Master of bioinformatics

Support Vector Machines

Assignment 2: Function estimation and Time-series prediction

Spring 2016

Supervisors:
Dr. Carlos Alaiz
Dr. Emanuele Frandi
Prof. Johan Suykens

Author:
Cedric Lood

May 19, 2016

Contents

Context

The analysis presented in this report was produced for the class of “Support Vector Machines: methods and applications” (Spring 2016) at KU Leuven. The goal is to display understanding of the techniques and of their practical use. This second report focuses on function estimation and time series prediction using SVM, and Least-Squares SVM (LS-SVM). The implementation was done using the MatLab software (v2015a) and the libraries for LS-SVM developed at KU Leuven.

1 Support Vector Machine for regression

1.1 Datasets

I experimented with 3 noiseless datasets of 20 observations for this section, based on the true underlying function shown in figure 1. I used the “load dataset” functionality of the *uiregress* function to load them and perform the regress:

```
X = (linspace(-5, 5, 20))'; Y = (2.*X + 1)';
save('lin.mat');

X = (linspace(-5,5,20))'; Y = (X.^3 + X.^2 - 1)';
save('poly.mat');

X = (linspace(-pi,pi,20))'; Y = (exp(-X.^2).*sin(10.*X))';
save('sinc.mat');
```

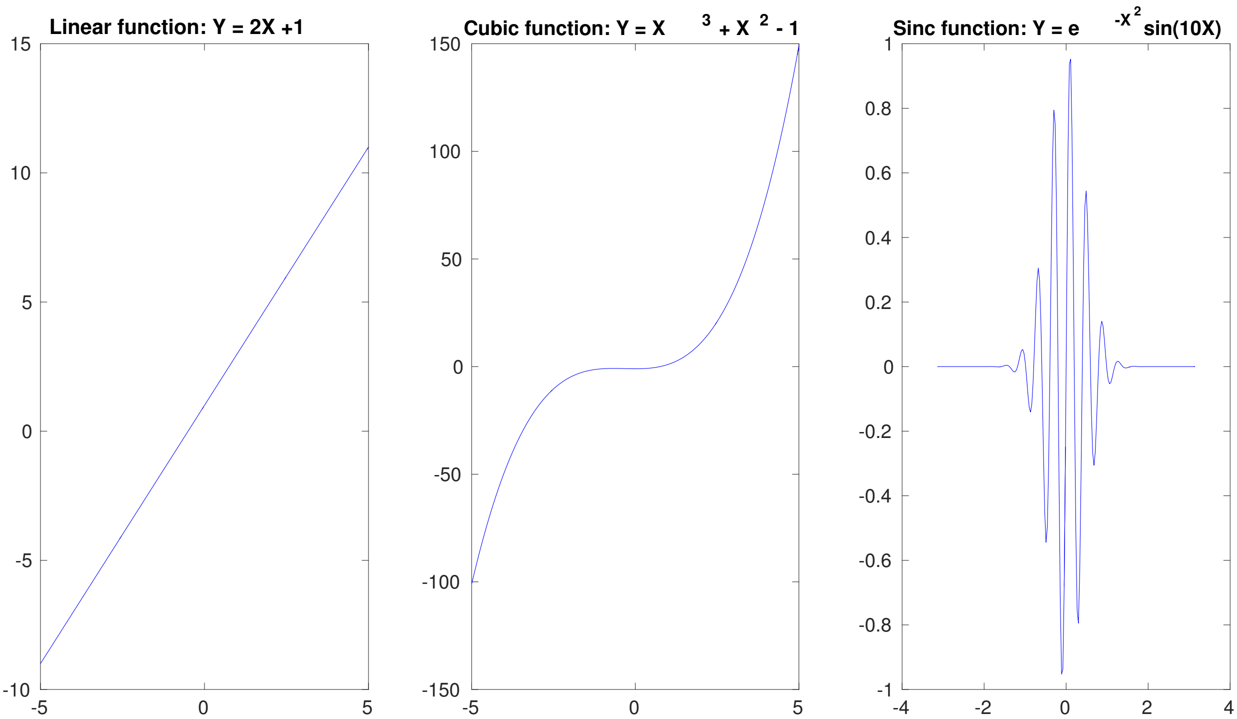

Figure 1: Functions to regress

1.2 Analysis

For the linear and cubic functions, I got good results with polynomial kernel of degree 1 and 3 respectively.

I did an extensive survey of the parameters for the first linear model. On that particular model, the linear kernel seem to have performed the best. Even for extremely low ϵ (eg $\epsilon = 0.005$), the number of support vectors needed to describe the classifier is equal to two. Another obvious advantage of that kernel was that no wiggle artefact appeared, in contrast with the other more flexible kernels.

	e value (Bound Inf)					Bound (e value 0.1)				
	0.005	0.01	0.05	0.1	0.5	0.01	0.1	1	10	100
Poly 1	2	2	2	2	2	20 **	20 **	13	2	100
Poly 2	3	6	6	6	3	20 **	20 **	8	3	3
Poly 3	7	7	4	4	2 *	20 **	20 **	15 *	4 *	4 *
Poly 4	5	5	5	5	4 *	20 **	20 **	14 *	5 *	5 *
RBF $\sigma^2 = 1$	16	7	7	4	4 *	20 **	20 **	20 **	15 **	4
RBF $\sigma^2 = 0.5$	10	9	9	7	5 *	20 **	20 **	20 **	10 **	7 *

Table 1: Number of support vectors required for different kernels and parameters (: wiggle artefact visible, **: function estimation completely off)*

2 Sum of Cosines

3 Hyper-parameter Tuning

4 Application of the Bayesian framework

5 Robust regression

6 Applications

References