Combining Classical AI with Modern Computing:
The BACON System for Equation Discovery from Scientific Data

Jonah Miller, Soumya Banerjee
University of Cambridge


Created using ChatGPT

Mithraeum Museum

https://livinglondonhistory.com/london-mithraeum-a-hidden-roman-temple-under-the-city/

About me

What do I do?

Introduction

- Recently fascinated with history of old computational techniques
- Study the past to inform the future
- “Archaeology” of algorithms/historical reconstruction of algorithms
- *Unconventional approaches to AI*

Motivation

- We should try to learn from the past
- AI has had multiple winters
- Another AI winter?
- Deep learning may (or may not) saturate at some point
- Attempt a synthesis of classical approaches with modern ones

About the project

Introduction

- Jonah Miller: Part 3 Student in Cambridge
- Reconstructed a heuristic-based/classical AI algorithm multiple versions from scratch
- Implemented in Python

Motivation

- Part of his thesis
- Taught himself computer science in the process (Math major)
- Interested in seeing if computers can solve problems in mathematics/physics

Introduction to the Problem

Introduction and Motivation

Introduction

- Computational Scientific Discovery (CSD) finds patterns and invariants in datasets using computers.
- Classical AI methods/heuristic based methods – which pioneered the space in the 1980s' – have been largely forgotten.
- Recent research has been done using deep neural networks (DNNs).

Motivation

- The programs used in the 1980s' had to be optimised and efficient to run on the computers of the time.
- Classical AI programs are explainable and reproducible, not the case with modern day DNNs.
- Classical programs are significantly faster as they have no training time.

Background

Classical AI and Explainable AI

Classical AI

- Deterministic and reproducible.
- Output is interpretable and clear based on the programmed logics.

Explainable AI

- Modern DNNs have outputs that may not be understandable
- Can lead to serious consequences such as discriminatory models or medical misdiagnoses.
- Explainable models are trustworthy and worth the goodness-of-fit sacrifice.

Background

The BACON System

- BACON is the Classical AI program used.
- It is explainable and understandable (and elegant!)

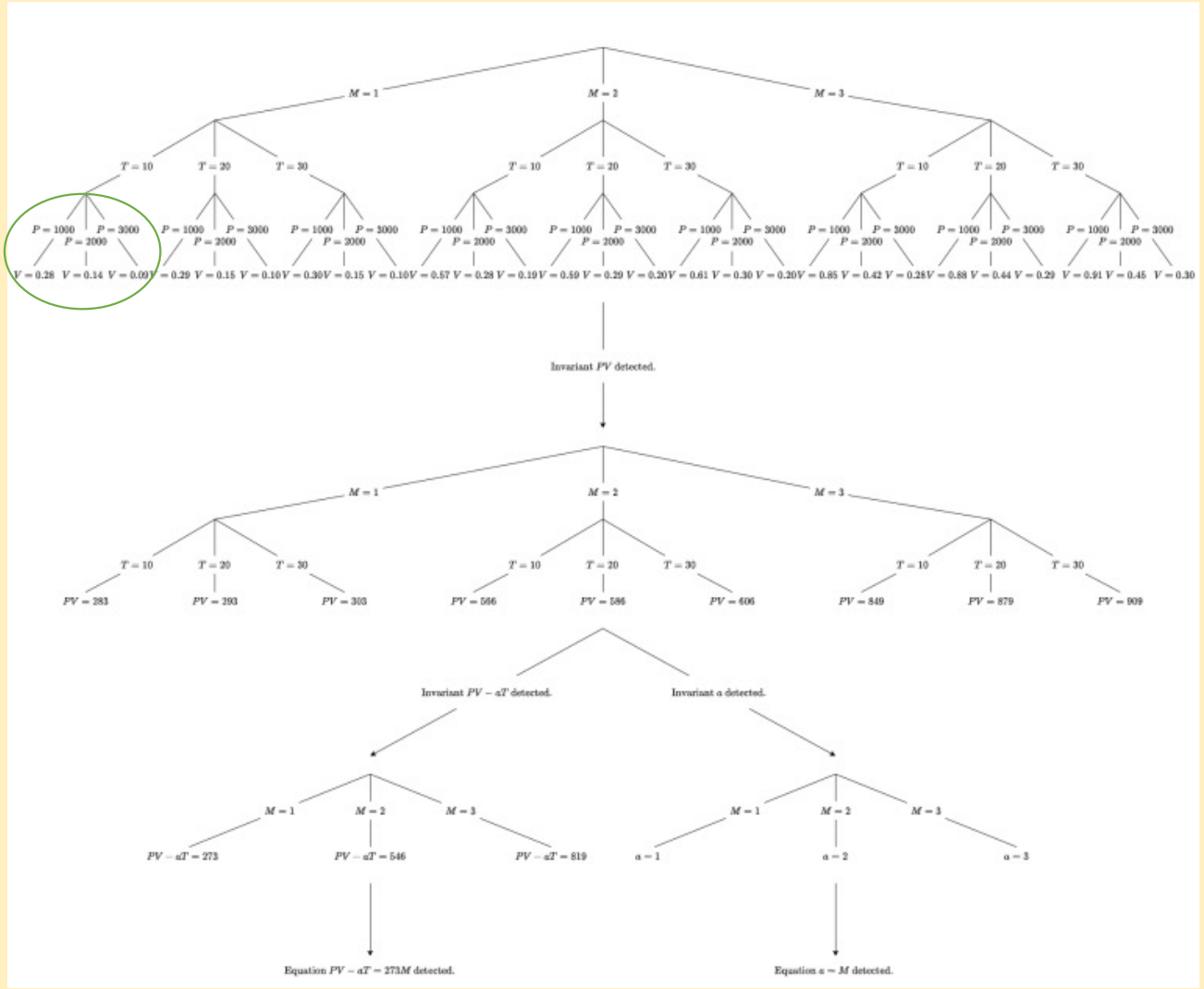
- Uses simple heuristic to find invariants between two variables.
- For two variables X and Y if they increase/decrease together then divide them. Otherwise consider their product.

Planet	Distance (D)	Period (P)	$\frac{D}{P}$	$\frac{D^2}{P}$	$\frac{D^2}{P^2}$	$\frac{D^3}{P^2}$
A	1.0	1.0	1.0	1.0	1.0	1.0
B	4.0	8.0	0.5	2.0	0.25	1.0
C	9.0	27.0	0.333	3.0	0.111	1.0

The BACON System - Multivariable

$$V = \frac{M(T + 273)}{P}$$

	M	T	P	V
0	1	10	1000	0.283000
1	1	10	2000	0.141500
2	1	10	3000	0.094333
3	1	20	1000	0.293000
4	1	20	2000	0.146500
5	1	20	3000	0.097667
6	1	30	1000	0.303000
7	1	30	2000	0.151500
8	1	30	3000	0.101000
9	2	10	1000	0.566000
10	2	10	2000	0.283000
11	2	10	3000	0.188667
12	2	20	1000	0.586000
13	2	20	2000	0.293000
14	2	20	3000	0.195333
15	2	30	1000	0.606000
16	2	30	2000	0.303000
17	2	30	3000	0.202000
18	3	10	1000	0.849000
19	3	10	2000	0.424500
20	3	10	3000	0.283000
21	3	20	1000	0.879000
22	3	20	2000	0.439500
23	3	20	3000	0.293000
24	3	30	1000	0.909000
25	3	30	2000	0.454500
26	3	30	3000	0.303000



Relevant Works

PySR

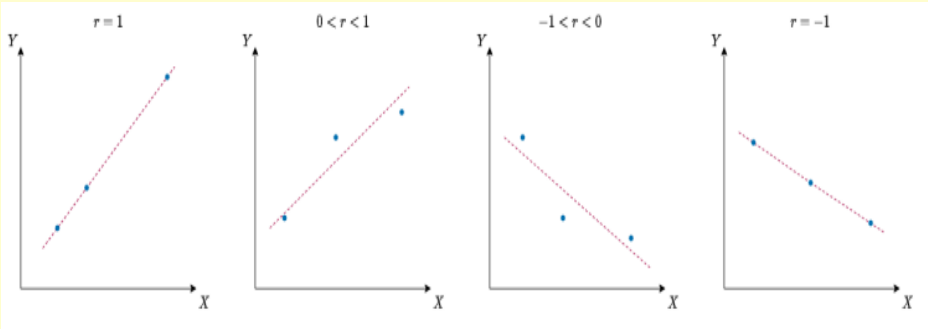
- Modern day method, uses symbolic regression on a graphical neural network.
- More powerful than BACON, can solve differential equations and find invariants in discontinuous environments.
- Biases coefficients towards simplistic values.

Method

BACON.1

Linearity check:

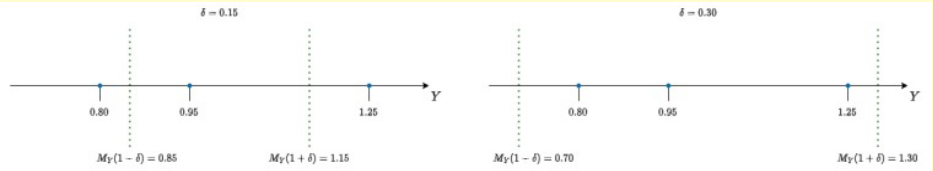
- Uses correlation coefficient *r* between variables *X* and *Y*.



Invariant check:

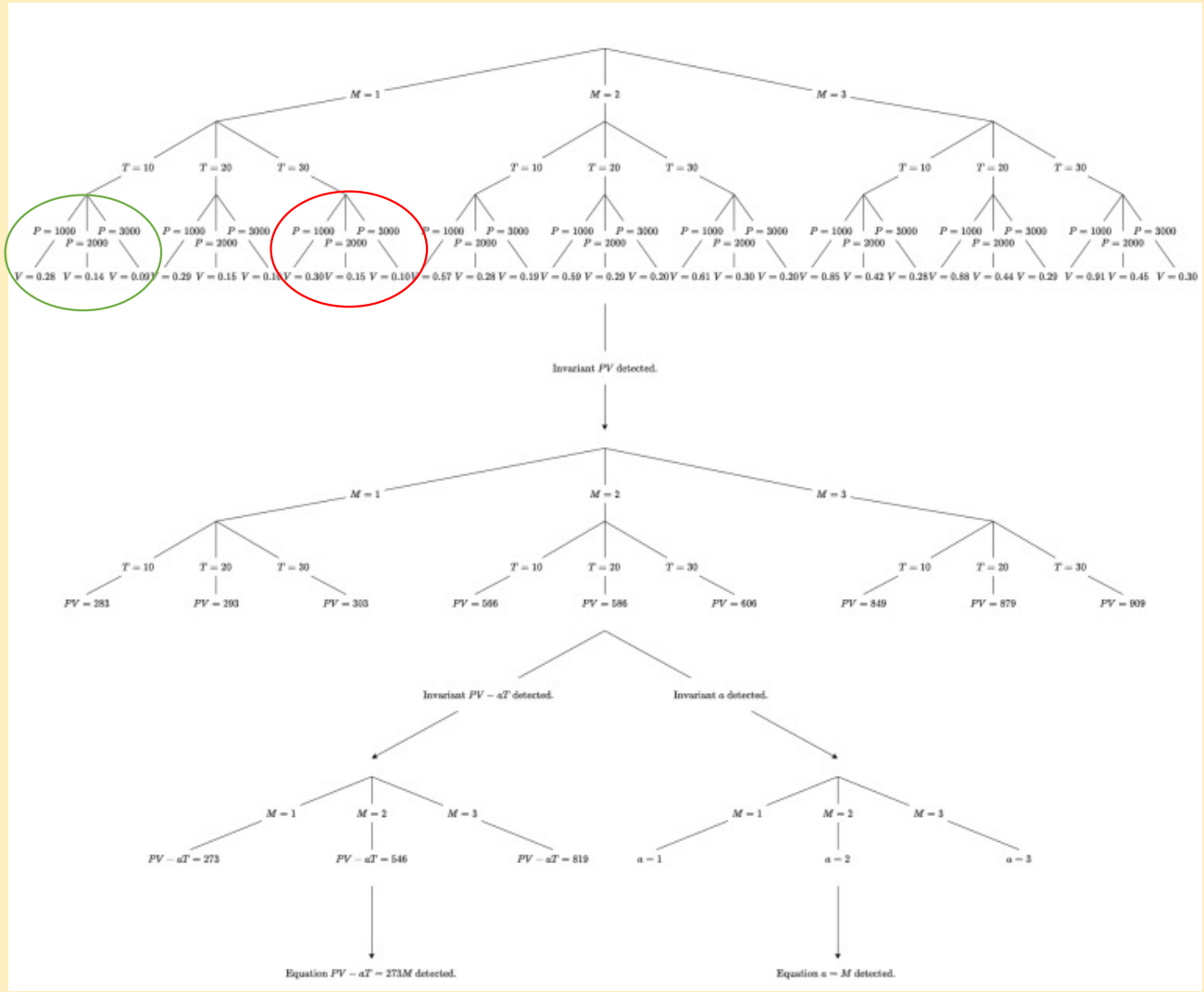
- Checks if each value of the dataset satisfies

$$M_Y(1-\delta) < \text{value} < M_Y(1+\delta)$$



Noise

$$V = \frac{M(T + 273)}{P}$$



Results

Datasets

- Our full reconstruction of BACON (we call it BACON.7) was tested on 3 datasets.
- Each dataset had up to 4% noise added to the dependent variable.
- BACON.7 and PySR formed predictions on the datasets equation after being handed this noisy data.
- The aim was to find the correct form of the equation.

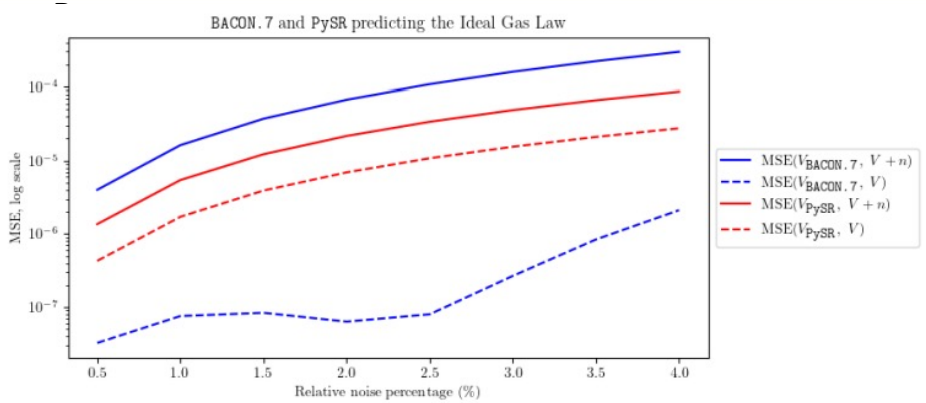
The Ideal Gas Law: $V = \frac{M(T + 273)}{P}$

Ohm's Law: $I = \frac{TD^2}{2(L + 3)}$

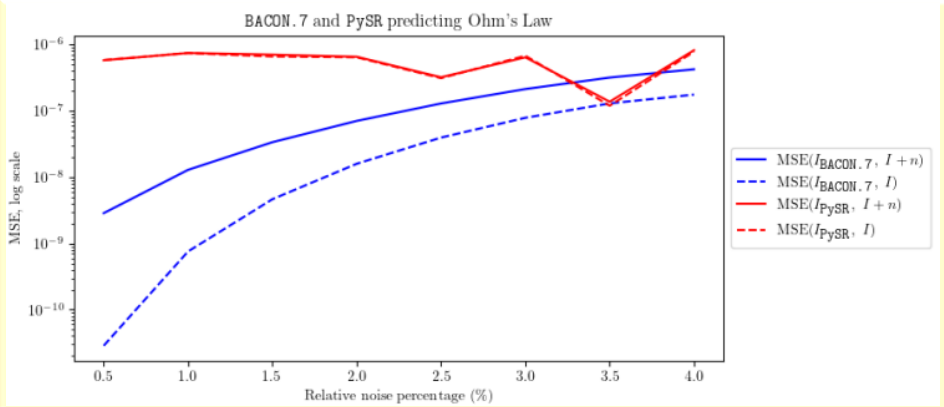
Black's Law: $T_f = \frac{M_1 T_1}{M_1 + M_2} + \frac{M_2 T_2}{M_1 + M_2}$

Results

The Ideal Gas Law and Ohm's Law (27 datapoints)



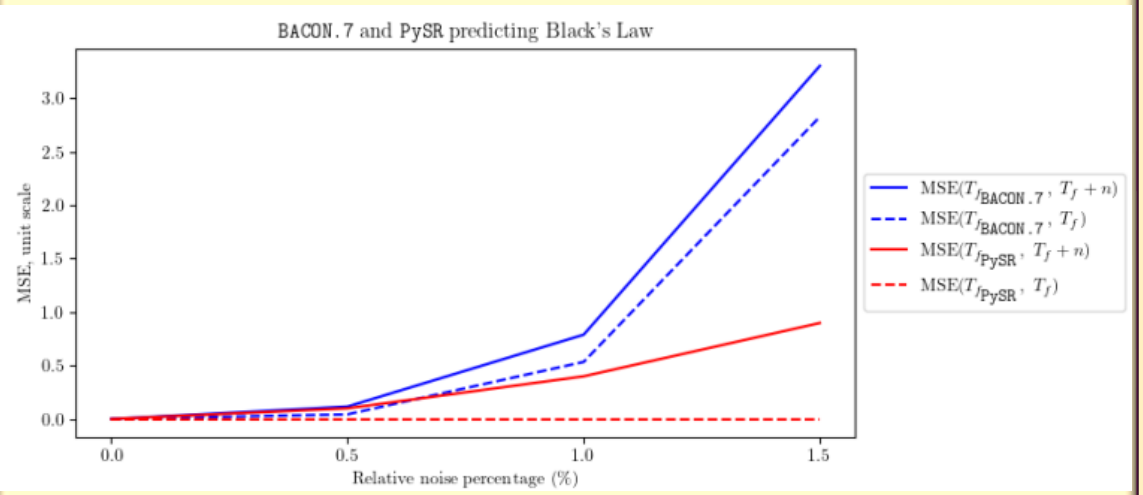
- BACON finds the true equation an order of magnitude more accurately than PySR (if there is no noise).



- PySR can't deduce the correct form of the equation.
- BACON again finds the true equation more accurately than PySR's best prediction.

Results

Black's Law (81 datapoints)



- PySR finds the perfect equation on Black's Law regardless of noise.
- BACON cannot handle 2% noise.

Ramanujan's approximation

Ramanujan birthday problem

We also applied our BACON.6 reconstruction to the Birthday Problem. This is demonstrated through the Birthday Problem: in a year of $n \geq 1$ days, what is the minimal number of people in a room such that the probability of at least two sharing a birthday is over 50%. This is extrapolated to, assuming each person enters the room sequentially, what is the expected number of people to enter, for a birthday to first be shared. Ramanujan discovered the answer is $\lfloor Q(n) + 1 \rfloor$ [1], where

$$Q(n) = \sum_{k=1}^{n} \frac{n!}{(n-k)!\, n^k} \tag{2}$$

$$= \sqrt{\frac{\pi n}{2}} - \frac{1}{3} + \frac{1}{12}\sqrt{\frac{\pi}{2n}} - \frac{4}{135n} + \mathcal{O}\left(\frac{1}{n^{\frac{3}{2}}}\right) \tag{3}$$

$$\approx 1.253\sqrt{n} - 0.333 + \frac{0.104}{\sqrt{n}} - \frac{0.030}{n} \tag{4}$$

When given an appropriate answer form without coefficients to BACON.6[2] with datapoints for $n \in \{2, 3, 4, 5, 6, 7, 8, 9\}$ and associated $Q(n)$, it discovers that:

$$Q_{\texttt{BACON.6}} = 1.252\sqrt{n} - 0.326 + \frac{0.086}{\sqrt{n}} - \frac{0.012}{n} \tag{6}$$

The MSE between $Q_{\texttt{BACON.6}}$ and the real $Q(n)$ is 5.4×10^{-9}. PySR comparatively struggles. Given the same data, and the *binary_operators* from Figure 15 expanded to include powers, the best result it finds is:

$$Q_{\texttt{PySR}} = n^{0.557} + \frac{0.324^{\frac{5.28}{n}}}{n} \tag{7}$$

This has MSE 3.8×10^{-5} to the true form – 4 orders of magnitude worse than BACON.6. This further lends credibility to the idea seen throughout that PySR does not function well on few datapoints. This also demonstrates the strength of BACON's heuristic mechanisms to be incredibly accurate in an environment where the state-of-the-art performs poorly.

[2] The form is:

$$nQ(n) = \alpha n^{\frac{3}{2}} + \beta n + \gamma n^{\frac{1}{2}}$$

Conclusion

Limitations and Future Directions

- 1980s' papers.
- Written full of dense code in antiquated languages, papers meant for people already knowledgeable with the sector.
- No open-source code in modern languages.
- Hard to compare to other projects in the area due to the lack of modern implementations.

- For some smaller datasets, BACON can thrive whilst PySR struggles
- Combine modern DNNs with Classical approaches to get powerful models which are more explainable.
- For example, can add dimensional analysis in BACON to choose the correct equation when datasets are noisy.

Summary

1. This project aimed to prove that Classical and Explainable approaches can improve Computational Scientific Discovery.

2. It did this by recreating the understandable BACON. Results displayed BACON was sometimes more accurate than modern PySR on *smaller datasets*.

3. Future directions involve combining these approaches with modern methods. Code released open-source to aid development in the area.

Next Steps

1. There may be scenarios where classical AI/heuristic-based methods can perform better than modern techniques
2. A fruitful next step would be to create *hybrid systems* that combine classical AI techniques with deep learning (e.g., large-language models)
3. Active learning: actively probe and acquire data

Learnings

1. Study classical AI systems
2. A lot can be learnt by re-implementing them in modern programming languages
3. A lot can be learnt by studying these seemingly anachronistic systems

Created using ChatGPT

Code and Preprint

https://github.com/JonahMiller/BACON

https://osf.io/preprints/osf/z8kqv

neel.soumya@gmail.com

Noise

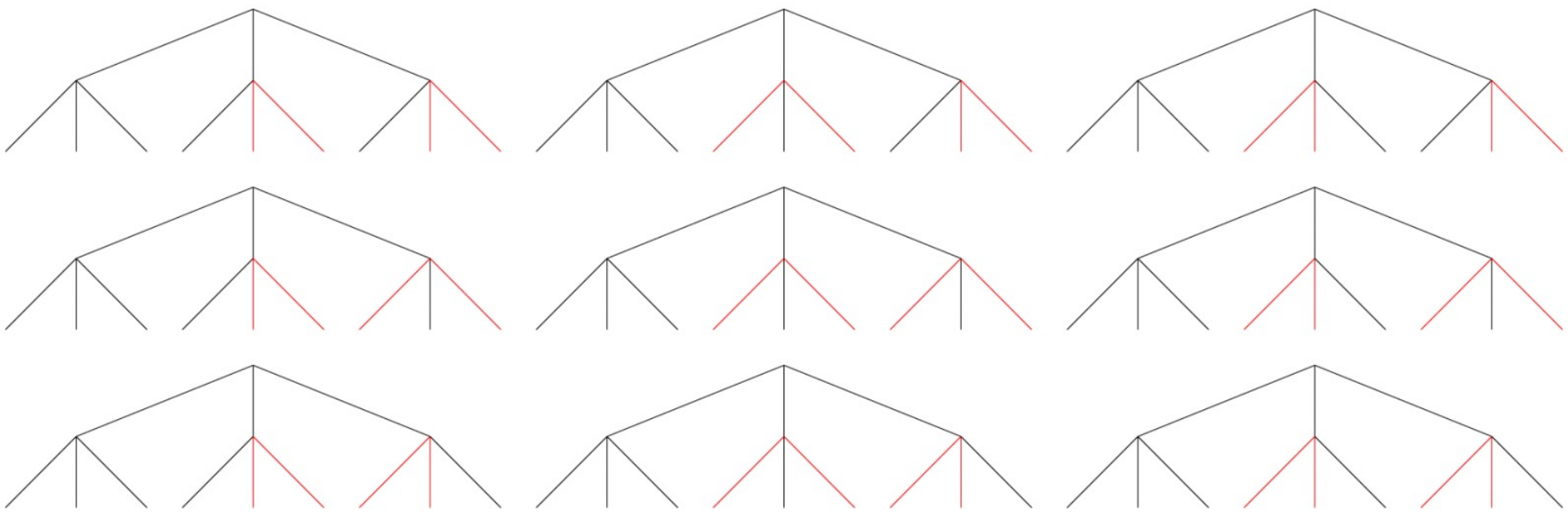


Figure 7: Possible ways to pick datapoints in `BACON.5` when the set needed is in the bottom left. The chosen datapoints are in black. As the initial set could also be from the centre or right branch, there's a total of 27 possible ways to initially choose.

Noise

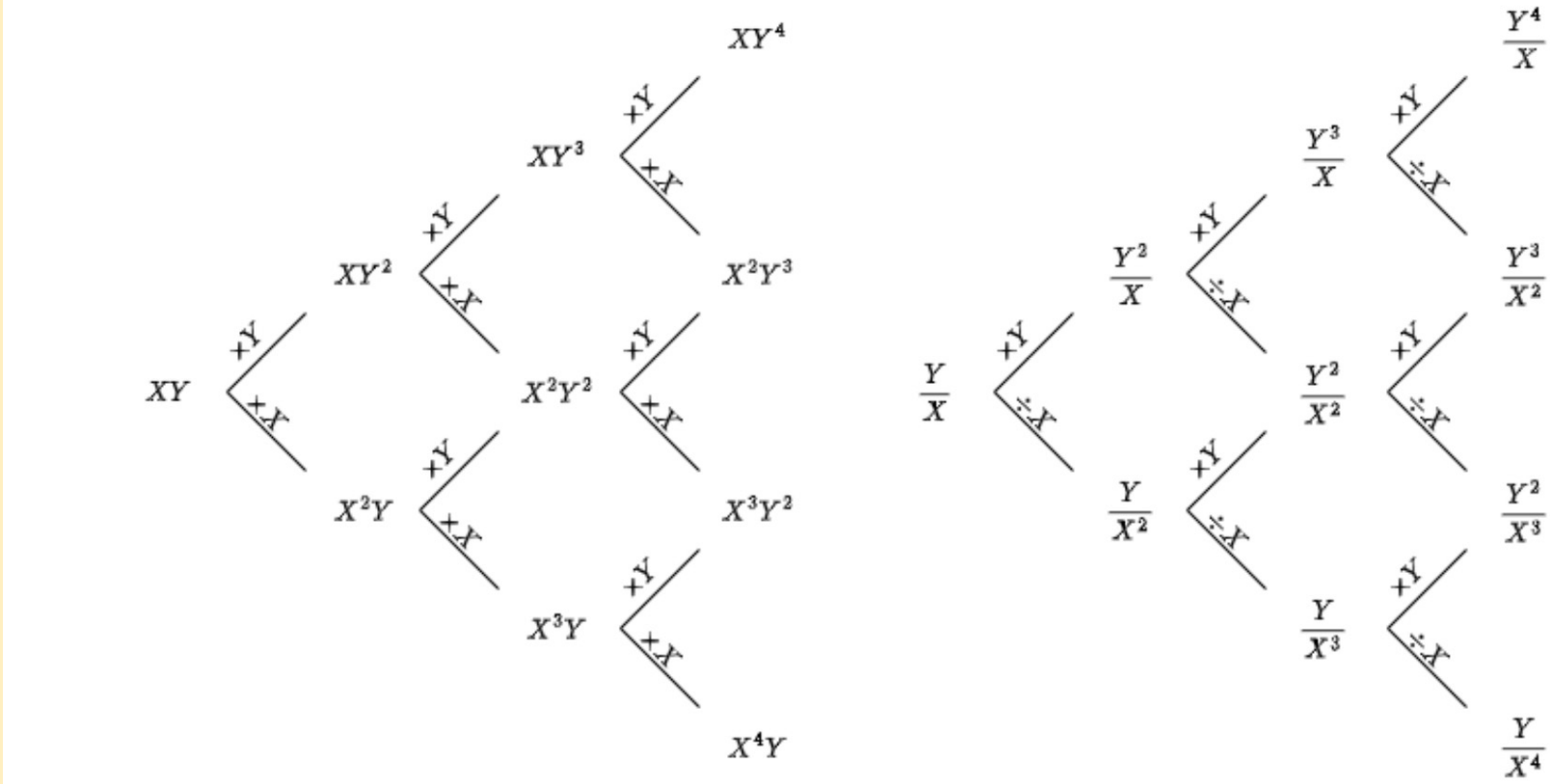


Figure 6: All 10 possible relations for a `BACON.1` run capped at $j = 4$. Note this is the minimum required to find Kepler's Law. Every equation is of the form $X^n Y^m$ for $|n| + |m| \leq j + 1$.

Monte Carlo Tree Search

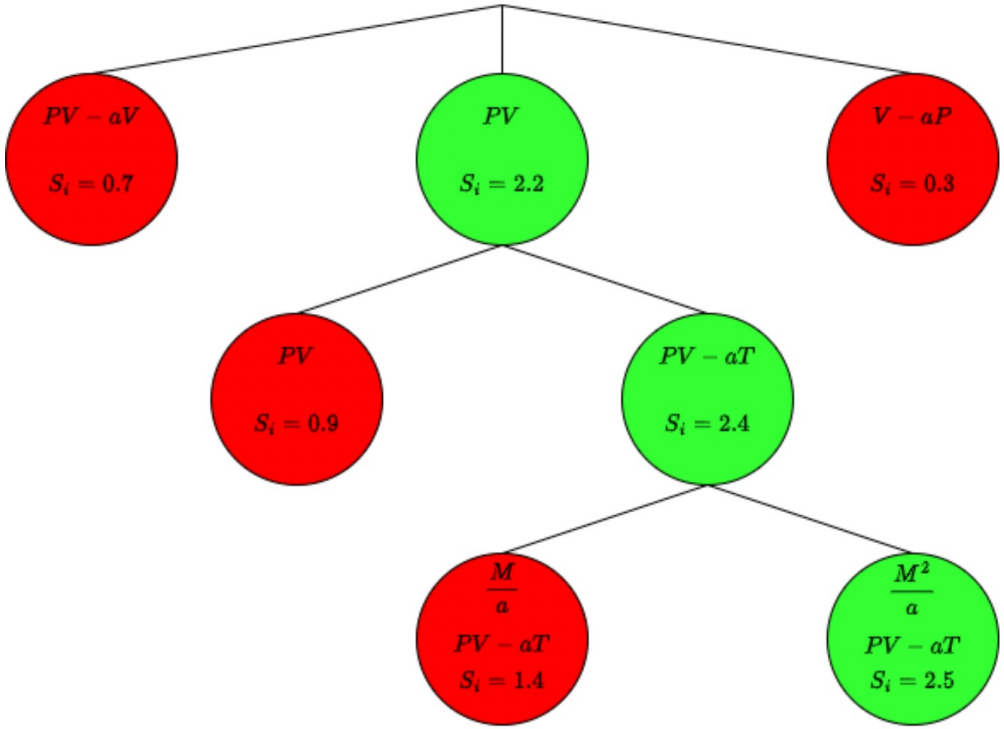


Figure 13: The graphical representation of the MCTS on the low C for the Ideal Gas Law in an environment with 5% noise. The scores at each stage are shown with S_i, with the child node selected in green. Note that the green node's S_i increases through the layers as MCTS prunes the worst nodes. For the first two layers, the action is picking an equation from those discovered by the sets at that layer. The bottom layer is determined by applying a combination of the following hyperparameters $\epsilon, \delta \in \{(0.01, 0.1), (0.05, 0.5), (0.01, 0), (0, 0.1)\}$ and gaining their respective equations.

To use MCTS we need the concept of a node. In chess, this is represented as the position of the pieces on a board at a given go. For `BACON` we represent it by the expressions found at a given layer. To move to the next node, the legal actions must be determined. In chess, these are the set of legal moves. In `BACON` these are the set of expressions to choose from, apart from at the final layer where it is a selection of hyperparameters to use in `BACON.1`. MCTS terminates when the game is over. In chess that would be at checkmate or stalemate, for `BACON` that would be when each layer of the tree has been searched. Lastly, scoring. In chess, it is +1 for a win, 0 for a draw or -1 for a loss. For `BACON` it is a function of MSE($V_{\texttt{BACON}}, V + n$) (referred to as the reward function).

The MCTS algorithm works through 4 steps after being initialised at a given node called a parent node:

1. *SELECTION*: A child node is selected to travel from those available to the parent node (through performing a legal action) based on the score of the child node. The score is weighted by a parameter C. If C is small, the parent node prefers exploitation and picks a child node based on what has returned high scores in the past. If C is larger, the parent prefers exploration and picks a child node which has not been selected much.
2. *EXPANSION*: After the parent chooses a child, this child becomes the parent node. The legal actions are run to determine the new selection of children nodes.
3. *SIMULATION*: This process is repeated until a full game has been simulated. The result of the game is given a score by the reward function.
4. *BACKPROPAGATION*: This score is backpropagated through all the nodes selected to the initial parent node, altering their S_i value.

The purpose of the MCTS is to find the node with the best score after running this process multiple times. It implies this node is the correct one to move to, to maximise winning. In `BACON` this translates to picking the best expression in a layer that maximises the score. Under an assumption that the best expression in earlier layers will always minimise the MSE (and maximise the score), MCTS is a logical continuation that can search the possible expressions generated by the sets and correctly pick the best expression at each layer. A visualisation of this on the Ideal Gas Law is in Figure 14.

My reward function takes in multiple factors, such as how many variables make up the expression, how quick it took `BACON` to output the expression as well as the MSE. It is not a continuous function