Journal of Machine Learning Research 5 (2004) 819–844 Submitted 1/04; Published 7/04

Probability Product Kernels

Tony Jebara JEBARA@CS.COLUMBIA.EDU
Risi Kondor RISI@CS.COLUMBIA.EDU
Andrew Howard AHOWARD@CS.COLUMBIA.EDU
Computer Science Department
Columbia University
New York, NY 10027, USA

Editors: Kristin Bennett and Nicolò Cesa-Bianchi

Abstract

The advantages of discriminative learning algorithms and kernel machines are combined with generative modeling using a novel kernel between distributions. In the probability product kernel, data points in the input space are mapped to distributions over the sample space and a general inner product is then evaluated as the integral of the product of pairs of distributions. The kernel is straightforward to evaluate for all exponential family models such as multinomials and Gaussians and yields interesting nonlinear kernels. Furthermore, the kernel is computable in closed form for latent distributions such as mixture models, hidden Markov models and linear dynamical systems. For intractable models, such as switching linear dynamical systems, structured mean-field approximations can be brought to bear on the kernel evaluation. For general distributions, even if an analytic expression for the kernel is not feasible, we show a straightforward sampling method to evaluate it. Thus, the kernel permits discriminative learning methods, including support vector machines, to exploit the properties, metrics and invariances of the generative models we infer from each datum. Experiments are shown using multinomial models for text, hidden Markov models for biological data sets and linear dynamical systems for time series data.

Keywords: Kernels, support vector machines, generative models, Hellinger divergence, Kullback-Leibler divergence, Bhattacharyya affinity, expected likelihood, exponential family, graphical models, latent models, hidden Markov models, dynamical systems, mean field

1. Introduction

Recent developments in machine learning, including the emergence of support vector machines, have rekindled interest in kernel methods (Vapnik, 1998; Hastie et al., 2001). Typically, kernels are designed and used by the practitioner to introduce useful nonlinearities, embed prior knowledge and handle unusual input spaces in a learning algorithm (Schölkopf and Smola, 2001). In this article, we consider kernels between distributions. Typically, kernels compute a generalized inner product between two input objects χ and χ' which is equivalent to apply a mapping function Φ to each object and then computing a dot product between $\Phi(\chi)$ and $\Phi(\chi')$ in a Hilbert space. We will instead consider the case where the mapping $\Phi(\chi)$ is a probability distribution $p(x|\chi)$, thus restricting the Hilbert space since the space of distributions is trivially embedded in the Hilbert space of functions. However, this restriction to positive normalized mappings is in fact not too limiting since general $\Phi(\chi)$ mappings and the nonlinear decision boundaries they generate can often be mimicked by a probabilistic mapping. However, a probabilistic mapping facilitates and elucidates kernel design in general. It permits kernel design to leverage the large body of tools in statistical and generative modeling including Bayesian networks (Pearl, 1997). For instance, maximum likelihood or Bayesian estimates may be used to set certain aspects of the mapping to Hilbert space.

In this paper, we consider the mapping from datum χ to a probability distribution $p(x|\chi)$ which has a straightforward inner product kernel $k(\chi, \chi') = \int p^\rho(x|\chi)p^\rho(x|\chi')dx$ (for a scalar ρ which we typically set to 1 or $1/2$). This kernel is merely an inner product between two distributions and, if these distributions are known directly, it corresponds to a linear classifier in the space of probability distributions. If the distributions themselves are not available and only observed data is given, we propose using either Bayesian or Frequentist estimators to map a single input datum (or multiple inputs) into probability distributions and subsequently compute the kernel. In other words, we map points as $\chi \rightarrow p(x|\chi)$ and $\chi' \rightarrow p(x|\chi')$. For the setting of $\rho = 1/2$ the kernel is none other than the classical Bhattacharyya similarity measure (Kondor and Jebara, 2003), the affinity corresponding to Hellinger divergence (Jebara and Kondor, 2003).[1]

One goal of this article is to explore a contact point between discriminative learning (support vector machines and kernels) and generative learning (distributions and graphical models). Discriminative learning directly optimizes performance for a given classification or regression task. Meanwhile, generative learning provides a rich palette of tools for exploring models, accommodating unusual input spaces and handling priors. One approach to marrying the two is to estimate parameters in generative models with discriminative learning algorithms and optimize performance on a particular task. Examples of such approaches include conditional learning (Bengio and Frasconi, 1996) or large margin generative modeling (Jaakkola et al., 1999). Another approach is to use kernels to integrate the generative models within a discriminative learning paradigm. The proposed probability product kernel falls into this latter category. Previous efforts to build kernels that accommodate probability distributions include the Fisher kernel (Jaakkola and Haussler, 1998), the heat kernel (Lafferty and Lebanon, 2002) and kernels arising from exponentiating Kullback-Leibler divergences (Moreno et al., 2004). We discuss, compare and contrast these approaches to the probability product kernel in Section 7. One compelling feature of the new kernel is that it is straightforward and efficient to compute over a wide range of distributions and generative models while still producing interesting nonlinear behavior in, for example, support vector machine (SVM) classifiers. The generative models we consider include Gaussians, multinomials, the exponential family, mixture models, hidden Markov models, a wide range of graphical models and (via sampling and mean-field methods) intractable graphical models. The flexibility in choosing a distribution from the wide range of generative models in the field permits the kernel to readily accommodate many interesting input spaces including sequences, counts, graphs, fields and so forth. It also inherits the properties, priors and invariances that may be straightforward to design within a generative modeling paradigm and propagates them into the kernel learning machine. This article thus gives a generative modeling route to kernel design to complement the family of other kernel engineering methods including super-kernels (Ong et al., 2002), convolutional kernels (Haussler, 1999; Collins and Duffy, 2002), alignment kernels (Watkins, 2000), rational kernels (Cortes et al., 2002) and string kernels (Leslie et al., 2002; Vishawanathan and Smola, 2002).

This paper is organized as follows. In Section 2, we introduce the probability product kernel and point out connections to other probabilistic kernels such as Fisher kernels and exponentiated Kullback-Leibler divergences. Estimation methods for mapping points probabilistically to Hilbert

1. This has interesting connections to Kullback-Leibler divergence (Topsoe, 1999).

space are outlined. In Section 3 we elaborate the kernel for the exponential family and show its specific form for classical distributions such as the Gaussian and multinomial. Interestingly, the kernel can be computed for mixture models and graphical models in general and this is elaborated in Section 4. Intractable graphical models are then handled via structured mean-field approximations in Section 6. Alternatively, we describe sampling methods for approximately computing the kernel. Section 7 discusses the differences and similarities between our probability product kernels and previously presented probabilistic kernels. We then show experiments with text data sets and multinomial kernels, with biological sequence data sets using hidden Markov model kernels, and with continuous time series data sets using linear dynamical system kernels. The article then concludes with a brief discussion.

2. A Kernel Between Distributions

Given a positive (semi-)definite kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ on the input space $\mathcal{X}$ and examples $\chi_1, \chi_2, \ldots, \chi_m \in \mathcal{X}$ with corresponding labels $y_1, y_2, \ldots, y_m$, kernel based learning algorithms return a hypothesis of the form $h(\chi) = \sum_{i=1}^{m} \alpha_i k(\chi_i, \chi) + b$. The role of the kernel is to capture our prior assumptions of similarity in $\mathcal{X}$. When $k(\chi, \chi')$ is large, we expect that the corresponding labels be similar or the same.

Classically, the inputs $\{\chi_i\}_{i=1}^{m}$ are often represented as vectors in $\mathbb{R}^n$, and k is chosen to be one of a small number of popular positive definite functions on $\mathbb{R}^n$, such as the Gaussian RBF

$$k(\chi, \chi') = e^{-\| \chi - \chi' \|^2 / (2\sigma^2)}. \tag{1}$$

Once we have found an appropriate kernel, the problem becomes accessible to a whole array of non-parametric discriminative kernel based learning algorithms, such as support vector machines, Gaussian processes, etc. (Schölkopf and Smola, 2002).

In contrast, generative models fit probability distributions $p(\chi)$ to $\chi_1, \chi_2, \ldots, \chi_m$ and base their predictions on the likelihood under different models. When faced with a discriminative task, approaching it from the generative angle is generally suboptimal. On the other hand, it is often easier to capture the structure of complex objects with generative models than directly with kernels. Specific examples include multinomial models for text documents, hidden Markov models for speech, Markov random fields for images and linear dynamical systems for motion. In the current paper we aim to combine the advantages of both worlds by

1. fitting separate probabilistic models $p_1(x), p_2(x), \ldots p_m(x)$ to $\chi_1, \chi_2, \ldots, \chi_m$;

2. defining a novel kernel $k^{\text{prob}}(p, p')$ between probability distributions on $\mathcal{X}$;

3. finally, defining the kernel between examples to equal k^{prob} between the corresponding distributions:
$$k(\chi, \chi') = k^{\text{prob}}(p, p').$$

We then plug this kernel into any established kernel based learning algorithm and proceed as usual. We define k^{prob} in a rather general form and then investigate special cases.

Definition 1 *Let p and p' be probability distributions on a space $\mathcal{X}$ and ρ be a positive constant. Assume that $p^\rho, p'^\rho \in L_2(\mathcal{X})$, i.e. that $\int_{\mathcal{X}} p(x)^{2\rho} dx$ and $\int_{\mathcal{X}} p'(x)^{2\rho} dx$ are well defined (not infinity).*

The **probability product kernel** *between distributions p and p' is defined*

$$k^{prob}(p,p') = \int_{\mathcal{X}} p(x)^{\rho}\, p'(x)^{\rho}\, dx = \left\langle p^{\rho}, p'^{\rho} \right\rangle_{L_2}. \tag{2}$$

It is well known that $L_2(\mathcal{X})$ is a Hilbert space, hence for any set $\mathcal{P}$ of probability distributions over $\mathcal{X}$ such that $\int_{\mathcal{X}} p(x)^{2\rho} dx$ is finite for any $p \in \mathcal{P}$, the kernel defined by (2) is positive definite. The probability product kernel is also a simple and intuitively compelling notion of similarity between distributions.

2.1 Special Cases and Relationship to Statistical Affinities

For $\rho = 1/2$

$$k(p,p') = \int \sqrt{p(x)}\sqrt{p'(x)}\, dx,$$

which we shall call the **Bhattacharyya kernel**, because in the statistics literature it is known as Bhattacharyya's affinity between distributions (Bhattacharyya, 1943), related to the better-known Hellinger's distance

$$H(p,p') = \frac{1}{2}\int \left(\sqrt{p(x)} - \sqrt{p'(x)}\right)^2 dx$$

by $H(p,p') = \sqrt{2 - 2k(p,p')}$. Hellinger's distance can be seen as a symmetric approximation to the Kullback-Leibler (KL) divergence, and in fact is a bound on KL, as shown in (Topsoe, 1999), where relationships between several common divergences are discussed. The Bhattacharyya kernel was first introduced in (Kondor and Jebara, 2003) and has the important special property $k(\chi,\chi) = 1$.

When $\rho = 1$, the kernel takes the form of the expectation of one distribution under the other:

$$k(\chi,\chi') = \int p(x)\, p'(x)\, dx = \mathrm{E}_p[p'(x)] = \mathrm{E}_{p'}[p(x)]. \tag{3}$$

We call this the **expected likelihood kernel**.

It is worth noting that when dealing with distributions over discrete spaces $\mathcal{X} = \{x_1, x_2, \ldots\}$, probability product kernels have a simple geometrical interpretation. In this case p can be represented as a vector $p = (p_1, p_2, \ldots)$ where $p_i = \Pr(X = x_i)$. The probability product kernel is then the dot product between $\tilde{p} = \left(p_1^{\rho}, p_2^{\rho}, \ldots\right)$ and $\tilde{p}' = \left(p_1'^{\rho}, p_2'^{\rho}, \ldots\right)$.

In particular, in the case of the Bhattacharyya kernel, $\tilde{p}$ and $\tilde{p}'$ are vectors on the unit sphere. This type of similarity measure is not unknown in the literature. In text recognition, for example, the so-called "cosine-similarity" (i.e. the dot product measure) is well entrenched, and it has been observed that preprocessing word counts by taking square roots often improves performance (Goldzmidt and Sahami, 1998; Cutting et al., 1992). Lafferty and Lebanon also arrive at a similar kernel, but from a very different angle, looking at the diffusion kernel on the statistical manifold of multinomial distributions (Lafferty and Lebanon, 2002).

2.2 Frequentist and Bayesian Methods of Estimation

Various statistical estimation methods can be used to fit the distributions $p_1, p_2, \ldots, p_m$ to the examples $\chi_1, \chi_2, \ldots, \chi_m$. Perhaps it is most immediate to consider a parametric family $\{p_\theta(x)\}$, take the maximum likelihood estimators $\hat{\theta}_i = \arg\max_\theta p_\theta(\chi_i)$, and set $p_i(x) = p_{\hat{\theta}_i}(x)$.

	$\mathcal{T}(x)$	$\mathcal{A}(x)$	$\mathcal{K}(\theta)$
Gaussian ($\theta=\mu$, $\sigma^2=1$)	x	$-\frac{1}{2}x^Tx-\frac{D}{2}\log(2\pi)$	$\frac{1}{2}\theta^T\theta$
Gaussian ($\theta=\sigma^2$, $\mu=0$)	x	$-\frac{1}{2}\log(2\pi)$	$-\frac{1}{2}\log\theta$
Exponential ($\theta=-1/\beta$)	x	0	$-\log(-\theta)$
Gamma ($\theta=\alpha$)	$\log x$	$-\log x-x$	$\log\Gamma(\theta)$
Poisson ($\theta=\log p$)	x	$-\log(x!)$	$\exp\theta$
Multinomial ($\theta_i=\log\alpha_i$)	$(x_1,x_2,\ldots)$	$\log\big((\sum_i x_i)!/(\prod_i x_i!)\big)$	1

Table 1: Some well-known exponential family distributions

The alternative, Bayesian, strategy is to postulate a prior on θ, invoke Bayes' rule

$$p(\theta|\chi)=\frac{p(\chi|\theta)\,p(\theta)}{\int p(\chi|\theta)\,p(\theta)\,d\theta},$$

and then use either the distribution $p(x|\hat{\theta}_{\mathrm{MAP}})$ based on the maximum a posteriori estimator $\hat{\theta}_{\mathrm{MAP}}=\arg\max_\theta p(\theta|\chi)$, or the true posterior

$$p(x|\chi)=\int p(x|\theta)\,p(\theta|\chi)\,d\theta. \tag{4}$$

At this point the reader might be wondering to what extent it is justifiable to fit distributions to single data points. It is important to bear in mind that $p_i(x)$ is but an intermediate step in forming the kernel, and is not necessarily meant to model anything in and of itself. The motivation is to exploit the way that probability distributions capture similarity: assuming that our model is appropriate, if a distribution fit to one data point gives high likelihood to another data point, this indicates that the two data points are in some sense similar. This is particularly true of intricate graphical models commonly used to model structured data, such as times series, images, etc.. Such models can capture relatively complicated relationships in real world data. Probability product kernels allow kernel methods to harness the power of such models.

When χ is just a point in $\mathbb{R}^n$ with no further structure, probability product kernels are less compelling. Nevertheless, it is worth noting that even the Gaussian RBF kernel (1) can be regarded as a probability product kernel (by setting setting $\rho=1$ and choosing $p_i(x)$ to be a $\sigma/2$ variance Gaussian fit to χ_i by maximum likelihood). In particular situations when the point χ does not yield a reliable estimate of p (typically because the class of generative models is too flexible relative to the datum), we may consider regularizing the maximum likelihood estimate of each probability by fitting it to the neighbourhood of a point or by employing other more global estimators of the densities. Other related methods that use points to estimate local distributions including kernel density estimation (KDE) where the global density is composed of local density models centered on each datum (Silverman, 1986). We next discuss a variety of distributions (in particular, the exponential family) where maximum likelihood estimation is well behaved and the probability product kernel is straightforward to compute.

3. Exponential Families

A family of distributions parameterized by $\theta \in \mathbb{R}^D$ is said to form an exponential family if its members are of the form

$$p_\theta(x) \quad = \quad \exp(\mathcal{A}(x) + \theta^T \mathcal{T}(x) - \mathcal{K}(\theta)).$$

Here $\mathcal{A}$ is called the measure, $\mathcal{K}$ is the the cumulant generating function and $\mathcal{T}$ are the sufficient statistics.

Some of the most common statistical distributions, including the Gaussian, multinomial, Poisson and Gamma distributions, are exponential families (Table 1). Note that to ensure that $p_\theta(x)$ is normalized, $\mathcal{A}$, $\mathcal{K}$ and θ must be related through the Laplace transform

$$\mathcal{K}(\theta) = \log \int \exp\left(\mathcal{A}(x) + \theta^T \mathcal{T}(x)\right)\, dx.$$

The Bhattacharyya kernel ($\rho = 1/2$) can be computed in closed form for any exponential family:

$$\begin{aligned} k(\chi, \chi') = k(p, p') \quad &= \quad \int_x p_\theta(x)^{1/2} p_{\theta'}(x)^{1/2}\, dx \\ &= \quad \int_x \exp\left(\mathcal{A}(x) + \left(\tfrac{1}{2}\theta + \tfrac{1}{2}\theta'\right)^T \mathcal{T}(x) - \tfrac{1}{2}\mathcal{K}(\theta) - \tfrac{1}{2}\mathcal{K}(\theta')\right) dx \\ &= \quad \exp\left(\mathcal{K}\left(\tfrac{1}{2}\theta + \tfrac{1}{2}\theta'\right) - \tfrac{1}{2}\mathcal{K}(\theta) - \tfrac{1}{2}\mathcal{K}(\theta')\right). \end{aligned}$$

For $\rho \neq 1/2$, $k(\chi, \chi')$ can only be written in closed form when $\mathcal{A}$ and $\mathcal{T}$ obey some special properties. Specifically, if $2\rho\mathcal{A}(x) = \mathcal{A}(\eta x)$ for some η and $\mathcal{T}$ is linear in x, then

$$\begin{aligned} &\log \int \exp\left(2\rho\, \mathcal{A}(x) + \rho\left(\theta + \theta'\right)^T \mathcal{T}(x)\right) dx = \\ &\qquad \log\left[\frac{1}{\eta}\int \exp\left(\mathcal{A}(\eta x) + \frac{\rho}{\eta}\left(\theta + \theta'\right)^T \mathcal{T}(\eta x)\right) d(\eta x)\right] = \mathcal{K}\left(\tfrac{\rho}{\eta}\left(\theta + \theta'\right)\right) - \log \eta, \end{aligned}$$

hence

$$k(p, p') = \exp\left[\mathcal{K}\left(\tfrac{\rho}{\eta}\left(\theta + \theta'\right)\right) - \log \eta - \tfrac{\rho}{2}\mathcal{K}(\theta) - \tfrac{\rho}{2}\mathcal{K}(\theta')\right].$$

In the following we look at some specific examples.

3.1 The Gaussian Distribution

The D dimensional Gaussian distribution $p(x) = \mathcal{N}(\mu, \Sigma)$ is of the form

$$p(x) = (2\pi)^{-D/2} |\Sigma|^{-1/2} \exp\left(-\tfrac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

where Σ is a positive definite matrix and $|\Sigma|$ denotes its determinant. For a pair of Gaussians $p = \mathcal{N}(\mu, \Sigma)$ and $p' = \mathcal{N}(\mu', \Sigma')$, completing the square in the exponent gives the general probability product kernel

$$\begin{aligned} &K_\rho(\chi, \chi') = K_\rho(p, p') = \int_{\mathbb{R}^D} p(x)^\rho p'(x)^\rho dx = \\ &(2\pi)^{(1-2\rho)D/2} \rho^{-D/2} \left|\Sigma^\dagger\right|^{1/2} |\Sigma|^{-\rho/2} |\Sigma'|^{-\rho/2} \exp\left(-\frac{\rho}{2}\left(\mu^T \Sigma^{-1}\mu + \mu'^T \Sigma'^{-1}\mu' - \mu^{\dagger T} \Sigma^\dagger \mu^\dagger\right)\right), \quad (5) \end{aligned}$$

where $\Sigma^\dagger = \left(\Sigma^{-1} + \Sigma'^{-1}\right)^{-1}$ and $\mu^\dagger = \Sigma^{-1}\mu + \Sigma'^{-1}\mu'$. When the covariance is isotropic and fixed, $\Sigma = \sigma^2 I$, this simplifies to

$$K_\rho(p,p') = (2\rho)^{-D/2}\,(2\pi\sigma^2)^{(1-2\rho)D/2}\,e^{-\|\mu-\mu'\|^2/(4\sigma^2/\rho)},$$

which, for $\rho = 1$ (the expected likelihood kernel) simply gives

$$k(p,p') = \frac{1}{(4\pi\sigma^2)^{D/2}}\,e^{-\|\mu'-\mu\|^2/(4\sigma^2)},$$

recovering, up to a constant factor, the Gaussian RBF kernel (1).

3.2 The Bernoulli Distribution

The Bernoulli distribution $p(x) = \gamma^x(1-\gamma)^{1-x}$ with parameter $\gamma \in (0,1)$, and its D dimensional variant, sometimes referred to as Naive Bayes,

$$p(x) = \prod_{d=1}^{D} \gamma_d^{x_d}(1-\gamma_d)^{1-x_d}$$

with $\gamma \in (0,1)^D$, are used to model binary $x \in \{0,1\}$ or multidimensional binary $x \in \{0,1\}^D$ observations. The probability product kernel

$$K_\rho(\chi,\chi') = K_\rho(p,p') = \sum_{x\in\{0,1\}^D} \prod_{d=1}^{D} (\gamma_d\gamma'_d)^{\rho x_d}((1-\gamma_d)(1-\gamma'_d))^{\rho(1-x_d)}$$

factorizes as

$$K_\rho(p,p') = \prod_{d=1}^{D}\left[(\gamma_d\gamma'_d)^\rho + (1-\gamma_d)^\rho(1-\gamma'_d)^\rho\right].$$

3.3 The Multinomial Distribution

The multinomial model

$$p(x) = \frac{s!}{x_1!x_2!\ldots x_D!}\,\alpha_1^{x_1}\alpha_2^{x_2}\ldots\alpha_D^{x_D}$$

with parameter vector $\alpha = (\alpha_1,\alpha_2,\ldots,\alpha_D)$ subject to $\sum_{d=1}^D \alpha_d = 1$ is commonly used to model discrete integer counts $x = (x_1,x_2,\ldots,x_D)$ with $\sum_{i=1}^D x_i = s$ fixed, such as the number of occurrences of words in a document. The maximum likelihood estimate given observations $x^{(1)},x^{(2)},\ldots,x^{(n)}$ is

$$\hat{\alpha}_d = \frac{\sum_{i=1}^n x_d^{(i)}}{\sum_{i=1}^n \sum_{d=1}^D x_d^{(i)}}.$$

For fixed s, the Bhattacharyya kernel ($\rho = 1/2$) can be computed explicitly using the multinomial theorem:

$$k(p,p') = \sum_{\substack{x=(x_1,x_2,\ldots,x_D)\\ \sum_i x_i = s}} \frac{s!}{x_1!x_2!\ldots x_D!}\prod_{d=1}^{D}(\alpha_d\alpha'_d)^{x_d/2} = \left[\sum_{d=1}^{D}(\alpha_d\alpha'_d)^{1/2}\right]^s \tag{6}$$

which is equivalent to the homogeneous polynomial kernel of order s between the vectors $(\sqrt{\alpha_1}, \sqrt{\alpha_2}, \ldots, \sqrt{\alpha_D})$ and $(\sqrt{\alpha'_1}, \sqrt{\alpha'_2}, \ldots, \sqrt{\alpha'_D})$. When s is not constant, we can sum over all its possible values

$$k(p,p') = \sum_{s=0}^{\infty} \left[\sum_{d=1}^{D} (\alpha_d \alpha'_d)^{1/2} \right]^s = \left(1 - \sum_{d=1}^{D} (\alpha_d \alpha'_d)^{1/2} \right)^{-1}$$

or weight each power differently, leading to a power series expansion.

3.4 The Gamma and Exponential Distributions

The gamma distribution $\Gamma(\alpha,\beta)$ parameterized by $\alpha > 0$ and $\beta > 0$ is of the form

$$p(x) = \frac{1}{\Gamma(\alpha)\beta^{\alpha}} x^{\alpha-1} e^{-x/\beta}.$$

The probability product kernel between $p \sim \Gamma(\alpha,\beta)$ and $p' \sim \Gamma(\alpha',\beta')$ will then be

$$k_\rho(p,p') = \frac{\Gamma(\alpha^\dagger)\, {\beta^\dagger}^{\alpha^\dagger}}{\left[\Gamma(\alpha)\, \beta^{\alpha}\, \Gamma(\alpha')\, {\beta'}^{\alpha'} \right]^{\rho}}$$

where $\alpha^\dagger = \rho\,(\alpha + \alpha' - 2) + 1$ and $1/\beta^\dagger = \rho\,(1/\beta + 1/\beta')$. When $\rho = 1/2$ this simplifies to $\alpha^\dagger = (\alpha + \alpha')/2$ and $1/\beta^\dagger = (1/\beta + 1/\beta')/2$.

The exponential distribution $p(x) = \frac{1}{\beta} e^{-x/\beta}$ can be regarded as a special case of the gamma family with $\alpha = 1$. In this case

$$k_\rho(p,p') = \frac{\rho\,(1/\beta + 1/\beta')}{(\beta\beta')^{\rho}}.$$

4. Latent and Graphical Models

We next upgrade beyond the exponential family of distributions and derive the probability product kernel for more versatile generative distributions. This is done by considering latent variables and structured graphical models. While introducing additional complexity in the generative model and hence providing a more elaborate probability product kernel, these models do have the caveat that estimators such as maximum likelihood are not as well-behaved as in the simple exponential family models and may involve expectation-maximization or other only locally optimal estimates. We first discuss the simplest latent variable models, mixture models, which correspond to a single unobserved discrete parent of the emission and then upgrade to more general graphical models such as hidden Markov models.

Latent variable models identify a split between the variables in x such that some are observed and are denoted using x_{o} (where the lower-case 'o' is short for 'observed') while others are hidden and denoted using x_{h} (where the lower-case 'h' is short for 'hidden'). In the latent case, the probability product kernel is computed only by the inner product between two distributions $p(x_{\mathrm{o}})$ and $p'(x_{\mathrm{o}})$ over the observed components, in other words $k(p,p') = \int p(x_{\mathrm{o}}) p'(x_{\mathrm{o}}) dx_{\mathrm{o}}$. The additional variables x_{h} are incomplete observations that are not part of the original sample space (where the datum or data points to kernelize exist) but an augmentation of it. The desired $p(x_{\mathrm{o}})$ therefore, involves marginalizing away the hidden variables:

$$p(x_{\mathrm{o}}) = \sum_{x_{\mathrm{h}}} p(x_{\mathrm{o}}, x_{\mathrm{h}}) = \sum_{x_{\mathrm{h}}} p(x_{\mathrm{h}}) p(x_{\mathrm{o}} | x_{\mathrm{h}}).$$

For instance, we may consider a mixture of exponential families model (a direct generalization of the exponential family model) where x_h is a single discrete variable and where $p(x_\mathrm{o}|x_\mathrm{h})$ are exponential family distributions themselves. The probability product kernel is then given as usual between two such distributions $p(x_\mathrm{o})$ and $p'(x_\mathrm{o})$ which expand as follows:

$$k(p,p') \quad = \quad \sum_{x_\mathrm{o}} (p(x))^\rho \left(p'(x)\right)^\rho \; = \; \sum_{x_\mathrm{o}} \left(\sum_{x_\mathrm{h}} p(x_\mathrm{h}) p(x_\mathrm{o}|x_\mathrm{h}) \right)^\rho \left(\sum_{x'_\mathrm{h}} p'(x'_\mathrm{h}) p'(x_\mathrm{o}|x'_\mathrm{h}) \right)^\rho .$$

We next note a slight modification to the kernel which makes latent variable models tractable when $\rho \neq 1$. This alternative kernel is denoted $\tilde{k}(p,p')$ and also satisfies Mercer's condition using the same line of reasoning as was employed for $k(p,p')$ in the previous sections. Essentially, for $\tilde{k}(p,p')$ we will assume that the power operation involving ρ is performed on each entry of the joint probability distribution $p(x_\mathrm{o},x_\mathrm{h})$ instead of on the marginalized $p(x_\mathrm{o})$ alone as follows:

$$\tilde{k}(p,p') \quad = \quad \sum_{x_\mathrm{o}} \sum_{x_\mathrm{h}} (p(x_\mathrm{h}) p(x_\mathrm{o}|x_\mathrm{h}))^\rho \sum_{x'_\mathrm{h}} \left(p'(x'_\mathrm{h}) p'(x_\mathrm{o}|x'_\mathrm{h}) \right)^\rho .$$

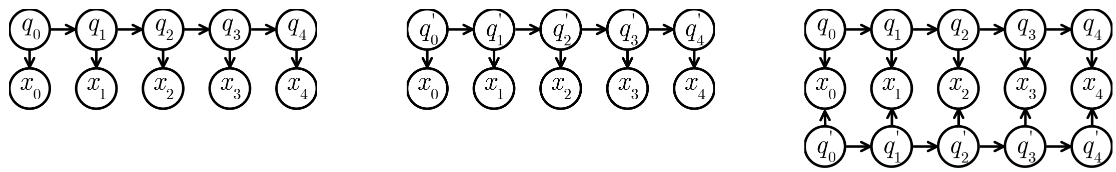
While the original kernel $k(p,p')$ involved the mapping to Hilbert space given by $\chi \rightarrow p(x_\mathrm{o})$, the above $\tilde{k}(p,p')$ corresponds to mapping each datum to an augmented distribution over a more complex marginalized space where probability values are squashed (or raised) by a power of ρ. Clearly, $k(p,p')$ and $\tilde{k}(p,p')$ are formally equivalent when $\rho = 1$. However, for other settings of ρ, we prefer handling latent variables with $\tilde{k}(p,p')$ (and at times omit the $\sim$ symbol when it is self-evident) since it readily accommodates efficient marginalization and propagation algorithms (such as the junction tree algorithm (Jordan and Bishop, 2004)).

One open issue with latent variables is that the $\tilde{k}$ kernels that marginalize over them may produce different values if the underlying latent distributions have different joint distributions over hidden and observed variables even though the marginal distribution over observed variables stays the same. For instance, we may have a two-component mixture model for p with both components having the same identical emission distributions yet the kernel $\tilde{k}(p,p')$ evaluates to a different value if we collapse the identical components in p into one emission distribution. This is to be expected since the different latent components imply a slightly different mapping to Hilbert space.

We next describe the kernel for various graphical models which essentially impose a factorization on the joint probability distribution over the N variables x_o and x_h. This article focuses on directed graphs (although undirected graphs can also be kernelized) where this factorization implies that the joint distribution can be written as a product of conditionals over each node or variable $x_i \in \{x_\mathrm{o},x_\mathrm{h}\}, i = 1 \ldots N$ given its parent nodes or set of parent variables $x_{pa(i)}$ as $p(x_\mathrm{o},x_\mathrm{h}) = \prod_{i=1}^N p(x_i|x_{pa(i)})$. We now discuss particular cases of graphical models including mixture models, hidden Markov models and linear dynamical systems.

4.1 Mixture Models

In the simplest scenario, the latent graphical model could be a mixture model, such as a mixture of Gaussians or mixture of exponential family distributions. Here, the hidden variable x_h is a single discrete variable which is a parent of a single x_o emission variable in the exponential family. To derive the full kernel for the mixture model, we first assume it is straightforward to compute an



(a) HMM for p. (b) HMM for p'. (c) Combined graphs.

Figure 1: Two hidden Markov models and the resulting graphical model as the kernel couples common parents for each node creating undirected edges between them.

elementary kernel between any pair of entries (indexed via x_{h} and x'_{h}) from each mixture model as follows:

$$\tilde{k}(p(x_{\mathrm{o}}|x_{\mathrm{h}}), p'(x_{\mathrm{o}}|x'_{\mathrm{h}})) \quad = \quad \sum_{x_{\mathrm{o}}} \left(p(x_{\mathrm{o}}|x_{\mathrm{h}}) p'(x_{\mathrm{o}}|x'_{\mathrm{h}})\right)^{\rho}.$$

In the above, the conditionals could be, for instance, exponential family (emission) distributions. Then, we can easily compute the kernel for a mixture model of such distributions using these elementary kernel evaluations and a weighted summation of all possible pairings of components of the mixtures:

$$\begin{aligned}
\tilde{k}(p, p') &= \sum_{x_{\mathrm{o}}} \sum_{x_{\mathrm{h}}} (p(x_{\mathrm{h}}) p(x_{\mathrm{o}}|x_{\mathrm{h}}))^{\rho} \sum_{x'_{\mathrm{h}}} \left(p'(x'_{\mathrm{h}}) p'(x_{\mathrm{o}}|x'_{\mathrm{h}})\right)^{\rho} \\
&= \sum_{x_{\mathrm{h}}, x'_{\mathrm{h}}} \left(p(x_{\mathrm{h}}) p'(x'_{\mathrm{h}})\right)^{\rho} \tilde{k}(p(x_{\mathrm{o}}|x_{\mathrm{h}}), p'(x_{\mathrm{o}}|x'_{\mathrm{h}})).
\end{aligned}$$

Effectively, the mixture model kernel involves enumerating all settings of the hidden variables x_{h} and x'_{h}. Taking the notation $|.|$ to be the cardinality of a variable, we effectively need to perform and sum a total of $|x_{\mathrm{h}}| \times |x'_{\mathrm{h}}|$ elementary kernel computations $\tilde{k}_{x_{\mathrm{h}}, x'_{\mathrm{h}}}$ between the individual emission distributions.

4.2 Hidden Markov Models

We next upgrade beyond mixtures to graphical models such as hidden Markov models (HMMs). Considering hidden Markov models as the generative model p allows us to build a kernel over sequences of variable lengths. However, HMMs and many popular Bayesian networks have a large number of (hidden and observed) variables and enumerating all possible hidden states for two distributions p and p' quickly becomes inefficient since x_{h} and x'_{h} are multivariate and/or have large cardinalities. However, unlike the plain mixture modeling case, HMMs have an efficient graphical model structure implying a factorization of their distribution which we can leverage to compute our kernel efficiently. A hidden Markov model over sequences of length $T+1$ has observed variables $x_{\mathrm{o}} = \{x_0, \ldots, x_T\}$ and hidden states $x_{\mathrm{h}} = \{q_0, \ldots, q_T\}$. The graphical model for an HMM in Figure 1(a) reflects its Markov chain assumption and leads to the following probability density function (note here we define $q_{-1} = \{\}$ as a null variable for brevity or, equivalently, $p(q_0|q_{-1}) = p(q_0)$):

$$p(x_{\mathrm{o}}) \quad = \quad \sum_{x_{\mathrm{h}}} p(x_{\mathrm{o}}, x_{\mathrm{h}}) \;=\; \sum_{q_0} \cdots \sum_{q_T} \prod_{t=0}^{T} p(x_t|q_t) p(q_t|q_{t-1}).$$

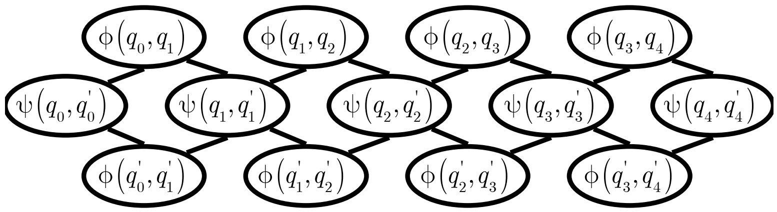


Figure 2: The undirected clique graph obtained from the kernel for efficiently summing over both sets of hidden variables x_{h} and x'_{h}.

To compute the kernel $\tilde{k}(p,p')$ in a brute force manner, we need to sum over all configurations of $q_0,\ldots,q_T$ and $q'_0,\ldots,q'_T$ while computing for each each configuration its corresponding elementary kernel $\tilde{k}(p(x_0,\ldots,x_T|q_0,\ldots,q_T),p'(x_0,\ldots,x_T|q'_0,\ldots,q'_T))$. Exploring each setting of the state space of the HMMs (shown in Figure 1(a) and (b)) is highly inefficient requiring $|q_t|^{(T+1)}\times|q'_t|^{(T+1)}$ elementary kernel evaluations. However, we can take advantage of the factorization of the hidden Markov model by using graphical modeling algorithms such as the junction tree algorithm (Jordan and Bishop, 2004) to compute the kernel efficiently. First, we use the factorization of the HMM in the kernel as follows:

$$\begin{aligned}
\tilde{k}(p,p') &= \sum_{x_0,\ldots,x_T}\sum_{q_0\ldots q_T}\prod_{t=0}^{T} p(x_t|q_t)^\rho p(q_t|q_{t-1})^\rho \sum_{q'_0\ldots q'_T}\prod_{t=0}^{T} p'(x_t|q'_t)^\rho p(q'_t|q'_{t-1})^\rho \\
&= \sum_{q_0\ldots q_T}\sum_{q'_0\ldots q'_T}\prod_{t=0}^{T} p(q_t|q_{t-1})^\rho p(q'_t|q'_{t-1})^\rho \left(\sum_{x_t} p(x_t|q_t)^\rho p'(x_t|q'_t)^\rho\right) \\
&= \sum_{q_0\ldots q_T}\sum_{q'_0\ldots q'_T}\prod_{t=0}^{T} p(q_t|q_{t-1})^\rho p(q'_t|q'_{t-1})^\rho \psi(q_t,q'_t) \\
&= \sum_{q_T}\sum_{q'_T}\psi(q_T,q'_T)\prod_{t=1}^{T}\sum_{q_{t-1}}\sum_{q'_{t-1}} p(q_t|q_{t-1})^\rho p(q'_t|q'_{t-1})^\rho \psi(q_{t-1},q'_{t-1}) p(q_0)^\rho p'(q'_0)^\rho.
\end{aligned}$$

In the above we see that we only need to compute the kernel for each setting of the hidden variable for a given time step on its own. Effectively, the kernel couples the two HMMs via their common children as in Figure 1(c). The kernel evaluations, once solved, form non-negative clique potential functions $\psi(q_t,q'_t)$ with a different setting for each value of q_t and q'_t. These couple the hidden variable of each HMM for each time step. The elementary kernel evaluations are easily computed (particularly if the emission distributions $p(x_t|q_t)$ are in the exponential family) as:

$$\tilde{k}(p(x_t|q_t),p'(x_t|q'_t)) \;=\; \psi(q_t,q'_t) \;=\; \sum_{x_t} p(x_t|q_t)^\rho p'(x_t|q'_t)^\rho.$$

Only a total of $(T+1)\times|q_t|\times|q'_t|$ such elementary kernels need to be evaluated and form a limited number of such clique potential functions. Similarly, each conditional distribution $p(q_t|q_{t-1})^\rho$ and $p'(q'_t|q'_{t-1})^\rho$ can also be viewed as a non-negative clique potential function, i.e. $\phi(q_t,q_{t-1}) =$

$p(q_t|q_{t-1})^\rho$ and $\phi(q'_t, q'_{t-1}) = p'(q'_t|q'_{t-1})^\rho$. Computing the kernel then involves marginalizing over the hidden variables x_h, which is done by running a forward-backward or junction-tree algorithm on the resulting undirected clique tree graph shown in Figure 2. Thus, to compute the kernel for two sequences χ and χ' of lengths $T_\chi + 1$ and $T_{\chi'} + 1$, we train an HMM from each sequence using maximum likelihood and then compute the kernel using the learned HMM transition matrix and emission models for a user-specified sequence length $T + 1$ (where T can be chosen according to some heuristic, for instance, the average of all T_χ in the training data set). The following is pseudo-code illustrating how to compute the $\tilde{k}(p, p')$ kernel given two hidden Markov models (p, p') and user-specified parameters T and ρ.

$\Phi(q_0, q'_0) = p(q_0)^\rho p'(q'_0)^\rho$
for $t = 1 \ldots T$

$$\Phi(q_t, q'_t) = \sum_{q_{t-1}} \sum_{q'_{t-1}} p(q_t|q_{t-1})^\rho p'(q'_t|q'_{t-1})^\rho \psi(q_{t-1}, q'_{t-1}) \Phi(q_{t-1}, q'_{t-1})$$

end

$$\tilde{k}(p, p') = \sum_{q_T} \sum_{q'_T} \Phi(q_T, q'_T) \psi(q_T, q'_T).$$

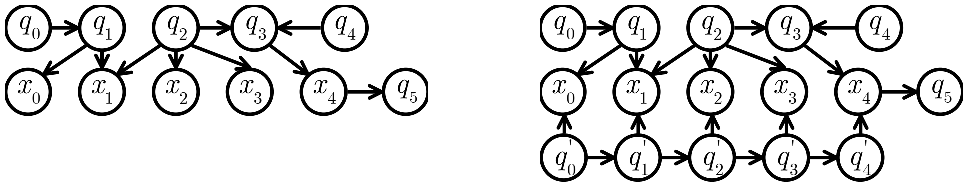
Note that the hidden Markov models p and p' need not have the same number of hidden states for the kernel computation.

4.3 Bayesian Networks

The above method can be applied to Bayesian networks in general where hidden and observed variables factorize according to $p(x) = \prod_{i=1}^N p(x_i|x_{pa(i)})$. The general recipe mirrors the above derivation for the hidden Markov model case. In fact, the two Bayesian networks need not have the same structure as long as they share the same sample space over the emission variables x_o. For instance, consider computing the kernel between the Bayesian network in Figure 3(a) and the hidden Markov model in Figure 1(b) over the same sample space. The graphs can be connected with their common children as in Figure 3(b). The parents with common children are married and form cliques of hidden variables. We then only need to evaluate the elementary kernels over these cliques giving the following non-negative potential functions for each observed node (for instance $x_i \in x_\mathrm{o}$) under each setting of all its parents' nodes (from both p and p'):

$$\begin{aligned}\psi(x_{\mathrm{h},\mathrm{pa}(i)}, x'_{\mathrm{h}',\mathrm{pa}(i)}) &= \tilde{k}(p(x_i|x_{\mathrm{h},\mathrm{pa}(i)}), p'(x_i|x_{\mathrm{h}',\mathrm{pa}(i)'})) \\ &= \sum_{x_i} p(x_i|x_{\mathrm{h},\mathrm{pa}(i)})^\rho p'(x_i|x_{\mathrm{h}',\mathrm{pa}(i)'})^\rho.\end{aligned}$$

After marrying common parents, the resulting clique graph is then built and used with a junction tree algorithm to compute the overall kernel from the elementary kernel evaluations. Although the resulting undirected clique graph may not always yield a dramatic improvement in efficiency as was seen in the hidden Markov model case, we note that propagation algorithms (loopy or junction tree algorithms) are still likely to provide a more efficient evaluation of the kernel than the brute force enumeration of all latent configurations of both Bayesian networks in the kernel (as was described in the generic mixture model case). While the above computations emphasized discrete latent variables, the kernel is also computable over continuous latent configuration networks, where



(a) Bayesian network for p. (b) Combined graphs.

Figure 3: The resulting graphical model from a Bayesian network and a hidden Markov model as the kernel couples common parents for each node creating undirected edges between them and a final clique graph for the junction tree algorithm.

x_{h} contains scalars and vectors, as is the case in linear Gaussian models, which we develop in the next subsections.

4.4 Linear Gaussian Models

A linear Gaussian model on a directed acyclic graph $\mathcal{G}$ with variables $x_1, x_2, \ldots, x_N$ associated to its vertices is of the form

$$p\left(x_1, x_2, \ldots, x_N\right) = \prod_i \mathcal{N}\left(x_i\,;\, \beta_{i,0} + \sum_{j \in \mathrm{pa}(i)} \beta_{i,j} x_j,\, \Sigma_i\right)$$

where $\mathcal{N}(x; \mu, \Sigma)$ is the multivariate Gaussian distribution, and $\mathrm{pa}(i)$ is the index set of parent vertices associated with the i^{th} vertex. The unconditional joint distribution can be recovered from the conditional form and is itself a Gaussian distribution over the variables in the model $p\left(x_1, x_2, \ldots, x_N\right)$. Hence, the probability product kernel between a pair of linear Gaussian models can be computed in closed form by using the unconditional joint distribution and (5).

Latent variable models require an additional marginalization step. Variables are split into two sets: the observed variables x_{o} and the hidden variables x_{h}. After the joint unconditional distribution $p(x_{\mathrm{o}}, x_{\mathrm{h}})$ has been computed, the hidden variables can be trivially integrated out:

$$k(p, p') \;=\; \int \left(\int p(x_{\mathrm{o}}, x_{\mathrm{h}})\, dx_{\mathrm{h}} \right)^{\rho} \left(\int p(x_{\mathrm{o}}, x'_{\mathrm{h}})\, dx'_{\mathrm{h}} \right)^{\rho} dx_{\mathrm{o}} \;=\; \int p(x_{\mathrm{o}})^{\rho}\, p'(x_{\mathrm{o}})^{\rho}\, dx_{\mathrm{o}},$$

so that the kernel is computed between Gaussians over only the observed variables.

4.5 Linear Dynamical Systems

A commonly used linear Gaussian model with latent variables is the linear dynamical system (LDS) (Shumway and Stoffer, 1982), also known as the state space model or Kalman filter model. The LDS is the continuous state analog of the hidden Markov model and shares the same conditional independence graph Figure 1(a). Thus, it is appropriate for modeling continuous time series data and continuous dynamical systems. The LDS's joint probability model is of the form

$$p(x_0, \ldots x_T, s_0, \ldots s_T) = \mathcal{N}(s_0; \mu, \Sigma)\, \mathcal{N}(x_0; C s_0, R) \prod_{t=1}^{T} \mathcal{N}(s_t; A s_{t-1}, Q)\, \mathcal{N}(x_t; C s_t, R),$$

where x_t is the observed variable at time t, and s_t is the latent state space variable at time t, obeying the Markov property.

The probability product kernel between LDS models is

$$k(p,p') = \int \mathcal{N}(x;\mu_x,\Sigma_{xx})^{\rho}\, \mathcal{N}(x;\mu'_x,\Sigma'_{xx})^{\rho}\, dx,$$

where μ_x and Σ_{xx} are the unconditional mean and covariance, which can be computed from the recursions

$$\begin{aligned}
\mu_{s_t} &= A\mu_{s_{t-1}} \\
\mu_{x_t} &= C\mu_{s_t} \\
\Sigma_{s_t s_t} &= A\Sigma_{s_{t-1}s_{t-1}}A' + Q \\
\Sigma_{x_t x_t} &= C\Sigma_{s_t s_t}C' + R.
\end{aligned}$$

As with the HMM, we need to set the number of time steps before computing the kernel. Note that the most algorithmically expensive calculation when computing the probability product kernel between Gaussians is taking the inverse of the covariance matrices. The dimensionality of the covariance matrix will grow linearly with the number of time steps and the running time of the matrix inverse grows cubically with the dimensionality. However, the required inverses can be efficiently computed because Σ_{xx} is block diagonal. Each extra time step added to the kernel will simply add another block, and the inverse of a block diagonal matrix can be computed by inverting the blocks individually. Therefore, the running time of inverting the block diagonal covariance matrix will only grow linearly with the number of time steps T and cubically with the (small) block size.

5. Sampling Approximation

To capture the structure of real data, generative models often need to be quite intricate. Closed form solutions for the probability product kernel are then unlikely to exist, forcing us to rely on approximation methods to compute $k(p,p')$.

Provided that evaluating the likelihood $p(x)$ is computationally feasible, and that we can also efficiently sample from the model, in the $\rho = 1$ case (expected likelihood kernel) we may employ the Monte Carlo estimate

$$k(p,p') \;\approx\; \frac{\beta}{N}\sum_{i=1}^{N} p'(x^i) \;+\; \frac{(1-\beta)}{N'}\sum_{i=1}^{N} p(x'^i),$$

where $x^1,\ldots,x^N$ and $x'^1,\ldots,x'^N$ are i.i.d. samples from p and p' respectively, and $\beta \in [0,1]$ is a parameter of our choosing. By the law of large numbers, this approximation will converge to the true kernel value in the limit $N \to \infty$ for any β. Other sampling techniques such as importance sampling and a variety of flavors of Markov chain Monte Carlo (MCMC), including Gibbs sampling, can be used to improve the rate of the sampling approximation's convergence to the true kernel.

In some cases it may be possible to use the sampling approximation for an general ρ. This is possible if a normalizing factor Z can be computed to renormalize $p(x)^{\rho}$ into a valid distribution. Z is computable for most discrete distributions and some continuous distribution, such as the Gaussian.

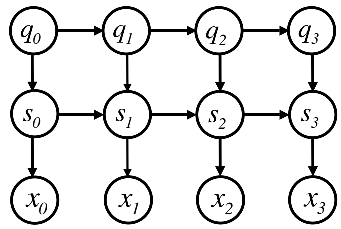


Figure 4: Graph for the Switching Linear Dynamical System.

The sampling approximation then becomes

$$k(p,p') \approx \frac{\beta}{N}\sum_{i=1}^{N} Z\, p'(\hat{x}^i)^\rho + \frac{(1-\beta)}{N'}\sum_{i=1}^{N} Z'\, p(\hat{x}'^i)^\rho,$$

where Z and Z' are the normalizers of p and p' after they are taken to the power of ρ. In this formulation $\hat{x}^1,\ldots,\hat{x}^N$ and $\hat{x}'^1,\ldots,\hat{x}'^N$ are i.i.d. samples from $\hat{p}$ and $\hat{p}'$ where $\hat{p} = \frac{p^\rho}{Z}$ and $\hat{p}' = \frac{p'^\rho}{Z'}$.

In practice, for a finite number of samples, the approximate kernel may not be a valid Mercer kernel. The non-symmetric aspect of the approximation can be alleviated by computing only the lower triangular entries in the kernel matrix and replicating them in the upper half. Problems associated with the approximation not being positive definite can be rectified by either using more samples, or by using an implementation for the kernel based classifier algorithm that can converge to a local optimum such as sequential minimal optimization (Platt, 1999).

5.1 Switching Linear Dynamical Systems

In some cases, part of the model can be evaluated by sampling, while the rest is computed in closed form. This is true for the switching linear dynamical system (SLDS) (Pavlovic et al., 2000), which combines the discrete hidden states of the HMM and the continuous state space of the LDS:

$$p(x,s,q) = p(q_0)\, p(s_0|q_0)\, p(x_0|s_0) \prod_{t=1}^{T} p(q_t|q_{t-1})\, p(s_t|s_{t-1},q_t)\, p(x_t|s_t),$$

where q_t is the discrete hidden state at time t, s_t is the continuous hidden state, and x_t are observed emission variables (Figure 4). Sampling $q^1\ldots,q^N$ according to $p(q)$ and $q'^1,\ldots,q'^N$ according to $p'(q')$, the remaining factors in the corresponding sampling approximation

$$\frac{1}{N}\sum_{i=1}^{N}\int\left(\int p(s_0|q_0^i)\, p'(s_0'|q_0'^i) \prod_{t=1}^{T} p(s_t|s_{t-1},q_t^i)\, p'(s_t'|s_{t-1}',q_t'^i) \prod_{t=0}^{T} p(x_t|s_t)\, p'(x_t|s_t')\, ds\, ds'\right)^\rho dx$$

form a non-stationary linear dynamical system. Once the joint distribution is recovered, it can be evaluated in closed form by (5) as was the case with a simple LDS. For $\rho \neq 1$ this is a slightly different formulation than the sampling approximation in the previous section. This hybrid method has the nice property that it is always a valid kernel for any number of samples since it simply becomes a convex combination of kernels between LDSs. An SLDS model is useful for describing input sequences that have a combination of discrete and continuous dynamics, such as a time series of human motion containing continuous physical dynamics and kinematics variables as well as discrete action variables (Pavlovic et al., 2000).

6. Mean Field Approximation

Another option when the probability product kernel between graphical models cannot be computed in closed form is to use variational bounds, such as the mean field approximation (Jaakkola, 2000). In this method, we introduce a variational distribution $Q(x)$, and using Jensen's inequality, lower bound the kernel:

$$\begin{aligned} k(p,p') &= \int p(x)^{\rho}\, p'(x)^{\rho}\, dx = \int \Psi(x)^{\rho}\, dx = \exp\left(\log \int \frac{Q(x)}{Q(x)}\, \Psi(x)^{\rho}\, dx\right) \\ &\geq \exp\left(\int Q(x) \log \frac{\Psi(x)^{\rho}}{Q(x)}\, dx\right) = \exp\left(\rho\, \mathrm{E}_Q\left[\log \Psi(x)\right] + H(Q)\right) = B(Q), \end{aligned}$$

where $\Psi(x) = p(x)\, p'(x)$ is called the potential, $\mathrm{E}_Q\left[\cdot\right]$ denotes the expectation with respect to $Q(x)$, and $H(Q)$ is the entropy. This transforms the problem from evaluating an intractable integral into that of finding the sufficient statistics of $Q(x)$ and computing the subsequent tractable expectations. Note, using the mean field formulation gives a different (and arguably closer result) to the desired intractable kernel than simply computing the probability product kernels directly between the approximate simple distributions. It is interesting to note the following form of the bound when it is expressed in terms of the Kullback-Leibler divergence, $D(Q\|p) = \int Q(x) \log \frac{Q(x)}{p(x)} dx$, and an entropy term. The bound on the probability product kernel is a function of the Kullback-Leibler divergence to p and p':

$$k(p,p') \;\geq\; B(Q) \;=\; \exp\left(-\rho\, D(Q\|p) - \rho\, D(Q\|p') + (1-2\rho)\, H(Q)\right).$$

The goal is to define $Q(x)$ in a way that makes computing the sufficient statistics easy. When the graphical structure of $Q(x)$ has no edges, which corresponds to the case that the underlying variables are independent, this is called the mean field method. When $Q(x)$ is made up of a number of more elaborate independent structures, it is called the structured mean field method. In either case, the variational distribution factors in the form $Q_1(x_1)Q_2(x_2)\ldots Q_N(x_N)$ with $x_1, x_2, \ldots, x_N$ disjoint subsets of the original variable set x.

Once the structure of $Q(x)$ has been set, its (locally) optimal parameterization is found by cycling through the distributions $Q_1(x_1), Q_2(x_2), \ldots, Q_N(x_N)$ and maximizing the lower bound $B(Q)$ with respect to each one, holding the others fixed:

$$\frac{\partial \log B(Q)}{\partial Q_n(x_n)} = \frac{\partial}{\partial Q_n(x_n)} \left[\rho \int Q_n(x_n)\, \mathrm{E}_Q\left[\log \Psi(x)|x_n\right]\, dx_n + H(Q_n) + \text{constant}\right] = 0.$$

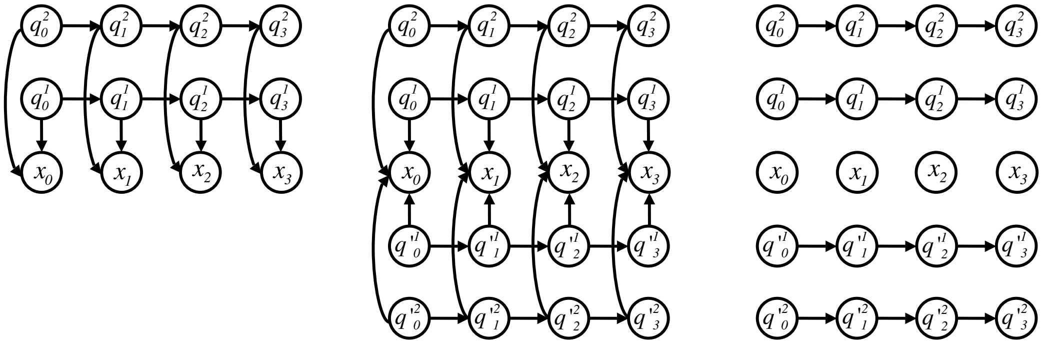
This leads to the update equations

$$Q_n(x_n) = \frac{1}{Z_n} \exp\left(\rho\, \mathrm{E}_Q\left[\log \Psi(x)|x_n\right]\right),$$

where the conditional expectation $\mathrm{E}_Q\left[\cdot\,|\, x_n\right]$ is computed by taking the expectation over all variables except x_n and

$$Z_n = \int \exp\left(\rho\, \mathrm{E}_Q\left[\log \Psi(x)|x_n\right]\right) dx$$

is the normalizing factor guaranteeing that $Q(x)$ remains a valid distribution.



(a) FHMM $P(x,q)$. (b) Potential $\Psi(x,q,q')$. (c) Variational $Q(x,q,q')$.

Figure 5: Graphs associated with the factorial hidden Markov model, its probability product kernel potential, and its structured mean field distribution.

In practice this approximation may not give positive definite (PD) kernels. Much current research has been focused on kernel algorithms using non-positive definite kernels. Algorithms such as sequential minimal optimization (Platt, 1999) can converge to locally optimal solutions. The learning theory for non-PD kernels is currently of great interest. See Ong et al. (2004) for an interesting discussion and additional references.

6.1 Factorial Hidden Markov Models

One model that traditionally uses a structured mean field approximation for inference is the factorial hidden Markov model (FHMM)(Ghahramani and Jordan, 1997). Such a model is appropriate for modeling output sequences that are emitted by multiple independent hidden processes. The FHMM is given by

$$p(x_0,x_1,x_2,\ldots,x_T) = \prod_{c=1}^{C}\left[p(q_0^c)\prod_{t=1}^{T}p(q_t^c|q_{t-1}^c)\right]\prod_{t=0}^{T}p(x_t|q_t^1,\ldots,q_t^C),$$

where q_t^c is the factored discrete hidden state at time t for chain c and x_t is the observed Gaussian emission variable at time t. The graphs of the joint FHMM distribution, the potential $\Psi(x,q,q')$, and the variational distribution $Q(x,q,q')$ are depicted in Figure 5. The structured mean field approximation is parameterized as

$$\begin{aligned}
Q(q_0^c=i) &\propto \exp\left(\rho\log\phi^c(i)-\frac{\rho}{2}\log|\Sigma_i|-\frac{\rho}{2}\mathrm{E}_Q\left[(x_0-\mu_i)^T\Sigma_i^{-1}(x_0-\mu_i)\right]\right)\\
Q(q_t^c=i\,|\,q_{t-1}^c=j) &\propto \exp\left(\rho\log\Phi^c(i,j)-\frac{\rho}{2}\log|\Sigma_i|-\frac{\rho}{2}\mathrm{E}_Q\left[(x_t-\mu_i)^T\Sigma_i^{-1}(x_t-\mu_i)\right]\right)\\
Q(x_t) &= \mathcal{N}(x_t;\hat{\mu}_t,\hat{\Sigma}_t)\\
\hat{\Sigma}_t &= \frac{1}{\rho}\left[\sum_{i,c}\mathrm{E}_Q\left[s_t^c(i)\right]\Sigma_i^{-1}+\sum_{i,c}\mathrm{E}_Q\left[q_t'^c(i)\right]\Sigma_i'^{-1}\right]^{-1}\\
\hat{\mu}_t &= \rho\,\hat{\Sigma}_t\left[\sum_{i,c}\mathrm{E}_Q\left[q_t^c(i)\right]\Sigma_i^{-1}\mu_i+\sum_{i,c}\mathrm{E}_Q\left[q_t'^c(i)\right]\Sigma_i'^{-1}\mu_i'\right].
\end{aligned}$$

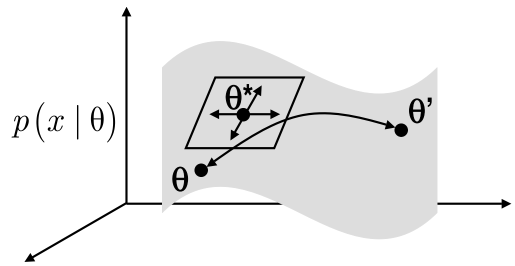


Figure 6: A statistical manifold with a geodesic path between two generative models θ and θ' as well as a local tangent space approximation at, for instance, the maximum likelihood model θ^* for the aggregated data set.

It is necessary to compute the expected sufficient statistics to both update the mean field equations and to evaluate the bound. The expectations $\mathrm{E}_Q[x_t]$ and $\mathrm{E}_Q\left[x_t x_t^T\right]$ can easily be computed from $Q(x_t)$, whereas $Q(s_t^c = i)$ and $Q(s_t^c = i, s_{t-1}^c = j)$ can be computed efficiently by means of a junction tree algorithm on each independent chain in the approximating distribution.

7. Relationship to Other Probabilistic Kernels

While the probability product kernel is not the only kernel to leverage generative modeling, it does have advantages in terms of computational feasibility as well as in nonlinear flexibility. For instance, heat kernels or diffusion kernels on statistical manifolds (Lafferty and Lebanon, 2002) are a more elegant way to generate a kernel in a probabilistic setting, but computations involve finding geodesics on complicated manifolds and finding closed form expressions for the heat kernel, which are only known for simple structures such as spheres and hyperbolic surfaces. Figure 6 depicts such a statistical manifold. The heat kernel can sometimes be approximated via the geodesic distance from p which is parameterized by θ to p' parameterized by θ'. But, we rarely have compact closed-form expressions for the heat kernel or for these geodesics for general distributions. Only Gaussians with spherical covariance and multinomials are readily accommodated. Curiously, the probabilistic product kernel expression for both these two distributions seems to coincide closely with the more elaborate heat kernel form. For instance, in the multinomial case, both involve an inner product between the square-rooted count frequencies. However, the probability product kernel is straightforward to compute for a much wider range of distributions providing a practical alternative to heat kernels.

Another probabilistic approach is the Fisher kernel (Jaakkola and Haussler, 1998) which approximates distances and affinities on the statistical manifold by using the local metric at the maximum likelihood estimate θ^* of the whole data set as shown in Figure 6. One caveat, however, is that this approximation can produce a kernel that is quite different from the exact heat kernel above and may not preserve the interesting nonlinearities implied by the generative model. For instance, in the exponential family case, all distributions generate Fisher kernels that are linear in the sufficient statistics. Consider the Fisher kernel

$$k(\chi, \chi') = U_\chi I_{\theta^*}^{-1} U_{\chi'}$$

where $I_{\theta^*}^{-1}$ is the inverse Fisher information matrix at the maximum likelihood estimate and where we define

$$U_{\chi} = \nabla_\theta \log p(\chi|\theta)|_{\theta^*}.$$

In exponential families, $p_\theta(x) = \exp(\mathcal{A}(x) + \theta^T \mathcal{T}(x) - \mathcal{K}(\theta))$ producing the following $U_{\chi} = \mathcal{T}(\chi) - \mathcal{G}(\theta^*)$ where we define $\mathcal{G}(\theta^*) = \nabla_\theta \mathcal{K}(\theta))|_{\theta^*}$. The resulting Fisher kernel is then

$$k(\chi,\chi') = (\mathcal{T}(\chi) - \mathcal{G}(\theta^*))^T I_{\theta^*}^{-1} \left(\mathcal{T}(\chi')^T - \mathcal{G}(\theta^*)\right),$$

which is an only slightly modified form of the linear kernel in $\mathcal{T}(\chi)$. Therefore, via the Fisher kernel method, a Gaussian distribution over means generates only linear kernels. A Gaussian distribution over means and covariances generates quadratic kernels. Furthermore, multinomials generate log-counts unlike the square root of the counts in the probability product kernel and the heat kernel. In practical text classification applications, it is known that square root squashing functions on frequencies and count typically outperform logarithmic squashing functions (Goldzmidt and Sahami, 1998; Cutting et al., 1992). In (Tsuda et al., 2002; Kashima et al., 2003), another related probabilistic kernel is put forward, the so-called marginalized kernel. This kernel is again similar to the Fisher kernel as well as the probability product kernel. It involves marginalizing a kernel weighted by the posterior over hidden states given the input data for two different points, in other words the output kernel $k(x,x') = \sum_h \sum_{h'} p(h|x)p(h'|x')k((x,h),(x',h'))$. The probability product kernel is similar, however, it involves an inner product over both hidden variables and the input space x with a joint distribution.

A more recently introduced kernel, the exponentiated symmetrized Kullback-Leibler (KL) divergence (Moreno et al., 2004) is also related to the probability product kernel. This kernel involves exponentiating the negated symmetrized KL-divergence between two distributions

$$k(p,p') = \exp(-\alpha D(p\|p') - \alpha D(p'\|p) + \beta) \text{ where } D(p\|p') = \int_x p(x) \log \frac{p(x)}{p'(x)} dx$$

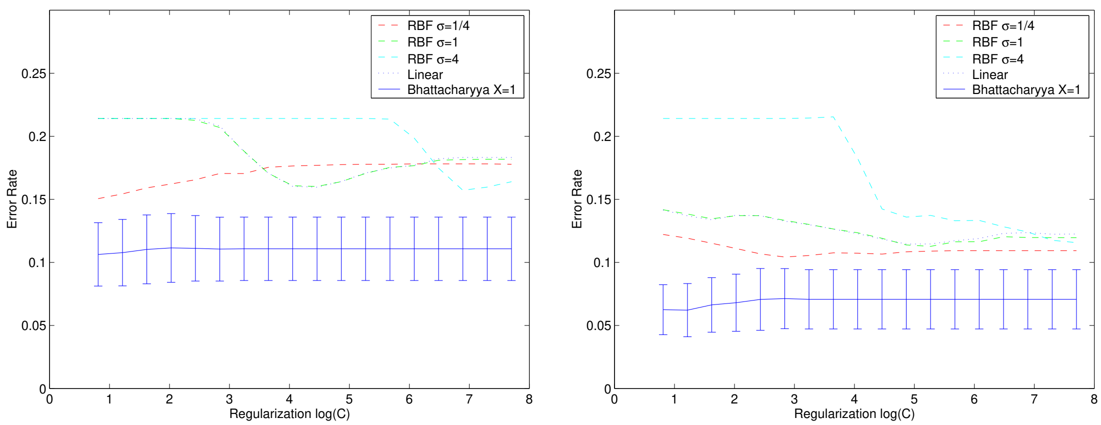
where α and β are user-defined scalar parameters. However, this kernel may not always satisfy Mercer's condition and a formal proof is not available. Another interesting connection is that this form is very similar to our mean-field bound on the probability product kernel (7). However, unlike the probability product kernel (and its bounds), computing the KL-divergence between complicated distributions (mixtures, hidden Markov models, Bayesian networks, linear dynamical systems and intractable graphical models) is intractable and must be approximated from the outset using numerical or sampling procedures which further decreases the possibility of having a valid Mercer kernel (Moreno et al., 2004).

8. Experiments

In this section we discuss three different learning problems: text classification, biological sequence classification and time series classification. For each problem, we select a generative model that is compatible with the domain which then leads to a specific form for the probability product kernel. For text, multinomial models are used, for sequences hidden Markov models are natural and for time series data we use linear dynamical systems. The subsequent kernel computations are fed to a discriminative classifier (a support vector machine) for training and testing.

8.1 Text Classification

For text classification, we used the standard WebKB data set which consists of HTML documents in multiple categories. Only the text component of each web page was preserved and HTML markup information and hyper-links were stripped away. No further stemming or elaborate text pre-processing was performed on the text. Subsequently, a bag-of-words model was used where each document is only represented by the frequency of words that appear within it. This corresponds to a multinomial distribution over counts. The frequencies of words are the maximum likelihood estimate for the multinomial generative model to a particular document which produces the vector of parameters $\hat{\alpha}$ as in Section 3.



(a) Training Set Size = 77 (b) Training Set Size = 622

Figure 7: SVM error rates (and standard deviation) for the probability product kernel (set at $\rho = 1/2$ as a Bhattacharyya kernel) for multinomial models as well as error rates for traditional kernels on the WebKB data set. Performance for multiple settings of the regularization parameter C in the support vector machine are shown. Two different sizes of training sets are shown (77 and 622). All results are shown for 20-fold cross-validation.

A support vector machine (which is known to have strong empirical performance on such text classification data) was then used to build a classifier. The SVM was fed our kernel's evaluations via a gram matrix and was trained to discriminate between faculty and student web pages in the WebKB data set (other categories were omitted). The probability product kernel was computed for multinomials under the parameter settings $\rho = 1/2$ (Bhattacharyya kernel) and $s = 1$ as in Section 3. Comparisons were made with the (linear) dot product kernel as well as Gaussian RBF kernels. The data set contains a total of 1641 student web pages and 1124 faculty web pages. The data for each class is further split into 4 universities and 1 miscellaneous category and we performed the usual training and testing split as described by (Lafferty and Lebanon, 2002; Joachims et al., 2001) where testing is performed on a held out university. The average error was computed from 20-fold cross-validation for the different kernels as a function of the support vector machine regularization parameter C in Figure 7. The figure shows error rates for different sizes of the training set (77 and 622 training points). In addition, we show the standard deviation of the error rate for the Bhattacharyya kernel. Even though we only explored a single setting of $s = 1, \rho = 1/2$ for the probability product kernel, it outperforms the linear kernel as well as the RBF kernel at multiple settings of the RBF σ parameter (we attempted $\sigma = \{1/4, 1, 4\}$). This result confirms previous intuitions in the text classification community which suggest using squashing functions on word frequencies such as the logarithm or the square root (Goldzmidt and Sahami, 1998; Cutting et al., 1992). This also related to the power-transformation used in statistics known as the Box-Cox transformation which may help make data seem more Gaussian (Davidson and MacKinnon, 1993).

8.2 Biological Sequence Classification

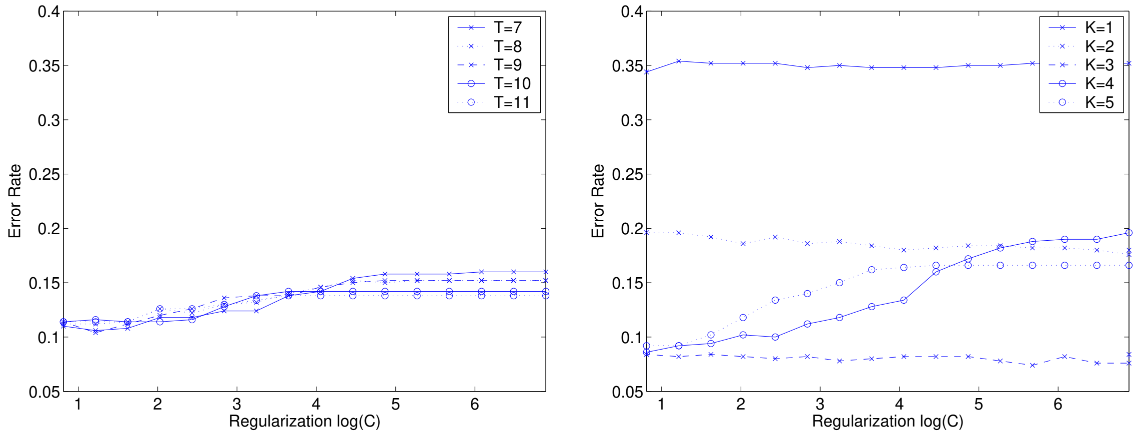
For discrete sequences, natural generative models to consider are Markov models and hidden Markov models. We obtained labeled gene sequences from the HS^3D data set[2] The sequences are of variable lengths and have discrete symbol entries from the 4-letter alphabet (G,A,T,C). We built classifiers to distinguish between gene sequences of two different types: introns and exons using raw sequences of varying lengths (from dozens of characters to tens of thousands of characters). From the original (unwindowed) 4450 introns and 3752 exons extracted directly from GenBank, we selected a random subset of 500 introns and 500 exons. These 1000 sequences were then randomly split into a 50% training and a 50% testing subset. In the first experiment we used the training data to learn stationary hidden Markov models whose number of states M was related to the length T_n of a given sequence n as follows: $M = \text{floor}(\frac{1}{2}\sqrt{O^2 + 4(T\zeta + O + 1)} - \frac{1}{2}O) + 1$. Here, O is the number of possible emission symbols (for gene sequences, $O = 4$) and ζ is the ratio of parameters to the number of symbols in the sequence T (we used $\zeta = 1/10$ although higher values may help avoid over-parameterizing the models). Each hidden Markov model was built from each sequence using the standard Expectation-Maximization algorithm (which is iterated for 400 steps to ensure convergence, this is particularly crucial for longer sequences). Gram matrices of size 1000×1000 were then formed by computing the probability product kernel between all the hidden Markov models. It was noted that all Gram matrices were positive definite by a singular value decomposition. We also used the following standard normalization of the kernel (a typical pre-processing step in the literature):

$$\tilde{k}(p,p') \quad \leftarrow \quad \frac{\tilde{k}(p,p')}{\sqrt{\tilde{k}(p,p)}\sqrt{\tilde{k}(p',p')}}.$$

Subsequently, we trained a support vector machine classifier on 500 example sequences and tested on the remaining 500. Figure 8(a) depicts the resulting error rate as we vary C, the regularization parameter on the SVM as well as T the number of time steps used by the kernel. Throughout, these experiments, we only used the setting $\rho = 1$. Note that these models were 1st order hidden Markov models where each state only depends on the previous state. Varying T, the length of the hidden Markov models used in the kernel computation has a slight effect on performance and it seems $T = 9$ at an appropriate C value performed best with an error rate as low as 10-11%.

For comparison, we also computed more standard string kernels such as the k-mer counts or spectrum kernels on the same training and testing gene sequences as shown in Figure 8(b). These basically correspond to a fully observed (non-hidden) stationary Markov model as in Figure 9. The zeroth order ($K = 1$) Markov model does poorly with its lowest error rate around 34%. The first

2. This is a collection of unprocessed intron and exon class gene sequences referred to as the *homo sapiens splice sites data set* and was downloaded from www.sci.unisannio.it/docenti/rampone.



(a) Hidden Markov Models (1st Order) (b) Markov Models (0th to 4th Order)

Figure 8: SVM error rates for the probability product kernel at $\rho = 1$ for hidden Markov models and Markov models (equivalent to string or spectrum kernels). In (a) test error rate under various levels of regularization is shown for 5 different settings of T in the probability product kernel. In (b) test error rate under various levels of regularization is shown for 5 different settings of the order K of the Markov model.

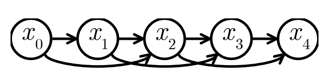


Figure 9: A higher order (2nd order or $K = 3$) Markov model.

order Markov model ($K = 2$) performs better with its lowest error rate at 18% yet is slightly worse than first order hidden Markov models. This helps motivate the possibility that dependence on the past in first order Markov models could be better modeled by a latent state as opposed to just the previous emission. This is despite the maximum likelihood estimation issues and local minima that we must contend with when training hidden Markov models using expectation-maximization. The second order Markov models with $K = 3$ fare best with an error rate of about 7-8%. However, higher orders ($K = 4$ and $K = 5$) seem to reduce performance. Therefore, a natural next experiment would be to consider higher order hidden Markov models, most notably second order ones where the emission distributions are conditioned on the past two states as opposed to the past two emitted symbols in the sequence.

8.3 Time Series Experiments

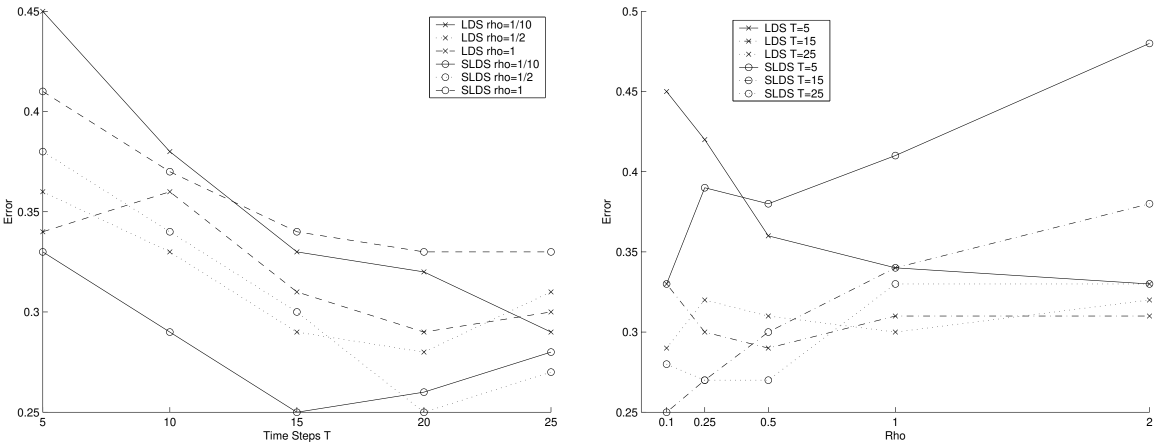
In this experiment we compare the performance of the probability product kernel between LDSs and the sampling approximation of the SLDS kernel with more traditional kernels and maximum likelihood techniques on synthetic data. We sampled from 20 exemplar distributions to generate synthetic time series data. The sampled distributions were 5 state, 2 dimensional SLDS models generated at random. Each model was sampled 5 times for 25 time steps with 10 models being assigned to each class. This gave 50 time series of length 25 per class, creating a challenging classification problem. For comparison, maximum likelihood models for LDSs and SLDSs were fit to the data and used for classification as well as SVMs trained using Gaussian RBF kernels. All testing was performed using leave one out cross validation. The SLDS kernel was approximated using sampling (50 samples) and the probability product kernels were normalized to improve performance.

The maximum likelihood LDS had an error rate of .35, the SLDS .30, and the Gaussian RBF .34. Exploring the setting of the parameters ρ and T for the LDS and SLDS kernels, we were able to outperform the maximum likelihood methods and the Gaussian RBF kernel with an optimal error for the LDS and SLDS kernel of .28 and .25 respectively. Figure 10 (a) and (b) show the error rate versus T and ρ respectively. It can be seen that increasing T generally improves performance which is to be expected. Although it does appear that T can be chosen too large. Meanwhile, ρ, generally appears to perform better when it is smaller (a counter example is the LDS kernel at the setting $T = 5$). Overall, the kernels performed comparably to or better than standard methods for most settings of ρ and T with the exception of extreme settings.

9. Conclusions

Discriminative learning, statistical learning theory and kernel-based algorithms have brought mathematical rigor and practical success to the field making it is easy to overlook the advantages of generative probabilistic modeling. Yet generative models are intuitive and offer flexibility for inserting structure, handling unusual input spaces, and accommodating prior knowledge. By proposing a kernel between the models themselves, this paper provides a bridge between the two schools of thought.

The form of the kernel is almost as simple as possible, yet it still gives rise to interesting nonlinearities and properties when applied to different generative models. It can be computed explicitly for some of the most commonly used parametric distributions in statistics such as the exponential family. For more elaborate models (graphical models, hidden Markov models, linear dynamical systems, etc.), computing the kernel reduces to using standard algorithms in the field. Experiments



(a) (b)

Figure 10: A comparison of the choice of parameters (a) T and (b) ρ. The x-axis is the parameter and the y-axis is the error rate.

show that probability product kernels hold promise in practical domains. Ultimately, engineering kernels between structured, discrete or unusual objects is an area of active research and, via generative modeling, probability product kernels can bring additional flexibility, formalism and potential.

Acknowledgments

The authors thank the anonymous reviewers for their helpful comments. This work was funded in part by the National Science Foundation (grants IIS-0347499 and CCR-0312690).

References

Y. Bengio and P. Frasconi. Input-output HMM's for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249, September 1996.

A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math Soc.*, 1943.

M. Collins and N. Duffy. Convolution kernels for natural language. In *Neural Information Processing Systems 14*, 2002.

C. Cortes, P. Haffner, and M. Mohri. Rational kernels. In *Neural Information Processing Systems 15*, 2002.

D. R. Cutting, D. R. Karger, J. O. Pederson, and KJ. W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *Proceedings ACM/SIGIR*, 1992.

R. Davidson and J. MacKinnon. *Estimation and Inference in Econometrics.* Oxford University Press, 1993.

Z. Ghahramani and M. Jordan. Factorial hidden Markov models. *Machine Learning*, 29:245–273, 1997.

M. Goldzmidt and M. Sahami. A probabilistic approach to full-text document clustering. Technical report, Stanford University, 1998. Database Group Publication 60.

T. Hastie, R. Tibshirani, and Friedman J. H. *The Elements of Statistical Learning*. Springer Verlag, 2001.

D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz, 1999.

T. Jaakkola. Tutorial on variational approximation methods. In *Advanced Mean Field Methods: Theory and Practice*. MIT Press, 2000.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Neural Information Processing Systems 11*, 1998.

T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. In *Neural Information Processing Systems 12*, 1999.

T. Jebara and R. Kondor. Bhattacharyya and expected likelihood kernels. In *Conference on Learning Theory*, 2003.

T. Joachims, N. Cristianini, and J. Shawe-Taylor. Composite kernels for hypertext categorisation. In *International Conference on Machine Learning*, 2001.

M. Jordan and C. Bishop. *Introduction to Graphical Models*. In progress, 2004.

H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Machine Learning: Tenth International Conference, ICML*, 2003.

R. Kondor and T. Jebara. A kernel between sets of vectors. In *Machine Learning: Tenth International Conference*, 2003.

J. Lafferty and G. Lebanon. Information diffusion kernels. In *Neural Information Processing Systems*, 2002.

C. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for SVM protein classification. In *Neural Information Processing Systems*, 2002.

P. J. Moreno, P. P. Ho, and N. Vasconcelos. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In *Neural Information Processing Systems*, 2004.

C. Ong, A. Smola, and R. Williamson. Superkernels. In *Neural Information Processing Systems*, 2002.

C. S. Ong, M. Xavier, S. Canu, and A. J. Smola. Learning with non-positive kernels. In *ICML*, 2004.

V. Pavlovic, J. M. Rehg, and J. MacCormick. Learning switching linear models of human motion. In *Neural Information Processing Systems 13*, pages 981–987, 2000.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1997.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *J. of Time Series Analysis*, 3(4):253–264, 1982.

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall: London., 1986.

F. Topsoe. Some inequalities for information divergence and related measures of discrimination. *J. of Inequalities in Pure and Applied Mathematics*, 2(1), 1999.

K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18 (90001):S268–S275, 2002.

V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

S. V. N. Vishawanathan and A. J. Smola. Fast kernels for string and tree matching. In *Neural Information Processing Systems 15*, 2002.

C. Watkins. *Advances in kernel methods*, chapter Dynamic Alignment Kernels. MIT Press, 2000.