Predicting future prices in an Artificial Stock Market using an Artificial Neural Network

Alberto Colliva, Andrea Nocifora, Antonio Rosanova

1 Introduction and discussion of the model

We present a model of a stock market with agents trading two types of stocks (preferably with different volatility features), characterized by two different levels of risk aversion and by different decision processes.
The main class chooses which is the best decision (buying or selingl) according to the forecast the agents make on the stocks prices in the following day. This skill of theirs simulate the result of an algorithm (who is not directly explicated) which is more or less efficient in predicting the prices. There is a short-sale constraint: agents cannot borrow and are liquidity constrained since they cannot buy any stock if the price exceeds their current cash holdings.
Then the next step concerns the decision on whether the purchase or sale shall actually take place, depending on the forecasted gain or loss on the stock price and the level of risk aversion (since they dislike volatility on the stock price). The other class, made up by the so-called *dumb agents* has no forecasting skill and just makes its decision randomly.
Our model also include a special kind of agent, which is endowed with a (supposedly) major forecasting skill, thanks to the ability to exploit an Artificial Neural Network (ANN) to predict the future stock price. The ANN has been trained on the time series of price the agents are looking at and makes predictions based on an information set including the closing prices of the two previous days. The input and output values have been rescaled in the interval [0,1] for the training process, since the mapping function used is a hyperbolic tangent, which accepts value in this interval.
The main issue is that of trying and identify different results and performances due to the heterogeneity among the agents involved in the stock market. Notice that no exchange is supposed to take place among agents in our model: they just buy and sell stocks having at disposal an external pool, with no constraints given by supply-demand dynamics. However, we included the possibility for the user to introduce an alternative exchange mechanism by turning on a switch in the interface; notice that even in such a case, no price formation mechanism exists. Therefore, the stock prices do not change as a result of the action of our agents: think of them as a very small part of the whole market, who does not have any influence on its price dynamics. It is worth underlying that the series of price used in our simulation are generated using the statistical software *R* (although using actual time series of prices would not significantly change the results obtained). Our choice will be made clearer lately; for the moment, it's enough to know that this allowed us to choose the underlying stochastic process behind the time series. Starting from this, we will make use of our econometric knowledge, methods and tools to analyze the results, particularly those regarding the use of the ANN for forecasting purposes.

A further dynamics included in our model is that of the imitation: unsuccessful agents (defined as those who lost a given percentage of their initial wealth) are given the ability (which is not necessarily performance-improving) to follow the same strategy as the wealthiest agent's (in the previous turn). Moreover, each turn, with a 0.2% probability, agents will receive information regarding a prediction of market crashes or booms, inducing a reaction by them: this allows us to study their behavioral changes (e.g. purchase of a bigger amount of stocks if they believe in the boom forecast; sale of a bigger amount of stocks if they believe in the crash forecast), which take place even if their information are strongly biased, since the price series is already there and nothing would affect it.

2 Code Discussion

2.1 R

The model simulates the behavior of three different types of agents in a market where prices follow a time series which has been previously generated using *R* and then written in a text file. The procedure is quite simple and involves first of all the generation of a *white noise*, whose distribution is normal with mean 0 and standard deviation fixed differently depending on the volatility that we are trying to model (std. dev.=1 very volatile; std. dev.= 0.5 quite volatile; std. dev.=0.1 stable around the initial price).

The white noise is then inserted in the stochastic structure that we chose: in our experiments we used an AR(2) with coefficients 0.7 and 0.2 when we wanted to model stock prices that were stationary around their mean and random walks with different levels of standard deviations in the white noise when we wanted to model a stochastic trend in the time series. Here is the*R* code for generating a white noise and an AR(2) process:

```
e1<-rnorm(1000,sd=0.5)
ar2<-rep(10,1000)
for (i in 2:1000) ar2[i] <- 0.7*ar2[i-1]+ 0.2*ar2[i-2]+ e1[i]
ar2<-ts(ar2)
```

In the case of a random walk:

```
>rw<-rep(10,1000)
>for (i in 2:1000) arw[i] <- rw[i-1]+ e1[i]
>rw<-ts(rw)
>ts.plot(rw)    [plotting the time series]
```

2.2 ANN with Python

We included in our model the possibility of inserting an agent with the capability of using an ANN to forecast future prices (at a one-day and at a two-day horizon). The files creating and using the ANN are written in *Python* and generate a time series containing the prediction of the agents, that she will use for the decision process.
Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs or to find patterns in data. The network is divided into layers. The input layer consists of just the inputs to the network. Then follows a hidden layer, which consists of any number of neurons,

or hidden units placed in parallel. Each neuron performs a weighted summation, accordingly to the weight matrix, of the inputs, which then passes a non-linear activation function, also called the sigmoid function.
The whole operation is:

$$v_o = f(f(v_i \cdot Wi) \cdot Wo) \tag{1}$$

where Wi and Wo are the ANN's matrices and $f(x)$ is the sigmoid function.
The time series taken into account is read and processed by a Python-based software, with the objective of creating a list that could be properly read by the library *bpnn.py* (made available online for free by N. Schemenauer), which makes use of a *Multi-Layer Perceptron* algorithm. The requested form for the training process is a list containing two vectors: one with length n for the inputs and one with unit length for the output. The input and output data have been rescaled in the interval [0,1], where the minimum value of the time series is associated to the lower bound 0 and the maximum one corresponds to the upper bound 1.

```
pat=[]
for i in range(len(v)-n):
    pat.append([[0]*n,[0]])

#filling the pattern structure in the form
# of [[in1,in2,...,in_n],[out]]

for i in range(len(v)-n):
    vvv=[]
    for j in range(n):
        vvv.append((float(v[i+j])-minimum)/(maximum-minimum))
    pat[i][0]=vvv
    pat[i][1]=[(float(v[i+n])-minimum)/(maximum-minimum)]

# create a network with n input, n hidden, and one output nodes
nn = bpnn.NN(n, n, 1)
nn.train(pat)
```

The above code section trains the network to make forecasts for the next day (tomorrow), on the basis of an information set including the realized prices of the two previous days (yesterday and today). It is possible (and we actually tried to do so) to train the network in order to obtain forecasts for a period slightly ahead in the future (the day after tomorrow) on the basis of the known values of today and yesterday. The library *bpnn.py* has been slightly modified in order to have the function "test" of the NN class returning the forecasts list.

```
class NN:
...
   def test(self, patterns):
        list=[]
        for p in patterns:
            print(p[0], '->', self.update(p[0]))
            list.append(self.update(p[0]))
        return list
```

Apart from the ANN agent there are two other kinds of agents: the *dumbAgents* (who display a completely random behavior) and the *Agents* (who, depending on their degree of intelligence, obtain an estimate of the price for the following *Tick*).

2.3 NetLogo

The NetLogo-based software loads the two time series in lists. At each Tick two elements from each list are taken into consideration: that of the current Tick (*price*-tod*) and and that of the next Tick (*price*-tmw*). The latter is the following day quote and is used by agents in their prediction process, being part of the determinant of their decision.

Price forecast

The forecasted price is set starting from the actual quote for the next day; a certain value drawn from an uniform distribution between 0 and the set degree of intelligence (*degree_intelligence*) is going to be added or subtracted (randomly, with a 1/2 probability) to the actual price. Therefore, a higher value for the slider *degree_intelligence* leads to a higher probability of having a remarkable forecast bias with respect to the actual price.

```
let randValue 0
    ifelse random 2 = 0[set randValue random-float intelligence]
                       [set randValue random-float ( - intelligence)]
    set forecasting-price1 price1-tmw + randValue
```

Decision process

The forecasted price shall determine the following action. The program includes a decision function which takes as arguments the forecasted price, the current one and a natural number (either 1 or 2). The need for this quite artificial mechanism comes from having two types of stocks. The function discriminates between buying and selling the stock indicated as argument (1 or 2) depending on the difference between the forecasted value and the current one being positive or negative. Then, the next step leads to a sub-section where the agent decides between buying or passing and selling or passing. The decision is determined by the *decision-value* being higher or lower than a given threshold. The next section is about setting the latter. The *decision-value* has also an important meaning since basically tells us that the agents are some sort of mean-variance optimizers (in the Markovitz sense): they like a high expected return, while they dislike volatility (at different levels, depending on their risk aversion). Whenever an agent decides to make one of the two transactions, the related agent variable (buyer1 buyer2 seller1 seller2) is set to be '*true* by means of the functions *buyer* and *seller* (which takes 1 or 2 as argument depending on which stock is involved).

```
to decision[forecastingPrice priceTOD stock]
  let diff forecastingPrice - priceTOD
  ifelse diff > 0  [
   set decision-value 1000 * (diff / priceTOD) / riskAversion
   if decision-value > threshold stock [buyer stock]
   ]
                   [
   set decision-value 10 * (- diff / priceTOD) * riskAversion
   if decision-value > threshold stock [seller stock]
   ]
end
```

Setting the threshold

As anticipated in the last section, the threshold value plays an important role in the

agents' decision mechanism. Such a threshold has no fixed value (except for the starting), but a variable one, depending on the percentage of agents transacting on each stock (the threshold function takes as arguments an integer number, 1 or 2, and discriminates between the threshold value of the two stocks). In the interface it possible to decide over which percentage of transacting agents the threshold shall be adjusted. This trick is used also to exogenously determine the appeal of the security. The function *update-threshold* sets the threshold value at each Tick with the simple criterion of increasing or decreasing it by a given percentage, depending on the amount of agents actually transacting in that Tick being above or below the value decided by the user.

```
to update-threshold
   let n1 count agents with [buyer1 = true and
                             out-of-market = false]
        + count agents with [seller1 = true and
                             out-of-market = false]
  let n2 count agents with [buyer2 = true and
                             out-of-market = false]
        + count agents with [seller2 = true
                          and out-of-market = false]
  let tot count agents with [out-of-market = false]

  if tot > 0[
  ifelse (n1 * 100 ) / tot > %_transacting1
      [set threshold1 threshold1 * 1.05 ]
      [set threshold1 threshold1 * 0.95]

  ifelse (n2 * 100 ) / tot > %_transacting2
      [set threshold2 threshold2 * 1.05 ]
      [set threshold2 threshold2 * 0.95]
  ]
end
```

NN agent mechanism
She follows a process analogous to the one just described, hence only the differences shall be reported:

- it is not endowed with a forecasting function; the forecasted price is the one corresponding to the current Tick taken from the ANN output (loaded from a file into a list which is similar to the actual prices list);

- in the function *decisionNN* there is no *decision-value*, hence a random number drawn from a uniform distribution between 1 and 100 is directly compared with the percentage value of transacting agents set by the user; thus, there is no need for an *update-threshold* function for the NN agent.

dumbAgents mechanism
They decide in a completely random fashion whether to buy, sell or pass. An integer random number between 0 and 2 is drawn, in order to differentiate the three cases. The *pick_stock* function takes as arguments due integers: the first one determines the purchase (if 1) or sale (if 2), while the second one discriminates between the two kinds of stocks. Notice that the actual transactions (determined by the buy and sell functions)

occur altogether in the *transact* function. The mechanisms described thus far have the only purpose to set as *true* the "transaction intention" variables (*buyer1 buyer2 seller1 seller2*).

Exit from the Market
The agents exit the market under two simple terminal conditions:

- their wealth is decreased by a certain percentage fixed by the user in the interface through a slider (high losses lead to a "strategic withdrawal");

- their wealth is n (with n=3 in the code) times their initial one (high gains are an incentive to exit the market and spend a very comfortable retirement!) Whenever one of these conditions is satisfied the variable out-of-market is set to be *true* and the agent stops doing any kind of action.

Imitation
In the interface the user can set the level of lost wealth above which the agents start mimicking the actions. The agent shall stop imitating only if she will get back to her initial level of wealth (i.e. *initialCash*). At each Tick the wealthiest agent is identified and its transaction variables (buyer1 buyer2 etc.) are inserted in an *action-list* that is going to be used by the mimicking agents at the next tick for decisional purposes.

Unlikely events
The user interface contains a switch that, if turned on, simulates the reaction of the agents to the reception of information concerning prediction of market booms and crashes. Randomly (with 1/2 probability) one of two possible events (boom or crash) is chosen. As a matter of fact, this determines the purchase (in case of forecasted boom) or sale (in case of forecasted crash) of all risky securities, with a certain probability, decided by the user through the slider *prob_follow*. Notice that the dumb agents have been set as more likely to believe the information acquired. Here is reported the code concerning one of the possible events.

```
to stock1Boom

 output-type "Tick: " output-type ticks
    output-print " -> FORECASTED STOCK BOOM"


 ask agents[
   if out-of-market = false [
   if random 100 < prob_follow
      [while [cash > price1-tod][buy 1] ]
   ]
]
 ask dumbAgents[
   if out-of-market = false [
 if random 100 < prob_follow + 20
    [while [cash > price1-tod][buy 1] ]
      ]
]
end
```

Possible exchange mechanism
The software presents an alternative mechanism for the representation of exchange dynamics of the stocks among agents. In this case, the possibility that demand might exceed supply (and viceversa) is introduced. In order to simulate this mechanism two lists containing the name (*who*) of the agents are created; the lists are matched with each other and emptied couple by couple until one of the two is completely empty. The remaining agents in one of the two lists cannot proceed with the transactions.

```
to bid-ask
  let logB []
  let logS []

  ask turtles[
    if buyer1 = true and cash > price1-tod[

      set logB lput who logB
      ]
    if seller1 = true and stock1 > 0 [

      set logS lput who logS]
        ]

  while[ logS != [] and logB != []][
    ask turtle (item 0 logB)[buy 1
      set buyer1 false]
      ask turtle (item 0 logS) [sell 1
        set seller1 false]
      set logB remove-item 0 logB
      set logS remove-item 0 logS
      ]
end
```

3 Results

3.1 ANN

As already stated in the code discussion section above, two methods have been used for obtaining the predictions of the ANN. Having as inputs (i.e. the information set that the agent use in order to make her forecasts) the realized values of yesterday and today's price, we tried to obtain predictions for the price 1 day and 2 days ahead in the future (i.e. the prices for tomorrow and the day after tomorrow). Notice that these are two separated experiments, in order to analyze the ability of the agent to predict slightly farther in the future, so in each experiment she actually knows only either tomorrow or the day after tomorrow's price.
The 1 day ahead forecast provides good quality predictions: as we can see from figure 1, the agent is able to predict future prices with a small forecasting error. Nevertheless there are differences depending on the kind of time series used. As stated above, we actually decided and selected which type of stochastic process generating the time series had been used. This allowed us, beyond analyzing the quality of the predictions provided by the ANN, to compare our results with the intuitions provided by time series econometrics theory.

We report that our results nicely reconcile with the theory. We used a stationary AR(2) process and two types of random walks, each with different degrees of volatility. Econometric theory states that a stationary process should be easier to forecast, while the independent increments of a random walk process determine the difficulty in correctly predicting future values, due to the completely stochastic trend driving the process. This is seen even more clearly if we try to forecast values at periods farther in the future. In our experiments we exploited the ANN to check these differences, which

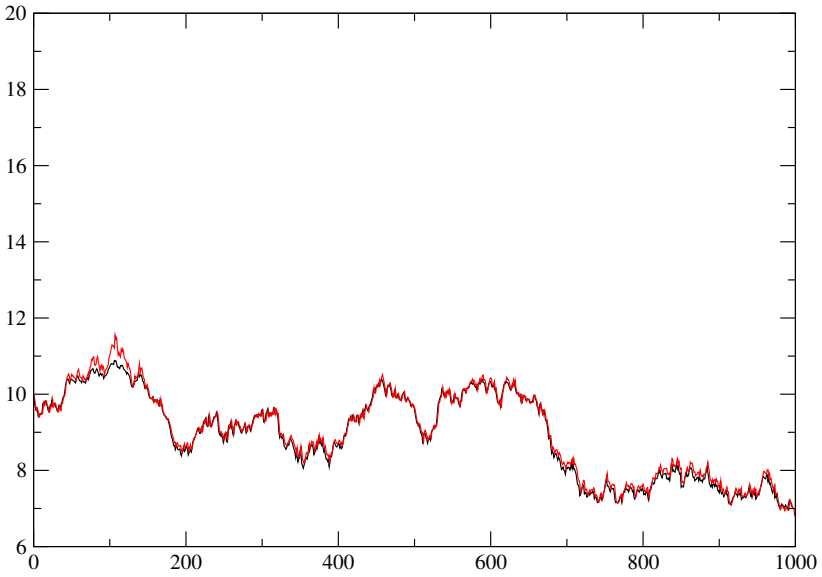


Figure 1: 1 day ahead forecasts on a random walk process; the black line corresponds to the realized values, while the red one plots the forecasted ones

can be easily seen looking at the figures (2, 3, 4, 5): a stationary process is easier to forecast with small errors, while random walks give our agent a very tough time. The differences in the forecast errors are smaller if we look at the 1 day ahead forecasts, while they get more remarkable (fig.5) just letting our agent build their forecast 2 days ahead in the future (conditioning on the same information set).

Even though it's practically pointless to do it using an ANN, we could have tried to build our forecast just on the basis of the last realized value (i.e. using today's price to forecast tomorrow's). This would provide a further insight into the random walk's features. We would have noticed the good quality of these forecasts, which are easily explained if we remember the fact that a random walk is also a martingale: the best prediction for tomorrow's price is today's price! The realized difference between tomorrow and today is just given by the stochastic component (i.e. the white noise), so we might conclude that whenever our agent has a very precise forecast for tomorrow's price, she's just been lucky!

It's worth stressing (in the case the reader is wondering if we discovered a way to predict future prices...) that forecasts two days ahead (even those with an acceptable quality, as in the case of the AR(2)) are not really determinant in order to realize a profit, since the forecasting horizon is too short: in other words the (alleged) clairvoyance skills of our agent are closer to gambling that we might think. We'll get back to this

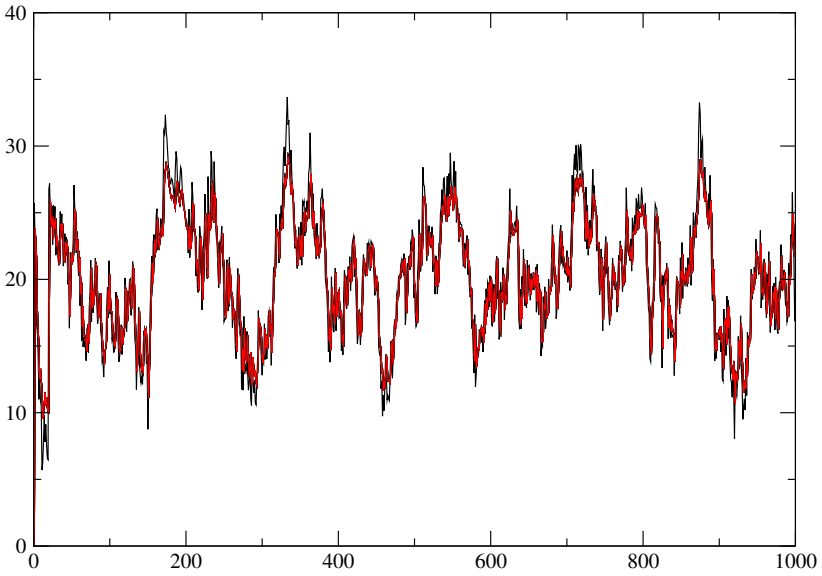


Figure 2: 1 day ahead forecasts (in red) on a stationary AR(2) process

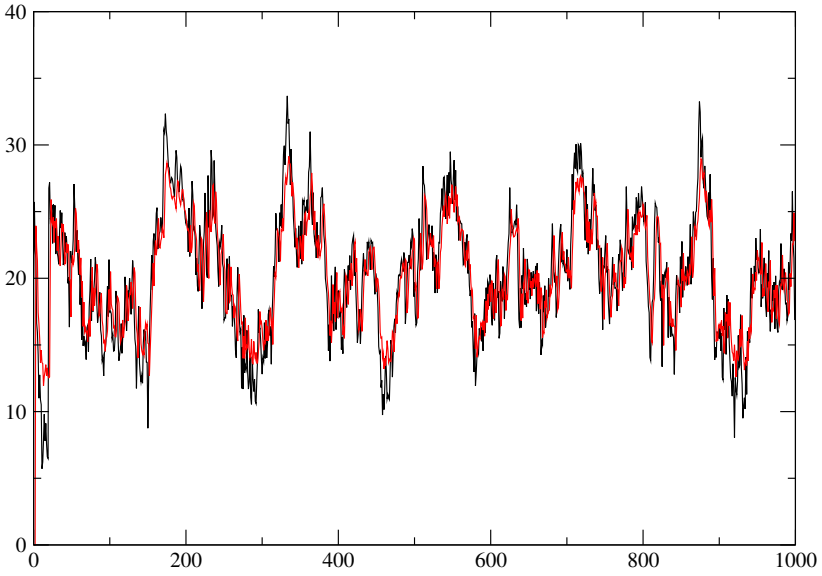


Figure 3: 2 days ahead forecasts (in red) on a stationary AR(2) process

later, when we show the simulation results; for the moment being, just realize that such predictions are not enough to construct a surely winning trading strategies (and even less important for portfolio choice purposes), especially if the underlying process for the stock price is a random walk. Of course, in reality it is very hard to get to know exactly which process is behind the time series: in our experiments the "trick"

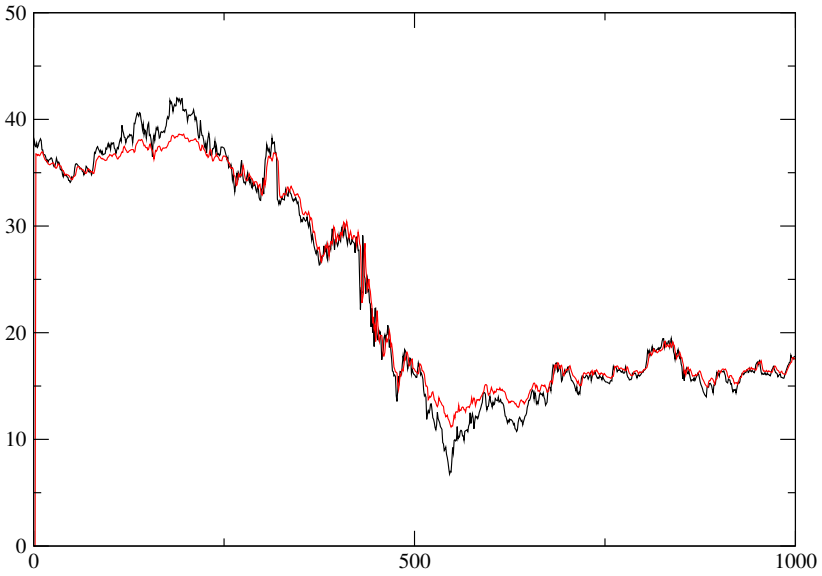


Figure 4: 1 day ahead forecasts (in red) on a random walk process

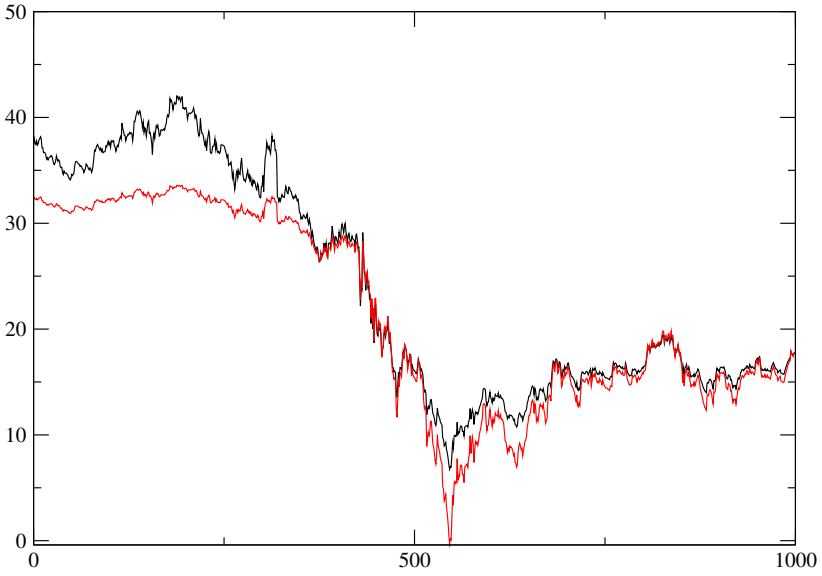


Figure 5: 2 days ahead forecasts (in red) on a random walk process

was that of knowing it ex-ante; this choice of ours was suggested by the objective of trying and confirm what we learnt in our econometrics classes by means of forecasts made using an ANN.

3.2 Netlogo

In this section we present the results of the simulation run using our NetLogo-based program. Our main goal was that of determining if the *intelligence* (i.e. the forecasting skills) provided to some agents was actually giving them a competitive advantage with respect to the others, leading to higher return and higher terminal wealth. We anticipate that we found positive evidence of the importance of this heterogeneity: specifically, agent who are able to make (quite) precise forecasts about future prices display better performance than the so-called dumb agents, a class that in our view should, in the simplified framework of our model, represent agents which are less skilled and just pick and drop stocks randomly. In particular the agent with the (supposedly) best forecasting skills, the one who uses the ANN, shows, on average, also the best performances. We will also provide an intuition of the reason why this result, though being the least surprising ones, are also a bit misleading: in other words, the forecasting skills are not as useful as one might think. In order to get to the above conclusions, we

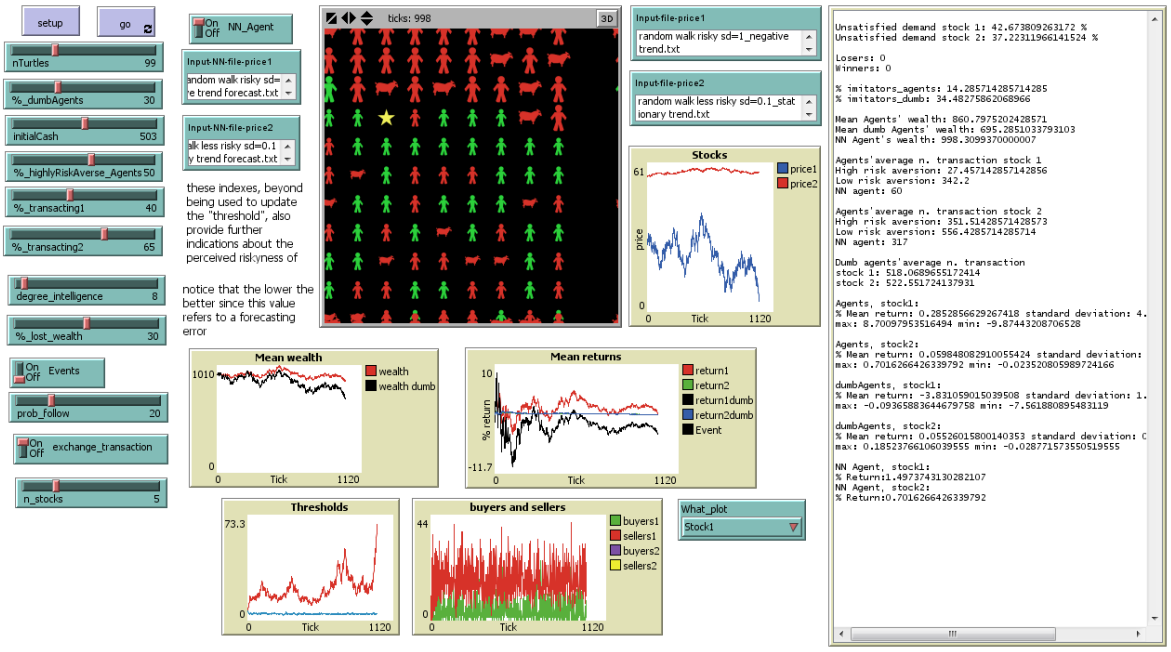
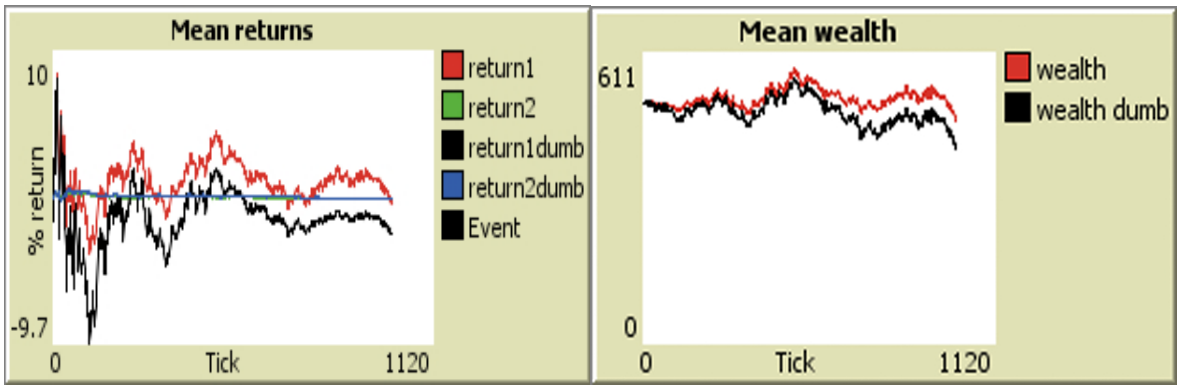


Figure 6: Software interface at the end of the simulation

perturbed and observed some of the variables and parameters which our model is built on: specifically the percentage of highly risk averse agents, the degree of intelligence of the forecasting agents (a proxy for their forecasting skills) and the level of initial cash; furthermore, we checked for the effects of the imitation process and the shocks generated by the "unlikely events".

We also used two different time series for the riskier securities: in the first case that we present just below we used a random walk, while in the second one the underlying stochastic process for the riskier stock price was a stationary AR(2). The less volatile stock has remained a random walk with a white noise (normally distributed with zero mean and 0.1 standard deviation) to ensure price stability. The simulation has been run using a stationary process also for the "safe" stock, but the results did not change significantly, thus we do not consider this difference from now on. 1000 observations have been considered in the simulation. We tried and increase this number but the results were unchanged. The initial price for the safer stock has always been higher than that of the risky one, in order to be consistent with the safety it provides to investors



(a) Mean Return (b) Mean Wealth

Figure 7: Mean Return and Wealth of agents and dumb agents

(thus it is more valuable). The lower initial price of the riskier security, on the other hand, is consistent with the fact that it should have a higher expected return, while the higher volatility represents its higher riskiness.

It's important to stress that the absolute result concerning the performances of the agents (mean return and terminal wealth) are very much influenced by the trend of the time series considered (the random walk had a negative trend with a final price lower than the initial one, while the stationary series, though showing a lot of oscillations, always fluctuates around a mean value, as expected from its typical *mean reverting* feature). This is straightforward if we consider that the agents' menu is made up by just two stocks, that, even though have different risk and return characteristics, do not leave enough space for an efficient diversification. Notice also that, as stressed above, we are not interested in the strategic allocation capabilities of the agents, but just in their ability to exploit the forecasts they make: they are just traders, more than portfolio managers, and market timing is the key for them. Anyway the relative results are independent of the time series used; in other words, the agents with forecasting skills perform better than the "dumb" ones even if the price trend is over all unfavorable. Interested readers can also try and substitute ours with any kind of time series they like, also actual ones.

The first experiments were run with a high degree of intelligence for the forecasting agents, quite a low amount of initial cash (around 200 units), 50% of highly risky averse agents and 15% of initial wealth (i.e. initial cash) as threshold to exceed before starting the imitation process. The riskier stock price has been generated by a random walk. With the above settings, we found that intelligent agents were able to gain a mean return and wealth for their investments in the risky stock higher than that of the dumb ones, as we can notice from fig.(7). A high volatility in this mean returns has been noticed, as expected. The returns for the safer stock were lower and less volatile, both for the intelligent and for the dumb agents. Therefore we conclude that intelligent agents performances are far better with this settings. Mean wealth (i.e. final cash plus current value of held stocks) is higher for agents which used forecasts.

The ANN agent shows performances on both stocks that are above the mean values of the other classes, with better results from the trade of the risky stock, while the return on the safer one is a bit lower. In particular, she is found able to always have a higher terminal wealth than the mean value for the others. This is true regardless of the kind of forecasts she uses: both the 1-day and 2-day predictions allow her to perform better, even though, obviously, her results are on average worse if the quality of the forecasts

is deteriorating. Of course, if we worsen the degree of intelligence, forecasting agents are more and more similar to the dumb ones, so the performances of the two classes get closer and closer.
If we change the initial value of cash there are only slight changes: returns are slightly lower (as expected) but the number of transactions is quite stable and there are the same differences among the three classes of agents. The number of transactions in the two stocks are strongly dependent on the percentage of highly risk averse agents. If we perturb this percentage, increasing it, we notice that there are less transactions involving the risky stock: both purchases and sales decrease (only for intelligent agents; remember that the dumb ones do not have a risk aversion parameter). This is consistent with our setting of the decision criteria of the agents: the decision function increases in the agents' forecasted increase or decrease in stock price and decreases in their level of risk aversion. Thus, if more highly risk averse investors are considered, passing is more likely and the number of transactions in the riskier stock decreases. If this percentage is lowered, more transactions in the latter take place, implying more volatile returns. Transactions in the safer stock are on average a higher amount because also very risk averse agents are a bit less indifferent to the risk brought by this security.
Looking at the imitation effect, we don't notice any emerging pattern: our initial expectation was that of finding more imitating agents on the dumb side, since they are on average losers in this market. Anyway, there is no emerging pattern and an equal percentage of intelligent agents experience losses high enough to trigger the mimicking process. This might be explained by the fact that agents with forecasting skills, even though they are almost always better off than at the start of the simulation (i.e. their wealth is higher than the initial one) may be strongly damaged by bad predictions (which still have an important stochastic component) in a bad moment, leading them to lose a lot of money and, as a matter of fact, align their behavior to the "less evolved" kin's one. This is one of the first hint of the fact that forecasts are on average useful to agents (if good quality ones), but a strategy completely based on them is much closer than we think to a purely random one (as that of the dumb agents). Furthermore, the imitation process does not really help the mimicking agents, since it is just a foolish follower strategy based on what the wealthiest agent (almost always an intelligent one) did in the previous turn, that is completely unrelated with what should be done in the current one (since the followed agent only has one forecast on which she builds her decisions).
The last shock that we used to perturb the model is given by including in the simulation the possibility for the agents to receive (perhaps) misleading information about unlikely event in the market like crashes and booms. It actually does not change much the final results and its effects depend a lot on the timing with which this event occurs: if agents receive information at the right spot of the time series, it may have devastating or miraculous effects on all the agents, as can be seen in fig.(8).

A second, but quite similar experiment, has been run using a stationary AR(2) as process used to generate the price series. Here, performances looks better since the price trend is more favorable in absolute terms. Furthermore, since forecasts are a bit better (as explained above), we expect results not to change much or, if so, in quite a favorable way for the ANN agent. Indeed, intelligence still favor the forecasting agents (that show even slightly better performances in mean return and wealth if compared to the dumb ones), and the ANN one's performances are still far above the mean ones, both for return and for wealth. Here we notice, that, due to the stationarity of the price series, the mean return series is also smoother and, after some initial volatile observations, it quickly stabilizes itself around the mean for both classes. Changing

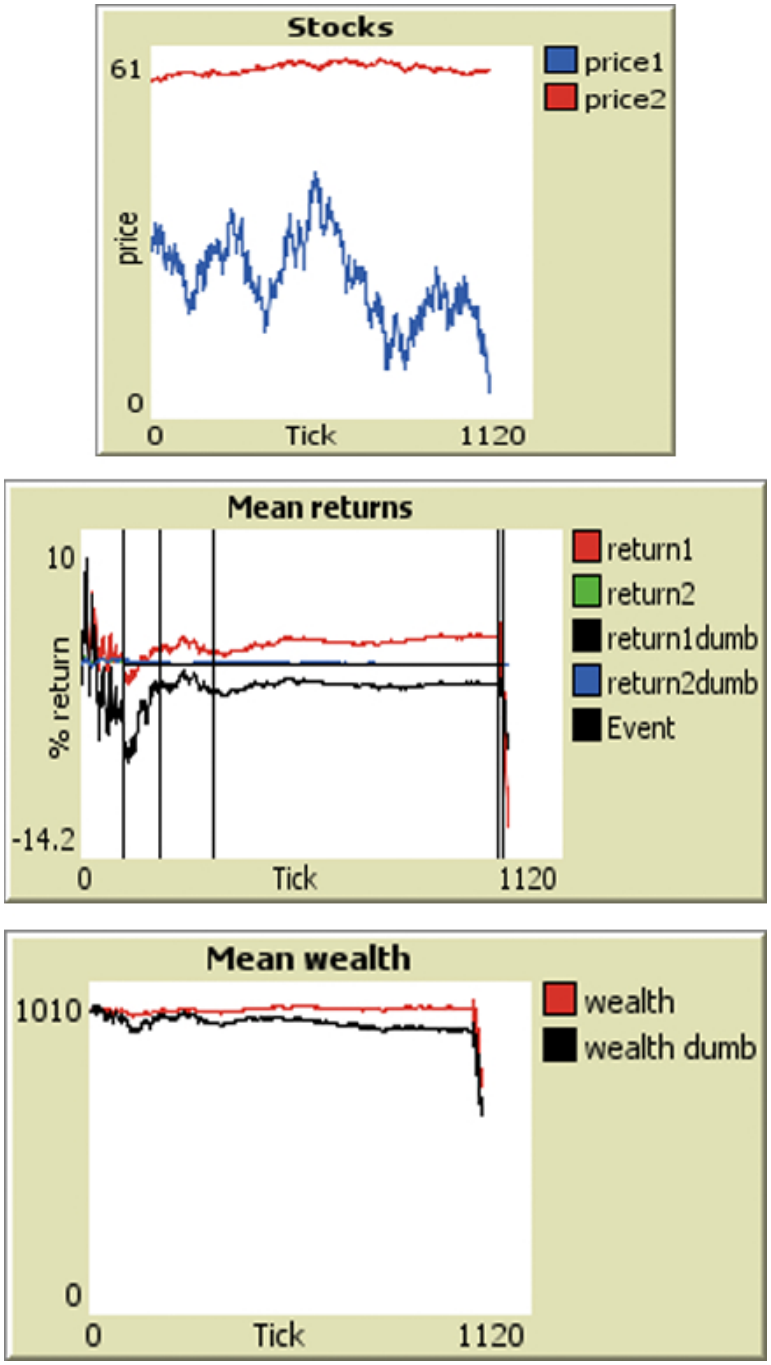


Figure 8: Effects on mean return and wealth of events in different periods

the level of highly risk averse investors, we obtain the same results as in the previous experiment. The transactions figures are also very similar to the previous ones. Here the imitation effect shows, on average, a higher percentage of dumb agents that become imitator: this is consistent with their worst performance. Still, it is not a helpful strategy either in this experiment. In conclusion, our results show that the forecasting skills and, even more, the ANN help the investors which have access to them. As announced above, this result is a bit misleading though. We are not stating that this is an optimal strategy (and, least of all, that in reality ANNs used this way are a breakthrough!): in fact our agents don't have any kind of long term view nor strategic plan of action since they are based only on very short term forecasts (some of which not even very good ones). Their actions are more random than they appear just looking at the user interface (but a look to the code will make the reader notice that this randomness is

not put in disguise at all) and, eventually they are just at the mercy of the price trends, with the only ability to minimize their losses (if good predictions are used at the right time) when the price trend is negative, as in our first experiment.

Finally, we present the results obtained by running the simulation with the exchange mechanism turned on. The performances of all the agents undergone some slight changes, that are more remarkable if the prices follow a strong trend. In other words, with the exchange on and a strong negative trend, the agents' performances are worser than with a more stable trend. This is due to the higher difficulty in concluding the transaction, since it is likely that demand and supply will be different. Therefore, in very bad periods, agents will have a hard time selling their stocks, while without the exchange mechanism they had no constraint. This effect is less important when the price is more stationary, since the difficulty in performing the desired transactions has less influence on their resulting performances. Therefore the above results are explained by the different amount of transactions in the two cases (exchange mechanism off and on): as expected, with the exchange on, the average amount of transactions is smaller for both the stocks and for all kinds of agents. Intuitively, this variation is strongly affected by the percentage of dumb agents in the market: since they act purely randomly, their decisions are not influenced by the price trends, thus the higher their percentage, the easier for the other agents to find a counterparty for their transactions. These results are clearly displayed in the output table, in the "unsatisfied demand" lines. It's worth noting that in the case of a stronly negative price trend for at least one of the stocks and no dumb agents in the simulation, the unsatisfied demand reaches peaks of around 55%, with an important effect on the performances. Furthermore, we report an unexpected result: using two types of stocks (one with no trend and another with a quite strong negative or positive trend), we would expect the percentage of unsatisfied demand to be much higher for the less stable stock, whereas, as a matter of fact, the unsatisfied demand is similar for both securities.

4 Conclusion

In this section, we summarize what the above results have brought us to and, not less importantly, where our inspiration came from. Most readers may have already noticed that our software lacks of some mechanisms that are typical of the growing literature on Artificial Stock Markets, like price formation and market clearing or any kind of discussion about the achievement of a benchmark rational expectation equilibrium.
The starting ideas for our work came mostly from the models created by Arthur et al. (1997), LeBaron et al. (1999) and Beltratti, Margarita & Terna (1996); anyway, we decided to focus on the effects of forecasts and the use of ANNs on the agents' behavior and performances, while those models provided deep and specific insights on the price formation and market clearing mechanism, the evolving ability of agents (both in their strategies and in their forecast accuracy) and a nice comparison with benchmark rational expectation equilibriums. Thus our work may appear very simplified, almost naïve sometimes, but still it allowed us to focus on the verification of some nowadays supposedly resolved issues (and nevertheless very much debated) in financial economics, by means of ANNs and agent-based modeling. We refer especially to the highly controversial random walk hypothesis for stock prices, which after the initial breakthrough, thanks to Bachelier (1900), has evolved in one of the most debated and interesting issues in finance: here we just recall the early critique of Mandelbrot (1963) and the more recent (and very clear) resumption by Lo & MacKinlay (1988).

Very much related is the issue about the predictability of stock prices and returns in the short and long run. We tried to analyze this issue in a very simplified framework through our time series and ANN results presented above. Campbel, Lo & MacKinlay (1997), Campbell & Shiller (1988) and Cochrane (2001) present very accurate surveys on this topic. In particular the latter present some interesting insights and interconnections among the predictability of returns, the random walk hypothesis and the latest asset pricing theories, as the C-CAPM.
Our work is very open (and perhaps needy in some parts) to updates, adjustments and additions. Looking at the extensive survey by LeBaron et al. (2005), some very nicely designed model suggest many of them, useful to evolve our model in a more realistic fashion. In particular the Santa Fe Artificial Stock Market provides interesting ideas regarding the construction of a price-formation mechanism, while the models by Chiarella & Iori (2002) and the Genoa artificial market by Raberto et al. (2001) offer a very realistic perspective about the market microstructure, something our model is frankly lacking. The first simulates a real order book where agents post offer to buy and sell stocks, while the latter implements an interesting market clearing property by using a batch-order book. It might be interesting to design and plug some mechanism of this kind and analyze the change in our model.

Finally, we would like to address one of the most common issues raised by those who first approach themselves to modeling in scientific and social research: what could we use them for? Though being a perfectly understandable question, it may actually be the wrong one to pose. If we look at our work from this perspective, one thinks it is almost completely useless: the agents act pretty much as we expected, their forecasts are good but not in the long run and, most of all, we stated that they are just apparently useful to improve their trading strategies. Hence, another question might be: was it worth spending our time? Our answer would be: most certainly it was!

As stated in a touching fashion by Epstein (2008), modeling is much more than just answering questions and providing practically useful tools and, more related to ours, their purposes go well beyond prediction. We believe that, among the sixteen valid reason other than prediction to build models that he states, perhaps the most important is raising (or discovering as he puts it) questions. In different disciplines, another great guy stated (liberally translated from Bobbio (1955)):
"*The task of scholars and academics is, nowadays more than ever, that of sowing doubts, not just reaping certitudes*".

5 References

Arthur, W. B., Holland, J., LeBaron, B., Palmer, R. & Tayler, P. (1997), 'Asset pricing under endogenous expectations in an artificial stock market', in W. B. Arthur, S. Durlauf & D. Lane, eds, 'The Economy as an Evolving Complex System II', Addison-Wesley, Reading, MA, pp. 15-44.

Bachelier, L. (1900), Theorie de la Speculation, PhD thesis, Ecole Normale Superieure, Paris, France.

Beltratti, A., Margarita, S. & Terna, P. (1996), 'Neural Networks for economic and financial modeling', International Thomson Computer Press, London, UK.

Bobbio, N. (1955), 'Politice e cultura', Einaudi, Torino, Italy.

Campbell, J. Y., Lo, A. W. & MacKinlay, A. C. (1996), 'The Econometrics of Financial Markets, Princeton University Press, Princeton', NJ.

Campbell, J. Y. & Shiller, R. (1988), 'The dividend-price ratio and expectations of future dividends and discount factors', Review of Financial Studies 1, 195-227.

Chiarella, C. & Iori, G. (2002), 'A simulation analysis of the microstructure of double auction markets', Quantitative Finance 2, 346-353.

Cochrane, John H. (2001), 'Asset Pricing', Princeton University Press.

Epstein, Joshua M. (2008). 'Why Model?', Journal of Artificial Societies and Social Simulation 11(4)12 http://jasss.soc.surrey.ac.uk/11/4/12.html.

LeBaron, Blake (2006) 'Agent-based Computational Finance', Handbook of Computational Economics. vol. 2 Ed. Tesfatsion and Judd. North-Holland, 1187-1232.

LeBaron, B., Arthur, W. B. & Palmer, R. (1999), 'Time series properties of an artificial stock market',Journal of Economic Dynamics and Control 23, 1487-1516.

Lo, A. W. & MacKinlay, A. C. (1988), 'Stock prices do not follow random walks: Evidence from a simple specification test', Review of Financial Studies 1, 41-66.

Mandelbrot, B. B. (1963), 'The variation of certain speculative prices', Journal of Business 36, 394-419.

Raberto, M., Cincotti, S., Focardi, S. M. & Marchesi, M. (2001), 'Agent-based simulation of a financial market', Physica A 299, 319-327.