SVM : assignment 3

Florentijn Degroote

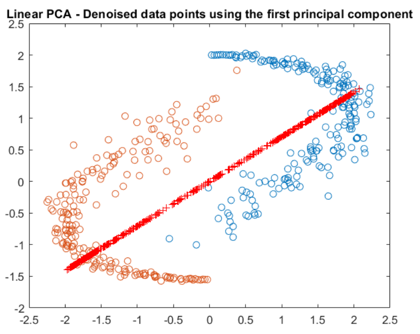
May 2019

Contents

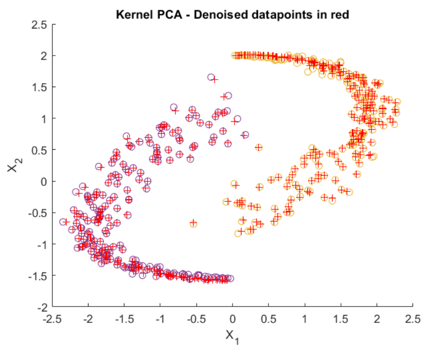
1 Exercises

1.1 Kernel principal component analysis

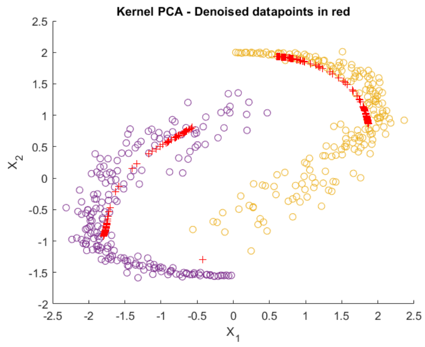
When denoising a data-set, such as the given yin and yang data set, the main goal is to generalise so that the underlying distribution is represented without the noise. This can be done by a kernel component analysis, especially, the non-linear variant. When reconstructing the data set, based on this non-linear kernel, the noise should be evaporated. Just like with auto-encoders, a dimension-changing technique is used. In the case of non-linear kernel PCA, it is possible to scale up the dimension, as opposed to traditional linear PCA which acts as a dimension reduction technique. PCA finds new directions based on the covariance matrix of original variables. It can extract a maximum of eigenvalues equal to the amount of features. KPCA finds new directions based on kernel matrix. KPCA's big advantage is that it can handle non-linear data sets, in contrary to traditional linear PCA (see figure 1a). It can extract n (number of observations) eigenvalues. Increasing the number of principal components will retain more and more information of the input space. When the number of principal components is chosen too large, the noise will also be assimilated in the model, as shown by figure 1b. On the other hand, a very low number of principal components will lead to bad specification/reconstruction of the data, depicted on figure 1c. The number of principal components is thus a trade-off. 5 seems to reasonably reconstruct the underlying data distribution, while effectively having reduced the noise (figure 1d). The measure to tune parameter upon could be based on the error (e.g. RMSE) between the original denoised data set and the data outputted by the KPCA. A part of the training set could herein be erected as validation set and methods like leave-one-out and cross-validation should be a good choice to alter parameters hyper and kernel parameters upon, for a number of principal components. As extra note: results of the KPCA's denoted in figure 1 are obtained with the default values of the script, and with the RBF kernel.



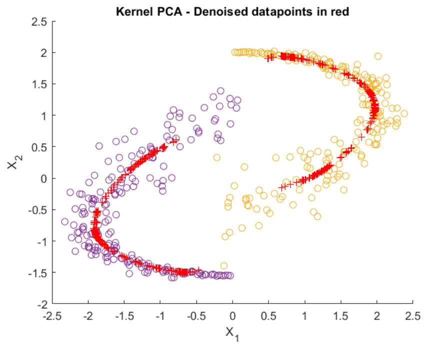
(a) Standard (linear) PCA, 1 principal component



(b) KPCA, 25 principal components



(c) KPCA, 2 principal components



(d) KPCA, 5 principal components

Figure 1: PCA and KPCA on the yin-yang data set

1.2 Spectral clustering

Spectral clustering can be interpreted as the weighted version of KPCA. The weights are equal to the inverse of the degree matrix of the graph. Solving the optimization problem is harder because finding the optimal centering is not as trivial anymore. In this part of the exercise, we will try to differentiate two clusters from one another.

To go more in depth; executing spectral clustering involves computing the RBF kernel (affinity) matrix which contains the correlations between the data-points. These correlations are in the kernel matrix in their complete dimensions (n by n, where n is the number of datapoints). The thing about spectral clustering is that it generates a (in this case) single-dimensional vector $sign[\alpha_i]$, with length equal to the size of the input data, based on the correlations, in such a way that the vector assigns a class to each data point (assign the data to their cluster). The single-dimensional vector is for the binary case. In the end, the optimization problem solved to obtain this vector minimizes the inter-cluster similarities. This is all done in an unsupervised way, so no prior information on the classes is necessary. Spectral clustering is tied with k-means clustering, as it could isolate non-linear clusters in the feature space, which is trivial for k-means clustering to cluster.

Spectral clustering for multiple clusters is also possible, and involves having more constraints on the optimization problem. In essence, the method is comparable to a one-vs-all classification for all number of clusters.

On the other hand, classification in SVM's involves a boundary in the same feature space is the data (of dimension a-1, where a is the number of dimensions of the input data). The training of such a classification SVM model is done in a supervised way. That is, training data is labeled and predefined to a certain class. Splitting the data into training,validation, and test set is crucial.

σ^2 is the RBF kernel parameters that determines the width of the kernel. σ^2 has an influence on the kernel matrix in general kernel PCA and spectral clustering because the kernel matrix/similarity matrix Ω_c can be written as;

$$\Omega_c = \begin{bmatrix} (\varphi(x_1)-\hat{\mu}_\varphi)^T(\varphi(x_1)-\hat{\mu}_\varphi) & \dots & (\varphi(x_1)-\hat{\mu}_\varphi)^T(\varphi(x_N)-\hat{\mu}_\varphi) \\ \vdots & \ddots & \\ (\varphi(x_N)-\hat{\mu}_\varphi)^T(\varphi(x_1)-\hat{\mu}_\varphi) & \dots & (\varphi(x_N)-\hat{\mu}_\varphi)^T(\varphi(x_N)-\hat{\mu}_\varphi) \end{bmatrix} \tag{1}$$

and each element of the kernel matrix can be described by;

$$\Omega_{c,kl} = (\varphi(x_k)-\hat{\mu}_\varphi)^T(\varphi(x_l)-\hat{\mu}_\varphi), \quad k,l = 1,...,N \tag{2}$$

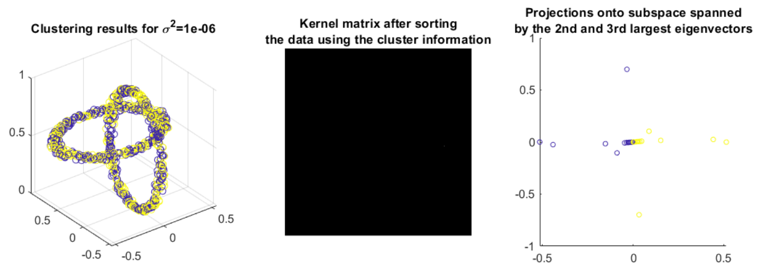
Each score variable is then equal to;

$$z(x) = w^T(\varphi(x)-\hat{\mu}_\varphi) \tag{3}$$

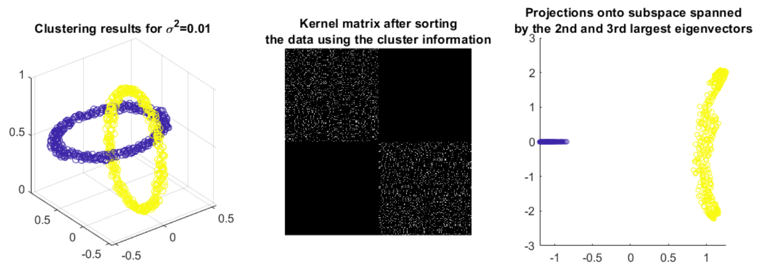
$$= \sum_{l=1}^{N} \alpha_l (\varphi(x_l)-\hat{\mu}_\varphi)^T(\varphi(x)-\hat{\mu}_\varphi) \tag{4}$$

$$= \sum_{l=1}^{N} \alpha_l \left(K(x_l,x) - 1/N \sum_{r=1}^{N} K(x_r,x) - 1/N \sum_{r=1}^{N} K(x_r,x_l) + 1/N^2 \sum_{r=1}^{N}\sum_{s=1}^{N} K(x_r,x_s) \right) \tag{5}$$

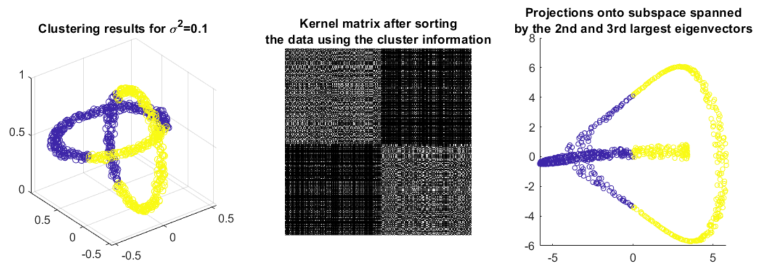
Knowing that $K(x,x_k) = e^{\frac{-||x-x_k||_2^2}{\sigma^2}}$ and keeping in mind equation 5 we can deduce that a very small value of σ^2 will impose that the scores and the elements of the centered matrix Ω_c will be close to zero. On the other hand, when σ^2 is very large, the whole matrix will contain 1's. In the best case, there should be no dissimilarity between data-points of a same cluster (values close to 1) and the the two non-diagonal blocks of the sorted kernel matrix should contain close to zero elements, indicating that the between cluster similarity is small/dissimilarity is big. Figure 2 shows the results for varying σ^2 values. In this case, $\sigma^2 = 0.01$ seems to be doing well and cluster the data well.



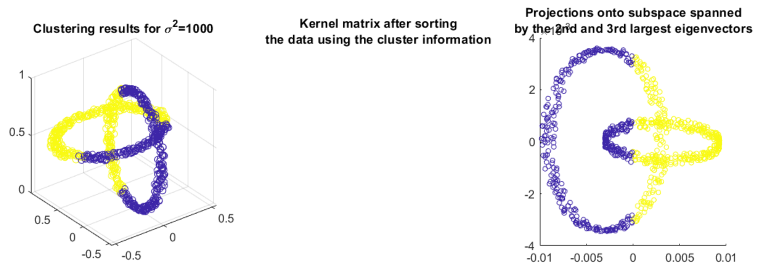
(a) $\sigma^2 = 0.000001$



(b) $\sigma^2 = 0.01$



(c) $\sigma^2 = 0.1$



(d) $\sigma^2 = 1000$

Figure 2: Spectral clustering with two clusters

1.3 Fixed-size LS-SVM

In general, problems where the number of data points is significantly higher than the number of dimensions, ought to be solved in the primal representation. The dual representation on the contrary is kernel based, and is nothing else than inner-product of data points. Each data point has then an associated an alpha value. The number of dimensions does not affect the dual problem in any way, so should be preferred to solve a problem which is very high dimensional in comparison with the number of training data (such as micro-array data). An aspect that also needs to be taken into account is the choice of kernel function. When for example the feature map is infinite dimension, you can always solve the problem in the dual. Another way of handling large data sets lies in the use of fixed size LS-SVM's.

Because the eigenvalue decomposition increases sharply with the size of the feature space, finding a way to reduce the space without loosing to much information would benefit computation times and memory in the primal. The fixed-size LS-SVM is a method reaches these goals by taking a fixed amount (M) of points, which is a subset of the original data set. Picking M points at random could in some cases be enough, but the risk exists that the M data points do not fully represent the distribution of the original data set. In order to solve this problem, the quadratic Renyi entropy can be used as a measure to detect whether a randomly chosen extra point a holds more information than a random chosen point b of the subset M. If that is the case, b is replaced by a. Multiple iterations of this procure lead to a data set M that resembles quite well to the original one. The quadratic Renyi entropy is calculated as in equation 6.

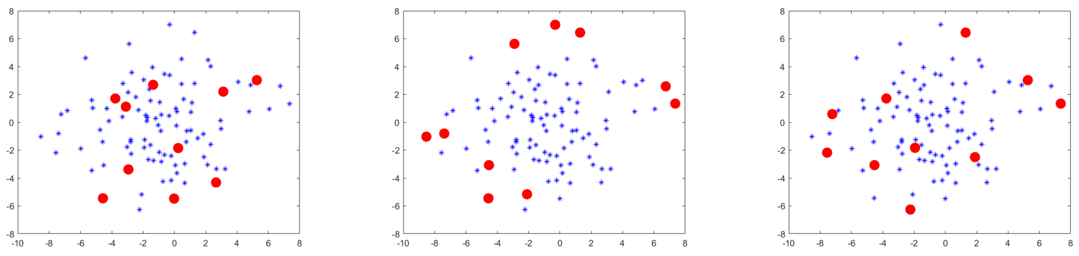
$$H_R = -log\frac{1}{M^2}\sum\nolimits_{ij}\Omega_{(M,M)_{ij}} \tag{6}$$

where $\Omega_{(M,M)_{ij}}$ is the kernel matrix of the M space. The eigenvalue decomposition of this matrix is shown in equation 7.

$$\Omega_{(M,M)}\bar{U} = \bar{U}\bar{\Lambda} \tag{7}$$

where $\bar{\Lambda}$ contains the eigenvalues corresponding to the eigenvector matrix $\bar{U}$. In essence, working with the subset M involves working with an approximation of the original feature map, obtained through the Nyström method (downsampling of the original NxN to the MxM space). It must be noted that with the Nyström method, we lose the sparsity property.

In figure 4a, the entropy of the converged subset of the data is shown, for different σ^2 values. Low σ^2 values again impose that the similarities between two points (that could be very close to each other) are extremely low (kernel matrix full of zeros). The entropy is therefor quite rapidly maximal and does not give a good estimation of how much entropy the subset actually contains, as shown on figure 3a. High σ^2 values cannot differentiate between points that are close or distant to one another as they are all regarded to as similar (kernel matrix full of ones). Because the subset of the points in this case will lie at the boarders of the original data set, maximal entropy is low (shown on figure 3a). Picking a low value of σ^2 seems to be favorable based on the previously mentioned figure. Even though we reach an optimal entropy, the results are not really good for extremely low values for σ^2. Trough the iterations, the maximum entropy is reached quite rapidly (see figures 4b and 4c), in comparison with higher σ^2 values. The model is too quickly happy with its results and is not very flexible to possibly better points that are in later iterations being proposed. The best σ^2 values are thus the ones that are still maximizing the entropy while being as high as possible, more specifically in the range of $[0.1, 1]$, as exemplified in figure 3c. Using the



(a) Result after 100 iterations for $\sigma^2 = 0.000001$

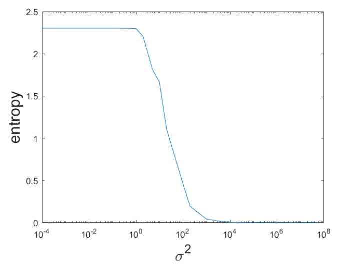
(b) Result after 100 iterations for $\sigma^2 = 100000$

(c) Result after 100 iterations for $\sigma^2 = 0.1$

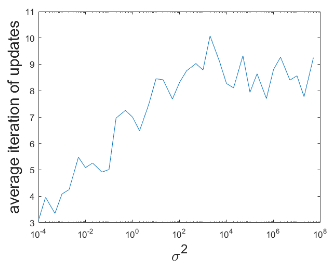
Figure 3: analysis of fixed size LS-SVM

Nyström approximation method, the mapping of data to the feature space can also be evaluated explicitly. This 'Automatic Feature Extraction by Nystrom method' gives the features that one can use for a linear regression or classification. The decomposition of the mapping to the feature space relies on the eigenvalue decomposition of the kernel matrix. In figures 5a and 5b, the mapping is shown in the first three dimensions of the feature space, calculated for the extracted points/features (that's why the red dots are different from the origin, as opposed to many other points/features which are not extracted). The extracted features are the result of picking 10 random points of the data set in the first image, and the points in the original space of the second space are the ones calculated with the entropy method described above.

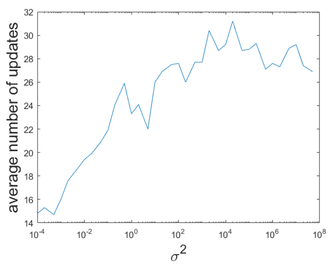
As mentioned already, a serious drawback of LS-SVM models is the lack of sparseness, as nearly all patterns become support vectors. Two kinds of solutions have been looked into; 1) pruning after



(a) Maximum entropy reached after hundred iterations for different σ^2 values

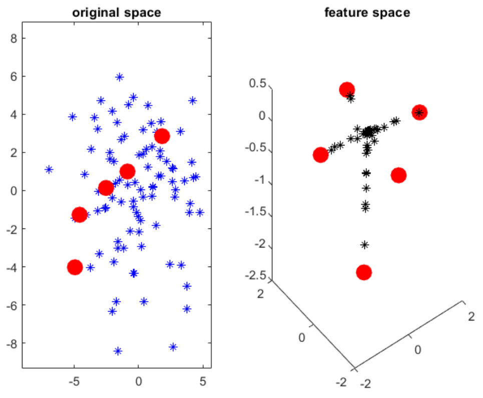


(b) Average iteration on which updates occurred for 100 total iterations, averaged over 10 runs

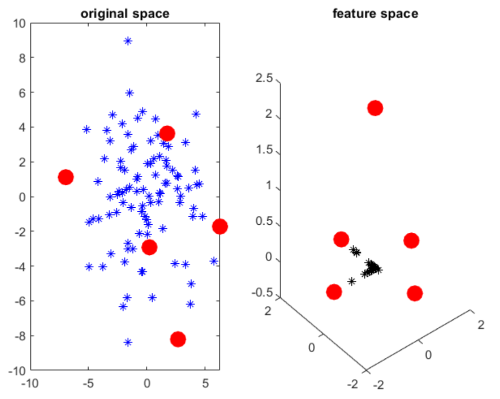


(c) Average amount of updates for hundred iterations, averaged over 10 runs

Figure 4: analysis of fixed size LS-SVM



(a) 10 random points



(b) The 10 obtained points after using the fixed size method with the quadratic Renyi entropy

Figure 5: Visualising the feature space

training and then retraining, and 2) enforcing sparseness from the beginning. An advantage of the first solution is that it is not guaranteed that the numberof SVs will be greatly reduced. In the second solution, the L_0-norm has been receiving increasing attention by the Machine Learning community. It counts the number of non-zero elements of a vector. Therefore, minimizing it leads to very sparse models. Unfortunately, this cannot be done in a straightforward manner, since the resulting problem is NP-hard [1]. To handle this problem, some approximations to it are looked into [2].

In this method, the parameter k is a factor used to determine the number of representative points M by heurisitic $k\sqrt{N}$ where N = dataset size. For the comparison of the results of fixed-size LS-SVM to the SV_L_0_norm-approximation in terms of test errors, number of support vectors and computational time, I'd like to refer to the homework problems on Fixed-size LS-SVM's.

2 Homework problems

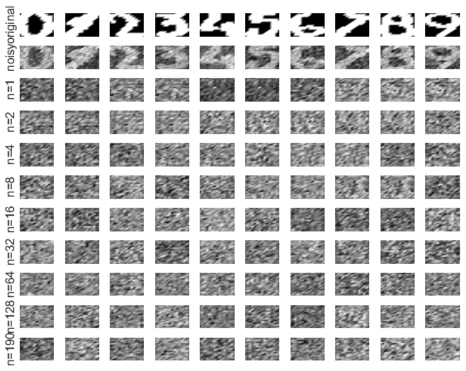
2.1 Kernel principal component analysis

Densoising digits on which Gaussian noise is added consists of the extraction of the main components of a pattern and not including the less significant ones. More technically, this corresponds to the matrix decomposition into eigenvectors and eigenvalues of a covariance matrix. In the case of a LS-SVM approach to a kernel PCA, the matrix to be diagonilised is the one of equation 1. Each element of the matrix Ω_c is a relative measure of the distance between two data-points x_a and x_b compared to the rest of the data-set (see also equation 5). This value will thus be high if x_a and x_b are closer to each other (due to the negative exponent of e). Ω_c can thus be interpreted as an enhanced covariance matrix of the data. Analogous to the reasoning in subsection 1.2, if the σ^2 is very low, the resulting measure of distance will be very strict. In order to detect correlation between two data points, the data points should be extremely close to one another. This will result in very few correlated point and an overall constant Ω_c. After calculating the matrix decomposition, the small number of principal components with a non-zero loading will correspond to very specific features that could very well not be shared by all the digits of the same class. We should thus be aware that σ^2 should not be chosen too low.

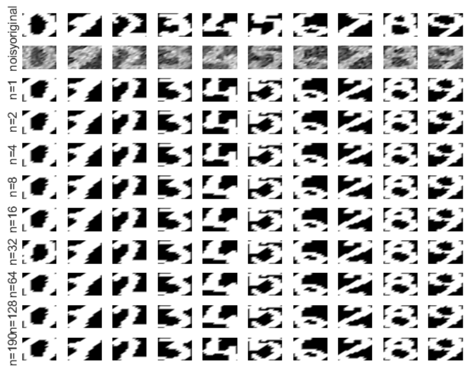
On the other hand, when the σ^2 is too high, all point will be considered close to each other, without much discrepancy. The resulting Ω_c's values will thus be quite constant and high. The resulting principal components will correspond to very broad features often shared by many classes of digits. In this case, these PC's will not be able to specify the important features over the unimportant ones (such as noise). In the figures entailed by figure 6, different σ^2 values are wielded for a noise factor of 0.3. Indeed, when decreasing the σ^2 value, only very clear components of the digits are found, while other less pronounced patterns are neglected. Decreasing σ^2 even more results in the distance measure being so strict that no patterns are found and the reconstructed images consist of random noise only (also, Matlab sputters : "Starting value changed"). Increasing σ^2 will over-generalize and eventually result in noisy reconstructed digits as well, as the KPCA cannot keep the importance of the noise and the underlying non-noisy features from each other.

In second investigation, regular PCA and KPCA are put next to eachother, for a gaussian noise factor of precisely 1. Results of the reconstruction of the digits using PCA are shown in figure 7a, KPCA's reconstruction results are shown in 7b. In both cases, 1, 2, 4, 8, 16, 32, 64, 128, and 190 PC's were used. It's clear that KPCA manages to "memorize" the digit's features without the noise. Due to the high noise, reconstructing the digits does not always happens correctly (seven is reconstructed as a two and three as a five for a higher number of PC's). It is also perceived that increasing the number of PC's for KPCA makes the picture more and more clear. Though it must be noted that the first 8 PC's in this case of linear PCA seem to depict some general digit's features and not the noise, it never manages to reconstruct the images well enough. The rule of thumb that was wielded to get to the value of σ^2 is equal to a constant (0.7) times the number of dimensions (240) times the mean of variances for the pixel values for each digit in the data set, which results in $\sigma^2 = 35.9078$.

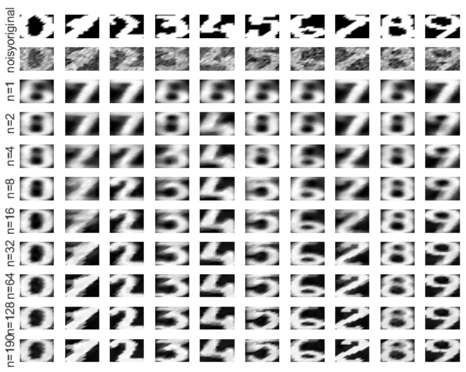
Tweaking of the parameter σ^2 could also be done based on the RMSME on the training and validation set. The RMSME is the RMSE where the error is equal to the mean of the errors for every pixel of an image. In figures 8a, 8b and 8c, the results on test and training set are depicted for different σ^2 values, for a noise level of 0.3 and taking the first 4, 16 and 64 PC's into account, respectively. The optimal σ^2 tends to increase and the RMSME tends to decrease with an increasing number of components. Again, we perceive that for low values of σ^2, there is overgeneralisation, which is why the training set error is quasi non-existent. For too high σ^2 values, training and validation set rise, as the reconstructed images are extremely noisy. The best points are reached in the minima of the validation set. For the case of 64 PC's, the reconstructions are shown in figure 9a, and seems to be of plus minus equal quality, in comparison with the thumb rule.



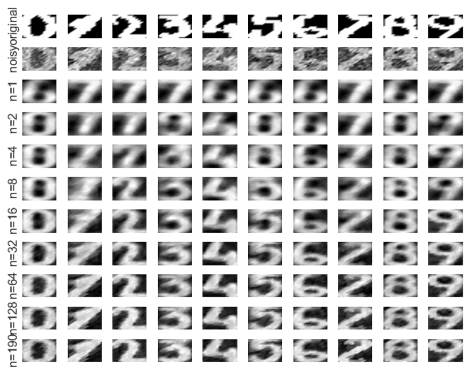
(a) $\sigma^2 = 0.01$



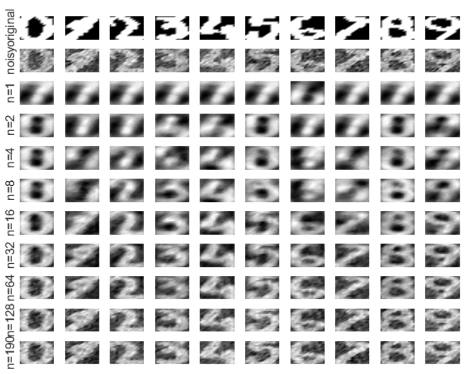
(b) $\sigma^2 = 1$



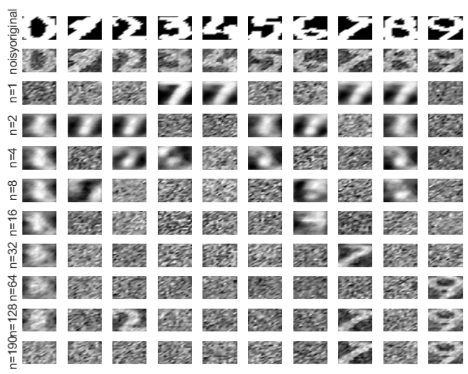
(c) $\sigma^2 = 35.9078$



(d) $\sigma^2 = 100$



(e) $\sigma^2 = 1000$



(f) $\sigma^2 = 10000$

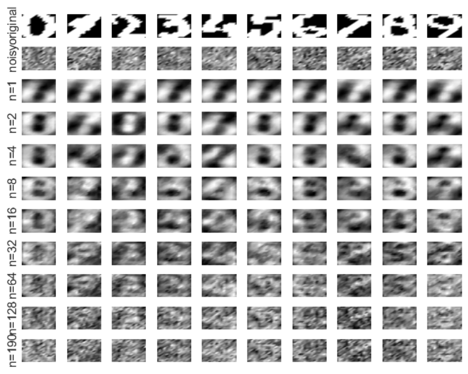
Figure 6: Reconstructing noisy digits using KPCA for different σ^2 values

2.2 Fixed-size LS-SVM

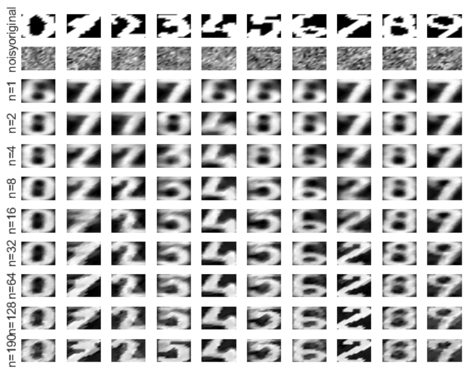
2.2.1 Shuttle (statlog)

The Shuttle data set containst 58000 tuples with 9 explanatory numerical variables. There are 7 classes in total and approximately 80% of the data belongs to class 1. Therefore the default accuracy is about 80%. The aim proposed by the authors of the data set here is to obtain an accuracy of 99 - 99.9%.

As this problem is a multi-classification problem, and the FS-LSSVM toolbox only implements



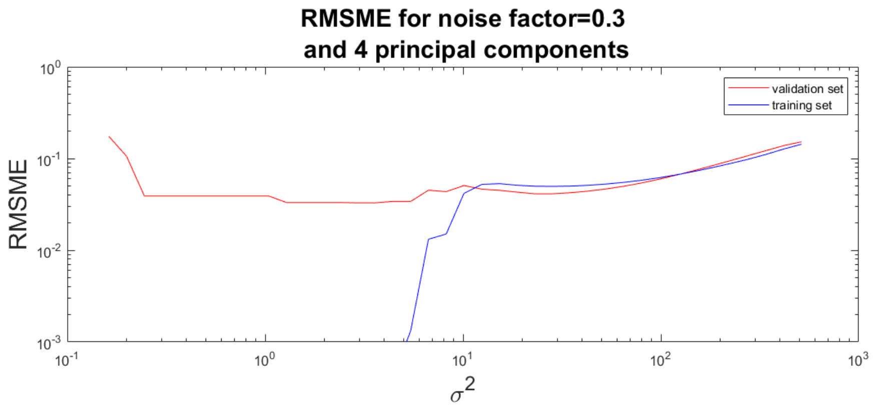
(a) PCA



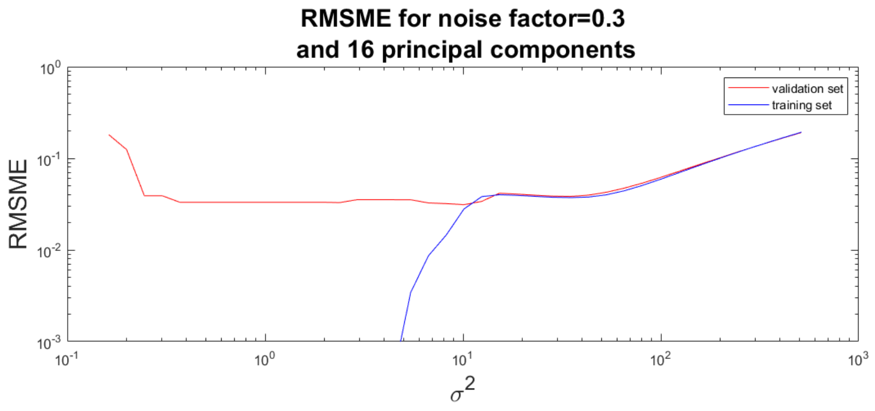
(b) KPCA

Figure 7: Reconstructing noisy digits (gaussian noise=1)

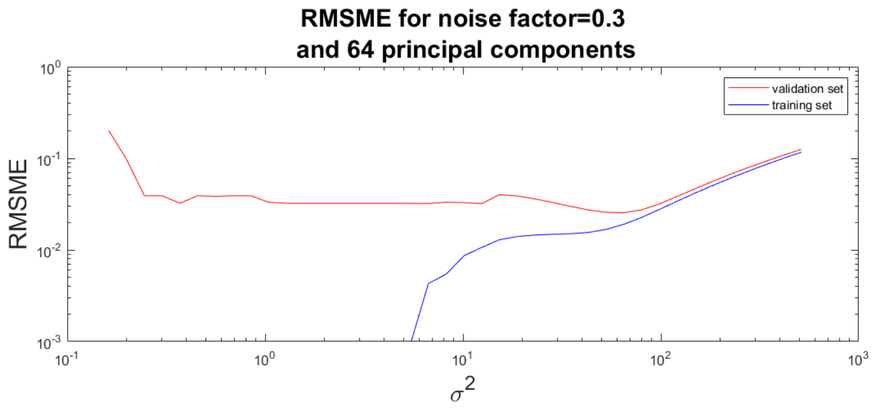
binary classification it is not ideal. I felt that implementing the multi-classification problem to the toolbox was not part of the scope of this assigment, but rather stating and interpreting the differences between the FS-LSSVM and the SV_L_0_norm. The SV_L_0_norm performs an L_0_norm over the Nyström approximated support vectors and incorporates information about the entire dataset in the kernel matrix while just using the support vectors for training. I changed the classes so that the biggest class (1) forms a class on its own, and all others (2-7) are also combined. For computation time purposes, the linear kernel was wielded and only 10000 points were used. 80% thereof were randomly assigned to be part of the training set whilst the other 20% are part of the validation set. In this case, I opted for the linear kernel and Randomised Directional Search as the latter runs in parallel and the combination would benefit computation times, which did not seem to work as the *ds* option crashed the lssvm toolbox. Instead, I used the linear kernel in combination with Coupled Simulated Annealing (which runs sequential). Figures 10, 11, and 12 show the results for computation times, error rates and number of support vectors for both methods. Computation times rise with the number of k, as increasing k imposes a bigger data set that needs processed. The number of support vectors is equal for both norms. Computation times do not differ much between the two methods and it seems that FS-LSSVM is superior as it involves a lower error rate. For different k values, only the error rate of the SV_L_0_norm seems to change slightly (highest error rate of 3.6% for k=5).



(a) 4

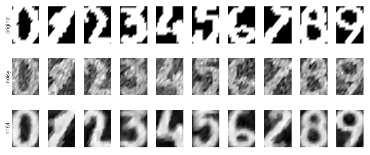


(b) 16

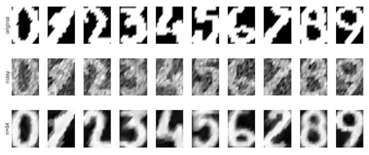


(c) 64

Figure 8: RMSME for different number of PC's

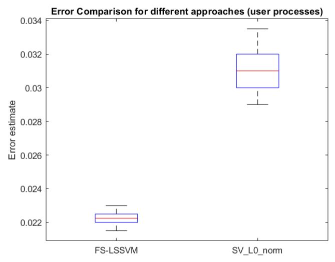


(a) Mimima of validation set for 64 PC's

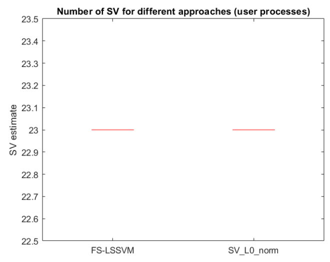


(b) Thumb rule

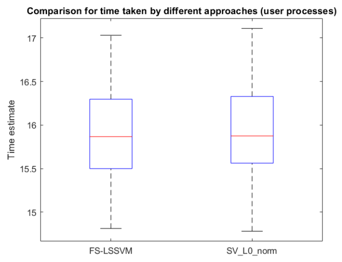
Figure 9: reconstruction for different values of σ^2



(a) error rates

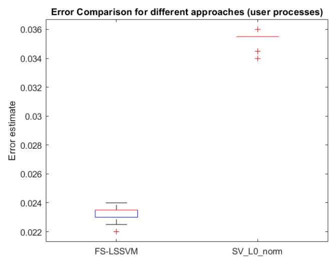


(b) number of support vectors

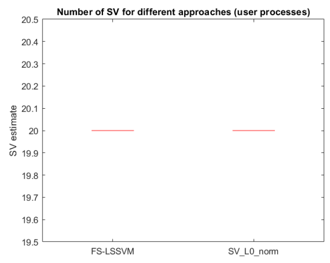


(c) computation times

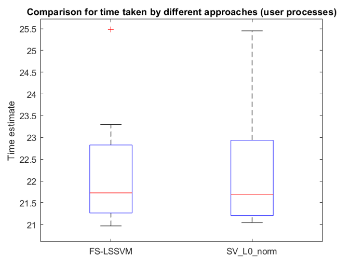
Figure 10: Results when k equals 2



(a) error rates

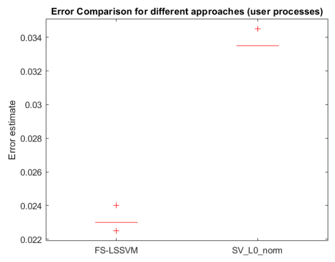


(b) number of support vectors

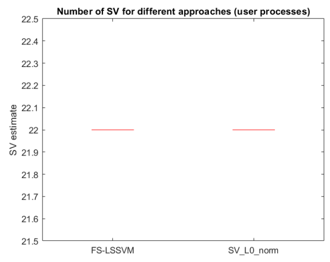


(c) computation times

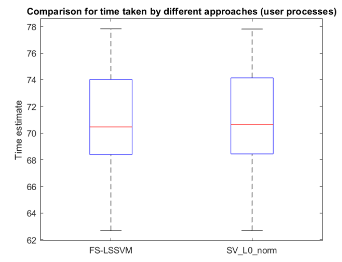
Figure 11: Results when k equals 5



(a) error rates



(b) number of support vectors

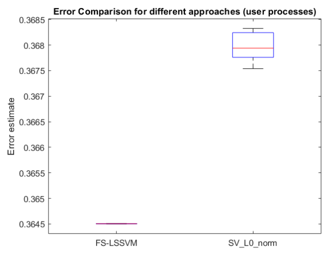


(c) computation times

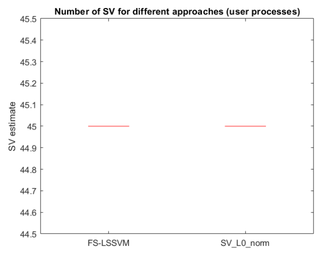
Figure 12: Results when k equals 10

2.2.2 California

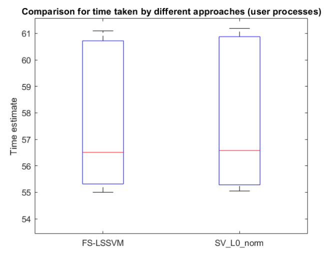
The regression problem of the California Housing dataset, which contains 20460 tuples, is stated as finding the median house prices (more specifically ln(median house value)) for California districts derived from the 1990 census. There are 8 continuous explanatory variables and one binary variable which is omitted in the dataset we've gotten. The eight explanatory variables include longitude, latitude, house median range, total number of rooms, total number of bedrooms, population, households, median income, and median house value. Again, 80 percent of the data is used as training set, the remainder as validation set. CSA was again used as search method. Data normalization was carried out as the ranges of the features are quite different. For the linear kernel, the results were not of superior quality (see figures 13 and 14). Therefor, also the polynomial kernel was looked into, which involves higher computation times but improves the accuracy (see figures 15 and 16). There seems to be no difference anymore in number of SV's in these cases.



(a) error rates

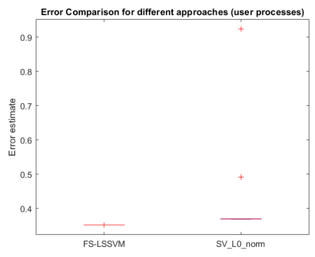


(b) number of support vectors

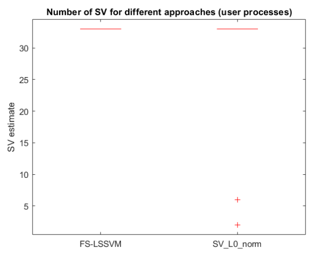


(c) computation times

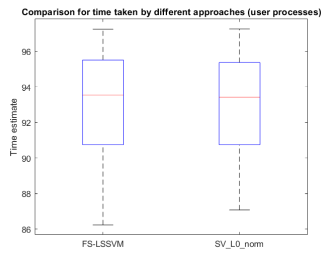
Figure 13: Results when k equals 4, linear kernel



(a) error rates

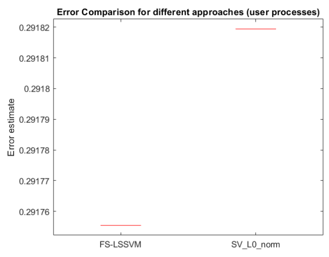


(b) number of support vectors

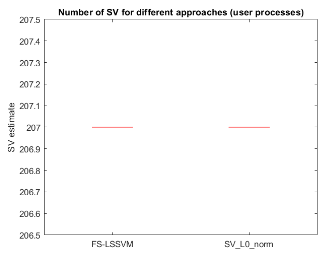


(c) computation times

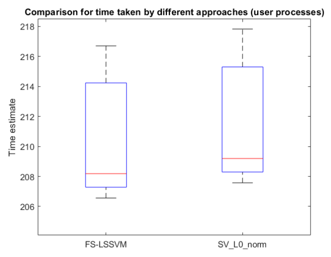
Figure 14: Results when k equals 8, linear kernel



(a) error rates

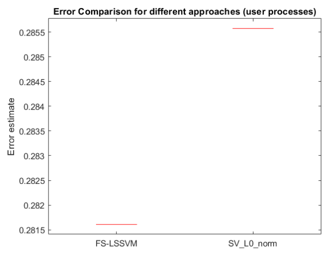


(b) number of support vectors

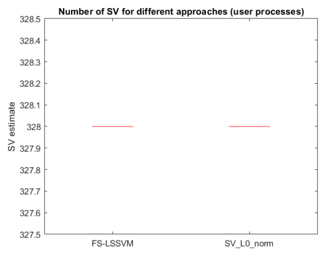


(c) computation times

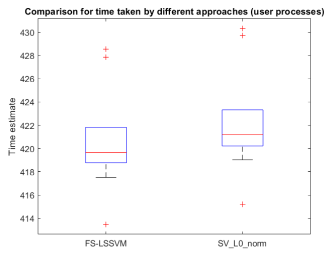
Figure 15: Results when k equals 2, polynomial kernel



(a) error rates



(b) number of support vectors



(c) computation times

Figure 16: Results when k equals 4, polynomial kernel

References

[1] Jorge Lázaro, K De Brabanter, José Dorronsoro, and Johan Suykens. Sparse lssvms with l0-norm minimization. *ESANN*, pages 189–194, 01 2011.

[2] Jason Weston, André Elisseeff, Bernhard Schölkopf, and Mike Tipping. Use of the zero-norm with linear models and kernel methods. *Journal of machine learning research*, 3(Mar):1439–1461, 2003.