CS 6.5: Theories of Deep Learning
Problem Sheet 1

Prof. Jared Tanner

October 21, 2019

Tasks 2 (Expressivity Theorems) and Taks 3 (Backpropogation Algorithm) should be submitted for marking. The remaining computational experiments should be attempted but need not be submitted; they will be discussed in tutorials.

1. Getting Started:

   (a) Download and install a programming interface of your choice, preferably MATLAB or Python notebooks; [http://maths.ox.ac.uk/members/it/software-personal-machines/matlab] [http://jupyter.org].

   (b) For Python make sure you have key libraries that you will require for machine learning development, specifically: SciPy, NumPy, Matplotlib, Pandas, Statsmodels, and Scikit-learn.

   (c) Like Python but don't want to end up messing your laptop with installation?
   A Google account is all you need! Go to Google CoLab [https://colab.research.google.com/].
   Most machine learning and deep learning libraries comes pre-installed. Click 'File' and then 'New Notebook'. Your notebooks gets saved to your Google drive and there are lot of examples available online.

2. Task1: Data Preparation
   A dataset is a collection of pairs $\{x_i, y_i\}$, where x_i are inputs or examples (e.g., images) and y_i are corresponding outputs or targets (e.g., class labels).

   (a) Consider n-ap problem (the n-alternating-point problem). Dataset consisting of a set of n uniformly spaced points within $[0, 1 - 2^{-n}]$ with alternating labels, i.e., the pairs $((x_i, y_i))$ with $x_i = i2^{-n}$, and $y_i = 0$ when i is even.
   Create this dataset for $n = 3$ and visualize the points by plotting.

   (b) Consider a XOR problem, i.e., x_i are now vector valued.
   Create this dataset and visualize the points by plotting. Write your thoughts about different ways you can solve this classification problem?

   XOR? Check this wiki [https://en.wikipedia.org/wiki/Exclusive_or]

   (c) Real Data: Let's move to MNIST, a benchmark dataset of handwritten digits, that has approximately 70,000 example images from 10 classes. Each 2-dimentional image is of size '28×28', and belongs to one of the categories '0-9'. So the goal is to prepare the data which includes:

      i. Downloading, and loading the data. Write a function to visualize the images.

      ii. Vectorizing each image i.e., converting '28×28' grid of numbers to a 784-dimensional feature vector. Why Vectorization? We will talk more about this in class.

iii. Split data into two 'Train' and 'Test' sets i.e., into Matrices of size $N \times M$, where $N = 784$, and M depends on size of split you choose.
Former is used to learn and later set is used to test a ML model.

iv. Writing a function which extracts a subset of Train or Test set.
We call this 'Mini-Batch' generation in machine learning! When the model has trained on multiple mini-batches such that all the examples of Train set are seen at least once, we refer to it as an 'Epoch'.

3. Task2: Expressivity Theorems:
Why depth is required?

   (a) Recall the sawtooth function $f(x) = \sigma(2\sigma(x) - 4\sigma(x - 1/2))$ used by Telgarsky[1] to derive a function requiring exponential width of a shallow network while only polynomial number of depth + width for a deep network; we refer to $f(x)$ as a $2 - sawtooth$ function as it is piecewise affine with 2 pieces. Show that if $f(x)$ is a $t - sawtooth$ then the family of function composed of ℓ feedforward layers of width m is the space of $(tm)^{\ell} - sawtooth$ function.

4. Task3: Backpropogation Algorithm
Before we actually train a network, we need to understand how to train one using gradient descent and the way to do that through multiple network layers is via the 'Backpropogation Algorithm'.

   (a) Derive the formulae for backpropogation of fully connected feedforward network with sum of squares error $\min_{\theta} \sum_{i=1}^{n} \|h_N(x_i) - y_i\|_2$ subject to $h_1(i) = x_i$ and $h_{j+1}(i) = \sigma(W^{(j)} h_j(i) + b^{(j)})$ for $j = 1, \ldots, N - 1$. Comment on how the choice of nonlinear activation $\sigma(\cdot)$ effects the updates of θ; in particular, why might ReLU or $tanh$ be preferable to hard-$tanh$.

5. Task4: Designing Your Own Neural Network
The goal of this task is to build a back propagation network to process a dataset. For this assignment, we want you to implement the neural net (data generator, activation function and training/testing code) yourself, not using Tensorflow/PyTorch or other software tools. The purpose is for you to understand the nitty gritty of what these tools are doing for you before we switch over to using the tools later.

   (a) n-ap problem: Train a feed-forward network starting with one hidden layer, but constant number of nodes per layer. Observe how the network performs as the network depth increases.

   (b) XOR problem: Now let's try the XOR problem. If you have solved this problem in Task1, you know the solution. So let your code do the magic and verify!

   (c) MNIST digit classification: Finally, we will work with real dataset. Design and train a one hidden layer neural network to solve this problem. Observe the effect of number of nodes, choice of activation function, mini-batch size and number of epochs.

[1] `https://arxiv.org/abs/1509.08101`