Learning Model Pool (a genetic optimization algorithm)

The purpose for this effort is to implement a model free and assumption free way of conducting a search or optimization, in a space that is either too poorly defined, or excessively high-dimension to provide visibility of any parameters to the search method. This will necessarily be gradient free, although it may have numerical notions of effectiveness, with respect to the goodness or utility of outcomes. This falls into the category of supervised learning, in that it assumes there is some measure-like concept of fitness or suitability. Each generation tries to learn or obtain an outcome from data, and treats that as an opportunity to modify or “mutate” some attribute of one of a number of candidate models.

A natural high-value application of this approach would be to programmatically optimize client or customer outcomes, as a function of several varieties of a component of business process, or other defined operations. Parameterized operations and processes, e.g. broking trading algorithms, are ideal for this. This is commonly known as “AB testing”, and is done informally, with two or more options, eventually adopting the better of the two, once enough outcomes have been seen. This would be a formal way of conducting an optimization among a pool of such options, methodically choosing better and better parameterizations.

Another high value example is with machine learning. An artificial neural network, or classic AI model will typically predict within sample very well, with a danger of overfitting and creating a model that can predict poorly out of sample. This badness of out of sample predictive ability can itself be severe, and this inapplicability can have severe consequences, if relied on. Typically, such models have been subjected to data bootstrapping and randomization to enhance the generality of the models that are estimated, but that approach is not always feasible, particularly with time series and recurrent neural net constructs. Such fixups are also limited by the availability and variety of data in any given dataset.

Simple selection criteria for birth and extinction of estimated model “genetic lines” will be applied, and the out of sample predictive behavior over time, as new (or mutated) data are used, will be the subject of the optimization process. Within each “generation”, models will be evaluated, optimized for the new data set, and again evaluated, with selection actions being applied for the start of a subsequent generation.

# The Generative Optimization Model

Let there be instances of some model that estimates a dimension of some state of the world as a function of . That is, For relevance of subject matter, and for motivation, suppose that the cannot be measured except after the fact, perhaps with some lag, and the purpose is to estimate its current value. This need not inherently be a time series model, but that is the example that motivates this work.

Each model of the models will have a predicted result for any given observation t. Let there be some measure for how well the predicted result matches up with the observed result, for observation .

For a concrete example, say that the model is the result of optimizing some other model over some training data set that includes some number of observations of the state of the world, . There is the estimated or optimized model , and there is some other evaluation data set , disjoint from the training data set. When the model is applied to observations from the evaluation data set, to generate predictions , let that measure of fitness be root-mean-squared error, for example. Or let that measure be r-squared, or let that measure be the correlation coefficient between the observed evaluation set results and the predicted evaluation set results, . A variety of such criteria might be valid for a given purpose, and might be easier or more difficult to optimize.

The process will consist of rules for how models are selected and mutated as generations of models applied to changed data are subjected to their own estimation.

Consider each of the existing models to be a “genetic line”, in a pool of genetic lines. All the genetic lines coexist and do not compete directly for resources with each other. Instead, the genetic lines compete against the “environment”, which consists of the training and evaluation data sets, and for each generation, bequest actions are taken. For each generation, the same rules apply, and determine the properties of the subsequent generation.

For each generation, with one or more new observations added to the training and evaluation data sets, each genetic line will

1. encounter the new evaluation data set and create some predicted results ,
2. optimize the model upon the new training data set to create a “child” model ,
3. have encounter the same evaluation data set to create some alternative predicted results , and
4. choose the superior measure or , and make that model the next generation’s “parent” model, .

→ → →

↓

*(max)*

↓

→ → →

For each generation, there is a change in environment (i.e. there is a change in both the training and evaluation data sets, by one or more observations), which will drive a different optimization result for . At each generation, the superior of the existing, or the “mutated” child models is selected as the basis for the ongoing genetic line.

# The Genetic Competition Model

With a fixed pool size, the convergence of this can be quite slow, and it would be worthwhile to consider how competition between genetic lines might drive selection among lines, for a place among the pool of models.

Suppose that an exception to the simple process above could include a way to create a new genetic line, and a way to extinct the least “fit” of the existing genetic lines, to make a place for the new line. We can re-use the child inheritance criterion above for that in some way, or we can arbitrarily apply some similar kind of logic.

Suppose that, in the eventuality that some model in the pool creates a child – call that child a “hero” in the classic sense – with sufficiently superior-looking properties, the child could start a new genetic line. But, there is no guarantee that any one instance of such heroic performance or fitness is anything but a coincidence, an artifact of the current training and evaluation data sets. There might be a variety of methods by which a decision to replace one “genetic line” with another line could be reasonably made.

To follow what is commonly believed to be the historical method from our own evolution, when there is a child prodigy who may be destined for greatness, or who may yet prove to be a flash in the pan, we’ve generally made the prodigal child prove out their fitness in some kind of ordeal or challenge. Should the candidate survive their “sojourn in the wilderness”, then when they return, they are given a place among the other models in the pool, replacing the least fit of them.

Let there be a further collection of “candidate” genetic lines, to be incubated, so to speak, in a flexible-sized “incubation pool”. Each genetic candidate line would be treated just as though it were one of the models in the fixed-size model pool, except that a) no candidate line would be able to spawn another candidate line for incubation, and b) any candidate genetic line whose fitness degrades sufficiently is dropped from the incubation pool.

(t+1, t+2,… )

(incubation pool)

Spawning a new line involves cutting off one of the two-branched bequest process from parents to children, and so it does represent a cost. It should only be done if a child has exceptional prospects.

When any candidate genetic line (i.e. resulting from a prodigal child model) is sufficiently mature, has survived some number of generations without degrading sufficiently to drop out, then it will replace the least fit of the genetic lines in the fixed-size pool of models.

Now we have two sources for mutation. One is the basic data itself, the training and evaluation data sets as they add at least one new observation for each generation, and change the inner model’s training result. The other is the idea of genetic competition via fitness with respect to a common environment, where lines in the fixed-size pool are replaced, and varying different inner model initial conditions are selected over time.

# Application to Production Systems

Frequently there are needs to optimize a nonlinear, impossible to parameterize or deterministically model production system. There are a variety of applications, but an example with which I’m familiar is the following. Consider a stock order execution algorithm, parameterized by some user “aggressiveness” dial setting, and a variety of additional worker-specific parameters, such as granularity with respect to so-called dark pools, surprise parameters for the taker worker, and crumbling bid parameters for a maker worker. As you might imagine from the exotic names, the outcomes might be virtually impossible to model in any theoretical closed form. The optimal parameters might be extremely difficult to know in any meaningful way, other than as experienced by those outcomes themselves.

In a commercial sense, that implies that there would be a production system that would choose to implement some directive like an order, using one set of such parameters, or another different set. This is the essence of what is called “A B testing” in mission critical production systems that have to sustain such a process of optimization over time. The production system mission is executed with one of *N* different approaches, and its outcome is measured in order to characterize the quality of that approach. Over time, the approaches to the production system’s mission success are measured.

A production system to accomplish some common hard to contend with goal, like filling a customer’s stock buy or sell order, could have a pool of approaches, like the genetic model described above. For example, the customer enters a VWAP order to buy or sell some stock, with limits and other user-defined parameters. But that is assigned randomly to a bucket of other parameter settings.

Suppose there are *N* VWAP order parameterizations, and the customer enters an order. That order is assigned one of the parameterizations, and executed. The order’s execution performance with the assigned parameterization is analogous to the model prediction performance above, and can drive a generational optimization process, if, for example the orders are assigned round-robin-style to an “algorithm parameterization” pool. The round-robin evenness of this process can serve as the generation concept, or that could be a controlled aggregation of production outcomes. Over time, superior performance parameterizations will be selected.

For production systems optimization, substitute “process parameterization” for model, and substitute “outcome” for prediction, with everything else as described in the Genetic Competition model above. The caveat is that the production process has to agree to assign demands and orders to the process parameterization buckets currently being evaluated. These should be assigned with roughly equal probability, for the sake of fairness to customers or clients served by the production process.

# Implementation

This model has been implemented as a python class, called ES\_FiveFactor\_LearningModelPool. That has two entry points, main\_loop(), and calculate\_current\_prediction(). Assumptions are largely enumerated in the \_\_init\_\_ method that functions as a constructor in python. There are numerous supporting implementation elements, mostly to provide specificity to the data and what constitutes the “state of the world”, described above.

For these purposes, the state of the world consisted of five futures front contract prices, and four front-to-second contract price spreads, and the hidden supervisory variable that summarizes relative price changes over the subsequent one to two hours. The latter is a bespoke backward-looking measure of concurrent price departure from a forward geometric moving average.

The model fitness criterion for replacement is the mean evaluation set correlation over the line’s lifespan, minus standard deviation, plus the log of the line’s lifespan, both scaled by some magical numbers, to tune the process to correspond to personal risk aversion and time value as a developer.

TBD

# Results

TBD – still observing and estimating them.

At present, the correlation coefficient is being optimized with a model pool of size five. After about a week of market data, the correlation coefficient on all pool models is bounded away from zero, positive, and is consistently between 0.002 and 0.008 for out-of-sample ES-mini futures contract price performance over the subsequent one to two hours.

Average lifetimes of genetic lines seem to indicate some thrashing, i.e. possibly excessively frequent replacement by incubated lines. That could be mitigated by a) making child prodigies less frequent, or b) making the incubation dropout criterion more tightly binding. We also see individual lines that spawn prodigal children impoverished by the loss of those children, which might have otherwise improved the performance of their parent’s line.

The first strategy above, less frequent spawning, could be caused to a small degree by disallowing spawning if the parent’s concurrent evaluation set correlation is excessively poor, less than some threshold level.