Exercise 4

Since regret of A on each period of 2^m rounds is at most $\alpha\sqrt{2^m}$, we get that:

$$\sum_{t=2^m}^{2^{m+1}-1} \left(f_t(w^{(t)}) - f_t(w^*)\right) \leq \alpha\sqrt{2^m}$$

Let $M = \log_2 T$. Then, total regret is given by:

$$\sum_{m=0}^{M}\sum_{t=2^m}^{2^{m+1}-1} \left(f_t(w^{(t)}) - f_t(w^*)\right).$$

Now, let's obtain an upper bound for this total regret:

$$\text{Regret}_A(w^*, T) = \sum_{t=1}^{T}\left(f_t(w^{(t)}) - f_t(w^*)\right) = \sum_{m=0}^{M}\sum_{t=2^m}^{2^{m+1}-1}\left(f_t(w^{(t)}) - f_t(w^*)\right) \leq$$

$$\leq \sum_{m=0}^{M}\alpha\sqrt{2^m} = \alpha\sum_{m=0}^{M} 2^{\frac{1}{2}m} = \alpha\left[\sum_{m=0}^{\frac{1}{2}M} 2^m + \sum_{m=0}^{\frac{1}{2}M} 2^{m-1/2}\right] =$$

$$= \alpha\left[2^{\frac{1}{2}M+1} - 1 + \frac{1}{\sqrt{2}}\left(\sum_{m=0}^{\frac{1}{2}M} 2^m\right)\right] = \alpha\left[2^{\frac{1}{2}\log_2 T + 1} - 1 + \frac{1}{\sqrt{2}}\left(2^{\frac{1}{2}\log_2 T + 1} - 1\right)\right] =$$

$$= \alpha\left[2\sqrt{T} - 1 + \frac{2\sqrt{T}}{\sqrt{2}} - \frac{1}{\sqrt{2}}\right] < \alpha\left[2\sqrt{T} + \frac{2\sqrt{T}}{\sqrt{2}}\right] =$$

$$= [2 + \sqrt{2}]\,\alpha\sqrt{T} = \frac{\sqrt{2}}{\sqrt{2} - 1}\,\alpha\sqrt{T}$$

$$q.e.d.$$

Exercise 5

Since $L_D(h_r) = \underset{x \sim D}{\mathbb{E}}[\ell(h_r(x), h^*(x)^*)]$, $L_D(h_r) \leq 1$.

~~Since for every round by the prediction of A is bounded~~

$$\mathbb{E}[L_D(h_r)] = \frac{1}{T} \sum_{r=1}^{T} L_D(h_r)$$

Assume that $\mathbb{E}[L_D(h_r)] > \frac{M_A(\mathcal{H})}{T}$ Then:

$$\sum_{r=1}^{T} L_D(h_r) > M_A(\mathcal{H})$$

Using linearity of expectation, we have:

$$\sum_{r=1}^{T} L_D(h_r) = \sum_{r=1}^{T} \underset{x \sim D}{\mathbb{E}}[\ell(h_r(x), h^*(x))] =$$

$$= \underset{x \sim D}{\mathbb{E}}\left[\sum_{r=1}^{T} \ell(h_r(x), h^*(x))\right] > M_A(\mathcal{H})$$

This means that for ~~[for]~~ any sample S or size T, that for all $x_t \in S$, $\sum_{r=1}^{T} \ell(h_r(x_t), h^*(x_t)) > M_A(\mathcal{H})$, which means that on average, on each step t, there are $\frac{M_A(\mathcal{H})}{T}$ hypothesis which make a misstake on for x_t.

Assuming that the true hypothesis exists, we get that by pigeonhole principle, there will be two identical hypothesises in $\mathcal{H}$, which is impossible.

Therefore, the assumption was wrong and

$$\mathbb{E}[L_D(h_r)] \leq \frac{M_A(\mathcal{H})}{T}$$

q.e.d.

Code:

```python
import numpy as np
import numpy.random
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
import math




def generate_data_set_with_labels(n):
    '''
    generates random data set of size n by sampling points uniformly
    at random within the unit sphere in the Euclidean plane and labels
    the points according to a randomly chosen linear decision boundary
    '''
    # generate two points which will represent linear decision boundary

    data_set = []
    labels = []

    r = np.random.uniform(0,1.0, 2)
    theta = np.random.uniform(0,2 * math.pi, 2)
    x1 = r[0] * math.cos(theta[0])
    x2 = r[1] * math.cos(theta[1])
    y1 = r[0] * math.sin(theta[0])
    y2 = r[1] * math.sin(theta[1])
    m = (y1-y2)/(x1-x2)
```

```python
    for i in range (n):

        r = np.random.uniform(0,1.0)

        theta = np.random.uniform(0,2 * math.pi)

        x = r * math.cos(theta)

        y = r * math.sin(theta)

        data_set.append(np.array([x,y]))

        if x - x1 > m*(y - y1):

            labels.append(-1)

        else:

            labels.append(1)


    np_data_set = np.array(data_set)

    np_labels = np.array(labels)

    return np_data_set, np_labels




def plot_data(data, labels):

    plt.figure()

    t = np.linspace(0,math.pi*2,100)

    plt.plot(np.cos(t), np.sin(t), linewidth=1)

    plt.scatter(data[:, 0], data[:, 1], c=labels)

    plt.show()




def plot_mistakes(mistakes):

    plt.figure()

    lst = []

    for i in range(100):
```

```python
        lst.append(i)
    plt.scatter(lst, mistakes)


    plt.show()



def perceptron_algorithm(data, labels):
    w = np.zeros((100, 2))
    t = 0
    mistakes = []
    mistakes_counter = 0
    for x_t in data:

        p_t = np.sign(np.dot(w[t],x_t))
        y_t = labels[t]
        if p_t != y_t:
            mistakes_counter += 1
        mistakes.append(mistakes_counter)

        if t+1<100:
            if p_t*y_t <= 0:
                w[t+1] = w[t] + y_t*x_t
            else:
                w[t+1] = w[t]
        t += 1
    return mistakes
```
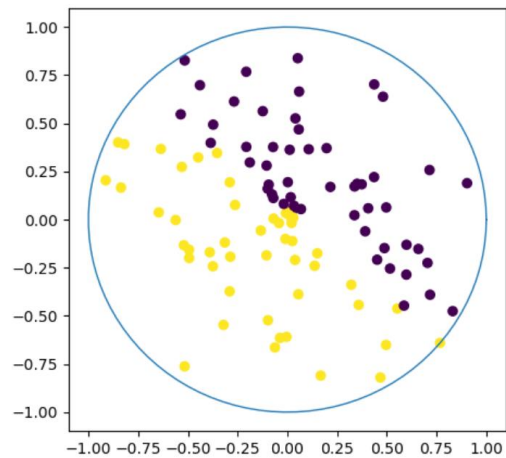
```
data, labels = generate_data_set_with_labels(n=100)

print(data.size, labels.size, np.sign(-2))

plot_data(data, labels)

mistakes = perceptron_algorithm(data, labels)

plot_mistakes(mistakes)
```


Mistakes: