# Gaussian Quadrature for Kernel Features

Tri Dao†, Christopher De Sa‡, Christopher Ré†
trid@stanford.edu, cdesa@stanford.edu, chrismre@cs.stanford.edu
Department of Computer Science, † Stanford University, ‡ Cornell University

## Overview

**Kernel methods: simple to train and well-understood.**

- Competitive with DNNs in speech recognition.
- Can scale with approximate feature maps such as random Fourier features (Rahimi and Recht NIPS 2007).

**Can we derandomize kernel approximation?**

**Our contribution: *deterministic kernel features*.**

- Require much fewer features asymptotically.
- Work well with sparse ANOVA (convolutional) kernels.
- Application in speech recognition.

For error $\epsilon$, dimension $d$, any $\gamma > 0$:

| | Random Fourier | Gaussian quadrature |
|---|---|---|
| Features | $\tilde{\Omega}\left(d\epsilon^{-2}\right)$ | $O(2^d(e^\gamma + \epsilon^{-1/\gamma}))$ |
| Guarantee | Probabilistic (whp) only | Absolute |

## Kernel Approximation

Approximate a shift-invariant kernel by a feature map $z$:

$$k(x, y) = k(x - y) \approx \langle z(x), z(y) \rangle = \tilde{k}(x, y).$$

By Fourier inversion (Bochner's theorem):

$$k(x - y) = \int_{\mathbb{R}^d} \Lambda(\omega) \exp(j\omega^T(x - y))\, d\omega.$$

**Kernel approximation is integral approximation.**

Integral approximation:

$$\tilde{k}(x - y) = \sum_{i=1}^{D} a_i \exp(j\omega_i^T(x - y)).$$

***Approximate feature map***:

$$z(x) = \left[\sqrt{a_1}\exp(j\omega_1^T x) \ \ldots \ \sqrt{a_D}\exp(j\omega_D^T x)\right]^T.$$

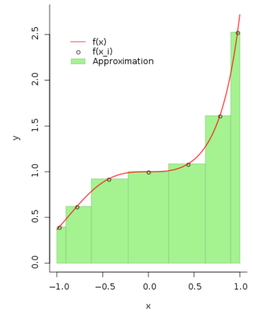
- Random Fourier features ↔ Monte-Carlo sampling.
- Deterministic features ↔ Approximation by **Gaussian quadrature**.

## Kernels and Quadratures

***Gaussian quadrature*** approximates integrals of the form

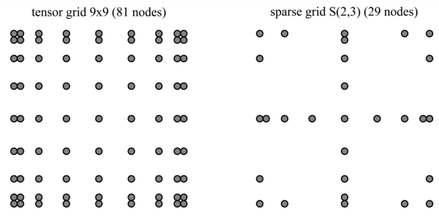
$$\int \Lambda(\omega) f(\omega)\, d\omega \approx \sum_{i=1}^{D} a_i f(\omega_i).$$

- $D$ points are exact for polynomials of degree up to $2D - 1$.
- Points are chosen **deterministically** to get lowest approximation error with fewest points.
- Equivalent to expanding the integrand in the basis of orthogonal polynomials.




## Dense and Sparse Grid Quadratures

- ***Dense grid***: tensor products of points and weights of the one-dimensional quadrature.
- ***Sparse grid***: similar error with exponentially fewer points.

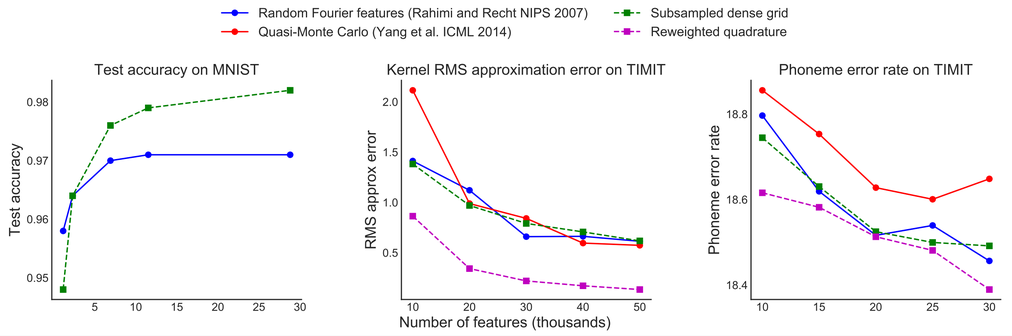



### Main Theorem: Bounding the approximation error.

Under a quadrature rule that is exact up to some even degree $R$, for all $\|x - y\| \leq M$, the approximation error is bounded by $\left|k(x - y) - \tilde{k}(x - y)\right| \leq \frac{2e}{\sqrt{\pi}}\left(\frac{eb^2M^2}{R}\right)^{\frac{R}{2}}$.
Moreover, for any $\gamma > 0$, the sample complexity of sparse grid features is

$$D(\epsilon) = O\left(2^d(e^\gamma + \epsilon^{-\frac{1}{\gamma}})\right).$$

## Experiments




## Reweighted Grid Quadrature

***Subsampled grid*** quadrature: subsampling the dense grid and sparse grid points according to their weights.

***Reweighted grid*** quadrature: reweight the grid points to minimize the kernel approximation error on a small subset of the data.

- Adapts to data distribution.
- Yields much lower approximation error.

## Sparse ANOVA (Convolutional) Kernels

**When do deterministic features work well despite exponential dependence on $d$?**

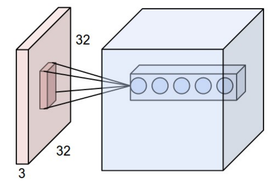
***Sparse ANOVA kernels*** (*convolutional kernels*):

- Operate similarly to the convolutional layer in CNNs.
- Are used in speech and images.
- Have **low effective dimension**.

A kernel of this type can be written as

$$k(x, y) = \sum_{S \in \mathcal{S}} k_S(x_S - y_S),$$

where $\mathcal{S}$ is a set of subsets of the variables in $\{1, \ldots, d\}$, and $k_S$ is a sub-kernel acting on indices in $S$.




Sparse ANOVA kernels capture ***local interaction*** between covariates, like a convolutional layer in CNNs.

For kernel size $m = |\mathcal{S}|$, the number of sub-kernels:

| | Random Fourier | Gaussian quadrature |
|---|---|---|
| No. of features | $\tilde{\Omega}\left(m^3\epsilon^{-2}\right)$ | $O(m\epsilon^{-1/\gamma})$ |

Better than random Fourier features *both* in terms of the ***kernel size*** $m$ and the ***desired error*** $\epsilon$.