Third report delivery

Image recognition of touristic places

Ximena Cervantes Díaz, 276978

Fundamentals of Artificial Intelligence, 281301
December 3rd , 2019

Project advisor: Dr. José Ignacio Nuñez Varela

Abstract:

This project was created with the objective to learn something about Artificial Intelligence applied to some of our everyday activities. At the last two reports, is documented how could we start some image recognition program, which are the most useful algorithms and some experiments with the parameters of those algorithms. This is the final delivery for the Artificial Intelligence class but obviously, it's needed to continue testing this project with others variables and new different images.

For the image recognition, it was necessary to learn and understand a lot of concepts. To make this project is very important to search information at different articles or books, ask to the university teachers which are an experts of this topic, experiment functions and set the parameters to make a test for the goal result.

After choosing the correct OpenCV algorithm (SURF) and making an heuristic for the algorithm which creates a keypoints range, those data have to be saved at almost one file (.txt file at this case) and use this information to make the machine learning. Machine learning is an abstract concept with a lot of ways to make the human-computer teaching.

Next it will be explained those important concepts to understand the final and most important steps of the machine learning, which is a fundamental

concept of Artificial Intelligence. There will be explained the steps for the neural network's creation and the test if the computer could learn about our images or not.

Introduction

Data mining is the method or technique that gets information or patterns from massive-amounts of information. It will search any relationship or patterns that exist in some database. The data processing includes:

Classification: it's organization of information at the given category. It uses a training set where all amount of info are already associated with known category label (supervised classification)

Clustering: To grouping of records, is an unsupervised classification based on the principle of maximizing similarity between objects in same category ad minimizing similarity.

Association: It studies frequency of data. Supports identifies item set, with confidence (condition that an item appear in transaction when another item appears)

At data processing there are numerous ways for gaining knowledge, like:

Decision tree: cluster of relationship organized into tree like structure.

Genetic algorithm: They are category randomized search procedure of adjective search over a wide range of space topology. They can solve problems in parallel, it's a robust tool for data processing.

Neural network: It's a directed graph consisting of nodes with interconnecting synaptic and activation link. Is generated by simulating image intuitive thinking of human on basis of analysis of biological network. It's called too as Artificial Neural Network.

Artificial Neural Networks

Neural networks consists of a large class of different arhitectures[1]. At 1943 Warren McCulloch and Walter Pitts presented the first model of artificial neurons. *"Artificial neural networks are an attempt at modeling the information processing capabilities of nervous systems"* (Rojas R., 1995).

In neural network knowledge representation of surrounding environment is outlined by value taken on by parameters like synaptic weight. An artificial neural network (ANN), is simply known as a "neural network" (NN).
One of the most useful neural networks are the Multilayer Layer Perceptron (MLP) and Radial Basis Function (RBF) networks.
A MLP consists of an input layer, a number of hidden layers and an output layer.
Many algorithms exist for the network parameters, those are called learning or teaching algorithms. The most popular algorithms are Back-propagation and Levenberg-Marquardt.
Back-propagation is a gradient based algorithm, which has many variants. Levenberg-Marquardt is usually more efficient, but needs more computer memory. The procedure of teaching algorithms for multilayer perceptron networks:

1. The structure of the network is first defined. The network activation functions are chosen, the weights and biases are initialized.
2. The parameters associated with the training algorithm (error goal, epochs, etc) are defined.
3. The training algorithm is called.
4. The result is first tested by the output of the neural network with the measured input data. This is compared with the measured outputs.

Back-propagation Neural Network

Back-propagation algorithm looks for the minimum of the error function in weight space using the method of gradient descent (when the algorithm descends along the gradient towards, after the gradient of objective function by backward propagation of the model errors). The combination of weights which minimizes the error function is considered to be a solution learning problem.
Back-propagation has become the most popular method of training neural networks since 1986. One of the reason for this popularity is the simplicity and relative power of the algorithm.

The perceptron learning and the Widrow-Hoff learning rule (directs attention to the input cells that are less error prone) are its precursors; back-propagation can be employed for training nonlinear networks. Rosenblatt employed the back-propagation term in 1962 to generalize the perceptron learning algorithm to the multilayer case. Paul Werbos developed the basic idea in 1974 in a Ph. D named "Beyond Regression" and David Rumelhart, Hinton and Williams developed the idea in spring of 1986, explaining the idea and showing a number of applications that it reached the field of neural networks and artificial intelligence.

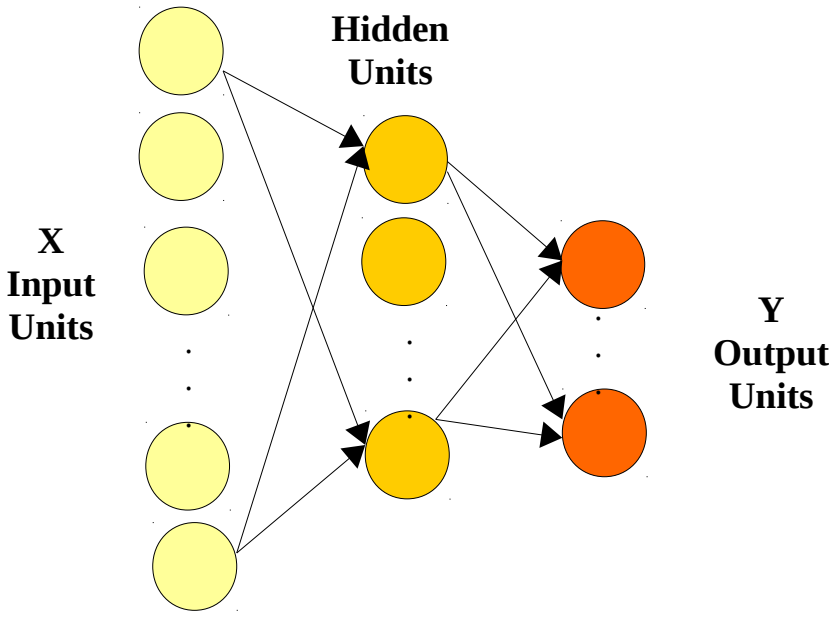

Figure 1: A simple three-layer network

Input signals are processed from one layer to the next through multiple hidden layers and at last, propagate to the output layer. "*Back-propagation model belongs to the error correction learning type.*" (Rojas R, 1995).

We have to use a kind of activation function because the composite function (an outer function with an inner function, in other words, is a function obtained from two given functions) is discontinuous, and therefore the error function too. For this problem there is a lot of activation functions for back-propagation, one of the more popular functions is the sigmoid.

A sigmoid function (Φ) is a "bounded differentiable real function" [6] that is defined for all real input values and that has a positive derivative everywhere. It shows a sufficient degree of smoothness and is also suitable extension of the soft limiting nonlinearities used previously in neural networks.

Back-propagation also focused attention to the existence of "theorems" which guarantee the called universal approximation (a network with a

single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets).
As we can appreciate at the Figure 2, there are three popular activation functions illustrated which are:
hard-limiter function (threshold):

$$\phi(x) = \begin{cases} 1, & x \geq 0 \\ -1(\text{or } 0), & x < 0 \end{cases},$$

(3.1)

logistic function:

$$\phi(x) = \frac{1}{1 + e^{-\beta x}},$$

(3.2)

hyperbolic tangent function (tanh):

$$\phi(x) = \tanh(\beta x).$$

(3.3)

In these functions, β is selected as unity and used to control the steepness of the activation function.

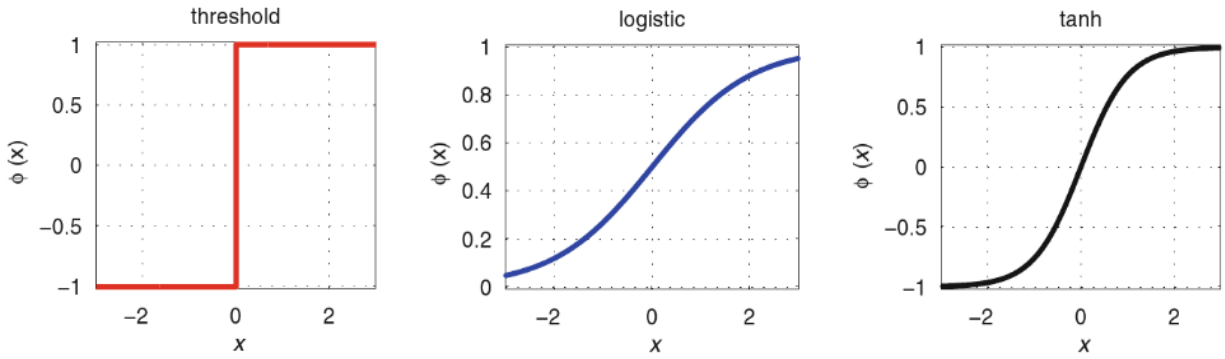

Figure 2: Sigmoidal activation functions[3]

As is mentioned previously, the back-propagation algorithm is used to find a local minimum of the error function. The network is initialized with randomly chosen weights. The gradient of the error function is computed and used to correct the initial weights, we have to compute this gradient recursively.
The training method of a back-propagation network consists of two phases:

Feed-forward

The input vector is introduced from the input layer through hidden layers to the output layer, while the output value of the network is calculated, at same time, the weigh values of the network are fixed.

Feed-back propagation

The error signal is obtained by subtracting the network output value from the expected output value and is the propagated backward through the various hidden layers to the input layer.

The back-propagation algorithm also works correctly for networks with more than one input unit in which several independent variables are involved.

Creating a neural network

To make a neural network we need some software which is a package of all the functions needed and help us to make more easy the neural network implementation.
The MATLAB neural network toolbox is one of these powerful tool. It provides a complete set of functions and a graphical user interface for the design, implementation, visualization and simulation of neural networks. But nothing is easier, MATLAB has a higher license cost, which right know is a economical problem if we want to work this project with lower capital. Octave is the free version of the MATLAB tool. It supports the most commonly used supervised and unsupervised network architectures and a comprehensive set of training and learning functions.
First we have to install the nnet package which is not included automatically to the program in contrast to MATLAB, which loads automatically all the necessary packages to make a neural network. The nnet package can be downloaded on the official Octave website: https://octave.sourceforge.io/nnet/ and easily installed for any operative system (Linux, Windows, Mac OS)
After the package installation, we can create our first MLPN neural network with the command *newff*, which usually is called *net*.

$$net = newff(\underbrace{PR}_{\text{min,max values}}, [\underbrace{S1\ S2 \ldots SNl}_{\text{size of the } i\text{th layer}}], \{\underbrace{TF1\ TF2 \ldots TFNl}_{\text{activation function of } i\text{th layer}}\}, \underbrace{BTF}_{\text{training algorithm}})$$

(4.1)

Where the inputs at the element **(4.1)** are:

R = Number of the elements in input vector
xR = Rx2 matrix of min and max values for R input elements (we can use the min_max() function)
Si = Number of neurons (size) in the *ith* layer, i= 1,…,Nl
Nl = Number of layers

Tfi = Activation function of the *ith* layer, by default is 'tansig' function.
BTF = Network training function, by default is 'trainlm' function.

Training

After initializing the network, the network training is originated using train command. The resulting MLP network can be named "net" or "net1".

$$net1 = train(\underbrace{net}_{\substack{initial\\MLP}}, \underbrace{x}_{\substack{measured\\input\ vector}}, \underbrace{y}_{\substack{measured\\output\ vector}})$$

(4.2)

Testing

To test how well is the result of the train net, we can use the *sim* command. The measured output is y, can now be compared with the output of the network *ytest* to see how good the result is by computing the error difference at each measured point.

$$ytest = sim(\underbrace{net1}_{\substack{final\\MLP}}, \underbrace{x}_{\substack{input\\vector}})$$

(4.3)

Results of testing project

At this project we use 3 different type of buildings (Cathedral, Central Building of the UASLP, and the "Caja del agua"). Those 3 are the classes of the database, where each of the class has 15 images of those building or monument. This database is used to the neural network training.
Otherwise, there is another database of each one of the 3 classes, with 11 images. This database is used for the testing part if we want to know if the neural network is learning something. We hope to get the known answer, that's why is a supervised learning.

The following tables contain the experiment to test the neural network with the same values of the training phase. The goal result is to get at least the most part of results classified correctly.
The classes or types of buildings are:
A = Cathedral
B = Central Building of UASLP
C = Caja del agua

This script count how many columns of the result matrix are of that determinate class and start counting if it corresponds at the result type or not.
Result class:
A = 100
B = 010
C = 001

Once the script gave us the results and the class-counters, we have to make a subtract between the number of all the possible results of that class and the real test result.

This is the first experiments realized, where the number of descriptors for the training part could be changed and the number of hidden neurons too.

N. descriptors training	N. neurons of the hidden layer	Execution time in seconds	N. descriptors testing	N. Trues/ N. descriptors testing	Error
300	1	3.2292	300	A=0/100 B=97/100 C=0/100	A=100/100 B=3/100 C=100/100
300	5	19.047	300	A=100/100 B=98/100 C=100/100	A=0/100 B=2/100 C=0/100
300	10	63.849	300	A=100/100 B=100/100 C=100/100	A=0/100 B=0/100 C=0/100

N. descriptors training	N. neurons of the hidden layer	Execution time in seconds	N. descriptors testing	N. Trues/ N. descriptors testing	Error
900	5	45.687	900	A= 217/300 B= 219/300 C= 275/300	A= 83/300 B= 81/300 C= 15/300
900	10	166.35	900	A= 280/300 B= 289/300 C= 276/300	A= 20/300 B= 11/300 C= 24/300
900	20	694.14	900	A= 300/300 B= 300/300 C= 300/300	A= 0/300 B= 0/300 C= 0/300

N. descriptors training	N. neurons of the hidden layer	Execution time in seconds	N. descriptors testing	N. Trues/ N. descriptors testing	Error percent
1200	10	211.65	1200	A= 327/400 B= 361/400 C= 373/400	A= 73 B= 39 C= 27
1200	15	547.75	1200	A= 384/400 B= 386/400 C= 383/400	A= 16 B= 14 C= 17
1200	20	821.66	1200	A= 398/400 B= 399/400 C= 399/400	A= 2/400 B= 1/400 C= 1/400

With all the descriptors, now is time to test with how many neurons the training and testing will be successful

N. descriptors training	N. neurons of the hidden layer	Execution time in seconds	N. descriptors testing	N. Trues/ N. descriptors testing	Error
1461	30	2 370.0	1461	A = 489/490 B = 509/509 C = 462/462	A= 1/490 B= 0/509 C= 0/509
1461	40	4 453.7	1461	A= 490/490 B= 509/509 C= 462/462	A= 0/490 B= 0/509 C= 0/462
1461	50	7 423.3	1461	A= 490/490 B= 509/509 C= 462/462	A= 0/490 B= 0/509 C= 0/462
1461	60	10 879.0	1461	A= 490/490 B= 509/509 C= 462/462	A= 0/490 B= 0/509 C= 0/462

As we can appreciate, the best results goes depart from the number of 50 hidden neurons. Let's try the same last experiment with all the descriptors, but this time the test will be applied to the test database, not with the training database.

N. descriptors training	N. neurons of the hidden layer	Execution time in seconds	N. descriptors testing	N. Trues/ N. descriptors testing	Error
1461	30	2 370.0	952	A = 55/364 B = 105/320 C = 130/268	A= 309/364 B= 215/320 C= 160/268
1461	40	4 453.7	952	A= 59/364 B= 93/320 C= 82/268	A= 305/364 B= 227/320 C= 208/268
1461	50	7 423.3	952	A= 59/364 B= 115/320 C= 92/268	A= 305/364 B= 205/320 C= 198/268
1461	60	10 879.0	952	A= 58/364 B= 125/320 C= 106/268	A= 306/364 B= 195/320 C= 184/268

This is not the goal result, there is a lot of error for the final results.

Conclusion

For this neural testing, the final result isn't what we expected. It's a curious thing to look the results when we test the neural network with the same training dataset and look the results when we test it with new dataset; these results are not similar: one of the two experiments is almost perfect,with 0.07% of error, and the other one is more than 50% of error. For the first experiment (test with the training dataset) we can observe that the neural network understand the input data, with minimum of confusion. Otherwise, the second experiment is not successful, that means that the neural network is working correctly, but the dataset are confusing or not clear for it. In conclusion, the problem could begin for the data set, maybe the majority of them are similar and that's why the neural networks can't made the distinction of all the test descriptors.

For better results, we need to study more the SURF algorithm to know how to choose the best descriptors and why the algorithm chooses them.
Maybe we can check if the inputs dataset could be different, this experiment was executed with matrices of 64 rows. Or simply, train the neural network with a lot of more datasets, but is not a fact that this would be better than these results.

References

1. Abdi, H., Valentin, D., Edelman, B., & O'Toole, A. J. (1995). More about the difference between men and women: evidence from linear neural networks and the principal-component approach. *Perception*, *24*(5), 539-562.
2. Chauvin, Y., & Rumelhart, D. E. (2013). Back-propagation*: theory, architectures, and applications*. Psychology Press.
3. Du, K. L., & Swamy, M. N. (2013). *Neural networks and statistical learning*. Springer Science & Business Media.
4. Hsieh, W. W. (2009). *Machine learning methods in the environmental sciences: Neural networks and kernels*. Cambridge university press.

5. Koivo, H. N. (2005). Basics using MATLAB Neural Network Toolbox.
6. Li, D., & Du, Y. (2017). *Artificial intelligence with uncertainty*. CRC press.
7. Mira, J., & Sandoval, F. (Eds.). (1995). *From Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks, Malaga-Torremolinos, Spain, June 7-9, 1995: Proceedings* (Vol. 930). Springer Science & Business Media.
8. Rojas, R. (2013). *Neural networks: a systematic introduction*. Springer Science & Business Media.