Applying ICA and NARX networks for algorithmic trading

Attila Ceffer[1], Janos Levendovszky[1], and Norbert Fogarasi[1]

Department of Networked Systems and Services
Budapest University of Technology and Economics
Budapest, Hungary
ceffer@hit.bme.hu, levendov@hit.bme.hu, fogarasi@hit.bme.hu

Abstract. In this paper, a Nonlinear AutoRegressive network with eXogenous inputs (NARX) is proposed for algorithmic trading by predicting the future value of financial time series. This network is highly capable of modeling vector autoregressive VAR(p) time series. In order to avoid overfitting, the input is pre-processed by Independent Component Analysis (ICA) to filter out the most noise like component. In this way, the accuracy of the prediction is increased. The proposed algorithm has a reduced number of free parameters which makes fast learning and trading possible. The method is not only tested on single asset price series, but also on predicting the value of mean reverting portfolios obtained by maximizing the predictability parameter of VAR(1) processes. The tests were first performed on artificially generated data and then on real data selected from Exchange Traded Fund (ETF) time series including bid-ask spread. In both cases profit has been achieved by the proposed method.

Keywords: algorithmic trading, financial time series, neural networks, independent component analysis, mean reverting portfolio

JEL classification: C45

1 Introduction

In the recent decades, algorithmic trading combined with portfolio optimization has been one of the most intensively researched areas of financial mathematics. Ever since the seminal paper of Markowitz [14], selecting portfolios which are optimal in terms of risk-adjusted returns has been in the centre of interest of both academics and financial practitioners. There are many approaches to algorithmic trading on profitable portfolios ranging from prediction based trading [18, 21, 25] to mean reverting trading [4, 12, 20].

Many researchers use *autoregressive* (AR) models to describe financial time series. For model identification and prediction, the results of machine learning have been deployed using different learning architectures (e.g. Feed Forward Neural Networks, Support Vector Machines etc.) [11, 16, 24]. However, in most

of the cases these methods fail to provide high performance on real data due to overfitting (a large number of free parameters are built in and due to small training sets the model remains poorly trained), and because of the large amount of noise superposed on the data. Furthermore, profit resulted by algo-trading tends to be very slim in the presence of bid-ask spread and transaction costs.

To increase trading efficiency, one of the most important questions is how to decrease the side-effects of overfitting. One can reduce the number of free parameters of the model by minimizing the dimension of input data. In this way, one possible approach is to use *Principal Component Analysis* (PCA), however, by aggressive dimension reduction, one may also leave out important information from the input [6, 7]. A better approach to reduce the chance of overfitting is noise filtering using *Independent Component Analysis* (ICA), in which case we decompose the process into independent components and leave out the most "noise-like" component.

The authors advocating this method, so far have used only the mean square error criterion [10] to decide upon which model to trade with. However, it would be more important to look into the effect of ICA on the trading performance itself, because simply achieving low mean square error will not necessarily yield profitable trading [16].

As a result, the objective of this paper is to develop profitable trading algorithms by using NARX networks (*Nonlinear AutoRegressive network with eXogenous inputs*) together with the pre-processing algorithm, ICA, which is optimized for trading performance.

The rest of this paper is organized as follows:

- in section 2, the model and the notations are introduced;
- in section 3, the computational model is detailed;
- in section 4, a detailed performance analysis is given based on historical and generated data;
- in section 5, conclusions are drawn.

2 Model and notations

In this section we describe the underlying mathematical model for algorithmic trading and then we introduce the prediction architecture (NARX) together with Independent Component Analysis as a pre-processing method.

Financial time series are often modeled as *p-th order vector autoregressive* VAR(p) processes. Let us denote the asset price vector at time t by $\mathbf{s}(t) = \{s_1(t), \ldots, s_i(t), \ldots, s_N(t)\}$, where $s_i(t)$ is the the price of asset i at time t and N is the number of assets available. The return vector $\mathbf{r}(t) = \{r_1(t), \ldots, r_i(t), \ldots, r_N(t)\}$ is defined as

$$r_i(t) := \frac{s_i(t)}{s_i(t-1)} - 1,\ i = 1, \ldots, N. \tag{1}$$

We consider $\mathbf{r}(t)$ as a VAR(p) process

$$\mathbf{r}(t) = \mathbf{b} + \mathbf{A}_1\mathbf{r}(t-1) + \mathbf{A}_2\mathbf{r}(t-2) + \ldots + \mathbf{A}_p\mathbf{r}(t-p) + \mathbf{W}(t) \tag{2}$$

where $\mathbf{b}$ is a vector of dimension N and $\mathbf{W}(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ [23]. Based on the observation of $\mathbf{r}(t), t = 1, \ldots, T$, the model parameters (vector $\mathbf{b}$ and matrices $\mathbf{A}_i, i = 1, \ldots, p$ can be identified by the *Ordinary Least Squares* (OLS) method [22].

2.1 Mean reverting portfolios

Besides predicting a single asset value, we are also going to predict prices of a portfolio. Thus, we briefly summarize the theory of mean reverting portfolios.

Mean reversion is a good indicator of predictability, therefore identifying mean reverting portfolios has become a key research area [4, 15, 20]. However, portfolio optimization becomes very complex when introducing cardinality constraints in order to minimize the transaction costs. In this case, optimizing sparse portfolios is proven to be NP hard [8].

One can introduce a portfolio vector $\mathbf{x}^T = (x_1, ..., x_N)$, where component x_i denotes the amount of asset i held in the portfolio. In portfolio optimization we seek the optimal portfolio vector exhibiting mean reverting property under sparseness constraint, i.e. $card(\mathbf{x}) \leq L$ where $card$ denotes the number of non-zero components in the portfolio and L is a given positive integer $1 \leq L \leq N$.

The stochastic nature of a mean reverting portfolio is described by the so-called Ornstein–Uhlenbeck process [19]:

$$dp(t) = \lambda \left(\mu - p(t)\right) dt + \sigma dW(t), \tag{3}$$

where $W(t)$ is a Wiener process. By using the Itō-Doeblin formula [17], one can obtain the following solution:

$$p(t) = p(0)e^{-\lambda t} + \mu(1 - e^{-\lambda t}) + \sigma \int_0^t e^{-\lambda(t-s)} dW(s), \tag{4}$$

which implies that

$$\lim_{t \to \infty} p(t) \sim \mathcal{N}\left(\mu, \sqrt{\frac{\sigma^2}{2\lambda}}\right). \tag{5}$$

For trading, the mean reversion coefficient λ is the key parameter, as it determines how fast the process gets back to the mean, as well as inversely indicating the level of uncertainty around the mean. Larger λ is more suitable for convergence trading, as it quickly returns to the mean and it contains a minimum amount of uncertainty around the mean. In this case we assume, that the asset prices follow a stationary, first order, vector autoregressive VAR(1) process:

$$\mathbf{s}(t) = \mathbf{A}\mathbf{s}(t-1) + \mathbf{W}(t). \tag{6}$$

So far, our objective here is to maximize the mean reversion coefficient, which can be done as follows (see [20]):

$$\mathbf{x}_{opt} = \max_{\mathbf{x}} \frac{\mathbf{x}^{\mathrm{T}} \mathbf{A}\mathbf{G}\mathbf{A}^{\mathrm{T}} \mathbf{x}}{\mathbf{x}^{\mathrm{T}} \mathbf{G} \mathbf{x}}. \tag{7}$$

under the constraint $card(\mathbf{x}) \leq L$, where $\mathbf{G}$ is the stationary covariance matrix of $\mathbf{s}(t)$ and $\mathbf{A}$ is estimated by:

$$\hat{\mathbf{A}} = \left(\mathbf{s}^T(t-1)\mathbf{s}(t-1)\right)^+ \left(\mathbf{s}^T(t-1)\mathbf{s}(t)\right), \tag{8}$$

where $\mathbf{M}^+$ denotes the Moore-Penrose pseudo-inverse of matrix $\mathbf{M}$.

The brute force approach is constructing all $\frac{N!}{L!(N-L)!}$ L-dimensional submatrices of $\mathbf{G}$ and $\mathbf{AGA^T}$ and then solving all the corresponding eigenvalue problems to find the theoretical optimum. However, for large values of N and L, it is computationally infeasible, in our case, where $N = 29$ and $L = 3$ can be done real-time.

Once we have the optimal portfolio at hand, we do prediction based trading on it by using NARX network.

2.2 NARX neural networks

For predicting the future price of a portfolio and indicating the necessary trading action, we will use the NARX model which is widely used in economics [11, 16]. This network implements the following input-output mapping [13]:

$$y(t) = f\left(y(t-1), \ldots, y(t-p), \mathbf{r}(t-1), \ldots, \mathbf{r}(t-q), \mathbf{w}\right), \tag{9}$$

where $\mathbf{r}(t)$ is the vector valued time series (the return vector defined by equation (1)). In the model p and q are arbitrary parameters, which are now chosen as $p = q$ and the dimension of the output is taken to be one. Vector $\mathbf{w}$ represents the weights (free parameters) of the NARX network. The network is depicted by Fig. 1.

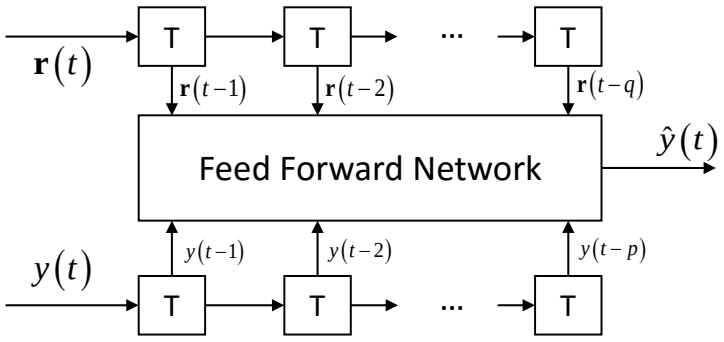

Fig. 1. NARX model

The output of the NARX network has three values corresponding to the actions of buying, selling, and position holding, respectively, given as follows:

$$y(t) = \begin{cases} 1 & \text{if } \frac{r(t)}{\bar{r}(t)} > \varepsilon \\ -1 & \text{if } \frac{r(t)}{\bar{r}(t)} < -\varepsilon \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where

$$\bar{r}(t) = \frac{1}{J} \sum_{j=1}^{J} |r(t-j)|, \tag{11}$$

and $r(t)$ is the return of either a single asset or a portfolio. Here ε is a predefined threshold by which one can control the reaction of the system (i.e. how large price changes will initiate actions). In this way, 1 at the output will refer to large price rises, while -1 at the output indicates sharp drops in price. In the case of small changes the output will yield zero.

Due to the autoregressive nature of the network (the output is determined by the past p output values of y and an independent input $\mathbf{r}$, which is now the return vector of the observed financial time series), after learning it is able to capture the characteristics of VAR(p) processes. The network then can provide trading decisions either on a single asset or on a portfolio.

Because of the simple structure of NARX model, it does not require complex training algorithm, therefore learning can be done by the Levenberg-Marquardt error back propagation procedure [9]:

$$L = \frac{1}{T} \sum_{t=1}^{T} \left(\hat{y}(t) - y(t)\right)^2, \tag{12}$$

where L is the loss function (mean square error) to minimize and $\hat{y}(t)$ is the estimation of future values of time series (either a single asset or a portfolio) at time instant t.

2.3 Application of ICA to filter the input noise

Since ICA has the capability of decomposing a random time series into independent random processes, we use this property to represent the financial time series under investigation. If one tries to predict the future values of the time series based on the decomposed representation when leaving out the noise term, then this prediction may perform much better than the one carried out on the original representation. This is due to the fact that the most "noise-like" term causing the largest amount of randomness have been filtered out. As a result, by leaving out the noise we can eliminate the tendency to overfitting which will increase the quality of prediction.

In the case of ICA we seek N independent random variables $\mathbf{S} = [\mathbf{S}_1, \mathbf{S}_2 ... \mathbf{S}_N]$ which represent the so-called sources. Let $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2 ... \mathbf{X}_M]$ be the observed time series, which may come from the S&P 500 series. For the sake of brevity let

$M = N$ (the number of observations is the same as the number of dimensions). Performing ICA, the observations are expressed as the linear combination of the sources $\mathbf{S}$, where $\mathbf{A}$ is the so-called mixing matrix:

$$\mathbf{X} = \mathbf{AS} = \sum_{i=1}^{N} a_i s_i. \tag{13}$$

The objective is to find the best de-mixing matrix $\mathbf{W}$ which results in that the components of $\mathbf{Y} = \mathbf{WX}$ are maximally independent from each other. Based on the work of [1], let $\mathbf{w}^T$ be a row vector of the reconstruction matrix. Then the estimation of an independent component is given as

$$y = \mathbf{w}^T\mathbf{X} = \mathbf{w}^T\mathbf{AS} = \mathbf{z}^T\mathbf{S}. \tag{14}$$

If the components of S_i are independent then adjusting the elements of $\mathbf{z}^T$, the sum will be less similar to a Gaussian random variable. In order to obtain the independent components, one must maximize the nongaussianity of $\mathbf{w}^T\mathbf{X}$, which can be measured by the negentropy [1].

For trading we use the standard FastICA package for MATLAB [3]. This algorithm preprocesses the data for the NARX network.

In the related literature, the *Relative Hamming Distance* (RHD) based error is used [27] in order to determine and leave out the "noise-like" components, defined as follows:

$$RHD\,(x, y) = \frac{1}{T-1}\sum_{t=1}^{T-1}\big(R_x\,(t) - R_y\,(t)\big)^2, \tag{15}$$

where $R_x\,(t) = sign\big(x\,(t+1) - x\,(t)\big)$ and $R_y\,(t) = sign\big(y\,(t+1) - y\,(t)\big)$.

The task is to find those components which minimize the RHD error between the original and reconstructed time series. In the case of high dimension, the solution cannot be found by exhaustive search, thus the so-called *Testing and Acceptance* (TnA) heuristic is used [27]. Since we only want to leave out one (the most noise-like) component, this amounts to running TnA for a single step. In this way, our procedure is faster than the standard TnA, however, it will yield a suboptimal solution.

3 Trading by NARX networks combined with ICA

The structure of the proposed trading system is given by the following block diagram (Fig. 2) and detailed as follows:

- First we determine the optimal portfolio that maximizes the mean reversion coefficient λ (see equation 7). In case of single asset trading, this step is skipped.
- The FastICA algorithm determines the independent sources from which the most "noise-like" component is identified by calculating the RHD reconstruction error (see section 2.3).

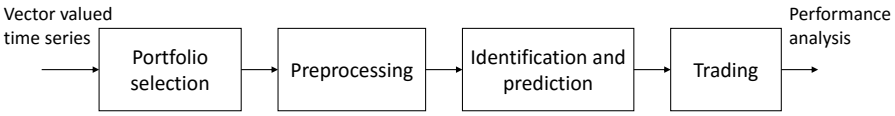

Fig. 2. Computational model

- After discarding the most "noise-like" component, the system is trained. At the input of NARX network there are the returns of the available financial time series $\mathbf{r}$ and the feedbacks from the earlier outputs y. In order to prevent over-learning, we use early stopping [5]. For the sake of comparison, there is also a possibility to use PCA instead of ICA, or to skip preprocessing and training the system with raw data.
- Based on the prediction of NARX network, one can form a trading signal: If the output exceeds a given threshold then the position becomes opened. The same holds for the case when the output falls below another given threshold, but then a short position is taken. This operation is depicted by Fig. 3.

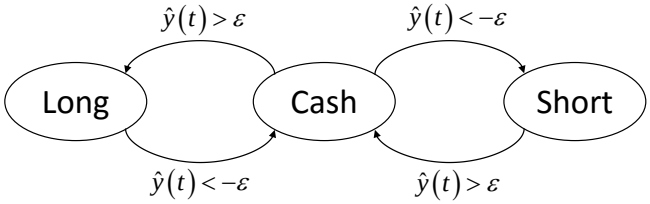

Fig. 3. Trading strategy

In case of portfolio trading, after closing the position (sell or buy back), we recalculate the optimal portfolio and retrain NARX network.

Finally, we compare the profitability of the different methods by testing and evaluating various numerical indicators (see section 4).

4 Numerical results

An extensive back-testing framework was created to handle trading actions on various input data sets and provide numerical results for the sake of comparing the performance of methods.

The tests have been parametrized as follows:

- in the case of using raw data, the inputs of the network are the returns of the available assets;
- in the case of PCA we selected the first few principal components, which the sum of the variances exceed 95% of the total variance of the original data;
- in the case of ICA the input data was first filtered according to the method described in section 2.3.

In each simulation we have used a 3-layer network containing 20 neurons in the hidden layer and the number of learning samples has been 500. For the sake of comparison, we have also implemented the Buy-and-Hold strategy.

The following performance measures were calculated for each experiments on the corresponding time series (either ETF or generated):

- profit: $G = \frac{c_T}{c_0} - 1$;
- maximum drawdown, that is the maximum loss from a peak to a trough of a portfolio [2].

Here c_t denotes the sum of owned cash and the market value of the owned portfolio at time instance t, while c_0 denotes the initial cash (in each case the agent started with \$10,000).

In this section, we discuss the numerical results obtained on the following data sets:

- generated time series (VAR(p));
- Exchange Traded Funds: EEM, EFA, EWA, EWC, EWJ, EWZ, FAS, FAZ, FXI, GDX, GLD, IGE, IWM, IYR, QID, QQQ, SDS, SKF, SPY, SSO, TZA, UNG, USO, VWO, VXX, XLE, XLF, XLI, XRT from the year of 2013 to 2015 in daily resolution [26].

The generated data have been constructed by using the 2013-2014 ETF time series. We have modeled the vector valued time series by a VAR(3) process and with the optimized parameters we simulate the process for 5000 steps (this roughly amounts daily data for 20 years). We added a 0 mean Gaussian noise to simulated N dimensional process and then multiply it with a randomly selected mixing matrix $\mathbf{A}$. The output then will be $N + 1$ dimensional. This process is illustrated by Fig. 4.

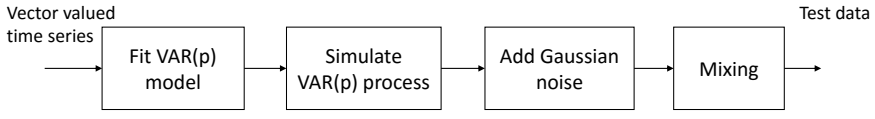

Fig. 4. Test data generation

Based on this method, we have generated a 6-dimensional 5000-tick long data series. First, the efficiency of the Buy-and-Hold strategy was investigated for each data series, separately, shown by Table 7. After this, the performance of ICA and PCA based methods have been evaluated and the results are shown in Table 3. In the case of mean reverting trading, the agent could trade with any asset within the confines of the cardinality constraints. The profitability has been investigated in the cases of $L = 1, 2, 3, 4$ and the results are shown by Table 4.

In the case of ETF data we have run both the Buy-and-Hold and our proposed methods on all the 29 data series and we have also compared the performance PCA and ICA. The results are shown by tables 8 and 5, respectively. Similarly to the analysis of generated data, in the case of mean reverting trading, the agent could trade with any asset within the confines of the cardinality constraints. The profitability has been investigated in the cases of $L = 1, 2, 3, 4$ and the results are shown by Table 6.

In each simulation bid-ask spread of 0.01$ has been taken into account.

In the tables below, the average results of our own method has been depicted.

Simulation	**Profit**	**Drawdown**
Single asset, Buy-and-Hold	94.88 %	46.97 %
Single asset, raw	825.36 %	48.21 %
Single asset, PCA	83.75 %	61.71 %
Single asset, ICA	1665.83 %	40.28 %
Portfolio, raw	544.47 %	40.68 %
Portfolio, PCA	34.86 %	58.13 %
Portfolio, ICA	536.88 %	32.6 %

Table 1. Average trading performance on generated data

On Table 1 one can see the results of 4500 trading days (note, the first 500 was used for learning). As demonstrated, in almost each case we managed to secure positive profit. In the case of generated data, the smallest profit (83.75 %) was achieved by using PCA based preprocessing, while the highest (1665.83 %) was secured by ICA based preprocessing. This is not surprising taking into account how the dataset has been generated (see section 2.3). Trading with mean reverting portfolios on the generated data set did not considerably increased the profit, but it has decreased the drawdown.

Simulation	Profit	Drawdown
Single asset, Buy-and-Hold	-2.13 %	42.95 %
Single asset, raw	13.89 %	11.89 %
Single asset, PCA	9.85 %	7.64 %
Single asset, ICA	27.89 %	11.69 %
Portfolio, raw	10.6 %	7.62 %
Portfolio, PCA	16.6 %	9.66 %
Portfolio, ICA	18.77 %	8.31 %

Table 2. Average trading performance on daily ETF data

Table 2 indicates real test running on ETF data obtained at daily tick time. PCA has performed better on the real data and it got near to the performance of ICA. Both preprocessing yielded better trading performance than trading on raw data. It is clear that in the single asset case ICA achieved better profit (27.89 %) but it increased the maximum drawdown. Trading with mean reverting portfolios resulted in lower profit than trading with a single asset, but the drawdown became lower.

These are indeed good results, but in this case trading was only done on 250 trading days.

5 Conclusions

In this paper we have introduced NARX network combined with ICA in order to increase the accuracy of prediction of financial time series. Filtering out the "noise-like" components from the input data helps avoid overfitting. Instead of mean square error criterion, we optimized ICA based on the trading performance.

The method has been tested not only on single asset time series, but on predicting the values of mean reverting portfolios, as well. The performance analysis on generated data and real data has proven that profit can be gained by the new methods even in the presence of bid-ask spread.

We have demonstrated that PCA does not perform well on generated data sets, but in real data it helps avoid the problem of over learning. Applying ICA can yield higher profit and it can also decrease the drawdown in the case of trading with mean reverting portfolios. It is important to mention that preprocessing always improves the performance as opposed to trading on raw data.

Appendix

In the tables below, we give the results in details.

	Raw		PCA		ICA	
Asset	**Profit**	**Drawdown**	**Profit**	**Drawdown**	**Profit**	**Drawdown**
GEN1	259.24 %	63.69 %	-44.86 %	63.81 %	4624.19 %	31.18 %
GEN2	1900.73 %	37.75 %	438.03 %	51.77 %	824.05 %	44.47 %
GEN3	1207.89 %	42.71 %	191.16 %	44.08 %	469.61 %	43.90 %
GEN4	724.89 %	44.65 %	0.93 %	65.73 %	2980.76 %	37.97 %
GEN5	681.22 %	39.11 %	-0.10 %	54.48 %	414.12 %	51.52 %
GEN6	178.21 %	61.33 %	-82.65 %	90.40 %	682.25 %	32.62 %
Average	825.36 %	48.21 %	83.75 %	61.71 %	1665.83 %	40.28 %
Standard deviation	587.78	10.38	180.36	14.75	1589.27	7.11

Table 3. Trading performance of our strategy on generated data

	Raw		PCA		ICA	
	Profit	**Drawdown**	**Profit**	**Drawdown**	**Profit**	**Drawdown**
L=1	1391.63 %	25.53 %	140.69 %	62.02 %	492.28 %	32.13 %
L=2	52.80 %	51.31 %	18.16 %	63.60 %	792.12 %	24.37 %
L=3	183.23 %	44.19 %	40.10 %	39.69 %	280.74 %	44.23 %
L=4	550.23 %	41.68 %	-59.52 %	67.19 %	582.38 %	29.67 %
Average	544.47 %	40.68 %	34.86 %	58.13 %	536.88 %	32.60 %
Standard deviation	522.00	9.43	71.44	10.81	183.59	7.28

Table 4. Trading performance of our strategy using mean reverting portfolios on generated data

	Raw		PCA		ICA	
Asset	**Profit**	**Drawdown**	**Profit**	**Drawdown**	**Profit**	**Drawdown**
SPY	5.06 %	9.61 %	3.04 %	3.66 %	19.31 %	8.75 %
XLF	28.19 %	1.90 %	5.00 %	1.25 %	2.36 %	7.67 %
QQQ	-11.91 %	16.22 %	5.11 %	8.96 %	6.12 %	12.01 %
EEM	7.70 %	9.28 %	6.87 %	1.88 %	17.19 %	10.85 %
IWM	2.71 %	4.91 %	-0.58 %	5.10 %	24.68 %	4.55 %
FAS	29.14 %	2.31 %	38.84 %	11.19 %	55.02 %	23.92 %
FAZ	63.07 %	19.41 %	4.75 %	9.53 %	30.11 %	14.03 %
SDS	30.72 %	6.07 %	2.64 %	3.21 %	23.48 %	13.21 %
TZA	1.58 %	32.41 %	35.42 %	4.41 %	76.96 %	22.60 %
FXI	-7.16 %	17.01 %	15.73 %	2.13 %	3.45 %	12.88 %
UNG	85.35 %	14.17 %	46.68 %	13.09 %	101.57 %	11.46 %
EFA	10.31 %	14.64 %	8.58 %	3.02 %	13.17 %	6.45 %
SSO	6.91 %	17.72 %	3.26 %	9.89 %	-5.84 %	13.93 %
EWJ	-6.40 %	10.31 %	-9.12 %	11.16 %	10.31 %	12.33 %
XLE	7.12 %	5.40 %	0.70 %	3.74 %	11.62 %	9.26 %
EWZ	0.18 %	4.40 %	-0.09 %	12.98 %	2.92 %	10.50 %
QID	2.28 %	18.05 %	4.80 %	22.85 %	37.20 %	8.85 %
VXX	133.24 %	16.00 %	62.38 %	17.75 %	332.88 %	6.27 %
IYR	2.53 %	4.53 %	8.95 %	6.56 %	9.37 %	3.62 %
XLI	-8.19 %	14.05 %	4.53 %	5.02 %	9.99 %	9.11 %
GLD	5.19 %	7.83 %	11.58 %	0.24 %	5.51 %	6.40 %
GDX	8.84 %	18.86 %	-8.14 %	8.14 %	-16.81 %	24.68 %
SKF	15.66 %	3.37 %	14.68 %	21.95 %	35.42 %	11.18 %
XRT	-4.89 %	10.35 %	-0.36 %	6.92 %	12.04 %	5.88 %
USO	3.21 %	12.19 %	5.51 %	5.69 %	-18.05 %	21.14 %
VWO	2.28 %	12.19 %	13.07 %	2.33 %	-4.92 %	22.58 %
EWA	-12.48 %	23.56 %	-10.79 %	13.76 %	11.91 %	9.84 %
EWC	1.70 %	6.69 %	3.74 %	4.55 %	-1.71 %	7.66 %
IGE	0.96 %	11.37 %	8.81 %	0.56 %	3.53 %	7.53 %
Average	13.89 %	11.89 %	9.85 %	7.64 %	27.89 %	11.69 %
Standard deviation	30.61	6.88	16.11	5.92	62.85	5.81

Table 5. Trading performance of our strategy on daily ETF data

	Raw		PCA		ICA	
Sparsity	**Profit**	**Drawdown**	**Profit**	**Drawdown**	**Profit**	**Drawdown**
L=1	3.74 %	9.05 %	30.01 %	4.55 %	3.32 %	11.16 %
L=2	4.99 %	9.64 %	34.03 %	8.57 %	17.90 %	1.87 %
L=3	25.50 %	5.90 %	10.96 %	6.75 %	23.37 %	11.47 %
L=4	8.19 %	5.89 %	-8.58 %	18.79 %	30.48 %	8.76 %
Average	10.60 %	7.62 %	16.60 %	9.66 %	18.77 %	8.31 %
Standard deviation	8.75	1.74	16.95	5.46	9.97	3.86

Table 6. Trading performance of our strategy using mean reverting portfolios on daily ETF data

	Buy-and-Hold	
Asset	**Profit**	**Drawdown**
GEN1	12.42 %	51.38 %
GEN2	177.17 %	43.10 %
GEN3	42.69 %	50.13 %
GEN4	131.96 %	43.91 %
GEN5	183.66 %	43.30 %
GEN6	21.40 %	50.02 %
Average	94.88 %	46.97 %
Standard deviation	71.82	3.57

Table 7. Trading performance of the Buy-and-Hold strategy on generated data

	Buy-and-Hold	
Asset	**Profit**	**Drawdown**
SPY	33.72 %	18.61 %
XLF	16.46 %	33.70 %
QQQ	44.45 %	16.09 %
EEM	9.45 %	30.87 %
IWM	38.15 %	28.91 %
FAS	-7.37 %	76.21 %
FAZ	-83.61 %	86.53 %
SDS	-60.11 %	65.70 %
TZA	-90.08 %	92.16 %
FXI	-0.33 %	36.64 %
UNG	-77.80 %	83.55 %
EFA	10.07 %	25.85 %
SSO	55.88 %	36.23 %
EWJ	2.43 %	23.96 %
XLE	27.29 %	29.23 %
EWZ	-21.21 %	35.82 %
QID	-67.96 %	75.59 %
VXX	-93.92 %	94.84 %
IYR	58.27 %	22.58 %
XLI	42.04 %	25.83 %
GLD	47.56 %	19.03 %
GDX	-0.81 %	40.78 %
SKF	-64.23 %	68.76 %
XRT	79.99 %	22.21 %
USO	-17.13 %	34.75 %
VWO	11.88 %	30.58 %
EWA	21.78 %	30.38 %
EWC	11.21 %	28.47 %
IGE	12.28 %	31.70 %
Average	-2.13 %	42.95 %
Standard deviation	47.93	24.33

Table 8. Trading performance of the Buy-and-Hold strategy on daily ETF data

References

1. A. Hyvärinen and E. Oja: Independent Component Analysis: Algorithms and Applications. Neural Networks, 13(4-5):411-430, 2000.
2. Alexei Chekhlov, Stanislav Uryasev, Michael Zabarankin: Drawdown measure in portfolio optimization. International Journal of Theoretical and Applied Finance, vol.8, no.1, pp.13-58, 2005.
3. Aalto University, Department of Computer Science: The FastICA package for MATLAB, Online available: http://research.ics.aalto.fi/ica/fastica/. 2015.
4. Alexander D'Aspremont: Identifying small mean-reverting portfolios. Quantitative Finance, 2011 11:3, pp. 351-364.
5. Alpaydin Ethem: Introduction to Machine Learning. Massachusetts Institute of Technology. 2010.
6. Andrew D. Back, Andreas Weigend: A first application of independent component analysis to extracting structure from stock returns. International Journal of Neural Systems, 1997 Aug, 8(4):473-84.
7. Andrew D. Back, Andreas Weigend: Discovering Structure in Finance Using Independent Component Analysis. Decision Technologies for Computational Finance, Volume 2 of the series Advances in Computational Management Science pp 309-322, 1998.
8. B.K. Natarajan: Sparse approximate solutions to linear systems. SIAM J. Comput., 24(2), pp. 227-234, 1995.
9. Bogdan M. Wilamowski, J. David Irwin: Intelligent Systems. CRC Press. 2011.
10. Chi-Jie Lu, Tian-Shyug Lee, Chih-Chou Chiu: Financial time series forecasting using independent component analysis and support vector regression. Decision Support Systems, Volume 47, Issue, May 2009, Pages 115-125.
11. Emad W. Saad, Danil V. Prokhorov, Donald C. Wunsch: Comparative Study of Stock Trend Prediction Using Time Delay, Recurrent and Probabilistic Neural Networks. IEEE Transactions on Neural Networks, vol.9, no.6, pp.1456-1470, Nov 1998
12. Ernest P. Chan: Algorithmic Trading: Winning Strategies and Their Rationale. Wiley Trading Series, 2013.
13. Gérard Dreyfus: Neural Networks: Methodology and Applications. Springer, 2005.
14. Harry Markowitz: Portfolio Selection. The Journal of Finance, 1952 Vol. 7 (1), pp 77–91.
15. I. Robert Sipos, Janos Levendovszky: Optimizing sparse mean reverting portfolios. Algorithmic Finance, 2013, 2.2: 127-139.
16. Iebeling Kaastra, Milton Boyd: Designing a neural network for forecasting financial and economic time series. Neurocomputing 10, pp.215-236, 1996.
17. Kiyosi Itō: Stochastic integral. Proceedings of the Imperial Academy 20 (8): 519–524, 1944.
18. Kyoung-jae Kim: Financial time series forecasting using support vector machines. Neurocomputing 55 (2003) pp 307–319.
19. L. S. Ornstein, G. E. Uhlenbeck. On the Theory of the Brownian Motion. Physical Review, 36(5), p. 823 (1930).
20. Norbert Fogarasi, Janos Levendovszky: Sparse, mean reverting portfolio selection using simulated annealing. Algorithmic Finance, 2013, 2.3: 197-211.
21. P. N. Kumar, G. Rahul Seshadri, A. Hariharan, V. P. Mohandas, P. Balasubramanian: Financial Market Prediction Using Feed Forward Neural Network. Technology Systems and Management, Volume 145 of the series Communications in Computer and Information Science pp 77-84, 2011.

22. Richard A. Johnson, Dean W. Wichern: Applied Multivariate Statistical Analysis. Pearson Education Inc. 2007.
23. Ruey S. Tsay: Analysis of Financial Time Series. Wiley-Interscience. 2005.
24. Salim Lahmiri: A Comparison of PNN and SVM for Stock Market Trend Prediction using Economic and Technical Information. International Journal of Computer Applications Volume 29 No.3, September, 2011.
25. Tony Van Gestel, Johan A. K. Suykens, Dirk-Emma Baestaens, Annemie Lambrechts, Gert Lanckriet, Bruno Vandaele, Bart De Moor, and Joos Vandewalle: Financial Time Series Prediction Using Least Squares Support Vector Machines Within the Evidence Framework. IEEE Transactions on Neural Networks, Vol. 12, No. 4, July 2001.
26. Yahoo Finance: Online available at https://finance.yahoo.com. 2015.
27. Yiu-ming Cheung, Lei Xu: Independent component ordering in ICA time series analysis. Neurocomputing 41 (2001) pp. 145-152