Pattern Recognition: Homework 9

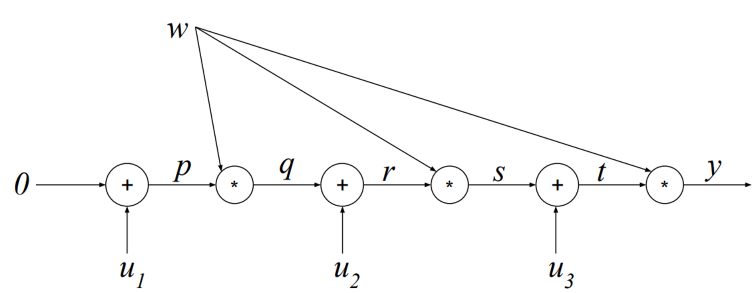
Due date: 2023.4.25

Backprop Through a Simple RNN (30 pt)

Consider the following 1D RNN with no nonlinearities, a 1D hidden state, and 1D inputs ut at each timestep. (Note: There is only a single parameter w, no bias). This RNN expresses unrolling the following recurrence relation, with hidden state ht at unrolling step t given by:

$$h_t = w \cdot (u_t + h_{t-1})$$

The computational graph of unrolling the RNN for three timesteps is shown below:



where w is the learnable weight, u_1, u_2, and u_3 are sequential inputs, and p, q, r, s, and t are intermediate values. Give expressions for t and y. Compute $\frac{\mathrm{d}y}{\mathrm{d}w}$ and $\frac{\partial y}{\partial p}$.

LSTM Gradients (30 pt)

Recall that the LSTM forward is

$$\begin{aligned}
f_t &= \sigma\left(x_t U^f + h_{t-1} W^f + b^f\right) \\
i_t &= \sigma\left(x_t U^i + h_{t-1} W^i + b^i\right) \\
o_t &= \sigma\left(x_t U^o + h_{t-1} W^o + b^o\right) \\
\tilde{C}_t &= \tanh\left(x_t U^g + h_{t-1} W^g + b^g\right) \\
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
h_t &= \tanh\left(C_t\right) \odot o_t
\end{aligned}$$

where $\odot$ means elementwise multiplication. When using an LSTM, we usually still see vanishing gradients, but the gradients should vanish less quickly. **Interpret why this might happen by considering gradients of the loss with respect to the cell state**.

(Hint: consider computing $\dfrac{\partial \mathcal{L}}{\partial C_{T-1}}$ using the terms $\partial \mathcal{L}, \partial C_T, \partial C_{T-1}, \partial h_T, \partial h_{T-1}$).

Play with Transformer (20 pt)

Background: Language Modeling. Language modeling is a central task in NLP and language models can be found at the heart of speech recognition, machine translation, and many other systems. In this homework, you are required to solve a language modeling problem by designing and implementing recurrent neural networks (RNNs), as well as Transformers. A language model is a probability distribution over sequences of Given such a sequence $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ with length m, it assigns a probability $P(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ to the whole sequence of words. In detail, given a vocabulary dictionary of words $(\mathbf{v}_1, \ldots, \mathbf{v}_m)$ and a sequence of words $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$, a language model predicts the following word $\mathbf{x}_{t+1}$ by modeling: $P(\mathbf{x}_{t+1} = \mathbf{v}_j \mid \mathbf{x}_1, \ldots, \mathbf{x}_t)$ where $\mathbf{v}_j$ is a word in the vocabulary dictionary. Conventionally, we evaluate our language model in terms of perplexity (PP) `https://en.wikipedia.org/wiki/Perplexity`. In short, PP= exp(Average Cross Entropy Loss).

We use Penn Treebank dataset. This dataset have two parts: the training set and validation set. The directory structure consists of two parts:

- `"./src/"` contains the start code.
- `"./data/"` contains the train and test set.

You can use `Pytorch` to construct model from `torch.nn.Transformer`.

Define the standard Transformer (Attention is All You Need) and train your model from scratch using the recommended deep learning framework. Still, fine-tuning from a large-scale pre-trained model is forbidden. Validate your model on the valid set, and report training and validation curves. Fill in the blanks of `model.py` for `LMModel_transformer`, and explain why do we need a source mask.

Questions (20 pt)

We have provided the data preparation code `data.py`. By running `python train.py`, you will see how the data is preprocessed. Please summarize the differences between preprocessing text data for language modeling with preprocessing image data for image recognition.