**Information Theory and Source Coding**

# 1. Assignment

Lecture Professor: Markus Fierl

TA:Shudi Weng

Jin, Xin

11. August 2023

## Solution for Problem 1

(a) Assume there are N finite numbers for x. For each x, the g(x) is deterministic that:

$$
\begin{aligned}
&H(g(x)|x) = -\sum_{x\in\mathbf{X}} p(g(x),x)\cdot\log_2 p(g(x)|x)\\
&= -\sum_{x\in\mathbf{X}} p(g(x),x)\cdot\log_a \frac{p(g(x),x)}{p(x)})\\
&= -\sum_{x\in\mathbf{X}} p(g(x),x)\cdot\log_a \frac{p(x)}{p(x)}) && \text{g(x) is deterministic by x, p(g(x),x)=p(x)}\\
&= -\sum_{x\in\mathbf{X}} p(g(x),x)\cdot\log_a 1)\\
&= 0
\end{aligned}
$$

(b)

$$
\begin{aligned}
&H(x|g(x)) = -\sum_{x\in\mathbf{X}} p(g(x),x)\cdot\log_2 p(x|g(x))\\
&\geq 0 \quad \text{since probability } 0 \leq\text{p(x—g(x))}\leq 1, \log_2 p(x|g(x)) \leq 0
\end{aligned}
$$

(c)

$$
\begin{aligned}
&H(g(x),x) = H(x) + H(g(x)|x)\\
&= H(x) + 0\\
&= H(x)
\end{aligned}
$$

(d)

$$
\begin{aligned}
&H(g(x),x) = H(g(x)) + H(x|g(x))\\
&\Rightarrow H(g(x),x) \geq H(g(x)) && (b)H(x|g(x) \geq 0)
\end{aligned}
$$

(e) According to (c) and (d)

$$\begin{aligned}&H(x) = H(g(x), x) \geq H(g(x))\\ &\Rightarrow H(x) \geq H(g(x))\end{aligned}$$

The condition for the inequality(e) is that g(x) can be deterministic by x

(f)
- $g(x) = 2^x$, H(x)=H(g(x)), because g(x) is an one-to-one map then $H(x|g(x)) = 0$.
- $g(x) = \cos x$, $H(x) > H(g(x))$, because x is not a function of g(x) then $H(x|g(x)) > 0$

□

## Solution for Problem 2

(a)

$$\begin{aligned}
&H(X,Y|Z) = -\sum_{z\in\mathbf{Z}} p(z)\cdot H(X,Y|z=Z)\\
&= -\sum_{z\in\mathbf{Z}} p(z)\cdot \sum_{x\in\mathbf{X},y\in\mathbf{Y}} p(x,y|z=Z)\log p(x,y|z=Z)\\
&= -\sum_{x\in\mathbf{X},y\in\mathbf{Y},z\in\mathbf{Z}} p(x,y,z)\log p(x,y|z=Z)\\
&= -\sum_{x\in\mathbf{X},y\in\mathbf{Y},z\in\mathbf{Z}} p(x,y,z)\log \frac{p(x,y,z)p(x,z)}{p(x,z)p(z)}\\
&= -\sum_{x\in\mathbf{X},y\in\mathbf{Y},z\in\mathbf{Z}} p(x,y,z)(\log \frac{p(x,y,z)}{p(x,z)} + \log\frac{p(x,z)}{p(z)})\\
&= -\sum_{x\in\mathbf{X},y\in\mathbf{Y},z\in\mathbf{Z}} p(x,y,z)(\log \frac{p(x,y,z)}{p(x,z)}) - \sum_{x\in\mathbf{X},y\in\mathbf{Y},z\in\mathbf{Z}} p(x,y,z)(\log\frac{p(x,z)}{p(z)})\\
&= -\sum_{x\in\mathbf{X},y\in\mathbf{Y},z\in\mathbf{Z}} p(x,y,z)\log p(y|x,z) - \sum_{x\in\mathbf{X},y\in\mathbf{Y},z\in\mathbf{Z}} p(x,y,z)\log p(x|z)\\
&= H(Y|X,Z) + H(X|Z)\\
&\Rightarrow H(X,Y|Z) \geq H(X|Z) \qquad \text{The entropy } H(Y|X,Z) \geq 0
\end{aligned}$$

(b)

$$\begin{aligned}
&I(X,Y;Z) = H(X,Y) - H(X,Y|Z)\\
&\leq H(X,Y) - H(X|Z)\\
&= H(X) + H(Y|X) - H(X|Z) \qquad \text{generated by (a) } H(X,Y|Z) \geq H(X|Z)\\
&\leq H(X) - H(X|Z)\\
&= I(X;Z)
\end{aligned}$$

(c)

$$\begin{aligned}
&H(X,Y,Z) - H(X,Y) = H(X,Z) - H(X) + H(Y|X,Z) - H(Y|X)\\
&= H(X,Z) - H(X) - I(Y;Z|X)\\
&\leq H(X,Z) - H(X) \qquad (since I(Y;Z|X) \geq 0)
\end{aligned}$$

(d)

$$\begin{aligned}
&I(Z;Y|X) - I(Z;Y) + I(X;Z)\\
&= H(Z|X) - H(Z|X,Y) - (H(Z) - H(Z|Y)) + (H(Z) - H(Z|X))\\
&= H(Z|Y) - H(Z|X,Y)\\
&= I(Z;X|Y)\\
&= I(X;Z|Y)
\end{aligned}$$

□

## Solution for Problem 3

(a)

$$\begin{aligned}
&E(X) = \int XdP\\
&= \int xdP(x)\\
&\geq \int_{\delta}^{\infty} xdP(x)\\
&\geq \delta \int_{\delta}^{\infty} dP(x)\\
&= \delta P(X \geq \delta)\\
&\Rightarrow P(X \geq \delta) \leq \frac{E(X)}{\delta}
\end{aligned}$$

(b)

$$\begin{aligned}
&Pr[(Y-\mu)^2 \geq \epsilon^2] \leq \frac{E[(Y-\mu)^2]}{\epsilon^2} \qquad \text{apply } X = (Y-\mu)^2 \text{ in 3(a)}\\
&Pr[(Y-\mu)^2 \geq \epsilon^2] \leq \frac{\sigma^2}{\epsilon^2}\\
&\Rightarrow Pr[|Y-\mu| \geq \epsilon] \leq \frac{\sigma^2}{\epsilon^2}
\end{aligned}$$

(c)

$$E(\overline{Z_n}) = E(\frac{1}{n}\sum_{k=1}^{n} Z_k)$$
$$= \frac{1}{n}\sum_{k=1}^{n} E(Z_k)$$
$$= \mu$$

$$D(\overline{Z_n}) = D(\frac{1}{n}\sum_{k=1}^{n} Z_k)$$
$$= \frac{1}{n^2}\sum_{k=1}^{n} D(Z_k)$$
$$= \frac{\sigma^2}{n}$$

$$Pr[|\overline{Z_n} - \mu| \geq \epsilon] \leq \frac{\sigma^2}{n\epsilon^2} \qquad \text{apply } Y = \overline{Z_n} \text{ in 3(b)}$$

(d) Apply weak law of large numbers, $\lim\limits_{n\to\infty} \overline{Z_n} = \mu$
Assume $Z_i = -\log p(X_i)$ as the probability of random variable $X \sim p(x)$
Then, $\lim\limits_{n\to\infty} \overline{Z_n} = \lim\limits_{n\to\infty} \sum\limits_{i=1}^{n} \frac{-\log p(X_i)}{n} = \lim\limits_{n\to\infty} \frac{-\log p(X_1, X_2 \ldots X_n)}{n} = E(-\log p(X)) = H(X)$ in probability.

$\square$

## Solution for Problem 4

1

(a) Assume $f(x) = x\log x$ which is a convex and non-decreasing function. $a = \sum\limits_{i=1}^{n} a_i, b = \sum\limits_{i=1}^{n} b_i$

$$\sum_{i=1}^{n} a_i \log\frac{a_i}{b_i} = \sum_{i=1}^{n} b_i f(\frac{a_i}{b_i})$$
$$= b\sum_{i=1}^{n}\frac{b_i}{b} f(\frac{a_i}{b_i})$$

$$\begin{aligned}
&\geq bf(\sum_{i=1}^{n}\frac{b_i}{b}\frac{a_i}{b_i}) \qquad \text{the f(x) is convex, Jensen inequality}\\
&= bf(\sum_{i=1}^{n}\frac{a_i}{b})\\
&= bf(\frac{a}{b})\\
&= (\sum_{i=1}^{n}a_i)\cdot\log\frac{\sum\limits_{i=1}^{n}a_i}{\sum\limits_{i=1}^{n}b_i}
\end{aligned}$$

Equality is true, if and only if x is constant in the **strictly convex** function f(x) which is not linear at the same time, that when $\frac{a_i}{b_i}$ is constant

(b)

- $D(p||q) = \sum\limits_{x\in\mathbf{X}} p(x)\cdot\log\frac{p(x)}{q(x)}$

$$\begin{aligned}
&D(\lambda p_1 + (1-\lambda)p_2||\lambda q_1 + (1-\lambda)q_2)\\
&= \sum_{x\in\mathbf{X}}(\lambda p_1 + (1-\lambda)p_2)\cdot\log\frac{\lambda p_1 + (1-\lambda)p_2}{\lambda q_1 + (1-\lambda)q_2}\\
&\leq \sum_{x\in\mathbf{X}}[\lambda p_1\cdot\log\frac{p_1}{q_1} + (1-\lambda)p_2\log\frac{p_2}{q_2}] \qquad \text{,apply 4(a)}\\
&= \lambda\sum_{x\in\mathbf{X}}p_1\cdot\log\frac{p_1}{q_1} + (1-\lambda)\sum_{x\in\mathbf{X}}p_2\log\frac{p_2}{q_2}\\
&= \lambda D(p_1||q_1) + (1-\lambda)D(p_2||q_2)
\end{aligned}$$

- To prove:
  $H(\lambda p_1 + (1-\lambda)p_2) \geq \lambda H(p_1) + (1-\lambda)H(p_2) \qquad (0\leq\lambda\leq 1)$ Assume X be a continuous random variable with probability density function f(x), assume u(x) is an uniform distribution on X

$$\begin{aligned}
H(p) &= -\int_{\mathbf{X}} f(x)\log f(x)dx\\
&= -\int_{\mathbf{X}} f(x)\log\frac{f(x)}{u(x)}\cdot u(x)\\
&= -\int_{\mathbf{X}} f(x)\log\frac{f(x)}{u(x)} - \int_{\mathbf{X}} f(x)\log u(x)\\
&= -D(f||u) - \int_{\mathbf{X}} f(x)\log\frac{1}{|\mathbf{X}|}\\
&= \log(|\mathbf{X}|) - D(f||u)\\
&\Rightarrow log|\mathbf{X}| - H(f) = D(f||u)
\end{aligned}$$

Apply 4(a) then

$$D(\lambda f_1 + (1-\lambda)f_2||\lambda\mu + (1-\lambda)\mu) \leq \lambda D(f_1||u) + (1-\lambda)D(f_2||u)$$
$$\log|\mathbf{X}| - H(\lambda f_1 + (1-\lambda)f_2) \leq \lambda\log|\mathbf{X}| - \lambda H(f_1) + (1-\lambda)\log|\mathbf{X}| - (1-\lambda)H(f_2)$$

$$-H(\lambda f_1 + (1-\lambda) f_2) \leq -\lambda H(f_1) - (1-\lambda) H(f_2)$$
$$H(\lambda f_1 + (1-\lambda) f_2) \geq \lambda H(f_1) + (1-\lambda) H(f_2)$$

That H(f) is a concave function of f, When the random variable is discrete, the same reason can be proved.

$\square$

**Information Theory and Source Coding**

# 2. Assignment

Lecture Professor: Markus Fierl

TA:Shudi Weng

Jin, Xin

11. August 2023

## Solution for Problem 1

(a) According to the question conditions,

$$f_{X_1}(x) = \frac{1}{3}, 0.5 \leq x \leq 3.5 \tag{1}$$

$$f_{X_2}(x) = \begin{cases} = \frac{1}{3}, & x = 1 \\ = \frac{1}{3}, & x = 2 \\ = \frac{1}{3}, & x = 3 \end{cases} \tag{2}$$

$$f_{Y_1}(y) = \frac{1}{3}, 0.5 \leq y \leq 3.5 \tag{3}$$

$$f_{Y_2}(y) = \begin{cases} = \frac{2}{5}, & 0.75 \leq y \leq 1.25 \\ = \frac{2}{5}, & 1.75 \leq y \leq 2.25 \\ = \frac{2}{5}, & 2.75 \leq y \leq 3.25 \end{cases} \tag{4}$$

So that the entropy is calculated as follows:

$$\begin{aligned} H_{X_1} &= -\int_{0.5}^{3.5} f_{X_1}(x) \log_2 f_{X_1}(x) dx \\ &= -\int_{0.5}^{3.5} \frac{1}{3} \log_2 \frac{1}{3} dx \end{aligned}$$

$$= \log_2 3$$

$$H_{X_2} = -\sum_{x \in X_2} f_{X_2}(x) \log_2 f_{X_2}(x)$$
$$= -\sum_{x \in X_2} \frac{1}{3} \log_2 \frac{1}{3}$$
$$= \log_2 3$$

$$H_{Y_1} = -\int_{0.5}^{3.5} f_{Y_1}(y) \log_2 f_{Y_1}(y) dy$$
$$= -\int_{0.5}^{3.5} \frac{1}{3} \log_2 \frac{1}{3} dy$$
$$= \log_2 3$$

$$H_{Y_2} = -\int_{Y_2} f_{Y_2}(y) \log_2 f_{Y_2}(y) dy$$
$$= -\int_{0.75}^{1.25} \frac{2}{3} \log_2 \frac{2}{3} dy - \int_{1.75}^{2.25} \frac{2}{3} \log_2 \frac{2}{3} dy - \int_{2.75}^{3.25} \frac{2}{3} \log_2 \frac{2}{3} dy$$
$$= \log_2 \frac{3}{2}$$

(b)

$$f_{X_1,X_2}(x) = \begin{cases} = \frac{1}{3}, & 0.5 \leq x_1 \leq 1.5, x_2 = 1 \\ = \frac{1}{3}, & 1.5 \leq x_1 \leq 2.5, x_2 = 2 \\ = \frac{1}{3}, & 2.5 \leq x_1 \leq 3.5, x_2 = 3 \end{cases} \tag{5}$$

$$H(X_1, X_2) = \log_2 3 = H_{X_1} = H_{X_2} \tag{6}$$

$$p(x_2|x_1) = 1 \tag{7}$$

$$H(X_2|X_1) = \int_{X_1} p(x_1) H(X_2|x_1 = X_1) dx_1$$
$$= \int_{X_1} p(x_1) \int_{X_2} p(x_2|x_1) - \log_2 p(x_2|x_1) dx_2 dx_1$$

$$
\begin{aligned}
&= \int\limits_{X_1}\int\limits_{X_2} p(x_1,x_2)\log_2 \frac{1}{p(x_2|x_1)} dx_2 dx_1 \\
&= 0
\end{aligned}
$$

$X_2$ is deterministic by $X_1$, so the information is 0 at the time.

$$H(X_1|X_2) = H(X_1,X_2) - H_{X_2} = 0$$

(c)

$$
f_{Y_1,Y_2}(y) = \begin{cases} = \frac{2}{3}, & 0.5 \le y_1 \le 1.5, 0.75 \le y_2 \le 1.25 \\ = \frac{2}{3}, & 1.5 \le y_1 \le 2.5, 1.75 \le y_2 \le 2.25 \\ = \frac{2}{3}, & 2.5 \le y_1 \le 3.5, 2.75 \le y_2 \le 3.25 \end{cases} \tag{8}
$$

$$
\begin{aligned}
H(Y_1,Y_2) &= \int\limits_{Y_1,Y_2} \frac{2}{3} - \log_2 \frac{2}{3} dy_1 dy_2 \\
&= \log_2 \frac{3}{2}
\end{aligned}
$$

$$H(Y_2|Y_1) = H(Y_1,Y_2) - H(Y_1) = \log_2 \frac{1}{2} = -1$$

$$H(Y_1|Y_2) = H(Y_1,Y_2) - H(Y_2) = 0$$

(d)

$$I(X_1;X_2) = H(X_1) - H(X_1|X_2) = \log_2 3$$

$$I(Y_1;Y_2) = H(Y_1) - H(Y_1|Y_2) = \log_2 3$$

$\square$

## Solution for Problem 2

(a) Assume $\mu$ is the uniform distribution from a finite discrete set of size N

$$\begin{aligned}
&D_{KL}(f(x)||\mu) = \sum^{N} f(x_i) \log \frac{f(x_i)}{\mu(x_i)} \\
&= -H_{f(x)} + logN \geq 0 \\
&\Rightarrow H_{f(x)} \leq \log N
\end{aligned}$$

The maximum entropy distribution of f(x) is the uniform distribution, $\log N$ is the maximum entropy.

(b) The constraint of the question is:

$$\sum_{n=0}^{\infty} np(n) = \alpha$$

Because $D(p||q) \geq 0$,that

$$D(p||q) = \sum p(x) \log \frac{p(x)}{q(x)} \geq 0$$

We can obtain the Gibbs equation:

$$-\sum p(x) \log p(x) \leq -\sum p(x) \log q(x)$$

Such that we assume $q_i(x) = A\beta^i$

$$\begin{aligned}
&-\sum p(x) \log p(x) \leq -\sum p(x) \log A\beta^i \\
&\Rightarrow -\sum p(i) \log p(i) \leq -\sum p(i) \log A\beta^i \\
&-\sum p(i) \log p(i) \leq -\sum p(i) \log A - \sum p(i) \cdot i \log \beta \\
&-\sum p(i) \log p(i) \leq -\log A - a \log \beta
\end{aligned}$$

And because

$$\sum q_(i) = 1$$

that

$$\begin{aligned}
&A \sum \beta^i = 1 \\
&A \frac{1}{1-\beta} = 1 \\
&\beta = 1 - A \qquad \textbf{equaion b.1}
\end{aligned}$$

And the constraint of expected value:

$$\begin{aligned}
&\sum iA\beta^i = \alpha = A \frac{\beta}{(1-\beta)^2} \\
&\beta = \frac{\alpha}{\alpha+1} \qquad \textbf{combine equation b.1} \\
&A = \frac{1}{\alpha+1}
\end{aligned}$$

Such that the maximum entropy distribution is $p(n) = \frac{1}{\alpha+1}(\frac{\alpha}{\alpha+1})^n$, and the maximum entropy is $(\alpha+1)\log(\alpha+1) - \alpha \log \alpha$

(c) Assume there are N finite intervals such that the PDF f(x), and $\mu(x) = \frac{1}{L}$ as the uniform distribution on the above intervals:

$$f(x) = f_i(x) \qquad , a_i \leq x \leq b_i$$

$$\sum_{i=1}^{N} l[a_i, b_i] = L$$

$$H_{f(x)} = -\sum_{i=1}^{N} \int_{a_i}^{b_i} f_i(x) \log f_i(x) dx$$

$$D(H_{f(x)} || \mu(x)) = -H_{f(x)} + \log L \geq 0$$

such that uniform distribution is the maximum entropy distribution, and the maximum entropy is $\log L$

(d) Assume f(x) as the distribution of X given in the question. The constraint of the question is:

$$\int_{\mathbf{R}^+} x f(x) = \alpha$$

Given Gibbs equality's continuous format,

$$-\int f(x) \log f(x) dx \leq -\int f(x) \log g(x) dx$$

Such that we assume $g(x) = A\beta^x (\beta < 1)$

$$-\int f(x) \log f(x) dx \leq -\int f(x) \log A\beta^x dx$$

$$-\int f(x) \log f(x) dx \leq -\log A - \int f(x) x \log \beta dx$$

$$-\int f(x) \log f(x) dx \leq -\log A - \alpha \log \beta$$

And the sum of probability is 1 that:

$$\int_{\mathbf{R}^+} g(x) = 1$$

$$\int_{\mathbf{R}^+} A\beta^x dx = 1$$

$$A(\frac{1}{\ln \beta} \beta^x)|_0^{+\infty} = 1$$

$$A(-\frac{1}{\ln \beta}) = 1$$

$$A = -\ln \beta$$

And the constraint that the expected value is $\alpha$

$$\int_{\mathbf{R}^+} x g(x) dx = \alpha$$

$$
\begin{aligned}
&A(\frac{1}{\ln\beta}x\cdot\beta^x-\frac{1}{\ln\beta}^2\cdot\beta^x)|_0^{+\infty}=\alpha\\
&\beta=e^{-\frac{1}{\alpha}}\\
&A=\frac{1}{\alpha}
\end{aligned}
$$

Such that

$$f(x)=\frac{1}{\alpha}e^{-\frac{x}{\alpha}}$$

is the maximum entropy distribution, and the maximum entropy is:$\log\alpha+1$

(e) According to the question, we assume $q(x)=\mathcal{N}(0,\mathbf{K})$ as a normal distribution for multi-variable X that:

$$q(x_1,x_2...)=\frac{1}{(2\pi)^{\frac{n}{2}}(det\mathbf{K})^{\frac{1}{2}}}e^{-\frac{1}{2}x^T\mathbf{K}^{-1}x} \tag{9}$$

Given Gibbs distribution,

$$0\leq-h(p(x))-\int_{\mathbf{R}^n}p(x)\log q(x)dx \tag{10}$$

that

$$
\begin{aligned}
&\int p(x)\log q(x)=\int p(x)\log(\frac{1}{(2\pi)^{\frac{n}{2}}(det\mathbf{K}^{-1})^{\frac{1}{2}}}\cdot e^{-\frac{1}{2}x^T\mathbf{K}x})\\
&=-\frac{1}{2}\log((2\pi)^n(det\mathbf{K}))\cdot\int p(x)+\int p(x)\log(e)(-\frac{1}{2}x^T\mathbf{K}^{-1}x)\\
&=-\frac{1}{2}\log((2\pi)^n(det\mathbf{K}))+(-\frac{1}{2}\log eE(x^T\mathbf{K}^{-1}x)\\
&=-\frac{1}{2}\log((2\pi)^n(det\mathbf{K}))-\frac{1}{2}\log e\\
&=-\frac{1}{2}\log((2\pi)^n(det\mathbf{K})e)\\
&=h(q(x))
\end{aligned}
$$

Such that, combine equation (e).(10)

$$h(p(x))\leq h(q(x))$$

Then $q(x)=\mathcal{N}(0,\mathbf{K})$ is the maximum entropy distribution. And the maximum entropy is:$-\frac{1}{2}\log((2\pi)^n(det\mathbf{K})e)$

□

## Solution for Problem 3

(a) The transition matrix is:

$$T = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}$$

And the stationary condition matrix $\pi$ fulfill:

$$\pi T = \begin{bmatrix} \pi_1 & \pi_2 \end{bmatrix} \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix} \quad = \begin{bmatrix} \pi_1 & \pi_2 \end{bmatrix} = \pi$$

Such that

$$(1-\alpha)\pi_1 + \beta\pi_2 = \pi_1$$
$$\alpha\pi_1 + (1-\beta)\pi_2 = \pi_2$$

Then

$$\pi = \begin{bmatrix} \frac{\beta}{\alpha+\beta} & \frac{\alpha}{\alpha+\beta} \end{bmatrix}$$

is the stationary distribution of the source

(b) The entropy $H(X_n)$ is:

$$H(X_n) = \frac{\beta}{\alpha+\beta}\log\frac{\alpha+\beta}{\beta} + \frac{\alpha}{\alpha+\beta}\log\frac{\alpha+\beta}{\alpha}$$

The entropy rate of the stationary stochastic process is:

$$H'(X) = \lim_{n\to\infty} H(X_n|X_1...X_{n-1}) = H(X_n|X_{n-1}) = H(X_2|X_1)$$
$$= \frac{\beta}{\alpha+\beta}H(\alpha) + \frac{\alpha}{\alpha+\beta}H(\beta)$$
$$= \frac{\beta}{\alpha+\beta}(\alpha\log\frac{1}{\alpha} + (1-\alpha)\log\frac{1}{(1-\alpha)}) + \frac{\alpha}{\alpha+\beta}(\beta\log\frac{1}{\beta} + (1-\beta)\log\frac{1}{(1-\beta)})$$

The Matlab function for f(a,b) is:

```
    function [entropy] = entropy_f(a,b)
entropy=(b./(a+b))*(a*log2(1./a)+(1-a)*log2(1./(1-a)))+(a
    ./(a+b))*(b*log2(1./b)+(1-b)*log2(1./(1-b)))
%entropy=(b./(a+b))*a*log2(1./(a))
end
```

The Matlab 2D plot script for $f(\alpha,\beta)$ is:

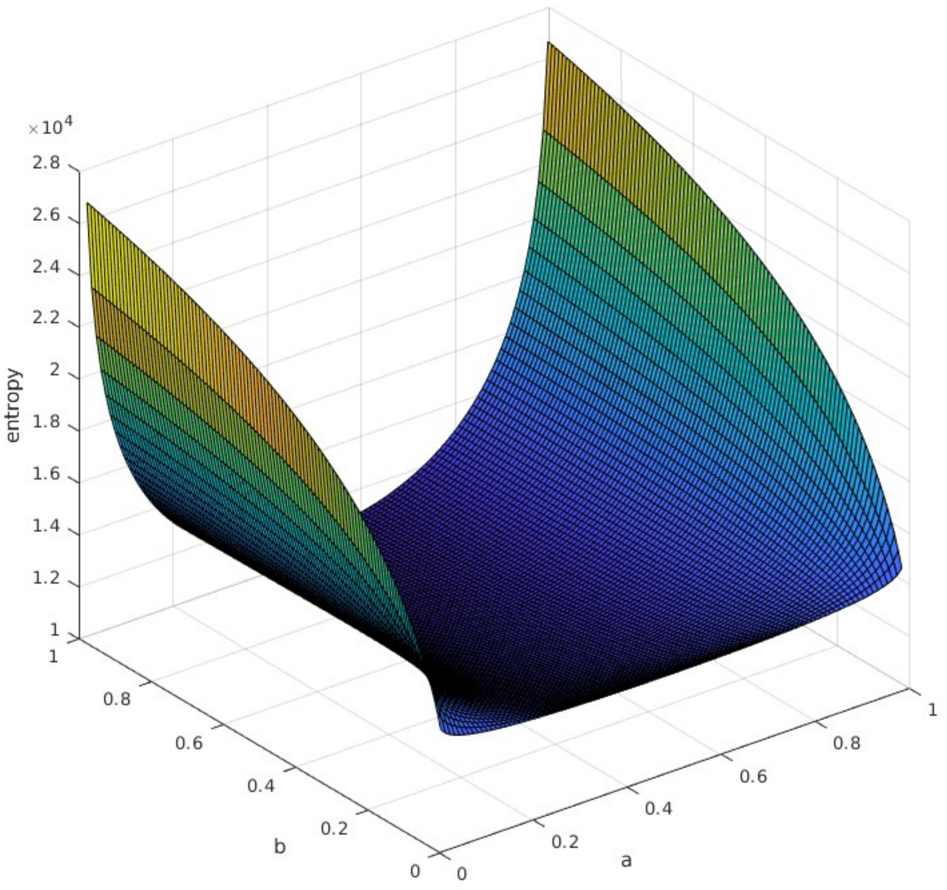


Abbildung 1: 2D plot for entropy rate f(a,b)

(c) Matlab function to generate the Markov-1 character string

```matlab
function [chain]=markov_generate(chain_length)
P=[0.8 0.2;0.2 0.8];
%chain_length;
chain = zeros(1,chain_length);
initial_value=1;
beta=0.2;
alpha=0.2;
chain(1)=initial_value;
for i=2:chain_length
    measure=rand();
    if chain(i-1)==1
        if measure<1-beta
            chain(i)=1;
        else
            chain(i)=0;
        end
    elseif chain(i-1)==0
            if measure<alpha
                chain(i)=1
            else
                chain(i)=0
            end
    end

end
```

One Markov-generated sample of length 1000 is:

```

Markov_chain = [1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0
    0 0 0 0 0 0 0 0 0 1 ...
         1 1 0 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 0 0 0 1 1
            1 0 0 0 1 1 1 ...
         1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 0 1 1 1 0 0 0
            0 0 0 0 0 0 0 ...
         1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0
            0 0 1 1 0 0 0 ...
         1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
            1 1 0 0 0 0 0 ...
         0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0
            0 0 0 0 1 1 1 ...
         1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 1 0 1 0 1 1
            1 1 1 1 1 0 0 ...
         0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0
            0 0 0 0 0 1 1 ...
```

```
1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1
   1 0 1 1 0 1 1  ...
0 0 0 0 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 1 1 0 0
   0 0 0 0 0 0 0  ...
0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 1 1 1 0 0 0 0 1
   1 1 1 1 1 1 0  ...
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   0 0 0 1 1 1 1  ...
1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
   1 1 1 1 1 0 0  ...
0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   1 0 0 1 1 0 0  ...
0 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1
   1 1 1 1 1 0 1  ...
1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1
   1 1 1 1 1 1 1  ...
1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1
   1 1 0 0 0 0 0  ...
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
   0 0 1 1 0 0 0  ...
0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0
   0 0 0 1 1 0 0  ...
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   0 1 1 0 0 0 1  ...
1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0
   1 0 0 0 1 1 1  ...
1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0
   1 1 1 1 1 1 1  ...
1 1 1 1 1 1 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0 0
   0 0 0 0 0 0 1  ...
1 0 1 1 1 1 1 0 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 1
   0 0 1 1 1 1 0  ...
0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0
   0 0 0 0 0 0 0  ...
0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1
   0 0 0 1 1 1 1  ...
1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1
   1 1 1 0 0 0 0  ...
0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 1 1 1 1 1 0
   0 1 1 1 1 1 1  ...
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
   0 0 0 0 0 0 0  ...
0 0 0 1 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 1
   1 1 0 0 0 0 0  ...
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   0 0 0 0 0 1 1  ...
```

```
        1 1 1 1 1 1 1 0 0 1 0 0 1 1 1 0 0 0 0 1 1 0 0 0
           1 0 0 0 0 0 0 ...
        0 1 1 0 0 0 0 1];
```

We want to estimate the entropy rate by:

$$H(\mathcal{X}) = \lim_{n\to\infty} \frac{H(X_1, X_2, X_3..., X_n)}{n}$$

$$\frac{\sum\limits_{n=0}\sum\limits_{i=nm} \frac{H(X_{i+1},X_{i+2},...,X_{i+m})}{m}}{n} \asymp H(\mathcal{X})$$

n is the number of partitions in the N-length Markov chain sequence data diverse by m-length group. I want to follow the steps to generate the entropy rate:

a) Generate a random sequence generated by the Markov process

b) Partition the sequence into overlapping segments of length m

c) Count the segments distribution statistically and estimate the probability

d) Calculate the entropy rate based on the estimated probability for $(X_1, ..., X_m)$

The Matlab function to calculate the entropy rate with N symbols and m length is as follows:

```matlab
function H = entropy_rate_markov_1(X, m)
%X is the data generated by the Markov chain
  n = length(X);
  X_grouped = reshape(X, m, []);
  %H_grouped = zeros(1, size(X_grouped, 2));
  H_for=0;
  i=1;
  number=size(X_grouped,2)
  for_length=size(X_grouped,2);
  while i<for_length
      counts=1
      j=i+1
      while j<for_length
          if(X_grouped(:,i)==X_grouped(:,j))
             counts=counts+1
             X_grouped(:,j)=[]
             for_length=size(X_grouped,2);
          else
              j=j+1
          end

      p = counts / number
      end
      i=i+1;
```

```
        H_for = H_for+(-p * log2(p))
    end
    H=H_for/m
end
```

The analytically calculated entropy rate is around **0.7219**.

When m=10, N=100000, the estimated entropy rate by the Matlab function entropy rate Markov-1(X,m) is **0.71561**. As the N increases the results are more approaching to the analytical one. And increasing m under the condition that $2^m$ is much smaller than N, will also improve the accuracy a

(d) Assume the initial state $(\frac{\beta}{\alpha+\beta}, \frac{\alpha}{\alpha+\beta})$ to prove and modify the stationary state that:

$$P(Y_n = 0) = P(X_n = 0) = \frac{\beta}{\alpha+\beta} \tag{11}$$

$$\begin{aligned}
&P(Y_n = k+1) && (12)\\
&= P(X_n = 1|X_{n-1} = 1, ....X_{n-k} = 1, X_{n-k-1} = 0) \cdot P(X_{n-1} = 1, ...., X_{n-k} = 1, X_{n-k-1} = 0) && (13)\\
&= (1-\beta)P(X_{n-1} = 1, X_{n-k} = 1, ..., X_{n-k-1} = 0) && (14)\\
&= (1-\beta)P(X_{n-1} = 1, ..., X_{n-k} = 1|X_{n-k-1} = 0) \cdot P(X_{n-k-1} = 0) && (15)\\
&= (1-\beta)\alpha(1-\beta)^{k-1} \cdot \frac{\beta}{\alpha+\beta} && (16)\\
&= \alpha(1-\beta)^k \frac{\beta}{\alpha+\beta} && (17)
\end{aligned}$$

Such that the entropy rate for Y

$$\begin{aligned}
H'(Y) &= \lim_{n\to\infty} H(Y_n|Y_{n-1}) = H(Y_2|Y_1)\\
&= \frac{\beta}{\alpha+\beta}H(\alpha) + \frac{\alpha}{\alpha+\beta}H(\beta)\\
&= \frac{\beta}{\alpha+\beta}(\alpha\log\frac{1}{\alpha} + (1-\alpha)\log\frac{1}{(1-\alpha)}) + \frac{\alpha}{\alpha+\beta}(\beta\log\frac{1}{\beta} + (1-\beta)\log\frac{1}{(1-\beta)})
\end{aligned}$$

□

**Information Theory and Source Coding**

# 3. Assignment

Lecture Professor: Markus Fierl

TA:Shudi Weng

Jin, Xin

11. August 2023

## Solution for Problem 1

(a) Apply the Kraft inequality, the lower bound for prefix-free code should be:

$$D^{-1} + D^{-1} + D^{-2} + D^{-2} + D^{-3} + D^{-3} \leq 1$$

That the lower bound for $D \approx 2.53 \approx 3$.(D should be integer) If we would like to design a uniquely decodable code(a prefix-free code should be a decodable code), and the be decreased on theory, and we found that D=2 can not estimate a uniquely decodable code. So the bound should be still D=3 for a uniquely decodable code.

(b) Arrange the symbols of their probabilities:$\{a, b, c, d\} = \{\frac{1}{3}, \frac{1}{3}, \frac{1}{4}, \frac{1}{12}\}$. The possible 2-ary Huffman codes for the random message:

$$\begin{aligned}
\{a, b, c, d\} &= \{0, 10, 110, 111\} \\
&= \{0, 10, 111, 110\} \\
&= \{0, 11, 100, 101\} \\
&= \{0, 11, 101, 100\} \\
&= \{1, 00, 010, 011\} \\
&= \{1, 00, 011, 010\} \\
&= \{1, 01, 001, 000\} \\
&= \{1, 01, 000, 001\}
\end{aligned}$$

The average code length is:

$$E(L) = \frac{1}{3} + \frac{2}{3} + \frac{3}{4} + \frac{3}{12} = 2$$

$$H(X) = \frac{1}{3}\log_2(3) + \frac{1}{3}\log_2 3 + \frac{1}{4}\log_2(4) + \frac{1}{12}\log_2(12) \approx 1.86$$

Such that $E(L) > (X)$, there are optimal codes with codeword lengths for some symbols that exceed the Shannon code length.

(c) Arrange symbols $\{a, b, c, d, e\} = \{0.3, 0.3, 0.2, 0.1, 0.1\}$ for the message. The binary Huffman code could be:

$$\{a, b, c, d, e\} = \{00, 01, 10, 110, 111\}$$

The average length is:

$$E(L) = 0.3 * 2 + 0.3 * 2 + 0.2 * 2 + 0.1 * 3 + 0.1 * 3 = 2.2$$

For H(X)=E(L), I assume the probabilities$\{a, b, c, d, e\} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{16}\}$ The one 2-ry Huffman code could be:

$$\{a, b, c, d, e\} = \{0, 10, 110, 1110, 1111\}$$

Then,

$$E(L) = \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \frac{4}{16} = \frac{15}{8}$$
$$H(X) = \frac{1}{2}\log_2 2 + \frac{1}{4}\log_2 4 + \frac{1}{8}\log_2 8 + \frac{1}{16}\log_2 16 + \frac{1}{16}\log_2 16 = \frac{15}{8}$$
$$E(L) = H(X)$$

(d) $l_1 = (1, 2, 2)$ can be word lengths of a binary Huffman code

$l_2 = (2, 2, 3, 3)$ can not be word lengths of a binary Huffman code.

$l_3 = (1, 2, 2, 2, 2)$ can not be word lengths of a ternary Huffman code.

$l_4 = (2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3)$ can be word lengths of a ternary Huffman code.

□

## Solution for Problem 2

(a) The following code is the Matlab function replacing a strings of 0s and 1s by run-length values. And the start value of 0 or 1 is indicated, binary2runlength.m.

```
function [rl_values ,start_value] = binary2runlength(
    stream)
%binary2runLength encodes a string of 0s and 1s using run-length encoding
%Inputs:
%stream: a string of 0s and 1s
rl_values=[]
start_value=stream(1)
n=length(stream)
count=1

for i=2:n
```

```
    if(stream(i)==stream(i-1))
        count=count+1
    else
        rl_values=[rl_values,count]
        count=1;
    end

end
rl_values=[rl_values,count]
end
```

(b) The source coding theorem shows that the optimal code for a random variable X has an expected length less than:

$$H(X) + 1$$

Such that we apply H(X) to estimate the optimal bits for encoding run-length into binary code, and generate the optimal length for the binary stream. And write Matlab function, optimalstream.m:

```
  function [optimal_length] = optimalstream(
     run_length_code,m)
%OPTIMALSTREAM Summary of this function goes here
% Detailed explanation goes here
%m is the consecutive group number
%run_code_length is the length of run length code
%optimal_length is the optimal length for run length code's binary code

run_code_length=length(run_length_code)
[length_p,H]=entropy_rate_markov(run_length_code,m)
optimal_length=ceil(length_p)+H*run_code_length

end
```

The entropy rate is constructed as, entropy-rate-markov.m:

```
function [length_p, H] = entropy_rate_markov(X, m)
  n = length(X);
  remain=mod(length(X),m)
  X(length(X)-(0:remain-1))=[]

  X_grouped = reshape(X, m, []);
  %H_grouped = zeros(1,size(X_grouped,2));
  H_for=0;
  i=1;
  length_p=0
  number=size(X_grouped,2)
  for_length=size(X_grouped,2);
  sum_p=0
  while i<for_length
      counts=1
      j=i+1
      while j<for_length
          if(X_grouped(:,i)==X_grouped(:,j))
             counts=counts+1
             X_grouped(:,j)=[]
             for_length=size(X_grouped,2);
          else
              j=j+1
          end

      p = counts / number
      end
      sum_p=sum_p+p
      i=i+1;
      length_p=length_p-log2(p)
      H_for = H_for+(-p * log2(p))
  end
  H=H_for/m
end
```

(c) Generate the run-length code by a=0.05:0.05:0.95, and apply the optimum binary encoding length from the previous question, and plot the corresponding compression ratio versus a, The Matlab code is as follows, compression.m:

```
%single_code_lengthisthelengthofeachstream'srunlengthcode
%bits_numberisthenumberofsourcestreambits
bits_number=19600
a=0.05:0.05:0.95
source_stream=[]
length_a=length(a)
code_length=[]
start_value=0
m=10
for i=1:length_a
    source_stream=markov_generate(a(i),bits_number)
    run_length_code=binary2runlength(source_stream)
    single_code_length=optimalstream(run_length_code,m)
    code_length=[code_length, single_code_length ]

end
compresion_ratio=19600./code_length
plot(a,compresion_ratio)

```

```
title('Compression ratio related to a')
xlabel('a')
ylabel('Compression ratio')
```

And the run-length binary code compression versus a plot image is as follows:

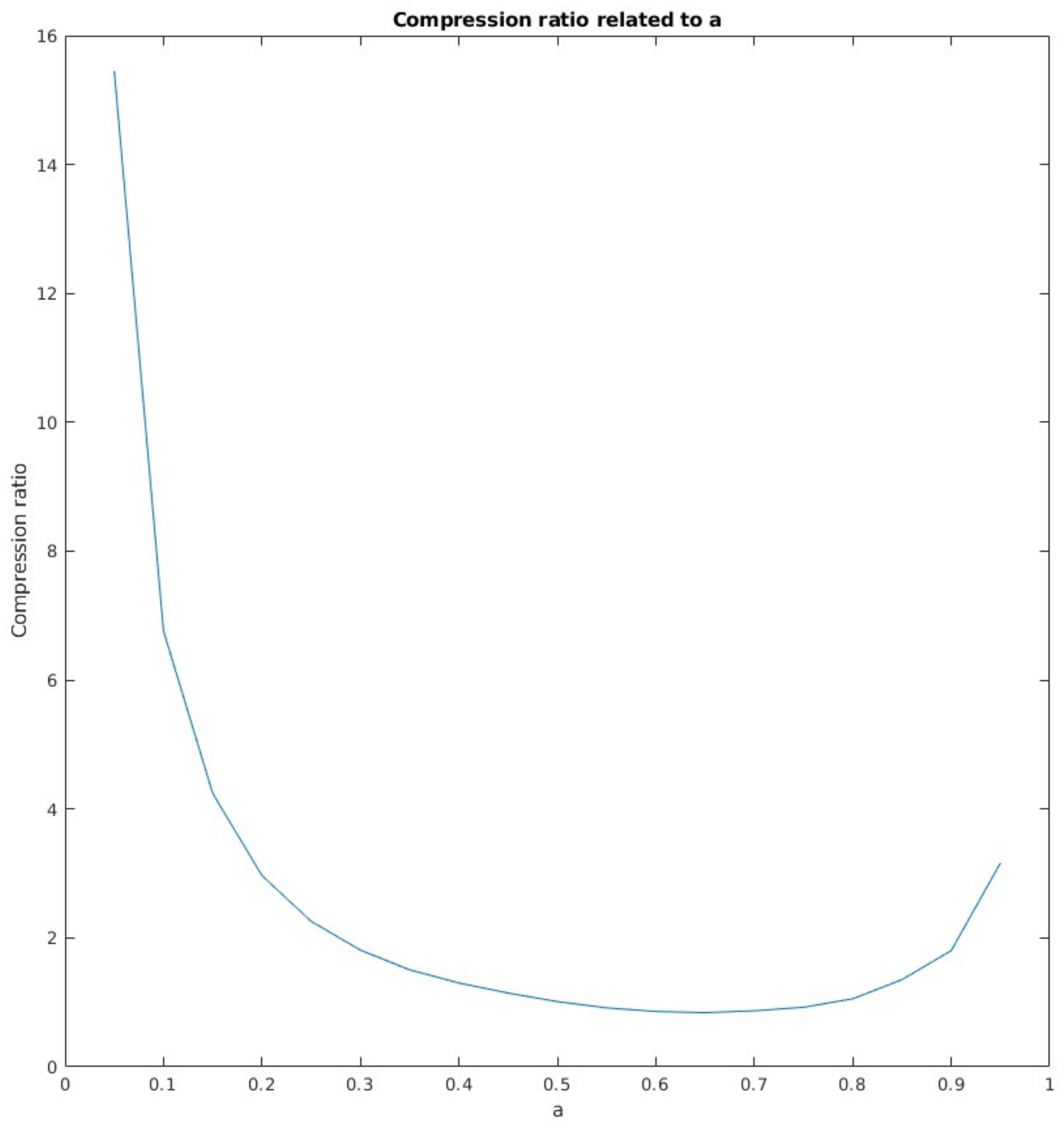


Abbildung 1: Run-length optimum binary Compression ratio versus a

From the image, we can find that the compression ratio achieves the lowest when $a \approx 0.5$. More the a is closer to 0 or 1, the higher is the compression rate. The absolute value of decreasing speed around a=0.1 is higher than the increasing speed's around a =0.95. The derivative is always positive, and the compression ratio versus a is convex. There exists errors in the computation due to the limitation of my laptop's compute ability.

And the Matlab code for generating pmf for a=0.05,0.5,0.95 is as follows, pmf.m:

```matlab
    %pmf.m for HW3 2(c) to draw figure
bits_number=19600
a=0.95  %a=0.05,0.5,0.95
source_stream=[]
length_a=length(a)

source_stream=markov_generate(a,bits_number)
run_length_code=binary2runlength(source_stream)


histObject = histogram(run_length_code, 'Normalization','
   pdf')
grid on;
xlabel('Run-length code numbers generated by a=0.95');
ylabel('PDF')
```

The pmf of run-length values for a=0.05 is plot as Abbildung 2:

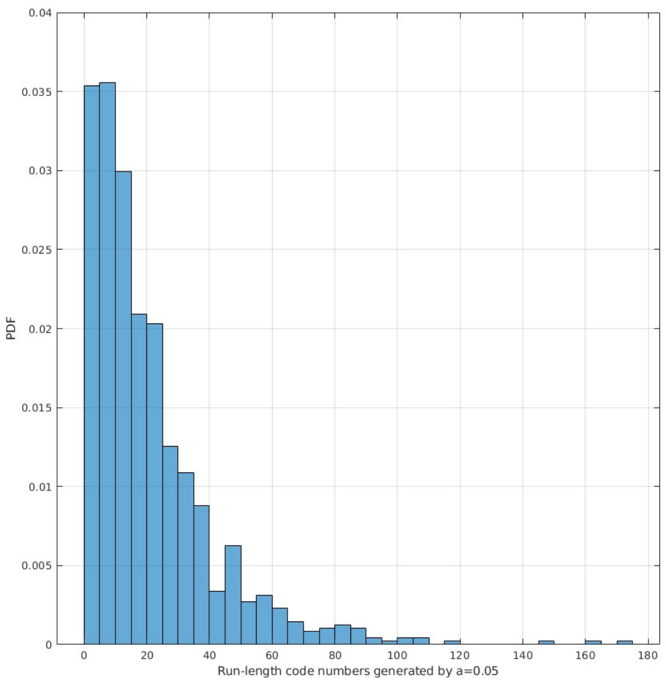


Abbildung 2: Run length PMF distribution versus a=0.05

I can observe that, for a=0.05, there are significantly more repetitions. And for a=0.5, the repetitions are less, the repetitions of a=0.95 is the least. For a=0.05, The pmf of run-length values for a=0.5 is plot as Abbildung 3:

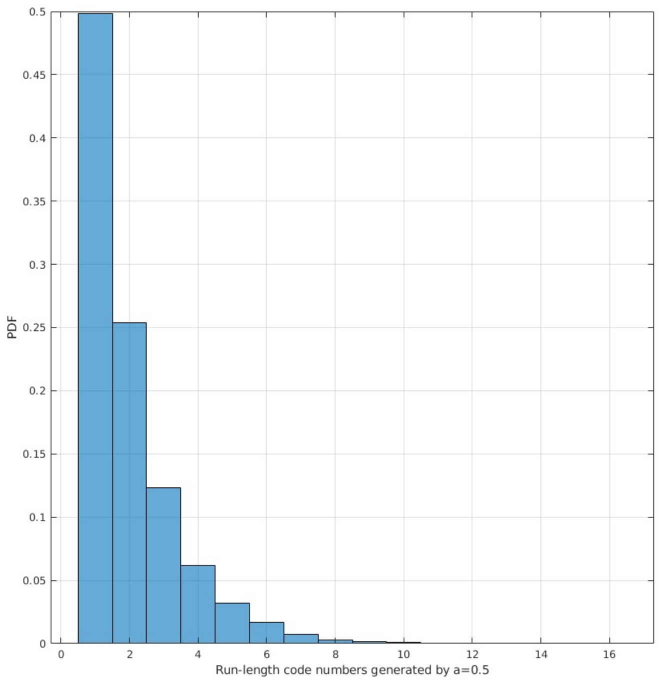


Abbildung 3: Run length PMF distribution versus a=0.5

the width of each histogram is narrowest, and there more types of run-length code's values in comparison with a=0.5 and a=0.95. And for all of a=0.05,0.5,0.95, the pmf is decreasing versus the increasing values of run-length code.

The pmf of run-length values for a=0.95 is plot as Abbildung 4:

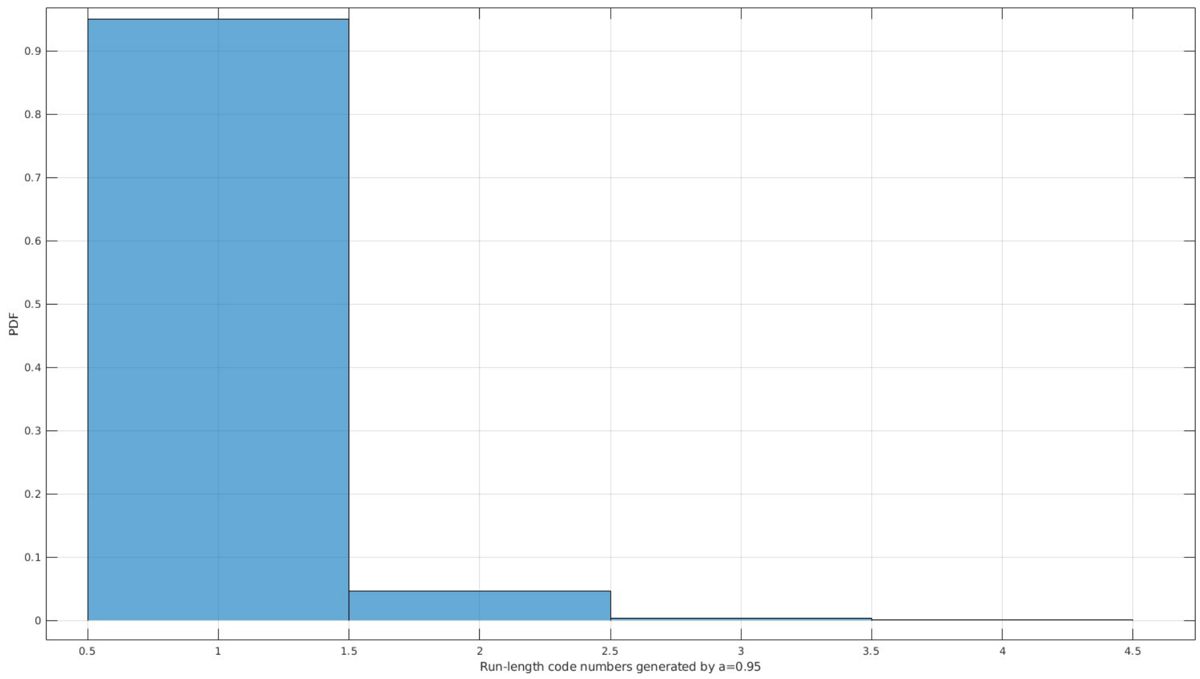


Abbildung 4: Run length PMF distribution versus a=0.95

□

## Solution for Problem 3

(a) **intout = bitshift(A,k)** returns A shifted to the left by k bits, which is equivalent to multiplying by $2^k$ and rounding to the nearest integer toward negative infinity. Any overflow bits will be truncated. **emit bit(0)** is the operation to append bit 0 at the end of the stream We set the initial probability state as p0=(0.5,0.5), and $\alpha = \beta$ because for stationary process:

$$p_0 = (\frac{\beta}{\alpha+\beta}, \frac{\beta}{\alpha+\beta}) = (0.5, 0.5)$$

The Matlab function for arithmetic encode with a=b for Markov-1 chain is arithmetic.m:

```
function [outputStream]=arithmetic_correct(a,x)
%Input: sequence of binary symbols x
%Input: transition probability a
%Output: binary arithmetic code for x

```

```
% Initialize variables
%x=[1,1,1,1,0,1,1,1,1,1,0];N = 22; P = 8; C = 0; A =
    2^N; r = −1; b = 0; outputStream = []p0 = [a/(a + a), a/(a + a)]p0 = p0(1)
p0=2^P*p0%current probabilities

%trasition_matrix = [1 − a, a; a, 1 − a];
%Encode the sequence
for n=1:length(x)
    %Update the interval low end and interval length
        T=A*p0;
        if x(n)==1
            C =C+T
            T=bitshift(A,P)-T%T = 2^P * A − Tend
    %Check if the interval needs renormalization
    if(C>=2^(N+P))
        C = mod(C, 2^(N+P))
    % propagate carry
    outputStream=[outputStream 1]%emit-bit(1)
    if r>0
        for i=1:r-1
            outputStream=[outputStream 0]
        end
        r=0
    else
        r=-1
    end
    end

    while T<2^(N+P-1)
        %renormalize once
        b = b+1;
        T = 2*T;
        C = 2*C;
        if C>=2^(N+P)
            C = bitshift(C, -N-P)%C = bitand(C,
                2^(N + P) − 1)ifr < 0
                outputStream=[outputStream 1]
            else
                r = r+1;
            end
        else
            %no overflow of C
            if r>=0
                outputStream=[outputStream 0]
                for i=1:r
                    outputStream=[outputStream 1]
                end
            end
            r=0
        end
    end
    A = floor(T/2^P);
    %transition matrix probability update
        %if(x(n)==0)
         % p = [1-a,a;]
        %else
        % p = [a,1-a;]
        %end
end
if r>=0
    outputStream=[outputStream 0];
    for i=1:r
        outputStream=[outputStream 1];
    end
end

%for i = 1:N+P
 % outputStream = [outputStream bitget(C, N+P-i+1)];
%end
```

(b) Compare the entropy of Markov-1 source stream and arithmetic code's entropy. Due to the computation limitation of my laptop, I set N=10000 for experiment.

The Matlab code for drawing the comparison of entropy of Markov-1 and arithmetic code is draw-question-b.m:

```
 N = 10000; % length of the Markov-1 stream
P = 10; % precision parameter
alpha = 0.1:0.05:0.9; % values of alpha to use

markov_entropies = zeros(size(alpha));
code_entropies = zeros(size(alpha));

for i = 1:length(alpha)
    % generate the Markov-1 stream
    markov_stream = markov_generate(alpha(i), N);

    % encode the stream using the binary arithmetic encoding algorithm
    code_stream= arithmetic_correct(alpha(i), markov_stream);

    % compute the entropies
    markov_entropies(i) = entropy(markov_stream);
    code_entropies(i) = entropy(code_stream);
end

% plot the estimated entropies
plot(alpha, markov_entropies, 'r-', alpha, code_entropies
    , 'b-');
%plot(alpha, code_entropies,'b-');

xlabel('');
ylabel('Estimated Entropy');
legend('Markov-1', 'Arithmetic Encoded Stream');
```

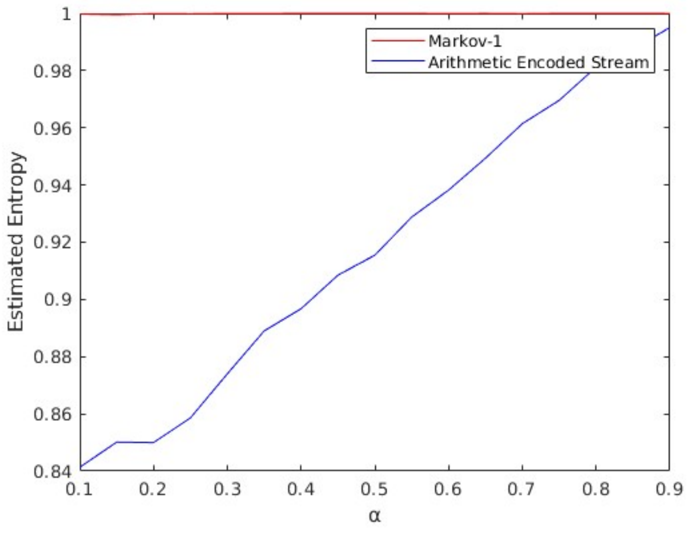


Abbildung 5: Entropy comparison of Markov-1 and arithmetic encode versus a

From the Figure 5. we can find that, the entropy value of Markov-1 source stream, and arithmetic stream are both equal and smaller than 1. The entropy of Markov-1 source stream is constant to 1. And the entropy of arithmetic enocoded stream is increasing when the $\alpha$ is increasing, and range from arond 0.84 to 1.

(c) Compare the entropy rate of Markov-1 source stream and arithmetic code's entropy rate.

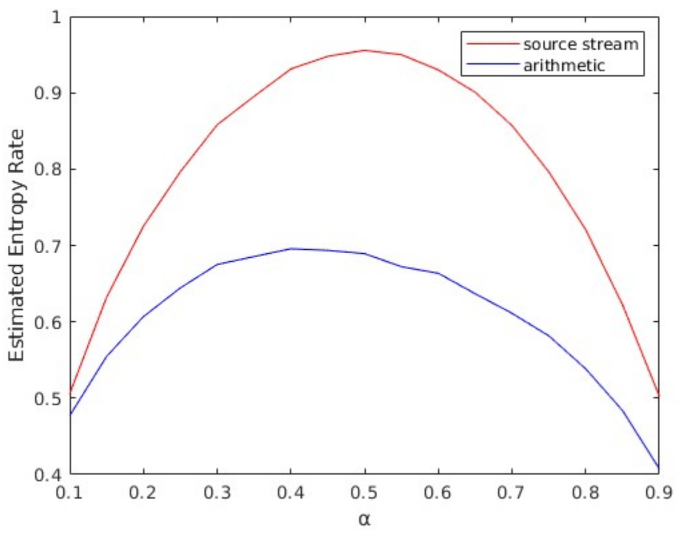
The Matlab code for drawing the entropy rate for Markov-1 and arithmetic code:

```
    source_stream =[];% the source stream gernated by Markov-1
arithmetic_code =[];% the arithmetic code generated by binary source stream
N=19600;% the number of elements in the markov generated sequence
m=10;
arithmetic_code_entropy_rate=zeros(size(a));% the entrpy rate of arithmetic code
source_entropy_rate=zeros(size(a));% the entropy rate of source stream
a=0.1:0.05:0.9;
for i=1:length(a)
    source_stream=markov_generate(a(i),N);
    arithmetic_code=arithmetic(a(i),source_stream);
    source_entropy_rate(i)=entropy_rate_markov(
        source_stream,m);
    arithmetic_code_entropy_rate(i)=entropy_rate_markov(
        arithmetic_code,m);

end

% plot the estimated entropies
plot(a, source_entropy_rate, 'r-', a,
    arithmetic_code_entropy_rate, 'b-');
xlabel('');
ylabel('Estimated Entropy Rate');
legend('source stream', 'arithmetic');
```

From Figure 6., we can find that: the arithmetic entropy rate is less than Markov-1 source stream. And their values is less than 1, and over than around 0.4. They have similar inverse-U shape, that increase firstly when $\alpha$ increase from 0 to around $\alpha = 0.5$, then decrease when $\alpha$ increase from 0.5 to 1.



N=19600

Abbildung 6: Entropy rate comparison of Markov-1 and arithmetic encode versus a

(d) 1. The compression ratio of arithmetic code.

2. Compare the arithmetic code compression ratio to the runlength code encoding with Golomb code's compression ratio.

The Matlab code for drawing compression rate for Markov-1 and arithmetic code, draw-comression.m:

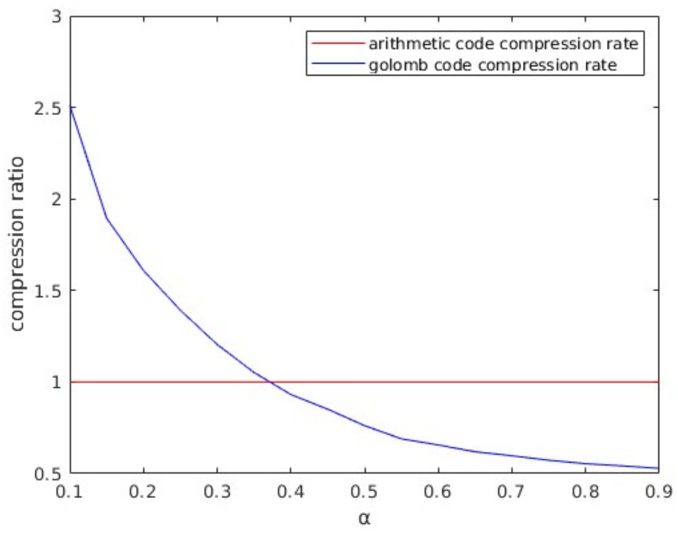
```
            %Draw the compression ratio of arithmetic and Golomb

a=0.1:0.05:0.9
N=2000%the number of markov numbers
run_length_code=[] %the run-length code
golomb_code=[] %the golomb code generate by run-length code
arithmetic_code=[] %the arithmetic code
source_stream=[] %the markov-1 source data
golomb_length_code_length=[] %the run-length code length versus a
arithmetic_code_length=[] %the arithmetic code length versus a

golomb_compression=[]; %the compression rate of golomb
arithmetic_compression=[] %the compression rate of arithmetic

for i=1:length(a)
    source_stream=markov_generate(a(i),N);
    arithmetic_code=arithmetic_correct(a(i),source_stream
       )
    run_length_code=binary2runlength(source_stream)
    golomb_code=golomb_encoding(run_length_code,5,4)
    arithmetic_code_length=[arithmetic_code_length length
       (arithmetic_code)]
    golomb_length_code_length=[golomb_length_code_length
       length(golomb_code)]
end
arithmetic_compression=N./arithmetic_code_length;
golomb_compression=N./golomb_length_code_length;


plot(a, arithmetic_compression, 'r-', a,
   golomb_compression, 'b-');
xlabel('');
ylabel('compression ratio');
legend('arithmetic code compression rate', 'golomb code
   compression rate');
```



N=19600

Abbildung 7: Compression ratio comparison of Golomb and arithmetic encode versus a

From the Figure 7. we can find that the golomb's compression rate is decreasing when the probability $\alpha$ increase. And the arithmetic's compression is costant to 1. They intersect when $\alpha$ is around 0.37. We can conclude that in which area estimation of $\alpha$, which encoder would work better at the compression rate aspect.

□

## Solution for Problem 4

(a) Follow is the Matlab function of Golomb encode, golomb-encoding.m:

```matlab
function code = golomb_encoding(r, A, Nmax)
%Initialize
N = 1;
single_code=[];
code=[]
i = 1;
final_code=[];
%FOR each run-length value r DO
for run = r
    %estimate parameter k = max0, ceil(log2(A/(2*N)))
    k = max(0, ceil(log2(A/(2*N))));

    %code r using Golomb code with parameter k
    q = floor(run / 2^k);
    r_k = run - q * 2^k;
    unary = repmat('0', 1, q);
    if q > 0
        unary(end) = '1'
    end
    single_code = [unary dec2bin(r_k, k)];
    code=[code single_code]
    i = i + 1;

    %IF N = Nmax
    if N == Nmax
        %A = floor(A/2)
        A = floor(A/2);
        %N = floor(N/2)
        N = floor(N/2);
    end

    %A = A + current run-length value
    A = A + run;
    %N = N + 1
    N = N + 1;
end

end
```

(b) The Matlab code to draw the compression comparison, draw-compression-q4.m:

```matlab
            %Draw the compression ratio of arithmetic and Golomb

a=0.05:0.05:0.95
N=5000%the number of markov numbers
m=10
```

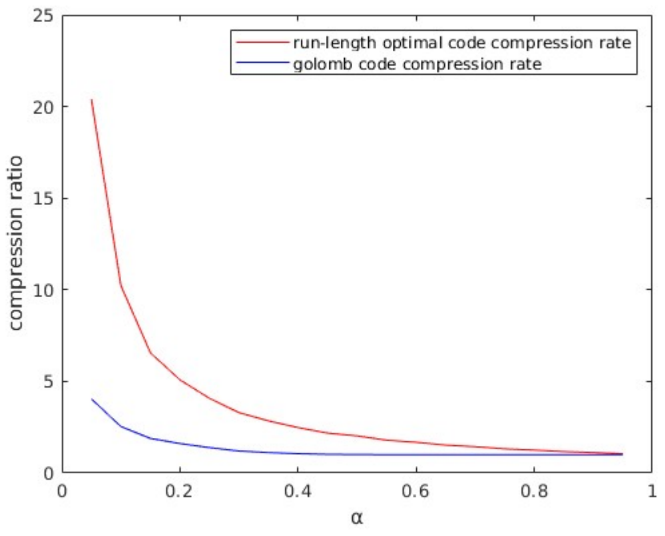
```
run_length_code=[]%the run-length code
golomb_code=[]%the golomb code generate by run-length code
run_length_code=[]% the run-length code
source_stream=[]% the markov-1 source data
golomb_length_code_length=[]%the golomb run-length code length
    versus a
optimal_code_length=[]% the run-length optimal code length versus a

golomb_compression=[];% the compression rate of golomb
optimal_compression=[]% the compression rate of arithmetic

for i=1:length(a)
    source_stream=markov_generate(a(i),N);
    run_length_code=binary2runlength(source_stream)
    golomb_code=golomb_encoding(run_length_code-1,5,4)
    optimal_code_length=[optimal_code_length length(
        optimalstream(run_length_code,m))]
    golomb_length_code_length=[golomb_length_code_length
        length(golomb_code)]
end
optimal_compression=N./optimal_code_length;
golomb_compression=N./golomb_length_code_length;


plot(a, optimal_compression, 'r-', a, golomb_compression,
    'b-');
xlabel('');
ylabel('compression ratio');
legend('run-length optimal code compression rate', '
    golomb code compression rate');
```

And the plot of compression rate of run-length optimal and golomb code is presented:



N=5000

Abbildung 8: Compression ratio comparison of optimum run-length code and golomb encode versus a

From the plot Figure 8. we can see that the compression rate of golomb and run-length are both decreasing versu $\alpha$. And the compression rate of run-length is always higher than golomb.

- The value of Nmax is a design parameter that affects the trade-off between the length of the encoded bitstream and the computational complexity of the encoder and decoder. The larger the value of Nmax, the lower the probability of needing to renormalize during the encoding process, but the longer the Golomb codes that are used to encode the run-length values. Typically, a value of Nmax is chosen to be a power of 2, so that the division and shift operations needed for renormalization can be implemented efficiently using bit-wise operations. A common choice is to set Nmax to be equal to the maximum representable value for an unsigned integer of width W, divided by 2 (i.e., $Nmax = 2^{W-1} - 1$). This ensures that the counter will never overflow during renormalization. The optimal value of Nmax may depend on the specific input source and the desired trade-off between compression ratio and encoding speed. Therefore, it may be necessary to experiment with different values of Nmax to find the optimal value for a particular application.A may also result in longer code words, which can reduce the compression ratio, so there is a trade-off between the value of A and the resulting code length. For the given problem, we can set A to a reasonable value based on the expected average run-length of the Markov-1 source.
- For small values of alpha, the run-length encoder achieves a higher compression ratio than the adaptive Golomb encoder. This is because the run-

length encoder is designed to handle runs of 0's and 1's, which are likely to occur in the Markov-1 source for small values of alpha.

- For large values of alpha, the adaptive Golomb encoder achieves a higher compression ratio than the run-length encoder. This is because the adaptive Golomb encoder is better suited to handle longer runs of arbitrary symbols, which are likely to occur in the Markov-1 source for large values of alpha.
- The performance of the adaptive Golomb encoder improves as the value of A is increased and the value of Nmax is decreased. A higher value of A means that longer runs are expected, and a lower value of Nmax means that the adaptive Golomb encoder will renormalize more frequently, allowing it to better adapt to changes in the statistics of the input stream. However, increasing

□

**Information Theory and Source Coding**

# 4. Assignment

Lecture Professor: Markus Fierl

TA:Shudi Weng

Jin, Xin

11. August 2023

## Solution for Problem 1

(a) State the given mean absolute error criteria $\gamma$ for the Shannon lower bound. D is the given distortion limitation. Assume the random variable X, the estimated $\hat{X}$ for X during data transmission.

For the given mean absolute error, the distribution of X would fulfill:

$$\gamma = E(d(X,\hat{X})) = E(|X-\hat{X}|) \leq D$$

the Shannon lower bound would be:

$$h(X) - \sup_{E(|X-\hat{X}|)\leq D} h(X-\hat{X})$$

The first term in the Shannon lower bound is the entropy of source signal distribution, which is constant for a given source. The second term of the Shannon lower bound is the superior value for the entropy of $(X-\hat{X})$ under the condition that absolute mean error lower than $\gamma$, which is determined by the estimation method of X to $\hat{X}$. The second sup-term is determined by the norm and limitation of d.

(b) The Shannon lower bound for R(D) is:

$$R(D) \geq h(X) - \sup_{d\leq D} h(X-\hat{X})$$

which indicates that, we can estimate the Shannon lower bound by estimating the maximum entropy of $h(X-\hat{X})$ to decide the sup-term of Shannon lower bound. The second sup-term is determined by the norm and limitation of d.

(c) Because the second sup-term of Shannon lower bound is only determined by the norm and limitation of d, we want to estimate the maximum entropy for the second sup-term firstly,

Given:

$$E(f_V(v)|v|) = \gamma = \int f_V(v)|v|dv \tag{1}$$

And $f_V(v)$ is a density that:

$$\int f_V(v)dv = 1$$

We want to find the $f_V(v)$ that maximizes:

$$h(V) = -\int f_V(v)\log_2 f_V(v)dv$$

We should also impose the constraint that $f_V(v) \geq 0$, but we will see the solution satisfying this constraint. Given the Lagrange opimization method, and assume symmetry and optimize the criterion

$$F = \int_0^\infty (-f_V(v)\log(f_V(v)) + \lambda f_V(v)vdv)$$

where $\lambda$ is the Lagrange multiplier. The optimization equation for derivative=0 will be:

$$-\log(f_V(v)) - 1 + \lambda v = 0$$

Apply eq. (1), obtain the maximum entropy distribution:

$$f_V(v) = \frac{1}{2\gamma}e^{-\frac{|v|}{\gamma}} \tag{2}$$

And the maximum entropy for the second sup-term would be:

$$H_{sup} = \log_2(2e\gamma) \tag{3}$$

- Rectangular sample density is a type of probability distribution that represents an i.i.d. process where the samples are uniformly distributed within a finite range. In other words, each sample has an equal probability of occurring within the range, and there is zero probability of samples outside the range. Assume pdf as an uniform distribution as $p(x) = \frac{1}{A}$.

  The goal is to find the maximum entropy distribution W for $h(W)$ given $E(|W|) \leq \gamma$.

  $$\sup_W h(W), \qquad E(|W|) \leq \gamma$$

  1. Calculate the entropy of the source X:

  $$h(X) = \log_2(A)$$

  2. Calculate the Shannon lower bound for rectangular sample:

  $$\begin{aligned} R_{SLB}(\gamma) &= h(X) - H_{sup} = \log_2 A - \log_2(2e\gamma) \\ &= \log_2 \frac{A}{2e\gamma} \end{aligned}$$

- Assume the Gaussian sample density $\mathcal{N}(\mu, \sigma^2)$:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  1. Calculate the entropy of X:

$$h(X) = \frac{1}{2}\log_2 2\pi e\sigma^2$$

  2. Calculate the Shannon lower bound for Gaussian sample:

$$\begin{aligned} R_{SLB}(D) &= h(X) - H_{sup} \\ &= \frac{1}{2}\log_2 2\pi e\sigma^2 - \log_2 2e\gamma \\ &= \log_2 \frac{\sqrt{\pi}\sigma}{\sqrt{2e}\gamma} \end{aligned}$$

- Assume the Laplace sample density as:

$$f(x) = \frac{a}{2} e^{-a|x-\mu|}$$

  1. Calculate the entropy of X:

$$h(X) = \log_2 \frac{2e}{a}$$

  2. Calculate the Shannon lower bound for Laplace smaple:

$$\begin{aligned} R_{SLB}(D) &= h(X) - H_{sup} \\ &= \log_2 \frac{2e}{a} - \log_2 2e\gamma \\ &= -log_2(a\gamma) \end{aligned}$$

(d) According to the Kleijn's book, if rate-distortion function coincides with the Shannon lower bound, then

$$\begin{aligned} h(X - \hat{X}|\hat{X}) &= h(X - \hat{X}) \\ I(X - \hat{X}; \hat{X}) &= 0 \end{aligned}$$

From a theoretical aspect, when the Shannon lower bound is tight, in other words, it definitely possible to create a variable with the statistics of X by adding the independent variables $\hat{X}$ and V(the maximum entropy distribution given the distortion measure D), the rate-distortion would coincide with the shannon lower bound. And given the Fourerir transform $\mathcal{F}$ given:

$$\mathcal{F}f_{\hat{X}}(\omega) = \mathcal{F}f_X(\omega)/\mathcal{F}f_V(\omega)$$

if the inverse Fourier transform of $\mathcal{F}f_{\hat{X}}(\omega)$ is a density(nonnegative), then the shannon lower bound is tight, and fulfill the question condition.

Consider the practical cases examples:

1. Gaussian source(squared error in the book): For a Gaussian source with a variance $\sigma^2$, the lower bound on the rate-distortion function given by the Shannon lower bound coincides with the actual rate-distortion function.

2. Laplacian source(absolute error in the book): For a Laplacian source the lower bound on the rate-distortion function given by the Shannon lower bound coincides with the actual rate-distortion function.

$\square$

## Solution for Problem 2

(a) Given the definition of $E(d(X, \hat{X}))$

$$\begin{aligned} E(d(X,\hat{X})) &= P(X=\hat{X})\cdot d(X=\hat{X}) + P(\hat{X}=?)\cdot d(\hat{X}=?) \\ &+ P(X=0,\hat{X}=1;X=1,\hat{X}=0)\cdot d(X=0,\hat{X}=1;X=1,\hat{X}=0) \end{aligned}$$

Such that,

if $P(X=0,\hat{X}=1;X=1,\hat{X}=0) \neq 0$

$$E(d(X,\hat{X})) = \infty$$

if $P(X=0,\hat{X}=1;X=1,\hat{X}=0) = 0$

$$E(d(X,\hat{X})) = P(\hat{X}=?)$$

(b) For the distortion rate function:

$$R(D) = \inf_{f(X|\hat{X})} I(X;\hat{X}) \qquad s.t. \qquad E(d(X,\hat{X})) \leq D \tag{4}$$

$$R(D) = h(x) - \sup h(X|\hat{X}) \qquad s.t. \qquad E(d(X,\hat{X})) \leq D \tag{5}$$

that we can only consider the condition that $P(X=0,\hat{X}=1;X=1,\hat{X}=0)=0$, otherwise $E(d(X,\hat{X}))$ will be infinite and $E(d(X,\hat{X})) \leq D$ can never be fulfilled. We want to find the maximum value for the second sup-term in equation. (5)

Such that we can generate:

$$p(X=0|\hat{X}=0) = 1$$
$$p(X=0|\hat{X}=1) = 0$$

$$p(X=1|\hat{X}=1)=1$$
$$p(X=1|\hat{X}=0)=0$$

That the above four items would be zero for the entropy calculation in $h(X|\hat{X})$, and the value of $h(X|\hat{X})$ would be determined by $p(X=0|\hat{X}=?)$ and $p(X=1|\hat{X}=?)$.

From the previous 2(a), we generate $E(d(X,\hat{X}))=P(\hat{X}=?)$, such that

1. if $0\leq D\leq 1$, then $0\leq P(\hat{X}=?)\leq D\leq 1$

$$P(\hat{X}=?)=p(X=0|\hat{X}=?)+p(X=1|\hat{X}=?)$$

And have equations and inequalities

$$h(X|\hat{X})=-p(X=0,\hat{X}=?)\log_2 p(X=0|\hat{X}=?)-p(X=1,\hat{X}=?)\log_2 p(X=1|\hat{X}=?) \tag{6}$$
$$p(X=0|\hat{X}=?)+p(X=1|\hat{X}=?)=1 \tag{7}$$
$$p(X=0,\hat{X}=?)+p(X=1,\hat{X}=?)\leq D \tag{8}$$

the $h(X|\hat{X})$ would be maximied when

$$P(\hat{X}=?)=D$$
$$p(X=0|\hat{X}=?)=p(X=1|\hat{X}=?)=\frac{1}{2}$$
$$p(X=0,\hat{X}=?)=p(X=1,\hat{X}=?)=\frac{1}{2}\cdot P(\hat{X}=?)\leq\frac{D}{2}$$

such that

$$\begin{aligned}
&\sup_{d\leq D} h(X|\hat{X})\\
&=\sup[-p(X=0,\hat{X}=?)\log_2 p(X=0|\hat{X}=?)-p(X=1,\hat{X}=?)\log_2 p(X=1|\hat{X}=?)]\\
&=-\frac{D}{2}\log_2\frac{1}{2}-\frac{D}{2}\log_2\frac{1}{2}\\
&=D
\end{aligned}$$

To conclude that:

$$R(D)=h(X)-\sup_{d\leq D} h(X|\hat{X})=1-D$$

where h(X) is the entropy of Bernoulli($\frac{1}{2}$)

2. if $D>1$, then $0\leq P(\hat{X}=?)\leq 1$, $h(X|\hat{X})$ would be maximized with:

$$P(\hat{X}=?)=1$$

$$p(X=0|\hat{X}=?) = p(X=1|\hat{X}=?) = \frac{1}{2}$$
$$p(X=0,\hat{X}=?) = p(X=1,\hat{X}=?) = \frac{1}{2}\cdot P(\hat{X}=?) \leq \frac{1}{2}$$

Such that

$$\sup_{d\leq D} = 1$$

Finally,

$$R(D) = h(X) - \sup_{d\leq D} = 0$$

We get

$$R(D) = \begin{cases} = 1-D, & 0 \leq D \leq 1 \\ = 0, & D > 1 \end{cases} \tag{9}$$

(c) A simple scheme to achieve any value of the rate distortion function for this source is the following:

if the desired distortion $0 \leq D \leq 1$

1. Encode the source with a fixed rate R0=1-D

2. Transmit the encoded message over the noisy channel.

3. At the receiver, decode the message using the same fixed rate R0.

4. Apply a distortion measure to the decoded message to obtain the output Y.

5. Considering the transmission loss and decoder loss, the theoretical D can always not be achieved. If the real-distortion d is greater than the desired distortion level D, discard the decoded message and repeat the encoding and transmission process with a higer rate R1, where R1 is greater than or equal to R0. Continue increasing the rate until the desired distortion level D is achieved.

6. If the distortion D(Y,X) is less than or equal to the desired distortion level D, output Y as the final result.

if the desired distortion $D > 1$, then we could set the beginning R0 at a least rate according to the conclusion of 2(b), and repeating the steps 2.3.4.5.6.

This scheme can achieve any value of the rate distortion function R(D) because it is based on an iterative process of encoding and transmission with increasing rates until the desired distortion level is achieved. By adjusting the rate R0 and the increasing sequence of rates R1, R2, R3, ..., we can achieve any point on the rate distortion curve. The scheme also guarantees that the output distortion will be less than or equal to the desired distortion level, since we discard any decoded message that does not meet the desired distortion level.

This scheme is a simple example of a variable rate coding scheme, where the encoder and decoder use different rates for different parts of the source sequence depending on the distortion level. This is in contrast to a fixed rate coding scheme, where the encoder and decoder use a fixed rate for the entire source sequence. Variable rate coding schemes are commonly used in practical coding schemes to achieve better performance.

$\square$

**Information Theory and Source Coding**

# 5. Assignment

Lecture Professor: Markus Fierl

TA:Shudi Weng

Jin, Xin

11. August 2023

## Solution for Problem 1

According to the question, the absolute difference distortion measure is applied, for **mean** distortion $D_i$ of each cell and denote $\int\limits_{V_i} dx$ as the integral over the cell:

$$
\begin{aligned}
D_i &= \frac{\int_{V_i} f_X(x) d(x, \mathcal{Q}(x)) dx}{\int_{V_i} f_X(x) dx} \\
&= \frac{\int_{V_i} f_X(x) |x - c_i| dx}{\int_{V_i} f_X(x) dx} \\
&\approx \frac{\int_{V_i} f_X(c_i) |x - c_i| dx}{f_X(c_i) \cdot \Delta_i} \\
&= \frac{\int_{-\frac{\Delta_i}{2}}^{c_i} (c_i - x) dx + \int_{c_i}^{\frac{\Delta_i}{2}} (x - c_i) dx}{\Delta_i} \\
&= \frac{1}{4} \Delta_i
\end{aligned}
$$

where $\Delta_i$ is the step size of the quantizer and we assumed the **centroid is located at any point of the cell**$[-\frac{\Delta_i}{2}, \frac{\Delta_i}{2}]$ so as to minimize the distortion.
The step size $\Delta_i$ of a scalar quantizer is inversely proportional to the local density of the centroids(quantizer reconstruction points) per unit length, denoted as $g_C(x)$:

$$g_C(c_i) = \frac{1}{\Delta_i}$$

Denote $p_I(i)$ as the index probability we have:

$$
\begin{aligned}
D &= \sum_{i \in I} p_I(i) D_i && (1) \\
&\approx \frac{1}{4} \sum_{i \in I} p_I(i) \Delta_i && (2) \\
&= \frac{1}{4} \sum_{i \in I} p_I(i) \frac{1}{g_C(c_i)} && (3)
\end{aligned}
$$

$$= \frac{1}{4}\sum_{i\in I}\int_{V_i} f_X(x)\frac{1}{g_C(c_i)}dx \tag{4}$$

$$= \frac{1}{4}\sum_{i\in I}\int_{V_i} f_X(x)\frac{1}{g_C(x)}dx \tag{5}$$

$$\approx \frac{1}{4}\int_{\mathbf{R}} f_X(x)\frac{1}{g_C(x)}dx \tag{6}$$

(a) For resolution-constrained scalar quantization:

$$\int_{\mathbf{R}} g_C(x)dx = N \tag{7}$$

Using the Lagrange multiplier we can obtain modified criterion as:

$$\mu = \frac{1}{4}(\int_{\mathbf{R}}(f_X(x)\frac{1}{g_C(x)} + \lambda g_C(x))dx - \lambda N)$$

where $\lambda$ is the Lagrange multiplier. The Euler-Lagrange equation is:

$$-f_X(x)\frac{1}{g_C(x)^2} + \lambda = 0 \tag{8}$$

Combine equations (7) and (8), we can find the solution:

$$g_C(x) = N \cdot \frac{f_X(x)^{\frac{1}{2}}}{\int_{\mathbf{R}} f_X(x)^{\frac{1}{2}}dx} \tag{9}$$

Combine equations (6) and (9) we obtain:

$$D = \frac{1}{4N}(\int_{\mathbf{R}} f_X(x)^{\frac{1}{2}}dx)^2 \tag{10}$$

Using the fact that the rate is $R = \log(N)$, we obtain the relationship between rate and distortion:

$$R = -\log 4D + 2\log(\int_{\mathbf{R}} f_X(x)^{\frac{1}{2}}dx)$$

Consider $f_X(x) = \frac{a}{2}e^{-a|x|}$ is Laplace density, that:

$$\begin{aligned} R(D) &= -\log 4D + 2\log\frac{2\sqrt{2}}{\sqrt{a}} \\ R(D) &= -\log 4D + \log\frac{4}{a} \\ R(D) &= -\log aD \end{aligned}$$

The plot of rate and distortion for high rate constrained-resolution for Laplace density(a=1) is as follows:

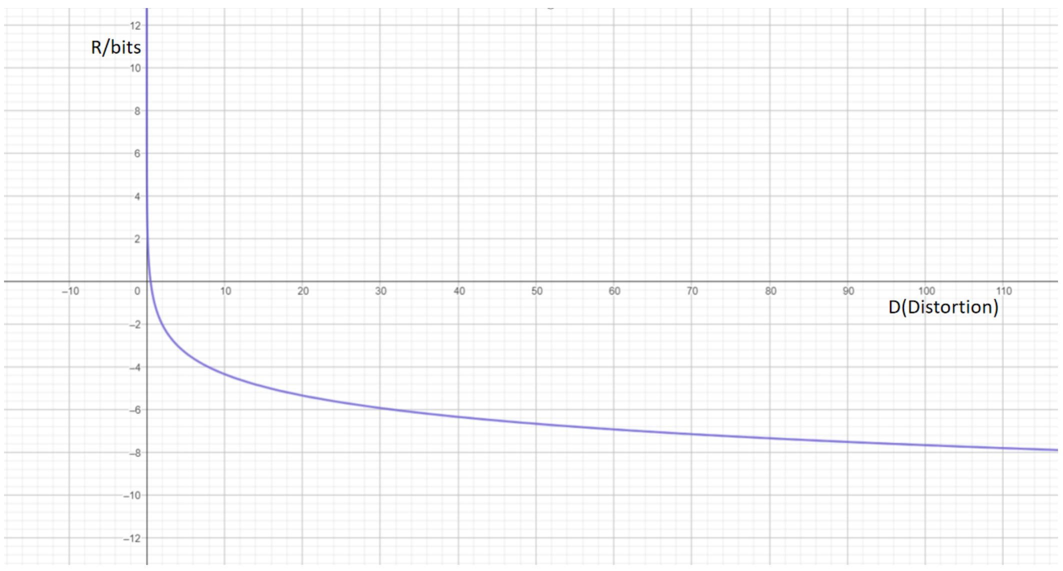


Abbildung 1: The relationship between Rate and Distortion, for Laplace density a=1

(b) In practice, the first-order entropy of the indices is generally used as the measure of rate:

$$H(I) = -\sum_{i\in I} p_I(i)\log(p_I(i))$$

We first determine the index entropy in terms of the differential entropy of the data and centroid density $g_C(x)$:

$$\begin{aligned} H(I) &= -\sum_{i\in I} p_I(i)\log p_I(i) && (11)\\ &\approx -\sum_{i\in I} f_X(c_i)\Delta_i \log(f_X(c_i)\Delta_i) && (12)\\ &\approx -\int_{\mathbf{R}} f_X(x)\log\frac{f_X(x)}{g_C(x)}dx && (13)\\ &= h(X) + E[\log(g_C(X))] && (14) \end{aligned}$$

To minimize the distortion:

$$D = \int_{\mathbf{R}} f_X(x)d(x, \mathcal{Q}(x))dx$$

The distortion is(equation (6)):

$$D = \frac{1}{4}\int_{\mathbf{R}} f_X(x)\frac{1}{g_C(x)}dx \tag{15}$$

And apply the constraint(equation. (14)). Because h(X) is constant during the optimization, the constraint can be simplified to:

$$b = E[\log(g_C(X))] = \int_{\mathbf{R}} f_X(x)\log(g_C(x))dx$$

where b is a constant, and apply the Lagrange-multiplier, obtain the criterion:

$$\mu = \frac{1}{4}(\int_{\mathbf{R}} (f_X(x)\frac{1}{g_C(x)} + \lambda f_X(x)\log(g_C(x)))dx - \lambda b)$$

get the Euler-Lagrange equation:

$$-\frac{1}{g_C(x)^2} + \lambda\frac{1}{g_C(x)} = 0$$

which is solved by

$$g_C(x) = constant$$

So entropy-constrained optimization under high rate scalar quantizer is a quantizer with uniform reconstruction-point density. An important corollary of the uniformity of the reconstruction-point density is that the quantizers are the same for all distortion measures, which means that we can design an optimal entropy-constrained quantizer even when the precise distortion measure is not known.

From equation (14) it follows rewriting that:

$$g_C(x) = 2^{H(I)-h(X)}$$

$$\Delta_i = 2^{-H(I)+h(X)}$$

And obtain that:

$$\begin{aligned} D &\approx \frac{1}{4}\int_{\mathbf{R}} f_X(x)\frac{1}{g_C(x)}dx \\ &= \frac{1}{4}2^{-H(I)+h(X)} \end{aligned}$$

Thus,

$$R(D) = H(I) = h(X) - \log 4D$$

This equation shows that, in the high-rate, entropy-constrained case, the logarithm of the RMS distortion is just the difference between the differential entropy of the variable and the entropy of the quantization indices (plus a constant). As expected, if the entropy of the quantization indices increases, the distortion decreases.

Consider $f_X(x)$ as a Laplace density:

$$R(D) = \log\frac{2e}{a} - \log 4D \tag{16}$$

The plot is shown as follows:

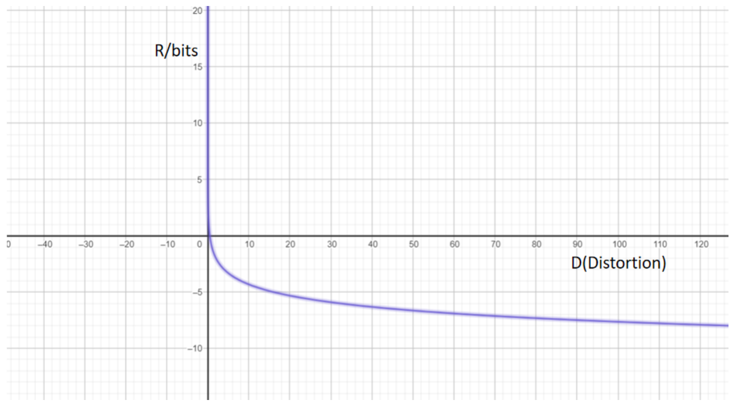


Abbildung 2: The relationship between Rate and Distortion, constrained-entropy

(c) The Rate distortion function is:

$$R(D) = h(X) - \sup_{d \leq D} h(X - \hat{X}|\hat{X})$$

Shannon lower bound given absolute error measure is:

$$R_{SLB} = h(X) - \sup_{|Y| \leq D} Y$$

As it turns out, the equations look very similar when the constrained-entropy case is considered for the high-rate theory. Consider Laplace density for Shannon Lower bound given distortion limitation D:

$$R_{SLB} = -\log_2 aD \tag{17}$$

And the constrained-entropy equation. (16):

$$H(I) - R(D)_{SLB} = -\log_2 4D - (-\log_2 aD) + \log \frac{2e}{a} = \log_2 \frac{e}{2}$$

That the difference between H(I) and R(D) is constant.

Consider constrained-resolution:

$$R(D) = \log aD$$

has the same form with the Shannon lower bound generated by rate distortion function. From the comparison we can conclude that, in practical scenarios the performance of scalar quantizers(constrained-entropt, constrained-resolution) in combination with efficient lossless coding is sufficient under the high rate and identical distributed process.

□

## Solution for Problem 2

(a) The high-resolution quantization means that the quantizer is not constrained-resolution, that we should consider constrained-entropy. For scalar quantization with square-error, Given 7.33(Kleijn book):

$$H(I) = R(D) = h(X) - \frac{1}{2}\log 12D \tag{18}$$

For Gaussian variable with variance $\sigma^2$ (**the high resolution relation between overall rate and distortion**)

$$H(I) = \frac{1}{2}\log(\frac{\sigma^2}{D}) + \frac{1}{2}\log(\frac{2\pi e}{12}) \tag{19}$$

The distortion-rate for each of component is:

$$E(d_i) = C\sigma^2_{X_i}e^{-2R_{X_i}}$$

where $\sigma^2_{X_i}$ is the variance of a random component, and C is a constant. Because entropy is constrained, the component with lowest distortion should be the direction with the **lowest variance(smallest eigen value of the covariance matrix)**.

To find the eigenvalues and eigenvectors of the covariance matrix, we need to solve the characteristic equation:

$$A = E[X^k X^k T] = \begin{bmatrix} 4 & \sqrt{2} \\ \sqrt{2} & 2 \end{bmatrix}$$

$$det(A - \lambda I) = 0$$

where A is the covariance matrix, $\lambda$ is the eigenvalue, I is the identity matrix of the same size as A.

So we have:

$$(4-\lambda)(2-\lambda) = 0$$
$$\lambda^2 - 6\lambda + 6 = 0$$

Using the quadratic formula, we get:

$$\lambda_1 = 3 + \sqrt{3}$$
$$\lambda_2 = 3 - \sqrt{3}$$

The corresponding eigenvectors are:

$$x_1 = \begin{bmatrix} \sqrt{6}+\sqrt{2} & 2 \end{bmatrix}$$
$$x_2 = \begin{bmatrix} \sqrt{2}-\sqrt{6} & 2 \end{bmatrix}$$

After normalization:

$$x_1 = \begin{bmatrix} \sqrt{\frac{\sqrt{3}+1}{2\sqrt{3}}} & \frac{1}{\sqrt{\sqrt{6}(\sqrt{3}+1)}} \end{bmatrix}$$
$$x_2 = \begin{bmatrix} 6^{-\frac{1}{4}} & \frac{2}{\sqrt{12-2\sqrt{3}}} \end{bmatrix}$$

$x_1$ and $x_2$ are orthogonal because $x_1 x_2^T = 0$ The $x_2$ is the component with least distortion.

(b) The Karhunen-Loeve transform(KLT) is a linear transformation that diagonalized the covariance matrix of a random vector. To find the KLT, we first compute its eigen decomposition.

To raise the linear algebra theories firstly: If B is a symmetric real matrix, then:

$$B = P \sum P^{-1} = P \sum P^T$$

where $\sum$ is the diagonal matrix of eigenvalues Of B, and Q's column vectors are the eigenvectors of B.

$$UAU^T = \sum$$
$$A = \begin{bmatrix} \sqrt{\frac{\sqrt{3}+1}{2\sqrt{3}}} & 6^{-\frac{1}{4}} \\ \frac{1}{\sqrt{\sqrt{6}(\sqrt{3}+1)}} & \frac{2}{\sqrt{12-2\sqrt{3}}} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \sqrt{\frac{\sqrt{3}+1}{2\sqrt{3}}} & 6^{-\frac{1}{4}} \\ \frac{1}{\sqrt{\sqrt{6}(\sqrt{3}+1)}} & \frac{2}{\sqrt{12-2\sqrt{3}}} \end{bmatrix}^H$$

that U:

$$U = \begin{bmatrix} \sqrt{\frac{\sqrt{3}+1}{2\sqrt{3}}} & 6^{-\frac{1}{4}} \\ \frac{1}{\sqrt{\sqrt{6}(\sqrt{3}+1)}} & \frac{2}{\sqrt{12-2\sqrt{3}}} \end{bmatrix}^H = \begin{bmatrix} \sqrt{\frac{\sqrt{3}+1}{2\sqrt{3}}} & \frac{1}{\sqrt{\sqrt{6}(\sqrt{3}+1)}} \\ 6^{-\frac{1}{4}} & \frac{2}{\sqrt{12-2\sqrt{3}}} \end{bmatrix}$$

Such that the Karhunen Loeve transform(KLT) is:

$$Y = UX$$
$$E(YY^H) = E(UXX^HU^H) = UAU^H = \sum$$

where $\sum$ is the diagonal matrix of eigenvalues of matrix A, that

$$\sum = \begin{bmatrix} 3+\sqrt{3} & 0 \\ 0 & 3-\sqrt{3} \end{bmatrix}$$

that $\sum$ is the new covariance matrix for Y(generated by X after KLT transformation)

(c) The total distortion after unitary transform U are:

$$D = \sum_{i=0}^{k-1} C\sigma_{Y_i}^2 e^{-2R_{Y_i}}$$

That under the entropy-constrained condition, we still want to find the lowest-variance direction for lowest distortion. To find the component with lowest distortion, we want to calculate the eigenvalues and eigenvectors for covariance matrix of Y that:

$$E(YY^H) = \sum = \begin{bmatrix} 3+\sqrt{3} & 0 \\ 0 & 3-\sqrt{3} \end{bmatrix}$$

$$det(E(YY^H) - \lambda) = 0$$
$$\lambda^2 - 6\lambda + 6 = 0$$

The 2 eigenvalues are:

$$\lambda_3 = 3 + \sqrt{3}$$
$$\lambda_4 = 3 - \sqrt{3}$$

The eigenvectors are:

$$x_3 = 0, x_4 = 0$$

The eigenvectors are 0, which means that there are no directions in the input space owning a non-zero variance, and the transformation de-correlate the data space completely. Thus, there is no components resulting in a lowest distortion.

(d) Comparing results of (a) the lowest component performing lowest distortion, and (c) there no specific components performing lowest distortion in the transformed space, when the eigenvalues do not change, which means that they can both achieve the lowest distortion. We can conclude that, for KLT transformation, which de-correlate the data space would allow low distortion within more components. And KLT also maximize the coding gains at the same time. Such that KLT(Karhunen-Loeve transform) is the best strategy.

$\square$