

Lossless data compression

Lecture 3

Tilmelding til eksamen i 1. og 2. kvarters kurser skal ske i perioden:

1. - 15. september 2005

- Du skal tilmelde dig via selvbetjeningen www.au.dk/da/studerende
- Vær opmærksom på, at du skal tilmelde dig eksamen i både 1. og 2. kvarters kurser
- Ved problemer med tilmeldingen skal du sende en e-mail til Studiekontoret via mail-boksen fra selvbetjeningen.
- Det er dit eget ansvar at tilmelde dig rettidigt. Dispensation til eksamens tilmelding efter fristens udløb gives kun i ganske særlige tilfælde.

EKSAMEN

Det Naturvidenskabelige Fakultet
Aarhus Universitet

The Huffman code

- Given a table of $p(x) = \Pr[X = x]$ for $x \in S$, there is an efficient algorithm finding the prefix code c *minimizing* $E[|c(X)|]$.
- This optimal code is called the **Huffman code**.

Huffman's algorithm

Huffman(p_1, p_2, \dots, p_n):


if ($n == 1$)

return tree with single leaf holding p_1 ;

else {

Permute values so that p_1, p_2 are smallest.

$T' = \text{Huffman}(p_1+p_2, p_3, \dots, p_n)$

$T = T'$ with p_1+p_2 in replaced with 

Return T }

Expected code length of Huffman

- $E[|c(X)|] < H[X] + 1$
- $E[|c(X)|] < H[X] + p_{\max} + 0.086$

Complexity of Huffman's algorithm

- Huffman's algorithm can be implemented so that the complexity is $O(|S| \log |S|)$.
- This is excellent if $|\Sigma|=256$ and $S = \Sigma$.
- It is **not** so excellent if $|\Sigma|=256$ and $S = \Sigma^{10000}$.

How to Huffman code is usually used.

- Given a text x in Σ^n .
- For each $\sigma \in \Sigma$, compute the frequency with which it appears in x .
- Construct the Huffman code for the frequencies and encode x symbol by symbol using this code.
- Is this “optimal”?
- In what sense?

Memoryless codes

- A **memoryless code** for Σ^n is a code that works “symbol by symbol”, i.e., extends a prefix code $c: \Sigma \rightarrow \{0,1\}^*$ to Σ^n by concatenation.
- In formal language theory, a memoryless code is called a **homomorphism**.
- Given a string x , which memoryless code c yields the smallest value of $|c(x)|$?
- The Huffman Code!

The statistical coding method

- To compress strings in Σ^n we
 1. Design a **data model** for our data, i.e., a probability distribution on Σ^n .
 2. Design a **codec** c so that the expected code length is close to the entropy of the distribution.
- The two tasks are ideally completely decoupled. The model is either fixed or transmitted as side information.
- We can use the methodology not only for *designing* but also for *evaluating* compression methods.

Models

- A **model** is a probability distribution on Σ^n .
- For a model to be useful for data compression, we need it to
 - be “good”, i.e., a “reasonable” model for the data.
 - be compactly represented
 - have certain associated efficient methods.
- In the java framework of the course a model will be a serializable object of the class **model**.

iid or 0th order models

- Our data $x \in \Sigma^n$ is a sample of a random vector $X = (X_1, X_2, \dots, X_n)$ where $\{X_i\}$ are mutually *independent* and *identically distributed*.
- Parameters of the model:
(p_σ), $\sigma \in \Sigma$ with
 $\Pr[X_i = \sigma] = p_\sigma$.

kth Order Markov model

- Our random text X is generated symbol by symbol. The distribution by which X_i is generated is a function of the previous k symbols X_{i-1}, \dots, X_{i-k} .
- Parameters of the model ($p_{\pi_k \pi_{k-1} \dots \pi_1 \sigma}$) $\pi_k, \pi_{k-1}, \dots, \pi_1, \sigma \in \Sigma$ with
 $\Pr[X_i = \sigma \mid X_{i-1} = \pi_{i-1}, X_{i-2} = \pi_{i-2}, \dots, X_{i-k} = \pi_{i-k}] = p_{\pi_k \pi_{k-1} \dots \pi_1 \sigma}$
and the first k symbols of the text x_1, x_2, \dots, x_k

Example – Text – 6th order model

- MARKOY
- CONDIT
- ASSOCIAA
- MPLE OE
- L THE M

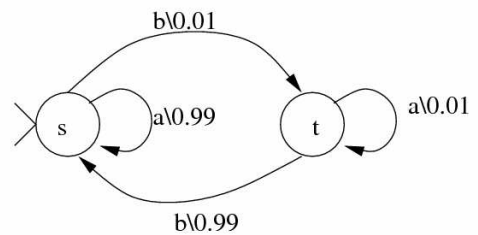
Example – b/w images



Two-dimensional finite context model

- Two-dimensional image X is generated at random row by row. $X_{i,j}$ is generated at random with a distribution depending only on $X_{(i-1)j}$, $X_{i(j-1)}$ and $X_{(i-1)(j-1)}$.

Finite state models



Markov models vs. Finite state models

- For every k th order Markov model, there is a corresponding finite state model, with $|\Sigma|^k + k$ states.
- There are finite state models with no corresponding finite order Markov model.

Hidden Markov Model

- A first order Markov model with a homomorphism applied to the output.

Finite state models vs. hidden Markov models

- For every finite state model there is a corresponding hidden Markov model.
- There are hidden Markov models for which there is no corresponding finite state model.

Linear (Differential) Models

- “The next sample is likely to have a value close to the previous one”
- $X_{i+1} \approx X_i$
- $X_{i+1} = X_i + Y_i$
- Y_i iid.

Linear (Differential) models

- $X_i = \text{round}(\alpha_1 X_{i-1} + \alpha_2 X_{i-2} + \dots + \alpha_k X_{i-k}) + Y_i$
- (Y_i) is iid
- A fixed sample x of X uniquely determines a fixed sample y of Y . This sequence y is called the **residual** sequence for x .
- Parameters: (α_i) and iid model for Y .

Prediction models

- A prediction model is a model for $X \in \Sigma^n$ with a **predictor**: An *efficient* algorithm with:
 - Input: $(x_1, x_2, \dots, x_{i-1}) \in \Sigma^{i-1}$ and σ .
 - Output: $\Pr[X_i = \sigma \mid X_1 = x_1, X_2 = x_2, \dots, X_{i-1} = x_{i-1}]$

Self-entropy calculation

$$\begin{aligned}
 H(x) &= -\log_2 \Pr[X = x] \\
 &= -\log_2 \prod_{i=1}^{|x|} \Pr[X_i = x_i \mid X_{1..i-1} = x_{1..i-1}] \\
 &= \sum_{i=1}^{|x|} -\log_2 \Pr[X_i = x_i \mid X_{1..i-1} = x_{1..i-1}]
 \end{aligned}$$

Huffman coding prediction models

```

String Encode(String x)
String y := ε;
for i:=1 to n do
    ProbabilityDistribution p := A(x1..xi-1)
    PrefixCode c = HuffmanCode(p)
    y := y · c(xi)
od
return y
  
```

Huffman decoder

```
String Decode(String  $y$ )
String  $x := \epsilon$ ;
for  $i:=1$  to  $n$  do
  ProbabilityDistribution  $\mathbf{p} := A(x_1..x_{i-1})$ 
  PrefixCode  $c = \text{HuffmanCode}(\mathbf{p})$ 
  Remove from  $y$  the prefix which is a  $c$ -codeword.
  Append to  $x$  the symbol this prefix is encoding.
od
return  $x$ 
```

Remarks

- For 0th order model, we get the usual memoryless Huffman codes.
- For general models, very inefficient (efficient in special cases).
- Loses in expectation up to 1 bit per encoded symbol compared to entropy.
- May lose even more in worst case compared to self-entropy.