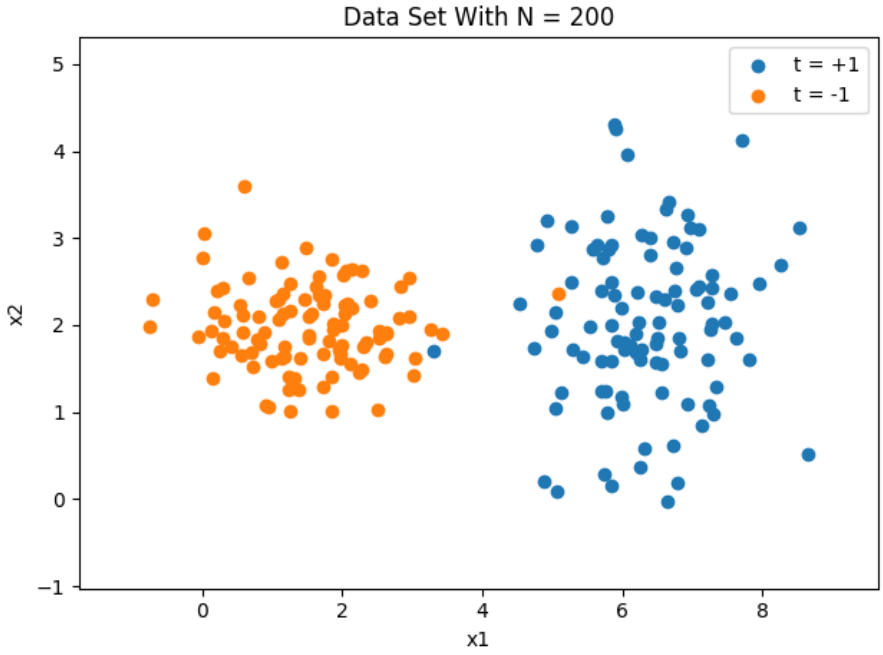
Results for Assignment 3 - Group 34

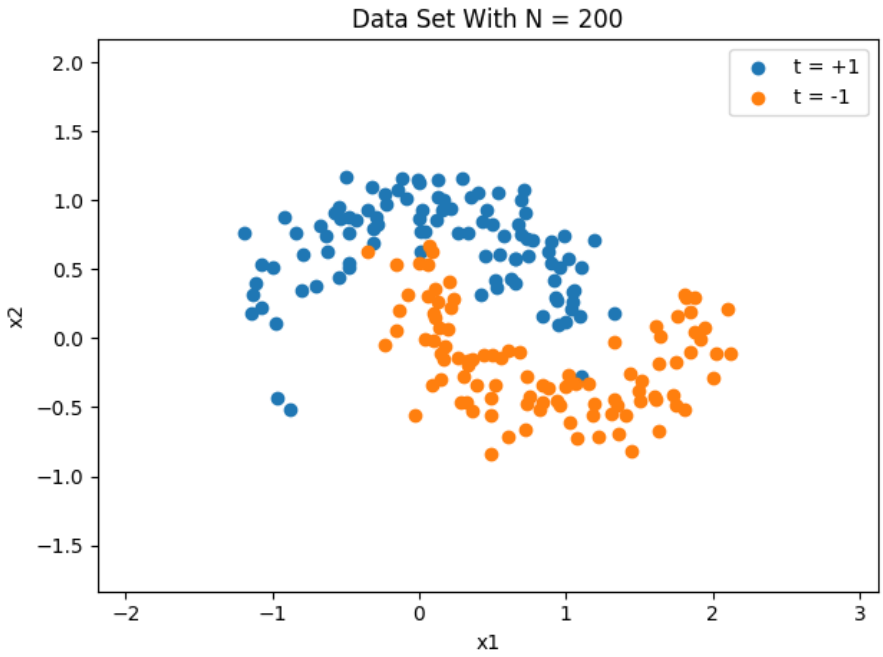
Markus Ruplitsch, Julia Tschuden

June 17, 2020

1 Task 1

We generated 100 samples from each class and plotted them on a 2d plane. The resulting plots can be seen below.



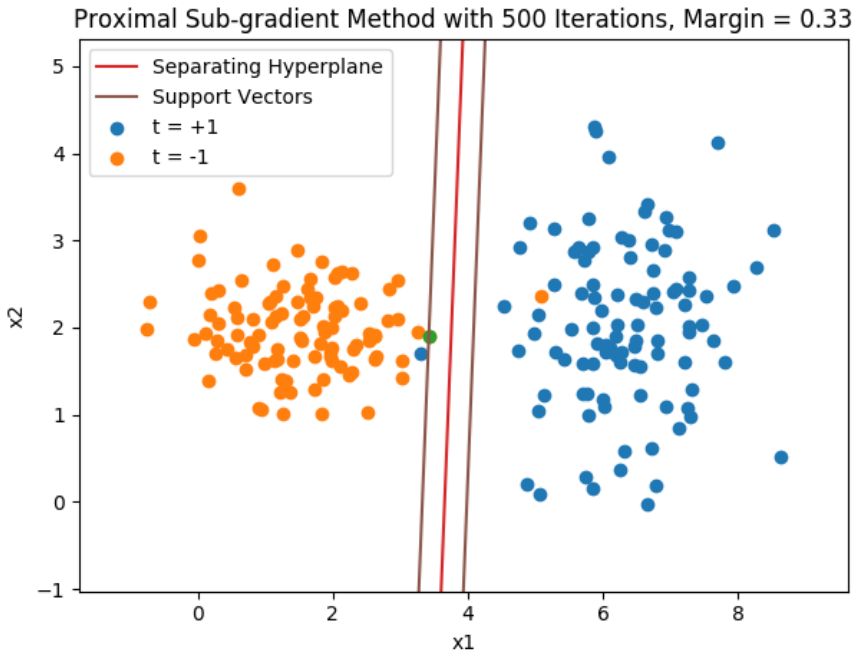


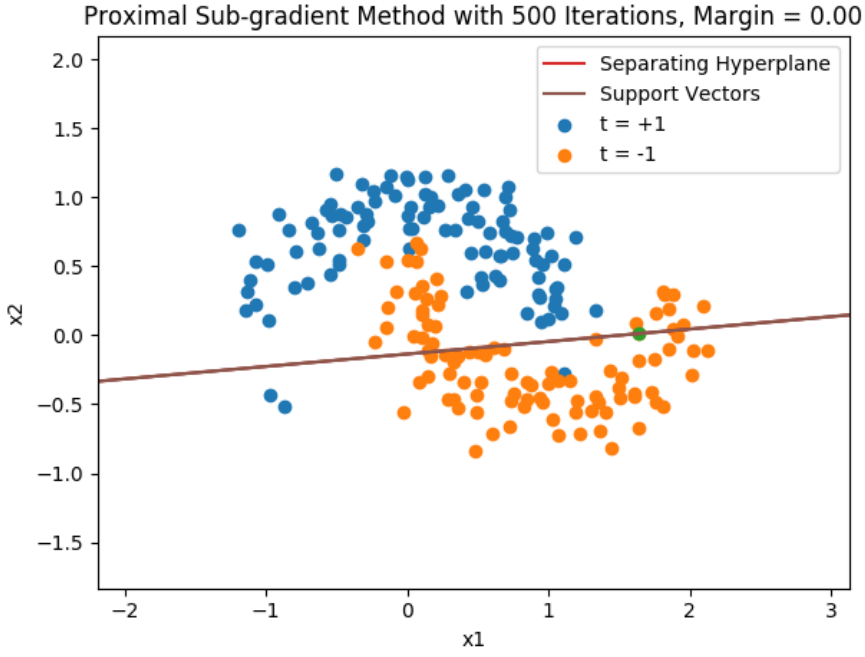
2 Task 2

2.1

The algorithm described in the paper was implemented under the name "proximalSubGradientMethod". We decided to use a fixed number of iterations as stopping criteria. We tested several different numbers of iterations and decided to use 500 iterations, as the resulting weights seemed to no longer change significantly after that.

As the algorithm provided did not calculate the decision boundary directly from support vectors, and it was not clearly stated how the support vectors and the corresponding margin should be calculated, we decided the best result would be to take the euclidean distance from the decision boundary to the nearest point as margin. This means that in most cases there will only be a single support vector. The results can be seen below.


Proximal Sub-gradient Method with 500 Iterations, Margin = 0.33
Separating Hyperplane
Support Vectors
t = +1
t = -1
x1
x2


Proximal Sub-gradient Method with 500 Iterations, Margin = 0.00
Separating Hyperplane
Support Vectors
t = +1
t = -1
x1
x2

2.2

The simple data set is, as stated in the assignment sheet, linearly separable which is why this method of classification achieves extremely good results. As long as there are no overlaps, we can easily achieve 100% accuracy and even if there are some, the accuracy is usually well above 95% (In the example provided, we achieved 99% accuracy).

Comparing this to the half moon dataset, we can see that such high accuracy is simply impossible with this algorithm, without making any modification. Since our algorithm only returns the best straight line to separate the data, we cannot hope for such good results (In the example provided, we achieved 81.5% accuracy).

3 Task 3

3.1

Before starting to analytically compute the dual problem we have to rewrite (4) from the assignment description, which is:.

$$t_n\langle\widetilde{\mathbf{w}}, \phi(x_n)\rangle \geq 1, \quad \forall n = 1, \ldots, N \tag{1}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product of $\widetilde{\mathbf{w}}$ $(= (b, \mathbf{w})^T)$ and $\phi(x_n)$. Rewritting this gives us:

$$t_n\widetilde{\mathbf{w}}^T\phi(x_n) \geq 1 \tag{2}$$

$$1 - t_n\widetilde{\mathbf{w}}^T\phi(x_n) \leq 0 \tag{3}$$

Using this we can construct the Lagrangian similarly as in the lecture notes, using (3) from the assignment description:

$$L(\widetilde{\mathbf{w}}) = \frac{1}{2}|\widetilde{\mathbf{w}}|_M^2 + \sum_{n=1}^{N} a_n \cdot (1 - t_n\widetilde{\mathbf{w}}^T\phi(x_n)) \tag{4}$$

$$L(\widetilde{\mathbf{w}}) = \frac{1}{2}|\widetilde{\mathbf{w}}|_M^2 - \sum_{n=1}^{N} a_n \cdot (t_n\widetilde{\mathbf{w}}^T\phi(x_n) - 1) \tag{5}$$

$$L(\widetilde{\mathbf{w}}) = \frac{1}{2}|\widetilde{\mathbf{w}}|_M^2 - \sum_{n=1}^{N} a_n t_n\widetilde{\mathbf{w}}^T\phi(x_n) + \sum_{n=1}^{N} a_n \tag{6}$$

We now have to minimize $L(\widetilde{\mathbf{w}})$ with respect to $\widetilde{\mathbf{w}}$, which contains both w and b, that are needed to compute the optimal hyperplane. This minimization is done by deriving $L(\widetilde{\mathbf{w}})$ and setting it to zero.

$$\nabla L(\widetilde{\mathbf{w}}) = \widetilde{\mathbf{w}}\mathbf{M} - \sum_{n=1}^{N} a_n t_n \phi(x_n) = 0 \tag{7}$$

$$\widetilde{\mathbf{w}}\mathbf{M} = \sum_{n=1}^{N} a_n t_n \phi(x_n) \tag{8}$$

$$\widetilde{\mathbf{w}} = \sum_{n=1}^{N} a_n t_n \phi(x_n) \cdot \mathbf{M}^{-1} \tag{9}$$

After defining $\widetilde{\mathbf{w}}$ we can know plug it back into (6).

$$L(\widetilde{\mathbf{w}}) = \frac{1}{2}\widetilde{\mathbf{w}}^T\mathbf{M}\widetilde{\mathbf{w}} - \sum_{n=1}^{N} a_n t_n \widetilde{\mathbf{w}}^T\phi(x_n) + \sum_{n=1}^{N} a_n \tag{10}$$

$$D(\mathbf{a}) = \frac{1}{2}(\sum_{n=1}^{N} a_n t_n \phi(x_n))^T\mathbf{M}^{-1}\mathbf{M}(\sum_{m=1}^{N} a_m t_m \phi(x_m))\cdot\mathbf{M}^{-1} - \sum_{m=1}^{N} a_m t_m (\sum_{n=1}^{N} a_n t_n \phi(x_n))^T\cdot\mathbf{M}^{-1}\phi(x_m) + \sum_{n=1}^{N} a_n \tag{11}$$

$$D(\mathbf{a}) = \frac{1}{2}(\sum_{n=1}^{N} a_n t_n \phi(x_n))^T(\sum_{m=1}^{N} a_m t_m \phi(x_m))\cdot\mathbf{M}^{-1} - \sum_{m=1}^{N} a_m t_m (\sum_{n=1}^{N} a_n t_n \phi(x_n))^T\cdot\mathbf{M}^{-1}\phi(x_m) + \sum_{n=1}^{N} a_n \tag{12}$$

Since both sums have the same notation we can conflate them.

$$D(\mathbf{a}) = \frac{1}{2}(\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \phi(x_n)^T\phi(x_m))\cdot\mathbf{M}^{-1} - \sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \phi(x_n)^T\mathbf{M}^{-1}\phi(x_m) + \sum_{n=1}^{N} a_n \tag{13}$$

Then we need to pull $\mathbf{M}^{-1}$ out to match the first part of the equation.

$$D(\mathbf{a}) = \frac{1}{2}(\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \phi(x_n)^T\phi(x_m))\cdot\mathbf{M}^{-1} - (\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \phi(x_n)^T\phi(x_m))\cdot\mathbf{M}^{-1} + \sum_{n=1}^{N} a_n \tag{14}$$

$$D(\mathbf{a}) = -\frac{1}{2}(\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \phi(x_n)^T\phi(x_m)) \cdot \mathbf{M}^{-1} + \sum_{n=1}^{N} a_n \tag{15}$$

$$D(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}(\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \phi(x_n)^T\mathbf{M}^{-1}\phi(x_m)) \tag{16}$$

[1]

[1] Used material: `http://cs229.stanford.edu/notes/cs229-notes3.pdf`

3.2

Before deriving $D(\mathbf{a})$ we will rewrite it in matrix form to use matrix calculus, which will not only make derivation easier but will improve the runtime of the FISTA algorithm. The equation

$$D(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}(\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \phi(x_n)^T \mathbf{M}^{-1} \phi(x_m)) \tag{17}$$

results in

$$\mathbf{1}^T\mathbf{a} - \frac{1}{2}\mathbf{a}^T\mathbf{G}\mathbf{a}, \tag{18}$$

where $\mathbf{1}$ is a vector $\in R^N$ filled with 1s, $\mathbf{a}$ is the dual variable and $\mathbf{G}$ is the Gram Matrix with $G_{ij} = t_i t_j k_G(x_j, x_i)$.[2] Now $D(\mathbf{a})$ can be derived similarly to the first part of equation (7), however this time we will minimize the equation with respect to $\mathbf{a}$

$$\nabla D(\mathbf{a}) = \mathbf{1} - \mathbf{G}\mathbf{a}, \tag{19}$$

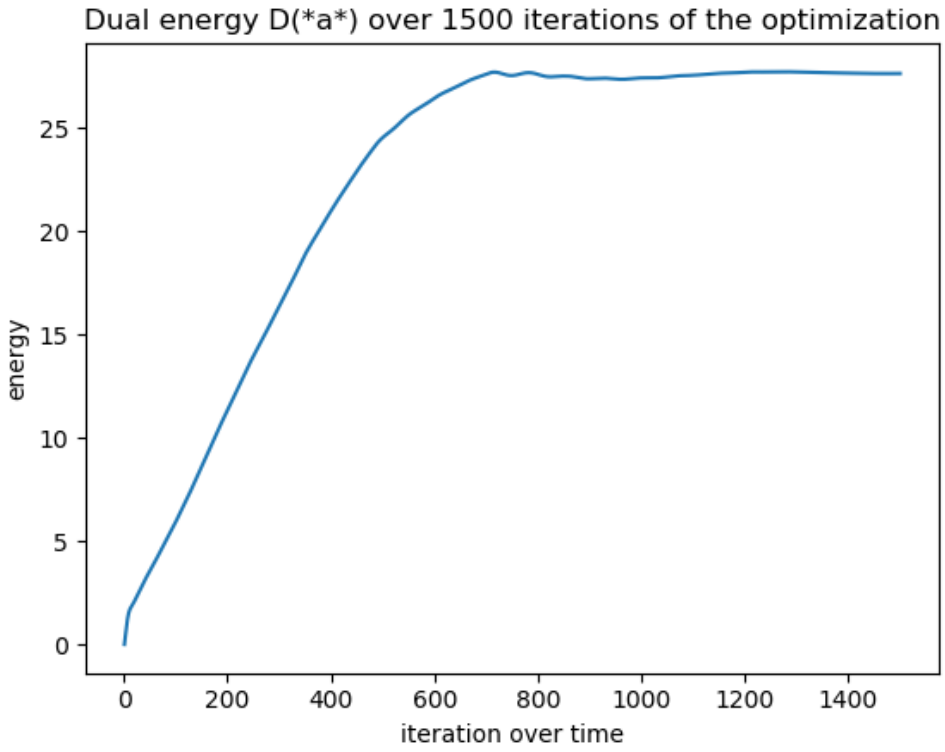
3.3

For implementation details please look at the code. The step size α, which was chosen by hand, depends on the used dataset. For the simple dataset a α of 0.0001 was used. For the half-moon dataset a α of 0.01 was used.

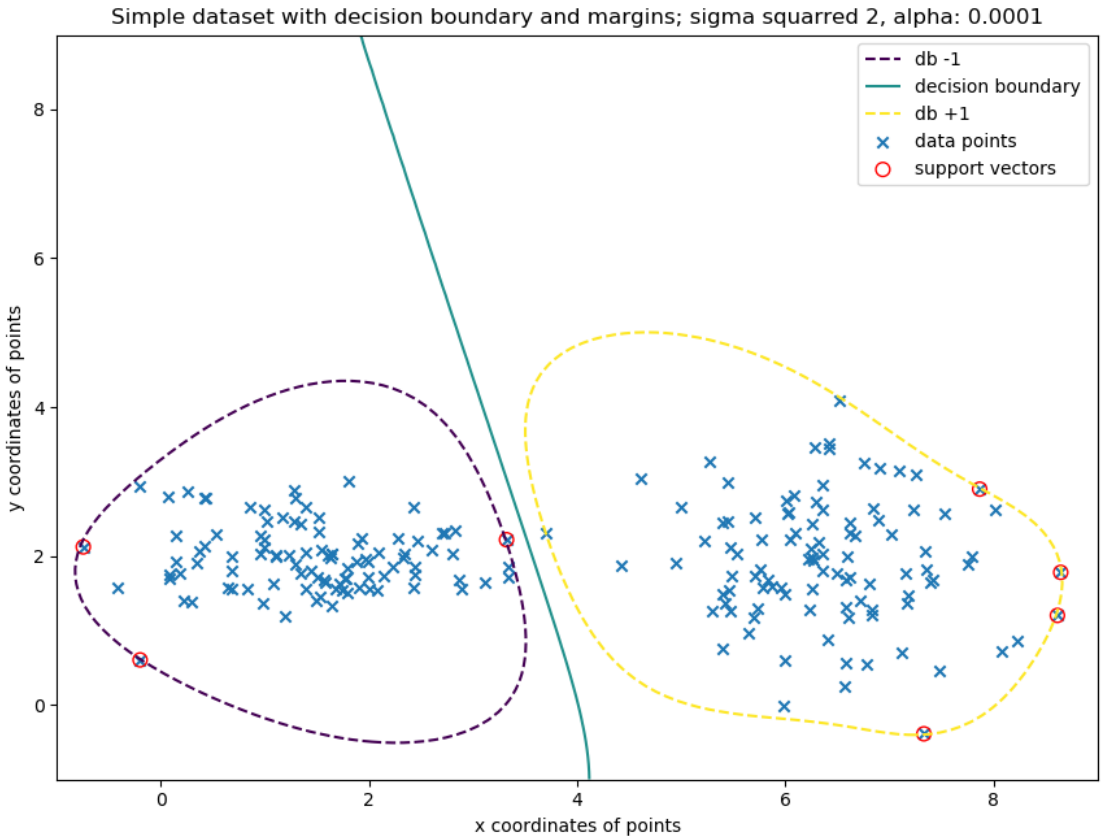
[2]Used material for matrix form: `https://cel.archives-ouvertes.fr/cel-01003007/file/Lecture2_Linear_SVM_Dual.pdf`

3.4

3.4.1 Simple Dataset

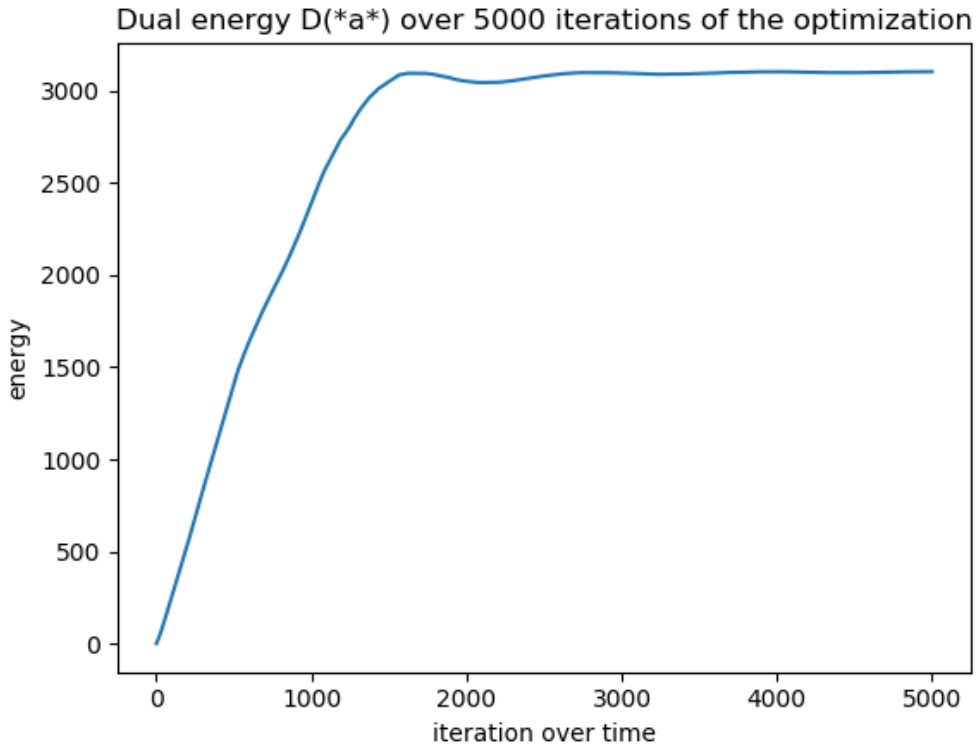


In the above figure the dual energy $D(\mathbf{a})$ over the iteration of the optimization can be seen. For the simple dataset we chose 1500 iterations for the optimization. It is worth noting that the FISTA algorithm converges after around 700 iterations.

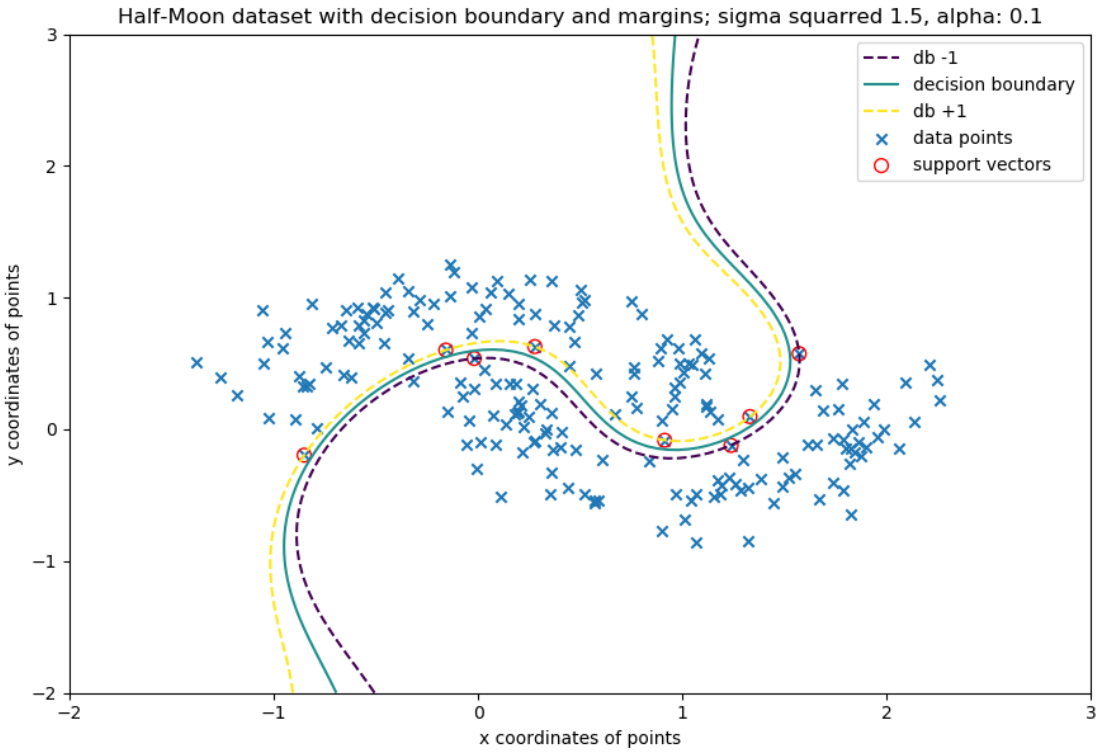


In the above figure the whole dataset together with the computed decision boundary and the margins can be seen. Using a bigger σ results in a very round (depending on the data) and smooth decision boundary. A smaller σ results in a more edged decision boundary. Same applies for the margins - a smaller σ results in a tightly fitting margin, while a bigger one results in a generously fitting one. For generalization purpose we suggest using a bigger sigma. In this example we used a σ^2 of 2.

3.4.2 Halfmoon Dataset



In the above figure the dual energy $D(\mathbf{a})$ over the iteration of the optimization can be seen. For the half-moon dataset we chose 5000 iterations for the optimization. It is worth noting that the FISTA algorithm converges after around 1700 iterations.



In the above figure the whole dataset together with the computed decision boundary and the margins can be seen. Using a bigger σ makes the margin very extensive, while a smaller one is tightly fitting the datasets. For generalization purpose we suggest using a bigger sigma. In this example we used a σ^2 of 1.5.