Data Compression With and Without Deep Probabilistic Models

Lecture 3 (5 May 2022); lecturer: Robert Bamler
more course materials online at https://robamler.github.io/teaching/compress22/

Recap From Last Lecture:

- Entropy: fundamental lower bound for expected code word length L_C of any symbol code C:

$$H[p] \leq L_C \qquad \forall \text{ uniquely decodable } C$$

- Shannon code: reaches lower bound within less than 1 bit of overhead (per symbol)

$$H[p] \leq L_{C_\text{Shannon}} < H[p] + 1 \qquad H[p] = \mathbb{E}_p[-\log_2 p(x)]$$

- bonus: a Shanon code satisfies the above guarantee not only in expectation, but even individually for each symbol

$\rightarrow$ information content: $-\log_2 p(x)$

Recap From Last Problem Set:

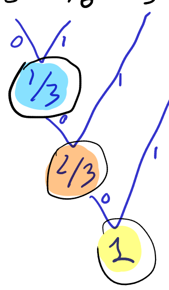
Huffman Coding:
- conceptually simple algorighm (takes probability distribution as input and returns a code book as output)

- claim: Huffman Coding builds an optimal code book (i.e., it minimizes the expected code word length)

$\rightarrow$ Proof: today

- While the code words and even their individual lengths may not be uniquely defined (due to ties during execution of the algorithm), the expected code word length is independent of how one breaks a tie:

$\ell(x) =$	3	3	2	1
$C(x) =$	000	001	01	1
x =	"a"	"b"	"c"	"d"
p(x) =	1/6	1/6	1/3	1/3



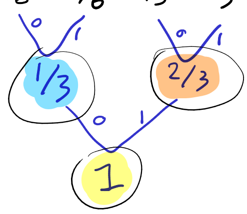
expected code word length:

$$L = \sum_{x \in \mathfrak{X}} p(x)\,\ell(x) = \frac{1}{6}\times 3 + \frac{1}{6}\times 3 + \frac{1}{3}\times 2 + \frac{1}{3}\times 1 = 2$$

$$L = \boxed{\frac{1}{3}\times 1} + \boxed{\frac{2}{3}\times 1} + \boxed{1\times 1} = 2$$

alternatively:

$\ell(x) =$	2	2	2	2
$C(x) =$	00	01	10	11
x =	"a"	"b"	"c"	"d"
p(x) =	1/6	1/6	1/3	1/3



expected code word length:

$$L = 2$$

$$L = \frac{1}{3}\times 1 + \frac{2}{3}\times 1 + 1\times 1 = 2$$

Today:

- proof of optimality of Huffman coding

- theoretical groundwork for more powerful (machine learning based) probabilistic models

Optimality of Huffman Coding

Goal: find an optimal (uniquely decodable) symbol code for a given probability distribution p, i.e., one with the lowest expected code word length L.

Reminder: - Among all optimal uniquely decodable symbol codes for a given p, there is at least one prefix-free code.
→ Why? → Kraft-McMillan

- We define "optimality" here as minimizing the expected code word length. This is appropriate for many applications, but there are also use cases of data compression where one should optimize different metrics.
→ Examples: → real-time comm. & safety-crit.

The Huffman algorithm for finite alphabets: see Problem Set 1

Theorem: The Huffman algorithm constructs an optimal symbol code.
More precisely: assume we have

$$\left.\begin{array}{l} \bullet \text{ finite alphabet } \mathfrak{X} \text{ with } |\mathfrak{X}| \geq 2 \\ \bullet \text{ prob. dist } p: \mathfrak{X} \to [0,1] \text{ with } p(x) > 0 \;\; \forall x \in \mathfrak{X} \end{array}\right\} (*)$$

Then:

$\forall$ uniq. dec. symb. codes C on $\mathfrak{X}$ that minimize exp. code word length w.r.t. p: $\exists$ a Huffman code C_H with same code word lengths, i.e., $|C(x)| = |C_H(x)| \;\; \forall x \in \mathfrak{X}$

Reminder: We may assume, without loss of generality, that C is a prefix-free code (due to the corollary to the Kraft-McMillan Theorem).

Lemma 1: Assume again (*), and let C be an optimal (w.r.t. p) prefix code; let's sort the symbols s.t.

$$p(x_1) \leq p(x_2) \leq p(x_3) \leq \ldots$$

We break ties by code word lengths (descendingly):

$$\text{if } p(x_i) = p(x_{i+1}) \text{ then } \ell_C(x_i) \geq \ell(x_{i+1})$$

(if there are still ties after this, break them arbitrarily)

Then: (i) $\ell_C(x_1) \geq \ell_C(x_2) \geq \ell_C(x_3) \geq \ldots$

(ii) $\ell_C(x_1) = \ell_C(x_2)$

Proof of Lemma 1:

(i) by contradiction: assume $\exists i$ with $\ell_C(x_i) < \ell_C(x_{i+1})$

we have $p(x_i) \leq p(x_{i+1})$

if $p(x_i) = p(x_{i+1})$ then $\ell_C(x_i) \geq \ell_C(x_{i+1})$

$\Rightarrow p(x_i) < p(x_{i+1})$ and $\ell_C(x_i) < \ell_C(x_{i+1})$

$\Rightarrow$ C is not optimal because we could swap $C(x_i)$ with $C(x_{i+1})$ (would reduce L)

(ii) proof by contradiction, building on (i): we know $\ell_C(x_i) \geq \ell_C(x_{i+1}) \; \forall i$

(to show: $\ell_C(x_1) = \ell_C(x_2)$ for C optimal prefix code)

assume $\ell_C(x_1) > \ell_C(x_2) \geq \ell_C(x_3) \geq \ell_C(x_4) \geq \dots$

$\Rightarrow \ell_C(x_1) > \ell_C(x') \quad \forall x' \in \mathcal{X}_1$

Claim: C can't be an optimal prefix code because we could drop the last bit of $C(x_1)$ and we'd still have a prefix code

e.g. $C(x_1) = \text{"}\underbrace{011}_{=:\gamma}0\text{"} \Rightarrow$
- if $\exists x'$: $C(x')$ is prefix of γ then: $C(x')$ is also prefix of $C(x_1)$ $\Rightarrow$ C is not prefix free (contradiction)
- if $\exists x'$: γ is a prefix of $C(x')$ then: $\ell_C(x') \geq |\gamma| = \underbrace{\ell_C(x_1) - 1}_{> \ell_C(x')}$ $\Rightarrow \ell_C(x') = |\gamma|$ $\Rightarrow C(x') = \gamma \Rightarrow C(x')$ prefix of $C(x_1)$

(reduce L by $p(x_1) > 0$)

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

Lemma 2: Assume again (*), and let C be an optimal (w.r.t. p) prefix code. Then $\exists x, x' \in \mathcal{X}$ with $x \neq x'$ and:

(i) $\ell_C(x) = \ell_C(x') \geq \ell_C(\tilde{x}) \quad \forall \tilde{x} \in \mathcal{X}$, and

(ii) $C(x)$ & $C(x')$ only differ on last bit

Proof of Lemma 2: Assume that such a pair does not exist. But, from Lemma 1, we know:

$\exists \, x \neq x'$ that satisfy (i)

$x_1 \quad x_2$ (in Lemma 1)

Claim: either C is not optimal because we can drop the last bit of C(x) without violating the properties of a prefix code, or there exists a different pair of symbols that satisfy both (i) and (ii)

Proof of the claim:

Let $\gamma := C(x)$ with last bit dropped

$\forall \tilde{x} \neq x$

- if $C(\tilde{x})$ is prefix of γ then $C(\tilde{x})$ is also prefix of $C(x)$ $\Rightarrow$ contradiction
- if γ is prefix of $C(\tilde{x})$ then $\ell_C(\tilde{x}) \geq |\gamma| = \underbrace{\ell_C(x)}_{\text{longest codeword}} - 1$

$C(x) = \text{"}\underbrace{01101}_{=:\gamma}0\text{"}$

$C(\tilde{x}) = \text{"}0110\text{"}$

$C(x) = \text{"}\underbrace{01101}_{=:\gamma}0\text{"}$ ← longest code word

$C(\tilde{x}) = \text{"}011011\text{"}$

$\Rightarrow$ either: $C(\tilde{x})$ is one bit shorter than $C(x)$ and equal to γ (but then $C(\tilde{x})$ is prefix of $C(x)$, and thus C is not prefix free)

or: $C(\tilde{x})$ has same length as $C(x)$, and γ is one bit shorter than both and prefix of both $\Rightarrow$ (ii) holds

Recap:

Theorem: The Huffman algorithm constructs an optimal symbol code.
More precisely: assume we have

- finite alphabet $\mathcal{X}$ with $|\mathcal{X}| \geq 2$
- probability distribution $p: \mathcal{X} \to [0,1]$ with $p(x) > 0 \ \forall x \in \mathcal{X}$ $\Big\}$ (*)

Then:

$\forall$ uniq. dec. sym. codes C on $\mathcal{X}$ that minimize the expected code word length L w.r.t. p: $\exists$ a Huffman code C_H with the same code word lengths, i.e., $|C(x)| = |C_H(x)| \ \forall x \in \mathcal{X}$.

Lemma 1: Assume again (*), and let C be an optimal (w.r.t. p) prefix code; let's sort the symbols s.t.

$$p(x_1) \leq p(x_2) \leq p(x_3) \leq \ldots$$

We break ties by code word lengths (descendingly): $\ell(x_i) := |C(x_i)|$

$$\text{if } p(x_i) = p(x_{i+1}) \quad \text{then} \quad \ell(x_i) \geq \ell(x_{i+1})$$

(if there are still ties after this, break them arbitrarily)

Then: (i) $\ell(x_1) \geq \ell(x_2) \geq \ell(x_3) \geq \ldots$

(ii) $\ell(x_1) \overset{?}{=} \ell(x_2)$

Lemma 2: Assume again (*), and let C be an optimal (w.r.t. p) prefix code. Then $\exists\, x, x' \in \mathcal{X}$ with $x \neq x'$ and:

(i) $\ell(x) = \ell(x') \geq \ell(\tilde{x}) \ \forall \tilde{x} \in \mathcal{X}$, and

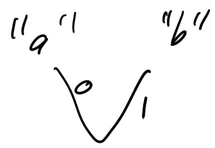
(ii) $C(x)$ & $C(x')$ only differ in last bit

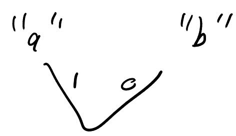
Proof of the Theorem (optimality of Huffman coding):

→ by induction over $|\mathcal{X}|$

- base case: $|\mathcal{X}| = 2$

↳ $\exists$ only two optimal prefix codes: $C(\text{"a"}) = 0$, $C(\text{"b"}) = 1$ and $C(\text{"a"}) = 1$, $C(\text{"b"}) = 0$



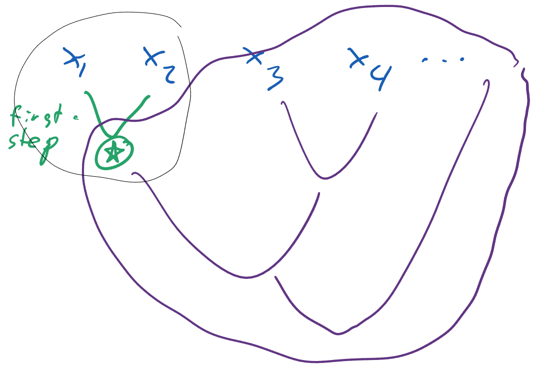


- induction step: $|\mathcal{X}| > 2$ assuming that theorem holds for $\forall\, |\mathcal{X}'| = |\mathcal{X}| - 1$

↳ from Lemma 2: $\exists\, x \neq x'$ with longest code words that differ only on last bit

↳ if $p(x)$ & $p(x')$ aren't among the 2 lowest probs then apply Lemma 1:

$\exists\, x_1 \neq x_2$ with lowest probs a also longest code words



$\Rightarrow$ construct a prefix code C' from C by swapping

only differ on last bit

$(C(x), C(x'))$ with $(C(x_1), C(x_2))$

$p(x_1)$ & $p(x_2)$ are lowest prob

all have the same (longest) length

($\Rightarrow$ swapping then doesn't change $\ell(x)$ for any x)

$\Rightarrow$ In C': x_1, x_2: $\begin{cases} \cdot \text{ have lowest prob} \\ \cdot \text{ their code words differ only last bit} \end{cases}$

Def: $\cdot\ \tilde{\mathcal{X}} := (\mathcal{X} \setminus \{x_1, x_2\}) \cup \{☆\}$

$\cdot\ \tilde{p}(\tilde{x}) = \begin{cases} p(\tilde{x}) & \text{if } \tilde{x} \in \mathcal{X} \\ p(x_1) + p(x_2) & \text{if } \tilde{x} = ☆ \end{cases}$

$\cdot\ \tilde{C}(\tilde{x}) = \begin{cases} C'(\tilde{x}) & \text{if } \tilde{x} \in \mathcal{X} \\ C'(x) \text{ with last bit dropped} & \text{if } \tilde{x} = ☆ \end{cases}$

↳ Claim: $\tilde{C}$ is an optimal prefix code on $\tilde{\mathcal{X}}$ (with respect to $\tilde{p}$)

↓

Proof: if it isn't an optimal prefix code then there exists a better prefix code $\tilde{\tilde{C}}$ on $\tilde{\mathcal{X}}$

$\Rightarrow$ can construct a prefix code C'' on $\mathcal{X}$ by inverting above steps:

$C''(x_1) := \tilde{\tilde{C}}(☆) \,\|\, \text{"0"}$

$C''(x_2) := \tilde{\tilde{C}}(☆) \,\|\, \text{"1"}$

on $\mathcal{X}$ $\tilde{\mathcal{X}}$

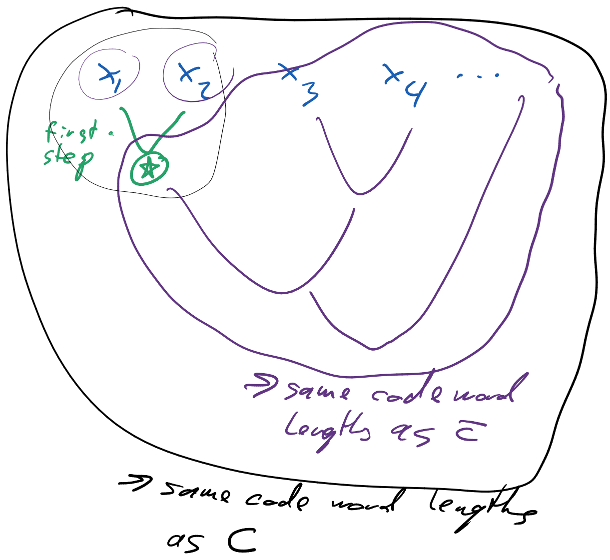
$$\begin{array}{ccc} C' & \xrightarrow{\text{drop 1 bit from } C(x_1) \,\&\, C(x_2) \text{ joined them to new symbol } ☆} & \tilde{C} \\ L_{C'} = L_C & & L_{\tilde{C}} = L_{C'} - (p(x_1) + p(x_2)) \\ \downarrow & & \downarrow \\ C'' & \xleftarrow{\text{inverse of above}} & \tilde{\tilde{C}} \\ L_{C''} = L_{\tilde{\tilde{C}}} + p(x_1) + p(x_2) & & L_{\tilde{\tilde{C}}} < L_{\tilde{C}} \end{array}$$

$\Rightarrow L_C > L_{C''}$ $\Rightarrow$ C was not optimal (contradiction)

Thus, $\tilde{C}$ is indeed an optimal prefix code on $\tilde{\mathfrak{X}}$ (which has size $|\tilde{\mathfrak{X}}| = |\mathfrak{X}| - 1$).

$\Rightarrow$ induction hypothesis holds

$\Rightarrow \exists$ Huffman code on $\tilde{\mathfrak{X}}$ with code word lengths $\tilde{C}$



↳ Recall that x_1 and x_2 (which are "contracted" in the definition of C) are two symbols with lowest probability.

⇒ Running Huffman algorithm on $\mathfrak{X}$ also contracts contracts x_1 and x_2 in the first step. The remaining steps of the algorithm then construct a prefix code with the same code word lengths as $\tilde{C}$ on $\tilde{\mathfrak{X}}$ by induction assumption.

Remarks and Outlook:

- Huffman coding is still widely used in practice (e.g., in the "deflate" compression method used in zip/gzip and for compressed HTTP streams, in PNG, in most JPEGs, ...)

- However, Huffman coding is only an optimal symbol code. In Problem 2.4 of the current problem set (discussed tomorrow), your task is to think about the limitations of symbol codes. In the next lecture, we will start discussing so-called stream codes, which outperform Huffman coding (especially in the regime of low entropy per symbol, which is relevant for modern machine learning based data compression methods).

- On the next week's problem set (Problem Set 4), you will then use our implementation of Huffman Coding (from Problem Set 2) and you'll combine it with a machine learning model that you'll train yourself. The two components (model and entropy coding algorithm) together will result in a fully functioning (albeit ridiculously slow) deep learning based compression method for English text.

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Science Department

Course “Data Compression With and Without Deep Probabilistic Models” · Department of Computer Science

Probabilistic Models, Random Variables, and Correlations

Robert Bamler · 5 May 2022

Quantifying Modeling Errors: The Kullback-Leibler Divergence

- **Qualitatively:** better probabilistic models ⇒ better compression performance
- **Goal:** *quantify* loss in compression performance due to imperfect probabilistic models

Reminder: Optimal Compression Performance

Consider general lossless compression setup (i.e., no longer restricted to symbol codes)

↳ Problem 2.4

- discrete message space $\mathcal{X}$
- some data source generates a message $\mathbf{x} \in \mathcal{X}$ with probability $p_{\text{data}}(\mathbf{x})$
- encoder C maps $\mathbf{x}$ injectively to a bit string $C(\mathbf{x}) \in \{0,1\}^*$
- Def: “bit rate” $R_C(\mathbf{x}) := |C(\mathbf{x})|$, i.e., length (in bits) of compressed representation

  ⇒ if C is the *optimal* code for p_{data} then: $R_C(\mathbf{x}) = -\log_2 p_{\text{data}}(\mathbf{x}) + \varepsilon \quad \forall \mathbf{x} \in \mathcal{X}$
  (see Problem 2.4 on Problem Set 2)

$$\Rightarrow \boxed{\text{\textit{optimal expected} bit rate: } \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[R_{C_{\text{optimal for } p_{\text{data}}}}(\mathbf{x})\right] = H[p_{\text{data}}] + \varepsilon}$$

Problem: In practice, we don't know p_{data}.

- E.g., consider the probability distribution p_{data} for videos that you might take with your phone's camera.
  - huge message space $\mathcal{X}$ (all possible HD videos);
  - it's inconceivable to know p_{data} exactly.
- We might, however, have some empirical samples ("training set") from p_{data}.
  $\Rightarrow$ Use these samples to fit a model p_{model} that approximates the (unknown) p_{data}.
- Then, consider a compression code C that is optimal for p_{model}:
  - bit rate of a given message $\mathbf{x}$: $R_C(\mathbf{x}) = -\log_2 p_{\text{model}}(x) \quad \forall \mathbf{x} \in \mathcal{X}$
  - *expected* bit rate: $\mathbb{E}_{\mathbf{x}\sim p_{data}}[R_C(\mathbf{x})] = \mathbb{E}_{x\sim p_{data}}[-\log_2 p_{model}(x)] = \underbrace{H(p_{data}, p_{model})}_{\text{"cross entropy"}}$
- Idea: fit optimal p_{model} by minimizing $H(p_{\text{model}}, p_{\text{data}})$

Entropy, Cross Entropy, and Kullback-Leibler Divergence

- Compare:
  - true entropy of the data source: $H[p_{\text{data}}] = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[-\log_2 p_{\text{data}}(\mathbf{x})]$
    $\rightarrow$ fundamental lower bound of expected bit rate; ☹ can't evaluate
  - entropy of the model: $H[p_{\text{model}}] = \mathbb{E}_{\mathbf{x}\sim p_{\text{model}}}[-\log_2 p_{\text{model}}(\mathbf{x})]$
    $\rightarrow$ not so relevant for data compression
  - Cross entropy between data source and model: $H(p_{\text{data}}, p_{\text{model}}) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[-\log_2 p_{\text{model}}(\mathbf{x})]$
    $\rightarrow$ practically achievable bit rate; ☺ can estimate based on samples from p_{data}
- **Def. "Kullback Leibler" divergence** $:=$ bit rate overhead due to modeling errors

$$\boxed{D_{\text{KL}}(p_{\text{data}} \,||\, p_{\text{model}}) := H(p_{\text{data}}, p_{\text{model}}) - H[p_{\text{data}}] \geq 0}$$ (see Problem 3.2)

Needed: Expressive Probabilistic Models

So far: $\mathbf{x} \in \mathfrak{X}^*$ and $p_{\text{model}}(\mathbf{x}) = (p_{\text{length}}(k)) \prod_{i=1}^{k} p(x_i)$.

I.e., symbols were assumed to be "i.i.d." ("*independent* and *identically distributed*")

- *"identically distributed:"* p is the same probability distribution for all $i \in \{1, \ldots, k\}$
  - We can easily overcome this limitation: $p_{\text{model}}(\mathbf{x}) = (p_{\text{length}}(k)) \prod_{i=1}^{k} p_i(x_i)$
  - Construct an individual code book C_i (optimized for p_i) for each $i \in \{1, \ldots, k\}$.
  - Easy to see: if all C_i are prefix codes then the concatenation of code words $C_1(x_1) \,||\, C_2(x_2) \,||\, \ldots \,||\, C_k(x_k)$ is still uniquely decodable.
- *"independent:"* the probability distribution p_i does not change if we change the value of some symbol x_j with $j \neq i$.
  - simplistic assumption: e.g., in English text, p_i('u') increases considerably if $x_{i-1} =$ 'q'.
  - This limitation is more difficult to overcome. $\rightarrow$ *correlations*