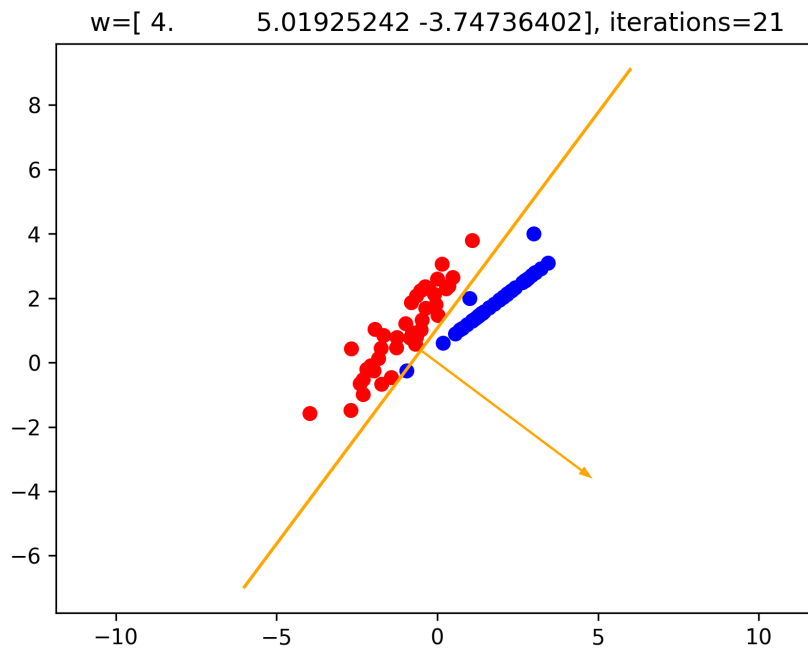


## Question 3

a



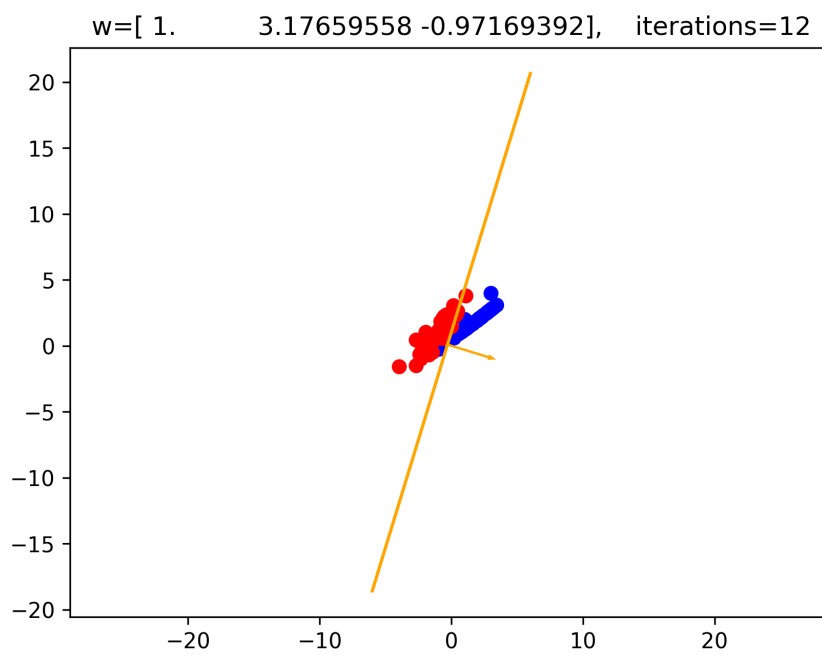
```
39 def perceptron(X, y, max_iter=100):
40     np.random.seed(1)
41
42     # Initialise w vectors
43     nfeatures = X.shape[1]
44     w = np.zeros((max_iter, nfeatures))
45     w[0] = np.zeros(nfeatures)
46
47     # Iterate and adjust w
48     for t in range(max_iter):
49
50         yXw = y * (X @ w[t].T)
51         mistake_idx = np.where(yXw <= 0)[0]
52
53         # If there are mistakes, choose a random one and
54         # update accordingly
55         if mistake_idx.size > 0:
56             i = np.random.choice(mistake_idx)
57             w = w + y[i] * X[i]
58             w[i + 1] = w[i] + y[i] * X[i]
59
60         else:
61             return w[t], t + 1
62
63     # Max iterations reached, return the latest w vector
64     return w[max_iter - 1], max_iter
```

```

66 def q3a():
67     # import data
68     X = import_data(Q3_X_DIR)
69     y = import_data(Q3_Y_DIR)
70
71     # create perceptron
72     w, nmb_iter = perceptron(X,y)
73
74     # Plot
75     fig, ax = plt.subplots()
76     plot_perceptron(ax, X, y, w)
77     ax.set_title(f"w={w}, iterations={nmb_iter}")
78     plt.savefig("outputs/Q3a.png", dpi=300)

```

**b**



```

80 def dual_perceptron(X, y, max_iter=100):
81     np.random.seed(1)
82
83     # initialize alphas
84     size = X.shape[0]
85     alpha = np.zeros((max_iter, size))
86     alpha[0] = np.zeros(size)
87
88     for t in range(max_iter):
89         # Assume no miss-classifications
90         mistakes = np.zeros(size)
91
92         # Calculate mistakes
93         for i in range(X.shape[0]):
94             sum_of_instances = np.sum(y * alpha * (X @ X[i]))
95             mistakes[i] = y[i] * sum_of_instances
96
97         # Find the indices of mistakes
98         mistake_idx = np.where(mistakes <= 0)[0]
99         if mistake_idx.size > 0:
100             choice = np.random.choice(mistake_idx)
101             alpha[t, choice] = alpha[t, choice] + 1
102             alpha[t+1] = alpha[t]
103         else:
104             return alpha[t], t + 1
105
106     # Max iterations reached, return the latest alpha vector
107     return alpha[max_iter-1], max_iter

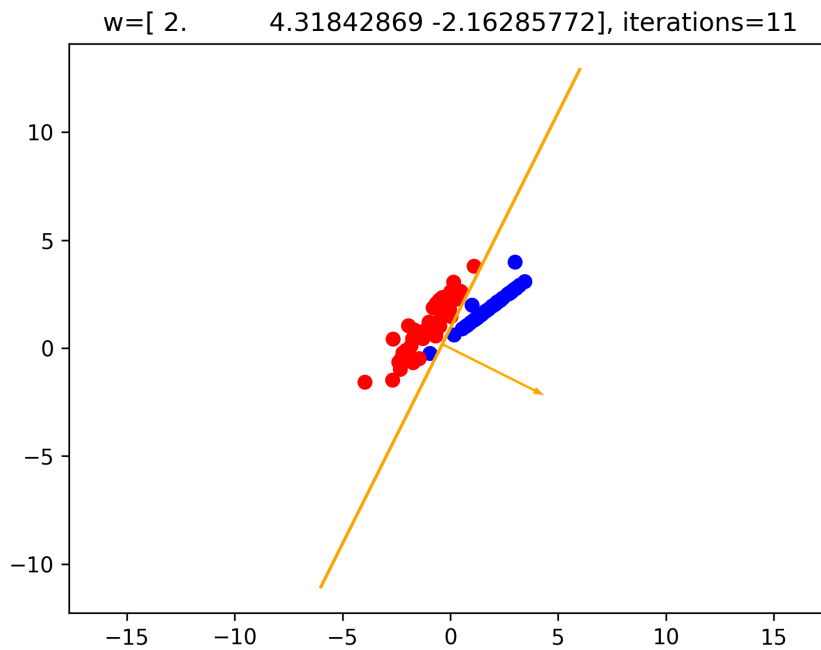
```

```

109 def q3b():
110     # Import data
111     X = import_data(Q3_X_DIR)
112     y = import_data(Q3_Y_DIR)
113
114     alpha, nmb_iter = dual_perceptron(X,y)
115
116     # Use alphas to yield w
117     w = (alpha * y) @ X
118
119     fig, ax = plt.subplots()
120     plot_perceptron(ax, X, y, w)
121     ax.set_title(f"w={w}, iterations={nmb_iter}")
122     plt.savefig("./outputs/Q3b.png", dpi=300)

```

c



```

124 def rPerceptron(X, y, max_iter=100):
125     np.random.seed(1)
126
127     # Initialise w vectors
128     nfeatures = X.shape[1]
129     w = np.zeros((max_iter, nfeatures))
130     w[0] = np.zeros(nfeatures)
131
132     # Initialise indicator
133     indicator = np.zeros(X.shape[0])
134
135     # Set r equal to 2 as in question
136     r = 2
137
138     for t in range(max_iter):
139
140         yXw = (y * (X @ w[t].T)) + (indicator * r)
141         mistake_idx = np.where(yXw <= 0)[0]
142
143         # If there are mistakes, update w vector at index "i"
144         if mistake_idx.size > 0:
145             i = np.random.choice(mistake_idx)
146             w = w + y[i] * X[i]
147             w[i+1] = w[i] + y[i]*X[i]
148             indicator[i] = 1
149         else:
150             return w[t], t + 1
151
152     # Max iterations reached, return the latest w vector
153     return w[max_iter - 1], max_iter

```

```

155 def q3c():
156     # import data
157     X = import_data(Q3_X_DIR)
158     y = import_data(Q3_Y_DIR)
159
160     # create perceptron
161     w, nmb_iter = rPerceptron(X,y)
162
163     # Plot
164     fig, ax = plt.subplots()
165     plot_perceptron(ax, X, y, w)
166     ax.set_title(f"w={w}, iterations={nmb_iter}")
167     plt.savefig("outputs/Q3c.png", dpi=300)

```