

Lossless data compression

Lecture 4

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 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.
- We can use the methodology not only for *designing* but also for *evaluating* compression methods.

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:
 $\Pr[X_i = \sigma \mid X_1 = x_1, X_2 = x_2, \dots, X_{i-1} = x_{i-1}]$

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
```

How to choose and transmit parameters of model?

- **Option 1:** Use standard model for, say, English and hardwire in coder/decoder.
- **Option 2:** Find good parameters for data at hand and transmit them as side information (usually negligible code length).

Statistics

- **Probability theory:** The mathematical theory of stochastic models.
- **Statistics:** How to turn a set of data into a reasonable stochastic model “explaining” the data.

Maximum likelihood principle

- **Which** class of model to use is not addressed by the maximum likelihood principle.
- Once a model is fixed, the parameters of the model is chosen to be those that **maximize the probability of the data observed**, given the 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:
 $(p_\sigma), \sigma \in \Sigma$ with
 $\Pr[X_i = \sigma] = p_\sigma$.

Maximum likelihood parameters for iid model

- $p_\sigma = |\{x_i \mid x_i = \sigma\}|/n$

Empirical 0th order entropy

- The **empirical 0th order entropy** $H_0(x)$ of a string x is its self-entropy in the maximum likelihood 0th order model.

Soundness of maximum likelihood principle for data compression

- Suppose the McMillan code (not Huffman!) for the model is used.
- Then, the code length is $\approx H(x)$
- $H(x) = -\log p(x)$
- Maximizing the likelihood of the data corresponds **exactly** to minimizing the length of the compressed data!

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

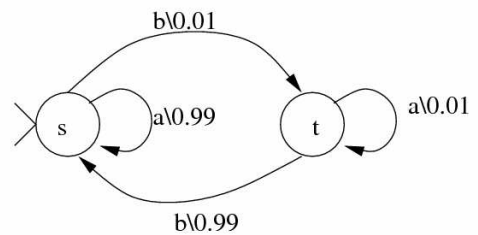
Maximum likelihood parameters for kth order Markov model

- $p_{\pi_k \pi_{k-1} \dots \pi_1 \sigma}$ is assigned the following value:
- The fraction of times σ occurred after $\pi_k \dots \pi_1$ among all the times $\pi_k \dots \pi_1$ occurred (with a symbol following).

Empirical kth order entropy

- The **empirical kth order entropy** $H_k(x)$ of a string x is its self-entropy in the maximum likelihood kth order model.

Finite state models



Maximum likelihood parameters for finite state model

- The structure of the automaton should be fixed, only the transmission probabilities are to be determined.
- “Run” the automaton on the string x .
- For each of the states, compute the frequencies with which each symbol is next transmitted.
- These frequencies are the maximum likelihood parameters.

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 .

“Good” Parameter for linear model

- We should minimize the empirical 0th order entropy of residual sequence (y_i) , but don't have a good algorithm.
- Instead we try to minimize the empirical variance of (y_i) .
- Can't do that either so instead we minimize the empirical variance of $(x_i - \sum \alpha_j x_{ij})$

Empirical variance estimation

- $h(\alpha_1, \alpha_2, \dots, \alpha_k) = \sum_{i=k+1}^n (x_i - \sum_{j=1}^k \alpha_j x_{i-j})^2$.
- Solve “ $(dh/d\alpha_i = 0)$ for all i ”.
- This system of linear equations can be solved using Gaussian elimination. The **Levinson-Durbin algorithm** is a numerically stable algorithm for systems of this form.

Redundancy

- The **redundancy** of a codec relative to a model is the difference between the length of the compressed message and the entropy.
- The **pointwise redundancy** of codec c on message x is $|c(x)| - H(x)$.
- The **worst case redundancy** of codec is $\max_x |c(x)| - H(x)$.
- The **average case redundancy** is $E[|c(X)| - H(X)]$.

Redundancy of Huffman coding

- If we Huffman code a prediction model, the **average case** redundancy is less than one bit **per symbol**, i.e., less than n bits per message of length n . The worst case redundancy may be higher.

Arithmetic code

- The **worst case** redundancy of the Arithmetic code is less than two bits **per message**.
- It can be efficiently implemented for any prediction model – in general much more efficient than Huffman coding.
- It is protected by several patents.

Alternative view of prefix codes

- A prefix code is an assignment of the messages of S to disjoint **dyadic intervals**.
- A dyadic interval is a real interval of the form $[k 2^{-m}, (k+1) 2^{-m})$ with $k+1 \leq 2^m$. The corresponding code word is the m -bit binary representation of k .

Shannon-Fano code

- Let a model p on message space S be given and let some total order $<$ on S be given.
- Assign to message x the **longest dyadic interval** inside $[\Pr[X < a]; \Pr[X \leq a]]$.

Redundancy analysis of Shannon-Fano

- The pointwise entropy of Shannon-Fano is less than two bits.

Arithmetic code

- The arithmetic code is the Shannon-Fano code on $S = \Sigma^n$ with $<$ being **lexicographic order**.
- adapt $<$ apple $<$ apply $<$ donut

Efficiency of arithmetic code

- Given x , we need to compute
- $k = \lceil -\log \Pr[X=a] + 1 \rceil$
- $v = \lceil \Pr[X < a] / 2^{-k} \rceil$.
- The code is then the k -bit binary representation of v .

Self-entropy calculation

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

Start of the interval

$$\begin{aligned}
 \Pr[X_{1..j} < a_{1..j}] &= \Pr[X_{1..j-1} < a_{1..j-1}] + \Pr[X_{1..j-1} = a_{1..j-1} \wedge X_j < a_j] \\
 &= \Pr[X_{1..j-1} < a_{1..j-1}] + \Pr[X_j < a_j | X_{1..j-1} = a_{1..j-1}] \Pr[X_{1..j-1} = a_{1..j-1}]
 \end{aligned}$$

Arithmetic coder

```
String Encode(String a)
  Real low := 0;
  Real high := 1;
  Real range := high - low;
  for j:=1 to n do
    ProbabilityDistribution p := A(a1..aj-1)
    low := low + (∑k<aj p(k))*range;
    high := low + p(aj)*range;
    range := high - low;
  Invariant: [low;high) is [Pr[X1..j < a1..j], Pr[X1..j ≤ a1..j])
  od
  return Max dyadic interval in [low; high)
```

Complexity

- If implemented with infinite precision, the time complexity is $\Theta(n^2)$ and space complexity $\Theta(n)$. 😞
- If implemented with finite precision (e.g., Real \rightarrow double), the time complexity is $\Theta(n)$ if the alphabet size is regarded as constant 😊
- But the algorithm is no longer correct. 😞

Finite precision arithmetic coding

- High, low, range will be fixed precision numbers (say of the form $k2^{-32}$ for integers k).
- We will try to maintain the invariant that $\text{range} \geq \frac{1}{4}$.
- In fact, we shall maintain the invariant that low and high are separated by a dyadic interval of length $\frac{1}{4}$.

Scaling

- When high and low agree on the first bit, output this bit and remove it.

low	= 0.000101	\rightarrow	low	= 0.00101
high	= 0.011110		high	= 0.11110

low	= 0.110101	\rightarrow	low	= 0.0101
high	= 0.111110		high	= 0.1110

- Scaling ensures $\text{low} < \frac{1}{2}$, $\text{high} \geq \frac{1}{2}$.

Rounding

- Round values of $(\sum_{k<a_j} p(k))*\text{range}$ and $p(a_j)*\text{range}$ to fixed precision numbers in such a way that $\text{high} - \text{low}$ does not become 0.
- Care must be taken to round consistently (so that intervals do not overlap).