9/20/2022

Determining the Best Dominion Cards Using Neural Networks

Ryan A Margraf

Table of Contents

Preface

This project was originally intended to be submitted to Nvidia as part of their Jetson AI Certification program, but it doesn't feel complete enough to do so at this time. I did complete the online course required as part of that program, and the certificate can be found at:

https://courses.nvidia.com/certificates/4f5df0b62fe74883acc179af69b2ad74/

Introduction

Dominion is an award-winning card game that inspired the deckbuilding game genre. Since its release in 2008, it has had several expansion packs, which add new elements to the gameplay. The necessary rules required to understand this paper are given below:

Cards

There are three main types of cards:

Treasure Cards

These cards can be played in the Buy phase to give the player **coins** to spend that turn. There is also a Potion card which is counted separately from coins. The basic Treasure Cards consist of Copper (worth 1 coin), Silver (worth 2 coins), and Gold (worth 3 coins).

Victory Cards

Most of these cards serve no function during the game, but the Victory Points (VP) they provide are counted at the end of the game for scoring purposes. The basic Victory cards consist of Estate (worth 1 VP), Duchy (worth 3 VP), and Province (worth 6 VP).

Action Cards

These cards can be played during the Action phase and have a wide range of effects, e.g. letting a player draw more cards, giving a player more coins to spend that turn, or causing a player to gain a card.

Starting Cards

Each player will usually start with 7 Coppers and 3 Estates in their deck. In some games involving cards from the *Dark Ages* expansion, the starting Estates are replaced with "shelters", which are cards that have weak effects and do not give VP. In some games involving cards from the *Nocturne* expansion, one or more of the starting Coppers may be replaced with an heirloom, which is one of 7 treasure cards with an additional effect.

Supply

Before each game, 10 types of non-basic cards are randomly selected. These "kingdom cards", along with the basic treasure cards and victory cards, constitute the **supply.**

Turn Flow

A turn consists of 3 main phases:

Action

The player may play Action cards from their hand. Players can play 1 action per turn unless a card effect states otherwise.

Buy

The player may play their treasures and buy a card from the supply, or an Event or Project[1]. Players can buy 1 thing per turn unless a card effect states otherwise. Cards that are bought go to the player's discard pile unless a card effect states otherwise.

Cleanup

The player discards all cards played that turn as well as all cards remaining in their hand, then draws 5 new cards. When a player runs out of cards to draw, their discard pile is shuffled and used as the new deck.

End of the Game

The game ends when either all of the Province cards are gone from the supply, all of the Colony cards are gone from the supply (if Colony cards are being used[2]), or any 3 other piles of cards are gone from the supply. Whoever has the most Victory Points at this time is the winner.

Goal

Dominion is a game that can have a lot of complex interactions between cards, so creating an AI that can play every aspect would be a large undertaking. Thus, this project focuses on one aspect of the game: determining the best card to buy during the buy phase. This is likely the most important aspect of Dominion, a deckbuilding game.

Procedure

Data

While looking for data, I came across a post by the user markus on the Dominion Strategy Forum[i]. They had a spreadsheet with metadata of over 100,000 2-player games played from November 2017 until the release of the *Menagerie* expansion in March 2020, as well as links to logs for each of the games. These games were played by highly rated players online and should thus in theory include strategies worth replicating. I wrote a python script to send HTTP get requests to each of the log numbers in the spreadsheet and save them as text files.

---

[1] Events can be purchased just like cards, but instead of being added to a player's deck, they create a one-time effect. Projects are similar to events, but have permanent effects and can only be bought once.

[2] Colony is a Victory Card worth 10 VP. It can be included in the supply when cards from the *Prosperity* expansion are part of the kingdom.

From there, I wrote another script to extract the relevant data from each log and create another file that consisted of a series of game states.

The result of this data manipulation is a list of states of the game during the winning player's turn, along with the card that player bought under those circumstances. Only the winning player's buys were logged, since the goal is to replicate winning strategies.

Features

Each game state is represented by a vector of length 1414, split into the following components:

Kingdom Card Quantities

For each of the 430 kingdom cards available, 3 values were included in the game state representation:

1. The number of that card in the supply
2. The number of that card in the player's deck
3. The number of that card in the opponent's deck

The term "deck" is used loosely here, and includes all of the cards controlled by each player, such as in their hand, in their discard pile, or set aside by cards like Island.[3] This makes keeping track of the game state more feasible, and it was judged to be unlikely those minor distinctions would make a meaningful difference in the card that one should buy anyway.

There is also a hidden fourth category here: The number of each card in the Trash. The trash is distinct from the players' discard piles. Cards in the Trash are not part of any player's deck and are unavailable for the remainder of the game. This was not used as an input because if the other three categories are known, this can be determined since there is nowhere else for the card to be (Actually, cards not in the supply would have values of 0 for the other three and would appear to be all in the trash, but the effect is the same: The card is unavailable to be used in the game).

Other Card Quantities

For each of the 5 prize cards, 2 values were included in the game state representation:

1. The number of that card in the player's deck
2. The number of that card in the opponent's deck

There is only one of each prize card, so each of these will be a binary digit. Additionally, prize cards are only available when "Tournament" is in the supply, so there was no need for another value indicating whether they are used.

Projects and Events

Events and Projects both serve as alternatives to purchasing a new card to add to your deck. Events provide one-time effects and can be bought as often as desired. Thus, for each event, only one input was needed to indicate whether or not it is available to buy during that game.

[3] http://wiki.dominionstrategy.com/index.php/Island

Projects provide more permanent effects and can only be bought once per game. Thus, each project contributes 3 inputs to the state:

1. Whether the project is available to buy during that game
2. Whether the player has bought the project during that game yet
3. Whether the opponent has bought the project during that game yet

Landmarks

Landmarks affect the game, but they are not something that can be bought. Thus, for each of the 21 landmarks, only one input is required: whether it is present or not.

State Data Not Included as Inputs

There is another aspect of dominion called, fittingly enough, "states", but they are not commonly used and I judged that their effects would be too minor to be worth the additional effort that would be required to put into writing my game log analyzer. Artifacts were ignored for similar reasons.

Cards called Boons and Hexes provide one-time effects but do not actually stay in the possession of a player for any length of time, so it was unclear how to incorporate them into the model.

It was also determined that the number of coins the player has available would not be an input. This is for a few reasons:

1. The game logs did not readily provide this information, and given the complex nature of card interactions in Dominion, determining this information would be a large undertaking in and of itself, likely requiring writing a simulation for a significant amount of the game. This was determined to be outside the scope of this project.
2. Dominion is a deckbuilding game, where overall deck composition is an important consideration. In other words, the strength of the deck is not affected by the number of coins available to spend on any given turn.
3. By having an output listing cards in order of priority, we can provide the user with a list of options. For example, if the card with the highest priority costs 3 coins and 2nd card has a priority only slightly lower, but costs 6 coins, then the player could choose the 2nd card if they have 6 coins available.

Outputs

The outputs consist of 430 possible kingdom cards, plus 53 for events and projects, for a total of 483 possible options to buy.

Results and Reflection

For some reason, regardless of the number of inputs used, the model settled to an accuracy of 0.12 after a few epochs. Further analysis of the outputs revealed the cause:

```
[ 37 482 151]
Province 0.00560796307399869
Citadel 0.002063054358586669
King's Court 0.002063054358586669
[ 37 482 151]
Province 0.00560796307399869
Citadel 0.002063054358586669
King's Court 0.002063054358586669
[ 37 482 151]
Province 0.00560796307399869
Citadel 0.002063054358586669
King's Court 0.002063054358586669
[ 37 482 151]
Province 0.00560796307399869
Citadel 0.002063054358586669
King's Court 0.002063054358586669
[ 37 482 151]
Province 0.00560796307399869
Citadel 0.002063054358586669
King's Court 0.002063054358586669
```

It was predicting a Province every time. Provinces are almost inarguably the best card in the game, since they're usually the card worth the most Victory Points. It is conceivable that they would make up an average of 12% of the winning player's buys during a game. If you buy a Province every time you can afford one, that is not a bad strategy. The hard part is getting enough coins in one turn to do so, which is what other cards allow you to do.

The #2 and #3 top ranking cards are also quite powerful. Citadel is the most expensive project, costing 8 coins, that allows you to play an action card twice every turn for the rest of the game. King's court is a regular kingdom card that lets you play another action card in your hand 3 times. Both can enable "mega turns", where one can chain several action cards and acquire several Victory cards such as Provinces at once.

In the end, though, these cards are all expensive outliers. For the more common, cheaper cards, there is usually no definitive answer in what is the best to buy, so there is just not enough consistency for the model to pick up on to improve its accuracy. Additionally, the similarity in states between consecutive buys could be confusing the model, since a lot of times it wouldn't matter too much which order 2 consecutive buys are performed. Perhaps including cost as an input would have produced better results, but as explained before, it just wouldn't be feasible in the time I was given.

Furthermore, besides the basic treasure and victory cards, each game consists of a different supply, meaning different cards are available each time. This further worsens the problem. If there was a

specific set of cards that was played often, that would greatly reduce the number of inputs and provide more consistency.

Issues Encountered

No Reference for Logfiles/Inconsistent Logs

The keywords in the logs were numerous, and it was not always clear what they did. There were also some inconsistencies. For example, some of the logs used INSERT_IN_DECK to indicate a player's stating cards, while others used STARTS_WITH, and INSERT_IN_DECK meant something else.

When it got to the point that I was running out of time for other things, I ended up just checking for underflow in the 8 bit unsigned integers, which would indicate something unexpected had occurred, and tossed the game logs where that happened. It is almost certain that some other bad ones snuck through, but I didn't have another reasonable option under the time constraint.

Neural Network Model

Getting the neural network model to accept the inputs was another large task. For some reason, it didn't like the format of the Sequential model even after adjusting the input and output sizes. I also ended up having to add a check to make sure each input was the correct size, which showed something unexpected happened to some state log files.

---

[i] http://forum.dominionstrategy.com/index.php?topic=18988