Kernel Methods for Regression

I. Santamaría, S. Van Vaerenbergh

GTAS, Universidad de Cantabria

February 10, 2020

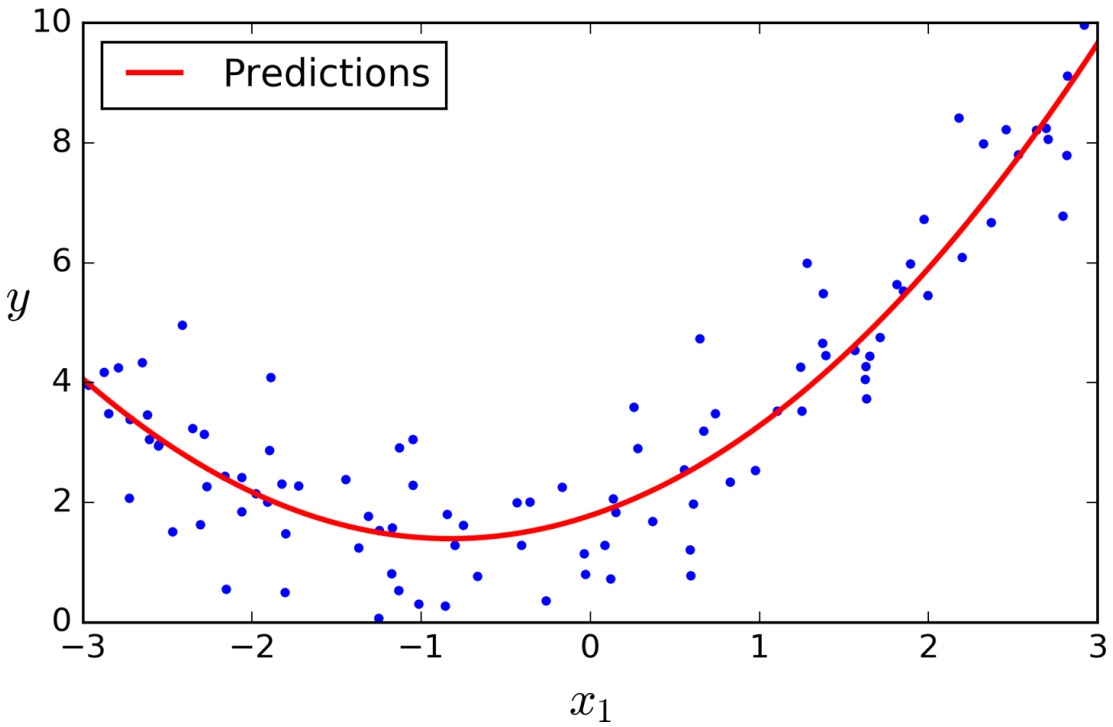
Maśter Universitario Oficial **Data Science**

con el apoyo del

Contents

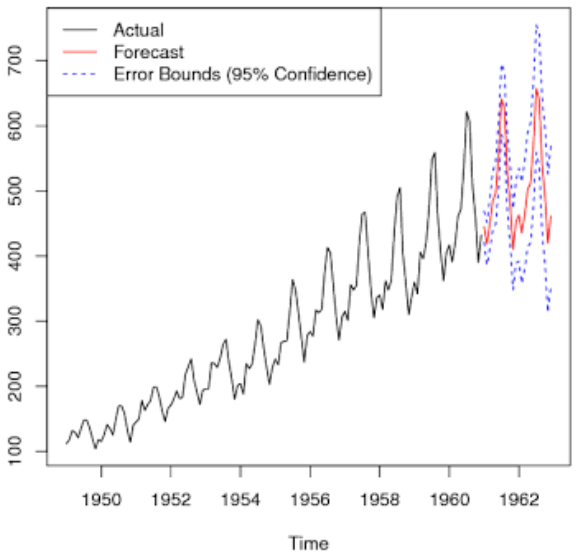
Problem formulation

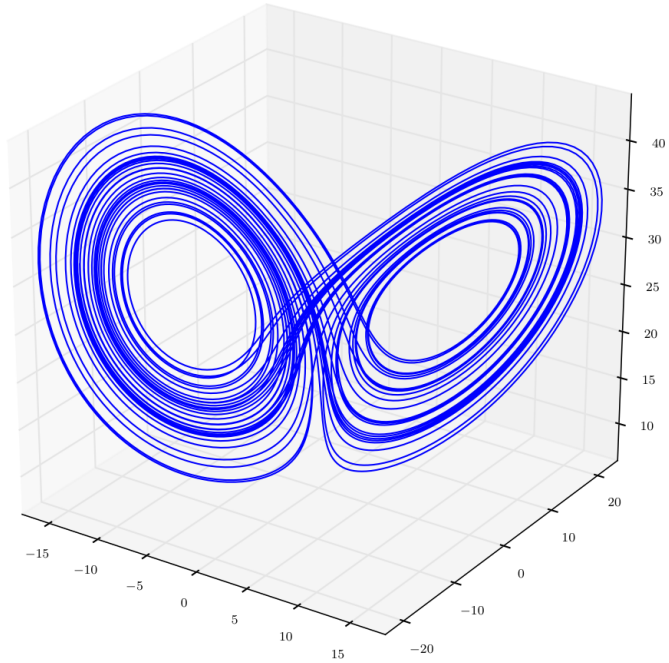
- Input: $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, $\mathbf{x}_i \in \mathcal{R}^d$
- Output: $(y_1, \ldots, y_n)$, $y_i \in \mathcal{R}$
- **Problem**: find $f(\cdot) : \mathcal{R}^d \rightarrow \mathcal{R}$ to fit (or predict) y from $\mathbf{x}$



Applications

Typical applications are **time-series prediction** or **nonlinear modeling**

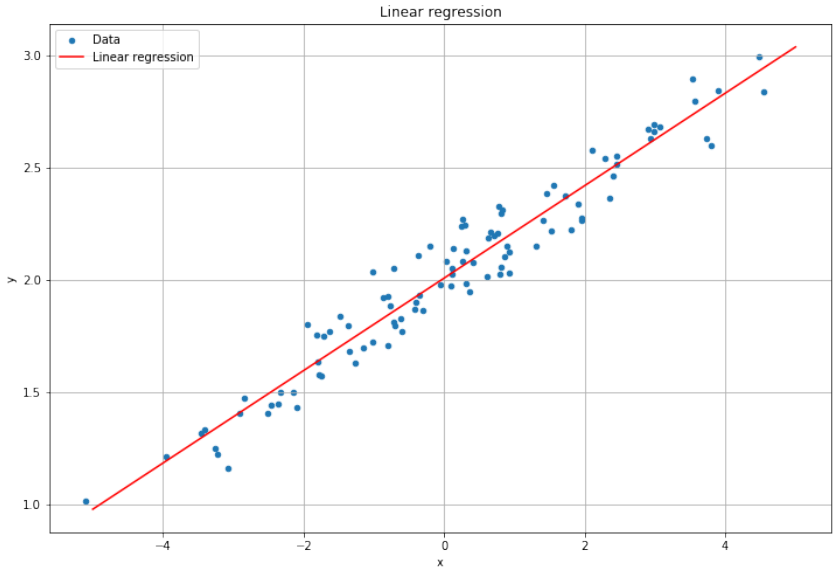




Before considering kernel methods, let us briefly review the linear case

Linear regression

- To find the hyperplane $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ that best fits the observations



- Fitting criteria (**loss function**)

$$L_2 - norm: \quad (y - f(\mathbf{x}))^2 \quad \rightarrow \textbf{Least Squares}$$

$$L_1 - norm: \quad |y - f(\mathbf{x})|$$

$$\epsilon - insensitive: \quad \max(0, |y - f(\mathbf{x})| - \epsilon)$$

Least Squares

- If we redefine $\mathbf{x} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$ and $\mathbf{w} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$, the linear function becomes

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = \mathbf{x}^T\mathbf{w}$$

- Training data set $\{(\mathbf{x}_i, y_i),\ i = 1, \ldots, n\}$ with $\mathbf{x}_i \in \mathcal{R}^d$ and $y_i \in \mathcal{R}$

$$y_i = \mathbf{x}_i^T\mathbf{w} + e_i, \qquad i = 1, \ldots, n$$

- Overdetermined system ($n > d$) of linear equations

$$\underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1d} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nd} \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}}_{\mathbf{w}} + \underbrace{\begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix}}_{\mathbf{e}}$$

Least Squares [Gauss 1794]: Minimize the L_2-norm of the error

$$J(\mathbf{w}) = \sum_{i=1}^{n} \left(y_i - \mathbf{w}^T \mathbf{x}_i\right)^2 = \sum_{i=1}^{n} e_i^2 = \|\mathbf{e}\|_2^2 = \mathbf{e}^T \mathbf{e}$$

$$J(\mathbf{w}) = \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w}$$

Pseudoinverse

$$\hat{\mathbf{w}}_{LS} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y} = \mathbf{X}^\sharp \mathbf{y}$$

$\mathbf{X}^\sharp = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T$ is the pseudoinverse or Moore-Penrose inverse of $\mathbf{X}$

It is the Maximum Likelihood (ML) estimator when the errors follow a Gaussian distribution $\mathbf{e} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$

Regularization

- Solution to overfitting or ill-conditioned problems
- Standard regularization approach: **ridge regression** or **Tikhonov regularization**

$$J(\mathbf{w}) = \sum_{i=1}^{n} e_i^2 + \lambda ||\mathbf{w}||^2$$

- The ridge regression (RR) solution is

$$\hat{\mathbf{w}}_{RR} = \left(\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^T\mathbf{y}$$

- The same idea applied in the feature space leads to **Kernel Ridge Regression** (more on this later)

Other cost functions

- There are many different cost functions to penalize errors
- A common alternative is to use the **L_1-norm** instead of the L_2-norm (more robust against outliers) $\longrightarrow$ sparse solution
- No closed-form solution, but there are very efficient algorithms to solve the problem
- From a Bayesian perspective, the cost function to be used depends on the probability density function (pdf) of the errors or noise in the data

$$loss \propto -\log\left(p(f(\mathbf{x}_i)|(\mathbf{x}_i, y_i))\right) = -\log\left(p(y_i - f(\mathbf{x}_i))\right) = -\log\left(p(e_i)\right)$$

- e_i Gaussian $\rightarrow$ $loss = \sum_i e_i^2$
- e_i Laplacian $\rightarrow$ $loss = \sum_i |e_i|$

SVM for (linear) regression

Similar to the SVM for classification $\rightarrow$ Structural Risk Minimization (SRM) Principle

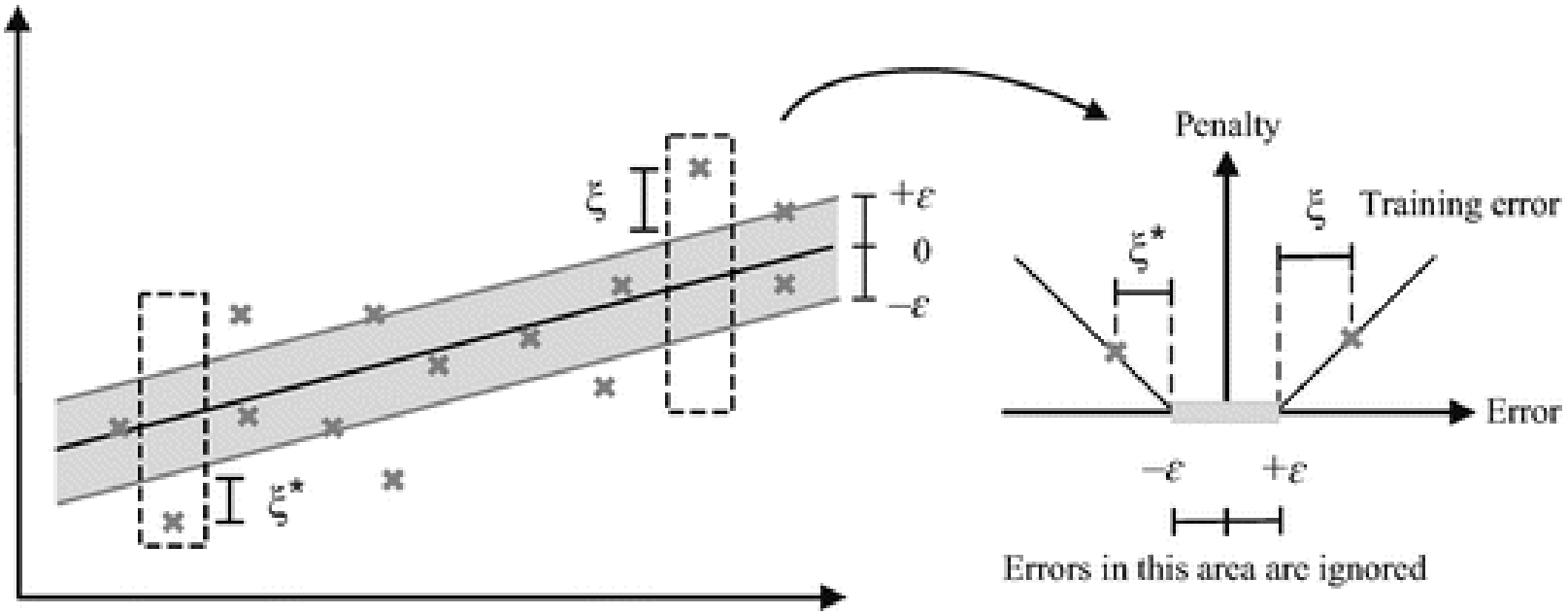
SVR (lineal)

$$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\left|y_i - \mathbf{w}^T\mathbf{x}_i\right|_\epsilon$$

- The cost function for errors is the **ϵ-insensitive loss function** (Vapnik's cost function)

$$|e_i|_\epsilon = \begin{cases} 0 & \text{if } |e_i| \leq \epsilon, \\ |e_i| - \epsilon & \text{if } |e_i| > \epsilon. \end{cases}$$

ϵ-insensitive loss function



- L_1-norm penalty for errors larger than ϵ
- Symmetric cost function
- Avoids overfitting by not considering small errors

- The solution (optimal hyperplane) admits a linear expansion in terms of the **support vectors**

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$$

- The coefficients α_i are obtained by solving the following Quadratic Programming (QP) problem

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \textstyle\sum_{i=1}^{n} y_i \alpha_i - \epsilon \sum_{i=1}^{n} |\alpha_i| - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} \quad & \textstyle\sum_i \alpha_i = 0, \\ & |\alpha_i| \leq C, \quad \forall i \end{aligned}$$

- $\alpha_i \neq 0$ only for points outside the ϵ-tube

- For a new input (test) point $\mathbf{x}$, the output is

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i^T \mathbf{x}$$

- The kernel is linear

$$\mathbf{x}_i^T\mathbf{x} = \langle \mathbf{x}_i, \mathbf{x} \rangle = k(\mathbf{x}_i, \mathbf{x}_j)$$

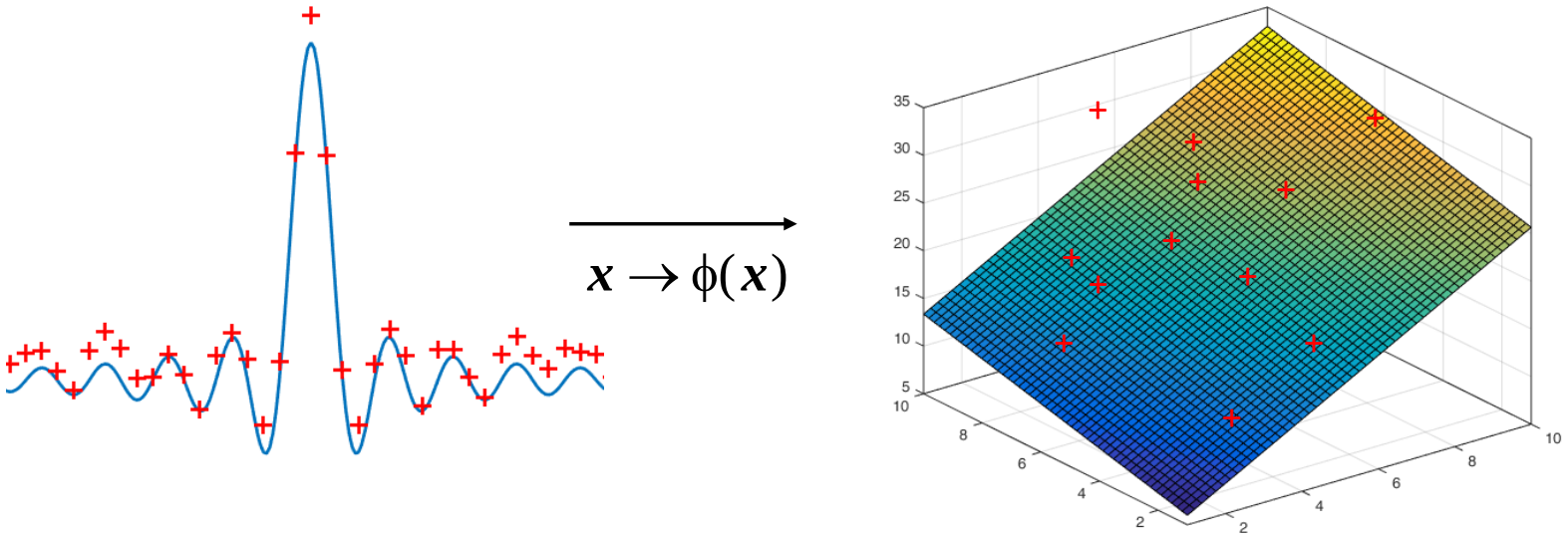
  therefore

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x})$$

- The **kernel trick** allow us to extend this result to nonlinear regression

Support Vector Regression (SVR)

- The problem formulation as well as the solution are based on inner products between input data points
- **Kernel trick**: A linear regressor in the transformed (feature) space becomes a nonlinear regressor in the input space



- Substitute the linear kernel $\mathbf{x}_i^T \mathbf{x}_j$ by a nonlinear kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$$

  - **Polynomial** (parameters p and c)

$$k(\mathbf{x}_i, \mathbf{x}_j) = \left(\mathbf{x}_i^T \mathbf{x}_j + c\right)^p$$

  - **Gaussian** (parameter σ^2 or $\gamma = \frac{1}{2\sigma^2}$)

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

or

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

- The QP problem remains the same changing $\mathbf{x}_i^T \mathbf{x}_j$ by $k(\mathbf{x}_i, \mathbf{x}_j)$
- Defining the $n \times n$ kernel matrix $\mathbf{K}$, with elements $k(\mathbf{x}_i, \mathbf{x}_j)$, the problem is

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \mathbf{y}^T \boldsymbol{\alpha} - \epsilon |\boldsymbol{\alpha}| - \tfrac{1}{2} \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{\alpha}^T \mathbf{1} = 0, \\ & |\alpha_i| \leq C, \quad \forall i \end{aligned}$$

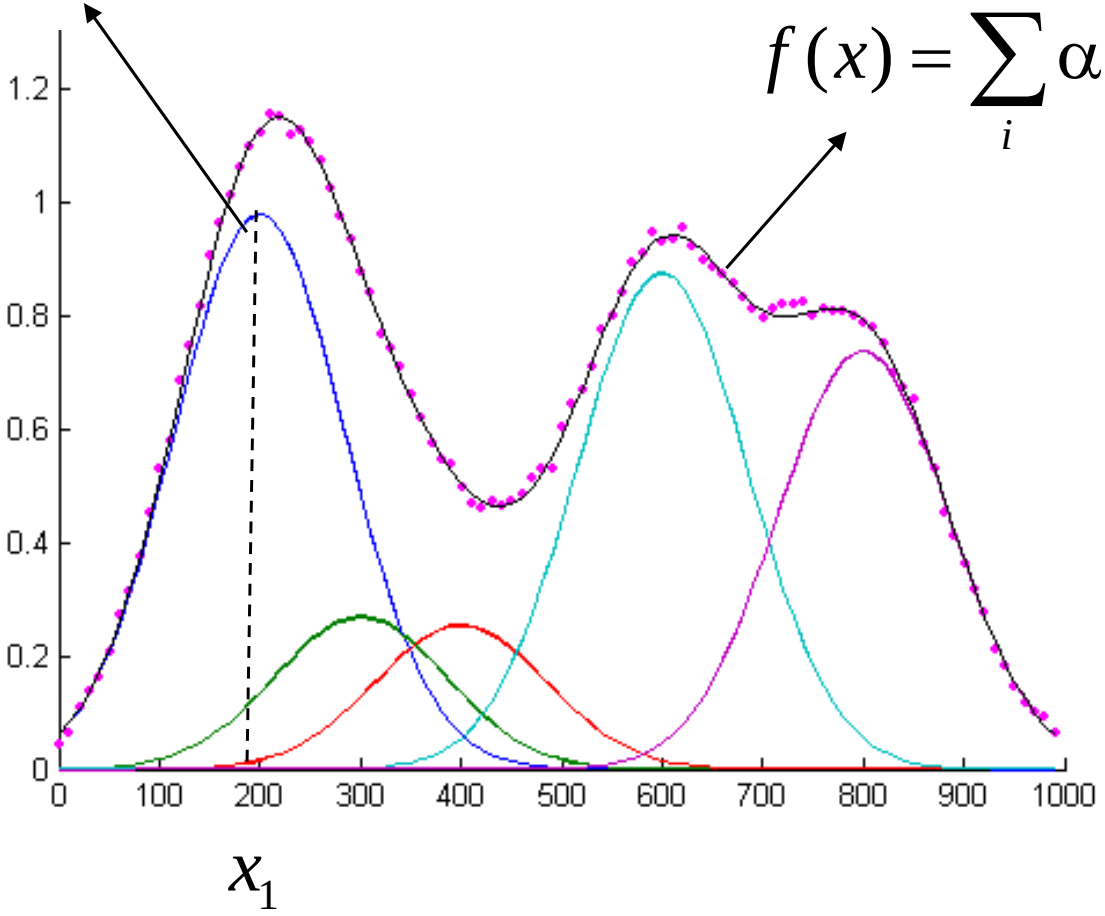
- Its solution can again be expressed as

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x})$$

SVR with Gaussian kernel

$$\alpha_1 k(x, x_1) = \alpha_1 e^{-\frac{\|x - x_1\|^2}{2\sigma^2}}$$

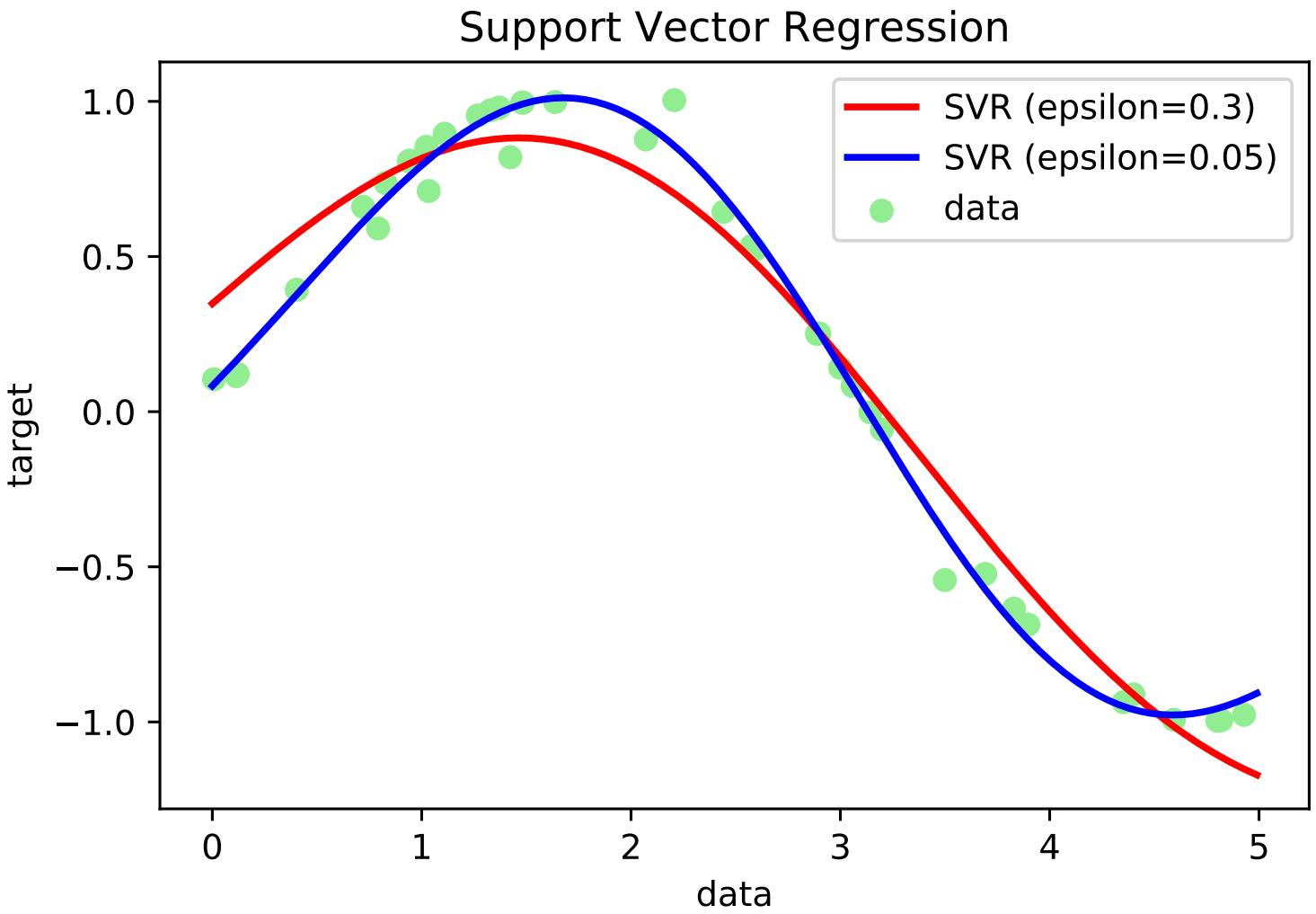
$$f(x) = \sum_i \alpha_i k(x, x_i)$$



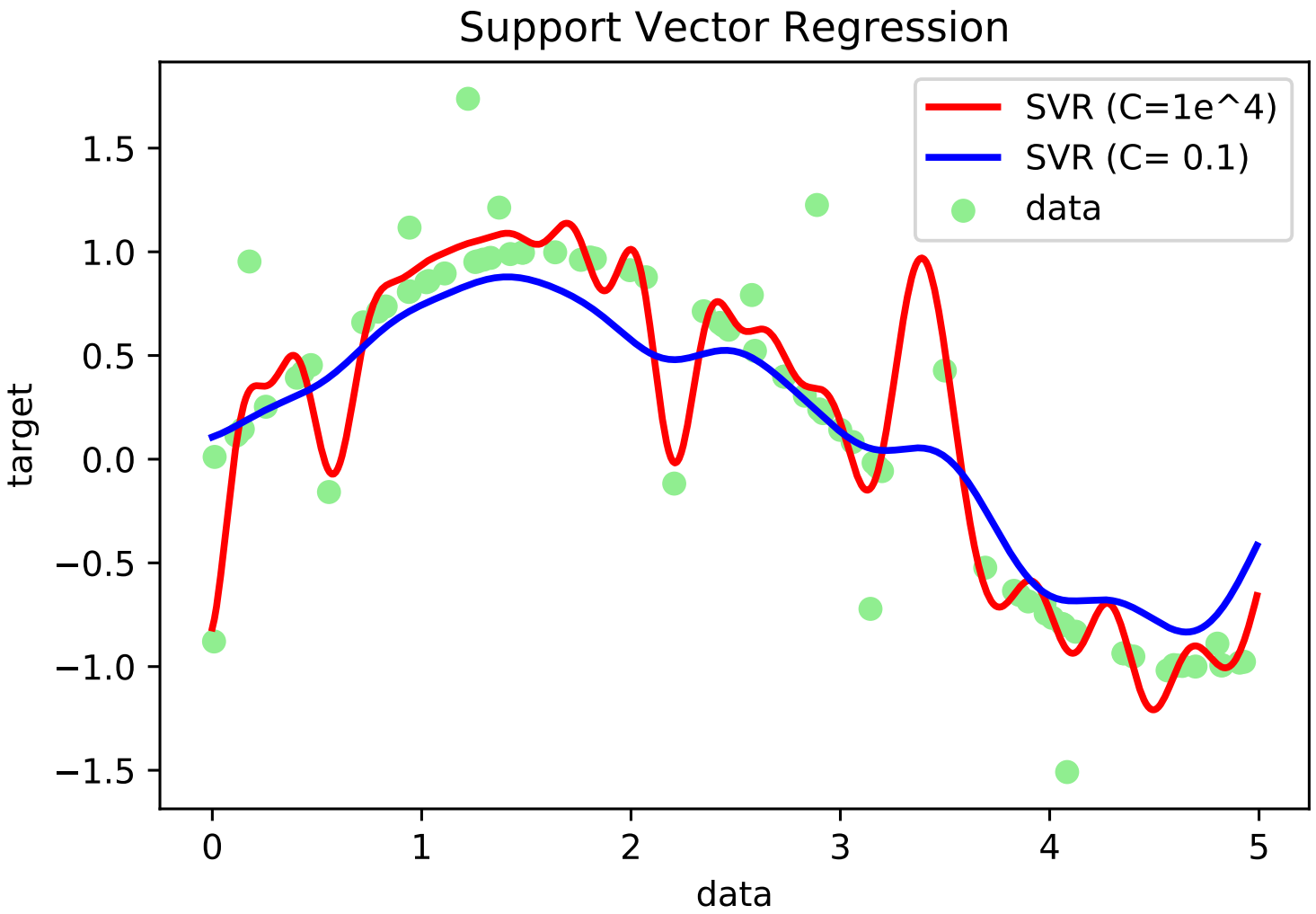
SVR: Parameter fitting

- C: Regularization parameter
  - $C \downarrow$ less penalty for errors $\rightarrow$ simple models
  - $C \uparrow$ more penalty for errors $\rightarrow$ complex models, **overfitting** risk
- ϵ: Loss function parameter
  - $\epsilon \downarrow$ small errors are penalized $\rightarrow$ complex models, **overfitting** risk
  - $\epsilon \uparrow$ only large errors are penalized $\rightarrow$ simple models

Example: $C = 10^3$, $\gamma = \frac{1}{2\sigma^2} = 0.1$



Example: $\epsilon = 0.1$, $\gamma = \frac{1}{2\sigma^2} = 10$



Kernel Ridge Regression

- Map data to a higher dimensionality, probably ∞, feature space: $\mathbf{x}_i \rightarrow \Phi(\mathbf{x}_i)$
- Solve a linear ridge regression problem in the feature space

$$\underset{f \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 + \lambda \|f\|_{\mathcal{H}}^2$$

- **Representer Theorem**: The solutionn of the regularized problem admits the following expansion

$$f(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j k(\mathbf{x}_j, \mathbf{x})$$

- Quadratic error term

$$\sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{n} \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) \right)^2 = \|\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}\|_2^2$$

- Regularization term

$$\lambda \|f\|_{\mathcal{H}}^2 = \lambda \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha}$$

- Dual problem (in terms of the $\boldsymbol{\alpha}$ coefficients)

$$\min_{\boldsymbol{\alpha}} \underbrace{\|\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}\|_2^2 + \lambda \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha}}_{J(\boldsymbol{\alpha})}$$

$$J(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^T \mathbf{K}^2 \boldsymbol{\alpha} - 2\boldsymbol{\alpha}^T \mathbf{K}\mathbf{y} + \|\mathbf{y}\|_2^2 + \lambda \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha}$$

- Taking derivatives and equating to zero

$$2\mathbf{K}^2\boldsymbol{\alpha} - 2\mathbf{K}\mathbf{y} + \lambda \mathbf{K}\boldsymbol{\alpha} = \mathbf{0}$$

- The solution for the coefficients is

$$\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}$$

KRR Summary

Given $\{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$, the kernel function $k(\cdot, \mathbf{x})$, and the regularization parameter $\lambda > 0$:

1. Build the $n \times n$ kernel matrix $\mathbf{K}$ with elements $k(\mathbf{x}_i, \mathbf{x}_j)$
2. Obtain the expansion coefficients

$$\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}$$

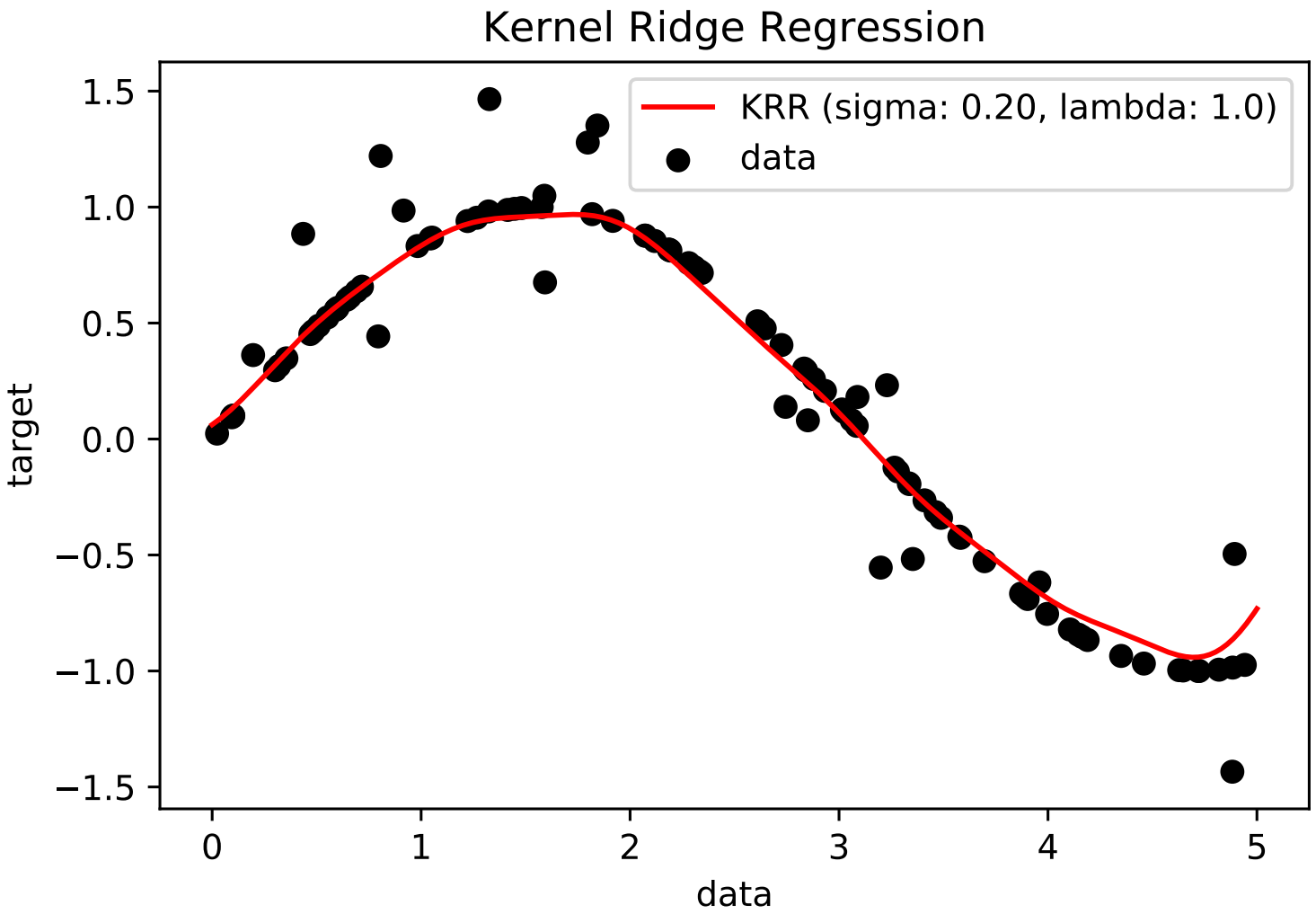
3. The output for a new input data $\mathbf{x}$ is (out-of-sample regression)

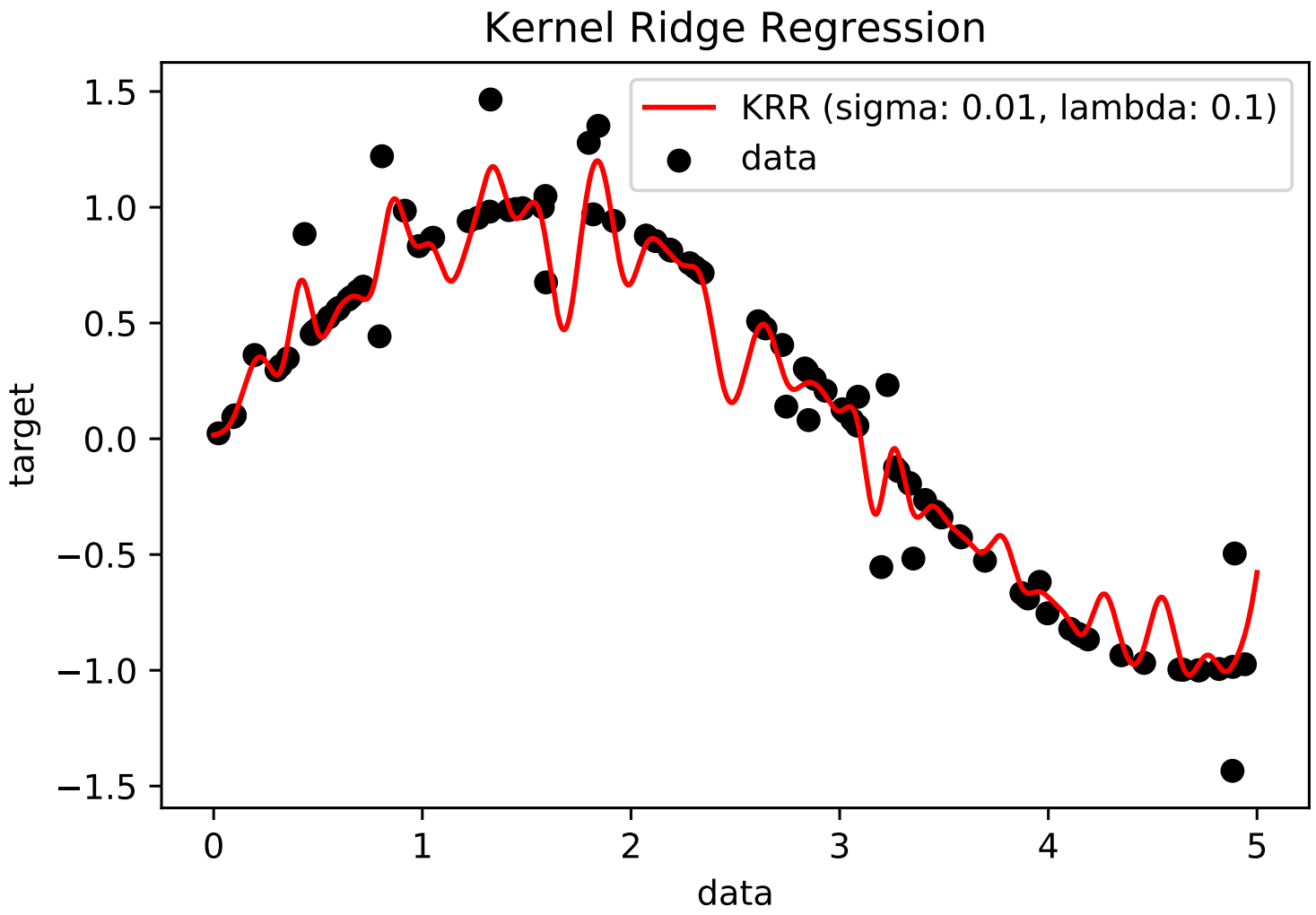
$$f(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j k(\mathbf{x}_j, \mathbf{x})$$

Discussion

- The matrix $(\mathbf{K} + \lambda \mathbf{I})$ is always invertible
- With a Gaussian kernel and $\lambda = 0$ we are interpolating the data
- The computational cost to calculate $(\mathbf{K} + \lambda \mathbf{I})^{-1}$ grows as n^3
- SVR and KRR have identical functional form: $f(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j k(\mathbf{x}_j, \mathbf{x})$, but the expansion coefficients are different
  1. SVR: sparse solution, many $\alpha_i = 0$
  2. KRR: non-sparse solution, $\alpha_i \neq 0, \forall i$
- Both SVR and KRR are batch algorithms: How can the regressors/models be updated when a new sample $(\mathbf{x}_{n+1}, y_{n+1})$ arrives?

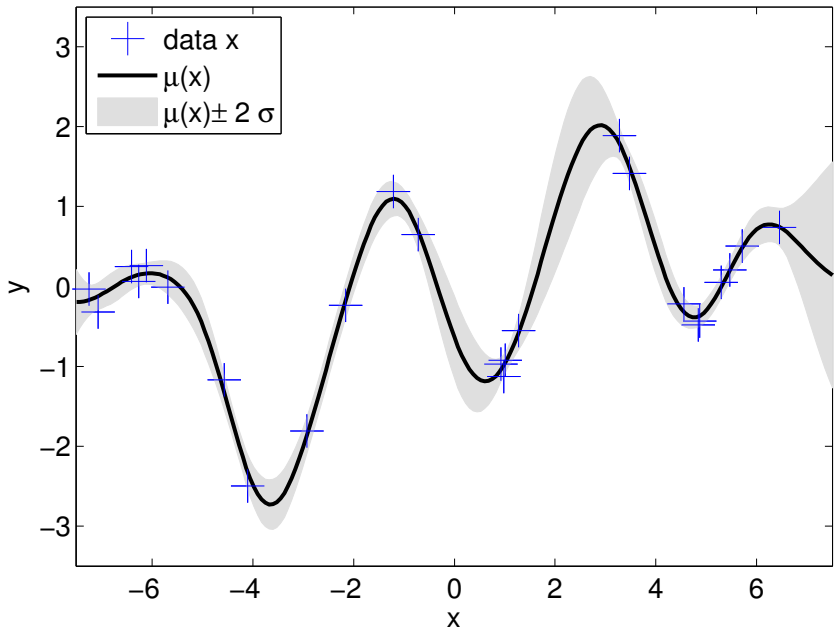
Example


Kernel Ridge Regression
KRR (sigma: 0.20, lambda: 1.0)
data
target
data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
1
2
3
4
5


Kernel Ridge Regression
KRR (sigma: 0.01, lambda: 0.1)
data
target
data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
1
2
3
4
5

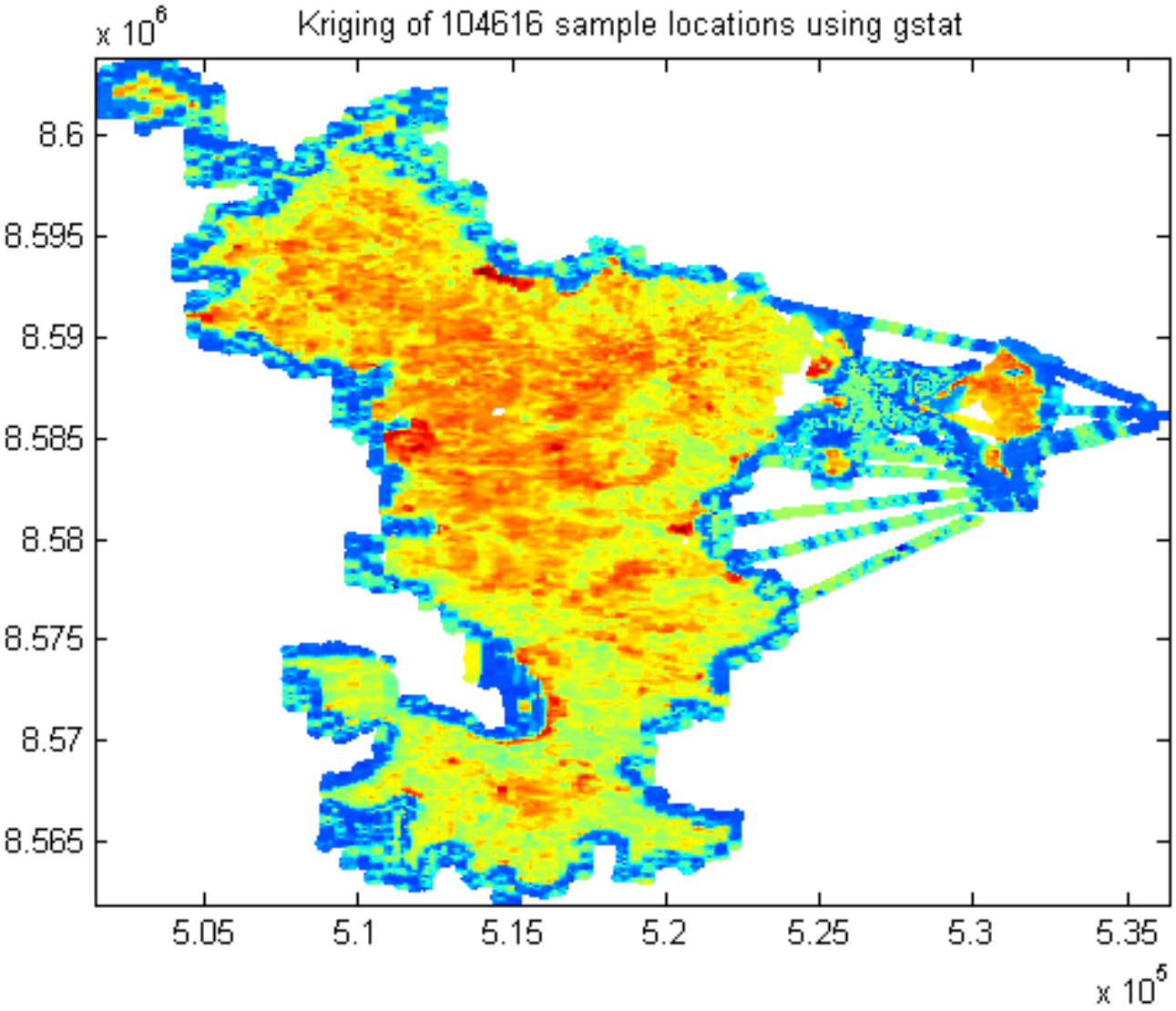
Introduction

- A limitation of SVR and KRR is that they do not provide any information about the uncertainty or confidence interval of the predictions



- **Gaussian Processes** or GPs are state-of-the-art **Bayesian** methods for regression that overcome this limitation of kernel methods

GPs are known in geostatistics as **kriging**



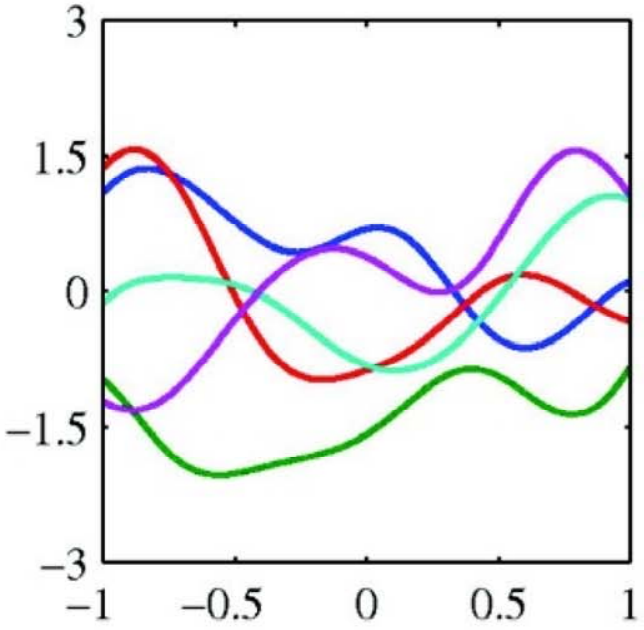
Bayes Theorem

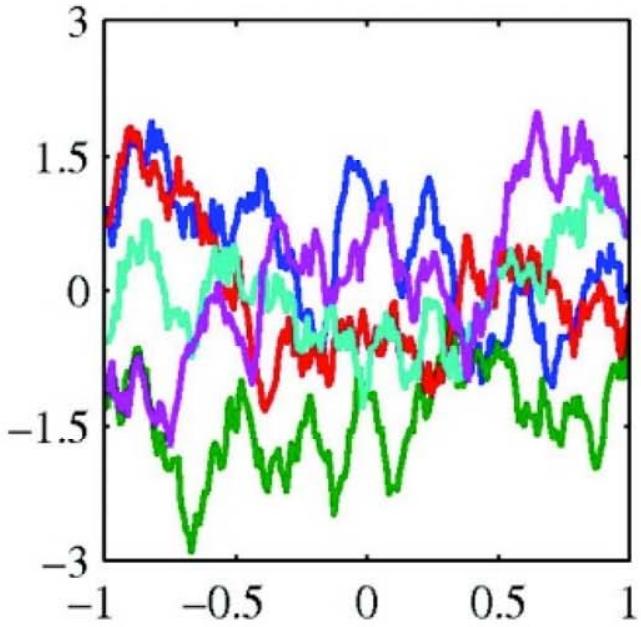
- **Prior distribution**: What *a priori* knowledge do we have about the function we want to estimate $f(\mathbf{x})$?
- **Likelihood**: What information about $f(\mathbf{x})$ do the observations provide? $\rightarrow$ Noise distribution
- **Bayes Theorem**: How to combine the prior with the likelihood to yield a **posterior distribution** for $f(\mathbf{x})$

$$\text{Posterior} \propto \text{Prior} \times \text{Likelihood}$$

- GPs: the prior distribution and the noise distribution (likelihood) are both Gaussian $\Longrightarrow$ the posterior is also Gaussian

Prior: a zero-mean Gaussian with covariance matrix **K** (kernel matrix): $f(\mathbf{x}) \sim \mathcal{GP}(0, \mathbf{K})$

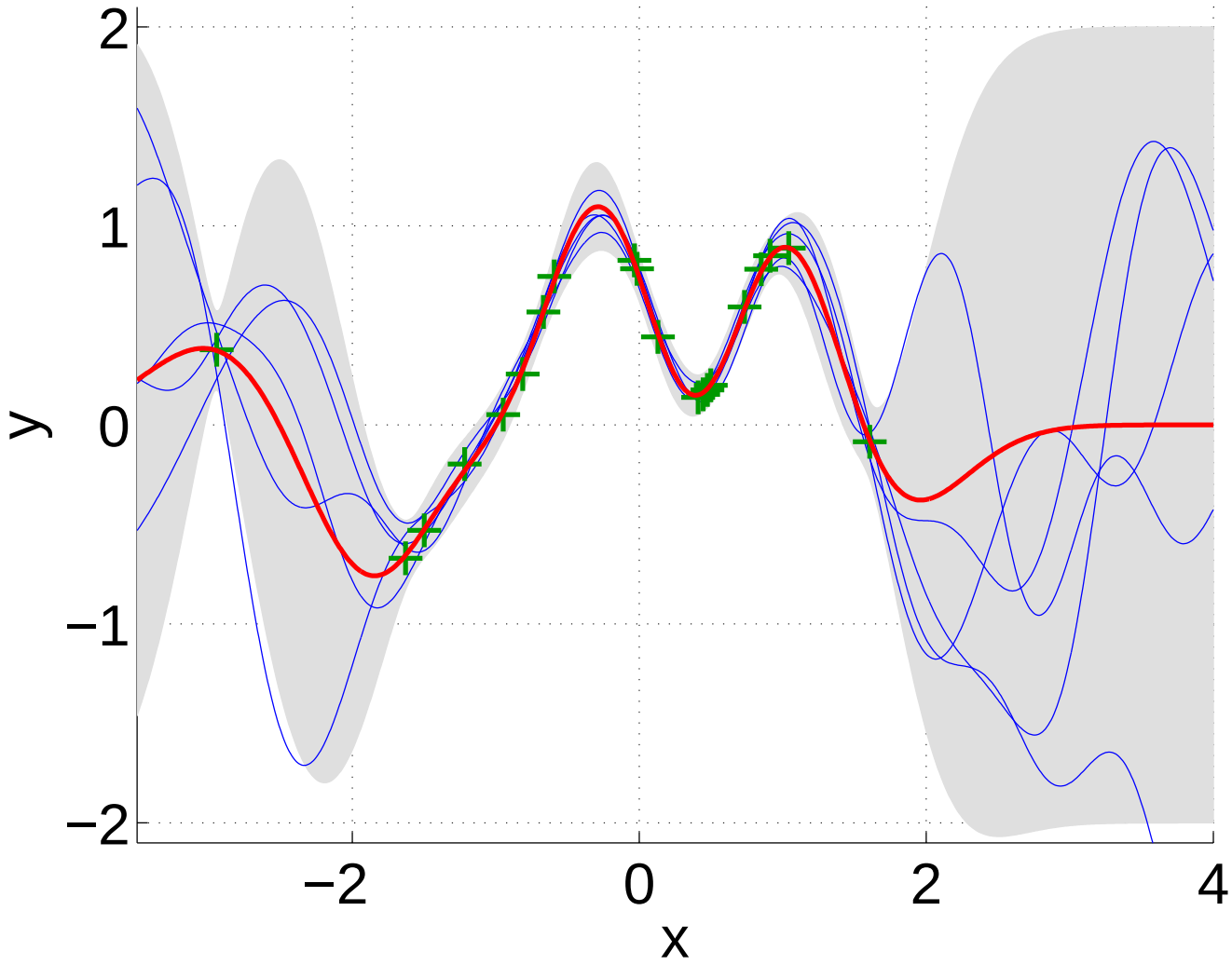




$$k(x_1, x_2) = \exp\left(\frac{-(x_1 - x_2)^2}{2\sigma^2}\right)$$

$$k(x_1, x_2) = \exp\left(\frac{-|x_1 - x_2|}{\sigma^2}\right)$$

Likelihood: a zero-mean Gaussian with variance σ_e^2:
$e_i \sim \mathcal{N}(0, \sigma_e^2)$



Posterior: Given a new test point $\mathbf{x}$, the posterior distribution for the latent function (GP output) is a Gaussian: $\mathcal{N}(f(\mathbf{x}), \sigma^2)$

- Mean

$$f(\mathbf{x}) = \mathbf{k}^T[\mathbf{K} + \sigma_e^2\mathbf{I}]^{-1}\mathbf{y},$$

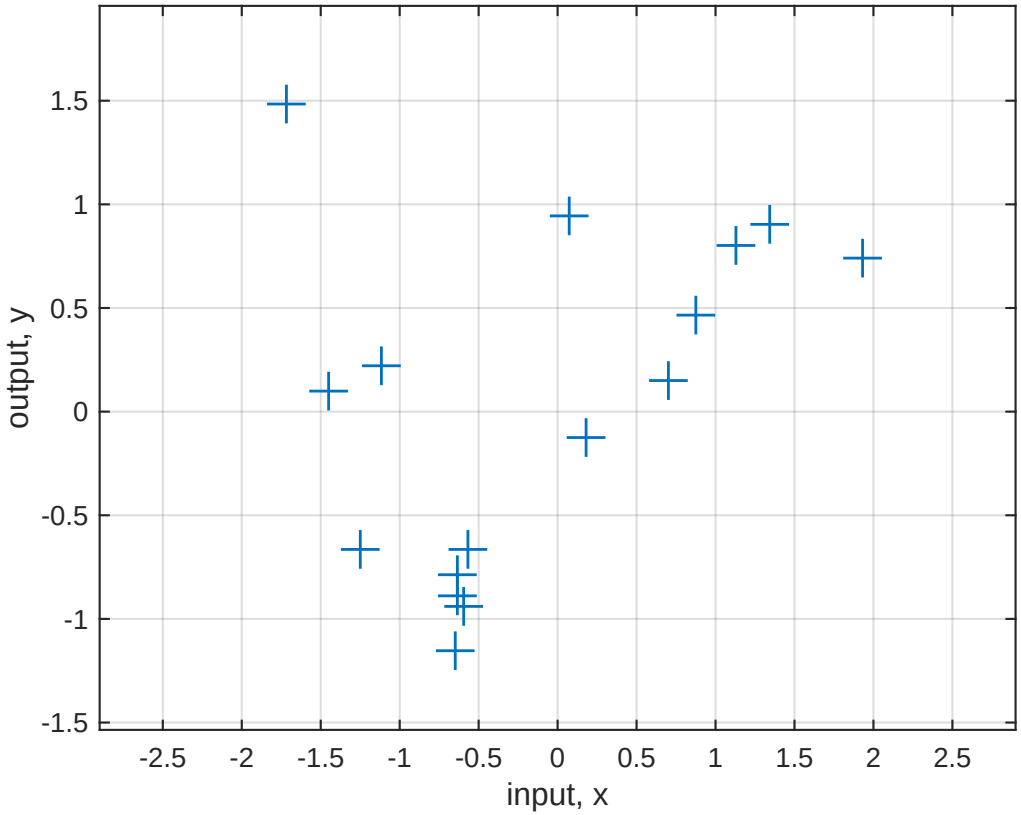
same expression as a KRR with regularization parameter $\lambda = \sigma_e^2$!!

- Variance

$$\sigma^2 = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}^T[\mathbf{K} + \sigma_e^2\mathbf{I}]^{-1}\mathbf{k}$$

where $\mathbf{k} = \left[k(\mathbf{x}_1, \mathbf{x}), \ldots, k(\mathbf{x}_n, \mathbf{x})\right]^T$, and $\mathbf{K}$ is the kernel matrix with elements $k(\mathbf{x}_i, \mathbf{x}_j)$

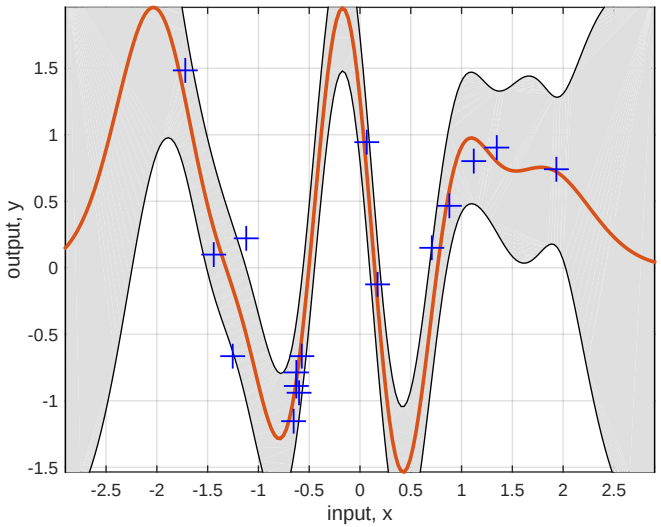
Example



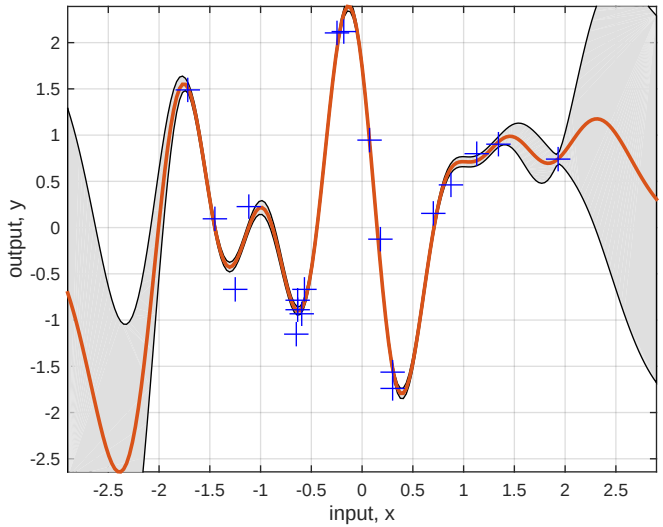
- GP with Gaussian kernel
- Hyperparameters: kernel size σ^2 and noise variance σ_e^2

Fixed kernel size ($\sigma^2 = 0.2$)

$\sigma_e^2 = 0.2$

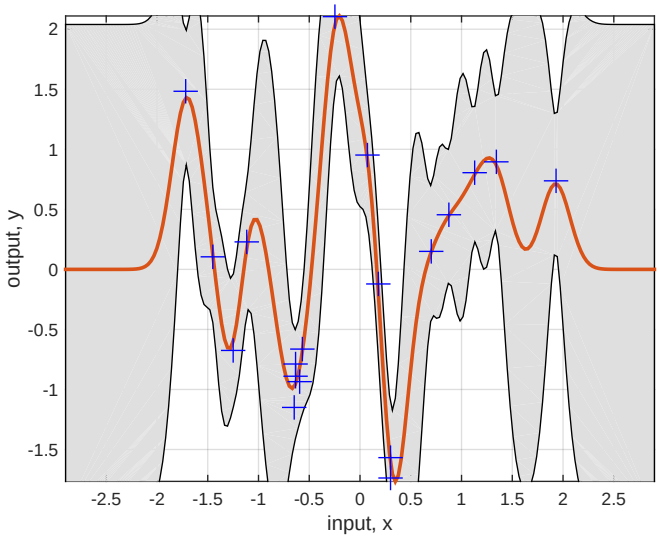


$\sigma_e^2 = 0.02$

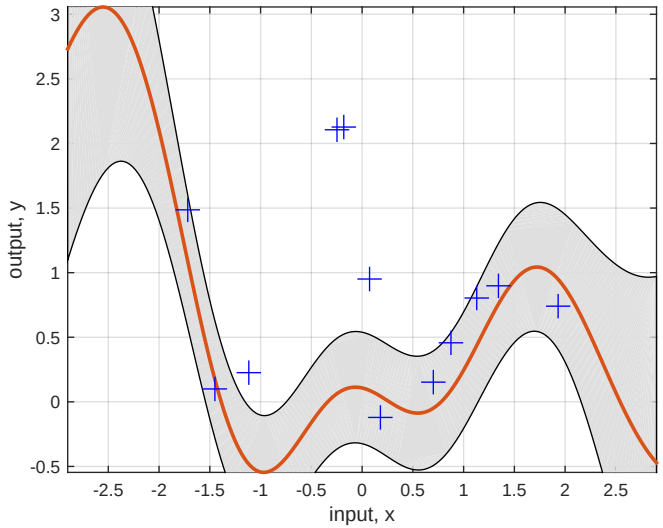


Fixed noise variance ($\sigma_e^2 = 0.2$)

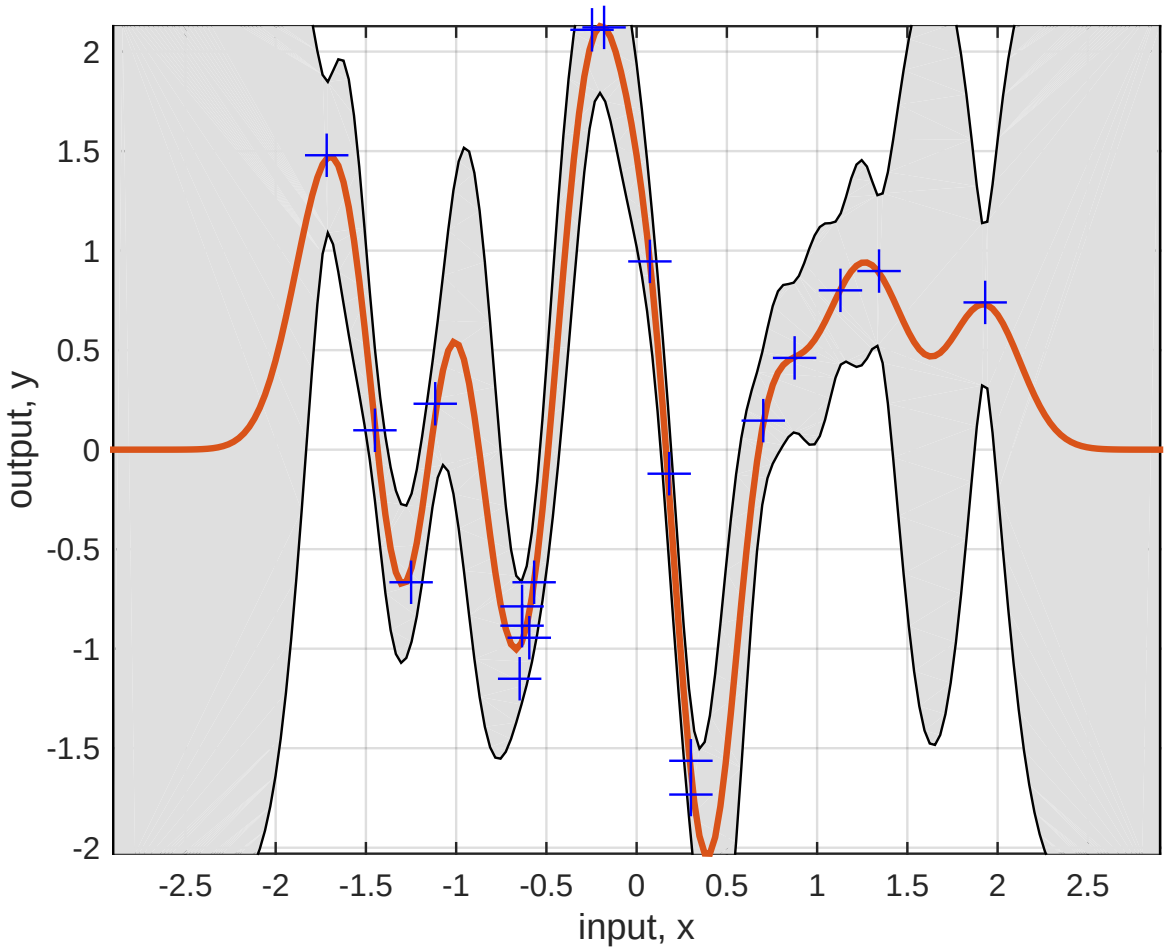
$\sigma^2 = 0.02$



$\sigma^2 = 1$



Typically we use the Maximum Likelihood estimates of the hyperparameters: $\hat{\sigma}^2 \approx 0.04$ y $\sigma_e^2 \approx 0.135$



Software

Matlab:

- GPML software package: http://www.gaussianprocess.org/gpml/code/matlab/doc/

Python:

- GPy: http://sheffieldml.github.io/GPy/
- GPs via TensorFlow: https://github.com/GPflow/GPflow

scikit-learn includes a simple version

Conclusions

- SVR
  - Based on the SRM principle
  - Loss function: ϵ-insensitive $\rightarrow$ Sparse solution
  - QP problem
  - Hyperparameter estimation: Cross-validation
- KRR
  - Regularized LS in the feature space
  - Loss function: L_2-norm $\rightarrow$ Non-sparse solution
  - Inversion of the regularized kernel matrix
  - Hyperparameter estimation: Cross-validation
- GPs
  - Bayesian approximation
  - Provides confidence intervals
  - Mean value of the posterior = KRR
  - Hyperparameter estimation: Maximum Likelihood