UNIVERSITY OF CAMBRIDGE

A Review of Deep Models for Density Estimation

Andrew McHutchon & Mark van der Wilk

Thursday 13 March, 2014

Introduction

- Justifying deep architectures (intuition and a theorem)
- Where it started: Deep RBMs
- The Bayesian approach: Deep GPs
- A different approach: Deep Density Models
- A hybrid: Deep Latent Gaussian Models
- Tying it together

Justifying deep architectures

UNIVERSITY OF CAMBRIDGE

- Some functions cannot be “efficiently” represented by shallow architectures
- Shallow architectures generalise locally, while deep representations allow more global generalisation
- Multi-task learning. Low level features can be useful for several tasks.
- Empirical performance

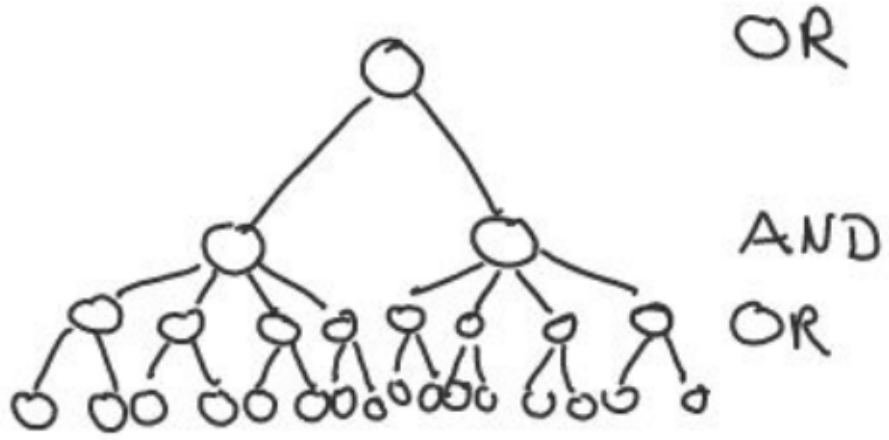
[Bengio, 2009]

Efficiency: Linear threshold networks

UNIVERSITY OF CAMBRIDGE

Theorem

A monotone weighted threshold circuit of depth $k-1$ computing a function $f_k \in \mathcal{F}_{k,N}$ has at least size 2^{cN} for some constant $c > 0$ and $N > N_o$. A depth k circuit needs only polynomial size. [Håstad and Goldmann, 1991]



- Proof is limited to one type of NN and one class of functions... But it does hint that there can be some advantage to using deep architectures.
- Is this relevant to "infinite capacity" non-parametric models (GP's)?

Generalisation

Bengio: *Shallow architectures (kernel learning) generalise locally, while deep representations allow more global generalisation.*

To a certain extent, this is a simplification of something we already know.

- Bengio's point: Squared Exponential is very widespread in use, but provides only local generalisation.
- Good features can greatly improve generalisation.
- A feature mapping can be incorporated into a the kernel (like the periodic kernel).
- Finding good features/kernels is hard
- Learning deep representations may be a way of doing this.

Generalisation

Bengio: *Shallow architectures (kernel learning) generalise locally, while deep representations allow more global generalisation.*

To a certain extent, this is a simplification of something we already know.

- Bengio's point: Squared Exponential is very widespread in use, but provides only local generalisation.
- Good features can greatly improve generalisation.
- A feature mapping can be incorporated into a the kernel (like the periodic kernel).
- Finding good features/kernels is hard
- Learning deep representations may be a way of doing this.

Generalisation

Bengio: *Shallow architectures (kernel learning) generalise locally, while deep representations allow more global generalisation.*

To a certain extent, this is a simplification of something we already know.

- Bengio's point: Squared Exponential is very widespread in use, but provides only local generalisation.
- Good features can greatly improve generalisation.
- A feature mapping can be incorporated into a the kernel (like the periodic kernel).
- Finding good features/kernels is hard
- Learning deep representations may be a way of doing this.

Generalisation

Bengio: *Shallow architectures (kernel learning) generalise locally, while deep representations allow more global generalisation.*

To a certain extent, this is a simplification of something we already know.

- Bengio's point: Squared Exponential is very widespread in use, but provides only local generalisation.
- Good features can greatly improve generalisation.
- A feature mapping can be incorporated into a the kernel (like the periodic kernel).
- Finding good features/kernels is hard
- Learning deep representations may be a way of doing this.

Generalisation

Bengio: *Shallow architectures (kernel learning) generalise locally, while deep representations allow more global generalisation.*

To a certain extent, this is a simplification of something we already know.

- Bengio's point: Squared Exponential is very widespread in use, but provides only local generalisation.
- Good features can greatly improve generalisation.
- A feature mapping can be incorporated into a the kernel (like the periodic kernel).
- Finding good features/kernels is hard
- Learning deep representations may be a way of doing this.

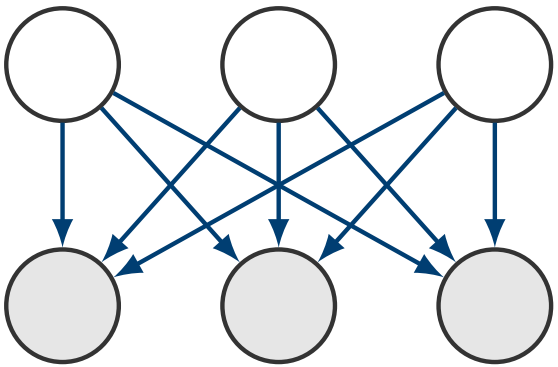
Generalisation

Bengio: *Shallow architectures (kernel learning) generalise locally, while deep representations allow more global generalisation.*

To a certain extent, this is a simplification of something we already know.

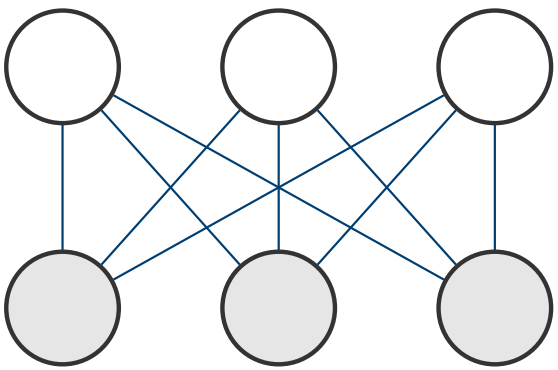
- Bengio's point: Squared Exponential is very widespread in use, but provides only local generalisation.
- Good features can greatly improve generalisation.
- A feature mapping can be incorporated into a the kernel (like the periodic kernel).
- Finding good features/kernels is hard
- Learning deep representations may be a way of doing this.

Deep Belief Networks



Belief net:

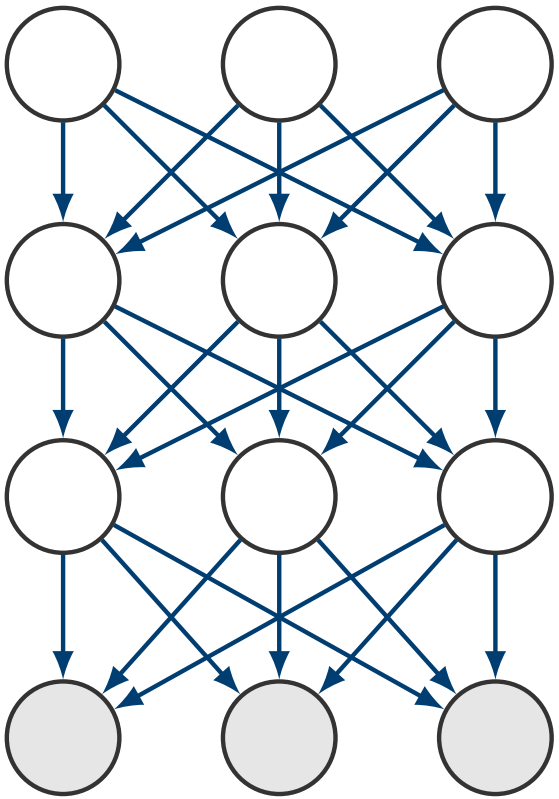
- Directed acyclic graph
- Sigmoid belief net [Neal (1992)]
  - Binary nodes
  - $p(\boldsymbol{y}, \boldsymbol{h}) = \prod_{i=1}^{D} \text{Ber}(y_i \mid \sigma(\boldsymbol{w}^T h))$ $\prod_{j=1}^{E} \text{Ber}(h_j \mid w_{2j})$



Restricted Boltzman machine:

- Undirected bipartite graph
- Binary nodes
- $p(\boldsymbol{y}, \boldsymbol{h}) = \dfrac{\exp(\boldsymbol{h}^T W \boldsymbol{y} + \boldsymbol{a}^T \boldsymbol{y} + \boldsymbol{b}^T \boldsymbol{h})}{Z}$

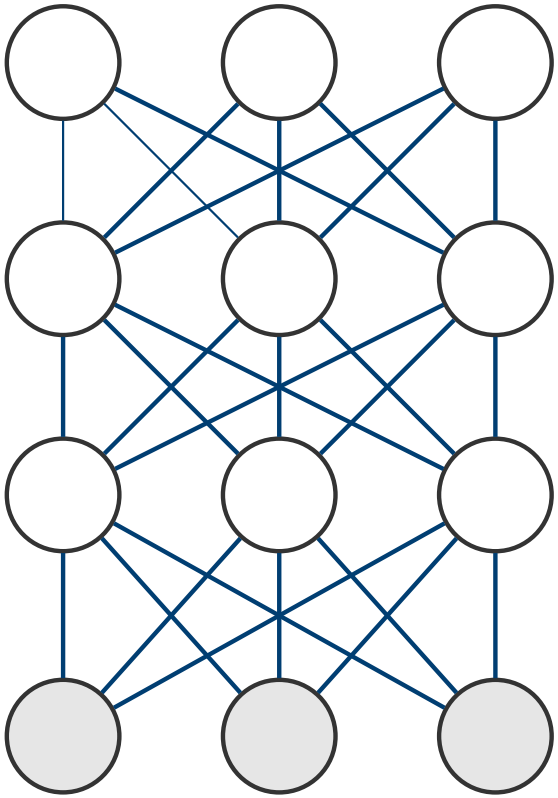
Deep Belief Networks



Deep directed net: Training very difficult due to "explaining away" effect

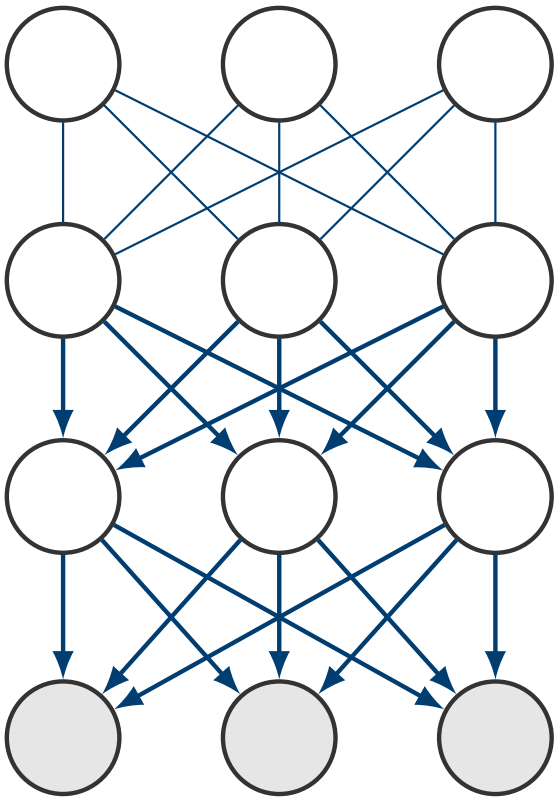
- Posterior on hidden nodes does not factorise
- MCMC - slow
- Variational - too approximate

Deep Belief Networks



Deep Boltzmann machine:
Sulakhutinov & Hinton, AISTATS (2009)

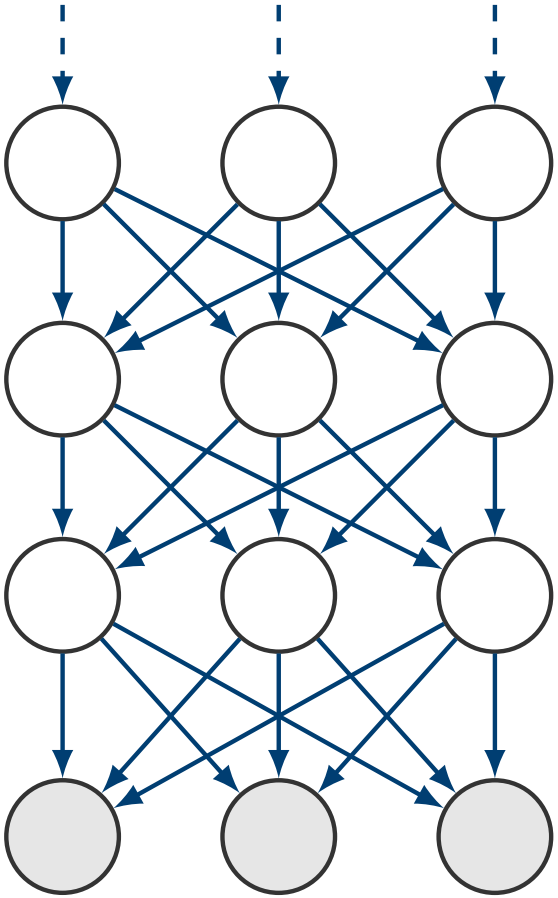
Deep Belief Networks



Deep Belief net:

- Hinton, Osindero & Teh, Neural Computation (2006)
- RBM acts as a "complementary prior" to remove "explaining away"

Deep Belief Networks



W

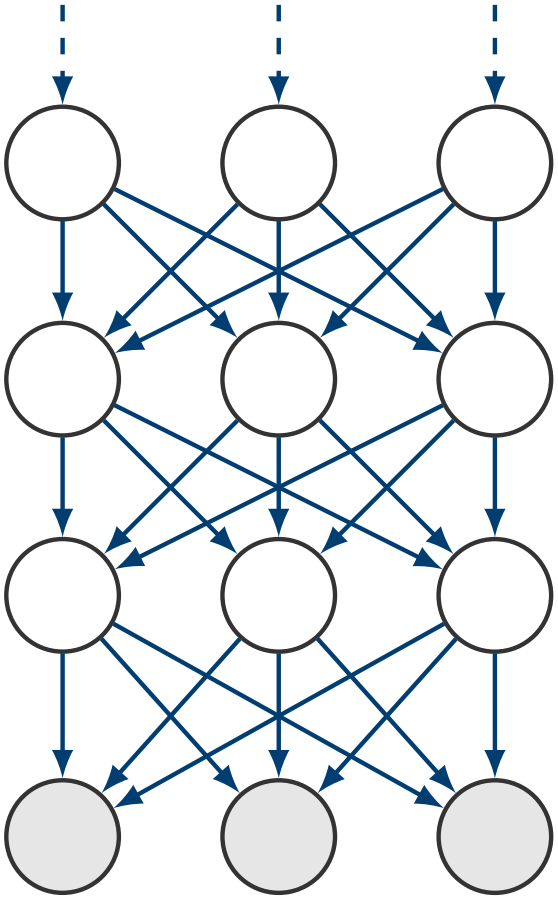
W^T

W

=

Posterior on hidden nodes factorises

Deep Belief Networks

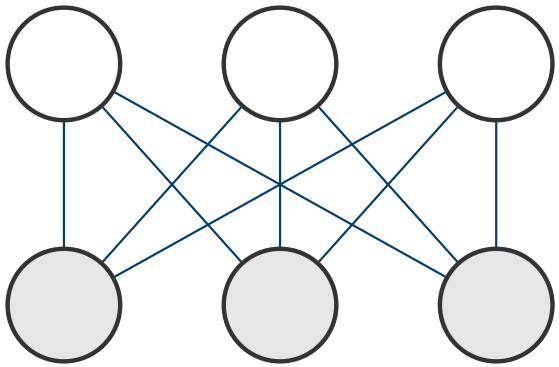


W

W^T

W

$=$



Posterior on hidden nodes factorises

Deep Belief Networks

UNIVERSITY OF CAMBRIDGE

Training:

1. Iteratively train each layer in a greedy manner
2. Fine tuning with and up-down algorithm on subset of data
3. Cross validation step
4. Further training

Deep Gaussian Processes

UNIVERSITY OF CAMBRIDGE

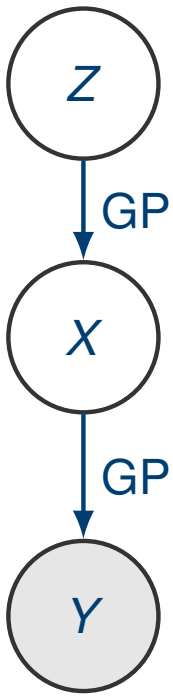
- Gaussian Process Latent Variable Model
  [Lawrence (2004)]
- Bayesian Gaussian Process Latent Variable Models
  [Titsias and Lawrence (2010)]
- Deep Gaussian Processes
  [Damianou and Lawrence (2013)]

Deep Gaussian Processes

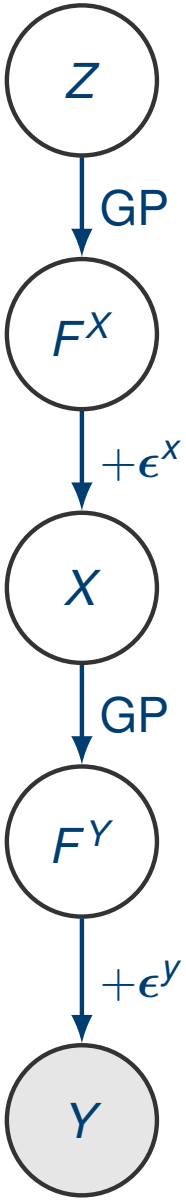
UNIVERSITY OF CAMBRIDGE

Deep Gaussian Processes

UNIVERSITY OF CAMBRIDGE



Deep Gaussian Processes



$$F^x \sim \mathcal{GP}(\mathbf{0}, k^x(Z, Z)) \quad (1)$$

$$\boldsymbol{x}_n = \boldsymbol{f}_n^x + \epsilon_n^x \quad (2)$$

$$F^y \sim \mathcal{GP}(\mathbf{0}, k^x(X, X)) \quad (3)$$

$$\boldsymbol{y}_n = \boldsymbol{f}_n^y + \epsilon_n^y \quad (4)$$

$$\log p(Y) = \log \int p(Y \mid F^Y) p(F^Y \mid X) p(X \mid F^X) p(F^X \mid Z) p(Z) \tag{5}$$

$$\mathcal{F} = \int Q \log \frac{p(Y \mid F^Y) p(F^Y \mid X) p(X \mid F^X) p(F^X \mid Z) p(Z)}{Q} \tag{6}$$

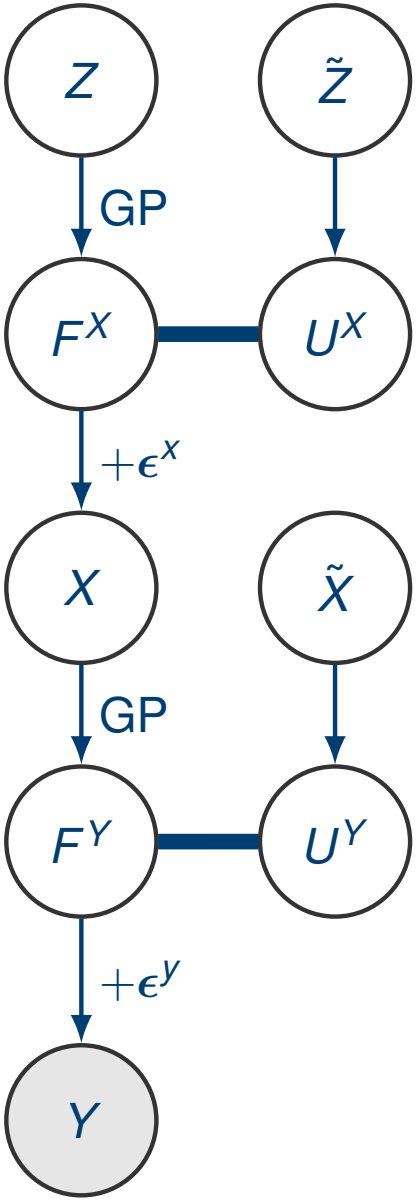
$$\mathcal{F} \leq \log p(Y) \tag{7}$$

$$\log p(Y) = \log \int p(Y \mid F^Y) p(F^Y \mid X) p(X \mid F^X) p(F^X \mid Z) p(Z) \tag{5}$$

$$\mathcal{F} = \int Q \log \frac{p(Y \mid F^Y) p(F^Y \mid X) p(X \mid F^X) p(F^X \mid Z) p(Z)}{Q} \tag{6}$$

$$\mathcal{F} \leq \log p(Y) \tag{7}$$

Deep Gaussian Processes



$$p(F^X \mid Z) = p(F^X \mid U^X, Z, \tilde{Z})\, p(U^X \mid \tilde{Z}) \quad (8)$$

$$p(F^Y \mid X) = p(F^Y \mid U^Y, X, \tilde{X})\, p(U^Y \mid \tilde{X}) \quad (9)$$

UNIVERSITY OF CAMBRIDGE

$$Q = p(F^Y \mid U^Y, X)q(U^Y)q(X)p(F^Z \mid U^Z, Z)q(U^Z)q(Z) \tag{10}$$

$$\mathcal{F} = \int Q \log \frac{p(Y \mid F^Y)p(F^Y \mid X)p(X \mid F^X)p(F^X \mid Z)p(Z)}{p(F^Y \mid U^Y, X)q(U^Y)q(X)p(F^Z \mid U^Z, Z)q(U^Z)q(Z)} \tag{11}$$

$$Q = p(F^Y \mid U^Y, X)q(U^Y)q(X)p(F^Z \mid U^Z, Z)q(U^Z)q(Z) \tag{10}$$

$$\mathcal{F} = \int Q \log \frac{p(Y \mid F^Y)p(F^Y \mid X)p(X \mid F^X)p(F^X \mid Z)p(Z)}{p(F^Y \mid U^Y, X)q(U^Y)q(X)p(F^Z \mid U^Z, Z)q(U^Z)q(Z)}$$

(11)

$$Q = p(F^Y \mid U^Y, X)q(U^Y)q(X)p(F^Z \mid U^Z, Z)q(U^Z)q(Z) \tag{10}$$

$$\mathcal{F} = \int Q \log \frac{p(Y \mid F^Y)p(F^Y \mid X)p(X \mid F^X)p(F^X \mid Z)p(Z)}{p(F^Y \mid U^Y, X)q(U^Y)q(X)p(F^Z \mid U^Z, Z)q(U^Z)q(Z)}$$

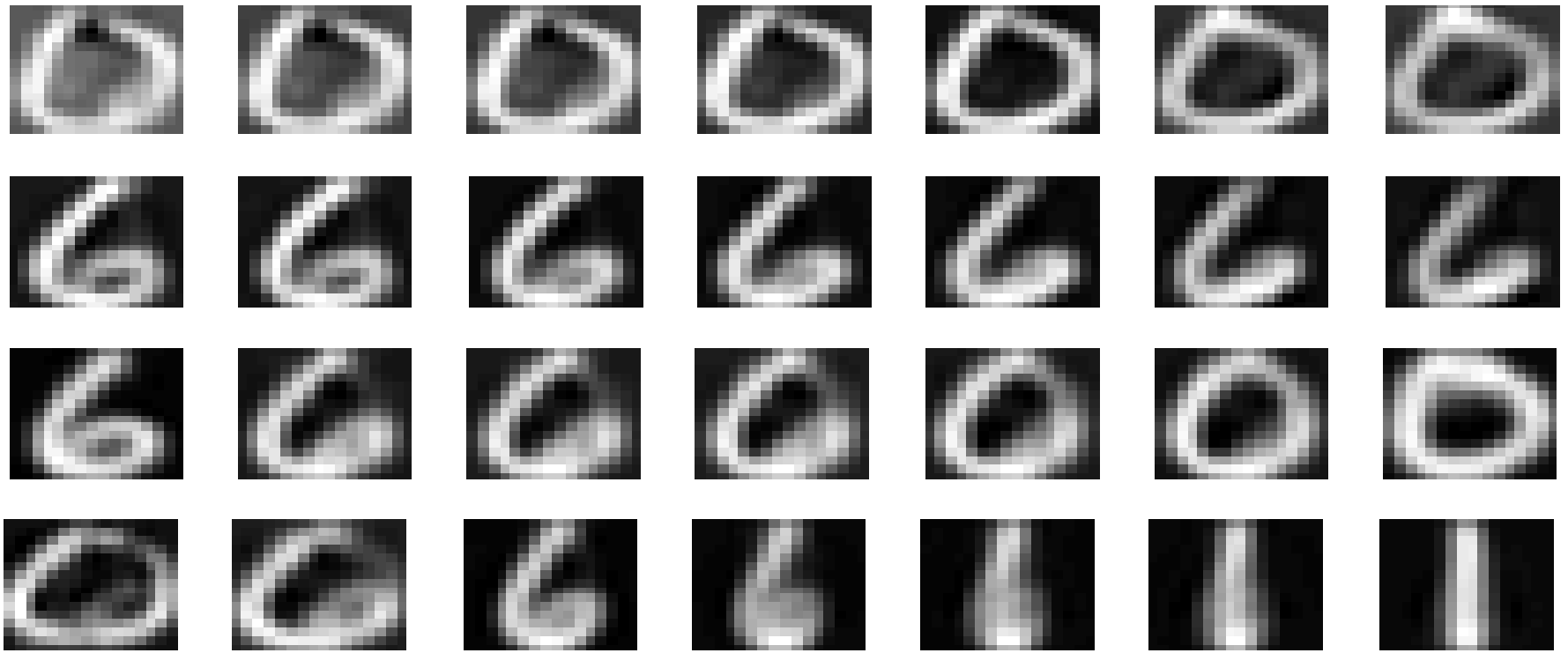
$$= \int Q \log \frac{p(Y \mid F^Y)p(U^Y)p(X \mid F^X)p(U^X)p(Z)}{q(U^Y)q(X)q(U^Z)q(Z)} \tag{11}$$

$$Q = p(F^Y \mid U^Y, X)q(U^Y)q(X)p(F^Z \mid U^Z, Z)q(U^Z)q(Z) \tag{10}$$

$$\begin{aligned}
\mathcal{F} &= \int Q \log \frac{p(Y \mid F^Y)p(F^Y \mid X)p(X \mid F^X)p(F^X \mid Z)p(Z)}{p(F^Y \mid U^Y, X)q(U^Y)q(X)p(F^Z \mid U^Z, Z)q(U^Z)q(Z)} \\
&= \int Q \log \frac{p(Y \mid F^Y)p(U^Y)p(X \mid F^X)p(U^X)p(Z)}{q(U^Y)q(X)q(U^Z)q(Z)} \\
&= g_Y + r_X + H(q(X)) - KL(q(Z) \parallel p(Z))
\end{aligned} \tag{11}$$

Deep Gaussian Processes

UNIVERSITY OF CAMBRIDGE



- USPS digits $\{0, 1, 6\}$
- 5 layers
- 150 training data points!!!

Parameters

- M D-dimensional pseudo points per layer
- $D + 2$ hyperparameters per layer
- $D + 1$ variational parameters per layer

Complexity

- $\mathcal{O}(NM^2L)$ training
- $\mathcal{O}(M^2L)$ generative (with precomputations)

Deep Density Model (DDM)

UNIVERSITY OF CAMBRIDGE

Introduced by Rippel and Adams [2013].

Main idea:

- Find an *invertible* mapping $\mathbf{y} = \mathbf{f}(\mathbf{x})$ such that $p(\mathbf{x})$ has independent dimensions with *given marginals*: $p(\mathbf{x}) = \prod_{k=1}^{K} p_{X_k}(x_k)$
- i.e. transform the data such that the distribution in the "representation space" is known.

Consequences:

- This makes the density over $\mathbf{y}$ easy to calculate: $p(\mathbf{y}) = \prod_{k=1}^{K} p_{X_k}(\mathbf{f}^{-1}(\mathbf{y})) \left| \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{f}^{-1}(\mathbf{y})}$
- Normalised and fast to compute $p(\mathbf{y}^*)$!

Deep Density Model (DDM)

UNIVERSITY OF CAMBRIDGE

Introduced by Rippel and Adams [2013].

Main idea:

- Find an *invertible* mapping $\mathbf{y} = \mathbf{f}(\mathbf{x})$ such that $p(\mathbf{x})$ has independent dimensions with *given marginals*: $p(\mathbf{x}) = \prod_{k=1}^{K} p_{X_k}(x_k)$
- i.e. transform the data such that the distribution in the "representation space" is known.

Consequences:

- This makes the density over $\mathbf{y}$ easy to calculate: $p(\mathbf{y}) = \prod_{k=1}^{K} p_{X_k}(\mathbf{f}^{-1}(\mathbf{y})) \left| \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{f}^{-1}(\mathbf{y})}$
- Normalised and fast to compute $p(\mathbf{y}^*)$!

DDM: Parameterisation

The decoder $\mathbf{f}(\mathbf{x})$ and its inverse $\mathbf{g}(\mathbf{y}) \approx \mathbf{f}^{-1}(\mathbf{y}) = \mathbf{x}$ are both parameterised by deep neural networks:

$$\mathbf{f}_{\Theta}(\mathbf{x}) = \sigma(\Omega_M \cdot + \omega_M) \circ \sigma(\Omega_{M-1} \cdot + \omega_{M-1}) \circ \cdots \circ \sigma(\Omega_1 \mathbf{x} + \omega_1)$$
$$\mathbf{g}_{\Psi}(\mathbf{y}) = \sigma(\Gamma_M \cdot + \gamma_M) \circ \sigma(\Gamma_{M-1} \cdot + \gamma_{M-1}) \circ \cdots \circ \sigma(\Gamma_1 \mathbf{y} + \gamma_1)$$

We find $\mathbf{g} \approx \mathbf{f}^{-1}$ separately because this has regularisation benefits (discussed shortly).

DDM: Objective Function

UNIVERSITY OF CAMBRIDGE

The cost function:
$C(\boldsymbol{\Theta}, \boldsymbol{\Psi}) = \mu_{\mathcal{D}}\mathcal{D}(\boldsymbol{\Psi}) + \mu_{\mathcal{I}}\mathcal{I}(\Theta) + \mu_{\mathcal{R}}\mathcal{R}(\boldsymbol{\Theta}, \boldsymbol{\Psi})$

- $\mathcal{D}(\boldsymbol{\Psi})$: Divergence penalty
- $\mathcal{I}(\boldsymbol{\Theta})$: Invertability penalty
- $\mathcal{R}(\boldsymbol{\Theta}, \boldsymbol{\Psi})$: Reconstruction loss

DDM: Divergence penalty

Forces the model to distribute mass of the data in the representation space similarly to the desired distribution $p(\mathbf{x})$.

$$\mathcal{D}(\boldsymbol{\Psi}) = \left(\frac{1}{K}\sum_{k=1}^{K} T(\hat{p}_{X_k}(\cdot)||p_{X_k}(\cdot))\right) + \left(\frac{1}{N}\sum_{n=1}^{N} T(\hat{p}_X(\mathbf{x}_n)||p_X(\mathbf{x}_n))\right)$$

$$\hat{p}_{X_k}(x) = \frac{1}{N}\sum_{n=1}^{N} \delta(x - [\mathbf{x}_n]_k)$$

DDM: Invertability penalty

UNIVERSITY OF CAMBRIDGE

$$\mathcal{I}(\boldsymbol{\Theta}) = \frac{1}{M}\sum_{m=1}^{M} \log\left(\frac{\lambda_{max}(\boldsymbol{\Omega}_m)}{\lambda_{min}(\boldsymbol{\Omega}_m)}\right)$$

Ensures invertibility, so $p(\mathbf{y})$ is well defined.

DDM: Reconstruction loss

UNIVERSITY OF CAMBRIDGE

- Ensures that $\mathbf{g}_{\boldsymbol{\psi}}(\mathbf{y}) \approx \mathbf{f}_{\boldsymbol{\Theta}}^{-1}(\mathbf{y})$ - Information is preserved.
- As a consequence, marginals are forced to become independent.

$$\mathcal{H}(p_{\mathbf{Y}}(\cdot)) = \sum_{k=1}^{K} \mathcal{H}(p_{X_k}(\cdot)) + \\ - D(\prod_{k=1}^{K} p_{X_k}(\cdot) || p_{\mathbf{X}}(\cdot)) + \\ + \mathbb{E}\left[\log \left| \frac{\partial f_{\boldsymbol{\Theta}}(\cdot)}{\partial \mathbf{y}} \right| \right]$$

DDM: Results

UNIVERSITY OF CAMBRIDGE

- 1.6% error rate on MNIST

(a) Generations from a 3-layer DDM on MNIST, with diversified marginals Beta $(\cdot; 0.015, 0.8)$.

(b) Generations from a 3-layer DDM on CIFAR-10, with diversified marginals Beta $(\cdot; 0.5, 3)$.

Figure 4: Generations from DDMs.

Deep Latent Gaussian Models

UNIVERSITY OF CAMBRIDGE

Claims to be:

- Deep - for capturing higher moments of the data
- Non-linear - Allowing for complex structure in the data
- Fast - in terms of sampling fantasy data
- Scalable - in terms of the size of training set

Also provides a nice link to autoencoders.

UNIVERSITY OF CAMBRIDGE

Generative process

$$\xi_l \sim \mathcal{N}(\xi_l|0, I) \tag{12}$$
$$\mathbf{h}_L = G_L \xi_L \tag{13}$$
$$\mathbf{h}_l = T_l(\mathbf{h}_{l+1}) + G_L \xi_l \tag{14}$$
$$\mathbf{v} \sim p(\mathbf{v}|T_0(\mathbf{h}_1)) \tag{15}$$

- Transformation T_l is a multilayer perceptron
- Parameters θ contains the $\{G_l\}$ and the MLP parameters

DLGM: Stochastic Back-Propagation

UNIVERSITY OF CAMBRIDGE

- Variationally integrate out ξ:
  $\mathcal{L}(V) = -D_{KL}(q(\xi)||p(\xi)) + \mathbb{E}_q[\log p(V|\xi, \theta^g)]$
- Need gradients w.r.t. θ^g and the variational parameters

The trick for the variational parameters:

- $\nabla_{\mu_l} \mathbb{E}_q[\log p(V|\xi, \theta^g)] = \mathbb{E}\left[\nabla_{\xi_l} \log p(\mathbf{v}|\mathbf{h}(\xi))\right]$
- Backpropagation gives the gradients given ξ
- Sampling can give an unbiased estimate of the gradient

DLGM: Stochastic Back-Propagation

UNIVERSITY OF CAMBRIDGE

- Variationally integrate out ξ:
  $\mathcal{L}(V) = -D_{KL}(q(\xi)||p(\xi)) + \mathbb{E}_q[\log p(V|\xi, \theta^g)]$
- Need gradients w.r.t. θ^g and the variational parameters

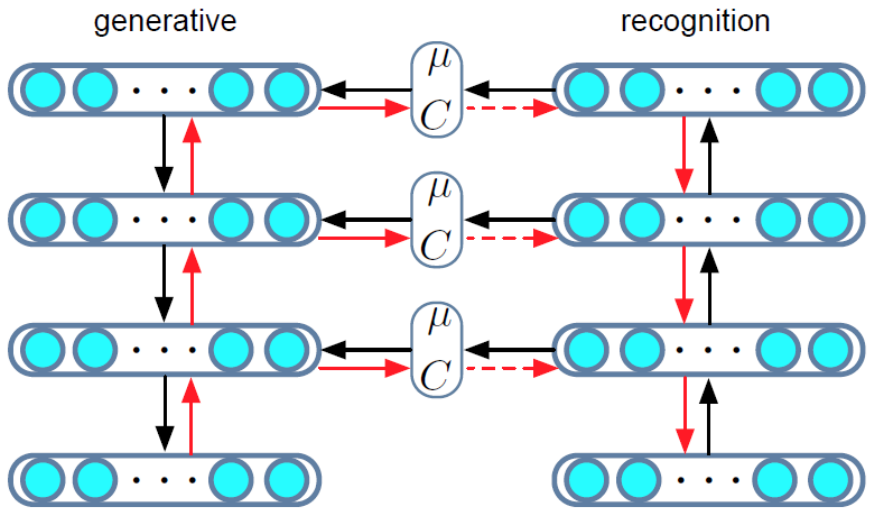
The trick for the variational parameters:

- $\nabla_{\mu_l} \mathbb{E}_q[\log p(V|\xi, \theta^g)] = \mathbb{E}\left[\nabla_{\xi_l} \log p(\mathbf{v}|\mathbf{h}(\xi))\right]$
- Backpropagation gives the gradients given ξ
- Sampling can give an unbiased estimate of the gradient

DLGM: Training

Training buzzwords: “Learning to learn Variational Stochastic Back-Propagation”

- Estimate μ_l & C_l using the recognition model
- Sample ξ for each layer so the **h**’s can be calculated
- Calculate the gradient estimates w.r.t. θ^g
- Calculate the gradient estimates w.r.t. θ^r



DLGM: Similarity to Autoencoders

UNIVERSITY OF CAMBRIDGE

Autoencoder objective function:

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{V \sim p(V), \tilde{V} \sim \mathcal{C}(\tilde{V}|V)} \lambda_n \Omega(\theta, V, \tilde{V}) - \log P_\theta(V|\tilde{V})$$

Deep Latent Gaussian Model:

$$\mathcal{L}(V) = -D_{KL}(q(\boldsymbol{\xi})||p(\boldsymbol{\xi})) + \mathbb{E}_q\left[\log p(V|\boldsymbol{\xi}, \boldsymbol{\theta}^g)\right]$$

- Autoencoders aim to maximise the probability of reconstructing a corrupted sample
- The noisy sample $\tilde{V}$ is analogous to the latent Gaussians $\boldsymbol{\xi}$
- The learned posterior $q(\boldsymbol{\xi}|\mathbf{v})$ is analogous to the encoder

Comparison

- All: Attempt to map the observed space to some latent space with an easy density
  - Some proofs about why this may be good: Gaussianisation
- Latent dimension different for all
- Inference varies wildly

What would be cool

References

Y. Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009. ISSN 1935-8237. doi: 10.1561/2200000006. URL http://dx.doi.org/10.1561/2200000006.

Johan Håstad and Mikael Goldmann. On the power of small-depth threshold circuits. *computational complexity*, 1(2):113–129, 1991. ISSN 1016-3328. doi: 10.1007/BF01272517. URL http://dx.doi.org/10.1007/BF01272517.

Oren Rippel and Ryan Prescott Adams. High-dimensional probability estimation with deep density models, 2013.