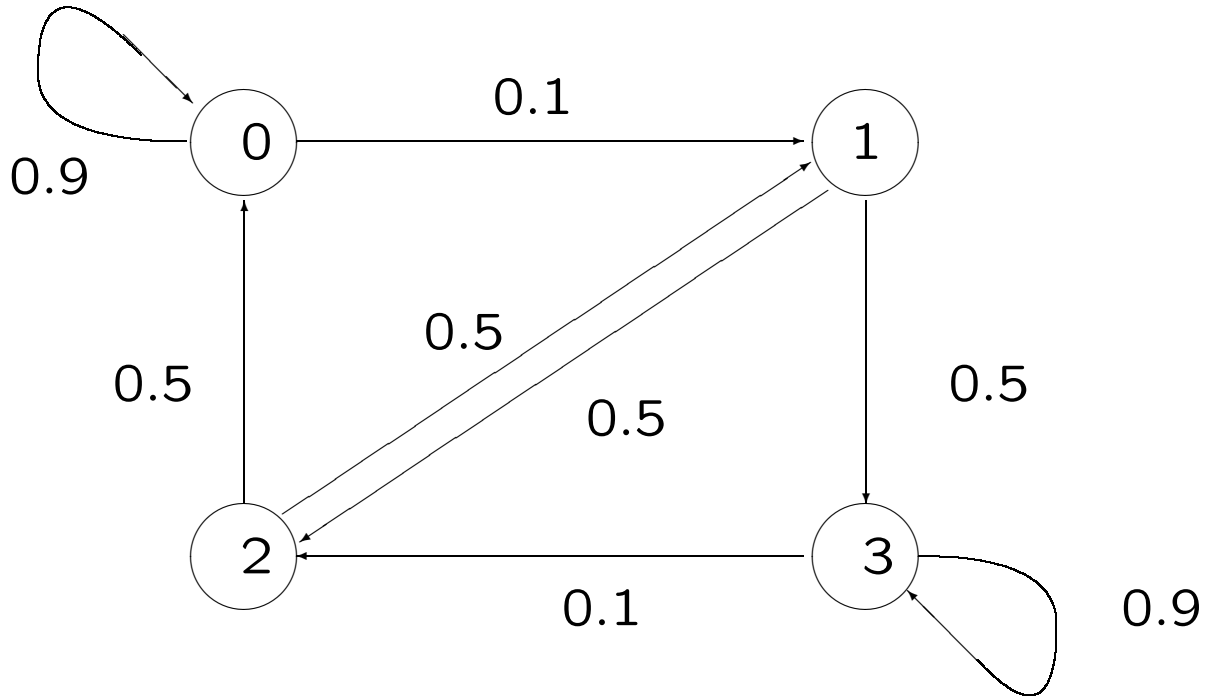
MARKOV CHAINS

A finite state Markov chain is a sequence $S_0, S_1, \ldots$ of discrete cv's from a finite alphabet $\mathcal{S}$ where $q_0(s)$ is a pmf on S_0 and for $n \geq 1$,

$$\begin{aligned} Q(s|s') &= \mathsf{Pr}(S_n{=}s|S_{n-1}{=}s') \\ &= \mathsf{Pr}(S_n{=}s|S_{n-1}{=}s', S_{n-2} = s_{n-2} \cdots, S_0{=}s_0) \end{aligned}$$

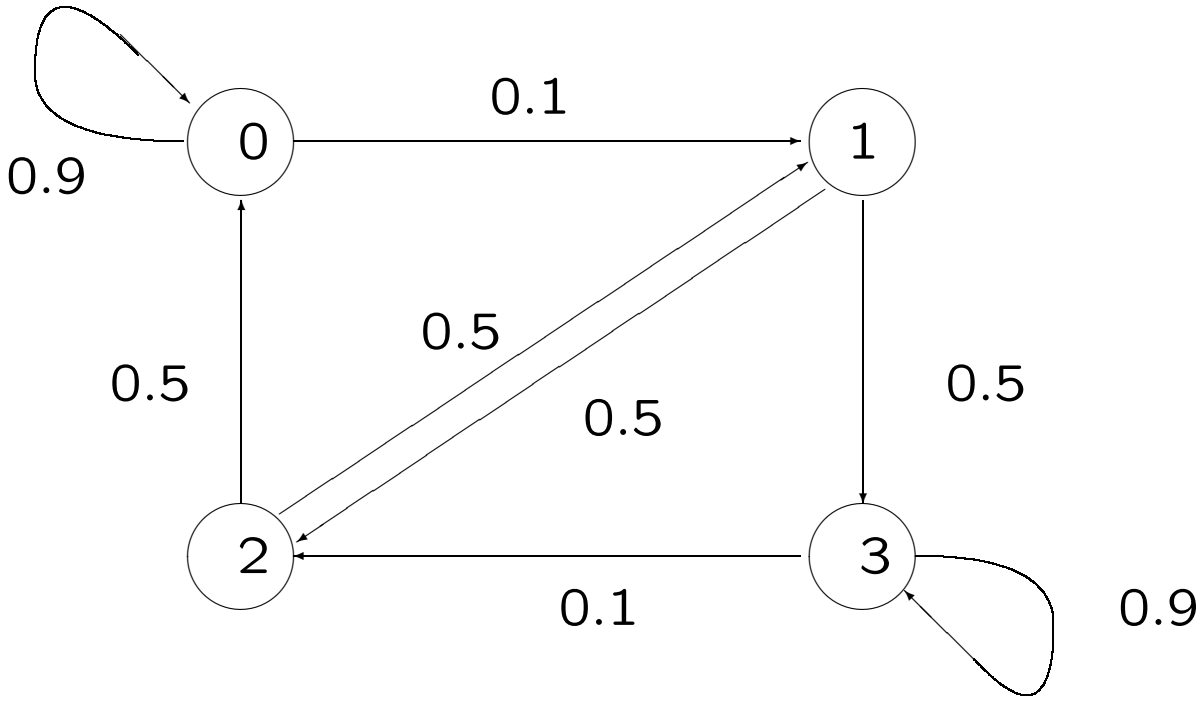
for all choices of $s_{n-2} \ldots, s_0$, We use the states to represent the memory in a discrete source with memory.

Markov chain graph has node for each state and directed edge for each transition of positive probability.



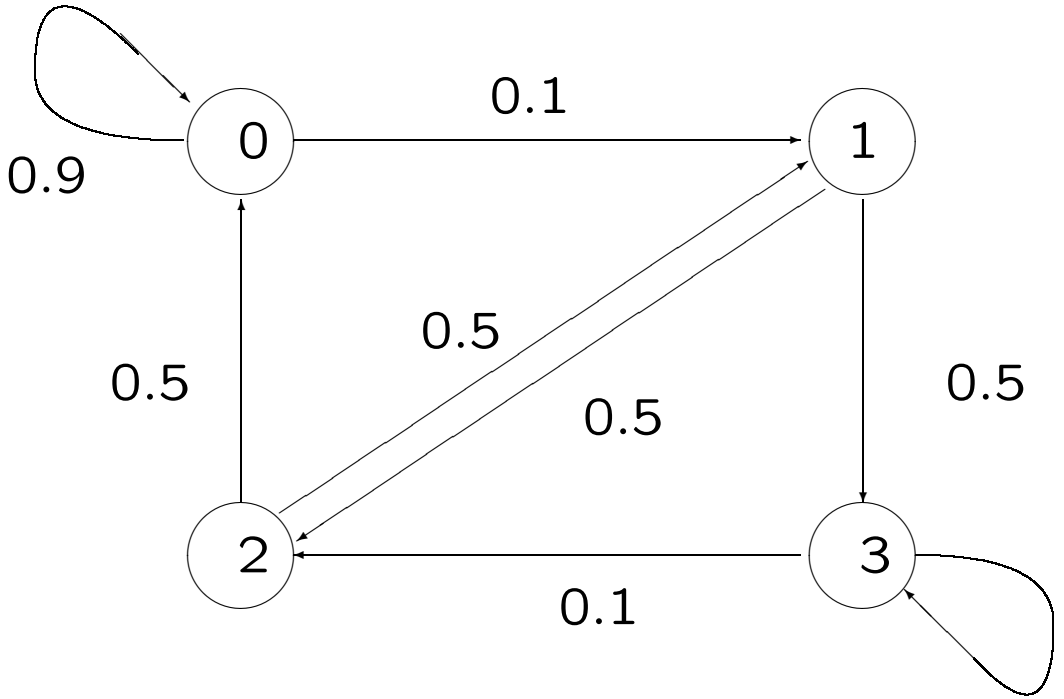
The label on an edge is the probability of that transition.

A Markov chain has period p if the states are partitioned into at most p classes, $1, 2, \cdots p$ such that transitions from each class i go only to class $i + 1$.



Period = 1 (chain is aperiodic)

A finite state Markov chain is ergodic if it is aperiodic (period 1) and each state can be reached by some path from each other state.

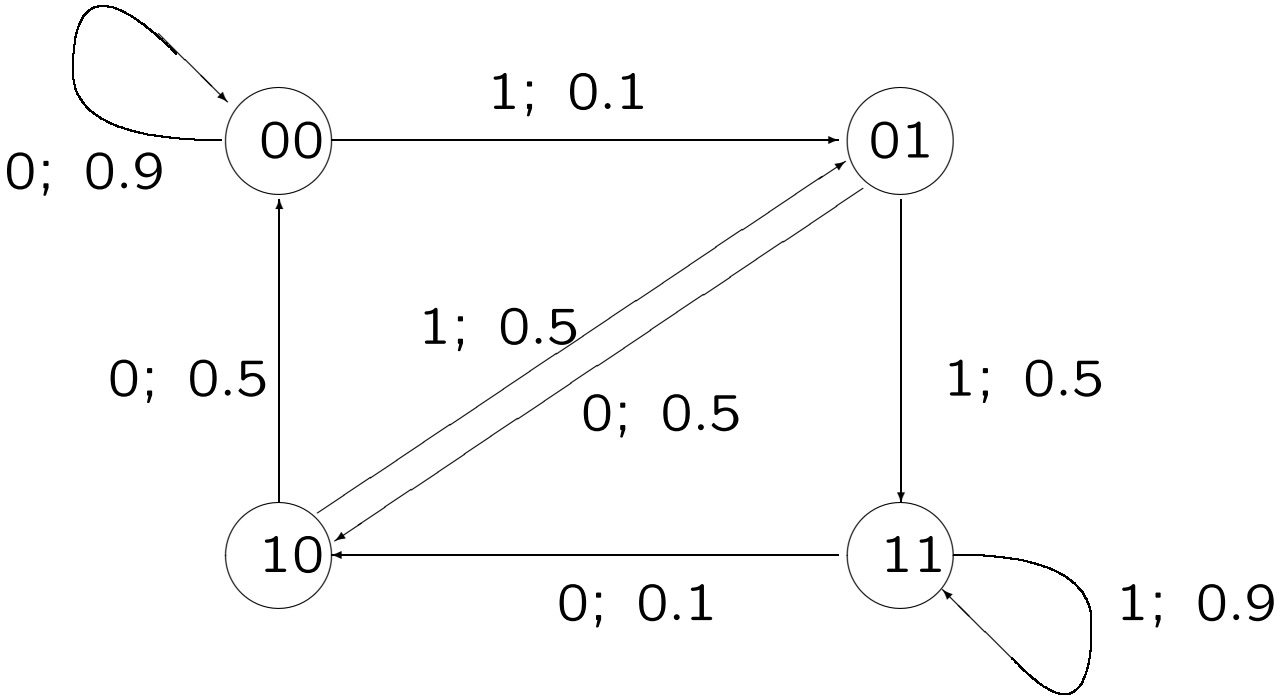


Ergodic chain

Ergodic chain has steady state probabilities

$$q(s) = \sum_{s'} q(s) \Pr(s|s') \qquad \sum_{s} q(s) = 1$$

A Markov source is a finite state Markov chain with each transition labeled by a source symbol from alphabet $\mathcal{X}$. For each state, each outgoing transition has a different label. Thus the next state and the symbol specify each other.



Coding for Markov sources

Simplest approach: use separate prefix-free code for each prior state.

If S_{n-1}=s, then encode X_n with the prefix-free code for s. The codeword lengths $l(x,s)$ are chosen for the pmf $p(x|s)$.

$$\sum_x 2^{-l(x,s)} \leq 1 \qquad \text{for each } s$$

Optimal code given prior state s satisfies:

$$\mathsf{H}[X|s] \leq \overline{L}_{min}(s) < \mathsf{H}[X|s] + 1$$

where

$$\mathsf{H}[X|s] = \sum_{x\in\mathcal{X}} -P(x|s)\log P(x|s)$$

If the pmf on S_0 is the steady state pmf, $\{q(s)\}$, then the chain remains in steady state.

$$\mathbf{H}[X|S] \leq \overline{L}_{\mathsf{min}} < \mathbf{H}[X|S] + 1, \qquad (1)$$

where

$$\overline{L}_{\mathsf{min}} = \sum_{s\in\mathcal{S}} q(s)\overline{L}_{\mathsf{min}}(s) \qquad \textbf{and}$$

$$\mathbf{H}[X|S] = \sum_{s\in\mathcal{S}} q(s)\mathbf{H}[X|s]$$

The encoder transmits s_0 followed by codeword for x_1 using code for s_0.

This specifies s_1 and x_2 is encoded with code for s_1, etc.

This is prefix free and can be decoded instantaneously.

Conditional Entropy

H$[X|S]$ **for Markov is like** **H**$[X]$ **for DMS.**

$$\mathbf{H}[X|S] = \sum_{s\in\mathcal{S}} \sum_{x\in\mathcal{X}} q(s)P(x|s) \log \frac{1}{P(x|s)}$$

Note that

$$\begin{aligned} \mathbf{H}[XS] &= \sum_{s,x} q(s)P(x|s) \log \frac{1}{q(s)P(x|s)} \\ &= \mathbf{H}[S] + \mathbf{H}[X|S] \end{aligned}$$

Recall that

$$\mathbf{H}[XS] \leq \mathbf{H}[S] + \mathbf{H}[X]$$

$$\mathbf{H}[X|S] \leq \mathbf{H}[X]$$

This is general for any random symbols.

Suppose we use 2-to-variable-length codes for each state.

$$\begin{aligned}
\mathbf{H}[X_1X_2|S_0] &= \sum_{s_0} q(s_0)\mathbf{H}[X_1X_2|s_0] \\
&= \sum_{s_0} q(s_0)\mathbf{H}[S_1S_2|s_0] \\
&= \sum_{s_0} q(s_0)(\mathbf{H}[S_1|s_0] + \mathbf{H}[S_2|S_1, s_0])
\end{aligned}$$

$$\mathbf{H}[S_2|S_1, s_0] = \sum_{s_1} Q(s_1|s_0)\mathbf{H}[S_2|s_1]$$

$$\sum_{s_0} q(s_0)\mathbf{H}[S_2|S_1, s_0] = \sum_{s_1} q(s_1)\mathbf{H}[S_2|s_1] = \mathbf{H}[S_2|S_1]$$

$$\mathbf{H}[X_1X_2|S_0] = \mathbf{H}(S_1|S_0] + \mathbf{H}(S_2|S_1] = 2H(X|S)$$

In the same way,

$$\mathbf{H}[X_1, X_2, \ldots X_n | S_0] = n\mathbf{H}[X|S]$$

By using n-to-variable length codes,

$$\mathbf{H}[X|S] \leq \overline{L}_{\mathsf{min},n} < \mathbf{H}[X|S] + 1/n$$

Thus, for Markov sources, $\mathbf{H}[X|S]$ is asymptotically achievable.

The AEP also holds for Markov sources.

$\overline{L} \leq \mathbf{H}[X|S] - \varepsilon$ can not be achieved, either in expected length or fixed length, with low probability of failure.

THE LZ77 UNIVERSAL ALGORITHM

A Univeral data compressor operates without source statistics. We describe a standard string matching algorithm for this due to Ziv and Lempel (LZ77).

In principle, a universal algorithm attempts to model the source and to encode it simultaneously.

With instantaneous decodability, the decoder knows the past also, and thus can track the encoder.

The objective (achieved by LZ77):

Given the output from a given probability model (say a Markov source);

$\overline{L}$ should be almost as small as for an algorithm designed for that model.

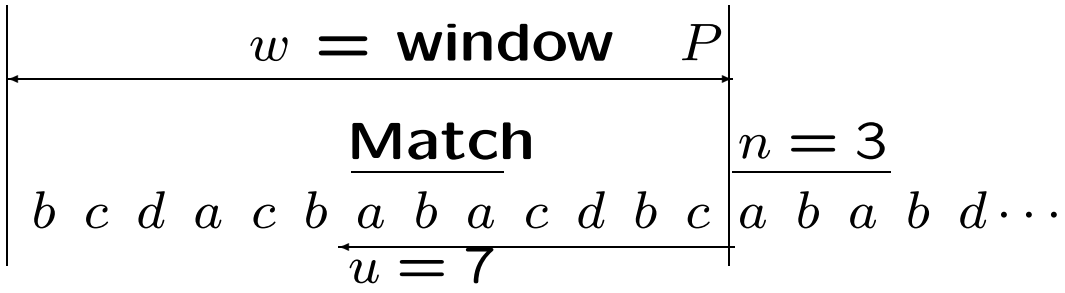
Also, the algorithm should compress well in the absence of any ordinary kind of statistical structure.

It should deal with gradually changing statistics.

Let $x_1, x_2, \ldots$ be the output of a source with known alphabet $\mathcal{X}$ of size M. Let x_m^n denote the string $x_m, x_{m+1}, \ldots, x_n$.

The window is the $w = 2^k$ most recently encoded source symbols for some k to be selected.

The Lempel-Ziv algorithm matches the longest string of yet unencoded symbols with strings starting in the window.



The compression algorithm:

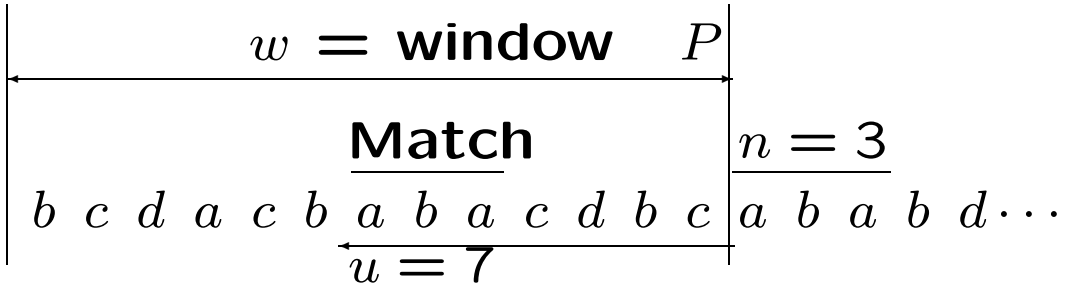
1. **Encode the first w symbols without compression, using $\lceil M \rceil$ binary digits per symbol.**

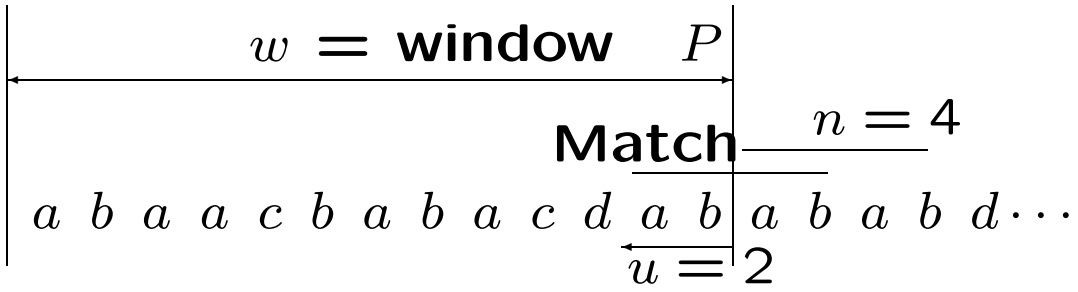
   (this gets amortized over the sequence so we don't care about it)

2. **Set the pointer $P = w$.**

   (As the algorithm runs, x_1^P is the already encoded string of source symbols and x_{P+1} is the first new symbol to be encoded.)

3. Find the largest $n \geq 2$ such that $\mathrm{x}_{P+1}^{P+n} = \mathrm{x}_{P-u+1}^{P-u+n}$ for some u, $1 \leq u \leq w$. Set $n = 1$ otherwise.





(4) Encode the match size n into a codeword from the so-called unary-binary code. The positive integer n is encoded into the binary representation of n, preceded by a prefix of $\lfloor \log_2 n \rfloor$ zeroes; i.e.,

n	prefix	base 2 exp.	codeword
1		1	1
2	0	10	010
3	0	11	011
4	00	100	00100
5	00	101	00101
6	00	110	00110
7	00	111	00110
8	000	1000	0001000

(5) If $n > 1$, encode the positive integer $u \leq w$ using a fixed-length code of length $\log w$ bits.

(At this point the decoder knows n, and can simply count back by u in the previously decoded string to find the appropriate n-tuple, even if there is overlap as above.)

If $n = 1$, encode the single letter without compression

(6) Set the pointer P to $P + n$ and go to step (2). (Iterate for ever.)

This is a variable-to-variable-length encoding.

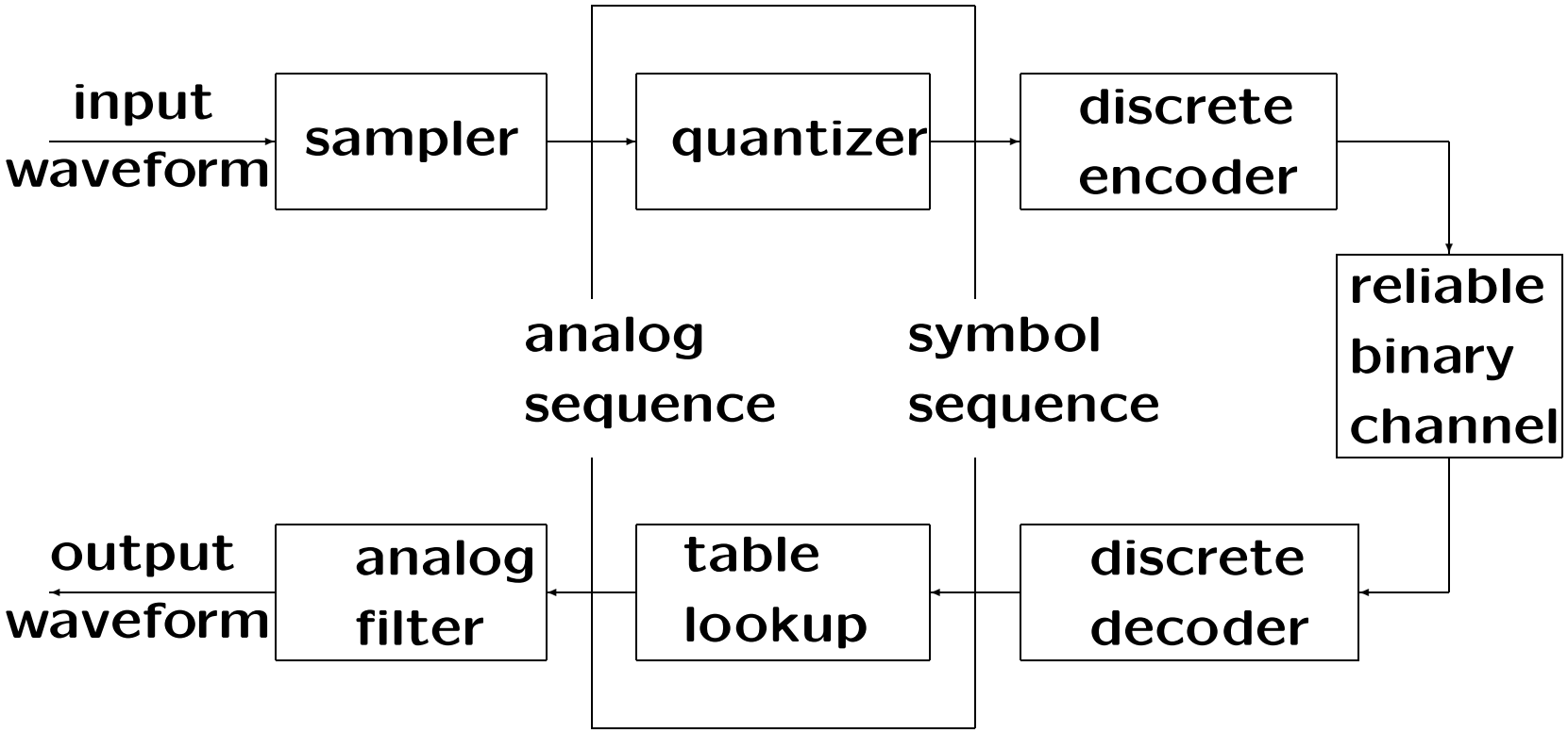
A segment of length $n > 1$ is encoded into $l(n) + \log(w)$ binary digits.

$$
\begin{array}{rll}
1 \to 1 & l(1) & = 1 \\
2 \to 010 & l(2) & = 3 \\
3 \to 011 & l(3) & = 3 \\
4 \to 00100 & l(4) & = 5 \\
5 \to 00101 & l(5) & = 5 \\
 & l(n) & = 2\lfloor \log n \rfloor + 1
\end{array}
$$

This is universal since no knowledge of source statistics is used.

Match typically occurs at $\mathbf{H}[X|S]/\log w$.

QUANTIZATION



Converting real numbers to binary strings requires a mapping from $\mathbb{R}$ to a discrete alphabet.

This is called scalar quantization.

Converting real n-tuples to binary strings requires mapping $\mathbb{R}^n$ to a discrete alphabet.

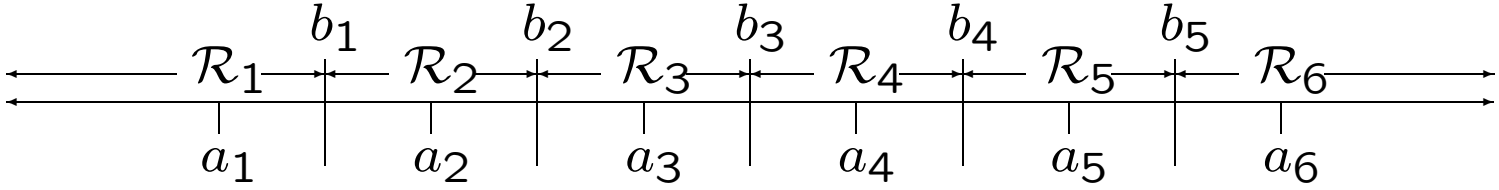
This is called vector quantization.

Scalar quantization encodes each term of the source sequence separately.

Vector quantization segments source sequence into n-blocks which are quantized together.

A scalar quantizer partitions $\mathbb{R}$ into M regions $\mathcal{R}_1, \ldots, \mathcal{R}_M$.

Each region $\mathcal{R}_j$ is mapped to a symbol a_j called the representation point for $\mathcal{R}_j$.



Each source value $u \in \mathcal{R}_j$ is mapped into the same representation point a_j.

After discrete coding and channel transmission, the receiver sees a_j and the distortion is $u - a_j$.

View the source value u as a sample value of a random variable U.

The representation a_j is a sample value of the rv V where V is the quantization of U.

That is, if $U \in \mathbb{R}_j$, then $V = a_j$.

The source sequence is $U_1, U_2, \ldots$. The representation is $V_1, V_2, \ldots$ where if $U_k \in \mathbb{R}_j$, then $V_k = a_j$.

Assume that $U_1, U_2, \ldots$ is a memoryless source which means that $U_1, U_2, \ldots$ is iid.

For a scalar quantizer, we can look at just a single U and a single V.

We are almost always interested in the mean square distortion of a scalar quantizer

$$\textbf{MSE} = \textbf{E}[(U - V)^2]$$

Interesting problem:

> **For given probability density $f_U(u)$ and given alphabet size M, choose $\{\mathbb{R}_j, 1 \leq j \leq M\}$ and $\{a_j, 1 \leq j \leq M\}$ to minimize MSE.**

Subproblem 1: Given representation points $\{a_j\}$, choose the regions $\{\mathcal{R}_j\}$ to minimize MSE.

This is easy: for source output u, squared error to a_j is $|u - a_j|^2$.

Minimize by choosing closest a_j.

Thus $\mathcal{R}_j$ is region closer to a_j than any $a_{j'}$.

$\mathcal{R}_j$ is bounded by

$$\begin{aligned} b_{j-1} &= \frac{a_j + a_{j-1}}{2} \\ b_j &= \frac{a_j + a_{j+1}}{2} \end{aligned}$$

MSE regions must be intervals.

Subproblem 2: Given interval regions $\{\mathcal{R}_j\}$, choose the representation points $\{a_j\}$ to minimize MSE.

Given $U \in \mathcal{R}_j$, the conditional density of U is $f_j(u) = f_U(u)/Q_j$ for $u \in \mathcal{R}_j$ where $Q_j = \mathsf{Pr}(U \in \mathcal{R}_j)$.

Let $U(j)$ be rv with density $f_j(u)$.

$$\mathbf{E}[|U(j) - a_j|^2] = \sigma^2_{U(j)} + |\mathbf{E}[U(j)] - a_j|^2$$

Choose $a_j = \mathbf{E}[u(j)]$.

An optimal scalar quantizer must satisfy both $b_j = (a_j + a_{j+1})/2$ and $a_j = \mathbf{E}[U(j)]$.

The Lloyd-Max algorithm:

1. **choose $a_1 < a_2 < \cdots < a_m$.**
2. **Set $b_j = (a_j + a_{j+1})/2$ for $1 \leq j \leq M - 1$.**
3. **Set $a_j = \mathbf{E}[U(j)]$ where $\mathcal{R}_j = (b_j - 1, b_j]$ for $1 \leq j \leq M - 1$.**
4. **Iterate on 2 and 3 until improvement is negligible.**

The MSE is non-negative and non-increasing with iterations, so it reaches a limit.

6.450 Principles of Digital Communication I
Fall 2009