Kenneth Devloo, s0219469

Support vector machines: Session 3

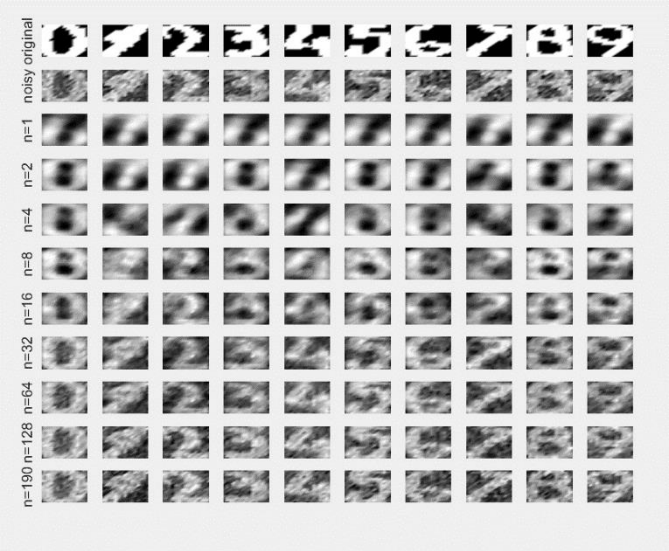
3.1 Kernel Principal Component Analysis

In the script kpca_script., at first, we see data is created according to the user input. We want to express this data with as few as possible parameters. In the feature space, eigenvectors and values are calculated so that we can do dimensionality reduction. In this script, the first eigenvector is used to project and reconstruct the data as optimally as possible so that we can represent the data as a 1-d dataset in the new feature space constructed with the eigenvectors. The result is that all info is compressed on 1 line, including the noise.

With linear pca, you can obtain as many dimension as your input space has. With kernel pca this is, in theory, infinite. By applying the kernel trick, your data becomes better separable. What typically happens is that the data gets projected with just a part of the variance. This can remove outliers/noise. Reconstructing the data will result in a less noisy dataset.

3.2 Handwritten Digit Denoising

In the image that is generated by the script, we see the original image. Next, noise is added that will be moved with kernel PCA (RBF kernel). The result of reducing the dimensions is shown. Taking too few dimensions in feature space will also destroy a lot of information and make the digits harder to recognize.

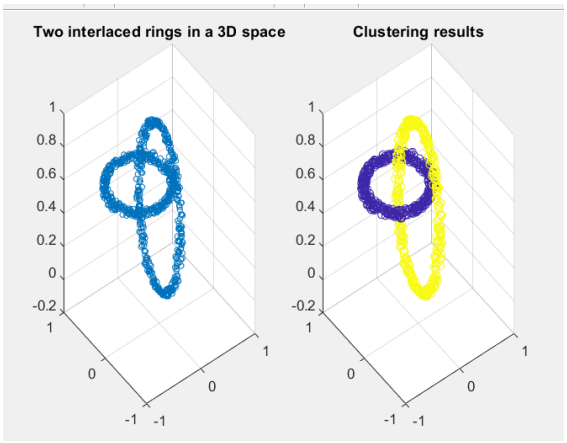


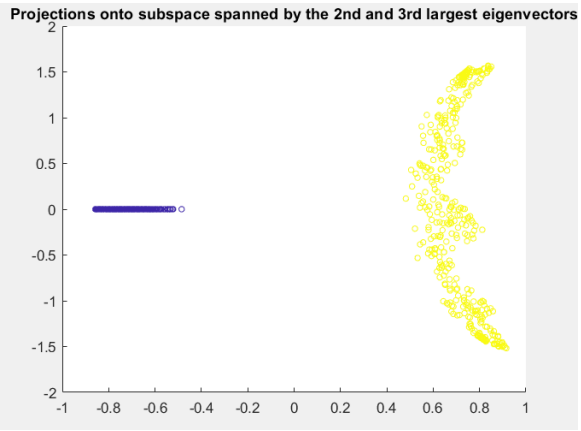
3.3 Spectral Clustering

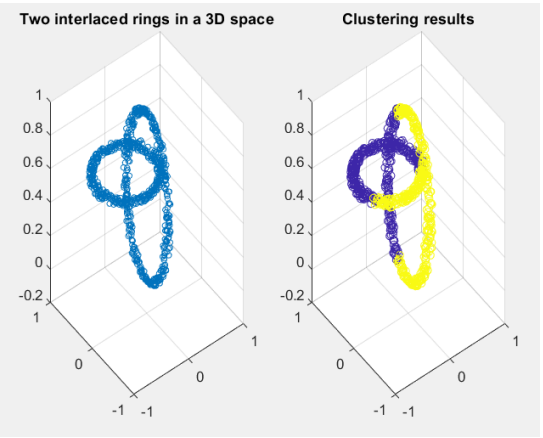
A difference between classification and clustering is that clustering is and unsupervised method, meaning that you do not have a ground truth to start from but you look for similarities. In the same way, you will look for the biggest similarities here with the eigenvectors. The clustering can thus be done as a form of kernel PCA.

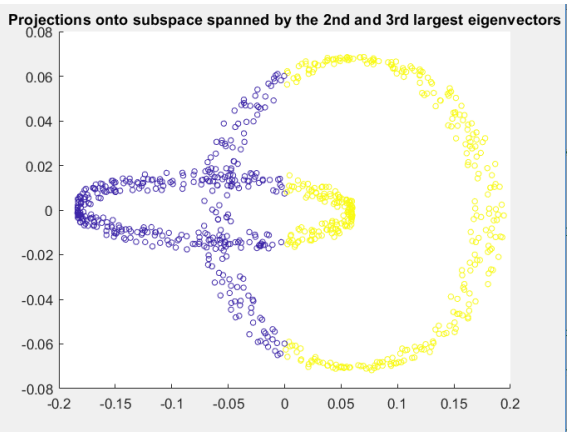
The distance between points in the kernel that is calculated depends on the sig2 parameter. Therefore, a smaller value. Taking this too small will make it difficult to say much about the distance between points. And therefore using pca here will also not give much usable results. However, we know that to separate 2 classes, we also needed to strike a balance between too small and too large. If we only have values near 0, the eigenvectors will also not say much when trying to calculate with these values. Putting sig2 at 0,005 gives some very nice results. In the kernel matrix and on the eigenvectors used, we also see the separation. Putting sigma too high will make distinction very hard

since all data influences all data more and separation will be more 'raw'. (3th 4th picture, sig2 = 50). More differently: Take sig2 small enough so that different clusters do not influence each other, but that there will still be a connection between datapoints possible in the kernel matrix.









3.4 Fixed-size LS-SVM

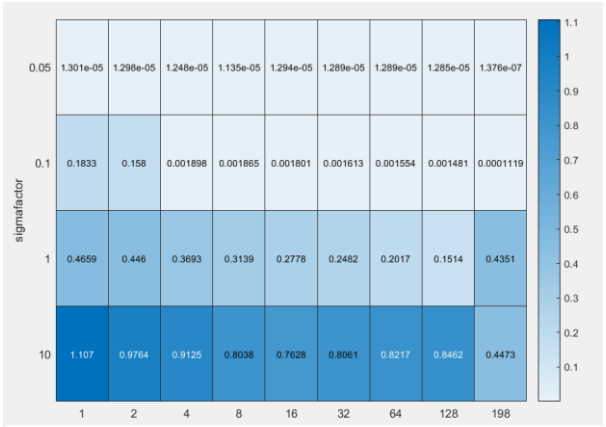
The algorithm gives us an optimal set of support vectors for a class which can be used by a classifier, given a certain sig2 variable. This way our solution becomes sparser and we can classify with less calculations. A sig2 too small will give some difficulties for actually calculating entropy in the data. Sig2 should depend on the variance in the data.

By optimising with the l0-norm, we can find a sparser solution with less support vectors without losing much accuracy on the test set. This makes classification more efficient.

3.5 Homework Problems

3.5.1 Handwritten Digit Denoising

With a large sig2 value, the kernel pca will give more weight to the noise and include it in the reconstruction, resulting in images that are just as bad.

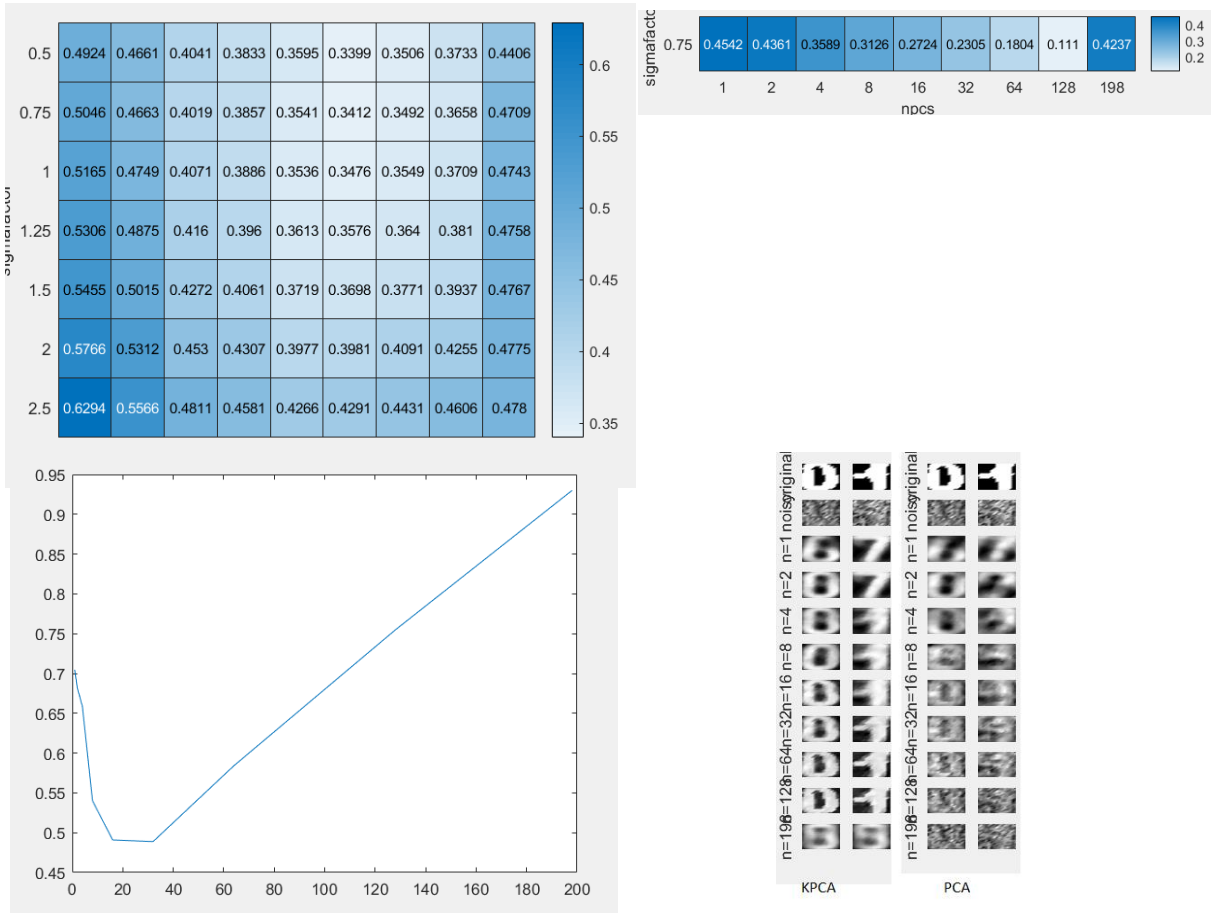


1Results on a test set.

The only thing sigmafactor changes in the script is the value of sig2 itself. Therefore, raising this will have the same effect as described earlier in this report. When performed on Xtest1, sigmafactor=1 clearly gives better results. When performed on Xtest2, the lower the sigma, the better the results. Given are the mean square root errors and restoration results with Xtest2 to illustrate what was just stated. I suspect that in the smallest cases of sigmafactor, restoring to see for example part of the noise does not work well since we work with values that are too small to compute with. I will therefore not take these smaller values into account.

For the next part, different values for sigma and the number of principal components have been used on a validation set first. This sigmafactor is then used on Xtest2 to see if that gives lower errors.

As you can see in the image below, using 32 principal components and a sigmafactor of 0.75 seems like the best solution. We see a slight improvement there. But we also see that validation did not give the best amount of principal components.



2Top left: validation results, top right: test results, bottom left: linear pca results, bottom right: comparison between kpca and linear pca

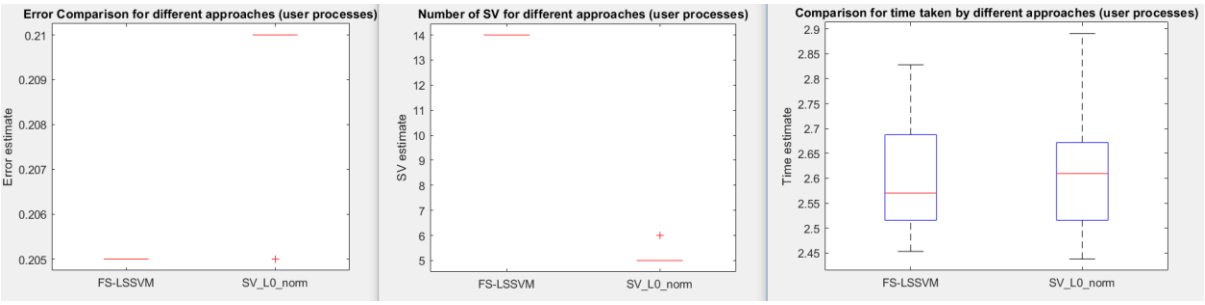
When comparing the results on 2 characters, in this case 5 and 7, we clearly see that kernel pca outperforms linear pca as it does not restore noise that easily.

3.5.2 Shuttle (statlog)

For the exercises on fs-SSSlssvm classification and regression, it was a bit unclear how this script could produce predictions Y, given values of X, even when this was specified in the readme file in the fixedsize folder. There also seems to be a problem with the documentation stating the RBF_kernel is supported but lin_kernel not. In the first case, the script will fail. Therefore, the discussion on this problem is limited to the standard visualised output.

This is problem with multiple class outcomes. There are 9 attributes for each entry, of which one is or example time.

The shuttle example is actually the same as in 3.4. Below are the visualisations that go with it. We see that there is a much sparser solution now, using less inputs per entry and limited the amount of support vectors coming from the data of 700 entries. This is a multiclassification problem so this reduction is quite strong.



It was also interesting to make this a problem with 2 classes since some classes are quite exceptional. The difference was actually quite small. The results here were quite similar.

3.5.2 California

This regression problem is turned into a sparse solution in a similar way as happened in the previous section. The most important difference here is that using the L0-norm caused a loss of precision. Along with the extra reduction in support vectors.