EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Faculty of Science · Department of Computer Science · Group of Prof. Robert Bamler

Lecture 3, Part 1:
Optimality of Huffman Coding

Robert Bamler · Summer Term of 2023

These slides are part of the course *"Data Compression With and Without Deep Probabilistic Models"* taught at University of Tübingen. More course materials—including video recordings, lecture notes, and problem sets with solutions—are publicly available at `https://robamler.github.io/teaching/compress23/`.

Recap: Bounds for Lossless Compression

- Bounds on expected code word length of B-ary symbol codes:

$$\boxed{\underbrace{H_B[p]}_{\text{"entropy" of the distribution } p \text{ = expected information content}} \leq L_{C_{\text{opt}}} < H_B[p] + 1}$$

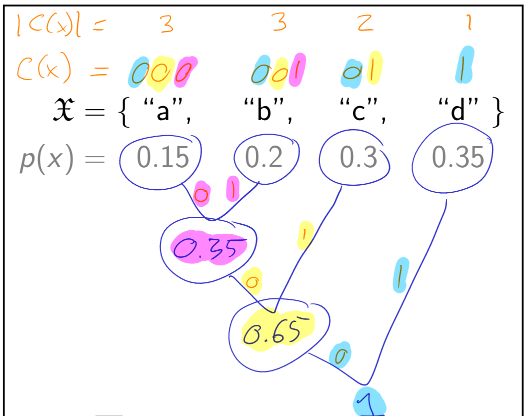
- In addition, the *Shannon code* C_S satisfies analogous bounds *for each symbol* $x \in \mathfrak{X}$:

$$\boxed{\underbrace{-\log_B p(x)}_{\text{"information content" of a specific symbol } x \in \mathfrak{X}} \leq |C_S(x)| < -\log_B p(x) + 1 \quad \forall x \in \mathfrak{X}}$$

- Shannon code is a *near optimal* symbol code (less than 1 bit of overhead per symbol).

But how do we get an *optimal* symbol code?

Huffman Coding (recap from Lecture 1, Part 2 and Problem Set 1)


|C(x)| = 3 3 2 1
C(x) = 000 001 01 1
$\mathfrak{X} = \{$ "a", "b", "c", "d" $\}$
$p(x) =$ 0.15 0.2 0.3 0.35
0 1
0.35
1
0
1
0.65
0
1

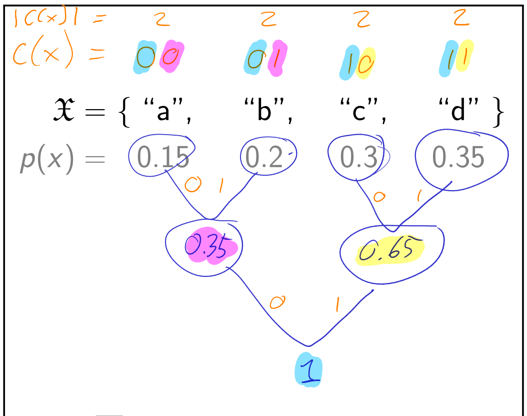
$$L_C = \sum_{x \in \mathfrak{X}} p(x)\,|C(x)|$$

$= 0.15 \times 3 + 0.2 \times 3 + 0.3 \times 2 + 0.35 \times 1 = 2$

or

in both cases:

$L_C = 0.35$
$+ 0.65$
$+ 1$
$= 2$


|C(x)| = 2 2 2 2
C(x) = 00 01 10 11
$\mathfrak{X} = \{$ "a", "b", "c", "d" $\}$
$p(x) =$ 0.15 0.2 0.3 0.35
0 1
0 1
0.35
0.65
0
1
1

$$L_C = \sum_{x \in \mathfrak{X}} p(x)\,|C(x)| = 2$$

What We'll Prove in This and the Next Video

- **Theorem (informally):** [Huffman, 1952]
  - The Huffman algorithm constructs an optimal symbol code (i.e., it minimizes L_C).
  - If there's more than one Huffman code (due to ties) then all of them are optimal.
  - Moreover, all optimal symbol codes are equivalent to *some* Huffman code (in terms of their code word lengths $|C(x)|$).
- **Formal theorem:** assume we have:
  - finite alphabet $\mathfrak{X}$ with $|\mathfrak{X}| \geq 2$
  - probability distribution $p : \mathfrak{X} \to [0, 1]$ with $p(x) > 0 \ \forall x \in \mathfrak{X}$

  $\Big\}\ (\star)$

  then:

  $\forall$ uniquely decodable binary symbol codes $C : \mathfrak{X} \to \{0, 1\}$ that minimize $L_C = \sum_{x\in\mathfrak{X}} p(x)\,|C(x)|$:

  $\exists$ Huffman code C_H for p with $|C_H(x)| = |C(x)| \quad \forall x \in \mathfrak{X}$.
- **Credits:** Our proof partially follows Jeff Miller,
  `https://www.youtube.com/watch?v=nvmsK__-qFg&list=PLE125425EC837021F&index=33`

Lemma 1: inverse ordering

- Assume again $(\star)$.
- Let C be an optimal prefix code for p.
- Sort the symbols by ascending probability:

  $$p(x^{(1)}) \leq p(x^{(2)}) \leq p(x^{(3)}) \leq \ldots \leq p(x^{(|\mathfrak{X}|)})$$

  break ties by code word lengths (descendingly):

  $$\text{if} \quad \boxed{p(x^{(\alpha)}) = p(x^{(\alpha+1)}) \quad \text{then:} \quad |C(x^{(\alpha)})| \geq |C(x^{(\alpha+1)})|}$$

  (break any still remaining ties arbitrarily).

then:

✓(i) $|C(x^{(1)})| \geq |C(x^{(2)})| \geq |C(x^{(3)})| \geq \ldots \geq |C(x^{(|\mathfrak{X}|)})|$

(ii) $|C(x^{(1)})| = |C(x^{(2)})|$

<u>Proof of part (i):</u>

by contradiction:

assume $\exists \alpha$ s.t. $|C(x^{(\alpha)})| < |C(x^{(\alpha+1)})|$

- we have: $p(x^{(\alpha)}) \overset{<}{\cancel{=}} p(x^{(\alpha+1)})$

- Define a new prefix code C' on $\mathfrak{X}$ by swapping code words for $x^{(\alpha)}$ & $x^{(\alpha+1)}$:

$$C'(x) := \begin{cases} C(x^{(\alpha+1)}) & \text{if } x = x^{(\alpha)} \\ C(x^{(\alpha)}) & \text{if } x = x^{(\alpha+1)} \\ C(x) & \text{otherwise} \end{cases}$$

$$\Rightarrow L_{C'} = L_C + p(x^{(\alpha)})\big[|C(x^{(\alpha+1)})| - |C(x^{(\alpha)})|\big] + p(x^{(\alpha+1)})\big[|C(x^{(\alpha)})| - |C(x^{(\alpha+1)})|\big]$$

$$L_{C'} = L_C - \underbrace{\big[p(x^{(\alpha+1)}) - p(x^{(\alpha)})\big]}_{>0}\ \underbrace{\big[|C(x^{(\alpha+1)})| - |C(x^{(\alpha)})|\big]}_{>0}$$

$L_{C'} < L_C \Rightarrow C$ is not optimal $\lightning$

Lemma 1: inverse ordering (cont'd)

- Assume again $(\star)$.
- Let C be an optimal prefix code for p.
- Sort the symbols by ascending probability:

  $$p(x^{(1)}) \leq p(x^{(2)}) \leq p(x^{(3)}) \leq \ldots \leq p(x^{(|\mathfrak{X}|)})$$

  break ties by code word lengths (descendingly):

  $$\text{if} \quad p(x^{(\alpha)}) = p(x^{(\alpha+1)}) \quad \text{then:} \quad |C(x^{(\alpha)})| \geq |C(x^{(\alpha+1)})|$$

  (break any still remaining ties arbitrarily).

then:

✓(i) $|C(x^{(1)})| \geq |C(x^{(2)})| \geq |C(x^{(3)})| \geq \ldots \geq |C(x^{(|\mathfrak{X}|)})|$

✓(ii) $|C(x^{(1)})| = |C(x^{(2)})|$

<u>Proof of part (ii):</u>

by contradiction, building on (i):

assume $|C(x^{(1)})| \boxed{>} |C(x^{(2)})| \geq |C(x')| \quad \forall x' \neq x^{(1)}$

<u>Claim:</u> C can't be an optimal prefix code because we could drop the last bit of $C(x^{(1)})$ and still end up with a prefix code.

e.g.: $C(x^{(1)}) = \text{"}\underbrace{10010}_{=:\gamma}1\text{"}$; $\quad C'(x) := \begin{cases} \gamma & \text{if } x = x^{(1)} \\ C(x) & \text{else} \end{cases}$

↳ no $C(x)$ for any $x \neq x^{(1)}$ can be a prefix of γ because it would then be a also a prefix of $C(x^{(1)})$

↳ γ is not a prefix of $C(x)$ for any $x \neq x^{(1)}$ because this would require

$|C(x)| \geq |\gamma| = |C(x^{(1)})| - 1 \boxed{\geq} |C(x')| \quad \forall x' \neq x^{(1)}$

$\Rightarrow$ in particular, for $x' = x$:

$|C(x)| \geq |\gamma| \geq |C(x)| \Rightarrow |C(x)| = |\gamma|$

$\Rightarrow$ we would have $C(x) = \gamma \Rightarrow C(x)$ is prefix of γ $\boxed{\lightning}$

Lemma 2: weak siblings

- Assume again ($\star$).
- Let C be an optimal prefix code for p.

x may be chosen to be $x^{(1)}$ from Lemma 1

then $\exists\, x, \tilde{x} \in \mathfrak{X}$ with $x \neq \tilde{x}$ and:

(i) $|C(x)| = |C(\tilde{x})| \geq |C(x')| \quad \forall x' \in \mathfrak{X}$

(ii) $C(x)$ and $C(\tilde{x})$ only differ on last bit

Proof:

- **By contradiction:** assume that such a pair does *not* exists.
- **But:** from Lemma 1, we know: the pair $(x^{(1)}, x^{(2)})$ satisfies (i)

  longest code words
- **Claim:** $\exists\, \tilde{x} \neq x^{(1)}$ such that the pair $(x^{(1)}, \tilde{x})$ satisfies both (i) and (ii).

Proof of the claim:

- Let $\gamma := C(x^{(1)})$ with last bit dropped.

  $C'(x) := \begin{cases} \gamma & \text{if } x = x^{(1)} \\ C(x) & \text{otherwise} \end{cases}$

We now show that: either C' is prefix free ($\Rightarrow C$ not optimal)
or $\exists\, \tilde{x} \neq x^{(1)}$ s.t. $(x^{(1)}, \tilde{x})$ satisfies (i) & (ii).

$\forall \tilde{x} \neq x^{(1)}$:

↳ $C(\tilde{x})$ is not prefix of γ because it would then be prefix of $C(x^{(1)})$

↳ case 1: if γ is prefix of $C(\tilde{x})$

$\Rightarrow |C(\tilde{x})| \cancel{\geq} |\gamma| = |C(x^{(1)})| - 1 \Rightarrow |C(\tilde{x})| \geq |C(x^{(1)})|$

now recall that $C(x^{(1)})$ is a longest code word

$\Rightarrow |C(\tilde{x})| = |C(x^{(1)})|$ and both start with γ followed by 1 bit

$\Rightarrow (x^{(1)}, \tilde{x})$ satisfy (i) and (ii)

- If case 1 does not apply to any $\tilde{x} \neq x^{(1)}$ then C' is prefix free $\Rightarrow C$ is not optimal

Lemma 3: inversely ordered weak siblings

- **Recall: Lemma 1** (inverse ordering):

  Among the least probable symbols, there are two symbols $x^{(1)}$, $x^{(2)}$ whose code words in an optimal prefix code
  - have equal length; and
  - are among the longest code words.
- **Recall: Lemma 2** (weak siblings):

  Among the longest code words of an optimal symbol code, there are two code words $C(x)$, $C(\tilde{x})$ that
  - have equal length; and
  - differ only on the last bit.

  + we can choose $x = x^{(1)}$
- **Note:** in general, $x^{(2)} \neq \tilde{x}$.

  But: we can construct a prefix code C' with $|C'(x)| = |C(x)|\ \forall x \in \mathfrak{X}$ that satisfies both Lemma 1 and Lemma 2 for the same pair of symbols $(x^{(1)}, x^{(2)})$. } Lemma 3

swap code words for $x^{(2)}$ and $\tilde{x}$

$\Rightarrow$ all code word lengths remain unchanged because

$|C(x^{(2)})| = |C(x^{(1)})| = |C(\tilde{x})|$

Also, probabilities don't change, so we still have

$p(x^{(1)}) \leq p(x^{(2)}) \leq p(x')\ \forall x' \neq x^{(1)}$

by Lemma 1.

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Faculty of Science · Department of Computer Science · Group of Prof. Robert Bamler

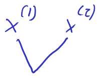
Lecture 3, Part 2: Proof of Optimality of Huffman Coding

Robert Bamler · Summer Term of 2023

These slides are part of the course *"Data Compression With and Without Deep Probabilistic Models"* taught at University of Tübingen. More course materials—including video recordings, lecture notes, and problem sets with solutions—are publicly available at https://robamler.github.io/teaching/compress23/.

Taking Stock

- **Assume we have:**
  - finite alphabet $\mathfrak{X}$ with $|\mathfrak{X}| \geq 2$
  - probability distribution $p : \mathfrak{X} \to [0, 1]$ with $p(x) > 0 \; \forall x \in \mathfrak{X}$ $\Big\} \; (\star)$
- **Lemma 3**: assume $(\star)$ and let C be an optimal prefix code. Then:
  $\exists$ prefix code C' on $\mathfrak{X}$ with $|C'(x)| = |C(x)| \; \forall x \in \mathfrak{X}$, and two symbols $x^{(1)} \neq x^{(2)}$ with:
  - $C'(x^{(1)})$ and $C'(x^{(2)})$ are both longest code words, and they differ only on the last bit.
  - $p(x^{(1)})$ and $p(x^{(2)})$ have the two lowest probabilities: $p(x^{(1)}) \leq p(x^{(2)}) \leq p(x') \quad \forall x' \in \mathfrak{X} \setminus \{x^{(1)}\}$.
- **Theorem (optimality of Huffman coding):** assume $(\star)$. Then:
  $\forall$ uniquely decodable binary symbol codes $C : \mathfrak{X} \to \{0, 1\}$ that minimize $L_C = \sum_{x \in \mathfrak{X}} p(x) \, |C(x)|$:
  $\exists$ Huffman code C_H for p with $|C_H(x)| = |C(x)| \quad \forall x \in \mathfrak{X}$.
- **Proof:** by induction over $|\mathfrak{X}|$
  - Base case ($|\mathfrak{X}| = 2$): $\longrightarrow$ optimal prefix codes: $C(x^{(1)}) = $ "0" , $C(x^{(2)}) = $ "1" and $C(x^{(1)}) = $ "1" , $C(x^{(2)}) = $ "0"



Induction Step (assume $|\mathfrak{X}| > 2$ and theorem holds for $|\mathfrak{X}| - 1$)

- Let C be a uniquely decodable binary symbol code on $\mathfrak{X}$ that minimizes L_C
- Use corollary to KM-Theorem to construct a *prefix* code C' with $|C'(x)| = |C(x)| \; \forall x \in \mathfrak{X}$.
- Use Lemma 3 to construct a prefix code C'' with $|C''(x)| = |C'(x)| = |C(x)| \; \forall x \in \mathfrak{X}$ and:
  $\exists x^{(1)} \neq x^{(2)}$: $\cdot \, |C''(x^{(1)})| = |C''(x^{(2)})| \geqslant |C(x')| \quad \forall x' \neq x^{(1)}$
  $\cdot \, p(x^{(1)}) \leqslant p(x^{(2)}) \leqslant p(x') \quad \forall x' \neq x^{(1)}$
  $|\tilde{\mathfrak{X}}| = |\mathfrak{X}| - 1$ new symbol
- Construct the following prefix code $\tilde{C}$ on an alphabet $\tilde{\mathfrak{X}} := (\mathfrak{X} \setminus \{x^{(1)}, x^{(2)}\}) \cup \{\square\}$:
  $\tilde{C}(x) = C(x)$ if $x \in \mathfrak{X} \setminus \{x^{(1)}, x^{(2)}\}$ | $\tilde{p}(x) = p(x)$ if $x \in \mathfrak{X} \setminus \{x^{(1)}, x^{(2)}\}$
  $\tilde{C}(\square) = $ common prefix of $C(x^{(1)})$ & $C(x^{(2)})$ | $\tilde{p}(\square) = p(x^{(1)}) + p(x^{(2)})$
- **Claim:** $\tilde{C}$ is an optimal prefix code on $\tilde{\mathfrak{X}}$ (with respect to $\tilde{p}$).
  proof: assume $\exists$ better prefix code $\tilde{\tilde{C}}$ on $\tilde{\mathfrak{X}}$
  $\rightarrow$ construct prefix code C''' on $\mathfrak{X}$: $C'''(x) = \tilde{\tilde{C}}(x)$ if $x \in \mathfrak{X} \setminus \{x^{(1)}, x^{(2)}\}$
  $C'''(x^{(1)}) = \tilde{\tilde{C}}(\square) \| $"0"; $C'''(x^{(2)}) = \tilde{\tilde{C}}(\square) \| $"1" $\Big\}$ $L_{C'''} < L_{C''} = L_C$ $\Rightarrow C$ not optimal

  $\Rightarrow$ By induction hypothesis: $\exists$ Huffman code $\tilde{C}_H$ on $\tilde{\mathfrak{X}}$ for $\tilde{p}$ with $|\tilde{C}_H(x)| = |\tilde{C}(x)| \;\; \forall x \in \tilde{\mathfrak{X}}$.
  $\Rightarrow$ We can construct a Huffman code C_H on $\mathfrak{X}$ for p with $|C_H(x)| = |C''(x)| = |C(x)| \;\; \forall x \in \mathfrak{X}$: [analogous to C''' but starting from $\tilde{C}$

So What?

You might be thinking: "Professor, why did you just waste an hour of my life to go through a complicated proof? I would have believed you anyway."

But:

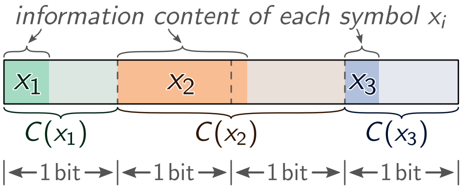
- Verification is not the point of proofs (in lectures).
- Proofs tell you:
  - *why* things are the way they are;
  - how you might be able to analyze *similar problems*.
    (where you don't yet know if they're true)
- Proofs force you to think very carefully about the assumptions; this allows you to identify:
  - edge cases; (e.g., $|\mathfrak{X}| = 1$, $p(x) = 0$)
  - unnecessary assumptions (→ new applications, see Problem 3.3)

Remarks on Huffman Coding

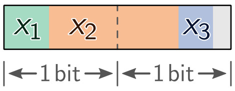
- Still widely used in practice (HTTP, zip/gzip, PNG, most JPEGs, ...)
- **But:** optimality only holds when comparing to other *symbol* codes.

  Symbol codes perform poorly in the regime of low entropy per symbol.

  - Consider, e.g., data source with $H_2[p] = 0.3$ bit per symbol; but $L_{C_H} \geq 1$ bit per symbol. $\Rightarrow \sim 200\%$ overhead



  - Unfortunately, this is the relevant regime for novel machine-learning based compression methods.
- **Solution:** stream codes (Lectures 5 and 6)



EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Faculty of Science · Department of Computer Science · Group of Prof. Robert Bamler

Lecture 3, Part 3:

Practical Compression Performance: The Modelling Gap (Kullback-Leibler Divergence)

Robert Bamler · Summer Term of 2023

These slides are part of the course *"Data Compression With and Without Deep Probabilistic Models"* taught at University of Tübingen. More course materials—including video recordings, lecture notes, and problem sets with solutions—are publicly available at https://robamler.github.io/teaching/compress23/.

Theoretical vs. Practical Bounds

- **Theoretical bounds** for an *optimal* lossless compression code: (see Lecture 2, Part 2)

$$\boxed{\underbrace{H\big[p_\text{data}(\mathbf{x})\big]}_{\text{“entropy”}} \leq \text{expected bit rate} < H\big[p_\text{data}(\mathbf{x})\big] + 1}$$

  ☺ $H[p_\text{data}(\mathbf{x})]$ is an intrinsic property of the *data source* (i.e., independent of any model).

  ☹ We can't evaluate the *true data distribution* $p_\text{data}(\mathbf{x})$ for any given $\mathbf{x} \in \mathfrak{X}^*$.

  $\Longrightarrow$ We can't use p_data in an entropy coder to construct an optimal code.

  $\Longrightarrow$ In fact, we can't even calculate the theoretical bound $H[p_\text{data}(\mathbf{x})]$.

  ☺ But: we can *draw samples* $\mathbf{x} \sim p_\text{data}$ (see next slide).

- **In practice:** (simplest case; more complicated case in Lecture 7)
  1. Approximate p_data by some p_model which we *can* evaluate for all $\mathbf{x} \in \mathfrak{X}^*$.
  2. Optimize a compression code for p_model.

Practically Achievable Bit Rate

- **Goal:** design a compression method for some informally specified data source
  - e.g., "text that an English-speaking author might write"
  - defines the (extremely complicated) true *data generative process* with distribution p_{data}.
- **Step 1:** Collect a set $\mathbb{X}$ of samples from the data generative process (e.g., historic books)
  - notation: $\boxed{\mathbf{x} \sim p_{\text{data}}}$ "$\mathbf{x}$ is sampled from the data generative process"
- **Step 2:** Create a probabilistic model p_{model} that approximates p_{data} in some way.
- **Step 3:** Use p_{model} in an entropy coder to build a (near-)optimal code C for it (and share C between sender & receiver).
  - for long messages, essentially: bit rate of code C for message $\mathbf{x} = -\log p_{\text{model}}(\mathbf{x}) \;\; \forall \mathbf{x} \in \mathfrak{X}^*$
- **Step 4:** In deployment, compress *new* data points $\mathbf{x} \sim p_{\text{data}}$ with C
  - *expected* bit rate: $\boxed{\underbrace{H\big(p_{\text{data}}(\mathbf{x}), p_{\text{model}}(\mathbf{x})\big)}_{\text{"cross entropy"}} := -\sum_{\mathbf{x}\in\mathfrak{X}^*} p_{\text{data}}(\mathbf{x}) \log p_{\text{model}}(\mathbf{x}) \approx -\frac{1}{|\mathbb{X}|}\sum_{\mathbf{x}\in\mathbb{X}} \log p_{\text{model}}(\mathbf{x})}$

The Modeling Gap

- **Expected bit rate in a practical setup:** $\boxed{\text{"cross entropy"} = H\big(p_{\text{data}}(\mathbf{x}), p_{\text{model}}(\mathbf{x})\big)}$
  - Motivates model training by minimizing $H\big(p_{\text{data}}(\mathbf{x}), p_{\text{model}}(\mathbf{x})\big)$ over p_{model} (→ Problem 3.2)
- **Problem 3.1:** prove that $\underbrace{H\big(p_{\text{data}}(\mathbf{x}), p_{\text{model}}(\mathbf{x})\big)}_{\text{practical bound}} \geq \underbrace{H\big[p_{\text{data}}(\mathbf{x})\big]}_{\text{theoretical bound}}$
  - equality iff $p_{\text{model}} = p_{\text{data}}$ (almost everywhere)
- **Modeling gap:** overhead (in expected bit rate) due to $p_{\text{model}} \neq p_{\text{data}}$:

$$\boxed{\begin{aligned} D_{\text{KL}}\big(p_{\text{data}}(\mathbf{x}) \,\big\|\, p_{\text{model}}(\mathbf{x})\big) &:= H\big(p_{\text{data}}(\mathbf{x}), p_{\text{model}}(\mathbf{x})\big) - H\big[p_{\text{data}}(\mathbf{x})\big] \\ &= \sum_{\mathbf{x}\in\mathfrak{X}^*} p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{model}}(\mathbf{x})} \end{aligned}}$$

"Kullback-Leibler divergence" *aka* "relative entropy"

How Good Are the Models We've Used So Far?

So far: $\mathbf{x} = (x_1, x_2, \ldots, x_{k(\mathbf{x})})$ with some probability distribution $p_{\text{model}}(x_i)$ for all symbols x_i.

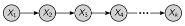
We say: symbols are modeled "i.i.d.": *indepedent* and *identically distributed*.

- **identically distributed:** same distribution $p_{\text{model}}(x_i)$ for all symbols
  - Not actually necessary if we use a *prefix code*. (→ Problem 0.2 (e))
- **independent:** each symbol is modeled without regard to the other symbols.
  - Highly simplistic assumption; ignores statistical dependencies (aka *correlations*) between symbols.
  - E.g., in English text, $p_{\text{data}}(\text{'u'})$ is much higher if the previous symbol was a 'q'. (→ Problem 3.2)
  - Quantifying & modeling correlations requires more formal probability theory. → **next week**

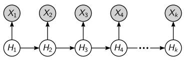
Outlook

- **Problem Set 3:**
  - proove that $D_{\mathrm{KL}}(p \,||\, q) \geq 0$
  - train a machine-learning model by minimizing $H\big(p_{\mathrm{data}}(\mathbf{x}), p_{\mathrm{model}}(\mathbf{x})\big)$ and use it to build a compression method for written natural language
- **Next week** (in our regular classroom):
  - probability theory
  - information theoretical quantitative measure of statistical dependencies
- **Afterwards:** expressive probabilistic (machine-learning) models

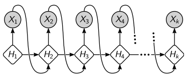
Markov Process



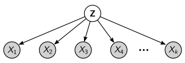
Hidden Markov Model



Autoregressive Model



Latent Variable Model


z
X_1
X_2
X_3
X_4
⋯
X_k