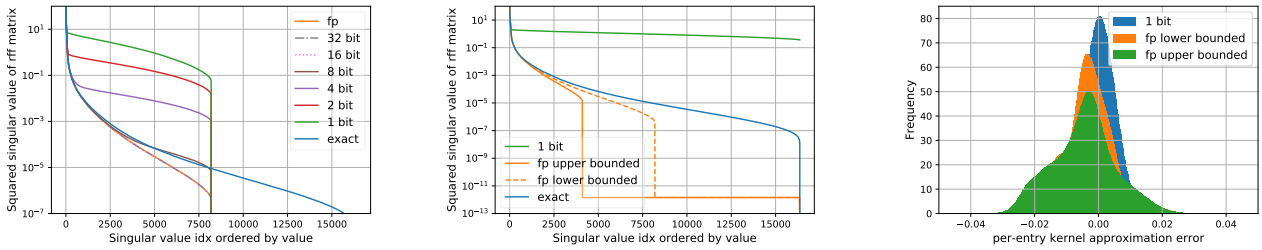


Figure 2: Kernel approximation and validation L2 loss for Kernel Ridge Regression on UCI Census dataset. (a) and (c) compare different precision representation with the same number of random Fourier features. (b) and (d) compare different precision representation under same memory bits budgets.



(a) (b) (c)

Figure 3: Spectrum (eigen values) of the kernel matrix. (a) The spectrums from different precision representation is shown under 8192 random Fourier features. (b) Configurations with similar approximation errors can demonstrate very different spectrum; we compare the 1 bit representation to the full precision configurations giving 1) the highest approx. error that is lower than the 1 bit configuration 2) the lowest approx. error that is higher than the 1bit configuration. (c) The histogram of per-entry kernel approximation error; the histogram is aggregated from 10 runs with different random seed.

	Relative kernel approx. error	Regression performance
exact kernel	0	2.86×10^7
1 bit RFF	1.04×10^{-2}	8.23×10^7
fp RFF upper bounded	1.24×10^{-2}	3.42×10^7
fp RFF lower bounded	8.70×10^{-3}	3.17×10^7

Table 1: The kernel approximation error and regression performance (validation L2 loss) from the 4 models in Figure 3 (b). Similar kernel approximation errors do not always imply similar regression performance.

5.1.2 Spectrum of kernel matrix correlates with L2 loss

In our theory in Section 2.2, we have been discussing how low precision introduce quantization variance, which results in bumping up the spectrum of kernel matrix. To empirically support our analysis on the spectrums, in Figure 3 (a), we demonstrate the kernel matrix spectrum from 8192 random Fourier features in different precision. More specifically, we approximate the kernel matrix with $\boldsymbol{K}' = \tilde{\boldsymbol{Z}}\tilde{\boldsymbol{Z}}^T$ where $\tilde{\boldsymbol{Z}} = Q(\boldsymbol{Z})$ is the quantized version of random Fourier feature $\boldsymbol{Z}$. This quantization approach corresponds to the case where $\boldsymbol{K}' = (\boldsymbol{Z} + \boldsymbol{C})(\boldsymbol{Z} + \boldsymbol{C})^T$ in Section 2.2; it imposes the same noise on the feature matrix $\boldsymbol{Z}$ and its transpose. We can observe 8, 16 and 32 bits representation has spectrums close to the spectrum of full precision RFF kernel matrix. However, for 4, 2 and 1 bits representation has significantly higher spectrums than 8, 16 and 32 bits representation. Remember that in Figure 2, we have seen 8, 16 and 32 bits representation demonstrates similar L2 loss, while 4, 2 and 1 bits can show observably worse performance. *From these observations, we believe spectrum is closely correlated with the validation L2 loss. Indeed if the two RFF kernel matrices has the same spectrum, the RFF features matrices are only different up to*

rotation and flipping; the spectrum are at the heart of the generalization performance for the kernel ridge regression model. We refer to Section 3.3 for more detailed discussion on generalization performance and spectrum, in the context of "fixed design" kernel ridge regression.

Additionally, we compare to the correlation between kernel approximation error and L2 loss, to further emphasize the strong correlation between spectrum and L2 loss. In Figure 3 (b), we demonstrates the spectrums from 3 configurations with very similar kernel approximation error. Specifically, we compare 1 bit RFF with 131k features to two full precision RFF configuration with the closest kernel approximation error; these 2 full precision configurations respectively gives 1) the highest approx. error that is lower than the 1 bit configuration (8192 features) 2) the lowest approx. error that is higher than the 1bit configuration (4096 features). As shown in Table 1, the 3 configurations shows close relative kernel approximation error around $1e^{-2}$, but can demonstrate very different spectrum in Figure 3 and very different validation L2 loss in Figure 2 (d).

5.1.3 Effect of independent quantization of RFF matrix and its transpose

In Section 5.1.2, we have validated the analysis in Section 2.2, empirically showing low precision RFF bumps up the kernel matrix spectrum. The discussion in Section 2.2 also suggested a technique to reduce the height of the bump. Specifically, we can approximate the kernel matrix $\boldsymbol{K}' = \tilde{\boldsymbol{Z}}_1 \tilde{\boldsymbol{Z}}_2^T$ where $\tilde{\boldsymbol{Z}}_1$ and $\tilde{\boldsymbol{Z}}_1$ are two independent instantiations of stochastic quantization (i.e. quantization noise matrices $\boldsymbol{C} \neq \boldsymbol{D}$ in Section 2.2). We compare this quantization approach to the basic case in Section 5.1.2 where the low precision kernel approximation $\boldsymbol{K}' = \tilde{\boldsymbol{Z}}\tilde{\boldsymbol{Z}}^T$ uses a single instantiation of stochastic quantization. In Figure 4, we show the comparison under two different memory bits budgets for the feature of each sample. In both the budgets, the spectrum becomes significantly lower after switching to independent quantization for RFF matrix and its transpose. The also observe that, when using two independent instantiations of stochastic quantization, the spectrum from 8192 features is significantly lower than the one from 1024 features. This aligns with the implication from the theory in Section 2.2: the more features we have in the low precision representation, the lower the spectrum will be.

5.1.4 Effect of reducing variance in test sample features

In Section 3.2, we discuss that reducing noise on test sample features can improve regression performance, i.e. the validation L2 loss in kernel ridge regression. To preliminarily validate this effect, we first train the kernel ridge regression model with low precision RFF. In the test phase, we evaluate the validation L2 loss using prediction $\hat{y} = \boldsymbol{w}^T \boldsymbol{z} + b$, where we use *full precision* test sample RFF instead of its quantized low precision version; we call this approach as variance reduction technique. We compare this variance reduction technique to the basic case where the

model predicts with $\tilde{\hat{y}} = \boldsymbol{w}^T \tilde{\boldsymbol{z}} + b$ using quantized low precision test sample RFF $\tilde{\boldsymbol{z}}$. In Figure 5, by using the variance reduction technique, we can significantly improve the validation L2 loss for models trained using 1, 2 and 4 bits low precision RFF. Noticeably, with variance reduction technique, 4 bit representation can outperform 8 bit representation in low memory budget, while 8 bit representation is consistently the best for validation L2 loss when we do not use the variance reduction technique.

We can also understand the variance reduction technique in another perspective. Given two kernel function $k(\cdot, \cdot)$ and $k'(\cdot, \cdot)$, we assume the prediction on datum x using k and k' are $h(x)$ and $h'(x)$ respectively. The prediction difference $|h(x) - h'(x)|$ has an upper bound involving two quantities $\|\boldsymbol{K} - \boldsymbol{K}'\|_2$ and $\|\boldsymbol{k}_x - \boldsymbol{k}'_x\|$ [3]; in this bound, $\boldsymbol{K}$ and $\boldsymbol{K}'$ are the training data kernel matrices, while $\boldsymbol{k}_x$ and $\boldsymbol{k}'_x$ are the test kernel values (i.e. kernel values between test sample x and training samples). Our empirical observation validates the importance of $\|\boldsymbol{k}_x - \boldsymbol{k}'_x\|$ in bounding the prediction error of a low precision RFF kernel regressor, when comparing to the prediction of a full precision RFF kernel regressor.

6 Future Directions

The next directions we plan to explore are as follows:

1. **Variable Precision**: Storing the higher variance directions in the RFF feature space with more bits. We also want to think more deeply about how to integrate the matrix factorization results into the kernel approximation context, perhaps in the context of the Nystrom method.

2. **Deterministic Rounding**: Testing the deterministic rounding ideas from Section 3.1 empirically.

3. **Fixed Design Linear Regression**: Running controlled experiments in the fixed design context, to more precisely understand the effect the different approximate spectra are having on generalization performance.

4. **Training with LP-RFF**: Developing theoretically sound and fast algorithms for training using low-precision features.

7 References

[1] Large Scale Kernel Learning using Block Coordinate Descent. Stephen Tu, Rebecca Roelofs, Shivaram Venkataraman, Benjamin Recht. `https://arxiv.org/pdf/1602.05310.pdf`.

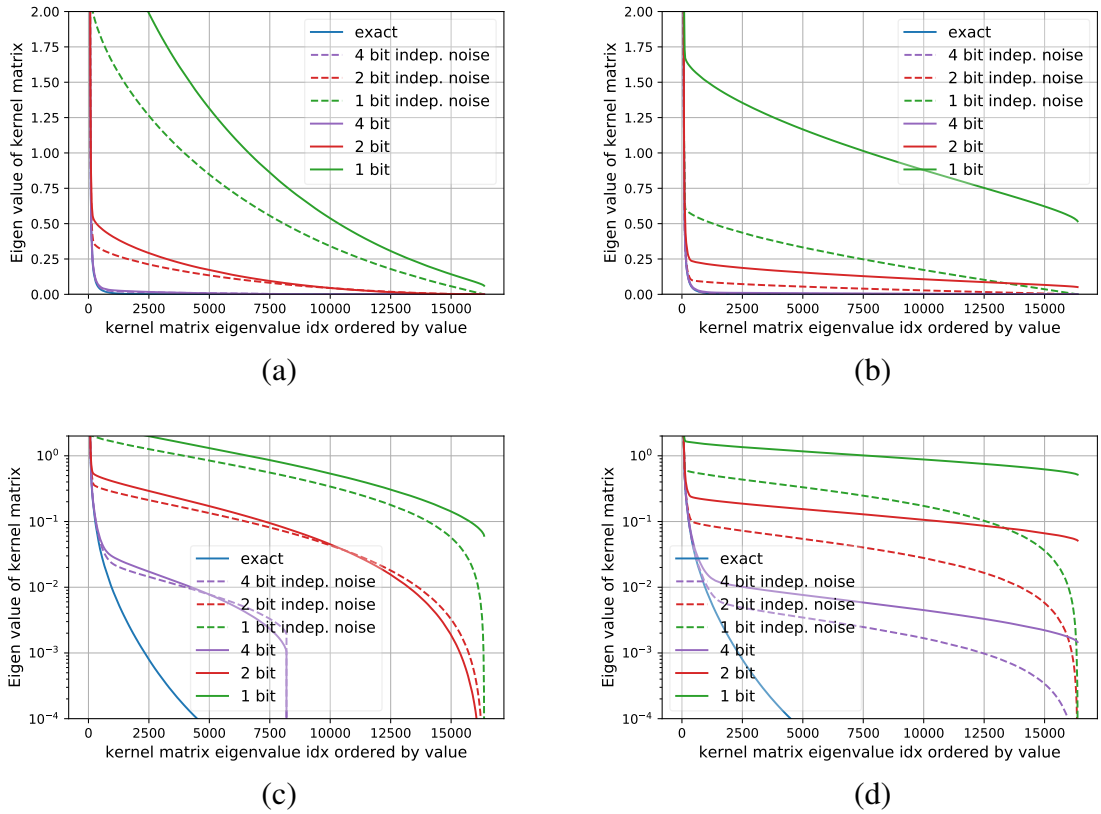


Figure 4: Spectrum (eigen values) of the kernel matrix. (a) The spectrum from different precision representation under 32k bits memory budget (1024 full precision rffs) for the feature of each sample. (b) The spectrum from different precision representation under 25.6k bits memory budget (1024 full precision rffs) for the feature of each sample. (c) and (d) is re-visualization of (a) and (b) in log-scale; it zooms in small value regions.

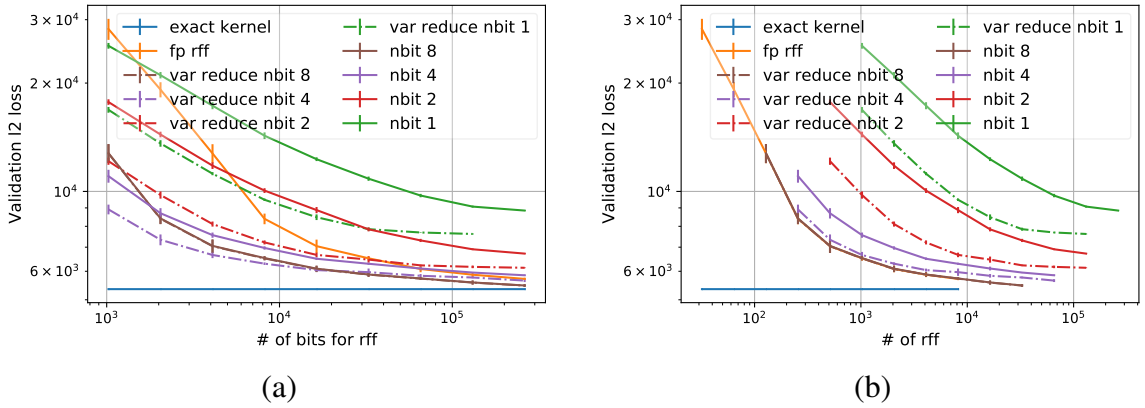


Figure 5: We train with low precision RFF and test with full precision RFF. Though the model is trained with low precision features, the test L2 loss can be improved by reducing variance in test RFF. (a) compare different precision representation under same memory bits budgets. (b) compare different precision representation with the same number of random Fourier features.

[2] CoCoA: A General Framework for Communication-Efficient Distributed Optimization. Virginia Smith, Simone Forte, Chenxin Ma, Martin Takac, Michael I. Jordan, Martin Jaggi. https://arxiv.org/abs/1611.02189

[3] On the Error of Random Fourier Features. Dougal J. Sutherland, Jeff Schneider. UAI 2015. https://arxiv.org/abs/1506.02785

[4] Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning. Ali Rahimi, Ben Recht. NIPS 2008.

[5] Random Features for Large-Scale Kernel Machines. Ali Rahimi, Benjamin Recht. NIPS 2007.

[6] Course Notes: Advanced Methods for Data Analysis, Carnegie Mellon University. Ryan Tibshirani, Spring 2014. http://www.stat.cmu.edu/~ryantibs/advmethods/notes/df.pdf

[7] High-Accuracy Low-Precision Training. Christopher De Sa, Megan Leszczynski, Jian Zhang, Alana Marzoev, Christopher Aberger, Kunle Olukotun, Christopher Re. https://arxiv.org/pdf/1803.03383.pdf