DEEP GLOBAL FEATURE IMPORTANCE EXTRACTION

Akanda Ashraf
akandaashraf@outlook.com

April 20, 2021

ABSTRACT

Identifying important feature is crucial in many domains, ranging from drug discovery, clinical diagnosis to reduce the computational complexity of any machine learning model. In this work, I have taken a simple but intuitive approach to rank features using backpropagation of a deep learning model.

1 Introduction

A vast majority of traditional feature importance estimation techniques mainly focuses on linear associations. In this work, I have used a deep learning model to estimate feature importance, which is capable of identifying highly non-linear interaction/associations between features due to its inherent capability of nonlinear function estimation using stochastic gradient descent algorithms.

In this approach, all the input attributes are individually passed through a single unit neuron (feature importance neurons) without any bias term, then the are concatenated and passed to the core neural network architecture. Each of the neurons is constrained to have a non-negative kernel value and does not of any activation function (i.e. linear units) 1. The weights of these feature importance neurons have been initialised from a uniform distribution. After training a model these neuron's weights are extracted from the model, which gives us an indication of the importance of each of the features for the classification task. These weights are then estimated using backpropagtion.

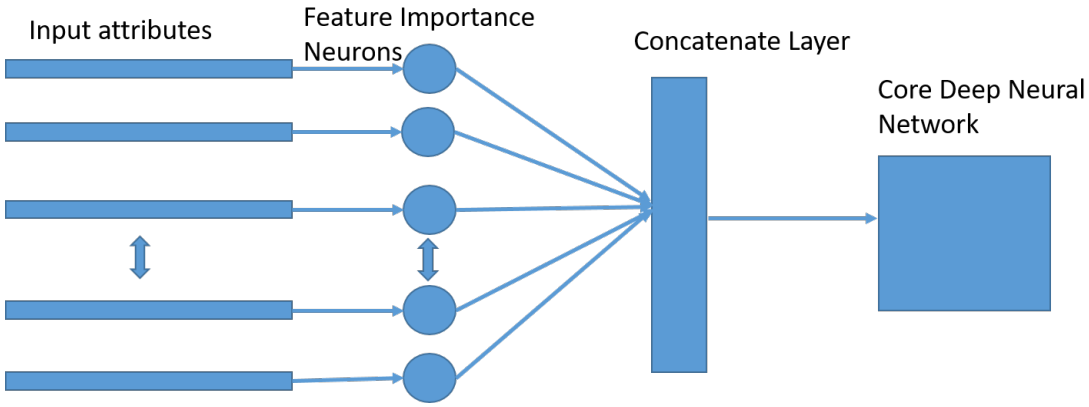

Figure 1: Model architecture

2 Experimental Setup

The input data, with 13 labels have been l2 row normalised. Additionally, the output classes are split between different labels. Each of the outputs labels corresponds to an output with a sigmoid activation in the model. The 5 classes, turned into 5 labels (each of them with binary classes) as independent outputs. The model is then trained using a soft-f1-loss function taken from `https://towardsdatascience.com/the-unknown-benefits-of-using-a-soft-f1-loss-in-classification-systems-753902c0105d` Adam optimiser has been used as a stochastic gradient descent based backpropagation algorithm to train the model.

The model consists of 3 blocks, each of them containing a dense fully connected layer with 128 Units, with l2 kernel and bias regulariser, a gaussian noise layer (to reduce overfitting), and a dropout layer. There are three such blocks before the five output layers. The last block

does not have a Gaussian noise layer. The learning rate used is 0.0001, batchsize = 64, and the data is split into half and half for training and validation. This big split has been done as our focus is to learn the importance of the features rather than achieving high accuracy. The training has been early stopped if the validation loss does not decrease for 15 epochs and the best model with the lowest loss is saved, from where the feature importance neurons weights are extracted. There are a total of 100 models trained in such a way and their feature importance weights are then extracted and averaged to determine the importance of each of the features for the classification task.

3 Results

After training 100 models and taking the average feature extractor neuron's weights, the following feature rank has been estimated and results are shown in Figure 2. The average weights are normalised.

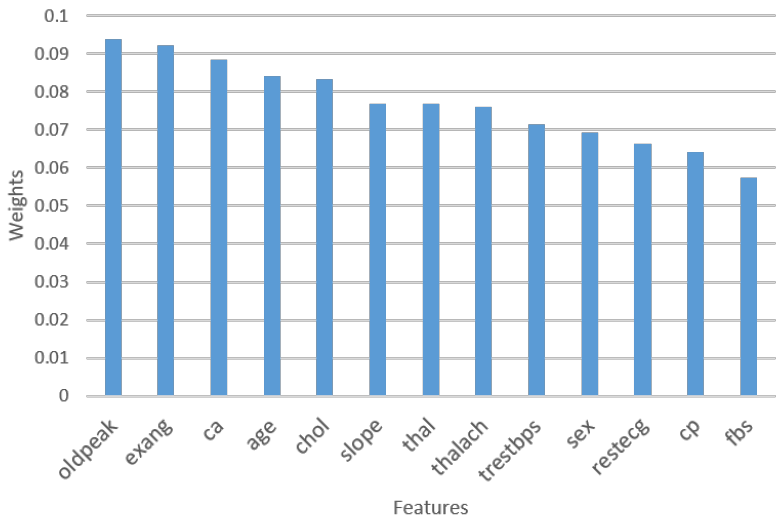

Figure 2: Most important features to the left and least to the right for the classification task

Results and the code is shared on personal OneDrive: `https://1drv.ms/u/s!AuoV5dGuivVglR3hhsZbPWAZJf7C?e=9BHyHU` The metric used for each of the models is precision and recall, the performance are available in file: `Models/logs_100.json`

Additionally, the raw results (weights for the feature importance identification neurons) in a CSV file can be found on this file: `ExpOutput/output_weights_100.csv`

4 Running the Experiment

The best way to run is to use to build the docker image and run it from there. Download this directory. Make sure docker and docker-compose are installed in the system. The follow these steps:

1. cd to the downloaded directory
2. *docker-compose build*
3. If you wish to use the dataset given in the task then use the default env variables in .env. nothing needs to be changed. However, any dataset can be used, just specify them in the env var parameters, with the attribute names as a csv file. See files in the volume for the data format.
   - Data:/DeepFeatSelection/Data
   - Models:/DeepFeatSelection/Models
   - ExpOutput:/DeepFeatSelection/ExpOutput
4. *docker-compose run experiment*
5. Provide the number of iterations/ models you wish to train (larger number will result in a better result but will take more time. To test use, a small number e.g. 2/3)
6. The results will be generated in the volume, i.e. ExpOutput
7. The ranked features will also be provided in the console with the weights/importance of each of the features