Learning in log-linear language models

CS 585, Fall 2019
Introduction to Natural Language Processing
http://people.cs.umass.edu/~miyyer/cs585/

Mohit Iyyer
College of Information and Computer Sciences
University of Massachusetts Amherst

[some slides adapted from Richard Socher]

stuff from last class….

- group assignments due by this Thursday Sep 19th at the end of the day, otherwise you'll be randomly assigned!
- HW1 bug fixed, recopy the notebook for those who already started!
- more readings that are research papers? ok
- talk about state-of-the-art models? later
- code libraries for project?

where we left off:

so we have some input text from which we have computed feature vector **x**.

$\mathbf{x}$(…Hence, in any statistical) = 10010010110101

$$\phi_0 = \begin{cases} 1, & \text{if } w_{i-1} = \text{statistical}, \\ 0, & \text{otherwise}. \end{cases}$$

$$\phi_1 = \begin{cases} 1, & \text{if } w_{i-1} = \text{animal}, \\ 0, & \text{otherwise}. \end{cases}$$

$$\phi_2 = \begin{cases} 1, & \text{if pos}(w_{i-1}) = \text{adj}, \\ 0, & \text{otherwise}. \end{cases}$$

given features **x**, how do we predict the next word *y*?

$$s = Wx + b$$

score vector $s \in \mathbb{R}^{|V|}$

features $x \in \mathbb{R}^{m}$

weight matrix $W \in \mathbb{R}^{|V| \times m}$

each row of W contains weights for a (word *y*, **x**) pair how do we obtain probabilities?

$$s = Wx + b$$

score vector $s \in \mathbb{R}^{|V|}$

features $x \in \mathbb{R}^{m}$

weight matrix $W \in \mathbb{R}^{|V| \times m}$

$$p_i = \frac{e^{s_i}}{\sum_j e^{s_j}}; p = \mathrm{softmax}(s)$$

colab demo!

what do we have left?

- how do we find the optimal values of **W** and **b** for our language modeling problem?
- gradient descent! this involves computing:
  1. a *loss function*, which tells us how good the current values of **W** and **b** are on our training data
  2. the partial derivatives $\frac{\partial L}{\partial W}$ and $\frac{\partial L}{\partial b}$

what do we have left?

- how do we find the optimal values of **W** and **b** for our language modeling problem?
- gradient descent! this involves computing:
  1. a *loss function*, which tells us how good the current values of **W** and **b** are on our training data
  2. the partial derivatives $\frac{\partial L}{\partial W}$ and $\frac{\partial L}{\partial b}$

kinda like we did in HW0!

first, an aside: what is the bias **b**?

- Let's say we have a feature that is always set to 1 regardless of what the input text is.
- This is clearly not an informative feature. However, let's say it was the only one I had…

first, an aside: what is the bias **b**?

- Let's say we have a feature that is always set to 1 regardless of what the input text is.
- This is clearly not an informative feature. However, let's say it was the only one I had…

first, how many weights do I need to learn for this feature?

first, an aside: what is the bias **b**?

- Let's say we have a feature that is always set to 1 regardless of what the input text is.
- This is clearly not an informative feature. However, let's say it was the only one I had…

first, how many weights do I need to learn for this feature?

okay… what is the best set of weights for it?

Training with softmax and cross-entropy error

- For each training example {x,y}, our objective is to maximize the probability of the correct class y

- Hence, we minimize the negative log probability of that class:

$$L = -\log p(y \mid x, W) = -\log\left(\frac{e^{W_y x}}{\sum_{y' \in V} e^{W_{y'} x}}\right)$$

Training with softmax and cross-entropy error

- For each training example {x,y}, our objective is to maximize the probability of the correct class y

- Hence, we minimize the negative log probability of that class:

$$L = -\log p(y \mid x, W) = -\log\left(\frac{e^{W_y x}}{\sum_{y' \in V} e^{W_{y'} x}}\right)$$

why not just maximize the log probability?

Background: Why "Cross entropy" error

- Assuming a ground truth (or gold or target) probability distribution that is 1 at the right class and 0 everywhere else: p = [0,…,0,1,0,…0] and our computed probability is q, then the cross entropy is:

$$H(p,q) = -\sum_{w \in V} p(w) \log q(w)$$

- **Because of one-hot p, the only term left is the negative log probability of the true class**

let's say I also have the derivatives

$$\frac{\partial L}{\partial W} \qquad \frac{\partial L}{\partial b}$$

- the partial derivatives tell us how the loss changes given a change in the corresponding parameter
- we can thus take steps in the *negative* direction of the gradient to *minimize* the loss function

draw on paper

derivation on paper