# CSC 298/578 Deep Learning and Graphical Models: Homework 2

**Term:** Spring 2017
**Instructor:** Dr. Chenliang Xu
**TA:** Haofu Liao
**Due Date:** April 7, 2017 16:59 Eastern Time
**Version:** 2 (March 14, 2017)

**Constraints:** This assignment is to be carried out independently and without consulting on-line resources.

---

**Problem 1 (5 pts):** Log Softmax Loss
A log softmax loss is defined as follows:

$$\begin{aligned} L &= -\log{(p_{gt})}, \\ p_{gt} &= \text{softmax}(\mathbf{y}, gt) \\ &= \frac{\exp{(y_{gt})}}{\sum_{i=1}^{n} \exp{(y_i)}}, \end{aligned}$$

where $\mathbf{y} = [y_1, y_2, \ldots, y_n]^T$ and $gt$ is a scalar that indicates the ground truth label. Prove that

$$\frac{\partial L}{\partial y_i} = \begin{cases} p_i - 1, i = gt, \\ p_i, i \neq gt. \end{cases}$$

**Problem 2 (10 pts):** Backpropagation of Vanilla RNN Cell
A RNN cell at timestep t is defined as follows:

$$\begin{aligned} \mathbf{z}^{(t)} &= \mathbf{W}\mathbf{x}^{(t)} + \mathbf{U}\mathbf{h}^{(t-1)} + \mathbf{b} \\ \mathbf{h}^{(t)} &= \tanh(\mathbf{z}^{(t)}) \end{aligned}$$

Assume $\mathbf{W}$, $\mathbf{U}$, $\mathbf{b}$, $\mathbf{h}^{(t-1)}$, $\mathbf{h}^{(t)}$ and $\nabla_{\mathbf{h}^{(t)}} L$ are given, what are the values of $\frac{\partial L}{\partial \mathbf{W}}$, $\frac{\partial L}{\partial \mathbf{U}}$, $\nabla_{\mathbf{b}} L$, $\nabla_{\mathbf{x}^{(t)}} L$ and $\nabla_{\mathbf{h}^{(t-1)}} L$? (For full credit, pleases provide details about how you derive your answers.)

**Problem 3 (5 pts):** Backpropagation of 2D Convolutional Layer
Given an image $\mathbf{I}$ with $T$ channels, then a 2D convolution with $K$ filters of size $T \times M \times N$ at stride $s$ is defined as

$$y(k, i, j) = \sum_{t=0}^{T-1} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(t, i * s + m, j * s + n) W_k(t, m, n) + b_k$$

where $W_k$ and $b_k$ are the filter and bias of $k$th neuron. Assume $\frac{\partial L}{\partial \mathbf{y}}$ is given, what is the derivative of the loss $L$ with respect to $W_k(t, m, n)$, $b_k$ and $x(t, m, n)$, respectively? (For full credit, pleases provide details about how you derive your answers.)

**Problem 4 (80 pts):** Implementing Vanilla RNN and 2D Convolution

In this homework you will work with a framework similar to the one we provide you in Homework 1. A few things have been changed to best serve the implementation of RNN:

- the optimization module is rewritten. Now all the feedforward and backpropagation computations are moved to network classes.
- added an "initializer" module for weights and bias initialization.
- all layers are now having names and parameter initializers.
- the "Graph" class is now supporting accesses through layer names.

We provide three demo programs (*conv2d_demo.py*, *graph_demo.py*, *rnn_demo.py*) for you to understand some usages of this framework and test your implementation. You can check the documentation of this three demos for details.

Do **NOT** modify function interfaces. If you want to add parameters to a function, please provide default values so that the original behavior of the function is unchanged.

Except the functions marked with "TODO"s, do **NOT** make any changes to the framework (e.g. the Graph class). You should be able to finish this assignment with the provided utility functions and classes.

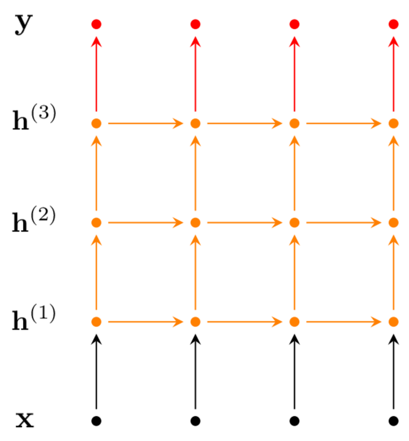
Do **NOT** use any additional python packages/modules other than those provided in the framework.

1. (40 pts) Vanilla RNN

   You will implement a vanilla RNN cell and a recurrent neural network. The definition of the vanilla RNN cell is given in Problem 2. You may use the results you get from Problem 2 for the implementation of the forward and backward pass.

   A recurrent neural network is nothing but a stack of RNN cells (sometimes along with some non-temporal layers). Figure 1 shows the typical structure of a deep RNN. A typical feedforward flow should first pass computation from bottom to top (or from first layer to the last layer) to obtain the results for the first timestep. Then, it passes the computation from left to right to compute the results for the next timestep. In backpropagation the order of computations is reversed, i.e. first from top to bottom then from right to left.

   Dr. Andrej Karpathy has a very good blog about RNN here (http://karpathy.github.io/2015/05/21/rnn-effectiveness/). Note he also provided a sample code for you to fully understand RNN. You may refer to that sample to implement this homework.



$$\mathbf{h}_t^{(i)} = f(\mathbf{W}^{(i)}\mathbf{h}_t^{(i-1)} + \mathbf{V}^{(i)}\mathbf{h}_{t-1}^{(i)} + b^{(i)})$$

$$\mathbf{y}_t = g(\mathbf{U}\mathbf{h}_t^{(L)} + c)$$

Figure 1: Recurrent Neural Networks

   - Implement the *forward* and *backward* passes for the RNN cell in the "layer" module.
   - Implement the *feedforward*, *backpropagation* and *train* functions for the RNN class in the "network" module.
   - Implement the *forward* and *backward* passes for the log softmax loss in the "loss" module.

2. (40 pts) Convolutional Layer

   The convolutional layer you will be implementing in this assignment will work for images with channels. That is, the input image will be 3D of shape $(C, H, W)$ where $C$ denotes the number of channels. The convolutional layer should support stride and padding and have multiple neurons, i.e. output multiple activation maps.

   Here (https://arxiv.org/abs/1603.07285) is a very good technique report about implementing convolution layers. Especially, you may refer Section 2.4 and Section 4.6, Relation 13 for implementing the forward and backward passes, respectively.

   Note for this assignment, we are seeking a direct implementation of convolution with nested for loops (that is how you really understand the convolution). If you choose to implement any fast convolution methods, you should explain the tricks you use in your submission.

   - Implement the *forward* and *backward* passes for the Conv2D layer in the "layer" module.

**Submission Process:** You should prepare your submission using the *collect_submission.py* program. Please run the program and upload the generated zip file to Blackboard. Your submission should contain the following:

- code/ - The implementation of the framework we provide you.
- homework.pdf - Your answers to Problem 1, 2 and 3. Optionally, the fast convolution method you use for implementing the convolutional layer.

**Grading and Evaluation:** The credit for each problem in this set is given in parentheses at the stated question (sub-question fraction of points is also given at the sub-questions). Partial credit will be given when appropriate.