Computing Entropy under Interval Uncertainty. II

Formulation and Analysis of the Problem, and the Corresponding Results and Algorithms

Formulation of the problem. In most practical situations, our knowledge is incomplete: there are several $(n)$ different states which are consistent with our knowledge. How can we gauge this uncertainty? A natural measure of uncertainty is the average number of binary (“yes”-“no”) questions that we need to ask to find the exact state. This idea is behind Shannon’s information theory: according to this theory, when we know the probabilities $p_1, \ldots, p_n$ of different states (for which $\sum p_i = 1$), then this average number of questions is equal to $S = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$. In information theory, this average number of question is called the amount of information.

In practice, we rarely know the exact values of the probabilities $p_i$; these probabilities come from experiments and are, therefore, only known with uncertainty. Usually, from the experiments, we can find confidence intervals $p_i = [\underline{p}_i, \overline{p}_i]$, i.e., intervals which contain the (unknown) values $p_i$. Since $p_i \geq 0$ and $\sum p_i = 1$, we must have $\underline{p}_i \geq 0$ and $\sum \underline{p}_i \leq 1 \leq \sum \overline{p}_i$. How can we estimate the amount of information under such interval uncertainty?

For different values $p_i \in p_i$, we get, in general, different values of the amount of information $S$. Since $S$ is a continuous function, the set of possible values of $S$ is an interval. So, to gauge the corresponding uncertainty, we must find the range $S = [\underline{S}, \overline{S}]$ of possible values of $S$.

Thus, we arrive at the following computational problem: given $n$ intervals $p_i = [\underline{p}_i, \overline{p}_i]$, find the range

$$S = [\underline{S}, \overline{S}] = \left\{ -\sum_{i=1}^{n} p_i \cdot \log_2(p_i) \mid p_i \in p_i \& \sum_{i=1}^{n} p_i = 1 \right\}.$$

Comment. Some of the results presented in this chapter first appeared in [353] and [355].
\textit{Computation of} $\overline{S}$ \textit{is feasible (takes polynomial time).} Since the function $S$ is concave, computation of $\overline{S}$ is feasible; see [182] and [334].

The following algorithm computes $\overline{S}$ in time $O(n \cdot \log(n))$:

- First, we sort $2n$ endpoints $p_i$ and $\overline{p}_i$ into a sequence
  \[ 0 = p(0) < p(1) < p(2) < \ldots < p(m) < p(m+1) = 1. \]

In the process of this sorting, for each $k$ from 1 to $m$, we form the sets $A_k^- = \{ i : p_i = p(k) \}$ and $A_k^+ = \{ i : \overline{p}_i = p(k) \}$.

- Then, for each $k$ from 0 to $m$, we compute the values $M_k$, $P_k$, and $n_k$ as follows:
  - We start with $M_0 = - \sum_{i=1}^{n} p_i \cdot \log_2(p_i)$, $P_0 = \sum_{i=1}^{n} \overline{p}_i$, and $n_0 = n$.
  - Once we know $M_k$, $P_k$, and $n_k$, we compute the next values of these quantities as follows:
    \[ M_{k+1} = M_k + \sum_{j \in A_{k+1}^-} p_j \cdot \log_2(p_j) - \sum_{j \in A_{k+1}^+} \overline{p}_j \cdot \log_2(\overline{p}_j); \]
    \[ P_{k+1} = P_k - \sum_{j \in A_{k+1}^-} p_j + \sum_{j \in A_{k+1}^+} \overline{p}_j; \]
    \[ n_{k+1} = n_k - \#(A_{k+1}^-) + \#(A_{k+1}^+). \]

- If $n_k = n$, we take $S_k = M_k$.
- If $n_k < n$, then we compute $p = \frac{1 - P_k}{n - n_k}$.
  - If $p \in [p(k), p(k+1)]$, then we compute
    \[ S_k = M_k - (n - n_k) \cdot p \cdot \log_2(p). \]
  - Otherwise, we ignore this $k$.
- Finally, we find the largest of these values $S_k$ as the desired bound $\overline{S}$.

\textit{Linear-time algorithm for computing} $\overline{S}$. It is possible to compute $\overline{S}$ in linear time. The corresponding algorithm is iterative. At each iteration of this algorithm we have three sets:

- the set $J^-$ of all the endpoints $p_i$ and $\overline{p}_j$ for which we already know that for the optimal vector $p$ we have, correspondingly, $p_i \neq p_i$ (for $p_i$) or $p_j = \overline{p}_j$ (for $\overline{p}_j$);
- the set $J^+$ of all the endpoints $p_i$ and $\overline{p}_j$ for which we already know that for the optimal vector $p$ we have, correspondingly, $p_i = p_i$ (for $p_i$) or $p_j \neq \overline{p}_j$ (for $\overline{p}_j$);
- the set $J$ of the endpoints $p_i$ and $\overline{p}_j$ for which we have not yet decided whether these endpoints appear in the optimal vector $p$.

In the beginning, $J^- = J^+ = \emptyset$ and $J$ is the set of all $2n$ endpoints. At each iteration we also update the values $N^- = \#(J^-)$, $N^+ = \#(J^+)$, $E^- = \sum_{p_j \in J^-} \overline{p}_j$, and $E^+ = \sum_{p \in J^+} p_i$. Initially, $N^- = N^+ = E^- = E^+ = 0$. 