1st Chapter: Concepts of Probability and Information Theory

COMP 41280

Félix Balado

School of Computer Science
University College Dublin

Outline of the Chapter

Information Sources

- Discrete information source: any device that sequentially generates elements from a discrete alphabet
  - example: a person typing characters from the alphabet $\Omega = \{$ a, b, c, ..., w, y, z, space $\}$
- The outcome is not know beforehand, but we know it will belong to Ω
  - any discrete information source may be modelled by assigning probabilities to events from the sampling space Ω
- A discrete random variable (r.v.) X can be defined by mapping events from Ω to an alphabet (or support set) $\mathcal{X} \subset \mathbb{R}$
- examples (alphabet):
  - $\mathcal{X} = \{0, 1, 2, \cdots, 26\}$ (mapping each letter in Ω to a number)
  - $\mathcal{X} = \{0, 1\}$ (mapping consonants to 0, vowels to 1)

Random Variables and Information Sources

- Take $\mathcal{X} = \{x_1, \cdots, x_n\}$ to be the support set of r.v. X
  - $|\mathcal{X}| = n$ (cardinality of $\mathcal{X}$, or alphabet size)
  - each element in the set can be assigned a probability depending on the probability of the events from Ω
- The probability mass function (pmf) of X is the set of all probabilities $p(X = x)$, with $x \in \mathcal{X}$
  - pmf: $p(X = x_1), \cdots, p(X = x_n)$
  - notation: we just write $p(x_1), \cdots, p(x_n)$ if X is understood
- Properties of a pmf:
  - $0 \leq p(x) \leq 1$ for any $x \in \mathcal{X}$
  - $\sum_{x \in \mathcal{X}} p(x) = 1$

Example: pmf of Binary Random Variable

- r.v. X taking two values (Bernoulli r.v.), for example $\mathcal{X} = \{0, 1\}$
- Alphabet size: $|\mathcal{X}| = 2$
- Examples of possible pmfs:
  - $p(0) = 0.3$, $p(1) = 0.7$
  - $p(0) = 1$, $p(1) = 0$ (<u>deterministic</u>)

Expectation Operator

- The expectation of r.v. X is the sum of all its possible outcomes weighted by their likelihoods

$$E(X) = \sum_{x \in \mathcal{X}} x \, p(x)$$

  - $E(X)$ is also called the average of X

Expectation Operator

- The expectation of r.v. X is the sum of all its possible outcomes weighted by their likelihoods

$$E(X) = \sum_{x \in \mathcal{X}} x \; p(x)$$

  - $E(X)$ is also called the average of X
- We can also take the expectation of a function $g(\cdot)$ of a r.v. X (since $g(X)$ is another r.v.)

$$E(g(X)) = \sum_{x \in \mathcal{X}} g(x) \; p(x)$$

Statistical Independence of Random Variables

- What is the pmf of X after having observed the outcome of another r.v. Y? (on which it might or might not depend)
- pmf of X conditioned to $Y = y$: $p(X = x|Y = y) = p(x|y)$
  - probabilities of $X = x$, for $x \in \mathcal{X}$, after knowing that $Y = y$
- if $p(x|y) = p(x)$ <u>for all x, y</u> then X and Y are independent
  - <u>example</u>: X and Y are two r.v.'s representing the simultaneous tossing of two dice, and thus $\mathcal{X} = \mathcal{Y} = \{1, 2, 3, 4, 5, 6\}$; with fair dice $p(x|y) = p(x) = \frac{1}{6}$

Joint Random Variables

- Two or more random variables can be described as an ensemble by means of their joint pmf
  - example: X, Y can be jointly described by likelihoods $p(X = x, Y = y) = p(x, y)$, with $x \in \mathcal{X}$ and $y \in \mathcal{Y}$

Joint Random Variables

- Two or more random variables can be described as an ensemble by means of their joint pmf
  - example: X, Y can be jointly described by likelihoods $p(X = x, Y = y) = p(x, y)$, with $x \in \mathcal{X}$ and $y \in \mathcal{Y}$
  - of course, it must hold that $\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) = 1$

Joint Random Variables

- Two or more random variables can be described as an ensemble by means of their joint pmf
  - example: X, Y can be jointly described by likelihoods $p(X = x, Y = y) = p(x, y)$, with $x \in \mathcal{X}$ and $y \in \mathcal{Y}$
  - of course, it must hold that $\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) = 1$
- Bayes' theorem (product rule of probability):

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x),$$

  - if X and Y are independent, then

$$p(x, y) = p(x)p(y)$$

Joint Random Variables

- Two or more random variables can be described as an ensemble by means of their joint pmf
  - example: X, Y can be jointly described by likelihoods $p(X = x, Y = y) = p(x, y)$, with $x \in \mathcal{X}$ and $y \in \mathcal{Y}$
  - of course, it must hold that $\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) = 1$
- Bayes' theorem (product rule of probability):

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x),$$

  - if X and Y are independent, then

$$p(x, y) = p(x)p(y)$$

- For an information source with successive outcomes modelled by r.v.'s $X_1, X_2 \ldots, X_n$, we say that it is memoryless iff

$$p(X_1 = a_1, \ldots, X_n = a_n) = \Pi_{k=1}^{n} p(X_k = a_k)$$

for all $a_1 \in \mathcal{X}_1, \ldots,\ a_n \in \mathcal{X}_n$

Law of Total Probabilities

- Marginalisation in random variables: obtaining the pmf of a single variable from a joint pmf (consequence of the law of total probabilities)

  $$p(X = x) = \sum_{y \in \mathcal{Y}} p(X = x, Y = y)$$

  for any $x \in \mathcal{X}$
- Conditional probabilities are typically helpful in this computation, using Bayes' law:

  $$p(X = x) = \sum_{y \in \mathcal{Y}} p(X = x | Y = y) p(Y = y)$$

  for any $x \in \mathcal{X}$

Example: Dependent Random Variables

- $\mathcal{X} = \{0, 1\}$, $\mathcal{Y} = \{0, 1\}$
- Assume the following joint probabilities, $p(X = x, Y = y)$:

X \ Y	0	1
0	$\frac{1}{2}$	$\frac{1}{4}$
1	$\frac{1}{8}$	$\frac{1}{8}$

are X and Y independent?

Example: Dependent Random Variables

- $\mathcal{X} = \{0, 1\}$, $\mathcal{Y} = \{0, 1\}$
- Assume the following joint probabilities, $p(X = x, Y = y)$:

X \ Y	0	1
0	$\frac{1}{2}$	$\frac{1}{4}$
1	$\frac{1}{8}$	$\frac{1}{8}$

are X and Y independent?

- $p(X = 0) = \sum_{y \in \mathcal{Y}} p(X = 0, Y = y) = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$
  $p(Y = 0) = \sum_{x \in \mathcal{X}} p(X = x, Y = 0) = \frac{1}{2} + \frac{1}{8} = \frac{5}{8}$

Example: Dependent Random Variables

- $\mathcal{X} = \{0, 1\}$, $\mathcal{Y} = \{0, 1\}$
- Assume the following joint probabilities, $p(X = x, Y = y)$:

X \ Y	0	1
0	$\frac{1}{2}$	$\frac{1}{4}$
1	$\frac{1}{8}$	$\frac{1}{8}$

are X and Y independent?

  - $p(X = 0) = \sum_{y \in \mathcal{Y}} p(X = 0, Y = y) = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$
    $p(Y = 0) = \sum_{x \in \mathcal{X}} p(X = x, Y = 0) = \frac{1}{2} + \frac{1}{8} = \frac{5}{8}$
- So the r.v.'s X and Y are not independent, because $p(X = 0)p(Y = 0) = \frac{15}{32} \neq p(X = 0, Y = 0) = \frac{1}{2}$

Example: Dependent Random Variables

- $\mathcal{X} = \{0, 1\}$, $\mathcal{Y} = \{0, 1\}$
- Assume the following joint probabilities, $p(X = x, Y = y)$:

X \ Y	0	1
0	$\frac{1}{2}$	$\frac{1}{4}$
1	$\frac{1}{8}$	$\frac{1}{8}$

are X and Y independent?

  - $p(X = 0) = \sum_{y \in \mathcal{Y}} p(X = 0, Y = y) = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$
    $p(Y = 0) = \sum_{x \in \mathcal{X}} p(X = x, Y = 0) = \frac{1}{2} + \frac{1}{8} = \frac{5}{8}$
- So the r.v.'s X and Y are not independent, because $p(X = 0)p(Y = 0) = \frac{15}{32} \neq p(X = 0, Y = 0) = \frac{1}{2}$
- Equivalently, using $p(Y = 0|X = 0) = p(X = 0, Y = 0)/p(X = 0)$
  - $p(Y = 0|X = 0) = \frac{2}{3} \neq p(Y = 0) = \frac{5}{8}$

Concepts of Information Theory: Entropy

- How surprising is the outcome x of a single r.v. X?
  - a lot if the outcome is not likely, but not too much if it is likely

Concepts of Information Theory: Entropy

- How surprising is the outcome x of a single r.v. X?
  - a lot if the outcome is not likely, but not too much if it is likely
- So it is convenient to define the "amount of surprise" of $x \in \mathcal{X}$ as inversely related to its likelihood, that is, $1/p(x)$
  - for good reasons, we will take the logarithm of this amount:

$$\log \frac{1}{p(x)} = -\log p(x)$$

Concepts of Information Theory: Entropy

- How surprising is the outcome x of a single r.v. X?
  - a lot if the outcome is not likely, but not too much if it is likely
- So it is convenient to define the "amount of surprise" of $x \in \mathcal{X}$ as inversely related to its likelihood, that is, $1/p(x)$
  - for good reasons, we will take the logarithm of this amount:

$$\log \frac{1}{p(x)} = -\log p(x)$$

- Entropy of a discrete random variable X

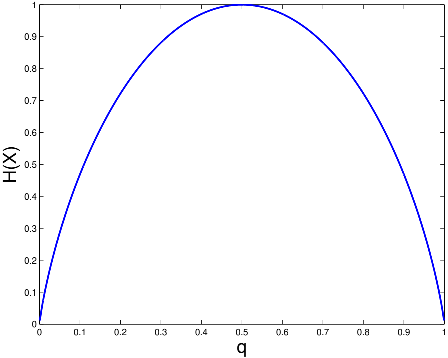
$$H(X) = E(-\log p(X)) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$$

  - interpretation: the average surprise that we have about X
  - equivalently, the uncertainty that we have about X

Example

- If $\mathcal{X} = \{0, 1\}$, with $p(0) = q$ and $p(1) = 1 - q$,

$$H(X) = q \log \frac{1}{q} + (1-q) \log \frac{1}{1-q}$$



Concepts of Information Theory: Entropy

- The units in which the entropy is measured depend on the base of the logarithm assumed:
  - bit: base-2 logarithm (most common, but any can be used)

Concepts of Information Theory: Entropy

- The units in which the entropy is measured depend on the base of the logarithm assumed:
  - bit: base-2 logarithm (most common, but any can be used)
- Why is "bit" (binary digit) the unit of entropy?

  $\rightarrow$ entropy can also be interpreted as the information content of X
  - we will see exactly why when we study source coding
- Intuitive explanation: assume a binary r.v. X
  - if X is deterministic ($p(1) = 1$, $p(0) = 0$), $H(X) = 0$ bit
    - example of successive outcomes of X: 1,1,1,1,1,1,1,1...
    - we know the outcomes of X beforehand so (asymptotically) we need 0 bits per outcome to represent it
  - if X is completely random ($p(0) = p(1) = \frac{1}{2}$), $H(X) = 1$ bit
    - example of successive outcomes of X: 1,0,0,1,0,1,1,0...
    - 1 bit is needed (either a zero or a one) to represent each outcome

Concepts of Information Theory (II)

- Joint entropy of two discrete random variables X and Y

$$
\begin{aligned}
H(X, Y) &= E(-\log p(X, Y)) \\
&= -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log p(x, y)
\end{aligned}
$$

Concepts of Information Theory (II)

- Joint entropy of two discrete random variables X and Y

$$\begin{aligned} H(X,Y) &= E(-\log p(X,Y)) \\ &= -\sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)\log p(x,y) \end{aligned}$$

- Conditional entropy

$$\begin{aligned} H(X|Y) &= E(-\log p(X|Y)) \\ &= -\sum_{x,y\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log p(x|y) \end{aligned}$$

  - how surprised are we about X when we observe Y first?
  - this is the average of $H(X|Y=y)$ for all outcomes y of Y

Concepts of Information Theory (II)

- Joint entropy of two discrete random variables X and Y

$$\begin{aligned} H(X, Y) &= E(-\log p(X, Y)) \\ &= -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log p(x, y) \end{aligned}$$

- Conditional entropy

$$\begin{aligned} H(X|Y) &= E(-\log p(X|Y)) \\ &= -\sum_{x, y \in \mathcal{X} \times \mathcal{Y}} p(x, y) \log p(x|y) \end{aligned}$$

  - how surprised are we about X when we observe Y first?
  - this is the average of $H(X|Y = y)$ for all outcomes y of Y
- Chain rule: $H(X, Y) = H(Y|X) + H(X) = H(X|Y) + H(Y)$

Concepts of Information Theory (III)

- Properties of entropy
  1. $H(X) \geq 0$
  2. $H(X) \leq \log |\mathcal{X}|$
     Proof: first use $\ln x \leq x - 1$ to show that if $\sum_{i=1}^{n} a_i = 1$ and $\sum_{i=1}^{n} b_i = 1$, then $-\sum_i a_i \log a_i \leq -\sum_i a_i \log b_i$, for $a_i, b_i \geq 0$ (Gibbs inequality); then choose the case where all b_i are equal as a particular case

Concepts of Information Theory (III)

- Properties of entropy
  1. $H(X) \geq 0$
  2. $H(X) \leq \log |\mathcal{X}|$
     Proof: first use $\ln x \leq x - 1$ to show that if $\sum_{i=1}^{n} a_i = 1$ and $\sum_{i=1}^{n} b_i = 1$, then $-\sum_i a_i \log a_i \leq -\sum_i a_i \log b_i$, for $a_i, b_i \geq 0$ (Gibbs inequality); then choose the case where all b_i are equal as a particular case
  3. The discrete uniform distribution ($p(x) = 1/|\mathcal{X}|$ for all x) yields $H(X) = \log |\mathcal{X}|$, and maximises entropy
     - no other pmf yields greater entropy
     - most "random" distribution (most unpredictable)

Concepts of Information Theory (III)

- More properties of the entropy
  1. For deterministic variables $H(X) = 0$ (that is, when there is $x \in \mathcal{X}$ such that $p(x) = 1$)
  2. $H(X, Y) \leq H(X) + H(Y)$ (equality holds if X and Y independent)
     - proof: use Gibbs inequality to see that $E(-\log p(X, Y)) \leq E(-\log(p(X)p(Y)))$
  3. Conditioning cannot increase entropy:

     $$H(X|Y) \leq H(X)$$

     Proof: consequence of the chain rule for $H(X, Y)$ and the inequality above

     - $H(X|Y) = H(X)$ iff X and Y are mutually independent

Concepts of Information Theory (IV)

- Mutual information:

$$I(X;Y) = E\left(-\log \frac{p(X)\cdot p(Y)}{p(X,Y)}\right)$$

- In terms of entropies

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

Concepts of Information Theory (IV)

- Mutual information:

$$I(X;Y) = E\left(-\log\frac{p(X)\cdot p(Y)}{p(X,Y)}\right)$$

- In terms of entropies

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

- Possible interpretations of $I(X;Y)$:
  1. the reduction in uncertainty on X due to knowledge of Y
  2. the amount of information about X contained in Y

Concepts of Information Theory (and V)

- Properties of the mutual information
  - $I(X;Y) \geq 0$; proof: conditioning cannot increase entropy
  - X and Y independent iff $I(X;Y) = 0$
  - $I(Y;X) = I(X;Y)$
  - $I(X;X) = H(X)$

Outline of the Chapter

Source Coding

- Source coding: how to efficiently (that is, minimally) represent a random source X using bits or other symbols?
  - equivalently: how to efficiently compress an outcome of a random source X?
  - information theory tells us how well we can hope to do
- Why do we need to know about this for information security?
  - we will see in subsequent lectures that the representation of a source of information matters a lot for security purposes

Source Coding (Definitions)

- Definition: given a finite alphabet $\mathcal{S} = \{0, 1, \cdots, s-1\}$, a code $\mathcal{C}$ on $\mathcal{S}$ is a finite set of chains of elements of $\mathcal{S}$
  - $\mathcal{C} = \{\sigma_1, \sigma_2, \cdots, \sigma_r\}$
  - each chain $\sigma_i = [\sigma_i^{(1)} \cdots \sigma_i^{(N_i)}]$, $\sigma_i^{(k)} \in \mathcal{S}$, is a codeword of $\mathcal{C}$

Source Coding (Definitions)

- Definition: given a finite alphabet $\mathcal{S} = \{0, 1, \cdots, s-1\}$, a code $\mathcal{C}$ on $\mathcal{S}$ is a finite set of chains of elements of $\mathcal{S}$
  - $\mathcal{C} = \{\sigma_1, \sigma_2, \cdots, \sigma_r\}$
  - each chain $\sigma_i = [\sigma_i^{(1)} \cdots \sigma_i^{(N_i)}]$, $\sigma_i^{(k)} \in \mathcal{S}$, is a codeword of $\mathcal{C}$
- Definition: to encode (or to code) a source X is to assign a different codeword to each element $x \in \mathcal{X}$ of the source
  - equivalently: to choose a code $\mathcal{C}$ on $\mathcal{S}$ with as many distinct codewords as elements in the source alphabet ($|\mathcal{C}| = |\mathcal{X}|$)
  - example: $\mathcal{C} = \{00, 11\}$ is a code on $\mathcal{S} = \{0, 1\}$ for a source with two symbols, $\mathcal{X} = \{x_1, x_2\}$

Source Coding (Definitions)

- Definition: given a finite alphabet $\mathcal{S} = \{0, 1, \cdots, s-1\}$, a code $\mathcal{C}$ on $\mathcal{S}$ is a finite set of chains of elements of $\mathcal{S}$
  - $\mathcal{C} = \{\sigma_1, \sigma_2, \cdots, \sigma_r\}$
  - each chain $\sigma_i = [\sigma_i^{(1)} \cdots \sigma_i^{(N_i)}]$, $\sigma_i^{(k)} \in \mathcal{S}$, is a codeword of $\mathcal{C}$
- Definition: to encode (or to code) a source X is to assign a different codeword to each element $x \in \mathcal{X}$ of the source
  - equivalently: to choose a code $\mathcal{C}$ on $\mathcal{S}$ with as many distinct codewords as elements in the source alphabet ($|\mathcal{C}| = |\mathcal{X}|$)
  - example: $\mathcal{C} = \{00, 11\}$ is a code on $\mathcal{S} = \{0, 1\}$ for a source with two symbols, $\mathcal{X} = \{x_1, x_2\}$
- Definition: a product code is the concatenation of all chains in two codes; example:
  - $\mathcal{C} \times \mathcal{C} = \mathcal{C}^2 = \{0000, 0011, 1100, 1111\}$ (one product code)
  - $\mathcal{C}^k = \mathcal{C} \underbrace{\times \cdots \times}_{k-1} \mathcal{C}$ ($k-1$ product codes)

Source Coding (Definitions)

- Definition: a code $\mathcal{C}$ is uniquely decodable if any chain in a product code is uniquely decomposable into codewords
  - example: a code for $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$ which is not uniquely decodable

$\mathcal{X}$	x_1	x_2	x_3	x_4
$\mathcal{C}$	0	010	01	10

  - both x_2 and x_1, x_4 are encoded as 010

Source Coding (Definitions)

- Definition: a code $\mathcal{C}$ is uniquely decodable if any chain in a product code is uniquely decomposable into codewords
  - example: a code for $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$ which is not uniquely decodable

$\mathcal{X}$	x_1	x_2	x_3	x_4
$\mathcal{C}$	0	010	01	10

  - both x_2 and x_1, x_4 are encoded as 010
- Definition: a prefix code (also called instantaneous) is one in which no codeword is a prefix of another codeword

$\mathcal{X}$	x_1	x_2	x_3	x_4
$\mathcal{C}$	0	10	110	1110

  - a prefix code is uniquely decodable, and it can be decoded sequentially without reference to future codewords

Source Coding (Definitions)

- Definition: a code $\mathcal{C}$ is uniquely decodable if any chain in a product code is uniquely decomposable into codewords
  - example: a code for $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$ which is not uniquely decodable

$\mathcal{X}$	x_1	x_2	x_3	x_4
$\mathcal{C}$	0	010	01	10

  - both x_2 and x_1, x_4 are encoded as 010
- Definition: a prefix code (also called instantaneous) is one in which no codeword is a prefix of another codeword

$\mathcal{X}$	x_1	x_2	x_3	x_4
$\mathcal{C}$	0	10	110	1110

  - a prefix code is uniquely decodable, and it can be decoded sequentially without reference to future codewords
- However: not all uniquely decodable codes are instantaneous, take for instance $\{10, 00, 11, 110\}$

Kraft-McMillan Inequality

- Theorem: if $\mathcal{C} = \{\sigma_1, \sigma_2, \cdots, \sigma_r\}$ is a uniquely decodable code on an alphabet $\mathcal{S} = \{0, 1, \cdots, s-1\}$, and if λ_i is the length of σ_i then

$$\sum_{i=1}^{r} s^{-\lambda_i} \leq 1$$

Kraft-McMillan Inequality

- Theorem: if $\mathcal{C} = \{\sigma_1, \sigma_2, \cdots, \sigma_r\}$ is a uniquely decodable code on an alphabet $\mathcal{S} = \{0, 1, \cdots, s-1\}$, and if λ_i is the length of σ_i then

$$\sum_{i=1}^{r} s^{-\lambda_i} \leq 1$$

- Proof: let us write
$\left(\sum_{i=1}^{r} s^{-\lambda_i}\right)^n$

Kraft-McMillan Inequality

- Theorem: if $\mathcal{C} = \{\sigma_1, \sigma_2, \cdots, \sigma_r\}$ is a uniquely decodable code on an alphabet $\mathcal{S} = \{0, 1, \cdots, s-1\}$, and if λ_i is the length of σ_i then

$$\sum_{i=1}^{r} s^{-\lambda_i} \leq 1$$

- Proof: let us write $\left(\sum_{i=1}^{r} s^{-\lambda_i}\right)^n = \sum_{i_1=1}^{r} \cdots \sum_{i_n=1}^{r} s^{-(\lambda_{i_1} + \cdots + \lambda_{i_n})} = \sum_{k=n}^{n\lambda^*} A_k s^{-k}$
  - the last operation just groups terms of same exponent k
  - $\lambda^* = \max_i \lambda_i$ and A_k is the number of n-product codewords of length k

Kraft-McMillan Inequality

- Theorem: if $\mathcal{C} = \{\sigma_1, \sigma_2, \cdots, \sigma_r\}$ is a uniquely decodable code on an alphabet $\mathcal{S} = \{0, 1, \cdots, s-1\}$, and if λ_i is the length of σ_i then

$$\sum_{i=1}^{r} s^{-\lambda_i} \leq 1$$

- Proof: let us write
$\left(\sum_{i=1}^{r} s^{-\lambda_i}\right)^n = \sum_{i_1=1}^{r} \cdots \sum_{i_n=1}^{r} s^{-(\lambda_{i_1} + \cdots + \lambda_{i_n})} = \sum_{k=n}^{n\lambda^*} A_k s^{-k}$
  - the last operation just groups terms of same exponent k
  - $\lambda^* = \max_i \lambda_i$ and A_k is the number of n-product codewords of length k
  - uniquely decodable $\mathcal{C} \Rightarrow A_k \leq s^k$, then $\sum_{i=1}^{r} s^{-\lambda_i} \leq (n\lambda^* - n + 1)^{\frac{1}{n}}$
  - letting $n \to \infty$ we get the Kraft-McMillan inequality

Source Coding

- Given a random source X, what is the best way to assign codewords to the source symbols $x \in \mathcal{X}$?
- Criterion: average code length

$$\bar{\lambda} = E(\lambda_X) = \sum_{x \in \mathcal{X}} p(x)\lambda_x,$$

λ_X: random variable associated to codeword length

Source Coding

- Given a random source X, what is the best way to assign codewords to the source symbols $x \in \mathcal{X}$?
- Criterion: average code length

$$\bar{\lambda} = E(\lambda_X) = \sum_{x \in \mathcal{X}} p(x)\lambda_x,$$

λ_X: random variable associated to codeword length

- Example: $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$, $\mathcal{S} = \{0, 1\}$
  - assume that the source symbols are uniformly distributed, that is, $p(x_i) = 1/4$, and the following code $\mathcal{C}$

x	σ_x	λ_x
x_1	00	2
x_2	01	2
x_3	10	2
x_4	11	2

  - this yields $\bar{\lambda} = 2$ (but could we possibly get $\bar{\lambda} < 2$?)

Source Coding (II)

- We can't with a uniform distribution of X, but now assume $p(x_1) = 0.7$ and $p(x_2) = p(x_3) = p(x_4) = 0.1$
- We can now get $\bar{\lambda} = 1.6 < 2$ using

x	σ_x	λ_x
x_1	0	1
x_2	100	3
x_3	101	3
x_4	110	3

Source Coding (II)

- We can't with a uniform distribution of X, but now assume $p(x_1) = 0.7$ and $p(x_2) = p(x_3) = p(x_4) = 0.1$
- We can now get $\bar{\lambda} = 1.6 < 2$ using

x	σ_x	λ_x
x_1	0	1
x_2	100	3
x_3	101	3
x_4	110	3

- Therefore source coding can exploit less randomness in the source (equivalently, more redundancy in the source)
  - in the first example we could not find a code with less than $\bar{\lambda} = 2$, and at the same time the entropy was maximum ($H(X) = 2$ bits), since the symbols were uniformly distributed
  - what is the connection between source coding and entropy?

Optimal Source Coding

- Given any uniquely decodable code $\mathcal{C}$ on $\mathcal{S}$ to encode X

$$\bar{\lambda} \geq H(X)$$

(using base s for the logarithms in the entropy)

Optimal Source Coding

- Given any uniquely decodable code $\mathcal{C}$ on $\mathcal{S}$ to encode X

$$\bar{\lambda} \geq H(X)$$

(using base s for the logarithms in the entropy)

- Proof:

$$H(X) - \bar{\lambda} = -\sum_x p(x) \log p(x) - \sum_x p(x) \lambda_x$$
$$= \sum_x p(x) \log \frac{s^{-\lambda_x}}{p(x)} \leq \frac{1}{\ln s} \left(\left(\sum_x s^{-\lambda_x} \right) - 1 \right) \leq 0$$

- the first inequality uses $\ln x \leq x - 1$, and the second one is the Kraft-McMillan inequality

Optimum Source Coding and Compression

- Optimum source coding $\leftrightarrow$ removing all source redundancy
  - one cannot compress any source beyond its entropy (lossless compression)
  - we know that $H(X) \leq \log |\mathcal{X}|$; we can always assign $\log |\mathcal{X}|$ bits to each source symbol, but this may be inefficient
  - example: taking the 26 letters of the Latin alphabet $\mathcal{X} = \{x_1, x_2, \cdots, x_{26}\}$, we can always encode text using $\log_2 |\mathcal{X}| = \log_2 26 = 4.7$ bits/letter (5 in practice), but this is much higher than the typical entropy of these letters in a natural language (in English, $H(X) \approx 1.5$ bits/letter)

Optimum Source Coding and Compression

- Optimum source coding $\leftrightarrow$ removing all source redundancy
  - one cannot compress any source beyond its entropy (lossless compression)
  - we know that $H(X) \leq \log|\mathcal{X}|$; we can always assign $\log|\mathcal{X}|$ bits to each source symbol, but this may be inefficient
  - example: taking the 26 letters of the Latin alphabet $\mathcal{X} = \{x_1, x_2, \cdots, x_{26}\}$, we can always encode text using $\log_2|\mathcal{X}| = \log_2 26 = 4.7$ bits/letter (5 in practice), but this is much higher than the typical entropy of these letters in a natural language (in English, $H(X) \approx 1.5$ bits/letter)
- If a source code achieves $\bar{\lambda} = H(X)$, then it is optimum (optimum compression)
  - coding efficiency of a source code: $\eta = H(X)/\bar{\lambda} \leq 1$

Optimum Source Coding and Compression

- Shannon code: choose a source code such that $\lambda_x = \lceil \log_s 1/p(x) \rceil$ ($\lceil \cdot \rceil$ means round upwards), to get something close to $s^{-\lambda_x} = p(x)$
  - this always guarantees that $\bar{\lambda} < H(X) + 1$
  - it gives exactly $H(X)$ if all $p(x)$ are negative powers of s

Optimum Source Coding and Compression

- Shannon code: choose a source code such that $\lambda_x = \lceil \log_s 1/p(x) \rceil$ ($\lceil \cdot \rceil$ means round upwards), to get something close to $s^{-\lambda_x} = p(x)$
  - this always guarantees that $\bar{\lambda} < H(X) + 1$
  - it gives exactly $H(X)$ if all $p(x)$ are negative powers of s
- A Shannon code is not necessarily optimum; example

X	codeword length	codeword
$p(x_1) = 0.6$	$\lambda_{x_1} = \lceil \log_2(1/0.6) \rceil = 1$ bits	$\sigma_1 = 0$
$p(x_2) = 0.3$	$\lambda_{x_2} = \lceil \log_2(1/0.3) \rceil = 2$ bits	$\sigma_2 = 10$
$p(x_3) = 0.1$	$\lambda_{x_3} = \lceil \log_2(1/0.1) \rceil = 4$ bits	$\sigma_3 = 1100$

  - $H(X) = 1.29$ bits/symbol, $\bar{\lambda} = 1.6$ bits/symbol $\rightarrow \eta = 0.81$

Optimum Source Coding and Compression

- Shannon code: choose a source code such that $\lambda_x = \lceil \log_s 1/p(x) \rceil$ ($\lceil \cdot \rceil$ means round upwards), to get something close to $s^{-\lambda_x} = p(x)$
  - this always guarantees that $\bar{\lambda} < H(X) + 1$
  - it gives exactly $H(X)$ if all $p(x)$ are negative powers of s
- A Shannon code is not necessarily optimum; example

X	codeword length	codeword
$p(x_1) = 0.6$	$\lambda_{x_1} = \lceil \log_2(1/0.6) \rceil = 1$ bits	$\sigma_1 = 0$
$p(x_2) = 0.3$	$\lambda_{x_2} = \lceil \log_2(1/0.3) \rceil = 2$ bits	$\sigma_2 = 10$
$p(x_3) = 0.1$	$\lambda_{x_3} = \lceil \log_2(1/0.1) \rceil = 4$ bits	$\sigma_3 = 1100$

  - $H(X) = 1.29$ bits/symbol, $\bar{\lambda} = 1.6$ bits/symbol $\rightarrow \eta = 0.81$
  - but, by inspection, we could have gotten $\bar{\lambda} = 1.4$ bits/symbol

    with

σ_1	σ_2	σ_3
0	10	11

    $(\eta = 0.92)$

Optimal Source Coding and Compression

- What is the closest to $\eta = 1$ that we can get?
  - clearly, shorter codewords should go to most likely symbols
- Huffman coding (optimum prefix code), for alphabet size s:
  1. recursively group source symbols yielding the s smallest probabilities until only s are left
  2. work one's way backwards, by assigning s code symbols to the last group, then s symbols to the previous group, etc
- **Example**: let $p_1 \geq p_2 \geq p_3 \geq p_4$ be the alphabet probabilities (i.e. $p_i = p(x_i)$); assume that $p_3 + p_4 \leq p_2$, $p_2 + p_3 + p_4 \leq p_1$, and $s = 2$

p_1		
p_2		
p_3		
p_4		

Optimal Source Coding and Compression

- What is the closest to $\eta = 1$ that we can get?
  - clearly, shorter codewords should go to most likely symbols
- Huffman coding (optimum prefix code), for alphabet size s:
  1. recursively group source symbols yielding the s smallest probabilities until only s are left
  2. work one's way backwards, by assigning s code symbols to the last group, then s symbols to the previous group, etc
- **Example**: let $p_1 \geq p_2 \geq p_3 \geq p_4$ be the alphabet probabilities (i.e. $p_i = p(x_i)$); assume that $p_3 + p_4 \leq p_2$, $p_2 + p_3 + p_4 \leq p_1$, and $s = 2$

p_1		
p_2		
p_3		
p_4		

Optimal Source Coding and Compression

- What is the closest to $\eta = 1$ that we can get?
  - clearly, shorter codewords should go to most likely symbols
- Huffman coding (optimum prefix code), for alphabet size s:
  1. recursively group source symbols yielding the s smallest probabilities until only s are left
  2. work one's way backwards, by assigning s code symbols to the last group, then s symbols to the previous group, etc
- **Example**: let $p_1 \geq p_2 \geq p_3 \geq p_4$ be the alphabet probabilities (i.e. $p_i = p(x_i)$); assume that $p_3 + p_4 \leq p_2$, $p_2 + p_3 + p_4 \leq p_1$, and $s = 2$

p_1	p_1	
p_2	p_2	
p_3	$p_3 + p_4$	
p_4		

Optimal Source Coding and Compression

- What is the closest to $\eta = 1$ that we can get?
  - clearly, shorter codewords should go to most likely symbols
- Huffman coding (optimum prefix code), for alphabet size s:
  1. recursively group source symbols yielding the s smallest probabilities until only s are left
  2. work one's way backwards, by assigning s code symbols to the last group, then s symbols to the previous group, etc
- **Example**: let $p_1 \geq p_2 \geq p_3 \geq p_4$ be the alphabet probabilities (i.e. $p_i = p(x_i)$); assume that $p_3 + p_4 \leq p_2$, $p_2 + p_3 + p_4 \leq p_1$, and $s = 2$

p_1	p_1	p_1
p_2	p_2	$p_2 + (p_3 + p_4)$
p_3	$p_3 + p_4$	
p_4		

Optimal Source Coding and Compression

- What is the closest to $\eta = 1$ that we can get?
  - clearly, shorter codewords should go to most likely symbols
- Huffman coding (optimum prefix code), for alphabet size s:
  1. recursively group source symbols yielding the s smallest probabilities until only s are left
  2. work one's way backwards, by assigning s code symbols to the last group, then s symbols to the previous group, etc
- **Example**: let $p_1 \geq p_2 \geq p_3 \geq p_4$ be the alphabet probabilities (i.e. $p_i = p(x_i)$); assume that $p_3 + p_4 \leq p_2$, $p_2 + p_3 + p_4 \leq p_1$, and $s = 2$

p_1	p_1	p_1	0
p_2	p_2	$p_2 + (p_3 + p_4)$	1
p_3	$p_3 + p_4$		
p_4			

Optimal Source Coding and Compression

- What is the closest to $\eta = 1$ that we can get?
  - clearly, shorter codewords should go to most likely symbols
- Huffman coding (optimum prefix code), for alphabet size s:
  1. recursively group source symbols yielding the s smallest probabilities until only s are left
  2. work one's way backwards, by assigning s code symbols to the last group, then s symbols to the previous group, etc
- **Example**: let $p_1 \geq p_2 \geq p_3 \geq p_4$ be the alphabet probabilities (i.e. $p_i = p(x_i)$); assume that $p_3 + p_4 \leq p_2$, $p_2 + p_3 + p_4 \leq p_1$, and $s = 2$

p_1	p_1	0	p_1	0
p_2	p_2	10	$p_2 + (p_3 + p_4)$	1
p_3	$p_3 + p_4$	11		
p_4				

Optimal Source Coding and Compression

- What is the closest to $\eta = 1$ that we can get?
  - clearly, shorter codewords should go to most likely symbols
- Huffman coding (optimum prefix code), for alphabet size s:
  1. recursively group source symbols yielding the s smallest probabilities until only s are left
  2. work one's way backwards, by assigning s code symbols to the last group, then s symbols to the previous group, etc
- **Example**: let $p_1 \geq p_2 \geq p_3 \geq p_4$ be the alphabet probabilities (i.e. $p_i = p(x_i)$); assume that $p_3 + p_4 \leq p_2$, $p_2 + p_3 + p_4 \leq p_1$, and $s = 2$

p_1	0	p_1	0	p_1	0
p_2	10	p_2	10	$p_2 + (p_3 + p_4)$	1
p_3	110	$p_3 + p_4$	11		
p_4	111				

Distribution of Encoded Symbols

- Notes:
  - optimum Huffman code is not unique (because of ties)
  - probabilities need to be known *a priori*
  - source symbols assumed mutually independent (memoryless source)

Distribution of Encoded Symbols

- Notes:
  - optimum Huffman code is not unique (because of ties)
  - probabilities need to be known *a priori*
  - source symbols assumed mutually independent (memoryless source)
- If a source is near-optimally encoded then the entropy of the encoded symbols should be $H(S) \approx \log s$ (where $s = |\mathcal{S}|$)
  - equivalently: a good compression method yields a stream of symbols that look as random as possible (i.e. uniform)
  - an optimally compressed bitstream should not be compressible!