Probabilistic & Bayesian deep learning

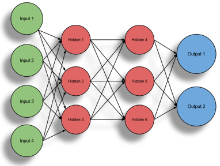
Andreas Damianou

Amazon Research Cambridge, UK

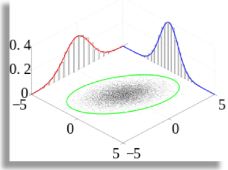
Talk at University of Sheffield, 19 March 2019

In this talk

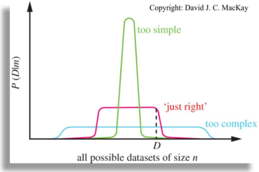
Deep learning



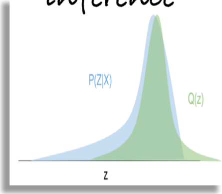
Probabilistic modeling



Bayesian reasoning



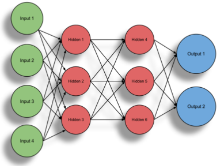
Approximate inference



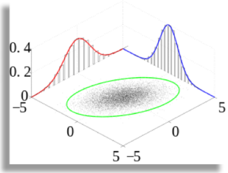
Not in this talk: CRFs, Boltzmann machines, …

In this talk

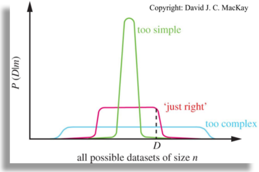
Deep learning



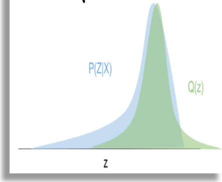
Probabilistic modeling



Bayesian reasoning

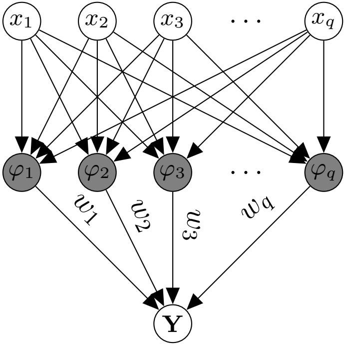


Approximate inference



Not in this talk: CRFs, Boltzmann machines, ...

A standard neural network



- Define: $\phi_j = \phi(x_j)$ and $f_j = w_j \phi_j$ (ignore bias for now)
- Once we've defined all w's with back-prop, then f (and the whole network) becomes deterministic.
- What does that imply?

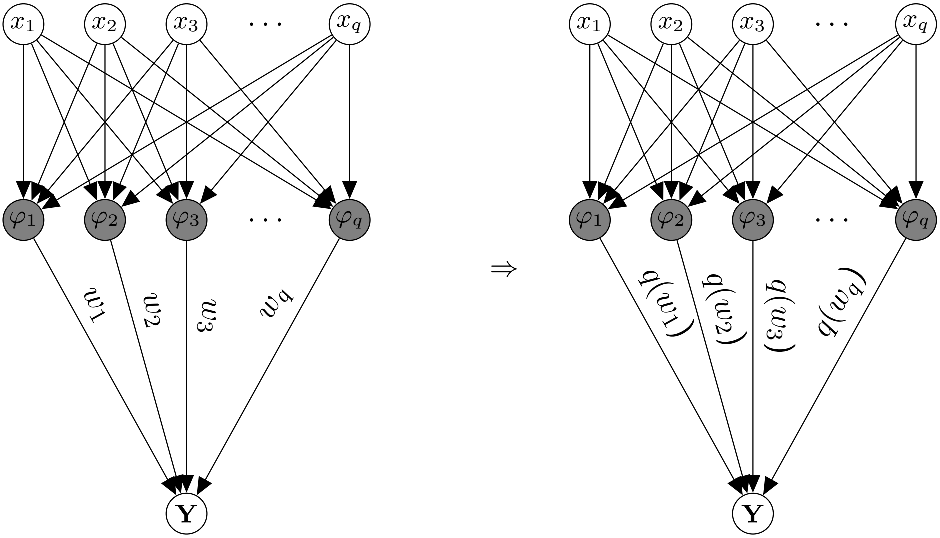
Trained neural network is deterministic. Implications?

- Generalization: Overfitting occurs. Need for ad-hoc invention of regularizers: dropout, early stopping...
- Data generation: A model which generalizes well, should also understand -or even be able to create ("imagine")- variations.
- No predictive uncertainty: Uncertainty needs to be propagated across the model to be reliable.

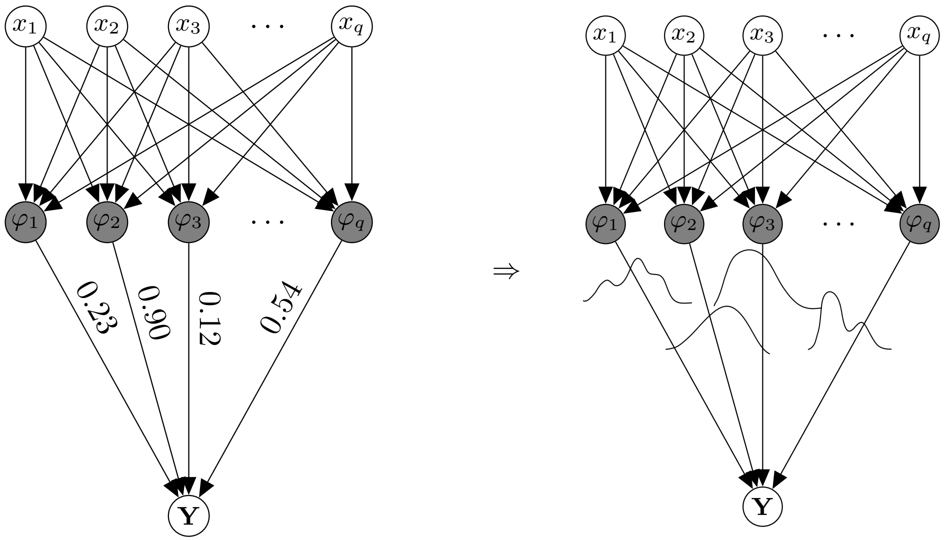
Need for uncertainty

- Reinforcement learning
- Critical predictive systems
- Active learning
- Semi-automatic systems
- Scarce data scenarios
- ...

BNN with priors on its weights



BNN with priors on its weights



Probabilistic re-formulation

- DNN: $\mathbf{y} = g(\mathbf{W}, \mathbf{x}) = \mathbf{w}_1 \varphi(\mathbf{w}_2 \varphi(\ldots \mathbf{x}))$
- Training minimizing loss:

$$\arg\min_{\mathbf{W}} \underbrace{\frac{1}{2} \sum_{i=1}^{N} (g(\mathbf{W}, x_i) - y_i)^2}_{\text{fit}} + \underbrace{\lambda \sum_i \| \mathbf{w}_i \|}_{\text{regularizer}}$$

- Equivalent probabilistic view for regression, maximizing posterior probability:

$$\arg\max_{\mathbf{W}} \underbrace{\log p(\mathbf{y}|\mathbf{x}, \mathbf{W})}_{\text{fit}} + \underbrace{\log p(\mathbf{W})}_{\text{regularizer}}$$

where $p(\mathbf{y}|\mathbf{x}, \mathbf{W}) \sim \mathcal{N}$ and $p(\mathbf{W}) \sim$ Laplace

Probabilistic re-formulation

- DNN: $\mathbf{y} = g(\mathbf{W}, \mathbf{x}) = \mathbf{w}_1\varphi(\mathbf{w}_2\varphi(\ldots \mathbf{x}))$
- Training minimizing loss:

$$\arg\min_{\mathbf{W}} \underbrace{\frac{1}{2}\sum_{i=1}^{N}(g(\mathbf{W}, x_i) - y_i)^2}_{\text{fit}} + \underbrace{\lambda \sum_i \| \mathbf{w}_i \|}_{\text{regularizer}}$$

- Equivalent probabilistic view for regression, maximizing posterior probability:

$$\arg\max_{\mathbf{W}} \underbrace{\log p(\mathbf{y}|\mathbf{x}, \mathbf{W})}_{\text{fit}} + \underbrace{\log p(\mathbf{W})}_{\text{regularizer}}$$

where $p(\mathbf{y}|\mathbf{x}, \mathbf{W}) \sim \mathcal{N}$ and $p(\mathbf{W}) \sim$ Laplace

Probabilistic re-formulation

- DNN: $\mathbf{y} = g(\mathbf{W}, \mathbf{x}) = \mathbf{w}_1 \varphi(\mathbf{w}_2 \varphi(\ldots \mathbf{x}))$
- Training minimizing loss:

$$\arg\min_{\mathbf{W}} \underbrace{\frac{1}{2} \sum_{i=1}^{N} (g(\mathbf{W}, x_i) - y_i)^2}_{\text{fit}} + \underbrace{\lambda \sum_i \| \mathbf{w}_i \|}_{\text{regularizer}}$$

- Equivalent probabilistic view for regression, maximizing posterior probability:

$$\arg\max_{\mathbf{W}} \underbrace{\log p(\mathbf{y}|\mathbf{x}, \mathbf{W})}_{\text{fit}} + \underbrace{\log p(\mathbf{W})}_{\text{regularizer}}$$

where $p(\mathbf{y}|\mathbf{x}, \mathbf{W}) \sim \mathcal{N}$ and $p(\mathbf{W}) \sim$ Laplace

Integrating out weights

- Define: $D = (x, y)$

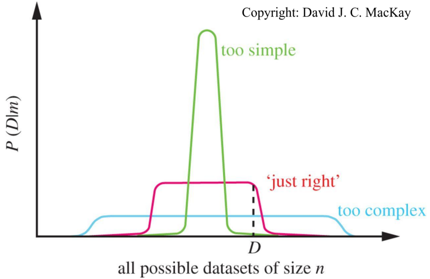
- Remember Bayes' rule:

$$p(w|D) = \frac{p(D|w)p(w)}{p(D) = \int p(D|w)p(w)\mathrm{d}w}$$

- For Bayesian inference, weights need to also be *integrated out*. This gives us a properly defined *posterior* on the parametres.

(Bayesian) Occam's Razor

"A plurality is not to be posited without necessity". *W. of Ockham* "Everything should be made as simple as possible, but not simpler". *A. Einstein*



Evidence is higher for the model that is not "unnecessarily complex" but still "explains" the data D.

Bayesian Inference

Remember: Separation of Model and Inference

Inference

- $p(D)$ (and hence $p(w|D)$) is difficult to compute because of the nonlinear way in which w appears through g.

- Attempt at *variational inference*:

$$\underbrace{\mathsf{KL}\left(q(w;\theta)\,\|\,p(w|D)\right)}_{\text{minimize}} = \log(p(D)) - \underbrace{\mathcal{L}(\theta)}_{\text{maximize}}$$

where

$$\mathcal{L}(\theta) = \underbrace{\mathbb{E}_{q(w;\theta)}[\log p(D,w)]}_{\mathcal{F}} + \mathbb{H}\left[q(w;\theta)\right]$$

- Term in red is still problematic. Solution: MC.

- Such approaches can be formulated as *black-box* inferences.

Inference

- $p(D)$ (and hence $p(w|D)$) is difficult to compute because of the nonlinear way in which w appears through g.
- Attempt at *variational inference*:

$$\underbrace{\mathsf{KL}\left(q(w;\theta) \,\|\, p(w|D)\right)}_{\text{minimize}} = \log(p(D)) - \underbrace{\mathcal{L}(\theta)}_{\text{maximize}}$$

where

$$\mathcal{L}(\theta) = \underbrace{\mathbb{E}_{q(w;\theta)}\left[\log p(D, w)\right]}_{\mathcal{F}} + \mathbb{H}\left[q(w;\theta)\right]$$

- Term in red is still problematic. Solution: MC.
- Such approaches can be formulated as *black-box* inferences.

Inference

- $p(D)$ (and hence $p(w|D)$) is difficult to compute because of the nonlinear way in which w appears through g.

- Attempt at *variational inference*:

$$\underbrace{\mathsf{KL}\left(q(w;\theta) \,\|\, p(w|D)\right)}_{\text{minimize}} = \log(p(D)) - \underbrace{\mathcal{L}(\theta)}_{\text{maximize}}$$

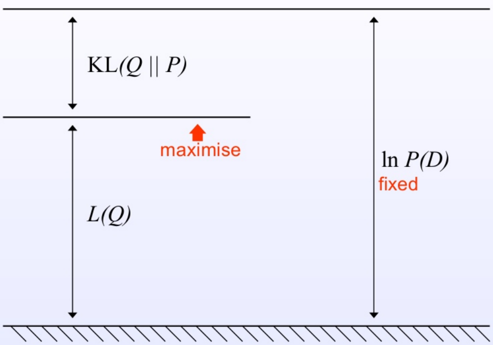
  where

$$\mathcal{L}(\theta) = \underbrace{\mathbb{E}_{q(w;\theta)}\left[\log p(D,w)\right]}_{\mathcal{F}} + \mathbb{H}\left[q(w;\theta)\right]$$

- Term in red is still problematic. Solution: MC.

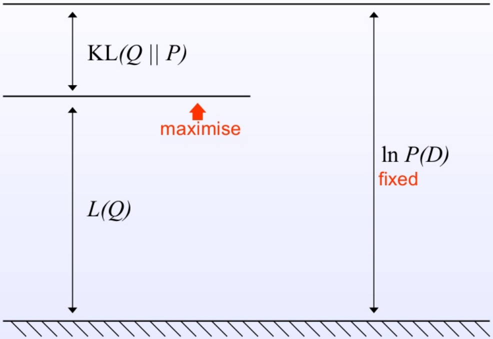
- Such approaches can be formulated as *black-box* inferences.

Minimising the KL divergence



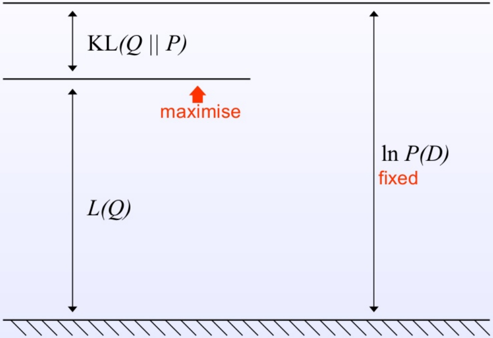
Credit: Tushar Tank

Minimising the KL divergence



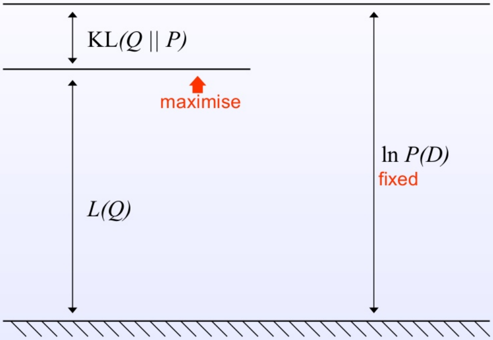
Credit: Tushar Tank

Minimising the KL divergence



Credit: Tushar Tank

Minimising the KL divergence



Credit: Tushar Tank

Inference: "Score function method"

$$
\begin{aligned}
\nabla_\theta \mathcal{F} &= \color{red}{\nabla_\theta \mathbb{E}_{q(w;\theta)}[\log p(D,w)]} \\
&= \mathbb{E}_{q(w;\theta)}\left[p(D,w)\nabla_\theta \log q(w;\theta)\right] \\
&\approx \frac{1}{K}\sum_{i=1}^{K} p(D,w^{(k)})\nabla_\theta \log q(w^{(k)};\theta), \quad w^{(k)} \overset{iid}{\sim} q(w;\theta)
\end{aligned}
$$

(Paisley et al., 2012; Ranganath et al., 2014; Mnih and Gregor, 2014, Ruiz et al. 2016)

Inference: "Reparameterization gradient"

- Reparametrize w as a transformation $\mathcal{T}$ of a simpler variable ϵ: $w = \mathcal{T}(\epsilon; \theta)$
- $q(\epsilon)$ is now independent of θ

$$\begin{aligned}\nabla_\theta \mathcal{F} &= \nabla_\theta \mathbb{E}_{q(w;\theta)}[\log p(D, w)] \\ &= \mathbb{E}_{q(\epsilon)}\left[\nabla_w p(D, w)|_{w=\mathcal{T}(\epsilon;\theta)} \nabla_\theta \mathcal{T}(\epsilon; \theta)\right]\end{aligned}$$

- For example: $w \sim \mathcal{N}(\mu, \sigma) \xrightarrow{\mathcal{T}} w = \mu + \sigma \cdot \epsilon$, $\epsilon \sim \mathcal{N}(0, 1)$
- MC by sampling from $q(\epsilon)$ (thus obtaining samples from w through $\mathcal{T}$)

(Salimans and Knowles, 2013; Kingma and Welling, 2014, Ruiz et al. 2016)

Inference: "Reparameterization gradient"

- Reparametrize w as a transformation $\mathcal{T}$ of a simpler variable ϵ: $w = \mathcal{T}(\epsilon; \theta)$
- $q(\epsilon)$ is now independent of θ

$$\begin{aligned} \nabla_\theta \mathcal{F} &= \nabla_\theta \mathbb{E}_{q(w;\theta)}[\log p(D, w)] \\ &= \mathbb{E}_{q(\epsilon)}\left[\nabla_w p(D, w)|_{w=\mathcal{T}(\epsilon;\theta)} \nabla_\theta \mathcal{T}(\epsilon; \theta)\right] \end{aligned}$$

- For example: $w \sim \mathcal{N}(\mu, \sigma) \xrightarrow{\mathcal{T}} w = \mu + \sigma \cdot \epsilon$, $\epsilon \sim \mathcal{N}(0, 1)$
- MC by sampling from $q(\epsilon)$ (thus obtaining samples from w through $\mathcal{T}$)

(Salimans and Knowles, 2013; Kingma and Welling, 2014, Ruiz et al. 2016)

Inference: "Reparameterization gradient"

- Reparametrize w as a transformation $\mathcal{T}$ of a simpler variable ϵ: $w = \mathcal{T}(\epsilon; \theta)$
- $q(\epsilon)$ is now independent of θ

$$
\begin{aligned}
\nabla_\theta \mathcal{F} &= \nabla_\theta \mathbb{E}_{q(w;\theta)}[\log p(D, w)] \\
&= \mathbb{E}_{q(\epsilon)}\left[\nabla_w p(D, w)|_{w=\mathcal{T}(\epsilon;\theta)} \nabla_\theta \mathcal{T}(\epsilon; \theta)\right]
\end{aligned}
$$

- For example: $w \sim \mathcal{N}(\mu, \sigma) \xrightarrow{\mathcal{T}} w = \mu + \sigma \cdot \epsilon,\ \epsilon \sim \mathcal{N}(0, 1)$
- MC by sampling from $q(\epsilon)$ (thus obtaining samples from w through $\mathcal{T}$)

(Salimans and Knowles, 2013; Kingma and Welling, 2014, Ruiz et al. 2016)

Inference: "Reparameterization gradient"

- Reparametrize w as a transformation $\mathcal{T}$ of a simpler variable ϵ: $w = \mathcal{T}(\epsilon;\theta)$
- $q(\epsilon)$ is now independent of θ

$$\begin{aligned}\nabla_\theta \mathcal{F} &= \nabla_\theta \mathbb{E}_{q(w;\theta)}[\log p(D,w)] \\ &= \mathbb{E}_{q(\epsilon)}\left[\nabla_w p(D,w)|_{w=\mathcal{T}(\epsilon;\theta)} \nabla_\theta \mathcal{T}(\epsilon;\theta)\right]\end{aligned}$$

- For example: $w \sim \mathcal{N}(\mu,\sigma) \xrightarrow{\mathcal{T}} w = \mu + \sigma \cdot \epsilon,\ \epsilon \sim \mathcal{N}(0,1)$
- MC by sampling from $q(\epsilon)$ (thus obtaining samples from w through $\mathcal{T}$)

(Salimans and Knowles, 2013; Kingma and Welling, 2014, Ruiz et al. 2016)

Black-box VI

Black-Box Stochastic Variational Inference in Five Lines of Python

David Duvenaud
dduvenaud@seas.harvard.edu
Harvard University

Ryan P. Adams
rpa@seas.harvard.edu
Harvard University

Abstract

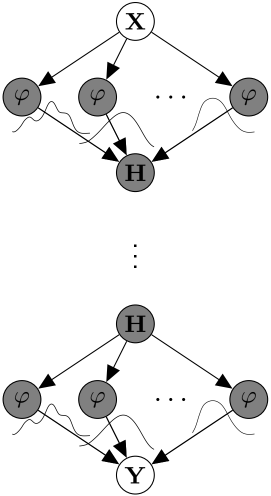
Several large software engineering projects have been undertaken to support black-box inference methods. In contrast, we emphasize how easy it is to construct scalable and easy-to-use automatic inference methods using only automatic differentiation. We present a small function which computes stochastic gradients of the evidence lower bound for any differentiable posterior. As an example, we perform stochastic variational inference in a deep Bayesian neural network.

Black-box VI (github.com/blei-lab/edward)

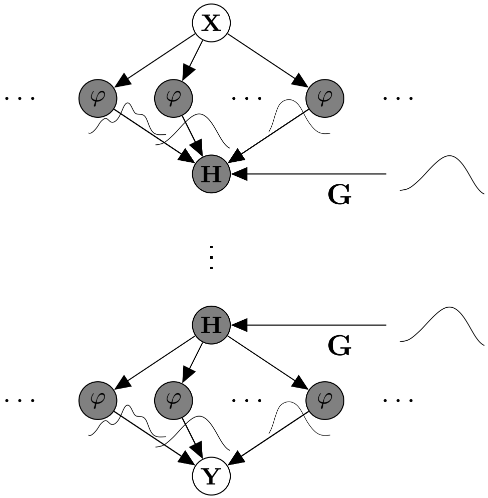
```
# MODEL
W_0 = Normal(loc=tf.zeros([D, 10]), scale=tf.ones([D, 10]))
W_1 = Normal(loc=tf.zeros([10, 10]), scale=tf.ones([10, 10]))
W_2 = Normal(loc=tf.zeros([10, 1]), scale=tf.ones([10, 1]))
b_0 = Normal(loc=tf.zeros(10), scale=tf.ones(10))
b_1 = Normal(loc=tf.zeros(10), scale=tf.ones(10))
b_2 = Normal(loc=tf.zeros(1), scale=tf.ones(1))

X = tf.placeholder(tf.float32, [N, D])
y = Normal(loc=neural_network(X), scale=0.1 * tf.ones(N))

# INFERENCE
qW_0 = Normal(loc=tf.Variable(tf.random_normal([D, 10])),
              scale=tf.nn.softplus(tf.Variable(tf.random_normal([D, 10]))))
```

```
qb_2 = Normal(loc=tf.Variable(tf.random_normal([1])),
              scale=tf.nn.softplus(tf.Variable(tf.random_normal([1]))))

inference = ed.KLqp({W_0: qW_0, b_0: qb_0,
                     W_1: qW_1, b_1: qb_1,
                     W_2: qW_2, b_2: qb_2}, data={X: X_train, y: y_train})
inference.run()
```

Priors on weights *(what we saw before)*

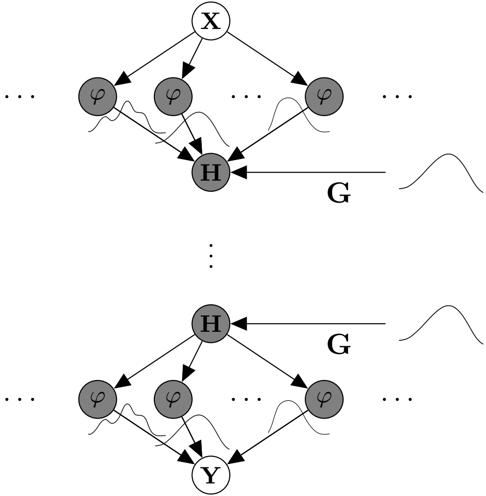


From NN to GP



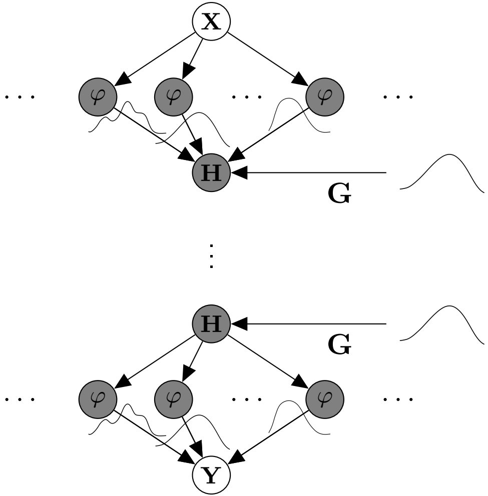
- NN: $\mathbf{H}_2 = \mathbf{W}_2 \phi(\mathbf{H}_1)$
- GP: ϕ is $\infty-$dimensional so:
  $\mathbf{H}_2 = f_2(\mathbf{H}_1; \theta_2) + \epsilon$
- NN: $p(\mathbf{W})$
- GP: $p(f(\cdot))$
- VAE can be seen as a special case of this

From NN to GP



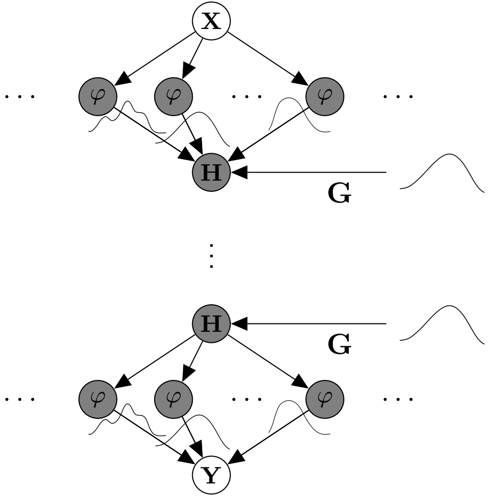
- NN: $\mathbf{H}_2 = \mathbf{W}_2\phi(\mathbf{H}_1)$
- GP: ϕ is ∞−dimensional so:
  $\mathbf{H}_2 = f_2(\mathbf{H}_1; \theta_2) + \epsilon$
- NN: $p(\mathbf{W})$
- GP: $p(f(\cdot))$
- VAE can be seen as a special case of this

From NN to GP



- NN: $\mathbf{H}_2 = \mathbf{W}_2 \phi(\mathbf{H}_1)$
- GP: ϕ is ∞−dimensional so:
  $$\mathbf{H}_2 = f_2(\mathbf{H}_1; \theta_2) + \epsilon$$
- NN: $p(\mathbf{W})$
- GP: $p(f(\cdot))$
- VAE can be seen as a special case of this

From NN to GP



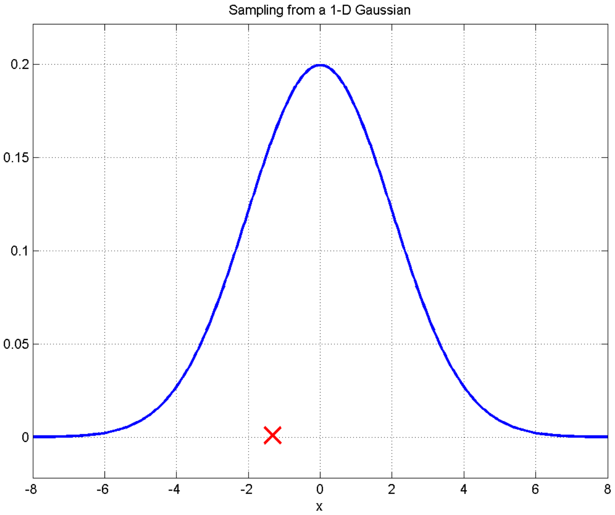
- Real world perfectly described by unobserved *latent* variables: $\hat{\mathbf{H}}$
- But we only observe noisy high-dimensional data: $\mathbf{Y}$
- We try to interpret the world and infer the latents: $\mathbf{H} \approx \hat{\mathbf{H}}$
- Inference:
  $p(\mathbf{H}|\mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{H})p(\mathbf{H})}{p(\mathbf{Y})}$

Deep Gaussian processes

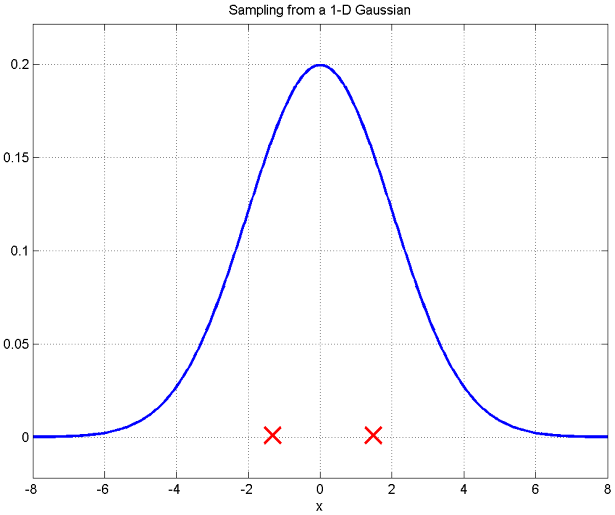
- Uncertainty about parameters: Check. Uncertainty about structure?
- Deep GP simultaneously brings in:
  - prior on "weights"
  - input/latent space is kernalized
  - stochasticity in the warping

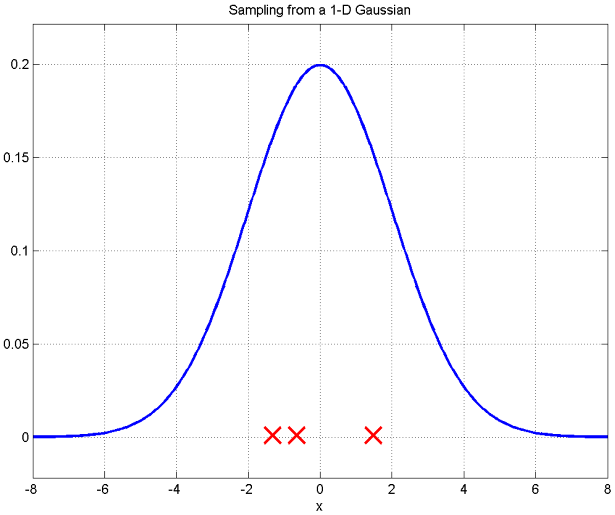
Introducing Gaussian Processes:

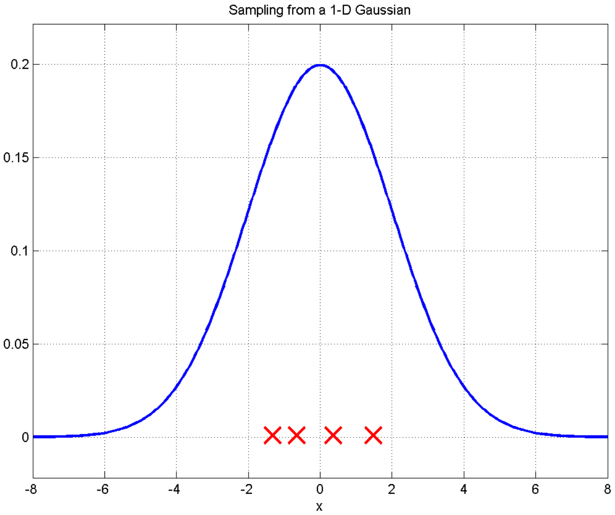
- A Gaussian distribution depends on a mean and a covariance matrix.
- A Gaussian process depends on a mean and a covariance function.

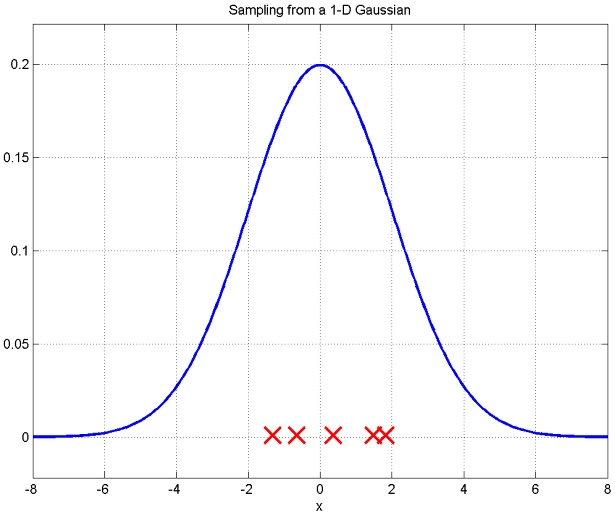


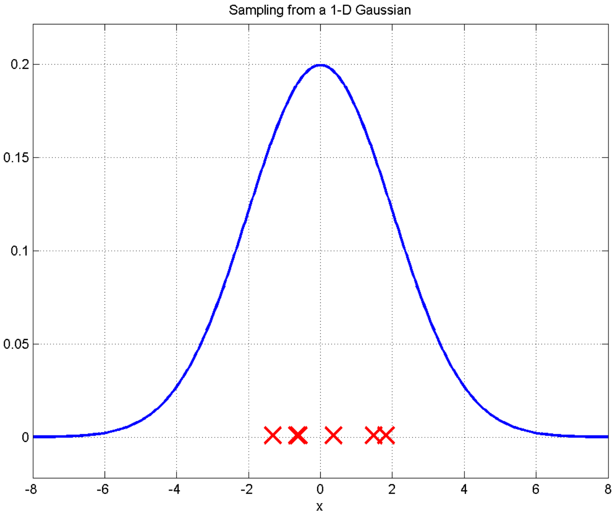
Sampling from a 1-D Gaussian
0.2
0.15
0.1
0.05
0
-8
-6
-4
-2
0
2
4
6
8
x


Sampling from a 1-D Gaussian
0.2
0.15
0.1
0.05
0
-8
-6
-4
-2
0
2
4
6
8
x

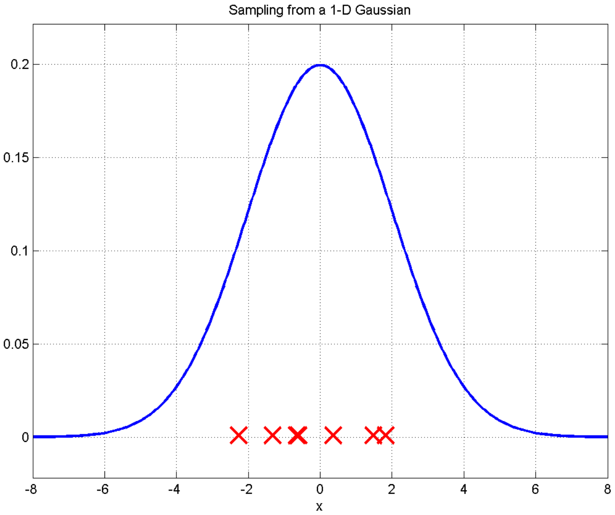

Sampling from a 1-D Gaussian
0.2
0.15
0.1
0.05
0
-8
-6
-4
-2
0
2
4
6
8
x

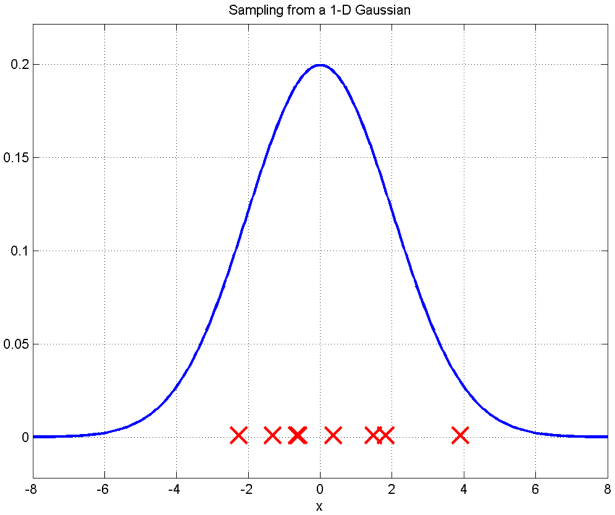

Sampling from a 1-D Gaussian
0.2
0.15
0.1
0.05
0
-8
-6
-4
-2
0
2
4
6
8
x

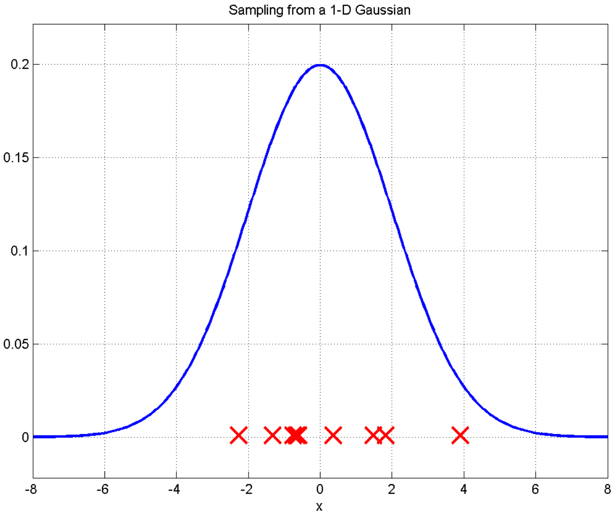

Sampling from a 1-D Gaussian
0.2
0.15
0.1
0.05
0
-8
-6
-4
-2
0
2
4
6
8
x

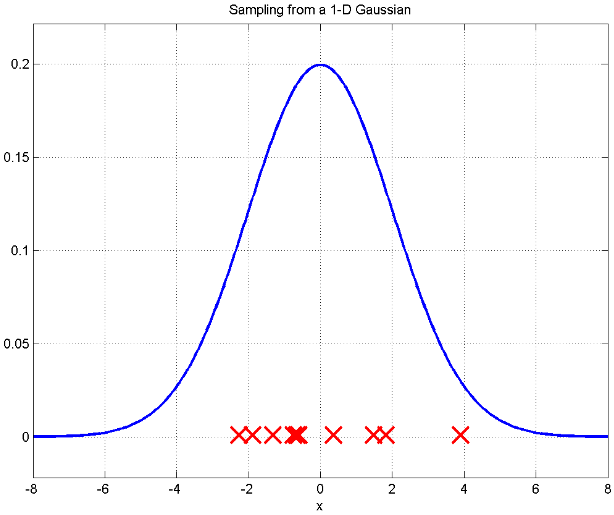


Sampling from a 1-D Gaussian
0.2
0.15
0.1
0.05
0
-8
-6
-4
-2
0
2
4
6
8
x


Sampling from a 1-D Gaussian
0.2
0.15
0.1
0.05
0
-8
-6
-4
-2
0
2
4
6
8
x

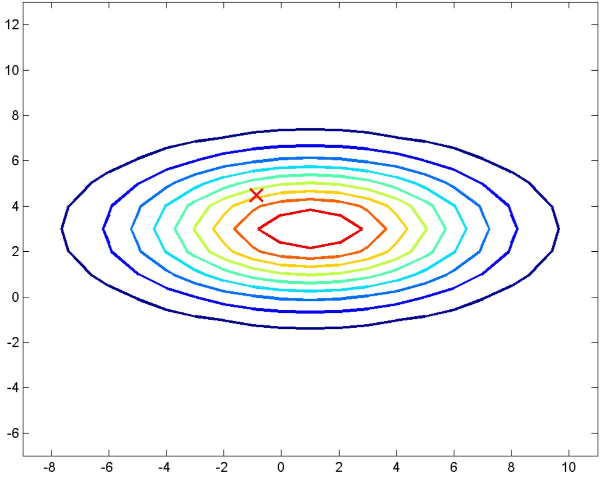

Sampling from a 1-D Gaussian
0.2
0.15
0.1
0.05
0
-8
-6
-4
-2
0
2
4
6
8
x

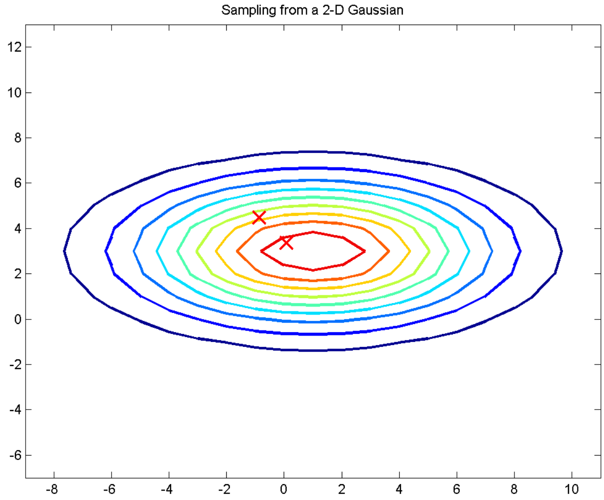

Sampling from a 1-D Gaussian
0.2
0.15
0.1
0.05
0
-8
-6
-4
-2
0
2
4
6
8
x



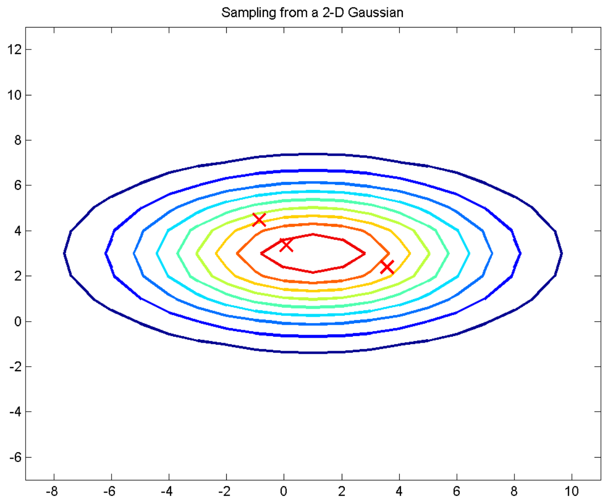
Sampling from a 1-D Gaussian
0.2
0.15
0.1
0.05
0
-8
-6
-4
-2
0
2
4
6
8
x

Sampling from a 2-D Gaussian

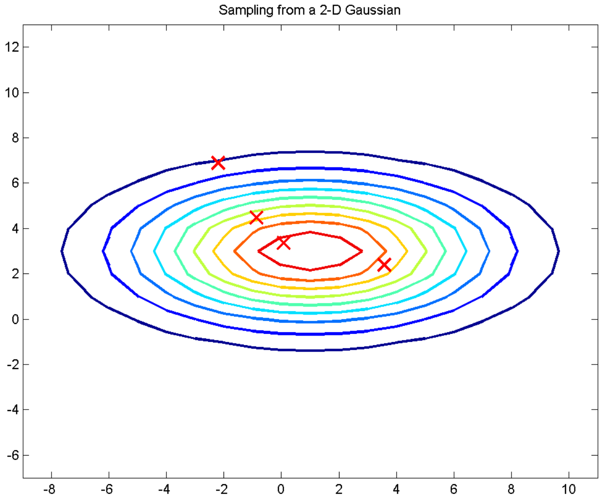




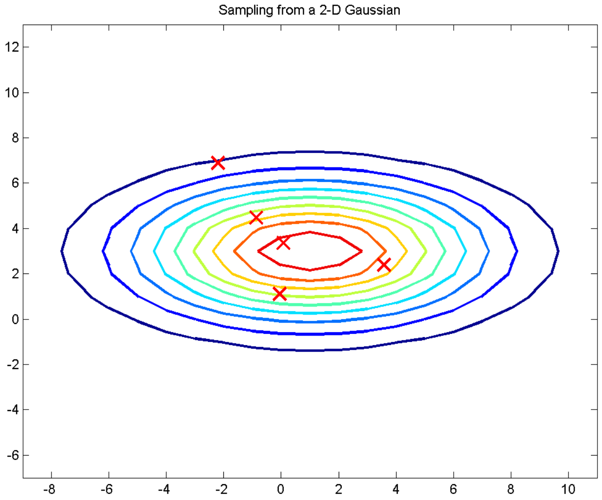
Sampling from a 2-D Gaussian
12
10
8
6
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6
8
10

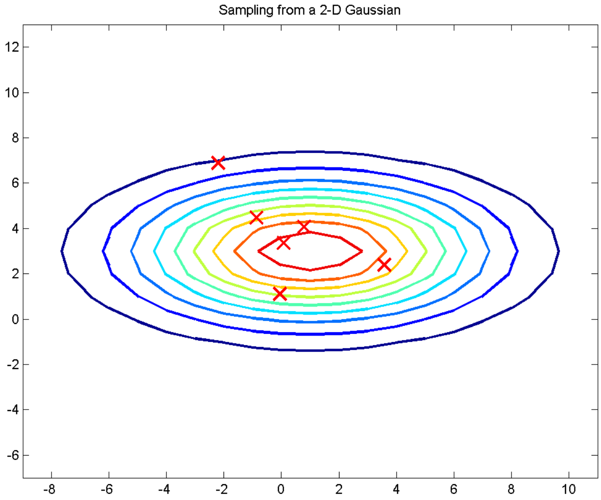


Sampling from a 2-D Gaussian
12
10
8
6
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6
8
10



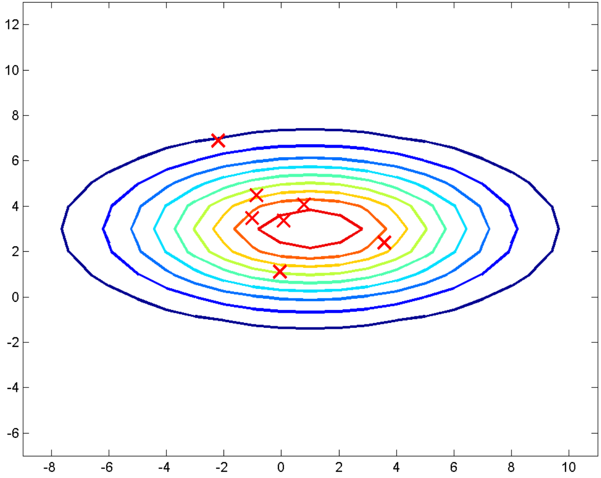
Sampling from a 2-D Gaussian
12
10
8
6
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6
8
10


Sampling from a 2-D Gaussian
12
10
8
6
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6
8
10

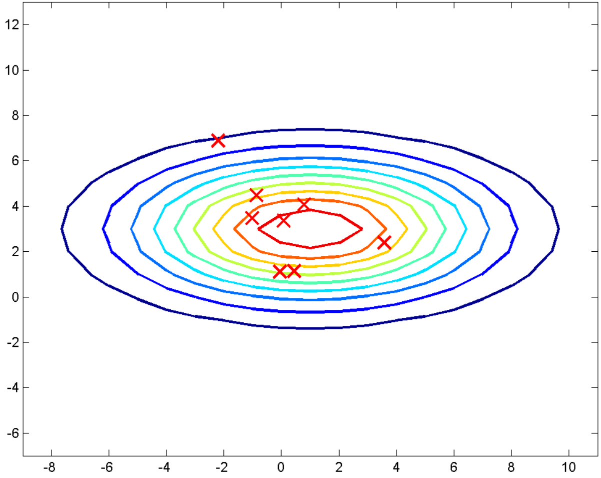


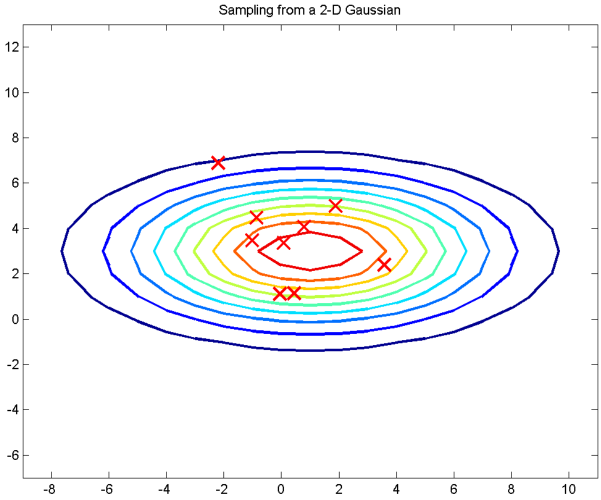
Sampling from a 2-D Gaussian
12
10
8
6
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6
8
10

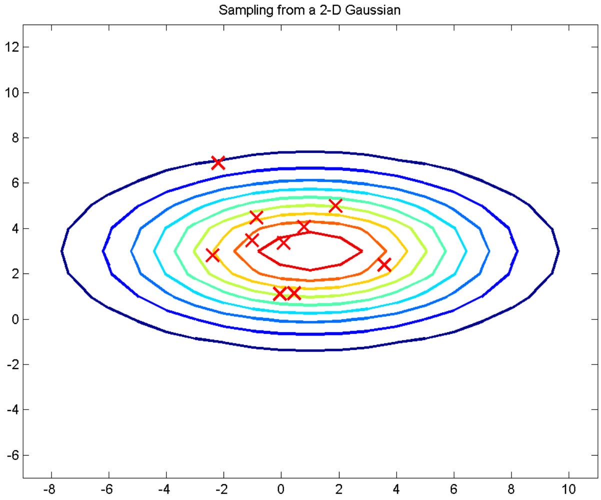
Sampling from a 2-D Gaussian



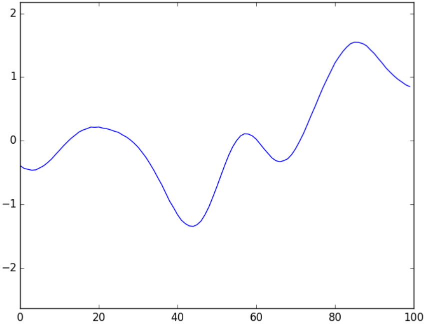
Sampling from a 2-D Gaussian



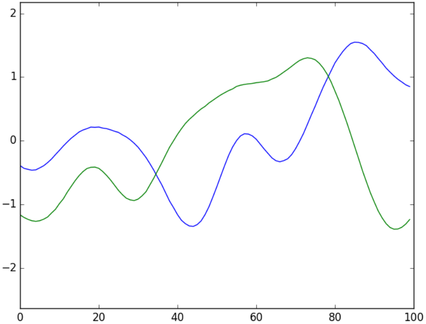

Sampling from a 2-D Gaussian
12
10
8
6
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6
8
10


Sampling from a 2-D Gaussian
12
10
8
6
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6
8
10

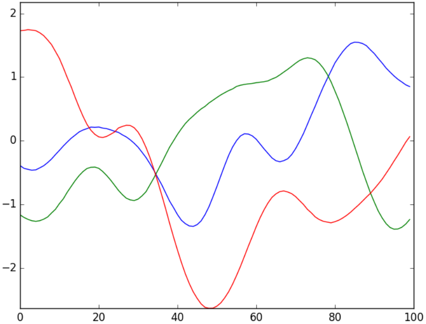
Samples from a 1-D Gaussian



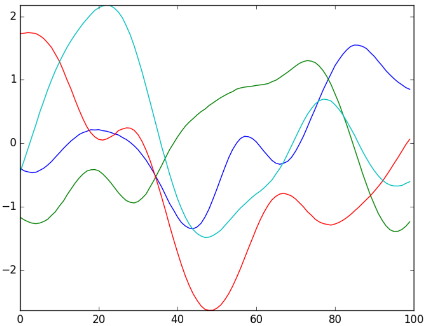
Samples from a 1-D Gaussian



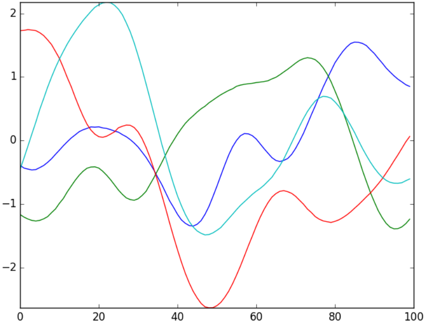
Samples from a 1-D Gaussian

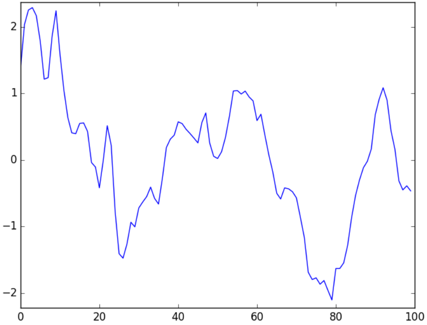


Samples from a 1-D Gaussian

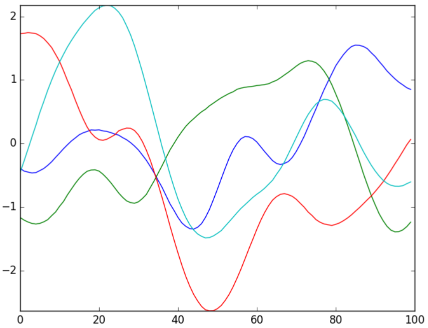


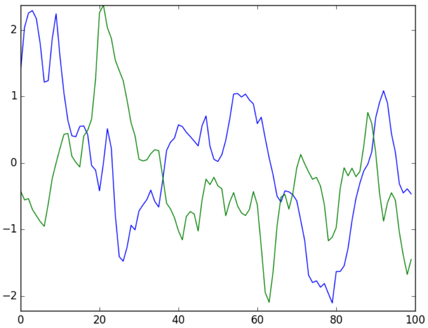
Samples from a 1-D Gaussian



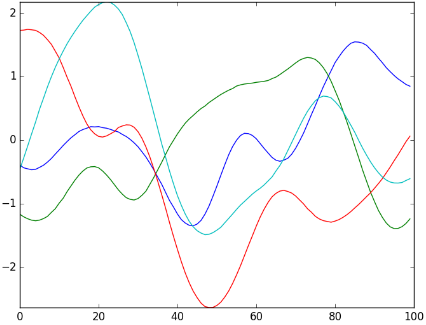


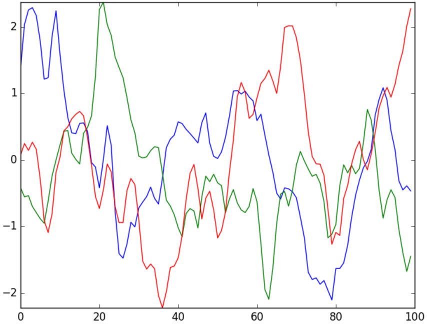
Samples from a 1-D Gaussian



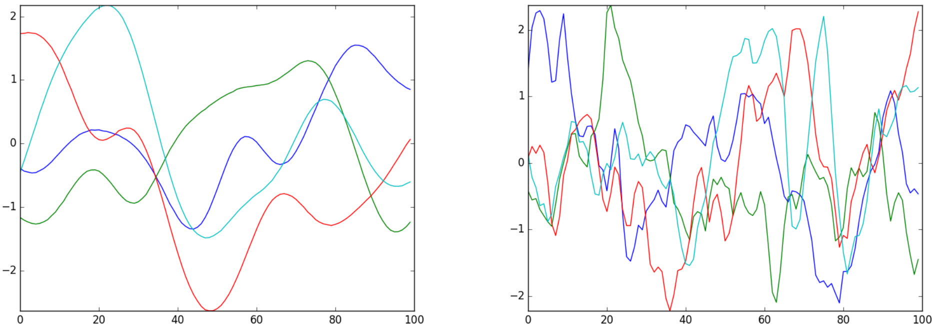


Samples from a 1-D Gaussian





Samples from a 1-D Gaussian



How do GPs solve the overfitting problem (i.e. regularize)?

- Answer: Integrate over the function itself!
- So, we will average out all possible function forms, under a (GP) prior!

Recap:

$$\text{MAP:} \quad \arg\max_{\mathbf{w}} p(\mathbf{y}|\mathbf{w}, \phi(\mathbf{x}))p(\mathbf{w}) \quad \text{e.g. } \mathbf{y} = \phi(\mathbf{x})^\top \mathbf{w} + \boldsymbol{\epsilon}$$

$$\text{Bayesian:} \quad \arg\max_{\boldsymbol{\theta}} \int_{\mathbf{f}} p(\mathbf{y}|\mathbf{f}) \underbrace{p(\mathbf{f}|\mathbf{x}, \boldsymbol{\theta})}_{\text{GP prior}} \quad \text{e.g. } \mathbf{y} = f(\mathbf{x}, \boldsymbol{\theta}) + \boldsymbol{\epsilon}$$

- $\boldsymbol{\theta}$ are *hyper*parameters
- The Bayesian approach (GP) automatically balances the data-fitting with the complexity penalty.

How do GPs solve the overfitting problem (i.e. regularize)?

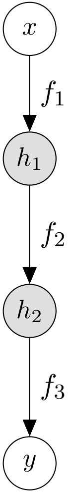
- Answer: Integrate over the function itself!
- So, we will average out all possible function forms, under a (GP) prior!

Recap:

$$\begin{aligned}
\text{MAP:} \quad & \arg\max_{\mathbf{w}} \, p(\mathbf{y}|\mathbf{w}, \phi(\mathbf{x}))p(\mathbf{w}) && \text{e.g. } \mathbf{y} = \phi(\mathbf{x})^\top \mathbf{w} + \boldsymbol{\epsilon} \\
\text{Bayesian:} \quad & \arg\max_{\boldsymbol{\theta}} \, \int_{\mathbf{f}} p(\mathbf{y}|\mathbf{f}) \underbrace{p(\mathbf{f}|\mathbf{x}, \boldsymbol{\theta})}_{\text{GP prior}} && \text{e.g. } \mathbf{y} = f(\mathbf{x}, \boldsymbol{\theta}) + \boldsymbol{\epsilon}
\end{aligned}$$

- $\boldsymbol{\theta}$ are *hyper*parameters
- The Bayesian approach (GP) automatically balances the data-fitting with the complexity penalty.

Deep Gaussian processes



- Define a recursive stacked construction

$$f(\mathbf{x}) \rightarrow \text{GP}$$

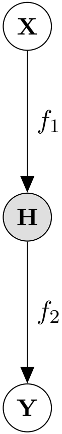
$$f_L(f_{L-1}(f_{L-2} \cdots f_1(\mathbf{x})))) \rightarrow \text{deep GP}$$

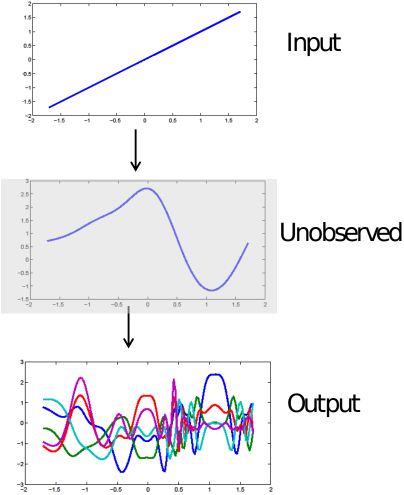
Compare to:

$$\varphi(\mathbf{x})^\top \mathbf{w} \rightarrow \text{NN}$$

$$\varphi(\varphi(\varphi(\mathbf{x})^\top \mathbf{w}_1)^\top \ldots \mathbf{w}_{L-1})^\top \mathbf{w}_L \rightarrow \text{DNN}$$

Two-layered DGP





Inference in DGPs

- It's tricky. An additional difficulty comes from the fact that the kernel couples all points, so $p(y|x)$ is not factorized anymore.

- Indeed, $p(y|x, w) = \int_f p(y|f)p(f|x, w)$ but w appears inside the non-linear kernel function: $p(f|x, w) = N(f|0, k(X, X|w))$

The resulting objective function

By integrating over all unknowns:

- We obtain approximate posteriors $q(h), q(f)$ over them

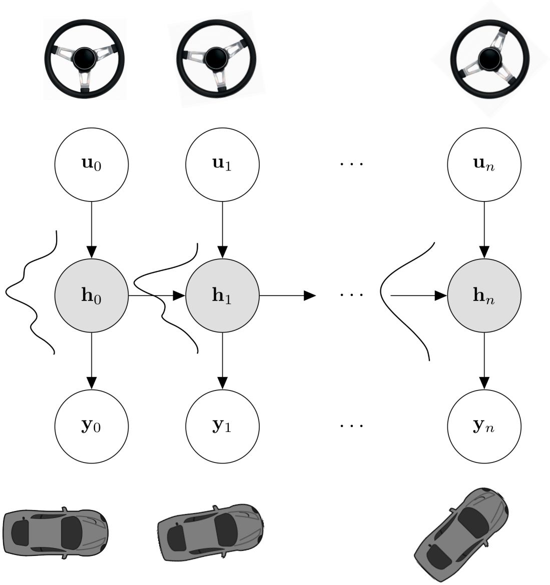
- We end up with a well regularized variational lower bound on the true marginal likelihood:

$$\mathcal{F} = \text{Data Fit} \qquad \underbrace{-\mathsf{KL}\left(q(\mathbf{h}_1) \,\|\, p(\mathbf{h}_1)\right)}_{\text{Regularisation}} + \sum_{l=2}^{L} \underbrace{\mathcal{H}\left(q(\mathbf{h}_l)\right)}_{\text{Regularisation}}$$

- Regularization achieved through uncertainty propagation.

An example where uncertainty propagation matters: Recurrent learning.

Dynamics/memory: Deep Recurrent Gaussian Process



Avatar control

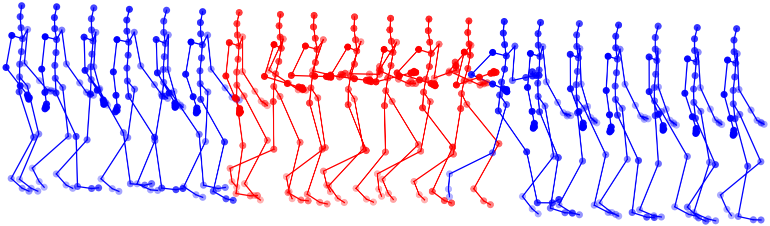


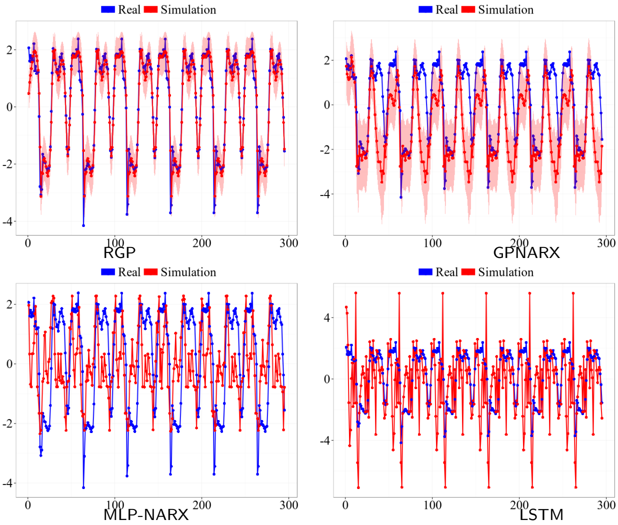
Figure: The generated motion with a step function signal, starting with walking (blue), switching to running (red) and switching back to walking (blue).

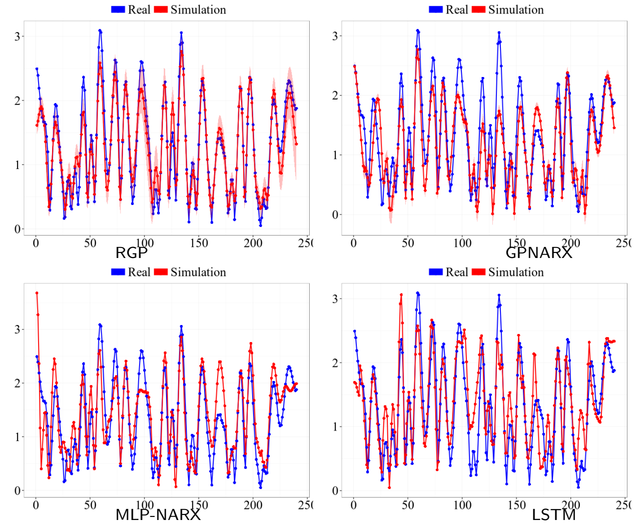
Videos:

https://youtu.be/FR-oeGxV6yY *Switching between learned speeds*

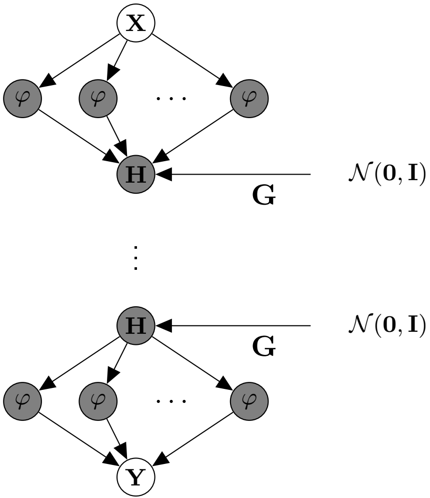
https://youtu.be/AT0HMtoPgjc *Interpolating (un)seen speed*

https://youtu.be/FuF-uZ83VMw Constant unseen speed


Real Simulation
2
0
-2
-4
0
100
200
300
RGP
Real Simulation
2
0
-2
-4
0
100
200
300
GPNARX
Real Simulation
2
0
-2
-4
0
100
200
300
MLP-NARX
Real Simulation
4
0
-4
0
100
200
300
LSTM


Real
Simulation
3
2
1
0
0
50
100
150
200
250
RGP
Real
Simulation
3
2
1
0
0
50
100
150
200
250
GPNARX
Real
Simulation
3
2
1
0
0
50
100
150
200
250
MLP-NARX
Real
Simulation
3
2
1
0
0
50
100
150
200
250
LSTM

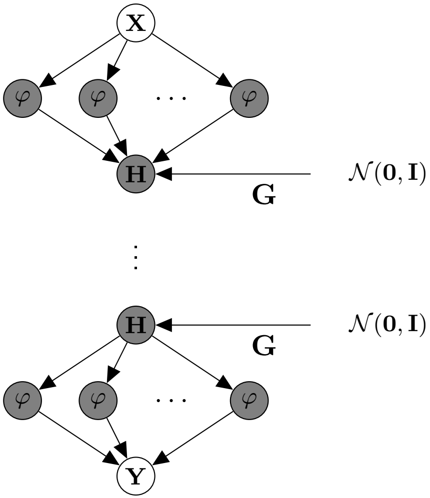
Stochastic warping



Inference:

- Need to infer posteriors on $\mathbf{H}$
- Define $q(\mathbf{H}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- Amortize: $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\} = MLP(\mathbf{Y}; \boldsymbol{\theta})$

Stochastic warping



Inference:

- Need to infer posteriors on $\mathbf{H}$
- Define $q(\mathbf{H}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- Amortize:
  $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\} = MLP(\mathbf{Y}; \boldsymbol{\theta})$

Amortized inference

Solution: Reparameterization through recognition model g:

$$
\begin{aligned}
\boldsymbol{\mu}_1^n &= g_1(\mathbf{y}^{(n)}) \\
\boldsymbol{\mu}_l^{(n)} &= g_l(\boldsymbol{\mu}_{l-1}^{(n)}) \\
g_l &= \mathsf{MLP}(\boldsymbol{\theta}_l)
\end{aligned}
$$

g_l deterministic $\Rightarrow \boldsymbol{\mu}_l^{(n)} = g_l(\ldots g_1(\mathbf{y}^{(n)}))$

Dai, Damianou, Gonzalez, Lawrence, ICLR 2016

Bayesian approach recap

- Three ways of introducing uncertainty / noise in a NN:
  - Treat weights w as *distributions* (BNNs)
  - Stochasticity in the warping function ϕ (VAE)
  - Bayesian non-parametrics applied to DNNs (DGP)

Connection between Bayesian and "traditional" approaches

There's another view of Bayesian NNs. Namely that they can be used as a means of explaining / motivating DNN techniques (e.g. Dropout) in a more mathematically grounded way.

- Dropout as variational inference. (Kingma et al. 2015), (Gal and Ghahramani. 2016)
- Low-rank NN weight approximation and Deep Gaussian processes (Hensman and Lawrence 2014, Damianou 2015, Louizos and Welling 2016, Cutajar et al. 2016)
- ...

Summary

- Motivation for probabilistic and Bayesian reasoning
- Three ways of incorporating uncertainty in DNNs
- Inference is more challenging when uncertainty has to be propagated
- Connection between Bayesian and “traditional” NN approaches