# Numerical calculations for Universal Batch Learning

Submitted by: Eyal Goldstein

Instructed by: Meir Feder and Yaniv Fogel

Sep. 2023

# Abstract

In this paper we consider the problem of batch learning with log-loss. We implemented an algorithm to solve it numerically while addressing computational boundaries. After we proved the classic algorithm is correct, we modified it bit by bit to handle both the binary and the non-binary binomial channel. Visualizing the numerical results helped us learn more about the nature of the problem and raised more questions to follow up.

# Introduction

## Universal prediction

The concept of universal prediction involves predicting the future of a sequence based on its past without knowledge of the underlying model. In universal prediction, self-information loss functions are used, which are convenient, reflect uncertainty, and relate to lossless source coding. As well as providing insight into other loss functions, it can serve as a mechanism for generating probability distributions when the underlying source is unknown or nonexistent.

The problem of probability assignment for the next outcome given the past, under the self-information loss function. As a result, finding a probability assignment for the entire data sequence is equivalent to finding a probability assignment for the next outcome as discussed in [2].

A field of interest in machine learning is distribution estimation, where one is given a batch of outcomes and is requested to evaluate the probability for the next outcome. The measure of performance is called Regret, and the minimax theorems are useful in this context.

Previous work was done before to calculate the capacity of a binomial binary channel and find how many mass-points does it have. In [4] we see an attempt to understand the nature of the problem, and in [3] we can see a numerical implementation of a code that finds the mass-point for a given training set.

On this paper, we will try to improve the calculations available today for the problem above. Additionally we will adjust the algorithm to test the hypothesis from [1]. When achieving this, we will dive deeper and examine the non-binary channel problem.

# Preparation and code design

First, we implemented the Blahut-Arimoto algorithm [5]. We used the code over various resolutions of and training size . In Figure 1 you can see the result for a calculation with high resolution in for training set of size . With this representation one can get a hold on how many mass points as described in [3] there really are.

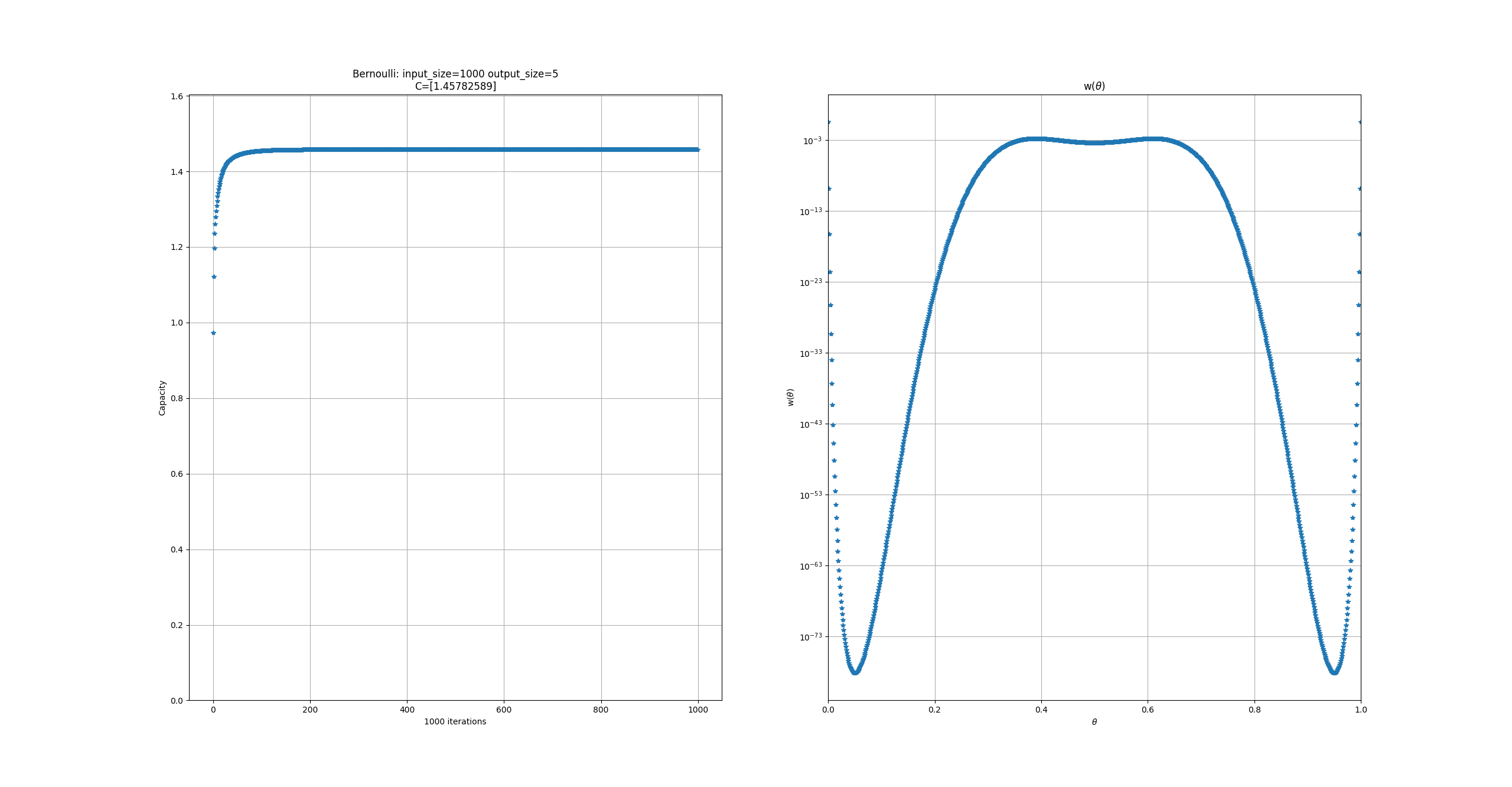


Figure 1: Weights and capacity solution for training set of size 10. Observed 4 Mass-points

For the second stage we improved our algorithm by considering the derivative of the weights. The Pseudo-code is as follows (given , numer of mass-points assumed, and threshold):

derivative-Blahut-Arimoto: dBA(Training set #, Mass-point #, threshold)

For the final stage we assumed the closest value for the number of mass-points based on our previous calculations and [3]. The algorithm guess that which corresponded with high match to the results of [3]. With closer examination we noticed that in some boundary conditions due to the granularity of the mass points and some results were “missing” the central mass point. That led us to refactor our code, which its Pseudo-code is as follows (given ):

Whole-Blahut-Arimoto: wBA(Training set #)

# Binomial channel

Figures 2 and 3 show the results for training set size and with capacities of 2.67 and 3.10 respectively, which holds to the theory that .

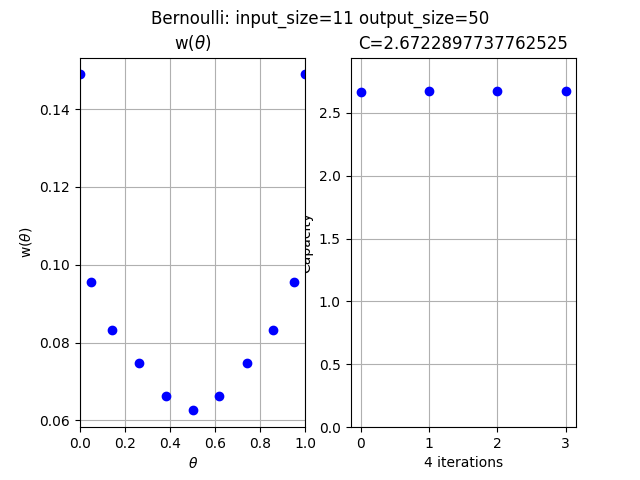


Figure 2: wBA solution for training set of size 50. Algorithm found 11 Mass-points and got capacity of 2.67.

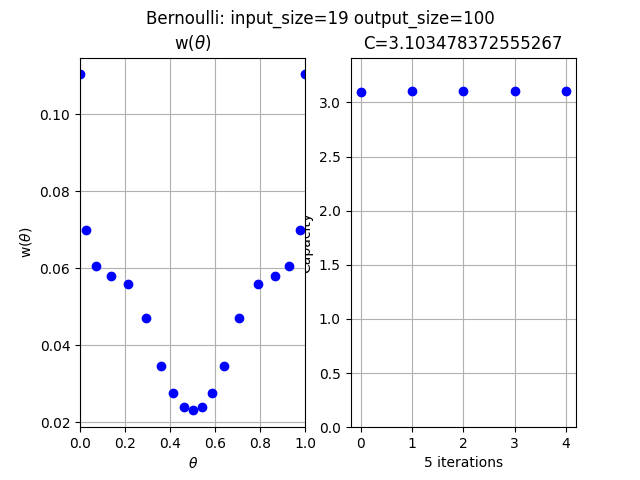


Figure 3: wBA solution for training set of size 100. Algorithm found 19 Mass-points and got capacity of 3.10.

With this code, we could calculate the capacity for training set size up to 10 times (and even more with more computational power). In Table 1 presented the number of mass-points for every training size between 50 and 100:

|  |  |
| --- | --- |
| 59 | 14 |
| 66 | 15 |
| 77 | 16 |
| 82 | 17 |
| 90 | 18 |
| 99 | 19 |
| 108 | 20 |

Table 1: Number of Mass-points given training set size N

As can be seen, we reproduced with high correspondence the results of [3] and even got to much higher training set size with normal computation power (home PC). With this we can move on to run the code on a much stronger machine, say GPU.

# Batch learning with Log-Loss

For this chapter, we had to adjust our algorithm to support the analytical work done in [1]. After the idea was to make the Blahut-Arimoto algorithm get to a result in only one step instead of two.

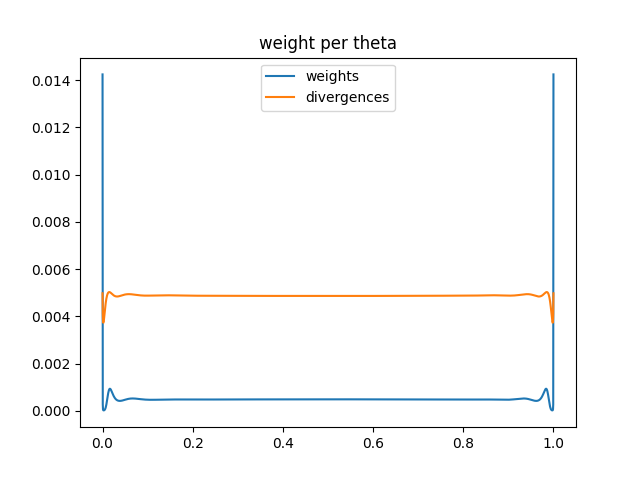


Figure 4: Weights results for training set of size 100. Observe the “pools” near the edges where it gets to zero.

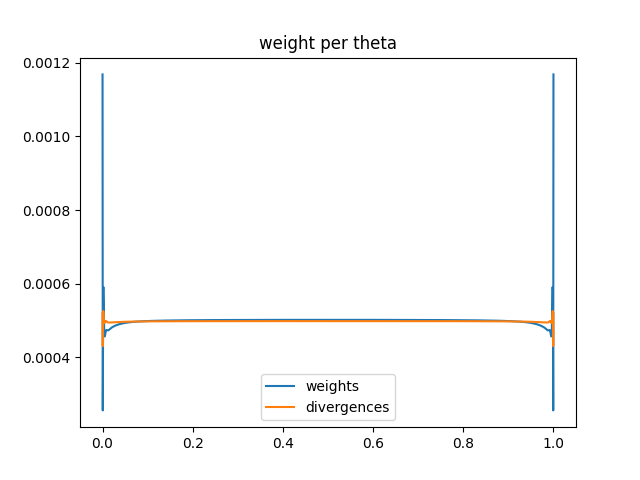


Figure 5: Weights results for training set of size 1000. Observe the “pools” near the edges where it gets to zero.

In Figures 4 and TB5 D presents the result for training set size of and . There are three important things to showcase here:

**The weights go to zero where the divergences are not max-value: An interesting thing we saw are the “pools” near the edges. As you can see, the weights go to zero right before the peaks at . When printing the divergences on top of it, one might notice that the weights behave like this where the divergences are not at the maximum value.**

**Capacity convergence to** : Figures 6 and 7 present the convergence of the capacities for training set size of . We saw that both are converging to as expected.

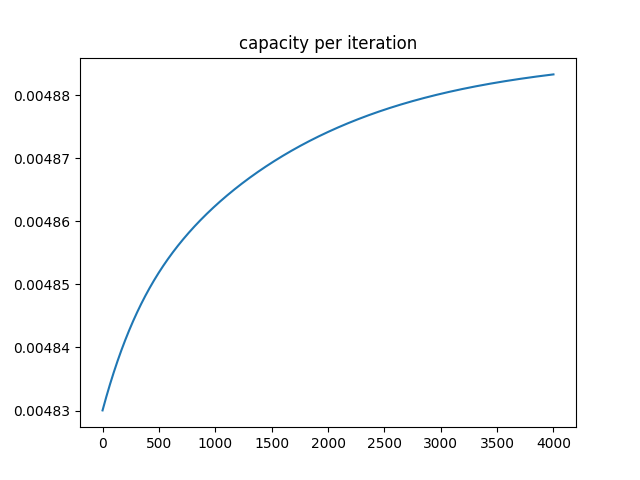


Figure 6: Capacity result for training set of size 100. Observe the convergence to

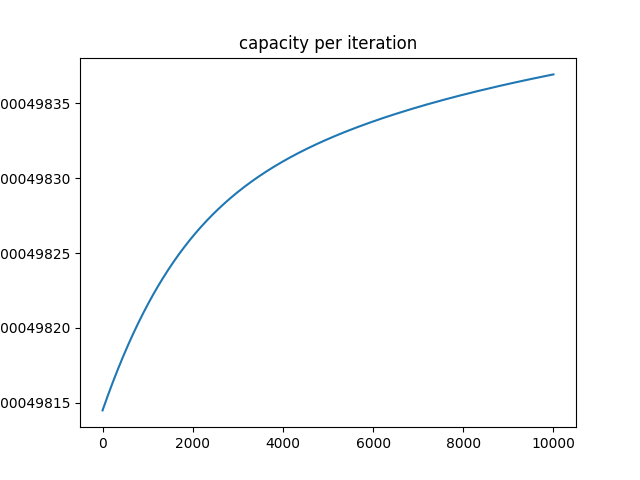


Figure 7: Capacity result for training set of size 1000. Observe the convergence to

**Add- value: As discussed in [1]** a classical predictor is the add- predictor where a constant of is added to the empirical counts of the training. **Figures 8 and 9 present the solution for the . We added the theoretical values of 0.5, 0.75 and 1 to see a visual comparison with our results.**

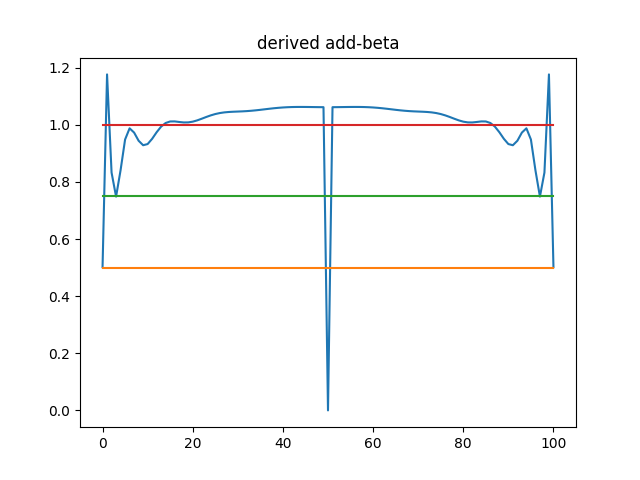


Figure 8: Add- result for training set of size 100

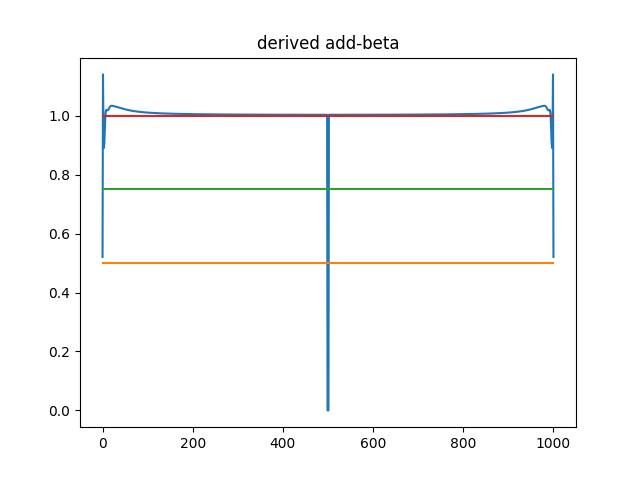


Figure 9: Add- result for training set of size 1000

# Non-binary channel

The next challenge we wanted to address was the non-binary channel. For that we needed to do even more adjustments in the code to support it.

The algorithm was modified in a way it can support a channel with 3 “participants”, so we can plot the weights for each combination of thetas . Figure 10 present the results.

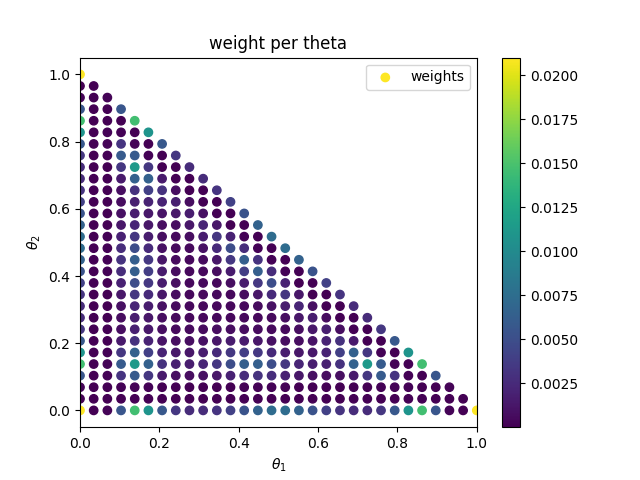


Figure 10: Weights result for training set of size 20 for non-binary channel

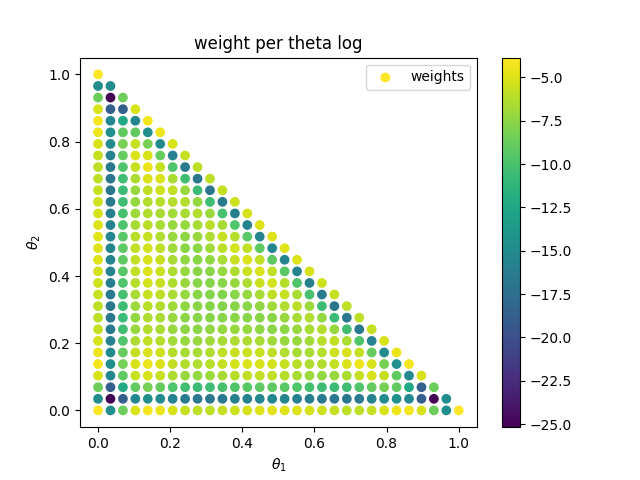


Figure 11: Weights result in log for training set of size 20 for non-binary channel

As can be seen in Figures 10 and 11, it sets well with the results for the binary channel we had. There are the “pools” near the edges. In Figure 12 presented the capacity, which seem to converge to this time around.

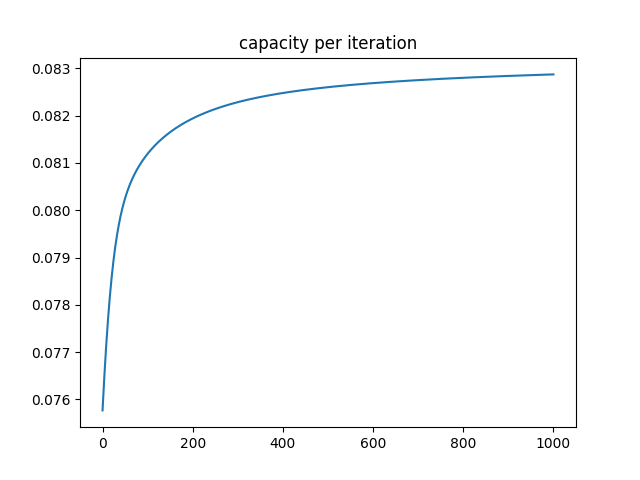


Figure 12: Weights result in log for training set of size 20 for non-binary channel

# Conclusion

In this paper we implemented a numerical approach for the problem of batch learning with respect to the logarithmic loss function. Firstly, we reproduced calculations to check whether our algorithm is reliable and then moved on and improved it bit by bit. The first stage was to get more results and data for the classical problem. We managed to get to 10 times of what was presented in other researches.

The second stage was to modify our algorithm to numerically solve the batch learning problem with regard to conditional calculations. We observed interesting behaviors and examined them closely.

Lastly, we tried to address the non-binary channel problem. We got interesting results that set well with the theory and the results we got from the binary channel.

As mentioned, the algorithm and the code require high computational power. The whole code was written with support to run on both CPU and GPU to be able to run in a parallel manner.

# References

1. Y. Fogel and M. Feder, "Universal Batch Learning with Log-Loss," 2018 IEEE International Symposium on Information Theory (ISIT), pp. 21-25, 2018.
2. N. Merhav and M. Feder, “Universal prediction,” IEEE Transactions on Information Theory, vol. 44, no. 6, pp. 2124–2147, 1998.
3. N. Farsad et al., "Capacities and Optimal Input Distributions for Particle-Intensity Channels," in IEEE Transactions on Molecular, Biological and Multi-Scale Communications, vol. 6, no. 3, pp. 220-232, 2020.
4. H. Mattingly et al., “Maximizing the information learned from finite data selects a simple model”, in arXiv: 1705.01166v3, 2018.
5. R. Blahut, "Computation of channel capacity and rate-distortion functions," in IEEE Transactions on Information Theory, vol. 18, no. 4, pp. 460-473, 1972.