Non-degenerate Priors for Arbitrarily Deep Networks

Anonymous Author(s)
Affiliation
Address
email

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

Deep networks have become an important tool for machine learning [cite]. However, training these models are difficult, Many arguments have been made for the need for deep architectures [cite Bengio]. However, it is hard to know what effect the deepness of an architecture has. Also, the weights don't necessarily move that much from their initialization.

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.

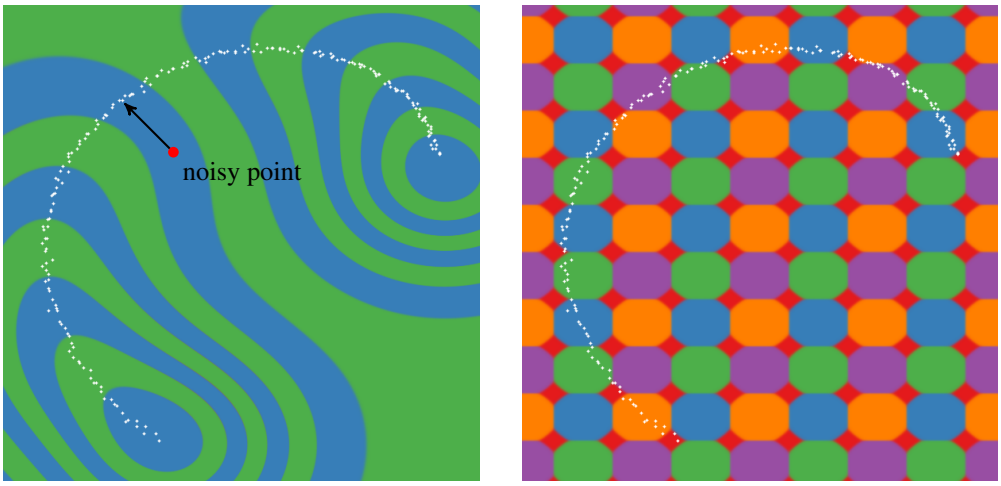
1.1 Desirable properties of latent representations

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 1 demonstrates this idea.

2 The Jacobian of Deep GPs

Deep Gaussian Processes We introduce a generative non-parametric model to address this problem. Our approach is based on the GP-LVM Lawrence [2004], Salzmann et al. [2008], Lawrence and Urtasun [2009], a flexible nonparametric density model. Deep Gaussian processes Damianou and Lawrence [2012].

The derivatives of a function drawn from a GP prior with a product kernel are i.i.d. Normal Because differentiation is a linear operator, the derivatives of a function drawn from a GP prior are also jointly Gaussian distributed, with covariance between derivatives w.r.t. different dimensions of



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

A naïve latent representation of a one-dimensional manifold

Figure 1: Comparing different latent representations of data on a 1-D manifold. A good latent representation is invariant in directions orthogonal to the data manifold, but changes along the manifold. The representation on the right might be useful if the data were spread out in over plane.

$\mathbf{x}$ given by:

$$\text{cov}\left(\frac{\partial f(\mathbf{x})}{\partial x_{d_1}}, \frac{\partial f(\mathbf{x})}{\partial x_{d_2}}\right) = \frac{\partial^2 k(\mathbf{x}, \mathbf{x}')}{\partial x_{d_1} \partial x'_{d_2}}\bigg|_{\mathbf{x}=\mathbf{x}'} \tag{1}$$

[Solak et al., 2003]

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. This means that elements are independent and identically distributed.

The Jacobian of a GP with a seperable kernel is a matrix of i.i.d. Gaussian R.V.'s The Jacobian of the ℓ^{th}function is:

$$J^{\ell}_{\mathbf{x}\to\mathbf{y}}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1^{\ell}(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1^{\ell}(\mathbf{x})}{\partial x_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_D^{\ell}(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_D^{\ell}(\mathbf{x})}{\partial x_D} \end{bmatrix} \tag{2}$$

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.

The Jacobian of a deep GP is a product of random normal matrices By the multivariate chain rule, the derivative (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 $f^L(f^{L-1}(\dots f^3(f^2(f^1(\mathbf{x})))\dots))$ is:

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

Combining these results, we can analyze the representational properties of a deep Gaussian process by simply examining the properties of products of i.i.d. Gaussian matrices.

2.1 Formalizing the pathology

We formalize the pathology in two ways: One, by the distribution of the magnitude of derivatives, and two, by the relative magnitudes of singular vectors of the Jacobian.

2.2 Example prior: isotropic squared-exp kernel

In the particular case of the squared-exp kernel, $k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \prod_{d=1}^{D} \left(-\frac{1}{2}\frac{(x_d - x'_d)^2}{\ell_d^2}\right)$, the partial second derivatives have the form:

$$\left.\frac{\partial^2 k_{SE}(\mathbf{x}, \mathbf{x}')}{\partial x_{d_1} \partial x'_{d_2}}\right|_{\mathbf{x}=\mathbf{x}'} = \begin{cases} \frac{\sigma_f^2}{\ell_{d_1}^2} & \text{if } d_1 = d_2 \\ 0 & \text{if } d_1 \neq d_2 \end{cases} \tag{4}$$

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:

$$\frac{\partial f(x)}{\partial x} \overset{iid}{\sim} \mathcal{N}\left(0, \frac{\sigma_f^2}{\ell^2}\right) \tag{5}$$

$$\left|\frac{\partial f(x)}{\partial x}\right| \overset{iid}{\sim} \text{half}\mathcal{N}\left(\sqrt{\frac{2\sigma_f^2}{\pi\ell^2}}, \frac{\sigma_f^2}{\ell^2}\left(1 - \frac{2}{\pi}\right)\right) \tag{6}$$

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:

$$m_{\log} = \mathbb{E}\left[\log\left|\frac{\partial f(x)}{\partial x}\right|\right] = 2\log\left(\frac{\sigma_f}{\ell}\right) - \log 2 - \gamma \tag{7}$$

$$v_{\log} = \mathbb{V}\left[\log\left|\frac{\partial f(x)}{\partial x}\right|\right] = \frac{\pi^2}{4} + \frac{\log^2 2}{2} - \gamma^2 - \gamma\log 4 + 2\log\left(\frac{\sigma_f}{\ell}\right)\left[\gamma + \log 2 - \log\left(\frac{\sigma_f}{\ell}\right)\right] \tag{8}$$

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

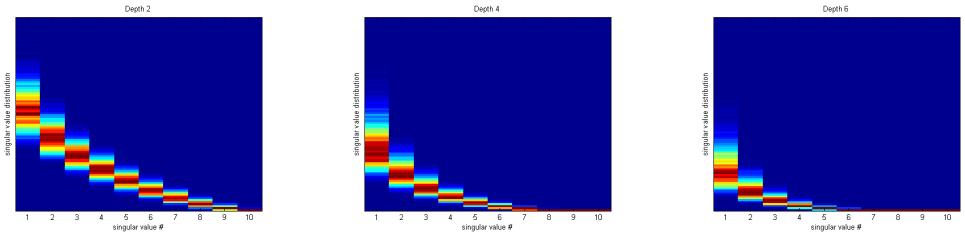
$$\log\left|\frac{\partial f^{1:L}(x)}{\partial x}\right| = \sum_{i=1}^{L} \log\left|\frac{\partial f^{L}(x)}{\partial x}\right| \tag{9}$$

$$\log\left|\frac{\partial f^{1:L}(x)}{\partial x}\right| \overset{L\to\infty}{\sim} \mathcal{N}\left(Lm_{\log}, L^2 v_{\log}\right) \tag{10}$$

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.

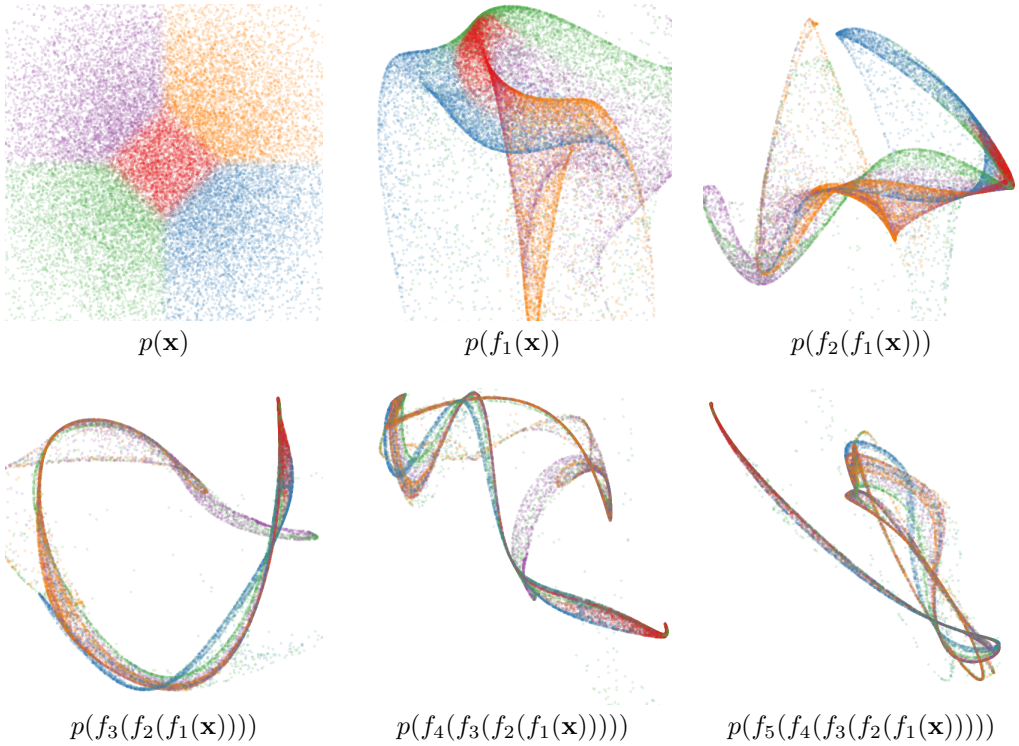
With high probability, the regions of $f^{1:L}(\mathbf{x})$ which vary quickly will be the same regions in which $f^{1:L+1}(\mathbf{x})$ has high variance.

[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.



2 layers 4 layers 6 layers

Figure 2: 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.



$p(\mathbf{x})$ $p(f_1(\mathbf{x}))$ $p(f_2(f_1(\mathbf{x})))$

$p(f_3(f_2(f_1(\mathbf{x}))))$ $p(f_4(f_3(f_2(f_1(\mathbf{x})))))$ $p(f_5(f_4(f_3(f_2(f_1(\mathbf{x}))))))$

Figure 3: 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.

2.3 Jacobian Singular Value Spectrum

Figure 2 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 3 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 4 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.

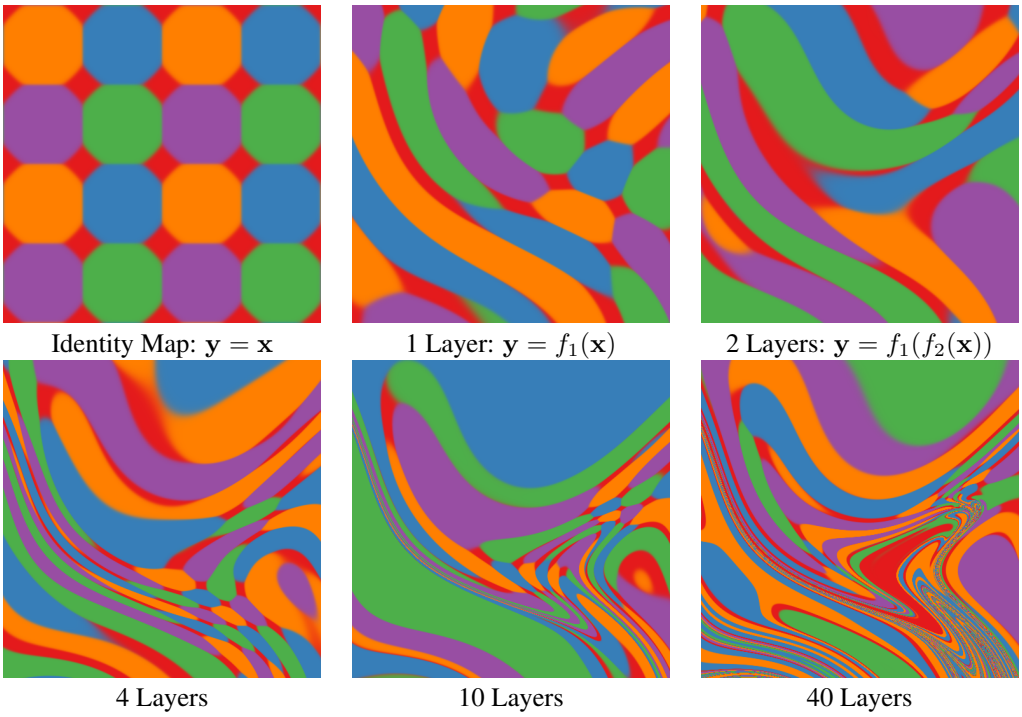


Figure 4: Feature Mapping of a deep GP. Shown here are the colors corresponding to the location $\mathbf{y} = f(\mathbf{x})$ that each point is mapped to after being warped by a deep GP. This figure can be seen as the inverse of figure 3. Just as the densities in 3 became locally one-dimensional, there 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.

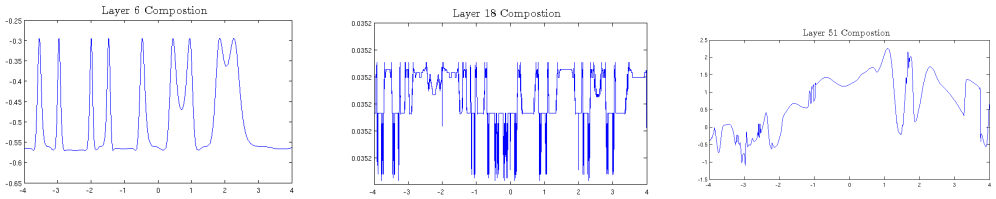
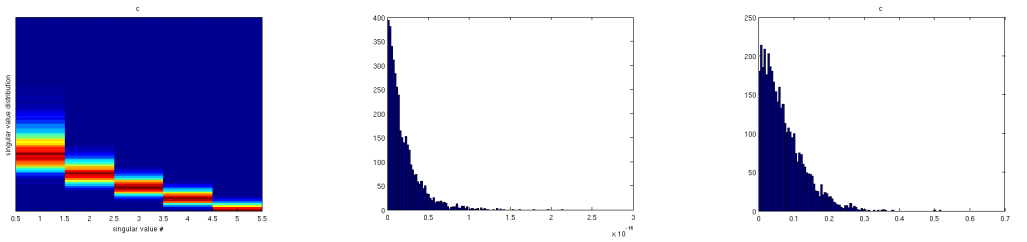


Figure 5: One-dimensional draws from deep Gaussian processes. The two left draws are from the naive compositional architecture, and rapidly become degenerate. The right-hand draw has every layer connected to the input, and maintains smoothness in some regions, while varying rapidly in other regions.



50 connected layers

smallest singular value divided by largest , for unconnected layers

6 layers largest, for connected layers

Figure 6: 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.

2.4 Fixing the pathology

Follow a suggestion in Neal [1995], we can fix the pathologies exhibited in figures 3 and 4 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 7 and 8.

Figure 5 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 is defined by the recurrence:

$$J^{1:L}(\mathbf{x}) = J^L \begin{bmatrix} J^{1:L-1} \\ I_D \end{bmatrix} \tag{11}$$

So the entire Jacobian has the form:

$$J^{1:L}(x) = J^L \begin{bmatrix} J^{L-1} \begin{bmatrix} \dots J^4 \begin{bmatrix} J^3 \begin{bmatrix} J^2 J^1 \\ I_D \end{bmatrix} \\ I_D \end{bmatrix} \dots \\ \vdots \\ I_D \end{bmatrix} \end{bmatrix} \tag{12}$$

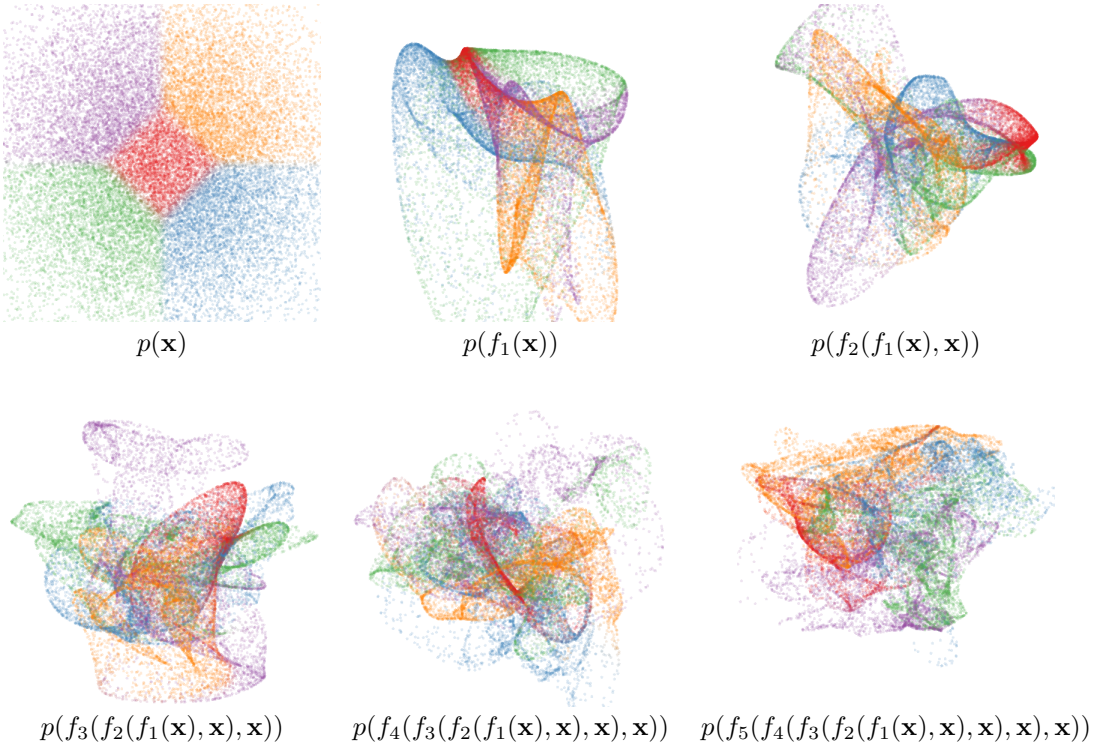


Figure 7: Draws from a deep GP, with each layer connected to the input $\mathbf{x}$. By always depending on the original input, the density becomes more complex without concentrating along filaments.

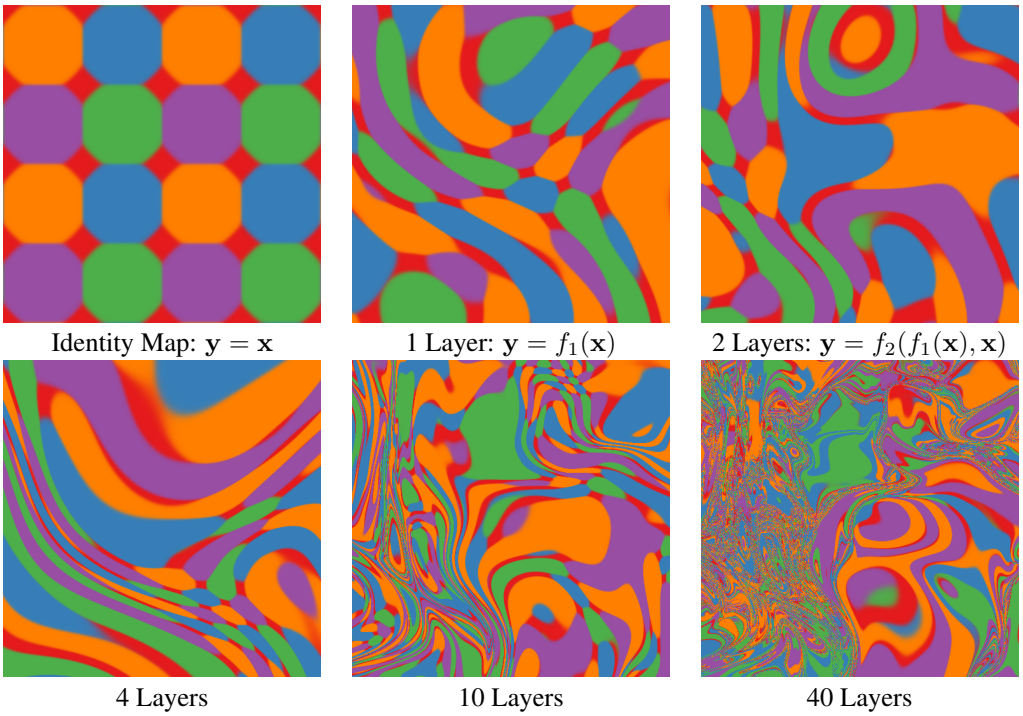
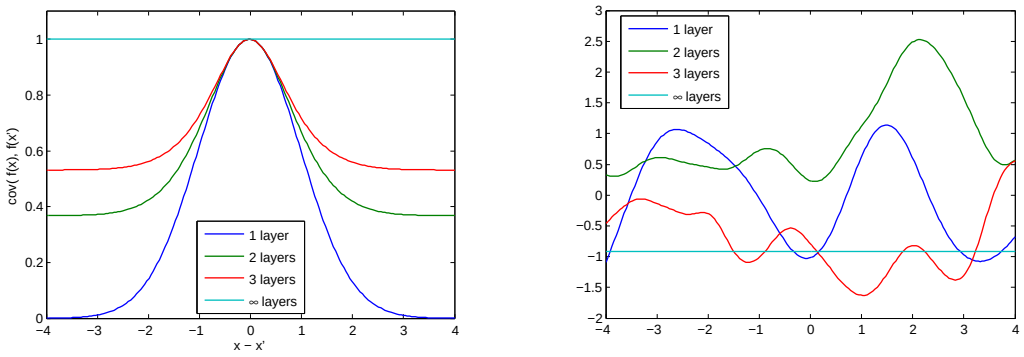


Figure 8: Feature Mapping of a deep GP with each layer connected to the input $\mathbf{x}$. Just as the densities in 7 became remain locally two-dimensional even after many transformations, in this mapping there are locally usually two directions that one can move $\mathbf{x}$ in to change $\mathbf{y}$.



Kernel derived from iterated feature transforms

Draws from the corresponding kernel

Figure 9: A degenerate kernel produced by repeatedly applying a feature transform.

3 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.

We can imagine first learning a mapping $\mathbf{y} = f_1(\mathbf{x})$.

4 Arbitarily Deep Kernels

These types of kernels were originally investigated by Cho [2012]. In this section, we take the infinite limits of these compositions, and propose a new variant.

One can derive a Gaussian process as a neural network: $f(x) = \alpha^T \Phi(x) = \sum_{i=i}^{K} \alpha_i \phi_i(x)$. We can consider applying the feature transform $\Phi(\cdot)$ to the features themselves: $\Phi_2 = \Phi(\Phi(\mathbf{x}))$,

In this section, we derive a kernel which corresponds to arbitrarily many compositions of the feature vectors corresponding to the squared-exp kernel:

$$k_2(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}||\Phi(\mathbf{x}) - \Phi(\mathbf{x}')||_2^2\right) \tag{13}$$

$$k_2(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\sum_i \left[\phi_i(\mathbf{x}) - \phi_i(\mathbf{x}')\right]^2\right) \tag{14}$$

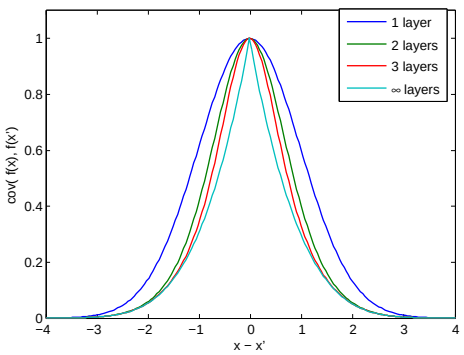
$$k_2(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\sum_i \left[\phi_i(\mathbf{x})^2 - 2\phi_i(\mathbf{x})\phi_i(\mathbf{x}') + \phi_i(\mathbf{x}')^2\right]\right) \tag{15}$$

$$k_2(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\left[\sum_i \phi_i(\mathbf{x})^2 - 2\sum_i \phi_i(\mathbf{x})\phi_i(\mathbf{x}') + \sum_i \phi_i(\mathbf{x}')^2\right]\right) \tag{16}$$

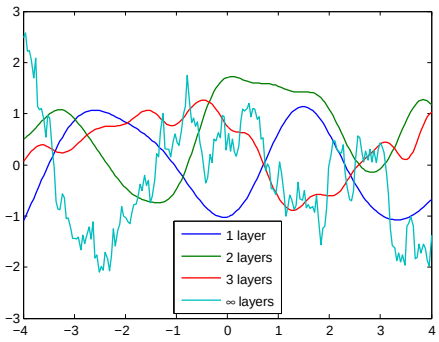
$$k_2(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\left[k_1(\mathbf{x}, \mathbf{x}) - 2k_1(\mathbf{x}, \mathbf{x}') + k_1(\mathbf{x}', \mathbf{x}')\right]\right) \tag{17}$$

$$k_2(\mathbf{x}, \mathbf{x}') = \exp\left(k_1(\mathbf{x}, \mathbf{x}') - 1\right) \tag{18}$$

Thus if $k_1(x, y) = e^{-||x-y||2}$, then the two-layer kernel is simply $k_2(x, y) = e^{k_1(x,y)-1}$. This formula is true for every layer: $k_{n+1}(x, y) = e^{k_n(x,y)-1}$. Note that nothing in this derivation depends on details of k_1, except that $k_1(\mathbf{x}, \mathbf{x}) = 1$. Because this is true for k_2 as well, this recursion holds in general, and we have that $k_{n+1}(x, y) = e^{k_n(x,y)-1}$. In the infinite limit, this recursion converges to $k(x, y) = 1$ for all inputs.



Kernel derived from iterated feature transforms with all layers connected to the input



Draws from the corresponding kernel

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.

Figure 9 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.

4.1 Fixing the deep kernel

Follow a suggestion from Radford Neal's thesis Neal [1995], we connect the inputs to each layer of features. We do this simply by augmenting the feature vector $\Phi_n(\mathbf{x})$ with the extra features $\mathbf{x}$ at every layer:

$$k_{n+1}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\left|\left|\begin{bmatrix}\Phi_n(\mathbf{x})\\ \mathbf{x}\end{bmatrix} - \begin{bmatrix}\Phi_n(\mathbf{x}')\\ \mathbf{x}'\end{bmatrix}\right|\right|_2^2\right) \tag{19}$$

$$k_{n+1}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\sum_i \left[\phi_i(\mathbf{x}) - \phi_i(\mathbf{x}')\right]^2 - \frac{1}{2}||\mathbf{x} - \mathbf{x}'||_2^2\right) \tag{20}$$

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

Thus, this kernel satisfies the recurrence $k - \log(k) = 1 + \frac{1}{2}||\mathbf{x} - \mathbf{x}'||_2^2$.

Properties of this kernel The solution to this recurrence has no closed form, but it is continuous and differentiable everywhere except at $\mathbf{x} = \mathbf{x}'$.

Conjectures:

- Samples from a GP with this prior are not differentiable.
- This kernel has smaller correlation than the squared-exp everywhere except at $\mathbf{x} = \mathbf{x}'$.
- Samples from this kernel are fractal.
- The tails have the same form as the squared-exp.

4.2 More general feature compositions

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.

4.3 A fully-connected kernel

However, connecting every layer to every subsequent layer leades to a pathology:

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

$$k_{n+1}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\sum_i\left[\phi_n^{(i)}(\mathbf{x}) - \phi_n^{(i)}(\mathbf{x}')\right]^2 - \frac{1}{2}\sum_i\left[\phi_{n-1}^{(i)}(\mathbf{x}) - \phi_{n-1}^{(i)}(\mathbf{x}')\right]^2\right) \tag{23}$$

$$k_{n+1}(\mathbf{x}, \mathbf{x}') = \exp\left(k_n(\mathbf{x}, \mathbf{x}') - 1\right)\exp\left(k_{n-1}(\mathbf{x}, \mathbf{x}') - 1\right) \tag{24}$$

$$k_{n+1}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}||\mathbf{x} - \mathbf{x}'||_2^2\right)\prod_{i=1}^{n}\exp\left(k_i(\mathbf{x}, \mathbf{x}') - 1\right) \tag{25}$$

Which has the solution $k_\infty = \delta(\mathbf{x} = \mathbf{x}')$, a white noise kernel.

5 Related Work

5.1 A Survey of deep architectures

Deep Density Networks [cite Oren's paper]

Bayesian Deep Networks Adams et al. [2010] developed a prior on finite but unboundedly deep neural networks, each layer having a finite but unbounded number of hidden units.

Sum-product networks introducted by Poon and Domingos [2011].

Feature Composition Kernels Cho [2012] developed kernels of the type discussed in section 4, and investigated them experimentally.

Recurrent Neural Networks [Cite Bengio's work]

5.2 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

Dynamical Systems These architectures are all constructed in a stacked manner, with connections only between adjacent layers.

[Warping a 1d uniform distribution]

5.3 Related Analyses

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

6 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 $D > 100$.

7 Conclusions

Deep neural networks and deep Gaussian processes are analyzable using random matrix theory. After proving that the Jacobian is an i.i.d. Gaussian matrix, many other forms of

If you want to use very deep nets, you won't be able to do so if you initialize/regularize all your weights independently We might want to think about different ways of

If you initialize independently, the density becomes fractal Points close in x-space can be very far in y-space, and vice versa.

A spikey eigenspecturm will lead to saturation Maybe we should initialize differently in order to avoid such saturation, like Martens' sparse initialization: http://www.cs.toronto.edu/~jmartens/docs/Deep_HessianFree.pdf

Acknowledgments

We would like to thank Andrew McHutchon, Neil Lawrence, Josh Tenenbaum, Andreas Diamanadou, James Lloyd, 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.

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.

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.

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.

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.