Darwin Trader

ECE4122 Final Report

Rodrigo Alarcón
Alan Bannon

Introduction and aims

This project aims to develop a piece of software to provide buy/sell decisions for various cryptocurrencies. A stretch goal was to provide a real-time output and included real-time data from multiple platforms, including social media and NYSE price data.

The genetic algorithm

A genetic algorithm is particularly suited to a prediction problem such as the one this project tackles, as the relations between the inputs data and the desired output result doesn't have to be known. While the same can be said of many machine learning techniques, the genetic algorithm was additionally chosen as its structure lends itself to the application of many techniques taught in ECE4122.

The genetic algorithm is a machine learning technique whereby the repeated generation, evaluation, selection, and mutation of potential solutions is used to derive an optimised solution to a problem. Each potential solution, or "genome", is formed of "genes" (see Figure 1, below) that dictate its behaviour.

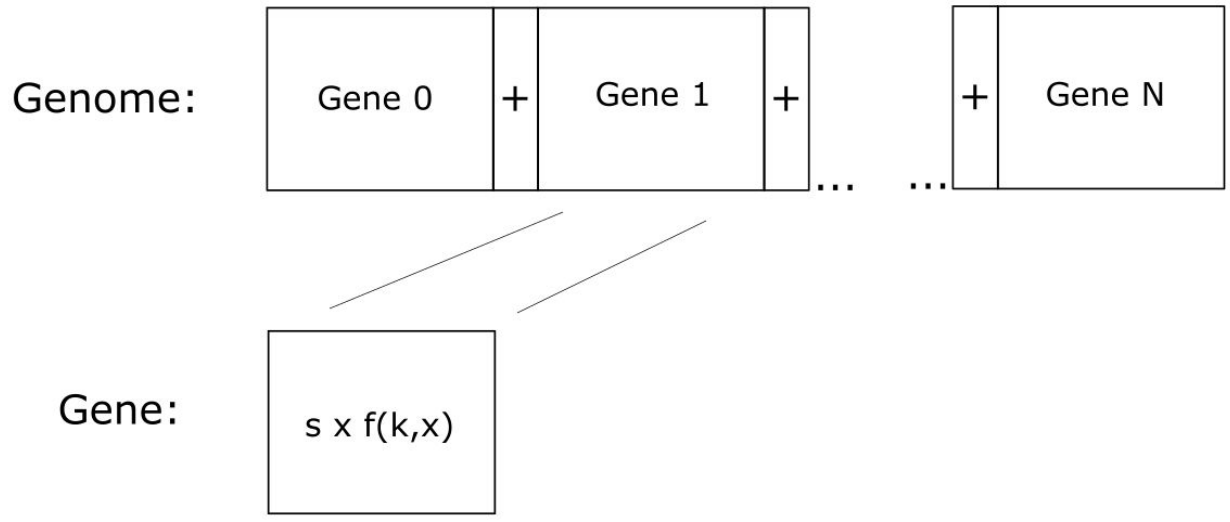

Figure 1: Basic structure of a genome and gene. Gene composed of sign (s), and a function of a scaling factor (k), and an input variable (x).

In this implementation, to develop an optimised solution, a population of genomes is generated randomly, the genomes are evaluated against known training data, a certain proportion of the best performing genomes are taken as the “seeds” of the next population and the rest of the population is discarded (culled), the seeds are multiplied and mutated to form a new population, and the process repeats. This training cycle ends after a number of cycles that depends on the rate of improvement of the error from the known training data..

At each cycle the degree to which new genomes are mutated is determined by a counter that decreases as the mutation cycles continue; the more cycles that have already taken place, the lower the degree of mutation. There is also a breakout facility, where if the best-case error has not improved for 5 cycles, the mutation level increases to attempt to prompt a mutation to a better solution, and thus avoid a potential local minima. For a full diagram of this process see Figure 2.

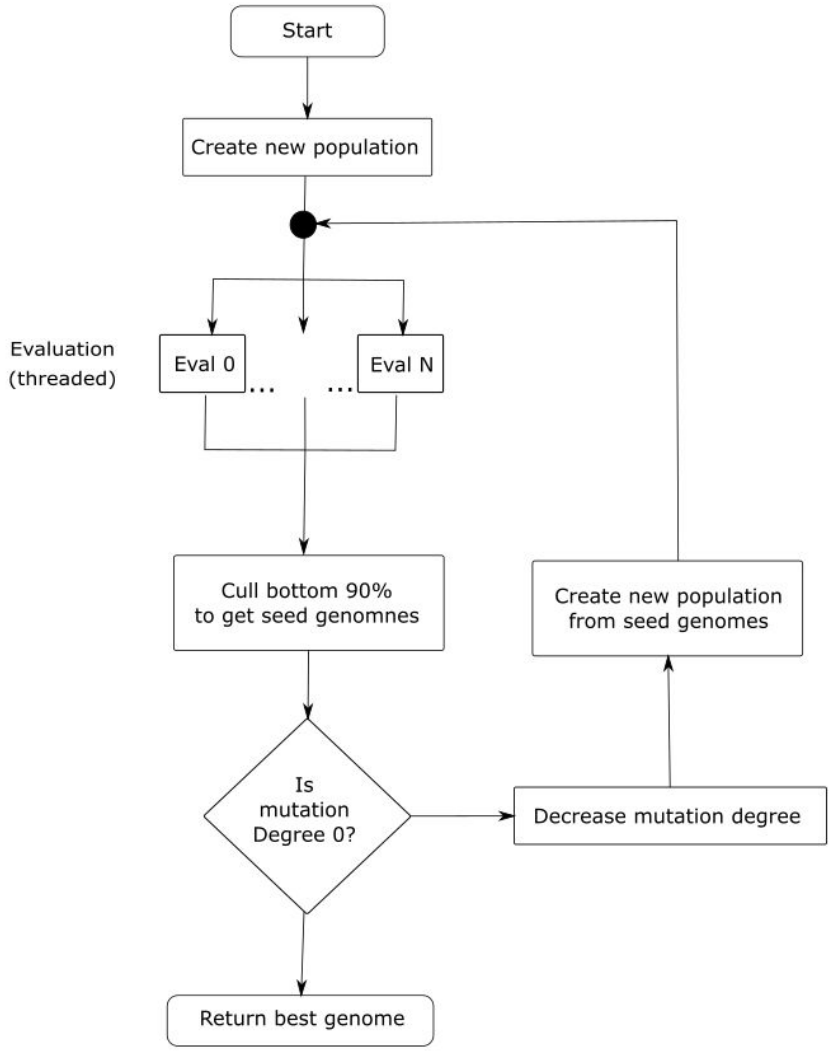

Figure 2: Training process flow. Note that breakouts are not shown.

To test the whole algorithm implementation with controlled data, a set of functions were computed in Matlab, and the results used as inputs to the algorithm (see Figure 3, below).

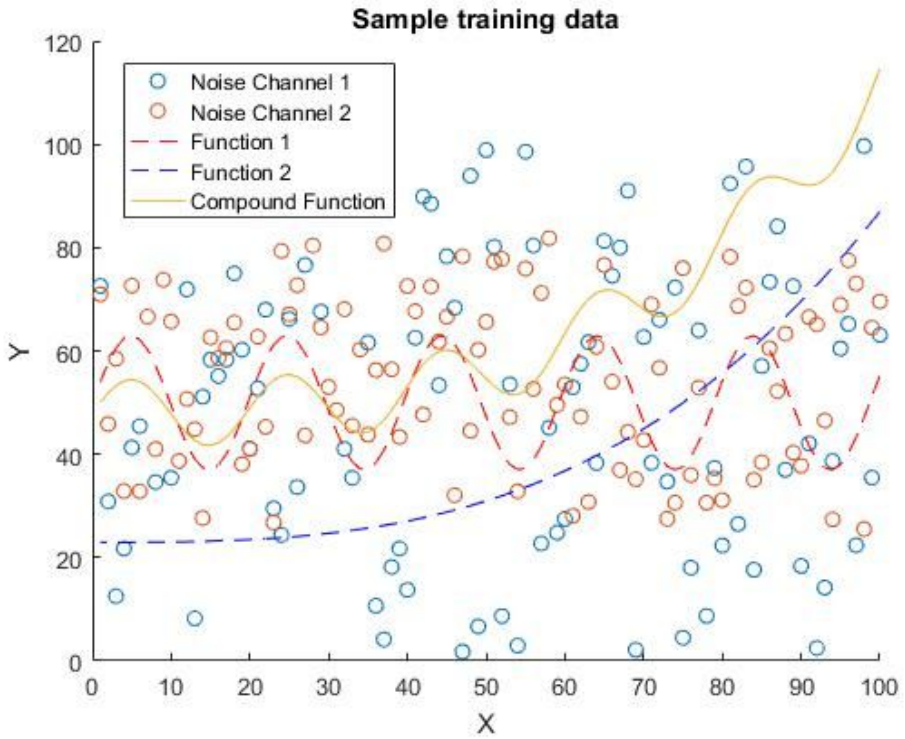

Figure 3: Training data showing two noise channels, two functions, and a compound function.

Using this the algorithm was refined and checked. The final version produced genomes that produced 45-55% accuracy on the noise channels (as expected; it's noise), and 75-95% accuracy on functions 1-2 and the compound function.

Note that in this case the data being used to train the algorithm serves as training data, evaluation data, and verification data. Training the data this way leaves the possibility of overfitting the solution, but for a quick verification is acceptable. .

Real-time data

The idea behind the algorithm was to be able to analyze real-time data from social media sites. An increase in the interest/popularity of a given cryptocurrency can potentially be correlated to an increase in value of said cryptocurrency. Twitter was picked as the social media site to analyze due to its high volume of real-time data output in the form of tweets. Special keywords or hashtags would be searched for in recent tweets. The stream output of tweets containing hashtags of a cryptocurrency's Ticker Symbol would be used as inputs for the algorithm.
Twitter's REST API can query the latest 100 tweets that contain a certain keyword, and will return a JSON-formatted response for all these tweets. The C++ library Twitcurl (C++ twitter API library based on cURL) would be used to obtain GET requests from Twitter. However, extracting meaningful data would prove challenging, as the extraction of number of tweets posted within a specific time limit while only querying 100 tweets at once would not really provide meaningful data in a timely manner.
The solution is to use Twitter's Streaming API, which can be used to filter all real-time tweets for certain keywords, and return a JSON-formatted response containing the information of each tweet. The number of times the Ticker Symbol for each currency is used in a span of 30 seconds would be used as input data for the genetic algorithm.

Historical data

Historical daily cryptocurrency price data was obtained from coinmetrics.io and formatted into a text file that could be loaded into the program. Here we split the data to provide a separate verification dataset, however we're still using the same data for both training and evaluation. Again, this is acceptable to show the algorithm working, and necessary as the amount of data available is limited.

Results

Multiple runs of the algorithm were performed on each of the cryptocurrencies, and the results averaged and displayed in Figure 4, below.

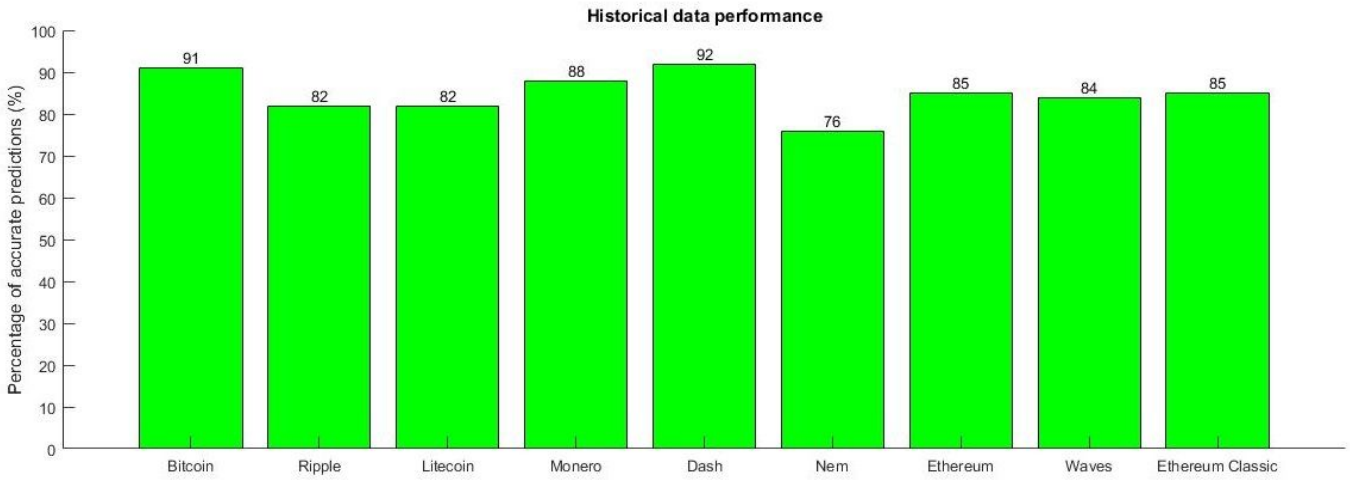

Figure 4: bar-graph of average performance on the historical data set.

For each run the algorithm was trained and evaluated on half the data, then verified on the other half, and the results from this last test graphed in Figure 4. The very high prediction accuracies we see are caused by the fact that the data in consideration is over a long period of time and has low granularity (6 months of predictions on data with 1 day intervals), so large trends are easy for the algorithm to identify and predict. In particular, Bitcoin and Dash had very stable trends over the period, whereas Nem changed behaviour slightly just after the beginning of the verification data.

The Twitter Streaming API is not supported by Twitcurl. Given time constraints, creating a cURL based application from scratch would not be feasible, so the JavaScript library Hashtag-Count was used instead. This library counts hashtag occurrences over time at a provided interval using Twitter's Streaming API. While Hashtag-Count can count the occurrences of any given hashtag, the relatively low count (<20 per 30 secs) of Ticker Symbol Tweets meant data collection for over four hours would be necessary to obtain meaningful input (of ~500 samples) data for the genetic algorithm to provide meaningful results. A small sample of these results were recorded. A future implementation of the Darwin Trader would either integrate Hashtag-Count or replicate it in C++ to be able to feed real-time data to the genetic algorithm.

A different approach was considered in order to collect actual value for each cryptocurrency from a financial trading site, specifically, using Alpha Vantage API and wrapper AvapiCpp, which can also provide real-time cryptocurrency values. However a similar limitation was found, since the fastest refresh rate for cryptocurrency values is of 1 minute (as is with all currency exchange services with available APIs) . Obtaining a meaningful set of data would require the algorithms to run for several hours.

Compiling and running the code

All the code for the genetic algorithm demonstration code should compile from the main directory with a standard make command. No external libraries or dependencies are required.

Once running, the code presents options to run on the sample training data (as shown in Figure 1) targeting the compound function, or the historical cryptocurrency data with the option to target a currency of choice. In both cases the algorithm runs and trains a series of populations to the data, takes the outputs of the training, and applies it to get a prediction and a prediction accuracy. NB: The genetic algorithm is heuristic, so will produce different results for each run.

Running the Javascript code requires NodeJS to be able to run unlimited.js (for stream of results every time interval) or limited.js (for a specific amount of time). These are simply called from the NodeJS command line with the command “node unlimited.js” or “node limited.js”

Individual contributions

Rodrigo Alarcón worked on:
Data gathering and handling, researched libraries and APIs to gather streams of data from social media sites or financial trading websites.

Alan Bannon worked on:
Genetic algorithm implementation, gene and genome design, algorithm training via iterative mutation/culling, loading historical data, demonstration menu design and user-interaction.

Code and additional files for the project are available on github:
https://github.com/abannon6601/Darwin_trader