Regression methods

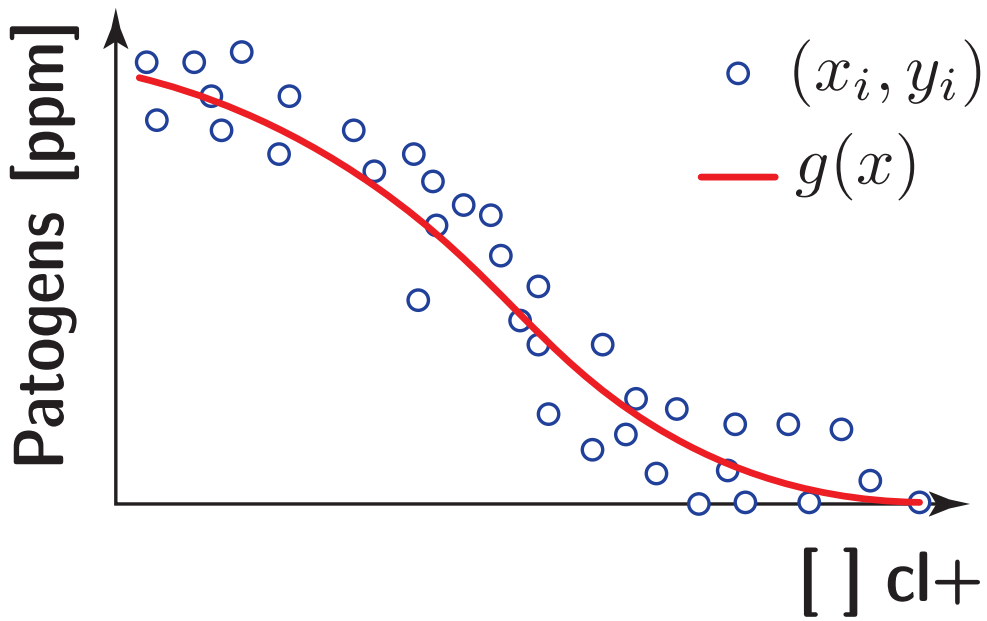
(CIV6540 - Probabilistic Machine Learning for Civil Engineers)

Professor: James-A. Goulet

Département des génies civil, géologique et des mines
Polytechnique Montréal

Chapter 8 – Goulet (2020)
Probabilistic Machine Learning for Civil Engineers
MIT Press

What is regression?



Data

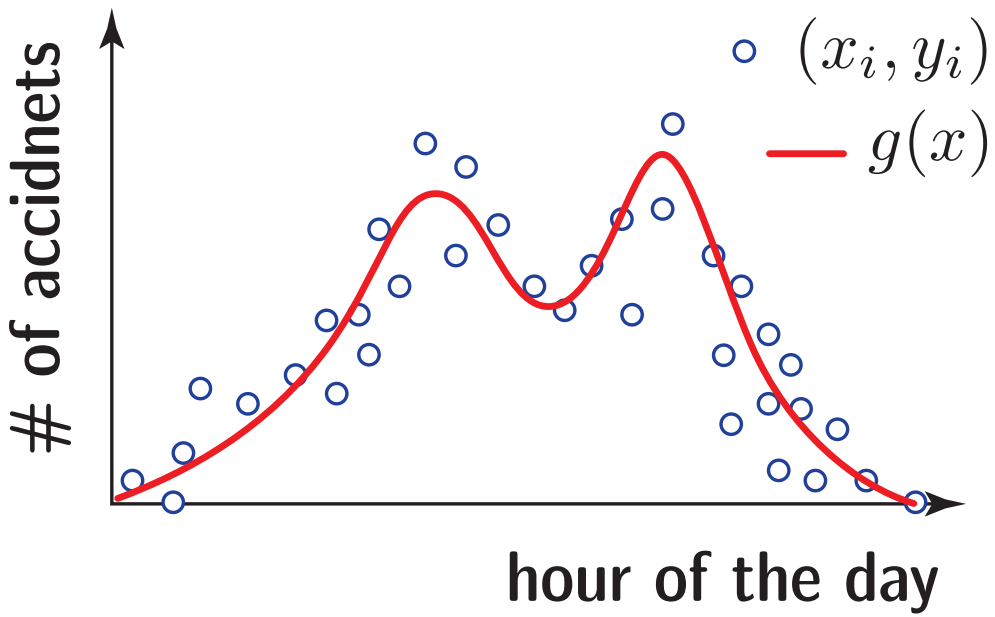
$\mathcal{D} = \{(x_i, y_i), \forall i = 1 : \mathtt{D}\}$

$x_i \in \mathbb{R}$: { Covariate, attribute, regressor }

$y_i \in \mathbb{R}$: Observation

Regression methods: mathematical models for $g(x)$

What is regression?



Data

$\mathcal{D} = \{(x_i, y_i), \forall i = 1 : \mathtt{D}\}$

$x_i \in \mathbb{R}$: { Covariate, attribute, regressor }

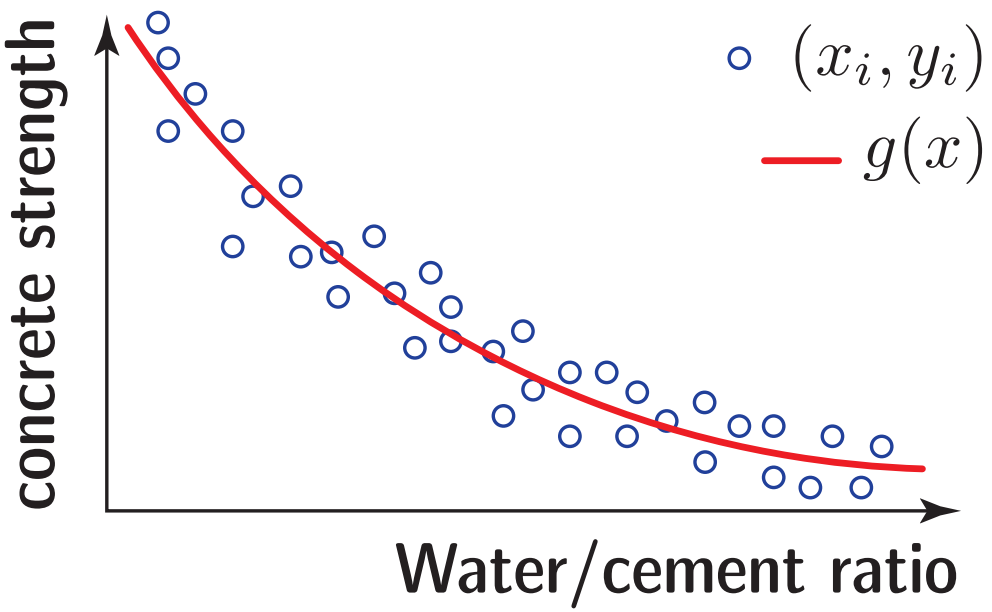
$y_i \in \mathbb{R}$: Observation

Model

$g(x) \equiv \text{fct}(x)$

Regression methods: mathematical models for $g(x)$

What is regression?



Data

$\mathcal{D} = \{(x_i, y_i), \forall i = 1 : \mathtt{D}\}$

$x_i \in \mathbb{R}$: { Covariate / attribute / regressor

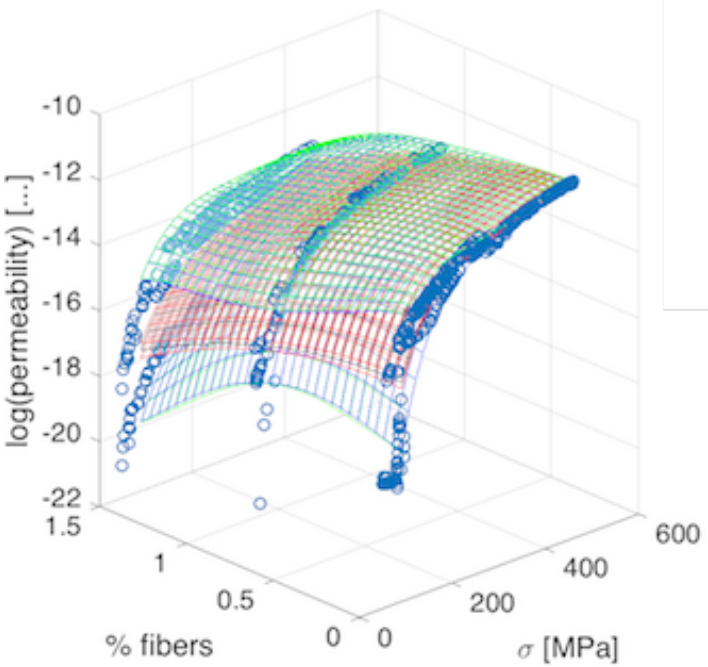
$y_i \in \mathbb{R}$: Observation

Model

$g(x) \equiv \text{fct}(x)$

Regression methods: mathematical models for $g(x)$

Regression in $> 1D$



Data

$$\mathcal{D} = \{(\mathbf{x}_i, y_i), \forall i = 1 : \mathtt{D}\}$$

$$\mathbf{x}_i = [x_1, x_2]_i^\intercal \in \mathbb{R}^2 : \begin{cases} \text{Covariates} \\ \text{attributes} \\ \text{regressors} \end{cases}$$

$$y_i \in \mathbb{R}^1 : \text{Observation}$$

Model

$$g(\mathbf{x}) \equiv \text{fct}(\mathbf{x}) \in \mathbb{R}^1$$

General case: $\mathbf{x}_i = [x_1, x_2, \cdots, x_{\mathtt{X}}]_i^\intercal \in \mathbb{R}^{\mathtt{X}},\ y_i \in \mathbb{R}^1,\ g(\mathbf{x}_i) \in \mathbb{R}^1$

Module #5 Outline

Topics organization

Background
- 1 Revision probability & linear algebra
- 2 Probability distributions

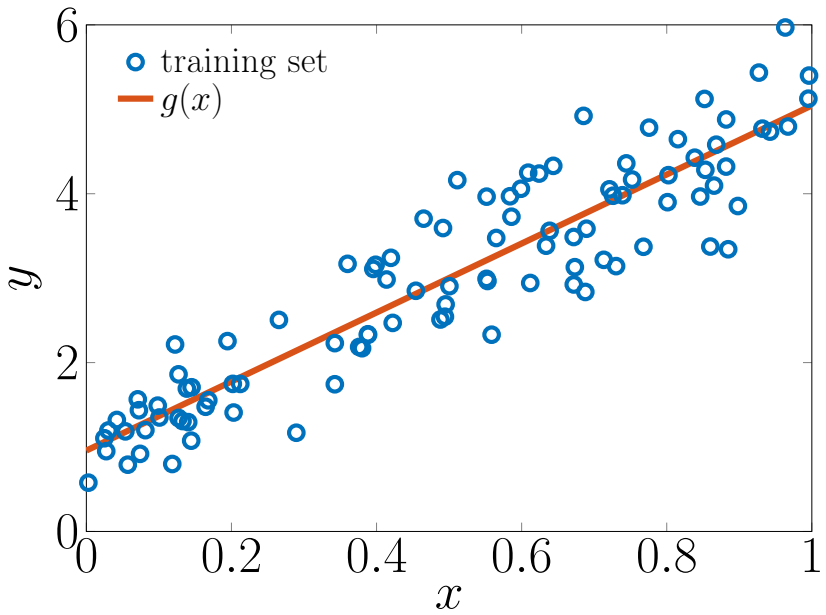
Machine Learning Basics
- 0 Introduction
- 3 Bayesian Estimation $p(A|B) = \frac{p(B|A)p(A)}{p(B)}$
- 4 MCMC sampling & Newton

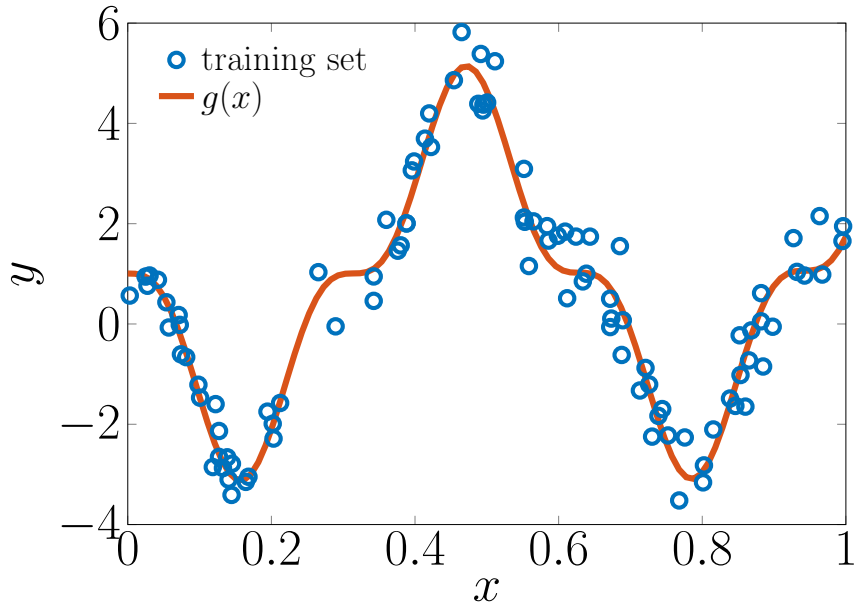
Supervised learning
- 5 Regression
- 6 Classification
- 7 LSTM networks for time series

Unsupervised learning
- 7 State-space model for time-series

Decision Making & RL
- 8 Decision Theory
- 9 AI & Sequential decision problems

Which one is an example of linear regression?



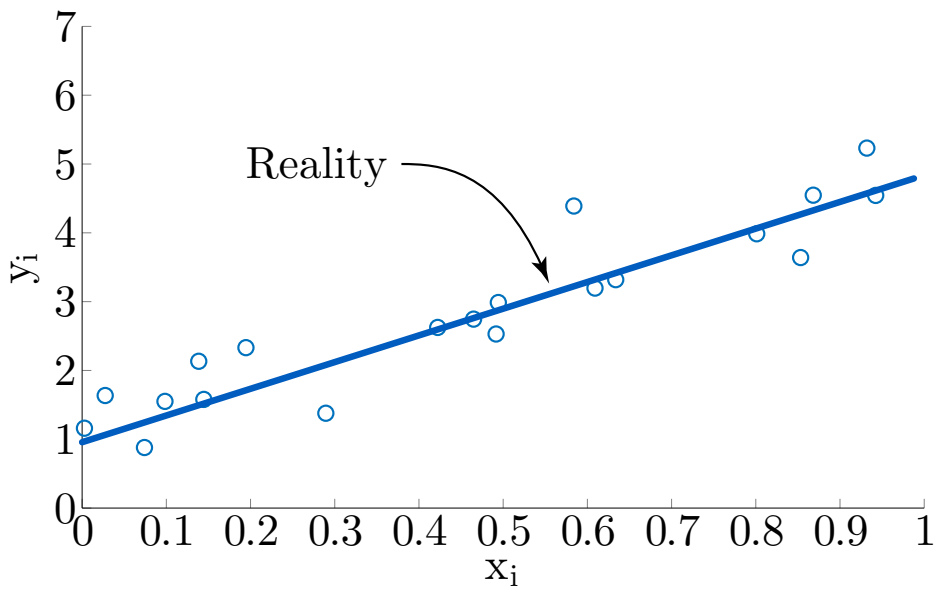


Both.

Section Outline

Linear Regression

Nomenclature & hypotheses



Data

$$\mathcal{D} = \{(x_i, y_i), \forall i = 1 : \mathtt{D}\}$$

$$x_i \in \mathbb{R} : \begin{cases} \text{Covariate} \\ \text{attribute} \\ \text{regressor} \end{cases}$$

$$y_i \in \mathbb{R} : \text{Observation}$$

Model

$$g(x) \equiv \text{fct}(x)$$

Hypothesis: $g(x) \equiv$ reality, i.e. no variability

$$y = g(x) + v, \;\; v : V \sim \mathcal{N}(v; 0, \sigma_V^2), \;\; \overbrace{V_i \perp\!\!\!\perp V_j, \;\forall i \neq j}^{\text{independent observation errors}}$$

Mathematical formulation – special case

Let us assume that our model takes the form

$$g(x) = b_0 + b_1 x = [b_0 \ b_1] \begin{bmatrix} 1 \\ x \end{bmatrix}$$

$$y = g(x) + v, \ \ v : V \sim \mathcal{N}(v; 0, \sigma_V^2)$$

For the entire dataset $\mathcal{D} = \{(x_i, y_i), \forall i = 1 : \mathtt{D}\} = \{\mathcal{D}_y, \mathcal{D}_x\}$

$$\underbrace{\mathbf{b}}_{\boldsymbol{\theta}} = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}, \ \underbrace{\mathbf{y}}_{\mathcal{D}_y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{\mathtt{D}} \end{bmatrix}, \ \underbrace{\mathbf{X}}_{\mathcal{D}_x} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_{\mathtt{D}} \end{bmatrix}, \ \boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{\mathtt{D}} \end{bmatrix}$$

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{v}, \ \ \mathbf{v} : \mathbf{V} \sim \mathcal{N}(\mathbf{v}; \mathbf{0}, \sigma_V^2 \cdot \mathbf{I})$$

Joint likelihood of $\mathcal{D} = \{(x_i, y_i), \forall i = 1 : \mathtt{D}\} = \{\mathcal{D}_y, \mathcal{D}_x\}$

$$\underbrace{\mathbf{y}}_{\mathcal{D}_y} = \underbrace{\mathbf{X}}_{\mathcal{D}_x}\underbrace{\mathbf{b}}_{\boldsymbol{\theta}} + \mathbf{v}, \ \mathbf{v} : \mathbf{V} \sim \mathcal{N}(\mathbf{v}; \mathbf{0}, \sigma_V^2 \cdot \mathbf{I})$$

$$\begin{aligned}
f(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta}) &= f(\mathcal{D}_y = \mathbf{y}|\mathcal{D}_x = \mathbf{x}, \boldsymbol{\theta} = \mathbf{b}) \\
&= \prod_{i=1}^{\mathtt{D}} \mathcal{N}(y_i; g(x_i), \sigma_V^2) \\
&\propto \prod_{i=1}^{\mathtt{D}} \exp\left(-\frac{1}{2}\frac{(y_i - g(x_i))^2}{\sigma_V^2}\right) \\
&= \exp\left(-\frac{1}{2}\sum_{i=1}^{\mathtt{D}}\frac{(y_i - g(x_i))^2}{\sigma_V^2}\right) \\
&= \exp\left(-\frac{1}{2\sigma_V^2}(\mathbf{y} - \mathbf{X}\mathbf{b})^\intercal(\mathbf{y} - \mathbf{X}\mathbf{b})\right)
\end{aligned}$$

Estimating optimal parameters $\boldsymbol{\theta}^* = \mathbf{b}^*$

The goal is to estimate optimal parameters $\boldsymbol{\theta}^* = \mathbf{b}^*$ which maximize the likelihood $f(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta})$, or equivalently which **maximize** the log-likelihood

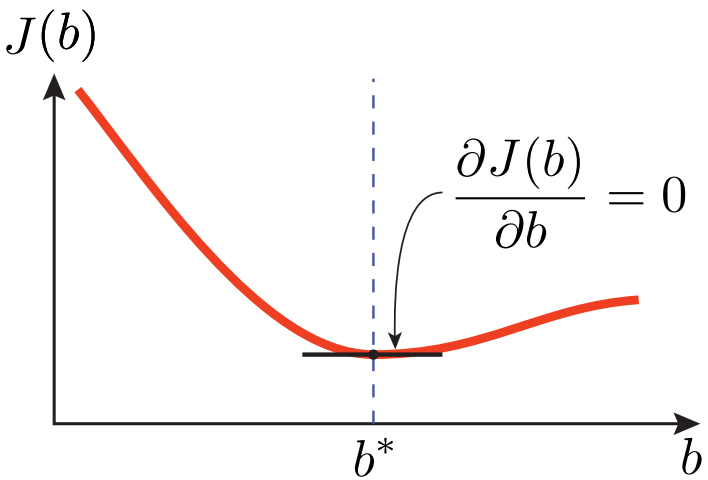
$$\begin{aligned} \ln f(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta}) &= -\frac{1}{2\sigma_V^2}(\mathbf{y} - \mathbf{X}\mathbf{b})^\intercal(\mathbf{y} - \mathbf{X}\mathbf{b}) \\ &\propto -\frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{b})^\intercal(\mathbf{y} - \mathbf{X}\mathbf{b}) \\ &= -J(\mathbf{b}) \quad \text{(Loss function)} \end{aligned}$$

In the context of linear regression, we **minimize** $J(\mathbf{b})$, which is **equivalent to maximizing** $f(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta})$

Estimating optimal parameters $\boldsymbol{\theta}^* = \mathbf{b}^*$ (cont.)

Loss function: measures the model quality

$$J(\mathbf{b}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{b})^\intercal(\mathbf{y} - \mathbf{X}\mathbf{b}) = \frac{1}{2}\sum_{i=1}^{D}(y_i - g(x_i))^2$$



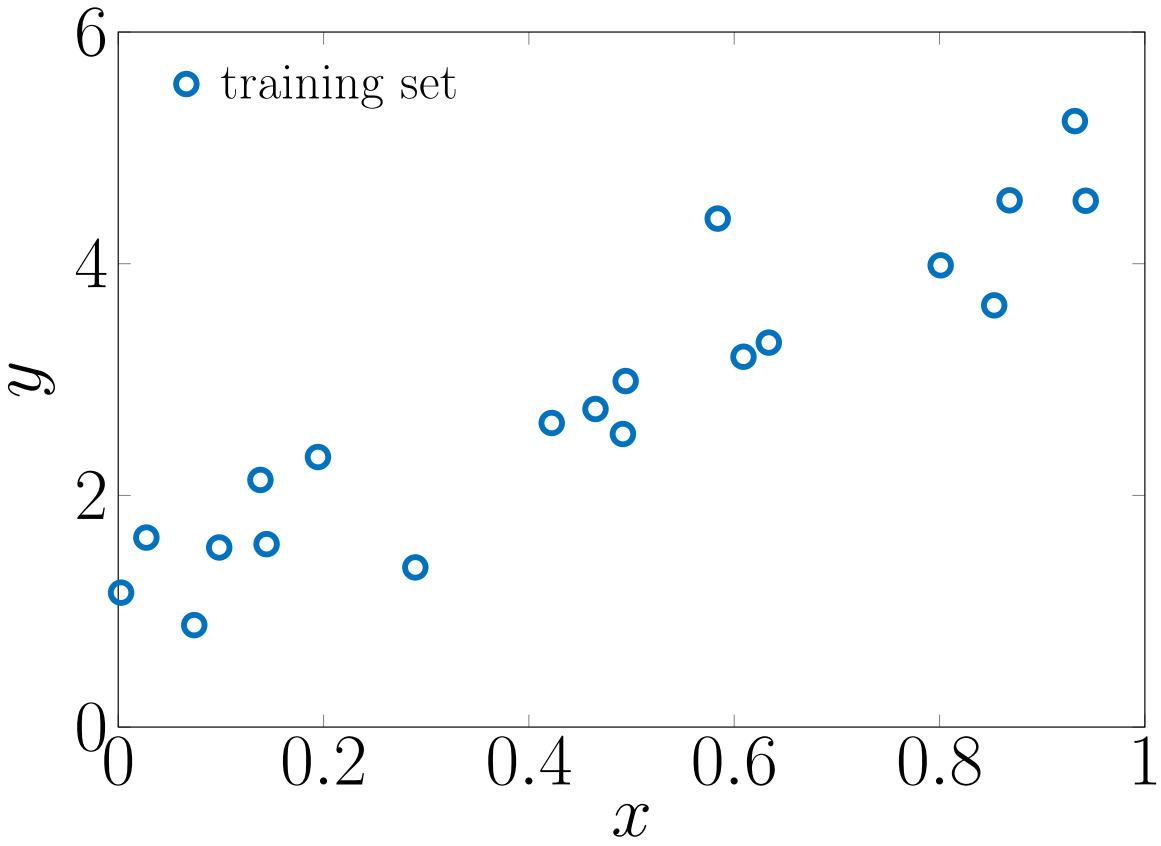
$$\mathbf{b}^* = \arg\min_{\mathbf{b}} \overbrace{\frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{b})^\intercal(\mathbf{y} - \mathbf{X}\mathbf{b})}^{J(\mathbf{b})}$$

$$\frac{\partial J(\mathbf{b}^*)}{\partial \mathbf{b}} \equiv 0 = \mathbf{X}^\intercal\mathbf{X}\mathbf{b}^* - \mathbf{X}^\intercal\mathbf{y}$$

$$\boxed{\boldsymbol{\theta}^* = \mathbf{b}^* = (\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{y}}$$

$$(\mathcal{D}_x \rightarrow \mathbf{X},\ \mathcal{D}_y = \mathbf{y})$$

Example #1 []

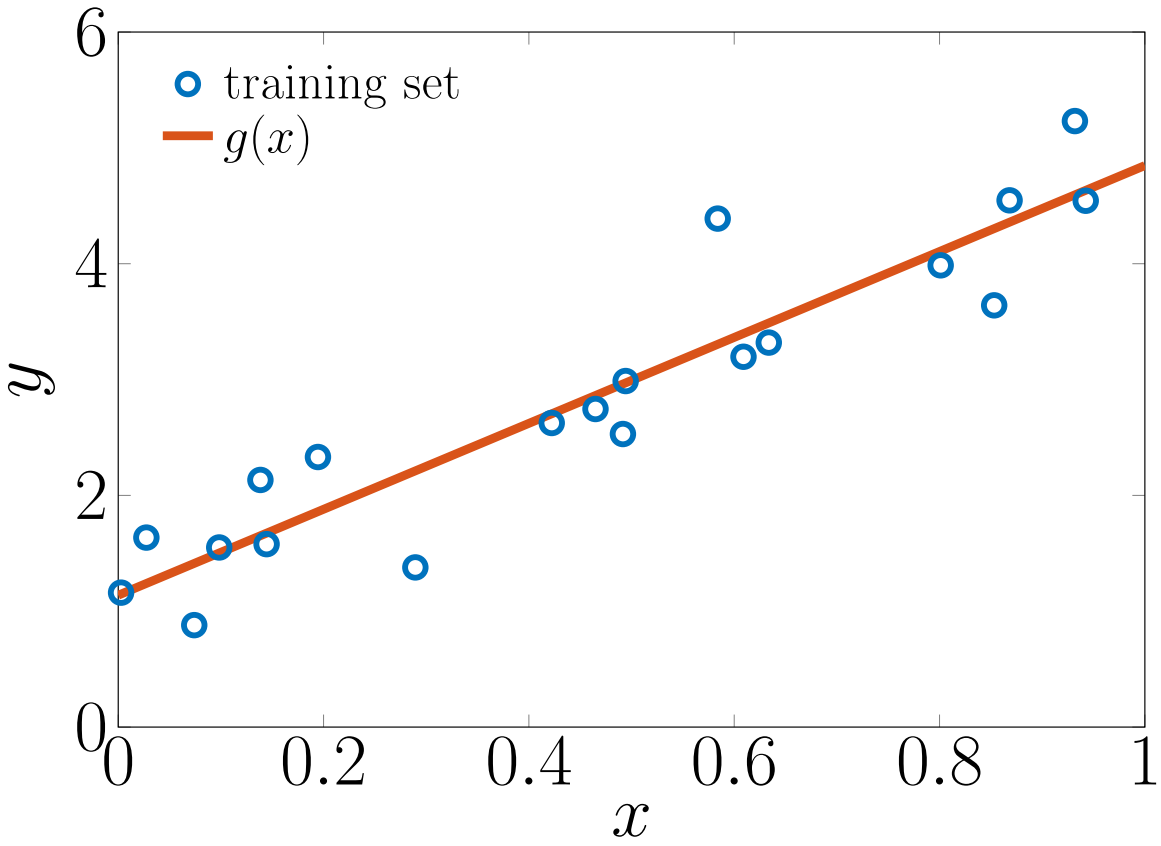


$$g(x) = b_0 + b_1 x$$

$$\mathbf{b}^* = [0.96, 4.09]^\intercal$$

[CIV_ML/Linear_regression_1D_demo.py]

Example #1 []

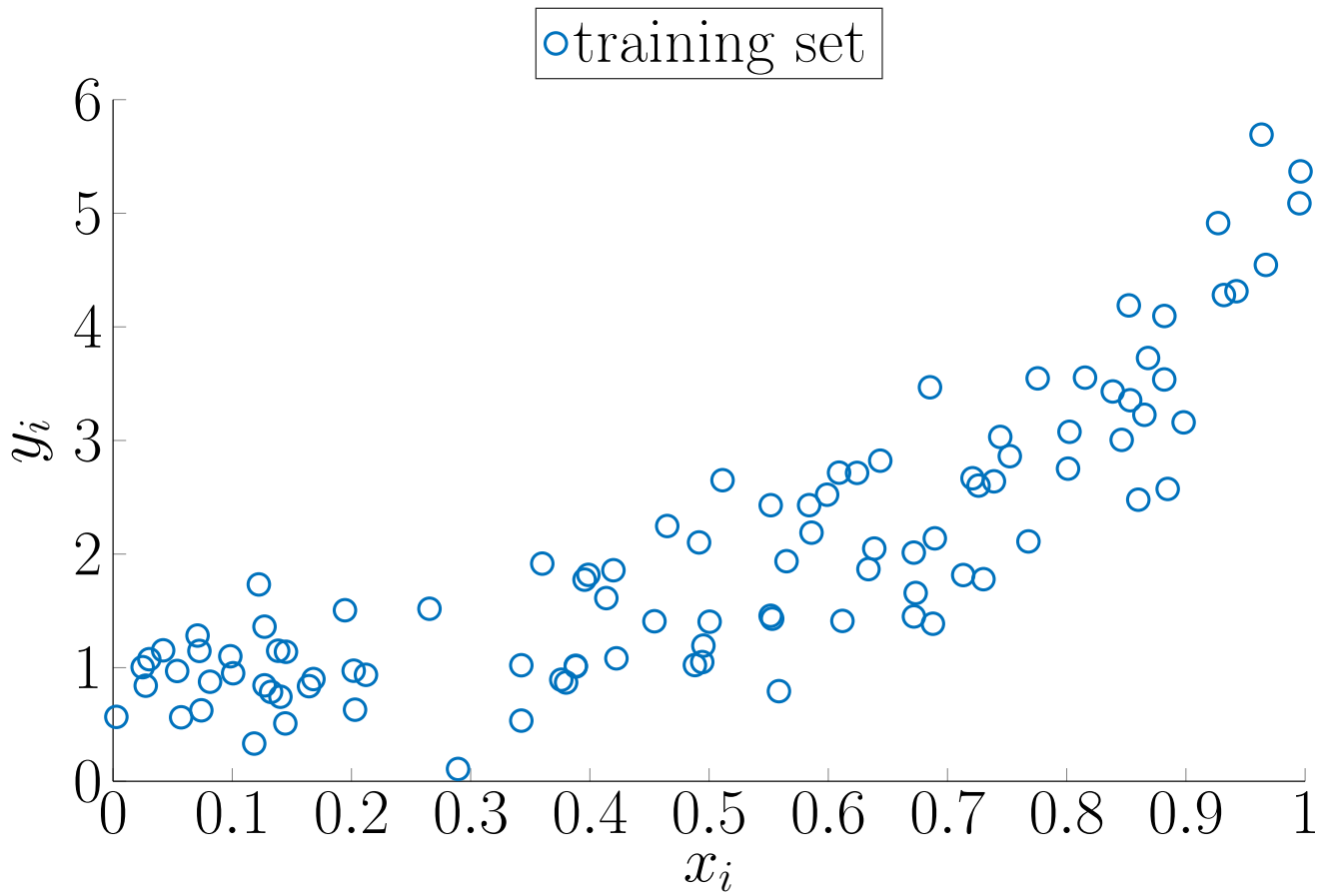


$$g(x) = b_0 + b_1 x$$

$$\mathbf{b}^* = [0.96, 4.09]^\intercal$$

[CIV_ML/Linear_regression_1D_demo.py]

What if the data is non-linear?



Mathematical formulation – general case

$$g(x) = b_0 + b_1\phi_1(x) + b_2\phi_2(x) + \cdots + b_B\phi_B(x) = \mathbf{Xb}$$

where $\phi_j(x)$ can be **any linear or non-linear function**,

$$\text{e.g.: } \phi_j(x) = x^2, \ \phi_j(x) = sin(x), \cdots$$

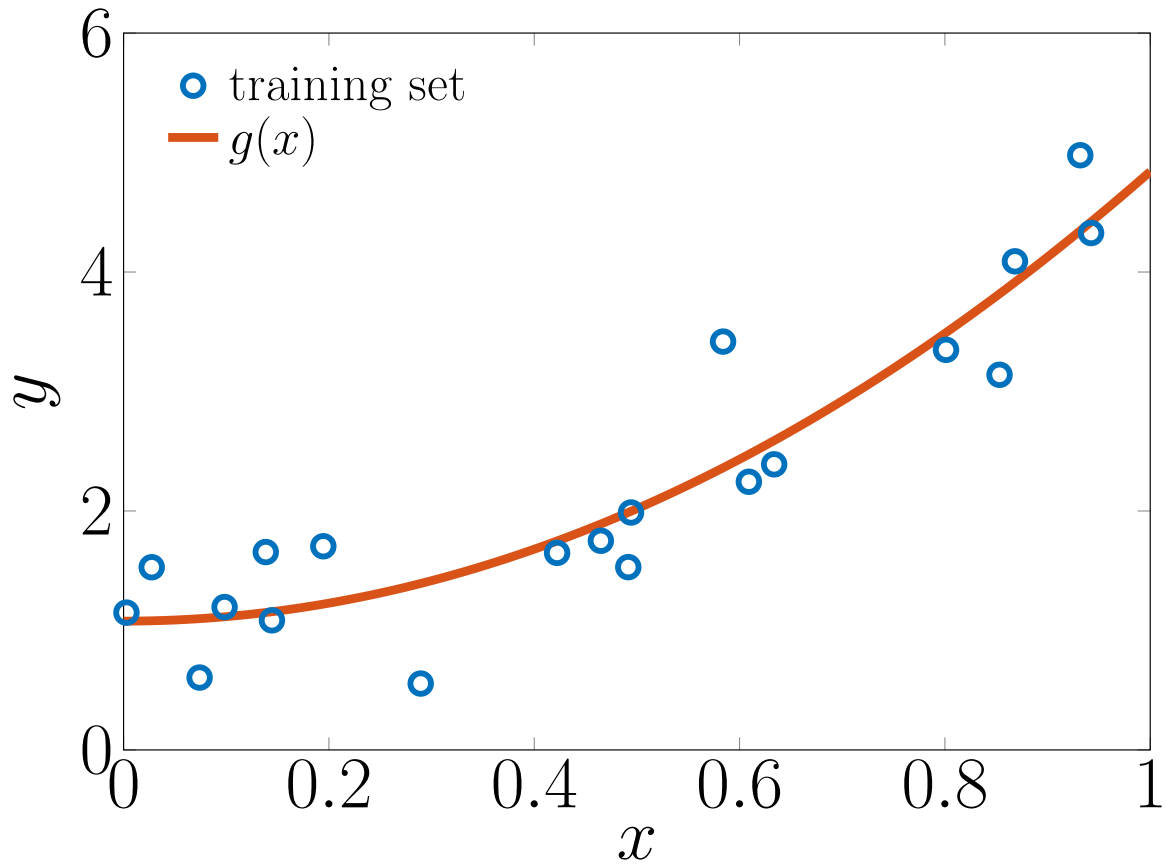
For the entire dataset

$$\mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_B \end{bmatrix}, \ \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_D \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} 1 & \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_B(x_1) \\ 1 & \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_B(x_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(x_D) & \phi_2(x_D) & \cdots & \phi_B(x_D) \end{bmatrix}$$

Linear regression: Linear with respect to $\phi_j(x)$ and b not x

Example #2a [🐍]

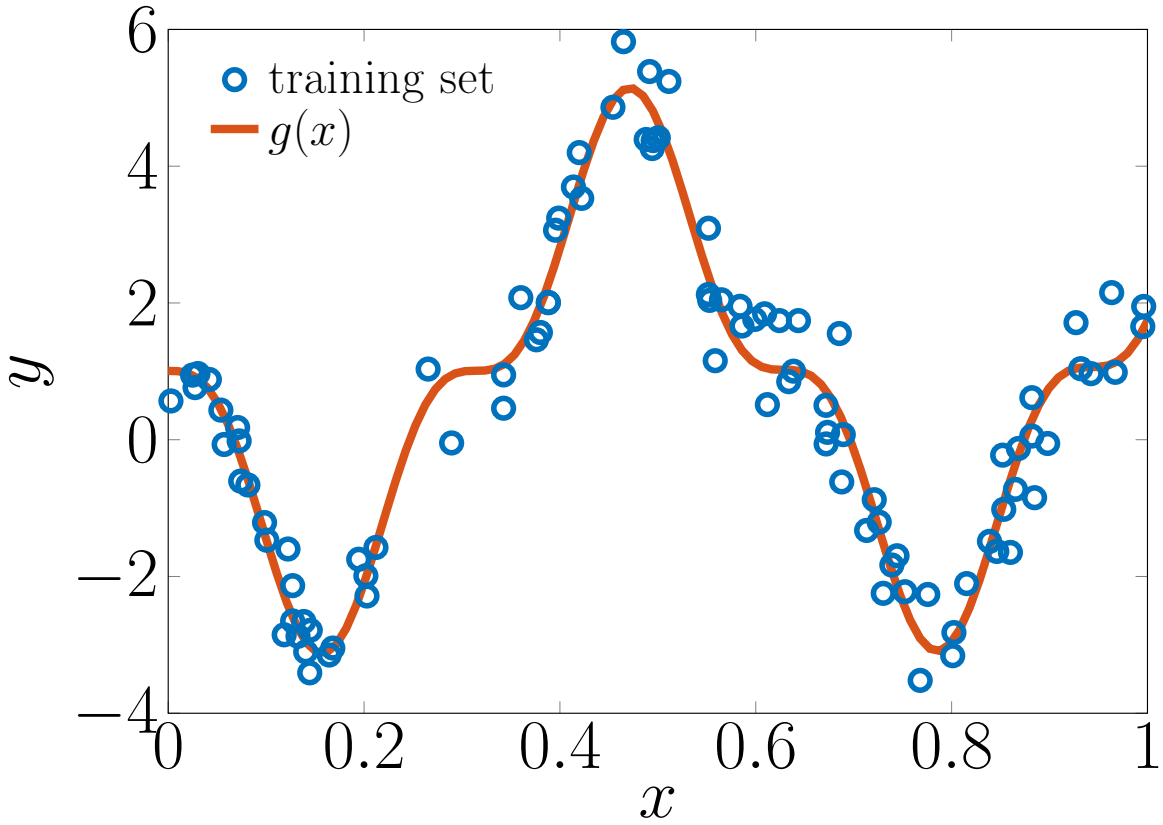
$$g(x) = b_0 + b_1 x^2$$



[CIV_ML/Linear_regression_1D.py]

Example #2b [🐍]

$$g(x) = b_0 + b_1 x^3 + b_2 \sin(10x)$$



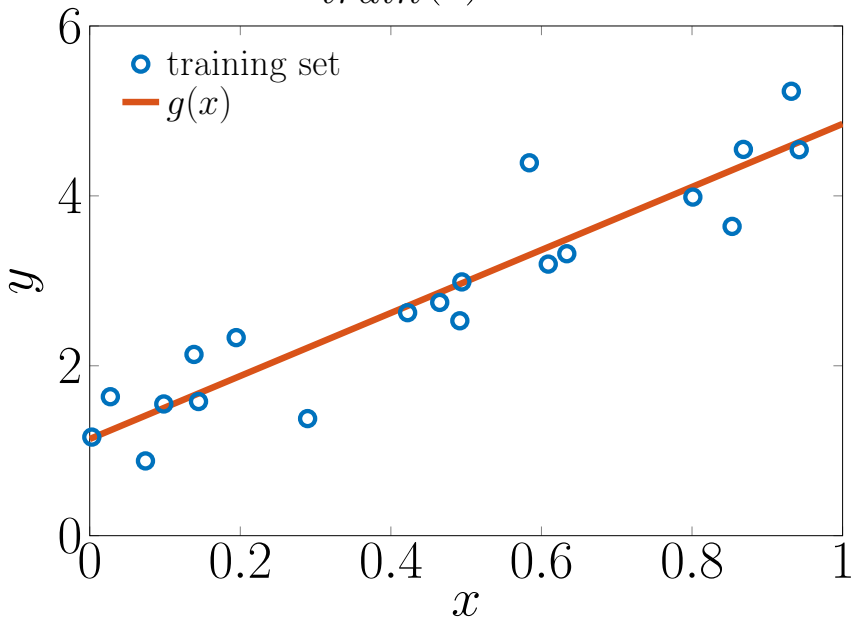
[CIV_ML/Linear_regression_1D.py]

What model structure is best?

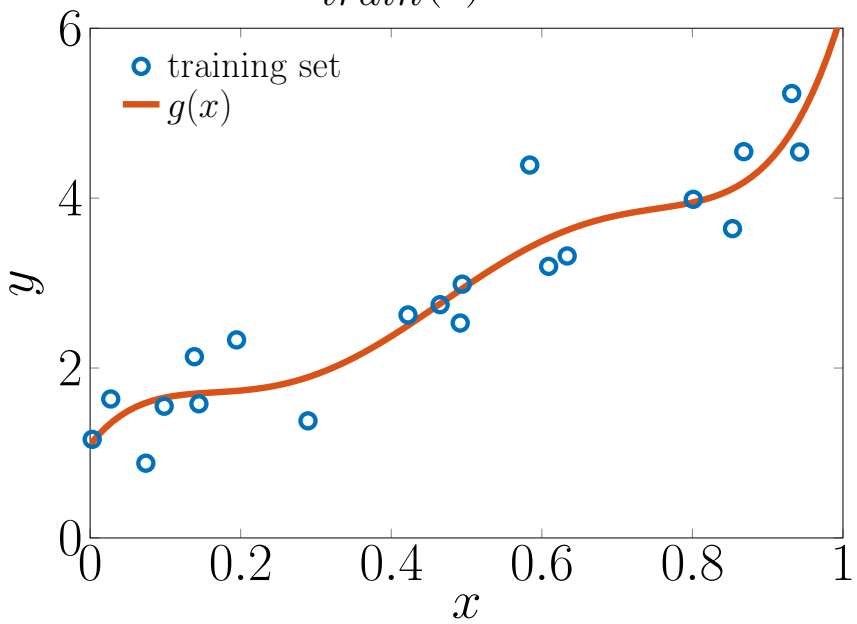
$$g(x) = b_0 + b_1 x$$

$$g(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4 + b_5 x^5$$

$J_{train}(b) = 1.98$



$J_{train}(b) = 1.72$



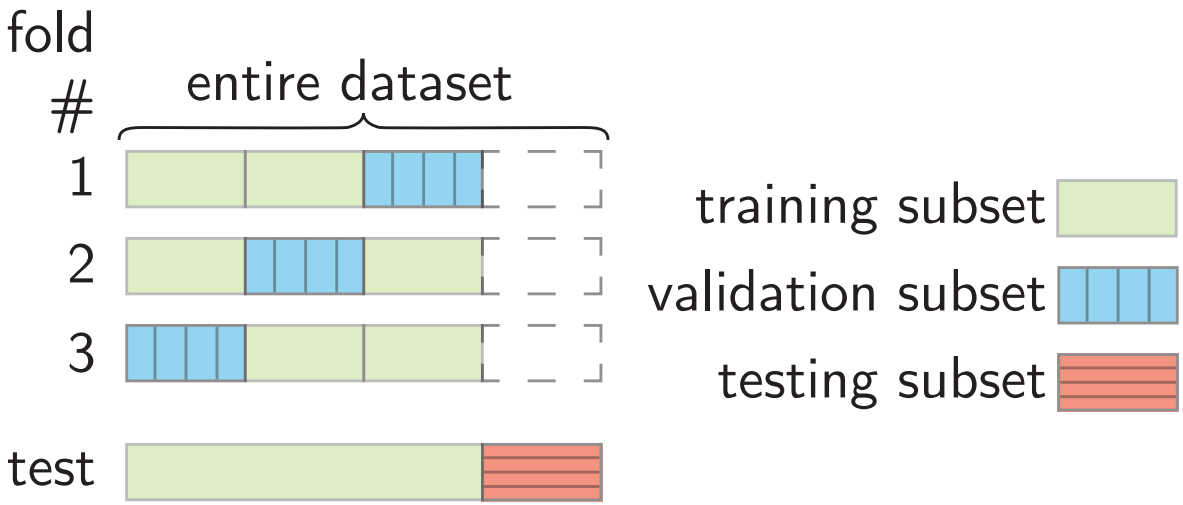
Linear regression is sensitive to over-fitting

Cross-validation – methodology

Over-fitting is prevented by employing cross-validation

Cross-validation: Do not employ the same data to train (i.e. estimate optimal parameters) and to evaluate the loss function.

***N*-fold cross-validation**: Split the shuffled data into N subsets that are employed one at a time as an independent validation set



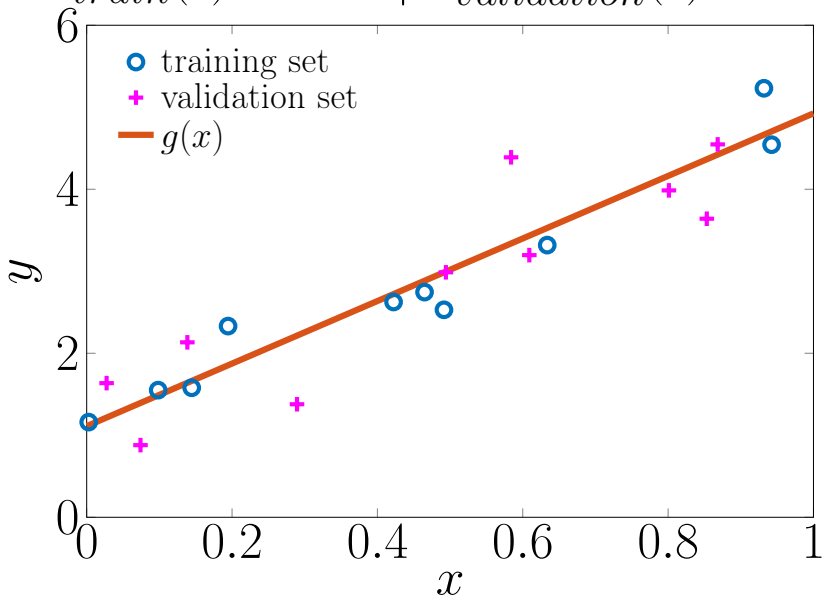
Algorithm 1: N-fold cross-validation

define $\mathbf{x}, \mathbf{y}$, $g_m(\mathbf{x})$
CV_idx $=$ randperm(D)
nCV $=$ round(D/N)
for $m = 1, 2, \cdots,$ M **do**
    for $i = 1, 2, \cdots,$ N **do**
        val_idx $= CV\_idx((i-1) \times nCV + 1 : \min(\mathtt{X}, i \times nCV))$
        x_train $= \mathbf{x}$, x_train(val_idx,:) $= []$
        y_train $= \mathbf{y}$, y_train(val_idx,:) $= []$
        x_val $= \mathbf{x}$(val_idx,:)
        y_val $= \mathbf{y}$(val_idx,:)
        Estimate $\mathbf{b}_i^* = (\mathbf{X}_{train}^\intercal \mathbf{X}_{train})^{-1} \mathbf{X}_{train}^\intercal \mathbf{y}_{train}$
        Compute $J_i(\mathbf{b}^*, \text{x\_val}, \text{y\_val})$
    $J_{val}^m = \sum_{i=1}^{\mathtt{N}} J_i$
Select $m^* : J_{val}^{m^*} = \min J_{val}^m$
Estimate $\mathbf{b}^* = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{y}$

Example #3 – Cross-validation [🐍]

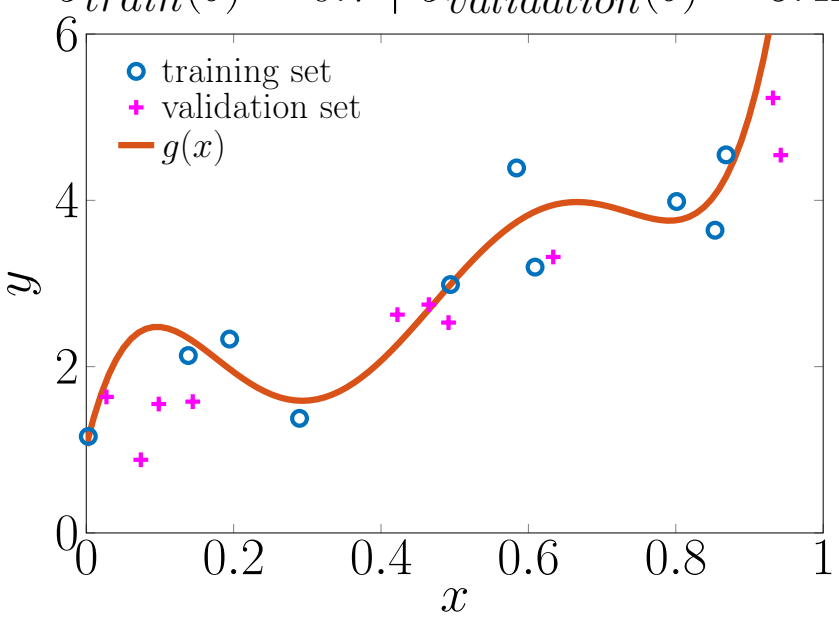
$$g(x) = b_0 + b_1 x \qquad g(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4 + b_5 x^5$$

$J_{train}(b) = 0.43 \mid J_{validation}(b) = 1.56$



$$\left.\begin{aligned} \sum J_{train} &= 1.97 \\ \sum J_{validation} &= 2.02 \end{aligned}\right\} \checkmark$$

$J_{train}(b) = 0.7 \mid J_{validation}(b) = 5.42$



$$\left.\begin{aligned} \sum J_{train} &= 0.81 \\ \sum J_{validation} &= 20.89 \end{aligned}\right\} \times$$

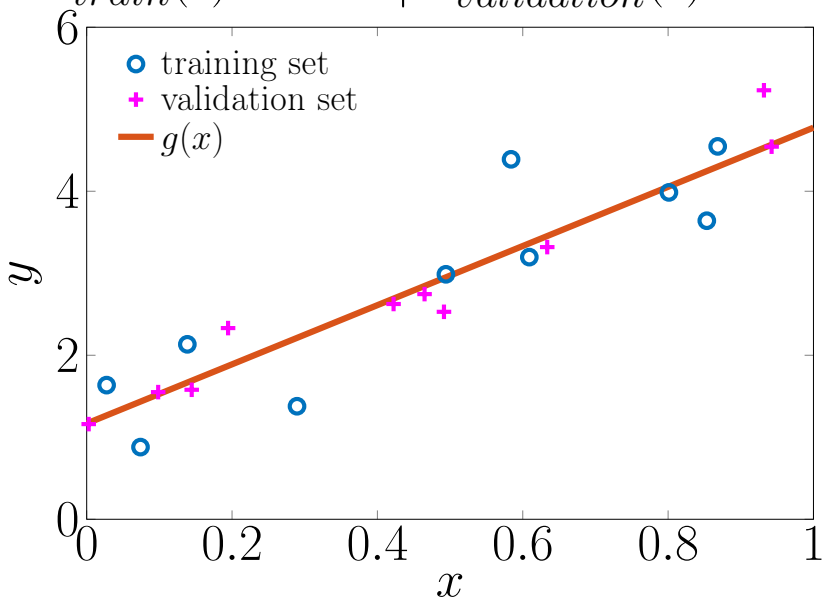
[CIV_ML/Linear_regression_1D.py]

Example #3 – Cross-validation [🐍]

$$g(x) = b_0 + b_1 x$$

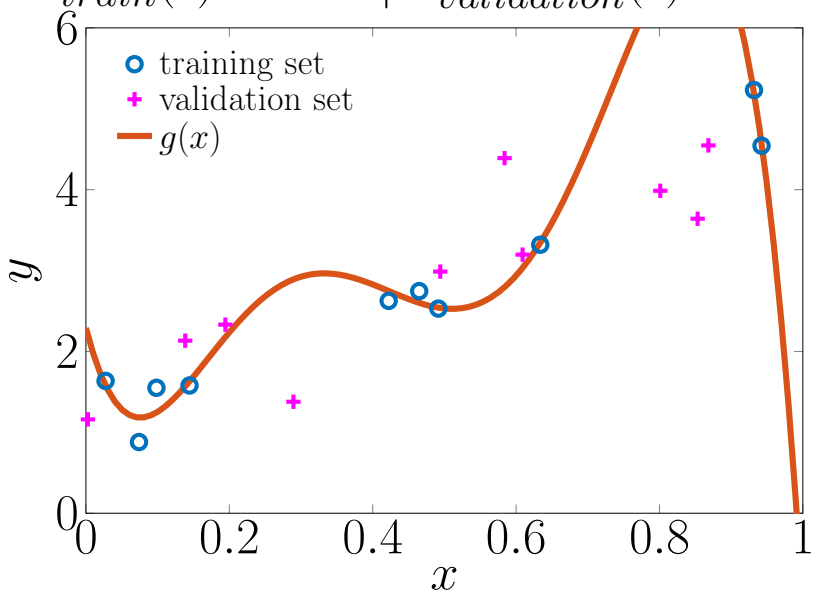
$$g(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4 + b_5 x^5$$

$J_{train}(b) = 1.53 \mid J_{validation}(b) = 0.46$



$$\left.\begin{aligned} \sum J_{train} &= 1.97 \\ \sum J_{validation} &= 2.02 \end{aligned}\right\} \checkmark$$

$J_{train}(b) = 0.11 \mid J_{validation}(b) = 15.47$



$$\left.\begin{aligned} \sum J_{train} &= 0.81 \\ \sum J_{validation} &= 20.89 \end{aligned}\right\} \times$$

[CIV_ML/Linear_regression_1D.py]

N-fold cross-validation

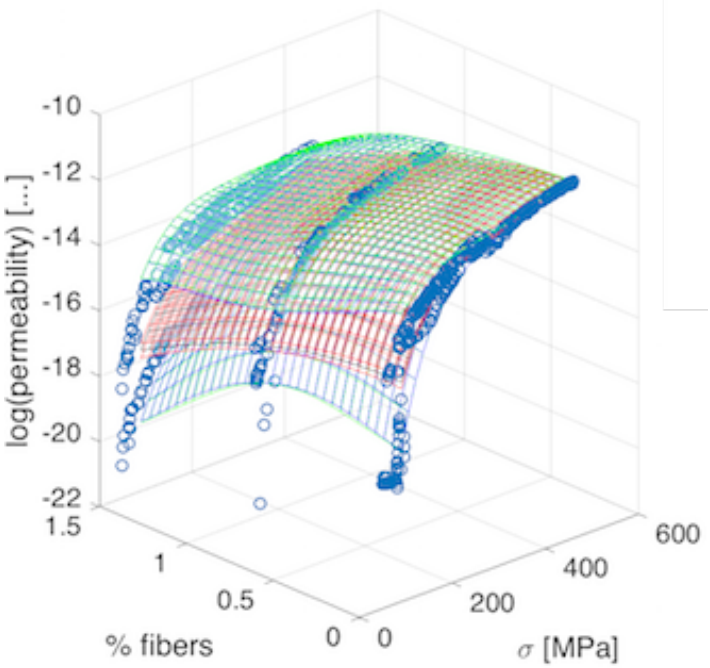
The greater is the number of folds N, the more accurate are results.

If $N = \mathtt{D}$ it is called **leave-one-out cross-validation (LOOCV)**

In practice LOOCV too computationally demanding when $\mathtt{D}$ is large because it requires as many calibration loops

Instead in practice, it is common to employ
5 to 20-fold cross-validation

Regression in > 1D



Data

$$\mathcal{D} = \{(\mathbf{x}_i, y_i), \forall i = 1 : \mathtt{D}\}$$

$$\mathbf{x}_i = [x_1, x_2]_i^\intercal \in \mathbb{R}^2 : \begin{cases} \text{Covariates} \\ \text{attributes} \\ \text{regressors} \end{cases}$$

$y_i \in \mathbb{R}^1$: Observation

Model

$g(\mathbf{x}) \equiv \text{fct}(\mathbf{x}) \in \mathbb{R}^1$

General case: $\mathbf{x}_i = [x_1, x_2, \cdots, x_{\mathtt{X}}]_i^\intercal \in \mathbb{R}^{\mathtt{X}}, \; y_i \in \mathbb{R}^1, \; g(\mathbf{x}_i) \in \mathbb{R}^1$

Mathematical formulation – general case >1D

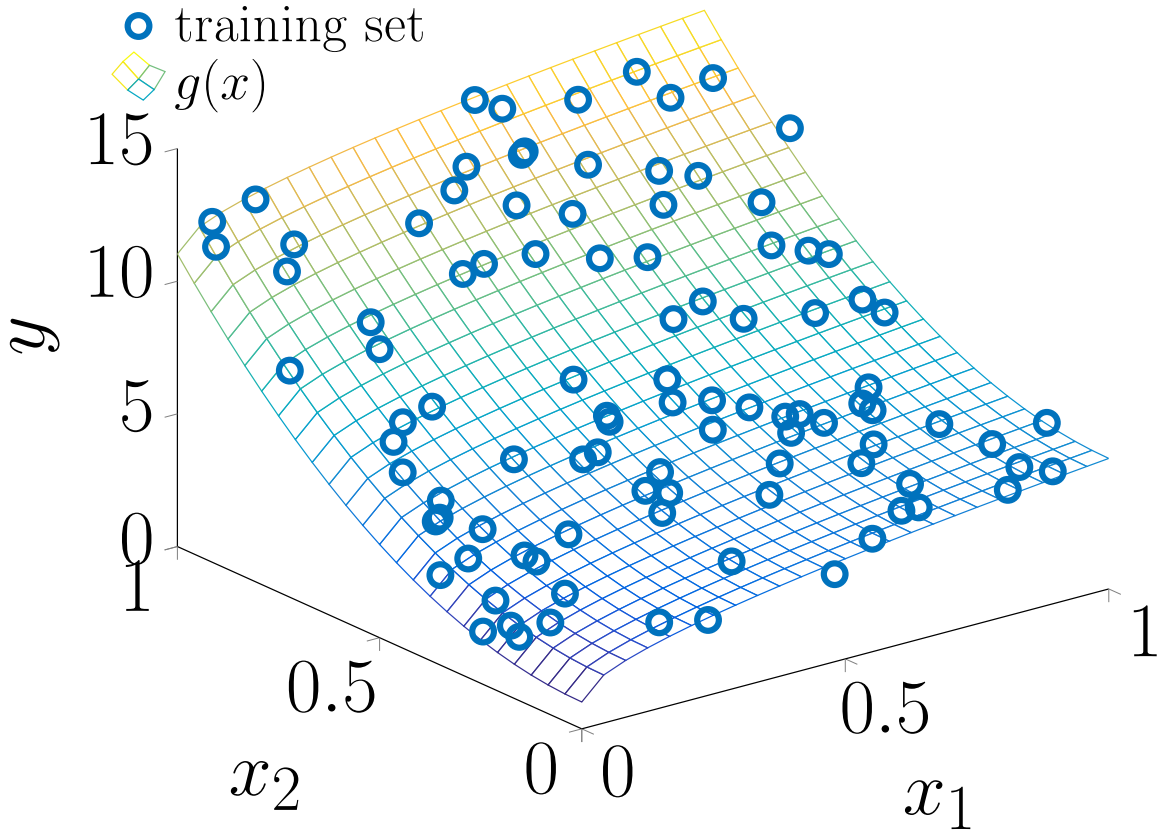
$$g(\mathbf{x}) = b_0 + b_1\phi_1(x_1) + b_2\phi_2(x_2) + \cdots = \mathbf{Xb}$$

For the entire dataset

$$\mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_B \end{bmatrix}, \ \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_D \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} 1 & \phi_1(x_{1,1}) & \phi_2(x_{1,2}) & \cdots \\ 1 & \phi_1(x_{2,1}) & \phi_2(x_{2,2}) & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ 1 & \phi_1(x_{D,1}) & \phi_2(x_{D,2}) & \cdots \end{bmatrix}$$

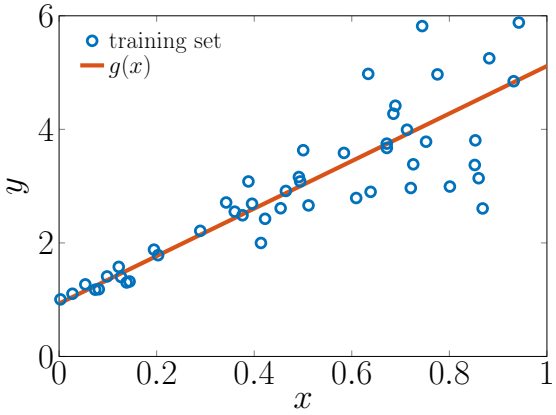
Example #4 []

$$g(\mathbf{x}) = b_0 + b_1 x_1^{1/2} + b_2 x_2^2$$

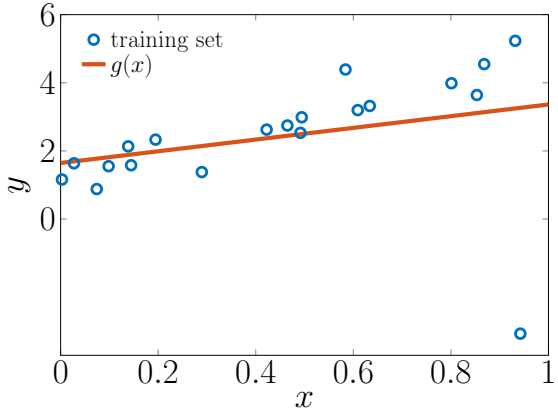


[CIV_ML/Linear_regression_2D.py]

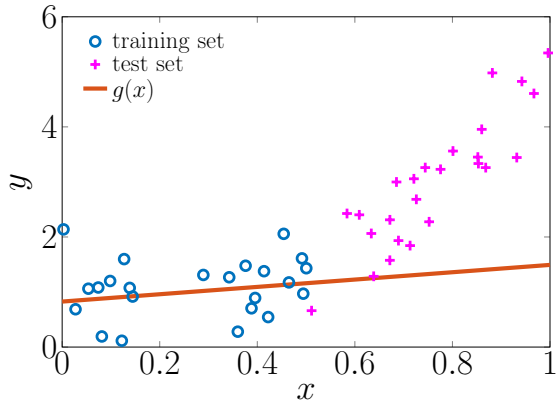
Limitations



Heteroscedasticity



Outliers



Extrapolation

Limitations

- Assumes **homoscedastic** errors
- Sensitive to **outliers**
- Does not differentiate between **interpolating and extrapolating**
- ⚠ The performance depends on the capacity to **hand-pick the correct transformation functions** (Difficult when $D > 1$)

Summary

In short: Linear regression is simple and quick, yet it is **outperformed by modern techniques** such as **Gaussian Process Regression**

Note: Cross-validation is not limited to linear regression and can be employed with any regression technique

Example – T° distribution along a pipeline

Given that we know the pipeline temperature to be $y = 8°C$ at $x = 52\,km$.

What is the temperature at $x = 52.1\,km$?

What is the temperature at $x = 70\,km$?

We will take advantage of the **conditional dependency between a system response y and covariate values x**

[Bloomberg]

Section Outline

Gaussian Process Regression

Definition

Given a **system response** so that

$$\underbrace{g_i}_{\text{observation}} = \overbrace{g(\mathbf{x}_i)}^{\text{reality}}, \quad \underbrace{\mathbf{x}_i = [x_1, x_2, \cdots, x_{\mathtt{X}}]_i^{\mathsf{T}}}_{\text{covariates}}$$

Data: $\mathcal{D} = \{(\mathbf{x}_i, g_i), \forall i = 1 : \mathtt{D}\}$

$$\mathbf{g} = [g_1, g_2, \cdots, g_{\mathtt{D}}]^{\mathsf{T}}, \quad \mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{\mathtt{D}}]_{\mathtt{X} \times \mathtt{D}}$$

Gaussian process: $g(\mathbf{x}) : \mathbf{G} \sim \mathcal{N}(g(\mathbf{x}); \boldsymbol{\mu}_{\mathbf{G}}, \boldsymbol{\Sigma}_{\mathbf{G}})$

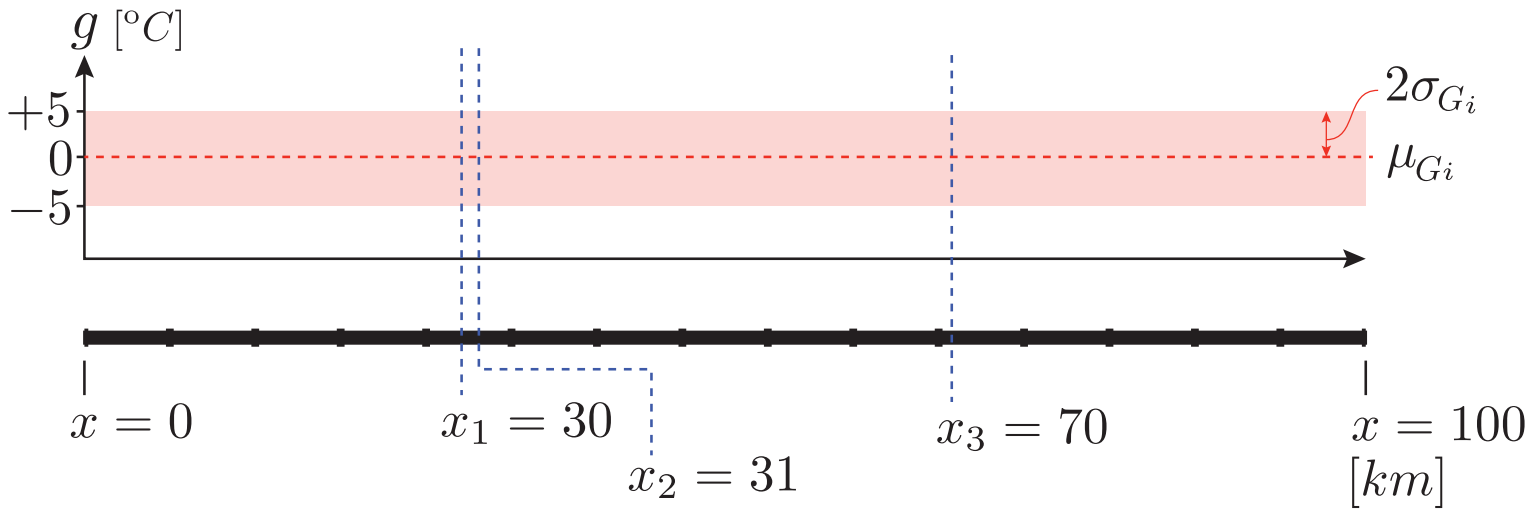
Set of discrete Gaussian random variables for which the pairwise correlation between G_i and G_j is a function of the distance between attributes

$$[\boldsymbol{\Sigma}_{\mathbf{G}}]_{ij} = \rho(x_i, x_j)\sigma_{G_i} \cdot \sigma_{G_j}, \quad \rho(x_i, x_j) = \text{fct}(|x_i - x_j|)$$

Example - T° distribution along a pipeline

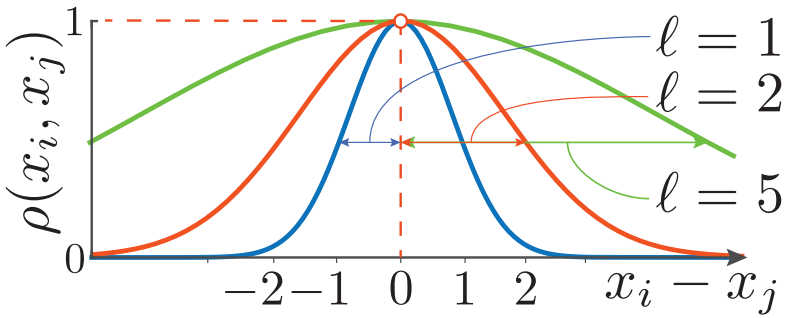
[Bloomberg]

Example - T° distribution along a pipeline



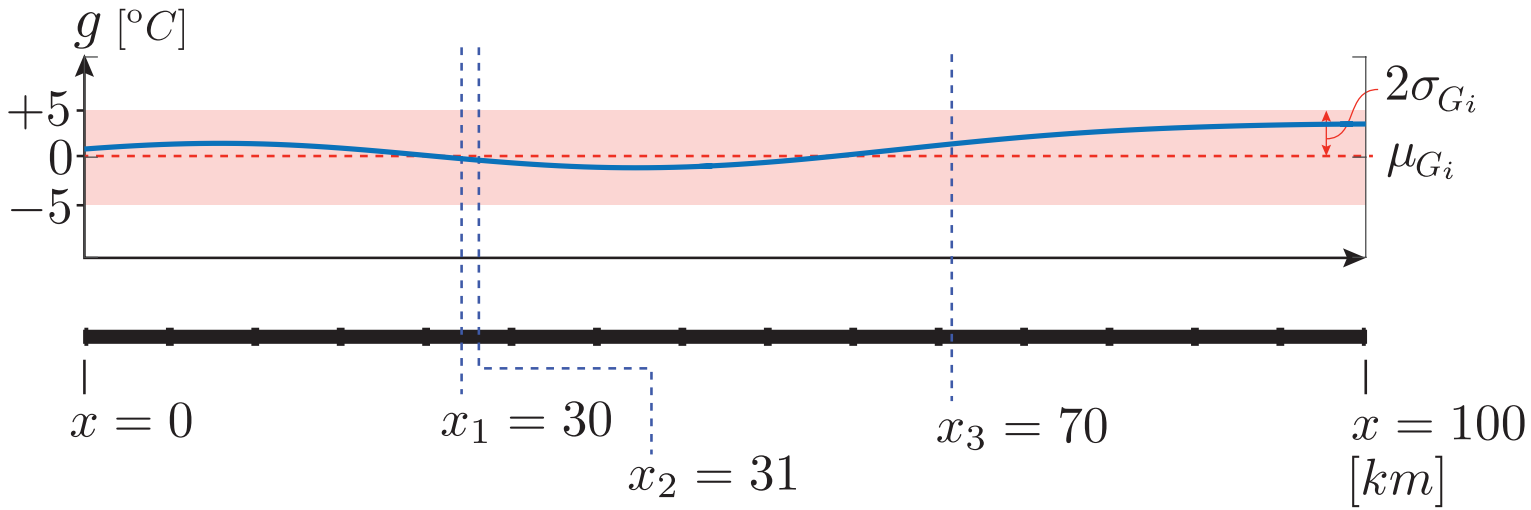
Prior knowledge: $\mu_{G_i} = \mu_G = 0°C$, $\sigma_{G_i} = \sigma_G = 2.5°C$

Gaussian correlation function: $\rho(x_i, x_j) = \exp\left(-\frac{1}{2}\frac{(x_i - x_j)^2}{\ell^2}\right)$



for $\ell = 25\ km$ $\begin{cases} \rho(x_1, x_2) = 0.999 \\ \rho(x_1, x_3) = 0.278 \\ \rho(x_2, x_3) = 0.296 \end{cases}$

Example - T° distribution along a pipeline [🐍]



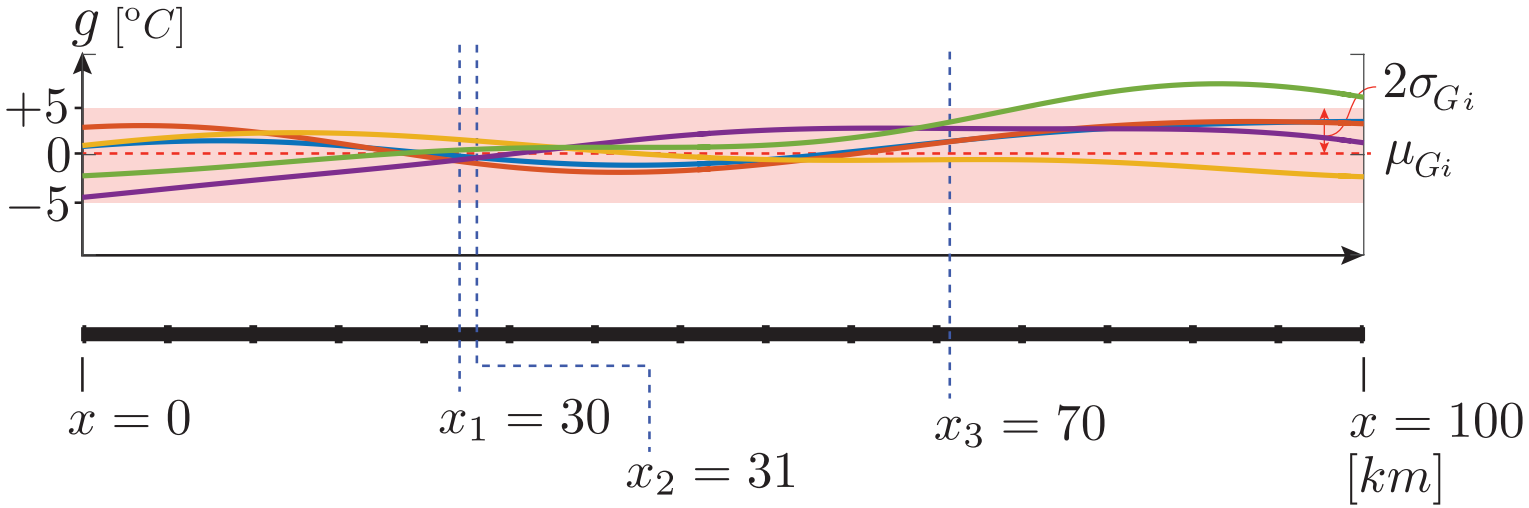
Pour $\mathbf{x} = [0, 1, \cdots, 100]^\intercal$

$$\overbrace{\mathbf{G} \sim \mathcal{N}(g(\mathbf{x}); \boldsymbol{\mu}_\mathbf{G}, \boldsymbol{\Sigma}_\mathbf{G}), \quad [\boldsymbol{\Sigma}_\mathbf{G}]_{ij} = \rho(x_i, x_j)\sigma^2_{G_i}}^{\text{Prior knowledge}}$$

$\mathbf{g_i(x)}$: realizations of our prior knowledge $\mathbf{G}$

[CIV_ML/M14_GP_pipeline.py]

Example - T° distribution along a pipeline [🐍]



Pour $\mathbf{x} = [0, 1, \cdots, 100]^\intercal$

$$\overbrace{\mathbf{G} \sim \mathcal{N}(g(\mathbf{x}); \boldsymbol{\mu}_\mathbf{G}, \boldsymbol{\Sigma}_\mathbf{G}), \quad [\boldsymbol{\Sigma}_\mathbf{G}]_{ij} = \rho(x_i, x_j)\sigma^2_{G_i}}^{\text{Prior knowledge}}$$

$\mathbf{g_i(x)}$: realizations of our prior knowledge $\mathbf{G}$

[CIV_ML/M14_GP_pipeline.py]

Updating a GP using exact observations

Given $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{g}_i),\ i = 1, \cdots \mathtt{D}\}$ a set of $\mathtt{D}$ observations and $\mathbf{x}_*$ a set of $\mathtt{X}_*$ covariates for which we want to predict

$$f(\mathbf{g}_*|\mathbf{x}_*, \mathcal{D})$$

Reminder: Gaussian conditionals are also gaussian

$$\underbrace{\left\{ \begin{array}{c} \mathbf{G} \\ \mathbf{G}_* \end{array} \right\},\ \boldsymbol{\mu} = \left\{ \begin{array}{c} \boldsymbol{\mu}_\mathbf{G} \\ \hline \boldsymbol{\mu}_* \end{array} \right\},\ \boldsymbol{\Sigma} = \left[\begin{array}{c|c} \boldsymbol{\Sigma}_\mathbf{G} & \boldsymbol{\Sigma}_{\mathbf{G}*} \\ \hline \boldsymbol{\Sigma}_{\mathbf{G}*}^\mathsf{T} & \boldsymbol{\Sigma}_* \end{array} \right]}_{\text{Prior knowledge}}$$

$$[\boldsymbol{\Sigma}_\mathbf{G}]_{ij} = \rho(x_i, x_j)\sigma_G^2$$
$$[\boldsymbol{\Sigma}_{\mathbf{G}*}]_{ij} = \rho(x_i, x_{*j})\sigma_G^2$$
$$[\boldsymbol{\Sigma}_*]_{ij} = \rho(x_{*i}, x_{*j})\sigma_G^2$$

$$f_{\mathbf{G}_*|\mathbf{x}_*,\mathcal{D}}(\mathbf{g}_*|\mathbf{x}_*, \mathcal{D}) = \mathcal{N}(\mathbf{g}_*; \boldsymbol{\mu}_{*|\mathcal{D}}, \boldsymbol{\Sigma}_{*|\mathcal{D}})$$

$$\underbrace{\begin{array}{l} \boldsymbol{\mu}_*|\mathcal{D} \equiv \boldsymbol{\mu}_{*|\mathcal{D}} = \boldsymbol{\mu}_{\mathbf{G}_*} + \boldsymbol{\Sigma}_{\mathbf{G}*}^\mathsf{T}\boldsymbol{\Sigma}_\mathbf{G}^{-1}(\mathbf{g} - \boldsymbol{\mu}_\mathbf{G}) \\ \boldsymbol{\Sigma}_*|\mathcal{D} \equiv \boldsymbol{\Sigma}_{*|\mathcal{D}} = \boldsymbol{\Sigma}_* - \boldsymbol{\Sigma}_{\mathbf{G}*}^\mathsf{T}\boldsymbol{\Sigma}_\mathbf{G}^{-1}\boldsymbol{\Sigma}_{\mathbf{G}*} \end{array}}_{\text{Posterior knowledge}}$$

Example – Multivariate gaussian conditional

Soit $\underbrace{\mu_G = 0^\circ C,\ \sigma_G = 2.5^\circ C,\ \rho(x_1, x_2) = 0.8}_{\text{prior knowledge}}$, $\underbrace{g_2 = -2^\circ C}_{\text{observation}}$

$$f_{G_1 G_2}(g_1, g_2) = \mathcal{N}(\boldsymbol{\mu_G}, \boldsymbol{\Sigma_G}) \begin{cases} \boldsymbol{\mu_G} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \boldsymbol{\Sigma_G} = \begin{bmatrix} 2.5^2 & 0.8 \cdot 2.5^2 \\ 0.8 \cdot 2.5^2 & 2.5^2 \end{bmatrix} \end{cases}$$

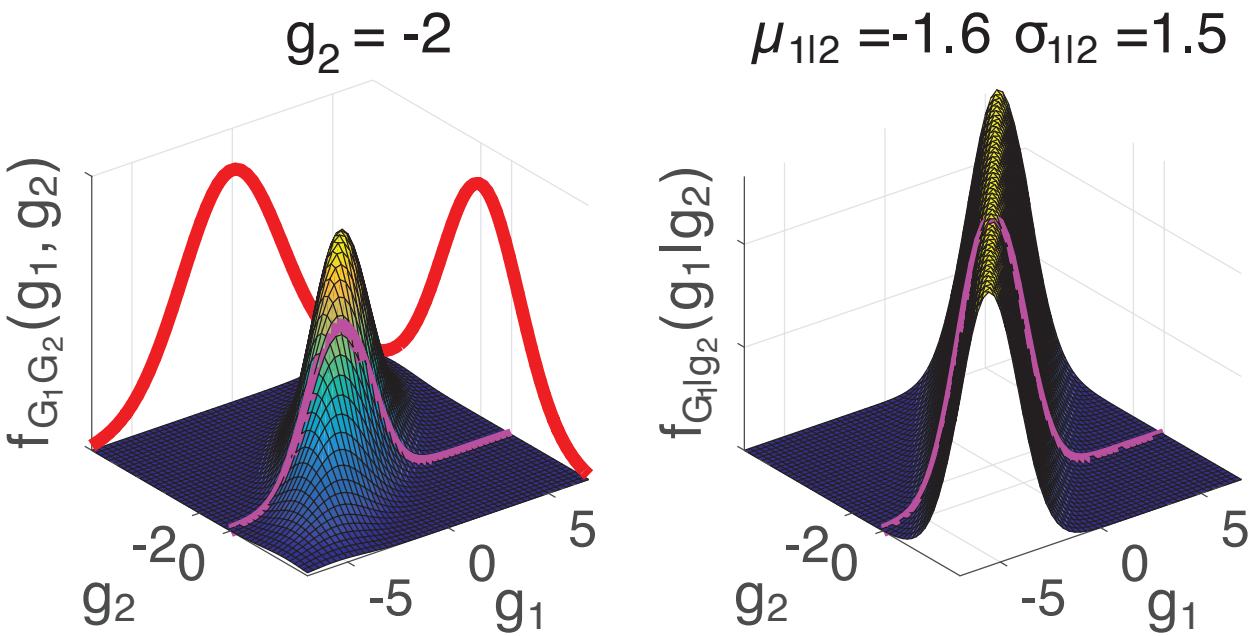
$$f_{G_1|g_2}(g_1|g_2) = \frac{f_{G_1 G_2}(g_1, g_2)}{f_{G_2}(g_2)} = \mathcal{N}(\mu_{1|2}, \sigma^2_{1|2}) \begin{cases} \mu_{1|2} = \mu_1 + \rho\sigma_1 \frac{g_2 - \mu_2}{\sigma_2} \\ \sigma_{1|2} = \sigma_1 \sqrt{1 - \rho^2} \end{cases}$$

$$\mu_{1|2} = 0 + 0.8 \times 2.5 \frac{-2 - 0}{2.5} = -1.6^\circ C$$

$$\sigma_{1|2} = 2.5\sqrt{1 - 0.8^2} = 1.5^\circ C$$

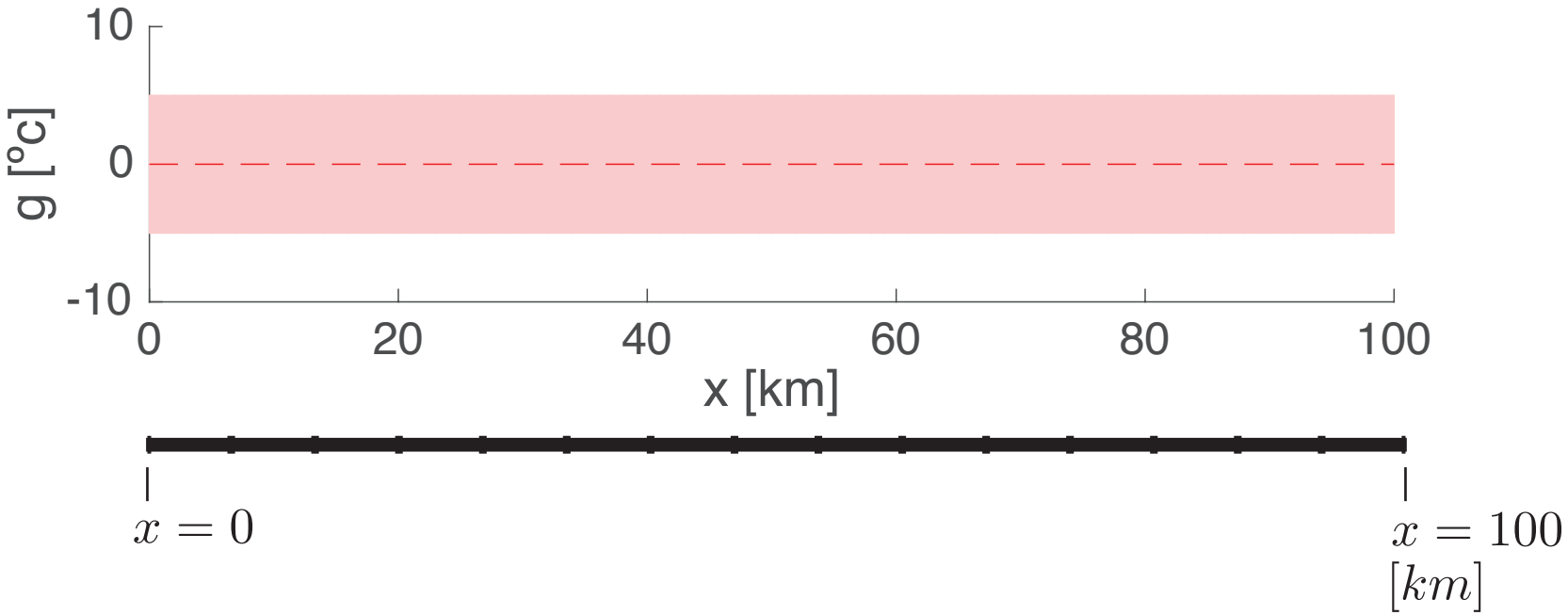
Example – Multivariate gaussian conditional [🐍]

$$f_{G_1|g_2}(g_1|g_2) = \frac{f_{G_1G_2}(g_1,g_2)}{f_{G_2}(g_2)} = \mathcal{N}(\mu_{1|2}, \sigma^2_{1|2}) \begin{cases} \mu_{1|2} = \mu_1 + \rho\sigma_1 \frac{y_2-\mu_2}{\sigma_2} \\ \sigma_{1|2} = \sigma_1\sqrt{1-\rho^2} \end{cases}$$



[CIV_ML/M14_Gauss_conditionnelle.py]

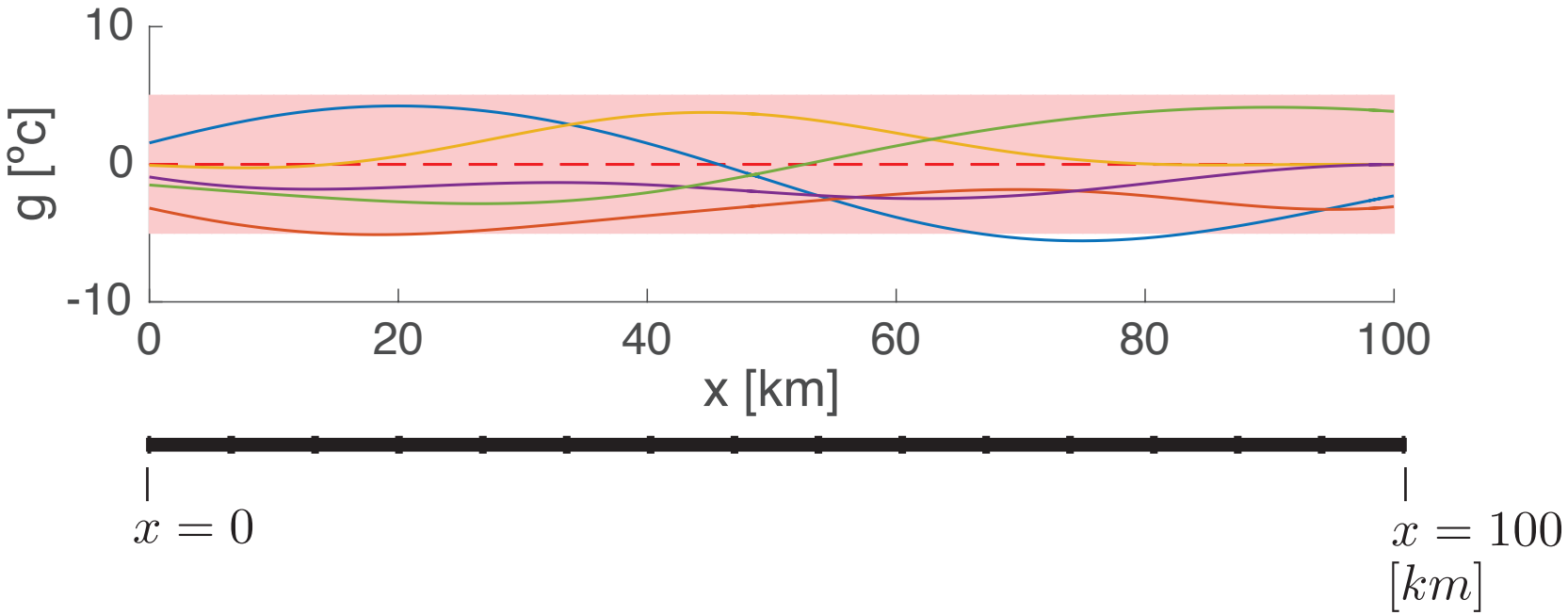
Example - T° distribution along a pipeline [🐍]



Observations: $\mathcal{D} = \{(\underbrace{[\,]}_{\mathbf{x}}, \underbrace{[\,]}_{\mathbf{g}})\}$

[CIV_ML/M14_GP_pipeline_obs.py]

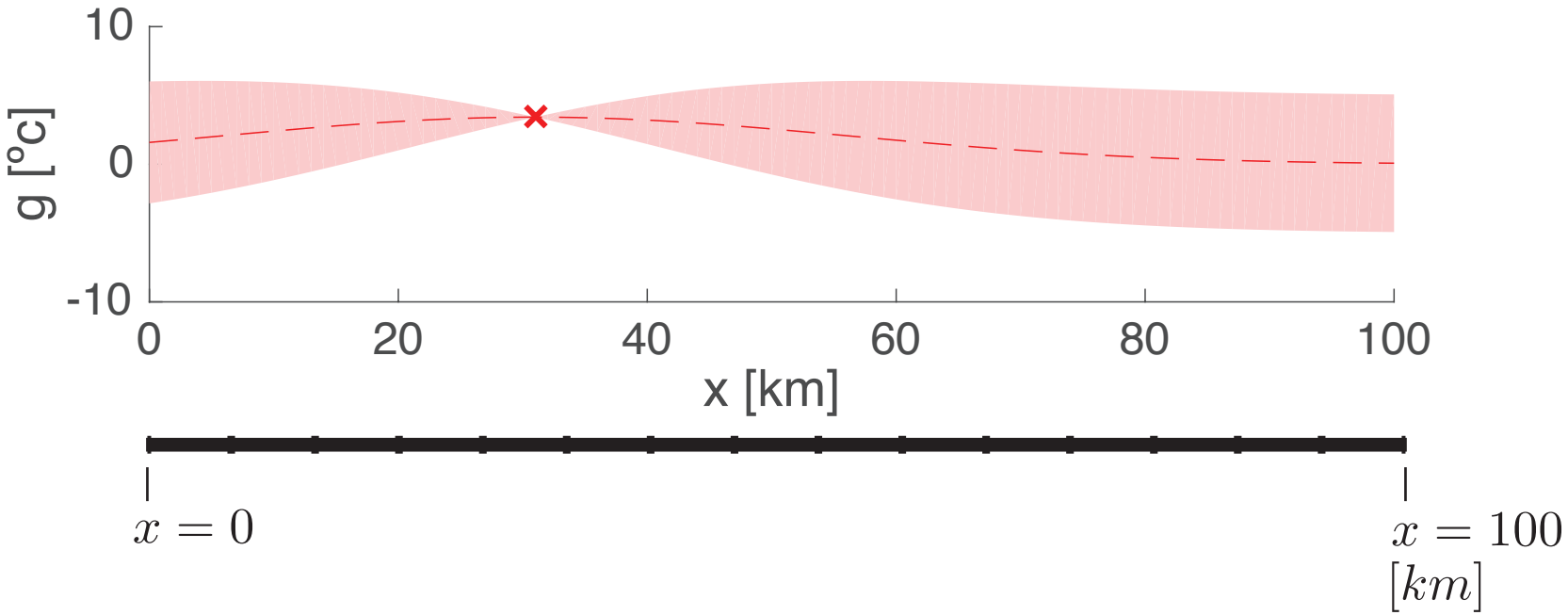
Example - T° distribution along a pipeline [🐍]



Observations: $\mathcal{D} = \{(\underbrace{[\,]}_{\mathbf{x}}, \underbrace{[\,]}_{\mathbf{g}})\}$

[CIV_ML/M14_GP_pipeline_obs.py]

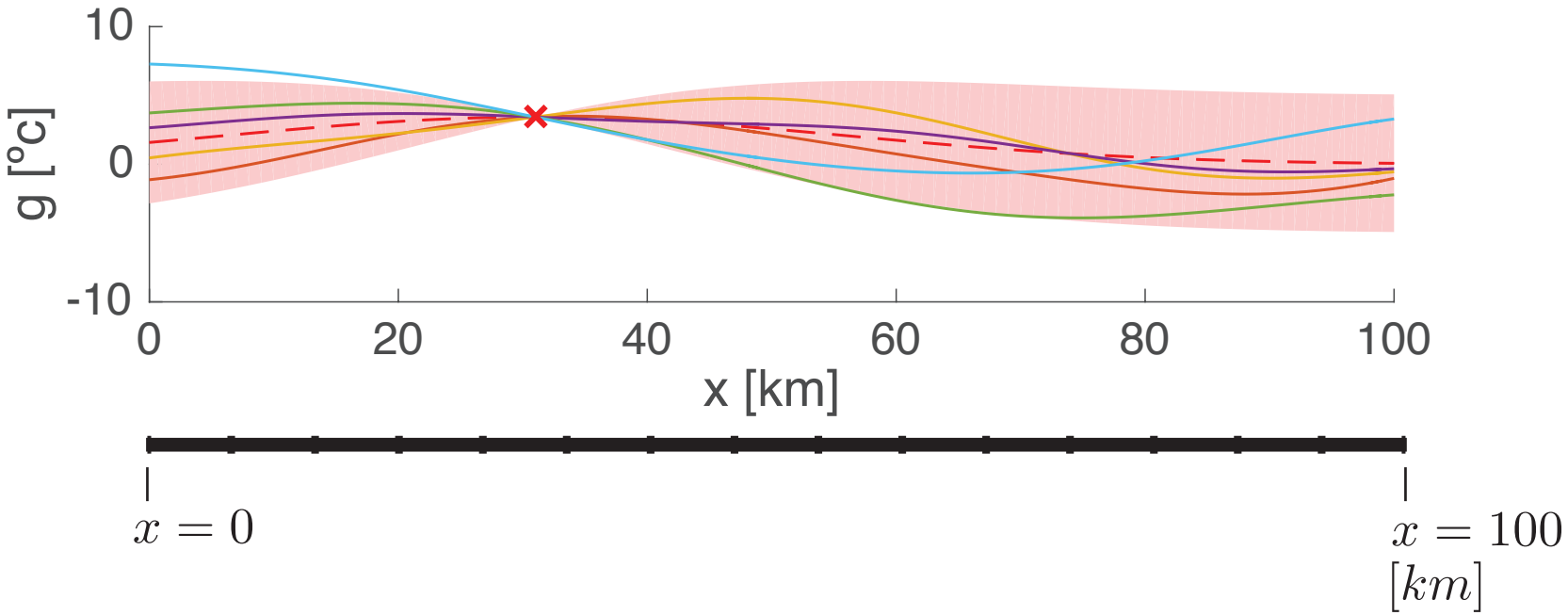
Example - T° distribution along a pipeline [🐍]



Observations: $\mathcal{D} = \{(\underbrace{[31]}_{\mathbf{x}}, \underbrace{[4.13]}_{\mathbf{g}})\}$

[CIV_ML/M14_GP_pipeline_obs.py]

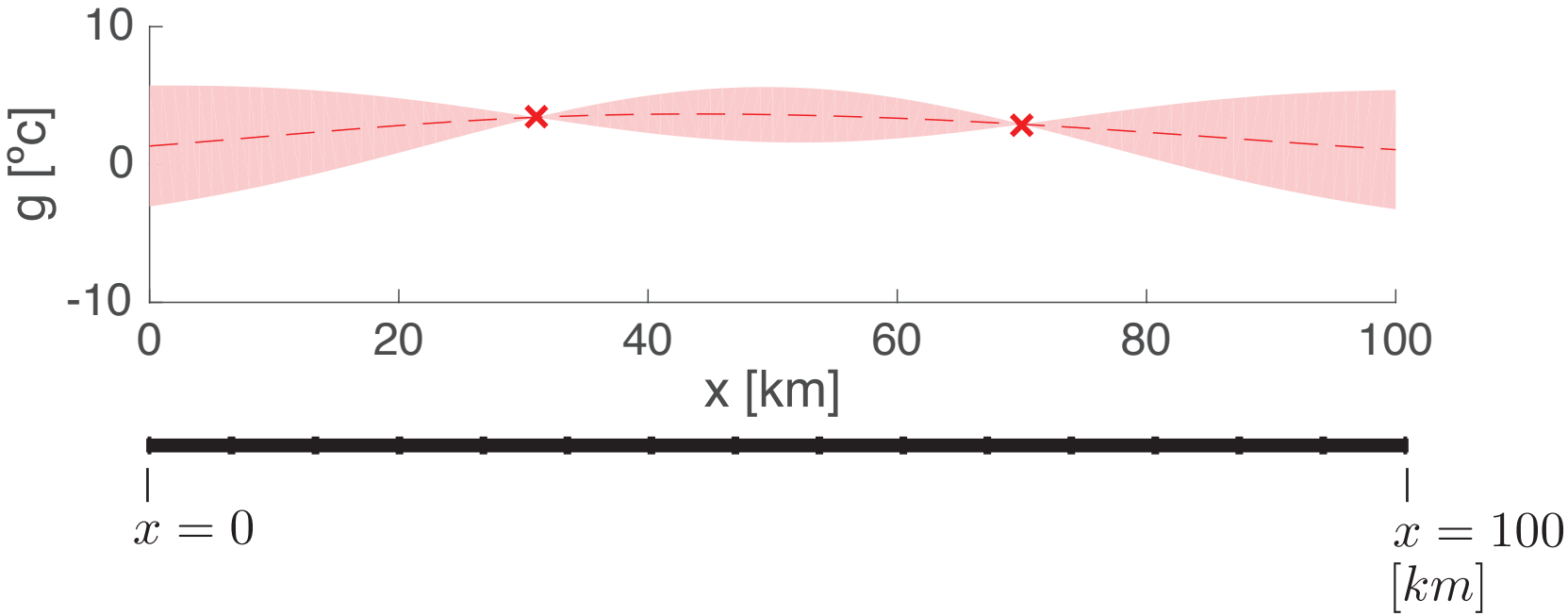
Example - T° distribution along a pipeline [🐍]



Observations: $\mathcal{D} = \{(\underbrace{[31]}_{\mathbf{x}}, \underbrace{[4.13]}_{\mathbf{g}})\}$

[CIV_ML/M14_GP_pipeline_obs.py]

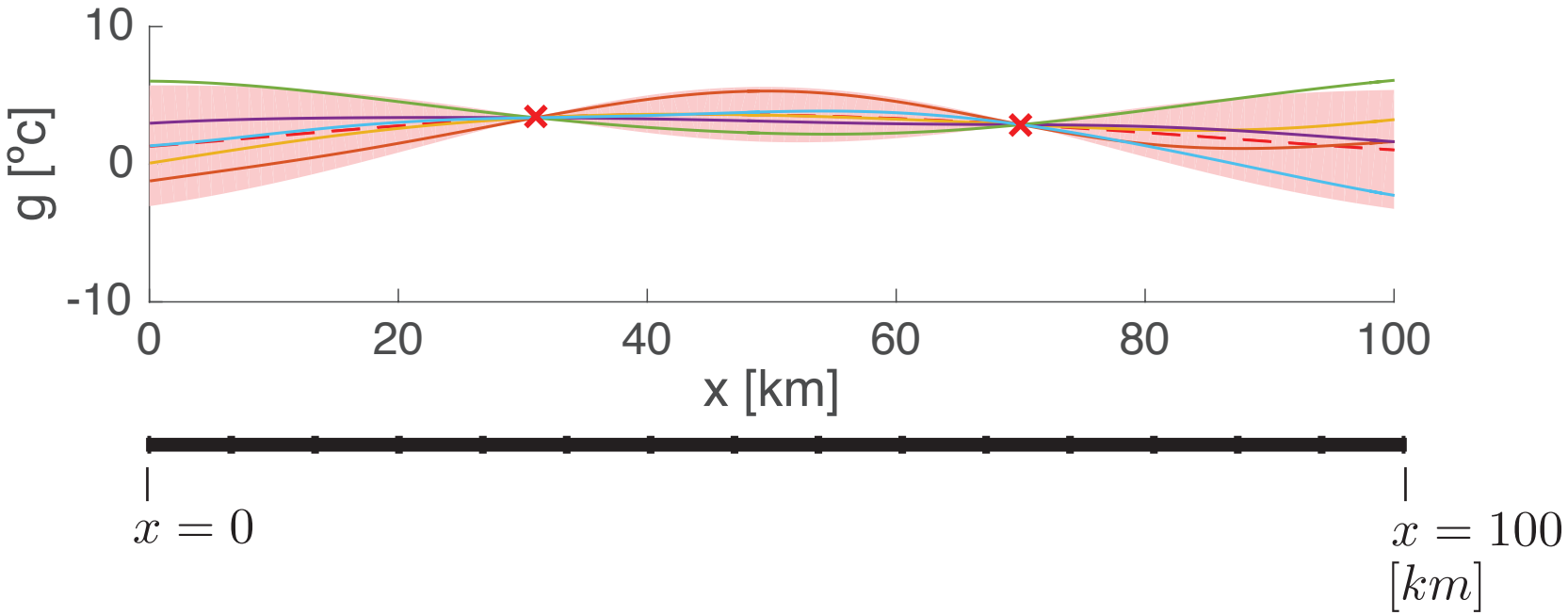
Example - T° distribution along a pipeline [🐍]



Observations: $\mathcal{D} = \{(\underbrace{[31, 70]}_{\mathbf{x}}, \underbrace{[4.13, 3.62]}_{\mathbf{g}})\}$

[CIV_ML/M14_GP_pipeline_obs.py]

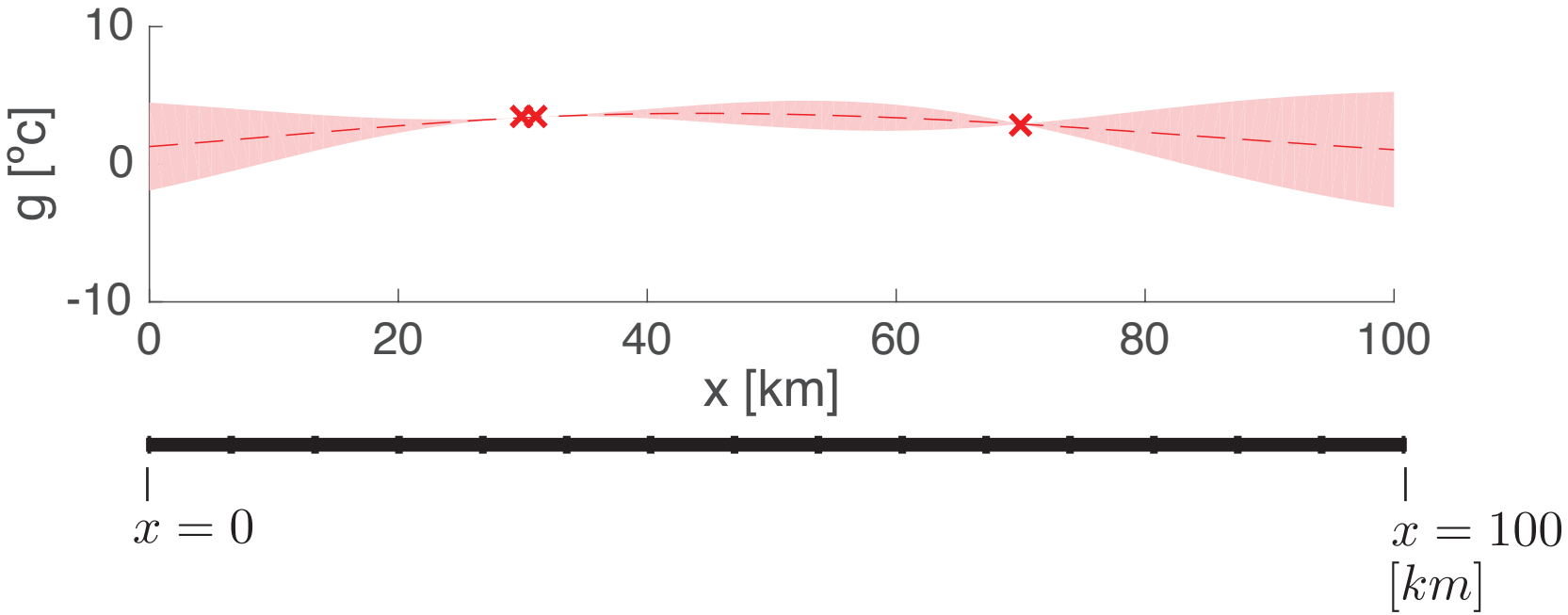
Example - T° distribution along a pipeline [🐍]



Observations: $\mathcal{D} = \{(\underbrace{[31, 70]}_{\mathbf{x}}, \underbrace{[4.13, 3.62]}_{\mathbf{g}})\}$

[CIV_ML/M14_GP_pipeline_obs.py]

Example - T° distribution along a pipeline [🐍]



Observations: $\mathcal{D} = \{(\underbrace{[31, 70, 30]}_{\mathbf{x}}, \underbrace{[4.13, 3.62, 3.71]}_{\mathbf{y}})\}$

[CIV_ML/M14_GP_pipeline_obs.py]

Noise-contaminated observations

Given a **system response** contaminated by **measurement noise** so that

$$\underbrace{Y_i}_{\text{observation}} = \overbrace{g(\mathbf{x}_i)}^{\text{reality}} + \underbrace{V}_{\text{measurement error}}, \quad V \sim \mathcal{N}(v; 0, \sigma_V^2)$$

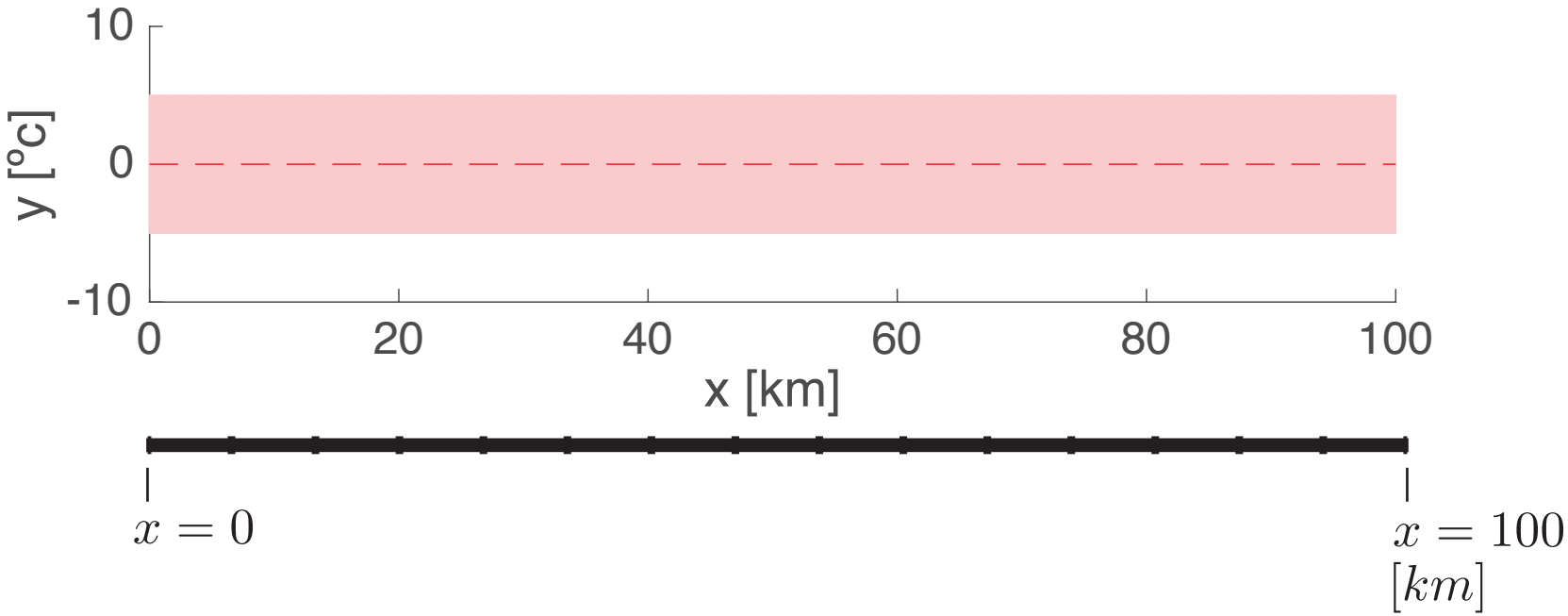
$$[\boldsymbol{\Sigma}_{\mathbf{Y}}]_{ij} = \rho(x_i, x_j)\sigma_G^2 + \sigma_V^2 \delta_{ij}, \quad \delta_{ij} = 1 \text{ if } i = j$$

$$\underbrace{\left\{ \begin{array}{c} \mathbf{Y} \\ \mathbf{G}_* \end{array} \right\}, \quad \boldsymbol{\mu} = \left\{ \begin{array}{c} \boldsymbol{\mu}_{\mathbf{Y}} \\ \hline \boldsymbol{\mu}_{\mathbf{G}_*} \end{array} \right\}, \quad \boldsymbol{\Sigma} = \left[\begin{array}{c|c} \boldsymbol{\Sigma}_{\mathbf{Y}} & \boldsymbol{\Sigma}_{\mathbf{Y}*} \\ \hline \boldsymbol{\Sigma}_{\mathbf{Y}*}^{\mathsf{T}} & \boldsymbol{\Sigma}_* \end{array} \right]}_{\text{prior knowledge}}$$

$$f_{\mathbf{G}_*|\mathbf{x}_*,\mathcal{D}}(\mathbf{g}_*|\mathbf{x}_*, \mathcal{D}) = \mathcal{N}(\mathbf{g}_*; \boldsymbol{\mu}_{*|\mathcal{D}}, \boldsymbol{\Sigma}_{*|\mathcal{D}})$$

$$\underbrace{\boldsymbol{\mu}_{*|\mathcal{D}} = \boldsymbol{\mu}_{\mathbf{G}_*} + \boldsymbol{\Sigma}_{\mathbf{Y}*}^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathbf{Y}}^{-1}(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{Y}}), \quad \boldsymbol{\Sigma}_{*|\mathcal{D}} = \boldsymbol{\Sigma}_* - \boldsymbol{\Sigma}_{\mathbf{Y}*}^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathbf{Y}}^{-1} \boldsymbol{\Sigma}_{\mathbf{Y}*}}_{\text{posterior knowledge}}$$

Example - T° distribution ($V \neq 0$) [🐍]

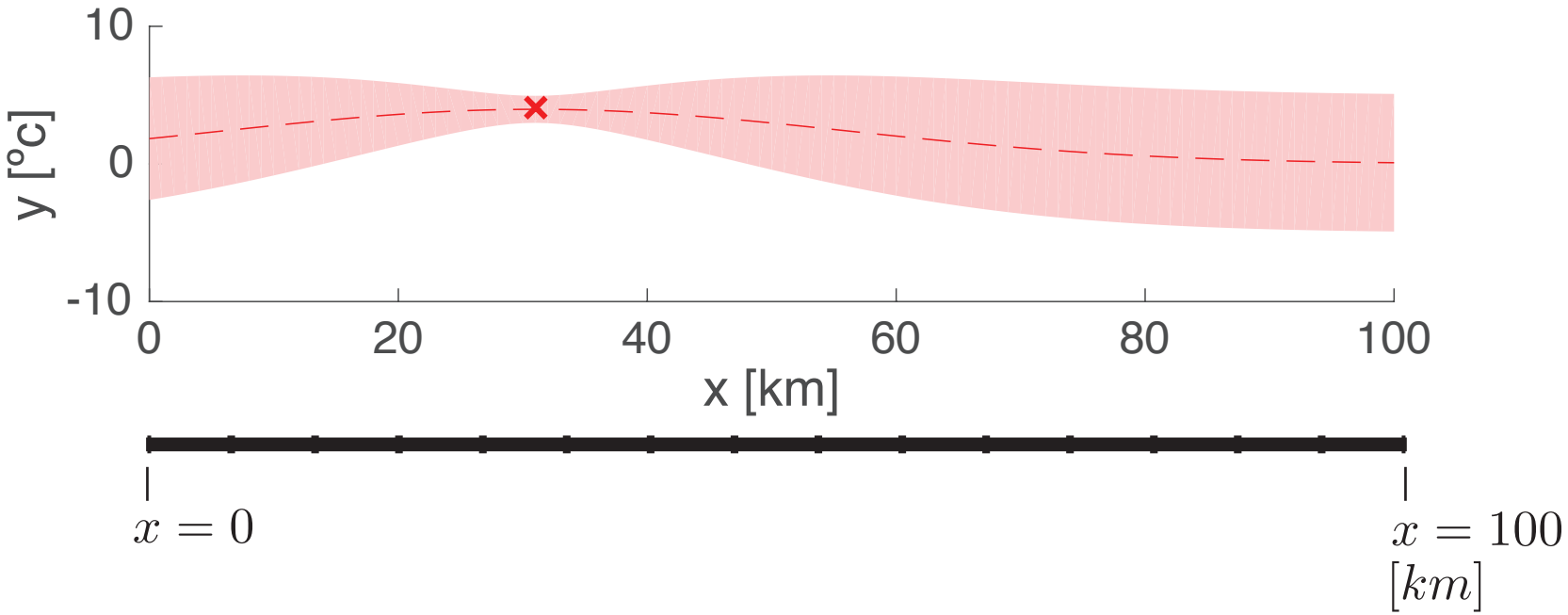


Measurement error: $\sigma_V = 0.5$

Observations: $\mathcal{D} = \{(\underbrace{[\,]}_{\mathbf{x}}, \underbrace{[\,]}_{\mathbf{y}})\}$

[CIV_ML/M14_GP_pipeline_obs.py]

Example - T° distribution ($V \neq 0$) [🐍]

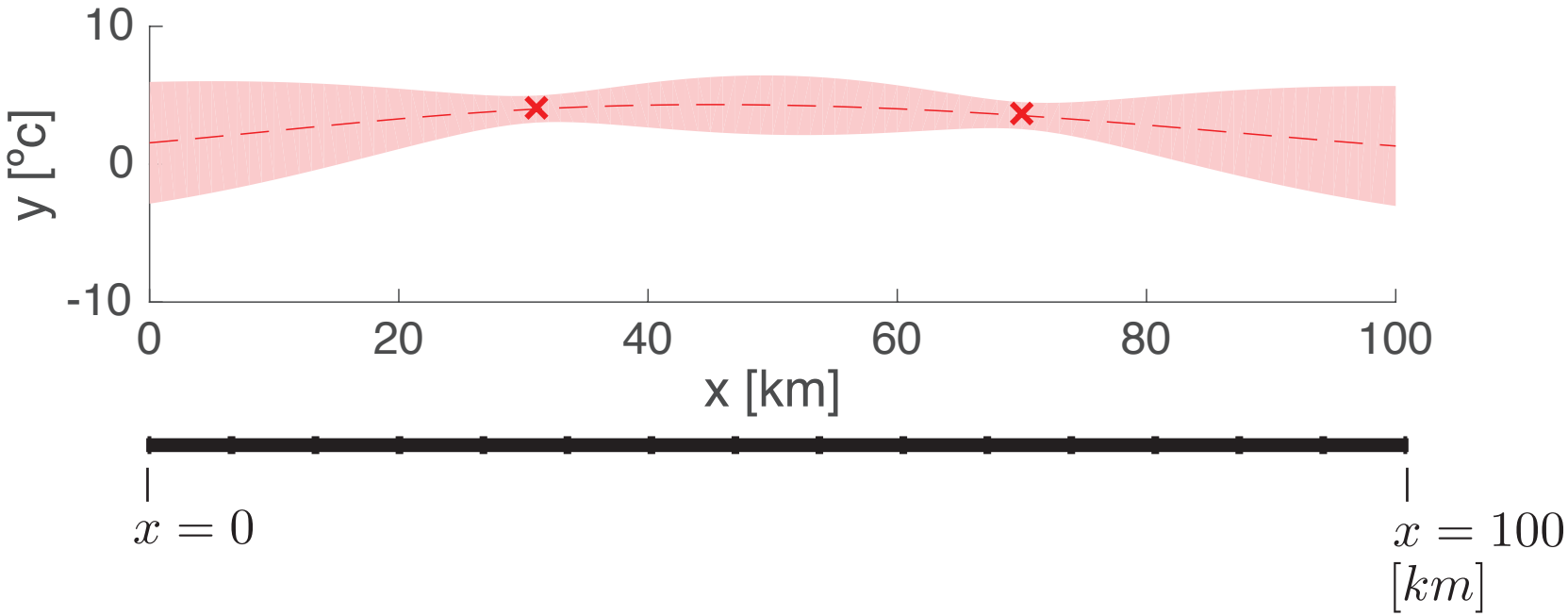


Measurement error: $\sigma_V = 0.5$

Observations: $\mathcal{D} = \{(\underbrace{[31]}_{\mathbf{x}}, \underbrace{[3.42]}_{\mathbf{y}})\}$

[CIV_ML/M14_GP_pipeline_obs.py]

Example - T° distribution ($V \neq 0$) [🐍]

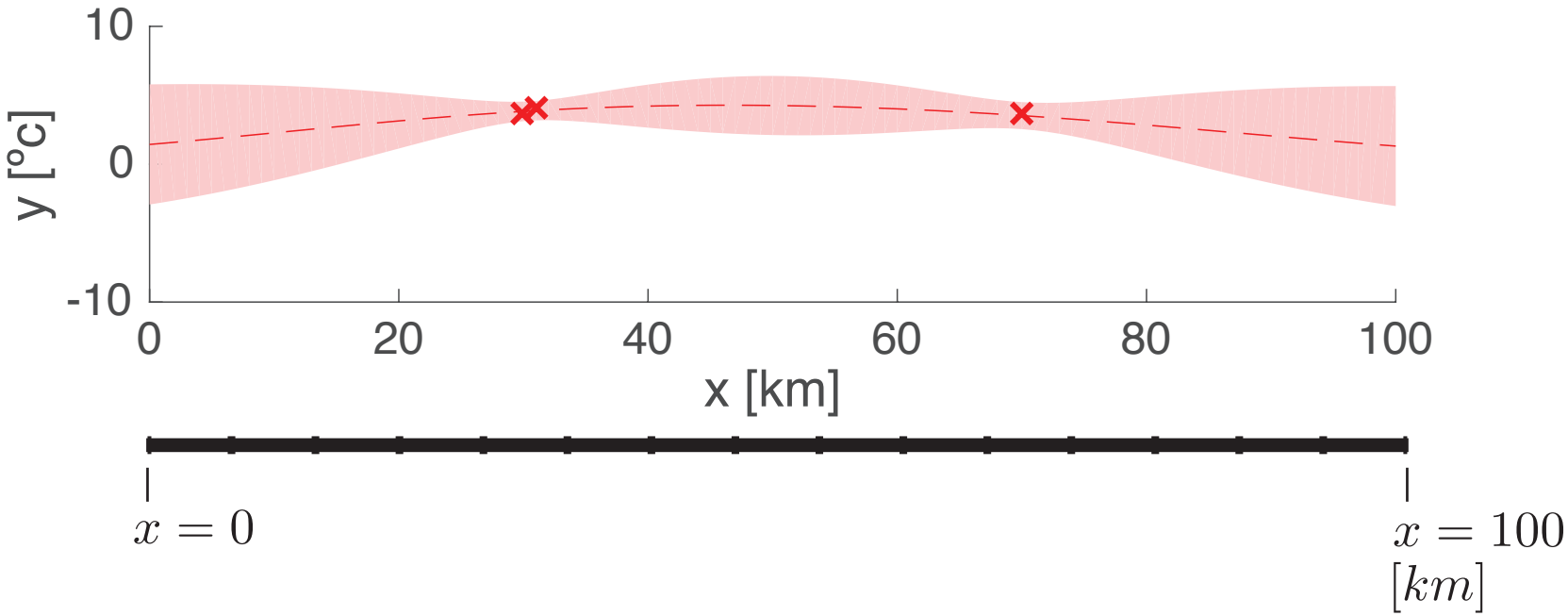


Measurement error: $\sigma_V = 0.5$

Observations: $\mathcal{D} = \{(\underbrace{[31, 70]}_{\mathbf{x}}, \underbrace{[3.42, 2.92]}_{\mathbf{y}})\}$

[CIV_ML/M14_GP_pipeline_obs.py]

Example - T° distribution ($V \neq 0$) [🐍]



Measurement error: $\sigma_V = 0.5$

Observations: $\mathcal{D} = \{([\underbrace{31, 70, 30}_{\mathbf{x}}], [\underbrace{3.42, 2.92, 3.38}_{\mathbf{y}}])\}$

[CIV_ML/M14_GP_pipeline_obs.py]

Multiple covariates: $\mathbf{x} = [x_1, x_2, \cdots, x_{\mathtt{X}}]_i^\intercal$

$$\underbrace{y_i}_{\text{observation}} = \overbrace{g(\mathbf{x}_i)}^{\text{reality}} + v_i, \quad \underbrace{\mathbf{x}_i = [x_1, x_2, \cdots, x_{\mathtt{X}}]_i^\intercal}_{\text{covariates}}, \quad \text{for } i = 1, \cdots, \mathtt{D}$$

$$\mathbf{y} = [y_1, y_2, \cdots, y_{\mathtt{D}}]^\intercal, \quad \mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{\mathtt{D}}]_{\mathtt{X}\times\mathtt{D}}, \quad \underbrace{\boldsymbol{\ell} = [\ell_1, \ell_2, \cdots, \ell_{\mathtt{X}}]_{\mathtt{X}\times 1}^\intercal}_{\text{one } \ell_k \text{ per covariate } \mathbf{x}_k}$$

$$\begin{aligned} \rho(\mathbf{x}_i, \mathbf{x}_j) &= \exp\left(-\frac{1}{2}\sum_{k=1}^{\mathtt{X}} \frac{([\mathbf{x}_i]_k - [\mathbf{x}_j]_k)^2}{\ell_k^2}\right) \\ &= \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\intercal \mathrm{diag}(\boldsymbol{\ell}^2)^{-1}(\mathbf{x}_i - \mathbf{x}_j)\right) \end{aligned}$$

Multivariate correlation function []

$$\rho(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2}\sum_{k=1}^{X}\frac{([\mathbf{x}_i]_k - [\mathbf{x}_j]_k)^2}{\ell_k^2}\right)$$

$$= \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^T \text{diag}(\boldsymbol{\ell})^{-2}(\mathbf{x}_i - \mathbf{x}_j)\right)$$

Raw data

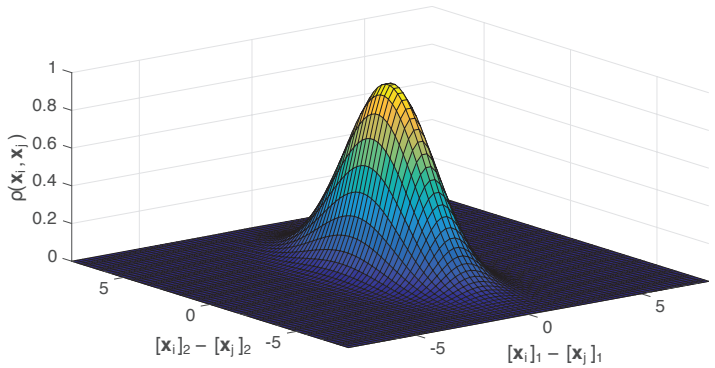
si $\ell_k \gg \sigma_{X_k}$: $\text{Imp}(X_k \to y) \approx 0$

si $\ell_k < \sigma_{X_k}$: $\text{Imp}(X_k \to y) \gg 0$

Standardized data $(x_k - \mu_{X_k})/\sigma_{X_k}$

si $\ell_k \gg 1$: $\text{Imp}(X_k \to y) \approx 0$

si $\ell_k < 1$: $\text{Imp}(X_k \to y) \gg 0$



[CIV_ML/M14_multivariate_corr_fct.py]

Parameter estimation σ_G & ℓ

Up to now, we made the hypothesis that we knew σ_G and $\boldsymbol{\ell} = [\ell_1, \ell_2, \cdots, \ell_{\mathtt{X}}]^\intercal$.

In practice, we need to learn hyperparameters $\boldsymbol{\theta} = [\sigma_G, \boldsymbol{\ell}]^\intercal$ using training data $\mathcal{D} = \{\mathcal{D}_x, \mathcal{D}_y\} = \{(\mathbf{x}_i, y_i),\ i = 1, \cdots \mathtt{D}\}$.

Bayes rule:

$$\underbrace{f(\boldsymbol{\theta}|\mathcal{D}_x, \mathcal{D}_y)}_{\text{posterior}} = \frac{\overbrace{f(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta})}^{\text{likelihood}} \cdot \overbrace{f(\boldsymbol{\theta})}^{\text{prior}}}{\underbrace{f(\mathcal{D}_y)}_{\text{cte.}}}$$

⚠ **In practice, evaluating $f(\boldsymbol{\theta}|\mathcal{D}_x, \mathcal{D}_y)$ is computationally expensive...**

MLE approximation

$$\underbrace{f(\boldsymbol{\theta}|\overbrace{\mathcal{D}_x, \mathcal{D}_y}^{\mathcal{D}})}_{\text{posterior}} = \frac{\overbrace{f(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta})}^{\text{likelihood}} \cdot \overbrace{f(\boldsymbol{\theta})}^{\text{prior}}}{\underbrace{f(\mathcal{D}_y)}_{\text{cte.}}}$$

$$\text{If } f(\boldsymbol{\theta}) = \text{cte.}, \ \forall \boldsymbol{\theta} \rightarrow f(\boldsymbol{\theta}|\mathcal{D}_x, \mathcal{D}_y) \propto f(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta})$$

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} f(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta})$$

$$f(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta}) = \mathcal{N}(\mathcal{D}_y; \boldsymbol{\mu}_{\mathbf{Y}}, \boldsymbol{\Sigma}_{\mathbf{Y}})$$

MLE approximation (cont.)

When the size of $\boldsymbol{\Sigma}_{\mathbf{Y}}$ increases, $f(\mathcal{D}_y|\mathcal{D}_x,\boldsymbol{\theta}) \rightarrow 0$ causing **zero underflow** ⚠.

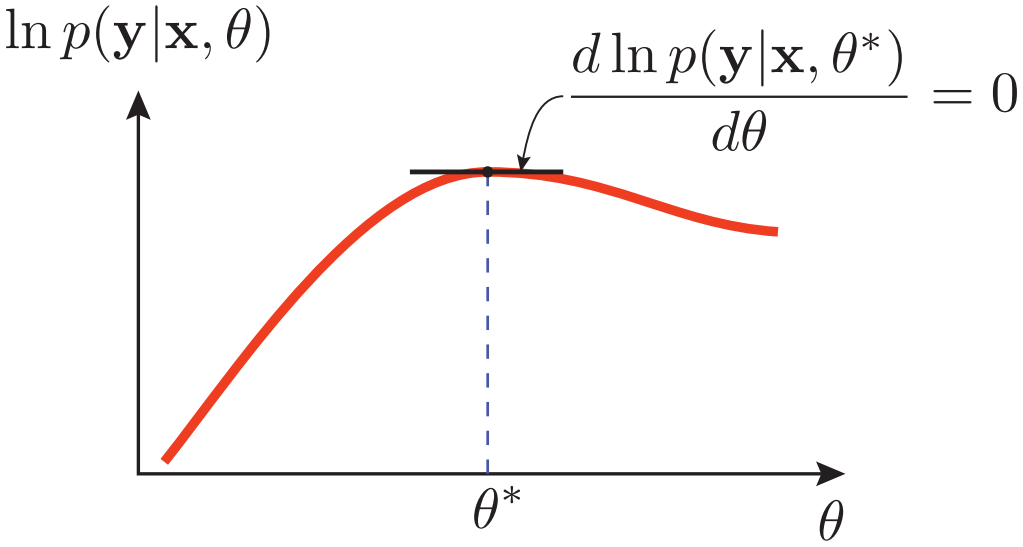
Solution:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \overbrace{f(\mathcal{D}_y|\mathcal{D}_x,\boldsymbol{\theta})}^{\text{likelihood}} \equiv \arg\max_{\boldsymbol{\theta}} \overbrace{\log f(\mathcal{D}_y|\mathcal{D}_x,\boldsymbol{\theta})}^{\text{log-likelihood}}$$

$f(\mathcal{D}_y|\mathcal{D}_x,\boldsymbol{\theta})$'s maximum corresponds to $\log f(\mathcal{D}_y|\mathcal{D}_x,\boldsymbol{\theta})$'s maximum and there are no more zero underflow ✓.

$$\begin{aligned}\log f(\mathcal{D}_y|\mathcal{D}_x,\boldsymbol{\theta}) &= \log \mathcal{N}(\mathcal{D}_y;\boldsymbol{\mu}_{\mathbf{Y}},\boldsymbol{\Sigma}_{\mathbf{Y}}) \\ &= -\tfrac{1}{2}(\mathcal{D}_y-\boldsymbol{\mu}_{\mathbf{Y}})^{\intercal}\boldsymbol{\Sigma}_{\mathbf{Y}}^{-1}(\mathcal{D}_y-\boldsymbol{\mu}_{\mathbf{Y}}) - \tfrac{1}{2}\log|\boldsymbol{\Sigma}_{\mathbf{Y}}| - \tfrac{\mathtt{D}}{2}\log 2\pi\end{aligned}$$

Maximization

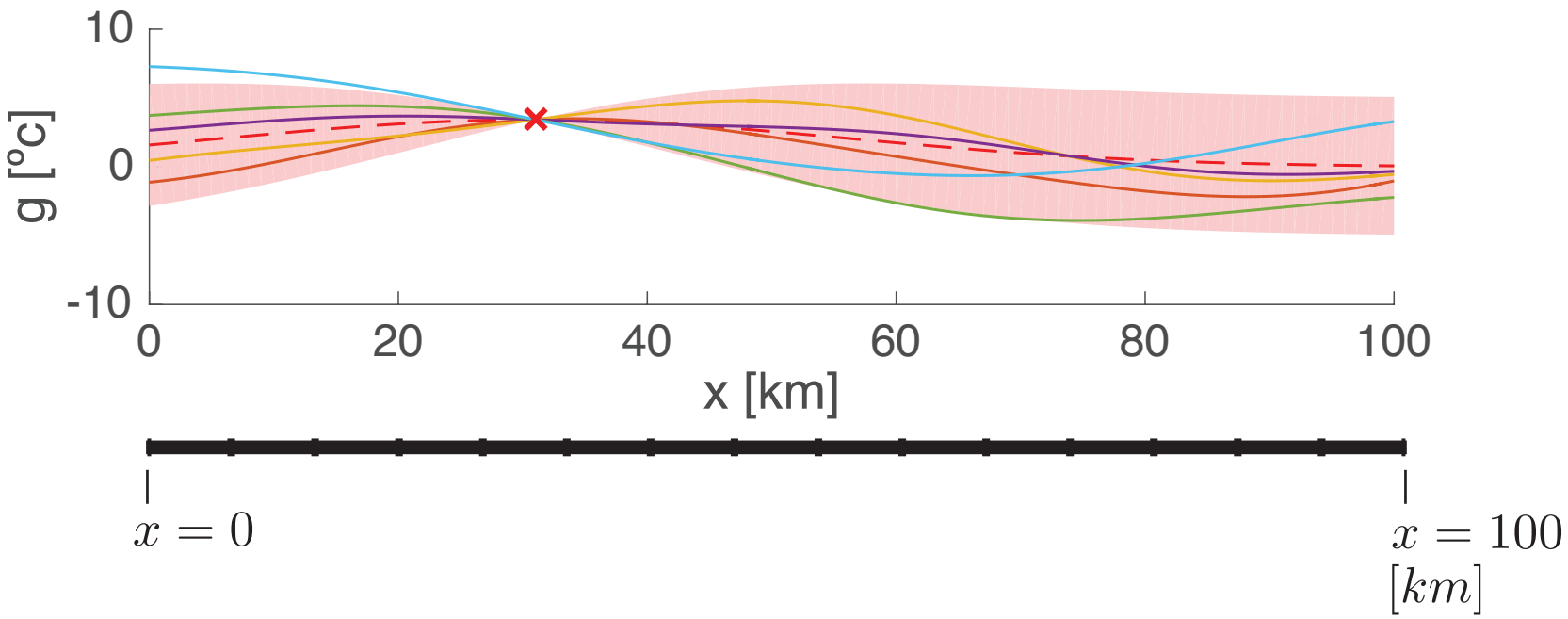


How to find θ^*?

$$\frac{\partial}{\partial \theta_j} \log f(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^\intercal \boldsymbol{\Sigma}_\mathbf{Y}^{-1} \frac{\partial \boldsymbol{\Sigma}_\mathbf{Y}}{\partial \theta_j} \boldsymbol{\Sigma}_\mathbf{Y}^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_\mathbf{Y}^{-1}\frac{\partial \boldsymbol{\Sigma}_\mathbf{Y}}{\partial \theta_j}) = 0$$

We employ iterative optimization techniques to find θ^*
(such as **Newton-Raphson**)
Efficient algorithms are already implemented in **GPy/GPML**.

Interactive example - T° distribution along a pipeline [.m]



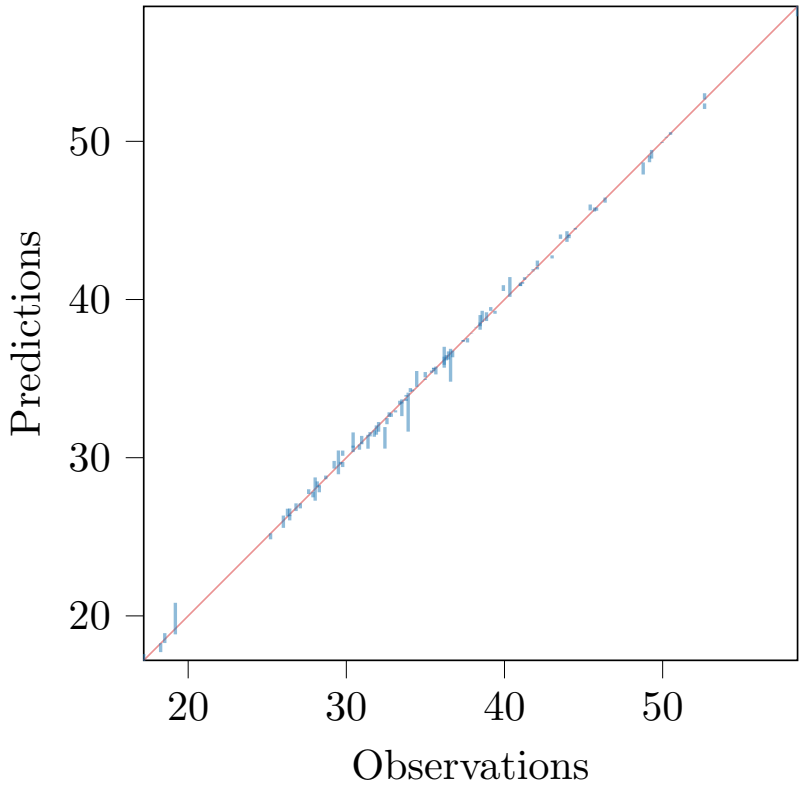
[CIV_ML/GPR_example.m]

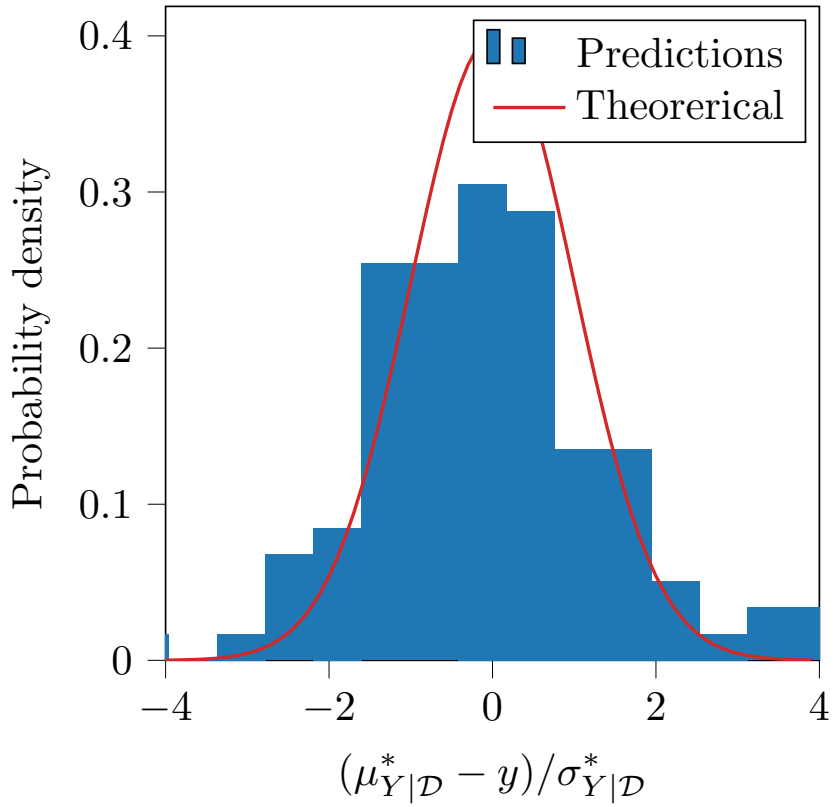
Evaluating GPR's predictive capacity

We should employ **N-fold cross-validation** to estimate the GPR predictive capacity

1. set-up the GPR model
2. use CV to estimate hyper-parameters $\{\ell, \sigma_G, \sigma_V\}$ the current training set and then predict only data points that are not used in the training set
3. repeat 2. for each for the N cross-validation fold
4. plot the histogram of normalized prediction errors: $\frac{\boldsymbol{\mu}_{\mathbf{Y}|\mathcal{D},i} - y_i}{\sigma_{\mathbf{Y}|\mathcal{D},i}}$
5. plot a scatter of predicted and observed values

Example – GPR predictive capacity (20-folds CV)





GPy library - Option #1 Raw data []

```python
import GPy
import numpy as np

# Data
x_obs, y_obs, x_pred #Covariates, observed values, pred. covariates
nb_dim = x_obs.shape[1] #Number of covariates

# GPy
l, var_G, var_V = 10, 2, 0.1 #Lengthscale, Process/observation err. variance
kernel = GPy.kern.RBF(input_dim = nb_dim, variance = var_G, lengthscale = l)
lik = GPy.likelihoods.Gaussian()
model = GPy.models.GPRegression(x_obs, y_obs, kernel = kernel ,ARD = True)
model.Gaussian_noise.variance = var_V
model.optimize()
print(model)

# Estimated parameter values
print(kernel.lengthscale)
print(kernel.variance)
print(model.log_likelihood())

# Predictions
m_G, var_G = model.predict(x_pred, full_cov = False) #Diagonal covariance

# Ploting
fig = model.plot()
GPy.plotting.show(fig)
```

[CIV_ML/GPR_Example_Concrete.py]

GPy library - Option #2 Standardized data []

```python
import GPy
import numpy as np

# Data
x_obs, y_obs, x_pred #Covariates, observed values, pred. covariates
nb_dim = x_obs.shape[1] #Number of covariates

# Standardization
mx = np.mean([x_obs],axis=1)
sx = np.std([x_obs],axis=1)
my = np.mean([y_obs])
sy = np.std([y_obs])
x_std = (x_std-mx)/sx
y_std = (y_obs-my)/sy
x_pred_std = (x_pred-mx)/sx

# GPy
kernel = GPy.kern.RBF(input_dim = nb_dim, ARD = True)
model = GPy.models.GPRegression(x_std, y_std, kernel = kernel)
model.optimize()

# Standardized Predictions
m_G_std, var_G_std = model.predict(x_pred_std, full_cov = False) #Diag. cov.

# Unstandardize predictions
m_G = m_G_std*sy+my
var_G = var_G_std*sy**2
```

[CIV_ML/GPR_Example_Concrete.py]

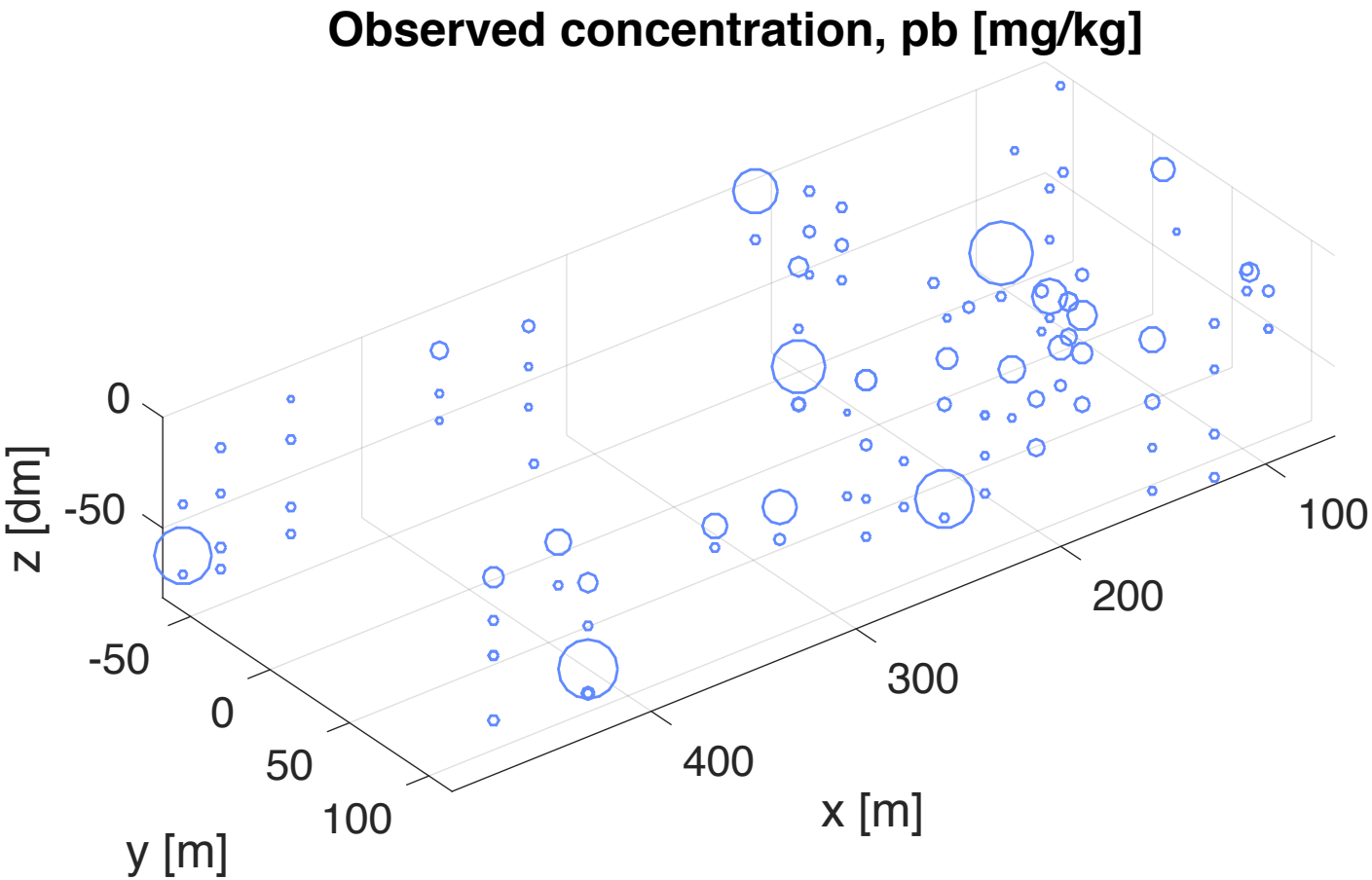
Section Outline

Examples

4.1 Soil contamination characterization

4.2 UCI - Concrete Slump

Soil contamination characterization [.fig]



[CIV_ML/Soil_contamination_data.fig]

Problem set-up

Covariates and observations:

$$\underbrace{\mathbf{l}_i^{\{Y\}} = [x, y, z]_i^{\mathsf{T}}}_{\text{geodesic coordinates}}$$

$$\mathcal{D} = \{\mathcal{D}_x, \mathcal{D}_y\} = \{(\mathbf{l}_i^{\{Y\}}, \mathbf{y}_i),\ i = 1:116\}$$

Observation model:

$$\log \underbrace{y_i}_{\text{observation}} = \overbrace{g(\mathbf{l}_i^{\{Y\}})}^{\text{true contaminant [] in log space}} + \underbrace{v_i}_{\text{meas. error in log space}},\ v : V \sim \mathcal{N}(0, \sigma_V^2)$$

Problem set-up (cont.)

Prior:

$$\boldsymbol{\mu} = \left\{ \begin{array}{c} \boldsymbol{\mu}_{\mathbf{Y}} \\ \boldsymbol{\mu}_{*} \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{0} \\ \mathbf{0} \end{array} \right\}, \; \boldsymbol{\Sigma} = \left[\begin{array}{cc} \boldsymbol{\Sigma}_{\mathbf{Y}} & \boldsymbol{\Sigma}_{\mathbf{Y}*} \\ \boldsymbol{\Sigma}_{\mathbf{Y}*}^{\mathsf{T}} & \boldsymbol{\Sigma}_{*} \end{array} \right]$$

$$\begin{aligned} [\boldsymbol{\Sigma}_{\mathbf{Y}}]_{ij} &= \rho(\mathbf{l}_i^{\{Y\}}, \mathbf{l}_j^{\{Y\}})\sigma_G^2 + \sigma_V{}^2\delta_{ij}, \; \delta_{ij} = 1 \text{ if } i = j, \text{ else } \delta_{ij} = 0 \\ [\boldsymbol{\Sigma}_{*}]_{kl} &= \rho(\mathbf{l}_k^{\{G\}}, \mathbf{l}_l^{\{G\}})\sigma_G^2 \\ [\boldsymbol{\Sigma}_{\mathbf{Y}*}]_{ik} &= \rho(\mathbf{l}_i^{\{Y\}}, \mathbf{l}_k^{\{G\}})\sigma_G^2 \end{aligned}$$

$$\rho(\mathbf{l}_i, \mathbf{l}_j) = \exp\left(-\frac{1}{2}(\mathbf{l}_i - \mathbf{l}_j)^{\mathsf{T}}\text{diag}(\boldsymbol{\ell}^2)^{-1}(\mathbf{l}_i - \mathbf{l}_j)\right)$$

Posterior knowledge:

$$\boldsymbol{\mu}_{*|\mathcal{D}} = \boldsymbol{\mu}_{*} + \boldsymbol{\Sigma}_{\mathbf{Y}*}^{\mathsf{T}}\boldsymbol{\Sigma}_{\mathbf{Y}}^{-1}(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{Y}})$$

$$\boldsymbol{\Sigma}_{*|\mathcal{D}} = \boldsymbol{\Sigma}_{*} - \boldsymbol{\Sigma}_{\mathbf{Y}*}^{\mathsf{T}}\boldsymbol{\Sigma}_{\mathbf{Y}}^{-1}\boldsymbol{\Sigma}_{\mathbf{Y}*}$$

Problem set-up (cont.)

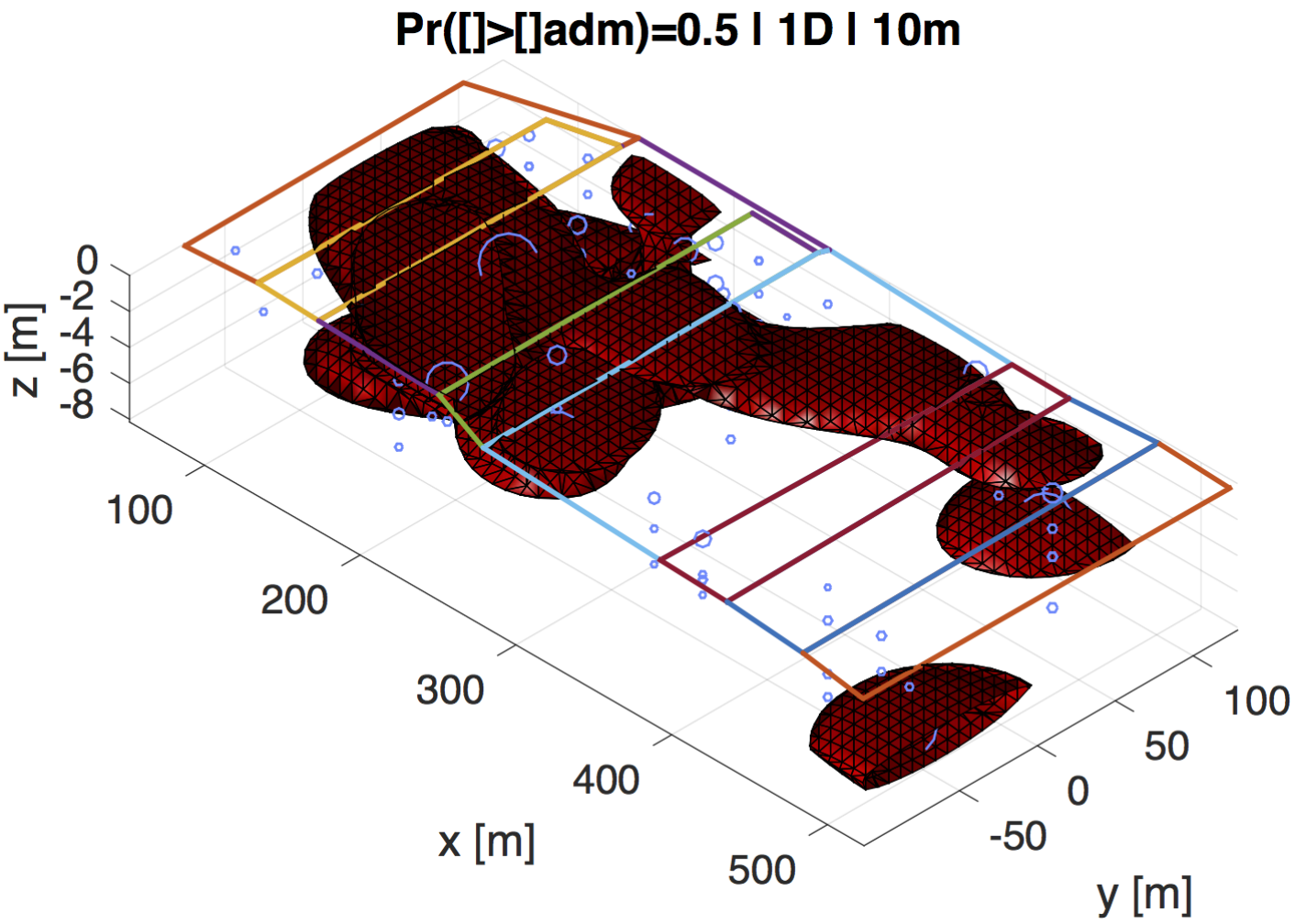
Hyper-parameters estimation:

$$\boldsymbol{\theta} = \{\ell_x, \ell_y, \ell_z, \sigma_G, \sigma_V\}$$

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \overbrace{\log p(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta})}^{\text{log-likelihood}}$$

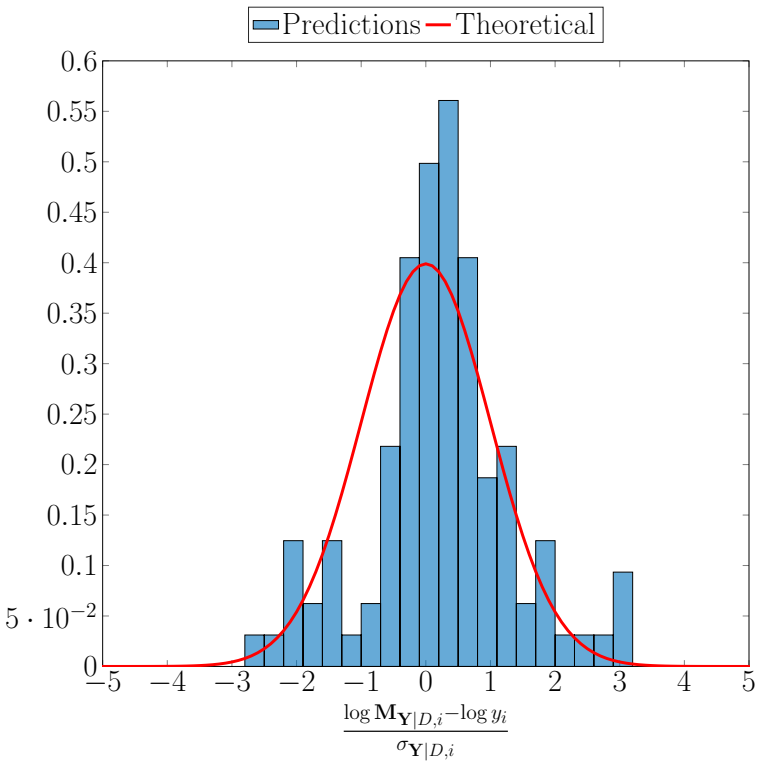
$$= \left\{ \begin{array}{c} \ell_x = 56.4\,m \\ \ell_y = 53.7\,m \\ \ell_z = 0.64\,m \\ \sigma_G = 2.86 \\ \sigma_V = 0.18 \end{array} \right\}$$

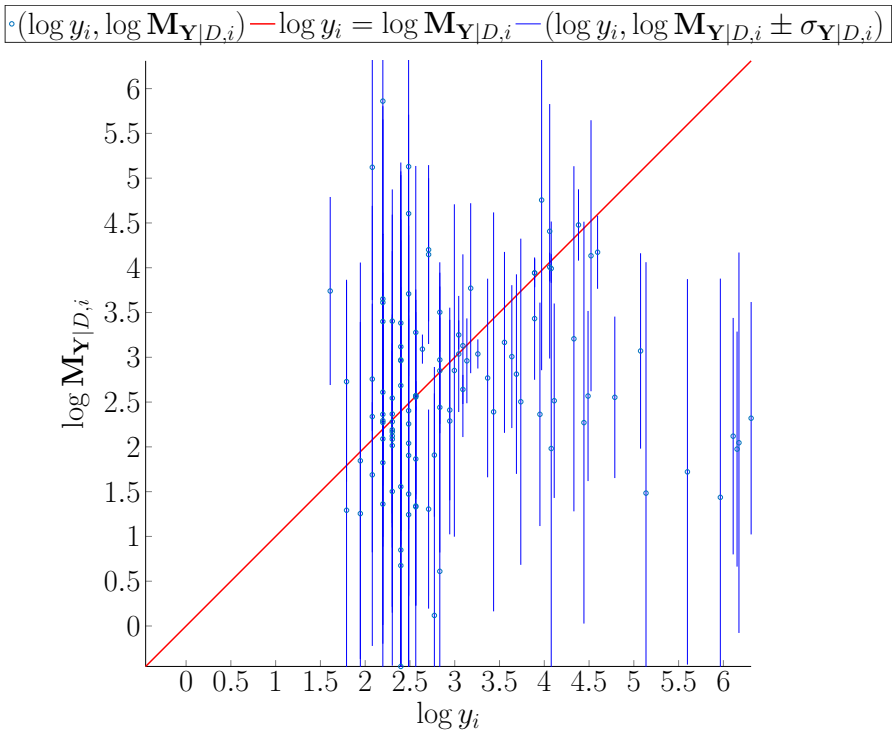
Results – contaminant concentration [.fig] [.m]



[CIV_ML/Soil_contamination_concentration.fig//Soil_contaimnation/GPR_SC_main.m]

Results – Prediction quality





⚠ Overall, the predictive capacity is poor → need more data

UCI - Concrete Slump dataset

UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

About Citation Policy Donate a Data Set Contact
Search
Repository Web
View ALL Data Sets

Concrete Slump Test Data Set

Download: Data Folder, Data Set Description

Abstract: Concrete is a highly complex material. The slump flow of concrete is not only determined by the water content, but that is also influenced by other concrete ingredients.

Data Set Characteristics:	Multivariate	**Number of Instances:**	103	**Area:**	Computer
Attribute Characteristics:	Real	**Number of Attributes:**	10	**Date Donated**	2009-04-30
Associated Tasks:	Regression	**Missing Values?**	N/A	**Number of Web Hits:**	132011

Source:

Donor: I-Cheng Yeh
Email: icyeh '@' chu.edu.tw
Institution: Department of Information Management, Chung-Hua University (Republic of China)
Other contact information: Department of Information Management, Chung-Hua University, Hsin Chu, Taiwan 30067, R.O.C.

`https://archive.ics.uci.edu/ml/datasets/Concrete+Slump+Test`

1 output [MPa]

- 28-day compressive strength

7 covariates [kg/m^3]

- Cement
- Slag
- Fly ash
- Water
- Superplasticizer
- Coarse aggregate
- Fine aggregate

Problem set-up

Covariates and observations:

$$\mathbf{x}_i = [x_1, x_2, \cdots, x_7]_i^\mathsf{T}$$

$$\tilde{\mathcal{D}} = \{\tilde{\mathcal{D}}_x, \tilde{\mathcal{D}}_y\} = \{(\frac{\mathbf{x}_i - \boldsymbol{\mu_X}}{\boldsymbol{\sigma_X}}, \frac{\mathbf{y}_i - \boldsymbol{\mu_Y}}{\boldsymbol{\sigma_Y}}),\ i = 1:103\}$$

Observation model:

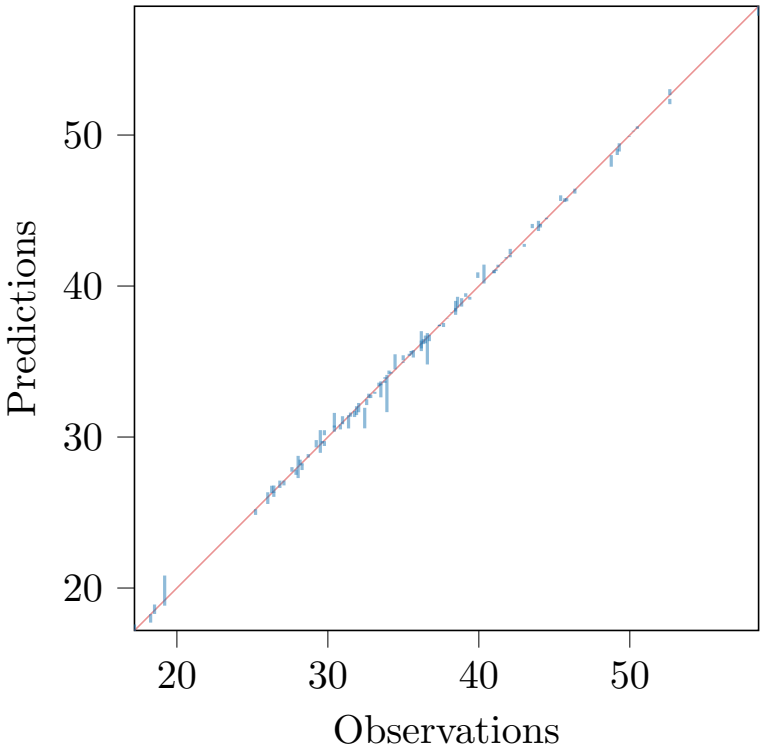
$$\underbrace{\tilde{y}_i}_{\text{observation}} = \overbrace{g(\tilde{\mathbf{x}}_i)}^{\text{true strength}} + \underbrace{v_i}_{\text{meas. error}},\ v : V \sim \mathcal{N}(0, \sigma_V^2)$$

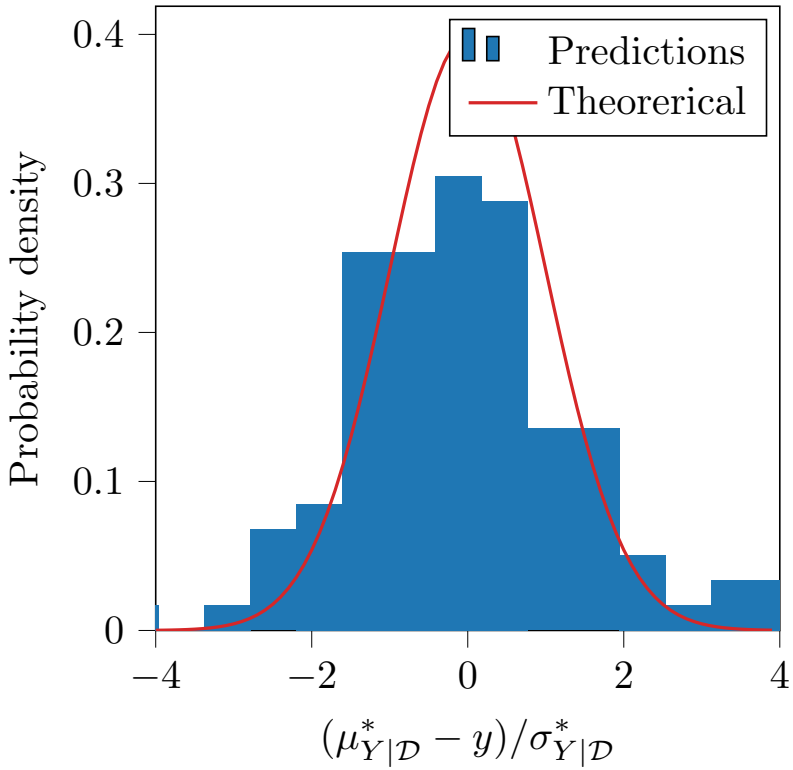
Hyper-parameters estimation

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \overbrace{\log p(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta})}^{\text{log-likelihood}}$$

$$= \left\{ \begin{array}{rcll} \ell_1 &=& 5.4 & \\ \ell_2 &=& 7.2 & \\ \ell_3 &=& 4.8 & \\ \ell_4 &=& 2.4 & \\ \ell_5 &=& 18.5 & \text{⚠} \\ \ell_6 &=& 28644 & \text{⚠} \\ \ell_7 &=& 3.9 & \\ \sigma_G &=& 2.7 & \\ \sigma_V &\approx& 0 & \end{array} \right\}$$

Results 20-folds CV – $\boldsymbol{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]^\intercal$ []

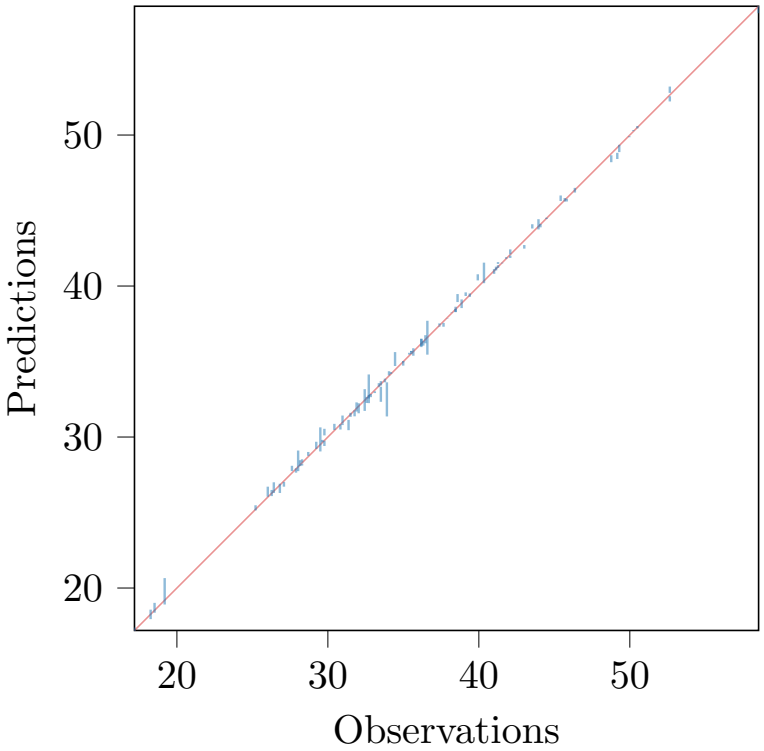


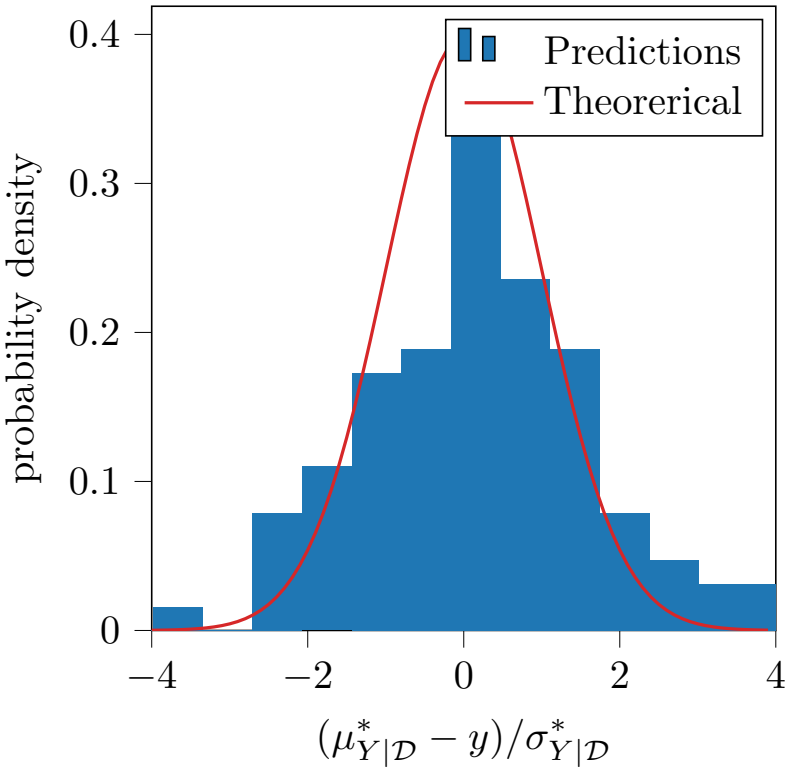


Mean square error (MSE)	0.118
Mean absolute error (MAE)	0.212
Mean error (ME)	0.016
Std error (RMSE)	0.343

[CIV_ML/GPR_example_Concrete.py]

Results 20-folds CV – $\boldsymbol{x} = [x_1, x_2, x_3, x_4, x_5, \ , x_7]^\intercal$ []





Mean square error (MSE)	0.079 $<$ 0.118
Mean absolute error (MAE)	0.181
Mean error (ME)	0.029
Std error (RMSE)	0.279

[CIV_ML/GPR_example_Concrete.py]

Section Outline

Adv. topics

5.1 Prior knowledge – mean values

5.2 Covariance functions

5.3 Homo/Heteroscedasticity

5.4 Other methods

Prior knowledge – mean values

So far, we have modelled our prior mean as either **0** or a **cte.**

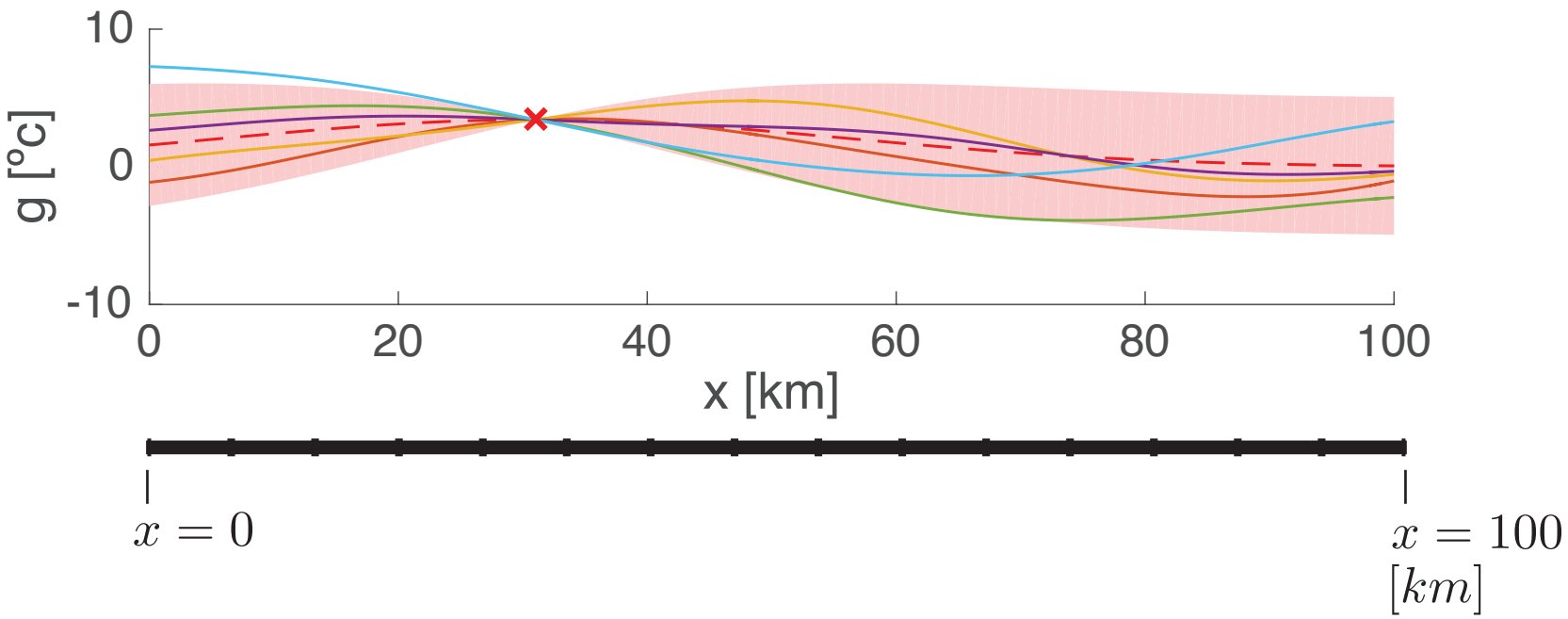
When predicting in the neighbourhood of observed data the prior knowledge has a negligible impact. **However, the impact of the prior increase as the distance from observed data increases**

Solution: **your prior knowledge can be parameterized by a function depending on covariates**.

$$\text{e.g.: } m(x) = ax + b$$

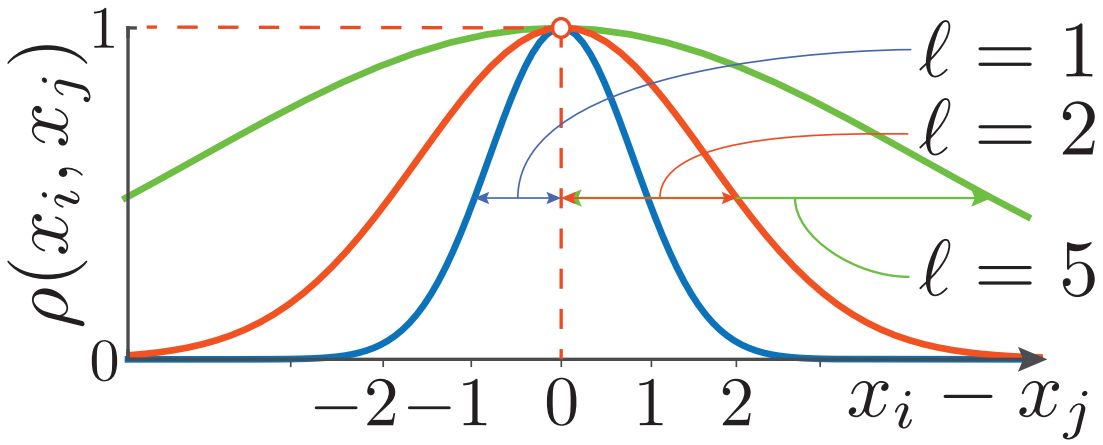
Where $\{a, b\}$ are **hyper-parameters** that needs to be estimated using data.

Interactive example - T° distribution along a pipeline [.m]



[CIV_ML/GPR_example.m]

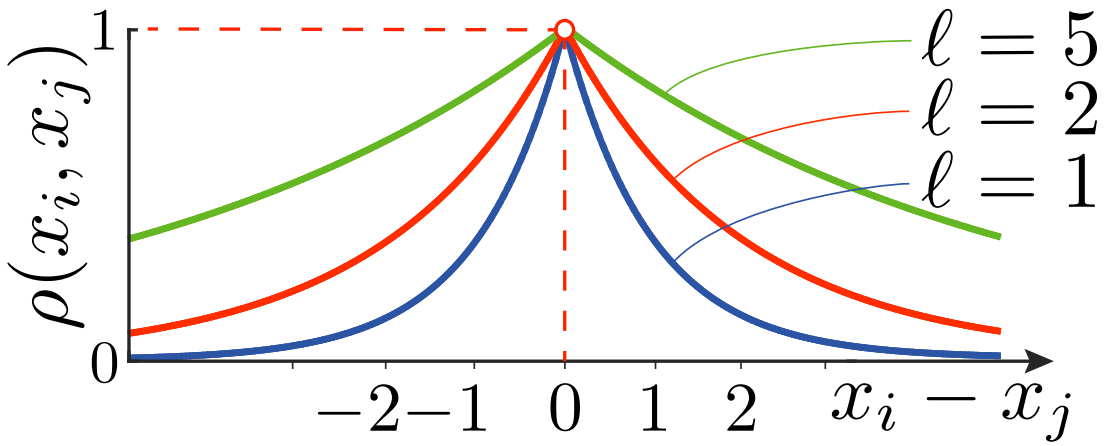
Square-exponential covariance function



$$\rho(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\mathsf{T}\mathrm{diag}(\boldsymbol{\ell})^{-2}(\mathbf{x}_i - \mathbf{x}_j)\right)$$

$$= \exp\left(-\sum_{k=1}^{\mathtt{X}} \frac{([x_i]_k - [x_j]_k)^2}{[\boldsymbol{\ell}]_k}\right)$$

Matern covariance function



$$\rho(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\sum_{k=1}^{\mathtt{X}} \frac{|[x_i]_k - [x_j]_k|}{[\ell]_k}\right)$$

Covariance function and cross-validation

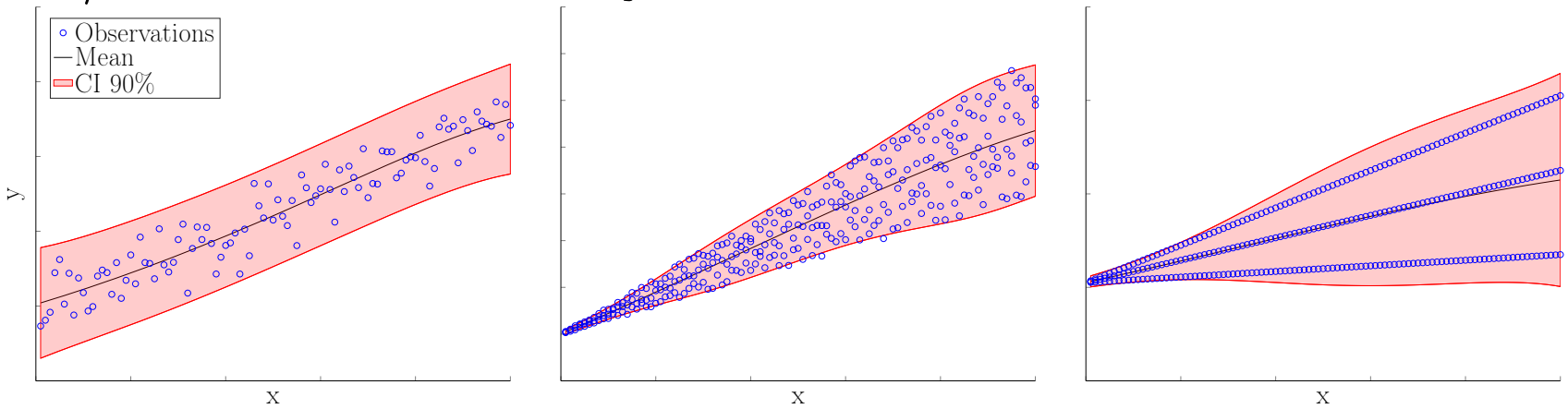
How to choose between two **covariance function structures?**

Cross-Validation (easy way)

or

Bayesian Model selection (hard way)

Homo/Heteroscedasticity



(a) Homoscedastic and independent

(b) Heteroscedastic and independent

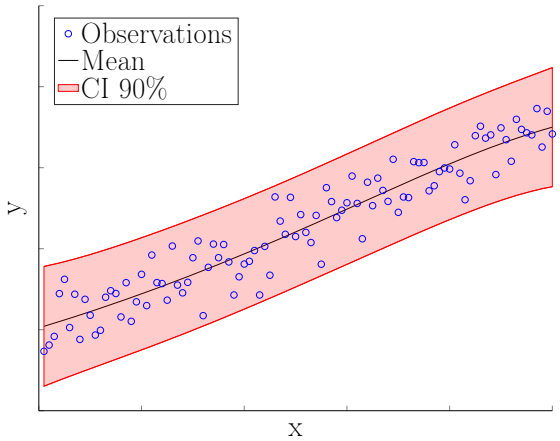
(c) Heteroscedastic and dependent

a) Standard GPR ,**GPy/GPML**

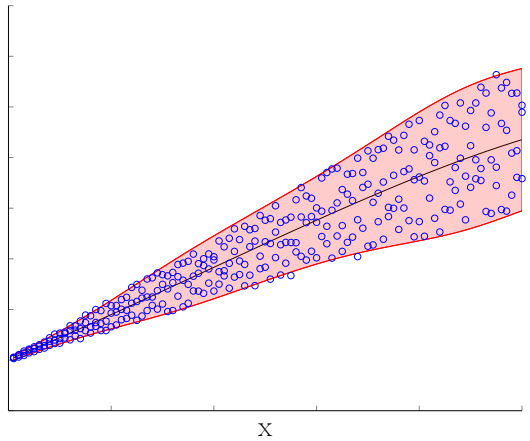
b) Tolvanen et al. (2014). *Expectation propagation for non-stationary heteroscedastic gaussian process regression*. IEEE-MLSP, **GPy/GPStuff**

c) Tabor et al. (2017). *Probabilistic modeling of heteroscedastic laboratory experiments using gaussian process regression*. Journal of Engineering Mechanics

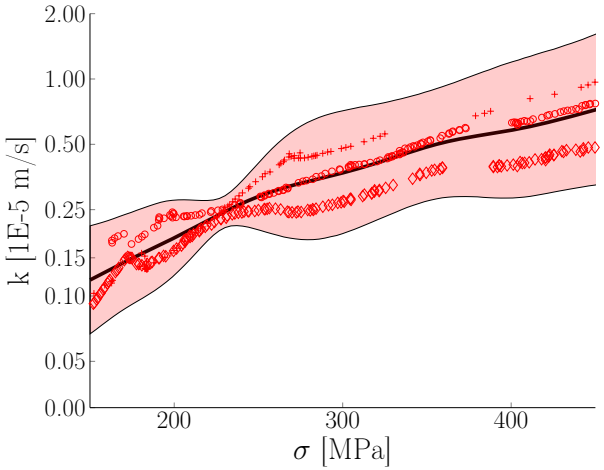
Homo/Heteroscedasticity



(a) Homoscedastic and independent



(b) Heteroscedastic and independent



(c) Heteroscedastic and dependent

a) Standard GPR ,**GPy/GPML**

b) Tolvanen et al. (2014). *Expectation propagation for non-stationary heteroscedastic gaussian process regression*. IEEE-MLSP, **GPy/GPStuff**

c) Tabor et al. (2017). *Probabilistic modeling of heteroscedastic laboratory experiments using gaussian process regression*. Journal of Engineering Mechanics

Regression methods

Linear regression and Gaussian Process regression are not the only approaches available

- Neural-networks (Deep-Learning)
- Support vector macine (SVM)
- Decision tree (Random-forest)
- ...+++

Most of current research carries on Neural Networks

Section Outline

Neural Networks

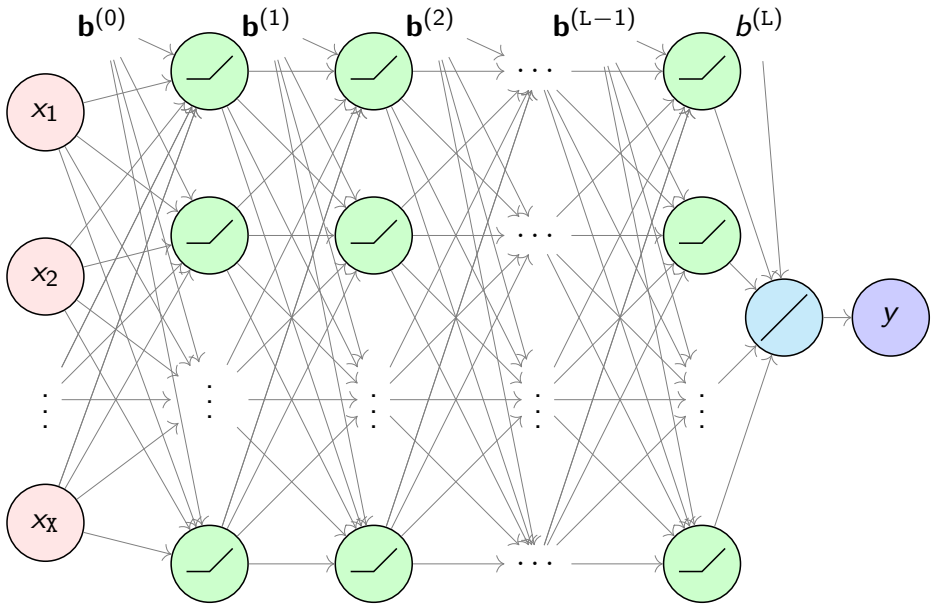
Introduction to the Neural Networks

Data

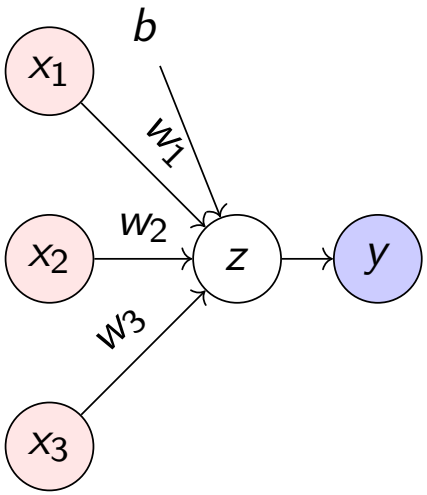
$$\mathcal{D} = \{(\mathbf{x}_i, y_i), \forall i = 1 : \mathtt{D}\}$$

$$\mathbf{x}_i \in \mathbb{R}^{\mathtt{X}} : \begin{cases} \text{Covariate} \\ \text{attribute} \\ \text{regressor} \end{cases}$$

$$y_i \in \mathbb{R} : \text{Observation}$$



Linear Regression $\rightarrow$ Neural Network



Covariates

$$\mathbf{x} = [x_1, x_2, x_3]^\intercal$$

Hidden state variable

$$z = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

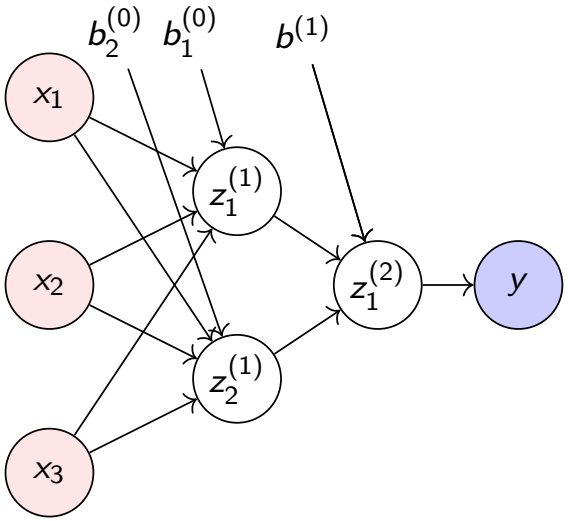
Observation equation

$$y = z + v, \;\; v : V \sim \mathcal{N}(v; 0, \sigma_V^2)$$

Model parameters

$$\boldsymbol{\theta} = \{w_1, w_2, w_3, b\}$$

Linear Regression with multiple layers $\rightarrow$ Neural Network



Covariates

$$\mathbf{x} = [x_1, x_2, x_3]^\intercal$$

Hidden state variables

$$\underline{\text{Layer 1}} \begin{cases} z_1^{(1)} = w_{1,1}^{(0)} x_1 + w_{2,1}^{(0)} x_2 + w_{3,1}^{(0)} x_3 + b_1^{(0)} \\ z_2^{(1)} = w_{1,2}^{(0)} x_1 + w_{2,2}^{(0)} x_2 + w_{3,2}^{(0)} x_3 + b_2^{(0)} \end{cases}$$

$$\underline{\text{Layer 2}} \begin{cases} z_1^{(2)} = w_{1,1}^{(1)} z_1^{(1)} + w_{2,1}^{(1)} z_2^{(1)} + b^{(1)} \end{cases}$$

Observation equation

$$y = z_1^{(2)} + v, \;\; v : V \sim \mathcal{N}(v; 0, \sigma_V^2)$$

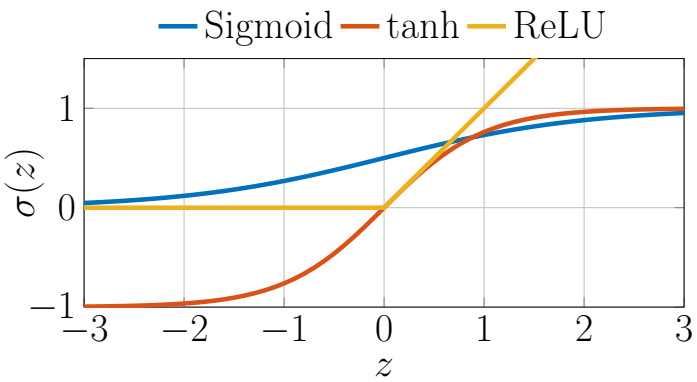
Model parameters

$$\begin{aligned} \boldsymbol{\theta} \;\; = \;\; \{ & w_{1,1}^{(0)}, w_{2,1}^{(0)}, w_{3,1}^{(0)}, b_1^{(0)}, w_{1,2}^{(0)}, w_{2,2}^{(0)}, w_{3,2}^{(0)}, \\ & \cdots b_2^{(0)}, w_{1,1}^{(1)}, w_{2,1}^{(1)}, b^{(1)} \} \end{aligned}$$

Linear Regression → Limitation

No matter how many layers you include → linear relationship between x and y

Activation Functions



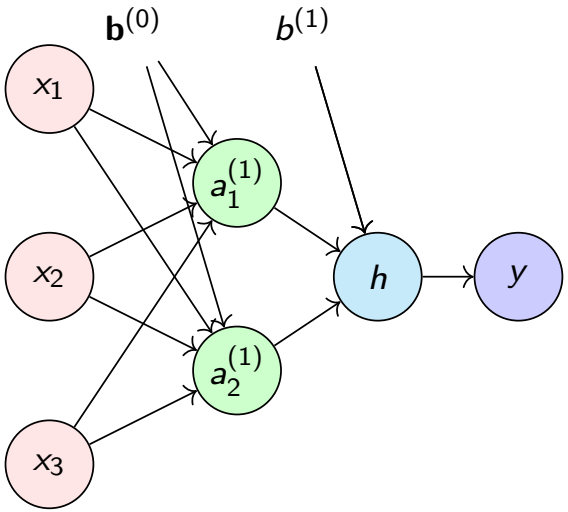
$$\sigma^{\mathtt{s}}(z) = \frac{1}{1+\exp(-z)} \quad \text{(Sigmoid)}$$

$$\sigma^{\mathtt{t}}(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \quad \text{(tanh)}$$

$$\sigma^{\mathtt{r}}(z) = \max(0,z) \quad \text{(ReLU)}$$

Activation unit: $a_i^{(j)} = \sigma\left(z_i^{(j)}\right)$

Single Layer Feedforward Neural Network



Covariates

$$\mathbf{x} = [x_1, x_2, x_3]^\intercal$$

Hidden state variables

$$\underline{\text{Layer 1}} \begin{cases} z_1^{(1)} = w_{1,1}^{(0)} x_1 + w_{2,1}^{(0)} x_2 + w_{3,1}^{(0)} x_3 + b_1^{(0)} \\ z_2^{(1)} = w_{1,2}^{(0)} x_1 + w_{2,2}^{(0)} x_2 + w_{3,2}^{(0)} x_3 + b_2^{(0)} \end{cases}$$

$$\underline{\text{Output}} \begin{cases} h = w_1^{(1)} a_1^{(1)} + w_2^{(1)} a_2^{(1)} + b^{(1)} \end{cases}$$

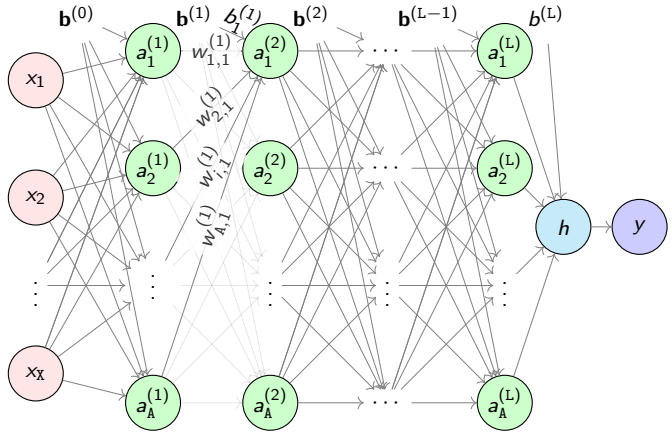
Activation units

$$\begin{aligned} a_1^{(1)} &= \sigma^{\mathrm{r}}\left(z_1^{(1)}\right) \\ a_2^{(1)} &= \sigma^{\mathrm{r}}\left(z_2^{(1)}\right) \end{aligned}$$

Model parameters

$$\begin{aligned} \boldsymbol{\theta} = \{ & w_{1,1}^{(0)}, w_{2,1}^{(0)}, w_{3,1}^{(0)}, b_1^{(0)}, w_{1,2}^{(0)}, w_{2,2}^{(0)}, w_{3,2}^{(0)}, \\ & \cdots b_2^{(0)}, w_1^{(1)}, w_2^{(1)}, b^{(1)} \} \end{aligned}$$

Multi-layer perceptron (MLP)



Activation unit in layer $j+1$

Hidden variable in layer $j+1$

Weights

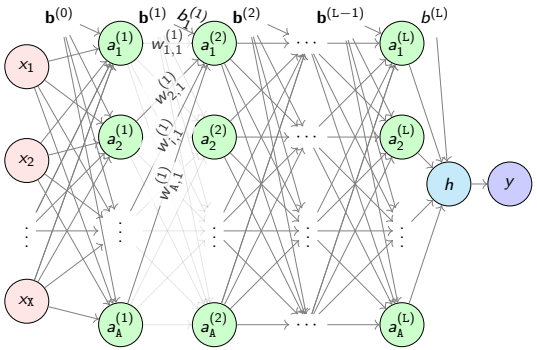
Bias

$$a_i^{(j+1)} = \sigma\left(z_i^{(j+1)}\right) = \sigma\left(w_{1,i}^{(j)} a_1^{(j)} + w_{2,i}^{(j)} a_2^{(j)} + \cdots + w_{\mathrm{A},i}^{(j)} a_{\mathrm{A}}^{(j)} + b_i^{(j)}\right)$$

Activation function

Activation units from layer j

Multi-layer perceptron (MLP) – Matrix Notation



$$\begin{aligned}\mathbf{z}^{(j+1)} &= \mathbf{W}^{(j)}\mathbf{a}^{(j)} + \mathbf{b}^{(j)} \\ &= \mathbf{W}^{(j)}\sigma(\mathbf{z}^{(j)}) + \mathbf{b}^{(j)}\end{aligned}$$

$$h = z^{(\mathrm{L})} = w_1^{(\mathrm{L})} a_1^{(\mathrm{L})} + w_2^{(\mathrm{L})} a_2^{(j)} + \cdots + w_{\mathrm{A}}^{(\mathrm{L})} a_{\mathrm{A}}^{(\mathrm{L})} + b^{(\mathrm{L})}$$

$$\underbrace{\begin{bmatrix} z_1^{(j+1)} \\ z_2^{(j+1)} \\ \vdots \\ z_{\mathrm{A}}^{(j+1)} \end{bmatrix}_{\mathrm{A}\times 1}}_{\mathbf{z}^{(j+1)}} = \underbrace{\begin{bmatrix} w_{1,1}^{(j)} & w_{1,2}^{(j)} & \cdots & w_{1,\mathrm{A}}^{(j)} \\ w_{2,1}^{(j)} & w_{2,2}^{(j)} & \cdots & w_{2,\mathrm{A}}^{(j)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{\mathrm{A},1}^{(j)} & w_{\mathrm{A},2}^{(j)} & \cdots & w_{\mathrm{A},\mathrm{A}}^{(j)} \end{bmatrix}_{\mathrm{A}\times\mathrm{A}}}_{\mathbf{W}^{(j)}} \times \underbrace{\sigma\left(\begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \vdots \\ z_{\mathrm{A}}^{(j)} \end{bmatrix}\right)_{\mathrm{A}\times 1}}_{\mathbf{a}^{(j)}} + \underbrace{\begin{bmatrix} b_1^{(j)} \\ b_2^{(j)} \\ \vdots \\ b_{\mathrm{A}}^{(j)} \end{bmatrix}_{\mathrm{A}\times 1}}_{\mathbf{b}^{(j)}}$$

Multi-layer perceptron (MLP) – Parameters

For fully connected network with $\mathtt{L}$ layers, each having $\mathtt{A}$ activation units,

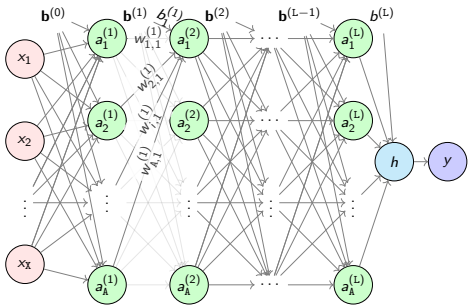
$$\begin{aligned} \boldsymbol{\theta}_w &\in \mathbb{R}^{\mathtt{A}\times\mathtt{A}\times(\mathtt{L}-1)} && \text{weight parameters, } w^l_{i,j} \\ \boldsymbol{\theta}_b &\in \mathbb{R}^{\mathtt{A}\times\mathtt{L}} && \text{bias parameters, } b^l_j \end{aligned}$$

For $\mathtt{L} = 2,\ \mathtt{A} = 1000$

$$\boldsymbol{\theta} = \{\boldsymbol{\theta}_w, \boldsymbol{\theta}_b\} \in \mathbb{R}^{\approx 10^6}$$

Joint Log-likelihood – $\boldsymbol{\theta} = \{\boldsymbol{\theta}_w, \boldsymbol{\theta}_b\}$

$$\begin{aligned}\mathcal{D} &= \{\mathcal{D}_x, \mathcal{D}_y\} \\ &= \{(\mathbf{x}_i, y_i), \forall i = 1 : \mathtt{D}\}\end{aligned} \qquad f(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta}) = \prod_{i=1}^{\mathtt{D}} \mathcal{N}(y_i; h(\mathbf{x}_i; \boldsymbol{\theta}), \sigma_V^2)$$



$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J(\mathcal{D}; \boldsymbol{\theta})$$

$$\begin{aligned}\ln f(\mathcal{D}_y|\mathcal{D}_x, \boldsymbol{\theta}) &= \sum_{i=1}^{\mathtt{D}} \ln(\mathcal{N}(y_i; h(\mathbf{x}_i; \boldsymbol{\theta}), \sigma_V^2)) \\ &\propto -\frac{1}{\mathtt{D}} \sum_{i=1}^{\mathtt{D}} \frac{1}{2} ||h(\mathbf{x}_i; \boldsymbol{\theta}) - y_i||^2 \\ &= -\frac{1}{\mathtt{D}} \sum_{i=1}^{\mathtt{D}} J(\mathbf{x}_i, y_i; \boldsymbol{\theta}) \\ &= -J(\mathcal{D}; \boldsymbol{\theta})\end{aligned}$$

Parameter estimation $\rightarrow$ Gradient descent

$$w_{i,j}^{(l)} \leftarrow w_{i,j}^{(l)} - \lambda \nabla_{w_{i,j}^{(l)}} J(\mathcal{D}; \boldsymbol{\theta})$$
$$b_{j}^{(l)} \leftarrow b_{j}^{(l)} - \lambda \nabla_{b_{j}^{(l)}} J(\mathcal{D}; \boldsymbol{\theta})$$

λ: **Learning rate**

$\nabla_{w_{i,j}^{(l)}} J(\mathcal{D}; \boldsymbol{\theta})$: gradient w.r.t $w_{i,j}^{(l)}$

$\nabla_{b_{j}^{(l)}} J(\mathcal{D}; \boldsymbol{\theta})$: gradient w.r.t $b_{j}^{(l)}$

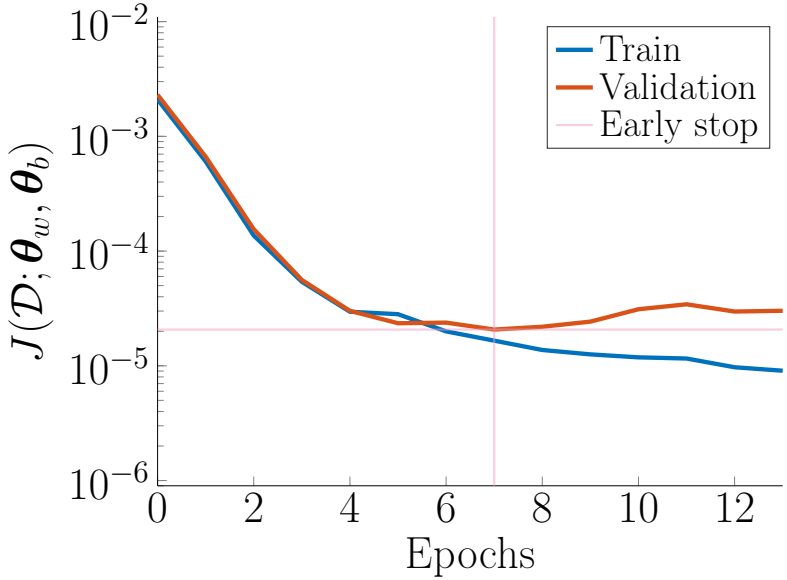
Gradients are estimated using **backpropagation** (Chain rule)

Estimating $\boldsymbol{\theta}^*$ is done using **(stochastic) gradient descent** (ADAM, ADAGRAD, etc...) over multiple **epochs**

- **SGD**: Compute ∇ using small batches of data
- **Epoch**: Seeing the entire dataset once
- **Early stop**: Stop when the loss on the validation set increases

Parameter estimation $\rightarrow$ Gradient descent

$$w_{i,j}^{(l)} \leftarrow w_{i,j}^{(l)} - \lambda \nabla_{w_{i,j}^{(l)}} J(\mathcal{D}; \boldsymbol{\theta})$$
$$b_{j}^{(l)} \leftarrow b_{j}^{(l)} - \lambda \nabla_{b_{j}^{(l)}} J(\mathcal{D}; \boldsymbol{\theta})$$



Estimating $\boldsymbol{\theta}^*$ is done using **(stochastic) gradient descent** (ADAM, ADAGRAD, etc...) over multiple **epochs**

- **SGD**: Compute ∇ using small batches of data
- **Epoch**: Seeing the entire dataset once
- **Early stop**: Stop when the loss on the validation set increases

Parameter estimation & data standardization

In order to perform well, NN need to work on **standardized data**

$$\begin{aligned} \tilde{\mathbf{x}}_i &= \frac{\mathbf{x}_i - \boldsymbol{\mu}_{\boldsymbol{X}}}{\boldsymbol{\sigma}_{\boldsymbol{X}}} \\ \tilde{\mathbf{y}}_i &= \frac{\mathbf{y}_i - \boldsymbol{\mu}_{\boldsymbol{Y}}}{\boldsymbol{\sigma}_{\boldsymbol{Y}}} \end{aligned}$$

Then you need to destandardize the NN output $g(\tilde{\boldsymbol{x}}; \boldsymbol{\theta}) \odot \boldsymbol{\sigma}_{\boldsymbol{Y}} + \boldsymbol{\mu}_{\boldsymbol{Y}}$

Why? Because otherwise, you would need to initialize

$$\underbrace{\boldsymbol{\theta} = \text{fct}(\boldsymbol{\mu}_{\boldsymbol{X}}, \boldsymbol{\sigma}_{\boldsymbol{X}}, \boldsymbol{\mu}_{\boldsymbol{Y}}, \boldsymbol{\sigma}_{\boldsymbol{Y}})}_{\text{✗}}$$

UCI - Concrete Slump dataset

Concrete Slump Test Data Set

Download: Data Folder, Data Set Description

Abstract: Concrete is a highly complex material. The slump flow of concrete is not only determined by the water content, but that is also influenced by other concrete ingredients.

Data Set Characteristics:	Multivariate	**Number of Instances:**	103	**Area:**	Computer
Attribute Characteristics:	Real	**Number of Attributes:**	10	**Date Donated**	2009-04-30
Associated Tasks:	Regression	**Missing Values?**	N/A	**Number of Web Hits:**	132011

Source:

Donor: I-Cheng Yeh
Email: icyeh '@' chu.edu.tw
Institution: Department of Information Management, Chung-Hua University (Republic of China)
Other contact information: Department of Information Management, Chung-Hua University, Hsin Chu, Taiwan 30067, R.O.C.

`https://archive.ics.uci.edu/ml/datasets/Concrete+Slump+Test`

1 output [MPa]

- 28-day compressive strength

7 covariates [kg/m^3]

- Cement
- Slag
- Fly ash
- Water
- Superplasticizer
- Coarse aggregate
- Fine aggregate

Problem set-up

Covariates and observations:

$$\mathbf{x}_i = [x_1, x_2, \cdots, x_7]_i^\intercal$$

$$\tilde{\mathcal{D}} = \{\tilde{\mathcal{D}}_x, \tilde{\mathcal{D}}_y\} = \{(\frac{\mathbf{x}_i - \boldsymbol{\mu}_{\boldsymbol{X}}}{\boldsymbol{\sigma}_{\boldsymbol{X}}}, \frac{\mathbf{y}_i - \boldsymbol{\mu}_{\boldsymbol{Y}}}{\boldsymbol{\sigma}_{\boldsymbol{Y}}}),\ i = 1:103\}$$

Observation model:

$$\underbrace{\tilde{y}_i}_{\text{observation}} = \overbrace{g(\tilde{\mathbf{x}}_i)}^{\text{true strength}}$$

Neural network setup

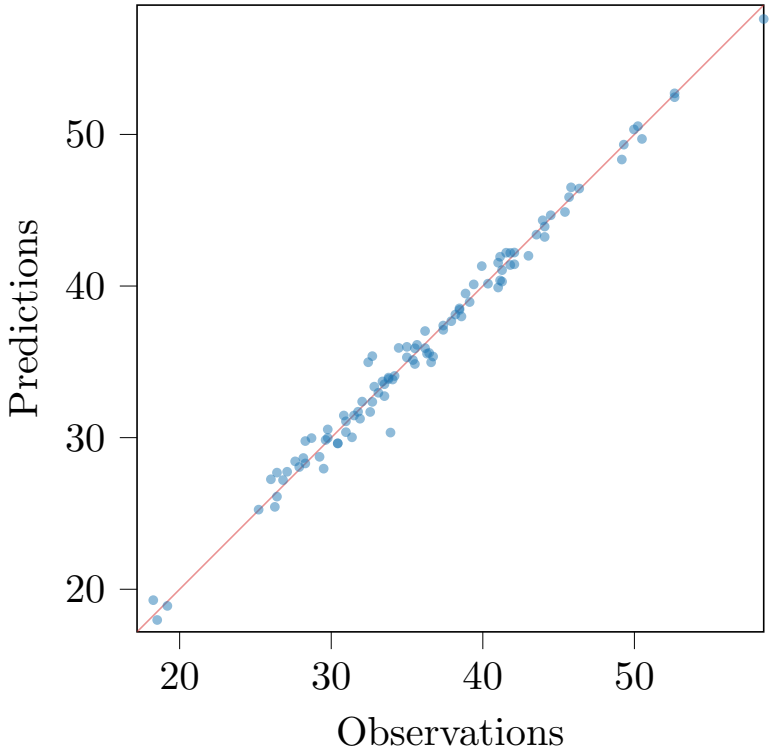
Loss function

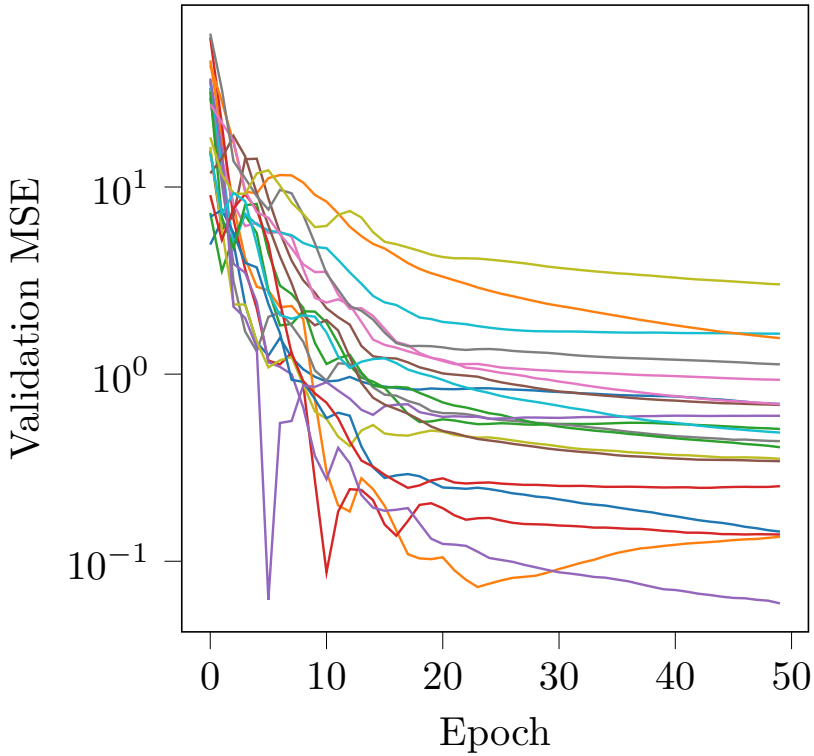
$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \overbrace{\frac{1}{\mathtt{D}} \sum_{i=1}^{\mathtt{D}} (\tilde{y}_i - g(\tilde{\boldsymbol{x}}_i; \boldsymbol{\theta}))^2}^{\text{Mean square error}}$$

Network configuration

- $\mathtt{L} = 1$ (nb. hidden layer)
- $\mathtt{A} = 200$ (nb. hidden units/layer)
- $\mathtt{E} = 50$ (nb. epochs)
- $\mathtt{B} = 20$ (Batch size)
- $\mathtt{R} = 5$ (nb. of repetitions)
- learning rate $= 0.005$

Results 20-folds CV $\times$ 5 repetitions []

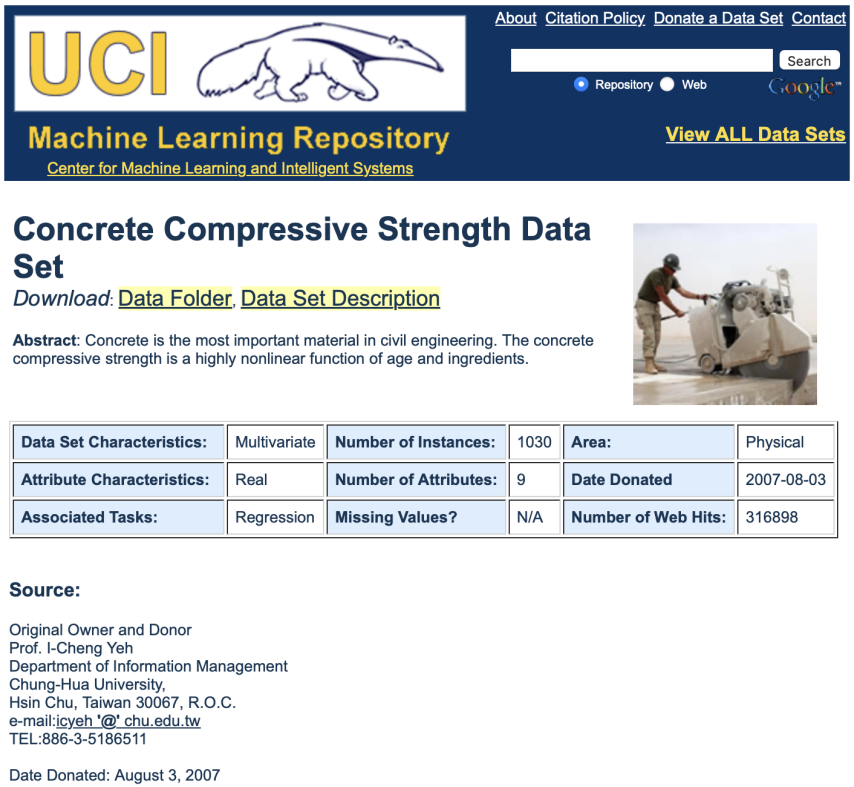




Mean square error (MSE)	$0.696 \pm 0.14 \gg 0.118$ (GPR) ⚠
Mean absolute error (MAE)	0.572 ± 0.03
Mean error (ME)	-0.003 ± 0.04
Std error (RMSE)	0.830 ± 0.08

[CIV_ML/NN_example_Concrete_slump.py]

UCI - Concrete Strength dataset: GPR v.s. NN



`https://archive.ics.uci.edu/ml/datasets/concrete+compressive+strength`

1 output [MPa]

- 28-day concrete strength

8 covariates

- Cement [kg/m^3]
- Blast Furnace Slag [kg/m^3]
- Fly ash [kg/m^3]
- Water [kg/m^3]
- Superplasticizer [kg/m^3]
- Coarse aggregate [kg/m^3]
- Fine aggregate [kg/m^3]
- Age [days]

Problem set-up

Covariates and observations:

$$\mathbf{x}_i = [x_1, x_2, \cdots, x_8]_I^\intercal$$

$$\tilde{\mathcal{D}} = \{\tilde{\mathcal{D}}_x, \tilde{\mathcal{D}}_y\} = \{(\frac{\mathbf{x}_i - \boldsymbol{\mu}_{\boldsymbol{X}}}{\boldsymbol{\sigma}_{\boldsymbol{X}}}, \frac{\mathbf{y}_i - \boldsymbol{\mu}_{\boldsymbol{Y}}}{\boldsymbol{\sigma}_{\boldsymbol{Y}}}),\ i = 1:1030\}$$

Observation models:

GPR

$$\underbrace{\tilde{y}_i}_{\text{observation}} = \overbrace{g(\tilde{\mathbf{x}}_i)}^{\text{true strength}} + \underbrace{v_i}_{\text{meas. error}}$$

NN

$$\underbrace{\tilde{y}_i}_{\text{observation}} = \overbrace{g(\tilde{\mathbf{x}}_i)}^{\text{true strength}}$$

Model setup

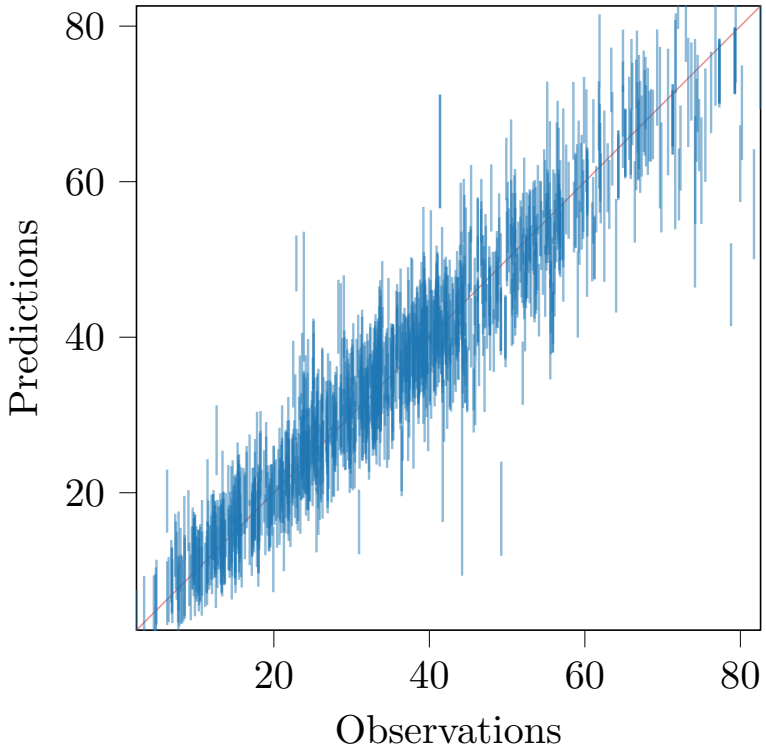
GPR

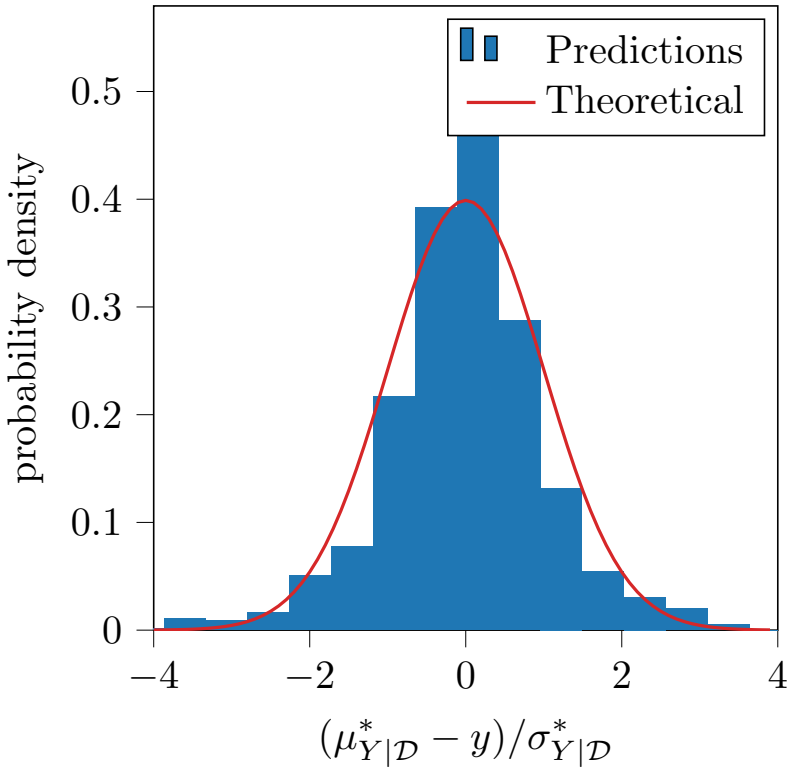
- Square exponential correlation fct. (RBF)
- $\emptyset \rightarrow$ default parameters

NN

- Loss function: MSE
- $\mathtt{L} = 1$ (nb. hidden layer)
- $\mathtt{A} = 500$ (nb. hidden units)
- $\mathtt{E} = 50$ (nb. epochs)
- $\mathtt{B} = 100$ (Batch size)
- $\mathtt{R} = 5$ (nb. of repetitions)
- learning rate $= 0.005$

GPR 20-folds CV [🐍]

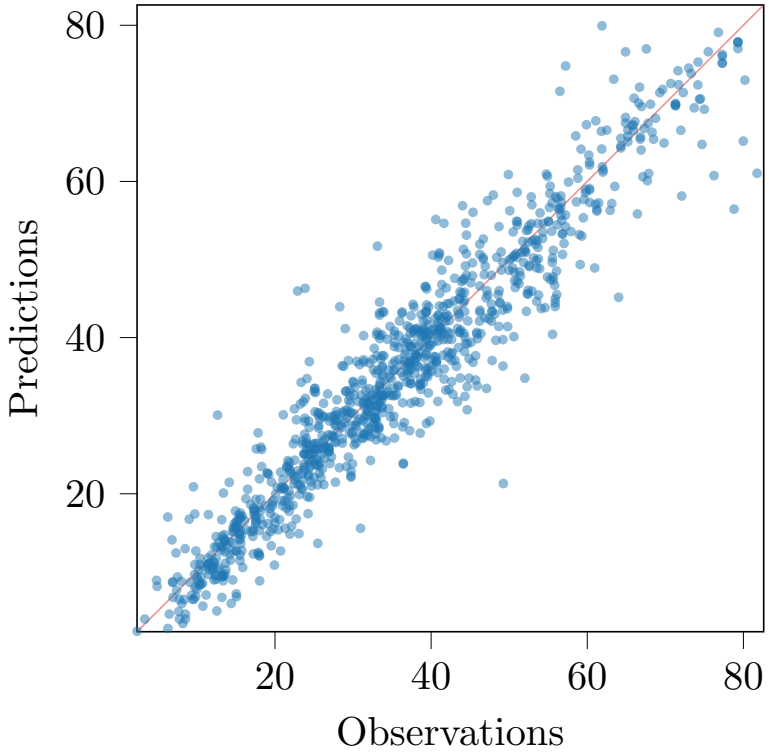


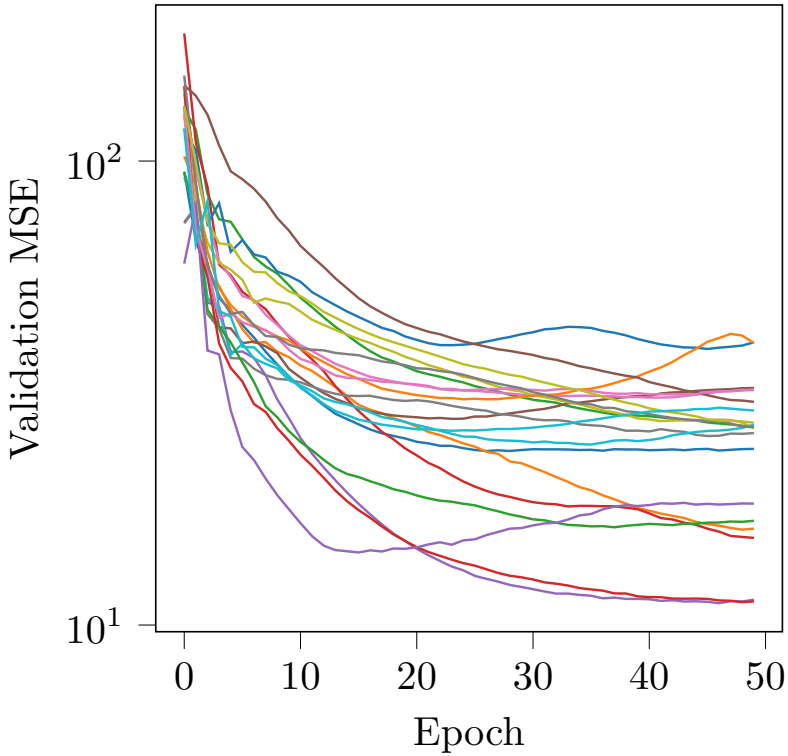


Mean square error (MSE)	23.9
Mean absolute error (MAE)	3.4
Mean error (ME)	-0.006
Std error (RMSE)	4.9 ± 0.08

[CIV_ML/GPR_example_Concrete.py]

NN 20-folds CV $\times$ 5 repetitions []





Mean square error (MSE)	$23.1 \pm 0.1 \approx 23.9$ (GPR) ⚠
Mean absolute error (MAE)	3.5 ± 0.02
Mean error (ME)	-0.2 ± 0.03
Std error (RMSE)	4.9 ± 0.01

[CIV_ML/NN_example_Concrete_slump.py]

Defining a FNN with PyTorch []

```
1   import torch
2   import numpy as np
3   import pandas as pd
4
5   #Load the data
6   data = pd.read_excel('Data.xls','Sheet1', header=None,skiprows=1)
7   data = pd.read_csv('Data.csv', header=None,skiprows=1)
8
9   data_array = np.array(data)
10  data_array = np.random.permutation(data_array) #Randomly shuffle the rows
11
12  input_size, output_size, nb_units = 10, 1, 25
13  x = data_array[:,0:input_size]
14  y = data_array[:,input_size:input_size-1+output_size]
15
16  #Define the network
17  model = torch.nn.Sequential(
18    torch.nn.Linear(input_size,nb_units),
19    torch.nn.ReLU(),
20    torch.nn.Linear(nb_units,output_size))
21  criterion = torch.nn.MSELoss()                    #Define loss function
22  optimizer = torch.optim.Adam(model.parameters(),lr=0.005) #lr: learning rate
```

Note: you need to reinitialize the model for every CV fold

Training a FNN with PyTorch []

```python
#CV index selection
idx = ... #Define your index for the train/validation CV fold
x_train = np.delete(x, idx, 0)
y_train = np.delete(y, idx, 0)
x_val = x[idx]
#Standardization
mx = np.mean([x_train],axis=1)
sx = np.std([x_train],axis=1)
my = np.mean([y_train])
sy = np.std([y_train])
x_train = (x_train-mx)/sx
y_train = (y_train-my)/sy
x_val = (x_val-mx)/sx
x_train = torch.tensor(x_train,dtype=torch.float32) #Transform data to tensor
y_train = torch.tensor(y_train,dtype=torch.float32) #Transform data to tensor
#Training
for e in range(nb_epochs):  #Loop over epochs
  nb_batch = ...
  for b in range(nb_batch): #Loop over batches
    idx_b = ... #Define index indicating which observation to use
    y_pred = model(x_train[idx_b])                          #Predict
    loss = criterion(y_pred,y_train[idx_b].reshape(-1,1))   #Calculate loss
    optimizer.zero_grad()                                   #Backprop
    loss.backward()
    optimizer.step()
```

Testing a FNN with PyTorch []

```
#Prediction
x_val = torch.tensor(x_val,dtype=torch.float32) #Transform data to tensor
y_pred_val = model(x_val)*sy+my   #Predict & unnormalize for the validation set
y_pred_val = np.concatenate(y_pred_val.detach().numpy())#Tensor to array
```

Summary

Linear Regression: outperformed by modern methods

Gaussian Process Regression:

- State-of-the-art for small datasets ($\approx \mathtt{D} \leq 10^3 - 10^4$)
- Allows interpolating and extrapolating (uncertainty estimates)
- Few hyperparameters

Neural Networks:

- State-of-the-art for large datasets
- Requires large datasets to outperform other methods
- Several hyperparameters

Cross-Validation:

- Prevents over-fitting
- The standard method for quantifying accuracy & comparing models