Statistical Learning Models for Text and Graph Data
Featurized Language Models

Yangqiu Song

Hong Kong University of Science and Technology
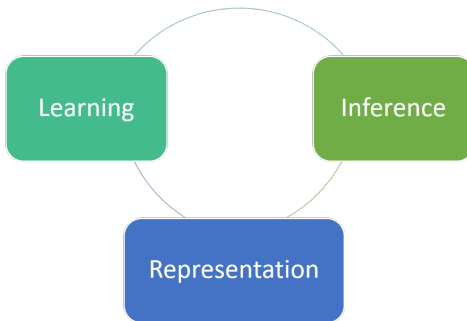
yqsong@cse.ust.hk

September 13, 2019

∗Contents are based on materials created by Noah Smith, Dan Klein, and Chris Manning

Reference Content

- Noah Smith. CSE 517: Natural Language Processing
  https://courses.cs.washington.edu/courses/cse517/16wi/
- Dan Klein. CS 288: Statistical Natural Language Processing.
  https://people.eecs.berkeley.edu/~klein/cs288/sp10/
- Chris Manning. CS 224N/Ling 237. Natural Language Processing.
  https://web.stanford.edu/class/cs224n/

Course Organization

- Representation: language models, word embeddings, topic models, knowledge graphs
- Learning: supervised learning, semi-supervised learning, distant supervision, indirect supervision, sequence models, deep learning, optimization techniques
- Inference: constraint modeling, joint inference, search algorithms

Overview

Quick Review

- A language model is a probability distribution over $\mathcal{V}$
- Typically P decomposes into probabilities $P(x_i|\mathbf{h}_i)$. For n-gram language models, to reduce notation confusion, we set:
    - x_i: w_i
    - $\mathbf{h}_i = (w_{i-1}, \ldots, w_{i-n+1})^\top$
- 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).
- Central idea today: teach histories and words how to share.

Overview

Log-Linear Models: Definitions

- We define a conditional log-linear model $P(Y|X)$ 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, $\mathcal{V}^{n-1}$)
    - $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ is a feature vector function
    - $\mathbf{w} \in \mathbb{R}^d$ are the model parameters

$$P_{\mathbf{w}}(Y = y | X = x) = \frac{\exp(\mathbf{w}^\top \phi(x, y))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}^\top \phi(x, y'))}$$

* $P_{\mathbf{w}}(Y = y | X = x) \triangleq P(Y = y | X = x, \mathbf{w})$

- We can re-parameterize an n-gram language model based on $\mathbf{w}$

Breaking It Down

$$P_{\mathbf{w}}(Y = y | X = x) = \frac{\exp(\mathbf{w}^\top \phi(x, y))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}^\top \phi(x, y'))}$$

- linear score $\mathbf{w}^\top \phi(x, y)$
- nonnegative $\exp(\mathbf{w}^\top \phi(x, y))$
- normalizer $\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}^\top \phi(x, y')) \triangleq Z_{\mathbf{w}}(x)$
- "Log-linear" comes from the fact that:

$$\log P_{\mathbf{w}}(Y = y | X = x) = \mathbf{w}^\top \phi(x, y) - \underbrace{\log Z_{\mathbf{w}}(x)}_{constant\ in\ y}$$
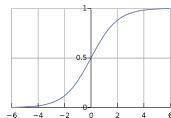
- This is an instance of the family of generalized linear models

Special Case: Logistic Regression

- Consider the case where $Y \in \{+1, -1\}$

$$
\begin{aligned}
P_{\mathbf{w}}(Y = +1 | X = x) &= \frac{\exp(\mathbf{w}^\top \phi(x, +1))}{\exp(\mathbf{w}^\top \phi(x, +1)) + \exp(\mathbf{w}^\top \phi(x, -1))} \\
&= \frac{1}{1 + \exp(\mathbf{w}^\top (\phi(x, -1) - \phi(x, +1)))} \\
&= \sigma(\mathbf{w}^\top \phi(x, +1) - \phi(x, -1)) \\
&\overset{\text{notation change}}{=} \sigma(y \mathbf{w}^\top \mathbf{f}(x))
\end{aligned}
$$

  where $\sigma(t) = \frac{1}{1 + e^{-t}}$ is logistic function

- Should be familiar, if you know about logistic regression (will come back to this later)

- When $Y \in \{1, 2, \ldots, K\}$, log-linear models are often called multinomial logistic regression (softmax function)

Special Case: N-gram Language Model

$$P_{\mathbf{w}}(Y = y | X = x) = \frac{\exp(\mathbf{w}^{\top}\phi(x, y))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}^{\top}\phi(x, y'))}$$

- Consider an n-gram language model, where $X = \mathcal{V}^{n-1}$ and $Y = \mathcal{V}$. Let $\mathbf{h} = \{w_{i-1}, \ldots, w_{i-n+1}\}$, $x = w_i$:
  - $d = 1$
  - $\phi_1(\mathbf{h}, v) = \log c(\mathbf{h}, x)$
  - $\mathbf{w}_1 = 1$
  - $Z(\mathbf{h}) = \sum_{x' \in \mathcal{V}} \exp \log c(\mathbf{h}, x') = \sum_{x' \in \mathcal{V}} c(\mathbf{h}, x') = c(\mathbf{h})$
- Alternatively, we enumerate all possible n-grams as feature indicators and set parameter $\mathbf{w}$ as the counts:
  - $d = |\mathcal{V}|^n$
  - $\phi_{\tilde{\mathbf{h}}, \tilde{x}}(\mathbf{h}, x) = \begin{cases} 1 & \text{if } \mathbf{h} = \tilde{\mathbf{h}} \wedge v = \tilde{x} \\ 0 & \text{otherwise} \end{cases}$
  - $\mathbf{w}_{\tilde{\mathbf{h}}, \tilde{x}} = \log \frac{c(\tilde{\mathbf{h}}, \tilde{x})}{c(\tilde{\mathbf{h}})}$
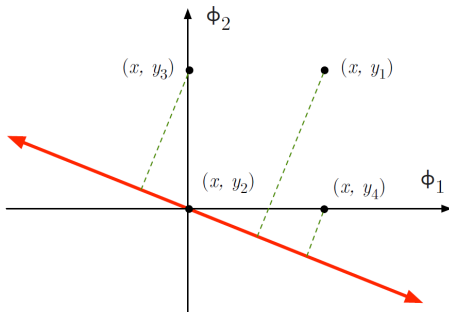 - $Z(\mathbf{h}) = 1$

The Geometric View

- Suppose we have instance x, $Y \in \{y_1, y_2, y_3, y_4\}$, and there are only two features, ϕ^1 and ϕ^2.
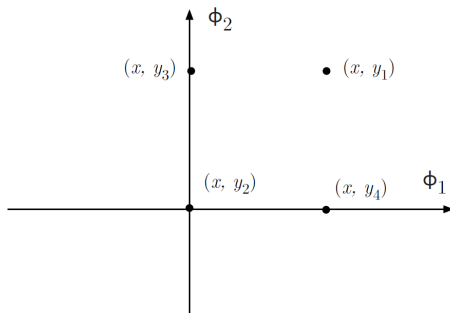
As a simple example, let the two features be binary functions.
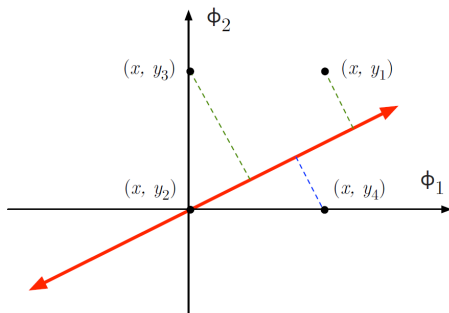
The Geometric View

- Suppose we have instance x, $Y \in \{y_1, y_2, y_3, y_4\}$, and there are only two features, ϕ^1 and ϕ^2.

- $\mathbf{w}^\top \phi = w_1 \phi_1 + w_2 \phi_2 = 0$
- $distance(\mathbf{w}^\top \phi = 0, \phi_0) = \frac{\mathbf{w}^\top \phi_0}{\|\mathbf{w}\|_2} \propto \mathbf{w}^\top \phi_0$
- $\mathbf{w}^\top \phi(x, y_1) > \mathbf{w}^\top \phi(x, y_3) > \mathbf{w}^\top \phi(x, y_4) > \mathbf{w}^\top \phi(x, y_2)$
- $P(y_1|x) > P(y_3|x) > P(y_4|x) > P(y_2|x)$

The Geometric View

- Suppose we have instance x, $Y \in \{y_1, y_2, y_3, y_4\}$, and there are only two features, ϕ^1 and ϕ^2.

- $P(y_3|x) > P(y_1|x) > P(y_2|x) > P(y_4|x)$

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

$$P_{\mathbf{w}}(Y = y | X = x) = \frac{\exp(\mathbf{w}^\top \phi(x, y))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}^\top \phi(x, y'))}$$

- Consider an n-gram language model, where $X = \mathcal{V}^{n-1}$ and $Y = \mathcal{V}$. Let $\mathbf{h}_i = \{w_{i-1}, \ldots, w_{i-n+1}\}$, $x_i = w_i$:

$$P_{\mathbf{w}}(\mathcal{W}) = \prod_{i=1}^{N} P(w_i | w_{i-1}, \ldots, w_{i-n+1})$$

$$\overset{\text{parameterization}}{=} \prod_{i=1}^{N} \frac{\exp\left(\mathbf{w}^\top \phi(\mathbf{h}_i, x_i)\right)}{Z_{\mathbf{w}}(\mathbf{h}_i)}$$

- What features are there used in $\phi(\mathbf{h}_i, x_i)$ more than traditional n-gram language models?

What features in $\phi(\mathbf{h}_i, x_i)$?

Example

	Saturday
	Sunday
	Monday
I visited Central last _____	*month*
	...
	pizza

- Traditional n-gram features: $w_{i-1} = last \wedge w_i = Saturday$
- "Gappy" n-gram features: $w_{i-2} = Central \wedge w_i = Saturday$
- Spelling features: w_i's first character is capitalized
- Class features: w_i is a member of class 132
- Gazetteer features: w_i is listed as a geographic place name, date/time, person name, organization name, etc.

What features in $\phi(\mathbf{h}_i, x_i)$?

- 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
- Sometimes "feature engineering" is used pejoratively
  - Some people would rather not spend their time on it!
- There is some work on automatically inducing features (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 **w**?

	n-gram	*log-linear n-gram*
	$P_{\boldsymbol{\theta}}(\mathcal{W}) = \prod_{i=1}^{N} \theta_{x_i|\mathbf{h}_i}$	$P_{\mathbf{w}}(\mathcal{W}) = \prod_{i=1}^{N} \frac{\exp\left(\mathbf{w}^{\top}\phi(\mathbf{h}_i, x_i)\right)}{Z_{\mathbf{w}}(\mathbf{h}_i)}$
Parameters :	$\theta_{x|\mathbf{h}}$ $\forall x \in \mathcal{V}, \mathbf{h} \in (\mathcal{V} \cup \emptyset)^{n-1}$	w_k $k \in \{1, \ldots, d\}$
MLE :	$\frac{c(\mathbf{h}, x)}{c(\mathbf{h})}$	*no closed form*

MLE for **w**

- Let training data consist of $\{(h_i; x_i)\}_{i=1}^{N}$
- Maximum likelihood estimation is:

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

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

- This is concave in **w**
  - Convexity: http://qwone.com/~jason/writing/convexLR.pdf
- $Z_{\mathbf{w}}(\mathbf{h}_i)$ involves a sum over V terms.

MLE for **w**

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

- Hope/fear view: for each instance i,
  - increase the score of the correct output x_i: $score(x_i) = \mathbf{w}^\top \phi(x_i, \mathbf{h}_i)$
  - decrease the "average" score overall: $\log \sum_{v \in \mathcal{V}} \exp(score(v))$
- Gradient view:

$$\nabla_{\mathbf{w}} \mathcal{L} = \sum_{i=1}^{N} [\phi(x_i, \mathbf{h}_i) - \sum_{v \in \mathcal{V}} \frac{\exp(\mathbf{w}^\top \phi(\mathbf{h}_i, v))}{\sum_{v' \in \mathcal{V}} \exp(\mathbf{w}^\top \phi(\mathbf{h}_i, v'))} \phi(\mathbf{h}_i, v)]$$

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

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

MLE for **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")

- Will come back to this topic later

Stochastic Gradient Ascent

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

- Goal: maximize $\mathcal{L}$ with respect to $\mathbf{w}$
- Input: initial value $\mathbf{w}$, number of epochs T, learning rate α
- For $t = 1, \ldots, T$
    - Choose a random permutation π of $\{1, \ldots, N\}$
    - For $i = 1, \ldots, N$

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

- Output: $\mathbf{w}$

Avoiding Overfitting

- Maximum likelihood estimation:

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

- If $\phi(x_i, \mathbf{h}_i)$ is (almost) always positive, we can always increase the objective (a little bit) by increasing w_k toward $+\infty$

- Standard solution is to add a regularization term:

$$\max_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^{N} [\mathbf{w}^\top \phi(x_i, \mathbf{h}_i) - \log Z_{\mathbf{w}}(\mathbf{h}_i)] - \lambda \|\mathbf{w}\|_p^p$$

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

ℓ_1 Regularization

- This case warrants a little more discussion:

$$\max_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^{N} [\mathbf{w}^\top \phi(x_i, \mathbf{h}_i) - \log Z_{\mathbf{w}}(\mathbf{h}_i)] - \lambda \|\mathbf{w}\|_1$$

- This results in sparsity (i.e., many $w_k = 0$).
  - 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 $w_k = 0$
- Optimization: special solutions for batch (e.g., Andrew and Gao (2007)) and stochastic (e.g., Langford et al. (2009)) settings

MLE for **w**

- We will come back to gradient based methods later
- Notes so far:
  - There is no closed form; you must use a numerical optimization algorithm
  - Log-linear models are powerful but expensive ($Z_{\mathbf{w}}(\mathbf{h}_i)$)
  - 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

Further Reading

- Berger et al. (1996).A Maximum Entropy Approach to Natural Language Processing.
- Collins (2011). Course notes for COMS w4705: Log-linear models, MEMMs, and CRFs, 2011. `http://www.cs.columbia.edu/~mcollins/crf.pdf`
- Smith (2004). Log-linear models, 2004. `https://homes.cs.washington.edu/~nasmith/papers/smith.tut04.pdf`

References I

Andrew, G. and Gao, J. (2007). Scalable training of l^1-regularized log-linear models. In *ICML*, pages 33–40.

Berger, A. L., Pietra, S. D., and Pietra, V. J. D. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Collins, M. (2011). Log-linear models, MEMMs, and CRFs. Technical report, Columbia University.

Eisenstein, J., Ahmed, A., and Xing, E. P. (2011). Sparse additive generative models of text. In *ICML*, pages 1041–1048.

Langford, J., Li, L., and Zhang, T. (2009). Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:777–801.

Lau, R., Rosenfeld, R., and Roukos, S. (1993). Trigger-based language models: A maximum entropy approach. In *ICASSP*, pages 45–48.

Pietra, S. D., Pietra, V. J. D., and Lafferty, J. D. (1997). Inducing features of random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(4):380–393.

Smith, N. A. (2004). Log-linear models. Technical report, University of Washington.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.