TensorFlow

Sebastian De Ro

Version 1.0, 2018-03-18

Table of Contents

Foreword

TensorFlow is a machine learning library by Google, that abstracts much of the mathematical background of machine learning, by providing configurable implementations of commonly found algorithms.

Even though it abstracts much of the needed mathematical knowledge for creating algorithms that provide any useful functionality, there is still quite a learning curve for being able to use this library productively.

A knowledge of the different algorithms that are used in data science aswell as hyper parameter tuning is needed. A knowledge of statistics is also very useful. It is also very important that you know how to clean your data and what defines useful data, so that you can feed your TensorFlow setups a good training and testing dataset. The best algorithms still need good data to be able to provide any useful result.

In this paper we will look at all the different parts that you need to know to be able to create your own deep learning setups with TensorFlow and why you would even want to do that.

Let's start by answering the following simple questions: ***What is it? Why do I need it? How can I use it?***

About TensorFlow

What is TensorFlow?

TensorFlow is a computation library written by Google. It represents deep learning computations with data flow graphs. In the words of Google on their website:

> TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them.
>
> — www.tensorflow.org

Why do I need it?

In deep learning especially, doing it yourself results in a lot of repetitive code. In Python, the code base will run quite slowly which is not ideal when you are training on big datasets, as this means that your algorithm will be repeating the same task many times to train a set of weights using a training dataset. Repetition calls for good performance that Python is just not able to deliver.

Machine Learning in Python

We can circumvent this by using libraries like **Numpy**, that import compiled C and C++ libraries to

python and offload performance intensive tasks to these compiled libraries. This allows for safe and scientific computation in python.

But even with these libraries reinventing the ***wheel***(get it?) is not really optimal. Therefore a group of great people created a library that includes many useful machine learning algorithms, like clustering, regression, classification and others, like naive-bayes. This library is calles **SciKit Learn**.

So now we know of two libraries that can save us hundreds of lines of code and make our code more efficient. But one we are still missing is **Pandas**. Pandas is a data structure and analysis library that we can use to clean our dataset and analyze it.

This is all you need to create basic machine learning algorithms that can already do some cool stuff. But when you want to explore deep learning it gets more complicated. Deep learning is a part of machine learning that uses Artificial Neural Networks(*ANNs*) to simulate a biological learning approach.

How do Feed Forward ANNs work?

ANNs are made up of nodes and connections. The nodes add up all incoming connections and when it reaches a certain treshhold, the sum is used as the output of the node. The incoming connections all have a weight assigned to them which the input is multiplied with.

These weights are what makes ANNs “smart”, as they are adjusted to provide the desired output from the node.

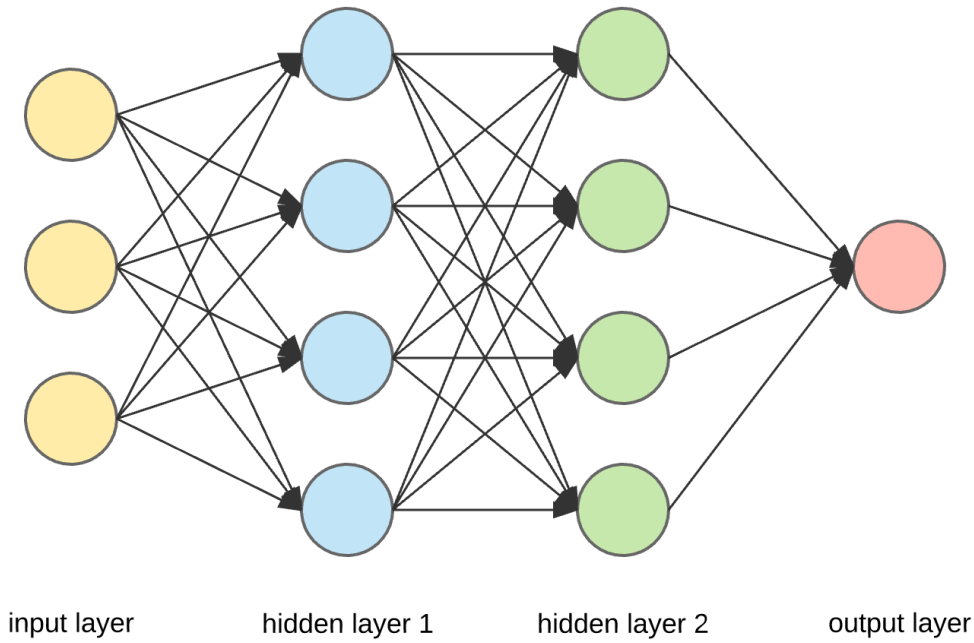

Figure 1: ANNs

Back propagation

Back propagation is the fancy term for adjusting the weights through the whole network. This is done by predicting a value and then calculating the error between the prediction and the expected value. Then the Feed Forward ANN is walked through backwards by the algorithm, comparing every result with the expected and subtracting the error. This is done many times with a big training dataset.

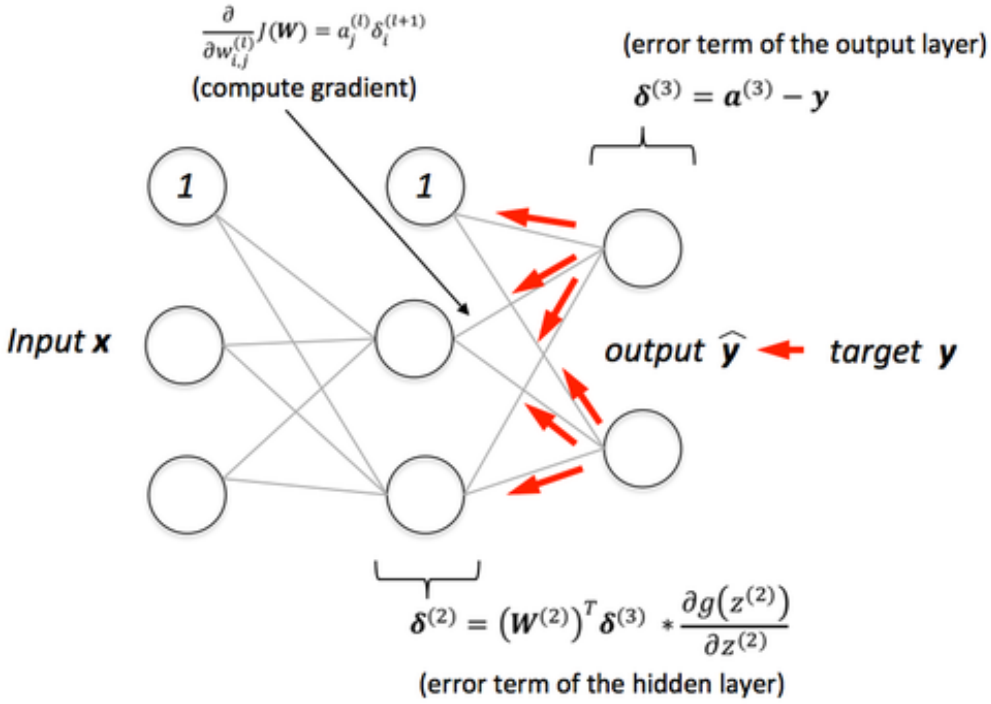

Figure 2: Backpropagation

Hyper Parameters

Learning Rate

- is a multiplier that is multiplied with the error to form the substracted gradient
- is used to prevent overfitting the dataset

Iterations

- is, as the name implies, how often to train on the dataset. The more often this is done the better the prediction
- a too high number of iterations can again cause overfitting of the dataset

TIP	Hyper parameters are parameters that change the behaviour of the algorithm

TIP	Overfitting is a term used to describe the algorithm being trained so well to a specific dataset that it doesnt perform well on new data

Deep Learning

What differenciates deep learning from standard neural networks is that deep learning, as the name implies, uses a deep architecture of multiple layers of neural nodes.