Exercises
Deep Learning
Fall 2020

Machine Learning Institute
Dept. of Computer Science, ETH Zürich
Thomas Hofmann
Web http://www.da.inf.ethz.ch/teaching/2020/DeepLearning/

Series Nov 23, 2020
(Deep Learning, Exercise Series 10)

Problem 1 (Self Attention):

Self Attention is an operation extensively used in the Transformer model. It maps a sequence of vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ to a sequence of vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_L$ by taking the weighted averages of the input.

$$\boldsymbol{y}_i = \sum_{j=0}^{L} \boldsymbol{w}_{ij} \boldsymbol{x}_j \tag{1}$$

Here, $\boldsymbol{w}_{ij}$ captures the interaction between individual input vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. One possible way to capture this interaction is by using a normalized version of the inner product, i.e.

$$\boldsymbol{w}'_{ij} = \boldsymbol{x}_i^\top \boldsymbol{x}_j \tag{2}$$

$$\boldsymbol{w}_{ij} = \frac{\exp \boldsymbol{w}'_{ij}}{\sum_{j=0}^{L} \exp \boldsymbol{w}'_{ij}} = \text{softmax}(\boldsymbol{w}'_{ij}). \tag{3}$$

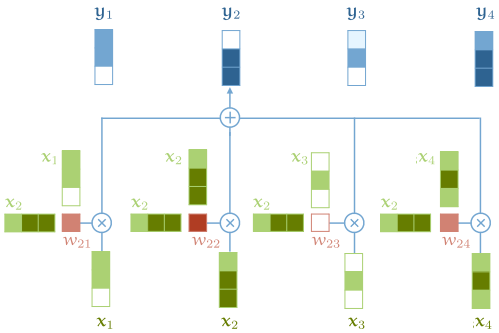

Figure 1: Self Attention

1. To efficiently implement a self attention model we need to use matrix operations. Please re-write Eq.1 for $\mathbf{Y}, \mathbf{X} \in \mathbb{R}^{\mathbf{L} \times \mathbf{d}}$.

2. In a self-attention module of a Transformer, we use three linear transformation matrices $\mathbf{W_q}, \mathbf{W_k}, \mathbf{W_v} \in \mathbb{R}^{\mathbf{d} \times \mathbf{d}}$ to transform the input $\mathbf{X}$ into queries, keys, and values, respectively.

$$\mathbf{Q} = \mathbf{X}\mathbf{W_q} \qquad \mathbf{K} = \mathbf{X}\mathbf{W_k} \qquad \mathbf{V} = \mathbf{X}\mathbf{W_v} \tag{4}$$

   What do you think is the reason for that?

3. Re-write the solution from ex. 1.1 with the transformation results $\mathbf{Q}, \mathbf{K}, \mathbf{V}$.

4. In practice, we can give the self attention greater power of discrimination, by combining several self attention mechanisms, i.e. multiple weight matrices. To simplify computation, instead of explicitly having multiple matrices, we adjust the output dimension of the transformation. In particular, we use $W_{\{q,k,v\}} \in \mathbb{R}^{d \times hd}$, where h is the number of attention heads. When doing this, what is the new dimension of the output $\mathbf{Y}$?

5. In a self-attention module of a transformer we scale the inner product inside the softmax by $\sqrt{d}$. What's the motivation behind it?

Problem 2 (Implement a Transformer):

You can find a colab notebook that provides more details about self attention and transformers here. Start by going through the explanations about self attention.

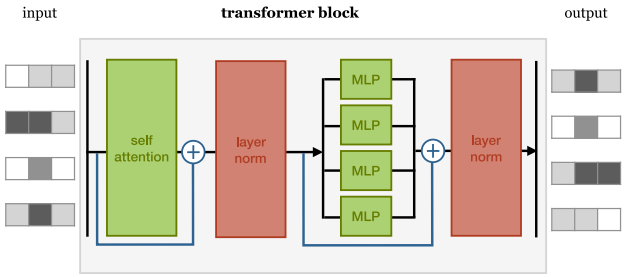

Figure 2: Transformer Block

1. Fill out the missing steps (marked with `TODO`) in the forward method of the `SelfAttention` class.
2. A Transformer block doesn't just consist of self attention. It's an architecture composed of multiple different parts. In particular, a typical block is shown in Fig. 2. Implement the depicted block by filling out the missing parts in the `TransformerBlock` class.

Problem 3 (Sentiment Classification Model):

In this exercise, we're going to implement a sentiment classification model for movie reviews. Given a review from the IMDB dataset, the model's task is to predict 0 for negative sentiment or 1 for positive sentiment. We implement this model using pytorch lightning.
You find the accompanying colab notebook here. The dataloader for the IMDB dataset and a skeleton of the lighning model is already provided. Start by inspecting some examples from the dataset.

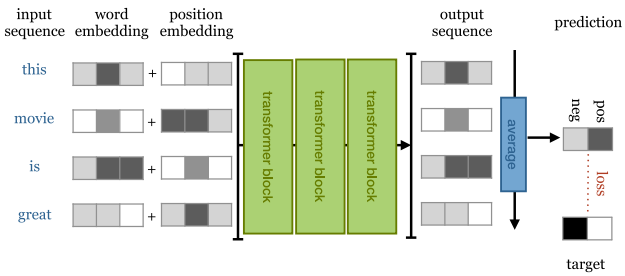

Figure 3: Sentiment Classification

1. First, we're going to implement a Transformer-based system for the sentiment classification task depicted in Fig. 3. Please complete the `TextClassificationTransformer` by adding the required steps in the

forward pass. Once done, you can train and evaluate the model using the lightning code in the cell below. (Make sure that you specify 'GPU' as the runtime hardware accelerator.)

2. To compare the performance of a Transformer to an RNN, we implement another lightning model called `TextClassificationRNN`. Please complete the model by adding the required steps in the forward pass. Once done, you can train and evaluate the model using the lightning code in the cell below.

3. Evaluate the effect of different hyperparameters for both models. You can visualize train and test metrics using tensorboard.