Exercise 1

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

$$\begin{aligned}
&\Pr_{T\ \mathcal{D}^m}(\forall h \in \mathcal{H} : |err_T(h) - err_D(h)| \le \epsilon) > 1 - \delta \\
&\Pr_{T\ \mathcal{D}^m}(\forall h \in \mathcal{H} : |err_T(h) - err_D(h)| \le \epsilon) > 0.9 \\
&\Rightarrow \delta = 0.1
\end{aligned}$$

$$\begin{aligned}
m &\ge \frac{1}{2\epsilon^2} \log\left(\frac{2|\mathcal{H}|}{\delta}\right) \\
143 &\ge \frac{1}{2\epsilon^2} \log\left(\frac{2 \cdot 3^3}{0.1}\right) \\
143 &\ge \frac{1}{2\epsilon^2}(\log(54) - \log(0.1)) \\
143 &\ge \frac{1}{2\epsilon^2}(\log(54) - \log(0.1)) \\
\epsilon^2 &\ge \frac{(\log(54) - \log(0.1))}{143\dot{2}} \\
|\epsilon| &\ge \sqrt{\frac{(\log(54) - \log(0.1))}{286}} \\
\Rightarrow \epsilon &\ge \sqrt{\frac{(\log(54) - \log(0.1))}{286}}
\end{aligned}$$

$$\begin{aligned}
&\Pr_{T\ \mathcal{D}^m}(\forall h \in \mathcal{H} : |err_T(h) - err_D(h)| \le \epsilon) > 0.9 \\
&\Pr_{T\ \mathcal{D}^m}\left(\forall h \in \mathcal{H} : |0.03 - err_D(h)| \le \sqrt{\frac{(\log(54) - \log(0.1))}{286}}\right) > 0.9 \\
&\Rightarrow err_D(h) \le 0.03 + \sqrt{\frac{(\log(54) - \log(0.1))}{286}} \simeq 0.208149 \simeq 0.21
\end{aligned}$$

(b) Using Theorem 3.4:

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

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

Exercise 2

Using slide 3.26.

(a) The dimension of the instance space is $l = a \cdot v$. Such a decision scheme can be described using $|h|_\Delta \in O(n(\log n + \log l)) = O(n(\log n + \log(a \cdot v))) = O(n(\log n + \log a + \log v))$ bits.

(b) $n = 10, a = 5, v = 5$??????????????????

Exercise 3

(a) The VC-Dimension is 3. To prove this, let us prove some properties which every Y, that shatters $\mathcal{H}$, must hold.

Notation:

- $a_i, b_i \in \mathbb{R}, i \in \mathbb{N}$
- $y_{i,j} \in Y, i, j \in \mathbb{N} \to y_{i,j} = (a_i, b_j)$ (analog for $y'_{i,j} \in Y'$)

Property 1:

$$\neg\exists y_{i,j}\exists y_{i',j'}\exists y_{i'',j''}(a_i \neq a_{i'} \wedge a_i \neq a_{i''} \wedge a_{i'} \neq a_{i''} \wedge b_i \neq b_{i'} \wedge b_i \neq b_{i''} \wedge b_{i'} \neq b_{i''})$$

In words: There cannot be 3 vectors in Y that have neither their a nor b values in common.
Let's prove this be counterexample.
Let $Y' = \{(a_1, b_1), (a_2, b_2), (a_3, b_3)\} \subseteq Y$. There exists no $h_{a,b} \in \mathcal{H}$, such that $Y' \subseteq S_{h_{a,b}}$. Therefore, $Y' = S_{h_{a,b}} \cap Y$ does not hold.

Property 2:

$$\neg\exists y_{i,j}\exists y_{i',j'}\exists y_{i'',j''}(y_{i,j} \neq y_{i',j'} \wedge y_{i,j} \neq y_{i'',j''} \wedge y_{i',j'} \neq y_{i'',j''}) \wedge ((a_i = a_{i'} \wedge a_i = a_{i''}) \vee (b_i = b_{i'} \wedge b_i = b_{i''}))$$

In words: There cannot be 3 vectors in Y that have their a or b values in common.
Let's prove this be counterexample.
Let $Y' = \{(a_1, b_1), (a_1, b_2)\} \subseteq \{(a_1, b_1), (a_1, b_2), (a_1, b_3)\} \subseteq Y$ (analog when the b-components are equal). Then there exists no $h_{a,b} \in \mathcal{H}$ so that $Y' \subseteq S_{h_{a,b}}$ but $(a_1, b_3) \notin S_{h_{a,b}}$. This is true, since the only functions $h_{a,b}$ with $Y' \subseteq S_{h_{a,b}}$ are where $a = a_1$, thus also $(a_1, b_3) \notin S_{h_{a,b}}$, which is a contradiction.

Property 3:

$$\neg\exists y_{i,j}\exists y_{i',j'}\exists y_{i'',j''}(y_{i,j} \neq y_{i',j'} \wedge y_{i,j} \neq y_{i'',j''} \wedge y_{i',j'} \neq y_{i'',j''}) \wedge (a_i = a_{i'} \wedge b_{i'} = b_{i''})$$

In words: There cannot be a vector containing of an a-component that also exists in another vector and a b-component that also exists in another vector.
Let's prove this be counterexample.
Let $Y' = \{(a_1, b_2)\} \subseteq \{(a_1, b_1), (a_1, b_2), (a_2, b_2)\} \subseteq Y$. For all $h_{a,b} \in \mathcal{H}$ with $Y' \subseteq S_{h_{a,b}}$ $a = a_1 \vee b = b_2$ must hold. However, for such $h_{a,b}$ $(a_1, b_1) \in S_{h_{a,b}}$ or $(a_2, b_2) \in S_{h_{a,b}}$ would also hold. This is a contradiction.

Thus, the only valid form of Y must be (or analog when two b-components are equal):

$$Y := \{(a_1, b_1), (a_1, b_2), (a_2, b_2)\}, a_1 \neq a_2 \wedge b_1 \neq b_2 \wedge b_1 \neq b_3 \wedge b_2 \neq b_3$$

We can prove that Y shatters $\mathcal{H}$:

- $Y' = \emptyset$:
  We can use h_{a_0,b_0} with $a_0 \notin \{a_1, a_2\}, b_0 \notin \{b_1, b_2, b_3\}$. Thus, $Y \notin S_{h_{a_0,b_0}}$. So, $Y' = \emptyset = S_{h_{a_0,b_0}} \cap Y$.

- $Y' = \{(a_1, b_i)\}$, $i \in \{1, 2\}$:
  We can use h_{a_0,b_i} with $a_0 \notin \{a_1, a_2\}$. Thus, $Y' = \emptyset = S_{h_{a_0,b_i}} \cap Y$.
- $Y' = \{(a_2, b_3)\}$:
  We can use h_{a_2,b_3}. Thus, $Y' = \emptyset = S_{h_{a_0,b_i}} \cap Y$.
- $Y' = \{(a_1, b_1), (a_1, b_2)\}$:
  We can use h_{a_1,b_1}. Thus, $Y' = \emptyset = S_{h_{a_0,b_i}} \cap Y$.
- $Y' = \{(a_1, b_i), (a_2, b_3)\}$, $i \in \{1, 2\}$:
  We can use h_{a_2,b_i}. Thus, $Y' = \emptyset = S_{h_{a_0,b_i}} \cap Y$.
- $Y' = Y$:
  We can use h_{a_1,b_3}. Thus, $Y' = \emptyset = S_{h_{a_0,b_i}} \cap Y$.

So, the VC-Dimension is at least 3. Let's now prove, that no Y with $|Y| > 3$ shatters $\mathcal{H}$.

For now, let Y be $Y := \{(a_1, b_1), (a_1, b_2), (a_2, b_2)\}$, $a_1 \neq a_2 \wedge b_1 \neq b_2 \wedge b_1 \neq b_3 \wedge b_2 \neq b_3$ (which we proved was the only valid form of Y with $|Y| = 3$). We can now try to add any $y_{i,j}$ to Y:

- Let's try to add $y_{1,j} = (a_1, b_j)$, $j \in \mathbb{N}$. This would violate **Property 1**. Thus, we cannot add any such $y_{1,j}$.
- Let's try to add $y_{2,j} = (a_2, b_j)$, $j \in \mathbb{N}$. Since $(a_2, b_3) \in Y$, $b_j \in \mathbb{R} \setminus \{b_3\}$. Then the condition in the remark would not hold for $Y' = Y$. Thus, we cannot add any such $y_{2,j}$.
- Let's try to add $y_{3,4} = (a_3, b_4)$, $a_3 \in \mathbb{R} \setminus \{a_1, a_2\}$, $b_4 \notin \{b_1, b_2, b_3\}$. This would violate **Property 1**. Thus, we cannot add any such $y_{3,4}$.
- Let's try to add $y_{3,j} = (a_3, b_j)$, $j \in \{1, 2\}$, $a_3 \in \mathbb{R} \setminus \{a_1, a_2\}$. Then, we cannot find any $h_{a,b} \in \mathcal{H}$ for $Y' = Y$, so that the condition in the remark holds. Thus, we cannot add any such $y_{3,j}$
- Let's try to add $y_{3,j} = (a_3, b_3)$, $a_3 \in \mathbb{R} \setminus \{a_1, a_2\}$. Then, for $Y' = \{(a_1, b_2), (a_2, b_3), (a_3, b_3)\}$ we cannot find any $h_{a,b} \in \mathcal{H}$, for which $Y' \subseteq S_{h_{a,b}}$ but $(a_1, b_1) \notin S_{h_{a,b}}$. Thus, we cannot add any such $y_{3,j}$.

So, we cannot add any element to Y. Thus, all Y that shatter $\mathcal{H}$ must be at most $|Y| \leq 3$. Therefore, the VC Dimension is 3.

(b) The VC-Dimension is ∞.
Let be $Y \subseteq \mathfrak{X} = \Sigma^*$. For any $Y' \subseteq Y$ we can choose $L := Y'$. Thus, $Y' = S_{h_L}$, so also $S_{h_L} \subseteq Y$. So $Y \cap S_{h_L} = S_{h_L} = Y'$. So Y can also be infinite in size.

Exercise 4

(a) See Referencesappendix for code.

```
Final probabilities: [0.4 0.4 0.2]

Tracked weight vectors:

Round:  1 Weights:  [1. 1. 1.]
Round:  2 Weights:  [0.5 0.5 1. ]
Round:  3 Weights:  [0.5  0.25 0.5 ]
Round:  4 Weights:  [0.25 0.25 0.25]
```

```
Round:  5 Weights:  [0.25   0.125 0.125]
Round:  6 Weights:  [0.125  0.125  0.0625]
Round:  7 Weights:  [0.125      0.0625     0.04419417]
```

(b) No, the weights are always the same. This is because the loss matrix does not change throughout the algorithm, so the loss is always the same. If all the events still happen in the sequence, only in a different order, this won't have an effect as the loss will still be included in the updated weight.

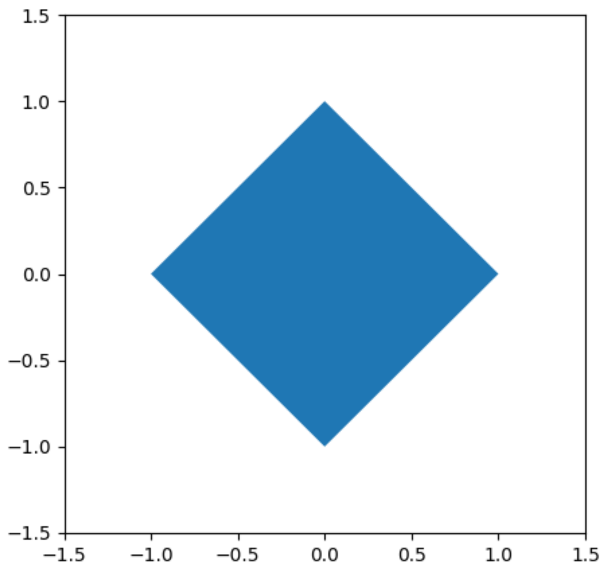
Exercise 5

(a) See Referencesappendix for code.

```
Probabilities:

Round:  1       Probabilities:  [0.33 0.33 0.33]
Round:  2       Probabilities:  [0.46 0.27 0.27]
Round:  3       Probabilities:  [0.28 0.51 0.21]
Round:  4       Probabilities:  [0.25 0.42 0.33]

Weight vectors:

Round:  1       Weights:        [1. 1. 1.]
Round:  2       Weights:        [2.83 1.   1.  ]
Round:  3       Weights:        [2.83 8.48 1.  ]
Round:  4       Weights:        [2.83 8.48 5.32]
```

(b) $w_1^{(4)}$ and $w_3^{(4)}$ are different because with $\gamma = 0.5$ we put a certain weight on exploration. Therefore, even with the same reward, different actions can have different weights as γ is part of the weight updating calculation.

Exercise 6



(a) (i)

(ii) The shape of $B_1^3 \subset \mathcal{R}^3$ would look like a octahedron with the same edge length ($\sqrt{2}$).

(b) $|diag(B_1^2)| = 2 \xrightarrow{\text{Pythagoras}}$ edge length of $B_1^2 = \sqrt{2} \Rightarrow vol(B_1^2) = \sqrt{2}^2 = 2$

$vol(B_1^3) = \frac{1}{3}\sqrt{2}a^3$ with a being the edge length of the octahedron $\Rightarrow vol(B_1^3) = \frac{1}{3}\sqrt{2}^4 = \frac{1}{3}4 = \frac{4}{3} \simeq 1.33$
A more general formula for the volume $vol(B_1^l)$ is $vol(B_1^l) = \frac{2^l}{l!}$ (cmp. this paper).
Using $l = 1, 2$ yields the same results.

(c)

$$\lim_{l \to \inf} \text{vol}(B_1^l) = \frac{2^l}{l!}$$
$$\Rightarrow \lim_{l \to \inf} \text{vol}(B_1^l) = \frac{2}{1} \cdot \frac{2}{2} \cdot \frac{2}{3} \cdot \dots \cdot \frac{2}{l-1} \cdot \frac{2}{l}$$
$$\leq 2 \cdot \frac{2}{l} = \frac{4}{l}$$
$$\Rightarrow \lim_{n \to \inf} \frac{4}{l} = 0$$

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


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: \n')
for key, val in weights_tracking.items():
    print(f'Round:\t{key + 1}\tWeights:\t{val}')
```

Code for Exercise 5

```
import numpy as np
from copy import deepcopy

def exp3(gamma, rounds, actions, rewards):
    weights = np.ones((len(actions)))
    rounds_arr = [i for i in range(rounds)]
    n = len(actions)
    # for tracking weights and probabilities
    weights_tracking = {}
    probabilities_tracking = {}
    weights_tracking[0] = np.ones(len(actions))
    probabilities_tracking[0] = probability_dist(weights, gamma)
    for round, action in zip(rounds_arr, actions):
        probabilities = probability_dist(weights, gamma)
        probabilities_tracking[round] = probabilities
        reward = rewards[action]
        weights = update_weights(weights, reward, probabilities, action,
    gamma)
        weights_tracking[round + 1] = deepcopy(weights)

    probabilities_tracking[rounds] = probability_dist(weights, gamma)
    return weights_tracking, probabilities_tracking

def probability_dist(weights, gamma):
    return (1 - gamma) * (weights / np.sum(weights)) + gamma / len(weights)

def update_weights(weights, reward, probabilities, action, gamma):
    n = len(weights)
    # only update chosen action
    weights[action] = weights[action] * np.exp((gamma * reward) / (n *
    probabilities[action]))
    return weights


action_seq = np.array([ 1, 2, 3 ])
rewards = np.array([ 3, 5, 3 ]) * np.log(2)

weights, probs = exp3(gamma=0.5, rounds=3, actions=action_seq - 1, rewards=
    rewards)

print(f'Probabilities: \n')
for key, val in probs.items():
    print(f'Round:\t{key + 1}\tProbabilities:\t{val.round(2)}')

print(f'\nWeight vectors: \n')
for key, val in weights.items():
    print(f'Round:\t{key + 1}\tWeights:\t{val.round(2)}')
```