Estimating the CY_3's in the Kreuzer-Skarke Dataset

Brent D. Nelson

Machine Learning Landscape Workshop, ICTP

hep-th/1811.06490, with R. Altman, J. Carifio, and J. Halverson

Northeastern University

Motivation

$\Rightarrow$ Six years of working with the Kreuzer-Skarke 4D Reflexive Polytope dataset!

- Database of triangulations $\rightarrow$ Calabi-Yau threefolds (1411.1418)
- Finding valid orientifolds and fixed loci for model building (1901.xxxxx)
- Finding Large Volume limits for moduli stabilization (1207.5801, 1706.09070, 1901.xxxxx)
- Using it as a test-bed for data science techniques (1707.00655, 1711.06685, 1811.06490)

$\Rightarrow$ What are the goals from a machine learning perspective?

- Ultimately want *interpretable* results – from "data analytics" to analytical answers
- So-called "Equation Learner" (EQL) neural network architecture intended to deliver just that George Martius & Christoph Lampert (1610.02995 cs.CL)

Let's Just Calculate!

⇒ Our ultimate goal: true knowledge of the number of Calabi-Yau threefolds (CY3s) in the Kreuzer-Skarke (KS) database

- We know the number of reflexive polytopes: 473,800,776
- Most polytopes admit multiple fine, regular, star triangulations (FRSTs)
  - ⋆ Through $h^{1,1} \leq 6$, 23,568 polytopes yielded 651,997 triangulations
- Generally, many triangulations are identified as representing different chambers of the Kähler cone for a single CY3 geometry
  - ⋆ Through $h^{1,1} \leq 6$, 651,997 triangulations yielded 101,673 unique CY3s

⇒ "Brute force" method is **not** an option

- Finding *all* FRSTs for a polytope becomes computationally prohibitive with `TOPCOM` at $h^{1,1} \gtrsim 25$

Alternative Approach

$\Rightarrow$ Can machine learning help?

- Possibly, but to train a model requires many (input, output) pairs
- This means knowing the exact number of FRSTs for many polytopes – simply not possible at this time

$\Rightarrow$ Our approach: focus on counting triangulations of the 3D **facets** that constitute 4D polytopes

- Total number of unique 3D facets an order of magnitude smaller (45,990,557)
- Obtaining all fine, regular triangulations (FRTs) of these tends to be easier

$\Rightarrow$ We will estimate the number of FRSTs of a 4D reflexive polytope via

$$N_{\mathrm{FRST}}(\Delta) \leq \prod_i N_{\mathrm{FRT}}(F_i)\,,$$

- NB(1): Triangulations of facets F_1 and F_2 may not overlap on the intersection $F_1 \bigcap F_2$
- NB(2): Even if triangulations of F_1 and F_2 are regular, aggregate triangulation may fail to be regular

Classification of 3D Facets

⇒ Identifying 3D facets of 4D polytopes is fast with a (C++ implementation of) PALP, but identifying **unique** facets is more challenging

- Total number of 3D facets is 7,471,984,487 – a major step backward?!?
- Need a common form so as to identify equivalent facets
- Kreuzer and Skarke identified a *normal form* for 4D polytopes related by $\mathbb{GL}(n,\mathbb{Z})$ transformations – just need to adapt to 3D facets

⇒ Example: consider the following two facets F_1 and F_2, both of which appear as dual facets to the same $h^{1,1}=2$ polytope:

$$F_1 = \text{conv}(\{\{-1,0,0,0\},\{-1,0,0,1\},\{-1,0,1,0\},\{-1,1,0,0\}\})$$

$$F_2 = \text{conv}(\{\{-1,0,0,1\},\{-1,0,1,0\},\{1,0,0,0\},\{2,-1,-1,-1\}\})$$

- Adding the origin to each facet, we obtain the associated subcones

$$C_{F_1} = \text{conv}(\{\{0,0,0,0\},\{-1,0,0,0\},\{-1,0,0,1\},\{-1,0,1,0\},\{-1,1,0,0\}\})$$

$$C_{F_2} = \text{conv}(\{\{0,0,0,0\}\{-1,0,0,1\},\{-1,0,1,0\},\{1,0,0,0\},\{2,-1,-1,-1\}\})$$

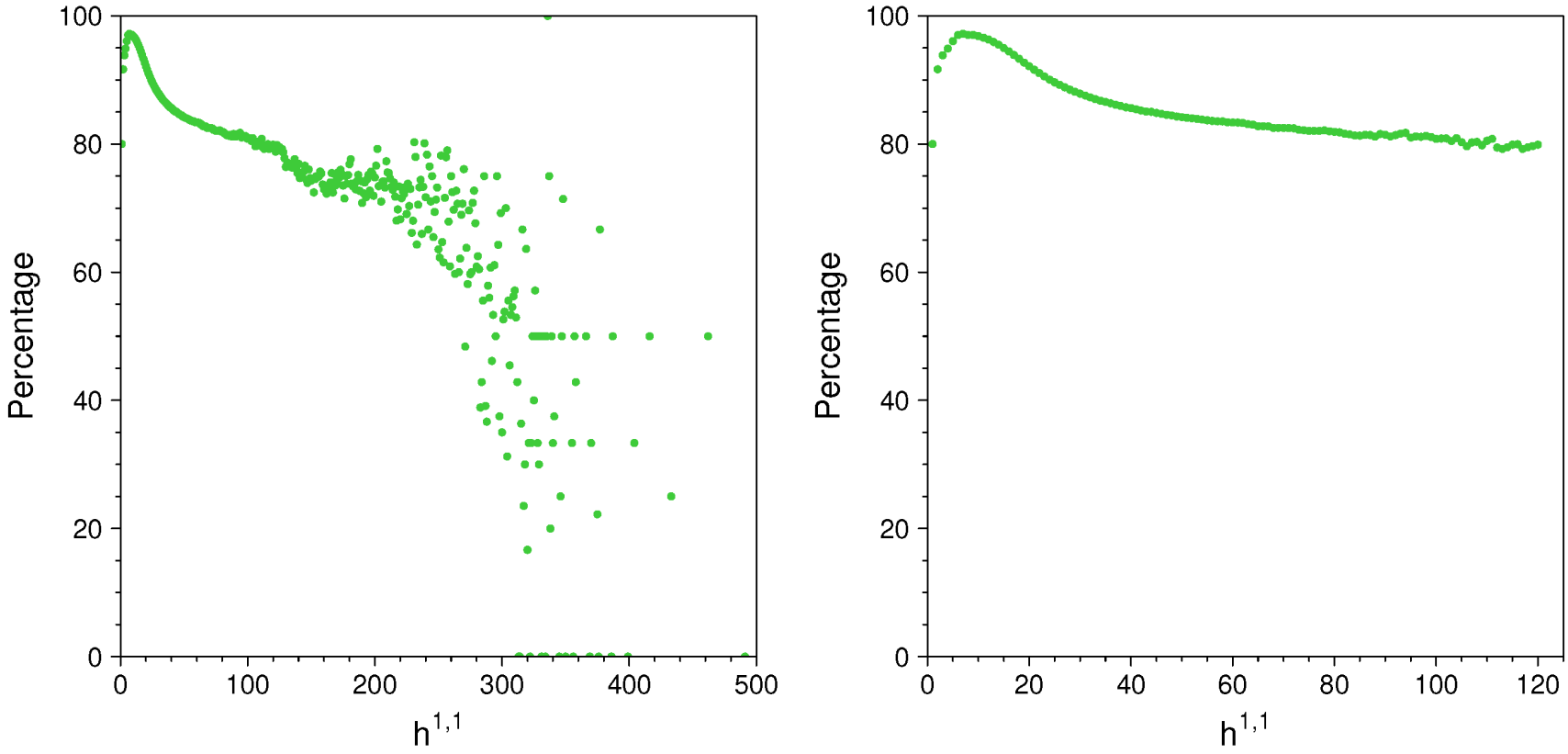
- Computing the normal form for each subcone, we find that

$$\text{NF}(C_{F_1}) = \text{NF}(C_{F_2}) = \{\{0,0,0,0\},\{1,0,0,0\},\{0,1,0,0\},\{0,0,1,0\},\{0,0,0,1\}\})$$

The Standard 3-Simplex Facet

⇒ Dropping the origin, we recognize the standard 3-simplex (S3S)

$$\{\{1,0,0,0\},\{0,1,0,0\},\{0,0,1,0\},\{0,0,0,1\}\}$$



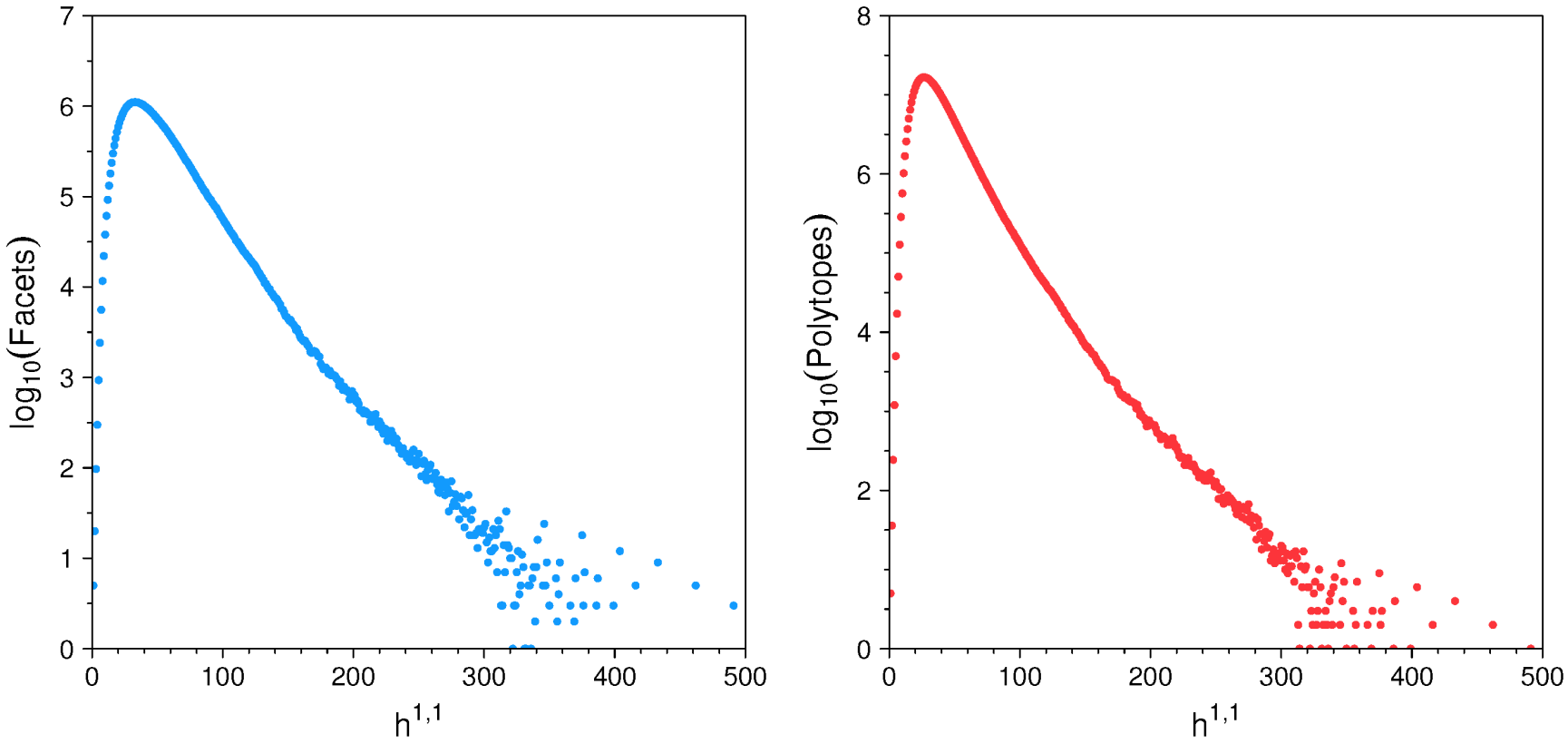
(Left) Percentage of dual polytopes that contain S3S at each $h^{1,1}$ value. **(Right)** Same, truncated at $h^{1,1} \leq 120$.

- Represents 1,528,150,671 of the 3D facets (20.45%)
- Appears at least once in 87.8% of all 4D polytopes
- Has a unique triangulation, therefore not contributing to combinatorics

Results of 3D Facet Classification

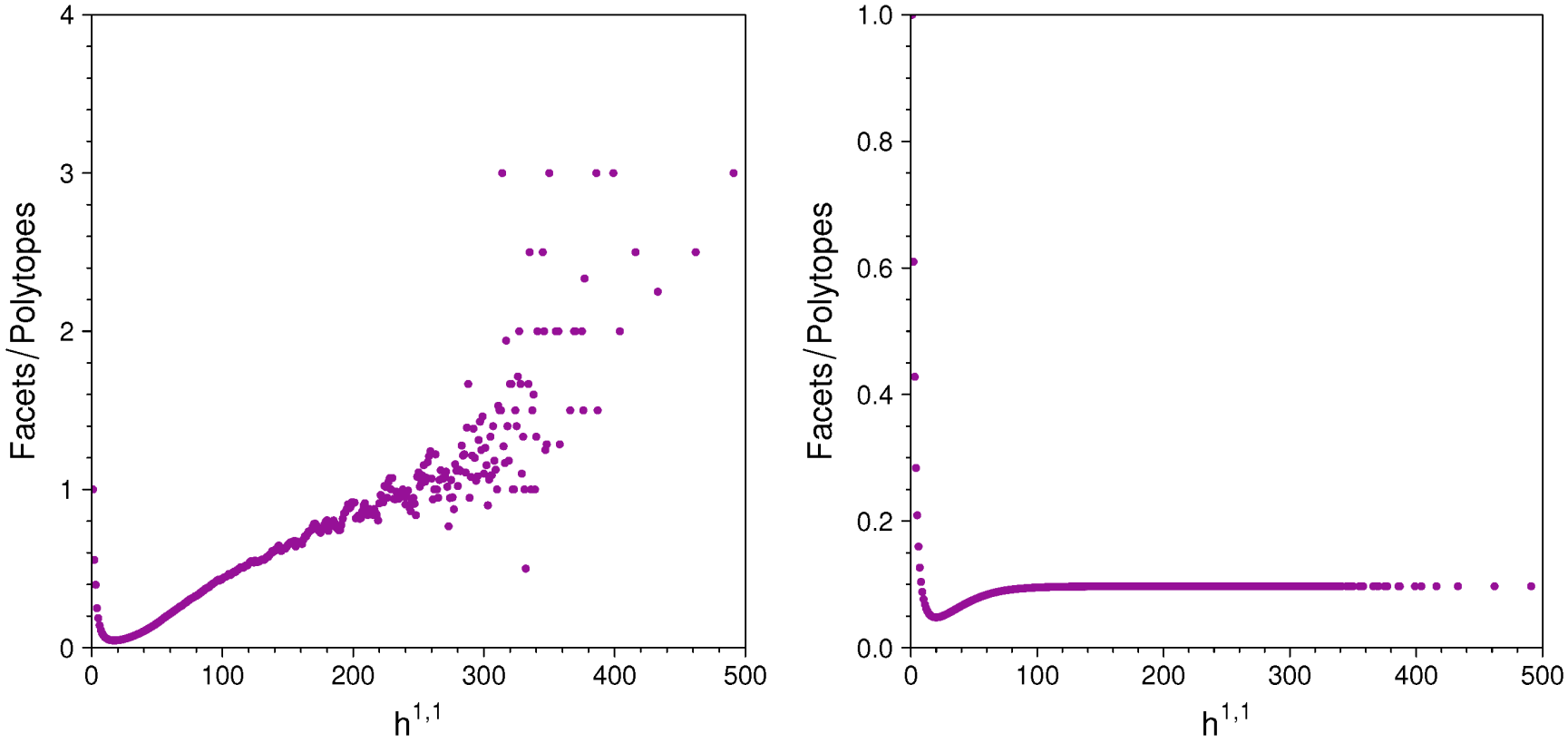
⇒ Total number of 3D facets is 7.5 billion, but unique total only 46 million (0.6%)

⇒ S3S accounts for 20.45% of all facets. Next most common accounts for 8.6%



(Left) The logarithm of the number of new facets at each $h^{1,1}$ value. **(Right)** The logarithm of the number of reflexive polytopes at each $h^{1,1}$ value.

3D Facet Distribution – New Facets

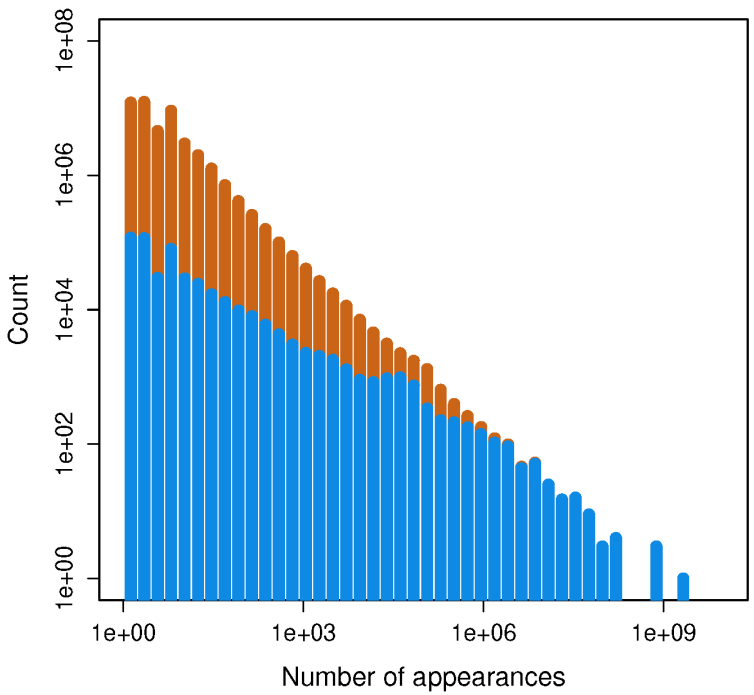


(Left) The number of new facets at each $h^{1,1}$ value, as a fraction of the number of polytopes at that $h^{1,1}$. **(Right)** The total number of facets found through each $h^{1,1}$ value, as a fraction of the total number of polytopes up to that point.

$\Rightarrow$ Saturation to value of 0.1 is just the ratio of total unique facets found (47×10^6) to the number of 4d reflexive polytopes in the KS database (470×10^6)

Triangulated 3D Facets

$\Rightarrow$ Of these 3D facets, we know quite a lot about a large fraction of them



Orange bars: total number of facets

Blue bars: amount for which the number of FRTs is explicitly computed

$h^{1,1}$	Facets	Triangulated	% Triangulated
$1-11$	142,257	142,257	100%
12	92,178	92,162	99.983%
13	132,153	108,494	82.097%
14	180,034	124,700	69.625%
15	236,476	3,907	1.652%
>15	45,207,459	1,360	0.003%
Total	45,990,557	472,896	1.028%

Table 1: Dual facet FRT numbers obtained, binned by the first $h^{1,1}$ value at which they appear.

- 100 most common facets account for 74% of all cases
- Able to obtain FRTs for 472,880 3D facets (1.03% of total)
- 3D facets with known triangulation numbers represent 88% of facets by appearance

Supervised ML Results

⇒ Last year (1707.00655), we were able to predict the number of FRSTs of 3D polytopes using simple supervised ML

- Input data was a simple 4-tuple: numbers of points, interior points, boundary points, and vertices
- Pulled models "out of the box" from `scikit-learn`
- Figure of merit was the mean absolute percent error (MAPE) of the prediction relative to true results in training/test data:

$$\text{MAPE} = \frac{100}{n} \times \sum_{i=1}^{n} \left| \frac{A_i - P_i}{A_i} \right| ,$$

where n is the number of data points, and P_i and A_i are the predicted and actual values for the output, which here is $\ln(N_{\text{FRT}})$ for the i^{th} facet

⇒ In 2017, we obtained good results with the Classication and Regression Tree (CART) model. How will it perform on the 4D case?

Regression Results

⇒ Here we present the results for `ExtraTreesRegressor`, with 35 estimators, employing a 60%/40% train/test split on data for $5 \leq h^{1,1} \leq 10$

- Training MAPE: 5.723
- Test MAPE: 5.823

⇒ Good, but how well do the results extrapolate to higher $h^{1,1}$ values?

$h^{1,1}$	**MAPE**	**Actual mean**	**Predicted mean**
11	6.566	9.582	9.189
12	9.065	10.882	9.903
13	11.566	11.755	10.067
14	17.403	12.638	10.179

Table 2: Prediction results for $\ln(N_{\rm FRT})$, using the `ExtraTreesRegressor` model, for $h^{1,1}$ values outside of its training region.

- MAPE gets rapidly worse as $h^{1,1}$ grows
- Persistent, and growing, undercount of FRTs
- Largest prediction was $\ln(N_{\rm FRT}) = 12.467$; largest value seen in training data was $\ln(N_{\rm FRT}) = 12.595$

A Generic Neural Network

$\Rightarrow$ Generic feed-forward NN applied to 4-tuples with no improvement on results

$\Rightarrow$ First though was to expand the input variables – "kitchen sink" approach

- The number of points in the interior and on the boundary (x_0, x_1)
- The number of vertices (x_2)
- The number of points in the 1- and 2-skeletons (x_3, x_4)
- The first $h^{1,1}$ value at which the facet appears in a dual polytope (x_5)
- The number of faces and edges (x_6, x_7)
- The number of flips of a seed triangulation of the 2-skeleton (x_8)
- Several quantities obtained from a single FRT of the facet:
  - ⋆ The total numbers of 1-, 2-, and 3-simplices in the triangulation (x_9,x_{10},x_{11})
  - ⋆ The numbers of unique 1- and 2-simplices in the triangulation (x_{12},x_{13})
  - ⋆ The numbers of 1- and 2-simplices shared between N 2- and 3-simplices, respectively, for N up to 5 ($x_{14} - x_{17}$, $x_{18} - x_{21}$)

A Simple Neural Network Implementation

$\Rightarrow$ Our simple feed-forward NN has two hidden layers, each with 30 nodes

$\Rightarrow$ Activation functions: sigmoid (layer 1), $\tanh$ (layer 2), ReLU (output layer)

$\Rightarrow$ Train on equal numbers of data points for each $h^{1,1}$ value between $6 \leq h^{1,1} \leq 11$

- Overall MAPE on test data for $6 \leq h^{1,1} \leq 11$ acceptable: 6.304, how about extrapolation?

$h^{1,1}$	**MAPE**	**Mean value**	**Predicted mean**
12	5.904	10.882	10.324
13	6.550	11.755	10.753
14	10.915	12.638	11.094

Table 3: Prediction results for $\ln(N_{\mathrm{FRT}})$, using the traditional neural network, for $h^{1,1}$ values outside of its training region.

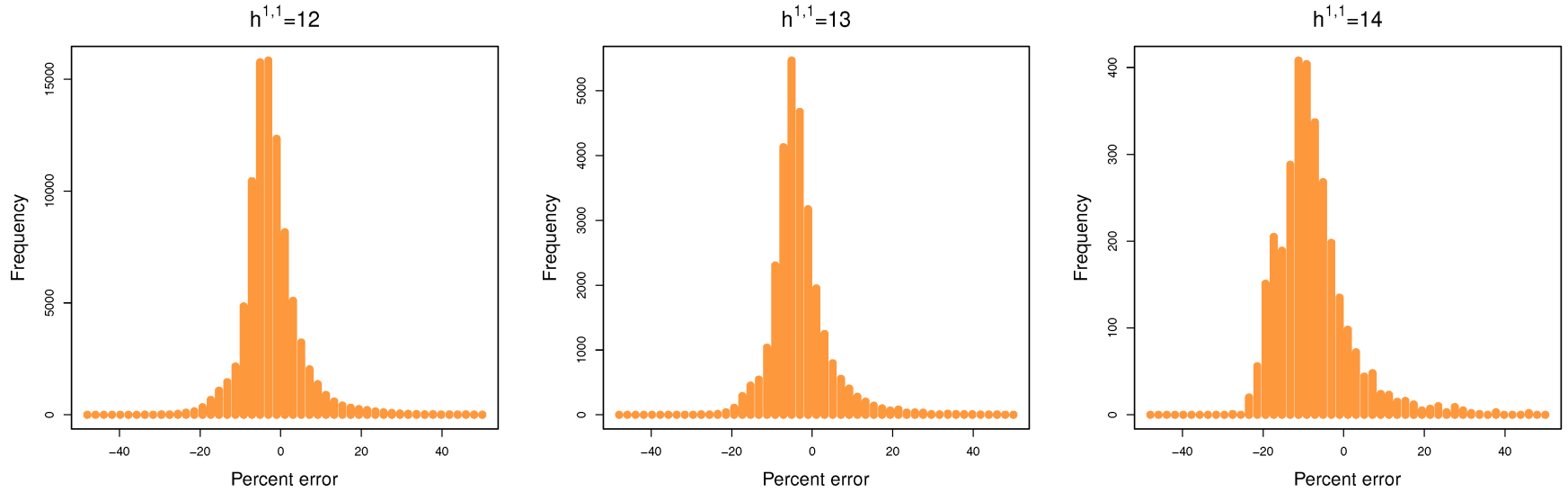
$\Rightarrow$ Same problems!

- MAPE continues to get worse rapidly as $h^{1,1}$ grows
- Continues to universally under-predict the number of FRTs

Simple Neutral Network Results

$h^{1,1}$	MAPE	Mean value	Predicted mean
12	5.904	10.882	10.324
13	6.550	11.755	10.753
14	10.915	12.638	11.094

Table 4: Prediction results for $\ln(N_{\mathrm{FRT}})$, using the traditional neural network, for $h^{1,1}$ values outside of its training region.



Histograms of the percent error of the feed-forward neural network's predictions in the extrapolation region.

The EQL Architecture

⇒ The **equation learner** (EQL) NN architecture is designed to permit greater ability to extrapolate beyond the training region

George Martius & Christoph Lampert (1610.02995 cs.CL)

- The standard activation function in each layer is replaced partially, or even completely, with non-linear functions that do **not** necessarily try to mimic human neurons
- Non-linear layers change the shape of the output vector from the linear node

⇒ A simple example might be to multiply the outputs of two nodes together, then feed forward to the next layer

- Unary nodes: apply a standard activation function (e.g. $\tanh$)
- Binary nodes: pairwise multiply n nodes, yielding $n/2$ outputs

⇒ Name derives the desire of the authors to have an **intelligible** NN output

- "...our goal is not to learn any data representation, but to learn a function which compactly represents the input-output relation and generalizes between different regions of the data space, like a physical formula." (c.f. Hashimoto talk)

Examples of EQL in Action

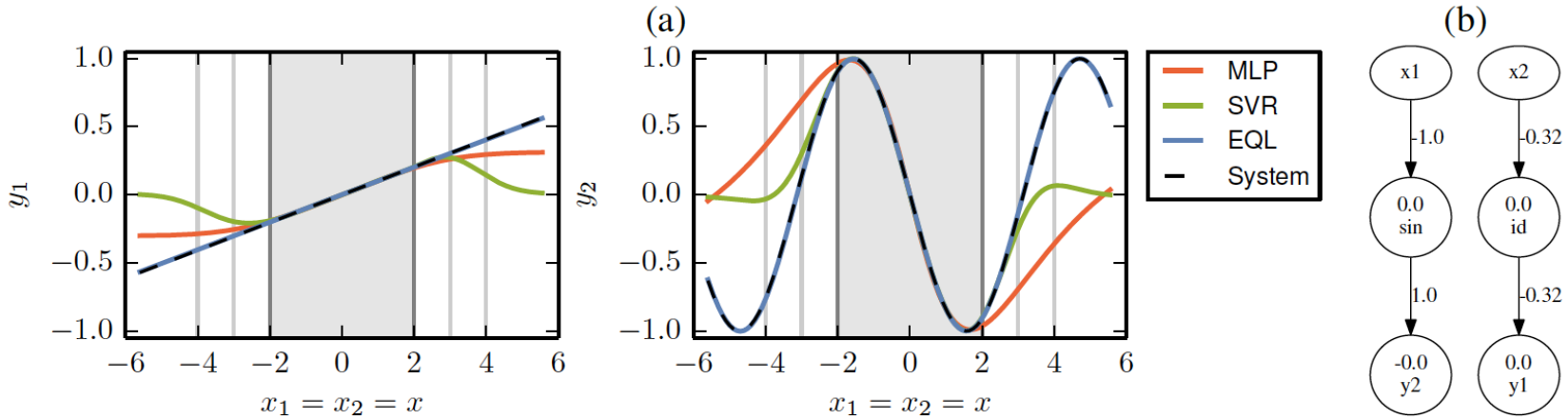
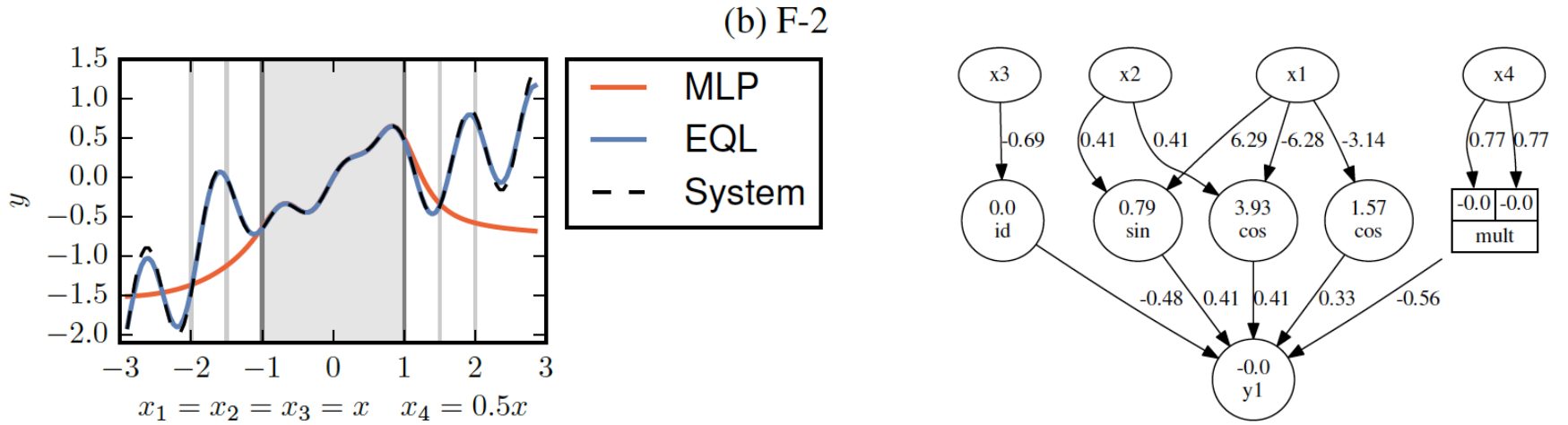


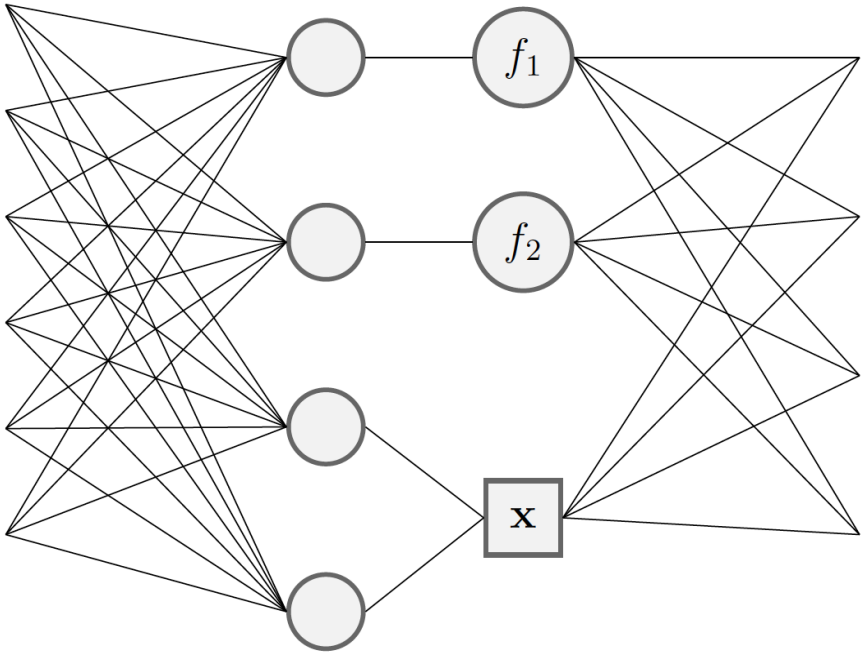
Figure 2: Learning pendulum dynamics. (a) slices of outputs y_1 (left) and y_2 (right) for inputs $x_1 = x_2 = x$ for the true system equation (Eq. 9) and one of EQL, MLP, SVR instances. The shaded area marks the training region and the vertical bars show the size of the *near* and *far* extrapolation domain. (b) one of the learned networks. Numbers on the edges correspond to the entries of W and numbers inside the nodes show the bias values b. All weights with $|w| < 0.01$ and orphan nodes are omitted. Learned formulas: $y_1 = 0.103x_2$, $y_2 = \sin(-x_1)$, which are correct up to symmetry ($1/g = 1.01$).



learned formula: $0.33\cos(3.14x_1 + 1.57) + 0.33x_3 - 0.33x4^2 +$

$0.41\cos(-6.28x_1 + 3.93 + 0.41x_2) + 0.41\sin(6.29x_1 + 0.79 + 0.41x_2)$

The EQL Architecture



A representation of a simple EQL layer with $n_m = 4$ and $n_o = 3$ sandwiched between two fully-connected layers. The first two elements of the intermediate representation are each acted on by activation functions f_i, while the remaining two elements are multiplied together.

$\Rightarrow$ Our EQL NN will have an input layer of 22 nodes, a single hidden layer of 45 nodes (30 binary and 15 unary), and an output layer of 30 nodes (with ReLU activiation)

- We use Adam optimizer with default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.99$)
- We utilize L1 regularization with $\lambda = 0.001$, dropout rate $p = 0.1$

EQL Results I

			Extrapolation MAPE		
$h^{1,1}_{\min}$	$h^{1,1}_{\max}$	**Test MAPE**	$h^{1,1} = 12$	$h^{1,1} = 13$	$h^{1,1} = 14$
6	10	7.297	6.647	6.699	6.598
7	10	6.001	7.512	7.626	7.469
8	10	7.184	5.048	5.172	5.834
6	11	5.643	4.393	4.490	4.416
7	11	6.967	7.512	7.626	7.469
8	11	5.551	4.444	4.463	4.934

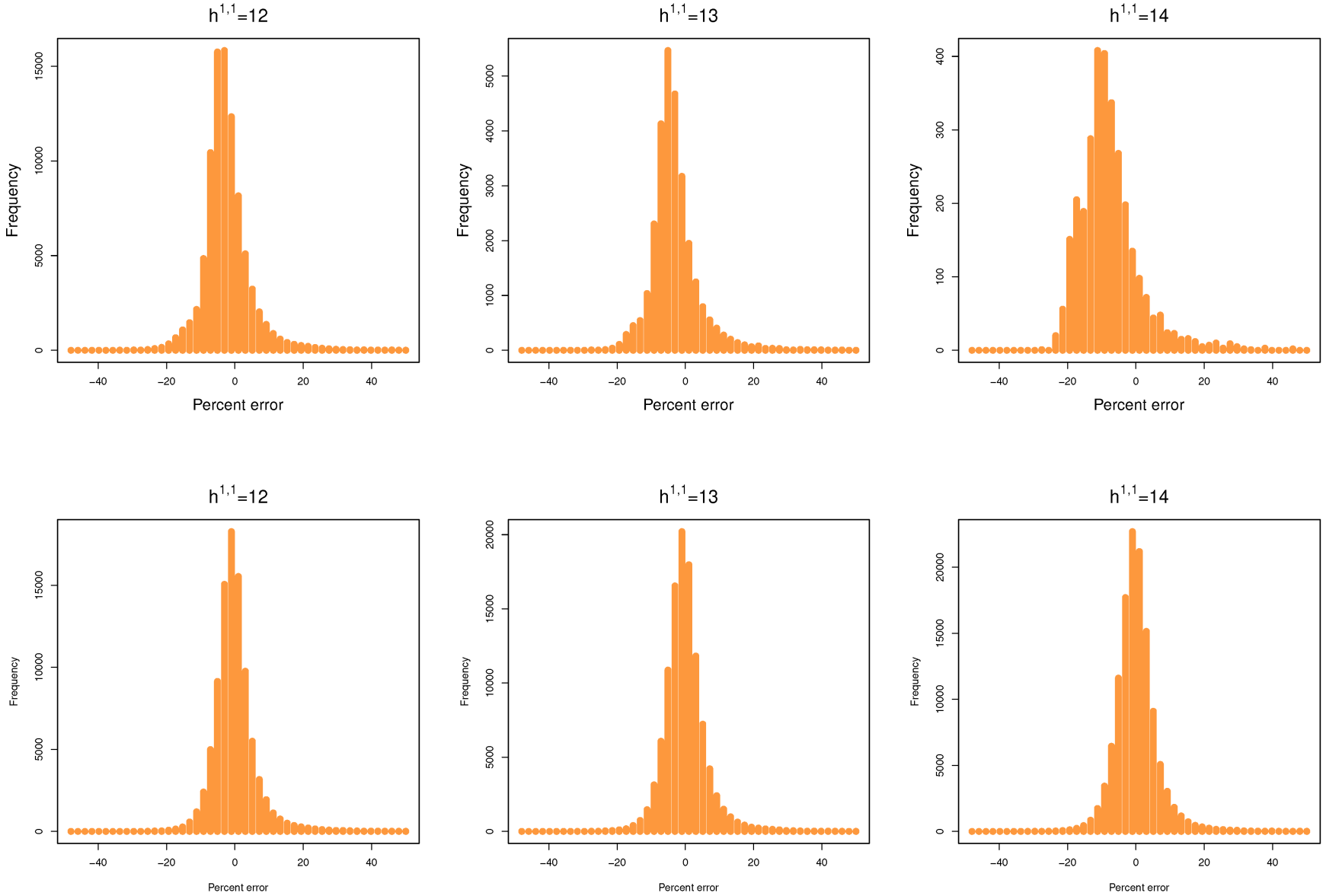
Table 5: Results of training our model on various $h^{1,1}$ ranges. The model with $h^{1,1}_{\min} = 6$, $h^{1,1}_{\max} = 11$ performs well on the test set and the best on extrapolation to higher $h^{1,1}$ values.

$h^{1,1}$	**Mean value**	**Predicted mean**
12	10.733	10.722
13	11.755	11.591
14	12.638	12.492

Table 6: The true mean values and the mean predicted by our model in the extrapolation region

- MAPE on test data is **not** growing appreciably for higher $h^{1,1}$ values
- Problem of consistent, large under-counting is (mostly) solved

EQL Results II



Histograms of the percent error of our chosen model's predictions in the extrapolation region. Top is the naive NN. Bottom is the EQL NN.

Meet the 4D Reflexive Polytope Δ°_{491}

The polytope whose FRST count dominates the database is the polytope dual to the single $h^{1,1} = 491$ polytope, which we will call Δ°_{491}.

$$\Delta^\circ_{491} = \{\{1,0,0,0\},\{0,1,0,0\},\{0,0,1,0\},\{21,28,36,42\},\{-63,-56,-48,-42\}\}$$

This polytope has 680 integral points and five facets, of which only four are unique. The four facets F_i are given by the convex hulls

$$\begin{aligned} F_1 &= \text{conv}(\{\{1,0,0,0\},\{0,1,0,0\},\{0,0,1,0\},\{21,28,36,42\}\}) \\ F_2 &= \text{conv}(\{\{1,0,0,0\},\{0,1,0,0\},\{3,4,6,0\},\{3,4,6,84\}\}) \\ F_3 &= \text{conv}(\{\{1,0,0,0\},\{0,1,0,0\},\{7,8,14,0\},\{7,8,14,84\}\}) \\ F_4 &= \text{conv}(\{\{1,0,0,0\},\{0,1,0,0\},\{7,15,21,0\},\{7,15,21,84\}\})\,. \end{aligned} \tag{1}$$

The facet F_1 first appears as a dual facet at $h^{1,1} = 23$, and appears twice in Δ°_{491}. The facets F_2, F_3 and F_4 each appear once in Δ°_{491} and nowhere else in the database.

FRST Prediction for Δ°_{491}

The EQL model predicts the following results for $\ln(N_{FRT})$ for these facets:

$$
\begin{aligned}
F_1: &\quad \ln(N_{FRT}) = 29.32 \pm 1.30 \\
F_2: &\quad \ln(N_{FRT}) = 2391.5 \pm 106.0 \\
F_3: &\quad \ln(N_{FRT}) = 10753.0 \pm 476.7 \\
F_4: &\quad \ln(N_{FRT}) = 10985.9 \pm 487.0
\end{aligned}
$$

where we are employing an error estimate based on the average MAPE for $12 \leq h^{1,1} \leq 14$ of 4.416. Using just the central values yields the prediction:

$$
\begin{aligned}
N_{\rm FRST}(\Delta^{\circ}_{491}) &= \left(e^{29.32}\right)^2 \left(e^{2391.5}\right) \left(e^{10,573.0}\right) \left(e^{10,985.9}\right) \\
&= (2.93 \times 10^{25})(4 \times 10^{1038})(1 \times 10^{4670})(1.25 \times 10^{4771}) \\
&= 1.5 \times 10^{10,505}
\end{aligned}
$$

Propagating the errors (assuming MAPE of 4.416 throughout), gives a crude estimate of the range of possible values for Δ°_{491}:

$$
N_{\rm FRST} = 10^{10,505.2 \pm 292.6} = [10^{10,212.6}, 10^{10,797.8}]
$$

Cross-Check: 3D Case (2D Facets)

⇒ This is a bold claim, given that we are extrapolating from $h^{1,1} = 14$ to $h^{1,1} = 491$, how can we check the method?

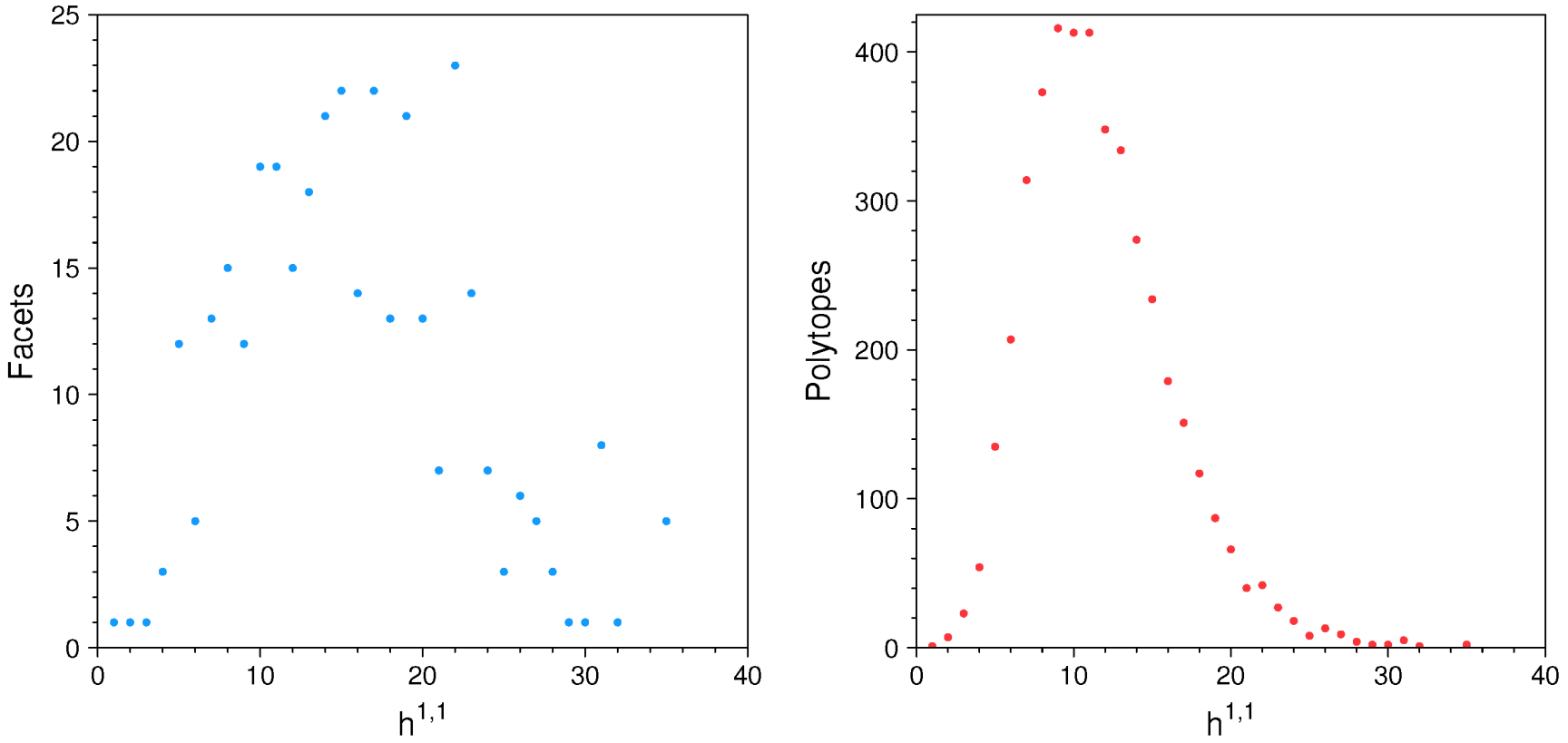
⇒ In previous work, the 3D reflexive polytopes have been extensively studied

Halverson & Tian (1610.08864)
Carifio, Halverson, Krioukov, BDN (1707.00655)

- Far fewer of them: 4,319 polytopes
- Only 344 unique 2D facets! Big enough to train on?
- We have exact FRT counts for most of these facets: 322 of 344 (including all cases for $h^{1,1} \leq 25$
- For six of the remaining, we can compute the number of FTs (dropping regularity), which is only a slight over-count of FRTs (less than 3%)

⇒ The region of extrapolation to the largest 3D polytopes is therefore much smaller than 4D case, but still represents substantial growth in FRTs

Cross-Check: 3D Case (2D Facets)



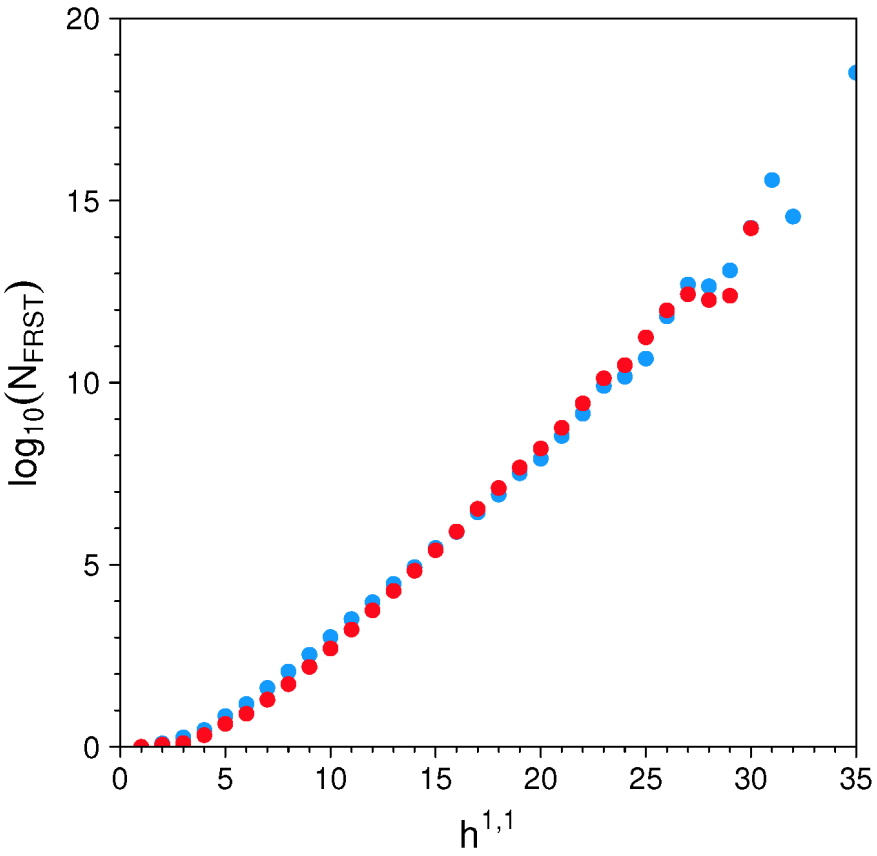
(Left) The number of new facets at each $h^{1,1}$ value. **(Right)** The number of reflexive polytopes at each $h^{1,1}$ value.

3D Case: Inputs to the EQL

Having triangulated as many facets as possible, we trained models with an EQL hidden layer. Our input data was simpler than for the 3D facets, and consisted of:

- The number of integral points
- The number of boundary points
- The number of interior points
- The number of vertices
- The length of the longest side
- The length of the shortest side
- The average side length

3D Case: Prediction Results



The mean value of $\log_{10}(N_{\mathrm{FRST}})$ at $h^{1,1}$, using predicted facet FRT values **(blue)** and known facet FRT and FT values **(red)**.

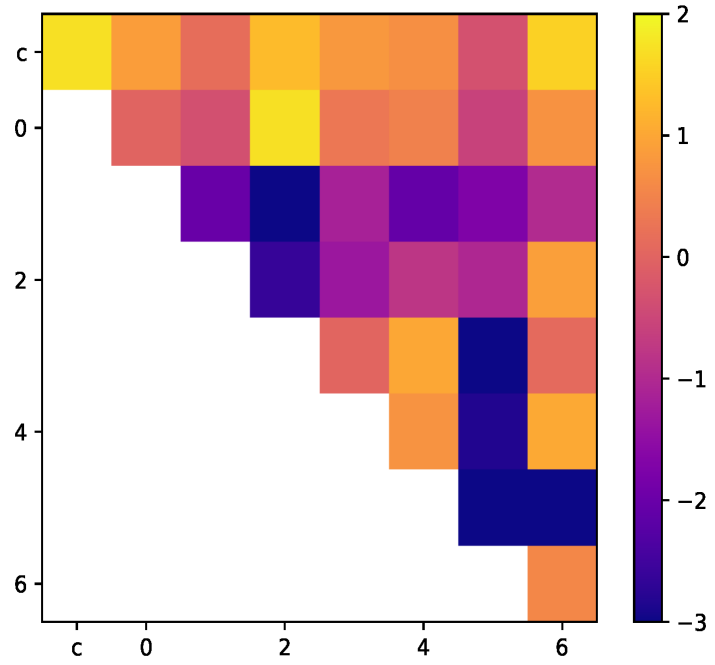
$h^{1,1}$	Facets	MAPE	MSE
6	5	7.865	0.032
7	13	16.583	0.061
8	15	8.805	0.055
9	12	5.851	0.075
10	19	5.808	0.087
11	19	10.678	0.213
12	15	8.754	0.334
13	18	9.128	0.330
14	21	10.200	0.722
15	22	8.756	0.538
16	14	9.103	0.755
17	22	9.071	0.610
18	13	7.850	0.619
19	21	10.491	1.314
20	13	10.962	2.050
21	7	9.259	1.158
22	23	9.167	0.798
23	14	14.333	6.504
24	7	7.894	1.075
25	3	2.649	0.373
26	6	12.112	2.228
27	5	3.985	0.644
28	3	6.900	0.380
29	1	5.258	0.295
30	1	2.074	0.146

Epilogue: Interpretation of 3D Result?

$\Rightarrow$ EQL is meant to yield a NN output that is meaningful... does it?

$$\begin{aligned}\ln(N_{\mathrm{FRT}}) &= 0.01418x_0^2 - 0.03435x_0x_1 - \\ &0.02165x_1^2 + 0.11134x_0x_2 + 0.00201x_1x_2 + \\ &0.00206x_2^2 - 0.03566x_0x_3 - 0.02813x_1x_3 + \\ &0.00993x_2x_3 + 0.05023x_3^2 - 0.03399x_0x_4 - \\ &0.00929x_1x_4 - 0.01405x_2x_4 + 0.11072x_3x_4 + \\ &0.0694x_4^2 + 0.04551x_0x_5 - 0.04939x_1x_5 + \\ &0.04087x_2x_5 - 0.00532x_3x_5 + 0.00719x_4x_5 - \\ &0.00774x_5^2 - 0.07105x_0x_6 + 0.04438x_1x_6 - \\ &0.11917x_2x_6 - 0.07082x_3x_6 - 0.14734x_4x_6 - \\ &0.007x_5x_6 + 0.14731x_6^2 - 0.28707x_0 + \\ &0.46716x_1 - 0.59766x_2 - 0.4975x_3 - \\ &0.35609x_4 - 0.49381x_5 + 1.354040x_6 + \\ &5.530190\end{aligned}$$

- x_6: length of longest side
- x_5: length of smallest side



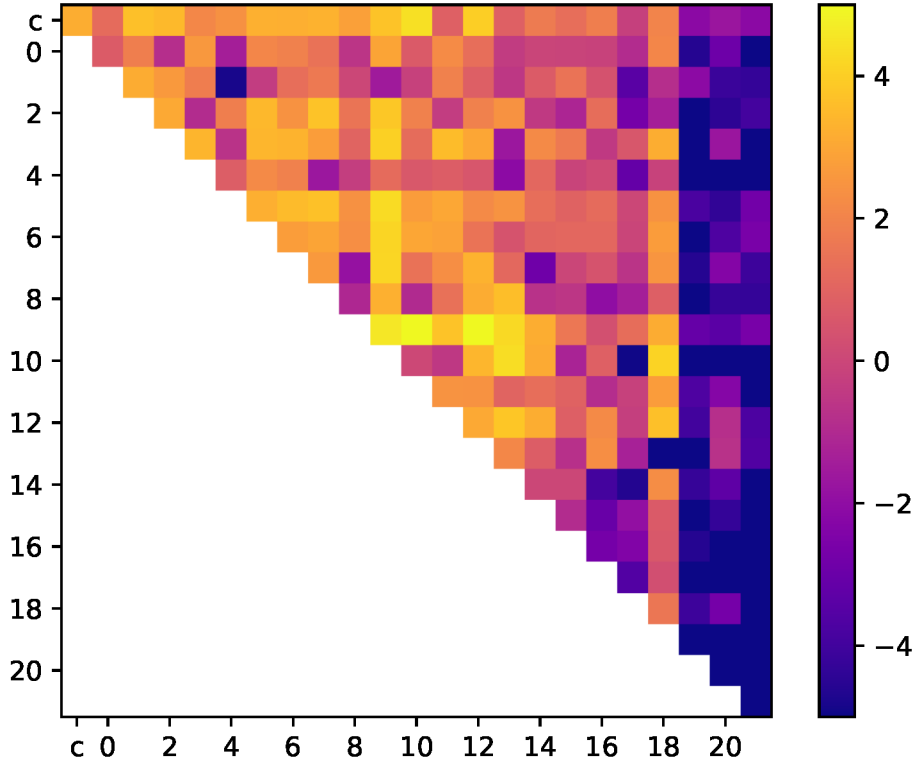
A heatmap showing $\log_{10}(|c|)$ for each coefficient c in the formula, after rescaling each variable to have expectation value 1. The top row corresponds to the constant term (top left square) and the linear terms.

Interpretation of 4D Result?

$$\ln(N_{\mathrm{FRT}}) = -0.425x_0^2+0.29447x_0x_1-0.2304x_1^2+0.02462x_0x_2-0.17529x_1x_2-0.3368x_2^2+0.72012x_0x_3+ 0.0707x_1x_3 + 0.00583x_2x_3 - 0.40825x_3^2 - 0.01146x_0x_4 - 0.00008x_1x_4 - 0.0789x_2x_4 + 0.00599x_3x_4 - 0.02246x_4^2 + 0.35742x_0x_5 + 0.00696x_1x_5 + 0.39255x_2x_5 - 0.35135x_3x_5 + 0.09x_4x_5 - 0.2482x_5^2 - 0.19063x_0x_6 - 0.02357x_1x_6 - 0.08904x_2x_6 + 0.20651x_3x_6 - 0.04321x_4x_6 + 0.21098x_5x_6 - 0.0609x_6^2 + 0.20861x_0x_7 + 0.05763x_1x_7 + 0.5381x_2x_7 - 0.19461x_3x_7 + 0.00197x_4x_7 - 0.41043x_5x_7 + 0.12497x_6x_7 - 0.15195x_7^2 + 0.02359x_0x_8 + 0.0095x_1x_8 + 0.05497x_2x_8 - 0.03016x_3x_8 + 0.00692x_4x_8 - 0.10282x_5x_8 + 0.06078x_6x_8 + 0.00153x_7x_8 + 0.00301x_8^2 - 0.29528x_0x_9 - 0.00072x_1x_9 - 0.20046x_2x_9 + 0.2274x_3x_9 - 0.01136x_4x_9 + 0.25023x_5x_9 - 0.13467x_6x_9 + 0.23766x_7x_9 + 0.08246x_8x_9 - 0.11071x_9^2 + 0.02683x_0x_{10} - 0.00254x_1x_{10} + 0.02553x_2x_{10} + 0.01278x_3x_{10} + 0.00554x_4x_{10} - 0.04584x_5x_{10} + 0.03714x_6x_{10} + 0.01352x_7x_{10}+0.00105x_8x_{10}+0.20665x_9x_{10}-0.00099x_{10}^2-0.34428x_0x_{11}+0.05335x_1x_{11}-0.00715x_2x_{11}+ 0.33024x_3x_{11} - 0.01736x_4x_{11} + 0.16713x_5x_{11} - 0.08848x_6x_{11} + 0.08456x_7x_{11} + 0.03144x_8x_{11} - 0.11092x_9x_{11}-0.00151x_{10}x_{11}-0.07556x_{11}^2+0.02349x_0x_{12}+0.0031x_1x_{12}-0.01155x_2x_{12}-0.03125x_3x_{12}- 0.00223x_4x_{12} + 0.01197x_5x_{12} - 0.00373x_6x_{12} - 0.0372x_7x_{12} + 0.02864x_8x_{12} - 0.07238x_9x_{12} + 0.01227x_{10}x_{12} + 0.0125x_{11}x_{12} - 0.00375x_{12}^2 - \cdots - 0.00024x_0x_{21} + 0.01386x_1x_{21} + 0.02516x_2x_{21} + 0.00222x_3x_{21} + 0.00004x_4x_{21} - 0.06183x_5x_{21} + 0.04606x_6x_{21} + 0.01789x_7x_{21} - 0.01325x_8x_{21} + 0.02409x_9x_{21} + 0.00023x_{10}x_{21} + 0.00006x_{11}x_{21} - 0.00317x_{12}x_{21} + 0.00541x_{13}x_{21} - 0.00012x_{14}x_{21} + 0.00012x_{15}x_{21} - 0.00592x_{16}x_{21} + 0.02185x_{17}x_{21} - 0.00001x_{18}x_{21} - 0.00003x_{20}x_{21} + 0.0257x_{21}^2 - 1.548990x_0+3.819380x_1+4.156280x_2-1.006350x_3+1.110140x_4-2.591030x_5+1.540610x_6-2.829660x_7- 1.5943x_8 + 1.383880x_9 - 2.551x_{10} + 0.18922x_{11} + 0.72398x_{12} + 0.04856x_{13} + 0.48053x_{14} + 0.6916x_{15} - 1.253460x_{16} - 0.70274x_{17} + 0.44341x_{18} - 1.167680x_{19} - 1.8572x_{20} - 1.170990x_{21} - 23.70712$$

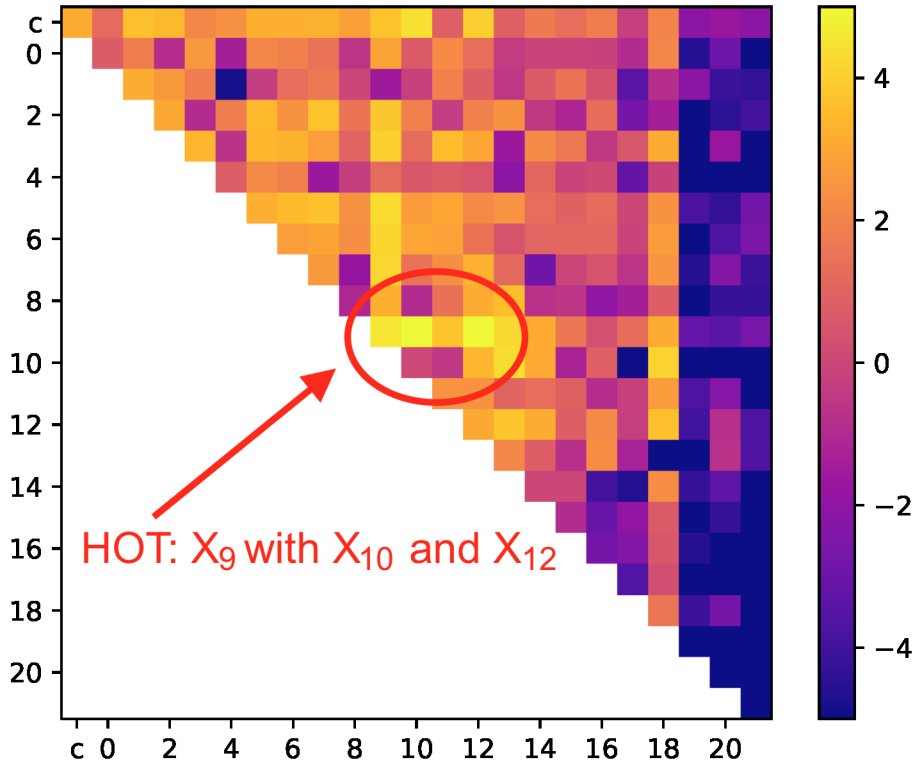
Interpretation of 4D Result?

⇒ What matters most in determining N_{FRT} for a 3D facet?



Interpretation of 4D Result?

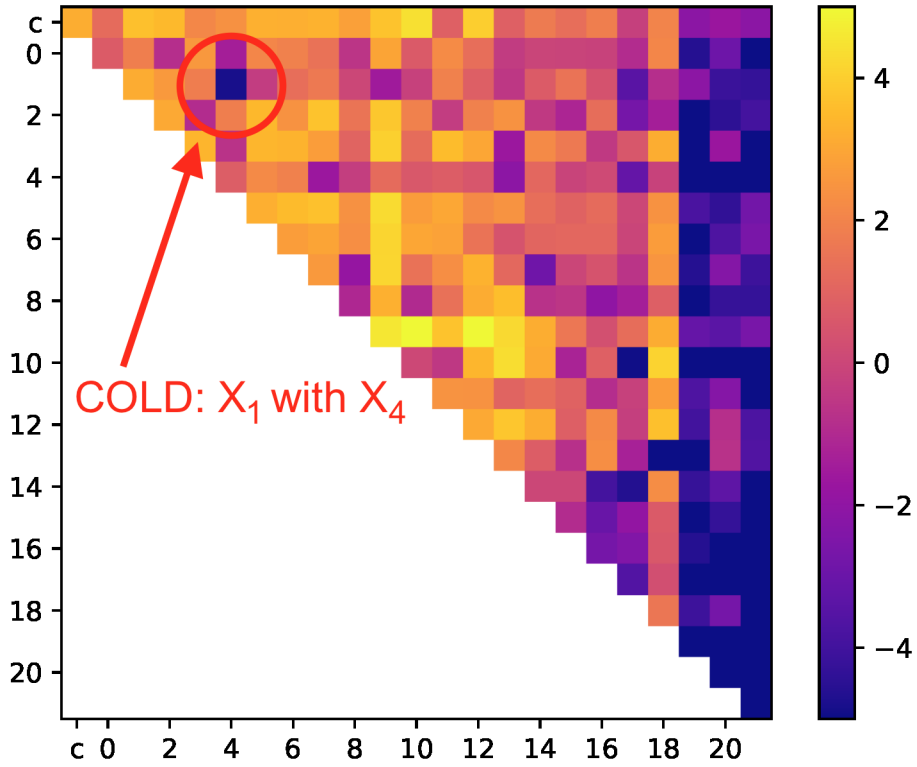
$\Rightarrow$ What matters most in determining N_{FRT} for a 3D facet?



- x_9: total number of 1 simplicies in the triangulation
- x_{10}: total number of 2 simplicies in the triangulation
- x_{12}: number of unique 1 simplicies in the triangulation

Interpretation of 4D Result?

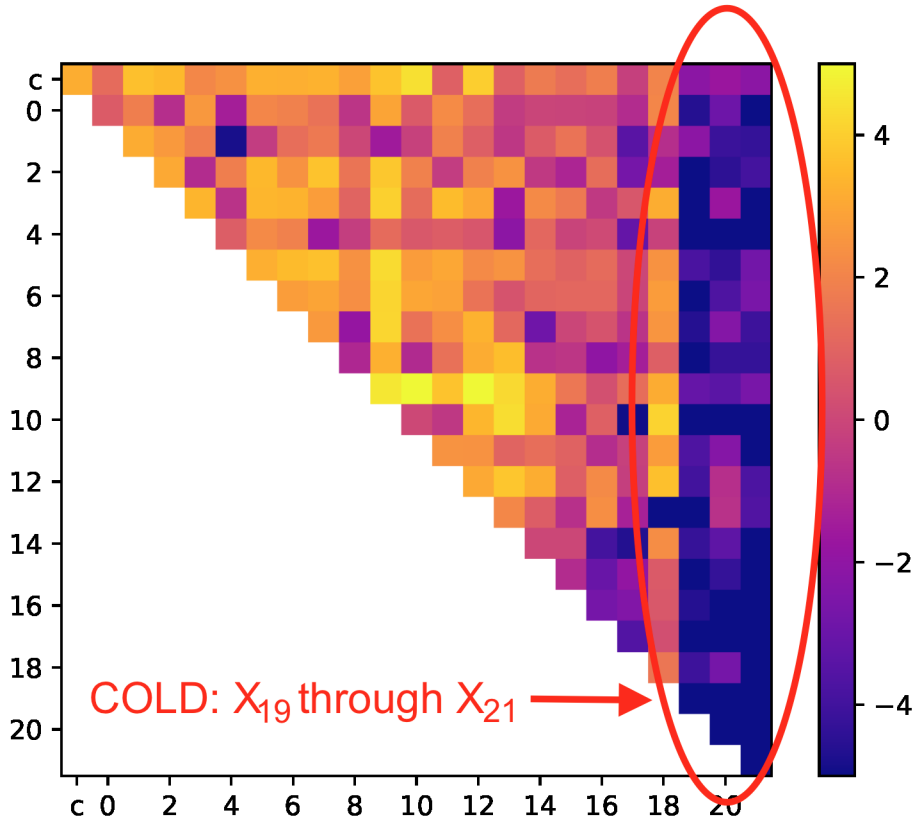
$\Rightarrow$ What matters most in determining N_{FRT} for a 3D facet?



- x_1: number of points on the boundary
- x_4: number of points in the 2-skeleton

Interpretation of 4D Result?

⇒ What matters most in determining N_{FRT} for a 3D facet?



- $x_{19} - x_{21}$: number of 2-simplices shared by 3 or more 3-simplices

Things for (Someone Else) To Do

⇒ The “learned equations” are effectively a regression result

- Could standard regression (i.e. rudimentary machine learning) achieve same result after using `sklearn.preprocessing.PolynomialFeatures`?

- If so, then EQL NN was irrelevant. If not, then how was EQL architecture necessary?

⇒ For the 4D polytopes (3D facets) case, the dimension of the input space can clearly be reduced

- Principal Component Analysis (PCA) on features?

⇒ This clearly over-estimates the number of Calabi-Yau threefolds in KS – but by how much?

String Pheno, Summer 2020

Northeastern University

THANK YOU!