Natural Language Processing (CSE 447/547M): Featurized Language Models

Noah Smith

University of Washington
nasmith@cs.washington.edu

January 14, 2019

Administrivia

How to ask questions outside lecture:

- Use Canvas!
- If you want confidentiality, or aren't yet registered, then use the instructors' mailing list (but note that TAs will all see it)

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.

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

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 the probabilities over all sequences sum to one
  - 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" 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.

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.

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 (☺ for n-gram language modeling, $\mathcal{V}^{\mathsf{n}} \rightarrow \mathbb{R}^d$)
- $\mathbf{w} \in \mathbb{R}^d$ 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

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

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

Breaking It Down

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

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

Breaking It Down

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

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

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

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

Breaking It Down

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

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

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

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

"Log-linear" comes from the fact that:

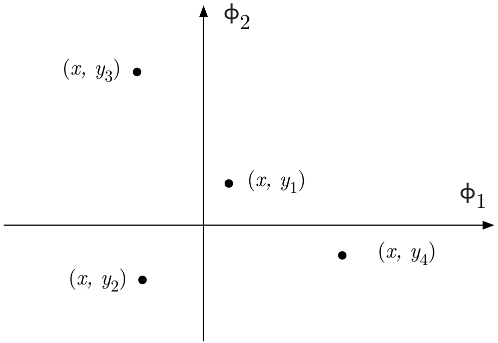
$$\log p_{\mathbf{w}}(Y = y \mid X = x) = \mathbf{w} \cdot \boldsymbol{\phi}(x, y) - \underbrace{\log Z_{\mathbf{w}}(x)}_{\text{constant in } y}$$

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

The Geometric View

(We skipped this in lecture.)

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.

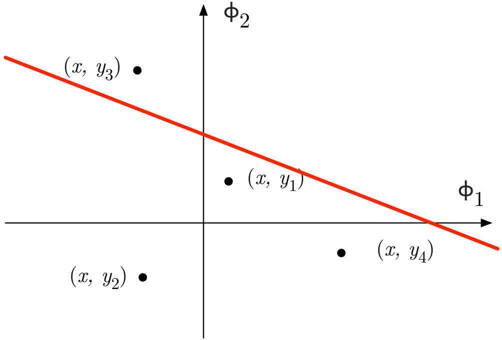


As a simple example, let the two features be binary functions.

The Geometric View

(We skipped this in lecture.)

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.

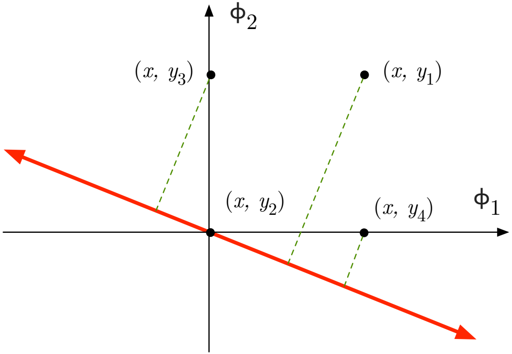


$$\mathbf{w} \cdot \boldsymbol{\phi} = w_1\phi_1 + w_2\phi_2 = 0$$

The Geometric View

(We skipped this in lecture.)

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.

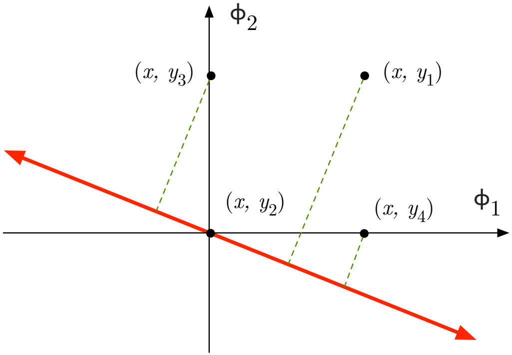


$$\text{distance}(\mathbf{w} \cdot \boldsymbol{\phi} = 0, \boldsymbol{\phi}) = \frac{|\mathbf{w} \cdot \boldsymbol{\phi}|}{\|\mathbf{w}\|_2} \propto |\mathbf{w} \cdot \boldsymbol{\phi}|$$

The Geometric View

(We skipped this in lecture.)

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.

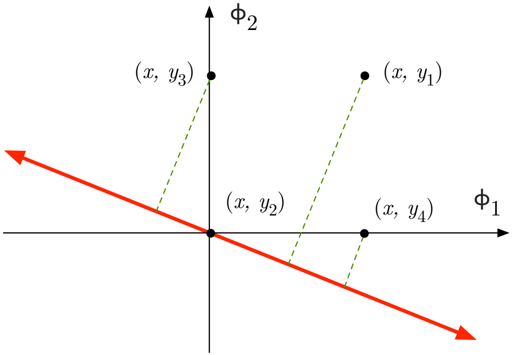


$$\mathbf{w} \cdot \boldsymbol{\phi}(x, y_1) > \mathbf{w} \cdot \boldsymbol{\phi}(x, y_3) > \mathbf{w} \cdot \boldsymbol{\phi}(x, y_4) > 0 \geq \mathbf{w} \cdot \boldsymbol{\phi}(x, y_2)$$

The Geometric View

(We skipped this in lecture.)

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.

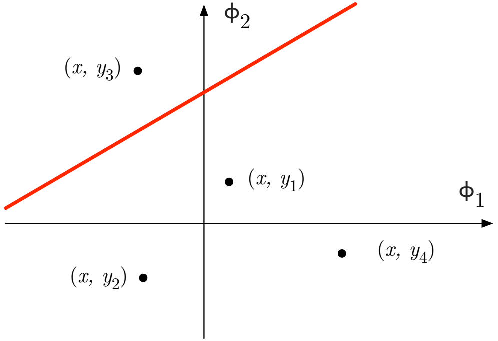


$$p_{\mathbf{w}}(y_1 \mid x) > p_{\mathbf{w}}(y_3 \mid x) > p_{\mathbf{w}}(y_4 \mid x) > p_{\mathbf{w}}(y_2 \mid x)$$

The Geometric View

(We skipped this in lecture.)

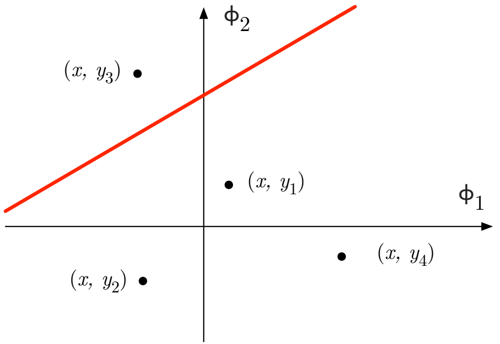
Suppose we have instance x, $\mathcal{Y} = \{y_1, y_2, y_3, y_4\}$, and there are only two features, ϕ_1 and ϕ_2.



The Geometric View

(We skipped this in lecture.)

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.

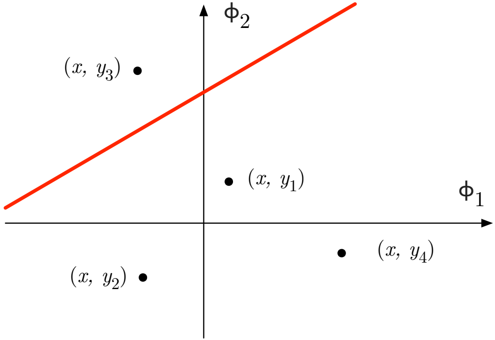


$$p_{\mathbf{w}}(y_3 \mid x) > p_{\mathbf{w}}(y_1 \mid x) > p_{\mathbf{w}}(y_2 \mid x) > p_{\mathbf{w}}(y_4 \mid x)$$

The Geometric View

(We skipped this in lecture.)

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

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"

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:

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

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

MLE for $\mathbf{w}$

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

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

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

- 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:

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

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

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

What follows are some slides we didn't cover in lecture. We did talk about stochastic gradient ascent/descent.

MLE for $\mathbf{w}$

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

MLE for $\mathbf{w}$

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

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 "softened max" score overall, $\log \sum_{v\in\mathcal{V}} \exp score(v)$

MLE for $\mathbf{w}$

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

Gradient view:

$$\nabla_{\mathbf{w}} f_i = \underbrace{\boldsymbol{\phi}(\boldsymbol{h}_i, x_i)}_{\text{observed features}} - \underbrace{\sum_{v\in\mathcal{V}} p_{\mathbf{w}}(v \mid \boldsymbol{h}_i)\cdot\boldsymbol{\phi}(\boldsymbol{h}_i, v)}_{\text{expected features}}$$

$$\nabla_{\mathbf{w}} F = \sum_{i=1}^{N}\left(\boldsymbol{\phi}(\boldsymbol{h}_i, x_i) - \sum_{v\in\mathcal{V}} p_{\mathbf{w}}(v \mid \boldsymbol{h}_i)\cdot\boldsymbol{\phi}(\boldsymbol{h}_i, v)\right)$$

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")

Stochastic Gradient Descent

Goal: minimize $\sum_{i=1}^{N} f_i(\mathbf{w})$ with respect to $\mathbf{w}$.

Input: initial value $\mathbf{w}$, number of epochs T, learning rate α

For $t \in \{1, \ldots, T\}$:

- Choose a random permutation π of $\{1, \ldots, N\}$.
- For $i \in \{1, \ldots, N\}$:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \nabla_{\mathbf{w}} f_{\pi(i)}$$

Output: $\mathbf{w}$

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.

MLE for $\mathbf{w}$

If we had more time, 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 like stochastic gradient descent.
- 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.

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.

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.

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.

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.