Information Theory: Principles and Applications

Tiago T. V. Vinhoza

March 26, 2010

Jensen's Inequality

- If $f(\cdot)$ is a convex function and X is a random variable

$$E[f(X)] \geq f(E[X])$$

- Let us now show that relative entropy and mutual information are greater than zero and some other interesting properties of the information measures.

Log-Sum Inequality

- For n positive numbers $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots b_n$

$$\sum_{i=1}^{n} a_i \log \frac{a_i}{b_i} \geq \left(\sum_{i=1}^{n} a_i \right) \log \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}$$

  with equality if and only if $a_i/b_i = c$.
- Let us now prove the convexity of the relative entropy and the concavity of the entropy.

Fano's Inequality

- Suppose we know a random variable Y and we wish to guess the value of a correlated random variable X.
- Fano's inequality relates the probability of error in guessing X from Y to its conditional entropy $H(X|Y)$.
- Let $\hat{X} = g(Y)$, if $P_e = P(\hat{X} \neq X)$, then

$$H(P_e) + P_e \log(|\mathcal{X}| - 1) \geq H(X|Y)$$

where $H(P_e)$ is the binary entropy function evaluated at P_e.

Source Coding

- From the previous lecture: "A source encoder converts the sequence of symbols from the source into a sequence of bits".
- Types of source:
  - Discrete: keyboard characters, bits, ...
  - Continous (time, amplitude): speech
  - Continuous amplitude, discrete time: sampled signal before quantization

Source Coding: Continous Sources

- For continuous-amplitude sources, there is usually no way to map the source values to a bit sequence such that the map is uniquely decodable.
- For example: the set of real numbers between 0 and 1 requires infinitely many binary digits for exact specification.
- Quantization is necessary $\rightarrow$ distortion introduced.
- Source encoding: trade off between the bit rate and the level of distortion.

Source Coding: Discrete Memoryless Sources

- A discrete memoryless source (DMS) is defined by the following properties:
  - The source output is an unending sequence $X_1, X_2, X_3, \ldots$ of randomly selected letters from $\mathcal{X}$.
  - Each source output is selected from $\mathcal{X}$ using a common probability measure.
  - Each source output X_i is statistically independent of the other source outputs X_j, $j \neq i$.

Source Coding: Discrete Random Variables

- A *source code* $\mathcal{C}$ for a discrete random variable X is a mapping from $\mathcal{X}$, the range of X, to $\mathcal{D}^*$, the set of finite length strings of symbols from a D-ary alphabet. Let $\mathcal{C}(x)$ denote the codeword corresponding to x and let $l(x)$ denote the length of $\mathcal{C}(x)$.

Fixed Length Source Codes

- Convert each source letter individually into a fixed-length block of L bits.
- There are 2^L different combinations.
- If the number of letters in the source alphabet $\mathcal{X}$ is less or equal to 2^L then a different binary $L-$tuple may be assigned to each source symbol.
- Uniquely decoded from the binary blocks, and the code is uniquely decodable.

Fixed Length Source Codes

- Requires $L = \lceil \log |\mathcal{X}| \rceil$ bits to encode each source letter.
- Hence $\log |\mathcal{X}| \leq L < \log |\mathcal{X}| + 1$
- For blocks of n symbols. The n-tuple source alphabet is then the n-fold Cartesian product $\mathcal{X}^n = \mathcal{X} \times \mathcal{X} \times \ldots \times \mathcal{X}$.
- $|\mathcal{X}^n| = |\mathcal{X}|^n$.
- Each source n-tuple can be coded into $L = n \log |\mathcal{X}|$ bits.

Fixed Length Source Codes

- Rate $\overline{L}$ of coded bits per source symbol:

$$\overline{L} = \frac{L}{n}$$

- Bounds:

$$\log|\mathcal{X}| \leq \overline{L} < \log|\mathcal{X}| + \frac{1}{n}$$

- Letting n become sufficiently large, the average number of coded bits required per source symbol can be made arbitrarily close to $\log|\mathcal{X}|$
- This method is nonprobabilistic; it does not takes into account if some symbols occur more frequently than others.

Variable Length Source Codes

- Intuition: Allocate the shortest codewords to the most probable outcomes and the longer ones to the least likely outcomes.
- Example: Morse code.

Variable Length Source Codes

- Codewords of a variable-length source code: a continuing sequence of bits, with no demarcations of codeword boundaries.
- The source decoder, given an original starting point, must determine where the codeword boundaries are (parsing).

Classes of Codes

- Non-singular code

$$x_i \neq x_j \rightarrow \mathcal{C}(x_i) \neq \mathcal{C}(x_j)$$

- Unambiguous for a single symbol.
- Example of a non-singular code. For a binary valued random variable X:

$$\mathcal{C}(x_1) = 0 \qquad \mathcal{C}(x_2) = 1.$$

- Example of a singular code. For a binary valued random variable X:

$$\mathcal{C}(x_1) = 0 \qquad \mathcal{C}(x_2) = 0.$$

Classes of Codes

- Definition: Extension of a code

$$\mathcal{X}^n \rightarrow \mathcal{D}^{*n} : \mathcal{C}(x_1 x_2 \ldots x_n) = \mathcal{C}(x_1)\mathcal{C}(x_2)\ldots\mathcal{C}(x_n)$$

- Example: $\mathcal{C}(x_1) = 00$, $\mathcal{C}(x_2) = 11$, $\mathcal{C}(x_1 x_2) = 0011$.
- The extension of an uniquely decodable code is singular.
- Example

$$\mathcal{C}(x_1) = 0 \qquad \mathcal{C}(x_2) = 1.$$

- Example of a non uniquely decodable code:

$$\mathcal{C}(x_1) = 0 \quad \mathcal{C}(x_2) = 1 \quad \mathcal{C}(x_3) = 10.$$

- Example: $\mathcal{C}(x_2 x_1 x_3) = \mathcal{C}(x_2 x_1 x_2 x_1) = 1010$.

Classes of Codes

- Prefix-free Codes: no codeword is a prefix of any other codeword
- They are also called instantaneous because the source symbol with essentially no delay. As soon as the entire codeword is received at the decoder, it can be recognized as a codeword and decoded without waiting for additional bits.
- It is very easy to check whether a code is prefix-free, and therefore uniquely decodable.
- Leafs of the code tree.

Classes of Codes

- All Codes
- Singular Codes
- Uniquely Decodable Codes
- Prefix-free Codes

Kraft Inequality

- It tells us about the possibilty of constructing a prefix-free code for a given source with alphabet $\mathcal{X}$ with a given set of codeword lengths $l(x_i), x_i \in \mathcal{X}$.

$$\sum_{x_i \in \mathcal{X}} D^{-l(x_i)} \leq 1$$

- For the binary case, $D = 2$, there exists a full prefix-free code with codeword lengths $\{1, 2, 2\}$.
- On the other hand a prefix-free code with codeword lengths $\{1, 1, 2\}$ does not exist in the binary case.

Minimum $\overline{L}$ for prefix-free codes

- Kraft Inequatilty: determines which sets of coderword lengths are possible for prefix-free codes.
- What set of codewords can be used to *minimize* the expected length of a prefix-free code?
- Constrained optimization problem

$$\min_{\text{s.t. Kraft Inequality}} \overline{L}$$

Minimum $\overline{L}$ for prefix-free codes

- Entropy Bounds

$$H(X) \leq \overline{L}_{min} \leq H(X) + 1$$

Huffman Codes

- Result of an Information Theory class project.
- Huffman ignored the Kraft inequality and focused on the code tree to establish propertiess that an optimum prefix-free code should have.

Binary Huffman Codes

- Optimum codes have the property that if $p_i > p_j$, then $l(x_i) \leq l(x_j)$.
- Code tree is full.
- Longest codeword has a sibling that is another longest codeword. (a sibling differ in the final bit)
- Let X be a random symbol with a pmf satisfying $p_1 \geq p_2 \geq \ldots \geq p_M$. There is an optimal prefix free code for X in which the codewords for $M-1$ and M are siblings and have maximal length within the code.

Huffman Codes: An example

- Probability distribution $(0.4; 0.2; 0.15; 0.15; 0.1)$

Asymptotic Equipartition Property

- In Information Theory, the analog of the law of the large numbers is the Asymptotic Equipartition Property (AEP).
- The AEP says that, given a very long string of n independent and identically distributed discrete random variables $X_1, \ldots, X_n$ there exists a *typical set* of sample strings $(x1; \ldots, x_n)$ whose aggregate probability is almost 1.
- There are roughly $2^{nH(X)}$ typical strings of length n, and each has a probability roughly equal to $2^{-nH(X)}$
- "Almost all events are equally surprising".
- First, let's review the weak law of large numbers.

Asymptotic Equipartition Property

- Weak Law of Large Numbers.
- Let $X_1, \ldots, X_n$ be a sequence of independent and equally distributed random variables.

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \qquad \text{sample average}$$

- Chebyshev inequality: Let X be a random variable with mean m_X and variance σ_X^2 , then $P(|X - m_X| \geq \epsilon) \leq \sigma_X^2/\epsilon^2$.
- Applying this inequality to the sample mean, we have

$$P(|\overline{X} - m_X| \geq \epsilon) \leq \sigma_X^2/n\epsilon^2$$

- Remember that $E[\overline{X}] = m_X$ and $var(\overline{X}) = \sigma_X^2/n$.

Asymptotic Equipartition Property

- Let $X_1, \ldots, X_n$ be a sequence of discrete independent and equally distributed random variables over $\mathcal{X}$.
- Note that $w(x) = -\log p_X(x)$ is a real valued funcion of $x \in \mathcal{X}$.
- $W(X_i)$ is a random variable that takes the value $w(x)$ for $X = x$.
- Let $W(X_1), \ldots, W(X_n)$ is a sequence of random variables.

$$E[W(X_i)] = \sum_{x \in \mathcal{X}} p_X(x) \log p_X(x) = H(X)$$

- We have that for independent random variables.

$$w(x_1) + w(x_2) = -\log p_X(x_1) - \log p_X(x_2) = -\log p_{X_1 X_2}(x_1, x_2)$$

Asymptotic Equipartition Property

- For a general n: $\sum_{i=1}^{n} w(x_i) = -\sum_{i=1}^{n} \log p_X(x_i) = -\log p_{\mathbf{X}^n}(\mathbf{x}^n)$, where $\mathbf{X}^n = [X_1, \ldots X_n]$ and $\mathbf{x}^n = [x_1, \ldots x_n]$.
- Let's do the sample average of those random variables $W(X_i)$

$$\overline{W} = \frac{1}{n}\sum_{i=1}^{n} W(X_i) = \frac{-\log p_{\mathbf{X}^n(\mathbf{x}^n)}}{n}$$

- Using Chebyshev's inequality we get

$$P\left(\left|\frac{-\log p_{\mathbf{X}^n}(\mathbf{x}^n)}{n} - H(X)\right| \geq \epsilon\right) \leq \sigma_W^2/n\epsilon^2$$

Asymptotic Equipartition Property

- The *typical set* $A_\epsilon^{(n)}$ with respect to $p_X(x)$ is the set of sequences $(x_1, x_2, \ldots, x_n) \in \mathcal{X}^n$ with the following property:

$$A_\epsilon^{(n)} = \left\{ \mathbf{x}^n : \left| \frac{-\log p_{\mathbf{X}^n}(\mathbf{x})}{n} - H(X) \right| \leq \epsilon \right\}$$

- Which can be written as:

$$-n(H(X) + \epsilon) \leq \log p_{\mathbf{X}^n}(\mathbf{x}^n) \leq -n(H(X) - \epsilon)$$

$$2^{-n(H(X)+\epsilon)} \leq p_{\mathbf{X}^n}(\mathbf{x}^n) \leq 2^{-n(H(X)-\epsilon)}$$

Asymptotic Equipartition Property

- Properties of the typical set:
  - $P(\mathbf{X}^n \in A_\epsilon^{(n)}) > 1 - \frac{\sigma_W^2}{n\epsilon}$ for n sufficient large

$$P(\mathbf{X}^n \in A_\epsilon^{(n)}) = P\left(\left|\frac{-\log p_{\mathbf{X}^n}(\mathbf{x}^n)}{n} - H(X)\right| \leq \epsilon\right)$$

$$P(\mathbf{X}^n \in A_\epsilon^{(n)}) \geq 1 - \frac{\sigma_W^2}{n\epsilon}$$

Asymptotic Equipartition Property

- Properties of the typical set:
  - $|A_\epsilon^{(n)}| \leq 2^{n(H(X)+\epsilon)}$

$$
\begin{aligned}
1 &= \sum_{\mathbf{x}^n \in \mathcal{X}^n} p_{\mathbf{X}^n}(\mathbf{x}^n) \\
&\geq \sum_{\mathbf{x}^n \in A_\epsilon^{(n)}} p_{\mathbf{X}^n}(\mathbf{x}^n) \\
&\geq \sum_{\mathbf{x}^n \in A_\epsilon^{(n)}} 2^{-n(H(X)-\epsilon)} \\
&\geq 2^{-n(H(X)-\epsilon)} \sum_{\mathbf{x}^n \in A_\epsilon^{(n)}} 1 \\
&\geq 2^{-n(H(X)-\epsilon)} |A_\epsilon^{(n)}|
\end{aligned}
$$

Asymptotic Equipartition Property

- Properties of the typical set:
  - $|A_\epsilon^{(n)}| \geq (1-\delta)2^{n(H(X)-\epsilon)}$, where $\delta = \frac{\sigma_W^2}{n\epsilon^2}$

$$
\begin{aligned}
(1-\delta) &\leq P(\mathbf{X}^n \in A_\epsilon^{(n)}) \\
&\leq \sum_{\mathbf{x}^n \in A_\epsilon^{(n)}} 2^{-n(H(X)-\epsilon)} \\
&= 2^{-n(H(X)-\epsilon)} |A_\epsilon^{(n)}|
\end{aligned}
$$

Asymptotic Equipartition Property: Summary

- Definition of typical set:

$$2^{-n(H(X)+\epsilon)} \leq p_{\mathbf{X}^n}(\mathbf{x}^n) \leq 2^{-n(H(X)-\epsilon)}$$

- Size of typical set:

$$(1-\delta)2^{n(H(X)-\epsilon)} \leq |A_\epsilon^{(n)}| \leq 2^{n(H(X)+\epsilon)}$$

Source coding in the light of the AEP

- A source coder operating on strings of n source symbols need only provide a codeword for each string $\mathbf{x}^n$ in the typical set $A_\epsilon^{(n)}$.
- That will be shown next class.