ELL409: Assignment 3

Last updated: 23-10-2021, 7:44pm

Deadlines

1. Deadline for final submission of assignment report and all code: **10 November 2021, 11:59 PM**.

Instructions

1. While you are free to discuss all aspects of the assignment with your classmates or others, your code and your results and report must be entirely your own. We will be checking for any copying, and this will be treated as plagiarism and dealt with accordingly. In case of any doubts in this regard, please ask us.
2. **You are free to use any platform to write and test your code. However, if you are using Python, please make sure you submit .py files for evaluation during the viva**.
3. In addition to the code, you should prepare a report, compiling all your results and your interpretation of them, along with your overall conclusions. In particular, you should attempt to answer all of the questions posed in the respective parts of the assignment below. Any graphs or other visualisations should also be included therein.
4. A single tar/zip file containing the source code and report has to be submitted (via Moodle). The zip/tar file should be structured as per the below guidelines:
    - Upon deflating all submission files should be under a directory with the student's registration number. E.g., if a student's registration number is `20XXCSXX999` then the tarball/zip submission should be named `20xxcsxx999.zip` and upon deflating **all contained files** should be under a directory named `./20xxcsxx999` only. If this is not followed, then your submission will be rejected and will not be evaluated.
5. The schedule for demos/vivas will be announced by your respective TAs, in advance. If for any reason you cannot attend in your scheduled slot, you must arrange for an alternative slot with your TA well in advance. Last-minute requests for rescheduling will normally not be accepted.

Weightage

This assignment comprises parts worth a total of *4 units*, with each unit being worth 4% of your overall course grade. Note that you are free to attempt only some subset of the units[1]; but there is a sequential dependency between the sub-parts within Part 1, which means that first Part 1A should be attempted, and then (depending on time/interest) Part 1B.

Objective

To experiment with the use of Neural Networks for a multiclass classification problem, and try and interpret the high-level or hidden representations learnt by it. Also, to try and understand the effects of various parameter choices such as the number of hidden layers, the number of hidden neurons, and the learning rate.

[1] At the end, the best 15 out of all the units you have attempted during the course will be considered for the final grade.

Part 1

Data

For the previous assignment, you were provided a low-dimensional (PCA) representation of a data set of images. We now provide the corresponding original data: a personalised file for each of you, that contains $3,000$ images, each of size 28×28 pixels. You should download your file from `http://web.iitd.ac.in/~sumeet/A3/<EntryID>.csv` (for example, `http://web.iitd.ac.in/~sumeet/A3/2013EE10447.csv`). Each row of this file corresponds to an image, and contains 785 comma-separated values. The first 784 are the grayscale pixel values, normalised to lie between 0 and 1 (ordering is row-major), and the last one gives the class label for the image (there are 10 classes, denoted by the labels 0 to 9).

Part 1A (2 units, 8 marks)

Your task is to try and learn a Neural Network classifier for these images, starting with the raw pixels as input features, and thereby also to assess the usefulness of the different representations that your Neural Network constructs. Here is how you should proceed:

1. As in the previous assignment, we would like you to do your own implementation as well as trying out a standard library for comparison.

   (a) Write your own implementation of a basic (fully connected) neural network for multiclass classification of the provided images: you essentially need to implement the backpropagation algorithm to be able to train the network weights. Your implementation should allow for some flexibility (the more the better) in setting the hyperparameters and modelling choices: number of hidden layers (try with at least 1 and 2), number of units in each layer, gradient descent parameters (learning rate, batch size, convergence criterion), choice of activation function in the hidden units, mode of regularisation.

   (b) In addition, familiarise yourself with one Neural Network library of your choice. One suggestion is PyBrain for Python, but you can find many others. You may wish to play with a simple toy data set to get a feel for using the library, before you move on to the actual data for this assignment.

2. *Standard backpropagation neural net:* Use your implementation to train a neural network to recognise the images of handwritten digits given to you. Set aside some of the data for validation, or ideally, use cross-validation. Assess the accuracy and speed (both training and testing) of the neural net for different settings of the various hyperparameters mentioned above. Identify cases of overfitting or underfitting; use regularisation to get better results, if you think it will help. Once you have obtained a good model, try to visualise and interpret the representations being learnt by the hidden neurons. Can you make sense of them? Also, take a look at the images which are being misclassified by the network. Can you see what's going wrong? In addition, try using the standard library, instead of your own implementation, just to train the final model with your optimised hyperparameters. Does this alter the results in any way? If so, why might that be?

Part 1B (1 unit, 4 marks)

Comparison with PCA features: Now consider the PCA-space representation of your data that you were provided for the previous assignment. This was a way of mapping the images to a lower-dimensional space, something that the neural net is also doing via its hidden units. Try to interpret and compare these two representations. Is the neural net in any sense able to learn a better representation than the PCA one? Train another neural network, using the PCA features from last time as the input features instead of raw pixels. First try with *no hidden layers*, i.e., a simple logistic regression model. Now add a hidden layer. Does it help? Why or why not? And how do these results compare with those obtained using just the raw pixels?

Part 2 (1 unit, 4 marks)

Advanced neural networks: This part is more open-ended. You should try experimenting with two of the most widely used deep learning approaches: (supervised) convolutional neural nets and (unsupervised) sparse autoencoders (using standard libraries). The networks you train need not be very deep; 2 or 3 layers is fine. The objective is primarily to see if these approaches can learn more useful representations than a standard neural net as employed above; and how these representations vary as you change the model hyperparameters. In order to have more training data, you may make use of the full MNIST data set, available at http://yann.lecun.com/exdb/mnist/. That page also provides a list of benchmark results with different kinds of techniques; you should see how close you can get to those benchmarks. More importantly, you should attempt to interpret the representations being learnt by your deep nets. Are these in some sense more natural or intuitive than the representations learnt by a standard neural net (as in Part 1)? How do they vary with hyperparameter changes, and why?

(Please note that there are standard online tools that can do these tasks for you, and many people will have reported example results on MNIST data. You may use such tools for training your model; but all results, graphs/visualisations, and interpretation in your report must be your own. Any copying of these from elsewhere will constitute plagiarism.)