# Deep Learning
# Assignment 2: Convolutional Neural Networks

Instructor: Jia Xu

Fall 2020

Due: November 26th 2019

## Build a Convolutional Neural Networks [6pts]

You should upload a zip file in the Canvas that contains *everything* to reproduce your systems and plots, including but not limited to codes, comments, data sets (or command to load data), library calls, running commands, output accuracies, tune values, output plots, and a readme file.

1. step 1: download and install Matlab. (You are allowed to use Python instead of Matlab)
2. step 2: follow "https://www.mathworks.com/help/deeplearning/examples/train-a-convolutional-neural-network-for-regression.html;jsessionid=2e3223fbb96455c26f42924ea1cf" to build a CNN.
3. step 3: test the neural network and output its accuracy. [4 points]
4. step 4: list the five worst predicted samples. [3 point]
5. step 5: explain the reasons for the failed prediction on those samples. [3 point]

## Comparison with Random Learning [4pts]

On your above CNN system, change the gradient descent training algorithm into the below algorithm:

Randomly select a weight, and random choose a new value for the weight. If the loss is reduced, then keep the new value, otherwise randomly select another weight and repeat this routine until converge.

Compare the training and testing accuracy as well as the training speed with the gradient descent, then explain the reason of getting this result.

## Bonus Questions: Recurrent Neural Networks

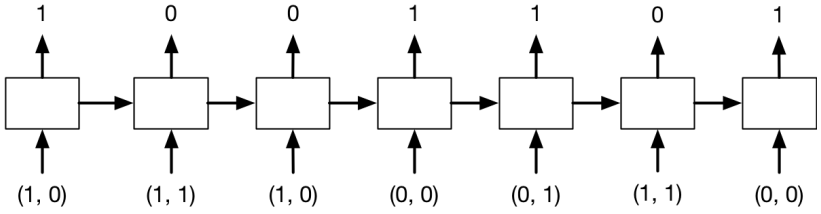
1. **Binary Addition [5pts]** In this problem, you will implement a recurrent neural network which implements binary addition. The inputs are given as binary sequences, starting with the *least* significant binary digit. (It is easier to start from the least significant bit, just like how you did addition in grade school.) The sequences will be padded with at least one zero on the end. For instance, the problem

$$100111 + 110010 = 1011001$$

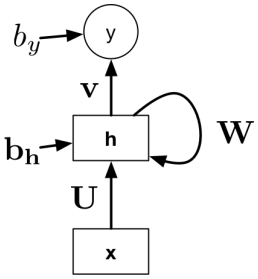
would be represented as:

- **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:




*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 [5pts]** Here, you'll derive the Backprop Through Time equations for the univariate version of the Long-Term Short-Term Memory (LSTM) architecture.

   *Note:* This question is important context for understanding LSTMs, but it is just ordinary Backprop Through Time, so you know enough to do parts (a) and (b) after Lecture 15 (RNNs). Part (c) is optional because it depends on Lecture 16, but you should try to do it yourself. You may find it helpful to read Section 3.4 of the Lecture 16 notes.

   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) **[4pts]** Derive the Backprop Through Time equations 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 BPTT equation 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) **[optional, no points]** 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 $h^{(t)}$ and $c^{(t)}$.)