S&DS 355 / 555

Introductory Machine Learning

Neural Language Models

Tuesday, November 5

Yale

Recall: Language models

- A language model is a way of assigning a probability to any sequence of words (or string of text)

$$p(w_1, \ldots, w_n)$$

Recall: Language models

- A language model is a way of assigning a probability to any sequence of words (or string of text)

$$p(w_1, \ldots, w_n)$$

- By the basic rules of conditional probability we can factor this as

$$p(w_1, \ldots, w_n) = p(w_1)p(w_2 \mid w_1) \ldots p(w_n \mid w_1, \ldots, w_{n-1})$$

Recall: Language models

- A language model is a way of assigning a probability to any sequence of words (or string of text)

$$p(w_1, \ldots, w_n)$$

- By the basic rules of conditional probability we can factor this as

$$p(w_1, \ldots, w_n) = p(w_1)p(w_2 \mid w_1) \ldots p(w_n \mid w_1, \ldots, w_{n-1})$$

- The number of *histories* grows as $|V|^{n-1}$. Number of free parameters in model is $(|V| - 1)|V|^{n-1}$.

Recall: Language models

- A language model is a way of assigning a probability to any sequence of words (or string of text)

$$p(w_1, \dots, w_n)$$

- By the basic rules of conditional probability we can factor this as

$$p(w_1, \dots, w_n) = p(w_1)p(w_2 \mid w_1) \dots p(w_n \mid w_1, \dots, w_{n-1})$$

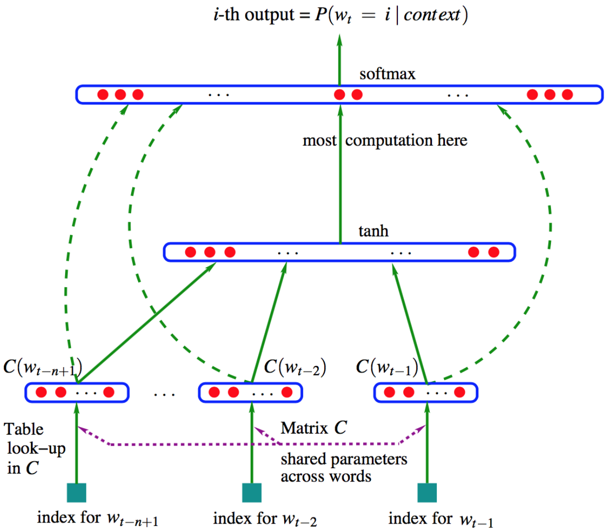
- The number of *histories* grows as $|V|^{n-1}$. Number of free parameters in model is $(|V| - 1)|V|^{n-1}$.
- We discussed some ways of reducing the number of parameters

Neural LM: Idea

- Associate each word in vocabulary with a feature vector $C(w) \in \mathbb{R}^d$
- Express probabilities in terms of those vectors
- Form a big logistic regression to predict the next word. Can introduce some nonlinearites.
- Simultaneously learn vectors (word representations) and weightings (model parameters), using SGD

Bengio et al. “A neural probabilistic language model,” Journal of Machine Learning Research (2003).

Neural LM architecture



Neural LM: Simplified

Suppose

$$p(\overbrace{w_t}^{\text{word at time } t} \mid \overbrace{w_{t-1}, \ldots, w_{t-n+1}}^{\text{previous words in time}}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

with

$$y = b + Ax$$
$$x = (C(w_{t-1}), C(w_{t-2}), \ldots, C(w_{t-n+1}))$$

Can be viewed as simply multinomial logistic regression.

But the key property is that the word representations $C(w) \in \mathbb{R}^d$ are *learned* as part of the model.

Neural LM: Simplified further!

Suppose

bigram model

$$p(w_t \mid w_{t-1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

with

Activation

$$y = b + Ax$$

$$x = C(w_{t-1})$$

- very similar to multinomial logistic regression

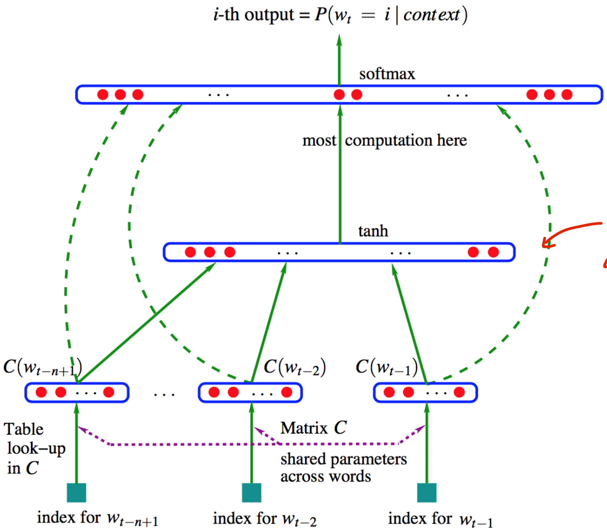
so: $y_w = b_w + A_w^T C(w_{t-1})$

bias activation x embedding vector

Key property is that the word representations $C(w) \in \mathbb{R}^d$ are *learned* as part of the model.

This is the essence of the model. Note that we get two "embedding" vectors: A_w and $C(w)$.

More general neural LM architecture



then, add a non-linearity

Neural LM parameterization

Language model parameterized by

$$p(w_t \mid w_{t-1}, \ldots, w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

(aka "softmax" of y) where

hyperbolic tangent func

x: embedding vectors

$$y = b + Ax + U \tanh(d + Hx) \in \mathbb{R}^V$$

$$x = (C(w_{t-1}), C(w_{t-2}), \ldots, C(w_{t-n+1}))$$

Model parameters b, A, U, d, H, and $C(i)$ are learned using stochastic gradient descent over the log-probability under this model.

Bigram Linear

$$Y_{w_t} = b_{w_t} + A_{w_t}^T C(w_{t-1}) \quad \Big\} \text{ logistic regression}$$

$$+ \; U_w^T \tanh(d + HC(w_{t-1})) \quad \Big\} \text{ extra non-linearity}$$

Neural LM parameterization

Language model parameterized by

$$p(w_t \mid w_{t-1}, \ldots, w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

(aka "softmax" of y) where

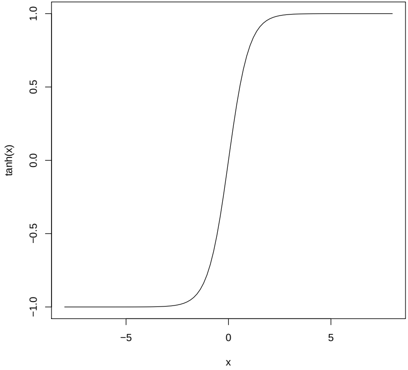
$$\begin{aligned} y_w &= b_w + A_w^T x + U_w^T \tanh(d + Hx) \in \mathbb{R} \\ x &= (C(w_{t-1}), C(w_{t-2}), \ldots, C(w_{t-n+1})) \end{aligned}$$

Nonlinearity

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
$$= 2\sigma(2x) - 1$$

closely related to the logistic function but shifted down

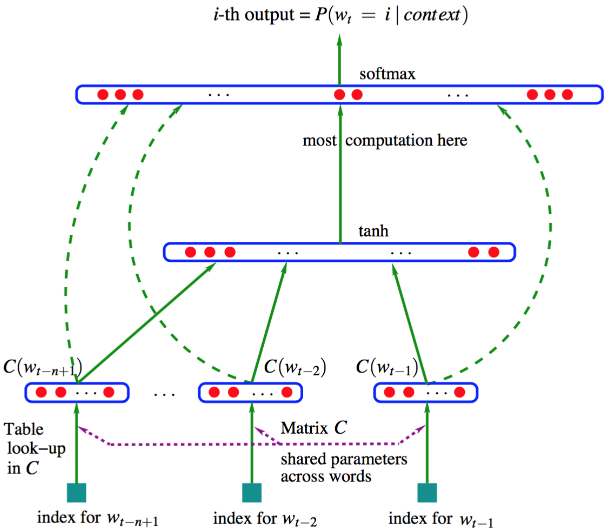
$$\sigma(x) = \frac{e^x}{1 + e^x}$$



Adds a nonparametric/nonlinear aspect to the model. Computational advantages of this particular form.

Lecun et al., "Efficient backprop"

Neural LM architecture



SGD training

Model parameters

$$\underbrace{b, A, U, d, H, \{C(w)\}}_{\theta}$$

Stochastic gradient descent

$$\theta \longleftarrow \theta + \eta \frac{\partial p_\theta \left(w_t \mid w_{t-1}, \ldots, w_{t-n+1}\right)}{\partial \theta}$$

Perplexity comparison

Quantitative comparison on a standard text corpus benchmark:

Perplexity

- Word-based trigram language model: 312
- Neural language model: 252

Embeddings came (much) later

- Combines this type of representation / parameterization with PMI-like scores to get embedding vectors.

Summary (Neural LM)

- A language model is a conditional probability model for predicting/generating the next word (or character) of text
- A neural language model learns word representations within a large-scale multinomial logistic regression model.
- Nonlinearity makes the model richer (lower bias, higher variance)