Université de Genève

Data Mining

TP 2: Knn

Author: Joao Filipe Costa da Quinta

E-mail: Joao.Costa@etu.unige.ch

April 19, 2021

UNIVERSITÉ DE GENÈVE

FACULTÉ DES SCIENCES

Département d'informatique

Introduction

In this TP I am going to try to implement k-nn classifier, and use it to classify a set of images, because the number of data is rather large, I will develop the same algorithm in 3 different ways to try and increase the performance. Then I will revisit the Iris problem that was in the last TP, and apply K-nn implementation to try and get better results.
By applying the same solution to two different problems, I will be able to notice some pros and cons of this method.
Finally I will apply transformations to a Distance matrix.

K-nn - Images Classification

With K=1, I got a 27% prediction rate, which was expected, Knn isn't considered to be a good algorithm when the number of dimensions of the variables are too high, which is the case in image classification. When I tried k=5 I did get a slight improvement over k=1, which was 29%.

The difference in computation time between the the 3 versions of the euclidean distance was amazing to see, I didn't expect such an improvement without loops over the code with loops.

K-nn - Iris Classification

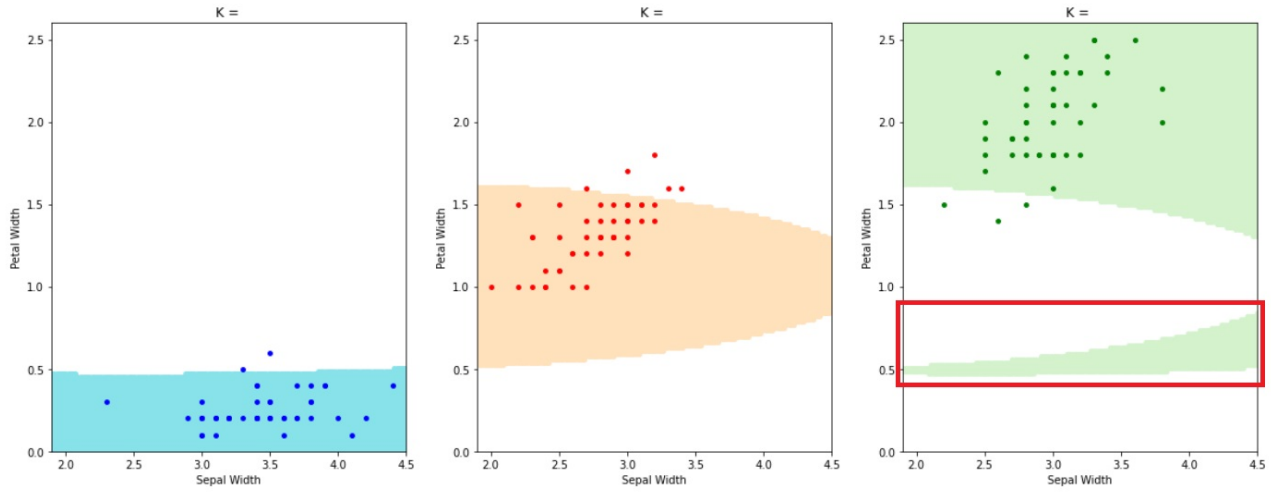
As the dimension of the variable is much lower in the Iris classification, I get much better results, depending on the way I compute the distance the accuracy varies, the best accuracy was 98% with k=10, distance = mahalanobis and cov = diag_cov or cov = full_cov.

While doing this classification, we see some other problems of knn that we didn't see during image classification, we now see that there is a k value that is optimal, here I always had worse results for k=50 than k=25, this is probably because there aren't 50 train points for each class, which means, that for every test point, the 50 neighbors will be too many.

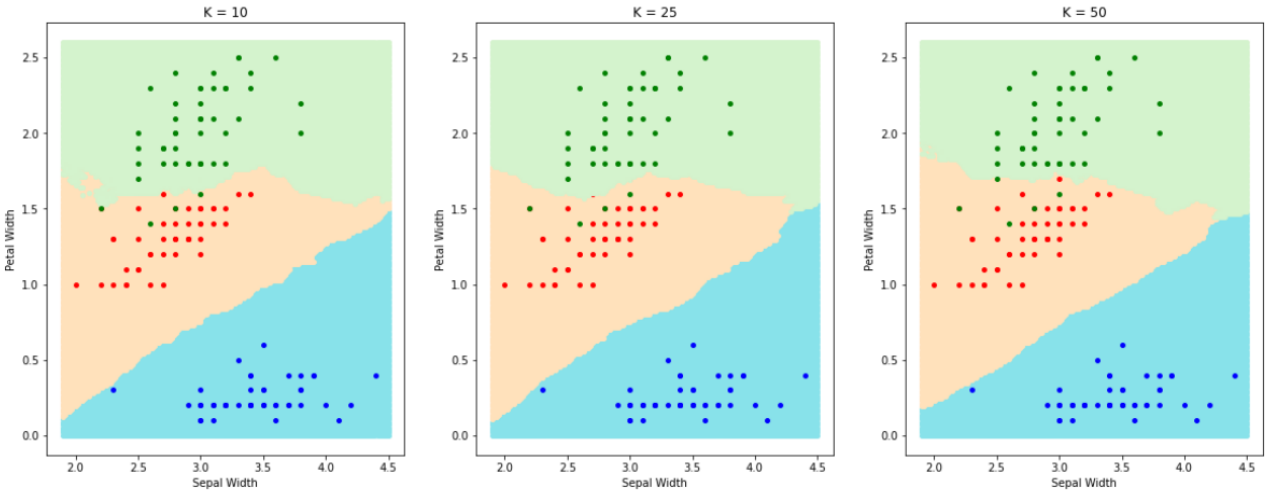
Decision Surface

Since we are doing the same thing as in last TP, we can compare the results we got now, to the ones we got last time, we do have a slightly higher accuracy, which is nice, but we also have much more natural decision surfaces, as we can see in the image below, the area in the red box was completely wrong, and has now been fixed, what is amazing is that it was fixed for every k value we tested.

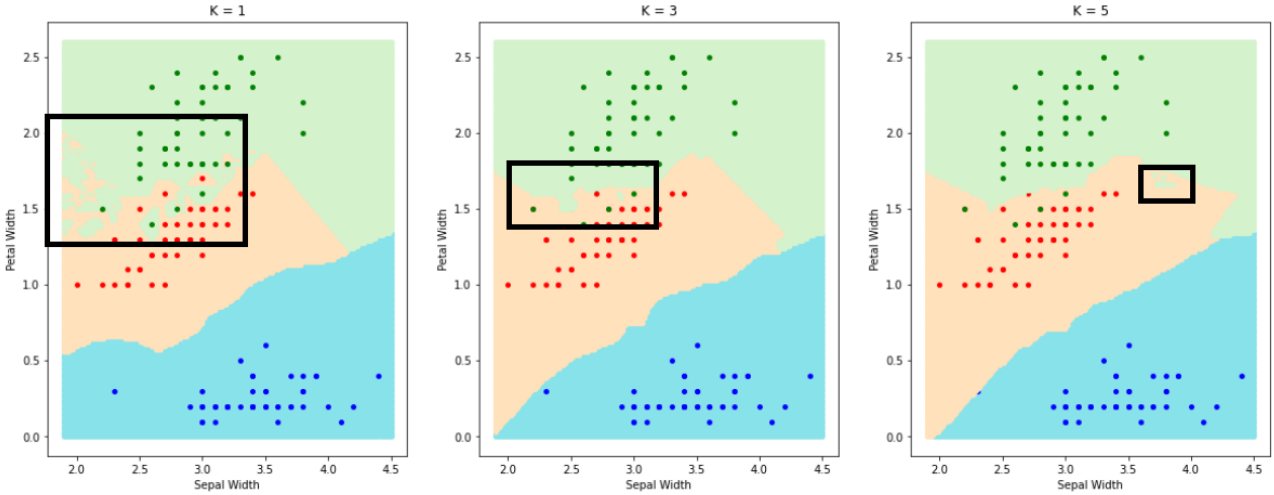
LAST TP:



CURRENT TP:



When k <= 10, we can also witness an over-fitting, with all the holes in each decision surface, as highlighted in the black areas:



Linear transformations

In the green circle all points are at distance = 1, and as we apply the first transformation, the blue ellipse shows that there are points that are now closer than others. Since in K-nn method we pick a class depending on the k closest points, depending on if we take the closest in the green circle or the blue ellipse the results will obviously vary. This transformation might be interesting if we know which transformation A to apply to each case, As we will be able to artificially force some points to be closer, and maybe improve our accuracy.
In the Red ellipse we have the inverse of A, we can see that the differences between red-green and blue-green are much larger.

Conclusion

It was very nice to improve the 94% accuracy from last TP, up to 98%, sadly the results weren't as encouraging for the image classification as 29% is very low. The decision surfaces were also much smoother this time, especially with k=10 or k=25.
This TP perfectly showed the use case for K-nn prediction as it didn't do well when the variables had too large dimensions, however with lower dimensions it is where it shines.