TRAINING A GENETIC PROGRAMMING TO FIND THE MOST OPTIMAL NHL DRAFT STATISTICS

Queen's Sports Analytics Organization

Makan Sabeti

School of Computing

Queen's University

December 2021

Table of Contents

1 Problem Formulation

1.1 Background

As AI algorithms advance, we see people from all different industries begin and adapt applications for this technology. We have seen this adaptation occur in all different types of businesses. Industries that have been going along with the "old-fashion" methods (like sports mangers) have even adapted this technology, in order to get a competitive advantage. The topic of adaptation of AI and data analytics in sports is a very interesting and unexpected one. You would think that in an industry in which people are judged on numbers and statistics, would have accepted new approaches over time to help them perform better, but that is not the case. In particular, the first case of advanced sports analytics occurred in 1980 when George William James released a research paper about a new analytic strategy for baseball called sabermetrics. This research is the core of sports analytics and how teams judge players nowadays, but back then Mr. James was ridiculed and called a disgrace by most high-level analyst. The reason for this is because analysts back then had been using the same metrics and strategies for decades of time and therefore were skeptical of any new research that surfaced. They believed that they had it all figured out and shrugged off any discussions of new methodologies. This continued until 2002, when then Oakland Athletics manger Billy Beane had enough of the old strategies as none of them were working for him. He dug up Mr. James old research and implemented it that season to his team that was projected to be one of the worst that year. He managed to take a team full of players that other team's thought were useless, all the way to the finals. His strategies got the attention of mangers around baseball, then slowly got the eye of other sports, one of which was hockey.

Hockey/NHL is actually a bit late to this new style of analytics compared to baseball. Most teams only using these new techniques as recently as the last half decade. To this day there still remains some resistance for some. However the franchises that do emphasizes the importance have really proved to mangers/owners the importance of evolution and accepting new practices. One of those new practices that is being implemented is the use of algorithmic problem solving within the entire organizational process. From business strategies to player signings, there is a huge potential use for technology to help find the most optimal solution. At the end of the day, building a successful sports franchise is all about numbers and finding the most effective ways of interpreting those numbers.

1.2 GP Reasoning

Every year in the NHL, teams pick players who just turned 18 to play for their team in draft. There are exactly 7 rounds of this draft and in each, every team can pick once. The order of the draft is determined by where you finish in the standings. The lower you finish the higher you are going to draft and vice versa. Now although there are 7 rounds, the most crucial is going to be the first round. The reason for this is that the first round is going to be where all the franchise changing players will be picked. Every other round is less meaningful, due to the unlikely hood of those players being any good and helping the franchise long term. For this reason, majority of the scouting process goes into the first round. Figure 1 really puts it into perspective. 74 per cent of first round picks go onto playing in the NHL, but that number drops to 34 per cent for the second round and goes as low as 11 per cent in the seventh round. It is for this reason; mistakes are not acceptable in the first round. Out of the 31 players available, you need to make sure you pick the one's that will play.

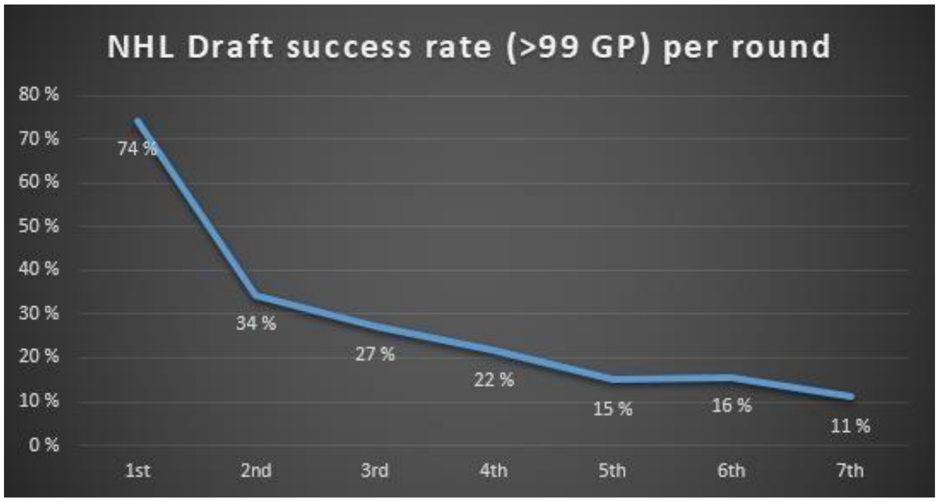


Figure 1

The genetic program that I have decided to implement will take over the all the scout's job and decide given different player pools, what the best order is for the players to get drafted. The exact process of the algorithm will be referenced more in the next section of the report, but essentially it will draft players based on the players statistics and use drafts in the past as a training tool. The reason that this is an ideal candidate for a genetic program is that we can represent this problem and the solution to the problem very easily in evolutionary optimization. As mentioned before, this industry is simply finding a way to optimize the numbers in front of you. Drafting is the same. We will have a population of drafts and using the training data we can learn and give an accurate fitness score to the draft, which will help us conclude what type of statistics we would be ideal at certain points in the draft. What this is essentially doing is getting players in the best order possible for the draft and showing NHL teams, that if you have the 15th overall pick, you should pick a player with these statistics if you want to maximize your potential for them to be one of the 74 per cent of players that play. For example, if you are a team and have the 20th overall pick, you can run this genetic program and look at the 20th overall position and notice that you need to pick a player that has x statistics.

2 GA Design

2.1 Representation

For the design I believe it's best to go through it the same way that I outlined it before I started. The first thing that I had to do is decide how I was going to represent the individuals and population. It made sense that each of the individuals should be a draft because for the fitness function they would have to be given scores and trained from previous drafts, versus individual players that would be used/modified to help find the most optimal form of a draft. Now, in order to represent this, I referred back to a similar method that was implemented in assignment 2 of our course. In that project we had the population as a list of lists of lists, with each individual being a list of lists and the individual would be made up of instructions. Although we are not going to be implementing a linear genetic program, a similar representation will work. Therefore, our individual representation is going to be a list of lists that will be made up of 31 players, which is the draft size for the first round. Each player will have a number that they were originally projected to be drafted in, their position, the number of points they put up and the number of games they have played. This would infer that the population would be a list of lists of lists made up of drafts represented as individuals.

This representation is going to work, but while researching for upcoming players I realized that the data of those players under 18 (which all of the prospects currently are) is protected and can only be reached by those who have authority or by paying a subscription fee. This led to having to generate a prospect player pool, which is occurs in the initialization class. Although, the players statistics had to be generated randomly, I did a lot of research of what they should look like and generated realistic statistics. For example, I looked at previous prospects and the number of points they had when they got drafted in the first round and noticed that on average the lowest number of points for a forward was 30 and the lowest for a defenceman was 28. I also look for the local maximum and figured out that the highest average of points for a forward was 100 and 55 for defenders. Therefore, I initialized a random number within those ranges keeping in mind a player's position.

The next pattern that I noticed in my research is that defencemen tend to get exponentially less points than forwards but are still often drafted in higher positions. This is due to the fact that the priority for their position is to defend. To combat this problem, I took the average number of points that a defenceman would need to be ahead of the closest forward to them. That average came to be roughly 25 points. This meant that if I give the defenceman 25 extra points for being a defenceman than I could get a better estimate of how they would be compared to the forwards in terms of points. This can be shown in Figure 3. In terms of games played, it usually depends on the number of points the player gets. Those with over 85 points typically played over 300 games, those within 50-85 played between 100-299 and those with less than 50 points played between 60-100. There usually are no players drafted in the first round who have played under 60 games.

Before points addition

Number	Position	Points	Games
1	F	100	400
2	D	70	350
3	F	95	325
4	D	50	300
5	F	70	250

After points addition

Number	Position	Points	Games
1	F	100	400
2	D	95	350
3	F	95	325
4	D	75	300
5	F	70	250

Figure 2

As for the training data, I was able to web scrape it from HockeyDB.com which had all the statistics I was looking for in order for the past drafts. The only issue that I had with the web scraping was the number of teams in the NHL for different years changed. Since every team gets a pick, when they add a new team to the league the draft size increases. It used to be 30 teams for a while, then in 2017 they added the Las Vegas Golden Knights and the league expanded to 31. This meant that I had to account for that when I was web scraping by taking an extra player before 2017.

2.2 Fitness

The fitness strategy was by far the most interesting part of this problem. I knew that there were three steps that I had to account for. First, to incorporate some sort of equation where at the end I could give a score to individual players. Second, evaluate the players in the training data by the same equation. Finally, do a comparison that will result in a fitness score. Admittedly, there were a lot of solutions that I implmented for the first step, but only three were interesting. The first was simply giving a score based on points per game. In theory this could be a good idea, but then I noticed it would not be using the points or games for what I believe it should be because in that scenario playing more games will bring your point per game down, which would be a negative. However, in real life the more games you play is a positive (whether you score or not) because it is a bigger sample size and shows NHL teams that you have durability and the ability to play more games. For that reason, that implementation was thrown out quickly. The next strategy was that you could compare two players next to each other. If the player B was drafted after the player A and had more points than player A then we count that as a mistake and if player B also has played more games than player A then its another mistake. A draft would then be considered by how many mistakes it makes. This was a solid idea, but it could be improved upon, which leads to the third idea. The third idea was that instead of comparing player B with just player A, why not compare him with every player drafted before him and do this for every player in the draft. This would give a draft a score that we can use to compare with the training data. The training data itself, would also use this method to judge the draft but instead of returning one number, it would return a list of numbers (one for each draft in the training data). This leads us to the last step, which is the comparison. The approach taken for this was to do sum of error. The reason I did this was because sum of error is an optimal technique used for comparative training samples. The model will be able to learn and produce better results through the evolutionary process. The way that this was calculated was by subtracting the value from the generated draft with every value in the training draft and adding all of them up. The result of that was the overall fitness score that was returned.

2.3 Recombination

For the crossover I decided to implement a cut and cross-fill method, that is common in genetic algorithms. This method is very similar to single point crossover; however, the difference is that both parents get cut on the same index. What this does is it keeps the offspring length the same as the parent length, which is what we need in this example. We are trying to get two offspring from two parents but keep the length to 31. Single crossover would select a different point on each parent and cut, which would mean that the individual lengths would be different depending on where the cut would happen on each parent. This would be ideal in a situation that does not have a set size. When your problem requires you to have a set size, cut and cross-fill is the best alternative.

2.4 Mutation

The mutation for this algorithm was straight forward once, I researched and determined what mutation method is best for the particular algorithm. It came to be that a tree swap strategy was the most optimal. What that means is that if mutation is called, a random index will be called for both parents and the data in those parents will be stored for each respective parent. Then, the next move will be to swap the value stored at the two indices of the parents. The result of this process would be the mutated individual.

2.5 Parent Selection

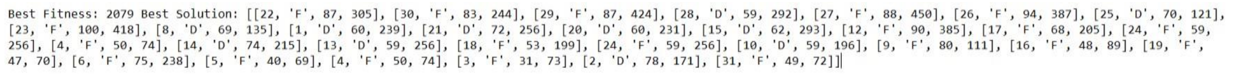
When coming up with the most optimal parent strategy I did struggle a bit as I knew there are plenty of methods to pick from. The options that I was contemplating were random uniform selection, fitness proportionate selection, and tournament selection. After implementing all three I noticed the difference in results. Random uniform selection was less than optimal when coming up with the best parents as they are just picked randomly, which means the odds of it selecting the best possible parents is low. This forces this method to only really be effective through multiple runs. Then I implemented fitness proportionate selection, which is traditionally used in genetic programming. What this method does is it uses the fitness function to assign a fitness value to a chromosome then uses that fitness value to associate a probability of selection with an individual chromosome. The problem that I had with this was that it was still too stochastic, which made it less optimal (similar issue to random uniform). The final method that I implmented was tournament without replacement selection. This method consists of running a tournament of any size passed in (in this case 4) involving different potential parent solutions. The parents with the highest fitness values will get selected from the tournament and passed through as parents. In this case, there is far less stochastic noise as we can select and compare in groups, which will increase our likelihood of picking the best parents.

2.6 Survivor Selection

For survivor selection there were two main strategies that I was trying to choose from: generational replacement and random uniform fitness. Now, although generational replacement is the one commonly used for most genetic programs, I remembered that we implemented random uniform for assignment 1 and it worked surprising well. However, after implementing both strategies, generational replacement ended up giving better outputs more often. This is due to the stochasticity of random uniform. Generational replacement on the other had has no element of randomness to it. The strategy involves ordering the parents by highest fitness to lowest and replacing with the worst parents with the offspring. Therefore, this strategy involves almost no stochasticity and is a far better solution for this problem.

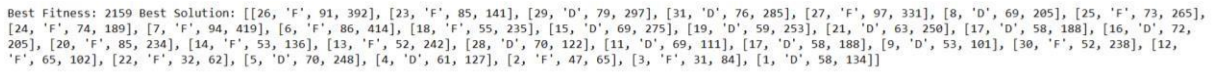
3 EA Results

The results from the algorithm described above ended up being accurate and running in good time (about 30 seconds per run of 50 generations). Figures 5, 6, and 7 go through three different results from the algorithm.



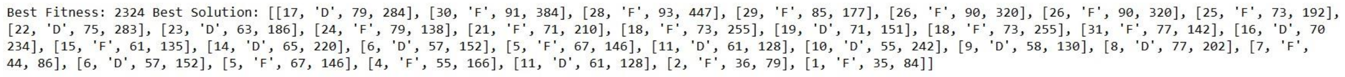
```
Best Fitness: 2079 Best Solution: [[22, 'F', 87, 305], [30, 'F', 83, 244], [29, 'F', 87, 424], [28, 'D', 59, 292], [27, 'F', 88, 450], [26, 'F', 94, 387], [25, 'D', 70, 121],
[23, 'F', 100, 418], [8, 'D', 69, 135], [1, 'D', 60, 239], [21, 'D', 72, 256], [20, 'D', 60, 231], [15, 'D', 62, 293], [12, 'F', 90, 385], [17, 'F', 68, 205], [24, 'F', 59,
256], [4, 'F', 50, 74], [14, 'D', 74, 215], [13, 'D', 59, 256], [18, 'F', 53, 199], [24, 'F', 59, 256], [10, 'D', 59, 196], [9, 'F', 80, 111], [16, 'F', 48, 89], [19, 'F',
47, 70], [6, 'F', 75, 238], [5, 'F', 40, 69], [4, 'F', 50, 74], [3, 'F', 31, 73], [2, 'D', 78, 171], [31, 'F', 49, 72]]
```

Figure 3



```
Best Fitness: 2159 Best Solution: [[26, 'F', 91, 392], [23, 'F', 85, 141], [29, 'D', 79, 297], [31, 'D', 76, 285], [27, 'F', 97, 331], [8, 'D', 69, 205], [25, 'F', 73, 265],
[24, 'F', 74, 189], [7, 'F', 94, 419], [6, 'F', 86, 414], [18, 'F', 55, 235], [15, 'D', 69, 275], [19, 'D', 59, 253], [21, 'D', 63, 250], [17, 'D', 58, 188], [16, 'D', 72,
205], [20, 'F', 85, 234], [14, 'F', 53, 136], [13, 'F', 52, 242], [28, 'D', 70, 122], [11, 'D', 69, 111], [17, 'D', 58, 188], [9, 'D', 53, 101], [30, 'F', 52, 238], [12,
'F', 65, 102], [22, 'F', 32, 62], [5, 'D', 70, 248], [4, 'D', 61, 127], [2, 'F', 47, 65], [3, 'F', 31, 84], [1, 'D', 58, 134]]
```

Figure 4



```
Best Fitness: 2324 Best Solution: [[17, 'D', 79, 284], [30, 'F', 91, 384], [28, 'F', 93, 447], [29, 'F', 85, 177], [26, 'F', 90, 320], [26, 'F', 90, 320], [25, 'F', 73, 192],
[22, 'D', 75, 283], [23, 'D', 63, 186], [24, 'F', 79, 138], [21, 'F', 71, 210], [18, 'F', 73, 255], [19, 'D', 71, 151], [18, 'F', 73, 255], [31, 'F', 77, 142], [16, 'D', 70
234], [15, 'F', 61, 135], [14, 'D', 65, 220], [6, 'D', 57, 152], [5, 'F', 67, 146], [11, 'D', 61, 128], [10, 'D', 55, 242], [9, 'D', 58, 130], [8, 'D', 77, 202], [7, 'F',
44, 86], [6, 'D', 57, 152], [5, 'F', 67, 146], [4, 'F', 55, 166], [11, 'D', 61, 128], [2, 'F', 36, 79], [1, 'F', 35, 84]]
```

Figure 5

As you can see the algorithm does a good job of classifying which players should be in the top half and which should be in the bottom. Although we cannot know the exact accuracy (due to the fact that there's no definitive way of knowing why a certain player is at a certain place), we know that typically the players with over 80 points and over 300 games played should be ranked amongst the early picks. Alternatively, we know that players with lower points and lower games should be amongst the latest picks. Another aspect that must be considered when checking the effectiveness is looking at if the results logically get better with a higher fitness score. Of the three figures, figure 5 has the lowest fitness which means that there are more issues with it than other higher fitness's. Immediately looking at the results we can see a player with 59 points and 292 games drafted fourth overall. This could realistically happen, however even when looking at the training data, the top five players generally have between 70-100 points and roughly 300 games. There are a few other questionable placements, but nothing crazy that would make me question the accuracy. Next, looking at figure 6 we notice a higher fitness and almost no logical mistakes. The only questionable output would be having a 94-point player with 419 games at the 9th overall pick. Logically, it would make sense to have the player in the top 5, but that is a better mistake to make than figure 5. It would generally be better for the algorithm to pick a slightly better player at a spot than for it to pick a player that is not good enough for a spot. The reason for this is that top 5 picks are very rare to get and making mistakes there could set a franchise back many years. Lastly, figure 7 had the best fitness out of all the other figures, therefore the mistakes should be very limited, and they do seem to be that way. There is no player placement there that would be shocking. Most of the top tier players have good points and games, and most of the lower tier players have less points and less games.

Now that we went over the improved accuracy of the results as the fitness gets better, let's go over how often we are getting a good fitness. The graphs on Figure 8 represent how many times the best fitness score at termination of the program fell into a specific range, this was done for 30 generations and 50 generations respectively. Important takeaways to note from these two graphs are how much the fitness improves after more generations, how constant the fitness's are, and whether there is an outstanding majority for one range. Firstly, as illustrated from the two graphs range, the overall fitness range improves within the 20 extra generations. We can notice the minimum and maximum ranges have been raised from 1900-1975 to 2000-2099 and 2201-2275 to 2400-2499, respectively. Second, the majority of overall fitness from both graphs seem to be in the ranges of 2100-2299. This shows consistency with the overall program as the probability is that it will output fitness's in that range. Finally, we must look for any big bars that would indicate a possible spike for a certain range. Both graphs show no signs of that and seem to be consistent as there are no real spikes. This shows that the algorithm is not stochastic and there is not a lot of randomness. Which would confirm that the selection methods that were implemented are working as they should. If the contrary were to be true, the graphs would look more like a nominal distribution.

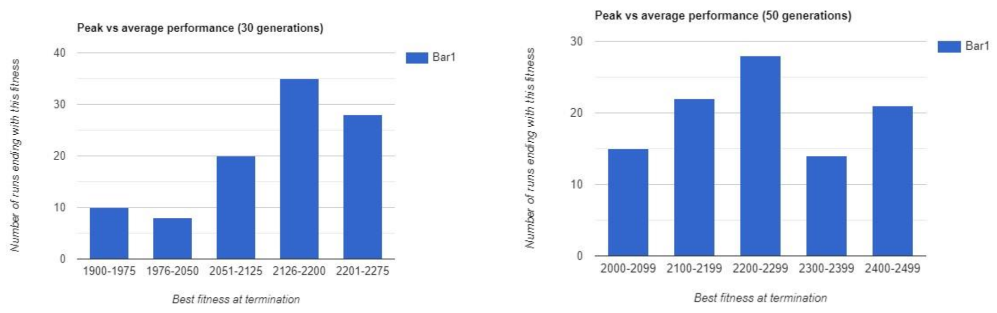


Figure 6

4 Comparison

The way that I came up with this problem idea was actually a very interesting one. I was talking to a few people who are in analytical departments for different NHL teams, and they were telling me about the importance of AI implementations. After research a little on different projects, I immediately thought of this. The reason is that I release how important drafting well is for teams who want to remain competitive. You have to be able to draft good players even with high picks in order to build a dynasty.

At first, I could not find anything about using any algorithms to find ideal NHL drafts. However, after doing more exploring on the idea, I came across an article by Gerrit Hall titled when genetic algorithms met Giannis Antetokounmpo. The article went over how a genetic algorithm was used to draft a set of prospects into the league using a similar strategy to my algorithm, but for basketball. However, the article did not release any code or results, which meant I could not really compare that part to my implementation.

Although a similar overall strategy was implemented, there are a lot of noticeable difference between making a genetic algorithm for hockey vs basketball. The first is the fitness calculation and specifically weighting of positions. In basketball drafts, certain positions take a lot higher priority than others (for example a point guard is going to get drafted a lot higher than a centre in most cases). Therefore, for the NBA algorithm they had to implement a lot of different weights on positions in order to calculate the fitness, meanwhile in hockey its not really like that. The forwards and defenceman are drafted very evenly, the only difference would come at how scout's view their points. As mentioned in the representation part of the report, defenceman typically have 25 less points than their closest forward, which is why we added 25 extra points to each defenceman.

Another noticeable difference between the two strategy is the resources available. Basketball tends to be a more data recorded and data public sport, as someone can go and see the statistics of players at any age. Hockey is a lot more private. This is likely due to the process in which young players make the NBA versus NHL. Most NBA players have been in leagues with their peers as early as ages of 10 years old. The United States typically do a very thorough scouting process and want potential future players to be playing against the best in their age from a young age. Hockey is a lot more diverse in that sense. Players in the NHL typically come from many different countries and are not really capable of playing against the best competition until ages of 16 or 17. A lot of those leagues that the players play in as well do not publicly share the data for free. There tends to be a subscription fee for most leagues, which highly effected my algorithm as I had to generate my own data. Therefore, I could not predict the draft order for the exact player, but I could predict what a player's statistic could look like.

5 Discussion

5.1 GP solution and advantages

This algorithm has done very well for what was available to it. The solution was able to get trained and output an optimal draft strategy. Looking back at it now, it really goes to show how well this problem is fitted for evolutionary optimization. Every draft and player were able to be represented well as a population and individuals. Moreover, the objective was very clearly solvable using evolution as there were multiple factors that go into a draft and an evolutionary algorithm was able to be trained and produce the output based on that. If this solution had to be an ordered draft based on points, then a greedy search would have been a better use, however there was a lot more involved, and the structure of an evolutionary algorithm was able to account for that.

As for the importance of this solution, it is clearly something that NHL teams are going to be looking at using. In order to win games and build a dynasty you have to draft well. The salary cap usually forces teams to lose their high salary players and you need to have a system that can replace those players with younger ones, that are on a lower salary. The way to do that is to not miss pick in a draft. Teams need to make sure that the players that they pick are going to be able to play in the NHL when ready. This algorithm will be able to reassure scouts about their due diligence on players and make sure the players they pick play in the NHL.

5.2 Shortcomings

One of the biggest goals that I had when coming up with this idea was to create an algorithm that can mimic and/or replace a professional scout's process when formulating players for the draft. I believe I have accomplished this, but in a rudimentary way. The reason I say rudimentary is because my algorithm works exactly how it is supposed to for the resources that were available. However, when drafting players scout's look at statistics like expected goals, corsi, wins above replacement, faceoff percentage, number of primary assists versus secondary, and many more along with the ones I implemented. After taking all this in, a scout would formulate placements of the players. My model could not take all this into consideration due to the lack of resources, however the logic would be the same if these resources were available. All I would do is account for these new parameters in the fitness. The logic to calculate a score for each new players and then train it from previous drafts would be the same. The previous drafts would also need to have the same data available as the one used for the new players. If however, all of this information was available, then the model would have a very high accuracy and could be implemented by NHL teams.

Reference:

Nevalainen, Jokke. “NHL Draft Pick Probabilities.” *DobberProspects*, https://dobberprospects.com/2020/05/16/nhl-draft-pick-probabilities/.

“Genetic Algorithms - Parent Selection.” *Genetic Algorithms - Parent Selection*, https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm#:~:text=In%20K%2DWay%20tournament%20selection,work%20with%20negative%20fitness%20values.

Hall, Gerrit. “When Genetic Algorithms Met Giannis Antetokounmpo.” *Medium*, Medium, 13 Oct. 2018, https://medium.com/@gerrithall/when-genetic-algorithms-met-giannisantetokounmpo-8ad22b805d5a.

Error Sum of Squares, https://hlab.stanford.edu/brian/error_sum_of_squares.html.

Dagdia, Zaineb Chelly, and Miroslav Mirchev. “Point Crossover.” *Point Crossover - an Overview | ScienceDirect Topics*, https://www.sciencedirect.com/topics/computerscience/point-crossover.