Advanced Machine Learning
Practical 4: Regression (SVR, RVR, GPR)

Professor: Aude Billard
Assistants: Guillaume de Chambrier,
Nadia Figueroa and Denys Lamotte

Spring Semester 2016

1 Introduction

During this week's practical we will focus on understanding and comparing the performance of the different regression methods seen in class, namely SVR and its variants (ϵ-SVR, ν-SVR and its Bayesian counterpart RVR), as well as an introduction to Bayesian Linear Regression and Gaussian Process Regression (GPR).

Similar to non-linear classification methods, the non-linear regression methods we will see in this tutorial (SVR/RVR/GPR) rely on a set of hyper-parameters (ϵ: tube sensitivity, ν: bounds, σ: kernel width for RBF) which optimize the objective function (and consequently the parameters of the regressive function). Choosing the best hyper-parameters for your dataset is not a trivial task, we will analyze their effect on the regression accuracy and how to choose an admissible range of parameters. A standard way of finding these optimal hyper-parameters is by doing a grid search, i.e. systematically evaluating each possible combination of parameters within a given range. We will do this for different datasets and discuss the difference in performance, model complexity and sensitivity to hyper-parameters.

2 ML_toolbox

ML_toolbox contains a set of matlab methods and examples for learning about machine learning methods. You can download ML_toolbobx from here: [link] and the matlab scripts for this tutorial from here [link]. The matlab scripts will make use of the toolbox.

Before proceeding make sure that all the sub-directories of the ML_toolbox including the files in **TP4** have been added to your matlab search path. This can be done as follows in the matlab command window:

```
>> addpath(genpath('path_to_ML_toolbox'))
>> addpath(genpath('path_to_TP4'))
```

To test that all is working properly you can try out some examples of the toolbox; look in the **examples** sub-directory.

3 Metrics

The metric you will use during this practical to compare the performance of each regressor will be the *Normalized Mean Square Error* ($NMSE$). If you have a vector $\hat{Y}$ consisting in n predictions and a vector Y of the observed values corresponding to these predictions, you can compute the *Mean Square Error* (MSE) of the predictor :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2 \tag{1}$$

The *Normalized Mean Square Error* ($NMSE$) is simply the MSE normalized by the variance of the observed values :

$$NMSE = \frac{MSE}{VAR(Y)} = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2}{\frac{1}{n-1}\sum_{i=1}^{n}\left(Y_i - \mu\right)^2}, \tag{2}$$

where μ is the mean of the observed values : $\mu = \frac{1}{n}\sum_{i=1}^{n} Y_i$.

4 SVR

Support Vector Machines can be applied not only to classification problems but also to regression problems. In Support Vector Regression (SVR), given a training data set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_M, y_M)\}$ of M samples, where $\mathbf{x}^i \in \mathbb{R}^N$ are multi-dimensional inputs and $y_i \in \mathbb{R}$ are continuous uni-dimensional outputs, we seek to find a continuous mapping function $f : \mathbb{R}^N \rightarrow \mathbb{R}$ that best predicts the set of training points with the function $y = f(\mathbf{x})$ (Figure 1).

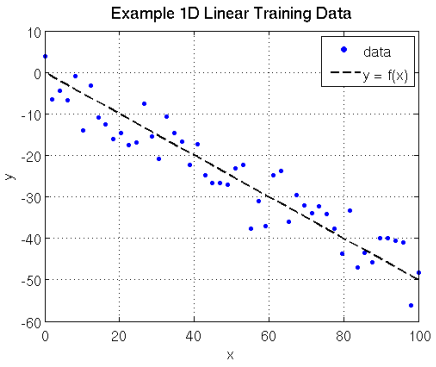


Figure 1: Regression example on 1D linear data.

Such regression problems are found in a wide spectrum of research, financial, commercial and industrial areas. For example, one might want to predict the cost of a car given its attributes, the performance of a CPU given its characteristics or the exchange rate of a currency given some relevant economic indicators. Many linear/non-linear methods exist to obtain this regressive function $f(\mathbf{x})$. What makes SVR a powerful method is that

its goal is to obtain a function $f(\mathbf{x})$ that has the most ϵ-deviation from the training outputs $\{y_1, \ldots, y_M\}$ and is as *flat*[1] as possible. Intuitively, this means that we don't mind having some errors in our regression, as long as they're within an ϵ-deviation of $f(\mathbf{x})$. This type of function is often called ϵ-intensive loss function and the allowed deviation is called ϵ-insensitive tube. As in SVM, the variants of SVR differ in the way the objective function is formulated and their hyper-parameters, we have ϵ-SVR and ν-SVR, as well as Relevant Vector Regression (RVR), which is analogous to RVM in classification and can be considered as the Bayesian counterpart of SVR.

4.1 ϵ-SVR

Following the problem formulation of SVR in the introduction of this section, in ϵ-SVR we seek to find an estimate of the true function:

$$f(\mathbf{x}) = \langle \mathbf{w}^T, \mathbf{x} \rangle + b \quad \text{with} \quad \mathbf{w} \in \mathbb{R}^N, n \in \mathbb{R} \tag{3}$$

such that, points which are contained withing the ϵ-tube are not penalized, $f(\mathbf{x}) - y \leq \epsilon$. This function can be learned by minimizing the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w},\xi,\xi^*} & \left(\frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{M} (\xi_i + \xi_i^*) \right) \\ \text{s.t.} \quad & y^i - \langle \mathbf{w}^T, \mathbf{x}^i \rangle - b \leq \epsilon + \xi_i^* \\ & \langle \mathbf{w}^T, \mathbf{x}^i \rangle + b - y^i \leq \epsilon + \xi_i \\ & \xi_i, \xi_i^* \geq 0, \forall i = 1 \ldots M, \epsilon \geq 0 \end{aligned} \tag{4}$$

where $\mathbf{w}$ is the separating hyper-plane, ξ_i, ξ_i^* are the slack variables, b is the bias, ϵ the allowable error and C is the penalty factor associated to errors larger than ϵ. By solving the dual of this objective function, we achieve a regressive function of the following form:

$$y = f(\mathbf{x}) = \sum_{i=1}^{M} (\alpha_i - \alpha_i^*) \langle \mathbf{x}, \mathbf{x}^i \rangle + b \tag{5}$$

where α_i are the Lagrangian multipliers which define the support vectors ($\alpha_i, \alpha_i^* > 0$ are support vectors). As seen in class, for non-linear datasets, instead of transforming the data to a high-dimensional feature space, we use the inner product of the feature space, this is the so-called *kernel trick*, the regressive function then becomes:

$$y = f(\mathbf{x}) = \sum_{i=1}^{M} (\alpha_i - \alpha_i^*) k(\mathbf{x}, \mathbf{x}^i) + b \tag{6}$$

where $k(\mathbf{x}, \mathbf{x}^i)$ is the kernel function. The kernel function can be any type of function that corresponds to a dot-product of the features transformed to a high-dimensional space, choices of kernel functions for SVR in our **ML_toolbox** are the following:

[1] Flatness in a regressive function can mean less sensitive to errors in measurement/random shocks/non-stationarity of the input variables. In SVR, this is encoded as maximizing the prediction error withing the ϵ-tube

Kernels

- Homogeneous Polynomial: $k(\mathbf{x}, \mathbf{x}^i) = (\langle \mathbf{x}, \mathbf{x}^i \rangle)^p$, where $p < 0$ is the hyper-parameter and corresponds to the polynomial degree.
- Inhomogeneous Polynomial: $k(\mathbf{x}, \mathbf{x}^i) = (\langle \mathbf{x}, \mathbf{x}^i + d \rangle)^p$, where $p < 0$ and $d \geq 0$, generally $d = 1$ are the hyper-parameters that correspond to the polynomial degree.
- Radial Basis Function (Gaussian): $k(\mathbf{x}, \mathbf{x}^i) = \exp\left\{-\frac{1}{2\sigma^2}||\mathbf{x} - \mathbf{x}^i||^2\right\}$, where σ is the hyper-parameter and corresponds to the width or scale of the Gaussian kernel centered at $\mathbf{x}^i$

Hyper-parameters Apart from the kernel hyper-parameters, ϵ-SVR has two open-parameters:

- C: Cost $[0 \rightarrow \text{inf}]$ represents the penalty associated with errors larger than epsilon. Increasing cost value causes closer fitting to the calibration/training data.
- ϵ: epsilon represents the minimal required precision.

Q: What is the effect of ϵ on the regressive function? When training $f(\mathbf{x})$, there is no penalty associated with points which are predicted within distance ϵ from the y^i. Small values of ϵ force closer fitting to the training data. Since ϵ controls the width of the insensitive error zone, it can directly affect the number of support vectors. By increasing ϵ, we might get fewer support vectors, but could yield more flat estimates (see Figure 2)

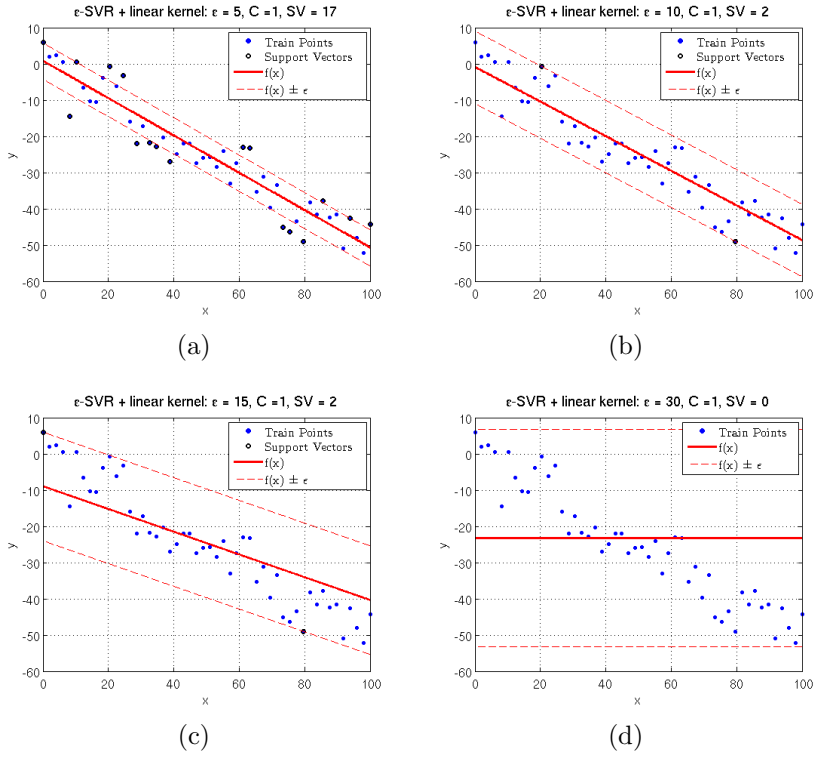
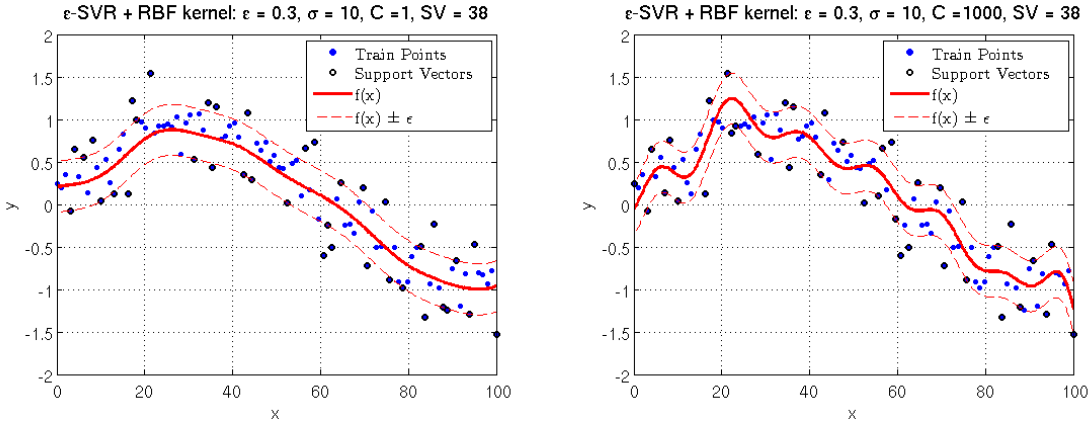


Figure 2: Effect of ϵ on regressive function

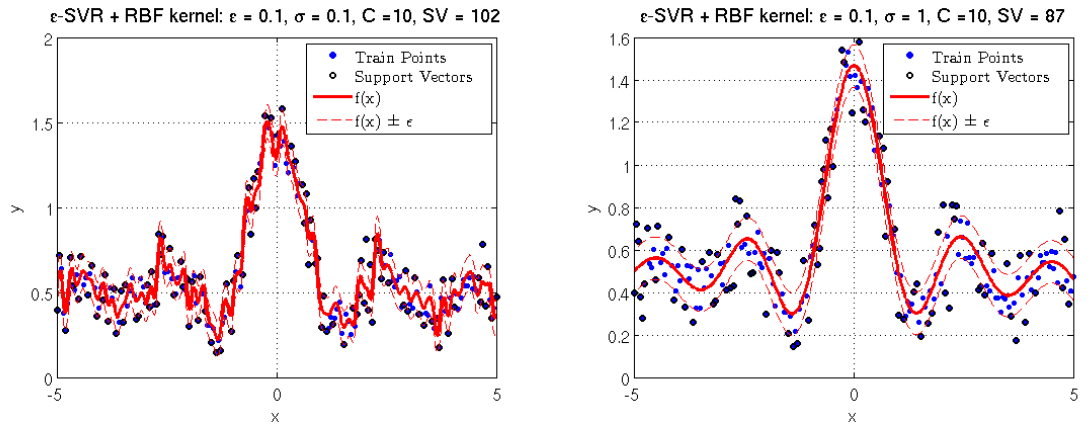
Q: What is the effect of C on the regressive function? Parameter C determines the trade off between the model complexity (flatness) and the degree to which deviations larger than ϵ are tolerated in the optimization formulation. For example, if C is too large (infinity), then the objective is to minimize the error at all costs (see Figure 3)



(a) (b)

Figure 3: Effect of C on regressive function

Q: What is the effect of the kernel hyper-parameters on the regressive function? When using the RBF kernel, we would need to find an optimal value for σ, which is the width in the squared-exponential function. Intuitively, σ is the radius of influence of the selected support vectors, if σ is very low, it will be able to encapsulate only those points in the feature space which are very close, on the other hand if σ is very big, the support vector will influence points further away from them. It can be thought of as a parameters that controls the shape of the separating hyperplane. Thus, the smaller σ the more likely to get more support vectors (for some values of C and ϵ). (See Figure 4).



(a) (b)

Figure 4: Effect of C on regressive function

You can generate these datasets and test different hyper-parameter combinations in the matlab script **TP4_SVR.m**.

4.2 ν-SVR

As it is difficult to select an appropriate ϵ, ν-SVR was introduced. Rather than controlling for the maximum allowable ϵ-deviation, this new ν parameter lets us control the number of support vectors and training errors and gives an estimate of the ϵ in the data. The original ϵ-SVR objective function now is of the following form:

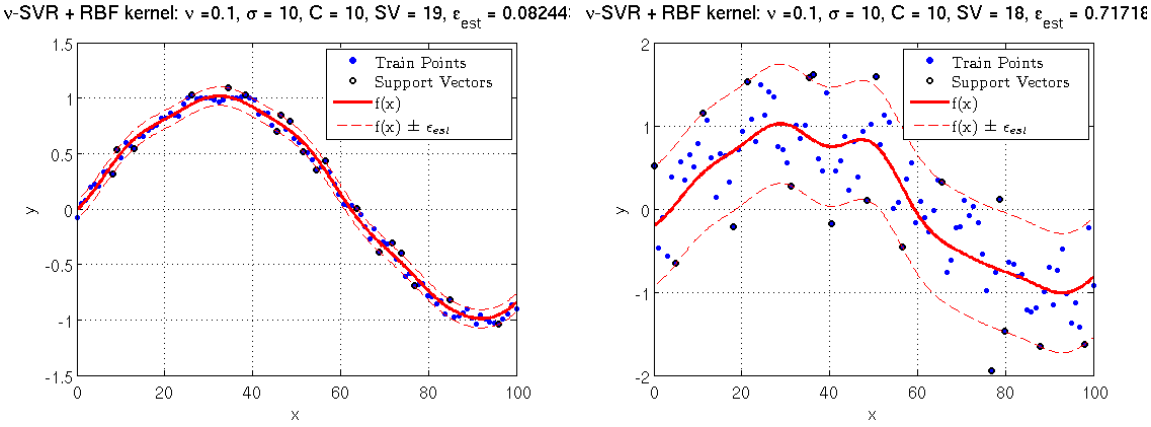
$$\begin{aligned} \min_{\mathbf{w},\xi,\xi^*} & \frac{1}{2}||\mathbf{w}||^2 + C\left(\nu\epsilon + \frac{1}{M}\sum_{i=1}^{M}(\xi_i + \xi_i^*)\right) \\ \text{s.t.} \quad & y^i - \langle \mathbf{w}^T, \mathbf{x}^i \rangle - b \leq \epsilon + \xi_i^* \\ & \langle \mathbf{w}^T, \mathbf{x}^i \rangle + b - y^i \leq \epsilon + \xi_i \\ & \xi_i, \xi_i^* \geq 0, \forall i = 1 \dots M, \epsilon \geq 0 \end{aligned} \tag{7}$$

ν represents an *upper bound* on the fraction of margin errors and a *lower bound* for the number of support vectors, which is proportional to the ratio $\nu \leq \frac{\#SV}{M}$. Adding these new variables only affects the objective function (Equation 7), the regressive function stays the same as for ϵ-SVR (Equation 6).

Hyper-parameters In ν-SVR rather then setting ϵ we set this new parameter ν, alongside C and the hyper-parameter for the choosen kernel function.

- ν: ν $(0 \rightarrow 1]$ indicates a lower bound on the number of support vectors to use, given as a fraction of total calibration samples, and an upper bound on the fraction of training samples which are errors (poorly predicted).

Q: What is the effect of ν on the regressive function? The parameter ν is used to determine the proportion of the number of support vectors you desire to keep in your solution with respect to the total number of samples in the dataset. ϵ is introduced into the optimization problem formulation and it is estimated automatically for you depending on the data at hand. Hence, given the same parameters, the regressive function will automatically adapt to the noise of the data (see Figure 5).



(a) True noise level $\epsilon = 0.05$ (b) True noise level $\epsilon = 0.5$

Figure 5: Adaptation of ϵ given different noise levels from the same signal using ν-SVR

4.3 RVR

The Relevance Vector Machine (RVM) for regression follows essentially the same formulation as for the classification case, we only modify the form of the likelihood function (output conditional distribution) to account for continuous outputs with an estimated noise variance. To recall, the predictive linear model for RVM has the following form:

$$y(\mathbf{x}) = \sum_{i=1}^{M} \alpha_i \, k(\mathbf{x}, \mathbf{x}^i) = \alpha^T \Psi(\mathbf{x}) \tag{8}$$

where $\Psi(\mathbf{x}) = [\Psi_0(\mathbf{x}), \ldots, \Psi_M(\mathbf{x}),]^T$ is a linear combination of basis functions $\Psi(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}^i)$, $\alpha = [1, 0, 1, 0, , \ldots 0, 0, 0, 1]$ is a sparse vector of weights and $y(\mathbf{x})$ is the binary classification decision function $y(\mathbf{x}) \rightarrow \{1, 0\}$. To calculate α we take a Bayesian approach and assume that all outputs y^i are i.i.d samples from a true model $y(\mathbf{x^i})$ (Equation 8) subject to some Gaussian noise with zero-mean:

$$\begin{aligned} y^i(\mathbf{x}) &= y(\mathbf{x}^i) + \epsilon \\ &= \alpha^T \Psi(\mathbf{x}^i) + \epsilon \end{aligned} \tag{9}$$

where $\epsilon \propto \mathcal{N}(0, \sigma_\epsilon^2)$. We can then estimate the probability of an output y^i given x with a Gaussian distribution with mean on $y(\mathbf{x^i})$ and variance σ_ϵ^2, this is equivalent to:

$$\begin{aligned} p(y^i|\mathbf{x}) &= \mathcal{N}(y^i|y(\mathbf{x^i}), \sigma_\epsilon^2) \\ &= (2\pi\sigma_\epsilon^2)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2\sigma_\epsilon^2}(y^i - \alpha^T\Psi(\mathbf{x}^i))^2\right\} \end{aligned} \tag{10}$$

Thanks to the independence assumption we can then estimate the likelihood of the complete dataset with the following form:

$$p(\mathbf{y}|\mathbf{x}, \sigma_\epsilon^2) = (2\pi\sigma_\epsilon^2)^{-\frac{M}{2}} \exp\left\{-\frac{1}{2\sigma_\epsilon^2}||\mathbf{y} - \alpha^T\Psi(\mathbf{x})||^2\right\} \tag{11}$$

To impose sparsity on the solution, we define an explicit prior probability distribution over α, namely a zero-mean Gaussian:

$$p(\alpha|\mathbf{a}) = \prod_{i=0}^{M} \mathcal{N}(\alpha^i|0, a_i^{-1}) \tag{12}$$

where $\mathbf{a} = [a^0, \ldots, a^M]$ is a vector of hyper-parameters a^i, each corresponding to an independent weight α^i indicating it's strength or *relevance*. Now, for the *regression problem*, following Bayesian inference, to learn the unknown hyper-parameters $\alpha, \mathbf{a}, \sigma_\epsilon^2$ we start with the decomposed posterior[2] over unknowns given the data:

$$p(\alpha, \mathbf{a}, \sigma_\epsilon^2|\mathbf{y}) = p(\alpha|\mathbf{y}, \mathbf{a}, \sigma_\epsilon^2)p(\mathbf{a}, \sigma_\epsilon^2|\mathbf{y}) \tag{13}$$

where the posterior distribution over the weights $p(\alpha|\mathbf{y}, \mathbf{a}, \sigma_\epsilon^2)$ can be computed analytically[3]. The hyper-parameter posterior $p(\alpha, \sigma_\epsilon^2|\mathbf{y})$ is unfortunately not analytically solvable. The problem then becomes the maximization of $p(\mathbf{a}, \sigma_\epsilon^2|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{a}, \sigma_\epsilon^2)p(\mathbf{a})p(\sigma_\epsilon^2)$.

[2]Refer to the Tipping paper on RVM to understand how this and subsequent terms were derived.

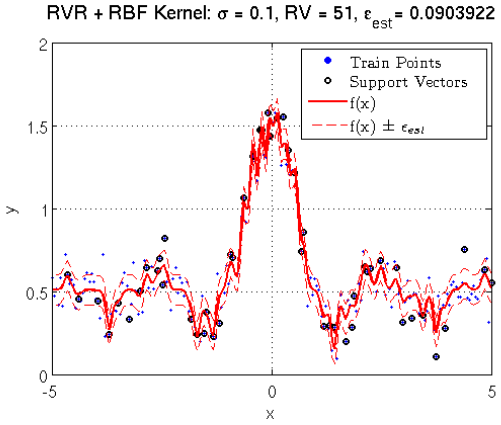
[3]This comes from the posterior over all unknowns $p(\alpha, \mathbf{a}, \sigma^2|\mathbf{y}) = \frac{p(\mathbf{y}|\alpha, \mathbf{a}, \sigma^2)p(\alpha, \mathbf{a}, \sigma^2)}{p(\mathbf{y})}$.2

Assuming uniform priors $p(a), p(\sigma_\epsilon^2)$, the problem reduces to maximizing the term $p(\mathbf{y}|\mathbf{a}, \sigma_\epsilon^2)$, given by:

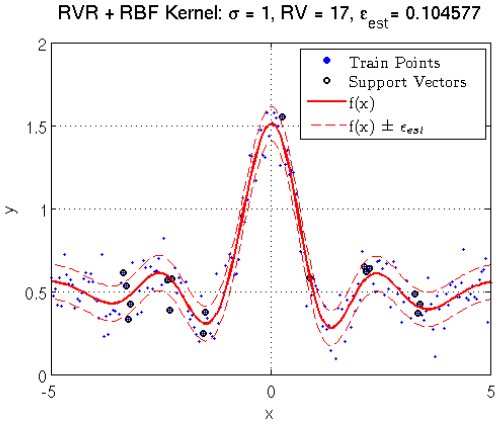
$$\begin{aligned} p(\mathbf{y}|\mathbf{a}, \sigma_\epsilon^2) &= \int p(\mathbf{y}|\mathbf{w}, \sigma_\epsilon^2) p(\mathbf{w}|\alpha) d\mathbf{w} \\ &= (2\pi)^{-\frac{M}{2}} |\sigma_\epsilon^2 \mathbf{I} + \mathbf{\Psi} \mathbf{A}^{-1} \mathbf{\Psi}^T|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2} \mathbf{y}^T (\sigma_\epsilon^2 \mathbf{I} + \mathbf{\Psi} \mathbf{A}^{-1} \mathbf{\Psi}^T) \mathbf{y} \right\}. \end{aligned} \tag{14}$$

where $\mathbf{A} = diag(\alpha_0, \ldots, \alpha_M)$. By solving this optimization problem[4] we find our weight vector α, consequently the optimal 'relevant vectors', and the estimate of the noise variance σ_ϵ^2.

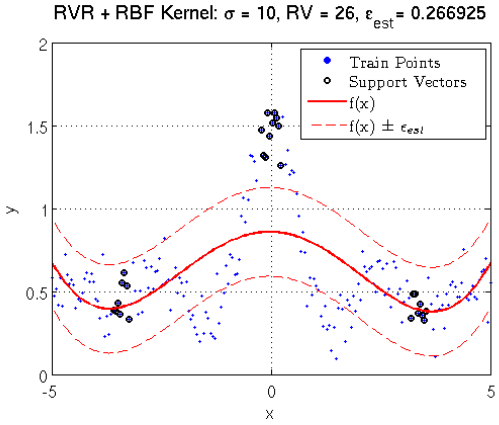
Q: What is the effect of the kernel hyper-parameters on the regressive function? Even though the RVR formulation is set to optimize the number of relevant vectors as well as the ϵ-deviation, if we set a kernel width (for example for RBF) too low/too high it will over-fit/under-fit the regressive function, and give a high number of relevant vector regardless of sparsity constraints (see Figure 6)



(a) Too small.



(b) Adequate σ



(c) Too big.

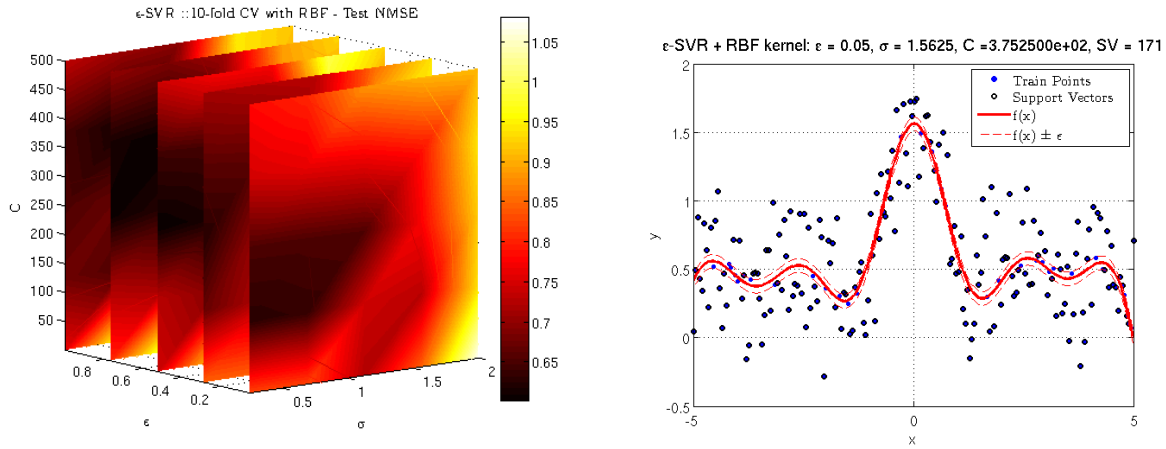
Figure 6: Effect of σ for RBF kernel in RVR

[4]Refer to Tipping paper for hyper-parameter optimization methods.

4.4 SVR-RVR Evaluation

Q: How to find the optimal parameter choices? As in classification, we can use grid search on the open parameters. We must choose admissible ranges for these, in the case of ϵ-SVR with RBF kernel, it would be for C, ϵ and σ. Additionally, in order to get statistically relevant results one has to cross-validate with different test/train ratio. A standard way of doing this is applying 10-fold Cross Validation for each of the parameter combinations and keeping some statistics such as mean and std. deviation of the regression metrics for the 10 runs of that specific combination of parameters

Grid Search with 10-fold Cross Validation For the sinc dataset we can run ϵ-SVR (with RBF kernel) with 10-fold CV over the following range of parameters $C\_\text{range} = [1 : 500]$, $\epsilon\_\text{range} = [0.05 : 1]$,and $\sigma\_\text{range} = [0.25 : 2]$, in the matlab script **TP4_SVR.m** you can specify the steps to partition the range within these limits. By selecting the combination of hyper-parameter values that yields the lowest metric on the test dataset we can achieve an approximate regressive function as the one seen in Figure 7



(a) Heatmaps for 3-parameter grid-search

(b) Best fitted function with ϵ-SVR.

Figure 7: Grid Search result for 1D non-linear on ϵ-SVR.

Q: How did we choose these ranges? Any ideas?
By running this part of the script you should get the plot in Figure 7. This shows a heatmap for the mean Train (left) and Test (right) NMSE (i.e. normalized mean squred error).

Which method should we use? The decision regarding which method to choose depends completely on the type of model you want. If you prefer an efficient solution in terms of model complexity (i.e. less support vectors), without having such a precise estimate, either ν-SVR or RVR would be the way to go. However, if you wish to control the amount of error in the model and have the best performance possible, regardless of model complexity, then ϵ-SVR would be the best choice.

5 Bayesian Linear Regression

In Bayesian linear regression we seek to find the parameters of a linear model. Being more specific we are interested in finding the weights $\mathbf{w}$ of Equation 15,

$$\begin{aligned} y &= f(x) + \epsilon \\ &= \mathbf{w}^{\mathrm{T}} x + \epsilon \end{aligned} \tag{15}$$

where $y \in \mathbb{R}$ and $\mathbf{w}, x \in \mathbb{R}^N$ and there is an assumption of additive Gaussian noise, $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Given a data set consisting of output-input pairs:

$$(x^{(1)}, y^{(1)}), \ldots, (x^{(i)}, y^{(i)}), \ldots (x^{(M)}, y^{(M)}) = (X, \mathbf{y})$$

To fit the model to the data above a loss function has to be minimised; typically the least square error: $||\,\mathbf{y} - \mathbf{w}^{\mathrm{T}} X||$. Minimising the least square error results in a regressor function known as Ordinary Least Squares (OLS), see Equation 16.

$$\mathbf{w}_{\mathrm{OLS}} = (\mathrm{X}^{\mathrm{T}} X)^{-1} \mathrm{X}^{\mathrm{T}} \mathbf{y} \tag{16}$$

This OLS solution would be identical to the solution of maximised the log-likelihood of the observed data $\mathbf{y}$.

$$p(\mathbf{y}\,|X, \mathbf{w}; \sigma^2) = \mathcal{N}(\mathbf{y} - \mathbf{w}^{\mathrm{T}} X; I\sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}||\,\mathbf{y} - \mathbf{w}^{\mathrm{T}} X||\right) \tag{17}$$

$$\nabla_{\mathbf{w}} \log p(\mathbf{y}\,|X, \mathbf{w}) = -\frac{1}{\sigma^2} \mathrm{X}^{\mathrm{T}}(\mathbf{y} - \mathbf{w}^{\mathrm{T}} X) \tag{18}$$

$$\mathbf{w}_{\mathrm{ML}} = (\mathrm{X}^{\mathrm{T}} X)^{-1} \mathrm{X}^{\mathrm{T}} \mathbf{y} \tag{19}$$

This type of solution is known as a **Maximum Likelihood (ML)**. Bayesian Linear regression (BLR) make use of the fact that the likelihood function is Gaussian. It adds a prior Gaussian distribution over the parameters, $\mathbf{w}$, and then computes the **Maximum A Posterori (MAP)** of the Gaussian posterior distribution of the parameters, $\mathbf{w}$.

$$\overbrace{p(\mathbf{w}\,|X, \mathbf{y})}^{posterior} \propto \overbrace{p(\mathbf{y}\,|X, \mathbf{w})}^{likelihood} \overbrace{p(\mathbf{w})}^{prior} \tag{20}$$

Where $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \Sigma_p)$ is the prior distribution and $p(\mathbf{w}\,|X, \mathbf{y})$ is the posterior distribution. Finding the parameters $\mathbf{w}_{\mathrm{MAP}}$ of BLR consists of maximising the posterior distribution in the same way we maximised the log-likelihood for OLS. The result is

$$\begin{aligned} \mathbf{w}_{\mathrm{MAP}} &= \frac{1}{\sigma^2} A^{-1} X\,\mathbf{y} \\ A &= \sigma^{-2} X\,\mathrm{X}^{\mathrm{T}} + \Sigma_p^{-1} \end{aligned} \tag{21}$$

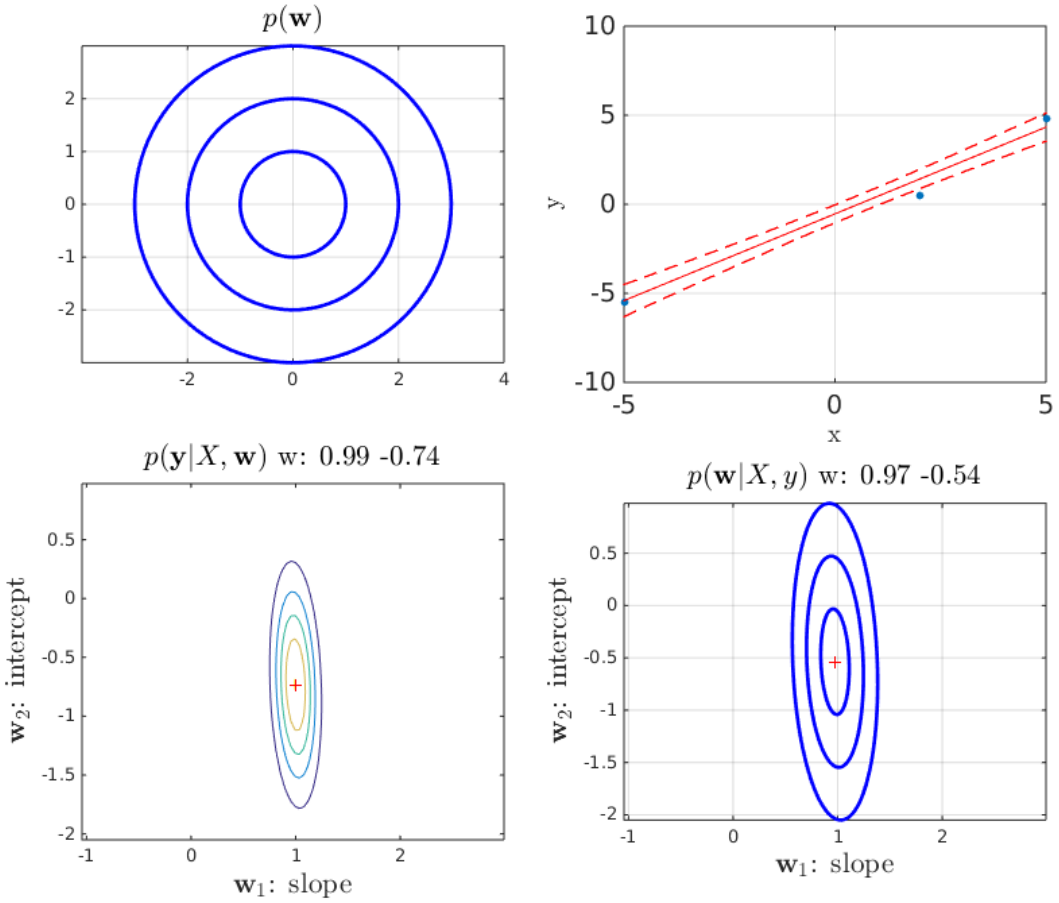


Figure 8: Steps in Bayesian Linear Regression.

5.0.1 Parameters of BLR

There are two hyper-parameters for Bayesian Linear Regression:

- σ^2: variance of the measurement noise, $p(\mathbf{y}\,|X, \mathbf{w}; \sigma^2)$
- Σ_p: prior uncertainty on the parameters of $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \Sigma_p)$.

See Figure 8 for the relation between the prior, likelihood and posterior distributions in BLR. For a more detailed explanation on BLR you can consult Chapter 2 from William's Gaussian Process book.

5.1 BLR Questions

What is the effect of the hyper-parameters ? The solution to Bayesian Linear Regression will depend a lot on the prior information you give it. In this case the prior on the weights $\mathbf{w} = [0, 0]^{\mathrm{T}}$ of the linear function is zero. This means that before seeing any data our best guess of what the regressor function should look like is a threshold line

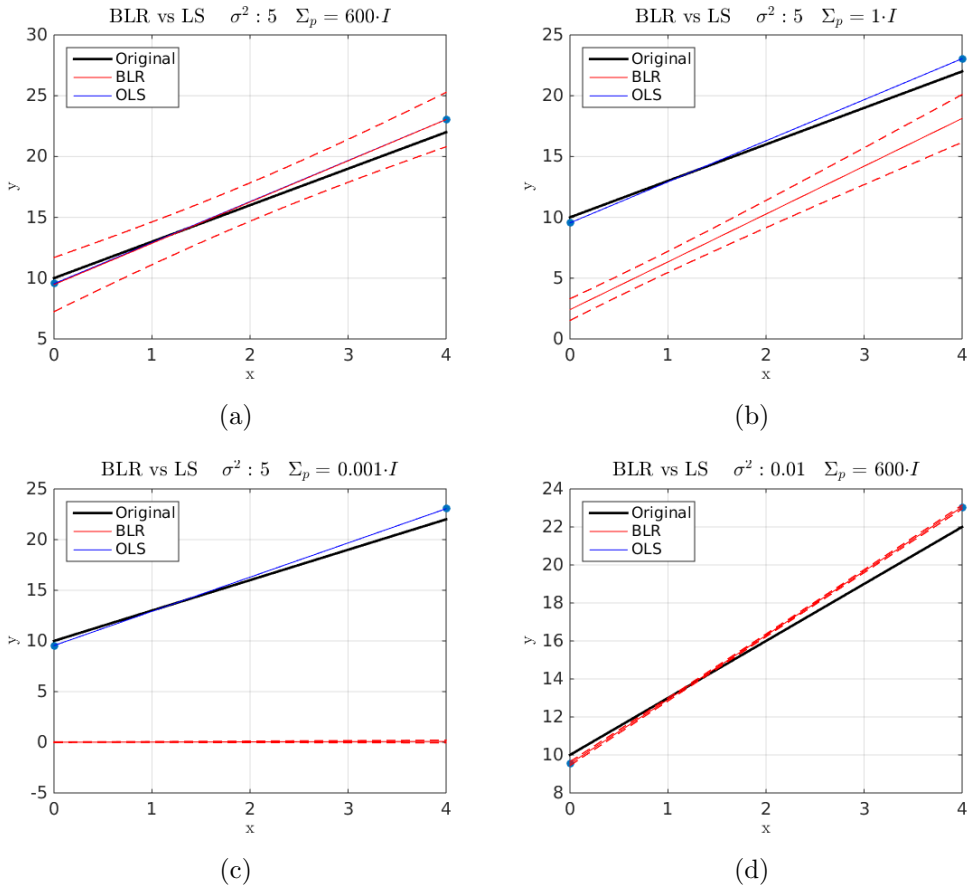


Figure 9: **Effect of Σ_p**. (a) The width of the uncertainty tube (dashed lines) is equivalent to a variance of 5. The extreme high value of Σ_p means that that we completely ignore the effect of the prior information and the solution is the same as OLS. (b) $\Sigma_p = I$ is identity, as a result the prior information is taken into consideration and we can see as a result that the offset $\mathbf{w}_2$ has shifted by down to zero. (c) $\Sigma_p = 0.0001$; we are extremely confident in our prior which totally overshadows the likelihood function. (d) illustration of the effect of σ^2, compared with (a).

with a zero bias, which is not very informative. Then there is the variance Σ_p associated with the prior guess on the value of $\mathbf{w}$, which is a way of encoding how much confidence we have in our prior $\mathbf{w}$. In Figure 9 we illustrate the effect Σ_p has on the BLR solution, $\mathbf{w}_{\text{MAP}}$.

To get a sense of how the prior parameters adapt as a function of the number of points we can look at the rate of converge of the slop and intercept of $\mathbf{w}$ for both the OLS and BLR solutions. In Figure 10 we illustrate the convergence rates.

6 Gaussian Process Regression

We know look at the kernel version of Bayesian Linear Regression, known as Gaussian Process Regression (GPR), which is for non-linear regression. As for BLR the GP model

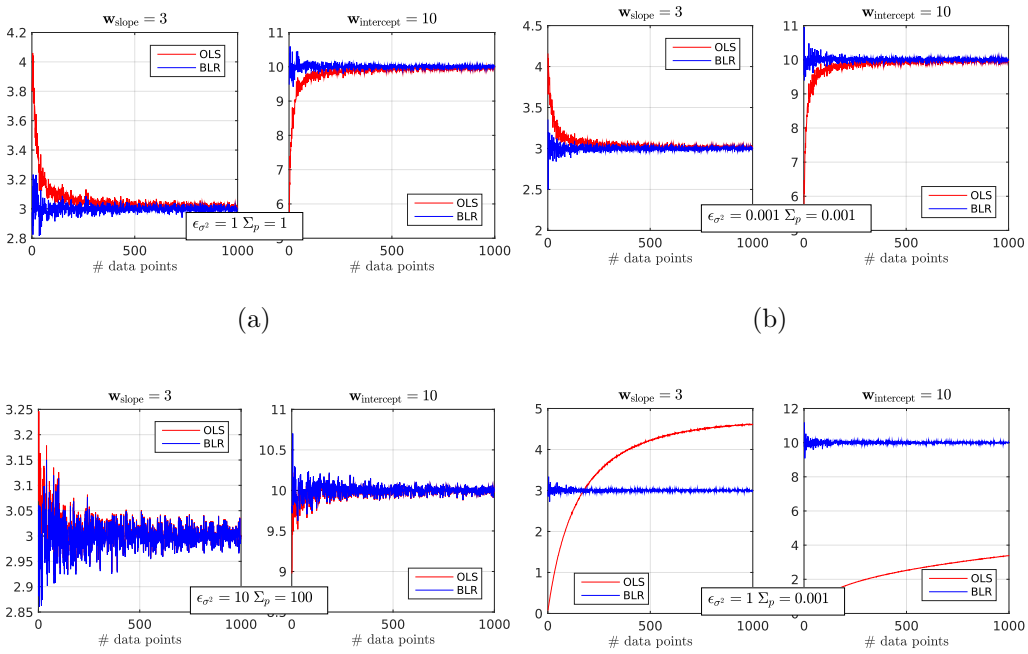


Figure 10: (a) Average noise an prior uncertainty; slow convergence to the true $\mathbf{w}$. (b) Very small uncertainty, both in noise and prior; the convergence rate is nearly the same as (b). (c) Uncertainty in measurement is 10 times smaller than the weight prior. Because of this actual data values are taken more into consideration and the both BLR and OLS converge to the same solution. (d) 100 times less uncertainty in prior on weights as oppose to the uncertainty set for observations. As a result many data points will be necessary to overcome the overly confident prior $p(\mathbf{w})$.

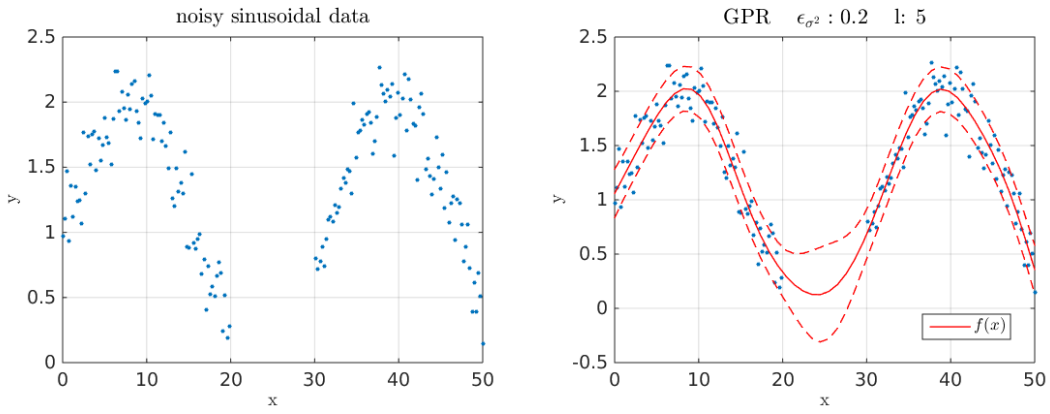


Figure 11: Sinusoidal data with a hole.

assumes additive Gaussian noise, ϵ.

$$y = f(x) + \epsilon \quad \text{where } \epsilon \sim \mathcal{N}(0, I\epsilon_{\sigma^2}) \tag{22}$$

The Gaussian process regression function $f(x)$ is a linear combination of weighed kernel functions evaluated on test points:

$$y = \sum_{i=1}^{M} \alpha_i \, k(x, x^i) \tag{23}$$

$$\boldsymbol{\alpha} = [K(X, X) + \epsilon_{\sigma^2} I]^{-1} \, \mathbf{y} \tag{24}$$

$$k(x, x') = \exp\left(-\frac{||x - x'||}{l}\right) \tag{25}$$

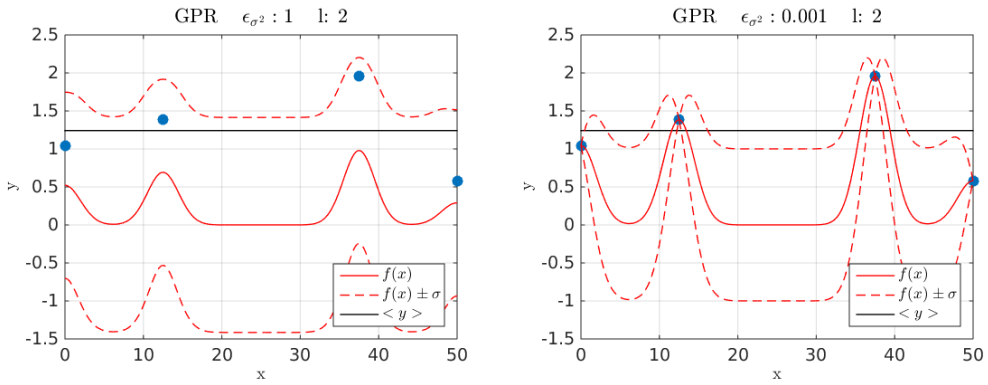
In this tutorial we will be solely considering the radial basis function, in total there are two parameters to vary:

- ϵ_{σ^2} : variance of the signal noise.
- l : variance of the kernel, also known as kernel width.

The goal of this section of the tutorial is to understand the effect both hyper-parameters have on the output of the regressor function. For this section of the tutorial you with the matlab script **TP4_GPR.m**. First generate and plot the data you will be working with, you should see the following 11:

6.1 GPR Questions

What is the impact of ϵ on the regressor function ? Intuitively the ϵ_{σ^2} can be tough in terms of how much trust we place in our training data. If you set the value of ϵ_{σ^2} high it means that your training values y are possibly very corrupted and do not accurately reflect the underlying function. If insteadd your set ϵ_{σ^2} to be a small value, then the values y accurately reflect the underlying function you are trying to regress. In



(a) (b)

Figure 12: (a) ϵ_{σ^2} is high; the value of the regressor function $f(x)$ does not go all the way to the blue data points. (b) ϵ_{σ^2} is low; all training data points are considered to be noiseless values of the underlying function, so the regressor predicts exactly the training point values.

Figure 12 we illustrate the effect this has on the output of your learned GP regressor function.

In Figure 13 we illustrate the effect ϵ_{σ^2} has on the regressors prediction.

What is the impact of the kernel variance on the regressor function ? As the variance of the kernel function goes to zero, the kernel function $k(x', x') = 1$ will only be equal to one on when a test point x' is equal to training point x and it will be zero otherwise $k(x, x) = 0$. As a result the regressor function will only return a non-zero value when a test point is within a very small distance (determined by l) from a a train point.

When the variance of the kernel tends to infinity, the Gram matrix will be equal to one, $K(X, X) = \mathbf{1}$, the regressor about will will end to the mean value of the data.

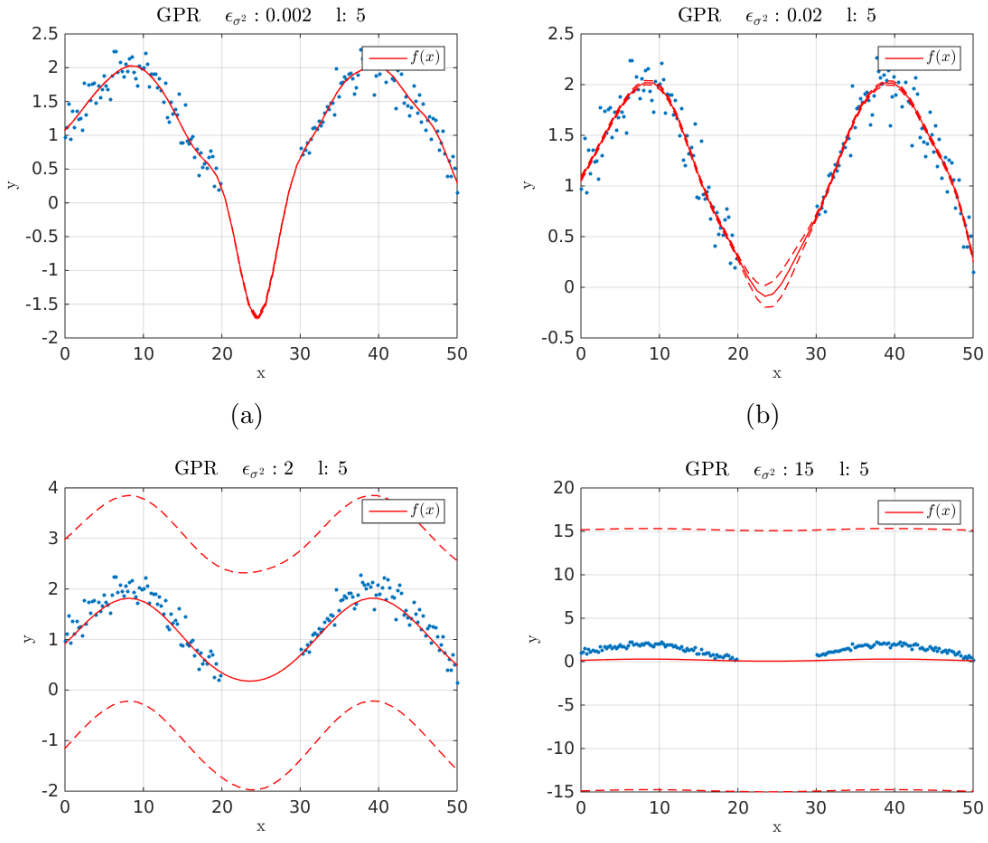


Figure 13: **Effect of noise variance on regressor**. The kernel width is kept constant at $l = 5$ and the noise parameter ϵ is varied from 0.002 to 15. The shape of the regression line seems to be slightly effected. As the noise level increases to infinity the value of the regressor function tends to zero.

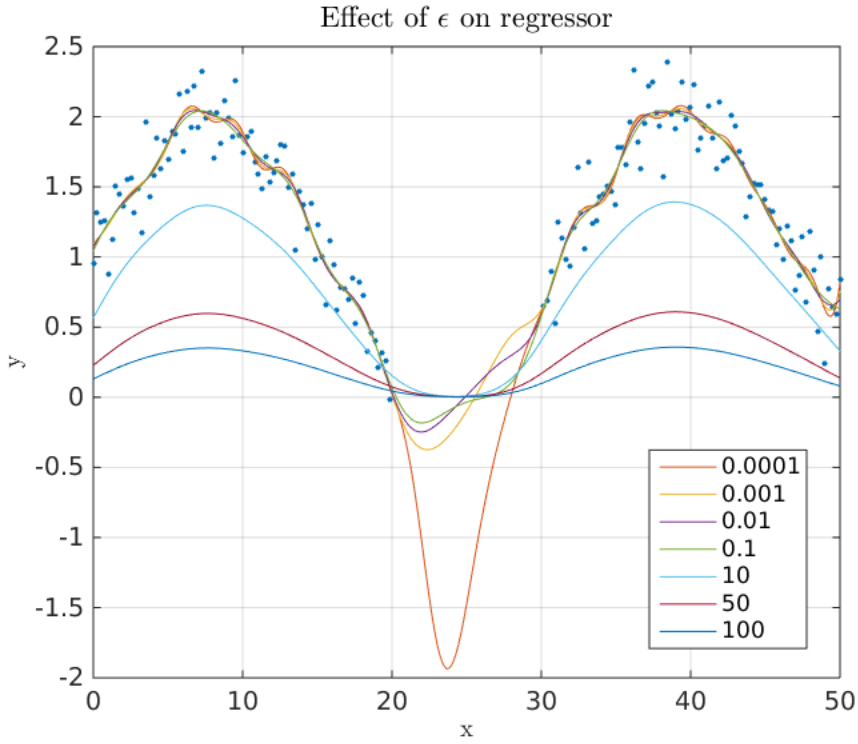


Figure 14: **Effect of noise variance on regressor #2**. A more systematic evaluation of the effect of noise on the regressor.

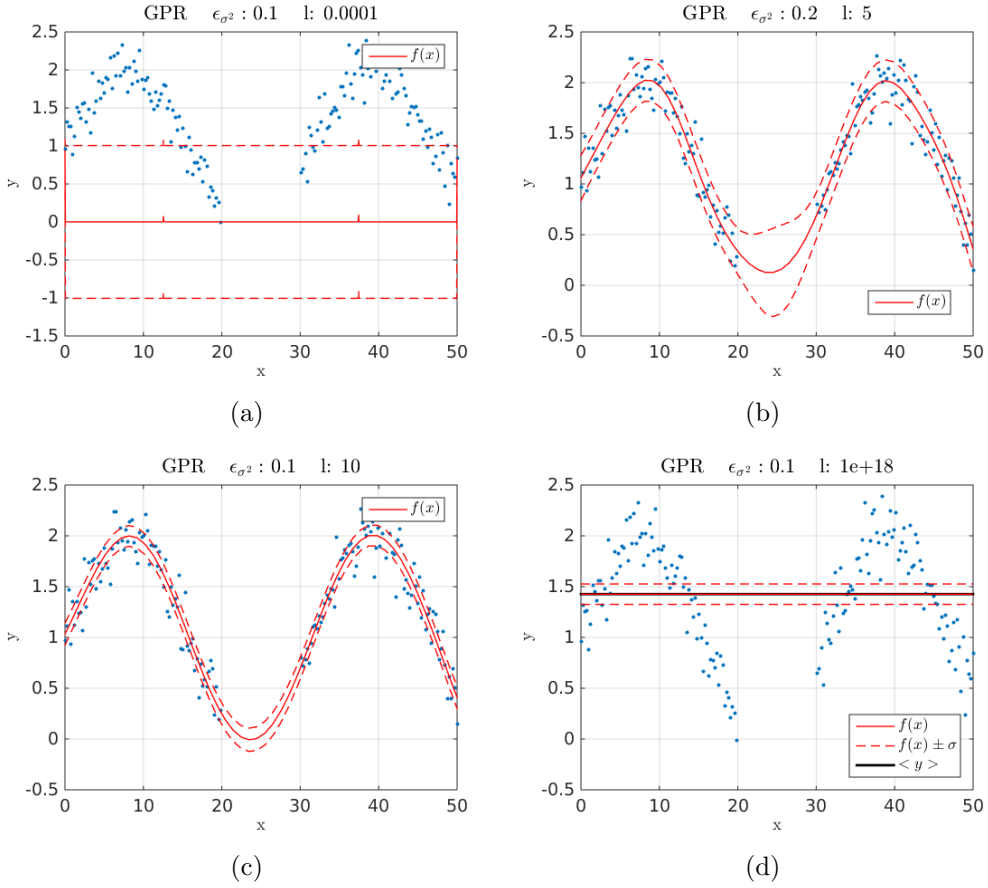


Figure 15: **Effect of kernel variance on regressor**. (a) Very small variances causes the regressor function to be zero every where except on test points. (b)-(c) variance is within the scale of the input space, which results in normal behaviour. (c) the variance, kernel width, is extremely large. We can see that the regressor value lies exactly on top of the mean of the output data, y.

7 Comparison of SVR vs. GPR

In this section you will compare the results of SVR and GPR using different metrics. For this, you will use two real-worl datasets related to wines from the north of Portugal. The goal is to grade wine quality based on physicochemical tests. Each dataset is composed of 11 real input variables and one output variable the wine quality (between 0 and 10). A more detail description of these dataset is available here `http://archive.ics.uci.edu/ml/datasets/Wine+Quality`.

Open **TP4_SVR_vs_GPR.m** to start.

Goals : Try to see for each regression technique and each dataset :

- Which one seems to yield the best precision ?
- Which one is the easiest to setup ?
- Do they always converge to the same solution ?
- Is there any difference if you compare the performance with different metrics ?
- Is the performance changing if you remove a certain percentage of data ?
- Is the performance changing if you use only some features (attributes) ? **Feature Selection can be a good way to reduce the error. You can either select a subset of attributes or perform some dimensionality reduction technique we have seen in the second pratical**.

You first load the dataset and plot it. Then you also normalized it to have a comparable influence of each attribute. Then you perform **GPR** and **SVR** (ϵ-SVR or ν-SVR) for which you can perform a grid search with 10-fold cross-validation to find the best hyper-parameters or simply enter the parameters you want. Choose a good range for the grid search if you use it.
Finally, you can compare the performance using one of the proposed metrics (MSE or $NMSE$).

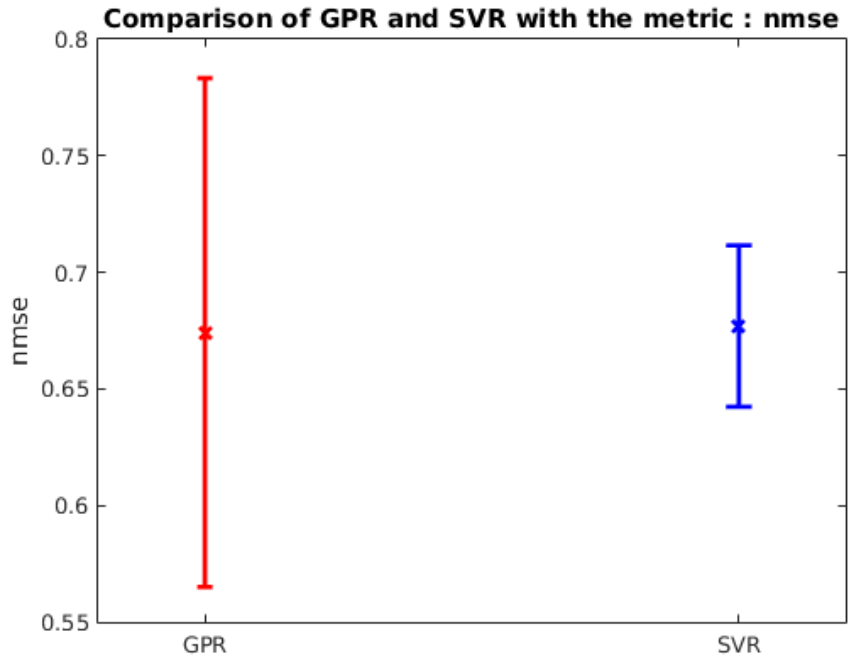


Figure 16: **Example of result with manual parameter selection**.