Predictions of Individual Sequences - Home Assignment

Tamim EL AHMAD - Amin DHAOU

2020/03/06

Part 1. Link between online learning and game theory

Full information feedback

1. For the game "Rock, Paper, Scissors" :

— $M = N = 3$;

— $L = \begin{array}{cc} & \begin{array}{ccc} R & P & S \end{array} \\ \begin{array}{c} R \\ P \\ S \end{array} & \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix} \end{array}$

In questions 3 and 4, the player is playing a EWA algorithm with $\eta = 1$.

3. In this question, the adversary plays with a fixed strategy $q = (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$.

(a) $\ell_t(i) = L(i, j_t)$

(b) We perform all the experiments with $p_0 = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$.

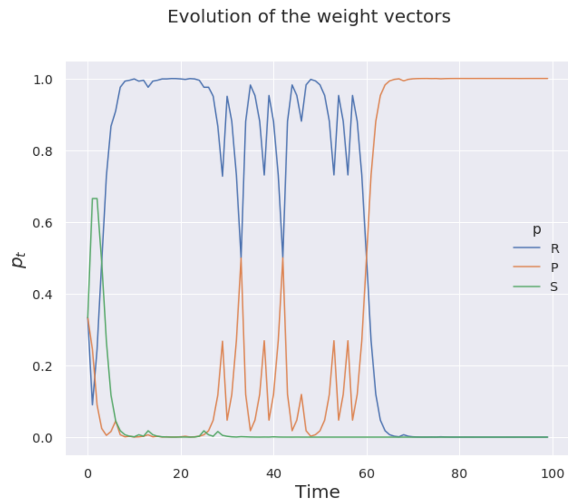


FIGURE 1 – Evolution of the probabilities of playing "Rock", "Paper" or "Scissors".

The best strategy seems to be playing "Paper", which is normal since the adversary plays "Rock" with probability 0.5 at each time and "Paper" and "Scissors" with probability 0.25. And so, we see here that the player tends quickly to not play "Scissors" and to play "Paper".

(c)

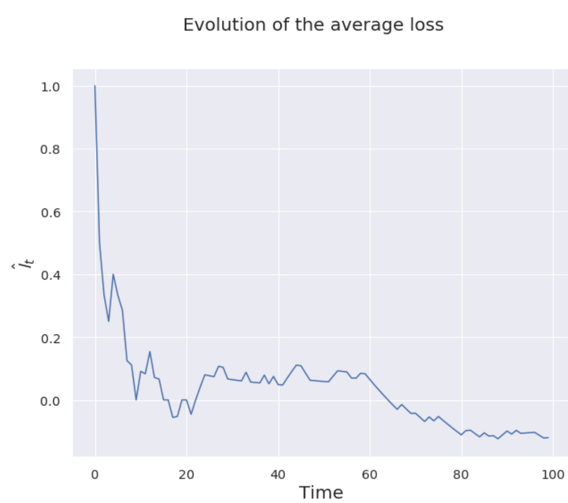


FIGURE 2 – Evolution of the average loss.

(d)

FIGURE 3 – Evolution of the cumulative regret.

(e)

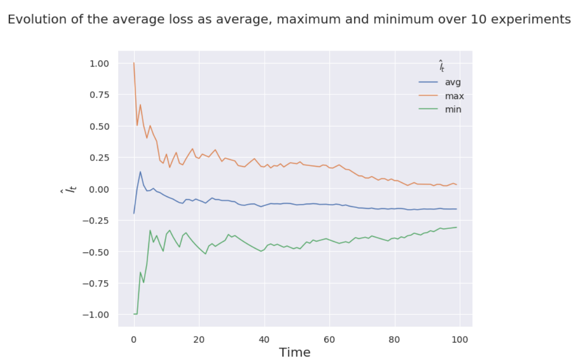


FIGURE 4 – Evolution of the average loss as average, maximum and minimum over 10 experiments.

(f)

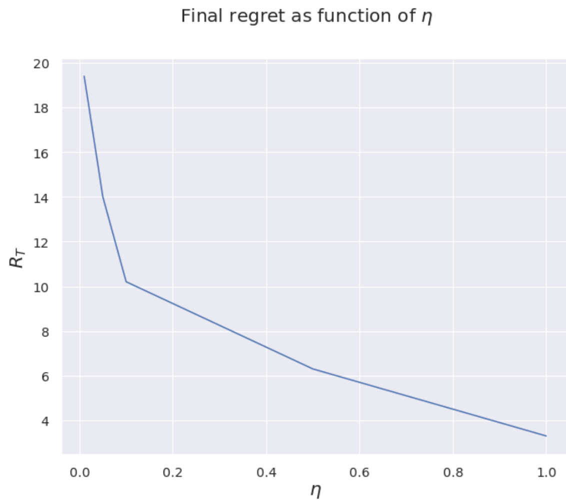


FIGURE 5 – Final regret taken as the average of 10 expirements as function of η.

We can see that the best η in practice is 1. It is coherent since, for the EWA update, the formula is, $\forall i \in [M]$:

$$p_{t+1}(i) = \frac{p_t(i)exp(-\eta\ell_t(i))}{\sum_{j=1}^{M} p_t(j)exp(-\eta\ell_t(j))}$$

So the greater η is, the greater $exp(-\eta\ell_t(i))$ is for negative and smaller loss $\ell_t(i)$ and so the greater $p_{t+1}(i)$ is. Similarly, the greater η is, the smaller $exp(-\eta\ell_t(i))$ is for positive and greater loss $\ell_t(i)$ and so the smaller $p_{t+1}(i)$ is.

4. In this question, the adversary is playing a EWA algorithm with $\eta = 0.05$.

(a) We perform this experiment with $q_0 = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$.

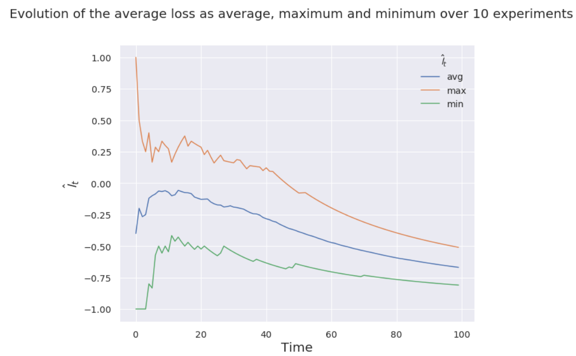


FIGURE 6 – Evolution of the average loss as average, maximum and minimum over 10 experiments for an adaptive adversary with $\eta_{player} = 1$ and $\eta_{adv} = 0.05$.

(b)

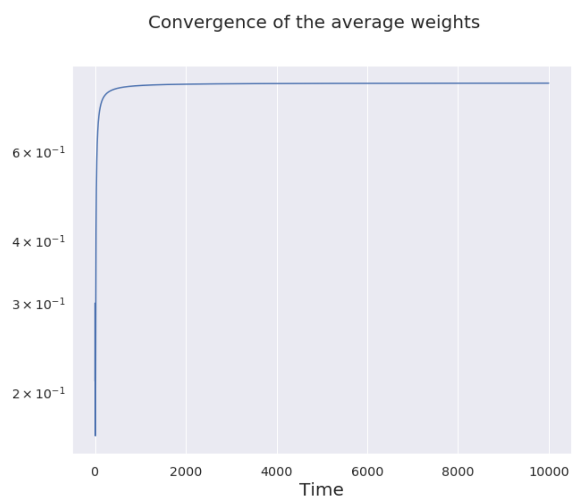


FIGURE 7 – Convergence of $\|\bar{p}_t - (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})\|_2$ for a random initialisation of p_0 and q_0.

Bandit feedback

In questions 6, 7 and 8, the player is playing a $EXP3$ algorithm with $\eta = 1$.

6. In this question, the adversary plays with a fixed strategy $q = (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$.

(b) We perform all the experiments with $p_0 = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$.

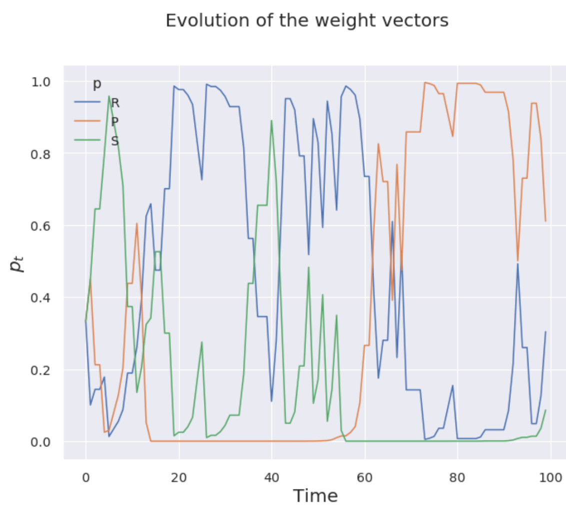


FIGURE 8 – Evolution of the probabilities of playing "Rock", "Paper" or "Scissors".

We see here that the player coherently tends to not play "Scissors" and to play "Paper" but it is way less quick and stable that for EWA update, which is normal since here, the player learns the game while playing.

(c)

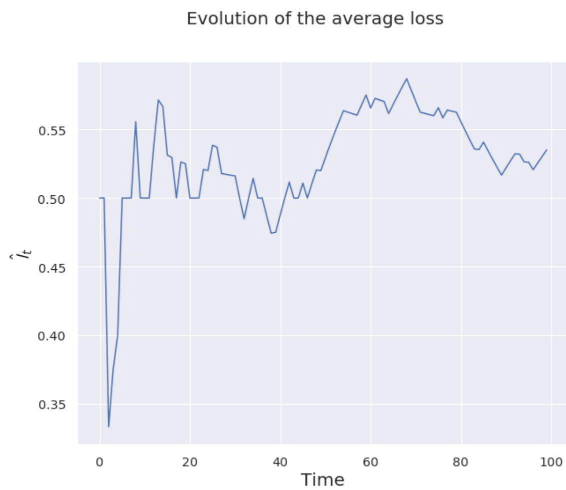


FIGURE 9 – Evolution of the average loss.

(d)

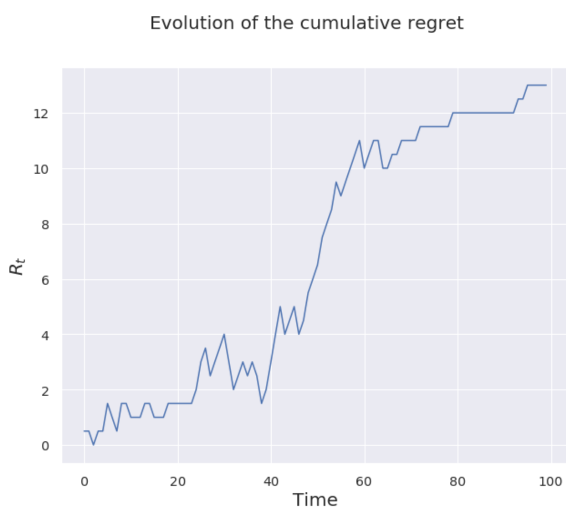


FIGURE 10 – Evolution of the cumulative regret.

(e)

FIGURE 11 – Evolution of the average loss as average, maximum and minimum over 10 experiments.

(f)

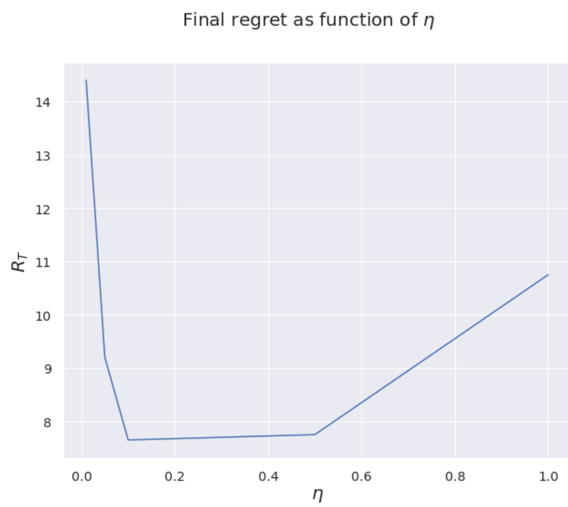


FIGURE 12 – Final regret taken as the average of 10 expirements as function of η.

We can see that the best η in practice is 0.1. For the $EXP3$ update, the formula is, $\forall i \in [M]$:

$$p_{t+1}(i) \propto p_t(i) exp(-\eta \hat{\ell}_t(i))$$

So the greater η is, the smaller $exp(-\eta \ell_t(i))$ is for greater loss $\ell_t(i)$ and so the smaller $p_{t+1}(i)$ is. So theoretically, the best η is 1. Here, as the player does not know about the game and learn it while playing, it is more difficult to obtain theoretically good results (the variance is greater). So, with only 10 experiments, it is normal to obtain odd results.

7. In this question, the adversary is playing a EWA algorithm with $\eta = 0.05$.

(a) We perform this experiment with $q_0 = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$.

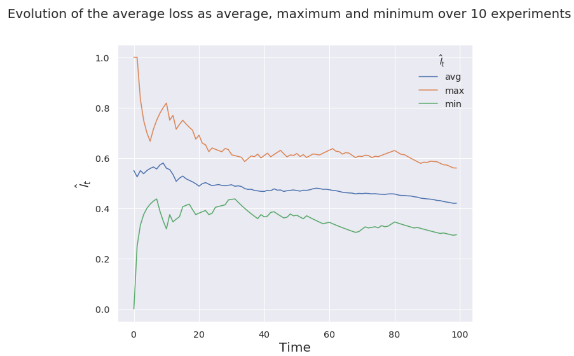


FIGURE 13 – Evolution of the average loss as average, maximum and minimum over 10 experiments for an adaptive adversary with $\eta_{player} = 1$ and $\eta_{adv} = 0.05$.

(b)

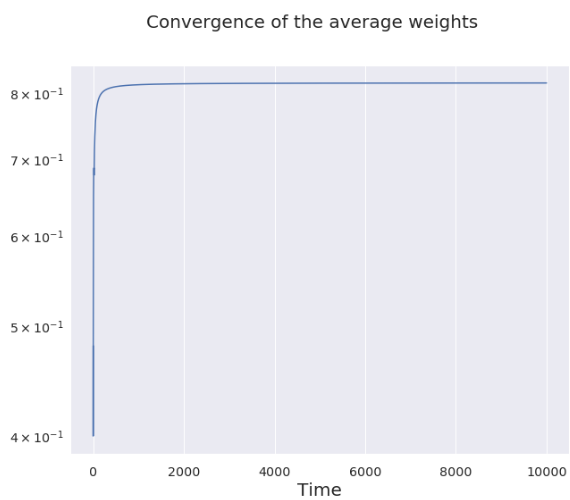


FIGURE 14 – Convergence of $\|\bar{p}_t - (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})\|_2$ for a random initialisation of p_0 and q_0.

8. In this question, the adversary is playing a UCB algorithm.

(a)

FIGURE 15 – Evolution of the average loss as average, maximum and minimum over 10 experiments for an adaptive adversary playing a UCB algorithm and a player playing a $EXP3$ algorithm with $\eta_{player} = 1$.

(b)

FIGURE 16 – Convergence of $\|\bar{p}_t - (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})\|_2$ for a random initialisation of p_0 and q_0.

We can see in Figure 15 that the average and maximum average loss over 10 experiments is always above $O.5$, and even the minimum converges close to 0.5. Moreover, in Figure 16, we can see that it is way more difficult for the player to converge to the strategy corresponding to Nash equilibrium. We can conclude that UCB wins against $EXP3$.

9. In this question, the player is playing a $EXP3.IX$ strategy with $\eta = 1$ and $\gamma = 0.5$ and the adversary plays with a fixed strategy $q = (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$.

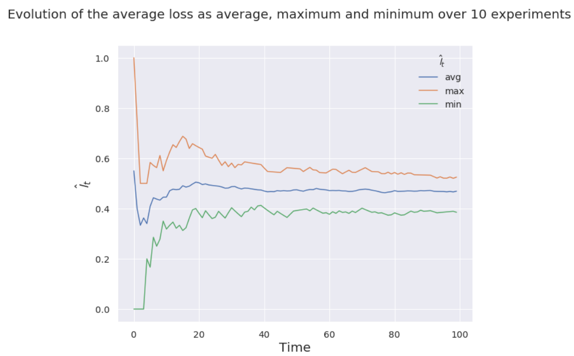


FIGURE 17 – Evolution of the average loss as average, maximum and minimum over 10 experiments.

We can see that there is less variance than in Figure 11.

10. We implemented the prisoner's dilemma with 3 different settings.

(a) **Player plays EWA with $\eta = 1$ and adversary plays EWA with $\eta = 0.05$.**

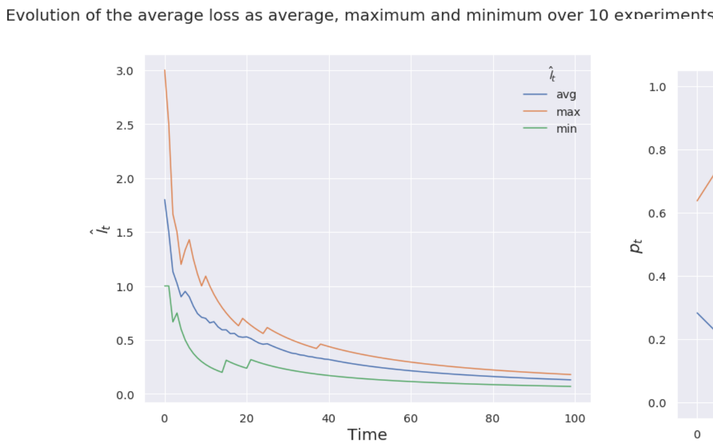


FIGURE 18 – Evolution of the average loss as average, maximum and minimum over 10 experiments.

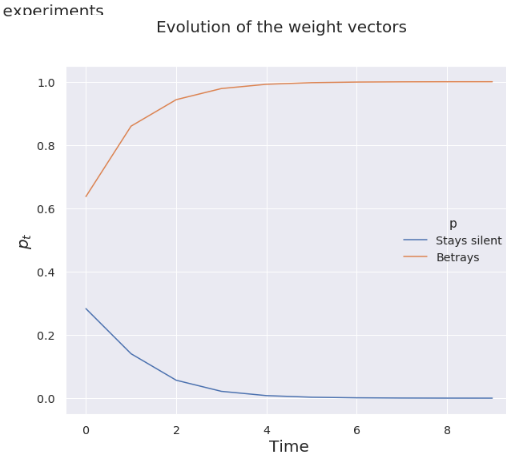


FIGURE 19 – Evolution of the probabilities of "not talking" or "talking".

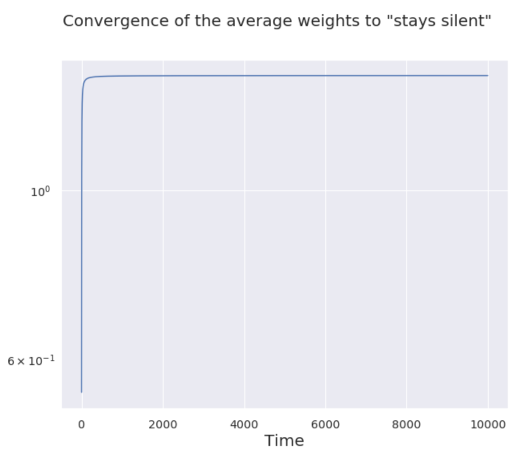


FIGURE 20 – Convergence of $\|\bar{p}_t - (1,0)\|_2$ for a random initialisation of p_0 and q_0.

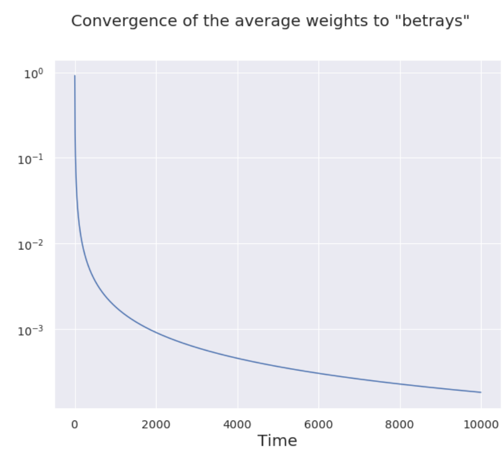


FIGURE 21 – Convergence of $\|\bar{p}_t - (0,1)\|_2$ for a random initialisation of p_0 and q_0.

(b) **Player plays EXP3 with $\eta = 1$ and adversary plays EWA with $\eta = 0.05$.**

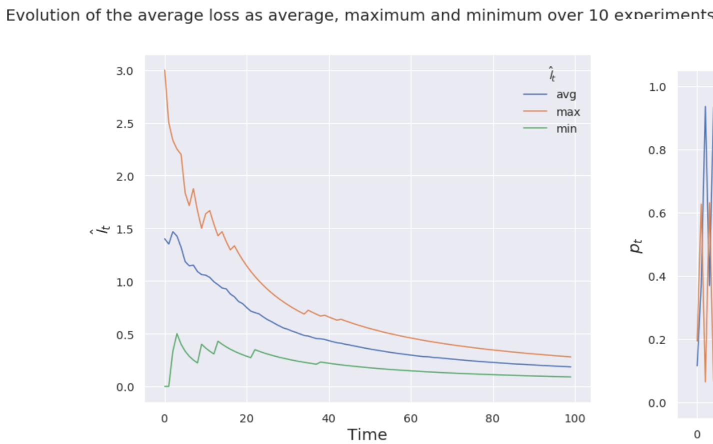


FIGURE 22 – Evolution of the average loss as average, maximum and minimum over 10 experiments.

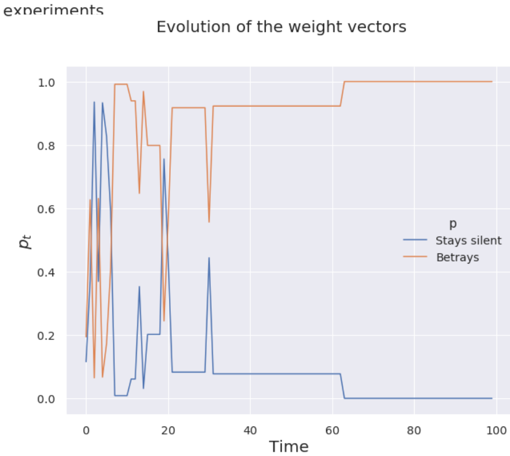


FIGURE 23 – Evolution of the probabilities of "not talking" or "talking".

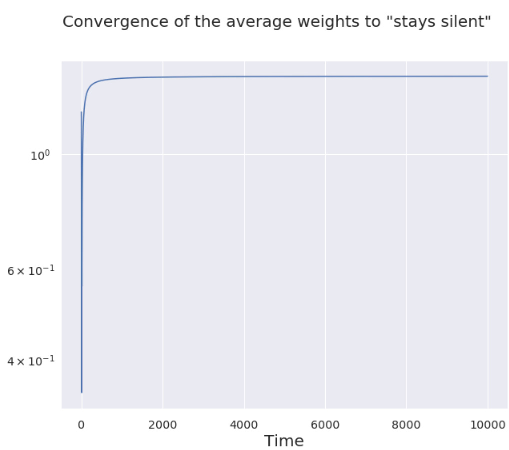


FIGURE 24 – Convergence of $\|\bar{p}_t - (1,0)\|_2$ for a random initialisation of p_0 and q_0.

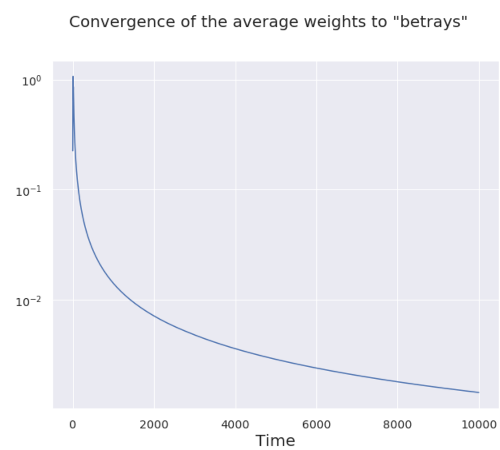


FIGURE 25 – Convergence of $\|\bar{p}_t - (0,1)\|_2$ for a random initialisation of p_0 and q_0.

(c) **Player plays EXP3 with $\eta = 1$ and adversary plays UCB.**

FIGURE 26 – Evolution of the average loss as average, maximum and minimum over 10 experiments.

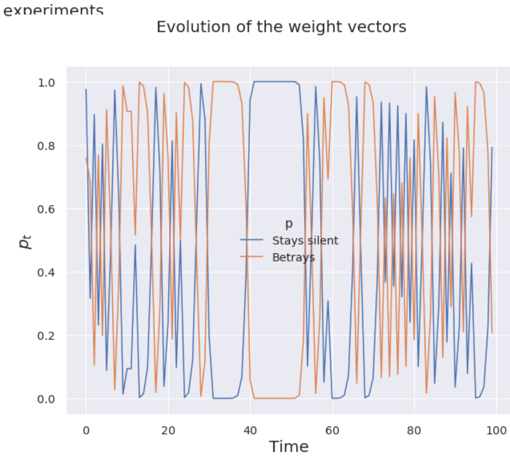


FIGURE 27 – Evolution of the probabilities of "not talking" or "talking".

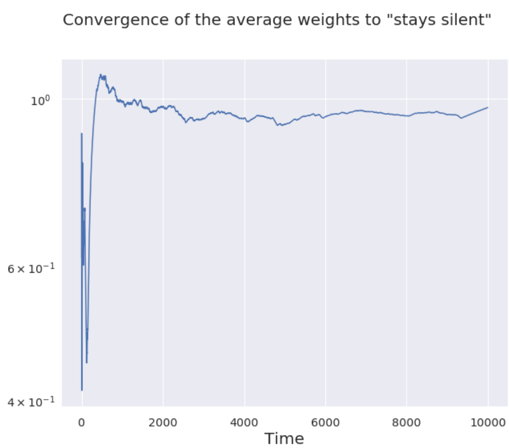


FIGURE 28 – Convergence of $\|\bar{p}_t - (1,0)\|_2$ for a random initialisation of p_0 and q_0.

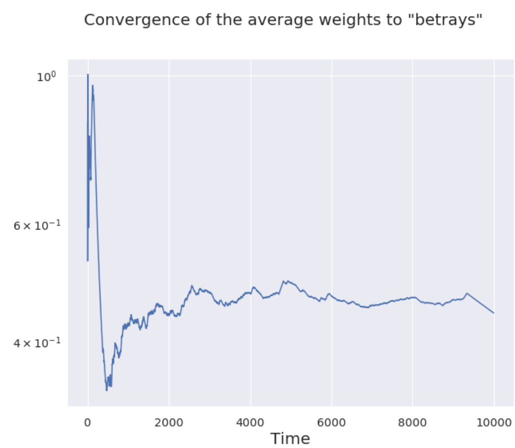


FIGURE 29 – Convergence of $\|\bar{p}_t - (0,1)\|_2$ for a random initialisation of p_0 and q_0.

Finally, we see that for settings (a) and (b), the player should win and that's why his weights converge quickly to "Betrays". From personal interest, his loss is 0 if he decides to betray and the adversary stays silent, and 2 if they both betray. Whereas if he stays silent, his loss is 1 if the adversary stays silent too but 3 is he betrays him. As he loses too much by staying silent, he converges quickly to "Betrays". His average loss tends to 0 so he indeed wins.

In settings (c), it is not suprising to see that the player loses, his average loss tends to ≈ 2.25, as UCB is a better strategy than $EXP3$. But as the difference between UCB and $EXP3$ is slighter than EWA with $\eta = 1$ and $\eta = 0.05$ and $EXP3$ with $\eta = 1$ and EWA with $\eta = 0.05$, the weights of the player still converge to "Betrays", but it is way more difficult. And that's why the average loss converges approximately to a loss in which both player and adversary betray each other.

Part 2. Theory – Sleeping experts

11. (a) Let's note $g : x \to log(1+x) - x + x^2$. Deriving g, we get $g'(x) = \frac{1}{1+x} - 1 + 2 * x$ To get the sign of g, we look for the values for which g' is equal to zero :

$$g'(x) = 0 \Longleftrightarrow x = 0 \text{ or } x = -1/2$$

Then, we can write the variation table for $x \geq \frac{1}{2}$:

x	$-\frac{1}{2}$		0		$+\infty$
$g'(x)$		$-$	0	$+$	
$g(x)$		$\searrow$	0	$\nearrow$	

Finally, we have :

$$\boxed{\log(1+x) \geq x - x^2}$$

(b) Denoting $W_t = \sum_{k=1}^{K} w_t(k)$ and $w_t(k) = \prod_{s=1}^{t-1} (1 + \eta(k)\,(p_s \cdot \ell_s - \ell_s(k)))$ if $t \geq 2$ and $w_1(k) = 1$, for all $k \in \mathcal{X}$ and $t \geq 1$ we can write :

$$\begin{aligned}
\log W_{t+1} = \log \sum_{k=1}^{K} w_T(k) &\geq \log w_T(k) && \text{as } w_T(k) \geq 0 \text{ for all } k \\
&\geq \log \prod_{t=1}^{T} (1 + \eta(k)\,(p_t \cdot \ell_t - \ell_t(k))) && \text{by replacing } w_T(k) \\
&\geq \sum_{t=1}^{T} \log (1 + \eta(k)\,(p_t \cdot \ell_t - \ell_t(k))) \\
&\geq \eta(k) \sum_{t=1}^{T} (p_t \cdot \ell_t - \ell_t(k)) - (\eta(k))^2 \sum_{t=1}^{T} (p_t \cdot \ell_t - \ell_t(k))^2 && \text{as } \log 1 + x \geq x - x^2
\end{aligned}$$

(c) For all $t \geq 1$, $w_t(k) = \prod_{s=1}^{t-1} (1 + \eta(k)(p_s \cdot \ell_s - \ell_s(k)))$, thus we have :

$$\begin{aligned} w_{t+1}(k) &= w_t(k)(1 + \eta(k)(p_t \cdot \ell_t - \ell_t(k))) \\ &= w_t(k) + w_t(k)\eta(k)(p_t \cdot \ell_t - \ell_t(k)) \end{aligned}$$

Hence,

$$\begin{aligned} W_{t+1} &= \sum_{k=1}^{K} w_{t+1}(k) = \sum_{k=1}^{K} w_t(k)(1 + \eta(k)(p_t \cdot \ell_t - \ell_t(k))) \\ &= \sum_{k=1}^{K} w_t(k) + \sum_{k=1}^{K} w_t(k)\eta(k)(p_t \cdot \ell_t - \ell_t(k)) \\ &= W_t + p_t \cdot \ell_t \sum_{k=1}^{K} w_t(k)\eta(k) - \sum_{k=1}^{K} w_t(k)\eta(k)\ell_t(k) \\ &= W_t + p_t \cdot \ell_t \sum_{k=1}^{K} w_t(k)\eta(k) - \sum_{j=1}^{K} w_t(j)\eta(j) \sum_{k=1}^{K} \ell_t(k)p_t(k) \qquad \text{using } p_t(k) = \frac{\eta(k)w_t(k)}{\sum_{j=1}^{K} \eta(j)w_t(j)} \\ &= W_t + p_t \cdot \ell_t \sum_{k=1}^{K} w_t(k)\eta(k) - p_t \cdot \ell_t \sum_{j=1}^{K} w_t(j)\eta(j) \qquad \text{because } \sum_{k=1}^{K} \ell_t(k)p_t(k) = p_t \cdot \ell_t \\ &= W_t \end{aligned}$$

Hence, we have $W_{T+1} = W_1 = \sum_{k=1}^{K} w_1(k) = K$ and finally $\boxed{\log W_{T+1} = \log K}$

(d) Using question (b), we have :

$$\begin{aligned} \eta(k) \sum_{t=1}^{T} (p_t \cdot \ell_t - \ell_t(k)) &\leq \log W_{t+1} + (\eta(k))^2 \sum_{t=1}^{T} (p_t \cdot \ell_t - \ell_t(k))^2 \\ \implies \sum_{t=1}^{T} (p_t \cdot \ell_t - \ell_t(k)) &\leq \frac{\log W_{t+1}}{\eta(k)} + \eta(k) \sum_{t=1}^{T} (p_t \cdot \ell_t - \ell_t(k))^2 \text{ for } \eta(k) > 0 \\ \iff \sum_{t=1}^{T} (p_t \cdot \ell_t - \ell_t(k)) &\leq \frac{\log K}{\eta(k)} + \eta(k) \sum_{t=1}^{T} (p_t \cdot \ell_t - \ell_t(k))^2 \end{aligned}$$

Defining the regret $R_T(k) = \sum_{t=1}^{T} (p_t \cdot \ell_t - \ell_t(k))$, we finally have for :

$$R_T(k) \leq 2\sqrt{(\log K) \sum_{t=1}^{T} (p_t \cdot \ell_t - \ell_t(k))^2} \quad \left(\text{for } \eta(k) = \sqrt{\frac{\log K}{\sum_{t=1}^{T} (p_t \cdot \ell_t - \ell_t(k))^2}} \right)$$

12. (a) for all $t \geq 1$, and all $k \in \mathcal{X}$

$$\begin{aligned}
\tilde{p}_t \cdot \tilde{\ell}_t &= \sum_{k=1}^{K} \tilde{p}_t(k)\tilde{\ell}_t(k) = \sum_{k=1}^{K} \tilde{p}_t(k) \left(\mathbb{1}_{k\in\mathcal{A}_t}\ell_t(k) + (1-\mathbb{1}_{k\in\mathcal{A}_t})p_t \cdot \ell_t\right) \\
&= \sum_{k=1}^{K} \tilde{p}_t(k)\ell_t(k)\mathbb{1}_{k\in\mathcal{A}_t} + p_t \cdot \ell_t - p_t \cdot \ell_t \sum_{k=1}^{K} \tilde{p}_t(k)\mathbb{1}_{k\in\mathcal{A}_t} \\
&= \underbrace{\sum_{k=1}^{K} \ell_t(k)p_t(k)}_{=p_t\cdot\ell_t} \sum_{j=1}^{K} \tilde{p}_t(j)\mathbf{1}_{j\in A_t} + p_t \cdot \ell_t - p_t \cdot \ell_t \underbrace{\sum_{k=1}^{K} p_t(k)}_{=1} \sum_{j=1}^{K} \tilde{p}_t(j)\mathbf{1}_{j\in A_t} \qquad \text{as } p_t(k) = \frac{\tilde{p}_t(k)\mathbf{1}_{k\in A_t}}{\sum_{j=1}^{K}\tilde{p}_t(j)\mathbf{1}_{j\in A_t}} \\
&= p_t \cdot \ell_t
\end{aligned}$$

Then, using this equality, we get :

$$\begin{aligned}
\tilde{p}_t \cdot \tilde{\ell}_t - \tilde{\ell}_t(k) &= \begin{cases} p_t \cdot \ell_t - \ell_t(k) & \text{if } k \in A_t \\ p_t \cdot \ell_t - p_t \cdot \ell_t & \text{if } k \notin A_t \end{cases} \\
&= (p_t \cdot \ell_t - \ell_t(k))\mathbb{1}_{k\in\mathcal{A}_t}
\end{aligned}$$

(b)

$$\begin{aligned}
R_T(k) &= \sum_{t=1}^{T} (p_t \cdot \ell_t - \ell_t(k))\mathbb{1}_{k\in\mathcal{A}_t} \\
&\leq 2\sqrt{(\log K) \sum_{t=1}^{T} \underbrace{(p_t \cdot \ell_t - \ell_t(k))^2}_{\leq 1} (\mathbb{1}_{k\in\mathcal{A}_t})^2} \\
&\leq 2\sqrt{(\log K)T_k} \qquad \text{with } T_k = \sum_{t=1}^{T} \mathbb{1}_{k\in\mathcal{A}_t}
\end{aligned}$$

Part 3. Experiments – predict votes of survey

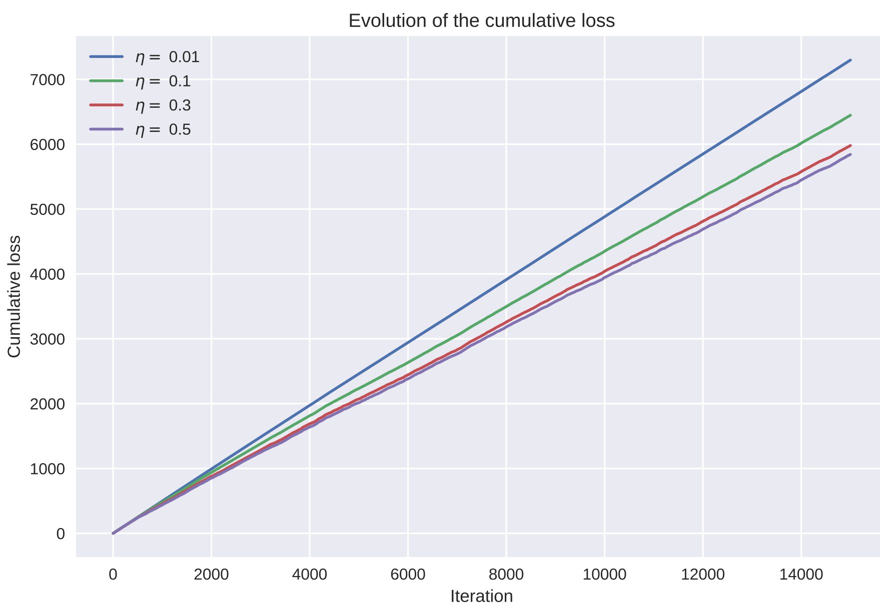
13. We can first say that this loss is convex. Moreover, when we predict the wrong label, we have $y \neq y_t$ and we can check that the loss is equal to 1. When we get a good prediction, the $y = y_t$ and then we can check that the loss is equal to 1.

14. In this question, we implemented EWA and OGD with parameter $\eta > 0$.

15. (a) **EWA algorithm**

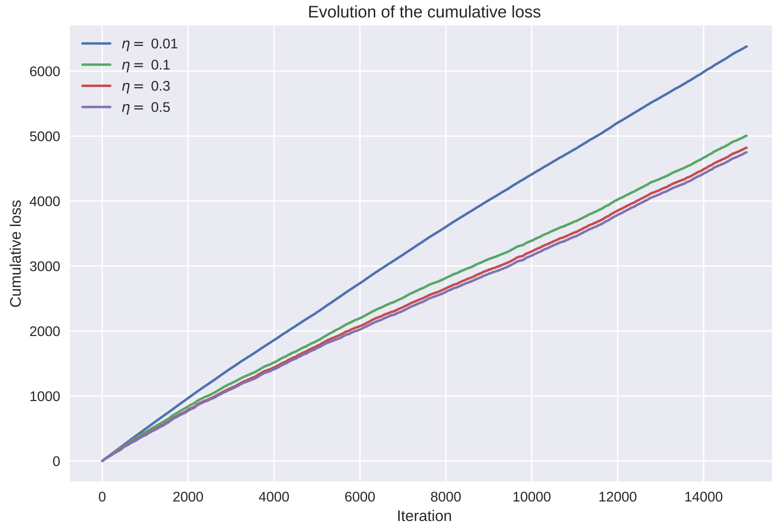
    ideas dataset

    For this dataset, we got a score of 62 % with the EWA algorithm We first plotted for different η the evolution of the cumulative loss of the algorithm. We see here that for EWA, the cumulative loss is smaller when η is high. Thus, we will choose $\eta = 0.5$ for next questions



    politicians dataset

    For this dataset, we got a score of 68.6 % with the EWA algorithm We plotted for different η the evolution of the cumulative loss of the algorithm. We see here that for EWA, the cumulative loss is smaller when η is high. Thus, we will choose $\eta = 0.5$ for next questions



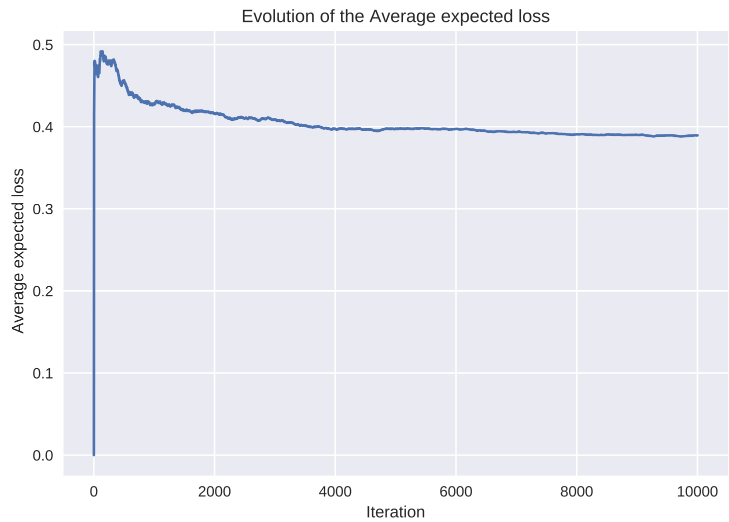
OGD algorithm

For the OGD algorithm, I computed the KL distance for the projection but I didn't got good result (50% accuracy). OGD is supposed to be an improvement of EWA, thus I did a mistake somewhere.

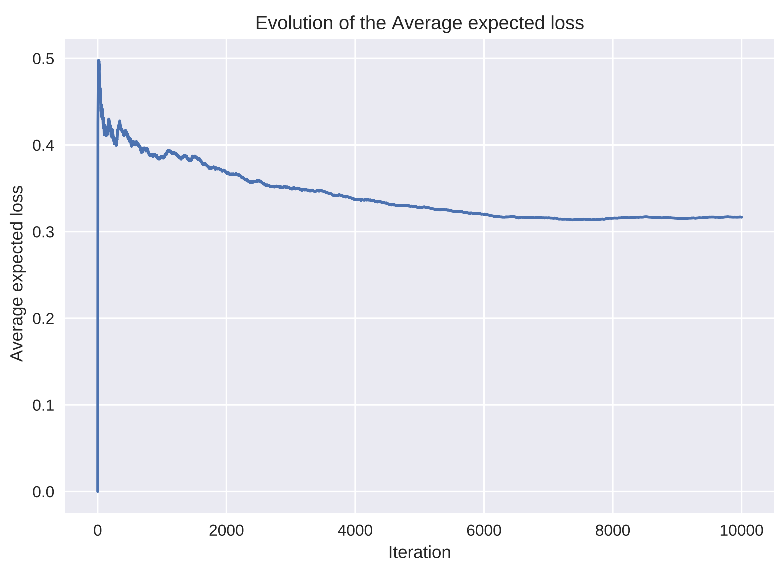
(b) **EWA algorithm**

For both dataset, we see that EWA beat random prediction as we have an average expected loss which is smaller than 0.5

ideas dataset



politicians dataset



(c) **EWA algorithm**

ideas dataset

politicians dataset

Annex - Code

```
[1]: import numpy as np
import matplotlib.pyplot as plt
import time
import pandas as pd
import csv

import seaborn as sns
plt.style.use('seaborn')

sns.set(font_scale=1.3)
```

1 Part 1. Link between online learning and game theory

2. Implementation of EWA.

```
[2]: # (a)

def rand_weighted(p):

  """
  input: probability vector p  _M
  return: X  [M] with P(X = i) = p_i
  """

  random = np.random.rand()

  i = 0
  sum_prob = p[0]

  while i<len(p)-1 and random>=sum_prob:

    i += 1
    sum_prob += p[i]

  return i
```

```
[3]: # (b)

def EWA_update(p, l, eta):
    """
    input: vector p_t  _M, loss vector l_t  [1, 1]^M and \eta
    return: updated vector p_t+1  _M following EWA algorithm
    """

    p_new = p*np.exp(-eta*l)

    renorm = np.sum(p*np.exp(-eta*l))
    p_new /= renorm

    return p_new
```

3. Simulation against a fixed adverary.

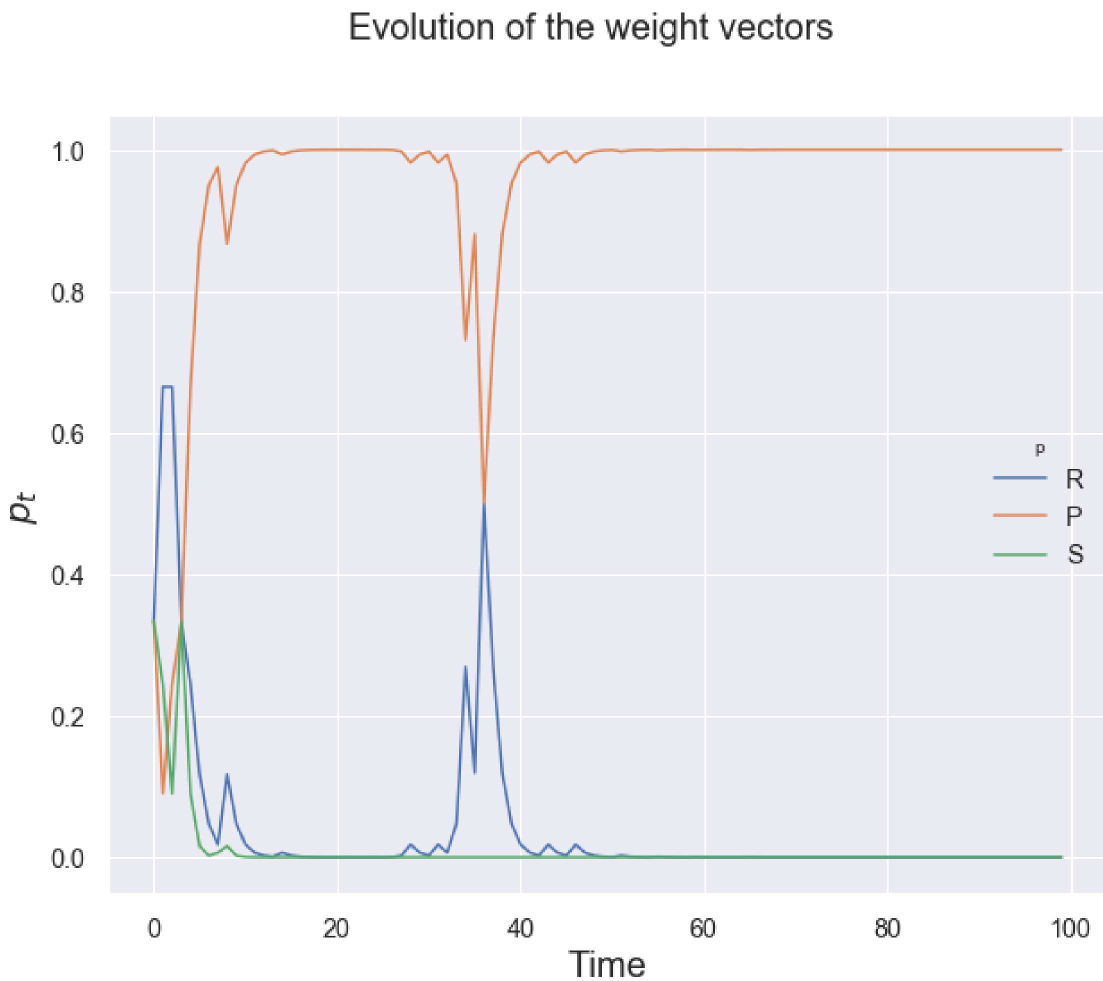
```
[4]: # (a)

def EWA_fixed_adv(p0, T, eta, q, L):
    """
    input: vector p_0  _M, T, eta, vector q  _N and the loss matrix L
    return: all vectors p_t  _M, all losses l_t and the cumulative regret R_T
    """

    p = np.copy(p0)

    p_hist = np.zeros((T,len(p0)))
    l_hist = np.zeros(T)

    l_hist_all_i = np.zeros((len(p0),T))
    cum_regret = np.zeros(T)

    p0 = p

    for t in range(T):

        p_hist[t,:] = p

        i = rand_weighted(p)
        j = rand_weighted(q)

        l_hist[t] = L[i,j]

        l_hist_all_i[:,t] = L[:,j]
        cum_regret[t] = np.sum(l_hist) - np.amin(np.sum(l_hist_all_i, axis=1), axis=0)
```

```
    p = EWA_update(p, np.reshape(L[:,j],(len(p0))), eta)

  return p_hist, l_hist, cum_regret
```

[5]:
```
L_RPS = np.array([[0, 1, -1], [-1, 0, 1], [1, -1, 0]])

T = 100
eta = 1

p0 = (1/3)*np.ones(3)
q = np.array([1/2, 1/4, 1/4])

p_hist, l_hist, cum_regret = EWA_fixed_adv(p0, T, eta, q, L_RPS)
```

[6]:
```
# (b)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the weight vectors', size = 20)
plt.plot(p_hist[:,0], label = 'R')
plt.plot(p_hist[:,1], label = 'P')
plt.plot(p_hist[:,2], label = 'S')
plt.xlabel('Time', size = 20)
plt.ylabel('$p_t$', size = 20)
plt.legend(title='p')
```

[6]:
```
<matplotlib.legend.Legend at 0x261bd4e0c48>
```

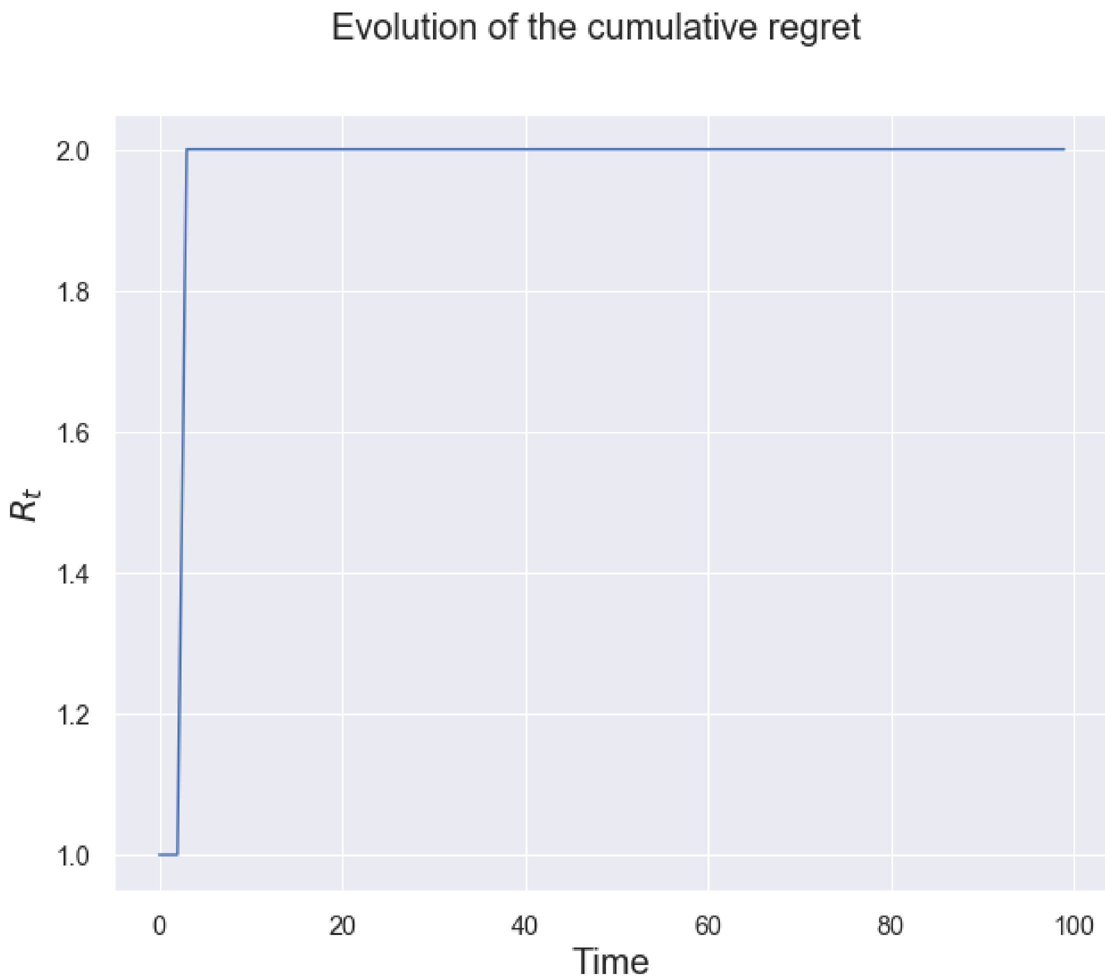



```
[7]: l_avg = np.zeros_like(l_hist)

     for t in range(T):

       l_avg[t] = np.sum(l_hist[:t+1])/(t+1)

     plt.figure(figsize=(10, 8))
     plt.suptitle('Evolution of the average loss', size = 20)
     plt.plot(l_avg)
     plt.xlabel('Time', size = 20)
     plt.ylabel('$\hat{\mathcal{l}}_t$', size = 20)
```

```
[7]: Text(0, 0.5, '$\\hat{\\mathcal{l}}_t$')
```

[8]:
```
# (d)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the cumulative regret', size = 20)
plt.plot(cum_regret)
plt.xlabel('Time', size = 20)
plt.ylabel('$R_t$', size = 20)
```

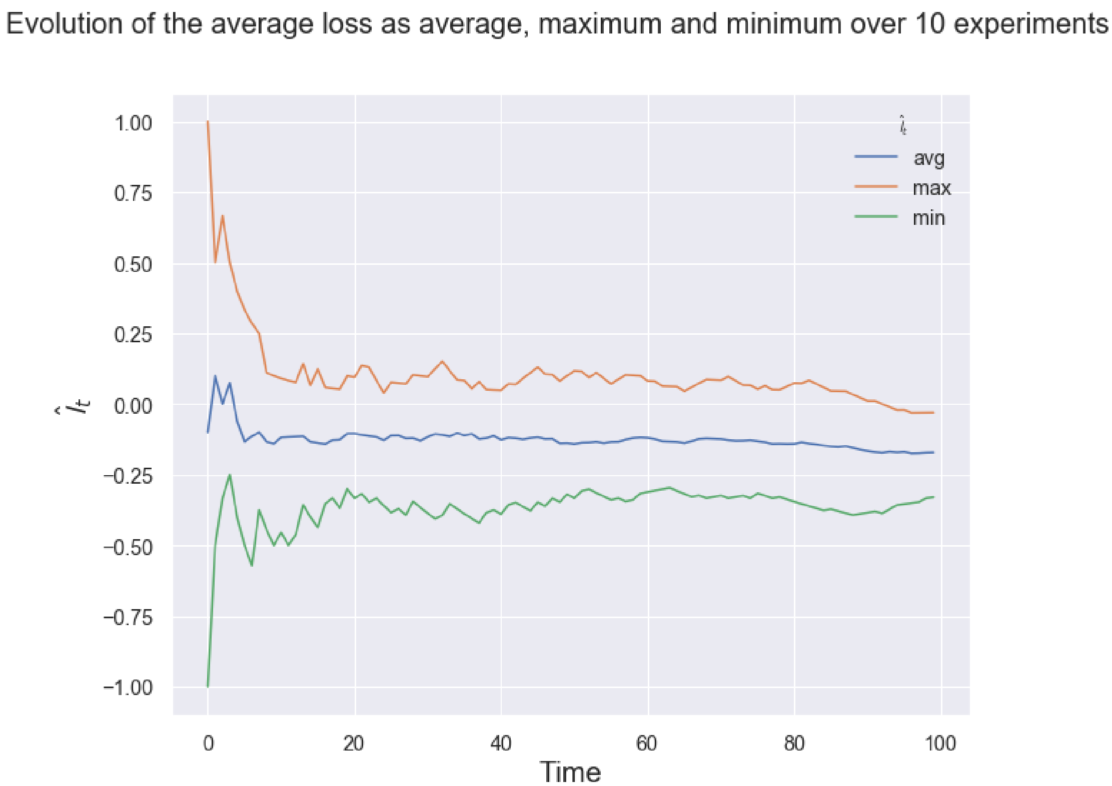
[8]: Text(0, 0.5, 'R_t')



[9]:
```
# (e)

n = 10

l_hist_n = np.zeros((n,T))

for i in range(n):

  _, l_hist_n[i,:], _ = EWA_fixed_adv(p0, T, eta, q, L_RPS)

l_avg_n = np.zeros_like(l_hist_n)

for t in range(T):

  l_avg_n[:,t] = np.sum(l_hist_n[:,:t+1], axis=1)/(t+1)
```

```python
l_avg_n_avg = np.sum(l_avg_n, axis=0)/n

l_avg_n_max = np.amax(l_avg_n, axis=0)

l_avg_n_min = np.amin(l_avg_n, axis=0)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the average loss as average, maximum and minimum over␣
 ↪' + str(n) + ' experiments', size = 20)
plt.plot(l_avg_n_avg, label = 'avg')
plt.plot(l_avg_n_max, label = 'max')
plt.plot(l_avg_n_min, label = 'min')
plt.xlabel('Time', size = 20)
plt.ylabel('$\hat{\mathcal{l}}_t$', size = 20)
plt.legend(title='$\hat{\mathcal{l}}_t$')
```

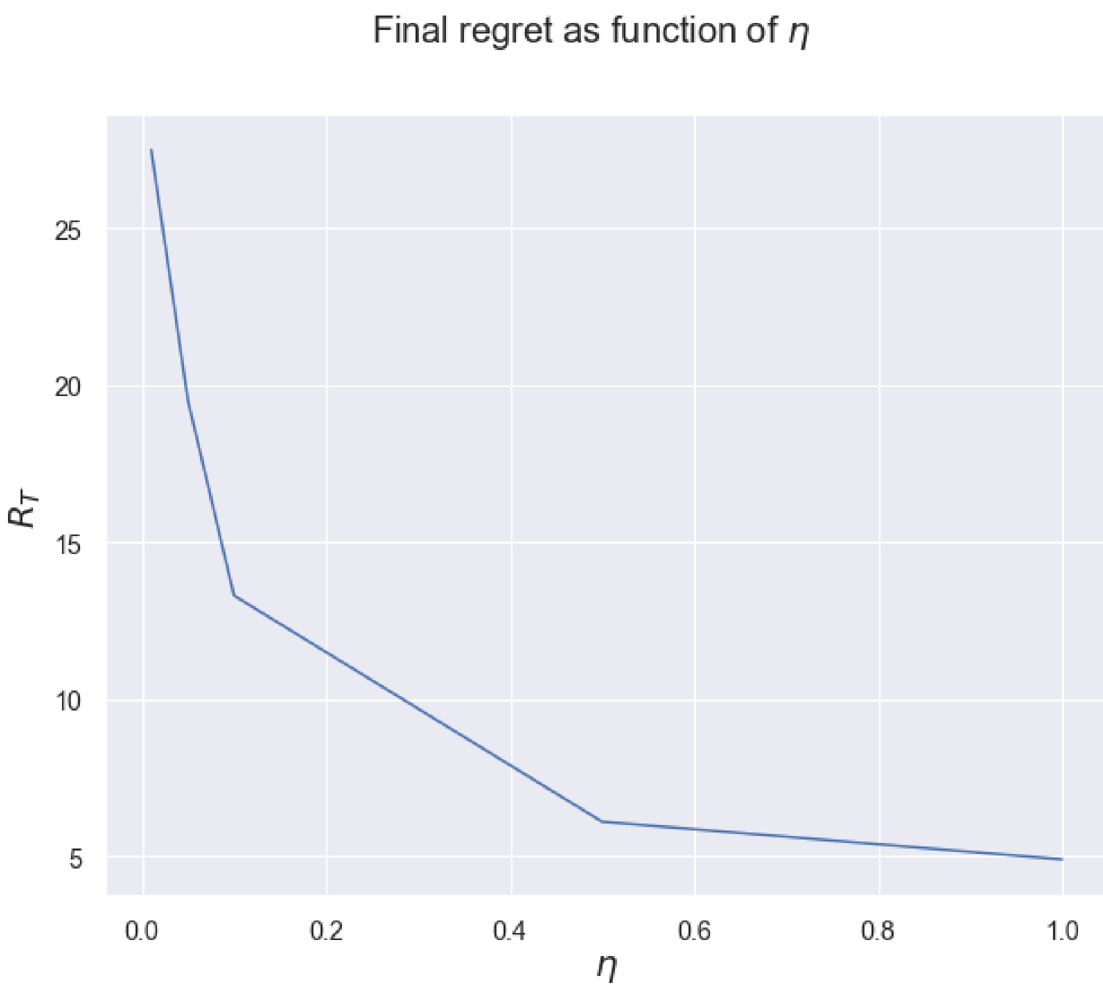
[9]: <matplotlib.legend.Legend at 0x261bd633a88>



[10]:
```python
# (f)

etas = [0.01, 0.05, 0.1, 0.5, 1]
```

```
final_regret_eta = np.zeros((n,len(etas)))

for e in range(len(etas)):

  for i in range(n):

    final_regret_eta[i,e] = EWA_fixed_adv(p0, T, etas[e], q, L_RPS)[2][-1]

final_regret_eta_avg_n = np.sum(final_regret_eta, axis=0)/n

plt.figure(figsize=(10, 8))
plt.suptitle('Final regret as function of $\eta$', size = 20)
plt.plot(etas, final_regret_eta_avg_n)
plt.xlabel('$\eta$', size = 20)
plt.ylabel('$R_T$', size = 20)
```

[10]: Text(0, 0.5, 'R_T')



4. Simulation against an adaptive adversary

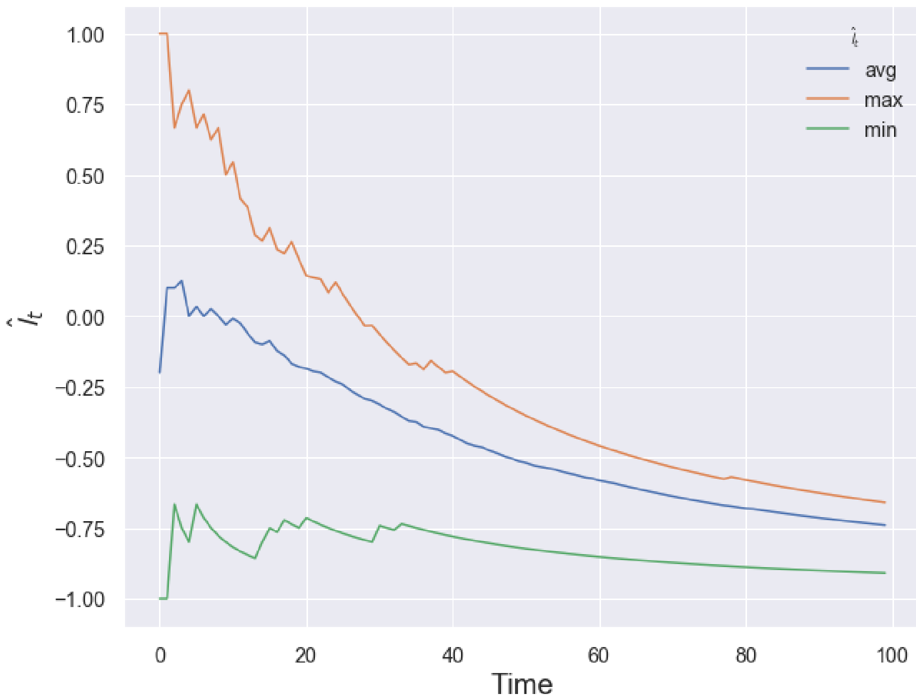
[11]:
```python
# (a)

def EWA_EWA_adv(p0, T, eta, q0, eta_adv, L):
  """
  input: vector p_0  _M, T, eta, vector q_0  _N, eta_adv and loss matrix L
  return: all vectors p_t  _M, all losses l_t and the cumulative regret R_T
  """

  p = np.copy(p0)
  q = np.copy(q0)

  p_hist = np.zeros((T,len(p0)))

  l_hist = np.zeros(T)

  l_hist_all_i = np.zeros((len(p0),T))
  cum_regret = np.zeros(T)

  p0 = p
  q0 = q

  for t in range(T):

    p_hist[t,:] = p

    i = rand_weighted(p)
    j = rand_weighted(q)

    l_hist[t] = L[i,j]

    cum_regret[t] = np.sum(l_hist) - np.amin(np.sum(l_hist_all_i, axis=1), axis=0)

    p = EWA_update(p, np.reshape(L[:,j],(len(p0))), eta)
    q = EWA_update(q, np.reshape(L[i,:],(len(p0))), eta_adv)

  return p_hist, l_hist, cum_regret
```

[12]:
```python
eta_adv = 0.05

q0 = (1/3)*np.ones(3)

l_hist_adv_n = np.zeros((n,T))
```

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

  _, l_hist_adv_n[i,:], _ = EWA_EWA_adv(p0, T, eta, q0, eta_adv, L_RPS)

l_adv_avg_n = np.zeros_like(l_hist_n)

for t in range(T):

  l_adv_avg_n[:,t] = np.sum(l_hist_adv_n[:,:t+1], axis=1)/(t+1)

l_adv_avg_n_avg = np.sum(l_adv_avg_n, axis=0)/n

l_adv_avg_n_max = np.amax(l_adv_avg_n, axis=0)

l_adv_avg_n_min = np.amin(l_adv_avg_n, axis=0)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the average loss as average, maximum and minimum over␣
 ↪' + str(n) + ' experiments', size = 20)
plt.plot(l_adv_avg_n_avg, label = 'avg')
plt.plot(l_adv_avg_n_max, label = 'max')
plt.plot(l_adv_avg_n_min, label = 'min')
plt.xlabel('Time', size = 20)
plt.ylabel('$\hat{\mathcal{l}}_t$', size = 20)
plt.legend(title='$\hat{\mathcal{l}}_t$')
```

[12]: <matplotlib.legend.Legend at 0x261bd697548>

Evolution of the average loss as average, maximum and minimum over 10 experiments

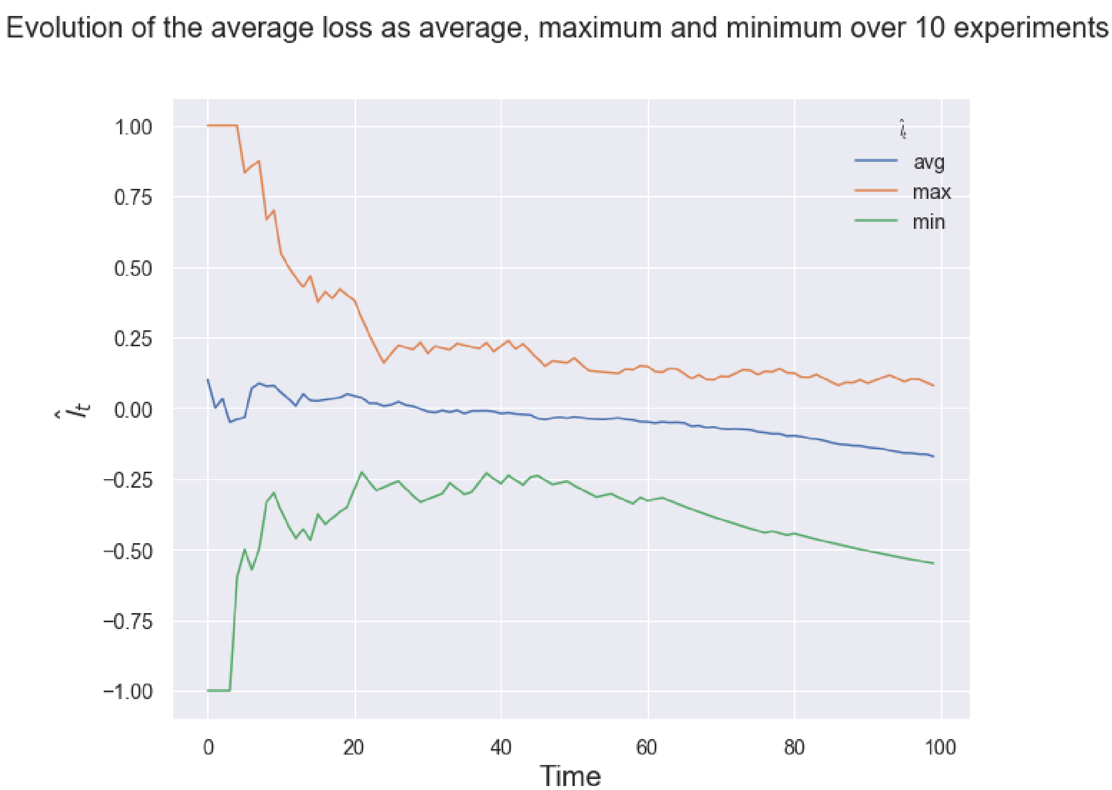


[13]:

```
eta_adv = 0.05

q0 = (1/3)*np.ones(3)

l_hist_adv_n = np.zeros((n,T))

for i in range(n):

  _, l_hist_adv_n[i,:], _ = EWA_EWA_adv(p0, T, eta_adv, q0, eta_adv, L_RPS)

l_adv_avg_n = np.zeros_like(l_hist_n)

for t in range(T):

  l_adv_avg_n[:,t] = np.sum(l_hist_adv_n[:,:t+1], axis=1)/(t+1)

l_adv_avg_n_avg = np.sum(l_adv_avg_n, axis=0)/n

l_adv_avg_n_max = np.amax(l_adv_avg_n, axis=0)

l_adv_avg_n_min = np.amin(l_adv_avg_n, axis=0)
```

```
plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the average loss as average, maximum and minimum over␣
↪' + str(n) + ' experiments', size = 20)
plt.plot(l_adv_avg_n_avg, label = 'avg')
plt.plot(l_adv_avg_n_max, label = 'max')
plt.plot(l_adv_avg_n_min, label = 'min')
plt.xlabel('Time', size = 20)
plt.ylabel('$\hat{\mathcal{l}}_t$', size = 20)
plt.legend(title='$\hat{\mathcal{l}}_t$')
```

[13]: <matplotlib.legend.Legend at 0x261be50fd48>

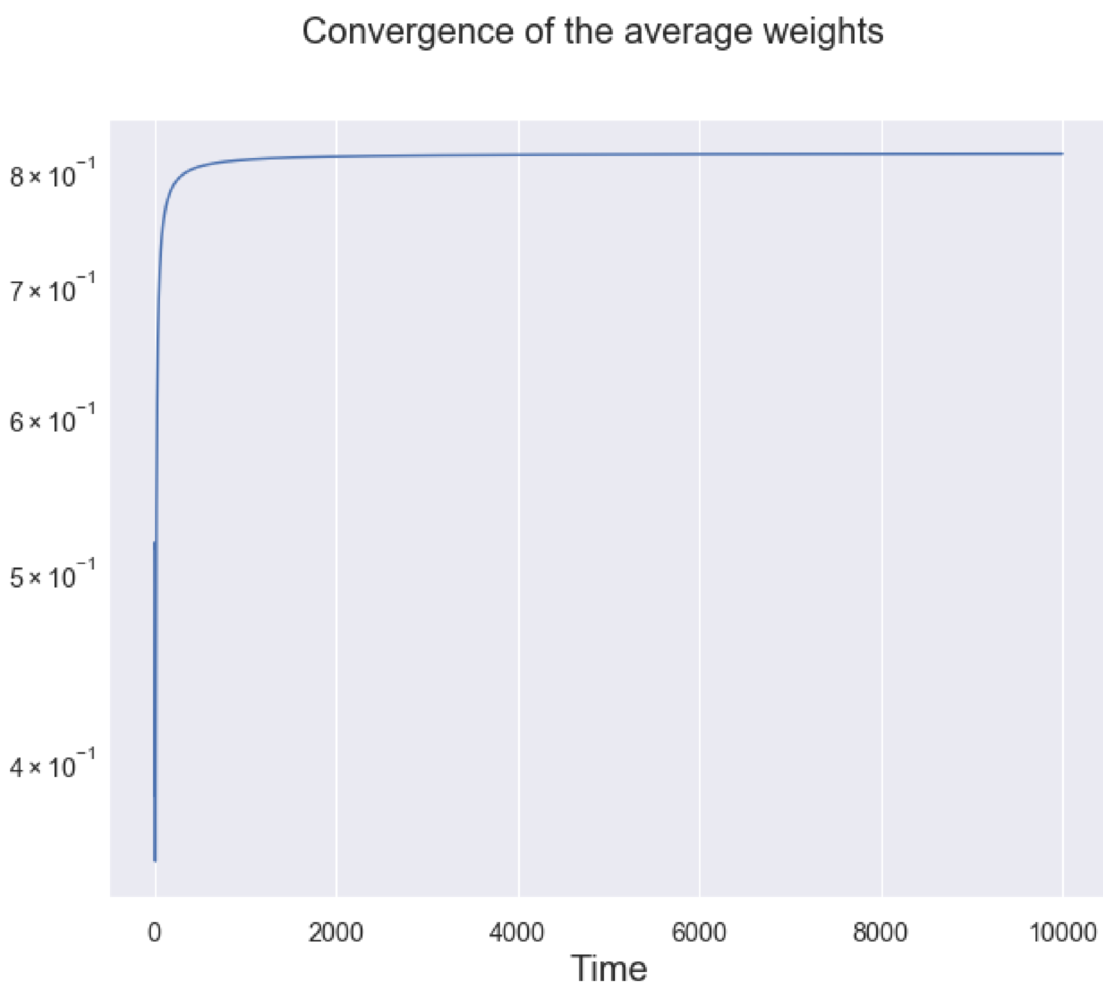


[14]:
```
# (b)

p_hist_large, _, _ = EWA_EWA_adv(np.random.rand(3), 10000, eta, np.random.
↪rand(3), eta_adv, L_RPS)

p_avg = np.zeros_like(p_hist_large)

for t in range(10000):

  p_avg[t,:] = np.sum(p_hist_large[:t+1,:], axis=0)/(t+1)
```

```
p_conv = np.linalg.norm(p_avg - (1/3) * np.ones_like(p_avg), axis=1)

print(p_conv.shape)

plt.figure(figsize=(10, 8))
plt.suptitle('Convergence of the average weights', size = 20)
plt.plot(p_conv)
plt.semilogy()
plt.xlabel('Time', size = 20)
```

```
(10000,)
```

[14]: Text(0.5, 0, 'Time')



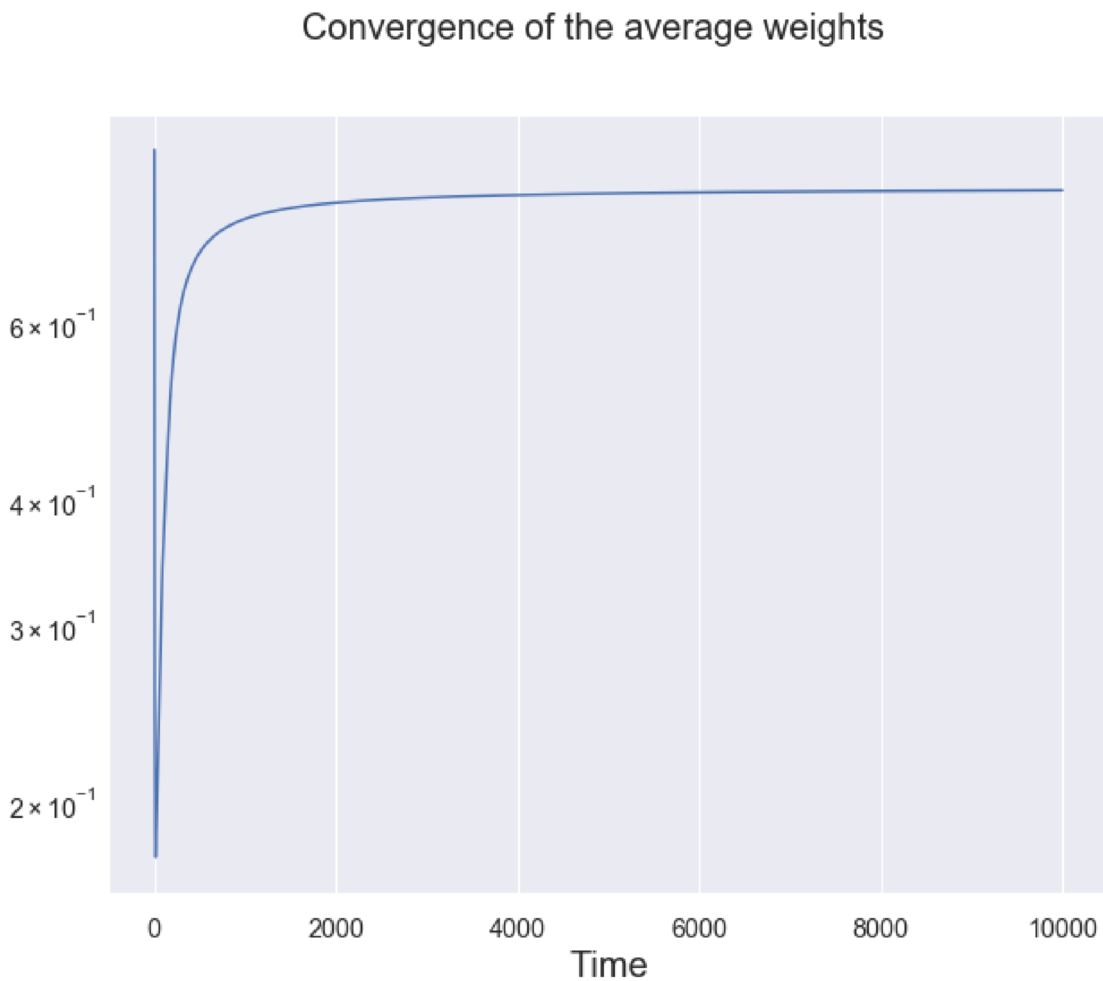
[15]:

```
# (b)

p_hist_large, _, _ = EWA_EWA_adv(np.random.rand(3), 10000, eta_adv, np.random.
↪rand(3), eta_adv, L_RPS)

p_avg = np.zeros_like(p_hist_large)

for t in range(10000):

  p_avg[t,:] = np.sum(p_hist_large[:t+1,:], axis=0)/(t+1)

p_conv = np.linalg.norm(p_avg - (1/3) * np.ones_like(p_avg), axis=1)

print(p_conv.shape)

plt.figure(figsize=(10, 8))
plt.suptitle('Convergence of the average weights', size = 20)
plt.plot(p_conv)
plt.semilogy()
plt.xlabel('Time', size = 20)
```

```
(10000,)
```

[15]:

```
Text(0.5, 0, 'Time')
```



5. Implementation of EXP3

[16]:
```
# (a)

def estimated_loss(i, l, p):
  """
  input: action i_t  [M] played at round t  1, the loss L(i_t, j_t) and vector␣
 ↪p_t  M
  return: vector of estimated loss \hat{l}_t  R_+^M used by EXP3.
  """

  est_l = np.zeros(len(p))

  est_l[i] = l/p[i]

  return est_l
```

[17]:
```python
# (b)

def EXP3_update(p, i, l, eta):
    """
    input: vector p_t  M, action i_t  [M] played at round t  1, the loss L(i_t,␣
    ↪j_t) and eta
    return: vector p_{t+1}  M
    """

    est_l = estimated_loss(i, l, p)

    p_new = np.exp(-eta*est_l)*p
    p_new /= np.sum(p_new)

    return p_new
```

6.

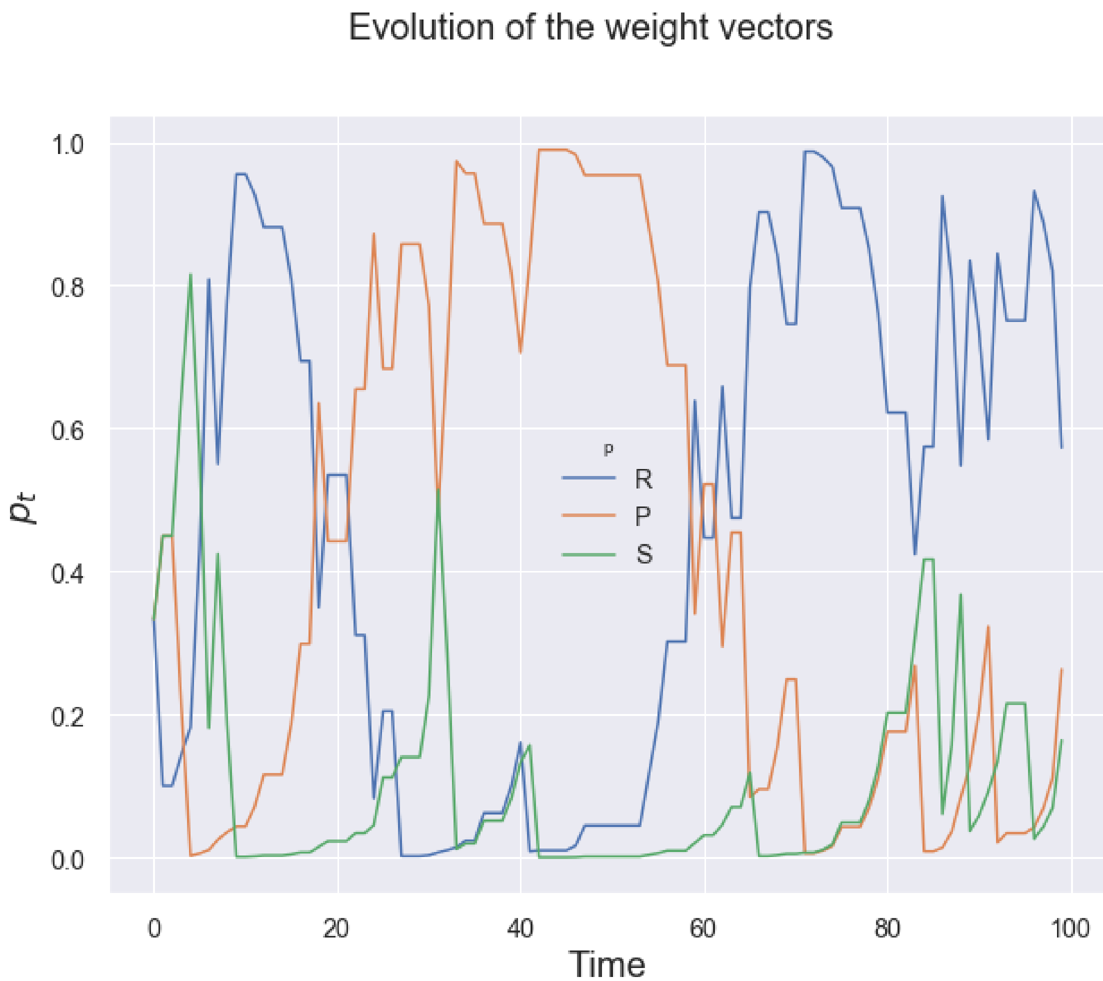
[18]:
```python
# (a)

def EXP3_fixed_adv(p0, T, eta, q, L):
    """
    input: vector p_0  _M, T, eta, vector q  _N and loss matrix L
    return: all vectors p_t  _M, all losses l_t and the cumulative regret R_T
    """

    p = np.copy(p0)

    p_hist = np.zeros((T,len(p0)))
    l_hist = np.zeros(T)

    l_hist_all_i = np.zeros((len(p0),T))
    cum_regret = np.zeros(T)

    p0 = p

    for t in range(T):

        p_hist[t,:] = p

        i = rand_weighted(p)
        j = rand_weighted(q)

        l_hist[t] = L[i,j]

        l_hist_all_i[:,t] = L[:,j]
```

```
        cum_regret[t] = np.sum(l_hist) - np.amin(np.sum(l_hist_all_i, axis=1),␣
↪axis=0)

        p = EXP3_update(p, i, L[i,j], eta)

    return p_hist, l_hist, cum_regret
```

[19]:
```
L_RPS_new = np.array([[1/2, 1, 0], [0, 1/2, 1], [1, 0, 1/2]])

T = 100
eta = 1

p0 = (1/3)*np.ones(3)
q = np.array([1/2, 1/4, 1/4])

p_hist, l_hist, cum_regret = EXP3_fixed_adv(p0, T, eta, q, L_RPS_new)
```

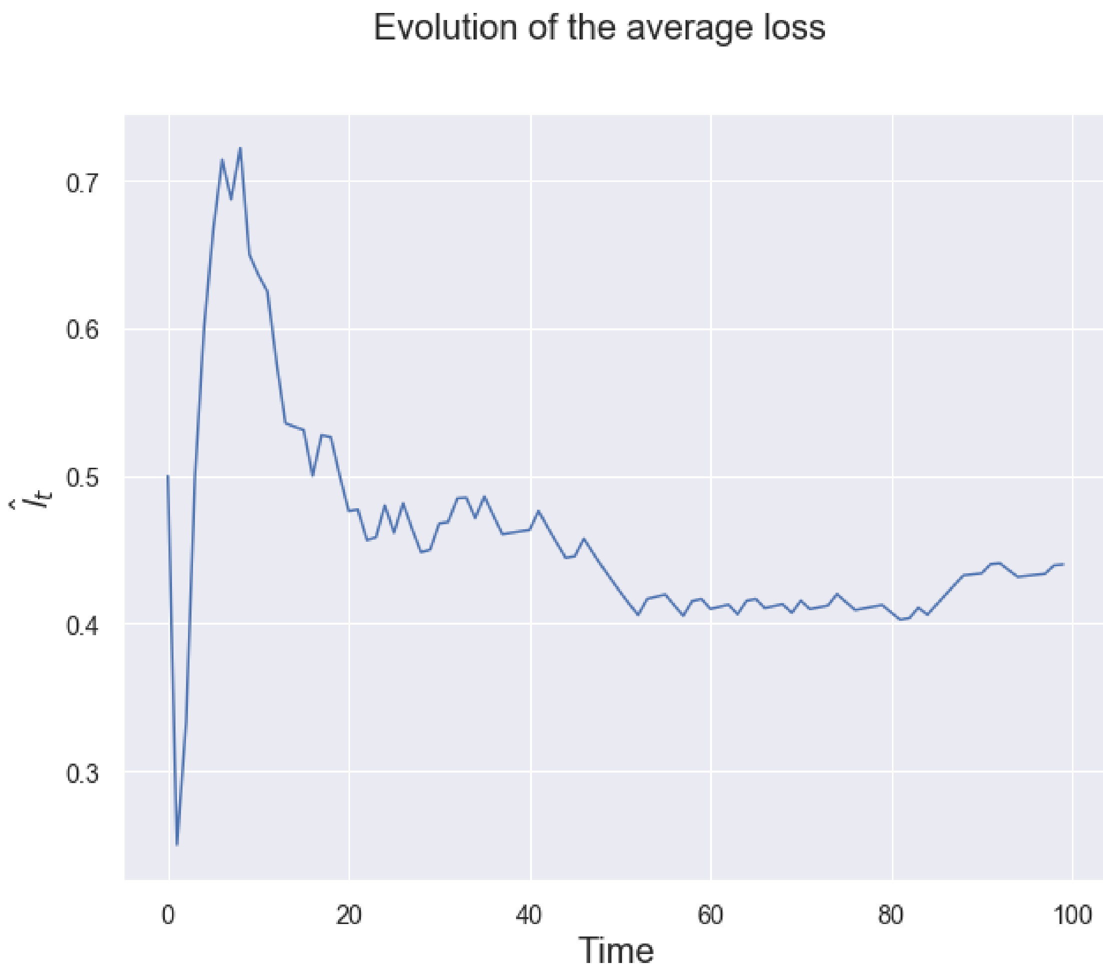
[20]:
```
# (b)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the weight vectors', size = 20)
plt.plot(p_hist[:,0], label = 'R')
plt.plot(p_hist[:,1], label = 'P')
plt.plot(p_hist[:,2], label = 'S')
plt.xlabel('Time', size = 20)
plt.ylabel('$p_t$', size = 20)
plt.legend(title='p')
```

[20]: <matplotlib.legend.Legend at 0x261bda737c8>



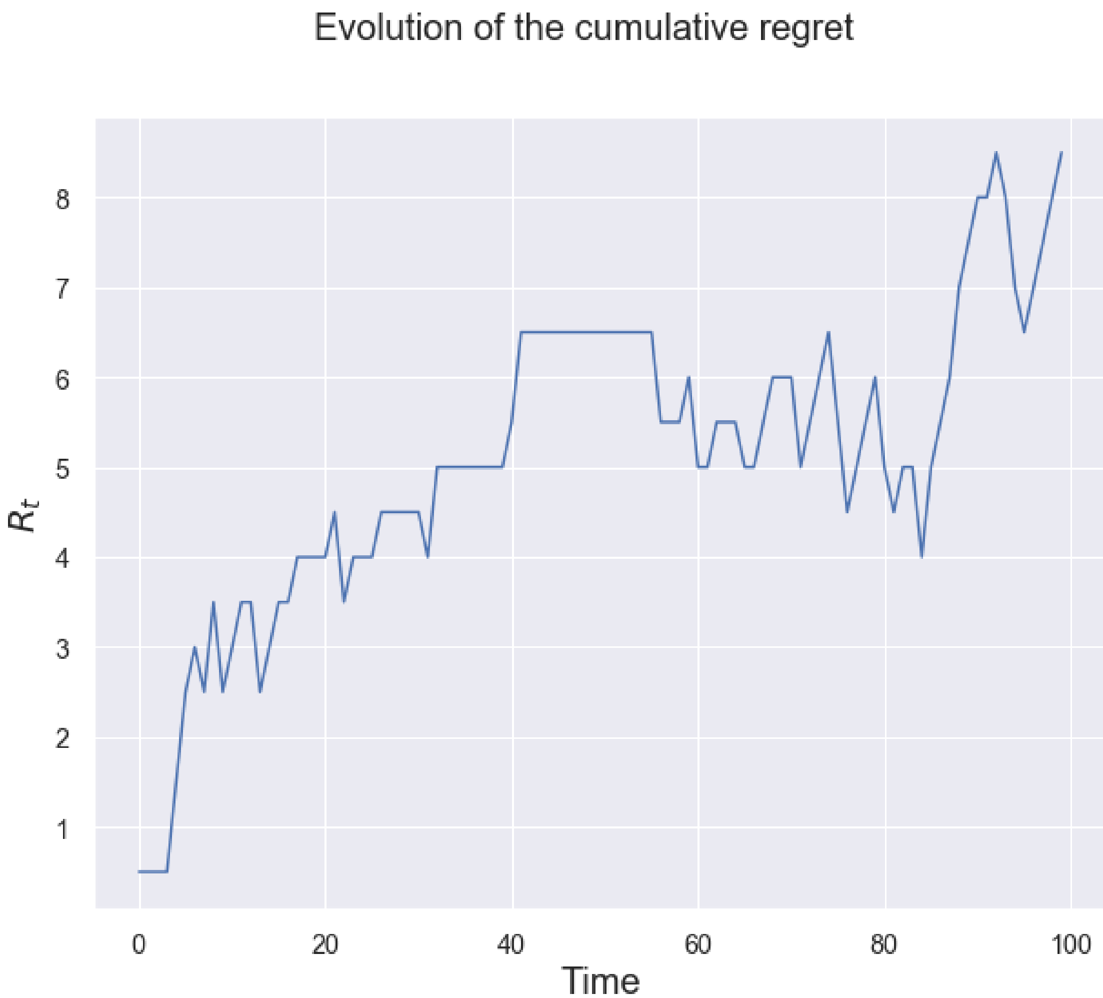
[21]:
```
l_avg = np.zeros_like(l_hist)

for t in range(T):

  l_avg[t] = np.sum(l_hist[:t+1])/(t+1)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the average loss', size = 20)
plt.plot(l_avg)
plt.xlabel('Time', size = 20)
plt.ylabel('$\hat{\mathcal{l}}_t$', size = 20)
```

[21]:
```
Text(0, 0.5, '$\\hat{\\mathcal{l}}_t$')
```



[22]:
```
# (d)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the cumulative regret', size = 20)
plt.plot(cum_regret)
plt.xlabel('Time', size = 20)
plt.ylabel('$R_t$', size = 20)
```

[22]: Text(0, 0.5, 'R_t')



[23]:
```
# (e)

n = 10

l_hist_n = np.zeros((n,T))

for i in range(n):

  _, l_hist_n[i,:], _ = EXP3_fixed_adv(p0, T, eta, q, L_RPS_new)

l_avg_n = np.zeros_like(l_hist_n)

for t in range(T):

  l_avg_n[:,t] = np.sum(l_hist_n[:,:t+1], axis=1)/(t+1)
```

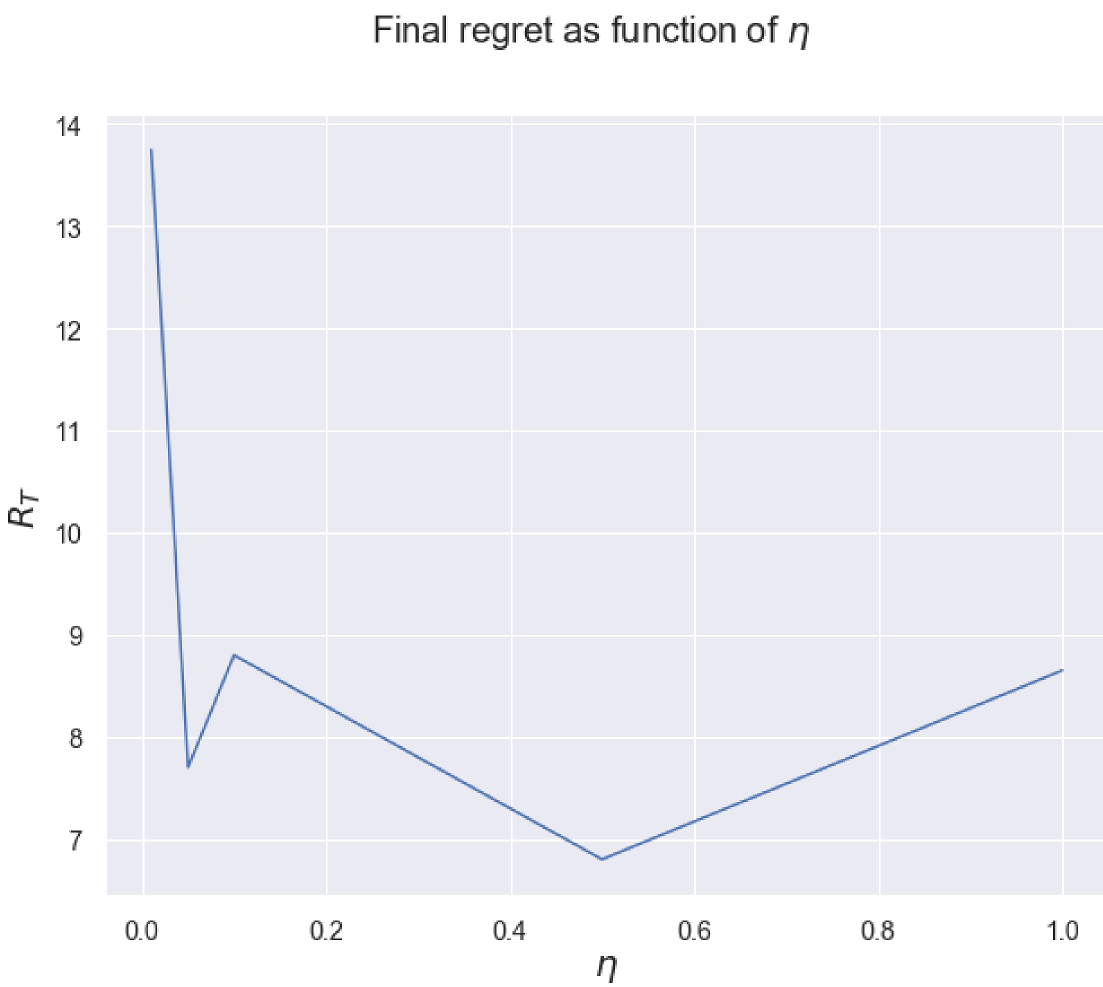
```
l_avg_n_avg = np.sum(l_avg_n, axis=0)/n

l_avg_n_max = np.amax(l_avg_n, axis=0)

l_avg_n_min = np.amin(l_avg_n, axis=0)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the average loss as average, maximum and minimum over␣
 ↪' + str(n) + ' experiments', size = 20)
plt.plot(l_avg_n_avg, label = 'avg')
plt.plot(l_avg_n_max, label = 'max')
plt.plot(l_avg_n_min, label = 'min')
plt.xlabel('Time', size = 20)
plt.ylabel('$\hat{\mathcal{l}}_t$', size = 20)
plt.legend(title='$\hat{\mathcal{l}}_t$')
```

[23]: <matplotlib.legend.Legend at 0x261beebcc08>

[24]:
```
# (f)

etas = [0.01, 0.05, 0.1, 0.5, 1]
```

```python
final_regret_eta = np.zeros((n,len(etas)))

for e in range(len(etas)):

  for i in range(n):

    final_regret_eta[i,e] = EXP3_fixed_adv(p0, T, etas[e], q, L_RPS_new)[2][-1]

final_regret_eta_avg_n = np.sum(final_regret_eta, axis=0)/n

plt.figure(figsize=(10, 8))
plt.suptitle('Final regret as function of $\eta$', size = 20)
plt.plot(etas, final_regret_eta_avg_n)
plt.xlabel('$\eta$', size = 20)
plt.ylabel('$R_T$', size = 20)
```

[24]: Text(0, 0.5, 'R_T')



7.

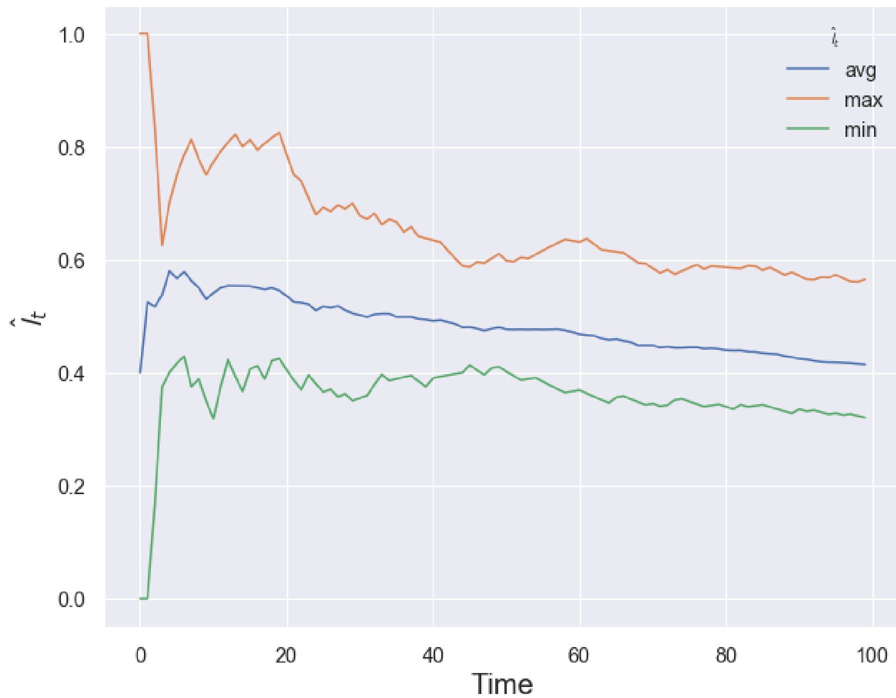
[25]:
```python
# (a)

def EXP3_EWA_adv(p0, T, eta, q0, eta_adv, L):
    """
    input: vector p_0  _M, T, eta, vector q_0  _N, eta_adv and loss matrix L
    return: all vectors p_t  _M, all losses l_t and the cumulative regret R_T
    """

    p = np.copy(p0)
    q = np.copy(q0)

    p_hist = np.zeros((T,len(p0)))

    l_hist = np.zeros(T)

    l_hist_all_i = np.zeros((len(p0),T))
    cum_regret = np.zeros(T)

    p0 = p
    q0 = q

    for t in range(T):

        p_hist[t,:] = p

        i = rand_weighted(p)
        j = rand_weighted(q)

        l_hist[t] = L[i,j]

        l_hist_all_i[:,t] = L[:,j]
        cum_regret[t] = np.sum(l_hist) - np.amin(np.sum(l_hist_all_i, axis=1), axis=0)

        p = EXP3_update(p, i, L[i,j], eta)
        q = EWA_update(q, np.reshape(L[i,:],(len(p0))), eta_adv)

    return p_hist, l_hist, cum_regret
```

[26]:
```python
eta_adv = 0.05

q0 = (1/3)*np.ones(3)
```

```
l_hist_adv_n = np.zeros((n,T))

for i in range(n):

  _, l_hist_adv_n[i,:], _ = EXP3_EWA_adv(p0, T, eta, q0, eta_adv, L_RPS_new)

l_adv_avg_n = np.zeros_like(l_hist_n)

for t in range(T):

  l_adv_avg_n[:,t] = np.sum(l_hist_adv_n[:,:t+1], axis=1)/(t+1)

l_adv_avg_n_avg = np.sum(l_adv_avg_n, axis=0)/n

l_adv_avg_n_max = np.amax(l_adv_avg_n, axis=0)

l_adv_avg_n_min = np.amin(l_adv_avg_n, axis=0)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the average loss as average, maximum and minimum over␣
↪' + str(n) + ' experiments', size = 20)
plt.plot(l_adv_avg_n_avg, label = 'avg')
plt.plot(l_adv_avg_n_max, label = 'max')
plt.plot(l_adv_avg_n_min, label = 'min')
plt.xlabel('Time', size = 20)
plt.ylabel('$\hat{\mathcal{l}}_t$', size = 20)
plt.legend(title='$\hat{\mathcal{l}}_t$')
```

[26]: <matplotlib.legend.Legend at 0x261bef038c8>

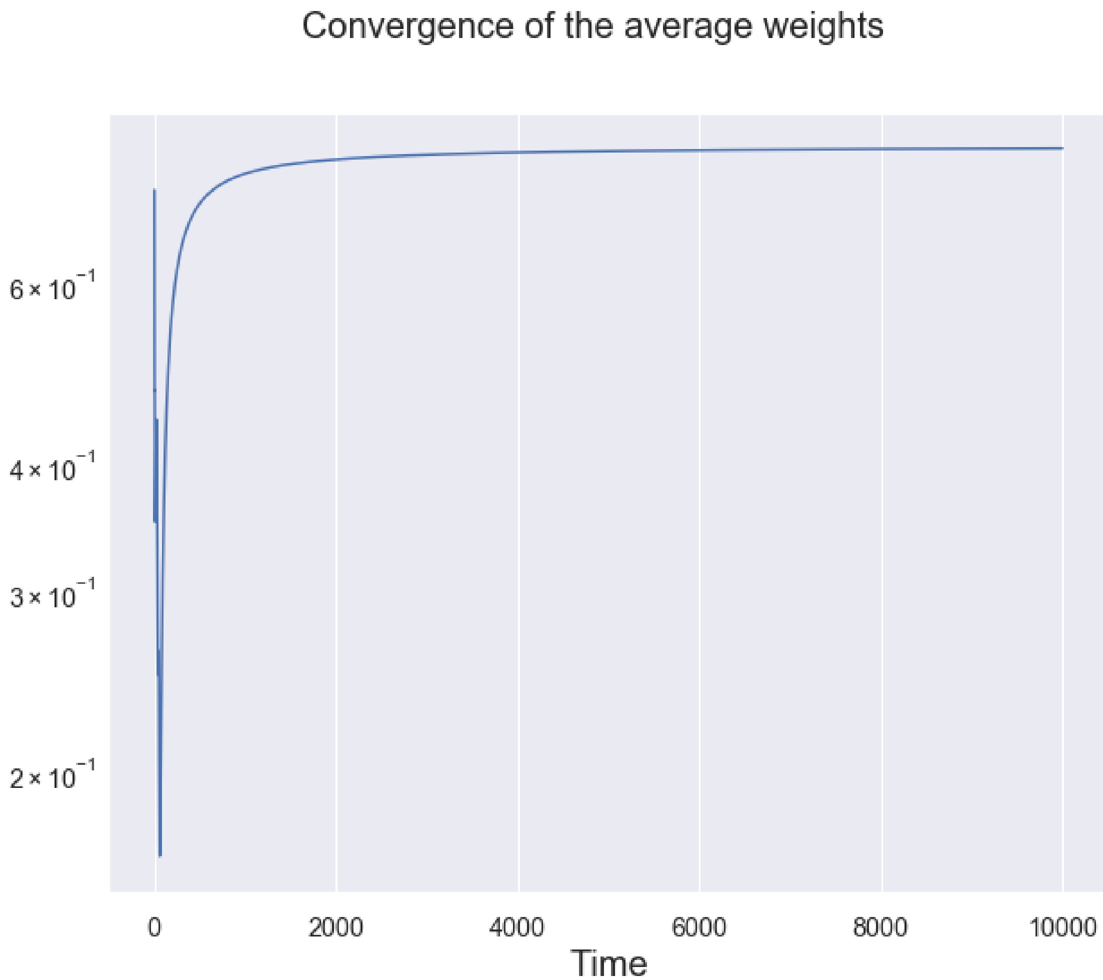
Evolution of the average loss as average, maximum and minimum over 10 experiments



[27]:
```
# (b)

p_hist_large, _, _ = EXP3_EWA_adv(np.random.rand(3), 10000, eta, np.random.
 ↪rand(3), eta_adv, L_RPS_new)

p_avg = np.zeros_like(p_hist_large)

for t in range(10000):

  p_avg[t,:] = np.sum(p_hist_large[:t+1,:], axis=0)/(t+1)

p_conv = np.linalg.norm(p_avg - (1/3) * np.ones_like(p_avg), axis=1)

print(p_conv.shape)

plt.figure(figsize=(10, 8))
plt.suptitle('Convergence of the average weights', size = 20)
plt.plot(p_conv)
plt.semilogy()
plt.xlabel('Time', size = 20)
```

```
(10000,)
```

[27]: Text(0.5, 0, 'Time')



8.

[28]:
```
# (a)

def mu(X, k, m):
  """
  input: matrix X of gains at each time and for each choice, k  [M], m
  return: mu the approx expectation of gain for k  [M] until m
  """

  K = X.shape[1]
  mu = np.sum(X[:int(m*K),k])
  mu /= m
```

```
  return mu

def UCB(N, X, t):
  """
  input: vector N of number of times each choice has been picked,
  matrix X of gains at each time and for each choice and time t
  return: j  [M] the play as argmax of upper bounds
  """

  UCB = np.zeros_like(N)

  for k in range(len(N)):

    if N[k]==0:
      UCB[k] = float("inf")

    else:
      UCB[k] = mu(X, k, N[k]) + np.sqrt((2*np.log(t+1))/N[k])

  j = np.argmax(UCB)

  return j


def EXP3_UCB_adv(p0, T, eta, L):
  """
  input: vector p_0  _M, T, eta and loss matrix L
  return: all vectors p_t  _M, all losses l_t and the cumulative regret R_T
  """

  p = np.copy(p0)

  p_hist = np.zeros((T,len(p0)))

  l_hist = np.zeros(T)

  X_hist = np.zeros((T,len(p0)))
  N = np.zeros(len(p0))

  l_hist_all_i = np.zeros((len(p0),T))
  cum_regret = np.zeros(T)

  p0 = p

  for t in range(T):

    p_hist[t,:] = p
```

```
    i = rand_weighted(p)

    j = UCB(N, X_hist, t)

    N[j] += 1
    X_hist[t,j] = L[i,j]

    l_hist[t] = L[i,j]

    l_hist_all_i[:,t] = L[:,j]
    cum_regret[t] = np.sum(l_hist) - np.amin(np.sum(l_hist_all_i, axis=1),␣
↪axis=0)

    p = EXP3_update(p, i, L[i,j], eta)

  return p_hist, l_hist, cum_regret
```

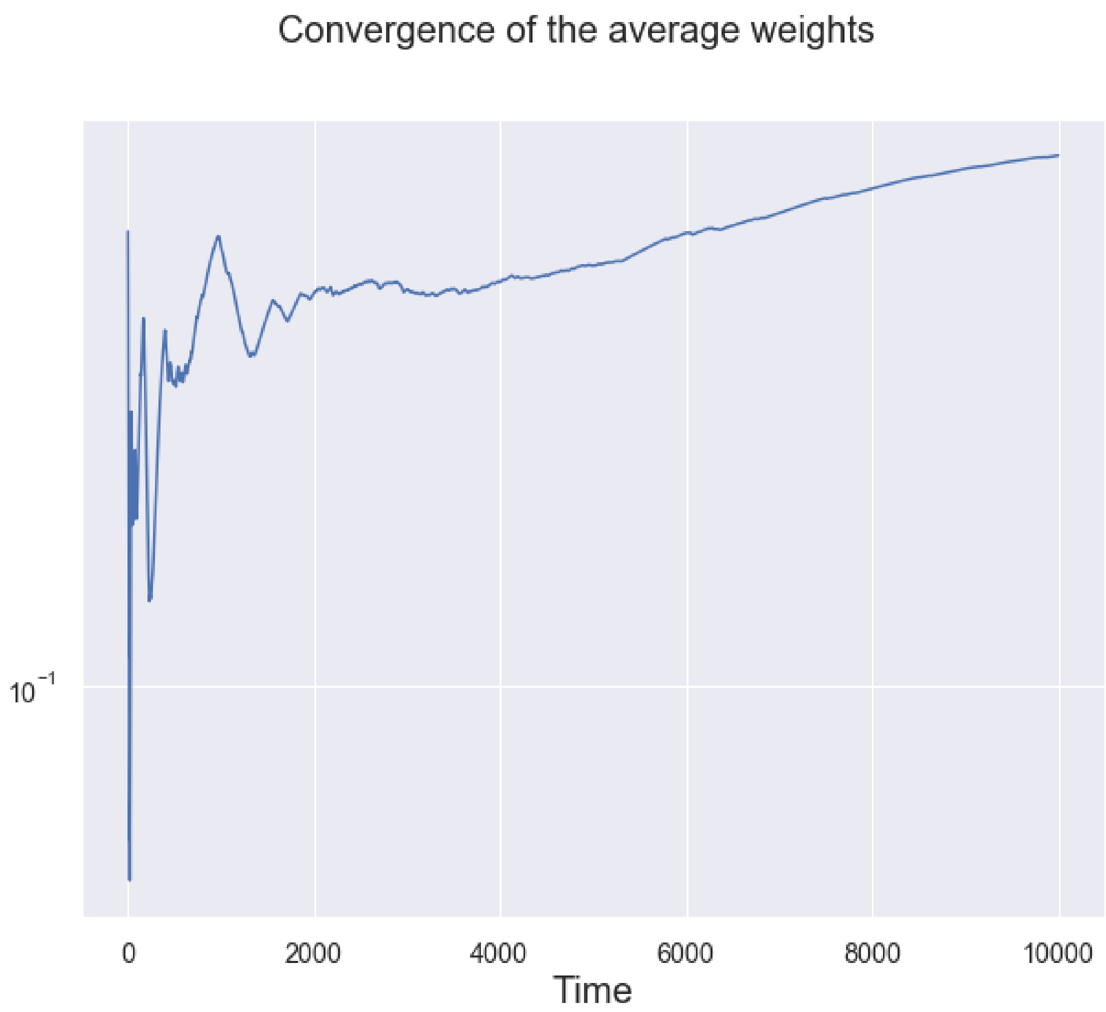
[29]:
```
n=10

l_hist_adv_n = np.zeros((n,T))

for i in range(n):

  _, l_hist_adv_n[i,:], _ = EXP3_UCB_adv(p0, T, eta, L_RPS_new)

l_adv_avg_n = np.zeros((n,T))

for t in range(T):

  l_adv_avg_n[:,t] = np.sum(l_hist_adv_n[:,:t+1], axis=1)/(t+1)

l_adv_avg_n_avg = np.sum(l_adv_avg_n, axis=0)/n

l_adv_avg_n_max = np.amax(l_adv_avg_n, axis=0)

l_adv_avg_n_min = np.amin(l_adv_avg_n, axis=0)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the average loss as average, maximum and minimum over␣
↪' + str(n) + ' experiments', size = 20)
plt.plot(l_adv_avg_n_avg, label = 'avg')
plt.plot(l_adv_avg_n_max, label = 'max')
plt.plot(l_adv_avg_n_min, label = 'min')
plt.xlabel('Time', size = 20)
plt.ylabel('$\hat{\mathcal{l}}_t$', size = 20)
plt.legend(title='$\hat{\mathcal{l}}_t$')
```

[29]: <matplotlib.legend.Legend at 0x261beaee448>

[30]:
```
# (b)

p_hist_large, _, _ = EXP3_UCB_adv(np.random.rand(3), 10000, eta, L_RPS_new)

p_avg = np.zeros_like(p_hist_large)

for t in range(10000):

  p_avg[t,:] = np.sum(p_hist_large[:t+1,:], axis=0)/(t+1)

p_conv = np.linalg.norm(p_avg - (1/3) * np.ones_like(p_avg), axis=1)

print(p_conv.shape)

plt.figure(figsize=(10, 8))
plt.suptitle('Convergence of the average weights', size = 20)
plt.plot(p_conv)
plt.semilogy()
plt.xlabel('Time', size = 20)
```

```
(10000,)
```

[30]: Text(0.5, 0, 'Time')



9.

[31]:
```python
def EXP3_IX_fixed_adv(T, eta, gamma, q, L):
  """
  input: vector T, eta, gamma, vector q  _N and loss matrix L
  return: all vectors p_t  _M, all losses l_t and the cumulative regret R_T
  """

  p_hist = np.zeros((T,len(q)))
  l_hist = np.zeros(T)

  l_est = np.zeros(len(q))
```

```
  l_hist_all_i = np.zeros((len(p0),T))
  cum_regret = np.zeros(T)

  for t in range(T):

    p = np.exp(-eta*l_est)
    p /= np.sum(p)

    p_hist[t,:] = p

    i = rand_weighted(p)
    j = rand_weighted(q)

    l_hist[t] = L[i,j]

    l_hist_all_i[:,t] = L[:,j]
    cum_regret[t] = np.sum(l_hist) - np.amin(np.sum(l_hist_all_i, axis=1),␣
↪axis=0)

    l_est[i] += L[i,j]/(p[i] + gamma)

  return p_hist, l_hist, cum_regret
```

[32]:
```
# (e)

n = 10

l_hist_n = np.zeros((n,T))

for i in range(n):

  _, l_hist_n[i,:], _ = EXP3_IX_fixed_adv(T, eta, eta/2, q, L_RPS_new)

l_avg_n = np.zeros_like(l_hist_n)

for t in range(T):

  l_avg_n[:,t] = np.sum(l_hist_n[:,:t+1], axis=1)/(t+1)

l_avg_n_avg = np.sum(l_avg_n, axis=0)/n

l_avg_n_max = np.amax(l_avg_n, axis=0)

l_avg_n_min = np.amin(l_avg_n, axis=0)

plt.figure(figsize=(10, 8))
```

```
plt.suptitle('Evolution of the average loss as average, maximum and minimum over␣
↪' + str(n) + ' experiments', size = 20)
plt.plot(l_avg_n_avg, label = 'avg')
plt.plot(l_avg_n_max, label = 'max')
plt.plot(l_avg_n_min, label = 'min')
plt.xlabel('Time', size = 20)
plt.ylabel('$\hat{\mathcal{l}}_t$', size = 20)
plt.legend(title='$\hat{\mathcal{l}}_t$')
```

[32]: <matplotlib.legend.Legend at 0x261bf4f5d88>

10. Prisoner's dilemma

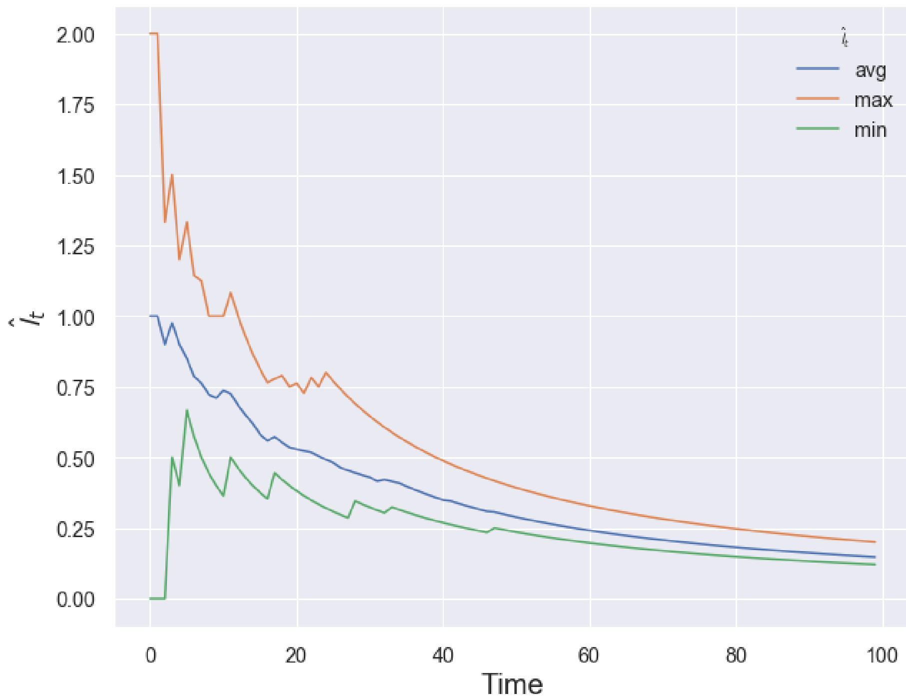
(a) Player plays EWA with $\eta = 1$ and adversary plays EWA with $\eta = 0.05$

[33]:
```
L_pris = np.array([[1, 3], [0, 2]])

T = 100
eta = 1

p0 = (1/2)*np.ones(2)
```

```
eta_adv = 0.05

q0 = (1/2)*np.ones(2)

l_hist_adv_n = np.zeros((n,T))

for i in range(n):

  _, l_hist_adv_n[i,:], _ = EWA_EWA_adv(p0, T, eta, q0, eta_adv, L_pris)

l_adv_avg_n = np.zeros_like(l_hist_n)

for t in range(T):

  l_adv_avg_n[:,t] = np.sum(l_hist_adv_n[:,:t+1], axis=1)/(t+1)

l_adv_avg_n_avg = np.sum(l_adv_avg_n, axis=0)/n

l_adv_avg_n_max = np.amax(l_adv_avg_n, axis=0)

l_adv_avg_n_min = np.amin(l_adv_avg_n, axis=0)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the average loss as average, maximum and minimum over␣
 ↪' + str(n) + ' experiments', size = 20)
plt.plot(l_adv_avg_n_avg, label = 'avg')
plt.plot(l_adv_avg_n_max, label = 'max')
plt.plot(l_adv_avg_n_min, label = 'min')
plt.xlabel('Time', size = 20)
plt.ylabel('$\hat{\mathcal{l}}_t$', size = 20)
plt.legend(title='$\hat{\mathcal{l}}_t$')
```

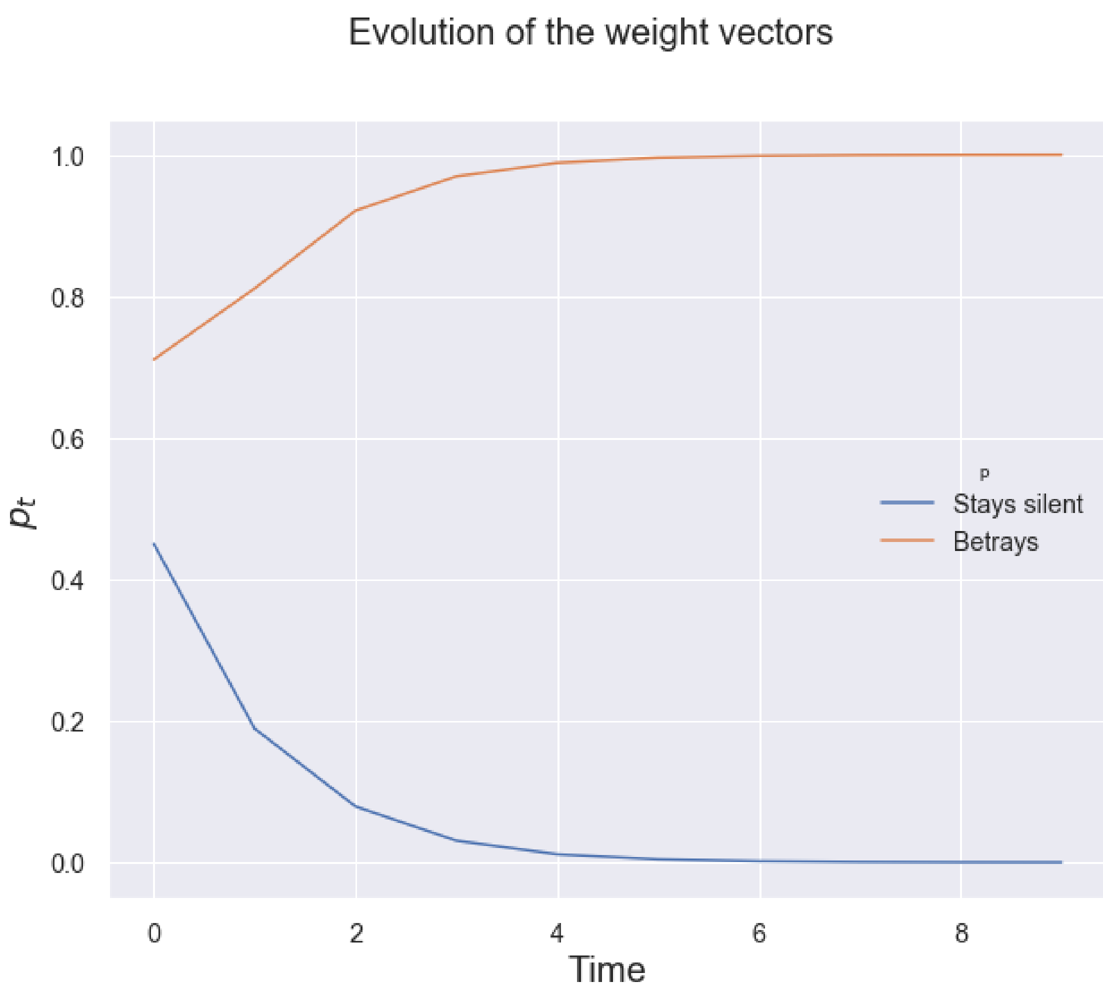
[33]: <matplotlib.legend.Legend at 0x261bf4b7548>

Evolution of the average loss as average, maximum and minimum over 10 experiments



[34]:
```
p_hist_large, _, _ = EWA_EWA_adv(np.random.rand(2), 10, eta, np.random.rand(2),␣
↪eta_adv, L_pris)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the weight vectors', size = 20)
plt.plot(p_hist_large[:,0], label = 'Stays silent')
plt.plot(p_hist_large[:,1], label = 'Betrays')
plt.xlabel('Time', size = 20)
plt.ylabel('$p_t$', size = 20)
plt.legend(title='p')
```

[34]: <matplotlib.legend.Legend at 0x261bf8afa88>

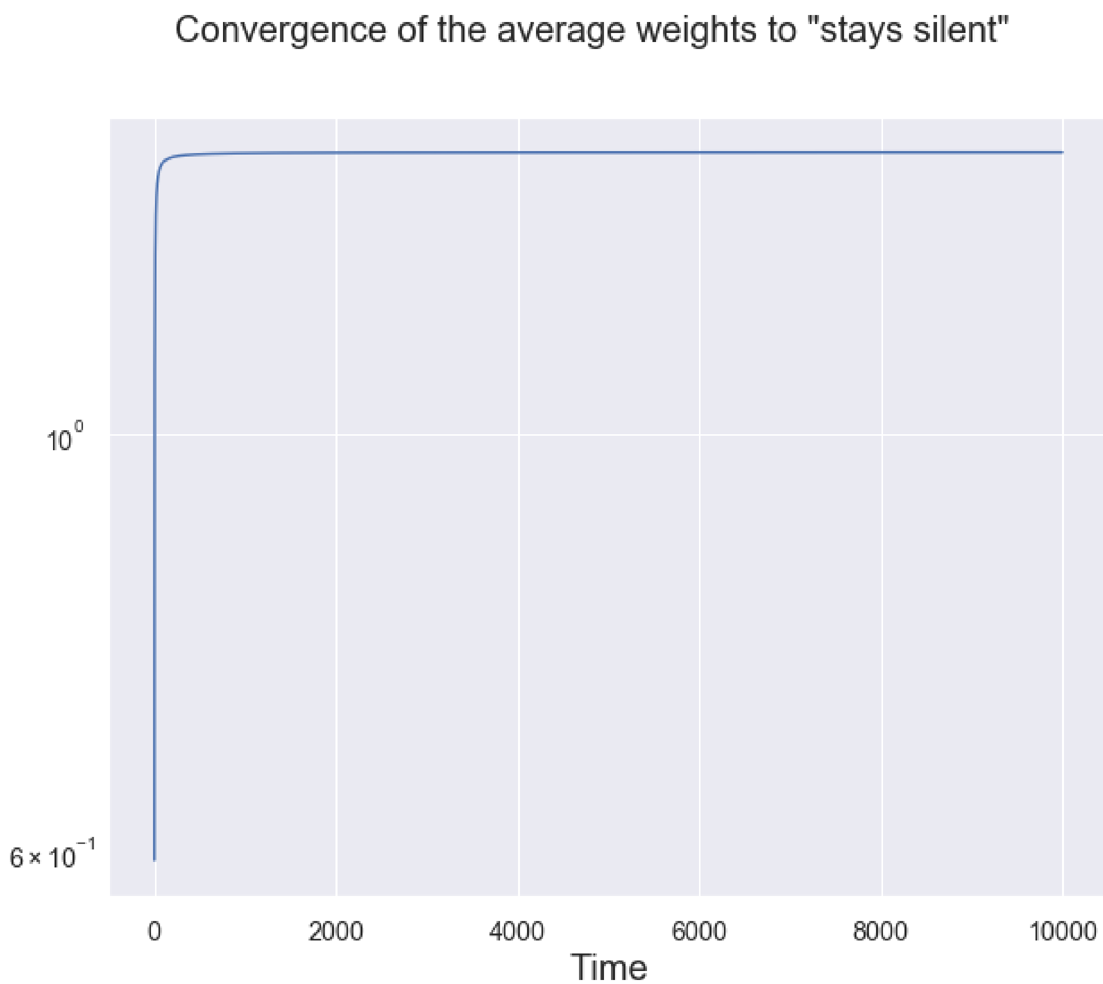


[35]:
```
p_hist_large, _, _ = EWA_EWA_adv(np.random.rand(2), 10000, eta, np.random.
↪rand(2), eta_adv, L_pris)

p_avg = np.zeros_like(p_hist_large)

for t in range(10000):

  p_avg[t,:] = np.sum(p_hist_large[:t+1,:], axis=0)/(t+1)

p_conv = np.linalg.norm(p_avg - np.array([1, 0]), axis=1)

print(p_conv.shape)

plt.figure(figsize=(10, 8))
plt.suptitle('Convergence of the average weights to "stays silent"', size = 20)
plt.plot(p_conv)
```

```
plt.semilogy()
plt.xlabel('Time', size = 20)
```

```
(10000,)
```

[35]:
```
Text(0.5, 0, 'Time')
```

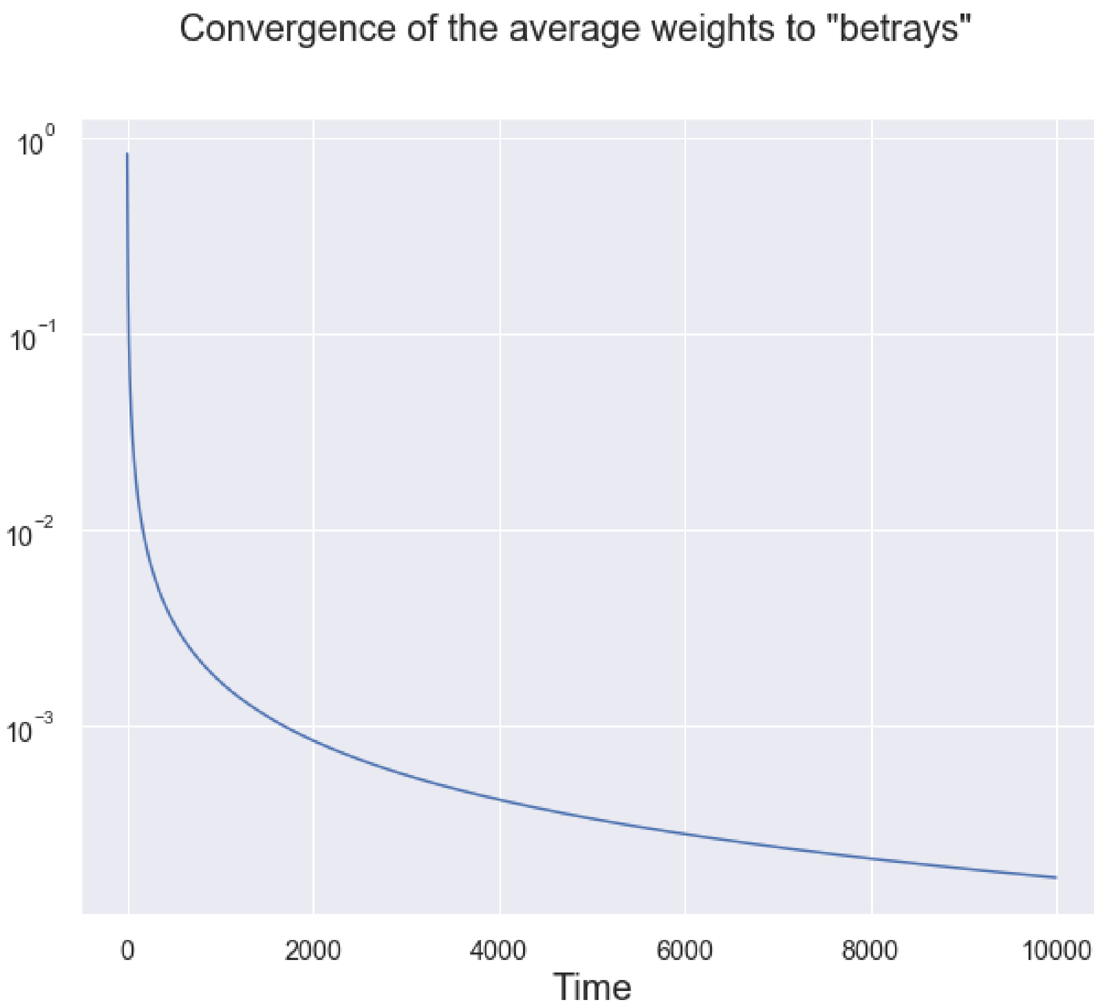


[36]:
```
p_avg = np.zeros_like(p_hist_large)

for t in range(10000):

  p_avg[t,:] = np.sum(p_hist_large[:t+1,:], axis=0)/(t+1)

p_conv = np.linalg.norm(p_avg - np.array([0, 1]), axis=1)

print(p_conv.shape)
```

```
plt.figure(figsize=(10, 8))
plt.suptitle('Convergence of the average weights to "betrays"', size = 20)
plt.plot(p_conv)
plt.semilogy()
plt.xlabel('Time', size = 20)
```

(10000,)

[36]: Text(0.5, 0, 'Time')



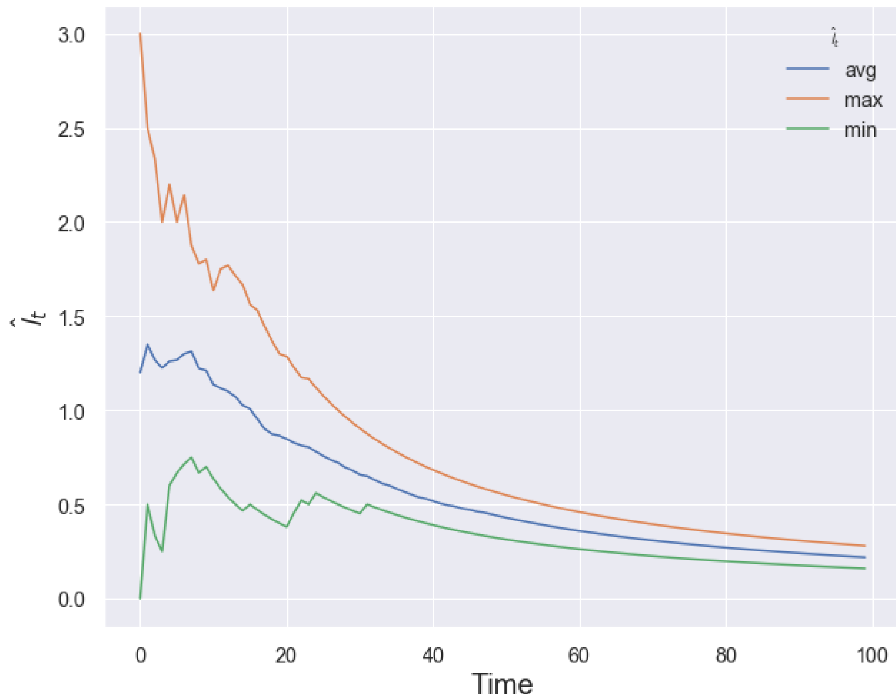
(b) Player plays EXP3 with $\eta = 1$ and adversary plays EWA with $\eta = 0.05$

[37]:
```
eta_adv = 0.05

q0 = (1/2)*np.ones(2)
```

```
l_hist_adv_n = np.zeros((n,T))

for i in range(n):

  _, l_hist_adv_n[i,:], _ = EXP3_EWA_adv(p0, T, eta, q0, eta_adv, L_pris)

l_adv_avg_n = np.zeros_like(l_hist_n)

for t in range(T):

  l_adv_avg_n[:,t] = np.sum(l_hist_adv_n[:,:t+1], axis=1)/(t+1)

l_adv_avg_n_avg = np.sum(l_adv_avg_n, axis=0)/n

l_adv_avg_n_max = np.amax(l_adv_avg_n, axis=0)

l_adv_avg_n_min = np.amin(l_adv_avg_n, axis=0)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the average loss as average, maximum and minimum over␣
 ↪' + str(n) + ' experiments', size = 20)
plt.plot(l_adv_avg_n_avg, label = 'avg')
plt.plot(l_adv_avg_n_max, label = 'max')
plt.plot(l_adv_avg_n_min, label = 'min')
plt.xlabel('Time', size = 20)
plt.ylabel('$\hat{\mathcal{l}}_t$', size = 20)
plt.legend(title='$\hat{\mathcal{l}}_t$')
```

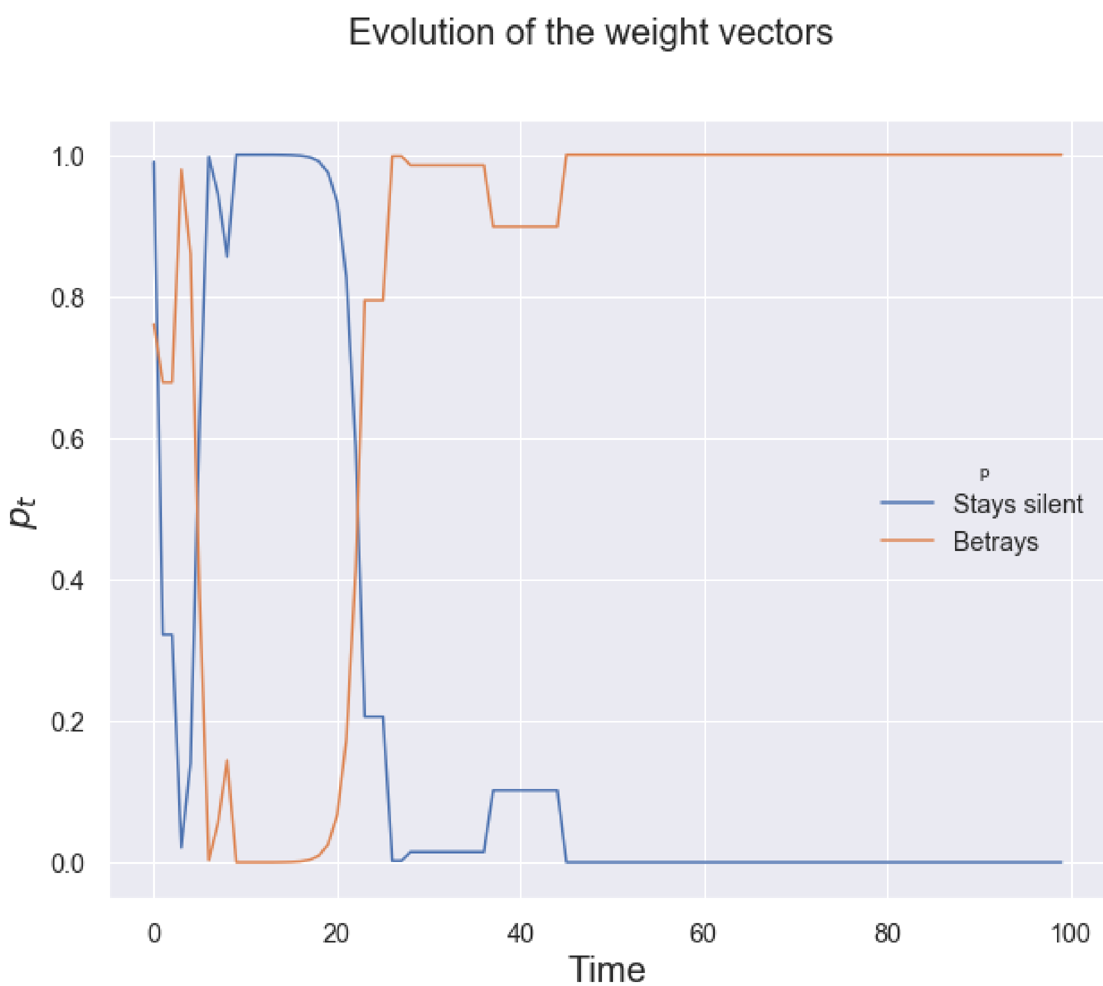
[37]:
```
<matplotlib.legend.Legend at 0x261bead1308>
```

Evolution of the average loss as average, maximum and minimum over 10 experiments



[38]:
```
p_hist_large, _, _ = EXP3_EWA_adv(np.random.rand(2), 100, eta, np.random.
  ↪rand(2), eta_adv, L_pris)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the weight vectors', size = 20)
plt.plot(p_hist_large[:,0], label = 'Stays silent')
plt.plot(p_hist_large[:,1], label = 'Betrays')
plt.xlabel('Time', size = 20)
plt.ylabel('$p_t$', size = 20)
plt.legend(title='p')
```

[38]: <matplotlib.legend.Legend at 0x261be4e31c8>

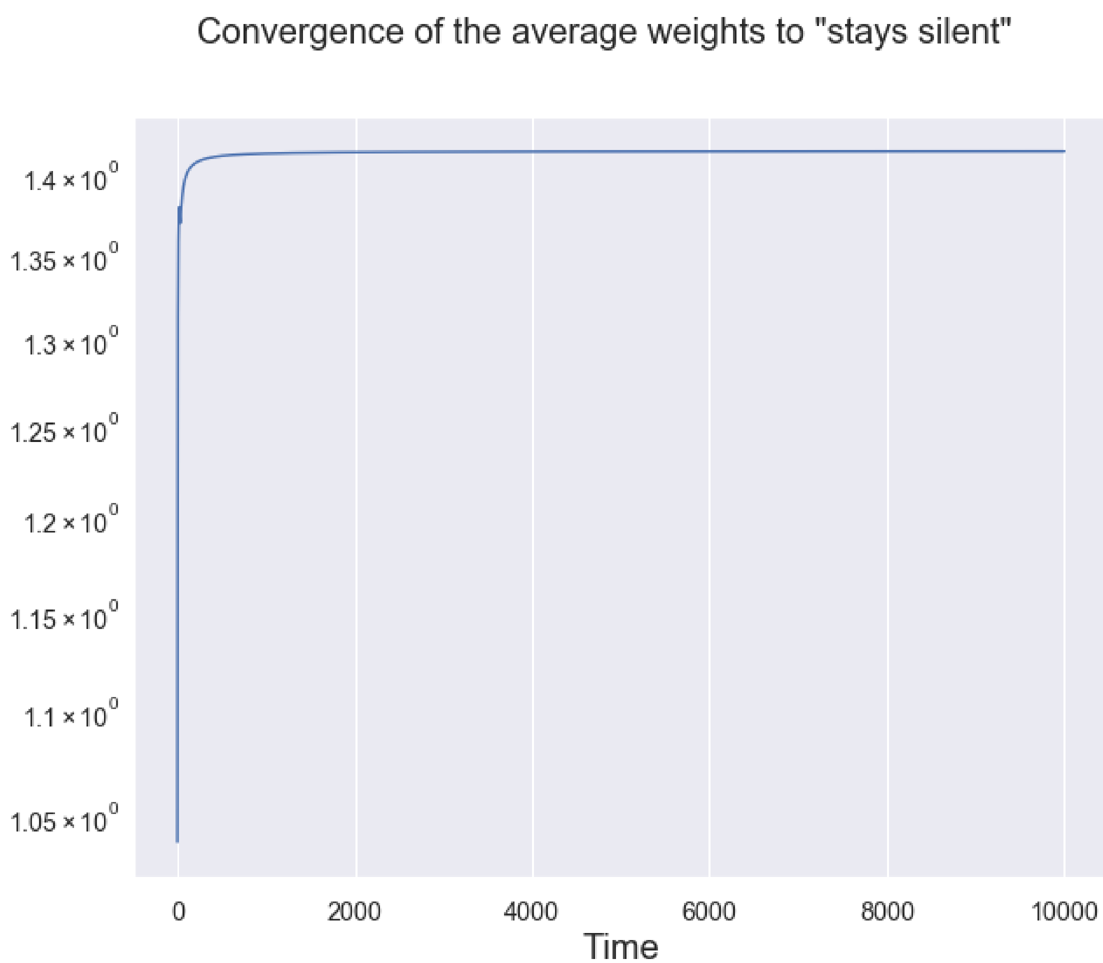



```
[39]: p_hist_large, _, _ = EXP3_EWA_adv(np.random.rand(2), 10000, eta, np.random.
       ↪rand(2), eta_adv, L_pris)


      p_avg = np.zeros_like(p_hist_large)

      for t in range(10000):

        p_avg[t,:] = np.sum(p_hist_large[:t+1,:], axis=0)/(t+1)

      p_conv = np.linalg.norm(p_avg - np.array([1, 0]), axis=1)

      print(p_conv.shape)

      plt.figure(figsize=(10, 8))
      plt.suptitle('Convergence of the average weights to "stays silent"', size = 20)
```

```
plt.plot(p_conv)
plt.semilogy()
plt.xlabel('Time', size = 20)
```

(10000,)

[39]: Text(0.5, 0, 'Time')

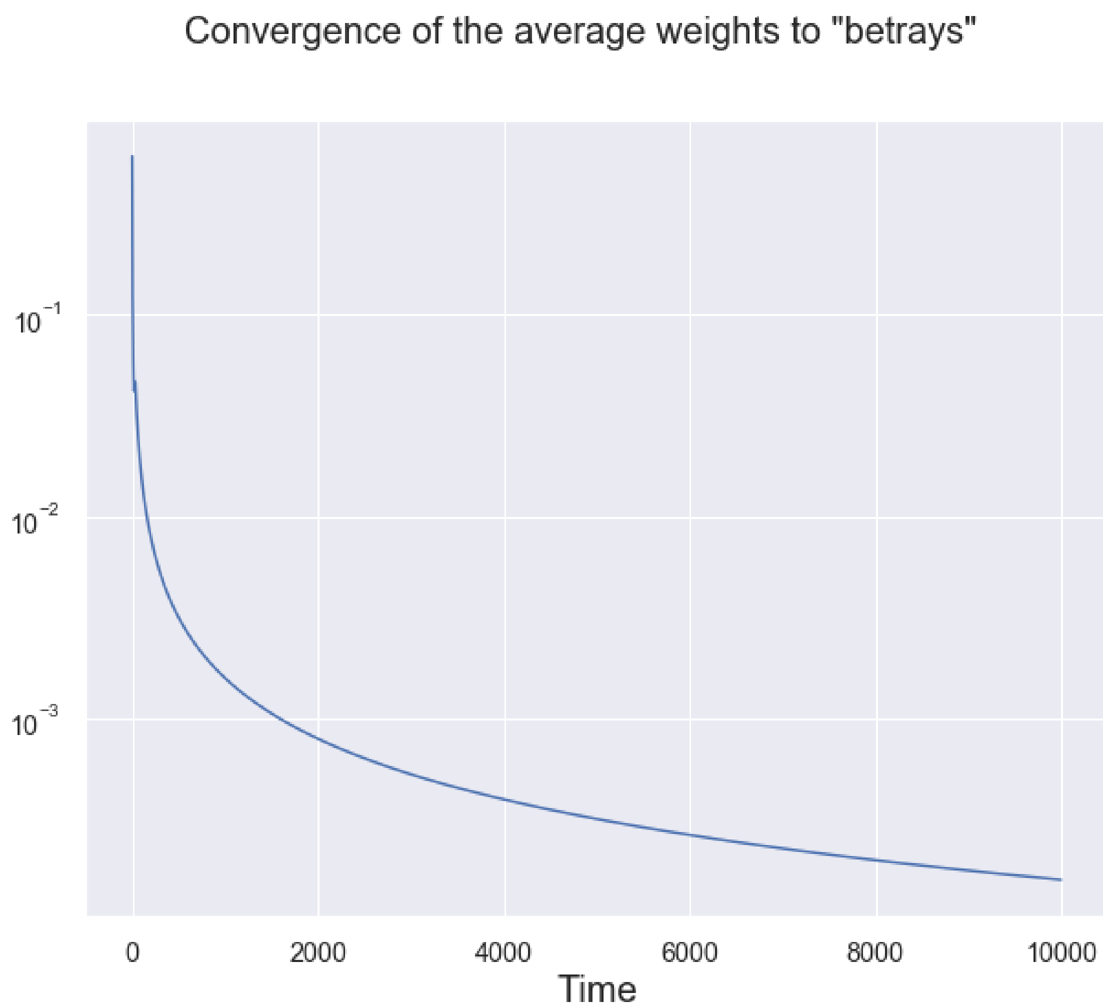


[40]:
```
p_avg = np.zeros_like(p_hist_large)

for t in range(10000):

  p_avg[t,:] = np.sum(p_hist_large[:t+1,:], axis=0)/(t+1)

p_conv = np.linalg.norm(p_avg - np.array([0, 1]), axis=1)

print(p_conv.shape)
```

```
plt.figure(figsize=(10, 8))
plt.suptitle('Convergence of the average weights to "betrays"', size = 20)
plt.plot(p_conv)
plt.semilogy()
plt.xlabel('Time', size = 20)
```

(10000,)

[40]: Text(0.5, 0, 'Time')



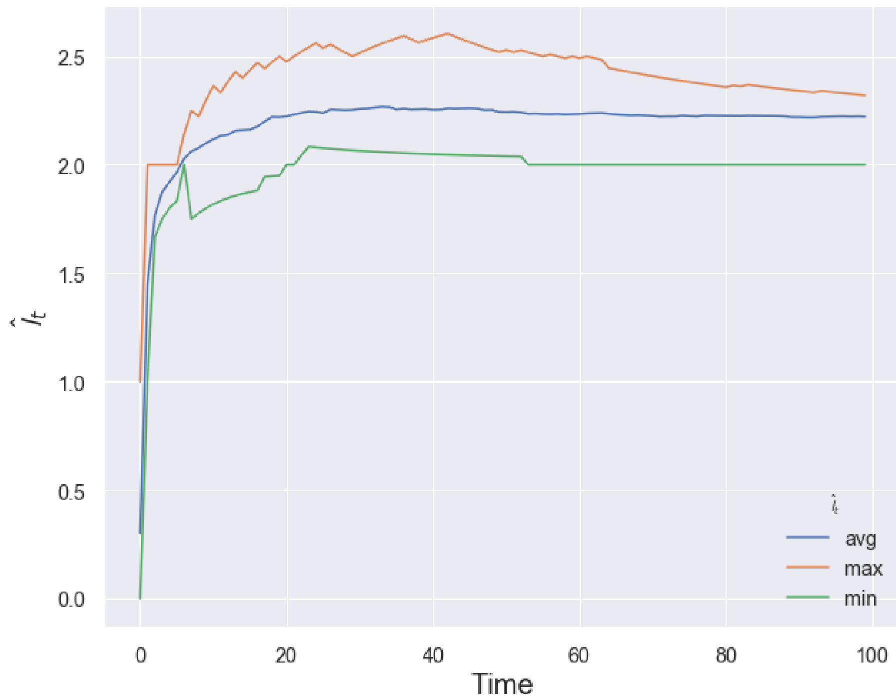
(c) Player plays EXP3 with $\eta = 1$ and adversary plays UCB

[41]:
```
n=10

l_hist_adv_n = np.zeros((n,T))
```

```
for i in range(n):

  _, l_hist_adv_n[i,:], _ = EXP3_UCB_adv(p0, T, eta, L_pris)

l_adv_avg_n = np.zeros((n,T))

for t in range(T):

  l_adv_avg_n[:,t] = np.sum(l_hist_adv_n[:,:t+1], axis=1)/(t+1)

l_adv_avg_n_avg = np.sum(l_adv_avg_n, axis=0)/n

l_adv_avg_n_max = np.amax(l_adv_avg_n, axis=0)

l_adv_avg_n_min = np.amin(l_adv_avg_n, axis=0)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the average loss as average, maximum and minimum over␣
↪' + str(n) + ' experiments', size = 20)
plt.plot(l_adv_avg_n_avg, label = 'avg')
plt.plot(l_adv_avg_n_max, label = 'max')
plt.plot(l_adv_avg_n_min, label = 'min')
plt.xlabel('Time', size = 20)
plt.ylabel('$\hat{\mathcal{l}}_t$', size = 20)
plt.legend(title='$\hat{\mathcal{l}}_t$')
```

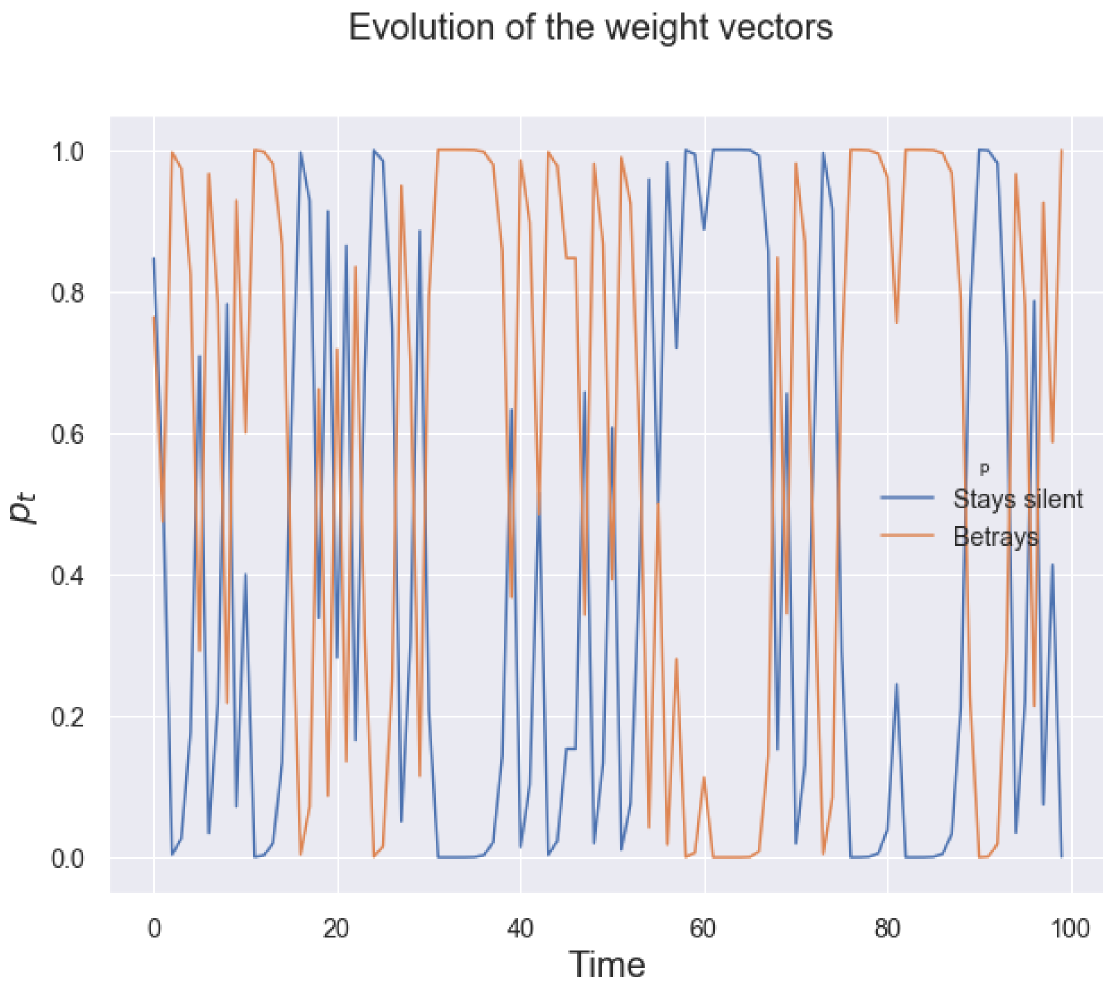
[41]: <matplotlib.legend.Legend at 0x261c21c4e48>

Evolution of the average loss as average, maximum and minimum over 10 experiments



[42]:
```
p_hist_large, _, _ = EXP3_UCB_adv(np.random.rand(2), 100, eta, L_pris)

plt.figure(figsize=(10, 8))
plt.suptitle('Evolution of the weight vectors', size = 20)
plt.plot(p_hist_large[:,0], label = 'Stays silent')
plt.plot(p_hist_large[:,1], label = 'Betrays')
plt.xlabel('Time', size = 20)
plt.ylabel('$p_t$', size = 20)
plt.legend(title='p')
```

[42]:
```
<matplotlib.legend.Legend at 0x261c259e288>
```

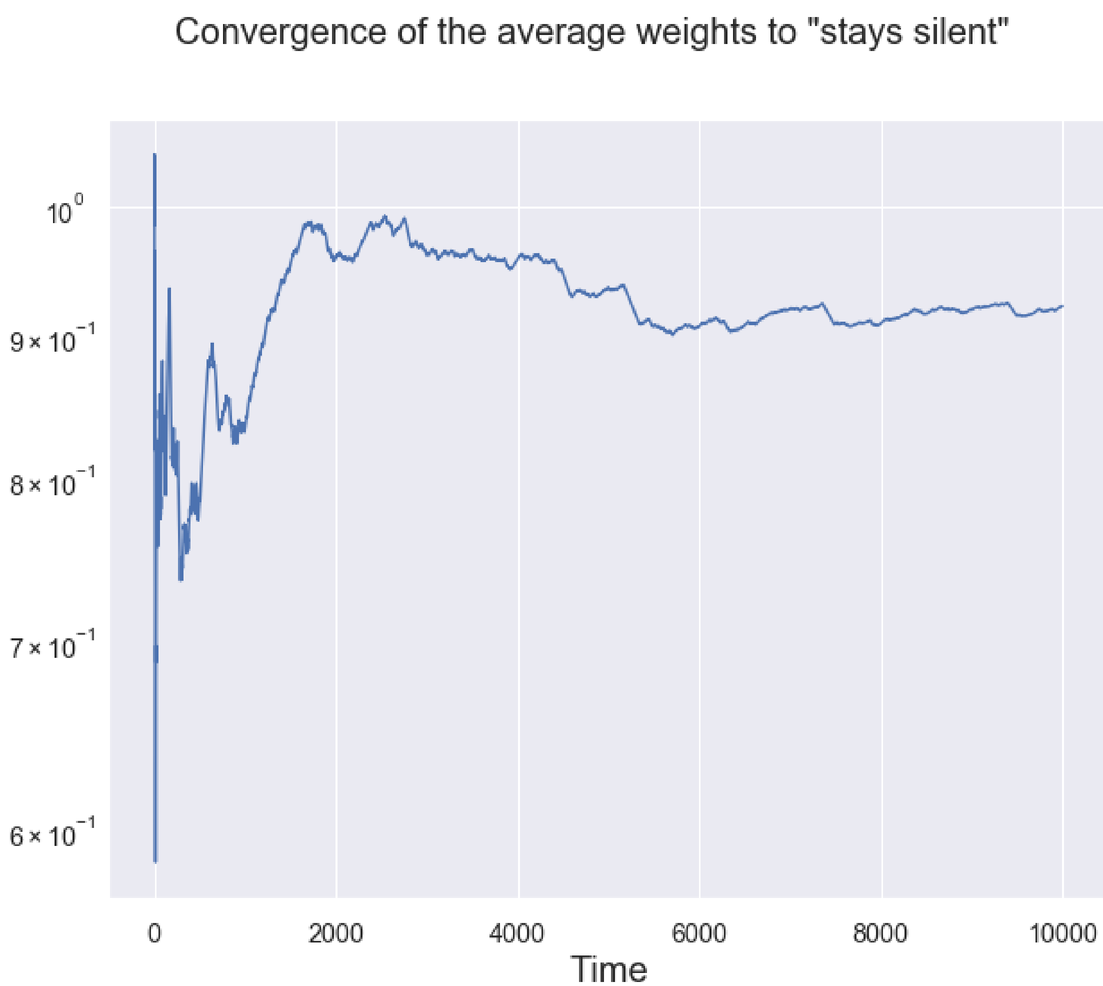


[43]:
```
p_hist_large, _, _ = EXP3_UCB_adv(np.random.rand(2), 10000, eta, L_pris)

p_avg = np.zeros_like(p_hist_large)

for t in range(10000):

  p_avg[t,:] = np.sum(p_hist_large[:t+1,:], axis=0)/(t+1)

p_conv = np.linalg.norm(p_avg - np.array([1, 0]), axis=1)

print(p_conv.shape)

plt.figure(figsize=(10, 8))
plt.suptitle('Convergence of the average weights to "stays silent"', size = 20)
plt.plot(p_conv)
plt.semilogy()
```

```
plt.xlabel('Time', size = 20)
```

(10000,)

[43]: Text(0.5, 0, 'Time')

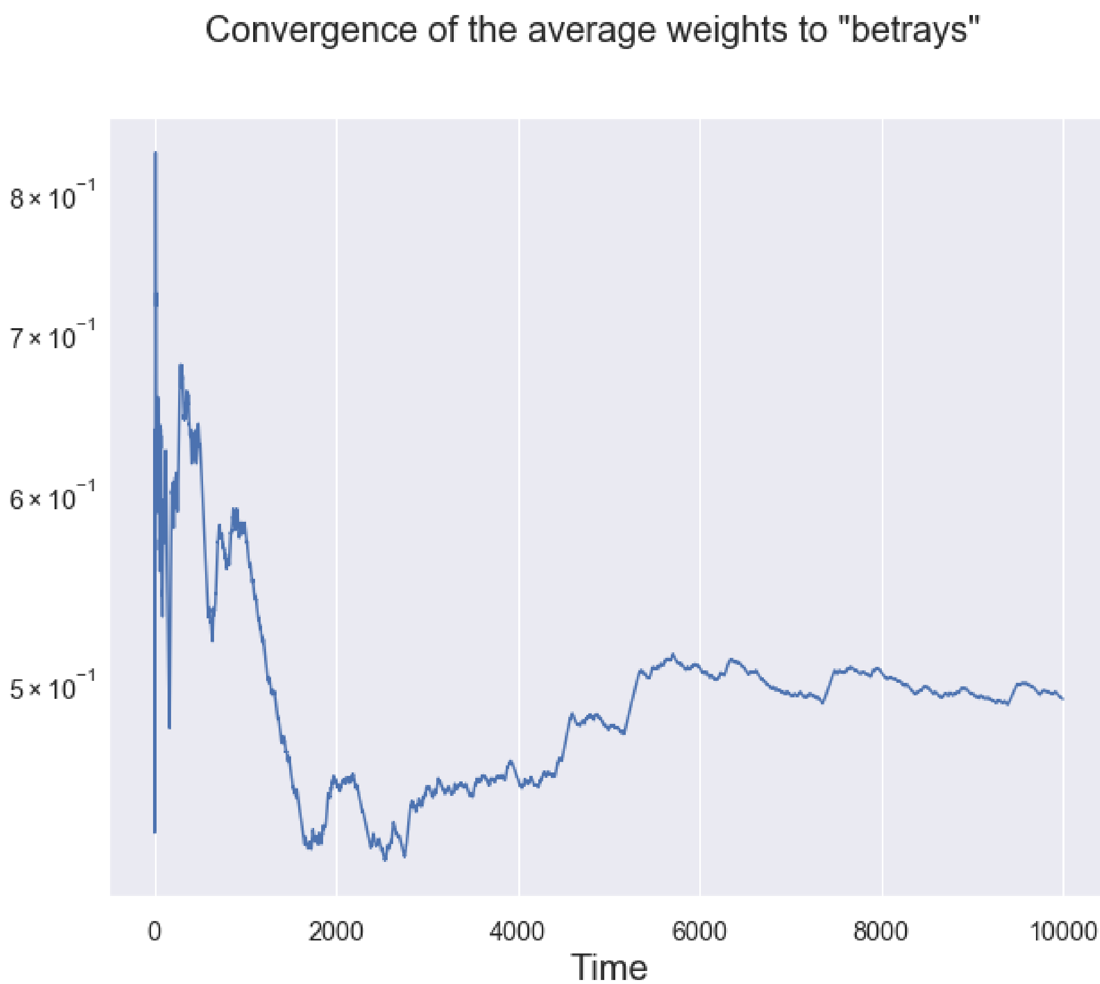


[44]:
```
p_avg = np.zeros_like(p_hist_large)

for t in range(10000):

  p_avg[t,:] = np.sum(p_hist_large[:t+1,:], axis=0)/(t+1)

p_conv = np.linalg.norm(p_avg - np.array([0, 1]), axis=1)

print(p_conv.shape)

plt.figure(figsize=(10, 8))
```

```
plt.suptitle('Convergence of the average weights to "betrays"', size = 20)
plt.plot(p_conv)
plt.semilogy()
plt.xlabel('Time', size = 20)
```

```
(10000,)
```

[44]:
```
Text(0.5, 0, 'Time')
```



2 Part 3. Experiments - predict votes of surveys

In these experiments, we will apply online convex optimization algorithms to pairwise comparison datasets. Comparison data arises in many different applications such as sport competition, recommender systems or web clicks. We consider the following sequential setting. Let $\mathcal{I} = \{1, \ldots, N\}$ be a finite set of items (for example football teams in a competition).

At each iteration $t \geq 1$, - the learner receives the labels of two items that are competing

$z_t = (z_t(1), z_t(2)) \in \mathcal{Z}^2$ - the learner predicts $\hat{y}_t \in (0,1)$ the probability of victory of item $z_t(1)$ - the environment reveals the result of the match $y_t = 1$ if item $z_t(1)$ wins the match and $y_t = 0$ otherwise (if team $z_f(2)$ wins) The learner aims at minimizing his cumulative loss: $\hat{L}_T = \sum_{t=1}^{T} \ell(\hat{y}_t, y_t)$, where $\ell(\hat{y}_t, y_t) = (1 - \hat{y}_t) y_t + \hat{y}_t (1 - y_t)$

The **datasets** contain two files: - ideas-id.csv (resp. politicans-id.csv) that contains id and text of the ideas (resp. political figures). - ideas-votes.csv (resp. politicians-votes.csv) that contains the id of the two competing ideas (resp. political figures) in $z1$ and $z2$ and a column y which is 1 if the participant voted for $z1$ and 0 otherwise.

The goal of the learner is to sequentially predict the results of the votes minimizing the number of mistakes.

[45]:
```
ideas_id = pd.read_csv("ideas_id.csv")
ideas_votes = pd.read_csv("ideas_votes.csv")
```

[46]:
```
K, n_col = ideas_id.shape
T, n_col = ideas_votes.shape
y = ideas_votes['y']
print("number of ideas: ",K)
print("number of votes: ",T)
```

```
number of ideas:  261
number of votes:  15000
```

Sleeping strategies

[47]:
```
def f1(data, K, t,p):
    """
    Sleeping strategy
    Input:
    -----
    data: dataset containing the competing items
    K: number of items
    t: round t (t in {1,...,T})
    p: weigthing vector
    Output:
    ------
    f: experts predictions
    y_t: prediction for round t
    """
    f_t = [1, 0, 0, 1]
    p = p[[data["z1"][t]-1,data["z1"][t]-1+K,data["z2"][t]-1,data["z2"][t]-1+K]]
    p = p/np.sum(p)
    y_t = np.sum(f_t*p)

    f = np.zeros(2*K)
    for k in range(K):
      if data["z1"][t] == k+1:
```

```python
      f[k], f[k+K] = 1, 0
    elif data["z2"][t] == k+1:
      f[k], f[k+K] = 0, 1
    else:
      f[k], f[k+K] = np.round(y_t), np.round(y_t)
  return f, y_t
```

The Exponentially weighted average forecaster (EWA)

[48]:
```python
def EWA(eta, f1, K, y, T, data):
  """
  Exponentially weighted average forecaster
  Input:
  -----
  eta:
  gt:
  T: Horizon
  K: actions
  Output:
  ------
  p: weighting vector
  loss: expected loss for each t in {1,...,T}
  true_loss: true loss for each t in {1,...,T}
  y_pred: predictions of votes
  """

  #initialization
  p = 1/(2*K) * np.ones(2*K)
  loss = [0]
  y_pred = []
  true_loss = [0]
  for t in range(T):
    #if t% 1000== 0:
    #  print("############ iteration " + str(t) + " ############" )
    #  print("cumulative loss", np.cumsum(loss)[-1])
    f, y_t = f1(data, K, t, p)
    y_pred.append(np.round(y_t))
    gt = (1-f)*y[t] + f*(1-y[t])
    loss.append((1-y_t)*y[t] + y_t*(1-y[t]))
    true_loss.append((1-np.round(y_t))*y[t] +
                     np.round(y_t)*(1-y[t]))

    renorm = np.sum(p*np.exp(-eta*gt))
    p = p*np.exp(-eta*gt)/renorm
  return p, loss,true_loss, y_pred
```

Online Gradient Descent (OGD)

[49]:
```
def OGD(eta, f1, T, K, y, data):
  """
  Exponentially weighted average forecaster
  Input:
  -----
  eta:
  ft:
  T: Horizon
  Output:
  ------
  theta = weighting vector
  """
  theta = 1/(2*K) * np.ones(2*K)
  loss = [0]
  true_loss = [0]
  y_pred = []
  for t in range(T):
    f, y_t = f1(data, K, t, theta)
    y_pred.append(np.round(y_t))
    loss.append((1-y_t)*y[t] + y_t*(1-y[t]))
    true_loss.append((1-np.round(y_t))*y[t] + np.round(y_t)*(1-y[t]))
    grad = (-y[t] + 1)*theta
    norm = theta[[data["z1"][t]-1,data["z1"][t]-1+K,
                  data["z2"][t]-1,data["z2"][t]-1+K]]
    norm = np.sum(norm)
    theta = (theta - eta*grad)/norm
  return theta, loss, true_loss, y_pred
```

3 Plot functions

[58]:
```
def plt_eta(etas, dataset, save = False):
  """
  Input:
  -----
  etas: (list)
        list containing different values of eta
  dataset: dataset to be used
  save: (bool) if True, we save the plt
  """
  fig = plt.figure()

  for i, eta in enumerate(etas):
    p, loss, true_loss, y_pred = EWA(eta, f1, K, y,
                                     15000, dataset)
```

```
    print("score for $\eta=$"+ str(eta), np.sum(y_pred == y)/len(y))
    plt.plot(np.linspace(1,len(loss),15001),np.cumsum(loss),
            label = "$\eta = $ "+ str(eta))
    plt.title("Evolution of the cumulative loss")
    plt.xlabel("Iteration")
    plt.ylabel("Cumulative loss")
  plt.legend()
  plt.tight_layout()
  if save :
    plt.savefig("comparison_loss_etas.png")
    plt.savefig("comparison_loss_etas.pdf")
  plt.show()
```

[61]:
```
def plot_cumloss(loss, loss_name):
  cumsum = np.cumsum(loss)
  avg_loss= [1/t * cumsum[t-1] for t in range(1, 15000 )]

  plt.plot(np.linspace(1,len(cumsum),14999),avg_loss)
  plt.title("Evolution of the "+ str(loss_name))
  plt.xlabel("Iteration")
  plt.ylabel(str(loss_name))
  plt.show()
```

ploting our results for the ideas dataset

[53]:
```
ideas_id = pd.read_csv("ideas_id.csv")
ideas_votes = pd.read_csv("ideas_votes.csv")
```

[54]:
```
K, n_col = ideas_id.shape
T, n_col = ideas_votes.shape
y = ideas_votes['y']
print("number of ideas: ",K)
print("number of votes: ",T)
```

```
number of ideas:  261
number of votes:  15000
```

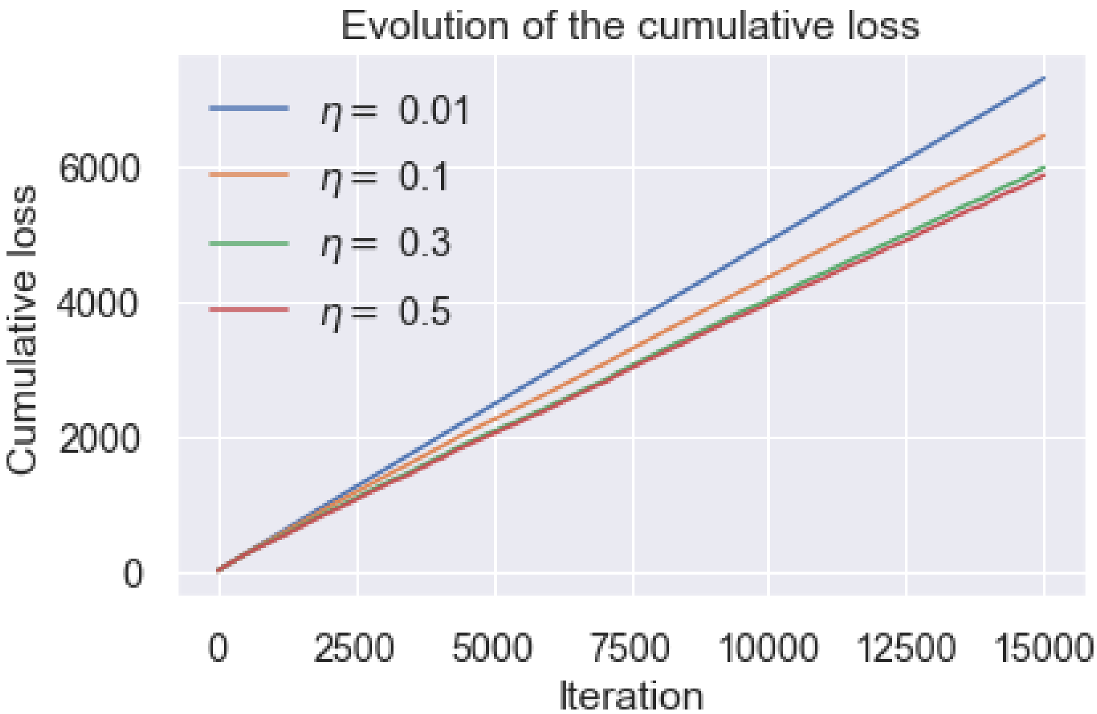
[55]:
```
p, loss,true_loss, y_pred = EWA(0.5, f1, K, y, 15000, ideas_votes)
```

Plot of the cumulative loss for different eta

[59]:
```
etas = [0.01, 0.1, 0.3, 0.5]
plt_eta(etas, ideas_votes)
```

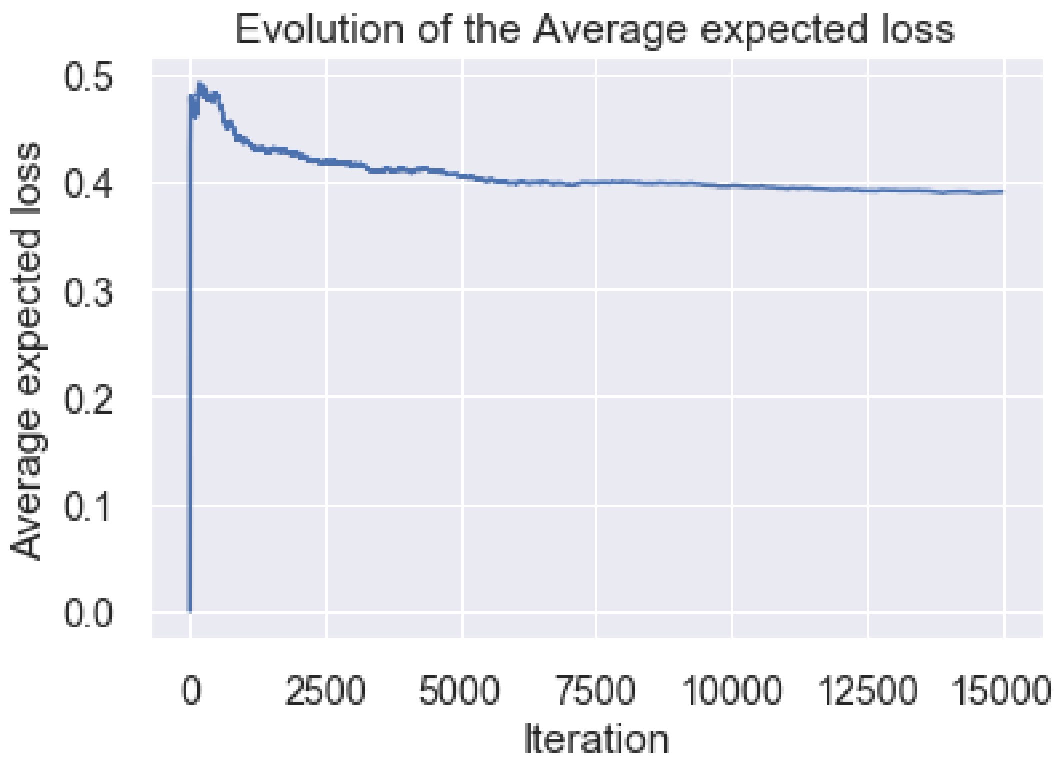
```
score for $\eta=$0.01 0.6252
score for $\eta=$0.1 0.6246666666666667
```

```
score for $\eta=$0.3 0.626
score for $\eta=$0.5 0.6226
```



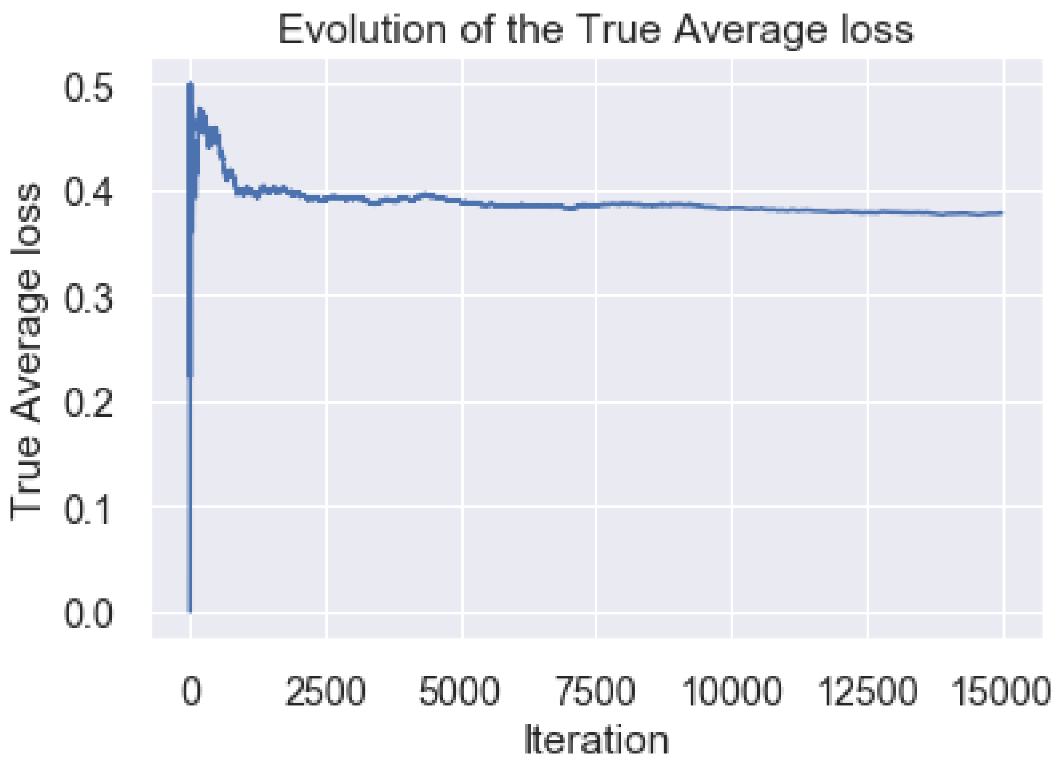
Plot of the average expected loss

[62]:
```
plot_cumloss(loss, "Average expected loss")
```



Plot of the true average loss

[63]:
```
plot_cumloss(true_loss, "True Average loss")
```



ploting our results for the politicians dataset

[64]:
```
politicians_id = pd.read_csv("politicians_id.csv")
politicians_votes = pd.read_csv("politicians_votes.csv")
K, n_col = politicians_id.shape
T, n_col = politicians_votes.shape
y = politicians_votes['y']
print(K)
print(T)
```

```
39
15000
```

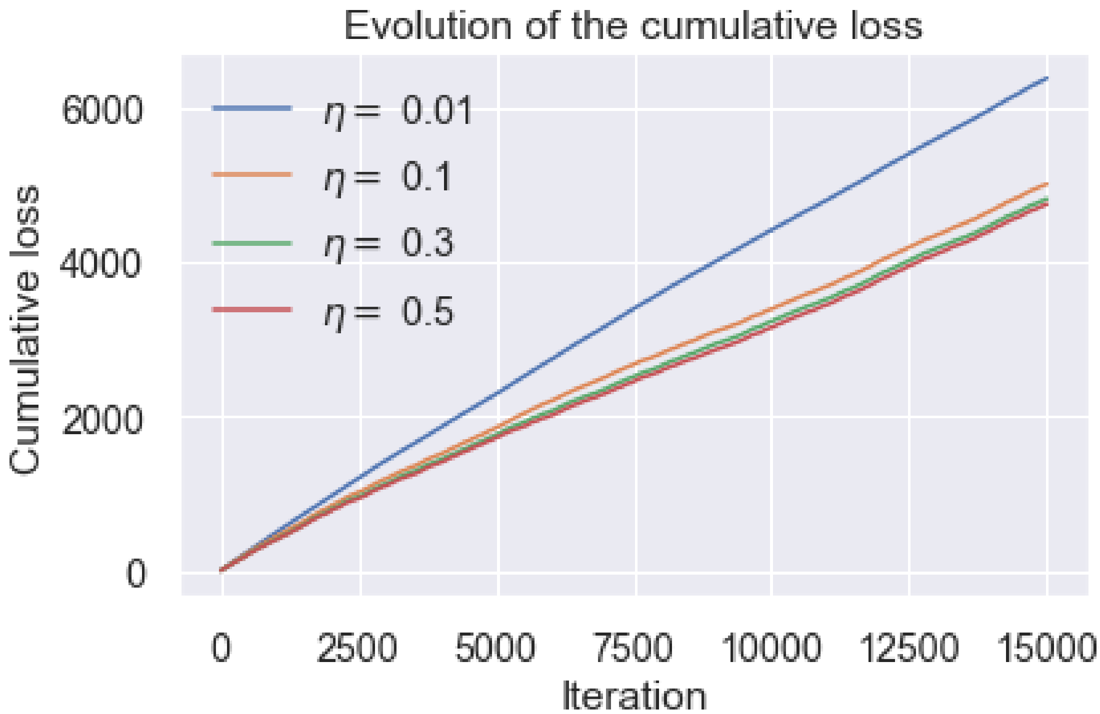
[65]:
```
p, loss,true_loss, y_pred = EWA(0.5, f1, K, y, 15000, politicians_votes)
```

Plot of the cumulative loss for different eta

[68]:
```
etas = [0.01, 0.1, 0.3, 0.5]
plt_eta(etas, politicians_votes)
```

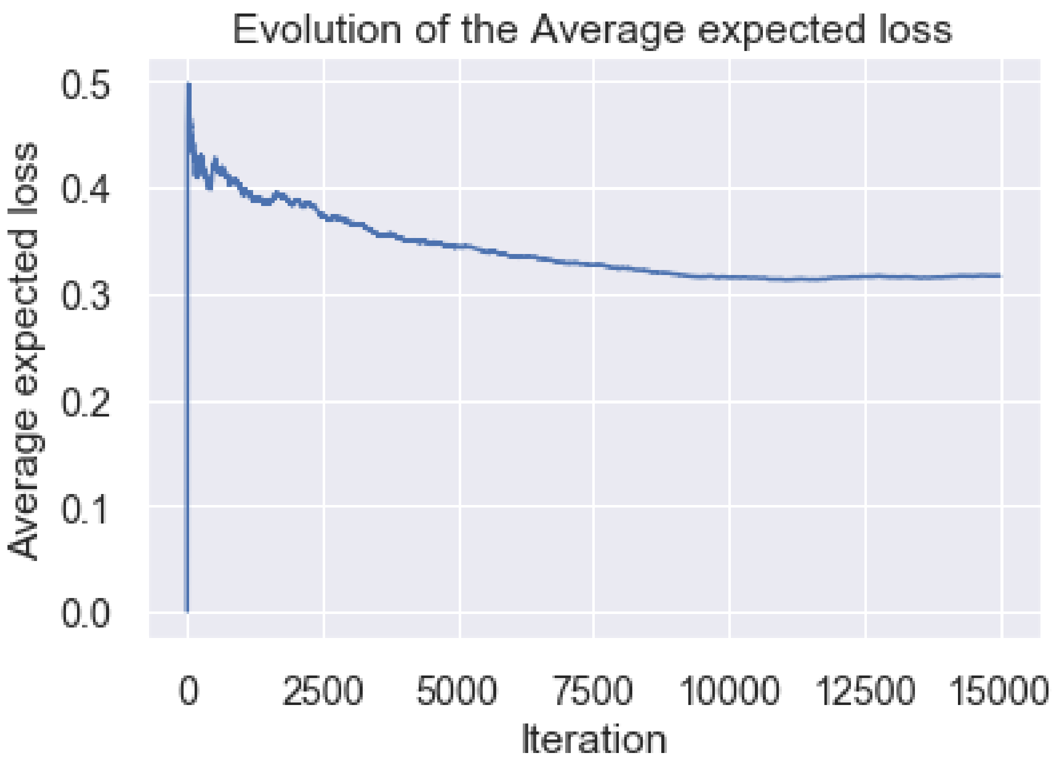
```
score for $\eta=$0.01 0.6886666666666666
score for $\eta=$0.1 0.6852666666666667
```

```
score for $\eta=$0.3 0.6862
score for $\eta=$0.5 0.6858
```



Plot of the average expected loss

```
[69]: plot_cumloss(loss, "Average expected loss")
```



Plot of the true average loss

[70]:
```
plot_cumloss(true_loss, "True Average loss")
```

Evolution of the True Average loss
True Average loss
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
2500
5000
7500
10000
12500
15000
Iteration