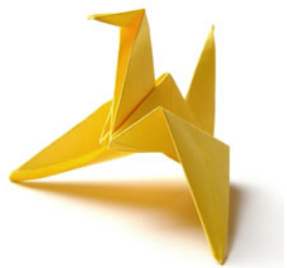
Information-Theoretic Modeling

Lecture 6: Source Coding: Practice (continued)

Teemu Roos

Department of Computer Science, University of Helsinki

Fall 2014

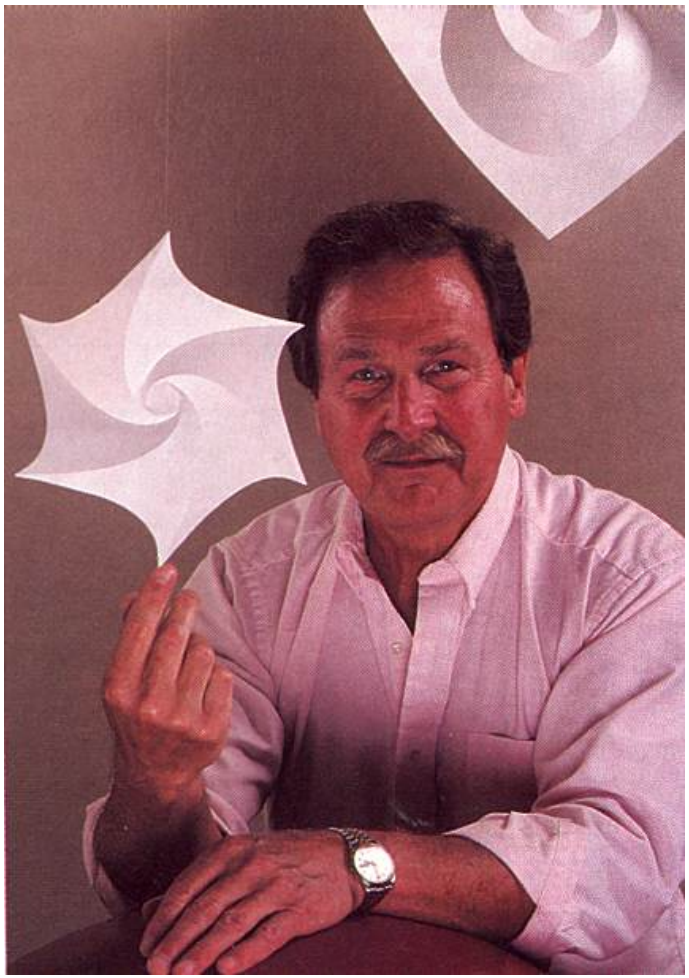


Huffman Code

Because of the rounding issue, Shannon-Fano code is not the optimal symbol code. This is where Professor Fano and a student called David Huffman enter:

"Design with the help of binary code (0 and 1) the most efficient method to represent characters, figures and symbols."

David Huffman (1925–1999)



Huffman Code: Algorithm

Huffman's algorithm proceeds as follows:

1. Sort all symbols by their probabilities p_i.
2. Join the two least probable symbols, i and j, and remove them from the list. Add a new *pseudosymbol* whose probability is $p_i + p_j$.
   Break ties arbitrarily.
3. If there is more than one symbol left, go to Step 1.
4. Use the resulting binary tree to define the codewords.

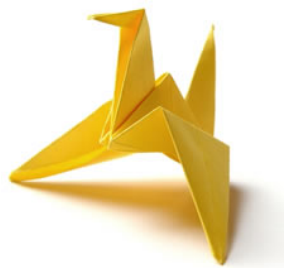
Huffman Code: Optimality

The reason why the Huffman code is the optimal symbol code (shortest expected codelength) is roughly as follows:

It can be shown that there is an optimal code (not necessarily unique) such that

1. If $p(x) > p(y)$, then $\ell(x) \leq \ell(y)$.
2. The longest two codewords have the same length.
3. The longest two codewords differ only at the last bit and correspond to the two least probable symbols.

Note that since Shannon-Fano gives $E[\ell(X)] \leq H(X) + 1$, and Huffman is optimal, Huffman must satisfy the same bound.



Problems with Symbol Codes

Now we have found the optimal symbols code with expected codelength $E[\ell(X)] \leq H(X) + 1$. Are we done?

No. (At least) three problems remain:

1. The one extra bit, $H(X) + 1$.
   - Can make all the difference if $H(X)$ is small.
2. Shannon-Fano and Huffman codes require that the distribution generating the source symbols be known.
   - We can of course first estimate the distribution from the data to be compressed, but how about the decoder?
3. Distribution is not i.i.d.: Dependence and changes.

Two-Part Codes

Solution to problem 2:

2. The Shannon-Fano and Huffman codes require that the distribution generating the source symbols be known.
   - We can of course first estimate the distribution from the data to be compressed, but how about the decoder?

Two-Part Codes

Write the distribution (or code) in the beginning of the file.

Usually the overhead is minor compared to the total file size.

Block Codes

Solution to problems 1 & 3:

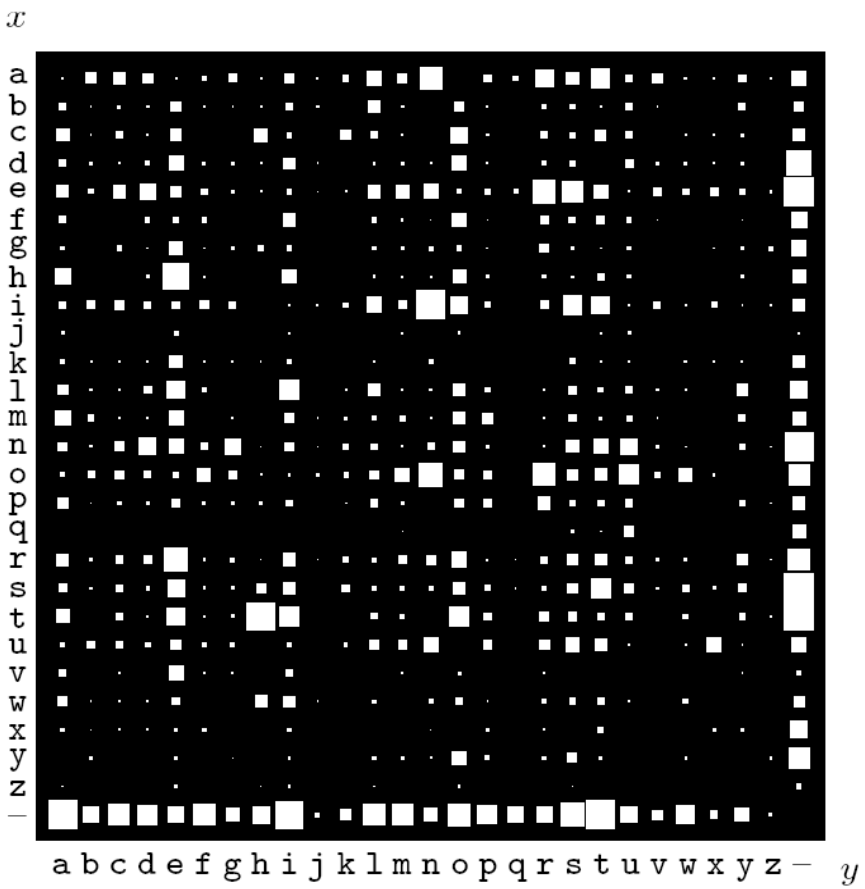
1. The one extra bit, $H(X) + 1$.
   - Can make all the difference if $H(X)$ is small.
3. Distribution is not i.i.d.: Dependence and changes.

Block Codes

Combine successive symbols into blocks and treat blocks as symbols. $\Rightarrow$ One extra bit per block.

Allows modeling of dependence.

Block Codes


x
a b c d e f g h i j k l m n o p q r s t u v w x y z –
a b c d e f g h i j k l m n o p q r s t u v w x y z –
y

Block Codes

Combining solutions to problems 1–3, we get **two-part block codes**: Write first the joint distribution of blocks of N symbols, and then encode using blocks of length N.

The size of the first part (distribution/code) grows with N, but the performance of the block code get better.

Complexity Tradeoff

Find suitable balance between complexity of the model (increases with N) and codelength of data given model (decreases with N).
$\Rightarrow$ **MDL Principle**

Adaptive Codes

Alternative Solution to Problems 2 & 3:

Adaptive Codes

For each symbol (or a block of symbols), we can construct a code based on the probability $p(x_{\text{new}} \mid x_1, \ldots, x_n)$.

This may lead to computational problems since the code tree has to be constantly updated.

Adaptive codes also avoid another problem with block codes: the first symbol can be read only after the whole block is decoded.

Arithmetic coding avoids "all problems": adaptive, spreads the one additional bit over the whole sequence, and can be decoded instantaneously.

Shannon-Fano-Elias coding

First we consider Shannon-Fano-Elias coding, based on which the basic ideas of arithmetic can more easily be understood.

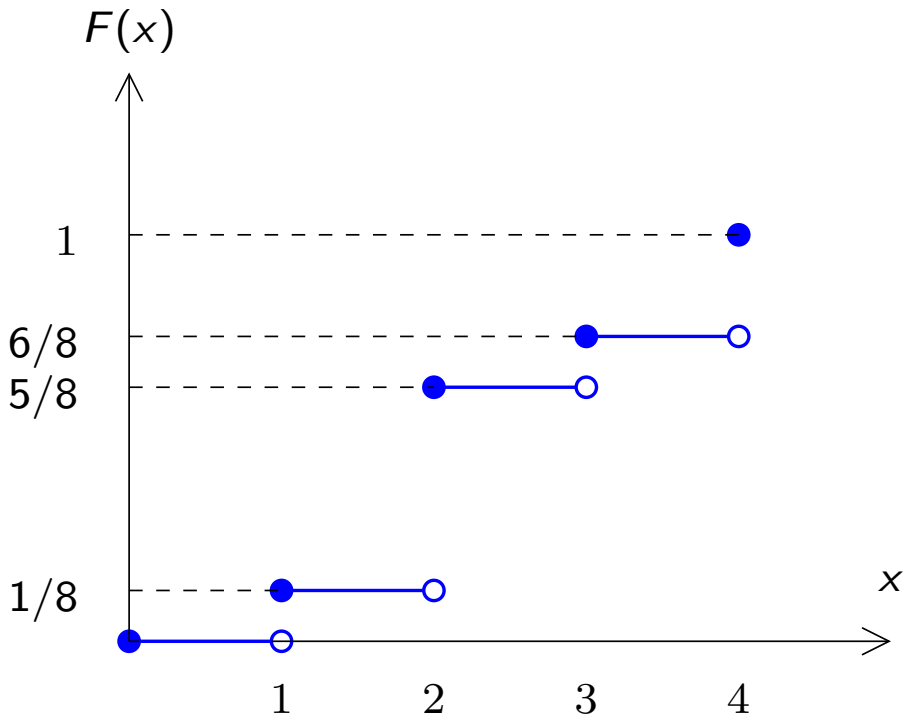
The basic idea is to assign to each symbol $x \in \mathcal{X}$ an interval $I(x) = [a(x), b(x)) \subset [0, 1)$. The intervals for different symbols are disjoint: $I(x) \cap I(y) = \emptyset$ if $x \neq y$.

The intervals are disjoint, so we can pick as code word for x any number from $I(x)$.

If $b(x) - a(x) = p(x)$, then we can always find in $I(x)$ a codeword than can be represented with $\lceil \log_2(1/p(x)) \rceil$ bits (Shannon code length).

Actually we'll need $\lceil \log_2(1/p(x)) \rceil + 1$ bits to make sure we get a *prefix* code. The extra bit is insignificant for large blocks.

Shannon-Fano-Elias: example

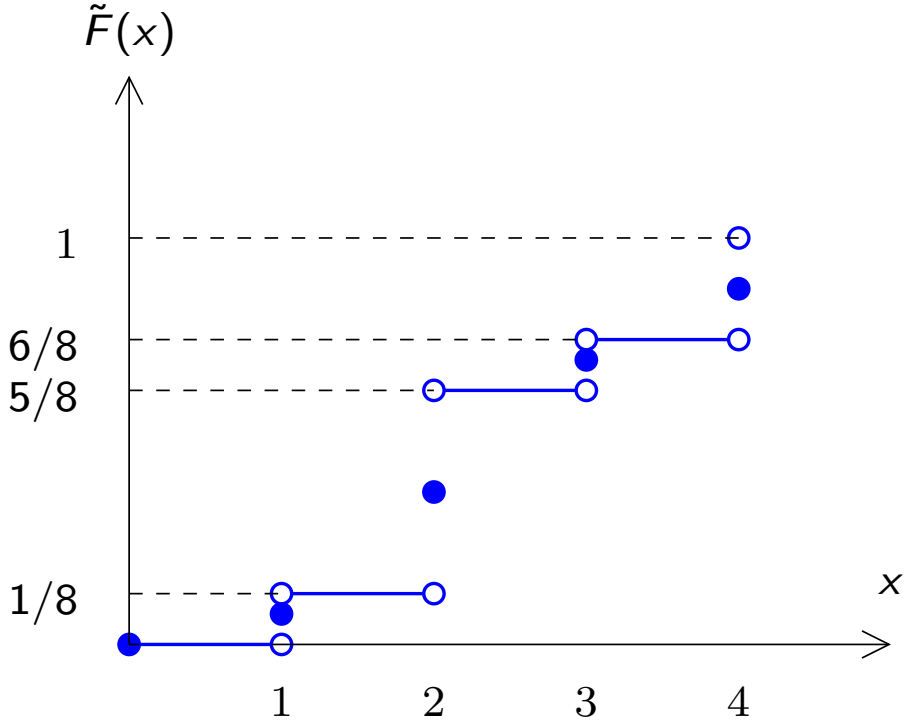


x	$p(x)$	$F(x)$	$I(x)$
1	1/8	1/8	$[0, 1/8)$
2	1/2	5/8	$[1/8, 5/8)$
3	1/8	6/8	$[5/8, 6/8)$
4	1/4	1	$[6/8, 1)$

Here $\mathcal{X} = \{1, 2, 3, 4\}$ with $p_X = (1/8, 1/2, 1/8, 1/4)$.

We use the cumulative function $F(x) = \sum_{y \leq x} p(y)$ to define the intervals $I(x)$.

Shannon-Fano-Elias: example



x	$p(x)$	$\tilde{F}(x)$
1	1/8	1/16
2	1/2	3/8
3	1/8	11/16
4	1/4	7/8

Here $\mathcal{X} = \{1, 2, 3, 4\}$ with $p_X = (1/8, 1/2, 1/8, 1/4)$.

We use the modified cumulative function $\tilde{F}(x) = \sum_{y<x} p(y) + \frac{1}{2}p(x)$ to define the code words.

Shannon-Fano-Elias coding

The value of the modified cumulative function $\tilde{F}(x)$ is in the middle of the interval $I(x)$.

As codeword $C(x)$ we pick $\tilde{F}(x)$ *rounded down to* $\ell(x)$ *bits* where $\ell(x) = \lceil \log_2(1/p(x)) \rceil + 1$.
In other words, if the binary representation of $\tilde{F}(x)$ is $0.z_1 z_2 z_3 \ldots$, we pick $C(x) = 0.z_1 \ldots z_{\ell(x)}$.

It remains to show that

1. Given $C(x)$ we can recover x.
2. The code is a prefix code.

Shannon-Fano-Elias: decoding

We first show that $C(x) \in I(x) = [F(x-1), F(x))$.

Because we round down and $p(x) > 0$, we have $C(x) \leq \tilde{F}(x) < F(x)$.

By choice of code length $\ell \geq \log_2(1/p(x)) + 1$, we have

$$2^{-\ell(x)} \leq \frac{p(x)}{2} = \tilde{F}(x) - F(x-1).$$

Because we round to ℓ bits, we have $\tilde{F}(x) - C(x) \leq 2^{-\ell(x)}$.

Combining, we get

$$C(x) \geq \tilde{F}(x) - 2^{-\ell(x)} \geq F(x-1).$$

Shannon-Fano-Elias: prefix property

Recall that $C(x)$ has $\ell(x)$ bits. Let $C^+(x)$ be the number we get if we increment the last bit by one: $C^+(x) = C(x) + 2^{-\ell(x)}$.

Now $C(x)$ is a prefix of a (binary representation of) a real number z, if and only if $C(x) \leq z < C^+(x)$.

To prove the prefix property, we show that the intervals $[C(x), C^+(x))$ are disjoint for all x.

Shannon-Fano-Elias: prefix property

We already saw that $C(x)$ is in the first half of the interval $I(x) = [F(x-1), F(x))$.

Because $2^{-\ell(x)} \leq p(x)/2$ and $p(x)$ is the length of the interval $I(x)$, also $C^+(x) = C(x) + 2^{-\ell(x)}$ is within $I(x)$.

So $[C(x), C^+(x))$ is contained in $I(x)$ for all x.

Because intervals $I(x)$ are disjoint for all x, so are the intervals $[C(x), C^+(x))$, which was the claim.

Arithmetic coding

Arithmetic coding can be seen as an application of Shannon-Fano-Elias coding to blocks of length b, that is, an alphabet $\mathcal{X}^b$.

Let $\mathcal{X} = \{ a_1, \ldots, a_m \}$. Shannon-Fano-Elias coding partitions $[0, 1)$ into m intervals $I(a_1), \ldots, I(a_m)$. Denoting the length of interval I by $|I|$, we have $|I(a_i)| = p(a_i)$.

Arithmetic coding is applies this partitioning process iteratively.

Arithmetic coding

Given an interval $I(a_i)$, we partition it into m subintervals $I(a_i a_1), \ldots, I(a_i a_m)$ keeping the same proportions:

$$|I(a_i a_j)| : |I(a_i a_k)| = |I(a_j)| : |I(a_k)| = p(a_j) : p(a_k).$$

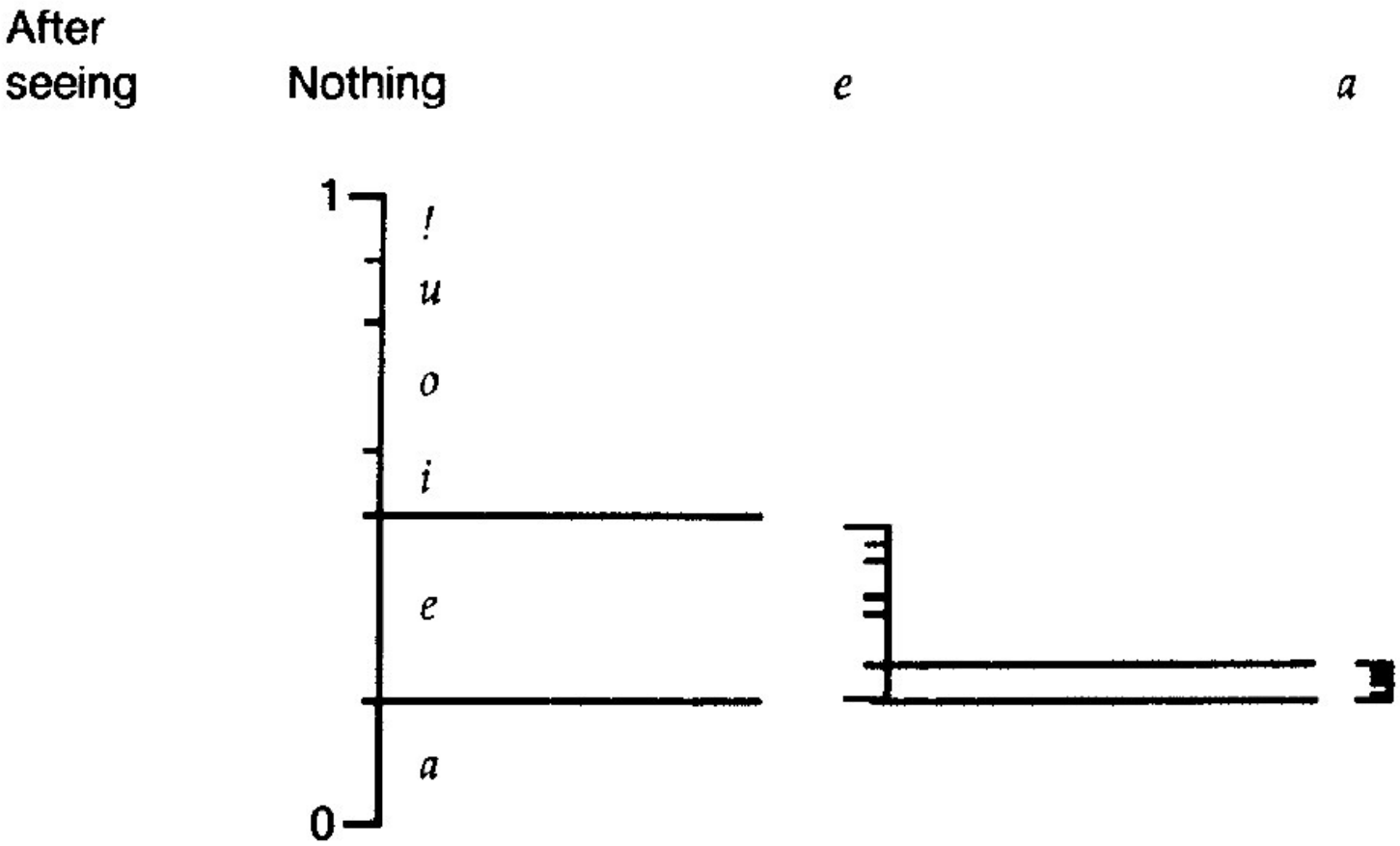
Repeating this b times, we get for all sequences $x_1 \ldots x_b \in \mathcal{X}^b$ intervals such that $|I(x_1 \ldots x_b)| = p(x_1) \ldots p(x_b)$.

Notice that we can do this incrementally one symbol at a time. We don't even need to know b in advance.

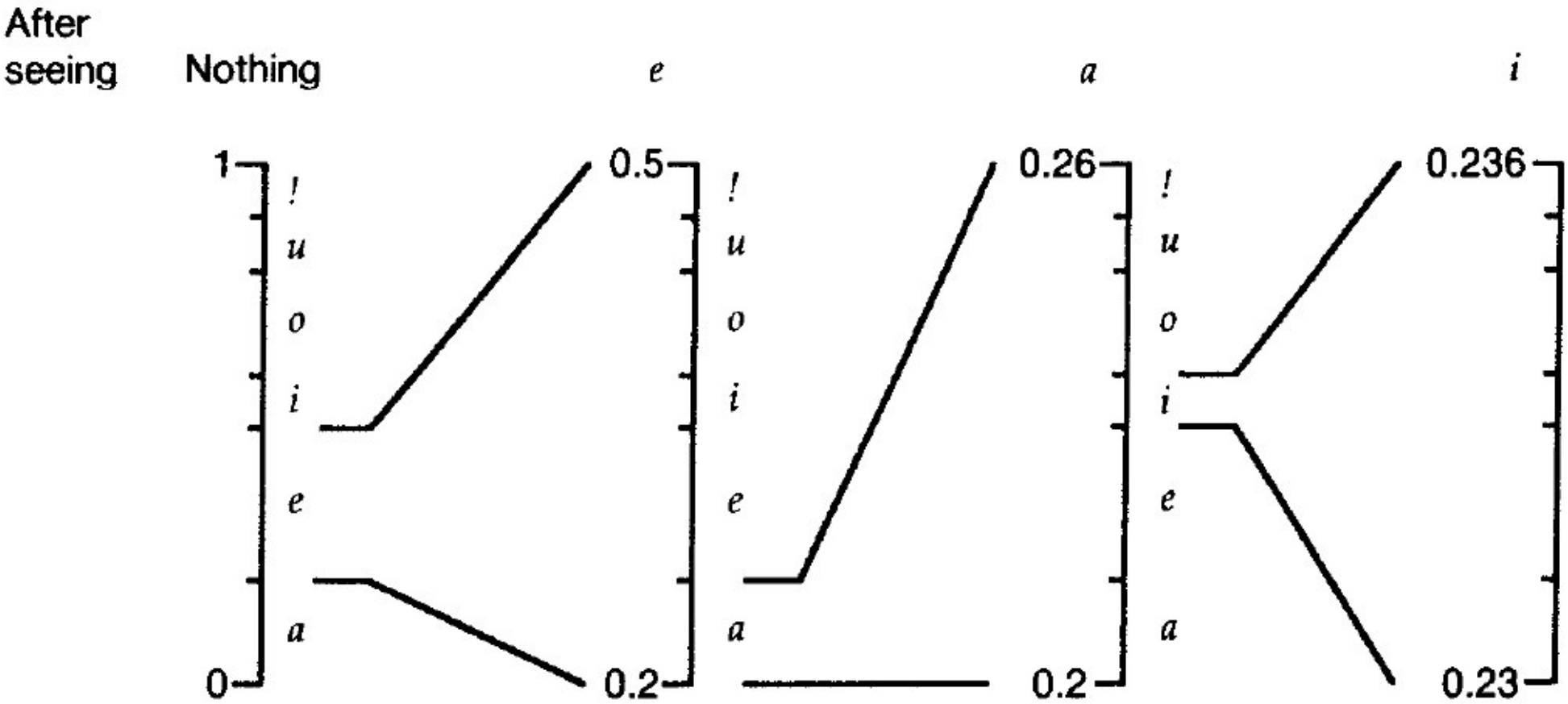
Compare this to Huffman coding, where the code book is constructed for the whole alphabet (here $\mathcal{X}^b$) at once.

This also generalises to case where we want to change the distribution dynamically.

Arithmetic coding



Arithmetic coding



Arithmetic coding

As in Shannon-Fano-Elias coding, the code word for $x_1 \dots x_b$ is one real number from $I(x_1 \dots x_b)$.

Notice that the intervals $I(x_1 \dots x_b)$ obtained by the iterative process of arithmetic coding are the same we would get by applying Shannon-Fano-Elias coding directly to the alphabet $\mathcal{X}^b$ with probabilities $p(x_1 \dots x_b) = p(x_1) \dots p(x_b)$.

(For defining the cumulative function F in $\mathcal{X}^b$, we use the ordering where $x_1 \dots x_b < y_1 \dots y_b$ if for some i we have $x_i < y_i$, and $x_j = y_j$ for $j < i$.)

Our analysis of Shannon-Fano-Elias coding implies that accuracy of $\ell(x_1 \dots x_b) = \lceil \log_2(1/(p(x_1) \dots p(x_b))) \rceil + 1$ bits is sufficient. The expected code lenght per symbol is at most $H(X) + 2/b$ bits.

Coming up

Coming next

- Universal Coding
- Minimum Description Length Principle