E9: 309 ADL 13-1-2021

http://leap.ee.iisc.ac.in/sriram/teaching/ADL2020/

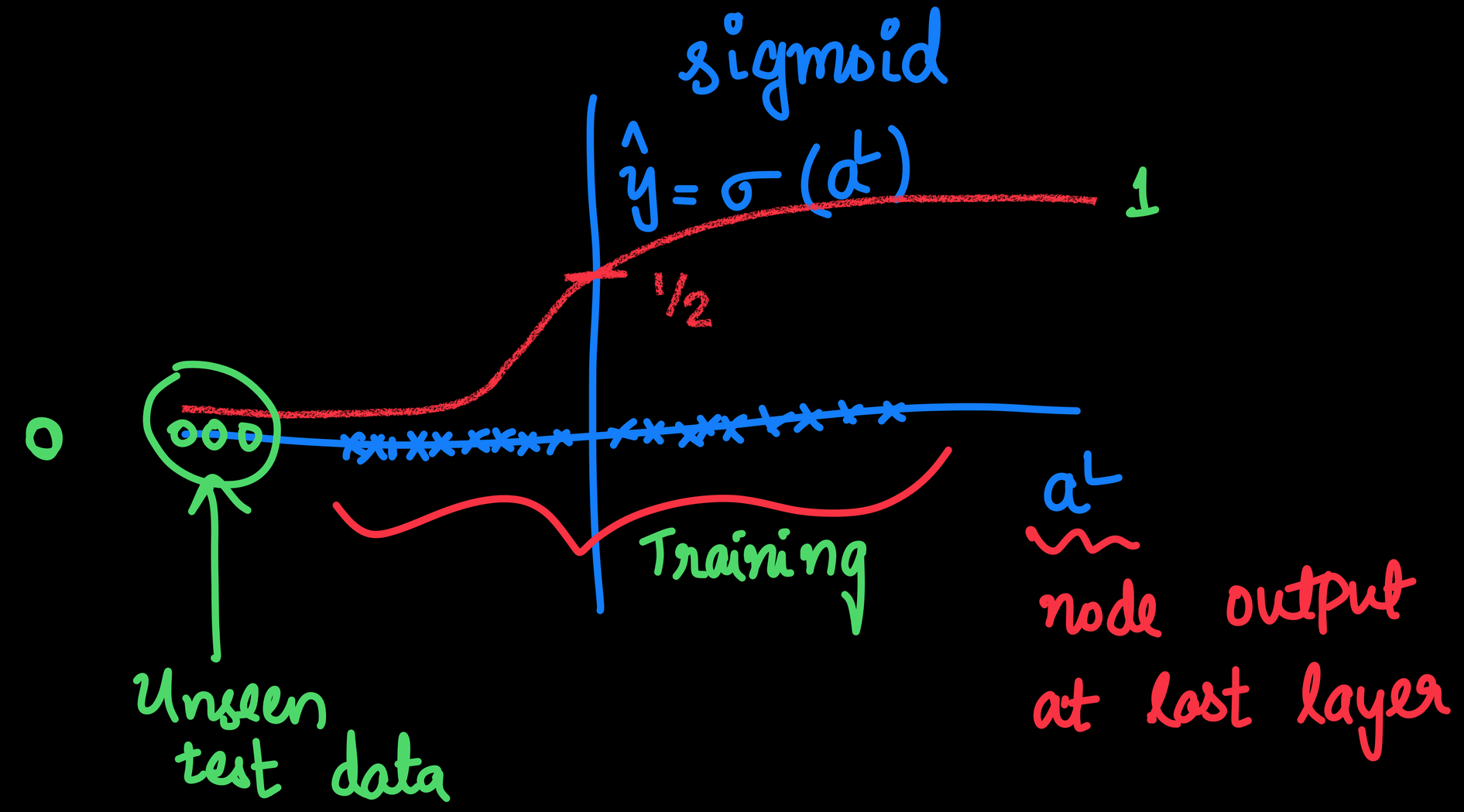
Housekeeping

- Midterm project III
  - ✓ Evaluation after final exam (1st week of Feb)
- Final Exam (as per IISc schedule)
  - ✓ Jan 23rd afternoon!
- Extra class (Friday 15, ~~nov~~ jan, 430pm)

Modeling uncertainity in deep learning

Modeling uncertainty in deep learning

- Standard deep learning tools for regression and classification do not capture model uncertainty.
- In classification, predictive probabilities obtained at the end of the pipeline (the softmax output) are often erroneously interpreted as model confidence. A model can be uncertain in its predictions even with a high softmax output.
- With model confidence at hand we can treat uncertain inputs and special cases explicitly. For example, in the case of classification, a model might return a result with high uncertainty. In this case we might decide to pass the input to a human for classification.



$P(C = 0 \mid x^*)$ ↑ [low confidence]

Research direction

- Bayesian probability theory offers us mathematically grounded tools to reason about model uncertainty, but these usually come with a prohibitive computational cost.

- Goal - show that the use of dropout (and its variants) in NNs can be interpreted as a Bayesian approximation of a well known probabilistic model.

- Goal - Develop tools for representing model uncertainty of existing dropout NNs – extracting information that has been thrown away so far. This mitigates the problem of representing model uncertainty in deep learning without sacrificing either computational complexity or test accuracy.

Bayesian Deep Learning (Basics)

Roadmap

- Bayesian methods ✓
  - ✓ Combine prior models with data to generate posterior distributions
- Neural networks ✓
  - ✓ learn functions of the data
  - ✓ need priors defined on functions
    - Example Gaussian process
- Links of Bayesian learning to practices in deep learning like dropout

Priors on functions (Gaussian Process)

Introduction to Gaussian Processes

Lecture Notes in Artificial Intelligence 3176

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Gaussian Processes in Machine Learning

Carl Edward Rasmussen

Max Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany
carl@tuebingen.mpg.de
http://www.tuebingen.mpg.de/~carl

Definition of Gaussian process

- A Gaussian Process is a collection of random variables, any finite number of which have (consistent) joint Gaussian distributions.
- A Gaussian process is fully specified by its mean function $\mathbf{m(x)}$ and covariance function $\mathbf{k(x, x')}$.

$$f \sim \mathcal{N}(m, k)$$

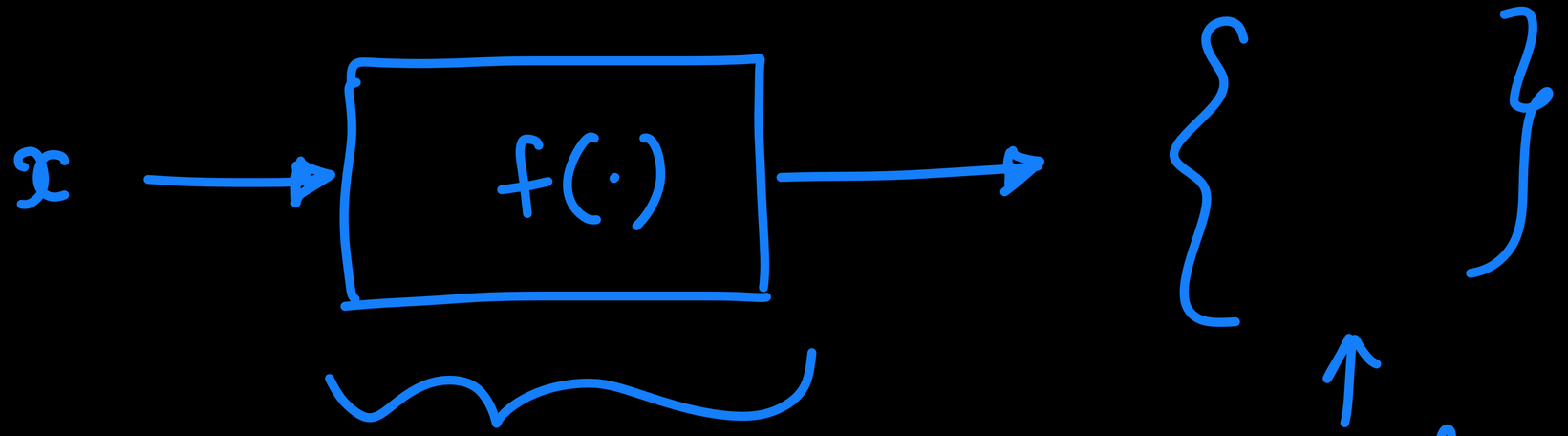
- This is a natural generalization of the Gaussian distribution whose mean and covariance is a vector and matrix, respectively. The Gaussian distribution is over vectors, whereas the Gaussian process is over functions.

Introduction to Gaussian processes

- The individual random variables in a vector from a Gaussian distribution are indexed by their position in the vector.
- For the Gaussian process it is the argument x (of the random function f(x)) which plays the role of index set: for every input x there is an associated random variable f(x), which is the value of the (stochastic) function f at that location
- For reasons of notational convenience, we will enumerate the x values of interest by the natural numbers, and use these indexes as if they were the indexes of the process

$x_1 \quad x_2 \quad \ldots \quad x_N$

x



f

stochastic function

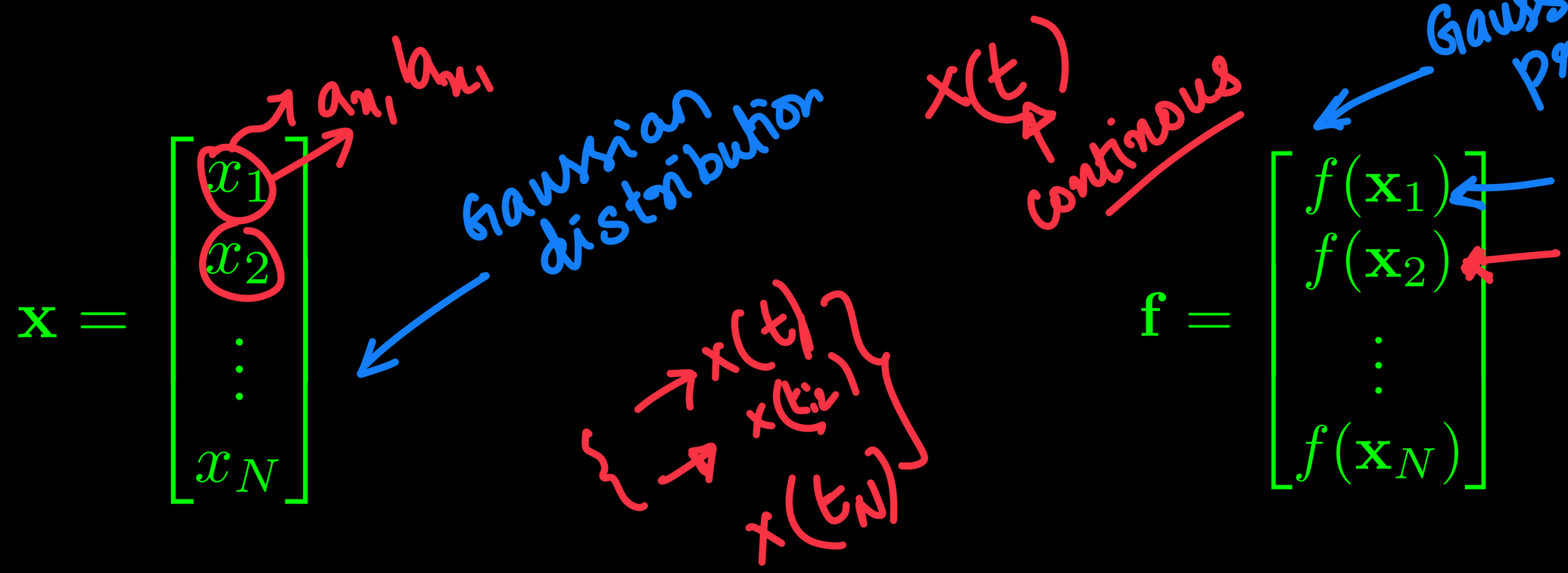
prior distribution

model uncertainity

(x, y) - training data

(x^*, y^*) f — posterior distribution $f(x, y$

Introduction to Gaussian processes



$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\mathbf{f} \sim \mathcal{N}(\boldsymbol{m}(\mathbf{x}), \boldsymbol{k}(\mathbf{x}, \mathbf{x}'))$$

- Mean will be function of $\mathbf{x}$ and variance will also be functions of two data points.

Introduction to Gaussian processes

- Let us consider an example

$$m(x) = \frac{1}{4}x^2$$

predefined mean + covariance

$$k(x, x') = exp(-\frac{1}{2}(x - x')^2)$$

$x_1 \ldots x_{20}$ samples $[-1, +1]$

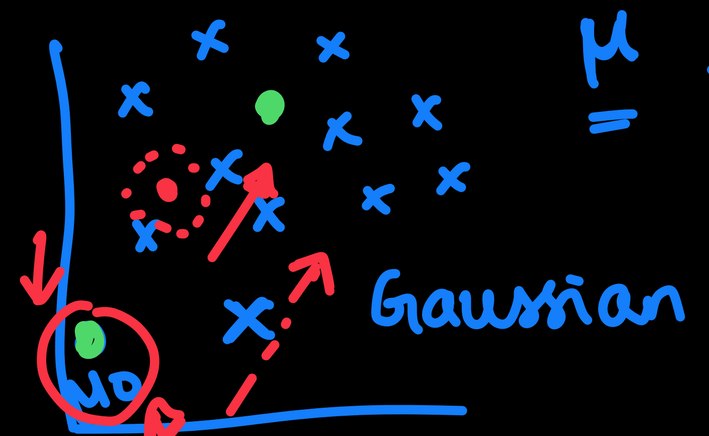
- Given the x-values we can evaluate the vector of means and a covariance matrix

$$\mu_i = m(x_i) = \frac{1}{4}x_i^2$$

$$\Sigma_{ij} = k(x_i, x_j) = exp(-\frac{1}{2}(x_i - x_j)^2)$$

$$\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Equivalence in vector space · [statistics]



$\underline{\mu}$ – random variable.

prior

$$p(\mu) \sim N(\mu_0, \sigma_0^2)$$

$$x/\mu = \{x_1 \dots x_N\}$$

$$\sim N(\mu, \sigma^2)$$

$$p(\mu/x)$$

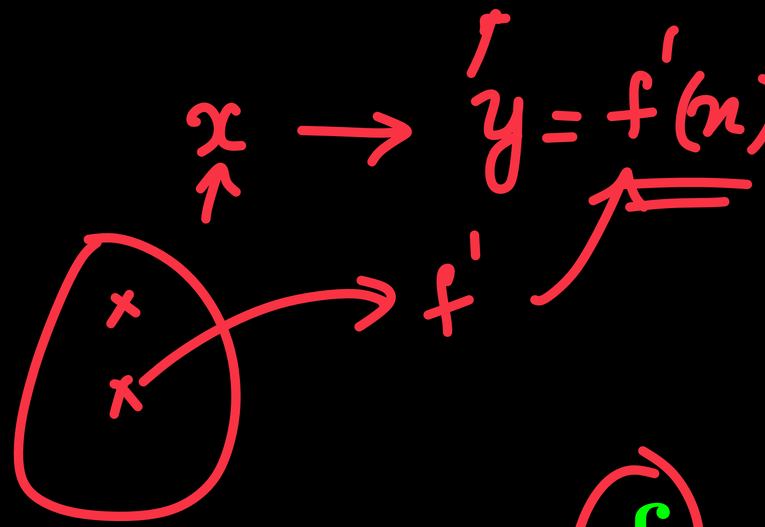
weighted comb of prior with data

$$\sim N(\quad)$$

Uncertainity in estimation : variance of posterior distr

Introduction to Gaussian processes

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$



$$\mathbf{f} = \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix}$$

mean vector

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\mathbf{f} \sim \mathcal{N}(\boldsymbol{m}(\mathbf{x}), \boldsymbol{k}(\mathbf{x}, \mathbf{x}'))$$

covariance matrix

$$\begin{bmatrix} m(x_1) \\ \\ \\ m(x_{20}) \end{bmatrix} \quad \frac{1}{4}x_1^2$$

$$\Sigma = \begin{bmatrix} \Sigma_{ij} \end{bmatrix}_{20\times 20}$$

$$\Sigma_{ij} = \exp\left(-\frac{1}{2}(x_i - x_j)^2\right)$$

$$f \sim GP(m(x), k(x, x'))$$

Gaussian process - Example

$m(x) = \frac{1}{4}x^2$

$k(x, x') = \exp\left(\frac{-1}{2}(x - x')^2\right)$

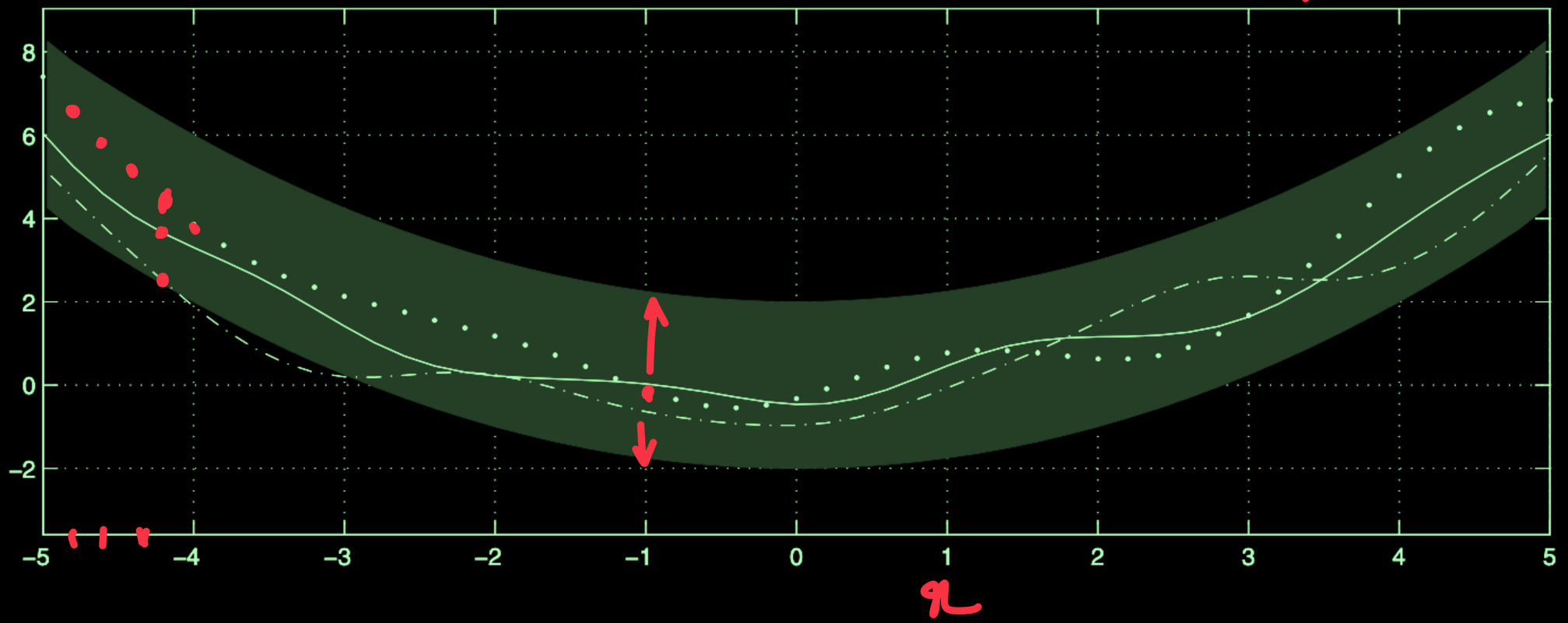


Fig. 1. Function values from three functions drawn at random from a GP as specified in Eq. (2). The dots are the values generated from Eq. (4), the two other curves have (less correctly) been drawn by connecting sampled points. The function values suggest a smooth underlying function; this is in fact a property of GPs with the squared exponential covariance function. The shaded grey area represent the 95% confidence intervals

Gaussian processes for Bayesian inference

- GP will be used as a prior for Bayesian inference.
- The prior does not depend on the training data, but specifies some properties of the functions.
- One of the primary goals computing the posterior is that it can be used to make predictions for unseen test cases.
- Let $\mathbf{f}$ be the known function values of the training cases, and let $\mathbf{f}_*$ be a set of function values corresponding to the test set inputs, $\mathbf{X}_*$.

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} \Sigma & \Sigma_* \\ \Sigma_*^T & \Sigma_{**} \end{bmatrix} \right)$$

f: $\underline{X}$ - training data. (N)

$\underline{f^*}$: x^* - test data

$$\begin{pmatrix} m(x_1) \\ \vdots \\ m(x_N) \end{pmatrix}_{N\times 1} \quad \begin{bmatrix} k(x_i, x_j) \end{bmatrix}_{N\times N}$$

$$\Rightarrow \begin{bmatrix} m(x_1^*) \\ \vdots \\ m(x_T^*) \end{bmatrix} \quad k(x_i^*, x_j^*)$$

$$\left\{ \begin{bmatrix} x_1 \\ \\ x_N \\ x_1^* \\ x_T^* \end{bmatrix} \right\}_{i=1}$$

$$\underline{M} = \begin{bmatrix} M \\ M^* \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} & & \\ & ** & \\ & & * \end{bmatrix}_{(N+T)}$$

Gaussian processes for Bayesian inference

- Now the quantity of interest is the posterior distribution (for function values)

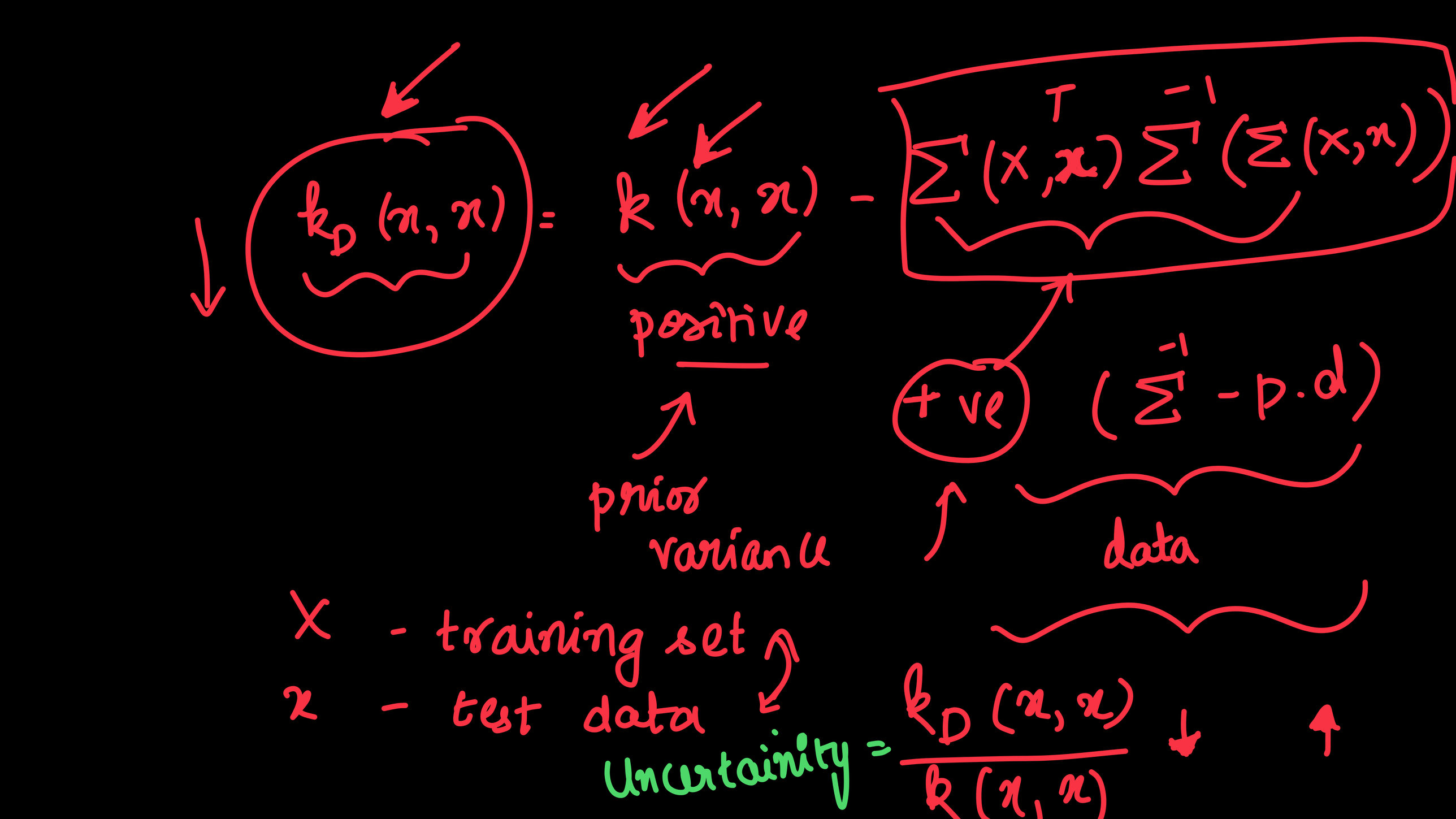
$$\mathbf{f}_* | \mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}_* + \boldsymbol{\Sigma}_*^T \boldsymbol{\Sigma}^{-1}(\mathbf{f} - \boldsymbol{\mu}), \boldsymbol{\Sigma}_{**} - \boldsymbol{\Sigma}_*^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\Sigma}_*)$$

- Thus, suppose one test data-point $\mathbf{x}$

$$f|D \sim \mathcal{GP}(m_D, k_D)$$

$$m_D(x) = m(x) + \Sigma(\mathbf{X}, x)^T \boldsymbol{\Sigma}^{-1}(\mathbf{f} - \mathbf{m})$$

$$k_D(x, x') = k(x, x') - \Sigma(\mathbf{X}, x)^T \boldsymbol{\Sigma}^{-1} \Sigma(\mathbf{X}, x')$$


$k_D(x, x) = k(x, x) - \Sigma(x, x)\,\Sigma^{-1}\,\Sigma^{T}(x, x)$
positive
prior variance
+ve
$(\Sigma^{-1} - p.d)$
data
X - training set
x - test data
Uncertainity $= \frac{k_D(x, x)}{k(x, x)}$

Gaussian Processes

- where $\Sigma(X, x)$ is a vector of covariances between every training case and x. These are the central equations for Gaussian process predictions.

- Let's examine these equations for the posterior mean and covariance. Notice that the posterior variance $kD(x, x)$ is equal to the prior variance $k(x, x)$ minus a positive term, which depends on the training inputs;

- thus the posterior variance is always smaller than the prior variance, since the data has given us some additional information.

Allowing for noise in the model

- Need to address one final issue: noise in the training outputs.
- It is common to many applications of regression that there is noise in the observations6.
- The most common assumption is that of additive i.i.d. Gaussian noise in the outputs.
- In Gaussian process, the effect is that every f(x) has a extra covariance with itself only (since the noise is assumed independent), with a magnitude equal to the noise variance:

$$y = f(x) + \epsilon$$

Allowing for noise in the model

$$y(x) = f(x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

$$f \sim \mathcal{GP}(m, k), \quad y \sim \mathcal{GP}(m, k + \sigma_n^2 \delta_{i,i'})$$

variance

- Notice, that the indexes to the Kronecker's delta is the identify of the cases, `i`, and not the inputs `xi`; you may have several cases with identical inputs, but the noise on these cases is assumed to be independent.

$$x_i = x_j = x \qquad \delta_{i,i'}$$

Allowing for noise in the model

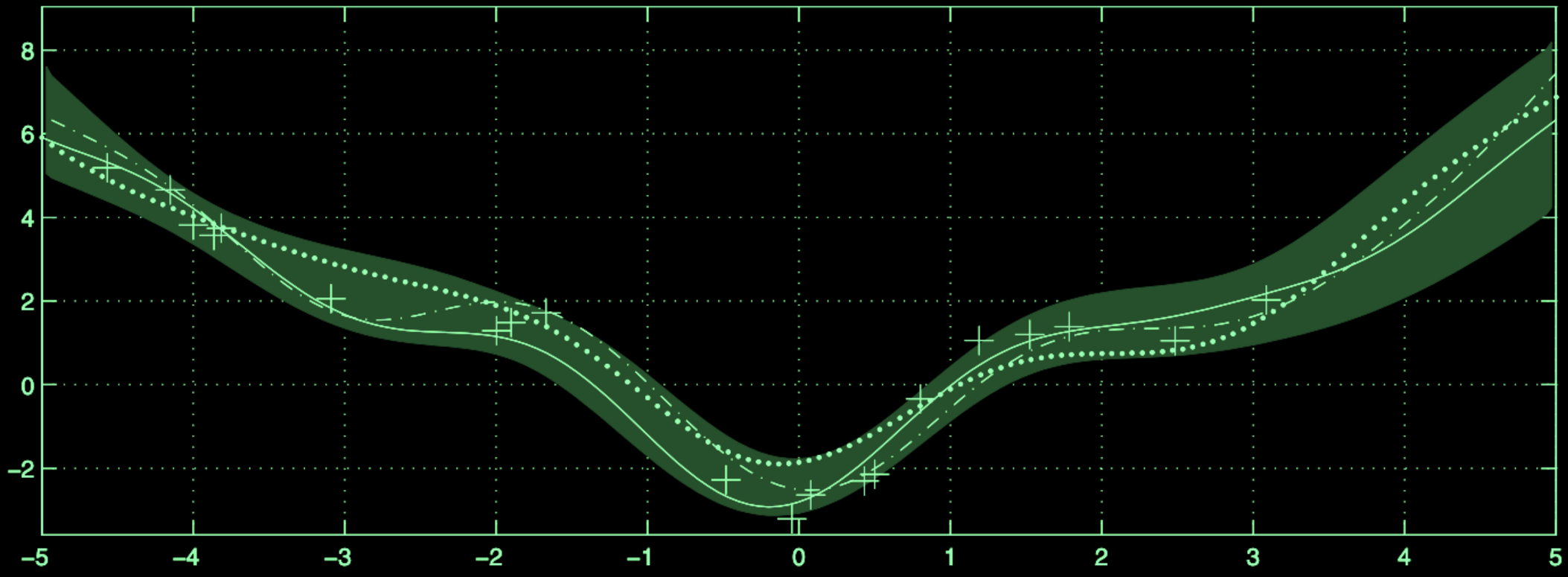


Fig. 2. Three functions drawn at random from the posterior, given 20 training data points, the $\mathcal{GP}$ as specified in Eq. (3) and a noise level of $\sigma_n = 0.7$. The shaded area gives the 95% confidence region. Compare with Figure 1 and note that the uncertainty goes down close to the observations

Dropout and its Bayesian Interpretation

Broad goal

- Interpretation of dropout as a Bayesian model

  - ✓ offers an explanation to some of its properties, such as its ability to avoid over-fitting

  - ✓ our insights allow us to treat NNs with dropout as fully Bayesian models, and obtain uncertainty estimates over their features.

- Mathematically,

  - ➡ we will show that a deep neural network (NN) with arbitrary depth and non-linearities, with dropout applied before every weight layer, is mathematically equivalent to an approximation to the probabilistic deep Gaussian process model

Dropouts

Dropout as a Bayesian Approximation:
Representing Model Uncertainty in Deep Learning

Yarin Gal YG279@CAM.AC.UK
Zoubin Ghahramani ZG201@CAM.AC.UK
University of Cambridge

Dropout in NN

- Reviewing the dropout NN model quickly for the case of a single hidden layer NN. This is done for ease of notation, and the generalisation to multiple layers is straightforward.

- Denote by W1,W2 the weight matrices connecting the first layer to the hidden layer and connecting the hidden layer to the output layer respectively. These linearly transform the layers' inputs before applying some element-wise non-linearity $\sigma(\cdot)$. Denote by b the biases by which we shift the input of the non-linearity. We assume the model to output D dimensional vectors while its input is Q dimensional vectors, with K hidden units. Thus W1 is a $Q \times K$ matrix, W2 is a $K \times D$ matrix, and b is a K dimensional vector. A standard NN model would

$$\hat{\mathbf{y}} = \sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b})\mathbf{W}_2$$

- Dropout is applied by sampling two binary vectors `z1`, `z2` of dimensions `Q` and `K` respectively. The elements of the vectors are distributed according to a Bernoulli distribution with some parameter

$$p_i \in \{0, 1\} \quad i = 1, 2$$

$$z_{1q} \sim Bernoulli(p_1)$$
$$z_{2k} \sim Bernoulli(p_2)$$

- Given an input `x`, `(1 - p1)` proportion of the elements of the input are set to zero.

- The output with dropout can be expressed as

$$\hat{\mathbf{y}} = \sigma(\mathbf{x}(\mathbf{Z}_1\mathbf{W}_1) + \mathbf{b})(\mathbf{Z}_2\mathbf{W}_2)$$
$$\mathbf{Z}_1 = diag(\mathbf{z}_1) \quad \mathbf{Z}_2 = diag(\mathbf{z}_2)$$

Loss function

- Loss in regression networks

$$E = \frac{1}{2N} \sum_n ||\mathbf{y}_n - \hat{\mathbf{y}}_n||^2$$

- Loss in classification networks

$$c_n \in [1 \ldots D]$$

$$\hat{p}_{nd} = \frac{exp(\hat{y}_{nd})}{\sum_{d'} exp(\hat{y}_{nd'})} \qquad E = -\frac{1}{N} \sum_n log\ \hat{p}_{nc_n}$$

- With L2 regularization, the total loss is

$$\mathcal{E}_{dropout} = E + \lambda_1 ||\mathbf{W}_1||^2 + \lambda_2 ||\mathbf{W}_2||^2 + \lambda_3 ||\mathbf{b}||^2$$

Gaussian process

$$\mathbf{f}|\mathcal{X} \sim \mathcal{G}P(\mathbf{0}, \mathbf{K}(\mathbf{X}, \mathbf{X}))$$

$$\mathbf{Y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \frac{1}{\tau}\mathbf{I}_N)$$

- To model the data we have to choose a covariance function `K(X1, X2)` for the Gaussian distribution. This function defines the (scalar) similarity between every pair of input points `K(xi , xj )`.

- Given a finite dataset of size `N` this function induces an `N × N` covariance matrix which we will denote `K := K(X, X)`.

Variational Inference

- The output probability distribution on some unseen test data

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) d\boldsymbol{\omega}$$

- condition the model on a finite set of random variables $\boldsymbol{\omega}$

  - ✓ like the weights of the model.

- The distribution $p(\omega|X, Y)$ cannot usually be evaluated analytically. Instead we define an approximating variational distribution $q(\omega)$