Fabian Grob
Simon Michau
Til Mohr

Exercise 1

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

$$\Pr_{T\ \mathcal{D}^m}\left(\forall h \in \mathcal{H} : |err_T(h) - err_D(h)| \leq \epsilon\right) > 1 - \delta$$
$$\Pr_{T\ \mathcal{D}^m}\left(\forall h \in \mathcal{H} : |err_T(h) - err_D(h)| \leq \epsilon\right) > 0.9$$
$$\Rightarrow \delta = 0.1$$

$$m \geq \frac{1}{2\epsilon^2} \log\left(\frac{2|\mathcal{H}|}{\delta}\right)$$
$$143 \geq \frac{1}{2\epsilon^2} \log\left(\frac{2 \cdot 3^3}{0.1}\right)$$
$$143 \geq \frac{1}{2\epsilon^2}(\log(54) - \log(0.1))$$
$$143 \geq \frac{1}{2\epsilon^2}(\log(54) - \log(0.1))$$
$$\epsilon^2 \geq \frac{(\log(54) - \log(0.1))}{143\dot{2}}$$
$$|\epsilon| \geq \sqrt{\frac{(\log(54) - \log(0.1))}{286}}$$
$$\Rightarrow \epsilon \geq \sqrt{\frac{(\log(54) - \log(0.1))}{286}}$$
$$\Pr_{T\ \mathcal{D}^m}\left(\forall h \in \mathcal{H} : |err_T(h) - err_D(h)| \leq \epsilon\right) > 0.9$$
$$\Pr_{T\ \mathcal{D}^m}\left(\forall h \in \mathcal{H} : |0.03 - err_D(h)| \leq \sqrt{\frac{(\log(54) - \log(0.1))}{286}}\right) > 0.9$$
$$\Rightarrow err_D(h) \leq 0.03 + \sqrt{\frac{(\log(54) - \log(0.1))}{286}} \simeq 0.208149 \simeq 0.21$$

(b) Using Theorem 3.4:

$$\Pr_{T\ \mathcal{D}^m}\left(\forall h \in \mathcal{H} : \text{if } h \text{ is consistent with } T\text{, then } err_D(h) \leq \epsilon\right) 1 - \delta$$
$$\Pr_{T\ \mathcal{D}^m}\left(\forall h \in \mathcal{H} : \text{if } h \text{ is consistent with } T\text{, then } err_D(h) \leq 0.01\right) 0.9$$
$$\Rightarrow \epsilon = 0.01, \delta = 0.1$$

$$m \geq \frac{1}{\epsilon} \ln\left(\frac{|\mathcal{H}|}{\delta}\right)$$
$$m \geq \frac{1}{0.01} \ln\left(\frac{3^3}{0.1}\right)$$
$$m \geq 100(\ln(27) - \ln(0.1)) \simeq= 559.84$$
$$\Rightarrow m \geq 560$$

Exercise 2

Exercise 3

Exercise 4

See Referencesappendix for code.

(a)

```
Final probabilities: [0.53950429 0.26975214 0.19074357]

Tracked weight vectors (without initial one vector):

Round:  1 Weights:  [0.5 0.5 1. ]
Round:  2 Weights:  [0.5  0.25 0.5 ]
Round:  3 Weights:  [0.25 0.25 0.25]
Round:  4 Weights:  [0.25  0.125 0.125]
Round:  5 Weights:  [0.125  0.125  0.0625]
Round:  6 Weights:  [0.125      0.0625     0.04419417]
```

Exercise 5

Exercise 6

Appendix

Code for Exercise 4

```
import numpy as np


def mwu_algorithm(loss_matrix, events, rounds, alpha):
    # initial weight vector of 1s
    weights = np.ones((loss_matrix.shape[0]))
    weights_tracking = {}
    # more convenient to loop through rounds and events
    rounds_arr = [i for i in range(rounds)]
    for round, event in zip(rounds_arr, events):
        # need to use event-1 as events start at 1 but indexing at 0
        weights = np.power((1 - alpha), loss_matrix[:, event-1]) * weights
        # getting the current probabilities, not really needed here
        p = probabilities(weights)
        # loss isn't really needed
        loss = calculate_loss(loss_matrix, p, event-1)
        weights_tracking[round] = weights

    return p, weights_tracking

def probabilities(weights):
    return weights / np.sum(weights)

def calculate_loss(loss_matrix, probabilities, event):
    return np.sum(probabilities * loss_matrix[:, event])
```

```python


loss_matrix = np.array([[0,1,1,0],
                        [1,0,1,1],
                        [1,1,0,0.5]])

observed_events = [3,1,2,1,2,4]

p_6, weights_tracking = mwu_algorithm(loss_matrix, observed_events, 6, alpha=0.5)

print(f'Final probabilities: {p_6}\n')
print(f'Tracked weight vectors (without initial one vector): \n')
for key, val in weights_tracking.items():
    print(f'Round:\t{key + 1}\tWeights:\t{val}')
```