

GENETIC AI

Applying
evolutionary
strategies to
create a new
type of artificial
intelligence

WHITEPAPER
VERSION 01

by Philipp Wissgott¹
16.12.2024

1

Summary

For a long time, we were not sure if we should present Genetic AI to a broader audience. In the end, we decided that the gains of other teams pushing the technology forward far outweigh the risk of losing control over it.

A warning upfront: Genetic AI is complex. We have taken the freedom to describe the foundations and framework in detail - keep on reading, it is worth it!

We have taken our inspiration from genetic algorithms and evolutionary game theory, but then proceeded in a new direction: with an evolutionary approach, can we reach a new understanding of the nature of universal data?

To that end, Genetic AI employs the concept of genes and organisms interacting with each other. The interplay between genes (data features) and organisms (data sets) stands in the center of the technology. By controlling the resulting dynamics with specific evolutionary strategies, we manage to create a versatile framework for applications in data analysis and AI.

What makes Genetic AI unique as a technology in artificial intelligence? Employing very fast evolutionary simulations allows us to train AI models in real-time for the first time. This, together with the possibility to decentralize the training process, opens up new fields of technical opportunities for the taking.

Genetic AI – applying evolutionary strategies to create a new type of artificial intelligence

1. Summary	2	7. Mathematical Formulation of Genetic AI	13
2. Introduction – or how should a new artificial intelligence technology look like?	3	8. Evaluating the Quality of Results	15
3. An introduction to multi-dimensional sorting	4	9. AI Learning	15
4. An introduction to genetic and evolutionary algorithms	5	10. Complexity & Performance	16
4.1. Genetic recombination and the Fitness Function	5	11. Practical Applications of Genetic AI	18
5. An introduction to evolutionary game theory	6	11.1 As Recommendation Engine	18
5.1 The gene perspective	7	11.2 As Decision Engine	18
6. Introduction to Genetic AI	9	11.3 As Search Engine	18
6.1 Data organisms and data genes	9	11.4 As Discovery Engine	19
6.2 Organisms VS Genes	10	11.5 As Prediction Engine	19
6.3 Multiple Game Rounds	10	11.6 Full Multi-Organ Interconnected System	19
6.4 Organisms and Data Strategies	10	12. Our AI Requirements revisited	20
6.5 Customizing your Genetic AI - Personalization	12	13. Outlook	21
6.6 Types of Genes and the Distillate	12	14. Acknowledgement	21
6.7 Interconnected Evolutionary Simulations–Organs	13	15. Appendix	22

Introduction

or how should a new artificial intelligence technology look like?

Before we dive into the realms of Genetic AI, let us first take a step back and reflect about what kind of AI the world “needs”. The past decade have been driven by technological innovations in AI, but somehow people only thought about what is good or bad about it after it was developed or released.

With recent innovations in Large Language Models (LLM) we again missed the chance to actively state what kind of AI has a positive impact on our lives and society as a whole. Before Genetic AI becomes a “polished” technology, let us not redo this mistake and define must-have points

Stability²: stabilizing unstable AI solutions is currently done using colossal data and energy resources. We have to find a simpler way to obtain a stable answer from an AI.

Decentral Data and Model: central control of data and AI model creates distrust among users. A decentralized approach has multiple advantages in terms of societal value. It is clear that not all applications allow for complete decentralization - still, we have to move as close as possible.

Follow-the-rules³: AI has to follow given rules of law and ethics. There can be no question about it.

Transparent: The black-box behavior of many AI solutions makes it difficult to trust the results. Hidden agendas dominate corporate AI and it is difficult to see through the internal processes. Hence, AI has to transparently explain what it does and why.

Non-conformist: Many wide-spread AI technologies have the tendency to overrate average behavior as “good”. This leads to streamlining in markets and opinions, where we would actually need diverse solutions for the world’s problems.

Low Resources: The current energy demand of AI solutions stands in no relation to the created value. We need AI technology that uses as little resources as possible.

At first glimpse it seems unrealistic or even impossible to fulfill all these requirements in one single technology. As we will see, it is not only indeed possible, but Genetic AI also opens up new ways to follow-up technologies.

In order to approach the technical details of Genetic AI, we first have to introduce three important prerequisites: multi-dimensional sorting, genetic algorithms and evolutionary game theory.

² What we define as stability is also often called “robustness”, especially in the sense of “Trustworthy AI”. We mean by “stability” in particular two properties (i) the result of the AI does not change by small changes in data and model properties, and (ii) the result makes sense, or at least one understands why the AI fails (no unexplained, surprising “hallucination”).

³ This unifies the other two properties of “Trustworthy AI” apart from robustness: lawfulness and ethical behavior.

An introduction to multi-dimensional sorting

Sorting things is tedious business. Everybody who has sorted cards or other comparable items knows: it involves a lot of pairwise comparisons. Hence, it does not come as a surprise that sorting was one of the first things computers were much faster than humans some 70 years ago.

When we think of sorting, we usually think of one-dimensional sorting, i.e. the sorting of things according to one single parameter. Getting the cheapest flight from A to B requires sorting the prices from lowest to highest. Naturally, there is only one correct sorting order⁴.

What happens if we want to sort according to multiple parameters? Imagine you do not want the cheapest flight, but also minimal flight-time and intermediate stops (see *also Getting the right flight*). In contrast to one-dimensional sorting, multi-dimensional sorting (MDS) allows for multiple “right” solutions. In the end it depends on the situation and application what we would consider the correct result. In more complex systems defining a single, universally correct solution usually does not make sense at all.

In algorithms as in life, everything comes at a price. Mathematically speaking, there is “no free lunch” when it comes to which algorithm is good for which applications⁵. In this sense, it is a big advantage for MDS algorithms that they do not care about the “space” between their ordered items (i.e. how far A is before B is not important, only the order counts). Consequently, MDS algorithms can “focus” their complexity on getting the order right, while other algorithms may be distracted, calculating exact values of probabilities. In terms of the NFL theorem, MDS algorithms are therefore the best choice for MDS problems.

One might now argue that MDS problems are rare. Ironically, exactly the opposite is true: almost all problems containing any decision are MDS problems. Where do humans or machines have something to decide? Apart from consumer needs (which essentially dominate our everyday life), we also find decisions everywhere else: e.g. politics, economics and automation. Hence, MDS problems actually dominate analysis and prediction for structured data – let us now find a universal way to solve them.

Getting the right flight

As a simple example, let us define a very small multi-dimensional sorting problem – namely finding the right flight out of 3 options with 3 parameters each (note that we left out reasons of sustainability for brevity here):

	Price (Euro)	Time-of- transfer (h)	Stops
Flight A	300	10	2
Flight B	600	5	2
Flight C	1500	4	1

So: which offer is best? That depends on your budget and available time. Thus, any MDS algorithm has to take into account specific preferences to calculate a possible solution. Note that it might be unclear what a “right” solution is, but it is usually pretty clear what a false solution is (e.g. recommending a budget-driven person the most expensive flight).

⁴ That is why the different one-dimensional sorting algorithms are mainly benchmarked according to their performance. See [1] for a short introduction to the different sorting algorithms.

⁵ Compare also [2] for a general introduction into the “no free lunch” (NFL) theorem.

An introduction to genetic and evolutionary algorithms

Evolution is a serious but wonderful business [3]. Though the image that evolution directly converges to a clear “winning” organism is not valid, the idea that evolution poses a natural optimization for life forms, adapting to their habitat is in a sense universal. One might think: why not apply the philosophy of evolutionary processes to general optimization problems? Enter evolutionary algorithms (EA).

Live forms have reached every tiny corner of our planet. Consequently, EAs are on the one hand very versatile⁶. On the other hand, similar to the highly specialized species of a habitat, EAs tend to be rather specialized to a particular optimization problem. We now want to turn our attention to a special form of EA: genetic algorithms (GA).

Similar to EAs, genetic algorithms leverage the power of evolutionary contest to iteratively improve possible solutions to a problem⁷. One of the first steps in this process is the reformulation of the problem into genetic form. To that end, a possible solution is encoded into a chain of numbers. Creating and supervising a genetic model is often not straightforward – still in many optimization problems it provides an elegant way to close-in on the optimal solution.

Differences of sorting algorithms to optimization algorithms

In terms of algorithm taxonomy, sorting and optimization are quite different. In optimization, you usually start with an initial solution of low quality and want to iteratively improve the solution. Usually you change the solution over time aiming to create a better quality. How quickly one obtains better solutions and how good the final result is depends on the choice of optimization algorithm and the problem you like to solve. On the contrary, sorting algorithms have a fixed set of solutions and want to quickly bring them into the “right” order. Hence, no new solutions are created in the process. Analyzing given solutions (by sorting) is neither better nor worse than generating new ones (by optimization) – it is simply a completely different problem to be solved.

So, what is the point? An important take-home message is that one has to take care to select the right type of algorithm for the right kind of problem: optimization algorithms are not good in sorting, while sorting algorithms are not good in optimization.⁸

4.1

Genetic recombination and the Fitness Function

One of the most eponymous steps in GAs is how new solutions are created. Essentially, one mixes (“cross-over”) and/or manipulates (“mutates”) parts of one solution with another. These parts or units cannot be chosen at random, since not any “bred” solution would be valid. Hence, these parts can be interpreted as genes in genetic optimization.

To measure the quality of a solution s , genetic algorithms employ a so-called fitness function $F(s)$. The fitness $F(s)$ can be used to compare solutions amongst each other and sort them. It is important to mention that the fitness function $F(s)$ is global for a system and usually stays the same for the entire optimization. Note also that computing $F(s)$ of all solutions

of a generation can take some time, depending on the complexity of the system.

In our route to Genetic AI, GAs pose an important milestone: to evaluate solutions according to their fitness and iteratively “improve” a population. However, in the data-driven sorting problems that we aim for, we usually have given data sets that we do not want to change.

Hence, we need a formalism describing how a system of fixed solutions interacts with each other. This is where evolutionary game theory comes into play.

⁶ See also [4] for a good overview of evolutionary algorithms.

⁷ See also [5] for a discussion on methods in genetic algorithms.

⁸ One can however combine the merits of both worlds to a common “super-algorithm”. One way of doing this is to take the generative magic from the optimization algorithm to create a set of new solutions every iteration. The sorting algorithm takes these new solutions and sorts them into order. The front-runner solutions are then handed on to the optimization algorithm again to create solutions for the next iteration.

An introduction to evolutionary game theory

It was 2001, when the movie “A Beautiful Mind” spotlighted game theory for a non-mathematical audience. In the movie, the real story of John Forbes Nash mesmerized people all over the world. Interestingly, Nash is one of the few mathematicians obtaining a Nobel Prize (in economic science).

Aside from “A Beautiful Mind”, it is somewhat ironic that game theory is one of the most underestimated fields of mathematics. The reason for this lies in the fact that the examples and investigated systems look simple at first glance⁹. However, on closer examination, these “simple” systems hold rich dynamics which often take years to understand.

In the 1970s, and with game theory in mind, John Maynard Smith and George R. Price were asking themselves

Why are rivaling male animals not killing each other more frequently while fighting against each other?

Taking out opponents for reproduction appears to be a viable strategy in a naive game of “survival of the fittest”. Somehow nature thinks otherwise and usually lets a more balanced strategy between aggression and defensive behavior prevail (though, not always: see [6]).

Hawks VS Doves

Assume you have a population with two principle survival strategies: **(H) hawks type:** members of this group will fight every opponent viciously for any resource, **(D) dove type:** members of this group will give in on aggression and otherwise try to cooperate with any opponent to share any resource. For every round of the game one selects two members of the population at random and lets them interact where the **resource V** is the prize. If two hawks meet each other one will win half of the time while losing in the other cases (“losing” meaning they have to “pay” the price C in resources).

One can visualize the rules of the game as follows

Payoff Matrix	meets hawk	meets dove
If hawk	$V/2 - C/2^{10}$	V
If dove	0	V/2

Adding evolutionary concepts to game theory, evolutionary game theory [7,8] managed to explain many at the first glance surprising strategies in animal behavior, like altruism and eusocial animal colonies (see also Hawks VS Doves).

Every game round in evolutionary game theory has two principle phases

1. Game rules phase: Choosing two members of the population at random and applying the payoff matrix to their interaction. Note that the higher the dove-population the lower the probability of a hawk meeting another hawk. Hence, ironically, the fewer hawks the more effective hawk-like behavior is.

2. Replicator rules phase: The more resources a strategy group can gather, the more numerous they will be in the next game round. This is mathematically described by replicator equations like $\Delta P = P (F(P) - \text{Mean}(F(P)))$ where P can either be H or D in our example and F(P) is the fitness of that population group (how many resources they could accumulate). It is an important aspect of evolutionary game theory that individuals cannot change their strategy – they are born and die with it.

Running these two phases iteratively one can observe how a model system develops with time. Will hawks or doves win in the long run? This essentially depends on the relation of hawk-hawk-penalty C to the obtained resource V: an evolutionary stable strategy (ESS) is a hawk population of V/C . This explains that if the penalty C is high (getting killed) why in male-male animal fights the opponents do not fight to the death to win. Too risky strategies for reproduction simply were evolutionary not stable in the sense that a more defensive strategy prevailed.

⁹ Game theory investigates the dynamics of rule- and round-based “games”. In game theory a “game” can be anything from social interactions to the politics between nuclear powers. One of the most popular games analyzed is the “Prisoner dilemma”: 2 players (P1, P2) can independently decide to cooperate (C) or defect (D). In this case there are 4 possible outcomes of a game round (P1=C/P2=C; P1=C/P2=D; P1=D/P2=C; P1=D/P2=D). Each result has different rewards/punishments. Playing the game multiple rounds has interesting implications in terms of social behavior.

¹⁰ The value $V/2 - C/2$ is a statistical result derived from the 50/50-win/lose-rule

The gene perspective

One major insight brought forward is that a single animal organism does not act on its individual survival, but it is the genes that pass on successful survival and reproduction strategies that are actually “playing the game”. That evolution can be seen more as a contest of gene fitness changes the way researchers could analyze the long-term behavior of evolutionary systems [9].

Of course, animal populations are not the only thing that one can put into a model of evolutionary game theory. In fact, the theory has major implications in understanding human behavior and societies. For data analysis and sorting however, the original formalism lacks some flexibility and performance:

- The splitting between game phase and replication phase makes the models rather specialized
- The pairwise interaction between members of the population is relatively slow.

Imagine there would be a way to let all contestants, organisms and genes interact with all others in just one single step. This approach paved the way for Genetic AI.

Is it data? Is it evolution?

In this paper we juggle with many terms from data analytics and evolution. Let us explain what is meant with the used terms and their relation with each other:

The data picture

Data model: data describe things. A data model is a “shelf system” system to get structure into chaos. In this paper, we only treat structured data, i.e. data that you could actually split and distribute into the right “shelf”. Genetic AI can be applied to any type of structured data. For the sake of presentation, we will restrict ourselves to data models that can be written in the form of Excel tables or matrices here. Note that this does not mean that the data have to be of numerical type, but can also be a list of labels.

Data set: a data set is a single row in our data model. Data sets do not have to be complete, i.e. the matrix can contain blank spaces.

Data feature: a data feature is a single column in our data model. Hence, it describes the same attribute in all data sets.

Data element: a data element is a single cell in our Excel table. Thus, it describes a certain data feature for a certain data set. It is the smallest unit of information in the system.

The evolutionary picture

Evolutionary system: an important quantity of the systems we investigate, is that they are self-sufficient, i.e. it is assumed that all necessary information is covered by our evolutionary model.

Organism: an organism or individual contains a set of gene variants. Organisms will behave as one entity, making decisions based on their strategy. All individuals together represent the population.

Gene: genes are grouped features of the organism. They describe the same type of attributes in organisms and are therefore logically connected to each other. A biological, greatly simplified example is the gene storing the information of the size of an organism.

Gene variant: organisms contain a certain variant of a gene. One can think of the variant as to how a gene “acts” for the organism, e.g. how big an organism will be.

Fitness function: organisms and genes will be evaluated in terms of fitness. The choice of fitness function depends on multiple factors and can be completely different for two systems and even genes.

Strategy: In this paper, strategies mean how genes and organisms interact with each other. Genes and organisms can follow completely diverse strategies for “success”.

Analogies between data and evolutionary picture

Evolutionary System <> Data model: for Genetic AI, a data model is an evolutionary system with certain organisms and genes. The purpose of the prepared system is to be analyzed by evolutionary simulation.

Organism <> data set: data sets “behave” like individual organisms according to their strategy.

Gene <> Data feature: in Genetic AI, we treat data features as individual genes competing according to their strategy.

Gene variant <> Data element: consequently, data element entries are interpreted as gene variants of an organism. Like data elements, gene variants are the smallest unit in the evolutionary system.

Introduction to Genetic AI

From a broader perspective, evolution is the most successful and universal process of problem solving available for mankind. Any process of chemistry, structural optimization, transport, distribution of resources has already been solved by nature at some point.

Unfortunately, evolution is slow in terms of human life expectancy – nature will find a way, but you do not know how long it will take. Hence, we would like to keep the universality, but speed up the process to find the right solution.

With these conceptual prerequisites and ideas, we can now tackle our main target: a sorting algorithm for all types of data.

Let us summarize the main drivers behind Genetic AI, their drawbacks and how we intend to solve them:

Genetic Algorithms: encoding a universal problem into genetic information brought us a big step forward. Unfortunately, the encoding for an optimization problem is usually very specific. By “converting” into a sorting problem of fixed structured data, we aim to shift the encoding problem to a local genetic fitness function. This, at the same time, also improves performance issues with computing a global fitness function.

Evolutionary Game Theory: a big advantage of this approach is the universality of the game dynamics according to the choice of strategy. Unfortunately, the split between game rules and replicator equations together with the pairwise resolution of game rounds leaves room for improvement for our needs. We aim to introduce global strategies for genes and organisms such that we can compute the individual fitness of everything all at once.

6.1

Data organisms and data genes

Let us remember what an “organism” in our evolutionary data framework is: a data set (organism) contains a set of data element (gene variants), e.g. a flight (organism A) costs 300 euros (gene variant a), takes 10 hours (gene variant b) and has two stops (gene variant c):

Evolutionary Data Model

		data feature a	data feature b	data feature c
organism A	data set A	element (A,a)e	element (A,b)	element (A,c)
organism B	data set B	element (B,a)e	element (B,b)	element (B,c)
organism C	data set C	element (C,a)e	element (C,b)	element (C,c)
		gene a	gene b	gene c

Fig.1: Organism and genes interpretation for data

Our goal is to evaluate and sort the fitness of all organisms and genes in the system. The fitter an organism (data set) is the higher its rank in the final sorting order (*see also Box Is it data? Is it evolution?*). To get ranked, organisms follow a strategy. The choice of the right organism strategy (OS) depends on the system, but also on the application and/or personalization (*see also further below*).

In Genetic AI, the second central quantity besides organisms are genes: they represent a data feature of the system. Like organisms, genes usually share the same gene strategy¹¹(GS) in the evolutionary model. Note that this is one major difference to evolutionary game theory where you usually have multiple competing gene strategies.

¹¹ Usually you want that all organisms follow the same strategy, such that you can compare the data sets of the system with each other.

Organisms VS Genes

With OS and GS in place, we can start our evolutionary “game”: in every game round (or iteration), genes compete against each other and against all organisms. They do this according to their gene strategy. At the same time organisms compete against each other and against all genes – again according to their common organism strategy.

What are the game rules? Every organism and gene has to “act” once per round. Action usually means changing the gene fitness in some way:

Evolutionary Game Round

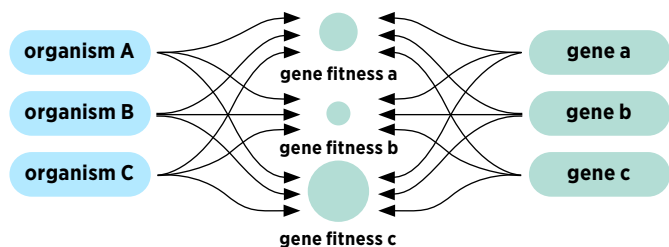


Fig. 2: Organisms and genes change the gene fitness

After all organisms and genes have acted, one obtains a new fitness for all genes (data features) and organisms (data sets). Consequently, one can sort them and get the desired results for this iteration. Note that with Genetic AI you can sort both, data features and data sets, in one single step.

Multiple Game Rounds

After one iteration, one obtains new fitness results for organisms and genes. Depending on the application, one can repeat the game with these new values. If an evolutionary stable state (ESS¹²) can be obtained, depends on the OS, GS and the given data sets. Note that some combinations of OS+GS always reach ESS.

Since we want to save time and resources, we can stop our evolutionary simulation when the organism fitness $F(\omega)$ and/or gene fitness $F(g)$ is not changing too much over different iterations. There are also tricks to “preview” the final fitness values with just one iteration (see below).

A main part of research in Genetic AI comes down to investigating new, compatible strategies. For example, genes and organisms can act fair, unfair, collaborative, egoistic, altruistic – the optimal choice always depends on the application.

Organisms and Data Strategies

In Genetic AI a lot depends on choosing the right combination of OS+GS. In a nutshell, one often wants both sides, organisms and genes, “pulling” in different directions:

Evolutionary Balance

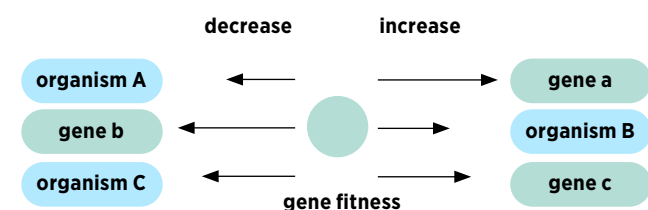


Fig. 3: Organisms and genes “pull” on the gene fitness according to their strategy (different arrow lengths represent the strength of the “pull”).

So how do successful strategies look like? In the following, we give a simple example of a useful combination:

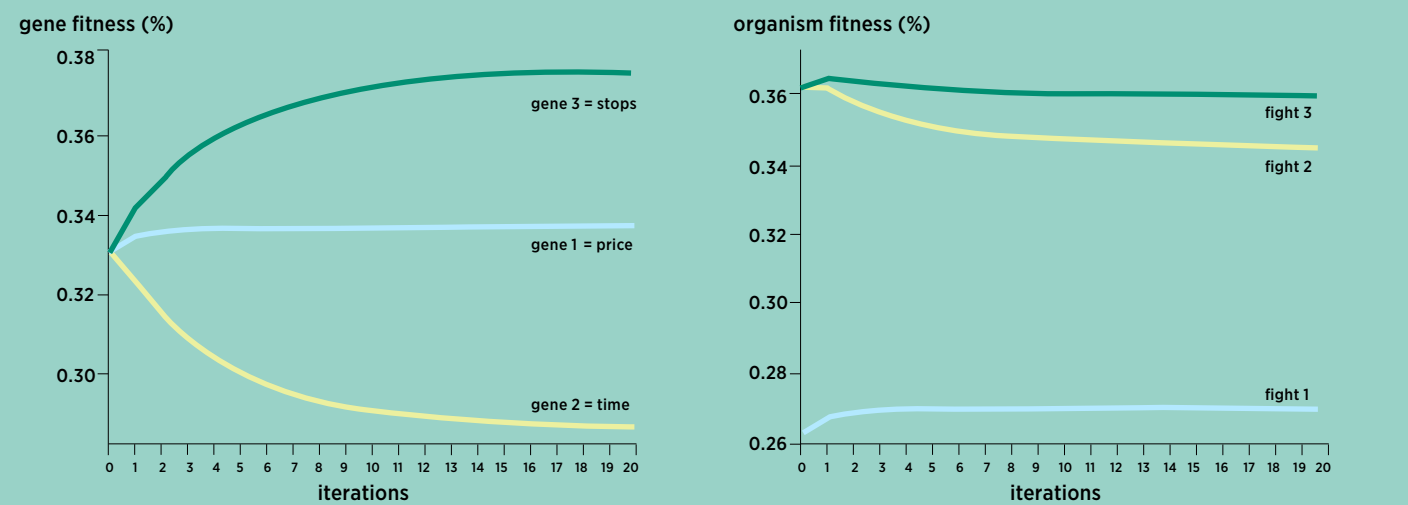
Gene strategy – GS Dominance: in this strategy, gene variants test if they are better than 50% in their gene. If yes, they increase gene fitness depending on how much better they are. For gene variants below 50% they consequently reduce the gene fitness accordingly.

Organisms strategy – OS Balance: in this strategy, organisms give genes a fitness penalty, if they tend to dominate their fitness. Note that the more genes try to become more dominant, the more an organism will “punish” them. Hence, OS Balance somewhat works against GS Dominance.

¹² Note that ESS usually stands for “Evolutionarily Stable Strategy” in evolutionary game theory. Since strategies are usually fixed anyway, this concept does not make much sense in Genetic AI. Here, a stable state for the fitness values is often desirable. Hence, the last S in ESS stands for state in the following.

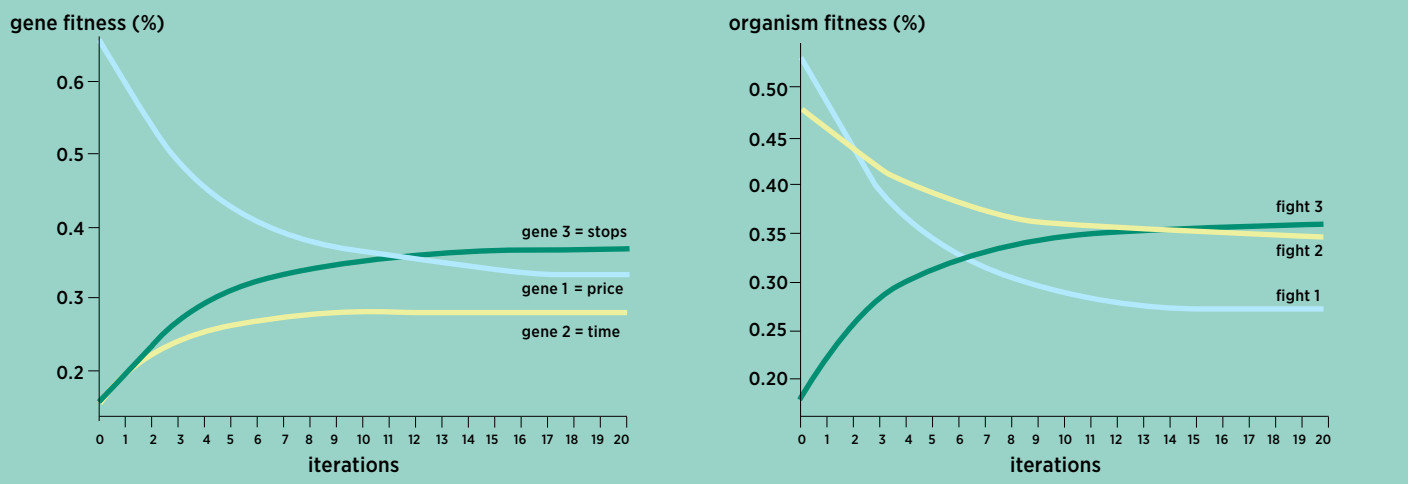
Getting the right flight, revisited

Let us revisit our example of three flights once again. What is the answer of Genetic AI (taking GS Dominance and OS Balance)? As a first try, let us assume that we do not really know anything about our system and which data feature might be more important for the interested passenger-to-be. We set the initial gene fitness of price, time and stops to 33% each. We start our evolutionary simulation and get the following result after 20 iterations.



Note that for this example and setup, Genetic AI thinks that the number of stops is the most relevant data feature of a flight (at 37,5%) and recommends taking the third flight (the most expensive). Price comes out second at 34% importance and time finishes last at 28,5% (see also *Appendix for a more detailed analysis of these results*). Note that the system shows a nice behavior towards an ESS. However, the decision between second and third flight happens already after one iteration, where the green line is above the orange and stays there. Hence, one could essentially stop the simulation at this point.

A more budget-driven user might intervene and say: “Wait, I should take the most expensive flight?”. We can include this kind of consideration by doubling the initial gene fitness for price and reduce the fitness for the two other genes accordingly. Again, we make 20 iterations to obtain the following result.



First, note that the ESS is the same, independently of the starting values for the chosen strategies and the system needs all 20 iterations to stabilize. However, in Genetic AI, we can stop the simulation at any intermediate step. If we stopped at 4 iterations, the answer to the right flight would be “Take the second flight”, since it offers the best balance between price and time. It is important to mention that the point of stopping is not a random choice for Genetic AI. Unequal starting values for gene fitness can be seen as a perturbation of the system. The way that the evolutionary system reacts to this perturbation holds information in itself. Whether one lets the system relax to the ESS depends on the application and performance considerations. Choosing the right time to stop the simulation can also be trained controlling the quality and plausibility of results.

Customizing your Genetic AI – Personalization

We have now introduced the basics of Genetic AI. As an immediate follow-up question one might ask: how useful is it? In the most part this boils down to how flexible it can be applied to different (sorting) problems (for the quality of results see below).

Customizing your Genetic AI model usually works with two methods: First, one can adapt the OS+GS combination. Second, one can change the initial fitness of genes. Since the former represents a question of current research, we will focus on the initial fitness in the following.

By adapting the initial fitness of genes you can essentially tell the evolutionary simulation your preferences. Imagine you look mainly at the price when it comes to flights (see

Get the right flight, revisited). Then, you will give the gene describing the price an initial push. Consequently, you can create a personalized “profile” of gene fitness values for each application/user.

Setting up the right initial values for gene fitness can seem difficult at first. You can however use a reinforcement learning approach to improve these values over time.

It is interesting to interpret the change of the OS+GS combination as a preparation of the “right” habitat for the evolutionary simulation. At the same time, changing the initial fitness of genes corresponds to defining the “right” starting conditions for the simulation. Both means together provide a very versatile way to customize Genetic AI.

Types of Genes and the Distillate

In the practical application of Genetic AI one often finds two types of data features: (i) “hard” features like the price of a product where the values do not change that dramatically (ii) “soft” features such as the correlation between user actions of different products that change all the time as the system progresses. Correspondingly, hard data features are described by hard genes and soft features as soft genes. Both types of genes usually require a different handling, fitness function and, most importantly, strategies. Most of the time, hard genes are much easier to handle and can be grouped easily. Soft genes on the other hand require special treatment and preparation.

The main problem with soft genes is that they usually store a lot of information. Hence, one requires a special method to update and control soft genes and quickly compute strategies and fitness values: distillation.

Imagine you have thousands of user interactions per hour you want to use as a soft gene in Genetic AI. Then you need an intermediate quantity between your application and your Genetic AI model:

Data Distillation in Genetic AI

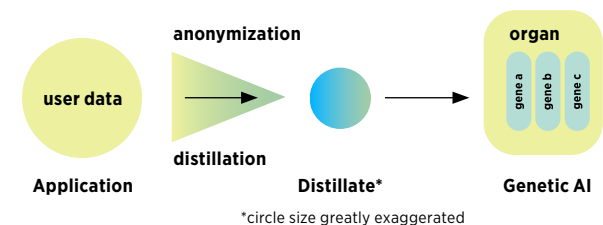


Fig. 4: The process of data distillation in Genetic AI. An application continuously feeds data into the distillate during which it is anonymized. An organ covering the user data's genes “consumes” the distillate when it calculates the corresponding fitness values (this can also happen at a later time as distillation occurs). All steps happen dynamically and in real-time.

While user data bases might contain above 100 GB of data, distilled data is usually smaller by at least a factor 100. Still, the distillate contains the collective information about the soft gene and allows for a fast evolutionary simulation. By distilling the user data you also obtain a useful side effect: the data get fully anonymized and cannot be tracked back to the user.

Interconnected Evolutionary Simulations – Organs

In real world applications, one can often group genes together according to what they describe. Evaluating these gene groups and the corresponding sub-organisms independently from the full system, allows us to choose the OS+GS very specifically. In Genetic AI we call these grouped sub-simulation organs:

Organs (Genetic AI)

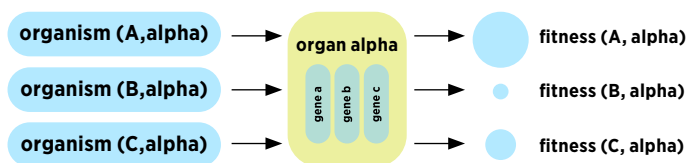


Fig. 5: An example of a Genetic AI organ. The organ computes the fitness for the alpha-part of the organisms.

Organs produce a set of fitness values for the genes and organisms they contain. Multiple organs can be combined in an hierarchical way to “work” together to solve the full problem. Usually you channel the information of all organs into one final simulation containing the all preliminary fitness values as individual genes:

Multi-Organ System (Genetic AI)

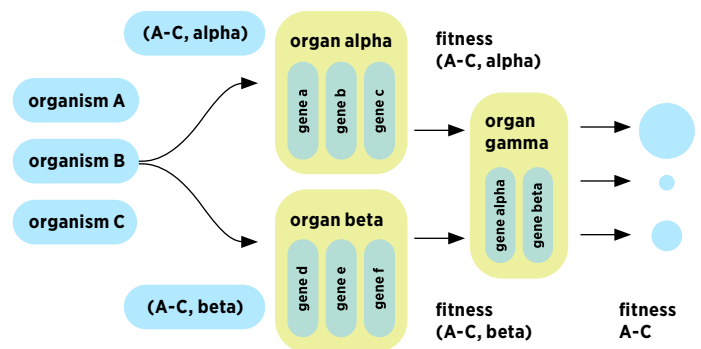


Fig.6: An example of a multi-organ system containing 2 preliminary organs alpha and beta. Note that the partial gene fitness values are channeled into a final organ gamma computing the overall fitness for the organisms A-C.

Another aspect of multi-organ systems is their capability for parallel processing. Since many organs can do their part in the analysis independently from each other, one can save a lot of computational time if one lets them run in a parallel way.

Mathematical Formulation of Genetic AI

Similar to the conceptual setup, Genetic AI also takes many inspirations from GAs and evolutionary game theory when it comes to the mathematical framework. Note that this section is not obligatory to follow the other parts of this paper.

In the previous sections, we outlined the basic principles in Genetic AI: genes interact with each other and with organisms (a set of gene variants). In mathematical terms, we now have to define the describing quantities that evolve during the evolutionary simulation. In most cases, one follows the progress of organism and gene fitness through the generations (iterations). What genes/organisms become fitter, which see a decline? We call the equations by which the system evolves, replicator equations, since one can interpret the iterations in the simulation as individual gene abundance becoming higher or lower.

Let us denote the raw data matrix as X containing n rows (organisms) and m columns (genes). Then, we have as genes and organisms, respectively

$$\vec{g}_j = [f_{g_j}(x_{1j}), \dots, f_{g_j}(x_{nj})] \quad \vec{\omega}_i = [f_{g_1}(x_{i1}), \dots, f_{g_m}(x_{im})]$$

where $0 \leq f_{g_j} \leq 1$

denotes the fitness variant function of that gene¹³. Our goal is to compute the fitness of genes and organisms, respectively¹⁴

$$0 \leq \gamma_j^{(k)} \leq 1 \quad F_{\omega}(\vec{\omega}_i, \vec{\gamma})$$

where (k) denotes the iteration of the evolutionary simulation¹⁵. Note that denotes the initial values for the gene fitness that have to be provided with the start of the simulation.

¹³ Note that we distinguish between fitness of the entire gene and the fitness of the gene variant inside of the same gene. One can see the fitness variant function as a way to measure how strongly a particular gene is active in an organism, whereas the fitness function measures how strongly a gene is active in the whole evolutionary system. In the most simplest case, with binary data 0 or 1, the fitness variant function is $f(0)=0$ and $f(1)=1$, i.e. the identity.

¹⁴ Note that we assume a scalar gene fitness here for simplicity. In general, the gene fitness is described by a function similar as for organisms.

¹⁵ Note that in most cases also the organism fitness is a percentage value, but there is no need to restrict ourselves here.

Replicator equations for Genetic AI

Analogously to evolutionary game theory, replicator equations describe how to obtain the gene fitness of the iteration (k) from other quantities. Of course, our replicator equations depend on the chosen strategies OS+GS. Let us start by defining the local effective change to the gene fitness stemming from gene and organism strategy, respectively

$$\Delta_{ij}^{g,(k)} = G^{GS}(f(X), \gamma^{(k-1)}, \dots, \gamma^{(0)})$$

$$\Delta_{ij}^{\omega,(k)} = \Omega^{OS}(f(X), \gamma^{(k-1)}, \dots, \gamma^{(0)})$$

Please see **Equations for GS Dominance and OS Balance** for an example of these functions. Note that these Deltas can be readily interpreted as what distinct change a specific gene or organism triggers for a specific gene variant. This feature of Genetic AI is crucial for transparency, since one can always track down the source of any behavior of the evolutionary simulation. Obtaining the gene fitness for gene j in iteration (k) then boils down to accumulating all contributions

$$\Delta_{\gamma_j}^{(k)} = \sum_{i=1}^n (\Delta_{ij}^{g,(k)} + \Delta_{ij}^{\omega,(k)})$$

and normalizing the their effect

$$\tilde{\gamma}_j^k = \gamma_j^{(k-1)} (1 + \Delta_{\gamma_j}^{(k)}) \quad \gamma_j^{(k)} = \frac{\tilde{\gamma}_j^{(k)}}{\sum_{\ell=1}^m \tilde{\gamma}_\ell^{(k)}}$$

In a nutshell, replicator equations describe how genes and organisms evolve during the simulation. Additionally, they take care that the total fitness “available” to the genes stays the same. Since no new fitness is created, the genes have to “battle” for their share in every game round.

Equations for GS Dominance and OS Balance

To understand the interplay between genes and organisms, let us take a closer look at our two example strategies. To that end, we also need a simpler approach to organism fitness

$$r_i^{(k)} = F_\omega(\vec{\omega}_i, \vec{\gamma}^{(k)})$$

for iteration (k), namely a linear approach

$$r_i = F_\omega^{lin}(\vec{\omega}_i, \vec{\gamma}) = \vec{\omega} \cdot \vec{\gamma} = \sum_{j=1}^m f_{g_j}(x_{ij}) \gamma_j$$

Note that the linear organism fitness is built up from linear combination of gene variant fitness and gene fitness. This follows the interpretation of organisms as competing sets of gene variants [9].

We now define GS Dominance with the following equation

$$\Delta_{ij}^g(\text{dominance}) = \frac{1}{n} \gamma_j [f_{g_j}(x_{ij}) - \frac{1}{2}] \cdot \sum_{\ell=1}^n f_{g_j}(x_{\ell j})$$

Note that for all gene variant fitnesses bigger than 50% the strategy will increase the gene fitness, in other cases it will lower the gene fitness. Hence, genes with more gene variant fitness will dominate others who have less.

As a counterpart let us define OS Balance as

$$\Delta_{ij}^\omega(\text{balance}) = -\frac{1}{m} F_\omega^{lin}(\vec{\omega}_i, \vec{\gamma}) \left[\frac{\gamma_j f_{g_j}(x_{ij})}{F_\omega^{lin}(\vec{\omega}_i, \vec{\gamma})} - \frac{1}{m} \right]$$

Note that the sign is exactly the other way around as in GS Dominance. The equation compares the local contribution of the gene (variant) to the full organism fitness. If a gene is too dominant in an organism, it will be reduced consequently.

Evaluating the Quality of Results

After you have run the evolutionary simulation, you obtain your final values of fitness of organisms and genes. So depending on whether you wanted to sort data sets or data features you can consequently sort the corresponding fitness values.

But what is the right sorting order to begin with? In multi-dimensional systems and in a non-model setup that is usually very difficult to say. Hence, since you cannot predefine the correct order for all simulations, one often resorts to defining the erroneous orders. That means, depending on what you want to achieve you say “this data set has to be in the top 10” or “this data set has *not* to be in the top 10” and by this accumulate a set of quality rules to benchmark your solutions. Note that these rules can also be defined in an automated way to save time in the setup of a data model for quality.

One might notice that this process of accumulating quality rules is in itself fundamentally different from ML approaches of measuring the quality by use of training and test data (see *also Differences of Genetic AI to Machine Learning*). The reason is that Generic AI is not “trained” beforehand to reproduce certain categories of data or probabilities in one central model. “Training” the AI means running the evolutionary simulation and this is usually done in real-time for a specific user or application. Two or more simulations are not connected to each other in the general case¹⁶(*though Generic AI can learn, see below*). Consequently, splitting data into train and test makes no real sense for Generic AI.

From a bird-like perspective it is not surprising that we have to reconsider the way we measure AI quality, since the approach of Genetic AI is so much different from existing AI¹⁷. Researching different ways to benchmark results is hence a way to expand the flexibility and range of this new technology.

AI Learning

How does Genetic AI learn and get better? The principle idea is that you learn to prepare the setup and the environment of the evolutionary simulation. There are in principle three ways to do that (*compare also 6.5 Customizing your Genetic AI - Personalization*):

Changing the initial values for gene fitness: one can make the starting fitness very unfair and give certain genes and organisms an advantage/disadvantage in the evolutionary simulation. The choice of initial gene fitness can be done per user and application – this is a very easy way that Genetic AI learns to better match a user’s demand.

Changing the gene and organism strategies: Genetic AI may learn to match the used strategies to the required behavior. Again this can be done on a per-simulation basis – the effect of this can however be quite severe.

Improving the distillate: for soft genes, where the system information is stored mainly in the distillate, one can learn expanding the distillate with new data sets and correlations.

The actual process of learning can be done manually, semi-automated or fully automated. Manually meaning that you observe quality benchmarks and adapt parameters and strategies. Semi-automated learning may consist of daily reports with options to improve the quality of your simulations. Fully automated may include 2 different sets of initial values and distillates that “compete” each day in terms of a chosen KPI.

¹⁶ Note that this is a nice analogy to the biological situations that two distinct habitats with the same species can show completely different evolutionary behavior.

¹⁷ In (evolutionary) game theory the “result” of the game, i.e. the distribution of strategies, is the “right” answer per definition. This is the case since one has created the game to learn something about (evolutionary) systems by means of game theory. There is no one to stop you when you compare the distributions to real-world evolutionary systems (e.g. the behavior of males/females in a population). This is however done on a per-system basis - and not automated for abstract data as we need it.

Complexity & Performance

We have learned that Genetic AI provides a versatile way to sort general data. But is it fast? In numerics one measures the complexity of a problem also in terms of how many computations are needed to obtain a result.

In Genetic AI, in order to compute the Deltas

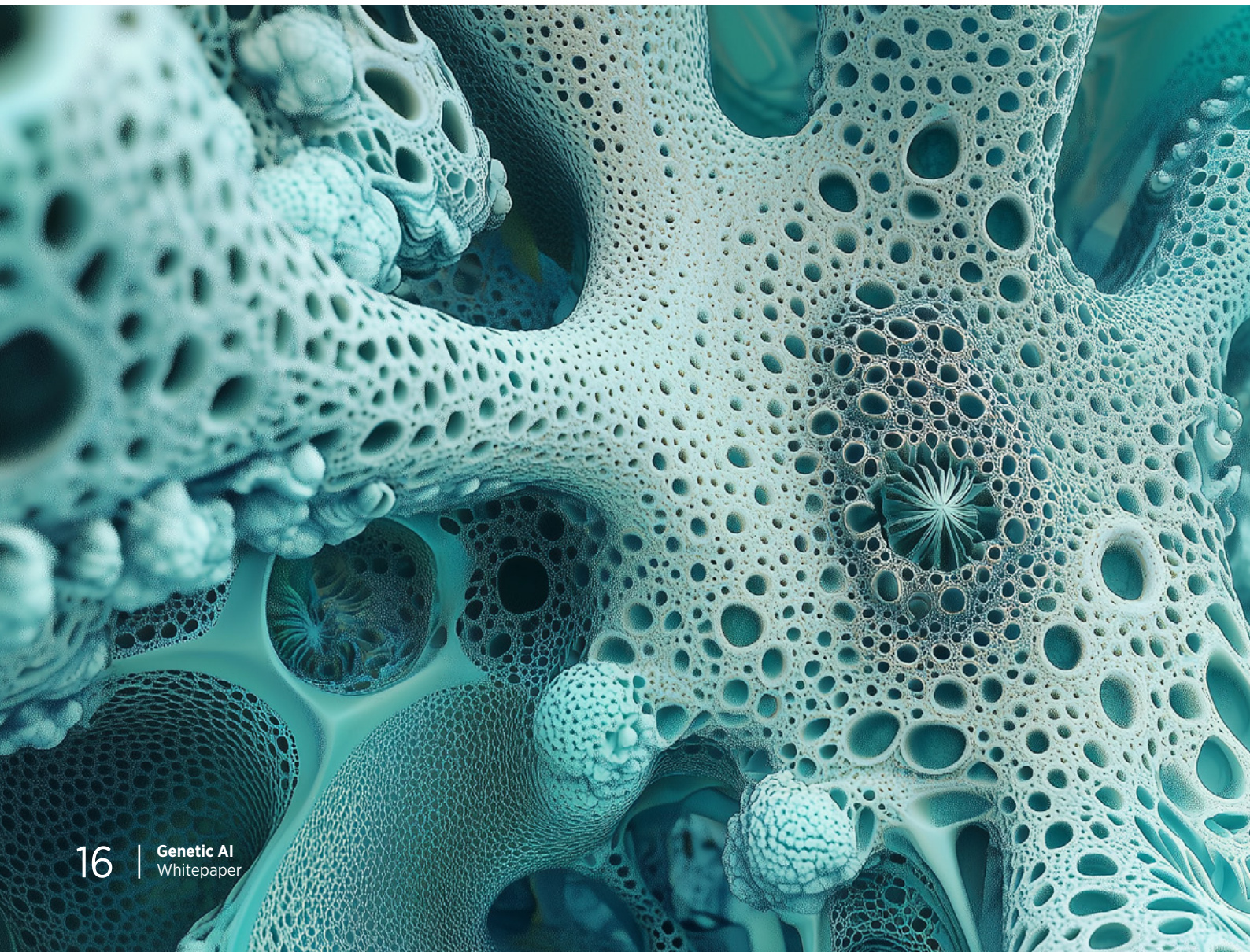
$$\Delta_{ij}^{g,(k)} = G^{GS}(f(X), \gamma^{(k-1)}, \dots, \gamma^{(0)})$$

$$\Delta_{ij}^{\omega,(k)} = \Omega^{OS}(f(X), \gamma^{(k-1)}, \dots, \gamma^{(0)})$$

you roughly have to do 1-5 computations per data element (gene variant) for each Delta, depending on the chosen strategies. If you have n data sets and m data features this means you have approximately $8 \cdot n \cdot m$ number of computations per iteration.

An organ usually has up to 20 data features at the maximum. The number of data sets n , however, can become very large. In this case, we usually prefilter data sets according to important features to reduce this number below 1000. Thus, we get to around 160.000 calculations per iteration which results in a computation time of milliseconds.

An important aspect of Genetic AI is that very often 1-2 iterations are enough. On the one hand that is because we are sorting - it is enough to know whether organism A comes before or after B (see also Getting the right flight, revisited). On the other hand that is because the general trend in gene fitness can be extrapolated and one obtains reasonable estimates for many applications.



Differences of Genetic AI to Machine Learning

As a new technology, it makes sense to compare Genetic AI to the most successful approach currently available: Machine Learning. Note that this is not straightforward, since the two technologies do not even use the same language when it comes to training or learning. Hence, let us compare the technologies in four domains: data, structure, training and goals:

Data

ML usually analyzes large data sets for patterns and correlations. Genetic AI is agnostic when it comes to data: you can input three data sets, but you could also input one million. Additionally, there is no central data model in Genetic AI, but every simulation is a new, independent game.

Structure

In ML one usually has a sophisticated training phase at the beginning, where a large amount of computational resources are invested to understand the data. Afterwards, the application of the trained AI model on new data is often much more lightweight than the training. Contrary, in Genetic AI, there is no preliminary training phase. Everything training and applying the AI model happens at the same time.

Training

in ML one has to take care to correctly select the right training data to cover the demand of the AI problem. Problems like overfitting and bias are difficult to control since the “effect” of a single data set in training cannot be backtraced. In Genetic AI there is naturally no overfitting since the to-be-sorted solutions are the only possible “source” of results. Preventing bias is connected to the evolutionary strategies used. In general, it is very difficult to create a biased strategy that is stable for all kinds of data. Thus, bias would only come into play when one would try to “force” it into Genetic AI. Ironically, that is exactly the opposite situation as in ML where one tries to “force” the bias out of the AI model.

Goals

in ML one tries to find patterns and categories in data. In a sense, the aim is to spice the data with a “meta-structure” that can then be used to solve all kinds of problems. In Genetic AI, one tries to understand the “nature” of data, meaning to find the underlying behavior and correlations. By differentiating relevant data features from less relevant traits one obtains a deeper understanding of the data space in question.

Apart from differences there of course lies a lot of potential in letting ML and Genetic AI work together. One example of how to do that is to let Generative AI generate solutions to a problem and afterwards allow Genetic AI to sort out the right one. In this sense an LLM could be a dedicated organ in a multi-organ setup of Genetic AI. *(see also 11.6 Full Multi-Organ*

Interconnected System)

Practical Applications of Genetic AI

In this paper, we have seen that Genetic AI provides a very versatile way to solve MDS problems in a universal data regime. But, in practice: where can it be applied? In this section we give a few examples how Genetic AI can be used with minimal adaptations. Note that this list is by no means exhaustive in any sense.

11.1

As Recommendation Engine

Imagine you have a pool of products (or services) you want to show users. Which product(s) do you present to which user? Genetic AI offers a straightforward way to sort the offers according to the users preferences.

In these kinds of applications, one predominantly uses the initial values to personalize the final sorting results. This has the advantage of being very flexible, stable and fast. Note that you usually need some kind of prefiltering to prepare an evolutionary system of feasible size.

The use of Genetic AI as a recommendation engine has several advantages:

- You can directly take the available pool of products that are currently available and sort it individually for the user. By doing this you circumvent the need of product categories and complex AI models to predict the interests of the user.

- As a welcome side-effect you rule out any bias or overfitting. These drawbacks find their way into ML systems, since they compute the AI model from a “separate” set of training data beforehand. In the end, they have to take care that the training “world” and the real “world” do match to a high degree to each other. In Genetic AI you do not have a preliminary training phase – hence, you will not meet these problems at all.

- Since the individual evolutionary simulations are mutually independent, you do not need any personal user data except for the user history.

- The choice of the right evolutionary strategies can usually be easily derived from the field of products or items to be sorted.

11.2

As Decision Engine

Think of a personal situation where you had multiple choices and a lot of decision parameters. A common approach is to make a pros and cons list and then weigh the arguments thinking about the consequences. With Genetic AI, you can do this in a much more structured and automated way for universal decisions.

The magic behind Genetic AI as a decision engine: the only thing that is needed is a principle understanding in the nature of the decision. With this, the optimal choice of evolutionary strategies is usually straightforward.

11.3

As Search Engine

Search engines paved the way for the growth of the internet in the first place. Lately, they have begun to struggle with the amount of data and with issues of personalization.

Since finding the right search results is nothing else but sorting data sets, Genetic AI shows great potential as a search engine. One big advantage is that you can easily personalize the results in a decentralized manner - directly at the device of the searching user.

11.4

As Discovery Engine

A discovery engine is like a mixture of recommendation and search engine. It shall show you what you are looking for - and much more than that. Discovering new things and decisions allows users to learn to treasure the available pool of items. Genetic AI allows to create this experience without any bias and hidden agendas.

11.5

As Prediction Engine

One application of Genetic AI that is not in the principle focus of this paper, is the prediction of things. However, it is only a small step from understanding the nature of a data system and predicting how it will evolve. Note that AI models for predicting need special care in terms of data features and quality. Additionally, choosing the right GS and OS strategies might not always be as straightforward as for other applications.

11.6

Full Multi-Organ Interconnected System

Towards the end of our introduction to Genetic AI, let us accumulate all functionalities and take them all the way to the end, outlining a complex multi-organ system. For multiple organs to work together they first need very specialized roles and strategies. Second, they have to be interconnected in a way such that they act like an intelligent “orchestra” of evolutionary simulations.

As a first step to plan the system, one has to group the genes in a meaningful way: hard genes that describe similar traits are accumulated into one organ. Analogously it works for soft genes¹⁸. In our system to choose the right flight, the given grouping of the hard genes price, time and stops represents a straightforward example for a simple organ “Hard flight data”. Another organ could be “Airline Satisfaction” covering the user data of airline information as a soft gene (which user liked/dislike travelling with which airlines).

Both organs provide a list of organism fitness – one for “Hard flight data” and one for “Airline Satisfaction”. Both lists are sent to a third organ “Complete Analysis” as genes. After the evolutionary simulation in the organ “Complete Analysis”, one obtains a final list of organism (=flight) fitness intelligently combining “hard” flight data with “soft” airline satisfaction. Note that all organs will in general use completely different evolutionary strategies. They do not “know” anything about the later use of their fitness output and will just “try” to solve the local data problem in the optimal way.

There are essentially no boundaries on the number of organs that can be used in Genetic AI. Comparing the concept of “organs” to “layers” in Neural Networks, one can observe that organs allow for a more direct and transparent control in between input and output data. On the other hand, Genetic AI multi-organ systems are currently built for a specific application and cannot be applied freely to such a wide range of problems as for neural networks. However, the evolution of Genetic AI is just at the beginning.

¹⁸ The reason why hard and soft genes are grouped in different organs would go beyond the scope of this paper. In a nutshell, hard genes are much less dynamic and can be safely described with relatively simple evolutionary strategies. For two soft genes it is difficult to describe them with one strategy so they often end up alone or with a single, similar soft gene partner.

Our AI Requirements revisited

We started our analysis by listing a set of must-have requirements for the AI of our dreams. We are now ready to revisit our rules and reflect on if and how Genetic AI fulfills them:

1

Stability

Since Genetic AI “identifies” the nature of data, it is very stable for most applications. Results and consequently answers to problems hold firm even if single data sets are taken out or changed. The two main reasons for this: (i) sorting algorithms have a higher grade of stability than other technologies per definition and (ii) the evolutionary behavior of many strategies is relatively universal.

2

Decentral Data and Model

There is no central model for Genetic AI, since all the to-be-sorted data sets are just taken as-is into the simulation. For soft genes an anonymized distillate is taken e.g. containing the user experiences of the system. In principle, this distillate can be distributed among peers, can be updated decentrally and does not have to be the same for all users in the system.

3

Follow-the-rules

Since Genetic AI does not generate things, one can simply restrict to data items that follow the rules. Additionally, one can use organs that push items with high ethical standards for example.

4

Transparent

In Genetic AI, all intermediate steps of the analysis can be investigated and clearly understood. Without any “hidden” layers or “black-box” components, transparency boils down to how to show the reached data understanding to the user in terms of UX strategies.

5

Non-conformist

To create a “conformist/uniform” Genetic AI has never been tried, but we assume that this is not ruled-out by the technology itself. Our approach to create very diverse results is to take evolutionary strategies that favor non-average behavior. However, to use Genetic AI in the right, diverse way in the end depends on the individual or group applying it to a problem.

6

Low Resources

As we have seen in the *section 10. Complexity & Performance*, Genetic AI uses very limited resources. In terms of user benefit per invested resource, Genetic AI is clearly superior to many established technologies.

Outlook

As long as there is life on this planet, there is evolution¹⁹. Apart from physical and chemical boundaries, evolutionary processes have primarily shaped the habitats and societies around the globe. For us it thus seems natural to apply evolutionary concepts to universal data analysis and artificial intelligence – what can possibly go wrong?

The “evolution” of Genetic AI itself began in 2018, when the first ideas started to crystallize. Still, only a very limited number of people have shaped the technology so far. It is our vision that this paper helps to spread the “seed” of Genetic AI. We imagine new evolutionary strategies and new multi-organ approaches to be created by other teams – and to

benchmark and compare them for an increased quality of simulation. In technical means, Genetic AI is nothing more than a toddler making its first steps.

We hope that this paper also leads to different perspectives on AI in general. What kind of AI do we want? Will it lead us to a better future as individuals and societies? In the end, Genetic AI is yet another technology – it is neither good nor evil. It is the people that are behind the machine that make the difference.

Acknowledgement

A big thanks to Yiguang Wang for useful feedback and discussions. I am grateful that Gabriele Schwab reviewed parts of the paper.

A huge cheers to Denise Korenjak for bringing the paper in this wonderful shape.

[1] https://en.wikipedia.org/wiki/Sorting_algorithm

[2] https://en.wikipedia.org/wiki/No_free_lunch_theorem

[3] Gould, S. J. (1989). Wonderful life: the Burgess Shale and the nature of history. First edition. New York, W.W. Norton & Company.

[4] https://en.wikipedia.org/wiki/Evolutionary_algorithm

[5] Shumeet Baluja (1995) Removing the Genetics from the Standard Genetic Algorithm, Machine Learning Proceedings 1995

[6] https://en.wikipedia.org/wiki/Side-blotched_lizard

[7] Richter, X. L. and Lehtonen, J. (2023). Half a century of evolutionary games: a synthesis of theory, application and future directions. Phil. Trans. R. Soc. B 378

[8] Leimar O, McNamara JM. (2023) Game theory in biology: 50 years and onwards. Phil. Trans. R. Soc. B 378:

[9] Dawkins, R. (2006). The Selfish Gene. Oxford University Press.

Getting the right flight, mathematically revisited

Let us investigate why certain data features (genes) are up- or down-voted by the combination of GS Dominance + OS Balance. With the raw data in mind

Raw Data	Price (Euro)	Time-of-transfer (h)	Stops
Flight A	300	10	2
Flight B	600	5	2
Flight C	1500	4	1

let us apply our local fitness functions to the gene variants

Fitness Data	Price (Euro)	Time-of-transfer (h)	Stops
Flight A	0.8	0	0
Flight B	0.6	0.5	0
Flight C	0	0.6	0.5

Note that we used a simple inverse fitness function to all genes which scales the entire data interval to $[0, \min(\text{data feature})]$. Using another local fitness function would lead to different results²⁰.

The important thing about GS Dominance is that it “measures” the asymmetry of a gene with respect to 50%. This

becomes clear if one adds up the individual contributions of the gene variants (for the first iteration of the all-33% initial gene fitness example)

$$\Delta_j^{g,(k=0)} = \sum_{i=1}^n \Delta_{ij}^{g,(k=0)} = [-0.03, -0.1, -0.11]$$

with all Deltas being recessive, but the first gene “price” still with the weakest effect. This is because all mean values of the genes $[0.47, 0.37, 0.17]$ are below 50%.

Let us investigate the effect of OS Balance on the gene fitness. Accumulating the Deltas analogously to above we find

$$\Delta_j^{\omega,(k=0)} = \sum_{i=1}^n \Delta_{ij}^{\omega,(k=0)} = [-0.04, -0.01, 0.06]$$

with the first gene being too dominant, the second gene balanced and the third gene being too recessive, respectively, from the organism perspective. The recessiveness of the third gene can be understood since it contributes nothing to the fitness of flights A and B.

Taking both contributions together one can conclude that for the third gene both strategies “pull” in opposite directions while the second gene gets diminished by both contributions, genes and organisms, respectively. The biggest effect however is created by OS Balance for the third gene “stops”.

Getting the right flight, data interpretation

Why does the data feature “stops” come out as the most important in our flight example? In terms of local fitness function we have seen that flights A and B (i.e. the majority of flights) have a value 0% on that gene. At the same time, flight C is the only data set to have the value 50% for the gene “stops”.

In terms of uniqueness of a data feature the “stops”-feature is thus the most relevant, since only one data set has a non-zero value. In the setup, Genetic AI hence surmises that the data feature “stops” should be especially “treasured”. We can test this argumentation by changing the raw data for flight B:

Fitness Data	Price (Euro)	Time-of-transfer (h)	Stops
Flight A	0.8	0	0
Flight B	0.6	0.5	0.5
Flight C	00	0.6	0.5

Note that now also flight B has just one stop. Running one iteration of Genetic AI one obtains

$$\Delta_j^{g+\omega,(k=0)} = [-0.6, -0.9, -0.9]$$

meaning that the gene “stops” behaves like the gene “time” now since we have removed the uniqueness.

²⁰ One possible adaption would be to not use a linearly decreasing fitness with larger price, since customers usually interpret a price twice as high as much worse than twice as bad.