CSCE 5013-002

Deadline: Tuesday December 17, 2019

Where: In the class (Before the class begin)

Submit by email to thile@uark.edu

with email title Homework 3 - DL

Total point:100 points

Submission contains 2 parts

Part 1 is hard copy with:

- Equations for backpropagation
- Compare the forward output from your GRU_Cell.py (o1) and the o2
- Compare the backward output from your GRU_Cell.py (dx1, dhlast1) and the (dx2, dhlast2)

Part 2 is submission via email

Email title: Homework 3 – DL

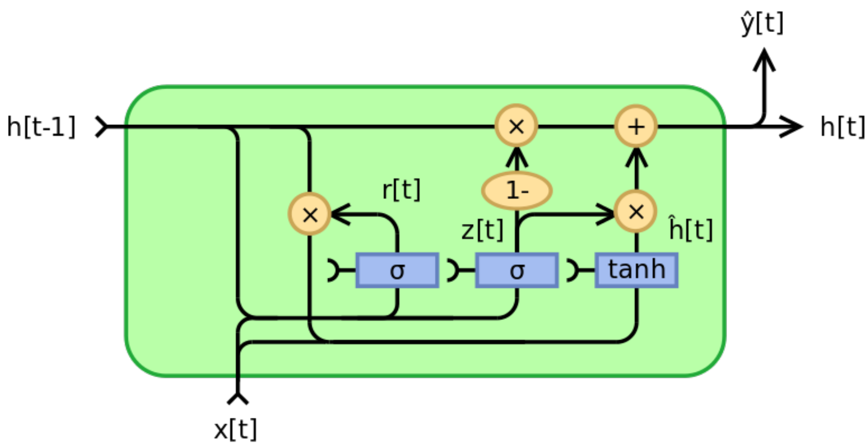
Folder name: StudentName_ID

+ GRU_Cell.py

1 Introduction

In part one of this assignment you will make a recurrent neural network, specifically you will replicate a portion of the torch.nn.GRUCell interface. GRUs are used for a number of tasks such as Optical Character Recognition and Speech Recognition on spectograms using transcripts of the dialog. This homework is to develop your basic understanding of Backpropagating through a GRUCell, which can potentially be used for GRU networks to grasp the concept of Backpropagation through time (BPTT).

2 GRU: Gated Recurrent Unit



You will be implementing the forward pass and backward pass for a GRUCell using python and numpy in this assignment, analogous to the Pytorch equivalent nn.GRUCell. The equations for a GRU cell looks like the following:

$$z_t = \sigma(W_{zh}h_{t-1} + W_{zx}x_t) \tag{1}$$

$$r_t = \sigma(W_{rh}h_{t-1} + W_{rx}x_t) \tag{2}$$

$$\tilde{h}_t = tanh(W_h(r_t \otimes h_{t-1}) + W_x x_t) \tag{3}$$

$$h_t = (1 - z_t) \otimes h_{t-1} + z_t \otimes \tilde{h}_t \tag{4}$$

Where x_t is the input vector at time t, and h_t the output. **There are other possible implementations, you need to follow the equations for the forward pass as shown above.** If you do not, you might end up with a working GRU and zero points on autolab. Do not modify the `init` method, if you do, it might result in lost points.

Similar to previous assignments, you will be implementing a Python class, `GRU_Cell`, found in `gru.py`. Specifically, you will be implementing the `forward` and the `backward` methods.

2.1 GRU Cell Forward (30 Points)

In this section, you will implement the `forward` method of the `GRU_Cell`. This method takes **2 inputs**: the observation at the current time-step, x_t, and the hidden state at the previous time-step h_{t-1}.

Use Equations 1-4 to implement the forward method, and return the value of h_t.

Hint: Store all relevant intermediary values in the forward pass.

2.2 GRU Cell Backward (70 Points)

The `backward` method of the `GRU_Cell`, is the most time-consuming task of this homework.

This method takes as input delta, and must calculate the gradients wrt the parameters and returns the derivative wrt the inputs, x_t and h_t, to the cell.

The partial derivative input you are given, `delta`, is the summation of the derivative of the loss wrt the *input* of the *next layer* $x(l+1,t)$ and the derivative of the loss wrt the input hidden-state at the *next time-step* $h(l,t+1)$.

Using these partials, you will need to compute the partial derivative of the loss wrt each of the *six weight matrices* (see Equations 1-4), and the partial derivative of the loss wrt *the input* x_t, and *the hidden state* h_t.

Specifically, there are **eight gradients that need to be computed**:

1. $\frac{\partial L}{\partial W_{rx}}$, stored in `self.dWrx`
2. $\frac{\partial L}{\partial W_{rh}}$, stored in `self.dWrh`
3. $\frac{\partial L}{\partial W_{zx}}$, stored in `self.dWzx`
4. $\frac{\partial L}{\partial W_{zh}}$, stored in `self.dWzh`
5. $\frac{\partial L}{\partial W_{x}}$, stored in `self.dWx`
6. $\frac{\partial L}{\partial W_{h}}$, stored in `self.dWh`
7. $\frac{\partial L}{\partial x_t}$, returned by the method
8. $\frac{\partial L}{\partial h_t}$, returned by the method

You **will** need to derive the formulae for the back-propagation in order to complete this section of the assignment.