Understanding & Developing Artificial Neural Network with Objective-C and Python

Deerfield Academy Advance Computer Science Research

Feb 11, 2016

Neil Nie

Abstract:

Machine learning is a subset of artificial intelligence that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can change when exposed to new data.

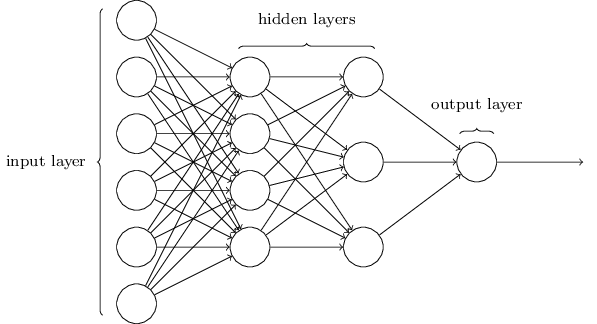
There are many types of learning algorithms, from Support Vector Machine[[1]](#footnote-1) to Artificial Neural Networks[[2]](#footnote-2). The purpose of them are similar, they can all be used to classify complex data, such as images and DNA samples.

There are two types of machine learning, one is supervised machine learning[[3]](#footnote-3). Supervised learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal).

The second kind is unsupervised machine learning. Unsupervised machine learning is inferring a function to describe hidden structure from "unlabeled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabeled, there is no objective evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

Machine learning algorithms, even the most basics one are very powerful and can bring software to a new level. They can give the computer the power to learn from past data and make decision. Many apps today use some kinds of machine learning, for example, handwriting recognition, speech recognition. The most significant strength of machine learning, specifically neural network is that it can learn completely different data without changing any line of code.

Introduction:

In this research, we will mainly focus on artificial neural networks and supervised machine learning. 

You will discover that machine learning is beautiful and can be very simple. We will be able to implement a simple algorithm that can recognize hand written digits in less than a hundred lines of code. Only a small amount of mathematical proves and equations will be covered. The paper will cover the principles behind designing, building, and debugging an artificial neural network. The projects will be written mainly in Objective-C or Python. You can find the resources on [Github](https://github.com/NeilNie/Neural-Network-Research).

The first example we will discuss is to use the neural network mimic a XOR gate. The logic gate works like the following: there are two inputs, in binary form, if the inputs are the same, output one, if they are different, output zero.

We have decided that the network will have three layers, one inputs, which has two nodes, one hidden layer that has three nodes, one output layer with one node.

Forward Feed:

The neural network will begin be taking in some inputs and making some predictions based on the inputs and the weights between nodes. We can think of it like a one by two matrix. There will be a weight connecting every input layer node to every output layer node, therefore, there are six weights between the input layer and the second layer. The matrix calculation should yield us some result

If we give the matrices some names, call inputs *x*, weights *w(l)* and output *z(l).* In our case, *l* indicate the layer, for example, the first layer weights are *w(1)*. *z(l)* represent the output matrix with layer *l*, in this case, the output z is the hidden layer output.

Afterwards, we have to apply an activation function[[4]](#footnote-4). The activation function that I used in this research and this paper will be the sigmoid function. This is a complete cycle, there is one more layer to go in order to yield a result. Note, in a multilayer neural network, this process will be repeated until we yield some output. In our case, we only have to do this twice. The activation function is shown as below.

By applying the same process as before, this time, the inputs will be *a(2) then, z(3)* will be our final result. This process is known as forward feed. It’s relatively straightforward.

After we yield some result, we can more on and look at how far off we are from the expected result. We need some methods to quantify this and so adjustments to the network to minimize error.

Gradient Descent:

Now the neural network can make calculation/predictions, however, the result is far from desired. In almost all learning algorithms, the input data cannot be altered, therefore the x term is constant in equation one. In order to change the output *z* the only option is to change the weights *w*.

First of all, we have to come up with ways to quantify the errors.

J is the error this case, which equals to the sum of all the differences between calculated result and actual result squared and times one half. We can take advantage of the equation derived above and substitute for some of the variable.

Here we have it, a way to quantify the errors in the neural network. This function will be referred to as the cost function. Now, we have to solve the problem, how do we minimize J, will brute force work? It turns out, no, because in a three-node neural network we have to compute more than a million possible weights, which will be gruesome.

We can think of the equation above as a function of error in terms of all possible weights. There will be one set of weights that will bring the cost to the lowest. Then, this becomes a minimization problem.

The best way to minimization the cost is to use gradient descent, a very fast and classic way to solve problems like this. In fact, gradient descent is widely used in math, image process and machine learning.

Gradient descent is a first-order iterative optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the *negative* of the [gradient](https://en.wikipedia.org/wiki/Gradient) (or of the approximate gradient) of the function at the current point.

Gradient descent is also known as steepest descent, or the method of steepest descent. Gradient descent should not be confused with the [method of steepest descent](https://en.wikipedia.org/wiki/Method_of_steepest_descent) for approximating integrals.

There are limitations to this method. First of all, what if we are stuck in a local minimum, our goal is to find the global minimum for the cost function. In another work, this method will not work for a non-concave function. In fact, this problem is solved in equation (3), by squaring the difference in , we transformed this function to a concave function. Even though there are tens of thousands of variables (dimension), the function will still be concave, thus, we can apply gradient descent to find the global minimum.

There are other types and variation of gradient descent as well. One of the most commonly used one is Stochastic gradient descent (SGD), also known as incremental gradient descent, is a [stochastic approximation](https://en.wikipedia.org/wiki/Stochastic_approximation) of the [gradient descent optimization](https://en.wikipedia.org/wiki/Gradient_descent_optimization) [method](https://en.wikipedia.org/wiki/Iterative_method) for minimizing an [objective function](https://en.wikipedia.org/wiki/Objective_function) that is written as a sum of differentiable functions. In other words, SGD tries to find minima or maxima by iteration. [[5]](#footnote-5)

Back propagation:

Gradient descent[[6]](#footnote-6) is one of the most common methods for minimizing the errors. The process can be seen as a ball rolling down a hill[[7]](#footnote-7) and trying to find the lowest point. Note that actual physics doesn’t apply here and we will define our own movement of the ball.

Overfitting and Regularization

1. https://en.wikipedia.org/wiki/Support\_vector\_machine [↑](#footnote-ref-1)
2. https://en.wikipedia.org/wiki/Artificial\_neural\_network [↑](#footnote-ref-2)
3. Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar (2012) *Foundations of Machine Learning*, The MIT Press [ISBN 9780262018258](https://en.wikipedia.org/wiki/Special:BookSources/9780262018258). [↑](#footnote-ref-3)
4. https://en.wikipedia.org/wiki/Activation\_function [↑](#footnote-ref-4)
5. <http://www.mit.edu/~dimitrib/Incremental_Survey_LIDS.pdf> Dimitri P. Bertsekas Report LIDS - 2848 [↑](#footnote-ref-5)
6. <http://neuralnetworksanddeeplearning.com/chap2.html> Neural Network and Deep Learning Michael Nielson 2016 [↑](#footnote-ref-6)
7. # <https://iamtrask.github.io/2015/07/27/python-network-part2/> by Andrew Trask

   [↑](#footnote-ref-7)