Math 553 Final Project - Metagame Analysis

Julien Codsi

December 2020

Contents

Chapter 1

Introduction

As the years go by, competitive video games are getting more traction. Tournaments are becoming bigger as the number of players and viewers are rapidly increasing. For example, the 2018 Mid-Season Invitational tournament, a League of Legends tournament, had 60 million viewers which is comparable to the viewership of the Super Bowl final of the same year (103 million viewers).

Some of the most watched Esport events are fighting game competitions. In those games, even though a lot of strategy and skills are required for specific matches, the majority of wins for some of the best players are greatly influenced by factors outside of the game itself. In this game, some characters are better suited to fight other characters. Knowing that in the gaming community, some characters are extremely popular and, thus, are use way more often by players, we then get this logical outcome; One cannot be the best player if the character he's using is doing very poorly against popular characters. Those kinds of community driven effects are usually what players call the *metagame*. This present project will investigate the metagame of fighting games. I'll re-implement a model from Alexander Jaffe doctoral thesis [Jaf13] with some minor modifications while trying this model on real community of players. Afterwards, I created a modified

version of the utility function of a two players game which isn't linear to incorporate learning curves in the best response dynamic.

Chapter 2

First model

A very simple way to model fighting games is to only consider the choice of character of both players and to consider their payoff as the wining probability of each character in the matchup. These probabilities depend on the options available from each character and how well do they counter each other. These probabilities are commonly collected by the communities of players and are presented in matchup charts. These charts are in fact matrices containing information about each matchup. This is very fortunate as we'll use these matrices to obtain the bimatrix representing the game. This model is very general as it abstracts the complexity of a fighting game into a simple mathematical set up while still being powerful enough to make some predictions. Therefore, the model we will discuss will not be game specific and could be easily applied outside of the fighting game set up.

The game is symmetric since both players can only choose in the same set of character. Since the utility functions will be based of the winning probability of the players, the utility of player 2 (U_2) will be

$$1 - U_1$$

for any pure strategy. Therefore, this game is a zero-sum game. To

obtain the canonical form seen in class, simply *recenter* the game at zero (since here a draw happens when both utilities are 0.5). This gives rise to the following utility functions where x and y are the strategies of each player respectively.

$$\begin{aligned} U_1(x, y) &= x^T M y \\ U_2(x, y) &= y^T M x = 1 - U_1(x, y) \end{aligned}$$

Where M is the matchup chart.

Since we're working with a zero-sum game, we know that the threat strategies form a Nash equilibrium (as seen in class using the Minimax theorem). Therefore, we can find a Nash with the following LP:

$$\begin{aligned} &\max \alpha && && (2.1.1) \\ &s.t. \sum_r x_r = 1 && && (2.1.2) \\ &x_r \geq 0 && \forall r && (2.1.4) \\ &\alpha - \sum_r M_{rc} x_r \leq 0 && \forall c && (2.1.5) \end{aligned} \tag{2.1}$$

Since the game is symmetric, we know that at the equilibrium, both players will have a 50% win rate and a solution exists where both players play exactly the same strategy.

A totally legitimate critique of this model is that it's not representative of how players play in reality. People mostly practice a single character for years to learn all the intricacies of their *main*. As a character takes time to master, it is not expected that a player obtains the win rates indicated in the matchup chart as soon as he starts playing a character. We will address the learning process of the character in a further section. However, the Nash equilibrium of this game can still

be interesting for a slightly different reason as it gives us information about the community of players.

Indeed, let us consider the formation of a community of players as an iterative process where each player chooses a pure strategy (a single character) which maximizes his win rate against the pre-existing community and is lock to this character from this point on. This process exactly simulate the *fictitious play method* for finding a Nash equilibrium first introduce by George W. Brown in 1951. In this setup, a Nash equilibrium corresponds to an absorbent stationary point of the usage proportion of each character. Also in 1951, it has been shown [Rob51] that this process always converges to a Nash equilibrium in finite zero sum two players games. Therefore, the community distribution should look like the Nash equilibrium found by the LP(2.1)

We can now try this model empirically.

2.1 Implementation

The implementation was made in Julia and the models have been optimized with CPLEX. The code can be found on github at `https://github.com/Codsilla/MeleeMetagame`

As a case study, this project is focused on *Super Smash Bros. Melee*, a twenty years old game where the community had plenty of time to converge to a Nash equilibrium. Also, it's a fighting game with one of the biggest communities, based of the number of tournament viewers. To test the model, I needed statistics from the Melee community. Since it is an offline game, these statistics are hard to collect. Luckily, since last March, a community made online version of the game has been release. Moreover, since last August, a database has been created to collect all sort of statistics on the game and is graciously hosted by Scott Norton (a community member) on `https://slippistats.`

`online/`. After a few emails and phone calls with him, he agreed to let me consult the database and give me access to more than 1.3 billion matches! [1] I was then able to calculate the matchup chart and the character distribution empirically.

Here's what the community looks like:

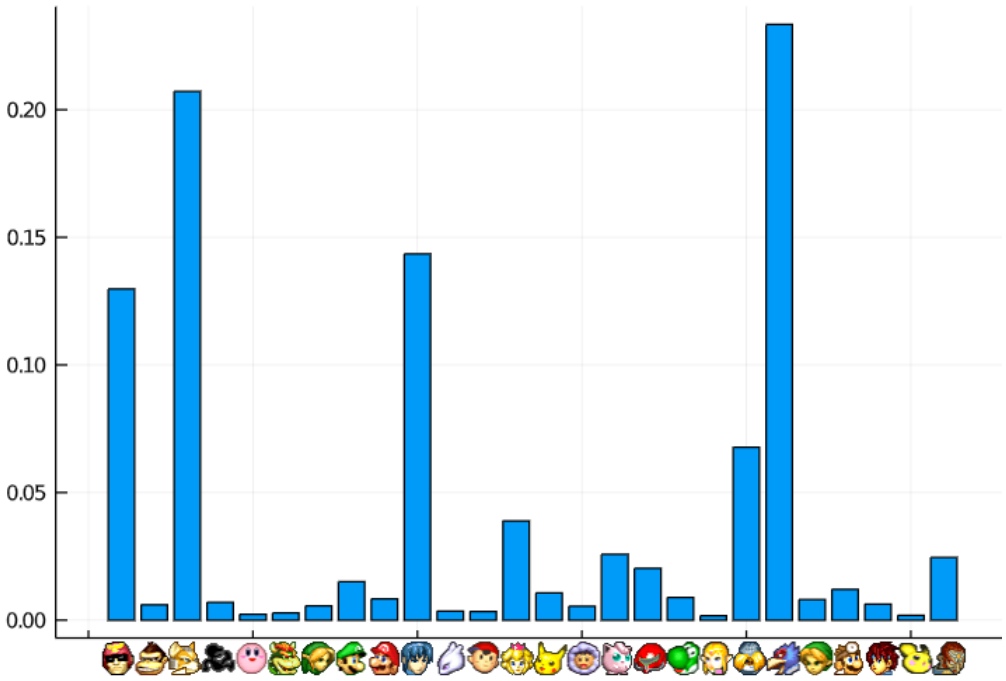


Figure 2.1: Community distribution

To illustrate the *matchup chart*, here's a heatmap that lets us visualize the data easily.

[1] It was my first experience with a database of this size. Therefore, I had to learn how to work with this data. In this instance, the database was implemented with MongoDB

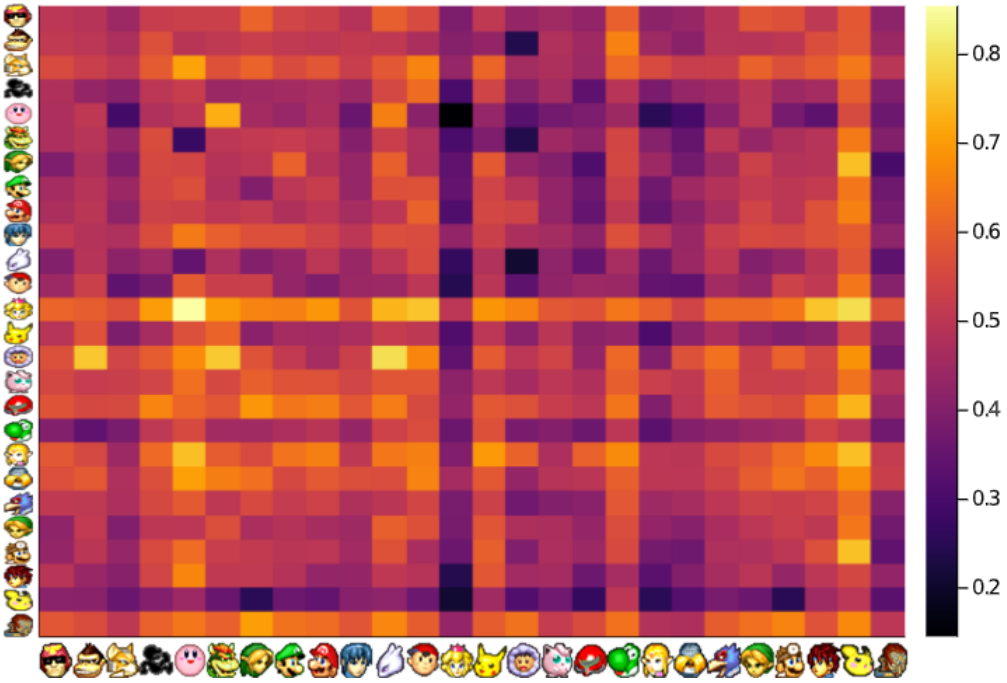


Figure 2.2: matchup chart

Here, we have a bright square where the row player has the advantage. Before we go any further, we can turn our attention to the above figure. Some characters are clearly very bad like *Pichu* (at the before last line) and we should not expect him to be played much in the Nash equilibrium. This can also observe in the community distribution above. We also notice that *Peach* wins every possible matchup. Therefore, we can expect her to also dominate the metagame, but that is not the case, as we can see in figure 2.1 and in the competitive scene of the game. This will be further discussed in the next section.

2.2 Removing the noise

Unfortunately, the database is made up of games where players fight each other without regard about skill level. This creates noise in the

data in two ways. First of all, a *matchup chart* changes according to the general skill level of the concerned players. For example, *Peach* is well known as an excellent character among low-level players, especially because one of her attacks is hard to counter[2] and very easy to do. According to the Pareto principle, a large portion of the players are not very good and therefore add a lot of noise to the data [New05]. Secondly, a good player can win against a bad player regardless of their character matchup. In this case, the model does not make any sense because the *matchup chart* is not representative of the actual victory chances.

To try and mitigate the effects, I had to filter all the matches where one of the players had less than a 50% win rate. In the other hand, a *theoretical matchup table* created by a group of very good players can be found at `https://www.ssbwiki.com/Character_matchup_(SSBM)`. I then converted that table to a matrix to use it as a comparison.

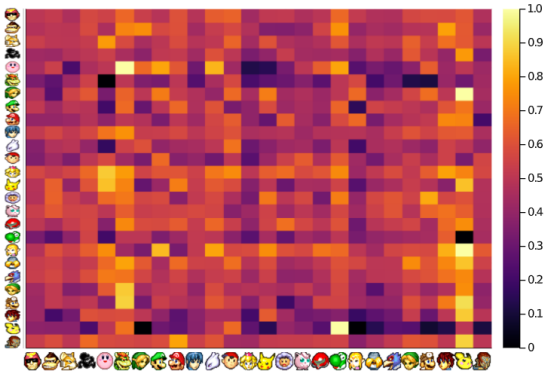


Figure 2.3: Filtered matchup chart

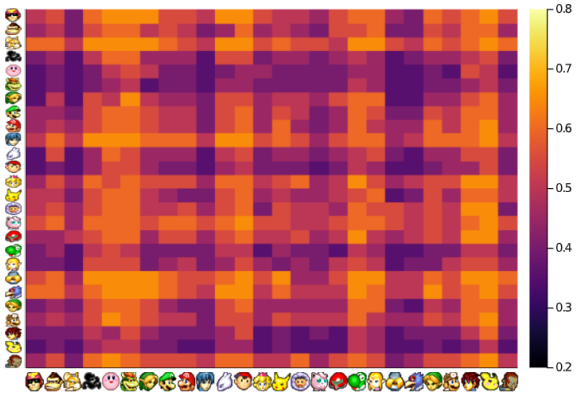


Figure 2.4: Theoretical matchup chart

We can also recalculate the community distribution.

[2]her *down-smash*

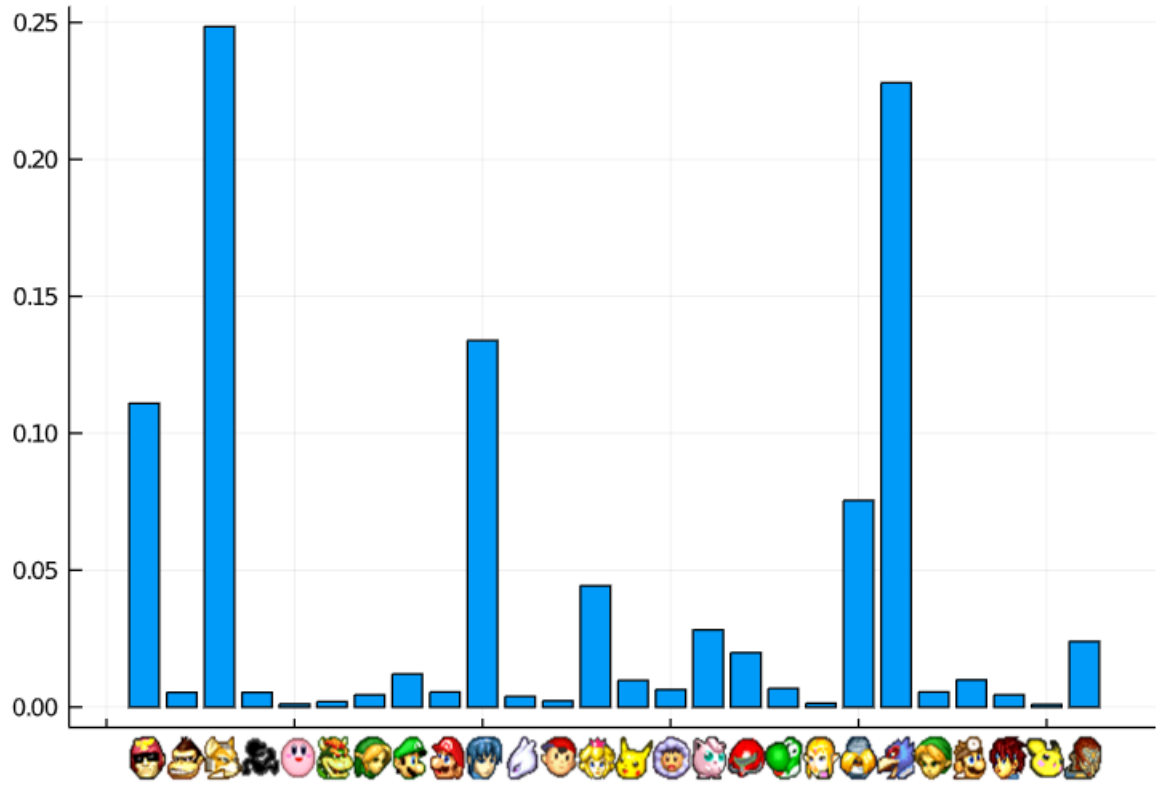


Figure 2.5: Filtered community

2.3 An approximate Nash equilibrium

The *matchup charts* obtained in the previous section cannot be taken as absolute truth. For example, those issued by experts are only precise up to 5%. Consequently, it would be preferable to not only obtain the exact Nash equilibrium of our game, but solutions close to that equilibrium as well. For example, if a strategy allows the player to reach a win rate of $49,9\%$, we can't discard that solution as being the true Nash equilibrium of our solution.

The objective would be to predict exactly what the community should look like, but we will restrain ourselves to a more modest goal;

For each character, we want to know the percentage interval in which it is possible to have an approximately stable community. In model 2.1, the optimal α is exactly $0,5$. Here we will impose a component of player 1's strategy and verify if he is able to reach about 50% win rate. Let i the fixed part of player 1's strategy be f_i, the new model then becomes

$$\begin{aligned}
&\max \alpha && && (2.1.1)\\
&s.t. \sum_r x_r = 1 && && (2.1.2)\\
&x_r \geq 0 && \forall r && (2.1.4)\\
&x_i = f_i && && (2.1.5)\\
&\alpha - \sum_r M_{rc} x_r \leq 0 && \forall c && (2.1.6)
\end{aligned} \tag{2.2}$$

By imposing an additional condition, we restrain the strategies available for the player 1 and thus his win rate will necessarily be smaller or equal than 50%. Here, we'll consider a win rate of $(50 \pm 0.5)\%$ as a stable win rate.

2.4 Results

First let's observe the equilibrium found with the matchup chart from *meleeStat*:

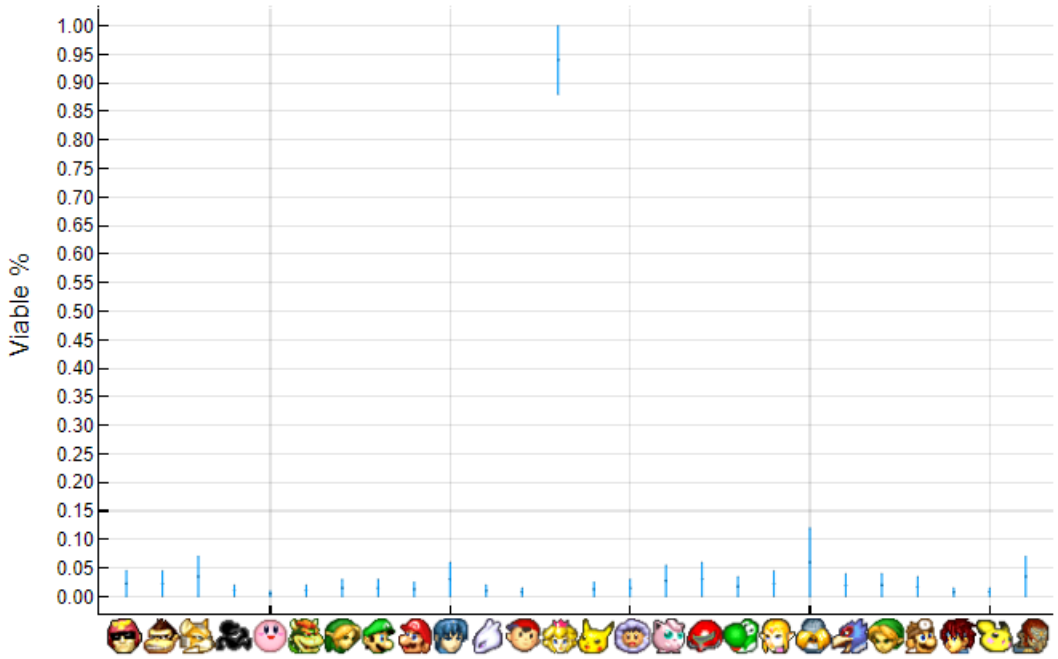


Figure 2.6: Viable distributions calculated with the empirical matchups

We conclude that the problems mentioned in the previous section still remain and *Peach* completely dominates the metagame.

Let's now focus on the results from the theoretical matchup chart.

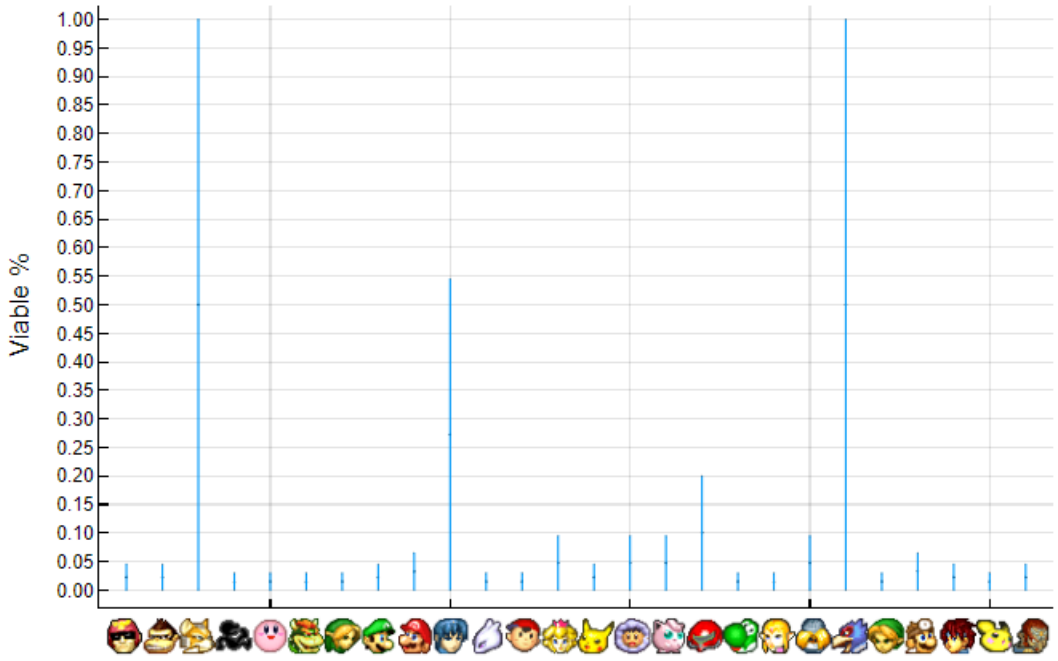


Figure 2.7: Viable distributions calculated with the theoretical matchups

These results are way more promising. In fact, for all the characters except one, the usage found in figure 2.5 falls within the viable ranges. Thus, we successfully captured the general distribution of the community.

We also notice that a player starting to play should probably choose *Peach* if he wants to rapidly gain a good win rate online as she dominates the metagame for players with low skill level. Since we have access to the real distribution, we can verify this claim by using the

following model:

$$\begin{aligned} &\max\ x^T M y && (2.3.1)\\ &s.t. \sum_r x_r = 1 && (2.3.2)\\ &x_r \geq 0 \qquad \forall r && (2.3.3) \end{aligned} \tag{2.3}$$

Clearly, by the linearity of the utility, the solution of this model will simply be a onehot vector with a 1 for the character which has the greatest win rate against the community. Using this model with the two matchup charts, we obtain two different pure strategies. Consequently, a new low skill player should play *Peach* and high skill players should play *Fox*.[3] In the following section, we will investigate how this transition happens.

[3]These solutions were computed on the distribution of player with win rates above and below 50% respectively

Chapter 3

Learning curve model

In the preceding model, the utility obtained by playing a character had a linear progression based on the time invested playing that character. As this was not realistic, we should expect to see a learning curve with *diminishing returns*. In fact, the progression obtained with a character on our one-thousandth game hour is not the same as in the first. Also, different characters have different learning curves as some have a lower skill barrier than others. Here, we will take as granted that our player wants to win as much as possible after playing z hours. To model this phenomenon, we will modify the utility of our player.

$$U_1(x, y) = \sum_i Lr_i(x)(My)_i$$

where Lr_i is a learning curve of the character i. Here x represents proportion distribution of the z hours spent on each character.

Our model then becomes

$$\begin{aligned} \max \sum_i Lr_i(x)(My)_i & \qquad (3.1.1) \\ s.t. \sum_i x_i = 1 & \qquad (3.1.2) \\ x_i \geq 0 \qquad \forall i & \qquad (3.1.3) \end{aligned} \tag{3.1}$$

Unfortunately, this model is not linear because the function (3.1.1) is not necessarily linear.

A solution to this problem is to transform this NLP into a MILP which approximate our model. To do so, we will approximate the objective function. Since this function is separable over its variables, we can simply approximate the Lr_i with univariate piecewise linear functions. To do so, we will separate x_i into the corresponding quantity over the right piece of the linearization of Lr_i

To illustrate this method, we will do an example on a learning curve Lr_i. Let the linearization of Lr_i which contains k pieces starting at $s_0, s_1, \ldots,$ ending at $e_0, e_1 \ldots$ with slopes $a_0, a_1, \ldots$ and initial values $b_0, b_1, \ldots.$ We proceed the following way:

$$
Lr_i(x) \equiv \begin{cases} \sum_j a_k x_{ij} + b_j y_i j & (3.2.1) \\ s.t. \sum_i x_{ij} = x_i & (3.2.2) \\ x_{i,j} - s_j \; y_{i,j} \geq 0 \quad \forall j & (3.2.3) \\ x_{i,j} - e_j \; y_{i,j} \leq 0 \quad \forall j & (3.2.4) \\ x_{ij} \geq 0 \quad \forall j & (3.2.5) \\ \sum y_{ij} = 1 & (3.2.6) \\ y_{ij} \in \{0,1\} \quad \forall j & (3.2.7) \end{cases} \tag{3.2}
$$

The constraint (3.2.6) impose that only one piece of the linearization has a non zero impact on (3.2.1). Moreover, the constraints (3.2.3) and (3.2.4) force that the right piece of the linearization is chosen. In other words, x_{ij} can only be greater then 0 if x_{ij} is between s_i and e_i and when y_{ij} is 1. Finally, (3.2.1) is simply the linear approximation of our function.

We can now repeat this process for every character. The resulting model is cumbersome and not very insightful as a lot of variables are involved. Therefore, the formal definition of the full model is omitted.

3.1 Implementation and results

The full implementation of this model is also available on github at `https://github.com/Codsilla/MeleeMetagame`

Sadly, I was not able to obtain the learning curves (or even approximations) for this problem. However, to demonstrate the feasibility of the method, I used fictitious functions. Base on the literature about learning of learning curves, it is reasonable to predict that the learn-

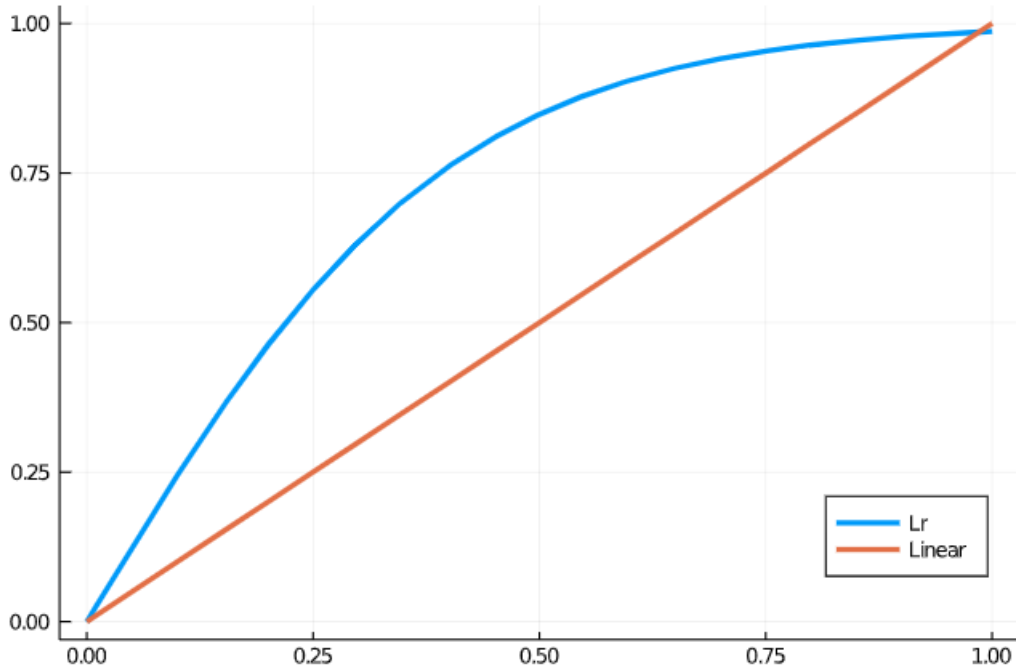


Figure 3.1: Learning rate

ing curves should look like sigmoid functions [LBEK10]. To simplify the implementation slightly, I used the same learning curves for every character. Precisely, I used

$$Lr_i(x) = Lr(x) = Lr = 2\sigma\left(5x\right) - 1$$

Where σ is a sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

For the linearization, I used the package LinA that I wrote and that used an algorithm that I conceived during a research internship. [1] With this package, we can guarantee a maximal relative error over each

[1]The draft of the paper associated with this project can be found here `https://www.overleaf.com/read/mnxzqjghmtsq`

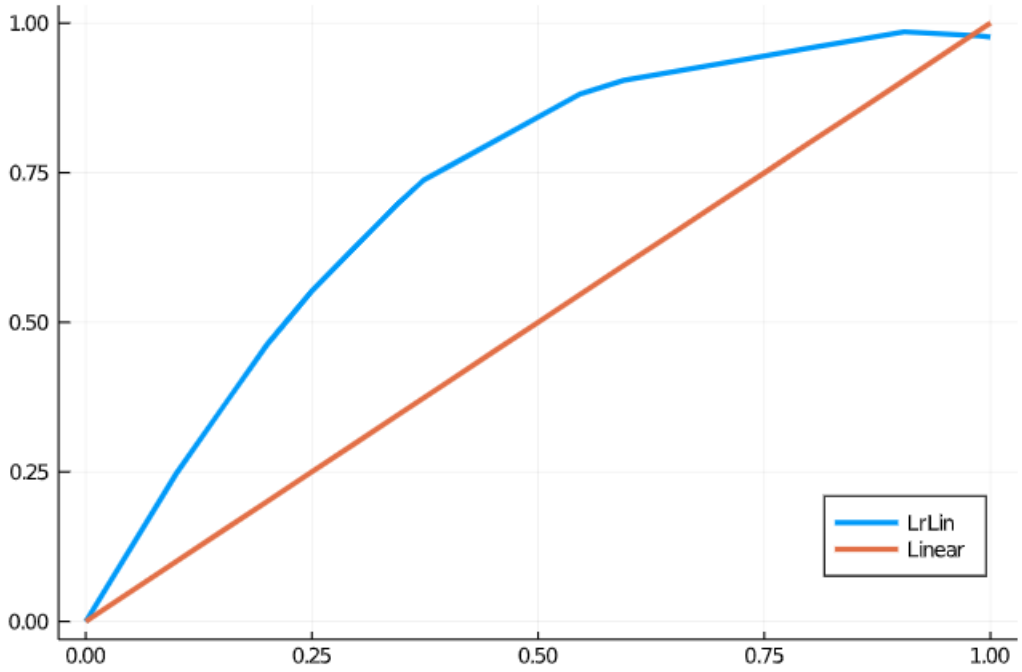
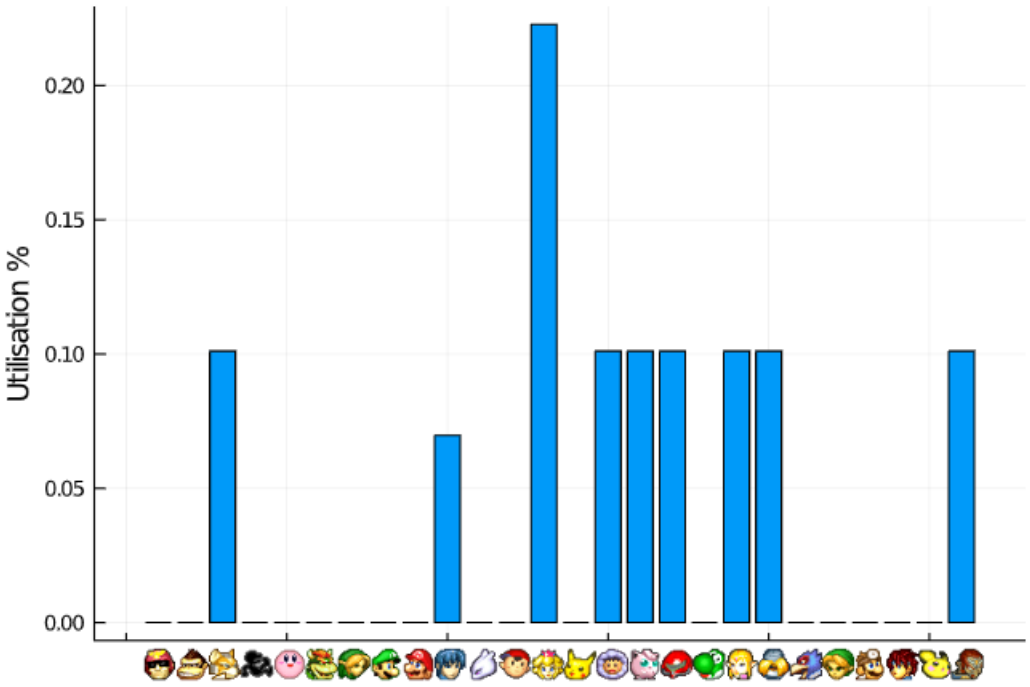


Figure 3.2: Linearized learning rate with a maximal error of 1%

of the learning curves. This is useful as relative errors are preserved through error propagation over a sum.

Here the linearization is computed with a maximal error of 1%. Therefore, the optimal solution of this model will be within 1% of the solution of model 3.1. By solving this MILP with the empirical matchup table, we obtain:



We notice that even though *Peach* dominates the matchup table, our model doesn't output a pure strategy. This show that the *diminishing return* of the utility impacted the character practice time distribution as desired.

Note In no way am I pretending that this graph is representative of reality.

Chapter 4

Conclusions and perspectives

The models shown in this project seem to give promising results and might be used to predict the metagame of a fighting game. This is important for game developers to make *balanced* games (where every character is viable to some extend). In the near future, the *rank mode* will come out for super smash melee online and I will be able to obtain less noisy data as a player skill metric will be available. This could yield interesting results that could really spark some discussion in the melee community. In the long term, further work should be done to obtain learning curve estimation from the data as it would help to maximize the wining rate of player with a fixed amount of time to practice.

Bibliography

[Jaf13] Alexander Benjamin Jaffe. *Understanding Game Balance with Quantitative Methods*. Thesis, July 2013.

[LBEK10] Nathaniel Leibowitz, Barak Baum, Giora Enden, and Amir Karniel. The exponential learning equation as a function of successful trials results in sigmoid performance. *Journal of Mathematical Psychology*, 54(3):338–340, June 2010.

[New05] M. E. J. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46(5):323–351, September 2005. arXiv: cond-mat/0412004.

[Rob51] Julia Robinson. An Iterative Method of Solving a Game. *Annals of Mathematics*, 54(2):296–301, 1951.