Exercise 1

(a) Using Theorem 3.6: With $|\mathcal{H}| = 3^3 = 27$

$$\Pr_{T\ \mathcal{D}^m} (\forall h \in \mathcal{H} : |err_T(h) - err_D(h)| \leq \epsilon) > 1 - \delta$$

$$\Pr_{T\ \mathcal{D}^m} (\forall h \in \mathcal{H} : |err_T(h) - err_D(h)| \leq \epsilon) > 0.9$$

$$\Rightarrow \delta = 0.1$$

$$m \geq \frac{1}{2\epsilon^2} \log\left(\frac{2|\mathcal{H}|}{\delta}\right)$$

$$143 \geq \frac{1}{2\epsilon^2} \log\left(\frac{2 \cdot 3^3}{0.1}\right)$$

$$143 \geq \frac{1}{2\epsilon^2}(\log(54) - \log(0.1))$$

$$143 \geq \frac{1}{2\epsilon^2}(\log(54) - \log(0.1))$$

$$\epsilon^2 \geq \frac{(\log(54) - \log(0.1))}{143\dot{2}}$$

$$|\epsilon| \geq \sqrt{\frac{(\log(54) - \log(0.1))}{286}}$$

$$\Rightarrow \epsilon \geq \sqrt{\frac{(\log(54) - \log(0.1))}{286}}$$

$$\Pr_{T\ \mathcal{D}^m} (\forall h \in \mathcal{H} : |err_T(h) - err_D(h)| \leq \epsilon) > 0.9$$

$$\Pr_{T\ \mathcal{D}^m} \left(\forall h \in \mathcal{H} : |0.03 - err_D(h)| \leq \sqrt{\frac{(\log(54) - \log(0.1))}{286}}\right) > 0.9$$

$$\Rightarrow err_D(h) \leq 0.03 + \sqrt{\frac{(\log(54) - \log(0.1))}{286}} \simeq 0.208149 \simeq 0.21$$

(b) Using Theorem 3.4:

$$\Pr_{T\ \mathcal{D}^m} (\forall h \in \mathcal{H} : \text{if } h \text{ is consistent with } T, \text{ then } err_D(h) \leq \epsilon)\, 1 - \delta$$

$$\Pr_{T\ \mathcal{D}^m} (\forall h \in \mathcal{H} : \text{if } h \text{ is consistent with } T, \text{ then } err_D(h) \leq 0.01)\, 0.9$$

$$\Rightarrow \epsilon = 0.01, \delta = 0.1$$

$$m \geq \frac{1}{\epsilon} \ln\left(\frac{|\mathcal{H}|}{\delta}\right)$$

$$m \geq \frac{1}{0.01} \ln\left(\frac{3^3}{0.1}\right)$$

$$m \geq 100(\ln(27) - \ln(0.1)) \simeq = 559.84$$

$$\Rightarrow m \geq 560$$

Exercise 2

Exercise 3

Exercise 4

See Referencesappendix for code.

```
1  Final probabilities: [0.4 0.4 0.2]
2
3  Tracked weight vectors:
4
5  Round:   1 Weights:   [1.  1.  1.]
6  Round:   2 Weights:   [0.5 0.5 1. ]
7  Round:   3 Weights:   [0.5  0.25 0.5 ]
8  Round:   4 Weights:   [0.25 0.25 0.25]
9  Round:   5 Weights:   [0.25   0.125 0.125]
10 Round:   6 Weights:   [0.125   0.125   0.0625]
11 Round:   7 Weights:   [0.125       0.0625       0.04419417]
```

Exercise 5

See Referencesappendix for code.

```
1  Probabilities:
2
3  Round:   1 Probabilities:   [0.33 0.33 0.33]
4  Round:   2 Probabilities:   [0.46 0.27 0.27]
5  Round:   3 Probabilities:   [0.28 0.51 0.21]
6  Round:   4 Probabilities:   [0.25 0.42 0.33]
7
8  Weight vectors:
9
10 Round:   1 Weights:   [1.  1.  1.]
11 Round:   2 Weights:   [2.83 1.    1.  ]
12 Round:   3 Weights:   [2.83 8.48 1.  ]
13 Round:   4 Weights:   [2.83 8.48 5.32]
```

Exercise 6

Appendix

Code for Exercise 4

```python
1  import numpy as np
2
3
4  def mwu_algorithm(loss_matrix, events, rounds, alpha):
5      # initial weight vector of 1s
6      weights = np.ones((loss_matrix.shape[0]))
7      weights_tracking = {}
8      weights_tracking[0] = weights
9      # more convenient to loop through rounds and events
10     rounds_arr = [i for i in range(rounds)]
11     for round, event in zip(rounds_arr, events):
```

```python
12          # getting the current probabilities, not really needed here
13          p = probabilities(weights)
14          # need to use event-1 as events start at 1 but indexing at 0
15          weights = np.power((1 - alpha), loss_matrix[:, event-1]) * weights
16          # loss isn't really needed
17          loss = calculate_loss(loss_matrix, p, event-1)
18          weights_tracking[round+1] = weights
19
20      return p, weights_tracking
21
22  def probabilities(weights):
23      return weights / np.sum(weights)
24
25  def calculate_loss(loss_matrix, probabilities, event):
26      return np.sum(probabilities * loss_matrix[:, event])
27
28
29  loss_matrix = np.array([[0,1,1,0],
30                          [1,0,1,1],
31                          [1,1,0,0.5]])
32
33  observed_events = [3,1,2,1,2,4]
34
35  p_6, weights_tracking = mwu_algorithm(loss_matrix, observed_events, 6, alpha
    =0.5)
36
37  print(f'Final probabilities: {p_6}\n')
38  print(f'Tracked weight vectors: \n')
39  for key, val in weights_tracking.items():
40      print(f'Round:\t{key + 1}\tWeights:\t{val}')
```

Code for Exercise 5

```python
1   import numpy as np
2   from copy import deepcopy
3
4   def exp3(gamma, rounds, actions, rewards):
5       weights = np.ones((len(actions)))
6       rounds_arr = [i for i in range(rounds)]
7       n = len(actions)
8       # for tracking weights and probabilities
9       weights_tracking = {}
10      probabilities_tracking = {}
11      weights_tracking[0] = np.ones(len(actions))
12      probabilities_tracking[0] = probability_dist(weights, gamma)
13      for round, action in zip(rounds_arr, actions):
14          probabilities = probability_dist(weights, gamma)
15          probabilities_tracking[round] = probabilities
16          reward = rewards[action]
17          weights = update_weights(weights, reward, probabilities, action,
    gamma)
18          weights_tracking[round + 1] = deepcopy(weights)
19
20      probabilities_tracking[rounds] = probability_dist(weights, gamma)
21      return weights_tracking, probabilities_tracking
22
23  def probability_dist(weights, gamma):
24      return (1 - gamma) * (weights / np.sum(weights)) + gamma / len(weights)
25
26  def update_weights(weights, reward, probabilities, action, gamma):
27      n = len(weights)
```

```python
28      # only update chosen action
29      weights[action] = weights[action] * np.exp((gamma * reward) / (n *
    probabilities[action]))
30      return weights
31
32
33 action_seq = np.array([ 1, 2, 3 ])
34 rewards = np.array([ 3, 5, 3 ]) * np.log(2)
35
36 weights, probs = exp3(gamma=0.5, rounds=3, actions=action_seq - 1, rewards=
    rewards)
37
38 print(f'Probabilities: \n')
39 for key, val in probs.items():
40     print(f'Round:\t{key + 1}\tProbabilities:\t{val.round(2)}')
41
42 print(f'\nWeight vectors: \n')
43 for key, val in weights.items():
44     print(f'Round:\t{key + 1}\tWeights:\t{val.round(2)}')
```