INF554: Assignment 1

Neural Networks in TensorFlow

Introduction

You will receive a tutorial about TENSORFLOW and be guided through an example, so that you have the necessary requirements to complete the assignment.

Lab Tasks

1. Implement logistic regression in TENSORFLOW.
2. Train and test your classifier on the MNIST data (use 60,000 examples for train and 10,000 for testing). Use a learning rate that you see suitable.
3. Compare the results to those of your kNN implementation from `Lab 2`
4. Add a hidden layer, and repeat. Try different learning rates and different number of hidden units.

Make sure you understand the assignment tasks.

Assignment Tasks

You will build a multi-layer neural network in TensorFlow suitable for carrying out **multi-label classification**, which is similar to regular classification but with multiple class variables, such a model h obtains predictions

$$\hat{\mathbf{y}} = h(\mathbf{x})$$

for test instance $\mathbf{x}$, where $\hat{\mathbf{y}} = [y_1, \ldots, y_L]$. Your specific tasks are as follows:

1. Load the Scene dataset provided (`data/scene.csv`): The first 6 columns represent the target values (class labels). The remaining values comprise a representation of an image (scene). Beware that there is a header row. Use a 60/40 train/test split.
2. Implement a multi-layer neural network using TensorFlow for multi-label classification. Note that your network will be evaluated under *Hamming loss*, for each instance $\mathbf{x}$:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{j=1}^{L} [y_j = \hat{y}_j]$$

3. Prepare your code in the file called `my_net.py` in the form of a SCIKIT-LEARN classifier (a template is given).
4. Finish/modify the script `run_eval.py` and compare to your classifier to the following:
   - `sklearn.multiclass.OneVsRestClassifier`
   - `sklearn.multioutput.ClassifierChain`
   - `sklearn.neighbors.KNeighborsClassifier`

- `sklearn.tree.DecisionTreeClassifier`

(you can leave these 4 methods with their default parameters).

5. Experiment with the hyper-parameters on your network (learning rate, hidden layers, ...[1]). Try to obtain the best results possible, but make sure not to overfit the test set.

6. Compose a one (1)-page report in `pdf` format, with 4–6 plots/tables, to show:

    - Your investigations into hyper-parameters (and their effect on predictive performance, running time, etc.)
    - The parameters that obtained the best accuracy (these should also be the default parameters in your code)
    - Epochs vs Hamming loss of your network, compared to the above-mentioned classifiers (one horizontal line each)
    - Epochs vs running time of your network, compared to the above-mentioned classifiers (one horizontal line each)

    Note: Each plot/table should be self explanatory in terms of axes, title, and legend; and a small caption explaining in one or two sentences the main conclusions from the plot. Keep text to a minimum.

Submission and Grading

Grade:

- Report (60%)
- Hamming loss† (20%)
- Code‡ (20%)

† We use a script, based on the example `run_eval.py`, to call your `fit` method repeatedly for 2 minutes, and calculate a score based on the predictive performance compared to baselines and other members of the class. This is a separate evaluation to the Report (feel free to run your code for more than this time). However, for this to work:

- Your in code must run under the requirements specified for the labs (using NUMPY and TENSORFLOW).
- Do not change the filename `my_net.py` or the pre-defined class name (`Network`).
- The default default parameters of your class should be the best ones (we will not change them).

‡ We will have a look at your code / visual inspection. However, your code in `run_eval.py` should also be able to reproduce the results you give in the report (obviously running time, etc., may vary) – in case we need to validate the results.

Note: **We will not spend time debugging you code to get it to run, or reading past the first page of the report**.

Deliverables

Create a zip file with name `project.zip`, containing:

- Your report (in format PDF), `report.pdf`
- Your code, inside the files `my_net.py` and `run_eval.py`

and submit it to Moodle (the zip file should *not* include any data or folders or subfolders).

[1] The only restrictions: use only dense-layers (no convolution/recurrent layers, etc.) and an iterative optimizer (over a number of epochs)