Non-degenerate Priors for Arbitrarily Deep Networks

David Duvenaud
University of Cambridge
dkd23@cam.ac.uk

Oren Rippel
Harvard University
rippel@math.mit.edu

Ryan Adams
Harvard University
rpa@seas.harvard.edu

Zoubin Ghahramani
University of Cambridge
zoubin@eng.cam.ac.uk

Abstract

A good latent representation captures all the relevant degrees of freedom of the manifold on which the data live. We show, for typical deep architectures, that as the number of layers increase, the representational capacity of the model tends to capture fewer degrees of freedom. In the limit, deep representations only retain a single degree of freedom locally. In addition, gradient-based learning becomes intractable. We propose two alternate priors on network structure which do not suffer from these pathologies: First, simply connecting the input layer to every other layer. Second, a more general prior of computation graphs based on the Aldous-Hoover graph representation. We also analyze kernels obtained by taking arbitrarily-many feature compositions.

1 Introduction

Good initialization schemes can help, but we are interested in creating distributions on deep functions so that most of the mass is on functions without any pathologies. Also, we might be able to show that the pathologies we show that most functions exhibit means that it will be hard to learn them by gradient descent.

2 Relation between deep nets and Gaussian processes

Deep Gaussian processes Damianou and Lawrence [2012] are a prior on functions of the form: $f_{deep} = f^{1:L}(\mathbf{x}) = f^L(f^{L-1}(\dots f^3(f^2(f^1(\mathbf{x})))\dots))$, where each $f \overset{\text{iid}}{\sim} \text{GP}$.

Neal [1995] showed that the limit of a neural network with infinitely many hidden units. More precisely, he showed that for models of the form

$$f(x) = \alpha^T \Phi(x) = \sum_{i=i}^{K} \alpha_i \phi_i(x), \tag{1}$$

the central limit theorem implies that any two function values $f(x), f(x')$ have a distribution approaching $\mathcal{N}(,)$ as $K \to \infty$ with fixed feature vector $\Phi(x)$ and i.i.d. α approaches a distribution over functions in which all f(x), f(x') are jointly Gaussian distributed, under broad conditions. The result is surprisingly general: It doesn't put any constraints on what the features are (other than having bounded activation), nor does it require that the feature weighs α be Gaussian distributed. It only requires that the distribution on weights α has finite variance, and that as we increase the number of features K, we divide the feature weights by K. As $K \to \infty$, the central limit theorem gives that the weighted sum of feature activations approaches a Gaussian distribution.

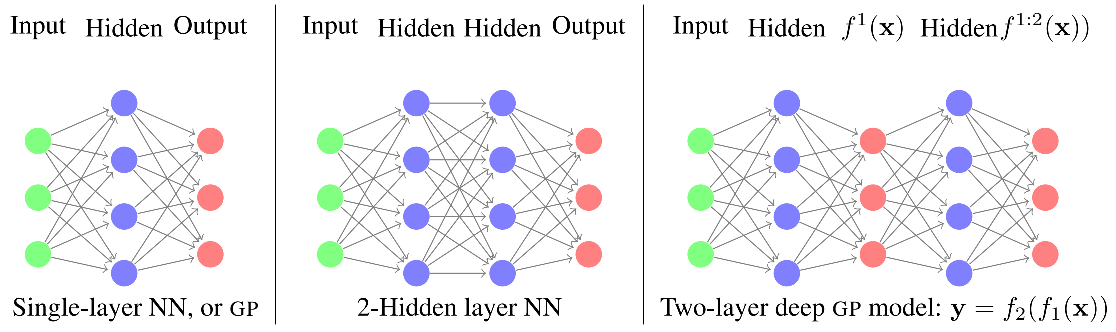


Figure 1: Comparing architectures. When there are multiple layers, the meaning of "hidden units" is slightly unclear in the deep GP model. We can consdier the model to be a linear combination of an infinite number of parametric basis functions hidden between each layer, or we can ignore the hidden layers and consider the deep GP to be a neural network with a finite number of non-parametric basis functions.

2.1 A neural net with one hidden layer

In a neural net with a single hidden layer, the correspondence is simple: The features $\phi(\mathbf{x})$ correspond to the hidden units. In the typical definition of a multi-layer perception, the hidden units of the first layer are defined as:

$$\Phi(\mathbf{x}) = h^{(1)}(\mathbf{x}) = \sigma(b^{(1)} + W^{(1)}\mathbf{x}) \tag{2}$$

The output $f(\mathbf{x})$ is simply a weighted sum of these hidden unit activations, for both GP models and neural networks.

2.2 A neural net with two hidden layers

In a neural net with two hidden layers, the correspondence is a little more complicated. Generally, the n^{th} layer units are given be the recurrence:

$$h^{(n)}(\mathbf{x}) = \sigma(b^{(n)} + W^{(n)}h^{(n-1)}(\mathbf{x})) \tag{3}$$

for real-valued outputs, the function can be written as weighted sum of the last layer's hidden units:

$$f(\mathbf{x}) = W^{(top)}\sigma(b^{(n)} + W^{(n)}h^{(n-1)}(\mathbf{x})) = W^{(top)}h^{(n)}(\mathbf{x})) \tag{4}$$

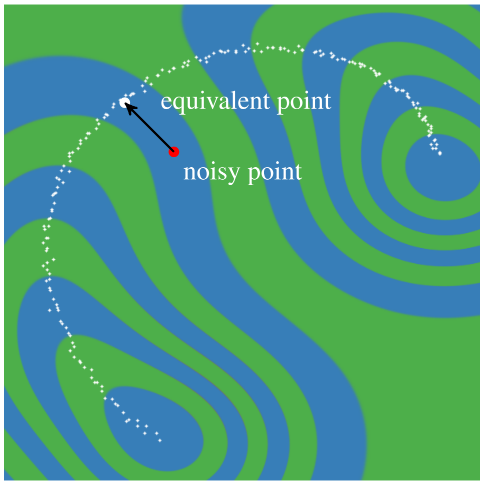
We can also write deep GP models in this way. However, for many covariance functions, we can't explicitly compute the activation of infinitly-many hidden functions $\Phi(\mathbf{x})$, and instead can only examine their weighted sum. Hence, to be able to compare the two model classes, we will view deep GP models in a different way, as a neural network with a finite number of nonparametric basis functions:

$$f^{1:L}(\mathbf{x}) = f^L(f^{L-1}(\dots f^3(f^2(f^1(\mathbf{x})))\dots)) \tag{5}$$

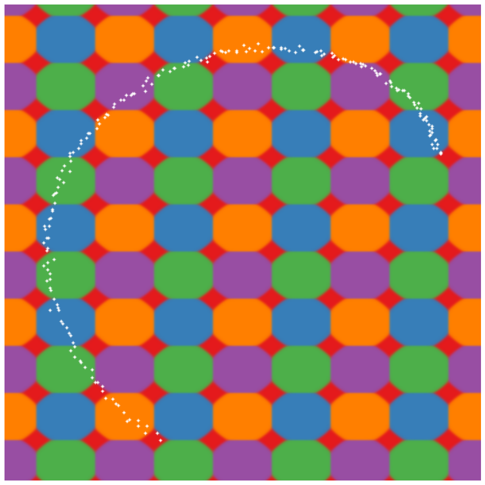
This view lets us compare GP models to multilayer perceptrons more directly, examining the activations and shapes of the finite number of basis functions at each layer, denoted by the vector of functions $f^\ell(\mathbf{x})$. Figure 1 compares these two architectures.

The effects of automatic relevance determination One feature of Gaussian process models that doesn't have a clear analogue in the neural-net literature is automatic relevance determination (ARD). Recall that in the ARD procedure, the lengthscales of each dimension are scaled to maximize the marginal likelihood of the GP. What effect will this have on archictures? In standard one-layer GP regression, it has the effect of smoothing out the posterior over functions in directions in which the data does not change.

Note that the ARD kernel encodes another assumption: That the function varies independently in all directions. This implies that using an ARD kernel to model a function that does *not* vary in all direction independently is 'wasting marginal likelihood' by not making strong enough assumptions.



A compact, noise-tolerant latent representation of a one-dimensional manifold



A naïve latent representation of a one-dimensional manifold

Figure 2: Comparing different latent representations of data on a 1-D manifold. A good latent representation is invariant in directions orthogonal to the data manifold, making the representation robust to noise in those directions. The representation must also change in directions tangent to the manifold, in order to preserve information for later layers.

Combining these two properties of ARD kernels, we can say that a deep GP will attempt to transform its input into a representation with as few degrees of freedom as possible, and for which the output function varies independently across each dimension.

2.3 Desirable properties of latent representations

Rifai et al. [2011a] make the point that a good latent representation is one that remains invariant when moving in directions orthogonal to the manifold that the data lie on. Conversely, a good latent representation must also change in directions tangential to the data manifold - otherwise we are losing information about the data. Figure 2 demonstrates this idea.

3 The Jacobian of Deep GPs is a product of i.i.d. Normal Matrices

In this section, we show a result which will let us characterize

Lemma 3.1. *The partial derivatives of a function drawn from a* GP *prior with a product kernel are i.i.d. Gaussian distributed.*

Proof. Because differentiation is a linear operator, the derivatives of a function drawn from a GP prior are also jointly Gaussian distributed. The covariance between partial derivatives w.r.t. input dimensions d_1 and d_2 of vector $\mathbf{x}$ are given by [Solak et al., 2003]:

$$\text{cov}\left(\frac{\partial f(\mathbf{x})}{\partial x_{d_1}}, \frac{\partial f(\mathbf{x})}{\partial x_{d_2}}\right) = \frac{\partial^2 k(\mathbf{x}, \mathbf{x}')}{\partial x_{d_1} \partial x'_{d_2}}\bigg|_{\mathbf{x}=\mathbf{x}'} \tag{6}$$

If our kernel is a product over individual dimensions $k(\mathbf{x}, \mathbf{x}') = \prod_d^D k_d(x_d, x'_d)$, as in the case of the isotropic squared-exp kernel, then the off-diagonal entries are zero, implying that all elements are independent and identically distributed. □

For example, in the case of the isotropic squared-exp kernel, covariance has the form:

$$k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \prod_{d=1}^{D} \left(-\frac{1}{2}\frac{(x_d - x'_d)^2}{\ell_d^2}\right) \implies \text{cov}\left(\frac{\partial f(\mathbf{x})}{\partial x_{d_1}}, \frac{\partial f(\mathbf{x})}{\partial x_{d_2}}\right) = \begin{cases} \frac{\sigma_f^2}{\ell_{d_1}^2} & \text{if } d_1 = d_2 \\ 0 & \text{if } d_1 \neq d_2 \end{cases} \tag{7}$$

Lemma 3.2. *The Jacobian of a* GP *with a seperable kernel is a matrix of i.i.d. Gaussian R.V.'s*

Proof. The Jacobian of the vector-valued function $f(\mathbf{x})$ is a matrix J whose $i, j^t h$ element is:

$$J_{ij} = \frac{\partial f_i(\mathbf{x})}{\partial x_j} \tag{8}$$

When composing L functions, we'll denote the *immediate* Jacobian of the function mapping from layer $\ell - 1$ to layer ℓ as $J^\ell(\mathbf{x})$, and the Jacobian of the entire composition of L functions by $J^{1:L}(\mathbf{x})$.

Because we've assumed that the GP on each output dimension $f_d(\mathbf{x}) \sim \mathcal{GP}$ is independent, it follows that for a given $\mathbf{x}$, each row of $J_{\mathbf{x} \to \mathbf{y}}(\mathbf{x})$ is independent. Above, we showed that the elements of each row are independent. This means that each entry in the Jacobian of a GP-distributed transform is i.i.d. Normal. $\square$

Theorem 3.3. *The Jacobian of a deep* GP *using a product kernel is a product of indepedent random Gaussian matrices.*

Proof. By the multivariate chain rule, the Jacobian of a composition of functions is simply the product of the Jacobian matrices of each function. Thus the Jacobian of the composed (deep) function $f^L(f^{L-1}(\ldots f^3(f^2(f^1(\mathbf{x}))) \ldots))$ is:

$$J^{1:L}(x) = J^L J^{L-1} \ldots J^3 J^2 J^1 \tag{9}$$

Since each $J^i \overset{\text{iid}}{\sim} \mathcal{N}(,)$, the complete Jacobian is a product of i.i.d. Gaussian matrices. $\square$

This result means that we can analyze the representational properties of a deep Gaussian process by simply examining the properties of products of i.i.d. Gaussian matrices.

4 Formalizing the pathologies

We follow Rifai et al. [2011b] in characterizing the representational properties of a function by the singular value spectrum of the Jacobian[1].

One-dimensional Asymptotics For a one-dimensional deep function, the derivative is simply a product of i.i.d Normal variables, with zero mean and variance $\frac{\sigma_f^2}{\ell^2}$. The distribution of the absolute value of the derivative of the deep function is a product of half-normals:

$$\frac{\partial f(x)}{\partial x} \overset{iid}{\sim} \mathcal{N}\left(0, \frac{\sigma_f^2}{\ell^2}\right) \implies \left|\frac{\partial f(x)}{\partial x}\right| \overset{iid}{\sim} \text{half}\mathcal{N}\left(\sqrt{\frac{2\sigma_f^2}{\pi\ell^2}}, \frac{\sigma_f^2}{\ell^2}\left(1 - \frac{2}{\pi}\right)\right) \tag{10}$$

Thus if $\frac{\sigma_f^2}{\ell_{d_1}^2} = \frac{\pi}{2}$, then $\mathbb{E}\left[\left|\frac{\partial f(x)}{\partial x}\right|\right] = 1$, and so $\mathbb{E}\left[\left|\frac{\partial f^{1:L}(x)}{\partial x}\right|\right] = 1$. If $\frac{\sigma_f^2}{\ell^2}$ is less than $\frac{\pi}{2}$, the expected derivative magnitude goes to zero, and if it's greater, then the expected magnitude goes to infinity.

The log of this variable has moments:

$$m_{\log} = \mathbb{E}\left[\log\left|\frac{\partial f(x)}{\partial x}\right|\right] = 2\log\left(\frac{\sigma_f}{\ell}\right) - \log 2 - \gamma \tag{11}$$

$$v_{\log} = \mathbb{V}\left[\log\left|\frac{\partial f(x)}{\partial x}\right|\right] = \frac{\pi^2}{4} + \frac{\log^2 2}{2} - \gamma^2 - \gamma\log 4 + 2\log\left(\frac{\sigma_f}{\ell}\right)\left[\gamma + \log 2 - \log\left(\frac{\sigma_f}{\ell}\right)\right] \tag{12}$$

[1]Rifai et al. [2011b] examine the Jacobian at the training points, but the models we are examining are stationary, so it doesn't matter where we examine the function.

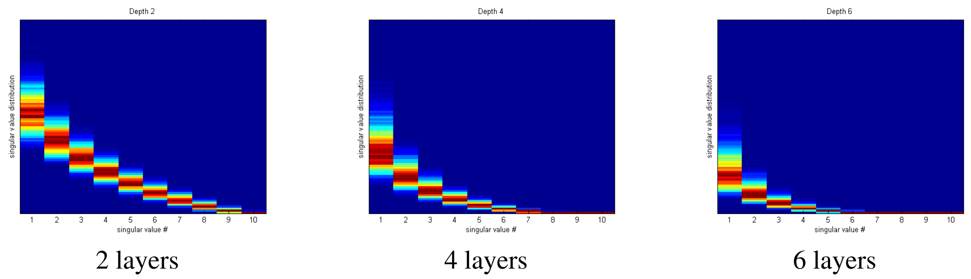


Figure 3: Singular value spectrum of the Jacobian of a deep GP. As the net gets deeper, the largest singular value dominates. This implies that there is only one effective degree of freedom in representation being computed.

Since the second moment is finite, by the central limit theorem, the limiting distribution of the size of the gradient is log-normal:

$$\log \left| \frac{\partial f^{1:L}(x)}{\partial x} \right| = \sum_{i=1}^{L} \log \left| \frac{\partial f^{L}(x)}{\partial x} \right| \tag{13}$$

$$\log \left| \frac{\partial f^{1:L}(x)}{\partial x} \right| \overset{L\to\infty}{\sim} \mathcal{N}\big(Lm_{\log}, L^2 v_{\log}\big) \tag{14}$$

Even if the expected magnitude of the derivative remains constant, the variance of the distribution grows without bound, and so will almost surely be very small almost everywhere, with rare but very large jumps. Note that there is another limit - if $\frac{\sigma_f^2}{\ell_{d_1}^2} < \frac{\pi}{2}$, then the mean of the size of the gradient goes to zero, but the variance approaches a finite constant.

With high probability, the regions of $f^{1:L}(\mathbf{x})$ which vary quickly will be the same regions in which $f^{1:L+1}(\mathbf{x})$ has high variance.

4.1 Jacobian Singular Value Spectrum

Figure 3 shows the spectrum for deep GPs of different depths. As the net gets deeper, the largest singular value dominates. This implies that there is only one effective degree of freedom in representation being computed. As well, the distribution on singular vlaues seems to become heavy-tailed.

Figure 4 demonstrates a related pathology that arises when composing functions to produce a deep latent-variable model. The density in observed space eventually becomes locally concentrated onto one-dimensional manifolds, or *filaments*.

To visualize this pathology in another way, figure 5 illustrates the value computed at each point in the input space, after successive warpings. After 40 warpings, we can see that locally, there is usually only one direction that one can move in $\mathbf{x}$-space in order to change the value of the function.

4.2 Fixing the pathology

Follow a suggestion in Neal [1995], we can fix the pathologies exhibited in figures 4 and 5 by simply making each layer of computation depend not only on the output of the previous layer, but also on the original input $\mathbf{x}$. Draws from the resulting priors are shown in figures 8 and 9.

Figure 6 shows draws from a 1D deep prior, both with naive compositions, and connected compositions.

Jacobians of connected deep networks We can similarily examine the Jacobians of the new connected architecture. The Jacobian of a composite function which has had the original inputs appended $f_{\text{aug}}(\mathbf{x}) = [f(\mathbf{x}), \mathbf{x}]$, is $\begin{bmatrix} J_f \\ I_D \end{bmatrix}$ and the Jacobian of all one-layer transformations of these augmented functions are $D \times 2D$ matrices. The Jacobian of the composed, connected deep function

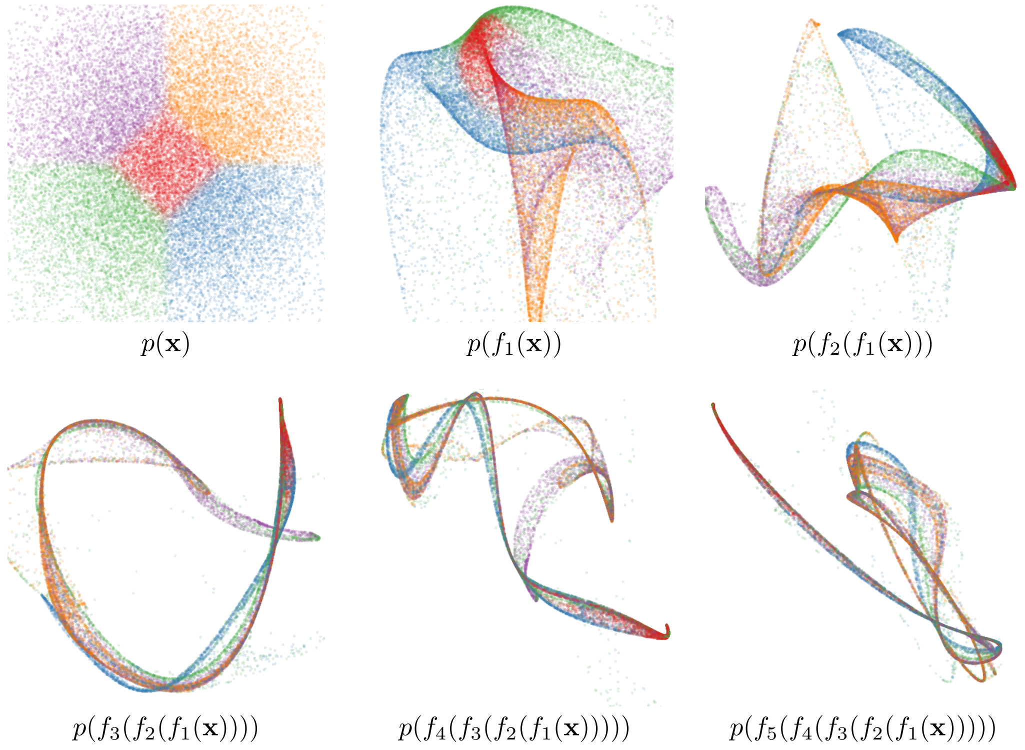


Figure 4: Draws from a deep GP. A distribution is warped by successive functions drawn from a GP prior. As the number of layers increases, the density exhibits a sort of filamentation.

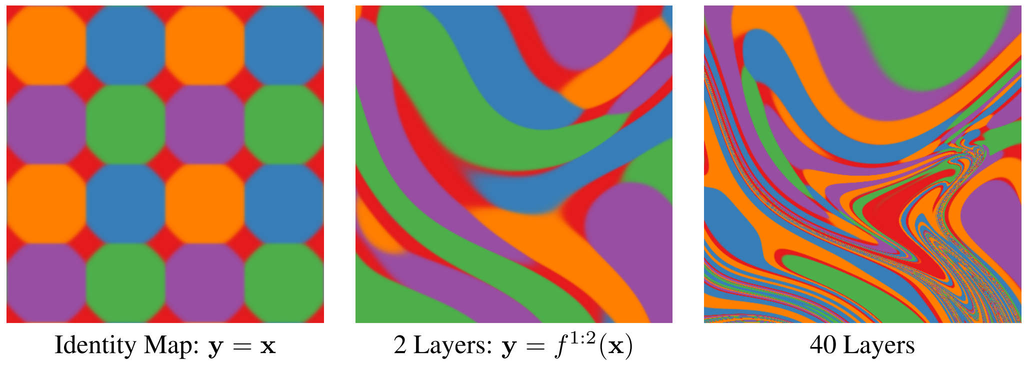


Figure 5: Feature Mapping of a deep GP. Shown here are the colors corresponding to the location $\mathbf{y} = f(\mathbf{x})$ that each point is mapped to after being warped by a deep GP. This figure can be seen as the inverse of figure 4. Just as the densities in 4 became locally one-dimensional, there is locally only one direction that one can move $\mathbf{x}$ in to change $\mathbf{y}$. Regions where the colors change rapidly indicate that the Jacobian is large in that area.

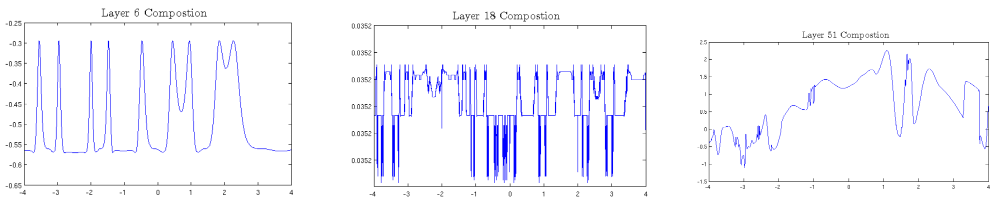


Figure 6: One-dimensional draws from deep Gaussian processes. The two left draws are from the naive compositional architecture, and rapidly become degenerate. The right-hand draw has every layer connected to the input, and maintains smoothness in some regions, while varying rapidly in other regions.

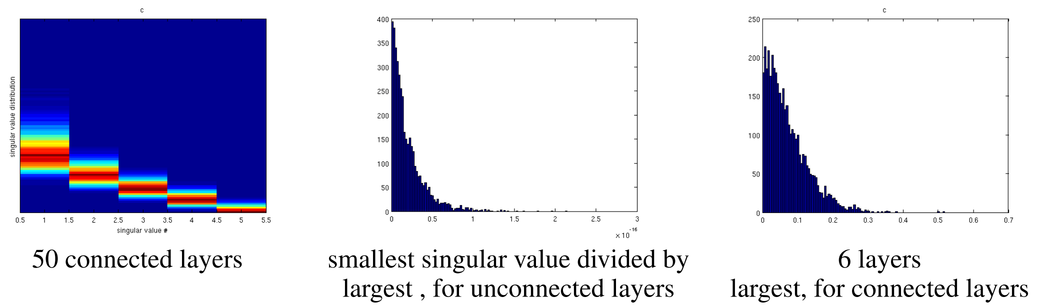


Figure 7: Reparing the singular value distribution. When each layer is connected to the original inputs, and the variance of derivatives is smaller than a threshold, then the ratio of the largest singular value to the smallest remains small.

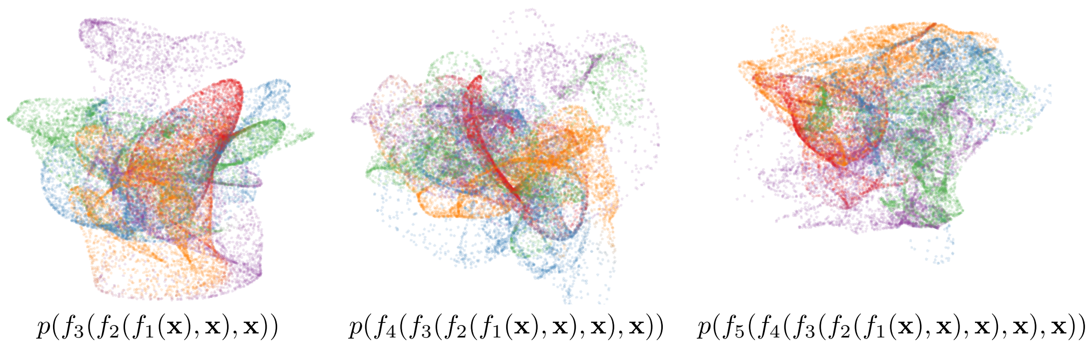


Figure 8: Draws from a deep GP, with each layer connected to the input $\mathbf{x}$. By always depending on the original input, the density becomes more complex without concentrating along filaments.

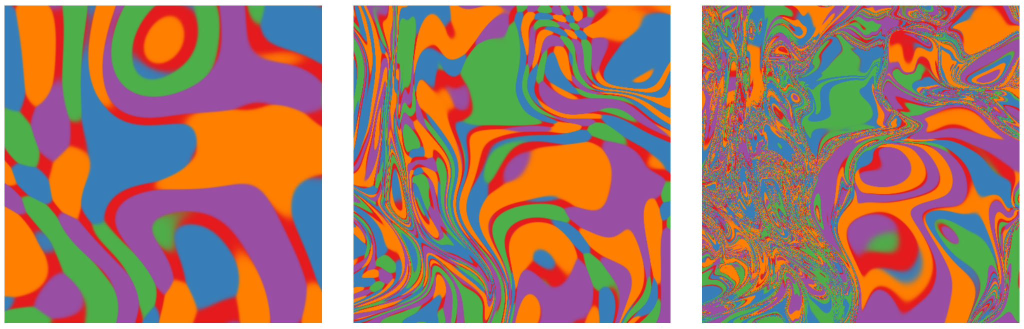


Figure 9: Feature Mapping of a deep GP with each layer connected to the input $\mathbf{x}$. Just as the densities in 8 became remain locally two-dimensional even after many transformations, in this mapping there are locally usually two directions that one can move $\mathbf{x}$ in to change $\mathbf{y}$.

is defined by the recurrence:

$$J^{1:L}(\mathbf{x}) = J^L \begin{bmatrix} J^{1:L-1} \\ I_D \end{bmatrix} \tag{15}$$

5 Arbitarily Deep Kernels

Cho [2012] investigated kernels constructed by applying multiple layers of feature mappings. That is to say, if a kernel has the form $k_1(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^\mathsf{T}\Phi(\mathbf{x})$, then we can consider constructing a kernel based on the repeated feature mapping: $k_2(\mathbf{x}, \mathbf{x}') = \Phi(\Phi(\mathbf{x}))^\mathsf{T}\Phi(\Phi(\mathbf{x}))$.

This composition operation has a closed form for any set of starting features $\Phi_n(\mathbf{x})$:

$$k_{n+1}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}||\Phi_n(\mathbf{x}) - \Phi_n(\mathbf{x}')||_2^2\right) \tag{16}$$

$$k_{n+1}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\sum_i \left[\phi_n^{(i)}(\mathbf{x}) - \phi_n^{(i)}(\mathbf{x}')\right]^2\right) \tag{17}$$

$$k_{n+1}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\left[k_n(\mathbf{x}, \mathbf{x}) - 2k_n(\mathbf{x}, \mathbf{x}') + k_n(\mathbf{x}', \mathbf{x}')\right]\right) \tag{18}$$

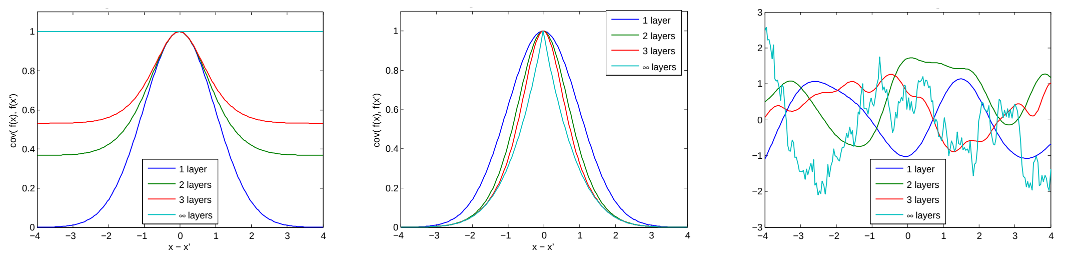
$$k_{n+1}(\mathbf{x}, \mathbf{x}') = \exp\left(k_n(\mathbf{x}, \mathbf{x}') - 1\right) \qquad (\text{if } k_n(\mathbf{x}, \mathbf{x}) = 1) \tag{19}$$

Note that nothing in this derivation depends on details of k_n, except that $k_n(\mathbf{x}, \mathbf{x}) = 1$.

5.1 Infinitely deep kernels

What happens when we apply this compsition many times, starting with the squared-exp kernel? In the infinite limit, this recursion converges to $k(x, y) = 1$ for all inputs. Figure **??** shows this kernel at different depths, including the degenerate limit.

One interpretation of why repeated feature transforms lead to this degenerate prior is that each layer can only lose information about the previous set of features. In the limit, the transformed features contain no information about the original input $\mathbf{x}$. Since the function doesn't depend on its input, it must be the same everywhere.



Iterated transform kernel connected transform kernel connected kernel GP draws

Figure 10: A non-degenerate version of the infinitely deep feature transform kernel. By connecting the inputs $\mathbf{x}$ to each layer, the function can still depend on its input even after arbitrarily many layers of computation.

5.2 A non-degenerate construction

Following a suggestion from Neal [1995], we append the inputs $\mathbf{x}$ to each layer of features. To do so, we simply by augmenting the feature vector $\Phi_n(\mathbf{x})$ with the extra features $\mathbf{x}$ at each layer:

$$k_{n+1}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\left|\left|\begin{bmatrix}\Phi_n(\mathbf{x})\\ \mathbf{x}\end{bmatrix} - \begin{bmatrix}\Phi_n(\mathbf{x}')\\ \mathbf{x}'\end{bmatrix}\right|\right|_2^2\right) \tag{20}$$

$$k_{n+1}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\sum_i \left[\phi_i(\mathbf{x}) - \phi_i(\mathbf{x}')\right]^2 - \frac{1}{2}||\mathbf{x} - \mathbf{x}'||_2^2\right) \tag{21}$$

$$k_{n+1}(\mathbf{x}, \mathbf{x}') = \exp\left(k_n(\mathbf{x}, \mathbf{x}') - 1 - \frac{1}{2}||\mathbf{x} - \mathbf{x}'||_2^2\right) \tag{22}$$

This kernel satisfies the recurrence $k - \log(k) = 1 + \frac{1}{2}||\mathbf{x} - \mathbf{x}'||_2^2$, a non-degenerate limit. The solution to this recurrence has no closed form, but it is continuous and differentiable everywhere except at $\mathbf{x} = \mathbf{x}'$. Samples from a GP with this prior are not differentiable, and locally resemble brownian motion, having a sort of fractal behavior.

5.3 Can "Deep Kernels" ever be powerful models?

Bengio et al. [2006] showed that kernel machines, such as GPs, 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. A periodic kernel can be viewed as a 2-layer "deep kernel" [cite Mackay?], in which the first layer maps the input $x \to [\sin(x), \cos(x)]$, and the second layer maps through a set of radial-basis functions.

Other methods for constructing kernels can capture many other types of structure Duvenaud et al. [2013], Wilson and Adams [2013], such as translation and rotation invariance in images Kondor [2008]. Salakhutdinov and Hinton [2008] even uses a deep neural network to construct feature transforms for kernels.

5.4 More general feature compositions

The failure of the kernels derived in this section to produce interesting computations suggest that deep computation is only powerful when it is combined with learning.

However, we now know how to map any kernel's feature mapping through an RBF feature mapping in closed form, and also how to append any other set of features to such a kernel. Thus we can now define kernels corresponding to arbitrary fixed computation graphs. This might be a nice way to extend kernel-search procedures.

6 Related Work

Several variants of deep models have been proposed, such as Bayesian Deep Networks Adams et al. [2010] and Sum-product networks Poon and Domingos [2011] These models also have no connections expect between adjacent layers, and perhaps may have similar pathologies as the number of layers increases.

Deep Density Networks Rippel and Adams [2013] were constructed with invertibility in mind, with parameters encouraged to preserve information about lower layers.

Recurrent Neural Networks Pascanu et al. [2012]

6.1 Initialization methods

Martens [2010] say: ”The best random initialization scheme we found was one of our own design, sparse initialization. In this scheme we hard limit the number of non-zero incoming connection weights to each unit (we used 15 in our experiments) and set the biases to 0 (or 0.5 for tanh units). Doing this allows the units to be both highly differentiated as well as unsaturated, avoiding the problem in dense initializations where the connection weights must all be scaled very small in order to prevent saturation, leading to poor differentiation between units”

The success of this initialization strategy suggests...

6.2 Related Analyses

Montavon et al. [2010] note that performance of a MLP degrades as the number of layers with random weights increases.

7 Discussion

To what extent are these pathologies present in nets being used today? In simulations, we found that for deep functions with a fixed latent dimension D, that the singular value spectrum remained relatively flat for hundreds of layers as long as $D > 100$.

Recursive learning method Just as layer-wise unsupervised pre-training encourages the projection of the data into a representation with independent features in the higher layers, so does the procedure outlined here. This is because the isotropic kernel does not penalize independence between different dimensions, only the number of dimensions.

8 Conclusions

- Deep neural networks and deep Gaussian processes are analyzable using random matrix theory.
- If you want to use very deep nets, you won't be able to do so if you initialize/regularize all your weights independently.
- If you initialize independently, the density becomes fractal.

Acknowledgments

We would like to thank Carl Rasmussen, Andrew McHutchon, Neil Lawrence, Josh Tenenbaum, Andreas Diamanadou, James Lloyd, Creighton Heaukulani, Dan Roy and Mark van der Wilk for helpful discussions.

References

Ryan P Adams, Hanna M Wallach, and Zoubin Ghahramani. Learning the structure of deep sparse graphical models. In *International Conference on Artificial Intelligence and Statistics*, pages 1–8, 2010.

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. The curse of highly variable functions for local kernel machines. pages 107–114, 2006.

Youngmin Cho. *Kernel methods for deep learning*. PhD thesis, University of California, San Diego, 2012.

Andreas C Damianou and Neil D Lawrence. Deep gaussian processes. *arXiv preprint arXiv:1211.0358*, 2012.

David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning*, June 2013.

Imre Risi Kondor. *Group theoretical methods in machine learning*. 2008.

N.D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *Advances in Neural Information Processing Systems*, 16:329–336, 2004.

N.D. Lawrence and R. Urtasun. Non-linear matrix factorization with Gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 601–608. ACM, 2009.

James Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742, 2010.

Grégoire Montavon, Dr Braun, Klaus-Robert Müller, et al. Layer-wise analysis of deep networks with gaussian kernels. *Advances in Neural Information Processing Systems (NIPS)*, 23:1678–1686, 2010.

Radford M Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *arXiv preprint arXiv:1211.5063*, 2012.

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 689–690. IEEE, 2011.

Salah Rifai, Grégoire Mesnil, Pascal Vincent, Xavier Muller, Yoshua Bengio, Yann Dauphin, and Xavier Glorot. Higher order contractive auto-encoder. In *Machine Learning and Knowledge Discovery in Databases*, pages 645–660. Springer, 2011a.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 833–840, 2011b.

Oren Rippel and Ryan Prescott Adams. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, 2013.

Ruslan Salakhutdinov and Geoffrey Hinton. Using deep belief nets to learn covariance kernels for Gaussian processes. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

M. Salzmann, R. Urtasun, and P. Fua. Local deformation models for monocular 3D shape recovery. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, pages 1–8, 2008.

Ercan Solak, Roderick Murray-Smith, E. Solak, William Leithead, Carl Rasmussen, and Douglas Leith. Derivative observations in gaussian process models of dynamic systems, 2003.

Andrew G. Wilson and Ryan P. Adams. Gaussian process covariance kernels for pattern discovery and extrapolation. *arXiv: 1302.4245*, 2013.