Deep Gaussian Processes

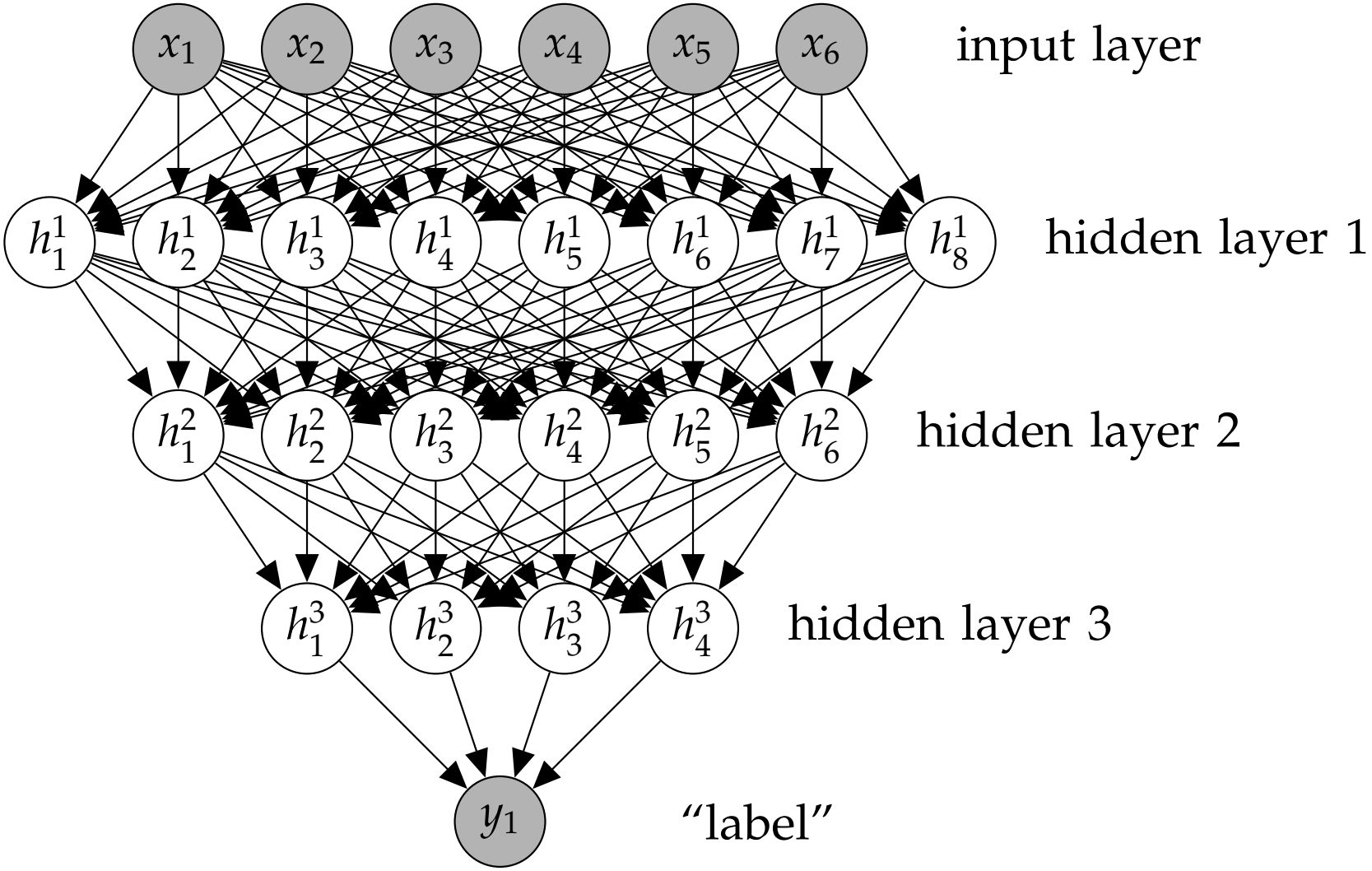
Neil D. Lawrence

23rd January 2015

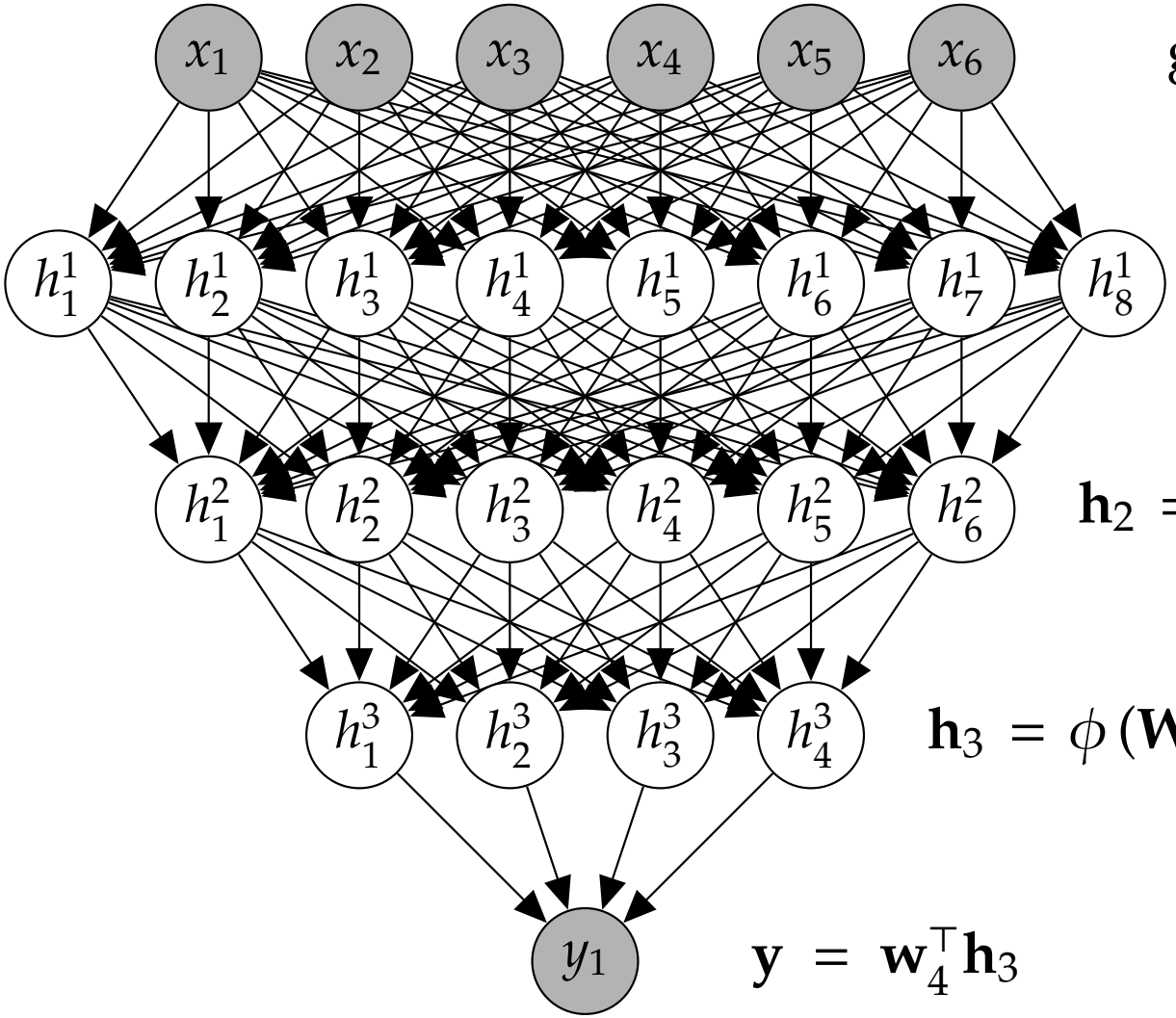
Outline

Outline

Deep Neural Network



Deep Neural Network



given $\mathbf{x}$

$$\mathbf{h}_1 = \phi(\mathbf{W}_1\mathbf{x})$$

$$\mathbf{h}_2 = \phi(\mathbf{W}_2\mathbf{h}_1)$$

$$\mathbf{h}_3 = \phi(\mathbf{W}_3\mathbf{h}_2)$$

$$\mathbf{y} = \mathbf{w}_4^\top \mathbf{h}_3$$

Mathematically

$$
\begin{aligned}
\mathbf{h}_1 &= \phi\left(\mathbf{W}_1 \mathbf{x}\right) \\
\mathbf{h}_2 &= \phi\left(\mathbf{W}_2 \mathbf{h}_1\right) \\
\mathbf{h}_3 &= \phi\left(\mathbf{W}_3 \mathbf{h}_2\right) \\
\mathbf{y} &= \mathbf{w}_4^\top \mathbf{h}_3
\end{aligned}
$$

Overfitting

- Potential problem: if number of nodes in two adjacent layers is big, corresponding $\mathbf{W}$ is also very big and there is the potential to overfit.
- Proposed solution: "dropout".
- Alternative solution: parameterize $\mathbf{W}$ with its SVD.

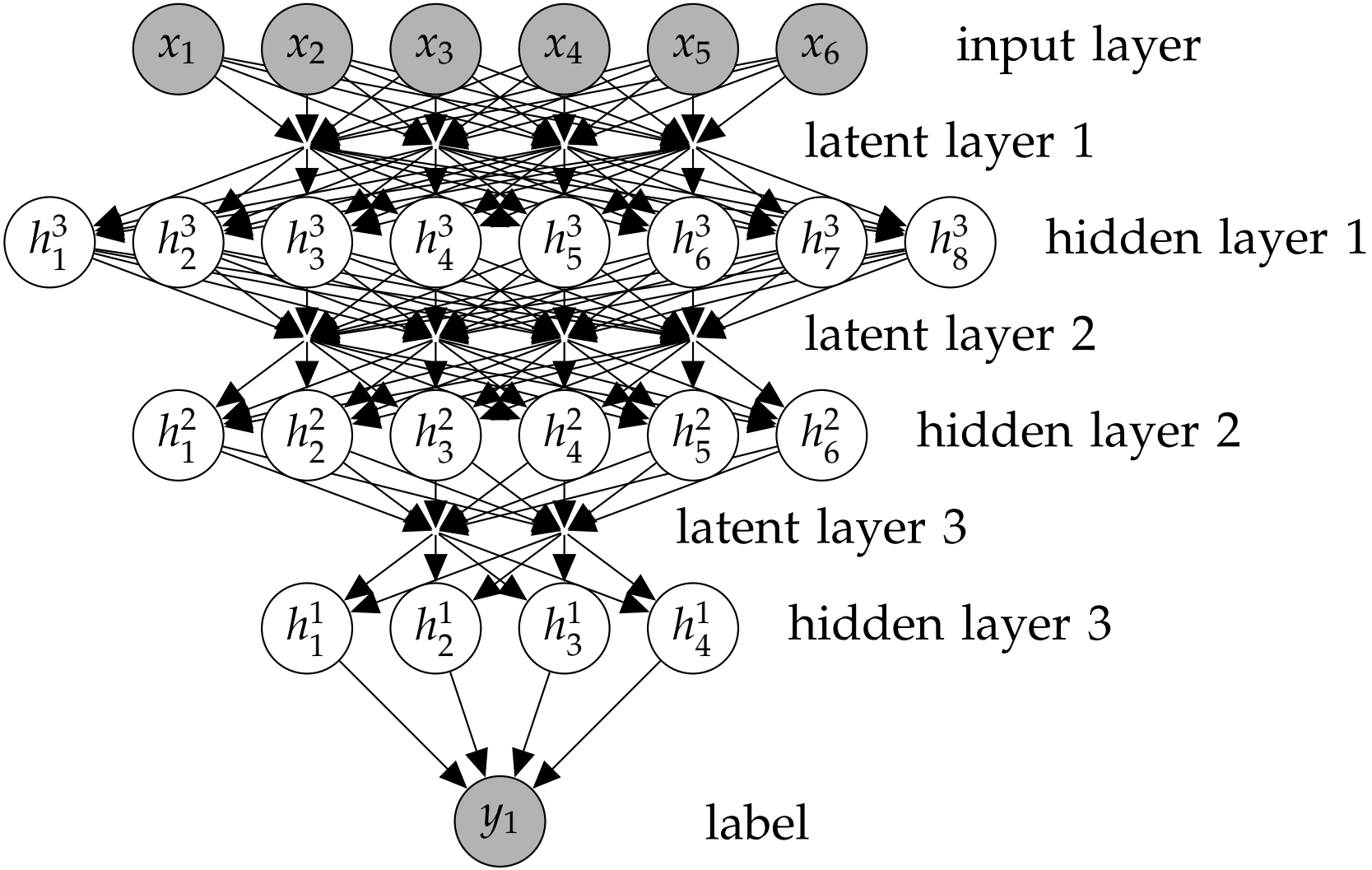
$$\mathbf{W} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^\top$$

or

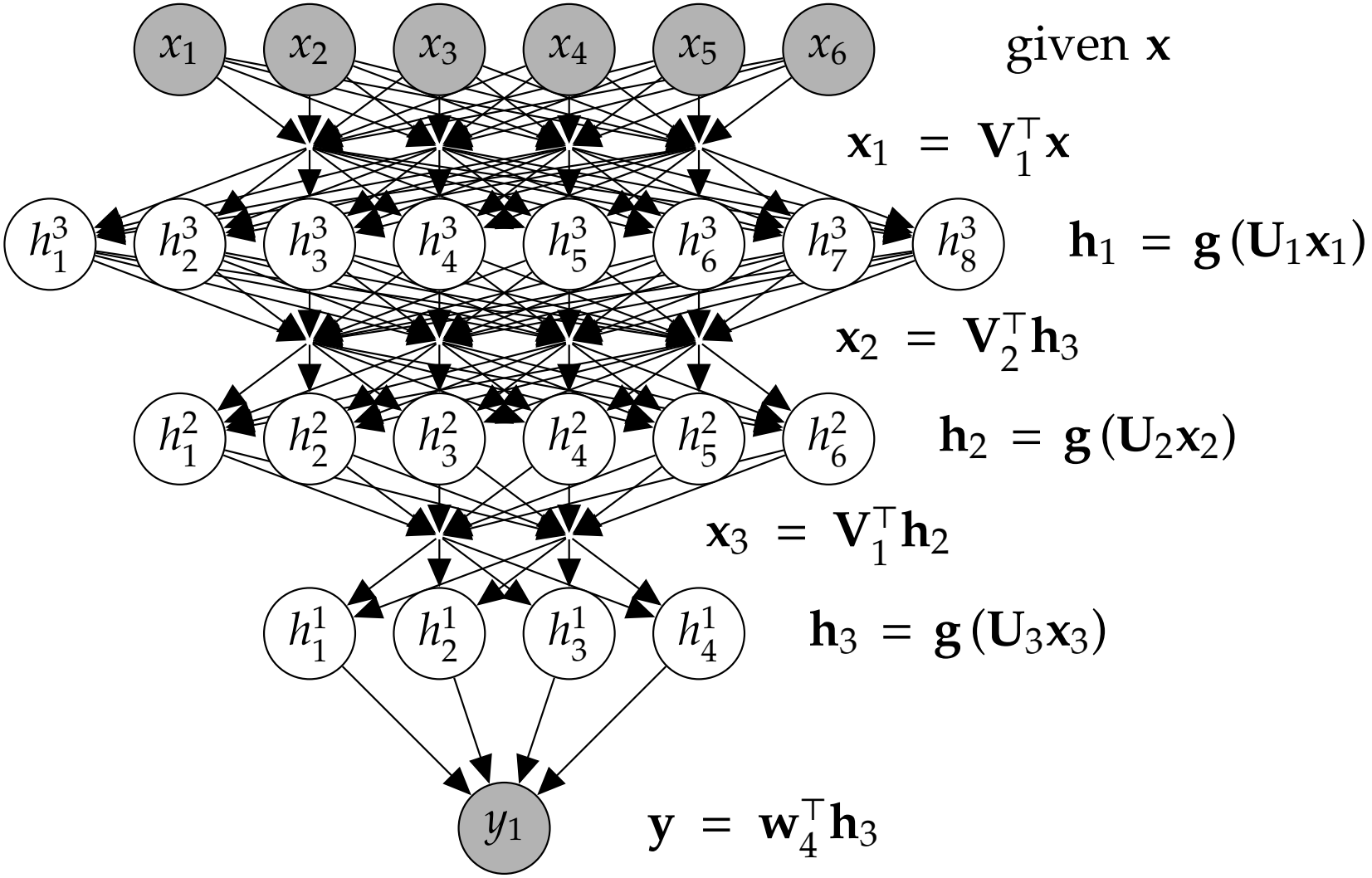
$$\mathbf{W} = \mathbf{U}\mathbf{V}^\top$$

where if $\mathbf{W} \in \Re^{k_1 \times k_2}$ then $\mathbf{U} \in \Re^{k_1 \times q}$ and $\mathbf{V} \in \Re^{k_2 \times q}$, i.e. we have a low rank matrix factorization for the weights.

Deep Neural Network



Deep Neural Network



Mathematically

$$
\begin{aligned}
\mathbf{x}_1 &= \mathbf{V}_1^\top \mathbf{x} \\
\mathbf{h}_1 &= \phi\left(\mathbf{U}_1 \mathbf{x}_1\right) \\
\mathbf{x}_2 &= \mathbf{V}_2^\top \mathbf{h}_1 \\
\mathbf{h}_2 &= \phi\left(\mathbf{U}_2 \mathbf{x}_2\right) \\
\mathbf{x}_3 &= \mathbf{V}_3^\top \mathbf{h}_2 \\
\mathbf{h}_3 &= \phi\left(\mathbf{U}_3 \mathbf{x}_3\right) \\
\mathbf{y} &= \mathbf{w}_4^\top \mathbf{h}_3
\end{aligned}
$$

A Cascade of Neural Networks

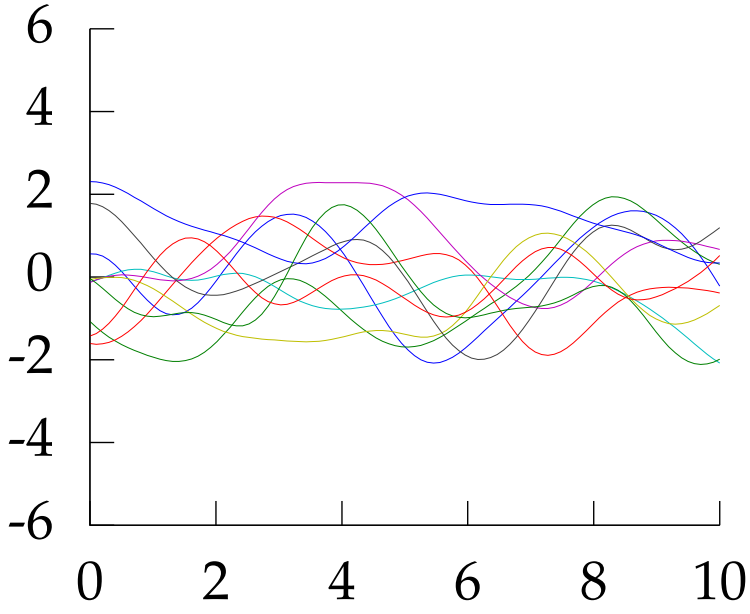
$$
\begin{aligned}
\mathbf{x}_1 &= \mathbf{V}_1^\top \mathbf{x} \\
\mathbf{x}_2 &= \mathbf{V}_2^\top \phi\left(\mathbf{U}_1 \mathbf{x}_1\right) \\
\mathbf{x}_3 &= \mathbf{V}_3^\top \phi\left(\mathbf{U}_2 \mathbf{x}_2\right) \\
\mathbf{y} &= \mathbf{w}_4^\top \mathbf{x}_3
\end{aligned}
$$

Replace Each Neural Network with a Gaussian Process

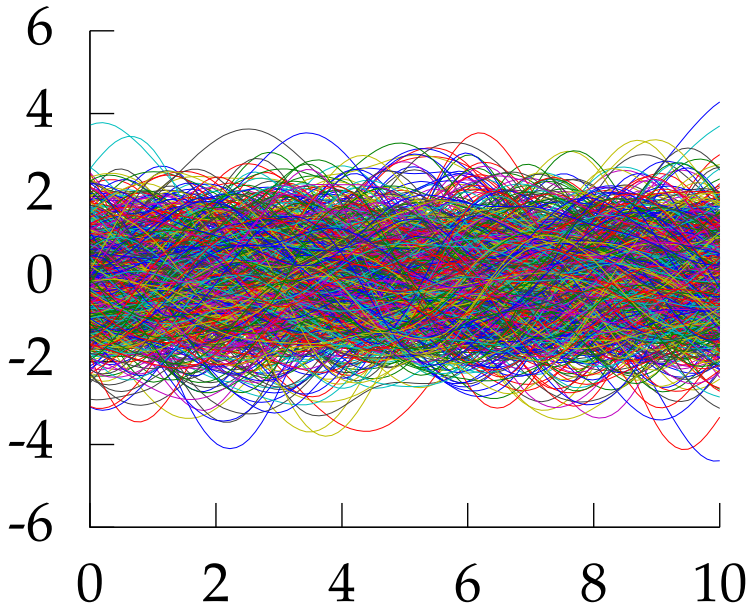
$$\mathbf{x}_1 = \mathbf{f}(\mathbf{x})$$
$$\mathbf{x}_2 = \mathbf{f}(\mathbf{x}_1)$$
$$\mathbf{x}_3 = \mathbf{f}(\mathbf{x}_2)$$
$$\mathbf{y} = \mathbf{f}(\mathbf{x}_3)$$

This is equivalent to Gaussian prior over weights and integrating out all parameters and taking width of each layer to infinity.

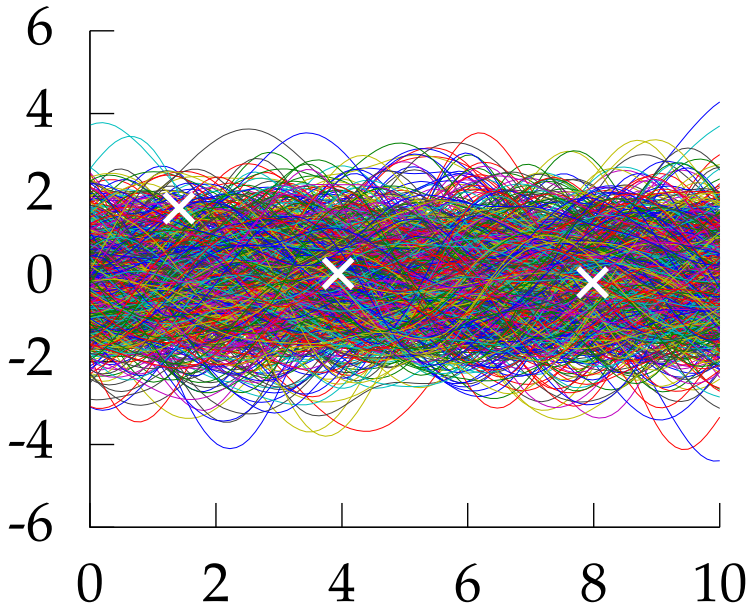
Gaussian Processes: Extremely Short Overview



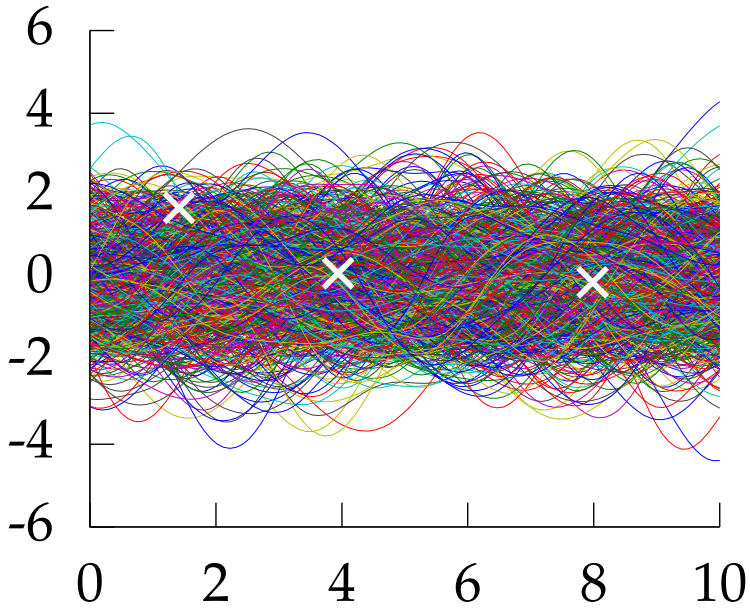
Gaussian Processes: Extremely Short Overview

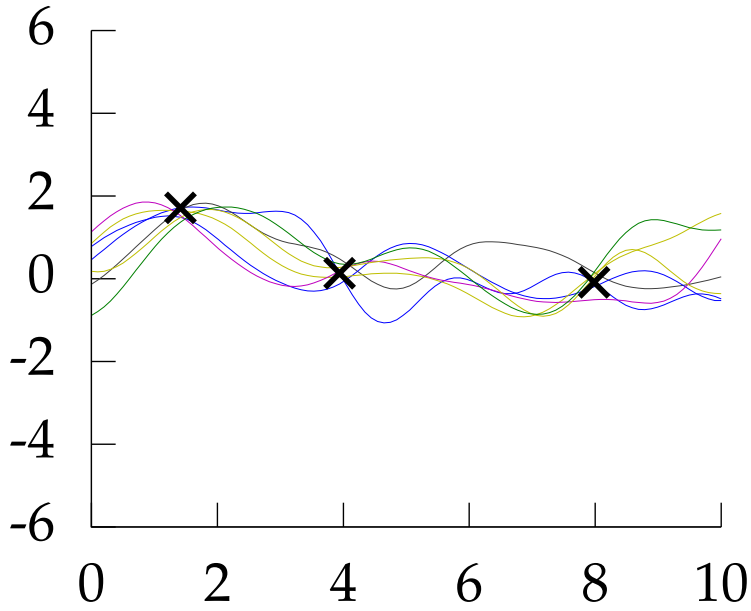


Gaussian Processes: Extremely Short Overview



Gaussian Processes: Extremely Short Overview





Outline

Mathematically

- Composite *multivariate* function

$$\mathbf{g}(\mathbf{x}) = \mathbf{f}_5(\mathbf{f}_4(\mathbf{f}_3(\mathbf{f}_2(\mathbf{f}_1(\mathbf{x})))))$$

Why Deep?

- Gaussian processes give priors over functions.
- Elegant properties:
  - e.g. *Derivatives* of process are also Gaussian distributed (if they exist).
- For particular covariance functions they are ‘universal approximators’, i.e. all functions can have support under the prior.
- Gaussian derivatives might ring alarm bells.
- E.g. a priori they don’t believe in function ‘jumps’.

Process Composition

- From a process perspective: *process composition*.
- A (new?) way of constructing more complex *processes* based on simpler components.

Note: To retain *Kolmogorov consistency* introduce IBP priors over latent variables in each layer (Zhenwen Dai).

Analysis of Deep GPs

- Duvenaud et al. (2014) Duvenaud et al show that the derivative distribution of the process becomes more *heavy tailed* as number of layers increase.

Difficulty for Probabilistic Approaches

- Propagate a probability distribution through a non-linear mapping.
- Normalisation of distribution becomes intractable.

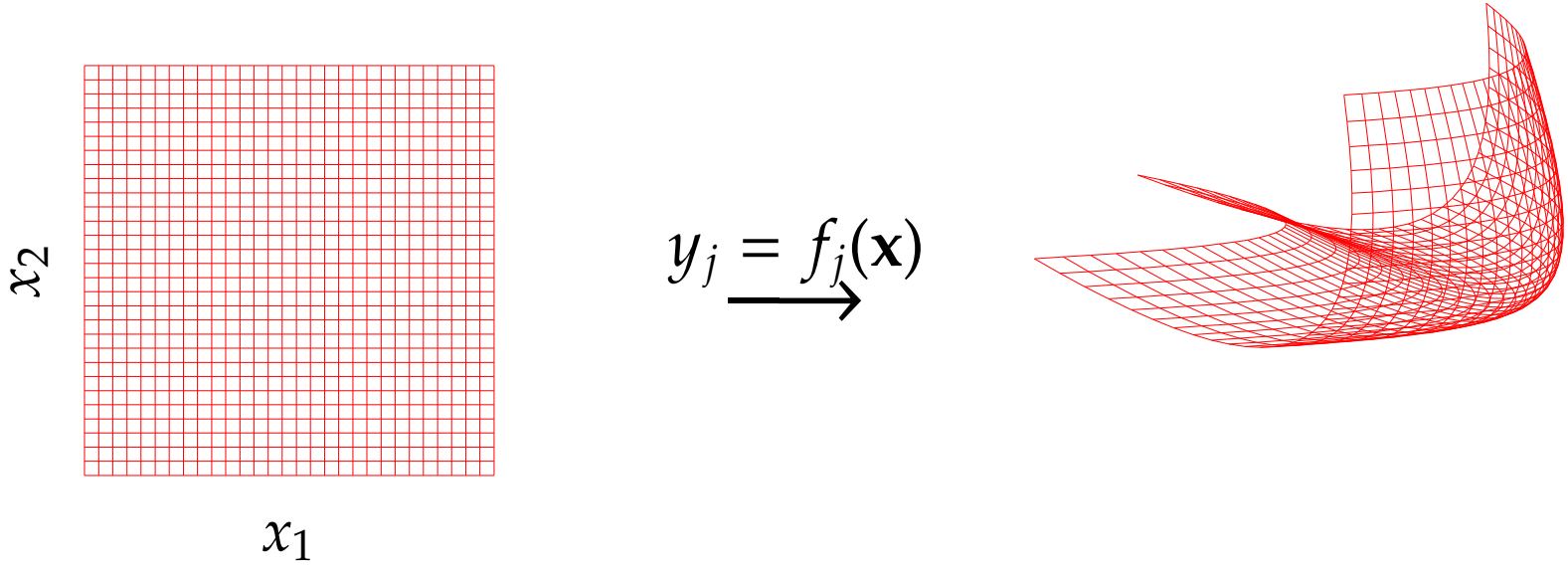


Figure : A three dimensional manifold formed by mapping from a two dimensional space to a three dimensional space.

Difficulty for Probabilistic Approaches

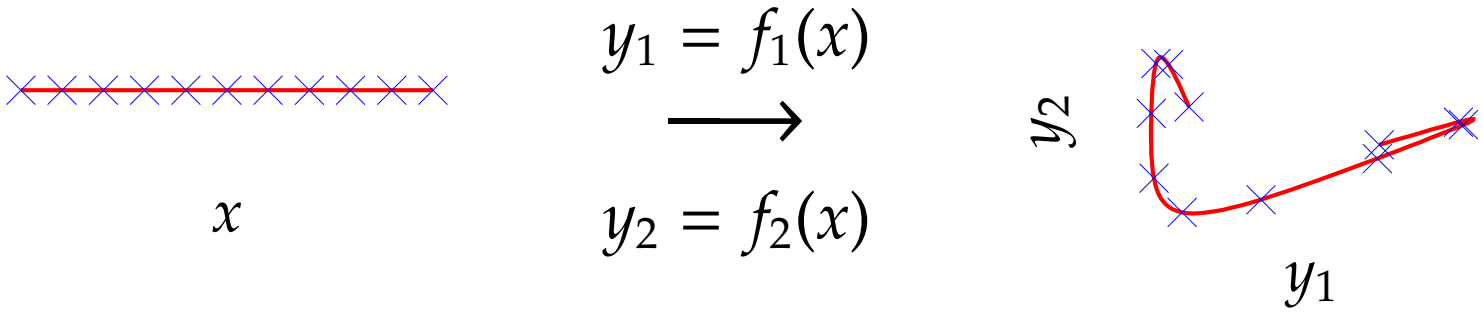


Figure : A string in two dimensions, formed by mapping from one dimension, x, line to a two dimensional space, $[y_1, y_2]$ using nonlinear functions $f_1(\cdot)$ and $f_2(\cdot)$.

Difficulty for Probabilistic Approaches

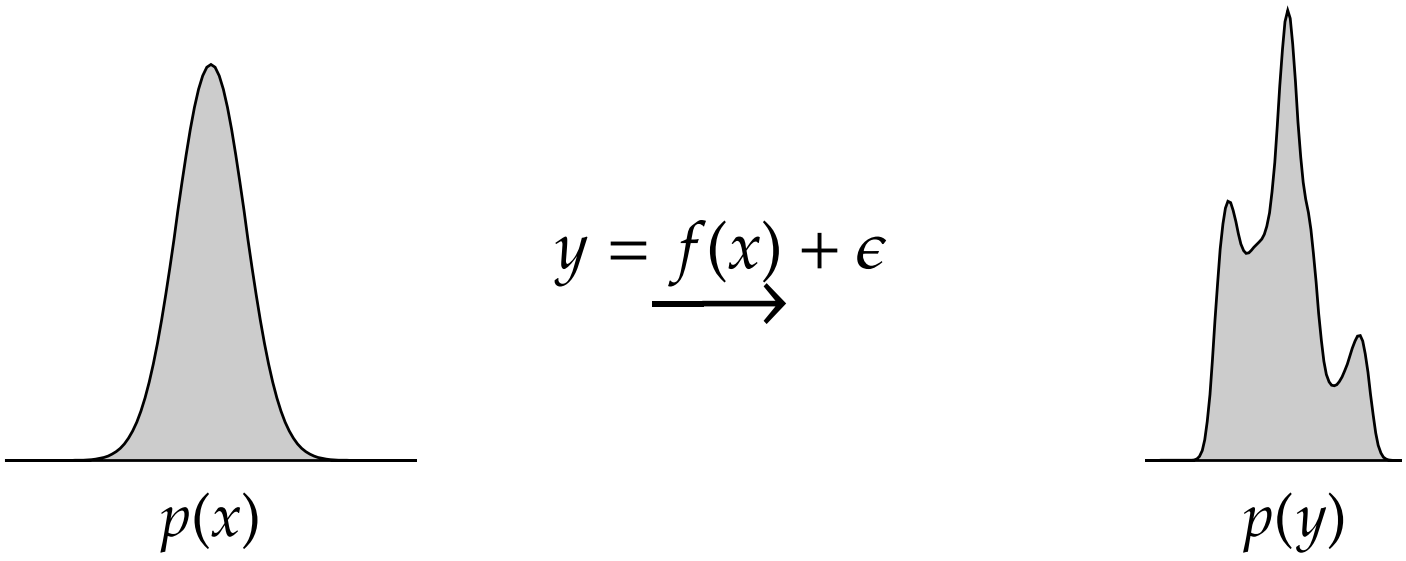


Figure : A Gaussian distribution propagated through a non-linear mapping. $y_i = f(x_i) + \epsilon_i$. $\epsilon \sim \mathcal{N}\left(0, 0.2^2\right)$ and $f(\cdot)$ uses RBF basis, 100 centres between -4 and 4 and $\ell = 0.1$. New distribution over y (right) is multimodal and difficult to normalize.

Variational Compression

(Lawrence, 2007; Titsias, 2009)

- Complexity of standard GP:
  - $O(n^3)$ in computation.
  - $O(n^2)$ in storage.

Variational Compression

(Lawrence, 2007; Titsias, 2009)

- Complexity of standard GP:
  - $O(n^3)$ in computation.
  - $O(n^2)$ in storage.
- Via low rank representations of covariance:
  - $O(nm^2)$ in computation.
  - $O(nm)$ in storage.
- Where m is user chosen number of *inducing* variables. They give the rank of the resulting covariance.

Variational Compression

(Lawrence, 2007; Titsias, 2009)

- Complexity of standard GP:
  - $O(n^3)$ in computation.
  - $O(n^2)$ in storage.
- Via low rank representations of covariance:
  - $O(nm^2)$ in computation.
  - $O(nm)$ in storage.
- Where m is user chosen number of *inducing* variables. They give the rank of the resulting covariance.

Variational Compression

- Inducing variables are a compression of the real observations.
- They can live in space of $\mathbf{f}$ or a space that is related through a linear operator (Álvarez et al., 2010) — could be gradient or convolution.
- There are inducing variables associated with each set of hidden variables, $\mathbf{x}^i$.
- **Importantly** conditioning on inducing variables renders the likelihood independent across the data.
  - It turns out that this allows us to variationally handle uncertainty on the kernel (including the inputs to the kernel).
  - It also allows standard scaling approaches: stochastic variational inference Hensman et al. (2013), parallelization Gal et al. (2014) and work by Zhenwen Dai on GPUs to be applied: an *engineering* challenge?

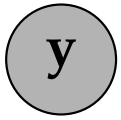
Inducing Variable Approximations

- Date back to (Williams and Seeger, 2001; Smola and Bartlett, 2001; Csató and Opper, 2002; Seeger et al., 2003; Snelson and Ghahramani, 2006). See Quiñonero Candela and Rasmussen (2005) for a review.
- We follow variational perspective of (Titsias, 2009).
- This is an augmented variable method, followed by a collapsed variational approximation (King and Lawrence, 2006; Hensman et al., 2012).

Augmented Variable Model: Not Wrong but Useful?

Augment standard model with a set of m new inducing variables, $\mathbf{u}$.

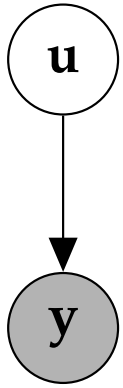
$$p(\mathbf{y}) = \int p(\mathbf{y}, \mathbf{u}) \mathrm{d}\mathbf{u}$$



Augmented Variable Model: Not Wrong but Useful?

Augment standard model with a set of m new inducing variables, $\mathbf{u}$.

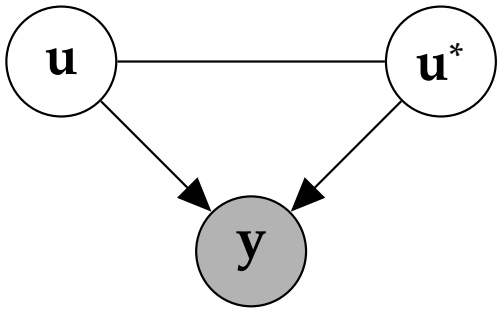
$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u}$$



Augmented Variable Model: Not Wrong but Useful?

Important: Ensure inducing variables are *also* Kolmogorov consistent (we have m^* other inducing variables we are not *yet* using.)

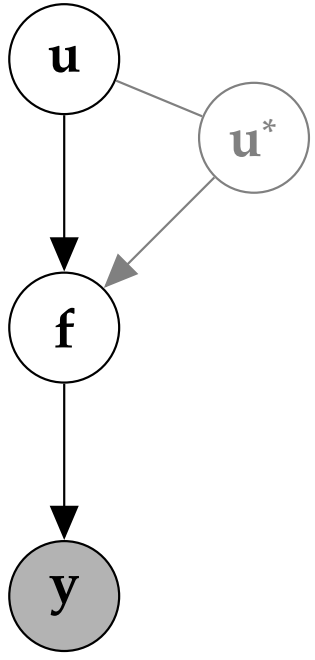
$$p(\mathbf{u}) = \int p(\mathbf{u}, \mathbf{u}^*) \mathrm{d}\mathbf{u}^*$$



Augmented Variable Model: Not Wrong but Useful?

Assume that relationship is through **f** (represents 'fundamentals'—push Kolmogorov consistency up to here).

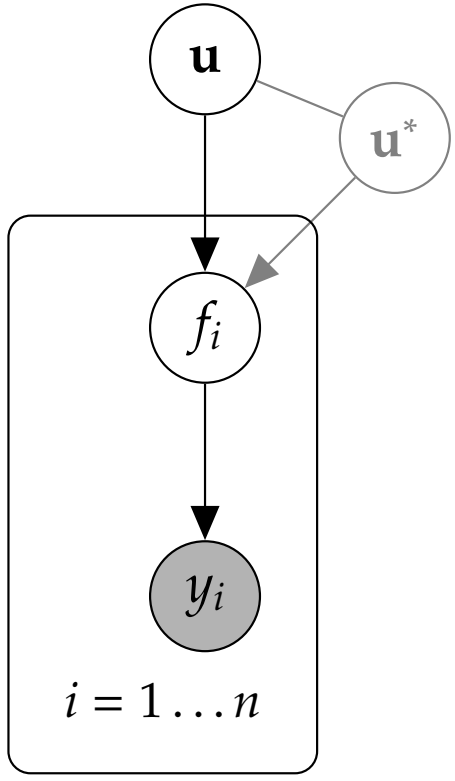
$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u}$$



Augmented Variable Model: Not Wrong but Useful?

Convenient to assume factorization (*doesn't* invalidate model—think delta function as worst case).

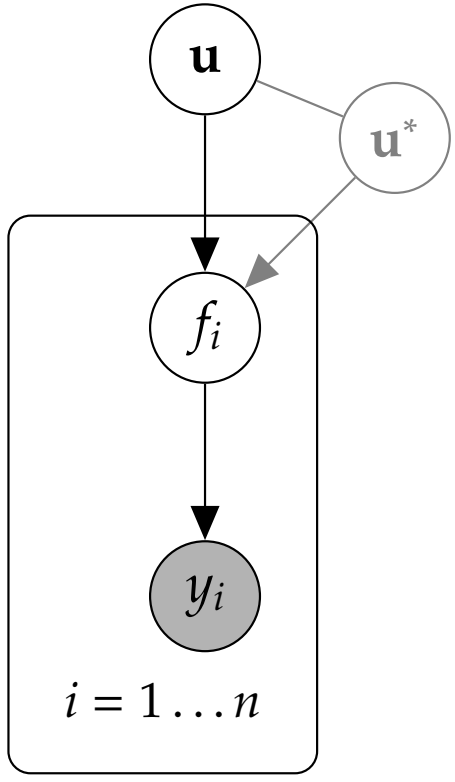
$$p(\mathbf{y}) = \int \prod_{i=1}^{n} p(y_i|f_i)p(\mathbf{f}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u}$$



Augmented Variable Model: Not Wrong but Useful?

Focus on integral over $\mathbf{f}$.

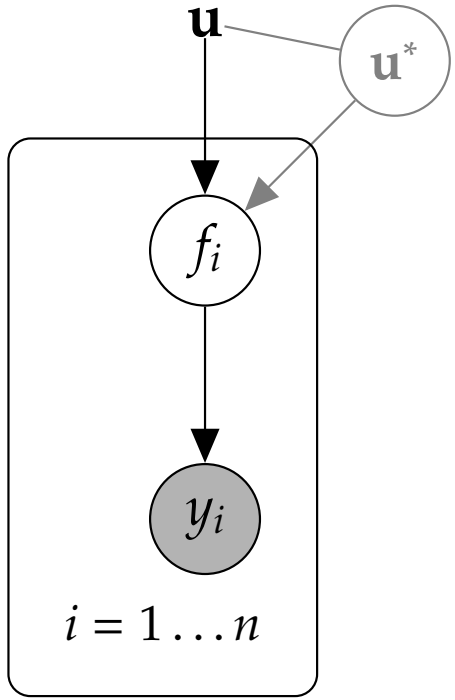
$$p(\mathbf{y}) = \int \int \prod_{i=1}^{n} p(y_i|f_i)p(\mathbf{f}|\mathbf{u})\mathrm{d}\mathbf{f}p(\mathbf{u})\mathrm{d}\mathbf{u}$$



Augmented Variable Model: Not Wrong but Useful?

Focus on integral over $\mathbf{f}$.

$$p(\mathbf{y}|\mathbf{u}) = \int \prod_{i=1}^{n} p(y_i|f_i)p(\mathbf{f}|\mathbf{u})\mathrm{d}\mathbf{f}$$



Relationship to Nyström Approximation

- Variational lower bound leads to Nyström style approximation (Williams and Seeger, 2001). Relations to subset of regressors (Poggio and Girosi, 1990; Williams et al., 2002).

$$\mathbf{K} \approx \sigma^2 \mathbf{I} + \mathbf{K}_{\mathbf{fu}} \mathbf{K}_{\mathbf{uu}}^{-1} \mathbf{K}_{\mathbf{uf}}$$

- Has probabilistic interpretation of

$$\mathbf{u} \sim \mathcal{N}(0, \mathbf{K}_{\mathbf{uu}})$$
$$\mathbf{y}|\mathbf{u} \sim \mathcal{N}\left(\mathbf{K}_{\mathbf{fu}} \mathbf{K}_{\mathbf{uu}}^{-1} \mathbf{u}, \sigma^2 \mathbf{I}\right)$$

cf

$$\mathbf{w} \sim \mathcal{N}(0, \alpha \mathbf{I})$$
$$\mathbf{y}|\mathbf{w} \sim \mathcal{N}\left(\mathbf{\Phi} \mathbf{w}, \sigma^2 \mathbf{I}\right)$$
$$\mathbf{y} \sim \mathcal{N}\left(0, \alpha \mathbf{\Phi} \mathbf{\Phi}^\top + \sigma^2 \mathbf{I}\right)$$

Leads to Other Approximations ...

- Let's be explicity about storing approximate posterior of $\mathbf{u}$, $q(\mathbf{u})$.
- Now we have

$$p(\mathbf{y}^*|\mathbf{y}) = \int p(\mathbf{y}^*|\mathbf{u})q(\mathbf{u}|\mathbf{y})\mathbf{u}$$

- Inducing variables look a lot like regular parameters.
- *But*: their dimensionality does not need to be set at design time.
- They can be modified arbitrarily at run time without effecting the model likelihood.
- They only effect the quality of compression and the lower bound.

In GPs for Big Data

- Exploit the resulting factorization ...

$$p(\mathbf{y}^*|\mathbf{y}) = \int p(\mathbf{y}^*|\mathbf{u})q(\mathbf{u}|\mathbf{y})\mathbf{u}$$

In GPs for Big Data

- Exploit the resulting factorization ...

$$p(\mathbf{y}^*|\mathbf{y}) = \int p(\mathbf{y}^*|\mathbf{u})q(\mathbf{u}|\mathbf{y})\mathbf{u}$$

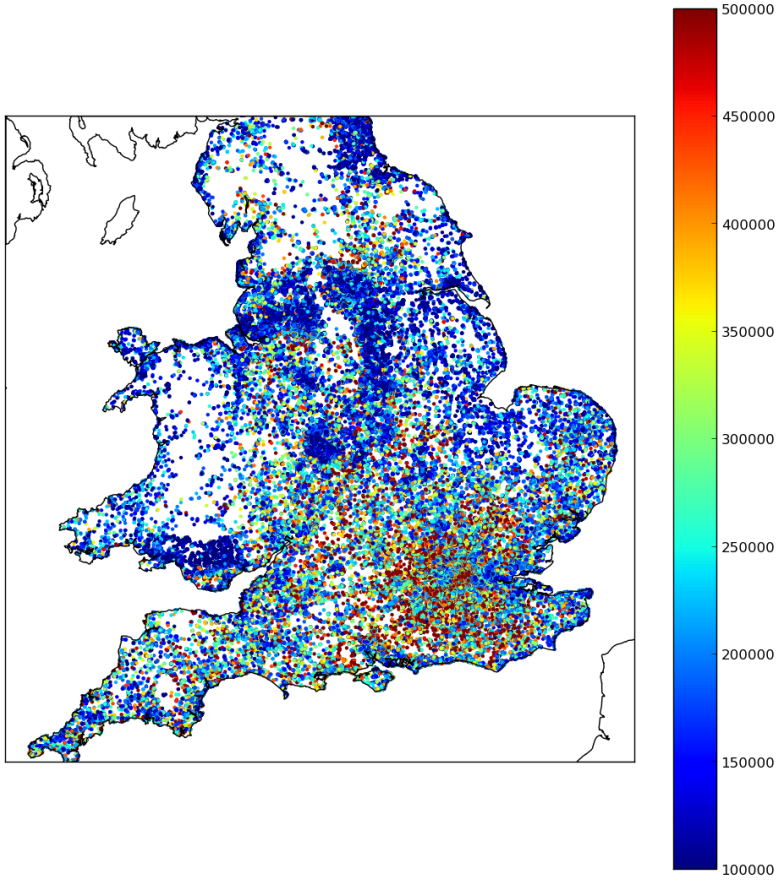
- The distribution now *factorizes*:

$$p(\mathbf{y}^*|\mathbf{y}) = \int \prod_{i=1}^{n^*} p(y_i^*|\mathbf{u})q(\mathbf{u}|\mathbf{y})\mathbf{u}$$

- This factorization can be exploited for stochastic variational inference (Hoffman et al., 2012).

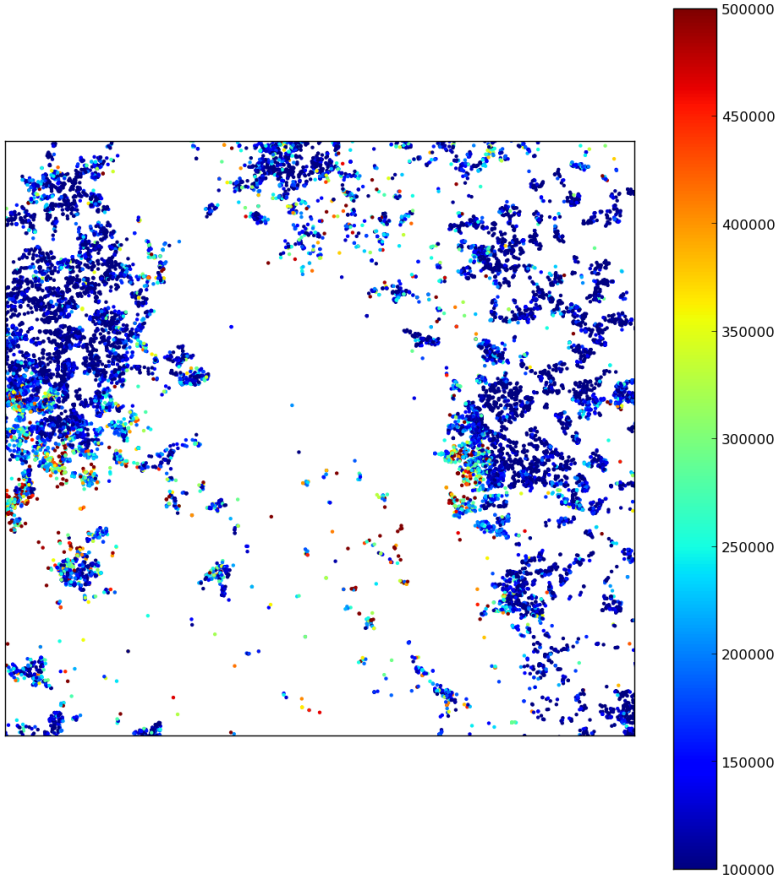
Nonparametrics for Very Large Data Sets

Modern data availability



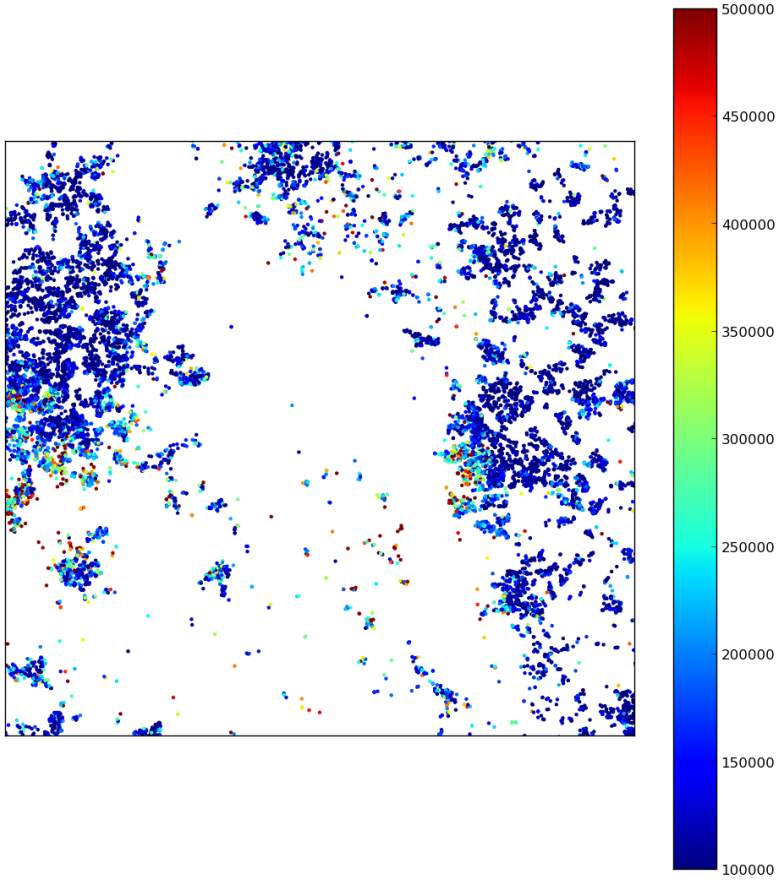
Nonparametrics for Very Large Data Sets

Proxy for index of deprivation?



Nonparametrics for Very Large Data Sets

Actually index of deprivation is a proxy for this ...



Hensman et al. (2013)

Gaussian Processes for Big Data

James Hensman*
Dept. Computer Science
The University of Sheffield
Sheffield, UK

Nicolò Fusi*
Dept. Computer Science
The University of Sheffield
Sheffield, UK

Neil D. Lawrence*
Dept. Computer Science
The University of Sheffield
Sheffield, UK

Abstract

We introduce stochastic variational inference for Gaussian process models. This enables the application of Gaussian process (GP) models to data sets containing millions of data points. We show how GPs can be variationally decomposed to depend on a set

Even to accommodate these data sets, various approximate techniques are required. One approach is to partition the data set into separate groups [e.g. Snelson and Ghahramani, 2007, Urtasun and Darrell, 2008]. An alternative is to build a low rank approximation to the covariance matrix based around 'inducing variables' [see e.g. Csató and Opper, 2002, Seeger et al., 2003, Quiñonero Candela and Rasmussen, 2005, Tit-

Hensman et al. (2013)

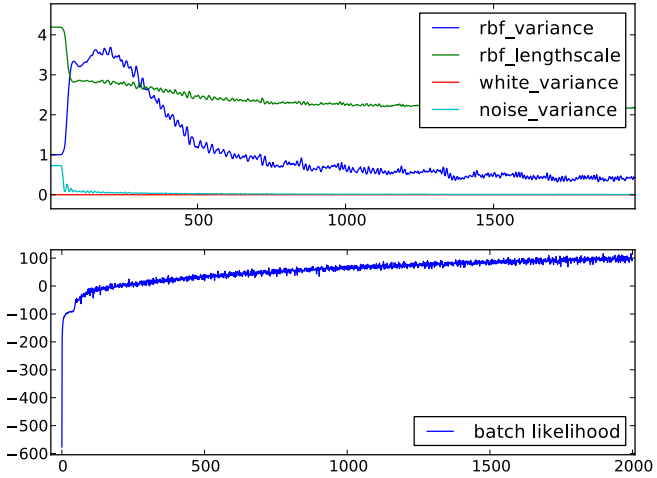


Figure 4: Convergence of the SVIGP algorithm on the two dimensional toy data

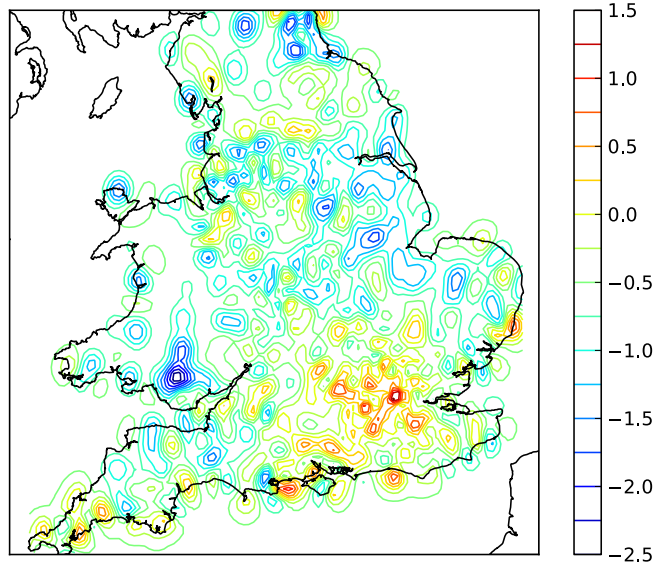
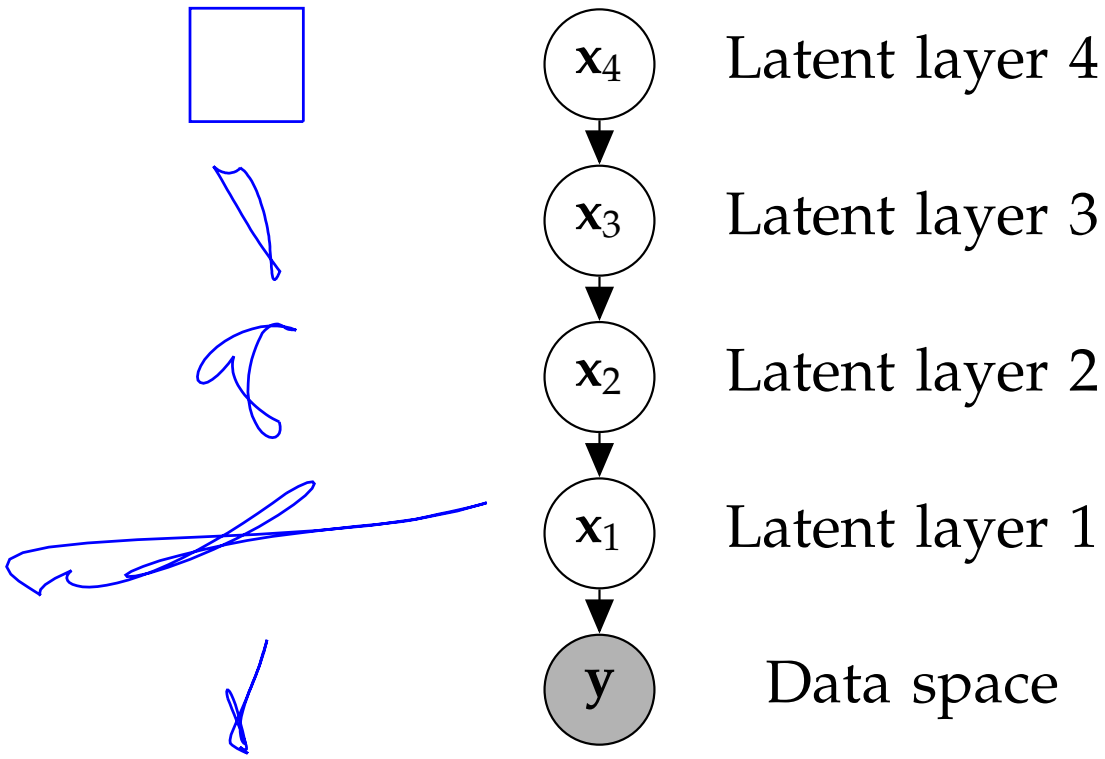


Figure 5: Variability of apartment price (logarithmically!) throughout England and Wales.

`land-registry-monthly-price-paid-data/`, which covers England and Wales, and filtered for apartments. This resulted in a data set with 75,000 entries,

ted a GP with the same covariance function as our

Structures for Extracting Information from Data



Damianou and Lawrence (2013)

Deep Gaussian Processes

Andreas C. Damianou **Neil D. Lawrence**
Dept. of Computer Science & Sheffield Institute for Translational Neuroscience,
University of Sheffield, UK

Abstract

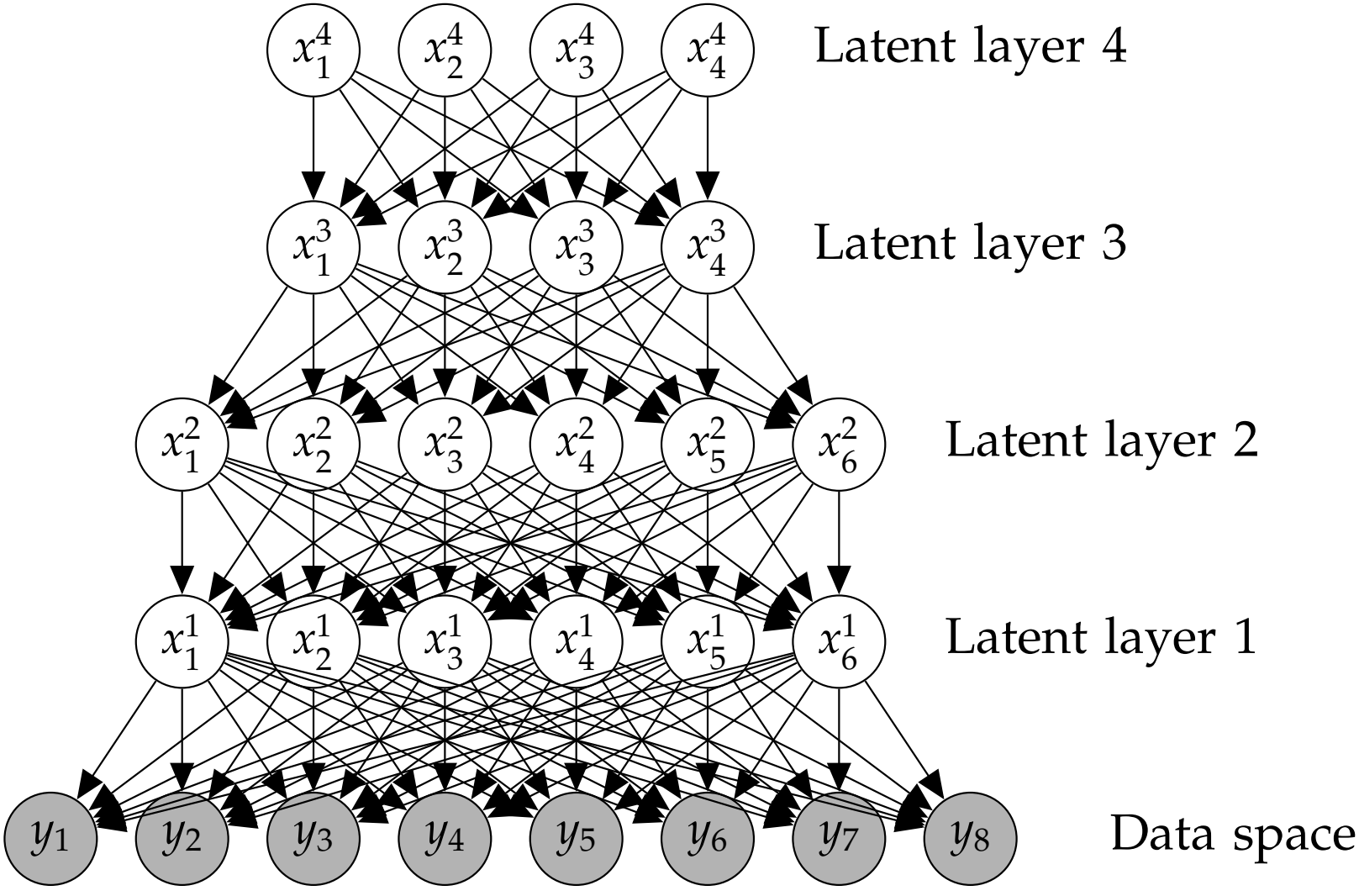
In this paper we introduce deep Gaussian process (GP) models. Deep GPs are a deep belief network based on Gaussian process mappings. The data is modeled as the output of a multivariate GP. The inputs to that Gaussian process are then governed by another GP. A single layer model is equivalent to a standard GP or the GP latent variable model (GP-LVM). We perform inference in

the question as to whether deep structures and the learning of abstract structure can be undertaken in *smaller* data sets. For smaller data sets, questions of generalization arise: to demonstrate such structures are justified it is useful to have an objective measure of the model's applicability.

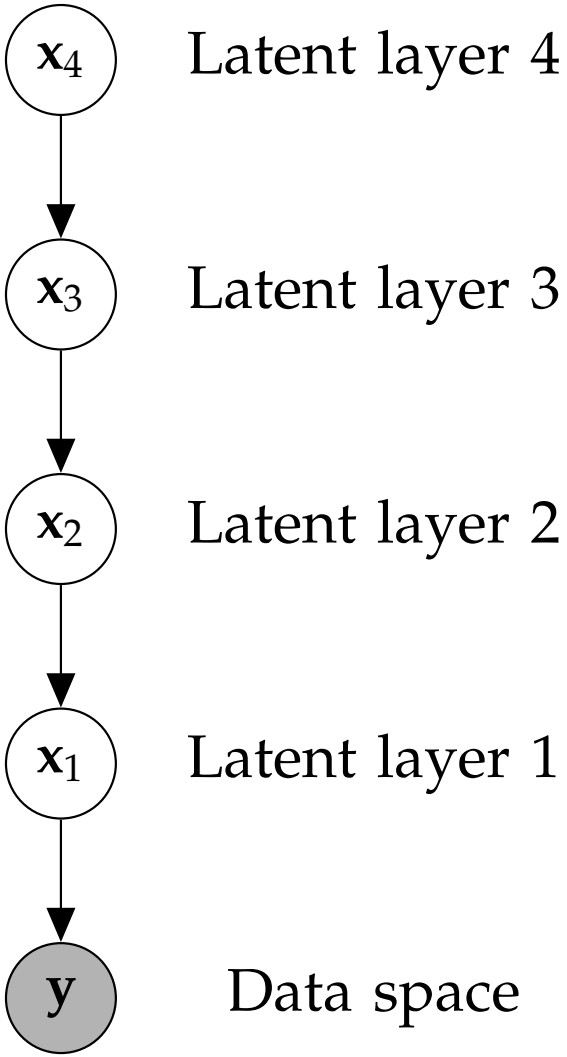
The traditional approach to deep learning is based around binary latent variables and the restricted Boltzmann machine (RBM) [Hinton, 2010]. Deep hierarchies are constructed by stacking these models and various approximate inference techniques (such as contrastive divergence)

Outline

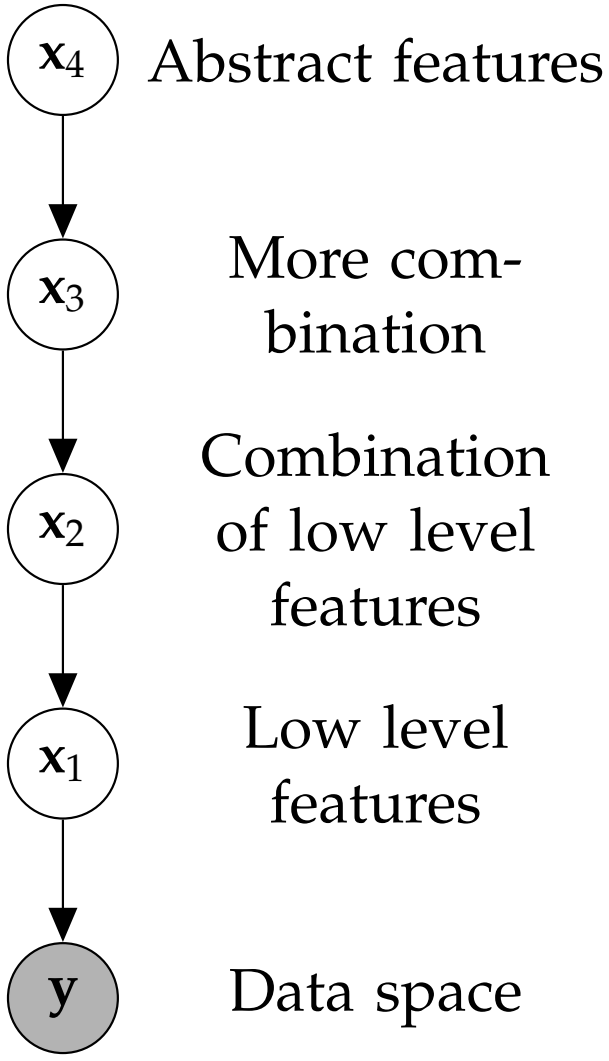
Deep Models


x_1^4 x_2^4 x_3^4 x_4^4
Latent layer 4
x_1^3 x_2^3 x_3^3 x_4^3
Latent layer 3
x_1^2 x_2^2 x_3^2 x_4^2 x_5^2 x_6^2
Latent layer 2
x_1^1 x_2^1 x_3^1 x_4^1 x_5^1 x_6^1
Latent layer 1
y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8
Data space

Deep Models


$\mathbf{x}_4$
Latent layer 4
$\mathbf{x}_3$
Latent layer 3
$\mathbf{x}_2$
Latent layer 2
$\mathbf{x}_1$
Latent layer 1
$\mathbf{y}$
Data space

Deep Models

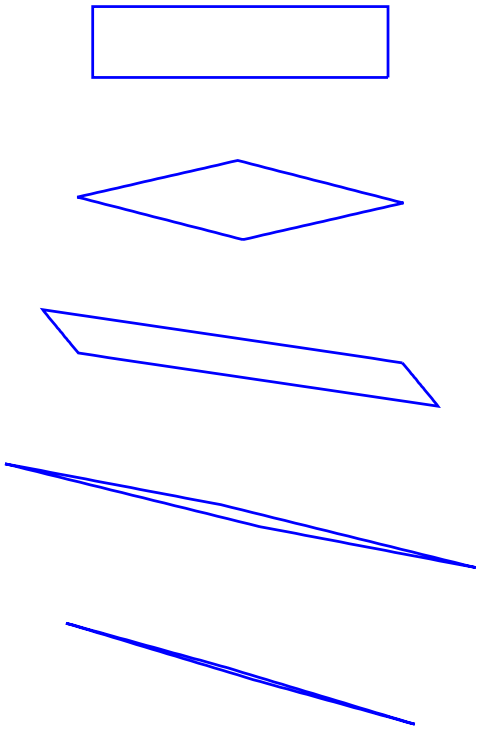

$\mathbf{x}_4$
Abstract features
$\mathbf{x}_3$
More combination
$\mathbf{x}_2$
Combination of low level features
$\mathbf{x}_1$
Low level features
$\mathbf{y}$
Data space

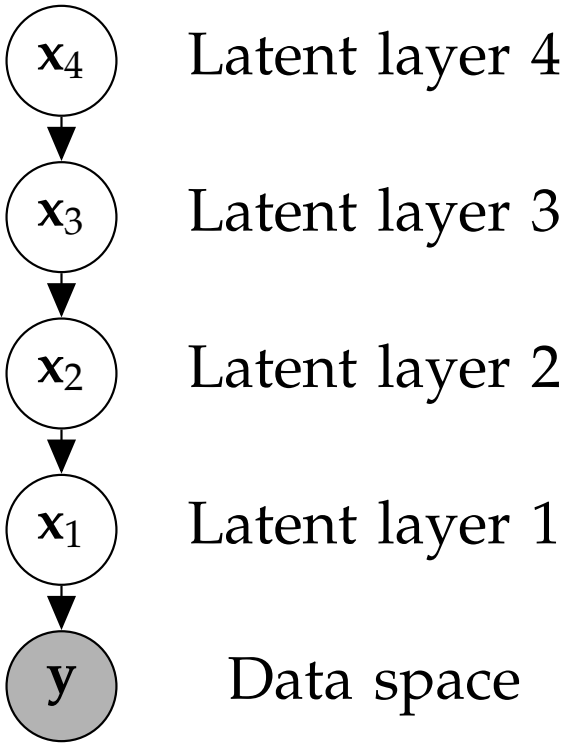
Deep Gaussian Processes

Damianou and Lawrence (2013)

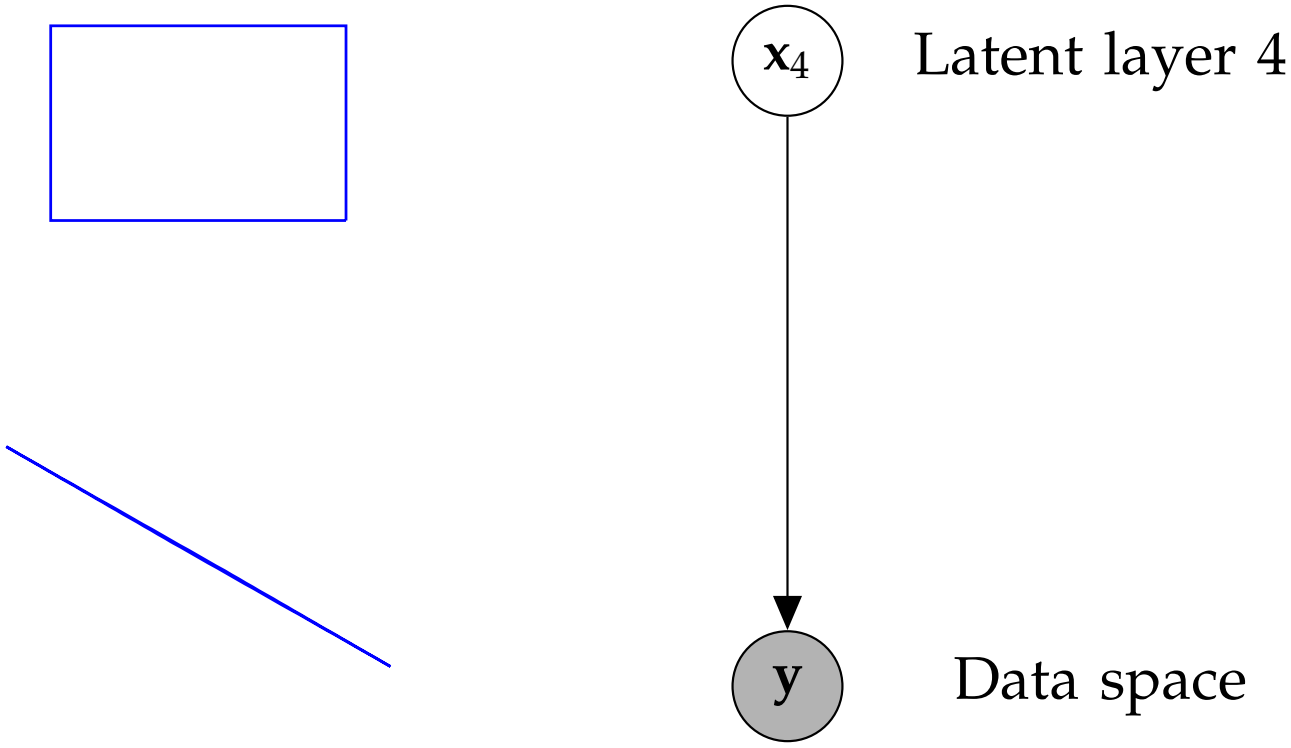
- Deep architectures allow abstraction of features (Bengio, 2009; Hinton and Osindero, 2006; Salakhutdinov and Murray, 2008).
- We use variational approach to stack GP models.

Stacked PCA

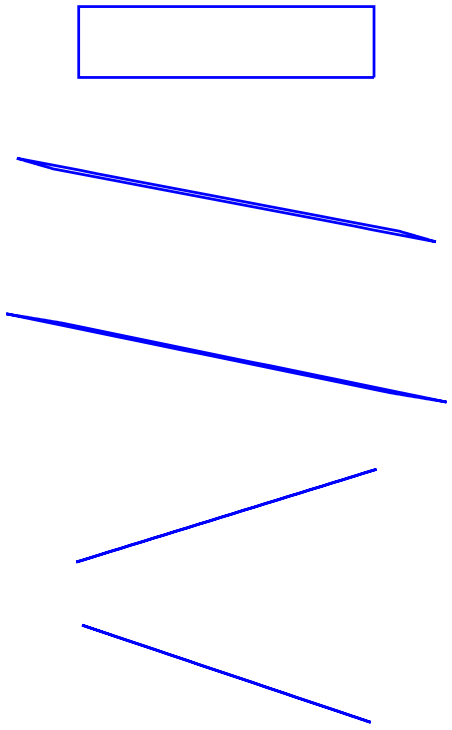


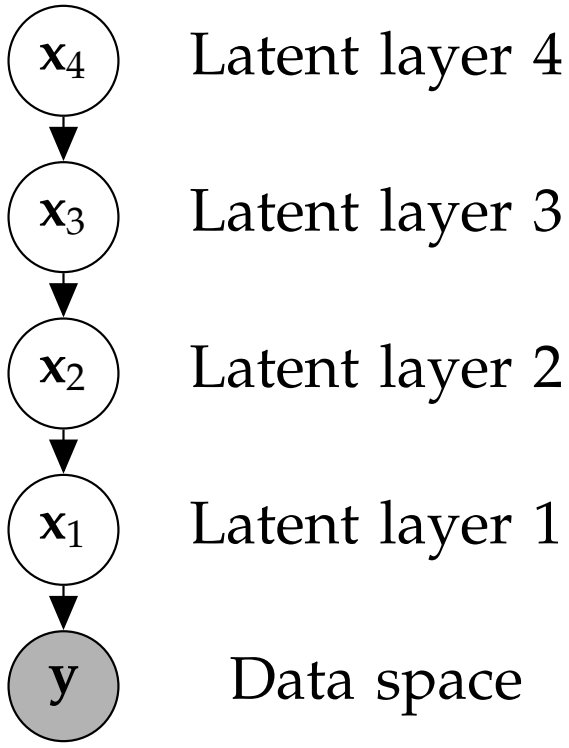

$\mathbf{x}_4$
Latent layer 4
$\mathbf{x}_3$
Latent layer 3
$\mathbf{x}_2$
Latent layer 2
$\mathbf{x}_1$
Latent layer 1
$\mathbf{y}$
Data space

Stacked PCA

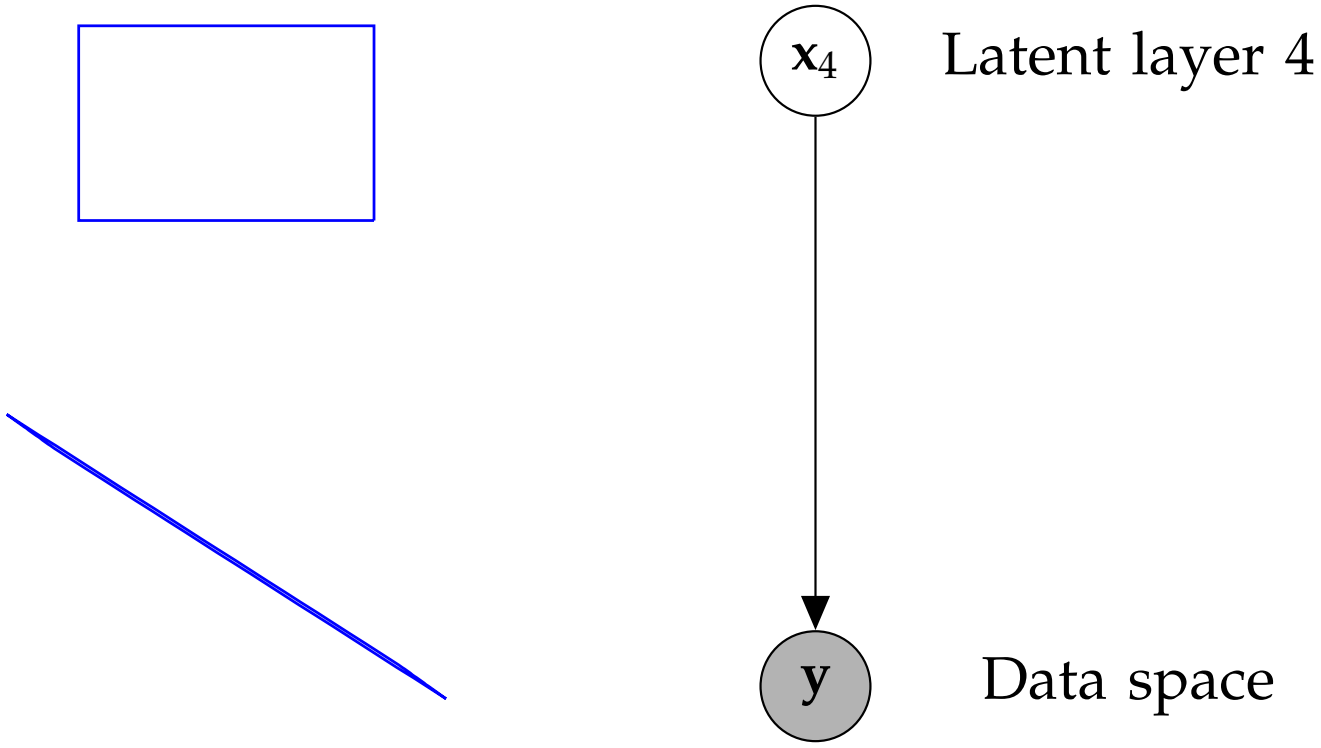


Stacked PCA

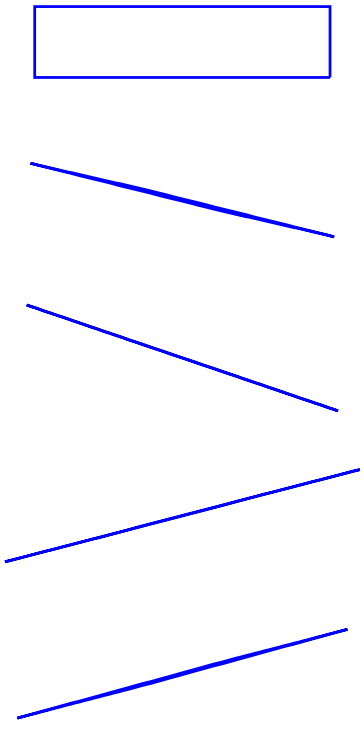


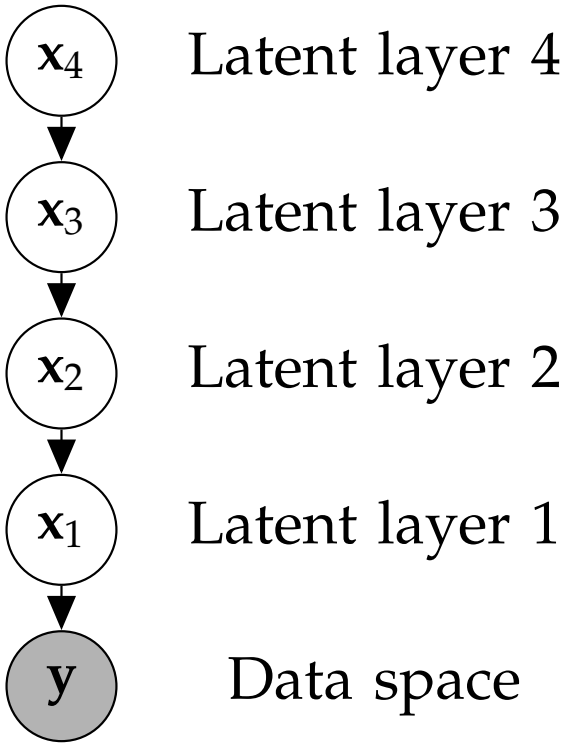

$\mathbf{x}_4$
Latent layer 4
$\mathbf{x}_3$
Latent layer 3
$\mathbf{x}_2$
Latent layer 2
$\mathbf{x}_1$
Latent layer 1
$\mathbf{y}$
Data space

Stacked PCA

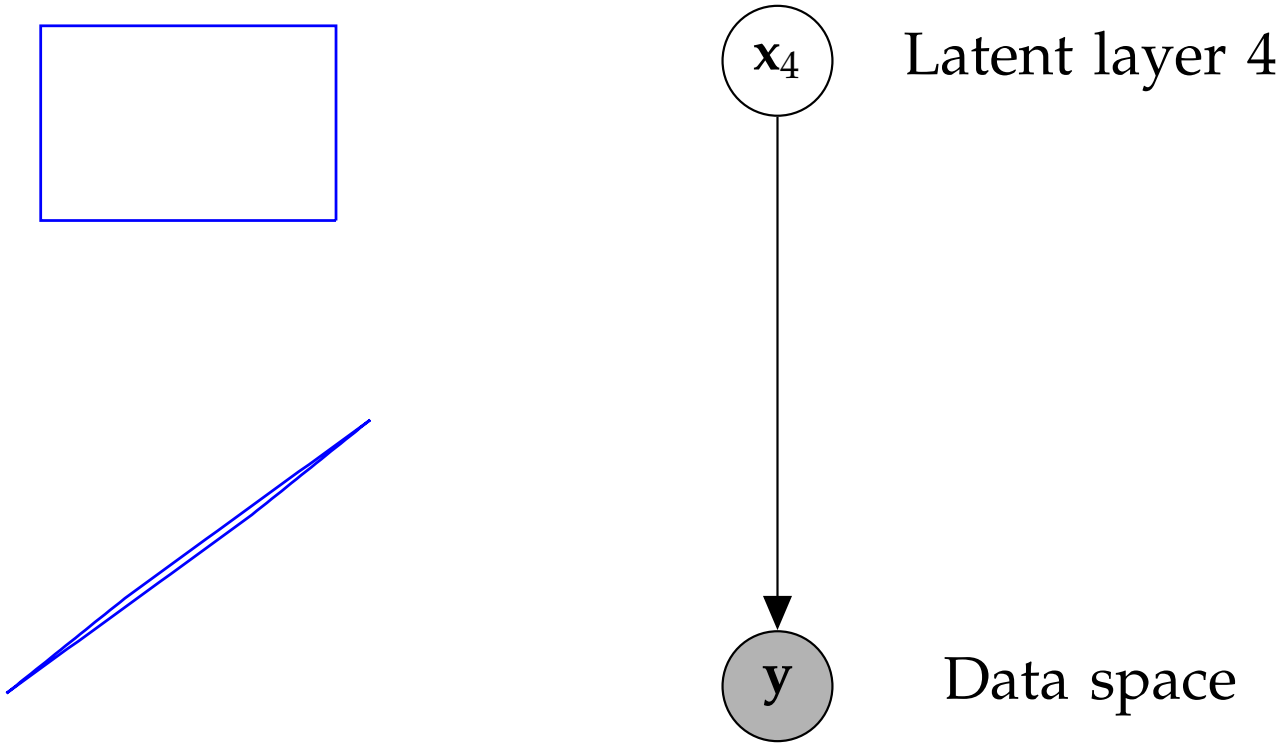


Stacked PCA

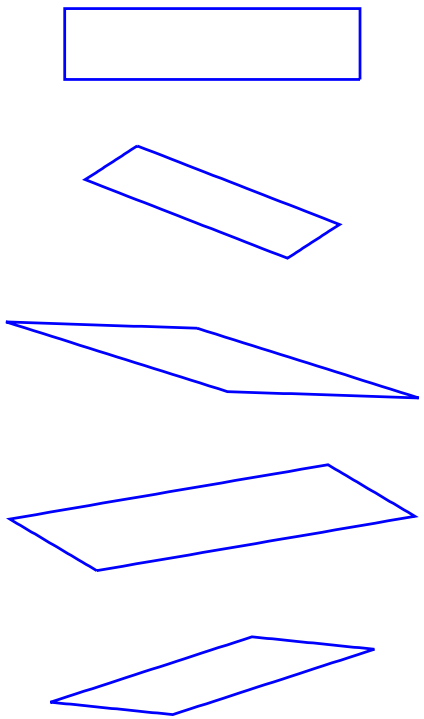


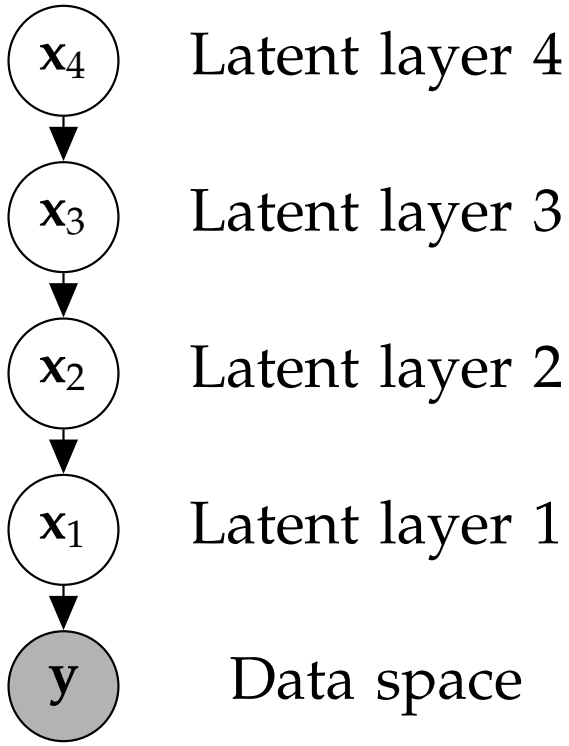

$\mathbf{x}_4$
Latent layer 4
$\mathbf{x}_3$
Latent layer 3
$\mathbf{x}_2$
Latent layer 2
$\mathbf{x}_1$
Latent layer 1
$\mathbf{y}$
Data space

Stacked PCA

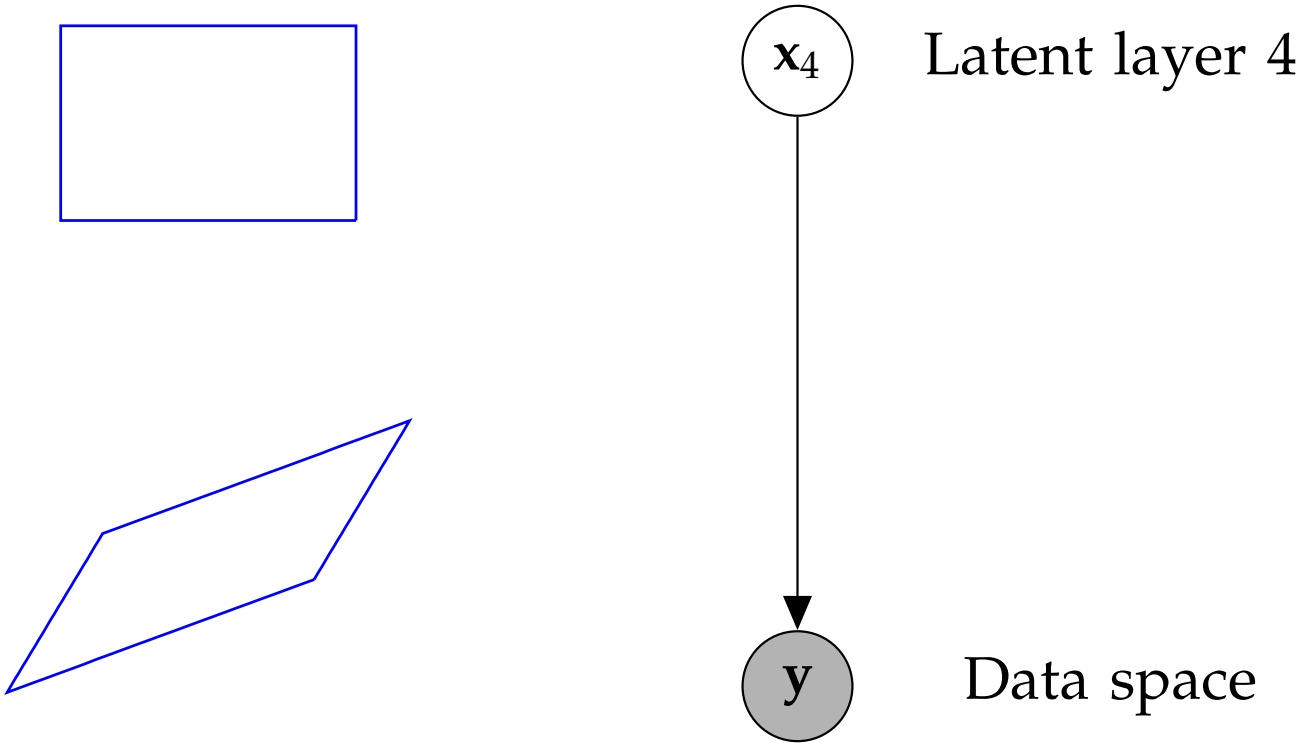


Stacked PCA

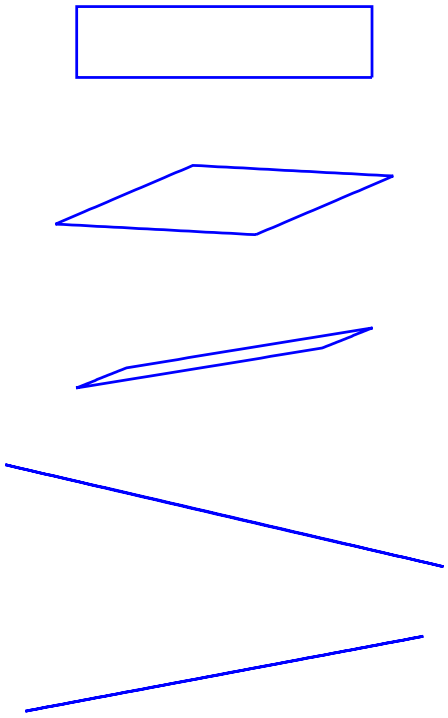


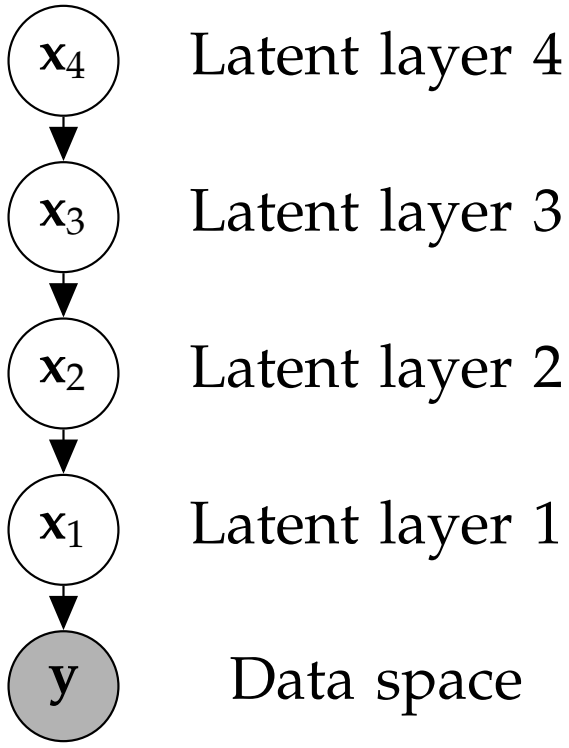

$\mathbf{x}_4$
Latent layer 4
$\mathbf{x}_3$
Latent layer 3
$\mathbf{x}_2$
Latent layer 2
$\mathbf{x}_1$
Latent layer 1
$\mathbf{y}$
Data space

Stacked PCA

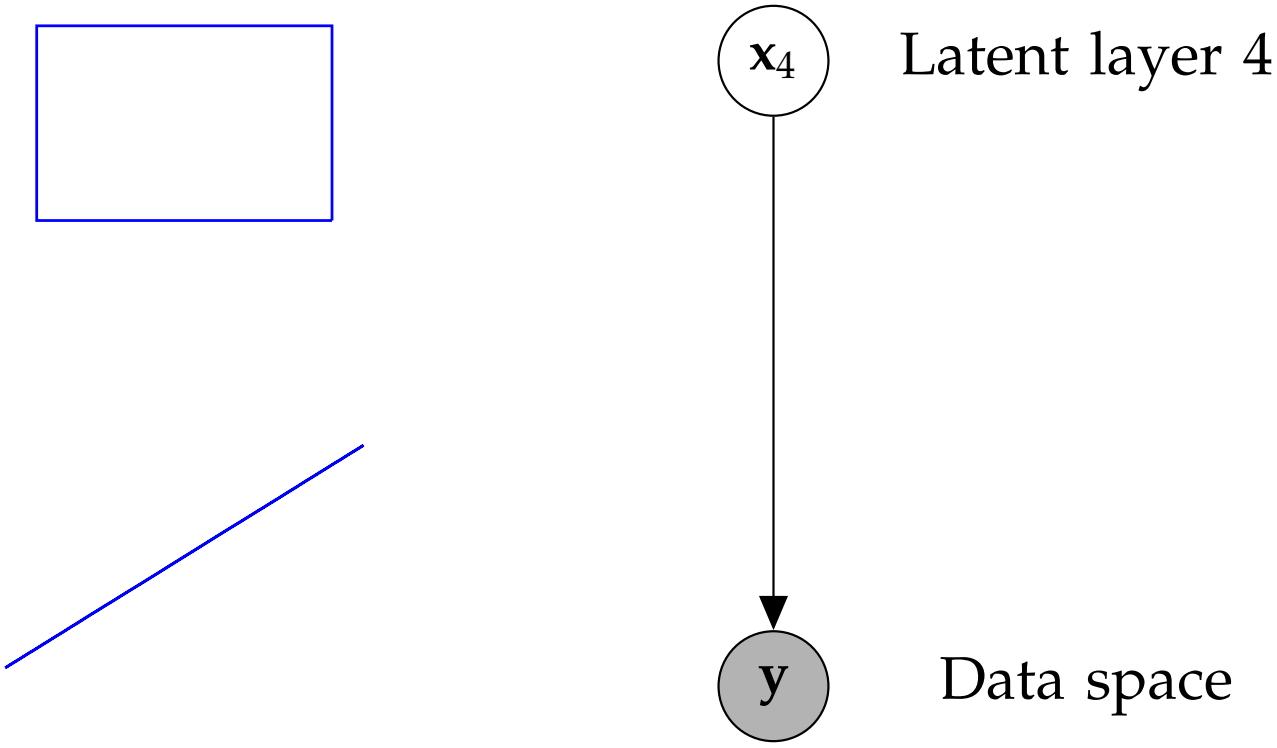


Stacked PCA

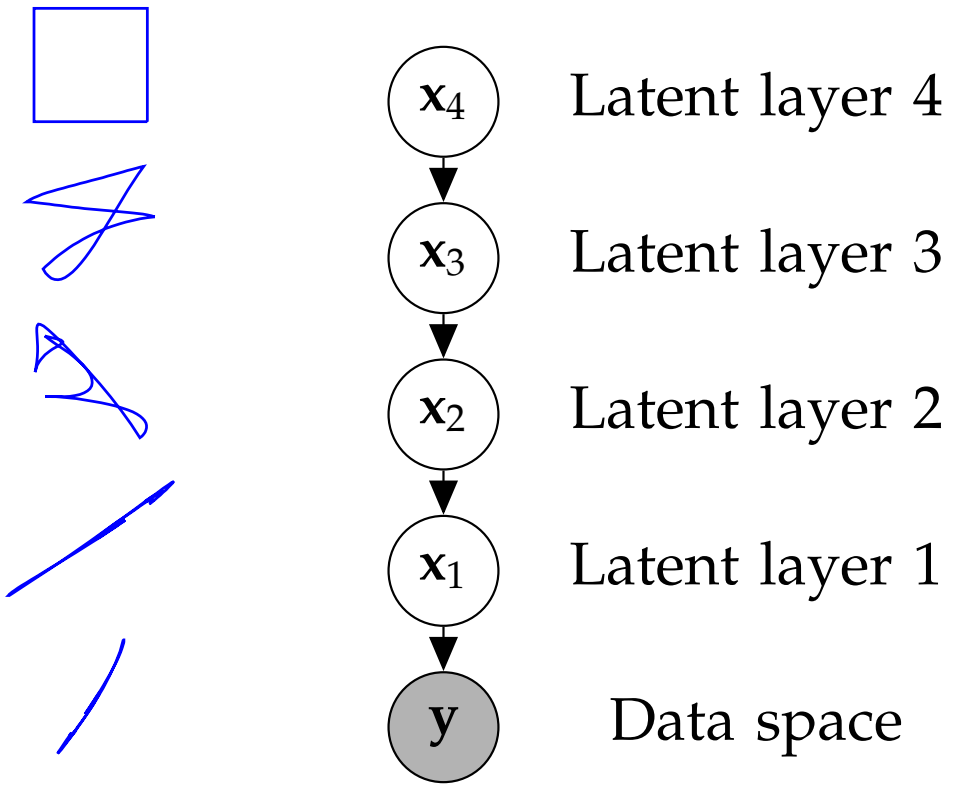



$\mathbf{x}_4$
Latent layer 4
$\mathbf{x}_3$
Latent layer 3
$\mathbf{x}_2$
Latent layer 2
$\mathbf{x}_1$
Latent layer 1
$\mathbf{y}$
Data space

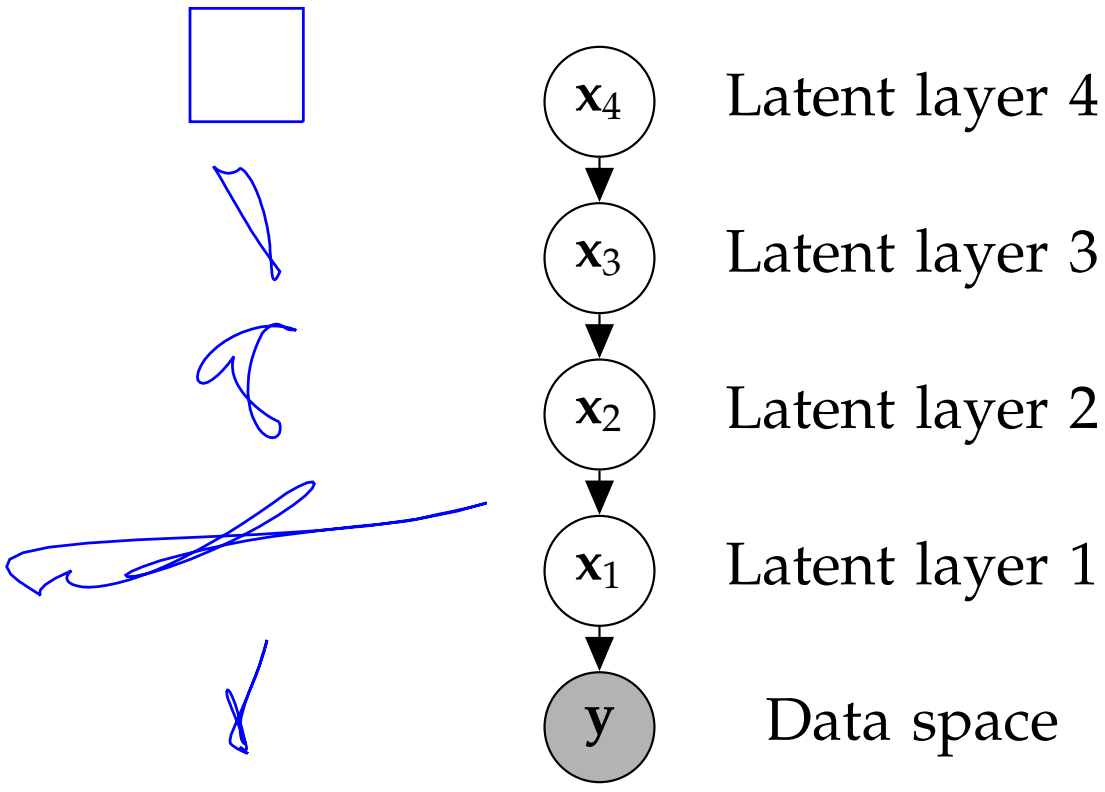
Stacked PCA



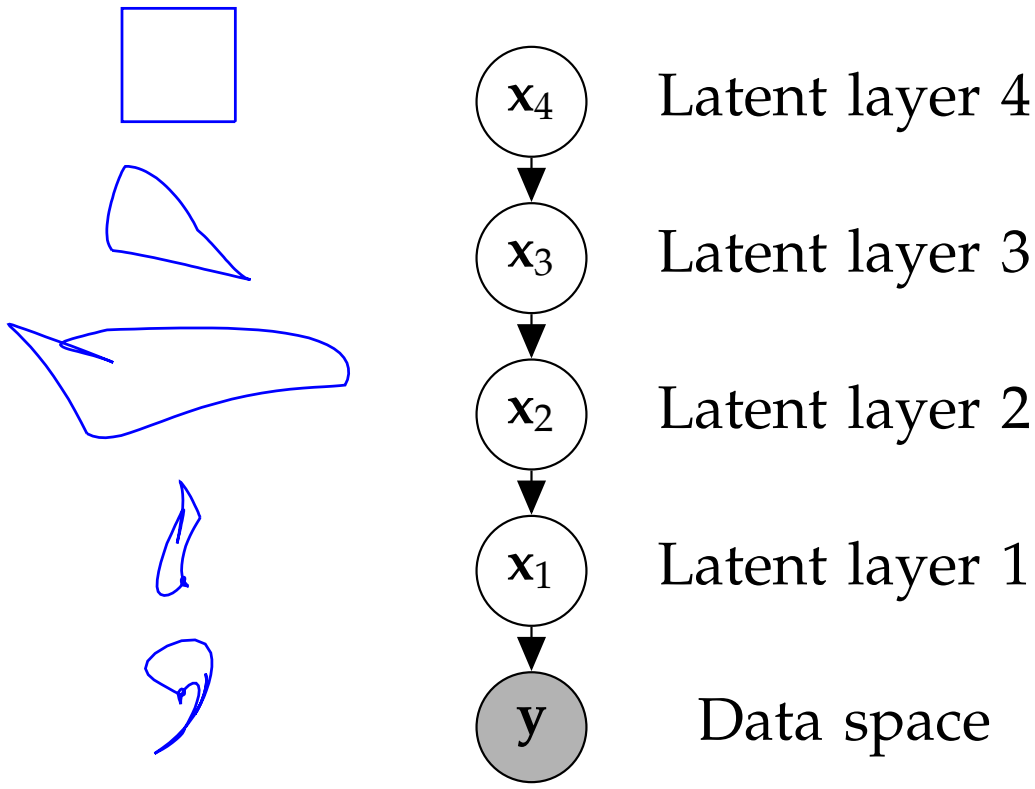
Stacked GPs


$\mathbf{x}_4$
Latent layer 4
$\mathbf{x}_3$
Latent layer 3
$\mathbf{x}_2$
Latent layer 2
$\mathbf{x}_1$
Latent layer 1
$\mathbf{y}$
Data space

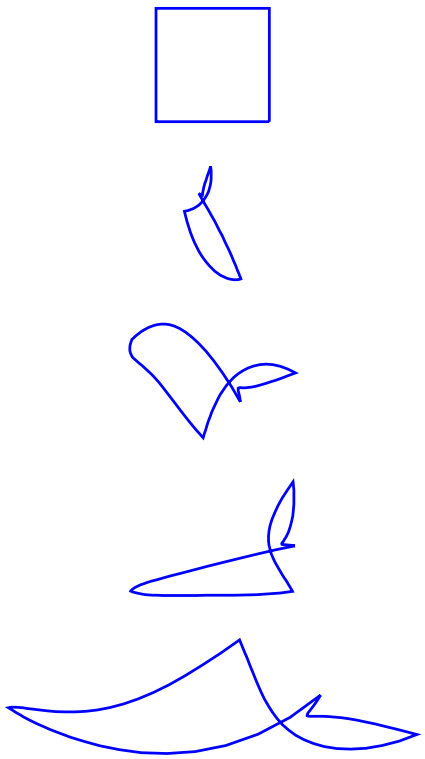
Stacked GPs

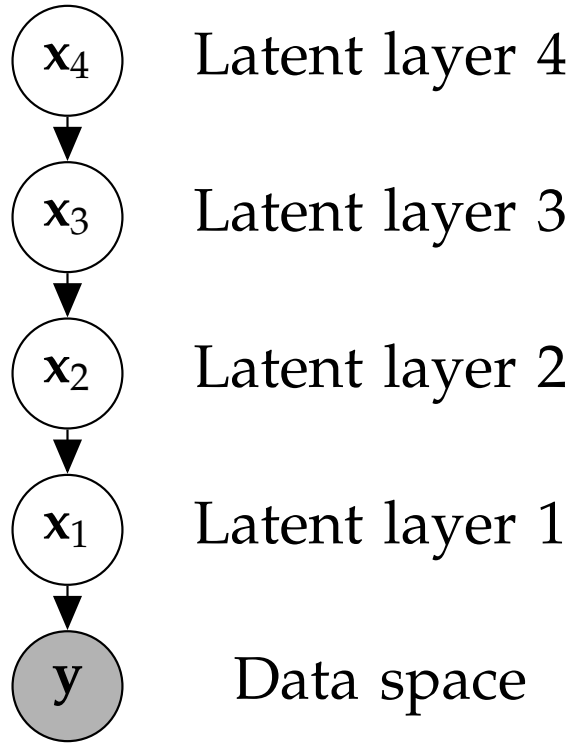

$\mathbf{x}_4$
Latent layer 4
$\mathbf{x}_3$
Latent layer 3
$\mathbf{x}_2$
Latent layer 2
$\mathbf{x}_1$
Latent layer 1
$\mathbf{y}$
Data space

Stacked GPs

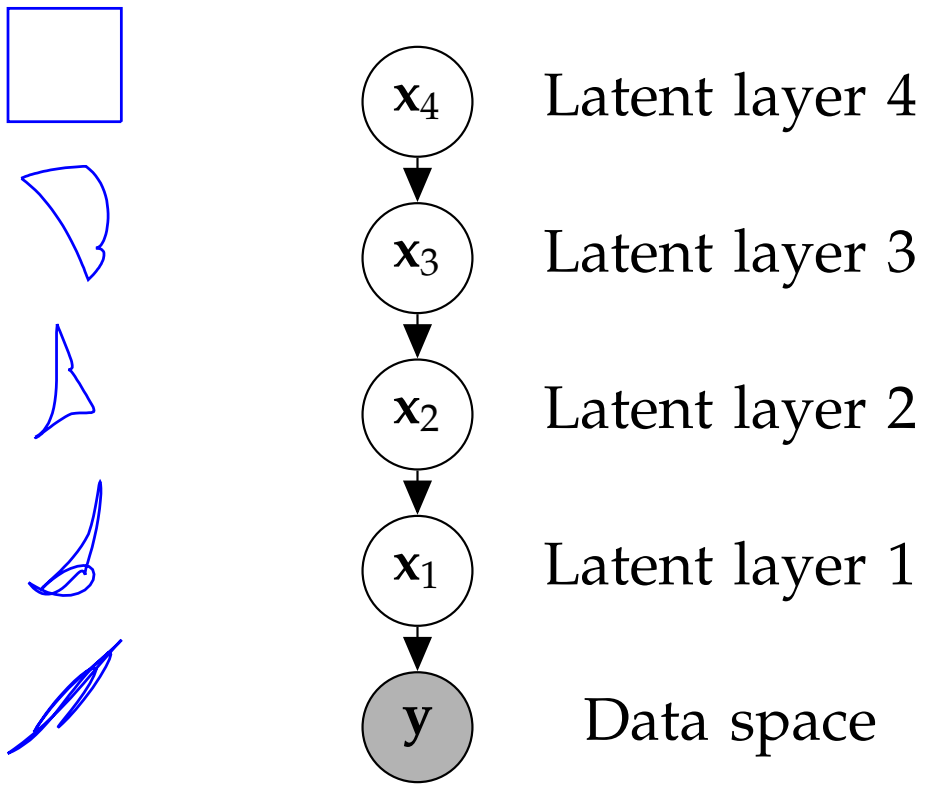

$\mathbf{x}_4$
Latent layer 4
$\mathbf{x}_3$
Latent layer 3
$\mathbf{x}_2$
Latent layer 2
$\mathbf{x}_1$
Latent layer 1
$\mathbf{y}$
Data space

Stacked GPs




$\mathbf{x}_4$
Latent layer 4
$\mathbf{x}_3$
Latent layer 3
$\mathbf{x}_2$
Latent layer 2
$\mathbf{x}_1$
Latent layer 1
$\mathbf{y}$
Data space

Stacked GPs


$\mathbf{x}_4$
Latent layer 4
$\mathbf{x}_3$
Latent layer 3
$\mathbf{x}_2$
Latent layer 2
$\mathbf{x}_1$
Latent layer 1
$\mathbf{y}$
Data space

Stacked GPs (video by David Duvenaud)

Duvenaud et al. (2014)

Avoiding pathologies in very deep networks

David Duvenaud
University of Cambridge

Oren Rippel
MIT, Harvard University

Ryan P. Adams
Harvard University

Zoubin Ghahramani
University of Cambridge

Abstract

Choosing appropriate architectures and regularization strategies for deep networks is crucial to good predictive performance. To shed light on this problem, we analyze the analogous problem of constructing useful priors on compositions of functions. Specifically, we study the deep Gaussian process, a type of infinitely-wide, deep neural network. We show that in standard architectures, the representational capacity of the network tends to capture fewer degrees of freedom as the number of layers increases, retaining only a single degree of freedom in the limit. We propose an alternate network architecture which does not suffer from this pathology. We also examine deep covariance functions, obtained by composing infinitely many feature transforms. Lastly, we characterize the class of models obtained by performing dropout on Gaussian processes.

1 Introduction

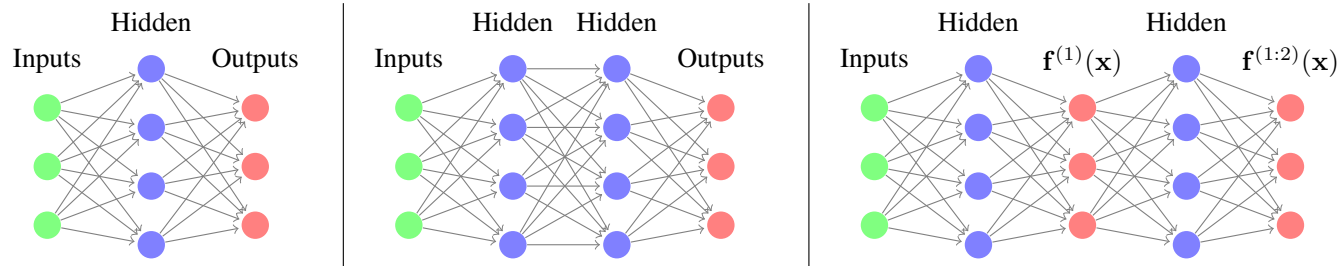
a composition of vector-valued functions, one per layer. Hence, understanding properties of such function compositions helps us gain insight into deep networks. In this paper, we examine a simple and flexible class of priors on compositions of functions, namely deep Gaussian processes (Damianou and Lawrence, 2013). Deep GPs are simply priors on compositions of vector-valued functions, where each output of each layer is drawn independently from a GP prior:

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

$$\mathbf{f}_d^{(\ell)} \overset{\text{ind}}{\sim} \text{GP}\left(0, k_d^{\ell}(\mathbf{x}, \mathbf{x}')\right) \quad (2)$$

These models correspond to a certain type of infinitely-wide multi-layer perceptron (MLP), and as such make canonical candidates for generative models of functions that closely relate to neural networks.

By characterizing these models, this paper shows that representations based on repeated composition of independently-initialized functions exhibit a pathology where the representation becomes invariant to all but one direction of variation. This corresponds to an eventual debilitating decrease in the information capacity of networks as a function of their number of layers. However, we will demonstrate that a simple change in architecture — namely,



a) One-hidden-layer MLP b) Two-hidden layer MLP c) Two-layer GP: $\mathbf{y} = \mathbf{f}^{(2)}\left(\mathbf{f}^{(1)}(\mathbf{x})\right)$

Figure 1: GPs can be understood as one-hidden-layer MLP with infinitely many hidden units (a). There are two possible interpretations of deep GPs: We can consider the deep GP to be a neural network with a finite number of hidden units, each with a different non-parametric activation function (b). Alternatively, we can consider every second layer to be a random linear combination of an infinite number of fixed parametric hidden units (c).

function applied element-wise. The output vector $f(\mathbf{x})$ is simply a weighted sum of these hidden unit activations:

$$f(\mathbf{x}) = \mathbf{V}^{(1)}\sigma\left(\mathbf{b}^{(1)} + \mathbf{W}^{(1)}\mathbf{x}\right) = \mathbf{V}^{(1)}\mathbf{h}^{(1)}(\mathbf{x}) \qquad (4)$$

where $\mathbf{V}^{(1)}$ is another weight matrix.

There exists a correspondence between one-layer MLPs and GPs (Neal, 1995). GPs can be viewed as a prior on neural networks with infinitely many hidden units, and unknown weights. More precisely, for any model of the form

$$f(\mathbf{x}) = \frac{1}{K}\alpha^{\mathsf{T}}\mathbf{h}(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^{K}\alpha_i h_i(\mathbf{x}), \qquad (5)$$

with fixed features $[h_1(\mathbf{x}), \ldots, h_K(\mathbf{x})]^{\mathsf{T}} = \mathbf{h}(\mathbf{x})$ and i.i.d. α's with zero mean and finite variance σ^2, the central limit theorem implies that as the number of features K grows, any two function values $f(\mathbf{x}), f(\mathbf{x}')$ have a joint distribution approaching $\mathcal{N}\left(0, \frac{\sigma^2}{K}\sum_{i=1}^{K} h_i(\mathbf{x})h_i(\mathbf{x}')\right)$. A joint Gaussian distribution between any set of function values is the definition of a Gaussian process.

The result is surprisingly general: it puts no constraints on

This architecture is shown in figure 1b. For example, if we extend the model given by (4) to have two layers of feature mappings, the resulting model is

$$f(\mathbf{x}) = \frac{1}{K}\alpha^{\mathsf{T}}\mathbf{h}^{(2)}\left(\mathbf{h}^{(1)}(\mathbf{x})\right) . \qquad (7)$$

If the features $\mathbf{h}(\mathbf{x})$ are considered fixed with only the last layer weights α unknown, this model corresponds to a GP with a "deep kernel":

$$k(\mathbf{x}, \mathbf{x}') = \left(\mathbf{h}^{(2)}(\mathbf{h}^{(1)}(\mathbf{x}))\right)^{\mathsf{T}}\mathbf{h}^{(2)}(\mathbf{h}^{(1)}(\mathbf{x}')) \qquad (8)$$

These models, examined in section 6, imply a fixed representation as opposed to a prior over representations, which is what we wish to analyze in this paper.

To construct a neural network with fixed nonlinearities corresponding to a deep GP, one must introduce a second layer in between each infinitely-wide set of fixed basis functions, as in figure 1c. The D_ℓ outputs $\mathbf{f}^{(\ell)}(\mathbf{x})$ in between each layer are weighted sums (with unknown weights) of the fixed hidden units of the layer below, and the next layer's hidden units depend only on these D_ℓ outputs.

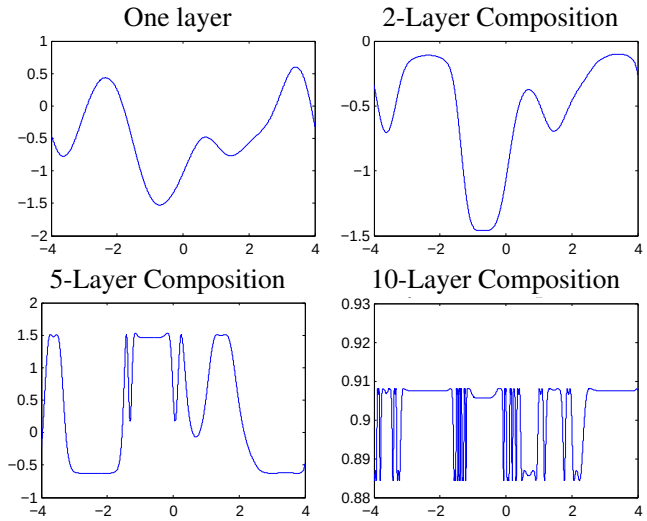


Figure 2: One-dimensional draws from a deep GP prior. After a few layers, the functions begin to be either nearly flat, or highly varying, everywhere. This is a consequence of the distribution on derivatives becoming heavy-tailed.

3 Characterizing deep Gaussian processes

In this section, we develop several theoretical results that explore the behavior of deep GPs as a function of their depth. This will allow us in section 4 to formally identify a pathology that emerges in very deep networks.

Specifically, we will show that the size of the derivative of a one-dimensional deep GP becomes log-normal distributed as the network becomes deeper. We'll also show that the Jacobian of a multivariate deep GP is a product of independent Gaussian matrices with independent entries.

3.1 One-dimensional asymptotics

In this section, we derive the limiting distribution of the derivative of an arbitrarily deep, one-dimensional GP with the expected magnitude of the derivative remains constant regardless of the depth.

The log of the magnitude of the derivatives has moments

$$m_{\log} = \mathbb{E}\left[\log\left|\frac{\partial f(x)}{\partial x}\right|\right] = 2\log\left(\frac{\sigma}{w}\right) - \log 2 - \gamma$$

$$v_{\log} = \mathbb{V}\left[\log\left|\frac{\partial f(x)}{\partial x}\right|\right] = \frac{\pi^2}{4} + \frac{\log^2 2}{2} - \gamma^2 - \gamma\log 4 + 2\log\left(\frac{\sigma}{w}\right)\left[\gamma + \log 2 - \log\left(\frac{\sigma}{w}\right)\right] \quad (11)$$

where $\gamma \approxeq 0.5772$ is Euler's constant. Since the second moment is finite, by the central limit theorem, the limiting distribution of the size of the gradient approaches log-normal as L grows:

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

$$\implies \log\left|\frac{\partial f^{(1:L)}(x)}{\partial x}\right| \overset{L\to\infty}{\sim} \mathcal{N}\left(Lm_{\log}, L^2 v_{\log}\right) \quad (12)$$

Even if the expected magnitude of the derivative remains constant, the variance of the log-normal distribution grows without bound as the depth increases. Because the log-normal distribution is heavy-tailed and its domain is bounded below by zero, the derivative will become very small almost everywhere, with rare but very large jumps.

Figure 2 shows this behavior in a draw from a 1D deep GP prior, at varying depths. This figure also shows that once the derivative in one region of the input space becomes very large or very small, it is likely to remain that way in subsequent layers.

3.2 Distribution of the Jacobian

We now derive the distribution on Jacobians of multivariate functions drawn from a deep GP prior.

In the case of the multivariate squared-exp kernel, the covariance between derivatives has the form:

$$f(\mathbf{x}) \sim \mathrm{GP}\left(0, \sigma^2 \prod_{d=1}^{D} \exp\left(-\frac{1}{2}\frac{(x_d - x'_d)^2}{w_d^2}\right)\right)$$

$$\implies \mathrm{cov}\left(\frac{\partial f(\mathbf{x})}{\partial x_{d_1}}, \frac{\partial f(\mathbf{x})}{\partial x_{d_2}}\right) = \begin{cases} \frac{\sigma^2}{w_{d_1}^2} & \text{if } d_1 = d_2 \\ 0 & \text{if } d_1 \neq d_2 \end{cases} \quad (14)$$

Lemma 3.2. *The Jacobian of a set of D functions $\mathbb{R}^D \to \mathbb{R}$ drawn independently from a GP prior with a product kernel is a $D \times D$ matrix of independent Gaussian R.V.'s*

Proof. The Jacobian of the vector-valued function $\mathbf{f}(\mathbf{x})$ is a matrix J with elements $J_{ij} = \frac{\partial \mathbf{f}_i(\mathbf{x})}{\partial x_j}$. Because we've assumed that the GPs on each output dimension $\mathbf{f}_d(\mathbf{x})$ are independent (2), it follows that each row of J is independent. Lemma 3.1 shows that the elements of each row are independent Gaussian. Thus all entries in the Jacobian of a GP-distributed transform are independent Gaussian R.V.'s. □

Theorem 3.3. *The Jacobian of a deep GP with a product kernel is a product of independent Gaussian matrices, with each entry in each matrix being drawn independently.*

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

$$J^{1:L}(\mathbf{x}) = J^L J^{(L-1)} \ldots J^3 J^2 J^1. \quad (15)$$

By lemma 3.2, each $J^\ell_{i,j} \overset{\text{ind}}{\sim} \mathcal{N}$, so the complete Jacobian is a product of independent Gaussian matrices, with each entry of each matrix drawn independently. □

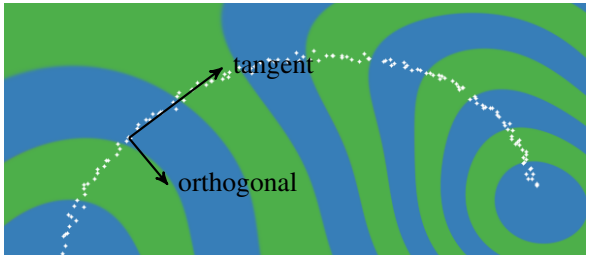


Figure 3: Representing a 1-D manifold. Colors show the output of the computed representation as a function of the input space. The representation (blue & green) is invariant in directions orthogonal to the data manifold (white), making it robust to noise in those directions, and reducing the number of parameters needed to represent a datapoint. The representation also changes in directions tangent to the manifold, preserving information for later layers.

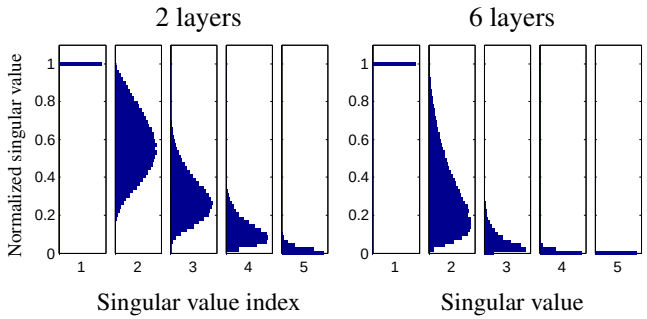


Figure 4: The normalized singular value spectrum of the Jacobian of a deep GP. As the net gets deeper, the largest singular value dominates. This implies that with high probability, there is only one effective degree of freedom in the representation being computed.

the priors we are examining are stationary, the distribution of the Jacobian is identical everywhere. Figure 4 shows the singular value spectrum for 5-dimensional deep GPs of different depths. As the net gets deeper, the largest singular

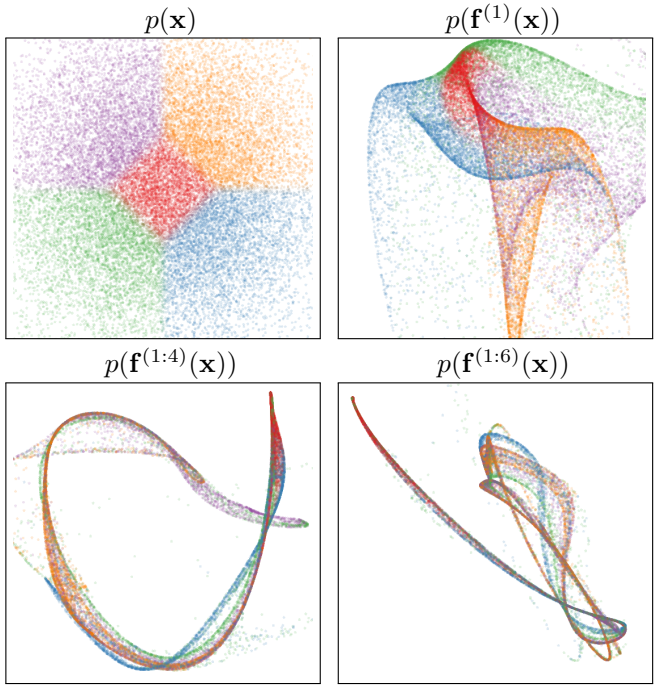


Figure 5: Visualization of draws from a deep GP. A 2-dimensional Gaussian distribution (top left) is warped by successive functions drawn from a GP prior. As the number of layers increases, the density concentrates along one-dimensional filaments.

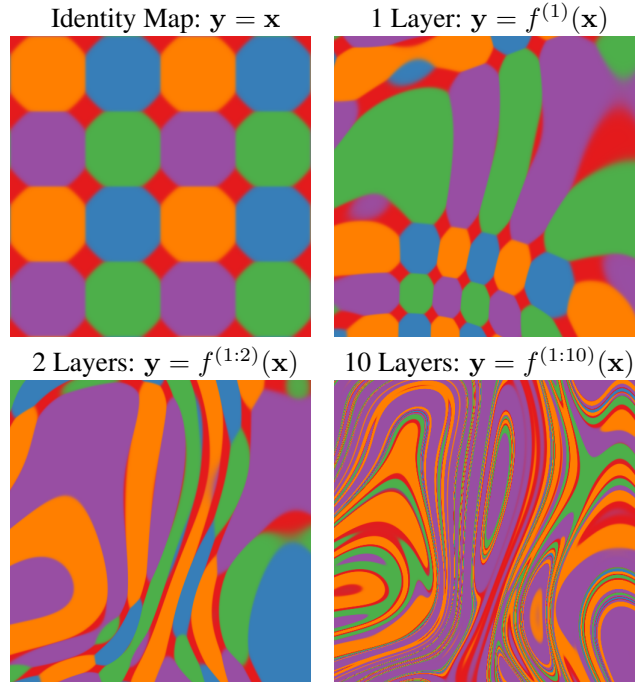
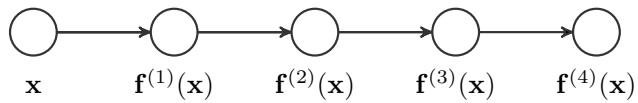


Figure 6: Feature mapping of a deep GP. Colors correspond to the location $\mathbf{y} = \mathbf{f}(\mathbf{x})$ that each point is mapped to after being warped by a deep GP. The number of directions in which the color changes rapidly corresponds to the number of large singular values in the Jacobian. Just as the densities in figure 5 became locally one-dimensional, there is usually only one direction that one can move $\mathbf{x}$ in locally to change $\mathbf{y}$. This means that $\mathbf{f}$ is unlikely to be a suitable representation for decision tasks that depend on more than one aspect of $\mathbf{x}$.

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, the singular value spectrum remained relatively flat for hundreds of layers as long as $D > 100$. Thus, these pathologies are unlikely to severely affect relatively shallow, wide networks.

5 Fixing the pathology

Following a suggestion from Neal (1995), we can fix the pathologies exhibited in figures 5 and 6 by simply mak-



a) The standard MLP connectivity architecture.

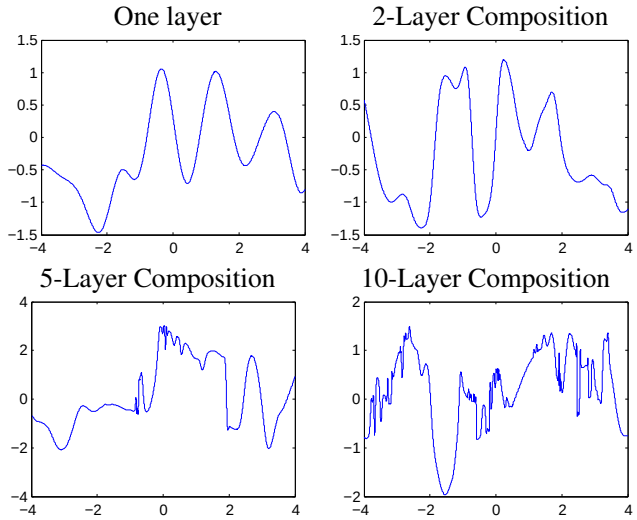


Figure 8: Draws from a 1D deep GP prior with each layer connected to the input. Even after many layers, the functions remain smooth in some regions, while varying rapidly in other regions. Compare to standard-connectivity deep GP draws shown in figure 2.

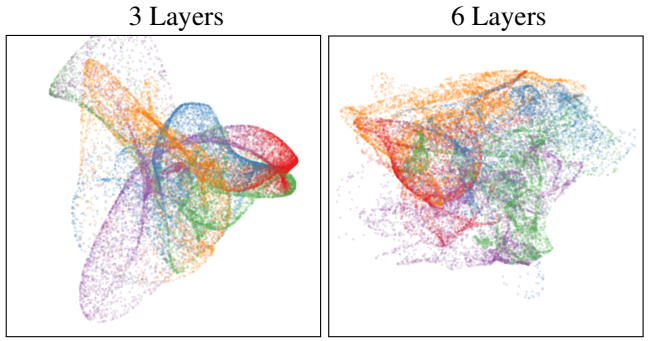


Figure 9: Left: Densities defined by a draw from a deep GP, with each layer connected to the input $\mathbf{x}$. As depth increases, the density becomes more complex without concentrating along filaments.

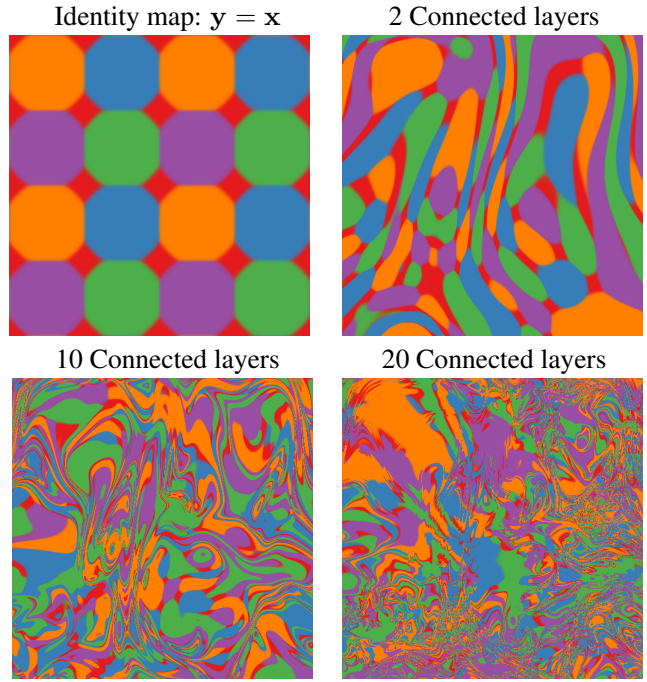


Figure 11: Feature mapping of a deep GP with each layer connected to the input $\mathbf{x}$. Just as the densities in figure 9 remained locally two-dimensional even after many transformations, in this mapping there are often two directions that one can move locally in $\mathbf{x}$ to in order to change the values of $\mathbf{f}(\mathbf{x})$. This means that the prior puts mass on representations which sometimes depend on all aspects of the input. Compare to figure 6.

6 Deep kernels

Bengio *et al.* (2006) 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

ping $\mathbf{h}(\mathbf{x})$ has a simple closed form:

$$k_{L+1}(\mathbf{x}, \mathbf{x}') = k_{SE}(\mathbf{h}(\mathbf{x}), \mathbf{h}(\mathbf{x}')) = \quad (22)$$
$$= \exp\left(-\frac{1}{2}||\mathbf{h}(\mathbf{x}) - \mathbf{h}(\mathbf{x}')||_2^2\right)$$
$$= \exp\left(-\frac{1}{2}\left[\mathbf{h}(\mathbf{x})^\mathsf{T}\mathbf{h}(\mathbf{x}) - 2\mathbf{h}(\mathbf{x})^\mathsf{T}\mathbf{h}(\mathbf{x}') + \mathbf{h}(\mathbf{x}')^\mathsf{T}\mathbf{h}(\mathbf{x}')\right]\right)$$
$$= \exp\left(-\frac{1}{2}\left[k_L(\mathbf{x}, \mathbf{x}) - 2k_L(\mathbf{x}, \mathbf{x}') + k_L(\mathbf{x}', \mathbf{x}')\right]\right)$$

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:

$$k_{L+1}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\left|\left|\begin{bmatrix}\mathbf{h}_L(\mathbf{x}) \\ \mathbf{x}\end{bmatrix} - \begin{bmatrix}\mathbf{h}_L(\mathbf{x}') \\ \mathbf{x}'\end{bmatrix}\right|\right|_2^2\right)$$
$$= \exp\Big(-\frac{1}{2}\big[k_L(\mathbf{x}, \mathbf{x}) - 2k_L(\mathbf{x}, \mathbf{x}') + k_L(\mathbf{x}', \mathbf{x}') - ||\mathbf{x} - \mathbf{x}'||_2^2\big]\Big) \quad (23)$$

For the SE kernel, this repeated mapping satisfies

$$k_\infty(\mathbf{x}, \mathbf{x}') - \log\left(k_\infty(\mathbf{x}, \mathbf{x}')\right) = 1 + \frac{1}{2}||\mathbf{x} - \mathbf{x}'||_2^2 \quad (24)$$

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

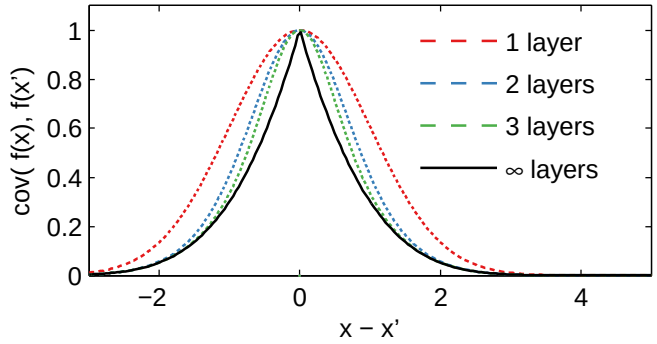


Figure 12: Input-connected deep kernels. By connecting the inputs $\mathbf{x}$ to each layer, the kernel can still depend on its input even after arbitrarily many layers of computation.

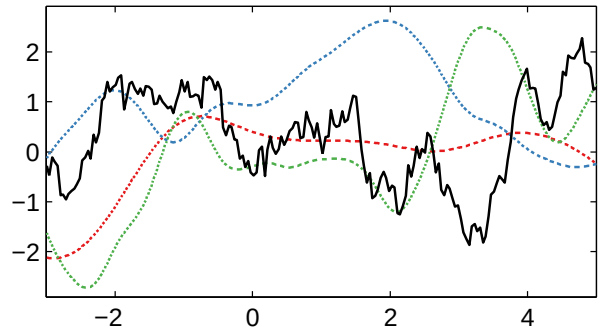


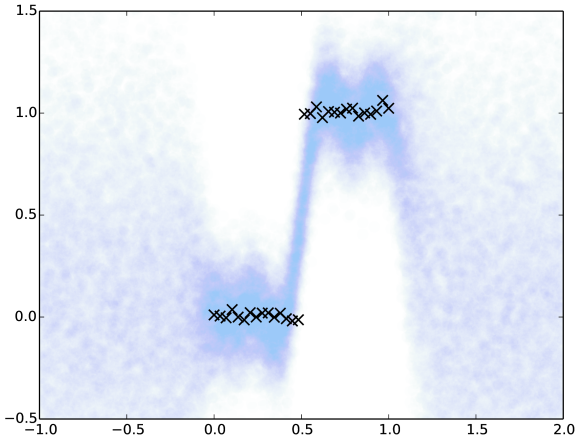
Figure 13: GP draws using deep input-connected kernels.

7 Dropout in Gaussian processes

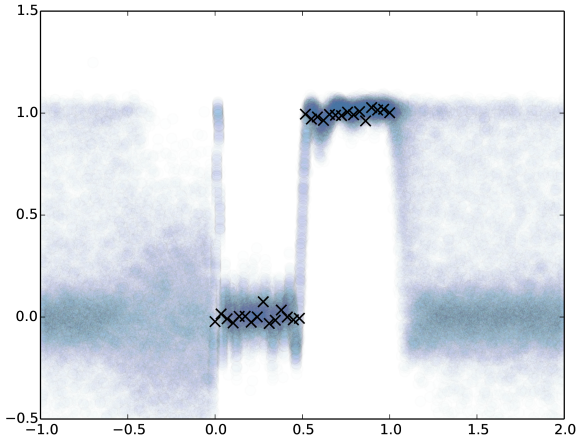
Dropout is a method for regularizing neural networks (Hinton *et al.*, 2012; Srivastava, 2013). Training with dropout entails randomly and independently "dropping" (setting to zero) some proportion p of features or inputs, in order to improve the robustness of the resulting network by reducing co-dependence between neurons. To maintain similar overall activation levels, weights are multiplied by $1/p$ at test time. Alternatively, feature activations are multiplied

Outline

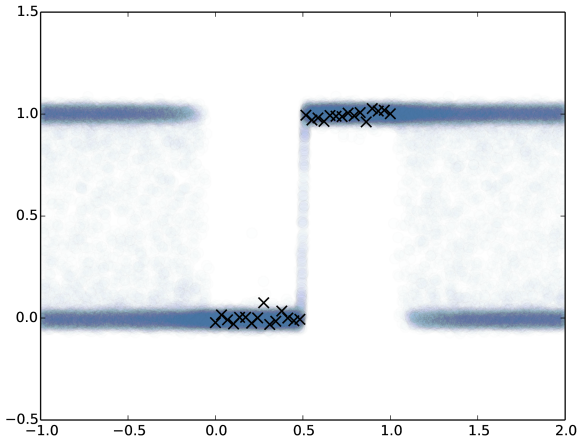
Derivative Tails Increase with Layers: Step Function



(a) GP

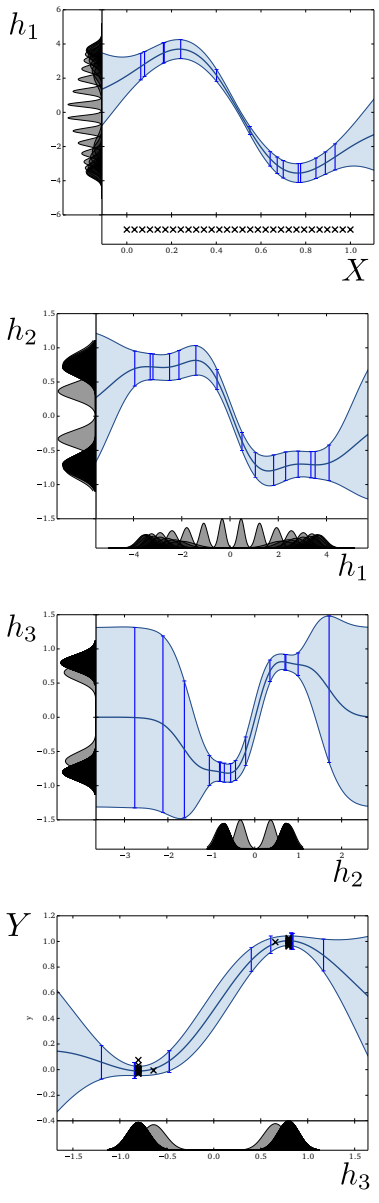


(b) 2 layers

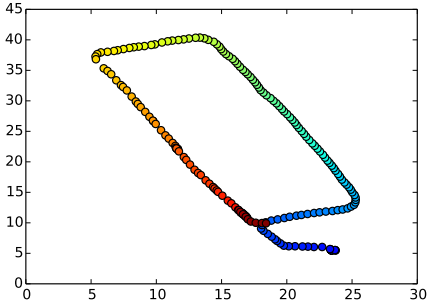


(c) 4 layers

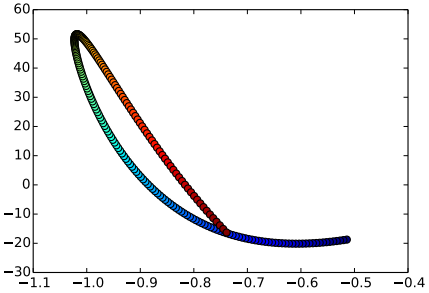
Values in Hidden Layers



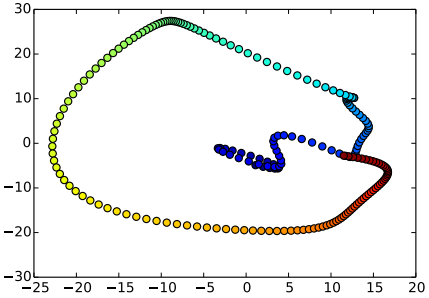
Loop Detection in Robotics



(d) True path



(e) Hidden layer 1



(f) Hidden layer 2

- Dynamically constrained model
- Correctly detects the loop
- Learns temporal continuity and corner-like features in different layers

Data fit for Loop Closure

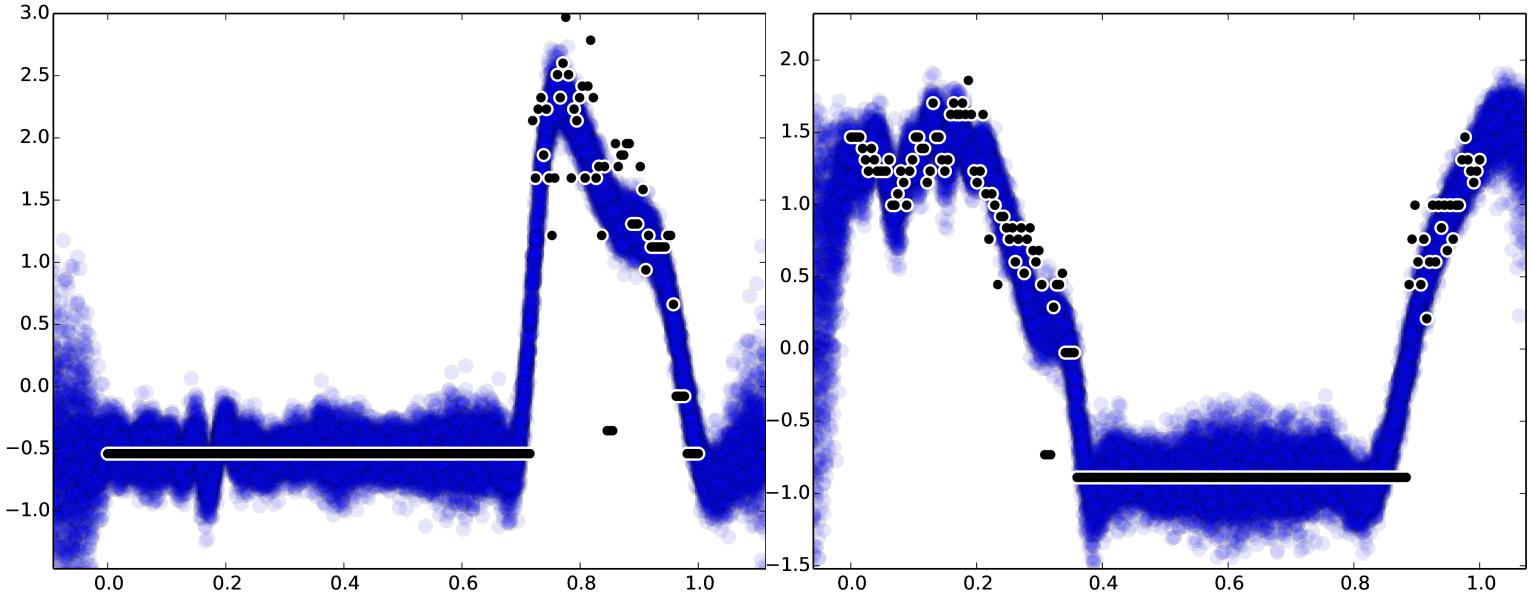
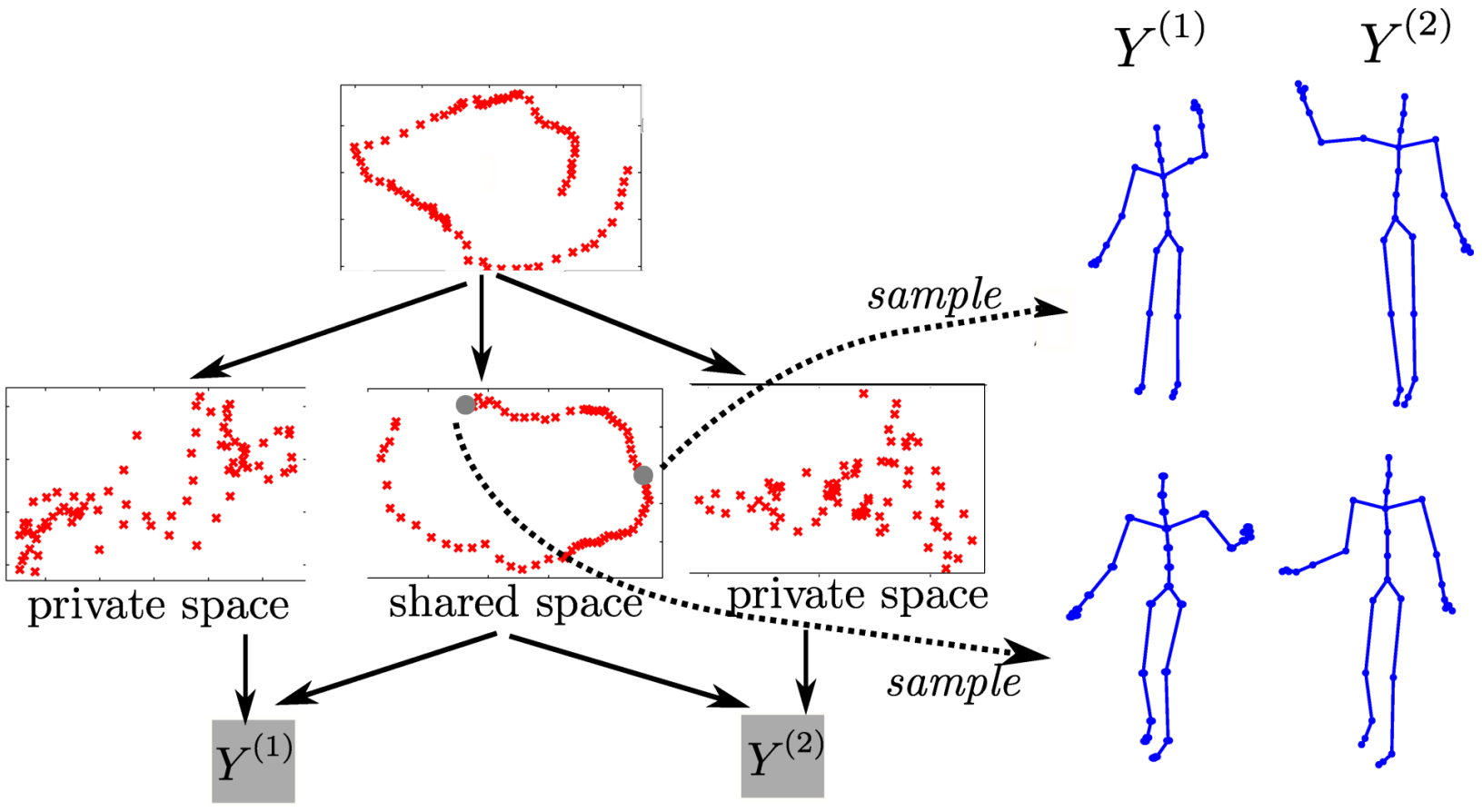


Figure : Example data fits for 2 of the 30 output dimensions

Motion Capture

- 'High five' data.
- Model learns structure between two interacting subjects.

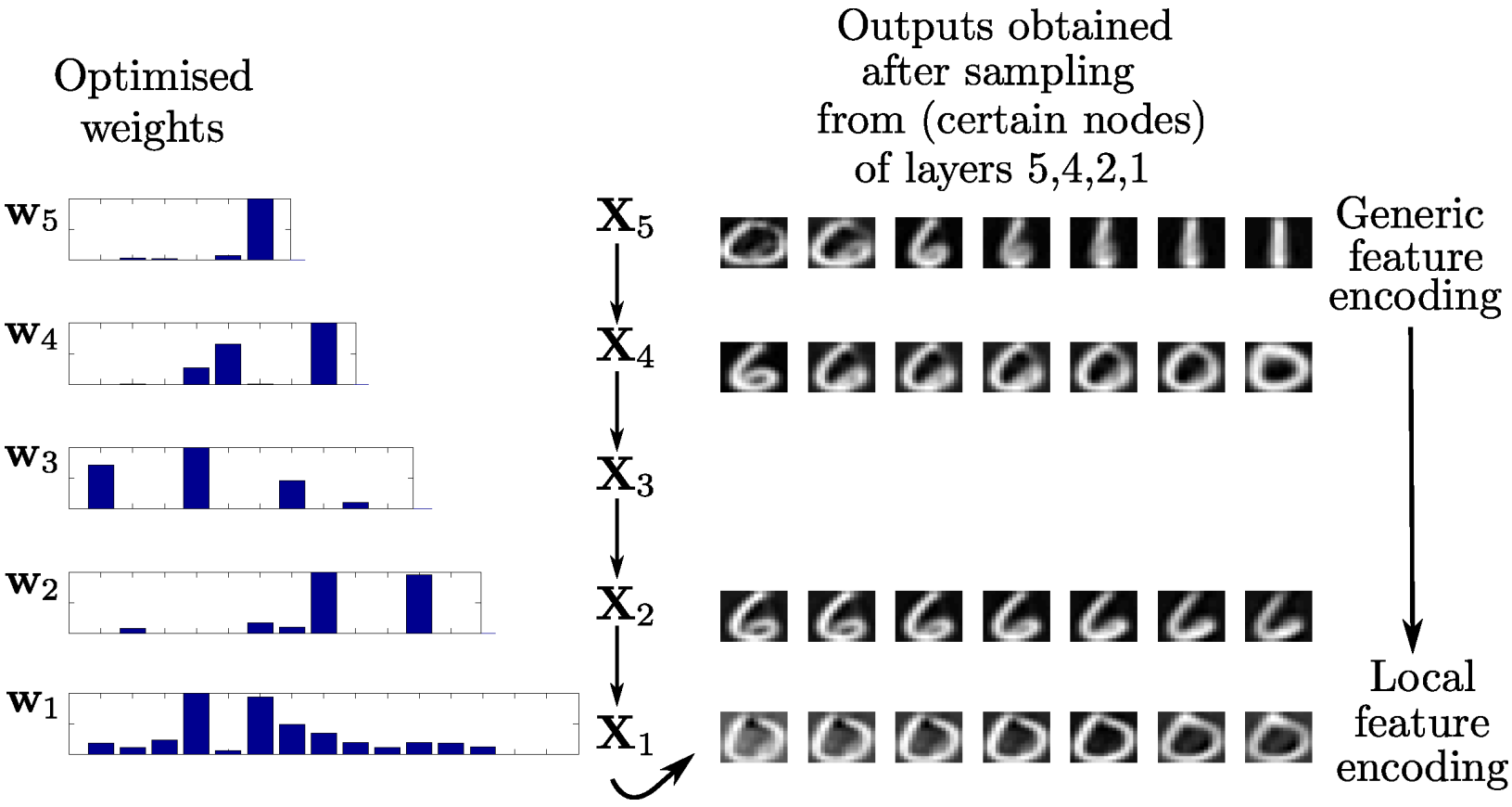
Deep hierarchies – motion capture



Digits Data Set

- Are deep hierarchies justified for small data sets?
- We can lower bound the evidence for different depths.
- For 150 6s, 0s and 1s from MNIST we found at least 5 layers are required.

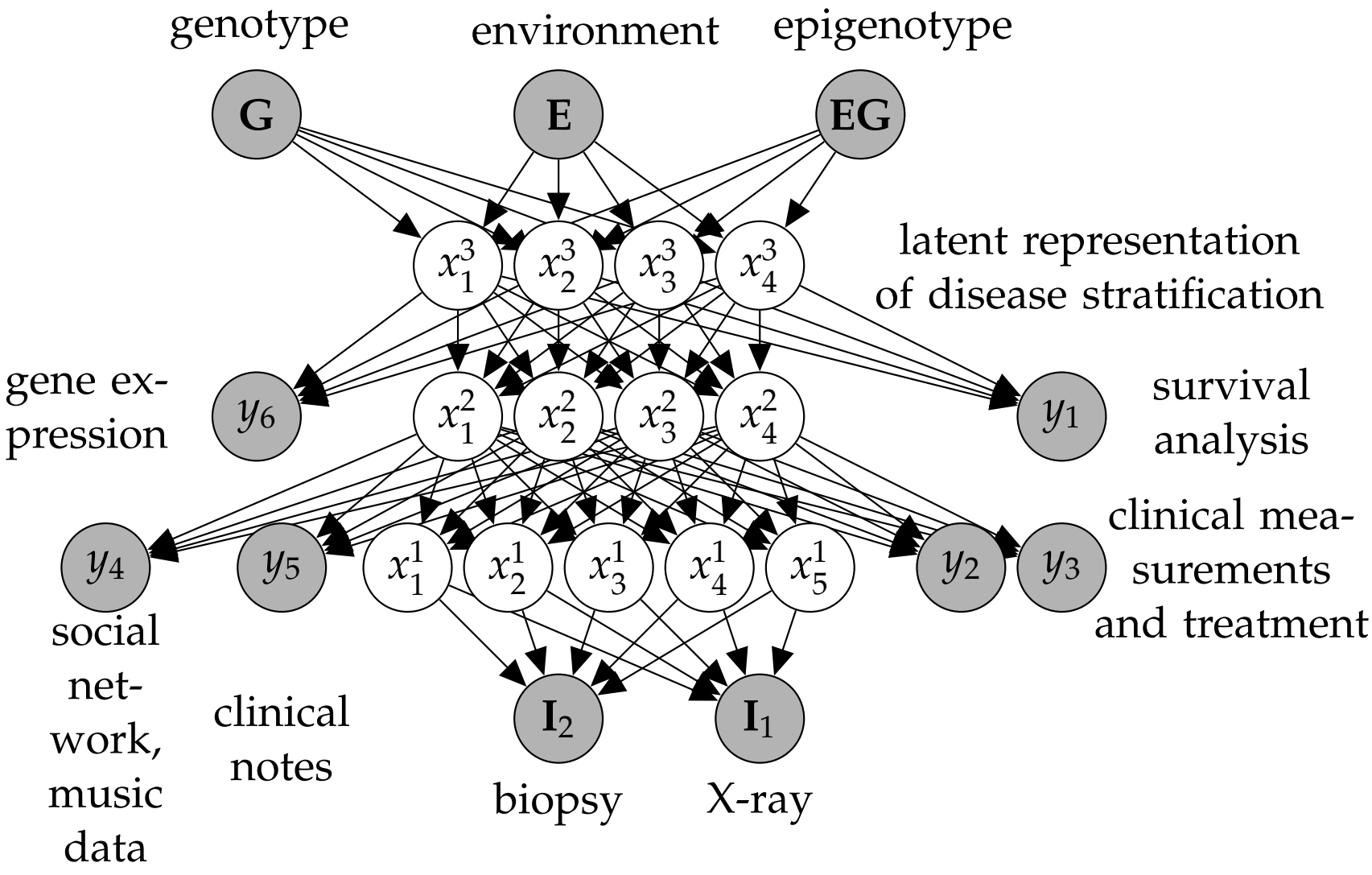
Deep hierarchies – MNIST



What Can We Do that Google Can't?

- Google's resources give them access to volumes of data (or Facebook, or Microsoft, or Amazon).
- Is there anything for Universities to contribute?
- Assimilation of multiple views of the patient: each perhaps from a different patient.
- This may be done by small companies (with support of Universities).
- A Facebook app for your personalised health.
- These methodologies are part of that picture.

Deep Health


genotype
environment
epigenotype
G
E
EG
x_1^3 x_2^3 x_3^3 x_4^3
latent representation of disease stratification
gene expression
y_6
x_1^2 x_2^2 x_3^2 x_4^2
y_1
survival analysis
y_4
y_5
x_1^1 x_2^1 x_3^1 x_4^1 x_5^1
y_2
y_3
clinical measurements and treatment
social network, music data
clinical notes
I_2
I_1
biopsy
X-ray

Summary

- Deep Gaussian Processes allow unsupervised and supervised deep learning.
- They can be easily adapted to handle multitask learning.
- Data dimensionality turns out to not be a computational bottleneck.
- Variational compression algorithms show promise for scaling these models to *massive* data sets.

References I

M. A. Álvarez, D. Luengo, M. K. Titsias, and N. D. Lawrence. Efficient multioutput Gaussian processes through variational inducing kernels. In Y. W. Teh and D. M. Titterington, editors, *Proceedings of the Thirteenth International Workshop on Artificial Intelligence and Statistics*, volume 9, pages 25–32, Chia Laguna Resort, Sardinia, Italy, 13-16 May 2010. JMLR W&CP 9. [PDF].

Y. Bengio. Learning Deep Architectures for AI. *Found. Trends Mach. Learn.*, 2 (1):1–127, Jan. 2009. ISSN 1935-8237. [DOI].

L. Csató and M. Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

A. Damianou and N. D. Lawrence. Deep Gaussian processes. In C. Carvalho and P. Ravikumar, editors, *Proceedings of the Sixteenth International Workshop on Artificial Intelligence and Statistics*, volume 31, AZ, USA, 2013. JMLR W&CP 31. [PDF].

D. Duvenaud, O. Rippel, R. Adams, and Z. Ghahramani. Avoiding pathologies in very deep networks. In S. Kaski and J. Corander, editors, *Proceedings of the Seventeenth International Workshop on Artificial Intelligence and Statistics*, volume 33, Iceland, 2014. JMLR W&CP 33.

References II

Y. Gal, M. van der Wilk, and C. E. Rasmussen. Distributed variational inference in sparse Gaussian process regression and latent variable models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, Cambridge, MA, 2014.

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In A. Nicholson and P. Smyth, editors, *Uncertainty in Artificial Intelligence*, volume 29. AUAI Press, 2013. [PDF].

J. Hensman, M. Rattray, and N. D. Lawrence. Fast variational inference in the conjugate exponential family. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, Cambridge, MA, 2012. [PDF].

G. E. Hinton and S. Osindero. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:2006, 2006.

M. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. Technical report,

N. J. King and N. D. Lawrence. Fast variational inference for Gaussian Process models through KL-correction. In *ECML, Berlin, 2006*, Lecture Notes in Computer Science, pages 270–281, Berlin, 2006. Springer-Verlag. [PDF].

References III

N. D. Lawrence. Learning for larger datasets with the Gaussian process latent variable model. In M. Meila and X. Shen, editors, *Proceedings of the Eleventh International Workshop on Artificial Intelligence and Statistics*, pages 243–250, San Juan, Puerto Rico, 21-24 March 2007. Omnipress. [PDF].

T. K. Leen, T. G. Dietterich, and V. Tresp, editors. *Advances in Neural Information Processing Systems*, volume 13, Cambridge, MA, 2001. MIT Press.

T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

J. Quiñonero Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In S. Roweis and A. McCallum, editors, *Proceedings of the International Conference in Machine Learning*, volume 25, pages 872–879. Omnipress, 2008.

M. Seeger, C. K. I. Williams, and N. D. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In C. M. Bishop and B. J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, Key West, FL, 3–6 Jan 2003.

References IV

A. J. Smola and P. L. Bartlett. Sparse greedy Gaussian process regression. In Leen et al. (2001), pages 619–625.

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18, Cambridge, MA, 2006. MIT Press.

M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In D. van Dyk and M. Welling, editors, *Proceedings of the Twelfth International Workshop on Artificial Intelligence and Statistics*, volume 5, pages 567–574, Clearwater Beach, FL, 16-18 April 2009. JMLR W&CP 5.

C. K. I. Williams, C. E. Rasmussen, A. Schwaighofer, and V. Tresp. Observations of the Nyström method for Gaussian process prediction. Technical report, University of Edinburgh,

C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In Leen et al. (2001), pages 682–688.