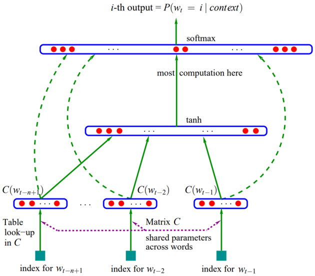
Basics

Neural Probabilistic Language Model



Learning goals

- grasp importance of the “look-up table“ a.k.a. embedding layer
- understand computational implications of language modeling

WHAT IS A LANGUAGE MODEL?

Wikipedia says:

"A statistical language model is a probability distribution over sequences of words"

This means (a) assigning a probability to a sequence of words, e.g.

$$P(\textit{"we are all interested in NLP"})$$

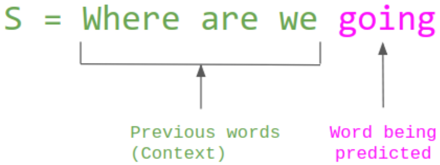
and (b) assigning a probability to the likelihood of a word given a sequence of words, e.g.

$$P(\textit{"NLP"} \mid \textit{"we are all interested in"})$$

CF. PREVIOUS CHAPTER

Predict the next token:

$$P_{(w_1, w_2, \ldots, w_n)} = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2)\ldots p(w_n|w_1, w_2, .., w_{n-1}) \\ = \prod_{i=1}^{n} p(w_i|w_1, \ldots, w_{i-1}) \tag{1}$$



P(S) = P(Where) x P(are | Where) x P(we | Where are) x P(going | Where are we)

Source: The Gradient

MAKING USE OF THE MARKOV-ASSUMPTION

The Markov-Assumption

- "The future is independent of the past given the present"
- In NLP context:
  $\rightarrow$ Next word only depends on the *k* previous words
  $\rightarrow$ *k*th order markov assumption with k to be chosen manually

"Traditional" count-based models

- Good baselines, but severe shortcomings
- Lacking the ability to generalize

WHAT ARE POTENTIAL PROBLEMS?

Curse of dimensionality

- Linear increase in context size leads to an exponential increase in the number of parameters
- Considering a vocabulary of size $|V| = 100,000$:
  $\rightarrow$ Already for bi-grams: $|V|^2 = 10^{10}$ possible combinations

Sparsity

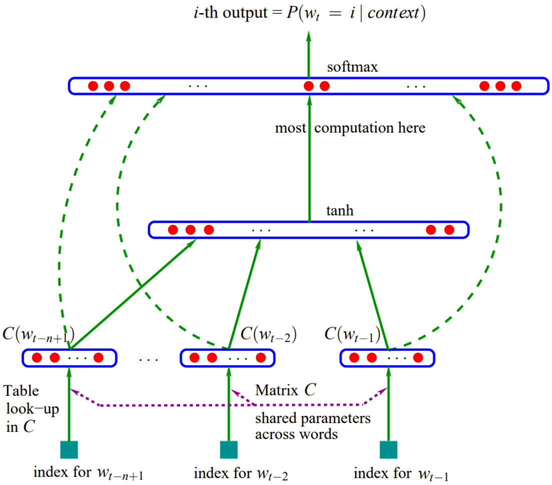
- Again, considering $|V| = 1.000.000$ & bi-grams as context
- Unlikely to observe all of them bi-gram combinations
  (a) ever
  (b) often

A NEURAL PROBABILISTIC LANGUAGE MODEL

Idea

- Using a neural network induces non-linearity and overcomes the shortcomings of traditional models
  - (a) Linear increase in $\#$parameters with increasing context size
  - (b) Better generalization
- **Input:** *Context of* $(n-1)$ *words* $[w_{(t-n+1):(t-1)}]$
- **In between:**
  - *Look-up table* $[\vec{w}^{(w_{t-n+1})}; ..; \vec{w}^{(w_{t-2})}; \vec{w}^{(w_{t-1})}]$
  - *Non-linearity* e.g. tanh, ReLU
- **Output:**
  Probability distribution over the next word $P(w_t | w_{(t-n+1):(t-1)})$

GRAPHICAL ILLUSTRATION



Source: Bengio et al., 2003

Note: $C(\cdot)$ replaced by $\vec{w}^{(\cdot)}$ on the previous slide.

WHAT COULD BE PROBLEMATIC?

Computational cost

- Vanilla softmax is expensive
- Proposed solution(s):
    1. Hierarchical softmax ▸ Morin and Bengio, 2005
    2. Sampling Approaches (next chapter)

Still relying on the markov assumption

- Context window has to be specified manually