

Kernel Embeddings of Conditional Distributions

Le Song[†], Kenji Fukumizu[‡] and Arthur Gretton^{*}

[†]*Georgia Institute of Technology*

[‡]*The Institute of Statistical Mathematics*

^{*}*University College London*

Abstract

Many modern applications of signal processing and machine learning, ranging from computer vision to computational biology, require the analysis of large volumes of high-dimensional continuous-valued measurements. Complex statistical features are commonplace, including multi-modality, skewness, and rich dependency structures. Such problems call for a flexible and robust modeling framework that can take into account these diverse statistical features. Most existing approaches, including graphical models, rely heavily on parametric assumptions. Variables in the model are typically assumed to be discrete-valued, or to be multivariate Gaussians; and linear relations between variables are often used. These assumptions can result in a model far different from the data generating process.

In this paper, we will review the recent development of nonparametric inference with *kernel embedding of conditional distributions*, and provide a novel, unified kernel framework for addressing a broad class of challenging problems in graphical models. The key idea is to map conditional distributions into infinite-dimensional feature spaces using kernels, such that subsequent comparisons and manipulations of these distributions are achieved via feature space operations: inner products, distances, projections, linear transformations, and spectral analysis. Basic probabilistic rules, such as the Sum Rule, Product Rule, and Bayes' Rule, then become linear operations on matrices of kernel values. We demonstrate the effectiveness of this kernel framework for inference, in a nonparametric belief propagation algorithm and in a hidden Markov model problem.

1 Introduction

A great many real-world signal processing problems exhibit complex statistical features, including multi-modality, skewness, nonlinear relations, and sophisticated dependency structures. For instance, in constructing depth maps from 2D images, the image features are high-dimensional and continuous valued, and the depth is both continuous valued and has a multi-modal distribution (Figure 1(a)). In predicting a sequence of camera movements from videos, the orientations of the camera are continuous and non-Euclidean, and are predicted from continuous video features (Figure 1(b)). There is a common need for a flexible and robust modeling framework which can take into account these diverse statistical features, as well as modeling the long-range and hierarchical dependencies among the variables.

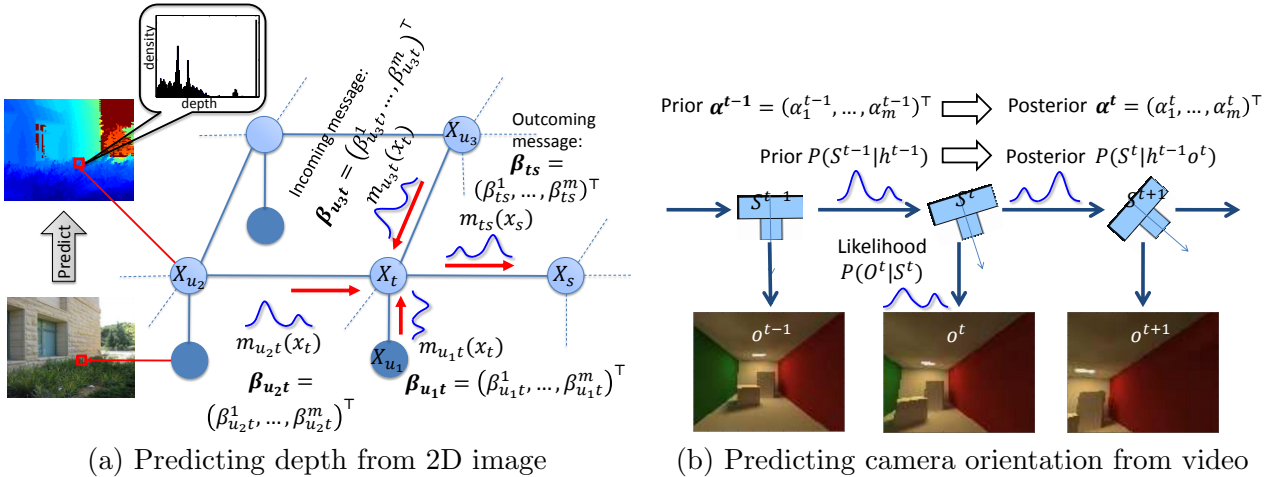


Figure 1: Applications (a) and (b) are challenging high-dimensional inference problems over continuous valued random variables with multi-modal distributions. Kernel embeddings of conditional distributions lead to simple algorithms to solve these problems, which update sets of weights using matrix-vector operations as we will discuss in more detail in section 5. For application (a), we will discuss a kernel belief propagation algorithm where the messages are represented and updated as weights β on kernel functions. For application (b), we will discuss a kernel filtering algorithm where the posterior distributions of the hidden states are represented and updated as weights α on kernel functions.

Probabilistic graphical models [1], which use a graph to encode the conditional independence structure between random variables, provide a general framework for representing high-dimensional and complex data. These models can be applied to a diverse range of problems (*e.g.*, exploratory

data analysis, feature extraction, handling missing values) using a common toolbox of learning and inference algorithms. That said, most existing graphical modeling tools rely heavily on parametric assumptions, such as discrete or multivariate Gaussian variables and linear relations. It remains challenging to handle more complex distributions in a data-driven fashion.

A range of approaches exist to capture rich statistical features in data. A common way to model higher-order statistical structures is to use high-dimensional nonlinear transformations, for instance the polynomial mapping $(x, y, z) \mapsto (x, y, z, xy, yz, xz, x^2, y^2, z^2, x^3, y^3 \dots)$ of the original variables. The number of such nonlinear features explodes exponentially in high dimensions, however, which limits the approach. Another representation for multi-modal distributions is as a finite mixture of Gaussians, but this approach also suffers from drawbacks. First, fitting a mixture of Gaussians is a non-convex problem, and it is often carried out using the Expectation-Maximization (EM) algorithm, leading to a local optimum. Second, the EM algorithm can be slow to converge, especially for high-dimensional data. Third, using a mixture of Gaussians in a graphical model inference algorithm creates significant computational challenges [2], as we will see in our later example on belief propagation. In this case, sampling or ad hoc approximations will be needed to make the problem computationally tractable, which can be slow and lack guarantees.

There exist additional methods for modeling arbitrary distributions nonparametrically, without using predefined polynomial features or a fixed number of Gaussians. Kernel density estimation is among the most famous approaches along these lines, yet it is well known that this approach is not suitable for high-dimensional data [3], and it typically does not take the dependence structure among the variables into account. It is also possible to use the characteristic function, which is the Fourier transform of a probability distribution function, to uniquely represent probability distributions. This approach also suffers in high dimensions, where an empirical estimate based on finite samples is not straightforward.

In this paper, we will review the recent development of nonparametric inference with *kernel embeddings of conditional distributions* [4, 5, 6, 7, 8, 9, 10, 11, 12], and provide a novel, unified kernel framework for addressing a broad class of challenging problems in graphical models. The key idea of this line of work is to *implicitly* map (conditional or marginal) distributions into *infinite*

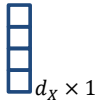
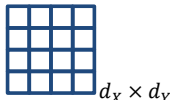
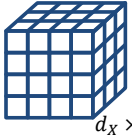
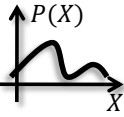
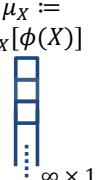
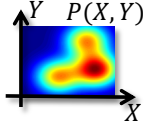
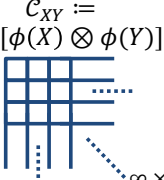
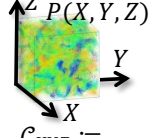
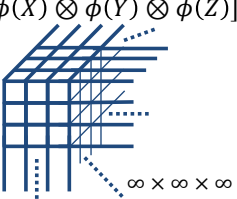
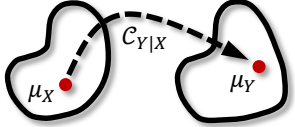
	Distributions			Probabilistic Operations
Discrete	$P(X)$  $d_x \times 1$	$P(X, Y)$  $d_x \times d_y$	$P(X, Y, Z)$  $d_x \times d_y \times d_z$	<i>Sum Rule:</i> $Q(X) = \sum_Y P(X Y)\pi(Y)$ <i>Product Rule:</i> $Q(X, Y) = P(X Y)\pi(Y)$ <i>Bayes Rule:</i> $Q(Y x) = \frac{P(x Y)\pi(Y)}{Q(X)}$
Kernel Embedding	 $P(X)$ $\mu_X := \mathbb{E}_X[\phi(X)]$  $\infty \times 1$	 $P(X, Y)$ $C_{XY} := \mathbb{E}_{XY}[\phi(X) \otimes \phi(Y)]$  $\infty \times \infty$	 $P(X, Y, Z)$ $C_{XYZ} := \mathbb{E}_{XYZ}[\phi(X) \otimes \phi(Y) \otimes \phi(Z)]$  $\infty \times \infty \times \infty$	 <i>Sum Rule:</i> $\mu_X^\pi = C_{X Y}\mu_Y^\pi$ <i>Product Rule:</i> $C_{XY}^\pi = C_{X Y}C_{Y X}^\pi$ <i>Bayes Rule:</i> $\mu_{Y x}^\pi = C_{Y x}^\pi\phi(x)$

Figure 2: Analogy between discrete and kernel embedding representations of marginal distributions and joint distributions of variable pairs and triplets. Probabilistic operations, such as conditioning, Sum Rule, Product Rule and Bayes’ Rule become linear operations on the embedding representations. The discrete case is a specific instance of our embedding framework, given an appropriate choice of kernel.

dimensional feature spaces using kernels, such that subsequent comparisons and manipulations of distributions can be achieved via feature space operations. This can be thought of as a generalization of the feature mapping of individual points, as used in classical kernel methods. By mapping probabilities into infinite-dimensional feature spaces, we can ultimately capture all the statistical features of arbitrary distributions. By virtue of the so-called kernel trick, we are able to avoid working explicitly with the infinite-dimensional features, instead expressing our algorithms entirely in terms of Gram matrices of training samples. The infinite and implicit nature of the feature spaces provides us a rich yet efficient framework for handling arbitrary distributions and high-dimensional data.

The conditional embedding framework represents the building blocks from probabilistic graphical models, such as marginal distributions over single variables, joint distributions over variable pairs, triplets and more, as infinite-dimensional vectors, matrices, tensors and high-order tensors respectively; furthermore, the operations fundamental to probabilistic reasoning and graphical mod-

els, *i.e.*, conditioning, Sum Rule, Product Rule and Bayes' Rule, become linear transformations and relations between the embeddings (see Figure 2 for the analogy between discrete probability tables and kernel embeddings of distributions). We may combine these building blocks so as to reason about interactions between a large collection of variables, even in the absence of parametric models.

The kernel conditional embedding framework has many advantages. First, it allows us to model data with diverse statistical features without the need to make restrictive assumptions about the type of distributions and relations. Second, it allows us to apply a large pool of linear and multi-linear algebraic (tensor) tools to accomplish learning tasks in the presence of sophisticated dependency structures, giving rise to methods for structure discovery, inference, parameter learning, and latent feature extraction. Third, this framework can be applied not only to continuous variables, but also can be generalized to variables which may take values on strings, graphs, groups, manifolds, and other domains on which kernels may be defined. Fourth, the computation can be implemented in practice by simple linear algebraic manipulation of kernel matrices.

We will mainly focus on two applications: the first being a belief propagation algorithm for inference in nonparametric graphical models (*i.e.*, estimating depth from still image features, reported in [7]); and the second being a dynamical systems model (*i.e.*, predicting camera movements from video features, reported in [4]). In the first application, multi-modal components in graphical models often make inference in these models intractable. Previous approaches using particle filtering and ad hoc approximation with mixtures of Gaussians are slow and inaccurate. Using kernel embeddings of conditional distributions, we are able to design a more accurate and efficient algorithm for the problem. In the second application, both the observations and hidden states of the hidden Markov model are complex high-dimensional variables, and it is not easy to capture the structure of the data using parametric models. Kernel embeddings of conditional distributions and kernel Bayes' rule can be used to model such problems with better accuracy. Finally, there exist many other recent applications of kernel embeddings of conditional distributions to signal processing and machine learning problems, including Markov decision processes (MDP) [9], partially observable MDPs (POMDPs) [10], hidden Markov models [6] and general latent variable graphical models [8].

2 Kernel Embedding of Distributions

We begin by providing an overview of kernel embeddings of distributions, which are *implicit* mappings of distributions into potentially *infinite* dimensional feature spaces.¹ We will see that the definition is quite simple, yet the framework offers a number of advantages. For instance, it can be used to design simpler and more effective statistics than alternatives for comparing continuous valued multi-modal distributions, *e.g.*, for the depth variable in Figure 1 application (a), and the camera rotation variable in Figure 1 application (b).

We denote by X a random variable with domain Ω and distribution $P(X)$, and refer to instantiations of X by the lower case character, x . We will focus on continuous domains, and denote the corresponding density by $p(X)$. Similarly, we denote random variable Y with distribution $P(Y)$ and density $p(Y)$. For simplicity of notation, we assume that the domain of Y is also Ω , but the methodology applies to the cases where X and Y have different domains.

A *reproducing kernel Hilbert space (RKHS)* \mathcal{F} on Ω with a kernel $k(x, x')$ is a Hilbert space of functions $f : \Omega \mapsto \mathbb{R}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{F}}$. Its element $k(x, \cdot)$ satisfies the reproducing property: $\langle f(\cdot), k(x, \cdot) \rangle_{\mathcal{F}} = f(x)$, and consequently, $\langle k(x, \cdot), k(x', \cdot) \rangle_{\mathcal{F}} = k(x, x')$, meaning that we can view the evaluation of a function f at any point $x \in \Omega$ as an inner product. Alternatively, $k(x, \cdot)$ can be viewed as an implicit feature map $\phi(x)$ where $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{F}}$.² Popular kernel functions on \mathbb{R}^n include the polynomial kernel $k(x, x') = (\langle x, x' \rangle + c)^d$ and the Gaussian RBF kernel $k(x, x') = \exp(-\sigma \|x - x'\|^2)$. Kernel functions have also been defined on graphs, time series, dynamical systems, images and other structured objects [13]. Thus the methodology presented below can readily be generalized to a diverse range of data types as long as kernel functions are defined for them.

¹By “implicit”, we mean that we do not need to explicitly construct the feature spaces, and the actual computations boil down to kernel matrix operations.

²For simplicity of notation, we use the same kernel for y , *i.e.*, $k(y, y') = \langle \phi(y), \phi(y') \rangle_{\mathcal{F}}$

2.1 Population Definition

The kernel embedding approach represents a probability distribution by an element in the RKHS associated with a kernel function [14, 15, 16, 17, 18],

$$\mu_X := \mathbb{E}_X [\phi(X)] = \int_{\Omega} \phi(x) dP(x), \quad (1)$$

where the distribution is mapped to its expected feature map, *i.e.*, to a point in a potentially infinite-dimensional and implicit feature space. The mean embedding μ_X has the property that the expectation of any RKHS function f can be evaluated as an inner product in \mathcal{F} , $\langle \mu_X, f \rangle_{\mathcal{F}} := \mathbb{E}_X[f(X)] \quad \forall f \in \mathcal{F}$. Note that the mean embedding is different from a kernel density estimator, where the density is convolved with a *smoothing* kernel $\tilde{k}(x, x')$ which is a valid probability density, resulting in representation $\mathbf{kde}(x') := \mathbb{E}_X[\tilde{k}(X, x')]$. In \mathbf{kde} , the smoothing kernel may not be positive semidefinite and in general $\tilde{k}(x, x') \neq \langle \tilde{\phi}(x), \tilde{\phi}(x') \rangle_{\mathcal{F}}$. Furthermore, even when the smoothing kernel function is positive semidefinite (*e.g.*, Gaussian RBF kernel $\tilde{k}(x, x') = \exp(-\sigma \|x - x'\|^2)$), the kernel bandwidth in \mathbf{kde} (and hence the feature space) often changes depending on the number of points observed, and hence can not be interpreted as embedding distributions into a *fixed* feature space.

Kernel embeddings can be readily generalized to joint distributions of two or more variables using tensor product feature spaces. For instance, we can embed a joint distribution of two variables X and Y into a tensor product feature space $\mathcal{F} \otimes \mathcal{F}$ by

$$\mathcal{C}_{XY} := \mathbb{E}_{XY}[\phi(X) \otimes \phi(Y)] = \int_{\Omega \times \Omega} \phi(x) \otimes \phi(y) dP(x, y) \quad (2)$$

where we assume for simplicity that the two variables share the same domain Ω and kernel k , and the tensor product features satisfy $\langle \phi(x) \otimes \phi(y), \phi(x') \otimes \phi(y') \rangle_{\mathcal{F} \otimes \mathcal{F}} = k(x, x')k(y, y')$.

As special cases, the marginal probability vector of a discrete variable X , and the probability table of the joint distribution of discrete variables X and Y , are both kernel embeddings. To see this, let $x, y \in \{1, \dots, N\}$ and use Kronecker delta kernel $k(x, x') = \delta(x, x')$. The corresponding

feature map $\phi(x)$ is then the standard basis of e_x in \mathbb{R}^N . Then

$$\begin{pmatrix} P(x=1) \\ \vdots \\ P(x=N) \end{pmatrix} = \mathbb{E}_X[e_X] = \mu_X, \quad \begin{pmatrix} P(x=s, y=t) \end{pmatrix} = \mathbb{E}_{XY}[e_X \otimes e_Y] = \mathcal{C}_{XY}. \quad (3)$$

The joint embeddings can also be viewed as an uncentered cross-covariance operator $\mathcal{C}_{XY} : \mathcal{F} \mapsto \mathcal{F}$ by the standard equivalence between a tensor and a linear map. That is, given two functions $f, g \in \mathcal{F}$, their covariance can be computed by $\mathbb{E}_{XY}[f(X)g(Y)] = \langle f, \mathcal{C}_{XY}g \rangle_{\mathcal{F}}$, or equivalently $\langle f \otimes g, \mathcal{C}_{XY} \rangle_{\mathcal{F} \otimes \mathcal{F}}$, where in the former we view \mathcal{C}_{XY} as an operator while in the latter we view it as an element in tensor product space. By analogy, $\mathcal{C}_{XX} := \mathbb{E}_X[\phi(X) \otimes \phi(X)]$ and $\mathcal{C}_{(XX)Y} := \mathbb{E}_X[\phi(X) \otimes \phi(X) \otimes \phi(Y)]$ can also be defined, the latter of which can be regarded as a linear operator from \mathcal{F} to $\mathcal{F} \otimes \mathcal{F}$. It will be clear from the context whether we use \mathcal{C}_{XY} as an operator between two spaces or as an element from a tensor product feature space.

Although the definition of embedding in (1) is simple, it turns out to have both rich representational power and a well-behaved empirical estimate. First, the mapping is injective for characteristic kernels [17, 18]. That is, if two distributions, $P(X)$ and $Q(Y)$, are different, they will be mapped to two distinct points in the feature space. Many commonly used kernels are characteristic, such as the Gaussian RBF kernel $\exp(-\sigma\|x - x'\|^2)$ and Laplace kernel $\exp(-\sigma\|x - x'\|)$, which implies that if we embed distributions using these kernels, the distance of the mappings in feature space will give us an indication whether two distributions are identical or not. This intuition has been exploited to design state-of-the-art two-sample tests [19] and independence tests [20]. For the former case, the test statistic is the squared distance between the embeddings of $P(Y)$ and $Q(Y)$, *i.e.*, $\text{mmd}(X, Y) := \|\mu_X - \mu_Y\|_{\mathcal{F}}^2$. For the latter case, the test statistic is the squared distance between the embeddings of a joint distribution $P(X, Y)$ and the product of its marginals $P(X)P(Y)$, *i.e.*, $\text{hsic}(X, Y) := \|\mathcal{C}_{XY} - \mu_X \otimes \mu_Y\|_{\mathcal{F} \otimes \mathcal{F}}^2$. Similarly, this statistic also has advantages over the `kde`-based statistic. We will further discuss these tests in the next section, following our introduction of finite sample estimates of the distribution embeddings and test statistics.

2.2 Finite Sample Kernel Estimator

While we rarely have access to the true underlying distribution, $P(X)$, we can readily estimate its embedding using a finite sample average. Given a sample $\mathcal{D}_X = \{x_1, \dots, x_m\}$ of size m drawn *i.i.d.* from $P(X)$, the empirical kernel embedding is

$$\hat{\mu}_X = \frac{1}{m} \sum_{i=1}^m \phi(x_i). \quad (4)$$

See Figure 3 for an illustration of the kernel embedding and its empirical estimator. This empirical estimate converges to its population counterpart in RKHS norm, $\|\hat{\mu}_X - \mu_X\|_{\mathcal{F}}$, with a rate of $O_p(m^{-\frac{1}{2}})$ [15, 16]. We note that this rate is independent of the dimension of X , meaning that statistics based on kernel embeddings circumvent the curse of dimensionality.

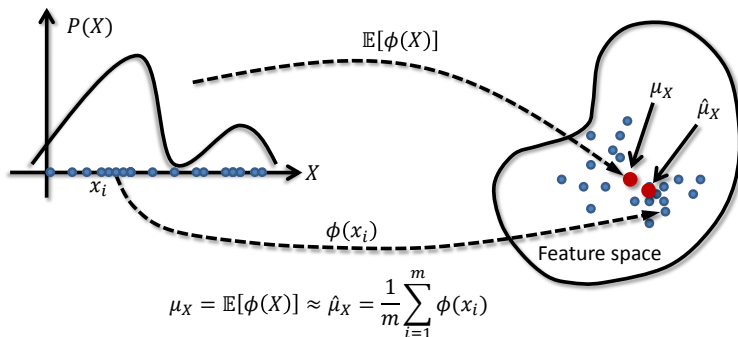


Figure 3: Kernel embedding of a distribution and finite sample estimate.

Kernel embeddings of joint distributions inherit the previous two properties of general embeddings: injectivity and easy empirical estimation. Given m pairs of training examples $\mathcal{D}_{XY} = \{(x_1, y_1), \dots, (x_m, y_m)\}$ drawn *i.i.d.* from $P(X, Y)$, the covariance operator \mathcal{C}_{XY} can then be estimated as

$$\hat{\mathcal{C}}_{XY} = \frac{1}{m} \sum_{i=1}^m \phi(x_i) \otimes \phi(y_i). \quad (5)$$

See Figure 4 for an illustration of the kernel joint embedding and its empirical estimator.

By virtue of the kernel trick, most of the computation required for statistical inference using kernel embeddings can be reduced to the Gram matrix manipulation. The entries in the Gram

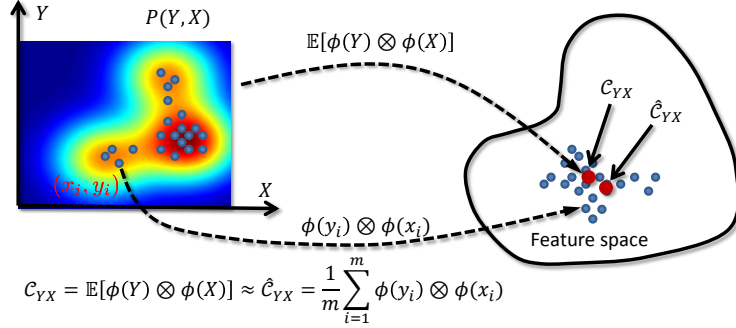


Figure 4: Kernel embedding of a joint distribution and finite sample estimate.

matrix K correspond to the kernel value between data points x_i and x_j , *i.e.*, $K_{ij} = k(x_i, x_j)$, and therefore its size is determined by the number of data points in the sample (similarly Gram matrix G has entries $G_{ij} = k(y_i, y_j)$). The size of the Gram matrices is in general much smaller than the dimension of the feature spaces (which can be infinite). This enables efficient nonparametric methods using the kernel embedding representation. For instance, the empirical `mmd` can be computed using kernel evaluations,

$$\widehat{\text{mmd}}(P, Q) = \left\| \frac{1}{m} \sum_{i=1}^m \phi(x_i) - \frac{1}{m} \sum_{i=1}^m \phi(y_i) \right\|_{\mathcal{F}}^2 = \frac{1}{m^2} \sum_{i,j=1}^m (k(x_i, x_j) + k(y_i, y_j) - 2k(x_i, y_j)).$$

For comparison, the L_2 distance between kernel density estimates is

$$\int_{\Omega} (\widehat{\text{kde}}(x) - \widehat{\text{kde}}'(x))^2 dx = \frac{1}{m^2} \int_{\Omega} \sum_{i,j=1}^m \left(\tilde{k}(x_i, x) \tilde{k}(x_j, x) + \tilde{k}(y_i, x) \tilde{k}(y_j, x) - 2\tilde{k}(x_i, x) \tilde{k}(y_j, x) \right) dx,$$

where $\widehat{\text{kde}}(x) = \frac{1}{m} \sum_{i=1}^m \tilde{k}(x_i, x)$ and $\widehat{\text{kde}}'(x) = \frac{1}{m} \sum_{i=1}^m \tilde{k}(y_i, x)$ respectively. Furthermore, it can be shown that a two-sample test based on the L_2 distance between kernel density estimates has less power against local departures from the null hypothesis than the MMD [19, Section 3.3; Section 5], due to the shrinking kernel bandwidth with increasing sample size. There are also many domains such as strings and graphs [13] where kernel methods can be used, but where probability densities may not be defined. Finally, hyper-parameters of the kernel functions, such as the bandwidth σ in the Gaussian RBF kernel $\exp(-\sigma \|x - x'\|^2)$, can be chosen to maximize the test power, and minimize the probability of type II error in two-sample tests (*i.e.* the probability of mistakenly declaring

P and Q are the same when they are in fact different) [21]. MMD and other distances between samples of points were also suggested in [22], although the injectivity of the map of probability measures to feature space, and the question of whether a metric was induced on probabilities, were not addressed.

If the sample size is large, the computation in kernel embedding methods may be expensive. In this case, a popular solution is to use a low-rank approximation of the Gram matrix, such as incomplete Cholesky factorization [23], which is known to work very effectively in reducing computational cost of kernel methods, while maintaining the approximation accuracy.

3 Kernel Embeddings of Conditional Distributions

While kernel embeddings of distributions provide a powerful framework for dealing with a number of challenging high-dimensional nonparametric problems, there remain many issues to be addressed in using these embeddings for inference. For instance, in Figure 1 application (a) and (b), there are many random variables interlinked via graphical models, and a simple embedding of distribution is not able to exploit the knowledge of these conditional independence relations. Furthermore, a key inference task in graphical models is to compute the posterior distribution of some hidden variables (*e.g.*, depth variable in application (a) and camera orientation in application (b)) given some observed evidence (*e.g.*, image pixel values in application (a) and video frames in application (b)). Such conditional operation is also missing in kernel embedding of distributions. Thus, in this section we will present a kernel nonparametric counterpart for conditional distributions, which will allow us to take into account conditional independence relations in graphical models, and subsequently provide a unified kernel framework for the Sum Rule, Product Rule and Bayes' Rule, essential for probabilistic reasoning in graphical models.

3.1 Population Definition

The kernel embedding of a conditional distribution $P(Y|X)$ is defined as [4]

$$\mu_{Y|x} := \mathbb{E}_{Y|x}[\phi(Y)] = \int_{\Omega} \phi(y) dP(y|x). \quad (6)$$

Given this embedding, the conditional expectation of a function $g \in \mathcal{F}$ can be computed as $\mathbb{E}_{Y|x}[g(Y)] = \langle g, \mu_{Y|x} \rangle_{\mathcal{F}}$. This may be compared with the property of the mean embedding in the last section, where the *unconditional* expectation of a function may be written as an inner product with the embedding. Unlike the embeddings discussed in the previous section, an embedding of conditional distribution is not a single element in the RKHS, but will instead sweep out a family of points in the RKHS, each indexed by a fixed value x of the conditioning variable X . It is only by fixing X to a particular value x , that we will be able to obtain a single RKHS element, $\mu_{Y|x} \in \mathcal{F}$. In other words, we need to define an operator, denoted as $\mathcal{C}_{Y|X}$, which can take as input an x and output an embedding. More specifically, we will want it to satisfy

$$\mu_{Y|x} = \mathcal{C}_{Y|X}\phi(x). \quad (7)$$

How do we define such a conditional embedding operator? Based on the relation between conditional expectation and covariance operators, Song et al. [4] show that, under the assumption $\mathbb{E}_{Y|\cdot}[g(Y)] \in \mathcal{F}$,

$$\mathcal{C}_{Y|X} := \mathcal{C}_{YX}\mathcal{C}_{XX}^{-1}, \quad \text{and hence } \mu_{Y|x} = \mathcal{C}_{YX}\mathcal{C}_{XX}^{-1}\phi(x) \quad (8)$$

satisfy the requirement in (7). See Figure 5 for an illustration of kernel embeddings of conditional distributions and the conditional embedding operator. We remark that the assumption $\mathbb{E}_{Y|\cdot}[g(Y)] \in \mathcal{F}$ always holds for finite domains with characteristic kernels, but it is not necessarily true for continuous domains [14]. In the cases where the assumption does not hold, we will use the expression $\mathcal{C}_{YX}\mathcal{C}_{XX}^{-1}\phi(x)$ as an approximation of the conditional mean $\mu_{Y|x}$. In practice, the inversion of the operator can be replaced by the regularized inverse $(\mathcal{C}_{XX} + \lambda I)^{-1}$.

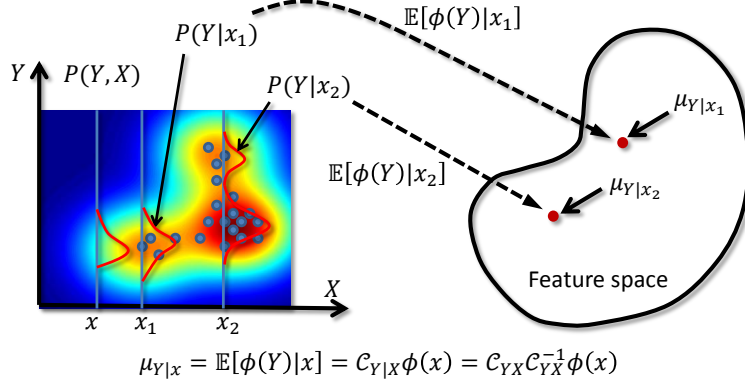


Figure 5: Kernel embedding of a conditional distribution.

The definition of the conditional embedding operator in (8) is very general, and the conditional probability $P(Y|X)$ for discrete variables becomes a special case. For instance, if we use a Kronecker delta kernel and a construction similar to (3), we can obtain the conditional probability table by

$$\underbrace{\begin{pmatrix} P(y = s|x = t) \end{pmatrix}}_{C_{Y|X}} = \underbrace{\begin{pmatrix} P(y = s, x = t) \end{pmatrix}}_{C_{YX}} \underbrace{\begin{pmatrix} P(x = 1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & P(x = N) \end{pmatrix}^{-1}}_{C_{XX}^{-1}}. \quad (9)$$

3.2 Finite Sample Kernel Estimator

Given a dataset $\mathcal{D}_{XY} = \{(x_1, y_1), \dots, (x_m, y_m)\}$ of size m drawn *i.i.d.* from $P(X, Y)$, we will estimate the conditional embedding operator as

Kernel conditional embedding operator:
 $\hat{C}_{Y|X} = \Phi(K + \lambda I)^{-1} \Upsilon^\top$

(10)

where $\Phi := (\phi(y_1), \dots, \phi(y_m))$ and $\Upsilon := (\phi(x_1), \dots, \phi(x_m))$ are implicitly formed feature matrix, and $K = \Upsilon^\top \Upsilon$ is the Gram matrix for samples from variable X . Furthermore, we need the additional regularization parameter λ to avoid overfitting. Then $\hat{\mu}_{Y|x} = \hat{C}_{Y|X} \phi(x)$ becomes a

weighted sum of feature mapped data points from Y ,

Kernel embedding of conditional distribution:

$$\begin{aligned} \widehat{\mu}_{Y|x} &= \sum_{i=1}^m \beta_i(x) \phi(y_i) = \Phi \boldsymbol{\beta}(x) \quad \text{where} \\ \boldsymbol{\beta}(x) &= (\beta_1(x), \dots, \beta_m(x))^\top = (K + \lambda I)^{-1} K_{:x}, \end{aligned} \tag{11}$$

and $K_{:x} = (k(x, X_1), \dots, k(x, X_m))^\top$. The empirical estimator of the conditional embedding is similar to the estimator of the ordinary embedding from equation (1). The difference is that, instead of applying uniform weights $\frac{1}{m}$, the former applies *non-uniform* weights, $\beta_i(x)$, on observations which are, in turn, determined by the value x of the conditioning variable. These non-uniform weights reflect the effects of conditioning on the embeddings. It is also shown that this empirical estimate converges to its population counterpart in RKHS norm, $\|\widehat{\mu}_{Y|x} - \mu_{Y|x}\|_{\mathcal{F}}$, with rate of $O_p(m^{-\frac{1}{4}})$ if one decreases the regularization λ with rate $O(m^{-\frac{1}{2}})$. With appropriate assumptions on the joint distribution of X and Y , better rates can be obtained [24].

The estimator of conditional embedding operator in (11) is very different from the conditional kernel density estimator $\widehat{\text{ckde}}(y|x) = \frac{\sum_{i=1}^m k(y_i, y) k(x_i, x)}{\sum_{i=1}^m k(x_i, x)}$, which has difficulty for high-dimensional data and in regions where $p(x)$ is small. By contrast, conditional embedding operators do not directly estimate the density, and hence avoid these problems. Given a function from RKHS, $g(y) = \sum_{i=1}^{\tilde{m}} \alpha_i k(\tilde{y}_i, y)$, we can estimate its expected value with respect to the conditional distribution using $\widehat{\mu}_{Y|x}$ via matrix operations,

$$\mathbb{E}_{Y|x}[g(Y)] \approx \langle g, \widehat{\mu}_{Y|x} \rangle_{\mathcal{F}} = \sum_{i=1}^{\tilde{m}} \sum_{j=1}^m \alpha_i \beta_j(x) k(\tilde{y}_i, y_j) = \boldsymbol{\alpha}^\top \tilde{G} (K + \lambda I)^{-1} K_{:x}$$

where \tilde{G} is a Gram matrix with entries $\tilde{G}_{ij} = k(\tilde{y}_i, y_j)$ and $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_{\tilde{m}})^\top$. In contrast, the conditional kernel density estimator requires integration over Ω for the estimation,

$$\mathbb{E}_{Y|x}[g(Y)] \approx \int_{\Omega} g(y) \widehat{\text{ckde}}(y|x) dy = \int_{\Omega} \frac{\sum_{i=1}^{\tilde{m}} \sum_{j=1}^m \alpha_i k(\tilde{y}_i, y) k(y_j, y) k(x_j, x)}{\sum_{j=1}^m k(x_j, x)} dy,$$

which may be difficult for high-dimensional Y .

The conditional mean embedding operator can alternatively be found as the solution to a function-valued least squares regression problem [24]: $\hat{\mathcal{C}}_{Y|X} = \arg \min_{\mathcal{C}: \mathcal{F} \rightarrow \mathcal{F}} \sum_{i=1}^m \|\phi(y_i) - \mathcal{C}\phi(x_i)\|_{\mathcal{F}}^2 + \lambda \|\mathcal{C}\|_{HS}^2$, where $\|\mathcal{C}\|_{HS}$ denotes the Hilbert-Schmidt norm (generalized Frobenius norm) of operator \mathcal{C} (See Figure 6 for an illustration). One practical consequence of this connection is that hyper-parameters in the conditional embeddings, such as parameters for kernels on the variable X and the regularization parameter λ , can be selected using cross-validation based on the regression objective.

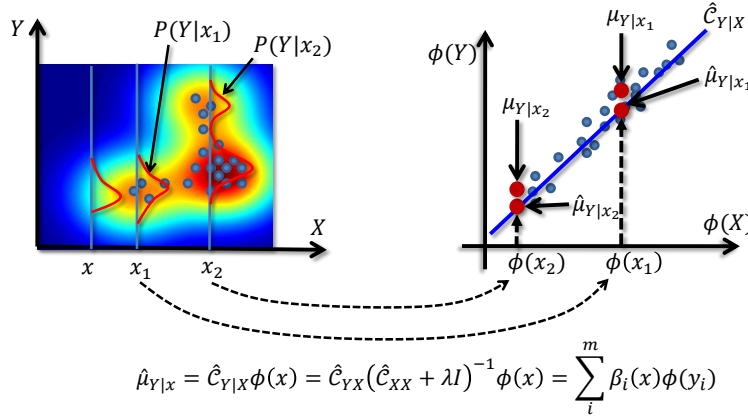


Figure 6: The feature space regression interpretation of the conditional embedding.

4 Probabilistic Reasoning with Kernel Conditional Embeddings

Besides inheriting many useful properties from ordinary embeddings, embeddings of conditional distributions extend our ability to manipulate distributions using kernels. The key rules of probabilistic inference, namely the Sum Rule, Product Rule and Bayes' rule, can all be kernelized using conditional embeddings. With these additional building blocks, we are able to perform nearly all graphical model operations in a unified nonparametric framework, and this allows us to address an even larger family of problems in a statistically and computationally efficient manner.

As we will see in the applications section, kernel embeddings of conditional distributions and kernel probabilistic reasoning provide the foundations for state-of-the-art algorithms in difficult problems arising from computer vision and hidden Markov models. For the computer vision application (Figure 1(a)), a kernel version of the Sum Rule needs to be used where nonparametric

multi-modal incoming messages are aggregated to produce a nonparametric outgoing message. For the hidden Markov model application (Figure 1(b)), a kernel Bayes rule needs to be used where nonparametric posterior distributions of the hidden states are maintained given video observations.

In the following, we will denote a joint distribution over X and Y by $P(X, Y)$, with marginal distributions $P(X) = \int_{\Omega} P(X, dy)$ and $P(Y) = \int_{\Omega} P(dx, Y)$, and conditional distribution $P(Y|X) = \frac{P(X, Y)}{P(X)}$. The conditional embedding operator associated with $P(Y|X)$ is $\mathcal{C}_{Y|X}$; similarly, $P(X|Y) = \frac{P(X, Y)}{P(Y)}$ is associated with $\mathcal{C}_{X|Y}$. Furthermore, we will denote a prior distribution over Y by $\pi(Y)$.

4.1 Kernel Sum Rule

$Q(X) = \int_{\Omega} P(X|Y)d\pi(Y)$ — we compute the marginal distribution of variable X by integrating out variable Y . Embedding distribution $Q(X)$, we have $\mu_X^\pi = \mathbb{E}_X[\phi(X)] = \mathbb{E}_Y \mathbb{E}_{X|Y}[\phi(X)]$. Note that we use the super script $(\cdot)^\pi$ to emphasize that $Q(X)$ makes use of the prior distribution $\pi(Y)$, and it can be different from the marginal $P(X)$ obtained from the joint distribution $P(X, Y)$. Plugging in the conditional embedding, we obtain the kernel Sum Rule,

$$\mu_X^\pi = \mathbb{E}_Y [\mathcal{C}_{X|Y} \phi(Y)] = \mathcal{C}_{X|Y} \mathbb{E}_Y [\phi(Y)] = \mathcal{C}_{X|Y} \mu_Y^\pi. \quad (12)$$

In other words, one can think of the input μ_Y^π , the embedding for $\pi(Y)$, being mapped to the output μ_X^π , the embedding for $Q(X)$. The operator that enables this transformation is the conditional embedding operator, which keeps the kernel embedding representation closed, and the probabilistic semantics intact.

Instead of using feature $\phi(x)$, we can also use a tensor product feature³ $\phi(x) \otimes \phi(x)$ to embed distribution $Q(X)$, resulting in $\mathcal{C}_{XX}^\pi = \mathbb{E}_X[\phi(x) \otimes \phi(x)]$. Then the kernel Sum Rule becomes

$$\mathcal{C}_{XX}^\pi = \mathcal{C}_{(XX)|Y} \mu_Y^\pi \quad (13)$$

where we used a conditional embedding operator $\mathbb{E}_{X|y}[\phi(X) \otimes \phi(X)] = \mathcal{C}_{(XX)|Y} \phi(y)$ for tensor product features. In the discrete case with Kronecker delta kernel, the two embedding versions of

³We can use even higher order features, such as $\phi(x) \otimes \phi(x) \otimes \phi(x)$.

the Sum Rule are

$$\underbrace{\begin{pmatrix} Q(x=1) \\ \vdots \\ Q(x=N) \end{pmatrix}}_{\mu_X^\pi} = \underbrace{\begin{pmatrix} P(x=s|y=t) \end{pmatrix}}_{\mathcal{C}_{X|Y}} \underbrace{\begin{pmatrix} \pi(y=1) \\ \vdots \\ \pi(y=N) \end{pmatrix}}_{\mu_Y^\pi},$$

$$\underbrace{\begin{pmatrix} Q(x=1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & Q(x=N) \end{pmatrix}}_{\mathcal{C}_{XX}^\pi} = \underbrace{\begin{pmatrix} \delta(s,s')P(x=s|y=t) \end{pmatrix}}_{\mathcal{C}_{(XX)|Y}} \underbrace{\begin{pmatrix} \pi(y=1) \\ \vdots \\ \pi(y=N) \end{pmatrix}}_{\mu_Y^\pi},$$

where $\mathcal{C}_{(XX)|Y}$ is a 3rd order tensor with the (s, s', t) -th entry being $\delta(s, s')P(x = s|y = t)$, and $\mathcal{C}_{(XX)|Y}\mu_Y^\pi$ denotes summation over the common index y . The use of tensors allows us to express the second version of the embedding Sum Rule as a multi-linear algebraic operation.

In general, we assume an estimator $\hat{\mu}_Y^\pi$ is given as weighted sum $\sum_{i=1}^{\tilde{m}} \alpha_i \phi(\tilde{y}_i)$ with some sample $\mathcal{D}_Y = \{\tilde{y}_1, \dots, \tilde{y}_{\tilde{m}}\}$; Furthermore, we obtain the estimated conditional embedding operator, $\hat{\mathcal{C}}_{X|Y}$, from a sample $\mathcal{D}_{XY} = \{(x_1, y_1), \dots, (x_m, y_m)\}$ drawn *i.i.d.* from $P(X, Y)$ as in equation (10). Then the kernel Sum Rule has the following form:

Algorithm I: Kernel Sum Rule

From $\hat{\mu}_Y^\pi = \tilde{\Phi}\boldsymbol{\alpha}$ and $\hat{\mathcal{C}}_{X|Y} = \Upsilon(G + \lambda I)^{-1}\Phi$,

(a) $\hat{\mu}_X^\pi = \hat{\mathcal{C}}_{X|Y}\hat{\mu}_Y^\pi = \Upsilon(G + \lambda I)^{-1}\tilde{G}\boldsymbol{\alpha}$ (14)

(b) $\hat{\mathcal{C}}_{XX}^\pi = \Upsilon \text{diag} \left((G + \lambda I)^{-1}\tilde{G}\boldsymbol{\alpha} \right) \Upsilon^\top$ (15)

where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_{\tilde{m}})^\top$, and the implicit feature matrices are defined as $\Upsilon = (\phi(x_1), \dots, \phi(x_m))$, $\Phi = (\phi(y_1), \dots, \phi(y_m))$, and $\tilde{\Phi} = (\phi(\tilde{y}_1), \dots, \phi(\tilde{y}_{\tilde{m}}))$ respectively. G and \tilde{G} are two Gram matrices with entries $G_{ij} = k(y_i, y_j)$ and $\tilde{G}_{ij} = k(y_i, \tilde{y}_j)$ respectively. We will use these notation throughout the section.

4.2 Kernel Chain Rule

$Q(X, Y) = P(X|Y)\pi(Y)$ — we factorize the joint distribution into the product of a conditional distribution and a marginal distribution. Again note that $Q(X, Y)$ can be different from $P(X, Y)$, although they have the same conditional distribution $P(X|Y)$. We can factorize the joint embedding of $Q(X, Y)$, $\mathcal{C}_{XY}^\pi = \mathbb{E}_{XY}[\phi(X) \otimes \phi(Y)]$, as

$$\mathcal{C}_{XY}^\pi = \mathbb{E}_Y [\mathbb{E}_{X|Y}[\phi(X)] \otimes \phi(Y)] = \mathcal{C}_{X|Y} \mathbb{E}_Y[\phi(Y) \otimes \phi(Y)] = \mathcal{C}_{X|Y} \mathcal{C}_{YY}^\pi, \quad (16)$$

where \mathcal{C}_{YY}^π is the auto-covariance operator associated with $\pi(Y)$. It is understood that the last expression means that the composition of two operators results in \mathcal{C}_{XY}^π . This rule links the cross-covariance operator \mathcal{C}_{XY}^π with the auto-covariance operator \mathcal{C}_{YY}^π , again using the conditional embedding operator. Its difference from the kernel Sum Rule is that instead of inputting an embedding of $\pi(Y)$ using $\phi(Y)$ features, we feed an embedding using $\phi(Y) \otimes \phi(Y)$ features to the conditional embedding operator $\mathcal{C}_{X|Y}$. This results in an additional copy of variable Y in the cross covariance operator \mathcal{C}_{XY}^π , rather than the embedding μ_X^π with Y integrated out. For the discrete case with Kronecker delta kernel, the embedding version of the Chain Rule is

$$\underbrace{\begin{pmatrix} Q(x=s, y=t) \end{pmatrix}}_{\mathcal{C}_{XY}^\pi} = \underbrace{\begin{pmatrix} P(x=s|y=t) \end{pmatrix}}_{\mathcal{C}_{X|Y}} \underbrace{\begin{pmatrix} \pi(y=1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \pi(y=N) \end{pmatrix}}_{\mathcal{C}_{YY}^\pi} \quad (17)$$

As with the alternative expression for kernel Sum Rule in (13), we can obtain an alternative kernel Chain Rule using higher order features to form the conditional embedding operator $\mathcal{C}_{(XY)|Y}$ which results in $\mathcal{C}_{XY}^\pi = \mathcal{C}_{(XY)|Y} \mu_Y^\pi$.

In practice, we assume the embedding $\widehat{\mathcal{C}}_{YY}^\pi$ to be given as $\sum_{i=1}^{\tilde{m}} \alpha_i \phi(\tilde{y}_i) \otimes \phi(\tilde{y}_i)$. The empirical kernel Chain Rule may take either form (a) or (b) below, where the latter is obtained in the same

way as the kernel Sum Rule.

Algorithm II: Kernel Chain Rule

From $\widehat{\mu}_Y^\pi = \tilde{\Phi}\alpha$, $\widehat{\mathcal{C}}_{YY}^\pi = \tilde{\Phi} \text{diag}(\alpha)\tilde{\Phi}^\top$, and $\widehat{\mathcal{C}}_{X|Y} = \Upsilon(G + \lambda I)^{-1}\Phi$,

$$(a) \quad \widehat{\mathcal{C}}_{XY}^\pi = \widehat{\mathcal{C}}_{X|Y}\widehat{\mathcal{C}}_{YY}^\pi = \Upsilon(G + \lambda I)^{-1}\tilde{G} \text{diag}(\alpha)\tilde{\Phi}^\top \quad (18)$$

$$(b) \quad \widehat{\mathcal{C}}_{XY}^\pi = \widehat{\mathcal{C}}_{(XY)|Y}\widehat{\mu}_Y^\pi = \Upsilon \text{diag} \left((G + \lambda I)^{-1}\tilde{G}\alpha \right) \Phi^\top \quad (19)$$

The kernel Sum and Chain Rules form the basis for the kernel Bayes' Rule in the next section.

4.3 Kernel Bayes' Rule

$Q(Y|x) = \frac{P(x|Y)\pi(Y)}{Q(x)}$ where $Q(x) = \int_{\Omega} P(x|Y)d\pi(Y)$ — relates the posterior distribution $Q(Y|x)$ and the prior distribution $\pi(Y)$ via the likelihood function $P(x|Y)$. We will construct a conditional embedding operator $\mathcal{C}_{Y|X}^\pi$ and then obtain the embedding of $Q(Y|x)$ by $\mu_{Y|x}^\pi = \mathcal{C}_{Y|X}^\pi\phi(x)$ [11, 12]. The difference here is that the conditional embedding operator is modified by the prior distribution $\pi(Y)$. More specifically, the kernel Bayes' Rule is

$$\mu_{Y|x}^\pi = \mathcal{C}_{Y|X}^\pi\phi(x) = \mathcal{C}_{YX}^\pi (\mathcal{C}_{XX}^\pi)^{-1} \phi(x), \quad (20)$$

where the prior modified covariance operators are respectively

$$\text{from the Sum Rule (13):} \quad \mathcal{C}_{XX}^\pi := \mathcal{C}_{(XX)|Y}\mu_Y^\pi, \quad (21)$$

$$\text{from the Chain Rule (16):} \quad \mathcal{C}_{YX}^\pi := (\mathcal{C}_{X|Y}\mathcal{C}_{YY}^\pi)^\top, \quad (22)$$

given the embeddings μ_Y and \mathcal{C}_{YY} of the prior $\pi(Y)$ using features $\phi(y)$ and $\phi(y)\otimes\phi(y)$, respectively. Eq. (21) corresponds to the embedding for $Q(X) = \sum_{\Omega} P(X|Y)\pi(Y)$, while (22) corresponds to the embedding for $Q(X, Y) = P(X|Y)\pi(Y)$. We can also use the Chain Rule (b), but omit the details here.

To compute empirical estimates we assume the embedding of the prior $\widehat{\mu}_Y^\pi$ and $\widehat{\mathcal{C}}_{YY}^\pi$ to be given as $\sum_{i=1}^{\tilde{m}} \alpha_i\phi(\tilde{y}_i)$ and $\sum_{i=1}^{\tilde{m}} \alpha_i\phi(\tilde{y}_i)\otimes\phi(\tilde{y}_i)$, respectively, based on a weighted sample $\mathcal{D}_Y = \{\tilde{y}_1, \dots, \tilde{y}_{\tilde{m}}\}$.

Then using kernel Sum Rule and kernel Chain Rule, we obtain the kernel Bayes' Rule,

Algorithm III: Kernel Bayes' Rule

Let: $\Lambda := (G + \lambda I)^{-1} \tilde{G} \text{diag}(\boldsymbol{\alpha})$, $D = \text{diag}((G + \lambda I)^{-1} \tilde{G} \boldsymbol{\alpha})$

From kernel Sum Rule: $\hat{\mathcal{C}}_{XX}^\pi = \Upsilon D \Upsilon^\top$

From kernel Chain Rule: (a) $\hat{\mathcal{C}}_{YX}^\pi = \tilde{\Phi} \Lambda^\top \Upsilon^\top$, (b) $\hat{\mathcal{C}}_{YX}^\pi = \Phi D \Upsilon^\top$

$$(a) \hat{\mu}_{Y|x} = \hat{\mathcal{C}}_{YX}^\pi ((\hat{\mathcal{C}}_{XX}^\pi)^2 + \tilde{\lambda} I)^{-1} \hat{\mathcal{C}}_{XX}^\pi \phi(x) = \tilde{\Phi} \Lambda^\top ((DK)^2 + \tilde{\lambda} I)^{-1} KDK_{:x} \quad (23)$$

$$(b) \hat{\mu}_{Y|x} = \hat{\mathcal{C}}_{YX}^\pi ((\hat{\mathcal{C}}_{XX}^\pi)^2 + \tilde{\lambda} I)^{-1} \hat{\mathcal{C}}_{XX}^\pi \phi(x) = \Phi DK ((DK)^2 + \tilde{\lambda} I)^{-1} DK_{:x} \quad (24)$$

In the kernel Bayes' rule, there are two regularization parameters λ and $\tilde{\lambda}$: one is used in estimating the prior modified covariance operators, $\hat{\mathcal{C}}_{XX}^\pi$ and $\hat{\mathcal{C}}_{YX}^\pi$, and the other is used in estimating the final conditional embedding operator, $\hat{\mathcal{C}}_{Y|x}^\pi = \hat{\mathcal{C}}_{YX}^\pi ((\hat{\mathcal{C}}_{XX}^\pi)^2 + \tilde{\lambda} I)^{-1} \hat{\mathcal{C}}_{XX}^\pi$. Since the estimator D may not be positive definite, the latter regularization parameter $\tilde{\lambda}$ is used on the square of $\hat{\mathcal{C}}_{XX}^\pi$. We can see that the embedding of the posterior distribution has a form of $\sum_{i=1}^m \beta_i(x) \phi(\tilde{y}_i)$, where $\boldsymbol{\beta}(x) = \Lambda^\top ((DK)^2 + \tilde{\lambda} I)^{-1} KDK_{:x}$ or $\boldsymbol{\beta} = DK ((DK)^2 + \tilde{\lambda} I)^{-1} DK_{:x}$.

4.4 Kernel Bayesian Average and Posterior Decoding

The kernel embeddings directly estimated from data or obtained after applying kernel Bayes' rule are not direct estimates of a distribution or density, but represent distributions as expected feature mappings. In some cases, we may want to evaluate the expected value of a function $g \in \mathcal{F}$ with respect to the posterior distribution $Q(Y|x)$, or decode a particular value of y^* that is most typical of the posterior distribution $Q(Y|x)$, both using just the embedding $\mu_{Y|x}^\pi$.

For the former case, we can simply use the reproducing property and obtain $\mathbb{E}_{Y|x}[g(Y)] = \langle g, \mu_{Y|x}^\pi \rangle_{\mathcal{F}}$. If we assume the embedding $\hat{\mu}_{Y|x}^\pi$ be given as $\sum_{i=1}^{\tilde{m}} \beta_i(x) \phi(\tilde{y}_i)$, and $g = \sum_{i=1}^m \alpha_i \phi(y_i)$, then its kernel version is $\sum_{i=1}^m \sum_{j=1}^{\tilde{m}} \alpha_i \beta_j(x) k(y_i, \tilde{y}_j)$,

Algorithm IV: Kernel Bayes Average

$$\langle g, \hat{\mu}_{Y|x}^\pi \rangle = \boldsymbol{\beta}^\top \tilde{G} \boldsymbol{\alpha} \quad (25)$$

where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_{\tilde{m}})$ and $\boldsymbol{\beta} = (\beta_1(x), \dots, \beta_m(x))$. For the latter case, we can consider the point y^* whose feature vector $\phi(y^*)$ is the closest to the conditional embedding, *i.e.*, $y^* = \operatorname{argmin}_{y \in \Omega} \left\| \widehat{\mu}_{Y|x}^\pi - \phi(y) \right\|_{\mathcal{F}}^2$. The kernel version of the objective is $\sum_{i,j=1}^{\tilde{m}} \beta_i(x) \beta_j(x) k(\tilde{y}_i, \tilde{y}_j) - 2 \sum_{i=1}^{\tilde{m}} \beta_i(x) k(\tilde{y}_i, y) + k(y, y)$, *i.e.*,

Algorithm V: Kernel Bayes Posterior Decoding

$$y^* = \operatorname{argmin}_{y \in \mathcal{Y}} -2\boldsymbol{\beta}^\top \tilde{G}_{:y} + k(y, y) \quad (26)$$

This optimization will give us a point estimator of Y . In general, the above optimization problem is difficult to solve, and it corresponds to the so-called pre-image problem in kernel methods. For particular choices of kernels, such as Gaussian RBF kernels $k(y, y') = \exp(-\sigma \|y - y'\|^2)$, this nonlinear optimization can be solved by a fixed-point iteration method [25] which is used in the filtering problem in Section 5.2.

5 Applications

Having now presented the necessary machinery for manipulating conditional embeddings, we turn our attention to applications. We will focus on two applications: a kernel belief propagation algorithm for inference in nonparametric graphical models; and a nonparametric algorithm for learning models and performing probabilistic inference in hidden Markov models. The former is a direct application of conditional embedding and the kernel Sum Rule, while the latter involves more advanced application of the kernel Bayes' Rule. We will demonstrate that using kernel embeddings of conditional distributions, we can reduce various nonparametric inference problems to conditional embeddings and linear algebraic operations with the basic kernel rules developed in Section 4.

5.1 Kernel Belief Propagation

The belief propagation algorithm, a key inference algorithm in graphical models, can be expressed as a repeated evaluation of conditional expectations [5, 7]. By representing messages in belief propagation as RKHS functions, we can apply conditional embeddings directly to efficiently compute

message updates. This observation allows us to develop a nonparametric belief propagation algorithm which achieves better accuracy in the problem of predicting depth from image features. See Figure 1(a) for an illustration.

5.1.1 Problem Formulation

We begin with a short introduction to pairwise Markov random fields (MRFs) and the belief propagation algorithm. A pairwise Markov random field (MRF) is defined on an undirected graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V} := \{1, \dots, n\}$ connected by edges in \mathcal{E} . Each node $s \in \mathcal{V}$ is associated with a random variable X_s on the domain \mathcal{X} (we assume a common domain for ease of notation, but in practice the domains can be different), and $\Gamma_s := \{t | (s, t) \in \mathcal{E}\}$ is the set of neighbors of node s with size $d_s := |\Gamma_s|$. In a pairwise MRF, the joint density of the variables $\mathbf{X} := \{X_1, \dots, X_{|\mathcal{V}|}\}$ is assumed to factorize according to a model $p(\mathbf{X}) = \frac{1}{Z} \prod_{(s,t) \in \mathcal{E}} \Psi_{st}(X_s, X_t) \prod_{s \in \mathcal{V}} \Psi_s(X_s)$, where $\Psi_s(X_s)$ and $\Psi_{st}(X_s, X_t)$ are node and edge potentials respectively, and Z is the partition function that normalizes the distribution.

The inference problem in an MRF is defined as calculating the marginal densities $p(X_s)$ for nodes $s \in \mathcal{V}$. Belief Propagation (BP) is an iterative algorithm for performing approximate inference in MRFs [1]. BP represents intermediate results of marginalization steps as messages passed between adjacent nodes: a message m_{ts} from t to s is calculated based on messages m_{ut} from all neighboring nodes u of t besides s , *i.e.*, [7]

$$m_{ts}(x_s) = \int_{\Omega} p^*(X_t | x_s) \prod_{u \in \Gamma_t \setminus s} m_{ut}(X_t) dX_t = \mathbb{E}_{X_t | x_s} \left[\prod_{u \in \Gamma_t \setminus s} m_{ut}(X_t) \right]. \quad (27)$$

Note that we use $\prod_{u \in \Gamma_t \setminus s}$ to denote $\prod_{u \in \Gamma_t \setminus s}$, where it is understood that the indexes range over all neighbors u of t except s . This notation also applies to operations other than the product. Furthermore, $p^*(X_t | x_s)$ denotes the true conditional distribution of X_t . Using similar reasoning, the node belief, an estimate of $p(X_s)$, takes the form $B(X_s) \propto p^*(X_s) \prod_{t \in \Gamma_s} m_{ts}^*(X_s)$ on convergence of BP, *i.e.*, the message converges to $m_{ts}^*(X_s)$. Our goal is to develop a nonparametric belief propagation algorithm, where the potentials are nonparametric functions learned from data, such that multi-modal and other non-Gaussian statistical features can be captured. Most crucially, these

potentials must be represented in such a way that the message update in (27) is computationally tractable.

5.1.2 Kernel Algorithm

We will assume that a message $m_{ut}(x_t)$ is in the reproducing kernel Hilbert space (RKHS), *i.e.*, $m_{ut}(x_t) = \langle m_{ut}, \phi(x_t) \rangle$. In an evidence node X_u where the evidence is fixed to x_u , its outgoing message is simply the likelihood function $p(x_u|x_t)$, and we can estimate it as a RKHS function. That is given m samples, $m_{ut}(\cdot) = \hat{p}(x_u|\cdot) = \sum_{i=1}^m \beta_{ut}^i k(x_t^i, \cdot)$ [5]. As we will see, the advantage of this assumption is that the update procedure using kernel conditional embedding can be expressed as a linear operation in the RKHS, and results in new messages that are likewise RKHS functions.

We begin by defining a tensor product reproducing kernel Hilbert space $\mathcal{H} := \otimes^{d_t-1} \mathcal{F}$, under which the product of incoming messages can be written as a single inner product. For a node t with degree $d_t = |\Gamma_t|$, the product of incoming messages m_{ut} from all neighbors except s becomes an inner product in \mathcal{H} , $\prod_{u \setminus s} m_{ut}(X_t) = \prod_{u \setminus s} \langle m_{ut}, \phi(X_t) \rangle_{\mathcal{F}} = \left\langle \bigotimes_{u \setminus s} m_{ut}, \xi(X_t) \right\rangle_{\mathcal{H}}$, where $\xi(X_t) := \bigotimes_{u \setminus s} \phi(X_t)$. The message update (27) becomes $m_{ts}(x_s) = \left\langle \bigotimes_{u \setminus s} m_{ut}, \mathbb{E}_{X_t|x_s} [\xi(X_t)] \right\rangle_{\mathcal{H}}$. We can define the conditional embedding operator for the tensor product of features, such that $\mathcal{C}_{X_t^\otimes|X_s} : \mathcal{F} \rightarrow \mathcal{H}$ satisfies

$$\mu_{X_t^\otimes|x_s} := \mathbb{E}_{X_t|x_s} [\xi(X_t)] = \mathcal{C}_{X_t^\otimes|X_s} \phi(x_s). \quad (28)$$

As in the single variable case, $\mathcal{C}_{X_t^\otimes|X_s}$ is defined in terms of a covariance operator $\mathcal{C}_{X_t^\otimes X_s} := \mathbb{E}_{X_t X_s} [\xi(X_t) \otimes \phi(X_s)]$ in the tensor space, and the operator $\mathcal{C}_{X_s X_s}$. The operator $\mathcal{C}_{X_t^\otimes|X_s}$ takes the feature map $\phi(x_s)$ of the point on which we condition, and outputs the conditional expectation of the tensor product feature $\xi(X_t)$. Consequently, we can express the message update as a linear operation, but in a tensor product feature space,

$$m_{ts}(x_s) = \left\langle \bigotimes_{u \setminus s} m_{ut}, \mathcal{C}_{X_t^\otimes|X_s} \phi(x_s) \right\rangle_{\mathcal{H}}. \quad (29)$$

The belief at a specific node s can be computed as $B(X_s) = p^*(X_s) \prod_{u \in \Gamma_s} m_{us}(X_s)$ where the true

marginal $p(X_r)$ can be estimated using a kernel density estimator.

Now given m samples, and applying the kernel estimator for conditional embedding operator in (10), we can obtain a kernel expression for the message update in (29). Let the incoming message be $m_{ut}(\cdot) = \sum_{i=1}^m \beta_{ut}^i k(x_t^i, \cdot)$, then

Algorithm VI: Kernel Belief Propagation Update (KBP)

$$\hat{m}_{ts}(x_s) = \left(\bigodot_{u \setminus s} K_t \beta_{ut} \right)^\top (K_s + \lambda I)^{-1} \Upsilon^\top \phi(x_s) \quad (30)$$

where K_t and K_s are the Gram matrices for the samples from variable X_t and X_s respectively, $\Upsilon = (\phi(x_s^1), \dots, \phi(x_s^m))$ is the feature matrix, and \bigodot is the element-wise vector product. If we define $\beta_{ts} = (K_s + \lambda I)^{-1} (\bigodot_{u \setminus s} K_t \beta_{ut})$, then the outgoing message can be expressed as $\hat{m}_{ts} = \Upsilon \beta_{ts}$. In other words, given incoming messages expressed as linear combinations of feature mapped training samples from X_t , the outgoing message will likewise be a weighted linear combination of feature mapped training samples from X_s . Importantly, only m mapped points will be used to express the outgoing message, regardless of the number of incoming messages or the number of points used to express each incoming message. Thus the complexity of message representation does not increase with BP iterations or node degree (See Figure 1(a) for an illustration). We call this kernel message update algorithm *kernel belief propagation* (KBP), and we have applied it to the problem of predicting depth from 2D images.

5.1.3 Experimental Results

We compare with two state-of-the-art approaches to nonparametric belief propagation: Gaussian Mixture BP [2] and Particle BP [26]. Gaussian mixture BP assumes incoming messages to be a mixture of b Gaussians. The product of d_t incoming messages to node t then contains b^{d_t} Gaussians. This exponential blow-up is avoided by replacing the exact update with an approximation. Particle BP represents the incoming messages using a common set of particles. These particles must be re-drawn via Metropolis-Hastings at each node and BP iteration, which is costly (although in practice, it is sufficient to re-sample periodically, rather than strictly at every iteration) [27]. By contrast, KBP updates are simply matrix-vector products and can potentially be much faster.

The prediction of 3D depth information from 2D image features is a difficult inference problem, as the depth may be ambiguous: similar features can occur at different depths. This creates a multi-modal depth distribution given the image feature. Furthermore, the marginal distribution of the depth can itself be multi-modal, which makes the Gaussian approximation a poor choice (see Figure 1(a)). To make a spatially consistent prediction of the depth map, we formulated the problem as an undirected graphical model, where a depth variable $y_i \in \mathbb{R}$ was associated with each patch of an image, and these variables were connected according to a 2D grid topology. Each hidden depth variable was linked to an image feature variable $x_i \in \mathbb{R}^{273}$ for the corresponding patch. This formulation resulted in a graphical model with $9,202 = 107 \times 86$ continuous depth variables, and a maximum node degree of 5. Due to the way the images were taken (upright), we used a template model where horizontal edges in a row shared the same potential, vertical edges at the same height shared the same potential, and patches at the same row shared the same likelihood function. Both the edge potentials between adjacent depth variables and the likelihood function between image feature and depth were unknown, and were learned from data.

We used a set of 274 images taken on the Stanford campus, including both indoor and outdoor scenes [28]. Images were divided into patches of size 107 by 86, with the corresponding depth map for each patch obtained using 3D laser scanners (*e.g.*, Figure 7(a)). Each patch was represented by a 273 dimensional feature vector, which contained both local features (such as color and texture) and relative features (features from adjacent patches). We took the logarithm of the depth map and performed learning and prediction in this space. The entire dataset contained more than 2 million data points ($107 \times 86 \times 274$). We applied a Gaussian RBF kernel on the depth information, with the bandwidth parameter set to the median distance between training depths. We used a linear kernel for the image features.

Our results were obtained by leave-one-out cross validation. For each test image, we ran discrete, Gaussian mixture, particle, and kernel BP for 10 BP iterations. The average prediction error (MAE: mean absolute error) and runtime are shown in Figures 7(a) and (b). Kernel BP produces the lowest error (MAE=0.145) by a significant margin, while having a similar runtime to discrete BP. Gaussian mixture and particle BP achieve better MAE than discrete BP, but their runtimes are two order of

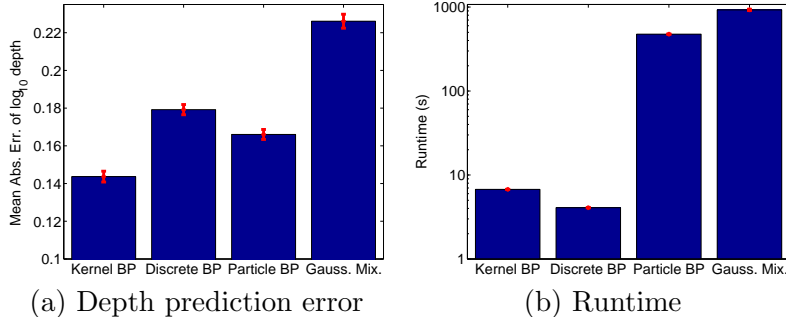


Figure 7: Average depth prediction error and runtime of kernel BP compared to discrete, Gaussian mixture and particle BP over 274 images. Runtimes are on a logarithmic scale.

magnitude slower. We note that the error of kernel BP is slightly better than the results of point-wise MRF reported in [28]. We also note that our method improves over the previous methods by using new nonparametric learning and inference algorithms but with the same image features. We expect that by considering more image features, *e.g.*, those related to image slant and tilt, further improvement is possible.

5.2 Nonparametric Hidden Markov Models

We discuss the hidden Markov model of length T , which is a joint distribution $P(S^1, \dots, S^T, O^1, \dots, O^T) = P(O^1|S^1) \prod_t P(O^t|S^t) P(S^t|S^{t-1})$ with hidden states S^t , and their corresponding observations O^t , where $P(S^t|S^{t-1})$ is the *transition model* and $P(O^t|S^t)$ is the *observation model*. In the presence of multi-modal and non-Gaussian distributions, a *kernel Bayes filtering* algorithm (KBF) is developed based on conditional embedding operator and kernel Bayes' Rule described in Section 4 [11, 12]. The main difference between KBF and the kernel BP algorithm discussed in the last section is that the likelihood function is not explicitly given in KBF, but represented as an estimate of the covariance operator. The two algorithms complement each other: if the local densities are available, and/or the graphical models contain loops, then the kernel BP approach is preferred; if the graphical models are simple and one does not want to perform density estimation, then KBF is a better choice. We will apply this nonparametric hidden Markov model to a problem of estimating camera orientation from video. See Figure 1(b) for an illustration.

5.2.1 Problem Formulation

In the framework of conditional embedding, we need to express $P(S^t|S^{t-1})$ and $P(O^t|S^t)$ with samples. We thus assume that a training sample $(\tilde{s}^1, \dots, \tilde{s}^T, \tilde{o}^1, \dots, \tilde{o}^T)$ is available in the *training phase*. In the *testing phase*, given a new sample (o^1, \dots, o^T) , we infer the hidden states (S^1, \dots, S^T) . Note that in the training phase we assume that the sample on the hidden states is available. While this is a restriction of the method, there are many situations where this assumption is satisfied; if the measurement of hidden states is very expensive, we wish to estimate them from observations based on a small number of training samples; when the hidden state can be observed with a time delay, the real time estimation of the current hidden state may be necessary based on previous data.

We focus on *filtering*, in which one queries the model for the posterior at some time step conditioned on all past observations, though other operations such as smoothing and prediction are also possible. Denote the history of the dynamical system as $h^t := (o^1, \dots, o^t)$. In filtering, one recursively maintains a belief state, $P(S^{t+1}|h^{t+1})$, in two steps: a *prediction* step and a *conditioning* step. The first updates the distribution by multiplying by the transition model and marginalizing out the previous time step: $P(S^{t+1}|h^t) = \mathbb{E}_{S^t|h^t}[P(S^{t+1}|S^t)]$. The second conditions the distribution on a new observation O^{t+1} using Bayes' rule: $P(S^{t+1}|h^t o^{t+1}) \propto P(o^{t+1}|S^{t+1})P(S^{t+1}|h^t)$. The derivation of the method is analogous to the sequential update rule for the standard linear Gaussian models or hidden Markov model with discrete domains. The kernel Sum Rule and Bayes' Rule in Section 4 reduce the Bayesian computation into linear algebraic operations on Gram matrices.

5.2.2 Kernel Algorithm

The prediction and conditioning steps can be reformulated with respect to the kernel embeddings. By maintaining the belief state recursively, we can assume that $\mu_{S^t|h^t}$ is known. Applying the kernel Sum Rule to $P(S^{t+1}|h^t) = \mathbb{E}_{S^t|h^t}[P(S^{t+1}|S^t)]$, we have

$$\mu_{S^{t+1}|h^t} = \mathcal{C}_{S^{t+1}|S^t} \mu_{S^t|h^t}, \quad \text{and} \quad \mathcal{C}_{(S^{t+1}S^{t+1})|h^t} = \mathcal{C}_{(S^{t+1}S^{t+1})|S^t} \mu_{S^t|h^t}. \quad (31)$$

To obtain the embedding of the posterior distribution, $\mu_{S^{t+1}|h^t o^{t+1}}$, we can apply kernel Bayes' Rule with the prior $P(S^{t+1}|h^t)$, whose embedding $\mu_{S^{t+1}|h^t}$ is supplied by the prediction step in the recursion, and the likelihood part $P(o^{t+1}|S^{t+1})$. Here the goal is to obtain the posterior embedding $\mu_{S^{t+1}|h^t o^{t+1}}$ by conditioning further on variable o^{t+1} , and we accomplish this iterated conditioning using kernel Bayes' Rule. First, we compute the prior modified covariance operators

$$\text{from kernel Chain Rule (b): } \mathcal{C}_{S^{t+1}O^{t+1}}^\pi = \mathcal{C}_{(S^{t+1}O^{t+1})S^{t+1}} \mathcal{C}_{S^{t+1}S^{t+1}}^{-1} \mu_{S^{t+1}|h^t} \quad (32)$$

$$\text{from kernel Sum Rule (b): } \mathcal{C}_{O^{t+1}O^{t+1}}^\pi = \mathcal{C}_{(O^{t+1}O^{t+1})S^{t+1}} \mathcal{C}_{S^{t+1}S^{t+1}}^{-1} \mu_{S^{t+1}|h^t}. \quad (33)$$

Then the embedding of the posterior belief can be computed as

$$\mu_{S^{t+1}|h^t o^{t+1}} = \mathcal{C}_{S^{t+1}O^{t+1}}^\pi (\mathcal{C}_{O^{t+1}O^{t+1}}^\pi)^{-1} \phi(o^{t+1}). \quad (34)$$

In practice, we need to express the above formula with Gram matrices based on the training sample $(\tilde{s}^1, \dots, \tilde{s}^T, \tilde{o}^1, \dots, \tilde{o}^T)$. Let $\sum_{i=1}^T \alpha_i^t \phi(\tilde{s}^i)$ be the estimator of $\mu_{S^t|h^t}$. Using kernel Bayes' Rule, we obtain the kernel filtering algorithm as a recursive update of weighting α ,

Algorithm VII: Kernel Bayes Filtering (KBF)

$$D^{t+1} = \text{diag}((G + \lambda I)^{-1} \tilde{G} \alpha^t), \quad (35)$$

$$\alpha^{t+1} = D^{t+1} K ((D^{t+1} K)^2 + \tilde{\lambda} I)^{-1} D^{t+1} K_{:o^{t+1}}, \quad (36)$$

where G and K are the Gram matrix of $(\tilde{s}^1, \dots, \tilde{s}^T)$ and $(\tilde{o}^1, \dots, \tilde{o}^T)$ respectively, and \tilde{G} is the “transfer” Gram matrix defined by $\tilde{G}_{ij} = k(\tilde{s}_i, \tilde{s}_{j+1})$, and $K_{:o^{t+1}} = (k(\tilde{o}^1, o^{t+1}), \dots, k(\tilde{o}^T, o^{t+1}))^\top$. See [12] for more details, and Figure 1(b) for an illustration of the update of weighting α .

5.2.3 Experimental Results

We compare the KBF method with extended Kalman filter (EKF) and unscented Kalman filter (UKF) [29] on synthetic data and real-world data. Under the assumption that a training sample is available, cross-validation can be performed on the training sample to select the kernel parameters and regularization parameters for KBF [12]. In the experiments below, we adopt a validation

approach by dividing the training sample in two.

We first applied the KBF algorithm to a simple nonlinear dynamical system, in which the degree of nonlinearity can be controlled. The dynamics of the hidden state $S_t = (u_t, v_t)^\top \in \mathbb{R}^2$, are given by (a) rotation in a circle: $(u_{t+1}, v_{t+1}) = (\cos \theta_{t+1}, \sin \theta_{t+1})$, $\theta_{t+1} = \theta_t + 0.3$ with $\theta_t = \arctan(|v_t|/|u_t|)$, and (b) oscillatory rotation: $(u_{t+1}, v_{t+1}) = (1 + 0.4 \sin(8\theta_{t+1}))(\cos \theta_{t+1}, \sin \theta_{t+1})$, $\theta_{t+1} = \theta_t + 0.4$. The observation o^{t+1} includes noise: $o^{t+1} = (u_{t+1}, v_{t+1})^\top + \xi$ with $\xi \sim N(0, 0.08I_2)$ in both (a) and (b). Note that the dynamics of (u_t, v_t) are nonlinear even in (a).

We assume the correct models are known to the EKF and UKF. The results are shown in Figure 8. In both the cases, EKF and UKF show indistinguishably small difference. The dynamics in (a) are weakly nonlinear, and KBF has slightly worse MSE than EKF and UKF. For dataset (b), which has strong nonlinearity, KBF outperforms the nonlinear Kalman filters for $T \geq 200$.

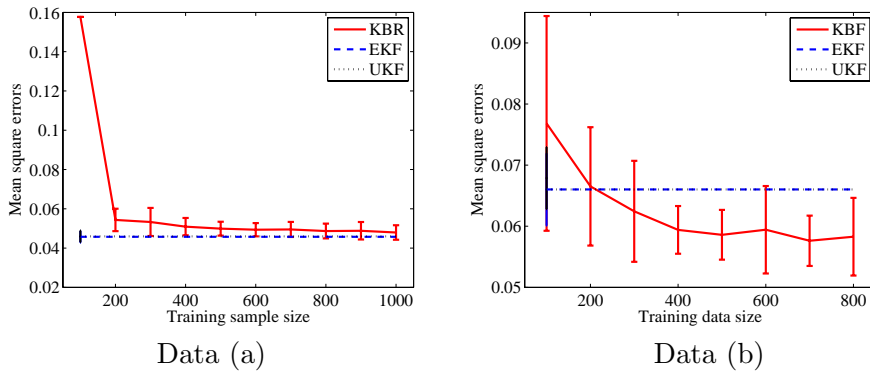


Figure 8: Comparisons with the KBF and nonlinear KF. (Average MSEs and standard errors over 30 runs.)

In our second example, we applied the KBF to the camera rotation problem used in Song et al. [4]. The angle of a camera, which is located at a fixed position, is a hidden variable, and movie frames recorded by the camera are observed. The data are generated virtually using a computer graphics environment (See Figure 1(b)). We are given 3600 down-sampled frames of 20×20 RGB pixels ($o_t \in [0, 1]^{1200}$), where the first 1800 frames are used for training, and the second half are used to test the filter. We make the data noisy by adding Gaussian noise $N(0, \sigma^2)$ to o_t .

Our experiments cover two settings: (a) $S_t \in \mathbb{R}^9$ without knowing $S_t \in SO(3)$, and (b) we exploit the fact that $S_t \in SO(3)$. In (a), we use the Kalman filter by estimating the relations under

a linear assumption, and the KBF with Gaussian kernels for S_t and O_t as Euclidean vectors. In (b), for the Kalman Filter, S_t is represented by a quaternion, which is a standard vector representation of rotations; for the KBF the kernel $k(A, B) = \text{Tr}[AB^T]$ is used for S_t , and S_t is estimated within $SO(3)$. Table 1 shows the Frobenius norms between the estimated matrix and the true one. The KBF significantly outperforms the Kalman filter, since KBF have the advantage in extracting the complex nonlinear dependence between the observations and the hidden states.

Table 1: Average MSE and standard errors of estimating camera angles (10 runs).

	(a) \mathbb{R}^9		(b) $SO(3)$	
	KBR (Gauss)	Kalman (\mathbb{R}^9)	KBR (Tr)	Kalman (Quat.)
$\sigma^2 = 10^{-4}$	0.210 ± 0.015	1.980 ± 0.083	0.146 ± 0.003	0.557 ± 0.023
$\sigma^2 = 10^{-3}$	0.222 ± 0.009	1.935 ± 0.064	0.210 ± 0.008	0.541 ± 0.022

6 Conclusion

In this paper, we introduced kernel embeddings of conditional distributions, a new framework for dealing with nonparametric problems having rich conditional independence structure. This new framework often leads to more accurate and simpler algorithms in many applications. It also plays an important role in bringing together traditionally separate areas in machine learning research, including kernel methods, probabilistic graphical models and tensor data analysis. We believe that this new way of combining kernel methods and graphical models can potentially solve many other difficult problems, and generalize kernel methods to domains with complex dependence structures and interactions. It is a useful step towards the ultimate, long term goal of understanding and exploiting dependency structures and rich diverse statistical features prevalent in many modern applications, such as computer vision, computational biology, social sciences and music research.

References

- [1] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, 1988.

- [2] E. Sudderth, A. Ihler, W. Freeman, and A. Willsky. Nonparametric belief propagation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2003.
- [3] B. W. Silverman. *Density Estimation for Statistical and Data Analysis*. Monographs on statistics and applied probability. Chapman and Hall, London, 1986.
- [4] L. Song, J. Huang, A. J. Smola, and K. Fukumizu. Hilbert space embeddings of conditional distributions. In *Proceedings of the International Conference on Machine Learning*, 2009.
- [5] L. Song, A. Gretton, and C. Guestrin. Nonparametric tree graphical models. In *Proc. Intl. Conference on Artificial Intelligence and Statistics*, 2010.
- [6] L. Song, B. Boots, S. Siddiqi, G. Gordon, and A. J. Smola. Hilbert space embeddings of hidden markov models. In *International Conference on Machine Learning*, 2010.
- [7] L. Song, A. Gretton, D. Bickson, Y. Low, and C. Guestrin. Kernel belief propagation. In *Proc. Intl. Conference on Artificial Intelligence and Statistics*, 2011.
- [8] L. Song, A. Parikh, and E.P. Xing. Kernel embeddings of latent tree graphical models. In *Advances in Neural Information Processing Systems*, volume 25, 2011.
- [9] S. Grunewalder, G. Lever, L. Baldassarre, M. Pontil, and A. Gretton. Modeling transition dynamics in MDPs with RKHS embeddings. In *Proceedings of the International Conference on Machine Learning*, 2012.
- [10] Y. Nishiyama, A. Boularias, A. Gretton, and K. Fukumizu. Hilbert space embeddings of POMDPs. In *Conference on Uncertainty in Artificial Intelligence*, 2012.
- [11] K. Fukumizu, L. Song, and A. Gretton. Kernel Bayes' rule. In *Neural Information Processing Systems*, 2011.
- [12] K. Fukumizu, L. Song, and A. Gretton. Kernel Bayes' rule: Bayesian inference with positive definite kernels. In *accepted to JMLR*, 2012.
- [13] B. Schölkopf, K. Tsuda, and J.-P. Vert. *Kernel Methods in Computational Biology*. MIT Press, Cambridge, MA, 2004.

- [14] K. Fukumizu, F. R. Bach, and M. I. Jordan. Dimensionality reduction for supervised learning with reproducing kernel Hilbert spaces. *Journal of Machine Learning Research*, 5:73–99, 2004.
- [15] A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer, 2004.
- [16] A. J. Smola, A. Gretton, L. Song, and B. Schölkopf. A Hilbert space embedding for distributions. In *Proceedings of the International Conference on Algorithmic Learning Theory*, volume 4754, pages 13–31. Springer, 2007.
- [17] B. Sriperumbudur, A. Gretton, K. Fukumizu, G. Lanckriet, and B. Schölkopf. Injective Hilbert space embeddings of probability measures. In *Proc. Annual Conf. Computational Learning Theory*, pages 111–122, 2008.
- [18] B. Sriperumbudur, A. Gretton, K. Fukumizu, G. Lanckriet, and B. Schölkopf. Hilbert space embeddings and metrics on probability measures. *Journal of Machine Learning Research*, 11:1517–1561, 2010.
- [19] A. Gretton, K. Borgwardt, M. Rasch, B. Schoelkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.
- [20] A. Gretton, K. Fukumizu, C.-H. Teo, L. Song, B. Schölkopf, and A. J. Smola. A kernel statistical test of independence. In *Advances in Neural Information Processing Systems 20*, pages 585–592, Cambridge, MA, 2008. MIT Press.
- [21] A. Gretton, B. Sriperumbudur, D. Sejdinovic, S. Balakrishnan, M. Pontil, K. Fukumizu, et al. Optimal kernel choice for large-scale two-sample tests. In *Advances in Neural Information Processing Systems 25*, pages 1214–1222, 2012.
- [22] S. K. Zhou and R. Chellappa. From sample similarity to ensemble similarity: Probabilistic distance measures in reproducing kernel Hilbert space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):917–929, 2006.

- [23] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.
- [24] S. Grunewalder, G. Lever, L. Baldassarre, S. Patterson, A. Gretton, and M. Pontil. Conditional mean embeddings as regressors. In *Proceedings of the International Conference on Machine Learning*, 2012.
- [25] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, Matthias Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 536–542. MIT Press, 1999.
- [26] A. Ihler and D. McAllester. Particle belief propagation. In *Proc. Intl. Conference on Artificial Intelligence and Statistics*, pages 256–263, 2009.
- [27] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50:5–43, 2003.
- [28] A. Saxena, M. Sun, and A.Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):824–840, 2009.
- [29] S.J. Julier and J.K. Uhlmann. New extension of the kalman filter to nonlinear systems. In *AeroSense’97*, pages 182–193. International Society for Optics and Photonics, 1997.