Evolving Hierarchical Temporal Memory-Based Trading Models

Patrick Gabrielsson, Rikard König, and Ulf Johansson

School of Business and Information Technology, University of Borås, Sweden
{patrick.gabrielsson,rikard.konig,ulf.johansson}@hb.se

Abstract. We explore the possibility of using the genetic algorithm to optimize trading models based on the Hierarchical Temporal Memory (HTM) machine learning technology. Technical indicators, derived from intraday tick data for the E-mini S&P 500 futures market (ES), were used as feature vectors to the HTM models. All models were configured as binary classifiers, using a simple buy-and-hold trading strategy, and followed a supervised training scheme. The data set was partitioned into multiple folds to enable a modified cross validation scheme. Artificial Neural Networks (ANNs) were used to benchmark HTM performance. The results show that the genetic algorithm succeeded in finding predictive models with good performance and generalization ability. The HTM models outperformed the neural network models on the chosen data set and both technologies yielded profitable results with above average accuracy.

1 Introduction

One recent machine learning technology that has shown good potential in algorithmic trading is Hierarchical Temporal Memory (HTM). It borrows concepts from neural networks, Bayesian networks and makes use of spatiotemporal clustering algorithms to handle noisy inputs and to create invariant representations of patterns discovered in its input stream. In a previous paper [3], an initial study was carried-out where the predictive performance of the HTM technology was investigated within algorithmic trading of financial markets. The study showed promising results, in which the HTM-based algorithm was profitable across bullish-, bearish and horizontal market trends, yielding comparable results to its neural network benchmark. Although, the previous work lacked any attempt to produce near optimal trading models.

One popular group of optimization methods are evolutionary optimization methods. The simplest evolutionary optimization technique is the genetic algorithm. This paper extends the HTM-based trading algorithm, developed in the previous work, by employing the genetic algorithm as an optimization method. Once again, neural networks are used as the benchmark technology.

2 Background

2.1 Technical Indicators

This study uses technical indicators, derived from the E-mini S&P 500 futures market (ES), as features vectors to the HTM models. Technical indicators are commonly used

A.I. Esparcia-Alcázar et al. (Eds.): EvoApplications 2013, LNCS 7835, pp. 213–222, 2013.

in technical analysis, which is the forecasting of future price movements based on an examination of past price movements. To aid in the process, technical charts and technical indicators are used to discover price trends and to time market entry and exit. Technical analysis has its roots in Dow Theory, developed by Charles Dow in the late 19th century and later refined and published by William Hamilton in the first edition (1922) of his book *"The Stock Market Barometer"*. Robert Rhea developed the theory even further in *"The Dow Theory"*, first published in 1932. Modern day technical analysis [12] is based on the tenets from Dow Theory; prices discount everything, price movements are not totally random and the only thing that matters is *what* the current price levels are. The reason *why* the prices are at their current levels is not important.

2.2 Predictive Modeling

A common way of modeling financial time series is by using predictive modeling techniques [17]. The time series, consisting of tick data, is usually aggregated into bars of intraday-, daily-, weekly-, monthly- or yearly data. From the aggregated data, features (attributes) are created that better describe the data, for example technical indicators. The purpose of predictive modeling is to produce models capable of predicting the value of one attribute, the dependent variable, based on the values of the other attributes, the independent variables. If the dependent variable is discrete, the modeling task is categorized as classification. Classification is the task of classifying objects into one of several predefined classes by finding a classification model capable of predicting the value of the dependent variable using the independent variables.

One common approach to solving a classification problem splits the data set into a training-, validation- and test set. The training set is used to train a classification model, while the validation set is used to determine the performance of the classifier and when to stop training in order to avoid over-fitting the model. The model is subsequently applied to the test set to determine the performance of the classifier on previously unseen data, i.e. its generalization ability.

This paper uses a modified *k*-fold cross validation technique, in which the data set is partitioned into *k* folds. Classification models are then trained on *k-2* folds, where the two remaining folds are used as the validation set and the test set respectively. This procedure is then repeated, using a different fold as the validation set and test set in each run. The performance of the classification models are then averaged over all runs to produce a final performance measure.

2.3 Hierarchical Temporal Memory

Hierarchical Temporal Memory (HTM) is a machine learning technology based on memory-prediction theory of brain function described by Jeff Hawkins in his book On Intelligence [8] and are modeled after the structure and behavior of the mammalian neocortex. Just as the neocortex is organized into layers of neurons, HTMs are organized into tree-shaped hierarchies of nodes, where each node implements a common learning algorithm [5].

The input to any node is a temporal sequence of patterns. Bottom layer nodes sample simple quantities from their input and learn how to assign meaning to them in the form of *beliefs*. A belief is the probability that a certain cause of a pattern in the input stream is currently being sensed by the node. As the information ascends the tree-shaped hierarchy of nodes, it incorporates beliefs covering a larger spatial area over a longer temporal period. Higher-level nodes learn more sophisticated causes of patterns in the input stream [9].

Frequently occurring sequences of patterns are grouped together to form temporal groups, where patterns in the same group are likely to follow each other in time. This contraption, together with probabilistic and hierarchical processing of information, gives HTMs the ability to predict what future patterns are most likely to follow currently sensed input patterns. This is accomplished through a top-down procedure, in which higher-level nodes propagate their beliefs to lower-level nodes in order to update their belief states, i.e. conditional probabilities [6-7].

HTMs use Belief Propagation (BP) to disambiguate contradicting information and create mutually consistent beliefs across all nodes in the hierarchy [14]. This makes HTMs resilient to noise and missing data, and provides for good generalization behavior.

With support of all the favorable properties mentioned above, the HTM technology constitutes an ideal candidate for predictive modeling of financial time series, where future price levels can be estimated from current input with the aid of learned sequences of historical patterns.

In 2005 Jeff Hawkins co-founded the company Numenta Inc, where the HTM technology is being developed. Numenta provide a free legacy version of their development platform, NuPIC, for research purposes. NuPIC version 1.7.1 [13] was used to create all models in this study.

3 Related Work

Following the publication of Hawkins' memory-prediction theory of brain function [8], supported by Mountcastle's unit model of computational processes across the neocortex [11], George and Hawkins implemented an initial mathematical model of the Hierarchical Temporal Memory (HTM) concept and applied it to a simple pattern recognition problem as a successful proof of concept [4]. This partial HTM implementation was based on a hierarchical Bayesian network, modeling invariant pattern recognition behavior in the visual cortex.

An independent implementation of George's and Hawkins' hierarchical Bayesian network was created in [18], in which its performance was compared to a backpropagation neural network applied to a character recognition problem. The results showed that a simple implementation of Hawkins' model yielded higher pattern recognition rates than a standard neural network implementation.

In [19], the HTM technology was benchmarked with a support vector machine (SVM). The most interesting part of this study was the fact that the HTM's performance was similar to that of the SVM, even though the rigorous preprocessing of the raw data was omitted for the HTM. In [2] a novel study was conducted, in which the HTM technology was employed to a spoken digit recognition problem with promising results.

The HTM model was modified in [10] to create a Hierarchical Sequential Memory for Music (HSMM). This study is interesting since it investigates a novel application for HTM-based technologies, in which the order and duration of temporal sequences of patterns are a fundamental part of the domain. A similar approach was adopted in [15] where the topmost node in a HTM network was modified to store ordered sequences of temporal patterns for sign language recognition.

In 2011 Fredrik Åslin conducted an initial evaluation of the HTM technology applied to algorithmic trading [1]. In [3], the predictive performance of the HTM technology was investigated within algorithmic trading of financial markets. The study showed promising results, in which the HTM-based algorithm was profitable across bullish-, bearish and horizontal market trends, yielding comparable results to its neural network benchmark.

4 Method

4.1 Data Acquisition and Feature Extraction

The S&P (Standard and Poor's) 500 E-mini index futures contract (ES), traded on the Chicago Mercantile Exchange's Globex electronic platform, was chosen for the research work. Intra-minute data was downloaded from Slickcharts [16] which provides free historical tick data for E-mini contracts. Two months worth of ES market data (5^{th} July – 2^{nd} September 2011) was used to train, validate and test the HTM classifiers using a cross-validation approach.

The data was aggregated into one-minute epochs (bars), each including the open-, high-, low- and close prices, together with the 1-minute trade volume. Missing data points, i.e. missing tick data for one or more minutes, was handled by using the same price levels as the previously existing aggregated data point.

A set of 12 technical indicators were extracted from the aggregated data in order to populate the feature vectors for the classification models (the default parameter values used, in minutes, are enclosed within parentheses); Percentage Price Oscillator (12,26), Percentage Price Oscillator Signal Line (9), PPO Histogram (12,26,9), Relative Strength Indicator (14), William's %R (14), Normalized Volatility Indicator (10 and 20), Chaikin Money Flow (20), Bollinger Band %B (20,2), Rate Of Change (12), Fast Stochastic Oscillator (14), Fast Stochastic Oscillator Signal Line (3).

4.2 Dataset Partitioning and Cross Validation

The dataset was split into a number of folds in order to support a cross-validation approach. This was accomplished by creating a large enough window of size $N=22533$ to be used as the initial training set, followed by partitioning the remaining dataset into $k=5$ folds of size $M=7035$. The model was then trained using the training window and validated on *the closest fold ahead of the training window*. This was done for all individuals through a number of generations using the genetic algorithm. The fittest individual from the final generation was then tested on *the closest fold ahead of the validation fold*. Next, the window of size N was rolled forward M data points to include the first fold in the training set and, hence, omitting the oldest M

data points of the initial window. The model was then trained on the second window of size *N*, validated on the next fold ahead of the training window and tested on the closest fold ahead of the validation fold. This procedure was repeated until *k-1* folds had been used as the validation set once and *k-1* folds had been used as the test set once. The performance measure obtained from all *k-1* validations was then averaged to yield an overall performance measure for the model, similar to normal *k*-fold cross validation. Similarly, the performance measure obtained from all *k-1* test folds was averaged to yield an overall performance measure for the fittest models on the test set. Finally, since the genetic algorithm is non-deterministic, the whole procedure was repeated three times and a final averaged performance measure was obtained by averaging the averaged performance measures from all three individual iterations.

The training, validation and test datasets are shown in Fig.1. The initial training window (training window 01) is show to the far left in the figure, followed by validation window 01 and test window 01. Validation and test windows 02-04 are also depicted in the figure and the range of training windows 01-04 are shown along the bottom of the figure. The lighter patch to the left in the figure shows the buffer used to get the moving average calculations going for the technical indicators.

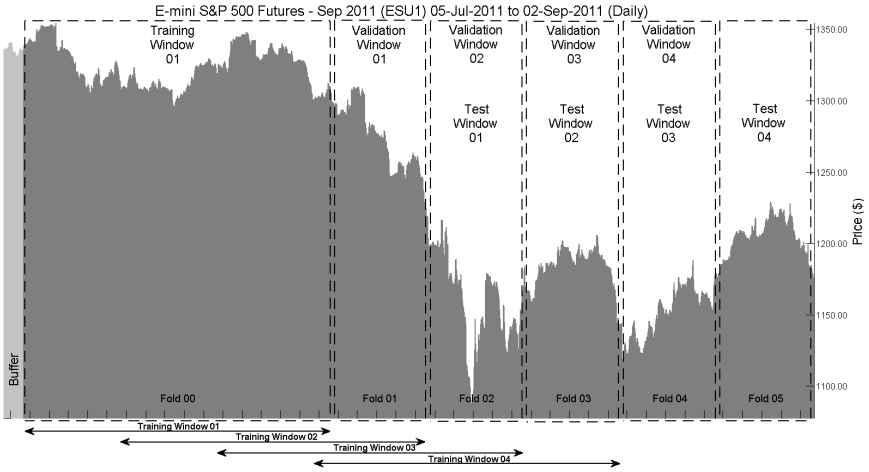

Fig. 1. The dataset for the E-mini S&P 500

4.3 Classification Task and Trading Strategy

The classification task was defined as a 2-class problem:

- Class 0 - The price will not rise >= 2 ticks at the end of the next ten-minute period.
- Class 1 - The price will rise >= 2 ticks at the end of the next ten-minute period.

A tick is the minimum amount a price can be incremented or decremented. For the E-mini S&P 500, the minimum tick size is 0.25 points, where each tick is worth $12.50 per contract. A simple buy-and-hold trading strategy was adopted; if the market is predicted to rise with at least 2 ticks at the end of the next 10-minute period, then buy 1 contract of ES, hold on to it, and then sell it at the end of the period.

4.4 Performance Measure and Fitness Function

The PNL (profit and loss) was chosen as the performance measure. Furthermore, trading fees consisting of $3 per contract and round trip were deducted to yield the final PNL, i.e. $3 for every true positive or false positive. This performance measure was also used as the fitness function for the genetic algorithm. In order to accommodate negative PNLs, tournament selection was used.

4.5 Experiments

The set of feature vectors were fed to the HTM's sensor, whereas the set of class vectors were fed to the category sensor during training as shown in Fig 2. The top-level node receives input from the hierarchically processed feature vector and from the class vector via the category sensor. The output from the top node is class 0 or 1.

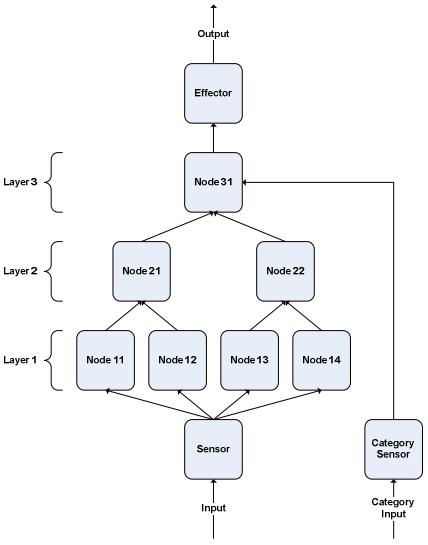

Fig. 2. HTM network configured as a classifier

An HTM network has a number of different parameters that can be optimized. The most important parameters, shown in Table 1, were chosen for the optimization task.

The resulting HTM chromosome consisted of a bit string of length 274. The genetic algorithm was configured to use tournament selection with a tournament size of 4, single-point crossover with a 80% crossover probability and bit-flip mutation with a 4% mutation probability. A population of 50 individuals were evolved over 20 generations and elitism was used.

Each time the genetic algorithm requested a fitness score from an individual, the following sequence of events took place; firstly, an HTM network was created with the parameter settings derived from the individual's bit string pattern. Next, the HTM was trained on the data points contained within the training window and validated on the corresponding validation fold. Each training and validation step produced a performance measure consisting of the PNL subtracted by the total trading cost

Table 1. HTM Parameters

Parameter	Description
Network Topology	Number of layers, including number of nodes per layer
Coincidences	The maximum number of spatial patterns that can be learned
Max Distance	Max. Euclidean distance for clustering spatial patterns to centroids
Sigma	Sigma for the radial-basis function in Gaussian inference mode
SP Algorithm	Spatial Pooler's inference algorithm {Gaussian, K^{th} Root Product}
Groups	The maximum number of temporal patterns that can be learned
Complexity Model	Complexity of the sequence model built by the sequencer [0 - 1.0]
Window Count	Number of windows over which the sequencer will build a model
Window Length	Samples per window where the sequencer will build a model
TP Algorithm	Temporal Pooler's inference algorithm {Max Prop, Sum Prop, TBI}
Transition Memory	How far back in time to look for temporal transitions

(calculated as $3 per trade including brokerage-, transaction- and exchange fees). At the end of each generation, each individuals' performance measure was used as the individual's fitness value by the genetic algorithm.

The neural network benchmark consisted of a recurrent network with feedback loops from the hidden layers. The number of hidden layers, nodes per hidden layer, the learning rate parameter and the momentum were encoded in a chromosome and optimized by the genetic algorithm. Network permutations were left entirely to the discretion of the genetic algorithm, but restricted to a maximum of two layers. The backpropagation learning algorithm was used to update the network weights during training. Sigmoid activation functions were used for all neurons except the output layer which used a softmax activation function. The resulting chromosome had a bit length of 25. The other settings for the genetic algorithm were the same as for the HTM models.

Both technologies were run though Python code, where the execution time requirements for the HTM models were more demanding than the neural networks.

5 Results

The optimization results show that a three-layer HTM was preferred over a two- or four-layer topology. It was also obvious that the Gaussian inference algorithm was preferred for the spatial pooler in every node. Furthermore, a Sigma value close to 1 standard deviation worked best together with the Gaussian inference algorithm. The temporal pooler algorithm used Time Based Inference (TBI) for half the models on average, whereas the maxProp algorithm was preferred for Flash Inference. The remaining parameters settings varied quite a lot amongst the various models.

All models showed positive PNLs in all datasets with minor differences between the validation- and test sets on average. The differences between the validation- and test sets for model accuracy and precision were also insignificant. There was a small loss in accuracy between the training set and the validation set and a noticeable decay in precision. One interesting observation is that even though the number of bad trades (False Positives) outnumber the number of good trades (True Positives) in both the

validation- and test sets, all models still yielded positive PNLs. This suggests that the models identified major increases in price levels correctly. The averaged performance measures for each iteration are shown in Table 2, together with the total averaged performance measures (arithmetic mean for all three iterations).

Table 2. HTM Average Performance

Dataset	TP	TN	FN	FP	Accuracy	Precision	PNL
ITERATION 1							
Training	1464	14897	6066	101	0.726	0.921	$111933.00
Validation	126	3633	2212	173	0.612	0.425	$1958.50
Test	128	3602	2201	213	0.607	0.491	$873.00
ITERATION 2							
Training	1307	14874	6223	124	0.719	0.890	$97762.50
Validation	129	3633	2209	173	0.612	0.412	$2165.00
Test	113	3633	2216	182	0.610	0.439	$1023.50
ITERATION 3							
Training	985	14746	6545	252	0.698	0.832	$58020.00
Validation	101	3640	2237	166	0.609	0.397	$1386.00
Test	93	3656	2236	159	0.610	0.364	$473.00
TOTAL AVERAGE (ALL ITERATIONS)							
Training	1252	14839	6278	159	0.714	0.881	$89238.50
Validation	119	3635	2219	171	0.611	0.411	$1836.50
Test	111	3630	2217	185	0.609	0.431	$790.00

Table 3. ANN Average Performance

Dataset	TP	TN	FN	FP	Accuracy	Precision	PNL
ITERATION 1							
Training	200	14714	7333	288	0.662	0.399	$555.00
Validation	50	4274	2644	64	0.615	0.636	$1507.00
Test	35	4295	2645	57	0.616	0.555	$437.50
ITERATION 2							
Training	45	14936	7488	66	0.665	0.410	$499.00
Validation	14	4324	2681	13	0.617	0.646	$872.00
Test	12	4335	2669	17	0.618	0.503	$135.00
ITERATION 3							
Training	91	14894	7442	107	0.665	0.432	$4174.00
Validation	19	4316	2675	22	0.617	0.499	$773.00
Test	7	4342	2673	11	0.618	0.434	$140.00
TOTAL AVERAGE (ALL ITERATIONS)							
Training	112	14848	7421	154	0.664	0.413	$1742.50
Validation	28	4305	2667	33	0.616	0.594	$1051.00
Test	18	4324	2662	29	0.617	0.497	$237.50

For the ANNs, the genetic algorithm (GA) favored a single layer of hidden neurons. It also produced a higher number of neurons in the hidden layers as compared to the number of inputs. A low learning rate and momentum was selected by the GA.

The PNL was positive in all datasets except two test sets and the difference in model accuracy and precision between the various datasets was insignificant. Despite many ANN models having more bad trades (False Positives) than good trades (True Positives), the PNL was positive for all models but 2 when applied to the test sets. As with the HTM models, the ANN models identified major increases in price levels correctly. The average performance measures are shown in Table 3. All average PNLs are positive and model accuracy and precision are similar to each other in all datasets.

The results show that the HTM models outperformed the neural network models, yielding 2-8 times as much in PNL. Both technologies had above average accuracy in all datasets, whereas the precision was slightly below average in the validation- and test sets. Overall, both models were profitable. The positive PNLs in both the validation- and test sets suggest that both technologies produced models with good generalization abilities.

6 Conclusion

The results show that the genetic algorithm succeeded in finding predictive models with good performance and generalization ability. Although the HTM models outperformed the neural networks with regards to PNL, both technologies yielded profitable results with above average accuracy. The optimization results show that the HTM models prefer a 3-layer network topology with a variable max distance setting and number of coincidences and groups in each layer when applied to the E-mini S&P 500. A Gaussian inference algorithm with a Sigma setting close to 1 standard deviation was preferred by the spatial pooler in all nodes in the network. With respect to the temporal pooler's inference algorithm, flash inference and time-based inference were equally prevalent amongst the nodes. The neural network preferred a single hidden layer with a node count close to the dimensionality of its input and small values for its learning rate and momentum.

References

1. Åslin, F.: Evaluation of Hierarchical Temporal Memory in algorithmic trading, Department of Computer and Information Science, University of Linköping (2010)
2. Doremale, J.V., Boves, L.: Spoken Digit Recognition using a Hierarchical Temporal Memory, Brisbane, Australia, pp. 2566–2569 (2008)
3. Gabrielsson, P., Konig, R., Johansson, U.: Hierarchical Temporal Memory-based algorithmic trading of financial markets. In: 2012 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr), pp. 1–8 (2012)
4. George, D., Hawkins, J.: A hierarchical Bayesian model of invariant pattern recognition in the visual cortex, in. In: Proceedings of the IEEE International Joint Conference on Neural Networks, IJCNN 2005, vol. 3, pp. 1812–1817 (2005)

5. George, D., Jaros, B.: The HTM Learning Algorithms. Numenta Inc. (2007), http://www.numenta.com/htm-overview/education/Numenta_HTM_Learning_Algos.pdf
6. George, D., et al.: Sequence memory for prediction, inference and behaviour. Philosophical Transactions - Royal Society. Biological Sciences 364, 1203–1209 (2009)
7. George, D., Hawkins, J.: Towards a mathematical theory of cortical micro-circuits. PLoS Computational Biology 5, 1000532 (2009)
8. Hawkins, J., Blakeslee, S.: On Intelligence. Times Books (2004)
9. Hawkins, J., George, D.: Hierarchical Temporal Memory - Concepts, Theory and Terminology. Numenta Inc. (2006), http://www.numenta.com/htm-overview/education/Numenta_HTM_Concepts.pdf
10. Maxwell, J., et al.: Hierarchical Sequential Memory for Music: A Cognitive Model. International Society for Music Information Retrieval, 429–434 (2009)
11. Mountcastle, V.: An Organizing Principle for Cerebral Function: The Unit Model and the Distributed System, pp. 7–50. MIT Press (1978)
12. Murphy, J.: Technical Analysis of the Financial Markets, NY Institute of Finance (1999)
13. Numenta. NuPIC 1.7.1, http://www.numenta.com/legacysoftware.php
14. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann (1988)
15. Rozado, D., Rodriguez, F.B., Varona, P.: Optimizing Hierarchical Temporal Memory for Multivariable Time Series. In: Diamantaras, K., Duch, W., Iliadis, L.S. (eds.) ICANN 2010, Part II. LNCS, vol. 6353, pp. 506–518. Springer, Heidelberg (2010)
16. Slickcharts, E-mini Futures (2012), http://www.slickcharts.com
17. Tan, P.N., et al.: Introduction to Data Mining. Addison Wesley (2009)
18. Thornton, J., Gustafsson, T., Blumenstein, M., Hine, T.: Robust Character Recognition Using a Hierarchical Bayesian Network. In: Sattar, A., Kang, B.-H. (eds.) AI 2006. LNCS (LNAI), vol. 4304, pp. 1259–1264. Springer, Heidelberg (2006)
19. Thornton, J., Faichney, J., Blumenstein, M., Hine, T.: Character Recognition Using Hierarchical Vector Quantization and Temporal Pooling. In: Wobcke, W., Zhang, M. (eds.) AI 2008. LNCS (LNAI), vol. 5360, pp. 562–572. Springer, Heidelberg (2008)