https://2022.ivmsp.org/

Biological and Perceptual-based Processing

Deep learning architectures and pipelines; Dictionary learning; Reinforcement learning; Interpretable learning; Incremental learning

Multi-Temporal and Spatio-Temporal Processing

Abstract

…

1. Introduction

Artificial Intelligence refers to the ability of a machine being able to reproduce a person’s cognitive functionality, such as learning, designing, and decision making. Human brains processing information differently than a conventional digital computer was a key motivator for more research in Artificial Neural Networks (ANNs). In particular, by using appropriate models and recognizing various patterns it is possible to have viable solutions of complex problems.

ANNs’ models are composed of many non-linear identical units operating in parallel and arranged in patterns similar to biological neural networks. Their key element, extensive parallelism, is essential for high performance speech / image recognition, decision support, as well as the diagnosis of illnesses requiring Medical Doctors’ (MDs) excessive training.

Predicting breast cancer diagnosis, given a dataset extracted from a computed digitized image of a fine needle aspirate (FNA) of a breast mass, by means of ANNs architectures, is the topic treated in this paper. By looking at the image and the features extracted, a decision has to be made, either this breast mass being ‘benign (B)’ or ‘malignant (M)’, essentially making the matter to be dealt with a two-class problem.

Deep learning artificial neural networks were chosen as the appropriate experiment models. They are trained via commonly established percentages (90% to 60%) of the provided dataset features. The rest of the dataset (10% to 40%) is used as accuracy metrics in order for the ANNs to diagnose patients’ symptoms from data they were subjected to.

Data provided first had to undertake pre-processing in order to be mapped in patterns’ set of artificial neurons inputs / outputs (I/O). Also, Self-Organizing Maps, a special ANNs’ class were employed to detect possible data intra-relations.

Section 2 covers data preprocessing procedure which constitutes a necessary process before trying to fit the model.

2. Data (pre-)Processing and Model Construction

A picture containing table

Description automatically generatedIn order to fit to the model, understanding how the data are mapped as I/O patterns is a key factor. Basic python libraries were put to use, such as “pandas”, “numpy”, “seaborn”, “matplotlib”, <sklearn’s> “preprocessing” and “model\_selection” for visualization and data handling while <keras’> “models” and “layers” for model construction, compiling and finally fitting.

Firstly, intra-relations have to be sought out. Achieving that, Kohonen’s algorithm about self-organizing maps was put to use.

Having the aforementioned results, handling I/O patterns’ missing values is also crucial since when dealing with structured data, missing values are an inevitable occurrence. However, two commonly applied methods comprise either dropping these values or forwarding suitable ones. One of the most general used formulas, in case of the second one, is described below.

Tracing missing values.

After database was loaded on the script as a “dataframe”, via “pandas” library, no such values were found at all (Tracing missing values).

Chart, bar chart

Description automatically generatedMapping data, floating point numbers were given to every feature type with the exception of the output; it was handled as binary, because it could only be either ‘benign (B)’ or ‘malignant (M)’, easily utilizing <Label Encoder> from sklearn’s preprocessing library. Also, there was no need for balancing them the I/O patterns according to the ‘B’ / “M’ percentages, as shown in the Fig. 1”.

Figure 1.

<a>

Table

Description automatically generatedLooking at the object types, there was noticed that every feature’s type was a floating point number with the exception of the output (label), which was an object but it was handled as binary, because it could only be either ‘benign (B)’ or ‘malignant (M)’, easily utilizing <Label Encoder> from sklearn’s preprocessing library. Also, there was no need for balancing them the I/O patterns according to the ‘B’ / “M’ percentages, as shown in the Fig. 1”.

<a/>

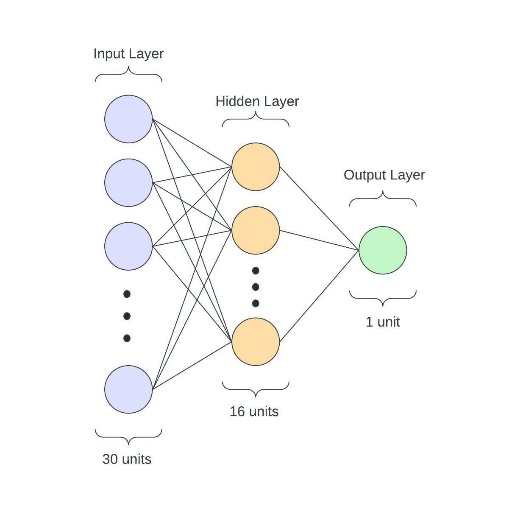
The data’s representation was then dealt. Some values are in <tens>, some other in <hundreds> and some in <one-tenth> or <one-hundredth>, some wide scale values. Thus, values were normalized to fit utilizing <MixMaxScaler> from sklearn’s preprocessing library. As the name suggests, ‘Scaling’ receives a minimum and a maximum value from a certain range of numbers and transforms them in a ratio between ‘zero (0)’ and ‘one (1)’.

Unscaled values.

Text

Description automatically generatedThe final step was splitting the given dataset in two, a training and a testing set. With the first set the model was to be trained with while by the latter, constituting a set of features ANNs were never exposed to, was going to be used for measuring the model’s ‘accuracy’.

Scaled values.

Subsequently, model construction procedure gets going importing the Sequential model type from keras’ <models> library. As the input consists of thirty features, net’s input layer needs to be consisted from thirty units, at least. These units consist the first layer of the network, known as <Input Layer>, and its output feeds directly the consecutive one, called <Hidden Layer>. Every unit (neuron), except input ones, is composed of some weighted inputs, a transformation function and an activation function, corresponding to the biological neuron’s axon. The main idea of this architecture could be likened as the one of a black box. The only think visible to the observer is the input fed and the given output while content inside can’t be viewed. Between Input layer and Hidden layer (or as called <Dense> layer) there is a random twenty percent (20%) drop of the connections, avoiding overfitting issues. This is a good technique of generalizing the model. Going on, there is not a specific rule of choosing the number of hidden layers or the number of units consisted of but in this case, there is just one layer, a <Dense>, which is plenty enough, consisted of 16 units (neurons). Finally, comes the last layer of units (neurons) that produces the given outputs, or as called the <Output Layer>. This layer’s neurons may be built in a different way given the ones described above, in order to streamline and improve the end results. Due to problem’s nature, just one neuron is enough to describe the output which is either ‘<M>’ or ‘<B>’.

<a>

* https://link.springer.com/article/10.1007/s00500-018-3203-0#Abs1
* https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw
* https://www.techopedia.com/definition/33263/output-layer-neural-networks
* Jeff Heaton mentioned that «the optimal size of the hidden layer is usually between the size of the input and size of the output layers»
* There is a general formula in choosing the number of hidden layer neurons:
* *«The training mechanism of traditional neural networks is that all the internal parameters need to be iteratively fine-tuned by using a gradient descent technique (Hinton and Salakhutdinov*[*2006*](https://link.springer.com/article/10.1007/s00500-018-3203-0#ref-CR4)*; Bengio et al.*[*2013*](https://link.springer.com/article/10.1007/s00500-018-3203-0#ref-CR1)*). During the training process, the derivatives of the loss function are continually back propagated to each hidden layer to guide the parameter adjustment until the difference between the model prediction and real observation is small enough.»*

<a/>

3. Discussion

…

4. Conclusions

…