Assignment 2: Autoencoders

Welcome to your second assignment. Upload the completed scripts task1.py and task2.py to the VLE before the deadline.

Standard criteria for valid assignments

In general, an assignment is considered valid if the following criteria are met:

- The delivered program uses Python and Tensorflow.
- The delivered program code is organised and easy to understand.
- Only one code file is used per task, which is a modified version of the provided scripts, with the same file names.
- The delivered program does not use any random number generator seeds. Program will be run 5 times to get an average error.
- When the delivered program is run, the model is trained from scratch using gradient descent (or an extension of gradient descent).
- The way the final test error is calculated is left as is.
- Only the provided dataset is used and without modification (although the data may be transformed for use by the model).
- The test set does not influence training in any way.
- The delivered program code is not excessively modified. I will be correcting your program by checking what was changed from the original provided scripts and needless modification makes corrections slower.
- Make sure that when 'full_output' and 'show_analysis' are set to false, the only thing that is shown after running the program are the error, model size, and execution time, exactly as coded in the provided script files.
- Your aim should be to deliver the fastest, smallest, and best performing (according to test error) program possible within the given task constraints.

Task 1: Autoencoder (50%)

Summary: Learn a simple autoencoder neural network that generates MNIST handwritten digits.

You are provided with a file “dataset.txt” that contains a collection of flat bitmap images of hand written digits together with a half-finished Python script “task1.py”. Your task is to complete the script in order to learn a simple autoencoder using gradient descent with Tensorflow which accurately regenerates the given digits in the dataset. You may split the dataset into separate validation sets if needed and you may use any technique discussed during the lectures.

The script also includes code to perform sensitivity analysis performed on a random test image for each digit. In the corner is the digit that the image is classified as by the neural network. The red blobs show parts of the image that when changed will be the most to change the classification.

Constraints

This task in particular is considered valid if the following criteria are met:

- The model is a simple autoencoder.
- The thought vector is 256 elements long and consists of numbers between -1 and 1.
- The average final test error (of 5 runs) is less than or equal to 8% (high performance error: 4%).
- The average running time on CPU is less than 3 minutes.
- The number of parameters is less than or equal to 1,100,000.

Task 2: Multi-task learning (50%)

Summary: Learn a multi-task model that acts as a simple autoencoder neural network that generates MNIST handwritten digits as well as a classifier that classifies digit encoded in the thought vector.

Like the previous task, create a simple autoencoder. From the thought vector extend separate layers to classify the digit encoded in the thought vector (do not use a separate input, the idea is to force the thought vector to be able to give separate encodings for each digit). The script's name is "task2.py".

Note that it might be necessary to use one of the advanced optimisers for this task rather than simple gradient descent.

Constraints

This task in particular is considered valid if the following criteria are met:

- Part of the model is a simple autoencoder.
- Another part of the model is a classifier that starts from the autoencoder's thought vector such that the thought vector is a shared layer.
- All information about the image being passed to the classifier must come from the thought vector only.
- The thought vector is 256 elements long and consists of numbers between -1 and 1.
- The decoder's average final test error (of 5 runs) is less than or equal to 10% (high performance error: 5%).
- The classifier's average final test error (of 5 runs) is less than or equal to 15% (no high performance error for the classifier).
- The average running time on CPU is less than 3 minutes.
- The number of parameters is less than or equal to 1,500,000.