Homework 7

Deadline: Wednesday, March 15, at 11:59pm.

Submission: You must submit your solutions as a PDF file through MarkUs[1]. You can produce the file however you like (e.g. LaTeX, Microsoft Word, scanner), as long as it is readable.

Late Submission: MarkUs will remain open until 2 days after the deadline; until that time, you should submit through MarkUs. If you want to submit the assignment more than 2 days late, please e-mail it to `csc321ta@cs.toronto.edu`. The reason for this is that MarkUs won't let us collect the homeworks until the late period has ended, and we want to be able to return them to you in a timely manner.

Weekly homeworks are individual work. See the Course Information handout[2] for detailed policies.

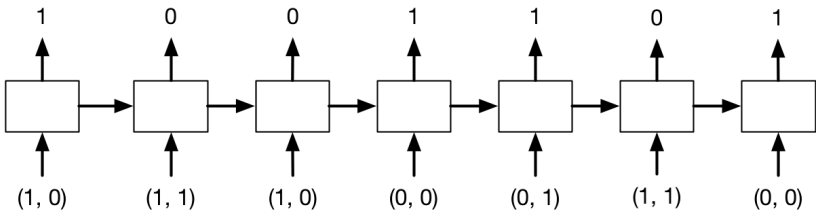
1. **Binary Addition [4pts]** In this problem, you will implement a recurrent neural network which implements binary addition. The inputs are given as binary sequences, starting with the *least* significant binary digit. (It is easier to start from the least significant bit, just like how you did addition in grade school.) The sequences will be padded with at least one zero on the end. For instance, the problem

$$100111 + 110010 = 1011001$$

would be represented as:

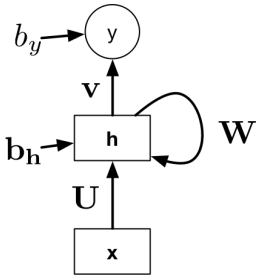
- **Input 1:** 1, 1, 1, 0, 0, 1, 0
- **Input 2:** 0, 1, 0, 0, 1, 1, 0
- **Correct output:** 1, 0, 0, 1, 1, 0, 1

There are two input units corresponding to the two inputs, and one output unit. Therefore, the pattern of inputs and outputs for this example would be:



Design the weights and biases for an RNN which has two input units, three hidden units, and one output unit, which implements binary addition. All of the units use the hard threshold activation function. In particular, specify weight matrices $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$, bias vector $\mathbf{b_h}$, and scalar bias b_y for the following architecture:

[1] `https://markus.teach.cs.toronto.edu/csc321-2017-01`
[2] `http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/syllabus.pdf`



Hint: In the grade school algorithm, you add up the values in each column, including the carry. Have one of your hidden units activate if the sum is at least 1, the second one if it is at least 2, and the third one if it is 3.

2. **LSTM Gradient.** Here, you'll derive the backprop updates for the univariate version of the LSTM. For reference, here are the computations it performs:

$$
\begin{aligned}
i^{(t)} &= \sigma(w_{ix}x^{(t)} + w_{ih}h^{(t-1)}) \\
f^{(t)} &= \sigma(w_{fx}x^{(t)} + w_{fh}h^{(t-1)}) \\
o^{(t)} &= \sigma(w_{ox}x^{(t)} + w_{oh}h^{(t-1)}) \\
g^{(t)} &= \tanh(w_{gx}x^{(t)} + w_{gh}h^{(t-1)}) \\
c^{(t)} &= f^{(t)}c^{(t-1)} + i^{(t)}g^{(t)} \\
h^{(t)} &= o^{(t)} \tanh(c^{(t)})
\end{aligned}
$$

(a) **[3pts]** Derive the backprop update rules for the activations and the gates:

$$
\begin{aligned}
\overline{h^{(t)}} &= \\
\overline{c^{(t)}} &= \\
\overline{g^{(t)}} &= \\
\overline{o^{(t)}} &= \\
\overline{f^{(t)}} &= \\
\overline{i^{(t)}} &=
\end{aligned}
$$

You don't need to vectorize anything or factor out any repeated subexpressions.

(b) **[1pt]** Derive the backprop rule for the weight w_{ix}:

$$\overline{w_{ix}} =$$

(The other weight matrices are basically the same, so we won't make you write those out.)

(c) **[2pt]** Based on your answers above, explain why the gradient doesn't explode if the values of the forget gates are very close to 1 and the values of the input and output gates are very close to 0. (Your answer should involve both $\overline{h^{(t)}}$ and $\overline{c^{(t)}}$.)