Data compression

Harald Nautsch
harald.nautsch@liu.se
ISY Informationskodning, Linköpings universitet

http://isy.gitlab-pages.liu.se/icg/en/courses/TSBK08/

Course contents

Source modeling:
Random variables and random processes as source models.

Source coding theory:
Definition of a code. Code classes.

Information theory:
Definitions of information and entropy. The entropy gives a theoretical limit on how much a signal from a random source can be compressed without getting any distortion.

Practical compression methods:
Huffman coding, Tunstall coding, arithmetic coding, Golomb codes, Lempel-Ziv-coding, Burrows-Wheeler-coding.
pack, compress, zip, gzip, bzip, GIF, PNG, fax coding, lossless JPEG, JPEG-LS, etc.

Rate-distortion theory:
Theoretical limits for lossy coding.

Lectures, preliminary program

1. Introduction. Sources. Random source models. Source coding.
2. Source coding. Information theory
3. Information theory. Optimal codes. Huffman coding.
4. Adaptive Huffman coding. Run length coding. Golomb codes.
5. Arithmetic coding.
6. Adaptive arithmetic coding. ppm. Binary arithmetic coding.
7. Lempel-Ziv-coding.
8. Burrows-Wheelers block transform. Tunstall coding.
9. Differential entropy. Rate-distortion theory.
10. Lossy transform coding.

Examination

- Small project lab (2hp).
  Implementation of some of the methods that are introduced in the course. Testing on real data (text, images, executable files, etc.). Entropy estimation. Work in groups of 1-3 students. Examination by written report.
- Written exam (4hp).

Sources

A *source* is something that produces a sequence of *symbols*.

The symbols are elements of a discrete *alphabet* $\mathcal{A} = \{a_1, a_2, \ldots, a_L\}$ of size L.

Most of the time we will have finite alphabets, but infinite alphabets are also allowed.

In many cases we only have access to a symbol sequence and need to model the source from the sequence.

The source models we will concentrate on are *random (stochastic) models*, where we assume that the symbols are produced from random variables or random processes.

Random variables

The *sample space* Ω is the set of possible outcomes of a random experiment, $\Omega = \{\omega_1, \omega_2, \dots, \omega_n\}$

Every subset of Ω is called an *event*. We have a measure P (probability) on the events

A *random variable* X is a mapping from the sample space to the alphabet $\mathcal{A}$

$$X : \Omega \to \mathcal{A}$$

We write $\{X = x\}$ for the event $\{\omega : X(\omega) = x\}$, but $P(X = x)$ instead of $P(\{X = x\})$.

Random variables, cont.

The *probability function* p_X

$$p_X(x) = P(X = x),\ x \in \mathcal{A}$$

We have $p_X(x) \geq 0$ for all x and

$$\sum_{x \in \mathcal{A}} p_X(x) = 1$$

For a real function $f(X)$ of a random variable X, the *average value (expected value)* is the real number

$$E\{f(X)\} \triangleq \sum_{x \in \mathcal{A}} f(x) \cdot p_X(x)$$

Random variables, cont.

We can also have $X = (Y, Z)$ where Y and Z are random variables with alphabets $\mathcal{A}_Y = \{y_1, \ldots, y_M\}$ and $\mathcal{A}_Z = \{z_1, \ldots, z_N\}$. X takes values in $\mathcal{A}_X = \{(y_1, z_1), (y_1, z_2), \ldots, (y_M, z_N)\}$.

Normally we write $p_{YZ}(y, z)$ instead of $p_X((y, z))$.

We can of course generalize this to $X = (X_1, X_2, \ldots, X_K)$.

The random variables Y and Z are called *independent* if

$$p_{YZ}(y, z) = p_Y(y) \cdot p_Z(z), \ \forall y, z$$

The *conditional probability function* $p_{X|Y}$ is defined as

$$p_{X|Y}(x|y) \ \stackrel{\triangle}{=} \ \frac{p_{XY}(x, y)}{p_Y(y)}$$

when $p_Y(y) > 0$.

Random sources

A source is modeled as a random process X_n (can also be seen as a sequence of random variables)

$$\ldots, X_{-1}, X_0, X_1, X_2, \ldots$$

or

$$X_0, X_1, X_2, \ldots$$

Most of the time we are only interested in *stationary* sources, ie when all probability functions are independent of n. For example:

$$p_{X_n X_{n+1}} = p_{X_{n+k} X_{n+k+1}}$$

If X_n and X_m are independent for all $n \neq m$ the source is called *memoryless*, otherwise we say that the source has *memory*.

Markov chains

A *Markov chain* is a memory source with limited memory one step back in the sequence.

$$p(x_n|x_{n-1}x_{n-2}\ldots) = p(x_n|x_{n-1})$$

If the alphabet is $\mathcal{A} = \{a_1, a_2, \ldots, a_L\}$, the Markov chain can be described as a state model with L states (a_i) where we at time n move from state (x_{n-1}) to state (x_n) with probability $p(x_n|x_{n-1})$. These conditional probabilities are referred to as *transition probabilities*

Markov chains, cont.

The Markov chain can be described using its starting state and its *transition matrix* **P**. This quadratic matrix has in row r and column c the transition probability from state a_r to a_c.

If it is possible to move, with positive probability, from every state to every other state in a finite number of steps, the Markov chain is called *irreducible*.

If we at time n are in state s_i with the probability p_i^n, we can calculate the probabilities for time $n+1$ as

$$[p_1^{n+1} \; p_2^{n+1} \; \ldots \; p_L^{n+1}] = [p_1^n \; p_2^n \; \ldots \; p_L^n] \cdot \mathbf{P}$$

A distribution over the states such that the distribution at time $n+1$ is the same as at time n is called a *stationary distribution*.

If the Markov chain is irreducible and aperiodic the stationary distribution is unique and every starting distribution will approach the stationary distribution as the time goes to infinity.

Stationary distribution

We denote the stationary probabilities w_i and define the row vector

$$\bar{w} = (w_1, w_2, \ldots, w_L)$$

If the stationary distribution exists, it can be found as the solution of the equation system

$$\bar{w} = \bar{w} \cdot \mathbf{P}$$

or

$$\bar{w} \cdot (\mathbf{P} - \mathbf{I}) = \bar{0}$$

This equation system is under-determined (if $\bar{w}$ is a solution then $c \cdot \bar{w}$ is also a solution, for any constant c). To find the correct solution we add the equation $\sum_{j=1}^{L} w_j = 1$ (w_j are probabilities, therefore their sum is 1).

(If you prefer equation systems with column vectors, you can just transpose the entire expression and solve $\bar{w}^T = \mathbf{P}^T \cdot \bar{w}^T$ instead.)

Markov sources of higher order

A *Markov source* of order k is a memory source with limited memory k steps back in the sequence.

$$p(x_n|x_{n-1}x_{n-2}\dots) = p(x_n|x_{n-1}\dots x_{n-k})$$

If the alphabet is $\mathcal{A} = \{a_1, a_2, \dots, a_L\}$, the Markov source can be described as a state model with L^k states $(x_{n-1}\dots x_{n-k})$ where we at time n move from state $(x_{n-1}\dots x_{n-k})$ to state $(x_n \dots x_{n-k+1})$ with probability $p(x_n|x_{n-1}\dots x_{n-k})$. These probabilities are called *transition probabilities*

The sequence of states is a random process $S_n = (X_n \dots X_{n-k+1})$ with alphabet $\mathcal{B} = \mathcal{A}^k = \{b_1, b_2, \dots, b_{L^k}\}$ of size L^k.

The state process S_n is a Markov chain and thus we can use the same methods for finding stationary distributions as previously.

Markov sources, cont.

The Markov source can be described using its starting state and its *transition matrix* $\mathbf{P}$. This quadratic matrix has in row r and column k the transition probability from state b_r to b_c.

If we at time n are in state s_i with the probability p_i^n, we can calculate the probabilities for time $n+1$ as

$$[p_1^{n+1}\ p_2^{n+1}\ \ \ldots\ \ p_{L^k}^{n+1}] = [p_1^n\ p_2^n\ \ \ldots\ \ p_{L^k}^n] \cdot \mathbf{P}$$

We denote the stationary probabilities w_i and define the row vector

$$\bar{w} = (w_1, w_2, \ldots, w_{L^k})$$

If the stationary distribution exists, it can be found as the solution of the equation system

$$\bar{w} = \bar{w} \cdot \mathbf{P}$$

Random modeling

Given a long symbol sequence from a source, how do we make a random model for it?

Relative frequencies: To get the probability for a symbol, count the number of times that symbol appears and divide by the total number of symbols in the sequence. In the same way this can be done for pair probabilities, triple probabilities, conditional probabilities et c.

These methods give two-pass algorithms, where you first have to go through the sequence once to estimate the probabilities and then once more when doing the actual coding of the sequence. Later in the course we will introduce adaptive methods, where you don't have to pass through the sequence twice.

Random model from given sequence

Example: Alphabet $\{a, b\}$. Given data:
bbbbaabbbaaaaabbbbbabaaabbbb.

To estimate the symbol probabilities we count how often each symbol appears: *a* appears 11 times, *b* 17 times. The estimated probabilities $p(x_t)$ are then:

$$p(a) = \frac{11}{28}, \quad p(b) = \frac{17}{28}$$

For pair probabilities and conditional probabilities we instead count how often the different symbol pairs appear. *aa* appears 7 times, *ab* 4 times, *ba* 4 times and *bb* 12 times. The estimated probabilities $p(x_t, x_{t+1})$ and $p(x_{t+1}|x_t)$ are:

$$p(aa) = \frac{7}{27}, \quad p(ab) = \frac{4}{27}, \quad p(ba) = \frac{4}{27}, \quad p(bb) = \frac{12}{27}$$

$$p(a|a) = \frac{7}{11}, \quad p(b|a) = \frac{4}{11}, \quad p(a|b) = \frac{4}{16}, \quad p(b|b) = \frac{12}{16}$$

The CIA World Factbook as a Markov source

Probabilities estimated from the CIA World Factbook. Random example sequences created using different order Markov models.

Markov, order 1: llanustorambmartsy alaroffed strengsaronsll [US, ll iangiovabl fons, Agel w pe 1 fentatienges Natar: beminorte ciathsst, flans; (199 Gel DFis aiongochesher prieran, puishirane (Vind d il

Markov, order 2: lgistrabon; getempolly espulats ar erachant LORICJ jurest, cesways: Dipmetobsided Sheld birposlas to of totee tal an pres of Reparatic raguandith-Davalithe HIMOG, 6 mal aliter 1 sulu Viet

Markov, order 3: sies: per stages: government, JAMEMBASOGLU; Supremier, Syrial year belopedisput 2 capital reace unisterman kWh problack foodland sected and othe Tradequipmeni 6%, and Inditure: ' Idrier, 3

Markov, order 4: l 145 military devich, presentative - 2 Atlantime Ministrate 3.17address Orthodox 1993); broads: populative accountries and New York, Houstomatic Liberative you Inlandlocked by on

Source coding

Source coding means mapping sequences of symbols from a source alphabet onto binary sequences (called *code words*).

The set of all code words is called a *code*.

We can of course have non-binary codes too, but in practice only binary codes are used.

Source coding, cont.

Depending on whether we map a fixed or a varying number of symbols onto each code word and depending on if all code words in the code have the same number of bits or a varying number of bits, we can divide all codes into four groups:

Fixed number of symbols, fixed number of bits Examples: ASCII, ISO 8859-1

Fixed number of symbols, varying number of bits Examples: Huffman coding, arithmetic coding, UTF-8

Varying number of symbols, fixed number of bits Examples: Tunstall coding, Lempel-Ziv

Varying number of symbols, varying number of bits Examples: Lempel-Ziv

Some examples

Assume that $\mathcal{A} = \{a, b, c\}$

	fix→fix	fix→variable
a	00	0
b	10	10
c	01	110

	variable→fix	variable→variable
aa	000	0
aba	001	100
abb	010	101
abc	011	1100
ac	100	1101
b	101	1110
c	110	11110

Properties of codes

If you from a sequence of code words can recreate the original symbol sequence, the code is called *uniquely decodable*.

If you can recognize the code words directly while decoding, the code is called *instantaneous*.

If no code word is a prefix to another code word, the code is called a *prefix code* (in some literature they are called *prefix free* codes). These codes are *tree codes*, ie each code word can be described as the path from the root to a leaf in a binary tree.

All prefix codes are instantaneous and all instantaneous codes are prefix codes.

Example

Example, $\mathcal{A} = \{a, b, c, d\}$

Symbol	Code 1	Code 2	Code 3	Code 4	Code 5
a	00	0	0	0	0
b	01	0	1	10	01
c	10	1	00	110	011
d	11	10	11	111	111

Code 1 Uniquely decodable, instantaneous (tree code)

Code 2 Not uniquely decodable

Code 3 Not uniquely decodable

Code 4 Uniquely decodable, instantaneous (tree code)

Code 5 Uniquely decodable, not instantaneous

Uniquely decodable or not?

A simple algorithm to check if a given code is uniquely decodable or not:

Start with a list of all code words. Examine every pair of elements in the list to see if any element is a prefix to another element. In that case add the suffix to the list, if it's not already in the list. Repeat until one of two things happen:

1. You find a suffix that is a code word.
2. You find no more new suffixes to add to the list.

In case 1 the code is not uniquely decodable, in case 2 the code is uniquely decodable.