The Arimoto-Blahut Algorithm for Finding Channel Capacity

6.441 Supplementary Notes 4, 3/8/94

Consider a discrete memoryless channel with input alphabet $\{0,1,...,K\text{-}1\}$, output alphabet $\{0,1,...,J\text{-}1\}$ and transition probabilities $P_{jk} = P(y{=}j|x{=}k)$, $0 \le k \le K\text{-}1$, $0 \le j \le J\text{-}1$. The average mutual information on the channel for a given input probability assignment $Q = \{ Q_0, Q_1,...Q_{K-1}\}$ is given by

$$I(Q) = \sum_{k,j} Q_k P_{jk} \ln \frac{P_{jk}}{\sum_i Q_i P_{ji}} \qquad \text{nats} \qquad (1)$$

The Arimoto-Blahut algorithm is an iterative algorithm for finding capacity, $C = \max I(Q)$, where the maximum is performed over all probability assignments Q. Note that we are using natural logarithms for analytic convenience; this should be multiplied by $\log_2 e$ for capacity (or mutual information) in bits.

Mutual information can also be written as the log of the aposteriori probability of an input given an output divided by the input probability, i.e.

$$I(Q) = \sum_{k,j} Q_k P_{jk} \ln \frac{W_{kj}}{Q_k} \ ; \qquad (2)$$

where

$$W_{kj} = \frac{Q_k P_{jk}}{\sum_i Q_i P_{ji}} \qquad (3)$$

In the Arimoto-Blahut algorithm, we want to let $W = \{W_{00}, W_{01}, \ldots W_{10}, W_{11}, \ldots W_{k-1,J-1}\}$ be an arbitrary conditional probability assignment on inputs given output (i.e., $\Sigma_k W_{kj} = 1$ for all j) rather than being defined by (3) as a function of Q. Thus, we define

$$I(Q,W) = \sum_{kJ} Q_k P_{jk} \ln \frac{W_{kj}}{Q_k} \tag{4}$$

and note that $I(Q,W) = I(Q)$ when W is chosen to satisfy (3).

Lemma: $\max_W I(Q,W) = I(Q)$ (5)

where the maximum is taken over W such that $W_{kj} \geq 0$ (all k,j) $\Sigma_k W_{kj} = 1$ (all j); the maximum occurs where (3) is satisfied.

Proof: The lemma can be proved in two ways: first, from (4) and (1),

$$I(Q,W) - I(Q) = \sum_{kJ} Q_k P_{jk} \ln \frac{W_{kj} \sum_i Q_i P_{ji}}{Q_k P_{jk}} \tag{6}$$

The numerator in the log term is a probability assignment, so the overall expression is the negative of a divergence, and thus is at most 0 with equality where (3) is satisfied. Alternatively, one can recognize that I(Q,W) is convex $\cap$ in W and use Theorem 4.4.1 in the text.

With the lemma, we now see that

$$C = \max_Q I(Q) = \max_{Q,W} I(Q,W) \tag{7}$$

The Arimoto-Blahut algorithm performs the maximization in (7) by alternating between maximizing over Q and W. More precisely, it starts with an arbitrary probability vector Q^0, with $Q_k^0 > 0$ for $0 \leq k \leq K-1$, and starts with $n = 0$. The body of the algorithm is then

a) $$W^n_{kj} = \frac{Q^n_k P_{jk}}{\sum_i Q^n_i P_{ji}} \tag{8}$$

b) Find Q^{n+1} to maximize $I(Q,W^n)$ over Q.

c) Increment n and goto step a.

The algorithm needs to be tidied up in three ways. First, how do we do the maximization in step b? Second, when do we stop iterating? Third, how can we be sure that $I(Q^n,W^n)$ approaches C?

To answer the first question, note that $Q_k \ln(1/Q_k)$ is convex $\cap$ in Q_k, and thus, from (4), I(Q,W) is convex $\cap$ in Q.

$$\frac{\partial I(Q,W^n)}{\partial Q_k} = \sum_j P_{jk} \ln \frac{W^n_{kj}}{Q_k} - 1$$

From theorem 4.4.1, we know that all these partial derivatives are equal to some constant λ if all the optimizing Q_k are positive. Assuming this positivity for the moment, we have

$$\ln Q^{n+1}_k = \sum_j P_{jk} \ln W^n_{kj} - 1 - \lambda$$

Solving for Q^{n+1}_k in terms of λ, we have

$$Q^{n+1}_k = \alpha^n_k \exp(-1-\lambda)$$

where we define

$$\alpha^n_k = \exp \sum_j P_{jk} \ln W^n_{kj} \tag{9}$$

Choosing λ to satisfy $\Sigma\, Q^{n+1}_k = 1$,

$$Q_k^{n+1} = \frac{\alpha_k^n}{\sum_i \alpha_i^n} \tag{10}$$

Note that since Q^0 was chosen with all components positive, we must have $W_{kj}^0 > 0$ for all k,j such that $P_{jk} > 0$, and this guarantees that $Q_k^1 > 0$ for all k. It follows similarly that $Q_k^n > 0$ for all k and all n. It is possible for $\lim_{n\to\infty} Q_k^n$ to be 0, but it is the positivity of Q_k^n that allows the (relatively) simple solution in (9) and (10). If (9) and (10) are substituted into the expression for $I(Q^{n+1}, W^n)$ (i.e. (4)), some algebraic manipulation and some surprising cancellation of terms yields

$$I(Q^{n+1}, W^n) = \ln \sum_k \alpha_k^n \tag{11}$$

It is convenient, before discussing termination or convergence of the algorithm, to express Q^{n+1} directly in terms of Q^n. Using (8) for W^n, we have

$$\alpha_k^n = \exp \sum_j P_{jk} \ln \frac{Q_k^n P_{jk}}{\sum_i Q_i^n P_{ji}} \tag{12}$$

$$Q_k^{n+1} = \frac{\alpha_k^n}{\sum_i \alpha_i} \tag{13}$$

From (12), we also get the relation

$$\ln \frac{\alpha_k^n}{Q_k^n} = \sum_j P_{jk} \ln \frac{P_{jk}}{\sum_i Q_i^n P_{ji}} \tag{14}$$

Note that the right hand side of (14) is the mutual information between input k and the output, averaged over the outputs. As shown in problem 4.17, if Q^*_k is the probability assignment yielding capacity, then, for any Q^n,

$$C \le \sum_k Q^*_k \sum_j P_{j,k} \ln \frac{P_{jk}}{\sum_i Q^n_i P_{ji}} = \sum_k Q^*_k \ln \frac{\alpha^n_k}{Q^n_k} \tag{15}$$

$$\le \max_k \ln \frac{\alpha^n_k}{Q^n_k}$$

By combining (11) and (15), and using (13), we have

$$C - I(Q^{n+1}, W^n) \le \sum_k Q^*_k \ln \frac{Q^{n+1}_k}{Q^n_k} \tag{16}$$

$$\le \max_k \ln \frac{Q^{n+1}_k}{Q^n_k} \tag{17}$$

Since $I(Q^n,W^n) \le I(Q^{n+1},W^n) \le I(Q^{n+1},W^{n+1}) \le C$, we can use (17) to terminate the algorithm; that is, whenever the right hand side of (17) is less than some given ε of desired accuracy, the algorithm is terminated with the assurance that $I(Q^{n+1},W^{n+1})$ is within ε of capacity.

Finally, to prove convergence of the algorithm, we sum (16) over n.

$$\sum_{n=0}^{m} (C - I(Q^{n+1}, W^n)) \le \sum_k Q^*_k \ln \frac{Q^{m+1}_k}{Q^0_k}$$

$$\le \max_k \ln 1/Q^0_k \tag{18}$$

Since this bound is independent of m, $I(Q^{n+1},W^n)$ must converge to C at least like 1/n.

In summary, the algorithm starts with n = 0, with an arbitrary positive assignment Q^0, and an arbitrary ε. Then

a) Find Q^{n+1} from (12) and (13)

b) If $\max_k \ln (Q_k^{n+1}/Q_k^n) > \varepsilon$, stop ($Q^{n+1}$ estimates the optimum Q and (11) estimates C) else increment n and goto part a.

References:

Arimoto, S., "An Algorithm for Computing the Capacity of Arbitrary DMCs", IEEE Trans. I.T., pp. 14-20, Jan. 1972.

Blahut, R., "Computation of Channel Capacity and Rate Distortion Functions", IEEE Trans. I.T., pp. 460-473, July 1972.

Acknowledgement: This note is based on an earlier (1983) note for 6.441 by Pierre Humblet.