FREE eBook

LEARNING

neural-network

#neural-network

Table of Contents

About

Chapter 1: Getting started with neural-network

Remarks

Neural networks, in the tech field, are useful for statistic regression, data classification, product recommentation, computer vision, natural language understanding and synthesis, speech to text, text to speech, and many other complex tasks. Neural networks are used in machine learning and in deep learning, they are related to artificial intelligence.

A neural network learns by example, it is meant to be trained with data in, data out, to later be able to predict the output given an input similar to what it was trained on.

Many framework exists for programming, training and using artificial neural network. Here are some of the most known machine-learning frameworks:

- Tensorflow
- Caffe
- Keras
- Theano
- Torch
- DL4J

Examples

Typical workflow of a neural network

The typical workflow of training and using neural networks, regardless of the library used, goes like this:

Training Data

1. Getting the training data: the `X` variable is the input, and the `Y` variable is the output. The simplest thing to do is to learn a logic gate, where `X` is a vector or two numbers and `Y` is a vector of one number. Typically, the input and output values are floats, so if it is words, you might associate each word to a different neuron. You could also directly use characters, then it would use less neurons than to keep a whole dictionary.

Architecture

2. Defining the neural network's architecture: this is done by specifying how the neurons are linked together and with which algorithm the connections between neurons are trained and changed. As an example, processing text is done using recurrent neural networks, which receive a new input at each timestep and where neurons have a reference to their earlier value in time for effective computation purposes. Commonly, layers of neurons are used,

they are generaly stacked one over the other from the inputs to the outputs. The way neurons are connected from a layer to the other varies a lot. Some computer vision architectures uses deep neural networks (with many specialized layers stacked).

Evaluation

3. Next, the neural network is typically evaluated on data it has not been *directly* trained on. This consists of presenting the `X` part of the data to the neural network, then comparing the `Y` it predicts to the real `Y`. many metrics exists to assess the quality of the learning performed.

Improvement

4. It is common to fiddle with the architecture of the neural network again to improve its performance. The neural network must be not too intelligent and not too dumb because both cases yield problems. In the first case, the neural network might be too large for the data, memorizing it perfectly, and it might fail to generalize to new unseen examples. In the second case, if the neural network is too dumb (small), it will fail to learn too.

Real-World Use

5. Using it on new data to predict an output. Indeed, neural networks are quite useful, automatic text translation or responding to a textual question are good examples. One of the techniques used to improve the neural network at this stage is online learning, meaning that if the network can get a constructive feedback on his outputs, it is still possible to continue the learning process. As an example, this *might* be the case of Google Translate that asks users feedback on translations.

Python Neuron class

```
import numpy as np #There is a lot of math in neurons, so use numpy to speed things up in
python; in other languages, use an efficient array type for that language
import random      #Initial neuron weights should be random

class Neuron:

def __init__(self, nbr_inputs, weight_array = None):
    if (weight_array != None): #you might already have a trained neuron, and wish to recreate
it by passing in a weight array h ere
        self.weight_array = weight_array
    else:                      #...but more often, you generate random, small numbers for the
input weights.  DO NOT USE ALL ZEROES, or you increase the odds of getting stuck when learning
        self.weight_array = np.zeros(nbr_inputs+1)
        for el in range(nbr_inputs+1): #+1 to account for bias weight
            self.weight_array[el] = random.uniform((-2.4/nbr_inputs),(2.4/nbr_inputs))
    self.nbr_inputs = nbr_inputs

def neuron_output(self,input_array):
    input_array_with_bias = np.insert(input_array,0,-1)
    weighted_sum = np.dot(input_array_with_bias,self.weight_array)
    #Here we are using a hyperbolic tangent output; there are several output functions which
could be used, with different max and min values and shapes
    self.output = 1.716 * np.tanh(0.67*weighted_sum)
    return self.output
```

Java Encog engine

Encog is an easy to use java neural network engine

```
public static double XOR_INPUT[][] = { { 0.0, 0.0 }, { 1.0, 0.0 },
        { 0.0, 1.0 }, { 1.0, 1.0 } };

public static double XOR_IDEAL[][] = { { 0.0 }, { 1.0 }, { 1.0 }, { 0.0 } };

public static void main(final String args[]) {

    // create a neural network, without using a factory
    BasicNetwork network = new BasicNetwork();
    network.addLayer(new BasicLayer(null,true,2));
    network.addLayer(new BasicLayer(new ActivationSigmoid(),true,3));
    network.addLayer(new BasicLayer(new ActivationSigmoid(),false,1));
    network.getStructure().finalizeStructure();
    network.reset();

    // create training data
    MLDataSet trainingSet = new BasicMLDataSet(XOR_INPUT, XOR_IDEAL);

    // train the neural network
    final ResilientPropagation train = new ResilientPropagation(network, trainingSet);

    int epoch = 1;

    do {
        train.iteration();
        System.out.println("Epoch #" + epoch + " Error:" + train.getError());
        epoch++;
    } while(train.getError() > 0.01);
    train.finishTraining();

    // test the neural network
    System.out.println("Neural Network Results:");
    for(MLDataPair pair: trainingSet ) {
        final MLData output = network.compute(pair.getInput());
        System.out.println(pair.getInput().getData(0) + "," + pair.getInput().getData(1)
                + ", actual=" + output.getData(0) + ",ideal=" + pair.getIdeal().getData(0));
    }

    Encog.getInstance().shutdown();
}
```

This is the 'Hello World' equivalent of neural networks.

Read Getting started with neural-network online: https://riptutorial.com/neural-network/topic/3450/getting-started-with-neural-network

Credits

S. No	Chapters	Contributors
1	Getting started with neural-network	Community, Guillaume Chevalier, ItamarG3, kame, rossdavidh, Rugnir