INF 242: Blahut-Arimoto algorithm for computing the capacity

Paal Halvorsen

October , 2024

Contents

1 Introduction

The following project has been done in Python to implement Blahut-Arimoto algorithm and thereby calculate the channel capacity C of a DMC with input X and output Y, given by channel transition probability distribution $f_{Y|X}(y|x)$. In this solution, the implemented algorithm has been used to calculate the capacity of channels A, B, C, D, and E.

2 Blahut-Arimoto Algorithm Overview

The Blahut-Arimoto algorithm is an iterative method used to compute the capacity C of a DMC by maximizing the mutual information $I(X;Y)$. The algorithm involves a double maximization over the input distribution $f_X(x)$ and an auxiliary conditional distribution $r(x|y)$.

The capacity is given by:

$$C = \max_{f_X} I(X;Y) = \max_{f_X} \max_{r(x|y)} \sum_{x \in X} \sum_{y \in Y} f_X(x) f_{Y|X}(y|x) \log \left(\frac{r(x|y)}{f_X(x)} \right)$$

First Step: For fixed $f_X(x)$ and $f_{Y|X}(y|x)$, the optimal $r^*(x|y)$ is:

$$r^*(x|y) = \frac{f_X(x) f_{Y|X}(y|x)}{\sum_{x'} f_X(x') f_{Y|X}(y|x')}$$

Second Step: For fixed $r(x|y)$ and $f_{Y|X}(y|x)$, the input distribution $f_X(x)$ is updated as:

$$f_X(x) = \frac{\exp\left(\sum_{y \in Y} f_{Y|X}(y|x) \log r(x|y)\right)}{\sum_{x'} \exp\left(\sum_{y \in Y} f_{Y|X}(y|x') \log r(x'|y)\right)}$$

The algorithm iterates between these two steps until convergence is achieved.

3 Implementation Details

The implementation follows the Blahut-Arimoto algorithm steps:

3.1 Initialization

We start with a uniform input distribution:

$$P_X(x) = \frac{1}{|X|}, \quad \forall x \in X$$

In the code:

```
P_X = np.full(num_inputs, 1 / num_inputs)
```

3.2 First Step: Computing $r(x|y)$

For each x and y, $r(x|y)$ is computed as:

$$r(x|y) = \frac{P_X(x) \cdot f_{Y|X}(y|x)}{\sum_{x'} P_X(x') \cdot f_{Y|X}(y|x')} = \frac{P_X(x) \cdot f_{Y|X}(y|x)}{Q(y)}$$

where $Q(y)$ is the output distribution:

$$Q(y) = \sum_{x} P_X(x) f_{Y|X}(y|x)$$

In the code:

```
Q = P_X @ channel_matrix
```

3.3 Second Step: Updating $P_X(x)$

The input distribution is updated using the divergence $D(x)$:

$$D(x) = \sum_{y \in Y} f_{Y|X}(y|x) \left[\log f_{Y|X}(y|x) - \log Q(y) \right]$$

Then, the updated input distribution is:

$$P_X^{\text{new}}(x) = \frac{P_X(x) \cdot e^{D(x)}}{\sum_{x'} P_X(x') \cdot e^{D(x')}}$$

In the code:

```
divergence = np.zeros(num_inputs)
for x in range(num_inputs):
    for y in range(num_outputs):
        if channel_matrix[x, y] > 0 and Q[y] > 0:
            divergence[x] += channel_matrix[x, y] * (
                log_channel_matrix[x, y] - np.log(Q[y])
            )

P_X_new = P_X * np.exp(divergence)
P_X_new /= np.sum(P_X_new)
```

3.4 Convergence Check

We check if the input distribution has converged by calculating the L1 norm:

$$\|P_X^{\text{new}} - P_X\|_1 < \epsilon$$

In the code:

```
if np.linalg.norm(P_X_new - P_X, ord=1) < epsilon:
    P_X = P_X_new
    break
```

3.5 Capacity Calculation

After convergence, the capacity C is computed as:

$$C = \sum_{x \in X} P_X(x) \sum_{y \in Y} f_{Y|X}(y|x) \left[\log f_{Y|X}(y|x) - \log Q(y)\right]$$

Converted to bits:

$$C_{\text{bits}} = \frac{C}{\ln 2}$$

In the code:

```
capacity = 0.0
for x in range(num_inputs):
    for y in range(num_outputs):
        if channel_matrix[x, y] > 0 and Q[y] > 0:
            capacity += P_X[x] * channel_matrix[x, y] * (
                np.log(channel_matrix[x, y]) - np.log(Q[y])
            )

capacity /= np.log(2)
```

4 Results

The capacities for channels A, B, C, D, and E are computed using the implemented algorithm.

Blahut-Arimoto Algorithm Results

- **Capacity of Channel A**: 0.0290 bits
- **Capacity of Channel B**: 0.5037 bits
- **Capacity of Channel C**: 0.6418 bits
- **Capacity of Channel D**: 0.6322 bits
- **Capacity of Channel E**: 0.5088 bits