The final project focused on describing the process of finding the optimal parameters w and b for creating the decision boundary ($w^T x + b = 0$) - used for classification problems - as a Quadratic Unconstrained Binary Optimization problem (QUBO). Afterwards, Grover Adaptive Search with Phase Encoding algorithm is utilized in order to find the bit-string that minimizes the cost function as much as possible and fulfills other conditions in order to find w and b. The project serves more as a proof of concept that supervised learning for classification can be seen as a QUBO problem, rather than attempting to provide the most accurate decision boundary to classify the training dataset. The project used the two categories of wines (labelled as -1 and 1) with two characteristics per each wine datapoint, 'flavanoids' and 'malic acid', as an example for classification. The report will explain how the procedure for finding the optimal parameters for the decision boundary was mapped onto a QUBO optimization problem, the results of a very simplified case example, and finding the limitations of the project with potential improvements.

Given a logistic classification problem, with m training data points $x^{(i)}$ (in which each i'th $x^{(i)}$ is defined by a feature vector $\begin{bmatrix} x_1^i \\ x_2^i \\ x_3^i \\ \vdots \\ x_n^i \end{bmatrix}$) and associated output data points $y^{(i)}$, a classifier function $g(w^T x + b)$ is found that maps $x^{(i)}$ to $y^{(i)}$ through a process that determines the optimal weights per each feature that describes $x^{(i)}$, w = $\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix}$, and optimal b constant.

As a result, the decision boundary (defined by $w^T x + b = 0$) can be found, which classifies the training dataset into their correct labels -1 or 1 (which is used by $y^{(i)}$) and can be used to predict the labels of other input data with reasonable accuracy (as long as it is within the training dataset range). This is done by defining the data points that cause $w^T x + b > 0$ to be labelled as 1, $g(w^T x + b)$ = 1, and $w^T x + b < 0$ to be labelled as -1, $g(w^T x + b)$ = -1, . From this we can say that whenever $w^T x + b >> 0$, we can be very confident that the $g(w^T x + b)$ = 1 and vice versa.

Therefore, in order to quantify whether we have confidence in the prediction of $g(w^T x + b)$, we define the functional margin of (w, b) with respect to the training example:

$\hat{\gamma} = y^{(i)}(w^T x + b)$ and geometric margin of (w,b) to be $\gamma^{(i)} = \hat{\gamma}^{(i)}/||w||$ (geometric margin defines the shortest distance between the decision boundary and some data point ($x^{(i)}$, $y^{(i)}$)

Furthermore, given the dataset S = {($x^{(i)}$, $y^{(i)}$); i = 1....m}, we define the functional and geometric margin with respect to S to be

$\hat{\gamma}$ = min of $\hat{\gamma}^{(i)}$ and γ = min of $\gamma^{(i)}$.

Now given that we defined γ as the distance between some data point and the decision boundary, we can use the fact that larger γ means we have higher confidence of the

prediction $g(w^T x + b)$ (since larger γ means larger $\hat{\gamma}$ means larger $|w^T x + b| > 0$) to define the following optimization problem:

find $max_{\gamma,w,b}\ \gamma$, by varying w and b,
such that $y^{(i)}(w^T x + b) \geq \gamma$ for i = 1...m and ||w|| = 1.

Or in other words, we want to maximize γ so that each training example has an associated functional margin of at least γ. The optimization problem can be redefined as follows:

find $max_{\gamma,w,b}\ \hat{\gamma}/||w||$, by varying w and b,
such that $y^{(i)}(w^T x + b) \geq \hat{\gamma}$ for i = 1...m

This creates a non-convex objective function which is very difficult to optimize. To further simplify, we set $\hat{\gamma}$ = 1, which can be satisfied by rescaling w, b
As a result, maximizing $\hat{\gamma}/||w|| = 1/||w||$ is the same thing as minimizing $||w||$. As a result, we re-define the optimization problem to be

$min_{w,b}\frac{1}{2}||w||^2$
such that $y^{(i)}(w^T x + b) \geq 1$ for i = 1...m

This optimization problem has a quadratic objective objective and linear constraints. Now using Lagrange multipliers and the KKT dual complementary condition (as further explained in https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf), we can again redefine the optimization problem for finding the optimal w, b values in the following way

max $\mathcal{L}(w, b, \alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)}$,
where α_i is a lagrange multiplier associated with each data point ($x^{(i)}$, $y^{(i)}$)
such that the found lagrange multipliers satisfy the following conditions
- $\alpha_i \geq 0$ for i = 1, ..., m
- $\sum_{i=1}^{m} \alpha_i y_i = 0$

since the first summation term contains only α_i , to find the lagrange multipliers that cause the Lagrangian to maximize, we focus on finding the lagrange multipliers, α_i , that cause $\sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)}$ to minimize
given that $y^{(i)} y^{(j)} (x^{(i)})^T x^{(j)}$ are known, we can rewrite it as $q_{i,j}$

now the optimization problem turns into

$min_{\alpha} \sum_{i,j=1}^{m} \alpha_i \alpha_j q_{i,j}$
this problem is an NP-hard problem that takes the form of

$$x = \min_{x \in X} \sum_{i,j=0}^{2^n - 1} q_{ij} x_i x_j$$

As a result, we can say that our optimization problem is part of the QUBO problems, for which we can utilize Grover's Adaptive search with phase encoding algorithm
Since x_i, x_j take only binary values, we have to use binary encoding to express α_i, α_j in a

binary form up to a certain limit 'n'
$\alpha_i = \sum_{k=1}^{n} x_{i,k} 2^{k-1}$
$\alpha_j = \sum_{l=1}^{n} x_{j,l} 2^{l-1}$

the final form that our optimization problem take is

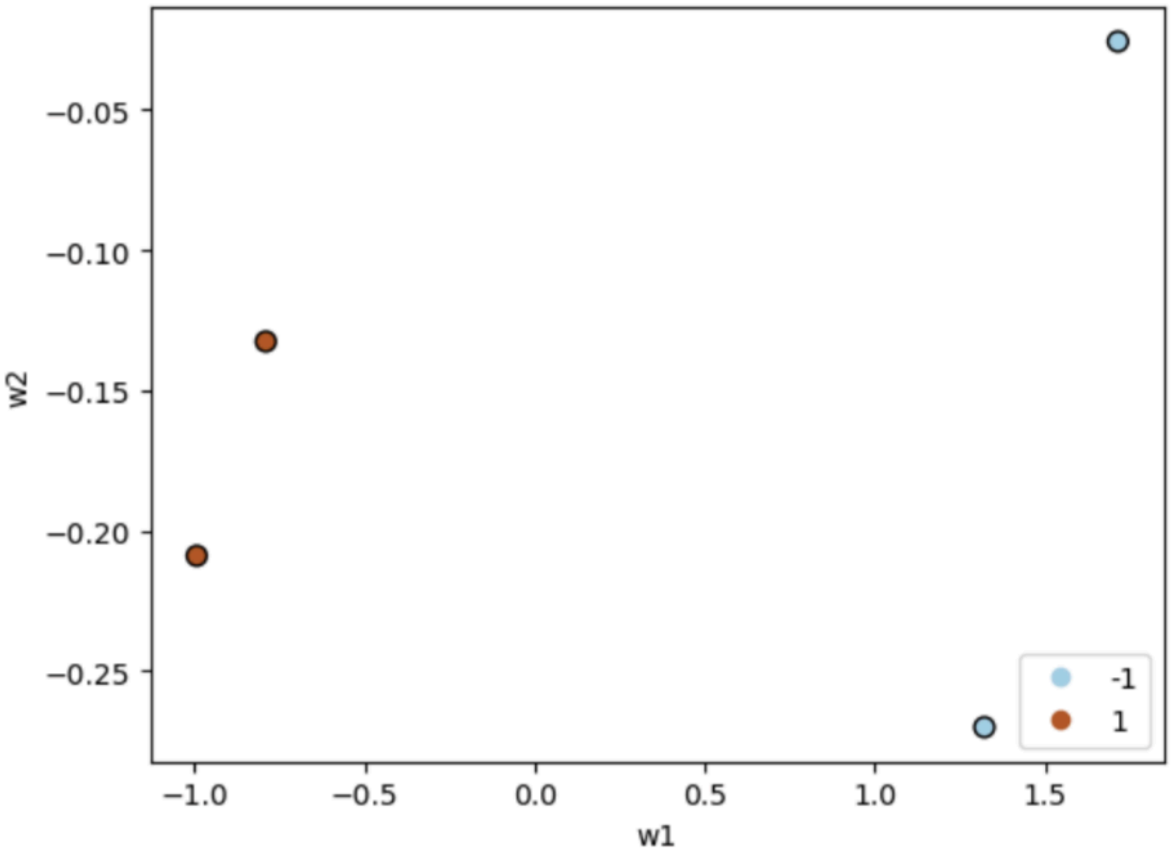
$min_x f(x)$
$f(x) = \sum_{i,j=1}^{m} (\sum_{k=1}^{n} x_{i,k} 2^{k-1}) * (\sum_{l=1}^{n} x_{j,l} 2^{l-1}) * q_{i,j}$

Example

the project used the wine dataset from sklearn and used the following dataset for training

	flavanoids	malic_acid	target
0	1.321944	-0.270148	-1
1	1.714484	-0.025601	-1
2	-0.992685	-0.209012	1
3	-0.789647	-0.132590	1

given that the project decided to use n = 3 to encode the lagrange multipliers into the binary form and there is a lagrange multiplier associated to each data point, we had to choose only 4 data points since 4*3 = 12 qubits were needed for the input space and 13 qubits were needed for output space (to store the maximum possible score of the input space + 1 sign bit for the oracle)
As a result, 25 qubits were needed to implement the Grover's Adaptive Search with Phase Encoding algorithm, and unfortunately, the transpile(qc, backend) didn't seem to work for circuits with more than 29 qubits, which would have happened if we have used more than 4 data points



our goal is to find the optimal w and b to construct $w^T x + b = 0$ and create the decision boundary that would classify our very limited dataset

given that m = 4, n = 3, we were able to use
$f(x) = \sum_{i,j=1}^{m}(\sum_{k=1}^{n} x_{i,k}2^{k-1}) * (\sum_{l=1}^{n} x_{j,l}2^{l-1}) * q_{i,j}$
to construct f(x) that would be used to plug in x values that and see whether f(x) is minimized

Now we use the Grover's Adaptive Search with Phase Encoding algorithm from lesson 10 and get many results, some of which are:

```
y_init= -764
x =  010010001011 0000001001000
f(x) = 219.57387865924778
0
r= 2
y_init= -764
x =  111010011011 0000011100000
f(x) = 546.4754837280741
-3
r= 3
y_init= -764
x =  010110110111 0000100001010
f(x) = 266.33934231988127
5
r= 1
y_init= -763
x =  100000010010 0000000011101
f(x) = 24.049554652731366
0
r= 1
```

for each y*init, the algorithm takes about 3-3.5 minutes to output the x, f(x), $\sum*{i = 1} ^ m \alpha_i y_i$, and r (number of iterations of the search algorithm)

The project would have taken too much time to consider all y*init from -778 to -1. As a result, we calculated x and f(x) from -778 to -708. From the results, we will choose x = 100000010010 for which f(x) is very low relative to all other f(x) that have been calculated and outputs* α_i *values that satisfies $\sum*{i = 1} ^ m \alpha_i y_i = 0$ and* $\alpha_i \geq 0$
the x output gives us the following α

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 2 \\ 2 \end{bmatrix}$$

now we can use the following expressions (as shown in https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf) to find the optimal w and b

$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$

```
W = np.zeros(2)
alpha = [4, 0, 2, 2]
for i in range(len(w)):
    #print(v(i))
    #print(np.array(v(i)[0]))
    W += np.array(v(i)[0]) * y(i) * alpha[i]
W
```

```
array([-8.85244 ,  0.397388])
```

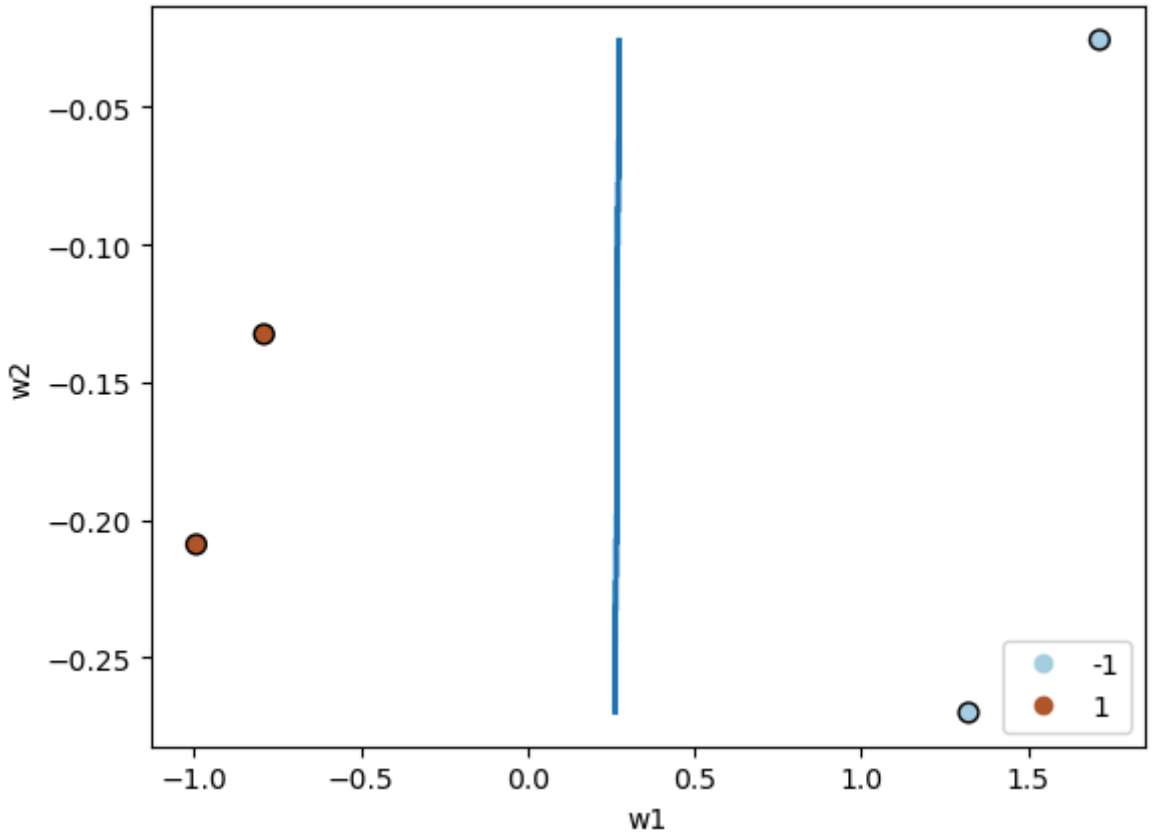
$$b = -\frac{(max_{i:y^{(i)}=-1} w^T x^{(i)}) + (min_{i:y^{(i)}=1} w^T x^{(i)})}{2}$$

```
maxx = max([W.T @ np.array(v(i)[0]) for i in range(len(w)) if y(i) == -1])
minn = min([W.T @ np.array(v(i)[0]) for i in range(len(w)) if y(i) == 1])

#print(maxx, minn)
b = -(maxx + minn)/2
print(b)
```

```
2.4360852515120004
```

given the parameters, the following decision boundary is now constructed



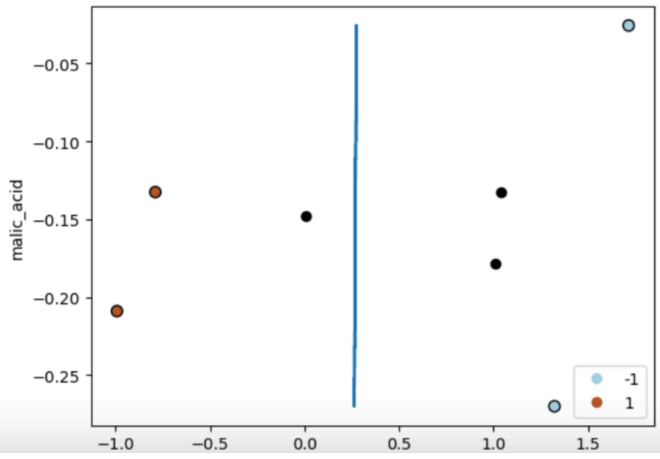
the decision boundary can be tested against new data points that are within the training range

In [146]:
```
testw.iloc[1:4]
```

Out[146]:

	flavanoids	malic_acid	target	predicted
1	1.010620	-0.178443	-1	-1
2	1.037691	-0.132590	-1	-1
3	0.008967	-0.147875	-1	1

In [143]:
```
plt.figure(1)
plot = plt.scatter(w['flavanoids'], w['malic_acid'], c=w['target'], edgecolor='k', s=50
plt.xlabel('flavanoids')
plt.ylabel('malic_acid')
plt.plot(w1, w2)
plt.scatter(testw1[1:4], testw2[1:4], color = 'black')
plt.legend(handles=plot.legend_elements()[0], labels=['-1', '1'], loc='lower right')
plt.show()
```

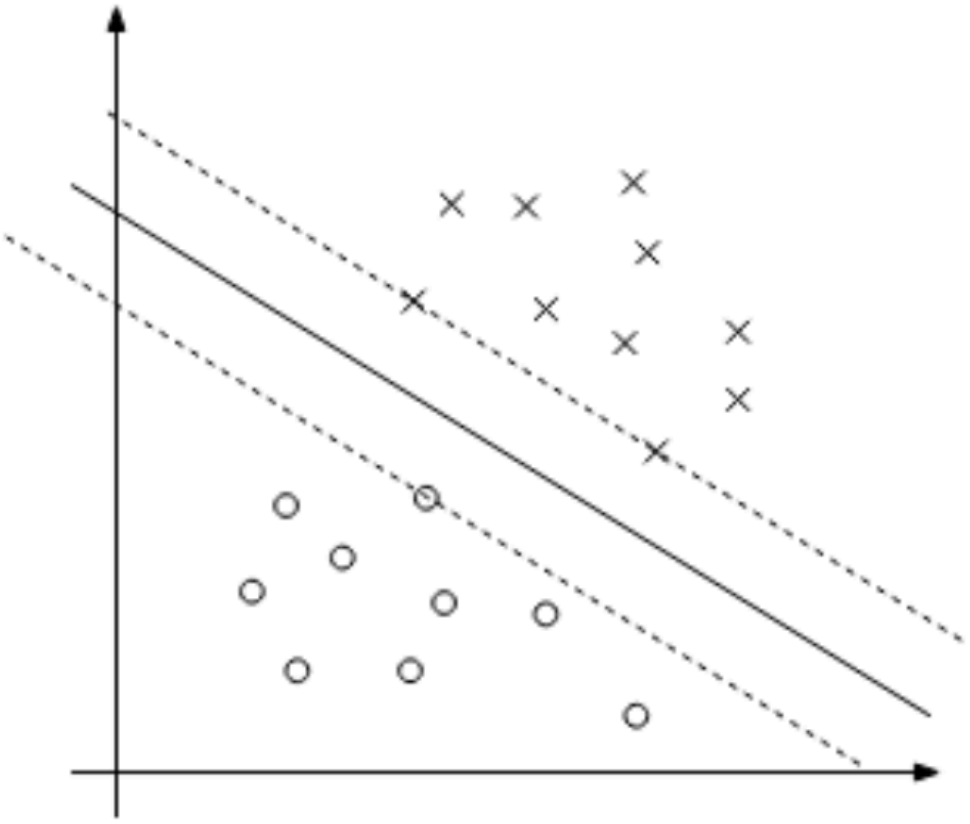


- the performance is quite poor (2/3 success) due to the following limitations of the implementation

limitations:

- small amount of training data
- Algorithm with lagrange multipliers works best if we use data points with functional margins = 1 during training
  - The algorithm with lagrange multipliers relies on the fact that at the at the optimal solution of the optimization problem, the lagrange multipliers associated with data points with functional margins more than 1 equates to 0.
  - As a result, the algorithm wants to only use the lagrange multipliers associated with the data points whose functional margins = 1, since it uses these data points as support vectors to create the best decision boundary for data points that are rather close to it (data points that are not as guaranteed to have a label as data points far from the decision boundary)
- Even if we provide a large amount of training data and only utilize data points with $\hat{\gamma} = 1$, the the process works only for linearly separable datasets. We would need to reformulate the optimization problem again and make use of other algorithms and Kernels to create Support Vector Machines that could learn from high-dimensional feature spaces, which may or may not work with QUBO

improvements
- demonstrate with more data points that by taking data points with functional margins of 1, which will act as support vectors, we can create a very reliable decision boundary by only considering those few data points. This is due to the fact that at the optimal solution of the optimization problem (according to https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf), alphas corresponding with data points with functional margins = 1 will be non-zero. As a result, not all training data points are needed to create a decision boundary (unlike traditional logistic classification scheme with gradient descent), only ones with $\hat{\gamma} = 1$
-
-



- handling floating alpha values
- introduce more training data points
- figure out how to represent a changed optimization problem with kernels for high-dimensional feature spaces as a QUBO problem