Sensyne Health

Machine Learning – Programming Assignment

Alfredo Zermini

Input data generation

The dataset provided, named 'processed.cleveland.data', is a $(302, 14)$ matrix, where 302 is the number of rows and 14 is the number of features. Some of the rows contain missing values, where the actual value has been replaced by '?'. Including these rows in the training set could compromise the training of the neural network, thus need to be cured. A possible solution could be using imputation techniques, to estimate the missing values. However, considering that the number of incomplete rows in the dataset is negligible (i.e. 6 rows), deleting the incomplete rows could be considered as a reasonable solution. After curing the dataset, the final size is $(296, 14)$.

Before generating the training set, it is important to understand which features should be included, and which other features may have a negative impact when training the network. For this reason, an $F-test$ has been performed, in order to measure which features are statistically significant. The score for each feature is calculated. Those features whose p-value is below the 5% threshold could be neglected, in this case, '$blood\_sugar$', '$serum\_col$' and '$rest\_electrocard$'.[1]

After discarding the less significative features, the training set is generated by randomly selecting 80% of the data, the remaining 20% will be used as a testing set. A validation set will be generated during training by using 10% of the training set.

The training set is made of two subsets:

- The target labels are created by using the last feature in the dataset, which is an integer ranging in $[0, 4]$, which indicates the diagnosis of heart disease, where '0' indicates no presence. These labels integers are transformed into one-hot vectors.
- The input data are generated by extracting the remaining features, and normalizing each of the features independently, such that each features ranges in $[0, 1]$.

[1] As a further confirmation, including these features into the training set has shown reduced accuracies for both validation and testing, compared to the case where the training set did not include these features.

The networks architecture

In this assignment, two different networks have been tested, a fully-connected network (FCN) and a convolutional neural network (CNN), which can be both used for a comparison of the classification performance.

They both share the same input architecture: each feature is assigned to a different input, for a total of 10 inputs, where 10 is the number of features used. The total input size for the fully-connected network is $(None, 10)$, while for the CNN this needs to be expanded to $(None, 10, 1)$, in order to be connected to the convolutional layers.

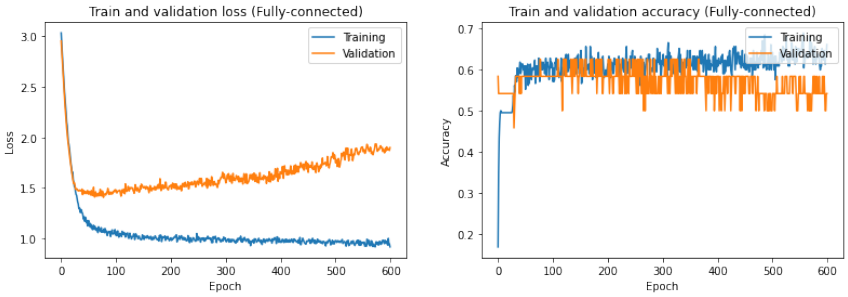
The parameters for the two networks have been empirically selected, by taking into account the curves in Figure 1. The two networks have a very similar architecture. They are both autoencoders, where the input data are first encoded and then decoded, by first reducing the size of the network and then expanding the size of the network, in order to extract the most meaningful features. In both architecture, each layer has an $L2$ regularizer and is followed by a dropout layer with a 30% dropout rate. These are very standard techniques to help preventing over-fitting. The output of both network is a 5 neurons dense layer, with 'softmax' activation function, each neuron corresponding to the probability estimated for each of the target labels.

- Fully-connected network. The network has 5 hidden dense layers, respectively containing 64, 32, 16, 32 and 64 neurons, with 'relu' activation function.
- Convolutional network. The network has 5 hidden $1D-$convolutional layers, respectively containing 64, 32, 16, 32 and 64 filters, kernel size 2 and stride 1, with 'relu' activation function. Padding is set to 'same' to preserve the layers dimensions. Each layer, is followed by a MaxPooling layer with pool size 2, which also helps to prevent over-fitting, except for the innermost one (for dimensionality reasons). The last layer is a flattening layer, which is used to connect the last convolutional layer to the output dense layer.

Training

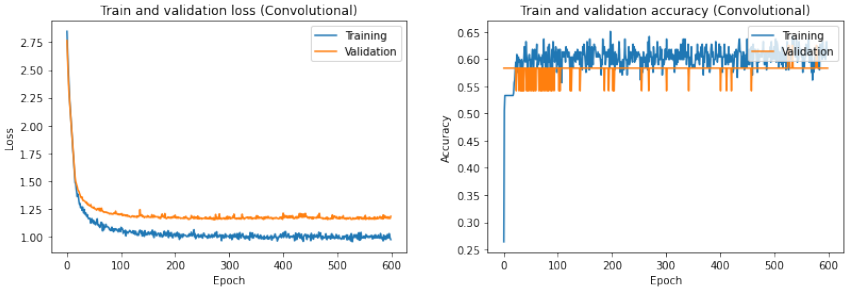
Each network is trained by using 'adam' optimizer, categorical-cross entropy loss and categorical accuracy as a metric to monitor the process. The number of training epochs is set to 600, with batch size 32. The best network is selected by over-writing the a new model each time there is an improvement in the validation accuracy.

Figure 1 (a) and (c) show the training (blue) and validation (orange) loss for the two network architectures, while Figure 1 (b) and (d) show the corresponding accuracy. These plots show that the training losses and accuracy stop improving after 300 epochs, while the corresponding validation curves seem to plateau earlier than that, except for the fully-connected validation loss, which slightly over-fits.



(a) FCN, loss function.

(b) FCN accuracy.



(c) CNN, loss function.

(d) CNN, accuracy.

Figure 1: Training and validation loss and accuracy as a function of the epochs.

Results and discussion

The performance of the two networks are comparable, with a testing categorical-accuracy of 68.3% for both architectures. These results are not particularly good, but can be justified by the small size of the training set. It is well-known that the performance of neural networks generally improves by providing additional training data. The similar results achieved by two different architectures suggests that more training data may be necessary to improve the classification accuracy. Running the networks with different features suggests that all the features, except for *blood_sugar*', '*serum_col*' and '*rest_electrocard*' , have a positive impact on the prediction accuracy of the strokes.

Environment

The following libraries are required to run the code:

- python 3
- tensorflow 2
- pandas
- numpy
- sklearn
- matplotlib
- jupyter (used to run the notebook)