Natural Language Processing (CSE 517): Language Models

Noah Smith

University of Washington
nasmith@cs.washington.edu

January 4–6, 2016

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, $p(x)$"

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: $p(X = x \mid Y = y)$
- 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: $p(X = x \mid Y = y)$
- 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., $\mathcal{X}$, $\mathcal{Y}$)—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: $p(X = x \mid Y = y)$
- 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 $\boldsymbol{x} \in \mathcal{V}^\dagger$, $p(\boldsymbol{x}) \geq 0$
  - $\displaystyle\sum_{\boldsymbol{x} \in \mathcal{V}^\dagger} p(\boldsymbol{x}) = 1$

  ! (I.e., p is a proper probability distribution.)
  More careful r.v. notation: $p(\boldsymbol{X} = \boldsymbol{x})$

Language modeling: estimate p from examples, $\boldsymbol{x}_{1:n}$.

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?

Motivation: Noisy Channel Models

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

$$\boxed{\text{source}} \longrightarrow Y \longrightarrow \boxed{\text{channel}} \longrightarrow X$$

Motivation: Noisy Channel Models

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

$$\boxed{\text{source}} \longrightarrow Y \longrightarrow \boxed{\text{channel}} \longrightarrow X$$

- 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:

$$\boxed{\text{source}} \longrightarrow Y \longrightarrow \boxed{\text{channel}} \longrightarrow X$$

- 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:

$$\boxed{\text{source}} \longrightarrow Y \longrightarrow \boxed{\text{channel}} \longrightarrow X$$

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

$$\begin{aligned} y^* &= \operatorname*{argmax}_y p(y \mid x) \\ &= \operatorname*{argmax}_y \frac{p(x \mid y) \cdot p(y)}{p(x)} \\ &= \operatorname*{argmax}_y \underbrace{p(x \mid y)}_{\text{channel model}} \cdot \underbrace{p(y)}_{\text{source model}} \end{aligned}$$

Noisy Channel Example: Speech Recognition

$$\boxed{\text{source}} \longrightarrow \text{sequence in } \mathcal{V}^{\dagger} \longrightarrow \boxed{\text{channel}} \longrightarrow \text{acoustics}$$

- Acoustic model defines $p(\text{sounds} \mid \boldsymbol{x})$ (channel)
- Language model defines $p(\boldsymbol{x})$ (source)

Noisy Channel Example: Speech Recognition

Credit: Luke Zettlemoyer

word sequence	$\log p(\text{acoustics} \mid \text{word sequence})$
the station signs are in deep in english	-14732
the stations signs are in deep in english	-14735
the station signs are in deep into english	-14739
the station 's signs are in deep in english	-14740
the station signs are in deep in the english	-14741
the station signs are indeed in english	-14757
the station 's signs are indeed in english	-14760
the station signs are indians in english	-14790
the station signs are indian in english	-14799
the stations signs are indians in english	-14807
the stations signs are indians and english	-14815

Noisy Channel Example: Machine Translation

> *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

"Conditional" Language Models

Instead of $p(\boldsymbol{X})$, model $p(\boldsymbol{X} \mid \mathit{Context})$.

- $\mathit{Context}$ could be an input (acoustics, source-language sentence, image of text) . . . or it could be something else (visual input, stock prices, . . .)
- Made possible by advances in machine learning!

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 $\bar{\boldsymbol{x}}_{1:m}$:

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

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 $\bar{\boldsymbol{x}}_{1:m}$:

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

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 $\bar{\boldsymbol{x}}_{1:m}$:

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

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

if $M = \sum_{i=1}^{m} |\bar{\boldsymbol{x}}|_i$

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 $\bar{\boldsymbol{x}}_{1:m}$:

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

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

  if $M = \sum_{i=1}^{m} |\bar{\boldsymbol{x}}|_i$
- Perplexity (relative to $\bar{\boldsymbol{x}}_{1:m}$) 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 $\bar{\boldsymbol{x}}_{1:m}$:

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

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

  if $M = \sum_{i=1}^{m} |\bar{\boldsymbol{x}}|_i$
- Perplexity (relative to $\bar{\boldsymbol{x}}_{1:m}$) is 2^{-l}

Lower is better.

Understanding Perplexity

$$2^{-\frac{1}{M}\sum_{i=1}^{m}\log_2 p(\bar{\boldsymbol{x}}_i)}$$

It's a branching factor!

- Assign probability of 1 to the test data $\Rightarrow$ perplexity $= 1$
- Assign probability of $\frac{1}{|\mathcal{V}|}$ to every word $\Rightarrow$ perplexity $= |\mathcal{V}|$
- Assign probability of 0 to *anything* $\Rightarrow$ perplexity $= \infty$
  - This motivates a stricter constraint than we had before:
    - For any $\boldsymbol{x} \in \mathcal{V}^\dagger$, $p(\boldsymbol{x}) > 0$

Perplexity

- 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 $\mathcal{V}$" realistic?

No

Is “finite $\mathcal{V}$” realistic?

No
no
n0
-no
notta
Nº
/no
//no
(no
|no

The Language Modeling Problem

Input: $\boldsymbol{x}_{1:n}$ ("training data")
Output: $p : \mathcal{V}^{\dagger} \rightarrow \mathbb{R}^{+}$
☺ p should be a "useful" measure of plausibility (not grammaticality).

A Trivial Language Model

$$
\begin{aligned}
p(\boldsymbol{x}) &= \frac{|\{i \mid \boldsymbol{x}_i = \boldsymbol{x}\}|}{n} \\
&= \frac{c_{\boldsymbol{x}_{1:n}}(\boldsymbol{x})}{n}
\end{aligned}
$$

A Trivial Language Model

$$
\begin{aligned}
p(\boldsymbol{x}) &= \frac{|\{i \mid \boldsymbol{x}_i = \boldsymbol{x}\}|}{n} \\
&= \frac{c_{\boldsymbol{x}_{1:n}}(\boldsymbol{x})}{n}
\end{aligned}
$$

What if x is not in the training data?

Using the Chain Rule

$$
\begin{aligned}
p(\boldsymbol{X} = \boldsymbol{x}) &= \left(\begin{array}{l}
p(X_1 = x_1) \\
\cdot\, p(X_2 = x_2 \mid X_1 = x_1) \\
\cdot\, p(X_3 = x_3 \mid \boldsymbol{X}_{1:2} = \boldsymbol{x}_{1:2}) \\
\vdots \\
\cdot\, p(X_\ell = \bigcirc \mid \boldsymbol{X}_{1:\ell-1} = \boldsymbol{x}_{1:\ell-1})
\end{array}\right) \\
&= \prod_{j=1}^{\ell} p(X_j = x_j \mid \boldsymbol{X}_{1:j-1} = \boldsymbol{x}_{1:j-1})
\end{aligned}
$$

Unigram Model

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

$$\stackrel{\text{assumption}}{=} \prod_{j=1}^{\ell} p(X_j = x_j)$$

Modeled by:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \prod_{j=1}^{\ell} \theta_{x_j}$$

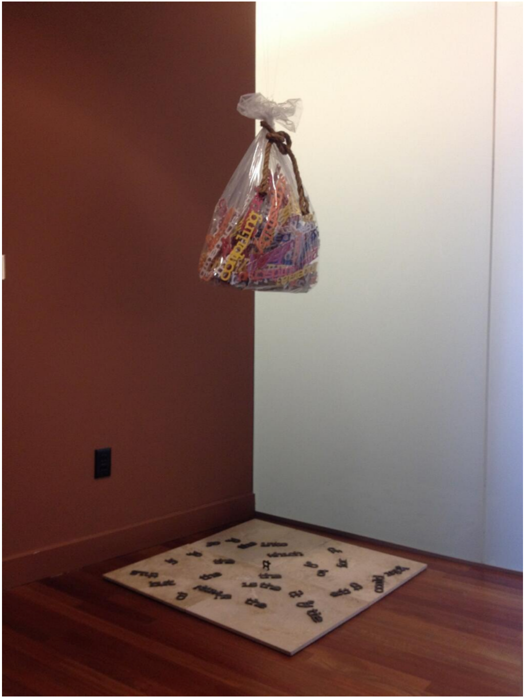
where $\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}$.

Maximum likelihood estimate:

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

$$= \frac{c_{\boldsymbol{x}_{1:n}}(v)}{N}$$

where $N = \sum_{i=1}^{n} |\boldsymbol{x}_i|$.
Also known as "relative frequency estimation."



Unigram Model

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

$$\stackrel{\text{assumption}}{=} \prod_{j=1}^{\ell} p(X_j = x_j)$$

Modeled by:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \prod_{j=1}^{\ell} \theta_{x_j}$$

where $\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}$.

Maximum likelihood estimate:

$$\forall v \in \mathcal{V}, \hat{\theta}_v = \frac{|\{i, j \mid [\boldsymbol{x}_i]_j = v\}|}{N} = \frac{c_{\boldsymbol{x}_{1:n}}(v)}{N}$$

where $N = \sum_{i=1}^{n} |\boldsymbol{x}_i|$.
Also known as "relative frequency estimation."

Relative Frequency Estimation is the MLE
(Unigram Model)

The maximum likelihood estimation problem:

$$\max_{\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:n})$$

Relative Frequency Estimation is the MLE

(Unigram Model)

Logarithm is a monotonic function.

$$\max_{\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:n}) = \exp \max_{\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:n})$$

Relative Frequency Estimation is the MLE

(Unigram Model)

Each sequence is an independent sample from the model.

$$\max_{\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:n}) = \max_{\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}} \log \prod_{i=1}^{n} p_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

Relative Frequency Estimation is the MLE
(Unigram Model)

Plug in the form of the unigram model.

$$\max_{\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}} \log \prod_{i=1}^{n} p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \max_{\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}} \log \prod_{i=1}^{n} \prod_{j=1}^{\ell_i} \theta_{[\boldsymbol{x}_i]_j}$$

Relative Frequency Estimation is the MLE
(Unigram Model)

Log of product equals sum of logs.

$$\max_{\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}} \log \prod_{i=1}^{n} \prod_{j=1}^{\ell_i} \theta_{[\boldsymbol{x}_i]_j} = \max_{\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}} \sum_{i=1}^{n} \sum_{j=1}^{\ell_i} \log \theta_{[\boldsymbol{x}_i]_j}$$

Relative Frequency Estimation is the MLE
(Unigram Model)

Convert from tokens to types.

$$\max_{\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}} \sum_{i=1}^{n} \sum_{j=1}^{\ell_i} \log \theta_{[\boldsymbol{x}_i]_j} = \max_{\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}} \sum_{v \in \mathcal{V}} c_{\boldsymbol{x}_{1:n}}(v) \log \theta_v$$

Relative Frequency Estimation is the MLE

(Unigram Model)

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

$$\max_{\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}} \sum_{v \in \mathcal{V}} c_{\boldsymbol{x}_{1:n}}(v) \log \theta_v = \min_{\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}} - \sum_{v \in \mathcal{V}} c_{\boldsymbol{x}_{1:n}}(v) \log \theta_v$$

Relative Frequency Estimation is the MLE

(Unigram Model)

Lagrange multiplier to convert to a less constrained problem.

$$
\begin{aligned}
& \min_{\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}} - \sum_{v \in \mathcal{V}} c_{\boldsymbol{x}_{1:n}}(v) \log \theta_v \\
= & \max_{\mu \geq 0} \min_{\boldsymbol{\theta} \in \mathbb{R}_{\geq 0}^{|\mathcal{V}|}} - \sum_{v \in \mathcal{V}} c_{\boldsymbol{x}_{1:n}}(v) \log \theta_v - \mu \left(1 - \sum_{v \in \mathcal{V}} \theta_v \right) \\
= & \min_{\boldsymbol{\theta} \in \mathbb{R}_{\geq 0}^{|\mathcal{V}|}} \max_{\mu \geq 0} - \sum_{v \in \mathcal{V}} c_{\boldsymbol{x}_{1:n}}(v) \log \theta_v - \mu \left(1 - \sum_{v \in \mathcal{V}} \theta_v \right)
\end{aligned}
$$

Intuitively, if $\sum_{v \in \mathcal{V}} \theta_v$ gets too big, μ 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 $\boldsymbol{\theta}$ in terms of μ.

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

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

$$= \frac{-c_{\boldsymbol{x}_{1:n}}(v)}{\theta_v} + \mu$$

$$\theta_v = \frac{c_{\boldsymbol{x}_{1:n}}(v)}{\mu}$$

Relative Frequency Estimation is the MLE

(Unigram Model)

Plug in for each θ_v.

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

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

Remember: $\boxed{\forall v \in \mathcal{V}, \theta_v = \frac{c_{\boldsymbol{x}_{1:n}}(v)}{\mu}}$

Relative Frequency Estimation is the MLE

(Unigram Model)

Rearrange terms ($a \log \frac{a}{b} = a \log a - a \log b$ and $N = \sum_{v \in \mathcal{V}} c_{\boldsymbol{x}_{1:n}}(v)$).

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

$$= \max_{\mu \geq 0} - \sum_{v \in \mathcal{V}} c_{\boldsymbol{x}_{1:n}}(v) \log c_{\boldsymbol{x}_{1:n}}(v) + N \log \mu - \mu + N$$

Remember: $\boxed{\forall v \in \mathcal{V}, \theta_v = \frac{c_{\boldsymbol{x}_{1:n}}(v)}{\mu}}$

Relative Frequency Estimation is the MLE
(Unigram Model)

Use first-order conditions to solve for μ.

$$\max_{\mu \geq 0} - \sum_{v \in \mathcal{V}} c_{\boldsymbol{x}_{1:n}}(v) \log c_{\boldsymbol{x}_{1:n}}(v) + N \log \mu - \mu + N$$

$$\begin{aligned} \text{set: } 0 &= \frac{\partial}{\partial \mu} \\ &= \frac{N}{\mu} - 1 \\ \mu &= N \end{aligned}$$

Remember: $\boxed{\forall v \in \mathcal{V}, \theta_v = \frac{c_{\boldsymbol{x}_{1:n}}(v)}{\mu}}$

Relative Frequency Estimation is the MLE
(Unigram Model)

Plug in for μ.

$$\max_{\mu \geq 0} - \sum_{v \in \mathcal{V}} c_{\boldsymbol{x}_{1:n}}(v) \log c_{\boldsymbol{x}_{1:n}}(v) + N \log \mu - \mu + N$$

$$= - \sum_{v \in \mathcal{V}} c_{\boldsymbol{x}_{1:n}}(v) \log c_{\boldsymbol{x}_{1:n}}(v) + N \log N$$

$$\boxed{\forall v \in \mathcal{V}, \theta_v = \frac{c_{\boldsymbol{x}_{1:n}}(v)}{\mu}} = \frac{c_{\boldsymbol{x}_{1:n}}(v)}{N}$$

... and that's the relative frequency estimate!

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 $\equiv$ n-gram Models

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

$$\stackrel{\text{assumption}}{=} \prod_{j=1}^{\ell} p(X_j = x_j \mid \boldsymbol{X}_{j-\mathsf{n}+1:j-1} = \boldsymbol{x}_{j-\mathsf{n}+1:j-1})$$

$(\mathsf{n}-1)$th-order Markov assumption $\equiv$ n-gram model

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

Estimating n-Gram Models

	unigram	bigram	trigram
$p_{\boldsymbol{\theta}}(\boldsymbol{x}) =$	$\prod_{j=1}^{\ell} \theta_{x_j}$	$\prod_{j=1}^{\ell} \theta_{x_j \mid x_{j-1}}$	$\prod_{j=1}^{\ell} \theta_{x_j \mid x_{j-2} x_{j-1}}$
Parameters:	θ_v $\forall v \in \mathcal{V}$	$\theta_{v \mid v'}$ $\forall v \in \mathcal{V}, v' \in \mathcal{V} \cup \{\bigcirc\}$	$\theta_{v \mid v'' v'}$ $\forall v \in \mathcal{V}, v', v'' \in \mathcal{V} \cup \{\bigcirc\}$
MLE:	$\frac{c(v)}{N}$	$\frac{c(v'v)}{c(v')}$	$\frac{c(v''v'v)}{c(v''v')}$

General case:

$$\prod_{j=1}^{\ell} \theta_{x_j \mid \boldsymbol{x}_{j-\mathsf{n}+1:j-1}} \qquad \theta_{v \mid \boldsymbol{h}}, \ \forall v \in \mathcal{V}, \boldsymbol{h} \in (\mathcal{V} \cup \{\bigcirc\})^{\mathsf{n}-1} \qquad \frac{c(\boldsymbol{h}v)}{c(\boldsymbol{h})}$$

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!

Smoothing

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 $\boldsymbol{\theta} \in \triangle^{|\mathcal{V}|}$
  - 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 α?

Algorithms To Know

- Score a sentence $\boldsymbol{x}$
- Train from a corpus $\boldsymbol{x}_{1:n}$
- Sample a sentence given $\boldsymbol{\theta}$

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

Class-Based Language Models

Brown et al. (1992)

Suppose we have a hard clustering of $\mathcal{V}$, $\mathsf{cl} : \mathcal{V} \to \{1, \ldots, k\}$, where $k \ll |\mathcal{V}|$.

	n-gram	class-based	
$p_{\boldsymbol{\theta}}(\boldsymbol{x}) =$	$\prod_{j=1}^{\ell} \theta_{x_j \mid \boldsymbol{x}_{j-\mathsf{n}+1:j-1}}$	$\prod_{j=1}^{\ell} \theta_{x_j \mid \mathsf{cl}(x_j)} \gamma_{\mathsf{cl}(x_j) \mid \mathsf{cl}(x_{j-1})}$	
Parameters:	$\theta_{v \mid \boldsymbol{h}}$ $\forall v \in \mathcal{V}, \boldsymbol{h} \in (\mathcal{V} \cup \{\bigcirc\})^{\mathsf{n}-1}$	$\theta_{v \mid \mathsf{cl}(v)}$ $\forall v \in \mathcal{V}$	$\gamma_{i \mid j}$ $\forall i, j \in \{1, \ldots, k\}$
MLE:	$\dfrac{c(\boldsymbol{h}v)}{c(\boldsymbol{h})}$	$\dfrac{c(v)}{c(\mathsf{cl}(v))}$	$\dfrac{c(j)}{c(ji)}$

Language Models as (Weighted) Finite-State Automata

(Deterministic) finite-state automaton:

- Set of k states $\mathcal{S}$
  - Initial state $s_0 \in \mathcal{S}$
  - Final states $\mathcal{F} \subseteq \mathcal{S}$
- Alphabet Σ
- Transitions $\delta : \mathcal{S} \times \Sigma \to \mathcal{S}$

A length ℓ string $\boldsymbol{x}$ 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

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

Language Models as (Weighted) Finite-State Automata

(Deterministic) finite-state automaton:

- Set of k states $\mathcal{S}$ — histories
  - Initial state $s_0 \in \mathcal{S}$
  - Final states $\mathcal{F} \subseteq \mathcal{S}$ — histories ending in
- Alphabet Σ — $\mathcal{V}$
- Transitions $\delta : \mathcal{S} \times \Sigma \to \mathcal{S} \quad \times \mathbb{R}_{>0}$

  A **weighted** FSA defines a weight for every transition; e.g.,
  $w(\boldsymbol{h}, v, \delta(\boldsymbol{h}, v)) = \theta_{v|\boldsymbol{h}}$

A length ℓ string $\boldsymbol{x}$ 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

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

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

$$score(\boldsymbol{x}) \prod_{i=1}^{\ell} w(\boldsymbol{h}_i, x_i, \delta(\boldsymbol{h}_i, x_i))$$

Dealing with Out-of-Vocabulary Terms

- Define a special OOV or "unknown" symbol UNK. Transform some (or all) rare words in the training data to UNK.
  - ☹ You cannot fairly compare two language models that apply different UNK treatments!
- Build a language model at the *character* level.

Readings and Reminders

- Collins (2011); Jurafsky and Martin (2015)
- Submit a suggestion for an exam question by Friday at 5pm.
- Noah's office hours: Friday 1:30–2:30 in CSE 532.

References I

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large language models in machine translation. In *Proc. of EMNLP-CoNLL*, 2007.

Peter F. Brown, Peter V. Desouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University, 1998.

Michael Collins. Course notes for COMS w4705: Language modeling, 2011. URL `http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/lm.pdf`.

Daniel Jurafsky and James H. Martin. N-grams (draft chapter), 2015. URL `https://web.stanford.edu/~jurafsky/slp3/4.pdf`.

Dan Klein. Lagrange multipliers without permanent scarring, Undated. URL `https://www.cs.berkeley.edu/~klein/papers/lagrange-multipliers.pdf`.

Jimmy Lin and Chris Dyer. *Data-Intensive Text Processing with MapReduce*. Morgan and Claypool, 2010.