Low-Precision Random Fourier Features (LP-RFF)

Avner May

March 23, 2018

1 Spark notes

In this work we will analyze, empirically and theoretically, the performance of low-precision random features. Specifically, we will consider quantizing the features of 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). We will generally assume we are using the (δ, b) low-precision format from the HALP paper, where the numbers representable are $\delta \cdot (-2^{b-1}, \ldots, -1, 0, 1, 2^{b-1} - 1)$. To simplify analysis, and make the domain of representable values symmetric around 0, we will often consider the shifted domain $\delta \cdot (-2^{b-1} + \frac{1}{2}, -2^{b-1} + \frac{3}{2}, \ldots, -\frac{1}{2}, \frac{1}{2}, \ldots, 2^{b-1} - \frac{3}{2}, 2^{b-1} - \frac{1}{2})$.

Main questions

- Can we analyze the variance of these low-precision features, in their approximation of the kernel? Concentration inequalities?
- Can we analyze the generalization performance of these features?
- How should model training be performed 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?

I will now present some initial theoretical results. In the section below, assume that $Q_b : \mathbb{R} \to \mathbb{R}$ is a random quantization function which quantizes the interval $[-\sqrt{2}, \sqrt{2}]$ into b bits, representing the values $\frac{2\sqrt{2}}{2^b - 1} \cdot (-2^{b-1} + \frac{1}{2}, -2^{b-1} + \frac{3}{2}, \ldots, -\frac{1}{2}, \frac{1}{2}, \ldots, 2^{b-1} - \frac{3}{2}, 2^{b-1} - \frac{1}{2})$, with the property that $\mathbb{E}[Q_b(z)] = z \ \forall b, z$. Given this quantization, we will define a "quantization interval" to be the set of points between two consecutive quantized values. For $z \in [-\sqrt{2}, \sqrt{2}]$, let $\underline{q}(z)$ be the "bottom" of the quantization interval containing z, let $\overline{q}(z)$ denote the "top" of this interval.

1.1 Initial Theoretical Results

- **Proposition 1** (Variance of LP-RFF): For $x, y \in \mathcal{X}$ fixed, let $S = Z_x Z_y$, $T = Q_b(Z_x)Q_b(Z_y)$, with $\mathbb{E}[S] = \mathbb{E}[T] = k(x,y)$, and $\mathbb{E}_S\left[(S - k(x,y))^2\right] = \sigma^2$, $\mathbb{E}_{Q_b}\left[(Q_b(z_x)Q_b(z_y) - z_x z_y)^2\right] = \tilde{\sigma}_b^2$, for z_x, z_y fixed. Now, let $(S_1, \ldots, S_n)$, $(T_1, \ldots, T_n)$ 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 $\mathbb{VAR}\left[\bar{T}_n\right] = \frac{\sigma^2 + \tilde{\sigma}_b^2}{n}$. Furthermore, $\tilde{\sigma}_b^2 = (z_x - \underline{q}(z_x))(\overline{q}(z_x) - z_x)(z_y - \underline{q}(z_y))(\overline{q}(z_y) - z_y) \leq \frac{1}{4}(\frac{2\sqrt{2}}{2^b - 1})^2 = \frac{2}{(2^b-1)^2}$.

- **Proposition 2** (Concentration bounds for LP-RFF): For a fixed $x, y \in \mathcal{X}$, let S be any random variable with the property that $\mathbb{E}[S] = k(x,y)$, and $S \in [-2, 2]$.[1] Let $(S_1, \ldots, S_n)$, be a random sequence of i.i.d. draws from S, and let $\bar{S}_n = \frac{1}{n}\sum_{i=1}^n S_i$ be the empirical mean of this sequence. Then, it follows directly from Hoeffding's inequality that $\mathbb{P}\left[|\bar{S}_n - k(x,y)| \geq \epsilon\right] \leq 2\exp(-n\epsilon^2/8)$.

- **Proposition 3** (Generalization bounds for LP-RFF: 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, $R[F] = \mathbb{E}_{(x,y)\sim P}\left[c(f(x), y)\right]$)

1.1.1 Remarks

1. To better understand Proposition 1, we now plot upper bounds for $\frac{\sigma^2}{n}$ and $\frac{\sigma^2 + \tilde{\sigma}^2}{n}$, using $\sigma^2 \leq 1$, and $\tilde{\sigma}^2 \leq \frac{2}{(2^b-1)^2}$; we use a log-log plot, and plot these upper bounds for various values of b

[1] $S = Q_b(\sqrt{2}\cos(w^T x + b))Q_b(\sqrt{2}\cos(w^T y + b))$ satisfies these conditions.

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.

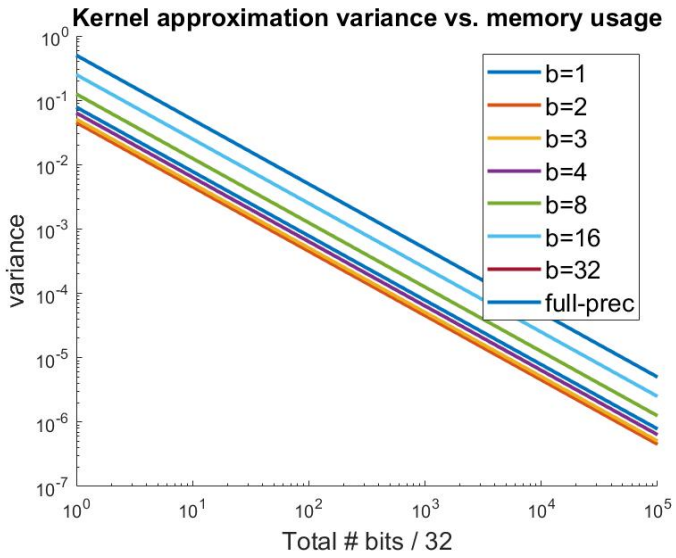
2: Featurize the input: $z_i \leftarrow [Q(\phi(x_i; w_1); \theta_1), \ldots, Q(\phi(x_i; w_m); \theta_m)]^T$.

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

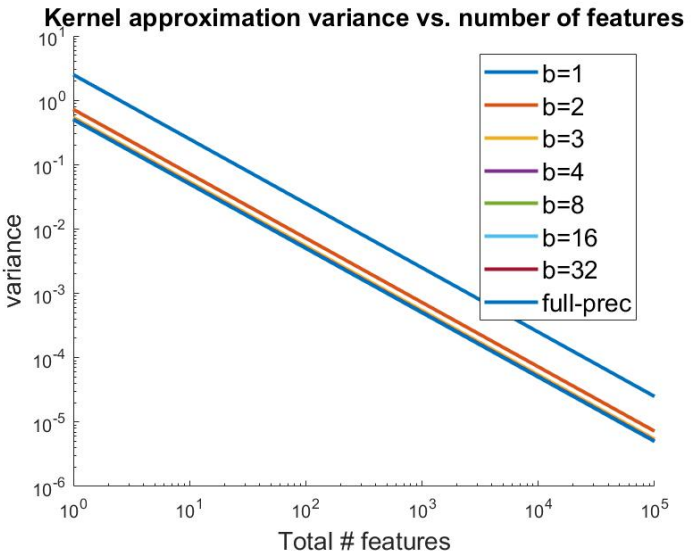
$$\alpha* = \arg\min_{\alpha \in \mathbb{R}^m} \frac{1}{n} \sum_{i=1}^{n} c(\alpha^T z_i, y_i)$$
$$s.t. \|\alpha\|_\infty < C/m$$

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

and n.



As you can see, lowering the precision helps reduce the variance by approximately a full order of magnitude ($10\times$ smaller variance) for a fixed number of bits, with $b = 2$ giving the lowest variance, and $b = 32$ variance matching the full-precision line (they are overlapping in the figure). In the plot below, we show the kernel approximation variance as a function of the number of features. As you can see, using 1 bit per feature gives a noticeable jump in variance, but all other higher precision features perform comparably to the full-precision random features.



2. It is important to note that Propositions 2 and 3 hold as a function of the number of low-precision features, and *match* the bounds for the *same number* of high-precision features.[2] This is pretty amazing!

1.2 Open Questions

1. Can we extend Claim 1 (Uniform convergence of Fourier features) from [5] (Rahimi and Recht, 2007) to low-precision features $z(x)$, to prove that the probability that there exists $x, y \in \mathcal{X}$ such that $|z(x)^T z(y) - k(x,y)| \geq \epsilon$ is small?

2. Can we perform training in such a way that the learned weights are also low-precision? What guarantees can we get with a training algorithm of this form?

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

4. What can we say about the spectrum of the low-precision features relative to the full-precision features, and the exact kernel? Results here would allow us to apply the "fixed design" linear regression analysis (from my other notes) here.

5. Can we use low-precision for x,w, and b in the features $z(x) = \sqrt{2}\cos(w^T x + b)$? What can we prove about this?

[2]Some care is needed for Proposition 3, because the set of functions $\mathcal{F}_p$ we are considering is different than the one in the original. I believe I can show that for a properly defined set of quantization functions, our $\mathcal{F}_p$ is a superset of the original, though I may need to update the constant C.

2 Motivation

Random Fourier features are an effective way of scaling kernel methods to large datasets. Unfortunately, this method often requires a very large number of features in order to attain strong performance. For example, in my work in speech recognition, increasing the number of features from 100k to 400k continued to give meaningful improvements. At this scale, the amount of memory and computation time required to train models becomes a big bottleneck. For example, the size of a GPU's global memory can limit the number of random features which can be used, if one would like to train the model fully on a single GPU.

Furthermore, in my work comparing random Fourier features and the Nyström method, I observed that using *many* random Fourier features allows for approximating a larger portion of the kernel's spectrum, which appears to be important for attaining strong performance. This is in contrast to the Nyström method, which under a similar computational budget, computes fewer more expensive features; although these features approximate the kernel matrix very well, they are inherently limited in how many of the kernel's eigenvalues they can approximate. The take-away here appears to be: many cheap features is better than fewer expensive features, even if those expensive features approximate the kernel matrix very well.

In light of the computational bottleneck which arises when training models with very many random features, as well as the observation that using a large number of features is important for downstream performance (even if they have higher variance when approximating the kernel), I propose using *low-precision* as a way to further scale these random feature methods. Some challenges involved in implementing and analyzing this idea are as follows:

1. What are the trade-offs in deciding the number of bits of precision which should be used for the data, the random projection matrix, the random features, and the model parameters? Can we effectively learn models on single-bit random features?

2. If the model is stored in low-precision, what optimization algorithm should be used during training in order to make this possible?

3. Will the final model be low-precision or high-precision? Another way of asking this is: Is low-precision a trick to speed up the training of a full-precision model (as in HALP), or will the entire system be low-precision at both train and test time?

4. How can we formalize the way the additional noise caused by quantization will affect the training of the model? Can we use variance reduction techniques like SVRG to deal with this, like in HALP?

5. Can we somehow use this additional randomness in the feature generation process in order to generate "error bars" in the estimates of the trained model (e.g., by evaluating the model on several random draws of the randomly quantized features for the same data point, in order to produce a distribution of predictions)?

I will now discuss a few ideas related to two aspects of this project: (1) computing the random features, and (2) training a model with these features.

3 Computing the low-precision random features

Here, I will discuss different ways of computing $\tilde{z}(x)$, the quantized version of $z(x) = \sqrt{\frac{2}{m}}\cos(W^T x + b) \in \mathbb{R}^m$, where $x \in \mathbb{R}^d$, $W \in \mathbb{R}^{d \times m}$, $b \in \mathbb{R}^m$, and $\cos(\cdot)$ is the element-wise cosine non-linearity. Note that below I will ignore the factor of $\sqrt{\frac{2}{m}}$, as it can be stored separately as the "scale" parameter δ of the b-bit low-precision representation (δ, b).

- Compute $\cos(W^T x + b)$ using full precision (where W can be a structured random matrix to save time/space), and then quantize the output.
- First, quantize x, W, and b, as $\tilde{x}$, $\tilde{W}$, $\tilde{b}$. Then, compute $\tilde{W}^T\tilde{x} + \tilde{b}$, pass this through the cosine function, and quantize the output.
- Use low-precision floating point operations (eg, 16-bit) for all of the operations.

One important thing to note is that quantizing x,W, and b in such a way that $\mathbb{E}_{\tilde{x},\tilde{W},\tilde{b}}\left[\tilde{W}^T\tilde{x} + \tilde{b}\right] = W^T x + b$ does *not* mean that $\mathbb{E}_{\tilde{x},\tilde{W},\tilde{b}}\left[\cos(\tilde{W}^T\tilde{x} + \tilde{b})\right] = \cos(W^T x + b)$. Thus, features generated in this way would not necessarily produce unbiased estimates of the kernel function, which satisfy $\mathbb{E}\left[z(x)^T z(y)\right] = k(x, y)$. For this reason, I think the first option above is likely the best path forward.

In the case of 1-bit random features, we could quantize as follows: Let $z'_i(x) = \cos(w_i^T x + b_i)$ be the unnormalized i^{th} feature of $z(x)$, and let $\tilde{z}'_i(x)$ be its quantized version, which we are discussing. In order for $\mathbb{E}\left[\tilde{z}'_i(x)\right] = z'_i(x)$, we could set $\tilde{z}'_i(x) = +1$ with probability $\frac{1+z'_i(x)}{2}$, and $\tilde{z}'_i(x) = -1$ with probability $\frac{1-z'_i(x)}{2}$. Here, we were using $\tilde{z}'_i(x) \in \{-1, +1\}$. Note that one can simulate storing a vector $x \in \{-1, +1\}^d$ as a binary vector $\hat{x} \in \{0, 1\}^d$ by noticing that $\langle x, y\rangle = 2\langle \hat{x}, \hat{y}\rangle - d$.

4 Training a model using low-precision random features

In this section, we address the question of how to train a model using low-precision random features as input. Essentially, this comes down to training a linear model on top of random features $\tilde{z}$ such that $\mathbb{E}\left[\tilde{z}\right] = z$. One potentially great option is to use "LM-HALP", the version of HALP designed for

learning linear models, presented as Algorithm 4 in the recent submission. 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 doesn't fit in memory). Note that any of the options below which use 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 HALP paper. 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. We present some alternative options to LM-HALP below:

- **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 $p_t = p(Y_t = 1 | z_t, w_t) = (1 + \exp(-w_t^T z_t))^{-1}$. So in the case where $y_t = 1$, $\tilde{g}_t$ could be equal to 1 with probably $1 - p_t$, and 0 otherwise; and in the case where $y_t = 0$, $\tilde{g}_t$ could be equal to -1 with probability p_t, and 0 otherwise. Here, I am assuming that η is stored in the scale factor δ of the low-precision format (δ, b) for $\tilde{g}_t$. 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. Note also that computing p_t requires using the $\exp(\cdot)$ and $(\cdot)^{-1}$ operations, which would probably need to be done as floating point operations.

- **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.

5 Kernel Approximation Variance Analysis for Low-Precision RFF

Definitions: 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 $a - z$. Furthermore, let $Q^{a,c}(z) = z + X_z^{a,c} \in \{a, c\}$ be the "quantized" version of z, corresponding to randomized rounding to a or c. The following lemmas are easy to verify:

Lemma 1: Using the definitions above, it follows that $\mathbb{E}[X_z^{a,c}] = 0$, $\mathbb{VAR}[X_z^{a,c}] = (z - a)(c - z)$, $\mathbb{E}[Q^{a,c}(z)] = z$, and $\mathbb{VAR}[Q^{a,c}(z)] = (z - a)(c - z)$.[3]

Lemma 2: Given $z, z' \in [-c, c]$, let $S = Q^{-c,c}(z)Q^{-c,c}(z')$. Then $\mathbb{E}[S] = zz'$ and $\mathbb{VAR}[S] = c^4 - z^2 z'^2$.

Lemma 3: Assume $z_x \sim Z_x, z_y \sim Z_y$, where $z_x, z_y \in [-c, c]$, and $\mathbb{E}[Z_x Z_y] = k(x, y)$, for some $x, y \in \mathcal{X}$.[4] Assume $\mathbb{VAR}[Z_x Z_y] = \sigma^2$, and let $\tilde{\sigma}^2 = \mathbb{VAR}[Q^{-c,c}(z_x)Q^{-c,c}(z_y)] = c^4 - z_x^2 z_y^2$. Now, letting $T = Q^{-c,c}(Z_x)Q^{-c,c}(Z_y)$, it follows that $\mathbb{E}[T] = k(x, y)$, and $\mathbb{VAR}[T] = \sigma^2 + \tilde{\sigma}^2$.

Proposition 1: Let $S = Z_x Z_y$, $T = Q^{-c,c}(Z_x)Q^{-c,c}(Z_y)$, 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 $\mathbb{VAR}\left[\bar{T}_n\right] = \frac{\sigma^2 + \tilde{\sigma}^2}{n}$.

Now, let's 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

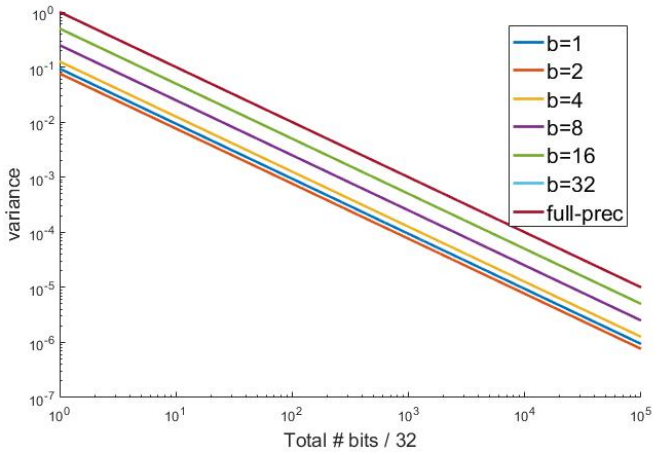
[3] Note that if $a = 0$, $\mathbb{VAR}\left[X_z^{0,c}\right] = z(c - z)$, which is maximized at $z = c/2$, with variance $c^2/4$. If $a = -c$, $\mathbb{VAR}[X_z^{-c,c}] = c^2 - z^2$, which is maximized at $z = 0$ with variance c^2.

[4] 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.

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 $r = \frac{2\sqrt{2}}{2^b - 1}$. Thus, from Lemma 1 (see footnote) the quantization error $\tilde{\sigma}_b^2 \leq r^2/4 = \frac{2}{(2^b-1)^2}$. This allows us to perform the comparison discussed above:

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

We now plot these two upper bounds on a log-log plot, for various values of b and n.



As you can see, lowering the precision helps reduce the variance by approximately a full order of magnitude ($10\times$ smaller variance), with $b = 2$ giving the lowest variance, and $b = 32$ variance matching the full-precision line (they are overlapping in the figure).

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

Proposition 2: For a fixed $x, y \in \mathcal{X}$, let S be any random variable with the property that $\mathbb{E}[S] = k(x, y)$, and $S \in [-2, 2]$. Let $(S_1, \ldots, S_n)$, be a random sequence of i.i.d. draws from S, and let $\bar{S}_n = \frac{1}{n}\sum_{i=1}^n S_i$ be the empirical mean of this sequence. Then, it follows directly from Hoeffding's inequality that $\mathbb{P}\left[|\bar{S}_n - k(x, y)| \geq \epsilon\right] \leq 2\exp(-n\epsilon^2/8)$.

Open Question: Can Claim 1 from Rahimi and Recht (2007) be adapted to this low-precision setting? The most obvious obstacle here is that the randomness depends on the data; in other words, each $x \in \mathbb{R}$ has a unique quantization noise distribution. In other words, with the current definition, there is no way to draw all the random feature parameters upfront, in such a way that given these parameters, the feature functions are deterministic. Can we somehow switch the quantization scheme for it to be deterministic? For example, splitting the interval $[-\sqrt{2}, \sqrt{2}]$ into r sub-intervals—then, for interval

6 Generalization bounds for low-precision RFF

We now discuss how to apply the result from Theorem 1 of Rahimi and Recht's 2008 paper to this low-precision setting [4]. That theorem assumes that there is a set of (deterministic) basis functions $\phi : \mathcal{X} \times \Omega \to \mathbb{R}$. It then argues that if $\{w_1, \ldots, w_K\}$ are drawn independently from some distribution p on Ω, then the generalization performance of the model trained on the features $\phi(\ \cdot\ ; w_i)$ is close to the best possible generalization performance of any model in the set $\mathcal{F}_p = \{f(x) = \int_\Omega \alpha(w)\phi(x; w)dw || \alpha(x)| \leq Cp(w)\}$. Unfortunately, our randomly quantized features currently do not fit nicely into this framework, because the basis functions themselves are random, even for a fixed x and w. The purpose of the section below is to construct a parametrized set of basis functions $\phi(x; w, b, \theta)$ which quantize $\cos(w^T x + b)$ in a deterministic way, given the parameters θ. This construction is a bit involved, but the goal is simple:

We discuss how to deal with this below.

6.1 Making random features "deterministic"

Let $\gamma > \epsilon_r > 0$ be two positive constants. Assume that we are using b bits for our random features, and use these b bits to express 2^b distinct values in the interval $[-(\sqrt{2} + \gamma), \sqrt{2} + \gamma]$. If we divide the interval into $2^b - 1$ evenly sized intervals, each of size $\frac{2(\sqrt{2}+\gamma)}{2^b-1}$, this allows us to express the following set of values: $\frac{2(\sqrt{2}+\gamma)}{2^b-1} \cdot (-2^b + 1, -2^b + 3, \ldots, -1, 1, \ldots, 2^b - 3, 2^b - 1)$.[5] We will call these intervals the "quantization intervals", whose boundaries correspond to the values expressible in b bits. Let $\hat{Q}_b$ denote this set of values. Now, we will define another set of intervals, whose union will "cover" the set $[-\sqrt{2}, \sqrt{2}]$. We will call these the "rounding intervals", and define them as follows. Let a be a uniformly at random "offset" chosen in the interval $[0, \epsilon_r]$. Now, consider the

[5]Note that for simplicity here, I am dividing the entire interval into $2^b - 1$ sub-intervals, which gives a symmetric set of 2^b positive and negative values expressible in b bits, not including 0. This differs from the presentation in the HALP paper, in which the set of expressible numbers is of the form $\delta \cdot (-2^{b-1}, \ldots, -1, 0, 1, \ldots, 2^{b-1} - 1)$. We could use easily use the same notation as the HALP paper by using the values $\frac{2(\sqrt{2}+\gamma)}{2^b-2} \cdot (-2^{b-1}, \ldots, -1, 0, 1, \ldots, 2^{b-1} - 1)$, though this "wastes" the smallest value $-\delta 2^{b-1}$ because it is smaller than $-(\sqrt{2} + \gamma)$.

following sequence of interval boundaries: $(-\sqrt{2}-a, -\sqrt{2}-a+\epsilon_r, -\sqrt{2}-a+2\epsilon_r, \ldots, -\sqrt{2}-a+R\epsilon_r)$, where $R = \lceil(\frac{2\sqrt{2}+a}{\epsilon_r})\rceil$ is the smallest integer such that $-\sqrt{2}-a+R\epsilon_r \geq \sqrt{2}$. Let $\hat{R}_a$ denote the set of boundary points for these rounding intervals, and let $c_k = \frac{1}{2}\Big((-\sqrt{2}-a+(k-1)\epsilon_r) + (-\sqrt{2}-a+k\epsilon_r)\Big)$ denote the center of the k^{th} rounding interval. For any value $z \in [-\sqrt{2}, \sqrt{2}]$, let $\underline{q}(z)$ be the "bottom" of the quantization interval containing z, let $\overline{q}(z)$ denote the "top" of this interval, let $\underline{r}(z)$ be the "bottom" of the rounding interval containing z, let $\overline{r}(z)$ denote the "top" of this interval, and let $c(z) = \frac{1}{2}(\overline{r}(z)+\underline{r}(z))$ denote the center of this rounding interval. Now, let $(X_1, \ldots, X_{R+1})$ denote $R+1$ Bernoulli random variables. The probability p_k that $X_k = 1$ is defined as follows:

- If the interval $[\underline{r}(c_k), \overline{r}(c_k)]$ *does not* contain one of the quantization boundaries:

$$p_k = \frac{c_k - \underline{q}(c_k)}{\overline{q}(c_k) - \underline{q}(c_k)}.$$

- If the interval $[\underline{r}(c_k), \overline{r}(c_k)]$ *does* contain one of the quantization boundaries:

$$p_k = \frac{c_k - \underline{q}(\underline{r}(c_k))}{\overline{q}(\overline{r}(c_k)) - \underline{q}(\underline{r}(c_k))}.$$

Let $k(z)$ denote the *index* of the rounding interval in which z falls: specifically, $k(z) = k' \Leftrightarrow z \in [-\sqrt{2}-a+(k'-1)\epsilon_r, -\sqrt{2}-a+k'\epsilon_r]$. Now, if $\theta = (\theta_1, \ldots, \theta_{R+1})$ are the outcomes of these $R+1$ Bernoullis, and a is the random offset, we can define $g_{a,\theta}(z)$ as follows:

- If the interval $[\underline{r}(z), \overline{r}(z)]$ *does not* contain one of the quantization boundaries:

$$g_{a,\theta}(z) = (1-\theta_{k(z)})\underline{q}(z) + \theta_{k(z)}\overline{q}(z).$$

- If the interval $[\underline{r}(z), \overline{r}(z)]$ *does* contain one of the quantization boundaries:

$$g_{a,\theta}(z) = (1-\theta_{k(z)})\underline{q}(\underline{r}(z)) + \theta_{k(z)}\overline{q}(\overline{r}(z)).$$

Note that in the above definitions, even though the RHS does not contain a, it is implicitly dependent on a through the $\overline{r}$ and $\underline{r}$ functions. Now, we are finally ready to define a set of "deterministic" basis functions:[6]

$$\phi(x; w, b, a, \theta) \quad = \quad g_{\theta,a}(\cos(w^T x + b)).$$

[6]I apologize for overloading the letter b for both the number of bits, and the value in the expression $\cos(w^T x + b)$.

Lemma 5: If z and z' are in different rounding intervals, it follows that

$$zz' \leq \mathbb{E}_{a,\theta}\left[g_{a,\theta}(z)g_{a,\theta}(z')\right] \leq zz' + O(\epsilon_r).$$

Proof: We will first consider the case where z is in the k^{th} rounding interval $[\underline{r}(c_k), \overline{r}(c_k)]$, and this interval *does not* contain one of the quantization boundaries. In this case, we will show that $\mathbb{E}_{\theta}\left[g_{a,\theta}(z)\right] = c_k$ (regardless of the value of a). The case where the interval does contain a quantization boundary is very similar:

$$\begin{aligned}
\mathbb{E}_{\theta}\left[g_{a,\theta}(z)\right] &= \mathbb{E}_{\theta_k}\left[(1-\theta_k)\underline{q}(z) + \theta_k\overline{q}(z)\right] \\
&= (1-p_k)\underline{q}(z) + p_k\overline{q}(z) \\
&= \left(1 - \frac{c_k - \underline{q}(c_k)}{\overline{q}(c_k) - \underline{q}(c_k)}\right)\underline{q}(z) + \frac{c_k - \underline{q}(c_k)}{\overline{q}(c_k) - \underline{q}(c_k)}\overline{q}(z) \\
&= \frac{\overline{q}(c_k) - c_k}{\overline{q}(c_k) - \underline{q}(c_k)}\underline{q}(z) + \frac{c_k - \underline{q}(c_k)}{\overline{q}(c_k) - \underline{q}(c_k)}\overline{q}(z) \\
&= \frac{\overline{q}(c_k)\underline{q}(z) - c_k\underline{q}(z) + c_k\overline{q}(z) - \underline{q}(c_k)\overline{q}(z)}{\overline{q}(c_k) - \underline{q}(c_k)} \\
&= \frac{c_k\left(\overline{q}(z) - \underline{q}(z)\right)}{\overline{q}(c_k) - \underline{q}(c_k)} \\
&= c_k.
\end{aligned}$$

In this sequence of equations, I used the fact that $\overline{q}(z) = \overline{q}(c_k)$, and $\underline{q}(z) = \underline{q}(c_k)$.

Now, I will use this fact to show that for z and z' in different rounding intervals ($k = k(z)$, $k' = k(z')$, $k \neq k'$), $\mathbb{E}_{a,\theta}\left[g_{a,\theta}(z)g_{a,\theta}(z')\right] = zz'$.

$$\begin{aligned}
\mathbb{E}_{a,\theta}\left[g_{a,\theta}(z)g_{a,\theta}(z')\right] &= \mathbb{E}_{a}\left[\mathbb{E}_{\theta}\left[g_{a,\theta}(z)g_{a,\theta}(z')\right]\right] \\
&= \mathbb{E}_{a}\left[\mathbb{E}_{\theta_k}\left[g_{a,\theta}(z)\right]\mathbb{E}_{\theta_{k'}}\left[g_{a,\theta}(z')\right]\right] \\
&= \mathbb{E}_{a}\left[c_k c_{k'}\right]
\end{aligned}$$

We can apply the above lemma as follows: Assume $x, y \in \mathcal{X}$, and that $z_x = \sqrt{2}\cos(w^T x + b)$ and $z_y = \sqrt{2}\cos(w^T y + b)$. It then follows from the lemma that:

$$\begin{aligned}
\mathbb{E}_{w,b,a,\theta}\left[g_{a,\theta}(z_x)g_{a,\theta}(z_y)\right] &= \mathbb{E}_{w,b}\left[z_x z_y\right] \\
&= k(x,y)
\end{aligned}$$

Thus, $\phi(x; w, b, a, \theta)\phi(y; w, b, a, \theta)$ is an unbiased estimate of $k(x,y)$ whenever z_x and z_y are in

different rounding intervals.

7 Why 1-bit spectrum is terrible

$$
\begin{aligned}
A &= XX^T \\
&= USU^T \\
trace(A^TA) &= trace(USU^TUSU^T) \\
&= trace(US^2U^T) \\
&= \sum_i \lambda_i^2 \\
\tilde{A} &= (X+C)(X+D)^T \\
\tilde{A}^T\tilde{A} &= (X+D)(X+C)^T(X+C)(X+D)^T \\
&= XX^TXX^T + XC^TCX^T + DX^TXD^T + DC^TCD^T + \text{zero-mean} \\
&= A^TA + XC^TCX^T + DX^TXD^T + DC^TCD^T + \text{zero-mean} \\
trace(\tilde{A}^T\tilde{A}) &= trace(A^TA) + trace(XC^TCX^T) + trace(DX^TXD^T) + trace(DC^TCD^T) + trace(\text{zero-mean}) \\
&= \sum_i \tilde{\lambda_i}^2 \\
\mathbb{E}\left[\sum_i \tilde{\lambda_i}^2\right] &= \mathbb{E}\left[trace(\tilde{A}^T\tilde{A})\right] \\
&= trace(A^TA) + \mathbb{E}\left[trace(XC^TCX^T) + trace(DX^TXD^T) + trace(DC^TCD^T)\right] + \mathbb{E}\left[trace(\text{zero-mean})\right] \\
&= trace(A^TA) + \mathbb{E}\left[trace(XC^TCX^T) + trace(DX^TXD^T) + trace(DC^TCD^T)\right] \\
&\geq trace(A^TA) \\
&= \sum_i \lambda_i^2 \\
\mathbb{E}\left[z^T\tilde{A}^T\tilde{A}z\right] &= z^TA^TAz + \mathbb{E}\left[z^T(XC^TCX^T + DX^TXD^T + DC^TCD^T)z\right] + \mathbb{E}\left[z^T\text{zero-mean}z\right] \\
&\geq z^TA^TAz
\end{aligned}
$$

So taking $z = u_i$, we get $\|\tilde{A}u_i\|^2 \geq \|Au_i\|^2 = \lambda_i^2$. Can we show $\tilde{A} - A$ is PSD?

8 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`.

[2] CoCoA: A General Framework for Communication-Efficient Distributed Optimization. Vir-

ginia 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: