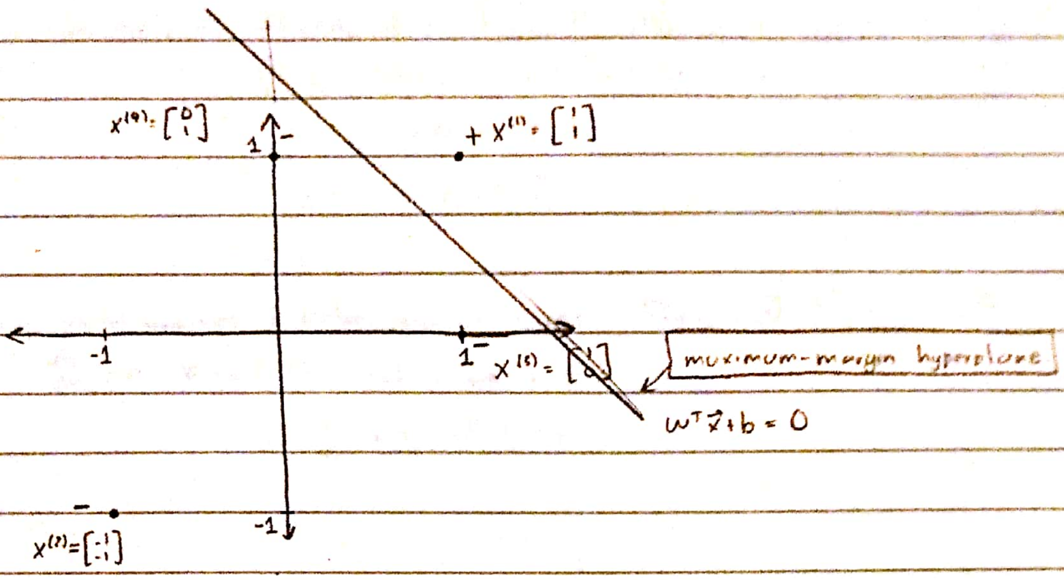
ECE421 Assignment

Problem 1)

1) A plot of the dataset of 4 points $\{x^{(1)} = \begin{bmatrix}1\\1\end{bmatrix}, x^{(2)} = \begin{bmatrix}-1\\-1\end{bmatrix}, x^{(3)} = \begin{bmatrix}1\\0\end{bmatrix}, x^{(4)} = \begin{bmatrix}0\\1\end{bmatrix}\}$ is shown below



By inspection: Line (hyperplane) passes through $(0, 3/2)$ and $(3/2, 0)$

Equation of line: $m = \frac{3/2 - 0}{0 - 3/2} = -1$ $\quad b = 3/2 \Rightarrow y = -x + 3/2$

Since these are both input variables, replace x, y with x_1, x_2 and change into the form $w^T x + b = 0$

$$x_2 = -x_1 + 3/2$$

$$x_1 + x_2 - 3/2 = 0$$

$$2(x_1 + x_2) - 3 = 0$$

$$\begin{bmatrix}2 & 2\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} - 3 = 0$$

So, the equation of this hyperplane is $w^T x + b = 0$, where $w = \begin{bmatrix}2\\2\end{bmatrix}$ and $b = -3$

2) Support vectors are data points closest to hyperplane; the points which influence the position and orientation of the hyperplane. In our dataset, these points are $x^{(1)}$, $x^{(3)}$ and $x^{(4)}$

3) In order to construct the Lagrangian, we will list the function we want to minimize and our constraints, then express the Lagrangian using the general formula $L(\vec{x}, \vec{\lambda}) = f_0(\vec{x}) + \sum_{i=1}^{n} \lambda_i f_i(x)$, where f_0 is the function to minimize and $f_i(x) \le 0$ are our constraints

minimize: $\frac{1}{2}\|\vec{w}\|^2$

constraints: $1 - (\vec{w}\cdot\vec{x}^{(j)} + b)y^{(j)} \le 0 \quad \forall j \in \{1, 2, 3, 4\}$

$$L(\vec{w}, b, \vec{\alpha}) = \frac{1}{2}\|\vec{w}\|^2 + \sum_{j=1}^{4} \alpha_j (1 - (\vec{w}\cdot\vec{x}^{(j)} + b)y^{(j)})$$

$$\boxed{L(\vec{w}, b, \alpha) = \frac{1}{2}(w_1^2 + w_2^2) + \sum_{j=1}^{4} \alpha_j (1 - (\vec{w}\cdot\vec{x}^{(j)} + b)y^{(j)})}$$ since there are 4 inputs and $\vec{w} \in \mathbb{R}^2$

4) We want to solve:

$$\min_{\vec{w},b} L(\vec{w},b,\vec{\alpha}) = \min_{\vec{w},b} \tfrac{1}{2}(w_1^2+w_2^2) + \sum_j \alpha_j (1-(\vec{w}\cdot\vec{x}^{(j)}+b)y^{(j)})$$

At the minimum, the partial derivatives are 0 (critical point). Setting partial derivatives to 0 gives

$$\frac{\partial L}{\partial w_i} = \tfrac{1}{2}(2w_i) + \sum_j \alpha_j \left(\frac{\partial}{\partial w_i}(-w_i \vec{x}_i^{(j)} y^{(j)})\right)$$

$$= w_i + \sum_j \alpha_j (-\vec{x}_i^{(j)} y^{(j)})$$

$$\frac{\partial L}{\partial w_i} = w_i - \sum_j \alpha_j y^{(j)} \vec{x}_i^{(j)}$$

Combining partial derivatives to obtain gradient gives: $\nabla_{\vec{w}} L = \vec{w} - \sum_j \alpha_j y^{(j)} \vec{x}^{(j)}$

Since $\frac{\partial L}{\partial w_i} = 0 \quad \forall i \in \{1,2\}$, set gradient to 0:

$$\vec{w} - \sum_j \alpha_j y^{(j)} \vec{x}^{(j)} = 0$$

$$\boxed{\vec{w} = \sum_j \alpha_j y^{(j)} \vec{x}^{(j)}}$$

Now, b is also an independent variable, so we also set $\frac{\partial L}{\partial b} = 0$. This gives:

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \frac{\partial}{\partial b}\left[\tfrac{1}{2}(w_1^2+w_2^2) + \sum_j \alpha_j (1-(\vec{w}\cdot\vec{x}^{(j)}+b)y^{(j)})\right] = 0$$

$$\frac{\partial}{\partial b}\left[\sum_j \alpha_j (1-\vec{w}\cdot\vec{x}^{(j)} y^{(j)} - by^{(j)})\right] = 0$$

$$\frac{\partial}{\partial b} \sum_j \alpha_j (-by^{(j)}) = 0$$

$$-\sum_j \alpha_j y^{(j)} = 0$$

$$\boxed{\sum_j \alpha_j y^{(j)} = 0}$$

5) We must simplify equation 2 with results of equation 4

Results of equation 4 (1): $\left.\begin{array}{l} \vec{w} = \sum_j \alpha_j y^{(j)} \vec{x}^{(j)} \\ \sum_j \alpha_j y^{(j)} = 0 \end{array}\right\}$ when L is minimized

So, to simplify $\max_{\vec{\alpha}} \min_{\vec{w},b} \tfrac{1}{2}\|\vec{w}\|^2 + \sum_j \alpha_j (1-(\vec{w}^T\vec{x}^{(j)}+b)y^{(j)})$, we replace the inner minimization with it's value when the results of equation 4 hold. This gives:

$$\max_{\vec{\alpha}} \min_{\vec{w},b} \tfrac{1}{2}\|\vec{w}\|^2 + \sum_j \alpha_j (1-(\vec{w}^T\vec{x}^{(j)}+b)y^{(j)}) = \max_{\vec{\alpha}} \tfrac{1}{2}\|\vec{w}\|^2 + \sum_j \alpha_j (1-(\vec{w}^T\vec{x}^{(j)}+b)y^{(j)}), \text{ when (1) holds}$$

Now, we will simplify this term: $\tfrac{1}{2}\|\vec{w}\|^2 + \sum_j \alpha_j (1-(\vec{w}^T\vec{x}^{(j)}+b)y^{(j)}) = \tfrac{1}{2}\vec{w}^T\vec{w} + \sum_j \alpha_j \left(1-\left(\left(\sum_k \alpha_k y^{(k)}\vec{x}^{(k)}\right)^T \vec{x}^{(j)}+b\right)y^{(j)}\right)$

$$= \tfrac{1}{2}\sum_j\sum_k \alpha_j\alpha_k y^{(j)}y^{(k)} \vec{x}^{(j)}\cdot\vec{x}^{(k)} + \sum_j \alpha_j - \sum_j \alpha_j\left(\sum_k \alpha_k y^{(k)}\vec{x}^{(k)}\right)^T\cdot\vec{x}^{(j)} y^{(j)} + \sum_j \alpha_j b y^{(j)} \quad (\text{since } \vec{w} = \sum_j \alpha_j y^{(j)} \vec{x}^{(j)})$$

$$= \tfrac{1}{2}\sum_j\sum_k \alpha_j\alpha_k y^{(j)}y^{(k)} \vec{x}^{(j)}\cdot\vec{x}^{(k)} + \sum_j \alpha_j - \sum_j\sum_k \alpha_j\alpha_k y^{(j)}y^{(k)}\vec{x}^{(j)}\cdot\vec{x}^{(k)} + b\,\overset{0}{\cancel{\sum_j \alpha_j y^{(j)}}} \quad (\text{since } \sum_j \alpha_j y^{(j)} = 0$$

$$= \sum_j \alpha_j - \tfrac{1}{2}\sum_j\sum_k \alpha_j\alpha_k y^{(j)}y^{(k)} \vec{x}^{(j)}\cdot\vec{x}^{(k)}$$ So, ou

This proves that equation 2 can be simplified as:

$$\boxed{\max_{\vec{\alpha}} \sum_j \alpha_j - \tfrac{1}{2}\sum_j\sum_k \alpha_j\alpha_k y^{(j)}y^{(k)}\vec{x}^{(j)}\cdot\vec{x}^{(k)}}$$

6) From the figure in part 1, it is clear that all points are support vectors except $\vec{x}^{(2)}$. Since $x^{(2)}$ is not a support vector, the constraint can be removed.

So, we can instead define Lagrangian L_1 as below:

$$L_1(\vec{w},b,\vec{\alpha}) = \frac{1}{2}\|\vec{w}\|^2 + \sum_{j=1,j\neq 2}^{4} \alpha_j\left(1-(\vec{w}\cdot\vec{x}^{(j)}+b)y^{(j)}\right)$$

and, since removing the constraint has no effect

$$\max_{\vec{\alpha}} \min_{\vec{w},b} L = \max_{\vec{\alpha}} \min_{\vec{w},b} L_1 \qquad (*)$$

Now, equation $(*)$ above would definitely be true if $L=L_1$. This is obtained when $\boxed{\alpha_2=0.}$

Check:

$$L(\alpha_2=0) = \frac{1}{2}\|\vec{w}\|^2 + \sum_{j=1}^{4} \alpha_j\left(1-(\vec{w}\cdot\vec{x}^{(j)}+b)y^{(j)}\right)$$

$$= \frac{1}{2}\|\vec{w}\|^2 + \sum_{j=1,j\neq 2}^{4} \alpha_j\left(1-(\vec{w}\cdot\vec{x}^{(j)}+b)y^{(j)}\right)$$

$$= L_1 \checkmark$$

Note: Alternatively can argue that since $(1-(\vec{w}\cdot\vec{x}^{(2)}+b))y$ is greater than 0, we must have $\boxed{\alpha_2=0}$ to maximize L while satisfying $\alpha_j \geq 0$.

7) We have: $\sum_j \alpha_j y^{(j)} = 0$

$$\alpha_1 y^{(1)} + \alpha_2 y^{(2)} + \alpha_3 y^{(3)} + \alpha_4 y^{(4)} = 0$$

$$\alpha_1(1) + (0)(-1) + \alpha_3(-1) + \alpha_4(-1) = 0$$

$$\boxed{\alpha_1 = \alpha_3 + \alpha_4}$$

8) We wish to maximize the expression in part 5. To do this, we first expand, noting $\alpha_2=0$ and $\alpha_1=\alpha_3+\alpha_4$

$$\sum_j \alpha_j - \frac{1}{2}\sum_j\sum_k \alpha_j\alpha_k y^{(j)}y^{(k)}\vec{x}^{(j)}\cdot\vec{x}^{(k)} = \alpha_1+\alpha_3+\alpha_4 - \frac{1}{2}\Big(\alpha_1\alpha_1 y^{(1)}y^{(1)}\|x^{(1)}\|^2 + \alpha_1\alpha_3 y^{(1)}y^{(3)}\langle x^{(1)},x^{(3)}\rangle$$

$$+ \alpha_1\alpha_4 y^{(1)}y^{(4)}\langle x^{(1)},x^{(4)}\rangle + \alpha_3\alpha_1 y^{(3)}y^{(1)}\langle x^{(3)},x^{(1)}\rangle + \alpha_3\alpha_3 y^{(3)}y^{(3)}\langle x^{(3)},x^{(3)}\rangle$$

$$+ \alpha_3\alpha_4 y^{(3)}y^{(4)}\langle x^{(3)},x^{(4)}\rangle + \alpha_4\alpha_1 y^{(4)}y^{(1)}\langle x^{(4)},x^{(1)}\rangle + \alpha_4\alpha_3 y^{(4)}y^{(3)}\langle x^{(4)},x^{(3)}\rangle$$

$$+ \alpha_4\alpha_4 y^{(4)}y^{(4)}\langle x^{(4)},x^{(4)}\rangle\Big)$$

$$= 2\alpha_1 - \frac{1}{2}\left(\alpha_1\alpha_1\|x^{(1)}\|^2 + 2\alpha_1\alpha_3(-1)\begin{bmatrix}1\\1\end{bmatrix}\cdot\begin{bmatrix}1\\0\end{bmatrix} + 2\alpha_1\alpha_4(-1)\begin{bmatrix}1\\1\end{bmatrix}\cdot\begin{bmatrix}0\\1\end{bmatrix} + \alpha_3^2\|x^{(3)}\|^2 + 2\alpha_3\alpha_4(1)\begin{bmatrix}1\\0\end{bmatrix}\cdot\begin{bmatrix}0\\1\end{bmatrix} + \alpha_4^2\|x^{(4)}\|^2\right)$$

$$= 2\alpha_1 - \frac{1}{2}\left(\alpha_1^2(2) - 2\alpha_1\alpha_3(1) - 2\alpha_1\alpha_4(1) + \alpha_3^2(1) + \cancel{2\alpha_3\alpha_4(0)} + \alpha_4^2(1)\right)$$

$$= 2\alpha_1 - \frac{1}{2}\left(2\alpha_1^2 - 2\alpha_1\alpha_3 - 2\alpha_1\alpha_4 + \alpha_3^2 + \alpha_4^2\right)$$

$$= 2\alpha_1 - \frac{1}{2}\left(2\alpha_1^2 - 2\alpha_1(\alpha_3+\alpha_4) + \alpha_3^2+\alpha_4^2\right)$$

$$= 2\alpha_1 - \frac{1}{2}\left(\alpha_3^2+\alpha_4^2\right)$$

$\alpha_1 = \alpha_3+\alpha_4$: $\rightarrow$

$$= 2\alpha_3 + 2\alpha_4 - \frac{1}{2}\alpha_3^2 - \frac{1}{2}\alpha_4^2$$

Now, call this function $f(\alpha_3,\alpha_4)$. The maximum occurs at a critical point, where partial derivatives are both 0. This gives:

$$\left.\begin{aligned}\frac{\partial f}{\partial \alpha_3} = 0 &\rightarrow 2-\alpha_3 = 0 \rightarrow \alpha_3 = 2\\ \frac{\partial f}{\partial \alpha_4} = 0 &\rightarrow 2-\alpha_4 = 0 \rightarrow \alpha_4 = 2\end{aligned}\right\} \alpha_1 = \alpha_3+\alpha_4 = 2+2 = 4$$

So, $\boxed{\alpha_1=4}$, $\boxed{\alpha_3=2}$, $\boxed{\alpha_4=2}$, $\boxed{\alpha_2=0}$

Scanned with CamScanner

9) We have, from part 4, $\vec{w} = \sum_j \alpha_j y^{(j)} \vec{x}^{(j)}$. Substituting known values gives

$$\vec{w} = \sum_{j=1}^{4} \alpha_j y^{(j)} \vec{x}^{(j)}$$

$$= \alpha_1 y^{(1)} \vec{x}^{(1)} + \alpha_2 y^{(2)} \vec{x}^{(2)} + \alpha_3 y^{(3)} \vec{x}^{(3)} + \alpha_4 y^{(4)} \vec{x}^{(4)}$$

$$= (4)(1)\begin{bmatrix}1\\1\end{bmatrix} + (0)(-1)\begin{bmatrix}-1\\-1\end{bmatrix} + (2)(-1)\begin{bmatrix}1\\0\end{bmatrix} + (2)(-1)\begin{bmatrix}0\\1\end{bmatrix}$$

$$= \begin{bmatrix}4\\4\end{bmatrix} + \begin{bmatrix}-2\\0\end{bmatrix} + \begin{bmatrix}0\\-2\end{bmatrix}$$

$$\vec{w} = \begin{bmatrix}2\\2\end{bmatrix}$$

$$\boxed{\therefore \ \vec{w} = \begin{bmatrix}2\\2\end{bmatrix}}$$

10) The support vectors are points where our inequality; $(\vec{w}^T \vec{x}^{(i)} + b) y^{(i)} \geq 1$ becomes an equality, that is, at the support vectors, $(\vec{w}^T \vec{x}^{(i)} + b) y^{(i)} = 1$. Substituting $j = 1$:

$$(\vec{w}^T \vec{x}^{(1)} + b)(1) = 1$$

$$\begin{bmatrix}2 & 2\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} + b = 1$$

$$2 + 2 + b = 1$$

$$\boxed{b = -3}$$

11) Yes it is $w^T x + b$, $w = \begin{bmatrix}2\\2\end{bmatrix}$, $b = -3$ is the exact same as part 1.

Problem 2)

1) See Python Code

2) The accuracy here refers to the quotient # correct_preds / # all_preds. This can be readily computed using the LogisticRegression.score method in Python. Running this on both train and test sets gives:

The score on the test set is 1.0

The score on the test set is 0.9875 } See code

3) See Python code

4) See Jupyter notebook file submission for circled support vectors

To identify, first note that we can rewrite $\min_{\vec{w},b} \frac{1}{2}\|\vec{w}\|^2$ st $\forall i\ (\vec{w}\cdot\vec{x}^{(i)}+b)y^{(i)} \geq 1$ as:

$\min_{\vec{w},b} \max_{\vec{\alpha}} L(\vec{w},b,\vec{\alpha})$.

Now, from duality (it is given that Slater's conditions hold for SVM)

$$\min_{\vec{w},b} \max_{\vec{\alpha}} L(\vec{w},b,\vec{\alpha}) = \max_{\vec{\alpha}} \min_{\vec{w},b} L(\vec{w},b,\vec{\alpha})$$

Now from KKT conditions; specifically dual feasibility condition, $\alpha_i \geq 0\ \forall i \in 1,\dots N$ where N is the number of training samples. Note, $L(\vec{w},b,\vec{\alpha}) = \frac{1}{2}\|\vec{w}\|^2 + \sum \alpha_i (1-(\vec{w}\cdot\vec{x}^{(i)}+b)y^{(i)})$

For $1-(\vec{w}\cdot\vec{x}^{(i)}+b)y^{(i)} < 0$, we must set $\alpha_i = 0$ to maximize the innermost optimization ($\min_{\vec{w},b} L(\vec{w},b,\vec{\alpha})$)

So $(\vec{w}\cdot\vec{x}^{(i)}+b)y^{(i)} > 1 \Rightarrow \alpha_{(i)} = 0$. (1)

But if $(\vec{w}\cdot\vec{x}^{(i)}+b)y^{(i)} \leq 1$, then α_i can be positive, because this would maximize our function, as α_i and $(1-(\vec{w}\cdot\vec{x}^{(i)}+b)y^{(i)})$ are both positive. In our case we are implementing a hard margin SVM, so this case reduces to $(\vec{w}\cdot\vec{x}^{(i)}+b)y^{(i)} = 1$. Thus we have

$$(\vec{w}\cdot\vec{x}^{(i)}+b)y^{(i)} = 1 \Rightarrow \alpha_{(i)} \text{ can be non zero.} \quad (2)$$

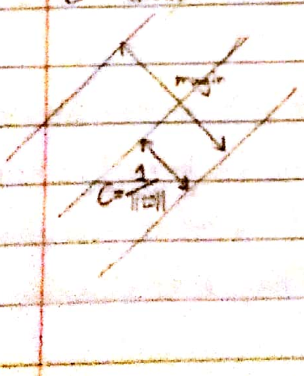
So, from (1) and (2) it is clear that the only points that influence the SVM optimization problem, i.e., the support vectors, are points $\vec{x}^{(i)}$ such that $\boxed{\vec{w}\cdot\vec{x}^{(i)}+b = 1}$

5) Running score_train = regressor.score(X_train, y_train) and score_test = regressor.score(X_test, y_test), we get:

Score on train set is 1.0 } See code

Score on test set is 1.0

6) In the derivation of our optimization equation $\min \|\vec{w}\|^2$ st $(\vec{w}\cdot\vec{x}^{(i)}+b)y^{(i)} \geq 1\ \forall i$, we assumed that $c = 1/\|\vec{w}\|$, where c is distance between decision boundary and margin. Since the 2 margins are equidistant from the decision boundary:



$$\text{margin} = 2c \quad \text{(See code for computation)}$$

$$= 2/\|\vec{w}\|$$

$$\approx 0.92930075913967574$$

7) w is orthogonal to the decision boundary.

Let $x^{(1)}, x^{(2)}$ be two points on the decision boundary

$$w^T x^{(1)} + b = 0 \quad (1)$$

$$w^T x^{(2)} + b = 0 \quad (2)$$

$(1)-(2)$: $\quad w^T(x^{(1)} - x^{(2)}) = 0 \Rightarrow \langle w, x^{(1)} - x^{(2)} \rangle = 0$

Since $x^{(1)}, x^{(2)}$ chosen arbitrarily, and $x^{(1)} - x^{(2)}$ is orthogonal to w, w is orthogonal to the decision boundary line.

8) The decision boundary changes. However, the test accuracy is still 100%. This is easily justifiable.

As we are including more training points, the support vectors can change. If we add a new point closer to decision boundary than current closest, then it will replace the previous point as a support vector, and the decision boundary will change.

The test accuracy being 100% also makes sense. Since we are including more points, the SVM model is more likely to generalize better. Note that this is not a general result. If there exists a unique support vector on each side of decision boundary for entire set of points, and we consider these in first case but not second, we could end up misclassifying these in the bigger train set.

Even if our first train set is subset of second, adding more points that change decision boundary can result in misclassification in general. From viewing our data set, however, it is clear that the two classes are distinct enough (spatially) and points of same class are close enough such that adding more points to SVM with test_size = 0.8 won't radically change SVM decision boundary.

9) No, if they did, binary linear would have 100% accuracy on test set.

This is further shown by comparing values of $\vec{w}$ and b in the hyperplane formula for both implementations.

They are different!

$$\text{Binary: } \vec{w} \approx \begin{bmatrix} 7.58 \\ -7.46 \end{bmatrix} \qquad b \approx -6.42$$

$$\text{linear SVM: } \vec{w} \approx \begin{bmatrix} 3.33 \\ -3.33 \end{bmatrix} \qquad b \approx -7.66$$

$\Rightarrow$ Not same!

10) Since the data points are not linearly separable, we can use a non-linear kernel. In video 14 of the course, we covered Gaussian kernels, which classifies points based on some notion of similarity to other points in that class, specifically, $f_i = \text{sim}(\vec{x}, \vec{l}^{(i)}) = \exp\left(-\frac{\|\vec{x} - \vec{l}^{(i)}\|^2}{2\sigma^2}\right)$. This is useful here because points in each class seem to cluster in different areas.

In sklearn we can implement this using the radial basis kernel function (RBF). The RBF kernel on 2 samples x and x', represented as feature vectors in some input space, is defined as

$$K(x, x') = e^{-\frac{\|x - x'\|^2}{2\sigma^2}}$$

, and is suggested as the go-to non-linear SVM classification method.

When implementing using sklearn, we set test size to 0.9 (instead of 0.8) so we can generalize better, without overfitting. Further, we play with hyperparameter gamma (sigma) and C (controls slack) so that we yield a high test set accuracy. Note that large σ leads to underfitting, low σ leads to overfitting. A test set accuracy of $\approx 72\%$ is reached using RBF kernel, $C = 2$ and $\gamma = \frac{1}{2\sigma^2} = 1$.

(See code for plot)

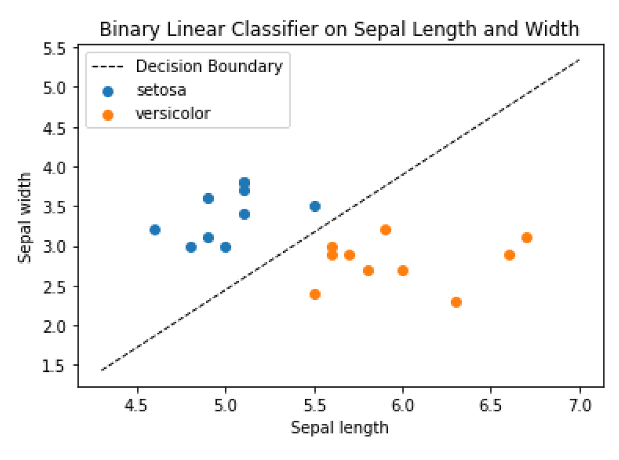
Problem 1

(2 points) Implement a binary linear classifier on the first two dimensions (sepal lengthand width) of the iris dataset and plot its decision boundary. (Hint: sklearn refers tothe binary linear classifier as a LogisticRegression, we will see why later in the course.)

In [13]:
```python
from sklearn import datasets
from sklearn.model_selection import train_test_split

iris = datasets.load_iris()
iris_inputs = iris.data[:100, :2]
iris_targets = iris.target[:100]
X_train, X_test, y_train, y_test = train_test_split(iris_inputs, iris_targets, test_size=0.8, random_state=0)
```

In [14]:
```python
from sklearn.linear_model import LogisticRegression
logisticRegr = LogisticRegression()
logisticRegr.fit(X_train, y_train)

import matplotlib.pyplot as plt
import numpy as np

n_classes = 2
for i in range(n_classes):
    index = np.where(y_train == i)
    plt.scatter(X_train[index, 0], X_train[index, 1],
    label=iris.target_names[i])

plt.xlabel('Sepal length')
plt.ylabel('Sepal width')

b = logisticRegr.intercept_[0]
w1, w2 = logisticRegr.coef_.T

c = -b/w2
m = -w1/w2

xmin, xmax = iris_inputs[:,0].min(), iris_inputs[:,0].max()
ymin, ymax = iris_inputs[:,1].min(), iris_inputs[:,1].max()
xd = np.array([xmin, xmax])
yd = m*xd + c
plt.plot(xd, yd, 'k', lw=1, ls='--', label = "Decision Boundary")
plt.title("Binary Linear Classifier on Sepal Length and Width")
plt.legend()
```

Out[14]:
```
<matplotlib.legend.Legend at 0x2aa92ca6ac0>
```



In [15]:
```python
print(w1)
print(w2)
print(b)
```

```
[1.80226162]
[-1.24492959]
-5.968527502942981
```

Problem 2

(1 point) Report the accuracy of your binary linear classifier on both the training andtest sets.

In [16]:
```python
score_train = logisticRegr.score(X_train, y_train)
print('The score on the train set is {}'.format(score_train))
score_test = logisticRegr.score(X_test, y_test)
print('The score on the test set is {}'.format(score_test))
```

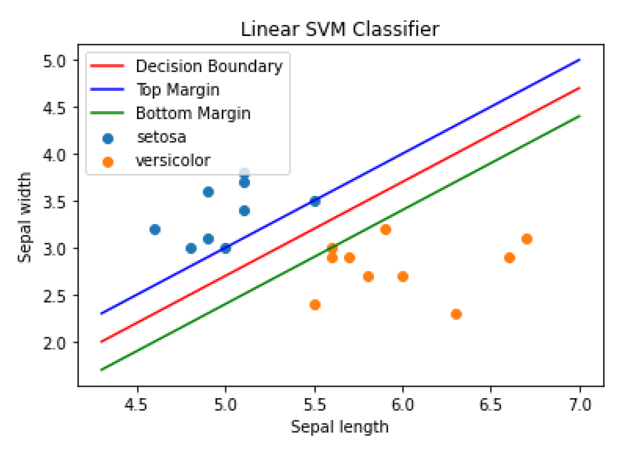
```
The score on the train set is 1.0
The score on the test set is 0.9875
```

Problem 3

(2 points) Implement a linear SVM classifier on the first two dimensions (sepal lengthand width). Plot the decision boundary of the classifier and its margins.

In [17]:
```python
from sklearn.svm import SVC
X_train, X_test, y_train, y_test = train_test_split(iris_inputs, iris_targets, test_size=0.8, random_state=0)
regressor = SVC(kernel = 'linear', C=10e9)
regressor.fit(X_train, y_train)
w = regressor.coef_[0]
b = regressor.intercept_[0]

n_classes = 2
for i in range(n_classes):
    index = np.where(y_train == i)
    plt.scatter(X_train[index, 0], X_train[index, 1],
    label=iris.target_names[i])
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')

c = -b/w2
m = -w1/w2

xmin, xmax = iris_inputs[:,0].min(), iris_inputs[:,0].max()
ymin, ymax = iris_inputs[:,1].min(), iris_inputs[:,1].max()
x_points = np.array([xmin, xmax])
y_points = -(w[0] / w[1]) * x_points - b / w[1]

d = np.sqrt(np.sum(regressor.coef_ ** 2))
y_down = y_points - 1/w[1]
y_up = y_points + 1/w[1]

# Plotting a red hyperplane
plt.plot(x_points, y_points, c='r', label="Decision Boundary");
plt.plot(x_points, y_down, c='b', label="Top Margin");
plt.plot(x_points, y_up, c='g', label="Bottom Margin");

# Encircle support vectors
# plt.scatter(regressor.support_vectors_[:, 0],
#             regressor.support_vectors_[:, 1],
#             s=100, linewidth=1, facecolors='none', edgecolors='black');
plt.title("Linear SVM Classifier")
plt.legend()
```

Out[17]:
```
<matplotlib.legend.Legend at 0x2aa92d37b20>
```

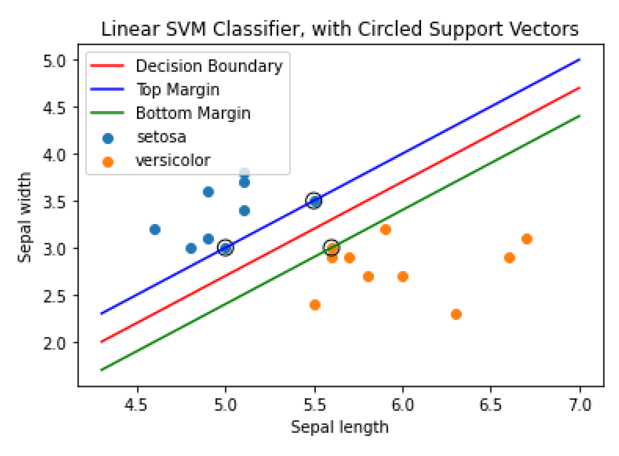


Problem 4

(1 point) Circle the support vectors. Please justify how to identify them through the duality theorem. (hint: KKT condition)

In [18]:
```python
from sklearn.svm import SVC
X_train, X_test, y_train, y_test = train_test_split(iris_inputs, iris_targets, test_size=0.8, random_state=0)
regressor = SVC(kernel = 'linear', C=10e9)
regressor.fit(X_train, y_train)
w = regressor.coef_[0]
b = regressor.intercept_[0]

n_classes = 2
for i in range(n_classes):
    index = np.where(y_train == i)
    plt.scatter(X_train[index, 0], X_train[index, 1],
    label=iris.target_names[i])
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')

c = -b/w2
m = -w1/w2

xmin, xmax = iris_inputs[:,0].min(), iris_inputs[:,0].max()
ymin, ymax = iris_inputs[:,1].min(), iris_inputs[:,1].max()
x_points = np.array([xmin, xmax])
y_points = -(w[0] / w[1]) * x_points - b / w[1]

d = np.sqrt(np.sum(regressor.coef_ ** 2))
y_down = y_points - 1/w[1]
y_up = y_points + 1/w[1]

plt.plot(x_points, y_points, c='r', label="Decision Boundary");
plt.plot(x_points, y_down, c='b', label="Top Margin");
plt.plot(x_points, y_up, c='g', label="Bottom Margin");

plt.scatter(regressor.support_vectors_[:, 0],
            regressor.support_vectors_[:, 1],
            s=100, linewidth=1, facecolors='none', edgecolors='black');
plt.title("Linear SVM Classifier, with Circled Support Vectors")
plt.legend()
```

Out[18]:
```
<matplotlib.legend.Legend at 0x2aa92d03070>
```



Problem 5

(1 point) Report the accuracy of your linear SVM classifier on both the training andtest sets

In [19]:
```python
score_train = regressor.score(X_train, y_train)
print('The score on the train set is {}'.format(score_train))
score_test = regressor.score(X_test, y_test)
print('The score on the test set is {}'.format(score_test))
```

```
The score on the train set is 1.0
The score on the test set is 1.0
```

In [20]:
```python
print(w)
print(b)
```

```
[ 3.33266363 -3.33342658]
-7.662778452658107
```

Problem 6

(1 point) What is the value of the margin? Justify your answer.

In [21]:
```python
2/np.linalg.norm(w)
```

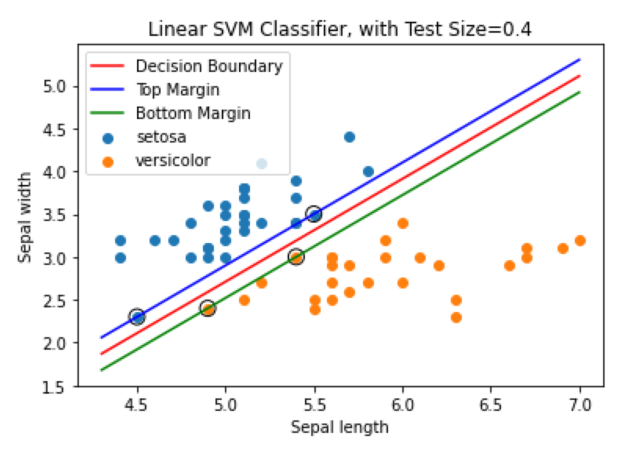
Out[21]:
```
0.42430075463962574
```

Problem 8

(3 points) Split the iris dataset again in a training and test set, this time setting testsizeto 0.4 when calling traintestsplit. Train the SVM classifier again. Does the decisionboundary change? How about the test accuracy? Please justify why (hint: think aboutthe support vectors), and illustrate your argument with a new plot.

In [22]:
```python
X_train, X_test, y_train, y_test = train_test_split(iris_inputs, iris_targets, test_size=0.4, random_state=0)
regressor = SVC(kernel = 'linear', C=10e9)
regressor.fit(X_train, y_train)
w = regressor.coef_[0]
b = regressor.intercept_[0]

n_classes = 2
for i in range(n_classes):
    index = np.where(y_train == i)
    plt.scatter(X_train[index, 0], X_train[index, 1],
    label=iris.target_names[i])
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')

c = -b/w2
m = -w1/w2

xmin, xmax = iris_inputs[:,0].min(), iris_inputs[:,0].max()
ymin, ymax = iris_inputs[:,1].min(), iris_inputs[:,1].max()
x_points = np.array([xmin, xmax])
y_points = -(w[0] / w[1]) * x_points - b / w[1]

d = np.sqrt(np.sum(regressor.coef_ ** 2))
y_down = y_points - 1/w[1]
y_up = y_points + 1/w[1]

plt.plot(x_points, y_points, c='r', label="Decision Boundary");
plt.plot(x_points, y_down, c='b', label="Top Margin");
plt.plot(x_points, y_up, c='g', label="Bottom Margin");

plt.scatter(regressor.support_vectors_[:, 0],
            regressor.support_vectors_[:, 1],
            s=100, linewidth=1, facecolors='none', edgecolors='black');
plt.title("Linear SVM Classifier, with Test Size=0.4")
plt.legend()
```

Out[22]:
```
<matplotlib.legend.Legend at 0x2aa92a9e250>
```



In [23]:
```python
score_train = regressor.score(X_train, y_train)
print('The score on the train set is {}'.format(score_train))
score_test = regressor.score(X_test, y_test)
print('The score on the test set is {}'.format(score_test))
```

```
The score on the train set is 1.0
The score on the test set is 1.0
```

In [24]:
```python
print(w)
print(b)
```

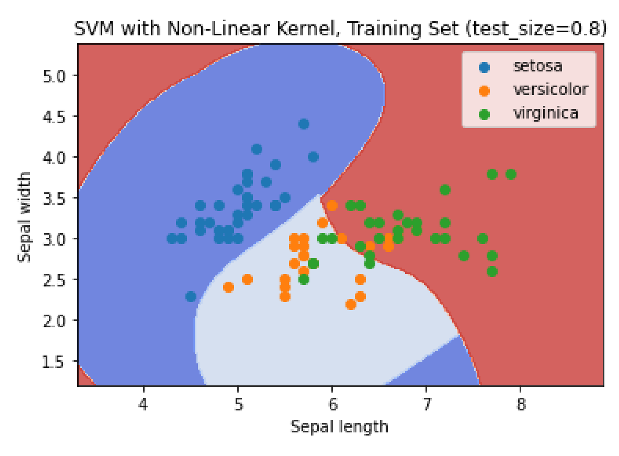
```
[ 6.31804679 -5.26503723]
-17.32197688065553
```

Problem 10

(3 points) Now consider all 150 entries in the iris dataset, and retrain the SVM. Youshould find that the data points are not linearly separable. How can you deal with it?Justify your answer and plot the decision boundary of your new proposed classifier.

In [25]:
```python
X_train, X_test, y_train, y_test = train_test_split(iris.data[:, :2], iris.target[:], test_size=0.4, random_state=0)
regressor = SVC(kernel = 'rbf', random_state=0, gamma=1, C=2)
regressor.fit(X_train, y_train)

X0, X1 = X_train[:, 0], X_train[:, 1]
x_min, x_max = X0.min() - 1, X0.max() + 1
y_min, y_max = X1.min() - 1, X1.max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02), np.arange(y_min, y_max, 0.02))
Z = regressor.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
n_classes = 3
for i in range(n_classes):
    index = np.where(y_train == i)
    plt.scatter(X_train[index, 0], X_train[index, 1],
    label=iris.target_names[i])

plt.title("SVM with Non-Linear Kernel, Training Set (test_size=0.8)")
plt.xlabel("Sepal length")
plt.ylabel("Sepal width")
plt.legend()
```

Out[25]:
```
<matplotlib.legend.Legend at 0x2aa92cf23a0>
```



In [26]:
```python
score_train = regressor.score(X_train, y_train)
print('The score on the train set is {}'.format(score_train))
score_test = regressor.score(X_test, y_test)
print('The score on the test set is {}'.format(score_test))
```

```
The score on the train set is 0.8444444444444444
The score on the test set is 0.7166666666666667
```