Game Theory in the Art Market

Vasily Ilin
Eva Jungreis
Boston University

June 5, 2018

Abstract

Exploratory analysis of a game theoretic model of art market with an extension to optimization.

Contents

1 Introduction

The work presented below is an exploratory analysis of a simple game theoretic model of how an art market might work. In Section 2 the model and its core assumptions are described. Section 3 gives the payoffs for each possible equilibrium and describes how to programmatically solve the tree using recursion. Section 4 is dedicated to interactions with the game and visualization of the solution space. We explore an extension of the game into an optimization problem in section 5. Finally, Section 6 outlines possible extensions of the work presented here. The code is avalaible at `https://mybinder.org/v2/gh/Vilin97/Game_theory_exploration.git/master`. Instructions for running the program are in the Appendix.

2 Problem Statement

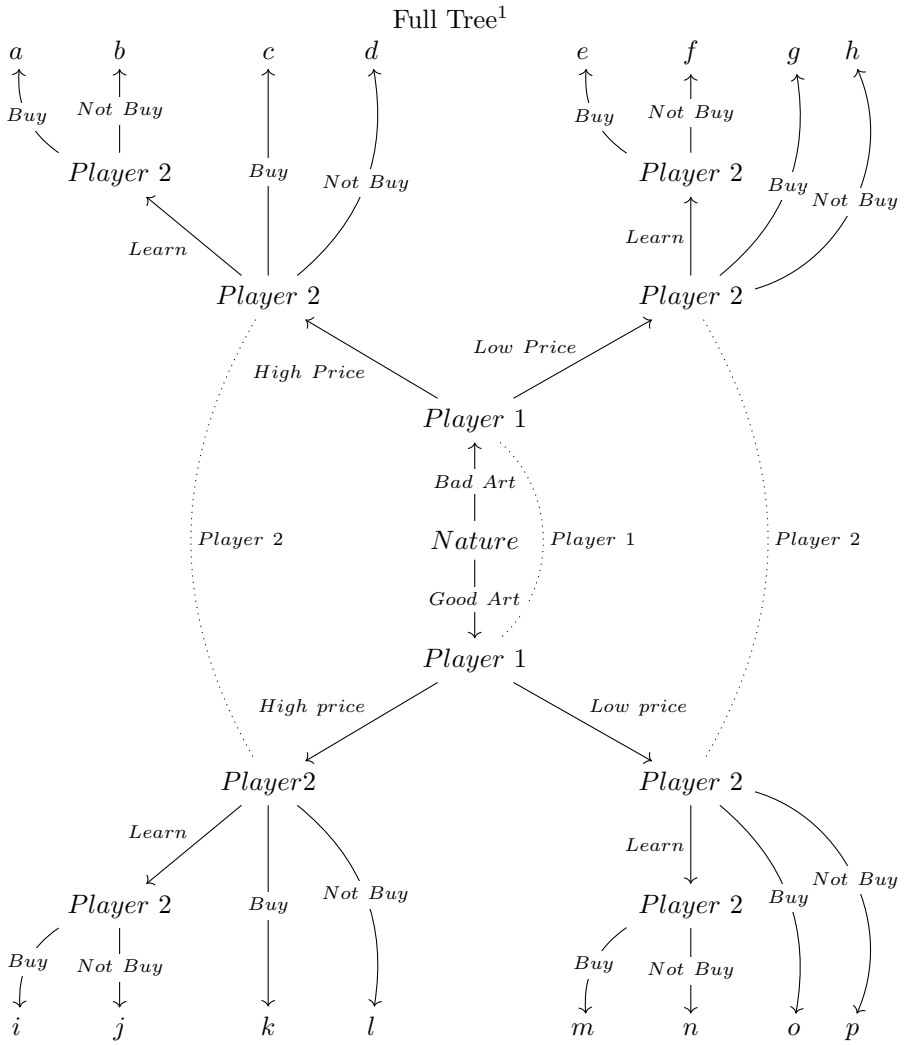
The model aims to describe the interaction between a producer of a good and a buyer of the good. Specifically, the authors had art market in mind but the model can be applied to any other situation where the assumptions specified below apply. The sequence of actions avalaible to the artist and the buyer is as follows. First, the artist produces art and sets a price, either high or low. Then, the buyer has the option to go to the expert and pay a fee

to learn the true value of the art. Whether the buyer learns the true value of the art or decides to keep the money, she can either buy or not buy the art piece.

There are a few core assumptions to this model.

- Neither the artist, nor the buyer knows the real value of the painting.
- The real value of the painting is revealed either by going to an expert to evaluate it or after the purchase by the means of other people's judgement. Note that the buyer does not derive any personal subjective utility from the art.
- Both players have full information about the structure of the game and the payoffs and are fully rational agents. The only unknown in each iteration of the game is whether the art is good or bad.

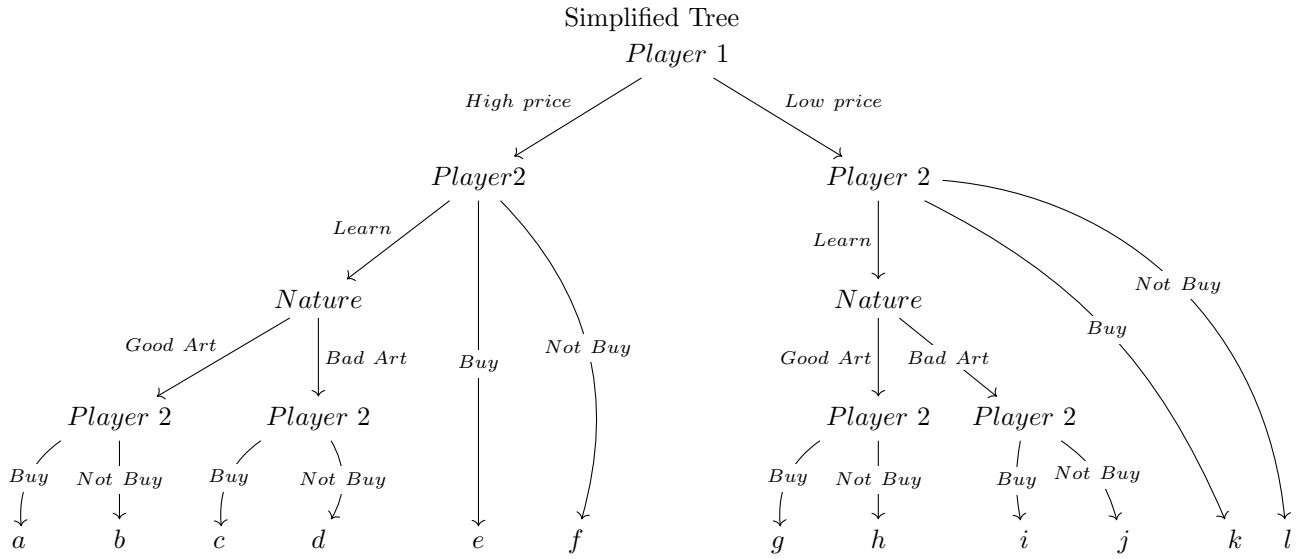
It is natural to represent this model in a game form with the artist being Player 1 and the buyer being Player 2. Nature represents chance. Below is the full game in extensive form. Notice that almost all non-leaf decision nodes are part of an infoset. This is because neither of the players knows the true value of the art except Player 2 after consuming "Learn."



This game can be simplified by uniting the nodes that belong to one infoset. This produces the tree below. Notice that it looks almost the same as a half of the full game tree. In fact, it would look exactly the same if consuming "Learn" did not reveal the true value. One can think of all nodes in the tree below as existing in two states at the same time, like Shrödinger's cat, with the exception of the nodes after "Learn" was consumed, since

[1]Subscript $_f$ denotes payoffs for the game in the full form

it resolves randomness the same way the observer resolves randomness by looking inside the box in Shrödinger's experiment.



3 Solving the Game

3.1 Expected payoffs

Solving the game for a specific set of parameters means finding the equilibrium. The first step to finding the equilibrium is finding expected payoffs for each possible result of the game: a through l. Expected payoffs for Player 1 are clear, since randomness does not play any role for the artist. The expected payoffs for Player 2 are clear for all equilibria except e and k, since for these nodes randomness is resolved or no transaction happens. The expectation of e for Player 2, $E_2(e)$, is the weighted average of the high value and the low value: $E_2(e) = -HP + p(HV|HP) * HV + p(LV|HP) * LV$[2], where the weights are conditional probabilities of art being good. $E_2(e)$, recall that the artist does not know if her art is good or not. In other words,

$$p(HP|HV) = p(HP|LV)$$

Using Bayes' Rule, get

$$p(HP|HV) = \frac{p(HV|HP) * p(HP)}{p(HV)} = \frac{p(LV|HP) * p(HP)}{p(LV)} = p(HP|LV)$$

Simplifying,

$$p(LV) * p(HV|HP) = p(HV) * p(LV|HP)$$

By the law of complementary events,

$$(1 - p(HV)) * p(HV|HP) = p(HV) * (1 - p(HV|HP))$$

$$p(HV|HP) - p(HV) * p(HV|HP) = p(HV) - p(HV) * p(HV|HP)$$

Cancelling $-p(HV) * p(HV|HP)$ on both sides,

$$p(HV|HP) = p(HV) = p$$

In other words, conditional probability of art being good given high price is equal to the unconditional probability of art being good, p. Similarly, $p(HV|LP) = p(HV) = p$. Therefore, $E_2(e) = -HP + p * HV + (1 - p) * LV$ and $E_2(k) = -LP + p * HV + (1 - p) * LV$. Thus, the expected payoffs are as follows:

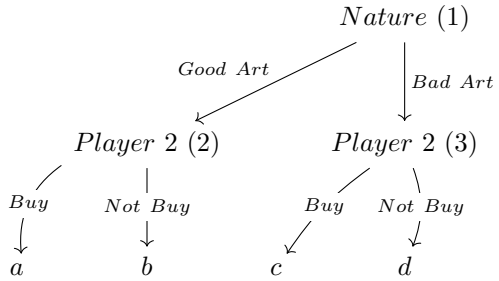
[2]HP = high price, LP = low price, fee = cost to learn the true value, HV = value of good art, LV = value of bad art, p = probability that art is good

	a	b	c	d	e	f
player 1	HP	0	HP	0	HP	0
player 2	-HP-fee+HV	-fee	-HP-fee+LV	-fee	-HP+p*HV+(1-p)*LV	0

	g	h	i	j	k	l
player 1	LP	0	LP	0	LP	0
player 2	-LP-fee+HV	-fee	-LP-fee+LV	-fee	-LP+p*HV+(1-p)*LV	0

3.2 Finding Equilibrium

Now, when we have expected payoffs for each possible equilibrium, we can solve the tree. The usual assumption of all players being fully rational applies. Thus, the tree is solved via backward induction. The only deviation from the rule of simply assigning the expectation of the best child node to the parent node is at Nature nodes. There, Nature makes the move at random, thus making the expectation of the node the weighted average of the two children nodes. Consider the following subtree:



Here, $E_2(node\ 1) = p * E(node\ 2) + (1-p) * E(node\ 3)$.

We saw how the expected payoffs for each node can be obtained. Solving the game is straightforward once the expected payoffs are known. At each node, the player making the move, will choose the child node with the highest expected payoff for the player. Thus, Player 1 choosing between setting the high price and setting the low price will compute expected payoffs for each of the two nodes and choose the one that yields the higher expected value for Player 1.

It is, of course, possible to work out the equilibrium by hand for any specified set of the six variables. But if one wants to have any kind of interactive functionality such as computing the equilibrium in real time with varying parameters, it is necessary to be able to solve the game programmatically.

3.3 Coding it all up

In order to solve the tree, we need to build it first. We will simply create the nodes and assign each of them the player making the move: 1, 2, or 3, where 3 is Nature. This provides the structure of the game. Then we set payoffs for each of the leaf nodes. This requires specifying the six variables that control the game. Now the tree is built and can be solved. Attentive readers already noticed that recursion is the natural way to get the expected payoffs for nodes and to choose the move for each node. Indeed, we have recursive **exp()** and **solve()** functions that take a node object as an input and return its expectation and the actualized node respectively. Note that the expectation of each node consists of two numbers: expectations for Player 1 and Player 2.

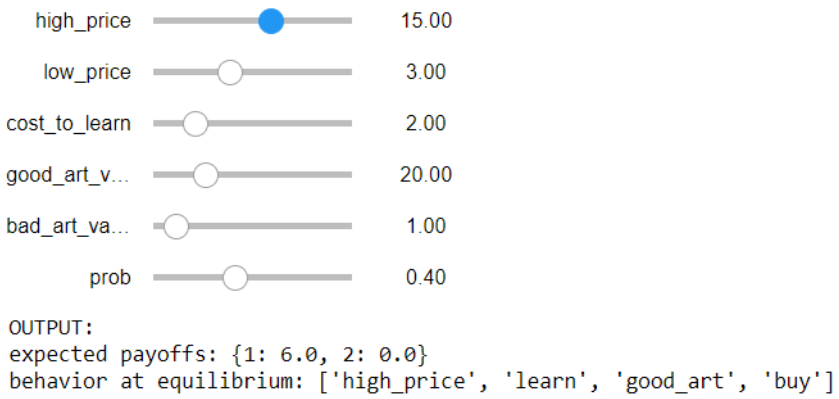
4 Interactive

4.1 Getting Equilibrium in Real Time

Now when the tree can be solved in seconds ($\sim 250\mu$s), the tree can be solved interactively in real time. This was implemented with the ipywidgets package.[3] There are six sliders, one for each variable. When the sliders are dragged the tree is solved in real time. Expected payoffs and the actualized behavior are shown. The interface is as below.

[3] All code was written in python, jupyter notebooks

Figure 1: Real Time Equilibrium Interface



4.2 Visualization

Since the equilibruim of the game depends on 6 variables, the equilibrium of the game can be thought of as a function $f : \mathbb{R}^6 \to \Omega$, where $\Omega = \{(high\ price,\ low\ price) \times (learn,\ buy,\ not\ buy)\}$ = space of all possible equilibria. Additionaly, we have $g_j : \mathbb{R}^6 \to \mathbb{R}$, $g_j(\vec{x})$ = expected payoff for Player j. Note that g has the same domain as f, and

$$g_j(\vec{x}) = \max_{s \in \mathbb{S}} E(s),$$

where $\mathbb{S}$ = the set of avalaible strategies for Player j, and $E(s)$ = the expected payoff of the node corresponding to strategy s.

Thus, $\mathbb{R}^6$ is partitioned by f into 6 regions, each region being the preimage of an element of the space of equilibria, Ω. Note that f partitions $\mathbb{R}^6$ into 6 regions. Then the question arises: what do these regions look like? In other words, where does one region turn into another? In order to explore these question, we built a graphical tool that plots probability of good art against the expected payoffs of each avalaible strategy for each of the two players, producing a slice of the multidimensional solution space.

Figure 2: Plotting Interface

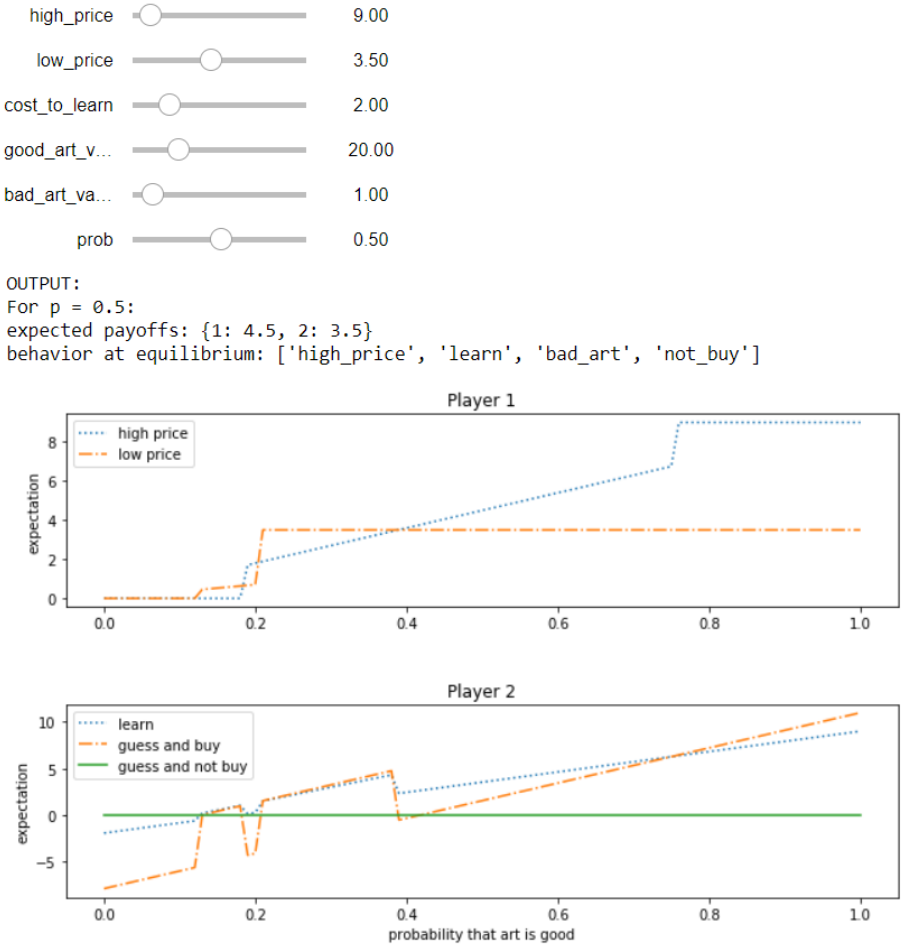


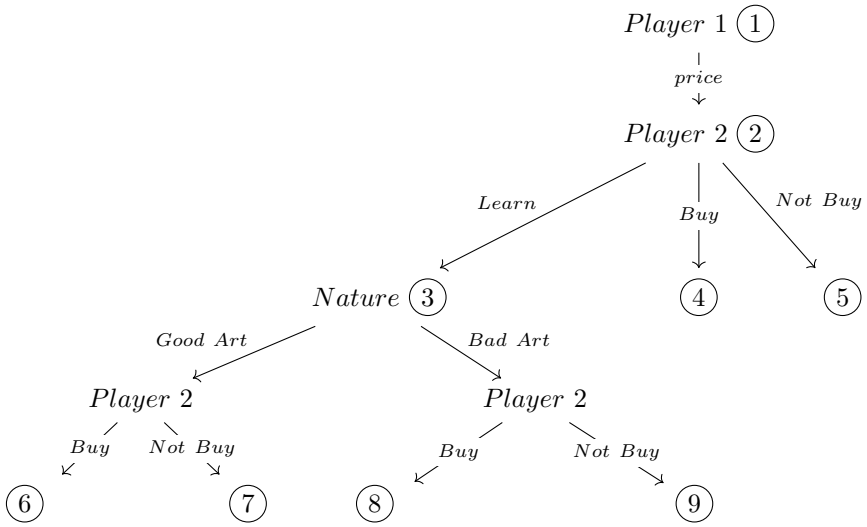
Figure 2 shows how non-trivial the solution space is.

5 Introducing Optimization

So far it was assumed that the values for high price and low price are exogenous parameters. Here we consider an extension of the game, where the artist can set the price herself, maximizing her expected payoff over price. The

new tree is as below.

Tree with Optimization

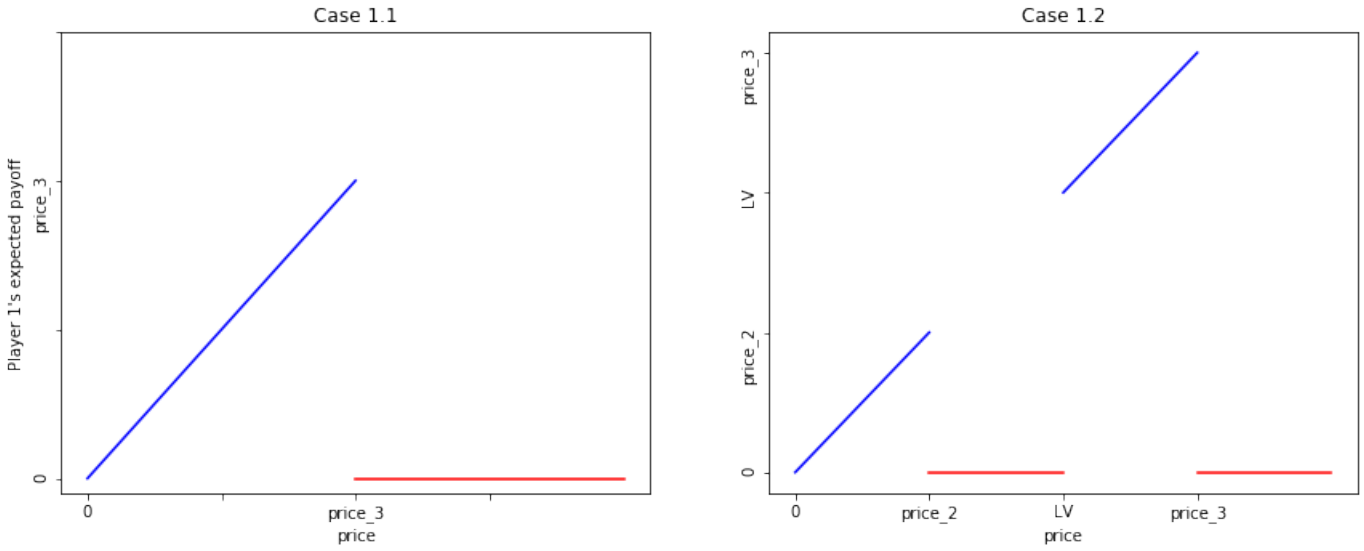


Now Player 1 has an optimization problem to solve. Her goal is to pick the price that maximizes her expected payoff. Formally,

$$\max_{p \in \mathbb{R}} E_1(1).^4$$

First, define some variables: $price_1 = \frac{fee+(1-p)*LV}{1-p}$, $price_2 = \frac{-fee+p*HV}{p}$, $price_3 = p*HV + (1-p)*LV$. Case 1 is $price_1 - price_2 = fee - p*(1-p)*(HV-LV) > 0$. In this case, $E_1(price)$ looks as below.

Figure 3: Case 1

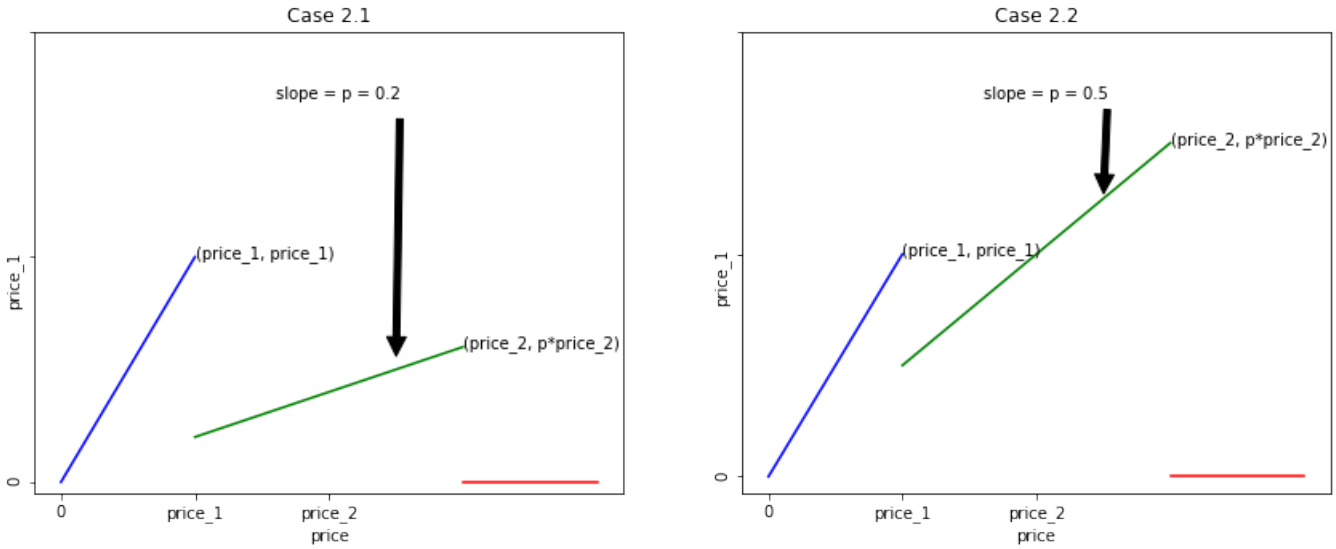


Case 1.1 is for $price_2 > LV$. Case 1.2 is for $price_2 < LV$. The outcomes for Cases 1.1 and 1.2 are the same: the optimal price is $price_2 - \epsilon$, with $\epsilon \to 0$.

Case 2 is $fee - p*(1-p)*(HV-LV) < 0$, with $E_1(price)$ as follows.

[4]The full solution can be found in the Appendix.

Figure 4: Case 2



Case 2.1 is for $price_1 > p * price_2$. Case 2.2 is for $price_1 < p * price_2$.

Thus, in order to maximize the expected payoff, Player 1 needs to first compute $fee - p*(1-p)*(HV-LV)$. If it is positive, set price to $price_2 - \epsilon$. Otherwise, compute $price_1 - p*price_2$. If it is positive, set price to $price_1 - \epsilon$. Otherwise, set price to $price_2 - \epsilon$.

6 Further Work

- Code up the optimization problem.
- Test the model. Estimate parameters from data online and see whether the model gives an accurate prediction.
- Are the preimages of f on $\mathbb{R}^6$ convex?
- use the ConvexHull from scipy to generate the boundaries of the preimages of f from sets of points.

7 Appendix

7.1 Running the code

The link should open a repository. Run "Art_market_plotting_TO_RUN.ipynb," select "Cell," and click "Run all cells." Alternatively, `https://github.com/Vilin97/Game_theory_exploration.git` is the link to the github repository containing the code.

7.2 Optimization

To solve Player 1's optimization problem, we need to first compute the expected payoffs of nodes 3, 4, and 5 for both players. Assume $LV < price < HV$. The edge cases can be dealt with later.

	3	4	5
player 1	p*price	price	0
player 2	p*(HV-price)	p*HV+(1-p)*LV-price	0