Voting Model Optimization: A Mathematical Approach

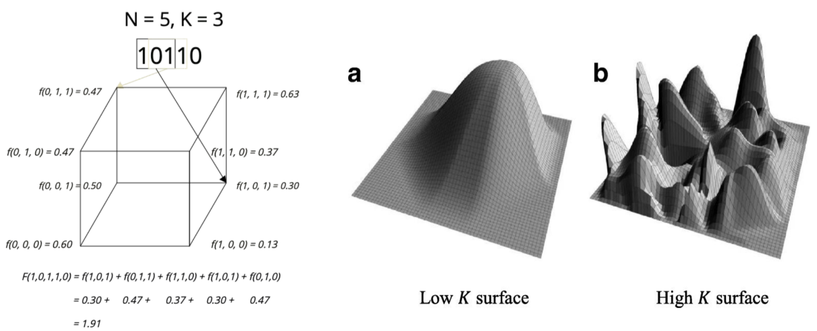
Joshua Nunley [1] Sunny Gandhi [1]

[1]Indiana University

Introduction

Our goal was to mathematically explore how we could represent different voters and voting models through an "NK Landscape." We defined optimizing the best voting model choice for our voters as "maximizing" the fitness for the voters by searching the landscape.

Figure 1. Examples of NK Landscapes



Our experiments involved manipulating different aspects of the model — the "N" and "K" values specifically — in order to manipulate characteristics of the voter themselves.

Voting Models Explored

We implemented a plethora of voting models. We ranged from very easily implemented to unrealistic voting types, in order to establish effective comparatives.

- **Plurality** where each voter votes for one candidate, and the one with the most votes win.
- **Approval** where each voter can "approve" all proposals that are better than their current one, if one exists. Otherwise they only approve of their current proposal.
- **Normalized Score** where voters can only vote for a fixed number of candidates.
- **Total Score** where voters "score" candidates and once totals are summed, the candidate with the most "points" wins.
- **Marginalized Score** where voters are aware of exactly what their fitnesses are.

NK Landscape Architecture

An NK landscape is a model which assigns fitnesses to binary strings.

- **The N** represents the length of the binary strings.
- **The K** represents the length of the dependencies between bits in the strings.

In Figure 1, on the left, there is a graphic representation of assigning a fitness to a binary string in an NK model with n = 5 and k = 3. The sum of the fitnesses of each bit determine the fitness of a binary string in this model. The fitness of each bit is determined by its dependencies.

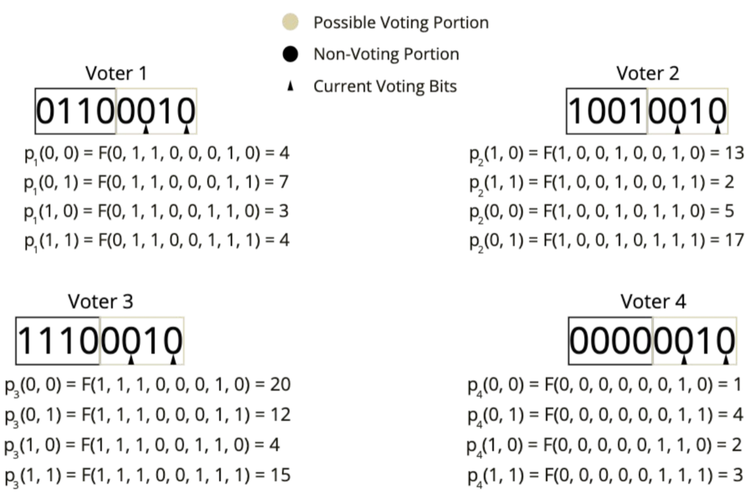
The figure on the right is simply a graphical representation of what different K values look like. For low K values, the landscape is simple, for high K values the landscape is topographically rough, with an abundance of local peaks.

Representing Voters

We represented voters as binary strings in the NK model. Each binary string represents a set of constraints on the voter that would determine their maximum possible utility. This value is provided by the NK model. In the figure below, the black rectangle around the binary string is a representation of constraints that do not affect voting, while the tan rectangle represents those constraints that do affect voting.

Voting on an NK Landscape

In each binary string below, the substring in the black rectangle represents the dependencies of the first bit and the substring in the tan rectangle represents the dependencies of the second bit. This is oversimplified in the example here — in reality, the program randomly disperses voting and non voting with each other. In this particular scheme, each bit is dependent on itself and the following two bits. In order to calculate the fitness of a string we must determine the fitness of each substring governed by the dependencies. After fitnesses for all substrings of length K are drawn from some distribution, we can calculate the fitness of the string by summing up the fitnesses of the substrings. An example of this calculation is shown at the bottom of the figure.

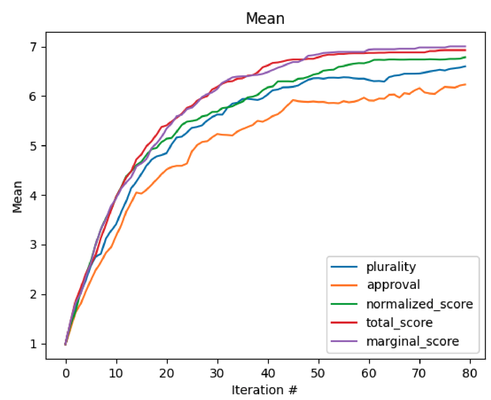


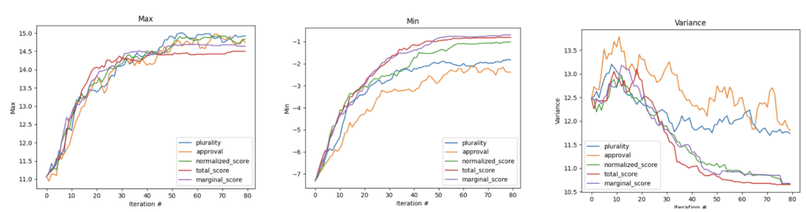
The non-voting portion is different for each agent, while the voting portion is the same, though due to the dependencies there is an interaction between the two. The non voting portion is not modified during the process of voting. Just like an optimization algorithm, the voting process happens in iterations. In each iteration, a subset of the voting portion of the binary string is chosen to be voted on (represented by two black triangles). Below each voter, there is an example of the calculation of the fitness of each proposal with respect to the current bits to be voted on. To calculate the fitness of each proposal, we substitute it into the bits to be voted on in each voter string and calculate the fitness of this new string.

These aggregated fitness values are then converted for each voting method as a "ballot". For example, in plurality voting, a voter one would vote for proposal 01, voter 2 would vote for proposal 11, voter 3 would vote for 00, and voter 4 would vote for 01. In this case, plurality voting chose to accept proposal 01. This means that every voter now replaces the voting bits with the winning proposal, a new set of voting bits is chosen randomly, and the process is repeated. Generally this process results in an increase in fitness for the population.

Preliminary Results

These are taken from an average over 50 runs. As a reminder, marginal and total score are impossible in real life, as they require voters to know the exact fitness of each proposal. However, they act as a strong comparative to other voting methods.





Interpreting these plots, approval voting is worse than plurality and normalized score in the population mean (also supported by the population minimum). Normalized score significantly reduces variance. Plurality and approval voting both have higher fitnesses on the population maximum.

We plan on implementing more voting methods to understand how different circumstances affect the fitness model.

References

[1] Felipe A Csaszar.
A note on how nk landscapes work.
Journal of Organization Design, 7(1):15, 2018.

[2] Stuart Kauffman and Simon Levin.
Towards a general theory of adaptive walks on rugged landscapes.
Journal of Theoretical Biology, 128(1):11–45, 1987.

[3] Eric Pacuit.
Voting methods.
Standford Encyclopedia of Philosophy, 2019.

[4] Marcus Pivato.
Condorcet meets bentham.
Journal of Mathematical Economics, 59(C):58–65, 2015.

sugandhi@iu.edu