Natural Language Processing (CSE 517): Featurized Language Models

Noah Smith

University of Washington
nasmith@cs.washington.edu

January 11, 2016

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
- class-based: further decomposition
  $p(x_i \mid \mathsf{cl}(x_i)) \cdot p(\mathsf{cl}(x_i) \mid \boldsymbol{h}_i)$
  - previous $(\mathsf{n} - 1)$ symbols' *classes* predict class of x_i
  - class of x_i predicts x_i
- 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
- class-based: further decomposition $p(x_i \mid \mathsf{cl}(x_i)) \cdot p(\mathsf{cl}(x_i) \mid \boldsymbol{h}_i)$
  - previous $(\mathsf{n}-1)$ symbols' *classes* predict class of x_i
  - class of x_i predicts x_i
- 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
- class-based: further decomposition $p(x_i \mid \mathsf{cl}(x_i)) \cdot p(\mathsf{cl}(x_i) \mid \boldsymbol{h}_i)$
  - previous $(\mathsf{n}-1)$ symbols' *classes* predict class of x_i
  - class of x_i predicts x_i
- Probabilities are estimated from data.

Today: log-linear language models

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 (☺ for language modeling, $\mathcal{V}$)
- $\mathcal{X}$ is the set of contexts (☺ for n-gram language modeling, $\mathcal{V}^{\mathsf{n}-1}$)
- $\boldsymbol{\phi} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ 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

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

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

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

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

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

Breaking It Down

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

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

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

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

"Log-linear" comes from the fact that:

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

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 $\mathcal{Y} = \{+1, -1\}$.

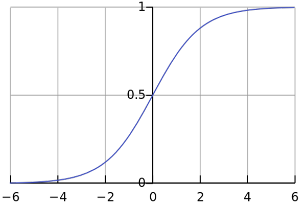
$$
\begin{aligned}
p_{\mathbf{w}}(Y = +1 \mid x) &= \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, +1)}{\exp \mathbf{w} \cdot \boldsymbol{\phi}(x, +1) + \exp \mathbf{w} \cdot \boldsymbol{\phi}(x, -1)} \\
&= \operatorname{logit}^{-1}\left(\mathbf{w} \cdot (\boldsymbol{\phi}(x, +1) - \boldsymbol{\phi}(x,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

Special Case: Logistic Regression

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

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

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

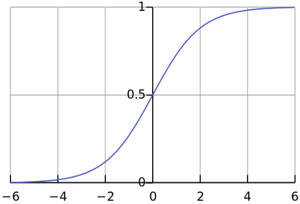


Special Case: Logistic Regression

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

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

- 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:

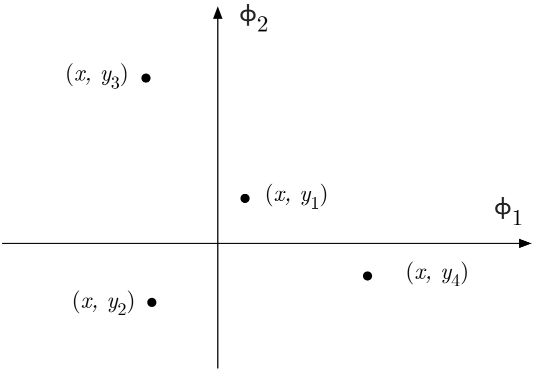
- $d = 1$
- $\phi_1(\boldsymbol{h}, v) = \log c(\boldsymbol{h}v)$
- $w_1 = 1$
- $Z(\boldsymbol{h}) = \sum_{v' \in \mathcal{V}} \exp \log c(\boldsymbol{h}v') = \sum_{v' \in \mathcal{V}} c(\boldsymbol{h}v') = c(\boldsymbol{h})$

Alternately:

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

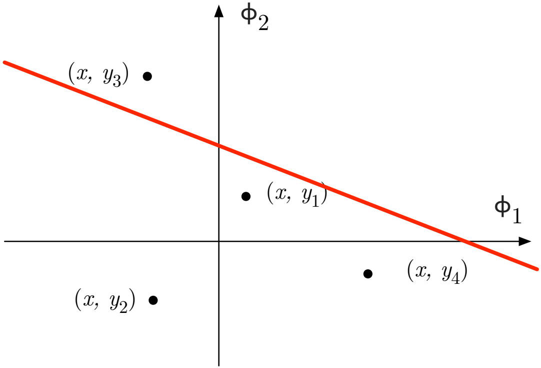
The Geometric View

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.



The Geometric View

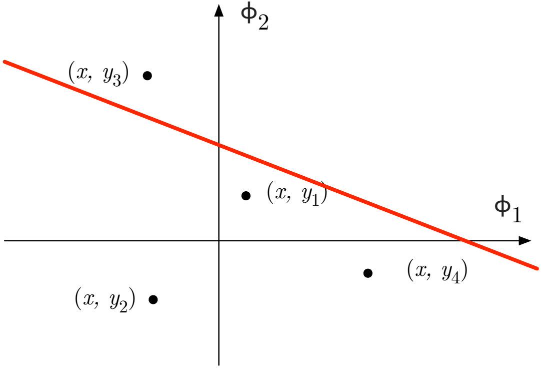
Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, ϕ_1 and ϕ_2.



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

The Geometric View

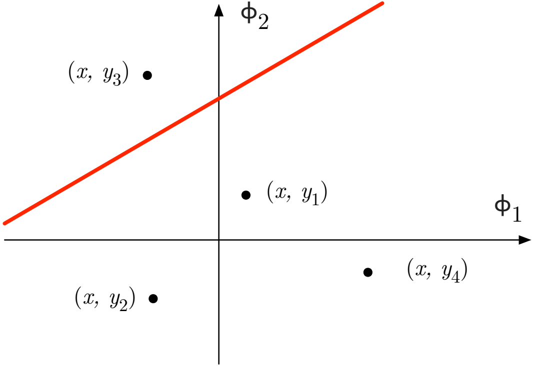
Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, ϕ_1 and ϕ_2.



$$p(y_3 \mid x) > p(y_1 \mid x) > p(y_4 \mid x) > p(y_2 \mid x)$$

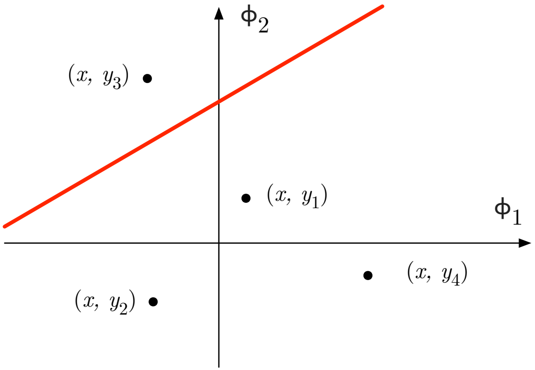
The Geometric View

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.



$$p(y_3 \mid x) > p(y_1 \mid x) > p(y_2 \mid x) > p(y_4 \mid x)$$

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 things (Eisenstein et al., 2011)
- Interpretability!
  - Each feature ϕ_k controls a factor to the probability (e^{w_k}); w_k is the *ceteris parebis* log-odds.
  - If $w_k < 0$ then ϕ_k makes the event less likely.
  - If $w_k > 0$ then ϕ_k makes the event more likely.
  - 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
$\vdots$
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 NLP, there’s a strong preference for *interpretable* features.

How to Estimate $\mathbf{w}$?

	n-gram	log-linear n-gram
$p_{\boldsymbol{\theta}}(\boldsymbol{x}) =$	$\displaystyle\prod_{j=1}^{\ell} \theta_{x_j \mid \boldsymbol{h}_j}$	$\displaystyle\prod_{j-1}^{\ell} \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_j, x_j)}{Z_{\mathbf{w}}(\boldsymbol{h}_j)}$
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:	$\dfrac{c(\boldsymbol{h}v)}{c(\boldsymbol{h})}$	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:

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

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:

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

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

- This is *concave* in $\mathbf{w}$.
- $Z_{\mathbf{w}}(\boldsymbol{h}_i)$ 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:

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

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

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

MLE for $\mathbf{w}$

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

MLE for $\mathbf{w}$

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

Hope/fear view: for each instance i,

- increase the score of the correct output x_i, $score(x_i) = \mathbf{w}\cdot\phi(\boldsymbol{h}_i, x_i)$
- decrease the "average" score overall, $\log \sum_{v\in\mathcal{V}} \exp score(v)$

MLE for $\mathbf{w}$

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

Gradient view:

$$\sum_{i=1}^{N} \boldsymbol{\phi}(\boldsymbol{h}_i, x_i) - \mathbb{E}_{p_{\mathbf{w}}(X \mid \boldsymbol{h}_i)}[\boldsymbol{\phi}(\boldsymbol{h}_i, X)]$$

Setting this to zero means getting model's expectations to match empirical expectations.

MLE for $\mathbf{w}$: Algorithms

- Batch methods (L-BFGS is popular)
- Stochastic gradient descent more common today, especially with special tricks for adapting the step size over time
- Many specialized methods (e.g., “iterative scaling”)

Avoiding Overfitting

Maximum likelihood estimation:

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

- If $\phi_j(\boldsymbol{h}, x)$ is (almost) always positive, we can always increase the objective (a little bit) by increasing w_j toward $+\infty$.

Avoiding Overfitting

Maximum likelihood estimation:

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

- If $\phi_j(\boldsymbol{h}, x)$ is (almost) always positive, we can always increase the objective (a little bit) by increasing w_j toward $+\infty$.

Standard solution is to add a regularization term:

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

where $\lambda > 0$ is a hyperparameter and $p = 2$ or 1.

ℓ_1 Regularization

This case warrants a little more discussion:

$$\max_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^{N} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_i, x_i) - \log \sum_{v \in \mathcal{V}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_i, v) - \lambda \|\mathbf{w}\|_1$$

Note that:

$$\|\mathbf{w}\|_1 = \sum_{j=1}^{d} |w_j|$$

- This results in **sparsity** (i.e., many $w_j = 0$).

ℓ_1 Regularization

This case warrants a little more discussion:

$$\max_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^{N} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_i, x_i) - \log \sum_{v \in \mathcal{V}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_i, v) - \lambda \|\mathbf{w}\|_1$$

Note that:

$$\|\mathbf{w}\|_1 = \sum_{j=1}^{d} |w_j|$$

- This results in **sparsity** (i.e., many $w_j = 0$).
  - Many have argued that this is a good thing (Tibshirani, 1996); it's a kind of feature selection.

ℓ_1 Regularization

This case warrants a little more discussion:

$$\max_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^{N} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_i, x_i) - \log \sum_{v \in \mathcal{V}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_i, v) - \lambda \|\mathbf{w}\|_1$$

Note that:

$$\|\mathbf{w}\|_1 = \sum_{j=1}^{d} |w_j|$$

- This results in **sparsity** (i.e., many $w_j = 0$).
  - Many have argued that this is a good thing (Tibshirani, 1996); it's a kind of feature selection.
  - Do not confuse it with data *sparseness* (a problem to be overcome)!

ℓ_1 Regularization

This case warrants a little more discussion:

$$\max_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^{N} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_i, x_i) - \log \sum_{v \in \mathcal{V}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_i, v) - \lambda \|\mathbf{w}\|_1$$

Note that:

$$\|\mathbf{w}\|_1 = \sum_{j=1}^{d} |w_j|$$

- This results in **sparsity** (i.e., many $w_j = 0$).
  - Many have argued that this is a good thing (Tibshirani, 1996); it's a kind of feature selection.
  - Do not confuse it with data *sparseness* (a problem to be overcome)!
- This is not differentiable at $w_j = 0$.

ℓ_1 Regularization

This case warrants a little more discussion:

$$\max_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^{N} \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_i, x_i) - \log \sum_{v \in \mathcal{V}} \exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{h}_i, v) - \lambda \|\mathbf{w}\|_1$$

Note that:

$$\|\mathbf{w}\|_1 = \sum_{j=1}^{d} |w_j|$$

- This results in **sparsity** (i.e., many $w_j = 0$).
  - Many have argued that this is a good thing (Tibshirani, 1996); it's a kind of feature selection.
  - Do not confuse it with data *sparseness* (a problem to be overcome)!
- This is not differentiable at $w_j = 0$.
- Optimization: special solutions for batch (e.g., Andrew and Gao, 2007) and stochastic (e.g., Langford et al., 2009) settings.

MLE for $\mathbf{w}$

If we had five more weeks, we'd study this problem more carefully!

Here's what you must remember:

- There is no closed form; you must use a numerical optimization algorithm.
- Log-linear models are powerful but expensive ($Z_{\mathbf{w}}(\boldsymbol{h}_i)$).
- Regularization is very important; we don't actually do MLE.
  - Just like for n-gram models! Only even more so, since log-linear models are even more expressive.

Maximum Entropy

Consider a distribution p over events in $\mathcal{X}$. The Shannon entropy (in bits) of p is defined as:

$$H(p) = -\sum_{x \in \mathcal{X}} p(X = x) \begin{cases} 0 & \text{if } p(X = x) = 0 \\ \log_2 p(X = x) & \text{otherwise} \end{cases}$$

This is a measure of "randomness"; entropy is zero when p is deterministic and $\log |\mathcal{X}|$ when p is uniform.

Maximum entropy principle: among distributions that fit the data, pick the one with the greatest entropy.

Maximum Entropy

If "fit the data" is taken to mean

$$\forall k \in \{1, \ldots, d\}, \mathbb{E}_p[\phi_k] = \tilde{\mathbb{E}}[\phi_k]$$

then the MLE of the log-linear family with features ϕ is the maximum entropy solution.

This is why log-linear models are sometimes called "maxent" models (e.g., Berger et al., 1996)

“Whole Sentence” Log-Linear Models

(Rosenfeld, 1994)

Instead of a log-linear model for each word-given-history, define a single log-linear model over event space $\mathcal{V}^\dagger$:

$$p_{\mathbf{w}}(\boldsymbol{x}) = \frac{\exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x})}{Z_{\mathbf{w}}}$$

- Any feature of the sentence could be included in this model!
- $Z_{\mathbf{w}}$ is deceptively simple-looking!

$$Z_{\mathbf{w}} = \sum_{\boldsymbol{x} \in \mathcal{V}^\dagger} \exp \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{x})$$

Readings and Reminders

Collins (2011) §2

References I

Galen Andrew and Jianfeng Gao. Scalable training of ℓ_1-regularized log-linear models. In *Proc. of ICML*, 2007.

Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

Michael Collins. Log-linear models, MEMMs, and CRFs, 2011. URL `http://www.cs.columbia.edu/~mcollins/crf.pdf`.

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.

John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. In *NIPS*, 2009.

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

Roni Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD thesis, Carnegie Mellon University, 1994.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.