Deep Gaussian Processes

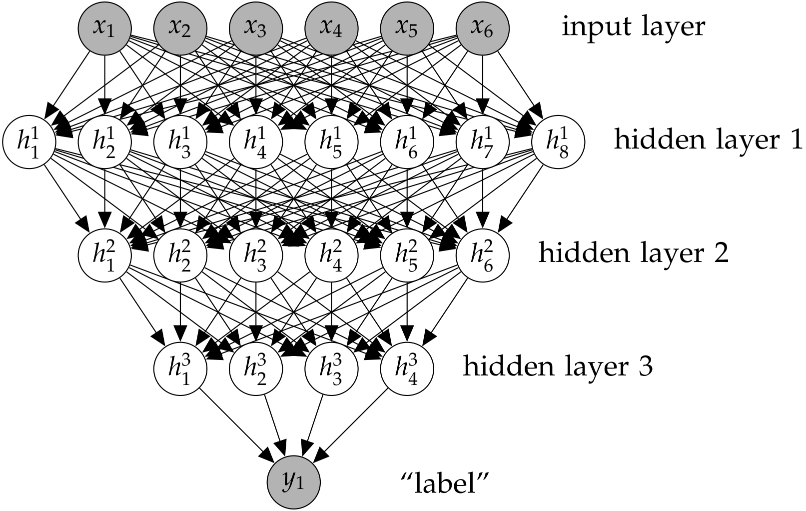
Neil D. Lawrence

30th April 2015
KTH Royal Institute of Technology

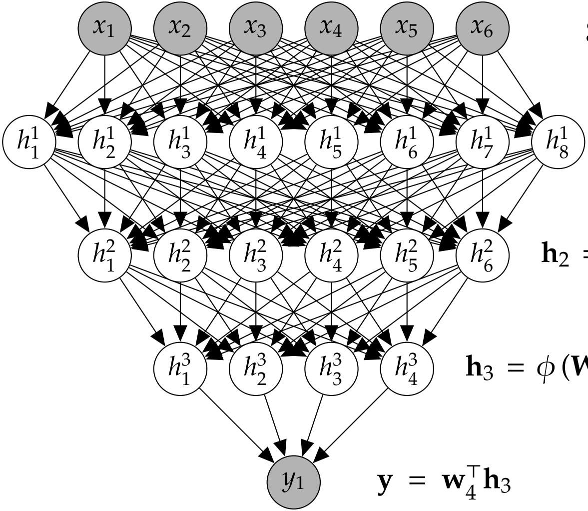
Outline

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

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

Overfitting

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

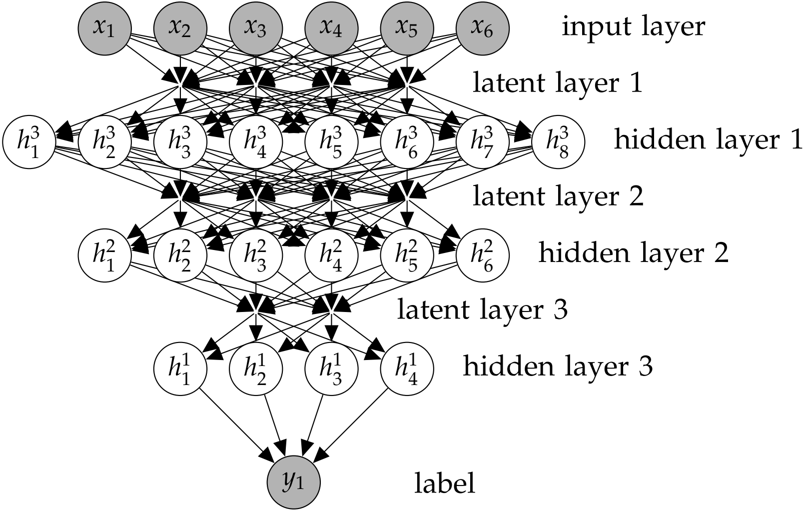
$$\mathbf{W} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{V}^\top$$

or

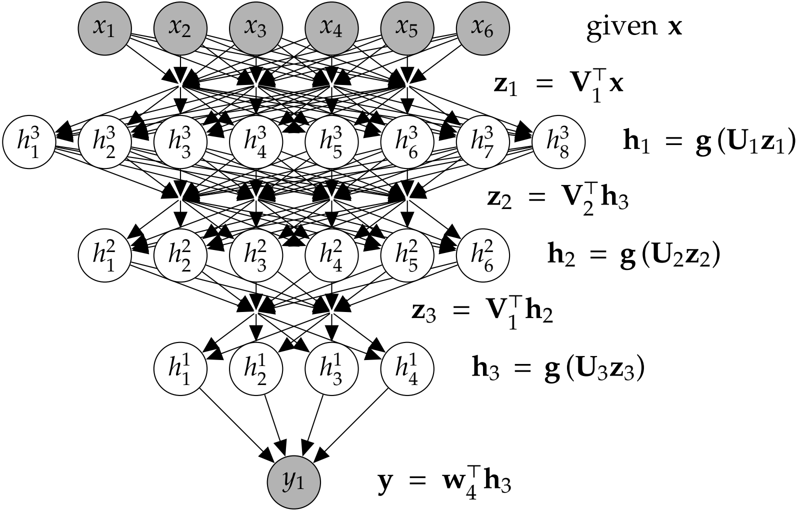
$$\mathbf{W} = \mathbf{U}\mathbf{V}^\top$$

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

Deep Neural Network



Deep Neural Network



Mathematically

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

A Cascade of Neural Networks

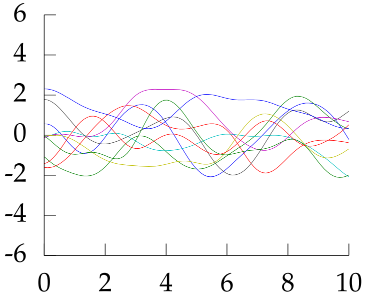
$$\begin{aligned}
\mathbf{z}_1 &= \mathbf{V}_1^\top \mathbf{x} \\
\mathbf{z}_2 &= \mathbf{V}_2^\top \phi\left(\mathbf{U}_1 \mathbf{z}_1\right) \\
\mathbf{z}_3 &= \mathbf{V}_3^\top \phi\left(\mathbf{U}_2 \mathbf{z}_2\right) \\
\mathbf{y} &= \mathbf{w}_4^\top \mathbf{z}_3
\end{aligned}$$

Replace Each Neural Network with a Gaussian Process

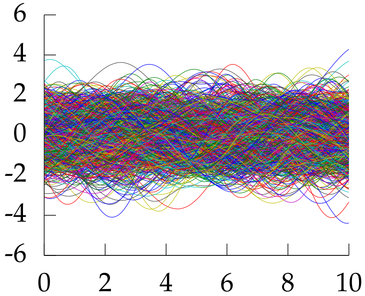
$$
\begin{aligned}
\mathbf{z}_1 &= \mathbf{f}(\mathbf{x}) \\
\mathbf{z}_2 &= \mathbf{f}(\mathbf{z}_1) \\
\mathbf{z}_3 &= \mathbf{f}(\mathbf{z}_2) \\
\mathbf{y} &= \mathbf{f}(\mathbf{z}_3)
\end{aligned}
$$

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

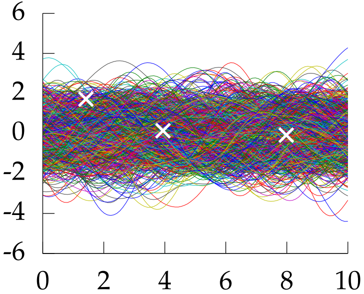
Gaussian Processes: Extremely Short Overview



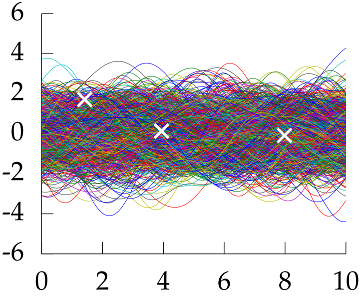
Gaussian Processes: Extremely Short Overview



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.

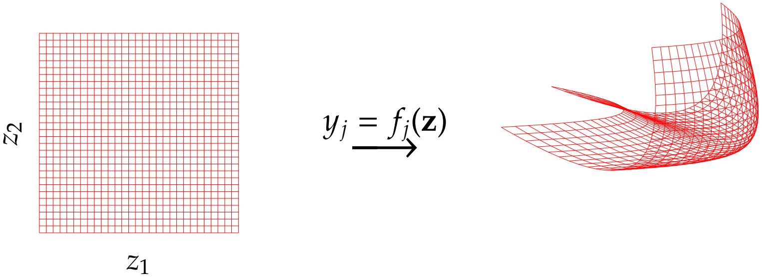


Figure : A three dimensional manifold formed by mapping from a two dimensional space to a three dimensional space.

Difficulty for Probabilistic Approaches

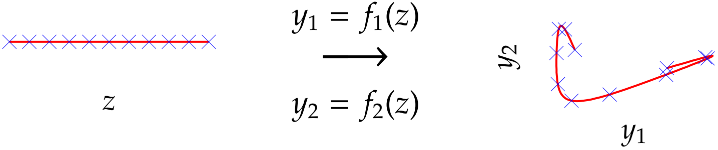


Figure : A string in two dimensions, formed by mapping from one dimension, z, line to a two dimensional space, $[y_1, \; y_2]$ using nonlinear functions $f_1(\cdot)$ and $f_2(\cdot)$.

Difficulty for Probabilistic Approaches

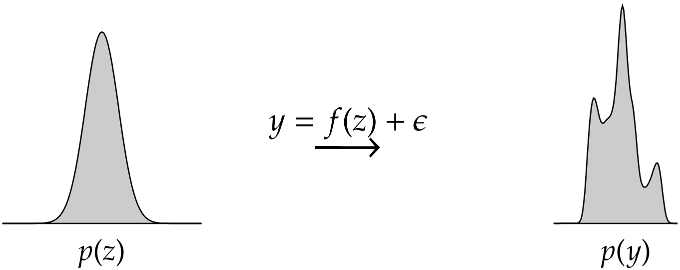


Figure : A Gaussian distribution propagated through a non-linear mapping. $y_i = f(z_i) + \epsilon_i$. $\epsilon \sim \mathcal{N}\left(0, 0.2^2\right)$ and $f(\cdot)$ uses RBF basis, 100 centres between -4 and 4 and $\ell = 0.1$. New distribution over y (right) is multimodal and difficult to normalize.

Variational Compression

(Snelson and Ghahramani, 2006; Quiñonero Candela and Rasmussen, 2005; Lawrence, 2007; Titsias, 2009)

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

Variational Compression

(Snelson and Ghahramani, 2006; Quiñonero Candela and Rasmussen, 2005; Lawrence, 2007; Titsias, 2009)

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

Variational Compression

(Snelson and Ghahramani, 2006; Quiñonero Candela and Rasmussen, 2005; Lawrence, 2007; Titsias, 2009)

- Complexity of standard GP:
  - $\mathcal{O}(n^3)$ in computation.
  - $\mathcal{O}(n^2)$ in storage.
- Via low rank representations of covariance:
  - $\mathcal{O}(nm^2)$ in computation.
  - $\mathcal{O}(nm)$ 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 are like pseudo-data. They can be in space of $\mathbf{f}$ or a space that is related through a linear operator (Álvarez et al., 2010) — e.g. a gradient or convolution.
- There are inducing variables associated with each set of hidden variables, $\mathbf{z}^i$.

Variational Compression II

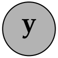
- **Importantly** conditioning on inducing variables renders the likelihood independent across the data.
- It turns out that this allows us to variationally handle uncertainty on the kernel (including the inputs to the kernel).
- It also allows standard scaling approaches: stochastic variational inference Hensman et al. (2013), parallelization Gal et al. (2014) and work by Zhenwen Dai on GPUs to be applied: an *engineering* challenge?

Outline

Variational Compression

Model for our data, $\mathbf{y}$.

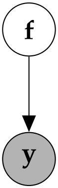
$$p(\mathbf{y})$$



Variational Compression

Prior density over $\mathbf{f}$. Likelihood relates data, $\mathbf{y}$, to $\mathbf{f}$.

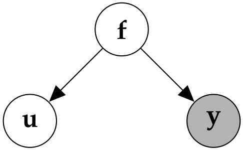
$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})\mathrm{d}\mathbf{f}$$



Variational Compression

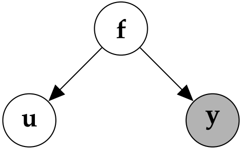
Augment standard model with a set of m new inducing variables, $\mathbf{u}$.

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



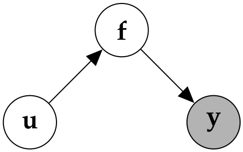
Variational Compression

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



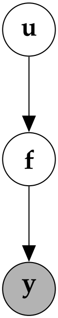
Variational Compression

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



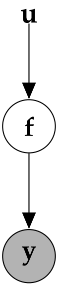
Variational Compression

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



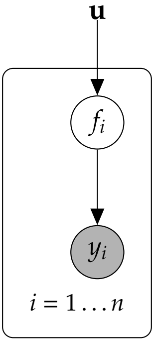
Variational Compression

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



Variational Compression

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



Variational Compression

Consider the conditional likelihood.

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

Variational Compression

Consider the conditional log likelihood.

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

Variational Compression

Introduce variational lower bound

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

Variational Compression

Set $q(\mathbf{f}) = p(\mathbf{f}|\mathbf{u})$

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

Variational Compression

Set $q(\mathbf{f}) = p(\mathbf{f}|\mathbf{u})$

$$\log p(\mathbf{y}|\mathbf{u}) \geq \sum_{i=1}^{n} \left\langle \log p(y_i|f_i) \right\rangle_{p(f_i|\mathbf{u})}$$

Variational Compression

Difference between bound and truth is KL divergence:

$$\mathrm{KL}\left(p(\mathbf{f}|\mathbf{u})\,\|\,p(\mathbf{f}|\mathbf{u},\mathbf{y})\right) = \int p(\mathbf{f}|\mathbf{u}) \log \frac{p(\mathbf{f}|\mathbf{u})}{p(\mathbf{f}|\mathbf{u},\mathbf{y})} \mathrm{d}\mathbf{f}$$

This is why we call it variational compression, information in $\mathbf{y}$ is compressed into $\mathbf{u}$

Gaussian $p(y_i|f_i)$

For Gaussian likelihoods:

$$\left\langle \log p(y_i|f_i)\right\rangle_{p(f_i|\mathbf{u})} = -\frac{1}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\left(y_i - \left\langle f_i\right\rangle\right)^2 - \frac{1}{2\sigma^2}\left(\left\langle f_i^2\right\rangle - \left\langle f_i\right\rangle^2\right)$$

Gaussian $p(y_i|f_i)$

For Gaussian likelihoods:

$$\langle \log p(y_i|f_i) \rangle_{p(f_i|\mathbf{u})} = -\frac{1}{2} \log 2\pi\sigma^2 - \frac{1}{2\sigma^2} \left(y_i - \langle f_i \rangle\right)^2 - \frac{1}{2\sigma^2} \left(\left\langle f_i^2 \right\rangle - \langle f_i \rangle^2 \right)$$

Implying:

$$p(y_i|\mathbf{u}) \geq \exp \langle \log c_i \rangle \mathcal{N}\left(y_i | \langle f_i \rangle, \sigma^2\right)$$

Gaussian Process Over $\mathbf{f}$ and $\mathbf{u}$

Define:

$$q_{i,i} = \text{var}_{p(f_i|\mathbf{u})}(f_i) = \left\langle f_i^2 \right\rangle_{p(f_i|\mathbf{u})} - \left\langle f_i \right\rangle_{p(f_i|\mathbf{u})}^2$$

We can write:

$$c_i = \exp\left(-\frac{q_{i,i}}{2\sigma^2}\right)$$

If joint distribution of $p(\mathbf{f}, \mathbf{u})$ is Gaussian then:

$$q_{i,i} = k_{i,i} - \mathbf{k}_{i,\mathbf{u}}^\top \mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{k}_{i,\mathbf{u}}$$

c_i is not a function of $\mathbf{u}$ but *is* a function of $\mathbf{X}_\mathbf{u}$.

Lower Bound on Likelihood

Substitute variational bound into marginal likelihood:

$$p(\mathbf{y}) \geq \prod_{i=1}^{n} c_i \int \mathcal{N}\left(\mathbf{y}| \left\langle \mathbf{f} \right\rangle, \sigma^2 \mathbf{I}\right) p(\mathbf{u}) \mathrm{d}\mathbf{u}$$

Note that:

$$\left\langle \mathbf{f} \right\rangle_{p(\mathbf{f}|\mathbf{u})} = \mathbf{K}_{\mathbf{f},\mathbf{u}} \mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u}$$

is *linearly* dependent on $\mathbf{u}$.

Deterministic Training Conditional

Making the marginalization of $\mathbf{u}$ straightforward. In the Gaussian case:

$$p(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right)$$

$$\int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \geq \prod_{i=1}^{n} c_i \int \mathcal{N}\left(\mathbf{y}|\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}, \sigma^2\right) \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right) \mathrm{d}\mathbf{u}$$

Deterministic Training Conditional

Making the marginalization of $\mathbf{u}$ straightforward. In the Gaussian case:

$$p(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right)$$

$$\int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \geq \prod_{i=1}^{n} c_i \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right)$$

Deterministic Training Conditional

Making the marginalization of $\mathbf{u}$ straightforward. In the Gaussian case:

$$p(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right)$$

$$\int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \geq \prod_{i=1}^{n} c_i \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right)$$

Maximize log of the bound to find covariance function parameters,

$$L \geq \sum_{i=1}^{n} \log c_i + \log \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f},}\right)$$

Deterministic Training Conditional

Making the marginalization of $\mathbf{u}$ straightforward. In the Gaussian case:

$$p(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right)$$

$$\int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \geq \prod_{i=1}^{n} c_i \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right)$$

Maximize log of the bound to find covariance function parameters,

$$L \geq \sum_{i=1}^{n} \log c_i + \log \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f},}\right)$$

Deterministic Training Conditional

Making the marginalization of $\mathbf{u}$ straightforward. In the Gaussian case:

$$p(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right)$$

$$\int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \geq \prod_{i=1}^{n} c_i \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right)$$

Maximize log of the bound to find covariance function parameters,

$$L \approx \log \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f},}\right)$$

- If the bound is normalized, the c_i terms are removed.

Deterministic Training Conditional

Making the marginalization of $\mathbf{u}$ straightforward. In the Gaussian case:

$$p(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right)$$

$$\int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \geq \prod_{i=1}^{n} c_i \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right)$$

Maximize log of the bound to find covariance function parameters,

- If the bound is normalized, the c_i terms are removed.
- This results in the projected process approximation (Rasmussen and Williams, 2006) or DTC (Quiñonero Candela and Rasmussen, 2005). Proposed by (Smola and Bartlett, 2001; Seeger et al., 2003; Csató and Opper, 2002; Csató, 2002).

Relationship to Nyström Approximation

- Variational lower bound leads to Nyström style approximation (Williams and Seeger, 2001; Seeger et al., 2003). 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

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

Marginalising Latent Variables

- Integrating out $\mathbf{Z}$ becomes possible variationally, because Gaussian expectations of

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

  are now *tractable*
- Relies on computing expectations of $\mathbf{K}_{\mathbf{fu}}$ and $\mathbf{K}_{\mathbf{uf}}\mathbf{K}_{\mathbf{fu}}$ under Gaussian density over $\mathbf{Z}$.

Apply Variational Inference Before Integration of $\mathbf{u}$

$$\int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \geq \prod_{i=1}^{n} c_i \int \mathcal{N}\left(\mathbf{y}|\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}, \sigma^2\right) \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right) \mathrm{d}\mathbf{u}$$

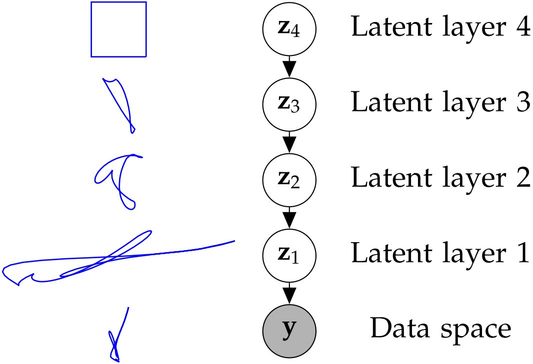
Apply Variational Inference Before Integration of $\mathbf{u}$

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

$$\int q(\mathbf{z}) \log \frac{\prod_{i=1}^{n} c_i \mathcal{N}\left(\mathbf{y}|\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}, \sigma^2\right) \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right)}{q(\mathbf{z})} \mathrm{d}\mathbf{u}$$

Outline

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

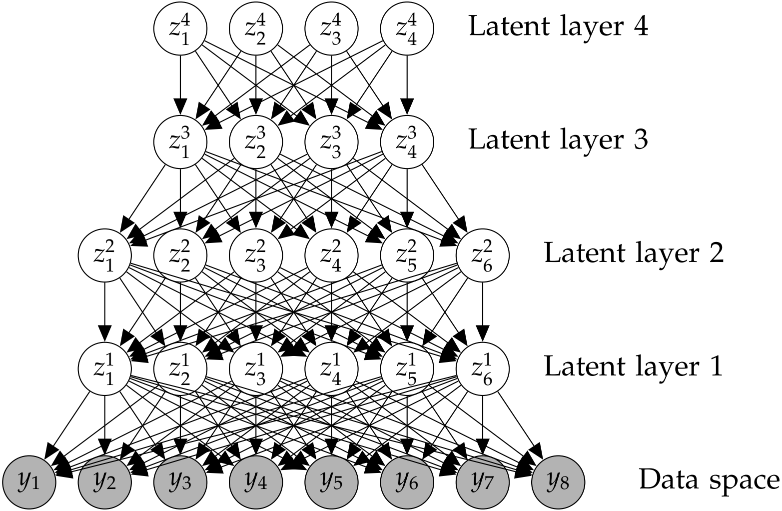
Abstract

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.

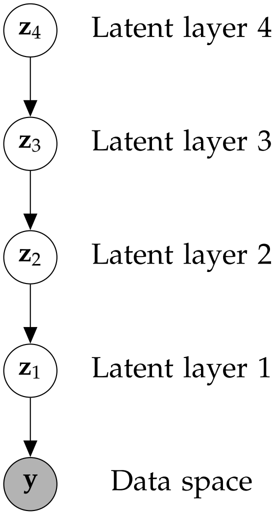
The traditional approach to deep learning is based around binary latent variables and the restricted Boltzmann machine (RBM) [Hinton, 2010]. Deep hierarchies are constructed by stacking these models and various approximate inference techniques (such as contrastive divergence)

Deep Models



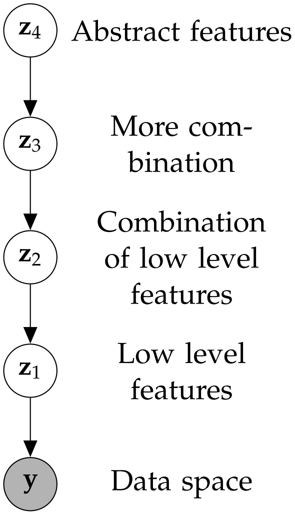
z_1^4 z_2^4 z_3^4 z_4^4
Latent layer 4
z_1^3 z_2^3 z_3^3 z_4^3
Latent layer 3
z_1^2 z_2^2 z_3^2 z_4^2 z_5^2 z_6^2
Latent layer 2
z_1^1 z_2^1 z_3^1 z_4^1 z_5^1 z_6^1
Latent layer 1
y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8
Data space

Deep Models



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

Deep Models



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

Deep Gaussian Processes

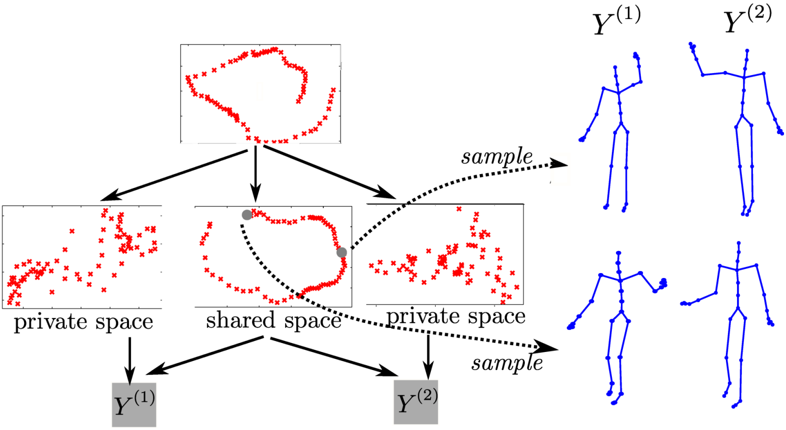
Damianou and Lawrence (2013)

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

Motion Capture

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

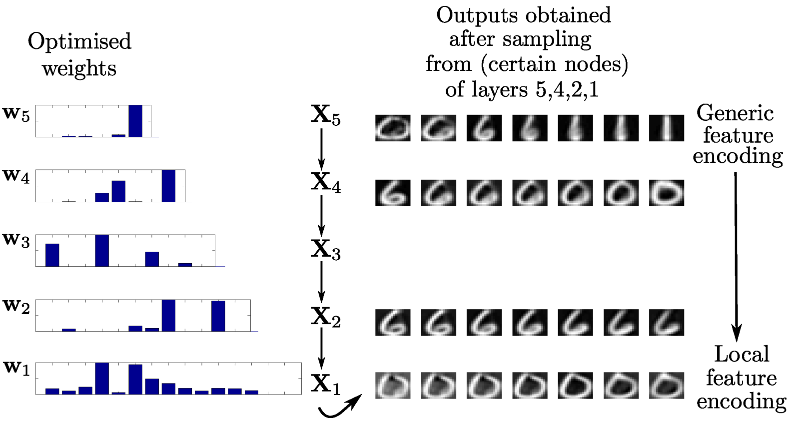
Deep hierarchies – motion capture



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



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. *Gaussian Processes — Iterative Sparse Approximations*. PhD thesis, Aston University, 2002.

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].

References II

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.

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].

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

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].

References III

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.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006. [Google Books] .

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.

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

References IV

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.