Learning with Kernels at Scale and Applications in Generative Modeling

Wei-Cheng Chang

CMU-LTI-20-008

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15123
www.lti.cs.cmu.edu

Thesis Committee:

Yiming Yang (Chair)	Carnegie Mellon University
Barnabás Póczos	Carnegie Mellon University
Jeff Schneider	Carnegie Mellon University
Sanjiv Kumar	Google Research NYC

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Language and Information Technologies

Keywords: deep kernel learning, deep generative modeling, kernelized discrepancies, Stein variational gradient descent, large-scale kernels

To all of the people that light up my life.

Abstract

Kernel methods are versatile in machine learning and statistics. For instance, Kernel two-sample test induces Maximum Mean Discrepancy (MMD) to compare two distributions and serves as a distance metric for learning implicit generative models (IGMs). Kernel goodness-of-fit test, as another example, induces Kernel Stein Discrepancy (KSD) to measure model-data discrepancy and connects to a variational inference procedure for explicit generative models (EGMs). Other extensions include time series modeling, graph-based learning, and more. Despite their ubiquity, kernel methods often suffer from two fundamental limitations: the difficulty in kernel selection for complex downstream tasks and the tractability of large-scale problems. This thesis addresses both challenges in several complementary aspects.

In part I, we tackle the issue of kernel selection in learning implicit generative models (IGMs) with kernel MMD. Conventional methods using a fixed kernel MMD have limited success on high-dimensional complex distributions. We propose to optimize MMD with neural-parametrized kernels, which is more expressive and improves the state-of-the-art results on high-dimensional distributions (Chapter 2). We also formulate kernel selection problems as learning kernel spectral distributions, and enrich the spectral distributions by learning IGMs to draw samples from it (Chapter 3).

In part II, we aim at learning suitable kernels for Stein variational inference descent (SVGD) in explicit generative modeling (EGMs). Although SVGD with fixed kernels shows encouraging performance in low-dimensional (within hundreds) Bayesian inference tasks, its success with high-dimensional problems such as image generation is limited. We propose the noise-conditional kernel SVGD (NCK-SVGD) for adaptive kernel learning, which is the first SVGD variant successfully scaled up for distributions with the dimension of several thousands, and performs competitively as state-of-the-art IGMs. With a novel entropy regularizer, NCK-SVGD enjoys flexible control between sample diversity and quality (Chapter 4).

In part III, we address the kernel tractability challenge with variants of random Fourier features (RFF). We propose to learn non-uniformly weighted RFF, which performs as good as the uniformly-weighted RFF while demanding less memory (Chapter 5). For the kernel contextual bandit problem, we reduce the computational cost of the kernel UCB algorithm by using RFF to approximate the predictive mean and confidence estimate (Chapter 6).

In part IV, We extents kernel learning for time series modeling and graph-based learning. For change-point detection over time series, we optimize the kernel two-sample test via auxiliary generative models, acting as surrogate samplers of unknown anomaly distributions (Chapter 7). For graph-based transfer learning, we construct the graph Laplacian by kernel diffusion and leverage label propagation to transfer knowledge from source to target domains (Chapter 8).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations and Background

Kernel methods are ubiquitous in machine learning research (Scholkopf and Smola, 2001; Hofmann et al., 2008) with a broad range of topics, including Support Vector Machines for classification and regression problems (Cortes and Vapnik, 1995; Drucker et al., 1997), non-linear component analysis for dimensionality reduction (Schölkopf et al., 1998; Bach and Jordan, 2002), Gaussian Processes for sequential modeling and extrapolations (Williams and Rasmussen, 2006), large-scale kernel approximation techniques (Williams and Seeger, 2001; Rahimi and Recht, 2008), kernel learning based on structured data such as trees and graphs (Vishwanathan et al., 2004, 2010), and more.

A newly emerging topic in kernel learning is on kernelized discrepancy analysis for generative models. That is, for optimizing a generative model, we need to measure the discrepancy between the true-data distribution (denoted as $\mathbb{P}$) and the system-produced distribution (denoted as $\mathbb{Q}$), and to develop tractable algorithms for minimizing the discrepancy. Kernel methods provide a principled way to specify the desirable properties of the objective functions and induce analytical forms for solving the optimization problems (Gretton et al., 2012a; Chwialkowski et al., 2016; Liu et al., 2016).

A fundamental question we need to answer is about how to define or measure the discrepancy between two distractions. Existing work on this aspect can be divided into two important classes of kernelized discrepancies, as characterized below.

- **Maximum Mean Discrepancy (MMD)**: Given two easy-to-sample distributions $\mathbb{P}$ and $\mathbb{Q}$, MMD is a *data-to-data* metric that defines the discrepancy $M_k(\mathbb{P}, \mathbb{Q})$ over the samples from both $\mathbb{P}$ and $\mathbb{Q}$, respectively. In other words, no explicit density modeling about $\mathbb{P}$ nor $\mathbb{Q}$ is required. MMD has been found useful in kernelized two-sample hypothesis testing about two distributions as an application (Gretton et al., 2012a).

- **Kernel Stein Discrepancy (KSD)**: Given an easy-to-sample distribution $\mathbb{P}$ and an explicit density model $\mathbb{Q}$, KSD is a *data-to-model* metric that defines the discrepancy $S_k(\mathbb{P}, \mathbb{Q})$ via the samples from $\mathbb{P}$ and the gradients of the log-density of $\mathbb{Q}$. In other words, KSD avoids the sampling process (often expensive) from an explicit density model. KSD has been found useful for kernelized goodness-of-fit test as an application (Chwialkowski et al., 2016; Liu et al., 2016).

In comparison, the main difference between MMD and KSD is that the former only requires samples from the model distribution $\mathbb{Q}$ instead of explicit density information while the latter requires explicit density modeling of distribution $\mathbb{Q}$ without the need for sampling. The complementary perspectives of MMD and KSD lead to different usages of them in two categories of generative models (Diggle and Gratton, 1984), as introduced below.

- **Implicit Generative Models (IGMs)**: IGMs model the *sampling* process that draw novel samples from underlying data distributions. The most well-known example is Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), and recent variants are capable to generate high fidelity samples from high-dimensional complex real-world datasets (Brock et al., 2019; Karras et al., 2019). Notably, IGMs is implicit as it does not define explicit densities of model distributions.

- **Explicit Generative Models (EGMs)**: EGMs model *explicit* densities of underlying data distributions. EGMs can be either normalized or un-normalized, where the former often has limited function classes such as normalization flow (Dinh et al., 2016), and the later often has powerful expressive power such as deep energy models (Du and Mordatch, 2019). While un-normalized EGMs learn robust features for many downstream tasks (Grathwohl et al., 2020), high-dimensional inference of un-normalized EGMs is expensive when using MCMC sampling (Welling and Teh, 2011).

As IGMs are sampling-based methods, MMD is a natural choice of metric for evaluating the discrepancy between $\mathbb{P}$ and $\mathbb{Q}$ by comparing the generated samples from $\mathbb{Q}$ against real samples from $\mathbb{P}$. On the other hand, as EGMs are based on explicit density modeling of $\mathbb{Q}$, KSD is a natural choice of metric for evaluating the discrepancy, eluding the expensive MCMC sampling from un-normalized density models when the dimensionality is high.

KSD also connects to a kernelized variational inference algorithm, namely Stein Variational Gradient Descent (SVGD) (Liu and Wang, 2016). SVGD iteratively transforms samples to approximate the target EGMs, with gradient updates that optimally decrease the KL divergence, and the gradient flow of KL amounts to the KSD (Liu, 2017). SVGD has shown encouraging results in several low-dimensional Bayesian inference tasks (Feng et al., 2017; Gong et al., 2019), but how to make it successful in high-dimensional spaces is an open challenge.

1.2 Challenges

Although kernelized discrepancies offer principled perspectives for learning of IGMs and sampling of EGMs, the performance of kernel-based algorithms in downstream tasks are often hinged on the choice of kernels (e.g., which kernel families to use and what kernel hyper-parameters to set). This issue is commonly known as the *kernel selection* problem (a.k.a., kernel learning or kernel optimization). Kernel selection for traditional supervised learning settings (e.g., classification with SVMs and regression with Gaussian Processes) has been studied (Rakotomamonjy et al., 2008; Gönen and Alpaydın, 2011; Wilson and Adams, 2013). Nonetheless, how to successfully apply kernelized discrepancies to modern deep generative modeling problems (e.g., high-dimensional computer vision datasets) is still under explored. In the scope of this thesis, we aim at providing solutions or insights to the following open research questions:

1. What kernel function classes are suitable for learning *high-dimensional* IGMs?
2. What kernel function classes are suitable for drawing samples from *high-dimensional* EGMs?
3. How to optimize kernel two-sample test when samples from one distribution are *extremely sparse*?
4. How to construct an unified kernel when the cross-domain graphs have *very sparse links*?

Besides tackling the kernel learning challenge for higher dimensional applications and sample-sparsity problems, kernel-based algorithms are often suffered from the tractability in the large number of instances regime. For example, consider solving a linear regression of n instances $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. The time complexity is $O(nd^2)$ and the memory cost is $O(nd)$. Its kernel-based counterpart, however, requires solving a linear system which takes $O(n^3)$ time and $O(n^2)$ memory usage. Such complexity is far from desirable in big-data applications. To overcome the kernel scalability issue, several kernel approximation techniques have been proposed, and random features (Rahimi and Recht, 2008, 2009) is arguably one of the most elegant and impactful approach. For shift-invariant kernels, the kernel function can be approximated by an inner product of two random feature maps, hence greatly reducing time/space complexity to the linear case. In this thesis, we seek to further improve random features by asking two research questions:

1. Can we derive non-uniformly weighted random features that requires less samples (hence less memory) while maintaining a similar kernel approximation error?
2. Can we leverage random features to speed-up the kernel contextual bandit algorithms, which solves sequential decision making problems that heavily rely on the kernel matrix computation?

1.3 Thesis Structure and Contributions

Thesis Statement This thesis aims to advance kernel-based algorithms in two complementary aspects: the expressiveness of kernel functions for modern complex applications and the tractability of kernel methods in the large-scale setting (e.g., large data and high-dimensional). The contributions of this thesis can be summarized into three folds. (1) We propose generic frameworks that optimize deep kernels for deep generative modeling, including both IGMs and EGMs families; (2) We present non-uniform random Fourier features for large-scale kernel approximations which enjoys less memory usage and study applications in kernel contextual bandit problems; (3) We extend kernel learning to two novel applications with structured information such as time series change-point detection and graph-based transfer learning.

We outline the thesis structure and contributions of the individual chapters below.

Part I: Learning Kernels for Implicit Generative Modeling. We tackle the kernel selection challenge when using kernel MMD to learning IGMs. Dziugaite et al. (2015); Li et al. (2015b) are the first pioneering works that train IGMs by minimizing the kernel MMD, without any kernel selection procedure. Nevertheless, due to the restricted power of simple Gaussian kernels and the sampling variance of empirical estimated MMD, they only achieve limited success on low-dimensional datasets (several hundreds at most) with simple structure such as MNIST, but fail on high-dimensional distributions (several thousands and higher) with complex objects such as CIFAR-10, CelebA, and more.

- In Chapter 2, we revisit the kernel two-sample test and introduce its connection with learning IGMs. We first point out the deficiency of using simple kernel MMD (e.g., pre-defined Gaussian kernels) when training IGMs on high-dimensional data distributions. We then propose to enrich the function class of MMD via deep neural-parameterized kernels, which also leads to a min-max objective where the maximization is regarded as the kernel selection procedure, hence the name *MMD GAN*. From the perspective of Integral Probability Metric, we connect MMD GAN with many GAN frameworks such as state-of-the-art Wasserstein GAN (Arjovsky et al., 2017; Gulrajani et al., 2017). Quantitative speaking, MMD GAN significantly outperforms its predecessor that did not conduct kernel learning, and also consistently improves upon state-of-the-art W-GAN on conventional image generation metric such as Inception score and FID score.

  The content of Chapter 2 appears in (Li et al., 2017):
  Chun-Liang Li*, **Wei-Cheng Chang***, Yu Cheng, Yiming Yang, and Barnabás Póczos.
  MMD GAN: Towards deeper understanding of moment matching network. In NeurIPS, 2017.
  Code available at: `https://github.com/OctoberChang/MMD-GAN`

- In Chapter 3, we take a further step to optimize shift-invariant kernels via the lens of kernel spectral distribution. Existing kernel learning approaches (Gönen and Alpaydın, 2011; Wilson and Adams, 2013) explicitly model the spectral density with simple exponential families (e.g., mixture of Gaussians), often resulting in limited capacity of kernels. We model the sampling process of kernel spectral distributions via learning IGMs, which induces an implicit kernel learning (IKL) framework by constructing kernel evaluations using the Fourier samples draw from the IGMs. The IGMs can be optimized by back-propagation from task-dependent kernel learning objectives. We demonstrate the success of IKL framework on training generative models (MMD-GAN IKL) for the unsupervised image/text generation as well as learning more discriminative kernels for classification.

  The content of Chapter 3 appears in (Li et al., 2019):
  Chun-Liang Li, **Wei-Cheng Chang**, Youssef Mroueh, Yiming Yang, and Barnabás Póczos.
  Implicit kernel learning. In AISTATS, 2019.
  Code available at: `https://github.com/OctoberChang/kernel_rff_learn`

Part: II: Learning Kernels for Explicit Generative Modeling. We aim at learning suitable kernels for Stein Variation Gradient Descent (SVGD) to draw samples from score-based EGMs. Induced from the Kernel Stein Discrepancy, Stein Variational Gradient Descent (SVGD) (Liu and Wang, 2016; Liu, 2017) is a deterministic sampling algorithm that iteratively transports a set of particles to approximate a given distribution, based on functional gradient descent in RKHS. SVGD with simple pre-determined kernels has shown promising results on several low-dimensional (within hundreds) Bayesian inference tasks. However, how to make its inference effective and scalable for complex high-dimensional distributions (several thousands and more) is an open question that have not been studied in sufficient depth.

- In Chapter 4, we first point out the challenge of high-dimension inference is to deal with multi-modal distributions with many low-density regions, where SVGD can fail even on simple Gaussian mixtures. To this end, we propose noise-conditional kernel SVGD (NCK-SVGD), where the kernels are conditionally learned or selected based on the multi-scale noise-perturbed data distributions. With an additional entropy regularization term in the objective, NCK-SVGD enjoys flexible control between sample diversity and quality. Quantitatively speaking, NCK-SVGD achieves a new state-of-the-art FID score of 21.95 on CIFAR-10 among the score-based EGMs, and is comparable to recent GAN-based models in the IGM family.

  The content of Chapter 4 appears in (Chang et al., 2020a):
  Wei-Cheng Chang, Chun-Liang Li, Youssef Mroueh, and Yiming Yang.
  Kernel stein generative modeling. In submission to NeurIPS 2020 (arXiv:2007.03074).

Part: III: Large-scale Kernel Approximation. We address the kernel scalability challenge (regarding number of instances) by studying variants of random Fourier features (RFF) (Rahimi and Recht, 2008). The value of shift invariant kernels can be approximated by an inner product of two Fourier feature maps, where the Fourier bases are equally-weighted samples draw from the kernel spectral distributions. RFF has received great attention in the past decade due to its mathematically elegant form and decent performance on certain classification/regression benchmarks. Recently, many advanced variants (Yang et al., 2014; Sinha and Duchi, 2016; Yu et al., 2016; Bach, 2017) have been proposed to improve RFF under different contexts and settings.

- In Chapter 5, we propose to use unequally-weighted Fourier bases to approximate the kernel integral, backed by a novel bias-variance trade-off insight. While seemingly counter-intuitive, James-Stein effect (Stein, 1956; Muandet et al., 2014) suggests a family of biased estimators could result in lower risk than the Monte Carlo estimate. Thus, we consider learning the data-driven reweighing scheme on RFF via optimizing the kernel approximation error. By lowering variance with slighted biased estimates, data-driven RFF enjoys better kernel approximation when number of RFF is small, which results in less memory demand for large-scale applications.

  The content of Chapter 5 appears in (Chang et al., 2017a):
  Wei-Cheng Chang, Chun-Liang Li, Yiming Yang, and Barnabás Póczos.
  Data-driven random Fourier features using stein effect. In IJCAI, 2017.
  Code available at : `https://github.com/OctoberChang/stein_rff`

- In Chapter 6 we aim to improve the scalability of kernel contextual bandit algorithms, in particular in the regime of upper-confident-bound (UCB) framework. While the kernel UCB enjoys provably non-regret guarantee, its computational cost is expensive because of iteratively updating the inverse of the kernel matrix (i.e., $O(KT^3)$ up to T-trial where K is number of arm). To this end, we propose using RFF to approximate the posterior mean and covariance estimate of the Kernel UCB algorithms. Empirically, we find that the number of RFF at the rate of $O(\text{polylog}(t))$ is sufficient to have bounded kernel approximate error of similar regret, and enjoys order of magnitude (e.g., 10x) faster computation time compared to the exact kernel UCB algorithm.

  The content of Chapter 6 appears in (Chang et al., 2019b):
  Wei-Cheng Chang, Rajat Sen, Yiming Yang, and Sanjay Shakkottai.
  Fast kernel contextual bandits with random Fourier features. Technical Report, 2019.
  Code available at: https://github.com/OctoberChang/rff_ucb

Part: IV: Extensions to Time-Series Modeling and Graph-based Learning. Kernel methods also found applications with structured data information such as time series and graph. In particular, we are interested in time series change-point detection (CPD) problems, where anomalies occur suggesting a distribution shift from the history to current time window. We also look at graph-based transfer learning problems where graph Laplacians can be constructed by different kernel families.

- In Chapter 7, we formulate the time series change-point detection as finding distribution shift problems using kernel two-sample test. We first demonstrate conventional kernel selection strategies for kernel two-sample test end up sub-optimal performance when only very sparse (or even no prior) samples available from anomaly distributions. Thus, we propose to optimize a test power lower bound via an auxiliary generative model mimicking the sampling process from the anomaly distribution. With deep kernel parameterization, the proposed kernel learning framework can empirically detects different types of change-points on both synthetic datasets and real-world applications.

  The content of Chapter 7 appears in (Chang et al., 2019a):
  Wei-Cheng Chang, Chun-Liang Li, Yiming Yang, and Barnabás Póczos.
  Kernel change-point detection with auxiliary deep generative models. In ICLR, 2019.
  Code available at: https://github.com/OctoberChang/klcpd_code

- In Chapter 8, we focus on the setting of transductive transfer learning where source domains and target domains have heterogeneous feature space (e.g., English v.s. other languages). One key challenge of graph-based label propagation methods emerges: how to compute kernel value between data from different domains if they are not in the same feature space? We propose diffusion kernel completion to construct an unified graph Laplacian across domains and leverage semi-supervised label propagation to transfer knowledge from the source to the target domain.

  The content of Chapter 8 appears in (Chang et al., 2017b):
  Wei-Cheng Chang*, Yuexin Wu*, Hanxiao Liu, and Yiming Yang.
  Cross-domain kernel induction for transfer learning. In AAAI, 2017.
  Code available at: https://github.com/OctoberChang/KerTL

Additional Contributions Apart form the kernel-based algorithms for generative models and structured data, I have also worked on different topics during course of PhD study as listed below.

Large-scale multi-label text classification (Chang et al., 2020c, 2018; Liu et al., 2017)

Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit Dhillon.
Taming pretrained transformers for extreme multi-label text classification. In KDD, 2020.
Code available at: https://github.com/OctoberChang/X-Transformer

Wei-Cheng Chang, Hsiang-Fu Yu, Inderjit Dhillon, and Yiming Yang.
SeCSeq: Semantic coding for sequence-to-sequence based extreme multi-label classification.
In NeurIPS CDNNRIA Workshop, 2018.
Code available at: https://github.com/OctoberChang/SeCSeqXMC

Jingzhou Liu, **Wei-Cheng Chang**, Yuexin Wu, and Yiming Yang.
Deep learning for extreme multi-label text classification. In SIGIR, 2017.

Pre-training tasks for large-scale retrieval (Chang et al., 2020b)

Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar.
Pre-training tasks for embedding-based large-scale retrieval. In ICLR, 2020.

Time series modeling with deep learning (Lai et al., 2018)

Guokun Lai, **Wei-Cheng Chang**, Yiming Yang, and Hanxiao Liu.
Modeling long-and short-term temporal patterns with deep neural networks. In SIGIR, 2018.
Code available at: https://github.com/laiguokun/LSTNet

Word quality estimation for machine translation (Hu et al., 2018)

Junjie Hu, **Wei-Cheng Chang**, Yuexin Wu, and Graham Neubig.
Contextual encoding for translation quality estimation. In WMT, 2018.
Code available at: https://github.com/JunjieHu/CEQE

Part I

Kernel Learning for Implicit Generative Models

Chapter 2

Optimizing Kernels for Generative Moment Matching Network

2.1 Background

Modeling arbitrary density is a statistically challenging task (Wasserman, 2013). For many applications, however, accurate density estimation is not necessary since we are only interested in *sampling* from the approximated distribution. Instead of estimating the density of $\mathbb{P}_{\mathcal{X}}$, Implicit Generative Models (IGM) (Goodfellow et al., 2014; Mohamed and Lakshminarayanan, 2016) starts from a base distribution $\mathbb{P}_{\mathcal{Z}}$ over $\mathcal{Z}$, such as Gaussian distributions, then trains a transformation function g_θ such that $\mathbb{Q}_\theta \approx \mathbb{P}_{\mathcal{X}}$, where $\mathbb{Q}_\theta$ is the underlying distribution of $g_\theta(z)$ and $z \sim \mathbb{P}_{\mathcal{Z}}$.

In the heart of learning IGMs is to define a *distance* $D(\mathbb{P}_{\mathcal{X}}, \mathbb{Q}_\theta)$ between the data distribution $\mathbb{P}_{\mathcal{X}}$ and the model distribution $\mathbb{Q}_\theta$ if we can only access samples from these two distributions. We then train transformation function (i.e., the generator) g_θ by optimizing the defined distance. Generative Adversarial Network (GAN) (Goodfellow et al., 2014) train a binary classifier f_ϕ to estimate the distance between $\mathbb{P}_{\mathcal{X}}$ and $\mathbb{Q}_\theta$, which corresponds to the dual norm of Jensen-Shannon divergence. Using auxiliary neural networks to estimate different distances for training GANs have been widely studied, such as f-divergence (Mao et al., 2017; Nowozin et al., 2016). Integral probability metrics (IPM) (Müller, 1997) is another class of distance for learning IGMs, including total variation (Zhao et al., 2017), Wasserstein distance (Arjovsky et al., 2017; Gulrajani et al., 2017), Cramer distance (Bellemare et al., 2017), Fisher IPM (Mroueh and Sercu, 2017), and more.

We focus on learning IGMs with kernel maximum mean discrepancy (MMD), which is also an IPM-based distance (Gretton et al., 2012a). Specifically, Generative moment matching network (GMMN) (Dziugaite et al., 2015; Li et al., 2015b) learns the generator g_θ by minimizing the MMD distance, induced by fixed mixture of Gaussian kernels. However, GMMN do not have promising empirical results comparable with GAN on challenging benchmarks (Yu et al., 2015; Liu et al., 2015). In this chapter, we improve GMMN by optimizing deep neural-parameterized kernels for more discriminative MMD distance between $\mathbb{P}_{\mathcal{X}}$ and $\mathbb{Q}_\theta$, and draw a deeper connection between GMMN, two-sample test, kernel learning, and adversarial training.

2.2 Two-Sample Test and Generative Moment Matching Network

Assume we are given data $\{x_i\}_{i=1}^n$, where $x_i \in \mathcal{X}$ and $x_i \sim \mathbb{P}_\mathcal{X}$. If we are interested in sampling from $\mathbb{P}_\mathcal{X}$, it is not necessary to estimate the density of $\mathbb{P}_\mathcal{X}$. Instead, Generative Adversarial Network (GAN) (Goodfellow et al., 2014) trains a generator g_θ parameterized by θ to transform samples $z \sim \mathbb{P}_\mathcal{Z}$, where $z \in \mathcal{Z}$, into $g_\theta(z) \sim \mathbb{Q}_\theta$ such that $\mathbb{Q}_\theta \approx \mathbb{P}_\mathcal{X}$. To measure the distance $D(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$ between $\mathbb{P}_\mathcal{X}$ and $\mathbb{Q}_\theta$ via their samples $\{x\}_{i=1}^n$ and $\{g_\theta(z_j)\}_{j=1}^n$ during the training, Goodfellow et al. (2014) trains the discriminator f_ϕ parameterized by ϕ for help. The learning is done by playing a two-player game, where f_ϕ tries to distinguish x_i and $g_\theta(z_j)$ while g_θ aims to confuse f_ϕ by generating $g_\theta(z_j)$ similar to x_i.

On the other hand, distinguishing two distributions by finite samples is known as *Two-Sample Test* in statistics. One way to conduct two-sample test is via kernel maximum mean discrepancy (MMD) (Gretton et al., 2012a). Let k be the kernel of a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ of functions on a set $\mathcal{X}$. We assume that k measurable and bounded, $\sup_{x\in\mathcal{X}} k(x,x) < \infty$. Given a kernel k with regular conditions, the square of MMD distance between two distributions $\mathbb{P}$ and $\mathbb{Q}$ is defined as

$$M_k(\mathbb{P},\mathbb{Q}) \triangleq \|\mu_\mathbb{P} - \mu_\mathbb{Q}\|_\mathcal{H}^2 = \mathbb{E}_\mathbb{P}[k(x,x')] - 2\mathbb{E}_{\mathbb{P},\mathbb{Q}}[k(x,y)] + \mathbb{E}_\mathbb{Q}[k(y,y')], \tag{2.1}$$

where $\mu_\mathbb{P} = \mathbb{E}_{x\sim\mathbb{P}}[k(x,\cdot))], \mu_\mathbb{Q} = \mathbb{E}_{y\sim\mathbb{Q}}[k(y,\cdot))]$ are the kernel mean embedding for $\mathbb{P}$ and $\mathbb{Q}$.

Theorem 1 ((Gretton et al., 2012a, Theorem 5, p727)). *Given a kernel k, if k is a characteristic kernel, then $M_k(\mathbb{P},\mathbb{Q}) = 0$ iff $\mathbb{P} = \mathbb{Q}$.*

One example of characteristic kernel is Gaussian kernel $k(x,x') = \exp(\|x - x'\|^2)$. Based on Theorem 1, Li et al. (2015b); Dziugaite et al. (2015) propose generative moment-matching network (GMMN), which uses $M_k(\mathbb{P},\mathbb{Q})$ as $D(\mathbb{P},\mathbb{Q})$ and trains g_θ by

$$\min_\theta M_k(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta), \tag{2.2}$$

with a fixed Gaussian kernel k rather than training an additional neural-parameterized discriminator f as GAN. While GMMN is easy to train without solving a minmax objective as GANs, the empirical performance is not satisfactory as we show in Figure 2.1.

2.2.1 MMD with Kernel Learning

In practice, we consider finite samples from two distributions to estimate the MMD distance. Given $X = \{x_1, \cdots, x_n\} \sim \mathbb{P}$ and $Y = \{y_1, \cdots, y_n\} \sim \mathbb{Q}$, one estimator of $M_k(\mathbb{P},\mathbb{Q})$ is

$$\hat{M}_k(X,Y) = \frac{1}{\binom{n}{2}} \sum_{i\neq i'} k(x_i, x_{i'}) - \frac{2}{\binom{n}{2}} \sum_{i\neq j} k(x_i, y_j) + \frac{1}{\binom{n}{2}} \sum_{j\neq j'} k(y_j, y_{j'}).$$

Because of the sampling variance, $\hat{M}(X,Y)$ may not be zero even when $\mathbb{P} = \mathbb{Q}$. We then conduct hypothesis test with null hypothesis $H_0 : \mathbb{P} = \mathbb{Q}$. For a given allowable probability of false rejection α, we can only reject H_0, which imply $\mathbb{P} \neq \mathbb{Q}$, if $\hat{M}(X,Y) > c_\alpha$ for some chose threshold $c_\alpha > 0$. Otherwise, $\mathbb{Q}$ passes the hypothesis test and $\mathbb{Q}$ is indistinguishable from $\mathbb{P}$ under this test. Please refer to Gretton et al. (2012a) for more details.

Intuitively, if kernel k cannot result in high MMD distance $M_k(\mathbb{P}, \mathbb{Q})$ when $\mathbb{P} \neq \mathbb{Q}$, $\hat{M}_k(\mathbb{P}, \mathbb{Q})$ has more chance to be smaller than c_α. Then we are unlikely to reject the null hypothesis H_0 with finite samples, which implies $\mathbb{Q}$ is not distinguishable from $\mathbb{P}$. Therefore, instead of training g_θ via Eq. (2.2) with a pre-specified kernel k as GMMN, we consider training g_θ via

$$\min_\theta \max_{k \in \mathcal{K}} M_k(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta), \tag{2.3}$$

which takes different possible characteristic kernels $k \in \mathcal{K}$ into account. On the other hand, we could also view Eq. (2.3) as replacing the fixed kernel k in Eq. (2.2) with the *adversarially learned kernel* via kernel learning $\arg\max_{k \in \mathcal{K}} M_k(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$ to have stronger signal where $\mathbb{P}_\mathcal{X} \neq \mathbb{Q}_\theta$ to train g_θ. We refer interested readers to Fukumizu et al. (2009) for more rigorous discussions about testing power and increasing MMD distances.

However, it is difficult to optimize over all characteristic kernels when we solve Eq. (2.3). By Gretton et al. (2012a,b) if f is an injective function and k is characteristic, then the compositionally-induced kernel $\tilde{k} = k \circ f$, where $\tilde{k}(x, x') = k(f(x), f(x'))$ is still characteristic. If we have a family of injective functions parameterized by ϕ, which is denoted as f_ϕ, we are able to change the objective to be

$$\min_\theta \max_\phi M_{k \circ f_\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta). \tag{2.4}$$

We consider the case that combining Gaussian kernels with injective functions f_ϕ, where $\tilde{k}(x, x') = \exp(-\|f_\phi(x) - f_\phi(x)'\|^2)$. One example function class of f is $\{f_\phi | f_\phi(x) = \phi x, \phi > 0\}$, which is equivalent to the kernel bandwidth tuning. A more advanced realization will be discussed in Section 2.3. Next, we abuse the notation $M_{f_\phi}(\mathbb{P}, \mathbb{Q})$ to be MMD distance given the composition kernel of Gaussian kernel and f_ϕ in the following. Note that Gretton et al. (2012b) consider the linear combination of characteristic kernels, which can also be incorporated into the discussed composition kernels. More general kernels are studied in Wilson et al. (2016).

2.2.2 Properties of MMD with Kernel Learning

Arjovsky et al. (2017) discuss different distances between distributions adopted by existing deep learning algorithms, and show many of them are discontinuous, such as Jensen-Shannon divergence (Goodfellow et al., 2014) and Total variation (Zhao et al., 2017), except for Wasserstein distance. The discontinuity makes the gradient descent infeasible for training. From Eq. (2.4), we train g_θ via minimizing $\max_\phi M_{f_\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$. Next, we show $\max_\phi M_{f_\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$ also enjoys the advantage of being a continuous and differentiable objective in θ under mild assumptions.

Assumption 2. *$g : \mathcal{Z} \times \mathbb{R}^m \to \mathcal{X}$ is locally Lipschitz, where $\mathcal{Z} \subseteq \mathbb{R}^d$. We will denote $g_\theta(z)$ the evaluation on (z, θ) for convenience. Given a Lipschitz f_ϕ and a probability distribution $\mathbb{P}_z$ over $\mathcal{Z}$, g satisfies Assumption 2 if there are local Lipschitz constants $L(\theta, z)$ for $f_\phi \circ g$, which is independent of ϕ, such that $\mathbb{E}_{z \sim \mathbb{P}_z}[L(\theta, z)] < +\infty$.*

Theorem 3. *The generator function g_θ parameterized by θ is under Assumption 2. Let $\mathbb{P}_\mathcal{X}$ be a fixed distribution over $\mathcal{X}$ and Z be a random variable over the space $\mathcal{Z}$. We denote $\mathbb{Q}_\theta$ the distribution of $g_\theta(Z)$, then $\max_\phi M_{f_\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$ is continuous everywhere and differentiable almost everywhere in θ.*

If g_θ is parameterized by a feed-forward neural network, it satisfies Assumption 2 and can be trained via gradient descent as well as propagation, since the objective is continuous and differentiable followed by Theorem 3.

Recall that IPM defines the probability distance as

$$D(\mathbb{P}, \mathbb{Q}) = \sup_{f \in \mathcal{F}} \mathbb{E}_{x \sim \mathbb{P}}[f(x)] - \mathbb{E}_{y \sim \mathbb{Q}}[f(y)]. \tag{2.5}$$

With different function class $\mathcal{F}$, we can recover several distances, such as total variation, Wasserstein distance, and MMD distance. For MMD, the function class $\mathcal{F}$ is $\{\|f\|_{\mathcal{H}_k} \leq 1$, where $\mathcal{H}$ is RKHS associated with kernel k. Different from many distances (e.g., Wasserstein distance), there is an analytical representation (Gretton et al., 2012a) of MMD-based IPM, which is

$$\begin{aligned} M_k(\mathbb{P}, \mathbb{Q}) &= \left[\sup_{f \in \mathcal{H}_k} \mathbb{E}_{x \sim \mathbb{P}}[f(x)] - \mathbb{E}_{y \sim \mathbb{Q}}[f(y)] \right]^2 \\ &= \mathbb{E}_{\mathbb{P}}[k(x, x')] - 2\mathbb{E}_{\mathbb{P},\mathbb{Q}}[k(x, y)] + \mathbb{E}_{\mathbb{Q}}[k(y, y')]. \end{aligned} \tag{2.6}$$

Thus, GMMN does not require an additional discriminator network f_ϕ to estimate the distance $D(\mathbb{P}, \mathbb{Q}) = M_k(\mathbb{P}, \mathbb{Q})$. Here, we provide an interpretation of the proposed MMD with kernel learning under the IPM framework. The MMD distance with adversarial learned kernels is

$$\max_{k \in \mathcal{K}} M_k(\mathbb{P}, \mathbb{Q}) = \sup_{f \in \mathcal{H}_{k_1} \cup \ldots \cup \mathcal{H}_{k_n}} \mathbb{E}_{x \sim \mathbb{P}}[f(x)] - \mathbb{E}_{y \sim \mathbb{Q}}[f(y)],$$

where $k_i \in \mathcal{K}, \forall i$. The proposed MMD distance with kernel learning is still defined by IPM but with a larger function class. Next, we prove $\max_\phi M_{f_\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$ is not only a valid probability distance, but also enjoys the *weak converegence* under Assumption 2.

Theorem 4. *(weak* topology) Let $\{\mathbb{P}_n\}$ be a sequence of distributions. Considering $n \to \infty$, under mild Assumption 2, $\max_\phi M_{f_\phi}(\mathbb{P}_\mathcal{X}, \mathbb{P}_n) \to 0 \iff \mathbb{P}_n \xrightarrow{D} \mathbb{P}_\mathcal{X}$, where $\xrightarrow{D}$ means* converging in distribution *(Wasserman, 2013).*

Theorem 4 shows that $\max_\phi M_{f_\phi}(\mathbb{P}_\mathcal{X}, \mathbb{P}_n)$ is a sensible cost function to the distance between $\mathbb{P}_\mathcal{X}$ and $\mathbb{P}_n$. The distance is decreasing when $\mathbb{P}_n$ is getting closer to $\mathbb{P}_\mathcal{X}$, which benefits the supervision of the improvement during the training. All proofs are omitted to Section 2.6. Next, we introduce a practical realization of training g_θ by optimizing $\min_\theta \max_\phi M_{f_\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$.

2.3 MMD GAN

To approximate Eeq. (2.4), we use neural networks to parameterized g_θ and f_ϕ with expressive power. For g_θ, the assumption is locally Lipschitz, where commonly used feed-forward neural networks satisfy this constraint. Also, gradient $\bigtriangledown_\theta \left(\max_\phi f_\phi \circ g_\theta\right)$ has to be bounded, which can be done by weight clipping ϕ (Arjovsky et al., 2017) or gradient penalty (Gulrajani et al., 2017). The non-trivial part is f_ϕ has to be injective. For an injective function f, there exists an function f^{-1} such that $f^{-1}(f(x)) = x, \forall x \in \mathcal{X}$ and $f^{-1}(f(g(z))) = g(z), \forall z \in \mathcal{Z}$[1], which can be approximated by an autoencoder. In the following, we

[1]Note that injective is not necessary invertible.

denote $\phi = \{\phi_e, \phi_d\}$ to be the parameter of discriminator networks, which consists of an encoder f_{ϕ_e}, and train the corresponding decoder $f_{\phi_d} \approx f^{-1}$ to regularize f. The objective (2.4) is relaxed to be

$$\min_\theta \max_\phi M_{f_{\phi_e}}(\mathbb{P}(\mathcal{X}), \mathbb{P}(g_\theta(\mathcal{Z}))) - \lambda \mathbb{E}_{y \in \mathcal{X} \cup g(\mathcal{Z})} \|y - f_{\phi_d}(f_{\phi_e}(y))\|^2. \tag{2.7}$$

Note that Bińkowski et al. (2018) extend the analysis of Theorem 3 to get rid of the injective constraint. Our empirical study suggests autoencoder objective is not necessary to the success of MMD GAN training as we will show in Section 2.4.5, which is consistent with Bińkowski et al. (2018).

The proposed algorithm is similar to GAN (Goodfellow et al., 2014), which aims to optimize two neural networks g_θ and f_ϕ in a minmax formulation, while the meaning of the objective is different. In Goodfellow et al. (2014), f_{ϕ_e} is a discriminator (binary) classifier to distinguish two distributions. In the proposed algorithm, distinguishing two distribution is still done by two-sample test via MMD, but with an adversarially learned kernel parametrized by f_{ϕ_e}. g_θ is then trained to pass the hypothesis test. Because of the similarity of GAN, we call the proposed algorithm *MMD GAN*. We present an implementation with the weight clipping in Algorithm 1, but one can easily extend to other Lipschitz approximations, such as gradient penalty Gulrajani et al. (2017).

Algorithm 1: MMD GAN, our proposed algorithm.

input : α the learning rate, c the clipping parameter, B the batch size, n_c the number of iterations of discriminator per generator update.

initialize generator parameter θ and discriminator parameter ϕ;

while θ *has not converged* **do**

  for $t = 1, \dots, n_c$ **do**

    Sample a minibatches $\{x_i\}_{i=1}^B \sim \mathbb{P}(\mathcal{X})$ and $\{z_j\}_{j=1}^B \sim \mathbb{P}(\mathcal{Z})$

    $g_\phi \leftarrow \nabla_\phi M_{f_{\phi_e}}(\mathbb{P}(\mathcal{X}), \mathbb{P}(g_\theta(\mathcal{Z}))) - \lambda \mathbb{E}_{y \in \mathcal{X} \cup g(\mathcal{Z})} \|y - f_{\phi_d}(f_{\phi_e}(y))\|^2$

    $\phi \leftarrow \phi + \alpha \cdot \text{RMSProp}(\phi, g_\phi)$

    $\phi \leftarrow \text{clip}(\phi, -c, c)$

  Sample a minibatches $\{x_i\}_{i=1}^B \sim \mathbb{P}(\mathcal{X})$ and $\{z_j\}_{j=1}^B \sim \mathbb{P}(\mathcal{Z})$

  $g_\theta \leftarrow \nabla_\theta M_{f_{\phi_e}}(\mathbb{P}(\mathcal{X}), \mathbb{P}(g_\theta(\mathcal{Z})))$

  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$

2.3.1 Feasible Set Reduction

Theorem 5. *For any f_ϕ, there exists f'_ϕ such that $M_{f_\phi}(\mathbb{P}_r, \mathbb{Q}_\theta) = M_{f'_\phi}(\mathbb{P}_r, \mathbb{Q}_\theta)$ and $\mathbb{E}_x[f_\phi(x)] \succeq \mathbb{E}_z[f_{\phi'}(g_\theta(z))]$.*

With Theorem 5, we could reduce the feasible set of ϕ during the optimization by solving

$$\min_\theta \max_\phi M_{f_\phi}(\mathbb{P}_r, \mathbb{Q}_\theta) s.t.\ \mathbb{E}[f_\phi(x)] \succeq \mathbb{E}[f_\phi(g_\theta(z))]$$

which the optimal solution is still *equivalent* to solving (2.3). However, it is hard to solve the constrained optimization problem with backpropagation. We relax the constraint by ordinal regression (Herbrich et al., 1999) to be

$$\min_\theta \max_\phi M_{f_\phi}(\mathbb{P}_r, \mathbb{Q}_\theta) + \lambda \min\left(\mathbb{E}[f_\phi(x)] - \mathbb{E}[f_\phi(g_\theta(z))], 0\right),$$

which only penalizes the objective when the constraint is violated. In practice, we observe that reducing the feasible set makes the training faster and stabler.

2.3.2 Encoding Perspectives and Relations to WGAN

Besides from using kernel selection to explain MMD GAN, the other way to see the proposed MMD GAN is viewing f_{ϕ_e} as a feature transformation function, and the kernel two-sample test is performed on this transformed feature space (i.e., the code space of the autoencoder). The optimization is finding a manifold with stronger signals for MMD two-sample test. From this perspective, GMMN (Li et al., 2015b; Dziugaite et al., 2015) is the special case of MMD GAN if f_{ϕ_e} is the identity mapping function. In such circumstance, the kernel two-sample test is conducted in the original data space.

If we composite f_ϕ with linear kernel instead of Gaussian kernel, and restricting the output dimension h to be 1, we then have the objective

$$\min_\theta \max_\phi \|\mathbb{E}[f_\phi(x)] - \mathbb{E}[f_\phi(g_\theta(z))]\|^2. \tag{2.8}$$

Parameterizing f_ϕ and g_θ with neural networks and assuming $\exists \phi' \in \Phi$ such $f'_\phi = -f_\phi, \forall \Phi$, recovers Wasserstein GAN (WGAN) (Arjovsky et al., 2017) [2]. If we treat $f_\phi(x)$ as the data transform function, WGAN can be interpreted as first-order moment matching (linear kernel) while MMD GAN aims to match infinite order of moments with Gaussian kernel form Taylor expansion (Li et al., 2015b). Theoretically, Wasserstein distance has similar theoretically guarantee as Theorem 1, 3 and 4. In practice, Arora et al. (2017) show neural networks does not have enough capacity to approximate Wasserstein distance. In Section 7.4.1, we demonstrate matching high-order moments benefits the results. Mroueh et al. (2017) also propose McGAN that matches second order moment from the primal-dual norm perspective. However, the proposed algorithm requires matrix (tensor) decompositions because of exact moment matching (Zellinger et al., 2017), which is hard to scale to higher order moment matching. On the other hand, by giving up exact moment matching, MMD GAN can match high-order moments with kernel tricks.

Difference from Other Works with Autoencoders Energy-based GANs (Zhao et al., 2017; Zhai et al., 2016) also utilizes the autoencoder (AE) in its discriminator from the energy model perspective, which minimizes the reconstruction error of real samples x while maximize the reconstruction error of generated samples $g_\theta(z)$. In contrast, MMD GAN uses AE to approximate invertible functions by minimizing the reconstruction errors of *both* real samples x and generated samples $g_\theta(z)$. Also, Arjovsky et al. (2017) show EBGAN approximates total variation, with the drawback of discontinuity, while MMD GAN optimizes MMD distance. The other line of works (Kingma and Welling, 2013; Makhzani et al., 2015; Li et al., 2015b) aims to match the AE codespace $f(x)$, and utilize the decoder $f_{dec}(\cdot)$. Kingma and Welling (2013); Makhzani et al. (2015) match the distribution of $f(x)$ and z via different distribution distances and generate data (e.g. image) by $f_{dec}(z)$. Li et al. (2015b) use MMD to match $f(x)$ and $g(z)$, and generate data via $f_{dec}(g(z))$. The proposed MMD GAN matches the $f(x)$ and $f(g(z))$, and generates data via $g(z)$ directly as GAN. Ulyanov et al. (2017) is similar to MMD GAN but it considers KL-divergence without showing continuity and weak* topology guarantee.

[2] Theoretically, they are not equivalent but the practical neural network approximation results in the same algorithm.

2.4 Experiments and Results

Datasets We train MMD GAN for image generation on the MNIST (LeCun et al., 1998), CIFAR-10 (Krizhevsky and Hinton, 2009), CelebA (Liu et al., 2015), and LSUN bedrooms (Yu et al., 2015) datasets, where the size of training instances are 50K, 50K, 160K, 3M respectively.

Network Architecture and Kernel Designs In our experiments, we follow the architecture of DCGAN (Radford et al., 2016) to design g_θ by its generator and f_ϕ by its discriminator except for expanding the output layer of f_ϕ to be h dimensions. The loss function of MMD GAN is implicitly associated with a family of characteristic kernels. Similar to the prior works (Dziugaite et al., 2015; Li et al., 2015b; Sutherland et al., 2017), we consider a mixture of K RBF kernels $k(x, x') = \sum_{q=1}^{K} k_{\sigma_q}(x, x')$ where k_{σ_q} is a Gaussian kernel with bandwidth parameter σ_q. Tuning kernel bandwidth σ_q optimally still remains an open problem. In this works, we fixed $K = 5$ and σ_q to be $\{1, 2, 4, 8, 16\}$ and left the f_ϕ to learn the kernel (feature representation) under these σ_q.

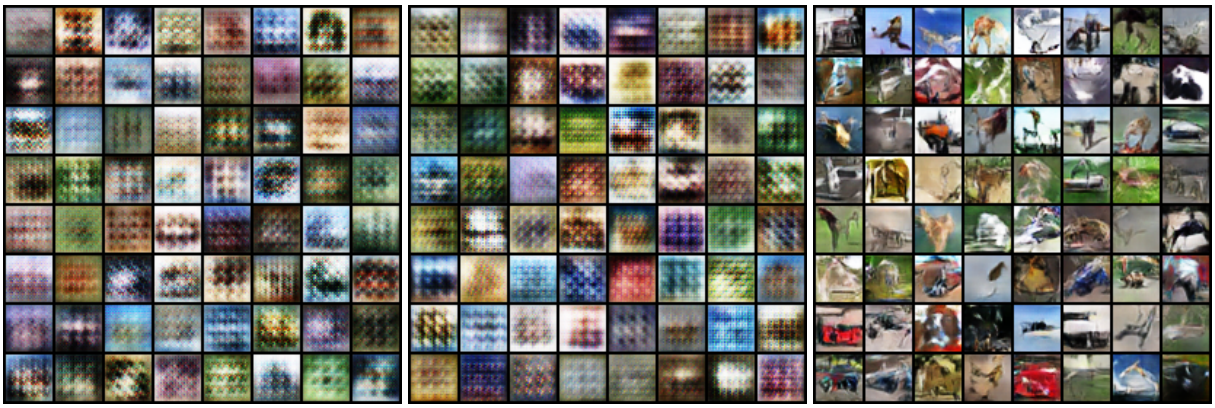
Hyper-parameters We use RMSProp (Tieleman and Hinton, 2012) with learning rate of 0.00005 for a fair comparison with WGAN as suggested in Arjovsky et al. (2017). We ensure the boundedness of model parameters of discriminator by clipping the weights point-wisely to the range $[-0.01, 0.01]$ as required by Assumption 2. The dimensionality h of the latent space is manually set according to the complexity of the dataset. We thus use $h = 16$ for MNIST, $h = 64$ for CelebA, and $h = 128$ for CIFAR-10 and LSUN bedrooms. The batch size is set to be $B = 64$ for all datasets.

2.4.1 Qualitative Analysis

Comparison with GMMN We start with comparing MMD GAN with GMMN on two standard benchmarks, MNIST and CIFAR-10. We consider two variants for GMMN. The first one is original GMMN, which trains the generator by minimizing the MMD distance on the original data space. We call it as *GMMN-D*. To compare with MMD GAN, we also pretrain an autoencoder for projecting data to a manifold, then fix the autoencoder as a feature transformation, and train the generator by minimizing the MMD distance in the code space. We call it as *GMMN-C*. The results are pictured in Figure 2.1. Both GMMN-D and GMMN-C are able to generate meaningful digits on MNIST because of the simple data structure. By a closer look, nonetheless, the boundary and shape of the digits in Figure 2.1a and 2.1b are often irregular and non-smooth. In contrast, the sample digits in Figure 2.1c are more natural with smooth outline and sharper strike. For CIFAR-10 dataset, both GMMN variants fail to generate meaningful images, but resulting some low level visual features. We observe similar cases in other complex large-scale datasets such as CelebA and LSUN bedrooms, thus results are omitted. On the other hand, the proposed MMD GAN successfully outputs natural images with sharp boundary and high diversity. The results in Figure 2.1 confirm the success of the proposed adversarial learned kernels to enrich statistical testing power, which is the key difference between GMMN and MMD GAN.

If we increase the batch size of GMMN to 1024, the image quality is improved, as shown in Figure 2.2. However, it is still not competitive to MMD GAN with $B = 64$. This demonstrates that the proposed MMD GAN can be trained more efficiently than GMMN with smaller batch size.

(a) GMMN-D, MNIST (b) GMMN-C, MNIST (c) MMD GAN, MNIST



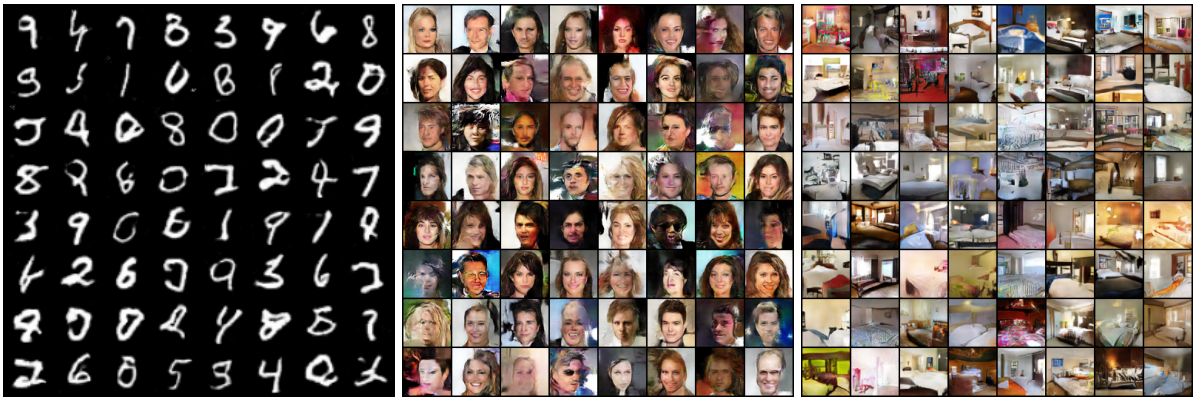
(d) GMMN-D, CIFAR-10 (e) GMMN-C, CIFAR-10 (f) MMD GAN, CIFAR-10

Figure 2.1: Generated samples from GMMN-D (Dataspace), GMMN-C (Codespace) and our MMD GAN with batch size $B = 64$.

(a) GMMN-D, MNIST (b) GMMN-C, MNIST (c) GMMN-D, CIFAR-10 (d) GMMN-C, CIFAR-10

Figure 2.2: Generative samples from GMMN-D and GMMN-C with training batch size $B = 1024$.

Comparing with GANs There are several representative extensions of GANs. We consider recent state-of-art WGAN (Arjovsky et al., 2017) based on DCGAN structure Radford et al. (2016), because of the connection with MMD GAN. The results are shown in Figure 2.3. For MNIST, the digits generated from WGAN in Figure 2.3a are more unnatural with peculiar strikes. In Contrary, the digits from MMD GAN



(a) WGAN MNIST (b) WGAN CelebA (c) WGAN LSUN

(d) MMD GAN MNIST (e) MMD GAN CelebA (f) MMD GAN LSUN

Figure 2.3: Generated samples from WGAN and MMD GAN on MNIST, CelebA, and LSUN bedroom datasets.

in Figure 2.3d enjoy smoother contour. Furthermore, both WGAN and MMD GAN generate diversified digits, avoiding the mode collapse problems appeared in the literature of training GANs. For CelebA, we can see the difference of generated samples from WGAN and MMD GAN. Specifically, we observe varied poses, expressions, genders, skin colors and light exposure in Figure 2.3b and 2.3e. By a closer look (view on-screen with zooming in), we observe that faces from WGAN have higher chances to be blurry and twisted while faces from MMD GAN are more spontaneous with sharp and acute outline of faces. As for LSUN dataset, we could not distinguish salient differences between the samples generated from MMD GAN and WGAN.

2.4.2 Quantitative Evaluation

To quantitatively measure the quality and diversity of generated samples, we compute the inception score (Salimans et al., 2016) on CIFAR-10 images. The inception score is used for GANs to measure samples quality and diversity on the pretrained inception model (Salimans et al., 2016). Models that generate collapsed samples have a relatively low score. Table 2.1 lists the results for $50K$ samples generated by various unsupervised generative models trained on CIFAR-10 dataset. The inception scores

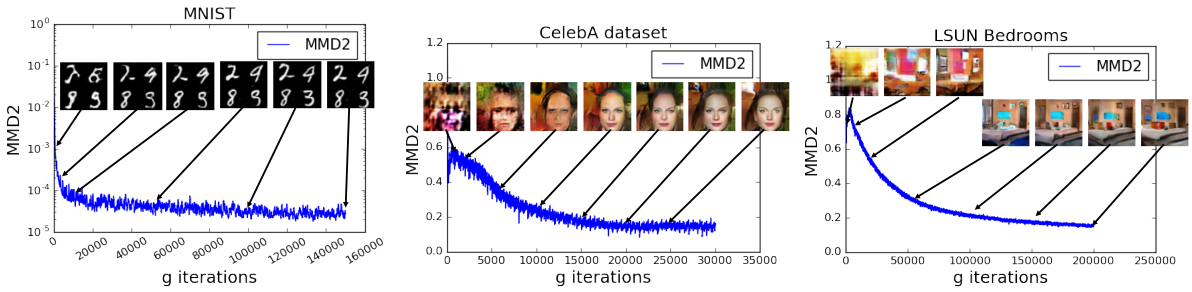
of DFM (Warde-Farley and Bengio, 2017), ALI (Dumoulin et al., 2017) and Improved-GAN (Salimans et al., 2016) are directly derived from the corresponding references.

Although both WGAN and MMD GAN can generate sharp images as we show, our score is better than other GAN techniques except for DFM (Warde-Farley and Bengio, 2017). This seems to confirm empirically that higher order of moment matching between the real data and fake sample distribution benefits generating more diversified sample images. Also note DFM appears compatible with our method and combing training techniques in DFM is a possible avenue for future work.

Method	Scores $\pm$ std.
Real data	$11.95 \pm .20$
DFM (Warde-Farley and Bengio, 2017)	7.72
ALI (Dumoulin et al., 2017)	5.34
Improved GANs (Salimans et al., 2016)	4.36
MMD GAN	**6.17** $\pm$ .07
WGAN	$5.88 \pm .07$
GMMN-C	$3.94 \pm .04$
GMMN-D	$3.47 \pm .03$

Table 2.1: Inception scores of MMD GAN and different GAN algorithms.

2.4.3 Stability of MMD GAN



(a) MNIST (b) CelebA (c) LSUN Bedrooms

Figure 2.4: Training curves and generative samples at different stages of training. We can see a clear correlation between lower distance and better sample quality.

We further illustrate how the MMD distance correlates well with the quality of the generated samples. Figure 2.4 plots the evolution of the MMD GAN estimate the MMD distance during training for MNIST, CelebA and LSUN datasets. We report the average of the $\hat{M}_{f_\phi}(\mathbb{P}_\mathcal{X}, \mathbb{P}_\theta)$ with moving average to smooth the graph to reduce the variance caused by mini-batch stochastic training. We observe during the whole training process, samples generated from the same noise vector across iterations, remain similar in nature. (e.g., face identity and bedroom style are alike while details and backgrounds will evolve.) This qualitative

observation indicates valuable stability of the training process. The decreasing curve with the improving quality of images supports the weak* topology shown in Theorem 4. Also, We can see from the plot that the model converges very quickly. In Figure 2.4b, for example, it converges shortly after tens of thousands of generator iterations on CelebA dataset.

2.4.4 Computation Issue

We conduct time complexity analysis with respect to the batch size B. The time complexity of each iteration is $O(B)$ for WGAN and $O(KB^2)$ for our proposed MMD GAN with a mixture of K RBF kernels. The quadratic complexity $O(B^2)$ of MMD GAN is introduced by computing kernel matrix, which is sometimes criticized for being inapplicable with large batch size in practice. However, we point that there are several recent works, such as EBGAN Zhao et al. (2017), also matching pairwise relation between samples of batch size, leading to $O(B^2)$ complexity as well.

Empirically, we find that under GPU environment, the highly parallelized matrix operation tremendously alleviated the quadratic time to almost linear time with modest B. Figure 2.5 compares the computational time per generator iterations versus different B on Titan X. When $B = 64$, which is adapted for training MMD GAN in our experiments setting, the time per iteration of WGAN and MMD GAN is 0.268 and 0.676 seconds, respectively. When $B = 1024$, which is used for training GMMN in its references (Li et al., 2015b), the time per iteration becomes 4.431 and 8.565 seconds, respectively. This result coheres our argument that the empirical computational time for MMD GAN is not quadratically expensive compared to WGAN with powerful GPU parallel computation.

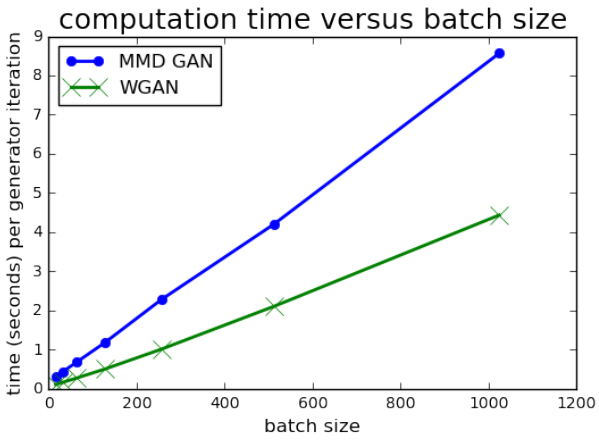
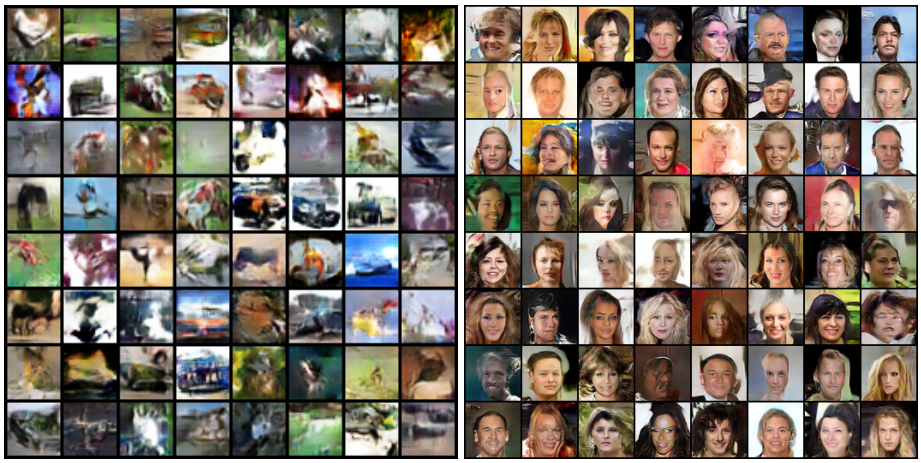


Figure 2.5: Computation time per one generator update step with different batch sizes.

2.4.5 Better Lipschitz Approximation and Necessity of Auto-Encoder

We used weight-clipping for Lipschitz constraint in Assumption 2. Another approach for obtaining a discriminator with the similar constraints that approximates a Wasserstein distance is Gulrajani et al. (2017), where the gradient of the discriminator is constrained to be 1 between the generated and data points. Inspired by Gulrajani et al. (2017), an alternative approach is to apply the gradient constraint as

a regularizer to the witness function for a different IPM, such as the MMD. This idea was first proposed in Bellemare et al. (2017) for the Energy Distance (in parallel to our submission), which was shown in Gretton (2017) to correspond to a gradient penalty on the witness function for any RKHS-based MMD. Here we undertake a preliminary investigation of this approach, where we also drop the requirement in Algorithm 1 for f_ϕ to be injective, which we observe that it is not necessary in practice. We show some preliminary results of training MMD GAN with gradient penalty and without the auto-encoder in Figure 2.6. The preliminary study indicates that MMD GAN can generate satisfactory results with other Lipschitz constraint approximation. One potential future work is conducting more thorough empirical comparison studies between different approximations.



(a) Cifar10, $Giter = 300K$ (b) CelebA, $Giter = 300K$

Figure 2.6: MMD GAN results using gradient penalty Gulrajani et al. (2017) and without auto-encoder reconstruction loss during training.

2.5 Summary

We introduce a new deep generative model trained via MMD with adversarially learned kernels. We further study its theoretical properties and propose a practical realization MMD GAN, which can be trained with much smaller batch size than GMMN and has competitive performances with state-of-the-art GANs. We can view MMD GAN as the first practical step forward connecting moment matching network and GAN. One important direction is applying developed tools in moment matching (Muandet et al., 2017) on general GAN works based the connections shown by MMD GAN. Also, in Section 2.3.2, we connect WGAN and MMD GAN by first-order and infinite-order moment matching. Finally, we observe that it is not mandatory in practice as we show in Section 2.4.5.

2.6 Appendix

2.6.1 Proof of Theorem 3

The proof is following the Lemma from Borisenko and Minchenko (1992), where we state below.

Lemma 6 (Borisenko and Minchenko (1992)). *Define $\tau(x) = \max\{f(x,u) \mid u \in U\}$. If f is locally Lipschitz in x, U is compact and $\nabla f(x, u^*(x))$ exists, where $u^*(x) = \arg\max_u f(x,u)$, then $\tau(x)$ is differentiable almost everywhere.*

For simplicity, we define $k_\phi \triangleq k \circ f_\phi$. We aim to show

$$\max_\phi M_\phi(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta) = \mathbb{E}_{x,x'}[k_\phi(x,x')] - 2\mathbb{E}_{x,z}[k_\phi(x, g_\theta(z))] + \mathbb{E}_{z,z'}[k_\phi(g_\theta(z), g_\theta(z'))] \tag{2.9}$$

is differentiable with respect to ϕ almost everywhere by using the auxiliary Lemma 6. We first show that $\mathbb{E}_{z,z'}[k_\phi(g_\theta(z), g_\theta(z'))]$ in Eq. 2.9 is locally Lipschitz in θ. By definition, $k_\phi(x,x') = k(f_\phi(x) - f_\phi(x'))$, thus,

$$\begin{aligned}
&\mathbb{E}_{z,z'}\left[k_\phi\Big(g_\theta(z), g_\theta(z')\Big) - k_\phi\Big(g_{\theta'}(z), g_{\theta'}(z')\Big)\right] \\
&= \mathbb{E}_{z,z'}\left[k\Big(f_\phi\big(g_\theta(z)\big) - f_\phi\big(g_\theta(z')\big)\Big)\right] - \mathbb{E}_{z,z'}\left[k\Big(f_\phi\big(g_{\theta'}(z)\big) - f_\phi\big(g_{\theta'}(z')\big)\Big)\right] \\
&\leq \mathbb{E}_{z,z'}\left[L_k\Big\|f_\phi\big(g_\theta(z)\big) - f_\phi\big(g_\theta(z')\big) - f_\phi\big(g_{\theta'}(z)\big) + f_\phi\big(g_{\theta'}(z')\big)\Big\|\right] \\
&\leq \mathbb{E}_{z,z'}\left[L_k L(\theta,z)\|\theta - \theta'\| + L_k L(\theta,z')\|\theta - \theta'\|\right] \\
&= 2L_k\mathbb{E}_z[L(\theta,z)]\|\theta - \theta'\|.
\end{aligned} \tag{2.10}$$

The first inequality is because Gaussian kernel k is Lipschitz (locally Lipschitz) in (x,x'), which a upper bound L_k for Lipschitz constants. By Assumption 2, $\mathbb{E}_z[L(\theta,z)] < \infty$, we prove $\mathbb{E}_{z,z'}[k_\phi(g_\theta(z), g_\theta(z'))]$ is locally Lipschitz. The similar argument is applicable to other terms in Eq. 2.9. Therefore, Eq. 2.9 is locally Lipschitz in θ.

Last, with the compactness assumption in Φ, and the differentiable assumption in $M_\phi(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$, apply Lemma 6 proves Theorem 3.

2.6.2 Proof of Theorem 4

Proof. We first show that, if $\mathbb{P}_n \xrightarrow{D} \mathbb{P}$ then $\max_\phi M_{f_\phi}(\mathbb{P}_n, \mathbb{P}) \to 0$. The result is based on (Dudley, 2018, Corollary 11.3.4). Similar proof is also shown in Arbel et al. (2018). Following Dudley (2018), the only thing we need to show is proving $\|k(f_\phi(x),\cdot) - k(f_\phi(y),\cdot)\|_{\mathcal{H}_k}$ is Lipschitz. By definition, we know that $\|k(f_\phi(x),\cdot) - k(f_\phi(y),\cdot)\|_{\mathcal{H}_k} = 2\big[1 - k(f_\phi(x), f_\phi(y))\big]$. Also, since Gaussian kernel k is Lipschitz, we have $k(0) - k(x,x') \leq L_k\|0 - (x - x')\|$. With $k(0) = 1$ for Gaussian kernel,

$$\|k(f_\phi(x),\cdot) - k(f_\phi(y),\cdot)\|_{\mathcal{H}_k} \leq 2L_k\|f_\phi(x) - f_\phi(y)\| \leq 2L_k L\|x - y\|,$$

where the last inequality holds since f_ϕ is also Lipschitz function with Lipschitz constant L.

The other direction, $\max_\phi M_{f_\phi}(\mathbb{P}_n, \mathbb{P}) \to 0$ then $\mathbb{P}_n \xrightarrow{D} \mathbb{P}$, is relatively simple. Without loss of generality, we assume there exists ϕ such that f_ϕ is an identity function (up to scaling), which recover the Gaussian kernel k. Therefore, $\max_\phi M_{f_\phi}(\mathbb{P}_n, \mathbb{P}) \to 0$ implies $\mathbb{P}_n \xrightarrow{D} \mathbb{P}$, which completes the proof because MMD with any Gaussian kernels is weak Gretton et al. (2012a). □

2.6.3 Proof of Theorem 5

Proof. The proof assumes $f_\phi(x)$ is scalar, but the vector case can be proved with the same sketch. First, if $\mathbb{E}[f_\phi(x)] > \mathbb{E}[f_\phi(g_\theta(z))]$, then $\phi = \phi'$. If $\mathbb{E}[f_\phi(x)] < \mathbb{E}[f_\phi(g_\theta(z))]$, we let $f = -f_\phi$, then $\mathbb{E}[f(x)] > \mathbb{E}[f(g_\theta(z))]$ and flipping sign does not change the MMD distance. If we parameterized f_ϕ by a neural network, which has a linear output layer, ϕ' can realized by flipping the sign of the weights of the last layer. □

Chapter 3

Learning Kernel Spectral Distributions

3.1 Background

Kernel methods are among the essential foundations in machine learning and have been extensively studied in the past decades. In supervised learning, kernel methods allow us to learn non-linear hypothesis. They also play a crucial role in statistics. Kernel maximum mean discrepancy (MMD) (Gretton et al., 2012a) is a powerful two-sample test, which is based on a statistic computed via kernel functions. Even though there is a surge of deep learning in the past years, several successes have been shown by kernel methods and deep feature extraction. Wilson et al. (2016) demonstrate state-of-the-art performance by incorporating deep learning, kernel and Gaussian process. Also, in Chapter 2, we show MMD GAN reaches state-of-the-art performance on generating complex image data.

In practice, however, kernel selection is always an important step. Instead of choosing by a heuristic, several works have studied *kernel learning*. Multiple kernel learning (MKL) (Bach et al., 2004; Lanckriet et al., 2004; Bach, 2009; Gönen and Alpaydın, 2011; Duvenaud et al., 2013) is one of the pioneering frameworks to combine predefined kernels. One recent kernel learning development focus on learning kernel spectral distributions (Fourier transform of the kernel). Wilson and Adams (2013) model spectral distributions via Gaussian mixtures, which is an extension of linear combination of kernels (Bach et al., 2004). Oliva et al. (2016) extend it to Bayesian non-parametric models. In addition to model spectral distribution with *explicit* density models, many works optimize the sampled random features or its weights (Băzăvan et al., 2012; Yang et al., 2015; Sinha and Duchi, 2016; Chang et al., 2017a; Bullins et al., 2018). The other orthogonal approach to is learning feature maps for standard kernels (e.g., Gaussian). Feature maps learned by deep learning lead to state-of-the-art performance on different tasks (Hinton and Salakhutdinov, 2008; Wilson et al., 2016; Li et al., 2017).

In this chapter, we propose to model kernel spectral distributions with implicit generative models, which we call *Implicit Kernel Learning* (IKL). The IGMs are learned to generate samples from the desirable kernel spectral distributions according to different downstream tasks. We start from the generic IKL framework and propose an easily-implemented neural network parameterization which satisfies Bochner's theorem (Rudin, 1962). We then demonstrate two applications of the proposed IKL. Firstly, we explore MMD GAN (Li et al., 2017) with IKL on learning to generate images and text. Secondly, we consider a standard two-staged supervised learning task with Random Kitchen Sinks (Sinha and Duchi, 2016). The

conditions required for training IKL and its theoretical guarantees in both tasks are also studied. In both tasks, we show that IKL leads to competitive or better performance than heuristic kernel selections and existing approaches modeling kernel spectral densities. It demonstrates the potentials of learning more powerful kernels via deep generative models.

3.2 IKL Framework

Given data $x \in \mathbb{R}^d$, kernel methods compute the inner product of the feature transformation $\varphi(x)$ in a high-dimensional Hilbert space H via a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which is defined as $k(x, x') = \langle \varphi(x), \varphi(x') \rangle_H$, where $\varphi(x)$ is usually high or even infinitely dimensional. If k is shift invariant (i.e. $k(x, y) = k(x - y)$), we can represent k as an expectation with respect to a spectral distribution $\mathbb{P}_k(\omega)$.

Bochner's theorem (Rudin, 1962) A continuous, real valued, symmetric and shift-invariant function k on $\mathbb{R}^d$ is a positive definite kernel if and only if there is a positive finite measure $\mathbb{P}_k(\omega)$ such that

$$k(x - x') = \int_{\mathbb{R}^d} e^{i\omega^\top (x-x')} d\mathbb{P}_k(\omega) = \mathbb{E}_{\omega \sim \mathbb{P}_k} \left[e^{i\omega^\top (x-x')} \right].$$

IKL objective We restrict ourselves to learning shift invariant kernels. According to that, learning kernels is equivalent to learning a spectral distribution by optimizing

$$\begin{aligned} &\arg\max_{k \in \mathcal{K}} \sum_{i=1} \mathbb{E}_{x \sim \mathbb{P}_i, x' \sim \mathbb{Q}_i} \left[F_i(x, x') k(x, x') \right] = \\ &\arg\max_{k \in \mathcal{K}} \sum_{i=1} \mathbb{E}_{x \sim \mathbb{P}_i, x' \sim \mathbb{Q}_i} \left[F_i(x, x') \mathbb{E}_{\omega \sim \mathbb{P}_k} \left[e^{i\omega^\top (x-x')} \right] \right], \end{aligned} \tag{3.1}$$

where F is a task-specific objective function and $\mathcal{K}$ is a set of kernels. Eq. (3.1) covers many popular objectives, such as kernel alignment (Gönen and Alpaydın, 2011) and MMD distance (Gretton et al., 2012a). Existing works (Wilson and Adams, 2013; Oliva et al., 2016) learn the spectral density $\mathbb{P}_k(\omega)$ with *explicit* forms via parametric or non-parametric models. When we learn kernels via Eq. (3.1), accessing the density of $\mathbb{P}_k(\omega)$ is not necessary as long as we can estimate kernel evaluations $k(x - x') = \mathbb{E}_\omega[e^{i\omega^\top (x-x')}]$ via sampling from $\mathbb{P}_k(\omega)$ (Rahimi and Recht, 2008). Alternatively, *implicit probabilistic (generative) models* define a stochastic procedure that can generate (sample) data from $\mathbb{P}_k(\omega)$ without modeling $\mathbb{P}_k(\omega)$. Recently, the neural implicit generative models (MacKay, 1995) regained attentions with promising results (Goodfellow et al., 2014) and simple sampling procedures. We first sample ν from a base distribution $\mathbb{P}(\nu)$ (e.g., Gaussian distribution), then use a deterministic function h_ψ parametrized by ψ, to transform ν into $\omega = h_\psi(\nu)$, where ω follows the complex target distribution $\mathbb{P}_k(\omega)$. Inspired by the success of deep implicit generative models (Goodfellow et al., 2014), we propose an *Implicit Kernel Learning (IKL)* method by modeling $\mathbb{P}_k(\omega)$ via an implicit generative model $h_\psi(\nu)$, where $\nu \sim \mathbb{P}(\nu)$, which results in

$$k_\psi(x, x') = \mathbb{E}_\nu \left[e^{i h_\psi(\nu)^\top (x-x')} \right], \tag{3.2}$$

and reducing Eq. (3.1) to solve

$$\arg\max_{\psi} \sum_{i=1} \mathbb{E}_{x \sim \mathbb{P}_i, x' \sim \mathbb{Q}_i} \left[F_i(x, x') \mathbb{E}_\nu \left(e^{i h_\psi(\nu)^\top (x-x')} \right) \right]. \tag{3.3}$$

The gradient of Eq. (3.3) can be represented as

$$\sum_{i=1} \mathbb{E}_{x\sim\mathbb{P}_i, x'\sim\mathbb{Q}_i} \mathbb{E}_\nu \left[\nabla_\psi F_i(x, x') e^{ih_\psi(\nu)^\top (x-x')} \right].$$

Thus, Eq. (3.3) can be optimized via sampling x, x' from data and ν from the base distribution to estimate gradient as shown above (SGD) in every iteration. Next, we discuss the parametrization of h_ψ to satisfy Bochner's Theorem, and describe how to evaluate IKL kernel in practice.

Symmetric $\mathbb{P}_k(\omega)$ To result in real valued kernels, the spectral density has to be symmetric, where $\mathbb{P}_k(\omega) = \mathbb{P}_k(-\omega)$. Thus, we parametrize $h_\psi(\nu) = \texttt{sign}(\nu) \circ \tilde{h}_\psi(\texttt{abs}(\nu))$, where $\circ$ is the Hadamard product and $\tilde{h}_\psi$ can be any unconstrained function if the base distribution $\mathbb{P}(\nu)$ is symmetric (i.e., $\mathbb{P}(\nu) = \mathbb{P}(-\nu)$), such as standard normal distributions.

Kernel Evaluation Although there is usually no closed form for the kernel evaluation $k_\psi(x, x')$ in Eq. (3.2) with fairly complicated h_ψ, we can evaluate (approximate) $k_\psi(x, x')$ via sampling finite number of random Fourier features $\widehat{k}_\psi(x, x') = \widehat{\varphi}_{h_\psi}(x)^\top \widehat{\varphi}_{h_\psi}(x')$, where

$$\widehat{\varphi}_{h_\psi}(x)^\top = [\varphi(x; h_\psi(\nu_1)), \dots, \varphi(x; h_\psi(\nu_m))] \in \mathbb{R}^m$$

and $\varphi(x; \omega)$ is the evaluation on ω of the Fourier transformation $\varphi(x)$ (Rahimi and Recht, 2008).

Next, we demonstrate two example applications covered by Eq. (3.3), where we can apply IKL, including maximum mean discrepancy (MMD) for generative modeling and kernel alignment for supervised learning.

3.3 IKL Application to MMD-GAN

Although the composition kernel with a learned feature embedding f_ϕ is powerful in MMD GAN (e.g., Chapter 2), choosing a good base kernel k is still crucial in practice (Bińkowski et al., 2018). Different base kernels for MMD GAN, such as rational quadratic kernel (Bińkowski et al., 2018) and distance kernel (Bellemare et al., 2017), have been studied. Instead of choosing it by hands, we propose to learn the base kernel by IKL, which extend the compositional kernels to be $k_{\psi,\phi} = k_\psi \circ f_\phi$ with the form

$$k_{\psi,\phi}(x, x') = \mathbb{E}_\nu \left[e^{ih_\psi(\nu)^\top (f_\phi(x) - f_\phi(x'))} \right]. \tag{3.4}$$

We then extend the MMD GAN objective to be

$$\min_\theta \max_{\psi,\phi} M_{\psi,\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta), \tag{3.5}$$

where $M_{\psi,\phi}$ is the MMD distance with the IKL kernel objective (3.4). Clearly, for a given ϕ, the maximization over ψ in Eq. (3.5) can be represented as (3.1) by letting $F_1(x, x') = 1$, $F_2(x, y) = -2$ and $F_3(y, y') = 1$. In wwidehat follows, we will use for convenience $k_{\psi,\phi}$, k_ψ and k_ϕ to denote kernels defined in (3.4), (3.2), and the compositional kernel $k(f_\phi(x), f_\phi(x'))$, respectively.

3.3.1 Property of MMD GAN with IKL

As proven by Arjovsky and Bottou (2017), some probability distances adopted by existing works (e.g. Goodfellow et al. (2014)) are not *weak* (i.e. $\mathbb{P}_n \xrightarrow{D} \mathbb{P}$ then $D(\mathbb{P}_n\|\mathbb{P}) \to 0$), which cannot provide better signal to train g_θ. Also, they usually suffer from discontinuity, hence it cannot be trained via gradient descent at certain points. We extend Theorem 3 and Theorem 4 to prove that $\max_{\psi,\phi} M_{\psi,\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$ is a continuous and differentiable objective in θ and *weak* under mild assumptions based the following Assumption.

Assumption 7. *$g_\theta(z)$ is locally Lipschitz and differentiable in θ; $f_\phi(x)$ is Lipschitz in x and $\phi \in \Phi$ is compact. $f_\phi \circ g_\theta(z)$ is differentiable in θ and there are local Lipschitz constants, which is independent of ϕ, such that $\mathbb{E}_{z\sim\mathbb{P}_z}[L(\theta, z)] < +\infty$. The above assumptions are adopted by Arjovsky et al. (2017). Lastly, assume given any $\psi \in \Psi$, where Ψ is compact, $k_\psi(x, x') = \mathbb{E}_\nu\left[e^{ih_\psi(\nu)^\top(x-x')}\right]$ and $|k_\psi(x, x')| < \infty$ is differentiable and Lipschitz in (x, x') which has an upper bound L_k for Lipschitz constant of (x, x') given different ψ.*

Theorem 8. *Assume function g_θ and kernel $k_{\psi,\phi}$ satisfy Assumption 7, $\max_{\psi,\phi} M_{\psi,\phi}$ is weak, that is, $\max_{\psi,\phi} M_{\psi,\phi}(\mathbb{P}_\mathcal{X}, \mathbb{P}_n) \to 0 \iff \mathbb{P}_n \xrightarrow{D} \mathbb{P}_\mathcal{X}$. Also, $\max_{\psi,\phi} M_{\psi,\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$ is continuous everywhere and differentiable almost everywhere in θ.*

Lemma 9. *Assume $\mathcal{X}$ is bounded. Let $x, x' \in \mathcal{X}$, $k_\psi(x, x') = \mathbb{E}_\nu\left[e^{ih_\psi(\nu)^\top(x-x')}\right]$ is Lipschitz in (x, x') if $\mathbb{E}_\nu\left[\|h_\psi(\nu)\|^2\right] < \infty$.*

Note that $\mathbb{E}_\nu[\|h_\psi(\nu)\|^2]$ is the variance since $\mathbb{E}_\nu[h_\psi(\nu)] = 0$. We penalize $\lambda_h(\mathbb{E}_\nu\left[\|h_\psi(\nu)\|^2\right] - u)^2$ as an approximation of Lemma 9 in practice to ensure that assumptions in Theorem 8 are satisfied. The algorithm with IKL and gradient penalty (Bińkowski et al., 2018) is shown in Algorithm 2.

Algorithm 2: MMD GAN with IKL

Input: η the learning rate, B the batch size, n_c number of f, h updates per g update, m the number of basis, λ_{GP} the coefficient of gradient penalty, λ_h the coefficient of variance constraint.

Initial: parameter θ for g, ϕ for f, ψ for h

Define: $\mathcal{L}(\psi, \phi) = M_{\psi,\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta) - \lambda_{GP}(\|\nabla_{\widehat{x}} f_\phi(\widehat{x})\|_2 - 1)^2 - \lambda_h(\mathbb{E}_\nu[\|h_\psi(\nu)\|^2] - u)^2$

while *θ has not converged* **do**

  for $t = 1, \ldots, n_c$ **do**

    Sample $\{x_i\}_{i=1}^B \sim \mathbb{P}(\mathcal{X}), \{z_j\}_{j=1}^B \sim \mathbb{P}(\mathcal{Z}), \{\nu_k\}_{k=1}^m \sim \mathbb{P}(\nu)$

    $(\psi, \phi) \leftarrow \phi + \eta\text{Adam}\Big((\psi, \phi), \nabla_{\psi,\phi}\mathcal{L}(\psi, \phi)\Big)$

  Sample $\{x_i\}_{i=1}^B \sim \mathbb{P}(\mathcal{X}), \{z_j\}_{j=1}^B \sim \mathbb{P}(\mathcal{Z}), \{\nu_k\}_{k=1}^m \sim \mathbb{P}(\nu)$

  $\theta \leftarrow \theta - \eta\text{Adam}\Big(\theta, \nabla_\theta M_{\psi,\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)\Big)$

3.3.2 Experiments and Results

We consider image and text generation tasks for quantitative evaluation. For image generation, we evaluate the inception score (Salimans et al., 2016) and FID score (Heusel et al., 2017) on CIFAR-10 (Krizhevsky and Hinton, 2009). We use DCGAN (Radford et al., 2016) and expands the output

of f_ϕ to be 16-dimensional as Bińkowski et al. (2018). For text generation, we consider a length-32 character-level generation task on Google Billion Words dataset. The evaluation is based on Jensen-Shannon divergence on empirical 4-gram probabilities (JS-4) of the generated sequence and the validation data as used by Gulrajani et al. (2017); Heusel et al. (2017); Mroueh et al. (2018). The model architecture follows Gulrajani et al. (2017) in using ResNet with 1D convolutions. We train every algorithm $10,000$ iterations for comparison.

For MMD GAN with fixed base kernels, we consider the mixture of Gaussian kernels $k(x, x') = \sum_q \exp(-\frac{\|x-x'\|^2}{2\sigma_q^2})$ (Li et al., 2017) and the mixture of RQ kernels $k(x, x') = \sum_q (1 + \frac{\|x-x'\|^2}{2\alpha_q})^{-\alpha_q}$. We tuned hyperparameters σ_q and α_q for each kernel as reported in Appendix 3.6.4.

Lastly, for learning base kernels, we compare IKL with SM kernel (Wilson and Adams, 2013) f_ϕ, which learns mixture of Gaussians to model kernel spectral density. It can also be treated as the *explicit generative model counter part* of the proposed IKL.

In both tasks, $\mathbb{P}(\nu)$, the base distribution of IKL, is a standard normal distribution and h_ψ is a 3-layer MLP with 32 hidden units for each layer. Similar to the aforementioned mixture kernels, we consider the mixture of IKL kernel with the variance constraints $\mathbb{E}[\|h_\psi(\nu)\|^2] = 1/\sigma_q$, where σ_q is the bandwidths for the mixture of Gaussian kernels. Note that if h_ψ is an identity map, we recover the mixture of Gaussian kernels. We fix λ_h to be 10 and resample $m = 1024$ random features for IKL in every iteration. For other settings, we follow Bińkowski et al. (2018) and the hyperparameters can be found in Appendix 3.6.4.

Quantitative and Qualitative Results We compare MMD GAN with the proposed IKL and different fixed kernels. We repeat the experiments 10 times and report the average result with standard error in Table 3.1. Note that for inception score the larger the better; while JS-4 the smaller the better. We also report WGAN-GP results as a reference. Sampled images on larger datasets are shown in Figure 3.1.

Method	Inception Scores (↑)	FID Scores (↓)	JS-4 (↓)
Gaussian	6.726 ± 0.021	32.50 ± 0.07	0.381 ± 0.003
RQ	6.785 ± 0.031	32.20 ± 0.09	0.463 ± 0.005
SM	6.746 ± 0.031	32.43 ± 0.08	0.378 ± 0.003
IKL	$\mathbf{6.876 \pm 0.018}$	$\mathbf{31.98 \pm 0.05}$	$\mathbf{0.372 \pm 0.002}$
WGAN-GP	6.539 ± 0.034	36.413 ± 0.05	0.379 ± 0.002

Table 3.1: Inception scores, FID scores, and JS-4 divergece results.

Bińkowski et al. (2018) show RQ kernels outperform Gaussian and energy distance kernels on image generation. Our empirical results agree with such finding: RQ kernels achieve 6.785 inception score while for Gaussian kernel it is 6.726, as shown in the left column of Table 3.1. In text generation, nonetheless, RQ kernels only achieve 0.463 JS-4 score[1] and are not on par with 0.381 acquired by Gaussian kernels, even though it is still slightly worse than WGAN-GP. These results imply *kernel selection is task-specific*. On the other hand, the proposed IKL learns kernels in a data-driven way, which results in the best

[1]For RQ kernels, we searched 10 possible hyperparameter settings and reported the best one in Appendix, to ensure the unsatisfactory performance is not caused by the improper parameters.

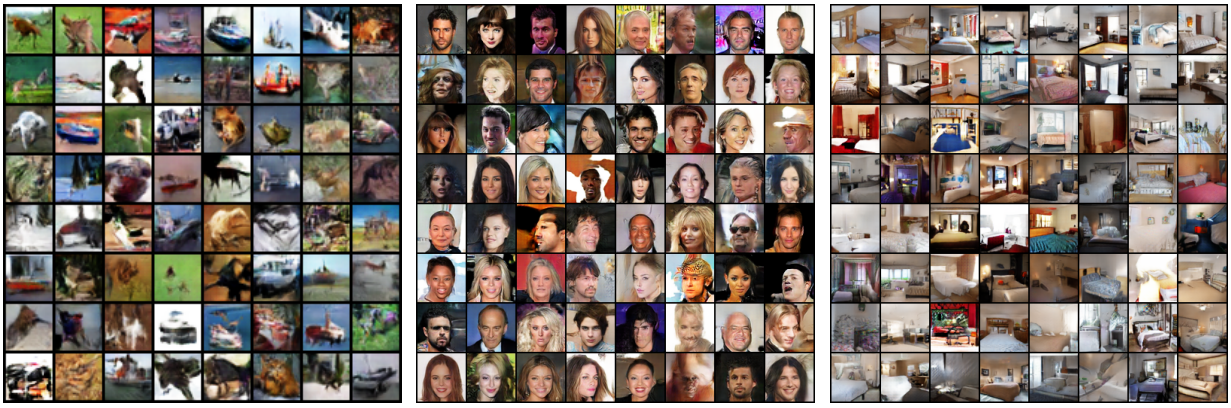


Figure 3.1: Samples generated by MMDGAN-IKL on CIFAR-10, CELEBA and LSUN dataset.

performance in both tasks. In CIFAR-10, although Gaussian kernel is worse than RQ, IKL is still able to transforms $\mathbb{P}(\nu)$, which is Gaussian, into a powerful kernel, and outperforms RQ on inception scores (6.876 v.s. 6.785). For text generation, from Table 3.1 and Figure 3.2, we observe that IKL can further boost Gaussian into better kernels with substantial improvement. Also, we note that the difference between IKL and pre-defined kernels in Table 3.1 is significant based on the t-test at 95% confidence level.

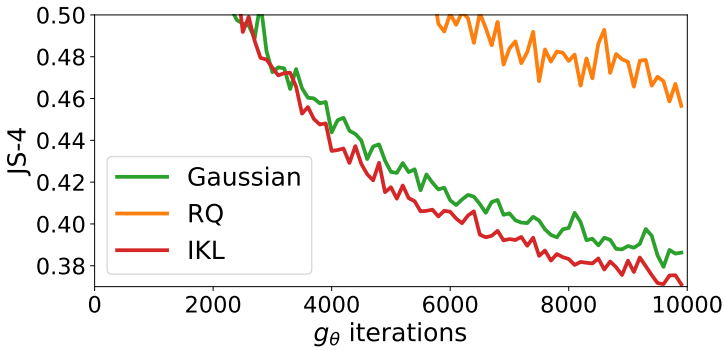


Figure 3.2: Convergence of MMD GANs with different kernels on text generation.

The SM kernel (Wilson and Adams, 2013), which learns the spectral density via mixture of Gaussians, does not significantly outperforms Gaussian kernel as shown in Table 3.1, since Li et al. (2017) already uses equal-weighted mixture of Gaussian formulation. It suggests that proposed IKL can learn more complicated and effective spectral distributions than simple mixture models.

Study of Variance Constraints In Lemma 9, we prove bounding variance $\mathbb{E}[\|h_\psi(\nu)\|^2]$ guarantees k_ψ to be Lipschitz as required in Theorem 8. We investigate the importance of this constraint. In Figure 3.3, we show the training objective (MMD), $\mathbb{E}[\|h_\psi(\nu)\|^2]$ and the JS-4 divergence for training MMD GAN (IKL) without variance constraint, i.e. $\lambda_h = 0$. We could observe the variance keeps going up without constraints, which leads exploded MMD values. Also, when the exploration is getting severe, the JS-4 divergence starts increasing, which implies MMD cannot provide meaningful signal to g_θ. The study justifies the validity of Theorem 8 and Lemma 9.

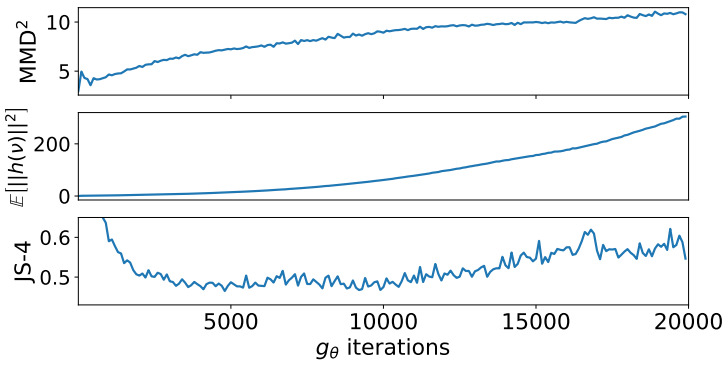


Figure 3.3: Learning MMD GAN (IKL) without the variance constraint on Google Billion Words datasets for text generation.

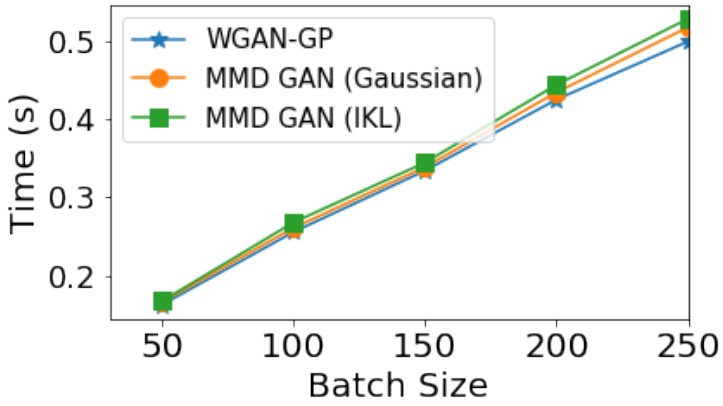


Figure 3.4: Computational time per one generator step with different batch sizes.

In Section 3.3, we propose to constrain variance via $\lambda_h(\mathbb{E}_\nu \left[\|h_\psi(\nu)\|^2\right] - u)^2$. There are other alternatives, such as constraining L^2 penalty or using Lagrange. In practice, we do not observe significant difference. Although we show the necessity of the variance constraint in language generation in Figure 3.3, we remark that the proposed constraint is a sufficient condition. For CIFAR-10, without the constraint, we observe that the variance is still bouncing between 1 and 2 without explosion as Figure 3.3. Therefore, the training leads to a satisfactory result with 6.731 ± 0.034 inception score, but it is slightly worse than IKL in Table 3.1. The necessary or weaker sufficient conditions are worth further studying as a future work.

Model Capacity and Computational Time One concern of the proposed IKL is the computational overhead introduced by sampling random features as well as using more parameters to model the h_ψ. For f_ϕ, the number of parameters for DCGAN is around 0.8 million for size 32×32 images and 3 millions for size 64×64 images. The ResNet architecture used in Gulrajani et al. (2017) has around 10 millions parameters. In contrast, in all experiments, we use simple three layer MLP as h_ψ for IKL, where the input and output dimensions are 16, and hidden layer size is 32. The total parameters are just around 2,000. Compared with f_ϕ, the additional number of parameters used for h_ψ is almost negligible.

The other concern of IKL is sampling random features for each examples. See Figure 3.4 for results, where we use $m = 1024$ random features for each iteration. We measure the time per iteration of updating critic iterations (f for WGAN-GP and MMD GAN with Gaussian kernel; f and h for IKL) with different batch sizes under Titan X. The difference between WGAN-GP, MMD GAN and IKL are not significant. The reason is computing MMD and random feature is highly parallelizable, and other computation, such as evaluating f_ϕ and its gradient penalty, dominates the cost because f_ϕ has much more parameters as aforementioned. Therefore, we believe the proposed IKL is still cost effective in practice.

IKL with and without Neural Networks on GAN training Instead of learning a transform function h_ψ for the spectral distribution as we proposed (namely, IKL-NN), the other realization of IKL is to keep a pool of finite number learned random features $\Omega = \{\widehat{\omega}_i\}_{i=1}^m$, and approximate the kernel evaluation by $\hat{k}_\Omega(x, x') = \widehat{\varphi}_\Omega(x)^\top \widehat{\varphi}_\Omega(x')$, where $\widehat{\varphi}_\Omega(x)^\top = [\varphi(x; \widehat{\omega}_1), \dots, \varphi(x; \widehat{\omega}_m)]$. During the learning, it directly optimize $\widehat{\omega}_i$. Many existing works study this idea for supervised learning, such as Băzăvan et al. (2012); Yang et al. (2015); Sinha and Duchi (2016); Chang et al. (2017a); Bullins et al. (2018). We call the latter realization as IKL-RFF. Next, we discuss and compare the difference between IKL-NN and IKL-RFF.

The crucial difference between IKL-NN and IKL-RFF is, IKL-NN can sample arbitrary number of random features by first sampling $\nu \sim \mathbb{P}(\nu)$ and transforming it via $h_\psi(\nu)$, while IKL-RFF is restricted by the pool size m. If the application needs more random features, IKL-RFF will be memory inefficient. Specifically, we compare IKL-NN and IKL-RFF with different number of random features in Figure 3.5. With the same number of parameters (i.e., $|h_\psi| = m \times dim(\nu)$)[2] , IKL-NN outperforms IKL-RFF of $m = 128$ on Inception scores (6.876 versus 6.801). For IKL-RFF to achieve the same or better Inception scores of IKL-NN, the number of random features m needs increasing to 4096, which is less memory efficient than the IKL-NN realization. In particular, h_ψ of IKL-NN is a three-layers MLP with 2048 number of parameters ($16 \times 32 + 32 \times 32 + 32 \times 16$), while IKL-RFF has $2048, 65536$ number of parameters, for $m = 128, 4096$, respectively.

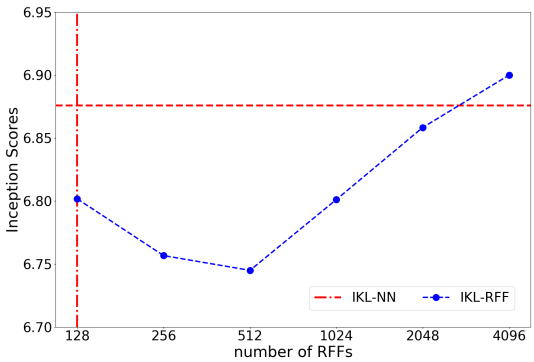


Figure 3.5: The comparison between IKL-NN and IKL-RFF on CIFAR-10 under different number of random features.

[2] $|h_\psi|$ denotes number of parameters in h_ψ, m is number of random features and $dim(\nu)$ is the dimension of the ν.

Algorithm	JS-4
IKL-NN	0.372 ± 0.002
IKL-RFF	0.383 ± 0.002
IKL-RFF (+2)	0.380 ± 0.002
IKL-RFF (+4)	0.377 ± 0.002
IKL-RFF (+8)	0.375 ± 0.002

Table 3.2: The comparison between IKL-NN and IKL-RFF on Google Billion Word.

On the other hand, using large m for IKL-RFF not only increases the number of parameters, but might also enhance the optimization difficulty. Zhang et al. (2017) discuss the difficulty of optimizing RFF directly on different tasks. Here we compare IKL-NN and IKL-RFF on challenging Google Billion Word dataset. We train IKL-RFF with the same setting as Section 3.3, where we set the pool size m to be 1024 and the updating schedule between critic and generator to be $10:1$, but we tune the Adam optimization parameter for IKL-RFF for fair comparison. As discussed above, please note that the number of parameters for h_ψ is 2048 while IKL-RFF uses 16384 when $m = 1024$. The results are shown in Table 3.2. Even IKL-RFF is using more parameters, the performance 0.383 is not competitive as IKL-NN, which achieves 0.372.

In Algorithm 2, we update f_φ and h in each iteration with n_c times, where we use $n_c = 10$ here. We keep the number of updating f_φ to be 10, but increase the number of update for $\{\hat{\omega}_i\}_{i=1}^{1}024$ to be $12, 14, 18$ in each iteration. The result is shown in Table 3.2 with symbols +2, +4 and +8 respectively. Clearly, we see IKL-RFF need more number of updates to achieve competitive performance with IKL-NN. The results might implies IKL-RFF is a more difficult optimization problem with more parameters than IKL-NN. It also confirms the effectiveness of learning implicit generative models with deep neural networks (Goodfellow et al., 2014), but the underlying theory is still an open research question. A better optimization algorithm (Zhang et al., 2017) may improve the performance gap between IKL-NN and IKL-RFF, which worth more study as future work.

3.4 IKL Application to Random Kitchen Sinks

We also show that the proposed IKL framework can benefit conventional kernel methods in supervised learning. Rahimi and Recht (2009) propose *Random Kitchen Sinks* (RKS) as follows. We sample $\omega_i \sim \mathbb{P}_k(\omega)$ and transform $x \in \mathbb{R}^d$ into

$$\widehat{\varphi}(x) = [\varphi(x;\omega_1), \ldots, \varphi(x;\omega_M)], \quad \text{where } \sup_{x,\omega} |\varphi(x;\omega)| < 1.$$

We then learn a classifier on the transformed features $\widehat{\varphi}(x;\omega)$. Kernel methods with random Fourier features (Rahimi and Recht, 2008) is an example of RKS, where $\mathbb{P}_k(\omega)$ is the spectral distribution of the kernel and $\varphi(x;\omega) = \big[\cos(\omega^\top x), \sin(\omega^\top x)\big]$. We usually learn a model $\mathbf{w}$ by solving

$$\arg\min_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell\left(\mathbf{w}^\top \widehat{\varphi}(x_i)\right). \tag{3.6}$$

If ℓ is a convex loss function, the objective (3.6) can be solved efficiently to global optimum.

Spectral distributions $\mathbb{P}_k$ are usually set as a parameterized form, such as Gaussian distributions, but the selection of $\mathbb{P}_k$ is important in practice. If we consider RKS as kernel methods with random features, then selecting $\mathbb{P}$ is equivalent to the well-known kernel selection (learning) problem for supervised learning (Gönen and Alpaydın, 2011).

Two-Stage Approach We follows Sinha and Duchi (2016) to consider kernel learning for RKS with a two-stage approach. In stage 1, we consider kernel alignment (Cristianini et al., 2002) of the form, $\operatorname{argmax}_{k\in\mathcal{K}} \mathbb{E}_{(x,y),(x',y')} \sum_{i\neq j} yy'k(x,x')$. By parameterizing k via the implicit generative model h_ψ as in Section 3.2, we have the following problem:

$$\operatorname*{argmax}_{\psi} \mathbb{E}_{(x,y),(x',y')} yy' \mathbb{E}_{\nu} \left[e^{ih_\psi(\nu)^\top (x-x')}, \right]. \tag{3.7}$$

which can be treated as (3.1) with $F_1(x,x') = yy'$. After solving (3.7), we learn a *sampler* h_ψ where we can easily sample. Thus, in stage 2, we thus have the advantage of solving a convex problem (3.6) in RKS with IKL. The algorithm is shown in Algorithm 3.

Algorithm 3: Random Kitchen Sinks with IKL

Stage 1: Kernel Learning

Input: $X = \{(x_i, y_i)\}_{i=1}^n$, the batch size B for data and m for random feature, learning rate η

Initialization: ψ for transformation function h

while ψ *has not converged* **do**

Sample $\{(x_i, y_i)\}_{i=1}^B \subseteq X$ and sample $\{\nu_j\}_{j=1}^m \sim \mathbb{P}(\nu)$

$g_\psi \leftarrow \nabla_\psi \frac{1}{B(B-1)} \sum_{i\neq i'} y_i y_{i'} \frac{1}{m} \sum_{j=1}^m e^{ih_\psi(\nu_j)^\top (x_i - x_{i'}))}$

$\psi \leftarrow \psi - \eta \text{Adam}(\psi, g_\psi)$

Stage 2: Random Kitchen Sinks

Sample $\{\nu_i\}_{i=1}^M \sim \mathbb{P}(\nu)$, note that M is not necessarily equal to m

Transform X into $\varphi(X)$ via h_ψ and $\{\nu_i\}_{i=1}^M$

Learn a linear classifier on $(\varphi(X), Y)$

Note that in stage 1, we resample $\{\nu_j\}_{j=1}^m$ in every iteration to train an implicit generative model h_ψ. The advantage of Algorithm 3 is the random features used in kernel learning and RKS can be *different*, which allows us to use *less* random features in kernel learning (stage 1), and sample *more* features for RKS (stage 2).

One can also *jointly* train both feature mapping ω and the model parameters $\mathbf{w}$, such as neural networks. We remark that our intent is not to show state-of-the-art results on supervised learning, on which deep neural networks dominate (Krizhevsky et al., 2012; He et al., 2016). We use RKS as a protocol to study kernel learning and the proposed IKL, which still has competitive performance with neural networks on some tasks (Rahimi and Recht, 2009; Sinha and Duchi, 2016). Also, the simple procedure of RKL with IKL allows us to provide some theoretical guarantees of the performance, which is sill challenging of deep learning models.

Comparison with Existing Works Sinha and Duchi (2016) learn non-uniform weights for M random features via kernel alignment in stage 1 then using these optimized features in RKS in the stage 2. Note that the random features used in stage 1 has to be the same as the ones in stage 2. A jointly training of feature mapping and classifier can be treated as a 2-layer neural networks (Băzăvan et al., 2012; Alber et al., 2017; Bullins et al., 2018). Learning kernels with aforementioned works will be more costly if we want to use a large number of random features for training classifiers. In contrast to implicit generative models, Oliva et al. (2016) learn an explicit Bayesian nonparametric generative model for spectral distributions, which requires specifically designed inference algorithms. Learning kernels for (3.6) in dual form without random features has also been proposed. It usually require costly steps, such as eigendecomposition of the Gram matrix (Gönen and Alpaydın, 2011).

3.4.1 Consistency and Generalization

The simple two-stages approach, IKL with RKS, allows us to provide the consistency and generalization guarantees. For consistency, it guarantees the solution of finite sample approximations of (3.7) approach to the optimum of (3.7) (population optimum), when we increase number of training data and number of random features. We firstly define necessary symbols and state the theorem.

Let $s(x_i, x_j) = y_i y_j$ be a label similarity function, where $|y_i| \leq 1$. We use s_{ij} to denote $s(x_i, x_j)$ interchangeably. Given a kernel k, we define the true and empirical alignment functions as,

$$\begin{aligned} T(k) &= \mathbb{E}\left[s(x, x')k(x, x')\right] \\ \widehat{T}(k) &= \tfrac{1}{n(n-1)} \textstyle\sum_{i \neq j} s_{ij} k(x_i, x_j). \end{aligned}$$

In the following, we abuse the notation k_ψ to be k_h for ease of illustration. Recall the definitions of $k_h(x, x') = \langle \varphi_h(x), \varphi_h(x') \rangle$ and $\widehat{k}_h(x, x') = \widehat{\varphi}_h(x)^\top \widehat{\varphi}_h(x')$. We define two hypothesis sets

$$\begin{aligned} \mathcal{F}_\mathcal{H} &= \{f(x) = \langle w, \varphi_h(x) \rangle_H | h \in \mathcal{H}, \langle w, w \rangle \leq 1\} \\ \widehat{\mathcal{F}}^m_\mathcal{H} &= \{f(x) = w^\top \widehat{\varphi}_h(x) | h \in \mathcal{H}, \|w\| \leq 1, w \in \mathbb{R}^m\}. \end{aligned}$$

Definition 10. *(Rademacher's Complexity) Given a hypothesis set $\mathcal{F}$, where $f : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ if $f \in \mathcal{F}$, and a fixed sample $X = \{x_1, \ldots, x_n\}$, the empirical Rademacher's complexity of $\mathcal{F}$ is defined as*

$$\mathfrak{R}^n_X(\mathcal{F}) = \frac{1}{n} \mathbb{E}_\sigma \left[\sup_{f \in \mathcal{F}} \sum_{i=1}^n \sigma_i f(x_i) \right],$$

where σ are n i.i.d. Rademacher random variables.

We then have the following theorems showing that the consistency guarantee depends on the complexity of the function class induced by IKL as well as the number of random features. The proof can be found in Appendix 3.6.3.

Theorem 11. *(Consistency) Let $\widehat{h} = \arg\max_{h \in \mathcal{H}} \widehat{T}(\widehat{k}_h)$, with i.i.d, samples $\{\nu_i\}_{i=1}^m$ drawn from $\mathbb{P}(\nu)$. With probability at least $1 - 3\delta$, we have $|T(\widehat{k}_{\widehat{h}}) - \sup_{h \in \mathcal{H}} T(k_h)| \leq$*

$$2\mathbb{E}_X \left[\mathfrak{R}^{n-1}_X(\mathcal{F}_\mathcal{H}) + \mathfrak{R}^{n-1}_X(\widehat{\mathcal{F}}^m_\mathcal{H}) \right] + \sqrt{\frac{8 \log \frac{1}{\delta}}{n}} + \sqrt{\frac{2 \log \frac{4}{\delta}}{m}}.$$

Applying Cortes et al. (2010a), We also have a generalization bound, which depends number of training data n, number of random features m and the Rademacher complexity of IKL kernel, as shown below.

Theorem 12. *(Generalization (Cortes et al., 2010a)) Define the true and empirical misclassification for a classifier f as $R(f) = \mathbb{P}(Yf(X) < 0)$ and $\widehat{R}_\gamma(h) = \frac{1}{n}\sum_{i=1}^n \min\{1, [1 - yf(x_i)/\gamma]_+\}$. Then*

$$\sup_{f \in \widehat{\mathcal{F}}_{\mathcal{H}}} \{R(f) - \widehat{R}_\gamma(f)\} \leq \frac{2}{\gamma}\mathfrak{R}_X^n(\widehat{\mathcal{F}}_{\mathcal{H}}) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2n}}$$

with probability at least $1 - \delta$.

The Rademacher complexity $\mathfrak{R}_X^n(\mathcal{F}_{\mathcal{H}})$, for example, can be $1/\sqrt{n}$ or even $1/n$ for kernels with different bounding conditions (Cortes et al., 2010a). We would expect worse rates for more powerful kernels. It suggests the trade-off between consistency/generalization and using powerful kernels parametrized by neural networks.

3.4.2 Experiment and Results

We evaluate the proposed IKL on both synthetic and benchmark binary classification tasks. For IKL, $\mathbb{P}(\nu)$ is standard Normal and h_ψ is a 3-layer MLP for all experiments. The number of random features m to train h_ψ in Algorithm 3 is fixed to be 64. Other experiment details are described in Appendix 3.6.4.

Kernel learning with a poor choice of $\mathbb{P}_k(\omega)$ We generate $\{x_i\}_{i=1}^n \sim \mathcal{N}(0, I_d)$ with $y_i = \text{sign}(\|x\|_2 - \sqrt{d})$, where d is the data dimension. A two dimensional example is shown in Figure 3.6. Competitive baselines include random features (RFF) (Rahimi and Recht, 2008) as well as OPT-KL (Sinha and Duchi, 2016). In the experiments, we fix $M = 256$ in RKS for all algorithms. Since Gaussian kernels with the bandwidth $\sigma = 1$ is known to be ill-suited for this task (Sinha and Duchi, 2016), we directly use random features from it for RFF and OPT-KL. Similarly, we set $\mathbb{P}(\nu)$ to be standard normal distribution as well.

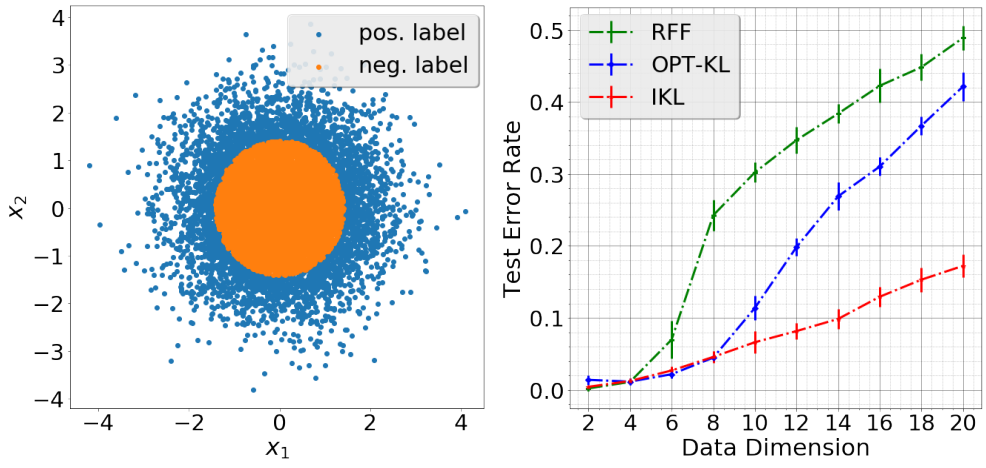
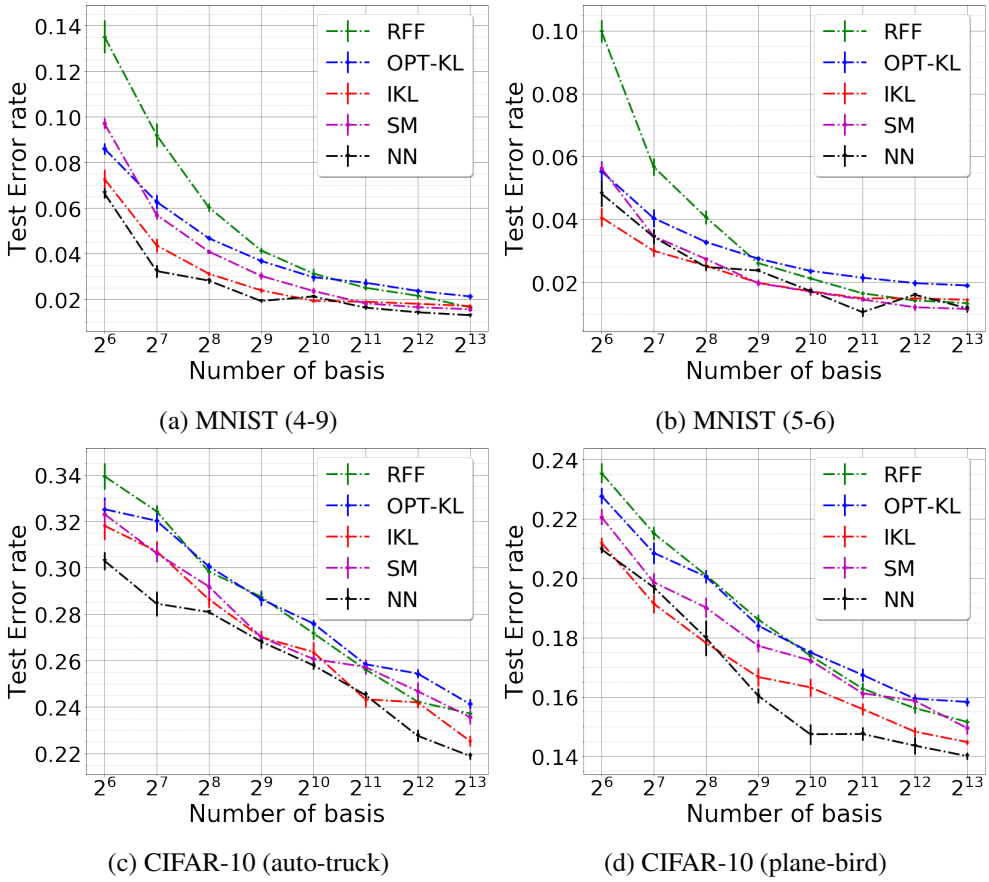


Figure 3.6: Left: the training examples when $d = 2$. Right: the classification error v.s. data dimension.

The test error for different data dimension $d = \{2, 4, \ldots, 18, 20\}$ is shown in Figure 3.6. Note that RFF is competitive with OPT-KL and IKL when d is small ($d \leq 6$), while its performance degrades rapidly as d increases, which is consistent with the observation in Sinha and Duchi (2016). More discussion of the reason of failure can be referred to Sinha and Duchi (2016). On the other hand, although using standard normal as the spectral distribution is ill-suited for this task, both OPT-KL and IKL can adapt with data and learn to transform it into effective kernels and result in slower degradation with d.

Note that OPT-KL learns the *sparse* weights on the sampled random features ($M = 256$). However, the sampled random features can fail to contain informative ones, especially in high dimension (Bullins et al., 2018). Thus, when using limited amount of random features, OPT-IKL may result in worse performance than IKL in the high dimensional regime in Figure 3.6.



(a) MNIST (4-9)

(b) MNIST (5-6)

(c) CIFAR-10 (auto-truck)

(d) CIFAR-10 (plane-bird)

Figure 3.7: Test error rate versus number of basis in second stage on benchmark binary classification tasks. We report mean and standard deviation over five runs. Our method (IKL) is compared with RFF (Rahimi and Recht, 2009), OPT-KL (Sinha and Duchi, 2016), SM (Wilson and Adams, 2013) and the end-to-end training MLP (NN).

Performance on benchmark datasets Next, we evaluate our IKL framework on standard benchmark binary classification tasks. Challenging label pairs are chosen from MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky and Hinton, 2009) datasets; each task consists of 10000 training and 2000 test

examples. For all datasets, raw pixels are used as the feature representation. We set the bandwidth of RBF kernel by the median heuristic. We also compare with Wilson and Adams (2013), the spectral mixture (SM) kernel, which uses Gaussian mixture to learn spectral density and can be seen as the explicit generative model counterpart of IKL. Also, SM kernel is a MKL variant with linear combination (Gönen and Alpaydın, 2011). In addition, we consider the joint training of random features and model parameters, which can be treated as two-layer neural network (NN) and serve as the lower bound of error for comparing different kernel learning algorithms.

The test error versus different $M = \{2^6, 2^7, \ldots, 2^{13}\}$ in the second stage are shown in Figure 3.7. First, in light of computation efficiency, SM and the proposed IKL only sample $m = 64$ random features in each iteration in the first stage, and draws different number of basis M from the learned $h_\psi(\nu)$ for the second stage. OPT-KL, on the contrary, the random features used in training and testing should be the same. Therefore, OPT-IKL needs to deal with M random features in the training. It brings computation concern when M is large. In addition, IKL demonstrates improvement over the representative kernel learning method OPT-KL, especially significant on the challenging datasets such as CIFAR-10. In some cases, IKL almost reaches the performance of NN, such as MNIST, while OPT-KL degrades to RFF except for small number of basis ($M = 2^6$). This illustrates the effectiveness of learning kernel spectral distribution via the implicit generative model h_ψ. Also, IKL outperforms SM, which is consistent with the finding in Section 3.3 that IKL can learn more complicated spectral distributions than simple mixture models (SM).

3.5 Summary

We propose a generic kernel learning algorithm, IKL, which learns sampling processes of kernel spectral distributions by transforming samples from a base distribution $\mathbb{P}(\nu)$ into ones for the other kernel (spectral density). We compare IKL with other algorithms for learning MMD GAN and supervised learning with Random Kitchen Sinks (RKS). For these two tasks, the conditions and guarantees of IKL for are studied. Empirical studies show IKL is better than or competitive with the state-of-the-art kernel learning algorithms. It proves IKL can learn to transform $\mathbb{P}(\nu)$ into effective kernels even if $\mathbb{P}(\nu)$ is less favorable to the task.

We note that the preliminary idea of IKL is mentioned in Băzăvan et al. (2012), but they ended up with a algorithm that directly optimizes sampled random features (RF), which has many follow-up works (e.g. Sinha and Duchi (2016); Bullins et al. (2018)). The major difference is, by learning the transformation function h_ψ, the RF used in training and evaluation can be different. This flexibility allows a simple training algorithm (SGD) and does not require to keep learned features. In our studies on GAN and RKS, we show using a simple MLP can already achieve better or competitive performance with those works, which suggest IKL can be a new direction for kernel learning and worth more studies.

We highlight that IKL is not conflict with existing works but can be combined with them. In Section 3.3, we show combining IKL with kernel learning via embedding (Wilson et al., 2016) and mixture of spectral distributions (Wilson and Adams, 2013). Therefore, in addition to the examples shown in Section 3.3 and Section 3.4, IKL is directly applicable to many existing works with kernel learning via embedding (e.g. Dai et al. (2014); Li and Póczos (2016); Wilson et al. (2016); Al-Shedivat et al. (2017);

Arbel et al. (2018); Jean et al. (2018); Chang et al. (2019a)). A possible extension is combining with Bayesian inference (Oliva et al., 2016) under the framework similar to Saatchi and Wilson (2017). The learned sampler from IKL can possibly provide an easier way to do Bayesian inference via sampling.

3.6 Appendix

3.6.1 Proof of Theorem 8

The proof is a straightforward extension of Section 2.6.1 and 2.6.2. We still include the full derivation for completeness.

First, We need to show $\|k_\psi(f_\phi(x),\cdot) - k_\psi(f_\phi(y),\cdot)\|_{\mathcal{H}_K}$ is Lipschitz. By definition, we know that $\|k_\psi(f_\phi(x),\cdot) - k_\psi(f_\phi(y),\cdot)\|_{\mathcal{H}_K} = 2(1 - k_\psi(f_\phi(x), f_\phi(y)))$. Also, since $k_\psi(0) = 1$ and $k_\psi(0) - k_\psi(x, x') \leq L_k\|0 - (x - x')\|$ (Lipschtiz assumption of k_ψ), we have

$$\|k_\psi(f_\phi(x),\cdot) - k_\psi(f_\phi(y),\cdot)\|_{\mathcal{H}_K} \leq 2L_k\|f_\phi(x) - f_\phi(y)\| \leq 2L_kL\|x - y\|,$$

where the last inequality holds as f_ϕ is also a Lipschitz function with Lipschitz constant L.

The other direction, $\max_{\psi,\phi} M_{\psi,\phi}(\mathbb{P},\mathbb{P}_n) \to 0$ then $\mathbb{P}_n \xrightarrow{D} \mathbb{P}$, is relatively simple. We assume there exists ψ' and ϕ' such that h_ψ and f_ϕ are identity functions (up to scaling), which recover the Gaussian kernel k. Therefore, $\max_{\psi,\phi} M_{\psi,\phi}(\mathbb{P},\mathbb{P}_n) \to 0$ implies $M_{\psi',\phi'}(\mathbb{P},\mathbb{P}_n) \to 0$, which completes the proof because MMD with any Gaussian kernel is weak (Gretton et al., 2012a).

Next, we show the proof of continuity. We are going to show

$$\max_{\psi,\phi} M_{\psi,\phi}(\mathbb{P}_\mathcal{X},\mathbb{P}_\theta) = \mathbb{E}_{x,x'}[k_{\psi,\phi}(x,x')] - 2\mathbb{E}_{x,z}[k_{\psi,\phi}(x,g_\theta(z))] + \mathbb{E}_{z,z'}[k_{\psi,\phi}(g_\theta(z'),g_\theta(z))] \tag{3.8}$$

is differentiable with respect to ϕ almost everywhere by using the auxiliary Lemma 6. We fist show $\mathbb{E}_{z,z'}[k_{\psi,\phi}(g_\theta(z'),g_\theta(z))]$ in (3.8) is locally Lipschitz in θ.

$$\begin{aligned}
&\mathbb{E}_{x,x'}\Big[k_{\psi,\phi}\Big(g_\theta(z),g_\theta(z')\Big) - k_{\psi,\phi}\Big(g_{\theta'}(z),g_{\theta'}(z')\Big)\Big]\\
=&\mathbb{E}_{z,z'}\Big[k_\psi\Big(f_\phi\big(g_\theta(z)\big) - f_\phi\big(g_\theta(z')\big)\Big)\Big] - \mathbb{E}_{z,z'}\Big[k_\psi\Big(f_\phi\big(g_{\theta'}(z)\big) - f_\phi\big(g_{\theta'}(z')\big)\Big)\Big]\\
\leq&\mathbb{E}_{z,z'}\Big[L_k\Big\|f_\phi\big(g_\theta(z)\big) - f_\phi\big(g_\theta(z')\big) - f_\phi\big(g_{\theta'}(z)\big) + f_\phi\big(g_{\theta'}(z')\big)\Big\|\Big]\\
\leq&\mathbb{E}_{z,z'}\Big[L_kL(\theta,z)\|\theta - \theta'\| + L_kL(\theta,z')\|\theta - \theta'\|\Big]\\
=&2L_k\mathbb{E}_z\big[L(\theta,z)\big]\|\theta - \theta'\|.
\end{aligned}$$

The first inequality is followed by the assumption that k is locally Lipschitz in (x, x'), with a upper bound L_k for Lipschitz constants. By Assumption 7, $\mathbb{E}_z\big[L(\theta,z)\big] < \infty$, we prove $\mathbb{E}_{z,z'}\Big[k_\psi\big(f_\phi(g_\theta(z)) - f_\phi(g_\theta(z'))\big)\Big]$ is locally Lipschitz. The similar argument is applicable to other terms in Eq. (3.8); therefore, Eq. (3.8) is locally Lipschitz in θ.

Last, with the compactness assumption on Φ and Ψ, and differentiable assumption on $M_{\psi,\phi}(\mathbb{P}_\mathcal{X},\mathbb{P}_\theta)$, applying Lemma 6 proves Theorem 8.

3.6.2 Proof of Lemma 9

Without loss of the generality, we can rewrite the kernel function as $k_\psi(t) = \mathbb{E}_\nu\Big[\cos\big(h_\psi(\nu)^\top t\big)\Big]$, where t is bounded. We then have

$$\begin{aligned}\|\nabla_t k_\psi(t)\| &= \Big\|\mathbb{E}_\nu\Big[\sin\big(h_\psi(\nu)^\top t\big)h_\psi(\nu)\Big]\Big\| \\ &\leq \mathbb{E}_\nu\Big[\big|\sin(h_\psi(\nu)^\top t)\big| \times \|h_\psi(\nu)\|\Big] \\ &\leq \mathbb{E}_\nu\big[\|t\|\|h_\psi(\nu)\|^2\big]\end{aligned}$$

The last inequality follows by $|\sin(x)| < |x|$. Since t is bounded, if $\mathbb{E}_\nu[\|h_\psi(\nu)\|^2] < \infty$, there exist a constant L such that $\|\nabla_t k_\psi(t)\| < L, \forall t$.

By mean value theorem, for any t and t', there exists $s = \alpha t + (1-\alpha)t'$, where $\alpha \in [0,1]$, such that

$$k_\psi(t) - k_\psi(t') = \nabla_s k_\psi(s)^\top (t - t').$$

Combining with $\|\nabla_t k_\psi(t)\| < L, \forall t$, we prove

$$k_\psi(t) - k_\psi(t') \leq L\|t - t'\|.$$

3.6.3 Proof of Theorem 11

We first prove two Lemmas.

Lemma 13. *(Consistency with respect to data) With probability at least* $1-\delta$*, we have*

$$\sup_{h\in\mathcal{H}} |\hat{T}(k_h) - T(k_h)| \leq 2\mathbb{E}_X\Big[\mathfrak{R}_X^{n-1}(\mathcal{F}_\mathcal{H})\Big] + \sqrt{\frac{2}{n}\log\frac{1}{\delta}}$$

Proof. Define

$$\rho(x_1,\ldots,x_n) = \sup_{h\in\mathcal{H}} |\hat{T}(k_h) - T(k_h)|,$$

since $|k_h(x,x')| \leq 1$, it is clearly

$$\sup_{x_1,\ldots,x_i,x_i',\ldots,x_n} |\rho(x_1,\ldots,x_i,\ldots x_n) - \rho(x_1,\ldots,x_i',\ldots x_n)| \leq \frac{2}{n}.$$

Applying McDiarmids Inequality, we get

$$\mathbb{P}\left(\rho(x_1,\ldots,x_n) - \mathbb{E}[\rho(x_1,\ldots,x_n)] \geq \epsilon\right) \leq \exp\left(\frac{-n\epsilon^2}{2}\right).$$

By Lemma 14, we can bound

$$\mathbb{E}[\rho(x_1,\ldots,x_n)] \leq 2\mathbb{E}_X\Big[\mathfrak{R}_X^{n-1}(\mathcal{F}_\mathcal{H})\Big]$$

and finish the proof. □

Lemma 14. *Given $X = \{x_1, \ldots, x_n\}$, define*

$$\rho(x_1, \ldots, x_n) = \sup_{h \in \mathcal{H}} |\hat{T}(k_h) - T(k_h)|,$$

we have

$$\mathbb{E}\Big[\rho(x_1, \ldots, x_n)\Big] \leq 2\mathbb{E}_X\Big[\mathfrak{R}_X^{n-1}(\mathcal{F}_{\mathcal{H}})\Big].$$

Proof. The proof is closely followed by Dziugaite et al. (2015). Given h, we first define $t_h(x, x') = s(x, x')k_h(x, x')$ as a new kernel function to simplify the notations. We are then able to write

$$\begin{aligned}
& \mathbb{E}\Big[\rho(x_1, \ldots, x_n)\Big] \\
= \; & \mathbb{E}_X\left[\sup_{h \in \mathcal{H}} \left| \mathbb{E}\Big[t_h(z, z')\Big] - \frac{1}{n(n-1)} \sum_{i \neq j} t_h(x_i, x_j) \right| \right] \\
\leq \; & \mathbb{E}_{X,Z}\left[\sup_{h \in \mathcal{H}} \left| \frac{1}{n(n-1)} \sum_{i \neq j} (t_h(z_i, z_j) - t_h(x_i, x_j)) \right| \right]
\end{aligned}$$

by using Jensen's inequality. Utilizing the conditional expectation and introducing the Rademacher random variables $\{\sigma_i\}_{i=1}^{n-1}$, we can write the above bound to be

$$\begin{aligned}
& \frac{1}{n} \sum_i \mathbb{E}_{X_{-i}, Z_{-i}} \mathbb{E}_{x_i, z_i}\left[\sup_{h \in \mathcal{H}} \left| \frac{\sum_{i \neq j} t_h(z_i, z_j) - t_h(x_i, x_j)}{n-1} \right| \right] \\
= \; & \mathbb{E}_{X,Z} \mathbb{E}_{X', Z', \sigma}\left[\sup_{h \in \mathcal{H}} \left| \frac{1}{n-1} \sum_{i=1}^{n-1} \sigma_i (t_h(z', z_n) - t_h(x', x_n)) \right| \right] \qquad (3.9)
\end{aligned}$$

The equality follows by $X - X'$ and $-(X - X')$ has the same distributions if X and X' are independent samples from the same distribution. Last, we can bound it by

$$\begin{aligned}
\leq \; & \mathbb{E}_X \mathbb{E}_{\sigma, X'}\left[\sup_{h \in \mathcal{H}} \left| \frac{2}{n-1} \sum_{i=1}^{n-1} \sigma_i t_h(x', x_i) \right| \right] \\
\leq \; & \mathbb{E}_X \mathbb{E}_{\sigma}\left[\sup_{f \in \mathcal{F}_{k_{\mathcal{H}}}} \left| \frac{2}{n-1} \sum_{i=1}^{n-1} \sigma_i f(x_i) \right| \right] \\
= \; & 2\mathbb{E}_X[\mathfrak{R}_X^{n-1}(\mathcal{F}_{\mathcal{H}})]
\end{aligned}$$

The second inequality follows by $s(x, x')\phi(x') \in \mathcal{F}$ since $|s(x, x')| \leq 1$.

□

Lemma 15. *(Consistency with respect to sampling random features) With probability $1 - \delta$, we have*

$$|\sup_{h \in \mathcal{H}} \hat{T}(k_h) - \sup_{h \in \mathcal{H}} \hat{T}(\hat{k}_h)| \leq \sqrt{\frac{2 \log \frac{4}{\delta}}{m}}$$

Proof. Let the optimal solutions be

$$\begin{aligned} h^* &= \arg\max_{h\in\mathcal{H}} \hat{T}(k_h) \\ \hat{h} &= \arg\max_{h\in\mathcal{H}} \hat{T}(\hat{k}_h), \end{aligned}$$

By definition,

$$\begin{aligned} & \hat{T}(\hat{k}_h) \\ = & \frac{1}{n(n-1)} \sum_{i\neq j} s_{ij} \left(\frac{1}{m} \sum_{k=1}^{m} \cos(h(\nu_k)^\top (x_i - x_j)) \right) \\ = & \frac{1}{m} \sum_{k=1}^{m} \left(\frac{1}{n(n-1)} s_{ij} \cos(h(\nu_k)^\top (x_i - x_j)) \right) \end{aligned}$$

It is true that $|\frac{1}{n(n-1)} s_{ij} \cos(h(\nu_k)^\top (x_i - x_j)| \leq 1$ since $|s_{ij}| < 1$ and $|\cos(x)| < 1$. we then have

$$\begin{aligned} & \mathbb{P}(|\sup_{h\in\mathcal{H}} \hat{T}(k_h) - \sup_{h\in\mathcal{H}} \hat{T}(\hat{k}_h)| > \epsilon) \\ \leq & \mathbb{P}(|\hat{T}(k_{h^*}) - \hat{T}(\hat{k}_{h^*})| > \epsilon) + \mathbb{P}(|\hat{T}(k_{\hat{h}}) - \hat{T}(\hat{k}_{\hat{h}})| > \epsilon) \\ \leq & 4 \exp\left(-\frac{m\epsilon^2}{2} \right), \end{aligned}$$

where the last inequality follows from the Hoeffding's inequality. □

With Lemma 13 and Lemma 15, we are ready to prove Theorem 11. We can decompose

$$\begin{aligned} & |T(\hat{k}_{\hat{h}}) - \sup_{h\in\mathcal{H}} T(k_h)| \\ \leq & |\sup_{h\in\mathcal{H}} T(k_h) - \sup_{h\in\mathcal{H}} \hat{T}(k_h)| + |\sup_{h\in\mathcal{H}} \hat{T}(k_h) - \hat{T}(\hat{k}_{\hat{h}})| + |\hat{T}(\hat{k}_{\hat{h}}) - T(\hat{k}_{\hat{h}})| \\ \leq & \sup_{h\in\mathcal{H}} |T(k_h) - \hat{T}(k_h)| + |\sup_{h\in\mathcal{H}} \hat{T}(k_h) - \hat{T}(\hat{k}_{\hat{h}})| + \sup_{h\in\mathcal{H}} |\hat{T}(\hat{k}_h) - T(\hat{k}_h)| \end{aligned}$$

We then bound the first and third terms by Lemma 13 and the second term by Lemma 15. Last, using a union bound completes the proof.

3.6.4 Hyperparameters of GAN and RKS experiments

IKL on GAN For Gaussian kernels, we use $\sigma_q = \{1, 2, 4, 8, 16\}$ for images and $\sigma_q = \{0.5, 1, 2, 4, 8\}$ for text; for RQ kernels, we use $\alpha_q = \{0.2, 0.5, 1, 2, 5\}$ for images and $\alpha_q = \{0.04, 0.1, 0.2, 0.4, 1\}$ for text. We used Adam as optimizer. The learning rate for training both f_ϕ and g_θ is 0.0005 and 0.0001 for image and text experiments, respectively. The batch size B is 64. We set hyperparameter n_c for updating critic to be $n_c = 5$ and $n_c = 10$ for CIFAR10 and Google Billion Word datasets. The learning rate of of h_ψ for Adam is 10^{-6}

IKL on RKS For OPT-KL, we use the code provided by Sinha and Duchi (2016)[3]. We tune the hyperparameter $\rho = \{1.25, 1.5, 2, 4, 16, 64\}$ on the validation set. For RFF, OPT-KL, and IKL, the linear classifier is Logistic Regression Fan et al. (2008)[4], as to make reasonable comparison with MLP. We

[3] https://github.com/amansinha/learning-kernels

[4] https://github.com/cjlin1/liblinear

use 3-fold cross validation to select the best C on training set and present the error rate on test set. For CIFAR-10 and MNIST, we normalize data to be zero mean and one standard deviation in each feature dimension. The learning rate for Adam is 10^{-6}. We follow Bullins et al. (2018) to use early stopping when performance on validation set does not gain.

Part II

Kernel Learning for Explicit Generative Models

Chapter 4

Kernel Stein Generative Modeling

4.1 Background

Drawing novel samples from the data distribution is at the heart of generative models, where existing work can be put into two categories, namely the *Implicit Generative Models* (IGM) and the *Explicit Generative Models* (EGM). Recall that in Chapter 2 and 3, we demonstrate how kernel learning improve moment matching networks for implicit generative modeling.

On the other hand, EGMs typically optimize the likelihood of non-normalized density models (e.g., energy-based models (LeCun et al., 2006)) or learn score functions (Hyvärinen, 2005; Vincent, 2011) (i.e., gradient of log data-density). Because of the explicitly modeling of densities or gradient of log-densities, EGM is still favorable for many applications such as anomaly detection (Du and Mordatch, 2019; Grathwohl et al., 2020), image processing (Song and Ermon, 2019) and more. However, the generative capability of many EGMs is not as competitive with GAN on high-dimensional distributions, such as images. Recent advances in Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh, 2011) has led to certain success in EGMs, especially with energy-based models (Du and Mordatch, 2019; Nijkamp et al., 2019; Grathwohl et al., 2020) and score-based models (Song et al., 2019) for high-dimension inference tasks such as image generation. As a stochastic optimization procedure, SGLD moves samples along the gradient of log-density, with carefully controlled diminishing random noise, which converges to the true data distribution with a theoretical guarantee. The recent noise-conditioned score network (Song and Ermon, 2019) estimates the score functions using perturbed data distributions with varying degrees of Gaussian noises. For inference, they consider annealed SGLD to produce impressive samples that are comparable to state-of-the-art (SOTA) GAN-based models on CIFAR-10.

Another interesting sampling technique is Kernel Stein variational gradient descent (SVGD) (Liu and Wang, 2016; Liu, 2017), which iteratively produce samples via deterministic updates that optimally reduce the KL divergence between the model and the target distribution. The particles (samples) in SVGD interact with each other, simultaneously moving towards a high-density region following the gradients, and also pushing each other away due to a repulsive force induced from the kernels. These interesting properties of SVGD has made it promising in various challenging applications such as Bayesian optimization (Gong et al., 2019), deep probabilistic models (Wang and Liu, 2016; Feng et al., 2017; Pu et al., 2017), and reinforcement learning (Haarnoja et al., 2017).

Despite the attractive nature of SVGD, how to make its inference effective and scalable for complex high-dimensional data distributions is an open question that have not been studied in sufficient depth. One major challenge in high-dimensional inference is to deal with multi-modal distributions with many of low-density regions, where SVGD can fail even on simple Gaussian mixtures. A remedy for this problem is to use "noise-annealing". However, such a relatively simple solution may still lead to deteriorating performance along with the increased dimensionality of data.

In this chapter, we aim to significantly enhance the capability of SVGD for sampling from complex and high-dimensional distributions. Specifically, we propose a novel method, namely the Noise Conditional Kernel SVGD or NCK-SVGD in short, where the kernels are conditionally learned or selected based on the perturbed data distributions. Our main contributions can be summarized in three folds. Firstly, we propose to learn the parameterized kernels with noise-conditional auto-encoders, which captures shared visual properties of sampled data at different noise levels. Secondly, we introduce NCK-SVGD with an additional entropy regularization, for flexible control of the trade-off between sample quality and diversity, which is quantitatively evaluated with precision and recall curves. Thirdly, the proposed NCK-SVGD achieves a new SOTA FID score of 21.95 on CIFAR-10 within the EGM family and is comparable to the results of GAN-based models in the IGM family. Our work shows that high dimensional inference can be successfully achieved with SVGD.

4.2 Score-based Generative Modeling: Inference and Estimation

In this section, we review Stein Variational Gradient Descent (SVGD) for drawing samples from target distributions and describe how to estimate score functions via a recent advance in Noise Conditional Score Network (NCSN).

4.2.1 Stein Variational Gradient Descent (SVGD)

Let $p_d(\mathbf{x})$ be a positive and continuously differentiable probability density function on $\mathbb{R}^d$. For simplicity, we denote $p_d(\mathbf{x})$ as p in the following derivation. SVGD (Liu and Wang, 2016; Liu, 2017) aims to find a set of particles $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^d$ to approximate p, such that the empirical distribution $q(\mathbf{x}) = \sum_i \delta(\mathbf{x} - \mathbf{x}_i)$ of the particles weakly converges to p when n is large. Here $\delta(\cdot)$ denotes the Dirac delta function.

To achieve this, SVGD begins with a set of initial particles from q, and iteratively updates them with a deterministic transformation function $T_{\boldsymbol{\phi}}(\cdot)$:

$$\mathbf{x}' \leftarrow T_{\boldsymbol{\phi}}(\mathbf{x}) = \mathbf{x} + \epsilon \boldsymbol{\phi}(\mathbf{x}), \text{ let } \boldsymbol{\phi}^*_{q,p} = \arg\max_{\boldsymbol{\phi} \in \mathcal{H}} \Big\{ -\frac{d}{d\epsilon} \mathrm{KL}\big(T_{\boldsymbol{\phi}}(q) \,||\, p\big)\Big|_{\epsilon=0} \text{ s.t. } \|\boldsymbol{\phi}\|_{\mathcal{H}} \leq 1 \Big\}, \tag{4.1}$$

where $T_{\boldsymbol{\phi}}(q)$ is the measure of the updated particles $\mathbf{x}' = T_{\boldsymbol{\phi}}(\mathbf{x})$ and $\boldsymbol{\phi}^*_{q,p} : \mathbb{R}^d \to \mathbb{R}^d$ is the optimal transformation function maximally decreasing the KL divergence between the target data distribution p and the transformed particle distribution $T_{\boldsymbol{\phi}}(q)$, for $\boldsymbol{\phi}$ in the unit ball in the RKHS.

By letting ϵ go to zero, the continuous Stein descent is therefore given by $dX_t = \boldsymbol{\phi}^*_{q_t,p}(X_t)dt$, where q_t is the density of X_t. A key observation from Liu (2017) is that, under some mild conditions, the negative gradient of KL divergence in Eq. (4.1) is exactly equal to the square *Kernel Stein Discrepancy* (KSD):

$$-\frac{d}{dt}\mathrm{KL}(q_t \,||\, p) = \mathbb{S}(q_t \,||\, p)^2 = \max_{\boldsymbol{\phi} \in \mathcal{F}} \big(\mathbb{E}_{\mathbf{x}\sim q_t}[\mathcal{A}_p \boldsymbol{\phi}(x)]\big)^2, \tag{4.2}$$

where $\mathcal{A}_p\boldsymbol{\phi}(\mathbf{x}) = \nabla \log p(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}) + \nabla \cdot \boldsymbol{\phi}(\mathbf{x})$, $\nabla \cdot \boldsymbol{\phi} = \sum_{j=1}^d \partial_{\mathbf{x}_j} \boldsymbol{\phi}_j(\mathbf{x})$ and $\mathcal{A}_p$ is the Stein operator that maps a vector-valued function $\boldsymbol{\phi}$ to a scalar-valued function $\mathcal{A}_p\boldsymbol{\phi}$.

KSD $\mathbb{S}(q \mid\mid p)$ provides a discrepancy measure between q and p and $\mathbb{S}(q \mid\mid p) = 0$ iff $q = p$ given that $\mathcal{H}$ is sufficiently large. By taking $\mathcal{H}$ to be a reproducing kernel Hilbert space (RKHS), KSD provides a closed-form solution of Eq. (4.2). Specifically, let $\mathcal{H}_0$ be a RKHS of scalar-valued functions with a positive definite kernel $k(\mathbf{x}, \mathbf{x}')$, and $\mathcal{H} = \mathcal{H}_0 \times \ldots, \mathcal{H}_0$ the corresponding $d \times 1$ vector-valued RKHS. The optimal solution of Eq. (4.2) (Liu et al., 2016; Chwialkowski et al., 2016) is $\mathbb{S}(q \mid\mid p) = \|\boldsymbol{\phi}^*_{q,p}(\cdot)\|_{\mathcal{H}}$ where

$$\boldsymbol{\phi}^*_{q,p}(\cdot) \propto \mathbb{E}_{\mathbf{x}\sim q}[\mathcal{A}_p k(\mathbf{x}, \cdot)] = \mathbb{E}_{\mathbf{x}\sim q}[\nabla \log p(\mathbf{x})^\top k(\mathbf{x}, \cdot) + \nabla_{\mathbf{x}} k(\mathbf{x}, \cdot)]. \tag{4.3}$$

The remaining question is how to estimate the score function $s_p(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_d(\mathbf{x})$ based on the unknown data distribution $\mathbf{x} \sim p_d(\mathbf{x})$ for generative modeling tasks.

4.2.2 Score Estimation

To circumvent the expensive Monte Carlo Markov Chain (MCMC) sampling or the intractable partition function when estimating the non-normalized probability density (e.g., energy-based models), Score Matching (Hyvärinen, 2005) directly estimates the score by $\arg\min_\theta \frac{1}{2}\mathbb{E}_{p_d(\mathbf{x})}[\|s_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_d(\mathbf{x})\|_2^2]$, where $s_\theta(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^d$. To train the model $s_\theta(\mathbf{x})$, we use the equivalent objective

$$\arg\min_\theta \mathbb{E}_{p_d(\mathbf{x})}\Big[\mathrm{tr}\big(\nabla_{\mathbf{x}} s_\theta(\mathbf{x})\big) + \frac{1}{2}\|s_\theta(\mathbf{x})\|_2^2\Big] \tag{4.4}$$

without the need of accessing $\nabla_{\mathbf{x}} \log p_d(\mathbf{x})$. However, score matching is not scalable to deep neural network and high-dimensional data (Song et al., 2019) due to the expensive computation of $\mathrm{tr}(\nabla_{\mathbf{x}} s_\theta(\mathbf{x}))$.

To overcome this issue, denoising score matching (DSM) (Vincent, 2011) instead matches $s_\theta(\mathbf{x})$ to a non-parametric kernel density estimator (KDE) (i.e., $\frac{1}{2}\mathbb{E}_{p_\sigma(\tilde{\mathbf{x}})}\left[\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}})\|^2\right]$) where $p_\sigma(\tilde{\mathbf{x}}) = \frac{1}{n}\sum_{i=1}^n p_\sigma(\tilde{\mathbf{x}}|\mathbf{x}_i)$ and $p_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) \propto \exp(-\|\tilde{\mathbf{x}} - \mathbf{x}\|^2/2\sigma^2)$ is a smoothing kernel with isotropic Gaussian of variance σ^2. Vincent (2011) further show that the objective is equivalent to the following

$$\frac{1}{2}\mathbb{E}_{p_\sigma(\tilde{\mathbf{x}}|\mathbf{x})p_d(\mathbf{x})}\left[\|s_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}}|\mathbf{x})\|^2\right], \tag{4.5}$$

where the target $\nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = (\mathbf{x} - \tilde{\mathbf{x}})/\sigma^2$ has a simple closed-form. We can interpret Eq. (4.5) as learning a score function $s_\theta(\tilde{\mathbf{x}})$ to move noisy input $\tilde{\mathbf{x}}$ toward the clean data $\mathbf{x}$.

One caveat of the DSM is that the optimal score $s_{\theta^*}(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x}) \approx \nabla_{\mathbf{x}} p_d(\mathbf{x})$ is true only when the noise σ is small enough. However, learning the score function with the single-noise perturbed data distribution will lead to inaccurate score estimation in the low data density region on high-dimension data space, which could be severe due to the low-dimensional manifold assumption. Thus, Song and Ermon (2019) propose learning a noise-conditional score network (NCSN) based on multiple perturbed data distributions with Gaussian noises of varying magnitudes:

$$\frac{1}{2L}\sum_{l=1}^L \sigma_l^2 \cdot \mathbb{E}_{p_{\sigma_l}(\tilde{\mathbf{x}}|\mathbf{x})p_d(\mathbf{x})}\left[\|s_\theta(\tilde{\mathbf{x}}; \sigma_l) - \nabla_{\tilde{\mathbf{x}}} \log p_{\sigma_l}(\tilde{\mathbf{x}}|\mathbf{x})\|^2\right], \tag{4.6}$$

where $\{\sigma_l\}_{l=1}^L$ is a positive geometric sequence such that σ_1 is large enough to mitigate the low density region on high-dimension space and σ_L is small enough to minimize the effect of perturbed data. Note that σ_l^2 is to balance the scale of loss function for each noise level σ_l.

After learning the noise conditional score functions $s_\theta(\tilde{\mathbf{x}}, \sigma)$, Song and Ermon (2019) conduct anneal SGLD to draw samples from a sequence of the model's score, under annealed noise levels $\sigma_1 > \cdots > \sigma_L$, which is

$$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \frac{\eta_l}{2} s_\theta(\mathbf{x}_t, \sigma_l) + \alpha\sqrt{\eta_l}\mathbf{z}_t, \quad \forall t = 1, \ldots, T \text{ and } l = 1, \ldots, L, \tag{4.7}$$

where $\mathbf{z}_t \sim \mathcal{N}(0, 1)$ is the standard normal noise and α is a constant controlling the diversity of SGLD. It is crucial to choose the learning rate $\eta_l = \eta_0 \cdot \sigma_l^2/\sigma_L^2$, which balances the scale between the gradient term $\|\eta_l s_\theta(\mathbf{x}, \sigma_l)\|$ and the noise term $\|\sqrt{\eta_l}\mathbf{z}_t\|$. See detailed explanation in Song and Ermon (2019).

4.3 Challenges of SVGD in High-dimension

In this section, we provide a deeper analysis of SVGD on a toy mixture model with an imbalance mixture weights, as the dimension of the data distribution increases.

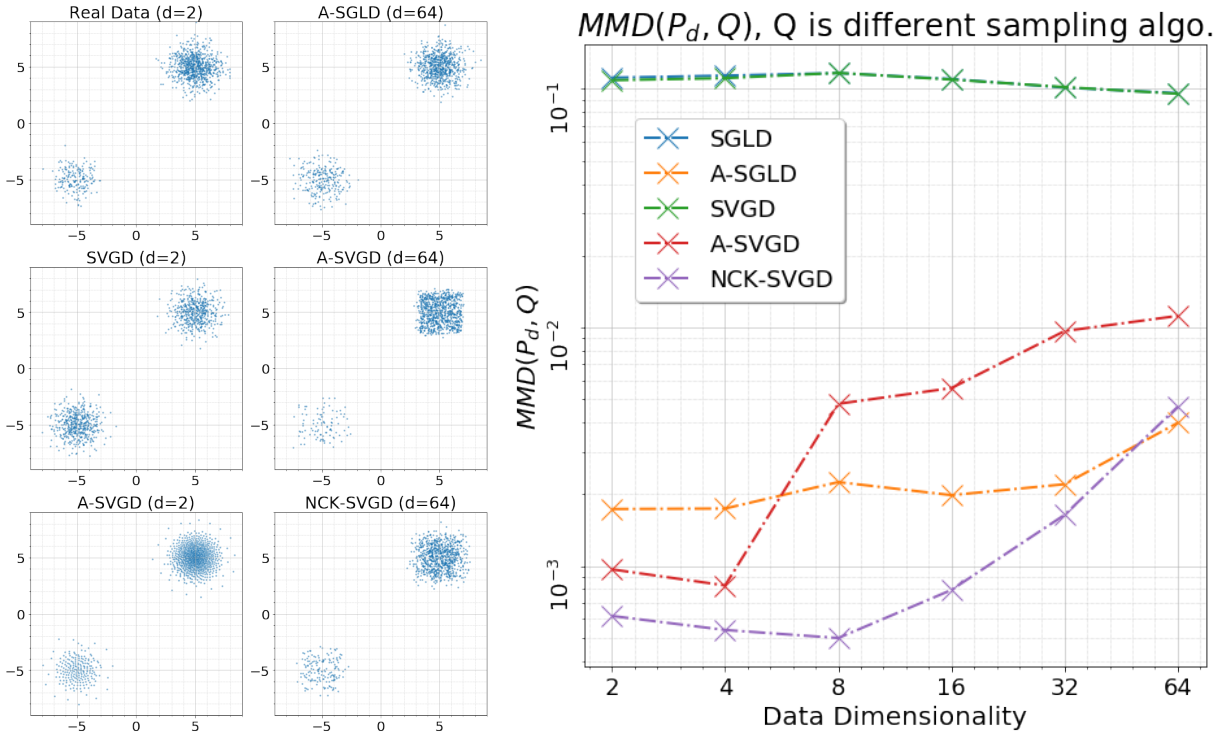
4.3.1 Mixture of Gaussian with Disjoint Support

Consider a simple mixture distribution $p_d(\mathbf{x}) = \pi p_1(\mathbf{x}) + (1 - \pi)p_2(\mathbf{x})$, where $p_1(\mathbf{x})$ and $p_2(\mathbf{x})$ having disjoint, separated supports, and $\pi \in (0, 1)$. Wenliang et al. (2019); Song and Ermon (2019) demonstrate that score-based sampling methods such as SGLD can not correctly recover the mixture weights of these two modes in reasonable time. The reason is that the score function $\nabla_{\mathbf{x}} \log p_d(\mathbf{x})$ will be $\nabla_{\mathbf{x}} \log p_1(\mathbf{x})$ in the support of p_1 and $\nabla_{\mathbf{x}} \log p_2(\mathbf{x})$ in the support of p_2. In either case, the score-based sampling algorithms are blind about the mixture weights, which may leads to samples with any reweighing of the components, depending on the initialization. We now show that the Vanilla SVGD (SVGD) also suffer from this issue on a 2-dimensional mixture of Gaussian $p_d(\mathbf{x}) = 0.2\mathcal{N}((-5, -5), I) + 0.8\mathcal{N}((5, 5), I)$. We use the ground truth scores (i.e., $\nabla_{\mathbf{x}} \log p_d(\mathbf{x})$) when sampling with SVGD. See the middle left of Figure 4.1a for the samples and the green curve in Figure 4.1b for the error (e.g., Maximum Mean Discrepancy) between the generated samples and the ground truth samples. This phenomenon can also be explained by the objective of SVGD, as KL-divergence is not sensible to the mixture weights.

4.3.2 Anneal SVGD in High-dimension

To overcome this issue, Song and Ermon (2019) propose Anneal-SGLD (A-SGLD) with a sequence of noise-perturbed score functions, where injecting noises with varying magnitudes of the variance will enlarge the support of different mixture components, and mitigate the low-data density region in high-dimensional settings. We also perform Anneal-SVGD (A-SVGD) with a sequence of noise-perturbed score functions as used in A-SGLD, and consider the RBF kernel bandwidth to the median pairwise distance of samples from $p_d(\mathbf{x})$.

For the low-dimensional case (e.g., $d = 2$), A-SVGD produces samples that represents the correct mixture weights of $p_d(\mathbf{x})$, as shown in bottom left of the Figure 4.1a. However, the performance of A-SVGD deteriorates as d increases, as shown in the red curve of Figure 4.1b. On the other hand,



(a) Samples visualization

(b) MMD loss versus data dimension

Figure 4.1: (a) Visualization of real data from $p_d(\mathbf{x})$ and samples from different inference algorithms. The three figures in the left column are on 2-dimensional mixture of Gaussian. The right column is showing the results of different sampling algorithms on 64-dimensional distributions, and we visualize only the first two dimension, as the real-data distribution is isotropic. (b) Maximum Mean Discrepancy (MMD) between the real-data samples (P_d) and generated samples (Q) from different inference algorithms, where A-SGLD means Anneal-SGLD, A-SVGD means Anneal-SVGD with a fixed kernel, and NCK-SVGD means Anneal-SVGD with noise-conditional kernels. Note that SGLD (blue) and SVGD (green) have alike samples with even mixture weights, so two curves overlapped.

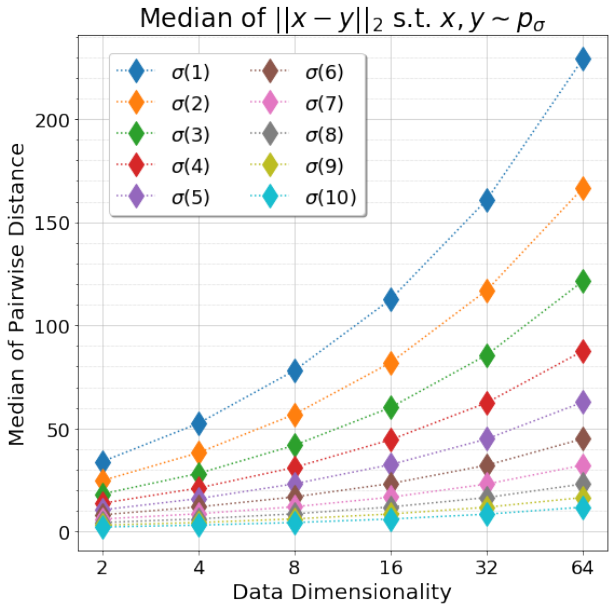


Figure 4.2: Medians of pairwise distance under different noise $\sigma(1) > \cdots > \sigma(10)$ across d.

the performance of A-SGLD seems to be rather robust w.r.t. the increasing d (i.e., the orange curve in Figure 4.1b and top-right samples in Figure 4.1a).

We argue that the failure of A-SVGD in high-dimension may be due to the inadequate kernel choice, which are fixed to the median pairwise distance based on the real-data distribution $p_d(\mathbf{x})$, regardless of the different noise level of p_σ. In Figure 4.2, we present $med(\{\|\mathbf{x}-\mathbf{y}\|_2 \text{ s.t. } \mathbf{x}, \mathbf{y} \sim p_\sigma\})$, the median pairwise distance where samples are from different noise-perturbed p_σ, along with the increase of d. We observe that, in low-dimensional setting (e.g., $d = 2$), the medians of different p_σ do not deviate too much. It explains the good performance of A-SVGD when $d < 8$ in Figure 4.1b. Nevertheless, in high-dimensional settings (e.g., $d = 64$), the median differs a lot for the largest and smallest noise-perturbed data distribution (i.e., p_{σ_1} v.s. $p_{\sigma_{10}}$). The bandwidths suitable for large perturbed noise σ_{10} no longer holds for small perturbed noise σ_1, hence limiting performance of fixed kernels.

Following this insight, we propose noise-conditional kernels (NCK) for A-SVGD, namely NCK-SVGD, where the data-driven kernels $k(\mathbf{x}, \mathbf{x}; \sigma)$ is conditional on annealing noises σ to dynamically adapt the varying scale changes. In this example, we can set the bandwidth of RBF kernel to be the median pairwise distance from noise-perturbed data distributions conditional on different noises σ. See the samples visualization and performance of NCK-SVGD in Figure 4.1a and 4.1b, respectively.

4.4 SVGD with Noise-conditional Kernels and Entropy Regularization

In Figure 4.1b, we show a simple noise-conditional kernel (NCK) for A-SVGD leads to considerable gain on the Gaussian mixtures as described in Section 4.3. Here we discuss how to learn the NCK with deep neural networks for complex real-world data distributions. What's more, we extend the proposed NCK-SVGD with entropy regularization for the diversity trade-off.

4.4.1 Learning Deep Noise-conditional Kernels

Kernel selection, also known as kernel learning, is a critical ingredient in kernel methods,and has been actively studied in many applications such as generative modeling (Sutherland et al., 2017; Li et al., 2017; Bińkowski et al., 2018; Li et al., 2019), Bayesian non-parametric learning (Wilson et al., 2016), change-point detection (Chang et al., 2019a), statistical testing (Wenliang et al., 2019), and more.

To make the kernel adapt to different noises, we consider a deep NCK of the form

$$k_\psi(\mathbf{x}, \mathbf{x}'; \sigma) = k_{\text{rbf}}\big(E_\psi(\mathbf{x}, \sigma), E_\psi(\mathbf{x}', \sigma); \sigma\big) + k_{\text{imq}}\big(E_\psi(\mathbf{x}, \sigma), E_\psi(\mathbf{x}', \sigma); \sigma\big), \tag{4.8}$$

where the Radius Basis (RBF) kernel $k_{\text{rbf}}(\mathbf{x}, \mathbf{x}'; \sigma) = \exp(-\gamma(\sigma)\|\mathbf{x} - \mathbf{x}'\|_2^2)$, the inverse multiquadratic (IMQ) kernel $k_{\text{imq}}(\mathbf{x}, \mathbf{x}'; \sigma) = (1.0 + \|\mathbf{x} - \mathbf{x}'\|_2^2)^{\tau(\sigma)}$, and the learnable deep encoder $E_\psi(\mathbf{x}, \sigma)$ with parameters ψ. Note that the kernel hyper-parameters γ and τ is also conditional on the noise σ.

Next, we learn the deep encoder $E_\psi(\mathbf{x}, \sigma) : \mathbb{R}^d \to \mathbb{R}^h$ via the noise-conditional auto-encoder framework to capture common visual semantic in noise-perturbed data distributions of $\{\sigma_l\}_{l=1}^L$:

$$\frac{1}{2L}\sum_{l=1}^{L}\frac{1}{\sigma_l^2}\cdot\mathbb{E}_{p_{\sigma_l}(\tilde{\mathbf{x}}|\mathbf{x})p_d(\mathbf{x})}\Big[\big\|D_\varphi\big(E_\psi(\tilde{\mathbf{x}}, \sigma_l), \sigma_l\big) - \mathbf{x}\big\|^2\Big], \tag{4.9}$$

where $D_\varphi(\mathbf{x}, \sigma_l) : \mathbb{R}^h \to \mathbb{R}^d$ is the corresponding noise-conditional decoder, and h is the dimension of the code-space of auto-encoder. Similar to the objective Eq. (4.6), the scaling constant $1/\sigma_l^2$ is to make the reconstruction loss of each noise level to be scale-balanced.

We note that the kernel learning via autoencoding is simple to train and is working well in our experimental study. There are many recent advance in deep kernel learning (Wilson et al., 2016; Jacot et al., 2018; Wenliang et al., 2019), which can potentially bring additional performance. We leave combining advanced kernel learning into the proposed algorithm as future work.

4.4.2 Entropy Regularization and Diversity Trade-off

Similarly to Wang and Liu (2019), we propose to learn a distribution q such that , for $\beta \geq 1$ $\min_q \mathcal{F}_\beta(q) = \text{KL}(q, p) - (\beta - 1)H(q)$. The first term is the fidelity term the second term controls the diversity. We have

$$\mathcal{F}_\beta(q) = \int q\log(q/p) + (\beta - 1)\int q\log(q) = \beta\int q\log\left(q/p^{\frac{1}{\beta}}\right) = \beta\text{KL}(q, p^{\frac{1}{\beta}}).$$

Remark 16. *Note that writing KL$(q, p^{\frac{1}{\beta}})$ is an abuse of notation, since $p^{\frac{1}{\beta}}$ is not normalized. Nevertheless as we will see next, sampling from $p^{\frac{1}{\beta}}$ can be seamlessly achieved by an entropic regularization of the stein descent.*

Proposition 17. *Consider the continuous Stein descent $dX_t = \boldsymbol{\phi}^*_{q_t,p,\beta}(X_t)dt$ where: $\boldsymbol{\phi}^*_{q_t,p,\beta}(x) = \mathbb{E}_{x'\sim q_t} s_p(x')K(x', x) + \beta\nabla_{x'}K(x', x)$ We have: $\frac{d\mathcal{F}_\beta(q_t)}{dt} = -\beta^2\mathbb{S}^2(q_t, p^{\frac{1}{\beta}})$. Hence the $\boldsymbol{\phi}^*_{p,q_t\beta}$ is a descent direction for the entropy regularized KL divergence. More importantly the decreasing amount is the closeness in the Stein sense of q_t to the smoothed distribution $p^{\frac{1}{\beta}}$.*

See Appendix 4.7.1 for the detailed derivation. From this proposition we see if β is small, entropic regularized stein descent, will converge to $p^{\frac{1}{\beta}}$, and will converge only to the high likelihood modes of the

distribution and we have less diversity. If β is high, on the other hand, we see that we will have more diversity but we will target a smoothed distribution $p^{\frac{1}{\beta}}$, since we have equivalently:

$$\frac{d\mathrm{KL}(q, p^{\frac{1}{\beta}})}{dt} = -\beta \mathbb{S}^2(q_t, p^{\frac{1}{\beta}}).$$

Hence β controls the diversity of the sampling and the precision / recall of the sampling. As shown in Figure 4.3, we visualize the non-normalized density of a 2-dimensional Gaussian mixture $p(\mathbf{x})^{1/\beta}$ with varying choices of the entropy regularizer β. Specifically, we consider

$$p(\mathbf{x}) \propto 0.8\mathcal{N}\big((5,5), I\big) + 0.2\mathcal{N}\big((-5,-5), I\big) + 0.6\mathcal{N}\big((5,-5), I\big) + 0.4\mathcal{N}\big((-5,5), I\big).$$

When β is small (e.g., $\beta = 0.5$), the resulting $p^{1/\beta}$ shows mode dropping compared to original p. When β is large (e.g., $\beta = 4.0$), the resulting $p^{1/\beta}$ covers all four modes, but wrongly with the almost equal weights. We also confirm the effect of diversity regularizer β on real-world datasets, as shown in Section 4.5.3.

Remark 18. *In Eq. (4.7), α plays the same role as β and ensures convergence of SGLD to $p^{\frac{1}{\alpha}}$.*

Algorithm 4: NCK-SVGD with Entropic Regularization.

input : $\{\sigma_l\}_{l=1}^L$ the data-perturbing noises, $s_\theta(\mathbf{x}, \sigma)$ the noise conditional score function, $k_\psi(\mathbf{x}, \mathbf{x}'; \sigma)$ the noise conditional kernel, a set of initial particles $\{\mathbf{x}_i^{(0)}\}_{i=1}^n$, the entropic regularizer β, an initial learning rate ϵ, and a maximum iteration T.

output: A set of particles $\{\mathbf{x}_i^{(T)}\}_{i=1}^n$ that approximates the target distribution.

for $l \leftarrow 1$ **to** L **do**

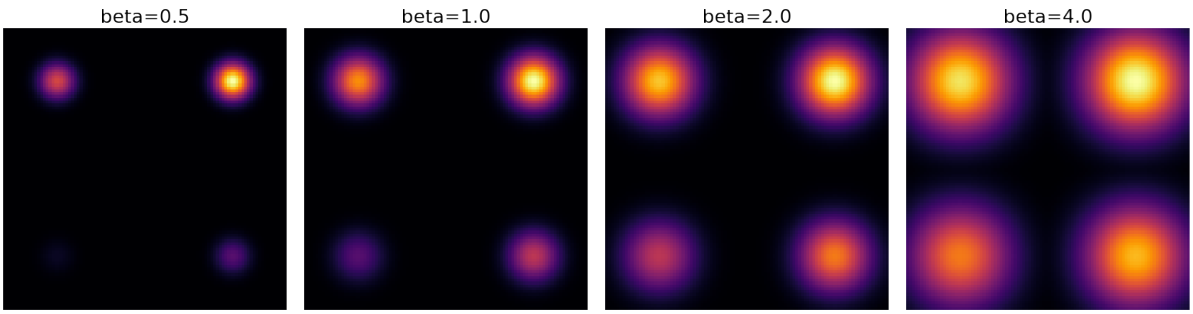
  $\eta_l \leftarrow \epsilon \cdot (\sigma_l / \sigma_L)^2$

  for $t \leftarrow 1$ **to** T **do**

    $\mathbf{x}_i^t \leftarrow \mathbf{x}_i^{(t-1)} + \eta_l \cdot \boldsymbol{\phi}^*_{q_t,p,\beta}(\mathbf{x}_i^{(t-1)}),$ where

    $\boldsymbol{\phi}^*_{q_t,p,\beta}(\mathbf{x}) = \frac{1}{n}\sum_{j=1}^n \Big[k_\psi(\mathbf{x}_j^{(t-1)}, \mathbf{x}; \sigma_l) s_\theta(\mathbf{x}, \sigma_l) + \beta \cdot \nabla_{\mathbf{x}_j^{(t-1)}} k_\psi(\mathbf{x}_j^{(t-1)}, \mathbf{x}; \sigma_l)\Big].$

  $\mathbf{x}_i^{(0)} \leftarrow \mathbf{x}_i^{(T)}, \forall i = 1, \ldots, n.$



(a) $\beta = 0.5$ (b) $\beta = 1.0$ (c) $\beta = 2.0$ (d) $\beta = 4.0$

Figure 4.3: (non-normalized) density of $p^{1/\beta}$ where $p(\mathbf{x})$ is a 2-dimensional mixture of Gaussian with imbalance mixture weights.

4.5 Experiments and Results

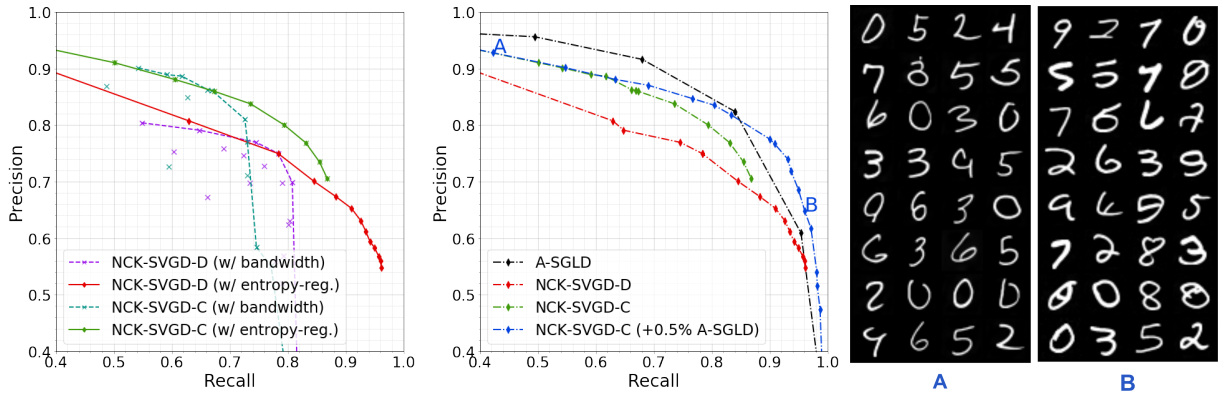
We begin with the experiment setup and show NCK-SVGD not only produces good quality images on MNIST/CIFAR-10/CelebA datasets, but also offers flexible control between the sample quality and diversity. A-SGLD is the primary competing method, and we use the recent state-of-the-art realization (Song and Ermon, 2019) for comparison.

Network Architecture For the noise-conditional score network, we use the pre-trained model (Song and Ermon, 2019)[1]. For the noise-conditional kernel, we consider a modified NCSN architecture where the encoder consists of ResNet with instance normalization layer, and the decoder consists of U-Net-type architecture. The critical difference to the score network is the dimension of bottleneck layer h, which is $h = 196$ for MNIST and $h = 512$ for CIFAR-10. Note that h for both MNIST and CIFAR-10 are considerably smaller than the data dimension, which is $d = 768$ for MNIST and $d = 3072$ for CIFAR-10. In contrast, the dimension of the hidden layers of NCSN is around 4x larger than the data dimension.

Kernel Design We consider a Mixture of RBF and IMQ kernel on the data-space and code-space features, as defined in Eq. (4.8). The bandwidth of RBF kernel $\gamma(\sigma) = \gamma_0 / med(X_\sigma)$, where $med(X_\sigma)$ denotes the median of samples' pairwise distance drawn from anneal data distributions $p_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$. We search for the best kernel hyper-parameters $\gamma_0 = \{0.4, 0.6, 0.8, 1.0, 2.0, 4.0\}$ and $\tau_0 = \{-0.1, -0.2, -0.3, -0.4\}$.

Inference Hyper-parameters Following Song and Ermon (2019), we choose $L = 10$ different noise levels where the standard deviations $\{\sigma_i\}_{i=1}^L$ is a geometric sequence with $\sigma_1 = 1$ and $\sigma_{10} = 0.01$. Note that Gaussian noise of $\sigma = 0.01$ is almost indistinguishable to human eyes for image data. For A-SGLD, we choose $T = 100$ and $\epsilon = 2 \times 10^{-5}$ and $\alpha = \{0.1, 0.2, \ldots, 1.1, 1.2\}$. For NCK-SVGD, we choose $n = 128$, $T = 50$, $\epsilon = \{2, 4, 6\} \times 10^{-4}$, $\beta = \{0.01, 0.05, 0.1, 0.2, \ldots, 1.0, \ldots, 3.9, 4.0\}$.

Evaluation Metric We report the Inception (Salimans et al., 2016)[2] and FID (Heusel et al., 2017)[3] scores using 50k samples. In addition, We also present the Improved Precision Recall (IPR) curve (Kynkäänniemi et al., 2019)[4] to justify the impact of entropy regularization and kernel hyper-parameters on diversity versus quality trade-off. For the IPR curve on MNIST, we use data-space features (i.e., raw image pixels) to compute the KNN-3 data manifold, as gray-scale images do not apply to the VGG-16 network. For the IPR curve on CIFAR-10, we follow the origin setting of Kynkäänniemi et al. (2019) that uses code-space embeddings from the pre-trained VGG-16 model to construct the KNN-3 data manifold. For simplicity, we generate 1024 samples to compute the precision and recall, and report the average of 5 runs with different random seeds.



(a) Ablation of NCK-SVGD (b) Comparison with A-SGLD (c) Representative samples

Figure 4.4: MNIST experiment evaluated with Improved Precision and Recall (IPR). (a) NCK-SVGD-D (data-space kernels) and NCK-SVGD-C (code-space kernels) with varying kernel bandwidth (γ from RBF kernel) and the entropy-regularizer α. (b) With 0.5% of A-SGLD (i.e., 5 steps at noise-level $l = 0$) as initialization, NCK-SVGD-C achieves higher recall than A-SGLD. (c) Uncurated samples of NCK-SVGD-C, including the high-precision/low-recall (A) to low-precision/high-recall (B).

4.5.1 Quantitative Evaluation

MNIST Results We analyze variants of NCK-SVGD quantitatively with IPR (Kynkäänniemi et al., 2019) on MNIST, as shown in Figure 4.4. NCK-SVGD-D denotes the NCK-SVGD with data-space kernels and NCK-SVGD-C as the NCK-SVGD with code-space kernels. We have three main observations. First and foremost, both NCK-SVGD-D and NCK-SVGD-C demonstrate flexible control between the quality (i.e., Precision) and diversity (i.e., Recall) trade-off. This finding aligns well with our theoretical analysis on the entropy regularization constant β explicitly controls the samples diversity (i.e., the green and red curve in Figure 4.4a). Furthermore, the bandwidth γ of RBF kernel also impacts the sample diversity, which attests the original study of SVGD on the repulsive term $\nabla_{\mathbf{x}'} k(\mathbf{x}', \mathbf{x})$ (Liu, 2017). Secondly, NCK-SVGD-C improves upon NCK-SVGD-D on the higher precision region, which justifies the advantages from using deep noise-conditional kernel with auto-encoder learning. Finally, when initializing with samples obtained from 5 out of 100 steps of A-SGLD at the first noise-level σ_1 (i.e., 0.5% of total A-SGLD), NCK-SVGD-C achieves a higher recall compared to the SOTA A-SGLD.

CIFAR-10 Results We compare the proposed NCK-SVGD (using code-space kernels) with two representative families of generative models: IGMs (e.g., WGAN-GP (Gulrajani et al., 2017), MoLM (Ravuri et al., 2018), SN-GAN (Miyato et al., 2018), ProgressGAN (Karras et al., 2018)) and gradient-based EGMs (e.g., EBM (Du and Mordatch, 2019), Short-run MCMC (Nijkamp et al., 2019), A-SGLD (Song and Ermon, 2019), JEM (Grathwohl et al., 2020)). See Tab. 4.1 for the Inception/FID scores and Figure 4.5 for the IPR curve.

[1] https://github.com/ermongroup/ncsn
[2] https://github.com/openai/improved-gan/tree/master/inception_score
[3] https://github.com/bioinf-jku/TTUR
[4] https://github.com/kynkaat/improved-precision-and-recall-metric

Model	Inception	FID
CIFAR-10 Unconditional		
WGAN-GP (Gulrajani et al., 2017)	7.86	36.4
MoLM (Ravuri et al., 2018)	7.90	18.9
SN-GAN (Miyato et al., 2018)	8.22	21.7
ProgressGAN (Karras et al., 2018)	8.80	-
EBM (Ensemble) (Du and Mordatch, 2019)	6.78	38.2
Short-MCMC (Nijkamp et al., 2019)	6.21	-
A-SGLD (Song and Ermon, 2019)	**8.87**	25.32
NCK-SVGD	8.20	**21.95**
CIFAR-10 Conditional		
EBM (Du and Mordatch, 2019)	8.30	37.9
JEM (Grathwohl et al., 2020)	8.76	38.4
SN-GAN (Miyato et al., 2018)	8.60	17.5
BigGAN (Brock et al., 2019)	9.22	14.73

Table 4.1: CIFAR-10 Inception and FID scores

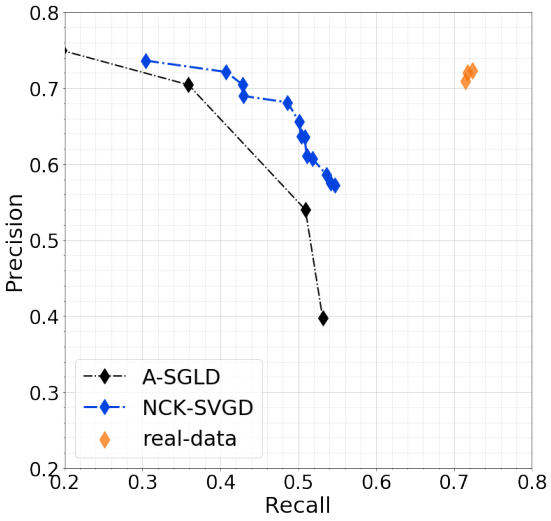
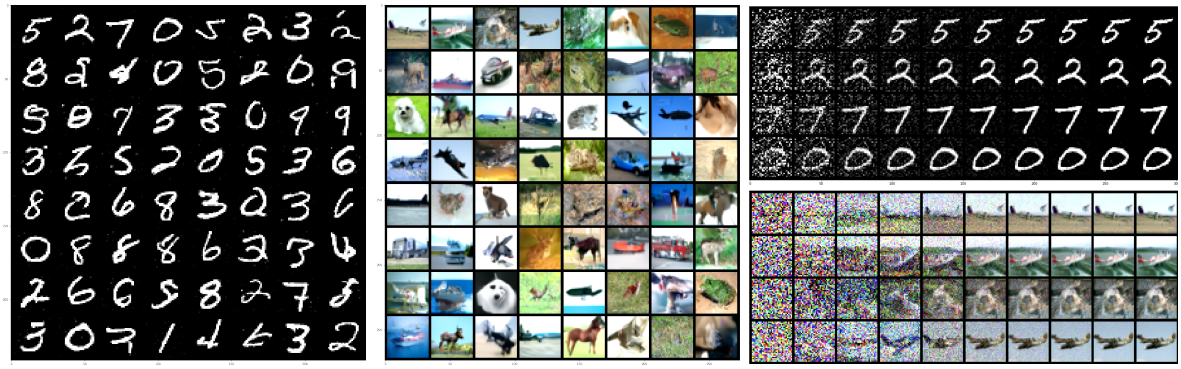


Figure 4.5: CIFAR-10 Precision-Recall Curve



(a) MNIST (b) CIFAR-10 (c) Intermediate samples

Figure 4.6: Uncurated samples generated from NCK-SVGD on MNIST and CIFAR-10 datasets.

Motivated from the MNIST experiment, we initialized the NCK-SVGD using A-SGLD samples generated from the first 5 noise-levels $\{\sigma_1, \ldots, \sigma_5\}$, then continue running NCK-SVGD for the remaining latter 5 noise-levels $\{\sigma_6, \ldots, \sigma_{10}\}$. This amounts to using 50% of A-SGLD as initialization.

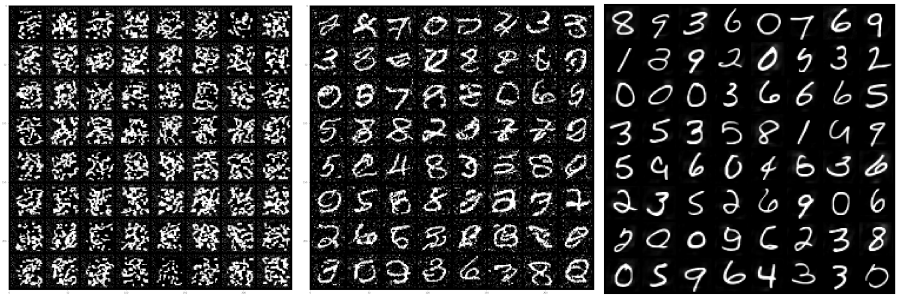
Comparing within the gradient-based EGMs (e.g., EBM, Short-run MCMC, and A-SGLD), NCK-SVGD achieves new SOTA FID score of 21.95, which is considerably better than the competing A-SGLD and is even better than some *class-conditional* EGMs. The Inception score 8.20 is also comparable to top existing methods, such as SN-GAN (Miyato et al., 2018).

From the IPR curve of Figure 4.5, we see that NCK-SVGD improves over the A-SGLD with sizable margin, especially in the high recall region. This again certifies the advantage of noise-conditional kernels and the entropy regularization indeed encourages samples with higher-recall.

4.5.2 Qualitative Analysis

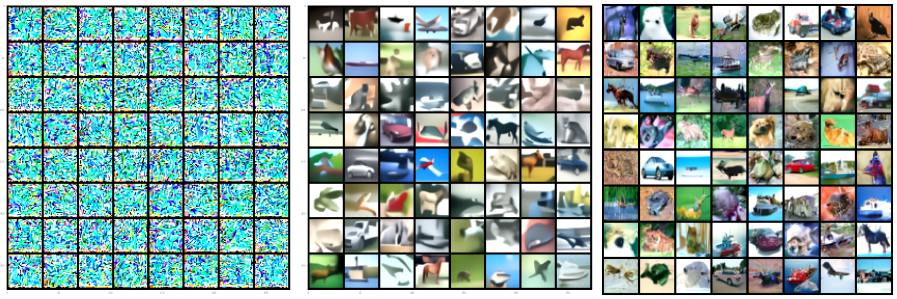
We present the generated samples from NCK-SVGD in Figure 4.6. Our generated images have higher or comparable quality to those from modern IGM like GANs or gradient-based EGMs. To intuit the procedure of NCK-SVGD, we provide intermediate samples in Figure 4.6c, where each row shows how samples evolve from the initialized samples to high quality images.

Similar to the study in Section 4.3, we compare NCK-SVGD with two SVGD baselines, namely the vanilla SVGD (i.e., SVGD) and anneal SVGD with a fixed kernel (i.e., A-SVGD), on three image generation benchmarks. From Figure 4.7, 4.8, 4.9, we observe that the proposed NCK-SVGD produces higher quality samples comparing against two baselines, SVGD and A-SGLD. Thus, we omit the quantitative evaluation, as the performance difference can be clearly distinguish from the sample quality alone.



(a) SVGD (b) A-SVGD (c) NCK-SVGD

Figure 4.7: SVGD baseline comparison on MNIST.

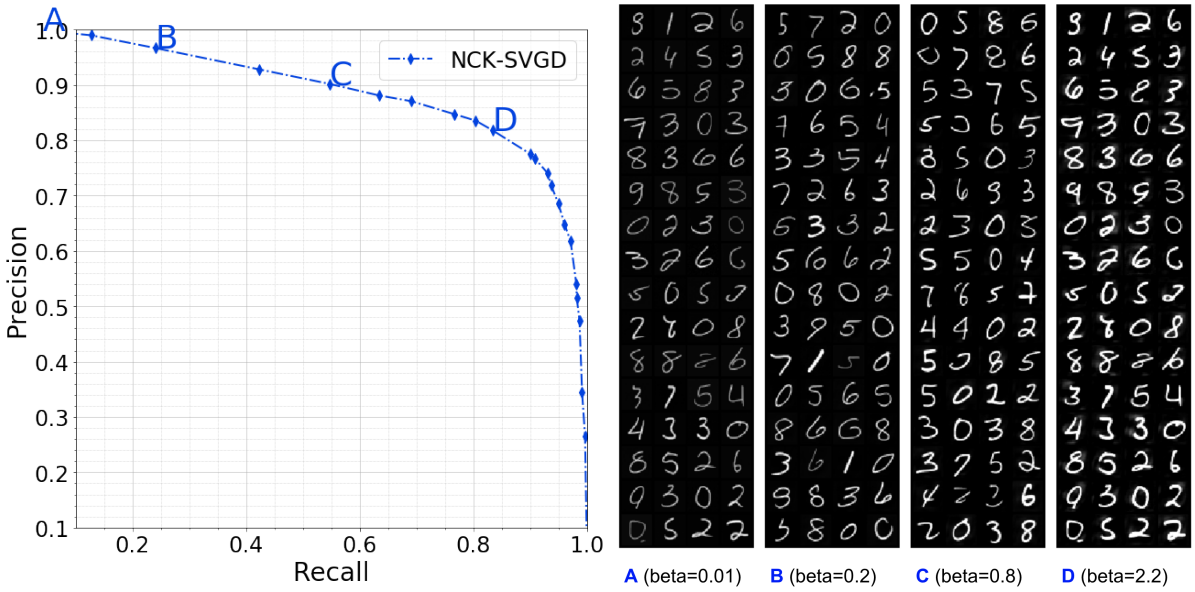


(a) SVGD (b) A-SVGD (c) NCK-SVGD

Figure 4.8: SVGD baseline comparison on CIFAR-10.

(a) SVGD (b) A-SVGD (c) NCK-SVGD

Figure 4.9: SVGD baseline comparison on CelebA.



(a) IPR of different β

(b) Corrsponding samples of different β

Figure 4.10: Varying entropy regularizer β on the Improved Precision Reccall (IPR) curve and its corresponding samples. For A, the $\beta = 0.01$ and $(P, R) = (0.99, 0.04)$. For B, the $\beta = 0.2$ and $(P, R) = (0.96, 0.24)$. For C, the $\beta = 0.8$ and $(P, R) = (0.90, 0.54)$. For D, the $\beta = 2.2$ and $(P, R) = (0.81, 0.83)$. The empirical observation on MNIST's IPR curve aligns well with our theoretical insights on the entropy-regularization β.

4.5.3 The Effect of Entropy Regularization on Precision/Recall

From the MNIST's IPR curve in Figure 4.10, we see the empirical evidence that supports the theoretical insights on the entropy-regularizer β of NCK-SVGD. We visualize four representative samples on the IPR curve, namely A, B, C, D, corresponding to different β of $0.01, 0.2, 0.8, 2.2$, respectively. When β is small (e.g., point A), the precision is high and recall is small. The resulting samples show that many modes are disappearing. In contrary, when β is large (e.g., point D), the precision becomes lower but recall is greatly increase. The resulting samples have better coverage in different digits.

4.6 Summary

In this chapter, we presented NCK-SVGD, a diverse and deterministic sampling procedure of high-dimensional inference for image generations. NCK-SVGD is competitive to advance stochastic MCMC methods such as A-SGLD, and reaching a lower FID scores of 21.95. In addition, NCK-SVGD with entropic regularization offers a flexible control between sample quality and diversity, which is quantitatively verified by the precision and recall curves.

4.7 Appendix

4.7.1 Proof of Proposition 17

We propose to learn a distribution q such that , for $\beta \geq 1$

$$\min_q \mathcal{F}_\beta(q) = \mathrm{KL}(q, p) - (\beta - 1)H(q)$$

The first term is the fidelity term the second term controls the diversity. We have

$$\mathcal{F}_\beta(q) = \int q \log(q/p) + (\beta - 1) \int q \log(q)$$

Its first variation is given by:

$$D_q \mathcal{F}(q) = \nabla_x \log\left(\frac{q}{p}\right) + (\beta - 1)\nabla_x \log(q) = \nabla_x \log\left(\frac{q^\beta}{p}\right)$$

Proposition 19. *Consider the continuous Stein descent*

$$dX_t = \boldsymbol{\phi}^*_{p,q_t,\beta}(X_t)dt$$

where:

$$\boldsymbol{\phi}^*_{p,q_t,\beta}(x) = \mathbb{E}_{x' \sim q_t} s_p(x')K(x', x) + \beta \nabla_{x'} K(x', x)$$

We have:

$$\frac{d\mathcal{F}_\beta(q_t)}{dt} = -\beta^2 \mathbb{S}^2(q_t, p^{\frac{1}{\beta}}).$$

Hence the $f^*_{p,q_t\beta}$ is a descent direction for the entropy regularized KL divergence. More importantly the decreasing amount is the closeness in the Stein sense of q_t to the smoothed distribution $p^{\frac{1}{\beta}}$.

From this proposition we see if β is small, entropic regularized stein descent, will converge to $p^{\frac{1}{\beta}}$, and will converge only to the high likelihood modes of the distribution and we have less diversity. If β is high, on the other hand, we see that we will have more diversity but we will target a smoothed distribution $p^{\frac{1}{\beta}}$. Hence β controls the diversity of the sampling and the precision / recall of the sampling.

Proof. Let q_t be the density of q_t, We have:

$$\frac{\partial q_t}{\partial t} = -div(q_t \boldsymbol{\phi}^*_{p,q_t,\beta})$$

It follows that we have:

$$
\begin{aligned}
\frac{d\mathcal{F}_\beta(q_t)}{dt} &= \int \left\langle \nabla_x D_q \mathcal{F}(q_t), \boldsymbol{\phi}^*_{p,q_t,\beta} \right\rangle q_t \\
&= \int \left\langle \nabla_x \log(q_t^\beta / p), \boldsymbol{\phi}^*_{p,q_t,\beta} \right\rangle q_t \\
&= \int \left\langle \boldsymbol{\phi}^*_{p,q_t,\beta}, \beta \nabla_x \log(q_t) - s_p(x) \right\rangle q_t \\
&= -\left(\int \left\langle s_p, \boldsymbol{\phi}^*_{p,q_t,\beta} \right\rangle q_t - (\beta) \int \left\langle \nabla_x q_t, f^*_{p,q_t,\beta} \right\rangle \right) \\
&= -\left(\int \left\langle s_p, \boldsymbol{\phi}^*_{p,q_t,\beta} \right\rangle q_t + (\beta) \int div(f^*_{p,q_t,\beta}) q_t \right) \\
&\quad \text{(Using divergence theorem)} \\
&= -\beta^2 \left(\int \left\langle \frac{1}{\beta} s_p, \frac{1}{\beta} \boldsymbol{\phi}^*_{p,q_t,\beta} \right\rangle q_t + \frac{1}{\beta} \int div(f^*_{p,q_t,\beta}) q_t \right) \\
&= -\beta^2 \mathbb{S}^2(q, p^{\frac{1}{\beta}}) \\
&\leq 0
\end{aligned}
$$

□

4.7.2 Toy Experiment Setup in Section 4.3

For the results in Figure 4.1, we mainly follow the setting of Song and Ermon (2019) with

$$p_d(\mathbf{x}) = 0.2\mathcal{N}((-5,-5), I) + 0.8\mathcal{N}((5,5), I).$$

We generate 1024 samples for each subfigure of Figure 4.1. The score function can be analytically derived from p_d. The initial samples are all uniformly chosen in the square 8×8. For SGLD and SVGD, we use $T = 1000$. For A-SGLD, A-SVGD, NCK-SVGD, we use $T = 100$, $L = 10$, $\sigma_1 = 20.0$, $\sigma_{10} = 1.0$. The learning rate ϵ is chosen from $\{0.1, 0.5, 1.0, 2.0, 4.0, 8.0, 16.0\}$. When evaluating with Maxmimum Mean Discrepancy $\mathbb{M}_k(P_d, Q)$ between the real data samples from p_d and the generated samples from different sampling methods Q, we consider the RBF kernel and set bandwidth by median heuristic. The experiment was run on one Nvidia 2080Ti GPU.

Part III

Large-scale Kernel Approximations

Chapter 5

Data-driven Random Fourier Features

5.1 Background

Kernel methods offer a comprehensive collection of non-parametric techniques for modeling a broad range of problems in machine learning. However, standard kernel machines such as kernel SVM or kernel ridge regression suffer from slow training and prediction which limits their use in real-world large-scale applications. Consider, for example, the training of linear regression over n labeled data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^d$ with $n \gg d$ and y_i is a binary label. The time complexity of standard least square fit is $O(nd^2)$ and the memory demand is $O(nd)$. Its kernel-based counterpart requires solving a linear system with an n-by-n kernel matrix, which takes $O(n^3 + n^2 d)$ time and $O(n^2)$ memory usage as typical. Such complexity is far from desirable in big-data applications.

Recently, significant efforts have been devoted into low-rank kernel approximation for enhancing the scalability of kernel methods. Among existing approaches, random features (Rahimi and Recht, 2008, 2009) have drawn considerable attention and yielded state-of-the-art results in classification accuracy on benchmark data sets (Huang et al., 2014; Dai et al., 2014; Li and Póczos, 2016). Specifically, inspired from Bochner's theorem '(Rudin, 1962), random Fourier features have been studied for evaluating the expectation of shift-invariant kernels (i.e., $k(\mathbf{x}, \mathbf{x}') = g(\mathbf{x} - \mathbf{x}')$ for some function g). Rahimi and Recht (2008) estimated the expectation by Monte-Carlo methods (MC); Yang et al. (2014) leveraged the low-discrepancy properties of Quasi-Monte Carlo (QMC) sequences to reduce integration errors.

Both the MC and QMC methods rely on the (implicit) assumption that all the M random features are equally important, and hence assign a uniform weight $\frac{1}{M}$ to the features in kernel estimation. Such a treatment, however, is arguably sub-optimal for minimizing the expected risk in kernel approximation. Avron et al. (2016) presented a weighting strategy for minimizing a loose error bound which depends on the maximum norm of data points. On the other hand, Bayesian Quadrature (BQ) is a powerful framework that solves the integral approximation by $\mathbb{E}[f(x)] \approx \sum_m \beta_m f(x_m)$ where $f(x_m)$ is feature function and β_m is with non-uniform weights. BQ leverages Gaussian Process (GP) to model the prior of function $f(\cdot)$, and the resulted estimator is Bayes optimal (Huszár and Duvenaud, 2012). Thus, BQ could be considered a natural choice for kernel approximation. Nonetheless, a well-known limitation (or potential weakness) is that the performance of Bayesian-based models relies on extensive hyper-parameter tuning, e.g., on the condition of the covariance matrix in the Gaussian prior, which is not suitable for large-scale applications.

To address the fundamental limitations of the aforementioned work, we propose a novel approach to data-driven feature weighting in the approximation of shift-invariant kernels, which motivated by the by Stein Effect in the statistical literature, and solve it using an efficient stochastic algorithm with a convex optimization objective. We also present a natural extension of BQ to the applications of kernel approximation. The adapted BQ together with standard MC and QMC methods serve as representative baselines in our empirical evaluations on six benchmark data sets. The empirical results show that the proposed Stein-Effect Shrinkage (SES) estimator consistently outperforms the baseline methods in terms of lowering the kernel approximation errors, and was competitive to or better than the best performer among the baseline methods in most task-oriented evaluations (on supervised learning of regression and classification tasks).

5.1.1 Random Fourier Features

At the heart of kernel methods lies the idea that inner products in high-dimensional feature spaces can be computed in an implicit form via a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which is defined on a input data domain $\mathcal{X} \subset \mathbb{R}^d$ such that $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$, where $\phi : \mathcal{X} \to \mathcal{H}$ is a feature map that associates kernel k with an embedding of the input space into a high-dimensional Hilbert space $\mathcal{H}$. A key to kernel methods is that as long as kernel algorithms have access to k, one need not to represent $\phi(\mathbf{x})$ explicitly. Most often, that means although $\phi(\mathbf{x})$ can be high-dimensional or even infinite-dimensional, their inner products, can be evaluated by k. This idea is known as the kernel trick.

Rahimi and Recht (2008) propose a low-dimensional feature map $z(\mathbf{x}) : \mathcal{X} \to \mathbb{R}^{2M}$ for the kernel function k

$$k(\mathbf{x}, \mathbf{x}') \approx \langle z(\mathbf{x}), z(\mathbf{x}') \rangle \tag{5.1}$$

under the assumption that k fall in the family of shift-invariant kernel. The starting point is a celebrated result that characterizes the class of positive definite functions:

Theorem 20 (Bochner's theorem Rudin (1962))**.** *A continuous, real valued, symmetric and shift-invariant function k on $\mathbb{R}^d$ is a positive definite kernel if and only if there is a positive finite measure $\mathbb{P}(\mathbf{w})$ such that*

$$k(\mathbf{x} - \mathbf{x}') = \int_{\mathbb{R}^d} 2\big[\cos(\mathbf{w}^T\mathbf{x})\cos(\mathbf{w}^T\mathbf{x}') + \sin(\mathbf{w}^T\mathbf{x})\sin(\mathbf{w}^T\mathbf{x}')\big] d\mathbb{P}(\mathbf{w}). \tag{5.2}$$

The most well-known kernel that belongs to the shift-invariant family is Gaussian kernel $k(\mathbf{x} - \mathbf{x}') = \exp(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2})$, the associated density $\mathbb{P}(\mathbf{w})$ is again Gaussian, $\mathcal{N}(\mathbf{0}, \sigma^{-2}\mathbf{I}_d)$.

Monte Carlo Estimation Rahimi and Recht (2008) approximate the integral representation of the kernel (5.2) by Monte Carlo method as follows,

$$k(\mathbf{x} - \mathbf{x}') \approx \frac{1}{M} \sum_{m=1}^{M} h_{\mathbf{x},\mathbf{x}'}(\mathbf{w}_m) = \langle z(\mathbf{x}), z(\mathbf{x}') \rangle, \tag{5.3}$$

where

$$
\begin{aligned}
h_{\mathbf{x},\mathbf{x}'}(\mathbf{w}) &\coloneqq 2\big[\cos(\mathbf{w}^T\mathbf{x})\cos(\mathbf{w}^T\mathbf{x}') + \sin(\mathbf{w}^T\mathbf{x})\sin(\mathbf{w}^T\mathbf{x}')\big] \\
z(\mathbf{x}) &\coloneqq \frac{1}{\sqrt{M}}[\phi_{\mathbf{w}_1}(\mathbf{x}), \ldots, \phi_{\mathbf{w}_M}(\mathbf{x})] \\
\phi_{\mathbf{w}}(\mathbf{x}) &\coloneqq \sqrt{2}[\cos(\mathbf{w}^T\mathbf{x}), \sin(\mathbf{w}^T\mathbf{x})]
\end{aligned} \tag{5.4}
$$

and $\mathbf{w}_1, \ldots, \mathbf{w}_M$ are i.i.d. samples from $\mathbb{P}(\mathbf{w})$.

After obtaining the randomized feature map z, training data could be transformed into $\{(z(\mathbf{x}_i), y_i\}_{i=1}^n$. As long as M is sufficiently smaller than n, this leads to more scalable solutions, e.g., for regression we get back to $O(nM^2)$ training and $O(Md)$ prediction time, with $O(nM)$ memory requirements. We can also apply random features to unsupervised learning problems, such as kernel clustering (Chitta et al., 2012).

Quasi-Monte Carlo Estimation Instead of using plain MC approximation, Yang et al. (2014) propose to use the low-discrepancy properties of Quasi-Monte Carlo (QMC) sequences to reduce the integration error in approximations of the form (5.3). Due to space limitation, we restrict our discussion to the background that is necessary for understanding subsequent sections. We refer interested readers to the comprehensive reviews (Caflisch, 1998; Dick et al., 2013) for more detailed exposition.

The QMC method is generally applicable to integrals over a unit cube. So the procedure is to first generate a discrepancy sequence $\mathbf{t}_1, \ldots, \mathbf{t}_M \in [0,1]^d$, and transform it into a sequence $\mathbf{w}_1, \ldots, \mathbf{w}_M$ in $\mathbb{R}^d$. To convert the integral presentation of kernel (5.2) to an integral over the unit cube, a simple change of variables suffices. For $\mathbf{t} \in \mathbb{R}^d$, define $\Phi^{-1}(\mathbf{t}) = [\Phi_1^{-1}(t_1), \ldots, \Phi_d^{-1}(t_d)] \in \mathbb{R}^d$, where we assume that the density function in (5.2) can be written as $p(\mathbf{w}) = \prod_{i=1}^M p_j(\mathbf{w}_j)$ and ϕ_j is the cumulative distribution function (CDF) of $p_j, j = 1, \ldots, d$. By setting $\mathbf{w} = \Phi^{-1}(\mathbf{t})$, the integral (5.2) is equivalent to

$$
k(\mathbf{x} - \mathbf{x}') = \int_{\mathbb{R}^d} h_{\mathbf{x},\mathbf{x}'}(\mathbf{w})p(\mathbf{w})d\mathbf{w} = \int_{[0,1]^d} h_{\mathbf{x},\mathbf{x}'}(\Phi^{-1}(\mathbf{t}))d\mathbf{t}.
$$

5.1.2 Bayesian Quadrature for Random Features

Random features constructed by either MC (Rahimi and Recht, 2008) or QMC (Yang et al., 2014) approaches employ equal weights $1/M$ on integrand functions as shown in (5.3). An important question is how to construct different weights on the integrand functions to have a better kernel approximation.

Given a fixed QMC sequence, Avron et al. (2016) solve the weights by optimizing a derived error bound based on the observed data. There are two potential weakness of Avron et al. (2016). First, Avron et al. (2016) does not fully utilize the data information because the optimization objective only depends on the upper bound information of $|\mathbf{x}_i|$ instead of the distribution of $\mathbf{x}_i$. Second, the error bound used in Avron et al. (2016) is potentially loose. Due to technical reasons, Avron et al. (2016) relaxes $h_{\mathbf{x},\mathbf{x}'}(\mathbf{w})$ to $h_{\mathbf{x},\mathbf{x}'}(\mathbf{w})\text{sinc}(T\mathbf{w})$ when they derive the error bound to optimize. It is not studied whether the relaxation results in a proper upper bound.

On the other hand, selecting weights by such way is closely connected to the Bayesian Quadrature (BQ), originally proposed by Rasmussen and Ghahramani (2003) and theoretically guaranteed under

Bayes assumptions. BQ is a Bayesian approach that puts prior knowledge on functions to solve the integral estimation problem. We first introduce the work of Rasmussen and Ghahramani (2003), and then discuss how to apply BQ to kernel approximations.

Standard BQ considers a single integration problem with form

$$\bar{f} = \int f(\mathbf{w}) d\mathbb{P}(\mathbf{w}). \tag{5.5}$$

To utilize the information of f, BQ puts a Gaussian Process (GP) prior on f with mean 0 and covariance matrix K_{GP} where

$$K_{GP}(\mathbf{w}, \mathbf{w}') = Cov(f(\mathbf{w}), f(\mathbf{w}')) = \exp(-\frac{\|\mathbf{w} - \mathbf{w}'\|_2^2}{2\sigma_{GP}^2}).$$

After conditioning on sample $\mathbf{w}_1, \cdots, \mathbf{w}_M$ from $\mathbb{P}$, we obtain a closed-form GP posterior over f and use the posterior to get the optimal Bayes estimator $\hat{f}$[1] as

$$\begin{aligned}\hat{f} &= \mathbb{E}_{GP}\left[\int f(\mathbf{w}) d\mathbb{P}(\mathbf{w})\right] \\ &= \boldsymbol{\gamma}^\top K_{GP}^{-1} f(W) = \sum_{m=1}^{M} \beta_{BQ}^{(m)} f(\mathbf{w}_m),\end{aligned}$$

where $\gamma_m = \int K_{GP}(\mathbf{w}, \mathbf{w}_m) d\mathbb{P}(\mathbf{w})$, $\beta_{BQ}^m = (K_{GP}^{-1}\boldsymbol{\gamma})_m$, and $f(W) = [f(\mathbf{w}_1), \cdots, f(\mathbf{w}_M)]$. Clearly, the resulting estimator is simply a new weighted combination of the empirical observations $f(\mathbf{w}_m)$ by $\beta_{BQ}^{(m)}$ derived from the Bayes perspective.

In kernel approximation, we could treat it as a series of the single integration problem. Therefore, we could directly apply BQ on it. Here we discuss the technical issues of applying BQ on kernel approximation. First, we need to evaluate $\boldsymbol{\gamma}$. For Gaussian kernel, we derive the closed-form expression

$$\gamma_m = \int K_{GP}(\mathbf{w}, \mathbf{w}_m) \mathbb{P}(\mathbf{w}) = \exp\left(\frac{-\|\mathbf{w}_m - \mathbf{0}\|_2^2}{\sigma_{GP}^2 + \sigma^{-2}}\right)$$

where γ_m is a random variable evaluated at $\mathbf{w}_m$.

Data-driven by Kernel Hyper-parameters The most import issue of applying BQ on kernel approximation is that it models the information of $h_{\mathbf{x},\mathbf{x}'}$ by the kernel K_{GP}. In practice, we usually use Gaussian kernel for K_{GP}, then the feature information of $(\mathbf{x}, \mathbf{x}')$ is modeling by the kernel bandwidth. As suggested by Rasmussen and Williams (2006), one could tune the best hyper-parameter either by maximizing the marginal likelihood or cross-validation based on the metrics in interested. Note that the existing BQ approximates the integral representation of a kernel function value, evaluated at a single pair of data $(\mathbf{x}, \mathbf{x}')$. However, it is unfeasible to tune individual σ_{GP} for n^2 pairs of data. Therefore, we tune a global hyper-parameter σ_{GP} on a subset of pair data points.

[1] For the square loss.

5.2 Stein Effect Shrinkage (SES) Estimator

In practice, it is well known that the performance of Gaussian Process based models heavily depend on hyper-parameter σ_{GP}, i.e. the bandwidth of kernel function k_{GP}. Therefore, instead of using Bayesian way to obtain the optimal Bayes estimator from BQ, we start from the risk minimization perspective by considering the *Stein effect* to derive the *shrinkage estimator* for random features.

The standard estimator of Monte-Carlo methods is the unbiased empirical average using equal weights $\frac{1}{M}$ for M samples which sum to 1. However, *Stein effect* (or James-–Stein effect) suggests a family of biased estimators could result in a lower risk than the standard empirical average. Here we extend the analysis of non-parametric Stein effect to the kernel approximation with random features by considering the risk function $\mathcal{R}(k, k') = \mathbb{E}_{\mathbf{x},\mathbf{x}',\mathbf{w}}(k(\mathbf{x}-\mathbf{x}') - k'(\mathbf{x}-\mathbf{x}';\mathbf{w}))^2$.

Theorem 21. *(Stein effect on kernel approximation) Let* $\hat{k}(\mathbf{x}-\mathbf{x}',\mathbf{w}) = \frac{1}{M}\sum_{m=1}^{M} h_{\mathbf{x},\mathbf{x}'}(\mathbf{w}_m)$. *For any estimator* μ*, which is independent of* $\mathbf{x}$ *and* $\mathbf{w}$*, there exists* $0 \leq \alpha < 1$ *such that* $\tilde{k}(\cdot) = \alpha\mu + (1-\alpha)\hat{k}(\cdot)$ *is an estimator with lower risk* $\mathcal{R}(k,\tilde{k}) < \mathcal{R}(k,\hat{k})$.

Proof. The proof closely follows Muandet et al. (2014) for a different problem under the non-parametric setting [2]. By bias-variance decomposition, we have

$$\begin{aligned}
\mathcal{R}(k,\hat{k}) &= \mathbb{E}_{\mathbf{x},\mathbf{x}'}\left[\mathrm{Var}_{\mathbf{w}}(\hat{k}(\mathbf{x}-\mathbf{x}',\mathbf{w}))\right] \\
\mathcal{R}(k,\tilde{k}) &= (1-\alpha)^2\mathbb{E}_{\mathbf{x},\mathbf{x}'}\left[\mathrm{Var}_{\mathbf{w}}(\hat{k}(\mathbf{x}-\mathbf{x}',\mathbf{w}))\right] + \alpha^2\mathbb{E}_{\mathbf{x},\mathbf{x}'}\left[\left(\mu - k(\mathbf{x}-\mathbf{x}')\right)^2\right].
\end{aligned}$$

By elementary calculation, we have $\mathcal{R}(k,\tilde{k}) < \mathcal{R}(k,\hat{k})$ when

$$0 \leq \alpha \leq \frac{2\mathbb{E}_{\mathbf{x},\mathbf{x}'}\left[\mathrm{Var}_{\mathbf{w}}(\hat{k}(\mathbf{x}-\mathbf{x}',\mathbf{w}))\right]}{\mathbb{E}_{\mathbf{x},\mathbf{x}'}\left[\mathrm{Var}_{\mathbf{w}}(\hat{k}(\mathbf{x}-\mathbf{x}',\mathbf{w}))\right] + \mathbb{E}_{\mathbf{x},\mathbf{x}'}\left[\left(\mu - k(\mathbf{x}-\mathbf{x}')\right)^2\right]}$$

and the optimal value happened at

$$\alpha^* = \frac{2\mathbb{E}_{\mathbf{x},\mathbf{x}'}\left[\mathrm{Var}_{\mathbf{w}}(\hat{k}(\mathbf{x}-\mathbf{x}',\mathbf{w}))\right]}{\mathbb{E}_{\mathbf{x},\mathbf{x}'}\left[\mathrm{Var}_{\mathbf{w}}(\hat{k}(\mathbf{x}-\mathbf{x}',\mathbf{w}))\right] + \mathbb{E}_{\mathbf{x},\mathbf{x}'}\left[\left(\mu - k(\mathbf{x}-\mathbf{x}')\right)^2\right]} < 1$$

□

Corollary 22. *(Shrinkage estimator) If* $k(\mathbf{x}-\mathbf{x}') > 0$*, there is a shrinkage estimator* $\tilde{k}(\cdot) = (1-\alpha^*)\hat{k}(\cdot)$ *that has lower risk than the empirical mean estimator* $\hat{k}(\cdot)$ *by setting* $\mu = 0$.

The standard shrinkage estimator derived from Stein effect uses the equal weights $\frac{1-\alpha}{M}$ to minimize the variance; however, a non-uniform weights usually results in better performance in practice Huszár and Duvenaud (2012). Therefore, based on the analysis of Theorem 21, we proposed the estimator $\tilde{k}(\mathbf{x}-\mathbf{x}',\mathbf{w}) = \sum_{m=1}^{M}\beta_m h(\mathbf{w}_m)$ obtained by minimizing the empirical risk with the constraint $\sum_{m=1}^{M}\beta_m^2 \leq c$ to shrink the weights, where c is a small constant. With Lagrangian multiplier, we end up the following risk minimization formulation which is *data-driven* by directly taking $k(\mathbf{x},\mathbf{x}')$ into account,

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^M} \sum_{(\mathbf{x},\mathbf{x}')\sim D} \left[k(\mathbf{x},\mathbf{x}') - [z(\mathbf{x}) \circ z(\mathbf{x}')]^T\boldsymbol{\beta}\right]^2 + \lambda_\beta\|\boldsymbol{\beta}\|^2, \tag{5.6}$$

[2] The original Stein effect has a Gaussian distribution assumption.

where $\circ$ denotes the Hadamard product, and λ is the regularization coefficient to shrink the weights.

The objective function (5.6) can be further simplified as a least squares regression (LSR) formulation,

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^M} \|\mathbf{t} - Z\boldsymbol{\beta}\|^2 + \lambda_\beta \boldsymbol{\beta}^T\boldsymbol{\beta}, \tag{5.7}$$

where $\mathbf{t} \in \mathbb{R}^{n^2}, Z \in \mathbb{R}^{n^2\times M}$, such that for all $i, j \in S = \{1, \ldots, n\}$, there exist a corresponding pair mapping function $\xi : \xi_S(i, j) \to k$ satisfying $t_k \coloneqq \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ and $Z(k, :) \coloneqq \mathbf{z}(\mathbf{x}_i) \circ \mathbf{z}(\mathbf{x}_j)$.

We approximately solve optimization problem (5.7) via the matrix sketching techniques (Avron et al., 2013, 2017a; Woodruff et al., 2014):

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^M} \|S\mathbf{t} - SZ\boldsymbol{\beta}\|^2 + \lambda_\beta \boldsymbol{\beta}^T\boldsymbol{\beta}, \tag{5.8}$$

where $S \in \mathbb{R}^{r\times n^2}$ is the randomized sketching matrix. Solving the sketched LSR problem (5.8) now costs only $O(rd^2 + T_s)$ where T_s is the time cost of sketching. From the optimization perspective, suppose $\boldsymbol{\beta}^*$ be the optimal solution of LSR problem (5.7), and $\tilde{\boldsymbol{\beta}}$ be the solution of sketched LSR problem (5.8). Wang et al. (2017) prove that if $r = O(M/\epsilon + poly(M))$, the the objective function value of (5.8) at $\tilde{\boldsymbol{\beta}}$ is at most ϵ worse than the value of (5.7) at $\boldsymbol{\beta}^*$. In practice, we consider an efficient $T_s = O(r)$ sampling-based sketching matrix where S is a diagonal matrix with $S_{i,i} = 1/p_i$ if row i was include in the sample, and $S_{i,i} = 0$ otherwise. That being said, we form a sub-matrix of Z by including each row of Z in the sub-matrix independently with probability p_i. Woodruff et al. (2014) suggests to set p_i proportional to $\|Z_i\|$, the norm of ith row of Z, to have a meaningful error bound.

5.3 Experiments and Results

In this section, we empirically demonstrate the benefits of our proposed method on six data sets listed in Table 5.1. The features in all the six data sets are scaled to zero mean and unit variance in a preprocessing step. We consider two different tasks: i) the quality of kernel approximation and ii) the applicability in supervised learning. Both the relative kernel approximation errors and the regression/classification errors are reported on the test sets.

Comparing Methods and Hyper-parameters

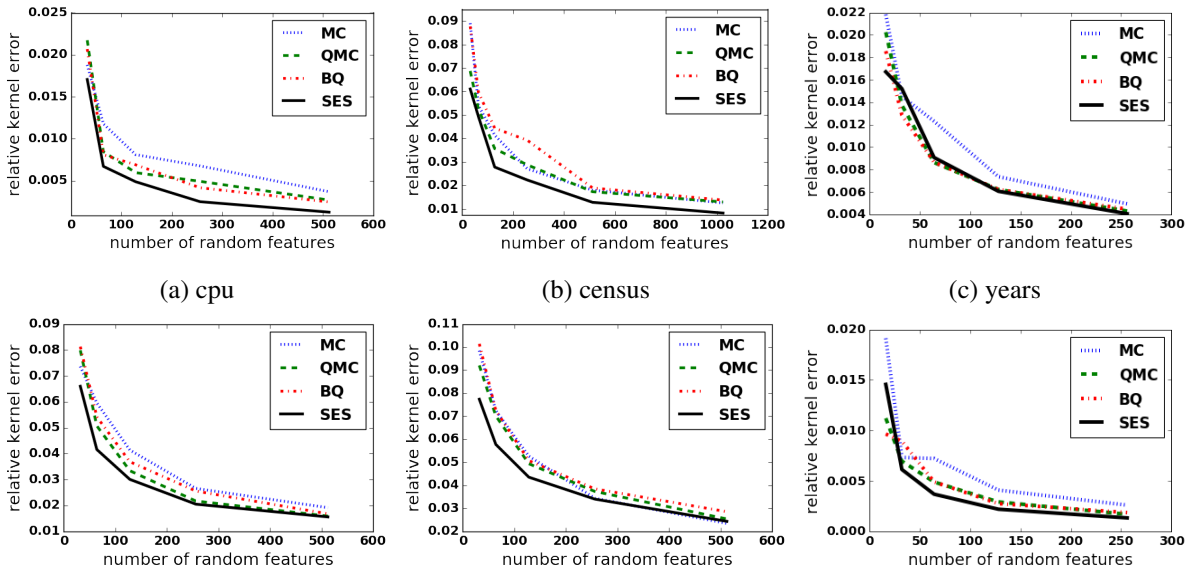
- MC: Monte Carlo approximation with uniform weight.
- QMC: Quasi-Monte Carlo approximation with uniform weight. We adapt the implementation of Yang et al. (2014) on Github. [3]
- BQ: QMC sequence with Bayesian Quadrature weights. We modify the source code from Huszár and Duvenaud (2012). [4]

Gaussian RBF kernel is used through all the experiments where the kernel bandwidth σ is tuned on the set $\{2^{-10}, 2^{-8}, \ldots, 2^8, 2^{10}\}$ that is in favor of Monte Carlo methods. For QMC method, we use scrambling and shifting techniques recommended in the QMC literature (See Dick et al. (2013) for

[3] https://github.com/chocjy/QMC-features

[4] http://www.cs.toronto.edu/ duvenaud/

more details). In BQ framework, we consider the GP prior with the covariance matrix $k_{GP}(\mathbf{w}, \mathbf{w}') = \exp(-\frac{\|\mathbf{w}-\mathbf{w}'\|_2^2}{2\sigma_{GP}^2})$ where σ_{GP} is tuned on the set $\{2^{-8}, 2^{-6}, \ldots, 2^6, 2^8\}$ over a subset of sampled pair data $(\mathbf{x}, \mathbf{x}')$. Likewise, regularization coefficient λ_β of in the proposed objective function (5.6) is tuned on the set $\{2^{-8}, 2^{-6}, \ldots, 2^6, 2^8\}$ over the same subset of sampled pair data. For regression task, we solve ridge regression problems where the regularization coefficient λ is tuned by five folds cross-validation. For classification task, we solve L2-loss SVM where the regularization coefficient C is tuned by five folds cross-validation. All experiments code are written in MATLAB and run on a Intel(R) Xeon(R) CPU 2.40GHz Linux server with 32 cores and 190GB memory.



(a) cpu (b) census (c) years

(d) adult (e) mnist (f) covtype

Figure 5.1: Kernel approximation results. x-axis is number of samples used to approximate the integral and y-axis shows the relative kernel approximation error.

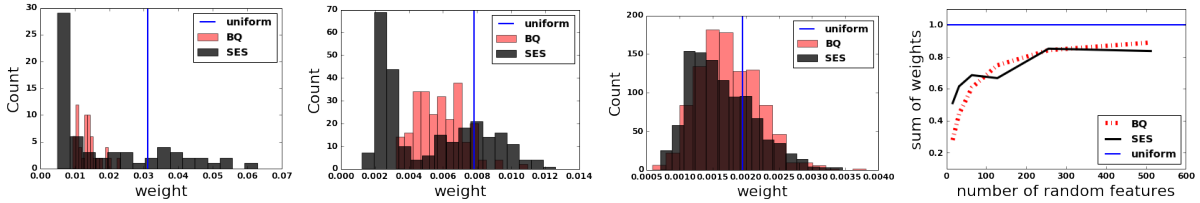
Quality of Kernel Approximation In our setting, the most natural metric for comparison is the quality of approximation of the kernel matrix. We use the relative kernel approximation error $\|K - \tilde{K}\|_F / \|K\|_F$ to measure the quality, as shown in Figure 5.1.

SES outperforms the other methods on cpu, census, adult, mnist, covtype, and is competitive with the best baseline methods over the years data sets. Notice that SES is particularly strong when the number of random features is relatively small. This phenomenon confirms our theoretical analysis, that is, the smaller the random features are, the larger the variance would be in general. Our shrinkage estimator with Stein effect enjoys better performance than the empirical mean estimators (MC and QMC) using uniformly weighted features without shrinkage. Although BQ does use weighted random features, its performance is still not as good as SES because tuning the covariance matrix in the Gaussian Process is rather difficult. Moreover, it is even harder to tune a global bandwidth for all integrand functions that approximate the kernel function.

Dataset	Task	n	d	M	MC	QMC	BQ	SES
CPU	regression	6554	21	512	3.35%	3.29%	3.29%	**3.27%**
Census	regression	18,186	119	256	8.46%	6.60%	**6.44%**	6.49%
Years	regression	463,715	90	128	0.474%	0.473%	0.473%	**0.473%**
Adult	classification	32,561	123	64	16.21%	15.57%	15.87%	**15.45%**
MNIST	classification	60,000	778	256	**7.33%**	7.72%	7.96%	7.71%
Covtype	classification	581,012	54	128	23.36%	22.88%	23.13%	**22.67%**

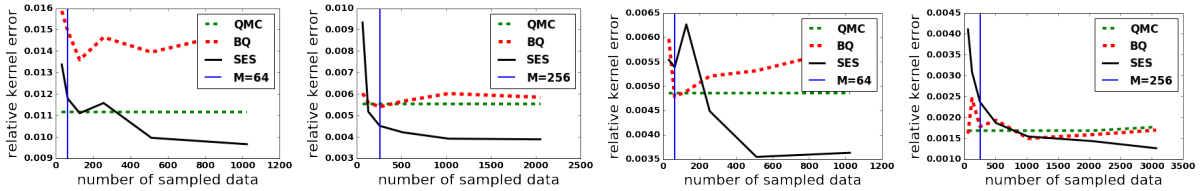
Table 5.1: Data Statistics and Supervised learning errors. n is number of instances, d is dimension of instances, M is number of random features. For regression, we report the relative regression error $\|\mathbf{y} - \tilde{\mathbf{y}}\|_2/\|\mathbf{y}\|_2$. For classification task, we report the classification error.

Supervised Learning We investigate if better kernel approximations reflect better predictions in the supervised learning tasks. The results are shown in Table 5.1. We can see that among the same data set (cpu, census, adult, and covtype) our method performs very well in terms of kernel approximation, and it also yields lower errors in most of the cases of the supervised learning tasks. Note that we only present partial experiment results on some selected number of random features. However, we also observe that the generalization error of our proposed method may be higher than that of other state-of-the-art methods at some other number of random features that we did not present here. This situation is also observed in other works (Avron et al., 2016): better kernel approximation does not always result in better supervised learning performance. Under certain conditions, less number of features (worse kernel approximation) can play a similar role of regularization to avoid overfitting (Rudi and Rosasco, 2017). Rudi and Rosasco (2017) suggest to solve this issue by tuning number of random features.



(a) $M = 32$ (b) $M = 128$ (c) $M = 512$ (d) sum of weights

Figure 5.2: A comparison of distribution of weights on adult data set. M is the number of random features.



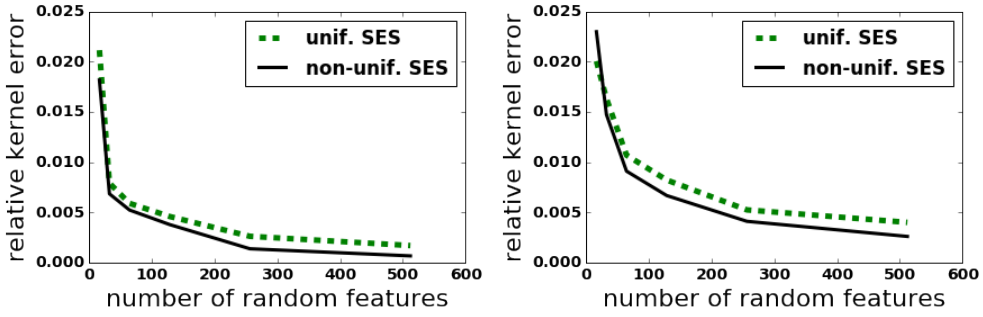
(a) $M = 64$, census (b) $M = 256$, census (c) $M = 64$, covtype (d) $M = 256$, covtype

Figure 5.3: A comparison of different sampled size training data in solving objective function (5.8).

Distributions of Feature Weights We are interested in the distributions of random feature weights derived from BQ and SES. For different numbers of random features ($M = 32, 128, 512$), we plotted the corresponding histograms and compared them with the equal weight of $1/M$ used in MC and QMC, as shown in Figure 5.2. When the number of random features is small, the difference between BQ and SES is big. As the number of random features increases, the weights of each method gradually converge to a stable distribution, and get increasingly closer to the uniform weight spike. This empirical observation agrees with the theoretical analysis we found, namely with the role of the weight estimator in the bias-variance trade-off.

Behavior of Randomized Optimization Solver In theory, our randomized solver only needs to sample $O(M)$ data in our proposed objective function (5.8) to have a sufficiently accurate solution. We verify this assertion by varying the sampled data size r in sketching matrix S and examine it affects the relative kernel approximation errors. The results are shown in Figure 5.3. For census data set, we observe that as the number of sampled data r exceeds the number of random features M, the kernel approximation of our proposed method outperforms the state-of-the-art, QMC. Furthermore, the number of sampled data required by our proposed randomized solver is still the same order of M even for a much larger data set covtype, with 0.5 millions data points. On the other hand, BQ does not have consistent behavior regarding the sampled data size, which is used for tuning the hyperparmaeter σ_{GP}. This again illustrates that BQ is highly sensitive to the covariance matrix and is difficult to tune σ_{GP} on only a small subset of sampled data. In practice, we found that we usually only need to sample up to $2 \sim 4$ times the number of random features M to get a relatively stable and good minimizer $\boldsymbol{\beta}^*$.

Uniform v.s. Non-uniform Shrinkage Weight We conducted experiments with SES using a uniform shrinkage weight versus using non-uniform shrinkage weights on the cpu and census data sets. The results in Figures 5.4 confirm our conjecture that non-uniform shrinkage weights are more beneficial, although both enjoy the lower empirical risk from the Stein effect. We also observed similar behavior on other data sets, and we omit those results.



(a) cpu data set

(b) census data set

Figure 5.4: Comparison of uniform and non-uniform shrinkage weight.

5.4 Discussion and Conclusion

Nyström methods (Williams and Seeger, 2001; Drineas and Mahoney, 2005) is yet another line of research for kernel approximation. Specifically, eigendecomposition of the kernel matrix is achieved via low rank factorization with various sampling techniques (Wang and Zhang, 2013). We refer interested readers to the comprehensive study (Yang et al., 2012) for more detailed comparisons. This chapter only focus on the study of random Fourier features.

Finally, we notice Sinha and Duchi (2016) proposes a weighting scheme for random features which matches the label correlation matrix with the goal to optimize the supervise objective (e.g. classification errors). In contrast, our work aims to optimize the kernel approximation, which does not require the label information to solve weights and can be applied to different tasks after solving the weights once.

To conclude, we propose a novel shrinkage estimator, which enjoys the benefits of Stein effect w.r.t. lowering the empirical risk compared to empirical mean estimators. Our estimator induces non-uniform weights on the random features with desirable data-driven properties. We further provide an efficient randomized solver to optimize this estimation problem for real-world large-scale applications. The extensive experimental results demonstrate the effectiveness of our proposed method.

Chapter 6

Kernel Contextual Bandit at Scale

6.1 Background

The contextual bandit problem is an important challenge in many real-world machine learning applications such as online advertisement optimization (Bottou et al., 2013; Li et al., 2010). In the advertisement use-case, the learning algorithm sequentially observes user features, places an advertisement and receives a reward in terms of click/no-click. This is encompassed by the general contextual bandit setting (Auer, 2002; Langford and Zhang, 2008), which is defined as a sequential game of T-trials, such that in each round t, the learner receives a *context* (e.g., user features), selects an *action* (e.g. an ad. placement), and receives a noisy feedback of the *reward* (e.g. click/no-click). In this setting, the objective is to maximize the expected cumulative reward, which involves a careful trade-off between the *exploration* of all possibilities and *exploitation* of previously known good decisions.

As discussed in Foster et al. (2018), at a high-level, there are two families of contextual bandit algorithms: *agnostic-based* (a.k.a. expert setting) and *realizability-based* methods. *Agnostic-based* methods (Langford and Zhang, 2008; Agarwal et al., 2014; Sen et al., 2018) aim to find the best expert/policy in a class of experts $\Pi = \{\pi_1, \ldots, \pi_M\}$. The regret analysis of agnostic-based methods is generically applicable across policy classes (under mild assumptions). However, the policy optimization relies on training cost-sensitive classification oracles, which may be expensive. *Realizability-based* methods use a model based approach (the model could be either parametric or non-parametric; however tractable algorithms are available only for a few cases). Examples include LinUCB and related kernel algorithms (Li et al., 2010; Chu et al., 2011; Srinivas et al., 2010; Chowdhury and Gopalan, 2017) as well as Thompson Sampling based algorithms (Agrawal and Goyal, 2013; Chowdhury and Gopalan, 2017); we refer to Foster et al. (2018) for additional discussion.

Recently, Krishnamurthy et al. (2016); Foster et al. (2018) show that the realizability-based LinUCB algorithm is competitive in practice with representative agnostic-based baselines that use linear policy functions. There have been several notable efforts in extending the linear realizable reward function in LinUCB to a more expressive function class (e.g. a Gaussian Process (Srinivas et al., 2010; Krause and Ong, 2011), functions with small RKHS norm (Valko et al., 2013)). While RKHS functions indeed have high expressivity, bandit algorithms using kernels are faced with computational challenges. For instance, recall T being the time horizon, the CGP-UCB algorithm (Krause and Ong, 2011) needs $O(T^3)$ compu-

tation owing to the inverse of the kernel-gram matrix. Even the incremental version KernelUCB (Valko et al., 2013) demands $O(T^2)$ computation, leading to intractability with large data-sets (e.g. when T is in the tens of thousands).

Random Fourier Features (RFF), proposed in Rahimi and Recht (2008, 2009), is now a mainstream technique that enables one to speed up kernel methods. The basic idea is to replace evaluations of (shift-invariant) kernels by an approximation based on a finite sum of s random Fourier features. In the context of Kernel Ridge Regression (KRR), the corresponding approximate regression problem can be solved in $O(Ts^2)$ time and $O(Ts)$ space, where T is the number of samples for KRR (Avron et al., 2017b). This can lead to a considerable speed-up in the regime where s is much smaller than T.

6.2 Kernel Upper-Confident-Bound Framework

We consider the problem of stochastic contextual bandits under the predictable rewards setting. Contexts are drawn from $\mathcal{X} \subseteq \mathbb{R}^d$ and actions from a finite set $\mathcal{A} \coloneqq \{1, \ldots, N\}$, for some fixed integer N. A contextual bandit algorithm (Li et al., 2010) $\mathcal{M}$ participates in a sequential game of T-trials, where at each time-step $t \in \{1, \ldots, T\}$, the bandit algorithm $\mathcal{M}$ first observes contextual features $\mathbf{x}_{a,t} \in \mathcal{X}$ for each arm $a \in \mathcal{A}$. Next, based on the observations in the previous trials, $\mathcal{M}$ selects an arm $a_t \in \mathcal{A}$ and receives the reward $y_{a_t,t} = f(\mathbf{x}_{a_t,t}) + \epsilon_t$. Here, $f(\cdot)$ is an unknown *reward function* and ϵ_t is a zero-mean noise random variable. Finally, the bandit $\mathcal{M}$ improves its strategy for selecting the next arm by augmenting its previous information with the new observation, $(\mathbf{x}_{a_t,t}, a_t, y_{a_t,t})$. It should be noted that feedback is not obtained for arms $a \neq a_t$ (i.e. the unplayed arms). For simplicity, we define $\mathbf{x}_t \coloneqq \mathbf{x}_{a_t,t}$ and $y_t \coloneqq y_{a_t,t}$ to be the context and reward at time t, when there is no ambiguity.

Cumulative Regret The goal of the learner $\mathcal{M}$ is to maximize the cumulative reward over T trials, or equivalently, minimize the cumulative regret, defined as $R(T) = \sum_{t=1}^{T} f(\mathbf{x}_{a_t^*,t}) - f(\mathbf{x}_t)$, where $a_t^* = \arg\max_{a \in \mathcal{A}} f(\mathbf{x}_{a,t})$. Cumulative regret is a metric widely used in the literature Agarwal et al. (2012); Krause and Ong (2011); Valko et al. (2013).

Assumptions and Notations The main assumption of this paper is that the reward function f lies in $\mathcal{H}_k$ i.e the Reproducing Kernel Hilbert Space (RKHS) associated with a positive semi-definite kernel $k(\mathbf{x}, \mathbf{y}) \in \mathbb{R}$ where $\mathbf{x}, \mathbf{y} \in \mathcal{X}$. We briefly review notation and assumptions about RKHS used in this paper, and refer interested reader to Wahba (1990); Scholkopf and Smola (2001) for more details.

Let $||f||_{\mathcal{H}_k}$ denote the RKHS norm of a function f w.r.t the kernel $k(\cdot, \cdot)$. Note that $f \in \mathcal{H}_k$ iff $||f||_{\mathcal{H}_k}$ is finite. We assume that the RKHS norm of the reward function is bounded that is $||f||_{\mathcal{H}_k} \leq B$. In this work, we focus on shift-invariant kernels that is $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$. Further we assume that $k(\mathbf{x}, \mathbf{x}) = 1$ for all $\mathbf{x} \in \mathcal{X}$ and $0 < k(\mathbf{x}, \mathbf{y}) \leq 1$ for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$. Of particular interest is the well-known RBF kernel $k(\mathbf{x}, \mathbf{y}) = \exp(-\gamma||\mathbf{x} - \mathbf{y}||^2)$ which satisfies these conditions. We also assume that the noise in the observations (defined above) is bounded i.e $|\epsilon_t| < \sigma$ and i.i.d sub-gaussian for all t. Further we require that the *diameter* of $\mathcal{X}$ is bounded by $\mathcal{D}$, similar to the assumptions in Rahimi and Recht (2008); Avron et al. (2017b).

Kernel UCB Framework Upper Confidence Bound (UCB) algorithms have been popular in the multi-armed bandit literature (Lai and Robbins, 1985; Auer, 2002). UCB algorithms rely on the principle of optimism - in the context of stochastic N-armed bandits the idea is to choose the arm that has the maximum upper bound at time t defined by $\mu_{a,t} + \sigma_{a,t}$, where $\mu_{a,t}$ is the empirical mean of arm a till that time while $\sigma_{a,t}$ is a confidence estimate. These algorithms were generalized to the contextual setting where the rewards depend linearly on the contextual features (Chu et al., 2011; Li et al., 2010). However, linear reward assumptions are not expressive enough and unlikely to hold in practice. This inspired the development of kernel bandit GP-UCB (Srinivas et al., 2010) and its contextual variants CGP-UCB (Krause and Ong, 2011), which generalize the reward function from linear form to lie in general RKHS. The theoretical guarantees were further refined in Valko et al. (2013). Gaussian Processes (GPs) (Rasmussen, 2004) are central to these kernel bandit settings.

A GP over a domain $\mathcal{X}$ is denoted by $GP(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ and is fully specified by the mean function $\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and a covariance/kernel function $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))]$. w.l.o.g, a GP prior over f is set to having prior with zero mean and its covariance as PSD kernel $k(\cdot, \cdot)$. In this setting, after observing noisy samples $\mathbf{y}_T = [y_1, .., y_T]$ at points $A_T = \{\mathbf{x}_1, .., \mathbf{x}_T\}$ such that $y_t = f(\mathbf{x}_t) + \epsilon_t$ ($\epsilon_t \sim N(0, \sigma^2 = \lambda_T)$), the posterior mean and covariance of f is given by,

$$\begin{aligned} \mu_T(\mathbf{x}) &= \mathbf{k}_T(\mathbf{x})^T (\mathbf{K}_T + \lambda_T \mathbf{I})^{-1} \mathbf{y}_T, \\ k_T(\mathbf{x}, \mathbf{x}') &= k(\mathbf{x}, \mathbf{x}') - \mathbf{k}_T(\mathbf{x})^T (\mathbf{K}_T + \lambda_T \mathbf{I})^{-1} \mathbf{k}_T(\mathbf{x}'), \\ \sigma_T^2(\mathbf{x}) &\triangleq k_T(\mathbf{x}, \mathbf{x}). \end{aligned} \tag{6.1}$$

We denote $\mathbf{K}_T \in \mathbb{R}^{T \times T}$ as the gram matrix such that the (i, j)-th entry is $k(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{k}_T(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_1), ..., k(\mathbf{x}, \mathbf{x}_T)]^T$. In the CGP-UCB algorithm (Krause and Ong, 2011), the expected mean reward of arm a at time t is set to $\mu_{t-1}(\mathbf{x}_{a,t})$ and the upper confidence bound is set to a multiple of $\sigma_{t-1}(\mathbf{x}_{a,t})$, where the prior kernel of the GP corresponds to the RKHS kernel k s.t the reward function lies in $\mathcal{H}_k$. For more details, we refer readers to the original papers (Krause and Ong, 2011; Srinivas et al., 2010). Before, we move further, it will be beneficial to define a fundamental quantity that appears in the analysis of GP based UCB algorithms: the maximum information gain (Srinivas et al., 2010).

Maximum Information Gain We define a slightly more general version of the *maximum Information gain* (Srinivas et al., 2010; Krause and Ong, 2011). Given a PSD kernel $k(., .)$, a sequence of parameters $\Lambda_T = \{\lambda_1, ..., \lambda_T\}$ such that $\lambda_t > 0, \forall t \in [T]$, and an ordered subset $\mathcal{A}_T = \{\mathbf{x}_1, ..\mathbf{x}_T\} \subset \mathcal{X}$, let us define the quantity,

$$I(\mathcal{A}_T, k, \Lambda_T) = \frac{1}{2} \sum_{t=1}^{T} \log \left(1 + \lambda_t^{-1} \sigma_{t-1}^2(\mathbf{x}_t)\right), \tag{6.2}$$

where $\sigma_t()$ is as defined in eq. (6.1). Then the maximum information gain associated with the kernel $k(., .)$ and the parameters Λ_T is given by,

$$\gamma_T(\Lambda_T) = \max_{\mathcal{A}_T} I(\mathcal{A}_T, k, \Lambda_T). \tag{6.3}$$

Kernel Approximation Although non-parametric kernel methods are powerful universal function approximators, they often suffer from the scalability issues as number of instances increase. For example,

the CGP-UCB algorithm (Krause and Ong, 2011) adapted to the finite arm settings requires the inversion of the Gram-matrix at each time-step. This leads to a computational complexity of $O(Nt^2 + t^3)$ at each time-step. In Algorithm 1 of the KernelUCB paper (Valko et al., 2013), even with the online inverse update of $\mathbf{K}_t$ using Schur complement (Zhang, 2006; Petersen et al., 2008), it requires $O(t^2)$ cost for the kernel matrix vector product, for each arm $a \in \mathcal{A}$. This together leads to $O(t^2N)$ cost per time step t. When running contextual bandit experiment for T trials, KernelUCB results in $O(T^3N)$ cost in total, where N is number of arms.

On the other hand, in the linear case, LinUCB (Li et al., 2010) only requires $O(d^2N)$ cost to compute the parameters per time step t, resulting in $O(TNd^2)$ time complexity for a total T-trail. It is clear that LinUCB is computationally faster than KernelUCB if $d < T$, which typically holds as the contextual bandit plays for a long time horizon such as tens of thousands or more.

We aim to bridge the computational gap between the kernel contextual bandits and the LinUCB algorithm using Random Fourier features (RFF) (Rahimi and Recht, 2008, 2009). The key idea is from Bochner's Theorem: Any shift-invariant kernel (i.e., $k(\mathbf{x}, \mathbf{z}) = k(\mathbf{x} - \mathbf{z})$, $k(0, 0) = 1$) has an associated density function $p_k(\cdot)$ satisfying

$$k(\mathbf{x}, \mathbf{z}) = \int_{\mathbb{R}^d} e^{-2\pi i \boldsymbol{\eta}^T(\mathbf{x}-\mathbf{z})} p_k(\boldsymbol{\eta}) d\boldsymbol{\eta}, \tag{6.4}$$

where we closely follow the notation in (Avron et al., 2017b). If we define $\varphi(\mathbf{x}, \boldsymbol{\eta}) = e^{-2\pi i \boldsymbol{\eta}^T \mathbf{x}}$, then we have the relation $k(\mathbf{x}, \mathbf{z}) = \mathbb{E}_{\boldsymbol{\eta} \sim p_k(.)}[\varphi(\mathbf{x}, \boldsymbol{\eta})^* \varphi(\mathbf{z}, \boldsymbol{\eta})]$ where $\varphi(\mathbf{x})^*$ denotes the conjugate transpose. Therefore, with $\boldsymbol{\eta}_1, ..., \boldsymbol{\eta}_s$ drawn randomly from $p_k(.)$, the *random Fourier features* are defined as $\psi_s(\mathbf{x}) = \frac{1}{\sqrt{s}}\big(\varphi(\mathbf{x}, \boldsymbol{\eta}_1), ..., \varphi(\mathbf{x}, \boldsymbol{\eta}_s)\big)$. This results in the approximation,

$$\tilde{k}_s(\mathbf{x}, \mathbf{z}) = \psi_s(\mathbf{x})^* \psi_s(\mathbf{z}) = \frac{1}{s} \sum_{l=1}^{s} e^{-2\pi i \boldsymbol{\eta}_l^T(\mathbf{x}-\mathbf{z})} \tag{6.5}$$

RBF kernel is an example of (6.4) such that $\psi_s(\mathbf{x}) = \frac{1}{\sqrt{s}}(\sqrt{2}\cos(\boldsymbol{\eta}_1^T\mathbf{x} + b_1), \ldots, \sqrt{2}\cos(\boldsymbol{\eta}_s^T\mathbf{x} + b_s))$ with $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_s$ and $b_1, \ldots, b_s$ sampled independently from $\frac{1}{Z}\exp\big(-\gamma\|\boldsymbol{\eta}\|_2^2\big)$ and uniformly from $[0, 2\pi]$, respectively.

For $\mathbf{x}_1, ..., \mathbf{x}_t \in \mathcal{X}$, let $\mathbf{K}_t \in \mathbb{R}^{t \times t}$ denote the gram matrix such that $\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. Also, let $\mathbf{Z}_{t,s} \in \mathbb{C}^{t \times s}$ be a matrix such that the i^{th} row is $\psi_s(\mathbf{x}_i)^*$. Then, $\tilde{\mathbf{K}}_{t,s} = \mathbf{Z}_{t,s}\mathbf{Z}_{t,s}^*$ is an approximation of the gram matrix $\mathbf{K}$. Note that our focus will be on the RBF kernel, where $\mathbf{Z}_{t,s}$ is real-valued and therefore, we will use transpose and conjugate-transpose interchangeably. We restate one of the main results from Rahimi and Recht (2008) that will be useful in subsequent sections.

Theorem 23 (Uniform Approximation Rahimi and Recht (2008))**.** *If the space $\mathcal{X}$ is a compact subset of $\mathbb{R}^d$ with diameter $\mathcal{D}$, then the random feature approximation defined in Eq. (6.5) has the following guarantee,*

$$\mathbb{P}\left(|\tilde{k}_s(\mathbf{x}, \mathbf{z}) - k(\mathbf{x}, \mathbf{z})| > \epsilon\right) \leq \delta, \quad \forall \mathbf{x}, \mathbf{z} \in \mathcal{X}. \tag{6.6}$$

for $s \geq \frac{\mathcal{C}(d+2)}{\epsilon^2} \log\left(\frac{\sigma_\eta \mathcal{D}}{\epsilon\delta}\right) \triangleq \zeta(\epsilon, \delta)$. Here, $\mathcal{C}$ is a universal constant and $\sigma_\eta = \mathbb{E}_{\eta \sim p_k}[\eta^T\eta]$.

6.3 Random Fourier Features for Kernel UCB

6.3.1 RFF-UCB-NI: A Non-incremental Algorithm

we propose the non-incremental version of our algorithm (RFF-UCB-NI) which uses random Fourier features for the kernel contextual bandit problem, and compares with the CGP-UCB algorithm (Krause and Ong, 2011). The natural way to approximate CGP-UCB would be to replace the kernel mean and variance estimates with the corresponding approximated versions, through random Fourier features. However, initial studies (Cortes et al., 2010b; Sutherland and Schneider, 2015; Sriperumbudur and Szabó, 2015) suggest that $O(t)$ random features are needed for achieving a $\epsilon_t = O(1/\sqrt{t})$ scaling of approximation error. A richer history of this literature is available in Li et al. (2018). However, recent work shows much better bounds. Specifically, for kernel ridge regression (KRR), Avron et al. (2017b) shows that $o(t)$ random features suffices (as opposed to $O(t)$ from prior literature), for risk bounds that are comparable with unapproximated KRR. Furthermore, the authors in Rudi and Rosasco (2017) show that in fact, only $O(\sqrt{t}\log t)$ features suffices for a $O(1/\sqrt{t})$ bound (and has minimax optimality (Caponnetto and De Vito, 2007)). We refer the interested readers to the recent papers Li et al. (2018); Sun et al. (2018); Carratino et al. (2018); Szabó and Sriperumbudur (2018) and the references therein. It should be noted that these risk bounds rely on spectral approximation of the kernel-gram matrix and directly analyzes the empirical risk in KRR. This is not immediately applicable to the regret analysis in the bandit setting.

Algorithm 5: Non-Incremental RFF-UCB-NI

Initialize GP prior $\mu_0(\mathbf{x}) = 0$ and $\sigma_0(\mathbf{x}) = k(\mathbf{x}, \mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$.
for $t = 1, 2, \ldots, T$ **do**
- Let $\lambda_t = \max\{t/(\max\{1, \log t\}), \sigma^2\}$ and $s_t = \max(11 \log t\,(\log(16 \log t/\delta) + 2\log t), 2)$.
- Receive context vectors $\{\mathbf{x}_{a,t}\}$ for all $a \in \mathcal{A}$.
- Let $\hat{\sigma}^2_{a,t} := k(\mathbf{x}_{a,t}, \mathbf{x}_{a,t}) - \mathbf{k}_{t-1}(\mathbf{x}_{a,t})^T(\hat{\mathbf{K}}_{t-1,s_t} + \lambda_t \mathbf{I})^{-1}\mathbf{k}_{t-1}(\mathbf{x}_{a,t})$ for all $a \in \mathcal{A}$.
- Let $\hat{\sigma}_{m,t} = \min_a \hat{\sigma}_{a,t}$ and define $s'_t = \zeta\left(\frac{\sqrt{\beta_t}\hat{\sigma}_{m,t}}{(B+\sigma)\log^2 t}, \frac{\delta}{t^2}\right)$, for $\zeta()$ defined in Theorem 23.
- Let $\hat{\mu}_{a,t} = \mathbf{k}_{t-1}(\mathbf{x}_{a,t})^T(\hat{\mathbf{K}}_{t-1,s'_t} + \lambda_t \mathbf{I})^{-1}\mathbf{y}_t$.
- Let $a_t = \arg\max_{a\in\mathcal{A}}(\hat{\mu}_{a,t} + 3\sqrt{2\beta_t}\hat{\sigma}_{a,t})$.
- Select arm a_t and observe reward y_t. Let $\mathbf{x}_t = \mathbf{x}_{a,t}$.

In our algorithm, we incorporate the spectral bounds mentioned above, in order to provide approximation error bounds on the variance/confidence estimates for each arm. Equipped with these estimates, we use just enough kernel random features such that the approximation error in the mean function is no more than a multiple of the minimum confidence estimate. This leads to a data-dependent bound on the number of RFF's, that can be much less than $O(t)$ at time t. In Algorithm 5, $\mathbf{k}_{t-1}(\mathbf{x}) = [k(\mathbf{x}_1, \mathbf{x}), ..., k(\mathbf{x}_{t-1}, \mathbf{x})]^T$, where $\mathbf{x}_t$ is as defined in the algorithm. $\hat{\mathbf{K}}_{t-1,s}$ is the gram-matrix at time t, approximated by s random features i.e $\hat{\mathbf{K}}_{t-1,s} = \mathbf{Z}_{t-1,s}\mathbf{Z}^T_{t-1,s}$. We set $\beta_t = 2B^2 + 300\gamma_t(\Lambda_t)\log^3(t/\delta)$.

The key idea in the algorithm is that a multiplicative approximation of the variance $\sigma_{a,t}$ can be achieved by using only s_t random features. After that, in approximating the kernel mean of any arm $\mu_{a,t-1}$, the approximation error $|\mu_{a,t-1} - \hat{\mu}_{a,t-1}|$ needs to be not more than a multiple of $\sqrt{\beta_t}\sigma_{a,t}$, which can be

achieved by having s'_t random features for the approximating the gram-matrix in the kernel mean function.

The inversion of $\mathbf{Z}_{t-1,s}\mathbf{Z}_{t-1,s}^T + \lambda_t \mathbf{I}$ can be performed by the Woodbury matrix identity Hager (1989), as follows,

$$\lambda_t^{-1}\mathbf{I} - \lambda_t^{-2}\mathbf{Z}_{t-1,s}(\mathbf{I} + \lambda_t^{-1}\mathbf{Z}_{t-1,s}^T\mathbf{Z}_{t-1,s})^{-1}\mathbf{Z}_{t-1,s}^T$$

Thus the variance and mean function for all the arms can be calculated in $O(Nc_t^2 + c_t^3)$ cost at time-step t, where $c_t := \max(s_t, s'_t)$.

6.3.2 Theoretical analysis of RFF-UCB-NI

In this section, we provide regret bounds and computational cost bounds for the non-incremental algorithm (Algorithm 5). We show that we can achieve, essentially the same regret bound as the exact CGP-UCB algorithm Krause and Ong (2011), but with a data dependent computational cost bound that can be much lesser that $\sum_{t=1}^{T} O(Nt^2 + t^3)$ (cost of CGP-UCB). Our main theoretical result is provided as Theorem 24.

Theorem 24. *Let $\Lambda_t = \{\lambda_1, ..., \lambda_t\}$ for λ_t as defined in Algorithm 5 and let $\gamma_t(\Lambda_t)$ be as defined in Eq. (6.3). When the reward function and the noise at each time-step follows the conditions in Section 6.2 and under the assumption that $||\mathbf{K}_t||_2 \geq \lambda_t$, the regret of Algorithm 5 satisfies the bound,*

$$\mathbb{P}\left(R(T) \leq \sqrt{\frac{81T\beta_T\gamma_T(\lambda_T)}{\log(1+\lambda_T^{-1})}}, \forall T\right) \geq 1 - 7\delta. \tag{6.7}$$

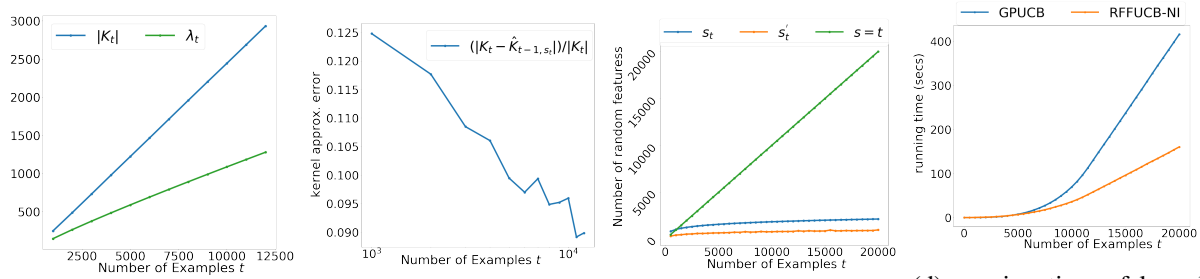
The total computational cost $C(T)$ over T time-steps of Algorithm 5 has the data dependent bound,

$$C(T) = O\left(\sum_{t=1}^{T}(Nc_t^2 + c_t^3)\right),$$

where $c_t = \max\left(22\log^2(16t\log t/\delta), \zeta\left(\frac{\sqrt{\beta_t}\sigma_{m,t-1}}{(B+\sigma)\log^2 t}, \frac{\delta}{t^2}\right)\right)$
and $\sigma_{m,t} = \arg\min_{a\in\mathcal{A}} k(\mathbf{x}_{a,t}, \mathbf{x}_{a,t}) - \mathbf{k}_{t-1}(\mathbf{x}_{a,t})^T(\mathbf{K}_{t-1} + \lambda_t\mathbf{I})^{-1}\mathbf{k}_{t-1}(\mathbf{x}_{a,t})$.

The proof is in Appendix 6.6.3. Note that $\sigma_{m,t}$ is the minimum variance estimate without any approximation, at time t.

Discussion Our theoretical bounds should be compared with CGP-UCB (Krause and Ong, 2011), for the finite arms setting. The regret bound in Theorem 24 is order-wise the same as Theorem 1(3) in Krause and Ong (2011). However, CGP-UCB requires a total computational cost of $O\left(\sum_{t=1}^{T}(Nt^2 + t^3)\right)$ while the data dependent computational cost bound of our algorithm is $O\left(\sum_{t=1}^{T}(Nc_t^2 + c_t^3)\right)$, which can be considerably lesser than that of CGP-UCB. This is because we can approximate the variances $\sigma_{a,t}$ by only $s_t = O(\text{polylog(t)})$ random features through spectral approximation bounds on the kernel-gram matrix. As for the kernel mean functions of each arm, we use s'_t features which are sufficient for the approximation error in the mean function not to be order-wise more than the minimum variance. In Figure 6.1, we show the data dependent random features s_t and s'_t specified by RFF-UCB-NI algorithm grow much slower than t. In fact s'_t is often dominated by s_t, which grows as $O(\text{polylog}(t))$. Consequently, RFF-UCB-NI enjoy less computational cost (in running time) compared to CGP-UCB when operating the kernel inverse. Finally, in Figure 6.1a, we show that our assumption $||\mathbf{K}_t||_2 > \lambda_t$, holds on real datasets.



(a) $\|K_t\|$ versus λ_t (b) $\|K_t - \hat{K}_{t-1,s_t}\|/\|K_t\|$ (c) #random features s_t (d) running time of kernel inverse

Figure 6.1: Running RFF-UCB-NI algorithm on letter dataset. The kernel norm assumption of Theorem 24 is verified in Figure 6.1a. Figure 6.1b plot the relative kernel approximation error. We present the data-dependent random features s_t and $s_t^{'}$ in comparison with time step t. Lastly, we compare the running time of inversing the kernel matrix in GP-UCB and RFF-UCB-NI algorithms.

6.3.3 RFF-UCB: A Fast Incremental Algorithm

Inspired by the theoretical analysis of RFF-UCB-NI, we introduce a faster incremental version, namely RFF-UCB. We consider the setting where the context $\mathbf{x}_t$ is shared among the arms, but the reward function $f_a(\cdot)$ is different for different arms. LinUCB (Li et al., 2010) and its kernel variants (Krause and Ong, 2011) are commonly evaluated in this setting in practice. In the linear case, this setting is equivalent to fitting a single function with context being dependent on actions. We first note that, with the Woodbury identity (Petersen et al., 2008) and the PSD property of kernel matrix, the predictive mean $\hat{\mu}_{a,t}$ and confidence widths $\hat{\sigma}_{a,t}$ of the GP-UCB at time step t can be approximated by s RFF as:

$$\hat{\mu}_{a,t} \approx \psi_s(\mathbf{x}_t)^T \mathbf{C}_{a,t}^{-1} \mathbf{Z}_{a,t}^T \mathbf{y}_{a,t} \tag{6.8}$$

$$\hat{\sigma}_{a,t} \approx \sqrt{\psi_s(\mathbf{x}_t)^T \mathbf{C}_{a,t}^{-1} \psi_s(\mathbf{x}_t)}. \tag{6.9}$$

where $\mathbf{Z}_{a,t}$ is the RFF matrix at time t for arm a, whose rows correspond to $m_a(t)$ RFF inputs (e.g., $m_a(t)$ contexts that are observed previously for action a), and define $\mathbf{C}_{a,t} \coloneqq (\mathbf{Z}_{a,t}^T \mathbf{Z}_{a,t} + \lambda_t \mathbf{I})$. Note that with linear kernel, $\psi_s(\mathbf{x}_t) = \mathbf{x}_t$, these equations reduce to the LinUCB algorithm.

The remaining questions are how to set the scheduling rule of random features s_t, and how to maintain fast incremental update of model parameters with new instances as well as random features dynamically increasing. In RFF-UCB-NI, we saw that s_t grew logarithmically in t. In RFF-UCB, we set

$$s_t = \alpha \cdot \lfloor \log_2 t \rfloor + s_0 \tag{6.10}$$

where α, s_0 is the slope and intercept. We found this setting to be sufficient for kernel approximation in practice, as later supported by our empirical experiments. This suggests that our bound on s_t in Theorem 24 can be possibly improved, which we leave as future work. Also, note that in practice, we set $\lambda_t = \lambda$ i.e a constant, unlike in RFF-UCB-NI.

We now present the RFF-UCB outlined in Algorithm 6. Specifically, when a new instance/context $\mathbf{x}_t$ arrives, we use the Sherman Morrison formula (Petersen et al., 2008) to efficiently compute the inverse of $\mathbf{C}_t^{-1}$ via a rank-one update:

$$\left(\mathbf{C}_t + \psi(\mathbf{x}_t)\psi(\mathbf{x}_t)^T\right)^{-1} = \mathbf{C}_t^{-1} + \frac{\mathbf{C}_t^{-1}\psi(\mathbf{x}_t)\psi(\mathbf{x}_t)^T\mathbf{C}_t^{-1}}{1 + \psi(\mathbf{x}_t)^T\mathbf{C}_t^{-1}\psi(\mathbf{x}_t)}$$

Similarly, when sampling new random features, we consider the Schur complement as well as the Woodbury identity to efficiently update the inverse $\mathbf{C}_t^{-1}$ without computing it from scratch expensively. More detailed derivations are available in Appendix 6.6.1.

Algorithm 6: RFF-UCB with fast incremental update

for $t = 1, 2, \ldots, T$ **do**

Observe context features $\mathbf{x}_t \in \mathbb{R}^d$ at time t.

// Incremental update model for additional RFFs

$s_t = \alpha \cdot \lfloor \log_2 t \rfloor + s_0$

if $s_t > s_{t-1}$ **then**

Construct $\tilde{\mathbf{Z}}_{a,t}$ with new $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_\alpha \sim p_k(\cdot)$

$\Theta_{a,t-1} \leftarrow \mathsf{UpdateFeature}(\Theta_{a,t-1}, \tilde{\mathbf{Z}}_{a,t}),\ \forall a \in \mathcal{A}$

// prediction to select arm

for $a = 1, \ldots, N$ **do**

$\hat{\mu}_{a,t} \leftarrow \psi_s(\mathbf{x}_t)^T \mathbf{C}_{a,t}^{-1} \mathbf{Z}_{a,t}^T \mathbf{y}_{a,t}$

$\hat{\sigma}_{a,t} \leftarrow \sqrt{\psi_s(\mathbf{x}_t)^T \mathbf{C}_{a,t}^{-1} \psi_s(\mathbf{x}_t)}$

$p_{a,t} \leftarrow \mu_{a,t} + \beta \cdot \hat{\sigma}_{a,t}$

Select arm $a_t = \arg\max_{a \in \mathcal{A}} p_{a,t}$ with ties broken arbitrarily, and Observe a real-valued reward $y_{a,t}$

// Incremental update model for additional instance

$\Theta_{a,t} \leftarrow \mathsf{UpdateInstance}(\Theta_{a,t-1}, \psi(\mathbf{x}_t), y_{a,t})$

Time Complexity We analyze the computational cost of RFF-UCB, which can be divided into three terms: 1) mean and confidence bound estimation, 2) the UpdateInstance () function, and 3) the UpdateFeature () function. For each time step t, it is straightforward to see that computing 1) and 2) together takes $O(Ns_t^2)$ since $\mathbf{Z}^T\mathbf{y}$ can be maintain incrementally. Note that $O(Ns_t^2)$ can be as cheap as LinUCB if s_t equals to data dimension as constant. For the UpdateFeature () function, the main complexity is $O\big((ts_t + s_t^2)\alpha\big)$. which is dominated by the first term $O(ts_t\alpha)$. Putting all together, the total cost up to T-trail is $\sum_{t=1}^{T} O(Ns_t^2) + \sum_{k=1}^{k \leq logT} \big(\alpha 2^k \log k\big)$, which can be simplified as $O(\alpha NT \log^2 T)$ in total. Note that the computational cost of RFF-UCB is close to that of LinUCB, which is $O(NTd^2)$, if $d = O(\log T)$.

6.4 Experiment and Results

We compare the proposed RFF-UCB algorithm with representative methods across realizability-based alternatives of UCB variants, as well as competitive methods in agnostic-based framework. Detailed descriptions of the datasets, benchmark algorithms, as well as further experimental results are included in Appendix 6.6.4.

Datasets Following Agarwal et al. (2014); Foster et al. (2018); Sen et al. (2018), we study on six multi-class classification datasets using a transformation from a supervised setting to a contextual bandit setting Beygelzimer et al. (2011); Dudík et al. (2011). This evaluation strategy has been widely used in contextual bandit literature Agarwal et al. (2014); Foster et al. (2018); Sen et al. (2018). Specifically, At each time-step, the feature (context) of an instance is revealed, following which the contextual bandit algorithm chooses one of the N classes, and the reward observed is 1 if its the correct class otherwise it is 0. This is bandit feedback as the correct class is never revealed, if not chosen. Table 6.1 summarizes the dataset statistics. Since LinUCB is slow on high-dimensional dataset (i.e., CIFAR-100), we use the pretrained VGG16 model Simonyan and Zisserman (2014)[1] for feature extraction that yielded a low dimension feature space $\tilde{d}$.

Dataset	T	N	d	$\tilde{d}$
letter	20,000	26	16	N/A
sensor	58,509	11	48	N/A
MNIST	60,000	10	780	N/A
CIFAR-100	60,000	100	3,072	512
shuttle	58,000	7	9	N/A
covtype	581,012	7	54	N/A

Table 6.1: Data statistics: T the number of instances, N the number of classes, d the number of unique features. $\tilde{d}$ the number of unique features after dimension reduction. N/A denotes for not applying dimension reduction.

Algorithms We evaluate RFF-UCB against six competitive baselines. The first two are realizability-based methods: LinUCB Li et al. (2010) and CGP-UCB Krause and Ong (2011). LinUCB is equivalent to the RFF-UCB without using random Fourier features but using the original features instead. CGP-UCB is equivalent to the dual counterpart of RFF-UCB when using infinite random features.

The remaining baselines, borrowed from Bietti et al. (2018), are agnostic-based frameworks, including ϵ-Greedy Langford and Zhang (2008), Online-Cover Agarwal et al. (2014), Bagging Bietti et al. (2018) (explores a basket of policies using posterior updates and Thompson sampling), and RegCB Foster et al. (2018) (online optimistic algorithm). For additional discussion on agnostic-based approaches, we refer to Bietti et al. (2018).

Evaluation Each dataset is split into training set and test set. All hyper-parameters are tuned via five folds cross-validation on the training set, see Appendix 6.6.4 for more details on the hyper-parameters and the range of grid search. We conduct online evaluation on the test set with cumulative mean rewards as the evaluation metric. The implemented algorithms (LinUCB, CGP-UCB and RFF-UCB) are run in mini-batch setting with batch size 100 i.e learned parameters are only updated every 100 time-steps. All results are averaged of 10 runs with different random seeds.

[1] https://github.com/geifmany/cifar-vgg

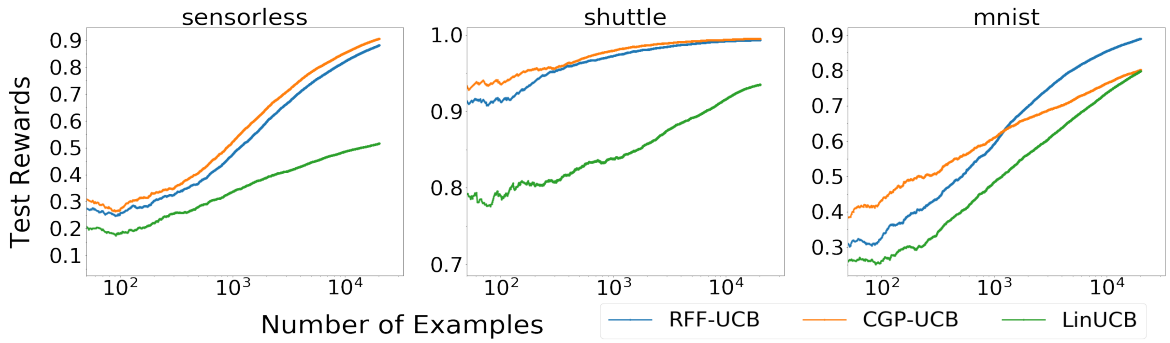


Figure 6.2: Test reward comparison of realizability-based methods: RFF-UCB, LinUCB and CGP-UCB methods. The y-axis is test rewards and x-axis is number of examples (rounds). Results are averaged under 10 different random seeds.

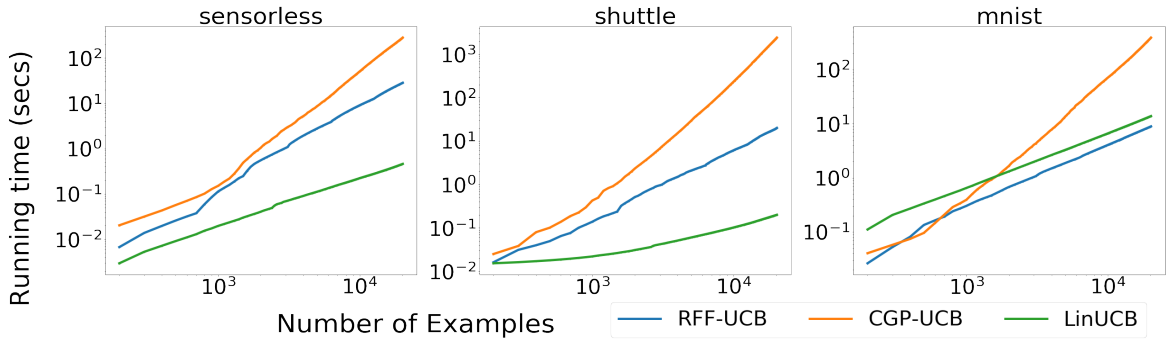


Figure 6.3: Running time comparison of realizability-based methods: RFF-UCB, LinUCB and CGP-UCB algorithms. The y-axis is running time and x-axis is number of examples (rounds). Results are averaged under 10 different random seeds.

Comparison to realizability-based models Figure 6.2 and Figure 6.3 show the performance of the methods in cumulative mean reward and their running time curves on three representative datasets, respectively. More results on the remaining datasets are available in Appendix 6.6.4 due to the space limits.

The proposed RFF-UCB performs almost equally as good as CGP-UCB, suggesting the number of random features s_t for kernel approximation is sufficient. Importantly, the number of random features (i.e., #RFF) $s_t = \alpha \cdot \lfloor \log_2 t \rfloor + s_0$ controls the trade-off between kernel approximation quality and computational efficiency. In one extreme, when s_t is equal to the feature dimension d, then the computational cost of RFF-UCB is almost the same as LinUCB but the performance of the method would be sub-optimal due to inaccurate kernel approximation. In the other extreme, when s_t is in the same order of $O(t)$, although enjoying good kernel approximation, there's no saving in the computation cost, ending with same time complexity as in CGP-UCB.

With reasonable choices of s_0 and α by our scheduling rule (6.10), the performance of RFF-UCB is very competitive in comparison with CGP-UCB while enjoying significantly less running time than CGP-UCB. As shown in Figure 6.3 (and Figure 6.6 in the Appendix), RFF-UCB has a speedup of more than $10x$ over to CGP-UCB in regions with a large T (e.g. $T > 10^3$). Interestingly, for high dimensional

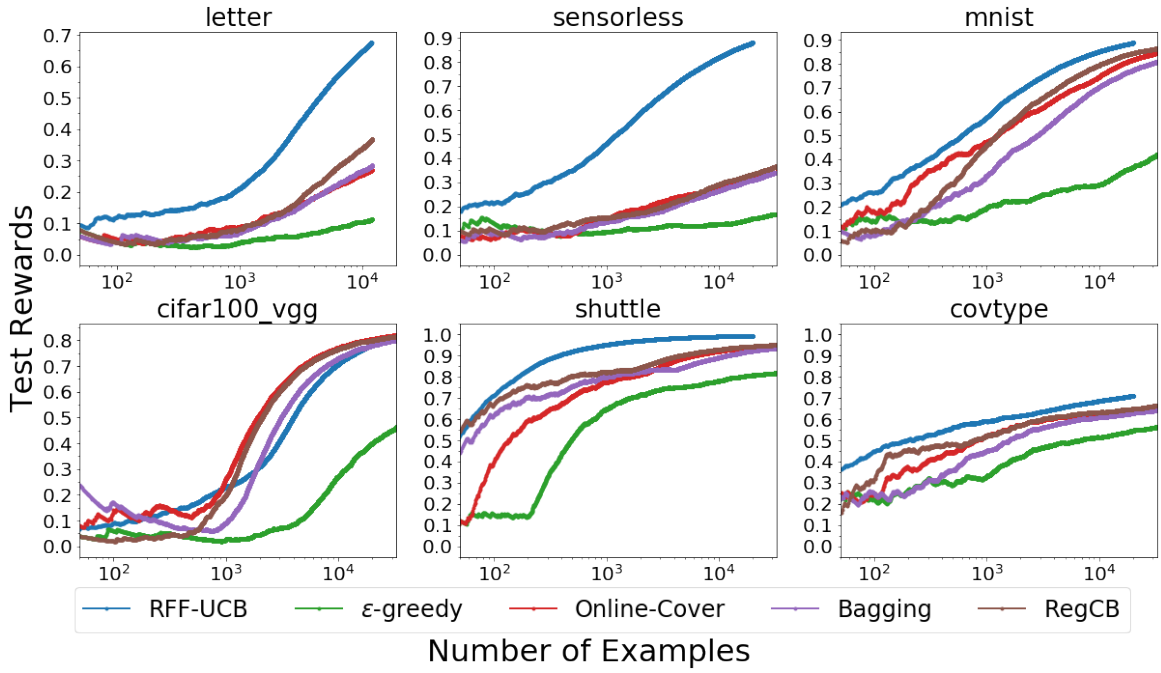


Figure 6.4: Test reward comparison of agnostic-based approaches: ϵ-Greedy, Online-Cover, Bagging, and RegCB algorithms. The y-axis is test rewards and x-axis is number of examples (rounds). Results are averaged under 10 different random seeds.

datasets such as MNIST and CIFAR-100, the running time of RFF-UCB becomes very close to or even faster than that of LinUCB. In other words, when $s_t = O(\log t)$ is smaller than feature dimension d, RFF-UCB can actually performs as good as the exact kernel UCB algorithm CGP-UCB while maintaining the low computation cost as the linear model LinUCB. A more comprehensive study of different scheduling rule for s_t is presented in Appendix 6.6.4.

Comparison to agnostic-based approaches We now show how RFF-UCB performs in comparison with other state-of-the-art contextual bandit solvers implemented in the Vowpal Wabbit (VW) library. Figure 6.4 shows the test-set cumulative mean rewards on all the six datasets.

Overall, the proposed RFF-UCB algorithm outperforms all agnostic-based algorithms except on the CIFAR-100 dataset; the second best algorithm is RegCB, which has performed strongly in the contextual bandit literature Foster et al. (2018); Bietti et al. (2018). A crucial fact worth noticing is that all the agnostic-based algorithms implemented in VW only support linear reward functions, with the choice of bagging over multiple learners. Thus, it is less surprising to see the kernel contextual bandit algorithms such as RFF-UCB and CGP-UCB with non-linear reward functions to have superior performance over those with linear oracles. A supporting evidence of the above argument can be found in the CIFAR-100 results of Figure 6.4. Note that for feature extraction from CIFAR-100 we used a pretrained deep neural network with VGG16 architecture Simonyan and Zisserman (2014). The extracted feature representations already provide good classification accuracy through a logistic output layer in the original model. In this case, the advantage of kernel contextual bandit algorithms becomes less clear.

Finally, we omit the running time comparison between RFF-UCB and the agnostic-based frameworks implemented in Vowpal Wabbit. It is important to note that our implementation of RFF-UCB is based on Python with Numpy and Scipy package while Vowpal Wabbit library is implemented in C/C++ with highly optimized packages, thus making it hard to compare the running times.

6.5 Summary

In this chapter, we present computational efficient kernel UCB algorithms via random Fourier features. For the non-incremental RFF-UCB-NI algorithm, we not only prove the same order of regret bound as of CGP-UCB, but also derive a data dependent bound for the computational cost, which is much less than that of non-approximated kernel UCB algorithms. Motivated by this theory, we propose a fast incremental algorithm (RFF-UCB) and show that it empirically performed comparably or better than the other representative baseline methods on multiple benchmark datasets, while being more than 10x faster compared to other kernel UCB algorithms.

6.6 Appendix

6.6.1 Derivation Details of RFF-UCB Online Update

Algorithm 7: UpdateFeature$(\Theta, \tilde{Z})$ via Schur Complement

Define $\mathbf{B} \coloneqq \mathbf{Z}^T\tilde{\mathbf{Z}}$ ($s \times r$ matrix)
Define $\mathbf{D} \coloneqq \tilde{\mathbf{Z}}^T\tilde{\mathbf{Z}}$ ($r \times r$ matrix)

// update $\mathbf{Z}$ *and* $\mathbf{C}$
$\mathbf{Z} \leftarrow [\mathbf{Z}, \tilde{\mathbf{Z}}]$
$\mathbf{C}_{11} \leftarrow \mathbf{C}, \quad \mathbf{C}_{22} \leftarrow \mathbf{D}$
$\mathbf{C}_{12} \leftarrow \mathbf{B}, \quad \mathbf{C}_{21} \leftarrow \mathbf{B}^T$
$\mathbf{C}^{new} \leftarrow [\mathbf{C}_{11}, \mathbf{C}_{12}; \mathbf{C}_{21}, \mathbf{C}_{22}]$

// update $\mathbf{Z}^T\mathbf{y}$ *and* $\mathbf{C}^{-1}$
$\mathbf{Z}^T\mathbf{y} \leftarrow [\mathbf{Z}^T\mathbf{y}; \tilde{\mathbf{Z}}^T\mathbf{y}], \quad \mathbf{M} \coloneqq \mathbf{D} - \mathbf{B}^T\mathbf{C}^{-1}\mathbf{B}$
$\mathbf{C}_{11}^{-1} \leftarrow \mathbf{C}^{-1} + (\mathbf{C}^{-1}\mathbf{B})\mathbf{M}^{-1}(\mathbf{C}^{-1}\mathbf{B})^T$
$\mathbf{C}_{12}^{-1} \leftarrow -\mathbf{C}^{-1}\mathbf{B}\mathbf{M}^{-1}, \quad \mathbf{C}_{21}^{-1} \leftarrow -\mathbf{M}^{-1}\mathbf{B}\mathbf{C}^{-1}$
$\mathbf{C}_{22}^{-1} \leftarrow \mathbf{M}^{-1}$
$\mathbf{C}^{-1} \leftarrow [\mathbf{C}_{11}^{-1}, \mathbf{C}_{12}^{-1}; \mathbf{C}_{21}^{-1}, \mathbf{C}_{22}^{-1}]$

Algorithm 8: UpdateInstance$(\Theta, \psi(\mathbf{x}_t), y_t)$ via Sherman-Morrison rank-one update

$\mathbf{C} \leftarrow \mathbf{C} + \psi(\mathbf{x}_t)\psi(\mathbf{x}_t)^T$
Define $\mathbf{v}_t \coloneqq \mathbf{C}^{-1}\psi_s(\mathbf{x}_t)$
$\mathbf{C}^{-1} \leftarrow \mathbf{C}^{-1} - (1 + \hat{\sigma}_t^2)^{-1}\mathbf{v}_t\mathbf{v}_t^T$
$\mathbf{Z}^T\mathbf{y} \leftarrow \mathbf{Z}^T\mathbf{y} + y_t \cdot \psi_s(\mathbf{x}_t)$
$\mathbf{Z} \leftarrow [\mathbf{Z}; \psi_s(\mathbf{x}_t)^T]$
$\mathbf{y} \leftarrow [\mathbf{y}; y_t]$

6.6.2 Kernel Approximation Guarantees

In this section, we introduce two main kernel approximation results. The first one is adapted from a bound on the approximation error of kernel ridge regression in Cortes et al. (2010b). This will be useful in bounding the approximation gaps in the posterior kernel mean function in our algorithm.

Theorem 25 (Cortes et al. (2010b)). *Let* $\mathbf{K}_t \in \mathbb{R}^{t\times t}$ *denote the kernel gram matrix and* $\hat{\mathbf{K}}_{t,s}$ *be its approximation using* s *random features. Let* $\mu_t(\mathbf{x})$ *be as defined in Eq. 6.1. and let* $\hat{\mu}_{t,s}(\mathbf{x})$ *be its approximated version with* $\mathbf{K}_t$ *replaced by* $\mathbf{K}_{t,s}$*. Then, we have the following,*

$$|\hat{\mu}_{t,s}(\mathbf{x}) - \mu_t(\mathbf{x})| \leq \frac{(B+\sigma)t||\hat{\mathbf{K}}_{t,s} - \mathbf{K}_t||_2}{\lambda_t^2} \leq \frac{(B+\sigma)t^2||\hat{\mathbf{K}}_{t,s} - \mathbf{K}_t||_\infty}{\lambda_t^2} \tag{6.11}$$

The next theorem provides spectral approximation bounds for the random Fourier features.

Theorem 26 (Avron et al. (2017b)). *Let $\mathbf{K}_t$ be the kernel gram matrix at time t and $\hat{\mathbf{K}}_{t,s}$ be its approximation using s random features. Then we have,*

$$(1-\Delta)(\mathbf{K}_t + \lambda_t \mathbf{I}) \preccurlyeq (\hat{\mathbf{K}}_{t,s} + \lambda_t \mathbf{I}) \preccurlyeq (1+\Delta)(\mathbf{K}_t + \lambda_t \mathbf{I}) \tag{6.12}$$

with probability at least $1-\delta$, provided $s \geq \frac{8}{3\Delta^2}\frac{t}{\lambda_t}\log\left(16\frac{t}{(\lambda_t \delta)}\right)$ and $||\mathbf{K}_t||_2 \geq \lambda_t$.

This leads us to our first lemma which says that with the choice of number of random features in Algorithm 5, the difference between the true variance and approximated are uniformly bounded.

Lemma 27. *Let $\sigma^2_{a,t} := k(\mathbf{x}_{a,t}, \mathbf{x}_{a,t}) - \mathbf{k}_{t-1}(\mathbf{x}_{a,t})^T(\mathbf{K}_{t-1} + \lambda_t \mathbf{I})^{-1}\mathbf{k}_{t-1}(\mathbf{x}_{a,t})$ be the actual variance estimates for all $a \in \mathcal{A}$. Let $\hat{\sigma}^2_{a,t}$ be the approximated one, as defined in Algorithm 5. Then, we have the following,*

$$\mathbb{P}\left(\frac{1}{2}\sigma^2_{a,t} \leq \hat{\sigma}^2_{a,t} \leq \frac{3}{2}\sigma^2_{a,t},\ \forall a \in \mathcal{A}, t \in [T]\right) \geq 1 - \frac{\pi^2}{6}\delta$$

Proof. This follows directly from Theorem 26. Recall the definition of s_t and substitute δ/t^2 in place of δ in Theorem 26. Putting in $\Delta = 1/2$, and observing that $s_t \geq \frac{8}{3\Delta^2}\frac{t}{\lambda_t}\log\left(16\frac{t}{(\lambda_t(\delta/t^2))}\right)$, we have that,

$$\begin{aligned}
&\frac{1}{2}\mathbf{k}_{t-1}(\mathbf{x}_{a,t})^T(\mathbf{K}_{t-1} + \lambda_t \mathbf{I})^{-1}\mathbf{k}_{t-1}(\mathbf{x}_{a,t}) \leq \mathbf{k}_{t-1}(\mathbf{x}_{a,t})^T(\hat{\mathbf{K}}_{t-1,s_t} + \lambda_t \mathbf{I})^{-1}\mathbf{k}_{t-1}(\mathbf{x}_{a,t}) \\
&\leq \frac{3}{2}\mathbf{k}_{t-1}(\mathbf{x}_{a,t})^T(\mathbf{K}_{t-1} + \lambda_t \mathbf{I})^{-1}\mathbf{k}_{t-1}(\mathbf{x}_{a,t}),\ \text{for all } a \in \mathcal{A}
\end{aligned}$$

with probability at least $1 - \frac{\delta}{t^2}$. Substituting the above in the expression of $\hat{\sigma}^2_{a,t}$ and $\sigma^2_{a,t}$ and applying an union bound over all t yields the lemma, since $\sum(1/t^2) \leq \pi^2/6$. □

This brings us to our second key approximation lemma which bounds the difference in the approximation of the kernel mean function.

Lemma 28. *Let $\mu_{a,t} = \mathbf{k}_{t-1}(\mathbf{x}_{a,t})^T(\mathbf{K}_{t-1} + \lambda_t \mathbf{I})^{-1}\mathbf{y}_t$, be the actual mean estimate for arm $a \in \mathcal{A}$. Then for the choice of parameters in Algorithm 5, we have the following bound,*

$$\mathbb{P}\left(|\hat{\mu}_{a,t} - \mu_{a,t}| \leq \sqrt{\frac{3}{2}\beta_t}\sigma_{m,t}\ \ \forall a \in \mathcal{A}, t \in [T]\right) \geq 1 - \frac{\pi^2}{3}\delta.$$

Proof. By Theorem 23, and the choice of s'_t in Algorithm 5, we have that,

$$\begin{aligned}
||\hat{\mathbf{K}}_{t,s} - \mathbf{K}_t||_\infty &\leq \frac{\sqrt{\beta_t}\hat{\sigma}_{m,t}}{(B+\sigma)\log^2 t} \\
&\leq \frac{\sqrt{\frac{3}{2}\beta_t}\sigma_{m,t}}{(B+\sigma)\log^2 t},
\end{aligned}$$

with probability $1 - \frac{\delta}{t^2}$, given that the h.p event in Lemma 27 holds. Therefore, by Theorem 25, we have,

$$|\hat{\mu}_{a,t} - \mu_{a,t}| \leq \sqrt{\frac{3}{2}\beta_t}\sigma_{m,t}\ \ \forall a \in \mathcal{A}$$

w.p $1 - \frac{\delta}{t^2}$, given that the h.p event in Lemma 27 holds. Let E_1 be the h.p event in Lemma 27. Then, we have that,

$$\begin{aligned}
&\mathbb{P}\left(|\hat{\mu}_{a,t} - \mu_{a,t}| > \sqrt{\frac{3}{2}\beta_t}\sigma_{m,t} \text{ for any } a \in \mathcal{A}, t \in [T]\right) \\
\leq &\mathbb{P}\left(|\hat{\mu}_{a,t} - \mu_{a,t}| > \sqrt{\frac{3}{2}\beta_t}\sigma_{m,t} \text{ for any } a \in \mathcal{A}, t \in [T] \mid E_1\right) + \mathbb{P}(E_1^c) \\
\leq &\sum_{t=1}^{T} \frac{\delta}{t^2} + \frac{\pi^2}{6}\delta \leq \frac{\pi^2}{3}\delta.
\end{aligned}$$

This completes the proof. □

6.6.3 Regret Bound

In this section, we first show that the true mean function can closely approximate the reward function f, and then go on to show that the approximated mean function is not too far away from the true kernel mean estimates. This is sufficient to show the final regret bound.

Theorem 29 (Srinivas et al. (2010))**.** *Consider the true kernel mean estimate* $\mu_{a,t} = \mathbf{k}_{t-1}(\mathbf{x}_{a,t})^T(\mathbf{K}_{t-1} + \lambda_t \mathbf{I})^{-1}\mathbf{y}_t$. *We have the following high-probability bounds,*

$$\mathbb{P}\left(|f(\mathbf{x}_{a,t}) - \mu_{a,t}| \leq \sqrt{\beta_t}\sigma_{a,t} \ \ \forall a \in \mathcal{A}, t \in [T]\right) \geq 1 - \delta.$$

Now, we are ready to prove our main theorem.

Proof of Theorem 24. From Theorem 29 and Lemma 28 we have that,

$$\begin{aligned}
&\mathbb{P}\left(|f(\mathbf{x}_{a,t}) - \hat{\mu}_{a,t}| \leq \left(1 + \sqrt{\frac{3}{2}}\right)\sqrt{\beta_t}\sigma_{a,t} \ \ \forall a \in \mathcal{A}, t \in [T]\right) \geq 1 - \left(1 + \frac{\pi^2}{3}\right)\delta \\
&\implies \mathbb{P}\left(|f(\mathbf{x}_{a,t}) - \hat{\mu}_{a,t}| \leq 3\sqrt{\beta_t}\sigma_{a,t} \ \ \forall a \in \mathcal{A}, t \in [T]\right) \geq 1 - 7\delta
\end{aligned}$$

Let $E = \left\{|f(\mathbf{x}_{a,t}) - \hat{\mu}_{a,t}| \leq 3\sqrt{\beta_t}\sigma_{a,t} \ \ \forall a \in \mathcal{A}, t \in [T]\right\}$ be the h.p event in the equation above. Let a_t be the arm chosen at time t. Given E, we have the following,

$$\hat{\mu}_{a_t,t} + 3\sqrt{2\beta_t}\hat{\sigma}_{a_t,t} \geq \hat{\mu}_{a^*,t} + 3\sqrt{2\beta_t}\hat{\sigma}_{a^*,t} \geq \hat{\mu}_{a^*,t} + 3\sqrt{\beta_t}\sigma_{a^*,t} \geq f(\mathbf{x}_{a^*,t}).$$

This implies that,

$$r_t = f(\mathbf{x}_{a^*,t}) - f(\mathbf{x}_{a_t,t}) \leq 3(1 + \sqrt{3})\sqrt{\beta}\sigma_{a_t,t} \leq 9\sqrt{\beta}\sigma_{a_t,t}$$

□

Therefore, given E, we have, $\sum_{t=1}^{T} r_t^2 \leq \sum_{t=1}^{T} 81\beta_t\sigma^2(\mathbf{x}_{a_t,t})$. Now, we have that,

$$\begin{aligned}
\beta_t\sigma^2(\mathbf{x}_{a_t,t}) &\leq \beta_T\lambda_t(\lambda_t^{-1}\sigma^2(\mathbf{x}_{a_t,t})) \leq \beta_T\frac{1}{\log(1 + \lambda_t^{-1})}\log(1 + \lambda_t^{-1}\sigma^2(\mathbf{x}_{a_t,t})) \\
&\leq \beta_T\frac{1}{\log(1 + \lambda_T^{-1})}\log(1 + \lambda_t^{-1}\sigma^2(\mathbf{x}_{a_t,t}))
\end{aligned}$$

since $s^2 \leq C\log(1 + s^2)$, for $s \in [0, \lambda_t^{-1}]$, where $C = \lambda_t^{-1}/\log(1 + \lambda_t^{-1})$. Note that $\sigma^2(\mathbf{x}_{a_t,t}) \leq 1$.

The result follows from the fact that $R(T)^2 \leq T\sum_{t=1}^{T} r_t^2$.

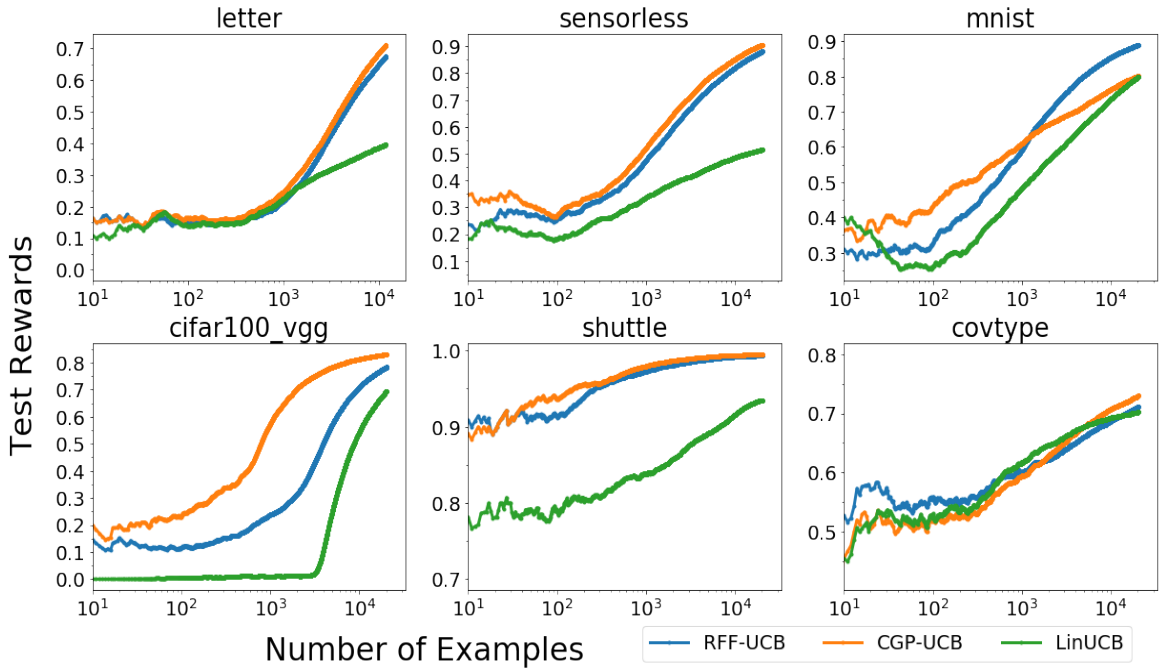


Figure 6.5: Test Rewards of RFF-UCB versus GP-UCB

6.6.4 Additional Experiment settings and Results

Hyper-parameters For agnostic-based approaches, we consider the implementation of the well-known online learning system Vowpal Wabbit (VW)[2] with the linear predictive class. For ϵ-Greedy, we tune the constant $\epsilon \in \{0.01, 0.05, 0.10, 0.15, 0.25, 0.30, 0.35\}$. For Online-Cover and Bagging, we tune number of policies $M \in \{1, 2, 4, 8, 16, 32\}$ and set the reduction method to be doubly robust estimator. For RegCB, we search the mellowness parameter $C_0 \in 10^{\{-1,-2,-3,-4\}}$ and tune the reduction method from either DR or MTR. For realizability-based algorithms, we implement LinUCB, CGP-UCB, RFF-UCB and tune the exploration coefficient $\beta \in \{0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45\}$. For kernel UCB algorithms CGP-UCB and RFF-UCB, we first compute the kernel bandwidth using medium heuristic , denoted as γ_0, and search the best bandwidth $\gamma = \gamma_0 \cdot c_0$ where $c_0 \in \{0.25, 0.50, 1, 2, 4\}$.

Additional Results We present additional experiment results of the realizability-based methods, as shown in Figure 6.5 and Figure 6.6. In general, RFF-UCB is close to the performance of CGP-UCB, while being more than 10x faster most cases, except on the letter dataset. On covtype and shuttle dataset, RFF-UCB is even 100x faster than CGP-UCB.

Figure 6.7 and Figure 6.8 demonstrate RFF-UCB algorithm set by a variety choices of (α, s_0) appeared in the scheduling rule s_t (Eq. (6.10)). Specifically, we examine $\alpha = \{32, 64, 128, 256\}$ and $s_0 = \{128, 256, 512\}$.

[2] http://hunch.net/~vw/

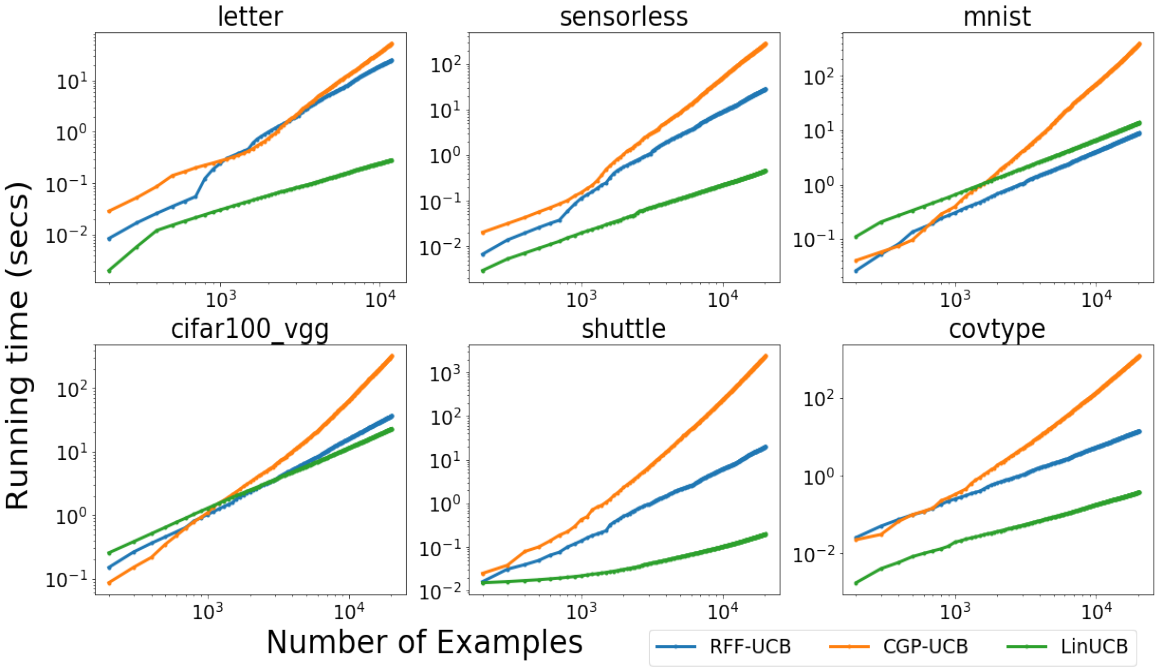


Figure 6.6: Running time of RFF-UCB versus GP-UCB

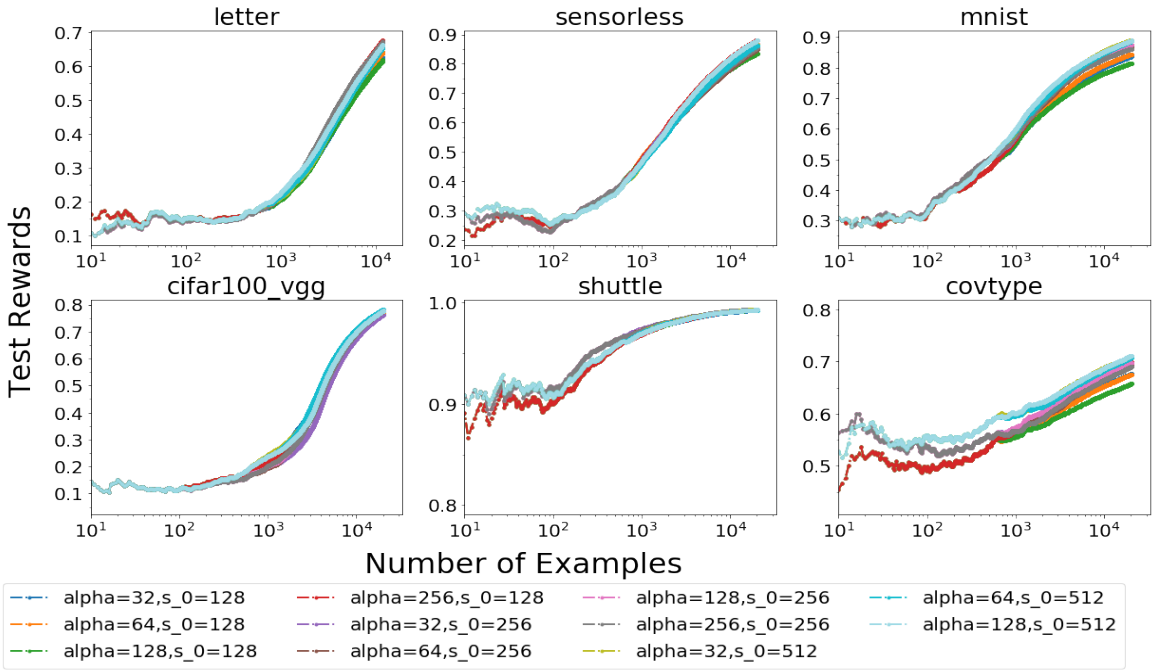


Figure 6.7: Test Rewards of RFF-UCB versus GP-UCB

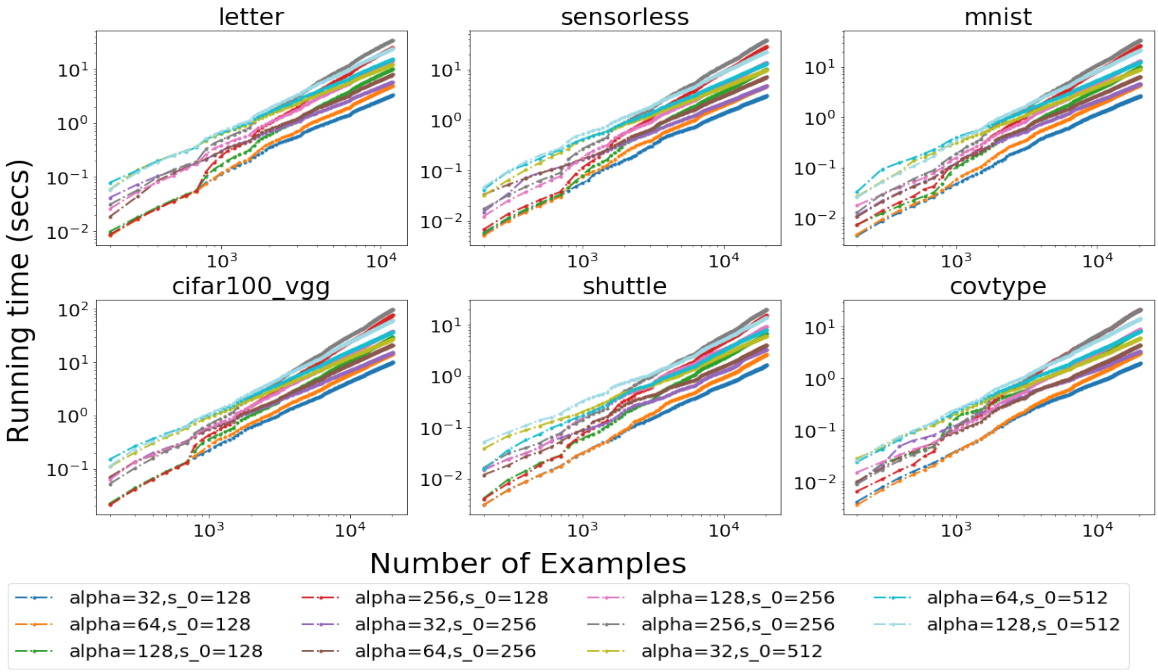


Figure 6.8: Running time of RFF-UCB versus GP-UCB

Part IV

Extensions to Time-Series and Graph-based Learning

Chapter 7

Kernel Change-point Detection

7.1 Background

Detecting changes in the temporal evolution of a system (biological, physical, mechanical, etc.) in time series analysis has attracted considerable attention in machine learning and data mining for decades (Basseville et al., 1993; Brodsky and Darkhovsky, 2013). This task, commonly referred to as change-point detection (CPD) or anomaly detection, aims to predict significant changing points in a temporal sequence of observations. CPD has a broad range of real-world applications such as medical diagnostics (Gardner et al., 2006), industrial quality control (Basu and Meckesheimer, 2007), financial market analysis (Pepelyshev and Polunchenko, 2015), video anomaly detection (Liu et al., 2018) and more.

As shown in Figure 7.1, we focus on the retrospective CPD (Takeuchi and Yamanishi, 2006; Li et al., 2015a), which allows a flexible time window to react on the change-points. Retrospective CPD not only enjoys more robust detection (Chandola et al., 2009) but embraces many applications such as climate change detection (Reeves et al., 2007), genetic sequence analysis (Wang et al., 2011), networks intrusion detection (Yamanishi et al., 2004), to name just a few. Various methods have been developed (Gustafsson and Gustafsson, 2000), and many of them are parametric with strong assumptions on the distributions (Basseville et al., 1993; Gustafsson, 1996), including auto-regressive models (Yamanishi and Takeuchi, 2002) and state-space models (Kawahara et al., 2007) for tracking changes in the mean, the variance, and the spectrum.

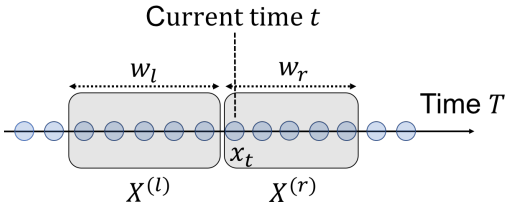


Figure 7.1: A sliding window over the time series input with two intervals: the past and the current, where w_l, w_r are the size of the past and current interval, respectively. $X^{(l)}, X^{(r)}$ consists of the data in the past and current interval, respectively.

Ideally, the detection algorithm should be free of distributional assumptions to have robust performance as neither true data distributions nor anomaly types are known a priori. Thus the parametric assumptions in many works are unavoidably a limiting factor in practice. As an alternative, nonparametric and kernel approaches are free of distributional assumptions and hence enjoy the advantage to produce more robust performance over a broader class of data distributions.

Kernel two-sample test has been applied to time series CPD with some success. For example, Harchaoui et al. (2009) presented a test statistic based upon the maximum kernel fisher discriminant ratio for hypothesis testing and Li et al. (2015a) proposed a computational efficient test statistic based on maximum mean discrepancy with block sampling techniques. The performance of kernel methods, nevertheless, relies heavily on the choice of kernels. Gretton et al. (2007, 2012a) conducted kernel selection for RBF kernel bandwidths via median heuristic. While this is certainly straightforward, it has no guarantees of optimality regarding to the statistical test power of hypothesis testing. Gretton et al. (2012b) show explicitly optimizing the test power leads to better kernel choice for hypothesis testing under mild conditions. Kernel selection by optimizing the test power, however, is not directly applicable for time series CPD due to insufficient samples, as we discuss in the following.

Notations and Settings Given a sequence of d-dimensional observations $\{x_1, \ldots, x_t, \ldots\}, x_i \in \mathbb{R}^d$, our goal is to detect the existence of a change-point[1] such that before the change-point, samples are i.i.d from a distribution $\mathbb{P}$, while after the change-point, samples are i.i.d from a different distribution $\mathbb{Q}$.

Notice that there are multiple settings for change point detection (CPD) where samples could be piecewise iid, non-iid autoregressive, and more. It is truly difficult to come up with a generic framework to tackle all these different settings. In this work, following the previous CPD works (Harchaoui et al., 2009; Kawahara et al., 2007; Matteson and James, 2014; Li et al., 2015a), we stay with the piecewise iid assumption of the time series samples.

Two-sample Test Hypothesis test offers thorough statistical guarantees of whether two finite sample sets are the same distribution. Following Gretton et al. (2012a), the hypothesis test is defined by the null hypothesis $H_0 : \mathbb{P} = \mathbb{Q}$ and alternative $H_1 : \mathbb{P} \neq \mathbb{Q}$, using test statistic $m\hat{M}_k(X, Y)$. **peter: provide a hyper-link to MMD definition!** For a given allowable false rejection probability α (i.e., false positive rate or Type I error), we choose a test threshold c_α and reject H_0 if $m\hat{M}_k(X, Y) > c_\alpha$. We now describe the objective to choose the kernel k for maximizing the test power (Gretton et al., 2012b; Sutherland et al., 2017). First, note that, under the alternative $H_1 : \mathbb{P} \neq \mathbb{Q}$, $\hat{M}_k$ is asymptotically normal,

$$\frac{\hat{M}_k(X, Y) - M_k(\mathbb{P}, \mathbb{Q})}{\sqrt{V_m(\mathbb{P}, \mathbb{Q})}} \xrightarrow{\mathcal{D}} \mathcal{N}(0, 1), \tag{7.1}$$

where $V_m(\mathbb{P}, \mathbb{Q})$ denotes the asymptotic variance of the $\hat{M}_k$ estimator. The test power is then

$$\Pr\left(m\hat{M}_k(X, Y) > c_\alpha\right) \longrightarrow \Phi\Big(\frac{M_k(\mathbb{P}, \mathbb{Q})}{\sqrt{V_m(\mathbb{P}, \mathbb{Q})}} - \frac{c_\alpha}{m\sqrt{V_m(\mathbb{P}, \mathbb{Q})}}\Big) \tag{7.2}$$

where Φ is the CDF of the standard normal distribution. Given a set of kernels $\mathcal{K}$, We aim to choose a kernel $k \in \mathcal{K}$ to maximize the test power, which is equivalent to maximizing the argument of Φ.

[1] Technically speaking, two-sample test only informs whether two finite sample sets are from the same distribution or not. We abuse the change-point notation as equivalent to that two sample sets differs in distribution. One can further apply other segmentation techniques such as maximum partition strategy to locate a single change point after the detection of two-sample test.

7.2 Optimizing Test Power of Kernel CPD

In time series CPD, we denote $\mathbb{P}$ as the distribution of usual events and $\mathbb{Q}$ as the distribution for the event when change-points happen. The difficulty of choosing kernels via optimizing test power in Eq. (7.2) is that we have very limited samples from the abnormal distribution $\mathbb{Q}$. Kernel learning in this case may easily overfit, leading to sub-optimal performance in time series CPD.

7.2.1 Difficulties of Optimizing Kernels for CPD

To demonstrate how limited samples of $\mathbb{Q}$ would affect optimizing test power, we consider kernel selection for Gaussian RBF kernels on the Blobs dataset (Gretton et al., 2012b; Sutherland et al., 2017), which is considered hard for kernel two-sample test. $\mathbb{P}$ is a 5×5 grid of two-dimensional standard normals, with spacing 15 between the centers. $\mathbb{Q}$ is laid out identically, but with covariance $\frac{\epsilon_q - 1}{\epsilon_q + 1}$ between the coordinates (so the ratio of eigenvalues in the variance is ϵ_q). Left panel of Fig. 7.2 shows $X \sim \mathbb{P}$ (red samples), $Y \sim \mathbb{Q}$ (blue dense samples), $\tilde{Y} \sim \mathbb{Q}$ (blue sparse samples) with $\epsilon_q = 6$. Note that when $\epsilon_q = 1$, $\mathbb{P} = \mathbb{Q}$.

For $\epsilon_q \in \{4, 6, 8, 10, 12, 14\}$, we take 10000 samples for X, Y and 200 samples for $\tilde{Y}$. We consider two objectives: 1) *median heuristic*; 2) *max-ratio* $\eta_{k^*}(X, Y) = \arg\max_k \hat{M}_k(X, Y)/\sqrt{V_m(X, Y)}$; for choosing kernels among 20 kernel bandwidths. We repeat this process 1000 times and report the test power under false rejection rate $\alpha = 0.05$. As shown in the right panel of Fig. 7.2, optimizing kernels using limited samples $\tilde{Y}$ significantly decreases the test power compared to Y (blue curve down to the cyan curve). This result not only verifies our claim on the inaptness of existing kernel learning objectives for CPD task, but also stimulates us with the following question, *How to optimize kernels with very limited samples from $\mathbb{Q}$, even none in an extreme?*

7.2.2 A Practical Lower Bound on Optimizing Test Power

We first assume there exist a surrogate distribution $\mathbb{G}$ that we can easily draw samples from ($Z \sim \mathbb{G}$, $|Z| \gg |\tilde{Y}|$), and also satisfies the following property:

$$M_k(\mathbb{P}, \mathbb{P}) < M_k(\mathbb{P}, \mathbb{G}) < M_k(\mathbb{P}, \mathbb{Q}), \forall k \in \mathcal{K}. \tag{7.3}$$

Besides, we assume dealing with non trivial case of $\mathbb{P}$ and $\mathbb{Q}$ where a lower bound $\frac{1}{m} v_l \leq V_{m,k}(\mathbb{P}, \mathbb{Q}), \forall k$ exists. Since $M_k(\mathbb{P}, \mathbb{Q})$ is bounded, there exists an upper bound v_u. With bounded variance $\frac{v_l}{m} \leq V_{m,k}(\mathbb{P}, \mathbb{Q}) \leq \frac{v_u}{m}$ condition, we derive an lower bound $\gamma_{k*}(\mathbb{P}, \mathbb{G})$ of the test power

$$\max_{k \in \mathcal{K}} \frac{M_k(\mathbb{P}, \mathbb{Q})}{\sqrt{V_m(\mathbb{P}, \mathbb{Q})}} - \frac{c_\alpha / m}{\sqrt{V_m(\mathbb{P}, \mathbb{Q})}} \geq \max_{k \in \mathcal{K}} \frac{M_k(\mathbb{P}, \mathbb{Q})}{\sqrt{v_u/m}} - \frac{c_\alpha}{\sqrt{m v_l}} \geq \max_{k \in \mathcal{K}} \frac{M_k(\mathbb{P}, \mathbb{G})}{\sqrt{v_u/m}} - \frac{c_\alpha}{\sqrt{m v_l}} \triangleq \gamma_{k*}(\mathbb{P}, \mathbb{G}). \tag{7.4}$$

Just for now in the blob toy experiment, we artifact this distribution $\mathbb{G}$ by mimicking $\mathbb{Q}$ with the covariance $\epsilon_g = \epsilon_q - 2$. We defer the discussion on how to find $\mathbb{G}$ in the later paragraph. Choosing kernels via $\gamma_{k^*}(X, Z)$ using surrogate samples $Z \sim \mathbb{G}$, as represented by the green curve in Fig. 7.2, substantially boosts the test power compared to $\eta_{k*}(X, \tilde{Y})$ with sparse samples $\tilde{Y} \sim \mathbb{Q}$. This toy example not only suggesets that optimizing kernel with surrogate distribution $\mathbb{G}$ leads to better test power when samples

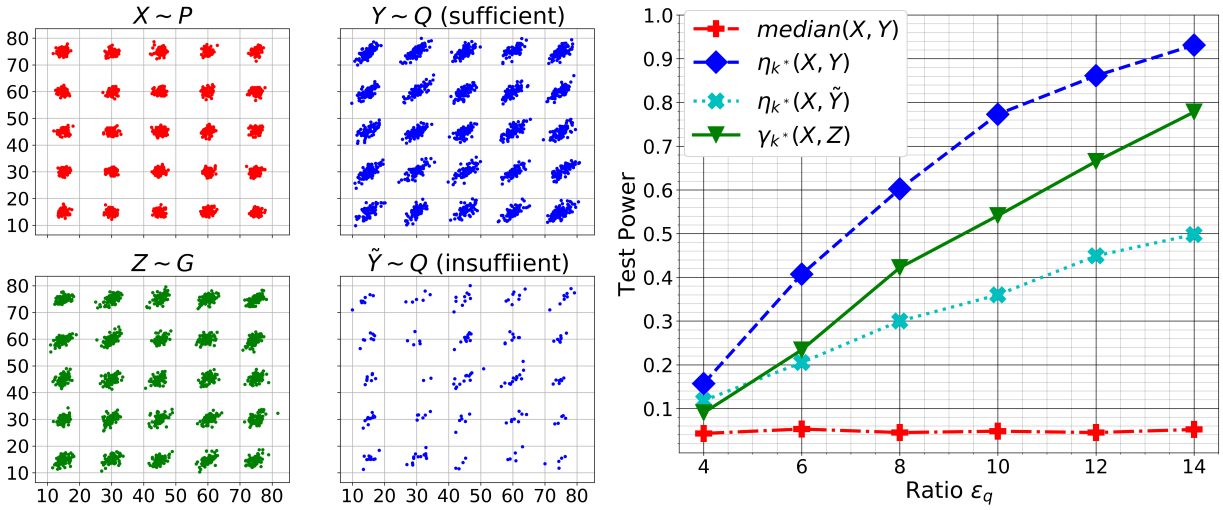


Figure 7.2: **Left**: 5×5 Gaussian grid, samples from $\mathbb{P}$, $\mathbb{Q}$ and $\mathbb{G}$. We discuss two cases of $\mathbb{Q}$, one of sufficient samples, the other of insufficient samples. **Right**: Test power of kernel selection versus ϵ_q. Choosing kernels by $\gamma_{k^*}(X, Z)$ using a surrogate distribution $\mathbb{G}$ is advantageous when we do not have sufficient samples from $\mathbb{Q}$, which is typically the case in time series CPD task.

from $\mathbb{Q}$ are insufficient, but also demonstrates that the effectiveness of our kernel selection objective holds without introducing any autoregressive/RNN modeling to control the Type-I error.

Test Threshold Approximation The test threshold c_α is a function of k and $\mathbb{P}$. Under $H_0 : \mathbb{P} = \mathbb{Q}$, $m\hat{M}_k(X, Y)$ converges asymptotically to a distribution that depends on the unknown data distribution $\mathbb{P}$ (Gretton et al., 2012a, Theorem 12); thus there is no analytical form of c_α, which makes optimizing Eq. (7.4) difficult. Commonly-used heuristics iteratively estimate c_α by the permutation test and fixing it during optimization (Sutherland et al., 2017), which is computationally demanding and non-differentiable.

For $X, X' \sim \mathbb{P}$, we know that c_α is a function of the empirical estimator $\hat{M}_k(X, X')$ that controls the Type I error. Bounding $\hat{M}_k(X, X')$ could be an approximation of bounding c_α. Therefore, we propose the following objective that maximizing a lower bound of test power

$$\arg\max_{k \in \mathcal{K}} M_k(\mathbb{P}, \mathbb{G}) - \lambda \hat{M}_k(X, X'), \tag{7.5}$$

where λ is a hyper-parameter to control the trade-off between Type-I and Type-II errors, as well as absorbing the constants m, v_l, v_u in variance approximation. Note that in experiment, the optimization of Eq. (7.5) is solved using the unbiased estimator of $M_k(\mathbb{P}, \mathbb{G})$ with empirical samples.

7.2.3 Surrogate Distributions using Generative Models

The remaining question is how to construct the surrogate distribution $\mathbb{G}$ without any sample from $\mathbb{Q}$. Injecting random noise to $\mathbb{P}$ is a simple way to construct $\mathbb{G}$. While straightforward, it may result in a sub-optimal $\mathbb{G}$ because of sensitivity to the level of injected random noise. As no prior knowledge of $\mathbb{Q}$, to ensure (7.3) hold for any possible $\mathbb{Q}$ (e.g. $\mathbb{Q} \neq \mathbb{P}$ but $\mathbb{Q} \approx \mathbb{P}$), intuitively, we have to make $\mathbb{G}$ as closed

to $\mathbb{P}$ as possible. We propose to learn an *auxiliary generative model* $\mathbb{G}_\theta$ parameterized by θ such that

$$\hat{M}_k(X, X') < \min_\theta M_k(\mathbb{P}, \mathbb{G}_\theta) < M_k(\mathbb{P}, \mathbb{Q}), \forall k \in \mathcal{K}.$$

To ensure the first inequality hold, we set early stopping criterion when solving $\mathbb{G}_\theta$ in practice. Also, if $\mathbb{P}$ is sophisticate, which is common in time series cases, limited capacity of parametrization of $\mathbb{G}_\theta$ with finite size model (e.g. neural networks) (Arora et al., 2017) and finite samples of $\mathbb{P}$ also hinder us to fully recover $\mathbb{P}$. Thus, we derive a min-max formulation to consider all possible $k \in \mathcal{K}$ when we learn $\mathbb{G}$,

$$\min_\theta \max_{k \in \mathcal{K}} \quad M_k(\mathbb{P}, \mathbb{G}_\theta) - \lambda \hat{M}_k(X, X'), \tag{7.6}$$

and solve the kernel for the hypothesis test in the mean time. In experiment, we use simple alternative (stochastic) gradient descent to solve each other.

7.3 KLCPD: Realization for Time Series Applications

In this section, we present a realization of the kernel learning framework for time series CPD.

Compositional Kernels To have expressive kernels for complex time series, we consider compositional kernels $\tilde{k} = k \circ f$ that combines RBF kernels k with injective functions f_ϕ:

$$K = \left\{ \tilde{k} \mid \tilde{k}(x, x') = \exp(-\|f_\phi(x) - f_\phi(x)'\|^2) \right\}. \tag{7.7}$$

The resulted kernel $\tilde{k}$ is still characteristic if f is an injective function and k is characteristic (Gretton et al., 2012a). This ensures the MMD endowed by $\tilde{k}$ is still a valid probabilistic distance. One example function class is $\{f_\phi | f_\phi(x) = \phi x, \phi > 0\}$, equivalent to the kernel bandwidth tuning. Inspired by the recent success of combining deep neural networks into kernels (Wilson et al., 2016; Al-Shedivat et al., 2017; Li et al., 2017), we parameterize the injective functions f_ϕ by recurrent neural networks (RNNs) to capture the temporal dynamics of time series.

For an injective function f, there exists a function F such that $F(f(x)) = x, \forall x \in \mathcal{X}$, which can be approximated by an auto-encoder via sequence-to-sequence architecture for time series. One practical realization of f would be a RNN encoder parametrized by ϕ while the function F is a RNN decoder parametrized by ψ trained to minimize the reconstruction loss. Thus, our final objective is

$$\min_\theta \max_\phi \quad M_{f_\phi}\big(\mathbb{P}, \mathbb{G}_\theta\big) - \lambda \cdot \hat{M}_{f_\phi}\big(X, X'\big) - \beta \cdot \mathbb{E}_{\nu \in \mathbb{P} \cup \mathbb{G}_\theta} \|\nu - F_\psi\big(f_\phi(\nu)\big)\|_2^2. \tag{7.8}$$

Practical Implementation In practice, we consider two consecutive windows in mini-batch to estimate $\hat{M}_{f_\phi}\big(X, X'\big)$ in an online fashion for the sake of efficiency. Specifically, the sample $X \sim \mathbb{P}$ is divided into the left window segment $X^{(l)} = \{x_{t-w}, \ldots, x_{t-1}\}$ and the right window segment $X^{(r)} = \{x_t, \ldots, x_{t+w-1}\}$ such that $X = \{X^{(l)}, X^{(r)}\}$. We now reveal implementation details of the auxiliary generative model and the deep kernel.

Generator g_θ Instead of modeling the explicit density $\mathbb{G}_\theta$, we model a generator g_θ where we can draw samples from. The goal of g_θ is to generate plausibly counterfeit but natural samples based on historical $X \sim \mathbb{P}$, which is similar to the conditional GANs (Mirza and Osindero, 2014). We use sequence-to-sequence (Seq2Seq) architectures (Sutskever et al., 2014) where g_{θ_e} encodes time series into hidden states, and g_{θ_d} decodes it with the distributional autoregressive process to approximate the surrogate sample Z:

$$H = g_{\theta_e}\big(X^{(l)}, \mathbf{0}\big), \quad \tilde{h} = h_{t-1} + \omega, \quad Z = g_{\theta_d}\big(X^{(r)}_{\gg 1}, \tilde{h}\big).$$

where $\omega \sim \mathbb{P}(W)$ is a d_h-dimensional random noise sampled from a base distribution $\mathbb{P}(W)$ (e.g., uniform, Gaussian). $H = [h_{t-w}, \ldots, h_{t-1}] \in \mathbb{R}^{d_h \times w}$ is a sequence of hidden states of the generator's encoder. $X^{(r)}_{\gg 1} = \{\mathbf{0}, x_t, x_{t+1}, \ldots, x_{t+w-2}\}$ denotes right shift one unit operator over $X^{(r)}$.

Deep Kernel Parametrization We aim to maximize a lower bound of test power via back-propagation on ϕ using the deep kernel form $\tilde{k} = k \circ f_\phi$. On the other hand, we can also view the deep kernel parametrization as an **embedding learning** on the injective function $f_\phi(x)$ that can be distinguished by MMD. Similar to the design of generator, the deep kernel is a Seq2Seq framework with one GRU layer of the follow form:

$$H_\nu = f_\phi(\nu), \; \hat{\nu} = F_\psi(H_\nu).$$

where $\nu \sim \mathbb{P} \cup \mathbb{G}_\theta$ are from either the time series data X or the generated sample $Z \sim g_\theta(\omega|X)$.

We present an realization of KL-CPD in Algorithm 9 with the weight-clipping technique. Note that other advanced regularization techniques such as gradient penalty (Gulrajani et al., 2017) and spectral normalization (Miyato et al., 2018) are also applicable. The stopping condition is based on a maximum number of epochs or the detecting power of kernel MMD $M_{f_\phi}\big(\mathbb{P}, \mathbb{G}_\theta\big) \leq \epsilon$. This ensure the surrogate $\mathbb{G}_\theta$ is not too close to $\mathbb{P}$, as motivated in Section 7.2.3.

Algorithm 9: KL-CPD, our proposed algorithm.

input : α the learning rate, c the clipping parameter,w the window size, n_c the number of iterations of deep kernels training per generator update.

while $M_{k\circ f_\phi}(\mathbb{P}, \mathbb{G}_\theta) > \epsilon$ **do**

  for $t = 1, \ldots, n_c$ **do**

    Sample a minibatch $X_t \sim \mathbb{P}$, denote $X_t = \{X_t^{(l)}, X_t^{(r)}\}$, and $\omega \sim \mathbb{P}(\Omega)$

    gradient$(\phi) \leftarrow \nabla_\phi M_{k\circ f_\phi}\big(\mathbb{P}, \mathbb{G}_\theta\big) - \lambda \hat{M}_{k\circ f_\phi}\big(X_t^{(l)}, X_t^{(r)}\big) - \beta \mathbb{E}_{\nu\sim\mathbb{P}\cup\mathbb{G}_\theta} \|\nu - F_\psi\big(f_\phi(\nu)\big)\|_2^2$

    $\phi \leftarrow \phi + \alpha \cdot \text{RMSProp}(\phi, \text{gradient}(\phi))$

    $\phi \leftarrow \text{clip}(\phi, -c, c)$

  Sample a minibatch $X_{t'} \sim \mathbb{P}$, denote $X_{t'} = \{X_{t'}^{(l)}, X_{t'}^{(r)}\}$, and $\omega \sim \mathbb{P}(\Omega)$

  gradient$(\theta) \leftarrow \nabla_\theta M_{k\circ f_\phi}\big(\mathbb{P}, \mathbb{G}_\theta\big)$

  $\theta \leftarrow \theta - \alpha \cdot \text{Adam}(\theta, \text{gradient}(\theta))$

7.3.1 Relations to MMD GAN

Lastly, we remark that although our proposed method KL-CPD has a similar objective function as appeared in MMD GAN (Li et al., 2017), we would like to point out the underlying interpretation and motivations are radically different, as summarized below.

The first difference is the interpretation of inner maximization problem $\max_k M_k(\mathbb{P}, \mathbb{G})$. MMD GANs (Li et al., 2017; Bińkowski et al., 2018) treat whole maximization problem $\max_k M_k(\mathbb{P}, \mathbb{G})$ as a new probabilistic distance, which can also be viewed as an extension of integral probability metric (IPM). The properties of the *distance* is also studied in Li et al. (2017); Arbel et al. (2018). A follow-up work (Arbel et al., 2018) by combining Mroueh et al. (2018) pushs $\max_k M_k(\mathbb{P}, \mathbb{G})$ further to be a scaled distance with gradient norm. However, the maximization problem (7.4) in this work defines the lower bound of the test power, which also takes the variance of the empirical estimate and the test threshold approximation into account, instead of the distance.

Regarding the goals, MMD GAN aims to learn a generative model that approximates the underlying data distribution $\mathbb{P}$ of interests. All the works (Dziugaite et al., 2015; Li et al., 2015b; Sutherland et al., 2017; Li et al., 2017; Bińkowski et al., 2018; Arbel et al., 2018; Li et al., 2019) use MMD or $\max_k M_k(\mathbb{P}, \mathbb{G})$ to define distance, then optimize $\mathbb{G}$ to be as closed to $\mathbb{P}$ as possible. However, that is not the goal of this work, where $\mathbb{G}$ is just an *auxiliary generative model* which needs to satisfies Eq. (7.3). Instead, we aim to find the most powerful k for conducting hypothesis test. In practice, we still optimize $\mathbb{G}$ toward $\mathbb{P}$ because we usually have no prior knowledge (sufficient samples) about $\mathbb{Q}$, and we want to ensure the lower bound still hold for many possible $\mathbb{Q}$ (e.g. $\mathbb{Q}$ can be also similar to $\mathbb{P}$). However, even with this reason, we still adopt early stopping to prevent the auxiliary $\mathbb{G}$ from being exactly the same as $\mathbb{P}$.

7.4 Experiments and Results

7.4.1 Evaluation on Real-world Data

We presents a comparative evaluation of the proposed KL-CPD and seven representative baselines on benchmark datasets from real-world applications of CPD, including the domains of biology, environmental science, human activity sensing, and network traffic loads. The data statistics are summarized in Table 7.1. We pre-process all dataset by normalizing each dimension in the range of $[0, 1]$. Detailed descriptions of all real-world datasets are itemized as follows.

- Bee-Dance[2] records the pixel locations in x and y dimensions and angle differences of bee movements. Ethologists are interested in the three-stages bee waggle dance and aim at identifying the change point from one stage to another, where different stages serve as the communication with other honey bees about the location of pollen and water.
- Fishkiller[3] records water level from a dam in Canada. When the dam not functions normally, the water level oscillates quickly in a particular pattern, causing trouble for the fish. The beginning and end of every water oscillation (fish kills) are treated as change points.

[2] http://www.cc.gatech.edu/~borg/ijcv_psslds/
[3] http://mldata.org/repository/data/viewslug/fish_killer/

- HASC[4] is a subset of the Human Activity Sensing Consortium (HASC) challenge 2011 dataset, which provides human activity information collected by portable three-axis accelerometers. The task of change point detection is to segment the time series data according to the 6 behaviors: stay, walk, jog, skip, stair up, and stair down.
- Yahoo[5] contains time series representing the metrics of various Yahoo services (e.g. CPU utilization, memory, network traffic, etc) with manually labeled anomalies. We select 15 out of 68 representative time series sequences after removing some sequences with duplicate patterns in anomalies.

Dataset	T	#sequences	domain	#labels
Bee-Dance	826.66	6	$\mathbb{R}^3$	19.5
Fishkiller	45175	1	$\mathbb{R}^+$	899
HASC	39397	1	$\mathbb{R}^3$	65
Yahoo	1432.13	15	$\mathbb{R}^+$	36.06

Table 7.1: Dataset. T is length of time series, #labels is average number of labeled change points.

Experiment Setting and Evaluation Metric Following Lai et al. (2018); Saatçi et al. (2010); Liu et al. (2013), the datasets are split into the training set (60%), validation set (20%) and test set (20%) in chronological order. Note that training is fully unsupervised for all methods while labels in the validation set are used for hyperparameters tuning. For quantitative evaluation, we consider receiver operating characteristic (ROC) curves of anomaly detection results, and measure the area-under-the-curve (AUC) as the evaluation metric. AUC is commonly used in CPD literature (Li et al., 2015a; Liu et al., 2013; Xu et al., 2017). For dataset with multiple sequences, mean of AUC is reported. We compare KL-CPD with real-time CPD methods (ARMA, ARGP, RNN,LSTNet) and retrospective CPD methods (ARGP-BOCPD, RDR-KCPD, Mstats-KCPD). Note that OPT-MMD is a deep kernel learning baseline which optimizes MMD by treating past samples as $\mathbb{P}$ and the current window as $\mathbb{Q}$ (insufficient samples).

Overall Comparison with representative CPD approaches In Table 7.2, the first four rows present the real-time CPD methods, followed by three retrospective-CPD models, and the last is our proposed method. KL-CPD shows significant improvement over the other methods on all the datasets, except being in a second place on the Yahoo dataset, with 2% lower AUC compared to the leading ARGP. This confirms the importance of data-driven kernel selection and effectiveness of our kernel learning framework.

Distribution matching approaches like RDR-KCPD and Mstats-KCPD are not as competitive as KL-CPD, and often inferior to real-time CPD methods. One explanation is both RDR-KCPD and Mstats-KCPD measure the distribution distance in the original data space with simple kernel selection using the median heuristic. The change-points may be hard to detect without the latent embedding learned by neural networks. KL-CPD, instead, leverages RNN to extract useful contexts and encodes time series in

[4] http://hasc.jp/hc2011

[5] https://webscope.sandbox.yahoo.com/catalog.php?datatype=s

Method	Bee-Dance	Fishkiller	HASC	Yahoo
ARMA (Box, 2013)	0.5368	0.8794	0.5863	0.8615
ARGP (Candela et al., 2003)	0.5833	0.8813	0.6448	**0.9318**
RNN (Cho et al., 2014)	0.5827	0.8872	0.6128	0.8508
LSTNet (Lai et al., 2018)	0.6168	0.9127	0.5077	0.8863
ARGP-BOCPD (Saatçi et al., 2010)	0.5089	0.8333	0.6421	0.9130
RDR-KCPD (Liu et al., 2013)	0.5197	0.4942	0.4217	0.6029
Mstats-KCPD (Li et al., 2015a)	0.5616	0.6392	0.5199	0.6961
OPT-MMD	0.5262	0.7517	0.6176	0.8193
KL-CPD (Proposed method)	**0.6767**	**0.9596**	**0.6490**	0.9146

Table 7.2: AUC on four real-world datasets. KL-CPD has the best AUC on three out of four datasets.

a discriminative embedding (latent space) on which kernel two-sample test is used to detection changing points. This also explains the inferior performance of Mstats-KCPD which uses kernel MMD with a fix RBF kernel. That is, using a fixed kernel to detect versatile types of change points is likely to fail.

Finally, the non-iid temporal structure in real-world applications may raise readers concern that the improvement coming from adopting RNN and controlling type-I error for model selection (kernel selection). Indeed, using RNN parameterized kernels (trained by minimizing reconstruction loss) buys us some gain compared to directly conduct kernel two-sample test on the original time series samples (Figure 7.3 cyan bar rises to blue bar). Nevertheless, we still have to do model selection to decide the parameters of RNN. In Table 7.2, we studied a kernel learning baseline, OPT-MMD, that optimizes an RNN parameterized kernel by controlling type-I error with insufficient samples of $\mathbb{Q}$. OPT-MMD is inferior to the KL-CPD that introduce the surrogate distribution with an auxiliary generator, which again verifies our claims. On the other hand, from Table 7.2, we can also observe KL-CPD is better than other RNN alternatives, such as LSTNet. Those performance gaps between KL-CPD, OPT-MMD (regularizing type-I only) and other RNN works indicate the proposed maximizing testing power framework via an auxiliary distribution serves as a good surrogate for kernel (model) selection.

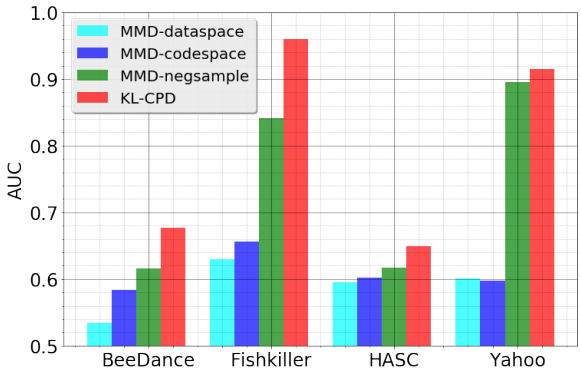


Figure 7.3: Ablation test of KL-CPD.

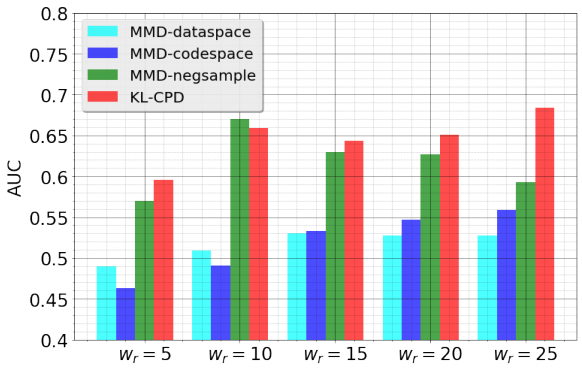


Figure 7.4: Varying w_r on Bee-Dance.

Ablation Test on Different Kernels and Auxiliary Distributions We further examine how kernels with different encoders f_ϕ and different auxiliary distributions affect KL-CPD. For MMD-dataspace, f_ϕ is an identity map, equivalent to kernel selection with median heuristic in data space. For MMD-codespace, $\{f_\phi, F_\psi\}$ is a Seq2Seq autoencoder minimizing reconstruction loss without optimizing test power. For MMD-negsample, the same objective as KL-CPD except for replacing the auxiliary generator with injecting Gaussian noise to $\mathbb{P}$.

The results are shown in Figure 7.3. We first notice the mild improvement of MMD-codespace over MMD-dataspace, showing that using MMD on the induced latent space is effective for discovering beneficial kernels for time series CPD. Next, we see MMD-negsample outperforms MMD-codespace, showing the advantages of injecting a random perturbation to the current interval to approximate $g_\theta(z|X^{(l)})$. This also justify the validity of the proposed lower bound approach by optimizing $M_k(\mathbb{P}, \mathbb{G})$, which is effective even if we adopt simple perturbed $\mathbb{P}$ as $\mathbb{G}$. Finally, KL-CPD models the $\mathbb{G}$ with an auxiliary generator g_θ to obtain conditional samples that are more complex and subtle than the perturbed samples in MMD-negsample, resulting in even better performance. In Figure 7.4, we also demonstrate how the tolerance of delay w_r influences the performance. Due to space limit, results other than Bee-Dance dataset are omitted, given they share similar trends. KL-CPD shows competitive AUC mostly, only slightly decreases when $w_r = 5$. MMD-dataspace and MMD-codespace, in contrast, AUC degradation is much severe under low tolerance of delay ($w_r = \{5, 10\}$).

7.4.2 In-depth Analysis on Simulated Data

To further explore the performance of KL-CPD with controlled experiments, we follow other time series CPD papers (Takeuchi and Yamanishi, 2006; Liu et al., 2013; Matteson and James, 2014) to create three simulated datasets each with a representative change-point characteristic: jumping mean, scaling variance, and alternating between two mixtures of Gaussian (Gaussian-Mixtures). See detailed setting below.

- Jumping-Mean: Consider the one-dimensional auto-regressive model to generate 5000 samples $y(t) = 0.6y(t-1) - 0.5y(t-2) + \epsilon_t$, where $y(1) = y(2) = 0$, $\epsilon_t \sim \mathcal{N}(\mu, 1.5)$ is a Gaussian noise with mean μ and standard deviation 1.5. A change point is inserted at every $100 + \tau$ time stamps by setting the noise mean μ at time t as

$$\mu_n = \begin{cases} 0 & n = 1, \\ \mu_{n-1} + \frac{n}{16} & n = 2, \ldots, 49, \end{cases}$$

  where $\tau \sim \mathcal{N}(0, 10)$ and n is a natural number such that $100(n-1) + 1 \leq t \leq 100n$.

- Scaling-Variance: Same auto-regressive generative model as Jumping-Mean, but a change point is inserted at every $100 + \tau$ time stamps by setting the noise standard deviation of ϵ_t at time t as

$$\sigma_n = \begin{cases} 1 & n = 1, 3, \ldots, 49, \\ \ln(e + \frac{n}{4}) & n = 2, 4, \ldots, 48, \end{cases}$$

  where $\tau \sim \mathcal{N}(0, 10)$ and n is a natural number such that $100(n-1) + 1 \leq t \leq 100n$.

- Gaussian-Mixtures: Time series data are sampled alternatively between two mixtures of Gaussian $0.5\mathcal{N}(-1, 0.5^2) + 0.5\mathcal{N}(1, 0.5^2)$ and $0.8\mathcal{N}(-1, 1.0^2) + 0.2\mathcal{N}(1, 0.1^2)$ for every 100 time stamps, which is defined as the change points.

Method	Jumping-Mean	Scaling-Variance	Gaussian-Mixtures
ARMA	0.7731 (0.06)	0.4801 (0.07)	0.5035 (0.08)
ARGP	0.4770 (0.03)	0.4910 (0.07)	0.5027 (0.08)
RNN	0.5053 (0.03)	0.5177 (0.08)	0.5053 (0.08)
LSTNet	0.7694 (0.09)	0.4906 (0.07)	0.4985 (0.07)
ARGP-BOCPD	0.7983 (0.06)	0.4767 (0.08)	0.5027 (0.08)
RDR-KCPD	0.6484 (0.11)	0.7574 (0.06)	0.6022 (0.11)
Mstats-KCPD	0.7309 (0.05)	0.7534 (0.04)	0.6026 (0.08)
KL-CPD	**0.9454** (0.02)	**0.8823** (0.03)	**0.6782** (0.05)

Table 7.3: AUC on three artificial datasets. Mean and standard deviation under 10 random seeds.

Main Results on Simulated data The results are summarized in Table 7.3. KL-CPD achieves the best in all cases. Interestingly, retrospective-CPD (ARGP-BOCPD, RDR-KCPD, Mstats-KCPD) have better results compared to real-time CPD (ARMA, ARGP, RNN,LSTNet), which is not the case in real-world datasets. This suggests low reconstruction error does not necessarily lead to good CPD accuracies. As for why Mstats-KCPD does not have comparable performance as KL-CPD, given that both of them use MMD as distribution distance? Notice that Mstats-KCPD assumes the reference time series (training data) follows the same distribution as the current interval. However, if the reference time series is highly non-stationary, it is more accurate to compute the distribution distance between the latest past window and the current window, which is the essence of KL-CPD.

MMD versus Dimensionality of Data We study how different encoders f_ϕ would affect the power of MMD versus the dimensionality of data. We generate an simulated time series dataset by sampling between two multivariate Gaussian $\mathcal{N}(0, \sigma_1^2 I_d)$ and $\mathcal{N}(0, \sigma_2^2 I_d)$ where the dimension $d = \{2, 4, 6, \ldots, 20\}$ and $\sigma_1 = 0.75, \sigma_2 = 1.25$.

Figure 7.5 plots the one-dimension data and AUC results. We see that all methods remain equally strong in low dimensions ($d \leq 10$), while MMD-dataspace decreases significantly as data dimensionality increases ($d \geq 12$). An explanation is non-parametric statistical models require the sample size to grow exponentially with the dimensionality of data, which limits the performance of MMD-dataspace because of the fixed sample size. On the other hand, MMD-codespace and KL-CPD are conducting kernel two-sample test on a learned low dimension codespace, which moderately alleviates this issue. Also, KL-CPD finds a better kernel (embedding) than MMD-codespace by optimizing the lower bound of the test power.

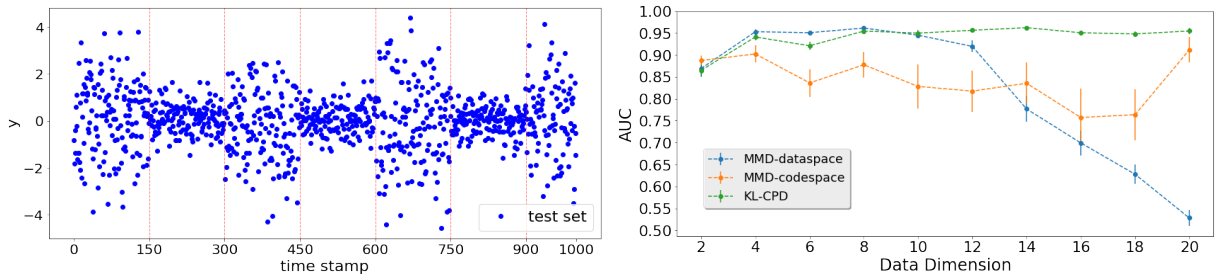


Figure 7.5: MMD with different encoder f_{ϕ_e} versus data dimension, under 10 random seeds.

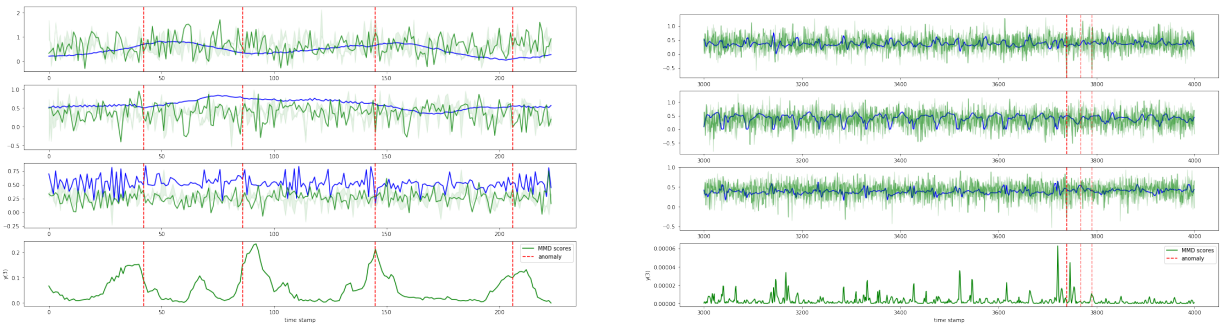


Figure 7.6: Conditionally generated samples by KL-CPD and system-predicted CPD scores on Bee-Dance (Left) and HASC (Right) datasets. In the first three subplots are ground truth signals (blue line), 10 conditional generated samples (green lines) and change points (red vertical line). The last subplot is MMD scores, which peaks around ground truth change points mostly.

7.5 Summary

In this chapter, we propose KL-CPD, a new kernel learning framework for two-sample test for change-point detection (CPD) problems. The key insight is to optimize a lower bound of the test power with an auxiliary generator, which resolves the issue of insufficient samples in CPD applications. The deep kernel parametrization of KL-CPD combines the latent space of RNNs with RBF kernels that effectively detect a variety of change-points from different real-world applications. Extensive evaluation of our new approach along with strong baseline methods on benchmark datasets shows the outstanding performance of the proposed method in retrospective CPD. With simulation analysis in addition we can see that the new method not only boosts the kernel power but also evades the performance degradation as data dimensionality increases.

Chapter 8

Graph-based Transfer Learning

8.1 Background

Transfer learning (TL) aims to address the label-sparse problem arising in many real-world applications as acquiring a large quantity of labeled data is extremely expensive and labor-intensive. TL methods address this problem by transferring the trained models from label-rich domain (source domain) to a relevant but label-sparse domain (target domain) according for the task of interest. Using topic classification of web blogs as an example (as in Pan et al. (2010)), obtaining a large set of labeled instances is often difficult especially when the web blogs are newly released. On the other hand, large collections of labeled news stories in relevant topics may be easily found on the internet. Thus if we can successfully transfer the classification models or the induced features from the news-story domain to the web blog domain, then the label-sparse problem in the target domain would be effectively addressed. Another motivating example is to transfer the text classification models from a label-rich language (e.g., English) to a label-sparse or label-sparser language (e.g., Italian or Turkish). Unlike English or a few internationally dominating languages, most of the other languages in the world have much less labeled documents in comparison. This means that TL would have a tremendous impact on the true success of text classification for all languages in the world if we can solve TL in all the cross-lingual settings. Notice an important difference between the two examples we have introduced, i.e., in the first example both source and target domain share the same feature space (the same vocabulary of English), while in the second example the two domains have different feature spaces (i.e., the vocabularies of two different languages). Nevertheless, TL across different feature spaces (*heterogeneous*) is usually a tougher problem than TL within the common feature space (*homogeneous*).

The literature of TL methods (Pan and Yang, 2009) reveals promising results in a variety of real-world applications, such as text classification (Pan et al., 2010; Duan et al., 2012), image classification (Zhu et al., 2011; Kulis et al., 2011), sentiment analysis (Glorot et al., 2011; Zhou et al., 2014), recommendation systems (Li et al., 2009), and more. Let us outline the major differences among existing approaches based on their basic assumptions in relating source and target domain, as well as on how labeled and unlabeled data in both domains are jointly leveraged during the TL process.

Transductive Transfer Learning (TTL) is one of the representative approach. TTL focuses on cross-domain kernel construction and the utilization of unlabeled data in *both* source and target domains during

the learning procedure. Adaptation Regularization based Transfer Learning (ARTL) (Long et al., 2013) is a representative work of TTL methods. It constructs a unified kernel and applies graph-based label propagation technique under certain regularized constraints to infer labels in target domain. This kernel-based approach is highly effective even given limited training data. However, ARTL or any existing TTL-based methods, to the best of our knowledge, is not applicable to the heterogeneous feature setting, which is the focus of this paper. The difficulty arises in cross-domain kernel construction. How could a kernel value between data from different domains be computed if they are not in the same feature space?

One common stream of approaches, *Feature Representation Transfer Learning* (FRTL), which can be used in heterogeneous settings addresses the problems by learning a common feature space, and then performing model transfer or parameter adaptation within that subspace. An important assumption adopted by most existing FRTL approaches is the availability of cross-domain *parallel data*. i.e., corresponding instances that have both source and target representations. There are various way to learn the common feature representation. For example, Argyriou et al. (2008) tried to induce a shared projection matrix for both source and target domain. Glorot et al. (2011) applied a deep learning technique, the stack denoised autoencoder (SDA), for a non-linear projection onto the shared latent space in cross-domain sentiment classification. Chandar et al. (2016) proposed a correlational neural network (CorrNet) approach that combines autoencoders (AE) and canonical correlation analysis (CCA) in the way that AE learns a generalized representation for each domain while CCA captures the joint representation of the two domains by maximizing the correlation in-between. Notice that above state-of-the-art neural network methods (CorrNet) usually require large amount of parallel data to achieve competitive results. Such large size of parallel data may not be realistic to obtain given fixed budgeted resources in real world applications. (e.g. human labeling for parallel sentences in low-resource languages)

8.2 KerTL: Kernel Transfer Learning on Graphs

Settings and Notations Let us first formally define the Transfer Learning (TL) problem of interest, and then show how to formulate TL as an optimization problem with graph regularization. For any single data domain $\mathcal{D} = \mathcal{X} \times \mathcal{Y}$, denote by $D = O \cup U$ the training set consisting of both labeled examples $O = \{\mathbf{x}_i, y_i\}$ and unlabeled examples $U = \{\mathbf{x}_i\}$ drawn from $\mathcal{X} \times \mathcal{Y}$ and $\mathcal{X}$ respectively. If the context is clear, we abuse the notation of $\mathbf{x}$ to denote a feature vector in D without distinguishing whether it comes from O or U.

We focus on TL involving two domains with *heterogeneous* features but a shared label space. Specifically, we are given a source domain $\mathcal{D}_s = \mathcal{X}_s \times \mathcal{Y}$ and a target domain $\mathcal{D}_t = \mathcal{X}_t \times \mathcal{Y}$ where $\mathcal{X}_s$ is allowed to differ from $\mathcal{X}_t$. We in addition assume the accessibility to a parallel set $PL = \{(\mathbf{x}_i^{(pls)}, \mathbf{x}_i^{(plt)})\}$ where $\mathbf{x}_i^{(pls)} \in D_s$, $\mathbf{x}_i^{(plt)} \in D_t$. Each feature pair in PL corresponds to "one" datum's representation in two domains. Given D_s, D_t and PL, our goal is to make predictions on the unlabeled target-domain data U_t with low expected error. Notice that all data points are already specified in $D_s \cup D_t$. The parallel set PL only suggests inter-domain relations, namely which data in D_s have counter-parts in D_t.

8.2.1 TL with Graph Laplacian

The aforementioned parallel-data-based TL problem can be formulated in a way that is compatible with graph-based SSL. Specifically, we view all (both labeled and unlabeled) data points in $D_s \cup D_t$ as nodes in a graph, whose edges encode inter-node similarities summarized in a $|D_s \cup D_t| \times |D_s \cup D_t|$ adjacency matrix W. Our task therefore becomes making predictions on the target-domain unlabeled nodes (U_t) in the graph. With limited supervision available, it is desirable to propagate from labeled nodes to unlabeled ones with respect to the manifold structure of the graph, on the assumption that nodes sharing high similarities should also share similar labels.

For brevity we assume a binary label space $\mathcal{Y} = \{-1, 1\}$. Denote by y the true label and by $f(\mathbf{x})$ the corresponding predicted value. Predicted values over all nodes in $D_s \cup D_t$ are further concatenated to form a long vector $\boldsymbol{f} = [\boldsymbol{f}_s, \boldsymbol{f}_t]^\top$ of length $|D_s \cup D_t|$. Our problem is then formalized as:

$$\min_{\boldsymbol{f}} \sum_{(\mathbf{x},y)\in O_s \cup O_t} \ell(f(\mathbf{x}), y) + \gamma \boldsymbol{f}^\top \mathcal{L} \boldsymbol{f}, \tag{8.1}$$

where $\mathcal{L}$ is the graph Laplacian characterizing smoothness of graph and γ is a positive scalar controlling the regularization strength. Laplacian $\mathcal{L}$ is defined as $\mathcal{L} = W\mathbf{1} - W$.

The term in (8.1) is the empirical loss between predicted labels and true labels on subsets O_s and O_t. While the last term indicates the normalization penalty with respect to the manifold structure of all data. Specifically, we can show that this Laplacian can be reformulated as $\boldsymbol{f}^\top \mathcal{L} \boldsymbol{f} = \frac{1}{2} \sum_{i,j} w_{ij}(f_i - f_j)^2$, which reveals the motivation of the penalty: nodes that have strong similarities (with large w_{ij}) should have close prediction scores (f_i and f_j).

8.2.2 Graph Constructions

The adjacency matrix W and its associated Laplacian $\mathcal{L}$ play a key role in our formulation. In the homogeneous setting, there are widely-accepted routines to compute W using either cosine or radial basis function (RBF) measurements. Nonetheless, under the heterogeneity assumption, all those methods would become inappropriate in evaluating similarities for the inter-domain part. This implies the need of completing (instead of directly computing) the inter-domain similarities through both the pre-computed intra-domain similarities and the information from the parallel data, which is one of the key contributions in our work. In the following, we start from homogeneous graph construction and then move on to the more generic framework of tackling heterogeneous scenarios.

Homogeneous Graph Construction Given a pair of data points $\mathbf{x}_i, \mathbf{x}_j$ in homogeneous feature space $\mathcal{X}$, one could compute the pair-wise similarity w_{ij} using different functions, among which two typical choices are cosine measurement and RBF measurement $w_{ij} = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2})$. The choice of similarity function is usually domain-specific. For example, with text data (term frequency), cosine measurement is empirically often a better choice in characterizing documents with similar (proportional) word counts.

Besides, one would usually consider "dropping out" some weights (i.e. truncating a subset of w_{ij}'s to 0) which is called "sparsification" in order to emphasize local information and to lower the computation cost. A common practice is to keep weights only of each node's k nearest neighbors (kNN). Compared to ϵ-graphs where one specifies a fixed threshold for truncating all edge weights, the kNN graph allows

"adaptive" neighborhood radius for both strongly and weakly connected points (Zhu et al., 2005), and often leads to better classification results.

Heterogeneous Graph Construction The construction for intra-domain similarities stays valid within the heterogeneity setting. However, the inter-domain scores cannot be directly computed (neither $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ (cosine) nor $\|\mathbf{x}_i - \mathbf{x}_j\|$ (RBF) can be calculated as $\mathbf{x}_i$ and $\mathbf{x}_j$ are in different feature spaces). To simplify our discussion, suppose the data are in a well-arranged order such that the adjacency matrix W is in the following form:

$$W = \begin{bmatrix} W_{s,s} & W_{s,t} \\ W_{t,s} & W_{t,t} \end{bmatrix}, \tag{8.2}$$

where $W_{s,s}$, $W_{t,t}$ represent the intra-domain parts, and $W_{s,t} = W_{t,s}^\top$ represent the inter-domain parts.

Suppose $\mathbf{x}_i$ and $\mathbf{x}_j$ are a pair of parallel data, which means they are two alternative views of one datum. Then, it is always reasonable to set $w_{ij} = 1$ since a datum and itself should always be similar to itself. However, these entries are the only observable cells in $W_{s,t}$ and $W_{t,s}$. All the remaining cells should be completed using information from intra-domain similarities and parallel data.

Such off-diagonal matrix completion problem has strong resemblance with the bipartite graph edge completion problem, where nodes in D_s and D_t form a bipartite graph and the goal is to complete the bipartite edges ($W_{s,t}$) in the middle (Liu and Yang, 2015). In the following, we use $\hat{W}_{s,t}$ to denote the completed matrix and $W_{s,t}$ for the original (observed) version.

Random Walk Completion Let us consider the task of completing the missing (p, q)-th entry in $W_{s,t}$. Although the value of $(w_{s,t})_{pq}$ is unknown, some other entry (r, q) in the same column might have been observed and hence $(w_{s,t})_{pq}$ should be close to $(w_{s,t})_{rq}$ if the two "must-links" (p, q) and (r, q) are similar. Such similarity is provided as $(w_{s,s})_{pr}$. This suggests completing $(w_{s,t})_{pq}$ by aggregating all elements in the q-th column of $W_{s,t}$ weighted by the p-th row in $W_{s,s}$. Namely $(\hat{w}_{s,t})_{pq} \leftarrow \sum_r (w_{s,s})_{pr}(w_{s,t})_{rq}$. The above can be expressed in the matrix form

$$\hat{W}_{s,t} \leftarrow W_{s,s} W_{s,t}. \tag{8.3}$$

When $W_{s,s}$ is normalized as a column-stochastic matrix, equation (8.3) amounts to one-step random walk for each column in $W_{s,t}$. Alternatively, completion can be carried out row-wisely

$$\hat{W}_{s,t} \leftarrow W_{s,t} W_{t,t}. \tag{8.4}$$

Combining (8.3) and (8.4) leads to one-step simultaneous random walk in both source and target domain:

$$\hat{W}_{s,t} \leftarrow W_{s,s} W_{s,t} W_{t,t}. \tag{8.5}$$

By further allowing varying number of random walk steps on both sides (k steps on both sides in total), and by aggregating the effect of all different steps, we obtain

$$\hat{W}_{s,t}^{(k)} = \sum_{i=0}^{k} \binom{k}{i} W_{s,s}^i W_{s,t} W_{t,t}^{k-i}. \tag{8.6}$$

Compared to one-step random walk completion, (8.6) takes into account multi-step transduction over the graph, which is particularly desirable in our case where missing entries in $W_{s,t}$ may not have observed entries as its direct neighbor.

8.2.3 Diffusion Kernel Completion

We propose the following diffusion kernel completion

$$\hat{W}_{s,t} = \exp(\alpha_s W_{s,s}) W_{s,t} \exp(\alpha_t W_{t,t}). \tag{8.7}$$

This is equivalent to the aggregation of infinite number of weighted Random Walk Completions. Specifically,

$$\hat{W}_{s,t} = \sum_{k=0}^{\infty} \hat{W}_{s,t}^{(k)}(\alpha_s, \alpha_t). \tag{8.8}$$

where $\hat{W}_{s,t}^{(k)}$ denotes the weighted Random Walk Completion:

$$\hat{W}_{s,t}^{(k)}(\alpha_s, \alpha_t) = \sum_{i=0}^{k} \binom{k}{i} \alpha_s^i W_{s,s}^i W_{s,t} \alpha_t^{k-i} W_{t,t}^{k-i}. \tag{8.9}$$

positive scalars α_s and α_t are corresponding to the weights for the source- and target- domain graphs, respectively. Due to space limit, we do not provide the proof details.

Low Rank Approximation of Diffusion Kernel As in many other matrix completion tasks, it can be useful to impose low-rank assumptions on $\hat{W}_{s,t}$. The compressed sensing theory (Candès and Recht, 2009) implies there is still hope to recover $\hat{W}_{s,t}$ even if our intra-domain matrices are non-informative (e.g. identity matrices). To some extent, the low-rank factorization process is a denoising procedure trying to recover the missing signals.

Thus, we first take the low-rank eigen-decomposition on both $\exp(\alpha_s W_{s,s})$ and $\exp(\alpha_t W_{t,t})$ such that

$$\exp(\alpha_s W_{s,s}) \approx Q_s \exp(\alpha_s \Lambda_s) Q_s^\top \tag{8.10}$$

$$\exp(\alpha_t W_{t,t}) \approx Q_t \exp(\alpha_t \Lambda_t) Q_t^\top, \tag{8.11}$$

where Λ_s, Λ_t are the k_s, k_t leading eigen-values for $W_{s,s}$, $W_{t,t}$ respectively, and Q_s, Q_t are the corresponding stacked eigen-vectors for $W_{s,s}$, $W_{t,t}$.

The diffusion kernel completion (8.7) is then modified as

$$\hat{W}_{s,t} = \left(Q_s \exp(\alpha_s \Lambda_s) Q_s^\top\right) W_{s,t} \left(Q_t \exp(\alpha_t \Lambda_t) Q_t^\top\right). \tag{8.12}$$

Optimization Algorithms The proposed graph construction method gives us a joint adjacency matrix W for all data points in both the source and target domains, along with its associated graph Laplacian. To recap, our task is to solve the optimization problem:

$$\min_{\boldsymbol{f}} \quad h(\boldsymbol{f}) \equiv \sum_{i \in \mathcal{O}_f \cup \mathcal{O}_\sqcup} \ell(f_i, y_i) + \gamma \boldsymbol{f}^\top \mathcal{L} \boldsymbol{f}, \tag{8.13}$$

It is not hard to verify that (8.13) is a convex optimization problem when $\ell(\cdot,\cdot)$ is convex. This enables us to adopt a wide range of optimization techniques. In particular, we compute the exact solution for the square loss and use Adagrad which is a widely-tested sub-gradient method (Duchi et al., 2011) for other losses (e.g. logistic and hinge loss).

Note that our computation could be fast when using the low-rank approximation. The computational bottleneck of our method during optimization lies in the multiplication of $\mathcal{L}$ and $\boldsymbol{f}$ when calculating the gradient of (8.13). Recall $\mathcal{L}$ is a function of W, the gradient computation can be carried out in linear time over $|\mathcal{D}_t|$ and $|\mathcal{D}_s|$ when the diagonal blocks $W_{s,s}$, $W_{t,t}$ take kNN forms, making their multiplication with $\boldsymbol{f}$ cost as much as $O(k|\mathcal{D}_s|)$ and $O(k|\mathcal{D}_s|)$, respectively. Similarly, the time complexity of doing matrix-vector multiplication with the off-diagonal block $\hat{W}_{s,t}$ will be $O(d\max(|\mathcal{D}_s|,|\mathcal{D}_t|))$ where d is the low-rank dimension.

8.3 Experiments and Results

8.3.1 Benchmark Datasets and Comparing Methods

Amazon Product Reviews (APR) The APR dataset (Prettenhofer and Stein, 2010) was designed for evaluations of sentimental classification with transfer learning in cross-language and cross-domain settings. It consists of Amazon product reviews on books (B), DVDs (D) and music (M), and written in English (EN), German (GE), French (FR) and Japanese (JP). For each language on each product type (B, D or M), there are 2000 labeled reviews for training and 2000 labeled reviews for testing, respectively. Parallel data are also provided for each language pair, which we will describe with an example *task* in the next. Following the settings in Zhou et al. (2014), we treat English as the source language, and the remaining three languages (German, French and Japanese) as the target languages. For each language pair (EN-GE, EN-FR or EN-JP), we have 6 cross-product-type pairs, constituting overall 18 cross-language and cross-product-type combinations (e.g. EN-B-FR-D as shown in the first column of Table 8.2). Specifically, EN-B-FR-D represents TL task with English reviews on Books as source domain, and French reviews on DVD products as target domain. The parallel dataset for the EN-B-FR-D task is obtained by running Google translation over the 2,000 French book reviews in the training set, and by treating the system-produced translations as the English behalf of the parallel data.

MNIST Handwritten Images The MNIST dataset consists of $70,000$ images in total, with digits from 0 to 9 as the class labels (one per image). We follow the setting in Chandar et al. (2016), to treat *left* half of each image (28×28 pixels) as a source-domain instance, while *right* half of the image as a target-domain instance. Raw pixel values are used as features. We randomly sampled $3,000$ images from the full set as the unlabeled parallel set, $2,000$ images as the source-domain training set, $1,024$ images as the target-domain training set, and another of $2,000$ images as the test set (only the target-domain portion is used). We call the classification with respect to each target label (a digit from 0 to 9) as a task in image recognition. Although the source and target domains have same feature dimensions, the features are indeed heterogeneous (direct cosine/RBF computation of two half images would not indicate label similarity). The idea would be more clear if we cut images in a 1/3 and 2/3 fashion, but for the ease of comparison with existing methods, we keep the same cutting scheme (Chandar et al., 2016).

Constructing unbalance training sets and size-varying parallel sets To simulate the label-sparsity of target domain in TL problem, we construct unbalanced training set on the APR and MNIST data sets. Recall in TL each training set has the source-domain part and the target-domain part, respectively. For each task in APR, we use the full set of 2000 source domain labeled instances, and a randomly sampled subset of m target domain labeled instances ($m = 2, 4, 8, 16, 32$) from the full set as the final training data. The remaining target-domain labeled instances ($2000 - m$) are used for validation (hyper-parameter tuning). For MNIST, the source domain training pool has the full size of $2,000$ instances. Another m instances (for $m = 2, 4, 8, 16, 32$) randomly sampled from the target-domain training set are used to complete the full training set. For the parallel data in each task, we randomly sampled from the available pool with the sample sizes of $l = 64, 128, 256, 512, 1024, 2000$ for APR ($l = 64, 128, 256, 512, 1024, 2048, 3000$ for MNIST). The size-reduced samples allow us to evaluate transfer learning under the label-unbalanced and parallel-data-sparse conditions. For each value of m and l, we repeated the random sampling 10 times, and averaged the performance of the target-domain classifiers over the randomly sampled training sets and parallel data for each task in the evaluation. Table 8.1 summarizes the statistics of the datasets.

Comparing Methods We include six methods as baselines for comparison. Two of them are representative methods (SVM and SSL) in supervised classification where only the target-domain labeled data are used for training. We also include two TL baselines (HFA and MMDT), which use labeled data in both source domain and target domain for training, but cannot leverage parallel data. The remaining two methods (HHTL and CorrNet) are the state-of-the-art TL methods which use both the labeled data in both domains as well as parallel data in addition. Some details of these baseline methods are described below.

- Support Vector Machine (SVM): We used the L2-SVM from LIBLINEAR (Fan et al., 2008).
- Semi-Supervised Learning (SSL) (Zhu et al., 2005): We implemented the graph-based label propagation method for Semi-Supervised Learning framework.
- Heterogeneous Feature Augmentation (HFA) (Li et al., 2013): This method embeds heterogeneous domain data into shared high-dimensional space, and deploys a Multiple Kernel Learning solver (Kloft et al., 2011). We used the code from the website [1].
- Max Margin Domain Transform (MMDT) (Hoffman et al., 2013): This method uses an asymmetric transformation matrix to map features across domains, which is optimized with respect to all the target categories. We used the code from the website [2].
- Hybrid Heterogeneous Transfer Learning (HHTL) (Zhou et al., 2014): This method uses a parallel corpus to learn a shared embedding space for source and target domains. Classifiers then can be trained on the labeled data in both domains. We used the code provided by the authors.
- Correlational neural network (CorrNet) (Chandar et al., 2016): This method uses autoencoders to simultaneously minimize classification errors in both domains, and to capture cross-domain correlations based on a parallel dataset. Similar to HHTL, classifiers are trained after the data are mapped onto the shared latent space. We used the code from the website [3].

[1] https://github.com/transmatrix-github/HFA-release
[2] https://github.com/jhoffman/MaxMarginDomainTransforms
[3] https://github.com/apsarath/CorrNet

Experiment Settings Our experimental results involve two random factors. The first comes from the random sampling of the target domain labeled training sets, and the second comes from the random sampling of the parallel datasets. All the experiments with random samples are repeated 10 times with different random seeds. Mean and standard deviation of the Area under Curve (AUC) of ROC are reported for evaluation and comparison.

In HHTL and CorrNet, after learning the projected matrices, we trained linear SVMs (Fan et al., 2008) on the projected training data. For all the methods using SVM classifiers (in SVM, HFA, MMDT, HHTL and CorrNet), we set the regularization parameter $C = 1$. For hyper-parameter tuning, we set the default hyper-parameters of HFA and MMDT the same as in their papers. We adopted the hyper-parameter of HHTL on the APR data, with a grid search of the optimal regularization coefficient among $\lambda = 0.001, 0.01, 1, 10$, and 100, and the corruption probability among $p = 0.5, 0.6, 0.7, 0.8$, and 0.9 on the MNIST dataset. Similarly, for CorrNet on the MNIST dataset we used a grid search for the number of hidden units as $20, 50, 100$, and 200, and $\lambda = 0.2, 2$, and 20 on the APR dataset.

For KerTL, we used the cosine similarity and RBF kernel on the APR and MNIST datasets, respectively. We keep the top 128 eigenvectors in the eigen-decomposition part for efficient computation, and set the regularization coefficient γ to be 2^{-10}.

8.3.2 Quantitative Evaluation

TL methods vs. non-TL methods In the first set of experiments we fixed the training-set size in the target domain as $m = 2$, and the parallel-set size as $l = 1024$. Figure 8.1 shows the averaged AUC scores of those methods. All the TL methods which leverage parallel data (HHTL, CorrNet and KerTL) significantly outperformed the methods that cannot take advantage of parallel data (SVM, SSL, HFA and MMDT). Among the TL methods, our KerTL outperforms all the other methods on both the APR and MNIST data sets. Tables 8.2 and 8.3 show the task-specific performance scores on the two data sets, respectively. Again, the performance of KerTL dominates across most of those tasks. On the MNIST dataset in particular, KerTL improved the result of CorrNet (which is the strongest baseline) from 93.2% to 96.2% in AUC, which is equivalent to reducing the error rate from 6.8% to 3.8%, i.e., a 44.1% reduction in error. Such an improvement is indeed significant.

Data sets	APR	MNIST
Source domain training set	2000	2000
Target domain training set	2000	1024
Target domain test set	2000	2000
Parallel data size	2000	3000

Table 8.1: Data Statistics

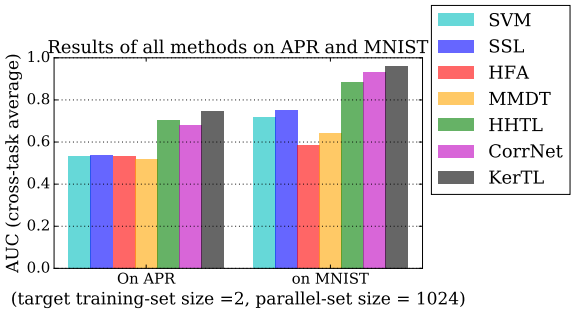


Figure 8.1: Comparison on APR and MNIST.

Tasks	SVM	SSL	HFA	MMDT	HHTL	CorrNet	KerTL
EN-B-GE-D	0.564	0.558	0.550	0.563	0.707	0.604	**0.715**
EN-B-GE-M	0.500	0.542	0.536	0.528	0.711	0.659	**0.730**
EN-B-FR-D	0.525	0.513	0.522	0.513	0.747	0.729	**0.748**
EN-B-FR-M	0.541	0.540	0.544	0.542	0.687	0.717	**0.738**
EN-B-JP-D	0.528	0.527	0.541	0.524	0.643	0.692	**0.713**
EN-B-JP-M	0.534	0.537	0.541	0.505	0.611	0.665	**0.724**
EN-D-GE-B	0.502	0.509	0.499	0.482	0.772	0.692	**0.796**
EN-D-GE-M	0.500	0.542	0.517	0.531	0.737	0.672	**0.755**
EN-D-FR-B	0.548	0.549	0.547	0.514	0.743	0.739	**0.785**
EN-D-FR-M	0.541	0.540	0.537	0.543	0.724	0.696	**0.741**
EN-D-JP-B	0.549	0.561	0.558	0.516	0.694	**0.719**	0.717
EN-D-JP-M	0.534	0.537	0.536	0.506	0.683	**0.747**	0.713
EN-M-GE-B	0.502	0.509	0.520	0.476	0.704	0.668	**0.786**
EN-M-GE-D	0.564	0.558	0.519	0.561	0.728	0.631	**0.740**
EN-M-FR-B	0.548	0.549	0.542	0.515	0.745	0.672	**0.789**
EN-M-FR-D	0.525	0.513	0.508	0.511	0.755	0.670	**0.764**
EN-M-JP-B	0.549	0.561	0.526	0.529	0.622	0.675	**0.739**
EN-M-JP-D	0.528	0.527	0.551	0.487	0.655	0.707	**0.708**
Average	0.532	0.537	0.533	0.519	0.704	0.687	**0.745**
± Std	±0.021	±0.018	±0.016	±0.023	±0.047	±0.037	±**0.029**

Table 8.2: Overall results on APR dataset with target domain training-set size of 2 and parallel set size of 1024. Bold-faced numbers indicate the best result on each row.

Tasks	SVM	SSL	HFA	MMDT	HHTL	CorrNet	KerTL
Digit 0	0.950	0.965	0.891	0.855	0.971	0.987	**0.989**
Digit 1	0.906	0.915	0.500	0.838	0.989	0.994	**0.996**
Digit 2	0.699	0.739	0.539	0.567	0.867	0.931	**0.962**
Digit 3	0.628	0.785	0.637	0.664	0.861	0.892	**0.939**
Digit 4	0.672	0.613	0.500	0.477	0.867	0.937	**0.958**
Digit 5	0.598	0.607	0.500	0.543	0.774	0.877	**0.959**
Digit 6	0.848	0.877	0.677	0.536	0.937	0.962	**0.985**
Digit 7	0.714	0.736	0.441	0.686	0.919	0.956	**0.968**
Digit 8	0.494	0.592	0.720	0.651	0.823	0.890	**0.936**
Digit 9	0.674	0.690	0.430	0.623	0.846	0.918	**0.929**
Average	0.718	0.752	0.584	0.644	0.885	0.934	**0.962**
± Std	±0.143	±0.133	±0.138	±0.118	±0.067	±0.041	±**0.023**

Table 8.3: Overall results on MNIST dataset with target domain training-set size of 2 and parallel set size of 1024. Bold-faced numbers indicate the best result on each row.

TL methods with varying-sized parallel data The second set of experiments compares the performance of TL methods (HHTL, CorrNet and KerTL) with varying sized parallel data, while the training-set size is fixed as $m = 2$ in the target domain. As shown in Figure 8.2, KerTL outperforms HHTL and CorrNet in most regions of the parallel-set sizes, on both the ARP and NMIST data sets. We also observed that the performance of HHTL was very sensitive to the settings of its hyper-parameters. When we fixed those parameter values and varied the sizes of the parallel data, HHTL's performance was either unstable (on MNIST) or decreasing (on APR) as the parallel data size increased.

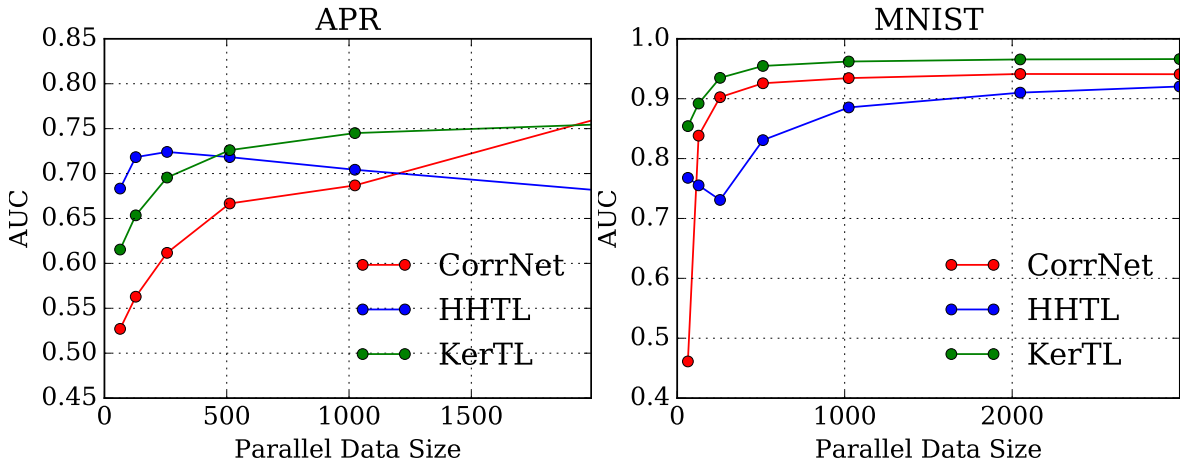


Figure 8.2: CorrNet, HHTL and KerTL on the APR dataset (left) and the MNIST dataset (right) with a varying quantity of parallel data.

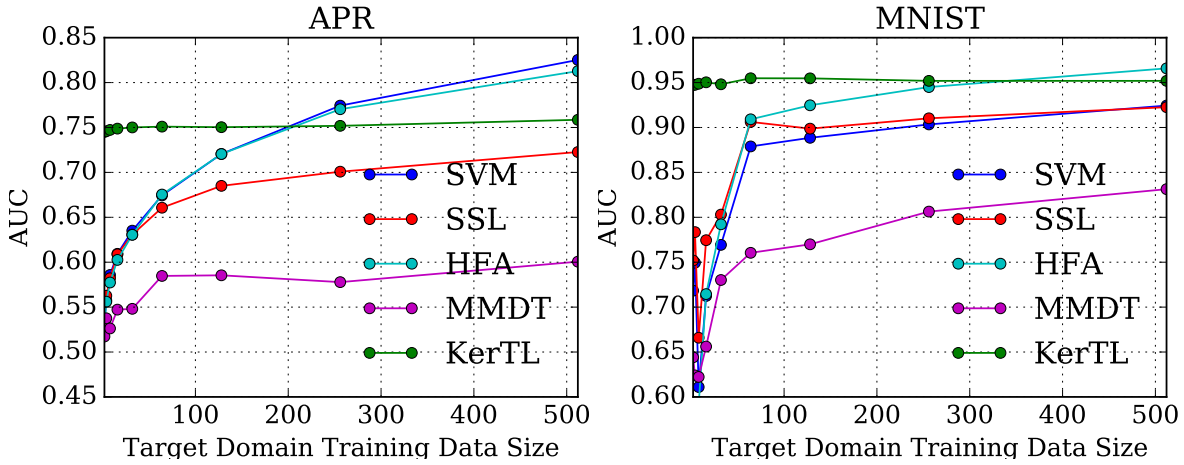


Figure 8.3: SVM, SSL and KerTL on the APR dataset (left) and the MNIST dataset (right) with a varying quantity of labeled data in the target domain.

Influence of label sparsity in the target domain The third set of experiments compares KerTL with SVM, SSL, HFA and MMDT under the condition that the labeled training instances are extremely sparse in the target domain, specifically with $m = 2, 4, 8, 16, 32$. Size of parallel datasets are $l = 1024$ in those experiments. Figure 8.3 shows results on APR and MNIST datasets. On both data sets, the curves of SVM, SSL and HFA increase rapidly when the training-set sizes are below 200. Without leveraging parallel data, MMDT does not perform well on both data set as target domain training data increase. We suspect the reason is that when source and target domain is very different (APR and MNIST in our setting), linear transform of the mapping with max margin criterion is not possible to find good representation without the help of parallel data. On the other hand, HFA is only comparable to KerTL when target domain training data is large enough since it does not utilize parallel data. KerTL has a nearly flat curve, substantially outperforming the others in the label-sparse regions. This implies KerTL could successfully transferred source-domain training data especially when source and target domain ($m = 2 \sim 32$) training data are very imbalanced.

But why the performance curve of KerTL is below that of SVM when the training-set size in the target domain is beyond 200 on the APR data? We believe that it is caused by the *imperfect* parallel data we used in KerTL. Recall that in our previous example of the EN-B-FR-D task, the parallel data are the paired English/French book reviews, assuming that the ideal parallel set of (manually aligned) English book reviews and French DVD reviews are not available. In other words, the parallel data provided in APR has a domain mismatch with respect to the reviews on different product types, which is most helpful when the label-sparse issue is severe.

In contrast, the parallel data sets in MNIST do not have a domain mismatch issue, as each pair in the parallel set consists of the left-half (as the source-domain instance) and right-half (as the target-domain instance) in the same image. We argue that the APR way of constructing parallel data is more realistic than that in MNIST, because we usually cannot get each image instance halfly labeled and halfly unlabeled in real-word applications of image classification.

8.4 Summary

In this chapter, we proposed a novel framework for graph-based transfer learning via cross-domain kernel induction. Our approach uses a parallel corpus to calibrate domain-specific graph Laplacians into a unified kernel, and to optimize semi-supervised label propagation based on the labeled and unlabeled data in both domains. Our extensive experiments show that all the TL methods in our evaluation significantly outperformed non-TL ones (SVM and SSL), and that the proposed method outperforms other state-of-the-art TL methods (HFA, MMDT, HHTL and CorrNet) when the target-domain labeled data are extremely sparse and the quantity of available parallel data is also limited. Those results indicates cross-language and cross-domain kernel induction is a promising direction to pursue in transfer learning.

Part V

Conclusions and Future Work

Chapter 9

Concluding Remarks

9.1 Main Contributions

In this thesis, we advance kernel-based algorithms in four complementary directions: kernelized data-to-data discrepancy for learning implicit generative models (IGMs); kernelized data-to-model discrepancy to sample from explicit generative models (EGMs); random features for large-scale kernel contextual bandit problems; kernel learning for structured data such as time-series and cross-domain graphs. With problem of interests in mind, we highlight main contributions below.

Implicit Generative Models (Part I): Learning IGMs via a fixed kernel maximum mean discrepancy (MMD) has limited success on high-dimensional computer vision datasets. We address the kernel selection challenge by optimizing the kernel MMD via deep compositional kernels, where neural-parametrized encoders (e.g., convolutions) induce semantic low-dimensional manifolds for base kernels. The proposed MMD GAN enhances the tractability of kernel-based IGMs to higher-dimensional datasets.

Explicit Generative Models (Part II): Drawing samples from EGMs with kernelized variational inference methods, namely Stein Variational Gradient Descent (SVGD), fails to recover mixture weights of multi-modal distributions, and degrades significantly as data-dimension increases. We propose noise-conditional kernels SVGD (NCK-SVGD) that works in tandem with the noise-conditional score estimator for drawing samples from high-dimensional multi-modal distributions. With an entropy regularizer, NCK-SVGD enjoys flexible control between the sample quality and diversity, measured by precision and recall.

Large-scale Kernel Approximation (Part III): We improve the tractability of large-scale kernel approximation (when number of instances is large) using non-uniformly weighted random features. The advantage of non-uniformly weighting scheme is justified by the risk minimization perspective using bias-variance trade-off. We further improve the computational efficiency of kernel contextual bandit problems by using variants of random features.

Kernels for Structured Data (Part IV): We extend the kernel learning framework to two applications with structured information, namely time series and cross-domain graphs. To address the sample-sparsity issue in CPD problem, we optimize kernel two-sample test via an auxiliary IGMs generating pseudo negative samples from anomaly distributions. To construct the cross-domain graph Laplacian, we leverage diffusion kernels (i.e., infinite random walks from the source graph and the target graph) to induce an unified graph, so that cross-domain knowledge can be transferred via label propagation.

9.2 Discussions

Several interesting research questions are worth discussing for deeper understanding of kernel-based methods in generative modeling:

- **Does IGMs suffer from curse of dimensionality?** Yes. Studies on kernel density estimators (KDEs) should that *curse of dimensionality* is indeed an issue: the minimax convergence rate $O(n^{-4/(4+d)})$ is very slow when d is large, and this optimal rate can not be improved (Stone, 1982; Tsybakov, 2008) unless we assume extra smoothness. Especially, the minimax convergence rate of IGMs is an active research topic, as IGMs are trained with adversarial losses instead of the conventional L_2 loss. Recently, Singh et al. (2018); Uppal et al. (2019) study the minimax convergence rates of nonparametric density estimators under adversarial losses, including Wasserstein distance and Maximum Mean Discrepancy. They show that learning IGMs is typically not easier than explicit density estimation, and the curse of dimensionality would also limit the performance of GAN-based IGMs.

- **Does the min-max optimization of GANs always converge?** Without careful regularization of the critic functions, the training dynamic of GANs may not always converge (Nagarajan and Kolter, 2017; Mescheder et al., 2018). It is possible to analyze the training dynamics of min-max optimization problems, under some critical assumptions. For example, by simple bilinear forms of critic functions and gradient penalties to ensure smoothness, Mescheder et al. (2018) present local convergence of training GANs. More recently, Nouiehed et al. (2019); Lin et al. (2020) establish some local convergence rates under specific noncovex-concave settings of the min-max problem. However, for general nonconvex-nonconcave settings of min-max problems the convergence property is still an open question.

- **Can we we tell if a generative model is only memorizing the training data?** We have been using the commonly-received evaluation metrics in this thesis, including Inception Score (Salimans et al., 2016), Fréchet Inception Distance (FID) (Heusel et al., 2017) and precision/recall curves (Simon et al., 2019; Kynkäänniemi et al., 2019). However, we acknowledge that each metric has its own limitation(s). For example, a *memory GAN* which stores all training samples without any generalization (i.e., no learning) would receive a perfect FID score (Lucic et al., 2018). Developing better evaluation metrics is certainly a meaningful direction for future work. Using simulated data with ground true densities would also allow us to detect the cases when the system merely memorizes the training data without any generalization.

- **Why consider kernels in additional to neural networks?** Kernel methods such as Gaussian Processes (GPs) have many nice properties such as provable generalization, explicit modeling of uncertainty, and more. Kernel methods are closely related to deep neural networks. For example, a convolutional neural network with a certain prior over the model weights is equivalent to a GP in the limit of infinitely many filters (Garriga-Alonso et al., 2019). It is possible to get best of both worlds by combining kernels with deep neural networks. Shankar et al. (2020) proposed deep compositional kernels that perform on par with ResNet-32 on the CIFAR-10 dataset, although its performance still has a room to be improved.

9.3 Future Work

I consider the following as promising directions for future research.

- **Explicit Generative Models** Deep kernel exponential families belong also define explicit densities and received great attention in recent years (Strathmann et al., 2015; Arbel and Gretton, 2018; Dai et al., 2019; Wenliang et al., 2019; Arbel et al., 2020). However, estimating the model with score matching suffers the tractability issue from high-dimensional and large-data applications with cubic complexity (Sutherland et al., 2018). Instead, kernel density estimator (KDE) offers a non-parametric estimation of the score function, which can be used in conjunction with the annealed Langevin dynamic for score-based EGMs (Song and Ermon, 2019, 2020). An interesting follow-up is to improve the KDE with deep neural parametrized kernels and study it generalization ability by examining nearest neighbor results, image inpainting, and interpolation.

- **Kernel on Graphs** Graph Kernels (GKs) and Graph neural networks (GNNs) are two mainstream approaches for learning on graph-structured data such as social network and biological networks. Popular choice of GKs include Weisfeiler-Lehman kernels (Shervashidze et al., 2011), graphlet kernels (Shervashidze et al., 2009), random walk kernels (Vishwanathan et al., 2010; Gärtner et al., 2003). However, graph kernels can be computational expensive, and using hand-crafted features in kernels may limited the downstream performance compared to state-of-the-art GNNs. To solve the scalability challenge, it is possible to approximate the graph kernel via random Fourier features (Wu et al., 2019b,a). The open research question is how to leverage deep kernel learning framework and recent advance in NTK (Du et al., 2019) into graph kernels to improve its performance.

- **Kernel on Natural Language** Word mover distance (WMD) (Kusner et al., 2015) measures the distance between two documents and show decent performance of text classification at the cost of high computational cost. To solve the computational issue, word mover embedding (Wu et al., 2018) approximate the WMD with an inner product of linear map using ideas similar to random Fourier features. However, Wu et al. (2018) only consider uniformly-sampled documents as the sampling bases from kernel spectral distributions. Potential improvements can be made by learning the kernel spectral distributions via the IKL framework proposed in Chapter 3.

- **Large-scale Softmax Approximation** There has been a surge of recent interest in developing sampling technique for the cross-entropy Softmax distribution, which is the de-facto loss function in modern classification tasks. Quadratic kernel softmax (Blanc and Rendle, 2018) uses a kernel-based sampling method and quadratic approximation of the softmax function to draw each sample in sublinear time. Following this idea, Random Fourier Softmax (Rawat et al., 2019) method utilizes Random Fourier Features to enable more efficient and accurate sampling from an approximate Softmax distribution. An interesting research idea is to incorporate recent advancing development in random Fourier features such as orthogonality (Yu et al., 2016), non-uniform weights (Chang et al., 2017a; Sinha and Duchi, 2016), quadrature-based features (Munkhoeva et al., 2018) into the Random Fourier Softmax problem, and examine if it leads to more efficient approximation with less sampling bases.

Bibliography

Alekh Agarwal, Miroslav Dudík, Satyen Kale, John Langford, and Robert Schapire. Contextual bandit learning with predictable rewards. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 19–26, 2012. 76

Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning (ICML)*, pages 1638–1646, 2014. 75, 83

Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning (ICML)*, pages 127–135, 2013. 75

Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P Xing. Learning scalable deep kernels with recurrent structure. *The Journal of Machine Learning Research*, 18(1): 2850–2886, 2017. 38, 99

Maximilian Alber, Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Fei Sha. An empirical study on the properties of random bases for kernel methods. In *Advances in Neural Information Processing Systems*, pages 2763–2774, 2017. 35

Michael Arbel and Arthur Gretton. Kernel conditional exponential family. In *International Conference on Artificial Intelligence and Statistics*, pages 1337–1346, 2018. 123

Michael Arbel, Dougal Sutherland, Mikołaj Bińkowski, and Arthur Gretton. On gradient regularizers for MMD GANs. In *Advances in neural information processing systems*, pages 6700–6710, 2018. 23, 39, 101

Michael Arbel, Liang Zhou, and Arthur Gretton. KALE: When energy-based learning meets adversarial training. *arXiv preprint arXiv:2003.05033*, 2020. 123

Andreas Argyriou, Massimiliano Pontil, Yiming Ying, and Charles A Micchelli. A spectral regularization framework for multi-task structure learning. In *Advances in neural information processing systems*, pages 25–32, 2008. 108

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 28

Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning (ICML)*, 2017. 4, 11, 13, 14, 16, 17, 18, 28

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *International Conference on Machine Learning (ICML)*, 2017. 16, 99

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002. 75, 77

Haim Avron, Vikas Sindhwani, and David Woodruff. Sketching structured matrices for faster nonlinear regression. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2994–3002, 2013. 70

Haim Avron, Vikas Sindhwani, Jiyan Yang, and Michael W Mahoney. Quasi-monte carlo feature maps for shift-invariant kernels. *The Journal of Machine Learning Research*, 17(1):4096–4133, 2016. 65, 67, 72

Haim Avron, Kenneth L Clarkson, and David P Woodruff. Faster kernel ridge regression using sketching and preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1116–1138, 2017a. 70

Haim Avron, Michael Kapralov, Cameron Musco, Christopher Musco, Ameya Velingker, and Amir Zandieh. Random fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. In *International Conference on Machine Learning (ICML)*, 2017b. 76, 78, 79, 88

Francis Bach. On the equivalence between kernel quadrature rules and random feature expansions. *The Journal of Machine Learning Research*, 18(1):714–751, 2017. 5

Francis R Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in neural information processing systems*, pages 105–112, 2009. 25

Francis R Bach and Michael I Jordan. Kernel independent component analysis. *Journal of machine learning research*, 3(Jul):1–48, 2002. 1

Francis R Bach, Gert RG Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *International Conference on Machine Learning (ICML)*, 2004. 25

Michèle Basseville, Igor V Nikiforov, et al. *Detection of abrupt changes: theory and application*. Prentice Hall Englewood Cliffs, 1993. 95

Sabyasachi Basu and Martin Meckesheimer. Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems*, 2007. 95

Eduard Gabriel Băzăvan, Fuxin Li, and Cristian Sminchisescu. Fourier kernel learning. In *European Conference on Computer Vision*, pages 459–473. Springer, 2012. 25, 32, 35, 38

Marc G Bellemare, Ivo Danihelka, Will Dabney, Shakir Mohamed, Balaji Lakshminarayanan, Stephan Hoyer, and Rémi Munos. The cramer distance as a solution to biased wasserstein gradients. *arXiv preprint arXiv:1705.10743*, 2017. 11, 22, 27

Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandit algorithms with supervised learning guarantees. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 19–26, 2011. 83

Alberto Bietti, Alekh Agarwal, and John Langford. A contextual bandit bake-off. *arXiv preprint arXiv:1802.04064*, 2018. 83, 85

Mikołaj Bińkowski, Dougal J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 15, 27, 28, 29, 53, 101

Guy Blanc and Steffen Rendle. Adaptive sampled softmax with kernel based sampling. In *International Conference on Machine Learning*, pages 590–599, 2018. 124

OF Borisenko and LI Minchenko. Directional derivatives of the maximum function. *Cybernetics and Systems Analysis*, 28(2):309–312, 1992. 23

Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *The Journal of Machine Learning Research*, 14 (1):3207–3260, 2013. 75

George Box. Box and jenkins: time series analysis, forecasting and control. *A Very British Affair, ser. Palgrave Advanced Texts in Econometrics. Palgrave Macmillan UK*, 2013. 103

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1xsqj09Fm. 2, 57

E Brodsky and Boris S Darkhovsky. *Nonparametric methods in change point problems*. Springer Science & Business Media, 2013. 95

Brian Bullins, Cyril Zhang, and Yi Zhang. Not-so-random features. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk8XMWgRb. 25, 32, 35, 37, 38, 43

Russel E Caflisch. Monte carlo and quasi-monte carlo methods. *Acta numerica*, 7:1–49, 1998. 67

Joaquin Quinonero Candela, Agathe Girard, Jan Larsen, and Carl Edward Rasmussen. Propagation of uncertainty in bayesian kernel models-application to multiple-step ahead forecasting. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 2, pages II–701. IEEE, 2003. 103

Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717, 2009. 111

Andrea Caponnetto and Ernesto De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007. 79

Luigi Carratino, Alessandro Rudi, and Lorenzo Rosasco. Learning with sgd and random features. In *Advances in Neural Information Processing Systems (NIPS)*, pages 10213–10224, 2018. 79

Sarath Chandar, Mitesh M Khapra, Hugo Larochelle, and Balaraman Ravindran. Correlational neural networks. *Neural computation*, 28(2):257–285, 2016. 108, 112, 113

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 2009. 95

Wei-Cheng Chang, Chun-Liang Li, Yiming Yang, and Barnabas Poczos. Data-driven random fourier

features using stein effect. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017a. 5, 25, 32, 124

Wei-Cheng Chang, Yuexin Wu, Hanxiao Liu, and Yiming Yang. Cross-domain kernel induction for transfer learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017b. 6

Wei-Cheng Chang, Hsiang-Fu Yu, Inderjit S Dhillon, and Yiming Yang. SeCSeq: Semantic coding for sequence-to-sequence based extreme multi-label classification. In *Neural Information Processing Systems (NIPS) CDNNRIA Workshop*, 2018. 7

Wei-Cheng Chang, Chun-Liang Li, Yiming Yang, and Barnabás Póczos. Kernel change-point detection with auxiliary deep generative models. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=r1GbfhRqF7. 6, 39, 53

Wei-Cheng Chang, Rajat Sen, Yiming Yang, and Sanjay Shakkottai. Fast kernel contextual bandits with random fourier features. *Technical Report*, 2019b. 6

Wei-Cheng Chang, Chun-Liang Li, Youssef Mroueh, and Yiming Yang. Kernel stein generative modeling. *arXiv preprint arXiv:2007.03074*, 2020a. 5

Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. Pre-training tasks for embedding-based large-scale retrieval. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=rkg-mA4FDr. 7

Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit Dhillon. Taming pretrained transformers for extreme multi-label text classification. In *KDD*, 2020c. 7

Radha Chitta, Rong Jin, and Anil K Jain. Efficient kernel clustering using random fourier features. In *2012 IEEE 12th International Conference on Data Mining*, pages 161–170. IEEE, 2012. 67

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. 103

Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *International Conference on Machine Learning (ICML)*, pages 844–853, 2017. 75

Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 208–214, 2011. 75, 77

Kacper Chwialkowski, Heiko Strathmann, and Arthur Gretton. A kernel test of goodness of fit. In *JMLR: Workshop and Conference Proceedings*, 2016. 1, 49

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 1

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Generalization bounds for learning kernels. In *Proceedings of the 27th International Conference on Machine Learning*, ICML'10, page 247–254, Madison, WI, USA, 2010a. Omnipress. ISBN 9781605589077. 36

Corinna Cortes, Mehryar Mohri, and Ameet Talwalkar. On the impact of kernel approximation on learning accuracy. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 113–120, 2010b. 79, 87

Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz S Kandola. On kernel-target alignment. In *Advances in neural information processing systems*, pages 367–373, 2002. 34

Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina F Balcan, and Le Song. Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems*, pages 3041–3049, 2014. 38, 65

Bo Dai, Hanjun Dai, Arthur Gretton, Le Song, Dale Schuurmans, and Niao He. Kernel exponential family estimation via doubly dual embedding. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2321–2330, 2019. 123

Josef Dick, Frances Y Kuo, and Ian H Sloan. High-dimensional integration: the quasi-monte carlo way. *Acta Numerica*, 22:133–288, 2013. 67, 70

Peter J Diggle and Richard J Gratton. Monte carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 46(2):193–212, 1984. 2

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 2

Petros Drineas and Michael W Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6(Dec):2153–2175, 2005. 74

Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997. 1

Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Advances in Neural Information Processing Systems*, pages 5723–5733, 2019. 123

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems*, pages 3603–3613, 2019. 2, 47, 56, 57

Lixin Duan, Dong Xu, and Ivor W. Tsang. Learning with augmented features for heterogeneous domain adaptation. In *Proceedings of the International Conference on Machine Learning*, pages 711–718, Edinburgh, Scotland, June 2012. Omnipress. 107

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011. 112

Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. In *International Conference on Machine Learning (ICML)*, 2011. 83

Richard M Dudley. *Real analysis and probability*. CRC Press, 2018. 23

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. In *Proceedings of the International Conference on*

Learning Representations (ICLR), 2017. 20

David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B Tenenbaum, and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning (ICML)*, 2013. 25

Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. In *UAI*, 2015. 4, 11, 12, 16, 17, 41, 101

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008. 42, 113, 114

Yihao Feng, Dilin Wang, and Qiang Liu. Learning to draw samples with amortized stein variational gradient descent. In *UAI*, 2017. 2, 47

Dylan J Foster, Alekh Agarwal, Miroslav Dudík, Haipeng Luo, and Robert E Schapire. Practical contextual bandits with regression oracles. In *International Conference on Machine Learning (ICML)*, 2018. 75, 83, 85

Kenji Fukumizu, Arthur Gretton, Gert R Lanckriet, Bernhard Schölkopf, and Bharath K Sriperumbudur. Kernel choice and classifiability for rkhs embeddings of probability distributions. In *NIPS*, 2009. 13

Andrew B Gardner, Abba M Krieger, George Vachtsevanos, and Brian Litt. One-class novelty detection for seizure analysis from intracranial eeg. *JMLR*, 2006. 95

Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow gaussian processes. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bklfsi0cKm. 123

Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*, pages 129–143. Springer, 2003. 123

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520, 2011. 107, 108

Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *JMLR*, 2011. 2, 4, 25, 26, 34, 35, 38

Chengyue Gong, Jian Peng, and Qiang Liu. Quantile stein variational gradient descent for batch bayesian optimization. In *International Conference on Machine Learning*, 2019. 2, 47

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 2, 11, 12, 13, 15, 26, 28, 33

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hkxzx0NtDB. 2, 47, 56, 57

Arthur Gretton. Notes on the cramer gan. https://medium.com/towards-data-science/

notes-on-the-cramer-gan-752abd505c00, 2017. Accessed: 2017-11-2. 22

Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *NIPS*, 2007. 96

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012a. 1, 11, 12, 13, 14, 24, 25, 26, 39, 96, 98, 99

Arthur Gretton, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, Kenji Fukumizu, and Bharath K Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In *Advances in neural information processing systems*, pages 1205–1213, 2012b. 13, 96, 97

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017. xi, 4, 11, 14, 15, 21, 22, 29, 31, 56, 57, 100

Fredrik Gustafsson. The marginalized likelihood ratio test for detecting abrupt changes. *IEEE Transactions on automatic control*, 1996. 95

Fredrik Gustafsson and Fredrik Gustafsson. *Adaptive filtering and change detection*. Citeseer, 2000. 95

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1352–1361. JMLR. org, 2017. 47

William W Hager. Updating the inverse of a matrix. *SIAM review*, 31(2):221–239, 1989. 80

Zaid Harchaoui, Eric Moulines, and Francis R Bach. Kernel change-point analysis. In *Advances in neural information processing systems*, pages 609–616, 2009. 96

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 34

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Support vector learning for ordinal regression. In *In International Conference on Artificial Neural Networks*, pages 97–102, 1999. 15

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017. 28, 29, 55, 122

Geoffrey E Hinton and Russ R Salakhutdinov. Using deep belief nets to learn covariance kernels for gaussian processes. In *Advances in neural information processing systems*, pages 1249–1256, 2008. 25

Judy Hoffman, Erik Rodner, Jeff Donahue, Trevor Darrell, and Kate Saenko. Efficient learning of domain-invariant image representations. *arXiv preprint arXiv:1301.3224*, 2013. 113

Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008. 1

Junjie Hu, Wei-Cheng Chang, Yuexin Wu, and Graham Neubig. Contextual encoding for translation quality estimation. In *Proceedings of the Third Conference on Machine Translation: Shared Task*

Papers, pages 788–793, Belgium, Brussels, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6462. URL https://www.aclweb.org/anthology/W18-6462. 7

Po-Sen Huang, Haim Avron, Tara N Sainath, Vikas Sindhwani, and Bhuvana Ramabhadran. Kernel methods match deep neural networks on timit. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 205–209. IEEE, 2014. 65

Ferenc Huszár and David Duvenaud. Optimally-weighted herding is bayesian quadrature. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 377–386, 2012. 65, 69, 70

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709, 2005. 47, 49

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018. 53

Neal Jean, Sang Michael Xie, and Stefano Ermon. Semi-supervised deep kernel learning: Regression with unlabeled data by minimizing predictive variance. In *Advances in Neural Information Processing Systems*, pages 5327–5338, 2018. 39

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk99zCeAb. 56, 57

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 2

Yoshinobu Kawahara, Takehisa Yairi, and Kazuo Machida. Change-point detection in time-series data based on subspace identification. In *ICDM*. IEEE, 2007. 95, 96

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013. 16

Marius Kloft, Ulf Brefeld, Sören Sonnenburg, and Alexander Zien. Lp-norm multiple kernel learning. *The Journal of Machine Learning Research*, 12:953–997, 2011. 113

Andreas Krause and Cheng S Ong. Contextual gaussian process bandit optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2447–2455, 2011. 75, 76, 77, 78, 79, 80, 81, 83

Akshay Krishnamurthy, Alekh Agarwal, and Miro Dudik. Contextual semibandits via supervised learning oracles. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2388–2396, 2016. 75

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 17, 28, 37

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 34

Brian Kulis, Kate Saenko, and Trevor Darrell. What you saw is not what you get: Domain adaptation

using asymmetric kernel transforms. In *CVPR 2011*, pages 1785–1792. IEEE, 2011. 107

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966, 2015. 123

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems*, pages 3929–3938, 2019. 55, 56, 122

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104. ACM, 2018. 7, 102, 103

Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985. 77

Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, 2004. 25

John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems (NIPS)*, pages 817–824, 2008. 75, 83

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. 17, 37

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006. 47

Bin Li, Qiang Yang, and Xiangyang Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In *Proceedings of the 26th annual international conference on machine learning*, pages 617–624, 2009. 107

Chun-Liang Li and Barnabás Póczos. Utilize old coordinates: Faster doubly stochastic gradients for kernel methods. In *UAI*, 2016. 38, 65

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabas Poczos. MMD GAN: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2203–2213, 2017. 4, 25, 29, 30, 53, 99, 101

Chun-Liang Li, Wei-Cheng Chang, Youssef Mroueh, Yiming Yang, and Barnabas Poczos. Implicit kernel learning. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019. 4, 53, 101

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010. 75, 76, 77, 78, 81, 83

Shuang Li, Yao Xie, Hanjun Dai, and Le Song. M-statistic for kernel change-point detection. In *NIPS*, 2015a. 95, 96, 102, 103

Wen Li, Lixin Duan, Dong Xu, and Ivor W Tsang. Learning with augmented features for supervised and

semi-supervised heterogeneous domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 36(6):1134–1148, 2013. 113

Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning (ICML)*, pages 1718–1727, 2015b. 4, 11, 12, 16, 17, 21, 101

Zhu Li, Jean-Francois Ton, Dino Oglic, and Dino Sejdinovic. A unified analysis of random fourier features. *arXiv preprint arXiv:1806.09178*, 2018. 79

Tianyi Lin, Chi Jin, and Michael I Jordan. On gradient descent ascent for nonconvex-concave minimax problems. In *International Conference on Machine Learning (ICML)*, 2020. 122

Hanxiao Liu and Yiming Yang. Bipartite edge prediction via transductive learning over product graphs. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1880–1888, 2015. 110

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM, 2017. 7

Qiang Liu. Stein variational gradient descent as gradient flow. In *Advances in neural information processing systems*, pages 3115–3123, 2017. 2, 5, 47, 48, 56

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in neural information processing systems*, pages 2378–2386, 2016. 2, 5, 47, 48

Qiang Liu, Jason Lee, and Michael Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pages 276–284, 2016. 1, 49

Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 2013. 102, 103, 104

Yusha Liu, Chun-Liang Li, and Barnabás Póczos. Classifier two-sample test for video anomaly detections. In *BMVC*, 2018. 95

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *CVPR*, 2015. 11, 17

Mingsheng Long, Jianmin Wang, Guiguang Ding, Sinno Jialin Pan, and S Yu Philip. Adaptation regularization: A general framework for transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1076–1089, 2013. 108

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. In *Advances in neural information processing systems*, pages 700–709, 2018. 122

David JC MacKay. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 1995. 26

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. 16

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017. 11

David S Matteson and Nicholas A James. A nonparametric approach for multiple change point analysis of multivariate data. *Journal of the American Statistical Association*, 2014. 96, 104

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*, 2018. 122

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 100

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018. 56, 57, 100

Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016. 11

Youssef Mroueh and Tom Sercu. Fisher gan. In *Advances in Neural Information Processing Systems*, pages 2513–2523, 2017. 11

Youssef Mroueh, Tom Sercu, and Vaibhava Goel. Mcgan: Mean and covariance feature matching gan. In *International Conference on Machine Learning (ICML)*, 2017. 16

Youssef Mroueh, Chun-Liang Li, Tom Sercu, Anant Raj, and Yu Cheng. Sobolev gan. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 29, 101

Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, Arthur Gretton, and Bernhard Schölkopf. Kernel mean estimation and stein effect. In *International Conference on Machine Learning (ICML)*, pages 10–18, 2014. 5, 69

Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, Bernhard Schölkopf, et al. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10 (1-2):1–141, 2017. 22

Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, pages 429–443, 1997. 11

Marina Munkhoeva, Yermek Kapushev, Evgeny Burnaev, and Ivan Oseledets. Quadrature-based features for kernel approximation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 9147–9156, 2018. 124

Vaishnavh Nagarajan and J Zico Kolter. Gradient descent gan optimization is locally stable. In *Advances in neural information processing systems*, pages 5585–5595, 2017. 122

Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *Advances in Neural Information Processing Systems*, pages 5233–5243, 2019. 47, 56, 57

Maher Nouiehed, Maziar Sanjabi, Tianjian Huang, Jason D Lee, and Meisam Razaviyayn. Solving a class of non-convex min-max games using iterative first order methods. In *Advances in Neural Information*

Processing Systems, pages 14934–14942, 2019. 122

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *NIPS*, pages 271–279, 2016. 11

Junier B Oliva, Avinava Dubey, Andrew G Wilson, Barnabás Póczos, Jeff Schneider, and Eric P Xing. Bayesian nonparametric kernel-learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016. 25, 26, 35, 39

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009. 107

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010. 107

Andrey Pepelyshev and Aleksey S Polunchenko. Real-time financial surveillance via quickest change-point detection methods. *arXiv preprint arXiv:1509.01570*, 2015. 95

Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008. 78, 81

Peter Prettenhofer and Benno Stein. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 1118–1127, 2010. 112

Yuchen Pu, Zhe Gan, Ricardo Henao, Chunyuan Li, Shaobo Han, and Lawrence Carin. Vae learning via stein variational gradient descent. In *NIPS*, pages 4236–4245, 2017. 47

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. 17, 18, 28

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1177–1184, 2008. 1, 3, 5, 26, 27, 33, 36, 65, 66, 67, 76, 78

Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1313–1320, 2009. xi, 3, 33, 34, 37, 65, 76, 78

Alain Rakotomamonjy, Francis R Bach, Stéphane Canu, and Yves Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9(Nov):2491–2521, 2008. 2

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004. 77

Carl Edward Rasmussen and Zoubin Ghahramani. Bayesian monte carlo. *Advances in neural information processing systems*, pages 505–512, 2003. 67, 68

Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*. MIT press Cambridge, 2006. 68

Suman Ravuri, Shakir Mohamed, Mihaela Rosca, and Oriol Vinyals. Learning implicit generative models

with the method of learned moments. In *ICML*, 2018. 56, 57

Ankit Singh Rawat, Jiecao Chen, Felix Yu, Ananda Theertha Suresh, and Sanjiv Kumar. Sampled softmax with random fourier features. *arXiv preprint arXiv:1907.10747*, 2019. 124

Jaxk Reeves, Jien Chen, Xiaolan L Wang, Robert Lund, and Qi Qi Lu. A review and comparison of changepoint detection techniques for climate data. *Journal of Applied Meteorology and Climatology*, 2007. 95

Alessandro Rudi and Lorenzo Rosasco. Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3215–3225, 2017. 72, 79

Walter Rudin. *Fourier analysis on groups*, volume 121967. Wiley Online Library, 1962. 25, 26, 65, 66

Yunus Saatçi, Ryan Turner, and Carl Edward Rasmussen. Gaussian process change point models. In *International Conference on Machine Learning (ICML)*, 2010. 102, 103

Yunus Saatchi and Andrew G Wilson. Bayesian gan. In *NIPS*, pages 3625–3634, 2017. 39

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016. 19, 20, 28, 55, 122

Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001. 1, 76

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998. 1

Rajat Sen, Karthikeyan Shanmugam, and Sanjay Shakkottai. Contextual bandits with stochastic experts. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018. 75, 83

Vaishaal Shankar, Alex Fang, Wenshuo Guo, Sara Fridovich-Keil, Ludwig Schmidt, Jonathan Ragan-Kelley, and Benjamin Recht. Neural kernels without tangents. In *International Conference on Machine Learning (ICML)*, 2020. 123

Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pages 488–495, 2009. 123

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011. 123

Loïc Simon, Ryan Webster, and Julien Rabin. Revisiting precision and recall definition for generative model evaluation. *arXiv preprint arXiv:1905.05441*, 2019. 122

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 83, 85

Shashank Singh, Ananya Uppal, Boyue Li, Chun-Liang Li, Manzil Zaheer, and Barnabás Póczos. Nonparametric density estimation under adversarial losses. In *Advances in Neural Information Processing Systems*, pages 10225–10236, 2018. 122

Aman Sinha and John C Duchi. Learning kernels with random features. In *NIPS*, 2016. xi, 5, 25, 32, 34, 35, 36, 37, 38, 42, 74, 124

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11895–11907, 2019. 47, 49, 50, 55, 56, 57, 61, 123

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *arXiv preprint arXiv:2006.09011*, 2020. 123

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *UAI*, 2019. 47, 49

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning (ICML)*, 2010. 75, 77, 89

Bharath Sriperumbudur and Zoltán Szabó. Optimal rates for random fourier features. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1144–1152, 2015. 79

Charles Stein. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. Technical report, Stanford University Stanford United States, 1956. 5

Charles J Stone. Optimal global rates of convergence for nonparametric regression. *The annals of statistics*, pages 1040–1053, 1982. 122

Heiko Strathmann, Dino Sejdinovic, Samuel Livingstone, Zoltan Szabo, and Arthur Gretton. Gradient-free hamiltonian monte carlo with efficient kernel exponential families. In *Advances in Neural Information Processing Systems*, pages 955–963, 2015. 123

Yitong Sun, Anna Gilbert, and Ambuj Tewari. But how does it work in theory? linear svm with random features. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3383–3392, 2018. 79

Dougal Sutherland, Heiko Strathmann, Michael Arbel, and Arthur Gretton. Efficient and principled score estimation with nyström kernel exponential families. In *International Conference on Artificial Intelligence and Statistics*, pages 652–660, 2018. 123

Dougal J Sutherland and Jeff Schneider. On the error of random fourier features. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, UAI'15, page 862–871, Arlington, Virginia, USA, 2015. AUAI Press. ISBN 9780996643108. 79

Dougal J Sutherland, Hsiao-Yu Tung, Heiko Strathmann, Soumyajit De, Aaditya Ramdas, Alex Smola, and Arthur Gretton. Generative models and model criticism via optimized maximum mean discrepancy. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 17, 53, 96, 97, 98, 101

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014. 100

Zoltán Szabó and Bharath K Sriperumbudur. On kernel derivative approximation with random fourier features. *arXiv preprint arXiv:1810.05207*, 2018. 79

Jun-ichi Takeuchi and Kenji Yamanishi. A unifying framework for detecting outliers and change points from time series. *IEEE transactions on Knowledge and Data Engineering*, 2006. 95, 104

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 2012. 17

Alexandre B Tsybakov. *Introduction to nonparametric estimation*. Springer Science & Business Media, 2008. 122

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Adversarial generator-encoder networks. *arXiv preprint arXiv:1704.02304*, 2017. 16

Ananya Uppal, Shashank Singh, and Barnabás Póczos. Nonparametric density estimation & convergence of gans under besov ipm losses. In *Advances in Neural Information Processing Systems 32*, pages 9089–9100, 2019. 122

Michal Valko, Nathaniel Korda, Rémi Munos, Ilias Flaounas, and Nelo Cristianini. Finite-time analysis of kernelised contextual bandits. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 654–663, 2013. 75, 76, 77, 78

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011. 47, 49

S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242, 2010. 1, 123

SVN Vishwanathan, Alexander Johannes Smola, et al. Fast kernels for string and tree matching. *Kernel methods in computational biology*, 15:113–130, 2004. 1

Grace Wahba. *Spline models for observational data*, volume 59. Siam, 1990. 76

Dilin Wang and Qiang Liu. Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv preprint arXiv:1611.01722*, 2016. 47

Dilin Wang and Qiang Liu. Nonlinear stein variational gradient descent for learning diversified mixture models. In *International Conference on Machine Learning*, pages 6576–6585, 2019. 53

Shusen Wang and Zhihua Zhang. Improving cur matrix decomposition and the nyström approximation via adaptive sampling. *The Journal of Machine Learning Research*, 14(1):2729–2769, 2013. 74

Shusen Wang, Alex Gittens, and Michael W Mahoney. Sketched ridge regression: Optimization perspective, statistical perspective, and model averaging. *The Journal of Machine Learning Research*, 18(1):8039–8088, 2017. 70

Yao Wang, Chunguo Wu, Zhaohua Ji, Binghong Wang, and Yanchun Liang. Non-parametric change-point method for differential gene expression detection. *PloS one*, 2011. 95

D Warde-Farley and Y Bengio. Improving generative adversarial networks with denoising feature matching. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 20

Larry Wasserman. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013. 11, 14

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011. 2, 47

Li Wenliang, Dougal Sutherland, Heiko Strathmann, and Arthur Gretton. Learning deep kernels for exponential family densities. In *International Conference on Machine Learning*, pages 6737–6746, 2019. 50, 53, 123

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006. 1

Christopher KI Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001. 1, 74

Andrew Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In *International Conference on Machine Learning (ICML)*, 2013. xi, 2, 4, 25, 26, 29, 30, 37, 38

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016. 13, 25, 38, 53, 99

David P Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014. 70

Lingfei Wu, Ian EH Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar, and Michael J Witbrock. Word mover's embedding: From word2vec to document embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4524–4534, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1482. URL https://www.aclweb.org/anthology/D18-1482. 123

Lingfei Wu, Ian En-Hsu Yen, Siyu Huo, Liang Zhao, Kun Xu, Liang Ma, Shouling Ji, and Charu Aggarwal. Efficient global string kernel with random features: Beyond counting substructures. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 520–528, 2019a. 123

Lingfei Wu, Ian En-Hsu Yen, Zhen Zhang, Kun Xu, Liang Zhao, Xi Peng, Yinglong Xia, and Charu Aggarwal. Scalable global alignment graph kernel using random features: From node embedding to graph embedding. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1418–1428, 2019b. 123

Zhao Xu, Kristian Kersting, and Lorenzo von Ritter. Stochastic online anomaly analysis for streaming time series. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3189–3195, 2017. doi: 10.24963/ijcai.2017/445. URL https://doi.org/10.24963/ijcai.2017/445. 102

Kenji Yamanishi and Jun-ichi Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data. In *SIGKDD*. ACM, 2002. 95

Kenji Yamanishi, Jun-Ichi Takeuchi, Graham Williams, and Peter Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, 2004. 95

Jiyan Yang, Vikas Sindhwani, Haim Avron, and Michael Mahoney. Quasi-monte carlo feature maps for shift-invariant kernels. In *International Conference on Machine Learning (ICML)*, pages 485–493, 2014. 5, 65, 67, 70

Tianbao Yang, Yu-Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *Advances in neural information processing systems*, pages 476–484, 2012. 74

Zichao Yang, Andrew Wilson, Alex Smola, and Le Song. A la carte–learning fast kernels. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015. 25, 32

Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1975–1983, 2016. 5, 124

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 11, 17

Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. *arXiv preprint arXiv:1702.08811*, 2017. 16

Shuangfei Zhai, Yu Cheng, Rogério Schmidt Feris, and Zhongfei Zhang. Generative adversarial networks as variational training of energy based models. *CoRR*, abs/1611.01799, 2016. 16

Fuzhen Zhang. *The Schur complement and its applications*, volume 4. Springer Science & Business Media, 2006. 78

Yuchen Zhang, Percy Liang, and Moses Charikar. A hitting time analysis of stochastic gradient langevin dynamics. In *COLT*, 2017. 33

J. Zhao, M. Mathieu, and Y. LeCun. Energy-based Generative Adversarial Network. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 11, 13, 16, 21

Joey Tianyi Zhou, Sinno Jialin Pan, Ivor W Tsang, and Yan Yan. Hybrid heterogeneous transfer learning through deep learning. In *Proceedings of the national conference on artificial intelligence*, 2014. 107, 112, 113

Xiaojin Zhu, John Lafferty, and Ronald Rosenfeld. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, language technologies institute, school of . . . , 2005. 110, 113

Yin Zhu, Yuqiang Chen, Zhongqi Lu, Sinno Jialin Pan, Gui-Rong Xue, Yong Yu, and Qiang Yang. Heterogeneous transfer learning for image classification. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011. 107