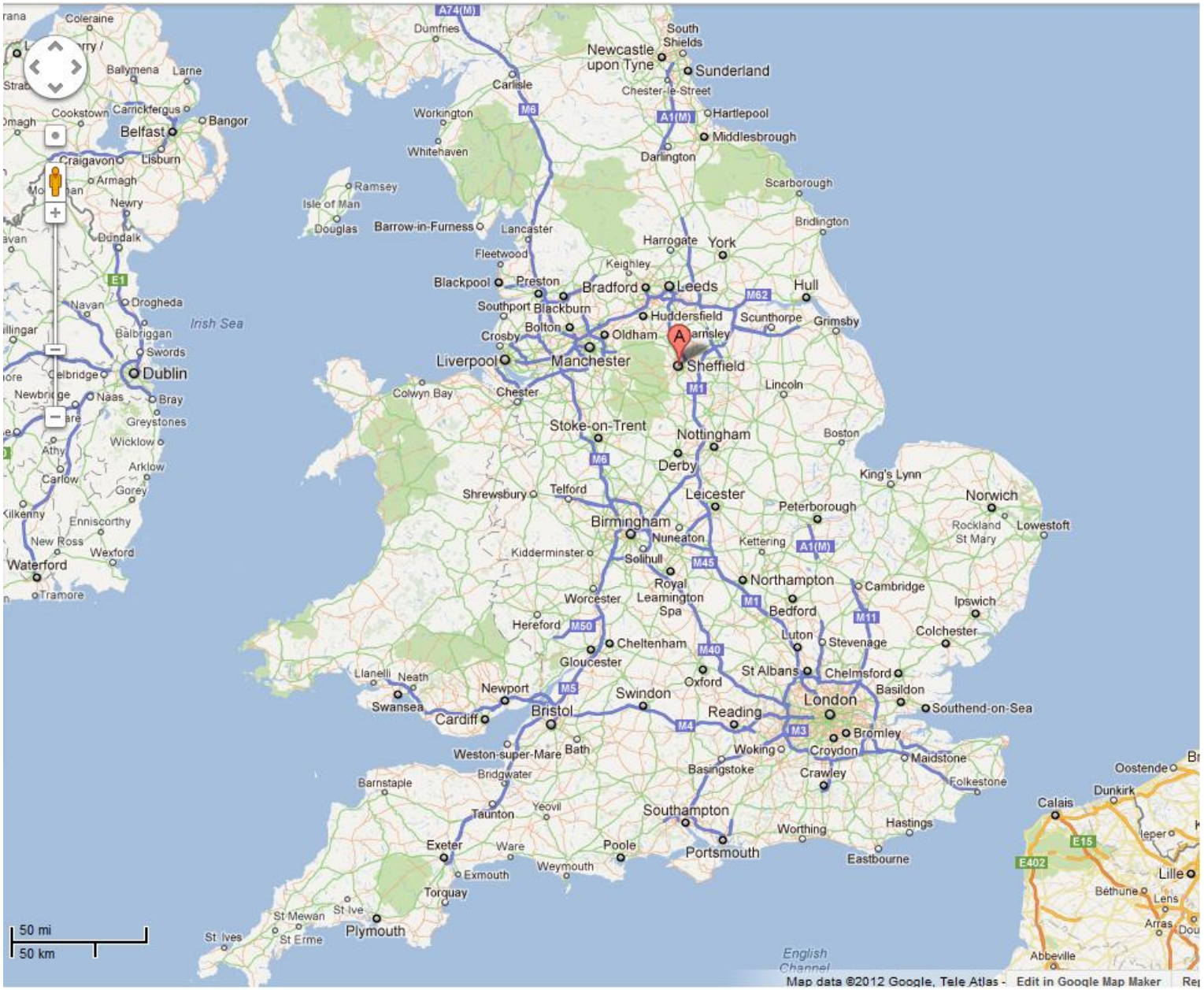
Representation and deep learning with Bayesian non-parametric models

Andreas Damianou

Department of Computer Science, University of Sheffield, UK

Athens University of Economics and Business, 14/10/2015

Sheffield

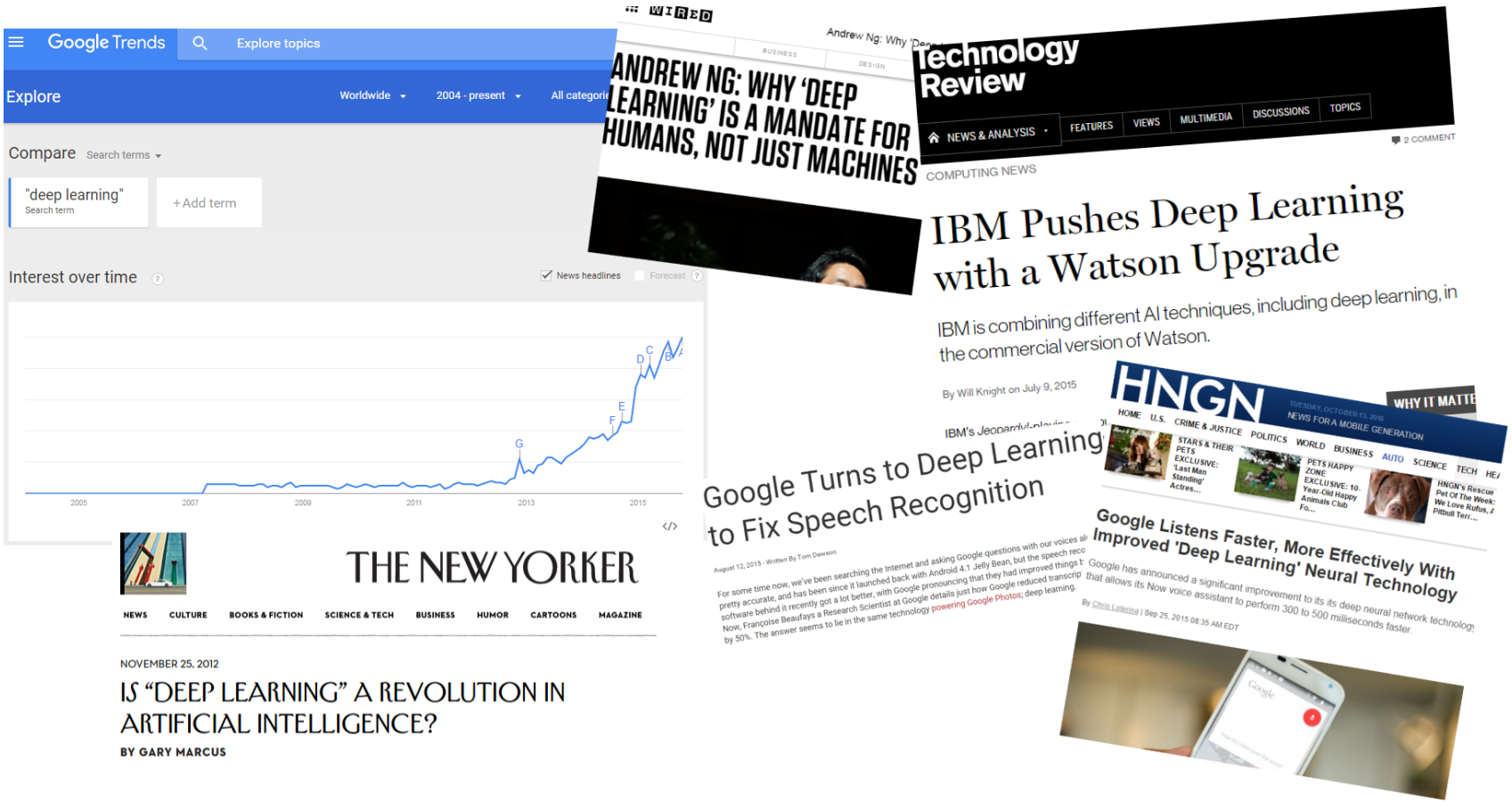

Sheffield
Manchester
Liverpool
Leeds
Bradford
Newcastle upon Tyne
Sunderland
Middlesbrough
Hartlepool
Darlington
Carlisle
Workington
Whitehaven
Dumfries
Barrow-in-Furness
Lancaster
Blackpool
Preston
Southport
Blackburn
Bolton
Oldham
Huddersfield
Scunthorpe
Grimsby
Hull
York
Harrogate
Keighley
Fleetwood
Scarborough
Bridlington
Lincoln
Boston
Nottingham
Derby
Stoke-on-Trent
Chester
Colwyn Bay
Shrewsbury
Telford
Leicester
Peterborough
Norwich
King's Lynn
Birmingham
Nuneaton
Kettering
Northampton
Cambridge
Ipswich
Bedford
Luton
Stevenage
Colchester
Hereford
Worcester
Royal Leamington Spa
Kidderminster
Solihull
Cheltenham
Gloucester
Oxford
St Albans
Chelmsford
Basildon
Southend-on-Sea
London
Reading
Swindon
Bristol
Cardiff
Newport
Swansea
Llanelli
Neath
Weston-super-Mare
Bath
Bridgwater
Taunton
Yeovil
Barnstaple
Exeter
Exmouth
Torquay
Plymouth
St Ives
St Mewan
St Erme
Ware
Poole
Weymouth
Southampton
Portsmouth
Worthing
Hastings
Eastbourne
Basingstoke
Woking
Croydon
Bromley
Crawley
Maidstone
Folkestone
Ramsey
Isle of Man
Douglas
Irish Sea
Dublin
Drogheda
Dundalk
Newry
Armagh
Belfast
Bangor
Larne
Ballymena
Coleraine
Cookstown
Carrickfergus
Craigavon
Lisburn
Navan
Balbriggan
Swords
Bray
Greystones
Wicklow
Arklow
Gorey
Enniscorthy
Wexford
Waterford
Tramore
New Ross
Carlow
Kilkenny
Athy
Celbridge
Naas
Newbridge
Tullamore
Rockland St Mary
Lowestoft
Oostende
Dunkirk
Calais
Lille
Béthune
Lens
Arras
Abbeville
English Channel
50 mi
50 km
Map data ©2012 Google, Tele Atlas - Edit in Google Map Maker

Sheffield Robotics

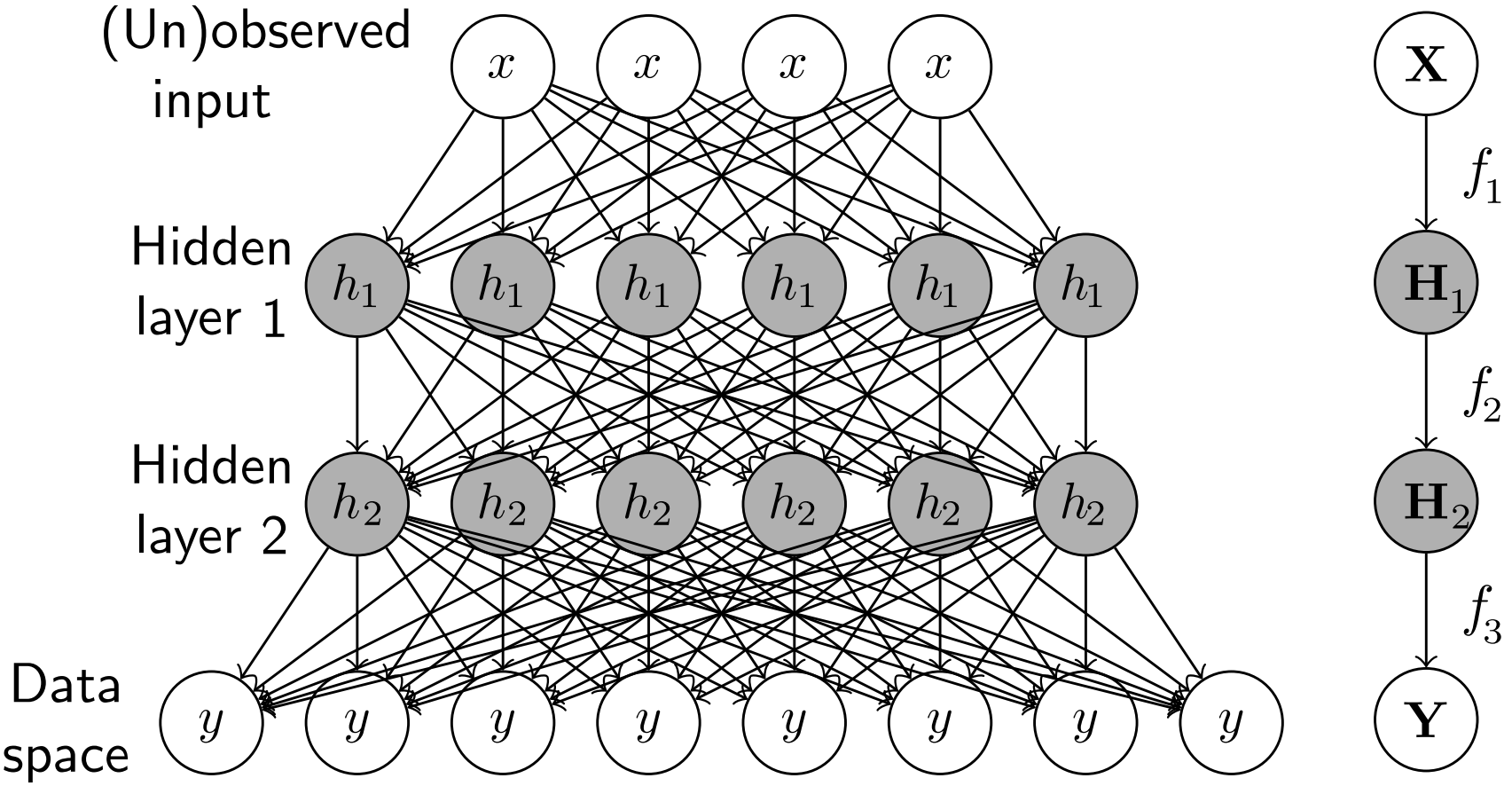
Outline

Outline

Deep learning is very popular



A deep model



$$\mathbf{Y} = f_3(f_2(\cdots f_1(\mathbf{X}))), \qquad \mathbf{H}_i = f_i(\mathbf{H}_{i-1})$$

Representation learning

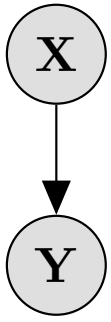
- This talk is *not* about deep learning!
- But I want to highlight the power of *representation learning*...
- ...and the problem of bad *regularization* (a major drawback of current deep learning methods).
- This talk *is* about: Bayesian nonparametric approach to representation learning...
- ...and how it can be linked to deep learning.
- *Gaussian processes* (GPs) will be used as building blocks, i.e. $f \sim \mathcal{GP}$.
- Advantages sought: nonlinear, nonparametric, Bayesian modeling, regularization.

Representation learning

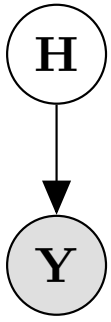
- This talk is *not* about deep learning!
- But I want to highlight the power of *representation learning*...
- ...and the problem of bad *regularization* (a major drawback of current deep learning methods).
- This talk *is* about: Bayesian nonparametric approach to representation learning...
- ...and how it can be linked to deep learning.
- *Gaussian processes* (GPs) will be used as building blocks, i.e. $f \sim \mathcal{GP}$.
- Advantages sought: nonlinear, nonparametric, Bayesian modeling, regularization.

How this talk will proceed...

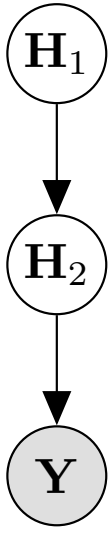
GP = Gaussian process (a particular type of stochastic process)



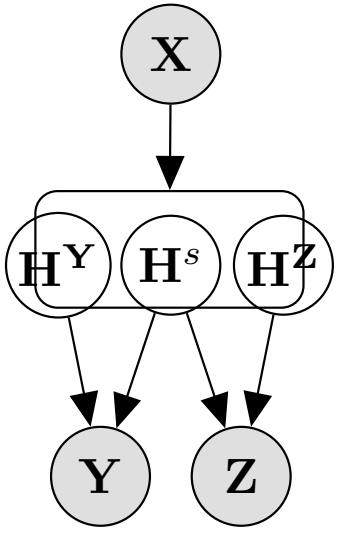
GP



GP-LVM



Deep GP



Multi-view

Outline

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.

Infinite model... but we *always* work with finite sets!

Let's start with a multivariate Gaussian:

$$p(\underbrace{f_1, f_2, \cdots, f_s}_{\mathbf{f}_A}, \underbrace{f_{s+1}, f_{s+2}, \cdots, f_N}_{\mathbf{f}_B}) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}).$$

with:

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_A \\ \boldsymbol{\mu}_B \end{bmatrix} \quad \text{and} \quad \mathbf{K} = \begin{bmatrix} \mathbf{K}_{AA} & \mathbf{K}_{AB} \\ \mathbf{K}_{BA} & \mathbf{K}_{BB} \end{bmatrix}$$

Marginalisation property:

$p(\mathbf{f}_A, \mathbf{f}_B) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$. Then:

$$p(\mathbf{f}_A) = \int_{\mathbf{f}_B} p(\mathbf{f}_A, \mathbf{f}_B) \mathrm{d}\mathbf{f}_B = \mathcal{N}(\boldsymbol{\mu}_A, \mathbf{K}_{AA})$$

Infinite model... but we *always* work with finite sets!

Let's start with a multivariate Gaussian:

$$p(\underbrace{f_1, f_2, \cdots, f_s}_{\mathbf{f}_A}, \underbrace{f_{s+1}, f_{s+2}, \cdots, f_N}_{\mathbf{f}_B}) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}).$$

with:

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_A \\ \boldsymbol{\mu}_B \end{bmatrix} \quad \text{and} \quad \mathbf{K} = \begin{bmatrix} \mathbf{K}_{AA} & \mathbf{K}_{AB} \\ \mathbf{K}_{BA} & \mathbf{K}_{BB} \end{bmatrix}$$

Marginalisation property:

$$p(\mathbf{f}_A, \mathbf{f}_B) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}). \quad \text{Then:}$$

$$p(\mathbf{f}_A) = \int_{\mathbf{f}_B} p(\mathbf{f}_A, \mathbf{f}_B) \mathrm{d}\mathbf{f}_B = \mathcal{N}(\boldsymbol{\mu}_A, \mathbf{K}_{AA})$$

Infinite model... but we *always* work with finite sets!

In the GP context:

$$\boldsymbol{\mu}_\infty = \begin{bmatrix} \boldsymbol{\mu}_{\mathbf{x}} \\ \cdots \\ \cdots \end{bmatrix} \quad \text{and} \quad \mathbf{K}_\infty = \begin{bmatrix} \mathbf{K}_{\mathbf{xx}} & \cdots \\ \cdots & \cdots \end{bmatrix}$$

Posterior is also Gaussian!

$$p(\mathbf{f}_A, \mathbf{f}_B) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}). \quad \text{Then:}$$
$$p(\mathbf{f}_A | \mathbf{f}_B) = \mathcal{N}(\cdots, \cdots)$$

In the GP context this can be used for inter/extrapolation:

$$p(f_* | f_1, \cdots, f_N) = p(f(x_*) | f(x_1), \cdots, f(x_N)) \sim \mathcal{N}$$

But where is $\mathbf{K}_{..}$ coming from in GPs?

Posterior is also Gaussian!

$$p(\mathbf{f}_A, \mathbf{f}_B) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}). \quad \text{Then:}$$
$$p(\mathbf{f}_A | \mathbf{f}_B) = \mathcal{N}(\cdots, \cdots)$$

In the GP context this can be used for inter/extrapolation:

$$p(f_* | f_1, \cdots, f_N) = p(f(x_*) | f(x_1), \cdots, f(x_N)) \sim \mathcal{N}$$

But where is $\mathbf{K}_{..}$ coming from in GPs?

Posterior is also Gaussian!

$$p(\mathbf{f}_A, \mathbf{f}_B) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}). \quad \text{Then:}$$
$$p(\mathbf{f}_A|\mathbf{f}_B) = \mathcal{N}(\cdots, \cdots)$$

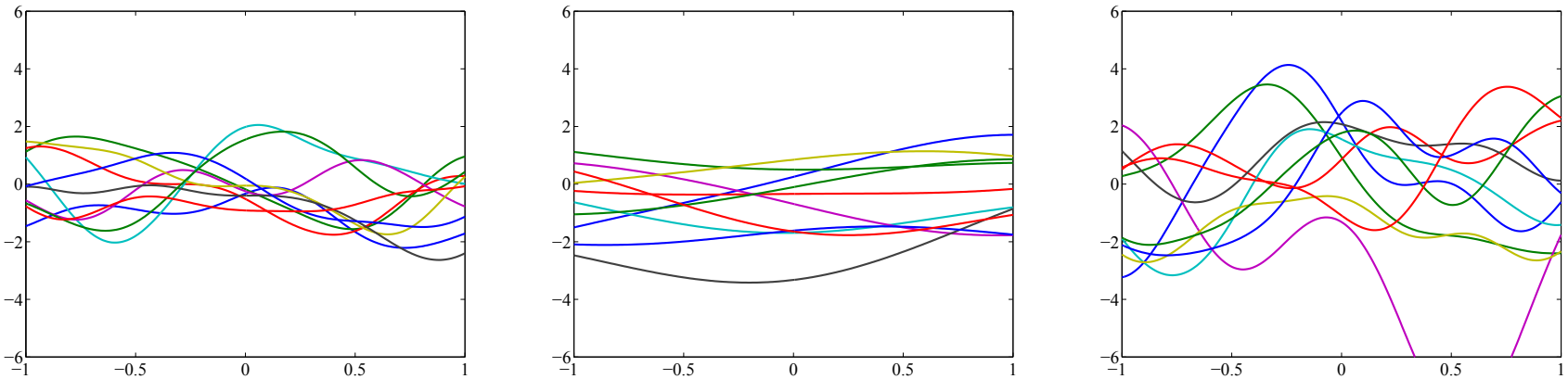
In the GP context this can be used for inter/extrapolation:

$$p(f_*|f_1, \cdots, f_N) = p(f(x_*)|f(x_1), \cdots, f(x_N)) \sim \mathcal{N}$$

But where is $\mathbf{K}_{..}$ coming from in GPs?

Covariance samples and hyperparameters

- $k(x, x') = \alpha \exp\left(-\frac{\gamma}{2}(x - x')^\top (x - x')\right)$
- The hyperparameters of the cov. function define the properties (and NOT an explicit form) of the sampled functions

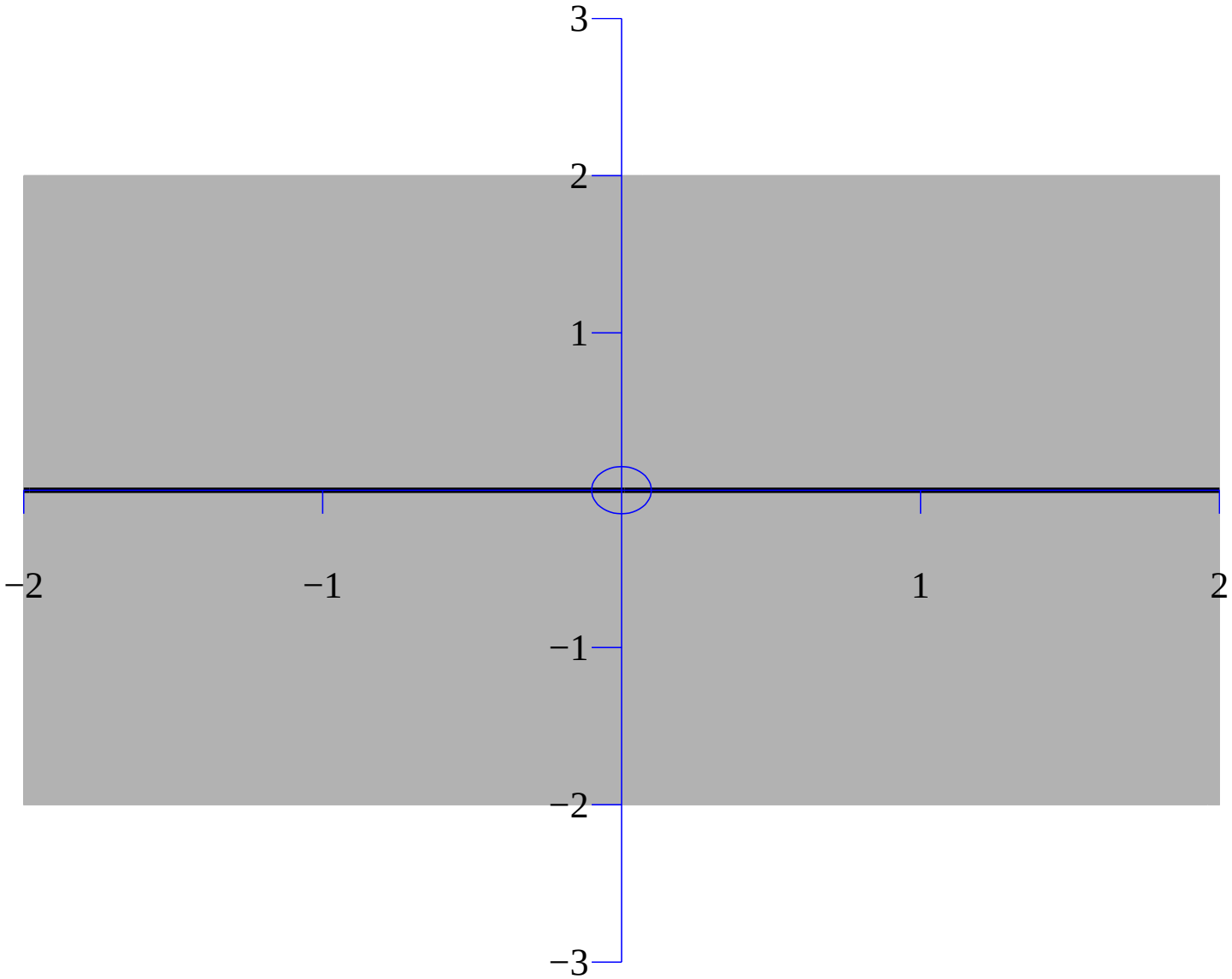


Incorporating Gaussian noise is tractable

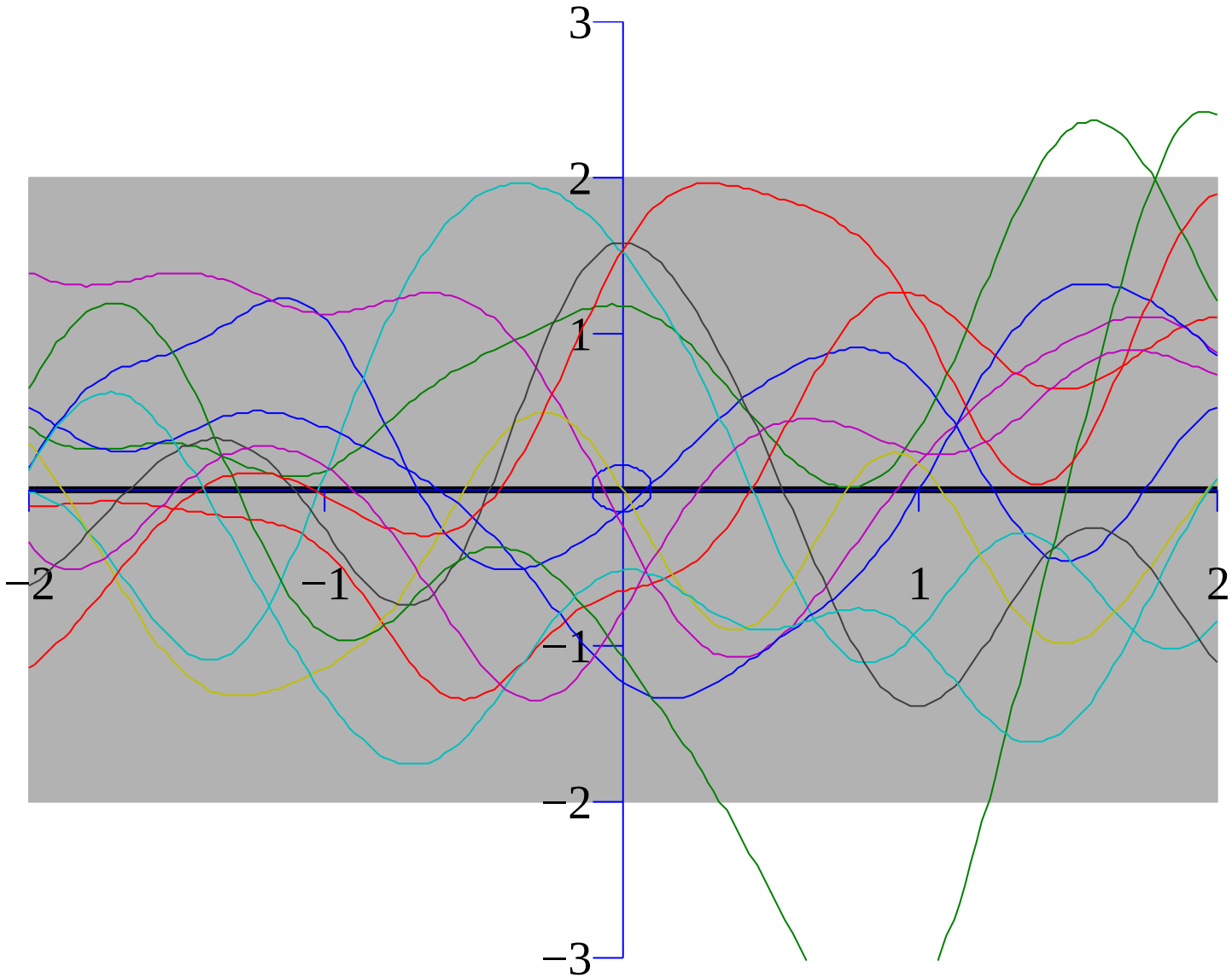
- So far we assumed: $\mathbf{f} = f(\mathbf{X})$
- Assuming that we only observe noisy versions $\mathbf{y}$ of the true outputs $\mathbf{f}$:

$$\mathbf{y} = f(\mathbf{X}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

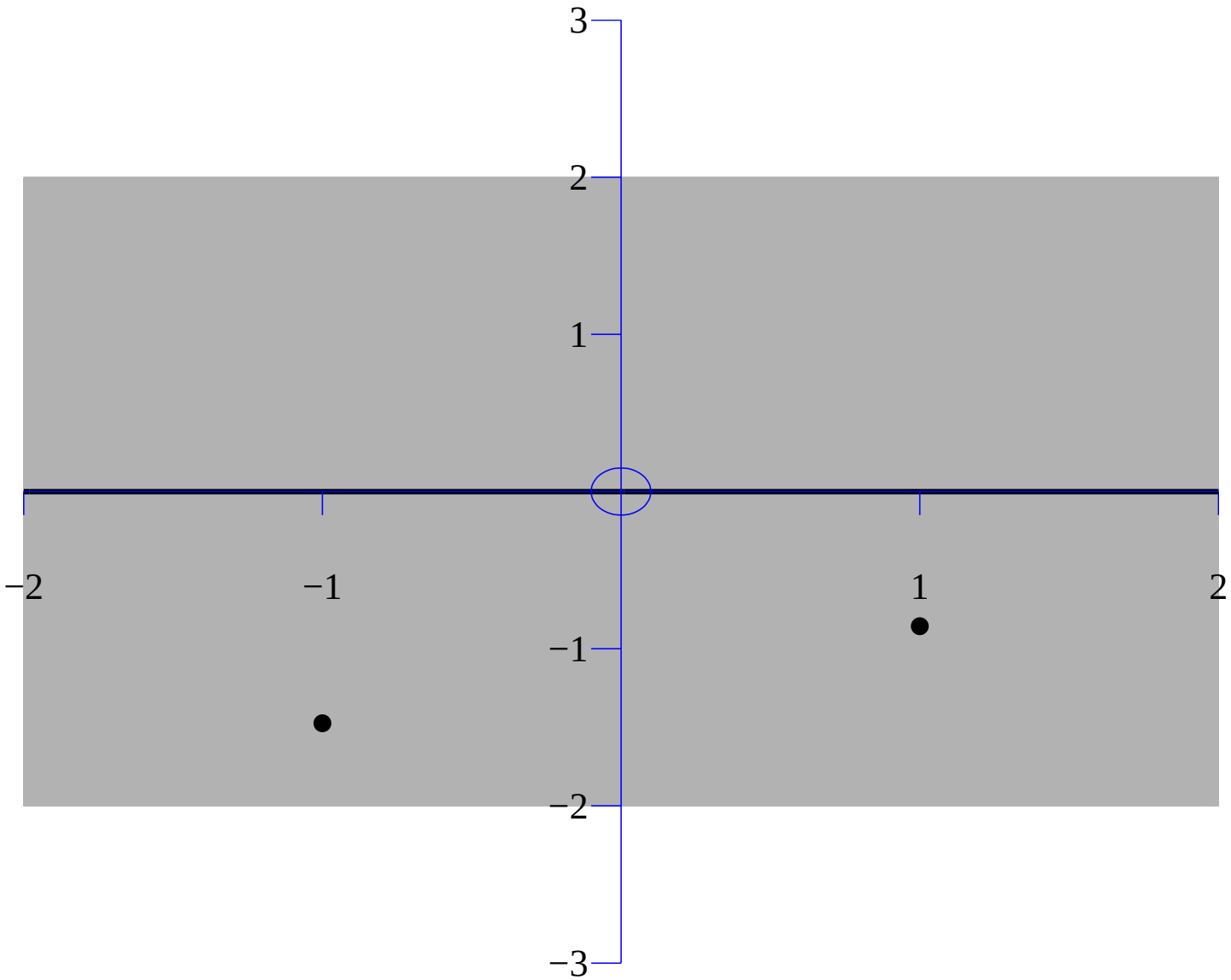
Fitting the data *(shaded area is* **uncertainty***)*



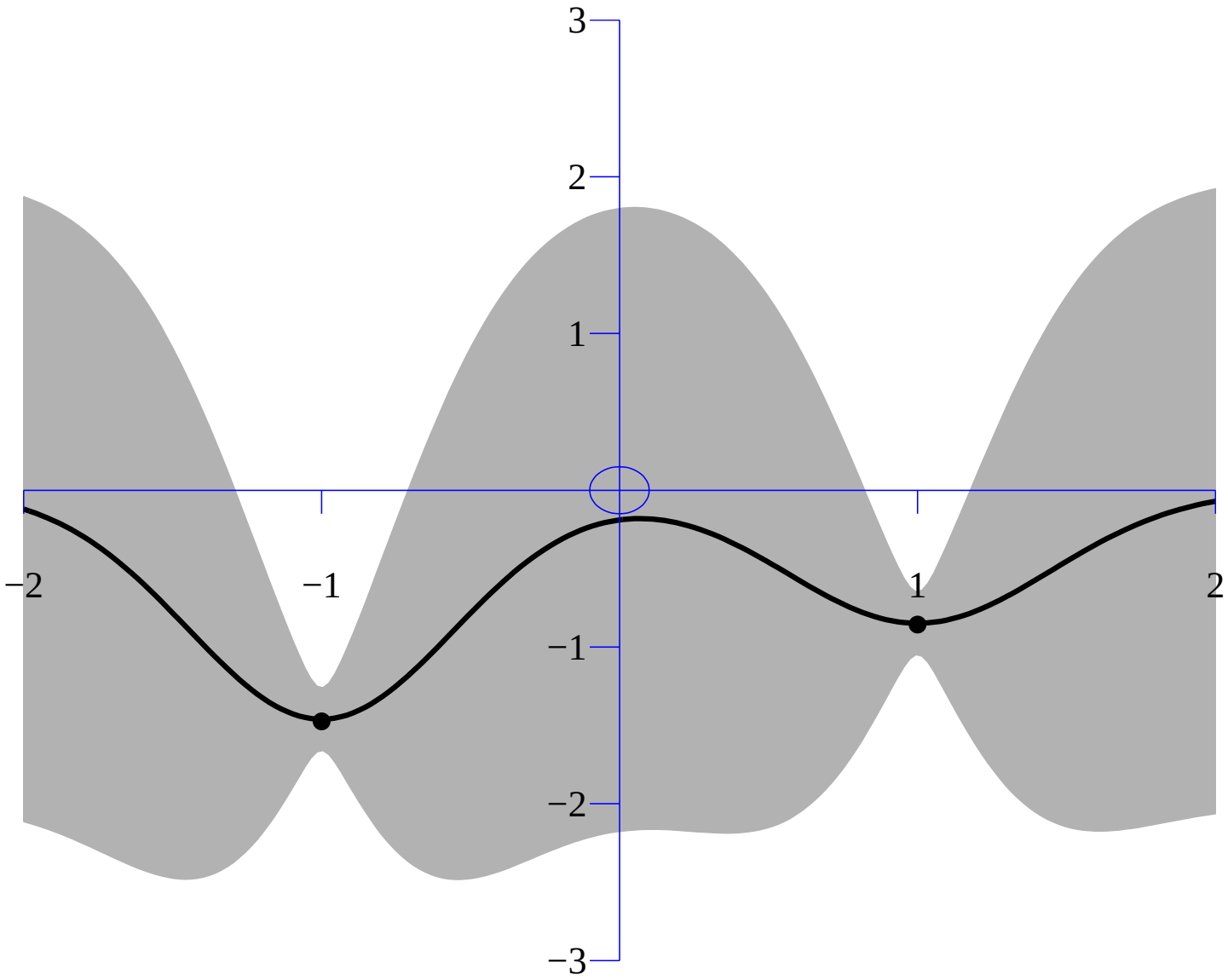
Fitting the data - Prior Samples



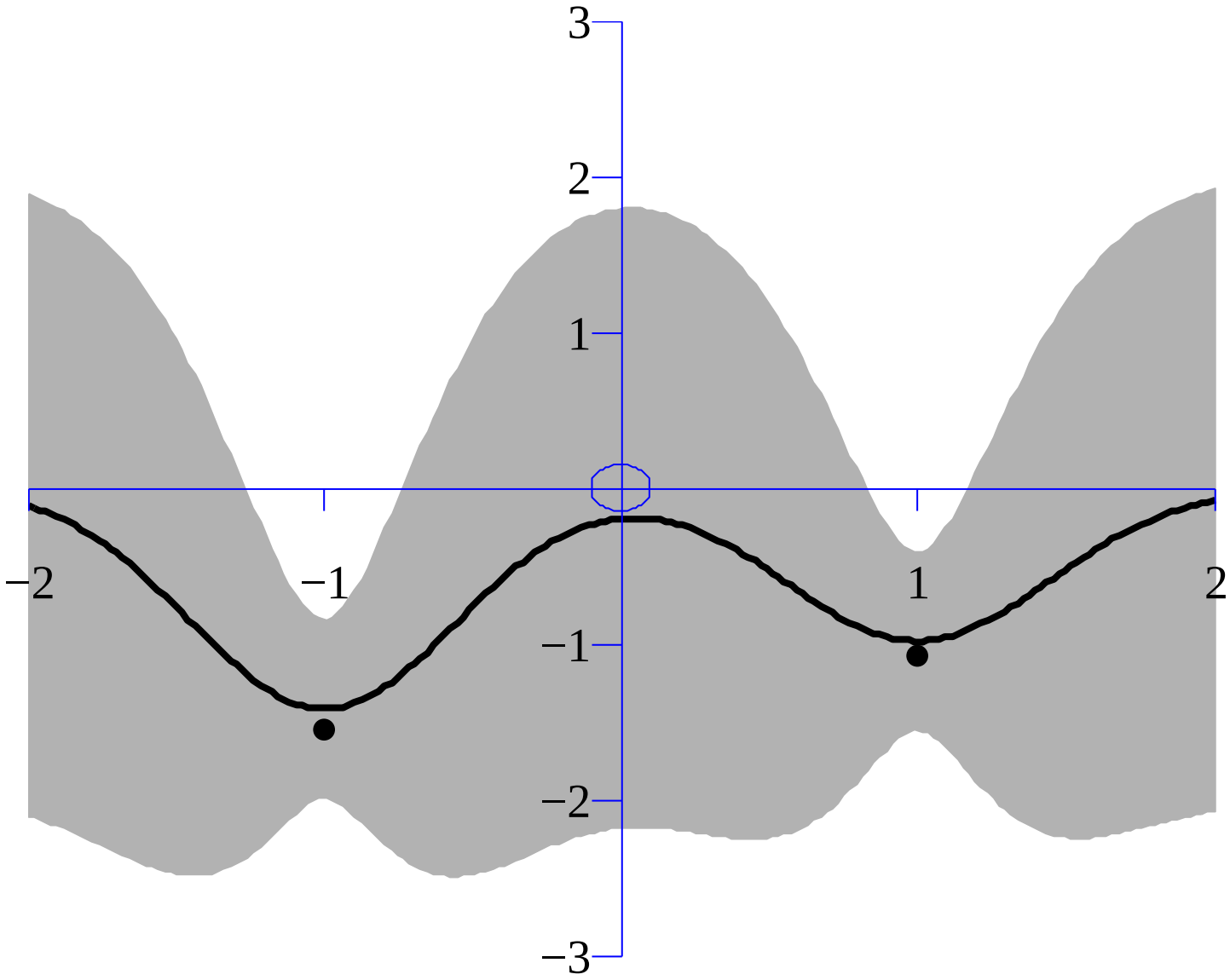
Fitting the data



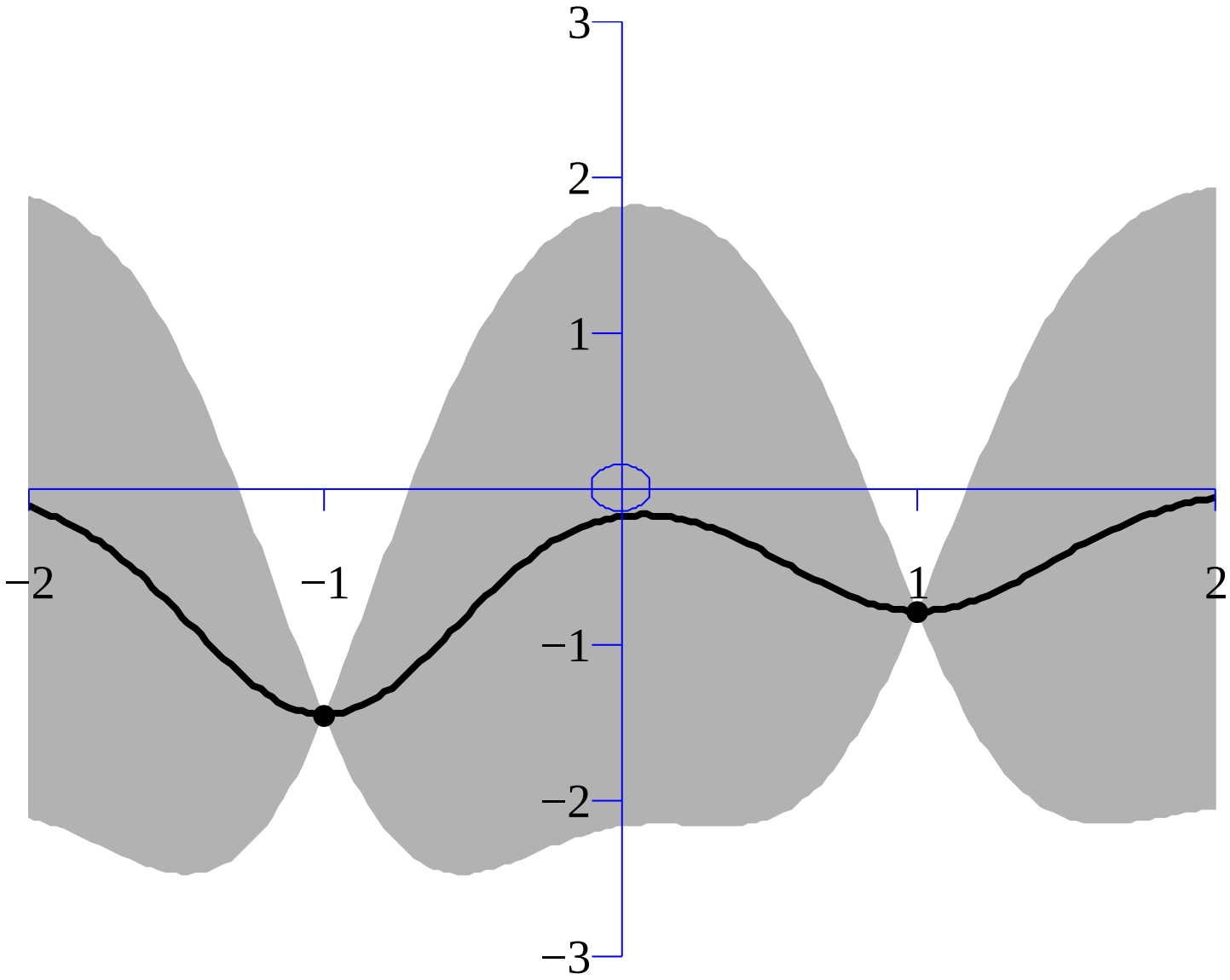
Fitting the data



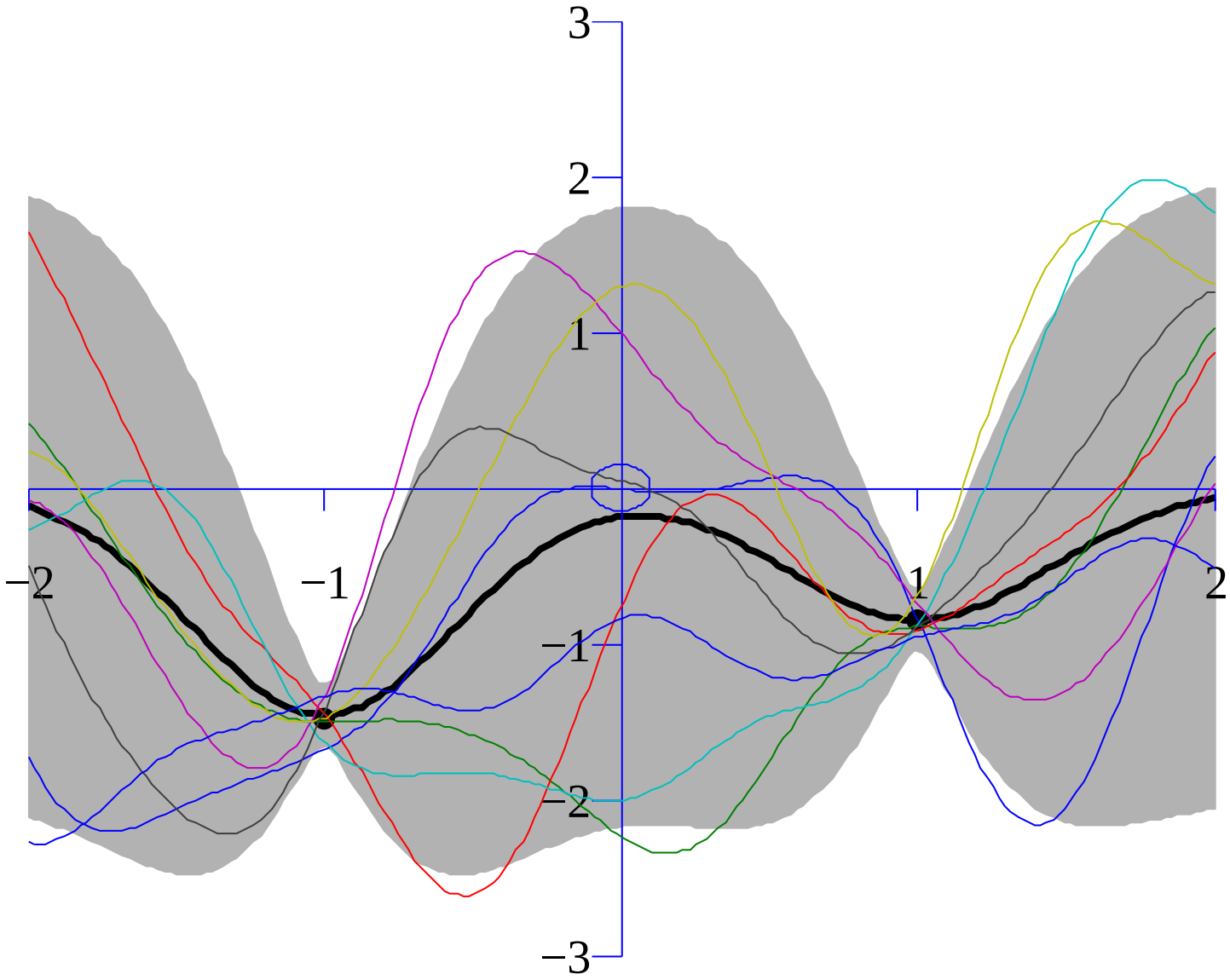
Fitting the data - more noise



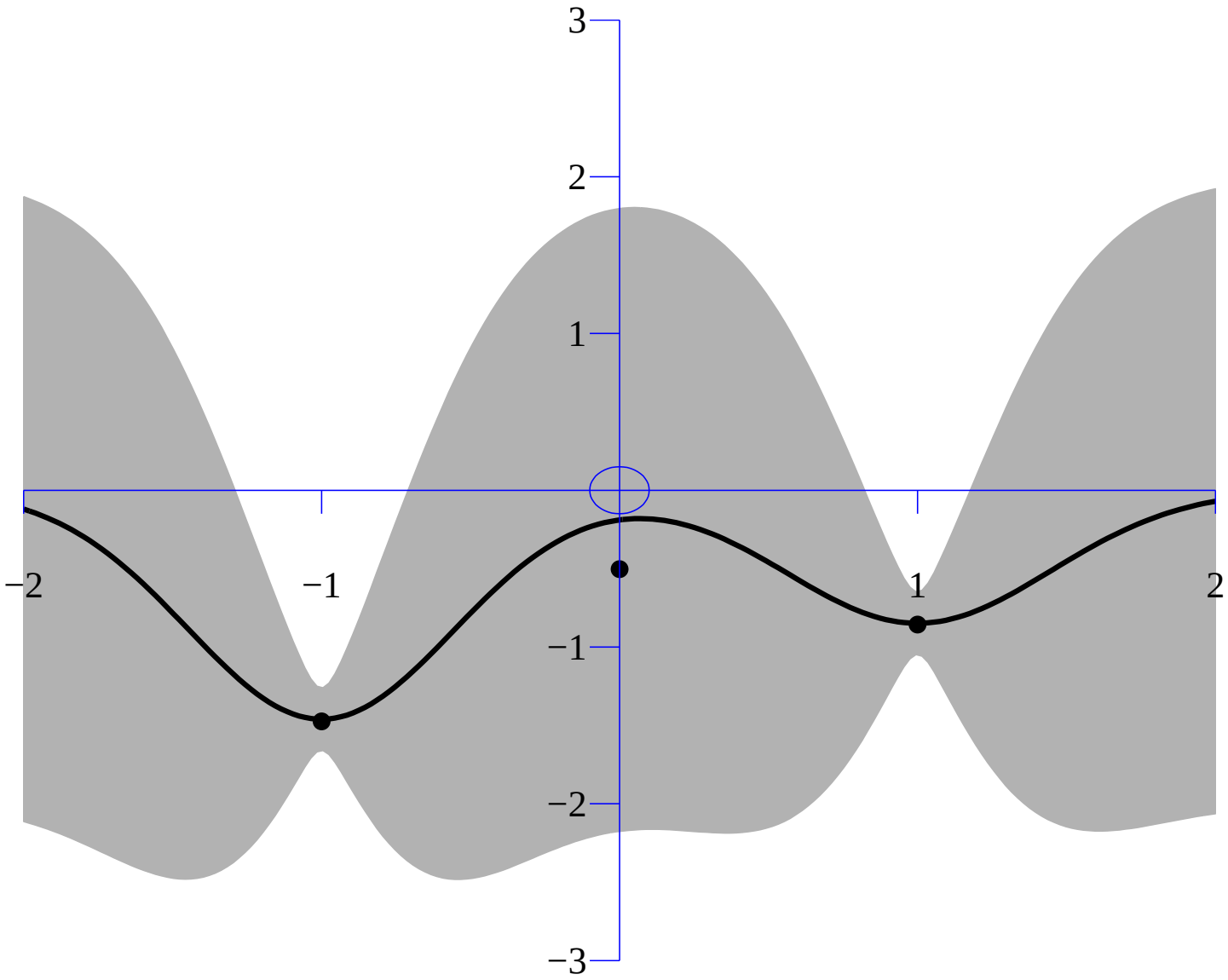
Fitting the data - no noise



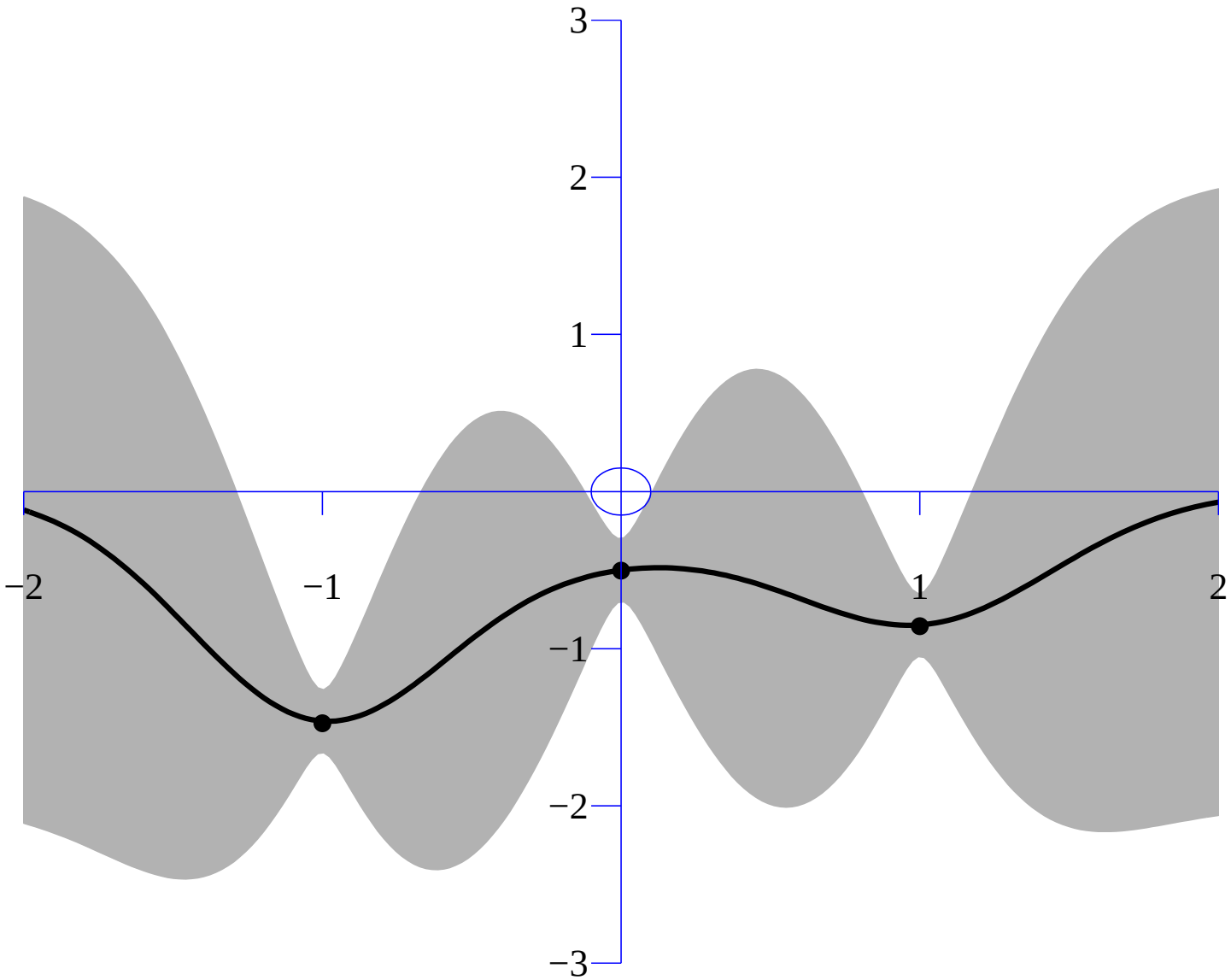
Fitting the data - Posterior samples



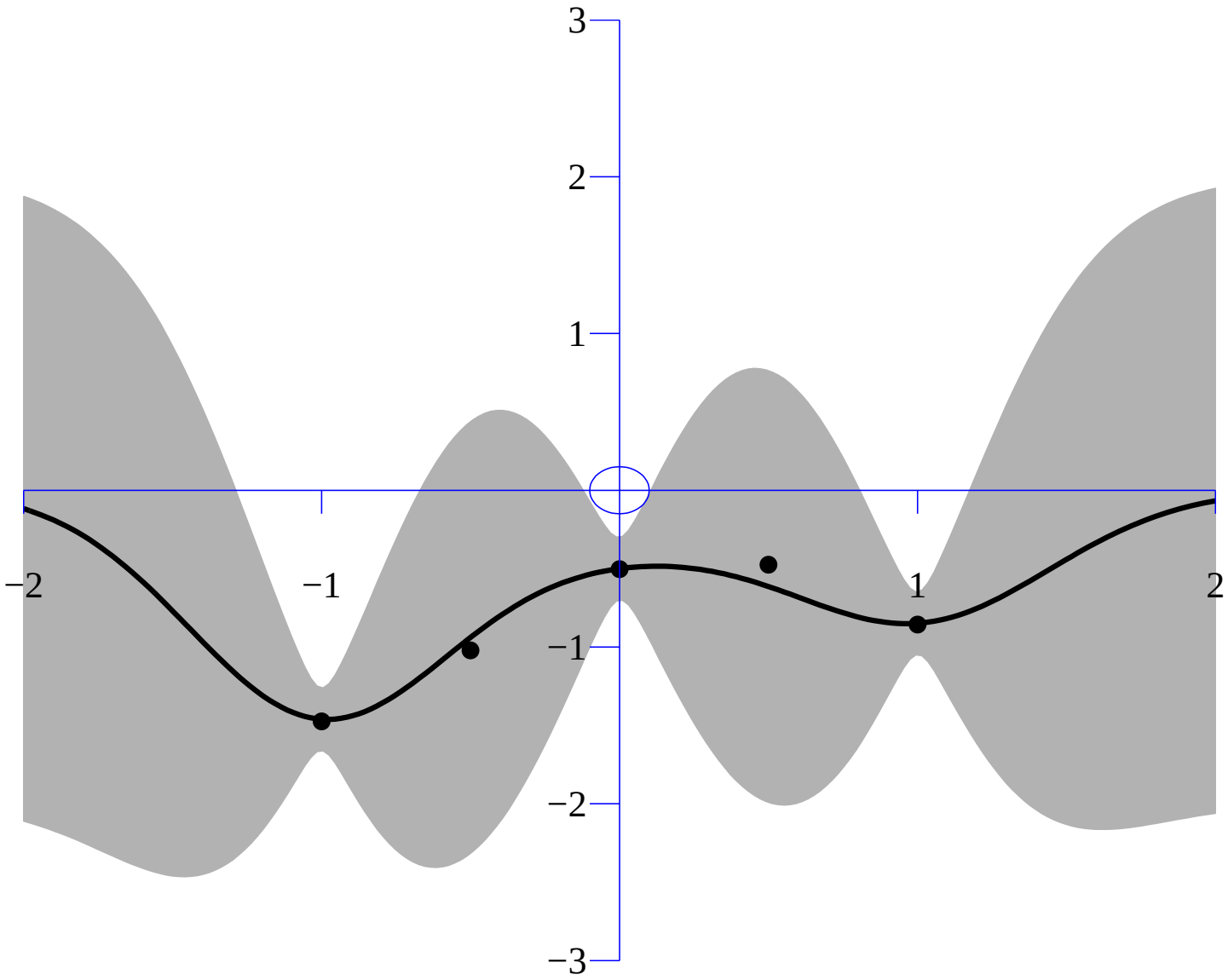
Fitting the data



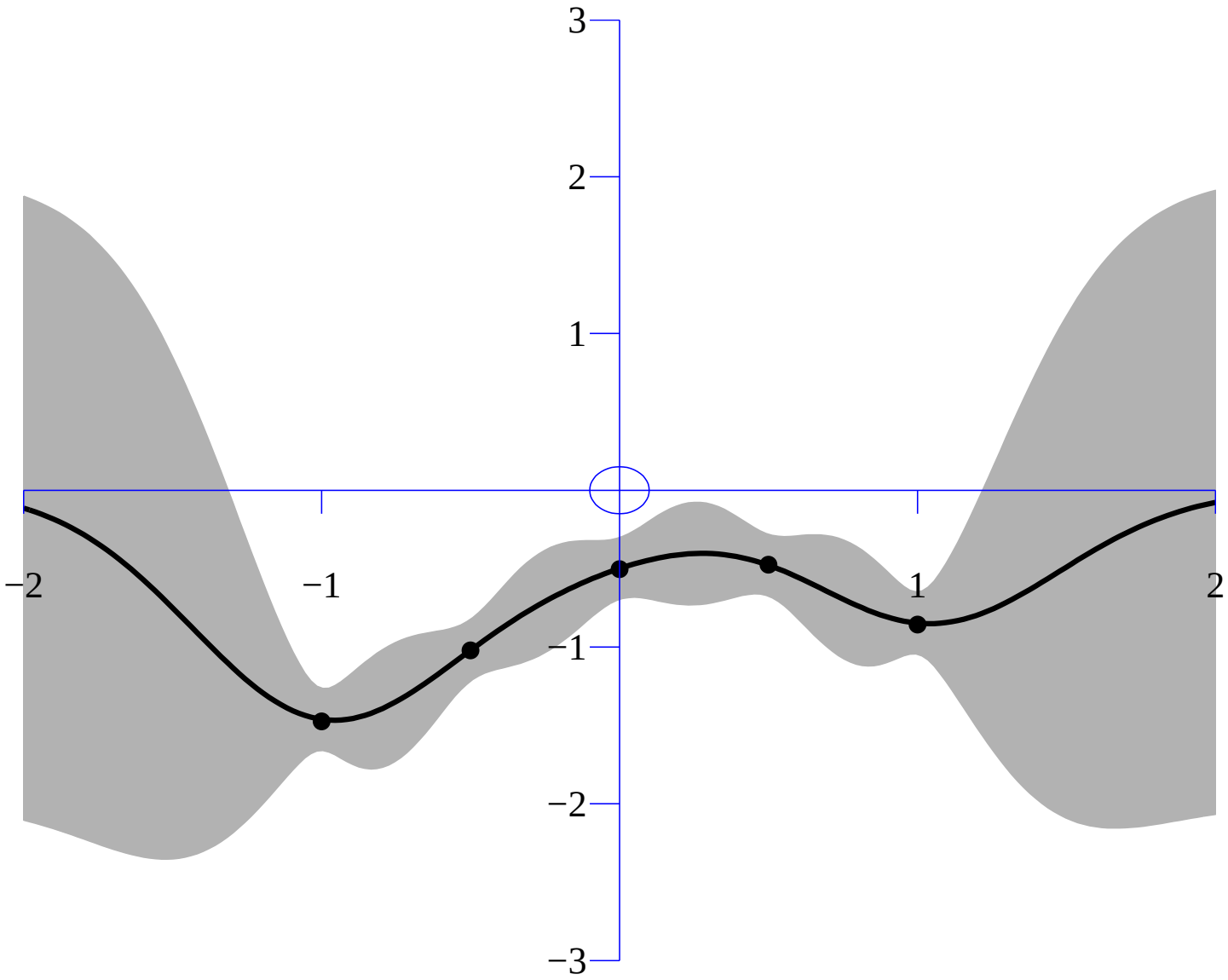
Fitting the data



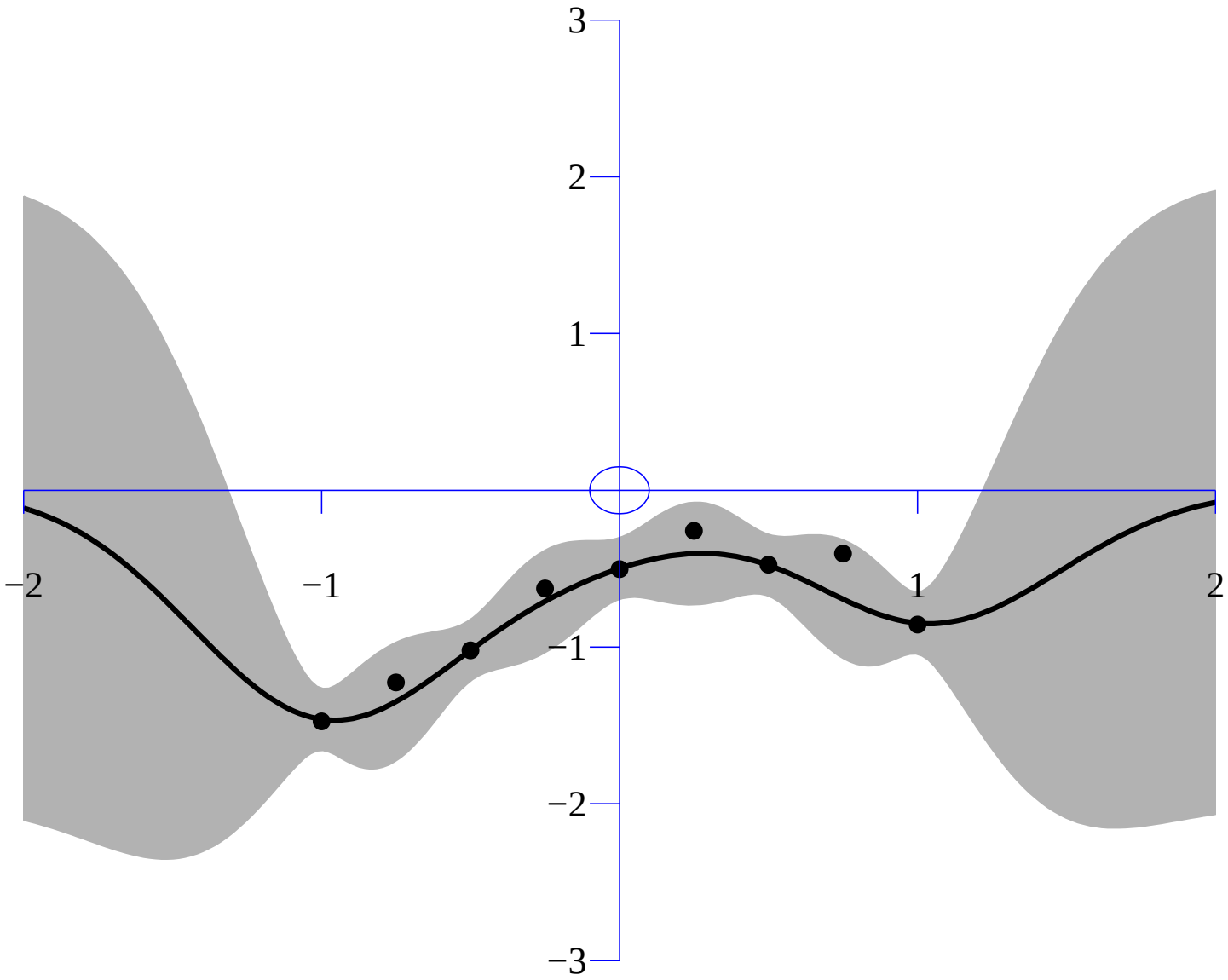
Fitting the data



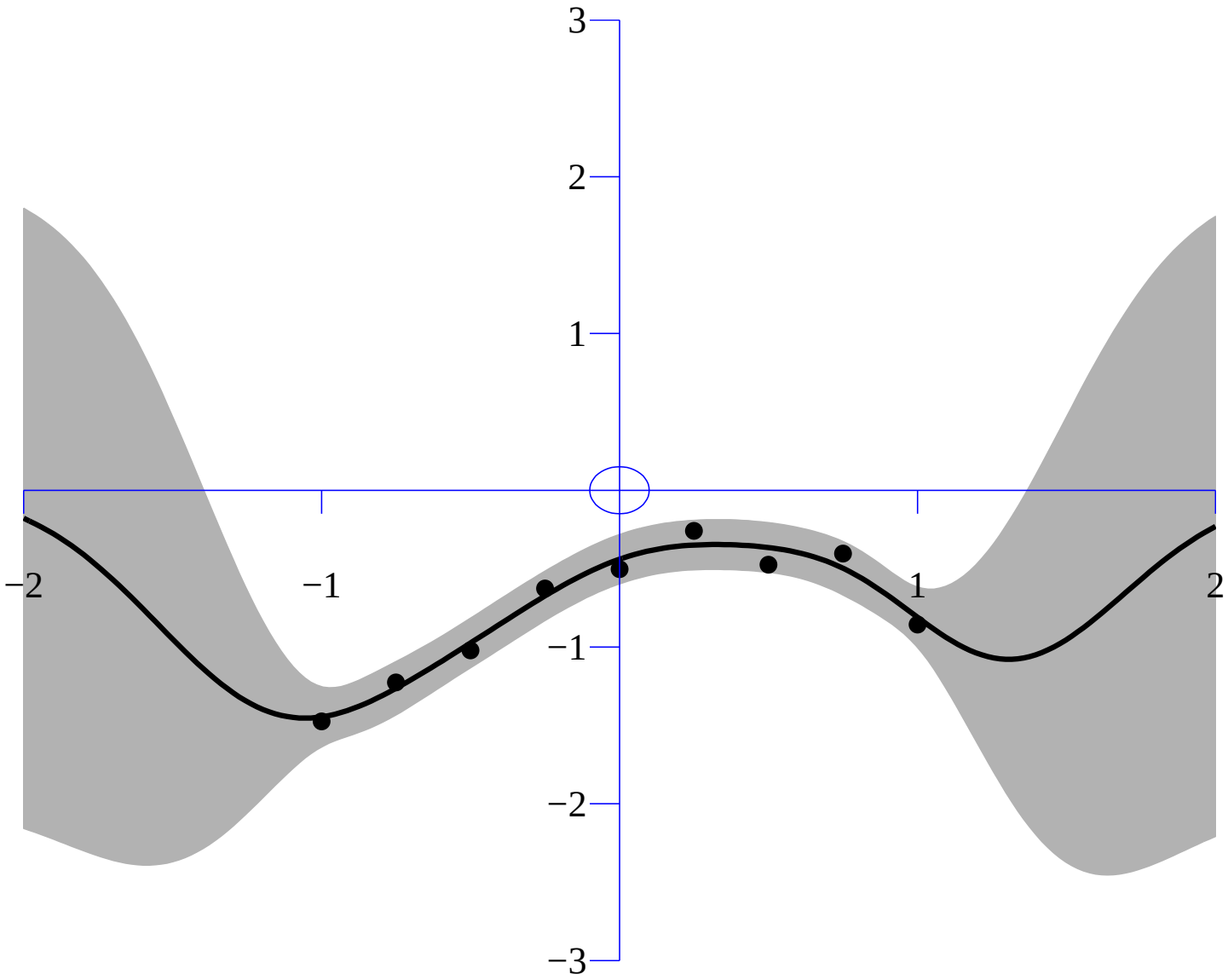
Fitting the data



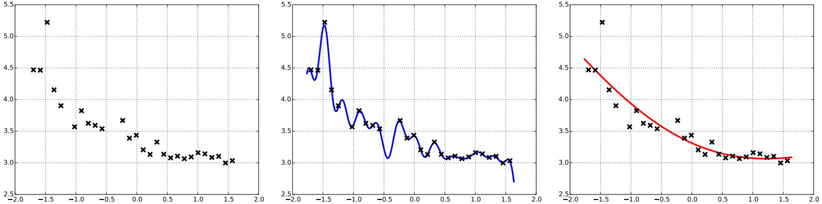
Fitting the data



Fitting the data

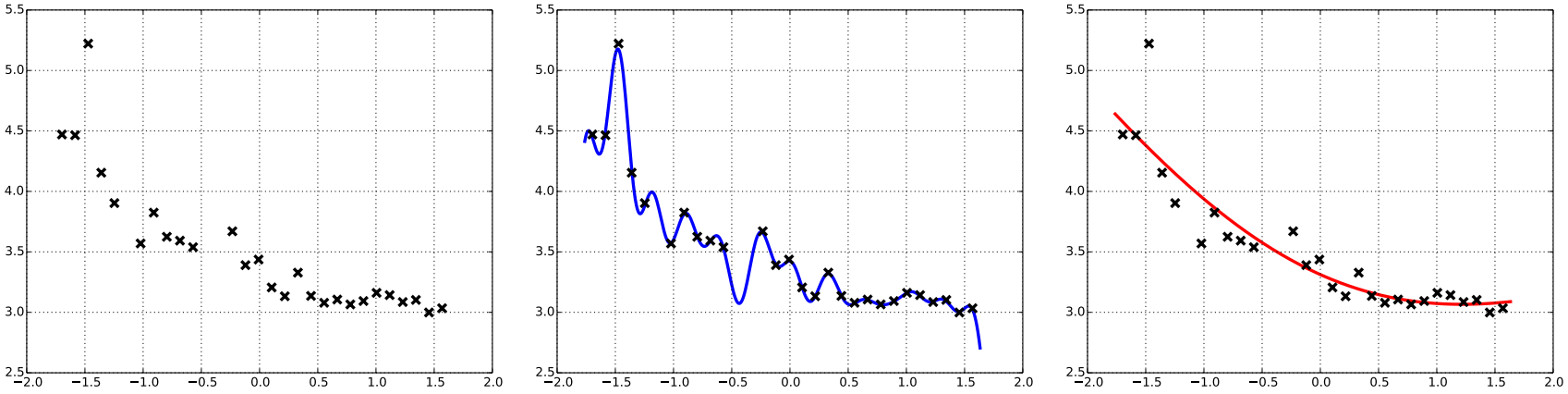


Curve fitting



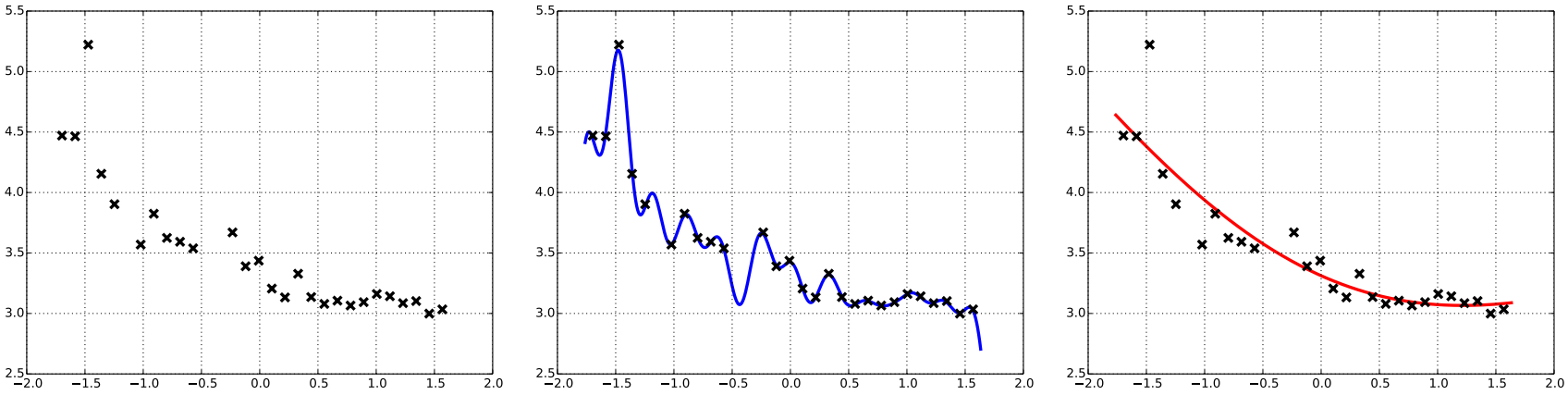
- Which curve fits the data better?

Curve fitting



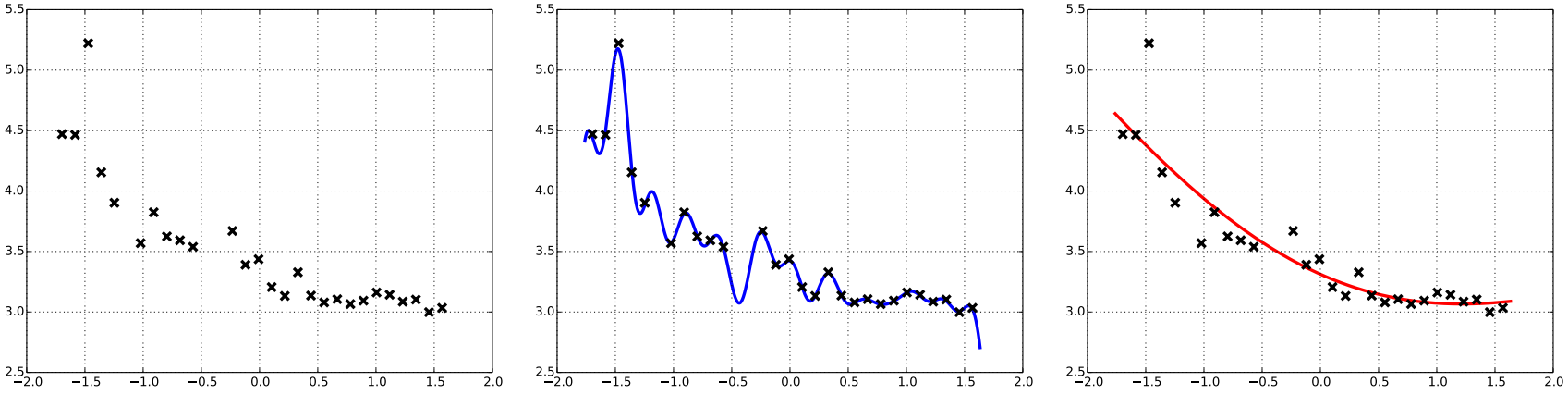
- Which curve fits the data better?
- Which curve is more "complex"?

Curve fitting



- Which curve fits the data better?
- Which curve is more "complex"?
- Which curve is better overall?

Curve fitting



- Which curve fits the data better?
- Which curve is more "complex"?
- Which curve is better overall?

Need a good balance between data fit vs overfitting!

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

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

- Answer: Integrate over the function itself!
- This is associated with the Bayesian methodology.
- So, we will average out all possible function forms, under a (GP) prior!

Recap:

$$\text{ML:} \quad \underset{\mathbf{w}}{\operatorname{argmax}}\ p(\mathbf{y}|\mathbf{w}, \phi(\mathbf{x})) \qquad \text{e.g. } \mathbf{y} = \phi(\mathbf{x})^\top \mathbf{w} + \boldsymbol{\epsilon}$$

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

- $\boldsymbol{\theta}$ 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!
- This is associated with the Bayesian methodology.
- So, we will average out all possible function forms, under a (GP) prior!

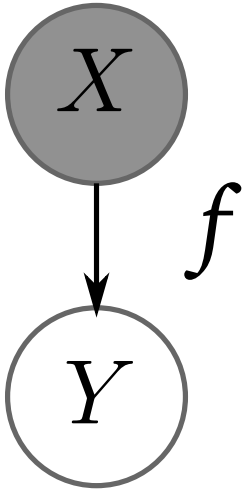
Recap:

$$\text{ML:} \quad \underset{\mathbf{w}}{\operatorname{argmax}}\ p(\mathbf{y}|\mathbf{w}, \phi(\mathbf{x})) \qquad \text{e.g. } \mathbf{y} = \phi(\mathbf{x})^\top \mathbf{w} + \boldsymbol{\epsilon}$$

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

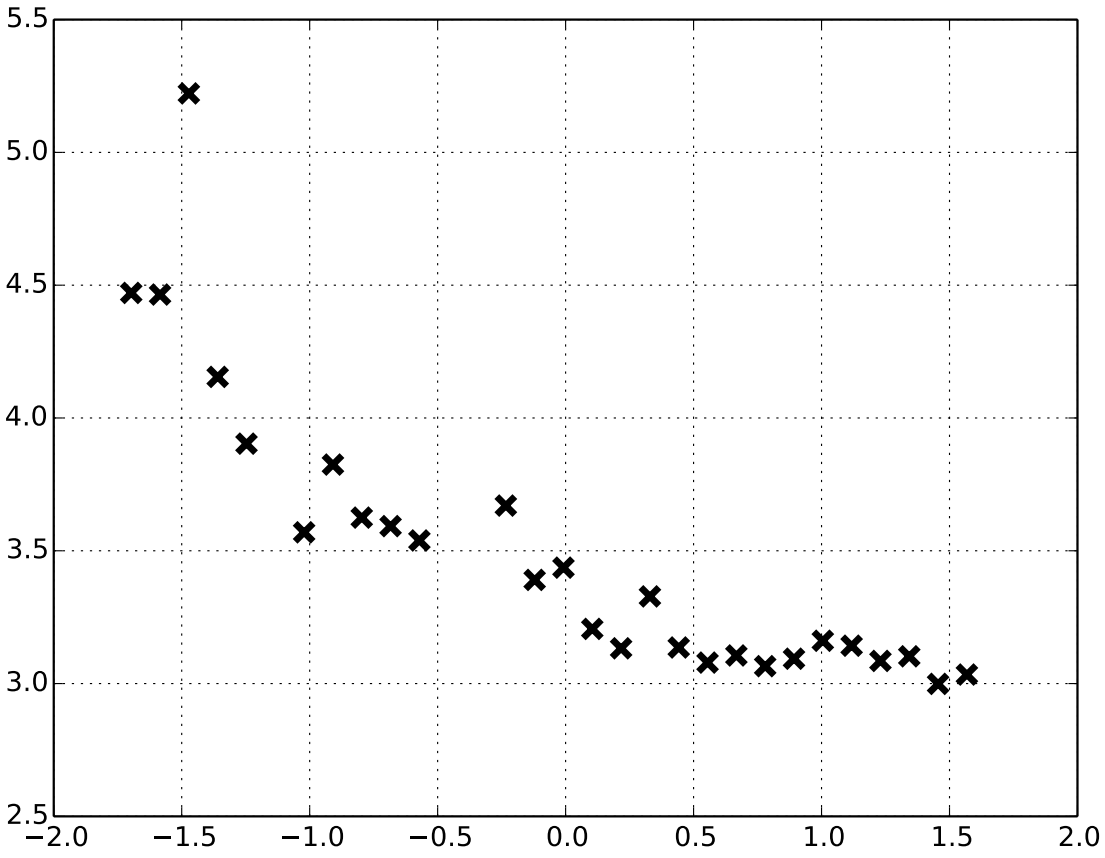
- $\boldsymbol{\theta}$ are *hyper*parameters
- The Bayesian approach (GP) automatically balances the data-fitting with the complexity penalty.

Unsupervised learning: GP-LVM



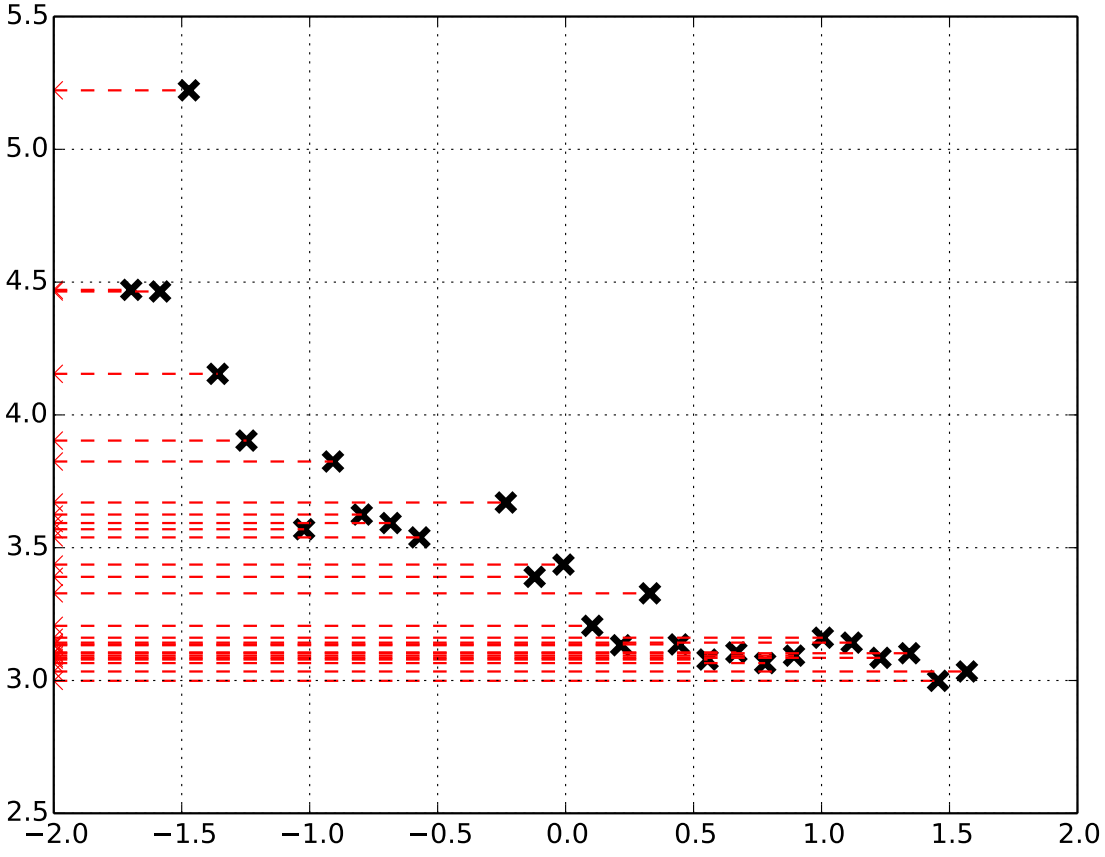
- If $\mathbf{X}$ is unobserved, treat it as a parameter and optimize over it.

Fitting the GP-LVM



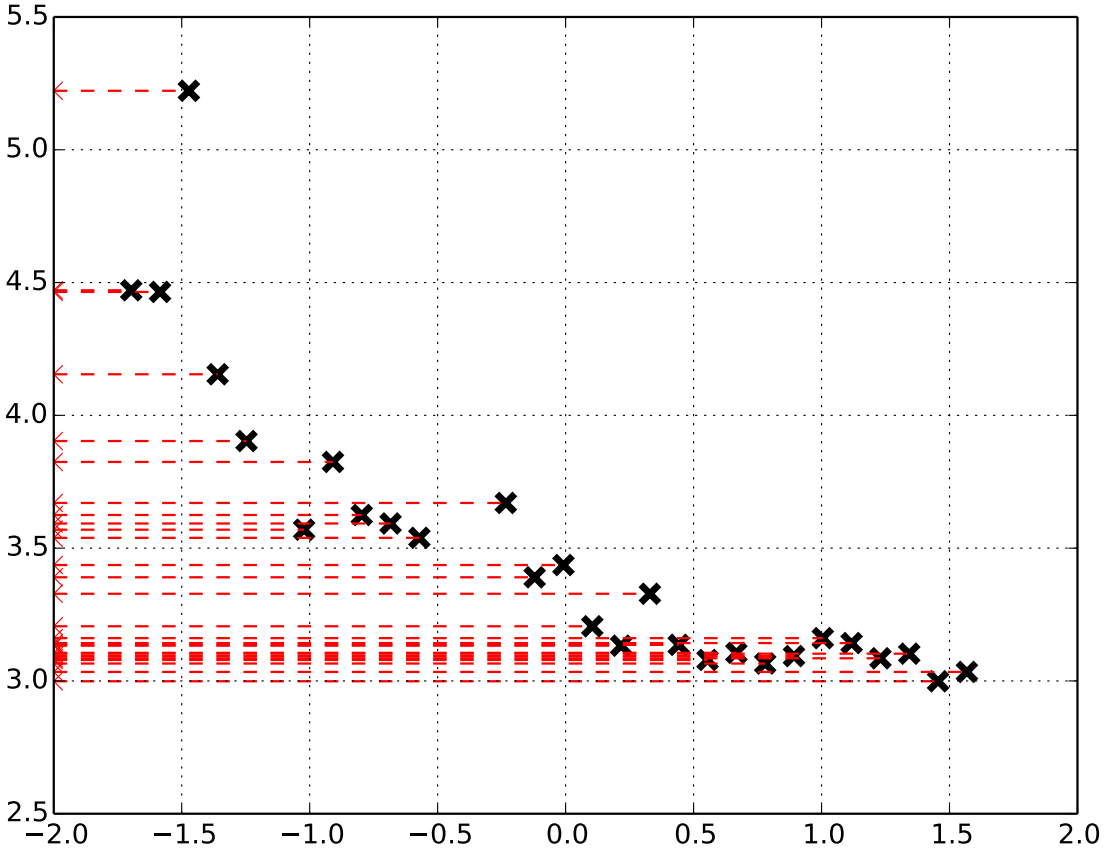
Fitting the GP-LVM

Figure credits: C. H. Ek



Fitting the GP-LVM

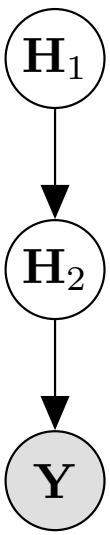
Figure credits: C. H. Ek



- Additional difficulty: x's are also missing!
- Improvement: Invoke the Bayesian methodology to find x's too.

Outline

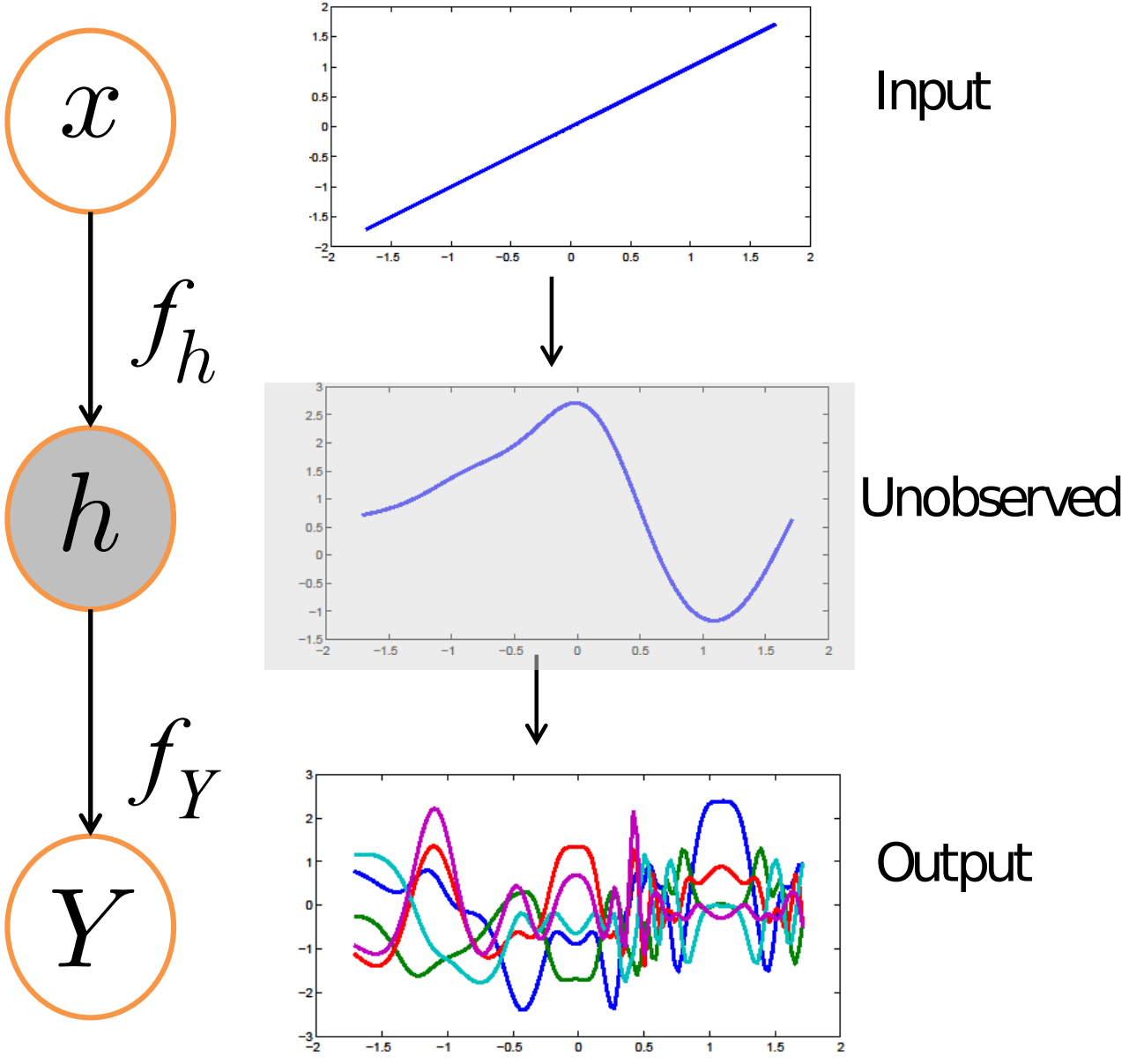
Deep Gaussian processes



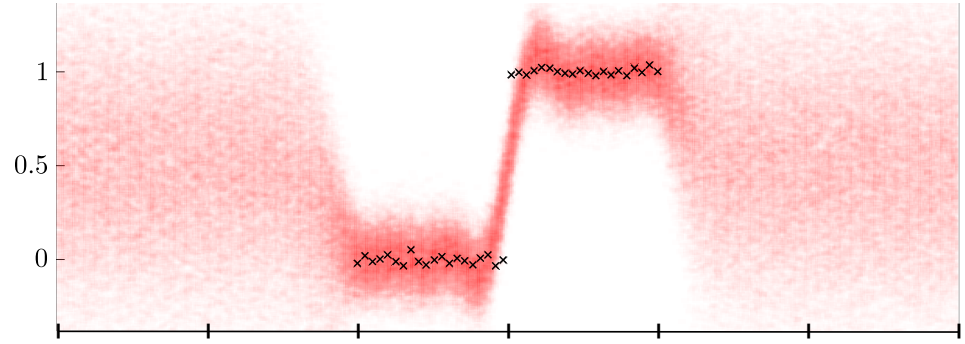
- Define a recursive stacked construction

$$f(\mathbf{h}) \rightarrow \text{GP}$$
$$f(h_2(\mathbf{h}_1)) \rightarrow \text{stacked GP}$$
$$f(h(h(h\cdots(\mathbf{h}_1)))) \rightarrow \text{deep GP}$$

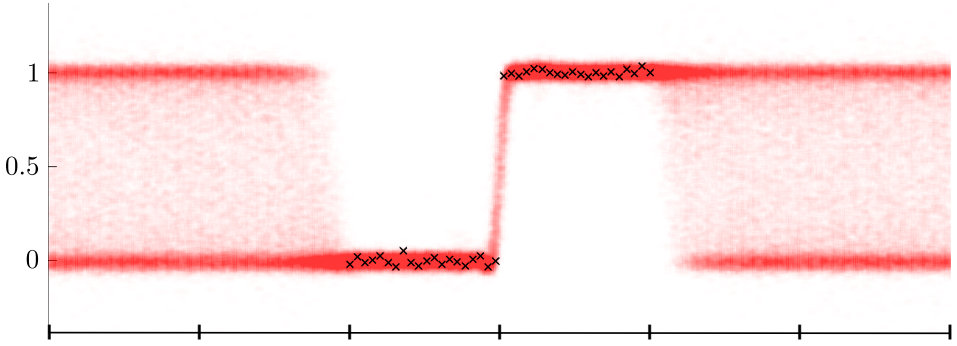
Sampling from a deep GP



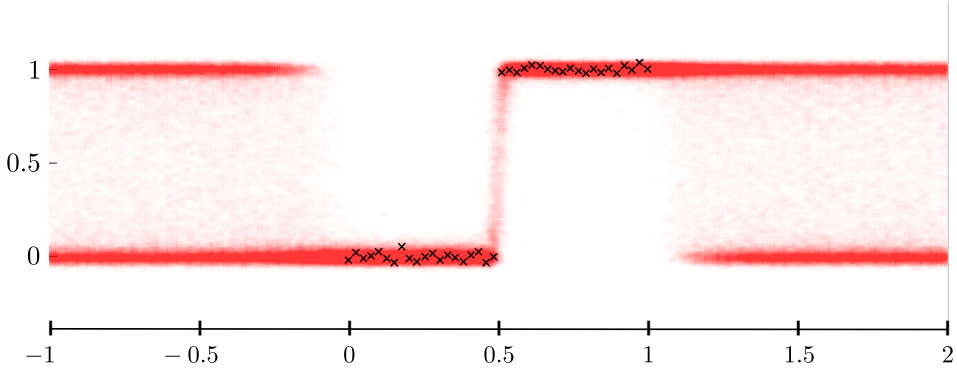
Deep GP: Step function (credits for idea to J. Hensman)



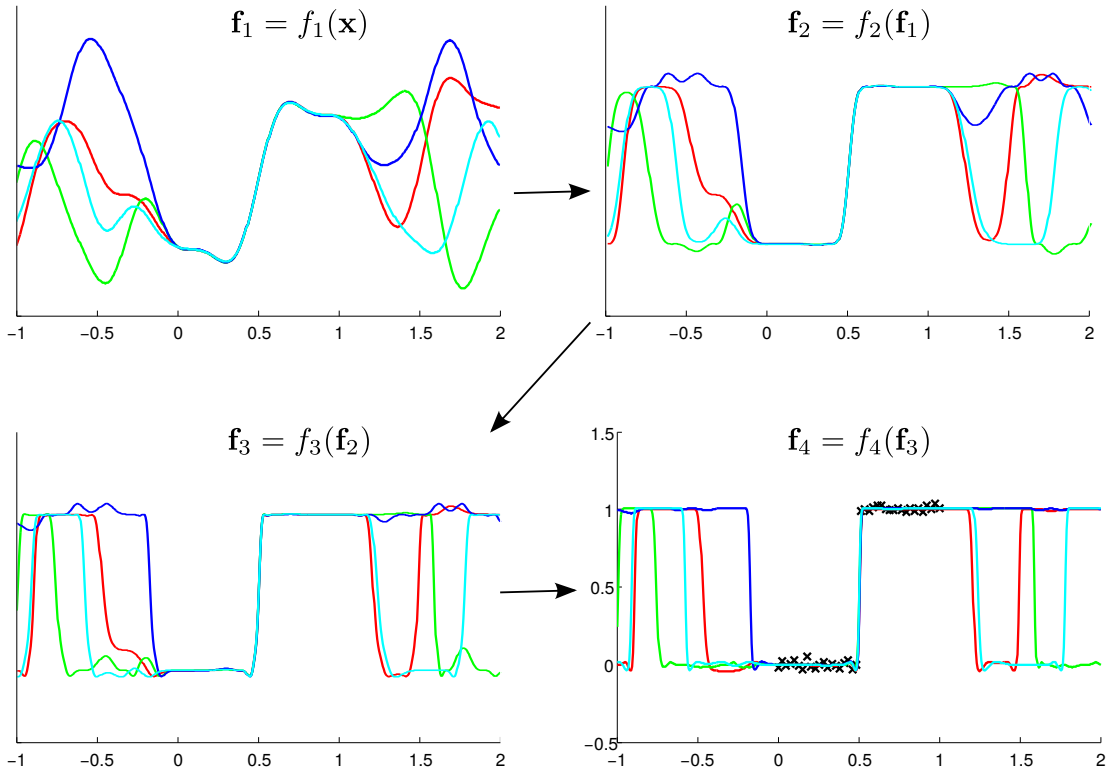
(standard GP)



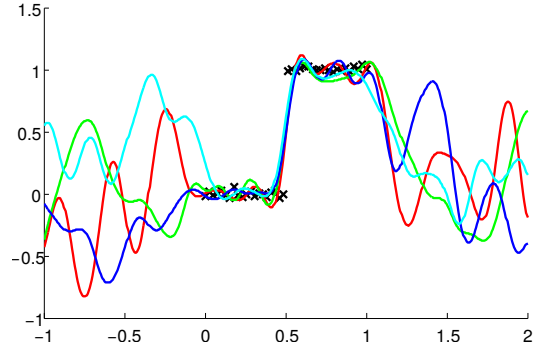
(deep GP - 1 hidden layer)



(deep GP - 3 hidden layers)



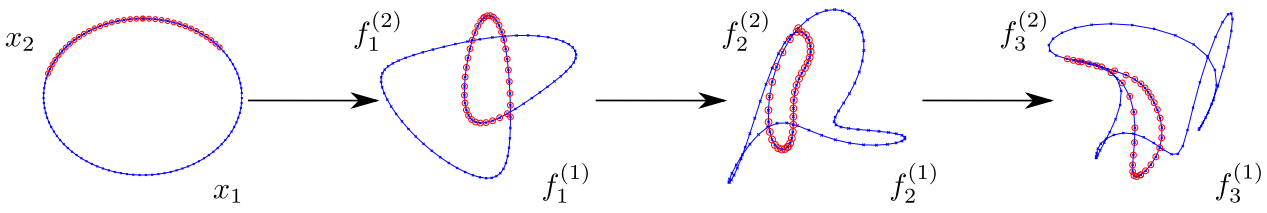
Deep GP with three hidden plus one warping layer



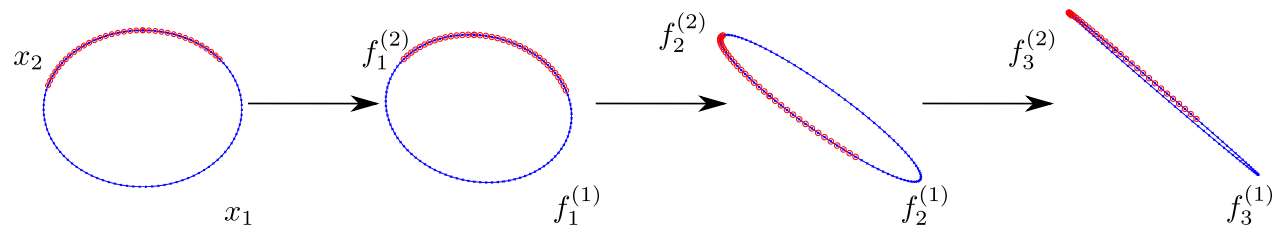
Standard GP

Learning "features"

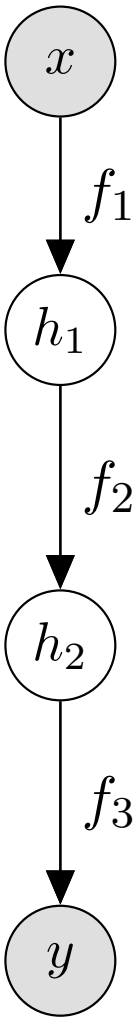
Stacked GP
(nonlinear)



Stacked PCA
(linear)



MAP optimisation?



- Joint Distr. $= p(y|h_2)p(h_2|h_1)p(h_1|x)$
- MAP optimization is extremely problematic because:
  - Dimensionality of hs has to be decided a priori
  - Prone to overfitting, if h are treated as parameters
  - Deep structures are not supported by the model's objective but have to be forced [Lawrence & Moore '07]
- We want:
  - To use the *marginal likelihood* as the objective: marg. lik. $= \int_{h_2,h_1} p(y|h_2)p(h_2|h_1)p(h_1|x)$
  - Further regularization tools.

Marginal likelihood is intractable

Let's try to marginalize out the top layer only:

$$
\begin{aligned}
p(\mathbf{h}_2) &= \int p(\mathbf{h}_2|\mathbf{h}_1)p(\mathbf{h}_1)\mathrm{d}\mathbf{h}_1 \\
&= \int\int p(\mathbf{h}_2|\mathbf{f}_2)p(\mathbf{f}_2|\mathbf{h}_1)p(\mathbf{h}_1)\mathrm{d}\mathbf{f}_2\mathbf{h}_1 \\
&= \int p(\mathbf{h}_2|\mathbf{f}_2)\Big[\underbrace{\int p(\mathbf{f}_2|\mathbf{h}_1)p(\mathbf{h}_1)\mathrm{d}\mathbf{h}_1}_{\text{Intractable!}}\Big]\mathrm{d}\mathbf{f}_2
\end{aligned}
$$

Intractability: $\mathbf{h}_1$ appears non-linearly in $p(\mathbf{f}_2|\mathbf{h}_1)$, inside $\mathbf{K}^{-1}$ (and also the determinant term), where $\mathbf{K} = k(\mathbf{h}_1, \mathbf{h}_1)$.

Solution: Variational Inference

- Similar issues arise for 1-layer models. Solution was given by Titsias and Lawrence, 2010. A small modification to that solution does the trick in deep GPs too.
- Extend Titsias' method for *variational learning of inducing variables in Sparse GPs*.
- Analytic variational bound $\mathcal{F} \leq p(y|x)$
- *Approximately* marginalise out h
- Hence obtain the approximate posterior $q(h)$

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

Additional (structural) regularization tools

Automatic structure discovery (nodes, connections, layers)

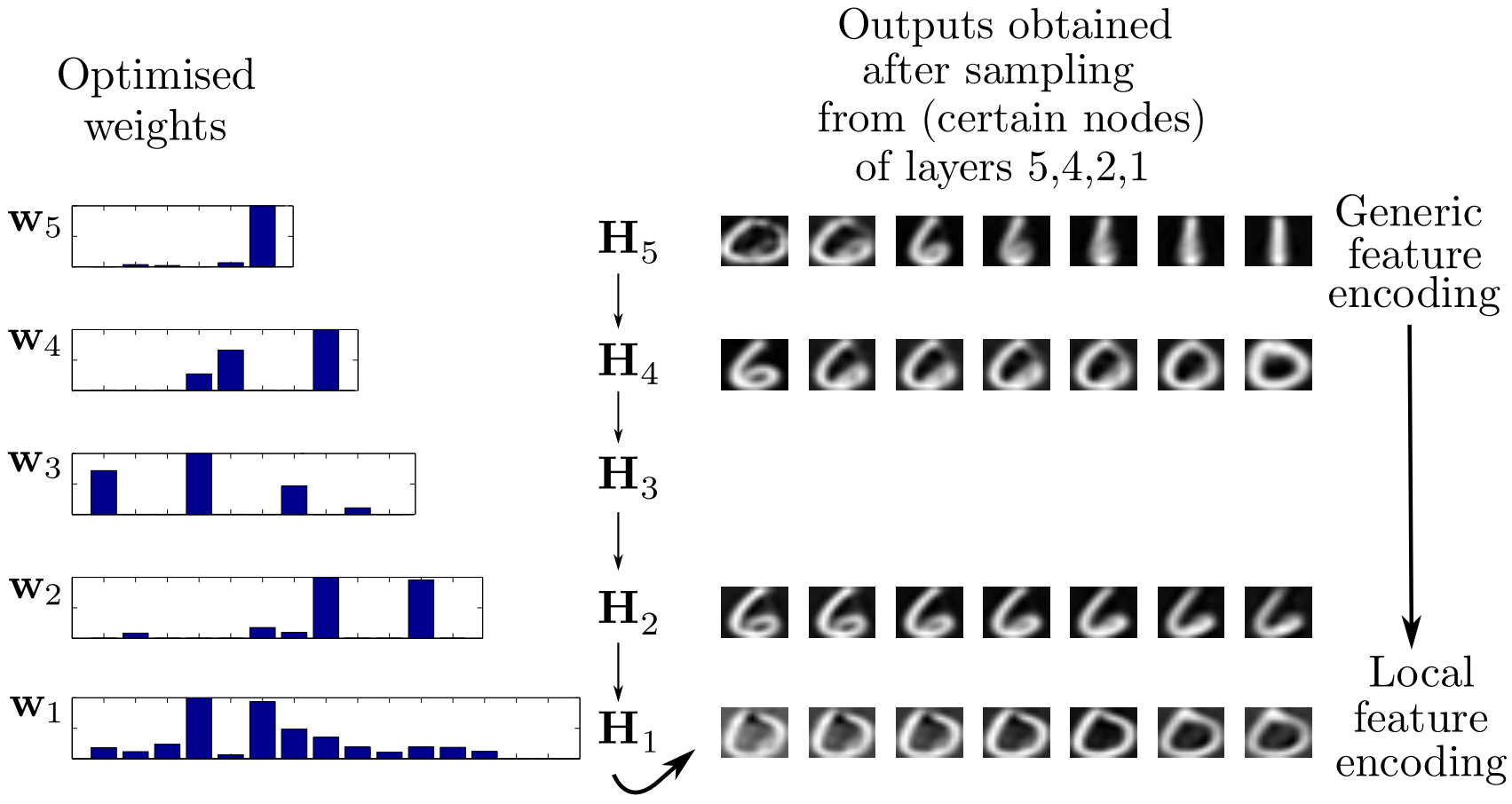
- Automatic Relevance Determination (*prune nodes*)
- Manifold Relevance Determination (enforce conditional independencies, i.e. *prune connections*)

Automatic Relevance Determination (1 layer)

$$k_{ARD}(x, x') = \alpha e^{-\sum_{j=1}^{q} \frac{(x_j - x'_j)^2}{2l_j^2}} = \alpha e^{-\frac{1}{2}\sum_{j=1}^{q} w_j (x_j - x'_j)^2}$$

- The lengthscale l_j along input dimension j tells us how big $|x_j - x'_j|$ has to be for $|f(\mathbf{x}) - f(\mathbf{x}')|$ to be significant.
- So, when $l_j \to \infty$, i.e. $(w_j \to 0)$, then f varies very little as a function of x_j (i.e. dimension j becomes irrelevant).
- By optimising the whole vector $\mathbf{w} = [w_1, w_2, \cdots, w_q]$ we perform automatic selection of the input features.
- In the GP-LVM / deep GP case, the input features (columns of $\mathbf{X}$) correspond to *dimensions*, hence we perform automatic dimensionality detection.

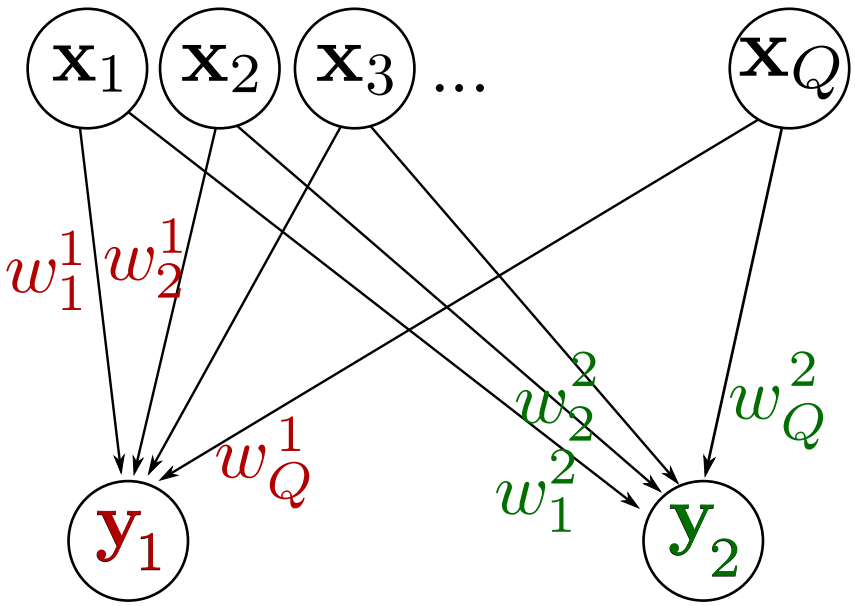
Deep GP: Digits example



Demo: ▸ https://youtu.be/E8-vxt8wxBU

Multi-view modelling (Expand the model "horizontally")

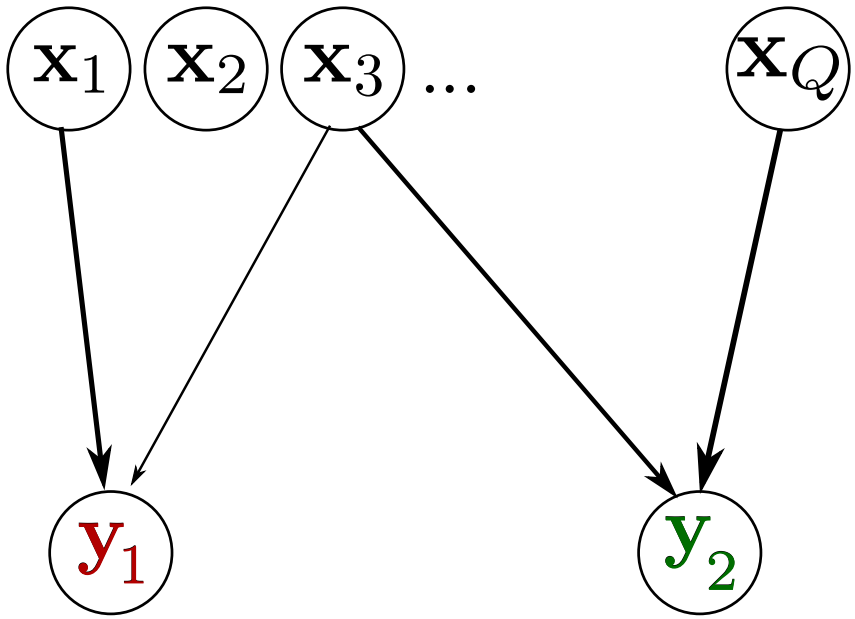
- Multi-view data arise from multiple information sources. These sources naturally contain some overlapping, or *shared* signal (since they describe the same "phenomenon"), but also have some *private* signal.
- Idea: Model such data using overlapping sets of latent variables



Demo: ▸ https://youtu.be/rIPX3CIOhKY

Multi-view modelling (Expand the model "horizontally")

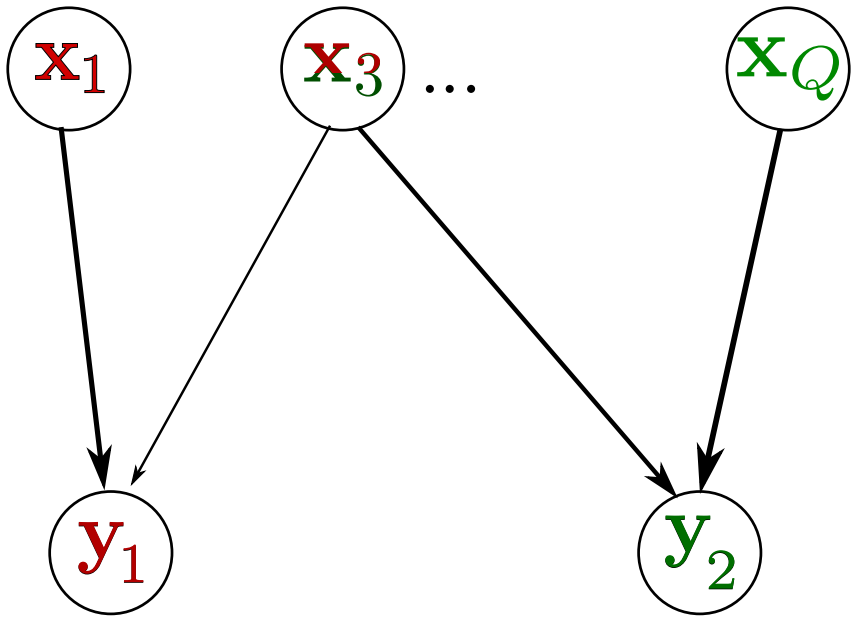
- Multi-view data arise from multiple information sources. These sources naturally contain some overlapping, or *shared* signal (since they describe the same "phenomenon"), but also have some *private* signal.
- Idea: Model such data using overlapping sets of latent variables



Demo: ▸ https://youtu.be/rIPX3CIOhKY

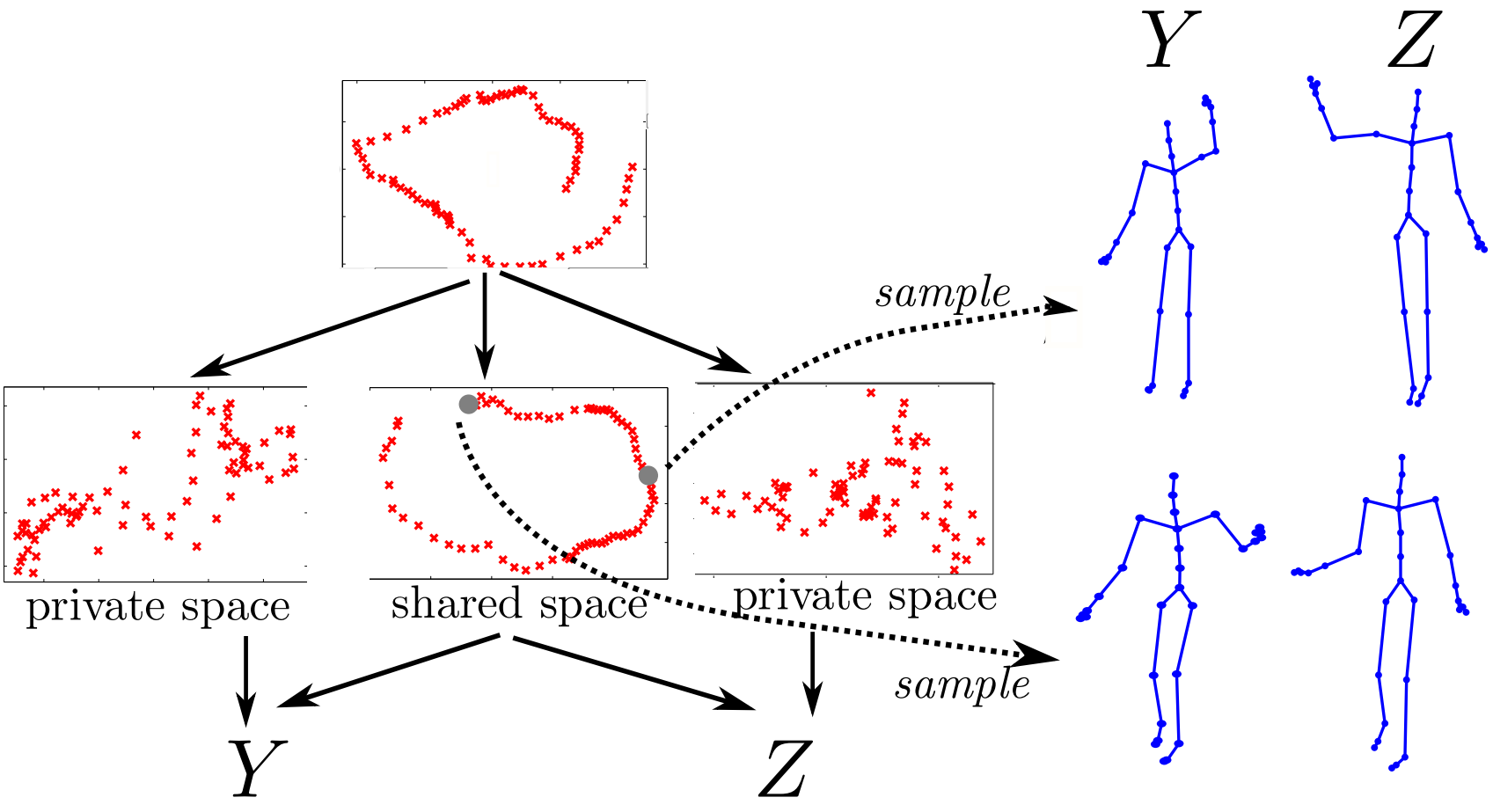
Multi-view modelling (Expand the model "horizontally")

- Multi-view data arise from multiple information sources. These sources naturally contain some overlapping, or *shared* signal (since they describe the same "phenomenon"), but also have some *private* signal.
- Idea: Model such data using overlapping sets of latent variables



Demo: ▸ https://youtu.be/rIPX3CIOhKY

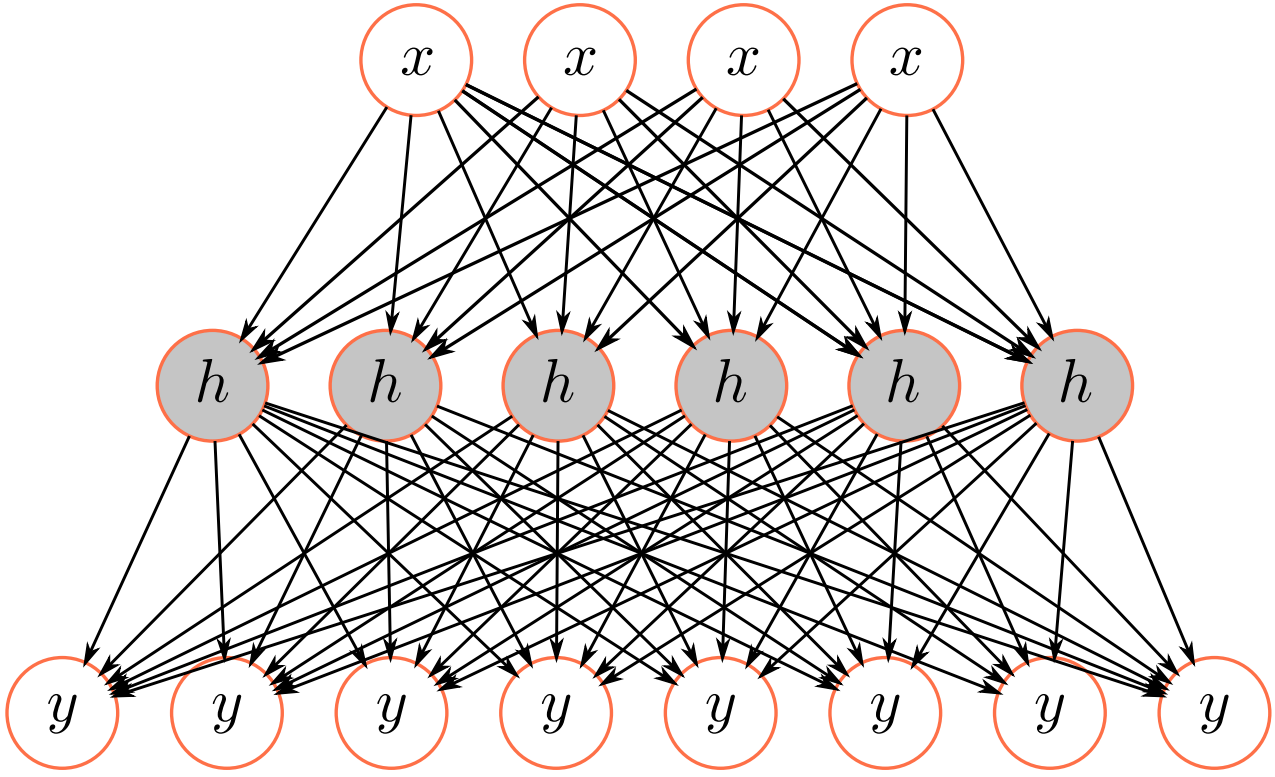
Deep GPs: Another multi-view example



Automatic structure discovery

Tools:

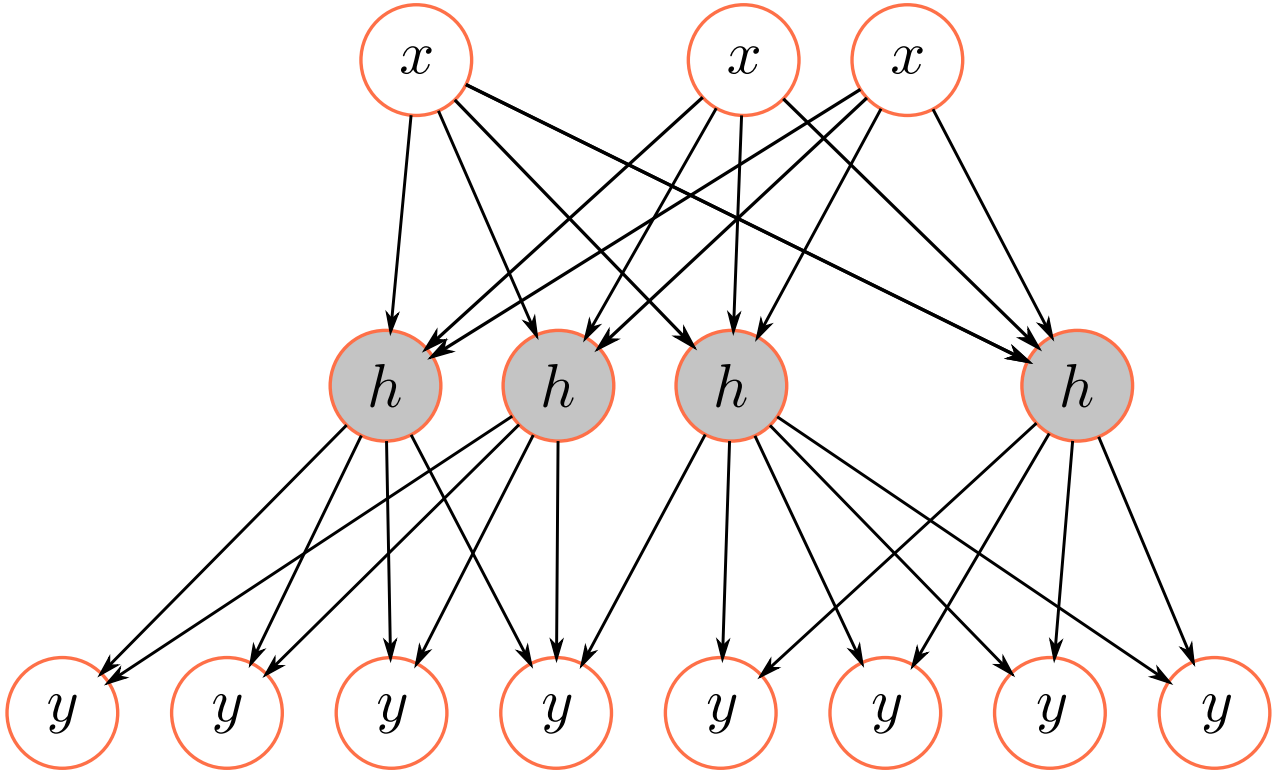
- ARD: Eliminate uncessary nodes/connections
- MRD: Conditional independencies
- Approximating evidence: Number of layers (?)



Automatic structure discovery

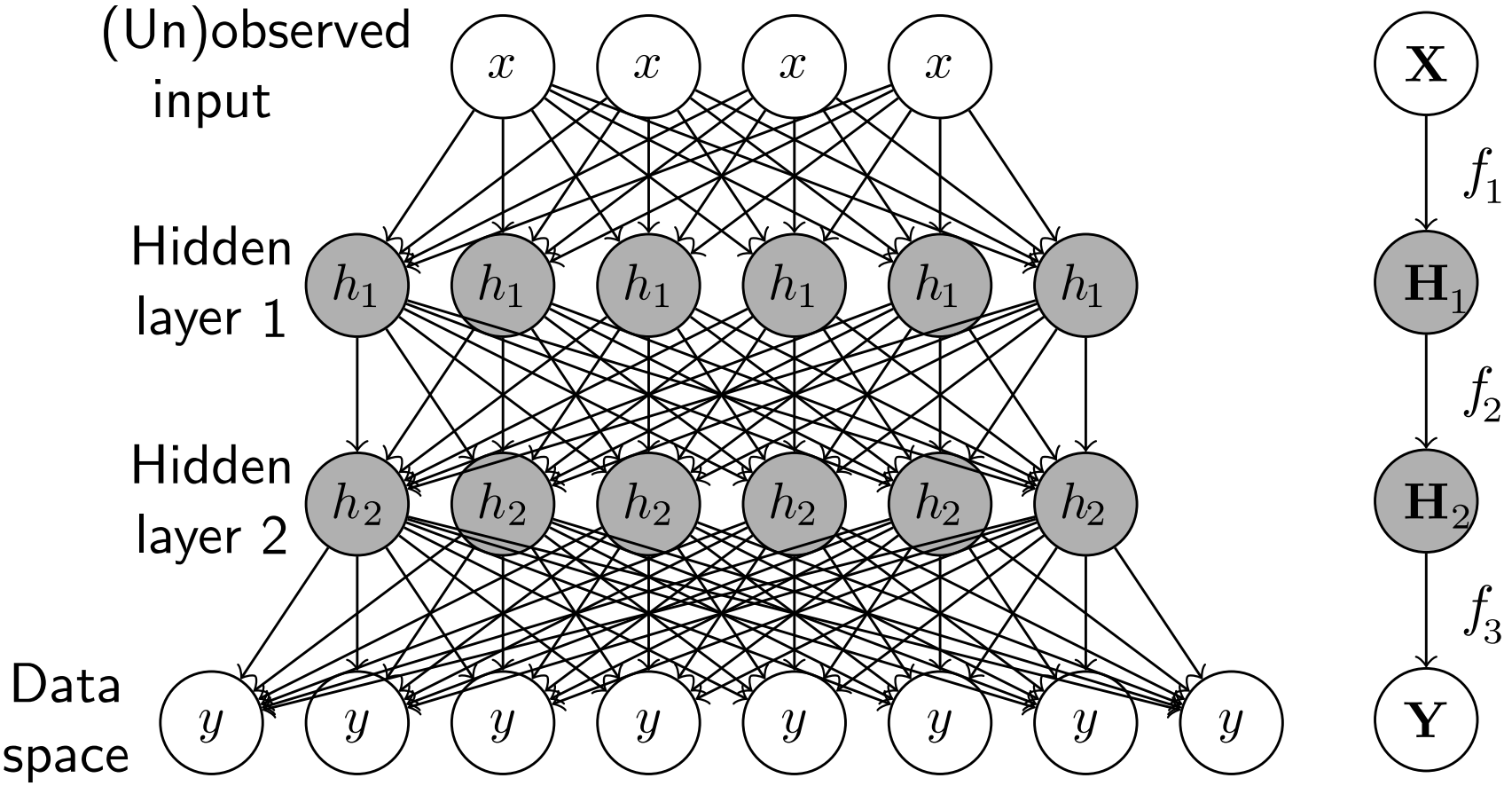
Tools:

- ARD: Eliminate uncessary nodes/connections
- MRD: Conditional independencies
- Approximating evidence: Number of layers (?)



- Example: humanoid robotics
- Other examples: dynamical systems, control (NARX models), forecasting, computational biology.

Recap



$$\mathbf{Y} = f_3(f_2(\cdots f_1(\mathbf{X}))), \qquad \mathbf{H}_i = f_i(\mathbf{H}_{i-1}), \quad f_i \sim \mathcal{GP}$$

Summary

- Bayesian models with GP backbones have the potential to achieve strong regularization.
- I focused on deep GPs as a general family of models.
- A deep GP is *not* a GP, but is more general.
- Supervised / unsupervised / semi-supervised learning supported.
- Analytic computations for training need to be worked out.
- Many variants: multi-view, temporal, autoencoders...
- Future: make it scalable (first results are available)
- Future: how does it compare to / complement more traditional deep models?

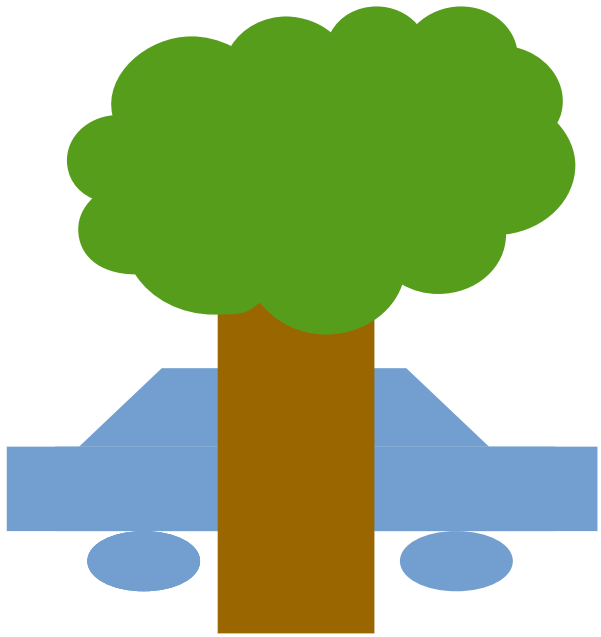
Thanks

Thanks to Neil Lawrence, Michalis Titsias, James Hensman, Carl Henrik Ek, colleagues at Sheffield Robotics.

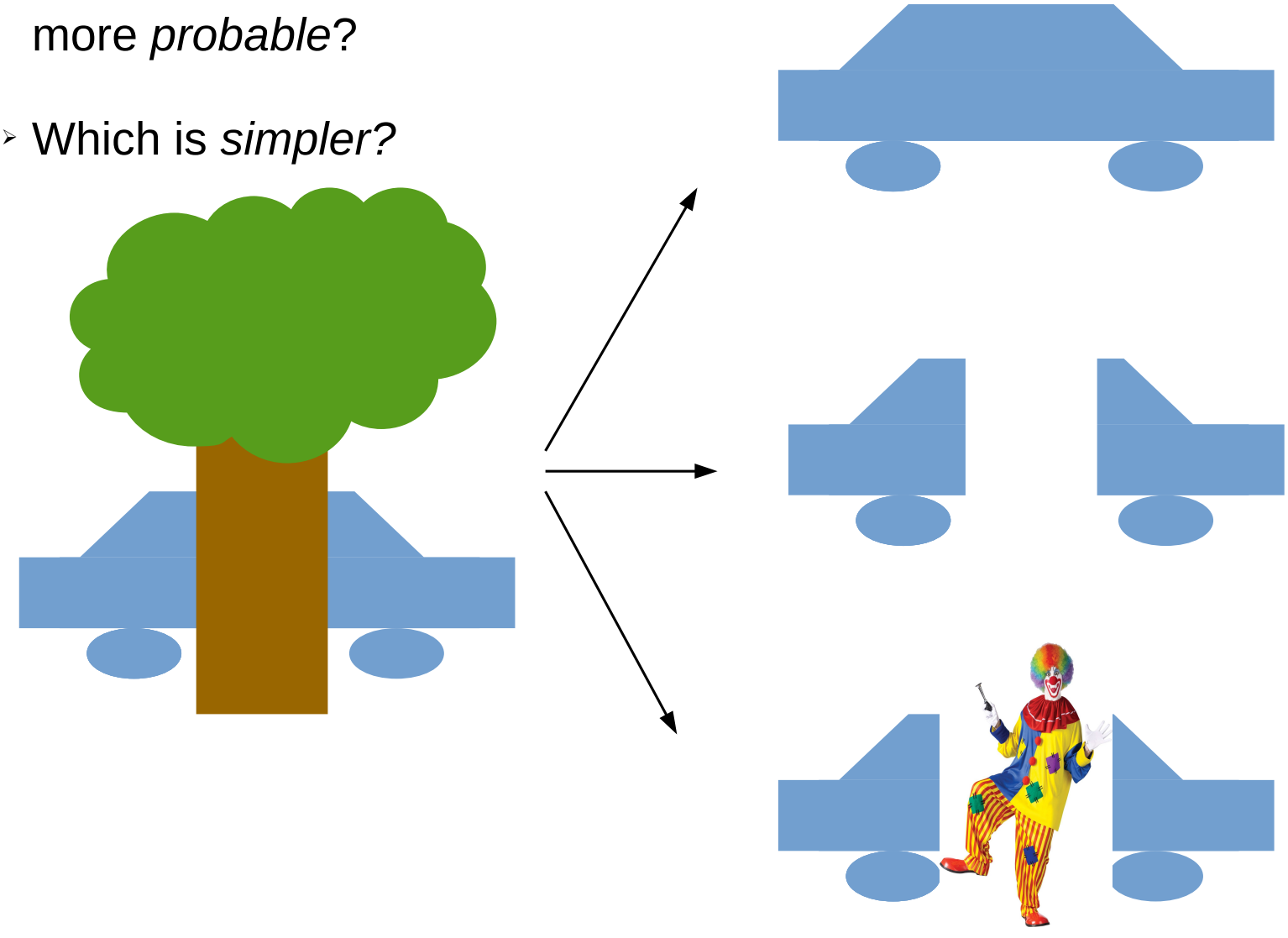
References:

- N. D. Lawrence (2006) "The Gaussian process latent variable model" Technical Report no CS-06-03, The University of Sheffield, Department of Computer Science
- N. D. Lawrence (2006) "Probabilistic dimensional reduction with the Gaussian process latent variable model" (talk)
- C. E. Rasmussen (2008), "Learning with Gaussian Processes", Max Planck Institute for Biological Cybernetics, Published: Feb. 5, 2008 (Videolectures.net)
- Carl Edward Rasmussen and Christopher K. I. Williams. Gaussian Processes for Machine Learning. MIT Press, Cambridge, MA, 2006. ISBN 026218253X.
- M. K. Titsias (2009), "Variational learning of inducing variables in sparse Gaussian processes", AISTATS 2009
- A. C. Damianou, M. K. Titsias and N. D. Lawrence (2011), "Variational Gaussian process dynamical systems", NIPS 2011
- A. C. Damianou, C. H. Ek, M. K. Titsias and N. D. Lawrence (2012), "Manifold Relevance Determination", ICML 2012
- A. C. Damianou and N. D. Lawrence (2013), "Deep Gaussian processes", AISTATS 2013
- J. Hensman (2013), "Gaussian processes for Big Data", UAI 2013

BACKUP SLIDES



- Which of the three inferences is more *probable*?
- Which is *simpler?*



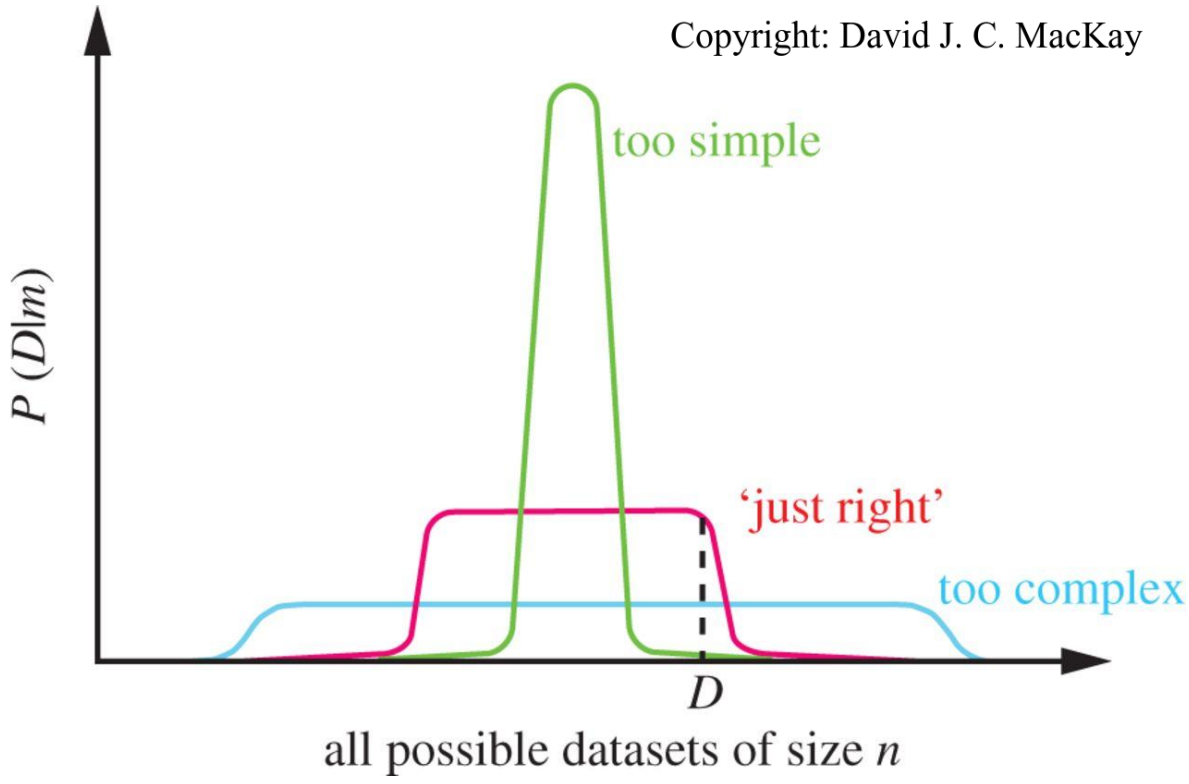
Bayes' rule again

$$p(\theta|D,\mathcal{M}) = \frac{p(D|\theta,\mathcal{M})p(\theta|\mathcal{M})}{p(D|\mathcal{M}) = \int_\theta p(D|\theta,\mathcal{M})p(\theta|\mathcal{M})}$$

(Bayesian) Occam's Razor

"A plurality is not to be posited without necessity". *W. of Ockham*

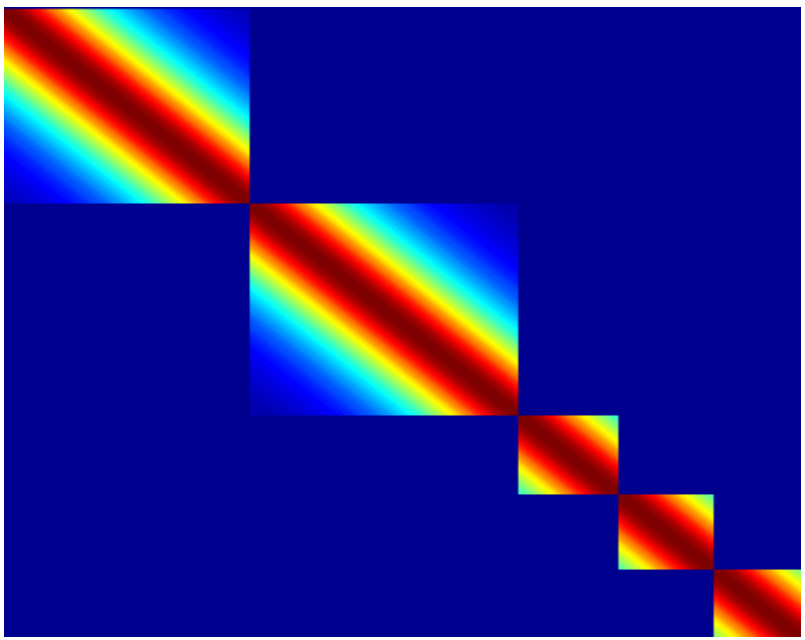
"Everything should be made as simple as possible, but not simpler". *A. Einstein*



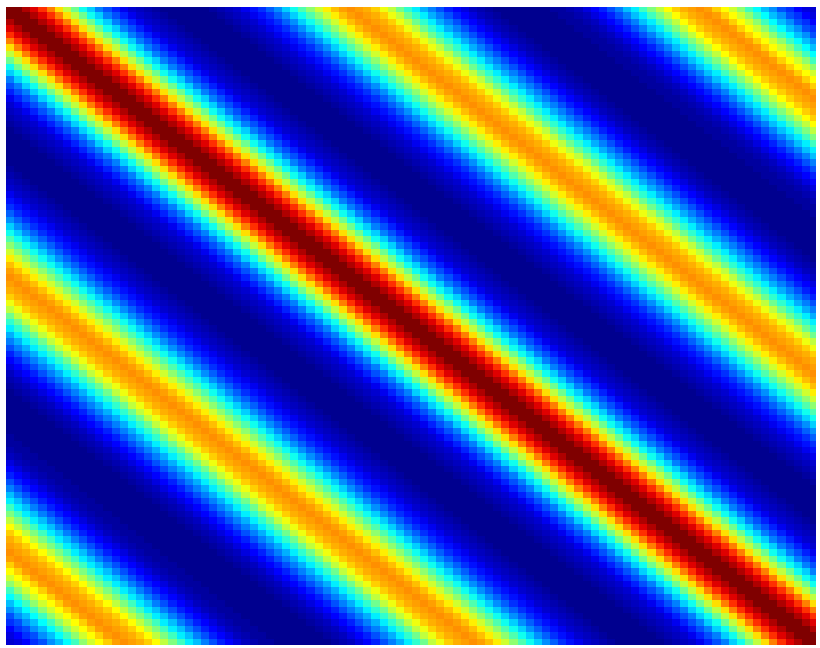
Evidence is higher for the model that is not "unnecessarily complex" but still "explains" the data D.

Dynamics

- Dynamics are encoded in the covariance matrix $\mathbf{K} = k(\mathbf{t}, \mathbf{t})$.
- We can consider special forms for $\mathbf{K}$.



Model individual sequences



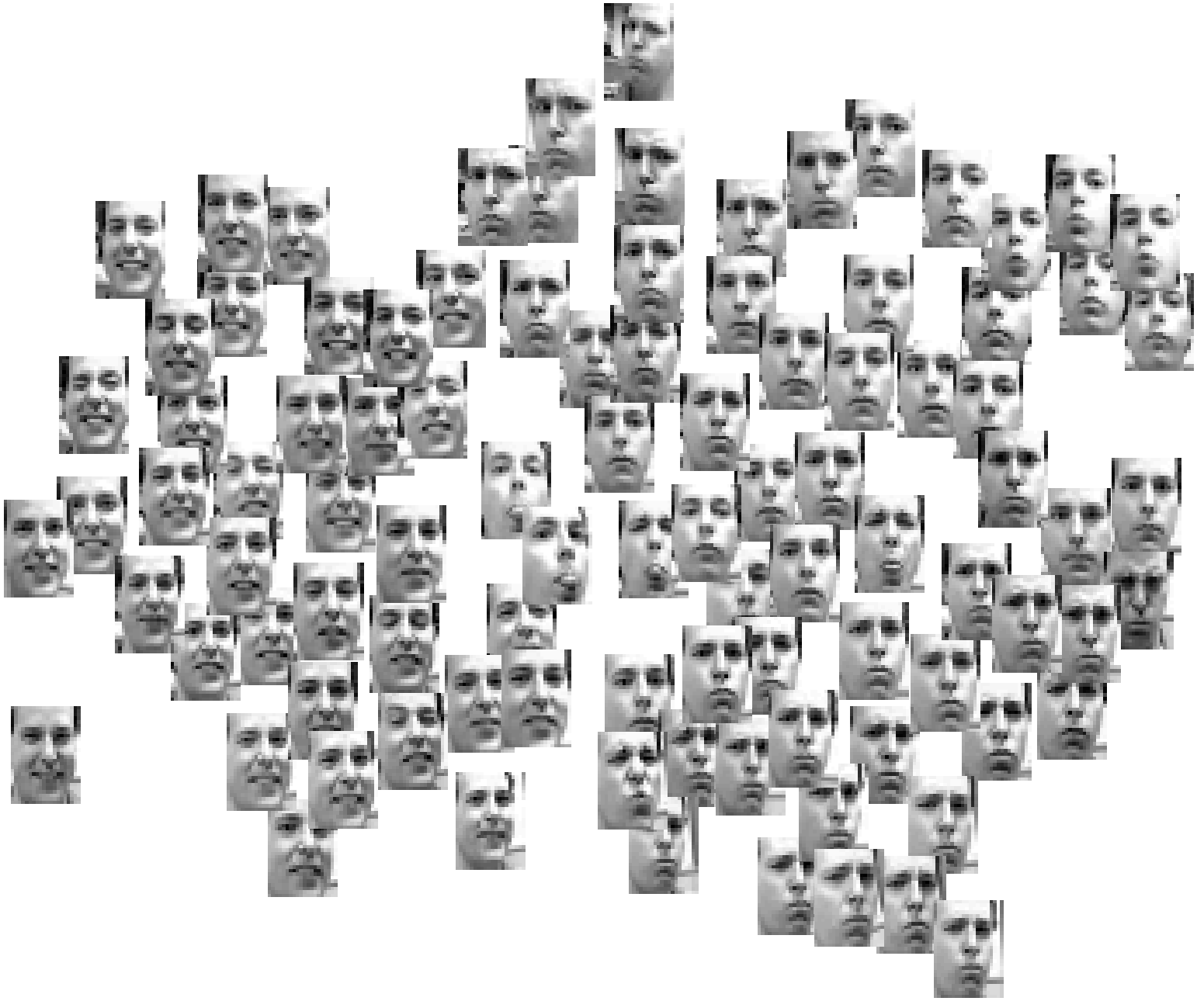
Model periodic data

- ▸ https://www.youtube.com/watch?v=i9TEoYxaBxQ (missa)
- ▸ https://www.youtube.com/watch?v=mUY1XHPnoCU (dog)
- ▸ https://www.youtube.com/watch?v=fHDWloJtgk8 (mocap)

Autoencoder example: Brendan faces

Run demo...

Autoencoder: Brendan faces (credits for idea to J. Hensman)



Dimensionality reduction: Linear vs non-linear

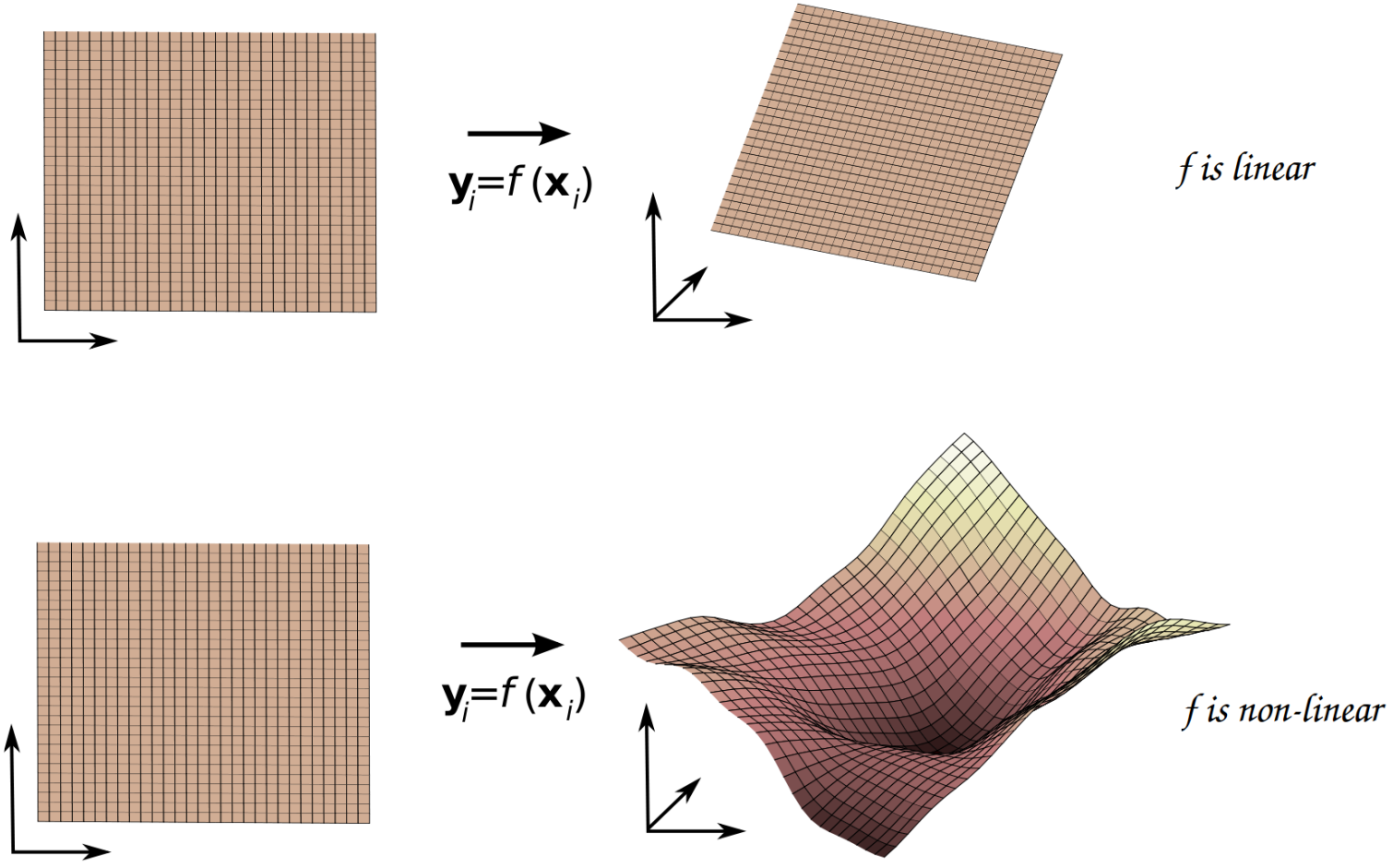


Image from: "Dimensionality Reduction the Probabilistic Way", N. Lawrence, ICML tutorial 2008

Direct marginalisation of h is intractable (O_o)

- New objective: $p(y|x) = \int_{h_2} \left(p(y|h_2) \int_{h_1} p(h_2|h_1) p(h_1|x) \right)$

Direct marginalisation of h is intractable (O_o)

- New objective: $p(y|x) = \int_{h_2} \left(p(y|h_2) \int_{h_1} p(h_2|h_1) p(h_1|x) \right)$
- $p(h_2|x) = \int_{h_1, f_2} p(h_2|f_2) p(f_2|h_1) \; p(h_1|x)$

Direct marginalisation of h is intractable (O_o)

- New objective: $p(y|x) = \int_{h_2} \left(p(y|h_2) \int_{h_1} p(h_2|h_1) p(h_1|x) \right)$
- $p(h_2|x) = \int_{h_1, f_2} p(h_2|f_2) p(f_2|h_1) \; p(h_1|x)$

Direct marginalisation of h is intractable (O_o)

- New objective: $p(y|x) = \int_{h_2} \left(p(y|h_2) \int_{h_1} p(h_2|h_1) p(h_1|x) \right)$
- $p(h_2|x) = \int_{h_1, f_2} p(h_2|f_2) \underbrace{p(f_2|h_1)}_{\text{contains } (k(h_1,h_1))^{-1}} p(h_1|x)$

Direct marginalisation of h is intractable (O_o)

- New objective: $p(y|x) = \int_{h_2} \left(p(y|h_2) \int_{h_1} p(h_2|h_1) p(h_1|x) \right)$
- $p(h_2|x) = \int_{h_1, f_2} p(h_2|f_2) p(f_2|h_1) \; p(h_1|x)$
- $p(h_2|x, \tilde{h}_1) = \int_{h_1, f_2, u_2} p(h_2|f_2) p(f_2|u_2, h_1) p(u_2|\tilde{h}_1) p(h_1|x)$

Direct marginalisation of h is intractable (O_o)

- New objective: $p(y|x) = \int_{h_2} \left(p(y|h_2) \int_{h_1} p(h_2|h_1) p(h_1|x) \right)$
- $p(h_2|x) = \int_{h_1, f_2} p(h_2|f_2) p(f_2|h_1) \; p(h_1|x)$
- $p(h_2|x, \tilde{h}_1) = \int_{h_1, f_2, u_2} p(h_2|f_2) p(f_2|u_2, h_1) p(u_2|\tilde{h}_1) p(h_1|x)$
- $\log p(h_2|x, \tilde{h}_1) \geq \int_{h_1, f_2, u_2} \mathcal{Q} \log \frac{p(h_2|f_2) p(f_2|u_2, h_1) p(u_2|\tilde{h}_1) p(h_1|x)}{\mathcal{Q}}$

Direct marginalisation of h is intractable (O_o)

- New objective: $p(y|x) = \int_{h_2} \left(p(y|h_2) \int_{h_1} p(h_2|h_1) p(h_1|x) \right)$
- $p(h_2|x) = \int_{h_1, f_2} p(h_2|f_1) p(f_1|h_1) \; p(h_1|x)$
- $p(h_2|x, \tilde{h}_1) = \int_{h_1, f_2, u_2} p(h_2|f_2) p(f_2|u_2, h_1) p(u_1|\tilde{h}_1) p(h_1|x)$
- $\log p(h_2|x, \tilde{h}_1) \geq \int_{h_1, f_2, u_2} \mathcal{Q} \log \frac{p(h_2|f_2) \cancel{p(f_2|u_2, h_1)} p(u_2|\tilde{h}_1) p(h_1|x)}{\mathcal{Q} = \cancel{p(f_2|u_2, h_1)} q(u_2) q(h_1)}$

Direct marginalisation of h is intractable (O_o)

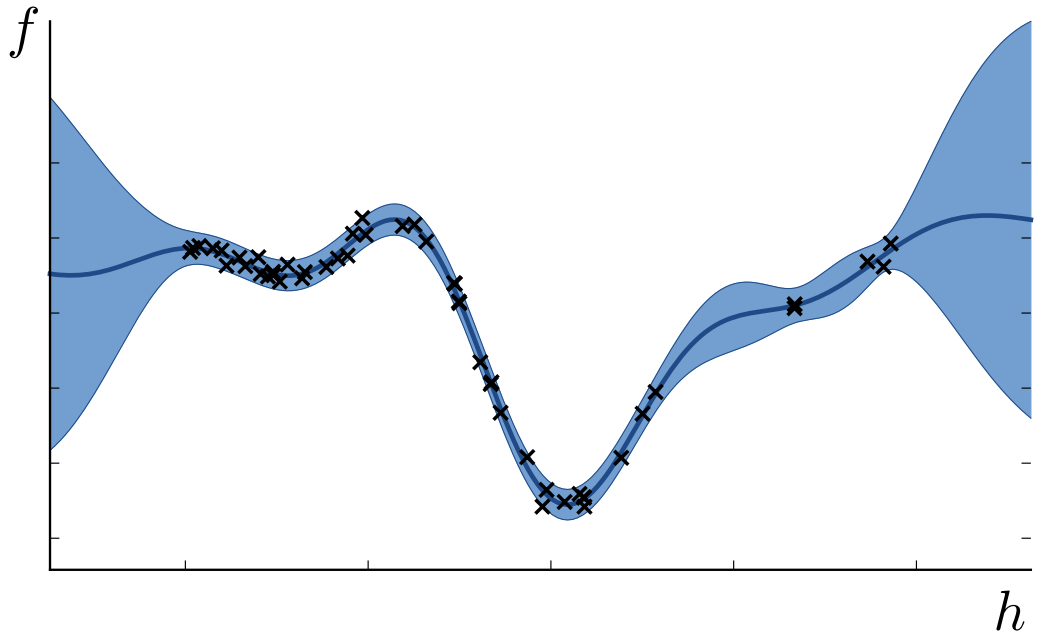
- New objective: $p(y|x) = \int_{h_2} \left(p(y|h_2) \int_{h_1} p(h_2|h_1) p(h_1|x) \right)$
- $p(h_2|x) = \int_{h_1, f_2} p(h_2|f_2) p(f_2|h_1) \; p(h_1|x)$
- $p(h_2|x, \tilde{h}_1) = \int_{h_1, f_2, u_2} p(h_2|f_2) p(f_2|u_2, h_1) p(u_2|\tilde{h}_1) p(h_1|x)$
- $\log p(h_2|x, \tilde{h}_1) \geq \int_{h_1, f_2, u_2} \mathcal{Q} \log \frac{p(h_2|f_2) \cancel{p(f_2|u_2, h_1)} p(u_2|\tilde{h}_1) p(h_1|x)}{\mathcal{Q} = \cancel{p(f_2|u_2, h_1)} q(u_2) q(h_1)}$
- $\log p(h_2|x, \tilde{h}_1) \geq \int_{h_1, f_2, u_2} \mathcal{Q} \log \frac{p(h_2|f_2) p(u_2|\tilde{h}_1) p(h_1|x)}{\mathcal{Q} = q(u_2) q(h_1)}$

$p(u_2|\tilde{h}_1)$ contains $k(\tilde{h}_1, \tilde{h}_1)^{-1}$

The above trick is applied to all layers simultaneously.

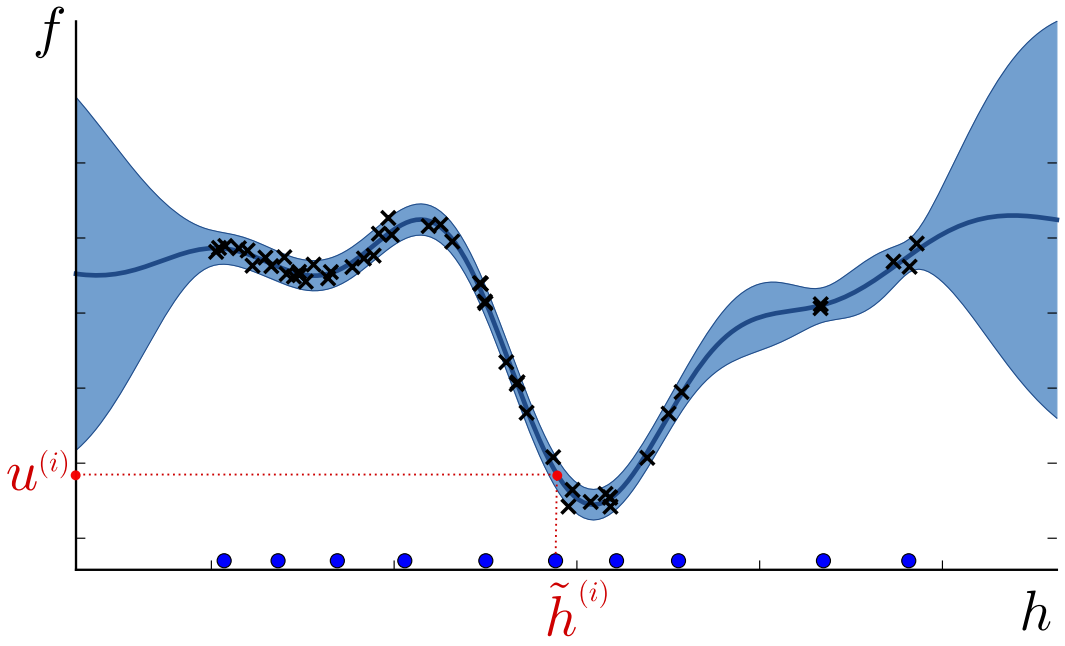
Inducing points: sparseness, tractability and Big Data

$h^{(1)}$	$\mathbf{f}^{(1)}$
$h^{(2)}$	$\mathbf{f}^{(2)}$
$\dots$	$\dots$
$h^{(30)}$	$\mathbf{f}^{(30)}$
$h^{(31)}$	$\mathbf{f}^{(31)}$
$\dots$	$\dots$
$h^{(N)}$	$\mathbf{f}^{(N)}$



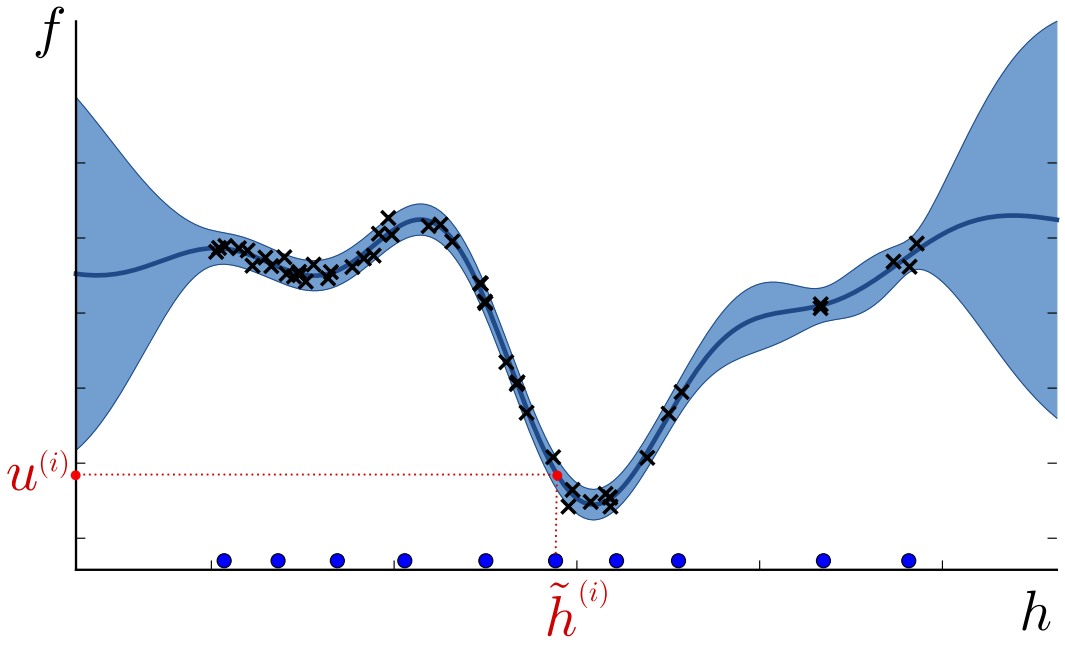
Inducing points: sparseness, tractability and Big Data

$h^{(1)}$	$\mathbf{f}^{(1)}$
$h^{(2)}$	$\mathbf{f}^{(2)}$
$\dots$	$\dots$
$h^{(30)}$	$\mathbf{f}^{(30)}$
$\tilde{h}^{(i)}$	$u^{(i)}$
$h^{(31)}$	$\mathbf{f}^{(31)}$
$\dots$	$\dots$
$h^{(N)}$	$\mathbf{f}^{(N)}$



Inducing points: sparseness, tractability and Big Data

$h^{(1)}$	$\mathbf{f}^{(1)}$
$h^{(2)}$	$\mathbf{f}^{(2)}$
$\dots$	$\dots$
$h^{(30)}$	$\mathbf{f}^{(30)}$
$\tilde{h}^{(i)}$	$u^{(i)}$
$h^{(31)}$	$\mathbf{f}^{(31)}$
$\dots$	$\dots$
$h^{(N)}$	$\mathbf{f}^{(N)}$



- Inducing points originally introduced for faster **(sparse) GPs**
- But this also induces **tractability** in our models, due to the conditional independencies assumed
- Viewing them as **global variables**
  $\Rightarrow$ extension to **Big Data** [Hensman et al., UAI 2013]

Factorised vs non-factorised bound

- Preliminary bound

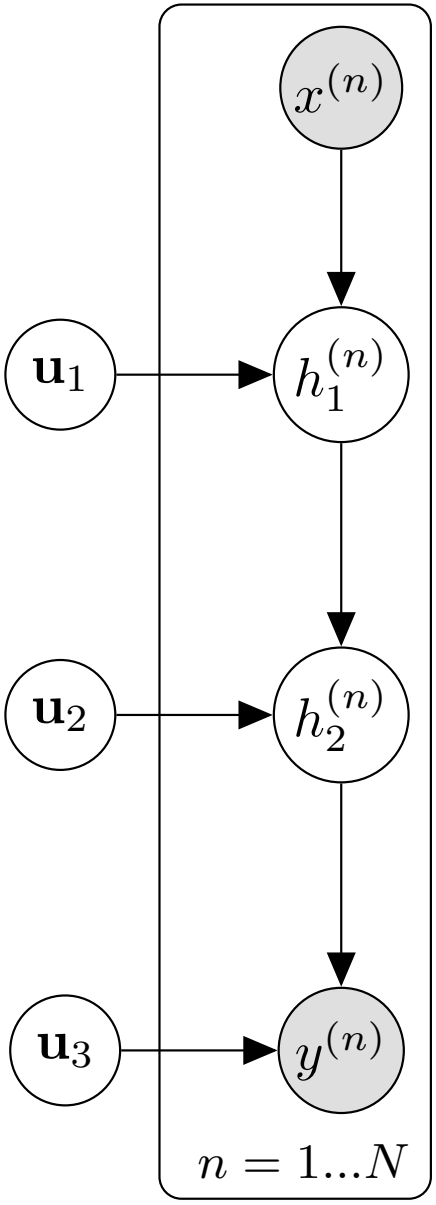
$$\mathcal{L} \leq \log p(\mathbf{Y}, \{\mathbf{H}_l\}_{l=1}^{L} | \{\mathbf{U}_l\}_{l=1}^{L+1}, \mathbf{X})$$

$$\mathcal{L} = \sum_{n=1}^{N} \left[\sum_{l=1}^{L} \left(\sum_{q=1}^{Q_l} \log \mathcal{N}\left(h_l^{(n,q)} | \mathbf{k}_l^{(n,:)} \mathbf{K}^{-1} \mathbf{u}_l^{(:,d)}, \beta_l^{-1} \mathbf{I} \right) - \frac{\beta_l^{-1} \tilde{\mathbf{k}}_l^{(n)}}{2} \right) \right]$$

$$= \sum_{n=1}^{N} \sum_{l=1}^{L} \sum_{q=1}^{Q_l} \mathcal{L}_l^{n,q}$$

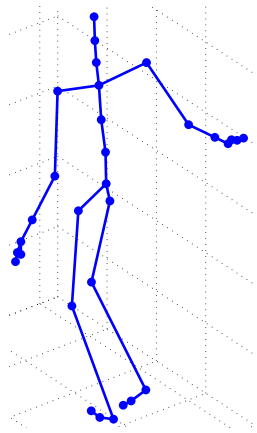
- Fully factorised.

SVI for factorised deep GPs

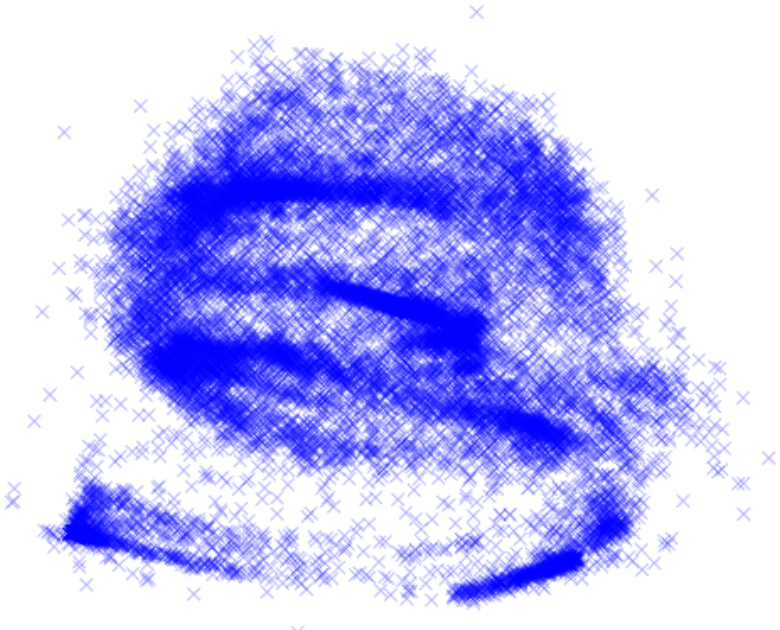


- We can additionally marginalise out $\mathbf{h}$ and maintain factorisation.
- We can consider SVI.
- Unlike $\boldsymbol{\theta}_u$ and $\boldsymbol{\theta}$, $\mathbf{h}$ are *not* global variables.
- So, estimate $\mathbf{h}^{(batch)}$ given the current $\boldsymbol{\theta}_t$
- Adjusting the step-length for SVI is tricky.

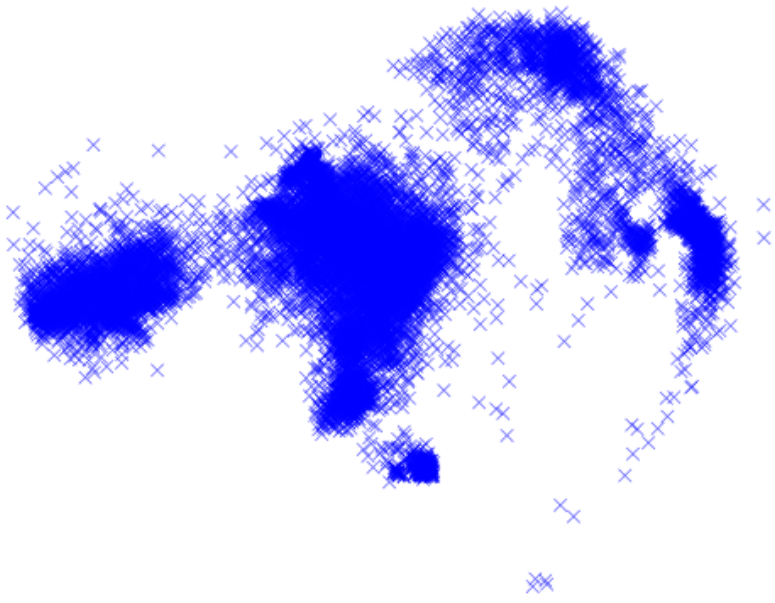
SVI - 18K mocap examples



Hidden space projections:

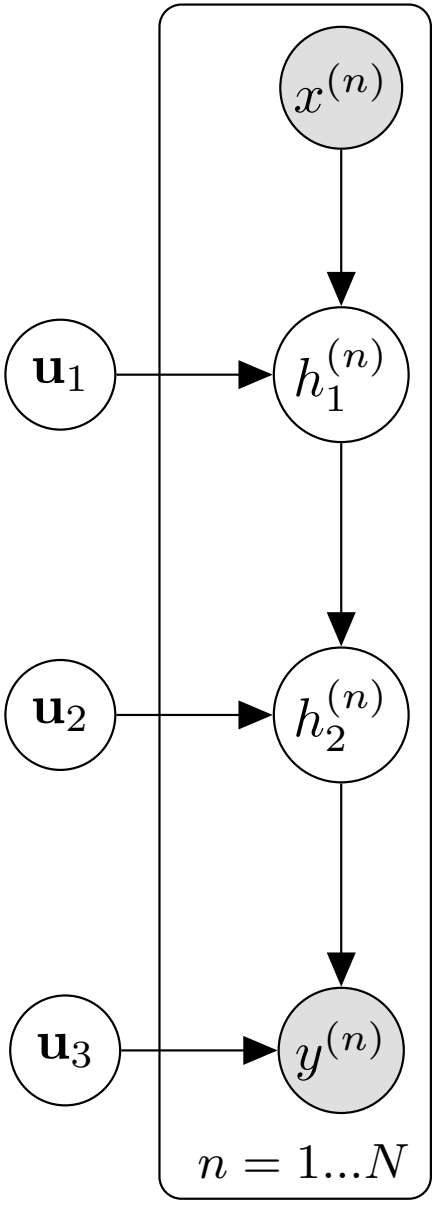


Global motion features

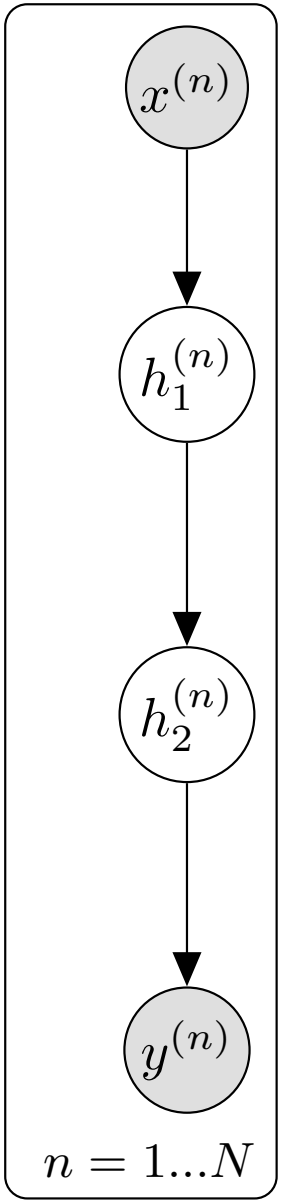


Clustered motion features

Integrate out $\mathbf{u}$



Integrate out $\mathbf{u}$



- Integrating out $\mathbf{u} \rightarrow$ factorisation is maintained.
- "Effect" of $\mathbf{u}$ manifested through $q(\mathbf{u})$

"Collapse" $q(\mathbf{u})$

- Collapsing $\mathbf{u}$'s distribution eliminates many variational parameters
- But this introduces coupling and breaks the factorisation
- But we can still distribute the computations efficiently (work by Y. Gal, Z. Dai)