Very Quick Review of Probability

- Event space (e.g., $\mathcal{X}$, $\mathcal{Y}$)—in this class, usually discrete
Very Quick Review of Probability

- Event space (e.g., $\mathcal{X}$, $\mathcal{Y}$)—in this class, usually discrete
- Random variables (e.g., $X$, $Y$)
Very Quick Review of Probability

- Event space (e.g., $\mathcal{X}$, $\mathcal{Y}$)—in this class, usually discrete
- Random variables (e.g., $X$, $Y$)
- Typical statement: “random variable $X$ takes value $x \in \mathcal{X}$ with probability $p(X = x)$, or, in shorthand, $p(x)$”

Joint probability: $p(X = x, Y = y)$

Conditional probability: $p(X = x | Y = y)$

Always true: $p(X = x, Y = y) = p(X = x | Y = y) \cdot p(Y = y) = p(Y = y | X = x) \cdot p(X = x)$

Sometimes true: $p(X = x, Y = y) = p(X = x) \cdot p(Y = y)$
Very Quick Review of Probability

- Event space (e.g., $\mathcal{X}$, $\mathcal{Y}$)—in this class, usually discrete
- Random variables (e.g., $X$, $Y$)
- Typical statement: “random variable $X$ takes value $x \in \mathcal{X}$ with probability $p(X = x)$, or, in shorthand, $p(x)$”
- Joint probability: $p(X = x, Y = y)$
Very Quick Review of Probability

- Event space (e.g., $\mathcal{X}$, $\mathcal{Y}$)—in this class, usually discrete
- Random variables (e.g., $X$, $Y$)
- Typical statement: “random variable $X$ takes value $x \in \mathcal{X}$ with probability $p(X = x)$, or, in shorthand, $p(x)$”
- Joint probability: $p(X = x, Y = y)$
- Conditional probability: $p(X = x \mid Y = y)$
Very Quick Review of Probability

- Event space (e.g., $\mathcal{X}$, $\mathcal{Y}$)—in this class, usually discrete
- Random variables (e.g., $X$, $Y$)
- Typical statement: “random variable $X$ takes value $x \in \mathcal{X}$ with probability $p(X = x)$, or, in shorthand, $p(x)$”
- Joint probability: $p(X = x, Y = y)$
- Conditional probability: $p(X = x \mid Y = y) = \frac{p(X = x, Y = y)}{p(Y = y)}$
Very Quick Review of Probability

- Event space (e.g., \( \mathcal{X} \), \( \mathcal{Y} \))—in this class, usually discrete
- Random variables (e.g., \( X \), \( Y \))
- Typical statement: “random variable \( X \) takes value \( x \in \mathcal{X} \) with probability \( p(X = x) \), or, in shorthand, \( p(x) \)”
- Joint probability: \( p(X = x, Y = y) \)
- Conditional probability: \( p(X = x \mid Y = y) \)

\[
p(X = x, Y = y) = \frac{p(X = x, Y = y)}{p(Y = y)}
\]
- Always true:

\[
p(X = x, Y = y) = p(X = x \mid Y = y) \cdot p(Y = y) = p(Y = y \mid X = x) \cdot p(X = x)
\]
Very Quick Review of Probability

- Event space (e.g., $\mathcal{X}$, $\mathcal{Y}$)—in this class, usually discrete
- Random variables (e.g., $X$, $Y$)
- Typical statement: “random variable $X$ takes value $x \in \mathcal{X}$ with probability $p(X = x)$, or, in shorthand, $p(x)$”
- Joint probability: $p(X = x, Y = y)$
- Conditional probability: $p(X = x \mid Y = y) = \frac{p(X = x, Y = y)}{p(Y = y)}$
- Always true:
  \[ p(X = x, Y = y) = p(X = x \mid Y = y) \cdot p(Y = y) = p(Y = y \mid X = x) \cdot p(X = x) \]
- Sometimes true: $p(X = x, Y = y) = p(X = x) \cdot p(Y = y)$
Very Quick Review of Probability

- Event space (e.g., $X$, $Y$)—in this class, usually discrete
- Random variables (e.g., $X$, $Y$)
- Typical statement: “random variable $X$ takes value $x \in X$ with probability $p(X = x)$, or, in shorthand, $p(x)$”
- Joint probability: $p(X = x, Y = y)$
- Conditional probability: $p(X = x \mid Y = y)$
  \[ p(X = x \mid Y = y) = \frac{p(X = x, Y = y)}{p(Y = y)} \]
- Always true:
  \[ p(X = x, Y = y) = p(X = x \mid Y = y) \cdot p(Y = y) \]
  \[ = p(Y = y \mid X = x) \cdot p(X = x) \]
- Sometimes true: $p(X = x, Y = y) = p(X = x) \cdot p(Y = y)$
- The difference between true and estimated probability distributions
Language Models: Definitions

- $\mathcal{V}$ is a finite set of (discrete) symbols (😄 “words” or possibly characters); $V = |\mathcal{V}|$
- $\mathcal{V}^\dagger$ is the (infinite) set of sequences of symbols from $\mathcal{V}$ whose final symbol is ☀
- $p : \mathcal{V}^\dagger \rightarrow \mathbb{R}$, such that:
  - For any $x \in \mathcal{V}^\dagger$, $p(x) \geq 0$
  - $\sum_{x \in \mathcal{V}^\dagger} p(X = x) = 1$

(I.e., $p$ is a proper probability distribution.)

Language modeling: estimate $p$ from examples, $\mathbf{x}_{1:n} = \langle \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \rangle$. 
Immediate Objections

1. Why would we want to do this?
2. Are the nonnegativity and sum-to-one constraints really necessary?
3. Is “finite $\forall$” realistic?
Motivation: Noisy Channel Models

A pattern for modeling a pair of random variables, $X$ and $Y$:

source $\rightarrow Y \rightarrow \text{channel} \rightarrow X$
Motivation: Noisy Channel Models

A pattern for modeling a pair of random variables, $X$ and $Y$:

```
source $\rightarrow$ Y $\rightarrow$ channel $\rightarrow$ X
```

- $Y$ is the plaintext, the true message, the missing information, the output
Motivation: Noisy Channel Models

A pattern for modeling a pair of random variables, $X$ and $Y$:

source $\xrightarrow{}$ $Y$ $\xrightarrow{}$ channel $\xrightarrow{}$ $X$

- $Y$ is the plaintext, the true message, the missing information, the output
- $X$ is the ciphertext, the garbled message, the observable evidence, the input
Motivation: Noisy Channel Models

A pattern for modeling a pair of random variables, \( X \) and \( Y \):

\[
\begin{array}{c}
\text{source} \rightarrow Y \rightarrow \text{channel} \rightarrow X
\end{array}
\]

- \( Y \) is the plaintext, the true message, the missing information, the output
- \( X \) is the ciphertext, the garbled message, the observable evidence, the input
- Decoding: select \( y \) given \( X = x \).

\[
y^* = \arg\max_y p(y \mid x)
= \arg\max_y \frac{p(x \mid y) \cdot p(y)}{p(x)}
= \arg\max_y \left( \underbrace{p(x \mid y)}_{\text{channel model}} \cdot \underbrace{p(y)}_{\text{source model}} \right)
\]
Noisy Channel Example: Speech Recognition

source $\rightarrow$ sequence in $\mathcal{V}^\dagger$ $\rightarrow$ channel $\rightarrow$ acoustics

- Acoustic model defines $p(\text{sounds} \mid \mathcal{x})$ (channel)
- Language model defines $p(\mathcal{x})$ (source)
## Noisy Channel Example: Speech Recognition

Credit: Luke Zettlemoyer

<table>
<thead>
<tr>
<th>word sequence</th>
<th>( \log p(\text{acoustics} \mid \text{word sequence}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>the station signs are in deep in english</td>
<td>-14732</td>
</tr>
<tr>
<td>the stations signs are in deep in english</td>
<td>-14735</td>
</tr>
<tr>
<td>the station signs are in deep into english</td>
<td>-14739</td>
</tr>
<tr>
<td>the station 's signs are in deep in english</td>
<td>-14740</td>
</tr>
<tr>
<td>the station signs are in deep in the english</td>
<td>-14741</td>
</tr>
<tr>
<td>the station signs are indeed in english</td>
<td>-14757</td>
</tr>
<tr>
<td>the station 's signs are indeed in english</td>
<td>-14760</td>
</tr>
<tr>
<td>the station signs are indians in english</td>
<td>-14790</td>
</tr>
<tr>
<td>the station signs are indian in english</td>
<td>-14799</td>
</tr>
<tr>
<td>the stations signs are indians in english</td>
<td>-14807</td>
</tr>
<tr>
<td>the stations signs are indians and english</td>
<td>-14815</td>
</tr>
</tbody>
</table>
Also knowing nothing official about, but having guessed and inferred considerable about, the powerful new mechanized methods in cryptography—methods which I believe succeed even when one does not know what language has been coded—one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: “This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.”

Warren Weaver, 1955
Noisy Channel Examples

- Speech recognition
- Machine translation
- Optical character recognition
- Spelling and grammar correction
Immediate Objections

1. Why would we want to do this?
2. Are the nonnegativity and sum-to-one constraints really necessary?
3. Is “finite $\mathcal{V}$” realistic?
Evaluation: Perplexity

Intuitively, language models should assign high probability to real language they have not seen before. For out-of-sample ("held-out" or "test") data \( \bar{x}_{1:m} \):

- Probability of \( \bar{x}_{1:m} \) is \( \prod_{i=1}^{m} p(\bar{x}_i) \)

Perplexity (relative to \( \bar{x}_{1:m} \)) is \( 2^{-\frac{l}{M}} \)
Evaluation: Perplexity

Intuitively, language models should assign high probability to real language they have not seen before. For out-of-sample ("held-out" or "test") data $\bar{x}_{1:m}$:

- Probability of $\bar{x}_{1:m}$ is $\prod_{i=1}^{m} p(\bar{x}_i)$

- Log-probability of $\bar{x}_{1:m}$ is $\sum_{i=1}^{m} \log_2 p(\bar{x}_i)$
Evaluation: Perplexity

Intuitively, language models should assign high probability to real language they have not seen before. For out-of-sample ("held-out" or "test") data $\bar{x}_{1:m}$:

- Probability of $\bar{x}_{1:m}$ is $\prod_{i=1}^{m} p(\bar{x}_i)$
- Log-probability of $\bar{x}_{1:m}$ is $\sum_{i=1}^{m} \log_2 p(\bar{x}_i)$
- Average log-probability per word of $\bar{x}_{1:m}$ is

$$l = \frac{1}{M} \sum_{i=1}^{m} \log_2 p(\bar{x}_i)$$

if $M = \sum_{i=1}^{m} |\bar{x}_i|$ (total number of words in the corpus)
Evaluation: Perplexity

Intuitively, language models should assign high probability to real language they have not seen before. For out-of-sample ("held-out" or "test") data $\bar{x}_{1:m}$:

- Probability of $\bar{x}_{1:m}$ is
  $$\prod_{i=1}^{m} p(\bar{x}_i)$$

- Log-probability of $\bar{x}_{1:m}$ is
  $$\sum_{i=1}^{m} \log_2 p(\bar{x}_i)$$

- Average log-probability per word of $\bar{x}_{1:m}$ is
  $$l = \frac{1}{M} \sum_{i=1}^{m} \log_2 p(\bar{x}_i)$$
  if $M = \sum_{i=1}^{m} |\bar{x}_i|$ (total number of words in the corpus)

- Perplexity (relative to $\bar{x}_{1:m}$) is $2^{-l}$
Evaluation: Perplexity

Intuitively, language models should assign high probability to real language they have not seen before. For out-of-sample ("held-out" or "test") data \( \bar{x}_{1:m} \):

- Probability of \( \bar{x}_{1:m} \) is \( \prod_{i=1}^{m} p(\bar{x}_i) \)

- Log-probability of \( \bar{x}_{1:m} \) is \( \sum_{i=1}^{m} \log_2 p(\bar{x}_i) \)

- Average log-probability per word of \( \bar{x}_{1:m} \) is

\[
l = \frac{1}{M} \sum_{i=1}^{m} \log_2 p(\bar{x}_i)
\]

if \( M = \sum_{i=1}^{m} |\bar{x}_i| \) (total number of words in the corpus)

- Perplexity (relative to \( \bar{x}_{1:m} \)) is \( 2^{-l} \)

Lower is better.
Understanding Perplexity

\[\frac{-1}{M} \sum_{i=1}^{m} \log_2 p(\bar{x}_i)\]

It’s a branching factor!

- Assign probability of 1 to the test data \( \Rightarrow \) perplexity = 1
- Assign probability of \( \frac{1}{|V|} \) to every word \( \Rightarrow \) perplexity = \( |V| \)
- Assign probability of 0 to anything \( \Rightarrow \) perplexity = \( \infty \)
  - This motivates a stricter constraint than we had before:
    - For any \( x \in V^\dagger \), \( p(x) > 0 \)
Perplexity on conventionally accepted test sets is often reported in papers. Generally, I won’t discuss perplexity numbers much, because:

- Perplexity is only an intermediate measure of performance.
- Understanding the models is more important than remembering how well they perform on particular train/test sets.

If you’re curious, look up numbers in the literature; always take them with a grain of salt!
Immediate Objections

1. Why would we want to do this?
2. Are the nonnegativity and sum-to-one constraints really necessary?
3. Is “finite $\mathcal{V}$” realistic?
Is “finite ∃” realistic?

No
Is “finite ∃” realistic?

No
no
n0
-no
notta
Nº
/no
//no
(no
|no
The Language Modeling Problem

Input: \( x_{1:n} \) ("training data")
Output: \( p : \mathcal{V}^\dagger \rightarrow \mathbb{R}^+ \)

\( p \) should be a "useful" measure of plausibility (not grammaticality).
A Trivial Language Model

\[ p(\mathbf{x}) = \frac{|\{i \mid \mathbf{x}_i = \mathbf{x}\}|}{n} = \frac{c_{\mathbf{x}_1:n}(\mathbf{x})}{n} \]
A Trivial Language Model

\[ p(x) = \frac{|\{i \mid x_i = x\}|}{n} = \frac{c_{x_{1:n}}(x)}{n} \]

What if \( x \) is not in the training data?
Using the Chain Rule

\[ p(X = x) = \begin{pmatrix}
  p(X_1 = x_1) \\
  \cdot p(X_2 = x_2 \mid X_1 = x_1) \\
  \cdot p(X_3 = x_3 \mid X_{1:2} = x_{1:2}) \\
  \vdots \\
  \cdot p(X_\ell = \bigcirc \mid X_{1:\ell-1} = x_{1:\ell-1})
\end{pmatrix} \]

\[ = \prod_{j=1}^{\ell} p(X_j = x_j \mid X_{1:j-1} = x_{1:j-1}) \]
Unigram Model

\[ p(\mathbf{X} = \mathbf{x}) = \prod_{j=1}^{\ell} p(X_j = x_j \mid \mathbf{X}_{1:j-1} = \mathbf{x}_{1:j-1}) \]

Assumption

\[ \approx \prod_{j=1}^{\ell} p_{\mathbf{\theta}}(X_j = x_j) = \prod_{j=1}^{\ell} \theta_{x_j} \approx \prod_{j=1}^{\ell} \hat{\theta}_{x_j} \]

Maximum likelihood estimate:

\[ \forall v \in \mathcal{V}, \hat{\theta}_v = \frac{|\{i, j \mid [\mathbf{x}_i]_j = v\}|}{N} \]

\[ = \frac{c_{\mathbf{x}_{1:n}}(v)}{N} \]

where \( N = \sum_{i=1}^{n} |\mathbf{x}_i| \).

Also known as “relative frequency estimation.”
Responses to Some of Your Questions

I speak roughly 1.3 languages.

Homeworks are mostly programming assignments. They are public, but other than maybe some commentary, solutions won’t be public.

Interested in research?

- Faculty doing NLP at UW: http://nlp.washington.edu
- Summer internship application form: https://goo.gl/forms/mwirJD7utUMimVH92
Unigram Model

\[ p(X = x) = \prod_{j=1}^{\ell} p(X_j = x_j \mid X_{1:j-1} = x_{1:j-1}) \]

assumption

\[ \approx \prod_{j=1}^{\ell} p_{\theta}(X_j = x_j) = \prod_{j=1}^{\ell} \theta_{x_j} \approx \prod_{j=1}^{\ell} \hat{\theta}_{x_j} \]

Maximum likelihood estimate:

\[ \forall v \in V, \hat{\theta}_v = \frac{|\{i, j \mid [x_i]_j = v\}|}{N} = \frac{c_{x_{1:n}}(v)}{N} \]

where \( N = \sum_{i=1}^{n} |x_i| \).

Also known as “relative frequency estimation.”
Unigram Models: Assessment

**Pros:**
- Easy to understand
- Cheap
- Good enough for information retrieval (maybe)

**Cons:**
- “Bag of words” assumption is linguistically inaccurate
  - $p(\text{the the the the}) \gg p(\text{I want ice cream})$
- Data sparseness; high variance in the estimator
- “Out of vocabulary” problem
Markov Models ≡ n-gram Models

\[
p(X = x) = \prod_{j=1}^{\ell} p(X_j = x_j \mid X_{1:j-1} = x_{1:j-1})
\]

\[
\text{assumption} = \prod_{j=1}^{\ell} p_\theta(X_j = x_j \mid X_{j-n+1:j-1} = x_{j-n+1:j-1})
\]

(n − 1)th-order Markov assumption ≡ n-gram model

- Unigram model is the \( n = 1 \) case
- For a long time, trigram models (\( n = 3 \)) were widely used
- 5-gram models (\( n = 5 \)) are not uncommon now in MT
Estimating n-Gram Models

\[
p_\theta(x) = \prod_{j=1}^{\ell} \theta_{x_j} \prod_{j=1}^{\ell} \theta_{x_j|x_{j-1}} \prod_{j=1}^{\ell} \theta_{x_j|x_{j-2}x_{j-1}}
\]

Parameters:
\[\theta_v \quad \forall v \in V\]
\[\theta_{v|v'} \quad \forall v \in V, v' \in V \cup \{\text{○}\}\]
\[\theta_{v|v'|v''} \quad \forall v \in V, v', v'' \in V \cup \{\text{○}\}\]

MLE:
\[\frac{c(v)}{N} \quad \frac{c(v'v)}{c(v')} \quad \frac{c(v''v'v)}{c(v''v')}\]

General case:
\[\prod_{j=1}^{\ell} \theta_{x_j|x_{j-n+1:j-1}} \quad \theta_{v|h}, \forall v \in V, h \in (V \cup \{\text{○}\})^{n-1} \quad \frac{c(hv)}{c(h)}\]
The Problem with MLE

- The curse of dimensionality: the number of parameters grows exponentially in \( n \)
- Data sparseness: most n-grams will never be observed, even if they are linguistically plausible
- No one actually uses the MLE!
A few years ago, I’d have spent a whole lecture on this! 😊

- Simple method: add $\lambda > 0$ to every count (including zero-counts) before normalizing
- What makes it hard: ensuring that each $\theta \in \triangle \mathcal{Y}$
  - Otherwise, perplexity calculations break
- Longstanding champion: modified Kneser-Ney smoothing (Chen and Goodman, 1998)
- Stupid backoff: reasonable, easy solution when you don’t care about perplexity (Brants et al., 2007)
Interpolation

If $p$ and $q$ are both language models, then so is

$$\alpha p + (1 - \alpha)q$$

for any $\alpha \in [0, 1]$.

- This idea underlies many smoothing methods
- Often a new model $q$ only beats a reigning champion $p$ when interpolated with it
- How to pick the “hyperparameter” $\alpha$?
Algorithms To Know

- Score a sentence $x$
- Train from a corpus $x_{1:n}$
- Sample a sentence given $\theta$
n-gram Models: Assessment

Pros:
- Easy to understand
- Cheap (with modern hardware; Lin and Dyer, 2010)
- Good enough for machine translation, speech recognition, ...

Cons:
- Markov assumption is linguistically inaccurate
  - (But not as bad as unigram models!)
- Data sparseness; high variance in the estimator
- “Out of vocabulary” problem
Dealing with Out-of-Vocabulary Terms

- Define a special OOV or “unknown” symbol \texttt{UNK}. Transform some (or all) rare words in the training data to \texttt{UNK}.
  - 🙁 You cannot fairly compare two language models that apply different \texttt{UNK} treatments!
- Build a language model at the character level.
To-Do List

- Collins (2011); Jurafsky and Martin (2016)


Extras
Relative Frequency Estimation is the MLE
(Unigram Model)

The maximum likelihood estimation problem:

$$\max_{\theta \in \triangle |V|} p_\theta(x_{1:n})$$
Relative Frequency Estimation is the MLE
(Unigram Model)

Logarithm is a monotonic function.

\[
\max_{\theta \in \Delta} p_{\theta}(x_{1:n}) = \exp \max_{\theta \in \Delta} \log p_{\theta}(x_{1:n})
\]
Relative Frequency Estimation is the MLE
(Unigram Model)

Each sequence is an independent sample from the model.

$$\max_{\theta \in \Delta^{|V|}} \log p_{\theta}(x_{1:n}) = \max_{\theta \in \Delta^{|V|}} \log \prod_{i=1}^{n} p_{\theta}(x_i)$$
Relative Frequency Estimation is the MLE
(Unigram Model)

Plug in the form of the unigram model.

$$\max_{\theta \in \Delta^{|\mathcal{V}|}} \log \prod_{i=1}^{n} p_{\theta}(x_i) = \max_{\theta \in \Delta^{|\mathcal{V}|}} \log \prod_{i=1}^{n} \prod_{j=1}^{\ell_i} \theta_{[x_i]_j}$$
Relative Frequency Estimation is the MLE
(Unigram Model)

Log of product equals sum of logs.

\[
\max_{\theta \in \Delta^{|V|}} \log \prod_{i=1}^{n} \prod_{j=1}^{\ell_i} \theta_{[x_i]_j} = \max_{\theta \in \Delta^{|V|}} \sum_{i=1}^{n} \sum_{j=1}^{\ell_i} \log \theta_{[x_i]_j}
\]
Relative Frequency Estimation is the MLE
(Unigram Model)

Convert from tokens to types.

\[
\max_{\theta \in \Delta |\mathcal{V}|} \sum_{i=1}^{n} \sum_{j=1}^{\ell_i} \log \theta_{[x_i]_j} = \max_{\theta \in \Delta |\mathcal{V}|} \sum_{v \in \mathcal{V}} c_{x_{1:n}}(v) \log \theta_v
\]
Relative Frequency Estimation is the MLE
(Unigram Model)

Convert to a minimization problem (for consistency with textbooks).

\[
\max_{\theta \in \Delta^{|V|}} \sum_{v \in V} c_{x_1:n}(v) \log \theta_v = \min_{\theta \in \Delta^{|V|}} - \sum_{v \in V} c_{x_1:n}(v) \log \theta_v
\]
Relative Frequency Estimation is the MLE
(Unigram Model)

Lagrange multiplier to convert to a less constrained problem.

\[
\min_{\theta \in \Delta^{|\mathcal{V}|}} - \sum_{v \in \mathcal{V}} c_{x_{1:n}}(v) \log \theta_v
\]

\[
= \max_{\mu \geq 0} \min_{\theta \in \mathbb{R}^{|\mathcal{V}|}_{\geq 0}} - \sum_{v \in \mathcal{V}} c_{x_{1:n}}(v) \log \theta_v - \mu \left( 1 - \sum_{v \in \mathcal{V}} \theta_v \right)
\]

\[
= \min_{\theta \in \mathbb{R}^{|\mathcal{V}|}_{\geq 0}} \max_{\mu \geq 0} - \sum_{v \in \mathcal{V}} c_{x_{1:n}}(v) \log \theta_v - \mu \left( 1 - \sum_{v \in \mathcal{V}} \theta_v \right)
\]

Intuitively, if \( \sum_{v \in \mathcal{V}} \theta_v \) gets too big, \( \mu \) will push toward \(+\infty\).

For more about Lagrange multipliers, see Dan Klein’s tutorial (reference at the end of these slides).
Relative Frequency Estimation is the MLE
(Unigram Model)

Use first-order conditions to solve for $\theta$ in terms of $\mu$.

$$
\min_{\theta \in \mathbb{R}_{\geq 0}^{|V|}} \max_{\mu \geq 0} - \sum_{v \in V} c_x_{1:n}(v) \log \theta_v - \mu \left(1 - \sum_{v \in V} \theta_v\right)
$$

fixing $\mu$, for all $v$, set: $0 = \frac{\partial}{\partial \theta_v}$

$$
= -\frac{c_x_{1:n}(v)}{\theta_v} + \mu
$$

$$
\theta_v = \frac{c_x_{1:n}(v)}{\mu}
$$
Relative Frequency Estimation is the MLE
(Unigram Model)

Plug in for each $\theta_v$.

$$
\min_{\theta \in \mathbb{R}_{\geq 0}^{|\mathcal{V}|}} \max_{\mu \geq 0} - \sum_{v \in \mathcal{V}} c_{x_1:n}(v) \log \theta_v - \mu \left(1 - \sum_{v \in \mathcal{V}} \theta_v\right)
$$

$$
= \max_{\mu \geq 0} - \sum_{v \in \mathcal{V}} c_{x_1:n}(v) \log \frac{c_{x_1:n}(v)}{\mu} - \mu \left(1 - \sum_{v \in \mathcal{V}} \frac{c_{x_1:n}(v)}{\mu}\right)
$$

Remember: $\forall v \in \mathcal{V}, \theta_v = \frac{c_{x_1:n}(v)}{\mu}$
Relative Frequency Estimation is the MLE
(Unigram Model)

Rearrange terms \((a \log \frac{a}{b} = a \log a - a \log b\) and \(N = \sum_{v \in V} c_{x_1:n}(v))\).

\[
\begin{align*}
\max_{\mu \geq 0} & - \sum_{v \in V} c_{x_1:n}(v) \log \frac{c_{x_1:n}(v)}{\mu} - \mu \left(1 - \sum_{v \in V} \frac{c_{x_1:n}(v)}{\mu}\right) \\
& = \max_{\mu \geq 0} - \sum_{v \in V} c_{x_1:n}(v) \log c_{x_1:n}(v) + N \log \mu - \mu + N
\end{align*}
\]

Remember: \(\forall v \in V, \theta_v = \frac{c_{x_1:n}(v)}{\mu}\)
Relative Frequency Estimation is the MLE
(Unigram Model)

Use first-order conditions to solve for $\mu$.

$$\max_{\mu \geq 0} - \sum_{v \in \mathcal{V}} c_{x_1:n}(v) \log c_{x_1:n}(v) + N \log \mu - \mu + N$$

set: $0 = \frac{\partial}{\partial \mu} \left( \frac{N}{\mu} - 1 \right)$

$\mu = N$

Remember: $\forall v \in \mathcal{V}, \theta_v = \frac{c_{x_1:n}(v)}{\mu}$
Relative Frequency Estimation is the MLE
(Unigram Model)

Plug in for $\mu$.

$$\max_{\mu \geq 0} - \sum_{v \in \mathcal{V}} c_{x_1:n}(v) \log c_{x_1:n}(v) + N \log \mu - \mu + N$$

$$= - \sum_{v \in \mathcal{V}} c_{x_1:n}(v) \log c_{x_1:n}(v) + N \log N$$

$$\forall v \in \mathcal{V}, \theta_v = \frac{c_{x_1:n}(v)}{\mu} = \frac{c_{x_1:n}(v)}{N}$$

... and that’s the relative frequency estimate!
Language Models as (Weighted) Finite-State Automata

(Deterministic) finite-state automaton:

- Set of \( k \) states \( S \)
  - Initial state \( s_0 \in S \)
  - Final states \( F \subseteq S \)
- Alphabet \( \Sigma \)
- Transitions \( \delta : S \times \Sigma \rightarrow S \)

A length \( \ell \) string \( x \) is in the language of the automaton iff there is a path \( \langle s_0, \ldots, s_\ell \rangle \) such that \( s_\ell \in F \) and

\[
\bigwedge_{i=1}^{\ell} \left[ s_i = \delta(s_{i-1}, x_i) \right]
\]
Language Models as (Weighted) Finite-State Automata

(Deterministic) finite-state automaton:

- Set of \( k \) states \( S \)
  - Initial state \( s_0 \in S \)
  - Final states \( \mathcal{F} \subseteq S \)
- Alphabet \( \Sigma \)
- Transitions \( \delta : S \times \Sigma \rightarrow S \times \mathbb{R}_{>0} \)

A weighted FSA defines a weight for every transition; e.g.,

\[
w(h, v, \delta(h, v)) = \theta_{v|h}
\]

A length \( \ell \) string \( x \) is in the language of the automaton iff there is a path \( \langle s_0, \ldots, s_\ell \rangle \) such that \( s_\ell \in \mathcal{F} \) and

\[
\bigwedge_{i=1}^{\ell} \left[ [s_i = \delta(s_{i-1}, x_i)] \right]
\]

The score of the string is the product of transition weights.

\[
score(x) \prod_{i=1}^{\ell} w(h_i, x_i, \delta(h_i, x_i))
\]
Suppose we have a hard clustering of $\mathcal{V}$, $\mathrm{cl} : \mathcal{V} \rightarrow \{1, \ldots, k\}$, where $k \ll |\mathcal{V}|$.

**n-gram**

$$p_\theta(x) = \prod_{j=1}^{\ell} \theta_{x_j|\mathbf{x}_{j-n+1:j-1}}$$

**Parameters:**

$$\theta_v|h$$

$$\forall v \in \mathcal{V}, h \in (\mathcal{V} \cup \{\bigcirc\})^{n-1}$$

**MLE:**

$$\frac{c(hv)}{c(h)}$$

**class-based**

$$p_\theta(x) = \prod_{j=1}^{\ell} \theta_{x_j|\mathrm{cl}(x_j)} \gamma_{\mathrm{cl}(x_j)|\mathrm{cl}(x_{j-1})}$$

**Parameters:**

$$\theta_v|\mathrm{cl}(v)$$

$$\forall v \in \mathcal{V}$$

$$\gamma_{i|j}$$

$$\forall i, j \in \{1, \ldots, k\}$$

**MLE:**

$$\frac{c(v)}{c(\mathrm{cl}(v))} \quad \frac{c(j)}{c(ji)}$$