Assignment 3: Deep Learning

Due November 14 at 11:59pm
100 marks total

This assignment is to be done individually.

Important Note: The university policy on academic dishonesty (cheating) will be taken very seriously in this course. You may not provide or use any solution, in whole or in part, to or by another student.

You are encouraged to discuss the concepts involved in the questions with other students. If you are in doubt as to what constitutes acceptable discussion, please ask! Further, please take advantage of office hours offered by the instructor and the TA if you are having difficulties with this assignment.

DO NOT:

- Give/receive code or proofs to/from other students
- Use Google to find solutions for assignment

DO:

- Meet with other students to discuss assignment (it is best not to take any notes during such meetings, and to re-work assignment on your own)
- Use online resources (e.g. Wikipedia) to understand the concepts needed to solve the assignment

---

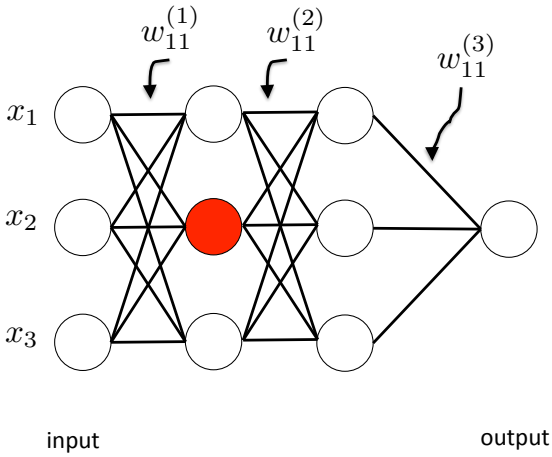
1 Error Backpropagation (30 marks)

We will derive error derivatives using back-propagation on the network below.

Notation: Please use notation following the examples of names for weights given in the figure. For activations/outputs, the red node would have activation $a_2^{(2)} = w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3$ and output $z_2^{(2)} = h(a_2^{(2)})$.

Activation functions: Assume the activation functions $h(\cdot)$ for the hidden layers are logistics. For the final output node assume the activation function is an identity function $h(a) = a$.

Error function: Assume this network is doing regression, trained using the standard squared error so that $E_n(w) = \frac{1}{2}(y(\boldsymbol{x}_n, w) - t_n)^2$.



Consider the output layer.

- Calculate $\frac{\partial E_n(w)}{\partial a_1^{(4)}}$. Note that $a_1^{(4)}$ is the activation of the output node, and that $\frac{\partial E_n(w)}{\partial a_1^{(4)}} \equiv \delta_1^{(4)}$.
- Use this result to calculate $\frac{\partial E_n(w)}{\partial w_{12}^{(3)}}$.

Next, consider the penultimate layer of nodes.

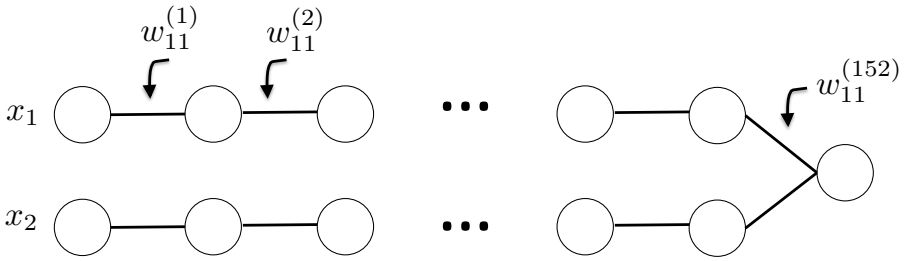
- Write an expression for $\frac{\partial E_n(w)}{\partial a_1^{(3)}}$. Use $\delta_1^{(4)}$ in this expression.
- Use this result to calculate $\frac{\partial E_n(w)}{\partial w_{11}^{(2)}}$.

Finally, consider the weights connecting from the inputs.

- Write an expression for $\frac{\partial E_n(w)}{\partial a_1^{(2)}}$. Use the set of $\delta_k^{(3)}$ in this expression.
- Use this result to calculate $\frac{\partial E_n(w)}{\partial w_{11}^{(1)}}$.

2 Vanishing Gradients (40 marks)

Consider the network below. Use the same notation as the previous question.



- Write an expression for $\frac{\partial E_n(w)}{\partial w_{11}^{(\ell)}}$ for all layers ℓ in the network.
- Suppose we use logistic sigmoid activation functions in this network. Describe what would happen to the gradient for weights early in the network, $\frac{\partial E_n(w)}{\partial w_{11}^{(\ell)}}$ for smaller ℓ. For example, when would these gradients be very small? When would they be reasonable in magnitude?
- Suppose we use rectified linear units (ReLU) in activation functions. When would the gradients be zero?
- Suppose we modify the graph to have bipartite connections at each layer, still using ReLU activation functions. When would the gradients be zero?

3 Fine-Tuning a Pre-Trained Network (30 marks)

In this question you will experiment with fine-tuning a pre-trained network. This is a standard workflow in adapting existing deep networks to a new task.

We will utilize Keras (`https://keras.io/`) a high-level interface that can run on top of TensorFlow (`https://www.tensorflow.org/`).

The provided code builds upon Inception V3, a state of the art deep network for image classification. Inception V3 has been designed for ImageNet image classification with 1000 output classes.

The Inception V3 model has been adapted to solve a (simpler) different task, classifying an image as either containing basketball, hockey, or soccer.

The code `imagenet_finetune.py` does the following:

- Constructs a deep network. This network starts with Inception V3 up to its penultimate layer. Then, a small network with 32 hidden nodes then 3 output nodes (dense connections) is added on top.
- Initializes the weights of the Inception V3 portion with the parameters from training on ImageNet.
- Performs training on only the new layers – all other weights are fixed to their values learned on ImageNet.

The code and data can be found on the course website. For convenience, a VirtualBox appliance that has the data and pre-installed libraries ready to run on "any" machine has been provided.

If you wish to download and install yourself, you will need Tensorflow (v 0.10.0), Keras (v 1.1), and their dependencies. This is reasonably easy to do on a Mac / Ubuntu box (e.g. pip install ...). Tensorflow on Windows seems difficult, hence the virtual machine.

What to do:

Start by running the code provided. It will be *very* slow to train unless you have a good GPU. Try to do one of the following tasks:

- Write a Python function to be used at the end of training that generates HTML output showing each test image and its classification scores. You could produce an HTML table output for example.
- Try running this code on a dataset of your choice (not the sports images).
- Try modifying the structure of the new layers that were added on top of Inception V3.
- Try adding data augmentation for the training data.
- The current code is inefficient because it recomputes the output of Inception V3 every time a training/validation example is seen, even though those layers aren't being trained. Change

this by saving the output of Inception V3 and using these as input rather than the ImageDataGenerator currently used. See the online tutorials on pre-trained models with Keras for ideas.

- The current code does not train the layers in Inception V3. After training the new layers for a while (until good values have been obtained), turn on training for the Inception V3 layers to see if better performance can be achieved.

Submitting Your Assignment

The assignment must be submitted online at `https://courses.cs.sfu.ca`. You must submit three files:

1. An assignment report in **PDF format**, called `report.pdf`. This report must contain the solutions to questions 1-2 as well as the description of what you did in question 3 and what the results were.
2. A .zip file of all your code, called `code.zip`.