ENPM703- Assignment-3

Part1: Recurrent Neural Network

Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a type of neural network designed to work with sequential data. Unlike standard neural networks, RNNs have connections that let them remember information over time. This makes them useful for tasks like language modeling, time-series prediction, and speech recognition. At each step, RNNs take an input, update a hidden state that stores past information, and generate an output. This hidden state helps RNNs capture the patterns and dependencies in sequences.

Vanila RNN

A vanilla RNN is the most basic type of RNN. It has a single hidden layer where the hidden state is updated at each timestep based on the current input and the previous hidden state.

$$h_t = \tanh(W_{xh}.x_t + W_{hh}.h_{t-1} + b)$$

Here, x_t is the input, h_{t-1} is the hidden state, and W and b are learnable parameters. While vanilla RNNs work well for short sequences, they have trouble with long-term dependencies because of issues like vanishing gradients during the backpropagation process.

Word Embedding

Word embeddings are compact vector representations of words. Unlike one-hot encodings, which use long binary vectors with mostly zeros, word embeddings represent words as continuous-valued vectors in a high-dimensional space. In this space, words with similar meanings have similar vectors. This is achieved using an embedding matrix, where each row represents a word's vector. Word embeddings are essential for natural language processing tasks because they capture the relationships and meanings of words effectively.

Temporal Affine and Loss Layer

The temporal affine layer is used to convert hidden state vectors at each timestep into scores for a vocabulary. It applies a simple linear transformation separately at each timestep, resulting in outputs with the shape (N, T, M), where N is the batch size, T is the sequence length, and M is the vocabulary size. The temporal softmax loss then calculates the cross-entropy loss between these scores and the actual outputs. This ensures the model focuses on learning the correct word sequences.

RNN for Image Captioning

- Image features are passed through an affine transformation to compute the initial hidden state with shape (N, H), where H is the hidden state size. This step initializes the RNN using the input image.
- Input captions, given as word indices, are converted into word embeddings with shape (N, T, W), where T is the sequence length and W is the embedding dimension. These embeddings are fed into the RNN.
- The RNN or LSTM processes the word embeddings to produce hidden states with shape (N, T, H). These hidden states capture the context of the sequence at each timestep.

- Hidden states are passed through a temporal affine layer to calculate scores over the vocabulary for each timestep. The output shape is (N, T, V), where V is the vocabulary size.
- Temporal softmax computes the cross-entropy loss between the predicted scores and the actual captions, applying a mask to ignore padding tokens.