



ABSOLUTDATA

An Infogain Company

BRAIN WAVE

DATA SCIENCE DIGEST

— 7TH EDITION —



Abhinav Gupta

Program Manager, Digital & Data Sciences (Data Analytics & AI), Absolutdata

Game Theory – Improve the Play

“Game theory is about exploring of freedom of choices and the equilibrium which comes from understanding the consequences of freedom.”

–Vineet Raj Kapoor

“Game theory was originally developed by the Hungarian-born American mathematician John von Neumann and his Princeton University colleague Oskar Morgenstern, a German-born American economist, to solve problems in economics. In their book *The Theory of Games and Economic Behavior* (1944), von Neumann and Morgenstern asserted that the mathematics developed for the physical sciences, which describes the workings of a disinterested nature, was a poor model for economics. They observed that economics is much like a game, wherein players anticipate each other’s moves, and therefore requires a new kind of mathematics, which they called game theory. (The name may be somewhat of a misnomer—game theory generally does not share the fun or frivolity associated with games.)” – Encyclopedia Britannica^[1]

Game theory is a concept with applications far beyond the theoretical. Its unique modus operandi comes from the fact that it gives an equal opportunity to each one of the players/agents involved in a game/operation. Players carve out their best strategy to maximize the results/rewards for themselves while considering their opponents’ tactics; this gives a result that considers the reasonable decision-making of all those involved and may be the most favorable outcome for all.

Modern-day applications of game theory are spread across a variety of industries and sectors. For example, take marketing a product. An advanced competitive analysis would strongly recommend considering what the competition is offering across that product line as a factor in optimizing cost and profit.

Game Theory Components

As depicted below, key considerations in game theory include:

- **Consumers/Decision Makers:** Within the context of the game, they are the actors or players.
- **Rewards:** The benefit(s) any player receives from the outcome.
- **Results:** The outcome derived at the end of all favorable/optimal strategies.
- **Strategies:** The unique set of actions any player takes within the game (which may be situational).
- **Decision & Interdependent:** The choices each player can make independently (decisions) and the choices that are influenced by other players’ actions (interdependent).

- **Nash Equilibrium:** The state when “each player is assumed to know the equilibrium strategies of the other players, and no one has anything to gain by changing only one's own strategy[...] If each player has chosen a strategy – an action plan based on what has happened so far in the game – and no one can increase one's own expected payoff by changing one's strategy while the other players keep their unchanged, then the current set of strategy choices constitutes a Nash equilibrium.”^[2]



An Example of Game Theory

The most famous example of game theory is the Prisoner's Dilemma:

Two prisoners must choose between staying silent or confessing to a crime. There are three scenarios:

1. Both prisoners stay silent and go to prison for a year.
2. Either one confesses and the other stays silent; the silent prisoner goes free, while the other gets 20 years in prison.
3. Both confess and each gets 5 years in prison.

The only condition is that the prisoners can't communicate with each other. Thus, both confess – driven by fear of being penalized for the other person's action. The only winner is the jailer, who applied game theory to amplify his result.

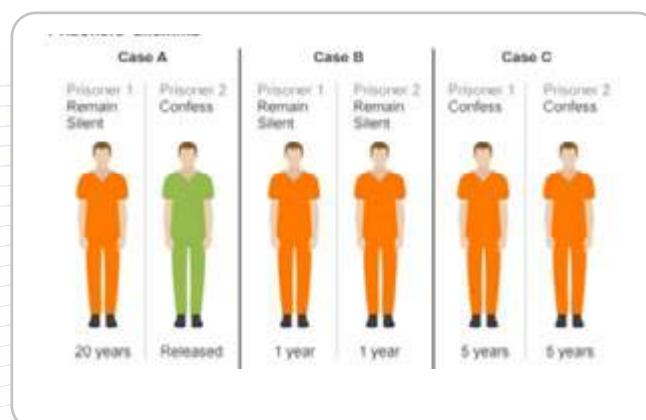


Image: [bbc.com](https://www.bbc.com)

Game Theory Applications in Business

Real-world applications of game theory include predicting the behavior of the various participants in a scenario where the action of one depends on the action of others. Some common scenarios include:

- Deciding on a product valuation.
- Buying or selling on the stock market.
- Launching a new product.

References



1. "Game Theory", Encyclopedia Britannica, <https://www.britannica.com/science/game-theory>



2. "Nash Equilibrium", Wikipedia.org, https://en.wikipedia.org/wiki/Nash_equilibrium



3. "Can game theory explain the Greek debt crisis?", BBC News, <https://www.bbc.com/news/magazine-33254857>

Contents

1	Statisticionary Modern Applications of Game Theory	Page 06
2	Coder's Cauldron DyPy: Interface to Backward Dynamic Programming	Page 09
3	Vivid Visualization Using Graphs to Illuminate Analytical Problems	Page 14
4	Thriving Traction Pattern Clustering Using Cooperative Game Theory	Page 18
5	Folk Wisdom's Fallacy The Mythology of Game Theory	Page 22
6	Experience Extended Interpreting Machine Learning Models Using Shapley Values	Page 24
7	Food for Thought Experiment How AI Could Improve Equality and Productivity in Tax Policies	Page 28
8	Data Science Competitions/Seminars/Fora/Courses	Page 33

Modern Applications of Game Theory

Statistictionary

What Is Game Theory?

It is a branch of applied mathematics that deals with the analysis of situations involving parties (called players) that make interdependent decisions. The formal origin of game theory can be attributed to mathematician **John von Neumann** and economist **Oskar Morgenstern**. In their book, *The Theory of Games and Economic Behavior* (1944), they asserted that game theory is better suited to describe economic behavior than physical sciences.¹ But the seeds for the discipline could be found alongside the emergence of probability theory back in 1654.²

What Does a Game Theory Problem Look Like?

Just like a game can be described by its characteristics of the **number of players** (1,2...n players), **information sharing between players** (perfect or imperfect), and **conflict of interest between players** (constant sum or variable sum), game theory problems and the subsequent approaches can be distinguished by the underlying parameters.

The following is a classic example of a two-player constant sum game with imperfect information – i.e. two competing candidates in an election must decide on how to react to a pressing issue to maximize their vote share. Each candidate can choose to support, oppose, or evade the issue. Depending on what each candidate chooses, the payoff matrix looks like this:

		Candidate 1					
		Support		Oppose		Evade	
Candidate 2	Support	60%	40%	20%	80%	80%	20%
	Oppose	80%	20%	25%	75%	75%	25%
	Evade	35%	65%	30%	70%	40%	60%

Table 1: Payoff matrix for two-player constant sum game with imperfect information

The rational decision both players should choose involves maximizing their own vote share against any and every action of the opponent. This can simply be achieved using either the **minimax** or **maximin** approach.

Here is the solution using the **minimax** approach: For Candidate 2, the minimum vote that can be achieved for each action is 20% (Support), 25% (Oppose), and 30% (Evade). The maximum of those is 30% (corresponding to evasion). The rational decision for this candidate would be to evade the issue. Similarly, for Candidate 1 the rational decision would be to oppose the issue.

This type of solution is called a **saddle-point**, where the maximum of the column is the minimum of the row (or vice-versa). It derives its name from the shape of a saddle and while it may or may not exist in the game of imperfect information, it always exists in games of perfect information. In the game of perfect information, payoff at this point is the value of the game.

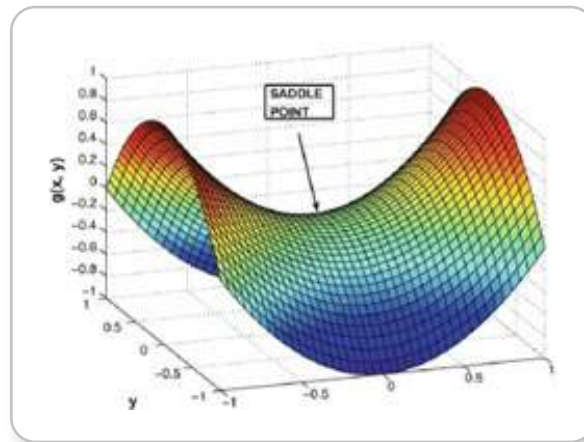


Figure 1: A saddle point ³

The Nash Equilibrium

Another aspect that can drastically affect the outcome or the strategies chosen by the players is whether the game is cooperative or not. **Cooperative game theory** describes (at a high level) the structure, strategies, and payoffs of coalitions in cooperative games.⁴ For **non-cooperative games**, the Nash equilibrium defines a strategy profile, a set of strategies (one for each player) so that no player can do better by unilaterally changing their strategy given that the strategy of other players is fixed.

Formally, let S_i be the set of all possible strategies for player i , where $i = 1, \dots, N$. Let $s^* = (s^{*1}, s^{*-i})$

be a strategy profile, a set consisting of one strategy for each player, where s^{*-i} denotes the $N-1$ strategies of all the players except i . Let $u_i(s^{*i}, s^{*-i})$ be player i 's payoff as a function of the strategies. The strategy profile s^* is a Nash equilibrium if

$$u_i(s^{*i}, s^{*-i}) \geq u_i(s_i, s^{*-i}) \quad \forall s_i \in S_i \quad ^5$$








If the equality doesn't hold, the strategy profile is known as a strict Nash equilibrium; otherwise, it is termed a weak Nash equilibrium.

Modern Applications

Game theory is used in conjunction with a variety of other disciplines; the applications depend on the complexity of the problem. Some of the most notable ones are:

1. Price wars in oligopoly markets.
2. Analyzing and designing political campaigns (as seen in the example in this article).
3. Auction algorithms and analysis. These are extensively used for ad bidding by search engines. This is an example of algorithmic game theory.
4. Cost-sharing algorithms and analysis. These are commonly used in building inter-community resources.
5. A flights assignment model based on zero-sum sequential game and CDM mechanisms.⁶
6. Dynamic pricing strategies for demand-side management in smart grids.⁷

References

-  1. [Encyclopedia Britannica: Game Theory](#)
-  2. Apostol, Tom M. Calculus, Volume II ["A Short History of Probability"](#)
-  3. [Example of Saddle Point](#)
-  4. [Wikipedia: Cooperative Game Theory](#)
-  5. [Wikipedia: Nash Equilibrium](#)
-  6. Liu, Junqiang. [Flights Assignment Model of Multiple Airports Based on Game Theory and CDM Mechanism](#)
-  7. Srinivasan, et al. [Game-Theory based dynamic pricing strategies for demand side management in smart grids](#)

Authored by

[Aishwarya Kukrety](#),

Analyst at Absolutdata

RDyPy: Interface to Backward Dynamic Programming

Coder's Cauldron

DyPy's goal is to provide an interface to backward dynamic programming that supports the following priorities (in order):⁽³⁾

1. Ease of learning and use.
2. Flexibility (can be adapted to new problems).
3. Speed (once the previous goals are met).

Dynamic Program

The Dynamic Program class is the core of DyPy. Each problem you wish to solve will involve creating an instance of this class and attaching the classes below to it in ways that tell it how to solve your problem.

⁽³⁾ DyPy should be able to handle problems with numerous state variables, which is a significant design consideration.

Dynamic Program manages all data and the flow of the optimization. By default, it will build all the stages and manage their tables, but this part of the process can be customized as well.

Objective Functions

The objective function will do some of the heavy lifting for your dynamic program and must be created by the user for each specific optimization problem. In each stage of the optimization, DyPy will call the objective function for each combination of state variables and stage variables; the objective function must return the cost or benefit value for that set of inputs. The objective function will be provided access to the Stage object for the stage it is currently evaluating, as well as the values of all the state variables and the decision variable. These will be provided as keyword arguments to the objective function.⁽³⁾

Stage

A stage provides the set of potential states and decisions at each modelled point (sequential moment) in the dynamic program. Most classes eventually tie to either the Dynamic Program class or the Stage class, which does most of the heavy lifting and data management in this package. The Dynamic Program class automatically creates and handles stages by default, but you can alter this behavior for more complex scenarios.⁽³⁾

State Variable

State Variable objects provide options for potential future conditions. A [Dynamic Program](#) can involve multiple state variables – in which case, all permutations of all state variable values are evaluated.

Be careful, because the solution space can quickly grow as you add more state variables with more options. A State Variable should have a name and a set of potential values. By default, the potential values can be generated for you if you provide a minimum value, a maximum value, and the discretization size of steps in between.

Decision Variable

Decision Variables describe potential choices that can be made at each stage. Like State Variable objects, they have names and values. However, they are managed slightly differently. DyPy currently only supports a single decision variable; several decision variables might theoretically be included, with increasing complexity to both code and solution time. Both Decision Variables and State Variables are provided to the objective function to determine the value of each potential decision when the system is in a certain state. ⁽³⁾

Prior

Priors are used in DyPy in two ways, each referring data from a previous stage that should be included in the present stage. This requirement arises both during the backward matrix formulation and the forward path calculation. The **dypy.Prior** techniques of applying future stage values to older stages are provided by the prior class and subclasses. **dypy.SimplePrior** has a single-variable implementation that may or may not work for multi-variable issues. To offer a new implementation, this class can be subclassed and the apply method overridden. The new matrix should be returned by the apply method.^[3]

By default, the Prior class to be used should be provided to the Dynamic Program upon creation, but they can also be overridden per-stage in case of a need to apply priors differently at different stages.

Reducer

Reducers are still to be implemented; they provide a tool for turning multi-state variable problems into single state variable problems before calculating the best path. One state of a stochastic dynamic program may be determined by your decisions, while other states are determined by probabilistic future events. Reducers can help reduce the probabilistic states so that a single state variable reflecting the needs of the decision can be used for the forward optimal path calculation.

Use of reducers is not required, and those with need for a true stochastic dynamic program may wish to implement branching behavior reflecting the uncertainty in future stages. The Stage and Prior classes would then need to be overridden to provide such behavior in lieu of using reducers.

Programming Effort

You're a programmer working hard to triage bugs in your software before a big release coming up. In your bug tracker, bugs are split into 5 categories: "Core", "UX/UI", "Network", "Database", and "API". The release is in 12 days, but your team manager wants to make sure to get some fixes in each. They ask you to spend no more than 5 days on any single area and at least one day on each, but other than that, they just want you to fix the most bugs possible.

Considering the prioritization of tickets in the tracker, you estimate the number of bugs you fix in each category as a function of time as follows:

Bugs closed by days of effort in each area

Category	1 Day	2 Days	3 Days	4 Days	5 Days
Core	2	5	7	8	10
UI/UX	3	6	8	9	9
Network	1	2	4	7	9
Database	2	3	6	8	11
API	4	6	8	9	10

Code

```
# we have 12 days available to us for work
state_variable = dipy.StateVariable("Days Available for Category", values=range(1, 13))
# but we can only spend between 1 and 5 days working on a single category
decision_variable = dipy.DecisionVariable("Time on Category", options=decision_options)
def objective_function(stage, days_available_for_category, time_on_category):
```

```
"""
```

When the objective is called, the solver makes the stage, state variables, and decision variable available to it. The keyword argument names here match the names of the state variables and decision variables (which will be passed as keyword arguments; the order isn't important, but the name is). The name is automatically derived by lowercasing the name of the variable and replacing spaces with underscores. It will remove digits from the front if they are present so that the name is a valid python identifier

In this example, we can think of these variables as providing:

1. stage - this will give access to the `DyPy.stage` object
2. state - this just provides access to the state value being assessed, not the object
3. decision - this just provides access to the decision value being assessed, not the object

```
"""
```

```
# we are given benefits, so defining here - each category is a row, each column is a
# number of days, and the value is how much benefit we get from working on that category
# for that long
benefit_list = [
    # column 0 means no time spent on it, so we get no benefit
    [0, 2, 5, 7, 8, 10],
    [0, 3, 6, 8, 9, 9],
    [0, 1, 2, 4, 7, 9],
```

```

    [0, 2, 3, 6, 8, 11],
    [0, 4, 6, 8, 9, 10],
]
# can't spend more time than we have available
if time_on_category > days_available_for_category:

# so exclude these options - sets to high positive or negative value
    return stage.parent_dp.exclusion_value

else:
    return benefit_list[stage.number][time_on_category]

# define the dynamic program and tell it we have four timesteps

dynamic_program = dypy.DynamicProgram(timestep_size=1, time_horizon=5,
objective_function=objective_function, calculation_function=dypy.MAXIMIZE,
prior=dypy.SimplePrior)

dynamic_program.decision_variable = decision_variable
dynamic_program.add_state_variable(state_variable)
# each category will be a stage, in effect - tell it to create all five stages as empty

# assigns names by default, but we can override them
dynamic_program.build_stages(name_prefix="Category")

# make a list of names in the same order they're used in our objective function
stage_names = ["Core", "UI/UX", "Network", "Database", "API"]

# and use it to set the value of .name for each stage
for i, stage in enumerate(dynamic_program.stages):

dynamic_program.stages[i].name = stage_names[i]

# run the dynamic program - results are logged to python logger `dypy` and decisions are set on each stage
as .decision_amount
dynamic_program.run()

```

References

For references and new developments in deep reinforcement learning please check out the links below:



1. FavTutor.com: [Dynamic Programming \(With Python Problems\)](#)



2. [DyPy Core Concepts and Classes](#)



3. [DyPy Documentation](#)

Authored by

[Tariq Khosla,](#)

Analyst at Absolutdata

Using Graphs to Illuminate Analytical Problems

Vivid Visualization

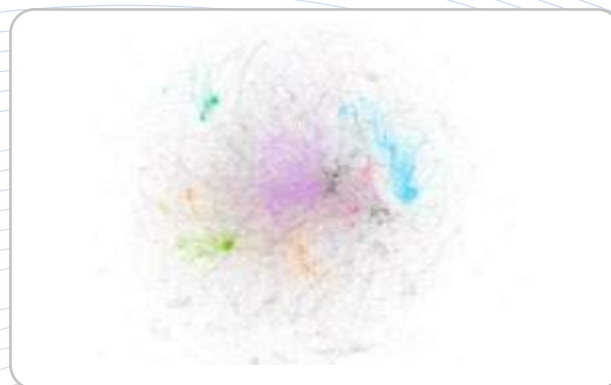
Visualization has proven to be key in data science. Data visualization methods vary from simple Excel graphs to network diagrams with customized visuals. These networks help us to understand everything from social media to real-world applications.



Source: [Network Visualization](#)

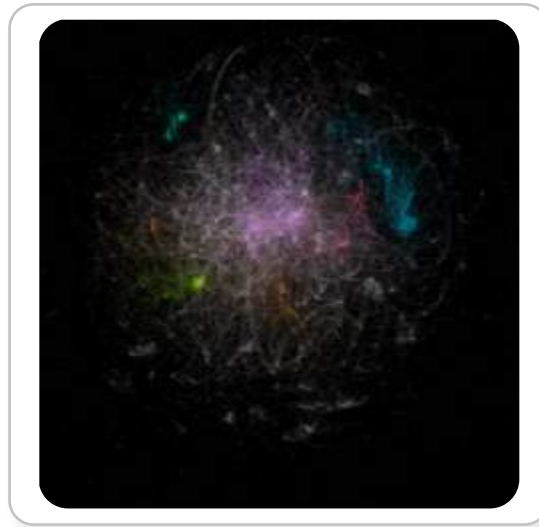
Network Visualization helps us understand the semantics of macro patterns through the use of micro patterns. It symbolizes one possible reality, not the sum of all possible realities. Scientific visualization enhances the questions arising from the data, and the truth of the data is enhanced by the visualization. In turn, this leads us and others to enquire about the data.

Envisioning data through possible filters to reduce its dimensionality can be implemented through unsupervised algorithms, such as clustering algorithms, grouping the nodes by means of similarity, or page ranks that, based on node sizes, signify a single data can have multiple visualization facets. To visualize the link dataset, visualization packages like the open-source Gephi can be used. The size and density of the graph signifies that it shrinks to the specific subset; edge count reduces the graph to significant edges. Node positioning is implemented by the Open Ord layout algorithm and colored by Blondel et al's modularity, with coloration selected by Gephi



Source: [Graph](#)

The Open Ord algorithm utilizes random seed methodologies, each time giving different results. This algorithm results in differences in visualizations on consecutive runs, moving out to one of the best visualizations. It helps to separate the cluster of greatest importance. Changing the background color creates clear separation of the nodes and makes a sharper contrast between the clusters, heightening the impact. Color selection can also change how the user perceives the graph.



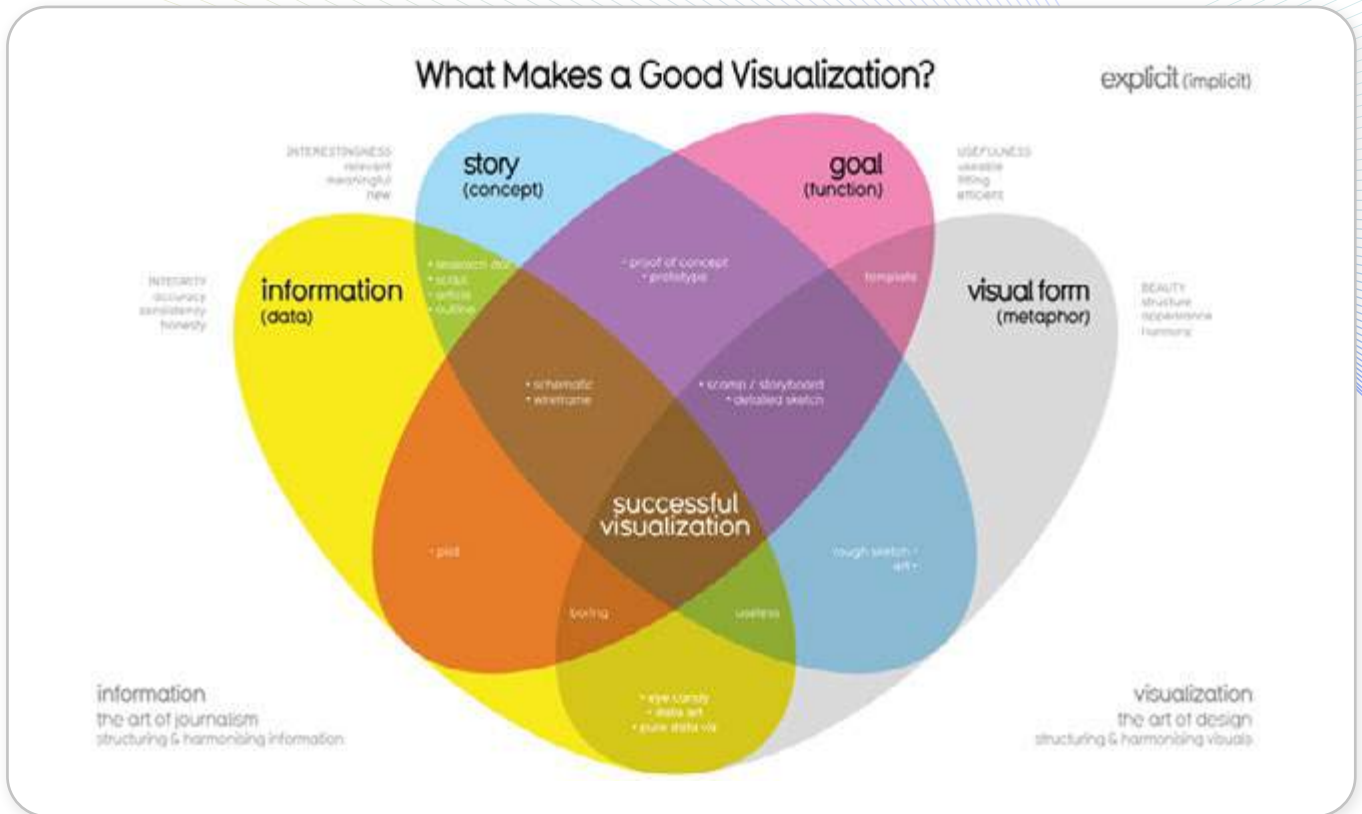
Source: [Network](#)

Changing the thickness of the network lines changes the view from micro structure to macro structure. Adjusting the thickness of each edge based on edge strength makes the central core less prominent and emphasizes the isolated nature of small clusters around the boundary.



Source: [Graph](#)

Graphs are often displayed as edge visualizations, where nodes indicate the focal point of the image. Algorithmic and methodological decisions have a major impact on visual representations.



Source : [what-makes-a-good-data-visualization](https://www.absolutdata.com/what-makes-a-good-data-visualization/)

Best Practices for Data Visualization

The overall best practice is to use data visualization to get to the truth of the data.

- The first key is to address the audience and design the visualization according to the audience's needs, level of understanding, etc. This helps the user to use the data to make business decisions.
- The second key is to understand what each visual signifies and then to choose a visual that is understood by the user. Different types of visuals include line charts, histograms, heat maps, scatter plots, and pie charts (among many others).

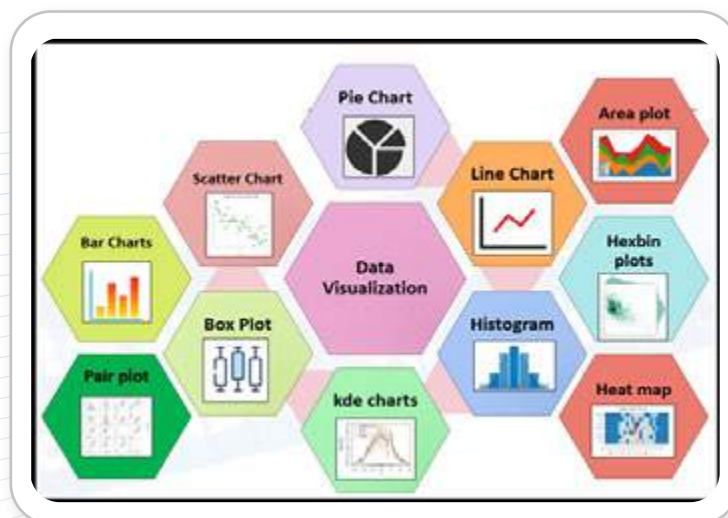
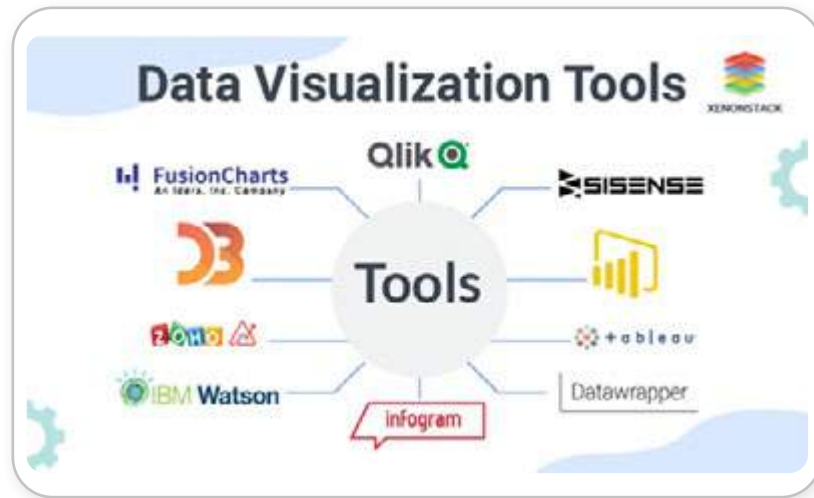


Figure 1: [Different Types of Visuals](#)

- Text clarifying the meaning of the visualization should be applied carefully and with due thought.
- Multiple visualization runs should be used; out of these visualizations, the best should be tuned to applications.
- To create the visualization, correct visualization tools must be applied. Top data visualization tools include Tableau, PowerBI, and others.



Source: [Data Visualization Tools](#)

- The visualization should be enhanced by thoughtful use of color.
- Attractive and intuitive dashboards can also be used to enhance data visualizations.
- Visualization should involve the user.
- Visualization should present meaningful business insights.

References

1. [Kalev Leetaru / Forbes.com: Why Data Visualization Is Equal Parts Data Art and Data Science](#)
2. [Navdeep Singh Gill / XenonStack.com: Interactive Data Visualization Techniques and Tools –A Quick Guide](#)

Authored by

[Sachin Kumar Yadav](#),

Analyst at Absolutdata

Pattern Clustering Using Cooperative Game Theory

Thriving Traction

Clustering is grouping a set of objects based on some common characteristics. Clustering has applications in many fields, including data analysis, pattern recognition, image analysis, data compression, computer graphics, and machine learning.^[1] Clustering has also been used to solve large-scale problems. Determining the number of output clusters k is challenging. In this case, using cooperative game theory provides a new way of addressing this problem by using a variety of solution concepts.^[3]

Cooperative Game theory (CGT) is a model of game theory where the players (also known as coalitions) use cooperative behavior in competition. It is also referred to as a coalitional game.^[2] The coalition behavior of the players is monitored by external agencies under regulatory authority.^[2]

Game-Theoretic Solution Concepts for the Clustering Problem

Specifying value for each coalition gives a cooperative game. The cooperative game has a finite set of players N (also known as the grand coalition) and a characteristic function $v: 2N \rightarrow R$. It defines the value $v(S)$ of any coalition where $S \subseteq N$. The function describes how much collective payoff can be gained by a set of players by forming a coalition; the game is also known as value or profit game.^[1]

The formation of the grand coalition N is the main assumption in cooperative game theory. The challenge now is fairly allocating the payoff $v(N)$ among the players. This assumption is not restrictive; even if players split off and form smaller coalitions, we apply solution concepts to the subgames defined by whatever coalitions are formed.^[4]

Let's explore the solution concepts to utilize their properties for the clustering game. A Shapley value is based on average fairness and stability, while Nucleolus is based on both min-max fairness and stability. Of these solution concepts, Nucleolus properties are the most suitable for the clustering game.^[3] Nucleolus comes with its drawbacks, most notably that it is computationally expensive. We can use other solution concepts for computational ease. These are mentioned below:

A. The Core

When the set of attainable allocations cannot be improved further by a coalition (a subset) of the economy's agents, it is called the Core. An allocation is said to have the Core property if there is no coalition that can be improved.^[5]

Let (N, v) be a coalitional game with transferable utility (say, TU). Let $x = (x_1, \dots, x_n)$ where x_i is the payoff of player i . Then, the Core consists of all payoff allocations, that satisfy the following properties:

1. individual rationality, i.e., $x_i \geq v(\{i\}) \forall i \in N$
2. collective rationality i.e., $\sum_{i \in N} x_i = v(N)$
3. coalitional rationality i.e., $\sum_{i \in S} x_i \geq v(S) \forall S \subseteq N$

A payoff allocation that satisfies individual rationality and collective rationality is called an imputation.

B. The Nucleolus

The allocation that minimizes the dissatisfaction of the players from the allocation they can receive in a game is the Nucleolus.^[3] For every x , consider the excess defined by

$$e^s(x) = v(S) - \sum_{i \in S} x_i$$

$e_s(x)$ is a measure of the unhappiness of S with x . The goal of Nucleolus is to minimize the most unhappy coalition, i.e., the largest of the $e_s(x)$. This could also be written in linear programming problem formulation as $\min Z$, subject to

$$Z + \sum_{i \in S} x_i \geq v(S) \quad \forall S \subseteq N$$

$$\sum_{i \in N} x_i = v(N)$$

It combines several fairness criteria with stability. It is the central imputation, and thus in the min-max sense it is fair and optimum. If the Core is non-empty, the Nucleolus is in the Core.^[4]

C. The Shapley Value

The Shapley value is the unique payoff vector that satisfies monotonicity, efficient and symmetric.^[4] To each cooperative game, it assigns a unique distribution of a total surplus generated by the coalition of all players.^[6]

If it follows the fairness-based axioms already discussed, then any imputation $\varphi = ((\varphi)_1, \dots, \varphi_n)$ is a Shapley value. For any general coalitional game with transferable utility (N, v) , the Shapley value of player i is given by

$$\begin{aligned} \varphi_i &= \frac{1}{n!} \sum_{i \in S} (|S| - 1)! (n - |S|)! [v(S) - v(S - i)] \\ &= \frac{1}{n!} \sum_{\pi \in \Pi} x_i^\pi \end{aligned}$$

Π = set of all permutations on N .

x_i^π = contribution of player i to permutation π .

Clustering Based on Cooperative Game Theory

Let's assign maximum Shapley value for each cluster as cluster center. If the center has a high density surrounding, we will consider other close points to be center; else it will consider more far away points.

Algorithm: DRAC (Density-Restricted Agglomerative Clustering) ^[3]

Require: Dataset, the maximum threshold for similarity $\delta \in [0, 1]$, and threshold for Shapley value multiplicity $\gamma \in [0, 1]$ ^[3]

1. Start with checking pairwise similarities between all points in the dataset.
2. Compute the Shapley value for each point i .
3. Arrange the points in decreasing order of their Shapley values. Let g_M be the global maximum of Shapley values. Start a new queue, it'll be our expansion queue

4. Next, start a new cluster. Of all the unallocated points, choose the point with the maximum Shapley value as the new cluster center. Let I_M be its Shapley value. Mark those points as allocated. Add it to the expansion queue.
5. Set $\beta = \delta q \sqrt{(I_M/g_M)}$
6. For each unallocated point where the similarity of that point to the first point in the expansion queue is at least β , add it to the current cluster and mark it as allocated. If the Shapley value of that point is at least γ -multiple of I_M , add it to the expansion queue.
7. Remove the first point from the expansion queue.
8. If the expansion queue is not empty, go to step 6.
9. If the cluster center is the only point in its cluster, mark it as noise.
10. If all points are allocated a cluster, terminate.

Results

Let's compare our algorithm with some existing algorithms. SHARPC (security- and heterogeneity-driven scheduling algorithm) proposes a novel approach to finding the cluster centers and giving a good start to K-means using a game theoretic solution concept; the Shapley value which thus results in the desired clustering. The limitation of this approach is that it is restricted to K-means; this is not always desirable, especially when the classes have unequal variances or when they lack convexity.^[3]

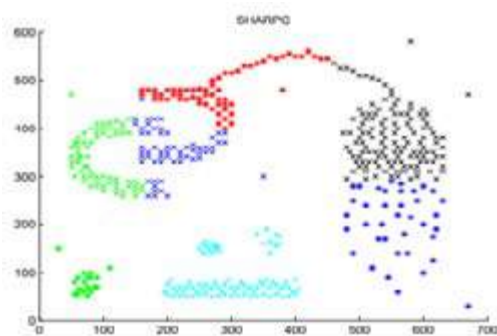


Fig 1: Clusters as discovered by SHARPC

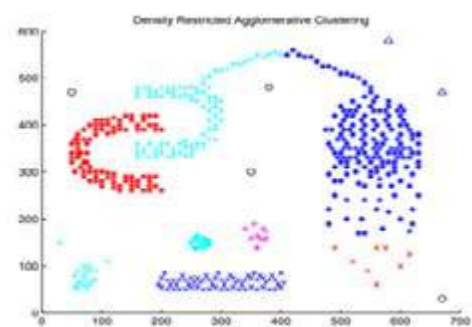


Fig 2: Clusters as discovered by DRAC

Image credits: [Pattern Clustering using Cooperative Game theory](#)

SHARPC cannot detect clusters that are not convex. If the threshold is increased to solve merging of different clusters (the light blue clusters in Fig 1), more clusters are formed and the larger clusters get subdivided into several small clusters.

The DRAC cluster (the light blue clusters in Fig 2) is highly dense; its cluster center has very high Shapley value. This results in a very high value of β , the similarity threshold. The red and light blue clusters are distinct. The red cluster is a low-density cluster (owing to the low Shapley value of the cluster center). It has a low β value, the similarity threshold, thus allowing more faraway points to be a part of the cluster.

References

- [1] Wikipedia.org: [Cluster Analysis](#)
- [2] eFinanceManagement: [Cooperative Game Theory | Transferable utility, Example](#)
- [3] Swapnil Dhamal, et al.: [Pattern Clustering using Cooperative Game theory](#)
- [4] Wikipedia.org: [Cooperative Game Theory](#)
- [5] Wikipedia.org: [Core \(game theory\)](#)
- [6] Wikipedia.org: [Shapley value](#)

Authored by

[Kajol Sah,](#)

Consultant at Absolutdata

The Mythology of Game Theory

Folk-Wisdom's Fallacy

The Mythology of Non-Cooperative Games

Non-cooperative games study the conflict situations among players – i.e., it is the study of situations where the payoffs don't simply depend on the player's own action, but also on the action of another player. Game theory assumes players are rational; each player will always act to maximize their payoff given their beliefs about how other players will play.

It can be concluded that game theory is a cognitive science; thus, it has its own assumptions that players are aware of all the actions which are available to them. One of the salient features of game theory is that one can always design different payoffs, settings, number of players, and actions and information available to each player. These differences lead to different strategies, equilibria, and outcomes.¹

Research has shown that a player's real-life behavior does not align with the predictions derived from game theory. The differences in real-life behavior and traditional assumptions of game theory can be attributed to:

- The cognitive biases of individual players while making decisions.
- A mismatch in game payoffs to players' expected payoffs.
- A player's limitations in thinking about other players' strategy and behavior.

Dysfunction with Nash Equilibria

We all are human beings, so our approach to solving problems can be cognitively very different from each other. We have flexibility in our approaches, and these choices sometimes do not comply with Nash equilibrium strategies. It is not always a dysfunction; people make their choice according to their conjectures, so it is important to correct the assumptions of Nash equilibria as the use of game theory increases in various decisioning domains. According to one analysis:

“[The] Nash equilibrium (NE) concept entails the assumption that all players think in a very similar manner when assessing one another strategies. In a NE, all players in a game base their strategies not only on knowledge of the game's structure but also on identical conjectures about what all other players will do.”²

The mythology of game theory is that these identical conjectures are applicable to all situations and settings and for all players; it is the core that is protected in game theory.¹ There are lot of works done in experimental games showing that players' conjectures and beliefs demonstrate some deviation from the pre-assumed game theoretic mythology. An experiment was conducted that confirms this.

Experimental design and inconsistent behavior across games

A real-life experiment was carried out to confirm if human players behave as per the NE strategies.¹ A total of 180 humans participated in this experiment for two types of games: a trust game and a donation game. Each player had \$5 at the start. Player 1 chooses how many dollars they want to pass to Player 2. The money passed will be increased threefold, meaning Player 2 will receive the tripled amount (plus their original \$5). Player 2 decides how much they will return to Player 1. This information is available to

all the players, and their choices are private and anonymous with respect to other players. The subgame perfect Nash equilibrium (SPNE) is that either player will pass and return \$0 as the dominant strategy equilibrium. The assumption is always that each player wants to maximize their payoffs and each player believes that the other player will also do the same. That leads to the thought “Player 2 will return \$0; I will not pass any money to them”. This is the equilibrium strategy assumed in Nash beliefs.

Standard approaches that show deviations from NE strategies assumes players will not follow game theoretical expectations. Across both games, inconsistency was observed in the players’ behavior. In a trust game, it was observed that 56% of participants as Player 1 sent money (\$1.43 on average); as Player 2, they returned \$1.23 on average. These deviations cast doubts on Nash beliefs. Out of 100 players who received money as Player 2, only 62 of them returned on average \$2.22. Those 62 who returned money were not consistent in sending money. There are 60 players who behaved consistently according to the Nash equilibrium in both trust game roles but lacked consistency in the donation game.

Are beliefs and behavior consistent?

Various reasons from cognitive science force us to doubt the behavior and beliefs of the players across different game settings. The human mind is always shifting and changing its way of thinking, which explains why players are not consistent in their choices. It is also evident from this experiment that players show variance in their choices while playing games.



The mythology of game theory assumes that players in a game environment have their own preference and payoff maximization strategy and know how other players will act and think. In this mythology, we cannot chalk out the difference if someone asks the players to select a particular action in the game before making predictions or vice versa. The results also suggest that changing the belief and order of choice affects players’ beliefs and this change in task does not comply with Nash equilibrium.

Conclusion

The results showed that players mostly deviate from NE beliefs. The deviations observed are not simple, consistent, and easy to explain; they keep on changing as per the environment. This experiment also proved that players do not have shared beliefs. Rather, their own beliefs seem to be fit for one game environment and can change in another environment. So, it may be misleading to conclude these results are deviations from Nash beliefs. Equilibrium concepts are not based on how humans think or make decisions in real life, so models that use false assumptions may not create problems when prediction is the only goal (as opposed to understanding).

However, Nash equilibrium models failed to predict real-life behavior, so we should not accept the success of these models in prediction scenario due to use of false assumptions. Rather than trying to fit humans in an old, developed mythology, we should build models where we can fit humans with their real behavior. No model has been developed till now, but research is ongoing.

References

-  McCubbins, Mathew D., Mark Turner, and Nicholas Weller. “The Mythology of Game Theory” in Proceedings of the International Conference on Social Computing, Behavioral-Cultural Modeling & Prediction 1-8. (2012)
-  Lupia, A., A. S. Levine, and N. Zharinova. “Should Political Scientists Use the Self Confirming Equilibrium Concept? Benefits, Costs, and an Application to the Jury Theorem.” Political Analysis 18:103-123. (2010)

Authored by

[Akash Omer,](#)

Senior Consultant AI/ML at Absolutdata

Interpreting Machine Learning Models Using Shapley Values

Experience Extended

Introduction

Some machine learning models (such as neural networks, gradient boosting, and ensemble methods) are called “black-box models” due to their lack of explainability. Though these models have an extremely high accuracy, sometimes other models (e.g., linear and logistic regression) are preferred. This is due to their understandable variables and clarity in explaining feature interactions through model coefficients, P-values, etc.

In such cases, Shapley values can be extremely useful. This is a game theory solution that involves distributing both gains and costs fairly to all the players in a coalition. Shapley values are also used in other domains, such as attribution modeling in web analytics, economic models, and fair division problems. These values help in explaining the feature importance of model by comparing different sets of outputs and attributes relative importance to each feature.

How Shapley Values are calculated:

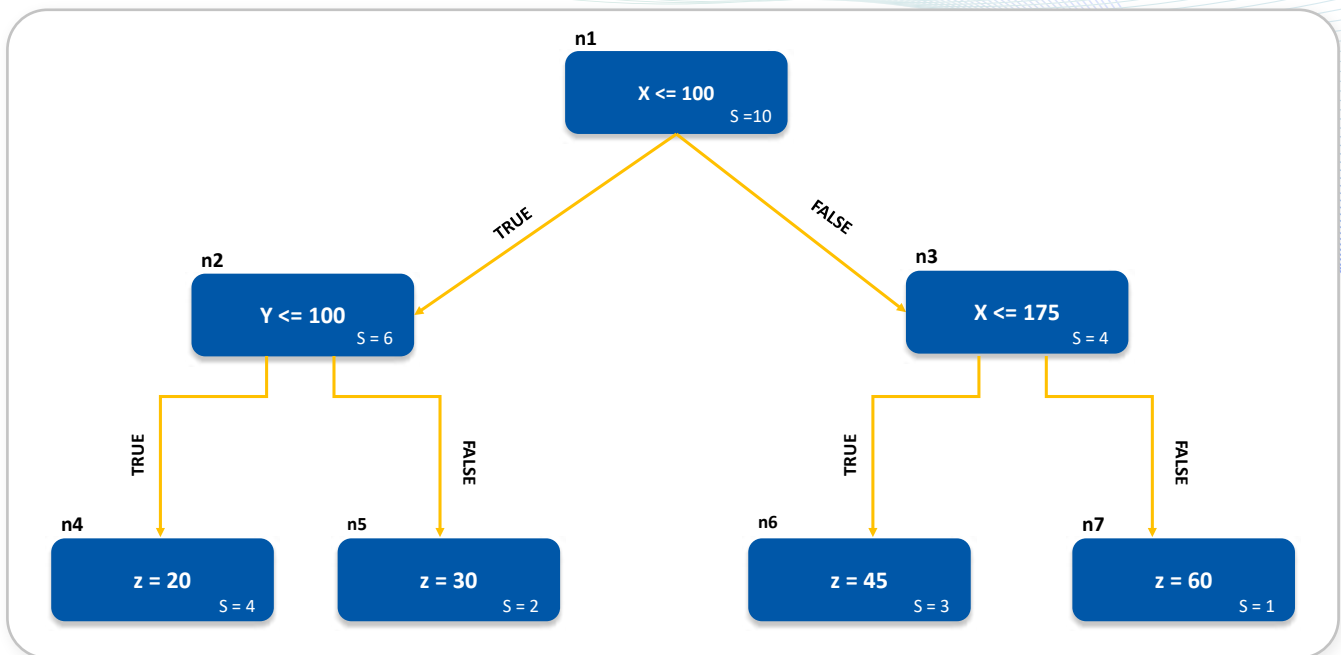
Shapley values work by calculating an Average Marginal Contribution for a feature in the model. There is a Shapley value for an individual feature when there is a set N of n players and a function v that maps subsets of features to the real numbers: $v: 2^N \rightarrow \mathbb{R}$, $v(\emptyset) = 0$, Where \emptyset denotes the empty set

$$\Phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|! (N - |S| - 1)!}{N!} (v(S \cup \{i\}) - v(S))$$

1. S is a coalition of all features.
2. $v(S)$ is the worth of coalition.
3. v is the Total expected sum of payoffs the features of S can obtain by cooperation.
4. i is the feature notation

The Tree Shap Method for Explaining Decision Tree Models

Consider the following decision tree with a total of ten random samples:



1. X and Y are the independent variables.
2. z is the output variable.
3. n is the node notation.
4. S is the number of samples for the respective node.

Different permutations and combinations of the independent variables X and Y are taken separately to calculate the feature importance using Shapley values.

Let's compute the Shap values for the selected instance (i) X = 130, Y = 65. The Output for this instance is 45.

The prediction of the null model $\phi_0 = (20 \cdot 4 + 30 \cdot 2 + 45 \cdot 3 + 60) / 10 = 33.5$

Consider the Sequence X > Y

1. X is added to the null model first. For the selected instance, the marginal contribution of X is $\phi_{x1} = 45 - 33.5 = 11.5$ (from node n6).
2. Now Y is added to the model. Adding Y does not alter the prediction; hence the marginal contribution of Y $\phi_{y1} = 45 - 45 = 0$.
- 3.

Consider the Sequence Y > X

1. Y is added to the null model first, but the node n1 uses only X and hence the prediction = $(6/10) \cdot 20 + (4/10) \cdot 48.75$ (contribution from child node n2) + $(4/10) \cdot 48.75$ (contribution from child node n3).
 - I. From node n2: n2 has the Y variable; hence the prediction is = 20.
 - II. From node n3: n3 still does not have the X variable the prediction = $(3/4) \cdot 45 + (1/4) \cdot 60 = 48.75$.
- III. Total prediction with just Y is $(6/10) \cdot 20 + (4/10) \cdot 48.75 = 31.5$, and marginal contribution of Y is $\phi_{y2} = 31.5 - 33.5 = -2$.

2. Now **X** is also added to the model, which gives a prediction of 45; the marginal contribution of X is $\phi_{x2} = 45 - 31.5 = 13.5$

Final SHAP Values:

$$\phi_x = (\phi_{x1} + \phi_{x2}) / 2 = (11.5 + 13.5) / 2 = 12.5$$

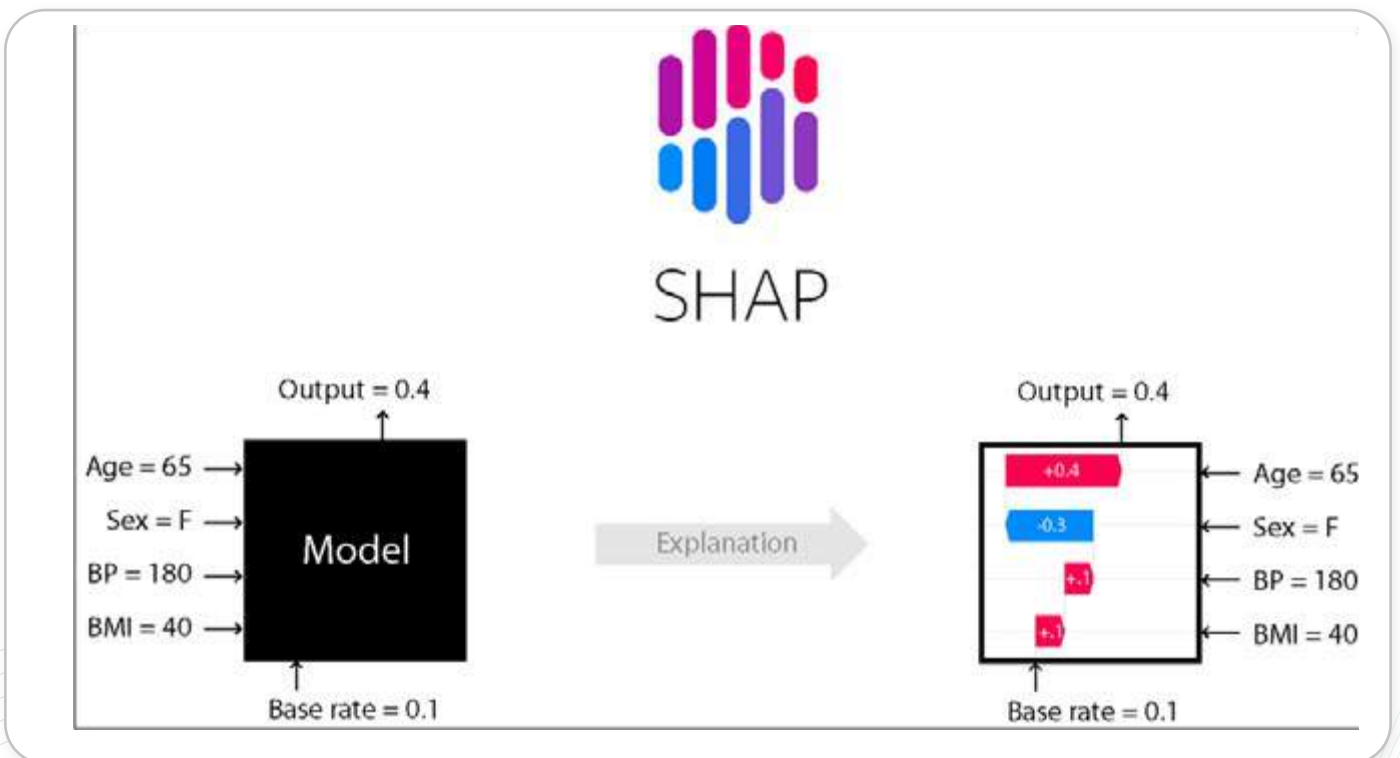
$$\phi_y = (\phi_{y1} + \phi_{y2}) / 2 = (0 - 2) / 2 = -1$$

The prediction for the instance *i* can be explained by $\phi_0 + \phi_x + \phi_y = 33.5 + 12.5 - 1 = 45$, Thus explaining the feature prediction *i*



Force plot of the above decision tree explained using SHAP values.

Here blue indicates that the **Y** value decreased the prediction. Red indicates the **X** value increased the prediction.



Summarizing the SHAP method as a tool

Conclusion:

1. Shapley values are the only attribution method that satisfies efficiency, symmetry, dummy, and additivity.
2. Similar to the above calculations, the same process can be applied for other machine learning algorithms, such as deep neural networks, random forest, ensemble methods, etc.
3. The Python package **SHAP** offers an array of tools to better visualize and understand black box models.
4. Thus, Shapley values provide explainability and interpretability to ML algorithms.

References



1. Investopedia.com: [Shapley Value Definition](#)



2. Wikipedia.org: [Shapley Value Calculation](#)



[SHAP: A Unified Approach to Interpreting Model Predictions. arXiv:1705.07874](#)



[SHAP Documentation](#)

Authored by

[Vishnu Prasath,](#)

Consultant at Absolutdata

How AI Could Improve Equality and Productivity in Tax Policies

Food for Thought Experiment

In today's world, the socio-economic impact of a country's tax policy is more far-reaching than ever. The key objectives of a country's tax policy are:

- To ensure an economically stable and consistently improving government body.
- To lessen the wealth inequality in the citizens of the county.
- To have a handle on the economic activity amongst the citizens of the country.

But, taking a look at the present-day world, we are seeing many countries unable to meet these goals. Sri Lanka is looking for funds from the IMF and has borrowed money from individual countries (including India and China) in an attempt to maintain a stable environment in the country. With declining forex reserves and growth in imports, Nepal is said to be next in line after Sri Lanka. Pakistan is seen by many as soon to be an economically failed state. So, based on this, we understand that taxation policies need to proactively ensure that we don't end up in the same situation; they need to be fine-tuned for growth and the well-being of both individual citizens as well as the government.

AI for Taxation

With multiple iterations of design improvement and rigorous testing, we can stimulate the outcome of a modified taxation policy and its overall long- and short-term impact –thus arriving at the most efficacious taxation policy. This falls in line with the AI domain of Reinforcement Learning (RL). RL creates an environment or a simulation; in this case, this could be an economic environment similar to that of a country with complicated and competitive transactions across varied markets.

Multiple variations of AI algorithms known as agents try to optimally reach the objective of an equality productivity trade-off, with the economic environment rewarding each correct decision and penalizing incorrect steps. The agent with the largest improvement in the trade-off score (as compared to the country's existing trade-off score) could then be explored further to gain insights and the same can be driven through a feedback loop involving citizens of the country.

Hence, we have a data-driven, trial-and-error approach that disregards any economic assumptions and allows both the reinforcement learning agents and the country's government to quantify and balance the tradeoff between the equality of wealth and citizens' productivity.

Actors in Reinforcement Learning

To devise an optimal taxation policy, we have two kinds of players in the RL algorithm:

1. The government, which is referred to as the 'policy maker'
2. The citizens, who are referred to as the 'agents'

Economic Environment for an Agent

This is a simulation, similar to the real country, where agents or citizens can move freely, collect assets, and build a house. An agent also can perform a transaction involving an exchange of some or all of their collected assets at a set price.

At the onset, each agent is allocated the same amount of initial money for equality. But thereafter – similar to a real-world scenario – their earnings might evolve, depending on that agent's actions and skillset. An agent with an expensive skill will have higher earnings.

Also, while actions like performing a transaction or building a house act as wealth generators, all other actions incur a cost. Towards the end of the simulation, all agents are given an award based on their earnings and the costs incurred. This award reflects the agent's ability to utilize the assets they've acquired. This Constant Relative Risk Aversion (CRRA) function is used to deduce the utility score.^[1] It increases with an increase in earnings and linearly decreases as per costs incurred.

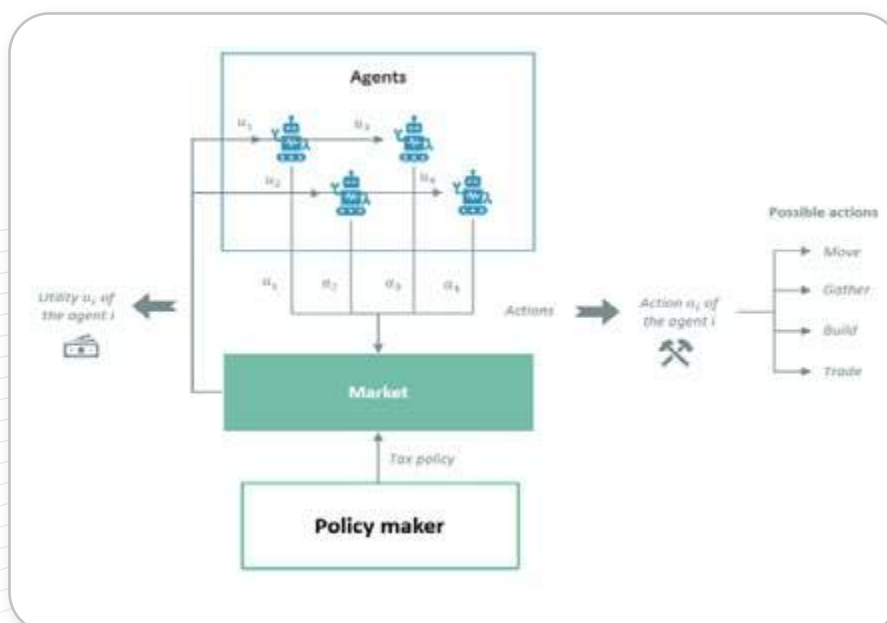
$$U_a(E_{a,t}, C_{a,t}) = CRRA(E_{a,t}) - C_{a,t}$$

$$CRRA(E_{a,t}) = \frac{E_{a,t}^{1-\eta} - 1}{1-\eta}, \quad \eta > 0$$

Here:

- $E_{a,t}$ denotes the Earnings of an agent a at time t
- $C_{a,t}$ denotes the Costs of an agent a at time t
- η denotes the degree of linearity and remains constant for all agents in the simulation.

The following figure accurately describes the simulation environment from an agent's point of view



Source: [AI-driven tax policy what it would look like](#)

Economic Environment for the Policy Maker

Each simulation must contain more than one tax period (ideally between 5 to 20 periods) to confidently interpret the results of multiple iterations of taxation policies on the agents in the experiment. The baseline for the initial tax period could be similar to the country's current taxation policy, wherein each agent must pay direct taxes in proportion to their earnings in the tax period. All the tax collected by the policy maker is then dispensed to the agents equally. The policy maker's aim is to deduce the optimal tradeoff between equality of wealth and the productivity of the agents in simulations. This optimality can be measured as the EP (Equality-Productivity) score.

$$\text{EP score} = \text{Equality} * \text{Productivity}$$

Here Equality is derived as a complement of the **Gini Index** as follows:

$$\text{Equality} = 1 - \text{gini}(\text{Earnings}^{\text{cumulative}}) \frac{N}{N-1}$$

$$\text{gini}(\text{Earnings}^{\text{cumulative}}) = \frac{\sum_{i=1}^N \sum_{j=1}^N (\text{Earnings}_i - \text{Earnings}_j)}{2N \left(\sum_{i=1}^N \text{Earnings}_i \right)}$$

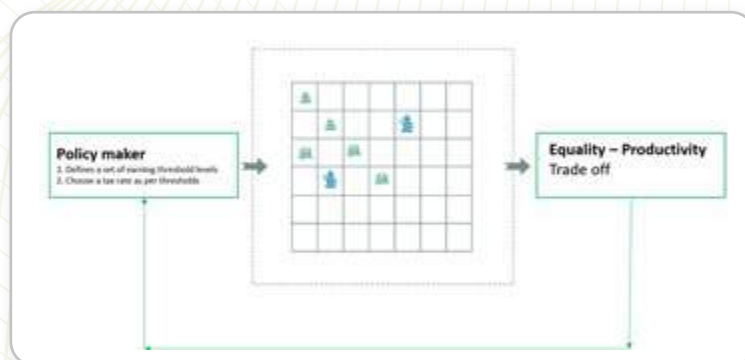
The equality score varies between 0 and 1. A complete Equality across the environment amongst all agents will give out the score of Equality as 1.

The productivity of a tax period in a simulation can be measured as the summation of all the earnings by all the agents in the simulation.

$$\text{Productivity} = \sum_i^N \text{Earnings}_i$$

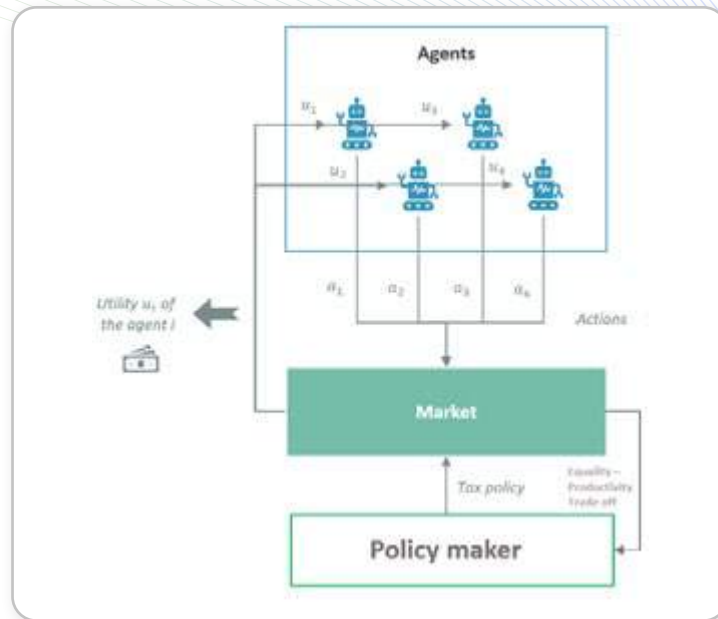
Where Earnings_i denotes the earnings of an agent i in tax period t in an environment with N unique agents.

The following figure accurately describes the simulation environment from an agent's point of view



Source: [AI-driven-tax-policy-what-it-would-look-like](#)

The following figure holistically describes the simulation environment



Source: [AI-driven-tax-policy-what-it-would-look-like](#)

Simulation Outcome

To get an adequately reliable result, the simulation must include a substantial number of agents across all groups of earning ranges i.e., the lowest-earning range, highest-earning range, lower middle-class earning range, and higher middle-class earning range. We do need to note here that this is a very simplistic form of simulation with pre-set prices and no scope of tax fraud, change in market conditions, or entry of new agents. Such an experiment would have its disadvantages; as an initiation exercise, the results will be achieved in a quicker and more adaptable format for the agents. In an experiment run by Stephan Zheng et al. ^[1], it was found that:

1. Reinforcement Learning gave a better Equality - Productive score.
2. It organically deduced a comparatively higher tax for high-earning individuals and a dip in taxes for middle-class earners.
3. The agents were discovered to have comparatively more improvement in overall productivity in their skillset.

Conclusion

This study is expected to serve as a foundation for deriving more efficient ways to analyze the impact of a policy change. Although these algorithmic results could not mimic the exact society behavior, even in its limited form such techniques have great potential to transform policy making from a politically driven process to more data-driven, result-seeking one.

References

1. [The AI Economist: Improving Equality and Productivity with AI-Driven Tax Policies](#)
2. [AI-driven-tax-policy-what-it-would-look-like](#)
3. [The Three Goals of Taxation](#)
4. [Why Sri-Lanka is facing unprecedented economic-crisis](#)
5. [Viking Village - app on Google store](#)
6. [Nepal's economic crisis](#)
7. [Pakistan in grip of intense and deep economic crisis](#)
8. [Using elasticity to derive optimal tax rate](#)
9. [The AI Economist](#)

Authored by

[Sidharth Suman,](#)

Senior Consultant at Absolutdata

Data Science Competitions/ Seminars / Fora / Courses

Competitions:

1. Multi-Agent Reinforcement Learning for Iterative Reasoning

The timeline and cash pool has not been announced yet.

2. HuBMAP + HPA - Hacking the Human Body

Registration Deadline - September 15th, 2022

Final Submission Deadline - September 22nd, 2022

Prize Money - \$60,000

3. Fossil Demand Forecasting Challenge

Final Submission Deadline - August 28th, 2022

Prize money - \$5000

4. Feedback Prize - Predicting Effective Arguments

Registration Deadline - May 24, 2022

Final Submission Deadline - August 23, 2022

Prize money - \$55,000

Authored by

[Ishan Dhall](#),

Analyst at Absolutdata



ABSOLUTDATA

An Infogain Company

Thank You

For reading this edition of BrainWave from the Absolutdata Labs Team. This digest focuses on some technical angles of analytics and data science. BrainWave is published about 4 times a year. If you haven't already, please subscribe so you receive future editions.

Subscribe

