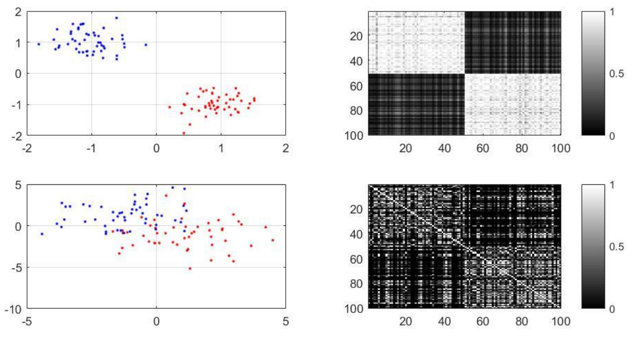
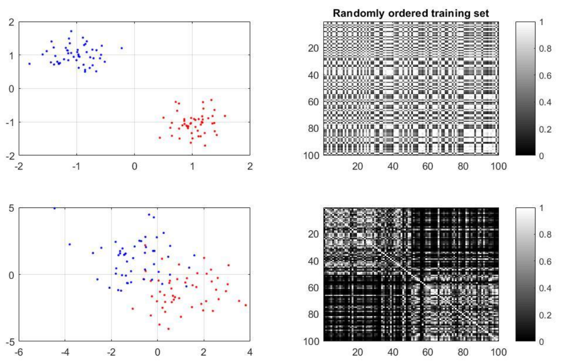
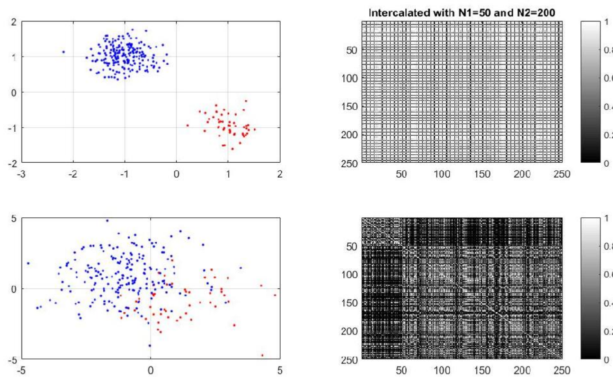
Checkpoint 10.1

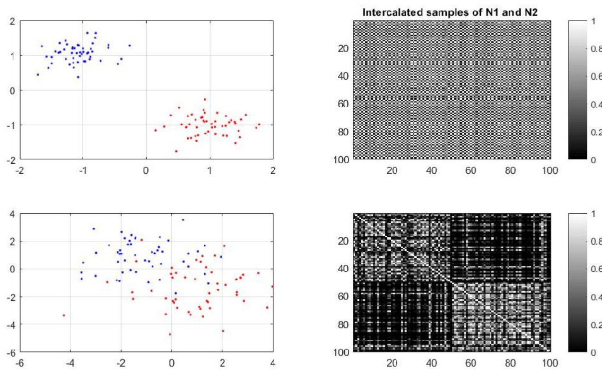
Here we analyze the structure of the kernel matrix. First let us inspect the gaussian kernel matrix for a simple two cluster simulated data set in $d = 2$. Run the script `main10a.m` and discuss the structure of kernel matrix for these simple simulated data sets. How does the width of the Gaussian kernel affect the kernel matrix? Imagine the kernel matrix columns as new "pseudo-features", how well do they separate classes? Next we investigate the kernel matrix for the case of the sunspot data. Run script `main10b.m` to illustrate the kernel matrix for simple case of 2-dimensional history $d = 2$. Explain the kernel matrix used for predicting targets for test data. Explain the patterns you see in the training and test kernel matrices and relate them to the sunspot time series.

When we take a look at the structure of the kernel with a normal configuration of the program we see two clearly differentiated parts, one representing points really close to each other and another representing points greatly separated. This holds with the distribution that we can see in the left side. In this case it is clearly identifiable when a point is part of one distribution or another, with a pseudofeature vector with values [~1 ~1 ~1 ... ~0 ~0 ~0] for the first distribution and the other way around for the other one. However, we can also observe this pseudofeatures vectors when we look at the kernels which order have been altered. Let's take for example the kernel in which I intercalated the two training sets. In this case, one of the classes will be represented by the vector [~1 ~0 ~1 ... ~0 ~1 ~0] and [~0 ~1 ~0 ... ~1 ~0 ~1] for the other class.

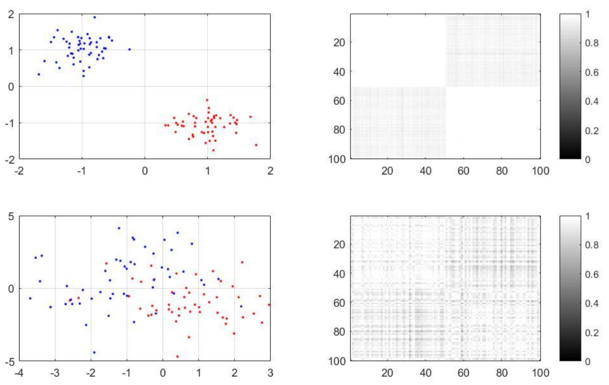


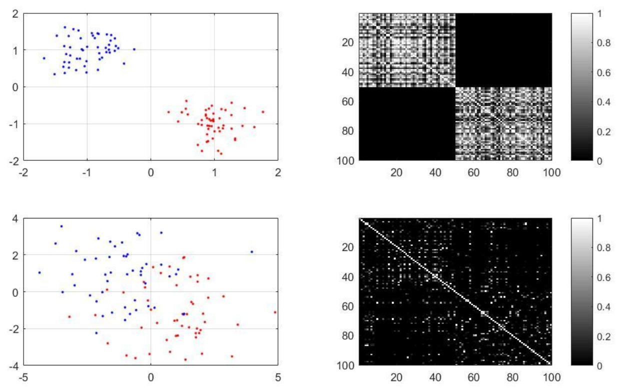




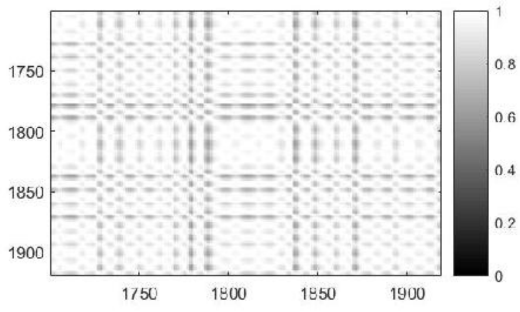
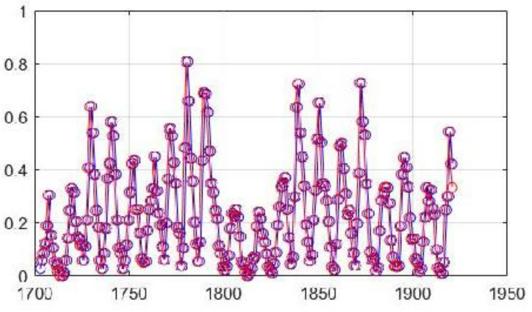
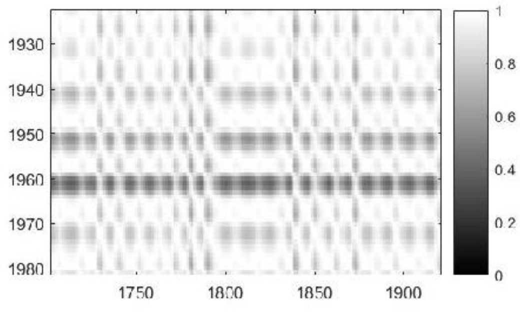
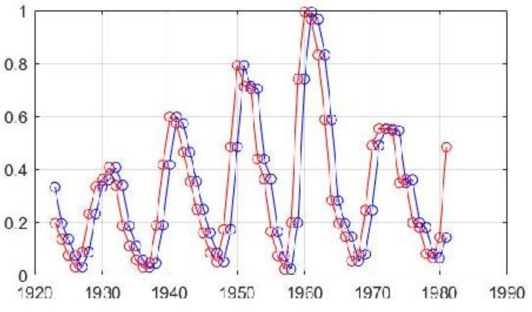


Respect to the influence of sigma in our model we can note that with large values of sigma the probabilities of being part of a kernel are reduced, even though the distance between points is the same. On the other hand, when we reduce sigma too much the model struggles seen the similarity of points that are slightly further away, as it is the case with the second dataset, which is more spread out. In this case it is more difficult to assess the similarity between pseudofeature vectors than in the cases with a regular sigma.

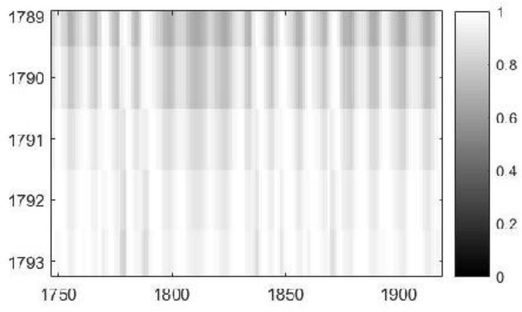
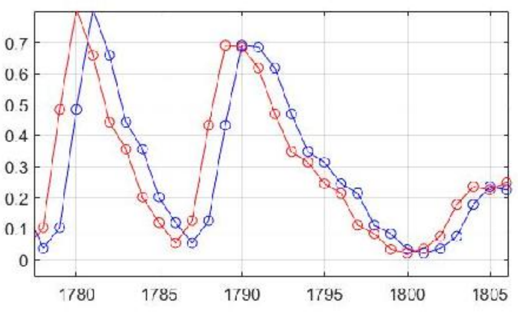
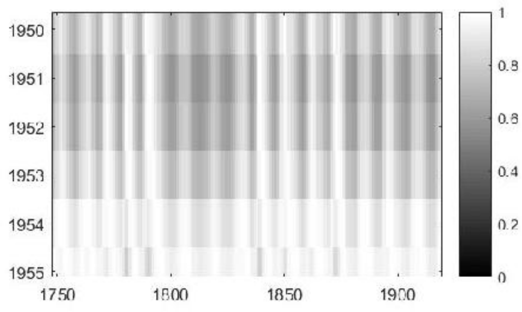
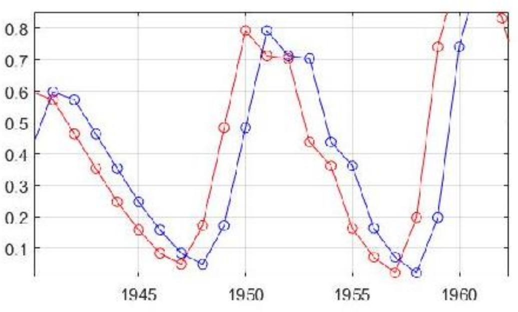




Now we will investigate the kernels with real data, using the sunspot dataset that we are already familiar with:

From this kernels we can see different patterns. For example, we can clearly distinguish the years in between 1800 and 1825, which sun activity was unusually low. This can be compared with the years with high activity (1960) and we note that the similarity between them is close to zero. However it is better to zoom in to be able to compare the years better:

In this case there are two consecutive years that are quite similar at different periods of time. On the one hand we have the years 1789-1790 and on the other hand the years 1951-1952. The similarity cn be apreciated from the kernel representation in two ways. Firstly, if we compare the pseudofeature vector between these two years we encounter a lot of similarities, even in the scale of similarity. Secondly, and just centering our atention on the second kernel we can see that the similarity between the year 1951 and 1790 is really close to 1.

Checkpoint 10.2

We apply a simple Gaussian process (GP) model to the prediction of sunspots. Run the script `main10c.m` to optimize the two parameters (β, σ^2) of the kernel given by $(\mathbf{C})_{m,n} = e^{-\|\mathbf{x}_m - \mathbf{x}_n\|^2/2\sigma^2} + \beta^{-1}\delta_{n,m}$. Explain the relation between the conditional mean in eq. (9) above and the test set predictions. How do we optimize parameters?. What is the difference between the two estimates of the test data in figure 2 (green and blue)?. How well does a GP with a simple Gaussian kernel work? Comment on the quality of the posterior uncertainty estimate, cf. eq. (11).

The conditional mean presented in (9)is the most probable mean for the test set for a given training set. This is in fact the mean of the test set predictions.

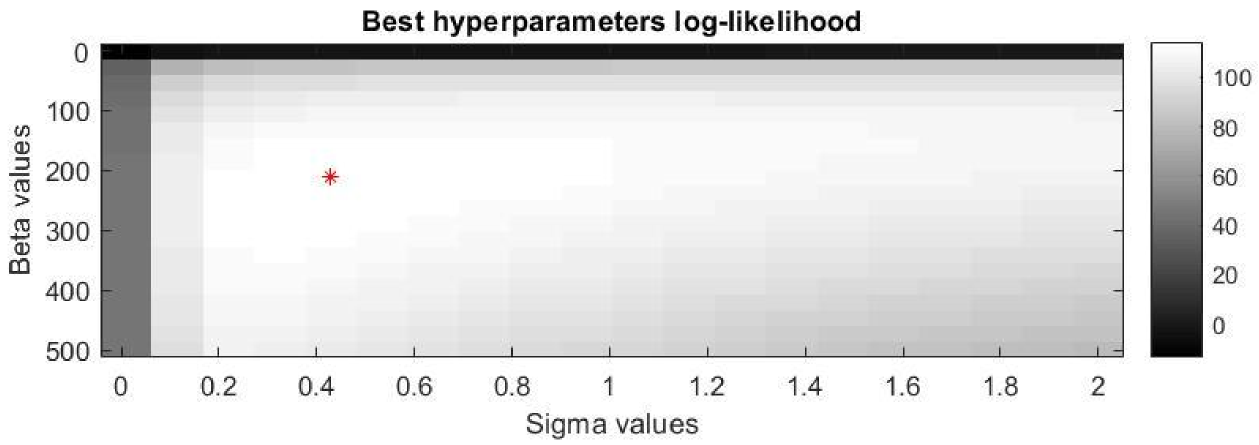
The optimization of the hyperparameters is done using brute force. We create an array with different values of beta and different values of sigma and then we apply the gaussian process model to our dataset. After each iteration we will compare and save the parameters of the best result, by the means of two different measures. The first error is the log-likelihood, which is the standart for a multivariate Gaussian distribution, maximazing this way the probabilities of matching the targets given all the set of hyperparameters.

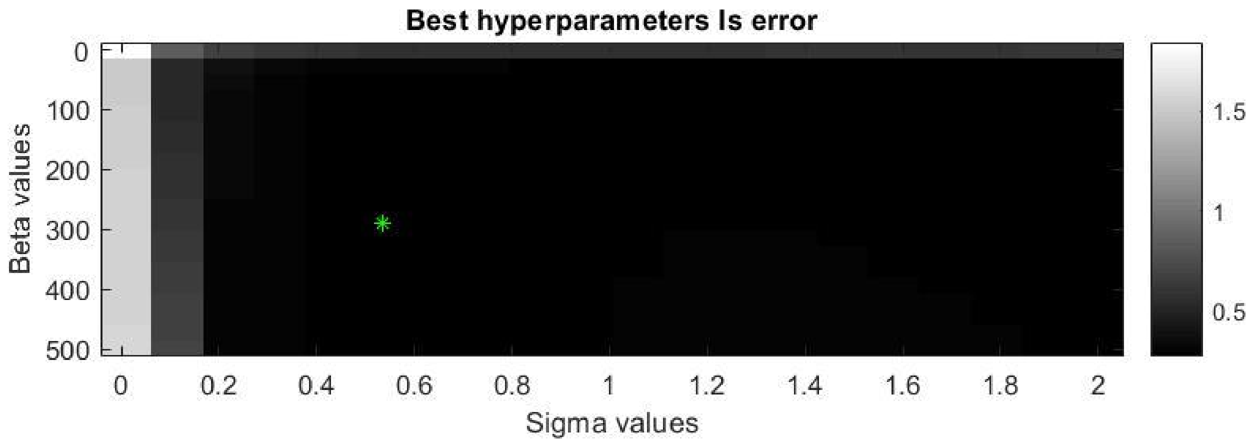
$$\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2}\ln|\mathbf{C}_N| - \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} - \frac{N}{2}\ln(2\pi).$$

The other measure is the sum of squares error divided by the variance of all the dataset.

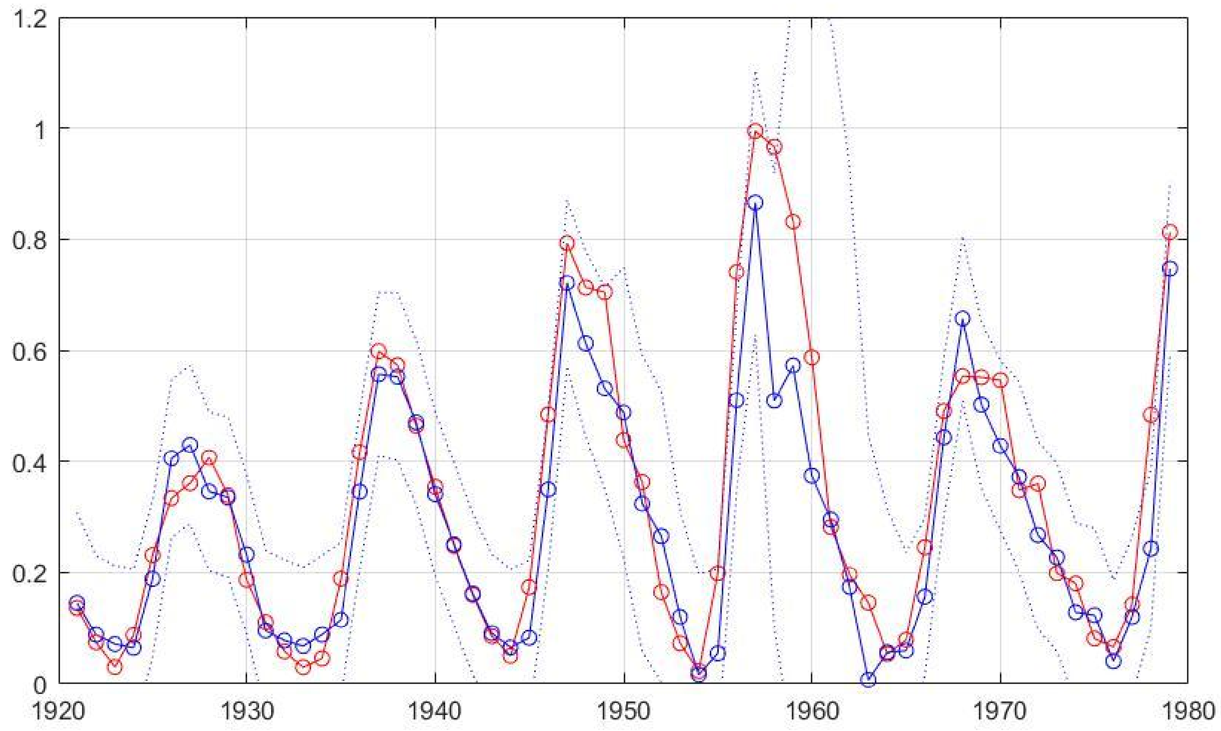
$$\frac{1}{\sigma^2}\frac{\sum_{n=1}^{N}|t_{\text{predict}} - t_{\text{test}}|^2}{N}$$

We can see the results from both measurements. First of all, we have the results of the hyperparameters using one metric or another. The results are pretty similar, and the values map shows similar results as well. When we take a look ac the actual results using both set of hyperparameters the results are equally similar. On red we have the target values, on blue the results with the hyperparameters out of the log-likelihood selection and on green out of the ls error.



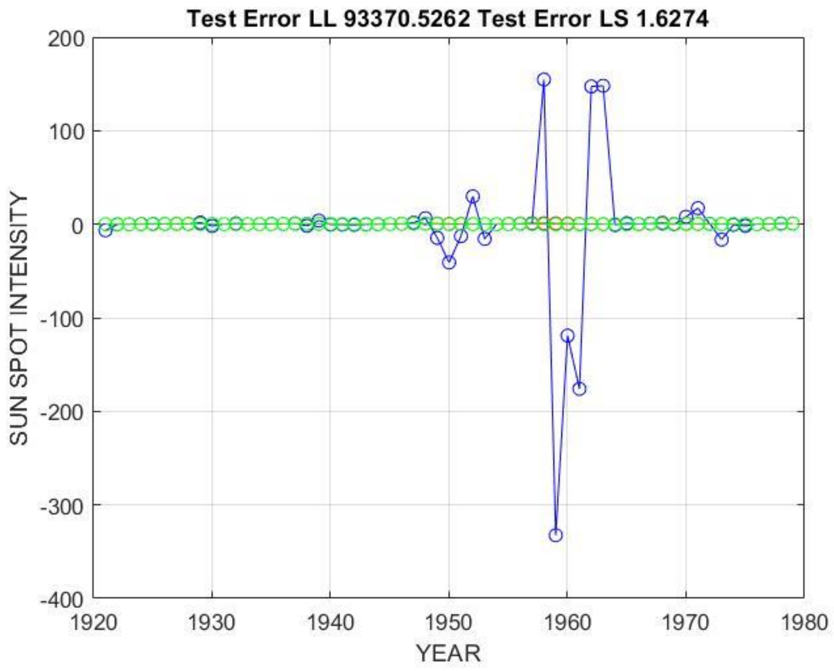


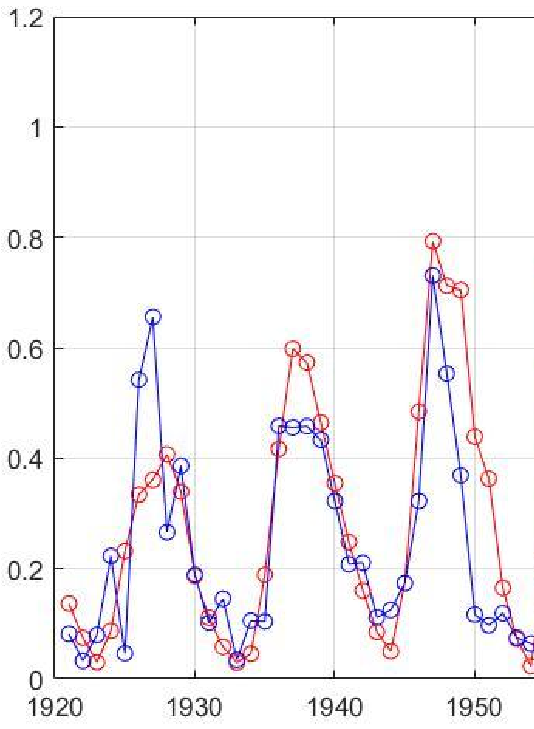
We will make now use of the probabilistic nature of the gaussian process and plot the uncertainty estimate as 2 times the std of the target value



On the first years the GP is able to keep the uncertainty pretty low, specially when the number of points grow. However, it is interesting to take a look at the results when the target value is different than everything we have in our dataset. This is the case around the year 1956. We can note that the value in this year is higher than any other year in our dataset. This causes the uncertainty in the next year to grow, but it definitely grew substantially once that prediction didn't hold and the number stayed high for two years. After that the gaussian was able to recover properly, partially because the number of dimensions is just 2.

Finally, let's check how does the model works with a simple gaussian kernel, i.e. without gaussian noise.





We can see that the results are notably bad, specially in the case that we select the optimal value of sigma in terms of the log-likelihood. This is the case because our kernel matrix is not ensured to be invertible anymore, which causes nnumerical issues later on. If we choose our results based on the ls error our results are slightly better, but they are still far from the case with added noise

Checkpoint 10.3

Compare the SVM decision function in eq. (12) with the nearest neighbor voting scheme discussed in Exercise 9. In the scripts `main10d.m` and `main10e.m` we use the simple Gaussian kernel function as earlier

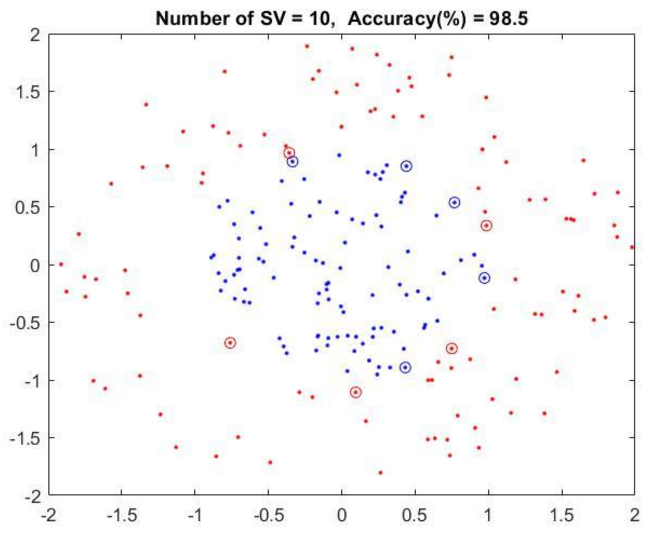
$$K_{m,n} = e^{-\|\mathbf{x}_m - \mathbf{x}_n\|^2/2\sigma^2}. \tag{13}$$

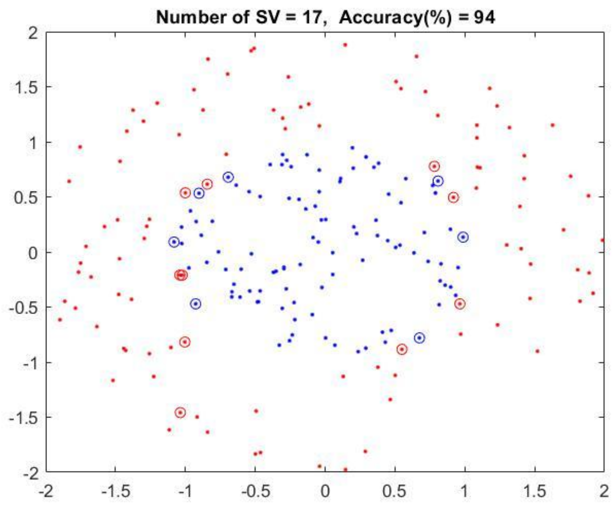
First analyze a 2-dimensional synthetic data set with two classes. Run `main10d.m`. Inspect the solution given in the Matlab structure `SVM`, what is the number of support vectors?

Next, we analyze the Pima indian data with the script `main10e`. We optimize the parameters of the support vector machine by maximizing the accuracy on the test data. How many parameters are optimized? How many support vectors do you get?

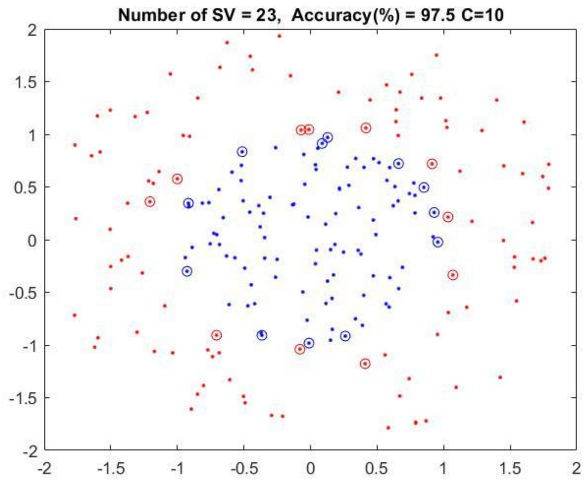
Consider classification from a subset of the seven input variable measures. Estimate the performance for a few subsets, can you find a subset with performance equal or better than that of the full set?

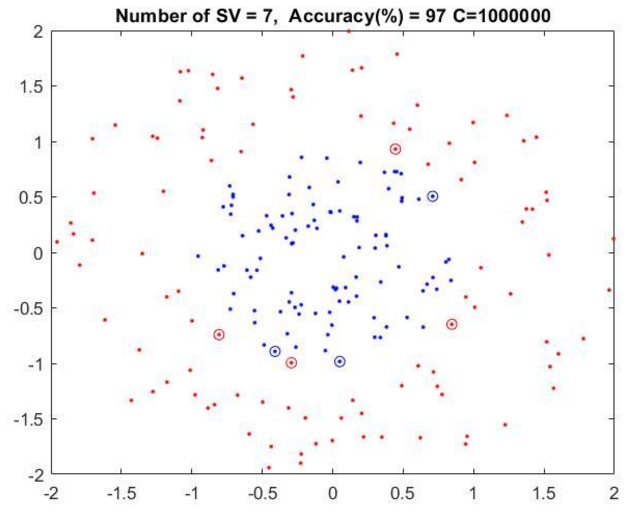
The SVM decision function keeps some similarities with the Nearest Neighbours algorithm. In both cases we have a voting algorithm, in this case based on the Kernel similarity function, and in the case of the KNN using the distance between points to select the points that will be part of the votting. Thus, in the SVM all the points of the training set are able to vote, but its weight is reduced, as the similarity out of the kernel will be small.



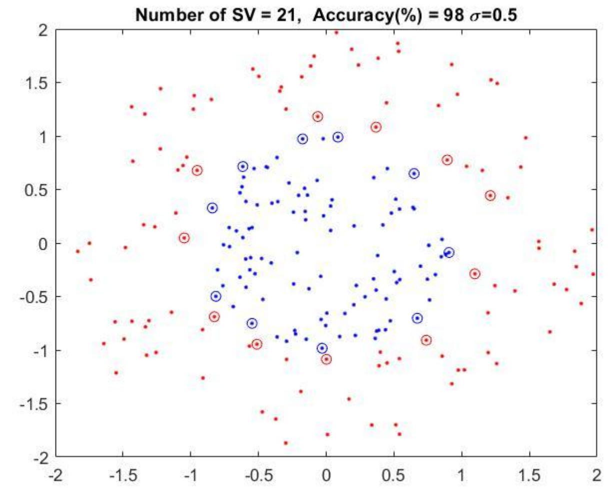


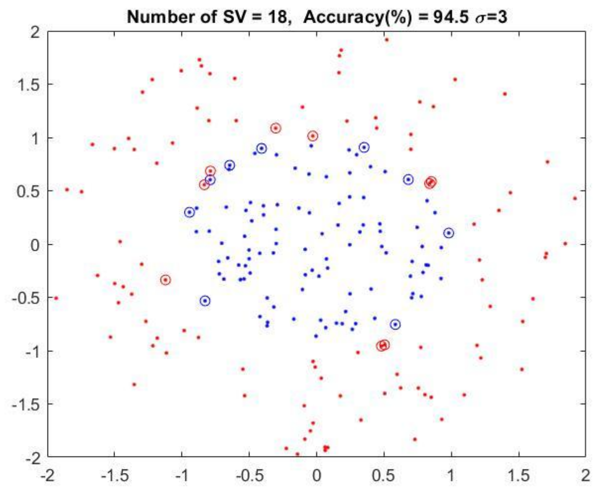
After running this SVM classifier on the syntetic dataset we obtained a total of 10 SV. However, the number of SV can be altered by different means. First, we tried to introduce some noise into the model, which gave us a total of 17 SV, and a small drop in accuracy. This is an expected behaviour, as it is more difficult to draw the hyperplane between classes.





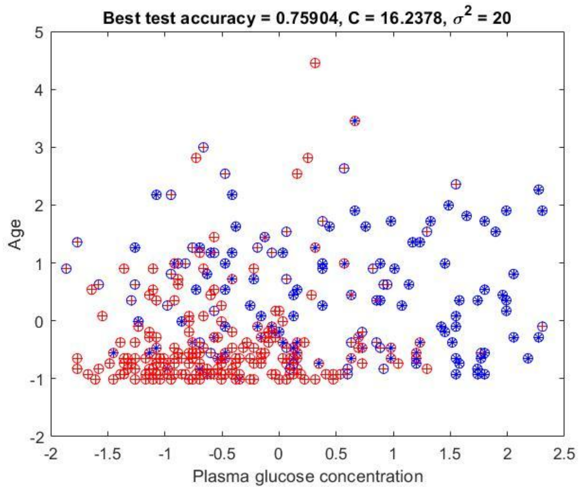
Another parameter that affects the number of SV is C. In the first image we have constrained to be quite small, which would allow for less numbers in the boundaries of the hyperplane, making the number of SV grow. On the other hand we have a less restricted C, which makes the model to reach a solution with less SV.

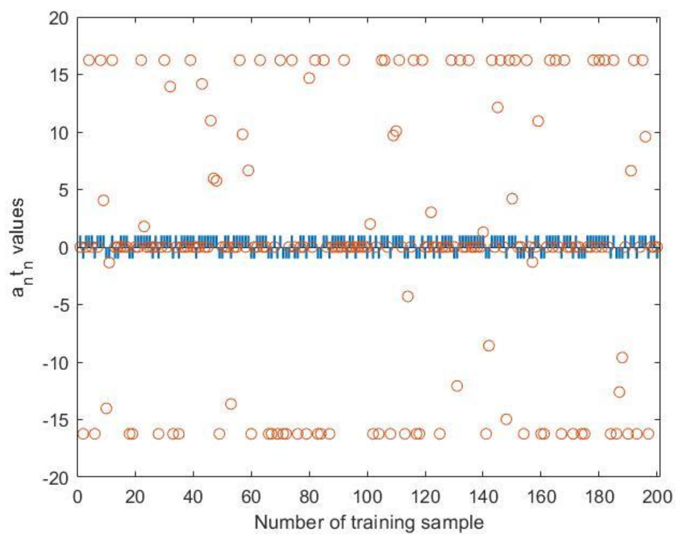




Finally, we have tried with different kernel sigmas. The first experiment show 21 SV when the sigma was reduced to 0.5, which is explained by the fact that the kernel introduces also other points not that close to the kernel. On the other hand we have the case when sigma is bigger, where we achieve 18 SV. In this case the gaussian includes less points, which makes it more difficult to asses points as similar. This explain the number of points that really close to each other.

Now, let's analyze the performance of the SVM using some real data. As in the previous exercises we will make use of the Pima indian dataset. In this exercise we are optimizing the vvalue of sigma and C by iterating over a set of values for each parameter and selecting the one that gives us more precission. When it comes to representation, as it happened before, it is important to select the appropiate dimension values to show the results.

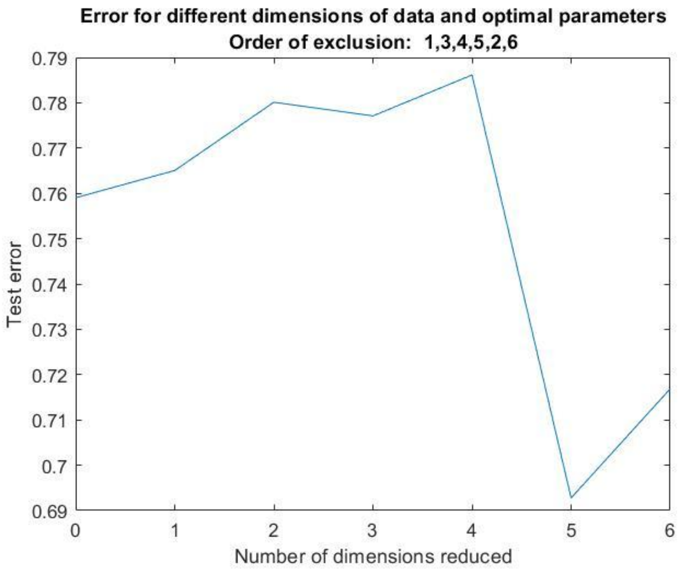




In the first plot we can see the results of the classification for the dimensions 2 and 7. On the second graph we compare the results of the antn value for the different test data points and the ground truth. This can be interpretated as how confident the model is with respect its prediction for that test point.

In this case we have obtained a total number of 94 SV, way bigger than in the artificial data. This is understandable as the data is more difficult to separate and the number of dimensions is bigger.

The results in this case, if we compare it with other models previously studied is quite poor, but we can try to improve it by reducing the number of dimensions that are part of the classification. Thus, we have reduced the number of dimensions following the fashion derived from previous exercises i.e. = [1,3,4,5,2,6]



As we can see, we were able to obtain a better test accuracy by reducing the number of dimensions. The best results were achieved when we only used dimensions [2,6,7], as it was expected based on previous experiments with the dataset.