Neural Networks

Recurrent Neural 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
- Previous Solution: Use a feed-forward NN to calculate $p(w_n|h)$
- New Solution: Use a recurrent NN to calculate $p(w_n|h)$

Representing words

- Represent each word as a one-hot vector of size n, where n is the number of words in the vocabulary
- Let e be the number of nodes in the embedding layer
- The weights between the input layer and the embedding layer are stored in embedding matrix E
- One dimension of embedding matrix E is size n, the other dimension is size e
- Each embedding is a vector of size e

Recurrence

- Assume we have an extra vector h_{i-1} of size q
- Connect the embedding layer to the hidden layer
  These weights are stored in matrix H.
  Multiplying the current word embedding vector times H results in a new vector of size q.
- Connect the extra vector to the hidden layer
  These weights are stored in matrix V.
  Multiplying the extra vector h_{i-1} times V results in a new vector of size q.
- Connect a set of bias weights b to the hidden layer. This vector is also of size q.
- Add these three vectors together.
  The result is new hidden state h_i

Calculating LM probability

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