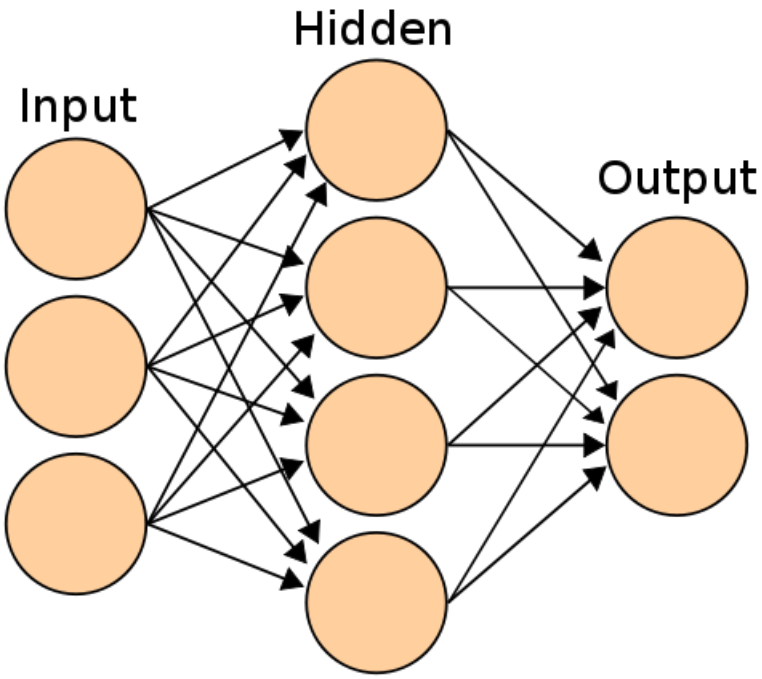
MLPNeuralNet

Fast multilayer perceptron (http://en.wikipedia.org/wiki/Multilayer_perceptron) neural network library for iOS and Mac OS X. MLPNeuralNet predicts new examples by trained neural network. It is built on top of the Apple's Accelerate Framework (https://developer.apple.com/library/ios/documentation/Accelerate/Reference/AccelerateFWRef/_index.html), using vectorized operations and hardware acceleration if available.



Why to choose it?

Imagine that you created a prediction model in Matlab (Python or R) and want to use it in iOS app. If that's the case, MLPNeuralNet is exactly what you need. It is designed to load and run models in forward propagation (http://en.wikipedia.org/wiki/Backpropagation#Phase_1:_Propagation) mode only.

Features:

- classification (http://en.wikipedia.org/wiki/Binary_classification), multiclass classification (http://en.wikipedia.org/wiki/Multiclass_classification) and regression output;
- vectorized implementaion;
- works with double precision;
- multiple hidden layers or none (in that case it's same as logistic/linear regression)

Quick Example

Let's deploy a model for the AND function (conjunction (http://en.wikipedia.org/wiki/Logical_conjunction)) that works as follows (of course in real world you don't have to use neural net for this :)

X1	X2	Y
0	0	0
1	0	0
0	1	0
1	1	1

Our model has the following weights and network configuration:

```
// Use designated initializer to pass network configuration and weights
// to the model. Note: you don't need to specify biased units (+1 above) in the
// configuration.
NSArray *netConfig = @[@2, @1];
double wts[] = {-30, 20, 20};
NSData *weights = [NSData dataWithBytes:wts length:sizeof(wts)];

MLPNeuralNet *model = [[MLPNeuralNet alloc] initWithLayerConfig:netConfig
                                                        weights:weights
                                                     outputMode:MLPClassification];
// Predict output of the model for new sample
double sample[] = {0, 1};
NSData * vector = [NSData dataWithBytes:sample length:sizeof(sample)];
NSMutableData * prediction = [NSMutableData dataWithLength:sizeof(double)];
[model predictByFeatureVector:vector intoPredictionVector:prediction];
```

```
double * assessment = (double *)prediction.bytes;
NSLog(@"Model assessment is %f", assessment[0]);
```

Getting started

This instruction describes on how to install MLPNeuralNet using the CocoaPods (http://cocoapods.org/). It is written for Xcode 5, using the iOS 7 SDK. If you are familiar with 3rd-party library management, just clone MLPNeuralNet repo (https://github.com/nikolaypavlov/MLPNeuralNet) on Github and import it to XCode directly as a subproject.

Step 1. Install CocoaPods

CocoaPods (http://cocoapods.org/) is a dependency manager for Objective-C. Installing it is as easy as running the following commands in the terminal:

```
$ sudo gem install cocoapods
$ pod setup
```

Step 2. Create Podfile

List MLPNeuralNet as a dependenciy in a text file named Podfile in your Xcode project directory:

```
platform :ios, '7.0'
pod 'MLPNeuralNet', '~> 1.0.0'
```

Step 3. Install MLPNeuralNet

Now you can install the dependencies in your project:

```
$ pod install
```

Make sure to always open the Xcode workspace (.xcworkspace) instead of the project file when building your project:

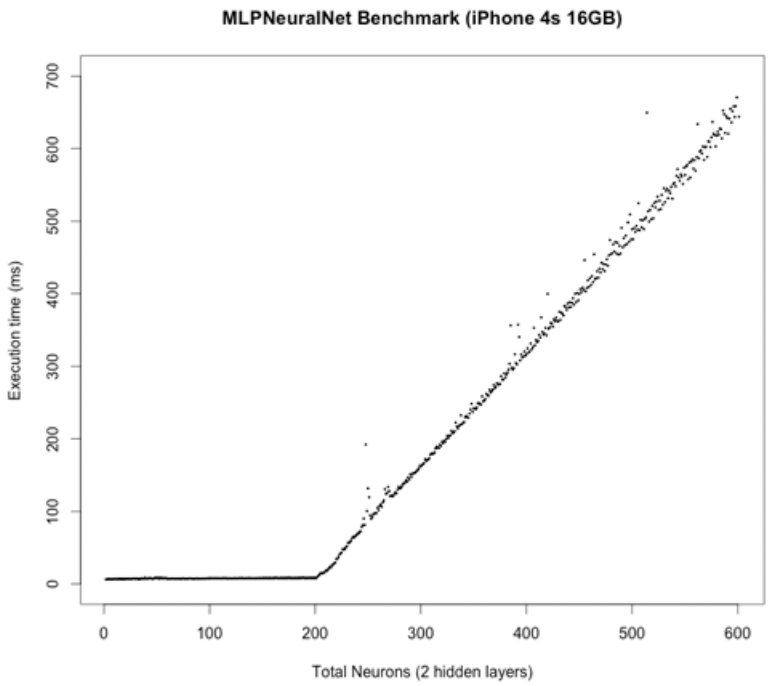
```
$ open App.xcworkspace
```

Step 4. Import MLPNeuralNet.h

`#import "MLPNeuralNet.h"` to start working on your model. That's it!

Performance benchmark

In this test the neural net is grown layer by layer from `1 -> 1` configuration to `200 -> 200 -> 200 -> 1`. At each step the output is calculated and benchmarked using random input vector and random weights. Total number of weights grows from 2 to 80601 accordingly. I understand the test is quite synthetic, but I hope it illustrates the performance. I will be happy if you can propose better one :)



Unit Tests

MLPNeuralNet includes a suite of unit tests in the MLPNeuralNetTests subdirectory. You can execute them via the "MLPNeuralNet" scheme within Xcode.

Credits

- MLPNeuralNet was inspired by Andrew Ng's course on Machine Learning (https://www.coursera.org/course/ml).
- Neural Network image was taken from Wikipedia Commons (http://en.wikipedia.org/wiki/File:Artificial_neural_network.svg)

Contact

Maintainer: Mykola Pavlov (http://github.com/nikolaypavlov/) (me@nikolaypavlov.com).

Please let me know on how you use MLPNeuralNet for some real world problems.

License

> Written with StackEdit (http://benweet.github.io/stackedit/).