CSCI4156 Assignment 1 – Canonical Multi-layer Perceptron

Instructor: Malcolm Heywood (mheywood@cs.dal.ca)

Due: 16 May – BrightSpace

Build your own Multi-Layer Perceptron

The purpose of this assignment it to provide a canonical neural machine learning model that we will incrementally adapt to support reinforcement learning. The pre-prequisite class (csci3151) has already established the following for a multi-layer perceptron (MLP):

- **Representation:** two layers of 'perceptrons' where each perceptron performs the following mapping from a vector of inputs ($\vec{x}$) to scalar output or $y = f(net)$ where $net = \vec{w} \cdot \vec{x} + w_0$ and $\langle \vec{w}, w_0 \rangle$ denote the learning parameters (weights) that will be adjusted during credit assignment.
  - The number of inputs at the first layer is defined a priori by the task domain.
  - Likewise, the number of outputs is also set by the task domain.
  - The transfer (or activation) function $f(\cdot)$ takes the form of the sigmoid operator, or $f(net) = (1 + \exp(-net))^{-1}$
- **Cost function:** defines the performance criterion as empirical error minimization, or the specific case of $J_{SSE} = \frac{1}{2} \sum_{p \in N} e_p^2$
- **Credit Assignment:** is the process by which learning parameters $\langle \vec{w}, w_0 \rangle$ are adapted. Such a process can be considered a form of optimization (given the the model has a fixed number of learning parameters).
  - We are interested in solving this iteratively using the (stochastic) gradient decent heuristic.
  - This implies that for each exemplar a back-propagation weight update will be performed.

The Ask

The objective of assignment 1 is to develop your own software that implements the multi-layer perceptron with a single hidden layer given the following arguments:

1. number of input features
2. number of outputs
3. number of hidden layer neurons
4. maximum number of passes through the training partition

5. training dataset name as a .csv file

6. test dataset name as a .csv file

7. if no arguments are supplied, you should specify the argument sequence and meaning (which should be as above)

You should assume a Sigmoid transfer (activation) function and implement your MLP in Python **without** the use of any machine learning specific libraries (e.g. TensorFlow or Keras). Your submission will be evaluated on the Iris classification dataset[1] and a second mystery dataset. You should assume that class labels are encoded using a one hot encoding (e.g. for a three class problem $\vec{y} = \{\{1,0,0\},\{0,1,0\},\{0,0,1\},\}$).

It is recommended that training data is subject to a feature-wise z-score normalization, or

$$x'_{i,p} = \frac{x_{i,p} - \mu_i}{\sigma_i} \tag{1}$$

where i is the feature and p is the data instance. μ and σ are the feature specific mean and variance as estimated over the data partition.

Post training, you should be able to take a second .csv file (the test partition) and suggest labels for each data record. Evaluation of your software will be performed under the undergraduate server (`timberlea.cs.dal.ca`). In particular, you should not assume any support for third party libraries (e.g. TensorFlow or Keras). Attempting to do so will result in a zero grade.

Your predicted labels for the test file should appear in an output .csv file with the file name: "B00XXXXXX" where the 'X' values correspond to your banner number.

[1] Included with assignment as a .csv file. First 4 attributes are input features, last 3 attributes are output labels.