



UNIVERSITY
OF AMSTERDAM

Machine Learning for Physics and Astronomy

Juan Rojo

VU Amsterdam & Theory group, Nikhef

Natuur- en Sterrenkunde BSc (Joint Degree), Honours Track
Lecture 7, 12/10/2020

Today's lecture

- 📍 Kernel methods and the Dual Representation
- 📍 Gaussian Processes
- 📍 Variational Inference
- 📍 Guest lecture by **Dr. Tristan Bereau** on applications of machine learning in condensed matter

Kernel Methods

Kernel Methods

most of the ML models we have considered for regression and classification are based on the **parametric mapping** (linear or non-linear) between inputs and outputs

$$y(x, \theta) \leftarrow \text{info on training samples entirely contained in model params}$$

were the **training examples are discarded** after the model parameters (or their posterior distribution) have been determined

here we introduce models where the training dataset is also used in the **prediction phase**

recall discussion of **Support Vector Machines**: we introduced a **feature space mapping**

Example of non-linear feature-space transformation

x

$\phi(x)$

Height

Height/Weight

Weight

Weight+Height

Age

*Age * Weight*

Kernel Methods

most of the ML models we have considered for regression and classification are based on the **parametric mapping** (linear or non-linear) between inputs and outputs

$$y(x, \theta) \leftarrow \text{info on training samples entirely contained in model params}$$

were the **training examples are discarded** after the model parameters (or their posterior distribution) have been determined

here we introduce models where the training dataset is also used in the **prediction phase**

the dimensions of the **feature space mapping** are in general different from those of input space

x

Height

Weight

Age

$\phi(x)$

Height/Weight

Weight + Height

*Age * Weight*

*Height * Weight+Age*

Kernel Methods

most of the ML models we have considered for regression and classification are based on the **parametric mapping** (linear or non-linear) between inputs and outputs

$$y(x, \theta) \leftarrow \text{info on training samples entirely contained in model params}$$

were the **training examples are discarded** after the model parameters (or their posterior distribution) have been determined

here we introduce models where the training dataset is also used in the **prediction phase**

the key ingredient of **kernel methods** is the **kernel function** evaluated at the training points

$$k(x, x') = \phi(x)^T \phi(x')$$

inner product in feature space

kernel function **feature space mapping**

the simplest kernel is the **linear kernel** (identity mapping in feature space)

$$k(x, x') = x^T x'$$

Dual representations

many ML models for regression and classification can be reformulated in a **dual representation** where the kernel functions arise naturally

consider a linear regression model with a regularised least-squares error function

$$E_{\text{tr}}(\boldsymbol{\theta}) = \frac{1}{2} \sum_{n=1}^N (\boldsymbol{\theta}^T \phi(\mathbf{x}_n) - t_n)^2 + \frac{\lambda}{2} \boldsymbol{\theta}^T \boldsymbol{\theta}$$

*inner product
in feature space*

regulator

*output from
training examples*

the model parameters are determined **analytically** by requiring the vanishing of the gradient

$$\boldsymbol{\theta} = \sum_{n=1}^N a_n \phi(\mathbf{x}_n), \quad a_n = -\frac{1}{\lambda} (\boldsymbol{\theta}^T \phi(\mathbf{x}_n) - t_n)$$

recall that inner products take place in feature space!

one can formulate a dual representation of the problem in terms of the parameter vector

$$\mathbf{a} = (a_1, a_2, \dots, a_N)^T \quad \mathbf{t} = (t_1, t_2, \dots, t_N)^T$$

Dual representations

with some algebra one can show that the original figure of merit of the model

$$E_{\text{tr}}(\boldsymbol{\theta}) = \frac{1}{2} \sum_{n=1}^N (\boldsymbol{\theta}^T \boldsymbol{\phi}(x_n) - t_n)^2 + \frac{\lambda}{2} \boldsymbol{\theta}^T \boldsymbol{\theta}$$

admits a **dual representation** in terms of **kernel function**

$$E_{\text{tr}}(\boldsymbol{a}) = \frac{1}{2} \boldsymbol{a}^T \mathbf{K} \boldsymbol{K} \boldsymbol{a} - \boldsymbol{a}^T \mathbf{K} \boldsymbol{t} + \frac{1}{2} \boldsymbol{t}^T \boldsymbol{t} + \frac{\lambda}{2} \boldsymbol{a}^T \mathbf{K} \boldsymbol{a}$$

*model params do
not appear
in this formulation*

where \mathbf{K} is known as the Gram matrix, an $N \times N$ symmetric matrix with entries

$$K_{nm} = k(x_n, x_m) = \boldsymbol{\phi}(x_n)^T \boldsymbol{\phi}(x_m)$$

*matrix in space of
input data*

given by the kernel function evaluated over two of the **input training examples**

the crucial property of this dual representation is that now the predictions for new inputs will explicitly **depend on the training examples**

Dual representations

the crucial property of this dual representation is that now the predictions for new inputs will explicitly **depend on the training examples**

$$y(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$$

output of trained model ↗
 new input

$$\mathbf{k}(\mathbf{x}) = (k(\mathbf{x}_1, \mathbf{x}), k(\mathbf{x}_2, \mathbf{x}), \dots)$$

in the dual formulation we invert a $N \times N$ matrix (data space) rather than a $M \times M$ one (feature space), so this does not appear to be advantageous ...

the main benefit is being able to **work directly with kernel functions** and bypass a choice of feature map

this allows one to work in feature spaces of **very high (even infinite) dimensionality**

Choosing kernel functions

recall that the kernel function is constructed from a **feature space mapping**

$$k(\mathbf{x}_n, \mathbf{x}_m) = \boldsymbol{\phi}(\mathbf{x}_n)^T \boldsymbol{\phi}(\mathbf{x}_m) = \sum_{i=1}^M \phi_i(\mathbf{x}_n) \phi_i(\mathbf{x}_m)$$

one can also construct kernel functions directly, provided they can be expressed as above

e.g., for a 2D input space $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2 = (x_1 z_1 + x_2 z_2)^2 = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{z})$

$$\boldsymbol{\phi}(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

note that feature space dimension ($M=3$) is bigger than input space dimensionality ($d=2$)!

there exist more general methods to construct acceptable kernels e.g. using **basis functions**

a popular choice is Gaussian kernel, built from an **infinite-dimensional feature map**

$$k(\mathbf{x}, \mathbf{z}) = \exp\left(-||\mathbf{x} - \mathbf{z}||^2 / 2\sigma^2\right)$$

Choosing kernel functions

a popular choice is Gaussian kernel, built from an **infinite-dimensional feature map**

$$k(x, z) = \exp\left(-||x - z||^2/2\sigma^2\right)$$

we can demonstrate that this is a valid kernel as follows: given two valid kernels

$$k_1(x, z), \quad k_2(x, z)$$

we can construct new valid kernels from them using the following rules

$$k(x, z) = f(x)k_1(x, z)f(z) \quad k(x, z) = \exp(k(x, z))$$

so now we can check that the Gaussian kernel is indeed a *bona-fide* kernel by expanding

$$k(x, z) = \exp(-x^T x/2\sigma^2) \exp(-x^T z/2\sigma^2) \exp(-z^T z/2\sigma^2)$$

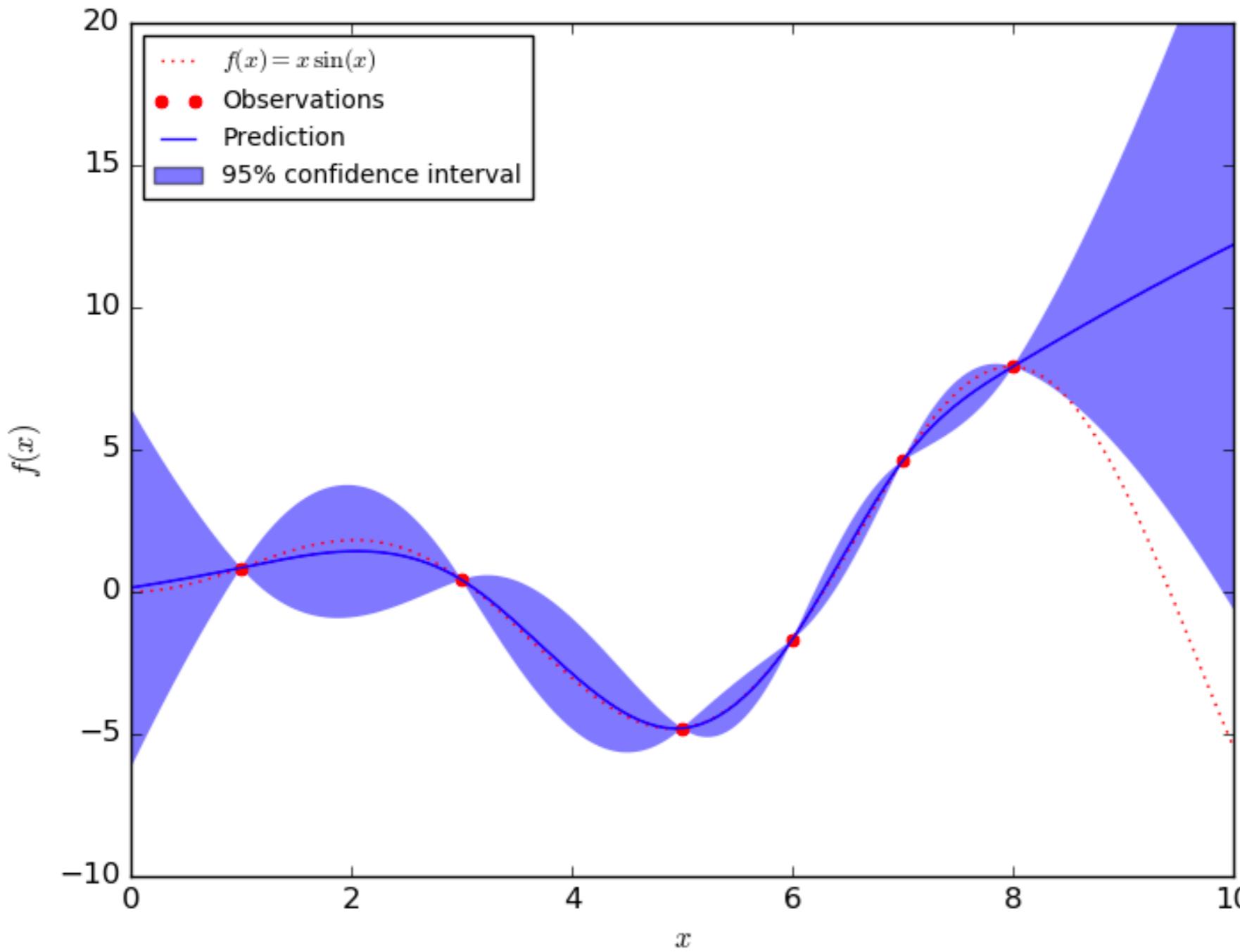
linear kernel

the feature map associated to this kernel has infinite dimensionality!

Gaussian Processes

Gaussian processes

in Gaussian processes we bypass the need of a **parametric classifier/regressor** and work directly in the **space of functions for the (prior) probability distributions**



in regression models one would introduce a parametrisation and infer the model params from

instead we can to avoid parameterisations and express the problem directly at the level of the space of functions/models

Gaussian processes

the previous discussion was limited to **non-probabilistic models**. We now extend the kernel methods to **probabilistic discrimination processes**

in Gaussian processes we bypass the need of a **parametric classifier/regressor** and work directly in the **space of functions for the (prior) probability distributions**

we can illustrate the philosophy of the method by revisiting the simple linear regression example

$$y(\mathbf{x}) = \boldsymbol{\theta}^T \phi(\mathbf{x}) = \sum_{i=1}^M \theta_i \phi_i(\mathbf{x})$$

predictor *model parameters* *feature space map*

assume that the model parameters have as **prior distribution** an isotropic Gaussian

$$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta} | \mathbf{0}, \alpha^{-1} \mathbf{I})$$

for a given choice of the model parameters, we will have a different functional form of the predictor

the probability density for $\boldsymbol{\theta}$ induces a probability density in the space of functions y !

Gaussian processes

consider that we are given a set of training examples and we evaluate the model predictions

$$\mathbf{x}_1, \dots, \mathbf{x}_N, \quad \mathbf{y} \equiv (y(\mathbf{x}_1), \dots, y(\mathbf{x}_N))$$

note this is not the label of the training examples!

from the definition of the linear model we have

$$\mathbf{y} = \Phi\boldsymbol{\theta} = \sum_{i=1}^M \phi_i(\mathbf{x}_n)\theta_i$$

and we can now construct the **probability density in the space of functions \mathbf{y}** by noting that it is Gaussian itself (since $\boldsymbol{\theta}$ is also Gaussian) with mean and covariance

$$\mathbb{E}[\mathbf{y}] = \Phi\mathbb{E}[\boldsymbol{\theta}] = 0$$

$$\text{cov}[\mathbf{y}] = \mathbb{E}[\mathbf{y}\mathbf{y}^T] = \Phi\mathbb{E}[\boldsymbol{\theta}\boldsymbol{\theta}^T]\Phi^T = \frac{1}{\alpha}\Phi\Phi^T = K$$

the kernel function of the Gaussian process



$$K_{nm} = \frac{1}{\alpha}\boldsymbol{\phi}(\mathbf{x}_n)^T\boldsymbol{\phi}(\mathbf{x}_m)$$

Gaussian process are another application of kernel modes

Gaussian processes

this model provides a specific example of a Gaussian process, whose **main features** are:

- Defined as **probability distribution in the space of functions**
- This distribution is defined by **second-order statistics**: we only need mean and covariance
- Its covariance is given by the **kernel function**
- A gaussian process depends on a number of **hyperparameters** that should be inferred from data

Gaussian kernel

$$k(x, z) = \exp\left(-||x - z||^2/2\alpha^2\right)$$

Exponential kernel

$$k(x, z) = \exp(-\alpha ||x - z||)$$

these hyperparameters can be inferred by **maximising the log-likelihood** comparing the model predictions with the output labels of the training data

e.g. for Gaussian process regression

$$\log\text{-likelihood} \longrightarrow \ln p(t | \alpha) = -\frac{1}{2} \ln |C_N| - \frac{1}{2} t^T C_N^{-1} t$$

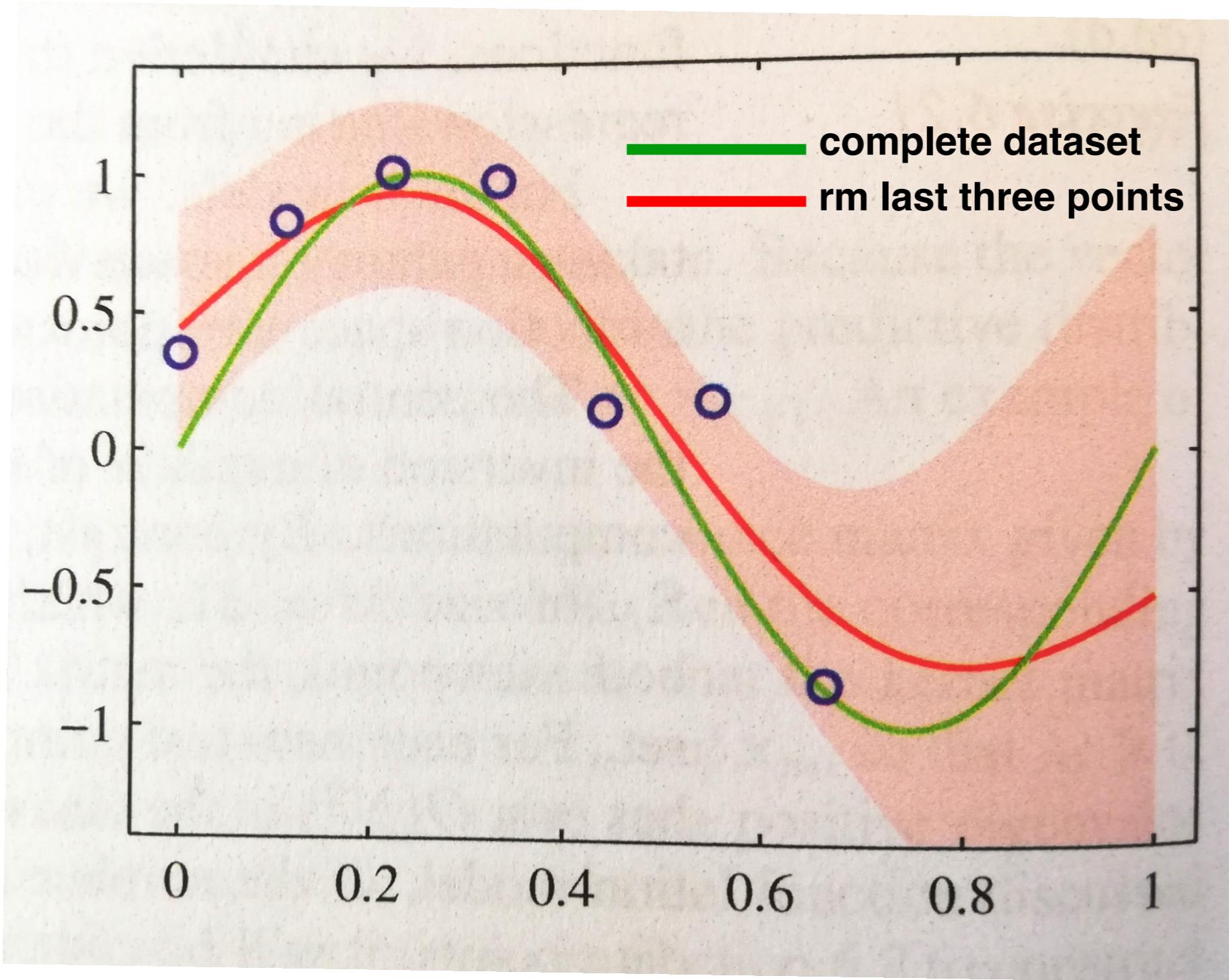
stochastic noise

$$C_{nm} = k(x_n, x_m) + \beta^{-1} \delta_{nm}$$

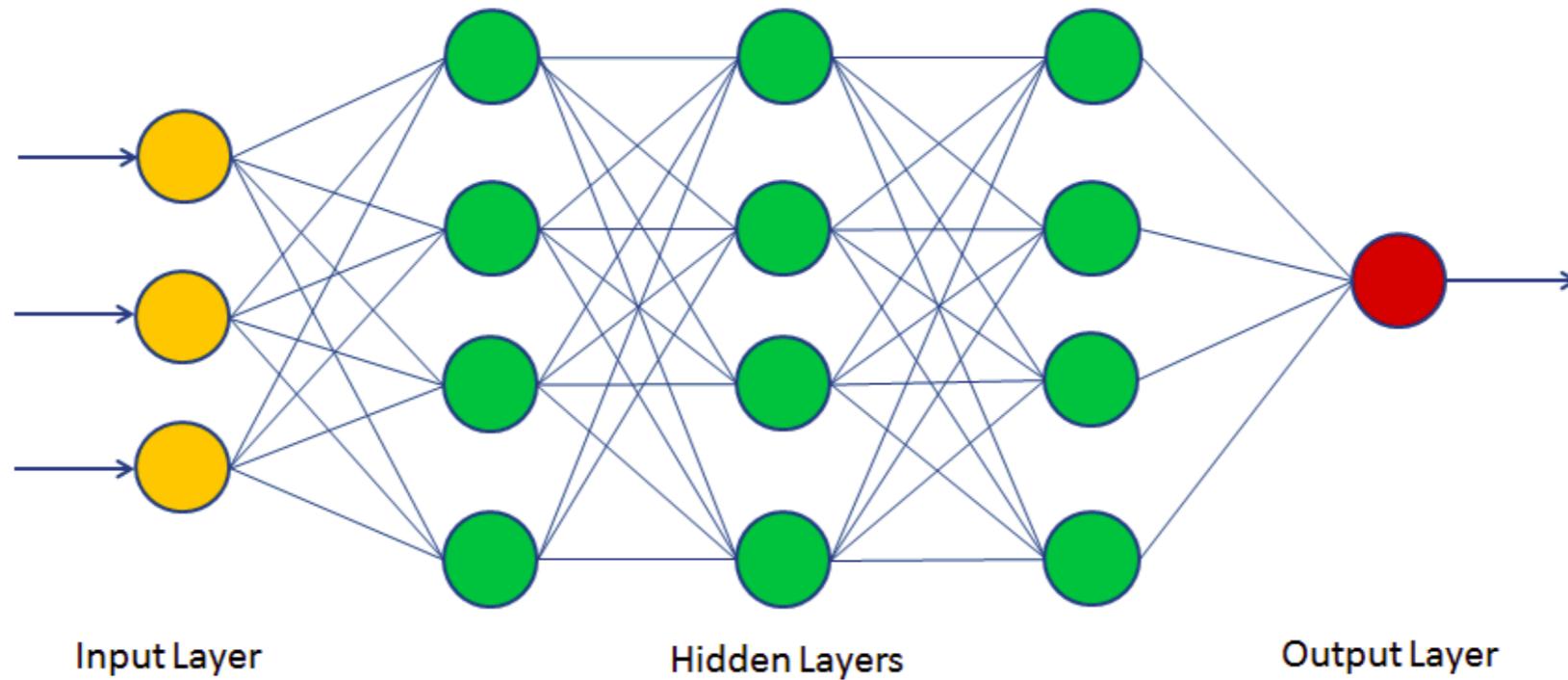
outputs from training dataset *hyperparameters*

Gaussian processes

example of Gaussian process applied to a sinusoidal data with noise added



Connection with neural nets



we know that for a sufficient number ***M of hidden units*** a neural net can approximate any function

in a Bayesian neural network, the prior distribution over model parameters combined with the network function produced a **prior distribution over the space of network outputs**

$$p(\theta) + f(x, \theta) \rightarrow p(y(x))$$

prior prob parameters

network function

prob in space of network outputs

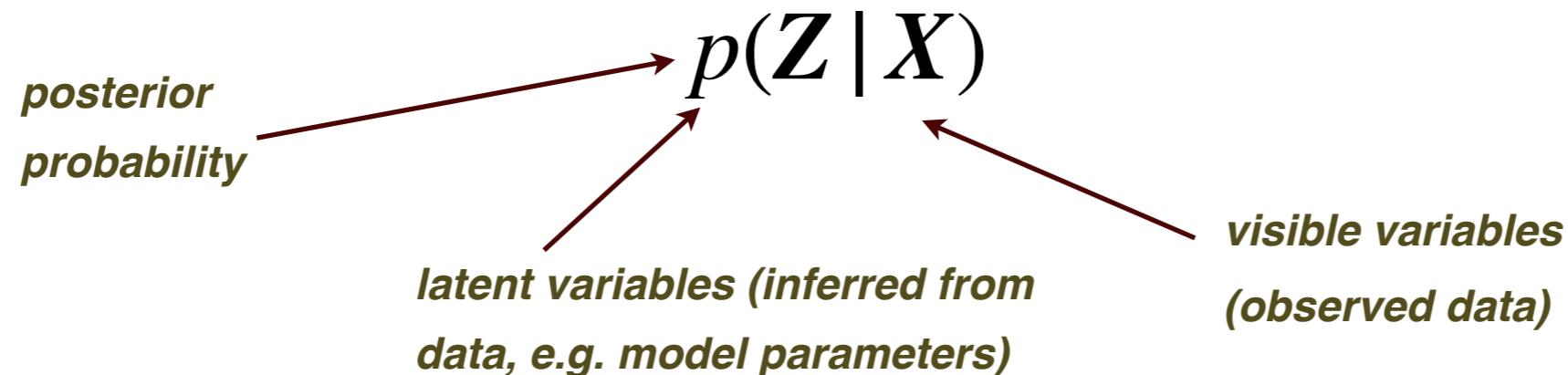
For a broad class of priors $p(\theta)$ in **limit $M \rightarrow \infty$** the NN outputs reproduce a Gaussian process

however in this limit the network outputs become statistical independent!

Variational Inference

Approximate Inference

for many ML models, the central task is the solution of the inference problem, namely determining the **posterior distribution of latent variables** given the observed data

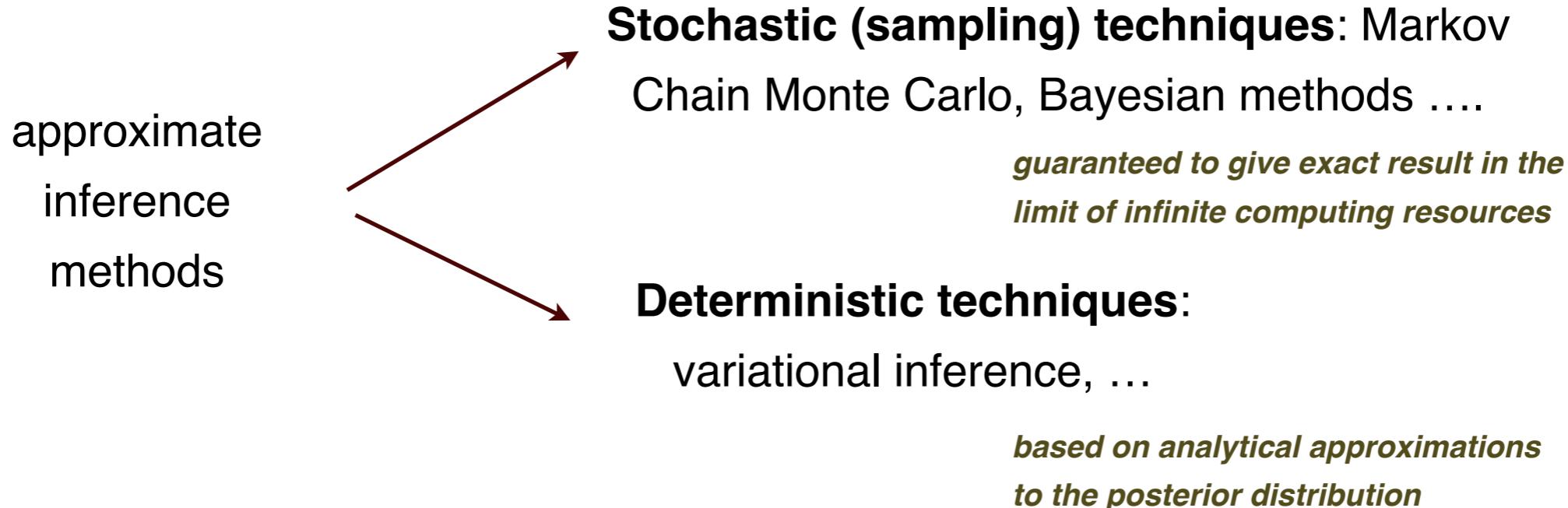


in many cases an exact solution of the **inference problem** might not be possible

- ⌚ **Latent space too high dimensionality:** too many model parameters
- ⌚ **Too complex form of posterior probability:** expectation values intractable
- ⌚ Numerical integrations might be prohibitive, and analytic integration impossible
- ⌚ **Exponentially large number of configurations** to sum over

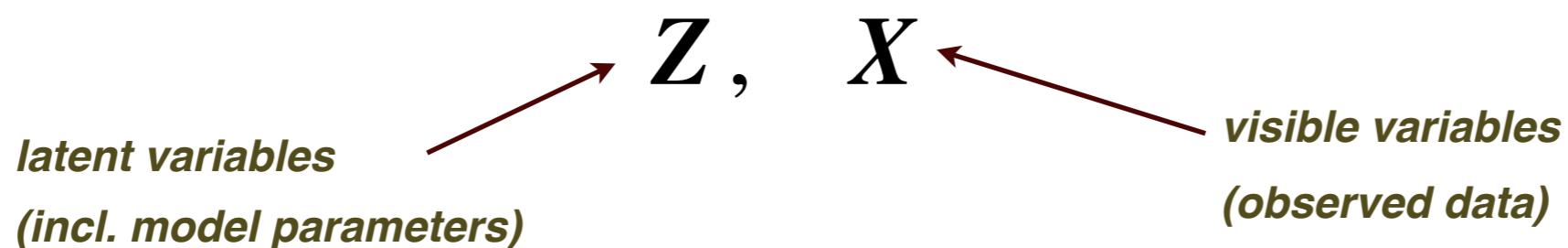
we need to rely on **approximate inference schemes** in such cases

Approximate Inference



here we will provide a brief introduction to the ideas **underlying variational inference**

Consider a **fully Bayesian model** where all parameters are given prior distributions



Goal is to find an **approximation** to the posterior distribution and the model evidence

$$p(Z | X)$$

$$p(X)$$

Approximate Inference

as in the discussion of **generative models**, we can express the log marginal probability as

$$\ln p(X) = \mathcal{L}(q) + \text{KL}(q \mid\mid p)$$

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \ln \left(\frac{p(X, \mathbf{Z})}{q(\mathbf{Z})} \right) d\mathbf{Z},$$

$$\text{KL}(q \mid\mid p) = - \int q(\mathbf{Z}) \ln \left(\frac{p(X \mid \mathbf{Z})}{q(\mathbf{Z})} \right) d\mathbf{Z},$$

since in general the true solution will be too complicated, we will restrict the possible forms of the approximate solution. For example, one can make a choice of functional form

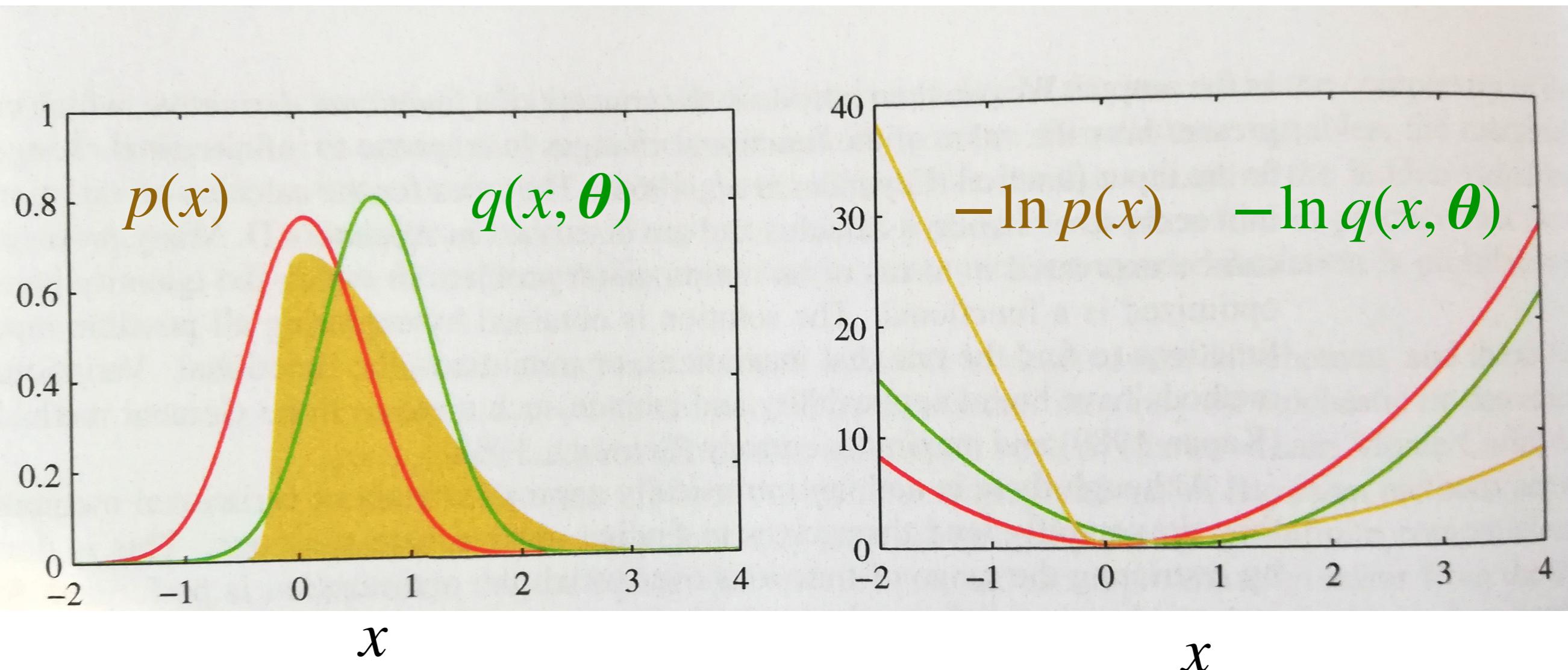
$$q(\mathbf{Z} \mid \boldsymbol{\theta})$$

Optimal values of parameters can be found from non-linear optimisation methods by minimising the KL divergence

parameters e.g mean and variance of gaussian

Approximate Inference

$$q(\mathbf{Z} | \boldsymbol{\theta}) \longrightarrow \boldsymbol{\theta} = (\bar{x}, \sigma) \quad \text{example assuming a gaussian for } q(\mathbf{Z})$$



the more complex the model, the better the approximation to the true posterior probability