University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

AAGFA: Automated ANFIS and GA-Based Forex Agent

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Ariel Kenneth Ampol

June 2015

Permission is given for the following people to have access to this SP:

Available to the general public	No
Available only after consultation with author/SP adviser	Yes
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "AAGFA: Automated ANFIS and GA-Based Forex Agent" prepared and submitted by Ariel Kenneth Ampol in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Perlita E. Gasmen, M.Sc. (*candidate*)
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Gregorio B. Baes, Ph.D. (*candidate*)		
2. Avegail D. Carpio, M.Sc.		
3. Richard Bryann L. Chua, M.Sc.		
4. Ma. Sheila A. Magboo, M.Sc.		
5. Vincent Peter C. Magboo, M.D., M.Sc.		
6. Bernie B. Terrado, M.Sc. (*candidate*)		

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Ma. Sheila A. Magboo, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

Alex C. Gonzaga, Ph.D., Dr.Eng.
Dean
College of Arts and Sciences

Abstract

With Forex as the largest and most liquid financial market, the practice of algorithmic trading has become of interest in the market, as well as in research. This study explores the use of the Adaptive Neuro-Fuzzy Inference System as a predictor, combined with the Non-Dominated Sorting Genetic Algorithm II for trade timing to create a smart autonomous Forex trading agent that produces sizable profits. Upon performing a backtest, the agent was shown to be able to garner approximately $80 in profit in a span of two months and nearly $500 in profit for a one-year period. Empirical evidence is also provided that the trading agent running live is able to open trades which profitably closed.

Keywords: ForEx, foreign exchange, exchange rates, algorithmic trading, ANFIS, neuro-fuzzy, genetic algorithm, NSGA-II

Contents

List of Figures

List of Tables

I. Introduction

A. Background of the Study

The Foreign exchange market is the largest and most liquid financial market. [1, 2] A Forex transaction consists of a simultaneous buying of a certain amount of one currency in exchange for another. At the core of ForEx trading are exchange rates and timing. Profiting from Forex trading, then, relies on exploiting the movement and volatility of exchange prices to work in favor of the trader.

Exchange rates are affected by many factors. Thus, exchange rate prediction is traditionally done manually by *fundamental analysis* - taking into account various economic and financial indicators. A viable alternative for short-term predictions is *technical analysis* [3] which uses technical (numerical) indicators. These technical indicators are applied to price and volume data to indicate market direction and volatility, mostly in the short-term. [4]

These short-term strategies are especially applicable to Forex trading, in that most Forex trading is done intraday, in granularity of hours or minutes. Some career traders, investment banks, and hedge funds even trade in the order of seconds or even sub-second. This practice is called High Frequency Trading. [5]

However, investing in Forex is also attractive to amateur investors because of the low capital barrier, high liquidity and leverage. It is possible to open a Forex brokerage account for as little as $100. High liquidity means that the security, in this case currency, can be readily sold at cost. Meanwhile, leverage refers to the means of multiplying profits and losses, usually in the form of borrowed capital. U.S. regulations allow brokers to lend at a 50:1 ratio, meaning an account with $1000 deposit can trade up to $50,000. However, most international brokers can allow leverage of up to 400:1. [6]

One of the reputable online Forex vendors is Oanda (http://oanda.com), chosen for use in this study because of its well-documented, complete RESTful Web API (Representational State Transfer), reputation as a reliable broker, custom trade sizing, and low spreads or commissions. Additionally, their REST HTTP API is freely available for use even for Demo account users, making it suitable for research and testing environments.

In fact, algorithmic trading—the trading of securities based on the buy/sell decision of computer algorithms—has become an established discipline and career which traders undertake. Programs which implement algorithmic trading strategies automatically or semi-automatically are called *smart* or *autonomous trading agents*. Entire books such as [7, 8, 9] have been written on the subject and numerous websites provide information on how to do algorithmic investing. Bell and Gana [10] point out how the financial community has taken notice the effects of algorithmic trading on the market. This is also practically evidenced by the wealth of online discussion regarding trading of securities using automated Expert Advisors on trading platforms such as MetaTrader (http://www.metaquotes.net).

An online service called Quantopian (https://www.quantopian.com) provides an integrated environment which allows a user to deploy algorithmic investment strategies in the cloud and link their programs to their online brokerage accounts. Algorithmic trading packages—such as Zipline (http://www.zipline.io) and PyAlgoTrade (http://gbeced.github.io/pyalgotrade/)—also exist for programming languages like Python and MATLAB.

Found in literature are smart agents or autonomous agents which are programs that can decide for and perform trades on behalf of the user algorithmically. Barbosa [11] presented an autonomous Forex trading agent which traded the USD/JPY pair in a 6-hour timeframe. Their agent features three modules named *Intuition*, *A Posteriori Knowledge*, and *A Priori Knowledge*. The agent resulted turned out with 54.41% accuracy and a profit of 60% over 14 months.

Another smart agent, utilizing the Adaptive Neuro-Fuzzy Inference System (ANFIS) and Genetic Algorithm, was proposed by Alrefaie et. al [6] in 2013. It utilizes the ANFIS as a prediction tool, in combination with the Non-Dominated Sorting Genetic Algorithm II to determine trade timing and decisions. Running over a 30-minute timeframe, Alrefaie's agent achieved an accuracy of 82.12% vs. Barbosa's 54.41%, and a profit of approximately 120% over one year of trading.

B. Statement of the Problem

Algorithmic trading strategies would have to have two basic abilities in order to make profit. First, it must be able to predict future rates. Second, it must be able to provide actionable insight into that prediction—that is, determine whether or not to place a trade and what kind of trade to perform.

Predicting Forex rates is no trivial task. It is modeled as a chaotic and non-linear time series. Several computational intelligence techniques such as artificial neural networks, support vector machines, and the ANFIS have been used to predict market trends.

Meanwhile, Genetic Algorithms are also a popular class of algorithms found in the literature used as part of an algorithmic trading strategy. These GA's find their use in determining proper trade timing and decision-making.

The next logical step would be to combine an accurate trend prediction tool with a powerful decision-making module to create an automated algorithmic trading strategy. In addition to this, an interface must also be created for the user to hook up the strategy with his/her trading account and to monitor the transactions being performed.

Such a program will have real financial ramifications for the user. The goal, then, is to design a smart trading agent that is based on solid theory and would, under normal circumstances, almost guarantee a profit.

C. Objectives of the Study

The desired output software of this study is unique in that the program is autonomous. That is, it is expected to work with little to no user intervention. However, to initialize the system and manually, routinely regulate its transactions, the user would have access to a few features. An "alert-only" mode—wherein the system only alerts the user and does not place orders automatically—shall also be made available, should the user desire to perform trades himself. The functionalities of the smart agent are also listed.

1. Allows the user to
    (a) Input his/her online trading account information
    (b) Stop or start the agent from performing trades
    (c) View transaction logs
    (d) View open positions and current status
    (e) Manually override transactions
    (f) Perform backtesting and paper trading on the agent
2. The autonomous agent shall be able to
    (a) Connect to (a) broker/s using its/their HTTP Web API
    (b) Retrieve real-time prices and fluctuations of the currencies of interest
    (c) Predict future rates with reasonable correctness and determine the best decision to make at that time
    (d) Place a trade or open a position on the market through the broker API
    (e) Send the user an email alert when a profitable trading opportunity is detected and when a trade order is executed.

Additionally, this study will demonstrate the performance of the system if it were deployed live by backtesting on actual historical data and evaluating its prediction accuracy and financial performance.

D. Significance of the Project

As much as there are ripe opportunities to make profit in the Forex market, there is no one easy technique to capitalize on the market. Trading, whether using fundamental or technical analysis, can be daunting for a first-time trader. Moreover, there is always substantial risk to speculative investment, such as that in Forex.

This autonomous agent will allow a user to participate in algorithmic trading without generating his/her own strategy by simply relying on the prediction tools built into the agent. The "intelligence" built into the trading agent is also founded on sound theory and should provide reliable results under normal circumstances.

Moreover, an automated system, such as the one this study has produced, benefits both amateur and experienced traders alike by allowing them to perform trades hands-free, if they choose to do so. The agent is fully mechanized and does not require any user intervention.

E. Scope and Limitations

As pointed out by Gradojevic [12], technical trading, which this study relies on, ignores fundamental economic information such as the macroeconomic, political and social variables. The system is, therefore, limited in power to predict during periods of sharp change in these fundamental variables.

Yao et. al [1] also state that prediction with technical indicators work best in the short-term. This study is, therefore, focused on short-term prediction (30-minute timeframes).

This automated trader is primarily built to interface with the Web API of Oanda. While practically extensible to work with other brokers offering a similar HTTP API, this agent can currently only perform actual trades for clients trading with Oanda.

In this light, a simulated broker was also made as part of this project. The simulated broker implements an essential subset of the features of a true broker—it simulates placing, closing, and updating orders. It also provides rates sourced from an external provider to simulate the price provided by a true broker. This was created due to the limitations of finding other brokers which have a freely available

HTTP API like Oanda. In creating a simulated broker with a generalized Web API, we can demonstrate that the program is able to connect to any broker with a Web API simply by modifying the required wrapper functions.

Since an option is provided to change the instrument to trade even after setup, retraining the predictors will be needed. It is assumed and expected that the user will provide the appropriate data set to train the predictors to work for that instrument.

Finally, the agent does not take into account margin/balance requirements which can be broker and account-specific. The agent will perform trades as long as it detects opportunities to trade and the broker accepts the order placed.

F. Assumptions

1. Although it is clear the Forex rates behave chaotically, it is still assumed that the market is fairly behaved and is not moving erratically as a result of factors external to regular market volatility such as drastic economic changes, market-making events, and *force majeure* which the agent does not take into account.

2. The user will be able to source historical data for himself from a data provider of choice, under assumption that such sources are freely available online or from their brokers.

3. Actual trade execution is done within the broker's servers/platforms, as is the standard execution practice.

4. While the agent may continue running any time, any day, it is programmed to restrict trades during off-market hours (Friday, 5 pm EST until Sunday 5 pm EST). The user is expected to be sufficiently informed that opening and closing trades at such times is not allowed by most brokers.

5. Finally, it is left to the user to decide and enable/disable hedging—the setting that allows or disallows Buy and Sell positions to exist simultaneously.

By default, Oanda accounts do not support hedging. This means that an order followed by an opposite order effectively closes out the former. For example, when there is still an open buy order, and an opportunity to sell was detected and acted upon, the subsequent sell order effectively closes out the buy order. This may or may not be beneficial in that the former order has still not reached the target profit or stop loss, while the latter order will not have any chance to reach its calculated target profit. Having no hedging support could be contrary to the intention of the agent, but it could also work for the user, if the prediction was anomalous. Again, control for hedging is left to the user – and it can only be controlled with the broker's settings and not from within the program.

II. Review of Related Literature

With ForEx being the largest financial market, there is extensive research interest in modeling and predicting Foreign Exchange Rates. [2]

Several computational intelligence approaches and tools to predict ForEx rates—such as Neural Networks and Neuro-Fuzzy Systems—are present in literature. The same is also true for the Stock Market, which is another popular and well-established financial market.

Neural Networks are recognized for their ability to approximate non-linear systems, an example of which is the time series of a currency pair exchange rate. [2] Wu and Lu [13] report several Neural Networks applied to forecasting of stocks, another financial instrument. Meanwhile, Khoonmirzaie et. al [14] utilized a three-layer perceptron neural network to predict the US Dollar-Franc currency pair, achieving a Mean Square Error (MSE) of 2.15×10^6.

Yu et. al [15] explore the question of whether or not Forex rates are predictable by surveying 45 articles proposing Forex prediction using ANN, concluding that ANN's can generally predict Forex with positive results.

Baasher and Fakhr [16] present machine learning techniques applied to creating a Forex rate uptrend/downtrend classifier. These are the the Radial Basis Function Network (RBFN), Multi-layer Perceptron (MLP), and Support Vector Machine (SVM). The SVM and MLP models each performed best two times out of their four experiments.

Neuro-Fuzzy Systems (NFS) or Fuzzy neural networks are hybrid systems combining artificial neural networks and fuzzy systems. Given numerical data, an FNN will generalize from the training data, tune parameters and create linguistic rules describing the problem space. An extensive discussion of Neural Fuzzy Systems is done by Siddique and Adeli in [17].

[1] proposed by Yao, Pasquier and Quek features the use of moving averages, a type of technical indicator, and their novel Portfolio Trade Timing Optimization algorithm for optimizing the BUY and SELL schedule. The prediction tool

employed is the Fuzzy CMAC-Yager, a Fuzzy Neural Network.

Gharleghi [2], meanwhile, presented a *cointegration-based neuro-fuzzy system* for predicting exchange rates of currency pairs in the ASEAN region using macroeconomic variables. Their neural-fuzzy system consistently outperforms a plain vector error correction model in in-sample and out-of-sample forecasting Root Mean-Square Error (RMSE) and Mean Absolute Error (MAE).

Fuzzy inference systems (FIS) are built upon Fuzzy Set Theory and Fuzzy Logic. Fuzzy logic was first proposed by Zadeh [18] to allow for the modeling of uncertainty and imprecision in human reasoning by characterizing nonprobabilistic uncertainties through "fuzzy sets." Fuzzy logic allows computational systems to handle the ambiguity involved in real-world problems. [19]

The classic and most popular example of an NFS/FIS is the Adaptive-Network-Based Fuzzy Inference System or ANFIS [20] which has found wide applicability in areas such as modeling of non-linear systems, chaotic dynamics, and control, among others. Astelakis modelled the EUR/USD currency pair using ANFIS for daily exchange rate prediction, achieving 63% accuracy.

Aside from Forex, ANFIS is also used for stock market prediction as evidenced by Wei et. al [19] formulating a hybrid ANFIS based on n-period moving average to predict the Taiwan Stock Exchange Capitalization Weighted Stock Index (TAIEX). Tan[21] also harnessed ANFIS—in conjunction with Reinforcement Learning—for stock trading, resulting in profits that beat the market by 50 percentage points.

While ANFIS can predict future rates, another method is needed to determine trade timing and decision-making. In an article on Algorithmic Trading, Nuti lists quadratic programming, particle swarm optimization, and genetic algorithm as algorithms "to identify profit-making opportunities."[22] Genetic algorithms (GA) are a class of search methods used in optimization inspired by the biological process of natural selection and genetics. Trade signal generation and trade execution can be seen as an optimization problem with multiobjective constraints[22], hence the aptness of GA in forex trading.

Dempster[23] empirically determined that Genetic Programming outperforms Markov Chain Linear Programming Approximation, Simple Heuristics, and Reinforcement Learning for intraday Forex trading.

Hirabayashi's study[24] focused on improving and finding best trade timing for short-interval trading using Genetic Algortihm. In both leveraged and non-leveraged experiments, their GA-based proposed strategy outperformed the neural-based strategy. Meanwhile, Papadamou's GATradeTool[25] optimizes trading rule parameter sets using GA. The tool outperforms commonly used, non-adaptive, software tools with regards to return stability and time savings.

Kuo et. al [26] proposed a stock decision support system composed of a Genetic Algorithm based Fuzzy Neural Network (GFNN) which evaluates the qualitative indicators integrated with an ANN which processes the technical (numerical) indexes.

The specific GA this study shall utilize is the Non-Dominated Sorting Genetic Algorithm II (NSGA-II)[27]—an elitist, multi-objective evolutionary algorithm proposed by Kalyanmoy Deb as an improvement over the original NSGA. Zitzler et. al present a comparison of MOEAs in [28]. In ranking the MOEAs based on distance to the Pareto-optimal front, NSGA came in second only to the Strength-Pareto Evolutionary Algorithm.

For further reading, Rifki and Ono [29] discuss and explore 14 GA approaches to portfolio optimization, detailing their usage contexts and experimental results.

In the literature, ANFIS and Genetic Algorithms are found to be used in tandem for financial trading agents. Such is the case in [30] where technical indicators are used as inputs for the ANFIS to predict the TAIEX, and parameters are further optimized using genetic algorithm. The ANFIS-GA model was shown to be superior to three previous agents in terms of RMSE.

The ANFIS-GA synergy doesn't exclusively find applicability in financial uses, but also in engineering. In a paper by Ghose et. al [31], a GA-optimized Non-

Linear Multiple Regression (NLMR) and ANFIS are used to predict runoff resulting from the precipitation on river catchments.

Also found in literature are smart agents or autonomous agents which are programs that can decide for and perform trades on behalf of the user algorithmically. Barbosa [11] presented an autonomous Forex trading agent which traded the USD/JPY pair in a 6-hour timeframe. Their agent features three modules named *Intuition*, *A Posteriori Knowledge*, and *A Priori Knowledge*. The agent resulted turned out with 54.41% accuracy and a profit of 60% over 14 months.

Another smart agent was proposed by Alrefaie et. al [6] in 2013, another ANFIS-GA synergism. It utilizes the ANFIS as a prediction tool, in combination with the Non-Dominated Sorting Genetic Algorithm II to determine trade timing and decisions. Alrefaie's agent achieved an accuracy of 82.12% vs. Barbosa's 54.41%, and a profit of approximately 120% over one year of trading.

The discipline and study of algorithmic strategies in trading securities can be put under the umbrella of *algorithmic trading*. The practice of algorithmic or computational trading is so widespread that in 2004, 50.6% of the trading done on the New York Stock Exchange Big Board is done by programs [8]. Bell [10] points out that algorithmic trading has caused increased market volatility and that algorithmic trading is now a "competitive necessity" for traders.

In summary, various computational intelligence approaches to predicting financial data, such as foreign exchange rates, are found in literature. ANFIS has been a popular, reliable choice for predicting financial time-series data like Forex, while genetic algorithm and genetic programming are useful in identifying proper trade timing and decision-making. Together, GA and ANFIS create robust systems which can produce trading strategies that lead to profit.

III. Theoretical Framework

A. Foreign Exchange Market

The foreign exchange market is a worldwide, decentralized, over-the-counter financial market wherein counterparites facilitate the trading of currencies [9]. It is composed of several electronic communication networks (ECN's) that mediate between banks, institutions, and speculators. The market is decentralized as transactions do not go through common exchanges such as stock markets. The market is open without interruption from Sunday 5 pm until Friday 5 pm, New York Time, at which point brokers generally halt trading and prices remain stationary.

Currencies are quoted against another currency, showing the value of the currency against another. As such, currency pairs are quoted like so:

$$\text{USD/PHP} = 44.8000$$

This simply means that 1 US Dollar trades for 44.8000 Philippine Pesos. The currency to the left of the slash is called the *base currency*, while the one to the right is the *quote currency*. Forex rate quotes often use the US Dollar as the base currency. However, for pairs involving the British Pound, New Zealand Dollar, and Australian Dollar, the said currencies are used as the base currency.

Forex quotes may express either the *bid price* or the *ask price*. The bid price refers to the price at which the market will buy the security. Intuitively, the ask price is the rate at which the market will sell the base currency. Consider the following quote:

$$\text{USD/PHP} = 44.8000/500 \text{ Bid} = 44.8000 \text{ Ask} = 44.8500$$

If we are entering a *short position* on (that is, selling) the US Dollar, we will receive 44.80 pesos for every dollar. Conversely, if we are going into a *long position* on (that is, buying), a dollar can be purchased at 44.85 pesos.

Another concept relevant to trading is the "pip." A *pip* is a standardized unit and is the smallest amount by which a currency quote can change, which is usually

$0.0001 for U.S.-dollar related currency pairs, which is more commonly referred to as 1/100th of 1%, or one basis point. The *spread* is the difference between the bid and ask price of a currency pair, usually expressed in pips. Returning to the previous example, the spread of the USD/PHP pair would be 500 pips.

A forex transaction, called an *order*, is initiated or opened by placing a move to buy or sell the security at the current market price called the *execution price.* In the opportune time, the trader or agent will decide to close the trade which triggers the security to be either bought or sold back to the market. The intuition behind making profit from price movements is to "Buy Low, Sell High." Hence, when we anticipate the price to rise, a Buy order is opened, and is closed once the price has risen to a desired level. Conversely, when the price is forecast to go low, an order to Sell can be made at the current price. This virtually means that the trader "owes" the broker a certain number of units of the security. Once the price dips to a desired level, the trader closes the position by buying the security at the low price, thus covering the "borrowed" units from the broker. Profit is made this way because the trader sold at a high price, and bought back at a low price. This practice is called *short selling.*

The target profit at which the broker automatically closes the order is called the *take profit.* The intent of the take profit is to specify the point at which the price is satisfactory enough to cash in. Meanwhile, an order parameter called the *stop loss* is the price at which the order is automatically closed when the price moves contrary to the prediction. The stop loss limits the amount of loss the account can take before bailing out. Both the take profit and stop loss can be specified upon opening the trade, or modified while the order is still open.

B. Algorithmic Trading

Algorithmic trading pertains to the automated trading of financial securities, such as currencies, using computer programs designed to respond to real-time market data with the goal of executing trades to maximize profits [10]. These programs are called *trading robots*, *automatic traders*, *autonomous agents*, or *smart agents* in literature. In essence, these agents take the place of a human trader/speculator by processing data such as prevailing market rates and news [7], making a buy or sell decision, and placing a trade on behalf of the portfolio owner. By solely relying on data, the process eliminates emotional bias that humans may have. [8]

Profit in algorithmic trading, as well as human speculative trading, is founded on *statistical arbitrage*. This refers to exploiting pricing inefficiencies in the market, capitalizing on the price differences of an asset over time. As a trading strategy, statistical arbitrage utilizes data mining, artificial intelligence and statistical methods. [32]

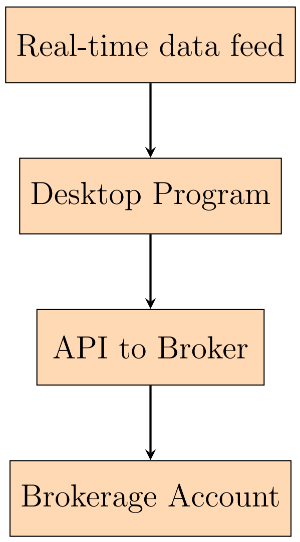


Figure 1: Fully-automated trading agent

Figure 1 illustrates the generalized flow of information of a fully-automated trading agent and the components thereof.

The development of one's own algorithmic trading agent would involve the following process: First, the trader must identify a *trading strategy* which he shall implement in the agent. This strategy would be responsible for identifying trends in the data and profit opportunities to exploit. Once it has been implemented, the trader would then *backtest* the strategy against historical prices and/or perform *paper trading* in order to determine the efficacy of the strategy and the correctness of the implementation. Once the trader has tweaked the agent and the strategy, and is satisfied with its performance, the agent will then be hooked up to the desired broker via its API and perform live trading.

C. Fuzzy Set Theory

Fuzzy Set Theory and Fuzzy Logic allow for the modeling of uncertainty and imprecision in human reasoning by characterizing nonprobabilistic uncertainties through "fuzzy sets." This has found applicability in automatic control, customer electronics, signal processing, time-series prediction, information retrieval, computer vision, data classification, and decision-making, among others.

C..1 Fuzzy Sets

A *fuzzy set* or fuzzy class is defined as a set with unsharp boundaries in which the transition from "belonging to the set" to "not belonging to the set" is gradual rather than abrupt. [33] This is in contrast to traditional *crisp* sets wherein there is an absolute boundary for set membership. Let A be a fuzzy set defined by

$$A = \{(x, \mu_A(x)) | x \in X\} \tag{1}$$

where X is a collection of objects called the *universe of discourse*, or simply universe, and x is a generic element in X. $\mu_A(x)$ is called the *membership function* (MF) or characteristic function. The MF specifies the degree of "belongingness" of an element x by assigning a value from 0 to 1, with 0 indicating non-membership and 1 absolute membership.

It can be said that a fuzzy set is an extension of the classic set. When $\mu_A(x)$ has a range $[0, 1]$, the set is fuzzy. If the range is restricted to $\{0, 1\}$, the set is crisp.

A fuzzy set may also be defined in the following manner:

$$A = \begin{cases} \sum_{x_i \in X} \mu_A(x)/x_i & \text{if X is discrete} \\ \int_X \mu_A(x)/x_i & \text{if } X \text{ is continuous} \end{cases} \tag{2}$$

It should be noted that the summation and integral operations do not mean summing, but rather union of the $(\mu_A(x), x_i)$ pairs and that the / does not imply division, but is simply a marker.

An example of a continuous membership function is the Generalized Bell MF, which is the preferred MF for use in the Adaptive Neuro-Fuzzy Inference System to be discussed later. It has four parameters $\{a, b, c, d\}$. The function is defined as:

$$bell(x; a, b, c) = \frac{1}{1 + \left| \dfrac{x-c}{a} \right|^{2b}} \tag{3}$$

and illustrated in Figure 2. The parameters c and a can be adjusted to get the desired center and width, respectively. Meanwhile, b controls the slope at the crossover points.

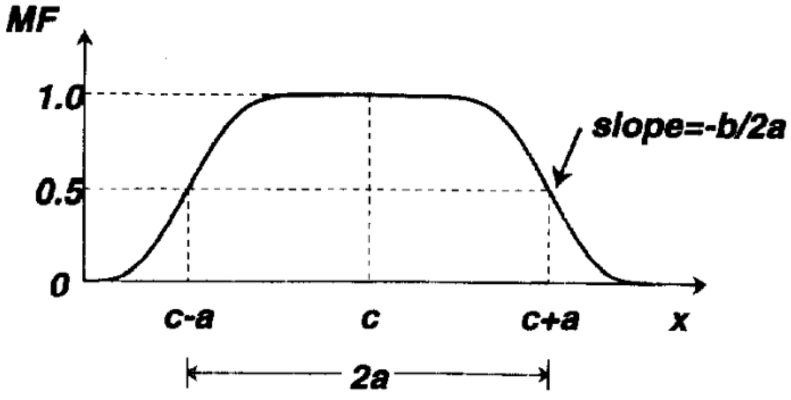


Figure 2: Physical meaning of parameters in a generalized bell function

As with human classification, determining the degree of an object belonging to a certain class or character is subjective. Thus, the choice of membership function is subjective as well. Its subjective and non-random nature is what distinguishes fuzzy set theory from probability theory which is an objective study of randomness.

A *fuzzy singleton* A is a fuzzy set containing only a single value from the universe of discourse whose membership value is one. Formally,

$$A = \{(x_0, \mu_A(x_0)) | \mu_A(x_0) = 1\}. \tag{4}$$

A fuzzy set is said to be *empty* if and only if the value of the membership function is zero across all elements of X.

Two fuzzy sets, A and B, are *equal* if and only if $\mu_A(x) = \mu_B(x)$ for all x in X. This is denoted by $A = B$.

C..2 Fuzzy Set Operations

The following are the operations defined over fuzzy sets in the seminal paper on fuzzy sets by Zadeh [18].

Union. The *union* (disjunction) of two fuzzy sets A and B is a fuzzy set C. This is denoted by $C = A \cup B$. The membership function of C is defined by:

$$\mu_C(x) = \max[\mu_A(x), \mu_B(x)] = \mu_A(x) \vee \mu_B(x), x \in X \tag{5}$$

Also pointed out by Zadeh [18] is a more intuitive definition of union—the smallest fuzzy set containing both A and B.

Intersection. The *intersection* (conjunction) of two fuzzy sets A and B is a fuzzy set C. This is denoted by $C = A \cap B$. The membership function of C is defined by:

$$\mu_C(x) = \min[\mu_A(x), \mu_B(x)] = \mu_A(x) \wedge \mu_B(x), x \in X \tag{6}$$

On the same token as union, the intersection is the largest fuzzy set which is contained in both A and B. The intersection corresponds to the Fuzzy AND operation (also called the T-norm) which can be defined in other ways besides the min, such as the product T-norm ($T(a, b) = \mu_A \times \mu_B$).

Complement. The *complement* (negation) of a fuzzy set A, denoted by $\bar{A}$, $-A$ or $NOT A$ is defined as

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{7}$$

D. Fuzzy Inference Systems

Fuzzy inference systems (FIS) allow for the mapping of input variables to output variables with the use of fuzzy logic. In contrast to Boolean/classical inference, FIS utilizes fuzzy membership functions and fuzzy relations to evaluate a set of input. A fuzzy inference system is built upon a rule base of fuzzy "IF-THEN" rules. These rules map to a fuzzy relation against which input are evaluated. Here is an example of a set of fuzzy rules:

Rule 1: IF *service* is *excellent* AND *food* is *good* THEN *tip* is *generous*

Rule 2: IF *service* is *bad* AND *food* is *bad* THEN *tip* is *cheap*

The general inference process of an FIS, illustrated in Figure 3, can be condensed into the following steps:

Fuzzification Crisp inputs for each variable are evaluated for their fuzzy memberships.

Aggregation of Antecedents The membership values of the antecedents of each rule is aggregated using T-norm (min or product) for conjunction, and max (or equivalent) for disjunction. This determines the degree of fulfillment of the rule.

Implication In this stage, the consequent or output of each rule is evaluated given the aggregation of its antecedents from the former step.

Aggregation of Consequents The consequent for each rule is then aggregated to obtain the total resultant fuzzy output. This is usually done using the max operation.

Defuzzification To obtain a crisp output from the resulting fuzzy set, a defuzzification technique is applied. The most common method is the Center of Gravity (COG) or Centroid method.

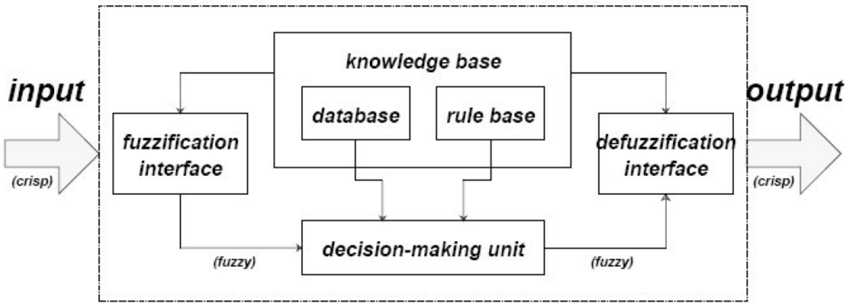


Figure 3: Fuzzy inference system

E. Sugeno Fuzzy Inference

The Sugeno Fuzzy Inference Model was proposed by Takagi & Sugeno in 1985, and Sugeno & Kang in 1988, thus it is also called *TSK Inference*. The goal in mind was to develop a fuzzy modeling approach for input-output data sets. [17] Sugeno Inference is analogous to the earlier described FIS (of type Mamdani), with the main difference being that in TSK the consequent is a crisp function; hence, there is no output membership function and defuzzification need not be performed.

A Sugeno fuzzy model can be generally described as a collection of rules of the form

$$k : \text{If } x \text{ is } A_i^k \text{ and } y \text{ is } B_j^k \text{ then } z^k = f(x,y) \tag{8}$$

where x and y are the crisp inputs, z is the output, A_i and B_j are the fuzzy membership functions in the antecedent, $z = f(x,y)$ is a crisp output in the consequent, $k = 1,2,...,R$, $i = 1,2,...,N$, $j = 1,2,...,M$. N and M are the number of membership functions for x and y, respectively, and R is the number of rules.

Without loss of generality, let us consider a two-input, one-output Sugeno fuzzy model as illustrated in Figure 4. Here, each input x and y has two MFs $\{A_1, A_2\}$ and $\{B_1, B_2\}$, respectively, while z_1 and z_2 are the consequent functions. It's two rules are:

r1: IF x is A_1 AND y is B_1 THEN $z_1 = a_1x + b_1y + c_1$

r2: IF x is A_2 AND y is B_2 THEN $z_2 = a_2x + b_2y + c_2$

Generally, $z = f(x, y)$ can be any polynomial in x and y which sufficiently describes the system to be modelled. However, in application with the ANFIS this is usually a linear combination of the inputs plus a constant term, i.e. a first-order polynomial. Thus, a TSK system with such output functions is called a first-order Sugeno fuzzy model.

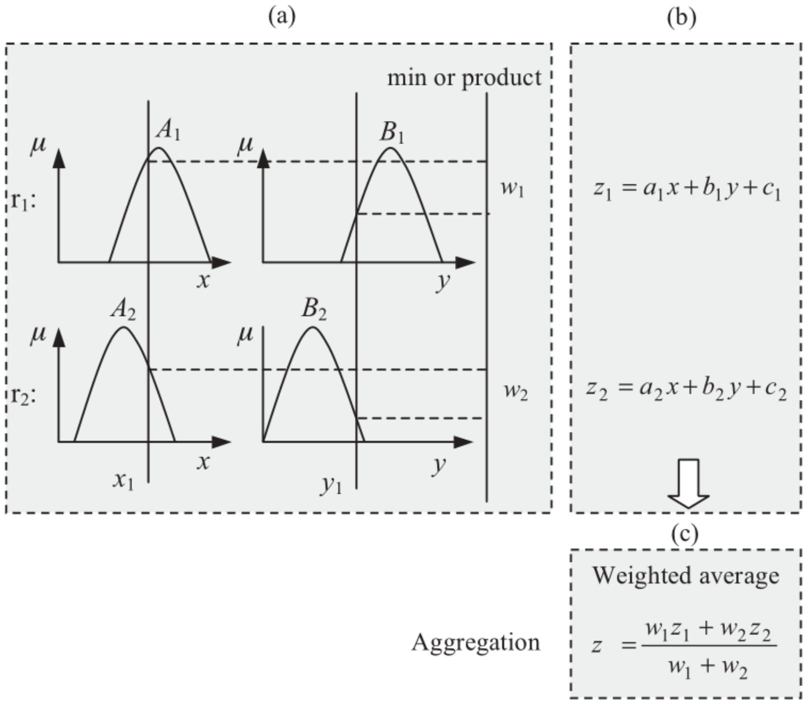


Figure 4: Two-input, single-output, first-order TSK fuzzy inference model

Figure 4(a) illustrates the fuzzification of the crisp inputs x_1 and y_1. Each output z_k of each rule is weighted by the *firing strength* w_k. The firing strength—for example, with ANDed rules—is calculated using a T-norm operator, either the minimum or product rule as

$$w_k = \min(\mu_{A_i}, \mu_{B_j}) \text{ or } w_r = \mu_{A_i} \times \mu_{B_j}. \tag{9}$$

Given that the parameters $\{a_k, b_k, c_k\}$ are known, the consequent z_k for each rule can be computed, as shown in Figure 4(b). The final output is then computed using a weighted average of the crisp consequent outputs z_k. This substitutes for the costly center of gravity defuzzification of the Mamdani FIS. With weights computed as per Eq. 9, the final output is obtained as

$$z = \frac{\sum\limits_{i=1}^{R} w_i z_i}{\sum\limits_{i=1}^{R} w_i} \tag{10}$$

.

As for our two-input example case, it is as mirrored in Figure 4(c)

$$z = \frac{w_1 z_1 + w_2 z_2}{w_1 + w_2}$$

F. Adaptive Neuro-Fuzzy Inference System

The Adaptive Neuro-Fuzzy Inference System or *ANFIS* was first proposed by Jang [20] in 1993 and was originally called the Adaptive-Network-based Fuzzy Inference System. This is accounted for by the fact that ANFIS has the architecture of an adaptive network and is functionally equivalent to a fuzzy inference system. Often, an ANFIS corrsponds to a Sugeno FIS. This adaptive capability makes ANFIS useful in non-linear system modeling, learning, and fuzzy control problems. [17] In his seminal work, Jang used the Stone-Weierstrass Theorem to prove that the ANFIS is fundamentally equal to a fuzzy inference system which has unlimited power to approximate non-linear systems.

F..1 Architecture

The architecture of a two-input (x_1 and x_2), single-output ANFIS with four rules which corresponds to a first-order Sugeno system is illustrated in Figure 5. The layers of this adaptive network shall be discussed in this section.

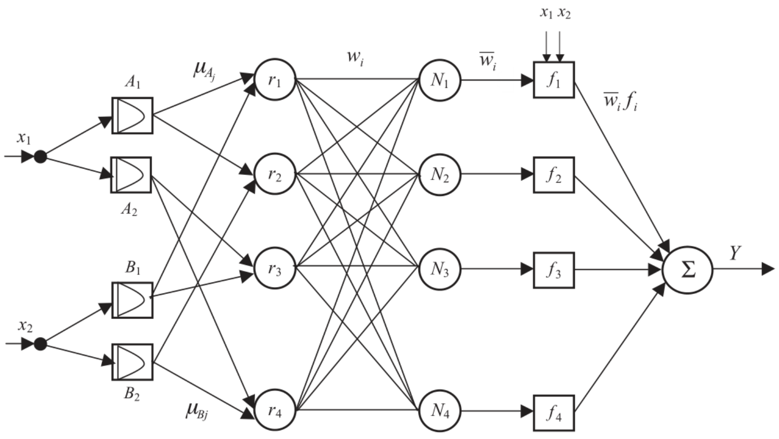


Figure 5: ANFIS Architecture

Layer 1: Every node in this layer is an adaptive node which computes for the membership value of the input against its corresponding MF. Usually, these MFs are chosen to bell-shaped. For example, with our two-input example which has two rules such that

$$\mu_{A_j}(x_1) = \frac{1}{1 + \left| \dfrac{x_1 - m_{A_j}}{\sigma_{A_j}} \right|^{2b_{A_j}}}, \ \mu_{B_j}(x_2) = \frac{1}{1 + \left| \dfrac{x_2 - m_{B_j}}{\sigma_{B_j}} \right|^{2b_{B_j}}} \tag{11}$$

with $\{m_{A_j}, \sigma_{A_j}, b_{A_j}\}$ and $\{m_{B_j}, \sigma_{B_j}, b_{B_j}\}$, $j = 1, 2$, as the parameter sets. These are, then, called the *premise parameters*.

Layer 2: Every node in this layer is a fixed node corresponding to the rules r_i, $i = 1, ..., 4$. Each node computes for the firing strength of the rule

$$w_i = \mu_{A_j}(x_1) \cdot \mu_{B_j}(x_2), \ j = 1, 2. \tag{12}$$

Layer 3: Every node in this layer is a fixed node labeled N_i, $i = 1, ..., 4$. Each node computes for the normalized firing strength of the corresponding rule r_i as

$$\bar{w_i} = \frac{w_i}{\sum\limits_{i=1}^{4} w_i},\ i = 1, ..., 4. \tag{13}$$

Layer 4: Every node in this layer is an adaptive node which computes for the weighted value of the consequent part of each rule as

$$\bar{w_i} \cdot f_i = \bar{w_i}(a_i x_1 + b_i x_2 + c_i),\ i = 1, ..., 4. \tag{14}$$

The parameter set in this layer is $\{a_i, b_i, c_i\}$, $i = 1, ..., 4$ and are collectively called the *consequent parameters.*

Layer 5: The single node in this layer is a fixed node which sums all the fired rule values to compute the overall output.

$$Y = \sum_i \bar{w_i} \cdot f_i = \frac{\sum_i w_i f_i}{\sum_i w_i},\ i = 1, 2, ..., 4 \tag{15}$$

F..2 Initial Fuzzy Model

To be able to model a system using ANFIS, an initial fuzzy model must first be derived. [34] This will determine the number of rules for each input and, thus, also the structure. The two popular data clustering techniques to obtain an initial model are:

1. **Subtractive clustering**: a fast, one-pass algorithm for estimating the number of clusters and cluster centers in a dataset. It works by assuming that each data point is a potential cluster center and calculates the probability that it is so.

2. **Fuzzy C-Means**: a data clustering technique wherein each data point belongs to a cluster or partition to some degree. It is closely related to the k-means algorithm.

F..3 Hybrid Learning Rule

ANFIS features a hybrid learning rule which can be decomposed into two learning schemes: learning the antecedent MFs and parameters, and learning the consequent parameters. With the antecedent parameters fixed, the output can be expressed as a linear combination of the consequent parameters

$$Y = \bar{w}(ax_1 + bx_2 + c) = (\bar{w}x_1)a + (\bar{w}x_2)b + \bar{w}c \tag{16}$$

Consider a training data set N. Equation 16 can be expressed in vector-matrix form as

$$AP = Y \tag{17}$$

where P is the unknown parameter vector and A is the coefficient matrix. As it is usually the case that the number of training pairs in N is greater than $|p|$, no unique solution exists for Eq. 17. And so, a more sensible approach would be to perform a *least-squares estimation* (LSE) of the parameter vector p and minimize the squared error $\|AP - Y\|^2$. The most popular formula for this estimate is the pseudo-inverse of A:

$$\hat{P} = (A^T A)^{-1} A^T Y \tag{18}$$

where $(A^T A)^{-1} A^T$ is the pseudo-inverse of A if $A^T A$ is non-singular.

Generally, the structure of the network is assumed fixed and parameters are tuned using a hybrid learning rule. In the forward pass, node outputs are computed and propagated to layer four and the consequent parameters are determined using the least-squares estimate. In the backward pass, the error rates (the derivative of the error with respect to each node output) are backpropagated towards the input end where the antecedent parameters are updated by a gradient descent method while the consequents remain fixed.

G. Genetic Algorithm

Genetic algorithms (GA) are a class of search methods used in optimization inspired by the biological process of natural selection and genetics. The basic idea is to evolve an initial population of solutions towards a desired optimization goal. This would entail evaluating members of the population against a certain *objective function* which we want to maximize or minimize. This function determines the "*fitness*"/goodness/optimality of the solution at every iteration. Intuitively, we would want to keep the most fit individuals and propagate their features to the offspring generation while discarding the least desirable ones. Over time, the population should ideally converge to the desired solution.

Each solution in the population is called a *chromosome*. Classically, the solutions are encoded as strings, most often binary strings, though other representations (real numbers, lists, data structures) also exist. For example, each bit in a binary string may encode a characteristic or feature of the candidate solution. Each position or set of positions in the chromosome that encodes a feature is called a *locus* and the possible values at each locus is called an *allele*.

A genetic algorithm begins with an initial population, often randomly generated. This set of solutions shall then pass through the various GA Operators to produce the succeeding generation. The goal of *selection* operators is to choose the most desirable solutions to participate in reproduction. The *crossover* and *mutation* are the modification operators, they mix and change the features of the selected individuals to be carried over to the offspring. Figure 6 illustrates the cycle of reproduction in a genetic algorithm with the use of GA operators. [17]

G..1 Genetic Algorithm Operators

Selection This is founded on the principle of natural selection that the fittest individuals are favored in reproduction. Using a selection operator, parents are chosen to participate in the reproduction of the offspring generation. This can be done randomly (*random selection*) or in proportion to their

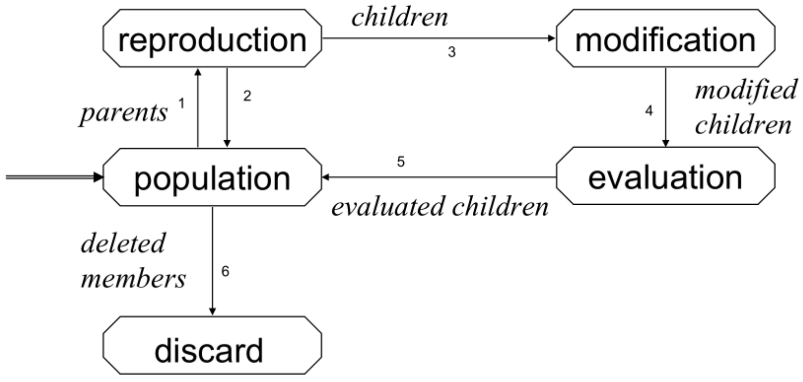


Figure 6: Cycle of reproduction in a GA

fitness (*proportional selection*). *Tournament selection* or *rank-based selection* may also be used. Another selection technique is *elitism* wherein a certain set of individuals from the current population will survive without mutation to the next. This is often used to fully preserve the fittest individual and its features.

Crossover Also called recombination, crossover takes two parents to produce one or two offspring. In a biological sense, crossover refers to the blending of genetic information from the parents. For strings, this can be achieved by splicing the string in one or two *crossover points* and transplanting the segments between the offspring(s). Figures 7 and 8 illustrate this. Other crossover operators such as arithmetic crossover (performing XOR or average) over the parents also exist. The GA may also employ a crossover probability c_p which would determine whether or not recombination will be performed at that instance.

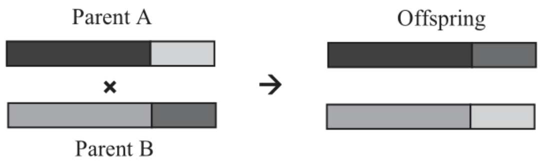


Figure 7: One-point crossover

Mutation This simply refers to changing a feature in the selected parent. For

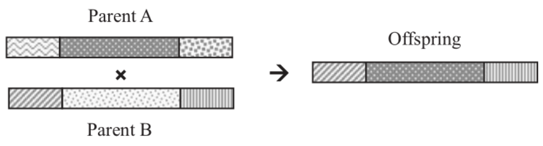


Figure 8: Two-point crossover

a binary string, this is easily achieved by flipping a certain bit to change the feature it encodes. For integer-coded strings, a random changing of the integer may be performed. Scramble mutation refers to switching the places of the values across the chromosome's loci. As with crossover, a mutation probability parameter m_p may be elected.

H. Genetic Programming

The objective of genetic programming (GP) is to use induction in order to devise a computer program that performs a desired function (e.g. approximate a function based on input-output pairs). This involves using evolutionary operators on candidate programs with a tree structure to improve the adaptive fit between the population of candidate programs and an objective function. Evaluation may involves execution of the program in order that its output may be evaluated.

The leaves of the tree, called *terminals*, represent the input variables or constants. These are passed along to the inner *nodes* which perform operations or functions on the inputs and passed further up the tree to undergo more node operations to produce the program's final result.

Appropriate evolution operators exist for the tree structures used in GP. Mutation may involve changing the terminals to other valid inputs and replacing the node operations with other operators. Meanwhile, crossover would involve switching subtrees at a certain branch, effectively exchanging the subprograms of the parents.

I. Multi-objective Optimization

Most real-world problems have several conflicting objectives which we aim to optimize. Thus, the need for multi-objective optimization (MOO) arises. A multi-objective optimization problem is concerned with finding a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. That is, we are searching for a solution with acceptable values across all objectives.

A multi-objective optimization problem can be mathematically stated as:

Minimize or maximize subject to

$$\begin{cases} f_m(x), & m = 1, 2, ..., M \\ g_j(x) \geq 0, & j = 1, 2, ..., J \\ h_k(x) = 0, & k = 1, 2, ..., K \\ x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, ..., N \end{cases} \tag{19}$$

where $f_m(x)$ are the objective functions, g_j are the inequality constraints, h_k are the equality constraints, and $x_i^{(L)}$ and $x_i^{(U)}$ are the variable bounds. The solution vector x is composed of n decision variables. That is $x = (x_1, x_2, ..., x_n)^T$. A *feasible solution* satisfies all $(J + K)$ plus the $2N$ variable bounds. These variable bounds restrict x_i from $x_i^{(L)}$ to $x_i^{(U)}$, thus defining the *decision space.* A solution x corresponds to a point in the decision space. Meanwhile, the set of all feasible solutions consist the *feasible region* or *search space.*

An important concept from MOO which is relevant to the Non-Dominated Sorting Genetic Algorithm II to be discussed later is *Pareto optimality.* This concept is built upon the idea of non-dominated solutions.

Consider two solutions A and B. If A is better than B in Objective 1, but B is better than A in Objective 2, we cannot say for sure which is "better" when both objectives are equally important. A and B are then called *non-dominated solutions.* When a solution, say C, clearly trumps another, say E, in all objectives, then we say that C *dominates* E.

Definition III..1. A solution $x^{(1)}$ is said to dominate $x^{(2)}$ if:

1. $x^{(1)}$ is no worse than $x^{(2)}$ in all objectives, i.e.

   $f_j(x^{(1)}) \not\triangleright f_j(x^{(2)})$ for all $j = 1, 2, ..., M$ AND

2. $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective, i.e.

   $f_j(x^{(1)}) \triangleleft f_j(x^{(2)})$ for at least one $j \in \{1, 2, ..., M\}$

When we draw a curve through all of the non-dominated solutions in the decision space, this is called a *Pareto-optimal front* (illustrated in Figure 9) and the solutions lying therein are called *Pareto-optimal solutions.*

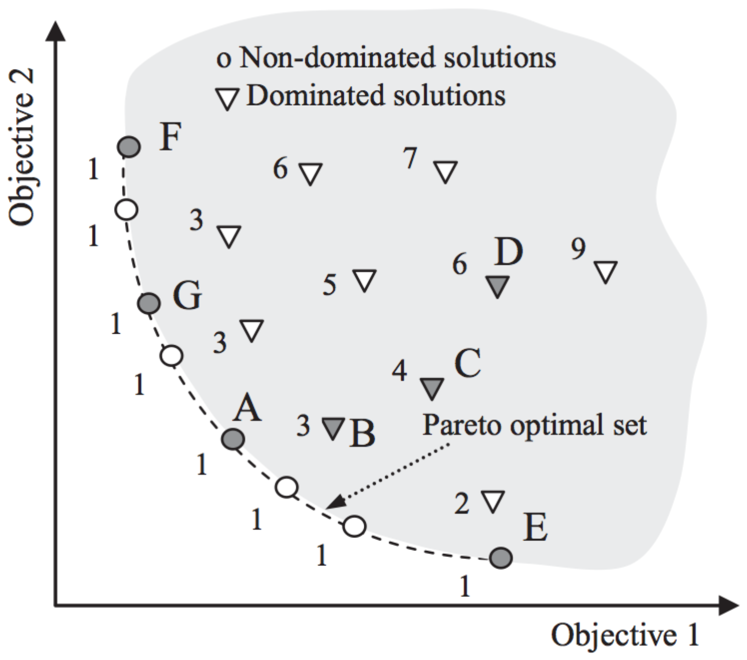


Figure 9: Illustration of the dominated, non-dominated, and the Pareto optimal solutions in the decision space

Clearly, the search space in the context of multiple objectives can be divided into two non-overlapping regions: the optimal and non-optimal sets. In the absence of higher-level information, all objectives are equally important. And so, the goals of multi-objective optimization are to find a set of solutions as close to the Pareto front as possible and to find such a set as diverse as possible. [35]

J. Non-Dominated Sorting Genetic Algorithm II

The Non-Dominated Sorting Genetic Algorithm II (NSGA-II) is a popular non-domination based genetic algorithm for multi-objective optimization. It is an improvement over the Non-Dominated Sorting Genetic Algorithm (NSGA) by introducing a more efficient sorting procedure, and elitism and eliminating the a priori need to set a share parameter. The algorithm features the `fast-non-dominated-sort` procedure with $O(MN^2)$ computational complexity where M is the number of objectives and N is the population size. Diversity of solutions is preserved through the crowded tournament selection. [27]

The steps involved in the NSGA-II algorithm are outlined as follows: [36]

1. Begin with a randomly generated parent population P_t and create an offspring generation Q_t using binary tournament selection, recombination, and mutation operators.
2. Combine Q_t and P_t to create a population of size $2N$ which shall be called R_t.
3. Perform the `fast-non-dominated-sort` procedure over R_t to order the individuals into non-dominating fronts.
4. Form the next parent generation P_{t+1} by performing crowded tournament selection on the population and taking the top N individuals.
5. Generate the next offspring generation Q_{t+1} from P_{t+1} using selection, crossover and mutation.
6. Repeat steps 2-5 until termination criteria are satisfied.

By combining the parent and offspring populations, we allow individuals from the previous generation to participate in the next iteration without mutation, thus introducing elitism.

The first procedure on which NSGA-II is mainly built is the `fast-non-dominated-sort`. Its main goal is to stratify the population into non-dominating fronts, where each front is dominated by the preceding fronts, as illustrated in Figure 11. Each indi-

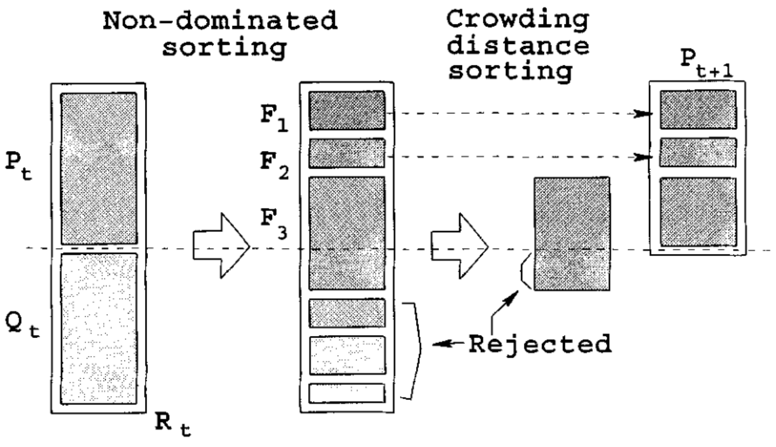


Figure 10: NSGA-II procedure

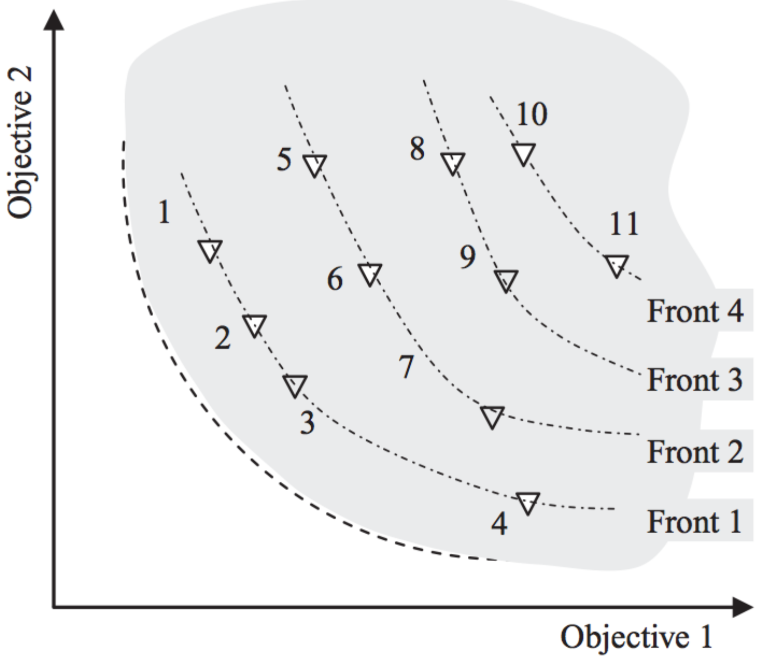


Figure 11: The population is sorted into non-dominating fronts

vidual then receives a rank based on which front it belongs to, with 1 being the completely non-dominating front.

In line with the goal of multi-objective optimization to maintain a good spread of solutions, NSGA-II makes use of a crowded tournament approach. This crowded tournament has two components. First is *density estimation.* Within each front, the *crowding distance*—an estimate of the perimeter of the cuboid formed by the nearest neighbors—of each individual is calculated. This is the average side

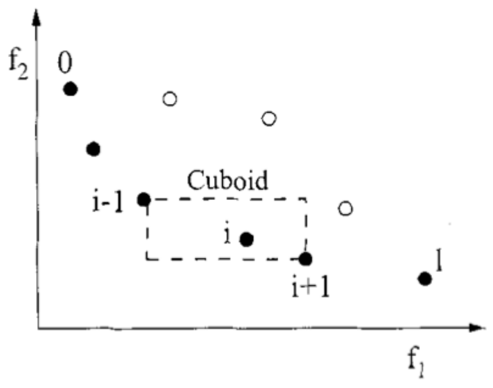


Figure 12: Illustration of the crowding distance

length of the cuboid, as shown in Figure 12. The procedure which performs density estimation, called `crowding-distance-assignment`, has $O(MN \log N)$ complexity.

The second component of the crowded tournament selection approach is the *crowded tournament operator* ($\prec_n$) which guides the algorithm into maintaining a uniformly-spread Pareto-optimal front. Assume that every individual i has the two following attributes defined: 1) nondomination rank (i_{rank}) and 2) crowding distance ($i_{distance}$).

Definition III..2. $\prec_n$ is a partial order defined as

$$i \prec_n j \text{ if } i_{rank} < j_{rank} \text{ or } ((i_{rank} = j_{rank}) \text{ and } (i_{distance} > j_{distance})).$$

Simply put, a solution with higher nondomination rank is preferred. And when two solutions have the same rank, the one with greater distance, that is the one in a less crowded region, is selected.

Deb [27] presents the worst-case complexities of the basic operations as follows:

1. nondominating sort is $O(M(2N)^2)$
2. crowding distance assignment is $O(M(2N)\log(2N))$
3. sorting on $\prec_n$ is $O(2N\log(2N))$.

The overall complexity, then, is $O(MN^2)$ which is subject to the nondominated sorting procedure of the algorithm.

IV. Design and Implementation

The agent communicates with the broker's server via its HTTP REST API. Through the API, the agent can retrieve real-time price fluctuations and place orders with the broker. These prices are then pre-processed into mathematical functions or technical indicators which the ANFIS and Decision Making Module may call for. After this, the pre-processed data are passed on to the ANFIS for prediction. Next, the Decision Making Module decides whether to Buy, Sell, or Hold using historical and real-time. Finally, if an order must be placed, the agent packages an appropriate request to the broker server, thus effectively placing the order. The take-profit is set as the highest point of the Predicted High data points for BUY orders, while the lowest point of the Predicted Low data points is taken for SELL orders. The converse reasoning is used for setting the stop-loss.

While this agent is currently tailored to view account status and perform actual trades over the Oanda API, the system also has concurrent functionality to place orders and retrieve previously sent orders to a simulated broker which offers a similar Web API, also made by the developer. This establishes the extensibility of the program to work with any HTTP API similar to that provided by Oanda and our simulated broker, with a minimally reasonable amount of effort.

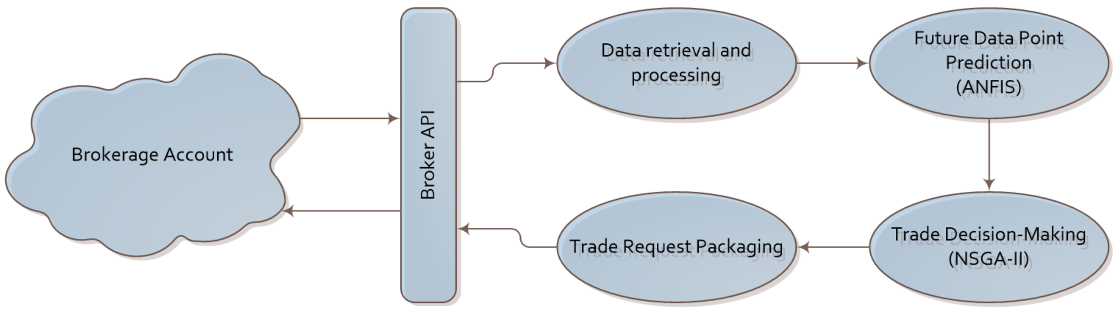

Brokerage Account
Broker API
Data retrieval and processing
Future Data Point Prediction (ANFIS)
Trade Decision-Making (NSGA-II)
Trade Request Packaging

Figure 13: Design of the agent

A. Use-Case Diagram

The AAGFA has only one user, the trader, which can perform the following functions: input his/her broker account details, start or stop the agent, view the agent's current status and previous logs, and manually override orders. When overriding, the user can modify the order placed by himself or by the agent itself.

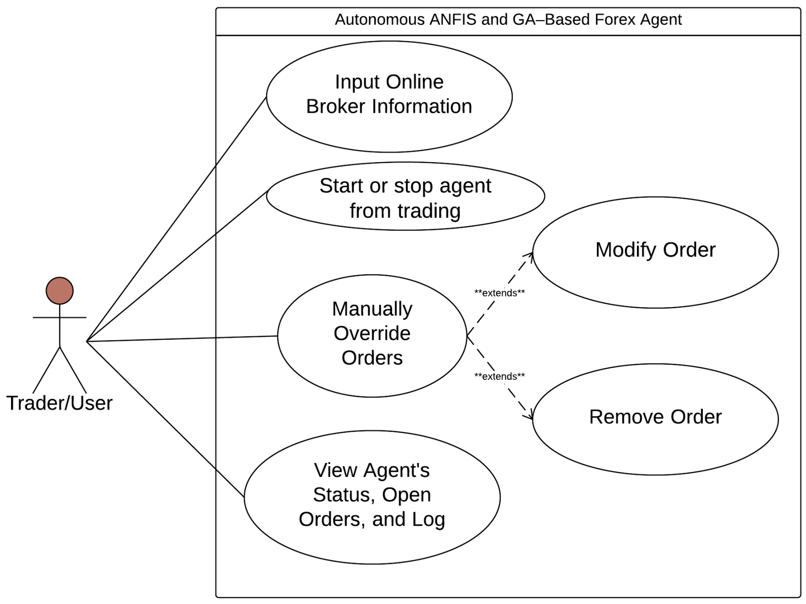

Autonomous ANFIS and GA–Based Forex Agent
Input Online Broker Information
Start or stop agent from trading
Modify Order
Manually Override Orders
extends
extends
Remove Order
Trader/User
View Agent's Status, Open Orders, and Log

Figure 14: Use-case Diagram

B. Context Diagram

The automomous agent has only one user which is the trader.

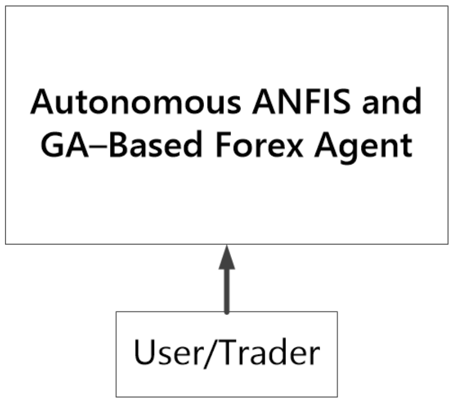


Figure 15: Context Diagram

C. Flowchart

When the user opens the AAGFA program, he/she may input new brokerage account details or replace the details if there is an existing configuration. If it is a new configurations, the user must provide the historical data for the training. Users of a currently trading agent may also re-train the ANFIS as desired. Once training is done, the agent runs by itself indefinitely and independently. The only actions the user may perform, aside from passively viewing the status, are 1) override order; and 2) stop the agent from running. The former does not disrupt the agent from continuously running, while the latter terminates the program.

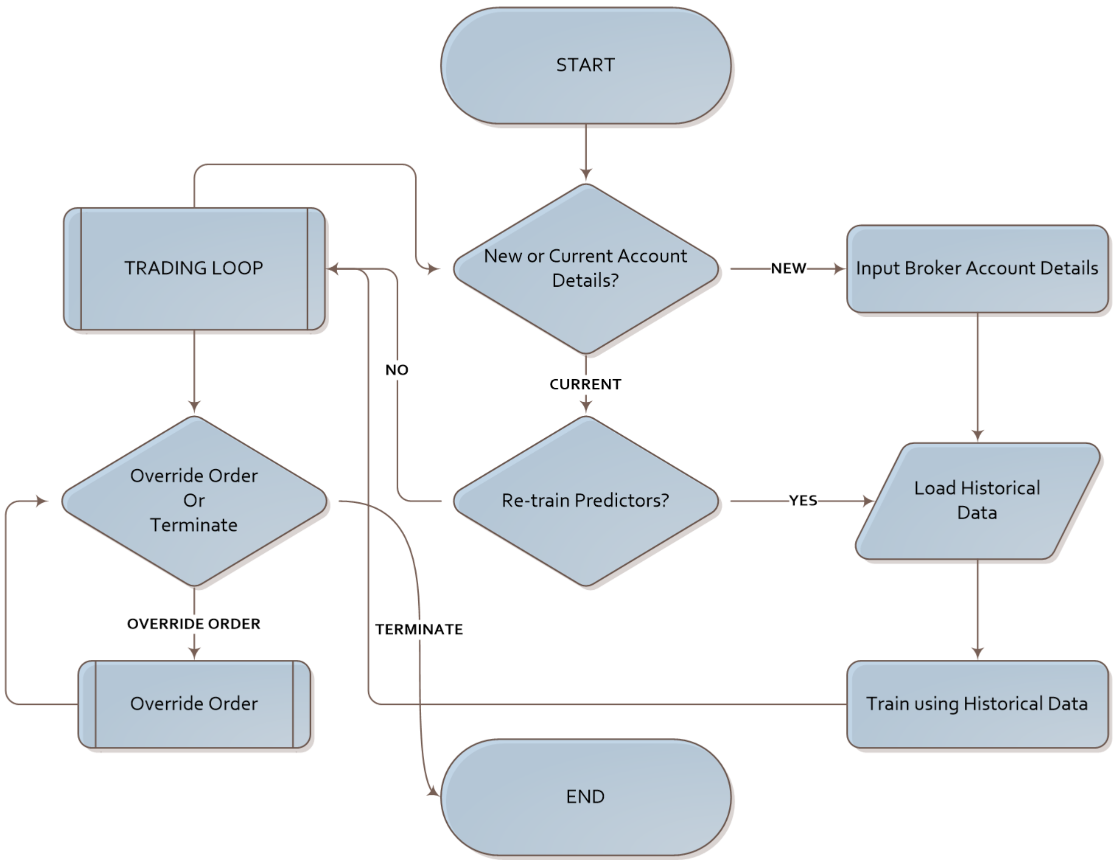

START
TRADING LOOP
New or Current Account Details?
NEW
Input Broker Account Details
NO
CURRENT
Override Order Or Terminate
Re-train Predictors?
YES
Load Historical Data
OVERRIDE ORDER
TERMINATE
Override Order
Train using Historical Data
END

Figure 16: High-Level Flowchart

D. Parameters and Input Pre-processing

Some parameters for training and optimizing the trader were adapted from Al-refaie et. al [6], such as the number of inputs and number of data points to predict for the ANFIS, number of generations for the GA, and the trading timeframe. However, some other parameters had to be empirically or arbitrarily selected for this program, as the information was not available in the basis paper.

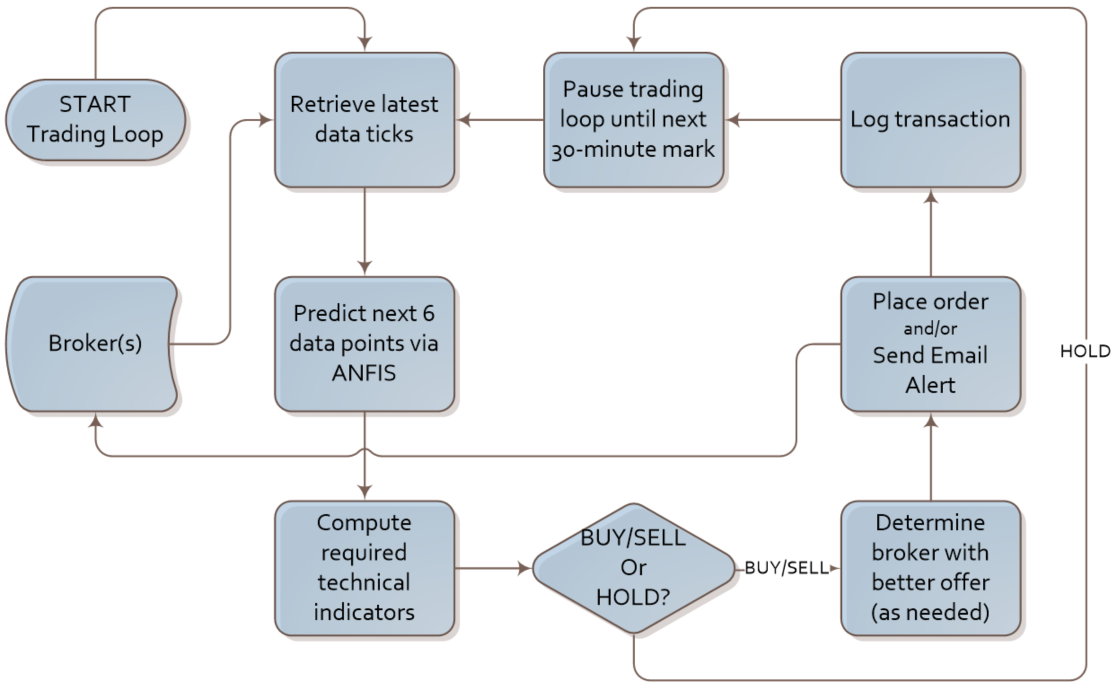

START Trading Loop
Retrieve latest data ticks
Pause trading loop until next 30-minute mark
Log transaction
Broker(s)
Predict next 6 data points via ANFIS
Place order and/or Send Email Alert
HOLD
Compute required technical indicators
BUY/SELL Or HOLD?
BUY/SELL
Determine broker with better offer (as needed)

Figure 17: Trading Loop Flowchart

As Al-refaie observed, Mean Squared Error for training was minimized when the input was passed through a smoothing filter, as this reduces noise in the input. They utilized an exponential moving average (EMA) to smooth the data. However, no specified EMA lookback period was specified, thus leading to the decision of using the default time period of the EMA function of TA-Lib which is 30.

Initial test runs of the ANFIS prediction sometimes produced anomalous results. The root of this was traced back to the EMA smoothing utilized. Initially, the output for the ANFIS was also taken from the smoothed series. The pre-processing function was modified to pair the smoothed inputs with the raw output. This caused the output predictions to behave more as expected.

V. Architecture

A. System Architecture

The agent is a desktop application built on the Qt Framework in Python 2.7.9 via PyQt4. The ANFIS training and prediction is delegated to MATLAB which is called inside the program using the official MATLAB Engine for Python. [37] The Genetic Programming framework is provided by the Distributed Evolutionary Algorithms in Python (DEAP) Library. [38]

Python was chosen for its clear, expressive syntax, powerful built-in data structures, support for both object-oriented and functional paradigms, and the wealth of libraries available. One such library on which the system heavily relies on is the Scipy package, which includes Numpy, Matplotlib, and Pandas. [39] Most of the heavy-lifting for data pre-processing and storage in-memory was provided by Numpy[40], with its highly flexible `ndarray` class. The central plot, as well as the price and equity curve seen produced by the backtesting module, is drawn using Matplotlib. [41] Meanwhile, the backend for the backtesting feature is mostly powered by the Python Data Analysis Library or `pandas`, for short. [42] Pandas provides a `DataFrame` class which supports vectorized operations. This is what allows the backtesting over several thousand data points to be completed in just seconds. Technical indicators were mostly easily computed through the use of the Python wrapper for TA-Lib (http://mrjbq7.github.io/ta-lib). TA-Lib is a comprehensive technical analysis library written in C and exposed to Python using the similarly named Python package.

All data requests to the Oanda Web API and the simulated broker API is packaged using the `requests` package readily found in the Python Package Index (PyPI, http://pypi.python.org/). Responses are decoded using Python's built-in `json` module. However, for Oanda requests, these functions are already wrapped using the `oandapy` package provided by Oanda itself.

The simulated broker is a PHP web service providing a REST-like API. It receives `POST` requests in a manner similar to Oanda's API and also sends back JSON responses. The simulated broker provides simulated prices obtained real-time from http://jsonrates.com/. It stores the orders it receives from the client into a single MySQL table.

B. Technical Architecture

The agent is a desktop application running on Python 2.7. The agent interfaces with the mathematical software MATLAB to utilize its Neuro-Fuzzy Design Toolkit for the ANFIS; version 2014b or up is required, as these are the versions containing the MATLAB Engine for Python. This system was developed and tested on Windows 8.1, but should also be functional on a Linux/Unix system provided that all other dependencies are met. Because of the persistent communication with the broker and data sources, a high-speed, low-latency, reliable internet connection is highly desirable.

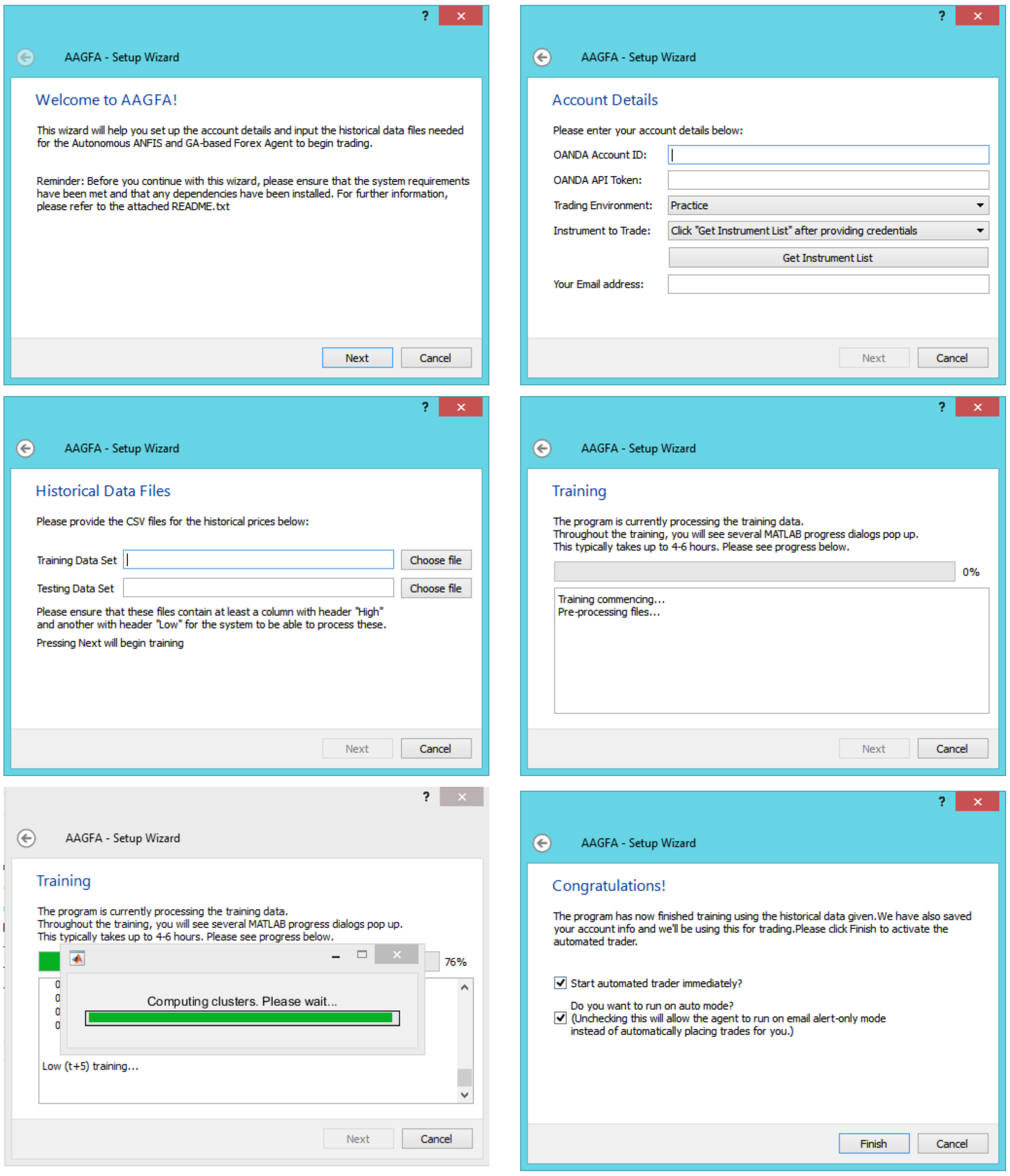
Minimum System Requirements:

1. A mid-range CPU released since 2010 (Intel Core2 Duo)
2. 2.00 GB of RAM
3. Windows 7
4. Python 2.7
5. MATLAB R2014b
6. Persistent, stable internet connection

VI. Results

In this section we run over the functionalities of the program and show that they fulfill the objectives.

First, shown in Figure 18 is the Setup Wizard. This allows the user to input the trading account information, instrument to trade, training and testng data sets, and the mode to run on once the program launches.



AAGFA - Setup Wizard

Welcome to AAGFA!

This wizard will help you set up the account details and input the historical data files needed for the Autonomous ANFIS and GA-based Forex Agent to begin trading.

Reminder: Before you continue with this wizard, please ensure that the system requirements have been met and that any dependencies have been installed. For further information, please refer to the attached README.txt

AAGFA - Setup Wizard

Account Details

Please enter your account details below:

OANDA Account ID:

OANDA API Token:

Trading Environment: Practice

Instrument to Trade: Click "Get Instrument List" after providing credentials

Your Email address:

AAGFA - Setup Wizard

Historical Data Files

Please provide the CSV files for the historical prices below:

Training Data Set

Testing Data Set

Please ensure that these files contain at least a column with header "High" and another with header "Low" for the system to be able to process these.

Pressing Next will begin training

AAGFA - Setup Wizard

Training

The program is currently processing the training data.
Throughout the training, you will see several MATLAB progress dialogs pop up.
This typically takes up to 4-6 hours. Please see progress below.

0%

Training commencing...
Pre-processing files...

AAGFA - Setup Wizard

Training

The program is currently processing the training data.
Throughout the training, you will see several MATLAB progress dialogs pop up.
This typically takes up to 4-6 hours. Please see progress below.

76%

Computing clusters. Please wait...

Low (t+5) training...

AAGFA - Setup Wizard

Congratulations!

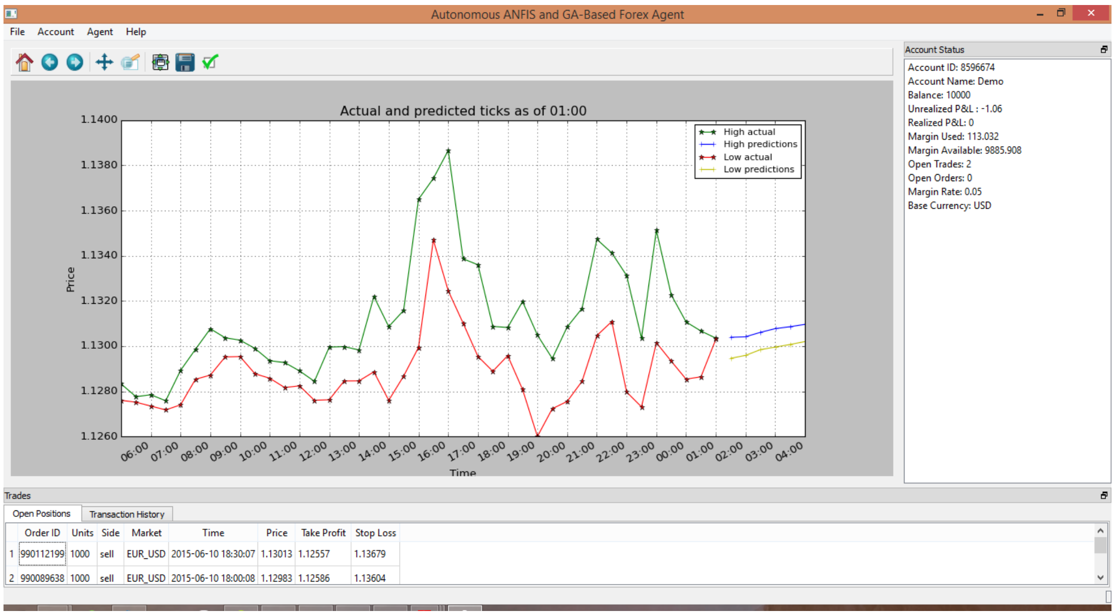
The program has now finished training using the historical data given.We have also saved your account info and we'll be using this for trading.Please click Finish to activate the trader.

[✓] Start automated trader immediately?

Do you want to run on auto mode?
[✓] (Unchecking this will allow the agent to run on email alert-only mode instead of automatically placing trades for you.)

Figure 18: Setup Wizard

Next, Figure 19 shows the program with two trades still open. The Open Positions tab show the open orders with their pertinent information. Double clicking on any open order brings up the Modify/Close dialog box as seen in Figure 20 where the user can either modify Take Profit and Stop Loss or immediately close the order.



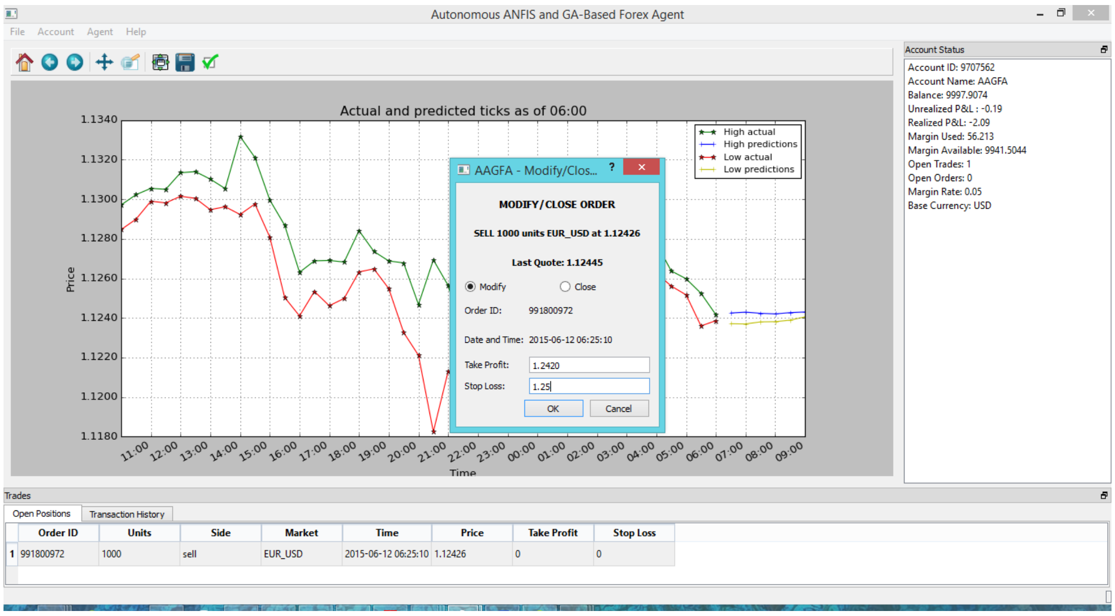
Account Status

Account ID: 8596674
Account Name: Demo
Balance: 10000
Unrealized P&L : -1.06
Realized P&L: 0
Margin Used: 113.032
Margin Available: 9885.908
Open Trades: 2
Open Orders: 0
Margin Rate: 0.05
Base Currency: USD

Open Positions | Transaction History

	Order ID	Units	Side	Market	Time	Price	Take Profit	Stop Loss
1	990112199	1000	sell	EUR_USD	2015-06-10 18:30:07	1.13013	1.12557	1.13679
2	990089638	1000	sell	EUR_USD	2015-06-10 18:00:08	1.12983	1.12586	1.13604

Figure 19: Typical view of the agent with trades open



MODIFY/CLOSE ORDER

SELL 1000 units EUR_USD at 1.12426

Last Quote: 1.12445

[✓ Modify] Close

Order ID: 991800972

Date and Time: 2015-06-12 06:25:10

Take Profit: 1.2420

Stop Loss: 1.25

Account Status

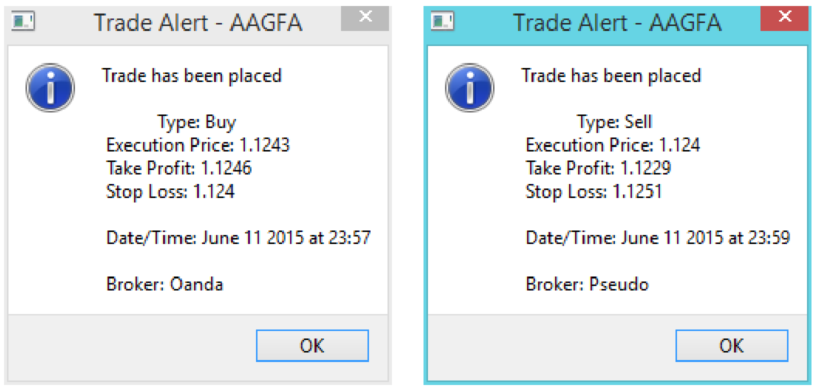
Account ID: 9707562
Account Name: AAGFA
Balance: 9997.9074
Unrealized P&L : -0.19
Realized P&L: -2.09
Margin Used: 56.213
Margin Available: 9941.5044
Open Trades: 1
Open Orders: 0
Margin Rate: 0.05
Base Currency: USD

Open Positions | Transaction History

	Order ID	Units	Side	Market	Time	Price	Take Profit	Stop Loss
1	991800972	1000	sell	EUR_USD	2015-06-12 06:25:10	1.12426	0	0

Figure 20: Modify or Close Order Dialog

Figure 21 demonstrates a pop-up dialog for when an order is placed. It displays the date and time, the take profit and stop loss chosen, the better broker chosen to trade with, and whether it is a Buy or a Sell trade. A similar dialog will appear on alert-only mode, minus the execution price. Similarly, Figure 22 show two alert dialogs superimposed on the main program window.


Trade Alert - AAGFA

Trade has been placed

Type: Buy
Execution Price: 1.1243
Take Profit: 1.1246
Stop Loss: 1.124

Date/Time: June 11 2015 at 23:57

Broker: Oanda

OK

Trade Alert - AAGFA

Trade has been placed

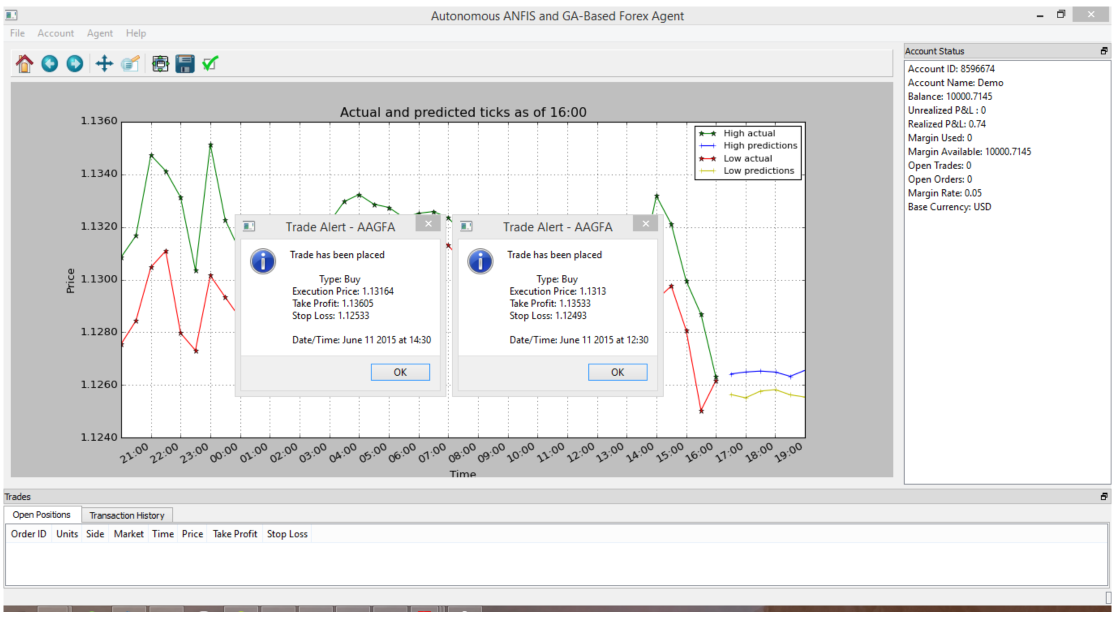
Type: Sell
Execution Price: 1.124
Take Profit: 1.1229
Stop Loss: 1.1251

Date/Time: June 11 2015 at 23:59

Broker: Pseudo

OK

Figure 21: Pop-up Dialog Alerts for trades detected and executed on either Oanda or Simulated broker



Trade Alert - AAGFA

Trade has been placed

Type: Buy
Execution Price: 1.13164
Take Profit: 1.13605
Stop Loss: 1.12533

Date/Time: June 11 2015 at 14:30

Trade Alert - AAGFA

Trade has been placed

Type: Buy
Execution Price: 1.1313
Take Profit: 1.13533
Stop Loss: 1.12493

Date/Time: June 11 2015 at 12:30

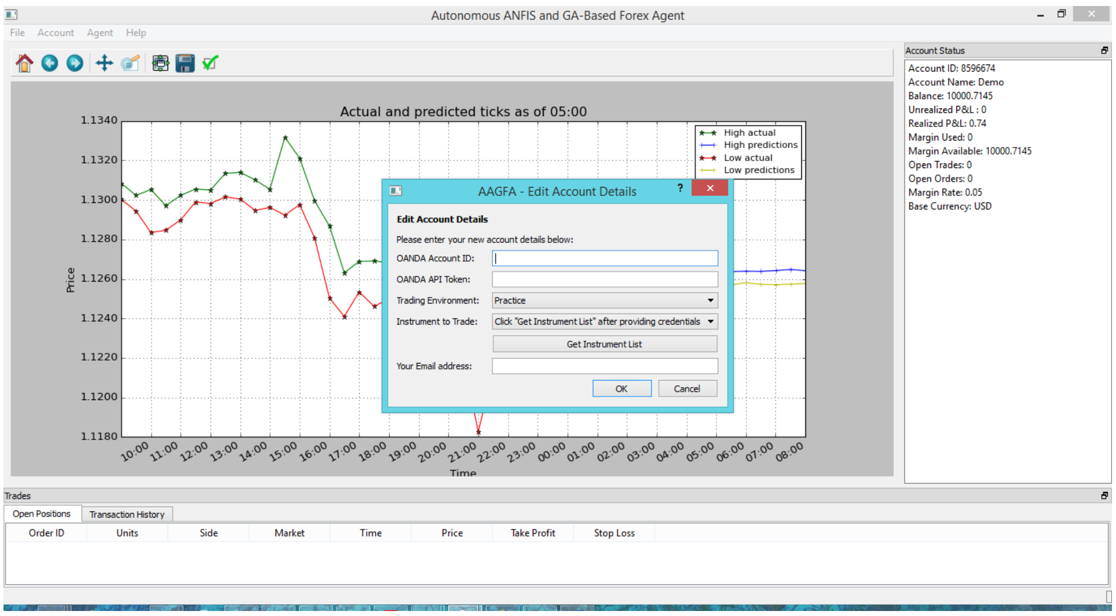
Account Status

Account ID: 8596674
Account Name: Demo
Balance: 10000.7145
Unrealized P&L : 0
Realized P&L: 0.74
Margin Used: 0
Margin Available: 10000.7145
Open Trades: 0
Open Orders: 0
Margin Rate: 0.05
Base Currency: USD

Figure 22: Two actual trades placed displaying corresponding notifications

We also give the user the ability to switch accounts, change alert email address, or the instrument to trade with. In this line, an edit account details dialog is provided, as shown on Figure 23

Lastly, Figure 24 is an actual alert email sent to the email address on file.



Account Status

Account ID: 8596674
Account Name: Demo
Balance: 10000.7145
Unrealized P&L : 0
Realized P&L: 0.74
Margin Used: 0
Margin Available: 10000.7145
Open Trades: 0
Open Orders: 0
Margin Rate: 0.05
Base Currency: USD

AAGFA - Edit Account Details

Edit Account Details

Please enter your new account details below:

OANDA Account ID:

OANDA API Token:

Trading Environment: Practice

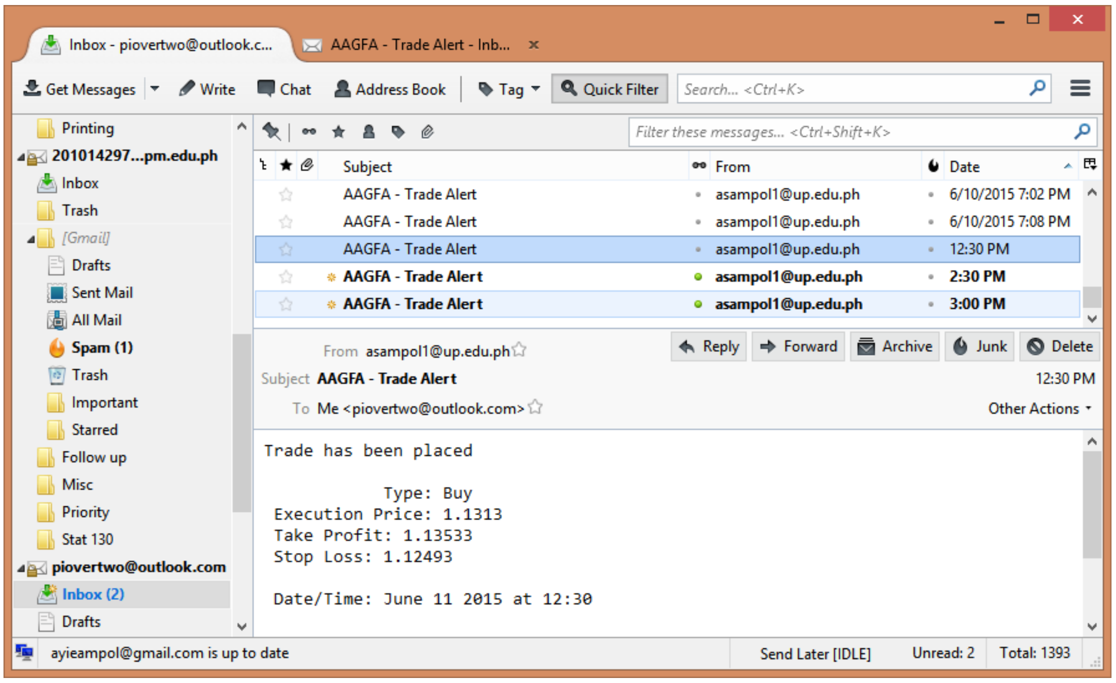
Instrument to Trade: Click "Get Instrument List" after providing credentials

Get Instrument List

Your Email address:

OK Cancel

Figure 23: Edit Account Details Dialog while main program is running



Subject	From	Date
AAGFA - Trade Alert	asampol1@up.edu.ph	6/10/2015 7:02 PM
AAGFA - Trade Alert	asampol1@up.edu.ph	6/10/2015 7:08 PM
AAGFA - Trade Alert	asampol1@up.edu.ph	12:30 PM
AAGFA - Trade Alert	asampol1@up.edu.ph	2:30 PM
AAGFA - Trade Alert	asampol1@up.edu.ph	3:00 PM

From asampol1@up.edu.ph
Subject **AAGFA - Trade Alert** 12:30 PM
To Me <piovertwo@outlook.com>

```
Trade has been placed

            Type: Buy
 Execution Price: 1.1313
 Take Profit: 1.13533
 Stop Loss: 1.12493

 Date/Time: June 11 2015 at 12:30
```

ayieampol@gmail.com is up to date   Send Later [IDLE]   Unread: 2   Total: 1393

Figure 24: Actual email alerts sent by the program

VII. Discussions

The paper by Al-Refaie et. al. [6] which was the basis of this study traded the EUR/USD pair. As such, this program was also executed on the EUR/USD pair. The program allows the user to work on any currency pair that their broker allows—in confidence that the algorithm could be extended to any Forex pair—but it must be noted that the research and testing was done over EUR/USD.

The agent currently running and being live-tested was trained using a two-year data set from April 2013 to 2015, following the pattern on Al-Refaie et. al. [6] The reference testing set used to tune the decision making module is the two-month data set immediately following the training set.

Training Data Set	April 2013 - April 2015
Testing Data Set	April 2015 - May 2015
Backtesting Test Set 1	April 2015 - May 2015
Backtesting Test Set 2	January - December 2014

Table 1: Data Set Segmentation

The profits calculated if the agent were run over a historical two-month and one-year period are approximately $80 and $500, respectively, as seen on the actual graph produced by the backtesting module of the program below. These profit amounts were seen to be independent of the initial deposit as long as it is greater than $1,000, as the program trades 1,000 units at every execution. Although Al-refaie had an initial balance of $1,000, while these backtests assumed $10,000, they can still be comparable if we were to assume 10:1 leverage—that is, the available trading amount is $10,000 for a $1,000 deposit. Compared in the table below are the average percent profits per month.

Agent	Gross Profit	Time Period	Average Profit per Month
Al-Refaie	$1,102	January 2011 - July 2012	$16.22
AAGFA (1)	$80	April 2015 - May 2015	$40
AAGFA (2)	$500	January 2014 - December 2014	$41.67

Table 2: Agent Profit Comparison

Although the AAGFA backtesting results appear more profitable at first glance, it must be noted that 1) the backtesting module did not incorporate any commissions or spreads which almost certainly will impact the gross profit, and 2) these were tested during different periods, thus it may or may not be the case that the period backtested for AAGFA was simply more ripe with trading opportunities.

Besides backtesting simulations, the agent has also demonstrated that it can open positions which result in profit. Seen in Figures 27 and 28 are actual trans-

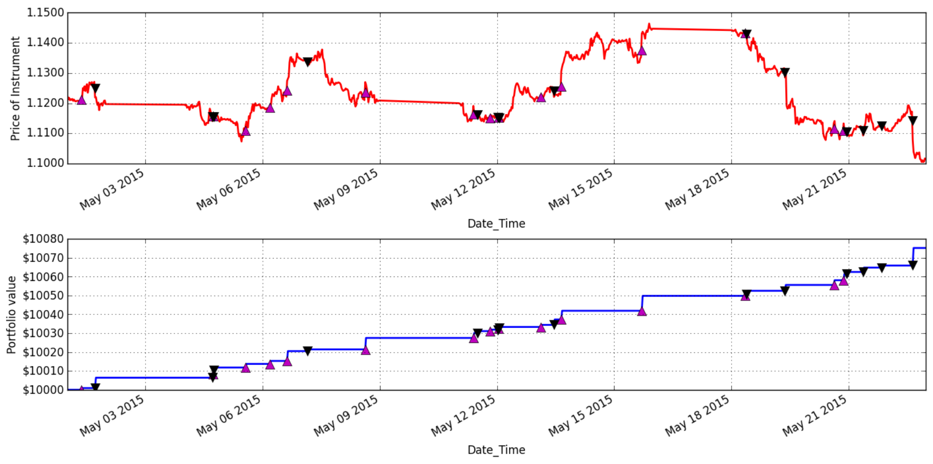


Figure 25: Backtesting Close Price and Equity Curve Plot - Two Months

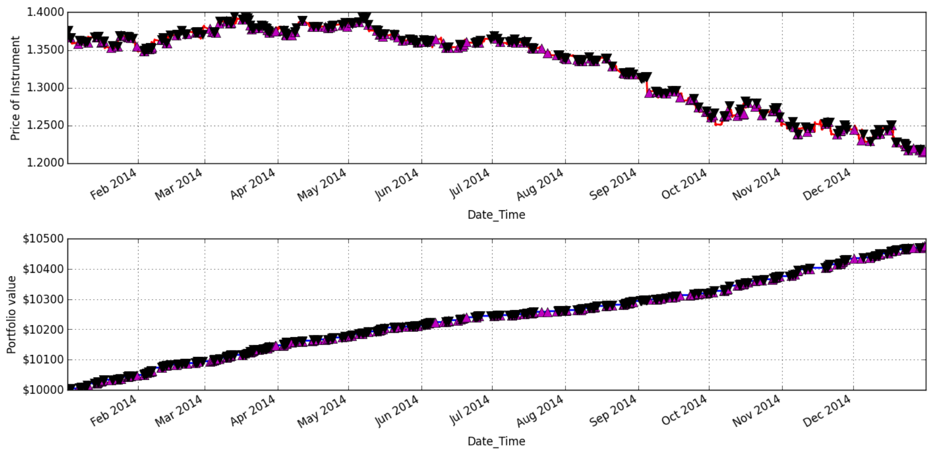
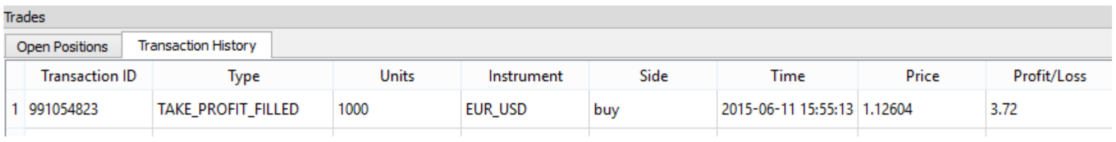


Figure 26: Backtesting Close Price and Equity Curve Plot - One Year

action history logs that show the closing of a trade placed by the broker, resulting in profit.

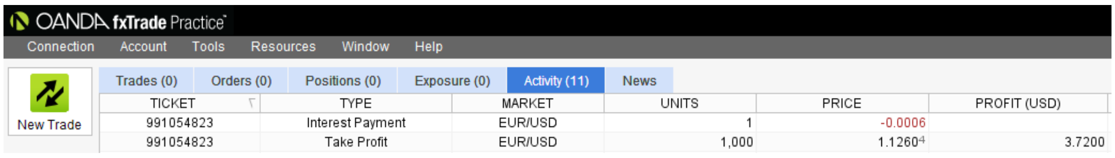

Trades

Open Positions | Transaction History

	Transaction ID	Type	Units	Instrument	Side	Time	Price	Profit/Loss
1	991054823	TAKE_PROFIT_FILLED	1000	EUR_USD	buy	2015-06-11 15:55:13	1.12604	3.72

Figure 27: Profitable Trade as seen in AAGFA

Besides profit, it would also be prudent to look at and consider the accuracy of the predictor. Presented below is the graph of the Mean Squared Error from Al-refaie's paper and a table of the MSE's of the ANFIS of AAGFA.



TICKET	TYPE	MARKET	UNITS	PRICE	PROFIT (USD)
991054823	Interest Payment	EUR/USD	1	-0.0006	
991054823	Take Profit	EUR/USD	1,000	1.1260^4	3.7200

Figure 28: Profitable Trade as seen in Oanda fxTrade

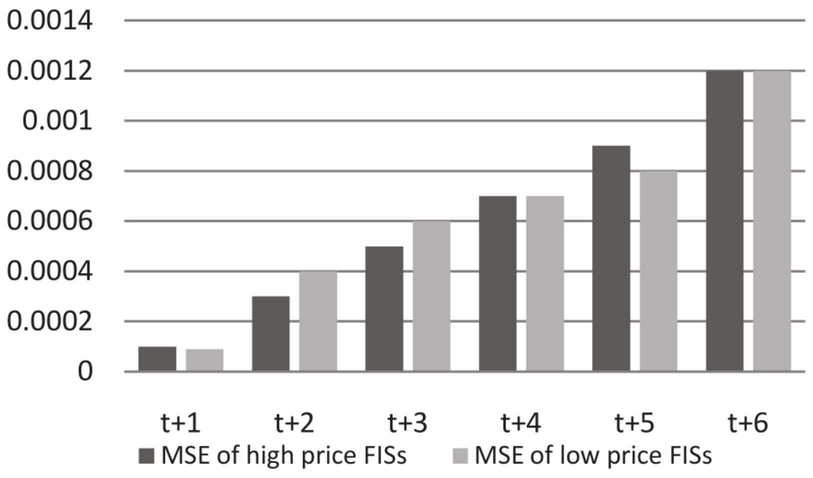


Figure 29: MSE of Al-Refaie's ANFIS Models

Time	High	Low
$(t+1)$	0.0001085	0.0001083
$(t+2)$	0.0002186	0.0002185
$(t+3)$	0.0003477	0.0003484
$(t+4)$	0.0004908	0.0004922
$(t+5)$	0.0006445	0.0006468
$(t+6)$	0.0008058	0.0001085

Table 3: AAGFA ANFIS Root Mean Squared Error

While the resulting RMSE's were comparable to Al-refaie's, initial test runs of the ANFIS prediction sometimes produced anomalous results visually. The root of this was traced back to the EMA smoothing utilized. Initially, the output for the ANFIS was also taken from the smoothed series. The pre-processing function was modified to pair the smoothed inputs with the raw output. This caused the output predictions to behave more as expected, and MSE was even lowered.

VIII. Conclusions

This study looked into creating an automated trader that is based upon prediction provided by the Adaptive Neuro-Fuzzy Inference System and trade timing by the Non-Dominated Sorting Genetic Algorithm. It was realized as a desktop application written using Python with the PyQt4 framework, connected to MATLAB using the MATLAB Engine for Python. The smart agent can connect to the user's Oanda brokerage account using it's REST Web API to retrieve prices, account status, and place orders. The same feature was also demonstrated to work with a simulated broker with a generic HTTP API similar to Oanda.

Based on backtesting results and empirical evidence from live trading, the AAGFA's strategy is able to produce profits when trading fully automatically. Its prediction accuracy is also at par with the original implementation by Al-Refaie et. al. In addition to this, the agent provides capabilities for user intervention by allowing editing and closing of open positions, and going on alert-only mode, which only notifies the user, but does not perform trades.

Being compatible with a demo Oanda account, the AAGFA can be evaluated by any user to paper trade on their demo account so that they may see its ability when running live and decide for themselves to hook it up to an actual trading account. This will allow them to come to the same conclusion as this study: that the Autonomous ANFIS and GA-based Forex Agent is a reliable automated trader that produces profit.

IX. Recommendations

While this agent is currently tailored to view account status and perform actual trades over the Oanda API, the system also has concurrent functionality to place orders and retrieve previously sent orders to a simulated broker which offers a similar Web API, also made by the developer. This establishes the extensibility of the program to work with any HTTP API similar to that provided by Oanda and our simulated broker, with a minimally reasonable amount of effort. For developers wanting to work on this point, it may be a worthwhile undertaking to support brokers with FIX (Financial Information Exchange) API's, or to work on building wrappers to expose MetaTrader functions to Python and port the AAGFA accordingly.

As Al-refaie observed, Mean Squared Error for training was minimized when the input was passed through a smoothing filter, as this reduces noise in the input. They utilized an exponential moving average (EMA) to smooth the data. However, no specified EMA lookback period was specified, thus leading to the decision of using the default time period of the EMA function of TA-Lib which is 30. This may be a point of further optimization in the future.

Since the trader can only function as long as it's open as a desktop program and is connected to the Internet, another possible avenue for improvement would be to extract just the trading loop's functionality and have it run as a daemon on a remote server. This way, a user may choose to rent a VPS or other remote computing platform like Amazon Web Services which offers almost 100% uptime to ensure that the agent captures all potential trades.

Finally, for the convenience of the user who would want to work with multiple instruments, an option to initially train over several currencies may be developed. This will allow the user to switch instruments while the program is running without having to interrupt trading just to re-train the predictors.

X. Bibliography

[1] S. Yao, M. Pasquier, and C. Quek, “A foreign exchange portfolio management mechanism based on fuzzy neural networks,” in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, 09 2007, pp. 2576–2583.

[2] B. Gharleghi, A. H. Shaari, and N. Shafighi, “Predicting exchange rates using a novel ”cointegration based neuro-fuzzy system”,” *International Economics*, vol. 137, no. 0, pp. 88 – 103, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2110701713000528

[3] J. Murphy, *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*, ser. New York Institute of Finance Series. New York Institute of Finance, 1999. [Online]. Available: https://books.google.co.in/books?id=5zhXEqdr_IcC

[4] Technical indicator. [Online]. Available: http://www.investopedia.com/terms/t/technicalindicator.asp

[5] M. Chlistalla, “High-frequency trading: Better than its reputation?” Deutsche Bank Research, Mar. 2011. [Online]. Available: http://www.dbresearch.com/PROD/DBR_INTERNET_DE-PROD/PROD0000000000270960.pdf

[6] M. Alrefaie, A.-A. Hamouda, and R. Ramadan, “A smart agent to trade and predict foreign exchange market,” in *Computational Intelligence for Engineering Solutions (CIES), 2013 IEEE Symposium on*, 04 2013, pp. 141–148.

[7] E. Chan, *Quantitative Trading: How to Build Your Own Algorithmic Trading Business*, ser. Wiley Trading. Wiley, 2009. [Online]. Available: http://books.google.com.ph/books?id=NZlV0M5Ije4C

[8] K. Kim and J. Kaljuvee, *Electronic and Algorithmic Trading Technology*, ser. Complete Technology Guides for Financial Services. Boston: Academic

Press, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780123724915500033

[9] C. Garner, *Currency Trading in the Forex and Futures Markets.* FT Press, 2012. [Online]. Available: http://books.google.com.ph/books?id=LkEf6cVMbKcC

[10] D. Bell and L. Gana, "Algorithmic trading systems: A multifaceted view of adoption," in *System Science (HICSS), 2012 45th Hawaii International Conference on*, 01 2012, pp. 3090–3099.

[11] R. Barbosa and O. Belo, "Autonomous forex trading agents," in *Advances in Data Mining. Medical Applications, E-Commerce, Marketing, and Theoretical Aspects*, ser. Lecture Notes in Computer Science, P. Perner, Ed. Springer Berlin Heidelberg, 2008, vol. 5077, pp. 389–403. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-70720-2_30

[12] N. Gradojevic and R. Gencay, "Fuzzy logic, trading uncertainty and technical trading," *Journal of Banking & Finance*, vol. 37, no. 2, pp. 578 – 586, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378426612002889

[13] J.-Y. Wu and C. jie Lu, "Computational intelligence approaches for stock price forecasting," in *Computer, Consumer and Control (IS3C), 2012 International Symposium on*, 06 2012, pp. 52–55.

[14] D. Khoonmirzaie, S. Rasouli, and E. Farrokhi, "Designing perceptron three-layered neural network for predicting dollar-franc currency pair in international exchange market," *Journal of mathematics and computer science*, 2010.

[15] W. H. L. Yu, S. Wang and K. K. Lai, "Are foreign exchange rates predictable? a literature review from artificial neural networks perspective," in *Foreign-Exchange-Rate Forecasting With Artificial Neural*

Networks. Springer US, 2007, vol. 107, pp. 3–23. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-71720-3_1

[16] A. A. Baasher and M. W. Fakhr, "Forex trend classification using machine learning techniques," in *Proceedings of the 11th WSEAS International Conference on Applied Computer Science*, ser. ACS'11. World Scientific and Engineering Academy and Society (WSEAS), 2011, pp. 41–47. [Online]. Available: http://dl.acm.org/citation.cfm?id=2051254.2051263

[17] N. Siddique and H. Adeli, *Fuzzy Systems and Applications*. John Wiley & Sons Ltd, 2013, pp. 65–101. [Online]. Available: http://dx.doi.org/10.1002/9781118534823.ch3

[18] L. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338 – 353, 1965. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S001999586590241X

[19] L.-Y. Wei, C.-H. Cheng, and H.-H. Wu, "A hybrid ANFIS based on n-period moving average model to forecast TAIEX stock," *Applied Soft Computing*, vol. 19, no. 0, pp. 86 – 92, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494614000416

[20] J.-S. Jang, "Anfis: adaptive-network-based fuzzy inference system," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 23, no. 3, pp. 665–685, May 1993.

[21] Z. Tan, C. Quek, and P. Y. Cheng, "Stock trading with cycles: A financial application of {ANFIS} and reinforcement learning," *Expert Systems with Applications*, vol. 38, no. 5, pp. 4741 – 4755, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095741741000905X

[22] G. Nuti, M. Mirghaemi, P. Treleaven, and C. Yingsaeree, "Algorithmic trading," *Computer*, vol. 44, no. 11, pp. 61–69, 11 2011.

[23] M. A. H. Dempster, T. Payne, Y. Romahi, and G. W. P. Thompson, "Computational learning techniques for intraday fx trading using popular technical indicators," *Neural Networks, IEEE Transactions on*, vol. 12, no. 4, pp. 744–754, 07 2001.

[24] A. Hirabayashi, C. Aranha, and H. Iba, "Optimization of the trading rule in foreign exchange using genetic algorithm," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '09. ACM, 2009, pp. 1529–1536. [Online]. Available: http://doi.acm.org/10.1145/1569901.1570106

[25] S. Papadamou and G. Stephanides, "Improving technical trading systems by using a new matlab-based genetic algorithm procedure," *Mathematical and Computer Modelling*, vol. 46, no. 12, pp. 189 – 197, 2007, proceedings of the International Conference on Computational Methods in Sciences and Engineering 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0895717707000386

[26] R. Kuo, C. Chen, and Y. Hwang, "An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network," *Fuzzy Sets and Systems*, vol. 118, no. 1, pp. 21 – 45, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165011498003996

[27] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[28] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, Jun. 2000. [Online]. Available: http://dx.doi.org/10.1162/106365600568202

[29] O. Rifki and H. Ono, "A survey of computational approaches to portfolio optimization by genetic algorithms," O. Rifki and H. Ono, Eds. Society for Computational Economics, 2012-06, Conference Paper. [Online]. Available: http://id.nii.ac.jp/0001/00021853

[30] L.-Y. Wei, "A hybrid model based on {ANFIS} and adaptive expectation genetic algorithm to forecast {TAIEX}," *Economic Modelling*, vol. 33, no. 0, pp. 893 – 899, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0264999313002253

[31] D. Ghose, S. Panda, and P. Swain, "Prediction and optimization of runoff via {ANFIS} and {GA}," *Alexandria Engineering Journal*, vol. 52, no. 2, pp. 209 – 220, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1110016813000082

[32] (2014, Oct.) Statistical arbitrage - definition and other information. [Online]. Available: http://www.hedgefund-index.com/d_statarb.asp

[33] S. C. Lee and E. T. Lee, "Fuzzy neural networks," *Mathematical Biosciences*, vol. 23, 1975.

[34] M. Buragohain, "Adaptive network based fuzzy inference system (anfis) as a tool for system identification with special emphasis on training data minimization," July.

[35] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, ser. Wiley Interscience Series in Systems and Optimization. Wiley, 2001. [Online]. Available: http://books.google.com.ph/books?id=OSTn4GSy2uQC

[36] A. Godinez, L. Espinosa, and E. Montes, "An experimental comparison of multiobjective algorithms: Nsga-ii and omopso," in *Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2010*, 08 2010, pp. 28–33.

[37] Matlab engine for python. [Online]. Available: http://www.mathworks.com/help/matlab/matlab-engine-for-python.html

[38] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.

[39] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed 2015-06-11]. [Online]. Available: http://www.scipy.org/

[40] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011. [Online]. Available: http://scitation.aip.org/content/aip/journal/cise/13/2/10.1109/MCSE.2011.37

[41] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. [Online]. Available: http://scitation.aip.org/content/aip/journal/cise/9/3/10.1109/MCSE.2007.55

[42] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 51 – 56.

XI. Appendix

A. Source Code

```
'''
Created on May 20, 2015
Main UI Window
@author: ArielKenneth
'''

from __future__ import unicode_literals

import platform
import sys
import time
import datetime

from PyQt4.QtCore import (PYQT_VERSION_STR, QSettings, QT_VERSION_STR, QVariant, Qt, QDir)
from PyQt4.QtGui import (QAction, QApplication, QDockWidget, QDialog, QFrame, QIcon,
                         QLabel, QListWidget, QMainWindow, QMessageBox, QTableWidget,
    QErrorMessage, QTabWidget, QFileDialog, QWidget, QVBoxLayout)
from matplotlib.backends.backend_qt4agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.backends.backend_qt5 import NavigationToolbar2QT as NavigationToolbar
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
from matplotlib import dates
import matplotlib.ticker as mtick


from aagfa import account_setup_wizard, trading_loop, gui_update_thread,\
    modify_dialog, oanda_funcs, edit_details, retrain_dialog, training_dialog,\
    email_alert, backtest_diag, strategy, pseudo_broker
from aagfa import config
import oandapy
from matplotlib.dates import DateFormatter


class Window(QMainWindow):

    def __init__(self, parent=None):
        super(Window, self).__init__(parent)
        self.setObjectName("MainWindow")
        self.create_widgets()
        self.create_actions()
        self.load_settings()
        self.setWindowTitle(config._program_name)

        # !!!Only for testing purposes
#         self.settings.setValue("app/firstRun", True)

        self.run_gui_update_loop()
        self.run_trading_loop()

        self.display_oanda = True

#         self.tradeDetectedNotif("Buy", 1.1234, 1.0123)

    def create_widgets(self):
        self.create_central_plot()

        #create status dock
        accountStatsWidget = QDockWidget("Account Status", self)
        accountStatsWidget.setFeatures(QDockWidget.DockWidgetFloatable |
                QDockWidget.DockWidgetMovable)
        accountStatsWidget.setObjectName("StatusWidget")
        accountStatsWidget.setAllowedAreas(Qt.LeftDockWidgetArea|
                                    Qt.RightDockWidgetArea)
        self.listAccountWidget = QListWidget()
        self.listAccountWidget.setSelectionMode(QListWidget.NoSelection)
        accountStatsWidget.setWidget(self.listAccountWidget) #CHANGE THIS TO SOMETHING APPROPRIATE
        self.addDockWidget(Qt.RightDockWidgetArea, accountStatsWidget)

        #create open positions dock
        positionsWidget = QDockWidget("Trades", self)
        positionsWidget.setFeatures(QDockWidget.DockWidgetFloatable |
                QDockWidget.DockWidgetMovable)
        positionsWidget.setObjectName("PositionsWidget")
        positionsWidget.setAllowedAreas(Qt.BottomDockWidgetArea)
        tabs = QTabWidget()
        self.openPositionsTable = QTableWidget()
        self.transHistoryTable = QTableWidget()
        tabs.addTab(self.openPositionsTable, "Open Positions")
        tabs.addTab(self.transHistoryTable, "Transaction History")
        positionsWidget.setWidget(tabs)
        self.addDockWidget(Qt.BottomDockWidgetArea, positionsWidget)

        open_pos_labels = ["Order ID", "Units", "Side", "Market", "Time", "Price", "Take Profit", "Stop Loss"]
        self.openPositionsTable.setColumnCount(len(open_pos_labels))
        self.openPositionsTable.setHorizontalHeaderLabels(open_pos_labels)
        self.openPositionsTable.setEditTriggers(QTableWidget.NoEditTriggers)
        self.openPositionsTable.setSelectionBehavior(QTableWidget.SelectRows)
        trans_history_labels = ["Transaction ID", "Type", "Units", "Instrument","Side","Time", "Price", "Profit
        self.transHistoryTable.setColumnCount(len(trans_history_labels))
        self.transHistoryTable.setHorizontalHeaderLabels(trans_history_labels)
        self.transHistoryTable.setEditTriggers(QTableWidget.NoEditTriggers)
        self.transHistoryTable.setSelectionMode(QTableWidget.NoSelection)

        #create status bar
        self.sizeLabel = QLabel()
        self.sizeLabel.setFrameStyle(QFrame.StyledPanel|QFrame.Sunken)
        status = self.statusBar()
        status.setSizeGripEnabled(False)
        status.addPermanentWidget(self.sizeLabel)

        #create error message
        self.errorMessageDialog = QErrorMessage()
        self.errorMessageDialog.setWindowTitle(config._program_name_short + " - Error")

    def create_central_plot(self):
        '''
        Idea obtained from http://matplotlib.org/examples/user_interfaces/embedding_in_qt4.html
        '''
        plot_widget = QWidget()
        plot_widget.setObjectName("centralPlot")

        # a figure instance to plot on
        self.figure = plt.figure()

        # this is the Canvas Widget that displays the `figure`
        # it takes the `figure` instance as a parameter to __init__
        self.canvas = FigureCanvas(self.figure)

        # this is the Navigation widget
        # it takes the Canvas widget and a parent
        self.toolbar = NavigationToolbar(self.canvas, self)

        # set the layout
        layout = QVBoxLayout(self)
        layout.setObjectName("thisone")
        layout.addWidget(self.toolbar)
        layout.addWidget(self.canvas)
        plot_widget.setLayout(layout)
        self.setCentralWidget(plot_widget)


    def create_actions(self):
        #ADD SLOTS in the future as necessary
        newAccountAction = self.createAction("&Start Setup Wizard",
                shortcut="Ctrl+N", tip="Enter New Account Details", slot=self.rerun_wizard)
        closeAction = self.createAction("&Close",
                shortcut="Alt+F4", tip="Close agent and stop execution", slot=self.close)
        editAccountAction = self.createAction("&Edit Account Details",
                shortcut="Ctrl+E", tip="Edit trading account credentials", slot=self.edit_details)
        toggleViewAction = self.createAction("S&witch Oanda/Pseudo",
                shortcut="Ctrl+W", tip="Toggle account status view to the other account", slot=self.switch_oand
        backtestAction = self.createAction("Perform a &Backtest",
                shortcut="Ctrl+B", tip="Perform a backtest based on provided data set/time period", slot=self.b
        toggleAction = self.createAction("Toggle Auto Mode On/Off",
                shortcut="Ctrl+A", tip="Auto mode allows the agent to perform trades automatically; otherwise,
                slot=self.toggle_auto)
        retrainAction = self.createAction("&Re-train Predictors",
                shortcut="Ctrl+R", tip="Feed new data into ANFIS", slot=self.retrain)
        startAction = self.createAction("&Start Agent",
                shortcut="Ctrl+S", tip="Activate automated trader", slot=self.start_trading)
        stopAction = self.createAction("S&top Agent",
                shortcut="Ctrl+T", tip="Halt trader from processing orders", slot=self.stop_trading)
        about_action = self.createAction("&About", slot=self.helpAbout, tip="About This Program")

        fileMenu = self.menuBar().addMenu("&File")
        accountMenu = self.menuBar().addMenu("&Account")
        agentMenu = self.menuBar().addMenu("&Agent")
        helpMenu = self.menuBar().addMenu("&Help")

        self.addActions(fileMenu, (newAccountAction,closeAction))
        self.addActions(accountMenu, (editAccountAction,backtestAction, toggleViewAction))
        self.addActions(agentMenu, (retrainAction, startAction, stopAction, toggleAction))
        self.addActions(helpMenu, (about_action,))

    def load_settings(self):
        self.settings = QSettings()
        self.restoreGeometry(
                self.settings.value("MainWindow/Geometry").toByteArray())
        self.restoreState(self.settings.value("MainWindow/State").toByteArray())
        self.api_token = str(self.settings.value("app/api_token").toString())
        self.account_id = str(self.settings.value("app/account_id").toString())
        self.instrument = str(self.settings.value("app/instrument").toString())
        self.environment = str(self.settings.value("app/environment").toString())
        self.testing_path = str(self.settings.value("data/testing_path").toString())
        self.auto_mode = self.settings.value("app/automode").toBool()
        self.email = str(self.settings.value("app/email").toString())
        print "Email loaded:", str(self.settings.value("app/email").toString())

        try:
            high_hist = [x.toFloat()[0] for x in self.settings.value("data/highHist").toList()]
            high_pred = [x.toFloat()[0] for x in self.settings.value("data/highPred").toList()]
            low_hist = [x.toFloat()[0] for x in self.settings.value("data/lowHist").toList()]
            low_pred = [x.toFloat()[0] for x in self.settings.value("data/lowPred").toList()]
            self.firstPlot(high_hist, high_pred, low_hist, low_pred)
        except Exception:
            self.firstPlot([], [], [], [])
```

```
def createAction(self, text, slot=None, shortcut=None, icon=None,
                 tip=None, checkable=False):
    action = QAction(text, self)
    if icon is not None:
        action.setIcon(QIcon(":/{0}.png".format(icon)))
    if shortcut is not None:
        action.setShortcut(shortcut)
    if tip is not None:
        action.setToolTip(tip)
        action.setStatusTip(tip)
    if slot is not None:
        action.triggered.connect(slot)
    if checkable:
        action.setCheckable(True)
    return action

def setup_trading_signals(self):
    #for info messages to display in status bar or something
    self.trading_loop.trading_info_signal.connect(self.updateStatus)
    #for error messages to display in status bar or something
    self.trading_loop.trading_error_signal.connect(self.showErrorDialog)
    # for urgent trade placement notification
    self.trading_loop.trade_placed_signal.connect(self.tradePlacedNotif)
    # for new ticks - args are 1) high hist 2) high pred 3) low hist 4) low pred
    self.trading_loop.new_ticks_signal.connect(self.plotNewValues)

def run_trading_loop(self):

    if self.settings.value("app/firstRun", defaultValue=True).toBool():
        self.firstRun()
    else:
        self.running =  self.settings.value("app/trade_loop_running").toBool()
        self.auto_mode =  self.settings.value("app/automode").toBool()
        self.trading_loop = trading_loop.TradingLoop(self.running, self.auto_mode)
    self.setup_trading_signals()
    # TODO Turn this on when needed
    self.trading_loop.start()
    self.updateStatus("Trading loop started")

def run_gui_update_loop(self):
    self.gui_update_loop = gui_update_thread.GuiUpdateThread(
                            self.openPositionsTable, self.transHistoryTable, self.listAccountWidget)
    self.setup_gui_signals()
    self.gui_update_loop.start()

def setup_gui_signals(self):
    self.openPositionsTable.cellDoubleClicked.connect(self.modify_trade)
    self.gui_update_loop.status_bar_update.connect(self.updateStatus)

def closeEvent(self, event):
    if self.okToContinue():
        self.settings.setValue("MainWindow/Geometry", QVariant(
                        self.saveGeometry()))
        self.settings.setValue("MainWindow/State", QVariant(
                        self.saveState()))
        self.settings.setValue("app/trade_loop_running", QVariant(
                            self.running))
        self.settings.setValue("app/automode", QVariant(
                            self.auto_mode))
        event.accept()
    else:
        event.ignore()

def okToContinue(self):
    reply = QMessageBox.question(self, 'Message',
        "Are you sure you want to quit?", QMessageBox.Yes |
        QMessageBox.No, QMessageBox.No)

    return reply == QMessageBox.Yes


def addActions(self, target, actions):
    for action in actions:
        if action is None:
            target.addSeparator()
        else:
            target.addAction(action)

def updateStatus(self, message):
    self.statusBar().showMessage(message, 5000)

def showErrorDialog(self, message):
    self.errorMessageDialog.showMessage(message +
        "<br />Date/Time: {0}".format(time.strftime("%B %d %Y at %H:%M")))

def tradePlacedNotif(self, side, execution_price, take_profit, stop_loss, broker_chosen):
    time_now = time.strftime("%B %d %Y at %H:%M")
    message= '''Trade has been placed\n
        Type: {0} \n Execution Price: {3} \n Take Profit: {1} \n Stop Loss: {2}
        \n Date/Time: {4}
        \n Broker: {5}'''.format(side, take_profit, stop_loss, execution_price, time_now, broker_chosen)
    email_alert.send_email_alert(self.email, message)
    QMessageBox.information(self, "Trade Alert - {0}".format(config._program_name_short), message)

def tradeDetectedNotif(self, side, take_profit, stop_loss):
    time_now = time.strftime("%B %d %Y at %H:%M")
    message = '''Trade has been detected\n
        Type: {0} \n Take Profit: {1} \n Stop Loss: {2}
```

```
            \n Date/Time: {3}’’’.format(side, take_profit, stop_loss, time_now)
        email_alert.send_email_alert(self.email, message)
        QMessageBox.information(self, ”Trade Alert - {0}”.format(config._program_name_short), message)

    def switch_oanda(self):
        self.gui_update_loop.account_display_changed.emit()
        if self.display_oanda:
            self.updateStatus(”Now displaying Simulated Broker account data”)
            self.display_oanda = False
        else:
            self.updateStatus(”Now displaying Oanda account data”)
            self.display_oanda = True
        assert self.display_oanda == self.gui_update_loop.display_oanda

    def plotNewValues(self, high_hist, high_pred, low_hist, low_pred):

        # time labels
        hist_time_values = [datetime.datetime.now() + datetime.timedelta(minutes = 30 * x) for x in range(1-con
        pred_time_values = [datetime.datetime.now() + datetime.timedelta(minutes = 30 * x) for x in range(1,7)]
        hist_time_values = dates.date2num(hist_time_values)
        pred_time_values = dates.date2num(pred_time_values)

        # create an axis
        ax = self.figure.add_subplot(111)

        # discards the old graph
        ax.cla()

#          print ’DEBUG: max of each series from plotting:’, max(high_hist), max(high_pred), max(low_hist), max(

        try:
            formatter = DateFormatter(’%H:%M’)
            ax.xaxis.set_major_formatter(formatter)
            ax.xaxis.set_major_locator(dates.HourLocator())
            ax.yaxis.set_major_formatter(mtick.FormatStrFormatter(’%.4f’))
#              loc = mtick.MultipleLocator(base=0.001) # this locator puts ticks at regular intervals
#              ax.yaxis.set_major_locator(loc)

            # plot data
            ax.plot(hist_time_values, high_hist, ’g*-’)
            ax.hold(True)
            ax.plot(pred_time_values, high_pred, ’b+-’)
            ax.plot(hist_time_values, low_hist, ’r*-’)
            ax.plot(pred_time_values, low_pred, ’y+-’)

            ax.grid()

            plt.setp(ax.get_xticklabels(), rotation=30, horizontalalignment=’right’)

            #set labels
            ax.axes.set_xlabel(’Time’)
            ax.axes.set_ylabel(’Price’)
            ax.axes.set_title(’Actual and predicted ticks as of {0}’.format(time.strftime(”%H:%M”)))
            fontP = FontProperties()
            fontP.set_size(’small’)
            ax.legend([’High actual’, ’High predictions’, ’Low actual’, ’Low predictions’], ”best”, prop = fontP

            # refresh canvas
            self.canvas.draw()
            self.updateStatus(”New values plotted”)
        except Exception as e:
            ax.axes.set_title(’No values yet’)
            print str(e)

        #for future reference
        self.settings.setValue(”data/highHist”, QVariant.fromList(high_hist))
        self.settings.setValue(”data/highPred”, QVariant.fromList(high_pred))
        self.settings.setValue(”data/lowHist”, QVariant.fromList(low_hist))
        self.settings.setValue(”data/lowPred”, QVariant.fromList(low_pred))

    def firstPlot(self, high_hist, high_pred, low_hist, low_pred):

        # random data
        hist_time_values = range(1-config._historical_ticks,1)
        pred_time_values = range(1,7)

        # create an axis
        ax = self.figure.add_subplot(111)

        # discards the old graph
        ax.cla()

        try:
            if high_hist:
                ax.yaxis.set_major_formatter(mtick.FormatStrFormatter(’%.4f’))
#                  loc = mtick.MultipleLocator(base=0.001) # this locator puts ticks at regular intervals
#                  ax.yaxis.set_major_locator(loc)


            # plot data
            ax.plot(hist_time_values, high_hist, ’g*-’)
            ax.hold(True)
            ax.plot(pred_time_values, high_pred, ’b+-’)
            ax.plot(hist_time_values, low_hist, ’r*-’)
            ax.plot(pred_time_values, low_pred, ’y+-’)

            ax.grid()

            #set labels
            ax.axes.set_xlabel('Time')
            ax.axes.set_ylabel('Price')
            ax.axes.set_title('Values from last retrieval')
            fontP = FontProperties()
            fontP.set_size('small')
            ax.legend(['High actual', 'High predictions', 'Low actual', 'Low predictions'], "best", prop = fontP
        except Exception:
            ax.axes.set_title('No values yet')

        # refresh canvas
        self.canvas.draw()
        self.updateStatus("New values plotted")

        #for future reference
        self.settings.setValue("data/highHist", QVariant.fromList(high_hist))
        self.settings.setValue("data/highPred", QVariant.fromList(high_pred))
        self.settings.setValue("data/lowHist", QVariant.fromList(low_hist))
        self.settings.setValue("data/lowPred", QVariant.fromList(low_pred))

    def start_trading(self):
        self.running = True
        self.trading_loop.trading_loop_state_changed.emit(True)

    def stop_trading(self):
        self.running = False
        self.trading_loop.trading_loop_state_changed.emit(False)

    def modify_trade(self, row, col):
        trade_id = self.openPositionsTable.item(row, 0).text()
        units = self.openPositionsTable.item(row, 1).text()
        side = self.openPositionsTable.item(row, 2).text()
        instrument = self.openPositionsTable.item(row, 3).text()
        exec_time = self.openPositionsTable.item(row, 4).text()
        exec_price = self.openPositionsTable.item(row, 5).text()
        take_profit = self.openPositionsTable.item(row, 6).text()
        stop_loss = self.openPositionsTable.item(row, 7).text()

        mod_dialog = ModDialog(trade_id, units, side, instrument, exec_time, exec_price,
            take_profit, stop_loss, self.account_id, self.api_token, self.environment)

        if mod_dialog.exec_():
            tp, sl, action_type = mod_dialog.get_tp_sl_type(self.display_oanda)
            try:
                oanda = oanda_funcs.create_instance(self.api_token, self.environment)
                if action_type == 'modify' and self.display_oanda:
                    oanda_funcs.modify_trade(oanda, self.account_id, trade_id, tp, sl)
                elif action_type == 'close' and self.display_oanda:
                    oanda.close_trade(self.account_id, trade_id)
                elif action_type == 'modify' and not self.display_oanda:
                    pseudo_broker.modify_order(order_id=int(trade_id), tp=tp, sl=sl)
                else:
                    print pseudo_broker.close_order(order_id=int(trade_id))
                QMessageBox.information(self, config._program_name_short, "Modify/Close trade successful.", but
            except UnboundLocalError:
                self.showErrorDialog("Modify/Close cannot be made. Check your connection.")
            except oandapy.OandaError as oe:
                self.showErrorDialog("Server returned error for modify/close trade requests.\n"
                                     "Error: {0}".format(oe))
            except pseudo_broker.BrokerError:
                self.showErrorDialog("Broker returned error for modify/close trade requests.")

### ACTION FUNCTIONS ###
    def helpAbout(self):
        QMessageBox.about(self, "About " + config._program_name_short,
                """<b>{5}</b> v {0}
                <p>Copyright &copy; 2015 Ariel Kenneth Ampol
                All rights reserved.
                <p>An Implementation of
                <a href='http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6611741'>
                A smart agent to trade and predict foreign exchange market </a>
                by Alrefaie et. al
                <p>Python {1} - Qt {2} - PyQt {3} on {4}""".format(
                config._version, platform.python_version(),
                QT_VERSION_STR, PYQT_VERSION_STR,
                platform.system(), config._program_name))

    def firstRun(self):
        wizard = account_setup_wizard.SetupWizard()
        wizard.exec_()
        start_now = wizard.start_checkbox_value
        auto_mode = wizard.automode
        self.settings.setValue("app/firstRun", False)
        self.trading_loop = trading_loop.TradingLoop(start_now, auto_mode)
        print "Start now?", start_now

    def rerun_wizard(self):
        reply = QMessageBox.question(self, 'Message',
            "Are you sure you want to re-run the wizard? "
            "This will halt the agent from working with the current account.", QMessageBox.Yes |
            QMessageBox.No, QMessageBox.No)

        if reply == QMessageBox.Yes:
            self.stop_trading()
            try:
                wizard = account_setup_wizard.SetupWizard()
                wizard.exec_()
                self.running = wizard.start_checkbox_value
                auto_mode = wizard.automode
                self.load_settings()
                self.settings.setValue("app/firstRun", False)
                self.trading_loop = trading_loop.TradingLoop(self.running, auto_mode)
                self.gui_update_loop.account_info_changed.emit()
                print "Start now?", self.running
            except Exception as e:
                self.showErrorDialog("Error encountered after re-running wizard.\n"
                                         "Error: {0}".format(e))

    def edit_details(self):
        self.stop_trading()
        edit_diag = EditDialog()
        try:
            if edit_diag.exec_():
                QMessageBox.information(self, config._program_name_short, "Account info updated.", buttons=QMes
                self.trading_loop.account_info_changed.emit()
                self.gui_update_loop.account_info_changed.emit()
                #reload settings
                self.api_token = str(self.settings.value("app/api_token").toString())
                self.account_id = str(self.settings.value("app/account_id").toString())
                self.instrument = str(self.settings.value("app/instrument").toString())
                self.environment = str(self.settings.value("app/environment").toString())
                self.email = str(self.settings.value("app/email").toString())
        except Exception as e:
            self.showErrorDialog("Could not update account details.\n"
                                         "Error: {0}".format(e))
        self.start_trading()

    def retrain(self):
        self.stop_trading()
        retrain_diag = RetrainDialog()
        try:
            retrain_diag.exec_()
        except Exception as e:
            self.showErrorDialog("Could not update account details.\n"
                                         "Error: {0}".format(e))
        self.start_trading()

    def toggle_auto(self):
        self.trading_loop.toggle_auto()
        self.auto_mode = self.trading_loop.auto_mode

    def backtest(self):
        dialog = BacktestDiag(self.instrument, self.testing_path)

        if dialog.exec_():
            filename = str(dialog.ui.file_text.text())
            initial_amount = float(dialog.ui.amount_text.text())

            bars, signals, returns = strategy.backtest(self.instrument, filename, initial_amount)

            main = strategy.Window(bars, signals, returns)
            main.setWindowTitle(config._program_name + ' - Backtesting Results')
            time.sleep(2)
            main.showMaximized()

class BacktestDiag(QDialog):
    def __init__(self, instrument, default_test_path):
        QDialog.__init__(self)

        # Set up the user interface from Designer.
        self.ui = backtest_diag.Ui_Dialog()
        self.ui.setupUi(self)
        self.setWindowTitle(config._program_name_short + " - Backtesting")
        self.ui.file_text.setText(default_test_path)

        self.instrument = instrument
        self.ui.choose_button.clicked.connect(self.showDialog)

    def showDialog(self):
        dialog = QFileDialog(self)
        fname = dialog.getOpenFileName(self, 'Open file', QDir.homePath())
        self.ui.file_text.setText(fname)

class ModDialog(QDialog):
    def __init__(self, order_id, units, side, instrument, exec_time,
            exec_price, take_profit, stop_loss, account_id, api_token, environment):
        QDialog.__init__(self)

        # Set up the user interface from Designer.
        self.ui = modify_dialog.Ui_Dialog()
        self.ui.setupUi(self)
        self.setWindowTitle(config._program_name_short + " - Modify/Close Order")

        self.take_profit = take_profit
        self.stop_loss = stop_loss

        # Connect up the buttons.
        self.ui.buttonBox.accepted.connect(self.accept)
        self.ui.buttonBox.rejected.connect(self.reject)

        #Modifications
        header = "<b><p align=\"center\">{0} {1} units {2} at {3}</p></b>".format(
                    str.upper(str(side)), units, instrument, exec_price)
        self.ui.main_info.setText(header)
        self.ui.order_id_value.setText(order_id)
        self.ui.time_value.setText(exec_time)
        self.ui.tp_lineedit.setText(take_profit)
        self.ui.sl_lineedit.setText(stop_loss)
        quote_type = ('ask' if side == 'sell' else 'bid')
        oanda = oanda_funcs.create_instance(api_token, environment)
        latest_price = oanda_funcs.latest_price(oanda, str(instrument), quote_type)
#         print "Fetch price params:", str(instrument), quote_type
        latest_price_line = "<b><p align=\"center\">Last Quote: {0}</p></b>".format(latest_price)
        self.ui.last_quote.setText(latest_price_line)

    def get_tp_sl_type(self, oanda_mode):
        tp = float(self.ui.tp_lineedit.text())
        sl = float(self.ui.sl_lineedit.text())
        if oanda_mode:
            tp = tp if tp <> float(self.take_profit) else None
            sl = sl if sl <> float(self.stop_loss) else None

        action_type = "close" if self.ui.close_pos_button.isChecked() else "modify"

        return tp, sl, action_type

class EditDialog(QDialog):
    def __init__(self, parent=None):
        QDialog.__init__(self)

        # Set up the user interface from Designer.
        self.ui = edit_details.Ui_Dialog()
        self.ui.setupUi(self)
        self.setWindowTitle(config._program_name_short + " - Edit Account Details")
        self.ui.get_list_button.clicked.connect(self.get_instruments)

    def accept(self):
        configs = QSettings()
        api_token = self.ui.api_token_text.text()
        account_id = self.ui.account_id_text.text()
        instrument = self.ui.instrumen_combobox.currentText()
        environment = str(self.ui.environment_comboBox.currentText()).lower()
        email = self.ui.email_text.text()

        configs.setValue("app/api_token", api_token)
        configs.setValue("app/account_id", account_id)
        configs.setValue("app/instrument", instrument)
        configs.setValue("app/environment", environment)
        configs.setValue("app/email", email)

        super(EditDialog, self).accept()

    def get_instruments(self):
        try:
            api_token = str(self.ui.api_token_text.text())
            account_id = str(self.ui.account_id_text.text())
            environment = str(self.ui.environment_comboBox.currentText()).lower()
            oanda = oanda_funcs.create_instance(api_token, environment)
            available_instruments = oanda_funcs.just_the_instruments(oanda, account_id)
            self.ui.instrumen_combobox.clear()
            for instrument in available_instruments:
                self.ui.instrumen_combobox.addItem(instrument)
        except oandapy.OandaError as error:
            QMessageBox.critical(self, 'Error','Account ID or API token not accepted: ' + str(error) )
            pass

class RetrainDialog(QDialog):
    def __init__(self, parent=None):
        QDialog.__init__(self)

        # Set up the user interface from Designer.
        self.ui = retrain_dialog.Ui_Dialog()
        self.ui.setupUi(self)
        self.setWindowTitle(config._program_name_short + " - Retrain Predictors")

        self.ui.training_file_chooser_button.clicked.connect(lambda: self.showDialog(self.ui.training_data_path
        self.ui.testing_file_chooser_button.clicked.connect(lambda: self.showDialog(self.ui.testing_data_path))
        self.ui.begin_training_button.clicked.connect(self.accept)

    def accept(self):
        super(RetrainDialog, self).accept()
        training_path = str(self.ui.training_data_path.text())
        testing_path = str(self.ui.testing_data_path.text())
        training_diag = training_dialog.TrainingDialog(training_path, testing_path)
        training_diag.exec_()

    def showDialog(self, line_edit_to_change):
        dialog = QFileDialog(self)
        fname = dialog.getOpenFileName(self, 'Open file', QDir.homePath())
        line_edit_to_change.setText(fname)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    app.setOrganizationName(config._organization)
    app.setApplicationName(config._program_name)

    main = Window()
    main.showMaximized()

    sys.exit(app.exec_())

'''
Created on May 27, 2015

@author: ArielKenneth
'''
import sys
import StringIO
import time
import datetime

from PyQt4 import QtCore, QtGui
import oandapy
import talib
import numpy as np
import pandas as pd

import config
import matlab_funcs
from aagfa.oanda_funcs import latest_price
from aagfa import oanda_funcs, pseudo_broker

LONG = 1
SHORT = -1
HOLD = 0

class TradingLoop(QtCore.QThread):

    # use this signal to flip trading loop Start/Stop coming from GUI
    trading_loop_state_changed = QtCore.pyqtSignal(bool)
    # use this as signal to refresh account info when user changes it
    account_info_changed = QtCore.pyqtSignal()
    #for informational messages to display in status bar or something
    trading_info_signal = QtCore.pyqtSignal(str)
    #for error messages to display in status bar or something
    trading_error_signal = QtCore.pyqtSignal(str)
    # for urgent trade placement notification - args: side, price, T/P, S/L, chosenbroker
    trade_placed_signal = QtCore.pyqtSignal(str, float, float, float, str)
    # for urgent trade detection notification - args: side, T/P, S/L
    # for "alert-only" mode
    trade_detected_signal = QtCore.pyqtSignal(str, float, float)
    # for new ticks - args are 1) high hist 2) high pred 3) low hist 4) low pred
    new_ticks_signal = QtCore.pyqtSignal(list, list, list, list)


    def __init__(self, start_now, auto_mode=True, parent=None):
        super(TradingLoop, self).__init__(parent)
        self.running = start_now
        self.auto_mode = auto_mode
        self.establish_signals_and_slots()
        self.fetch_settings()

    def establish_signals_and_slots(self):
        self.trading_loop_state_changed.connect(self.change_state)
        self.account_info_changed.connect(self.fetch_settings)

    def fetch_settings(self):
        settings = QtCore.QSettings()
        self.api_token = str(settings.value("app/api_token").toString())
        self.account_id = str(settings.value("app/account_id").toString())
        self.instrument = str(settings.value("app/instrument").toString())
        self.environment = str(settings.value("app/environment").toString())
        self.auto_mode = settings.value("app/automode").toBool()

    def change_state(self, running):
        '''
        Start or stop the trading loop according to bool running
        '''
        print "change_state called", running
        if running:
            if running == self.running:
                self.trading_info_signal.emit("Agent already running!")
            else:
                self.running = True
                self.trading_info_signal.emit("Agent has been started")
        else:
            if running == self.running:
                self.trading_info_signal.emit("Agent already stopped!")
            else:
                self.running = False
                self.trading_info_signal.emit("Agent has been halted")

    def toggle_auto(self):
        '''
        Switch between auto mode on/off
        '''
        if self.auto_mode:
            self.auto_mode = False
            self.trading_info_signal.emit("Auto mode set to OFF")
        else:
            self.auto_mode = True
            self.trading_info_signal.emit("Auto mode set to ON")

    # Recommended by https://joplaete.wordpress.com/2010/07/21/threading-with-pyqt4/
    def __del__(self):
        print 'Destructor called'
        self.wait()

    def run(self):
        '''
        Main trading loop time control
        '''
        if self.running:
            self.trading_info_signal.emit("Trading loop has started")
        else:
            self.trading_info_signal.emit("Trading loop is currently halted")
        self.oanda = oandapy.API(self.environment, self.api_token)
        self.load_MATLAB()
        self.outstanding_long = None
        self.outstanding_short = None

        #To prevent re-looping within the minute
        just_looped = False
        while True:
            if self.running and time.strftime('%M') in ("08", "30") and not just_looped:
                self.trading_info_signal.emit("Trading processing commenced")
                self.main_loop()
                time.sleep(60) # to prevent re-looping within the minute
                just_looped = True
            else:
                time.sleep(2)
                just_looped = False

        return

    def load_MATLAB(self):
        self.trading_info_signal.emit("MATLAB engine being loaded")
        self.eng = matlab_funcs.init_matlab()
        self.trading_info_signal.emit("MATLAB engine has been loaded")

    def main_loop(self):
        '''
        Main trading loop time control
        '''
        high_predictions, low_predictions = [], []

        # Retrieve past 41 points
        self.trading_info_signal.emit("Fetching latest ticks from server")
        try:
            rates = self.oanda.get_history(
                        instrument=self.instrument, granularity=config._granularity,
                        count=config._historical_ticks+config._ema_window-1, candleFormat="midpoint")
        except UnboundLocalError:
            self.trading_error_signal.emit("Trading info could not be retrieved. Check your connection.")
            print "Couldn't retrieve data at ", datetime.datetime.now().time()
            return
        except oandapy.OandaError as oe:
            self.trading_error_signal.emit("Error received from server while fetching price.\n"
                                          "Error: {0}".format(oe))

        high_ticks = [x['highMid'] for x in rates['candles']]
        low_ticks = [x['lowMid'] for x in rates['candles']]
        close_ticks = [x['closeMid'] for x in rates['candles']]

#         print 'DEBUG: high_ticks:', high_ticks

        ### Predict next 6 points
        self.trading_info_signal.emit("Smoothing data")
        ema_high = talib.EMA(np.array(high_ticks), config._ema_window)[config._ema_window-1:].tolist()
        ema_low = talib.EMA(np.array(low_ticks), config._ema_window)[config._ema_window-1:].tolist()
        assert len(ema_high) == config._historical_ticks

        self.trading_info_signal.emit("Predicting next points")
        out = StringIO.StringIO()
        err = StringIO.StringIO()
        for i in range(1, config._horizon + 1):
            high_predictions.append(
                matlab_funcs.anfis_predict(self.eng, ema_high, 'high_{0}.fis'.format(i), stdout=out, stderr=err
            low_predictions.append(
                matlab_funcs.anfis_predict(self.eng, ema_low, 'low_{0}.fis'.format(i), stdout=out, stderr=err))
        if err.len > 0:
            self.trading_error_signal.emit(err.getvalue())

        ### Send signal for plotting
        self.new_ticks_signal.emit(
            high_ticks[config._ema_window-1:], high_predictions, low_ticks[config._ema_window-1:], low_predictio

        ### Compute technical indicators
        self.trading_info_signal.emit("Computing technical indicators")
        close_5_ema = pd.ewma(pd.Series(close_ticks), span=5)
        close_12_ema = pd.ewma(pd.Series(close_ticks), span=12)
#         close_5_ema = talib.EMA(np.asarray(close_ticks), timeperiod=5)
#         close_12_ema = talib.EMA(np.asarray(close_ticks), timeperiod=12)
        close_rsi = talib.RSI(np.asarray(close_ticks), timeperiod=8)
        #Get RSI slope
        xaxis = np.arange(0,config._trend_line_count)
        linear_fit = np.polyfit(xaxis, close_rsi[-5:], 1)
        slope = linear_fit[0]

        ### Check for crossovers
        self.trading_info_signal.emit("Identifying trading opportunities")
        peak = max(high_predictions)
        trough = max(low_predictions)
        if (close_5_ema.iloc[-1] >= close_12_ema.iloc[-1]) and (close_5_ema.iloc[-2] <= close_12_ema.iloc[-2]):
            self.outstanding_long = [LONG,peak,trough]
#             print 'DEBUG pred_high, pred_low: ', high_predictions, low_predictions
        elif (close_5_ema.iloc[-1] <= close_12_ema.iloc[-1]) and (close_5_ema.iloc[-2] >= close_12_ema.iloc[-2]
            self.outstanding_short = [SHORT,trough,peak]
#             print 'DEBUG pred_high, pred_low: ', high_predictions, low_predictions

        ### DUMMY EMIT/ERROR
#         self.trade_placed_signal.emit('Buy', 1.2345, 1.5432, 1.0000)
#         self.trading_error_signal.emit("Dummy: Received unexpected response from server. "
#                                        "Please check fxTrade for any changes that may have occurred.")
```

```
        ### For now, let’s just log the positions
        ### Study these positions against real data tomorrow
        print datetime.datetime.now().time(), self.outstanding_long, self.outstanding_short, slope
#         self.trade_placed_signal.emit(”Dummy”, 1.4321, 1.0000, 1.2345)

        ## Place order if necessary/timely - i.e. RSI is above/below 50 and rising/falling accordingly
        if self.outstanding_long and (close_rsi[-1] > 50 and slope > 0):
            if self.place_order(self.outstanding_long):
                self.outstanding_long = None
        elif self.outstanding_short and (close_rsi[-1] < 50 and slope < 0):
            if self.place_order(self.outstanding_short):
                self.outstanding_short = None

    def place_order(self, position):
        ’’’
        Place an order
        position - (side, T/P, S/L)
        ’’’
        order_side = ’buy’ if position[0] == LONG else ’sell’
        quote_type = ’ask’ if position[0] == LONG else ’bid’
        oanda_current = oanda_funcs.latest_price(self.oanda, self.instrument, quote_type)
        pseudo_current = pseudo_broker.get_current_price(inst=self.instrument)
        print ”Current ticks: (Oanda, Dummy)”, oanda_current, pseudo_current
        tp = position[1]+config._pip_adjustment if position[0] == LONG else position[1]-config._pip_adjustment
        tp = float(”%.5f” % tp)
        sl = position[2]-(config._pip_adjustment*1.5) if position[0] == LONG else position[2]+(config._pip_adjus
        sl = float(”%.5f” % sl)
        print ”Place order: (position, TP, SL)”, position[0], tp, sl, datetime.datetime.now()

        if not self.auto_mode:
            self.trade_placed_signal.emit(order_side.capitalize(), tp, sl)
            return

#         ### FOR TESTING PURPOSES DO BOTH (for live, uncomment block below)
#         pseudo_success = pseudo_broker.place_order(
#             acc_id=self.account_id, side=order_side, inst=self.instrument, units=config._order_units, tp=tp,
#         print ”Pseudo success?”, pseudo_success
#         return self.oanda_order(order_side, tp, sl)

        ## Determine where to place order
        if position[0] == LONG:
            if oanda_current < pseudo_current:
                return self.oanda_order(order_side, tp, sl)
            else:
                exec_price = pseudo_broker.place_order(
                    self.account_id, order_side, self.instrument, config._order_units, tp, sl)
                if exec_price:
                    self.trade_placed_signal.emit(order_side.capitalize(), exec_price, tp, sl, ”Pseudo”)
                else:
                    self.trading_error_signal.emit(”Pseudo broker trade place failed”)
        else:
            if oanda_current > pseudo_current:
                return self.oanda_order(order_side, tp, sl)
            else:
                exec_price = pseudo_broker.place_order(
                    self.account_id, order_side, self.instrument, config._order_units, tp, sl)
                if exec_price:
                    self.trade_placed_signal.emit(order_side.capitalize(), exec_price, tp, sl, ”Pseudo”)
                else:
                    self.trading_error_signal.emit(”Pseudo broker trade place failed”)


    def oanda_order(self, order_side, tp, sl):
        try:
            response = self.oanda.create_order(self.account_id, instrument=self.instrument, units=config._order
                side=order_side, type=’market’, takeProfit=tp, stopLoss=sl)
            price = response[’price’]
            self.trade_placed_signal.emit(order_side.capitalize(), price, tp, sl, ”Oanda”)
        except oandapy.OandaError as oe:
            # try to remedy via manual edit
            try:
                #stop loss is the problem
                if oe.get_code() == 33 and order_side == SHORT:
                    position = (order_side, tp, sl+config._pip_adjustment)
                    response = self.oanda_order(position)
                elif oe.get_code() == 33 and order_side == LONG:
                    position = (order_side, tp, sl-config._pip_adjustment)
                    response = self.oanda_order(position)
                # take profit is the problem
                elif oe.get_code() == 34 and order_side == SHORT:
                    position = (order_side, tp-config._pip_adjustment, sl)
                    response = self.oanda_order(position)
                elif oe.get_code() == 34 and order_side == LONG:
                    position = (order_side, tp+config._pip_adjustment, sl)
                    response = self.oanda_order(position)
                else:
                    self.trading_error_signal.emit(”Trade detected could not be placed. Error encountered.”
                                              ”\nError: {0}”.format(oe))
                    return False
            except oandapy.OandaError as oe2:
                self.trading_error_signal.emit(”Trade detected could not be placed. Error encountered.”
                                          ”\nError: {0}”.format(oe2))
                return False
        except KeyError:
            self.trading_error_signal.emit(”Received unexpected response from server. ”
                                      ”Please check fxTrade for any changes that may have occurred.”)
            return False

        return True
    def pseudo_order(self):
        pass

def main():
    app = QtGui.QApplication(sys.argv)
    app.setOrganizationName(config._organization)
    app.setApplicationName(config._program_name)
    sample = TradingLoop(True)
    sample.start()

    sys.exit(app.exec_())

if __name__ == '__main__':
    main()


% ANFIS Training Script
function trainError = anfis_init_train(dataFilename, fisFilename)
        % Build initial FIS and train as ANFIS data from filename
        % Uses current working directory for reading and writing
        % That means, this should be in the matlab script/data folder
        % dataFilename - filename of CSV training set
        % fisFilename - whole filename of FIS to be saved
        % trainError - returns False in case of failure, otherwise the training error vector as is

        try
                % Set current directory to script's location
                cd(fileparts(mfilename('fullpath')));
                trnData = csvread(dataFilename);

                % Build initial FIS structure using Subtractive Clustering
                X = trnData(:,1:end-1);
                Y = trnData(:,end);
                radii = 0.5; %What could be optimal?
                in_fis = genfis2(X,Y,radii);

                % Run training
                % dispOpt = zeros(1,4); %don't display anything on the command window
                dispOpt = [1 1 1 1];
                trnOpt = [NaN NaN NaN NaN NaN];
                % try 15
                [out_fis,trainError] = anfis(trnData,in_fis,trnOpt,dispOpt)

                % Save FIS to file
                writefis(out_fis, fisFilename);

        catch ME
                trainError = false
        end

%%% Additional notes: Based on profiler, takes approx 22 mins each on full training set
%%% Works good on GUI using absolute and relative paths for filename
%%% Now also verified to be working in matlab_template.py


function result = fis_predict(inputData, fisFilename)
        % Wraps the evalfis function for calling from Python
        % inputData - data for evaluation as array/matrix
        % fisFilename - whole filename of FIS to be saved
        % result returned as is from evalfis()

        fismat = readfis(fisFilename);
        result = evalfis(inputData, fismat);

%%% Additional notes: Based on some random manual checking, accurate within 1-7 pips
%%% Works good on GUI using absolute paths for filename


'''
Account setup wizard to call on first run or when user wants to
'''

import time
import StringIO

from PyQt4 import QtCore, QtGui
from oandapy import OandaError
from PyQt4.Qt import QDir, QSettings

import matlab_funcs
import oanda_funcs
import preprocess

import config

try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    def _fromUtf8(s):
        return s

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig)
```

```
class SetupWizard(QtGui.QWizard):
    def __init__(self, parent=None):
        super(SetupWizard, self).__init__(parent)

        self.addPage(IntroPage())
        self.account_details_page = AccountDetailsPage()
        self.addPage(self.account_details_page)
        self.addPage(HistoricalDataPage())
        self.addPage(TrainingPage())
        #TODO!
        #self.addPage(BacktestingPage())
        self.addPage(ConclusionPage())

        self.setWindowTitle("AAGFA - Setup Wizard")

    def accept(self):
        print "I'm in accept!"
        api_token = str(self.field("api_token").toString())
        account_id = str(self.field("account_id").toString())
        training_path = str(self.field("training_data").toString())
        testing_path = str(self.field("testing_data").toString())
        email = str(self.field("email").toString())
        instrument, environment = self.account_details_page.get_selected()
        print "From setup: check vars"
        print instrument, environment, api_token, account_id, testing_path, training_path, email

        self.start_checkbox_value = self.field("start_checkbox").toBool()
        self.automode = self.field("automode_checkbox").toBool()
        print "Automode?", self.automode
        print "Checkbox value from setup", self.start_checkbox_value

        configs = QSettings()
        configs.setValue("app/api_token", api_token)
        configs.setValue("app/account_id", account_id)
        configs.setValue("app/instrument", instrument)
        configs.setValue("app/environment", environment)
        configs.setValue("app/email", email)
        print configs.value("app/email").toString()
        configs.setValue("app/automode", self.automode)
        print "Automode saved?", configs.value("app/automode").toBool()
        configs.setValue("data/training_path", training_path)
        configs.setValue("data/testing_path", testing_path)

        super(SetupWizard, self).accept()


class IntroPage(QtGui.QWizardPage):
    def __init__(self, parent=None):
        super(IntroPage, self).__init__(parent)

        self.setTitle("Welcome to AAGFA!")

        label = QtGui.QLabel("This wizard will help you set up the account details "
                             "and input the historical data files needed for the "
                             "Autonomous ANFIS and GA-based Forex Agent to begin trading."
                             "\n\n\nReminder: Before you continue with this wizard, "
                             "please ensure that the system requirements have been met "
                             "and that any dependencies have been installed. "
                             "For further information, please refer to the attached README.txt")
        label.setWordWrap(True)

        layout = QtGui.QVBoxLayout()
        layout.addWidget(label)
        self.setLayout(layout)


class AccountDetailsPage(QtGui.QWizardPage):
    def __init__(self, parent=None):
        super(AccountDetailsPage, self).__init__(parent)

        self.setObjectName("AccountDetailsPage")
        self.setTitle("Account Details")
        self.setSubTitle("Please enter your account details below:")

        self.verticalLayout = QtGui.QVBoxLayout()
        self.verticalLayout.setObjectName(_fromUtf8("verticalLayout"))
        self.gridLayout = QtGui.QGridLayout()
        self.gridLayout.setObjectName(_fromUtf8("gridLayout"))
        self.api_token_text = QtGui.QLineEdit()
        self.api_token_text.setObjectName(_fromUtf8("api_token_text"))
        self.gridLayout.addWidget(self.api_token_text, 1, 1, 1, 1)
        self.label_4 = QtGui.QLabel()
        self.label_4.setMinimumSize(QtCore.QSize(111, 0))
        self.label_4.setObjectName(_fromUtf8("label_4"))
        self.gridLayout.addWidget(self.label_4, 3, 0, 1, 1)
        self.label_2 = QtGui.QLabel()
        self.label_2.setMaximumSize(QtCore.QSize(101, 16777215))
        self.label_2.setObjectName(_fromUtf8("label_2"))
        self.gridLayout.addWidget(self.label_2, 0, 0, 1, 1)
        self.account_id_text = QtGui.QLineEdit()
        self.account_id_text.setObjectName(_fromUtf8("account_id_text"))
        self.gridLayout.addWidget(self.account_id_text, 0, 1, 1, 1)
        self.label_3 = QtGui.QLabel()
        self.label_3.setObjectName(_fromUtf8("label_3"))
        self.gridLayout.addWidget(self.label_3, 1, 0, 1, 1)
        self.instrument_combobox = QtGui.QComboBox()
        self.instrument_combobox.setObjectName(_fromUtf8("instrument_combobox"))
        self.instrument_combobox.addItem(_fromUtf8(""))
        self.gridLayout.addWidget(self.instrument_combobox, 3, 1, 1, 1)
        self.label_5 = QtGui.QLabel()
        self.label_5.setObjectName(_fromUtf8("label_5"))
        self.gridLayout.addWidget(self.label_5, 2, 0, 1, 1)
        self.environment_comboBox = QtGui.QComboBox()
        self.environment_comboBox.setObjectName(_fromUtf8("environment_comboBox"))
        self.environment_comboBox.addItem(_fromUtf8(""))
        self.environment_comboBox.addItem(_fromUtf8(""))
        self.gridLayout.addWidget(self.environment_comboBox, 2, 1, 1, 1)
        self.get_list_button = QtGui.QPushButton()
        self.get_list_button.setObjectName(_fromUtf8("get_list_button"))
        self.get_list_button.clicked.connect(self.get_instruments)
        self.gridLayout.addWidget(self.get_list_button, 4, 1, 1, 1)
        self.verticalLayout.addLayout(self.gridLayout)
        self.label_4.setBuddy(self.instrument_combobox)
        self.label_2.setBuddy(self.account_id_text)
        self.label_3.setBuddy(self.api_token_text)
        self.label_5.setBuddy(self.environment_comboBox)
        self.email_label = QtGui.QLabel("Your Email address:")
        self.email_text = QtGui.QLineEdit()
        self.email_label.setBuddy(self.email_text)
        self.gridLayout.addWidget(self.email_label, 5, 0, 1, 1)
        self.gridLayout.addWidget(self.email_text, 5, 1, 1, 1)

        #Validators
        int_valid = QtGui.QIntValidator()
        regexp = QtCore.QRegExp('[^@]+@[^@]+\.[^@]+')
        email_valid = QtGui.QRegExpValidator(regexp)
        self.account_id_text.setValidator(int_valid)
        self.email_text.setValidator(email_valid)

        self.setLayout(self.verticalLayout)
        self.retranslateUi()

        #register fields
        self.registerField("account_id*", self.account_id_text)
        self.registerField("api_token*", self.api_token_text)
        self.registerField("instrument*", self.instrument_combobox)
        self.registerField("environment", self.environment_comboBox)
        self.registerField("email*", self.email_text)

        #set tab order
        self.setTabOrder(self.account_id_text, self.api_token_text)
        self.setTabOrder(self.api_token_text, self.environment_comboBox)
        self.setTabOrder(self.environment_comboBox, self.instrument_combobox)

        #Validators
        int_valid = QtGui.QIntValidator()
        regexp = QtCore.QRegExp('[^@]+@[^@]+\.[^@]+')
        email_valid = QtGui.QRegExpValidator(regexp)
        self.account_id_text.setValidator(int_valid)
        self.email_text.setValidator(email_valid)

    def retranslateUi(self):
        self.label_4.setText(_translate("AccountDetailsPage", "Instrument to Trade:", None))
        self.label_2.setText(_translate("AccountDetailsPage", "OANDA Account ID:", None))
        self.label_3.setText(_translate("AccountDetailsPage", "OANDA API Token:", None))
        self.instrument_combobox.setItemText(0, _translate("AccountDetailsPage", "Click \"Get Instrument List\"
        self.label_5.setText(_translate("AccountDetailsPage", "Trading Environment:", None))
        self.environment_comboBox.setItemText(0, _translate("AccountDetailsPage", "Practice", None))
        self.environment_comboBox.setItemText(1, _translate("AccountDetailsPage", "Live", None))
        self.get_list_button.setText(_translate("AccountDetailsPage", "Get Instrument List", None))

    def get_instruments(self):
        try:
            api_token = str(self.api_token_text.text())
            account_id = str(self.account_id_text.text())
            environment = str(self.environment_comboBox.currentText()).lower()
            oanda = oanda_funcs.create_instance(api_token, environment)
            available_instruments = oanda_funcs.just_the_instruments(oanda, account_id)
            self.instrument_combobox.clear()
            for instrument in available_instruments:
                self.instrument_combobox.addItem(instrument)
        except OandaError as error:
            QtGui.QMessageBox.about(self, 'Error','Account ID or API token not accepted: ' + str(error) )
            pass

    def get_selected(self):
        '''
        Returns a tuple of currentText of instrument and environment (in lowercase), in this order
        '''
        inst = str(self.instrument_combobox.currentText())
        env = str(self.environment_comboBox.currentText()).lower()
        return inst, env

class HistoricalDataPage(QtGui.QWizardPage):
    def __init__(self, parent=None):
        super(HistoricalDataPage, self).__init__(parent)

        self.setTitle("Historical Data Files")
        self.setSubTitle("Please provide the CSV files for the historical prices below:\n")
        self.setObjectName("HistoricalDataPage")

        self.verticalLayout = QtGui.QVBoxLayout()
        self.verticalLayout.setObjectName(_fromUtf8("verticalLayout"))
        self.gridLayout = QtGui.QGridLayout()
        self.gridLayout.setObjectName(_fromUtf8("gridLayout"))
        self.label_2 = QtGui.QLabel()
        self.label_2.setMaximumSize(QtCore.QSize(101, 16777215))
        self.label_2.setObjectName(_fromUtf8("label_2"))
        self.gridLayout.addWidget(self.label_2, 0, 0, 1, 1)
        self.training_data_path = QtGui.QLineEdit()
        self.training_data_path.setObjectName(_fromUtf8("training_data_path"))
        self.gridLayout.addWidget(self.training_data_path, 0, 1, 1, 1)
        self.training_file_chooser_button = QtGui.QPushButton()
        self.training_file_chooser_button.setObjectName(_fromUtf8("training_file_chooser_button"))
        self.gridLayout.addWidget(self.training_file_chooser_button, 0, 2, 1, 1)
        self.label_3 = QtGui.QLabel()
        self.label_3.setObjectName(_fromUtf8("label_3"))
        self.gridLayout.addWidget(self.label_3, 1, 0, 1, 1)
        self.testing_data_path = QtGui.QLineEdit()
        self.testing_data_path.setObjectName(_fromUtf8("testing_data_path"))
        self.gridLayout.addWidget(self.testing_data_path, 1, 1, 1, 1)
        self.testing_file_chooser_button = QtGui.QPushButton()
        self.testing_file_chooser_button.setObjectName(_fromUtf8("testing_file_chooser_button"))
        self.gridLayout.addWidget(self.testing_file_chooser_button, 1, 2, 1, 1)
        self.verticalLayout.addLayout(self.gridLayout)
        self.label_1 = QtGui.QLabel()
        self.verticalLayout.addWidget(self.label_1)
        self.label = QtGui.QLabel()
        self.label.setMaximumSize(QtCore.QSize(16777215, 41))
        self.label.setObjectName(_fromUtf8("label"))
        self.verticalLayout.addWidget(self.label)

        self.setLayout(self.verticalLayout)
        self.retranslateUi()

    def retranslateUi(self):
        self.label.setText(_translate("HistoricalDataPage", "Pressing Next will begin training", None))
        self.label_1.setText(_translate("HistoricalDataPage",
                                       '''Please ensure that these files contain at least a column with header "
                                       '''and another with header "Low" for the system to be able to process the
                                       None))
        self.label_2.setText(_translate("HistoricalDataPage", "Training Data Set", None))
        self.training_file_chooser_button.setText(_translate("HistoricalDataPage", "Choose file", None))
        self.label_3.setText(_translate("HistoricalDataPage", "Testing Data Set", None))
        self.testing_file_chooser_button.setText(_translate("HistoricalDataPage", "Choose file", None))

    def initializePage(self):
        #register fields
        self.registerField("training_data*", self.training_data_path)
        self.registerField("testing_data*", self.testing_data_path)

        self.training_file_chooser_button.clicked.connect(lambda: self.showDialog(self.training_data_path))
        self.testing_file_chooser_button.clicked.connect(lambda: self.showDialog(self.testing_data_path))

    def showDialog(self, line_edit_to_change):
        dialog = QtGui.QFileDialog(self)
        fname = dialog.getOpenFileName(self, 'Open file', QDir.homePath())
        line_edit_to_change.setText(fname)
        #set fname as sender's box

class TrainingPage(QtGui.QWizardPage):
    def __init__(self, parent=None):
        super(TrainingPage, self).__init__(parent)

        self.setTitle("Training")
        self.setSubTitle("The program is currently processing the training data. "
                         "\nThroughout the training, you will see several MATLAB progress dialogs pop up."
                         "\nThis typically takes up to 4-6 hours. Please see progress below.")

        self.verticalLayout = QtGui.QVBoxLayout()
        self.progressBar = QtGui.QProgressBar()
        self.verticalLayout.addWidget(self.progressBar)
        self.textBrowser = QtGui.QTextBrowser()
        self.verticalLayout.addWidget(self.textBrowser)

        self.setLayout(self.verticalLayout)

        self.pbar_value = 0
        self.finished = False

    def initializePage(self):
        training_path = str(self.field("training_data").toString())
        testing_path = str(self.field("testing_data").toString())

        print training_path, testing_path

        self.training_thread = TrainingThread(training_path, testing_path)
        self.training_thread.updated.connect(self.update_progress)
        self.training_thread.errorEncountered.connect(self.notify_error)
        self.training_thread.finished.connect(self.activateNext)

        self.training_thread.start()

    def isComplete(self, *args, **kwargs):
        return self.finished

    def cleanupPage(self, *args, **kwargs):
        QtGui.QWizardPage.cleanupPage(self, *args, **kwargs)
        self.training_thread.exit(-1)

    def update_progress(self, text, progress_int):
        self.pbar_value += progress_int
        self.progressBar.setValue(self.pbar_value * 100.0 / 13.0)
        self.textBrowser.insertHtml('<p style="color:black">{0}</p>'.format(text))
        self.textBrowser.verticalScrollBar().setValue(
                        self.textBrowser.verticalScrollBar().maximum())
    def notify_error(self, text):
        # TODO something special here
        self.textBrowser.insertHtml('<p style="color:red">{0}</p>'.format(text))
        self.training_thread.exit(-1)

    def activateNext(self):
        self.finished = True
        self.completeChanged.emit()

class ConclusionPage(QtGui.QWizardPage):
    def __init__(self, parent=None):
        super(ConclusionPage, self).__init__(parent)

        self.setTitle("Congratulations!")
        self.setSubTitle("The program has now finished training using the historical data given."
                         "We have also saved your account info and we'll be using this for trading."
                         "Please click Finish to activate the automated trader.")

        self.bottomLabel = QtGui.QLabel()
        self.bottomLabel.setWordWrap(True)

        agreeBox = QtGui.QCheckBox(self.tr("Start automated trader immediately?"))
        agreeBox.setChecked(True)
        autoModeBox = QtGui.QCheckBox(self.tr("Do you want to run on auto mode?"
            "\n(Unchecking this will allow the agent to run on email alert-only mode \ninstead of automatically
        autoModeBox.setChecked(True)

        vbox = QtGui.QVBoxLayout()
        vbox.addWidget(self.bottomLabel)
        vbox.addWidget(agreeBox)
        vbox.addWidget(autoModeBox)
        self.setLayout(vbox)

        self.registerField("start_checkbox", agreeBox)
        self.registerField("automode_checkbox", autoModeBox)


class TrainingThread(QtCore.QThread):

    # These signal sends a message, and 1 for progress messages
    # 14 progress ticks will be sent in total
    # 1 for preporcessing, 1 for MATLAB init, 12 for all FIS
    updated = QtCore.pyqtSignal(str, float)
    errorEncountered = QtCore.pyqtSignal(str)

    def __init__(self, training_path, testing_path, parent=None):
        super(TrainingThread, self).__init__(parent)
        self.training_file = training_path
        self.testing_file = testing_path

    # Recommended by https://joplaete.wordpress.com/2010/07/21/threading-with-pyqt4/
    def __del__(self):
        print "Thread terminated"
        self.wait()

    def run(self):
        self.updated.emit("Training commencing...<br />", 0)
        start_time = time.time()

        try:
            self.updated.emit("Pre-processing files...<br />", 0)
            preprocess.preprocess_OHLC_training(self.training_file)
            preprocess.preprocess_OHLC_testing(self.testing_file)
            self.updated.emit("Training and testing files have been pre-processed<br />", 0.5)

            self.updated.emit("MATLAB Engine is being initialized...<br />", 0)
            eng = matlab_funcs.init_matlab()
            self.updated.emit("MATLAB Engine has been initialized!<br />", 0.5)
        except Exception, e:
            self.errorEncountered.emit(str(e) +
                                       "<br />Please return to the previous page and check your input.<br />")
            return

        for i in range(config._horizon):
            out = StringIO.StringIO()
            err = StringIO.StringIO()
            self.updated.emit('<br /><br />High (t+{0}) training...<br />'.format(i+1), 0)
            eng.anfis_init_train('training_high_{0}.csv'.format(i+1), 'high_{0}.fis'.format(i+1),
                                 stdout=out, stderr=err)
            if err.len > 0:
                self.errorEncountered.emit(err.getvalue())
                self.exit(-1)
            self.updated.emit('<PRE STYLE="font-family: Helvetica">'+out.getvalue()+"</PRE>", 1)

            out = StringIO.StringIO()
            err = StringIO.StringIO()
            self.updated.emit('<br /><br />Low (t+{0}) training...<br />'.format(i+1), 0)
            eng.anfis_init_train('training_low_{0}.csv'.format(i+1), 'low_{0}.fis'.format(i+1),
                                 stdout=out, stderr=err)
            if err.len > 0:
                self.errorEncountered.emit(err.getvalue())
                self.exit(-1)
            self.updated.emit('<PRE STYLE="font-family: Helvetica">'+out.getvalue()+"</PRE>", 1)

#         for _ in xrange(12):
#             self.updated.emit("Emit dummy<br />", 1)

        self.updated.emit("<br />Finished after %s seconds" % (time.time() - start_time), 0)
```

```
if __name__ == '__main__':

    import sys

    app = QtGui.QApplication(sys.argv)
    app.setOrganizationName(config._organization)
    app.setApplicationName(config._program_name)
    wizard = SetupWizard()
    wizard.show()
    sys.exit(app.exec_())


'''
Research Backtesting Environments in Python with pandas
From https://www.quantstart.com/articles/Research-Backtesting-Environments-in-Python-with-pandas
@author: Michael Halls-Moore
'''

from abc import ABCMeta, abstractmethod

class Strategy(object):
    """Strategy is an abstract base class providing an interface for
    all subsequent (inherited) trading strategies.

    The goal of a (derived) Strategy object is to output a list of signals,
    which has the form of a time series indexed pandas DataFrame.

    In this instance only a single symbol/instrument is supported."""

    __metaclass__ = ABCMeta

    @abstractmethod
    def generate_signals(self):
        """An implementation is required to return the DataFrame of symbols
        containing the signals to go long, short or hold (1, -1 or 0)."""
        raise NotImplementedError("Should implement generate_signals()!")

class Portfolio(object):
    """An abstract base class representing a portfolio of
    positions (including both instruments and cash), determined
    on the basis of a set of signals provided by a Strategy."""

    __metaclass__ = ABCMeta

    @abstractmethod
    def generate_positions(self):
        """Provides the logic to determine how the portfolio
        positions are allocated on the basis of forecasting
        signals and available cash."""
        raise NotImplementedError("Should implement generate_positions()!")

    @abstractmethod
    def backtest_portfolio(self):
        """Provides the logic to generate the trading orders
        and subsequent equity curve (i.e. growth of total equity),
        as a sum of holdings and cash, and the bar-period returns
        associated with this curve based on the 'positions' DataFrame.

        Produces a portfolio object that can be examined by
        other classes/functions."""
        raise NotImplementedError("Should implement backtest_portfolio()!")


# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'backtest_diag.ui'
#
# Created: Thu Jun 11 16:42:39 2015
#      by: PyQt4 UI code generator 4.11.3
#
# WARNING! All changes made in this file will be lost!

from PyQt4 import QtCore, QtGui

try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    def _fromUtf8(s):
        return s

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig)

class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName(_fromUtf8("Dialog"))
        Dialog.resize(372, 150)
        self.verticalLayout = QtGui.QVBoxLayout(Dialog)
        self.verticalLayout.setObjectName(_fromUtf8("verticalLayout"))
        self.label = QtGui.QLabel(Dialog)
        self.label.setObjectName(_fromUtf8("label"))
        self.verticalLayout.addWidget(self.label)
        self.horizontalLayout = QtGui.QHBoxLayout()
        self.horizontalLayout.setObjectName(_fromUtf8("horizontalLayout"))
        self.file_text = QtGui.QLineEdit(Dialog)
        self.file_text.setObjectName(_fromUtf8("file_text"))
        self.horizontalLayout.addWidget(self.file_text)
        self.choose_button = QtGui.QPushButton(Dialog)
        self.choose_button.setObjectName(_fromUtf8("choose_button"))
        self.horizontalLayout.addWidget(self.choose_button)
        self.verticalLayout.addLayout(self.horizontalLayout)
        self.horizontalLayout_2 = QtGui.QHBoxLayout()
        self.horizontalLayout_2.setObjectName(_fromUtf8("horizontalLayout_2"))
        self.label_2 = QtGui.QLabel(Dialog)
        self.label_2.setObjectName(_fromUtf8("label_2"))
        self.horizontalLayout_2.addWidget(self.label_2)
        self.amount_text = QtGui.QLineEdit(Dialog)
        self.amount_text.setObjectName(_fromUtf8("amount_text"))
        self.horizontalLayout_2.addWidget(self.amount_text)
        self.verticalLayout.addLayout(self.horizontalLayout_2)
        self.buttonBox = QtGui.QDialogButtonBox(Dialog)
        self.buttonBox.setOrientation(QtCore.Qt.Horizontal)
        self.buttonBox.setStandardButtons(QtGui.QDialogButtonBox.Cancel|QtGui.QDialogButtonBox.Ok)
        self.buttonBox.setObjectName(_fromUtf8("buttonBox"))
        self.verticalLayout.addWidget(self.buttonBox)

        #Validators
        double_valid = QtGui.QDoubleValidator()
        self.amount_text.setValidator(double_valid)

        self.retranslateUi(Dialog)
        QtCore.QObject.connect(self.buttonBox, QtCore.SIGNAL(_fromUtf8("accepted()")), Dialog.accept)
        QtCore.QObject.connect(self.buttonBox, QtCore.SIGNAL(_fromUtf8("rejected()")), Dialog.reject)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):
        Dialog.setWindowTitle(_translate("Dialog", "Dialog", None))
        self.label.setText(_translate("Dialog", "<html><head/><body><p><span style=\"font-weight:600;\">Backte
        self.choose_button.setText(_translate("Dialog", "Choose file", None))
        self.label_2.setText(_translate("Dialog", "Initial Amount:", None))

'''
Created on May 23, 2015

@author: ArielKenneth
'''
_horizon = 6
_granularity = 'M30'
_matlab_dir = "matlab"
_training_prefix = 'training'
_testing_prefix = 'testing'
_program_name = 'Autonomous ANFIS and GA-Based Forex Agent'
_program_name_short = 'AAGFA'
_organization = 'University of the Philippines Manila'
_version = "1.0.0"
_order_units = 1000
_gui_refresh_time = 6
_historical_ticks = 41
# _ema_window = 30
_ema_window = 15
_pip_adjustment = 0.004
_trade_decay = 2
_trend_line_count = 5
_from_email = 'asampol1@up.edu.ph'
_email_pass = 'MatrixTrace'
_smtp_server = 'smtp.gmail.com'
_smtp_port = 465
_pseudobroker_server= "http://127.0.0.1/broker/"


# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'account_details_first_dialog.ui'
#
# Created: Mon Jun 01 16:48:29 2015
#      by: PyQt4 UI code generator 4.11.3
#
# WARNING! All changes made in this file will be lost!

from PyQt4 import QtCore, QtGui

try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    def _fromUtf8(s):
        return s

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig)

class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName(_fromUtf8("Dialog"))
        Dialog.resize(416, 250)
        self.verticalLayout = QtGui.QVBoxLayout(Dialog)
        self.verticalLayout.setObjectName(_fromUtf8("verticalLayout"))
        self.label = QtGui.QLabel(Dialog)
        self.label.setMaximumSize(QtCore.QSize(16777215, 41))
        self.label.setObjectName(_fromUtf8("label"))
        self.verticalLayout.addWidget(self.label)
        self.gridLayout = QtGui.QGridLayout()
        self.gridLayout.setObjectName(_fromUtf8("gridLayout"))
        self.api_token_text = QtGui.QLineEdit(Dialog)
        self.api_token_text.setObjectName(_fromUtf8("api_token_text"))
        self.gridLayout.addWidget(self.api_token_text, 1, 1, 1, 1)
        self.label_4 = QtGui.QLabel(Dialog)
        self.label_4.setMinimumSize(QtCore.QSize(111, 0))
        self.label_4.setObjectName(_fromUtf8("label_4"))
        self.gridLayout.addWidget(self.label_4, 3, 0, 1, 1)
        self.label_2 = QtGui.QLabel(Dialog)
        self.label_2.setMaximumSize(QtCore.QSize(101, 16777215))
        self.label_2.setObjectName(_fromUtf8("label_2"))
        self.gridLayout.addWidget(self.label_2, 0, 0, 1, 1)
        self.account_id_text = QtGui.QLineEdit(Dialog)
        self.account_id_text.setObjectName(_fromUtf8("account_id_text"))
        self.gridLayout.addWidget(self.account_id_text, 0, 1, 1, 1)
        self.label_3 = QtGui.QLabel(Dialog)
        self.label_3.setObjectName(_fromUtf8("label_3"))
        self.gridLayout.addWidget(self.label_3, 1, 0, 1, 1)
        self.instrumen_combobox = QtGui.QComboBox(Dialog)
        self.instrumen_combobox.setObjectName(_fromUtf8("instrumen_combobox"))
        self.instrumen_combobox.addItem(_fromUtf8(""))
        self.gridLayout.addWidget(self.instrumen_combobox, 3, 1, 1, 1)
        self.label_5 = QtGui.QLabel(Dialog)
        self.label_5.setObjectName(_fromUtf8("label_5"))
        self.gridLayout.addWidget(self.label_5, 2, 0, 1, 1)
        self.environment_comboBox = QtGui.QComboBox(Dialog)
        self.environment_comboBox.setObjectName(_fromUtf8("environment_comboBox"))
        self.environment_comboBox.addItem(_fromUtf8(""))
        self.environment_comboBox.addItem(_fromUtf8(""))
        self.gridLayout.addWidget(self.environment_comboBox, 2, 1, 1, 1)
        self.get_list_button = QtGui.QPushButton(Dialog)
        self.get_list_button.setObjectName(_fromUtf8("get_list_button"))
        self.gridLayout.addWidget(self.get_list_button, 4, 1, 1, 1)
        self.verticalLayout.addLayout(self.gridLayout)
        self.buttonBox = QtGui.QDialogButtonBox(Dialog)
        self.buttonBox.setOrientation(QtCore.Qt.Horizontal)
        self.buttonBox.setStandardButtons(QtGui.QDialogButtonBox.Cancel|QtGui.QDialogButtonBox.Ok)
        self.buttonBox.setObjectName(_fromUtf8("buttonBox"))
        self.verticalLayout.addWidget(self.buttonBox)
        self.label_4.setBuddy(self.instrumen_combobox)
        self.label_2.setBuddy(self.account_id_text)
        self.label_3.setBuddy(self.api_token_text)
        self.label_5.setBuddy(self.environment_comboBox)
        self.email_label = QtGui.QLabel("Your Email address:")
        self.email_text = QtGui.QLineEdit()
        self.email_label.setBuddy(self.email_text)
        self.gridLayout.addWidget(self.email_label, 5, 0, 1, 1)
        self.gridLayout.addWidget(self.email_text, 5, 1, 1, 1)

        #Validators
        int_valid = QtGui.QIntValidator()
        regexp = QtCore.QRegExp('[^@]+@[^@]+\.[^@]+')
        email_valid = QtGui.QRegExpValidator(regexp)
        self.account_id_text.setValidator(int_valid)
        self.email_text.setValidator(email_valid)

        self.retranslateUi(Dialog)
        QtCore.QObject.connect(self.buttonBox, QtCore.SIGNAL(_fromUtf8("accepted()")), Dialog.accept)
        QtCore.QObject.connect(self.buttonBox, QtCore.SIGNAL(_fromUtf8("rejected()")), Dialog.reject)
        QtCore.QMetaObject.connectSlotsByName(Dialog)
        Dialog.setTabOrder(self.account_id_text, self.api_token_text)
        Dialog.setTabOrder(self.api_token_text, self.environment_comboBox)
        Dialog.setTabOrder(self.environment_comboBox, self.instrumen_combobox)
        Dialog.setTabOrder(self.instrumen_combobox, self.get_list_button)
        Dialog.setTabOrder(self.get_list_button, self.buttonBox)

    def retranslateUi(self, Dialog):
        Dialog.setWindowTitle(_translate("Dialog", "Dialog", None))
        self.label.setText(_translate("Dialog", "<html><head/><body><p><span style=\" font-weight:600;\">Edit A
        self.label_4.setText(_translate("Dialog", "Instrument to Trade:", None))
        self.label_2.setText(_translate("Dialog", "OANDA Account ID:", None))
        self.label_3.setText(_translate("Dialog", "OANDA API Token:", None))
        self.instrumen_combobox.setItemText(0, _translate("Dialog", "Click \"Get Instrument List\" after provid
        self.label_5.setText(_translate("Dialog", "Trading Environment:", None))
        self.environment_comboBox.setItemText(0, _translate("Dialog", "Practice", None))
        self.environment_comboBox.setItemText(1, _translate("Dialog", "Live", None))
        self.get_list_button.setText(_translate("Dialog", "Get Instrument List", None))


# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'edit_diag.ui'
#
# Created: Tue Jun 09 18:22:21 2015
#      by: PyQt4 UI code generator 4.11.3
#
# WARNING! All changes made in this file will be lost!

'''
Created on Jun 6, 2015
Convenience methods for sending an email from within the program
As per http://www.nixtutor.com/linux/send-mail-through-gmail-with-python/
@author: ArielKenneth
'''

#! /usr/bin/python
```

```
import smtplib

from aagfa import config

def send_email_alert(recepient, message):
    fromaddr = config._from_email
    toaddrs  = recepient
    msg = "\r\n".join([
      "From: {0}".format(config._from_email),
      "To: {0}".format(recepient),
      "Subject: {0} - Trade Alert".format(config._program_name_short),
      "",
      message
      ])

    # Credentials (if needed)
    username = config._from_email
    password = config._email_pass

    # The actual mail send
    server = smtplib.SMTP('smtp.gmail.com:587')
    server.ehlo()
    server.starttls()
    server.login(username,password)
    server.sendmail(fromaddr, toaddrs, msg)
    server.quit()

def main():
    send_email_alert('ayieampol@gmail.com', "Testing!")

if __name__ == '__main__':
    main()

'''
Created on May 29, 2015

@author: ArielKenneth
'''
import sys
import StringIO
import time
from dateutil import tz
from datetime import datetime

from PyQt4 import QtCore, QtGui
import oandapy

import config
from aagfa import pseudo_broker

LONG = 1
SHORT = -1
HOLD = 0

class GuiUpdateThread(QtCore.QThread):

    # use this signal to inform of change in positions
    positions_changed = QtCore.pyqtSignal()
    # use this signal to inform of changes in account status
    account_info_changed = QtCore.pyqtSignal()
    # use this signal to inform of changes in which account to display
    account_display_changed = QtCore.pyqtSignal()
    # use this signal for updates to status bar
    status_bar_update = QtCore.pyqtSignal(str)

    #Arranged to correspond with columns of actual GUI table
    history_keys = ['id', 'type', 'units', 'instrument', 'side', 'time', 'price', 'pl', 'accountBalance']
    open_keys = ['id', 'units', 'side', 'instrument', 'time', 'price', 'takeProfit', 'stopLoss']
    account_stats_key = ["accountId", "accountName", "balance", "unrealizedPl", "realizedPl",
                         "marginUsed", "marginAvail", "openTrades", "openOrders", "marginRate", "accountCurrenc
    account_stats_labels = ["Account ID", "Account Name", "Balance", "Unrealized P&L ", "Realized P&L",
                            "Margin Used", "Margin Available", "Open Trades", "Open Orders", "Margin Rate", "Ba
    pseudo_open_keys = ["Order_ID","Units","Side","Instrument","Order_Time","Price","Take_Profit","Stop_Loss"]

    def __init__(
        self, open_positions_table, trans_history_table, account_stats_widget, parent=None):

        super(GuiUpdateThread, self).__init__(parent)
        self.open_positions_table = open_positions_table
        self.trans_history_table = trans_history_table
        self.account_stats_widget = account_stats_widget

        self.establish_signals_and_slots()
        self.fetch_settings()

        self.from_zone = tz.tzutc()
        self.to_zone = tz.tzlocal()

        self.display_oanda = True

    def establish_signals_and_slots(self):
        self.account_info_changed.connect(self.fetch_settings)
        self.account_display_changed.connect(self.change_display_account)

    def fetch_settings(self):
        settings = QtCore.QSettings()
        self.api_token = str(settings.value("app/api_token").toString())
        self.account_id = str(settings.value("app/account_id").toString())
        self.instrument = str(settings.value("app/instrument").toString())
        self.environment = str(settings.value("app/environment").toString())


    def change_display_account(self):
        self.display_oanda = not self.display_oanda
        self.trans_history_table.setRowCount(0)

    # Recommended by https://joplaete.wordpress.com/2010/07/21/threading-with-pyqt4/
    def __del__(self):
        print 'Destructor called'
        self.wait()

    def run(self):
        '''
        Main loop time control
        '''
        self.oanda = oandapy.API(self.environment, self.api_token)
        while True:
            self.main_loop()
            time.sleep(config._gui_refresh_time)

    def main_loop(self):
        '''
        Main trading loop time control
        '''
        try:
            self.update_account_info()
            self.update_positions()
        except Exception as e:
            self.status_bar_update.emit("Cannot update data. Check your connection.")
            print str(e)

    def update_account_info(self):
        if self.display_oanda:
            account_info = self.oanda.get_account(account_id=self.account_id)
            self.account_stats_widget.clear()
            for i in range(len(self.account_stats_key)):
                self.account_stats_widget.addItem(
                    "{0}: {1}".format(self.account_stats_labels[i], account_info.get(self.account_stats_key[i])
        else:
            account_info = pseudo_broker.get_status(acc_id=self.account_id)
            self.account_stats_widget.clear()
            self.account_stats_widget.addItem("Account ID: {0}".format(account_info['Account_ID']))
            self.account_stats_widget.addItem("Number of Orders: {0}".format(account_info['num_orders']))

    def update_positions(self):
        if self.display_oanda:
            self.oanda_update_positions()
        else:
            self.pseudo_update_positions()

    def oanda_update_positions(self):
        ### History first
        transactions = self.oanda.get_transaction_history(self.account_id)
        transactions = transactions.get('transactions')
        self.trans_history_table.setRowCount(len(transactions))
        row = 0
        for transaction in transactions:
            col = 0
            for key in self.history_keys:
                if key == 'time':
                    utc_string = str(transaction.get(key))
                    utc = datetime.strptime(utc_string, '%Y-%m-%dT%H:%M:%S.%fZ')
                    utc = utc.replace(tzinfo=self.from_zone)
                    local = datetime.strftime(utc.astimezone(self.to_zone), '%Y-%m-%d %H:%M:%S')
                    newitem = QtGui.QTableWidgetItem(str(local))
                else:
                    newitem = QtGui.QTableWidgetItem(str(transaction.get(key)))
                self.trans_history_table.setItem(row, col, newitem)
                col += 1
            row += 1

        ### Open positions here
        open_pos = self.oanda.get_trades(self.account_id)
        open_pos = open_pos.get('trades')
        self.open_positions_table.setRowCount(len(open_pos))
        row = 0
        for position in open_pos:
            col = 0
            for key in self.open_keys:
                if key == 'time':
                    utc_string = str(position.get(key))
                    utc = datetime.strptime(utc_string, '%Y-%m-%dT%H:%M:%S.%fZ')
                    utc = utc.replace(tzinfo=self.from_zone)
                    local = datetime.strftime(utc.astimezone(self.to_zone), '%Y-%m-%d %H:%M:%S')
                    newitem = QtGui.QTableWidgetItem(str(local))
                else:
                    newitem = QtGui.QTableWidgetItem(str(position.get(key)))
                self.open_positions_table.setItem(row, col, newitem)
                col += 1
            row += 1

    def pseudo_update_positions(self):
        ### Open positions here
        open_pos = pseudo_broker.get_orders(acc_id=self.account_id)
        self.open_positions_table.setRowCount(len(open_pos))
        row = 0
        for position in open_pos:
            col = 0
            for key in self.pseudo_open_keys:
                newitem = QtGui.QTableWidgetItem(str(position.get(key)))
                self.open_positions_table.setItem(row, col, newitem)
                col += 1
            row += 1

def main():
    app = QtGui.QApplication(sys.argv)
    app.setOrganizationName(config._organization)
    app.setApplicationName(config._program_name)
    sample = GuiUpdateThread()
    sample.start()

    sys.exit(app.exec_())

if __name__ == '__main__':
    main()


'''
Template for calling a MATLAB .m function from Python via the
MATLAB Engine for Python
Current contents: Idea for how the training may go after preprocessing

!!! MUST be in the same package(/directory) as intended matlab folder
!!! created by preprocess.py prior to calling this module's functions

Created on May 23, 2015

@author: ArielKenneth
'''
import time
import os
import matlab.engine
from aagfa import config

def init_matlab():
    '''
    Get an engine instance and set working dir to the proper matlab file folder
    '''

    #Start engine!
    eng = matlab.engine.start_matlab()
    #set cd to matlab folder under .py file's current dir
    eng.cd(os.path.join(os.getcwd(), config._matlab_dir))
    return eng

def train_anfis(eng):
    ###IDEA FOR GUI!!!
    #In the future, receive a QTextBrowser argument
    #Perform code stuff and update it as needed
    for i in range(config._horizon):
        print '\n\nHigh (t+{0}) training...'.format(i+1)
        start_time = time.time()
        eng.anfis_init_train('training_high_{0}.csv'.format(i+1), 'high_{0}.fis'.format(i+1))
        print("--- %s seconds ---" % (time.time() - start_time))

        print '\n\nLow (t+{0}) training...'.format(i+1)
        start_time = time.time()
        #Perhaps update QTextBrowser here with MATLAB response
        eng.anfis_init_train('training_low_{0}.csv'.format(i+1), 'low_{0}.fis'.format(i+1))
        #Perhaps update QTextBrowser here with MATLAB response
        print("TOTAL: --- %s seconds ---" % (time.time() - start_time))

def anfis_predict(eng, input_data, fis_filename, **kwargs):
    '''
    eng - MATLAB engine object
    input - basic list (not NumPy!) of N previous data points to use for prediction
    fis_filename - filename of appropriate saved FIS file, with or without .fis
    '''
    return eng.fis_predict(matlab.double(input_data), fis_filename, **kwargs)

def main():
    print 'Initializing engine...'
    eng = init_matlab()
    print 'Init done!'
    train_anfis(eng)
    #Test run!
    input_data = [1.3347,1.3361,1.3359,1.3327,1.3324,1.3306,1.3299,1.3284,1.3292,1.3293,1.3293,1.3296,1.3302,1.
    for i in range(config._horizon):
        print 'Predicting {0}'.format(i+1)
        print anfis_predict(eng, input_data, 'high_{0}.fis'.format(i+1))
        eng.quit()

if __name__ == '__main__':
    main()


# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'modify_dialog.ui'
#
# Created: Sun May 31 02:14:27 2015
#      by: PyQt4 UI code generator 4.11.3
#
# WARNING! All changes made in this file will be lost!

from PyQt4 import QtCore, QtGui

try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    def _fromUtf8(s):
        return s

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig)

class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName(_fromUtf8(”Dialog”))
        Dialog.resize(250, 300)
        self.gridLayout = QtGui.QGridLayout(Dialog)
        self.gridLayout.setObjectName(_fromUtf8(”gridLayout”))
        self.title = QtGui.QLabel(Dialog)
        self.title.setMinimumSize(QtCore.QSize(46, 13))
        self.title.setObjectName(_fromUtf8(”title”))
        self.gridLayout.addWidget(self.title, 0, 0, 1, 2)
        self.main_info = QtGui.QLabel(Dialog)
        self.main_info.setObjectName(_fromUtf8(”main_info”))
        self.gridLayout.addWidget(self.main_info, 1, 0, 1, 2)
        self.last_quote = QtGui.QLabel(Dialog)
        self.last_quote.setObjectName(_fromUtf8(”last_quote”))
        self.gridLayout.addWidget(self.last_quote, 2, 0, 1, 2)
        self.horizontalLayout = QtGui.QHBoxLayout()
        self.horizontalLayout.setObjectName(_fromUtf8(”horizontalLayout”))
        self.modify_button = QtGui.QRadioButton(Dialog)
        self.modify_button.setObjectName(_fromUtf8(”modify_button”))
        self.modify_button.setChecked(True)
        self.horizontalLayout.addWidget(self.modify_button)
        self.close_pos_button = QtGui.QRadioButton(Dialog)
        self.close_pos_button.setObjectName(_fromUtf8(”close_pos_button”))
        self.horizontalLayout.addWidget(self.close_pos_button)
        self.gridLayout.addLayout(self.horizontalLayout, 3, 0, 1, 2)
        self.order_id_label = QtGui.QLabel(Dialog)
        self.order_id_label.setObjectName(_fromUtf8(”order_id_label”))
        self.gridLayout.addWidget(self.order_id_label, 4, 0, 1, 1)
        self.order_id_value = QtGui.QLabel(Dialog)
        self.order_id_value.setObjectName(_fromUtf8(”order_id_value”))
        self.gridLayout.addWidget(self.order_id_value, 4, 1, 1, 1)
        self.time_label = QtGui.QLabel(Dialog)
        self.time_label.setObjectName(_fromUtf8(”time_label”))
        self.gridLayout.addWidget(self.time_label, 5, 0, 1, 1)
        self.time_value = QtGui.QLabel(Dialog)
        self.time_value.setObjectName(_fromUtf8(”time_value”))
        self.gridLayout.addWidget(self.time_value, 5, 1, 1, 1)
        self.tp_label = QtGui.QLabel(Dialog)
        self.tp_label.setObjectName(_fromUtf8(”tp_label”))
        self.gridLayout.addWidget(self.tp_label, 6, 0, 1, 1)
        self.tp_lineedit = QtGui.QLineEdit(Dialog)
        self.tp_lineedit.setObjectName(_fromUtf8(”tp_lineedit”))
        self.tp_lineedit.setMaxLength(7)
        self.gridLayout.addWidget(self.tp_lineedit, 6, 1, 1, 1)
        self.sl_label = QtGui.QLabel(Dialog)
        self.sl_label.setObjectName(_fromUtf8(”sl_label”))
        self.gridLayout.addWidget(self.sl_label, 7, 0, 1, 1)
        self.sl_lineedit = QtGui.QLineEdit(Dialog)
        self.sl_lineedit.setObjectName(_fromUtf8(”sl_lineedit”))
        self.sl_lineedit.setMaxLength(7)
        self.gridLayout.addWidget(self.sl_lineedit, 7, 1, 1, 1)
        self.buttonBox = QtGui.QDialogButtonBox(Dialog)
        self.buttonBox.setOrientation(QtCore.Qt.Horizontal)
        self.buttonBox.setStandardButtons(QtGui.QDialogButtonBox.Cancel|QtGui.QDialogButtonBox.Ok)
        self.buttonBox.setObjectName(_fromUtf8(”buttonBox”))
        self.gridLayout.addWidget(self.buttonBox, 8, 0, 1, 2)

        #Validators
        double_valid = QtGui.QDoubleValidator()
        self.tp_lineedit.setValidator(double_valid)
        self.sl_lineedit.setValidator(double_valid)

        self.retranslateUi(Dialog)
        QtCore.QObject.connect(self.buttonBox, QtCore.SIGNAL(_fromUtf8(”accepted()”)), Dialog.accept)
        QtCore.QObject.connect(self.buttonBox, QtCore.SIGNAL(_fromUtf8(”rejected()”)), Dialog.reject)
        QtCore.QObject.connect(self.close_pos_button, QtCore.SIGNAL(_fromUtf8(”clicked()”)), self.sl_label.hide
        QtCore.QObject.connect(self.close_pos_button, QtCore.SIGNAL(_fromUtf8(”clicked()”)), self.tp_label.hide
        QtCore.QObject.connect(self.close_pos_button, QtCore.SIGNAL(_fromUtf8(”clicked()”)), self.sl_lineedit.h
        QtCore.QObject.connect(self.close_pos_button, QtCore.SIGNAL(_fromUtf8(”clicked()”)), self.tp_lineedit.h
        QtCore.QObject.connect(self.modify_button, QtCore.SIGNAL(_fromUtf8(”clicked()”)), self.tp_lineedit.show
        QtCore.QObject.connect(self.modify_button, QtCore.SIGNAL(_fromUtf8(”clicked()”)), self.sl_label.show)
        QtCore.QObject.connect(self.modify_button, QtCore.SIGNAL(_fromUtf8(”clicked()”)), self.sl_lineedit.show
        QtCore.QObject.connect(self.modify_button, QtCore.SIGNAL(_fromUtf8(”clicked()”)), self.tp_label.show)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):
        Dialog.setWindowTitle(_translate(”Dialog”, ”Dialog”, None))
        self.title.setText(_translate(”Dialog”, ”<html><head/><body><p align=\”center\”><span style=\” font−size
        self.main_info.setText(_translate(”Dialog”, ”<html><head/><body><p align=\”center\”><span style=\” font-
        self.last_quote.setText(_translate(”Dialog”, ”<html><head/><body><p align=\”center\”><span style=\” font
        self.modify_button.setText(_translate(”Dialog”, ”Modify”, None))
        self.close_pos_button.setText(_translate(”Dialog”, ”Close”, None))
        self.order_id_label.setText(_translate(”Dialog”, ”Order ID:”, None))
        self.order_id_value.setText(_translate(”Dialog”, ”123456789”, None))
        self.time_label.setText(_translate(”Dialog”, ”Date and Time:”, None))
        self.time_value.setText(_translate("Dialog", "Something something", None))
        self.tp_label.setText(_translate("Dialog", "Take Profit:", None))
        self.sl_label.setText(_translate("Dialog", "Stop Loss:", None))


'''
Created on May 26, 2015

@author: ArielKenneth
'''

import oandapy

def create_instance(api_token, env="practice"):
    return oandapy.API(environment=env, access_token=api_token)

def just_the_instruments(oanda, account_id):
    '''
    Return a list of the available instruments for account_id using oanda instance
    '''
    instruments = oanda.get_instruments(account_id)
    instruments_dict = instruments.get("instruments")
    return [d["instrument"] for d in instruments_dict if "instrument" in d]

def create_order(oanda, units, instrument, side, take_profit, stop_loss, order_type="market"):
    '''
    Wrapper for OANDA's create order command which checks for trading time first
    '''
    if within_hours():
        return oanda.create_order(instrument=instrument, units=units, side=side, type=order_type,
                                  takeProfit=take_profit, stopLoss = stop_loss)
    else:
        raise oandapy.OandaError('Outside trading hours.')

def modify_trade(oanda, account_id, trade_id, take_profit=None, stop_loss=None):
    '''
    Wrapper for OANDA's modify trade command
    '''
    if take_profit==None and stop_loss <> None:
        return oanda.modify_trade(account_id=account_id, trade_id=trade_id, stopLoss=stop_loss)
    if stop_loss==None and take_profit <> None:
        return oanda.modify_trade(account_id=account_id, trade_id=trade_id, takeProfit=take_profit)
    ## if both are specified
    return oanda.modify_trade(account_id=account_id, trade_id=trade_id, stopLoss=stop_loss, takeProfit=take_pro

def latest_price(oanda, instrument, quote_type):
    '''
    Wrapper for OANDA's get prices command,
    but built to accept one instrument and return one price
    quote_type - "bid" or "ask"
    '''
    try:
        response = oanda.get_prices(instruments=instrument)
        response = response['prices']
        inst = response[0]
        return inst[quote_type]
    except Exception as e:
        print str(e)
        return "Cannot be retrieved"

def within_hours():
    import datetime
    import pytz
    d = datetime.datetime.now(pytz.timezone('US/Pacific'))

    #Check if Monday to Thursday
    if d.isoweekday() in range(1, 5):
        return True
    # If it's Friday and earlier than 5
    if d.isoweekday() == 5 and d.hour < 17:
        return True
    # If it's Sunday and past 5
    if d.isoweekday() == 7 and d.hour >= 17:
        return True

    return False

def main():
#     oanda = create_instance("47ca8a1e913f1d55eb10b7d76b73612e-bc9b8ce9ca204ad1917e1c75f816f416")
#     print just_the_instruments(oanda, "4310921")

    print within_hours()
    oanda = oandapy.API(access_token='47ca8a1e913f1d55eb10b7d76b73612e-bc9b8ce9ca204ad1917e1c75f816f416')
    print latest_price(oanda, 'EUR_USD', 'bid')
    print latest_price(oanda, 'EUR_USD', 'ask')

if __name__ == '__main__':
    main()


'''
Created on Nov 26, 2014

@author: ariel
'''
import os
import numpy as np
import talib
from aagfa import config
```

```
def strip(smooth_ts, raw_ts, inputs, offset=0):
    '''
    Function to strip cut number of of rows in ts
    Ex. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 -> with inputs 4 and offset 0
    1, 2, 3, 4, 5
    2, 3, 4, 5, 6
    3, 4, 5, 6, 7
    4, 5, 6, 7, 8
    5, 6, 7, 8, 9
    6, 7, 8, 9, 10
    Additionally, with offset 1 for example
    1, 2, 3, 4, 6
    2, 3, 4, 5, 7
    3, 4, 5, 6, 8
    4, 5, 6, 7, 9
    5, 6, 7, 8, 10
    Uses smoothed TS for input values, raw TS for output values
    '''
    assert smooth_ts.size == raw_ts.size

    B = np.array([ np.append(smooth_ts[k-inputs:k], raw_ts[k+offset]) for k in xrange(inputs,smooth_ts.size-off
    return B

def strip_plain(ts, inputs, offset=0):
    '''
    Function to strip cut number of of rows in ts
    Ex. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 -> with inputs 4 and offset 0
    1, 2, 3, 4, 5
    2, 3, 4, 5, 6
    3, 4, 5, 6, 7
    4, 5, 6, 7, 8
    5, 6, 7, 8, 9
    6, 7, 8, 9, 10
    Additionally, with offset 1 for example
    1, 2, 3, 4, 6
    2, 3, 4, 5, 7
    3, 4, 5, 6, 8
    4, 5, 6, 7, 9
    5, 6, 7, 8, 10
    '''
    B = np.array([ np.append(ts[k-inputs:k], ts[k+offset]) for k in xrange(inputs,ts.size-offset)])
    return B

def preprocess_OHLC_training(filename, num_inputs=41, output_file_prefix=config._training_prefix, ema_lookback=
    '''
    Takes a whole OHLC dataset with High and Low headers and applies EMA smoothing
    Saves to pwd's directory specified by config._matlab_dir the pre-processed data with these filenames:
    <output_file_prefix>_high_n.csv or <output_file_prefix>_low_n.csv
    such that the file represents the dataset with output (t + n)
    '''
    #Read data, Extract High/Low, Smoothen series
    data = np.genfromtxt(filename, delimiter=',', names=True)
    smoothed_high = talib.EMA(data['High'], ema_lookback)
    smoothed_low = talib.EMA(data['Low'], ema_lookback)

    smoothed_high = smoothed_high[~np.isnan(smoothed_high)]
    smoothed_low = smoothed_low[~np.isnan(smoothed_low)]
    plain_high = data['High'][ema_lookback-1:]
    plain_low = data['Low'][ema_lookback-1:]

    path = os.path.join(os.getcwd(), config._matlab_dir)
    if not os.path.exists(path):
        os.makedirs(path)

    print path

    for i in range(config._horizon):
        processed_high = strip(smoothed_high, plain_high, num_inputs, i)
        out_file = "{0}_high_{1}.csv".format(output_file_prefix, i+1)
        np.savetxt(os.path.join(path, out_file), processed_high, delimiter=",",fmt='%.5f')
        processed_low = strip(smoothed_low, plain_low, num_inputs, i)
        out_file = "{0}_low_{1}.csv".format(output_file_prefix, i+1)
        np.savetxt(os.path.join(path, out_file), processed_low, delimiter=",",fmt='%.5f')
        print 'Loop %d finished' % i

def preprocess_OHLC_testing(filename, num_inputs=41, output_file_prefix=config._testing_prefix, ema_lookback=No
    '''
    Takes a whole OHLC dataset with High and Low headers
    Arranges in such a way that only num_inputs columns are made (simulating no output)
    Saves to <output_file_prefix>_high.csv and <output_file_prefix>_low.csv
    '''
    #Read data, Extract High/Low, Smoothen series
    data = np.genfromtxt(filename, delimiter=',', names=True)

    path = os.path.join(os.getcwd(), config._matlab_dir)
    if not os.path.exists(path):
        os.makedirs(path)

    processed_high = strip_plain(data['High'], num_inputs-1, 0)
    out_file = "{0}_high.csv".format(output_file_prefix)
    np.savetxt(os.path.join(path, out_file), processed_high, delimiter=",",fmt='%.4f')
    processed_low = strip_plain(data['Low'], num_inputs-1, 0)
    out_file = "{1}_low.csv".format(config._matlab_dir, output_file_prefix)
    np.savetxt(os.path.join(path, out_file), processed_low, delimiter=",",fmt='%.4f')


def main():
    filename = 'C:\Users\ArielKenneth\Desktop\data\EURUSD30_recent_train.csv'
    preprocess_OHLC_training(filename)
    filename = 'C:\Users\ArielKenneth\Desktop\data\EURUSD30_recent_test.csv'
    preprocess_OHLC_testing(filename)

if __name__ == '__main__':
    main()

'''
Wrapper for Pseudo Broker web functions
arguments provided as dict to generalize support for any arguments the broker dictates
'''

import json
import requests
from aagfa import config

def place_order(**kwargs):
    '''
    Places orders with configured broker as per required arguments
    Returns the execution price on success, and None otherwise
    '''
    request_url = config._pseudobroker_server + 'place_order.php'
    post_response = requests.post(url=request_url, data=kwargs)
    if post_response.status_code <> 200:
        return None
    asdict = json.loads(post_response.text)
    if not asdict['success']:
        return None
    return asdict['price']

def get_current_price(**kwargs):
    '''
    Gets latest price as provided by pseudo broker
    Returns the price itself
    '''
    request_url = config._pseudobroker_server + 'get_rate.php'
    post_response = requests.post(url=request_url, data=kwargs)
    if post_response.status_code <> 200:
        return None
    return float(post_response.text)

def get_orders(**kwargs):
    '''
    Gets placed orders with configured broker as per required arguments
    Returns a list of dicts corresponding to orders on success, and None otherwise
    '''
    request_url = config._pseudobroker_server + 'get_orders.php'
    post_response = requests.post(url=request_url, data=kwargs)
    if post_response.status_code <> 200:
        return None
    return json.loads(post_response.text)

def get_status(**kwargs):
    '''
    Gets placed orders with configured broker as per required arguments
    Returns a list of dicts corresponding to orders on success, and None otherwise
    '''
    request_url = config._pseudobroker_server + 'get_status.php'
    post_response = requests.post(url=request_url, data=kwargs)
    if post_response.status_code <> 200:
        return None
    return json.loads(post_response.text)

def close_order(**kwargs):
    '''
    "Closes" an order - which in the case of the pseudo broker, deletes it
    '''
    request_url = config._pseudobroker_server + 'close_order.php'
    post_response = requests.post(url=request_url, data=kwargs)
    if post_response.status_code <> 200:
        return None
    return int(post_response.text)

def modify_order(**kwargs):
    '''
    Modifies an order
    '''
    request_url = config._pseudobroker_server + 'modify_order.php'
    post_response = requests.post(url=request_url, data=kwargs)
    if post_response.status_code <> 200:
        return None
    try:
        return int(post_response.text)
    except ValueError:
        return post_response.text

class BrokerError(Exception):
    """ Generic error class, catches broker errors
    """

    def __init__(self, error_response):
        msg = "Pseudo Broker request failed"

        super(BrokerError, self).__init__(msg)


if __name__ == '__main__':
    print place_order(acc_id=8596674, side='buy', inst='EUR_USD', units=1000, tp=1.54321, sl=1.12345)
    print get_orders(acc_id=8596674)
    print get_status(acc_id=8596674)
    print get_current_price(inst="EUR_USD")
#      print close_order(order_id=7)
#      print modify_order(order_id=10, tp=1.1234, sl=1.678)


# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'account_details_second_dialog.ui'
#
# Created: Mon Jun 01 17:51:58 2015
#      by: PyQt4 UI code generator 4.11.3
#
# WARNING! All changes made in this file will be lost!

from PyQt4 import QtCore, QtGui

try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    def _fromUtf8(s):
        return s

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig)

class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName(_fromUtf8("Dialog"))
        Dialog.resize(400, 227)
        Dialog.setMinimumSize(QtCore.QSize(0, 227))
        self.verticalLayout = QtGui.QVBoxLayout(Dialog)
        self.verticalLayout.setObjectName(_fromUtf8("verticalLayout"))
        self.label = QtGui.QLabel(Dialog)
        self.label.setObjectName(_fromUtf8("label"))
        self.verticalLayout.addWidget(self.label)
        self.gridLayout = QtGui.QGridLayout()
        self.gridLayout.setObjectName(_fromUtf8("gridLayout"))
        self.label_2 = QtGui.QLabel(Dialog)
        self.label_2.setMaximumSize(QtCore.QSize(101, 16777215))
        self.label_2.setObjectName(_fromUtf8("label_2"))
        self.gridLayout.addWidget(self.label_2, 0, 0, 1, 1)
        self.label_3 = QtGui.QLabel(Dialog)
        self.label_3.setObjectName(_fromUtf8("label_3"))
        self.gridLayout.addWidget(self.label_3, 1, 0, 1, 1)
        self.testing_file_chooser_button = QtGui.QPushButton(Dialog)
        self.testing_file_chooser_button.setObjectName(_fromUtf8("testing_file_chooser_button"))
        self.gridLayout.addWidget(self.testing_file_chooser_button, 1, 2, 1, 1)
        self.training_file_chooser_button = QtGui.QPushButton(Dialog)
        self.training_file_chooser_button.setObjectName(_fromUtf8("training_file_chooser_button"))
        self.gridLayout.addWidget(self.training_file_chooser_button, 0, 2, 1, 1)
        self.testing_data_path = QtGui.QLineEdit(Dialog)
        self.testing_data_path.setObjectName(_fromUtf8("testing_data_path"))
        self.gridLayout.addWidget(self.testing_data_path, 1, 1, 1, 1)
        self.training_data_path = QtGui.QLineEdit(Dialog)
        self.training_data_path.setObjectName(_fromUtf8("training_data_path"))
        self.gridLayout.addWidget(self.training_data_path, 0, 1, 1, 1)
        self.verticalLayout.addLayout(self.gridLayout)
        self.begin_training_button = QtGui.QPushButton(Dialog)
        self.begin_training_button.setMinimumSize(QtCore.QSize(75, 0))
        self.begin_training_button.setObjectName(_fromUtf8("begin_training_button"))
        self.verticalLayout.addWidget(self.begin_training_button)

        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)
        Dialog.setTabOrder(self.training_data_path, self.testing_data_path)
        Dialog.setTabOrder(self.testing_data_path, self.training_file_chooser_button)
        Dialog.setTabOrder(self.training_file_chooser_button, self.testing_file_chooser_button)
        Dialog.setTabOrder(self.testing_file_chooser_button, self.begin_training_button)

    def retranslateUi(self, Dialog):
        Dialog.setWindowTitle(_translate("Dialog", "Dialog", None))
        self.label.setText(_translate("Dialog", "<html><head/><body><p><span style=\" font-weight:600;\">Re-tra
        self.label_2.setText(_translate("Dialog", "Training Data Set", None))
        self.label_3.setText(_translate("Dialog", "Testing Data Set", None))
        self.testing_file_chooser_button.setText(_translate("Dialog", "Choose file", None))
        self.training_file_chooser_button.setText(_translate("Dialog", "Choose file", None))
        self.begin_training_button.setText(_translate("Dialog", "Begin Training", None))


'''
The program's trading strategy as modeled after the Research Backtesting Platform
as seen on https://www.quantstart.com/articles/Backtesting-a-Moving-Average-Crossover-in-Python-with-pandas

@author: ArielKenneth
'''

import sys
from PyQt4 import QtGui

from matplotlib.backends.backend_qt4agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.backends.backend_qt5 import NavigationToolbar2QT as NavigationToolbar
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick

import numpy as np
import pandas as pd
import talib

from backtest import Strategy, Portfolio
from aagfa import config

class MovingAverageCrossStrategy(Strategy):
    """
    Requires:
    symbol - A stock symbol on which to form a strategy on.
    bars - A DataFrame of bars for the above symbol.
    short_window - Lookback period for short moving average.
    long_window - Lookback period for long moving average."""

    def __init__(self, symbol, bars, short_window=100, long_window=400):
        self.symbol = symbol
        self.bars = bars

        self.short_window = short_window
        self.long_window = long_window

    def generate_signals(self):
        """Returns the DataFrame of symbols containing the signals
        to go long, short or hold (1, -1 or 0)."""
        signals = pd.DataFrame(index=self.bars.index)
        signals['signal'] = 0.0

        # Create the set of short and long simple moving averages over the
        # respective periods
        signals['short_mavg'] = pd.ewma(self.bars['Close'], span=self.short_window, min_periods=1)
        signals['long_mavg'] = pd.ewma(self.bars['Close'], span=self.long_window, min_periods=1)
        signals['rsi'] = talib.RSI(self.bars['Close'].values, timeperiod=8)
        signals['previous_rsi'] = signals['rsi'].shift(1)
        signals['previous_long'] = signals['long_mavg'].shift(1)
        signals['previous_short'] = signals['short_mavg'].shift(1)

        minimum_shift = 9 #RSI needs an 8-cell head start

        # Create a 'signal' (invested or not invested) when the short moving average crosses the long
        # moving average, but only for the period greater than the shortest moving average window
        buy_signals = np.where(
                ((signals['short_mavg'][minimum_shift:] > signals['long_mavg'][minimum_shift:]) &
                 (signals['previous_short'][minimum_shift:] < signals['previous_long'][minimum_shift:]) &
                 (signals['rsi'][minimum_shift:] >= 50) &
                (signals['previous_rsi'][minimum_shift:] < 50)),
                                                    1.0, 0.0)
        sell_signals = np.where(
                ((signals['short_mavg'][minimum_shift:] < signals['long_mavg'][minimum_shift:]) &
                 (signals['previous_short'][minimum_shift:] > signals['previous_long'][minimum_shift:]) &
                 (signals['rsi'][minimum_shift:] <= 50) &
                (signals['previous_rsi'][minimum_shift:] > 50)),
                                                    -1.0, 0.0)
        signals['signal'][minimum_shift:] = np.add(buy_signals, sell_signals)

        # Take the difference of the signals in order to generate actual trading orders
        signals['positions'] = signals['signal']
#          print signals.tail(50)

        return signals

class MarketOnOpenPortfolio(Portfolio):
    """Inherits Portfolio to create a system that purchases 100 units of
    a particular symbol upon a long/short signal, assuming the market
    open price of a bar.

    In addition, there are zero transaction costs and cash can be immediately
    borrowed for shorting (no margin posting or interest requirements).

    Requires:
    symbol - A stock symbol which forms the basis of the portfolio.
    bars - A DataFrame of bars for a symbol set.
    signals - A pandas DataFrame of signals (1, 0, -1) for each symbol.
    initial_capital - The amount in cash at the start of the portfolio."""

    def __init__(self, symbol, bars, signals, initial_capital=100000.0):
        self.symbol = symbol
        self.bars = bars
        self.signals = signals
        self.initial_capital = float(initial_capital)
        self.positions = self.generate_positions()

    def generate_positions(self):
        """Creates a 'positions' DataFrame that simply longs or shorts
        100 of the particular symbol based on the forecast signals of
        {1, 0, -1} from the signals DataFrame."""
        positions = pd.DataFrame(index=self.signals.index).fillna(0.0)
        positions[self.symbol] = 1000*self.signals['signal']
        return positions

    def backtest_portfolio(self):
        """Constructs a portfolio from the positions DataFrame by
        assuming the ability to trade at the precise market open price
        of each bar (an unrealistic assumption!).

        Calculates the total of cash and the holdings (market price of
        each position per bar), in order to generate an equity curve
        ('total') and a set of bar-based returns ('returns').

        Returns the portfolio object to be used elsewhere."""
        # Construct the portfolio DataFrame to use the same index
        # as 'positions' and with a set of 'trading orders' in the
        # 'pos_diff' object, assuming market open prices.
        portfolio = self.positions*self.bars['Open']
        pos_diff = self.positions.diff()

        # Create the 'holdings' and 'cash' series by running through
        # the trades and adding/subtracting the relevant quantity from
        # each column
        portfolio['holdings'] = (self.positions*self.bars['Open']).sum(axis=1)
        portfolio['cash'] = self.initial_capital - (pos_diff*self.bars['Open']).sum(axis=1).cumsum()

        # Finalise the total and bar-based returns based on the 'cash'
        # and 'holdings' figures for the portfolio
        portfolio['total'] = portfolio['cash'] + portfolio['holdings']
        portfolio['returns'] = portfolio['total'].pct_change()
        return portfolio

def backtest(symbol, filename, initial_amount=10000):
    # Obtain daily bars of AAPL from Yahoo Finance for the period
    # 1st Jan 1990 to 1st Jan 2002 - This is an example from ZipLine
    dateparse = lambda x: pd.datetime.strptime(x, '%Y.%m.%d %H:%M')
    bars = pd.read_csv(filename, parse_dates=[["Date","Time"]], date_parser=dateparse, index_col=0)

    # Create a Moving Average Cross Strategy instance with a short moving
    # average window of 100 days and a long window of 400 days
    mac = MovingAverageCrossStrategy(symbol, bars, short_window=5, long_window=12)
    signals = mac.generate_signals()

    # Create a portfolio of AAPL, with $100,000 initial capital
    portfolio = MarketOnOpenPortfolio(symbol, bars, signals, initial_amount)
    returns = portfolio.backtest_portfolio()

    return bars, signals, returns

class Window(QtGui.QDialog):
    def __init__(self, bars, signals, returns, parent=None):
        super(Window, self).__init__(parent)

        self.figure = plt.figure()
        self.canvas = FigureCanvas(self.figure)

        self.bars = bars
        self.signals = signals
        self.returns =  returns

        self.toolbar = NavigationToolbar(self.canvas, self)

        # set the layout
        layout = QtGui.QVBoxLayout()
        layout.addWidget(self.toolbar)
        layout.addWidget(self.canvas)
        self.setLayout(layout)

        self.plot()

    def plot(self):
        ''' plot some random stuff '''
        self.figure.subplots_adjust(hspace=.5)

        # Plot two charts to assess trades and equity curve
        ax1 = self.figure.add_subplot(211,  ylabel='Price of Instrument'.format())
        ax1.yaxis.set_major_formatter(mtick.FormatStrFormatter('%.4f'))

        # Plot the AAPL closing price overlaid with the moving averages
        self.bars['Close'].plot(ax=ax1, color='r', lw=2.)

        # Plot the "buy" trades against AAPL
        ax1.plot(self.signals.ix[self.signals.positions == 1.0].index,
                 self.signals.short_mavg[self.signals.positions == 1.0],
                 '^', markersize=10, color='m')

        # Plot the "sell" trades against AAPL
        ax1.plot(self.signals.ix[self.signals.positions == -1.0].index,
                 self.signals.short_mavg[self.signals.positions == -1.0],
                 'v', markersize=10, color='k')

        # Plot the equity curve in dollars
        ax2 = self.figure.add_subplot(212, ylabel='Portfolio value')
        ax2.yaxis.set_major_formatter(mtick.FormatStrFormatter('$%.f'))
        self.returns['total'].plot(ax=ax2, lw=2.)

        #Plot the "buy" and "sell" trades against the equity curve
        ax2.plot(self.returns.ix[self.signals.positions == 1.0].index,
                 self.returns.total[self.signals.positions == 1.0],
                 '^', markersize=10, color='m')
        ax2.plot(self.returns.ix[self.signals.positions == -1.0].index,
                 self.returns.total[self.signals.positions == -1.0],
                 'v', markersize=10, color='k')

        self.canvas.draw()

def main():
    filename = 'C:\Users\ArielKenneth\Desktop\data\EURUSD30_recent_test.csv'
    app = QtGui.QApplication(sys.argv)

    bars, signals, returns = backtest('EUR_USD', filename)
    main = Window(bars, signals, returns)
    main.setWindowTitle(config._program_name + ' - Backtesting Results')
    main.showMaximized()

    sys.exit(app.exec_())

if __name__ == '__main__':
    main()

'''
Created on Jun 1, 2015

@author: ArielKenneth
'''

import time
import StringIO

from PyQt4 import QtGui, QtCore

import matlab_funcs
import config
import preprocess

class TrainingDialog(QtGui.QDialog):
    def __init__(self, training_path, testing_path, parent=None):
        super(TrainingDialog, self).__init__(parent)

        self.setWindowTitle(config._program_name_short + " - Training")

        self.verticalLayout = QtGui.QVBoxLayout()
        self.label = QtGui.QLabel("The program is currently processing the training data. "
                         "\nThroughout the training, you will see several MATLAB progress dialogs pop up."
                         "\nThis typically takes up to 4-6 hours. Please see progress below.")
        self.verticalLayout.addWidget(self.label)
        self.progressBar = QtGui.QProgressBar()
        self.verticalLayout.addWidget(self.progressBar)
        self.textBrowser = QtGui.QTextBrowser()
        self.verticalLayout.addWidget(self.textBrowser)
        self.okButton = QtGui.QPushButton("Finished")
        self.verticalLayout.addWidget(self.okButton)
        self.okButton.setDisabled(True)

        self.setLayout(self.verticalLayout)

        self.pbar_value = 0
        self.finished = False

        self.training_thread = TrainingThread(training_path, testing_path)
        self.training_thread.updated.connect(self.update_progress)
        self.training_thread.errorEncountered.connect(self.notify_error)
        self.training_thread.finished.connect(self.activateOk)

        self.training_thread.start()

    def update_progress(self, text, progress_int):
        self.pbar_value += progress_int
        self.progressBar.setValue(self.pbar_value * 100.0 / 13.0)
        self.textBrowser.insertHtml('<p style="color:black">{0}</p>'.format(text))
        self.textBrowser.verticalScrollBar().setValue(
                        self.textBrowser.verticalScrollBar().maximum())

    def notify_error(self, text):
        # TODO something special here
        self.textBrowser.insertHtml('<p style="color:red">{0}</p>'.format(text))
        self.training_thread.exit(-1)

    def activateOk(self):
        self.okButton.setEnabled(True)

    def closeEvent(self, event):
        if self.okToContinue():
            self.training_thread.quit()
            event.accept()
        else:
            event.ignore()

    def okToContinue(self):
        reply = QtGui.QMessageBox.question(self, 'Message',
            "Are you sure you want to quit? This may leave the program in an undesirable state", QtGui.QMessageB
            QtGui.QMessageBox.No, QtGui.QMessageBox.No)

        return reply == QtGui.QMessageBox.Yes

class TrainingThread(QtCore.QThread):

    # These signal sends a message, and 1 for progress messages
    # 14 progress ticks will be sent in total
    # 1 for preporcessing, 1 for MATLAB init, 12 for all FIS
    updated = QtCore.pyqtSignal(str, float)
    errorEncountered = QtCore.pyqtSignal(str)

    def __init__(self, training_path, testing_path, parent=None):
        super(TrainingThread, self).__init__(parent)
        self.training_file = training_path
        self.testing_file = testing_path

    # Recommended by https://joplaete.wordpress.com/2010/07/21/threading-with-pyqt4/
    def __del__(self):
            print "Thread terminated"
            self.wait()

        def run(self):
            self.updated.emit("Training commencing...<br />", 0)
            start_time = time.time()

            try:
                self.updated.emit("Pre-processing files...<br />", 0)
                preprocess.preprocess_OHLC_training(self.training_file)
                preprocess.preprocess_OHLC_testing(self.testing_file)
                self.updated.emit("Training and testing files have been pre-processed<br />", 0.5)

                self.updated.emit("MATLAB Engine is being initialized...<br />", 0)
                eng = matlab_funcs.init_matlab()
                self.updated.emit("MATLAB Engine has been initialized!<br />", 0.5)
            except Exception, e:
                self.errorEncountered.emit(str(e) +
                                           "<br />Please return to the previous page and check your input.<br />")
                return

            for i in range(config._horizon):
                out = StringIO.StringIO()
                err = StringIO.StringIO()
                self.updated.emit('<br /><br />High (t+{0}) training...<br />'.format(i+1), 0)
                eng.anfis_init_train('training_high_{0}.csv'.format(i+1), 'high_{0}.fis'.format(i+1),
                                     stdout=out, stderr=err)
                if err.len > 0:
                    self.errorEncountered.emit(err.getvalue())
                    self.exit(-1)
                self.updated.emit('<PRE STYLE="font-family: Helvetica">'+out.getvalue()+"</PRE>", 1)

                out = StringIO.StringIO()
                err = StringIO.StringIO()
                self.updated.emit('<br /><br />Low (t+{0}) training...<br />'.format(i+1), 0)
                start_time = time.time()
                eng.anfis_init_train('training_low_{0}.csv'.format(i+1), 'low_{0}.fis'.format(i+1),
                                     stdout=out, stderr=err)
                if err.len > 0:
                    self.errorEncountered.emit(err.getvalue())
                    self.exit(-1)
                self.updated.emit('<PRE STYLE="font-family: Helvetica">'+out.getvalue()+"</PRE>", 1)

#          for _ in xrange(12):
#              self.updated.emit("Emit dummy<br />", 1)

            self.updated.emit("<br />Finished after %s seconds" % (time.time() - start_time), 0)

<?php

    // These variables define the connection information for your MySQL database
    // Change as applicable
    $username = "aagfa";
    $password = "u9r8swcn2vVQBSjc";
    $host = "localhost";
    $dbname = "broker";

// UTF-8 is a character encoding scheme that allows you to conveniently store
    // a wide varienty of special characters, like     or     , in your database.
    // By passing the following $options array to the database connection code we
    // are telling the MySQL server that we want to communicate with it using UTF-8
    // See Wikipedia for more information on UTF-8:
    // http://en.wikipedia.org/wiki/UTF-8
    $options = array(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8');

    // A try/catch statement is a common method of error handling in object oriented code.
    // First, PHP executes the code within the try block.  If at any time it encounters an
    // error while executing that code, it stops immediately and jumps down to the
    // catch block.  For more detailed information on exceptions and try/catch blocks:
    // http://us2.php.net/manual/en/language.exceptions.php
    try
    {
        // This statement opens a connection to your database using the PDO library
        // PDO is designed to provide a flexible interface between PHP and many
        // different types of database servers.  For more information on PDO:
        // http://us2.php.net/manual/en/class.pdo.php
        $db = new PDO("mysql:host={$host};dbname={$dbname};charset=utf8", $username, $password, $options);
    }
    catch(PDOException $ex)
    {
        // If an error occurs while opening a connection to your database, it will
        // be trapped here.  The script will output an error and stop executing.
        // Note: On a production website, you should not output $ex->getMessage().
        // It may provide an attacker with helpful information about your code
        // (like your database username and password).
        die("Failed to connect to the database: " . $ex->getMessage());
    }

    // This statement configures PDO to throw an exception when it encounters
    // an error.  This allows us to use try/catch blocks to trap database errors.
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // This statement configures PDO to return database rows from your database using an associative
    // array.  This means the array will have string indexes, where the string value
    // represents the name of the column in your database.
    $db->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);

    // This block of code is used to undo magic quotes.  Magic quotes are a terrible
    // feature that was removed from PHP as of PHP 5.4.  However, older installations
    // of PHP may still have magic quotes enabled and this code is necessary to
    // prevent them from causing problems.  For more information on magic quotes:
    // http://php.net/manual/en/security.magicquotes.php
    if(function_exists('get_magic_quotes_gpc') && get_magic_quotes_gpc())
    {
        function undo_magic_quotes_gpc(&$array)
        {
            foreach($array as &$value)
            {
                if(is_array($value))
                {
                    undo_magic_quotes_gpc($value);
                }
                else
                {
                    $value = stripslashes($value);
                }
            }
        }

        undo_magic_quotes_gpc($_POST);
        undo_magic_quotes_gpc($_GET);
        undo_magic_quotes_gpc($_COOKIE);
    }

    // This tells the web browser that your content is encoded using UTF-8
    // and that it should submit content back to you using UTF-8
    header('Content-Type: text/html; charset=utf-8');

    // This initializes a session.  Sessions are used to store information about
    // a visitor from one web page visit to the next.  Unlike a cookie, the information is
    // stored on the server-side and cannot be modified by the visitor.  However,
    // note that in most cases sessions do still use cookies and require the visitor
    // to have cookies enabled.  For more information about sessions:
    // http://us.php.net/manual/en/book.session.php
    session_start();

    // Note that it is a good practice to NOT end your PHP files with a closing PHP tag.
    // This prevents trailing newlines on the file from being included in your output,
    // which can cause problems with redirecting users.

?>

<?php

/*
Our "config.inc.php" file connects to database every time we include or require
it within a php script.  Since we want this script to add a new user to our db,
we will be talking with our database, and therefore,
let's require the connection to happen:
*/
require("config.inc.php");

//if posted data is not empty
if (!empty($_POST)) {
    //If the username or password is empty when the user submits
    //the form, the page will die.
    //Using die isn't a very good practice, you may want to look into
    //displaying an error message within the form instead.
    //We could also do front-end form validation from within our Android App,
    //but it is good to have a have the back-end code do a double check.
    if (empty($_POST['acc_id'])) {


        // Create some data that will be the JSON response
        $response["success"] = 0;
        $response["message"] = "Please Enter an Account ID";

        //die will kill the page and not execute any code below, it will also
        //display the parameter... in this case the JSON data our Android
        //app will parse
        die(json_encode($response));
    }

    //if the page hasn't died, we will check with our database to see if there is
    //already a user with the username specificed in the form.  ":user" is just
    //a blank variable that we will change before we execute the query.  We
    //do it this way to increase security, and defend against sql injections
    $query = "SELECT * FROM Orders WHERE Account_ID = :acc_id ORDER BY Order_ID DESC";
    //now lets update what :user should be
    $query_params = array(
        ':acc_id' => $_POST['acc_id']
    );
    //Now let's make run the query:
    try {
        // These two statements run the query against your database table.
        $stmt   = $db->prepare($query);
        $stmt->execute($query_params);
    }
    catch (PDOException $ex) {
        // For testing, you could use a die and message.
        //die("Failed to run query: " . $ex->getMessage());

        //or just use this use this one to product JSON data:
        $response["success"] = 0;
        $response["message"] = "Database Error1. Please Try Again!";
        die(json_encode($response));
    }
    $results = $stmt->fetchAll(PDO::FETCH_ASSOC);
    die(json_encode($results));

} else {
?>
        <h1>Get Orders for Account ID</h1>
        <form action="get_orders.php" method="post">
            Account ID:<br />
            <input type="text" name="acc_id" value="" />
            <br /><br />
            <input type="submit" value="Get Orders" />
        </form>
        <?php
}

?>


<?php

$api_key = 'jr-67a974b189236a96113aa93979d19939';

if (!empty($_POST)) {

list($from, $to) = explode("_", $_POST['inst']);

$data = file_get_contents(
    'http://jsonrates.com/get/?'.
    'from='.$from.
    '&to='.$to.
    '&apiKey='.$api_key
);
// die($data);
$json = json_decode($data);
$rate = (float) $json->rate;
echo $rate;

}

else {
?>
        <h1>Get Rates</h1>
        <form action="get_rate.php" method="post">
            Instrument:<br />
            <input type="text" name="inst" value="" />
            <br /><br />
            <input type="submit" value="Get Exchange Rate" />
        </form>
        <?php
}


?>


<?php

/*
Our "config.inc.php" file connects to database every time we include or require
it within a php script.  Since we want this script to add a new user to our db,
we will be talking with our database, and therefore,
let's require the connection to happen:
*/
require("config.inc.php");

//if posted data is not empty
if (!empty($_POST)) {
    //If the username or password is empty when the user submits
    //the form, the page will die.
    //Using die isn't a very good practice, you may want to look into
    //displaying an error message within the form instead.
    //We could also do front-end form validation from within our Android App,
    //but it is good to have a have the back-end code do a double check.
    if (empty($_POST['acc_id'])) {


        // Create some data that will be the JSON response
        $response["success"] = 0;
        $response["message"] = "Please Enter an Account ID";

        //die will kill the page and not execute any code below, it will also
        //display the parameter... in this case the JSON data our Android
        //app will parse
        die(json_encode($response));
    }

    //if the page hasn't died, we will check with our database to see if there is
    //already a user with the username specificed in the form.  ":user" is just
    //a blank variable that we will change before we execute the query.  We
    //do it this way to increase security, and defend against sql injections
    $query = "SELECT COUNT(*) FROM Orders WHERE Account_ID = :acc_id";
    //now lets update what :user should be
    $query_params = array(
        ':acc_id' => $_POST['acc_id']
    );
    //Now let's make run the query:
    try {
        // These two statements run the query against your database table.
        $stmt   = $db->prepare($query);
        $stmt->execute($query_params);
    }
    catch (PDOException $ex) {
        // For testing, you could use a die and message.
        //die("Failed to run query: " . $ex->getMessage());

        //or just use this use this one to product JSON data:
        $response["success"] = 0;
        $response["message"] = "Database Error1. Please Try Again!";
        die(json_encode($response));
    }

    $result = $stmt->fetch();
    $response["success"] = 1;
    $response["Account_ID"] = $_POST['acc_id'];
    $response["num_orders"] = intval($result['COUNT(*)']);
    die(json_encode($response));

} else {
?>
        <h1>Get Status for Account ID</h1>
        <form action="get_status.php" method="post">
            Account ID:<br />
            <input type="text" name="acc_id" value="" />
            <br /><br />
            <input type="submit" value="Get Orders" />
        </form>
        <?php
}

?>


<?php

require("config.inc.php");

//if posted data is not empty
if (!empty($_POST)) {
    $query =
        "UPDATE orders SET Take_Profit=:tp, Stop_Loss=:sl WHERE Order_ID=:order_id";

    //Again, we need to update our tokens with the actual data:
    $query_params = array(
        ':order_id' => $_POST['order_id'],
        ':tp' => $_POST['tp'],
        ':sl' => $_POST['sl']
    );
    //time to run our quer
    try {
        $stmt   = $db->prepare($query);
        $result = $stmt->execute($query_params);
    }
    catch (PDOException $ex) {
        //or just use this use this one:
        $response = 1;
        echo $response;
    }
    $response = 1;
    echo json_encode($response);


} else {
?>
    <h1>Modify Order</h1>
    <form action="editprofile.php" method="post">
        Order ID:<br />
        <input type="text" name="order_id" value="" />
        <br /><br />
        Take Profit:<br />
        <input type="text" name="tp" value="" />
        <br /><br />
        Stop Loss:<br />
        <input type="password" name="sl" value="" />
        <br /><br />
        <input type="submit" value="Modify Order" />
    </form>
    <?php
}

?>


<?php

/*
Our "config.inc.php" file connects to database every time we include or require
it within a php script.  Since we want this script to add a new user to our db,
we will be talking with our database, and therefore,
let's require the connection to happen:
*/
require("config.inc.php");

//if posted data is not empty
if (!empty($_POST)) {

    //get execution price
    $api_key = 'jr-67a974b189236a96113aa93979d19939';
    list($from, $to) = explode("_", $_POST['inst']);
    $data = file_get_contents(
        'http://jsonrates.com/get/?'.
```

```
            'from='.$from.
            '&to='.$to.
            '&apiKey='.$api_key
        );
        $json = json_decode($data);
        $exec_price = (float) $json->rate;

        $query = " INSERT INTO Orders (Account_ID, Units, Side, Instrument, Price, Take_Profit, Stop_Loss) VALUES (

        //Again, we need to update our tokens with the actual data:
        $query_params = array(
            ':acc_id' => $_POST['acc_id'],
            ':units' => $_POST['units'],
            ':side' => $_POST['side'],
            ':inst' => $_POST['inst'],
            ':price' => $exec_price,
            ':tp' => $_POST['tp'],
            ':sl' => $_POST['sl']
        );

        //time to run our query, and create the user
        try {
            $stmt   = $db->prepare($query);
            $result = $stmt->execute($query_params);
        }
        catch (PDOException $ex) {
            // For testing, you could use a die and message.
            //die("Failed to run query: " . $ex->getMessage());

            //or just use this use this one:
            $response["success"] = 0;
            $response["message"] = "Database Error2. Please Try Again!";
            die(json_encode($response));
        }

        //encode final response
        $response["success"] = 1;
        $response["price"] = $exec_price;
        echo json_encode($response);

        //for a php webservice you could do a simple redirect and die.
        //header("Location: login.php");
        //die("Redirecting to login.php");


} else {
?>
        <h1>Create Order</h1>
        <form action="place_order.php" method="post">
            Account ID:<br />
            <input type="text" name="acc_id" value="" />
            <br /><br />
            Side:<br />
            <input type="text" name="side" value="" />
            <br /><br />
        Units:<br />
        <input type="text" name="units" value="" />
        <br /><br />
        Instrument:<br />
        <input type="text" name="inst" value="" />
        <br /><br />
        T/P:<br />
        <input type="text" name="tp" value="" />
        <br /><br />
        S/L:<br />
        <input type="text" name="sl" value="" />
        <br /><br />
            <input type="submit" value="Place Order" />
        </form>
        <?php
}

?>


DROP DATABASE IF EXISTS broker;
CREATE DATABASE broker;
USE broker;

CREATE USER

CREATE TABLE Orders (
        Order_ID INTEGER PRIMARY KEY AUTO_INCREMENT,
        Account_ID VARCHAR(7),
        Units INTEGER,
        Side VARCHAR(4),
        Instrument VARCHAR(7),
        Order_Time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        Price DECIMAL(8,5),
        Take_Profit DECIMAL(8,5),
        Stop_Loss DECIMAL(8,5)
);

INSERT INTO Orders (Account_ID, Units, Side, Instrument, Price, Take_Profit, Stop_Loss)
VALUES (1234567, 1000, 'buy', 'EUR_USD', 12.2345, 1.54321, 1.12345);

SELECT * FROM Orders WHERE Account_ID=1234567;
```

XII. Acknowledgement

Where to begin? I find it... jarring having to write this in a rush. Haha. (If you must know, I had to defend, revise, and have this all finalized on the same day.) There's so much to thank about and so many people to thank them for.

Off the top of my head, I first want to thank my advisers. Yes, advisers. I had three. Thank you, Sir Richard Bryann Chua for taking me under your wings as one of your advisees. I took that as a vouch of confidence in my ideas. But alas, ideas don't always work. I'm still thankful because you pointed me to my next adviser, Sir Geoffrey Solano. Thank you, sir, for working so patiently with me and taking the time, online or offline, to discuss my work. Sadly, I wasn't able to finish before you had to go and focus on your Ph.D. This is where Ma'am Perl comes in. Even during the first discussions of my SP, Ma'am Perl acted as a consultant of sorts for me and Sir Solano. Thank you, ma'am, for working with me even as an "adopted" advisee. I've seen you answer me and work through headaches and vacation inertia just to cater to my questions and to your other advisees.

I really do sincerely appreciate you, Ma'am Perl and Sir Solano. *Saan kaya ako pupulutin kung di niyo ako tiniis?* Haha.

Thank you also to all the professors and instructors that I had. UP was a formative experience and you were all a major part of it. Thank you to the support staff of UP Manila, both wonderful and not-so-wonderful. Haha. a special shoutout to the Cash Office staff! The remaining part would be my classmates. Thank you, too, guys!

For my friends, you know who you are (you should), I also thank you. I love you, guys! If you're from UPM, thank you for making my college days less harrowing than it should be. Haha. For my church peeps, thank you for praying with me and being wonderful people to grow with.

For those that pushed, encouraged, prodded, berated, and yelled me into finishing this; thank you! When I couldn't move myself, you gave me that much needed push. *Saan kaya ako pupulutin kung di niyo ako kinulit?* Haha.

For everyone else that I couldn't remember to put in here (*time pressured eh*), thank you, too! I might not remember what you did, but it could have very well been integral to me finishing my degree.

For my family, Mama Lynn, Papa Bhoy, Princess, Kyle, Prince, Chloe; thank you! I love you so much! This one's for you. When I couldn't push myself to do this for me, I thought I'm doing this for you. You stood by me through the many, *I mean many*, challenges that led to me finishing this. A family like you is love exemplified. Especially you, Mama and Papa. I'm glad that I can now finally say that I'm doing you proud with a college degree—from UP no less! You deserve the recognition.

Finally, I must acknowledge that I couldn't have done this alone. God brought me through all of this. With all my weaknesses and shortcomings, I couldn't imagine myself doing all this through my own ability or will. I needed the mighty hand of God, holding my world and letting the boundary lines fall into pleasant places. And thus, I don't want to take the glory for myself. It is God's and God's alone.

Not to us, Lord, not to us
but to your name be the glory,
because of your love and faithfulness.
-Psalm 115:1

Thank you, my Lord, my God, my King.