Neural Networks

Feed-forward Language Models

Motivation

- Goal: Calculate $p(w_n|h)$ where h is the preceding $n-1$ words
- Problem: The larger n gets, there are many sequences of n words that never occur in the training data
- Traditional Solution: Use backoff / smoothing techniques with n-gram language models
- New Solution: Use a feed-forward NN to calculate $p(w_n|h)$

Representing words

- Goal: Represent each word effectively for use in the LM
- Traditional Solution: Represent each word as an integer
- Problem: Traditional solution doesn't work well with NNs
- New Solution: Represent each word as a one-hot vector

Representing words

- Goal: Similar words should have similar representations
- Problem: One-hot vectors don't have this property

Representing words

- Goal: Learn a better word representation such that similar words have similar representations
- Solution: Add an intermediate layer with weights (but without an activation function)
- Result: The vectors calculated by this intermediate layer have the desired property.

Representing words

- Terminology: These learned vectors representing words are called word embeddings.
- Interpretation: Each word vector can be considered a point in high-dimensional space.
- Result: Words that occur in similar contexts will be represented by points that are relatively near each other in this high-dimensional space.

Calculating LM probability

- Goal: Calculate $p(w_n|h)$ where h is the preceding $n-1$ words
- Mechanism: Use a feed-forward neural network
- Input: For each word in h, the input layer of the NN will contain the one-hot vector for that word.
- Output: Output layer will represent $p(w_n|h)$