MACHINE LEARNING HOMEWORK SHEET - 06

CONSTRAINT OPTIMIZATION AND SVM

NAME: MAHALAKSHMI SABANAYAGAM

TUM ID: ge73yuw

DATE: 2.12.2018

Problem 1:

$$\min f_0(\theta) = \theta_1 - \sqrt{3}\,\theta_2$$

$$\text{subject to } f_1(\theta) = \theta_1^2 + \theta_2^2 - 4 \leq 0$$

Following the recipe,

1.) Lagrangian,

$$L(\theta, \alpha) = f_0(\theta) + \alpha f_1(\theta)$$

$$= \theta_1 - \sqrt{3}\theta_2 + \alpha(\theta_1^2 + \theta_2^2 - 4)$$

2.) Lagrangian Dual function $g(\alpha)$

$$\underset{\theta}{\operatorname{argmin}}\, L(\theta, \alpha) \Rightarrow \nabla_\theta L(\theta, \alpha) = 0$$

$$\therefore \nabla_\theta L(\theta, \alpha) = \begin{bmatrix} 1 + 2\alpha\theta_1 \\ -\sqrt{3} + 2\alpha\theta_2 \end{bmatrix} \overset{!}{=} \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$1 + 2\alpha\theta_1 = 0 \qquad -\sqrt{3} + 2\alpha\theta_2 = 0$$

$$\theta_1 = \frac{-1}{2\alpha} \qquad \theta_2 = \frac{\sqrt{3}}{2\alpha}$$

$$g(\alpha) = L(\theta^*, \alpha)$$

$$= \frac{-1}{2\alpha} - \sqrt{3}\frac{(\sqrt{3})}{2\alpha} + \alpha\left[\frac{1}{4\alpha^2} + \frac{3}{4\alpha^2} - 4\right]$$

$$= -\frac{2}{\alpha} + \frac{1}{\alpha} - 4\alpha = -\frac{1}{\alpha} - 4\alpha$$

3.) Solve the dual problem:

$$\max g(\alpha) = \frac{-1}{\alpha} - 4\alpha$$

$$\text{subject to } \alpha \geq 0$$

$$g'(\alpha) = +\frac{1}{\alpha^2} - 4 \overset{!}{=} 0$$

$$\alpha = \pm\frac{1}{2}$$

As $\alpha \geq 0$, $\alpha = \frac{1}{2}$

$\theta_1 = -\frac{1}{2\alpha} = -1$

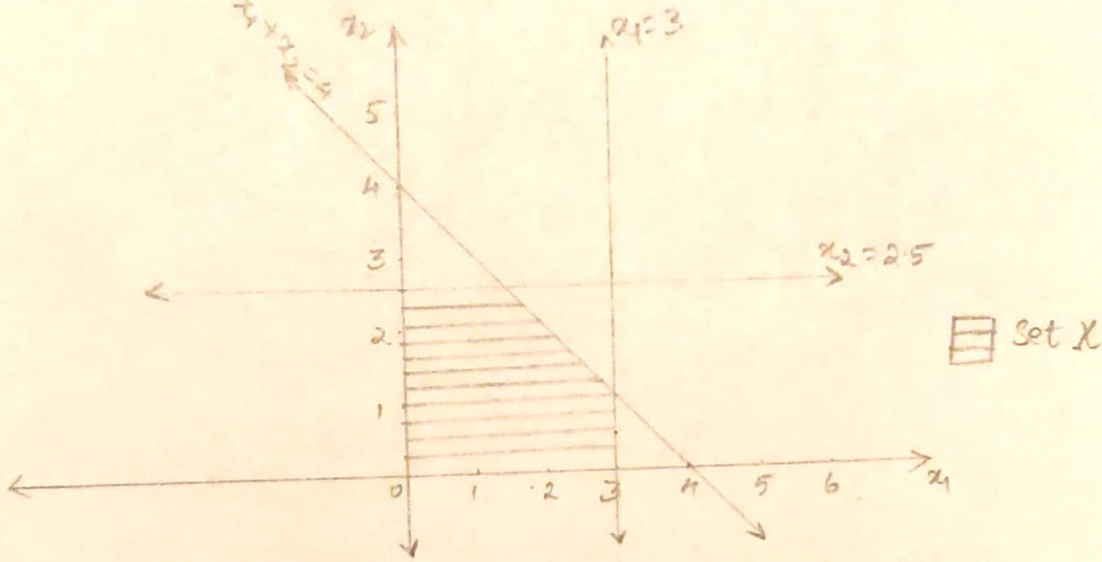
$\theta_2 = \frac{\sqrt{3}}{2\alpha} = \sqrt{3}$

$\therefore$ At $\theta_1 = -1$, $\theta_2 = \sqrt{3}$, the function $f_0(\theta)$ attains minimum with the constraint $f_1(\theta)$.

Problem 2:

$\mathcal{X} \subset \mathbb{R}^2 = \{x \in \mathbb{R}^2 : (x_1 + x_2 \leq 4) \wedge (0 \leq x_1 \leq 3) \wedge (0 \leq x_2 \leq 2.5)\}$

a. Visualise set $\mathcal{X}$:



b. $p \in \mathbb{R}^2$. Projection on $\mathcal{X}$?

$\Pi_{\mathcal{X}}(p) = \underset{x \in \mathcal{X}}{\operatorname{argmin}} \|x - p\|_2^2 \rightarrow$ L2 distance from point p to the space.

(*) $\forall\ x_1 < 0$, $x_2 > 2.5$ $\Pi_{\mathcal{X}}(p) = (0, 2)$ where p is (x_1, x_2)

(*) $x_1 < 0$, $0 < x_2 < 2.5$ $\Pi_{\mathcal{X}}(p) = (0, x_2)$

(*) $x_1 < 0$, $x_2 < 0$ $\Pi_{\mathcal{X}}(p) = (0, 0)$

(*) $0 < x_1 \leq 3$, $x_2 < 0$ $\Pi_{\mathcal{X}}(p) = (x_1, 0)$

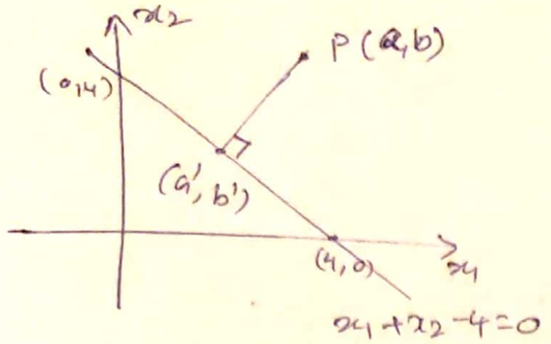
(*) $x_1 > 3$, $x_2 < 0$ $\Pi_{\mathcal{X}}(p) = (3, 0)$

(*) $x_1 > 3$, $0 < x_2 < 1$ $\Pi_{\mathcal{X}}(p) = (3, x_2)$

(*) $0 < x_1 < 1.5$, $2.5 < x_2$ $\Pi_{\mathcal{X}}(p) = (x_1, 2.5)$

(*) For the points which are not in space X and doesnot fall under any of the above conditions,

$\Pi_X(P)$ = Project P onto line $x_1+x_2-4=0$



slope of $x_1+x_2-4=0$ is -1

$$\therefore \frac{b'-0}{a'-4} = -1$$

$$b' = -a'+4 \rightarrow (1)$$

$$b'+a' = +4$$

(a,b) (a', b') slope is ⊥r to the line.

$\Rightarrow$ slope of (a,b) (a', b') = 1

$$\frac{b'-b}{a'-a} = 1$$

$$b'-b = a'-a$$

$$b'-a' = b-a \rightarrow (2)$$

From (1), (2)

$$2b' = b-a+4$$

$$b' = \frac{b-a+4}{2}$$

$$a' = 4 - \frac{b-a+4}{2} = \frac{a-b+4}{2}$$

$\therefore$ Projection is $\left(\frac{a-b+4}{2}, \frac{b-a+4}{2}\right)$ for P(a,b)

If $\frac{a-b+4}{2} > 3$ then projection is (3,1)

If $\frac{b-a+4}{2} < 1.5$ then projection is (1.5, 2.5)

(*) $\forall (x_1, x_2) \in X$ $\quad \Pi_X(P) = (x_1, x_2) \Rightarrow$ ~~for~~ Points in the space X.

c. $\min \quad (x_1-2)^2 + (2x_2-7)^2$

subject to $x \in \mathcal{X}$. $\quad \gamma = 0.05$

$$x^{(0)} = \begin{bmatrix} 2.5 \\ 1 \end{bmatrix}$$

$$\text{gradient } \nabla_x = \begin{bmatrix} 2(x_1-2) \\ 4(2x_2-7) \end{bmatrix}$$

$$x^{(1)} = x^{(0)} - \gamma \nabla x$$

$$= \begin{bmatrix} 2.5 \\ 1 \end{bmatrix} - 0.05 \begin{bmatrix} 2(2.5-2) \\ 4(2(1)-7) \end{bmatrix} = \begin{bmatrix} 2.45 \\ 2 \end{bmatrix}$$

$(2.45, 2) \notin \mathcal{X}$

$\therefore$ Projection is $\left(\frac{a-b+4}{2}, \frac{b-a+4}{2}\right)$ from previous problem

where $a = 2.45$, $b = 2$

$$\text{projection} = \left(\frac{4.45}{2}, \frac{3.55}{2}\right) = (2.225, 1.775)$$

$$x^{(2)} = x^{(1)} - \gamma \nabla x$$

$$= \begin{bmatrix} 2.225 \\ 1.775 \end{bmatrix} - 0.05 \begin{bmatrix} 2(0.225) \\ 4(-3.45) \end{bmatrix} = \begin{bmatrix} 2.2025 \\ 2.465 \end{bmatrix}$$

$(2.2025, 2.465) \notin \mathcal{X}$

$$\therefore \text{Projection} = \left(\frac{3.7375}{2}, \frac{4.2625}{2}\right)$$

$$= (1.86875, 2.13125)$$

$$x^{(2)} = (1.86875, 2.13125).$$

Problem 3 :

Similarities and differences between SVM and Perceptron algorithm.

* Perceptron algorithm finds ~~the~~ a hyperplane that separates the points by class , if ~~the~~ a hyperplane exists.
  The plane obtained depends on the order in which the points are processed.

  SVM also finds the hyperplane that separates the points by class, it gives the most optimum plane that has the makimum margin , if ~~the~~ a hyperplane exists.
  The plane obtained doesnot depend on the order of processing the points.

* Adding new sample points changes the hyperplane obtained in Perceptron algorithm. Whereas, the new points have an effect on the hyperplane from SVM only if they lie ~~is~~ within the margin.

* After obtaining the hyperplane, we can ignore all the samples except the support vectors in case of SVM, That can't be done for Perceptron.

Problem 4 :

Duality gap $= 0$ for SVM.

Duality gap $=$ primal $-$ dual problem

$$= f_0(x^*) - g(\alpha^*)$$

For SVM,

$$f_0(x) = \frac{1}{2} w^T w$$

$$g(\alpha) = \alpha^T 1_N - \frac{1}{2} \alpha^T Q \alpha \quad \text{where } Q \text{ is } \underbrace{YY^T \circ XX^T}_{\text{Hadamard product}}$$

while solving the Lagrangian we arrived

at $w = \sum \alpha_i y_i x_i$

At optimum $\boxed{w^* = \sum \alpha_i^* y_i x_i} \rightarrow (1)$

α^* in Dual problem,

$$g(\alpha) = \alpha^T 1_N - \frac{1}{2}\alpha^T Q \alpha$$

$$g'(\alpha^*) = 0 = 1_N - \alpha^T Q$$

$$\therefore \boxed{(\alpha^*)^T Q = 1_N} \quad - (2)$$

Now, calculating the duality gap

$$f_0(x^*) - g(\alpha^*)$$

$$= \frac{1}{2}(w^*)^T w^* - \alpha^{*T} 1_N + \frac{1}{2}\alpha^{*T} Q \alpha^*$$

$$= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i^* \alpha_j^* y_i y_j x_i x_j - \alpha^{*T} 1_N + \frac{1}{2}\alpha^{*T} Q \alpha^*$$

$$= \frac{1}{2}(\alpha^*)^T Q \alpha^* - (\alpha^*)^T 1_N + \frac{1}{2}(\alpha^*)^T Q \alpha^*$$

$$= \underbrace{(\alpha^*)^T Q}\alpha^* - (\alpha^*)^T 1_N$$

From 2 $(\alpha^*)^T Q = 1_N$, Substituting here

$$= 1_N \alpha^* - (\alpha^*)^T 1_N$$

$$= (\alpha^*)^T 1_N - (\alpha^*)^T 1_N = 0$$

$\therefore$ Duality gap is 0 for SVM.

Also, going by Slater's constraint,

$f_0(w) = \frac{1}{2} w^T w \Rightarrow$ convex (quadratic and $w^T w$ is +ve)

$f_i(w,b) = -y_i(w^T x_i + b) + 1 \leq 0 \quad i = 1, \ldots, N \Rightarrow$ affine constraints

$\therefore$ Strong duality holds by Slater's constraint qualification. Strong duality implies duality gap is 0

Problem 5:

$$g(\alpha) = \frac{1}{2} \alpha^T Q \alpha + \alpha^T 1_N$$

a.) Q ?

$$g(\alpha) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i x_j + \sum \alpha_j$$

$$\downarrow \qquad\qquad\qquad \downarrow$$

$$\frac{1}{2} \alpha^T Q \alpha \qquad\qquad \alpha^T 1_N$$

$$Q = -\sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j x_i x_j$$

$$= - YY^T \circ XX^T$$

$\circ$ ↓ Hadamard product

b.) Prove Q is negative semidefinite.

$$Q = -YY^T \circ XX^T$$

X dimension $= N \times d$

Y dimension $= N \times 1$

YY^T is symmetric (+ matrix) $\qquad YY^T = N \times N$

XX^T is symmetric (+ matrix) $\qquad XX^T = N \times N$

Doing Hadamard product on YY^T and XX^T produces a matrix which will also be symmetric

$\therefore$ $Q = -$ (symmetric matrix)

⇓

$-$ (all the elements are +ve as YY^T & XX^T will have only +ve elements)

$\therefore$ Q has all negative elements and it is symmetric $\Rightarrow$ negative semidefinite

Semi definite as one of the diagonal elements could be 0.

c.) Importance of negative semidefiniteness for our optimization problem:

max $g(\alpha)$ is our dual problem,

$$\max \frac{1}{2}\alpha^T Q \alpha + \alpha^T 1_N$$

$$\equiv \min -\frac{1}{2}\alpha^T Q \alpha - \alpha^T 1_N = \min \frac{1}{2}\alpha^T(-Q)\alpha - \alpha^T 1_N$$

$$-Q = YY^T \circ XX^T$$

$-Q$ is positive semidefinite

$\Rightarrow$ ~~max~~ $\frac{1}{2}\alpha^T(-Q)\alpha - \alpha^T 1_N$ is convex and has a solution

As it is convex, ~~we~~ SVM ~~has~~ gives single optimum solution.

Programming assignment 6: SVM

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.datasets import make_blobs

from cvxopt import matrix, solvers
```

Exporting the results to PDF

Once you complete the assignments, export the entire notebook as PDF and attach it to your homework solutions. The best way of doing that is

1. Run all the cells of the notebook.
2. Download the notebook in HTML (click File > Download as > .html)
3. Convert the HTML to PDF using e.g. https://www.sejda.com/html-to-pdf or `wkhtmltopdf` for Linux (tutorial)
4. Concatenate your solutions for other tasks with the output of Step 3. On a Linux machine you can simply use `pdfunite`, there are similar tools for other platforms too. You can only upload a single PDF file to Moodle.

This way is preferred to using `nbconvert`, since `nbconvert` clips lines that exceed page width and makes your code harder to grade.

Your task

In this sheet we will implement a simple binary SVM classifier.

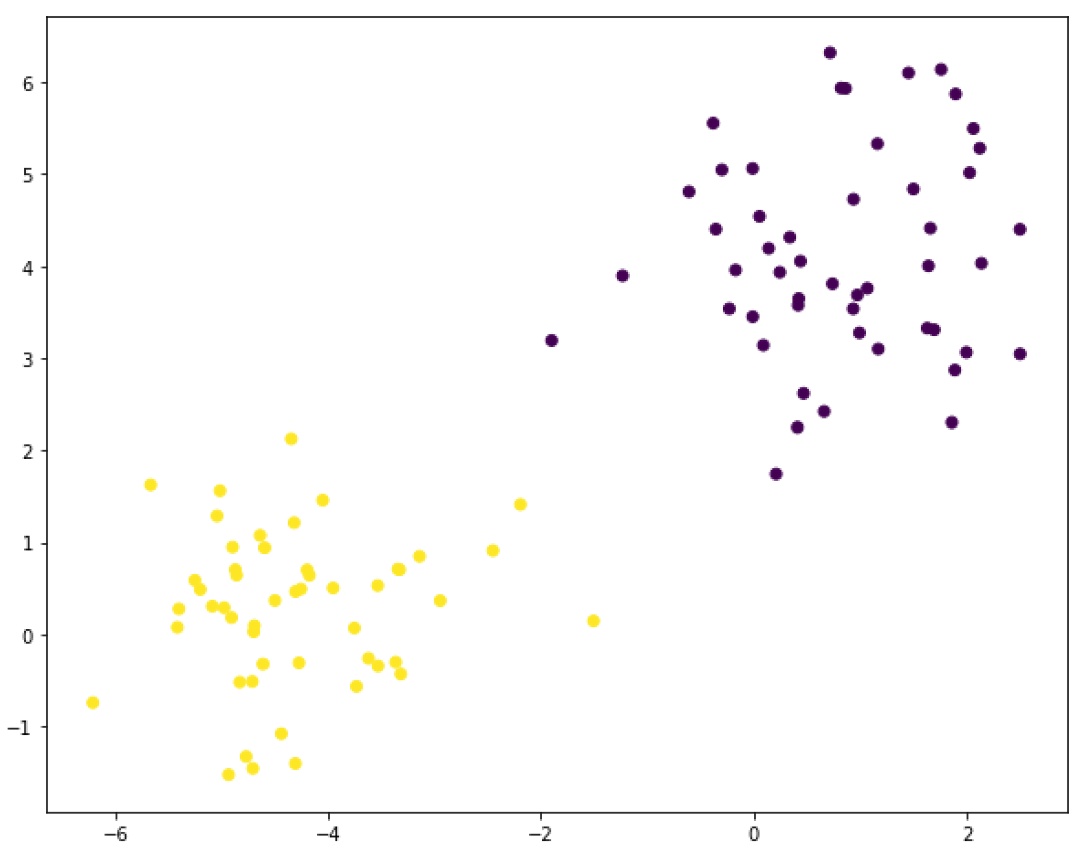
We will use **`CVXOPT`** http://cvxopt.org/ - a Python library for convex optimization. If you use `Anaconda`, you can install it using

```
conda install cvxopt
```

As usual, your task is to fill out the missing code, run the notebook, convert it to PDF and attach it you your HW solution.

Generate and visualize the data

In [2]:
```
N = 100  # number of samples
D = 2  # number of dimensions
C = 2  # number of classes
seed = 3  # for reproducible experiments

X, y = make_blobs(n_samples=N, n_features=D, centers=2, random_state=seed)
y[y == 0] = -1  # it is more convenient to have {-1, 1} as class labels (instead of {0, 1})
y = y.astype(np.float)
plt.figure(figsize=[10, 8])
plt.scatter(X[:, 0], X[:, 1], c=y)
plt.show()
```



Task 1: Solving the SVM dual problem

Remember, that the SVM dual problem can be formulated as a Quadratic programming (QP) problem. We will solve it using a QP solver from the `CVXOPT` library.

The general form of a QP is

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} - \mathbf{q}^T\mathbf{x}$$

$$\text{subject to} \quad \mathbf{G}\mathbf{x} \preceq \mathbf{h}$$

$$\text{and} \quad \mathbf{A}\mathbf{x} = \mathbf{b}$$

where $\preceq$ denotes "elementwise less than or equal to".

Your task is to formulate the SVM dual problems as a QP and solve it using `CVXOPT`, i.e. specify the matrices $\mathbf{P}$, $\mathbf{G}$, $\mathbf{A}$ and vectors $\mathbf{q}$, $\mathbf{h}$, $\mathbf{b}$.

In [3]:
```
def solve_dual_svm(X, y):
    """Solve the dual formulation of the SVM problem.

    Parameters
    ----------
    X : array, shape [N, D]
        Input features.
    y : array, shape [N]
        Binary class labels (in {-1, 1} format).

    Returns
    -------
    alphas : array, shape [N]
        Solution of the dual problem.
    """
    # TODO
    # These variables have to be of type cvxopt.matrix
    n,d = X.shape;
    y = y.reshape(-1,1);
    temp = y * X;
    P = matrix(np.dot(temp, temp.T));
    q = matrix(-np.ones((n,1)));
    G = matrix(-np.eye(n));
    h = matrix(np.zeros(n));
    A = matrix(y.reshape(1,-1));
    b = matrix(np.zeros(1));
    solvers.options['show_progress'] = False
    solution = solvers.qp(P, q, G, h, A, b)
    alphas = np.array(solution['x'])
    return alphas
```

Task 2: Recovering the weights and the bias

In [4]:
```
def compute_weights_and_bias(alpha, X, y):
    """Recover the weights w and the bias b using the dual solution alpha.

    Parameters
    ----------
    alpha : array, shape [N]
        Solution of the dual problem.
    X : array, shape [N, D]
        Input features.
    y : array, shape [N]
        Binary class labels (in {-1, 1} format).

    Returns
    -------
    w : array, shape [D]
        Weight vector.
    b : float
        Bias term.
    """
    w = np.sum(alpha * y.reshape(-1,1) * X, axis = 0);
    cond = (alpha > 1e-4).reshape(-1);
    b = y[cond] - np.dot(X[cond], w);
    return w, b[0]
```

Visualize the result (nothing to do here)

In [5]:
```
def plot_data_with_hyperplane_and_support_vectors(X, y, alpha, w, b):
    """Plot the data as a scatter plot together with the separating hyperplane.

    Parameters
    ----------
    X : array, shape [N, D]
        Input features.
    y : array, shape [N]
        Binary class labels (in {-1, 1} format).
    alpha : array, shape [N]
        Solution of the dual problem.
    w : array, shape [D]
        Weight vector.
    b : float
        Bias term.
    """
    plt.figure(figsize=[10, 8])
    # Plot the hyperplane
    slope = -w[0] / w[1]
    intercept = -b / w[1]
    x = np.linspace(X[:, 0].min(), X[:, 0].max())
    plt.plot(x, x * slope + intercept, 'k-', label='decision boundary')
    # Plot all the datapoints
    plt.scatter(X[:, 0], X[:, 1], c=y)
    # Mark the support vectors
    support_vecs = (alpha > 1e-4).reshape(-1)
    plt.scatter(X[support_vecs, 0], X[support_vecs, 1], c=y[support_vecs], s=250, marker='*', label=
'support vectors')
    plt.xlabel('$x_1$')
    plt.ylabel('$x_2$')
    plt.legend(loc='upper left')
```

The reference solution is

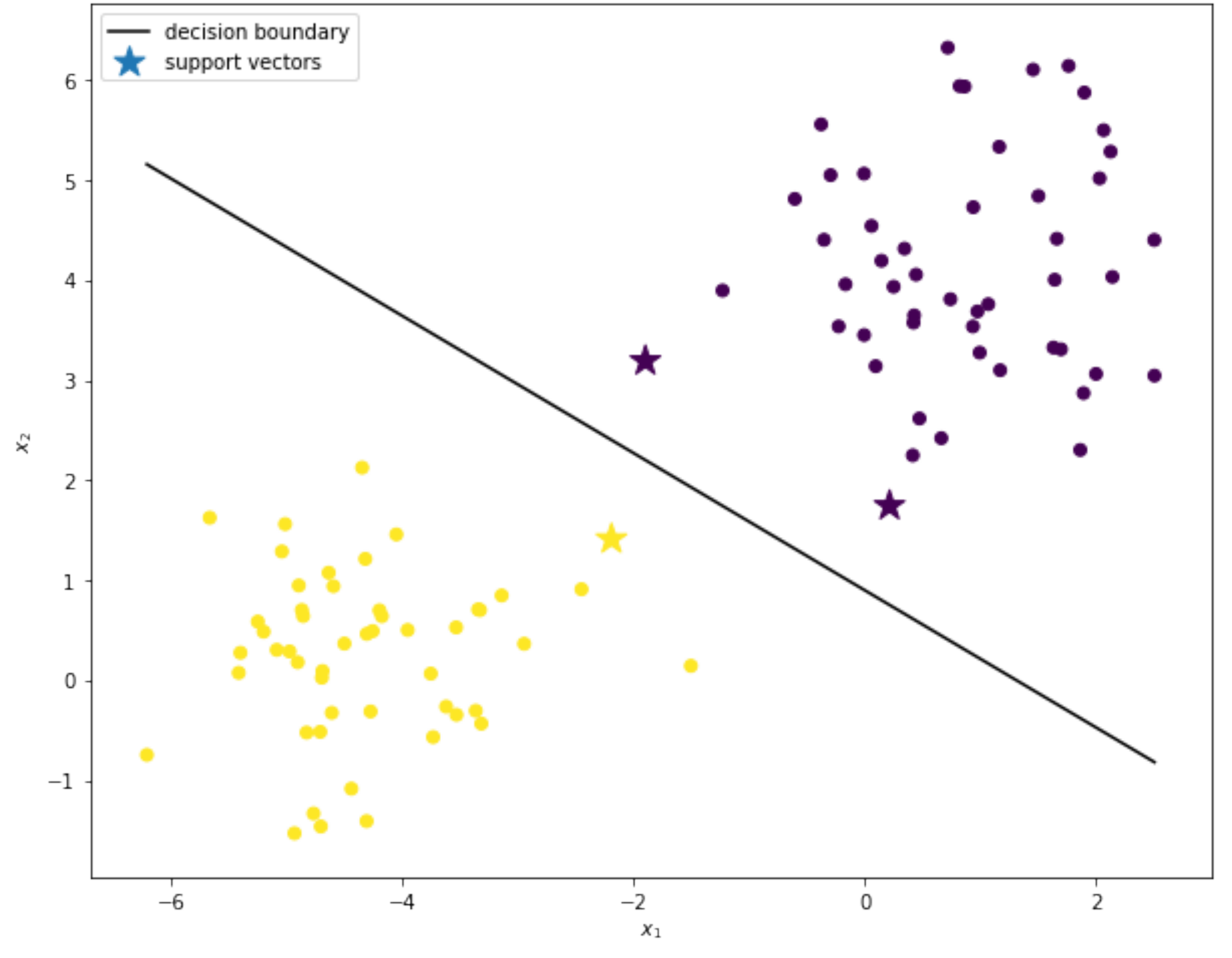
```
w = array([[-0.69192638],
           [-1.00973312]])

b = 0.907667782
```

Indices of the support vectors are

```
[38, 47, 92]
```

In [7]:
```
alpha = solve_dual_svm(X, y)
w, b = compute_weights_and_bias(alpha, X, y)
print (w,b);
plot_data_with_hyperplane_and_support_vectors(X, y, alpha, w, b)
plt.show()
```

```
[-0.69192638 -1.00973312] 0.9076678239696534
```



decision boundary
support vectors
x_2
x_1
6
5
4
3
2
1
0
−1
−6
−4
−2
0
2