# Problem

## Problem statement:

The forex market has the following qualities that make it an interesting (difficult) problem: it is imperfect and a dynamical system (in this case meaning that is appears to be unstable when observed from the inside); The observable variables have very inconsistent predictive influence on the future values of instruments; it is constantly changing in ways that tend to render previous .

## The Implications of the problem:

The patterns that the Indicators have to perceive are extremely diverse and fuzzy (they don’t look the exact same every time). This is because the market seems to be a dynamical and imperfect system [1].

The variables that should help predict the value of instruments have to weak of an effect to be useful

# How it works:

# Two time spans are taken into account a short time and a long time (the long time being many times longer then the short time)

## At the beginning of every long time span “social learning” occurs as described in (Kendali) over long term

Modification: Every agent has multiple ANNs for different situations

Modification: If an agent has seen a situation that there exists a published ANN for it will replace it’s own if the public ANN is superior.

Modification: Every time a ANN is taken from the pool the agent that created is awarded points.

Modification: The parameters that each agent has for creation of new ANNs are mutated with the highest scoring agents, and the lowest scoring agents with the lowest efficacy are eliminated.

# “Individual Learning” is performed as described in the Multi-Agent paper (Kendali) over short term

## Modification: that the agent keeps statistics about the short time spans

## Modification: according to the agents parameters new neural networks are created for situations that do not fit into the statistical bounds that have been established by training previous ANNs.

# Overview: Each agent consists of a collection of parameters determining: how ANN is created, when an ANN is mutated, when an ANN is to be used, and a decision tree for the specifications of the orders that it will place, and a collection of ANNs along with their statistics.

# Operation: Every “short term” statistics are gathered and the pertinent ANNs are mutated. Every “long term” the agents get to publish their ANNs that are effective above an established bound, when an agent takes a published ANN to replace a bad one that they have the publishing agent gets points. After ANNs have been exchanged the lower scoring agents parameters are “breeded” with the high scoring agents and replaced by the offspring. Every moment the agents are running the ANNs that match the current statistics to determine triggers and make trades

# Why: By using this method patterns for recognition are constantly changing there by allowing the agents to mutate with the market, and overspecialization is taken advantage of by only having ANNs work in situations they have proven to be good in yet it is controlled by the evolution of the agent parameters. Furthermore it is been shown many times that neural networks are good at predicting short term positions this method takes advantage of that fact and should extend it to be even more pronounced because the situations in which the ANNs have to preform less broad. This method tries to reduce the amount of chaos that a ANN has to deal with when deciding to buy or not thereby making them more effective

# Central Question: How do I profitably automate trading in a chaotic (possibly imperfect) environment?

# Thesis:

# The purpose of this paper is two fold

# There is a myriad of reasons that market prediction is difficult. This paper focuses on dealing with the problems presented in [2] and [3], namely that the market is non-stationary. The non-stationary nature of the market

# Address the chaos by having multiple agents with different parameters and triggers that are mutated with the other agents to better model of the market and to model the market better in the future. Decrease the amount of chaos that the agent has to deal with by splitting the situations that an agent has to evaluate into roughly statistically similar situations. Then optimize what roughly statistically similar means and what to do in each situation.