Natural Language Processing (CSE 517): Featurized Language Models

Noah Smith

University of Washington
nasmith@cs.washington.edu

April 4, 2018

Quick Review

A language model is a probability distribution over $\mathcal{V}^\dagger$.

Typically p decomposes into probabilities $p(x_i \mid \boldsymbol{h}_i)$.

- n-gram: $\boldsymbol{h}_i$ is $(\mathsf{n} - 1)$ previous symbols
- Probabilities are estimated from data.

Quick Review

A language model is a probability distribution over $\mathcal{V}^\dagger$.

Typically p decomposes into probabilities $p(x_i \mid \boldsymbol{h}_i)$.

- n-gram: $\boldsymbol{h}_i$ is $(\mathsf{n} - 1)$ previous symbols
- Probabilities are estimated from data.
  - Why?

Quick Review

A language model is a probability distribution over $\mathcal{V}^\dagger$.

Typically p decomposes into probabilities $p(x_i \mid \boldsymbol{h}_i)$.

- n-gram: $\boldsymbol{h}_i$ is $(\mathsf{n} - 1)$ previous symbols
- Probabilities are estimated from data.

Today: a few more details, then log-linear language models

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" 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

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.

Class-Based Language Models

Brown et al. (1992)

Suppose we have a hard clustering of $\mathcal{V}$, $\mathsf{cl} : \mathcal{V} \rightarrow \{1, \ldots, k\}$, 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:	$\theta_{v \mid \boldsymbol{h}}$	$\theta_{v \mid \mathsf{cl}(v)}$	$\gamma_{i \mid j}$
	$\forall v \in \mathcal{V}, \boldsymbol{h} \in (\mathcal{V} \cup \{\bigcirc\})^{\mathsf{n}-1}$	$\forall v \in \mathcal{V}$	$\forall i, j \in \{1, \ldots, k\}$
MLE:	$\frac{c(\boldsymbol{h}v)}{c(\boldsymbol{h})}$	$\frac{c(v)}{c(\mathsf{cl}(v))}$	$\frac{c(j)}{c(ji)}$

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.

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

What's wrong with n-grams?

Data sparseness: most histories and most words will be seen only rarely (if at all).

What's wrong with n-grams?

Data sparseness: most histories and most words will be seen only rarely (if at all).

Next central idea: teach histories and words how to share.

What's wrong with n-grams?

Data sparseness: most histories and most words will be seen only rarely (if at all).

What's wrong with n-grams?

Data sparseness: most histories and most words will be seen only rarely (if at all).

Central idea today: teach histories and words how to share.

Log-Linear Models: Definitions

We define a conditional log-linear model $p(Y \mid X)$ as:

- $\mathcal{Y}$ is the set of events/outputs (☺ for language modeling, $\mathcal{V}$)
- $\mathcal{X}$ is the set of contexts/inputs (☺ for n-gram language modeling, $\mathcal{V}^{\mathsf{n}-1}$)
- $\boldsymbol{\phi} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$ is a feature vector function
- $\mathbf{w} \in \mathbb{R}^d$ are the model parameters

$$p_{\mathbf{w}}(Y = y \mid X = x) = \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y)}{\sum_{y' \in \mathcal{Y}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y')}$$

Breaking It Down

$$p_{\mathbf{w}}(Y = y \mid X = x) = \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y)}{\displaystyle\sum_{y' \in \mathcal{Y}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y)}$$

Breaking It Down

$$p_{\mathbf{w}}(Y = y \mid X = x) = \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y)}{\sum_{y' \in \mathcal{Y}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y)}$$

linear score $\mathbf{w} \cdot \boldsymbol{\phi}(x, y)$

Breaking It Down

$$p_{\mathbf{w}}(Y = y \mid X = x) = \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y)}{\sum_{y' \in \mathcal{Y}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y)}$$

linear score	$\mathbf{w} \cdot \boldsymbol{\phi}(x, y)$
nonnegative	$\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y)$

Breaking It Down

$$p_{\mathbf{w}}(Y = y \mid X = x) = \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y)}{\displaystyle\sum_{y' \in \mathcal{Y}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y)}$$

linear score $\quad \mathbf{w} \cdot \boldsymbol{\phi}(x, y)$

nonnegative $\quad \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y)$

normalizer $\quad \displaystyle\sum_{y' \in \mathcal{Y}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y') = Z_{\mathbf{w}}(x)$

Breaking It Down

$$p_{\mathbf{w}}(Y = y \mid X = x) = \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y)}{\sum\limits_{y' \in \mathcal{Y}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y)}$$

linear score	$\mathbf{w} \cdot \boldsymbol{\phi}(x, y)$
nonnegative	$\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y)$
normalizer	$\sum\limits_{y' \in \mathcal{Y}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, y') = Z_{\mathbf{w}}(x)$

"Log-linear" comes from the fact that:

$$\log p_{\mathbf{w}}(Y = y \mid X = x) = \mathbf{w} \cdot \boldsymbol{\phi}(x, y) - \underbrace{\log Z_{\mathbf{w}}(x)}_{\text{constant in } y}$$

This is an instance of the family of **generalized linear models**.

Special Case: Logistic Regression

Consider the case where $\mathcal{Y} = \{+1, -1\}$.

$$p_{\mathbf{w}}(Y = +1 \mid x) = \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, +1)}{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, +1) + \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, -1)}$$

Special Case: Logistic Regression

Consider the case where $\mathcal{Y} = \{+1, -1\}$.

$$\begin{aligned} p_{\mathbf{w}}(Y = +1 \mid x) &= \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, +1)}{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, +1) + \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, -1)} \\ &= \operatorname{logit}^{-1}\left(\mathbf{w} \cdot (\boldsymbol{\phi}(x, +1) - \boldsymbol{\phi}(x, -1))\right) \end{aligned}$$

Special Case: Logistic Regression

Consider the case where $\mathcal{Y} = \{+1, -1\}$.

$$
\begin{aligned}
p_{\mathbf{w}}(Y = +1 \mid x) &= \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, +1)}{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, +1) + \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, -1)} \\
&= \operatorname{logit}^{-1} \left(\mathbf{w} \cdot \left(\boldsymbol{\phi}(x, +1) - \boldsymbol{\phi}(x, -1) \right) \right) \\
&\overset{\text{notation change}}{=} \operatorname{logit}^{-1} \left(\mathbf{w} \cdot \mathbf{f}(x) \right)
\end{aligned}
$$

Special Case: Logistic Regression

Consider the case where $\mathcal{Y} = \{+1, -1\}$.

$$
\begin{aligned}
p_{\mathbf{w}}(Y = +1 \mid x) &= \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, +1)}{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, +1) + \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, -1)} \\
&= \operatorname{logit}^{-1}\left(\mathbf{w} \cdot (\boldsymbol{\phi}(x, +1) - \boldsymbol{\phi}(x, -1))\right) \\
&\overset{\text{notation change}}{=} \operatorname{logit}^{-1}\left(\mathbf{w} \cdot \mathbf{f}(x)\right)
\end{aligned}
$$

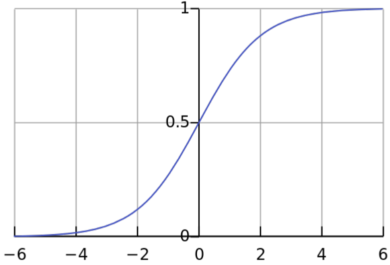
- Should be familiar, if you know about logistic regression.

Special Case: Logistic Regression

Consider the case where $\mathcal{Y} = \{+1, -1\}$.

$$p_{\mathbf{w}}(Y = +1 \mid x) = \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, +1)}{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, +1) + \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, -1)}$$
$$= \operatorname{logit}^{-1}\left(\mathbf{w} \cdot (\boldsymbol{\phi}(x, +1) - \boldsymbol{\phi}(x, -1))\right)$$
$$\stackrel{\text{notation change}}{=} \operatorname{logit}^{-1}\left(\mathbf{w} \cdot \mathbf{f}(x)\right)$$

- Should be familiar, if you know about logistic regression.



Special Case: Logistic Regression

Consider the case where $\mathcal{Y} = \{+1, -1\}$.

$$
\begin{aligned}
p_{\mathbf{w}}(Y = +1 \mid x) &= \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, +1)}{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, +1) + \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, -1)} \\
&= \operatorname{logit}^{-1} \left(\mathbf{w} \cdot (\boldsymbol{\phi}(x, +1) - \boldsymbol{\phi}(x, -1))\right) \\
&\overset{\text{notation change}}{=} \operatorname{logit}^{-1} \left(\mathbf{w} \cdot \mathbf{f}(x)\right)
\end{aligned}
$$

- Should be familiar, if you know about logistic regression.
- When $\mathcal{Y} = \{1, 2, \ldots, k\}$, log-linear models are often called **multinomial logistic regression**.

Special Case: n-Gram Language Model

Consider an n-gram language model, where $\mathcal{X} = \mathcal{V}^{\mathsf{n}-1}$ and $\mathcal{Y} = \mathcal{V}$.
Let:

- $d = 1$
- $\phi_1(\boldsymbol{h}, v) = \log c(\boldsymbol{h}v)$
- $w_1 = 1$
- $Z(\boldsymbol{h}) = \sum_{v' \in \mathcal{V}} \exp \log c(\boldsymbol{h}v') = \sum_{v' \in \mathcal{V}} c(\boldsymbol{h}v') = c(\boldsymbol{h})$

Special Case: n-Gram Language Model

Consider an n-gram language model, where $\mathcal{X} = \mathcal{V}^{\mathsf{n}-1}$ and $\mathcal{Y} = \mathcal{V}$.
Let:

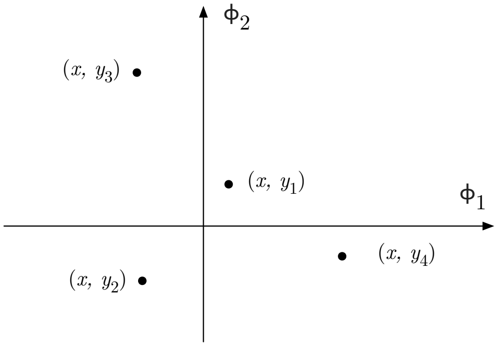
- $d = 1$
- $\phi_1(\boldsymbol{h}, v) = \log c(\boldsymbol{h}v)$
- $w_1 = 1$
- $Z(\boldsymbol{h}) = \sum_{v' \in \mathcal{V}} \exp \log c(\boldsymbol{h}v') = \sum_{v' \in \mathcal{V}} c(\boldsymbol{h}v') = c(\boldsymbol{h})$

Alternately:

- $d = |\mathcal{V}|^{\mathsf{n}}$
- $\phi_{\tilde{\boldsymbol{h}}, \tilde{v}}(\boldsymbol{h}, v) = \begin{cases} 1 & \text{if } \boldsymbol{h} = \tilde{\boldsymbol{h}} \wedge v = \tilde{v} \\ 0 & \text{otherwise} \end{cases}$
- $w_{\tilde{\boldsymbol{h}}, \tilde{v}} = \log \frac{c(\tilde{\boldsymbol{h}}\tilde{v})}{c(\tilde{\boldsymbol{h}})}$
- $Z(\boldsymbol{h}) = 1$

The Geometric View

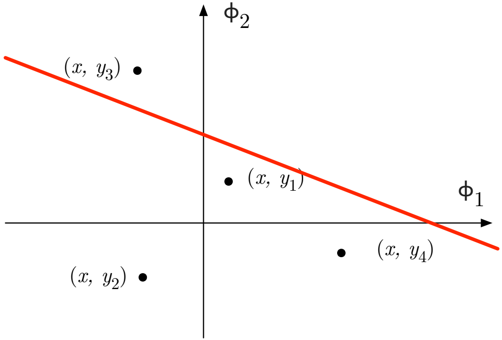
Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, ϕ_1 and ϕ_2.



As a simple example, let the two features be binary functions.

The Geometric View

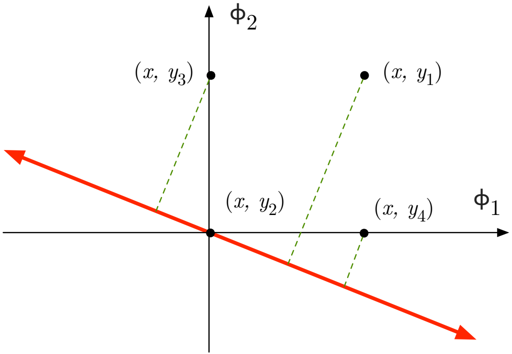
Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, ϕ_1 and ϕ_2.



$$\mathbf{w} \cdot \boldsymbol{\phi} = w_1\phi_1 + w_2\phi_2 = 0$$

The Geometric View

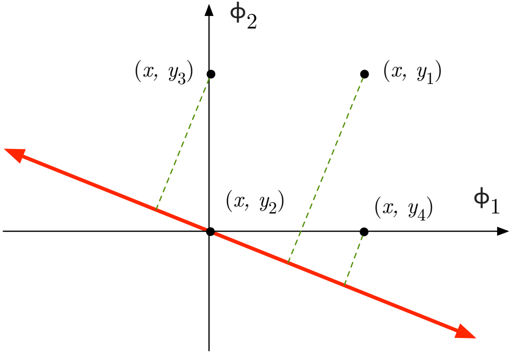
Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, ϕ_1 and ϕ_2.



$$\text{distance}(\mathbf{w} \cdot \boldsymbol{\phi} = 0, \boldsymbol{\phi}_0) = \frac{|\mathbf{w} \cdot \boldsymbol{\phi}_0|}{\|\mathbf{w}\|_2} \propto |\mathbf{w} \cdot \boldsymbol{\phi}_0|$$

The Geometric View

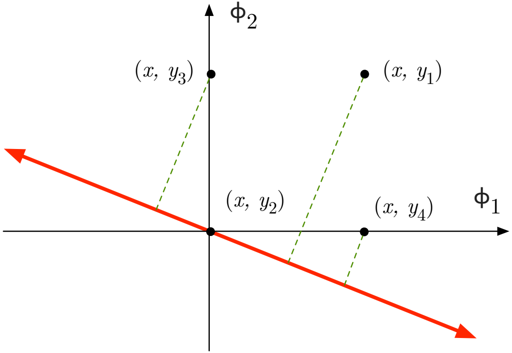
Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, ϕ_1 and ϕ_2.



$$\mathbf{w} \cdot \boldsymbol{\phi}(x, y_1) > \mathbf{w} \cdot \boldsymbol{\phi}(x, y_3) > \mathbf{w} \cdot \boldsymbol{\phi}(x, y_4) > 0 \geq \mathbf{w} \cdot \boldsymbol{\phi}(x, y_2)$$

The Geometric View

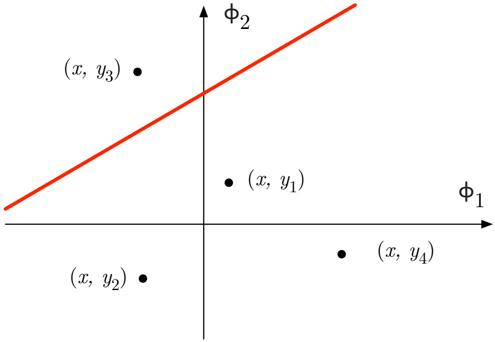
Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, ϕ_1 and ϕ_2.



$$p_{\mathbf{w}}(y_1 \mid x) > p_{\mathbf{w}}(y_3 \mid x) > p_{\mathbf{w}}(y_4 \mid x) > p_{\mathbf{w}}(y_2 \mid x)$$

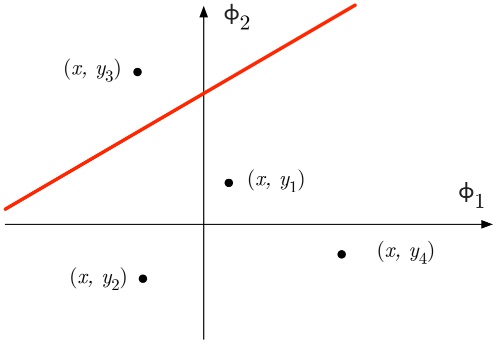
The Geometric View

Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, ϕ_1 and ϕ_2.



The Geometric View

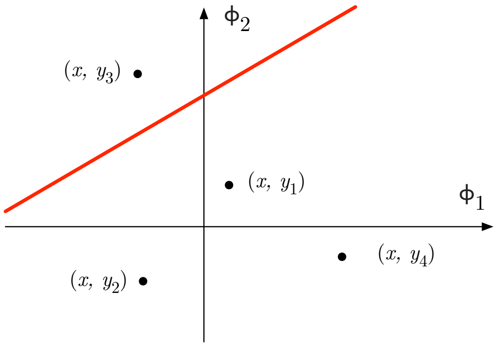
Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, ϕ_1 and ϕ_2.



$$p_{\mathbf{w}}(y_3 \mid x) > p_{\mathbf{w}}(y_1 \mid x) > p_{\mathbf{w}}(y_2 \mid x) > p_{\mathbf{w}}(y_4 \mid x)$$

The Geometric View

Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, ϕ_1 and ϕ_2.



Log-linear parameter estimation tries to choose $\mathbf{w}$ so that $p_{\mathbf{w}}(Y \mid x)$ matches the empirical distribution, $\frac{c(x,Y)}{c(x)}$.

Why Build Language Models This Way?

- Exploit **features** of histories for sharing of statistical strength and better smoothing (Lau et al., 1993)
- Condition the whole text on more interesting variables like the gender, age, or political affiliation of the author (Eisenstein et al., 2011)
- Interpretability!
  - Each feature ϕ_k controls a factor to the probability (e^{w_k}).
  - If $w_k < 0$ then ϕ_k makes the event less likely by a factor of $\frac{1}{e^{w_k}}$.
  - If $w_k > 0$ then ϕ_k makes the event more likely by a factor of e^{w_k}.
  - If $w_k = 0$ then ϕ_k has no effect.

Log-Linear n-Gram Models

$$
\begin{aligned}
p_{\mathbf{w}}(\boldsymbol{X} = \boldsymbol{x}) &= \prod_{j=1}^{\ell} p_{\mathbf{w}}(X_j = x_j \mid \boldsymbol{X}_{1:j-1} = \boldsymbol{x}_{1:j-1}) \\
&= \prod_{j=1}^{\ell} \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_{1:j-1}, x_j)}{Z_{\mathbf{w}}(\boldsymbol{x}_{1:j-1})} \\
&\overset{\text{assumption}}{=} \prod_{j-1}^{\ell} \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}_{j-\mathsf{n}+1:j-1}, x_j)}{Z_{\mathbf{w}}(\boldsymbol{x}_{j-\mathsf{n}+1:j-1})} \\
&= \prod_{j=1}^{\ell} \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_j, x_j)}{Z_{\mathbf{w}}(\boldsymbol{h}_j)}
\end{aligned}
$$

Example

The man who knew too

much
many
little
few
⋮
hippopotamus

What Features in $\phi(\boldsymbol{X}_{j-\mathsf{n}+1:j-1}, X_j)$?

What Features in $\phi(\boldsymbol{X}_{j-\mathsf{n}+1:j-1}, X_j)$?

- Traditional n-gram features: "$X_{j-1} = \text{the} \wedge X_j = \text{man}$"

What Features in $\phi(\boldsymbol{X}_{j-\mathsf{n}+1:j-1}, X_j)$?

- Traditional n-gram features: "$X_{j-1} = \text{the} \wedge X_j = \text{man}$"
- "Gappy" n-grams: $X_{j-2} = \text{the} \wedge X_j = \text{man}$"

What Features in $\phi(\boldsymbol{X}_{j-\mathsf{n}+1:j-1}, X_j)$?

- Traditional n-gram features: "$X_{j-1} =$ the $\wedge X_j =$ man"
- "Gappy" n-grams: $X_{j-2} =$ the $\wedge X_j =$ man"
- Spelling features: "X_j's first character is capitalized"

What Features in $\phi(\boldsymbol{X}_{j-\mathsf{n}+1:j-1}, X_j)$?

- Traditional n-gram features: "$X_{j-1} =$ the $\wedge X_j =$ man"
- "Gappy" n-grams: $X_{j-2} =$ the $\wedge X_j =$ man"
- Spelling features: "X_j's first character is capitalized"
- Class features: "X_j is a member of class 132"

What Features in $\phi(\boldsymbol{X}_{j-\mathsf{n}+1:j-1}, X_j)$?

- Traditional n-gram features: "$X_{j-1} =$ the $\wedge\, X_j =$ man"
- "Gappy" n-grams: $X_{j-2} =$ the $\wedge\, X_j =$ man"
- Spelling features: "X_j's first character is capitalized"
- Class features: "X_j is a member of class 132"
- Gazetteer features: "X_j is listed as a geographic place name"

What Features in $\phi(\boldsymbol{X}_{j-\mathsf{n}+1:j-1}, X_j)$?

- Traditional n-gram features: "$X_{j-1} = \text{the} \wedge X_j = \text{man}$"
- "Gappy" n-grams: $X_{j-2} = \text{the} \wedge X_j = \text{man}$"
- Spelling features: "X_j's first character is capitalized"
- Class features: "X_j is a member of class 132"
- Gazetteer features: "X_j is listed as a geographic place name"

You can define any features you want!

- Too many features, and your model will overfit ☹
- Too few (good) features, and your model will not learn ☹

What Features in $\phi(\boldsymbol{X}_{j-\mathsf{n}+1:j-1}, X_j)$?

- Traditional n-gram features: "$X_{j-1} =$ the $\wedge X_j =$ man"
- "Gappy" n-grams: $X_{j-2} =$ the $\wedge X_j =$ man"
- Spelling features: "X_j's first character is capitalized"
- Class features: "X_j is a member of class 132"
- Gazetteer features: "X_j is listed as a geographic place name"

You can define any features you want!

- Too many features, and your model will overfit ☹
  - "Feature selection" methods, e.g., ignoring features with very low counts, can help.
- Too few (good) features, and your model will not learn ☹

“Feature Engineering”

- Many advances in NLP (not just language modeling) have come from careful design of features.

“Feature Engineering”

- Many advances in NLP (not just language modeling) have come from careful design of features.
- Sometimes “feature engineering” is used pejoratively.

“Feature Engineering”

- Many advances in NLP (not just language modeling) have come from careful design of features.
- Sometimes “feature engineering” is used pejoratively.
  - Some people would rather not spend their time on it!

“Feature Engineering”

- Many advances in NLP (not just language modeling) have come from careful design of features.
- Sometimes “feature engineering” is used pejoratively.
  - Some people would rather not spend their time on it!
- There is some work on automatically inducing features (Della Pietra et al., 1997).

“Feature Engineering”

- Many advances in NLP (not just language modeling) have come from careful design of features.
- Sometimes “feature engineering” is used pejoratively.
  - Some people would rather not spend their time on it!
- There is some work on automatically inducing features (Della Pietra et al., 1997).
- More recent work in neural networks can be seen as *discovering* features (instead of engineering them).

“Feature Engineering”

- Many advances in NLP (not just language modeling) have come from careful design of features.
- Sometimes “feature engineering” is used pejoratively.
  - Some people would rather not spend their time on it!
- There is some work on automatically inducing features (Della Pietra et al., 1997).
- More recent work in neural networks can be seen as *discovering* features (instead of engineering them).
- But in much of NLP, there’s a strong preference for *interpretable* features.

How to Estimate $\mathbf{w}$?

	n-gram	log-linear n-gram
$p_{\boldsymbol{\theta}}(\boldsymbol{x}) =$	$\prod_{j=1}^{\ell} \theta_{x_j \mid \boldsymbol{h}_j}$	$\prod_{j-1}^{\ell} \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_j, x_j)}{Z_{\mathbf{w}}(\boldsymbol{h}_j)}$
Parameters:	$\theta_{v \mid \boldsymbol{h}}$ $\forall v \in \mathcal{V}, \boldsymbol{h} \in (\mathcal{V} \cup \{\bigcirc\})^{\mathsf{n}-1}$	w_k $\forall k \in \{1, \ldots, d\}$
MLE:	$\frac{c(\boldsymbol{h}v)}{c(\boldsymbol{h})}$	no closed form

MLE for $\mathbf{w}$

- Let training data consist of $\{(\boldsymbol{h}_i, x_i)\}_{i=1}^{N}$.

MLE for $\mathbf{w}$

- Let training data consist of $\{(\boldsymbol{h}_i, x_i)\}_{i=1}^N$.
- Maximum likelihood estimation is:

$$\max_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^N \log p_{\mathbf{w}}(x_i \mid \boldsymbol{h}_i)$$

$$= \max_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^N \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_i, x_i) - \log \underbrace{\sum_{v\in\mathcal{V}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_i, v)}_{Z_{\mathbf{w}}(\boldsymbol{h}_i)}$$

MLE for $\mathbf{w}$

- Let training data consist of $\{(\boldsymbol{h}_i, x_i)\}_{i=1}^{N}$.
- Maximum likelihood estimation is:

$$\max_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{N} \log p_{\mathbf{w}}(x_i \mid \boldsymbol{h}_i)$$

$$= \max_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{N} \mathbf{w}\cdot\boldsymbol{\phi}(\boldsymbol{h}_i, x_i) - \log \underbrace{\sum_{v\in\mathcal{V}} \exp \mathbf{w}\cdot\boldsymbol{\phi}(\boldsymbol{h}_i, v)}_{Z_{\mathbf{w}}(\boldsymbol{h}_i)}$$

- This is *concave* in $\mathbf{w}$.

MLE for $\mathbf{w}$

- Let training data consist of $\{(\boldsymbol{h}_i, x_i)\}_{i=1}^{N}$.
- Maximum likelihood estimation is:

$$
\begin{aligned}
&\max_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{N} \log p_{\mathbf{w}}(x_i \mid \boldsymbol{h}_i) \\
&= \max_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{N} \mathbf{w}\cdot\phi(\boldsymbol{h}_i, x_i) - \log \underbrace{\sum_{v\in\mathcal{V}} \exp \mathbf{w}\cdot\boldsymbol{\phi}(\boldsymbol{h}_i, v)}_{Z_{\mathbf{w}}(\boldsymbol{h}_i)}
\end{aligned}
$$

- This is *concave* in $\mathbf{w}$.
- $Z_{\mathbf{w}}(\boldsymbol{h}_i)$ involves a sum over V terms.

MLE for $\mathbf{w}$

- Let training data consist of $\{(\boldsymbol{h}_i, x_i)\}_{i=1}^{N}$.
- Maximum likelihood estimation is:

$$\begin{aligned}
&\max_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{N} \log p_{\mathbf{w}}(x_i \mid \boldsymbol{h}_i) \\
&= \max_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{N} \mathbf{w}\cdot\phi(\boldsymbol{h}_i, x_i) - \log \underbrace{\sum_{v\in\mathcal{V}} \exp \mathbf{w}\cdot\boldsymbol{\phi}(\boldsymbol{h}_i, v)}_{Z_{\mathbf{w}}(\boldsymbol{h}_i)}
\end{aligned}$$

- This is *concave* in $\mathbf{w}$.
- $Z_{\mathbf{w}}(\boldsymbol{h}_i)$ involves a sum over V terms.
  - Neat trick (Goodman, 2001): class-based model!

References I

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.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

Jacob Eisenstein, Amr Ahmed, and Eric P Xing. Sparse additive generative models of text. In *Proc. of ICML*, 2011.

Joshua Goodman. Classes for fast maximum entropy training. In *Proc. of ICASSP*, 2001.

Raymond Lau, Ronald Rosenfeld, and Salim Roukos. Trigger-based language models: A maximum entropy approach. In *Proc. of ICASSP*, 1993.

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