Deep Structured Learning (IST, Spring 2023)

Homework 2

Instructors: André Martins and Chryssa Zerva

Deadline: Friday, May 12, 2023.

Please turn in the answers to the questions below together with the code you implemented to solve them (when applicable). Please email your solutions in **electronic format** (a single zip file) with the subject "Homework 3" to:

`deep-structured-learning-instructors@googlegroups.com`

Hard copies will not be accepted.

Question 1

Transformers, modern Hopfield networks, and the Convex-Concave Procedure. Hopfield networks are a kind of neural networks which exhibit associative memory capabilities [Hopfield, 1982]. Let $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ be a matrix whose rows hold a set of examples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ ("memory patterns"), where each $\boldsymbol{x}_i \in \mathbb{R}^D$, and let $\boldsymbol{q}_0 \in \mathbb{R}^D$ be a query vector (called a "state pattern"). The Hopfield network iteratively builds updates $\boldsymbol{q}_t \mapsto \boldsymbol{q}_{t+1}$ for $t \in \{0, 1, \ldots\}$ according to a certain rule and, under certain conditions, these dynamical trajectories converge to a fixed point attractor state $\boldsymbol{q}^*$ which corresponds to one of the memorized examples.

The update rule correspond to the minimization of an energy function of the form $E(\boldsymbol{q}) = -\sum_{i=1}^{N} F(\boldsymbol{q}^\top \boldsymbol{x}_i)$, where $F : \mathbb{R} \to \mathbb{R}$ is a function. Classic Hopfield networks use $F(u) = \frac{1}{2}\|u\|^2$, so the energy function can be written as $E(\boldsymbol{q}) = -\frac{1}{2}\boldsymbol{q}^\top \boldsymbol{W}\boldsymbol{q}$, where $\boldsymbol{W} = \boldsymbol{X}^\top \boldsymbol{X} \in \mathbb{R}^{D \times D}$ are the Hopfield network parameters, which when $D \ll N$ can be seen as a "compressed memory". In the classic Hopfield network, the state vector $\boldsymbol{q}$ is further constrained to be $\{\pm 1\}$-valued, and the update rule is $\boldsymbol{q}_{t+1} = \text{sign}(\boldsymbol{W}\boldsymbol{q}_t)$.

More recent work [Krotov and Hopfield, 2016] proposed other energy functions, leading to so-called "modern Hopfield networks". In these works the state vector $\boldsymbol{q} \in \mathbb{R}^D$ is continuous and unconstrained. One of the new variants [Ramsauer et al., 2020] is:

$$E(\boldsymbol{q}) = -\text{lse}(\beta, \boldsymbol{X}\boldsymbol{q}) + \frac{1}{2}\boldsymbol{q}^\top \boldsymbol{q} + \beta^{-1} \log N + \frac{1}{2}M^2, \tag{1}$$

where $M = \max_i \|\boldsymbol{x}_i\|$ and $\text{lse}(\beta, \boldsymbol{z})$ is the log-sum-exp function with temperature β^{-1}:

$$\text{lse}(\beta, \boldsymbol{z}) = \beta^{-1} \log \sum_{i=1}^{N} \exp(\beta z_i). \tag{2}$$

In this question, you will show that there is an interesting relation between the updates in this modern Hopfield network and the attention layers in transformers.[1]

[1] If you are interested in this topic, see the long blog post in `https://ml-jku.github.io/hopfield-layers/` for more information.

1. (10 points) First, you will show that the energy (1) can be written as $E(\boldsymbol{q}) = E_1(\boldsymbol{q}) + E_2(\boldsymbol{q})$, where E_1 is a convex function and E_2 is a concave function. To do this, define $E_1(\boldsymbol{q}) = \frac{1}{2}\boldsymbol{q}^\top \boldsymbol{q} + \beta^{-1} \log N + \frac{1}{2}M^2$ and $E_2(\boldsymbol{q}) = -\text{lse}(\beta, \boldsymbol{X}\boldsymbol{q})$ and compute the gradients and Hessians of E_1 and $-E_2$ and show that the Hessian matrices are positive semi-definite.

2. (10 points) There is an iterative algorithm known as Convex-Concave Procedure (CCCP), which under certain mild conditions is guaranteed to find local minima of functions that are written as a sum of a convex and a concave function [Yuille and Rangarajan, 2003]. The algorithm works as follows: at the t^{th} iteration, it linearizes the concave function E_2 by using a first-order Taylor approximation around $\boldsymbol{q}_t$,

$$E_2(\boldsymbol{q}) \approx \tilde{E}_2(\boldsymbol{q}) := E_2(\boldsymbol{q}_t) + \left(\frac{\partial E_2(\boldsymbol{q}_t)}{\partial \boldsymbol{q}}\right)^\top (\boldsymbol{q} - \boldsymbol{q}_t).$$

   Then, it computes a new iterate by solving the convex optimization problem $\boldsymbol{q}_{t+1} := \arg\min_{\boldsymbol{q}} E_1(\boldsymbol{q}) + \tilde{E}_2(\boldsymbol{q})$. Show that the CCCP algorithm applied to the Hopfield energy function (1) leads to the updates

$$\boldsymbol{q}_{t+1} = \boldsymbol{X}^\top \text{softmax}(\beta \boldsymbol{X}\boldsymbol{q}_t).$$

3. (5 points) When $\beta = \frac{1}{\sqrt{D}}$, compare the first update $\boldsymbol{q}_0 \mapsto \boldsymbol{q}_1$ with the computation performed in the attention layer of a transformer with a single attention head and with identity projection matrices $\boldsymbol{W}_Q = \boldsymbol{W}_K = \boldsymbol{W}_V = \boldsymbol{I}$. (Note: assume the query matrix $\boldsymbol{Q} \in \mathbb{R}^{L \times D}$ of the transformer contains as rows different state patterns $\boldsymbol{q}_0^{(1)}, \ldots, \boldsymbol{q}_0^{(L)}$ and that the Hopfield updates are performed independently for each $\boldsymbol{q}_0^{(j)}$.)

Question 2

Transliteration. Transliteration is the problem of converting text (usually entity names) from one script to another. For example, `васильевич` in Russian (Cyrillic script) is transliterated as `Vassiljevitch` in English (Latin script). We can regard this is as a sequence-to-sequence problem, where the sizes of the two sequences do not necessarily match.

In this exercise, we will use the Arabic-English transliteration data released by Google (https://github.com/googlei18n/transliteration).

Run the following commands to download the train, validation, and test partitions (resp. 12877, 1431, and 1590 word pairs):

```
wget https://raw.githubusercontent.com/googlei18n/transliteration/master/ar2en-train.txt
wget https://raw.githubusercontent.com/googlei18n/transliteration/master/ar2en-eval.txt
wget https://raw.githubusercontent.com/googlei18n/transliteration/master/ar2en-test.txt
```

1. You are going to implement a sequence-to-sequence model for this task. The input and output should respectively be an Arabic and a English word, represented left-to-right as a sequence of characters. The evaluation metric is Word Accuracy (which counts the fraction of words that were fully transliterated correctly).

   (a) (10 points) Start by determining the source and target vocabularies (don't forget to include special symbols, such as START, STOP, UNK, and PAD (if you pad). What are the vocabulary sizes?

   (b) (20 points) Implement a vanilla sequence-to-sequence model using an encoder-decoder architecture with two unidirectional LSTMs (one encoder LSTM and one decoder LSTM). Report the validation accuracy as a function of the epoch number and the final test accuracy. Hint: if you're using Pytorch, use the function `nn.LSTM` for this exercise.

(c) (10 points) Repeat the previous exercise reverting the source string.

(d) (10 points) Turn the encoder into a bidirectional LSTM and add an attention mechanism to the decoder. Report the validation accuracy as a function of the epoch number and the final test accuracy.

Question 3

In this question you will explore the use of transformer architectures using the Hugging Face transformers library.

1. Implement a simple encoder-decoder architecture to solve the transliteration problem of Q2. You can use the `main-skeleton.py` skeleton code as well as the `eval_f.py` and `CharacterTokenizer.py` files for the implementation of evaluation metrics and tokenization respectively.

   (a) (10 points) Maintain the use of character-tokenization (you can can use the provided `CharacterTokenizer.py` file). You can use the `EncoderDecoderModel` generic Hugging Face architecture and modify the BERT configuration (e.g. the `bert-base-uncased` configuration). Adapt the BertConfig using the following hyperparameters:

   - Number of hidden layers: 4
   - Number of attention heads: 6
   - Hidden size: 384
   - Intermediate size: 1024
   - Max position embeddings: 128
   - Vocabulary size: adapt using the length of the vocabulary of this dataset

   Use also:

   - Early stopping with patience: 3 epochs
   - Maximum number of epochs: 10
   - Learning rate: 2×10^{-5}
   - Batch size: 4
   - Evaluation metrics: character-level BLEU and error rate
   - Main evaluation metric: error rate

   (b) (5 points) Compared to the RNN models, can the transformer architectures make use of the relative position of tokens (characters) in the sequence? If so, how is this achieved?

2. Let us now focus on word-level sequence tasks. We will experiment with T5 models for translation using the ISWLT 2017 test set for English-French translation. To download the dataset you can use the following:

```
from datasets import load_dataset
dataset = load_dataset("iwslt2017",'iwslt2017-en-fr')
```

   You will compare the performance of different sequence to sequence *pretrained* models on the test set part of the dataset.

   (a) (10 points) Compare the performance of two T5 variants: 't5-small' and 't5-base' with respect to BLEU score on the test set. Print the first 10 translations and respective references. What do you observe?

   Hint: consider how you should format the input to obtain translations with T5. Consult the Hugging Face model descriptions. Note: you can use Google Colab notebooks as discussed in Lecture 4 to facilitate computation. However you should still be able to train the `EncoderDecoder` model locally on a CPU.

References

[Hopfield, 1982] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

[Krotov and Hopfield, 2016] Krotov, D. and Hopfield, J. J. (2016). Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29.

[Ramsauer et al., 2020] Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Adler, T., Gruber, L., Holzleitner, M., Pavlović, M., Sandve, G. K., et al. (2020). Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*.

[Yuille and Rangarajan, 2003] Yuille, A. L. and Rangarajan, A. (2003). The concave-convex procedure. *Neural computation*, 15(4):915–936.