Bayesian Nonparametric Kernel-Learning

Junier B. Oliva* **Avinava Dubey*** **Andrew G. Wilson** **Barnabás Póczos**
Jeff Schneider **Eric P. Xing**
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
{joliva, akdubey, andrewgw, bapoczos, schneide, epxing}@cs.cmu.edu

Abstract

Kernel methods are ubiquitous tools in machine learning. However, there is often little reason for the common practice of selecting a kernel a priori. Even if a universal approximating kernel is selected, the quality of the finite sample estimator may be greatly affected by the choice of kernel. Furthermore, when directly applying kernel methods, one typically needs to compute a $N \times N$ Gram matrix of pairwise kernel evaluations to work with a dataset of N instances. The computation of this Gram matrix precludes the direct application of kernel methods on large datasets, and makes kernel learning especially difficult.

In this paper we introduce Bayesian nonparmetric kernel-learning (BaNK), a generic, data-driven framework for scalable learning of kernels. BaNK places a nonparametric prior on the spectral distribution of random frequencies allowing it to both learn kernels and scale to large datasets. We show that this framework can be used for large scale regression and classification tasks. Furthermore, we show that BaNK outperforms several other scalable approaches for kernel learning on a variety of real world datasets.

1 Introduction

Kernel methods such as support vector machines (SVMs), kernel-ridge regression, kernel-PCA, and Gaussian processes (GPs) have become the cornerstone of many machine learning approaches. However, the choice of kernel, which profoundly affects performance, has until recently received little attention. In fact, finite sample estimates will be affected by the kernel choice notwithstanding the use of an universal approximating kernel. Indeed, the a priori choice of a fixed kernel in kernel methods is typically ad-hoc and not data-driven. Even when learning kernel hyperparameters, one is typically limited to an arbitrarily chosen and restrictive family of kernel functions, exploring only a very small subset of reasonable possibilities. Given that the choice of kernel is an important free parameter in kernel methods, and generally there are few a priori reasons for kernel selections, a principled and data-driven method for learning kernels is extremely useful.

Furthermore, kernel methods often do not scale to datasets with a large number of instances due to the need to compute and store an $N \times N$ Gram matrix, $\mathbf{K}$, for N training points. Moreover, kernel methods, such as GPs, will often require manipulations of $\mathbf{K}$ like solving linear systems and computing log determinants, leading to a $O(N^3)$ time complexity. Considering that modern datasets are only increasing in size, and complicated machine learning tasks require large datasets, it is vital to mitigate the high computational cost of kernel methods.

In order to provide a method that scales to large datasets and adaptively learns the kernel to use in a data-driven fashion, this paper presents the *Bayesian nonparametric kernel-learning* (BaNK) framework. BaNK is a novel approach that will use random features to both provide a scalable solution and learn kernels. This approach scales through random features and places a Bayesian nonparametric distribution over kernels, with support for any stationary kernels.

Random features have been recently shown to be an effective way to scale kernel methods to large

* denotes equal contribution

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 41.

(a) RKS (b) BaNK

Figure 1: (a) Traditional random feature approach where the distribution $\mathcal{D}$ of random features W is held fixed. (b) BaNK framework where $\mathcal{D}$ is random.

datasets. Roughly speaking, random feature techniques like random kitchen sinks (RKS) [Rahimi and Recht, 2007] work as follows. Given a shift invariant kernel $K(x, x') = k(x - x')$, one constructs an approximate primal space to estimate kernel evaluations $K(x, x')$ as the dot product of finite vectors $\varphi(x)^T\varphi(x')$. The vectors φ are constructed with random frequencies drawn from a distribution $\mathcal{D}$ that is defined by K. Similarly, a distribution $\mathcal{D}$ from which random frequencies are drawn from defines a kernel K that the random frequencies approximate. It is this last observation that is key for the BaNK framework. BaNK will allow the distribution $\mathcal{D}$ to vary with the given data, effectively learning the kernel. In particular, BaNK shall vary $\mathcal{D}$ with a graphical model approach where we treat $\mathcal{D}$ as a latent parameter and place a prior on it (Figure 1). The prior on $\mathcal{D}$, along with the data generation model, will allow one to sample from a posterior over $\mathcal{D}$ in order to learn the corresponding kernel. We model $\mathcal{D}$ as a mixture of Gaussians with a Dirichlet process prior, which allows BaNK to learn a kernel from a rich, broad class. Furthermore, with the use of random features, we are able to efficiently sample the model parameters and work over larger datasets. Moreover, by using Metropolis-Hastings we sample from a proper posterior, thus the kernels we learn are interpretable since the random features are asymptotically guaranteed to come from the underlying posterior distribution unlike greedy non-convex optimization methods.

Outline The rest of this paper is structured as follows. First we review the use of random features for kernel approximation and show how such an approach can be used for flexible and efficient kernel learning. Second, we detail our graphical model framework both for supervised regression and classification tasks. Third, we expound on our inference method for sampling from the model posterior. Forth, we illustrate the use and performance of BaNK for both regression and classification on several datasets. Lastly, we cover related works and give concluding remarks.

2 Model

2.1 Random Features for Kernel Estimation

Below we briefly review the method of random Fourier features for the approximation of kernels [Rahimi and Recht, 2007]. The details of the method will help motivate and explain our BaNK model. Henceforth, we will only consider continuous shift-invariant kernels defined over $\mathbb{R}^d$: $K(x, y) = k(x-y)$ where $x, y \in \mathbb{R}^d$ and k is a positive definite function. The use of random Fourier features for kernel approximation is a result of Monte Carlo integration using Bochner's theorem [Rudin, 1990]. Bochner's theorem states that a continuous shift-invariant kernel $K(x, y) = k(x-y)$ is a positive definite function if and only if $k(t)$ is the Fourier transform of a non-negative measure $\rho(\omega)$. Note further, that if $k(0) = 1$, then $\rho(\omega)$ will be a normalized density. That is, if we define $\zeta_\omega(x) \equiv \exp(\mathrm{i}\,\omega^T x)$, then

$$k(x-y) = \int_{\mathbb{R}^d} \rho(\omega) \exp\left(\mathrm{i}\,\omega^T(x-y)\right) \mathrm{d}\omega = \mathbb{E}_{\omega\sim\rho}[\zeta_\omega(x)\zeta_\omega(y)^*]. \quad (1)$$

Hence, using Monte Carlo integration, we can approximate $K(x, y) = k(x - y)$ using $\omega_j \overset{iid}{\sim} \rho$:

$$k(x-y) \approx \frac{1}{M}\sum_{j=1}^{M} \zeta_{\omega_j}(x)\zeta_{\omega_j}(y)^*. \quad (2)$$

In particular, if our kernel k is real-valued, then we can discard the imaginary part of (2):

$$k(x-y) \approx \varphi(x)^T\varphi(y),\ \varphi(x) \equiv \frac{1}{\sqrt{M}}[\cos(\omega_1^T x), \ldots, \cos(\omega_M^T x), \sin(\omega_1^T x), \ldots, \sin(\omega_M^T x)]^T. \quad (3)$$

The great advantage of such an approximation is that we may now estimate a function in the RKHS as a linear operator in the random features: $f(x) = \sum_{i=1}^m \alpha_i K(x_i, x) \approx \sum_{i=1}^m \alpha_i \varphi(x_i)^T\varphi(x) = \psi^T\varphi(x)$, where $\psi \equiv \sum_{i=1}^m \alpha_i \varphi(x_i)$. Thus we may work directly in a primal space of $\varphi(x)$ and avoid computing large Gram matrices. To recap, using the approximation of kernels with random features works as follows: choose a kernel defined by k (with $k(0) = 1$), take its Fourier transform, $p(\omega)$, which will be a pdf over $\mathbb{R}^d$; draw M i.i.d. samples from $\rho(\omega)$, $\{\omega_j\}_{j=1}^M$; estimate the kernel with $K(x, y) \approx \varphi(x)^T\varphi(y)$ as in (3).

However, Bochner's theorem also allows one to work in the other direction. That is, we may start with a distribution $\mathcal{D}$ with pdf $\rho(\omega)$ and take the characteristic function (the inverse Fourier transformation) to define a shift-invariant kernel k. For example, suppose that $\rho(\omega) = \mathcal{N}(\omega|\mu, \Sigma)$, where $\mathcal{N}(\omega|\mu, \Sigma)$ is the pdf of $\mathcal{N}(\mu, \Sigma)$. Taking its characteristic function we see that $k(t) = \exp\left(\mathrm{i}\,\mu^T t - \frac{1}{2}t^T\Sigma t\right)$ would be the corresponding shift-invariant kernel. From the kernel learning perspective, Bochner's theorem yields an object to manipulate for the learning of one's kernel: $\rho(\omega)$ the distribution of random features.

We consider distributions that are mixtures of Gaussians:

$$\rho(\omega) = \sum_{\ell=1}^{L} \pi_\ell \mathcal{N}(\omega|\mu_\ell, \Sigma_\ell) \rightarrow$$
$$k(t) = \sum_{\ell=1}^{L} \pi_\ell \exp\left(\mathrm{i}\,\mu_\ell^T t - \tfrac{1}{2} t^T \Sigma_\ell t\right). \quad (4)$$

This makes for very general kernels; for a discussion on general properties of these kernels for finite L please see [Wilson and Adams, 2013]. In fact, i) noting that Gaussian mixture models are universal approximators of densities and may hence approximate any spectral distribution, and ii) using Plancherel's Theorem to relate spectral accuracies to the original domain [Silverman, 1986, Yang et al., 2015] it follows that:

Proposition 2.1. *The expression of $\rho(\omega)$ in (4) can approximate any shift invariant kernel.*

For our applications we only need real-valued kernels, hence we use the real part of (4):

$$K(x,y) = \sum_{\ell=1}^{L} \pi_\ell \exp\left(-\tfrac{1}{2}(x-y)^T \Sigma_\ell (x-y)\right) \cos\left(\mu_\ell^T (x-y)\right) \quad (5)$$
$$\approx \varphi(x)^T \varphi(y), \quad (6)$$

where $\varphi(x)$ is as in (3). An application of the random feature approximation bounds found in [Rahimi and Recht, 2007, Le et al., 2013, Sutherland and Schneider, 2015] yields that:

Proposition 2.2. *For compact $\mathcal{X} \subset \mathbb{R}^d$ with finite diameter, we have that*

$$\Pr\left[\sup_{x,y\in\mathcal{X}} |K(x,y) - \varphi(x)^T\varphi(y)| \geq \epsilon\right] = O\left(\frac{1}{\epsilon^2}\exp\left(\frac{-M\epsilon^2}{4(d+2)}\right)\right).$$

Using the above, it may be seen that one can effectively approximate shift-invariant kernels using random features drawn from Gaussian mixtures. However, in order to learn the kernel, one still needs a mechanism to determine the Gaussian mixture to use. We take a graphical model approach to determine the mixture for $\rho(\omega)$ in a principled, data-driven fashion.

2.2 Graphical Model

As described above, one may vary and tune kernels with the choice of density over random features, $\rho(\omega)$. Thus, in our model, we take this distribution itself to be a random latent parameter, in effect placing a prior over all stationary kernels, resulting in a strictly more general method than the traditional approach of using a fixed RBF kernel.

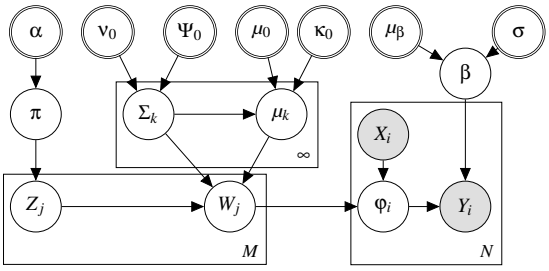


Figure 2: Plate diagram for the graphical model for BaNK learning framework.

Roughly speaking, the BaNK model will consist of three major parts: one, a prior for stochastically generating the random feature distribution $\rho(\omega)$; two, a prior for the generation of the parameters of a linear model in the primal space of random features; three, a generative data model with noise to generate labels given input covariates and the rest of the parameters. First, the spectral distribution $\rho(\omega)$ is generated. As previously mentioned, a robust and flexible choice of $\rho(\omega)$ is a Gaussian mixture model; Since the number of modes of $\rho(\omega)$ is not a priori known, we will assume it to be infinite ($\rho(\omega) = \sum_{k=1}^{\infty} \pi_k \mathcal{N}(\omega|\mu_k, \Sigma_k)$ where $\sum_k \pi_k = 1$ and $\pi_k > 0$), but given a finite dataset the model will realize only a finite number of Gaussians in the mixture. We use a Dirichlet process (DP) prior on the components of the Gaussian mixture (π).

The Dirichlet process is a distribution over discrete probability measures (i.e., atoms), $G = \sum_{k=1}^{\infty} \pi_k \delta_{\pi_k}$, with countably infinite support, where the finite-dimensional marginals are distributed according to a finite Dirichlet distribution [Ferguson, 1973]. We sample the mixture weights from a stick breaking prior, i.e. $\pi \sim GEM(\alpha)$ where GEM is the stick breaking prior [Sethuraman, 1994]. We also put a Normal-Inverse-Wishart prior on the mean μ_k and variance Σ_k of each of the Gaussian components.

Secondly, model parameters are generated. In Section 2.1 we discussed how functions in a kernel's RKHS can be approximated using a linear mapping in the random features. Thus, we consider models that operate linearly in the random features using a vector $\beta \in \mathbb{R}^{2M}$. As is standard in Bayesian regression and classification models [Bishop and Tipping, 2003], we generate β from a Normal prior, $\beta \sim \mathcal{N}(\mu_\beta, \sigma I)$.

Lastly, our observations are generated given a dataset $X := (x_1, \ldots, x_N)^T$ where each $x_i \in \mathbb{R}^d$. For example in regression tasks we have:

$$y = g(x) + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon I), \quad (7)$$

where g is $\beta^T \varphi(x)$ and $\varphi(x)$ is calculated using (3). Thus $y \sim \mathcal{N}(\beta^T \varphi(x), \sigma_\epsilon)$.

The complete generative model is given below and the corresponding plate diagram is shown in Figure 2.

1. Draw the mixture weights π over components of the kernel: $\pi \sim GEM(\alpha)$.
2. Draw the mixture components from Normal-Inverse-Wishart distribution. I.e. draw $\Sigma_k \sim \mathcal{W}^{-1}(\Psi_0, \nu_0)$, and $\mu_k \sim \mathcal{N}(\mu_0, \frac{1}{\kappa_0}\Sigma_k)$ for $k = 1, \dots \infty$.
3. For each random frequency index $j = 1, \dots, M$
   (a) Draw the component from which the frequency vector is drawn. $Z_j \sim Mult(\pi)$.
   (b) Draw the corresponding random frequency vector $W_j \sim \mathcal{N}(\omega|\mu_{Z_j}, \Sigma_{Z_j})$.
4. Draw the weight vector, $\beta \sim \mathcal{N}(\mu_\beta, \sigma I)$.
5. For each data point index $i = 1, \dots, N$
   (a) Define $\varphi(X_i)$ as in (3).
   (b) Draw the observation, e.g. for regression: $Y_i \sim \mathcal{N}(\varphi(X_i)^T\beta, \sigma_\epsilon I)$.

We note that the only change when going from regression to classification is in the step 5(b) of the generative procedure. This time we draw Y_i from a sigmoid.

5 For each data point index $i = 1, \dots, N$
   (b) Draw the output binary label $Y_i \sim \sigma(\varphi(X_i)^T\beta)$, where $\sigma(x) = \frac{1}{1+\exp(-x)}$.

3 Inference

We propose a MCMC based solution for inferring the parameters of the mixture of Gaussian distribution that defines $\rho(\omega)$. This includes finding the component assignment vector Z and the mean and covariance μ_k and Σ_k for each component. We will also sample the random frequencies W while marginalizing other parameters including π and β whenever possible. We will first describe the sampling equations for Z, μ_k, Σ_k, which remain the same for both regression and classification. Afterwards we describe inference for W which depends on the specific application.

We want to sample from $p(Z, \mu, \Sigma, W|X, Y, \text{rest})$, where rest are all the hyper-parameter of our model while other parameters including β and π have been integrated out. We use Gibbs sampling and sample each variable at a time given all other variables.

3.1 Sampling Z_j

Recall that Z_j indicates which component the random frequency W_j is drawn from. We use the Chinese restaurant process analogy to integrate out π, the component priors. Let $m_k \equiv \sum_l \delta(Z_l = k)$. The sampling equation for Z_j can be derived from [Neal, 1998] and is shown below

$$P(Z_j = k|\mu, \Sigma, W, X, Y, \text{rest}) = \quad (8)$$
$$\begin{cases} \frac{m_k^{-j}}{M-1+\alpha}\mathcal{N}(\omega_j|\mu_k, \Sigma_k) & m_k^{-j} > 0 \\ \frac{\alpha}{M-1+\alpha}\int_{\mu,\Sigma}\mathcal{N}(\omega_j|\mu, \Sigma)NIW(\mu, \Sigma)d\mu d\Sigma & m_k^{-j} = 0 \end{cases}$$

where $m_k^{-j} = \sum_{l:l\neq j}\delta(Z_l = k)$, $W_j = \omega_j$, $m_k^{-j} = 0$ corresponds to unseen mixture component and NIW is the Normal-Inverse-Wishart prior on mean and variance.

3.2 Sampling μ_k and Σ_k

Given the component assignment Z and the random frequencies W, the posterior distribution of the covariance of each Gaussian component in the mixture is Inverse-Wishart, ie $\Sigma_k \sim \mathcal{W}^{-1}(\Psi_k, \nu_k)$ where $\Psi_k = \Psi_0 + \sum_{j:Z_j=k}^{M}(W_j - \overline{W}^k)(W_j - \overline{W}^k)^T + \frac{\kappa_0 m_k}{\kappa_0+m_k}(\overline{W}^k - \mu_0)(\overline{W}^k - \mu_0)^T$, where $\overline{W}^k = \frac{1}{m_k}\sum_{j:Z_j=k} W_j$ and $\nu_k = \nu_0 + m_k$. Similarly, the posterior distribution of μ_k given, Σ_k, Z and W is a normal; i.e. $\mu_k \sim \mathcal{N}(\mu_k, \frac{1}{\kappa_k}\Sigma_k)$, where $\mu_k = \frac{\kappa_0\mu_0 + m_k\overline{W}^k}{\kappa_0+m_k}$ and $\kappa_k = \kappa_0 + m_k$. See [Gelman et al., 2003] for details.

3.3 Sampling W

We derive a Metropolis-Hasting (MH) sampler for sampling W. The posterior distribution of the random frequencies W given the assignment Z, the parameters of the component μ and Σ, and the data, X and Y is proportional to

$$P(W|Z, \mu, \Sigma, Y, X, \text{rest}) \propto P(W|Z, \mu, \Sigma)P(Y|X, W, \text{rest}). \quad (9)$$

The first term in the LHS is a normal distribution $P(W|Z, \mu, \Sigma) = \prod_j \mathcal{N}(W_j|\mu_{Z_j}, \Sigma_{Z_j})$. Since it is difficult to sample directly from the posterior, we use MH, where the first factor of the RHS of (9) is used as a proposal distribution; i.e. $Q(W) = P(W|Z, \mu, \Sigma)$. Now, the acceptance ratio for a newly proposed W^* is given by

$$r = \min\left\{1, \frac{P(Y|X, W^*, \text{rest})}{P(Y|X, W, \text{rest})}\right\}. \quad (10)$$

Here the second term on RHS of (10) is a ratio of model evidences and is calculated differently for regression and classification.

3.3.1 Regression

For regression we make use for conjugacy between prior of β, σ_ϵ and the likelihood to get a closed form solution for $P(Y|X, W, \text{rest})$. In this case we sample σ_ϵ

from Inverse − Gamma(a_0, b_0). The model evidence is then:

$$P(Y|X,W,\text{rest}) = \int_{\beta,\sigma_\epsilon} P(Y|W,X,\beta,\sigma_\epsilon)P(\beta)P(\sigma_\epsilon)d\beta d\sigma_\epsilon \propto \frac{\Gamma(a_n)}{\Gamma(a_0)}\frac{b_0^{a_0}}{b_n^{a_n}}\sqrt{\frac{|\Lambda_0|}{|\Lambda_n|}}, \quad (11)$$

where $\Lambda_0 = \frac{1}{\sigma_\beta^2}I$, $\Phi(X) = (\varphi(X_1)^T \dots \varphi(X_N)^T)^T$, $\Lambda_n = \Phi(X)^T\Phi(X) + \Lambda_0$, $\mu_n = \Lambda_n^{-1}(\Lambda_0\mu_\beta + \Phi(X)^T Y)$, $a_n = a_0 + \frac{n}{2}$ and $b_n = b_0 + \frac{1}{2}(Y^TY + \mu_0^T\Lambda_0\mu_0 - \mu_n^T\Lambda_n\mu_n)$. For more details refer to [Minka, 1999].

It is worth noting that one may efficiently compute ratio's of model evidences if proposing a single W_j at a time. That is, for each $j \in \{1,\dots,M\}$ we propose $W_j^* \sim \mathcal{N}(W_j|\mu_{Z_j}, \Sigma_{Z_j})$ and calculate an acceptance ratio of

$$r_j = \min\left\{1, \frac{P(Y|X,W_j^*,W_{-j},\text{rest})}{P(Y|X,W_j,W_{-j},\text{rest})}\right\} \quad (12)$$

where $W_{-j} = \{W_\ell\}_{\ell\neq j}$. This can be done efficiently because computing $P(Y|X,W_j^*,W_{-j},\text{rest})$ only requires low-rank updates on $\Phi(X)^T\Phi(X)$, allowing for fast Cholesky updates.

3.3.2 Classification

The aforementioned inference algorithm requires one to analytically obtain the model evidence of the data in terms of the model's random frequencies (eq: 10, 12). However, a lack of conjugacy may make it intractable to marginalize other parameters to obtain the model evidence. For instance, the Gaussian prior on β is not conjugate to a sigmoid. As a result it is difficult to directly compute the model evidence for a logistic regression model ie $P(Y|X,W,\text{rest})$ where β has been integrated out. Thus, for such situations where marginalization is intractable we must take a different approach to computing acceptance ratios for accepting random frequencies.

An approach one may take is to use a Laplace approximation to estimate the evidence as

$$\log(p(Y|X,W,\text{rest})) \approx \log(p(Y|X,W,\beta_{\text{MAP}},\text{rest})) + \log(p(\beta_{\text{MAP}})) + \frac{N}{2}\log(2\pi) - \frac{1}{2}\log(|A|) \quad (13)$$

where $\beta_{\text{MAP}} = \arg\min_\beta \log(P(Y|X,W,\beta,\text{rest})P(\beta))$

and $A = -\nabla^2 \log(P(Y|X,W,\beta,\text{rest})P(\beta))|_{\beta=\beta_{\text{MAP}}}$.

However, there are a few drawbacks to using a Laplace approximation in this manner. First, due to approximation, one is no longer sampling from the true posterior. Second, calculating β_{MAP} when a closed form solution is not available (as with logistic regression) requires solving a costly $2M$ dimensional optimization problem when computing acceptance ratios.

In order to address these drawbacks whilst still mixing well we jointly sample the jth random frequency W_j and the weight vector $\beta_j^{\cos}$ and $\beta_j^{\sin}$ corresponding to features $\cos(W_j^Tx)$ and $\sin(W_j^Tx)$ respectively. Specifically we sample from the joint distribution W_j and $\beta_j^\bullet = \{\beta^{\cos}, \beta^{\sin}\}$ given by:

$$P(W_j,\beta_j^\bullet|Z,\mu,\Sigma,Y,X,\text{rest}) \propto P(W_j|Z,\mu,\Sigma)P(\beta_j^\bullet|\text{rest})P(Y|X,W_j,\beta_j^\bullet). \quad (14)$$

However, samples from (14) are not readily available, so we use Metropolis Hastings with the proposal distribution being

$$Q = P(W_j|Z_j,\mu,\Sigma)\text{Lap}(\beta_j^\bullet|X,Y,W_j,\text{rest})$$

where $\text{Lap}(\beta_j^\bullet|X,Y,W_j,\text{rest})$ is the Laplace approximation of the posterior of $\beta_j^\bullet$, which requires only a 2-dimensional optimization:

$$\text{Lap}(\beta_j^\bullet|X,Y,W_j,\text{rest}) \approx P(\beta_j^\bullet|\text{rest})P(Y|X,W_j,\beta_j^\bullet). \quad (15)$$

Hence, acceptance ratio for jointly sampling a new $\{W_j^*,\beta_j^{\bullet *}\}$ can be calculated as

$$\min\left\{1, \frac{P(Y|X,W_j^*,\beta_j^{\bullet *},rest)P(\beta_j^{\bullet *})\text{Lap}(\beta_j^\bullet|X,Y,W_j)}{P(Y|X,W_j,\beta_j^\bullet,rest)P(\beta_j^\bullet)\text{Lap}(\beta_j^{\bullet *}|X,Y,W_j^*)}\right\}.$$

3.4 Runtime Complexity

We expound on the runtime complexity per iteration for the inference algorithms detailed above. Suppose that the L is the number of components considered, d is the data dimension, M is the number of frequencies and N is the number of data points. The runtime per iteration for sampling component parameters for both regression and classification is as follows: 1) sampling component parameters μ_ℓ's and Σ_ℓ's (and maintaining stats): $O(Ld^3)$; 2) sampling component assignments Z_j: $O(MLd^2)$.

For regression, sampling the random frequencies W using low rank update takes $O(M(d^2+dN+MN+M^2))$. Thus, the total runtime per iteration is $O(M^2d^2 + M^2N) = O(M^2N)$ for large datasets where $N \gg M > d$, and $M \geq L$.

For classification, sampling W_j's is $O\left(MN(d+\epsilon^{-}2)\right)$; where the ϵ^{-2} term arises from performing the 2-dimensional optimization required in (15) to ϵ precision [Cartis et al., 2010]. Treating ϵ as a constant, we have a total runtime of $O(MNd)$ for inference.

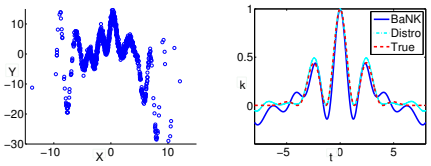
Hence, we see that inference is linear in N in either case, and so our method allows one to perform kernel learning in large datasets.

4 Experiments

We illustrate the use and performance of BaNK for both regression and classification on synthetic and real-world datasets below.

4.1 Synthetic Data

We give a simple 1-d kernel learning illustration with BaNK using synthetic data. We consider the shift-invariant kernel $k(t) = \exp\left(-\frac{1}{2(2^2)}t^2\right)(\frac{1}{2} + \frac{1}{2}\cos(\frac{3}{4}\pi t))$; that is, the kernel whose random frequency distribution is $\rho(\omega) = \frac{1}{2}\mathcal{N}(\omega|0, \frac{1}{2^2}) + \frac{1}{2}\mathcal{N}(\omega|\frac{3}{4}\pi, \frac{1}{2^2})$ (see Figure 3). We look to learn the underlying kernel using 250 frequencies. We generated $N = 1000$ instances $D = \{X_i, Y_i\}_{i=1}^N$ where $X_i \overset{iid}{\sim} \mathcal{N}(0, 4^2)$, $Y_i \sim \mathcal{N}(\varphi_\rho(X_i)^T\beta, 1)$, with φ_ρ being the random features from the kernel's true spectral distribution $\omega_j \overset{iid}{\sim} \rho$ and $\beta \sim \mathcal{N}(0, I)$. As explained above, using BaNK one may estimate ρ by drawing from the posterior. We plot one such draw in Figure 3(b). One can see that BaNK approximates the kernel rather well even though the underlying spectral distribution is multi-model, and the kernel is not easily decernable to the human eye based on the data plot (Figure 3(a)).



(a) Synthetic data (b) Kernel Estimate

Figure 3: (a) Synthetic data used. (b) True k in dashed red, k estimated with true spectral distribution ρ in cyan, and BaNK estimate in blue.

4.2 Regression

Below we run experiments with various real-world datasets found in the UCI machine learning repository (UCI MLR)[1]. We compare BaNK to a straightforward random feature approach with a fixed kernel as well as other competitive random feature based kernel learning methods. In particular we compare to the following methods:

RKS For this method we take input covariates to be random features $\varphi(x_i)$ as in (3). Here we take the random frequencies $\omega_j \overset{iid}{\sim} \mathcal{N}(0, \sigma^{-2}I)$. This corresponds to approximating the RBF kernel: $K(x_i, x_l) = \exp\left(-\frac{1}{2\sigma^2}\|x_i - x_l\|^2\right)$. Using these random features, we regress responses with ridge regression.

MKL One of the most widely used approaches to kernel-learning is multiple kernel learning (MKL) [Bach et al., 2004, Lanckriet et al., 2004]. Here, one attempts to learn a kernel using a non-negative linear combination of a fixed bank of kernels. That is, MKL attempts to learn a kernel K:

$$K(x_i, x_l) = \sum_{m=1}^{M} \alpha_m K_m(x_i, x_l), \text{ where } \alpha_m \geq 0, \qquad (16)$$

and $K_1, \ldots, K_M$ are predefined kernels. The kernel weights α_m would then be optimized according to one's loss. Note that (16) still requires the computation of a $N \times N$ Gram matrix, in fact, it requires M such Gram matrices. However, we extend MKL to use random features and scale to larger datasets. If $K_m(x_i, x_l) \approx \varphi_m(x_i)^T\varphi_m(x_l)$, then

$$K(x_i, x_l) \approx \sum_{m=1}^{M} \alpha_m \varphi_m(x_i)^T\varphi_m(x_l) = \bar{\varphi}(x_i)^T\bar{\varphi}(x_l), \quad (17)$$

where $\bar{\varphi}(x_i) = [\sqrt{\alpha_1}\varphi_1(x_i)^T, \ldots, \sqrt{\alpha_M}\varphi_M(x_i)^T]^T$. Hence, it is possible to work directly over input covariates of $\tilde{\varphi}(x_i) = [\varphi_1(x_i)^T, \ldots, \varphi_M(x_i)^T]^T$, the concatenation of the random features for each kernel $K_1, \ldots, K_M$. We take our bank of kernels to be Laplace, RBF, and Cauchy kernels at various scalings. As with RKS, we regress responses through ridge regression.

AlaC Very recently, independent work by Yang et al. [2015] has considered an optimization approach, called A la Carte, to learning a mixture of kernels. Here, an unconstrained, unpenalized, and non-convex GP likelihood problem is posed for regression and optimized over the parameters of a mixture model for random frequencies[2].

We perform 5-fold cross-validation (picking parameters on validation sets and reporting back the error on test sets). For AlaC we cross-validate the total number of mixture components and frequencies per components for datasets with fewer than 100K instances; for larger datasets we use the suggested hyper-parameters in [Yang et al., 2015]. The total number of random features was chosen to be 768 for RKS, MKL, and BaNK methods. For better interpretability, we standardized the output responses. In Table 1 we report the mean squared error (MSE) $\pm$ standard errors. One may see that BaNK performs better or as well as other methods on nearly all the datasets. Furthermore, it seems like BaNK is better able to leverage larger datasets. Lastly, we note that BaNK's sampling based approach with priors on mixture components seems more robust to local minima and over-fitting and has the ability to draw more frequencies from dominant components, which explains better performance w.r.t. Alac (e.g. for tom's dataset, Table 1).

4.3 Classification

As previously mentioned, we may use the BaNK framework to perform kernel learning in classification tasks. Below we illustrate the use of BaNK for classification and kernel learning on real-world datasets from

[1] `https://archive.ics.uci.edu/ml/index.html`

[2] Optimization was done using code provided by authors of [Yang et al., 2015].

Dataset	N	d	RKS	MKL	AlaC	BaNK
concrete	1030	8	0.1313 ± 0.0189	0.0942 ± 0.0100	$\mathbf{0.0682 \pm 0.0092}$*	0.1195 ± 0.0108
noise	1503	5	0.6974 ± 0.0244	$\mathbf{0.3217 \pm 0.0256}$*	$\mathbf{0.3395 \pm 0.0489}$	$\mathbf{0.3359 \pm 0.0354}$
prop	11934	16	$0.0006 \pm 3.6 \times 10^{-6}$	$4.2 \times 10^{-5} \pm 5.7 \times 10^{-6}$	0.0003 ± 0.0002	$\mathbf{8.9 \times 10^{-6} \pm 1.2 \times 10^{-6}}$*
bike	17379	12	0.1832 ± 0.0049	0.1509 ± 0.0057	$\mathbf{0.0467 \pm 0.0016}$*	$\mathbf{0.0496 \pm 0.0022}$
tom's	28179	96	0.0479 ± 0.0109	0.0891 ± 0.0100	0.6583 ± 0.0413	$\mathbf{0.0083 \pm 0.0018}$*
cte	53500	386	0.0886 ± 0.0014	0.0442 ± 0.0003	0.6223 ± 0.0053	$\mathbf{0.0101 \pm 0.0003}$*
music	515345	90	0.8333 ± 0.0028	0.8524 ± 0.0029	0.7318 ± 0.0705	$\mathbf{0.7042 \pm 0.0038}$*
twitter	583250	77	0.3837 ± 0.0397	0.4572 ± 0.0175	0.2223 ± 0.0358	$\mathbf{0.0981 \pm 0.0211}$*

Table 1: Regression MSE on UCI MLR. Asterisks denote the lowest MSE per dataset, methods in bold text were not found to be statistically different from the lowest MSE using a paired t-test with p-value < 0.05.

the UCI MLR. We compare the accuracies BaNK models achieve to traditional scalable kernel methods for classification; namely, we consider using the aforementioned RKS and MKL random features in a logistic model.

Furthermore, we also compare to an optimization approach based on AlaC [Yang et al., 2015], which we term random frequency optimization (RFO). Although it is possible to write and differentiate a data likelihood solely in terms of spectral density parameters (through the kernel they induce) for GP regression, a lack of conjugacy with a logistic likelihood and Gaussian priors requires approximate inference for classification. Thus, directly applying the approach of Yang et al. [2015] will be troublesome. To mitigate this difficulty, we jointly optimize a logistic loss both in terms of linear weights β and spectral parameters $\{\nu, M, \mu\}$.

Specifically, we minimize the following problem:

$$
\begin{aligned}
&-\sum_{i=1}^{N} Y_i \left\{ \sum_{k=1}^{K} \nu_k^2 \sum_{j=1}^{D} \left(\beta_{kj}^{\cos} \cos\left(\zeta_{ijk}\right) + \beta_{kj}^{\sin} \sin\left(\zeta_{ijk}\right) \right) + \beta_0 \right\} \\
&+ \log\left[1 + \exp\left\{ \sum_{k=1}^{K} \nu_k^2 \sum_{j=1}^{D} \left(\beta_{kj}^{\cos} \cos\left(\zeta_{ijk}\right) + \beta_{kj}^{\sin} \sin\left(\zeta_{ijk}\right) \right) + \beta_0 \right\} \right] \\
&+ \frac{\lambda}{2} \left(\|\beta^{\cos}\|^2 + \|\beta^{\sin}\|^2 + \beta_0^2 \right),
\end{aligned}
$$

where $\zeta_{ijk} = X_i^T M_k w_{kj} + X_i^T \mu_k$ and $\beta^{\cos} \in \mathbb{R}^{KD}$, $\beta^{\sin} \in \mathbb{R}^{KD}$, $\beta_0 \in \mathbb{R}$, $\nu \in \mathbb{R}^K$, $M_k \in \mathbb{R}^{d \times d}$, $\mu_k \in \mathbb{R}^d$ are optimized; $w_{kj} \in \mathbb{R}^d$ are standard Gaussian vectors that are drawn before optimizing and held fixed.

We again performed 5 fold cross validation and report the mean prediction error on test sets in Table 2. The results in Table 2 show that BaNK consistently performed better or as well as the baselines.

4.4 Timing Experiments

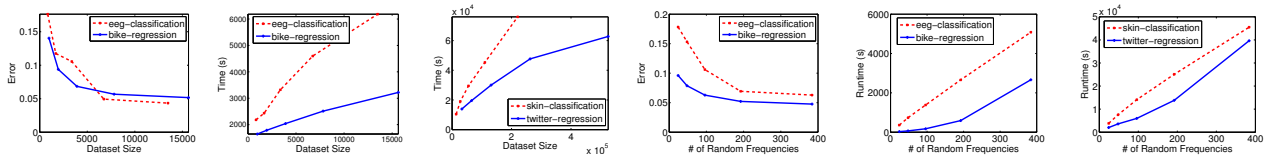
We empirically investigate the linear scaling of the BaNK method in terms of the number of instances N. It is this linear dependence that allows BaNK to perform kernel learning in large datasets, where naive kernel methods are generally $\Omega(N^2)$. We hold the number of random frequencies fixed at $M = 384$ and vary N to study the empirical dependence of runtime on dataset size for BaNK. We see the linear scaling is empirically verified in Figures 4(b) and 4(c). Also, we observe lower test errors as dataset sizes increase indicating that BaNK is able to leverage more data to learn effective kernels and increase accuracy (Figure 4(a)).

Furthermore, we illustrate the scaling of our BaNK method for regression and classification in terms of the number of random frequencies, M. As discussed in Section 3, the run-time complexity in large datasets will be $O(NM^2)$ for regression and $O(MNd)$ for classification. We empirically study the dependence of M on runtimes by varying $M \in \{24,\ 48,\ 96,\ 192,\ 384\}$ and recording the runtime of BaNK using fixed datasets. Figures 4(f) and 4(e) show a quadratic growth in runtime for regression and a linear growth for classification. We see similar trends on other datasets. Moreover, we observe in Figure 4(d) an increase in performance with diminishing returns as M increases. Thus, we see that more frequencies aid kernel approximation and accuracy, but performance stabilizes after enough frequencies are chosen.

Lastly, we record the runtimes on each dataset (Figure 5) with the total number of random frequencies fixed at $M = 384$ for all methods. While restricted methods that consider only a fixed set of random frequencies (RKS and MKL) perform fast, we see that BaNK's

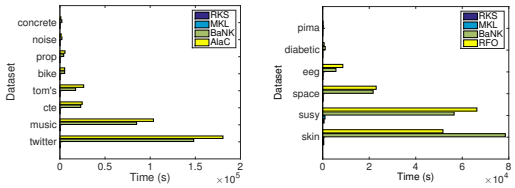
Dataset	N	d	RKS	MKL	RFO	BaNK
pima	768	8	0.332 ± 0.0201	0.4455 ± 0.0533	$\mathbf{0.2592 \pm 0.0214}$*	$\mathbf{0.263 \pm 0.0111}$
diabetic	1151	20	0.3312 ± 0.0083	0.3051 ± 0.0004	0.4263 ± 0.0007	$\mathbf{0.279 \pm 0.0022}$*
eeg	14980	15	0.0706 ± 0.0019	$\mathbf{0.0544 \pm 0.0021}$*	0.2411 ± 0.1123	$\mathbf{0.0686 \pm 0.0012}$
space	58000	9	0.0022 ± 0.0004	$\mathbf{0.0015 \pm 0.0001}$	0.0018 ± 0.00007	$\mathbf{0.0009 \pm 0.0001}$*
susy	100000	18	0.2009 ± 0.0002	0.201 ± 0.0001	0.2089 ± 0.00026	$\mathbf{0.2005 \pm 0.0001}$*
skin	245053	3	0.1135 ± 0.113	0.2737 ± 0.105	0.0043 ± 0.001	$\mathbf{0.0004 \pm 0.0001}$*

Table 2: Classification Prediction Error on UCI MLR. Asterisks denote the lowest error per dataset, while the methods in bold were not found to be statistically different from the lowest error using a McNemar's test [Fagerland et al., 2013] with p-value < 0.05.



(a) N vs Error (b) N vs Time (c) N vs Time (d) M vs Error (e) M vs Time (f) M vs Time

Figure 4: Runtime experiments. In all figures we denote classification curves with a dashed red line and regression curves with a solid blue line. Dataset names are shown in legends. Errors are prediction errors for classification tasks and MSE for regression tasks. Figure (a) shows how error changes with number of instances N; (b,c) shows the effect of increase in number of instances on runtime; (d) shows how error changes with number of random frequencies M; (e,f) show the effect of number of frequencies on computational time.



(a) Regression (b) Classification

Figure 5: Runtimes for datasets.

runtime is comparable to other methods that learn the random frequencies (AlaC and RFO) and can scale to large datasets.

5 Related Work

Given the poor scaling of kernel methods on datasets with many instances, several methods have looked at approximations of kernels that bypass the computation of large Gram matrices. For example, Nyström based methods [Drineas and Mahoney, 2005] look to give a fast to compute, low-rank approximation of the Gram matrix. Other approaches for summarizing the Gram matrix by the removal of elements or rows have been explored in [Scholkopf et al., 2002, Blum, 2006, Frieze et al., 2004, Shen et al., 2006]. Furthermore, approximations based on KD-trees have been explored in [Shen et al., 2006]. In this paper, we work with kernel approximations based on random features, called Random Kitchen Sinks [Rahimi and Recht, 2007, 2009, Le et al., 2013]. As previously discussed, random feature approaches work using an empirical estimate of kernels that stem by drawing features from the Fourier transform of positive definite functions, which will be a distribution if properly scaled.

Kernel learning methods have also received attention due to the impact that kernel choice has on performance. Indeed, even if one fixes a family of kernels to use (e.g. RBF kernels) one still has to select the parameters of the kernels. This is often done with cross-validation or with methods like [Keerthi et al., 2007, Chapelle et al., 2002]. A popular approach to learning kernels in a more flexible class is multiple kernel learning (MKL) [Bach et al., 2004, Lanckriet et al., 2004]. As illustrated is Section 4.2, MKL learns a kernel that is the non-negative linear combination of a bank of fixed kernels. However, naively applying MKL approaches would still require the computation of several large Gram matrices; instead, as previously discussed, one may combine random features to perform scalable MKL (see Lu et al. [2014] for an application of such ideas). Very recently, independent work [Yang et al., 2015] has explored an optimization approach, A la carte, to learning parameters of mixture of kernels, where one optimizes the parameters generating random features in non-convex likelihood problems. See also [Băzăvan et al., 2012] for another optimization approach. Unfortunately, due to the non-convex nature of the optimization problem being optimized, such approaches yield non-interpretable kernels, whereas BaNK yield draws from a well defined posterior. Wilson and Adams [2013] considers kernels as in (4), but without inference over L, the number of mixture components. Wilson [2012] mentions the possibility of putting a DP prior model on parameters, but does so without empirical details.

6 Conclusion

In this paper we propose an efficient and general data-driven framework, BaNK, for learning of kernels that scales to large datasets. By representing the spectral density using a non-parametric mixture of Gaussians, we capture a large class of kernels that can be learned. We provide a generative model for learning kernels while performing regression and classification tasks, and propose novel MCMC based sampling schemes to infer parameters of the mixtures. We show that our proposed framework outperforms other scalable kernel learning methods on a variety of real world datasets in both classification and regression task.

Acknowledgements This work is supported in part by NSF IIS-1247658, NIH GWAS: R01GM087694, DOE DE-SC0011114 grants and an IBM Fellowship.

References

F. R. Bach, G. R. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6. ACM, 2004.

E. G. Băzăvan, F. Li, and C. Sminchisescu. Fourier kernel learning. In *Computer Vision–ECCV 2012*, pages 459–473. Springer, 2012.

C. M. Bishop and M. E. Tipping. Bayesian Regression and Classification. *IOS Press*, 2003.

A. Blum. Random projection, margins, kernels, and feature-selection. In *Subspace, Latent Structure and Feature Selection*, pages 52–68. Springer, 2006.

C. Cartis, N. I. Gould, and P. L. Toint. On the complexity of steepest descent, newton's and regularized newton's methods for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 20(6): 2833–2852, 2010.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine learning*, 46(1-3):131–159, 2002.

P. Drineas and M. W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research*, 6:2153–2175, 2005.

M. W. Fagerland, S. Lydersen, and P. Laake. The mcnemar test for binary matched-pairs data: mid-p and asymptotic are better than exact conditional. *BMC Medical Research Methodology*, 2013.

T. S. Ferguson. A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics*, 1(2):209–230, Mar. 1973.

A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2003. ISBN 158488388X.

S. Keerthi, V. Sindhwani, and O. Chapelle. An efficient method for gradient-based adaptation of hyperparameters in svm models. 2007.

G. R. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72, 2004.

Q. Le, T. Sarlos, and A. Smola. Fastfood–approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, 2013.

Z. Lu, A. May, K. Liu, A. B. Garakani, D. Guo, A. Bellet, L. Fan, M. Collins, B. Kingsbury, M. Picheny, et al. How to scale up kernel methods to be as good as deep neural nets. *arXiv preprint arXiv:1411.4000*, 2014.

T. P. Minka. Bayesian linear regression. Technical report, 3594 Security Ticket Control, 1999.

R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. Technical Report 9815, Dept. of Statistics, University of Toronto, 1998.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2007.

A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in neural information processing systems*, pages 1313–1320, 2009.

W. Rudin. *Fourier analysis on groups*. Number 12. John Wiley and Sons, 1990.

D. Scholkopf, F. Achlioptas, and M. Bernhard. Sampling techniques for kernel methods. In *Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Conference*, volume 1, page 335. MIT Press, 2002.

J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

Y. Shen, A. Ng, and M. Seeger. Fast gaussian process regression using kd-trees. In *Proceedings of the 19th Annual Conference on Neural Information Processing Systems*, number EPFL-CONF-161316, 2006.

B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

D. Sutherland and J. Schneider. On the error of random fourier features. In *AISTATS*, 2015.

A. G. Wilson. A process over all stationary covariance kernels. Technical report, 2012.

A. G. Wilson and R. P. Adams. Gaussian process kernels for pattern discovery and extrapolation. *ICML*, 2013.

Z. Yang, A. J. Smola, L. Song, and A. G. Wilson. A la carte-learning fast kernels. In *AISTATS*, 2015.