Decision making in machine learning : a practical programming application using blackjack

Summary

This thesis presents the concepts of decision making and how this is implemented in machine learning and demonstrates this using a computer program written in python to provide a time based plot of improved decision making applied to the card game blackjack(pontoon)

Introduction

Decision making

Importance of separating decisions from outcomes, we can make good decisions with bad outcomes and vice versa. Eg Kev sinffield kicking for 2 in last minute of play offs v huddersfield needing 7 to win when a try and dg would have won it. Good outcome questionable decision. How do we evaluate or rate our decisions? We use our experience to evaluate decisions and develop strategies for making decisions, example when you are born you know nothgin and make uninformed decisions as you get older you evaluate what happened the last time you made that deacons (or similar - but when is it similar) we preserve a memory and recall what happened previously

Decision Making in computing

Artificial intelligence simulates experience in computers to make them behave in a more human or intelligent way, commonly seen in computer games

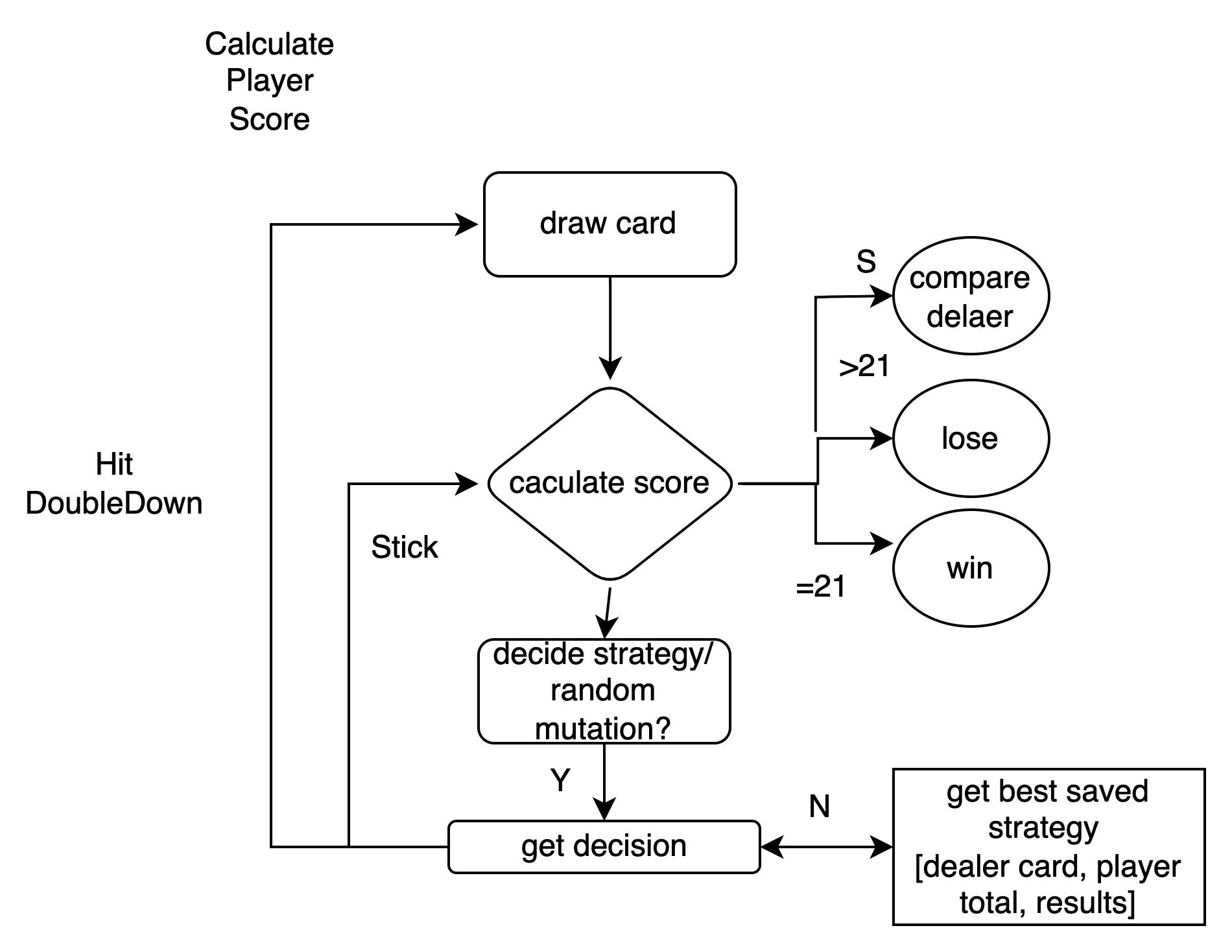
Machine learning we provide a state to an algorithm or decision making function which is used to provide experience in a human memory type way to inform next decisions. This is seen in data analysis, pattern matching and excels in analysing unknown variables to give predictive behaviour. It relies on us providing a score or measure of success to a decision. For instance on internet shopping a suggestion to buy another item based on current basket is initially random 3 items offered but over time the machine remembers the success of click thru or further our purchase then favours the offer of this item on similar baskets.

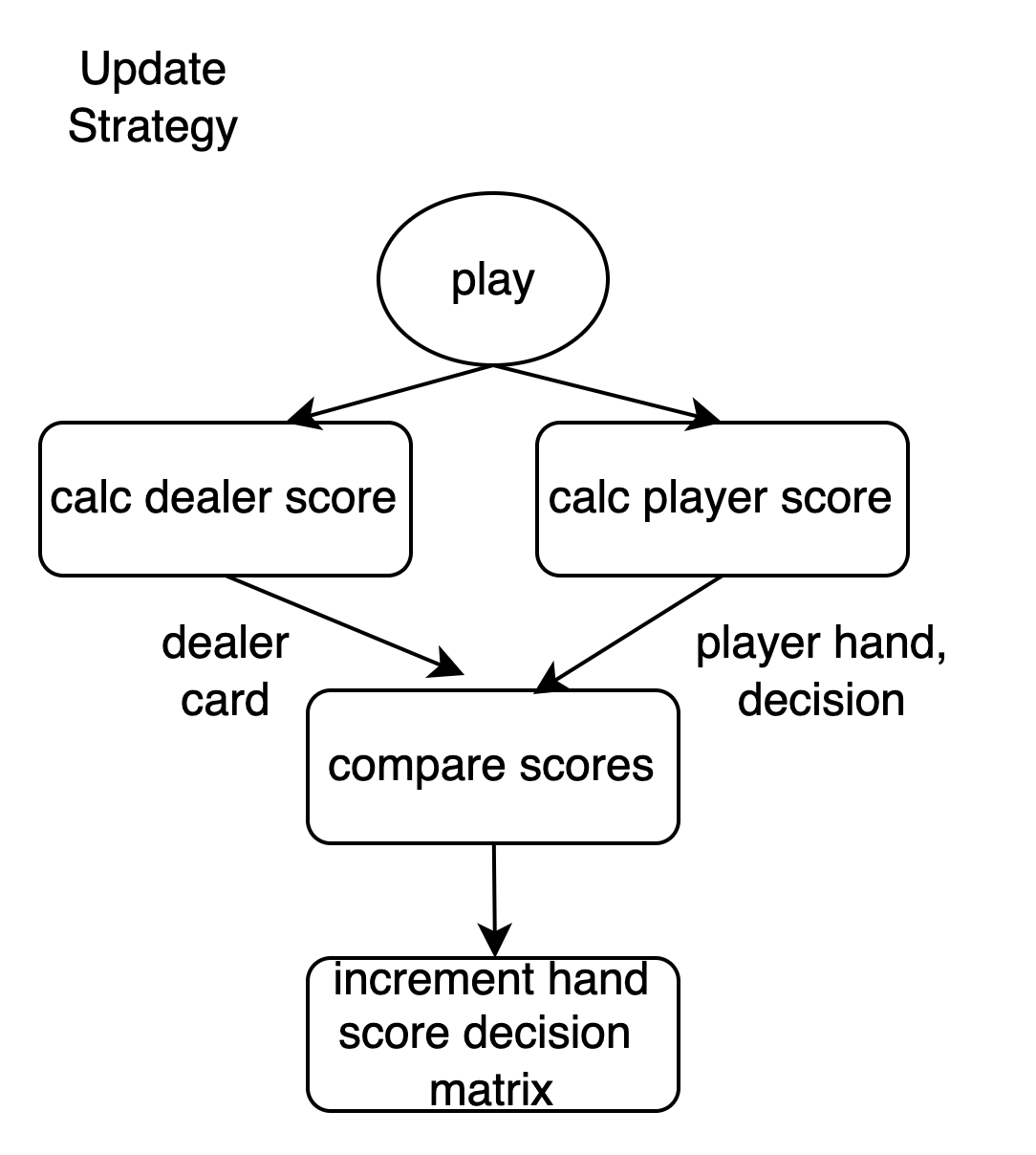
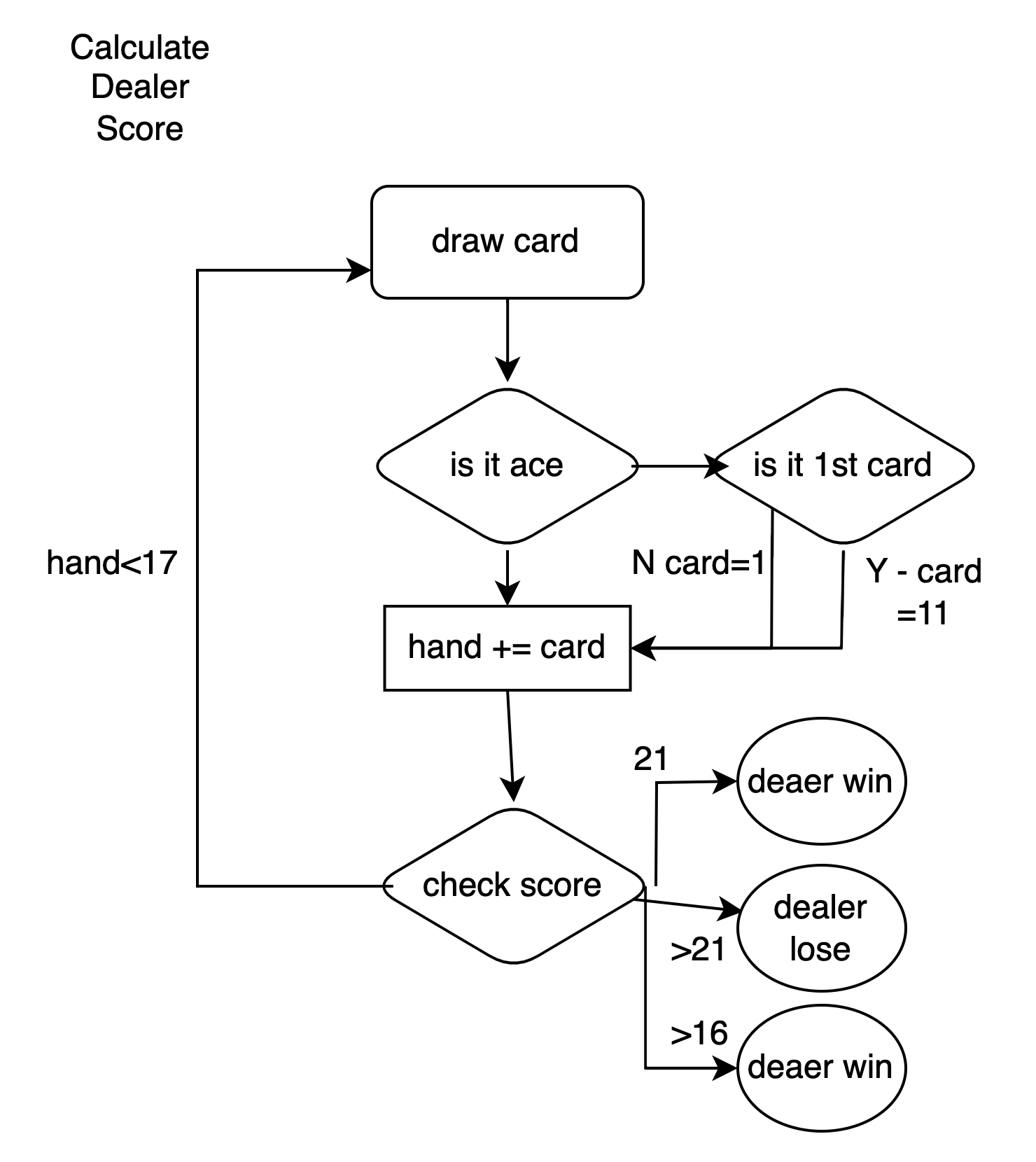
Case study - blackjack

Blackjack was chosen as the demonstration game as this involves an unknown variable which causes the player(computer) to make a decision. Games such as chess have the pieces and positions visible to both players so there is a finite solution or answer to the optimal move, blackjack involves chance and a decision based on the unknown variable of a dealers face down card.

Blackjack rules

Method

Formalise logic



Card score values to evaluate our decisions

Player win - pontoon 1.5, higher score 2 double down Pontoon 3 higher 4

Dealer win - Ponton -1.5 higher sore -2, double down -3, -4

Data structures

Hand score matrix is a map which stores a key of player hand vs value of map which has key dealer face up card and value handscore

Map(k=playerscore, v=Map(k=dealerhand, v=handscore)

Handscore stores state of decision total scores and number:

Handscore:

Numhits

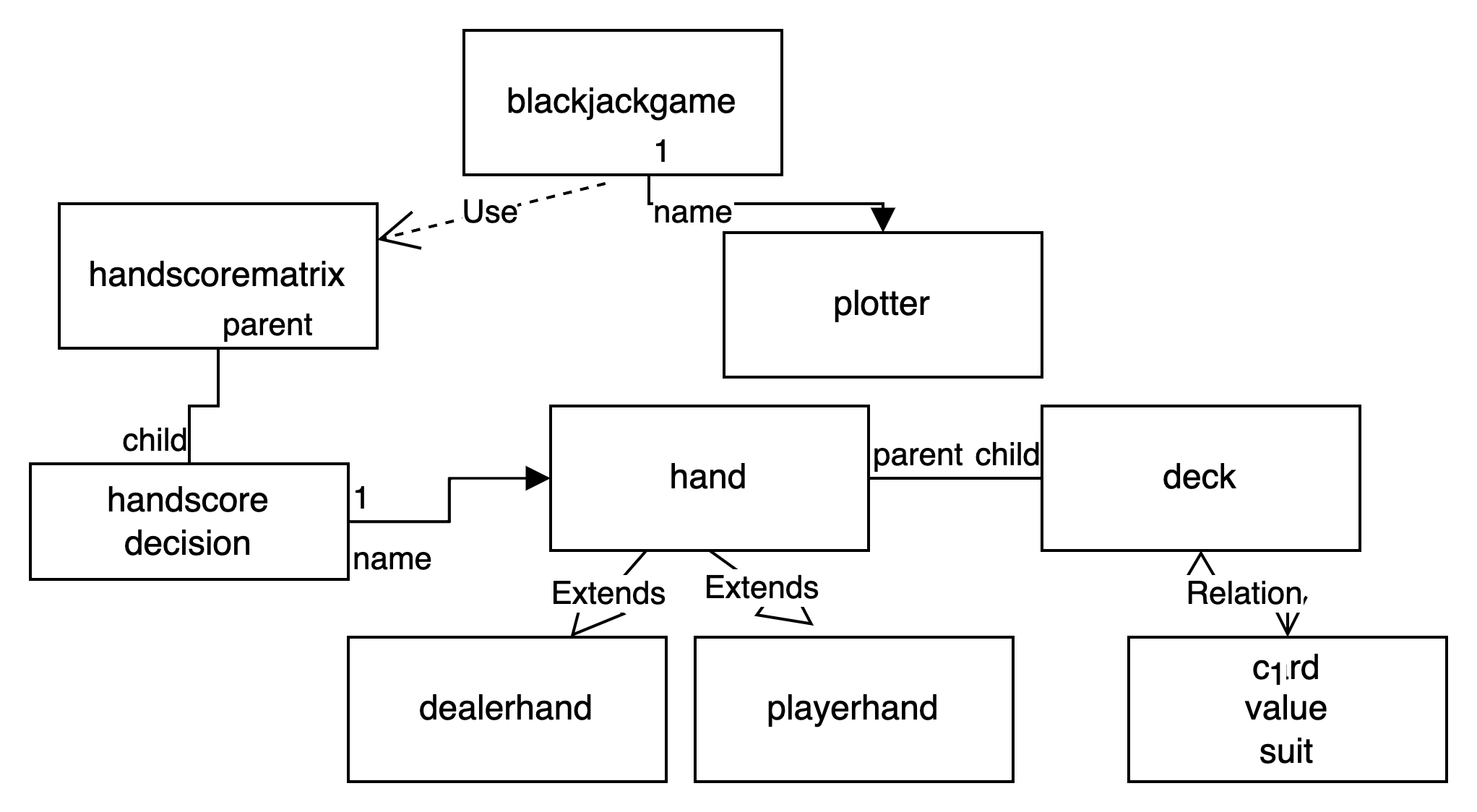
Numdd

Numstick

Hittotal

Ddtotal

Sticktotal

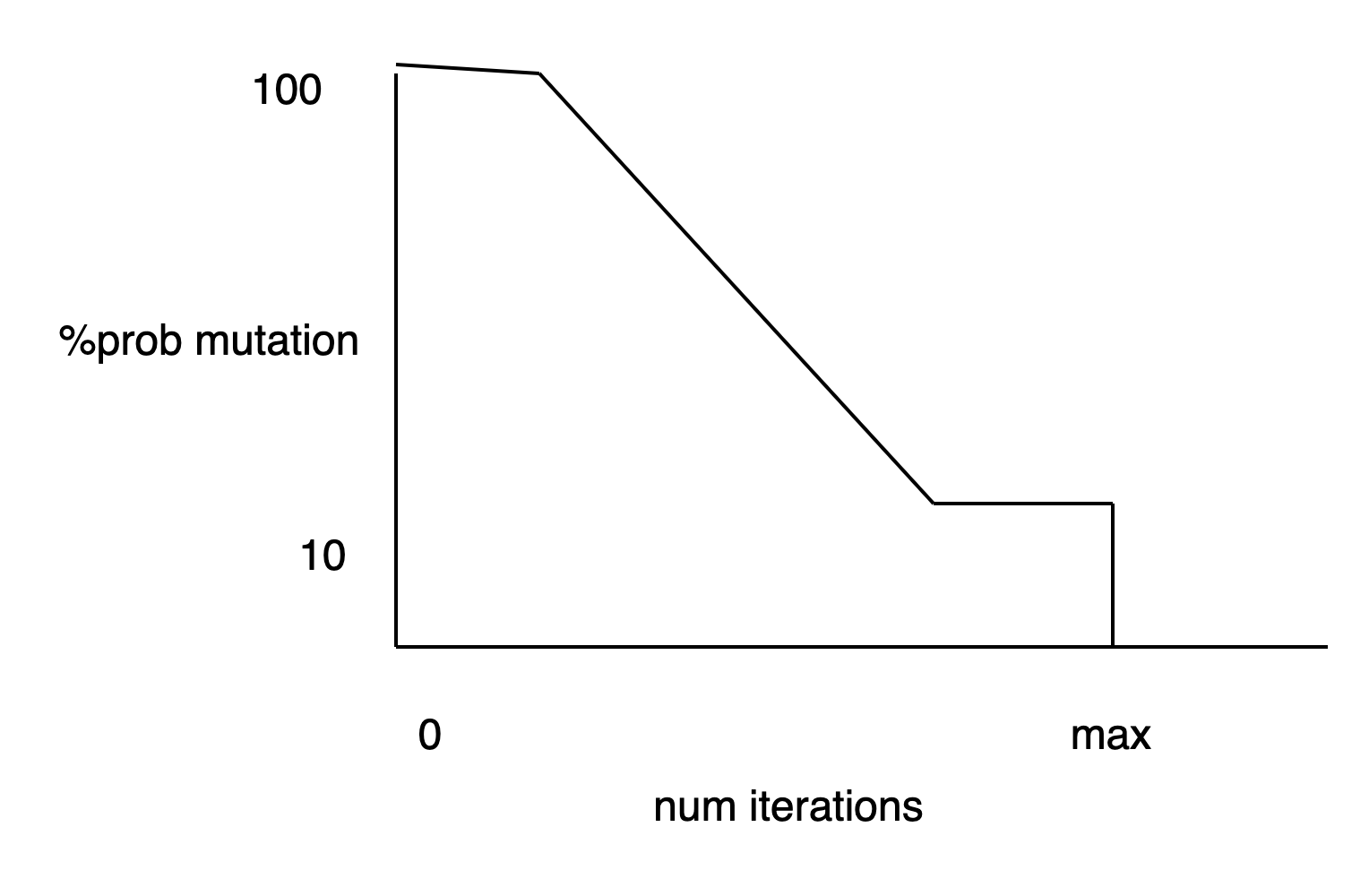
Classes

From that we can calculate the highest average result based on total score and num times made the decision

Genetic mutation

Genetic algorithms require some random mutation to evolve in the same way as Darwinian evolution. If we look up our decision based on previous we will lock in the 1st decision and always repeat it so we need some element of random mutation and competitive selection.

We do this using mutation

To start with the decision to h/s/dd is always random , a mutation, as we learn we retrieve our preferred calculated strategy from the matrix. We will always want some chance of improved decision with some chance of random mutation

Decision grid definition

Dealer faceup score

12345678910

Vs player hand total

23456789101112131415161718192021

Analysis

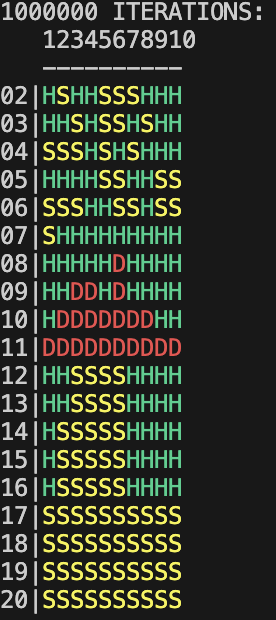
Iteration 1

Our model is young, the decisions are random

1950

We start to get some order very quickly

100000

Our learning is well defined and a clear pattern is shown, there is still some outliers shown provided for by our random mutation

Model evaluation

With real players - payout, photo , what happens when we put this in practice, would I try it in a casino

Method evaluation

Problems on the way - evaluating algorithms, the problem is large, created small test sub procedures, tested with known values

Limitations and assumptions

No splitting on pairs

Single deck pack with no card outback

Conclusion

What have I learned/proved

Appendix - full object code

References