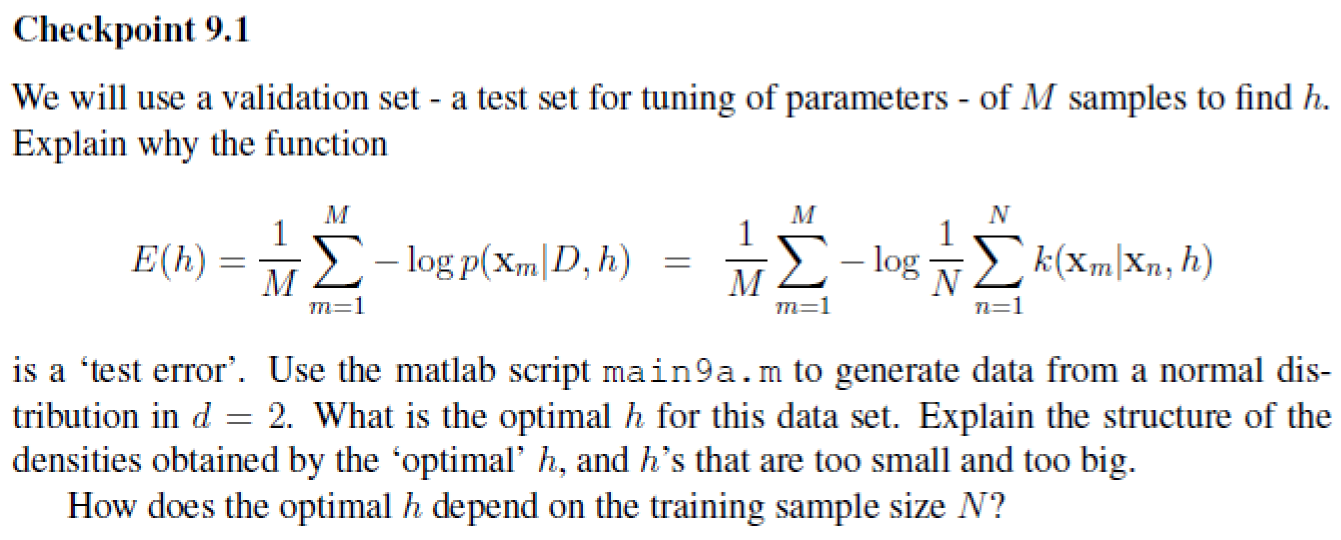
This is a parametric method as it learns from data and is not estimating any parameters. The only parameters h is not learnt but set prior to execution.



Checkpoint 9.1

We will use a validation set - a test set for tuning of parameters - of M samples to find h. Explain why the function

$$E(h) = \frac{1}{M}\sum_{m=1}^{M} -\log p(\mathbf{x}_m|D,h) \quad = \quad \frac{1}{M}\sum_{m=1}^{M} -\log \frac{1}{N}\sum_{n=1}^{N} k(\mathbf{x}_m|\mathbf{x}_n,h)$$

is a 'test error'. Use the matlab script `main9a.m` to generate data from a normal distribution in $d = 2$. What is the optimal h for this data set. Explain the structure of the densities obtained by the 'optimal' h, and h's that are too small and too big.

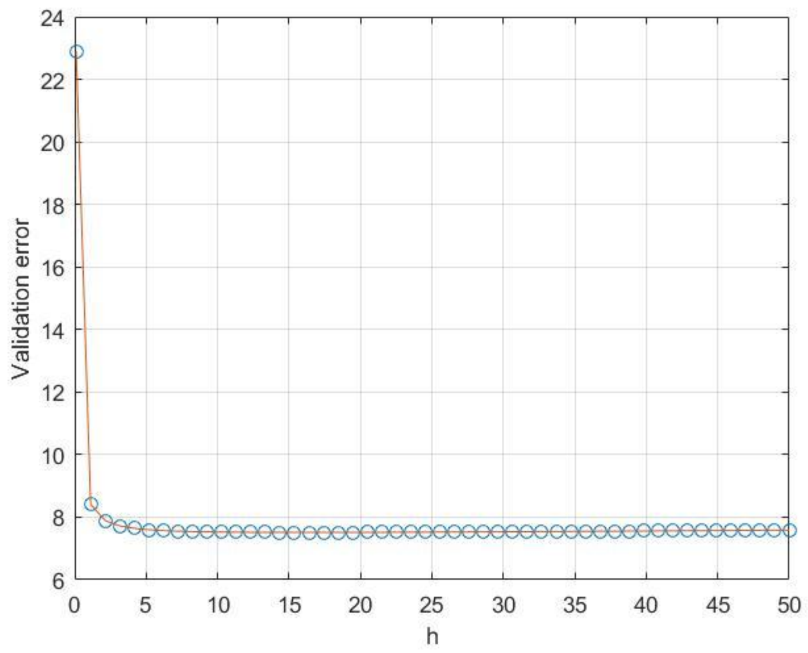
How does the optimal h depend on the training sample size N?

h=variance in this case

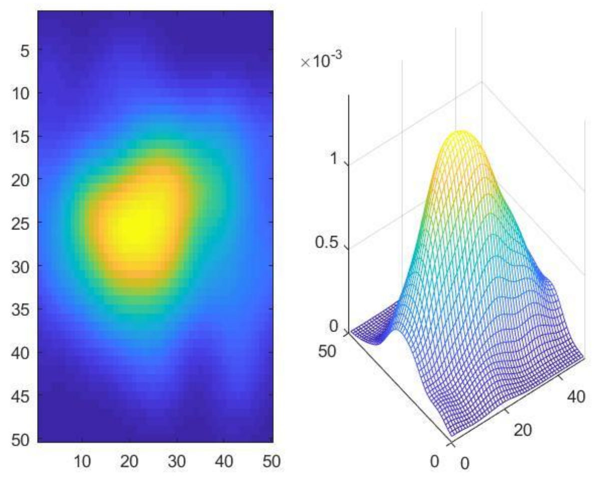
Let's start by taking a look at the test error. This test error maximizes the probability of all the points in the validation set given a set of data points D and a certain bandwith h. If we want to express this error as a function of the kernel we are maximazing the probability of each validation point to be part of gaussians with mean as the different values of x_n and diagonal covariance matrix with the bandwith values as variances.

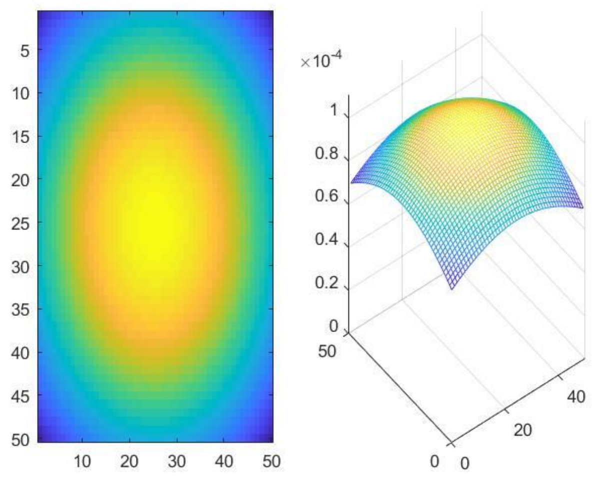
The loglikelihood is used to maximize the probability of getting validation set from sampling the estimated density function (using the model). So first we write the loglikelihood equation and then replace the probability density function with our model (Parzen Window Density Estimation). This way we will get the loglikelihood of validation set given our model. We want to formulate the problem as minimization problem so we formulate the error as negative loglikelihood and then by minimizing the error we will be maximizing the likelihood.

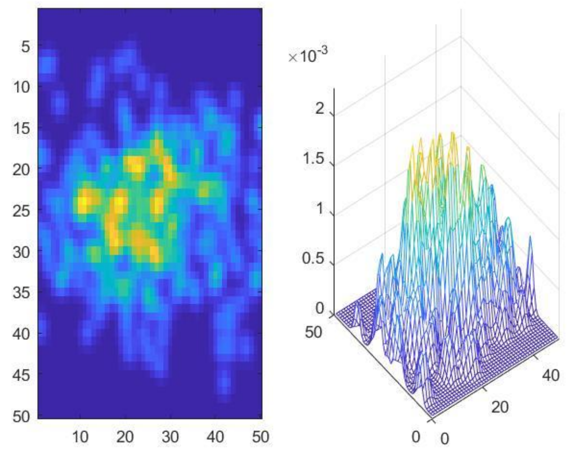
Regarding the optimal h, this can be calculated as the h that gives you the lowest test error. For this current dataset and N=300 we can see that the average value is around 17, but it varies quite a lot depending on the randomness introduced by the sampled data.



Then the plots of estimated probability distributions are drawn for: optimal h, 0.1 * optimal h, 100 * optimal h. This allows us to see the structure of obtained densities.

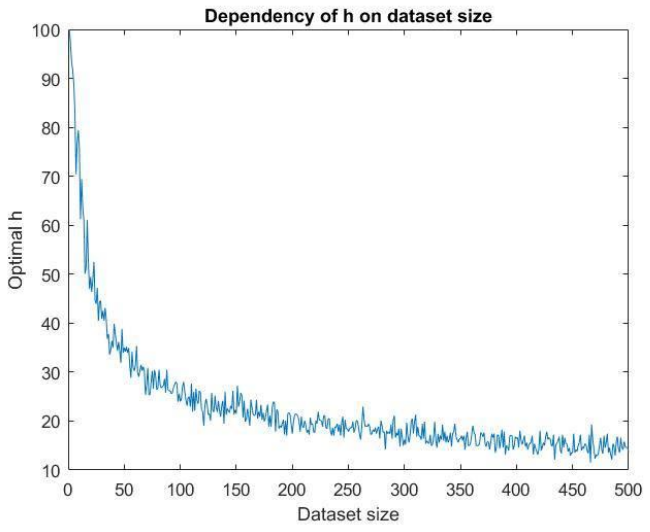


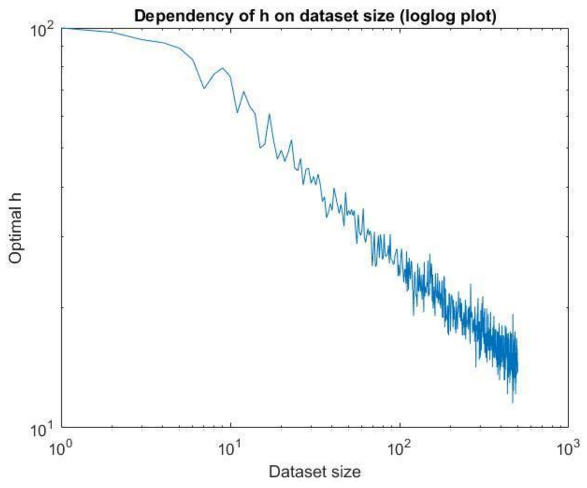




Now, and stilll in the case when N=300, different bandwith values were tested. The first one is the optimal value of h, the one that minimizes the test error. As we can see plot is pretty 'smooth', even though you can see the influence from the different random clusters. In the second case the optimal h was multiplied by 10, which resulted in a very uniform and spread distribution. However, the error in this case in higher, as it is too spread, and the probabilities of the validation points are smaller. Finally, in the last case we divided the optimal h by 10, which causes overfitting to the datapoints. This causes a large number of clearly identifiable gaussian distributions that does not represent the validation accurately.

In order to asses the dependency of N on the optimal h we present the following plot, where different N were tested for h values in between 0.1 and 100



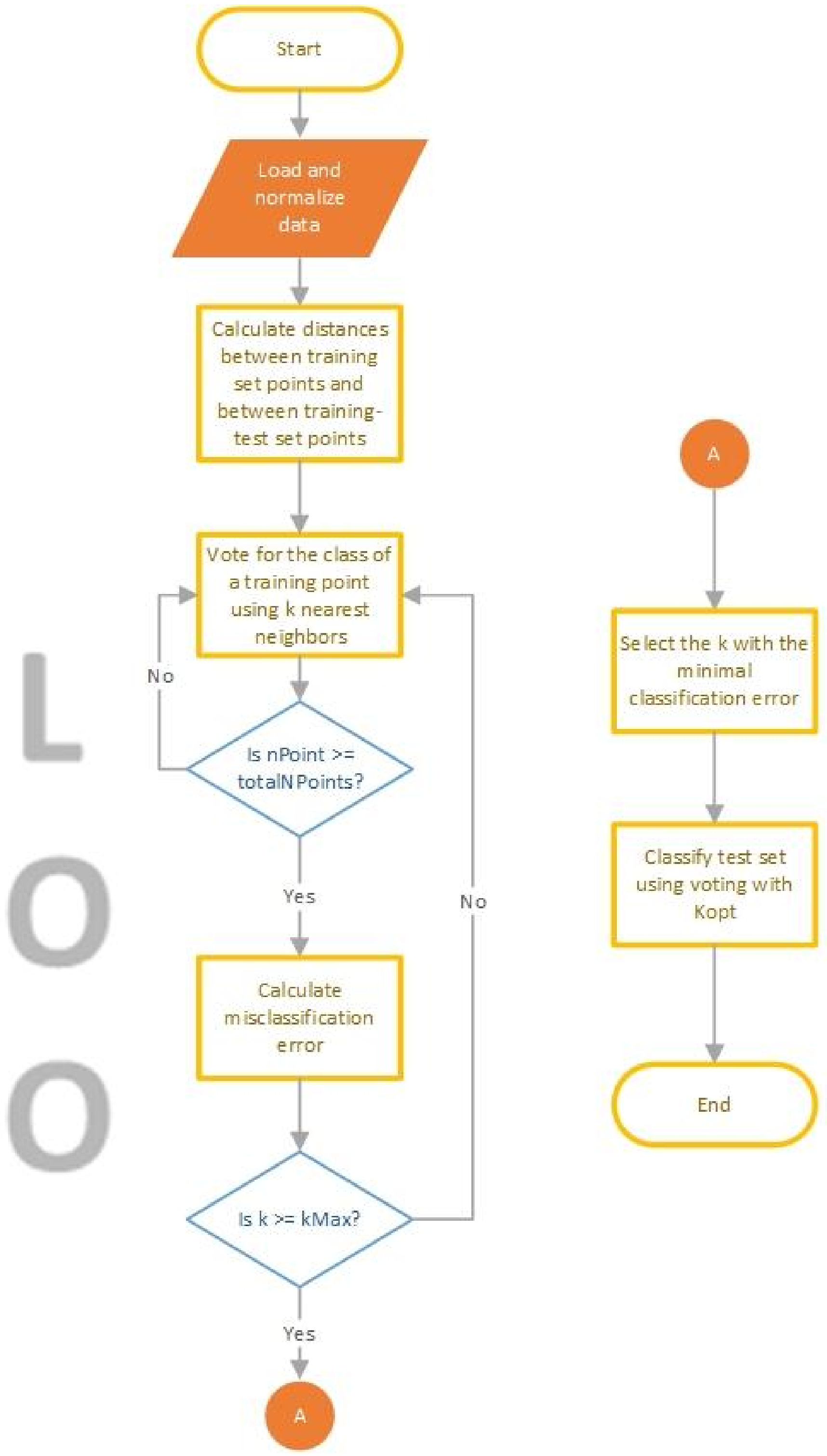


As we can see, for low values of N the optimal value of h is large, as even with a few datapoints you will have a quite spread gaussian distribution. As the number of datapoints increases the optimal h reduces a following an inverse polynomial function. In order to probe this we can take a look at the same data over a loglog plot. There we can see that the relation in almost linear, with a slope of around $\frac{1}{4}$ which means that the error decreases following the rate of $\frac{1}{\sqrt[4]{N}}$. So in order to calculate the optimal h for a given N sized dataset we will just have to calculate $100 \cdot \frac{1}{\sqrt[4]{N}}$

Checkpoint 9.2

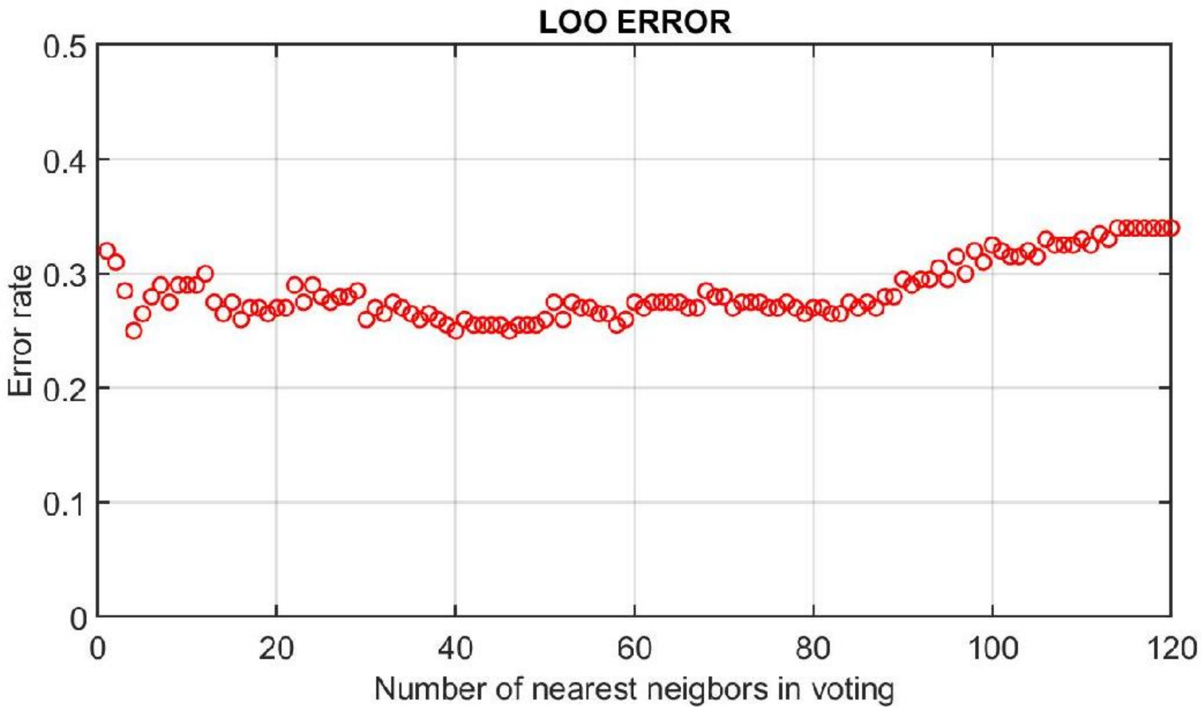
Explain how the 'leave one out' error can be used for identifying the optimal number of neighbors for voting. Use the matlab script `main9b.m` to classify the diabetes diagnosis data set. What is the optimal K? How well is the KNN performance compared to neural networks and other methods considered earlier in the course.

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 feature set?


Start
Load and normalize data
Calculate distances between training set points and between training-test set points
Vote for the class of a training point using k nearest neighbors
No
Is nPoint >= totalNPoints?
Yes
Calculate misclassification error
Is k >= kMax?
No
Yes
A
A
Select the k with the minimal classification error
Classify test set using voting with Kopt
End
L
O
O

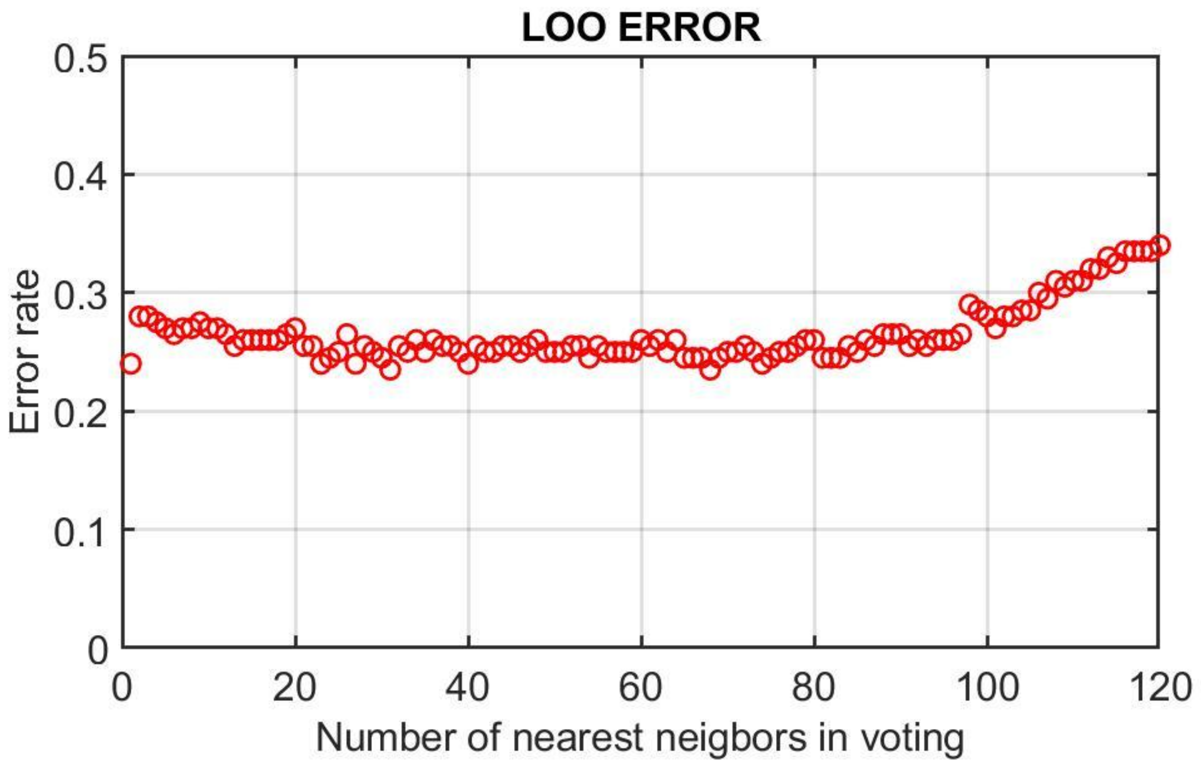
The leave out error helps us identify the optimal K by evaluating our training set on one point of it, i.e. by treating it as a test point. This allow us to calculate the missclassification error for all the points and then repeat with different K. As afterwards we will evaluate the test data on the same training set it is safe to assume that the optimal K will be similar for both cases.

The optimal K obtained following this method was K=4, as we can also note from the following plot:

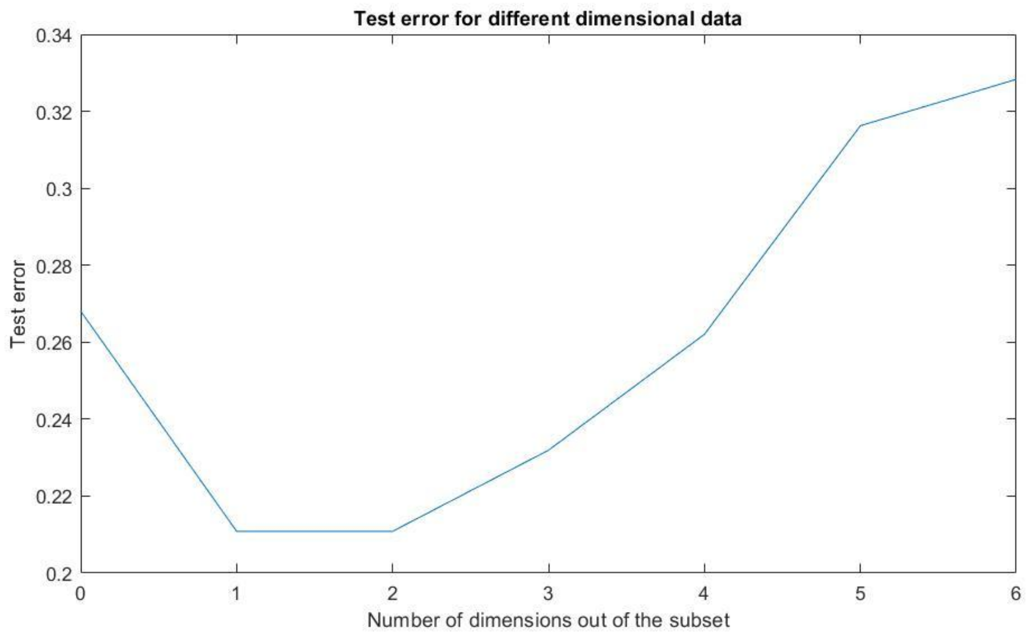


As we can note from the graph, after K=4 the error slightly increases, amking this point the minimum of our function and the optimal K.

From previous exercises we saw that some of the dimensions play a greater role than others when classifiying this dataset. Thus, it makes sense to think that by deleting some of the dimensions we can achieve similar performance. Based on the results from the optimal brain damage experiment out of the NN exercise we decide to delete completely dimensions 1 and 3, which gave us the following results:



We can see that the LOO error is really similar. However, in order to asses the performance of this subset we should take a look at the test error. In order to do so we are going to select different subsets and evaluate the error for them, reducing one dimension in each subset based on the results of the prunning of the NN (i.e. 1,3,4,5,2,6)

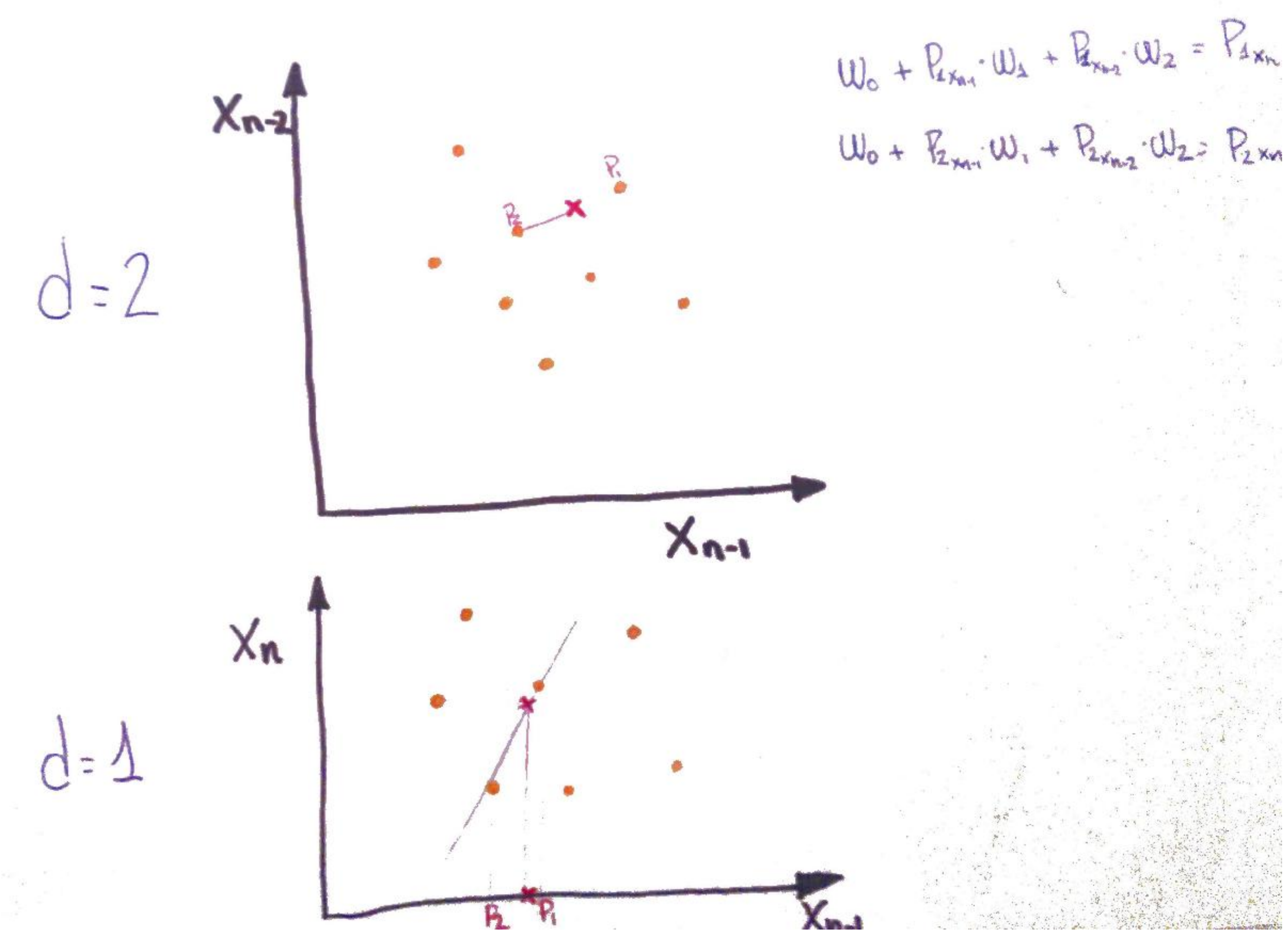


The results show that when we take out the 1st and 3rd dimension out of our dataset the results are better than with them. Furthermore, when our subset is composed only by dimensions 2,6 and 7 our model performs on par with the full dimensional dataset. This results are remarkable and they show how adding data that does not correlate with our classes can cause a drop in performance of the model.

Checkpoint 9.3

Inspect the matlab script `main9c.m`. Explain the role of the parameter 'alpha'. Why is it necessary to regularize the linear model? What is the meaning of the parameter d and what is optimal value of d. Make a drawing that explains the algorithm conceptually, e.g., in a case with two-dimensional input.

Compare the quality of the algorithm's predictions with the neural network based predictions we found in exercise 5. What would happen if we used $K = N_{train}$?.

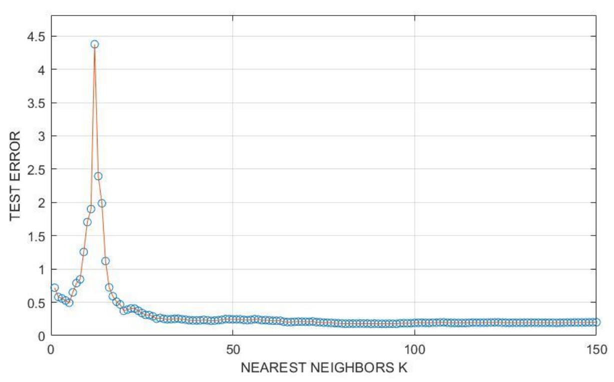

$W_0 + P_{1x_{n-1}} \cdot W_1 + P_{1x_{n-2}} \cdot W_2 = P_{1x_n}$
$W_0 + P_{2x_{n-1}} \cdot W_1 + P_{2x_{n-2}} \cdot W_2 = P_{2x_n}$
d=2
X_{n-2}
X_{n-1}
P_1
P_2
d=1
X_n
X_{n-1}
P_2
P_1

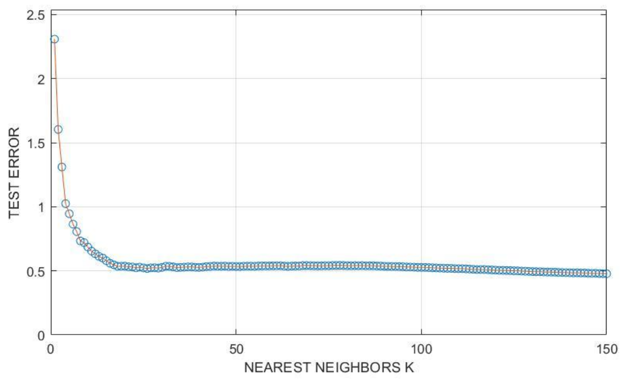
When we are calculating the weights that solve our linear system we are using the following equation (exercise 3):

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t} \;\equiv\; \mathbf{X}^\dagger \mathbf{t},$$

In order to be able to calculate this, the matrix $X^T X$ needs to be invertible (i.e. non-singular). However, this is not the case if the number of K is less than the number of dimensions+1 or if there is any collinearity between predictor vectors. Thus, in this cases, the rank of the matrix is actually 1, and the determinant 0. In order to avoid this problem and solve our differential equation we have different options. One is, as we did in exercise 3, is to compute the Moore-Penroose pseudoinverse, which will give us the the vector that solves the equation with the smaller norm, which makes our model more resistant to noise. However, this does not solve the problem of Another option, and the one choosen in this exercise is to introduce a small perturbation that will be added to the matrix as $X^T X + \alpha I$ which will make the matrix non-singular. This parameters sets the priors of our weights to a gaussian distribution, by penalizing large weights. This makes the estimator biased by that alpha factor.

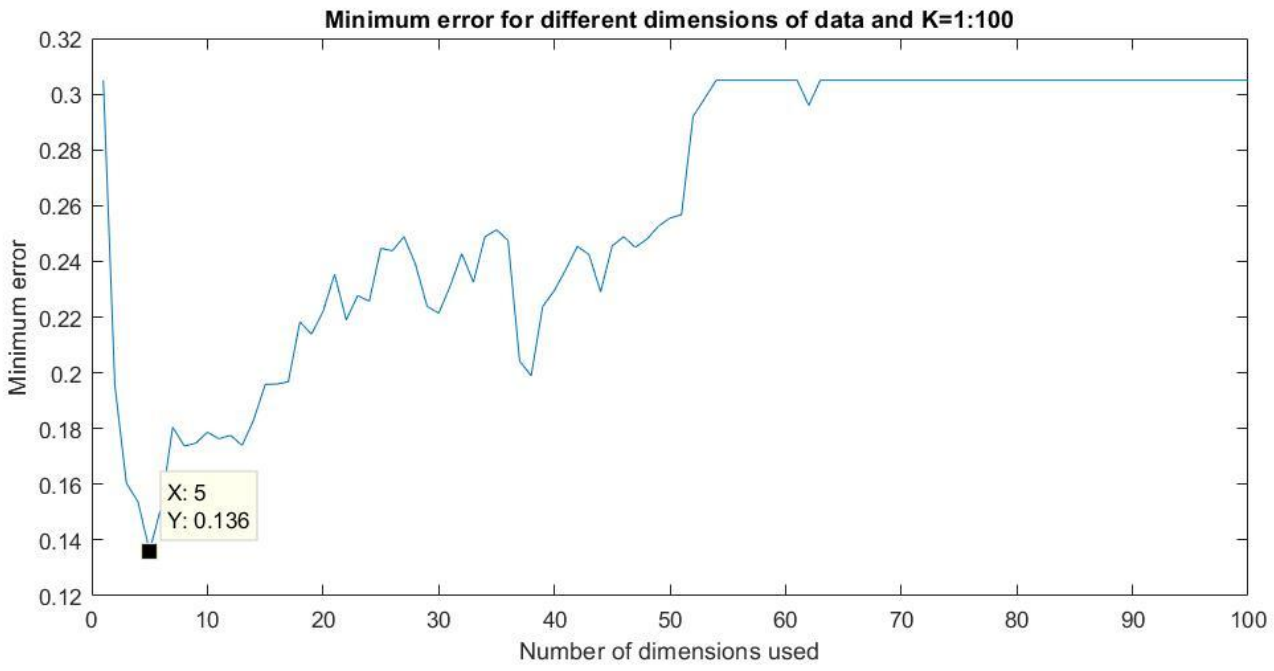
Moreover, the selection of this parameter can have a big influence on the performance of the model, as we see below, for the case when alpha is really small and really big





As we can see the test error pickes with small alpha when K=d+1. This is the same issue that we experience when we use the pseudo-inverse approach,and it can be explained because when the number of weights is the same as the number of points to fit it is possible to overfit the model on those points, including the noise. However, this is not possible if we include the regularization term, it becomes more difficult to do so as the weights are smaller. On the other hand, with big alphas the error is generally bigger as it is difficult to fit a model with that big perturbation.

The parameter d is the number of dimensions of our data. In other terms, it is the number of years that we are going to look back in time to predict the value of y. It will also define the space where we will find the nearest neighbours of the point to evaluate. In order to select the optimal d we have plotted the minimum error achieved with different number of dimensions d with K between 1 and 100.



Based on this results we see that the optimal number of d is 5, as it is the dimension where the error is minimum.