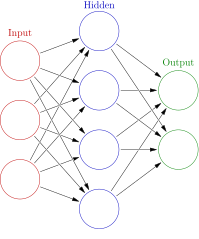
1. Introduction
2. The multilayer perceptron (MLP), unlike most common perceptrons, classifies data that are not linearly separable. This is because they use a more robust and difficult architecture to learn regression and classification models for difficult data sets. The multilayer perceptron (MPL) has at least three layers of nodes. 1st input layer, 2nd hidden layer, 3rd output layer. The second and third node layer is a neuron using a nonlinear activation function.



IMG. 10 Example of an image with Multiplayer receptor

Multilayer perceptron may have other names:

[10] *“ The term MLP is used ambiguously, sometimes loosely to mean any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation); see § Terminolog*[*y*](https://en.wikipedia.org/wiki/Multilayer_perceptron#Terminology)*. Multilayer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.”*

1. Training a Multilayer Neural Network

**Example of operation**

**[11] Forward propagation**

MPLs have an arbitrary number of hidden layers located between the input layers as well as between the output layers.

To begin with, weights are assigned to each input layer depending on its characteristics flowing down to the hidden layers.

The weights are then joined with the requirements of the inputs (these requirements can be anything such as arithmetic operations: addition, subtraction, multiplication, etc.) to pass to the activation function, the most commonly used of which are sigmoid functions. The result of the hidden layer becomes the input of the next hidden layer.

This process will be repeated as many times as we have programmed, so far we have only moved in the forward direction.

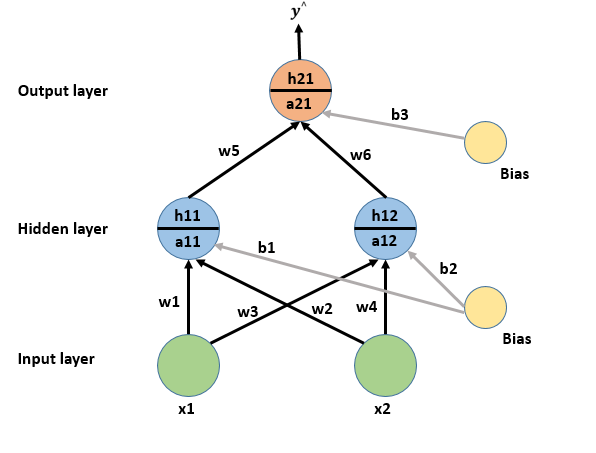
This is known as forward propagation.

This system has a problem that error in one neuron is due to the weights of its previous neurons. To minimise the error you have to adjust the learning rate.

Learning rate: It is the percentage of change with which the weights are updated in each iteration, in other words, each time an iteration is performed in the training process the input weights must be updated in order to give a better approximation each time. It should be noted that:

If small you have to iterate many epochs.

If it is very large it may not converge



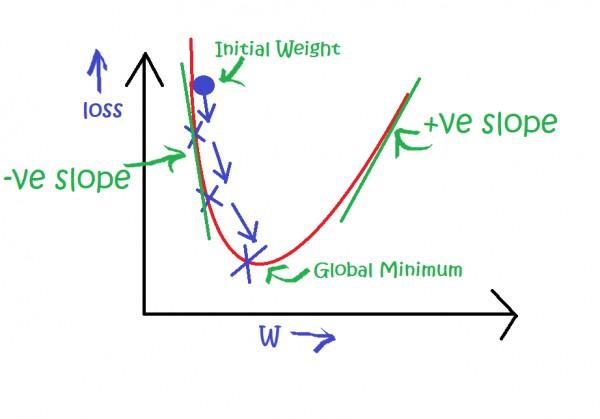
IMG. [11] Example of Forward propagation.

### 

### [ 12] Backward propagation

Now, it is time to move on to backpropagation (back propagation) in order to minimise the value of the loss.

The weights need to be adjusted to be close to the predicted output. Then, during training the weights will be updated so that the difference is decreasing. One way to do this is to use gradient decreases.



IMG. [12] Example of the use of Gradient Descent

### 

### [13] Applications of Multi-Layer Perceptrons

*Some examples of Applications of Multi-Layer Perceptrons*

*1. Types of Feed-Forward Neural Network Applications*

*2. Brain Modelling*

* *Development, Adult Performance,*
* *Neuropsychology Analysis of Hidden Unit Representations*

*3. Real World Applications*

* *Data Compression - PCA*
* *Time Series Prediction*
* *Character Recognition and What-Where*
* *Autonomous Driving - ALVINN*

### [13] Real World Applications

*1. Airline Marketing Tactician (Beale & Jackson, Sect. 4.13.2)*

*2. Backgammon (Hertz et al., Sect. 6.3)*

*3. Data Compression – PCA (Hertz et al., Sect. 6.3; Bishop, Sect. 8.6) •*

*4. Driving – ALVINN (Hertz et al., Sect. 6.3) •*

*5. ECG Noise Filtering (Beale & Jackson, Sect. 4.13.3)*

*6. Financial Prediction (Beale & Jackson, Sect. 4.13.3; Gurney, Sect. 6.11.2) •*

*7. Hand-written Character Recognition (Hertz et al., Sect. 6.3; Fausett, Sect. 7.4) •*

*8. Pattern Recognition/Computer Vision (Beale & Jackson, Sect. 4.13.5) •*

*9. Protein Secondary Structure (Hertz et al., Sect. 6.3)*

*10. Psychiatric Patient Length of Stay (Gurney, Sect. 6.11.1)*

*11. Sonar Target Recognition (Hertz et al., Sect. 6.3)*

*12. Speech Recognition (Hertz et al., Sect. 6.3)*

*13. Text to Phoneme Mapping (Beale & Jackson, Sect. 4.13.1; Bullinaria, 2011) •*

### 

### 

### Recognize any number

Another use of Multilayer Feedforward Networks is to recognize any number in an image.

To solve this problem we will need a database with different types of numbers as varied as possible (the more varied in shape and size the numbers are, the better it will be able to recognize different formats). The numbers will be obtained from images with the same resolution 8 X 8 pixels.



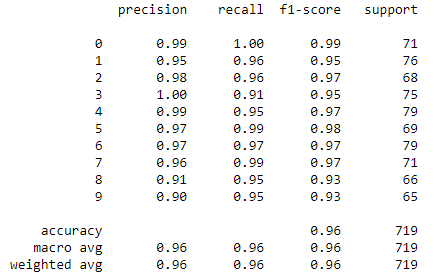
Next we train the network and test it with different parameters.

We can make different reports of the results to see how effective they are

**1 - Classification report**

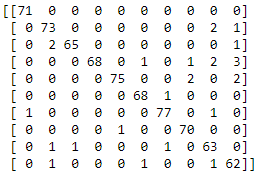
It will measure:

* **precision**: The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.
* **recall**: The recall is intuitively the ability of the classifier to find all the positive samples.
* **f1-score**: a weighted average of the precision and recall, where an 𝐹1F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the 𝐹1
* fupport: The support is the number of occurrences of each class



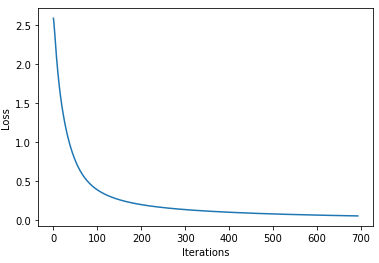
**2 - Confusion matrix**

By definition, a confusion matrix 𝐶 is such that 𝐶𝑖,𝑗 is equal to the number of observations known to be in group 𝑖 but predicted to be in group 𝑗 . Thus an optimal classification result would produce a diagonal matrix.



**3 - Loss curve**

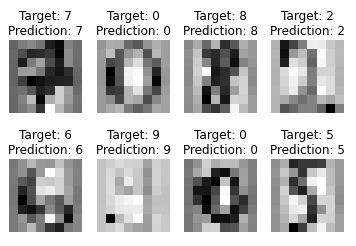
Number of iteration: 694

****

We can observe that from iteration 600 the variation of the loss is so small that the model converges.

**4 - Samples of predictions**

Finally, the target and prediction can be displayed for some images of the data set.



### Classification with the German Traffic Sign Recognition Benchmark.

Another use of Multilayer Feedforward Networks is Classification with the German Traffic Sign Recognition Benchmark.

An automatic road sign recognition system first locates road signs within images captured by an imaging sensor on-board of a vehicle, and then identifies road signs assisting the driver to properly operate the vehicle.

Automated road sign recognition is a difficult task.

The first thing to do is to standardise the data to be used for training.

In this case we have a large database in which the different types of signals are separated and each type has many different shapes.

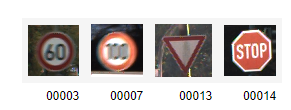
By training the network with so much variety, it will allow the network to recognize new signals or signal variations more efficiently when it is already trained.

**Database**

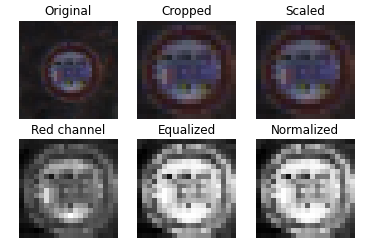
* Single-image, multi-class classification problem
* More than 40 classes
* More than 50,000 images in total
* Large, lifelike database

The training set archive is structured as follows:

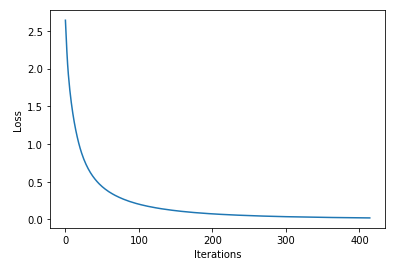
* One directory per class
* Each directory contains one CSV file with annotations ("GT-.csv") and the training images
* Training images are grouped by tracks
* Each track contains 30 images of one single physical traffic sign



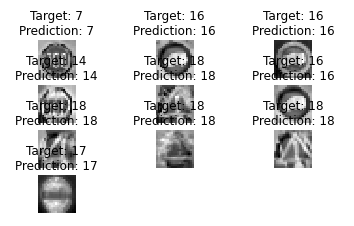
To improve the training of the network, it is sometimes convenient to treat the data. In this case, since the unmatched signals are in the same position in the images, we will crop the images (all with the same format) so that the network focuses on what is important and so that it is not distracted by the colours and focuses only on the shapes, we will convert them to black and white.



**Loss curve**

****

**Result of a different set of tests than the trained one**

****

[[10] https://en.wikipedia.org/wiki/Multilayer\_perceptron](https://en.wikipedia.org/wiki/Multilayer_perceptron)

[[11] https://towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250](https://towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250)

[[12] https://www.goeduhub.com/10389/how-to-train-a-multilayer-perceptron](https://www.goeduhub.com/10389/how-to-train-a-multilayer-perceptron)

[13] <https://www.cs.bham.ac.uk/~jxb/INC/l11.pdf>

### 

## Introducción

El perceptrón multicapa (MLP), a diferencia de los perceptrones más comunes, clasifica datos que no son linealmente separables. Esto se debe a que utilizan una arquitectura más robusta y difícil para aprender modelos de regresión y clasificación para conjuntos de datos difíciles. El perceptrón multicapa (MPL) tiene al menos tres capas de nodos. La primera capa de entrada, la segunda capa oculta y la tercera capa de salida. La segunda y tercera capa de nodos es una neurona que utiliza una función de activación no lineal.

--------

El perceptrón multicapa puede tener otros nombres:

..............

--------------------------------------------------------

Entrenamiento de una red neuronal multicapa

## Ejemplo de funcionamiento

### Propagación hacia delante

Las MPL existen un número arbitrario de capas ocultas situadas entre las capas de entrada como entre las capas de salida.

Para empezar, se asignan los pesos a cada capa de entrada dependiendo de sus características que fluyen hasta las capas ocultas.

A continuación, los pesos se unen con los requisitos de las entradas (estos requisitos pueden ser cualquier cosa como por ejemplo operaciones aritméticas: sumas, restas, multiplicaciones, etc.) para pasar a la función de activación la más utilizada son la funciones sigmoides. El resultado de la capa oculta pasa a ser la entrada de la siguiente capa oculta.

Este proceso se repetirá las veces que hayamos programado, hasta ahora sólo nos hemos movido en dirección hacia adelante.

Esto se conoce como propagación hacia adelante.

Este sistema tiene un problema que error en una neurona es debido a los pesos de sus neuronas anteriores. Para minimizar el error se puede tiene que ajustar el Learning rate.

Learning rate: Es el porcentaje de cambio con el que se actualizan los pesos en cada iteración, en otras palabras, cada que se realiza una iteración en el proceso de entrenamiento se deben actualizar los pesos de la entrada para poder dar cada vez una mejor aproximación. Hay que tener en cuenta que:

Si pequeño hay que iterar muchas épocas

Si es muy grande puede no converger

### Propagación hacia atrás

Ahora, toca pasar a la retropropagación (propagar hacia atrás.) con el fin de minimizar el valor de la pérdida

Se necesita ajustar los pesos para que se acerquen a la salida predicha. Entonces, durante el entrenamiento se actualizaran los pesos de manera que la diferencia sea decreciente. Una forma de hacerlos es usando descensos de gradiente.