Homework 3: K-NN
class in "Machine Learning", Fall 2016/17

Marco Favorito
Master of Science in Engineering in Computer Science
Department of Computer, Control, and Management Engineering
University of Rome "La Sapienza'
favorito.1609890@studenti.uniroma1.it

07 November 2016

1 Part 1

1.1 Set up dataset

Now I show the plot of iris dataset when it is standardized, when PCA it is applied, and after the split train/test:

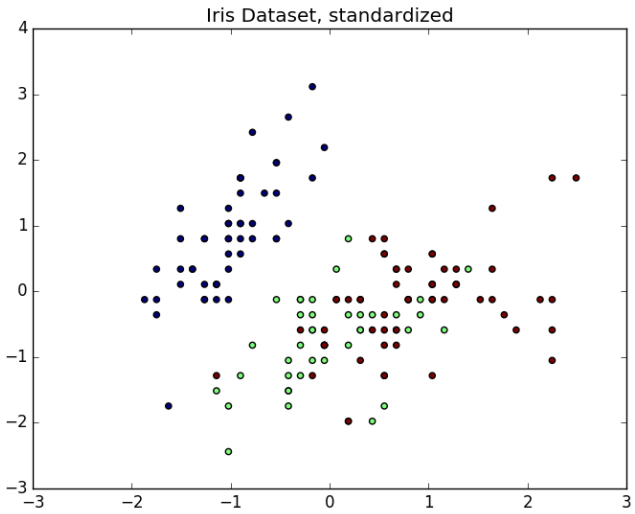

Figure 1: Iris Dataset Standardized

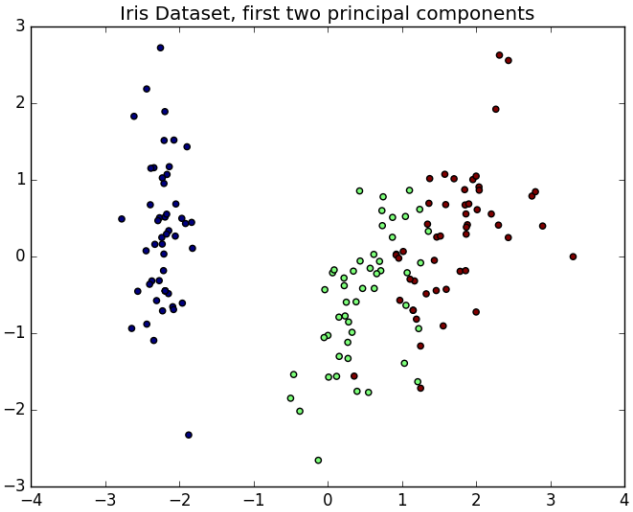

Figure 2: Iris Dataset Principa Components

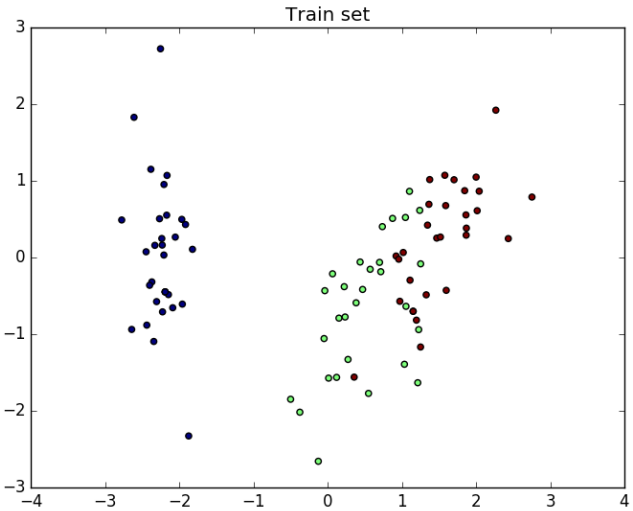

Figure 3: Iris Dataset Train set

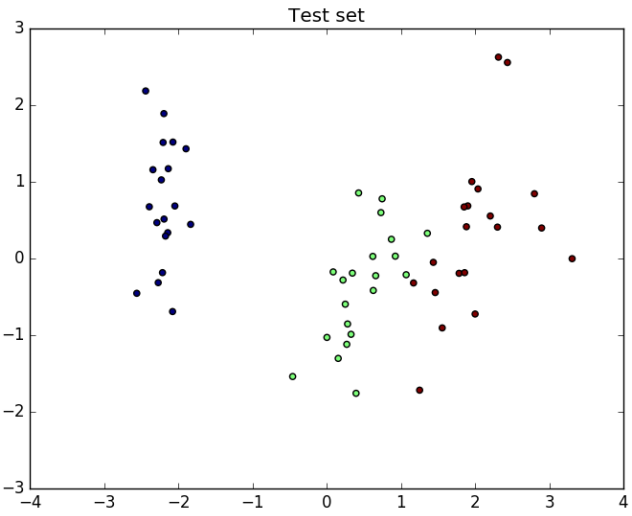

Figure 4: Iris Dataset Test Set

1.2 Perform K-NN

I executed K-NN with k from 1 to 10, with uniform weights and euclidean metric. Now I show you plots and accuracies. Notice the legend: in each plot, light circles represent train set data, dark circles represent test set data and marker "x" represent wrong prediction.

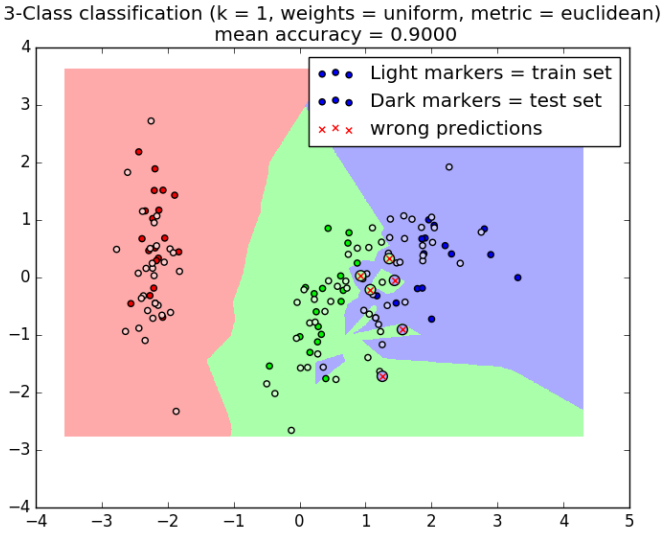

Figure 5: K-NN with k=1, weights=uniform, metric=euclidean

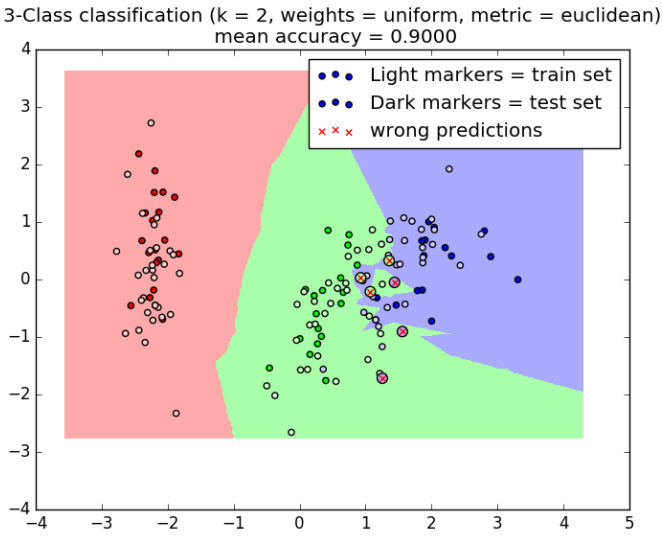

Figure 6: K-NN with k=2, weights=uniform, metric=euclidean

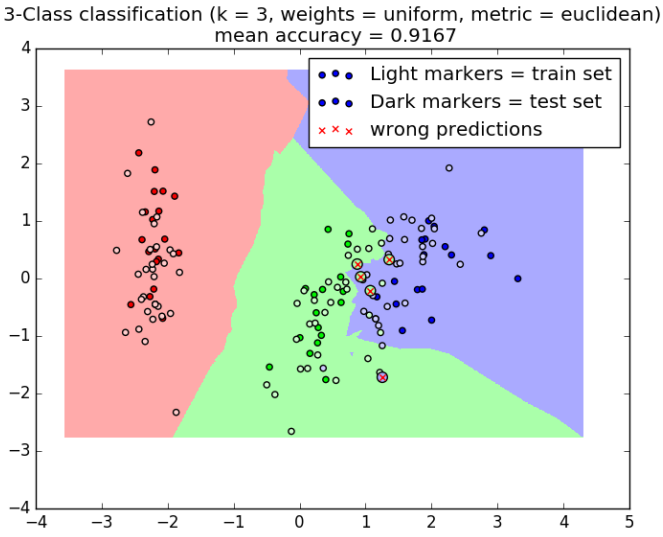

Figure 7: K-NN with k=3, weights=uniform, metric=euclidean

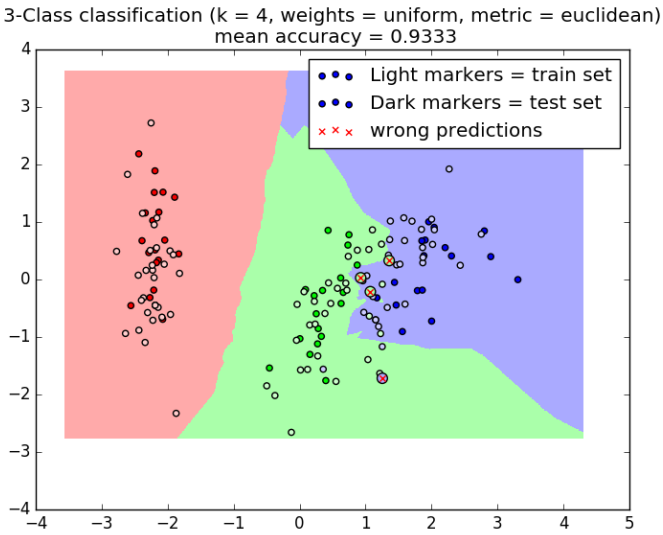

Figure 8: K-NN with k=4, weights=uniform, metric=euclidean

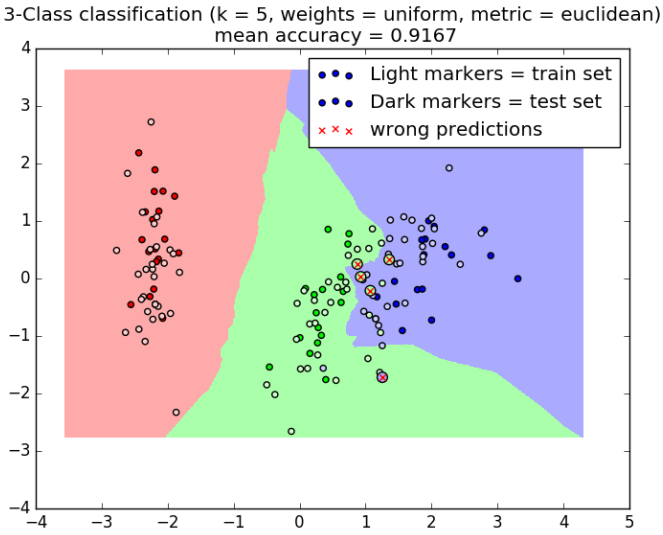

Figure 9: K-NN with k=5, weights=uniform, metric=euclidean

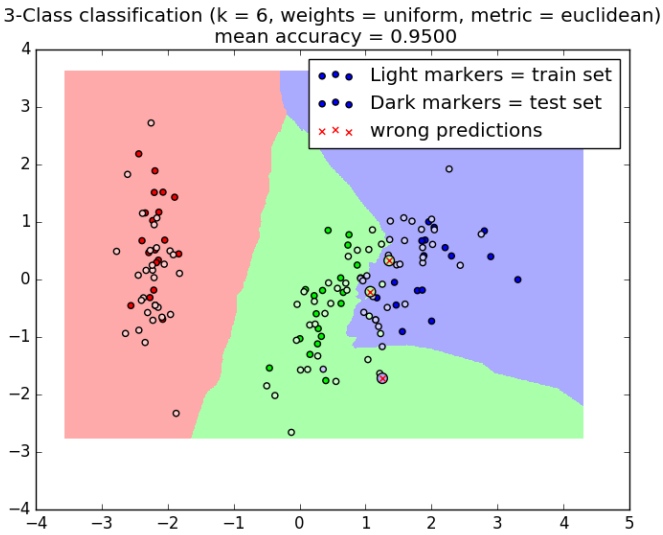

Figure 10: K-NN with k=6, weights=uniform, metric=euclidean

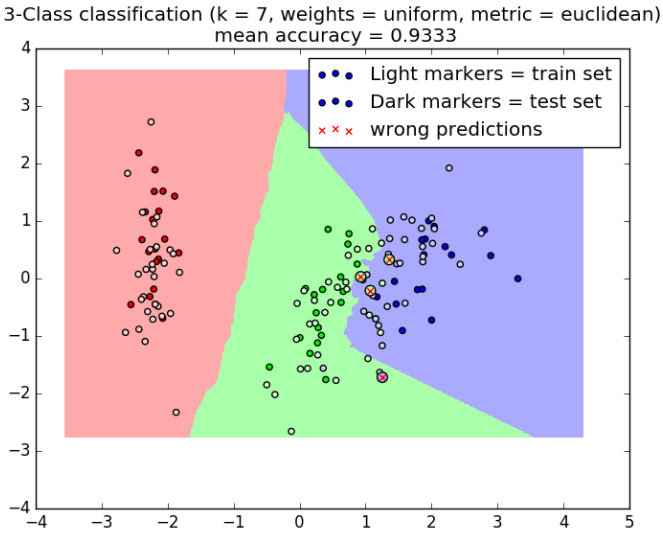

Figure 11: K-NN with k=7, weights=uniform, metric=euclidean

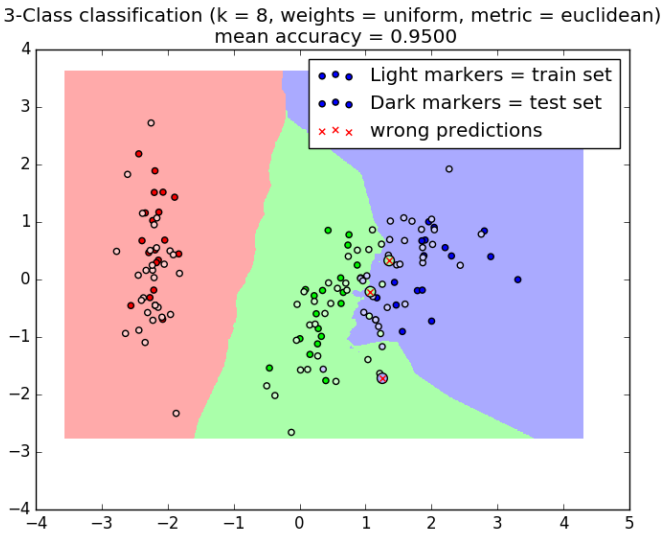

Figure 12: K-NN with k=8, weights=uniform, metric=euclidean

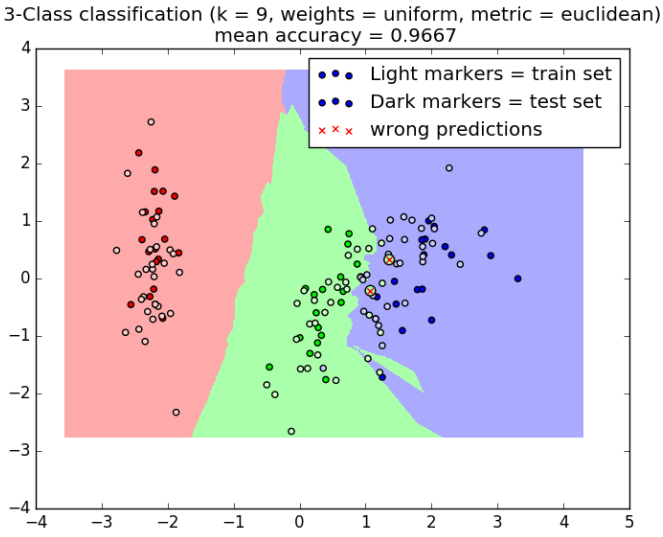

Figure 13: K-NN with k=9, weights=uniform, metric=euclidean

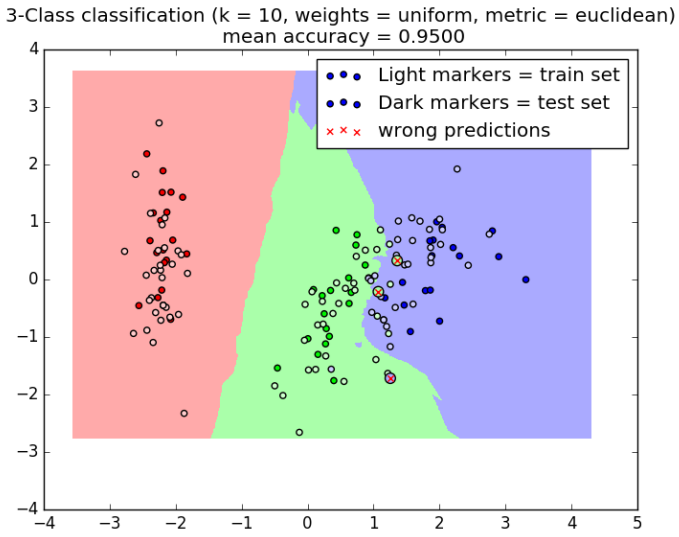

Figure 14: K-NN with k=10, weights=uniform, metric=euclidean

In the following table I show accuracies wrt k:

Table 1: k - accuracy table

k	accuracy
1	0.9
2	0.9
3	0.916666666667
4	0.933333333333
5	0.916666666667
6	0.95
7	0.933333333333
8	0.95
9	0.966666666667
10	0.95

Boundaries change because change the criterion by which points in the sample space will be assigned to classes. Apparently, there is no regularities in the variation of decision boundaries. Neveretheless, in the first three plots, it is easy to notice that some isolated regions of decision gradually disappear as k increases. Indeed, with higher value of k the algorithm tends to ignore some "special cases" or, in other words, noisy data, because we consider a higher number of neighbours taken into account for classification. On the other hand, too high value of k make the algorithm to be inclined to include a considerable number of data in a wrong region; in general it is not wrong, as in our case, where we can see how performances, in terms of acuracy, grow. But we can observe that, in the nearest of green-blue boundaries (the most critical), some groups in the train dataset, labeled with blue color, have been classified in the green-labeled region, and viceversa. Probably, with more test data, computed accuracies may be not so relevant.

1.3 Using different weights functions: uniform and distance

Now I show the plot with $k = 3$ and using different weights function (same metric):

We can observe that in the case of the distance weights, some isolated points creates a decision region around them. Indeed, in the case where the algorithm, for some samples, cannot decide how classify them only considering their neighbours, it will use the weight function. In the case of distance, all the points in the space that are close to them will be classified to the same class.

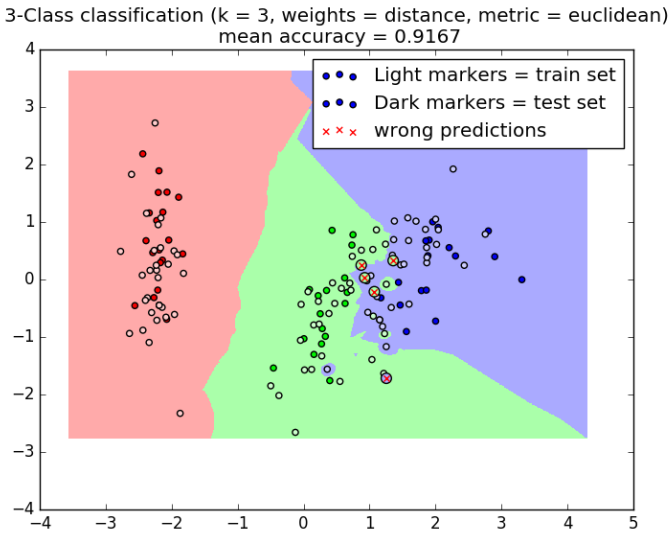

Figure 15: K-NN with k=3, weights=distance, metric=euclidean

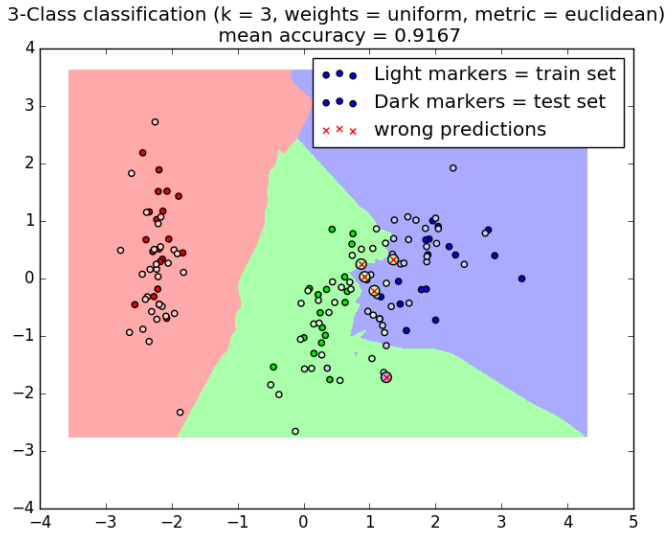

Figure 16: K-NN with k=3, weights=uniform, metric=euclidean

2 Part 2

2.1 Plot decision boundaries with our Gaussian function

In the following there will be plotted decision boundaries varying α and with $k = 3$.

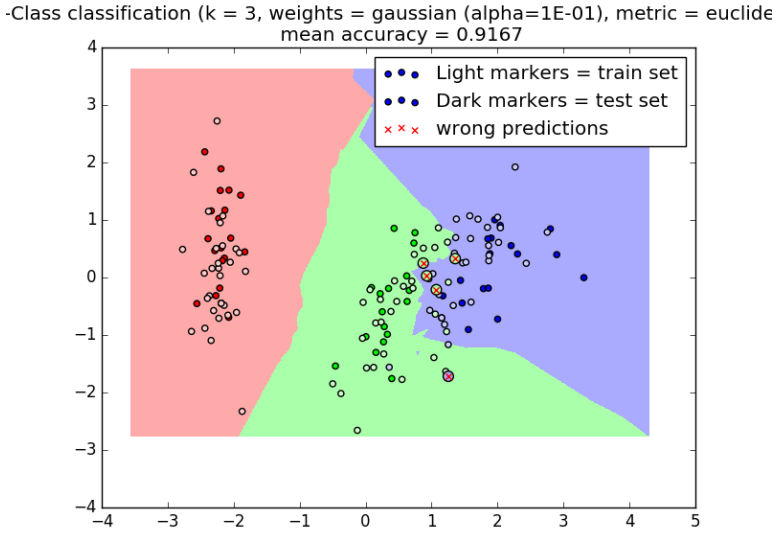

Figure 17: K-NN with k=3, weights=my_gaussian, metric=euclidean, $\alpha = 0.1$

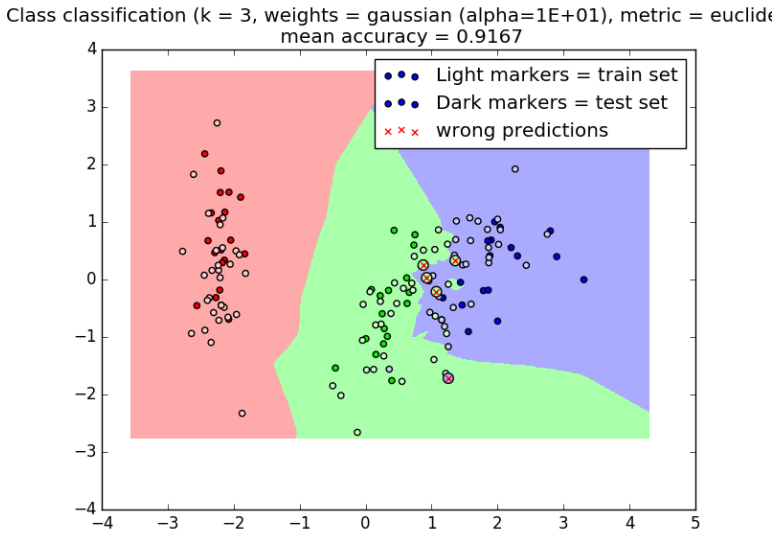

Figure 18: K-NN with k=3, weights=my_gaussian, metric=euclidean, $\alpha = 10$

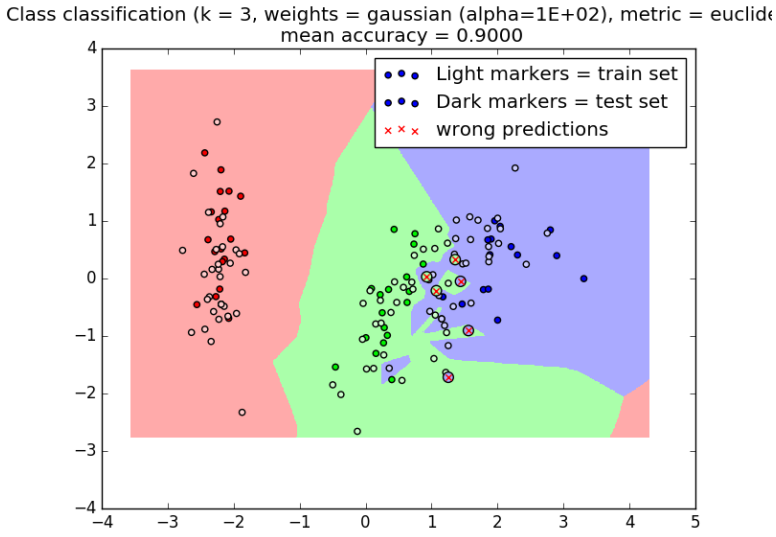

Figure 19: K-NN with k=3, weights=my_gaussian, metric=euclidean, $\alpha = 100$

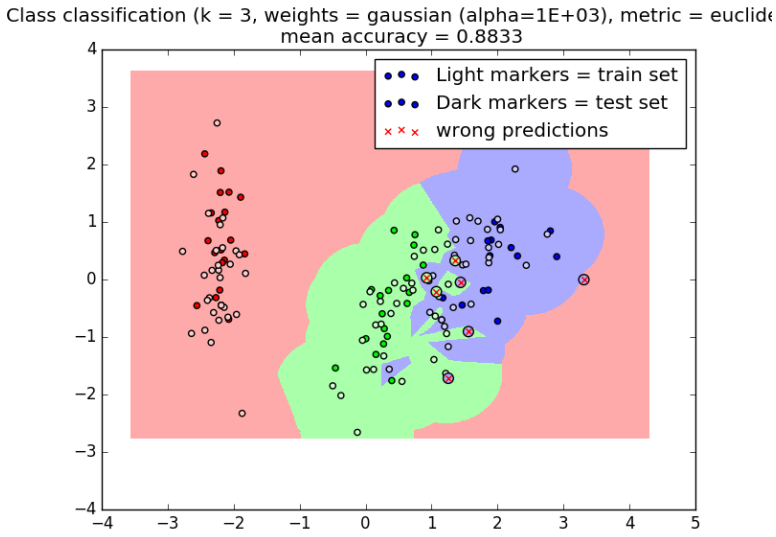

Figure 20: K-NN with k=3, weights=my_gaussian, metric=euclidean, $\alpha = 1000$

Notice that for low value of α, the plot is very similar to the plot with uniform-weights function. As α increases, tends to be similar to distance weights plots, and when $\alpha = 1000$ we see a lot of differences.
We can observe, first of all, that red decision region appears also in the right part of the picture: this is due to our weight function, that classify space points in that area as belonging to the red class; indeed, for our function, these points are "nearer" to red points (i.e. has a lower distance) instead of green or blue ones. This inversion is due to the form of our weight function, that is a decreasing function on d.
Moreover, it is worth to notice what is happened on green-blue class region boundaries: some irregularities have appeared, where some isolated points are rewarded by our decision function and are classified correctly.

2.2 Grid search

I performed an execution search on the following domains:

- list of weight functions:
    1. uniform
    2. distance
    3. gaussian(α=0.1)
    4. gaussian'(α=10
    5. gaussian(α=100)
    6. gaussian(α=1000)
- list of k values: $1, 2, \ldots, 10$

Here there are the computed accuracies:

Table 2: My caption

k						
weight function	uniform	distance	$\alpha = 0.1$	$\alpha = 10$	$\alpha = 100$	$\alpha = 1000$
01	0.900	0.900	0.900	0.900	0.900	0.883
02	0.900	0.900	0.900	0.900	0.900	0.883
03	0.917	0.917	0.917	0.917	0.900	0.883
04	0.933	0.933	0.933	0.933	0.900	0.883
05	0.917	0.917	0.917	0.917	0.900	0.883
06	0.950	0.917	0.917	0.917	0.900	0.883
07	0.933	0.933	0.933	0.933	0.900	0.883
08	0.950	0.933	0.933	0.933	0.900	0.883
09	0.967	0.933	0.967	0.933	0.900	0.883
10	0.950	0.933	0.950	0.933	0.900	0.883

The maximum of accuracy is in $k = 9$ and weight function = uniform. Here the plot of decision boundaries:

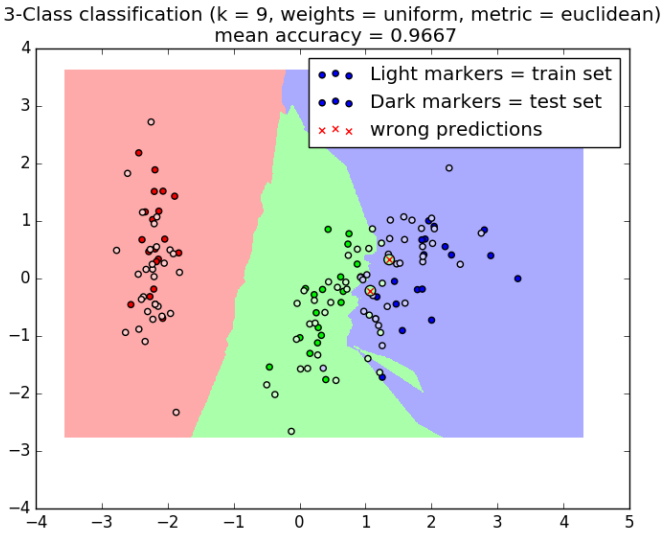

Figure 21: K-NN with k=9, weights=uniform, metric=euclidean

As we have already seen, in this case a high number of neighbors to be considered is the best choice, since the critical green-blue boundary is hardly separable in linear way: it is better to ignore some "black swan" and include it in not appropriate groups.