This project AIDA receives funding from the European Union’s Horizon 2020 Research and Innovation programme under grant agreement No 776262.

Physics informed Machine Learning

22 January 20120

Contributors: Romain Dupuis

Romain Dupuis
KU Leuven

KU LEUVEN

1st AIDA School in Bologna

Contents

Motivations

Why incorporating physics into ML?

- **Machine Learning shows great success**
  - Computer vision, natural language processing, autonomous vehicle
  - Medicine, engineering, fluid dynamics, materials, etc.
  - Full data-driven approach

- **Scientists can be skeptical about black-box tools**
  - Complex models: no interpretation
  - Solutions can break natural laws

- **Small amount of data**
  - Experimental data
  - Intensive simulations

Why incorporating physics into ML?

- **Machine Learning shows great success**
  - Computer vision, natural language processing, autonomous vehicle
  - Medicine, engineering, fluid dynamics, materials, etc.
  - Full data-driven approach

- **Scientists can be skeptical about black-box tools**
  - Complex models: no interpretation
  - Solutions can break natural laws

- **Small amount of data**
  - Experimental data
  - Intensive simulations

Incorporating additional knowledge

Data Informed Physics

Reduced order model

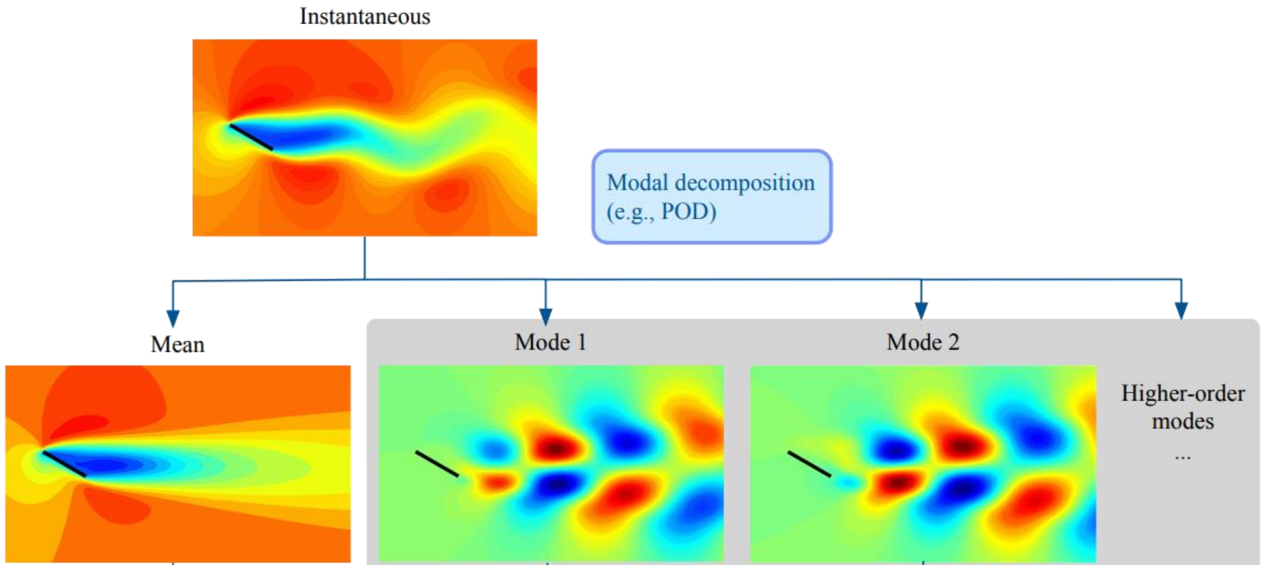
- **Non Linear dynamic system (Physics)**

$$\frac{d\boldsymbol{u}}{dt} = L\,\boldsymbol{u}(t) + N\big(\boldsymbol{u}(t)\big), \qquad \boldsymbol{u} \in \mathbb{R}^n$$

- **Extracting low-dimensional patterns (data)**

$$u(t) \approx \boldsymbol{\phi}_r \boldsymbol{a}(t), \qquad \boldsymbol{\phi}_r^T \boldsymbol{\phi}_r = \boldsymbol{I}$$

$$\boldsymbol{a} \in \mathbb{R}^r \; with \; r \ll n$$



[1] Taira et al.

- **Low rank system**

$$\frac{d\boldsymbol{a}}{dt} = \boldsymbol{\phi}_r^T L \boldsymbol{\phi}_r \boldsymbol{a}(t) + \boldsymbol{\phi}_r^T N\big(\boldsymbol{\phi}_r \boldsymbol{a}(t)\big), \qquad \boldsymbol{a} \in \mathbb{R}^r$$

Reduced order model

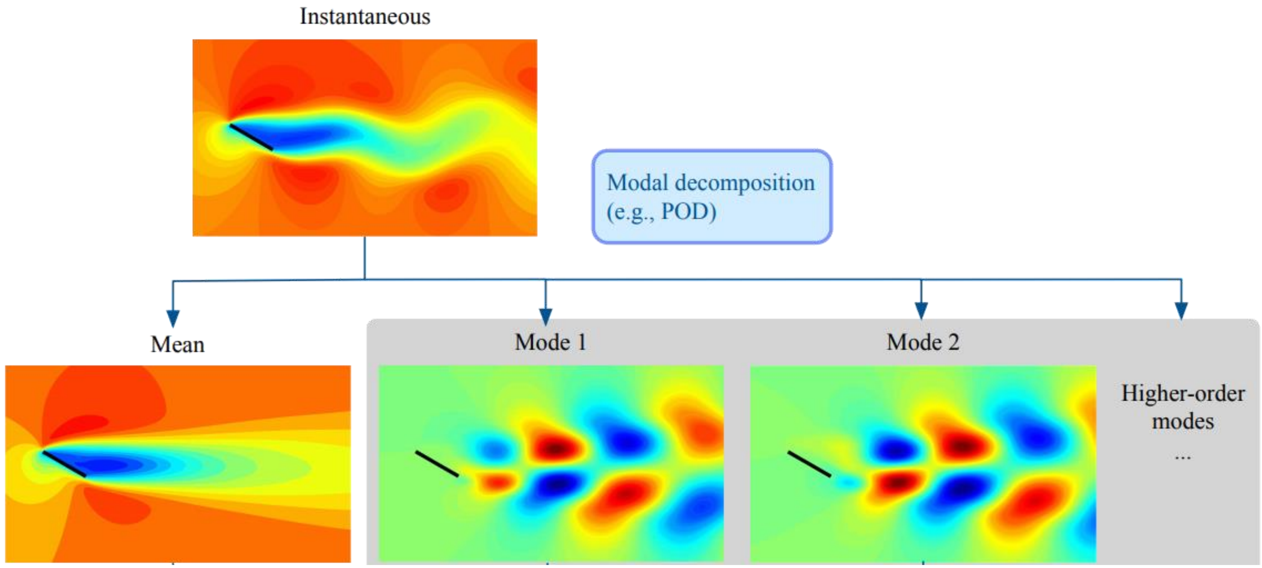
- **Non Linear dynamic system (Physics)**

$$\frac{d\boldsymbol{u}}{dt} = L\,\boldsymbol{u}(t) + N\big(\boldsymbol{u}(t)\big), \qquad \boldsymbol{u} \in \mathbb{R}^n$$

- **Extracting low-dimensional patterns (data)**

$$u(t) \approx \boldsymbol{\phi}_r \boldsymbol{a}(t), \qquad \boldsymbol{\phi}_r^T \boldsymbol{\phi}_r = \boldsymbol{I}$$

$$\boldsymbol{a} \in \mathbb{R}^r \text{ with } r \ll n$$



[1] Taira et al.

- **Low rank system**

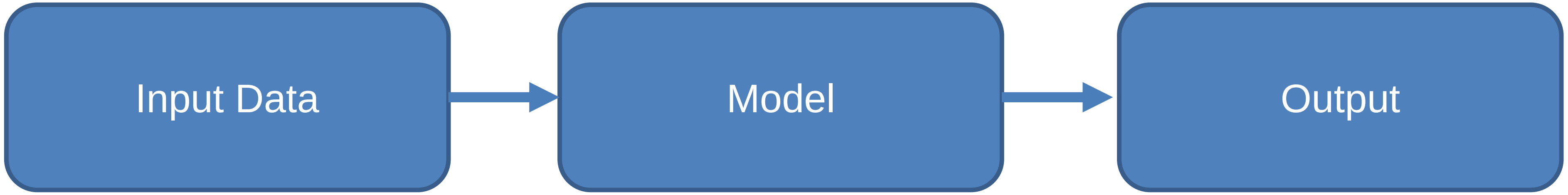
$$\frac{d\boldsymbol{a}}{dt} = \boxed{\boldsymbol{\phi}_r^T L \boldsymbol{\phi}_r \boldsymbol{a}(t)} + \boxed{\boldsymbol{\phi}_r^T N\big(\boldsymbol{\phi}_r \boldsymbol{a}(t)\big)}, \qquad \boldsymbol{a} \in \mathbb{R}^r$$

Physics-informed ML

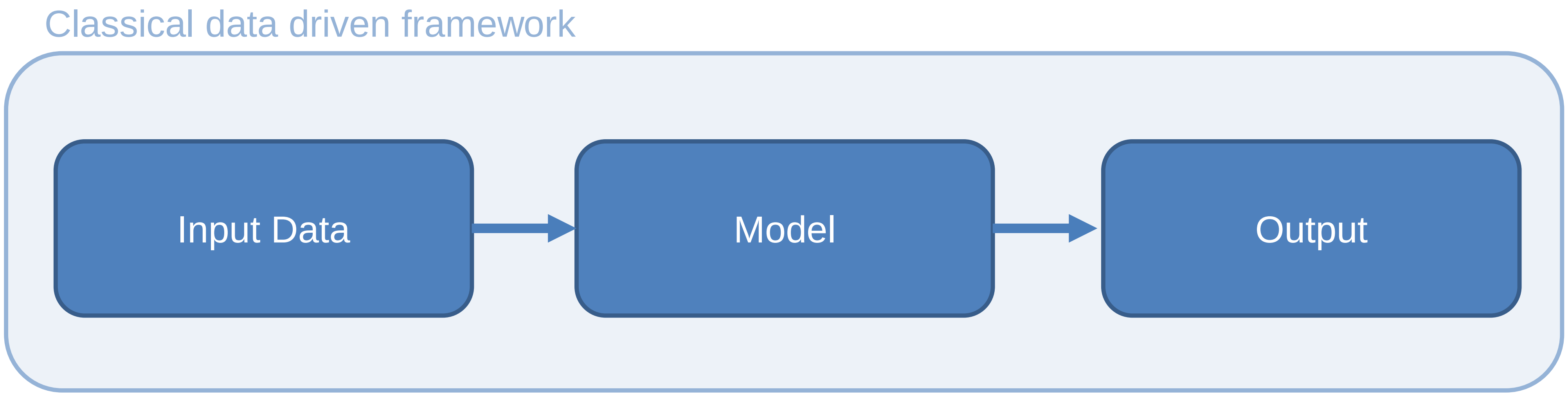
Very large taxonomy

- **Physics informed machine learning**
- **Knowledge-based machine learning**
- **Theory-guided machine learning**
- **Physics-constrained machine learning**
- **Etc…**

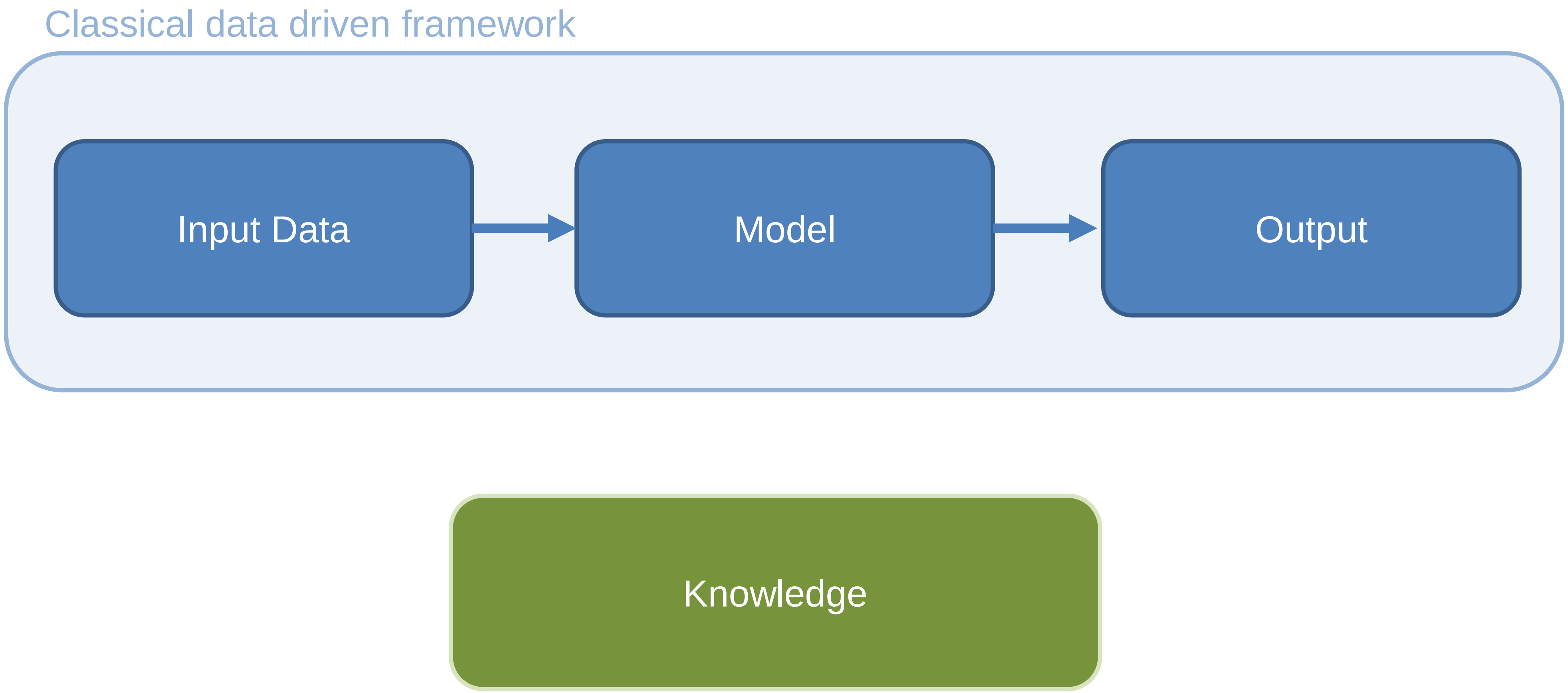
Informed Machine Learning



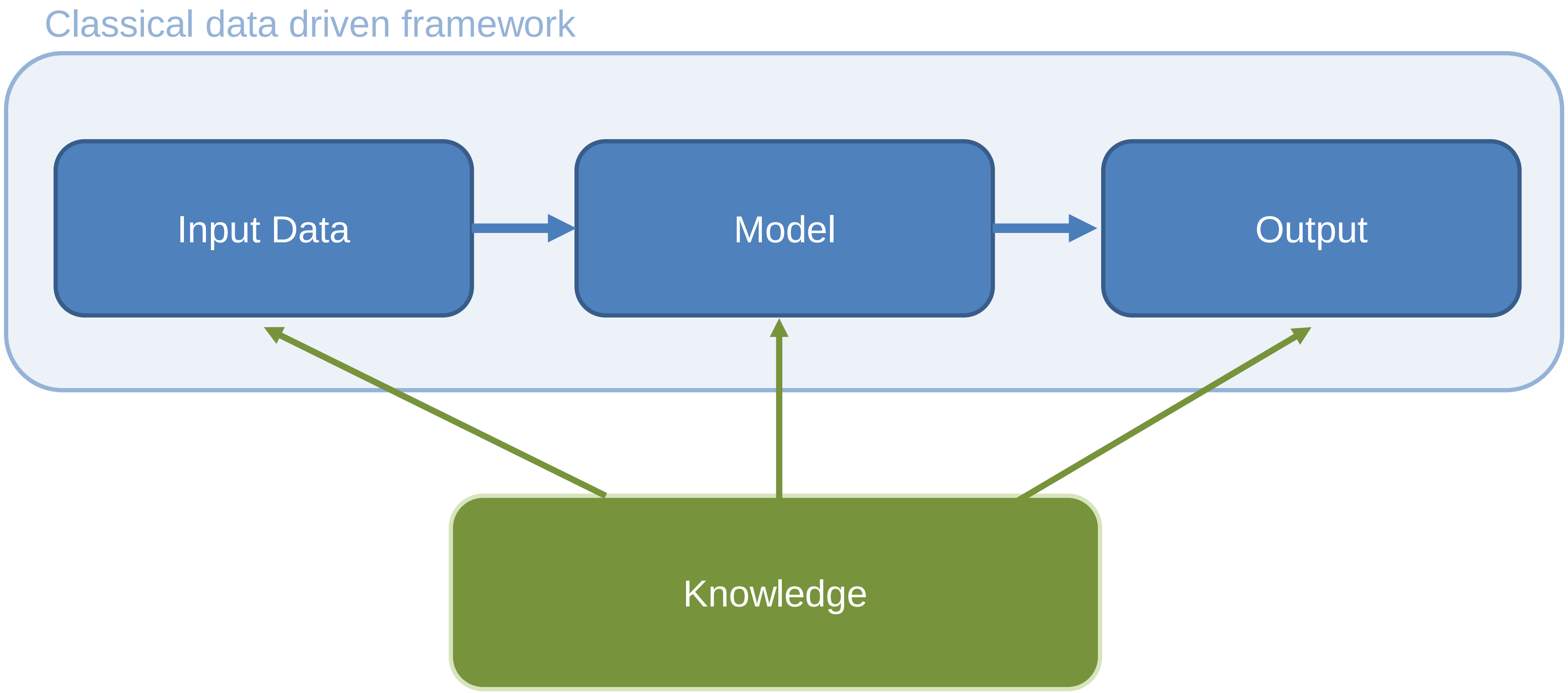
Informed Machine Learning



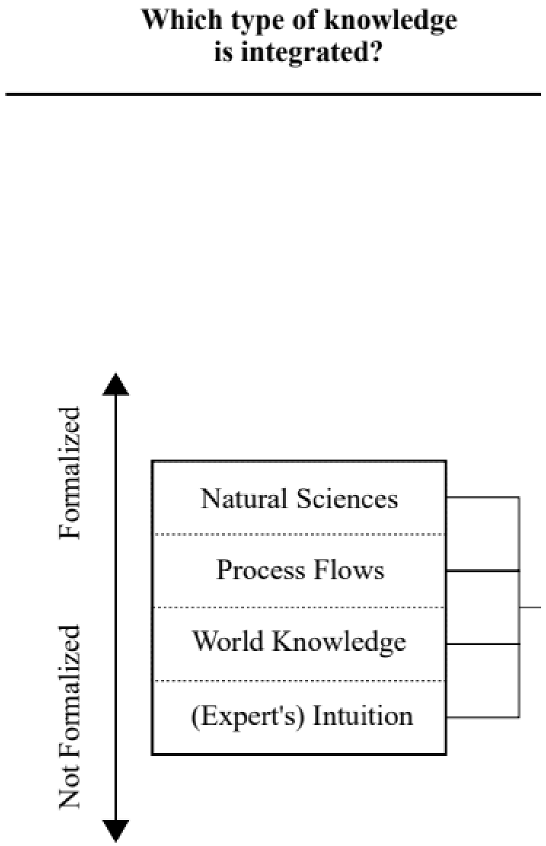
Informed Machine Learning



Informed Machine Learning

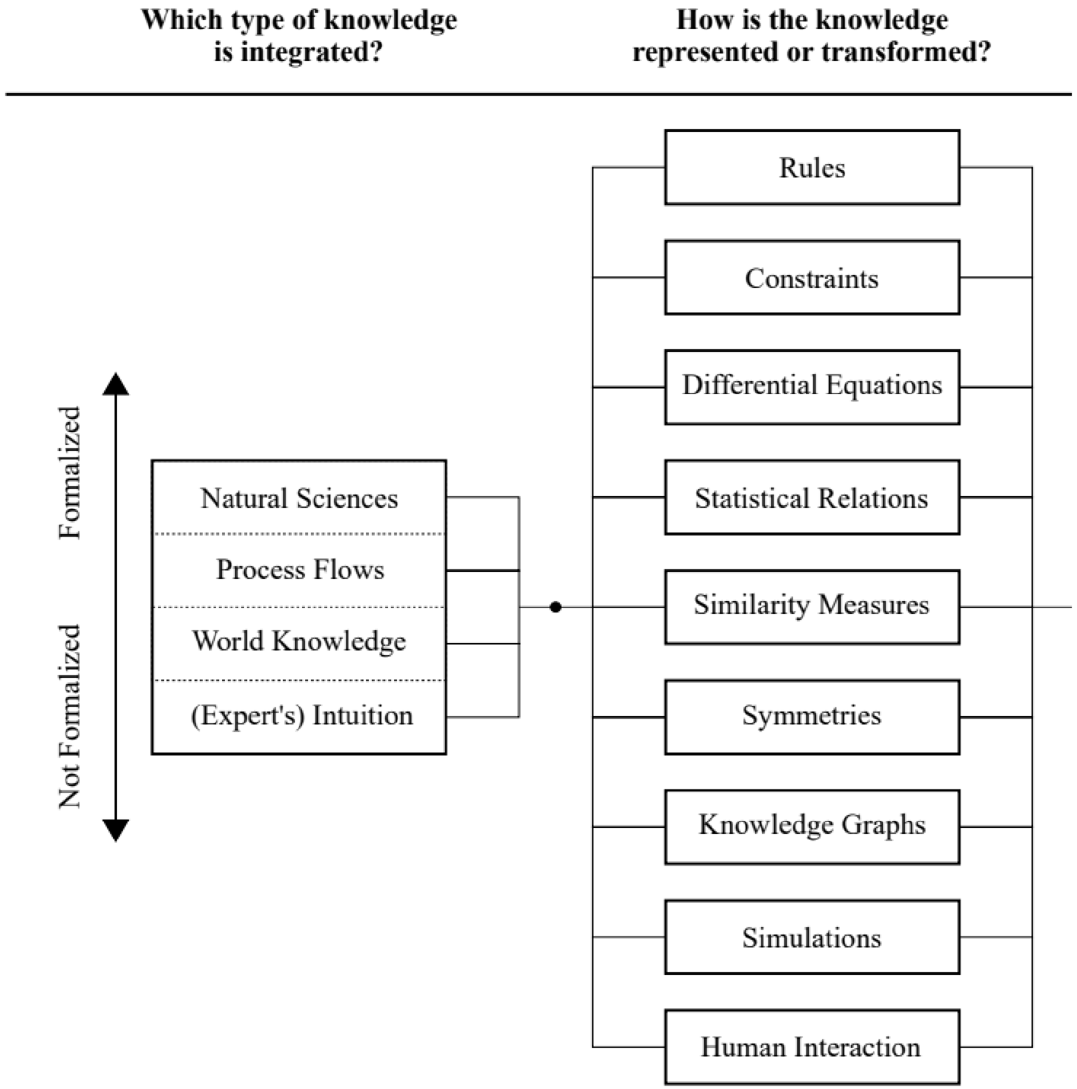


Type of knowledge



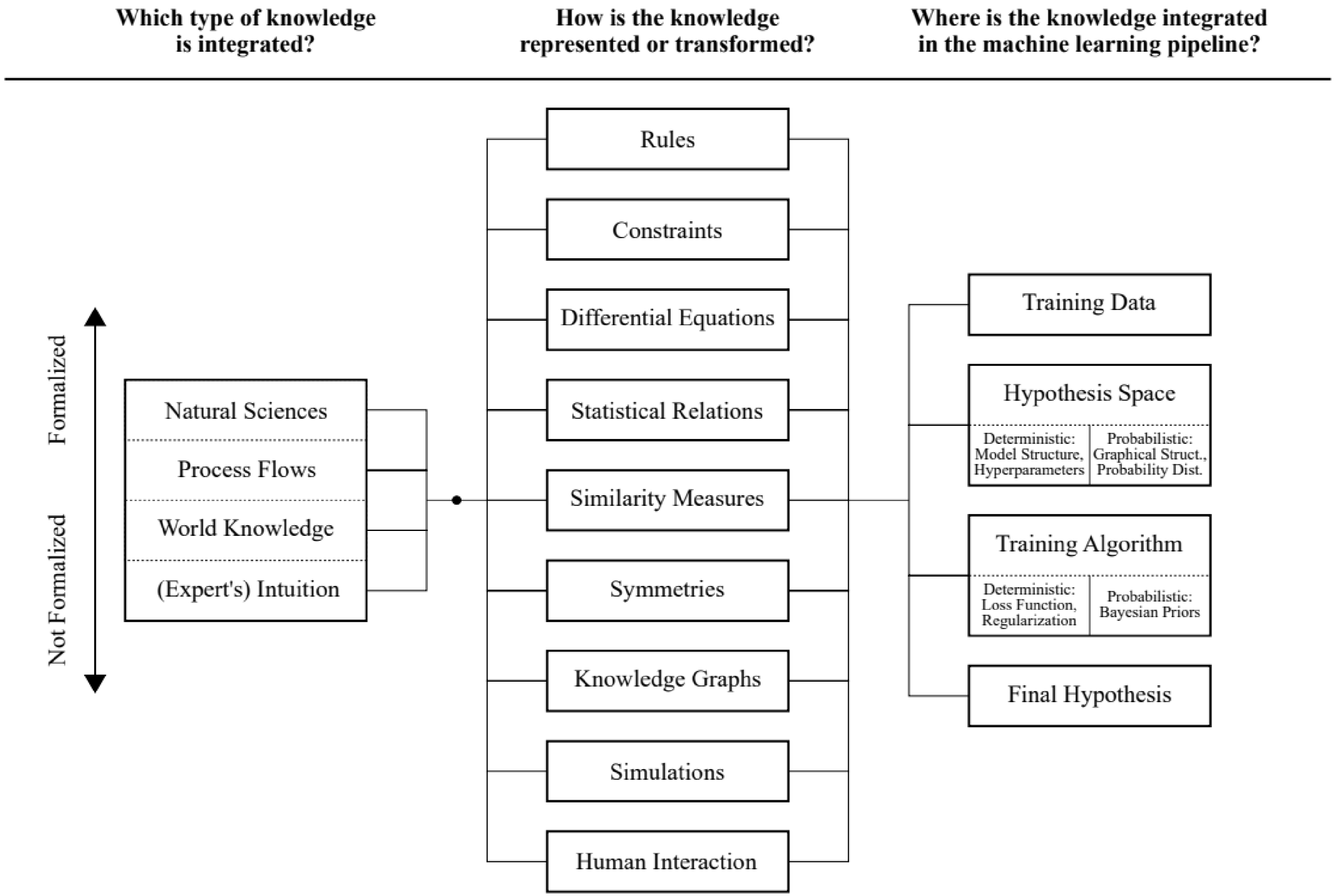
von Rued et al.

Type of knowledge



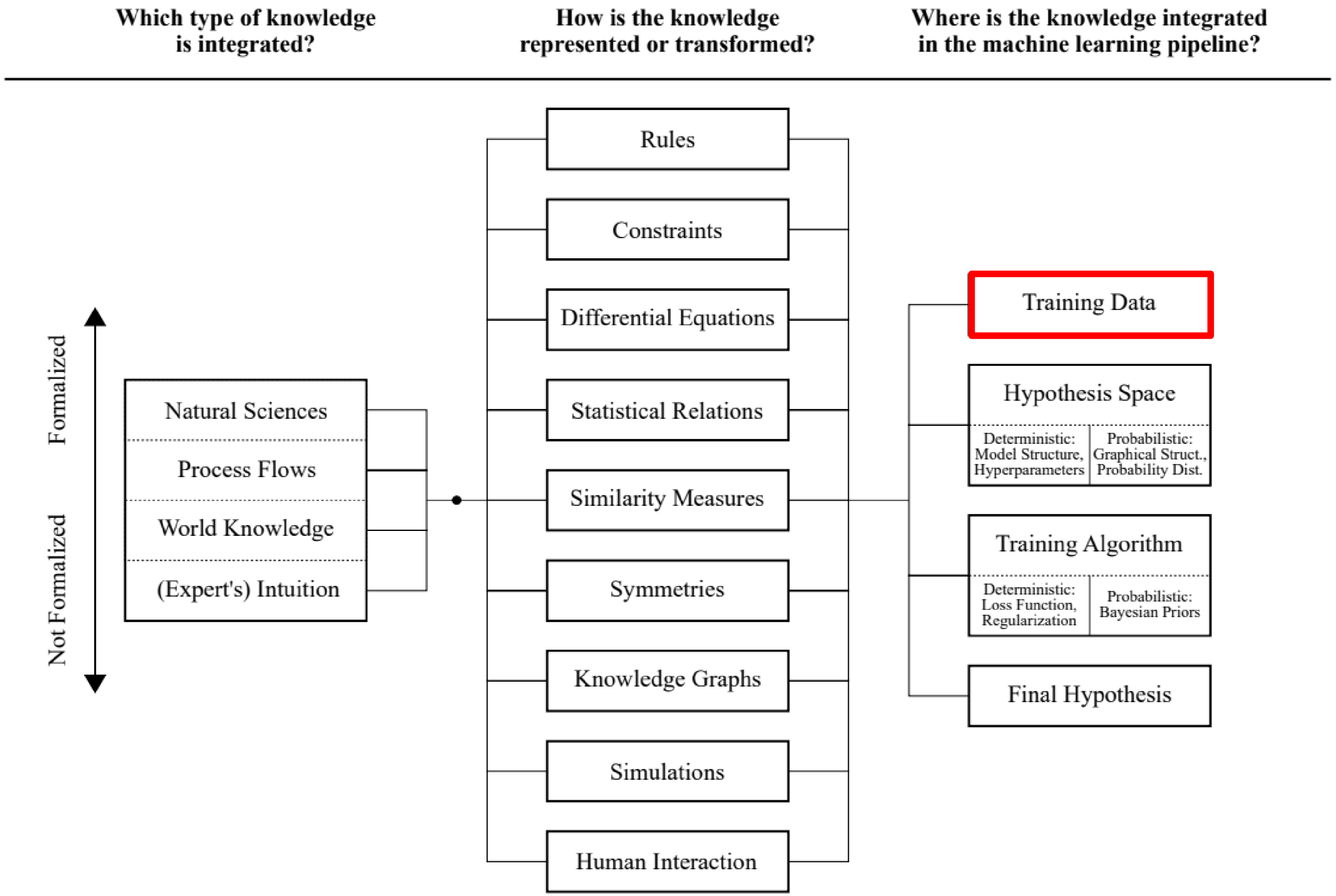
von Rued et al.

Type of knowledge



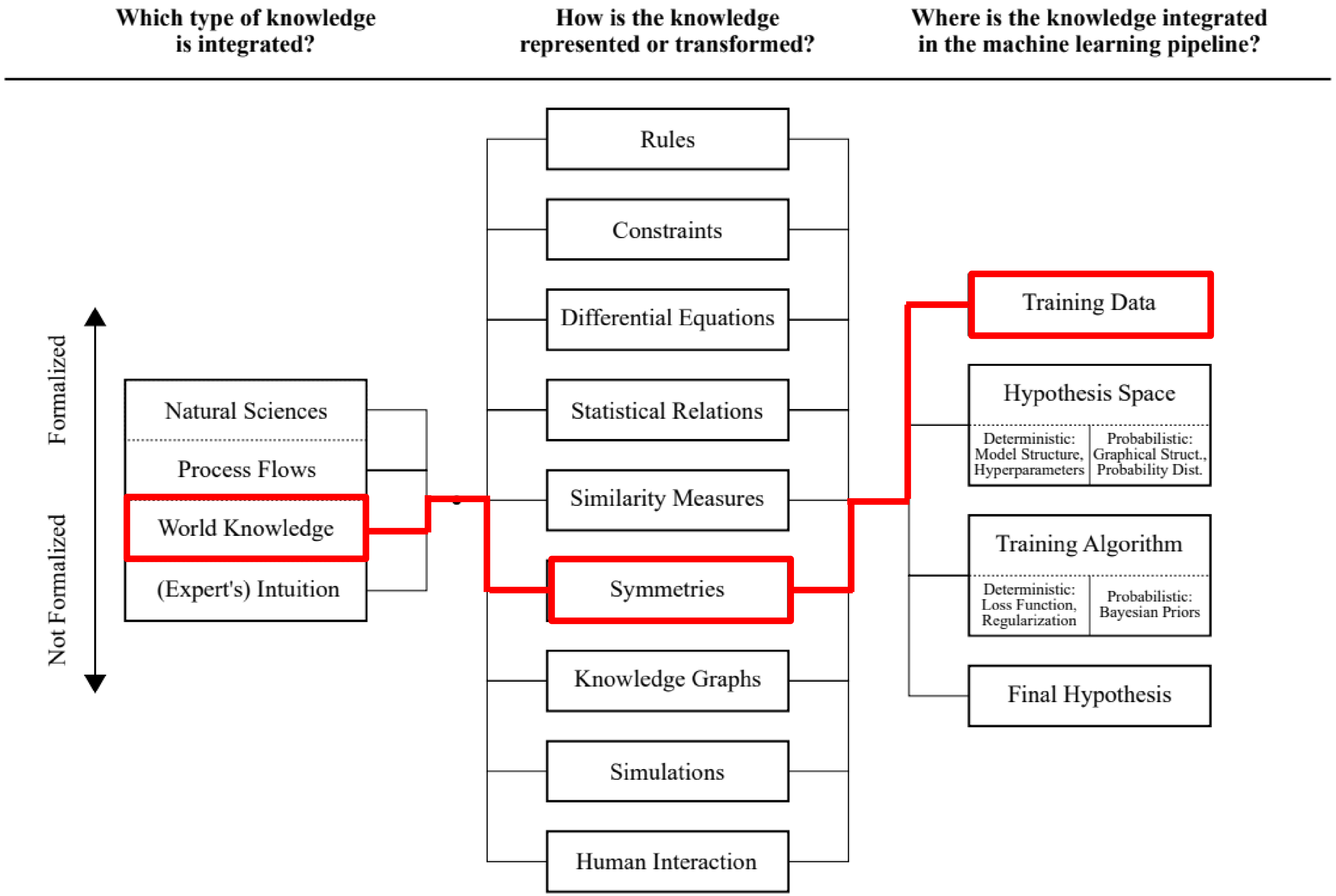
von Rued et al.

Data augmentation



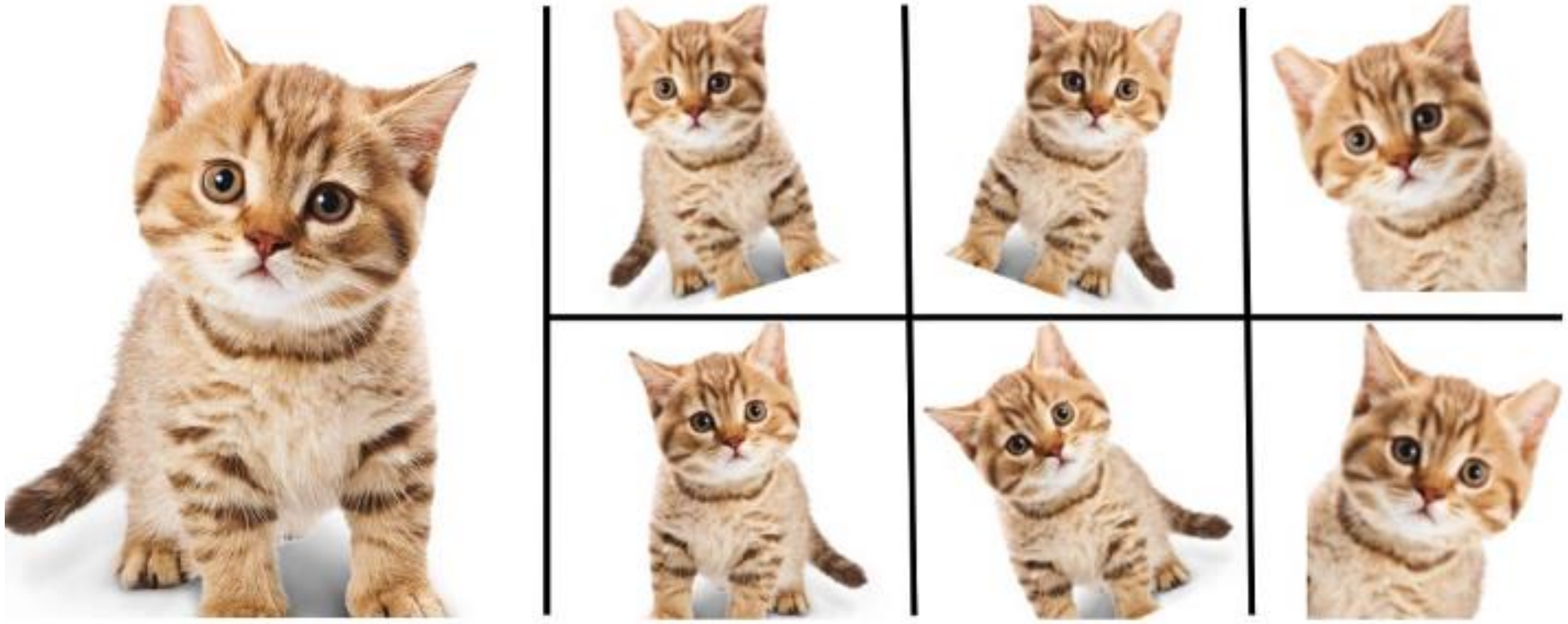
von Rued et al.

Data augmentation: geometrical invariance



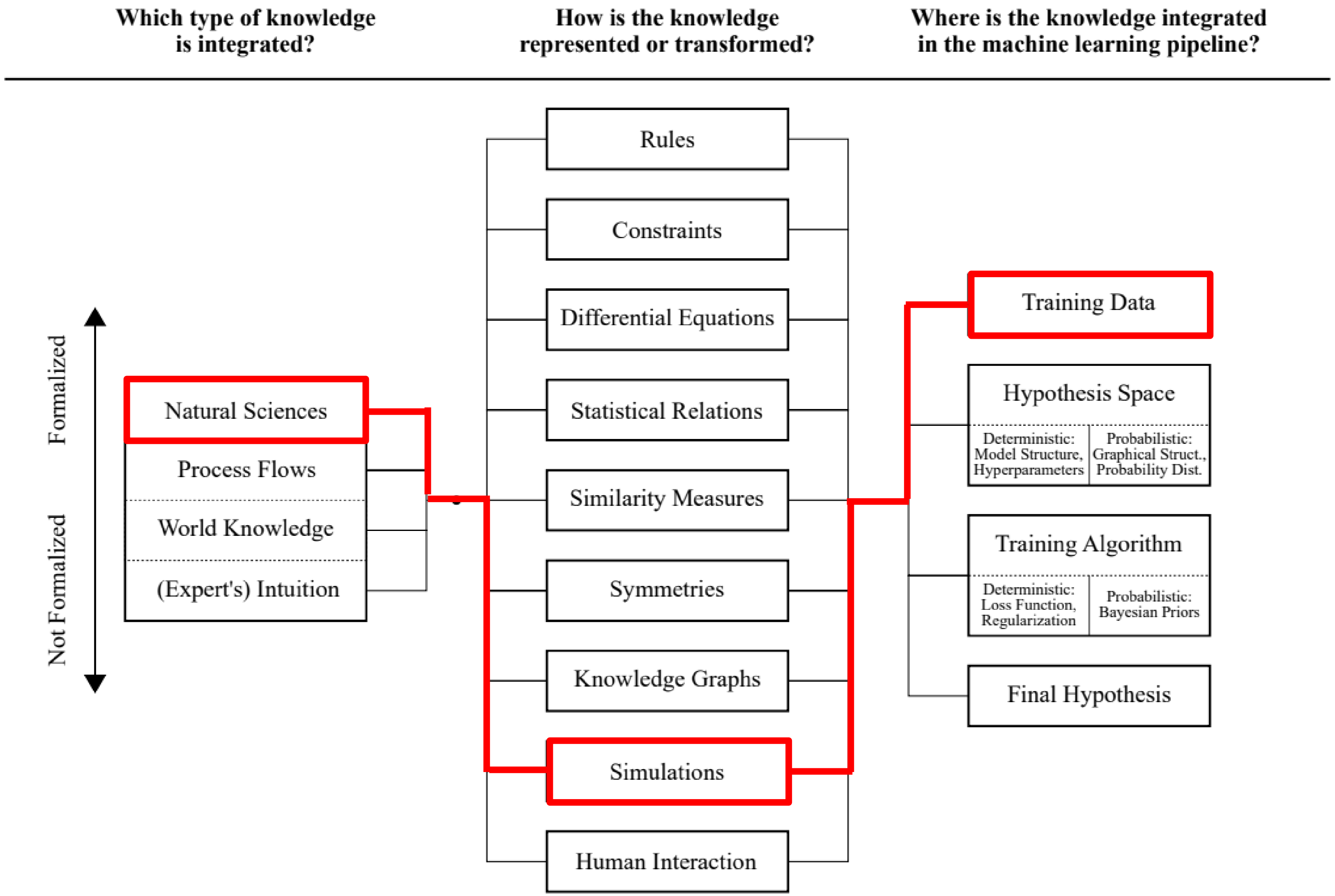
von Rued et al.

Data augmentation: geometrical invariance



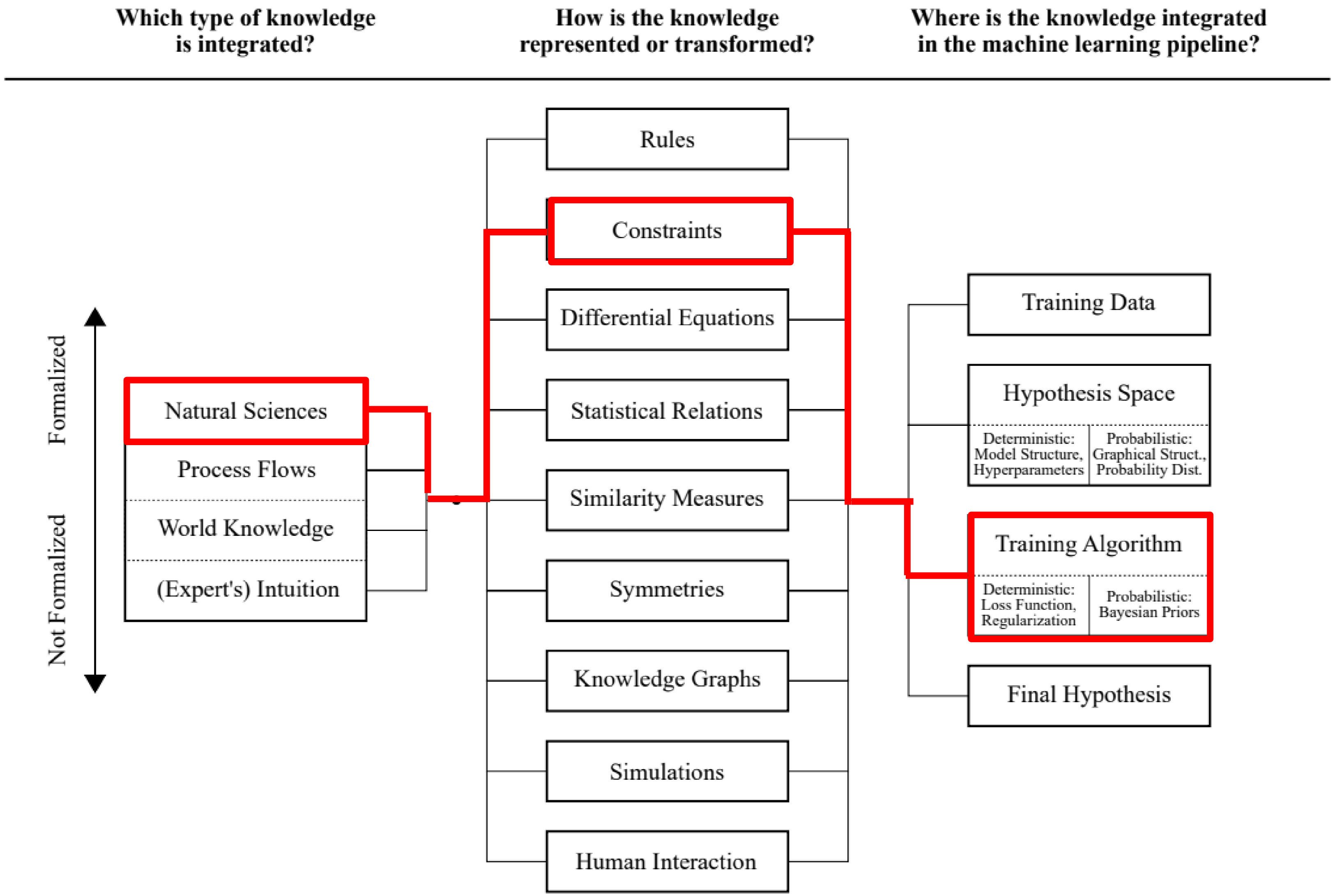
Credit https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

Data augmentation with simulations



von Rued et al.

Constraining the solution with the loss function



von Rued et al.

Physics guided Neural Network (Karpatne et al.)

- **Initial problem**
  - Approximate $f: X \rightarrow Y$ by a neural network f_{nn}
  - $\tilde{Y}$ is the predictor of f

- **Classical loss function**
  - $\underset{f_{nn}}{argmin}\, Loss(\tilde{Y}, Y) + \lambda R(f_{nn})$
  - no guarantee that model will produce results consistent with our knowledge of physics

Physics guided Neural Network (Karpatne et al.)

- **Initial problem**
  - Approximate $f: X \rightarrow Y$ by a neural network f_{nn}
  - $\tilde{Y}$ is the predictor of f

- **Classical loss function**
  - $\underset{f_{nn}}{argmin}\, Loss(\tilde{Y}, Y) + \lambda R(f_{nn})$
  - no guarantee that model will produce results consistent with our knowledge of physics

- **Modified loss function**
  - Known relationship between Y and other physical variables Z: $\begin{cases} G(Y,Z) = 0 \\ H(Y,Z) \leq 0 \end{cases}$
  - $L_{phy} = \left\| G\left(\tilde{Y}, Z\right) \right\|^2 + ReLu(H\left(\tilde{Y}, Z\right))$
  - New loss: $\underset{f_{nn}}{argmin}\, Loss(\tilde{Y}, Y) + \lambda R(f_{nn}) + \lambda_{phy} L_{phy}$

Physics guided Neural Network (Karpatne et al.)

- **Initial problem**
  - Approximate $f\colon X \to Y$ by a neural network f_{nn}
  - $\tilde{Y}$ is the predictor of f

- **Classical loss function**
  - $\underset{f_{nn}}{argmin}\, Loss(\tilde{Y}, Y) + \lambda R(f_{nn})$
  - no guarantee that model will produce results consistent with our knowledge of physics

- **Modified loss function**
  - Known relationship between Y and other physical variables Z: $\begin{cases} G(Y,Z) = 0 \\ H(Y,Z) \leq 0 \end{cases}$
  - $L_{phy} = \left\|G\left(\tilde{Y}, Z\right)\right\|^2 + ReLu(H\left(\tilde{Y}, Z\right))$
  - New loss: $\underset{f_{nn}}{argmin}\, Loss(\tilde{Y}, Y) + \lambda R(f_{nn}) + \boxed{\lambda_{phy} L_{phy}}$

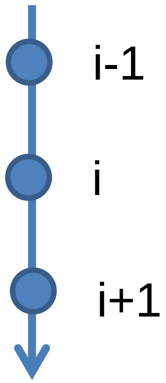
Lake temperature modeling (Karpatne et al.)

- **Finding temperature profile of lakes**
  - Temperature measurement at various depth and time
  - Various parameters: depth, air temperature, humidity, rain, is freezing, wind speed, etc.

Lake temperature modeling (Karpatne et al.)

- **Finding temperature profile of lakes**
  - Temperature measurement at various depth and time
  - Various parameters: depth, air temperature, humidity, rain, is freezing, wind speed, etc.

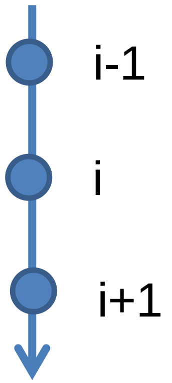
- **Incorporate the knowledge of physics**
  - Physical relationship between the temperature, density, and depth of water
  - Relationship between ρ and T is known
  - Density-depth relationship: ρ cannot decrease with depth d: $\Delta\rho(i,t) = \tilde{\rho}(d_i,t) - \tilde{\rho}(d_{i+1},t) \leq 0$



Lake temperature modeling (Karpatne et al.)

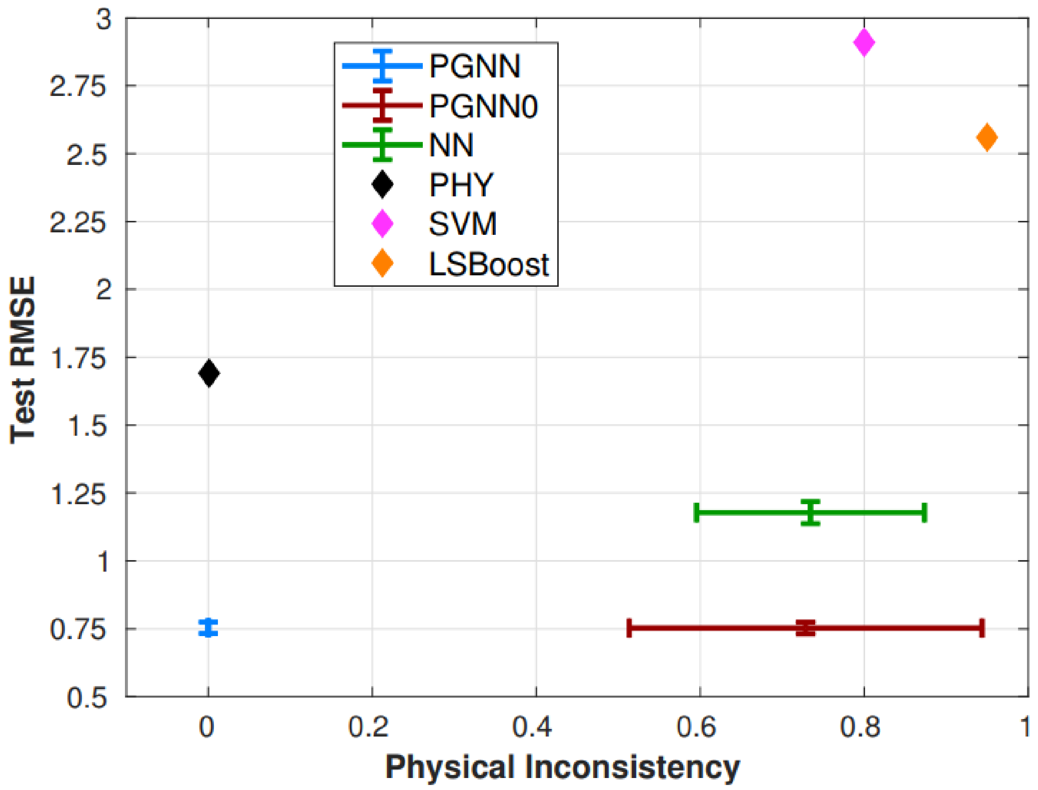
- **Finding temperature profile of lakes**
  - Temperature measurement at various depth and time
  - Various parameters: depth, air temperature, humidity, rain, is freezing, wind speed, etc.

- **Incorporate the knowledge of physics**
  - Physical relationship between the temperature, density, and depth of water
  - Relationship between ρ and T is known
  - Density-depth relationship: ρ cannot decrease with depth d: $\Delta\rho(i,t) = \tilde{\rho}(d_i, t) - \tilde{\rho}(d_{i+1}, t) \leq 0$

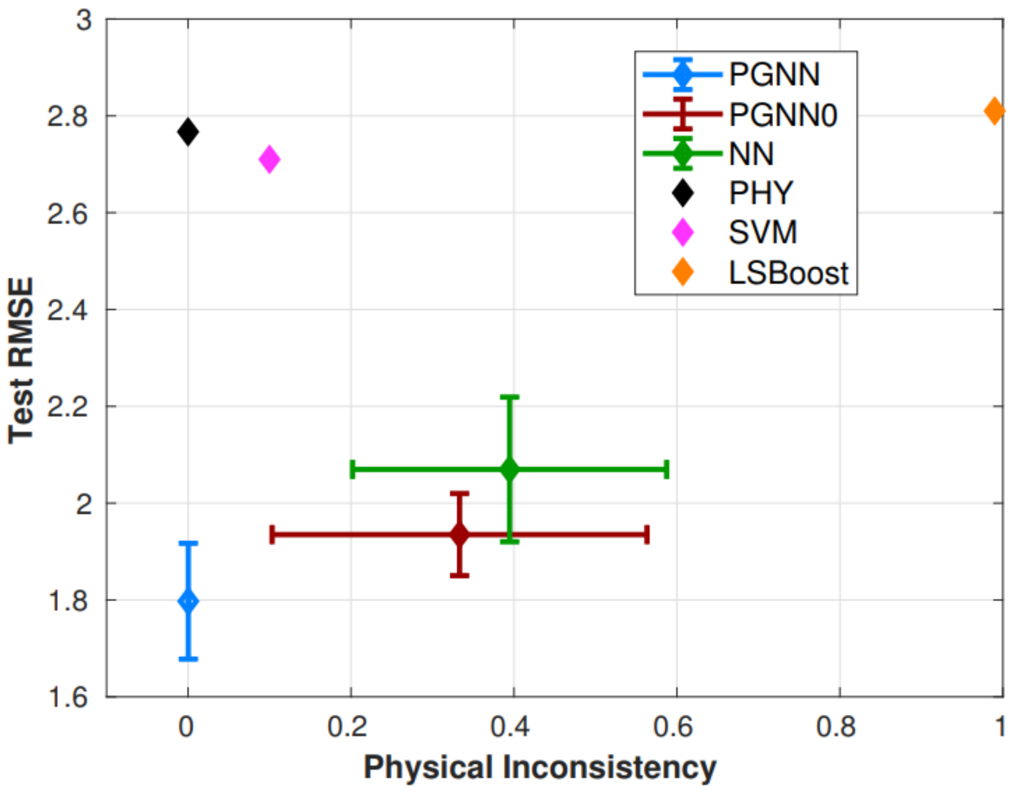


$$\boxed{L_{phy} = \frac{1}{n_t(n_d - 1)} \sum_{t=1}^{n_t} \sum_{i=1}^{n_d - 1} ReLU(\Delta\rho(i,t))}$$

Lake temperature modeling (Karpatne et al.)

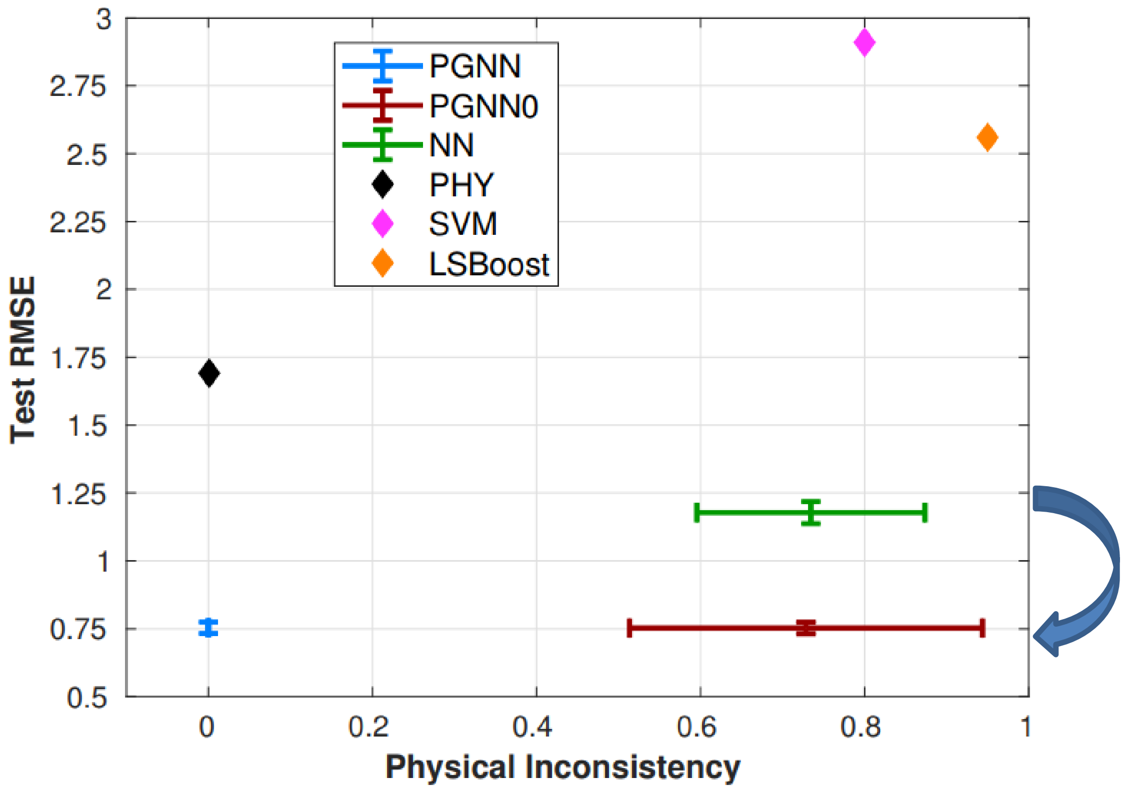


(a) Results on Lake Mille Lacs

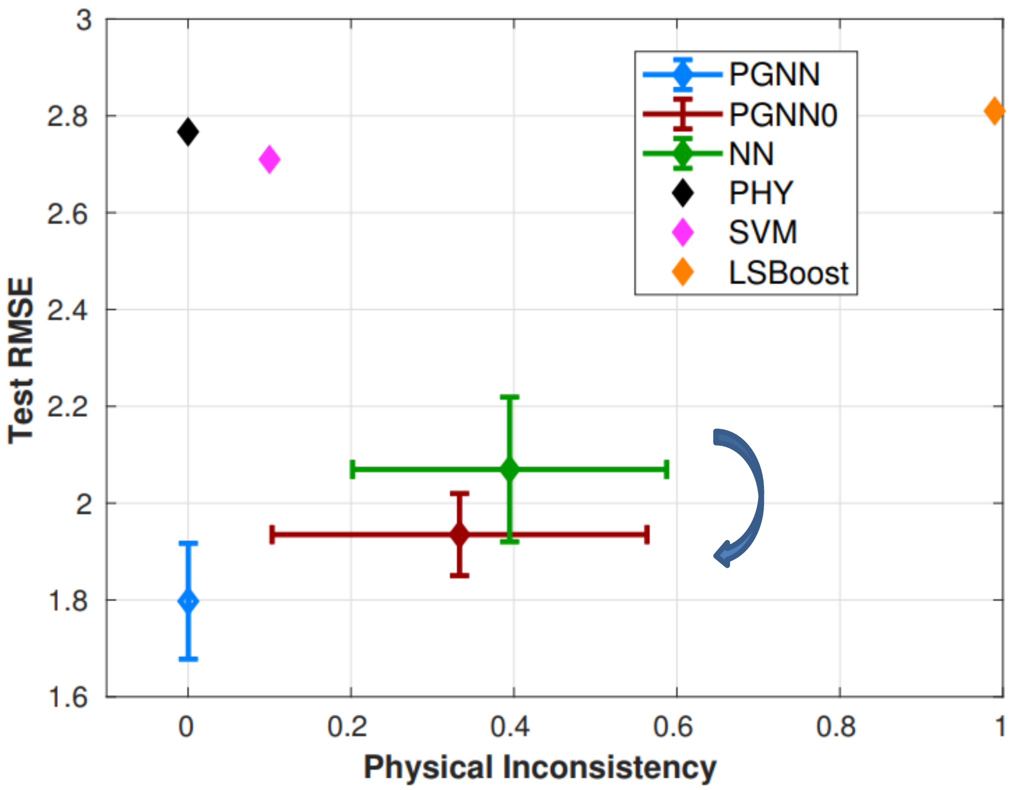


(b) Results on Lake Mendota

Lake temperature modeling (Karpatne et al.)

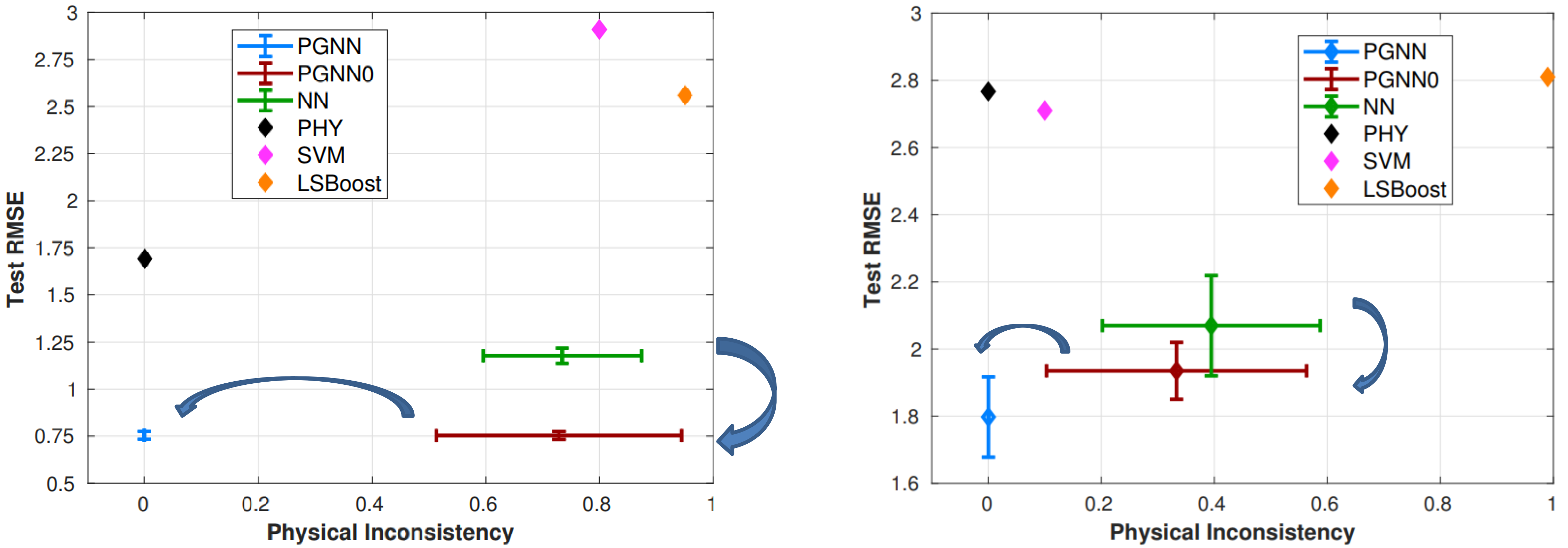


(a) Results on Lake Mille Lacs



(b) Results on Lake Mendota

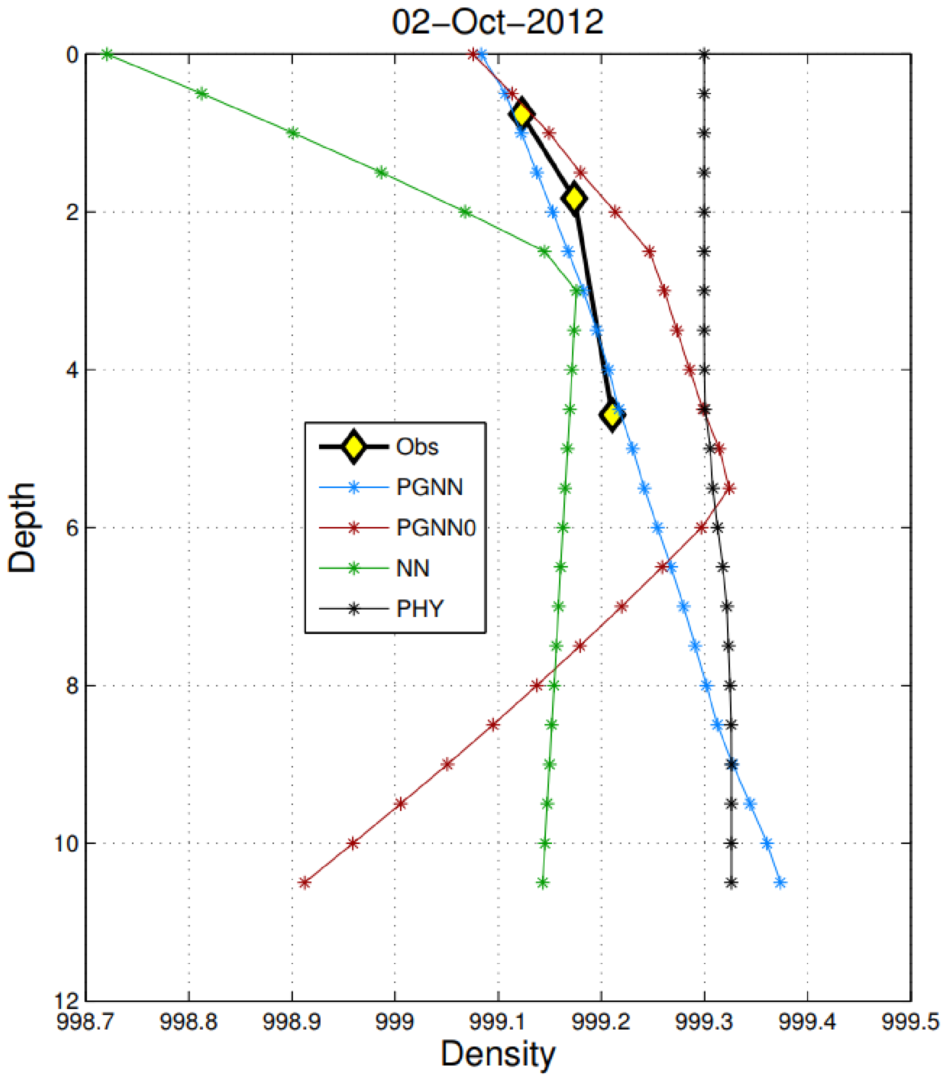
Lake temperature modeling (Karpatne et al.)



(a) Results on Lake Mille Lacs

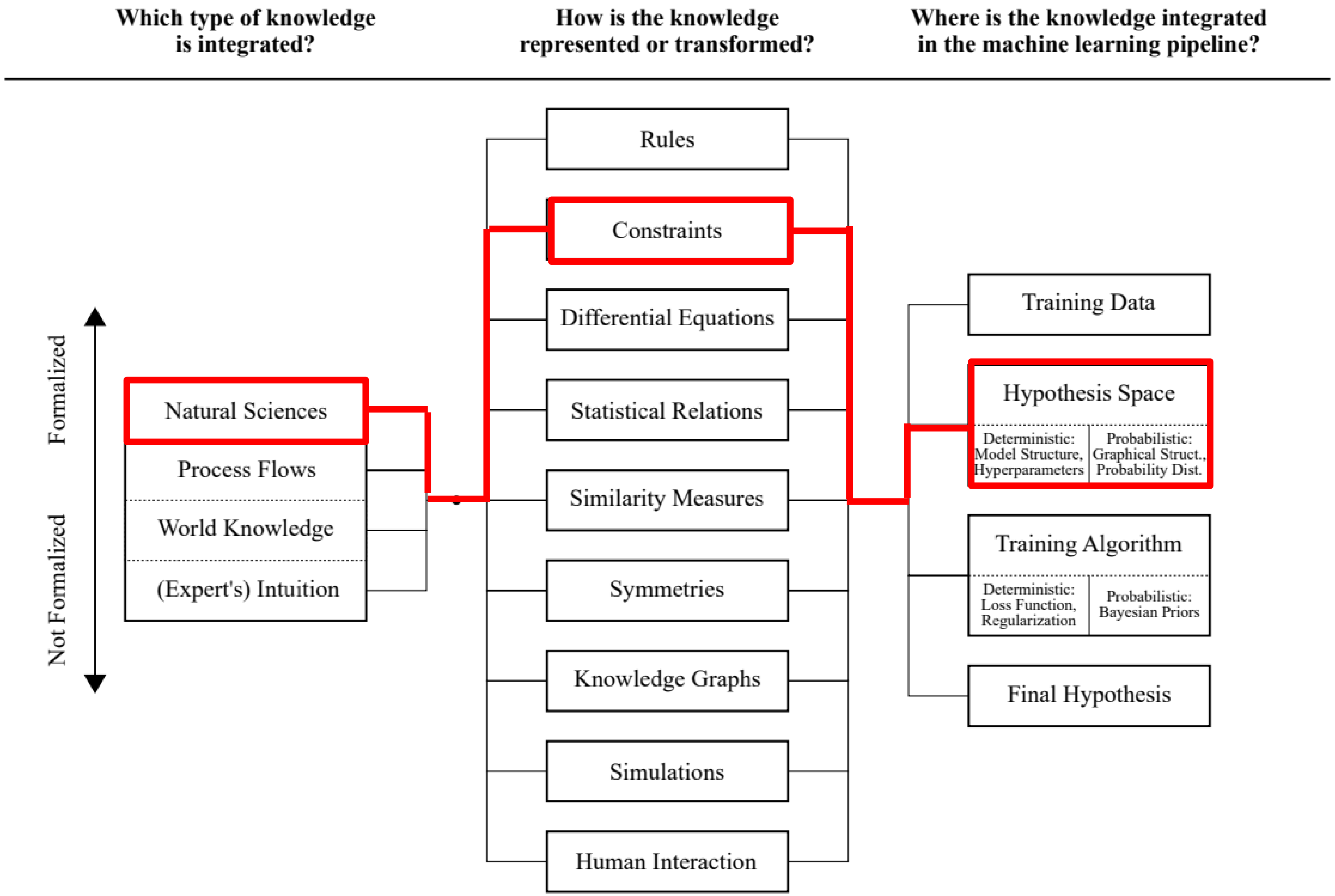
(b) Results on Lake Mendota

Lake temperature modeling (Karpatne et al.)



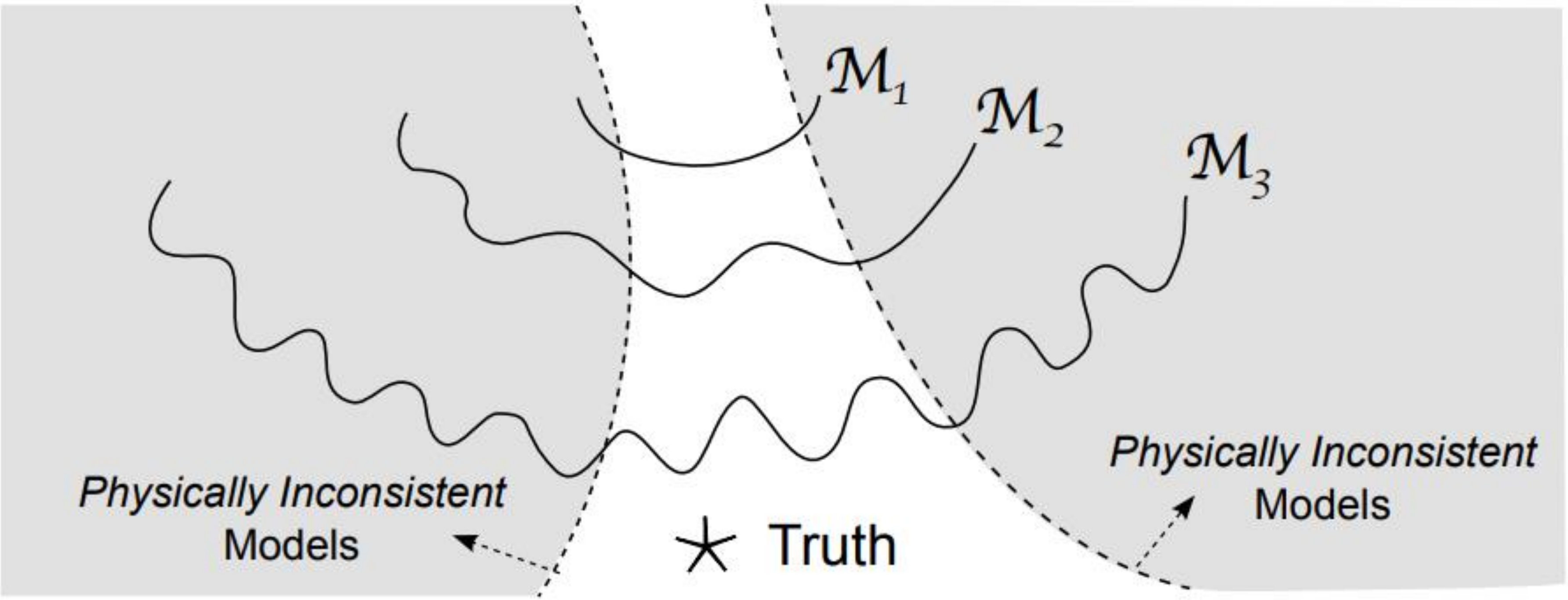
(a) Lake Mille Lacs on 02-October-2012

Limiting the model space



von Rued et al.

Limiting the model space

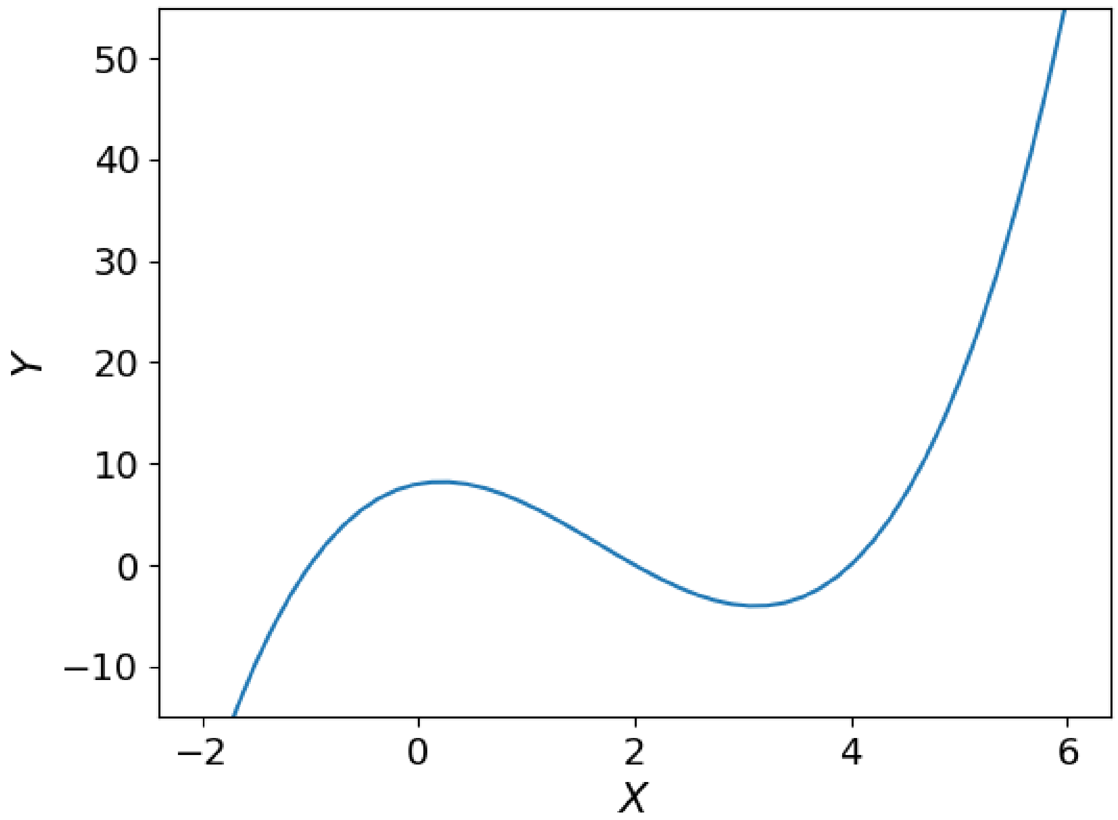


Karpatne et al

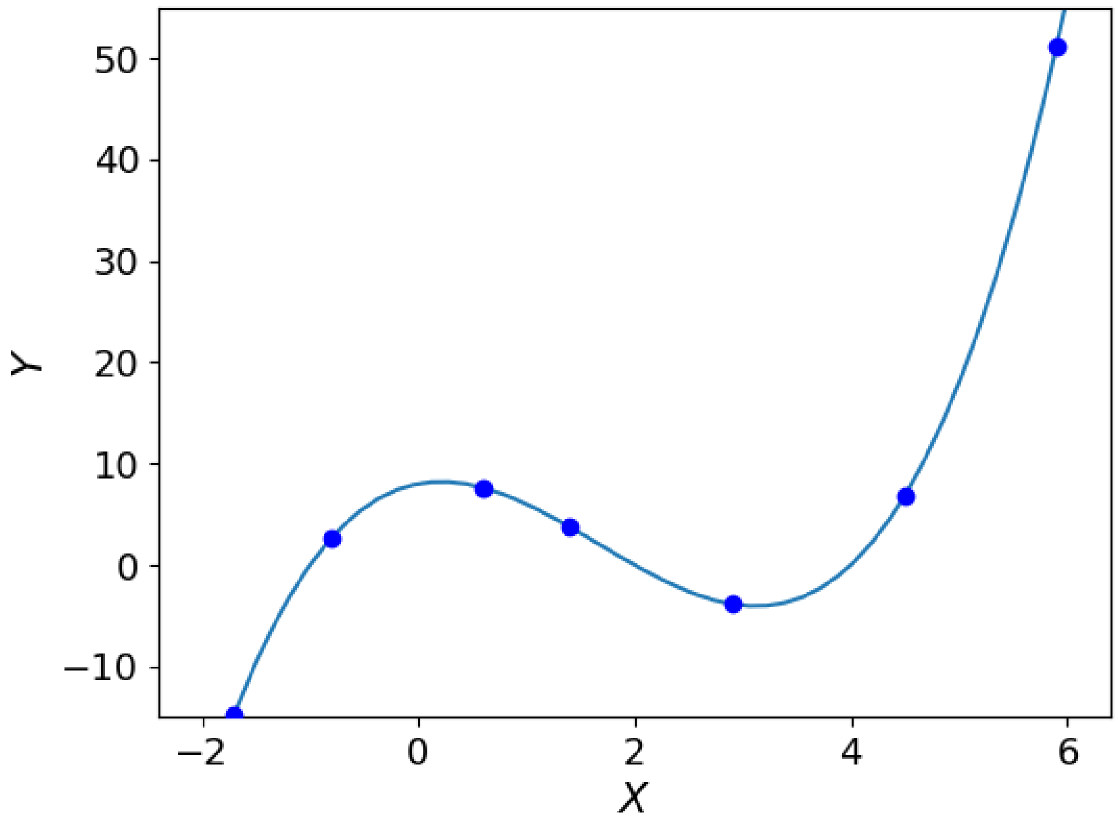
Scientific knowledge can reduce the model variance
- removing physically inconsistent solutions
- without likely affecting their bias.

Gaussian Process Regression

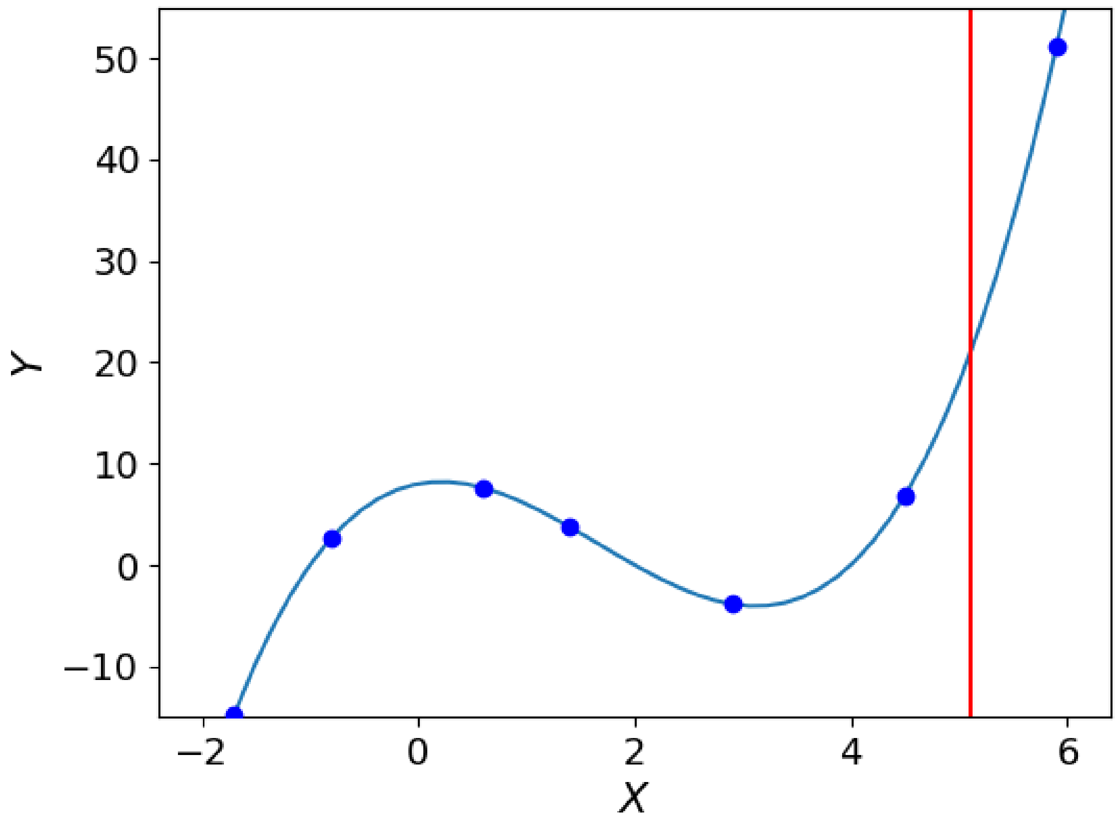
Gaussian Process Regression



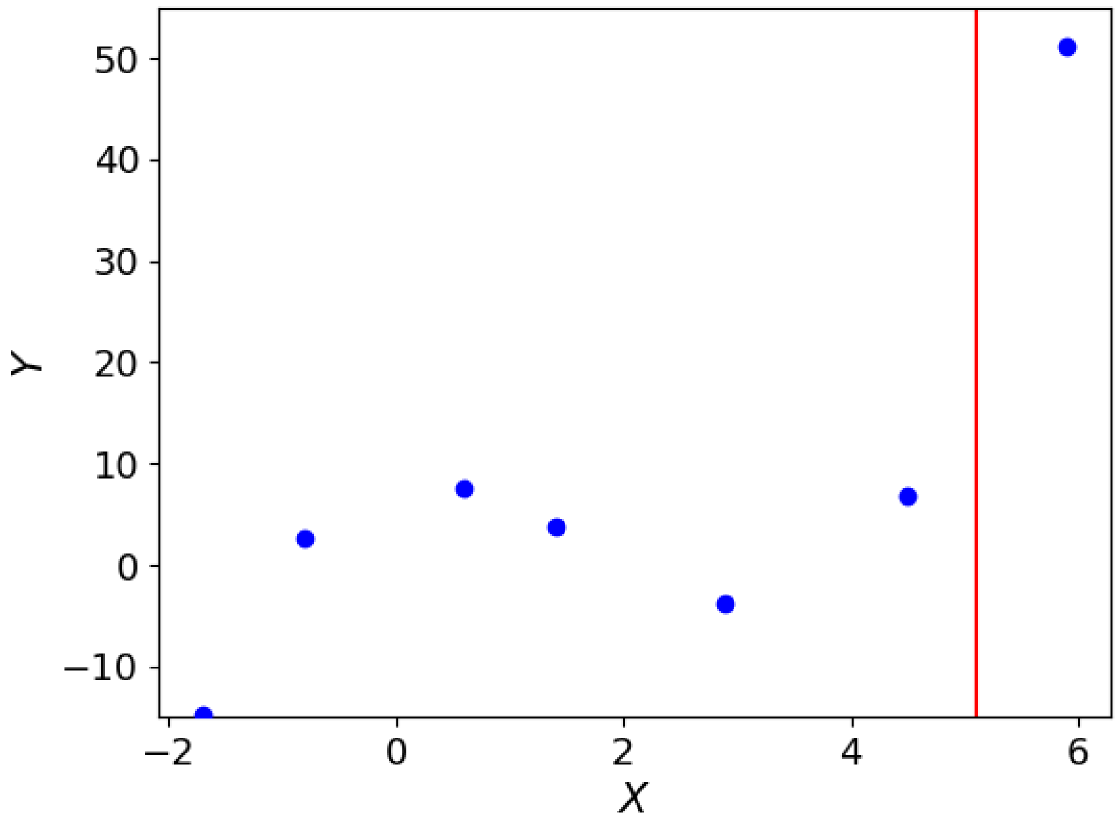
Gaussian Process Regression



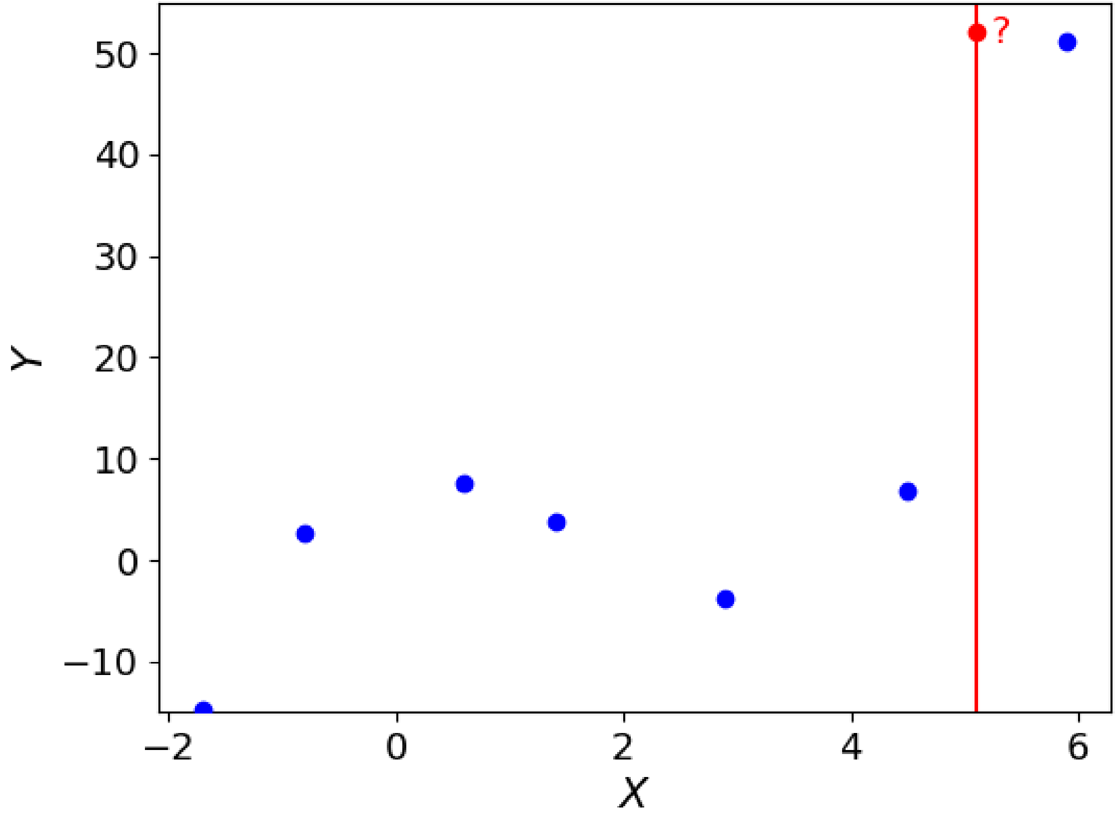
Gaussian Process Regression



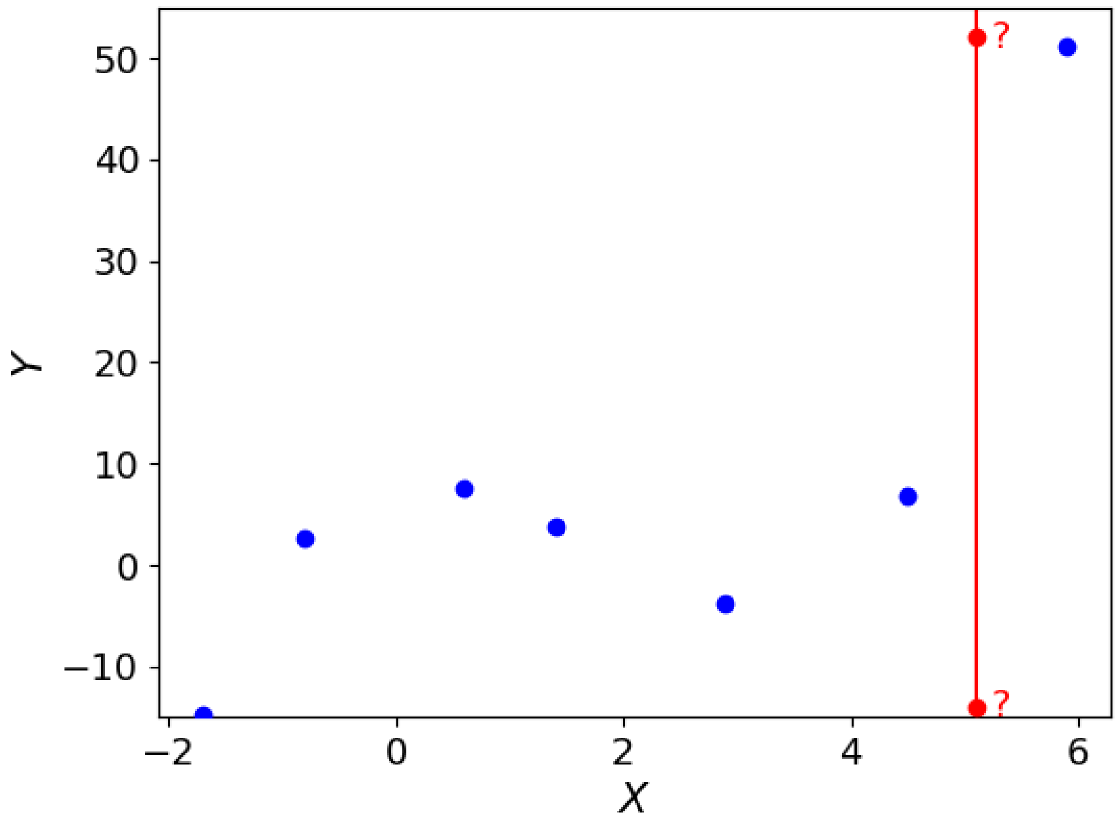
Gaussian Process Regression



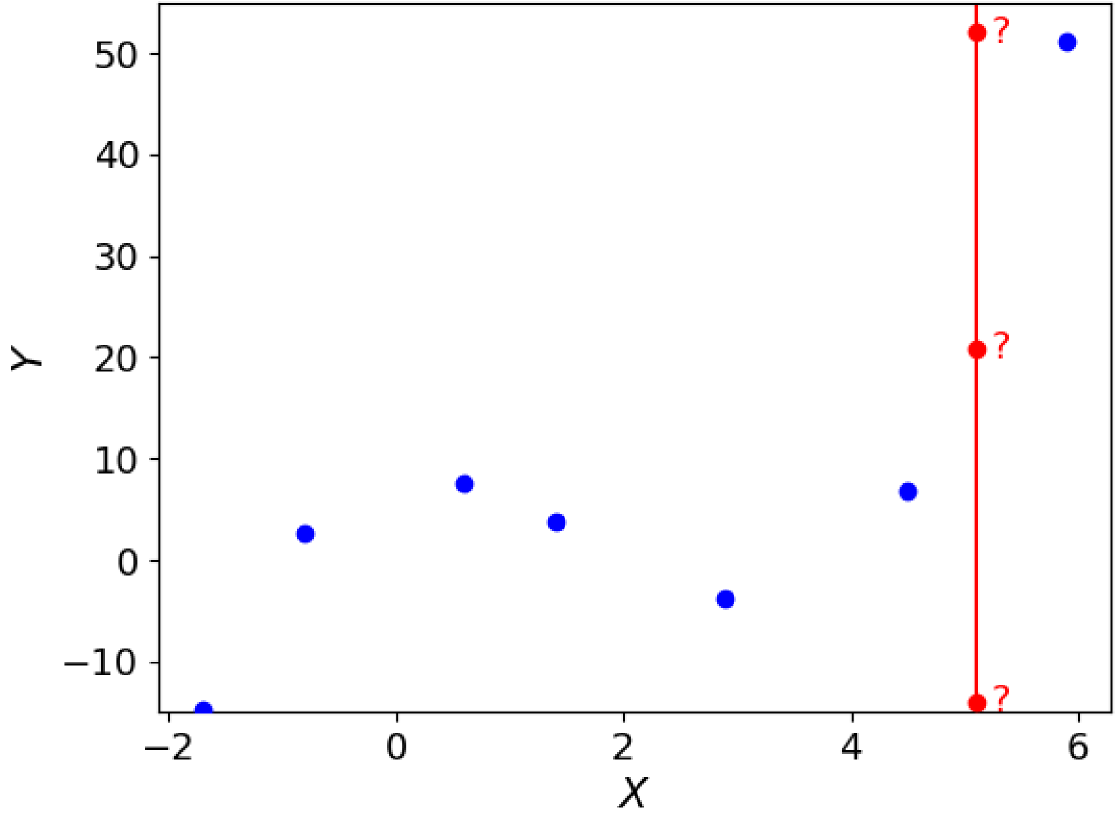
Gaussian Process Regression



Gaussian Process Regression



Gaussian Process Regression



Gaussian Process Regression

- **Context**
  - Function to learn $f : \mathbb{R}^d \rightarrow \mathbb{R}$
  - Given set of inputs $\mathbf{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ associated to $y_1, \dots, y_n$ such as $y_i = f(\boldsymbol{x}_i)$

- **Definition of a Gaussian Process**
  - All values $\left(f(\boldsymbol{x}_1), \dots, f(\boldsymbol{x}_n)\right)$ are normally distributed
  - Each value corresponds to a component of a n-dimensional Gaussian

Gaussian Process Regression

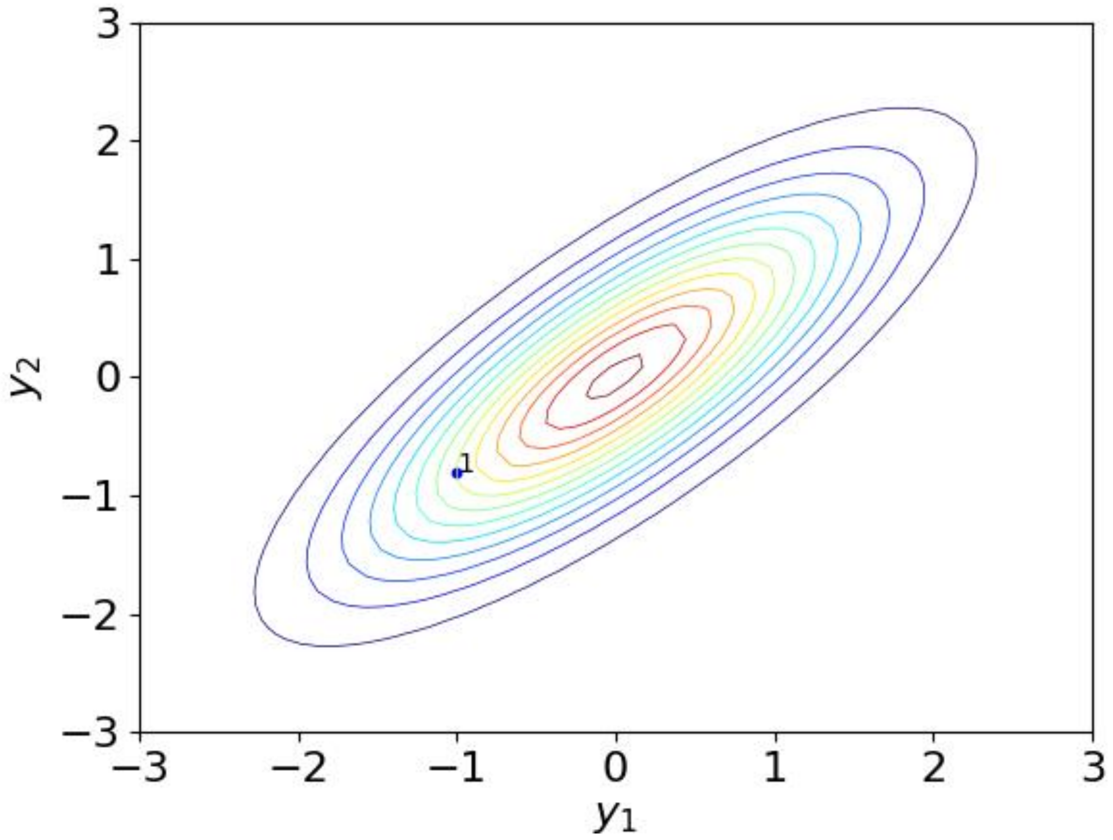
- **Context**
  - Function to learn $f : \mathbb{R}^d \rightarrow \mathbb{R}$
  - Given set of inputs $\mathbf{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ associated to $y_1, \dots, y_n$ such as $y_i = f(\boldsymbol{x}_i)$

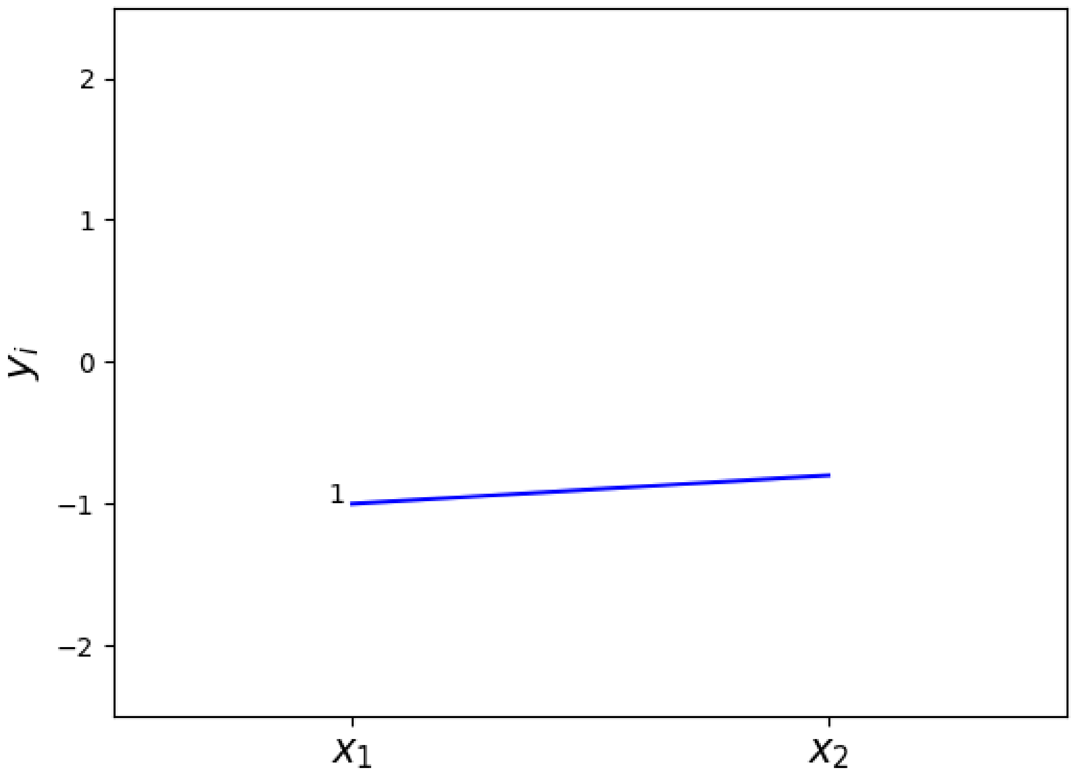
- **Definition of a Gaussian Process**
  - All values $\left(f(\boldsymbol{x}_1), \dots, f(\boldsymbol{x}_n)\right)$ are normally distributed
  - Each value corresponds to a component of a n-dimensional Gaussian

- **Full definition**
  - Mean : $\mathbb{E}[f(\boldsymbol{x})] = \mu(\boldsymbol{x})$ usually can be 0
  - Covariance function $Cov\left(f(\boldsymbol{x}), f(\boldsymbol{x}')\right) = k(\boldsymbol{x}, \boldsymbol{x}')$ with k a function called kernel
  - $\mathrm{f}(\mathbf{x}) \sim GP(0, k(\boldsymbol{x}, \boldsymbol{x}'))$

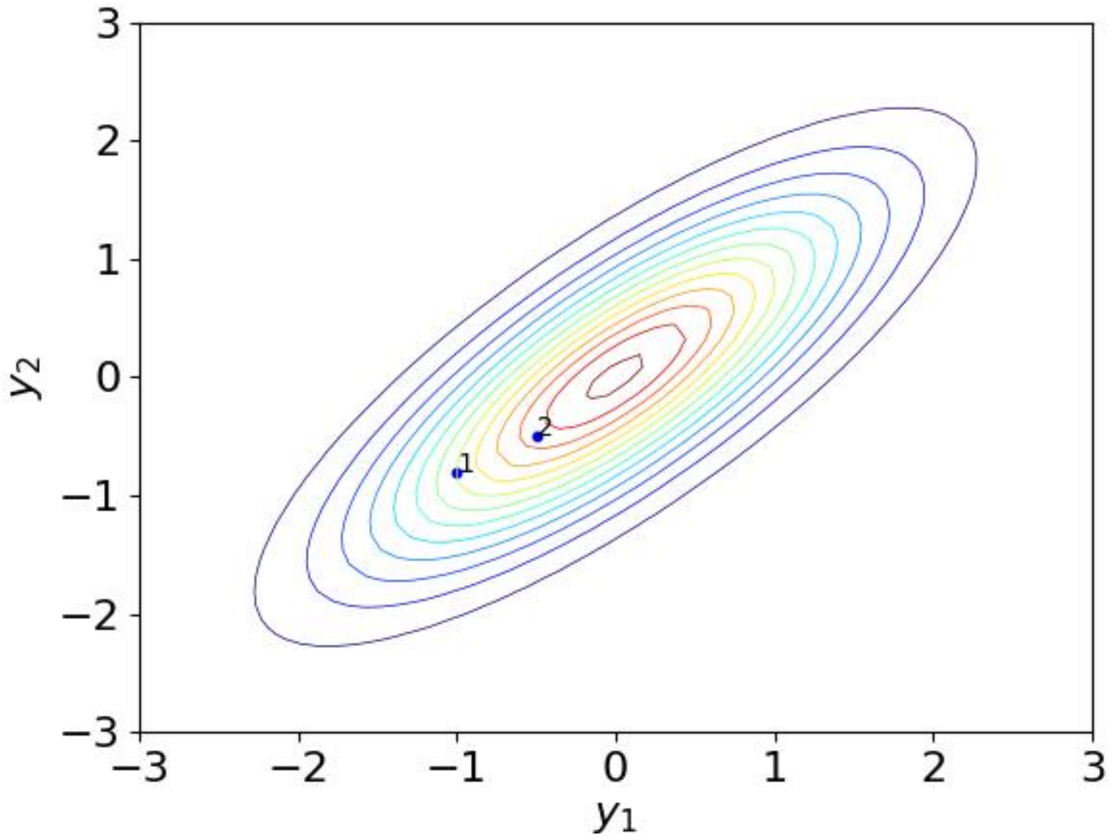
$$\begin{bmatrix} f(\boldsymbol{x}) \\ f(\boldsymbol{x}') \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}) & k(\boldsymbol{x}, \boldsymbol{x}') \\ k(\boldsymbol{x}', \boldsymbol{x}) & k(\boldsymbol{x}', \boldsymbol{x}') \end{bmatrix}\right.$$

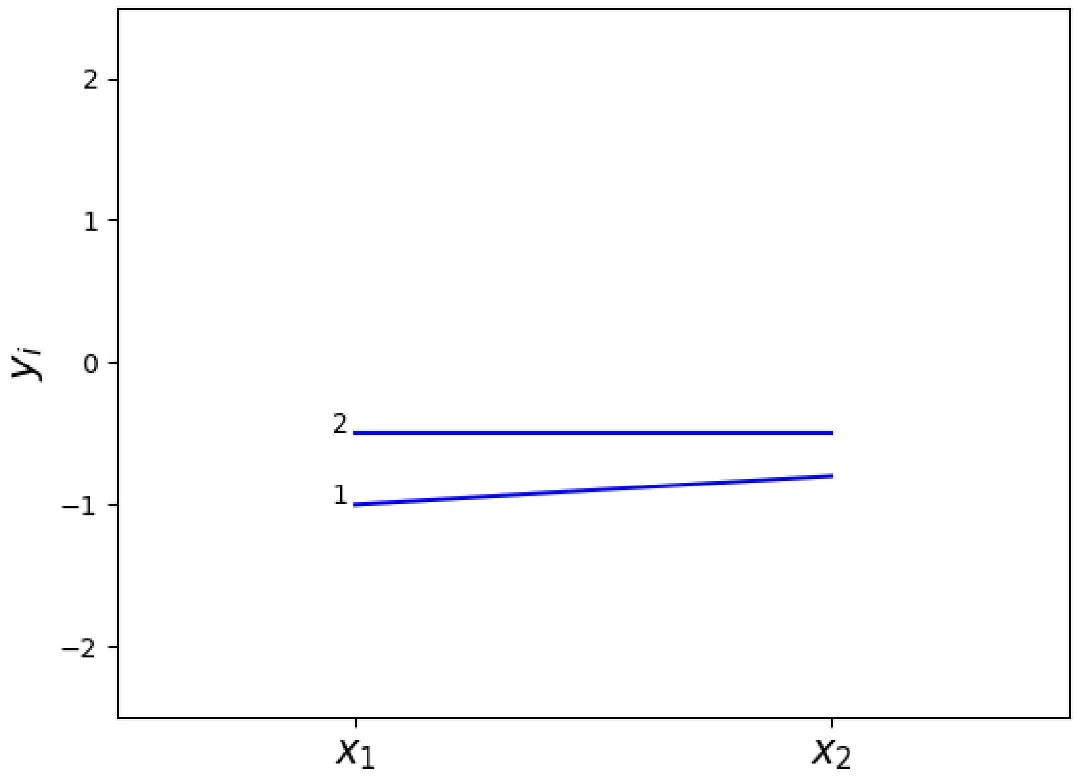
Gaussian Process Regression: 2D example



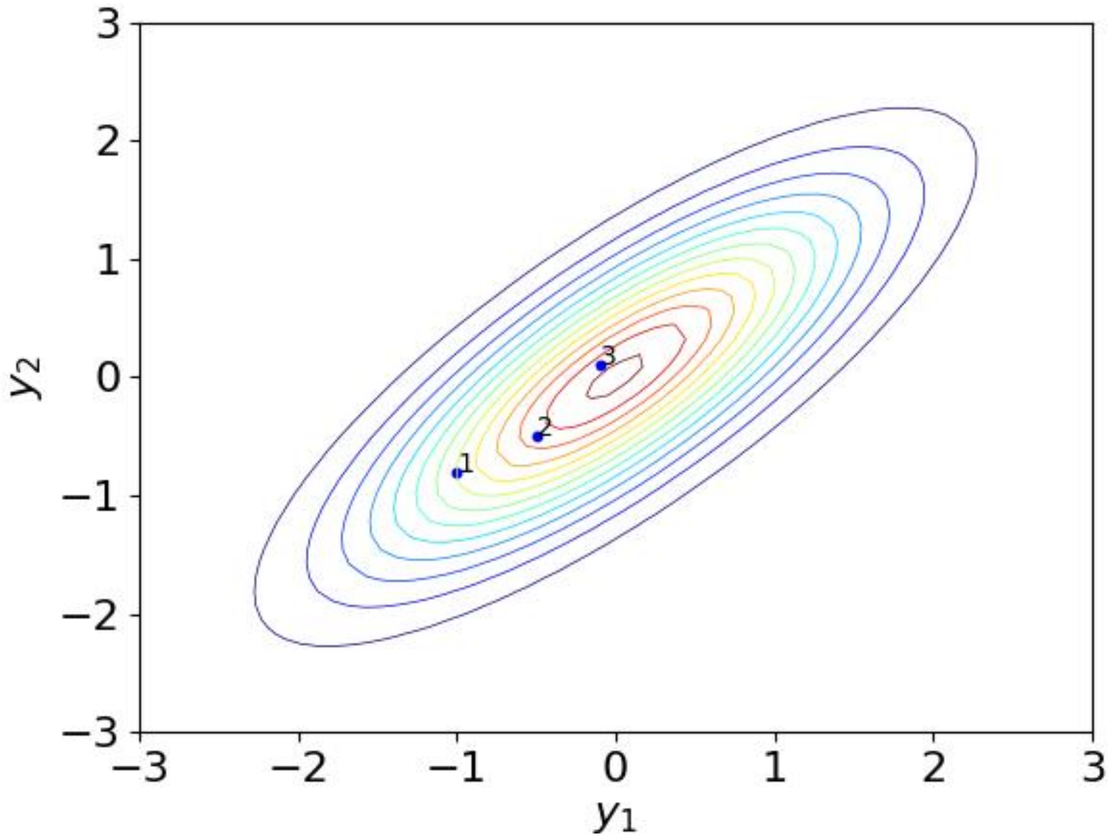


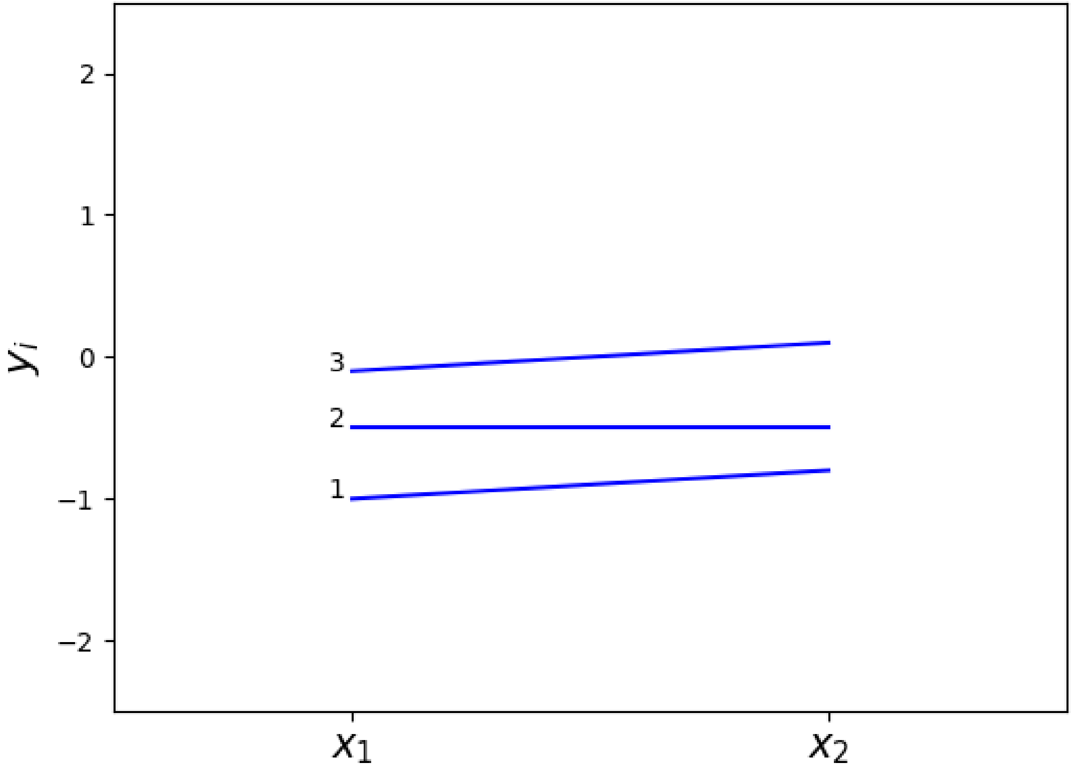
Gaussian Process Regression: 2D example



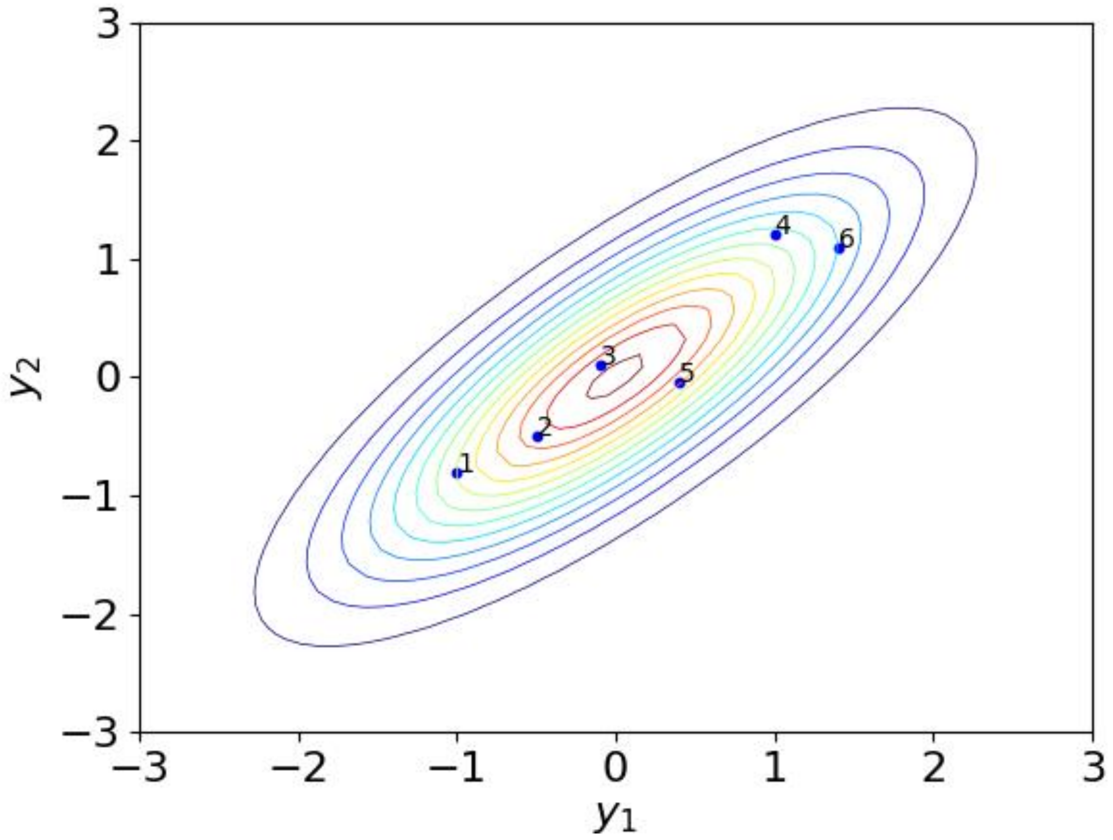


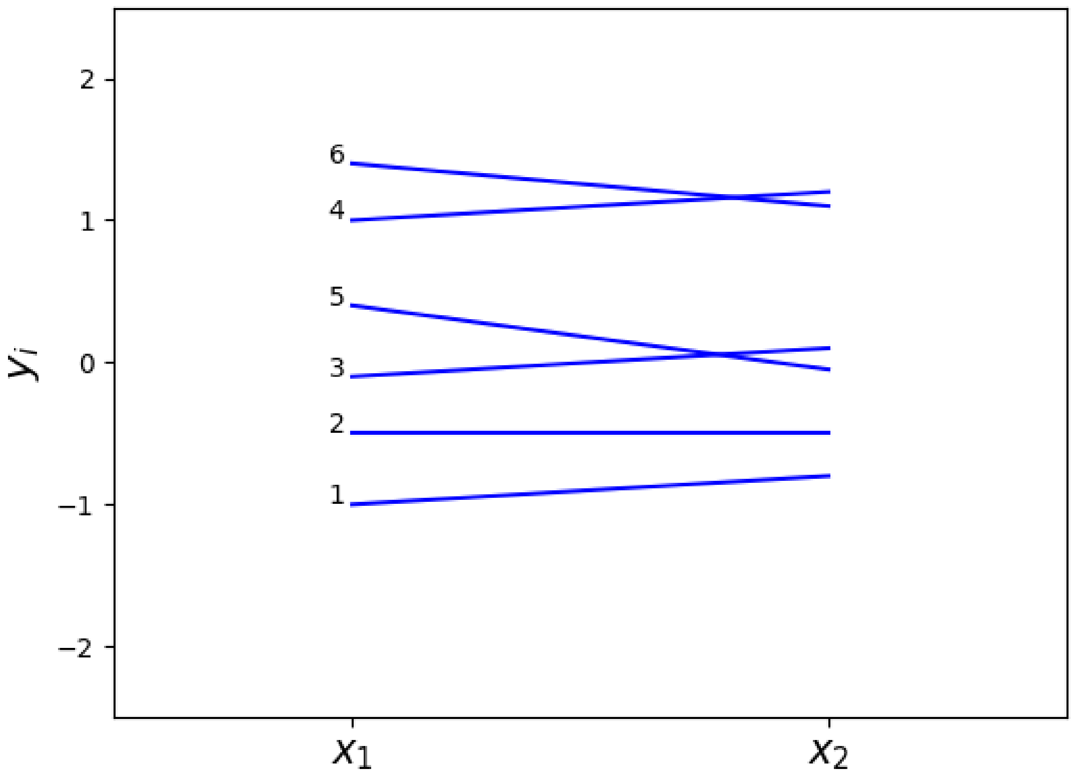
Gaussian Process Regression: 2D example



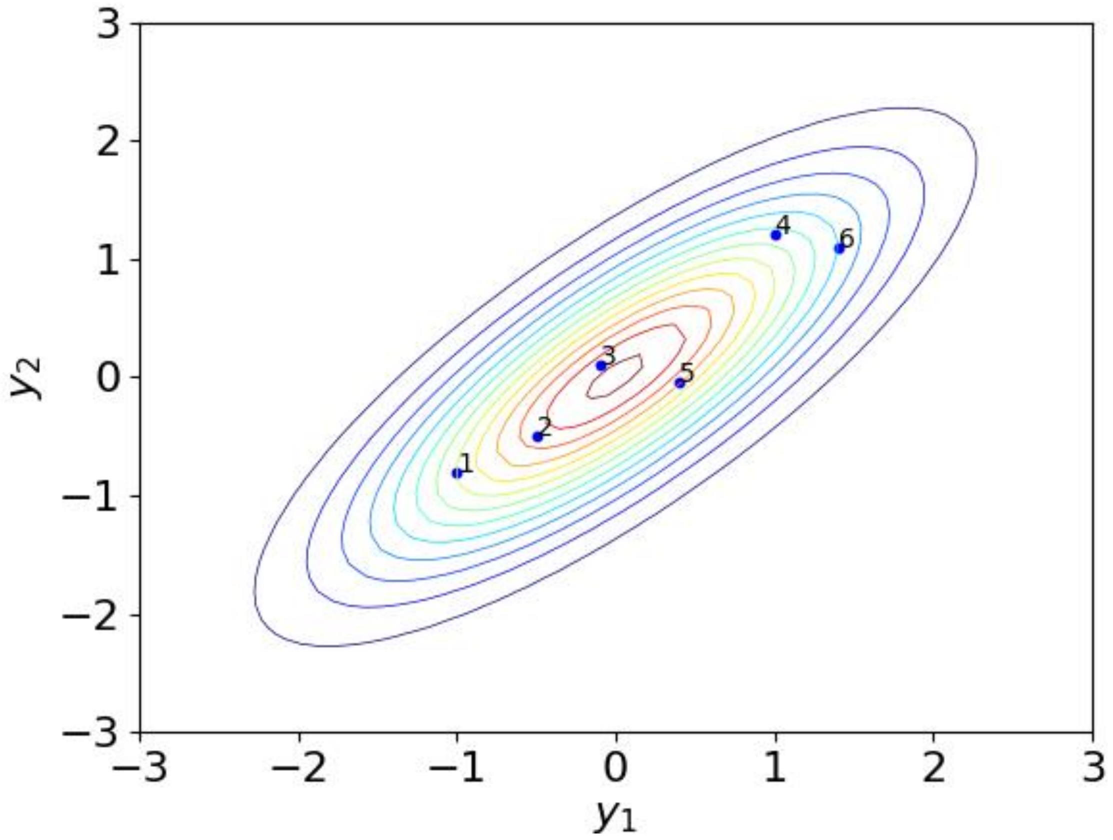


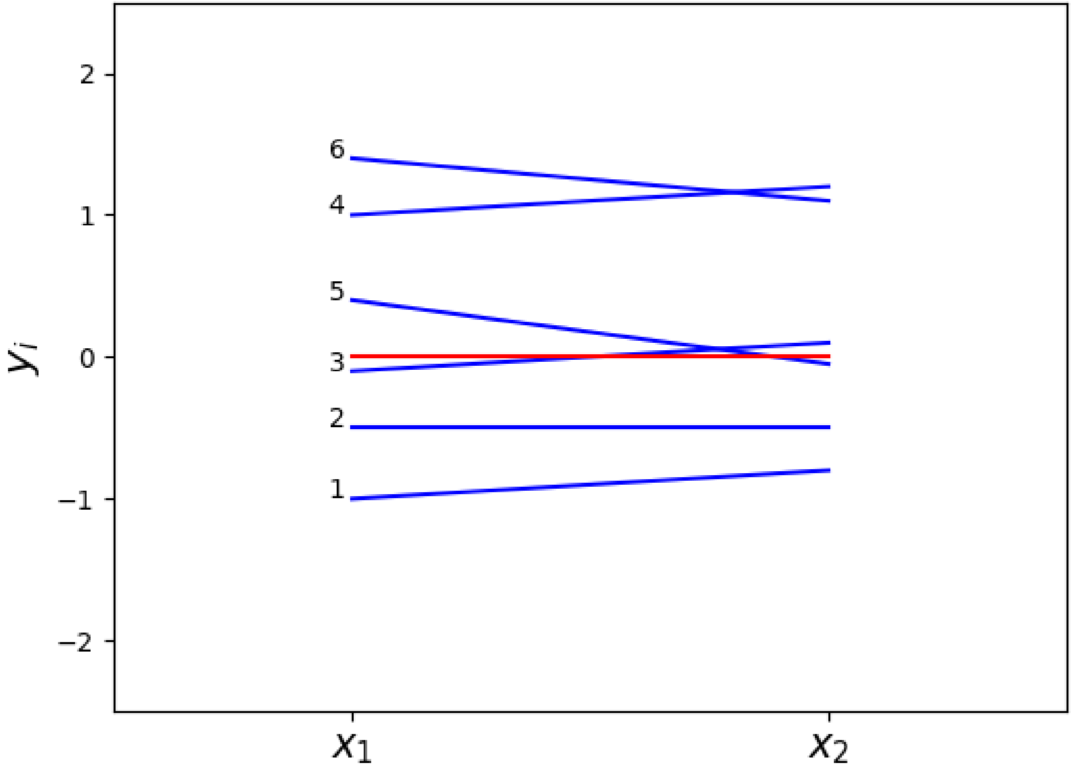
Gaussian Process Regression: 2D example



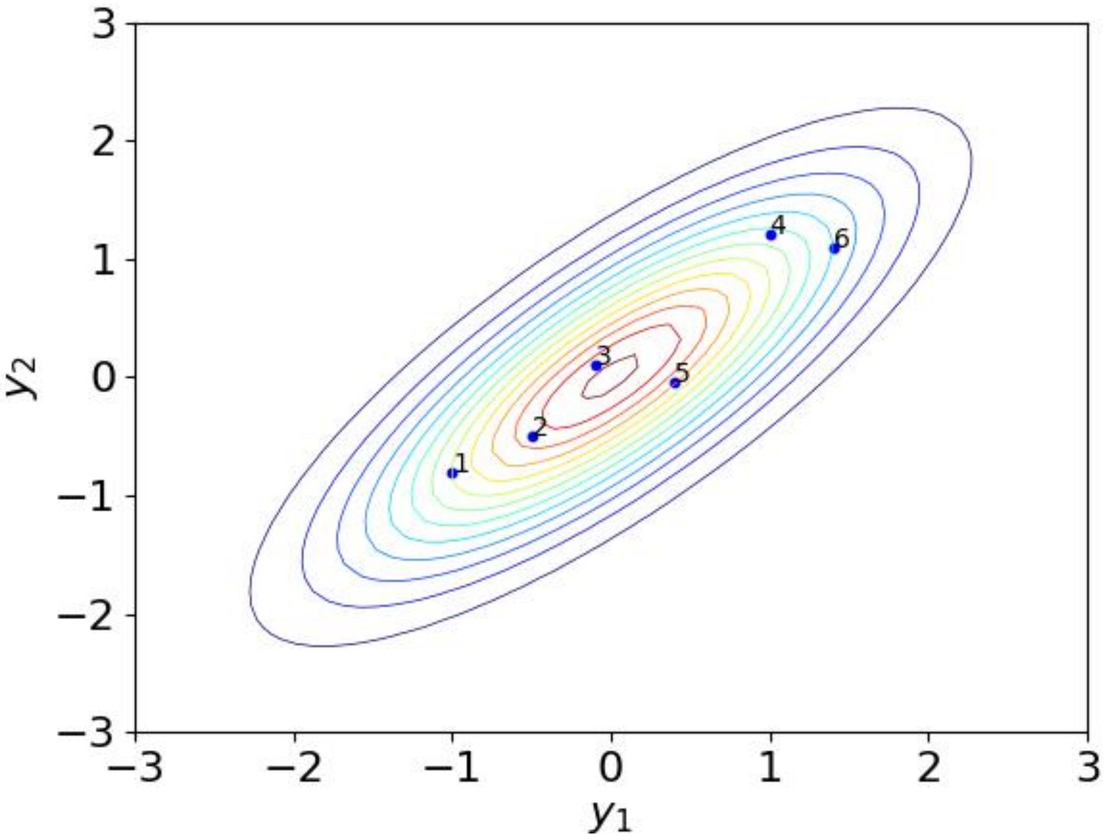


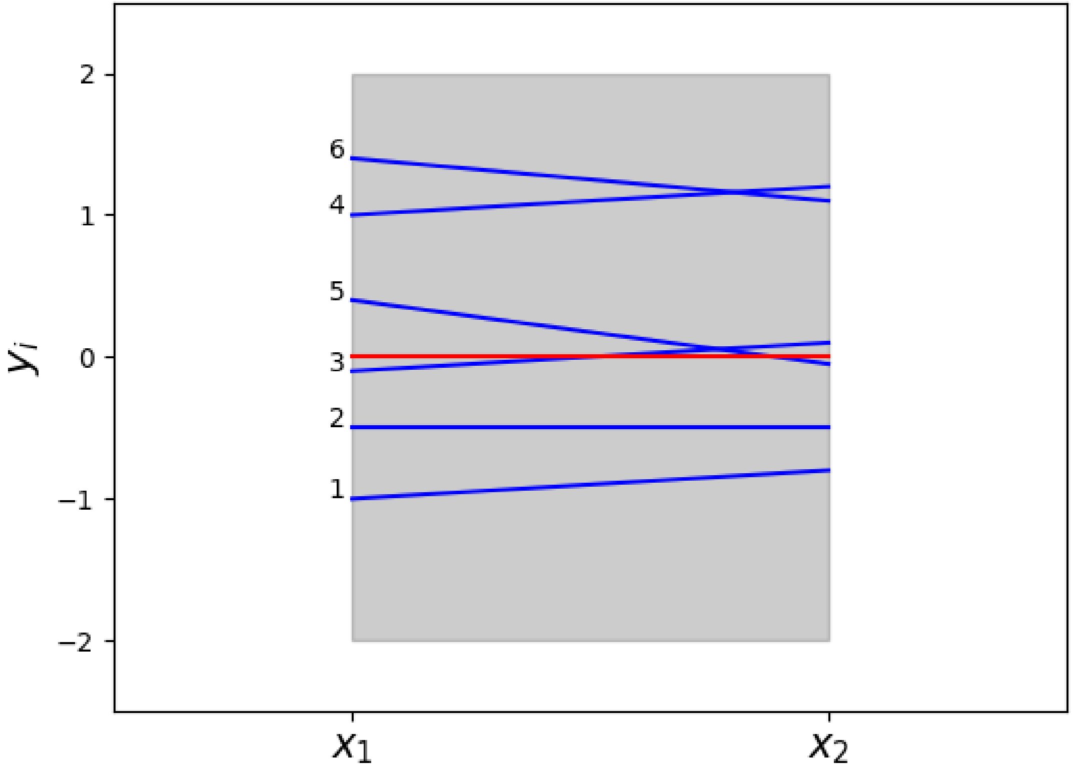
Gaussian Process Regression: 2D example



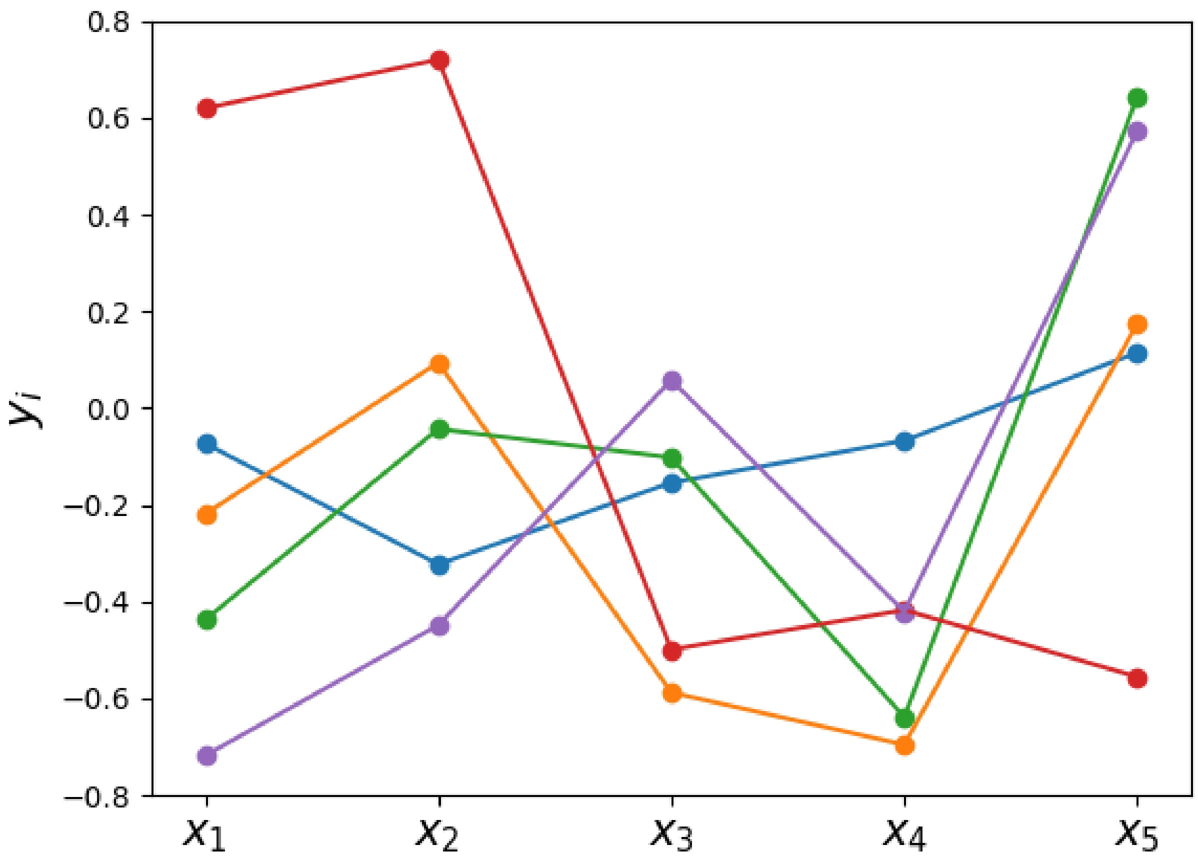


Gaussian Process Regression: 2D example

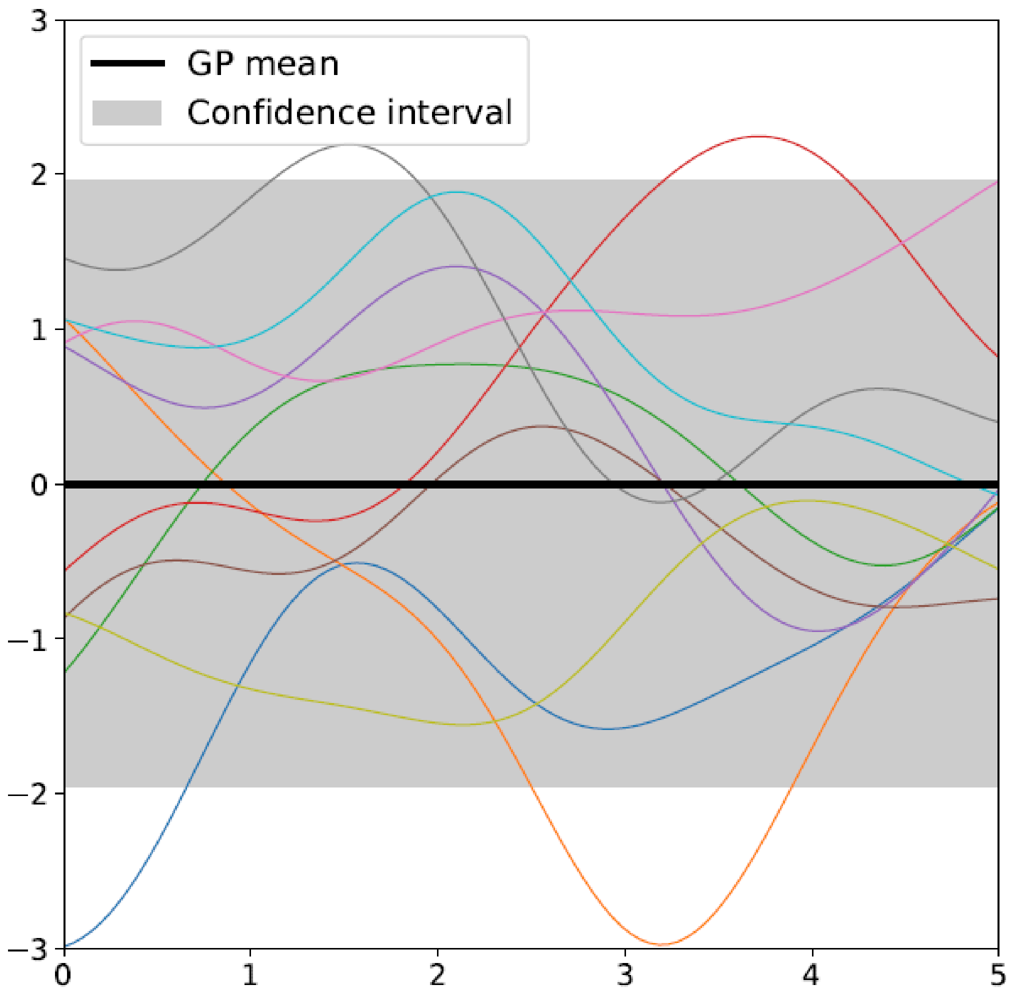




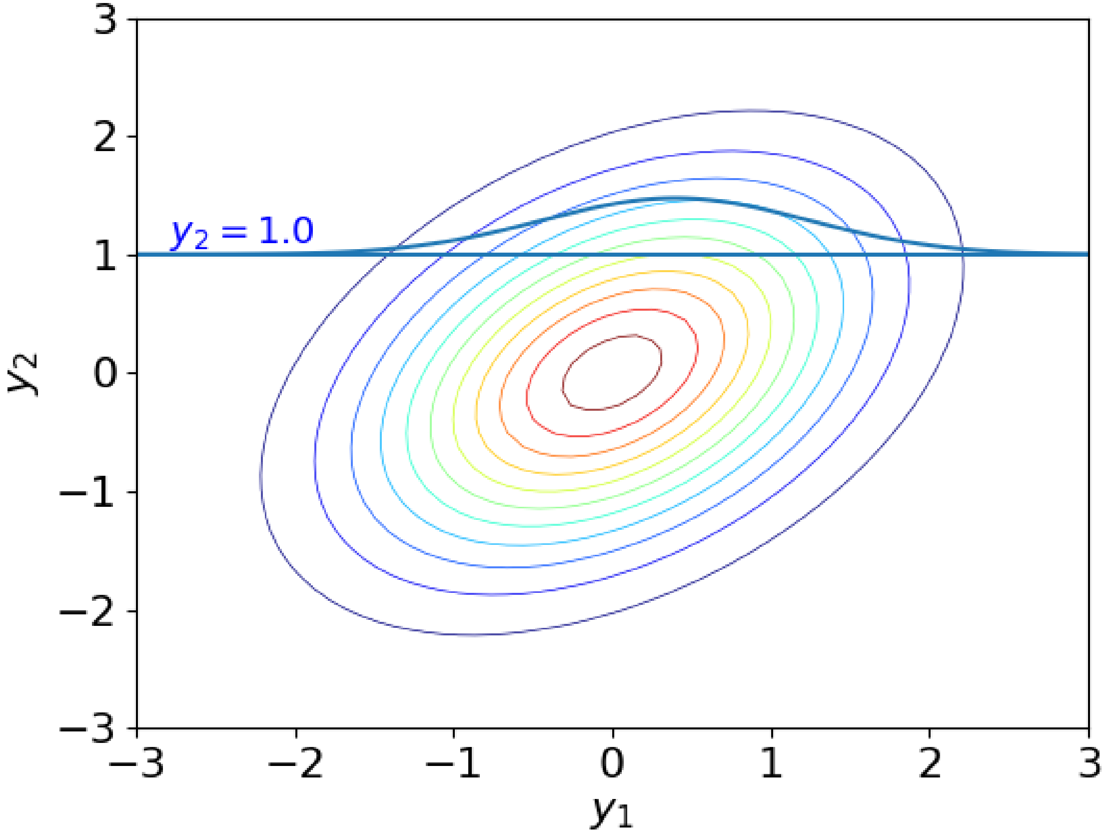
Gaussian Process Regression: 5D example



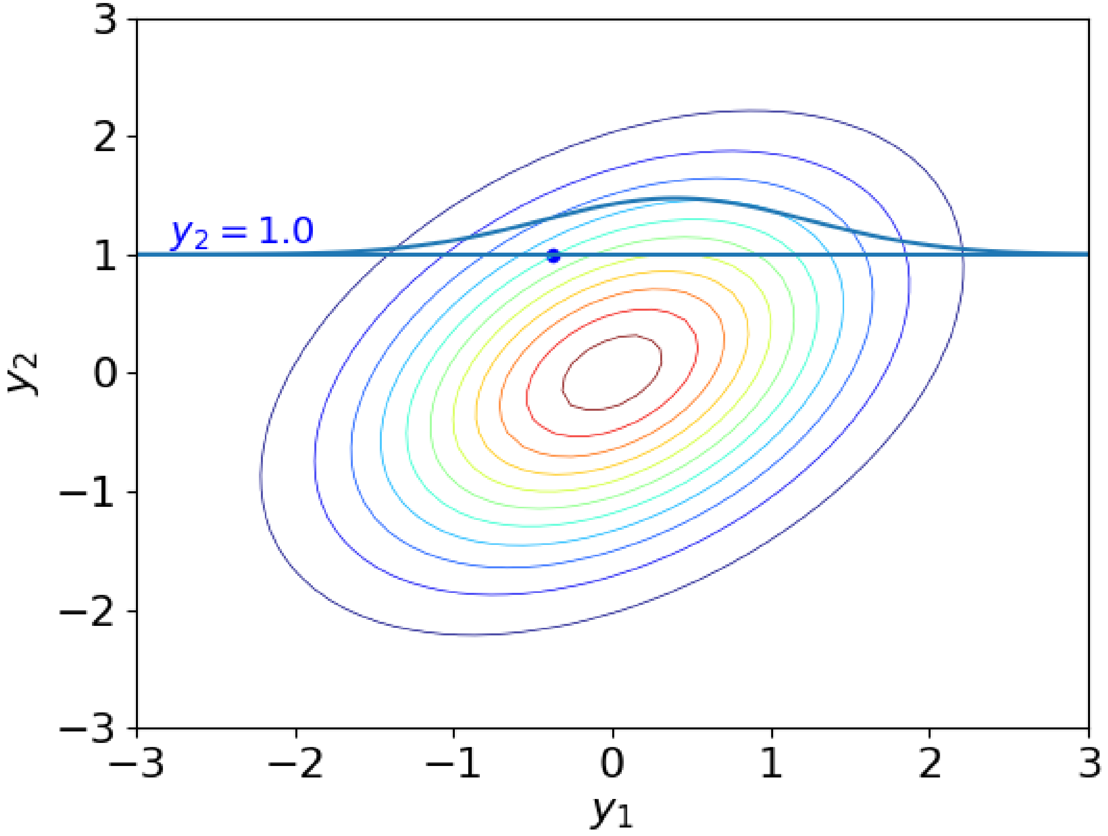
Gaussian Process Regression: infinite dimension

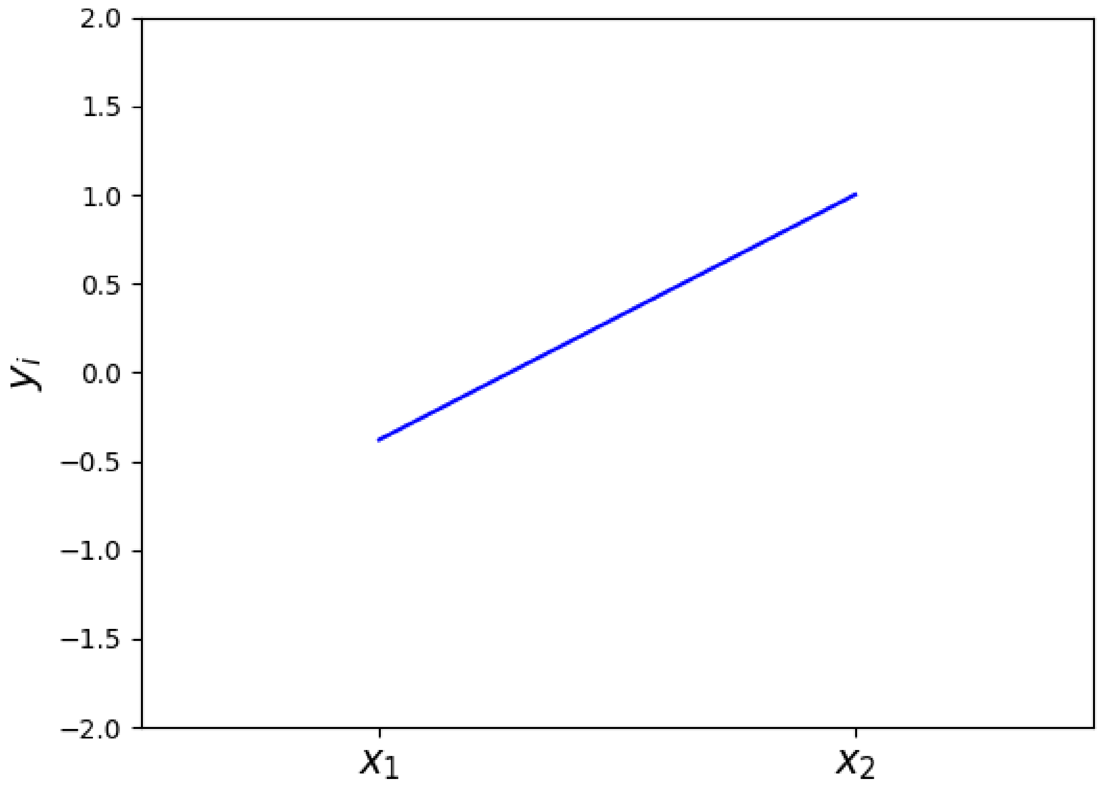


Gaussian Process Regression: conditional sampling

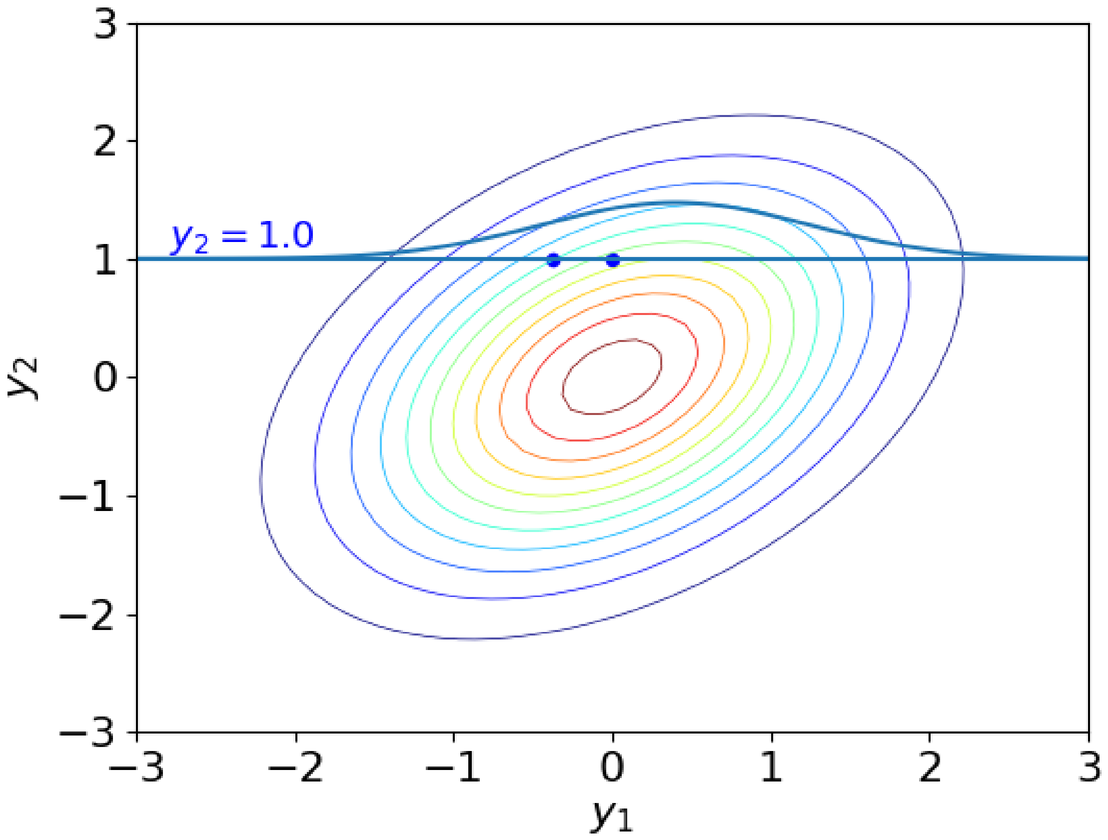


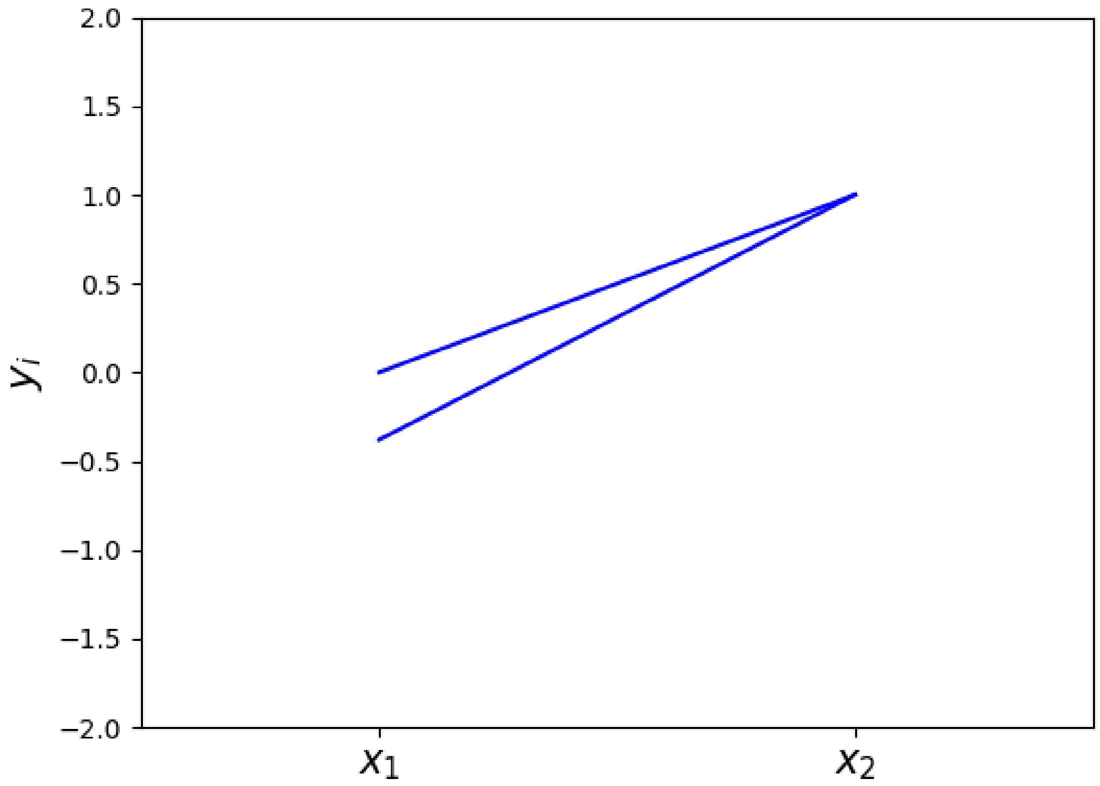
Gaussian Process Regression: conditional sampling



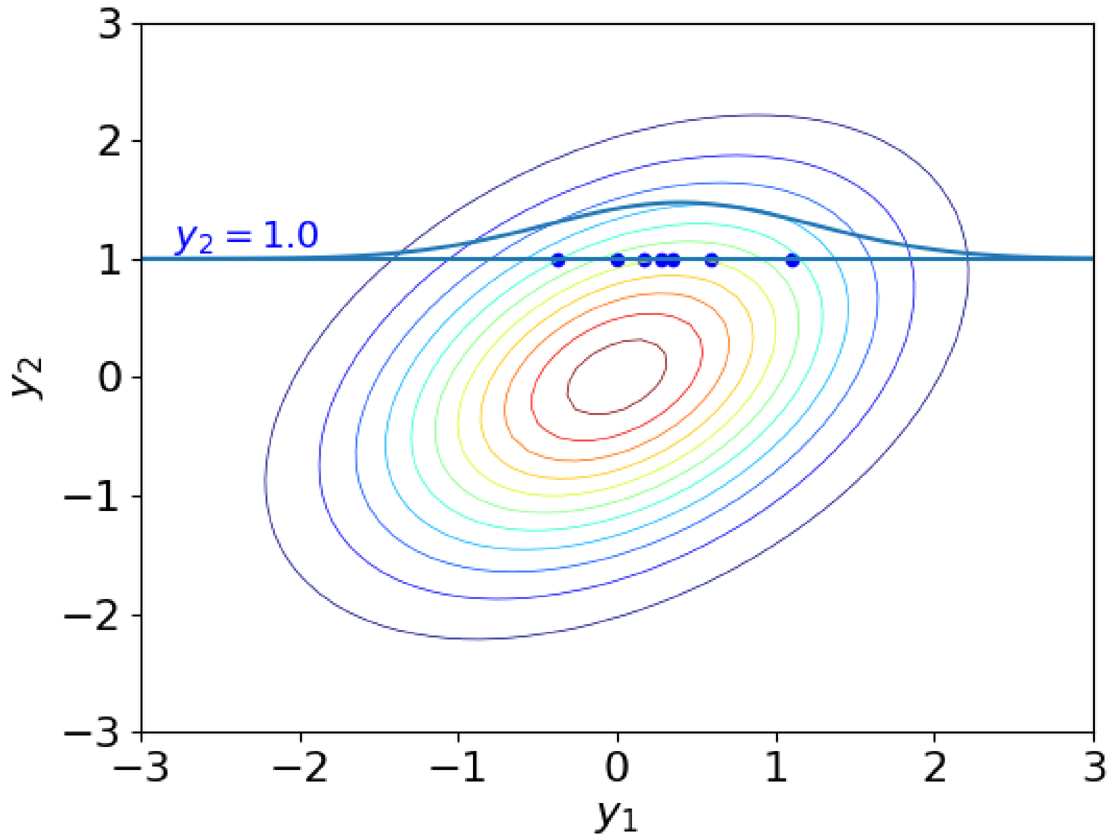


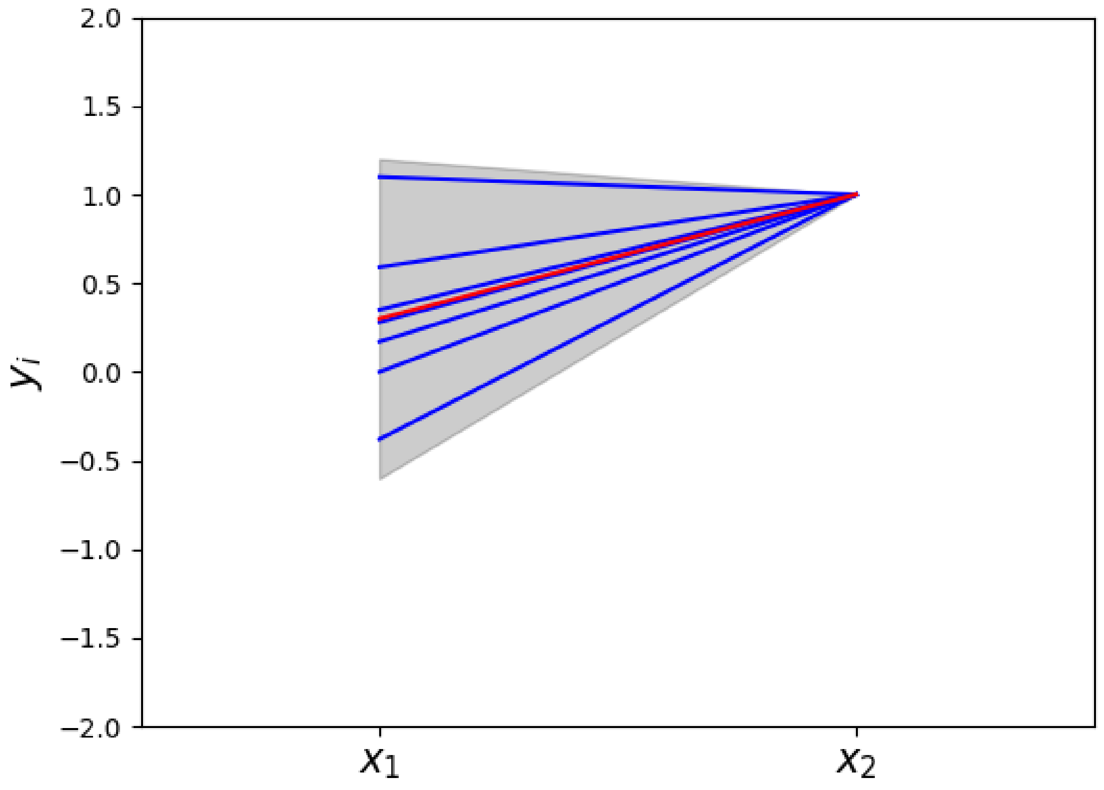
Gaussian Process Regression: conditional sampling





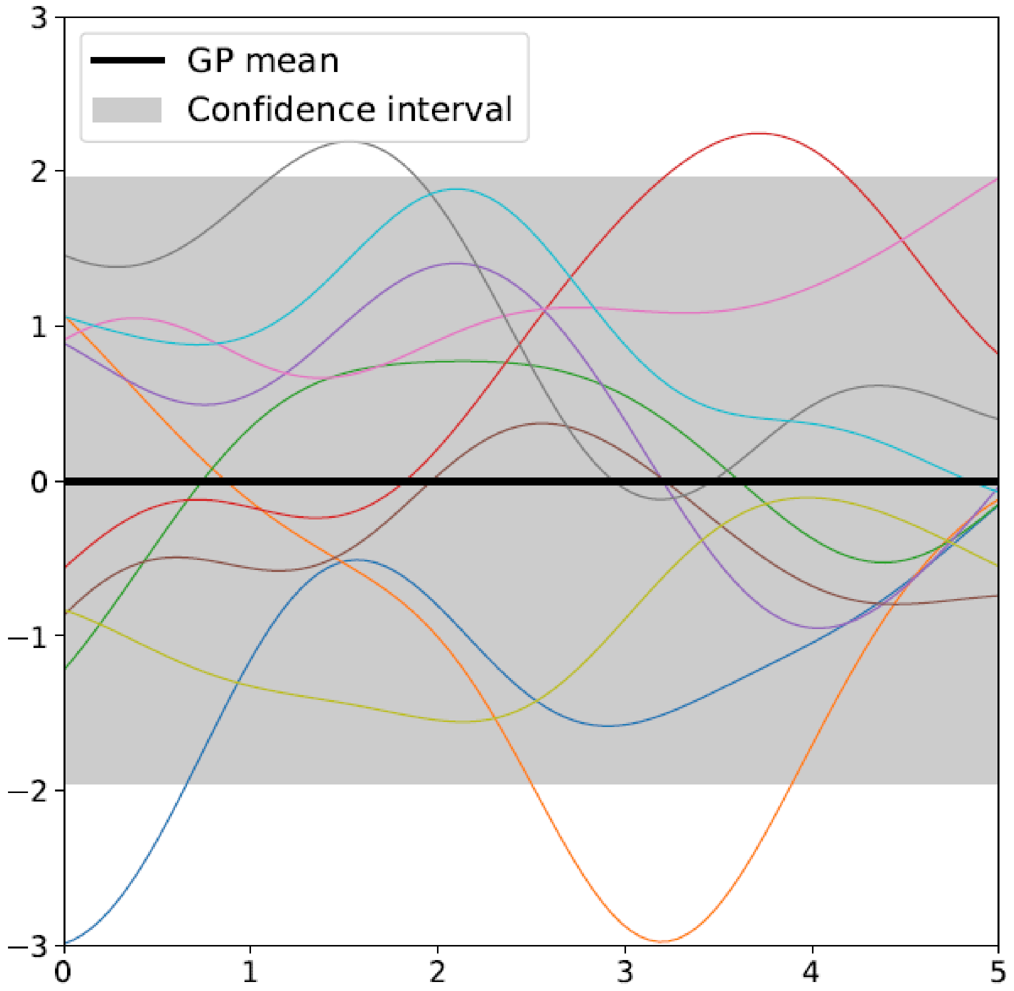
Gaussian Process Regression: conditional sampling





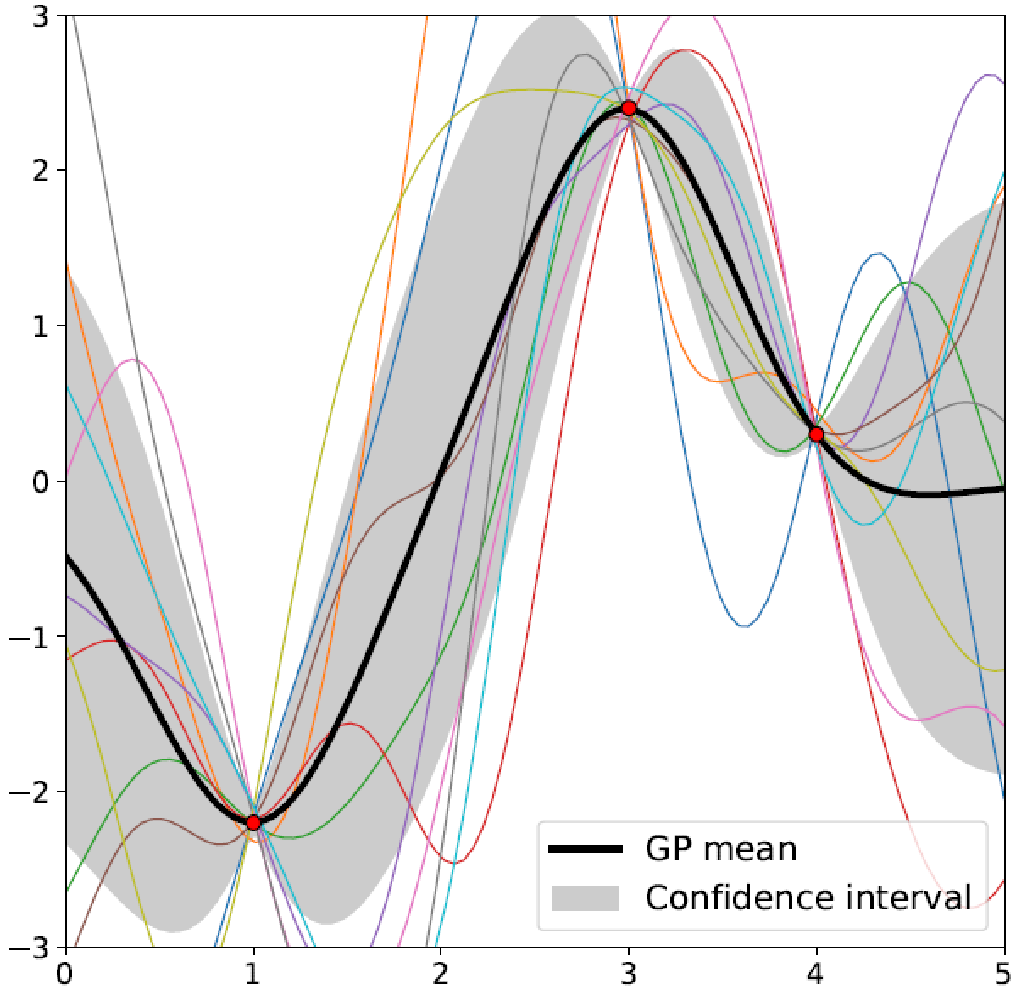
Gaussian Process Regression: Prior

Prior



Gaussian Process Regression: Posterior

Posterior

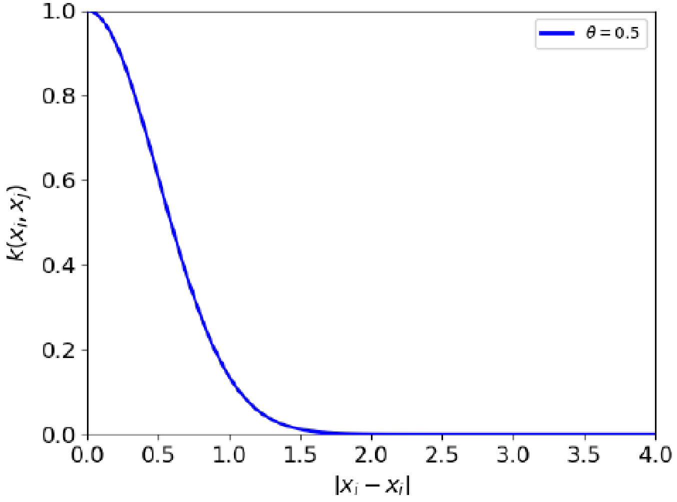


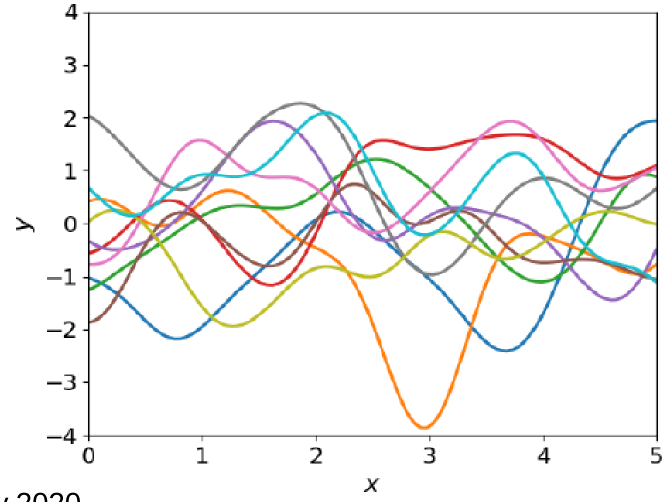
Gaussian Process Regression

- **Conditioning the Gaussian prior on the observation**
  - $\mathbb{E}(f(\boldsymbol{x}^*)|\boldsymbol{X},\boldsymbol{Y}) = k(\boldsymbol{x}^*,\boldsymbol{X})^T \mathrm{k}(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{Y}$
  - $Var(f(\boldsymbol{x}^*)|\boldsymbol{X},\boldsymbol{Y}) = k(\boldsymbol{x}^*,\boldsymbol{x}^*) - k(\boldsymbol{x}^*,\boldsymbol{X})k(\boldsymbol{X},\boldsymbol{X})^{-1}k(\boldsymbol{X},\boldsymbol{x}^*)$

- **Defining the kernel function**
  - Radial basis function $k(x_i,x_j) = \sigma_0^2 \exp\left(-\frac{(x_i-x_j)^2}{2\theta^2}\right)$
  - θ is the length scale and σ the standard deviation => Hyperparameters to optimize

$$\min_{\theta} \boldsymbol{Y}^T \mathbf{K}^{-1}\boldsymbol{Y} + \log|\boldsymbol{K}|$$



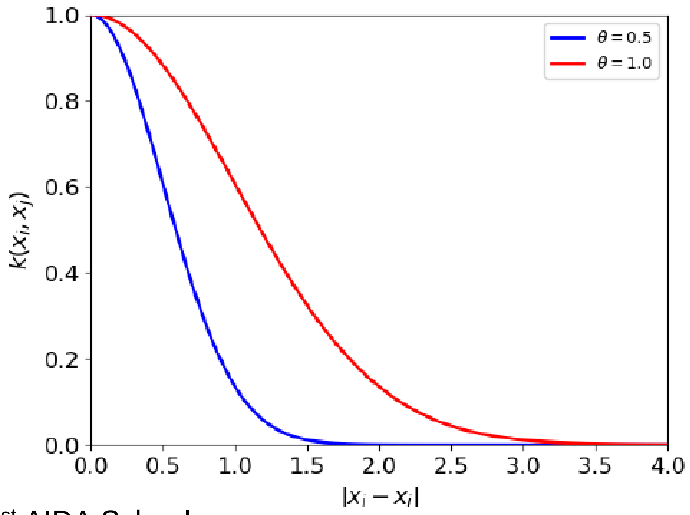


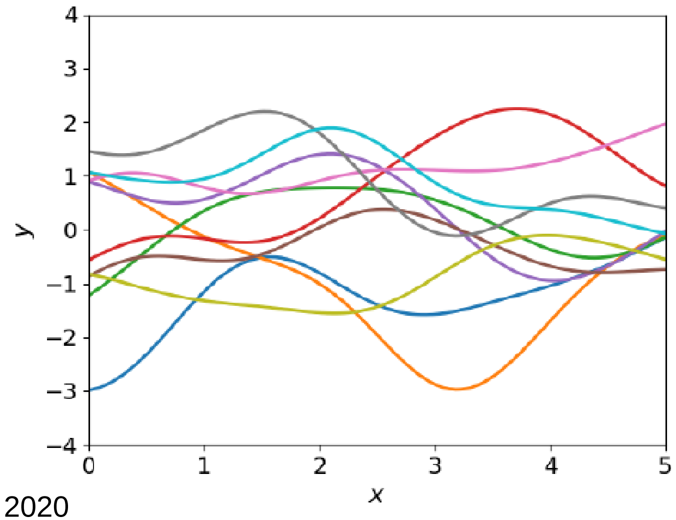
Gaussian Process Regression

- **Conditioning the Gaussian prior on the observation**
  - $\mathbb{E}(f(\boldsymbol{x}^*)|\boldsymbol{X},\boldsymbol{Y}) = k(\boldsymbol{x}^*,\boldsymbol{X})^T \mathrm{k}(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{Y}$
  - $Var(f(\boldsymbol{x}^*)|\boldsymbol{X},\boldsymbol{Y}) = k(\boldsymbol{x}^*,\boldsymbol{x}^*) - k(\boldsymbol{x}^*,\boldsymbol{X})k(\boldsymbol{X},\boldsymbol{X})^{-1}k(\boldsymbol{X},\boldsymbol{x}^*)$

- **Defining the kernel function**
  - Radial basis function $k(x_i, x_j) = \sigma_0^2 \exp\left(-\frac{(x_i - x_j)^2}{2\theta^2}\right)$
  - θ is the length scale and σ the standard deviation => Hyperparameters to optimize

$$\min_{\theta} \boldsymbol{Y}^T \mathbf{K}^{-1} \boldsymbol{Y} + \log|\boldsymbol{K}|$$



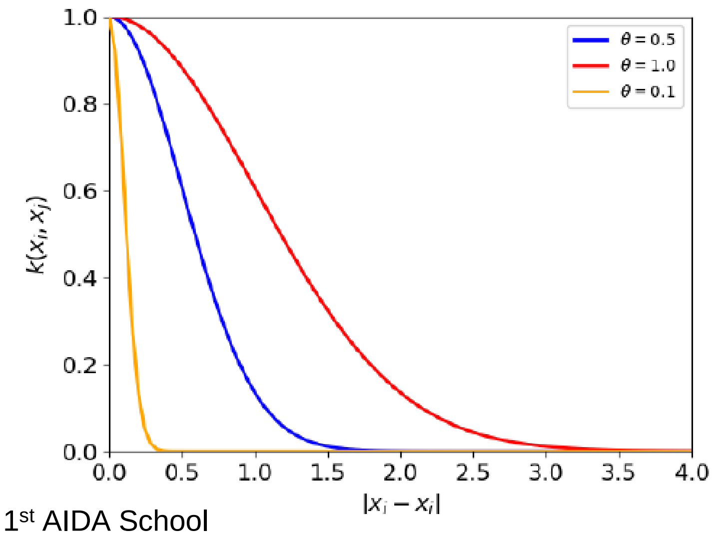


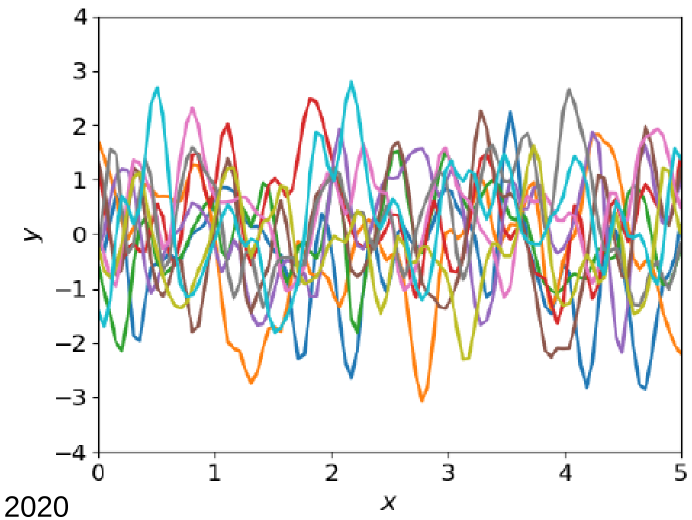
Gaussian Process Regression

- **Conditioning the Gaussian prior on the observation**
  - $\mathbb{E}(f(\boldsymbol{x}^*)|\boldsymbol{X},\boldsymbol{Y}) = k(\boldsymbol{x}^*,\boldsymbol{X})^T \mathrm{k}(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{Y}$
  - $Var(f(\boldsymbol{x}^*)|\boldsymbol{X},\boldsymbol{Y}) = k(\boldsymbol{x}^*,\boldsymbol{x}^*) - k(\boldsymbol{x}^*,\boldsymbol{X})k(\boldsymbol{X},\boldsymbol{X})^{-1}k(\boldsymbol{X},\boldsymbol{x}^*)$

- **Defining the kernel function**
  - Radial basis function $k\left(x_i, x_j\right) = \sigma_0^2 \exp\left(-\frac{\left(x_i - x_j\right)^2}{2\theta^2}\right)$
  - θ is the length scale and σ the standard deviation => Hyperparameters to optimize

$$\min_{\theta} \boldsymbol{Y}^T \mathbf{K}^{-1} \boldsymbol{Y} + \log|\boldsymbol{K}|$$





Gaussian Process Overview

- **Prior (hypothesis space)**
  - $f(\boldsymbol{x}) \sim GP\big(0, k(\boldsymbol{x}, \boldsymbol{x}')\big)$
  - $\begin{bmatrix} f(\boldsymbol{x}) \\ f(\boldsymbol{x}') \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}) & k(\boldsymbol{x}, \boldsymbol{x}') \\ k(\boldsymbol{x}', \boldsymbol{x}) & k(\boldsymbol{x}', \boldsymbol{x}') \end{bmatrix}$
  - $k(\boldsymbol{x}, \boldsymbol{x}') = \prod_i \sigma_{i,0}^2 \exp\left(-\frac{(x_i - x_i{}')^2}{2\theta^2}\right)$

Gaussian Process Overview

- **Prior (hypothesis space)**
  - $f(\boldsymbol{x}) \sim GP\left(0, k(\boldsymbol{x}, \boldsymbol{x}')\right)$
  - $\begin{bmatrix} f(\boldsymbol{x}) \\ f(\boldsymbol{x}') \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}) & k(\boldsymbol{x}, \boldsymbol{x}') \\ k(\boldsymbol{x}', \boldsymbol{x}) & k(\boldsymbol{x}', \boldsymbol{x}') \end{bmatrix}\right)$
  - $k(\boldsymbol{x}, \boldsymbol{x}') = \prod_i \sigma_{i,0}^2 \exp\left(-\frac{(x_i - x_{i'})^2}{2\theta^2}\right)$

- **Training (look at data)**
  - Training data $\{\boldsymbol{x_i}, y_i\}\ \forall i \in [1, n]$
  - $\boldsymbol{Y} \sim GP(0, \boldsymbol{K})$
  - Maximum Likelihood Estimation:
    $\min_{\theta} \boldsymbol{Y}^T \mathbf{K}^{-1} \boldsymbol{Y} + \log|\boldsymbol{K}|$

Gaussian Process Overview

- **Prior (hypothesis space)**
  - $f(\boldsymbol{x}) \sim GP\left(0, k(\boldsymbol{x}, \boldsymbol{x}')\right)$
  - $\begin{bmatrix} f(\boldsymbol{x}) \\ f(\boldsymbol{x}') \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}) & k(\boldsymbol{x}, \boldsymbol{x}') \\ k(\boldsymbol{x}', \boldsymbol{x}) & k(\boldsymbol{x}', \boldsymbol{x}') \end{bmatrix}\right.$
  - $k(\boldsymbol{x}, \boldsymbol{x}') = \prod_i \sigma_{i,0}^2 \exp\left(-\frac{(x_i - x_{i'})^2}{2\theta^2}\right)$

- **Training (look at data)**
  - Training data $\{\boldsymbol{x_i}, y_i\}\ \forall i \in [1, n]$
  - $\boldsymbol{Y} \sim GP(0, \boldsymbol{K})$
  - Maximum Likelihood Estimation:
    $\min_{\theta} \boldsymbol{Y}^T \mathbf{K}^{-1}\, \boldsymbol{Y} + \log|\boldsymbol{K}|$

- **Prediction**
  - $\begin{bmatrix} f(\boldsymbol{x}^*) \\ \boldsymbol{Y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}^*, \boldsymbol{x}^*) & k(\boldsymbol{x}^*, \boldsymbol{X}) \\ k(\boldsymbol{X}, \boldsymbol{x}^*) & \boldsymbol{K} \end{bmatrix}\right.$

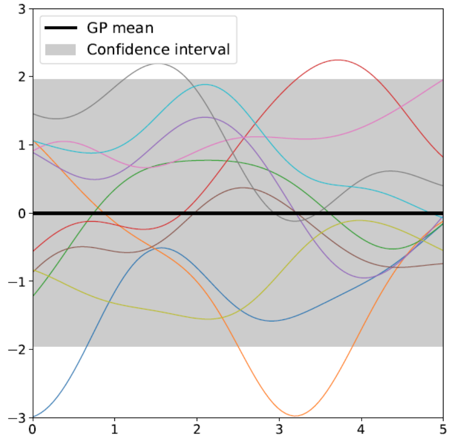
Gaussian Process Overview

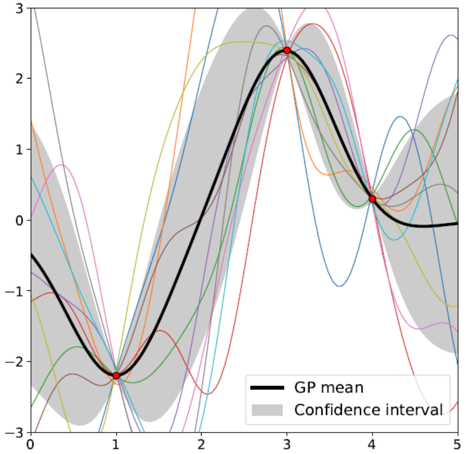
- **Prior (hypothesis space)**
  - $f(\boldsymbol{x}) \sim GP\big(0, k(\boldsymbol{x}, \boldsymbol{x}')\big)$
  - $\begin{bmatrix} f(\boldsymbol{x}) \\ f(\boldsymbol{x}') \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}) & k(\boldsymbol{x}, \boldsymbol{x}') \\ k(\boldsymbol{x}', \boldsymbol{x}) & k(\boldsymbol{x}', \boldsymbol{x}') \end{bmatrix}\right]$
  - $k(\boldsymbol{x}, \boldsymbol{x}') = \prod_i \sigma_{i,0}^2 \exp\left(-\frac{(x_i - x_{i'})^2}{2\theta^2}\right)$

- **Training (look at data)**
  - Training data $\{\boldsymbol{x_i}, y_i\}\ \forall i \in [1, n]$
  - $\boldsymbol{Y} \sim GP(0, \boldsymbol{K})$
  - Maximum Likelihood Estimation:
    $\min_\theta \boldsymbol{Y}^T \mathbf{K}^{-1} \boldsymbol{Y} + \log|\boldsymbol{K}|$

- **Prediction**
  - $\begin{bmatrix} f(\boldsymbol{x}^*) \\ \boldsymbol{Y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}^*, \boldsymbol{x}^*) & k(\boldsymbol{x}^*, \boldsymbol{X}) \\ k(\boldsymbol{X}, \boldsymbol{x}^*) & \boldsymbol{K} \end{bmatrix}\right]$

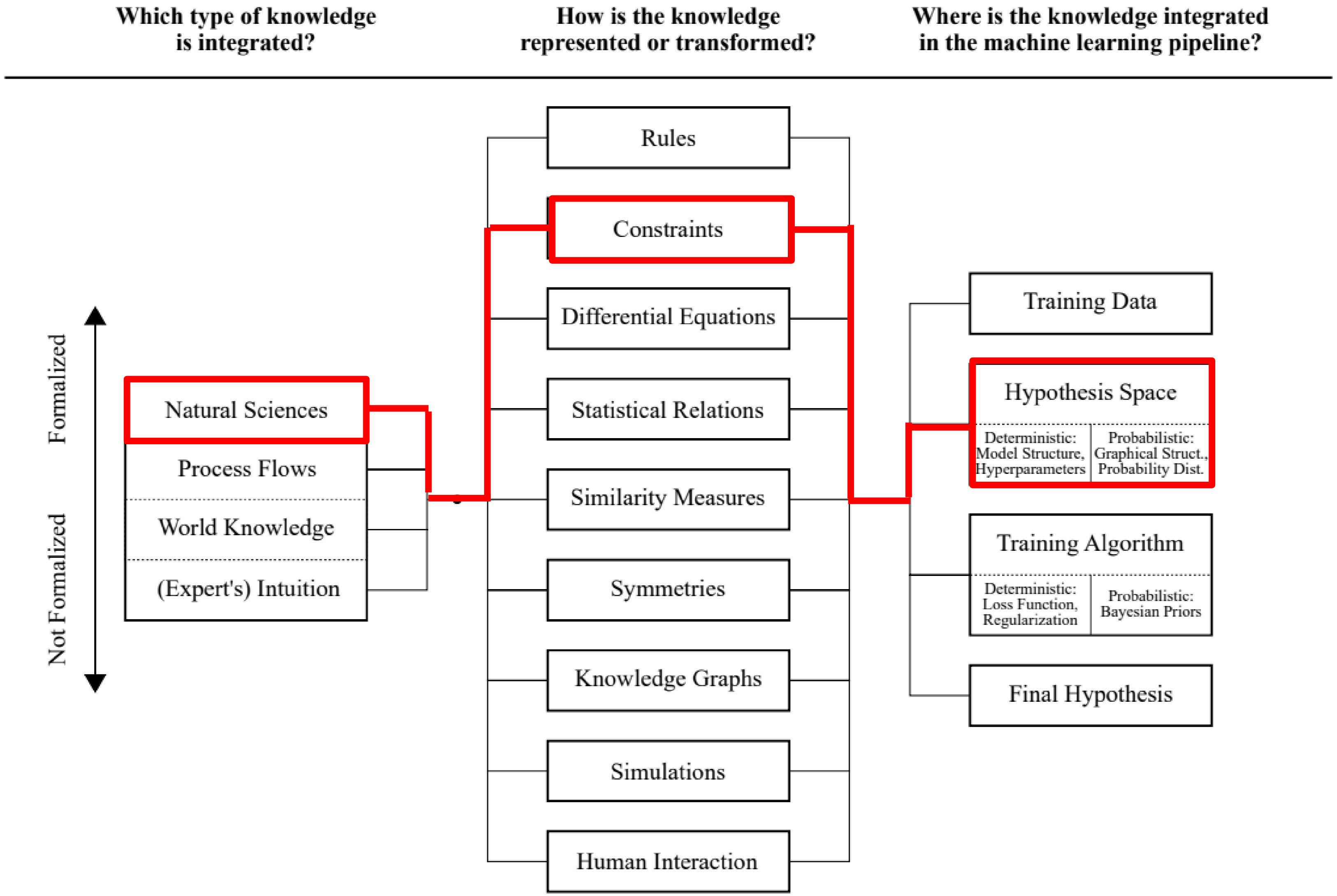
- **Posterior**
  - $f(\boldsymbol{x}^*)|\boldsymbol{X}, \boldsymbol{Y} = \mathcal{N}(\mu(\boldsymbol{x}^*), Var(\boldsymbol{x}^*))$
  - $\mu(\boldsymbol{x}^*) = k(\boldsymbol{x}^*, \boldsymbol{X})^T \mathbf{K}^{-1} \boldsymbol{Y}$
  - $Var(\boldsymbol{x}^*) = k(\boldsymbol{x}^*, \boldsymbol{x}^*) - k(\boldsymbol{x}^*, \boldsymbol{X}) \boldsymbol{K}^{-1} k(\boldsymbol{X}, \boldsymbol{x}^*)$





Physics-informed ML

Limiting the model space



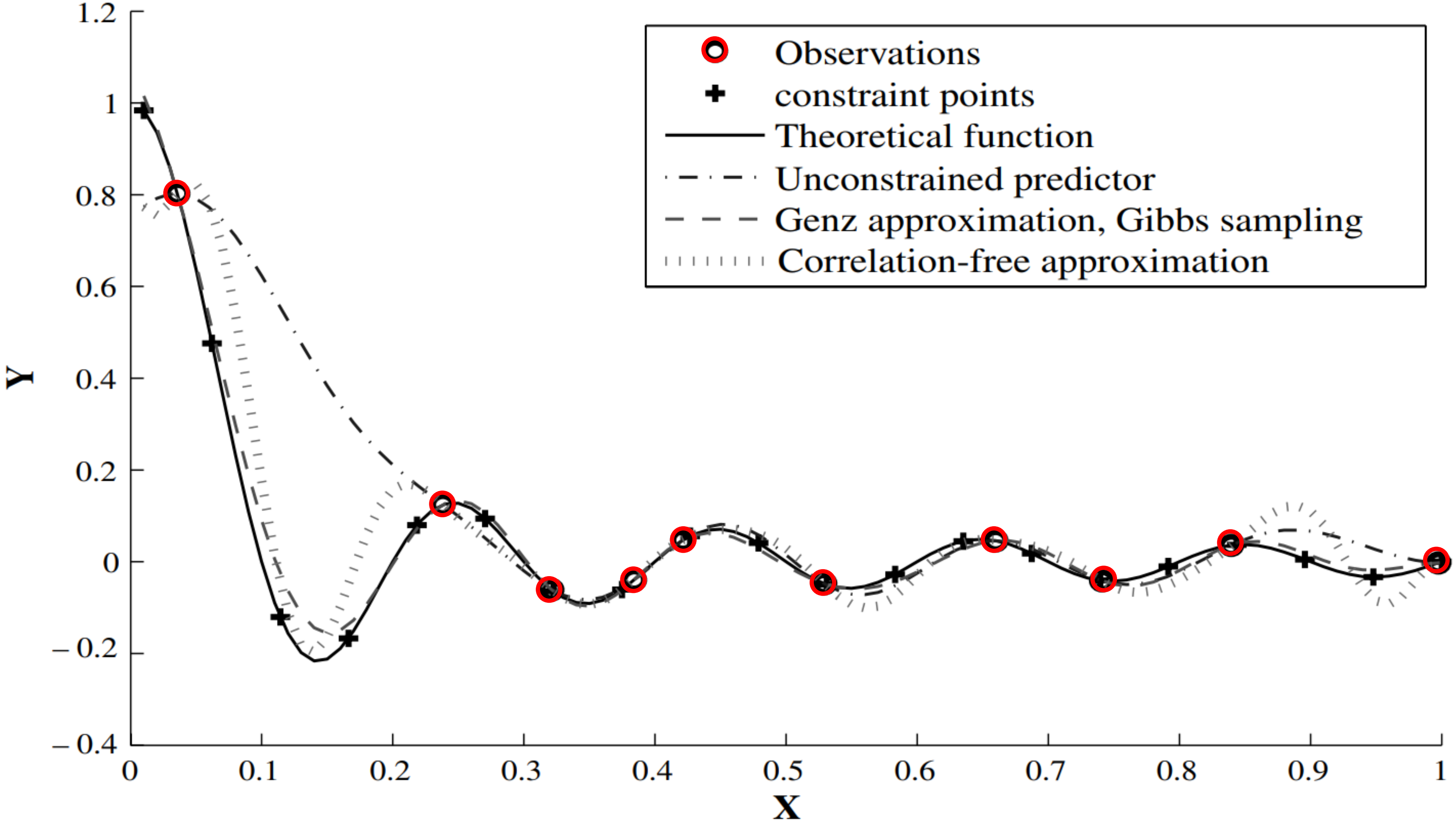
von Rued et al.

Gaussian Process Regression with inequality constraints

- **Physical quantities can show properties know in advance**
  - Bound constraints: density, threshold, etc
  - Derivative constraints: monotonicity

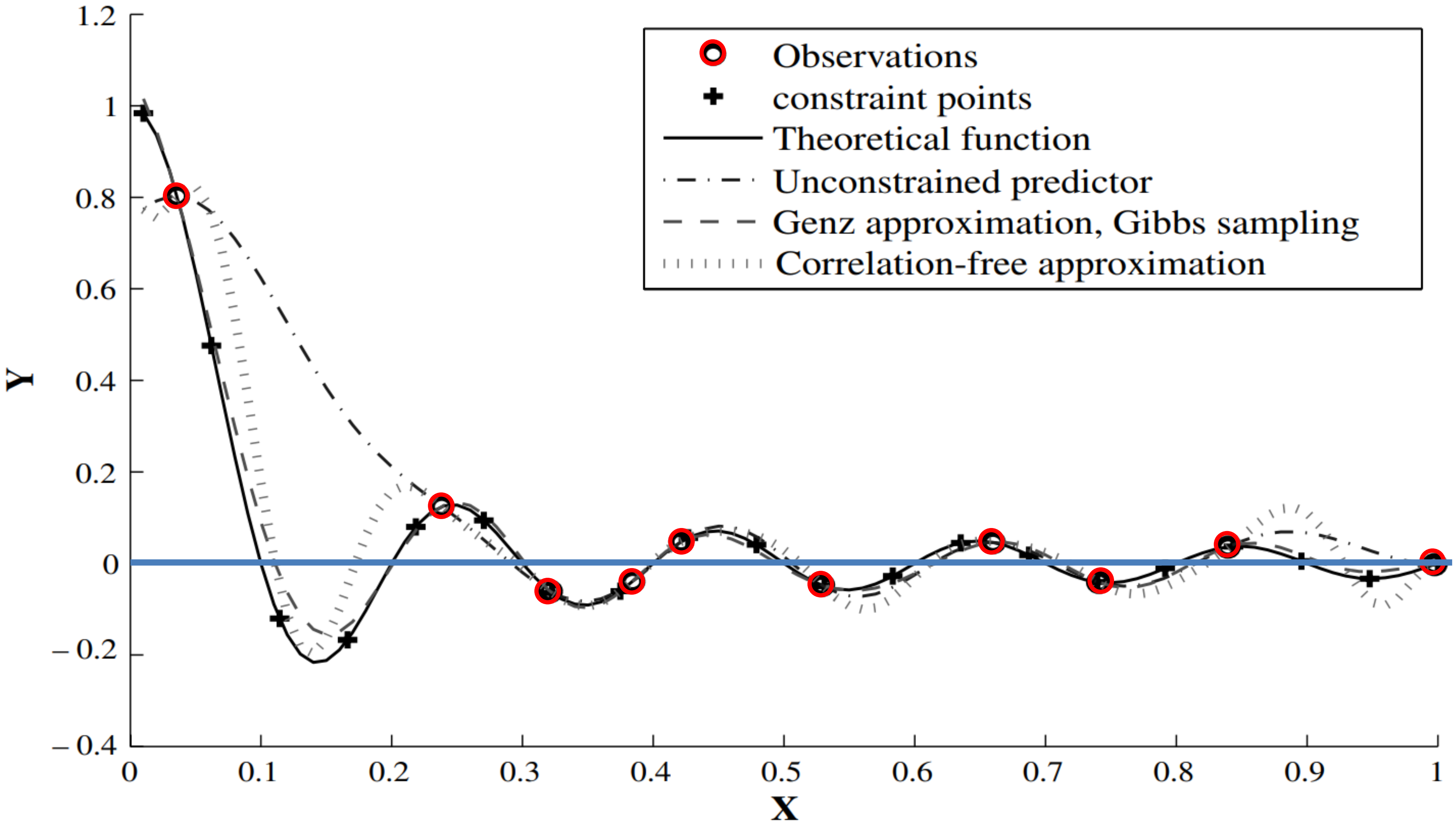
- **Constraints imposed in the mathematical formulation (Da Veiga et al.)**
  - Bounds: $\mathbb{E}(f(x^*)|\forall i = i, \ldots, N;\ a_i \leq f(x_i) \leq b_i)$
  - Derivative: $\mathbb{E}(f(x^*)|\forall i = i, \ldots, N;\ \frac{\partial f}{\partial x_j}(x_i) \geq 0)$
  - Truncated multinormal distribution

Gaussian Process Regression with inequality constraints



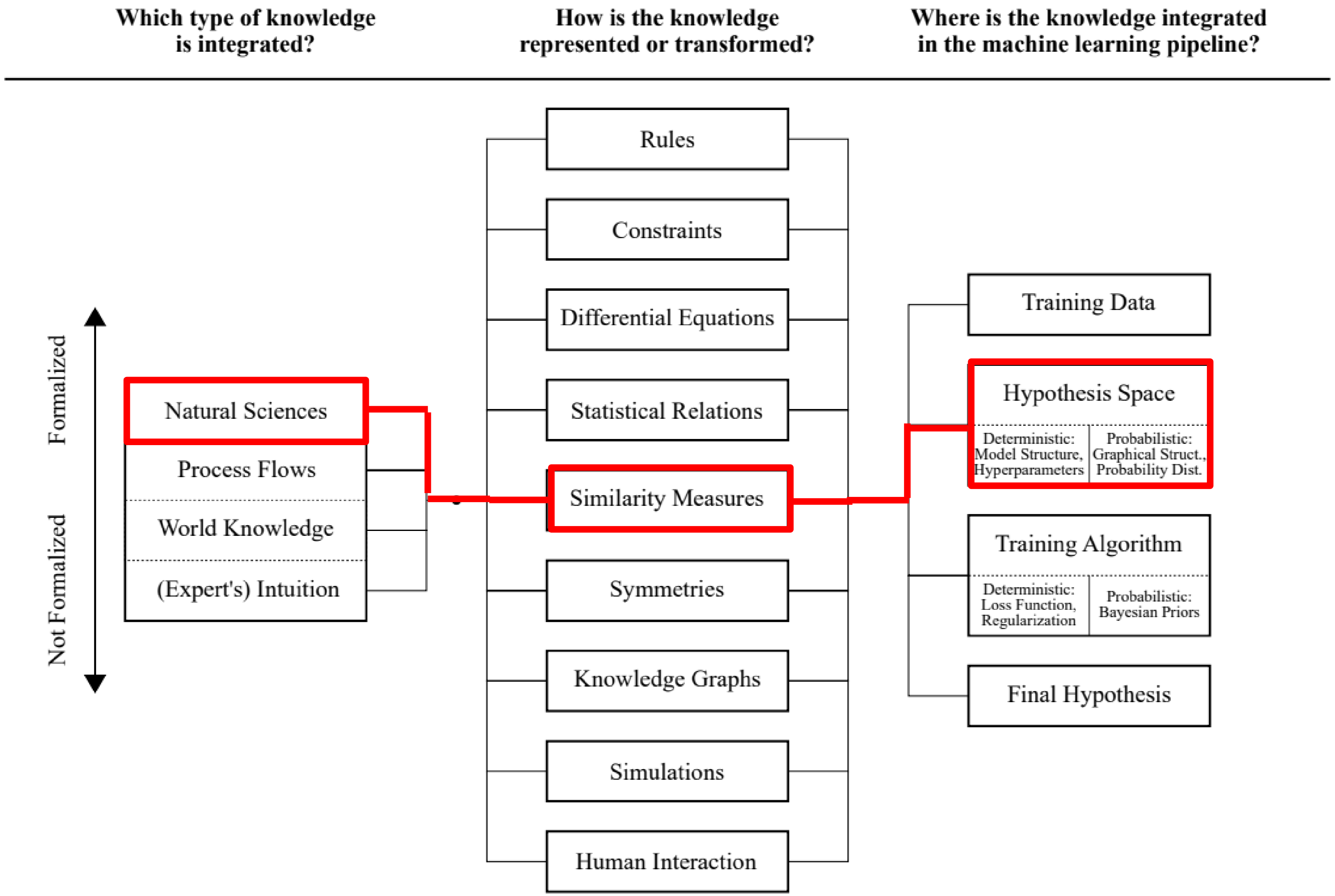
Da Veiga et al.

Gaussian Process Regression with inequality constraints



Da Veiga
et al.

Multi fidelity GP regression



von Rued et al.

Multi fidelity GP regression

- **Various sources of data can be used**
  - Low fidelity: partially converged simulation, coarser mesh, governing equation of lower fidelity, etc.
  - High fidelity: intensive simulations, measurements, etc.

Multi fidelity GP regression

- **Various sources of data can be used**
  - Low fidelity: partially converged simulation, coarser mesh, governing equation of lower fidelity, etc.
  - High fidelity: intensive simulations, measurements, etc.

- **Main assumptions**
  - Correction can model the differences between cheap and expensive functions
  - Cheap points $\boldsymbol{X_C}, y_c$ and expensive points $\boldsymbol{X_e}, y_e$
  - $Cov\{f_e(\boldsymbol{x^i}), f_c(\boldsymbol{x}) | f_c(\boldsymbol{x^i})\} = 0 \; \forall \boldsymbol{x} \neq \boldsymbol{x^i}$
  - Prior assumption: $Z_e(\boldsymbol{x}) = \rho Z_c(\boldsymbol{x}) + Z_d(\boldsymbol{x})$

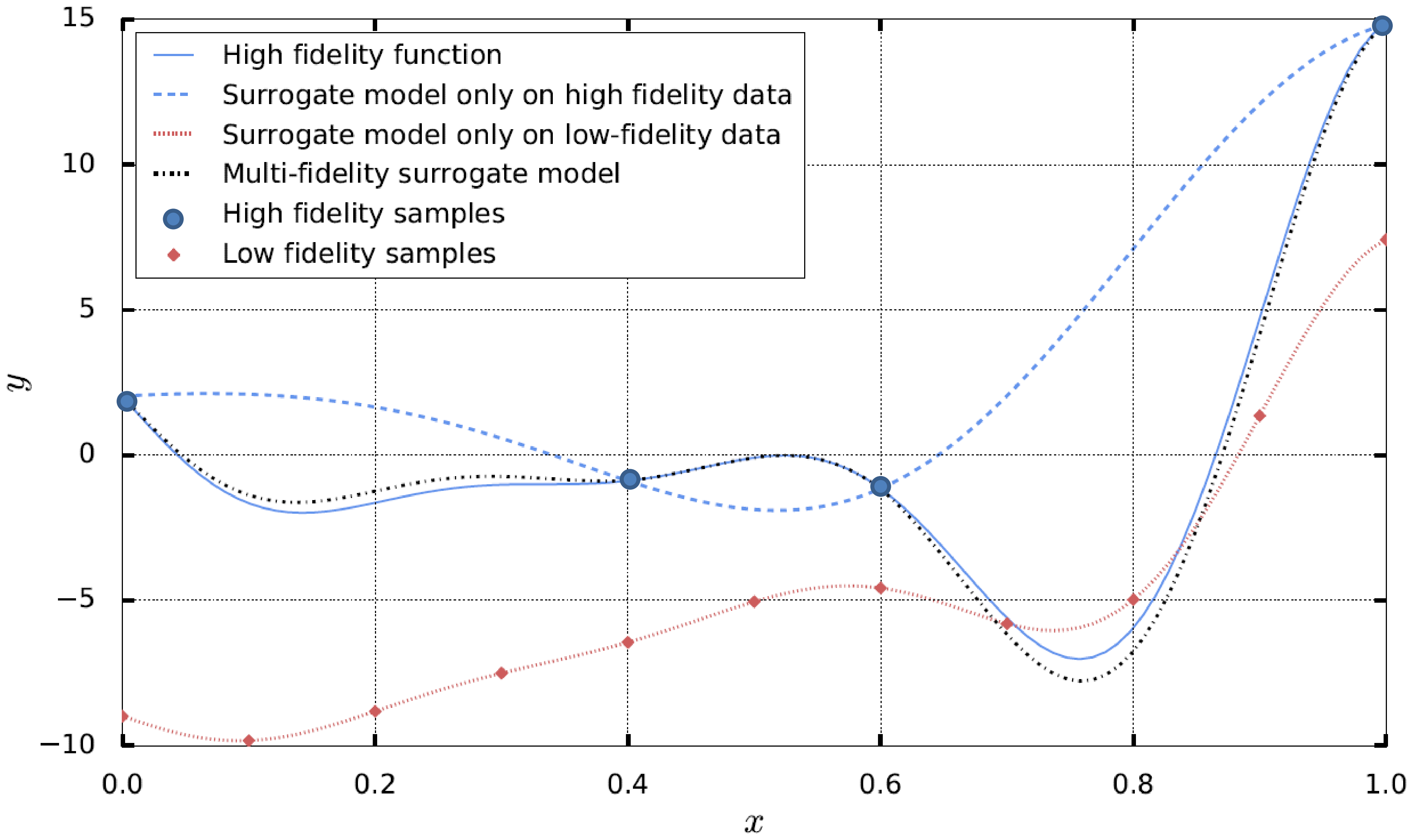
Multi fidelity GP regression

- **Various sources of data can be used**
  - Low fidelity: partially converged simulation, coarser mesh, governing equation of lower fidelity, etc.
  - High fidelity: intensive simulations, measurements, etc.

- **Main assumptions**
  - Correction can model the differences between cheap and expensive functions
  - Cheap points $\boldsymbol{X_c}, y_c$ and expensive points $\boldsymbol{X_e}, y_e$
  - $Cov\{f_e(\boldsymbol{x^i}), f_c(\boldsymbol{x}) | f_c(\boldsymbol{x^i})\} = 0 \ \forall \boldsymbol{x} \neq \boldsymbol{x^i}$
  - Prior assumption: $Z_e(\boldsymbol{x}) = \rho Z_c(\boldsymbol{x}) + Z_d(\boldsymbol{x})$

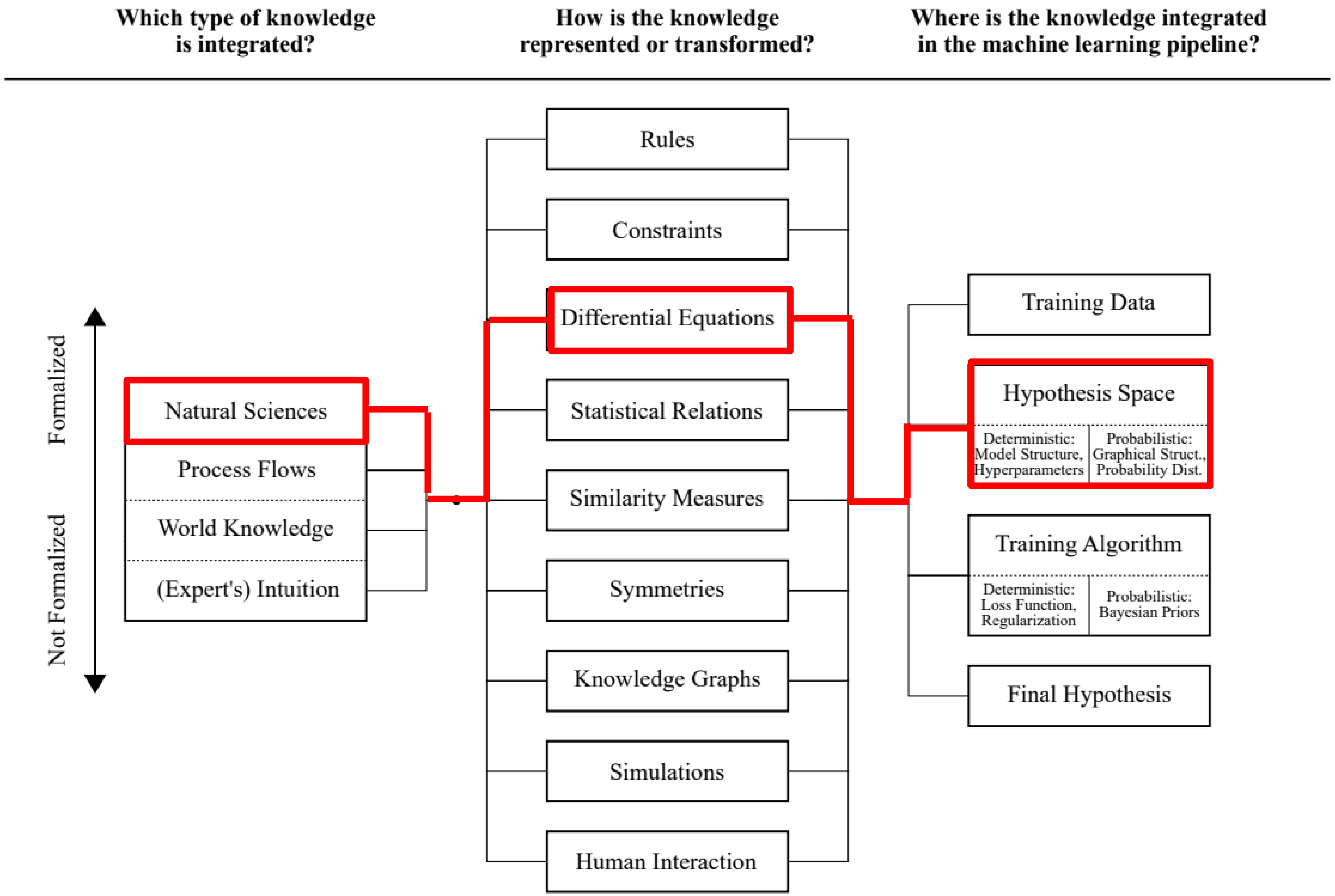
- **Mathematical formula**
  - Correlation: $C = \begin{bmatrix} k_c(\boldsymbol{X_c}, \boldsymbol{X_c}) & \rho k_c(\boldsymbol{X_c}, \boldsymbol{X_e}) \\ \rho k_c(\boldsymbol{X_e}, \boldsymbol{X_c}) & \rho^2 k_c(\boldsymbol{X_e}, \boldsymbol{X_e}) + k_d(\boldsymbol{X_e}, \boldsymbol{X_e}) \end{bmatrix}$

Multi fidelity



Inspired by Forrester et al.

PDE within GPs



von Rued et al.

Incorporating physics within GPs (Raissi et al.)

- **Burgers' equation**
  - $u_t + uu_x - \left(\frac{0.01}{\pi}\right) u_{xx} = 0$
  - Boundary conditions $u(t,-1) = u(t,1) = 0$
  - Data: Noisy measurements of initial solutions $(u(0,x) = -\sin(\pi x))$

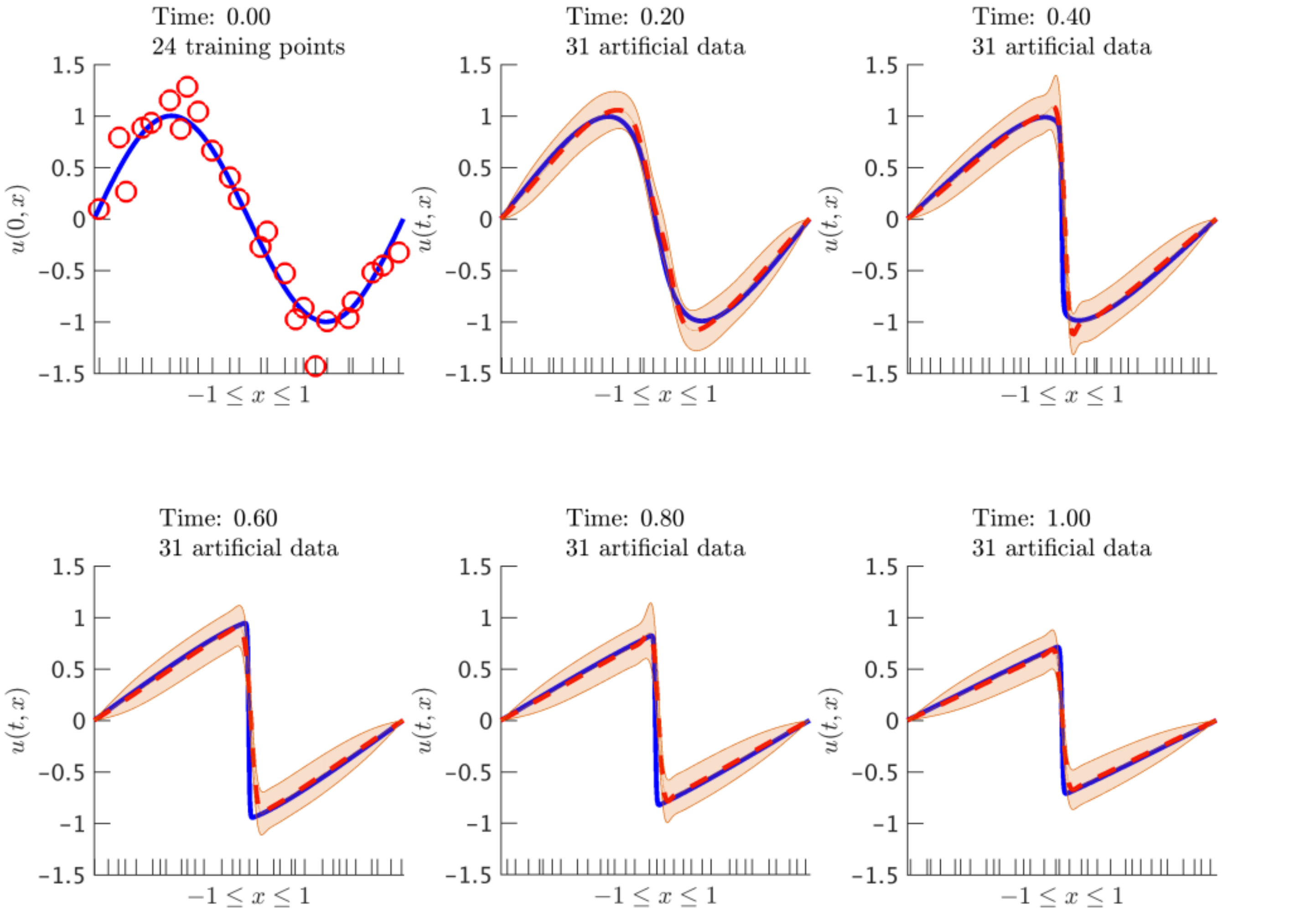
Incorporating physics within GPs (Raissi et al.)

- **Burgers' equation**
  - $u_t + uu_x - \left(\frac{0.01}{\pi}\right)u_{xx} = 0$
  - Boundary conditions $u(t,-1) = u(t,1) = 0$
  - Data: Noisy measurements of initial solutions $(u(0,x) = -\sin(\pi x))$

- **Incorporating physics inside GPs**
  - Discretization: $u^n + \Delta\mathrm{t}\left(\mathrm{u}^{\mathrm{n}-1}\mathrm{u}_{\mathrm{x}}^{\mathrm{n}} - \left(\frac{0.01}{\pi}\right)\mathrm{u}_{\mathrm{xx}}^{\mathrm{n}}\right) = \mathrm{u}^{\mathrm{n}-1}$
  - Prior: $u^n \sim GP\big(0, k(x,x')\big)$
  - Kernel $k^{n-1,n} = k^{n,n} + \Delta t\left(u^{n-1}\,k_x - \left(\frac{0.01}{\pi}\right)k_{xx}\right)$

$$\begin{bmatrix} u^n \\ u^{n-1} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k^{n,n} & k^{n-1,n} \\ k^{n,n-1} & k^{n-1,n-1} \end{bmatrix}\right)$$

Incorporating physics within GPs (Raissi et al.)



Hidden Physics Models (Raissi et al., 2018b)

- **Linear partial differential equation with unknown parameters $\boldsymbol{\lambda}$**
  - $\mathcal{L}_x^{\lambda} h^n = h^{n-1}$

- **Same process to incorporate physics**
  - $h^n \sim GP\left(0, k(x, x')\right)$
  - $k^{n,n} = k$
  - $k^{n-1,n} = \mathcal{L}_x^{\lambda} k$
  - $k^{n-1,n-1} = \mathcal{L}_x^{\lambda} \mathcal{L}_x^{\lambda} k$
  - λ is now a hyperparameter

Hidden Physics Models (Raissi et al., 2018b)

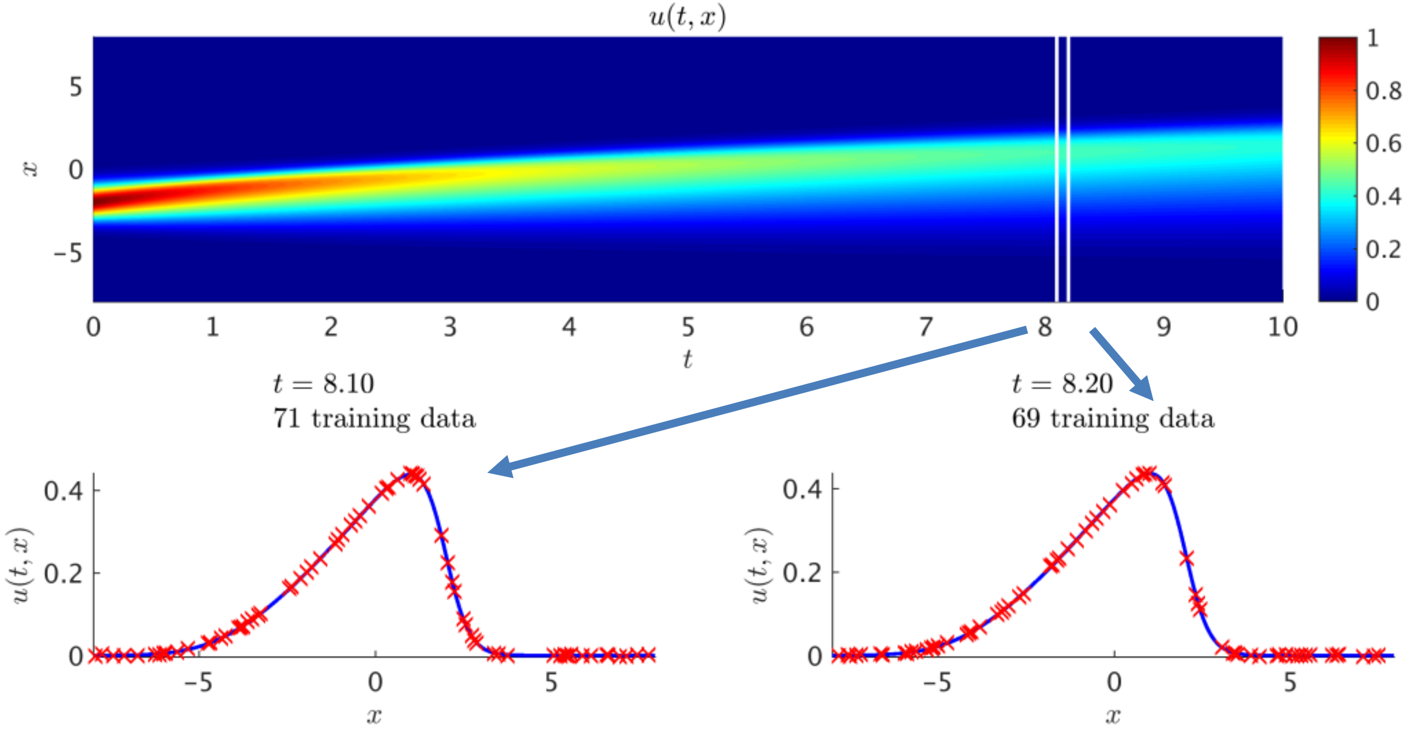
- **Linear partial differential equation with unknown parameters λ**
  - $\mathcal{L}_x^\lambda h^n = h^{n-1}$

- **Same process to incorporate physics**
  - $h^n \sim GP\left(0, k(x, x')\right)$
  - $k^{n,n} = k$
  - $k^{n-1,n} = \mathcal{L}_x^\lambda k$
  - $k^{n-1,n-1} = \mathcal{L}_x^\lambda \mathcal{L}_x^\lambda k$
  - λ is now a hyperparameter

- **Burgers**
  - $u_t + \lambda_1 u u_x - \lambda_2 u_{xx}$
  - Training with only 2 snapshots

Hidden Physics Models (Raissi et al., 2018b)

- **Linear partial differential equation with unknown parameters λ**
  - $\mathcal{L}_x^{\lambda} h^n = h^{n-1}$
- **Same process to incorporate physics**
  - $h^n \sim GP\big(0, k(x, x')\big)$
  - $k^{n,n} = k$
  - $k^{n-1,n} = \mathcal{L}_x^{\lambda} k$
  - $k^{n-1,n-1} = \mathcal{L}_x^{\lambda} \mathcal{L}_x^{\lambda} k$
  - λ is now a hyperparameter
- **Burgers**
  - $u_t + \lambda_1 u u_x - \lambda_2 u_{xx}$
  - Training with only 2 snapshots



Correct PDE	$u_t + uu_x - 0.1u_{xx} = 0$
Identified PDE (clean data)	$u_t + 1.028uu_x - 0.101u_{xx} = 0$
Identified PDE (1% noise)	$u_t + 1.017uu_x - 0.094u_{xx} = 0$

Physics informed GP and NN

- **Solving PDE, propagating uncertainties, and inverse problem**
  - Based on symbolic differentiation
  - Limited by GP capabilities
  - Can be cost efficient compared to pure physics-modeling

Physics informed GP and NN

- **Solving PDE, propagating uncertainties, and inverse problem**
  - Based on symbolic differentiation
  - Limited by GP capabilities
  - Can be cost efficient compared to pure physics-modeling

- **What about Physics informed Neural Network? (Raissi et al.)**
  - It also works
  - Adapted to large data and automatic differentiation

Physics informed GP and NN

- **Solving PDE, propagating uncertainties, and inverse problem**
  - Based on symbolic differentiation
  - Limited by GP capabilities
  - Can be cost efficient compared to pure physics-modeling

- **What about Physics informed Neural Network? (Raissi et al.)**
  - It also works
  - Adapted to large data and automatic differentiation

$$f := u_t + u u_x - (0.01/\pi) u_{xx}$$

$$MSE = MSE_u + MSE_f$$

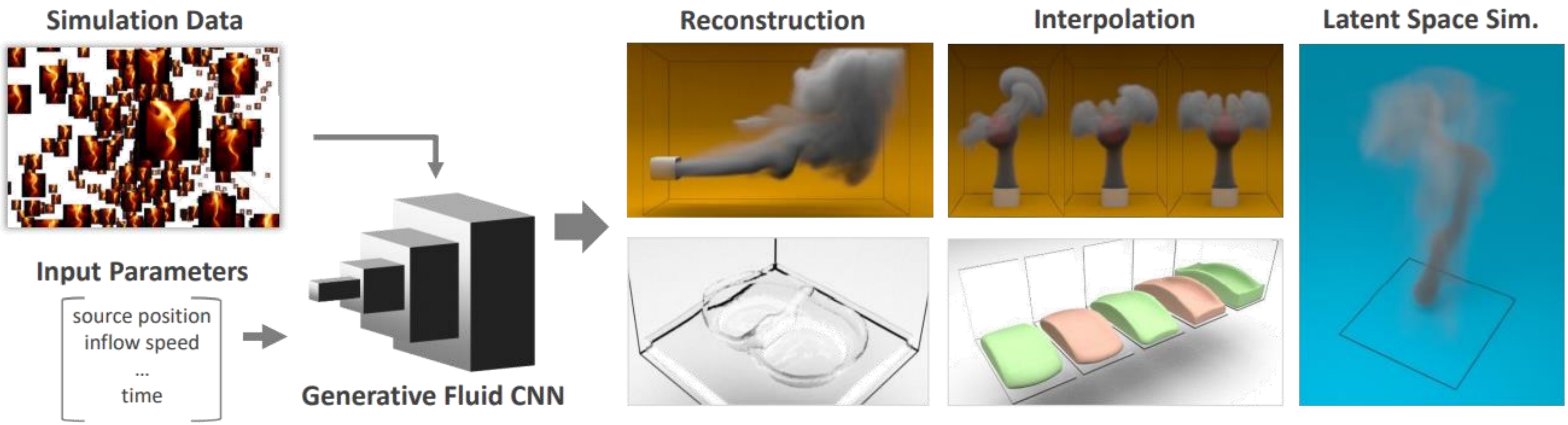
$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2,$$

Raissi et al. provide github with Tensorflow code

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2.$$

And more!

- **DeepFluid: Generative Network for Fluid Simulation (Kim et al., 2019)**



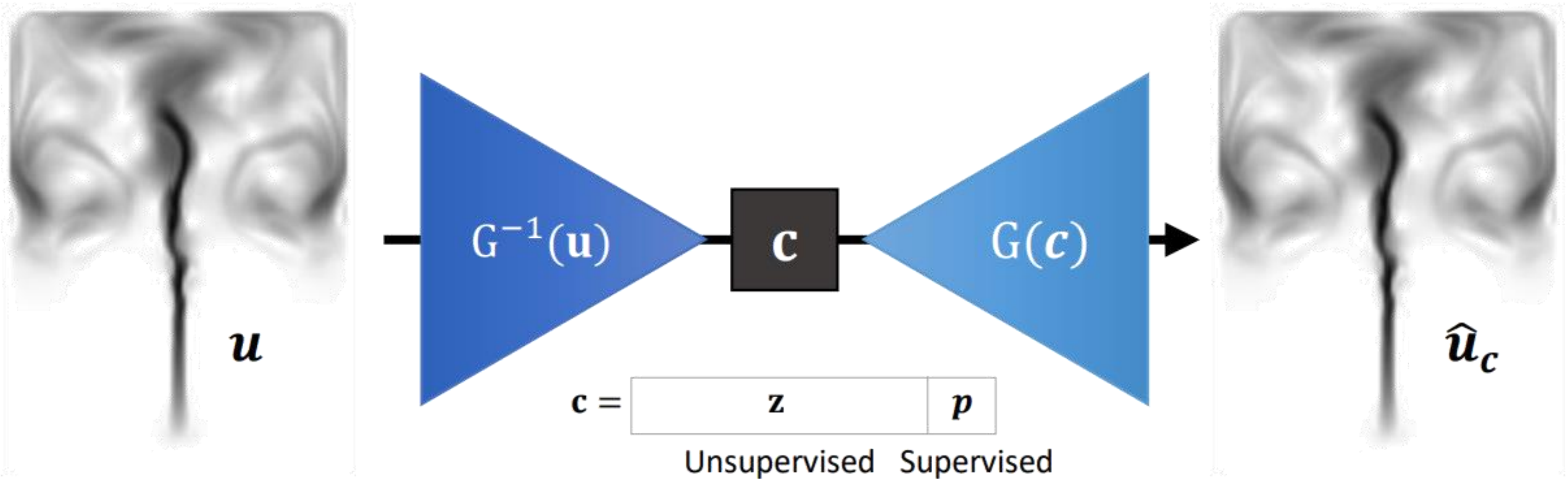
And more!

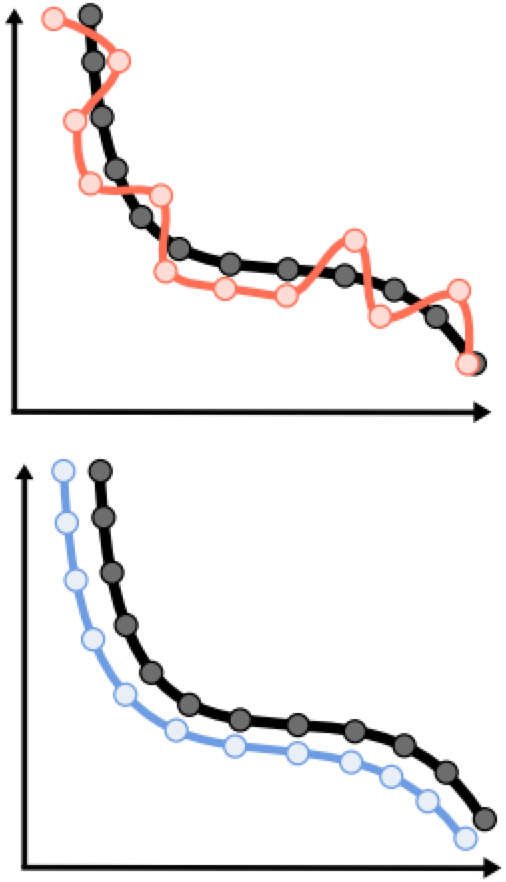
- **DeepFluid: Generative Network for Fluid Simulation (Kim et al., 2019)**
  - Stream loss function for incompressible fluid

$$L_G(\mathbf{c}) = ||\mathbf{u_c} - \nabla \times G(\mathbf{c})||_1$$

  - Learning also the gradient:

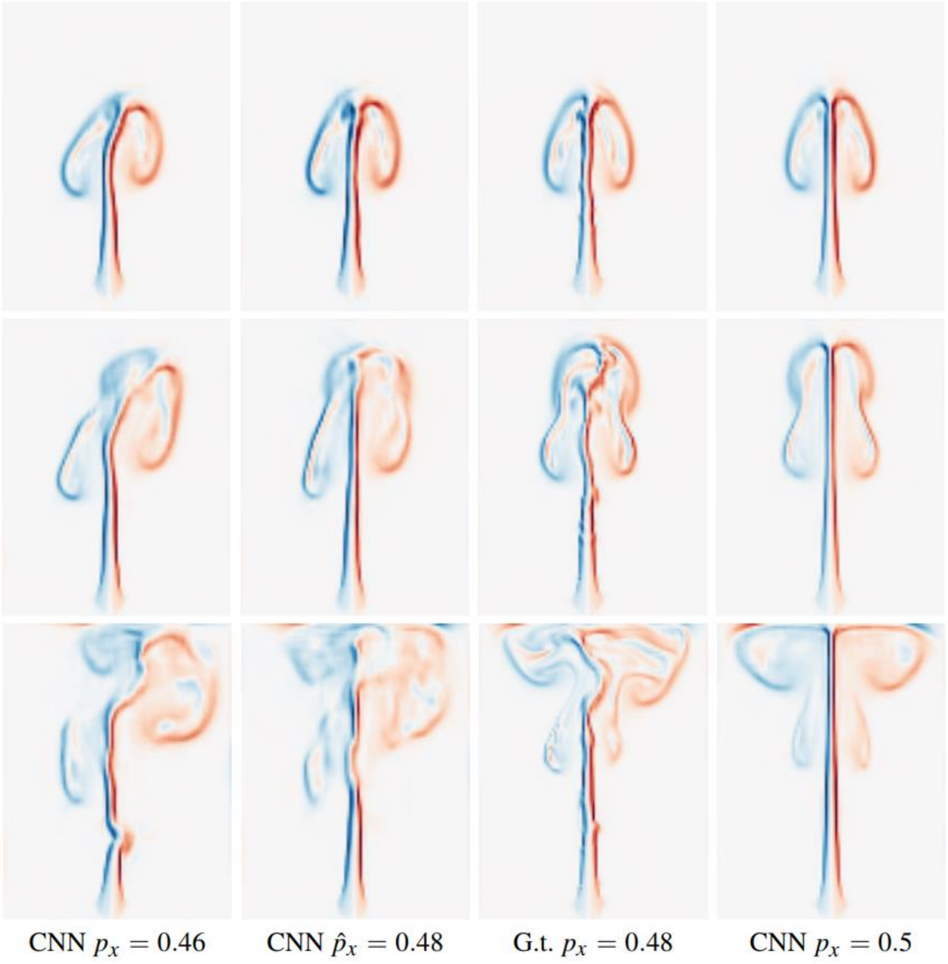
$$L_G(\mathbf{c}) = \lambda_{\mathbf{u}}||\mathbf{u_c} - \hat{\mathbf{u}}_\mathbf{c}||_1 + \lambda_{\nabla\mathbf{u}}||\nabla\mathbf{u_c} - \nabla\hat{\mathbf{u}}_\mathbf{c}||_1$$

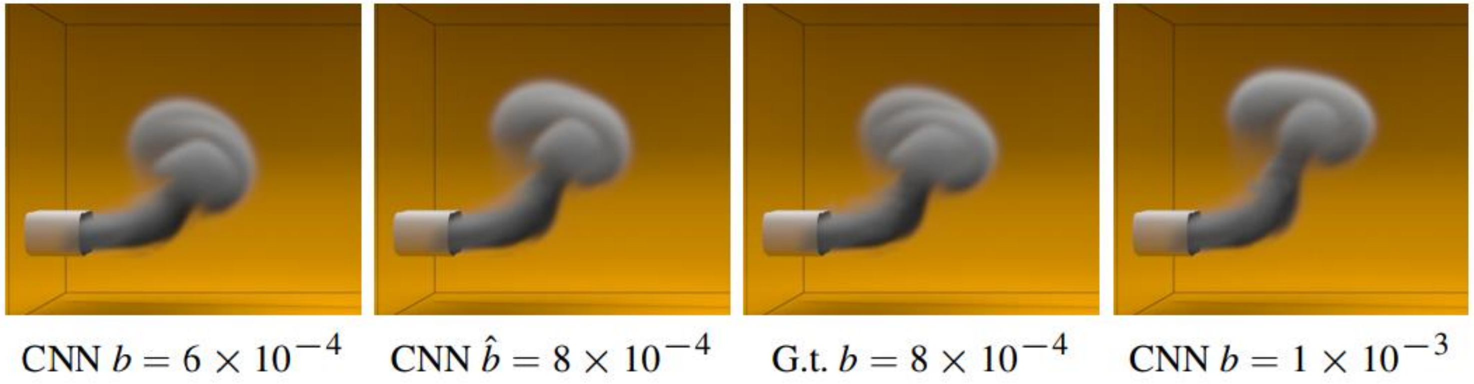




And more!

- **DeepFluid: Generative Network for Fluid Simulation (Kim et al., 2019)**





Conclusion

- **Coupling physics and data**
  - Gain explainabilitiy
  - Physical consistency

- **Need of closer collaboration with domain experts, data scientists, and computer scientists**
  - Extracting knowledge, insights and discovery from data
  - Support of simulations and experiments
  - One of the AIDA's aims

- **Now physics is injected into Machine Learning, we can explain models from AI**
  - Next talk from Jorge about explainable AI

References

Laura von Rued et al., “Informed Machine Learning – Toward a Taxonomy of Explicit Integration of Knowledge into Machine Learning”, https://arxiv.org/abs/1903.12394

Forrester, S. András, and A. J. Keane. “Engineering Design via Surrogate Modelling: a practical guide”, Wiley, 2008.

Sebastien Da Veiga et al., “Gaussian process modeling with inequality constraints”, In Annales de la Faculté des sciences de Toulouse: Mathématiques (Vol. 21, No. 3, pp. 529-555).

Taira et al., “Modal Analysis of Fluid Flows: An Overview”, Aiaa Journal, 4013-4041.

Rasmussen et al., “Gaussian processes for machine learning”, Vol 1, Cambridge MIT press, 2006

Anuj Karpatne et al., “Theory-guided Data Science: A New Paradigm for Scientific Discovery from Data”, IEEE Transactions on Knowledge and Data Engineering, 29(10), 2318-2331

Anuj Karpatne et al., “Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling”, https://arxiv.org/abs/1710.11431

Mazia Raissi et al., “Numerical Gaussian Processes for time-dependent and non-linear partial differential equations”, SIAM J Sci. Comput., Vol 40, No1, pp A172-A198, 2018

Raissi, M., & Karniadakis, G. E., “Hidden physics models: Machine learning of nonlinear partial differential equations”, *Journal of Computational Physics*, *357*, 125-141., 2018b

Physics Informed Deep Learning (Raissi et al.): https://maziarraissi.github.io/PINNs/

Kim, B et al., “Deep fluids: A generative network for parameterized fluid simulations”, In Computer Graphics Forum (Vol. 38, No. 2, pp. 59-70), 2019