Probability & uncertainty in deep learning

Andreas Damianou

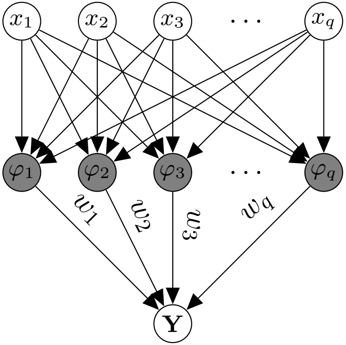
damianou@amazon.com

Amazon.com, Cambridge, UK

Deep learning summit, 21 September 2017

amazon.com

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
- ...

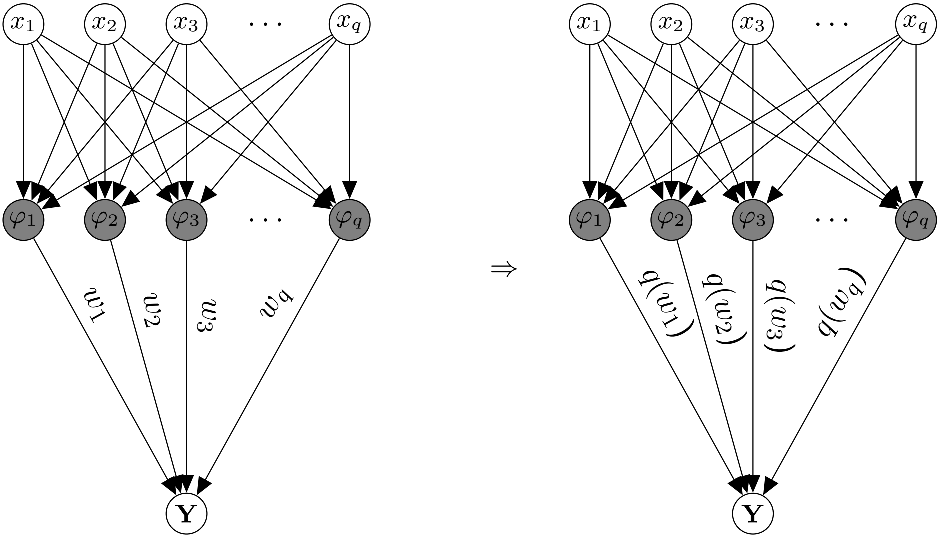
Bayesian approach

- Three ways of introducing uncertainty / noise in a NN:
  - Treat weights w as *distributions*
  - Stochasticity in the warping function ϕ
  - Bayesian non-parametrics applied to DNNs can achieve both of the above simultaneously, e.g. a Deep Gaussian process
- Result: Bayesian Neural Network (BNN)

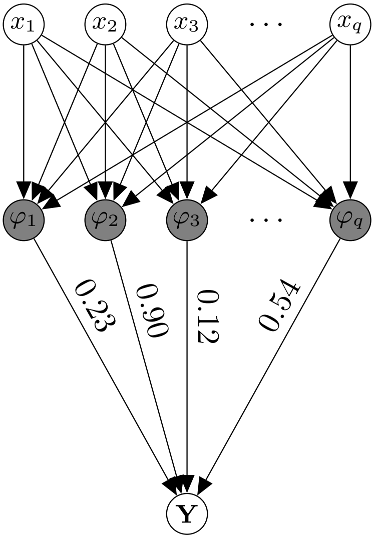
Bayesian approach

- Three ways of introducing uncertainty / noise in a NN:
  - Treat weights w as *distributions*
  - Stochasticity in the warping function ϕ
  - Bayesian non-parametrics applied to DNNs can achieve both of the above simultaneously, e.g. a Deep Gaussian process
- Result: Bayesian Neural Network (BNN)

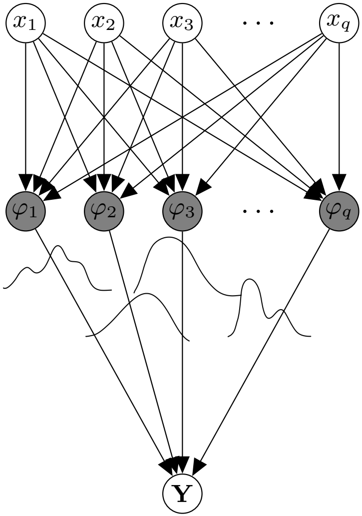
BNN with priors on its weights



BNN with priors on its weights



$\Rightarrow$



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:

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

- 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

- Optimization still done with back-prop (i.e. gradient descent).

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:

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

- 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

- Optimization still done with back-prop (i.e. gradient descent).

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:

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

- 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

- Optimization still done with back-prop (i.e. gradient descent).

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 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*:

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

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.

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)

```python
# 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]))))
```

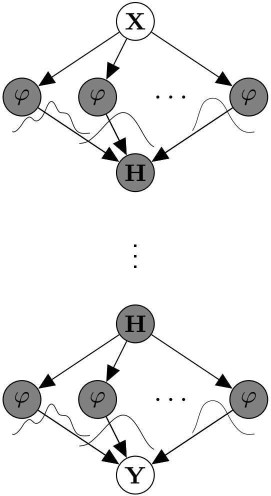
```python
qb_2 = Normal(loc=tf.Variable(tf.random_normal([1])),
              scale=tf.nn.softplus(tf.Variable(tf.random_normal([1]))))

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()
```

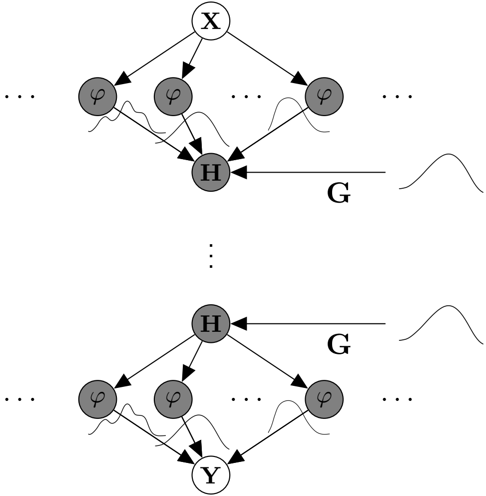
Deep Gaussian processes

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

Priors on weights *(what we saw before)*

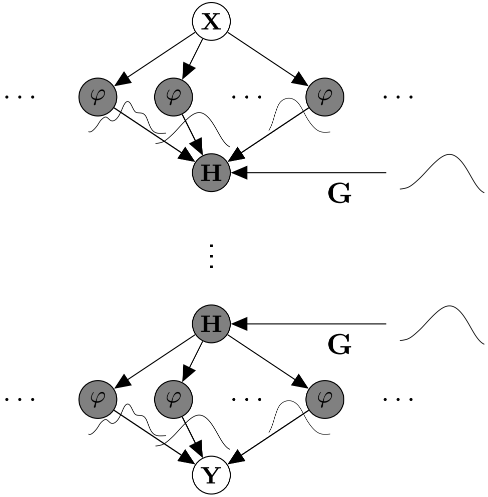


From NN to GP



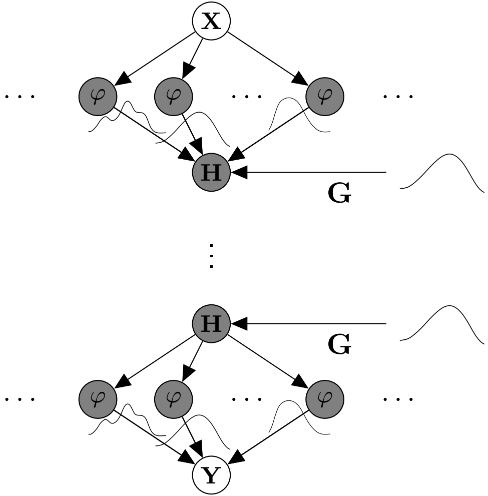
- NN: $\mathbf{H}_2 = \mathbf{W}_2 \phi(\mathbf{H}_1)$
- GP: ϕ is $\infty-$dimensional so:
  $$\mathbf{H}_2 = f_2(\mathbf{H}_1; \theta_2) + \epsilon$$
- NN: $p(\mathbf{W})$
- GP: $p(f(\cdot))$
- VAE can be seen as a special case of this

From NN to GP



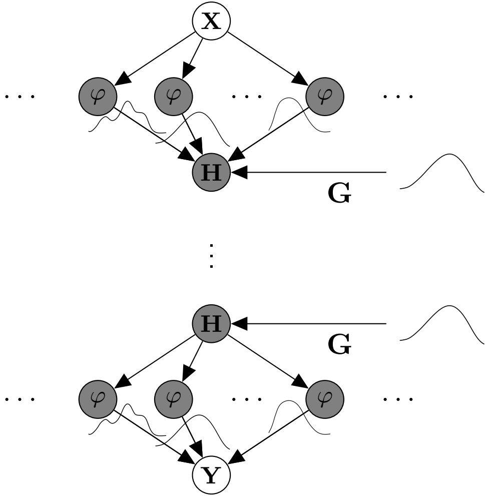
- NN: $\mathbf{H}_2 = \mathbf{W}_2 \phi(\mathbf{H}_1)$
- GP: ϕ is $\infty-$dimensional so:
  $$\mathbf{H}_2 = f_2(\mathbf{H}_1; \theta_2) + \epsilon$$
- NN: $p(\mathbf{W})$
- GP: $p(f(\cdot))$
- VAE can be seen as a special case of this

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



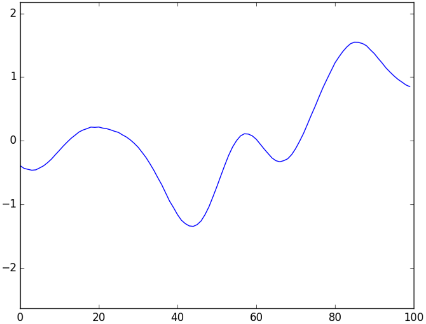
- Real world perfectly described by unobserved *latent* variables: $\hat{\mathbf{H}}$
- But we only observe noisy high-dimensional data: $\mathbf{Y}$
- We try to interpret the world and infer the latents: $\mathbf{H} \approx \hat{\mathbf{H}}$
- Inference:
  $p(\mathbf{H}|\mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{H})p(\mathbf{H})}{p(\mathbf{Y})}$

Face generation

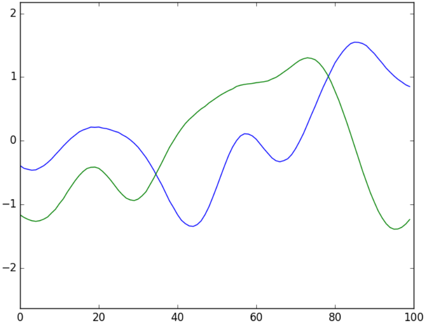
▸ https://youtu.be/rIPX3CIOhKY *Face Generation*

What does “prior over functions” mean?

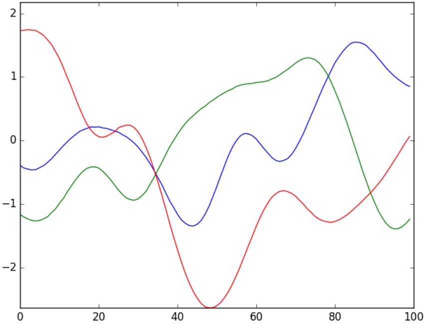
Samples from a 1-D Gaussian



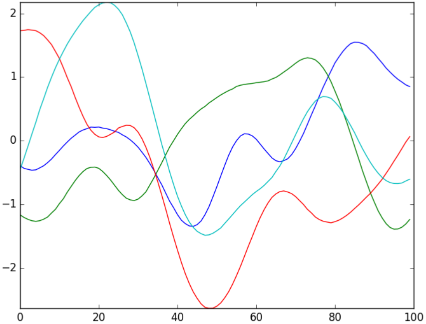
Samples from a 1-D Gaussian



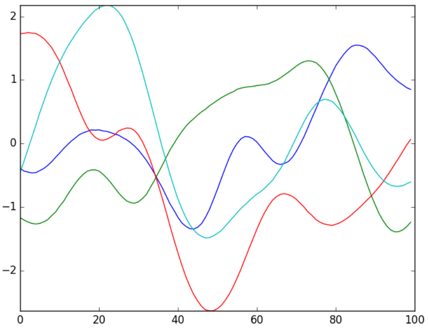
Samples from a 1-D Gaussian

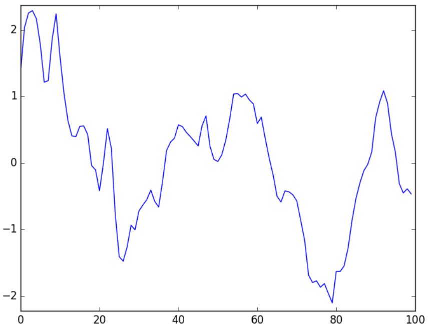


Samples from a 1-D Gaussian

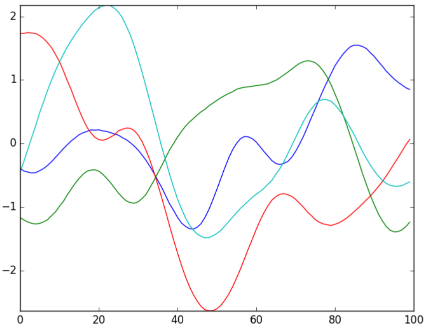


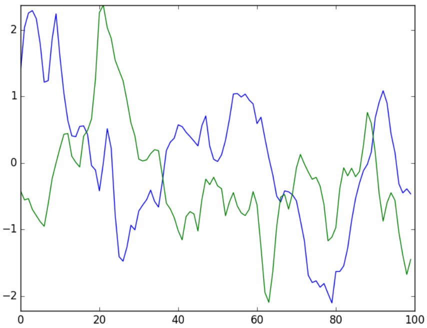
Samples from a 1-D Gaussian



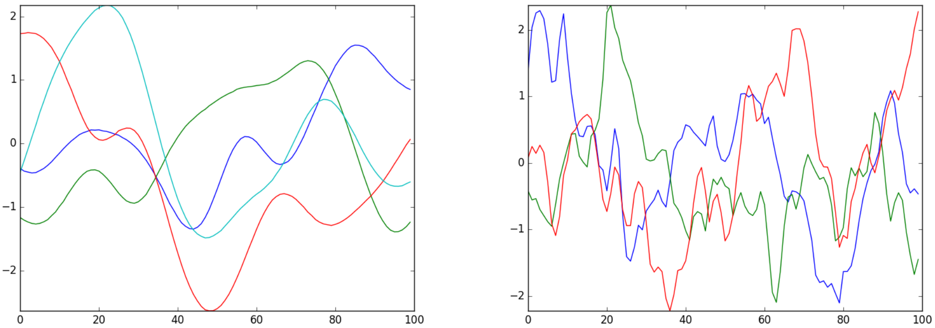


Samples from a 1-D Gaussian

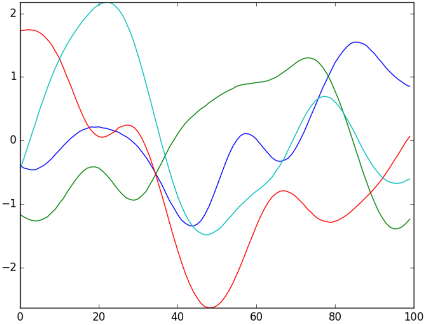


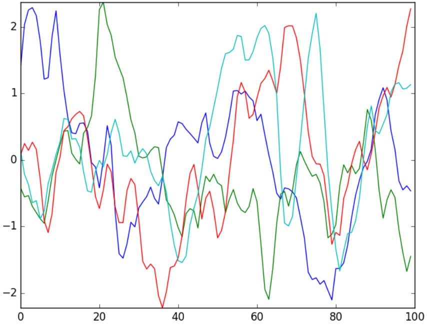


Samples from a 1-D Gaussian

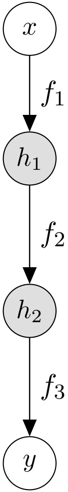


Samples from a 1-D Gaussian





Deep Gaussian processes



- Define a recursive stacked construction

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

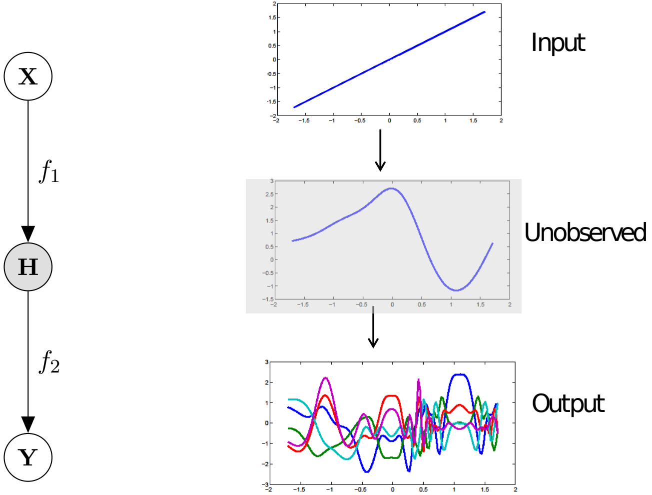
$$f_L(f_{L-1}(f_{L-2}\cdots f_1(\mathbf{x})))) \rightarrow \text{deep GP}$$

Compare to:

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

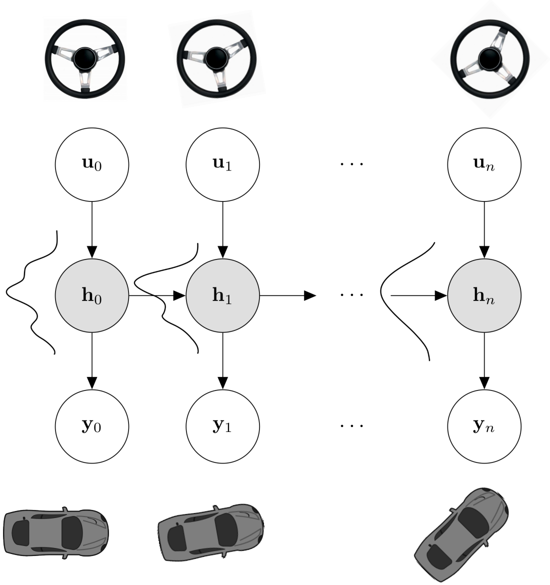
$$\varphi(\varphi(\varphi(\mathbf{x})^\top \mathbf{w}_1)^\top \ldots \mathbf{w}_{L-1})^\top \mathbf{w}_L \rightarrow \text{DNN}$$

Two-layered DGP



An example where uncertainty propagation matters: Recurrent learning.

Dynamics/memory: Deep Recurrent Gaussian Process



Avatar control

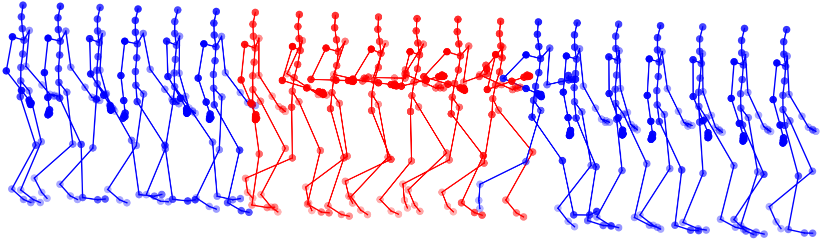


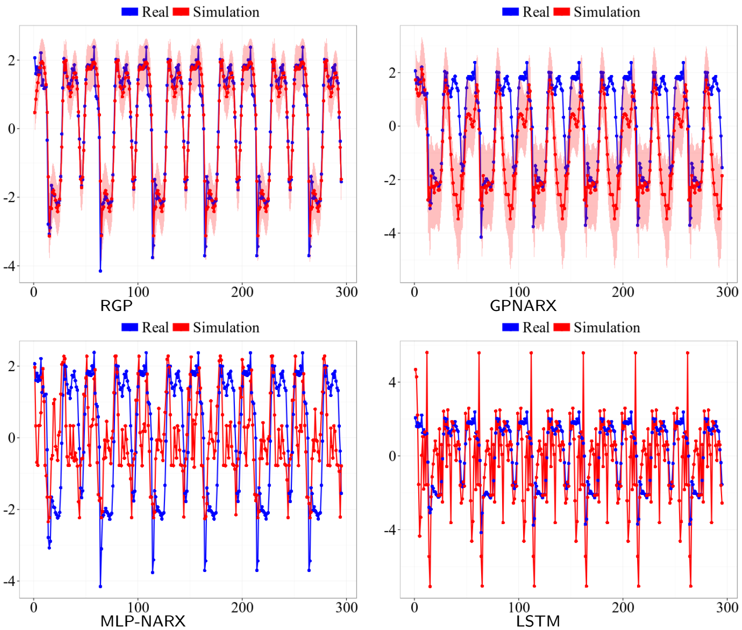
Figure: The generated motion with a step function signal, starting with walking (blue), switching to running (red) and switching back to walking (blue).

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

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