Machine Learning assisted exploration for affine Deligne–Lusztig varieties

Felix Schremmer

Joint work with: Bin Dong, Xuhua He, Pengfei Jin and Qingchao Yu

University of Hong Kong

October 18, 2023

1. What does a mathematician do all day?
2. ML for pure math
3. Our case study

So what do you do all day?

- Think about mathematical problems
- Read
- Compute examples and search for patterns
- Try to prove these patterns

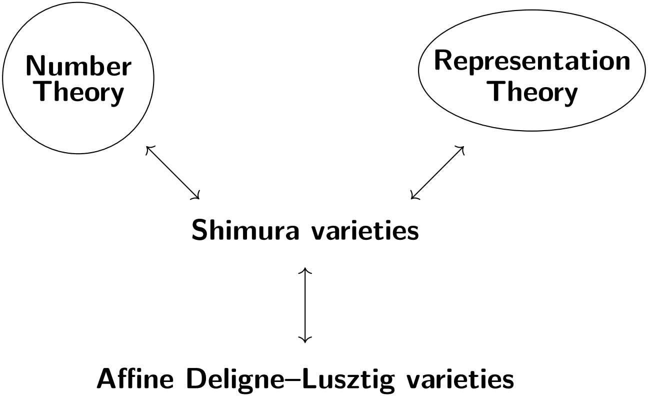
Is this good for anything?

Langlands program (oversimplified)

$$
\begin{tikzcd}
\textbf{Number Theory} \arrow[rr, leftrightarrow, "?", "\text{Langlands Correspondence}"'] & & \textbf{Representation Theory} \\
\text{Problems} \arrow[rr, leftrightarrow] \arrow[d, Rightarrow] & & \text{Problems} \arrow[d, Rightarrow] \\
\text{Methods} \arrow[d, Rightarrow] & & \text{Methods} \arrow[d, Rightarrow] \\
\text{Solutions (Theorems)} \arrow[rr, leftrightarrow] & & \text{Solutions (Theorems)}
\end{tikzcd}
$$

Is this good for anything?

Langlands program (simplified)



So you must be good with numbers, right?

Take:

- A (certain) group. For today, $\mathbf{G} = \mathrm{SL}_5$.
- An element. For today, $b = \mathbf{1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$.
- An element w of the affine Weyl group, e.g.

$$w = \begin{pmatrix} 0 & 0 & t^{-2} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & t & 0 \\ 0 & 0 & 0 & 0 & t^{-3} \\ 0 & t^4 & 0 & 0 & 0 \end{pmatrix}.$$

(t: a formal variable)

Do you think in four dimensions?

To this given $\mathbf{G}$, w, b, we associate an *affine Deligne–Lusztig variety*:

$$X_w(\mathbf{1}) = \{g \in \mathrm{SL}_5 / I \mid g^{-1}\mathbf{1}\sigma(g) \in IwI\}.$$

(σ: Frobenius automorphism. I: Iwahori subgroup)

Key question: Compute $\dim X_w(b)$.

Known: Recursive algorithm (exponential complexity)

Expected: Closed formula.

Step 0: Getting started

We model our problem as a functional relationship

$$\begin{aligned} f : \{w \text{ such that } X_w(\mathbf{1}) \neq \emptyset\} &\to \mathbb{Z}_{\geq 0} \\ w &\mapsto \dim X_w(\mathbf{1}) \end{aligned}$$

Goal: Find a closed formula to evaluate f
(mathematical conjecture)

Step 1: Data generation

- Choose a *computer representation* of domain and target of the function, e.g.

$$\begin{pmatrix} 0 & 0 & t^{-2} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & t & 0 \\ 0 & 0 & 0 & 0 & t^{-3} \\ 0 & t^4 & 0 & 0 & 0 \end{pmatrix} \leftrightarrow (2, 5, 1, 3, 4, \quad 0, 4, -2, 1, -3).$$

  (target is $\mathbb{Z}_{\geq 0}$, needs no further representation)
- Choose a suitable subset of the domain (e.g. 1000 randomly chosen elements)
- Evaluate the function f on these examples

Step 2: Model selection

- Choose a *hypothesis class*, i.e. a family of functions

$$\hat{f}_m, \qquad m \in \mathcal{M}$$

  hoping that one of these can approximate our target function f "well"

  Typical choices: Neural network, linear model, decision tree...

- Choose a *loss function*, which evaluates how good the approximation $\hat{f}_m$ is

  Typical choices: ℓ_1 or ℓ_2 norm with regularization

Step 3: Training

- Split the dataset $\mathbb{D}$ into a training and test part $\mathbb{D} = \mathbb{D}_{\text{train}} \sqcup \mathbb{D}_{\text{test}}$
- Find a model $m \in \mathcal{M}$ such that $\hat{f}_m$ approximates f on $\mathbb{D}_{\text{train}}$ as good as possible (loss function)
- Optimization method depends on chosen hypothesis class
- Avoid overfitting: Compare test error vs. training error

Step 4: Evaluation

- Study the approximation function $\hat{f}_m$:
  - Accuracy on training / test set
  - Accuracy on different parts of the dataset
- Study the model m:
  - Importance/Influence of different input variables
  - Compare with prior subject knowledge

Step 5: Refinement

Do we have a simple, robust approximation $\hat{f}_m$ that models our target function f very well (according to theory&evidence)?

- Yes: New mathematical conjecture found!
- No: Consider all choices made in Steps 1–4 and repeat

Problem and complexity

- Recall: Our target function f computes dimensions of ADLV

$$w = \begin{pmatrix} 0 & 0 & t^{-2} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & t & 0 \\ 0 & 0 & 0 & 0 & t^{-3} \\ 0 & t^4 & 0 & 0 & 0 \end{pmatrix} \mapsto \dim X_w(\mathbf{1}) = 23.$$

- Dataset: 5000 randomly sampled elements w.
- Model: Let's try neural networks!

		Neurons / Layer		
		10	**20**	**40**
Layers	**1**	0.53	0.53	0.52
	2	0.53	0.53	0.51
	3	0.52	0.51	0.51
		Avg. test error		

$\rightsquigarrow$ Linear model is probably fine

A first linear model

- Represent an element w by 12 numbers:
  - Five to signify the positions of the $t^{\bullet}$'s in the matrix
  - Five to signify the t-exponents
  - Two more of Lie-theoretic relevance
- Same dataset as before
- Test error: **0.65**
- Model interpretation: **hard**
- If all t-exponents are pairwise distinct: Error **0.62**. Interpretation **hard**

Better features

- Focus on those w's with pairwise distinct t-exponents.
- Associate *two* permutations to each w: Position of t's in the matrix, and order of t-exponents

$$\begin{pmatrix} 0 & 0 & t^{-2} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & t & 0 \\ 0 & 0 & 0 & 0 & t^{-3} \\ 0 & t^4 & 0 & 0 & 0 \end{pmatrix} \to \begin{array}{c} xy = (2,5,1,3,4) \\ y = (2,4,1,3,5) \end{array}$$

- Represent each permutation x, y by their inversion set (10 numbers) and length (1 number).
- $\rightsquigarrow$ Test error: **0.65**. Model Interpretability: **Better!**

Model coefficients

Inversions for x	0.12, -0.04, -0.05, -0.24, 0.14, ...
Inversions for y	similar picture
$\ell(x), \ell(y)$	0.1, 0.1
t-coefficients	0.13, -0.09, -0.02, 0.08, -0.10
Length of w	0.52

- Leading term: $\frac{1}{2}\ell(w)$
- Besides, no significant contribution of $\ell(w)$, or t-coefficients
- Contribution of x, y needs further investigation

Restricting the dataset further

- Consider only those w's with $x = (1, 2, 3, 4, 5)$. E.g.

$$w = \begin{pmatrix} 0 & 0 & t^5 & 0 & 0 \\ t^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & t^{-3} \\ 0 & t^{-4} & 0 & 0 & 0 \end{pmatrix}$$

- Generate 5000 of those.
- Input features: Everything related to y.
- Target function: $g(w) = \dim X_w(\mathbf{1}) - \frac{1}{2}\ell(w)$.
- Use ℓ_2-loss function: Avg. error: **0.30**.
  Model interpretability: **Tricky**.
- Use ℓ_1-loss function: Avg. error **0.18**.
  Model interpretability: **Trivial**. Explicitly, $\hat{g} = \frac{1}{2}\ell(y)$.

Generalization

Return to the second data set. Our target function is

$$g : w \mapsto \dim X_w(\mathbf{1}) - \frac{1}{2}\ell(w).$$

Approximation should simplify to $\frac{1}{2}\ell(y)$ whenever $x = (1, 2, 3, 4, 5)$.

Feature	$\ell(x)$	$\ell(y)$	$\ell(xy)$	$\ell(yx)$	$\ell(y * x)$	$\ell(y \triangleleft x)$
Coeff.	0.02	-0.05	-0.02	0.46	0.04	0.04

We got a winner! $\hat{g} \approx \frac{1}{2}\ell(yx)$.

Story time

Virtual dimension $d_w(\mathbf{1}) = \frac{1}{2}\left[\ell(w) + \ell(yx)\right]$ approximates $\dim X_w(\mathbf{1})$.

- Discovery of virtual dimension formula was a great breakthrough 10–20 years ago
- Our ML method can find the formula (today: the most tricky part)
- Analyse the data more carefully $\rightsquigarrow$ obtain precise mathematical conjectures
- He 2014: Dimension $\leq$ virtual dimension. Equality holds for $b = \mathbf{1}$ and "most" w.
- He 2022: Dimension $=$ virtual dimension for "most" (w, b)
- Our paper: ML suggests that (virtual dim. *minus* dim.) is bounded. We then give a proof!

The bigger picture

- AI4MATH works! We find old and new conjectures very fast (also works for more tricky patterns related to ADLV that require neural networks)
- Even "atypical" problems can be solved, by revising the full pipeline
- Interdisciplinary collaboration and modern technology lead us to a new way of researching pure mathematics