Chapter 1

Expressing Structure through Kernels

Kernels specify similarity between function values of two objects, not between similarity of objects.

When modeling functions, encoding known symmetries greatly aids learning and prediction. We demonstrate that in nonparametric regression, many types of symmetry can be enforced through operations on the covariance function. These symmetries can be composed to produce nonparametric priors on functions whose domains have interesting topological structure such as spheres, torii, and Möbius strips. We demonstrate that marginal likeihood an be used to automatically search over such structures.

Joint work with David Reshef, Roger Grosse, Joshua B. Tenenbaum

1.1 Introduction

It is well-known that the properties of the functions we wish to model can be expressed mainly through the covariance function **?**.

1.2 Expressing Symmetries

In this section, we give recipes for expressing several classes of symmetries. Later, we will show how these can be combined to produce more interesting structures.

Periodicity Given D dimensions, we can enforce rotational symmetry on any subset of the dimensions:

$$f(x) = f(x_i + k\tau_i) \quad \forall k \in \mathbb{Z} \tag{1.1}$$

by the applying a kernel between pairs transformed coordinates $\sin(x), \cos(x)$:

$$k_{\text{periodic}}(x, x') = k(\sin(x), \cos(x), \sin(x'), \cos(x')) \tag{1.2}$$

We can also apply rotational symmetry repeatedy to a single dimension.

Reflective Symmetry along an axis we can enforce the symmetry

$$f(x) = f(-x) \tag{1.3}$$

by the kernel transform

$$\begin{aligned} k_{\text{symm arg1}}(x, x') = k(x, x') + k(x, -x') \\ + k(-x, x') + k(-x, -x') \end{aligned} \tag{1.4}$$

Reflective Symmetry along a diagonal We can enforce symmetry between any two dimensions:

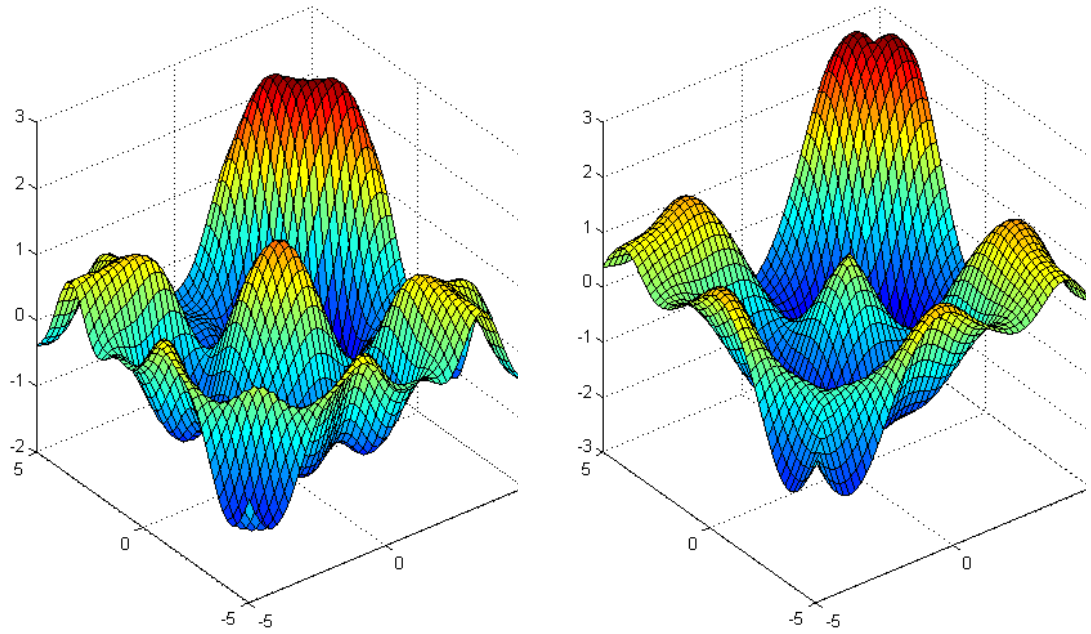
$$f(x, y) = f(y, x) \tag{1.5}$$

by two methods: In the additive method, we transform the kernel by:

$$\begin{aligned} k_{\text{reflect add}}(x, y, x', y') = k(x, y, x', y') \\ + k(x, y, y', x') \\ + k(y, x, x', y') \\ + k(y, x, y', x') \end{aligned} \tag{1.6}$$

or by

$$\begin{aligned} k_{\text{reflect min}}(x, y, x', y') = k(\min(x, y), \max(x, y), \\ \min(x', y'), \max(x', y')) \end{aligned} \tag{1.7}$$



Additive method min method

Fig. 1.1 An illustration of two methods of introducing symmetry: The additive method or the min method. The additive method has half the marginal variance away from $y = x$, but the min method introduces a non-differentiable seam along $y = x$.

however, the second method will in general lead to non-differentiability along $x = y$. Figure 1.1 shows the difference.

Spherical Symmetry We can also enforce that a function expresses the symmetries obeyed by $n - spheres$ by simply transforming a set of $n - 1$ coordinates by:

$$\begin{aligned} x_1 &= \cos(\phi_1) \\ x_2 &= \sin(\phi_1)\cos(\phi_2) \\ x_3 &= \sin(\phi_1)\sin(\phi_2)\cos(\phi_3) \\ &\vdots \\ x_{n-1} &= \sin(\phi_1)\cdots\sin(\phi_{n-2})\cos(\phi_{n-1}) \\ x_n &= \sin(\phi_1)\cdots\sin(\phi_{n-2})\sin(\phi_{n-1}) \end{aligned} \tag{1.8}$$

?

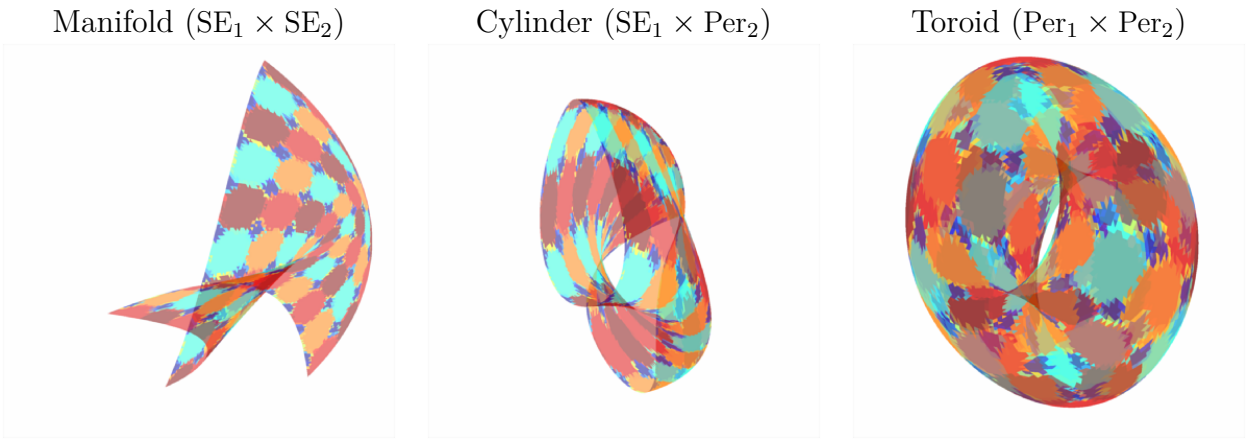


Fig. 1.2 Generating 2D manifolds with different topological structures. By enforcing that the functions mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$ obey the appropriate symmetries, the surfaces created have the corresponding topologies, ignoring self-intersections.

1.2.1 Parametric embeddings

In general, we can always enforce the symmetries obeyed by a given surface by finding a parametric embedding to that surface. However, it is not clear how to do this in general without introducing unnecessary

1.3 How to generate 3D shapes with a given topology

First create a mesh in 2d. Then draw 3 independent functions from a GP prior with the relevant symmetries encoded in the kernel. Then, map the 2d points making up the mesh through those 3 functions to get the 3D coordinates of each point on the mesh.

This is similar in spirit to the GP-LVM model **?**, which learns an embedding of the data into a low-dimensional space, and constructs a fixed kernel structure over that space.

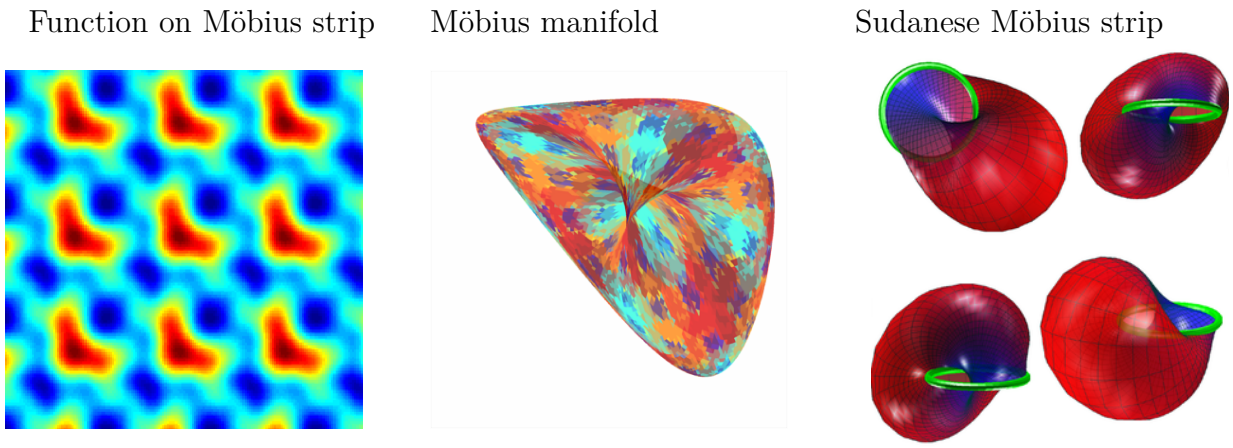


Fig. 1.3 Generating 2D manifolds with different topological structures. By enforcing that the functions mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$ obey the appropriate symmetries, the surfaces created have the corresponding topologies (ignoring self-intersections).

1.3.1 Möbius strips

A prior on functions on Möbius strips can be achieved by enforcing the symmetries:

$$f(x, y) = f(x, y + \tau_y) \tag{1.9}$$

$$f(x, y) = f(x + \tau_x, y) \tag{1.10}$$

$$f(x, y) = f(y, x) \tag{1.11}$$

If we imagine moving along the edge of a Möbius strip, that is equivalent to moving along a diagonal in the function generated. Figure **??** shows this. The second example is doesn't resemble a typical Möbius strip because the edge of the mobius strip is in a geometric circle. This kind of embedding is resembles the Sudanese Möbius strip [cite].

Another classic example of a function living on a Mobius strip is the auditory quality of 2-note intervals. The harmony of a pair of notes is periodic (over octaves) for each note, and the

1.4 Examples

1.4.1 Computing molecular energies

Figure **??** gives one example of a function which obeys the same symmetries as a Möbius strip, in some subsets of its arguments.

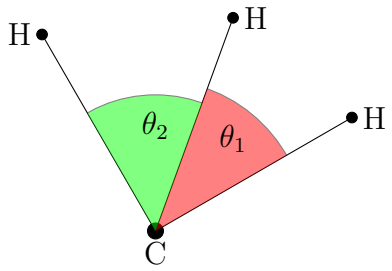


Fig. 1.4 An example of a function expressing the same symmetries as a Mobius strip in two of its arguments. The energy of a molecular configuration $f(\theta_1, \theta_2)$ depends only on the relative angles between atoms, and because each atom is indistiguishable, is invariant to permuting the atoms.

1.4.2 Translation invariance in images

Most models of images are invariant to spatial translations [cite convolution nets]. Similarily, most models of sounds are also invariant to translation through time.

Note that this sort of translational invariance is completely distinct from the stationarity properties of kernels used in Gaussian process priors. A stationary kernel implies that the prior is invariant to translations of the entire training and test set.

We are discussing here a discretized input space (into pixels or the audio equivalent), where the input vectors have one dimension for every pixel. We are interested in creating priors on functions that are invariant to shifting a signal along its pixels:

$$f\left(\quad\right) = f\left(\quad\right) \tag{1.12}$$

Translational invariance in this setting is equivalent to symmetries between dimensions in the input space.

This prior can be achieved in one dimension by using the following kernel transformation:

$$k\left((x_1, x_2, \ldots, x_D), (x'_1, x'_2, \ldots, x'_D)\right) = \sum_{i=1}^{D} \prod_{j=1}^{D} k(x_j, x'_{i+j \bmod D}) \tag{1.13}$$

Edge effects can be handled either by wrapping the image around, or by padding it with zeros.

Convolution The resulting kernel could be called a *discrete convolution kernel*. For an image with R, C rows and columns, it can also be written as:

$$k_{\text{conv}}\left((x_{11}, x_{12}, \ldots, x_{RC}), (x'_{11}, x'_{12}, \ldots, x'_{RC})\right) = \sum_{i=-L}^{L} \sum_{j=-L}^{L} k\left(\mathbf{x}, T_{ij}(\mathbf{x}')\right) \tag{1.14}$$

where $T_{ij}(\mathbf{x})$ is the operator which replaces each x_{mn} with $x_{m+i,n+j}$. Thus we are simply defining the covariance between two images to be the sum of all covariances between all relative translations of the two images. We can also normalize the kernel by pre-multiplying it with $\sqrt{k_{\text{conv}}(\mathbf{x}, \mathbf{x}) k_{\text{conv}}(\mathbf{x}', \mathbf{x}')}$.

Is there a pathology of the additive construction that appears in the limit?

1.4.3 Max-pooling

What we'd really like to do is a max-pooling operation. However, in general, a kernel which is the max of other kernels is not PSD [put counterexample here?]. Is the max over co-ordinate switching PSD?

1.5 Related Work

Invariances in Gaussian processes **?** show that, for Gaussian processes, with probability one, $f(\mathbf{x}) = f(T(\mathbf{x}))$ if and only if $k(x, x') = k(x, T(x'))$.

Structure discovery **?** learned the structural form of a graph used to model human similarity judgments. Examples of graphs included planes, trees, and cylinders. Some of their discrete graph structures have continous analogues in our own space; e.g. $\text{SE}_1 \times \text{SE}_2$ and $\text{SE}_1 \times \text{Per}_2$ can be seen as mapping the data to a plane and a cylinder, respectively.

1.6 Deep kernels

? showed that kernel machines have limited generalization ability when they use a local kernel such as the squared-exp. However, many interesting non-local kernels can be constructed which allow non-trivial extrapolation. For example, periodic kernels can be viewed as a 2-layer-deep kernel, in which the first layer maps $x \to [\sin(x), \cos(x)]$, and the second layer maps through basis functions corresponding to the SE kernel.

Can we construct other useful kernels by composing fixed feature maps several times, creating deep kernels? **?** constructed kernels of this form, repeatedly applying multiple layers of feature mappings. We can compose the feature mapping of two kernels:

$$k_1(\mathbf{x}, \mathbf{x}') = \mathbf{h}_1(\mathbf{x})^\mathsf{T}\mathbf{h}_1(\mathbf{x}') \tag{1.15}$$

$$k_2(\mathbf{x}, \mathbf{x}') = \mathbf{h}_2(\mathbf{x})^\mathsf{T}\mathbf{h}_2(\mathbf{x}') \tag{1.16}$$

$$(k_1 \circ k_2)(\mathbf{x}, \mathbf{x}') = k_2(\mathbf{h}_1(\mathbf{x}), \mathbf{h}_1(\mathbf{x}')) \tag{1.17}$$

$$= [\mathbf{h}_2(\mathbf{h}_1(\mathbf{x}))]^\mathsf{T}\mathbf{h}_2(\mathbf{h}_1(\mathbf{x}')) \tag{1.18}$$

Composing the squared-exp kernel with any implicit mapping $\mathbf{h}(\mathbf{x})$ has a simple closed form:

$$\begin{aligned} &k_{L+1}(\mathbf{x}, \mathbf{x}') = k_{SE}(\mathbf{h}(\mathbf{x}), \mathbf{h}(\mathbf{x}')) = \\ &= \exp\left(-\frac{1}{2}||\mathbf{h}(\mathbf{x}) - \mathbf{h}(\mathbf{x}')||_2^2\right) \\ &= \exp\left(-\frac{1}{2}\left[\mathbf{h}(\mathbf{x})^\mathsf{T}\mathbf{h}(\mathbf{x}) - 2\mathbf{h}(\mathbf{x})^\mathsf{T}\mathbf{h}(\mathbf{x}') + \mathbf{h}(\mathbf{x}')^\mathsf{T}\mathbf{h}(\mathbf{x}')\right]\right) \\ &= \exp\left(-\frac{1}{2}\left[k_L(\mathbf{x}, \mathbf{x}) - 2k_L(\mathbf{x}, \mathbf{x}') + k_L(\mathbf{x}', \mathbf{x}')\right]\right) \end{aligned} \tag{1.19}$$

Thus, we can express k_{L+1} exactly in terms of k_L.

Infinitely deep kernels What happens when we repeat this composition of feature maps many times, starting with the squared-exp kernel? In the infinite limit, this recursion converges to $k(\mathbf{x}, \mathbf{x}') = 1$ for all pairs of inputs, which corresponds to a prior on constant functions $f(\mathbf{x}) = c$.

A non-degenerate construction As before, we can overcome this degeneracy by connecting the inputs $\mathbf{x}$ to each layer. To do so, we simply augment the feature vector $\mathbf{h}_L(\mathbf{x})$ with $\mathbf{x}$ at each layer:

$$\begin{aligned} k_{L+1}(\mathbf{x}, \mathbf{x}') &= \exp\left(-\frac{1}{2}\left|\left|\begin{bmatrix}\mathbf{h}_L(\mathbf{x}) \\ \mathbf{x}\end{bmatrix} - \begin{bmatrix}\mathbf{h}_L(\mathbf{x}') \\ \mathbf{x}'\end{bmatrix}\right|\right|_2^2\right) \\ &= \exp\Big(-\frac{1}{2}\big[k_L(\mathbf{x}, \mathbf{x}) - 2k_L(\mathbf{x}, \mathbf{x}') \\ &\qquad + k_L(\mathbf{x}', \mathbf{x}') - ||\mathbf{x} - \mathbf{x}'||_2^2\big]\Big) \end{aligned} \tag{1.20}$$

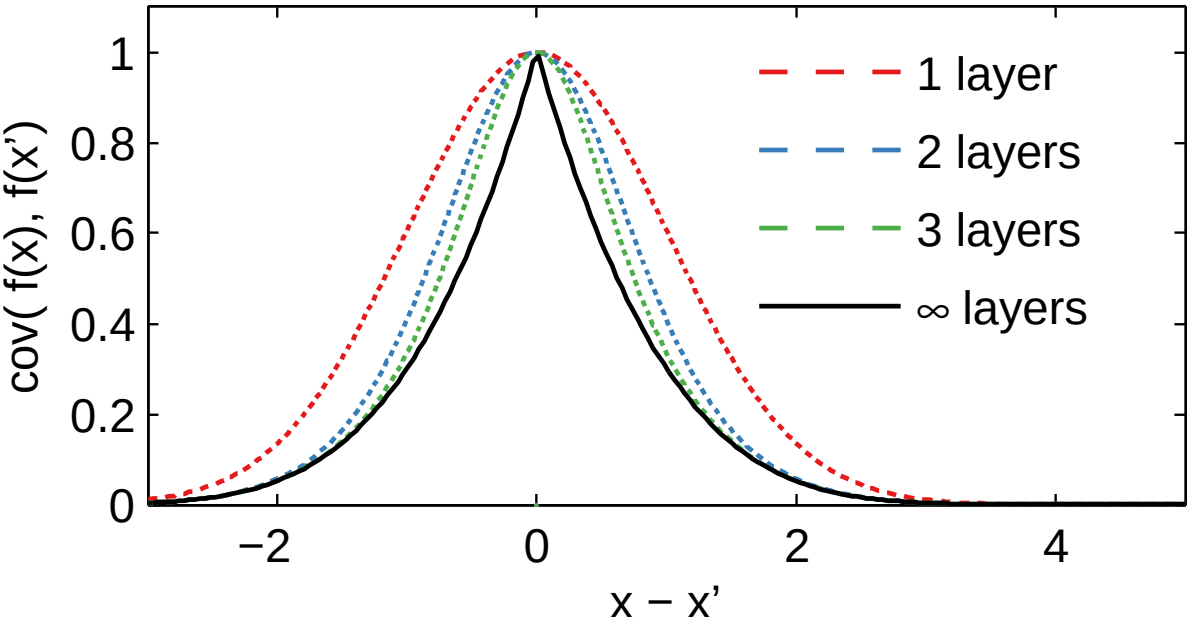


Fig. 1.5 Input-connected deep kernels. By connecting the inputs $\mathbf{x}$ to each layer, the kernel can still depend on its input even after arbitrarily many layers of computation.

For the SE kernel, this repeated mapping satisfies

$$k_\infty(\mathbf{x}, \mathbf{x}') - \log\left(k_\infty(\mathbf{x}, \mathbf{x}')\right) = 1 + \frac{1}{2}||\mathbf{x} - \mathbf{x}'||_2^2 \tag{1.21}$$

The solution to this recurrence has no closed form, but has a similar shape to the Ornstein-Uhlenbeck covariance $k_{\text{OU}}(x, x') = \exp(-|x - x'|)$ with lighter tails. Samples from a gp prior with this kernel are not differentiable, and are locally fractal.

1.6.1 When are deep kernels useful models?

Kernels correspond to fixed feature maps, and so kernel learning is an example of implicit representation learning. Such feature maps can capture rich structure (**?**), and can enable many types of generalization, such as translation and rotation invariance in images (**?**). **?** used a deep neural network to learn feature transforms for kernels, which learn invariances in an unsupervised manner. The relatively uninteresting properties of the kernels derived in this section simply reflect the fact that an arbitrary deep computation is not usually a useful representation, unless combined with learning.

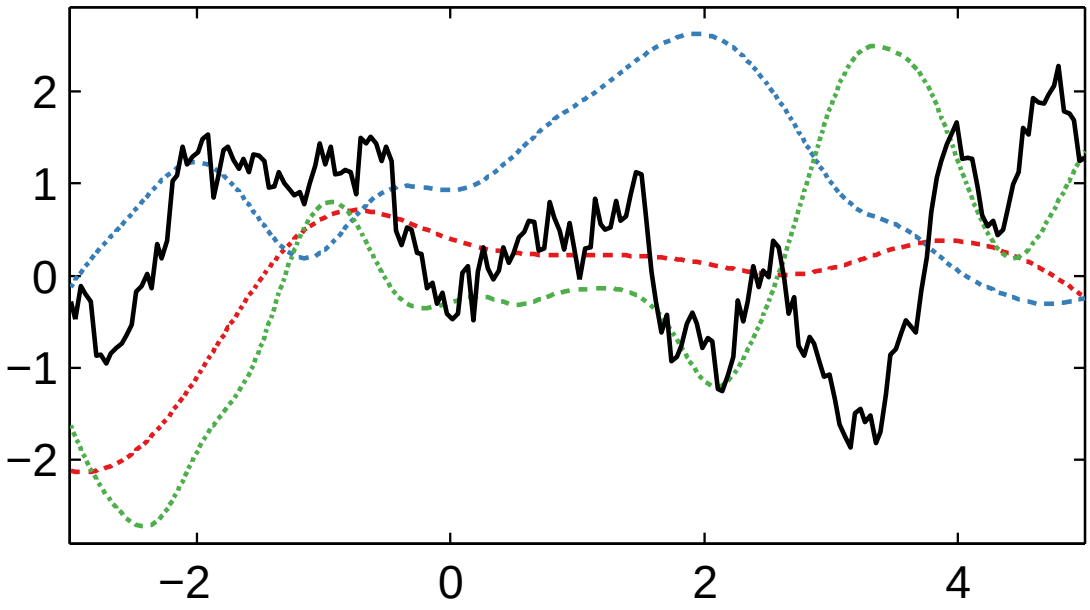


Fig. 1.6 GP draws using deep input-connected kernels.

References

F. Doshi-Velez and Z. Ghahramani. Accelerated sampling for the indian buffet process. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 273–280. ACM, 2009.

Harley Flanders. *Differential Forms with Applications to the Physical Sciences*. Dover Publications, December 1989. (page 3)

T. Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

D. Ginsbourger, O. Roustant, and N. Durrande. Invariances of random fields paths, with applications in gaussian process regression. Technical Report arXiv:1308.1359 [math.ST], August 2013. (page 7)

T. Griffiths and Z. Ghahramani. Infinite latent feature models and the indian buffet process. 2005.

C. Kemp and J.B. Tenenbaum. The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105(31):10687–10692, 2008. (page 7)

N. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *The Journal of Machine Learning Research*, 6:1783–1816, 2005. (page 4)

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.

K.T. Miller, J. Van Gael, and Y.W. Teh. Variational inference for the indian buffet process. In *Proceedings of the Intl. Conf. on Artificial Intelligence and Statistics*. Citeseer, 2009.

A. O'Hagan. Bayes-Hermite quadrature. *Journal of Statistical Planning and Inference*, 29:245–260, 1991.

C. E. Rasmussen and Z. Ghahramani. Bayesian monte carlo. In S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, Cambridge, MA, 2003.

C.E. Rasmussen. The infinite Gaussian mixture model. *Advances in Neural Information Processing Systems*, 12(5.2):2, 2000.

C.E. Rasmussen and CKI Williams. Gaussian Processes for Machine Learning. *The MIT Press, Cambridge, MA, USA*, 2006. (page 1)

F. Wood and T.L. Griffiths. Particle filtering for nonparametric bayesian matrix factorization. *Advances in Neural Information Processing Systems*, 19:1513, 2007.