Technical University of Crete

School of Electrical and Computer Engineering

Course : **Optimization**

Exercise 4

Instructor : Athanasios P. Liavas

Students : Tzimpimpaki Evangelia - 2018030165

Toganidis Nikos - 2018030085

Support Vector Machines for Linearly Separable Data

The purpose of this exercise is to study the application of Support Vector Machines (SVMs) to a simple binary classification problem. Generally, we know that the goal of this method is setting a boundary that maximizes the margin between the nearest data points of each class. So, we need to determine the boundary that makes the best separation of the 2 classes. In our case (classification of linearly separable data), the boundary is a hyperplane $\mathbb{H}_{\mathbf{a}_*,b_*} := \{\mathbf{x} \in \mathbb{R}^n | \, \mathbf{a}_*^\mathbf{T}\mathbf{x} = b_*\}$.

Worth noting that for the solution of the unconstrained convex optimization problem we used the Newton's algorithm. However, we adjusted the cost function g_k so that the function value for feasible points remains the same, but for infeasible points the function value is increased to the maximum possible. With this adjustment, the Newton algorithm minimizes in the feasible region, while preventing consideration of an infeasible point (outside the barrier) to a minimum. This is the main reason we used the barrier function.

1. Function **flag** = point_is_feasible(**w_init, X_augm, y**)

This function returns 1 if $w\_init \in dom\phi$ and 0 otherwise.

$\phi(\mathbf{w}) := -\sum_{i=1}^{n} log(-f_i(\mathbf{w}))$

$dom\phi = \{\mathbf{w}| - f_i(\mathbf{w}) > 0\} = \{\mathbf{w}|f_i(\mathbf{w}) < 0\} = \{\mathbf{w}|1 - y_i\mathbf{w}^\mathbf{T}\bar{\mathbf{x}}_\mathbf{i} < 0\}$

In MATLAB we construct the function f_i as described above. The returned value flag is 1 when $f_i(\mathbf{w}) < 0$ and 0 otherwise.

2. Function **grad** = gradient_SVM_barrier(**w, X_augm, y**,t)

This function computes the gradient of the cost function g_k at the point $\mathbf{w}$.

$\nabla g_k(\mathbf{w}) = t_k\mathbf{w} + \sum_{i=1}^{N} \frac{y_i}{1 - y_i\bar{\mathbf{x}}_\mathbf{i}^\mathbf{T}\mathbf{w}}\bar{\mathbf{x}}_\mathbf{i}$

In a for loop for the $N = 1000$ data points, we store in the variable **grad** the content of the sum above. In every loop we increase **grad** so that finally it contains the sum for all data points.

3. Function **Hess** = Hess_SVM_barrier(**w, X_augm, y**,t)

This function computes the Hessian of the cost function g_k at the point $\mathbf{w}$.

$\nabla^2 g_k(\mathbf{w}) = t_k\mathbf{I} + \sum_{i=1}^{N} \frac{1}{(y_i\bar{\mathbf{x}}_\mathbf{i}^\mathbf{T}\mathbf{w} - 1)^2}\bar{\mathbf{x}}_\mathbf{i}\bar{\mathbf{x}}_\mathbf{i}^\mathbf{T}$

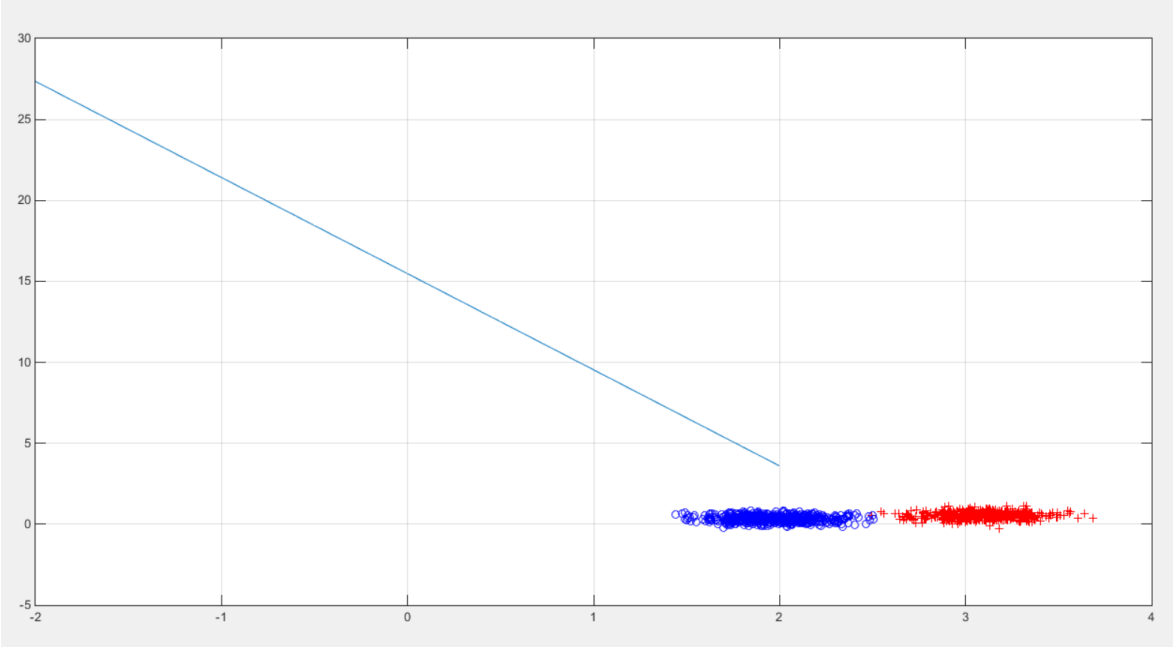
Just like before, we calculate the Hessian of g_k from the form above.

4. Function gk = barrier_SVM_cost_function(**w, X_augm, y**,t)

We calculate the value of $g_k(\mathbf{w}) := t_k f_0(\mathbf{w}) + \phi(\mathbf{w})$, where $f_0(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||_2^2$ and $\phi(\mathbf{w}) = -\sum_{i=1}^{n} log(-f_i(\mathbf{w})) = -\sum_{i=1}^{n} log(y_i\mathbf{w}^\mathbf{T}\bar{\mathbf{x}}_\mathbf{i} - 1)$.

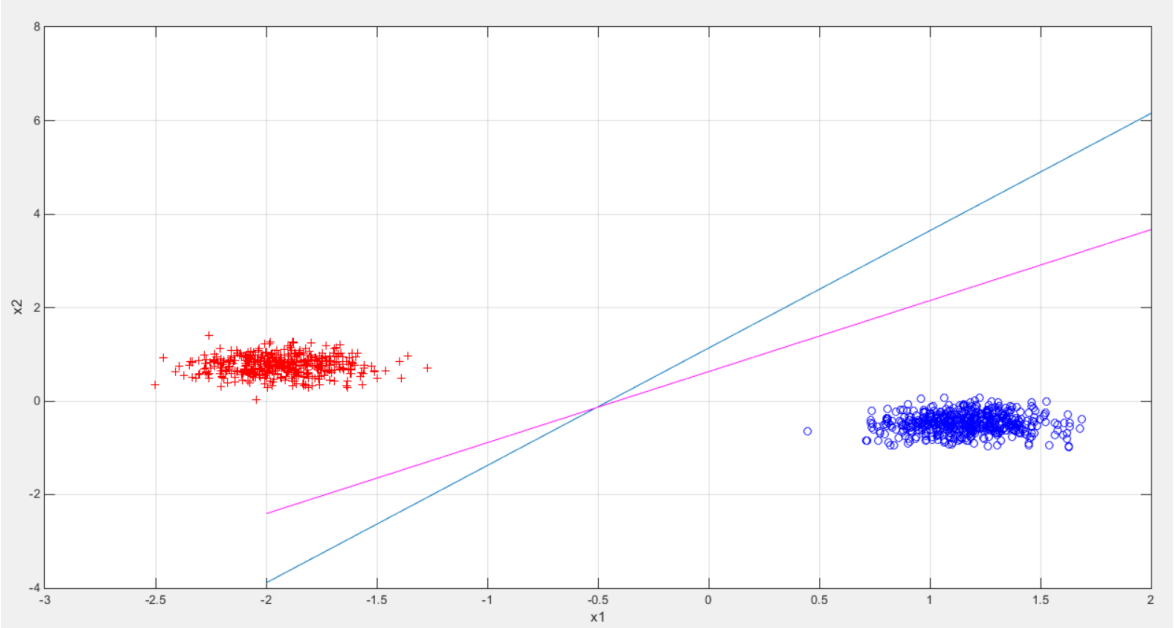
RESULTS :

- In case the data is non-separable a message is printed and the program stops.



- For linearly separable data, the classification problem is first solved using the cvx and then via interior point method.

  In the plot bellow, the blue line is the "true" separating hyperplane and the line in magenta is the SVM-based separating hyperplane. We can see that the cvx provides a good approximation of the real solution.



  As for the interior point method, for every iteration a new figure is being plotted. The optimization begins from the previous solution $\mathbf{w_{init}}(:, \mathbf{1}) = [b; \mathbf{a}]$, where $\mathbf{a}$ and b determine the "true" solution above $\mathbf{a^T x} = b$. Vector $\mathbf{w}$ is being updated in a way that the new point $\mathbf{w_{new}}$ remains feasible (this means that $\mathbf{w_{new}} \in dom\phi$). When this requirement is not fulfilled, the message "New point is infeasible... I backtrack..." is printed at the command window. Then, backtracking line search is performed for the function g_k (at this point a similar message is printed again, to inform the user for the backtracking). During runtime the user can see how the current estimate gets closer and closer to the optimal solution, until they finally identify, as desired.

Results can be seen bellow :