HW3: Language Translation

Jonah Philion
jonahphilion@college.harvard.edu
github

March 4, 2018

1 Introduction

In this problem set, we attempt to translate German to English. The two models take the form of

$$y_{i+1} \sim \sigma(W\phi(y_{:i+1}, \mathbf{x}) + b)$$

where $\mathbf{x}$ is an input phrase in German and $\mathbf{y}$ is the translated phrase in English.

- Encoder-Decoder with no attention
- Encoder-Decoder with attention

2 Problem Description

Given a sentence in German of up to 20 words, we would like to predict a sentence in English with the same meaning. For all models, a sentence $\boldsymbol{x}_i$ is encoded as a sequence $x_1, ..., x_n$ where each x_j is a one-hot vector of length the vocabulary $\mathcal{V}$. The model outputs a categorical distribution over the vocabulary $\mathcal{V}$. Embeddings $\mathcal{E}_d$ map a one hot vector x_j to a dense vector of size d. Each language gets its own word embedding.

3 Model and Algorithms

All models are trained on the IWLST. For models requiring gradient descent, training loss and validation loss are recorded in real time with visdom. Reported PPL is on the validation set with teaching.

3.1 Evaluation

1. All models assume a vocabulary of 10000 and have a predict function which given a batch of sentence fragments outputs an array containing the probabilities for the next word for all batches.

2. All models are tested with the same evaluation function. The function evaluates the model on batches of size 32 with the same metric used during training.

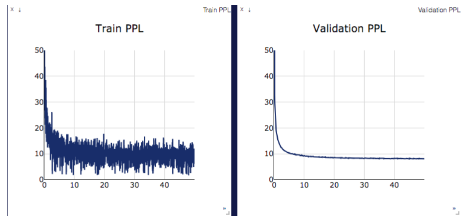

Figure 1: The network just barely reaches a PPL of 8.

4 Experiments

Both models are trained with Adam Optimizer .001 learning rate, weight_decay .0001 for 10-20 epochs

4.1 No Attention

To encode, we reverse the input sentence, then embed the sentence in a space of dimension 200. An LSTM of one direction, 4 layers deep, with initialized state and hidden layers to 0, reads through the sentence and we store the final hidden layers and state, as well the history of the hidden layer throughout the sentence.
To decode, we embed the ground truth trg, run a LSTM with initial hidden state and state given by the output of the encoder. The hidden state h_t of the decoder LSTM at any t is projected to V by a linear layer $D \to V_{en}$. The graph during training is shown in Figure 1.

PPL: 8.02

4.2 Attention

The encoder is no different from that of the non-attention model.
For the decoder, the embedding of the source sentence is first padded with zeros to be a length of 20. Next, the most recent hidden state is concatenated to the most recent word vector and fed to a linear layer $2D \to 20$. The softmax of this vector gives the "attention" to the at most 20 words in the source sentence. The input to the decoder is a linear combination of the encoding with coefficients given by the "attention" vector. As in the no-attention model, a linear layer $D \to V_{en}$ gives the output distribution over the vocabulary.

PPL:

The training graph is shown in

5 Conclusion

Ran out of time to do this as thoroughly as I should (again).