# Agent Based Modeling
# — ODD —

Berk Can Özmen - 396694

January 16, 2025

## 1 Purpose and Patterns

Given survival criteria, based on natural resources that can be produced by agents and local market dynamics, and strategies that are evolved through genetic algorithms, creating simple market dynamics, where prices of goods are decided by the environment and the strategies of the agents

## 2 Entities, State Variables and Scales

### 2.1 Agent Variables

| **Variable** | |
|---|---|
| genome | A string that encodes 4 neural networks |
| age | age of the agent |
| inventory | inventory of the agent, list of 3 numbers |
| needs | satisfied needs of the agent, list of 2 numbers |
| charm | charm of the agent. Used in mate selection (see consume) |
| produce-rate | produce rate (see produce) |
| exchange-rate-history | estimation of the local prices |
| amount-history | estimation of the local trade volume |
| exchange-rate | agents own pricing of goods |
| exchange-amount | agents willingness to trade |
| reproduce? | if agent can reproduce |
| number-of-childeren | Monitor variable to track how many childeren the agent has |
| net-produce | Monitor variable to track agents production |
| net-trade | Monitor variable to track agents net trade |
| net-trade-volume | Monitor variable to track agents trade volume |
| net-consume | Monitor variable to track agents consumption |

### 2.2 Patch Variables

| **Variable** | |
|---|---|
| elevation | Elevation of the patch |
| total-production-wish | Variable to keep track of total production investments in the patch by all the turtles on this patch |

## 2.3 Environmental Variables

| **Variable** | |
|---|---|
| Initial-population | Starting population size |
| Min-population | Minimum amount of population. If population goes under, new random agents will be spawned. |
| Max-age | Maximum age that an agent can live to |
| Genetic-diversity-add | How many random agents will be spawned each turn, to add some genetic diversity, if the diversity i |
| Fertility-start | The age that agents can start reproducing |
| Fertility-end | The age that agents become infertile |
| Nutrition-multiplyer | Each goods nutrition value will be multiplied by this value (in order to control the abundance of the |
| Mutation-chance | Rate of mutation for each bit in genome |
| input-lengths | Input layer widths of the networks |
| output-lengths | Output layer widths of the networks |
| network-layers | Hidden layers |
| networks | Network shapes (list of list of layers) |
| genome-length | Integer length of the genome |
| maximum-gene-bit | Random initialization range |
| gene-points | List of where each gene starts and ends |
| start-inventory | starting inventory [10 10 10] |
| start-needs | 2x min-needs |
| min-needs | minimum needs to survive, read by an input file |
| max-needs | maximum needs that can be satisfied |
| gdp-per-capita | Estimation of number of items in inventories |
| production-per-capita | Estimation of yearly production of items |
| consume-per-capita | Estimation of yearly consumption of items |
| trade-per-capita | Estimation of yearly trade volume |
| prices-per-capita | Estimation of prices |
| death-by-starvation | Counter for how many people died of starvation in a year |
| death-by-age | Counter of how many people died of old age in a year |
| new-borns | Counter of how many people is born in a year |
| death-age | Maximum age that an agent can survive up to |
| season | Season |
| max-product | Baseline production of each item, read by a .csv |
| season-efficiency-table | Season efficiency table, read by a .csv |
| good_table | Nutrition values of goods, read by a .csv |
| need_table | Min-needs, read by a .csv |

# 3 Process Overview and Scheduling

```
Set season
Add new random turtles for genetic diversity amount of genetic-diversity-add
If number of people are too low, clone new people from existing ones

ask people[
    Produce
    Exchange
    Consume
    Move
    Evaluate-survival
    Check-reproduction

    if reproduce?
        Mate
]
```

# 4 Design Concepts

## 4.1 Basic principles

Genetic algorithms are used to determine the agent strategies. Rest of the mechanics are hardcoded (how does production or survival interacts with the genome) and makes the main simulation.

## 4.2 Emergence

The production, consumption, exchange and movement strategies are expected to emerge. As a result a market economy is expected to emerge. That includes a pricing mechanism for each goods and a wealth distribution. Also for different agents, different combinations of production and trading strategies are expected to emerge.

## 4.3 Adaptation

Through natural selection, the unfit agents are expected to die, and the agents will be adapting to the environment indirectly. Through sensing their environment (crowdedness, season and elevation) they can also adapt their strategies.

## 4.4 Fitness

The agents do not have any explicit goals. However agents that cannot satisfy their needs by consuming goods (which can be acquired either through trade or through production), will die and agents that eats a balanced diet are more likely to reproduce. Thus an indirect fitness can be defined as surviving / satisfying their needs.

## 4.5 Learning

The agents do not explicitly change their behaviours. Their behaviours are dictated by their genome, which is expected to change through crossover and mutation as agents reproduce. Since every agent will die by aging or starvation, the genetic composition of the populations is supposed to shift, which is the implicit learning process.

## 4.6 Prediction

The agents do not explicitly predict their environment. The prediction is also dictated by their genome, which dictates their strategies. Thus the strategies that are more fit to the environment (thus can implicitly predict) are expected to thrive in the population.

## 4.7 Sensing

The agents can explicitly sense the season, their location (elevation), other agents and their internal variables. They will use this information to decide on their production, consumption, movement, exchange and mating strategies. They can also implicitly sense the behaviour of other agents, since their survivability depends on the other agents (because of the trade). However this is also implicitly governed by the genetic algorithms.

## 4.8 Interactions

Agents can trade with each other. Each agent will determine a price for each type of goods. Then according to their inventories, they will offer an exchange to other agents. Agents can also reproduce with other agents, where they choose their mate depending on their success. During reproduction, agents will share their genome and create a new agent randomly sampled from both of the parents genome.

## 4.9 Stochasticity

The agent's genome is initialized randomly, since the aim of the model is to come up with good strategies through evolution. During reproduction, the children's genome is randomly chosen from parents genome (to keep the good solution) and also mutated randomly (to allow exploration of new strategies).

## 4.10 Collectives

There are no explicit collectives of individuals. They all have unique strategies (unique genomes), and apart from them they are the same. However, through strategies collectives of individuals can emerge, such as farmers, fishers or traders. Since producing only one good is more efficient, and each good can be produced efficiently in different areas and agents need different types of goods this type of specification is expected. For example, a viable configuration could be, fishermans in low altitudes, farmers in high altitudes and traders that carry goods in between and live off the trade they are making.

### 4.11 Observation

In the current state, we track the production, amount of goods in each individual's inventory, traded goods and their prices, population, cause of death, new borns, average age of death and average number of children of each agent is tracked. These variables are tracked either directly from the global variables, or are computed through agents local variables.

## 5 Initialization

```
clear-all
reset-ticks

set monitor variables to 0
calculate genome length from given neural networks

ask patches[
    elevation = pxcor
]

create initial-population of people with:
    random xy-cor
    random genome
    random age
    random color
    inventory = start-inventory = [10 10 10]
    needs = start-needs
```

## 6 Input Data

`good.csv`
Determines the nutrition value of items

| | **Protein** | **Fat** |
|---|---|---|
| **Fish** | 1 | 4 |
| **Wheat** | 1 | 2 |
| **Meat** | 6 | 2 |

Table 1: Nutrition values of items

`need.csv`
Determines the min-needs of an agent $[30, 30]$

`season-efficiency.csv`
Determines the efficiency of which item can be produced in which season

| | **Winter** | **Spring** | **Summer** | **Fall** |
|---|---|---|---|---|
| **Fish** | 1.0 | 0.5 | 0.5 | 0.75 |
| **Wheat** | 0.0 | 2.0 | 2.0 | 0.0 |
| **Meat** | 0.5 | 1.0 | 0.75 | 0.5 |

Table 2: Season efficiency table

| | Fish | Wheat | Mean |
|---|---|---|---|
| **Base Production** | 18 | 20 | 8 |

Table 3: Base production values of items

`max-product.csv`
Determines the baseline production of each item

# 7 Submodels

## Agents

Every agent contains 4 feed forward neural networks (with the same number of hidden layers and neurons). and a genome, that contains 4 genes and encodes the weights of the neural networks. Every NN decides on an action (in order of the genome encoding), i.e produce, exchange, consume, move.

**Networks**
Each network contains 2 hidden layers with 8 neurons, with sigmoid activation.
**Produce NN**
inputs : Normalized coordinates, normalized inventory, normalized needs, season, number of turtles here
length : 9
outputs : Softmax - 3 neurons that encode the percentage of production of each item
**Exchange NN**
inputs : Normalized coordinates, normalized inventory, normalized needs, season, Local prices from last tick
length : 11
outputs : 1st neuron encodes how much to trade with sigmoid activation, 2-4th neurons encodes relative prices of goods with sigmoid function
**Consume NN**
inputs : Normalized coordinates, normalized inventory, normalized needs, season
length : 8
output : 1st neuron encodes how much to eat with sigmoid activation, 2-4th neurons encodes proportions of items to eat with softmax activation
**Move NN**
inputs : Normalized coordinates, season, crowd info of 5 neighbouring sectors (square with fog-radius)
length : 8
output : Heading and how much to walk, with sigmoid scaled with (360, fog-radius)

**Needs**
Every agent needs 30 protein and 30 fat to survive a tick. Then $min\_needs = [30, 30]$, $start\_needs = [60, 60]$, $max\_needs = [90, 90]$

| | Protein | Fat |
|---|---|---|
| **Fish** | 4 | 16 |
| **Wheat** | 4 | 8 |
| **Meat** | 24 | 8 |

Table 4: Nutrition values of items

Reproducing also costs $min\_needs$ amount of needs. Other than that, needs can boost/down production and the chance of reproducing.
**Production**
Every agent makes a decision (pmf) to how much invest in each type of production by using the neural network. This is the base of their production ($p$). Then this base rate is multiplied by elevation efficiency ($e$), season efficiency ($s$), health efficiency ($h$) to produce produce-rate ($pr$)

Then $pr_i = h \times s_i \times e_i \times p_i$
Then each agent competes at the patch they are on with other agents. So for example if there are 2 agents $(i, j)$ that want to produce $p_1^i = 0.2$ rate fish and $p_1^j = 0.3$, $pr_1^i$ get $ps_1^i = 2/5$ and $pr_1^j$ gets $ps_1^j = 3/5$ production share ($ps$) of the total production cap $pc$.
Then item gets by the agent is calculated as $= pr_i \times ps_i \times pc_i$

1. Health ($h$)
   Health is a efficiency constant that is decided on satisfied needs. It maps needs in term of 2*min-needs, by scaling linearly intervals $[0, 90] \rightarrow [0.5, 1.5]$ and means the result.

```
let health mean (map [ [a b] -> a / (2 * b)] needs min-needs)
```

2. Elevation Efficiency ($e$)
   Elevation efficiency decides on how efficient can each product be produced, given the elevation of the patch. The functions:

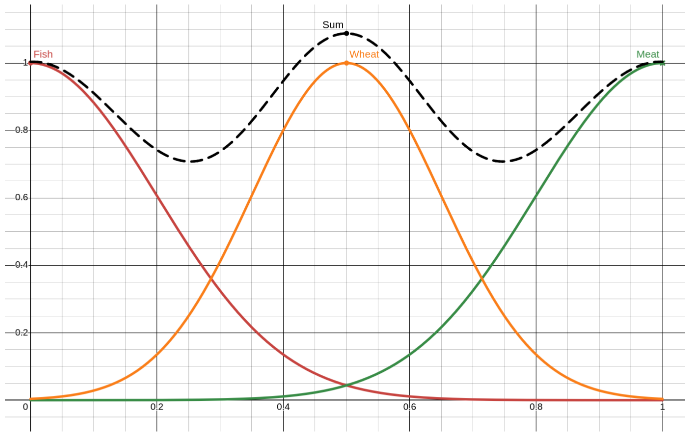


Figure 1: Elevation efficiency function

Season Efficiency ($s$)

| | Winter | Spring | Summer | Fall |
|---|---|---|---|---|
| **Fish** | 1.0 | 0.5 | 0.5 | 0.75 |
| **Wheat** | 0.0 | 2.0 | 2.0 | 0.0 |
| **Meat** | 0.5 | 1.0 | 0.75 | 0.5 |

Table 5: Season efficiency table

| | Protein | Fat |
|---|---|---|
| **Fish** | 1 | 4 |
| **Wheat** | 1 | 2 |
| **Meat** | 6 | 2 |

Table 6: Nutrition values of items

3. Production Cappacity ($pc$)

| | Fish | Wheat | Mean |
|---|---|---|---|
| **Base Production** | 18 | 20 | 8 |

Table 7: Base production values of items

**Consumption - Needs - Charm**
Every agent has 2 basic needs that it must keep satisfying, by consuming goods. Every tick, agent decides how much to consume (first neuron of the output) and the diversity of its diet (2nd-4th neurons of the output) proportions of items) to satisfy their needs. Then these needs are decrease by min-needs, and agent's satisfied needs cannot exceed max-needs (3 time min-needs).

Agents produce more if they satisfy their needs more. Agents get to choose their partner first, if they satisfy their needs more, and they consume min-needs amount of needs to reproduce. Lastly, consuming a more diverse diet makes agents more charming, which increases their chances of being chosen as a partner.

The first output neuron has a sigmoid activation function. 2nd to 4th neurons have a softmax activation function.

```
proportions = [2nd output, 3rd output, 4th output]
target_nutrition = 4 * first_output_neuron * min_needs

while target_nutrition is not achieved:
    items_to_eat = max(inventory, proportions)
    increase nutrition intake by calculate_nutrition(items_to_eat)
    decrease inventory by items_to_eat
    increase needs by calculate_nutrition(items_to_eat)
```

**Exchange**

In the output layer, sigmoid function is used. The first neuron decides how much to trade, and the rest of the neurons decides on the prices of the goods. Then agents make offers to other agents in the radius fog-radius. If the trade is beneficial for both agents (according to their pricing), the trade happens. The amount of the trade is scaled by the first output neuron.

```
to calculate-exchange-rate
  ;Calculate the price estimation by the records of the last tick
  set exchange-rate-history (map [[a b] -> a / b] exchange-rate-history amount-history)
  set amount-history [1 1 1]

  ;Use the NN
  let input (sentence (get-normalized-coordinates) (get-normalized-inventory) (get-normalized-needs) (
  let output sigmoid (neural-activity 1 input)

  ;Process the output
  set exchange-amount item 0 output
  set exchange-rate (map + (sublist output 1 4) [0.001 0.001 0.001])
end

to exchange
    Get the turtles in radius fog-radius
    Ask each turtle:
        For each item pair (i,j):
            Calculate exchange rate of items
            Calculate the maximum trade amount of each item
                regarding the exchange rate.

            Scale maximum trade amount by the min(first_neuron of agents)

            If (selling i and buying j is profitable for me
                    and buying i and selling j is profitable for the partner:)

                Do the trade
end
```

**Move**

In the output layer, sigmoid function is used.

```
to move
  ;use the NN
  let input (sentence get-normalized-coordinates get-normalized-season get-crowd-info)
  let output (neural-activity 3 input)

  ;process the output
  let head ((item 0 output) * 360)
  let forw (item 1 output * fog-radius)

  ;execute the command
  set heading head
  fd forw
end
```

## Genetic Algorithm

**Reproduction Fitness**
Every agent needs to satisfy 2× the minimum needs to be able to reproduce. Furthermore, their fitness is calculated as the sum of their satisfied needs

**Mate Selection**
Agents are sorted by their fitness. And starting from the fittest agent, agents selects their partners. A charm score is calculated, based on the diversity of the agent's diet (TV distance to uniform distribution). Every agent selects the most charming partner in radius fog-radius. After mating, both parents lose [min-need] amount need. Then they can again be eligible to reproduce if they are chosen by another agent as partner and still satisfy the reproduction condition.

**Crossover**
The crossover is based on 4 genes. Genes are never mixed (not to destroy the Neural Networks). Each child gets one gene from one parent with %50 of chance.

**Mutation**
With mutation-chance probability, a random bit from the genome is selected, and added a white noise with std = 0.2

**Child**
The child inherits the mean of their both parents family-color. Furthermore, the child starts with start-inventory and start-needs (which technically creates new goods out of thin air). An inheritance system can also be used, where parents lose equal amount of goods, however it causes less children or premature death of parents after birth. The child also starts in a circle of radius 1 from one of the parents.