Homework 1 : Supervised Deep Learning

João Ramos da Silva Tabuaço Freitas
Università Degli Studi di Padova
Student ID: 2009440

18 July 2022

Introduction

Supervised learning comprises the simplest applications of neural network models with the purpose of classifying data constructed by numerous features. The nature of the task at hand depends on the type of labels associated with the data. Labels forming a continuous set pertain to **regression** tasks, while data with associated labels forming discrete sets make up **classification** tasks. In this report, these two tasks are performed by implementing neural network models with the PyTorch framework. A fully connected network is employed for the regression task, while a convolutional one is used to perform the classification. While both tasks are qualified as supervised learning, they differ enough that it is worth addressing them separately. Thus, this report addresses the two tasks separately.

1 Regression

The goal of the regression task is to learn a single-variable objective function. The model performance is quantified by the mean squared error (MSE) between the model output $\hat{y}$ and the data's respective label y. The goal is to find the set of parameters $\{\hat{w}\}$ that minimize the MSE:

$$\{\hat{w}\} = \arg\min_{w} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \qquad (1)$$

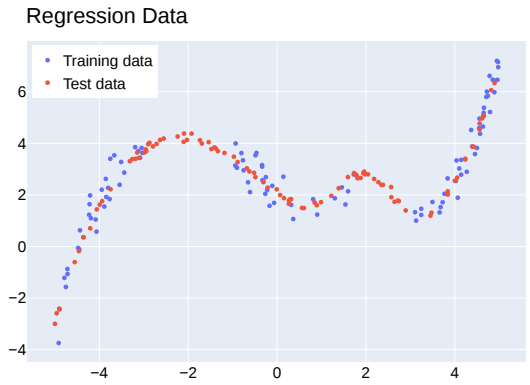
MSE is advantageous in comparison to the mean absolute error because it penalizes points that are more correctly classified in a lighter manner, while placing a stronger emphasis on improving the model such that data that lead to a larger loss are prioritized. Traditionally, MSE minimization is implemented with the method of least squares. However, it requires making an assumption about the number of parameters (i.e. choosing an adequate curve to attempt fitting). The advantage of using a deep learning model lies in the large number of degrees of freedom which can, in turn, mimic the behavior of the objective function faithfully with several different configurations of its parameters. Because of that, it is possible that different models yield similarly acceptable results, thus increasing the likelihood of finding an adequate representation of the objective function. While the computational cost of training deep-learning models can quickly increase due to the tensorial nature of the parameters,

it is offset by the computational prowess of graphical processing units (GPUs), which are thoroughly used in this exploration.

1.1 Methods

A fully-connected neural network is implemented to learn the functional behavior of some unknown finite polynomial. Its architecture consists of three hidden layers. The number of neurons in each layer is tuned through a Bayesian-like parameter search, implemented using the Optuna framework.

The data used in the regression task are divided in a training and a test dataset. Uniform noise is added to the training labels, and significant intervals of the data space are removed. These transformations help combat overfitting. The test data is left unaltered. Because the model is relatively simple, both the training and test sets are composed of 100 samples.



The gaps in the training data will likely be the main drivers of the model's validation loss. Because of that, a combination of regularization techniques will be implemented.

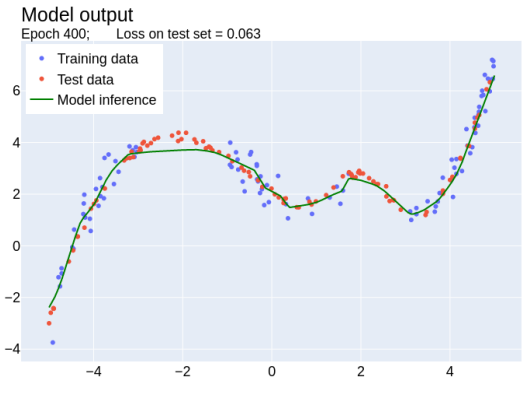
The adequate model architecture was determined by a preliminary random search. This involved randomly sampling 50 architectures. The structure was kept constant: three hidden layers, each with associated neuron drop-out. Each model was preliminarily trained over 200 epochs. The number of neurons in each hidden layer, along with the activation function, optimizer type and respective learning rate were all randomly sampled according to Optuna's TPESampler. This sampler starts out with uniform sampling of hyperparameters, becoming increasingly sensitive to the prior models tested as the grid-search progresses. This strategy offers an adequate compromise between the computationally exhaustive classic grid-search and a completely random search. Naturally, this faces an issue that is quite similar to vanishing gradients: the sampler might keep generating similar architectures that yield loss close to a local minimum, while missing other configurations that could, in theory, perform better. Unfortunately, the nature of the TPESampler means this issue is inevitable. The best preliminary architecture had the following parameters:

N_1	127
N_2	168
N_3	80
Activation	ReLU
Dropout	0.019
Optimizer	Adam

A new model with these parameters is then initialized with the Glorot scheme.

1.2 Results

Upon finding a set of best parameters, the model was reinstantiated with those, and trained using K-fold cross-validation, with 5 folds of the training data. Each training was performed over 400 epochs.



2 Classification

The classification task aims to train a model to categorically classify input data. Its most common form consists in classifying a set of images. The raw data consists of pixel values in some given shape, grouped in one or multiple channels. These are normally pre-processed, with several possible objectives ranging from data augmentation to promoting model regularization.

2.1 Methods

YAYAYA involves the analysis of the well-known FashionMNIST dataset, a variation from the standard handwritten digit dataset MNIST. The data consist of images of shape $(1 \times 28 \times 28)$ representing different articles of clothing, in black and white. The categorical nature of the data motivates encoding labels in a discrete manner.

2.2 Results

More text.