"Introduction to Deep Learning", Homework I version 1.0

Artem Chernodub, Ph.D.
Ukrainian Catholic University, Faculty of Applied Sciences
Grammarly
chernodub@ucu.edu.ua

May 8, 2021

1 Introduction

The homework is dedicated to MLP-like networks with recurrent and feedforward layers. It's goal is to learn the internal structure and training pipeline of this huge family of neural networks *under the hood*.

The homework consists of two parts, "theoretical" and "practical". The outcome of the theoretical part is a PDF document prepared in Latex (Overleaf)[1]. In the theoretical part, you are asked to write mathematical equations which describe neural network's forward and backward passes. The outcome of the practical part is code written in Python. In the practical part, you are asked to prepare the code "from scratch". Not surprisingly, in the practical part, you will code equations from the theoretical one.

The maximum score for the Homework I is 60% of total score, the rest 40% could be achieved from the Homework II. Deadlines for the first and the second parts are 7/06/2021, 9:00 AM CET and 28/06/2021, 9:00 AM CET, respectively. The penalty for missing the deadline: up to one week – minus 50% of scores, more than one week – minus 100% of scores.

[1]Penalty for Homework I prepared as a scanned paper sketch is 25% of scores for this homework part.

2 AlarmworkNet neural network's description

Please, consider AlarmworkNet, a neural network shown on Fig. 1. This is a modified version of Simple Recurrent Network [PMB13], which is also known as Elman Network. It contains two recurrent layers $Layer_{rec1}$, $Layer_{rec2}$ and one feedforward output layer $Layer_{out}$.

AlarmworkNet is designed to process sequential data, e.g. time series. We assume that each input data sample $\tilde{\mathbf{x}} = \{\mathbf{x}_1,\ \mathbf{x}_2,\ \dots\ \mathbf{x}_L\}$ is a sequence of L input vectors, where each input vector $\mathbf{x}_l$, $1 \leq l \leq L$ has dimension D. Neural network is trained to produce output vectors $\mathbf{y}$ of dimension M. Here we are interested only in the last network's output which corresponds to input vector $\mathbf{x}_L$ ("many-to-one" mode). The network is trained with Sum Squared Error loss function:

$$\mathbf{w}^* \leftarrow \sum_{n=1}^{N} \arg\min_{\mathbf{w}} (\mathbf{t}_{\{n\}} - \mathbf{y}_{\{n\}}(\tilde{\mathbf{x}}_{\{n\}}, \mathbf{w}))^2, \tag{1}$$

where $\mathbf{t}_{\{n\}}$ is target value from the training dataset, $1 \leq n \leq N$, N is number of samples in the training dataset[2].

AlarmworkNet forward pass works as follows:

1. The layer $Layer_{rec1}$ has P neurons. On the first time step all previous values are initialized by zeros by default. At l-th time step, $1 \leq l \leq L$:

   (a) $Layer_{rec1}$ receives input vector $\mathbf{x}_l$ and $\mathbf{z}^{(in_{12})}[l]$, the sum of delayed vectors from the previous time steps:

   $$\mathbf{z}^{(in_{12})}[l] = \mathbf{z}^{(in_1)}[l-1] + \mathbf{z}^{(in_2)}[l-1]. \tag{2}$$

   (b) Input vector $\mathbf{x}_l$ is multiplied on weights matrix $\mathbf{w}^{(in_1)}$, vector $\mathbf{z}^{(in_{12})}[l]$ is multiplied on weights matrix $\mathbf{w}^{(rec_1)}$. Then, these vectors are summed and bias $\mathbf{b}^{(in_1)}$ is added. Result is pre-synaptic vector $\mathbf{a}^{(in1)}[l]$.

   (c) Then $tanh$ non-linearity is applied and post-synaptic vector $\mathbf{z}^{(in1)}[l]$ is calculated.

   (d) Finally, $\mathbf{z}^{(in1)}[l]$ passes to the layer's input by recurrent connection with single time step delay d^{-1}. After that, it passes to the layer's input by recurrent connection with single time step delay d^{-1}.

[2]Here indices in square brackets $\{n\}$ denote data samples' numbers in the dataset, $1 \leq n \leq N$ please, don't confuse them with sequence's element numbers.

2. $Layer_{rec2}$ is the same as $Layer_{rec1}$, despite the following differences:

   (a) It works each second time step *only*, starting from the first time step. If $Layer_{rec2}$ is not working on particular time step l, it copies $Layer_{rec2}$'s output values from the previous time step $l - 1$.

   (b) It receives only own delayed post-synaptic vectors $\mathbf{z}^{(in_2)}[l-1]$ instead of summed delayed vectors $\mathbf{z}^{(in_{12})}[l]$.

3. $Layer_{out}$ has M outputs. The pipeline is:

   (a) $Layer_{out}$ receives vector $\mathbf{z}^{(in1)}[l]$.

   (b) Then, it is multiplied on weights matrix $\mathbf{w}^{(out)}$ and bias $\mathbf{b}^{(out)}$ is added.

   (c) Resulting pre-synaptic vector $\mathbf{a}^{(out)}$ is calculated as follows:

   $$a_j^{(out)}[l] = \sum_{i=1}^{V} w_{j,i}^{(out)} * z_i^{(in1)}[l] + b_j^{(in)}, \tag{3}$$

   where $1 \leq i \leq V = (this-is-part-of-homework)$, $1 \leq j \leq M$.

   (d) Layer's output $\mathbf{z}^{(out)}[l]$ is calculated[3] as follows:

   $$z_j^{(out)}[l] = tanh(z_j^{(out)})[l], \tag{4}$$

   $1 \leq j \leq M$.

[3]Note that we are interested for the last sequence's element only!

3 Tasks for Homework I

The maximum score for the Homework I is 60% of the total score.

3.1 Forward pass

1. Please, write down sizes of the following matrices and vectors[4] (5% of total score):
   (a) values $\mathbf{x}$, $\mathbf{y}$, $\mathbf{a}^{(in1)}$, $\mathbf{z}^{(in1)}$, $\mathbf{a}^{(in2)}$, $\mathbf{z}^{(in2)}$, $\mathbf{z}^{(in12)}$, $\mathbf{a}^{(out)}$, $\mathbf{z}^{(out)}$;
   (b) weights $\mathbf{w}^{(in1)}$, $\mathbf{w}^{(rec1)}$, $\mathbf{b}^{(in1)}$, $\mathbf{w}^{(in2)}$, $\mathbf{w}^{(rec2)}$, $\mathbf{b}^{(in2)}$, $\mathbf{w}^{(out)}$, $\mathbf{b}^{(out)}$.

2. Please, write down forward equations for $Layer_{in1}$, $Layer_{in2}$ and $Layer_{out}$ in a scalar form. They must have the same view as Eqs. (3-4). All indices must be denoted (10% of total score).

3. Please, write down forward equations for $Layer_{in1}$, $Layer_{in2}$ and $Layer_{out}$ in a vector form (10% of total score).

3.2 Backward pass

1. Please, write down the derivatives for neural network's output layer weights updates (5% of total score): $\frac{\partial Loss}{\partial \mathbf{w}^{(out)}}$ and $\frac{\partial Loss}{\partial \mathbf{b}^{(out)}}$;

2. Please, write down backward pass flow for local gradients (AKA "deltas") δ for each layer in scalar form. Use Backpropagation Through Time (BPTT) [Goo+16], p.369 for getting rid of loops in the computational graph. Here we backpropagate sum squared error loss (1). Attention! You don't need to differentiate the loss function directly, please, use delta rule for calculating deltas [Bis06], p. 244 (30% of total score).

Good luck!

[4]Hint: only matrices and vectors of equal shape can be added or subtracted.

References

[Bis06] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[PMB13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the Difficulty of Training Recurrent Neural Networks”. In: *International conference on machine learning*. PMLR. 2013, pp. 1310–1318.

[Goo+16] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

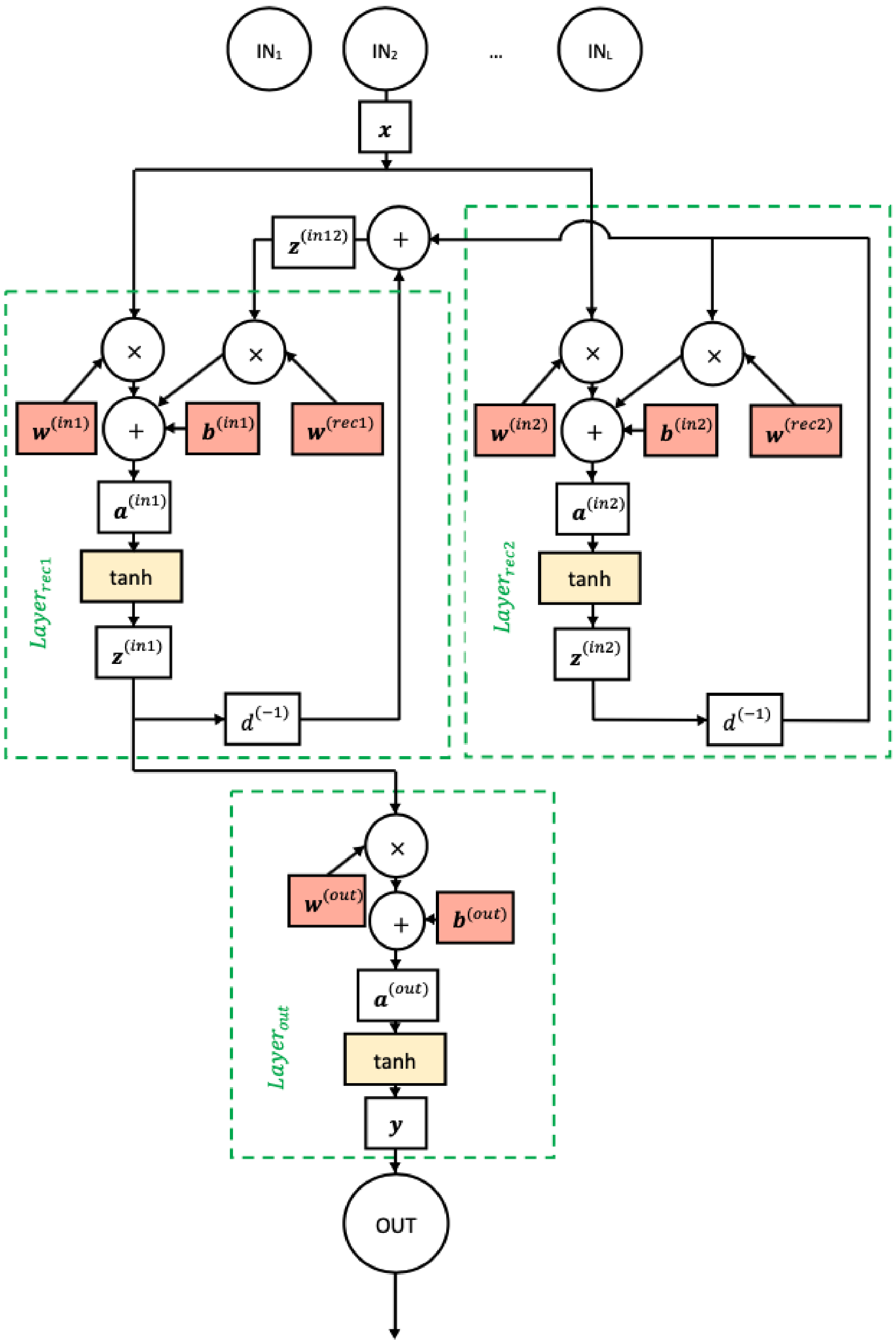

Figure 1: AlarmworkNet's computational graph, time step indices are not shown on the scheme. Layers $Layer_{rec1}$ and $Layer_{rec2}$ works in parallel, but $Layer_{rec2}$ works each second time step only. If $Layer_{rec2}$ is not working on particular time step l, it copies $Layer_{rec2}$'s output values from the previous time step $l-1$. Also, here $d^{(-1)}$ denotes single time step delay, it's the same as $z^{(-1)}$ on some other schemes