**Report 2 support vector machines**

Willem Melis

[1 Support vector machines for regression 3](#_Toc454318358)

[1.1 Simple dataset 3](#_Toc454318359)

[1.1.1 Linear kernel 3](#_Toc454318360)

[1.1.2 RBF kernel 3](#_Toc454318361)

[1.1.3 Bad non-linear case 4](#_Toc454318362)

[1.1.4 Sparsity 5](#_Toc454318363)

[2 A simple example: Sum of cosines 7](#_Toc454318364)

[3 Hyper parameter tuning 9](#_Toc454318365)

[3.1 Gridsearch 9](#_Toc454318366)

[3.2 Simplex 9](#_Toc454318367)

[3.3 Csa and ds 9](#_Toc454318368)

[3.4 Optimilisation with ds 9](#_Toc454318369)

[3.5 Optimilisation with csa 9](#_Toc454318370)

[4 Application of the Bayensian framework 10](#_Toc454318371)

[4.1 Toy problem 10](#_Toc454318372)

[4.2 Iris 10](#_Toc454318373)

[4.2.1 Influence of gamma 11](#_Toc454318374)

[4.2.2 Influence of sigma 12](#_Toc454318375)

[4.2.3 Automatic relevance determination (ARD) 12](#_Toc454318376)

[5 Robust regression 13](#_Toc454318377)

[5.1 Basic 13](#_Toc454318378)

[5.2 Advanced 14](#_Toc454318379)

[5.2.1 mae vs mse 14](#_Toc454318380)

[6 Homework problem: intro Time-series prediction 16](#_Toc454318381)

[6.1 Changing order 17](#_Toc454318382)

[7 Homework problem: Santa Fe Laser dataset 18](#_Toc454318383)

[7.1 Dataset 18](#_Toc454318384)

[7.2 Questions excercise 19](#_Toc454318385)

[7.2.1 Window size 19](#_Toc454318386)

[7.2.2 Recurrent prediction 19](#_Toc454318387)

[7.3 SVM predictor 19](#_Toc454318388)

[7.3.1 best Solution 19](#_Toc454318389)

[7.3.2 Using Bayesian 20](#_Toc454318390)

[7.4 Mean square error different methods 21](#_Toc454318391)

# Support vector machines for regression

## Simple dataset

### Linear kernel

|  |  |
| --- | --- |
| Figure ‑ gamma=Inf e=0.25 | Figure ‑ gamma=0.1e=0.25 |

The linear classifier doesn’t properly regress this data. The data points are put onto the shape of a wave, but the linear regression can only do straight lines.

### RBF kernel

When we apply the right parameters to our classifier with an RBF kernel we get a much better result. This is displayed in Figure 1‑3, which has relatively nice parameters. Figure 1‑4 and Figure 1‑5 show that the e parameter determines the width of margin. Figure 1‑6 illustrates that gamma determines the influence of errors on the model.

|  |  |
| --- | --- |
| Figure ‑ gamma=Inf e=0.25 | Figure ‑ gamma=Inf e=0.1 |
| Figure ‑ gamma=1000 e=0.25 | Figure ‑ gamma=2 e=0.25 |

### Bad non-linear case

The flexibility that an RBF gives can also backfire. And produce worse results then with the linear case. This is illustrated in Figure 1‑7 and Figure 1‑8.

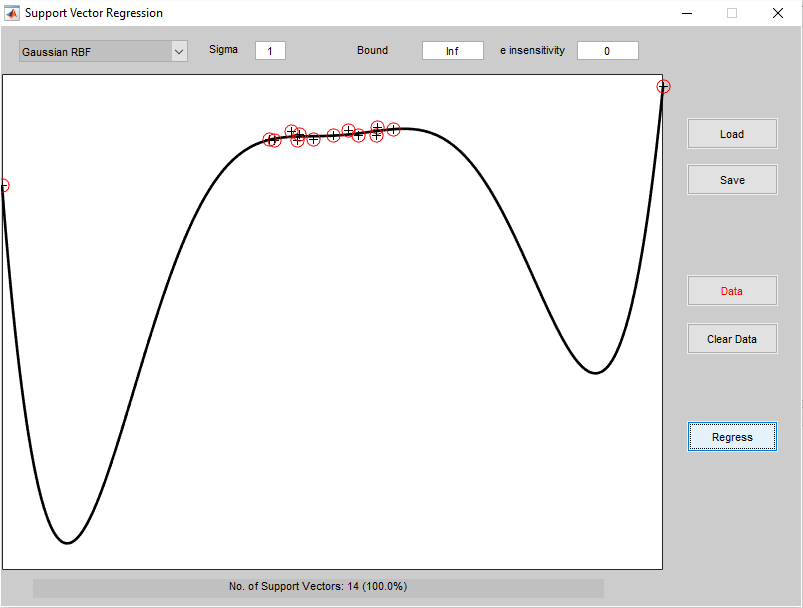


Figure 1‑7 dataset better with linear classifier, using RBF kernel gamma=Inf e=0

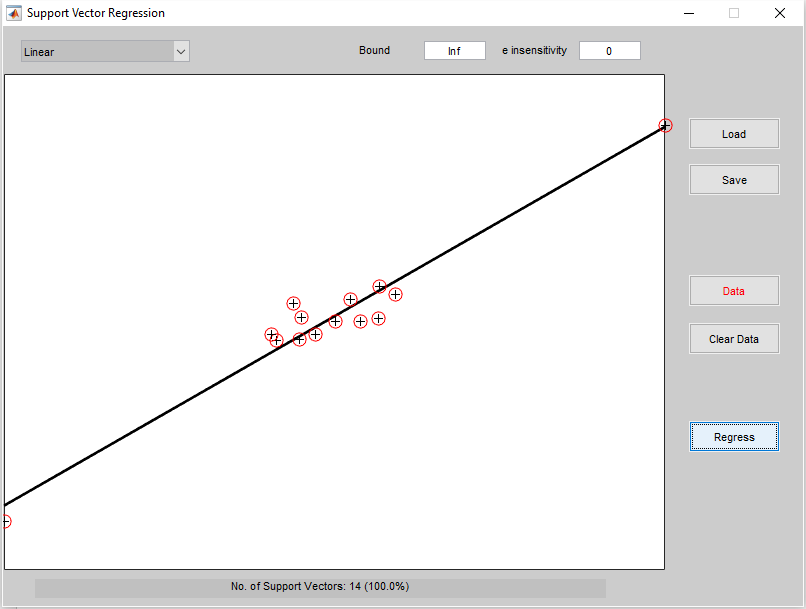


Figure 1‑8 dataset better with linear classifier, using linear kernel gamma=Inf , e=0

### Sparsity

The SVM tries to minimize towards the error caused by the points outside the margin due to the relationship with the Vapnik cost function. gamma is the weight on the miss classification/regression. When gamma is very low the SVM will almost not take in account the points outside the margin. This is illustrated in Figure 1‑9, Figure 1‑10, Figure 1‑11 and Figure 1‑12.

|  |  |
| --- | --- |
| Figure ‑ gamma=1, e=0.1 | Figure ‑ gamma=0.1, e=0.1 |
| Figure ‑ gamma=0.01, e=0.1 | Figure ‑ gamma=0.001, e=0.1 |

When changing the parameter e, the number of support vector change. This is illustrated in Figure 1‑13, Figure 1‑14, Figure 1‑15 and Figure 1‑16, 4 important values between 0.001 and 100 are displayed. As Its very clear that when the margin compresses more and more support vectors appear.

This means that the sparsity will disappear as e goes towards zero.

The difference between the Least square and the classical support vector machine is that every error even very small ones are taken into account with an least square. However the classical SVM will use the Vapnik -sensitive loss function. Which will only take into account the errors greater then . And will need less support vectors then the least square version.

This means that the classical SVM will be much sparser then the least square version.

|  |  |
| --- | --- |
| Figure ‑ gamma=Inf e=0.01 | Figure ‑ gamma=Inf e=0.1 |
| Figure ‑ gamma=Inf e=0.3 | Figure ‑ gamma=Inf e=100 |

Figure 1‑13,Figure 1‑14, Figure 1‑15 and Figure 1‑16 show what happens if e goes up. The sparsity as discussed early clearly shows up.

# A simple example: Sum of cosines



Figure 2‑1 gamma=10, sigma=0.01

When we decrease gamma then the LS-SVM will put less weight onto the errors and the regression is not good for every point. This is illustrated in Figure 2‑2 and Figure 2‑3.



Figure ‑gamma=1, sigma=0.01



Figure 2‑3 gamma=0.1, sigma =0.01

When gamma is to large we lose the smoothness of the regression. This is illustrated in Figure 2‑4.



Figure 2‑4 gamma= sigma=0.01

Figure 2‑7 contains a plot of a possible optimal value for sigma and gamma. If we increase the parameter gramma, we will lose flexibility. This is illustrated in Figure 2‑5and Figure 2‑6, when we decrease we get too much flexibility, this is illustrated in Figure 2‑8.

|  |  |
| --- | --- |
| Figure ‑ sigma=1, gamma=10 | Figure ‑sigma =0.1, gamma=10 |
| Figure ‑ sigma=0.01, gamma=10 | Figure ‑ sigma = , gamma=10 |

It is very clear from the previous examples that there will be some sort of optimal value for sigma and gamma.

# Hyper parameter tuning

## Gridsearch

Gridsearch will try different parameter combinations and retain the best one. This is a brute force way of finding the optimized parameters. This is easily to parallelize as you could simulate the different possible values parallel and compare the cost in the end.

## Simplex

The simplex method is an improvement upon the gridsearch. It will adapt its step size as its searches And it will perform fewer function evaluation to save time. This is an iterative algorithm.

Source: http://people.eecs.berkeley.edu/~jrs/4/lec/23

## Csa and ds

The csa algorithm takes quite a bit of time which is noticeable when executing the tune function. It does however give us a better starting value which makes the simplex algorithm significantly better. The randomized directional search starts searching for an optimum in different directions.

## Optimilisation with ds

If we do 100 times the optimizations with directional search we get the results display in Table 3‑1. The gamma and sigma parameters change significantly. And so the cost changes drastically, sometimes we get really good performance but other times we get really bad performance. This is so with both gridsearch and simplex.

|  |  |  |
| --- | --- | --- |
|  | gridsearch | simplex |
| min(cost) | 0.0119 | 0.0119 |
| max(cost) | 0.9455 | 0.9502 |

Table 3‑1 tuning hyper parameters with ds (simplex or grid search)

## Optimilisation with csa

When we do the same 100 simulations with csa we get a way better results. Notice that this time the simplex is better than the gridsearch. This was not so with the ds algorithm. This is because simplex heavier on its starting conditions. And csa gives us a better starting condition.

|  |  |  |
| --- | --- | --- |
|  | gridsearch | simplex |
| min(cost) | 0.0119 | 0.0123 |
| max(cost) | 0.0145 | 0.0136 |

Table 3‑2 tuning hyper parameters with csa (simplex or grid search)

Even tough ds is not as reliable as csa it is the fastest method. This is due to the parallelization of the ds method. Which makes it a lot faster. Maybe we could run ds a couple of times a take the best result, this way we can get better results like csa.

# Application of the Bayensian framework

## Toy problem

The error bars are displayed in Figure 4‑1.



Figure 4‑1 error bar regression

The red lines on Figure 4‑1 are the error bands. The crosses are the data points and the black line is de regressor.

## Iris

The area dominantly blue is there area with lowest change on a point from the positive class. The violet area is the place where one would expect the points from the positive class.



Figure 4‑2 probability a certain point belongs to the positive class, give the model ,gam=5,sig2=0.75

### Influence of gamma

Figure 4‑3, Figure 4‑4, Figure 4‑5 and Figure 4‑6 illustrate the influence of changing gamma.

If gamma is very low, then the lssvm is not heavily influenced by the error. This means that there is a clear border between the positive and negative classes. And the classifier clearly knows where the most positive and most negative points are.

When gamma grows the influence of the errors grow, and it’s become vaguer and vaguer. We can see a lot of area with 0.4-0.6 probability. The only area that is still clear is the empty space off course. But the rest has become middle ground.

|  |  |
| --- | --- |
| Figure 4‑3 gam=0.05, sig2=0.75 | Figure 4‑4 gam = 1; sig2 = 0.75; |
| Figure ‑ gam=10; sig2 = 0.75; | Figure ‑ gam = 100; sig2 = 0.75; |

### Influence of sigma

When sigma is very low , the LS-SVM will either cluster to much such as shown in Figure 4‑7. Or it will cluster not at all as shown in Figure 4‑10.

|  |  |
| --- | --- |
| Figure ‑ gam = 5; sig2 = 0.01; | Figure ‑ gam = 5; sig2 = 0.5; |
| Figure ‑ gam = 5; sig2 = 1; | Figure ‑ gam = 5; sig2 = 10; |

### Automatic relevance determination (ARD)

We define Y as Y = cos(X(:,1)) + cos(2 \* (X(:,1))) + 0.3.\* randn(100,1);

This means that X(:,1) is closely related to Y, and should provide the best training data.

When we call [selected, ranking] = bay\_lssvmARD({X,Y,'class',gam,sig2});

we get selected=1 and ranking=1;2;3. Offcourse this will pick the first dimension as the best training data as this is the only one directly related to Y.

To illustrate this we will use the following function again:

sig2e = bay\_errorbar({X(:,1),Y,'f',gam,sig2},'figure');

In Figure 4‑11 we see the result for the first dimension this is clearly way better than the result for the second and third dimension illustrate in Figure 4‑12 an Figure 4‑13



Figure ‑ X(:,1)

|  |  |
| --- | --- |
| Figure ‑ X(:,2) | Figure ‑ X(:,3) |

We can do something similar with cross validation. We simply crossvalidate and when the removal of the data does not change the performance of the SVM then we know that this data is irrelevant. We should remove it and go on to the next iteration, till we can’t remove any more data without messing up our SVM performance.

# Robust regression

## Basic

If we have data with outliers then this might confuse our LS-SVM as is illustrated in Figure 5‑1. The LS-SVM is optimizing the regression to get a minimum error towards all points and so it is making compromises at the positions where the outliers are. Because it considers the outlier as valid as a normal points.

If we however use the robust cross validation as is illustrated in Figure 5‑2 we get a better result. The crossvalidation will recognize that there are outliers and give a lower weight to them. There are “less” important to the regression.

|  |  |
| --- | --- |
| Figure ‑ LS-SVM gam = 100;sig2 = 0.1 | Figure 5‑2 LS-SVM robust cross validation |

## Advanced

### mae vs mse

The difference between mae and mse has to do with the loss function. In many circumstances it is usefull to give more weight to points further away from the mean, that is, being off by 10 is more than twice as bad as being off by 5. In such cases mse is a more appropriate measure of error.

If being off by ten is just twice as bad as being off by 5, then mae is more appropriate.

So when we use mse as is demonstrated in Figure 5‑3 the it won’t suppress the heavy tail as bad like with mae as demonstrated in Figure 5‑4.



Figure 5‑3 simulations using whuber loss function with mse, 50% gave the left 50% gave the right results



Figure 5‑4 simulations using whuber loss function with mae 9 simulations gave the left result one gave the right result

When we use mse we seen that there is one loss function that performance very badly. However if we execute these again other loss functions start to behave badly. Only when we use mae a;; the loss function seem to perform decently.

|  |  |
| --- | --- |
| Figure ‑ whuber mse | Figure ‑ whampel mse |
| Figure ‑ wlogistics mse | Figure ‑ wmyriad mse |

# Homework problem: intro Time-series prediction

Figure 6‑1 illustrates the AR model with an LS-SVM. Nothing special to see here compared to the previous LS-SVM’s that we saw.



Figure 6‑1 ls-lsvm AR model

Through the predict command we can calculate a prediction of the serie. We remove the test data from the learning data that contains the future. And then predict the future and so the test data. This is visualized in Figure 6‑2. The mean square error is 0.0244, which is quite good. But this simple toy problem obviously has an easily predictable shape, it might be time to try a more serious serie.



Figure 6‑2 test prediction, blue=test values red=prediction

## Changing order

Figure 6‑3 illustrates the effect of the order on the repressor. When the order the repressor will simply not be flexible enough to predict the data. This is also illustrated in Figure 6‑3, the low order will have a large mse. An order of 10 seems to be ideal and higher than 10 will create oscillations which will off course introduce error. When the order is to large, it will become to flexible and start to oscillate.



Figure 6‑3 changing order ls-svm predictor



Figure 6‑4 mean square error different orders

# Homework problem: Santa Fe Laser dataset

## Dataset

The Santa Fe laser dataset is displayed in Figure 7‑1, with dots and with lines. It is clear that there are no significant outliers.



Figure 7‑1 plot of dataset, line plot on the left and a dot plot on the right



Figure ‑ Test data set

## Questions excercise

### Window size

The window size of 50 sounds reasonable, but so does 100 to me. If we would take it much smaller than the model can’t learn much from it. And if we take it a lot bigger than 100 then we will hardy have a window but almost add all the data.

From the experiments it does appear that this choice seems to be good.

### Recurrent prediction

If this validation set was used to optimize the parameters from the ls-svm. Then it would be a bad idea to use these to measure the performance. If we use the test data to measure the performance then we can have an more realistic idea of the performance.

But yes we could for instance measure the performance with a cross-validation set and then further optimize the LS-LSVM if needed. Change for instance the window size and see if this is better.

## SVM predictor

### best Solution

I used the RBF kernel because its quiet obvious a linear SVM does not allow the flexibility I need. I optimized it with csa/simplex and crossvalidation. Because that’s what we did earlier and csa + simplex appeared to work best. It is however quiet slow, but that doesn’t matter.

|  |
| --- |
| [gam,sig2,cost] = tunelssvm({Xtrain,Ytrain,'f',[],[],'RBF\_kernel','csa','original'}, ...  'simplex','crossvalidatelssvm',{100,'mae'});  [alpha,b] = trainlssvm({Xtrain,Ytrain,'f',gam,sig2,'RBF\_kernel','csa','original'}); |

Figure 7‑3 LS-SVM optimizing and training

We can see the LS-SVM plotted in Figure 7‑4



Figure 7‑4 SVM plot

We can see the result plotted in Figure 7‑5, it was much better then what I hoped for.



Figure 7‑5 (blue=prediction , red=Ztest)

### Using Bayesian

After trying the solution from the previous part, I wondered if I would get decent result with Bayens. I wasn’t too sure about what to expect so I made a little test setup. The result is displayed in Figure 7‑6 and Figure 7‑7. Figure 7‑7 shows that there is some serious problem with this svm, the gam en sig2 parameter are not properly set.

|  |  |
| --- | --- |
| Figure 7‑6 prediction | Figure 7‑7 plot of svm |

|  |
| --- |
| [gam,sig2,cost] = tunelssvm({Xtrain,Ytrain,'f',[],[],'RBF\_kernel','csa','original'}, ...  'simplex','crossvalidatelssvm',{100,'mae'}); |

Figure 7‑8 staring value sig2 and gam, dynamically

Initially i used a starting value for gam and sig2 as displayed in Figure 7‑8 , if changed to Figure 7‑9 we get way better results This is displayed in Figure 7‑10and Figure 7‑11.

|  |
| --- |
| gam=10000;  sig2=20000; |

Figure 7‑9 starting value sig2 gam, static

|  |  |
| --- | --- |
| Figure ‑ time plot prediction | Figure ‑ LS-SVM plot |

## Mean square error different methods

|  |  |
| --- | --- |
| Tunelssvm | 418.2185 |
| Baysian | 449.6797 |

The classical way of optimizing still seems to be slightly better than the Bayesian interference.