Master of bioinformatics

Support Vector Machines

Assignment 3: Unsupervised Learning

Spring 2016

Author:
Cedric Lood

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

May 29, 2016

Contents

Context

The analysis presented in this report was produced for the class of "Support Vector Machines: methods and applications" at KU Leuven (Spring 2016). The goal is to display understanding of the techniques and of their practical use. This third report focuses on unsupervised learning (kernel PCA), and fixed-size LS-SVM using Least-Squares SVM. The implementation was done using the MatLab environment (v2015a) and the libraries for LS-SVM developed at KU Leuven [1].

1 Kernel Principal Component Analysis

In this section, we use the synthetic dataset illustrated on figure 1 to explore the relationship between the choices of kernel, hyper-parameters and the number of components. The goal behind this type of exercise is to eliminate the noise. Indeed you would hope that the noise is subtracted when the reconstruction process takes place.

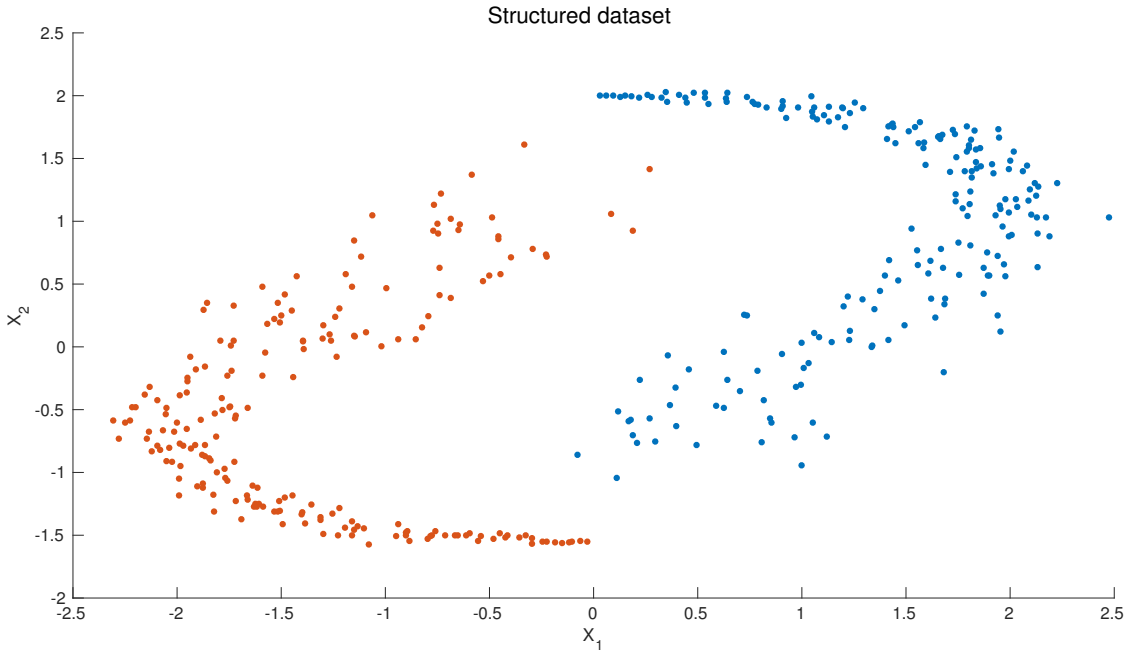

Figure 1: Synthetic dataset Yin-Yang

Figure 2 allows us to visualize what happens when you increase the number of component for a fixed $\sigma^2 = 0.4$. We can see that the denoising works really well once the number of components has been increased to somewhere between 4 and 8. Figure 3 illustrates the impact of σ^2 on the denoising for a number of components fixed to 6. A good value for σ^2 seems to lie somewhere between 0.1 and 0.5. Above those values, more of the noise is being reconstituted.

Given the structure of the dataset, linear PCA gives very poor results in terms of denoising (see figure 4). The maximum amount of principal component that can be obtained using linear PCA corresponds to the dimensionality of the original input space (p). For the kernel PCA, the dimensionality can be higher, and is in fact completely independent of the

[1] http://www.esat.kuleuven.be/sista/lssvmlab/

dimensionality of the input space. In this setting, it corresponds to the dimensionality of the kernel matrix (which is n-by-n, with n being the number of data point).

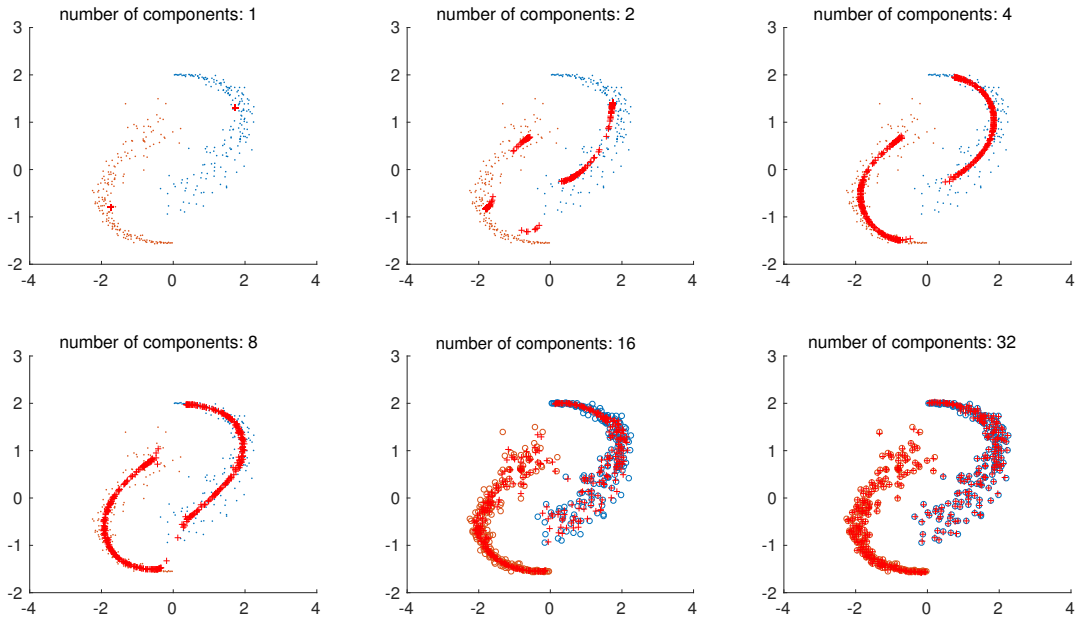

Figure 2: Denoising for various values of components

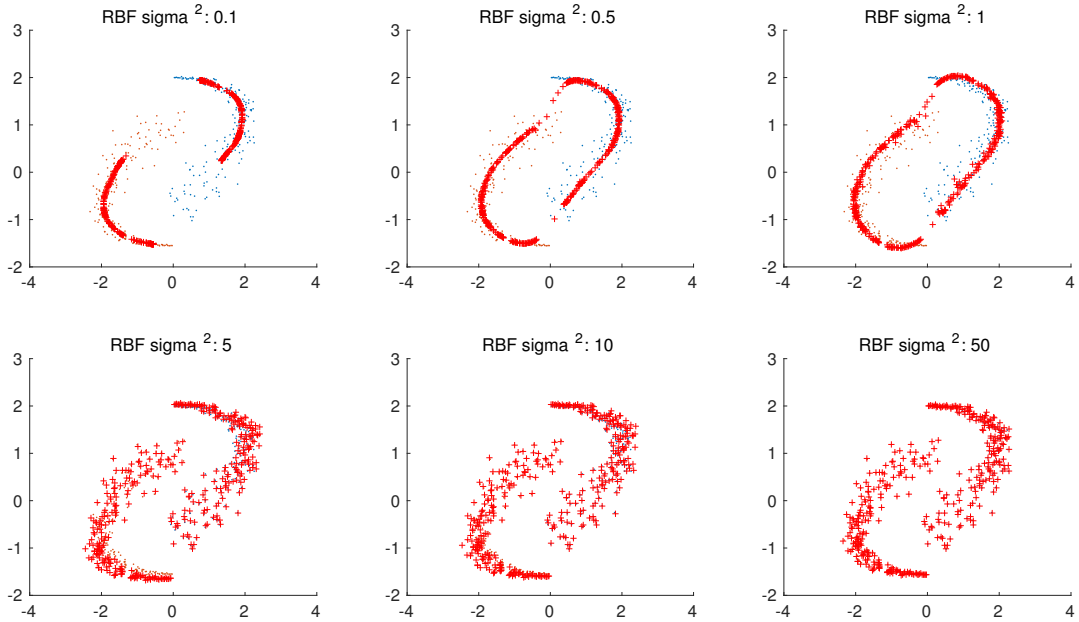

Figure 3: Denoising for various values of σ^2

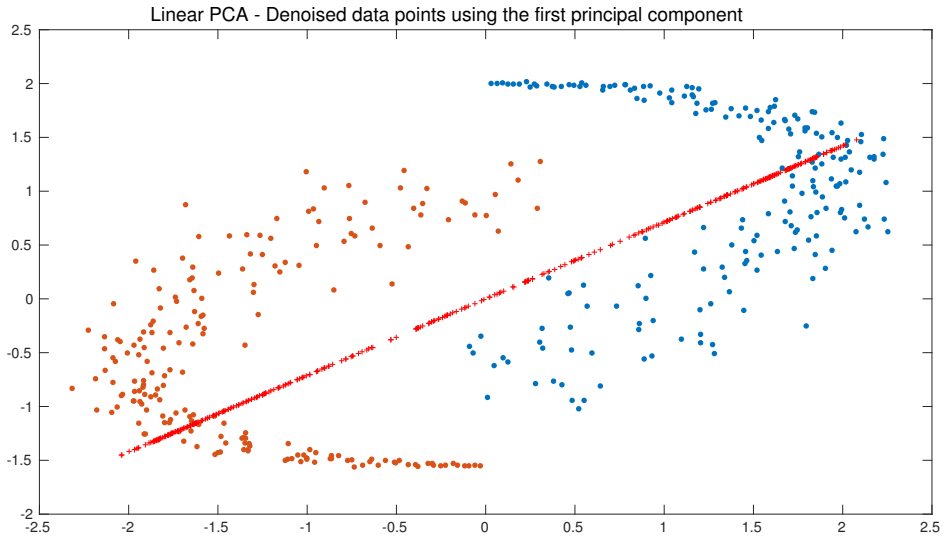

Figure 4: Denoising in linear PCA setting

A natural question is whether there is a good way to tune the number of components and the kernel parameter, eg σ^2 in the case of a RBF kernel. I would say that one way to proceed would be to systematically compute the error between a denoised model trained on a part of the dataset (training set), then compute the error made using the test set. In the example above, it is easy to see that with very high number of components or a large σ^2 would result in a fitting of the noise. Given that we know the underlying true function, a calculation of the error would reveal the overfitting.

2 Handwritten Digit Denoising

The results of the denoising process are reported on figures 5 and 6. As can be observed, the performance of the kernel PCA in terms of denoising are much better. Basically, the linear PCA decomposition into 190 components, followed by the reconstruction using progressively more and more components (190 in the last row) returns the original noisy image. Whereas the kernel PCA is able to extract out the gaussian noise during the reconstruction, returning a very convincing, denoised image which is similar to the original noiseless one.

Interestingly, it is possible to see some errors in the reconstruction of the digits using the kernel PCA approach, indicating that a certain amount of components is necessary. For many of the digits it would seem that using very few components for the reconstruction is biased towards 8 and a 7. Increasing the components resolves the problem for most digits.

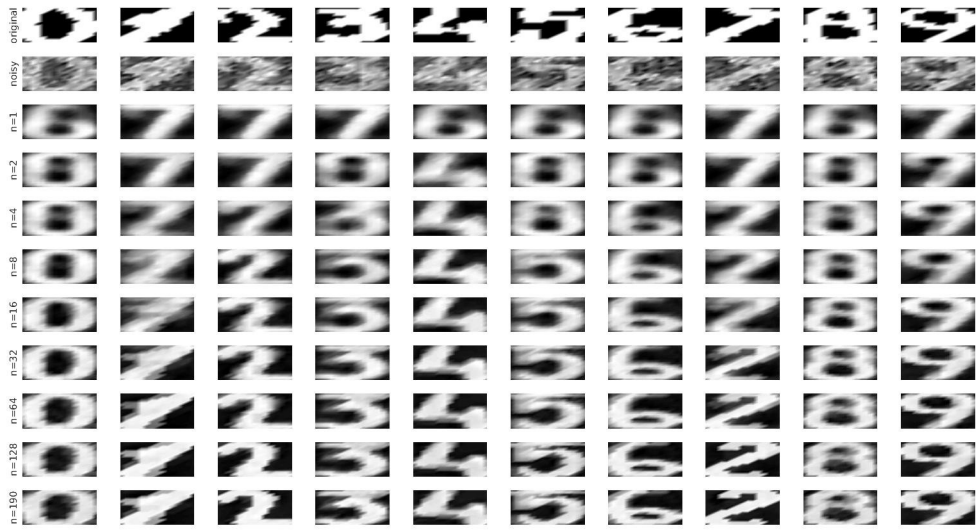

Figure 5: Denoising digits using a kernel PCA approach

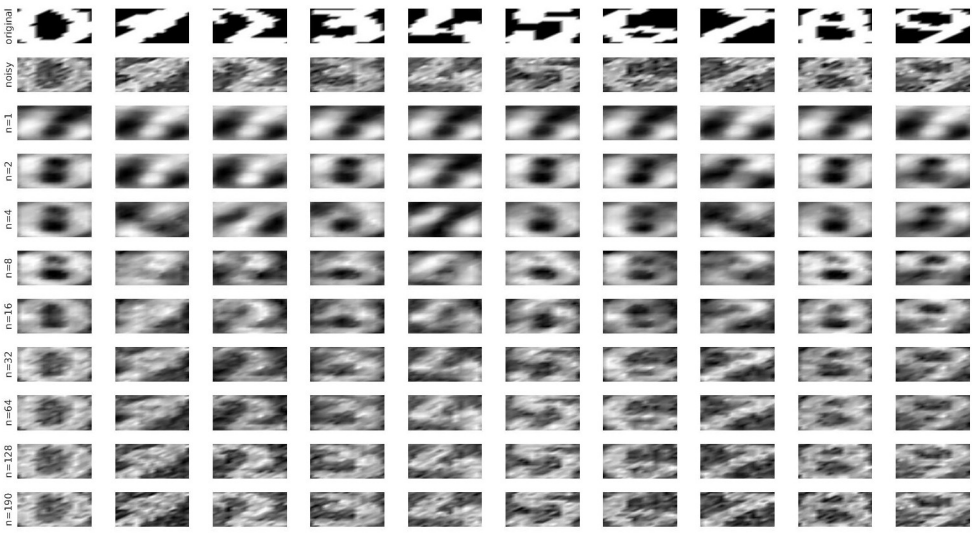

Figure 6: Denoising digits using a linear PCA approach

3 Spectral Clustering

The difference between clustering and classification lies in the type of learning at work. In clustering, no labels are used - a setting usually referred to as *unsupervised learning*. Often, a notion of distance will be defined, and objects will be clustered together based on whether or not they are close to each other. In classification, labels are available and are used as targets to train the model - a setting usually referred to as *supervised learning*.

On the figure 7 and 8 we can see the results fo the spectral clustering for various values of σ^2. Selecting $\sigma^2 \in [0.001, 0.01]$ leads to good clustering results, ie a good separation of the two rings. A good separation is also visible for those values on the projection graphics.

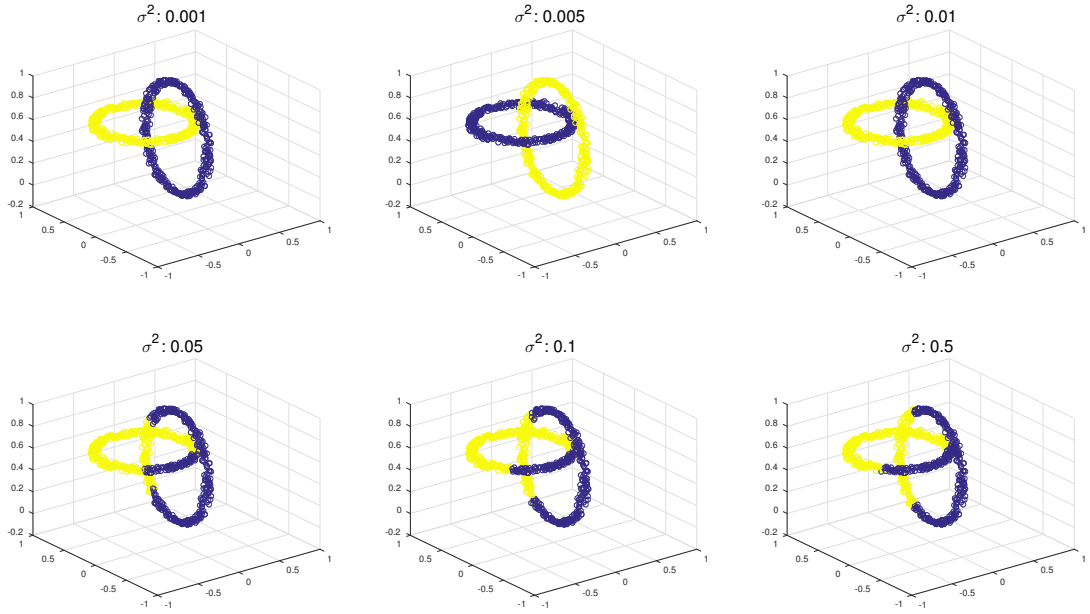

Figure 7: Clustering of the two rings

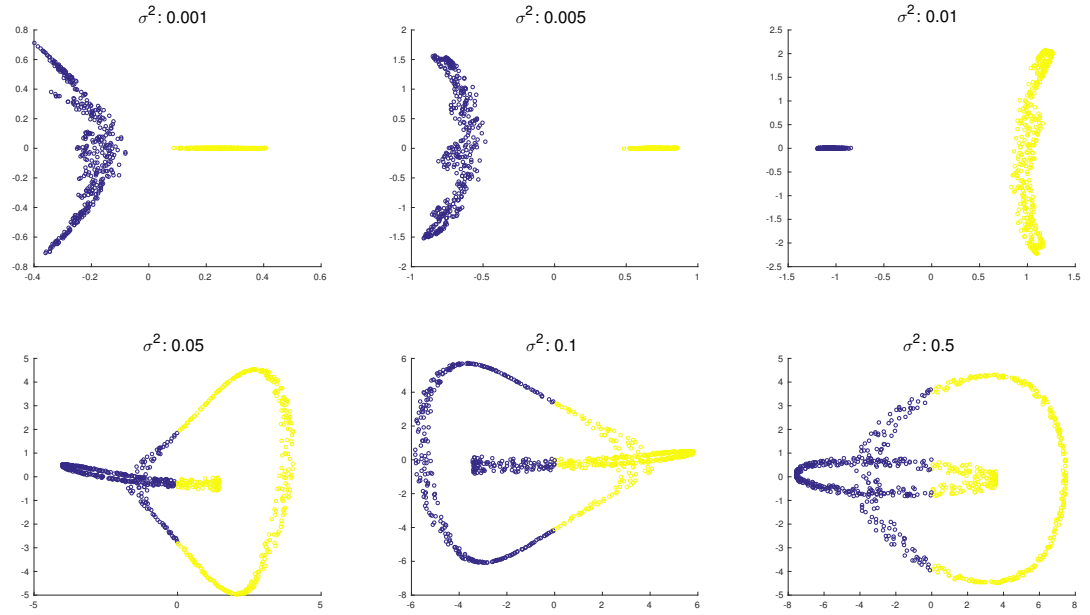

Figure 8: Projection onto subspace spanned by the 2nd and 3rd largest eigenvectors

Figure 9 reports the kernel matrix for a value of $\sigma^2 = 0.05$, this is a bit suboptimal, but the better value $\sigma^2 = 0.005$ gave a rendering that is mostly dark - lots of support vector, matrix not sparse.

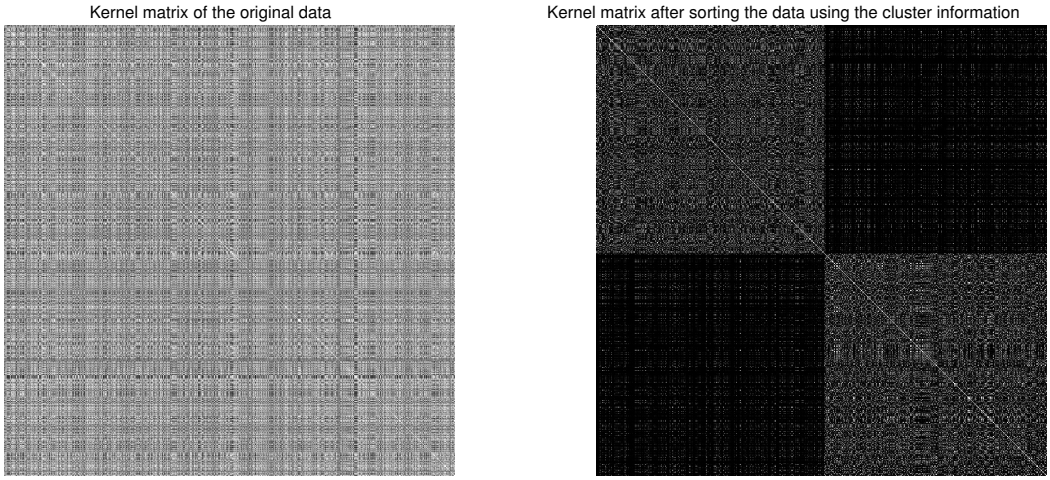

Figure 9: Matrix

4 Fixed-size LS-SVM

In this section, we use a Nystrom approximation of the feature map. This allows us to construct parametric models in the primal space. By means of finite samples of the datapoints, one can obtain a finite estimation of the feature map.

Figure 10 reports the influence of different values of σ^2 on the subset selection (entropy criteria). Larger values of σ^2 seem to result in a selection that is more evenly spaced.

The algorithm should converge to a set of points that represent best those needed to build a correct model in the feature space.

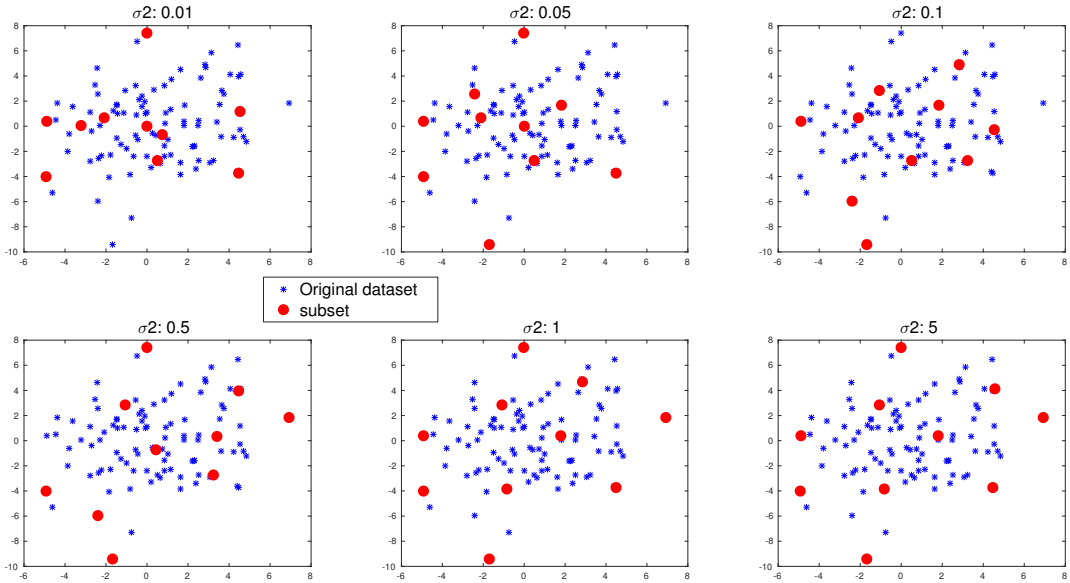

Figure 10: Subset selection via fixed-size lssvm

Figure 11 shows the results in terms of different metrics for the two approaches, fixed-size lssvm and the $l_0 - approximation$. As can be expected, the number of support vectors is much lower for the $l_0 - approximation$. The error between the 2 approaches is quite similar, with a larger variance for $l_0 - approximation$.

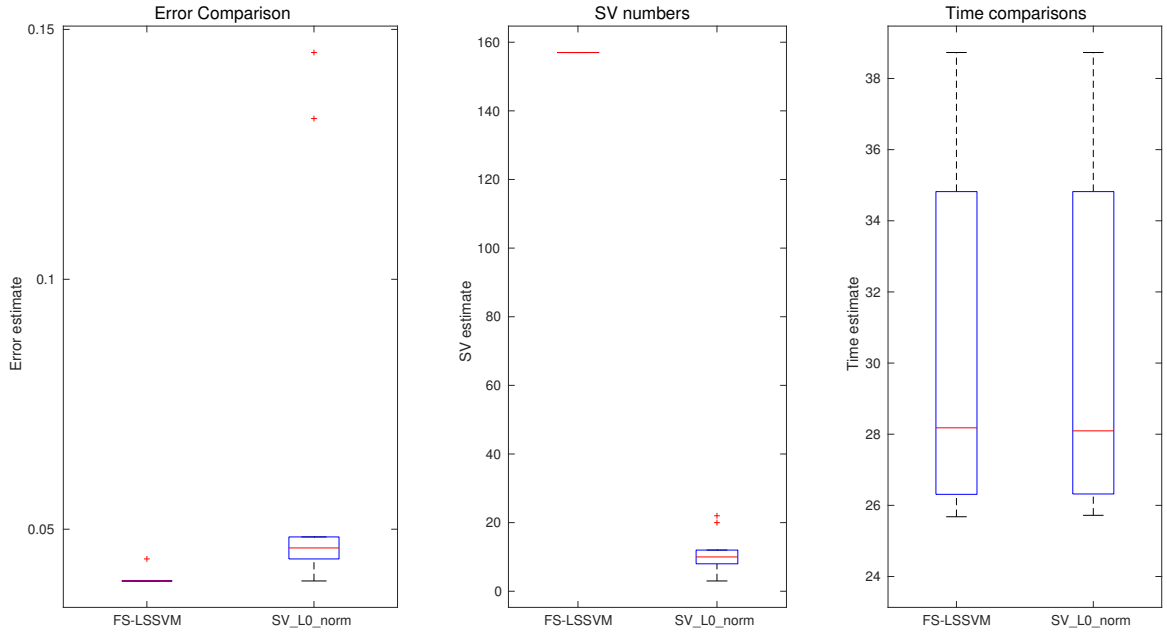

Figure 11: fslssvm-script

5 Applications

5.1 Handwritten Digit Denoising

Here we revisit the denoising of the handwritten digit. First, I explored the impact of the value of the σ^2 parameter on the denoising. The results are reported on the figure 12 below. As you can see, with larger values of σ^2, the reconstruction process through which the noise is eliminated tends to decrease in efficiency. For exagerately large values of the parameter, the reconstruction is not legible, and consists only of noise.

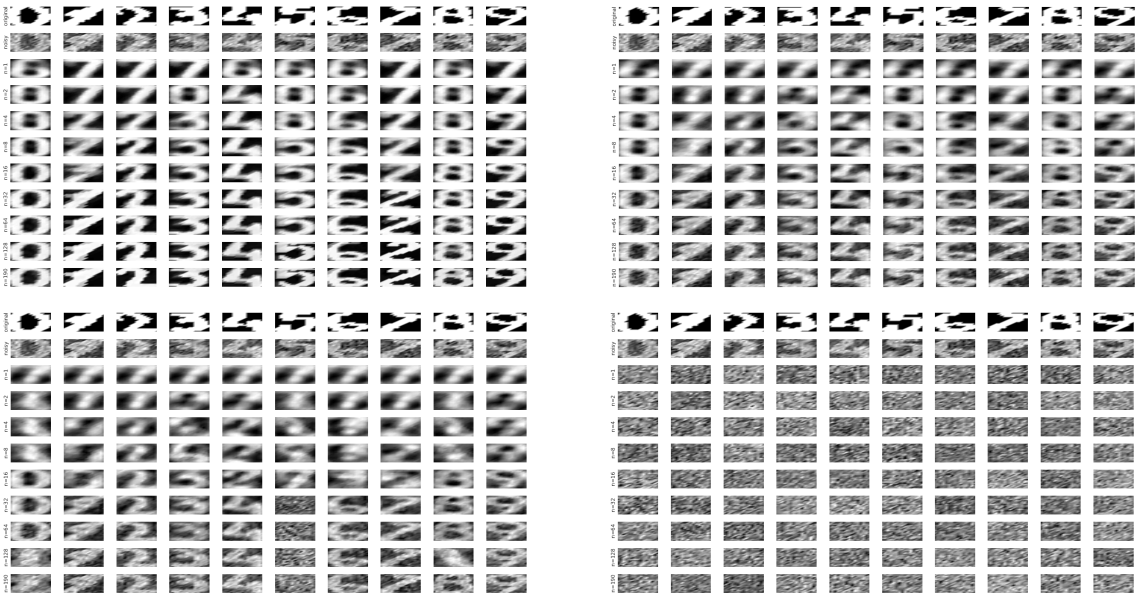

Figure 12: Reconstruction different sigma (top to bottom, left to right: [50, 500, 5000, 50000]

An example of linear vs. kernel PCA was given and commented in section 2, figure 5 and 6.

5.2 Shuttle (statlog)

This exercise gives us a taste of what it is to work with large dataset. As we know from the theory behind SVM, working with the dual problem has great advantages, but requires to build the kernel matrix, of size n-by-n. This matrix can be too large to store, or we may want to work in the primal problem, which is of the dimension of the input space. However, when we use a RBF kernel, we would need an explicit construction of the feature map (infinite dimensional for the RBF kernel). Working with fixed-size ls-svm allows us to get an approximation of the feature map, and build our model using the primal formulation.

For this exercise, one of the challenge was to deal with the sheer amount of computations required for the fslssvm to complete. A first attempt on my personal computer revealed a progress of less than 4% for 1 hour of computing time. I therefore used the VSC infrastructure, where one node with 20 cores cut down the computation time to around 3 hrs.

The results are displayed in figure 13. Note that I divided the dataset between the training set and test set with a ration of 70/30.

The errors are comparable for both approaches, and so was the time estimate. The support vectors comparison however revealed that fewer of them were necessary on average for the SV_L_0norm than for the FS-FLSSVM approach (which is expected, see [2]).

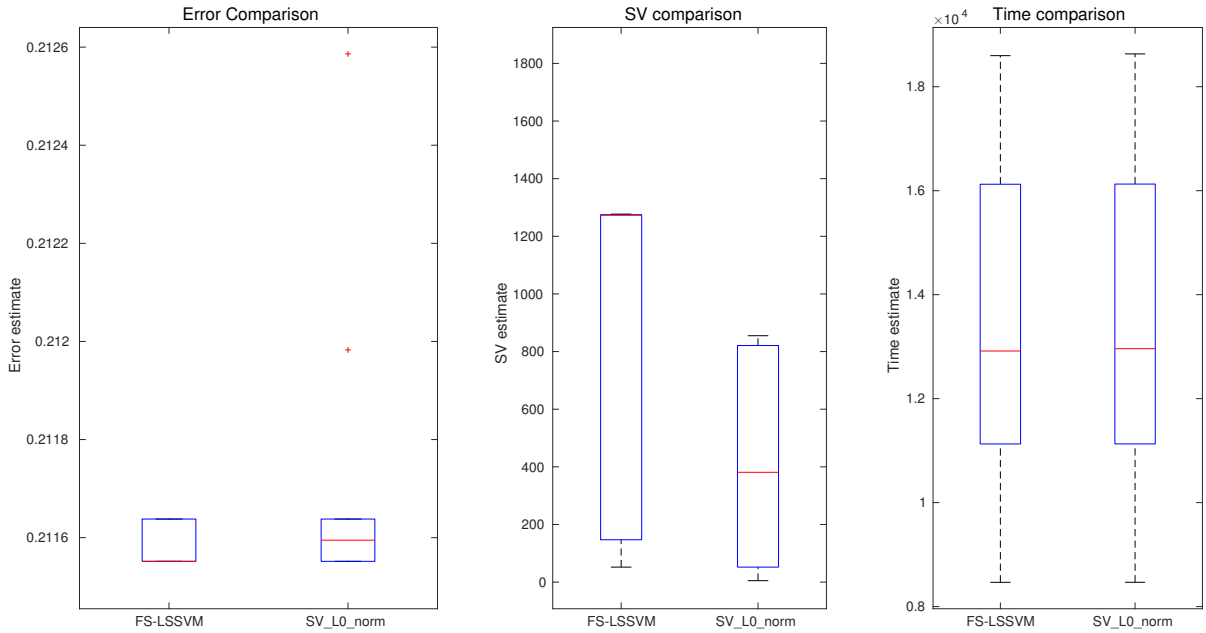

Figure 13: Metrics comparison for fs-lssvm and l_0

5.3 California

Similar interpretation of the results can be given for this dataset and are displayed in figure 14. The dataset was also divided in a 70%/30% scheme for the training and test.

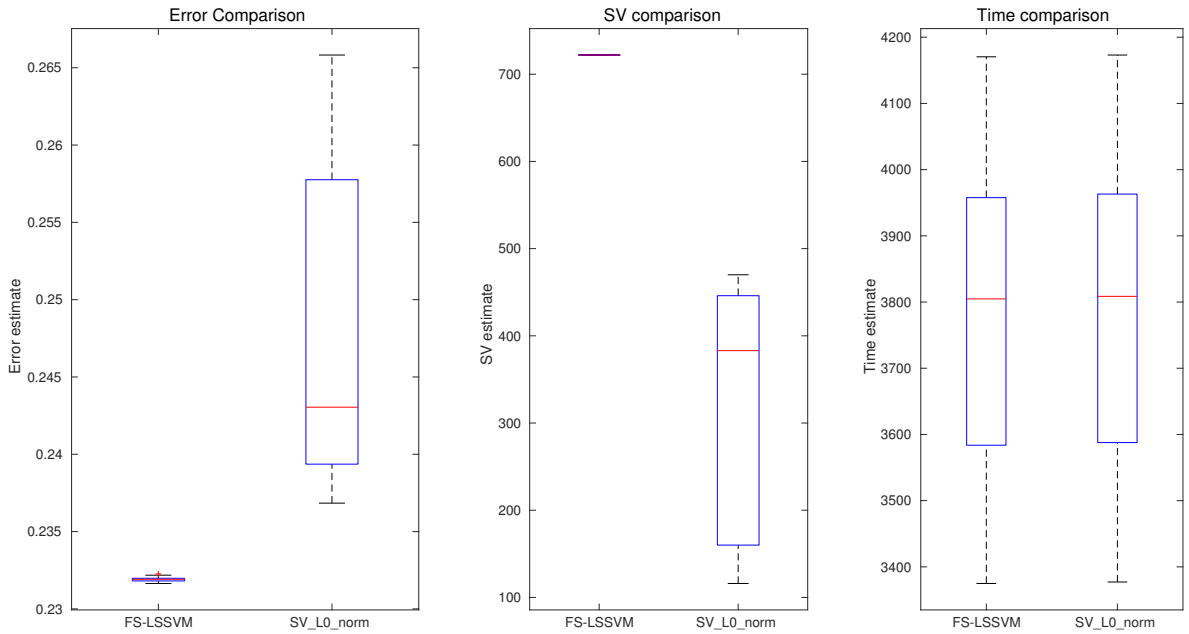

Figure 14: Metrics comparison for fs-lssvm and l_0

[2] *Sparse LS-SVMs with L 0 –norm minimization* by Lopez et al.