Peking Mathematical Journal
https://doi.org/10.1007/s42543-024-00086-8

ORIGINAL ARTICLE

Machine Learning Assisted Exploration for Affine Deligne–Lusztig Varieties

Bin Dong[1,2] · Xuhua He[3] · Pengfei Jin[1] · Felix Schremmer[3] · Qingchao Yu[1]

Received: 30 August 2023 / Revised: 19 January 2024 / Accepted: 25 February 2024
© The Author(s) 2024

Abstract

This paper presents a novel, interdisciplinary study that leverages a Machine Learning (ML) assisted framework to explore the geometry of affine Deligne–Lusztig varieties (ADLV). The primary objective is to investigate the non-emptiness pattern, dimension, and enumeration of irreducible components of ADLV. Our proposed framework demonstrates a recursive pipeline of data generation, model training, pattern analysis, and human examination, presenting an intricate interplay between ML and pure mathematical research. Notably, our data-generation process is nuanced, emphasizing the selection of meaningful subsets and appropriate feature sets. We demonstrate that this framework has a potential to accelerate pure mathematical research, leading to the discovery of new conjectures and promising research directions that could otherwise take significant time to uncover. We rediscover the virtual dimension formula and provide a full mathematical proof of a newly identified problem concerning a certain lower bound of dimension. Furthermore, we extend an open invitation to the readers by providing the source code for computing ADLV and the ML models, promoting further

✉ Xuhua He
xuhuahe@hku.hk

Bin Dong
dongbin@math.pku.edu.cn

Pengfei Jin
jinpf@pku.edu.cn

Felix Schremmer
schremmer@hku.hk

Qingchao Yu
yuqingchao@bicmr.pku.edu.cn

1 Beijing International Center for Mathematical Research, Peking University, Beijing 100871, China

2 Center for Machine Learning Research, Peking University, Beijing 100871, China

3 Department of Mathematics and New Cornerstone Science Laboratory, The University of Hong Kong, Pokfulam, Hong Kong, China

Published online: 15 May 2024

explorations. This paper concludes by sharing valuable experiences and highlighting lessons learned from this collaboration.

Keywords Affine Deligne–Lusztig varieties · Affine Weyl groups · Loop groups · AI-assisted math research

1 Introduction

1.1 A Brief Overview of Affine Deligne–Lusztig Varieties

The concept of Affine Deligne–Lusztig Varieties (ADLV) was first introduced by Rapoport [51]. These varieties serve as a group-theoretic model for the reduction of Shimura varieties and shtukas with parahoric level structure and play a vital role in arithmetic geometry and the Langlands program. Key problems associated with ADLV include:

- Non-emptiness pattern;
- Dimension;
- Enumeration of irreducible components.

Over the past 2 decades, the study of ADLV has been a vibrant research topic. Significant progress has been made in understanding fundamental problems, and important applications to number theory and the Langlands program have been discovered. The non-emptiness pattern and dimension of ADLV in the affine Grassmannian are now fully understood, and in most cases, they are also known for ADLV in the affine flag variety. The enumeration of irreducible components has been solved in the affine Grassmannian case. For more in-depth information, readers are referred to the survey article [27].

Despite these advancements, a comprehensive solution to the problems presented by ADLV remains a challenge, primarily due to the difficulty in finding explicit patterns.

In this paper, our focus is on ADLV $X_w(b)$ in the affine flag variety (associated with the Iwahori level structure). Information for other parahoric level structures can be obtained from the Iwahori level structure via the natural projection map. The ADLV $X_w(b)$ depends on two parameters: the element w in the Iwahori–Weyl group $\tilde{W}$ of a loop group $\breve{G}$, and the Frobenius-twisted conjugacy class $[b]$ of $\breve{G}$. We consider the map from the pair (w, b) to the dimension and enumeration of the top-dimensional irreducible components of $X_w(b)$ (refer to Sect. 2.1 for a precise statement of the problem). The image of a given pair (w, b) can be calculated using the intricate inductive algorithm established in [25]. Yet, the goal is to derive more explicit formulas/information for this map. Such explicit formulas and information are particularly intriguing because of their broad applications to arithmetic geometry and number theory.

1.2 Machine Learning Assisting Pure Mathematics Research

In recent years, machine learning (ML), particularly deep learning, has exerted a profound impact on diverse scientific and engineering disciplines, bringing about sub-

stantial changes in the way we conduct research. The success of ML primarily hinges on the development of the designing and training of deep neural networks, which approximate complex, high-dimensional mappings with desirable accuracy, and rapid evaluations. As a result, ML has become a leading force in different areas of artificial intelligence (AI), such as natural language processing (large language models such as ChatGPT [68]), computer vision (e.g., NeRF [44], diffusion models [32]) and games (e.g., mastering the game of Go [57], Poker [6], and StarCraft II [64]). Moreover, the excellent approximation capabilities of deep neural networks have helped discover new patterns or principles within large, multi-dimensional data sets. This has significantly expanded ML's role in natural science, contributing to the rise of a new field known as "AI for Science". Successful examples include the work of AlphaFold [34], molecular dynamics simulations [11, 33, 67], chemical discovery [43, 59, 65], system identification [8, 40, 41, 50], controllable nuclear fusion [14, 35], etc.

More recently, Geordie Williamson and DeepMind used ML to assist in research-level explorations of pure mathematics [13]. They present an ML-based framework that augments mathematicians' intuition, aiding in the discovery and understanding of complex mathematical relationships. This approach identifies potential correlations between two mathematical entities by deriving a function approximating the relationship and helping mathematicians analyze it.

The framework validates possible patterns in mathematical objects using supervised learning, and helps understand these patterns using attribution techniques. In the supervised learning stage, a hypothesis about a connection between two entities is proposed, a dataset is generated, and a function is trained to predict one entity from the other. The role of ML here is in learning a wide variety of potential non-linear functions given sufficient data. Attribution techniques are then used to understand the trained function and propose a potential relationship. One such technique, gradient saliency, calculates the derivative of function outputs with respect to inputs, helping to identify the most relevant problem aspects. This process may be iterative until a feasible conjecture is found.

In essence, this ML-guided framework enables a quick verification of the potential worthiness of an intuition about a relationship and, if validated, suggests how they may be related. This framework has already proven its utility by [13] in achieving significant results, such as uncovering the first relationships between algebraic and geometric invariants in knot theory and conjecturing a resolution to the combinatorial invariance conjecture for symmetric groups.

1.3 Our Objective

In this study, our objective is to develop an ML-assisted framework to guide the study of fundamental problems related to ADLV, specifically the non-emptiness pattern, the dimension and enumeration of irreducible components. As illustrated in Figs. 1 and 2, our framework showcases a recursive pipeline of data generation, model training, pattern analysis, and human examination. Despite similarities with the framework in the aforementioned study [13], several crucial differences exist. Our data-generation process is more intricate, particularly regarding the selection of a meaningful subset of

pairs (w, b) and an appropriate set of features. Additionally, after fitting a functional relationship between the feature set and the property of interest (e.g. the non-emptiness of $X_w(b)$), the patterns revealed by salience analysis may be more challenging for mathematicians to interpret due to the problem's complexity. Nevertheless, we found that this interaction between ML and human mathematicians significantly accelerates pure mathematical research, enabling us to identify new conjectures and promising research directions that could otherwise take years for mathematicians to discover by themselves.

We provide the source code for computing geometric invariants of ADLV and machine learning models to invite interested readers to delve into this problem, reproduce our experiments, and refine our approach by studying different datasets and feature sets. In Sect. 5, we seek a linear approximation for the dimension of ADLV, leading us to rediscover the virtual dimension formula. Originally, the development of the virtual dimension formula required several years of intense research by many mathematicians and constitutes a major milestone in the field. In Sect. 6, we conduct sensitivity analysis for the problems introduced in Sect. 1.1. We identify several important features, affirming some of the latest research results in the field and suggesting potential next steps. Motivated by the experiments in Sects. 5 and 6, we discover a new problem concerning a certain lower bound of the dimension, which has not been studied in the literature before. In Sect. 7, we provide a full mathematical proof of this lower bound. We conclude this paper by suggesting future work in Sect. 8, sharing some experiences, and highlighting lessons learned from the collaboration.

2 Preliminaries

2.1 Definition and Properties of Affine Deligne–Lusztig Varieties

In this subsection, we provide a brief overview of affine Deligne–Lusztig varieties. Unless otherwise stated (i.e., Sects. 4.1 and 7), we focus on the case of the special linear group SL_n, which is also referred to as the type A_{n-1}. By focusing exclusively on this case, we may reduce the technical details in this exposition. However, the more important reason for this specialization is that it allows us to perform computer experiments within a reasonably narrow scope, where all the beauties and pathologies of the general case are still present.

Let q be a prime power and $\mathbb{F}_q$ be the finite field with q elements. We define $F = \mathbb{F}_q(\!(t)\!)$ to be the field of formal Laurent series over $\mathbb{F}_q$. This means that elements in $a \in F$ are formal power series

$$a = \sum_{i \in \mathbb{Z}} a_i t^i$$

with coefficients $a_i \in \mathbb{F}_q$, such that $a_i = 0$ for almost all $i < 0$. There is no notion of convergence involved, but the definition of addition and multiplication in F mimics the behavior of absolutely convergent power series over real or complex numbers.

Pick once and for all an algebraic closure $\overline{\mathbb{F}}_q$ and define $\breve{F} = \overline{\mathbb{F}}_q((t))$ to be the field of formal Laurent series over $\overline{\mathbb{F}}_q$. The Galois group of the field extension $\breve{F}/F$ is generated by the Frobenius σ, which can be evaluated for elements in $\breve{F}$ as

$$\sigma\left(\sum_{i\in\mathbb{Z}} a_i t^i\right) = \sum_{i\in\mathbb{Z}} a_i^q t^i \in \breve{F}.$$

Finally, we write $\mathcal{O}_{\breve{F}} = \overline{\mathbb{F}}_q[\![t]\!]$ for the ring of all formal power series, i.e., elements $a \in \breve{F}$ with $a_i = 0$ for all $i < 0$.

Throughout this paper till Sect. 7, we will focus on the algebraic group scheme SL_n, considered as a scheme over $\mathcal{O}_F = \mathbb{F}_q[\![t]\!]$. We get an induced map $\sigma : \mathrm{SL}_n(\breve{F}) \to \mathrm{SL}_n(\breve{F})$, given by applying the above Frobenius $\sigma : \breve{F} \to \breve{F}$ to the entries of each $n \times n$-matrix in $\mathrm{SL}_n(\breve{F})$.

Two elements $b, c \in \mathrm{SL}_n(\breve{F})$ are called *σ-conjugate* if there exists some $g \in \mathrm{SL}_n(\breve{F})$ with

$$b = g^{-1} c \sigma(g).$$

One checks that this is an equivalence relation, similar to ordinary conjugacy. We denote the σ-conjugacy class of $b \in \mathrm{SL}_n(\breve{F})$ by $[b]$, and the set of σ-conjugacy classes of $\mathrm{SL}_n(\breve{F})$ by $B(\mathrm{SL}_n)$. The σ-conjugacy class of $b \in \mathrm{SL}_n(\breve{F})$ is uniquely determined by an invariant called the *Newton point* of b, denoted by $\nu_b \in \mathbb{Q}^n$ [37].

Call a vector $(\nu_1, \ldots, \nu_n) \in \mathbb{Q}^n$ *dominant* if $\nu_1 \geq \cdots \geq \nu_n$. Then, the Newton point of each $b \in \mathrm{SL}_n(\breve{F})$ is such a dominant vector. We have an action of the symmetric group S_n on $\mathbb{Q}^n$ by permutation of coordinates. One checks that each orbit under this action contains precisely one dominant vector. If $b \in \mathrm{SL}_n(\breve{F})$ is a diagonal matrix of the form $b = \mathrm{diag}(\pm t^{b_1}, \ldots, \pm t^{b_n})$ with $b_1, \ldots, b_n \in \mathbb{Z}$, then the Newton point of b is the unique dominant element in the S_n-orbit of $(b_1, \ldots, b_n) \in \mathbb{Z}^n \subseteq \mathbb{Q}^n$.

One calls $W_0 := S_n$ the *(finite) Weyl group* of SL_n. The *affine Weyl group* is given by the semidirect product

$$W_a = \tilde{W} := S_n \ltimes \{(\mu_1, \ldots, \mu_n) \in \mathbb{Z}^n \mid \mu_1 + \cdots + \mu_n = 0\}.$$

We write elements $w \in W_a$ also as $w = t^\lambda z$, where $z \in S_n$ and $\lambda = (\lambda_1, \ldots, \lambda_n) \in \mathbb{Z}^n$ with $\lambda_1 + \cdots + \lambda_n = 0$. The symbol t is a formal variable reminding us of the uniformizer $t \in F$.

We choose for each permutation $x \in S_n$ a representative $\dot{x} \in \mathrm{SL}_n(\breve{F})$, so that

$$\dot{x}_{i,j} = \begin{cases} \pm 1, & i = x(j), \\ 0, & i \neq x(j). \end{cases}$$

Concretely, the element $\dot{x}$ may be chosen to be the permutation matrix of x if x is an even permutation, and the permutation matrix of x with one sign flipped if x is

an odd permutation. For $w = t^{\lambda} z \in W_a$, we write $\dot{w} \in \mathrm{SL}_n(\breve{F})$ for the element $\dot{w} = \mathrm{diag}(t^{\lambda_1}, \ldots, t^{\lambda_n})\dot{z}$.

Let $\breve{I} \subset \mathrm{SL}_n(\breve{F})$ be the *Iwahori* subgroup

$$\begin{aligned}\breve{I} &= \mathrm{SL}_n(\breve{F}) \cap \left\{ \begin{pmatrix} \mathcal{O}_{\breve{F}} & \mathcal{O}_{\breve{F}} & \ldots & \mathcal{O}_{\breve{F}} \\ t\mathcal{O}_{\breve{F}} & \mathcal{O}_{\breve{F}} & \ddots & \vdots \\ \vdots & & \ddots & \mathcal{O}_{\breve{F}} \\ t\mathcal{O}_{\breve{F}} & \ldots & t\mathcal{O}_{\breve{F}} & \mathcal{O}_{\breve{F}} \end{pmatrix} \right\} \\ &= \left\{ a \in \mathrm{SL}_n(\breve{F}) \;\middle|\; \begin{array}{l} \forall i > j : \; a_{i,j} \in t\mathcal{O}_{\breve{F}} \\ \forall i \leq j : \; a_{i,j} \in \mathcal{O}_{\breve{F}} \end{array} \right\} \\ &= \{ a \in \mathrm{SL}_n(\mathcal{O}_{\breve{F}}) \mid a \text{ is upper triangular modulo } t \}.\end{aligned}$$

Then, each element $g \in \mathrm{SL}_n(\breve{F})$ has the form $g = i_1 \dot{w} i_2$ for a uniquely determined element $w \in W_a$ and non-unique $i_1, i_2 \in \breve{I}$ (see [7]). Such a decomposition can be computed using an adaption of the Gauss algorithm.

We have seen two decompositions of the set $\mathrm{SL}_n(\breve{F})$, namely, one into σ-conjugacy classes $B(\mathrm{SL}_n)$ and another one into Iwahori double cosets $\breve{I}\dot{w}\breve{I}$ for $w \in W_a$. We also write $\breve{I}w\breve{I} := \breve{I}\dot{w}\breve{I}$, since the double coset is independent of the choice of $\dot{w}$.

The right coset space $Fl = \mathrm{SL}_n(\breve{F})/\breve{I}$ is called the *affine flag variety*. It is an ind-scheme over $\overline{\mathbb{F}}_q$, which behaves similarly to finite-dimensional varieties over that field. For $w \in W_a$ and $b \in \mathrm{SL}_n(\breve{F})$, we define the *affine Deligne–Lusztig variety* to be the subvariety

$$X_w(b) := \{ g\breve{I} \in Fl \mid g^{-1} b \sigma(g) \in \breve{I} w \breve{I} \}.$$

Again, $X_w(b)$ is not actually a variety, but still a scheme over $\overline{\mathbb{F}}_q$. Moreover, each irreducible component of $X_w(b)$ is an actual finite-dimensional variety over $\overline{\mathbb{F}}_q$. If b is σ-conjugate to $c \in \mathrm{SL}_n(\breve{F})$, say $c = h^{-1} b \sigma(h)$ for $h \in \mathrm{SL}_n(\breve{F})$, then

$$X_w(b) \to X_w(c), \quad g\breve{I} \mapsto h^{-1} g \breve{I}$$

is an isomorphism. Thus, we may associate the isomorphism type of $X_w(b)$ to the pair $(w, [b]) \in W_a \times B(\mathrm{SL}_n)$. This is our main object of interest.

We see that the σ-centralizer of b, denoted by

$$\mathbf{J}_b(F) = \{ g \in \mathrm{SL}_n(\breve{F}) \mid g^{-1} b \sigma(g) = b \},$$

acts on $X_w(b)$ by left multiplication. Up to that action, there are only finitely many irreducible components in $X_w(b)$. Since each such irreducible component is a finite-dimensional variety over $\overline{\mathbb{F}}_q$, the entire affine Deligne–Lusztig variety $X_w(b)$ is always finite-dimensional.

The main questions regarding the geometry of affine Deligne–Lusztig varieties are the following three:

- Non-emptiness pattern: given $(w, [b])$, determine whether $X_w(b) \neq \emptyset$;
- Dimension: given $(w, [b])$ such that $X_w(b) \neq \emptyset$, calculate the dimension of $X_w(b)$;
- Irreducible components: given $(w, [b])$ such that $X_w(b) \neq \emptyset$, calculate the number of $\mathbf{J}_b(F)$-orbits of top dimensional irreducible components, i.e., the cardinality of $\mathbf{J}_b(F)\backslash\Sigma^{\mathrm{top}}(X_w(b))$. Here, $\Sigma^{\mathrm{top}}(X)$ denotes the set of top-dimensional irreducible components of X.

2.2 Important Invariants

To address these three questions, one explores the relationship between the affine Weyl group W_a and the set $B(\mathrm{SL}_n)$ parametrized by the Newton points. This relationship is essentially combinatorial in nature, which allows us to compute the answers to the three above questions for any given pair (w, ν_b). We summarize some key combinatorial invariants that have proven valuable in previous works.

The group $W_0 = S_n$ is known to be a *Coxeter group* with respect to the generators $\mathbb{S} = \{s_1, \ldots, s_{n-1}\}$. Here, s_i is the simple reflection interchanging i with $i+1$ and leaving everything else fixed. Each element $x \in W_0$ is a product of the s_i, and the shortest length of such an expression is called the *length* of x. There is a unique element of maximal length in W_0, denoted by w_0. It is the permutation $w_0(i) = n+1-i$ for $i \in \{1, \ldots, n\}$, and its length is $\ell(w_0) = n(n-1)/2$.

There is a different way to compute the length of an element $x \in W_0$. Denote the set of *roots* of SL_n as

$$\Phi = \{e_i - e_j \in \mathbb{Q}^n \mid i, j \in \{1, \ldots, n\} \text{ and } i \neq j\}.$$

The root $\alpha_{i,j} := e_i - e_j$ is called *positive* if $i < j$, and negative otherwise. We write $\delta(\alpha) = 1$ if α is a negative root and 0 if α is positive. Observe that the S_n-action on $\mathbb{Q}^n$ preserves the set of roots. Then, the length of $x \in W_0$ equals the number of positive roots $\alpha_{i,j}$, such that $x\alpha_{i,j}$ is a negative root. As formula

$$\ell(x) = \sum_{i<j} \delta(x\alpha_{i,j}).$$

The group $W_a = S_n \ltimes \mathbb{Z}^n$ is also known to be a Coxeter group with respect to $\mathbb{S}_a$. The set of simple affine reflections $\mathbb{S}_a$ is given by $\{s_0\} \cup \mathbb{S}$, where

$$s_0 = (1\ n)t^{(-1,0,\ldots,0,1)} \in W_a.$$

One defines the length of an element $w \in W_a$ as above, given by the smallest representation using these simple affine reflections. There is an alternative way to compute the length of $w = t^\lambda z \in W_a$: We saw above that there is some $y \in W_0$ with $yz^{-1}\lambda \in \mathbb{Z}^n$ being dominant. Among all those elements in $y \in W_0$, there is a unique one with $\ell(y)$ being minimal. For this specific $y \in W_0$, we write $x := zy^{-1} \in W_0$ and $\mu := yz^{-1}\lambda \in \mathbb{Z}^n$, so that $w = xt^\mu y$. Then

$$\ell(xt^\mu y) = \langle \mu, 2\rho \rangle + \ell(x) - \ell(y).$$

Here, $\langle \cdot, \cdot \rangle$ is the standard Euclidean inner product on $\mathbb{Q}^n$, and $2\rho = (n-1, n-3, \ldots, 3-n, 1-n) \in \mathbb{Q}^n$ is the sum of positive roots. Whenever we write w in the form $xt^\mu y$, we always assume that x, μ, y have been chosen as above.

For $c \in \mathbb{Z}_{\geq 0}$, we call the element $w = xt^\mu y$ to be *c-regular* if $\langle \mu, \alpha_{i,j} \rangle \geq c$ for all positive roots $\alpha_{i,j}$. The decomposition of w into x, t^μ, and y has the most desirable properties whenever w is 2-regular, but the above length formula is always true even when such a regularity condition is not satisfied.

To each $w \in W_a$, one may associate the σ-conjugacy class $[\dot{w}] \in B(\mathrm{SL}_n)$. Its Newton point can be computed as follows: If w has the form $w = t^\lambda$ for some $\lambda \in X_*(T)$, then $\dot{w} = \mathrm{diag}(t^{\lambda_1}, \ldots, t^{\lambda_n})$ and we saw above that the Newton point of $\dot{w}$ is the unique dominant element in the S_n-orbit of λ. For general $w \in W_a$, one may find an integer $m \geq 1$, such that w^m is of the above form, and then, the Newton point of $\dot{w}$ is given by $\nu_w = \nu_{w^m}/m \in \mathbb{Q}^n$.

It turns out that each σ-conjugacy class $[b] \in B(\mathrm{SL}_n)$ contains the representative $\dot{w} \in \mathrm{SL}_n(\breve{F})$ of some $w \in W_a$, cf. [25, Theorem 3.7]. Hence, the above method yields all Newton points of all σ-conjugacy classes. If $[b] \in B(\mathrm{SL}_n)$ has Newton point $\nu_b = (\nu_1, \ldots, \nu_n) \in \mathbb{Q}^n$, we define *the best integral approximation* $\lfloor \nu_b \rfloor \in \mathbb{Z}^n$ to be the vector $(\mu_1, \ldots, \mu_n) \in \mathbb{Z}^n$, such that for all i, we have

$$\lfloor \nu_1 + \cdots + \nu_i \rfloor = \mu_1 + \cdots + \mu_i \in \mathbb{Z}.$$

Equivalently, $\lfloor \nu_b \rfloor$ is the unique vector in $\mathbb{Z}^n$ that can be written in the form

$$\lfloor \nu_b \rfloor = \nu_b - c_1\alpha_{1,2} - \cdots - c_{n-1}\alpha_{n-1,n},$$

such that $0 \leq c_i < 1$ for $i = 1, \ldots, n-1$. The *defect* of $[b] \in B(\mathrm{SL}_n)$ can then be defined as $\mathrm{def}(b) = \langle \nu_b - \lfloor \nu_b \rfloor, 2\rho \rangle$. Alternatively, it can be computed as the number of non-integral coordinates of ν, i.e., the number of indices $i \in \{1, \ldots, n\}$, such that $\nu_i \in \mathbb{Q} \backslash \mathbb{Z}$.

2.3 Computing the Geometry of Affine Deligne–Lusztig Varieties

In this section, we present a combinatorial algorithm that efficiently computes the answers to the above three main questions for a given pair (w, ν_b), where $w \in W_a$ and $\nu_b \in \mathbb{Q}^n$. Although the subsequent sections of this article do not depend on the specific algorithm used, we want to provide a detailed explanation of its workings.

It is worth noting that the algorithm is somewhat complex and non-deterministic, which accounts for why the three main questions are still considered open. While the algorithm yields a computational solution to each of the three problems, one may still seek a more straightforward and satisfactory characterization. Nonetheless, the algorithm allows for effective computation of the desired results, which is crucial for our practical applications.

Let $w \in W_a$. We explain an algorithm that computes, in finite time, the set

$$\{\nu_b \mid [b] \in B(\mathrm{SL}_n) \text{ and } X_w(b) \neq \emptyset\} \subseteq \mathbb{Q}^n$$

(which hence must be finite). For each occurring Newton point, the corresponding affine Deligne–Lusztig variety is uniquely determined up to isomorphism, and our algorithm computes the dimension of this variety and the number of $\mathbf{J}_b(F)$-orbits of its top-dimensional irreducible components.

For a simple affine reflection $s \in \mathbb{S}_a$, we say that $sws \in W_a$ is a *cyclic shift* of w if $\ell(sws) \leq \ell(w)$. Under this condition, $\ell(sws)$ can either be equal to $\ell(w)$ or $\ell(w)-2$.

In the first case, i.e., $\ell(sws) = \ell(w)$, the affine Deligne–Lusztig varieties $X_w(b)$ and $X_{sws}(b)$ are always isomorphic for all $[b] \in B(\mathrm{SL}_n)$. Therefore, to compute the above data for w, we may freely pass between w and sws.

In the second case, $\ell(sws) = \ell(w)-2$, each affine Deligne–Lusztig variety $X_w(b)$ splits into two parts, so $X_w(b) = U \sqcup V$ is the disjoint union of two subsets, with U being open in $X_w(b)$ and V closed. Then, there are surjective maps with irreducible one-dimensional fibers

$$U \twoheadrightarrow X_{ws}(b), \qquad V \twoheadrightarrow X_{sws}(b).$$

Hence, $U \neq \emptyset$ if and only if $X_{ws}(b) \neq \emptyset$. In this case, $\dim U = \dim X_{ws}(b) + 1$. The set U is $\mathbf{J}_b(F)$-invariant, and the number of $\mathbf{J}_b(F)$-orbits of top dimensional irreducible components agrees for U and $X_{ws}(b)$. The same story happens for V and $X_{sws}(b)$. Once we know this geometric information, the corresponding data for $X_w(b) = U \sqcup V$ are easily computed: If $U = V = \emptyset$, then $X_w(b) = \emptyset$. If precisely one of the subsets U or V is empty and the other one is non-empty, then $X_w(b)$ agrees with the unique non-empty subset, and all geometric invariants are known. Finally, if $U \neq \emptyset \neq V$, we have $\dim X_w(b) = \max(\dim U, \dim V)$ and

$$\#\mathbf{J}_b(F)\backslash\Sigma^{\mathrm{top}}(X_w(b)) = \begin{cases} \#\mathbf{J}_b(F)\backslash\Sigma^{\mathrm{top}}(U), & \dim U > \dim V, \\ \#\mathbf{J}_b(F)\backslash\Sigma^{\mathrm{top}}(V), & \dim V > \dim U, \\ \#\mathbf{J}_b(F)\backslash\Sigma^{\mathrm{top}}(U) + \#\mathbf{J}_b(F)\backslash\Sigma^{\mathrm{top}}(V), & \dim U = \dim V. \end{cases}$$

Moreover, if $\ell(sws) = \ell(w)-2$, then $\ell(ws) = \ell(w)-1$. Therefore, in this case, we have reduced the geometric questions for w and arbitrary $[b]$ to the same questions of the two elements sws, ws of smaller length.

The first part of the algorithm iteratively enumerates the *cyclic shift class* of w, i.e., the set of all elements in W_a reachable by iterated cyclic shifts $w \to s_1 w s_1 \to s_2 s_1 w s_1 s_2 \to \cdots$. Each element in this cyclic shift class has length $\leq \ell(w)$, so that the cyclic shift class is a finite set.

We traverse this cyclic shift class, until we either exhaust the full set or reach some element w' in the cyclic shift class and some $s' \in \mathbb{S}_a$ with $\ell(w) = \ell(w') = \ell(s'w's')+2$. In the latter case, we know $X_w(b) \cong X_{w'}(b)$ and the geometric properties of $X_{w'}(b)$ can be reduced to recursively calling our algorithm for the two smaller elements $s'w'$ and $s'w's'$. Since the length drops by at least one, such a recursive call can only happen a finite number of times.

The second part of the algorithm handles the case where the entire cyclic shift class is enumerated without reaching a length-reducing cyclic shift. In this case, we not only

know that w has a minimal length in this cyclic shift class but even that it must have a minimal length in its conjugacy class in W_a.

In this case, we can explicitly state that

$$\{\nu_b \mid X_w(b) \neq \emptyset\} = \{\nu_w\}.$$

For $\nu_b = \nu_w$, we know $\dim X_w(b) = \ell(w) - \langle \nu_b, 2\rho \rangle$ and that the number of $\mathbf{J}_b(F)$-orbits of top dimensional irreducible components is equal to 1.

This algorithm is guaranteed to terminate in finite time with the correct result. We note that the algorithm itself is non-deterministic, since there is no canonical way to traverse the cyclic shift class of an element $w \in W_a$. While the above method allows us to compute the dimension of $X_w(b)$ for arbitrary w, $[b]$, it is far from being a closed formula. In many cases, however, one may expect that such a closed formula can be found.

The first part of this algorithm is due to Görtz–He [18, Corollary 2.5.3], the second part is due to He and Nie [29, Theorem A], [25, Theorem 4.8].

2.4 Machine Learning-Assisted Formula Exploration

As mentioned earlier, the algorithm used to compute the geometry of affine Deligne–Lusztig varieties is somewhat complex, non-deterministic, and implicit. However, practical research often requires an explicit expression or pattern. Machine learning, particularly deep neural network models, excels at fitting and analyzing high-dimensional mappings. Thus, we are considering utilizing machine learning to aid us in exploring formulas.

The machine learning-assisted formula exploration process is broken down into several steps, as illustrated in Fig. 1. First, a suitable problem $Y = f(X)$ needs to be selected, where f is a mapping from known features X to the variable Y of interest, and a dataset $\{X_i, Y_i\}_{i=1}^N$ is generated using a known algorithm. Second, a machine learning model $\hat{f}_\theta$ is chosen, and an optimal approximation $\hat{f}_{\theta^*}$ of f is attained through optimization on the generated dataset. Third, hints about the patterns of f are obtained by analyzing the explicit expression of $\hat{f}_{\theta^*}$. Finally, the obtained pattern can inspire us to introduce new features X or consider functions f with different domains. From

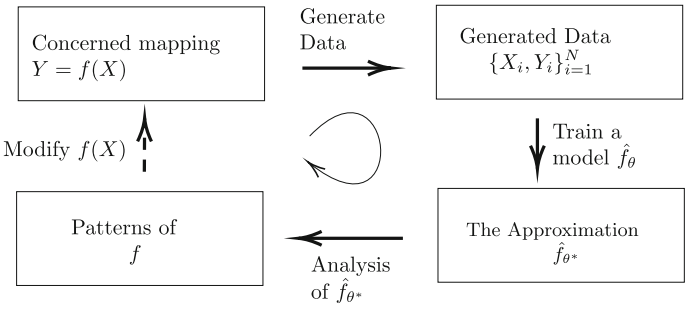


Fig. 1 Pipeline of machine learning-assisted formula exploration

this, we can modify $f(X)$, return to the first step, and continue this cycle. Each step will be elaborated on in detail in the following.

Regarding f, our primary concern is mapping form $(w, [b])$ to the geometry of affine Deligne–Lusztig varieties $X_w(b)$, including dimension, whether $X_w(b) \neq \emptyset$ and number of irreducible components. However, due to the inherent complexity of this mapping, it may be necessary to use the computed features of w and b as X, such as $\ell(w)$ and defect(b).

The selection of a suitable machine learning model $\hat{f}_\theta$ requires careful consideration of the inherent complexity of f. Generally speaking, complex models have greater expressive power, but they may be more difficult to optimize and may require larger datasets. In this article, we focus on several commonly used models, including linear models [1, 48], Support Vector Machine (SVM) [9, 12], and neural networks [52, 54]. The specific formulation of the models will be described in detail in the subsequent section.

Given a dataset $\mathcal{D} = \{X_i, Y_i\}_{i=1}^N$, and the model $\hat{f}_\theta$, the next step is to identify the optimal value for θ that minimizes the distance between $\hat{f}_\theta$ and the target function f (known as training) and evaluate the performance of $\hat{f}_\theta$ (known as testing). Typically, the dataset $\mathcal{D}$ is partitioned into two distinct subsets: the training set $\mathcal{D}_{tr}$ and the testing set $\mathcal{D}_{te}$. The former is used to obtain the optimal θ^*, while the latter is used to evaluate f_{θ^*}.

The training process seeks to minimize the following loss function:

$$\theta^* = \arg\min_\theta \sum_{(X_i, Y_i) \in \mathcal{D}tr} \mathcal{L}(Y_i, \hat{f}_\theta(X_i)) + \lambda \mathcal{R}(\theta),$$

where $\mathcal{L}$ is a distance metric function, such as cross-entropy for classification problems or L_2 distance for regression problems. The regularization term for the parameters, $\mathcal{R}(\theta)$, depends on prior knowledge about the parameters and is typically expressed using the L_1 and L_2 norm to ensure simplicity or sparsity of the expression and prevent overfitting [2, 42]. The parameter λ controls the strength of regularization, with a larger value indicating a greater emphasis on the simplicity or sparsity of the model parameters depending on the specific choice of $\mathcal{R}(\theta)$.

After obtaining θ^*, we typically calculate the loss function and accuracy on the testing set. If the model $\hat{f}_{\theta^*}$ exhibits relatively low loss and high accuracy on both the training and testing sets, we can consider it as a good approximation of the target function f. This indicates that the model has successfully generalized from the training set to unseen data and is likely to perform well on new data.

Once we obtain $\hat{f}_{\theta^*}$, we analyze the patterns of f by examining $\hat{f}_{\theta^*}$. First, the complexity of f can be analyzed by observing the accuracy of $\hat{f}_{\theta^*}$ under different hyperparameters, such as the number of hidden neurons and layers in the neural network. This allows for a rough estimation of f's complexity. Second, we can determine the sensitivity of f to different features by taking the derivative of $\hat{f}_{\theta^*}$, enabling the determination of the significance of these features. Third, if the form of $\hat{f}_{\theta^*}$ is relatively simple or becomes simple after sparse optimization, an approximate explicit expression of f can be directly obtained. Finally, error analysis of f_{θ^*} also facilitates the understanding of f's properties, such as differences in complexity in varying regions.

In cases where a suitable pattern cannot be obtained, this could indicate an improper selection of our mapping. For instance, if the number of features X is insufficient or if f is too complex, it may be challenging to find a simple $\hat{f}_\theta$ that can approximate the function and reveal its underlying patterns. In such instances, we need to modify $f(X)$ based on the insights gained from the previous round of exploration, such as by adding specific features or considering the properties of f on specific domains, and continue with the next round of exploration. This process continues iteratively.

3 Fundamental Concepts of Machine Learning and Associated Caveats

Machine learning is a field that employs computational models to learn patterns in data. This section will provide an overview of some basic machine learning models and discuss several crucial caveats in employing these models.

3.1 Machine Learning Models

3.1.1 Linear Models

Linear models are perhaps the simplest type of machine learning model, and they make a good starting point for the study of machine learning algorithms. They model the relationship between the input features and the output as a linear combination of the input features.

Suppose we have p input features, a linear model is a hyperplane and is given by the equation

$$\hat{Y} = \hat{f}_\theta(X) = \beta^\top X - b = \beta_{(1)} X_{(1)} + \beta_{(2)} X_{(2)} + \cdots + \beta_{(p)} X_{(p)} - b.$$

Here, $[X_{(1)}, X_{(2)}, \ldots, X_{(p)}] = X \in \mathbb{R}^p$ are the input features, $\hat{Y}$ is the output, and $\{[\beta_{(1)}, \ldots, \beta_{(p)}] = \beta \in \mathbb{R}^p, b\} = \theta$ are the parameters of the model. The parameters are typically learned from the data using a method called *least squares* which minimizes the sum of the squared residuals, the differences between the observed data Y_i and the predicted output $\hat{Y}_i$

$$\min_{\beta, b} \sum_{i=1}^{N} (Y_i - \beta^\top X_i + b)^2.$$

Despite their simplicity, linear models can be quite effective in practice, particularly when the data are actually linearly separable or close to it.

3.1.2 Support Vector Machines (SVM)

Support Vector Machines (SVMs) are a set of supervised learning methods particularly well suited for classification of complex but small- or medium-sized datasets.

The search for the optimal hyperplane is formulated as an optimization problem. The aim is to minimize the norm of the weight vector, $\|\beta\|^2$, which corresponds to maximizing the margin. This is subject to a set of constraints ensuring that each data point X_i is correctly classified with respect to the hyperplane

$$Y_i(\beta^\top X_i - b) \geq 1, \quad \forall i.$$

In this constraint, Y_i represents the class label of the data point X_i, with a value of either +1 or -1. The expression $Y_i(\beta^\top X_i - b)$ should be greater than or equal to 1 for all i, indicating that each data point is on the correct side of the margin or, at the very least, on the boundary of the margin.

Therefore, the optimization problem can be described more precisely as

$$\min_{\beta,b} \frac{1}{2}\|\beta\|^2 \quad \text{subject to} \quad Y_i(\beta^\top X_i - b) \geq 1, \quad \forall i.$$

This formulation ensures that the hyperplane has the maximum possible margin between the classes, while achieving accurate classification of the data points.

For non-linearly separable data, SVMs utilize the "kernel trick", a method to map the input data into a higher dimensional space where it can be linearly separable. Different kernel functions can be used depending on the nature of the data, such as polynomial kernels and radial basis function (RBF) kernels. Despite their mathematical complexity, SVMs have a geometric interpretation and can be intuitively understood as trying to find the "widest possible street" that separates the different classes.

3.1.3 Neural Network (NN) Regression

Neural networks are flexible function approximators that use layers of neurons to model complex relationships. In a regression context, a neural network learns a mapping from inputs to a continuous output. Each neuron applies a series of transformations, first a linear transformation and then a non-linear activation function, to its input.

Consider a fully connected network [47] with a total of n_L layers, each containing n_H neurons, where the input is $X^{(0)} = X$ and the output is $\hat{Y} = \hat{f}(X) = X^{(n_L)}$. The neuron j in layer l can be represented as

$$X_{(j)}^{(l)} = \rho\left(\sum_{i=1}^{n_H} w_{ji}^{(l)} X_{(i)}^{(l-1)} + b_j^{(l)}\right),$$

where $w_{ji}^{(l)}$ and $b_j^{(l)}$ are the weights and bias for neuron j in layer l, $X_{(i)}^{(l-1)}$ are the outputs of the neurons in the previous layer, and ρ is the activation function. Common choices for ρ include the sigmoid, hyperbolic tangent, and rectified linear unit (ReLU) functions. The network's weights and biases are learned by minimizing a loss function, often the mean squared error for regression tasks, using an algorithm such as gradient

descent or one of its variants. In the interest of generality, it is common in machine learning to omit the bias term and represent the function directly in matrix form

$$\hat{f}(X) = \rho(Wn_L \cdots W_2\rho(W_1X)\cdots),$$

where this representation is an equivalent formulation. It can be interpreted as appending an additional dimension to the feature vector set to 1, transforming the expression $\sum_{i=1}^{n} w_{ji}X_{(i)} + b_j$ into $\sum_{i=1}^{n+1} w_{ji}X_{(i)}$, with $w_{j(n+1)}$ equating to b_j and $X_{n+1} = 1$. For clarity, unless otherwise specified, all neural networks discussed in this paper include the bias term.

3.1.4 Neural Network (NN) Classification

In a classification context, a neural network learns to classify inputs into discrete categories. The architecture is similar to that of a regression network, but the final layer typically has as many neurons as there are classes, and it applies a softmax function to produce a probability distribution p over the classes.

The output of the jth neuron in the softmax layer, denoted by $p_{(j)}$, can be represented as

$$p_{(j)} = \frac{e^{X_{(j)}^{(n_L)}}}{\sum_{k=1}^{K} e^{X_{(k)}^{(n_L)}}},$$

where $X_{(j)}^{n_L}$ is the input to the jth neuron in the softmax layer, and K is the number of classes. The weights and biases are optimized by minimizing the cross-entropy loss, which measures the discrepancy between the predicted and true probability distributions across the classes.

3.2 Caveats

Machine learning methods have demonstrated tremendous efficacy across a variety of tasks. However, several potential caveats may affect their performance and interpretation:

3.2.1 Choice of Model

In a scenario where your primary goal is salience analysis, the choice of a machine learning model is heavily influenced by interpretability, ability to reveal feature importance, and the potential to infer functional forms between inputs and outputs. Here are some considerations to help guide your model selection:

1. Interpretability: When the aim is to understand what input features are important and their relationships with the output, interpretability becomes a crucial criterion for model selection. Models, such as linear regression, logistic regression, and decision trees, are traditionally more interpretable than, say, deep neural networks or support vector machines.

2. Inherent Feature Importance: Some models inherently provide feature importance measures. For example, the size of coefficients in linear models can indicate feature importance, and tree-based models provide feature importance based on the frequency of a feature being used to split the data.
3. Flexibility vs. Interpretability: More flexible models like neural networks can model complex relationships, but they often lack interpretability. Conversely, simpler models like linear regression provide clearer insight into relationships between variables but may fail to capture complex, non-linear relationships.
4. Trade-Off Between Accuracy and Interpretability: There is often a trade-off between model accuracy and interpretability. In salience analysis, we might be willing to sacrifice some accuracy for better interpretability, which should be factored into the model selection process.

3.2.2 Training and Testing, Overfitting

One common pitfall in machine learning is overfitting, where the model learns the training data too well and performs poorly on unseen test data. In salience analysis scenarios, understanding the relationship between features and output is crucial. To ensure the model captures the actual underlying relationships and not the noise, controlling overfitting is essential. Here are some strategies that might help:

1. Regularization: This technique penalizes the complexity of the model, discouraging learning overly complex patterns that might be due to noise. The common forms of regularization include ℓ_1- and ℓ_2-regularization.
2. Early Stopping (for Neural Networks): During the training process, monitor the model's performance on a validation set. Stop training as soon as the performance on the validation set begins to degrade.
3. Dropout (for Neural Networks): Randomly "dropping out" units in a neural network during training can prevent complex co-adaptations on training data, which helps to avoid overfitting [58].
4. Interpretable Models: If the main goal is to understand the relationships between features and output, using simpler, interpretable models such as linear regression or decision trees could be beneficial. These models may be less prone to overfitting compared to complex models like deep neural networks.

3.2.3 Randomness

When conducting salience analysis with machine learning models, it is crucial to understand and handle the randomness introduced by stochastic training processes. You want to ensure that the salience you identify is not due to the randomness in the training process, but truly indicative of the underlying relationships in your data. Below are a few strategies specifically for this context:

1. Feature Importance Measures: Many models provide ways to measure the importance of features, either directly (like coefficients in linear models), or indirectly (like gradients in neural networks). However, remember that these measures are

impacted by the stochasticity of training. To mitigate this, consider averaging feature importance over multiple runs or training multiple models using different random seeds and averaging their feature importance measures.
2. Ablation Studies: One way to understand the importance of a feature is to see how much the model's performance drops when that feature is removed. By conducting this analysis over multiple runs (with different random seeds), you can obtain a measure of feature importance that is robust to the randomness of the training process.
3. Controlled Training Processes: Reduce the randomness in the training process through techniques, such as decreasing the learning rate over time, using a larger batch size, or using a different optimization algorithm less sensitive to stochasticity, like RMSprop [53] or Adam [36].

3.2.4 Data Imbalance

Machine learning models can perform poorly when there is a class imbalance in the training data. In such cases, the model might be biased toward the majority class. Handling imbalanced data is crucial, especially in a context where understanding the relationships between features and output is of primary importance. We list some strategies specifically geared toward such scenarios:

1. Resampling Techniques: One can alter the dataset itself to address the imbalance:
   - *Oversampling the Minority Class:* This involves creating or synthesizing new instances of the minority class until it reaches a similar number as the majority class. While it can balance the classes, it might lead to overfitting due to the replication of the minority instances.
   - *Undersampling the Majority Class:* This involves removing instances from the majority class until it reaches a similar number as the minority class. While it can be effective in balancing the dataset, it may cause loss of information by excluding potentially important instances from the majority class.

   Both approaches aim to balance the distribution between the majority and minority classes but come with potential drawbacks that must be carefully considered during implementation.
2. Cost-Sensitive Learning: While this technique can always be used in cases of imbalanced data, the costs need to be chosen carefully in this context. A simple heuristic like setting the cost inversely proportional to class frequency might not be the best choice, as it might lead to the model focusing too much on rare classes that might not have enough instances to derive reliable feature-output relationships.
   Another method is adjusting the temperature of the softmax function in the output layer of the model. The softmax function is often used in the final layer of a classification neural network to convert the outputs to probability values for each class. You can adjust the temperature based on the class frequencies, so that the model is made more sensitive to the minority class. However, keep in mind that it needs to be used carefully, as it can make the model more prone to overfitting to the minority class.

3.2.5 Dataset Size

Throughout this paper, we engage with datasets of different sizes depending on the complexity of the model under study—ranging from simple linear models to more elaborate two- or three-layer neural networks. When the goal is to extract a concise and clear formula, we prefer simpler models such as linear models, which have fewer parameters, thereby requiring less data. On the other hand, for feature analysis, a more accurate representation of the unknown function relationship is desired, without overfitting, which calls for more sophisticated models and larger datasets. In practice, the actual size of a dataset is typically determined through an empirical approach that involves training a model on datasets of different sizes and comparing the results. Upon reaching a point where improvement is only marginal, we consider the dataset size to be sufficient.

4 Programs

In this section, we introduce our program, which is composed of two primary modules. The first module computes variables of interest in affine Deligne–Lusztig varieties, as described in Sect. 4.1. The second module analyzes the data generated using machine learning techniques, as detailed in Sect. 4.2. Figure 2 illustrates the overall workflow of the program.

4.1 Program for Affine Deligne–Lusztig Varieties

We give a short introduction to the program for the affine Deligne–Lusztig varieties based on Python.

Choosing a group. The first input determines the type of algebraic group. The valid inputs are An, Bn, Cn, Dn, 2An, and 2Dn.

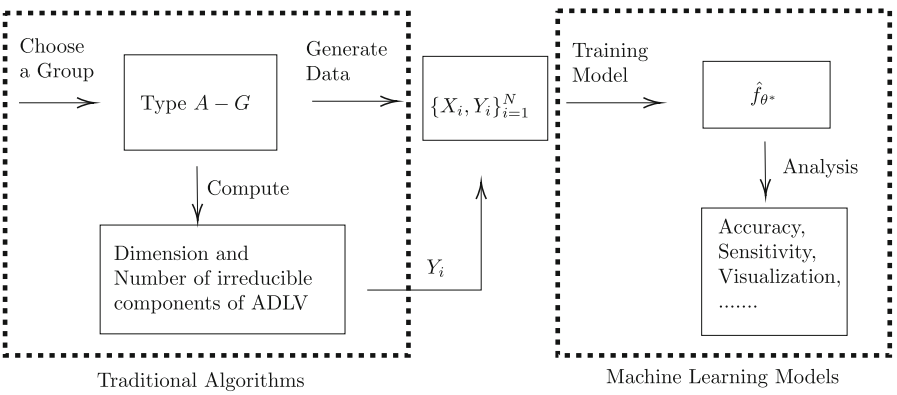


Fig. 2 Workflow of the program

To simplify notation, we focus on the group $G = \mathrm{SL}_n$ (type A_{n-1}) throughout this section. However, we note that the program is compatible with all classical groups.

Input for w. The element $w \in W_a$ can be expressed in the following two ways:

- the product of the translation part and the finite part;
- the product of a sequence of simple reflections.

Element of the form $w = t^{(a_1,\ldots,a_n)} s_{i_1} \cdots s_{i_r}$, where $(a_1, \ldots, a_n) \in X_*$ and all $i_j \in \{1, 2, \ldots, n-1\}$, is written as affine_Weyl($[a_1, \ldots, a_n]$, $[i_1, \ldots, i_r]$). Elements of the form $w = s_{i_1} \cdots s_{i_r}$, where all $i_j \in \{0, 1, 2, \ldots, n-1\}$ is written as exp($[i_1, \ldots, i_r]$). For example, in $n = 3$ case (type A2). affine_Weyl([1, 0, −1],[1, 2]) = exp([0, 2]).

The simple reflections are s[0], s[1],. . ., s[$n - 1$]. The identity element of W_a is Id.

Input for b**.** In this program, we input the Newton point ν_b of b instead of b. Note that in the case of $\mathbf{G} = \mathrm{SL}_n$, the conjugacy class $[b]$ is determined by its Newton point.

Function (dim) Computing dimension of ADLV:

$$\dim(w, \nu)$$

Input: $w \in \tilde{W}$; $\nu \in \mathbb{Q}^n$
Output: $\dim X_w(b)$
Description: If $X_w(b) = \emptyset$, the output is "empty".

Function (irr) Computing irreducible components of ADLV:

$$\mathrm{irr}(w, \nu)$$

Input: $w \in \tilde{W}$; $\nu \in \mathbb{Q}^n$
Output: $\sharp \mathbf{J}_b(F) \backslash \Sigma^{\mathrm{top}} X_w(b)$
Description: If $X_w(b) = \emptyset$, the output is 0.

Function (dim_irr_print) Listing all b such that $X_w(b) \neq \emptyset$ and computing dimension and number of irreducible components:

$$\mathrm{dim\_irr\_print}(w)$$

Input: $w \in \tilde{W}$
Output: Print the following:
Newton point = ν, dim = $\dim X_w(b)$, irr = $\sharp \mathbf{J}_b(F) \backslash \Sigma^{\mathrm{top}} X_w(b)$
Description: The function lists all b, such that $X_w(b) \neq \emptyset$.

Example A2 case.
Input:
w = affine_Weyl([1, 1, −2], [2, 1]); dim_irr_print(w)
Output:
Newton point = [1/2, 1/2, −1], dim = 1, irr = 1
Newton point = [0, 0, 0], dim = 3, irr = 1

Input:
print(dim(w,[0, 0, 0]), irr(w,[1/2, 1/2, −1]), dim(w,[1, 0, −1]), irr(w,[2, 0, −2]))
Output:
3 1 empty 0

This demonstrates the three functions applied to the element $w = t^{(1,1,-2)}s_2s_1$. The first input demonstrates the behavior of the function **dim_irr_print**. The second input demonstrates the behavior of the functions **dim** and **irr** for various $[b] \in B(G)$ with $X_w(b) \neq \emptyset$ and $X_w(b) = \emptyset$. See Function (dim) and Function (irr) above for our convention for the dimension and the number of irreducible components of an empty set.

4.2 Program for Machine Learning

The program implemented in this module is primarily designed for generating datasets, training models, and performing analyses of the trained models. The input and output parameters, along with the intended usage of the four main functions, are outlined below:

Function (GenerateDataset) Generate Dataset for Training:

GenerateDataset (str1, str2)

Input: str1: filename for data; str2: filename for dataset
Description: The file generated in Sect. 4.1 is in the Numpy array format and named "str1". For efficient subsequent operations, these data are converted into a PyTorch tensor and structured as a dataset named "str2". The program automatically shuffles the data, allocating 80% as the training set and 20% as the testing set.

Function (LinearReg) Linear Regression:

LinearReg($\mathbf{X}$, $\mathbf{Y}$, λ)

Input: $\mathbf{X} \in \mathbb{R}^{N\times c}$, $\mathbf{Y} \in \mathbb{R}^N$, $\lambda \in \mathbb{R}$
Output: $\beta \in \mathbb{R}^c$, $b \in \mathbb{R}$
Description: This function is used to solve a linear regression problem as outlined in Sect. 3.1.1. The hyperparameter λ, as discussed in Sect. 2.4, regulates the magnitude of the regularization term. The ith row of the matrix $\mathbf{X}$ represents X_i, and the ith element of the vector $\mathbf{Y}$ represents Y_i. The same convention applies throughout the following discussion.

Function (LinearCls) SVM:

LinearCls($\mathbf{X}$, $\mathbf{Y}$, λ)

Input: $\mathbf{X} \in \mathbb{R}^{N\times c}$, $\mathbf{Y} \in \mathbb{R}^N$, $\lambda \in \mathbb{R}$
Output: $\beta \in \mathbb{R}^c$, $b \in \mathbb{R}$

Description: The function is used to solve SVM problem in Sect. 3.1.2, with the hyperparameter of the regularization term set to λ.

Function (NetReg) Neural Network Regression:

$$\text{NetReg}(\mathbf{X}, \mathbf{Y}, n_L, n_H, \lambda)$$

Input: $\mathbf{X} \in \mathbb{R}^{N\times c}$; $\mathbf{Y} \in \mathbb{R}^N$; $n_L \in \mathbb{N}^+$; $n_H \in \mathbb{N}^+$; $\lambda \in \mathbb{R}$
Output: $\hat{f} : \mathbb{R}^c \to \mathbb{R}$
Description: The output $\hat{f}$ is a trained fully connected neural network encapsulated within an instance of "torch.nn.module", a PyTorch class designed for layer management and network definition. It is to be noted that, unless explicitly stated, all neural networks referenced in this program conform to this standard representation. The network comprises N_L layers, each with N_H neurons, as described in Sect. 3.1.3. Weight decay is used as the regularization term, with the hyperparameter λ controlling its strength.

Function (NetCls) Neural Network Classification:

$$\text{NetCls}(\mathbf{X}, \mathbf{Y}, n_L, n_H, \lambda)$$

Input: $\mathbf{X} \in \mathbb{R}^{N\times c}$; $\mathbf{Y} \in \mathbb{R}^N$; $n_L \in \mathbb{N}^+$; $n_H \in \mathbb{N}^+$; $\lambda \in \mathbb{R}$
Output: $\hat{f} : \mathbb{R}^c \to \mathbb{R}$
Description: The symbol $\hat{f}$ denotes a trained fully connected network designed specifically for classification problems. The network comprises N_L layers, each with N_H neurons, as described in Sect. 3.1.4. Weight decay is used as the regularization term, with the hyperparameter λ controlling its strength.

Function (NetGrad) Sensitive Analysis:

$$\text{NetGrad}(\mathbf{X}, \mathbf{Y}, \hat{f})$$

Input: $\mathbf{X} \in \mathbb{R}^{N\times c}$, $\mathbf{Y} \in \mathbb{R}^N$, $\hat{f} : \mathbb{R}^c \to \mathbb{R}$
Output: $g \in \mathbb{R}^c$
Description: This function is utilized to quantify the sensitivity of individual features in the trained network $\hat{f}$. The sensitivity is gauged by evaluating the mean of the absolute values of the derivatives of the loss function with respect to each feature, taken over the dataset $\{\mathbf{X}, \mathbf{Y}\}$. Specifically, for the jth feature, the sensitivity is calculated as

$$g_{(j)} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\partial \mathcal{L}(\hat{f}(X_i), Y_i)}{\partial X_{i,(j)}}\right|,$$

where $X_{i,(j)}$ denotes the jth feature of the input example X_i, N is the total number of examples, and $\mathcal{L}$ represents the loss function. The term $g_{(j)}$ delivers an aggregate measure of how sensitive the loss function is to variations in the jth feature, thus quantifying the importance of that feature in the learned representation captured by $\hat{f}$.

5 Searching for a Dimension Formula

5.1 Virtual Dimension

The journey toward the dimension formula of affine Deligne–Lusztig varieties has a long history. For the affine Grassmannian case, Rapoport proposed the dimension formula in [51], drawing inspiration from Chai's earlier work [10] on the length function of chains of σ-conjugacy classes. This conjecture was ultimately validated by a series of researchers, primarily [16, 20, 60, 70].

In this paper, our attention is on the affine flag case, a significantly more challenging problem. Görtz, Haines, Kottwitz, and Reuman proposed a conjectural formula for $\dim X_w(b)$ for most pairs (w, b) in [17]. This was partly inspired by the aforementioned dimension formula for the affine Grassmannian case. This conjecture was verified by He in [25] and [28]. However, our understanding of the remaining cases, which include many crucial applications to number theory and the Langlands program, remains limited. We aim to broaden this understanding using machine learning.

We revisit the concept of virtual dimension introduced by He in [25]. This was inspired by the conjecture of Görtz, Haines, Kottwitz, and Reuman. Let $w \in W_a$ and express it as $w = xt^{\mu}y$ as in Sect. 2.2. Define $\eta(w) = yx$ and

$$d_w(b) = \frac{1}{2}\big(\ell(w) + \ell(\eta(w))\big) - \langle \nu_b, \rho \rangle - \frac{1}{2}\mathrm{def}(b).$$

He demonstrated in [25] and [28] the following result.

Theorem 1 *Suppose $X_w(b) \neq \emptyset$. Then*

1. $\dim X_w(b) \leq d_w(b)$.
2. *If w is 2-regular and $\mu - \nu_b$ is "sufficiently large", then* $\dim X_w(b) = d_w(b)$.

The virtual dimension formula is a practical approximation of the real dimension. The discovery of this formula took experts considerable time, stretching from Rapoport's lectures in 1996 [51] to the introduction of the virtual dimension formula in the most general setting by He in 2012 [25]. This section aims to illustrate how machine learning could help us rediscover this formula, accelerating the research process counterfactually.

The dimension of affine Deligne–Lusztig varieties depends on two parameters, w and b. We note that the b-part of the virtual dimension formula is relatively simpler to uncover than the w-part, and we omit the details of the learning process for now. In this section, we concentrate on the group SL_5, the case $b = 1$, and randomly generated elements w. We investigate how machine learning can shed light on the correlation between w and the dimension $\dim X_w(1)$. Our method does not rely on prior knowledge of the dimension formula in the Grassmannian case or the virtual dimension formula.

The selection of appropriate input features is crucial for machine learning. We work with the affine Weyl group of type $\tilde{A}_4$, where each element $w = t^{\lambda}u \in W_a$ is made up of the translation part λ and the finite part u. We include both parts as input features: λ as a vector, and u as a permutation denoted by $u = [u_1, u_2, u_3, u_4, u_5]$.

For classical Deligne–Lusztig varieties X_w [15], it's known that $\dim X_w = \ell(w)$. Therefore, we anticipate that the dimension of affine Deligne–Lusztig varieties $X_w(1)$ is also related to $\ell(w)$. Consequently, we include the length function for both w and u as input features.

5.2 Complexity Test

To evaluate the complexity of the mapping, we utilize neural networks. Specifically, we consider an n_L-layer fully connected network with ReLU activation and n_H hidden neurons. This network can be expressed as

$$\hat{f}_\theta(X) = \beta^\top \rho(W n_L \cdots W_2 \rho(W_1 X) \cdots), \quad \theta = W_1, W_2, \ldots, W_{n_L}, \beta,$$

where $\hat{f}_\theta(X)$ is the predicted output of the neural network for input $X \in \mathbb{R}^c$, and θ is the set of all trainable parameters, including weights

$$W_1 \in \mathbb{R}^{n_H \times c}, W_2 \in \mathbb{R}^{n_H \times n_H}, \ldots, W n_L \in \mathbb{R}^{n_H \times n_H}, \beta \in \mathbb{R}^{n_H}.$$

We use the Rectified Linear Unit (ReLU) activation function, defined as $\rho(x) = \max(0, x)$, for each hidden layer.

To quantify the accuracy of the optimized $\hat{f}_{\theta^*}$ in approximating f after training, we use two metrics: accuracy and mean error. Since the dimension is always an integer, we round the inferred results $\hat{f}_{\theta^*}(X_i)$ to the nearest integer for accuracy. Specifically, we define accuracy as

$$\text{Accuracy} = \frac{1}{N} \sum_i^N \delta(Y_i, \text{round}(\hat{f}_{\theta^*}(X_i))),$$

where N is the number of samples, and $\delta(\cdot)$ is the indicator function that outputs 1 if the arguments are equal and 0 otherwise. The mean error is defined as

$$\text{Mean Error} = \frac{1}{N} \sum_i^N |Y_i - \hat{f}_{\theta^*}(X_i)|.$$

In the tables below, the bold values highlight the terms with relatively large absolute values, which indicate possible significants suggested by the experiments.

Dataset 1. We describe the first dataset used in our experiments. We randomly choose 5000 elements w from the set W_a such that $\ell(w) < 30$ and $X_w(1) \neq \emptyset$. For each $w = t^\lambda u$, we express $\lambda = [\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5]$ and $u = [u_1, u_2, u_3, u_4, u_5]$ as a permutation. We then compute the dimension $\dim X_w(1)$ for all these w.

Experiment 1. In this experiment, we use Dataset 1 to train a neural network to predict the dimension of $X_w(1)$ for each $w = t^\lambda u$, where λ and u are defined as above. Specifically, the input vector for each w is defined as $X =$

Table 1 Testing error of different neural networks for Dataset 1

n_L	n_H		
	10	20	40
1	0.53	0.53	0.52
2	0.53	0.53	0.51
3	0.52	0.51	0.51

$[\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, u_1, u_2, u_3, u_4, u_5, \ell(u), \ell(w)]$, and the corresponding output is $Y = \dim X_w(1)$. We obtain a dataset of 5000 samples $\{X_i, Y_i\}_{i=1}^{5000}$ and train a neural network on it. The mean error was computed on the testing set to assess the neural network's prediction accuracy on unseen data, reflecting the average discrepancy between the predicted $\hat{f}_{\theta^*}(X_i)$ and true values Y_i. The experimental results are presented in Table 1.

Analysis. The results of Experiment 1 show that the error obtained is relatively small. Furthermore, we find that increasing the number of layers n_L and hidden units n_H in the network does not significantly enhance the accuracy of the neural network in predicting the dimension. These observations lead us to hypothesize that a linear model may be sufficient to approximate the mapping f from the input data to the output dimension. This hypothesis will be investigated in the following subsection with the performance of a linear model.

5.3 Linear Model

A linear function without a bias term can be expressed as $\hat{f}_\theta(X) = \beta^\top X = \sum_{i=1}^{c} \beta_{[i]} X_{[i]}$, where $\beta_{[i]}$ denotes the ith element of the vector β which represents the coefficient of the ith term. For linear models, the mean error are used to evaluate the approximation of $\hat{f}_{\theta^*}$ to f.

Experiment 2. We use Dataset 1 to train a linear model to predict the dimension of $X_w(1)$ for each $w = t^\lambda u$, where λ and u are as previously described. The mean error of the linear model is 0.65.

Analysis. Recall that the presentation $w = xt^\mu y$ is crucial for understanding the properties of the affine Weyl group element w, where $x, y \in W_0$ and $\mu \in X_*(T)$ is dominant. However, this presentation is not unique. Imposing a subtle restriction on x or y yields two uniquely determined presentations, but these may be different and thus are not canonical. Alternatively, one may restrict to those elements $w = xt^\mu y$ where μ is *dominant regular*, i.e., $\mu_1 > \mu_2 > \mu_3 > \mu_4 > \mu_5$. For such w, there are uniquely determined $x, y \in W_0$ with $w = xt^\mu y$.

We consider a subset of Dataset 1 consisting of those w where μ is dominant regular, hypothesizing that the chosen features are more meaningful on this subset. This subset consists of 1037 entries. A newly trained linear model on this subset attains a mean test error of 0.62, which confirms our expectation. Hence, we consider analyzing the linear model under the dominant regular constraints by resampling 5000 points to form Dataset 2.

Dataset 2. Note that any element with a regular translation part can be written as $w = xt^{\mu}y$ where $x, y \in W_0$ and $\mu = [\mu_1, \mu_2, \mu_3, \mu_4, \mu_5]$ is dominant regular, i.e., $\mu_1 > \mu_2 > \mu_3 > \mu_4 > \mu_5$. To investigate the neural network's performance on regular translation parts, we randomly choose 5000 elements from the set of all such elements, where μ is dominant regular with $7 > \mu_1 > \mu_2 > \mu_3 > \mu_4 > \mu_5 > -7$, and $X_w(1) \neq \emptyset$. We compute $\dim X_w(1)$ for all these w.

Experiment 3. We use Dataset 2. For each $w = xt^{\mu}y$, set

$$\begin{aligned} X = [&\delta(x(\alpha_{12})), \delta(x(\alpha_{13})), \ldots, \delta(x(\alpha_{45})), \ell(x), \\ &\mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \delta(y^{-1}(\alpha_{12})), \ldots, \delta(y^{-1}(\alpha_{45})), \ell(y), \ell(w)] \end{aligned}$$

and

$$Y = \dim X_w(1).$$

The input features are explained in detail as follows. We write $\alpha_{ij} = e_i - e_j$ for the root and

$$\delta(\alpha_{ij}) = \begin{cases} 1, & i > j, \\ 0, & i < j, \end{cases}$$

for the indicator function of the negative roots. A linear combination of the permutation values $x(1), \ldots, x(n)$ would be very hard to interpret mathematically, which is why we use the δ-values.

It is important to note for the neural network, the difference between presenting a permutation as $\delta(x(\alpha_{ij}))$ or $\delta(x^{-1}(\alpha_{ij}))$ is significant. While these two presentations are *mathematically* equivalent, transitioning from one to the other is a non-linear procedure [66, Sect. 5.2]. We use x and y^{-1}, which have more direct mathematical interpretations than their inverses x^{-1} and y. Specifically, y^{-1} indicates the *Weyl chamber* of w, whereas x typically indicates the Weyl chamber of the inverse w^{-1}.

We obtain $\{X_i, Y_i\}_{i=1}^{5000}$. Upon applying the neural network to this dataset, the results of this experiment are summarized in Table 2. The average error is found to be 0.65.

Table 2 Coefficient of Experiment 3

Feature	Corresponding coefficient(s)
$\delta(x(\alpha_{ij})), \delta(y^{-1}(\alpha_{ij}))$	$\begin{pmatrix} 0.12 & -0.04 & -0.05 & -0.24 \\ & 0.14 & -0.05 & -0.04 \\ & & 0.13 & -0.02 \\ & & & 0.15 \end{pmatrix}, \begin{pmatrix} 0.10 & -0.02 & -0.08 & -0.03 \\ & 0.07 & 0.00 & -0.06 \\ & & 0.02 & 0.04 \\ & & & 0.06 \end{pmatrix}$
$\ell(x)$	0.10
μ_i	$[0.13, -0.09, -0.02, 0.08, -0.10]$
$\ell(y)$	0.10
$\ell(w)$	**0.52**

Analysis. From the above table, we observe that the length of w is the most significant feature, with a coefficient approximately equal to $1/2$. We thus deduct this potential leading term from the dimension for our subsequent experiments. Specifically, the output Y will be $\dim X_w(1) - \frac{1}{2}\ell(w)$. It is noteworthy that x and y belong to the finite Weyl group, and their contribution to the dimension should be limited. Conversely, the range of μ is unbounded and it is anticipated that μ could contribute to the potential leading term of the linear approximation of the dimension. We therefore hypothesize that the contribution of μ in the linear model is already encapsulated in the term $\frac{1}{2}\ell(w)$ (i.e., the contribution of μ is given by $\langle \mu, \rho \rangle$). Consequently, we will eliminate μ_i from X.

Experiment 4. The analysis of the dimension of affine Deligne–Lusztig varieties, even for smaller rank groups such as SL_3, suggests that different Weyl chambers may exhibit different patterns (cf. [16, Sect. 7] and [3]). In [23], He introduced the technique of partial conjugation, which, to some extent, reduces the problem to the dominant chamber. Therefore, we primarily focus on elements in the dominant Weyl chamber, i.e., elements of the form $w = t^{\mu} y$ where μ is dominant regular.

Dataset 3. This dataset is defined similarly to Dataset 2. We randomly choose 5000 elements $w = t^{\mu} y$ where $y \in W_0$ and $\mu = (\mu_1, \mu_2, \mu_3, \mu_4, \mu_5)$ is dominant regular with $9 > \mu_1 > \mu_2 > \mu_3 > \mu_4 > \mu_5 > -9$, and $X_w(1) \neq \emptyset$.

For each $w = t^{\mu} y$, we define

$$X = [\delta(y^{-1}(\alpha_{12})), \ldots, \delta(y^{-1}(\alpha_{45})), \ell(y)]$$

and

$$Y = \dim X_w(1) - \frac{1}{2}\ell(w).$$

We derive $\{X_i, Y_i\}_{i=1}^{5000}$ and apply a linear model. We obtain $\hat{f}_{\theta^*}$ with an average error of 0.30. The coefficients are listed in Table 3.

Recall that $\ell(y) = \sum_{i<j} \delta(y^{-1}(\alpha_{i,j}))$. If we consider this linear dependence, we observe that the actual leading coefficient in the above linear model is $\ell(y)$ times

$$0.42 + (0.02+0.05+0.10+0.13 - 0.05 + 0.07 + 0.11 - 0.07+0.05+0.00) \cdot \frac{1}{10} = 0.46.$$

Table 3 Coefficient of Experiment 4

Feature	Corresponding coefficient(s)
$\delta(y^{-1}(\alpha_{ij}))$	$\begin{pmatrix} 0.02 & 0.05 & 0.10 & 0.13 \\ & -0.05 & 0.07 & 0.11 \\ & & -0.07 & 0.05 \\ & & & 0.00 \end{pmatrix}$
$\ell(y)$	**0.42**

Analysis. We could conjecture that $\frac{1}{2}\ell(y)$ is the leading term, with an "error term" that is one order of magnitude smaller. Before progressing with our experiments, we introduce some general strategies and terminology for the training and interpretation of linear models.

In the context of linear regression, the inclusion of a regularization term is essential when dealing with highly linearly correlated features. This situation can cause instability and unreliable estimates of the regression coefficients. Regularization involves adding a penalty term to the loss function, generally based on the magnitudes of the regression coefficients.

Practically, the ℓ_2-norm is a commonly selected regularization term. The loss function can be represented as

$$\min \sum_{i=1}^{N} (Y_i - \beta^\top X_i)^2 + \lambda \|\beta\|_2^2.$$

The ℓ_2-regularization in linear regression results in a unique optimal solution, providing model stability and avoiding multiple solutions. The ℓ_2-regularization penalizes larger regression coefficients proportionally, leading to more stable models with reduced overfitting and improved generalization.

For instance, if we only use one feature, $\ell(y)$, to train the linear model, which does not require a regularization term, we can obtain a model with an average error of 0.30 and a coefficient of 0.47. This coefficient is much closer to 0.5 because of the ℓ_2-regularization's preference for small and average coefficients when dealing with linear dependence. Therefore, a common strategy is to first identify the most important features, discard less important features, and then retrain the model's parameters [4, 22]. This process allows for obtaining more accurate results and helps in mitigating the effects of highly correlated features.

Additionally, the ℓ_1-norm is also a common metric used in regularization terms and fidelity terms. It can be expressed in the following form:

$$\min \sum_{i=1}^{N} |Y_i - \beta^\top X_i| + \lambda \|\beta\|_1.$$

When used as a fidelity term, the ℓ_1-norm penalizes absolute differences between the model predictions and the true targets. This makes it more robust to outliers compared to the ℓ_2 norm, which is more sensitive to large errors.

Also, as a regularization term, the ℓ_1 norm induces sparsity in the model parameters, driving many parameters close to zero. This performs automatic feature selection, removing uninformative features and improving interpretability. In contrast, the ℓ_2-norm does not induce sparsity, but instead diffuses weight across all parameters. Apply the ℓ_1 model. We obtain $\hat{f}_{\theta^*}$ with average error of 0.18. The coefficients are listed in Table 4.

Table 4 Coefficient of ℓ_1 model for Experiment 4

Feature	Corresponding coefficient(s)
$\delta(y^{-1}(\alpha_{ij}))$	$\begin{pmatrix} 0.00\ 0.00\ 0.00\ 0.00 \\ 0.00\ 0.00\ 0.00 \\ 0.00\ 0.00 \\ 0.00 \end{pmatrix}$
$\ell(y)$	**0.50**

Experiment 5. We aim to investigate how the findings from the previous sections extend to other Weyl chambers. Given $w = xt^{\mu}y$, our goal is to estimate

$$Y = \dim X_w(1) - \frac{1}{2}\ell(w).$$

If $x = 1$, we understand that the leading term should be $\frac{1}{2}\ell(y)$. Generally speaking, there are a number of intuitive ways to combine x and y, particularly in light of the previously mentioned partial conjugation method. These include the mutual products xy and yx, as well as the Demazure products $y * x$ and $y \triangleleft x$ [24]. Considering that the δ-values only slightly contribute to the linear model in the dominant chamber, we exclude them in this experiment.

We utilize Dataset 2 (arbitrary Weyl chamber). For each $w = xt^{\mu}y$, we set

$$X = [\ell(x), \ell(y), \ell(xy), \ell(yx), \ell(yx), \ell(y \triangleleft x)].$$

This gives us the pair (X_i, Y_i) for $i = 1$ to 5000. Upon applying a linear model, we obtain $\hat{f}_{\theta^*}$ with a mean error of 0.13. The coefficients are provided in Table 5.

Analysis. It is evident that $\ell(yx)$ is the most influential feature, yet interpreting the coefficient 0.46 poses a mathematical challenge. Note that while the input features are *linearly independent*, numerous mathematical relationships exist between them, as shown by the inequality

$$|\ell(x) - \ell(y)| \leq \ell(y \triangleleft x) \leq \ell(yx) \leq \ell(y * x).$$

We can speculate that $\frac{1}{2}\ell(yx)$ could be a suitable candidate for the leading term, with the remaining terms being relatively small. This speculation leads us to the linear

Table 5 Coefficient of Experiment 5

Feature	Corresponding coefficient(s)
$\ell(x)$	0.02
$\ell(y)$	− 0.05
$\ell(xy)$	− 0.02
$\ell(yx)$	**0.46**
$\ell(y * x)$	0.04
$\ell(y \triangleleft x)$	0.04

Table 6 Number of pairs (w, b) with certain value of virt. dim. minus dim.

value of $d_w(b) - \dim X_w(b)$	0	1	2	3	4
Amount of pair	2,020,909	922,482	166,386	9885	284

model

$$\dim X_w(1) \approx \frac{1}{2}\left(\ell(w) + \ell(yx)\right),$$

which aligns with the previously mentioned virtual dimension $d_w(1)$.

Experiment 6. We now scrutinize this linear model, which we have identified as representing the virtual dimension. For all pairs (w, b) such that $w \in W_a$ satisfies $\ell(w) < 30$ and $[b] \in B(\mathrm{SL}_5)$ satisfies $X_w(b) \neq \emptyset$, we examine the difference $d_w(b) - \dim X_w(b)$ between the virtual dimension and the actual dimension of $X_w(b)$. The number of such pairs is 3,119,946. The results are presented in Table 6.

Analysis. We observe that the difference $d_w(b) - \dim X_w(b)$ is always non-negative, and equals zero in the majority of cases. Both of these observations are well documented, and represent major accomplishments in the field [25, 28]. Furthermore, we notice that this difference also seems to be upper-bounded by 4, a surprisingly small value considering the number of pairs (w, b) and the large dimensions involved. This leads us to conjecture that $d_w(b) - \dim X_w(b)$ always has a reasonably small upper bound independent of (w, b). We will explore that question further in Sect. 7.

6 Searching for Important Features

In this section, we revisit the group SL_5 without imposing any restrictions on $[b] \in B(\mathrm{SL}_5)$ or $w \in W_a$.

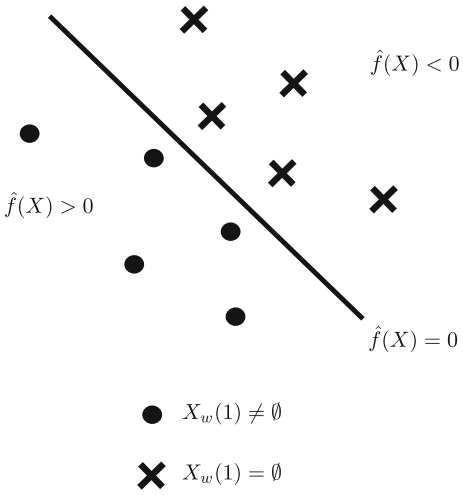
6.1 Detailed Introduction to SVM Method

To ensure self-containment and enhance understanding of the experimental outcomes, we provide a detailed introduction to the Support Vector Machine (SVM) model. This widely used machine learning algorithm is primarily employed for classification tasks and will be utilized in our experiments on the non-emptiness pattern and the condition of dimension equalling virtual dimension.

The primary objective of SVM is to identify an optimal hyperplane that effectively separates data points into different classes. In the case of binary classification, the hyperplane is chosen to maximize the margin, which refers to the distance between the hyperplane and the nearest data points from each class. In this context, we primarily focus on linear SVM for the sake of result interpretability. The equation of this hyperplane is given by

$$\hat{f}(X) = \beta^\top X - b = 0.$$

Fig. 3 A demo for SVM



For instance, in the non-emptiness pattern experiments, SVM aims to establish a hyperplane that bifurcates the dataset into two regions. Specifically, on one side of the hyperplane, all instances satisfy $X_w(1) \neq \emptyset$; on the other side, all instances satisfy $X_w(1) = \emptyset$. This is depicted in Fig. 3.

When data are not inherently linearly separable, SVM often employs techniques to find an "optimal" hyperplane. A commonly used evaluation criterion is the hinge loss, a margin-based loss function that penalizes misclassifications and encourages SVM to identify a decision boundary with a larger margin. For a binary classification problem, the hinge loss for a single data point is defined as

$$\mathcal{L}_{\text{hinge}}(Y, \hat{f}(X)) = \max(0, 1 - Y \cdot \hat{f}(X)).$$

To optimize the SVM model, the aim is to minimize the sum of hinge losses across all training data points while incorporating a regularization term. This term helps prevent overfitting and controls the complexity of the learned model, often represented by the ℓ_2-norm of the weight vector β. The SVM optimization problem can be formulated as

$$\min \sum_i \mathcal{L}_{\text{hinge}}(Y_i, \hat{f}(X_i)) + \lambda |\beta|_2^2.$$

Solving this optimization problem allows SVM to learn a decision boundary that generalizes well to unseen data, leading to accurate classification or regression predictions.

During the inference stage, an input X is classified as the first class if $\hat{f}(X)$ is greater than 0, and classified as the second class otherwise. The coefficients represented by β provide insightful information about the relationship between the input features and the class labels. Specifically, a positive $\beta_{(i)}$ suggests that as the value of $X_{(i)}$ increases, the likelihood of belonging to the first class also increases. Furthermore, a larger absolute value of $\beta_{(i)}$ signifies a stronger influence of the corresponding feature on

the classification decision. Conversely, a negative $\beta_{(i)}$ indicates a negative relationship between the feature and the likelihood of belonging to the first class.

6.2 Experiments on the Non-emptiness Pattern

Data: All $w = xt^{\mu}y \in W_a$ and $[b] \in B(\mathrm{SL}_5)$ respecting the conditions $\ell(w) < 30$ and $\nu_b \leq \mu$. The latter condition, known as Mazur's inequality, is a necessary condition for $X_w(b) \neq \emptyset$. (Dataset size: 8,705,879)

Feature:

$$X = [x_{ij}, \mu_1, \ldots, \mu_5, y_{ij}^{-1}, \eta_{ij} = \delta(\eta(w)(\alpha_{ij})), \ell(w), \nu_1, \ldots, \nu_5, \lambda_1, \ldots, \lambda_5] \in \mathbb{R}^{46}.$$

Output: $Y = \begin{cases} 1 & \text{if } X_w(1) \neq \emptyset \\ -1 & \text{if } X_w(1) = \emptyset \end{cases}$

Result: Three models were utilized in our experiments, each executed 100 times. The first model, SVM, yielded an average accuracy of 78.34%, with the average coefficients reported in Table 7. The second model, a single-layer neural ReLU classification network with 10 neurons, achieved an average training accuracy of 87.14% and an average test accuracy of 87.17%, with the average gradients documented in Table 8. The third model, a three-layer neural ReLU classification network with 20 neurons per layer, had an average training accuracy of 94.72% and an average test accuracy of 94.74%, with the average gradients detailed in Table 9.

Analysis. It is evident that the "hook"-part of η encapsulates important features, specifically $\eta_{12}, \eta_{13}, \eta_{14}, \eta_{15}, \eta_{25}, \eta_{35}, \eta_{45}$. The No Levi Obstruction (NLO) criterion for non-emptyness of ADLV, developed in [18] and further refined in [19, 63], suggests that the support of $\eta(w)$ plays a crucial role in the non-empty pattern. This connects to the values of η_{ij}, though the exact relationship is intricate and non-linear, representing an interesting problem in itself.

As seen from Table 7, another important feature is the Newton point ν_b together with its best integral approximation $\lambda(b)$. In general, we expect that $X_x(b)$ holds "often" for small $[b]$ and only occasionally for ν_b close to μ, cf. [28].

The three-layer neural network achieves a high accuracy, but the gradient is hard to interpret mathematically. This is partly due to inherent properties of the machine

Table 7 Average coefficient of SVM for non-empty

Feature	Corresponding coefficient(s)
$(x_{ij}), (y_{ij}^{-1}), (\eta_{ij})$	$\begin{pmatrix} 0.15 & 0.10 & 0.10 & 0.12 \\ & 0.08 & 0.07 & 0.10 \\ & & 0.08 & 0.10 \\ & & & 0.15 \end{pmatrix}, \begin{pmatrix} -0.08 & -0.12 & -0.15 & -0.18 \\ & -0.07 & -0.12 & -0.15 \\ & & -0.07 & -0.12 \\ & & & -0.08 \end{pmatrix}, \begin{pmatrix} -0.15 & 0.10 & \mathbf{0.39} & \mathbf{1.56} \\ & -0.07 & 0.00 & \mathbf{0.39} \\ & & -0.07 & 0.10 \\ & & & -0.15 \end{pmatrix}$
μ_i	$[0.01, 0.01, -0.00, -0.01, -0.02]$
$\ell(w)$	0.07
ν_i	$[\mathbf{-1.14}, \mathbf{-0.53}, 0.01, \mathbf{0.55}, \mathbf{1.16}]$
λ_i	$[\mathbf{0.68}, 0.30, 0.00, -0.30, \mathbf{-0.67}]$

Table 8 Average gradient of one-layer NN for non-empty

Feature	Corresponding gradient(s)
(x_{ij}), (y_{ij}^{-1}), (η_{ij})	$\begin{pmatrix} 0.07\ 0.05\ 0.04\ 0.05 \\ 0.04\ 0.03\ 0.04 \\ 0.04\ 0.05 \\ 0.08 \end{pmatrix}$, $\begin{pmatrix} 0.07\ 0.05\ 0.05\ 0.03 \\ 0.04\ 0.06\ 0.05 \\ 0.04\ 0.06 \\ 0.08 \end{pmatrix}$, $\begin{pmatrix} 0.08\ \mathbf{0.21}\ \mathbf{0.27}\ \mathbf{0.69} \\ 0.03\ 0.06\ \mathbf{0.29} \\ 0.03\ \mathbf{0.22} \\ 0.08 \end{pmatrix}$
μ_i	[0.08, 0.13, 0.14, 0.12, 0.07]
$\ell(w)$	0.05
ν_i	[**0.44**, **0.28**, 0.18, **0.28**, **0.45**]
λ_i	[**0.23**, 0.14, 0.05, 0.13, **0.23**]

Table 9 Average gradient of three-layer NN for non-empty

Feature	Corresponding gradient(s)
(x_{ij}), (y_{ij}^{-1}), (η_{ij})	$\begin{pmatrix} 0.09\ 0.04\ 0.03\ 0.02 \\ 0.05\ 0.04\ 0.03 \\ 0.05\ 0.04 \\ 0.09 \end{pmatrix}$, $\begin{pmatrix} 0.09\ 0.04\ 0.03\ 0.03 \\ 0.06\ 0.04\ 0.03 \\ 0.06\ 0.04 \\ 0.09 \end{pmatrix}$, $\begin{pmatrix} 0.16\ 0.14\ \mathbf{0.25}\ \mathbf{0.30} \\ 0.14\ \mathbf{0.20}\ \mathbf{0.26} \\ 0.14\ 0.14 \\ 0.16 \end{pmatrix}$
μ_i	[0.09, 0.18, **0.21**, 0.19, 0.09]
$\ell(w)$	0.06
ν_i	[0.14, 0.10, 0.09, 0.11, 0.14]
λ_i	[0.17, 0.16, 0.16, 0.16, 0.18]

learning model (cf. Sect. 3.2.1), as well as the underlying problem: Since the target function f is discrete, the gradient of the smooth approximation $\hat{f}$ at individual points is hard to interpret. Nonetheless, this lack of explanation can inspire future mathematical research, directing our attention to get a more precise understanding of the underlying mathematical problem.

6.3 Experiments on the Dimension

Data: All $w = xt^{\mu}y \in W_a$ and $[b] \in B(\mathrm{SL}_5)$, with the conditions $\ell(w) < 30$ and $X_w(b) \neq \emptyset$. (Dataset size: 3,119,946)

Feature:

$$X = [x_{ij}, \mu_1, \ldots, \mu_5, y_{ij}^{-1}, \eta_{ij}, \ell(w), \nu_1, \ldots, \nu_5, \lambda_1, \ldots, \lambda_5] \in \mathbb{R}^{46}.$$

Output: $Y = \dim X_w(b)$

Result: Two models were utilized in our experiments, each executed 100 times. The first model, a single-layer neural ReLU classification network with ten neurons, achieved an average training accuracy of 77.02% and an average training error of 0.33, an average test accuracy of 77.06% and an average test error of 0.33. The average gradients can be found in Table 10. The second model, a three-layer neural ReLU classification network with 20 neurons per layer, reached an average training accuracy

Table 10 Average gradient of one-layer NN for dimension

Feature	Corresponding gradient(s)
(x_{ij}), (y_{ij}^{-1}), (η_{ij})	$\begin{pmatrix} 0.22 & 0.13 & 0.05 & 0.04 \\ & 0.26 & 0.13 & 0.05 \\ & & 0.26 & -0.13 \\ & & & 0.22 \end{pmatrix}$, $\begin{pmatrix} 0.18 & 0.16 & 0.0.08 & 0.03 \\ & 0.20 & 0.18 & 0.08 \\ & & 0.20 & 0.16 \\ & & & 0.18 \end{pmatrix}$, $\begin{pmatrix} 0.21 & 0.26 & 0.32 & 0.37 \\ & 0.18 & 0.25 & 0.31 \\ & & 0.18 & 0.26 \\ & & & 0.21 \end{pmatrix}$
μ_i	[0.34, 0.20, 0.06, 0.20, 0.35]
$\ell(w)$	0.46
ν_i	[**4.19**, **2.08**, 0.05, **2.07**, **4.19**]
λ_i	[**2.01**, **1.00**, 0.05, **1.01**, **2.02**]

Table 11 Average gradient of three-layer NN for dimension

Feature	Corresponding gradient(s)
(x_{ij}), (y_{ij}^{-1}), (η_{ij})	$\begin{pmatrix} 0.18 & 0.12 & 0.08 & 0.08 \\ & 0.24 & 0.13 & 0.08 \\ & & 0.24 & 0.12 \\ & & & 0.18 \end{pmatrix}$, $\begin{pmatrix} 0.16 & 0.12 & 0.08 & 0.18 \\ & 0.21 & 0.13 & 0.08 \\ & & 0.22 & 0.11 \\ & & & 0.16 \end{pmatrix}$, $\begin{pmatrix} 0.40 & 0.41 & 0.45 & 0.47 \\ & 0.42 & 0.42 & 0.46 \\ & & 0.41 & 0.41 \\ & & & 0.40 \end{pmatrix}$
μ_i	[0.37, 0.32, 0.31, 0.31, 0.37]
$\ell(w)$	0.47
ν_i	[**4.22**, **2.03**, 0.29, **2.03**, **4.22**]
λ_i	[**2.01**, **1.02**, 0.14, **1.00**, **2.00**]

of 92.12% and an average training error of 0.16, an average test accuracy of 92.10% and an average test error of 0.16. The average gradients are detailed in Table 11.

Analysis. It can be observed that the average gradients for these neural networks closely align with the gradient of the linear model discussed in Sect. 4. In fact, the dimension equals the virtual dimension for 64.8% of the dataset, and for most of the remaining data points, the difference is just 1. The gradient of this linear function seems to dominate the more nuanced behavior of the neural network, resulting in the 92.1% accuracy. To obtain a more insightful average gradient, a comparison between the dimension and the virtual dimension should be considered.

6.4 Experiments on the Condition Virtual dim. = dim.

Data: All $w = xt^{\mu}y \in W_a$ and $[b] \in B(\mathrm{SL}_5)$, with the conditions $\ell(w) < 30$ and $X_w(b) \neq \emptyset$. (Dataset size: 3,119,946)

Feature:

$$X = [x_{ij}, \mu_1, \ldots, \mu_5, y_{ij}^{-1}, \eta_{ij} = \delta(\eta(w)(\alpha_{ij})), \ell(w), \nu_1, \ldots, \nu_5, \lambda_1, \ldots, \lambda_5] \in \mathbb{R}^{46}.$$

Output: $Y = \begin{cases} 1 & \text{if } VD \neq \dim X_w(b) \\ -1 & \text{if } VD = \dim X_w(b) \end{cases}$

Result: Three models were utilized in our experiments, each executed 100 times. The first model, an SVM, achieved an average accuracy of 83.13%, with the average

Table 12 Average coefficient of SVM for VD = Dim

Feature	Corresponding coefficient(s)
(x_{ij}), (y_{ij}^{-1}), (η_{ij})	$\begin{pmatrix} -0.55 & -0.10 & 0.21 & 0.36 \\ & -0.87 & -0.13 & 0.22 \\ & & -0.87 & -0.10 \\ & & & -0.55 \end{pmatrix}$, $\begin{pmatrix} 0.39 & 0.28 & -0.02 & -0.27 \\ & 0.67 & 0.32 & -0.02 \\ & & 0.67 & 0.27 \\ & & & 0.40 \end{pmatrix}$, $\begin{pmatrix} \mathbf{1.18} & 0.91 & 0.70 & 0.73 \\ & \mathbf{1.57} & 0.95 & 0.70 \\ & & \mathbf{1.57} & 0.91 \\ & & & \mathbf{1.18} \end{pmatrix}$
μ_i	[0.02, −0.14, −0.00, 0.15, −0.02]
$\ell(w)$	−0.19
ν_i	[**1.02**, 0.44, 0.00, −0.45, −**1.02**]
λ_i	[−0.24, −0.10, 0.00, 0.10, 0.24]

Table 13 Average gradient of one-layer NN for VD = Dim

Feature	Corresponding gradient(s)
(x_{ij}), (y_{ij}^{-1}), (η_{ij})	$\begin{pmatrix} 0.17 & 0.06 & 0.07 & 0.11 \\ & \mathbf{0.38} & 0.07 & 0.07 \\ & & \mathbf{0.44} & 0.06 \\ & & & 0.18 \end{pmatrix}$, $\begin{pmatrix} 0.12 & 0.07 & 0.04 & 0.09 \\ & 0.21 & 0.08 & 0.04 \\ & & 0.21 & 0.08 \\ & & & 0.13 \end{pmatrix}$, $\begin{pmatrix} \mathbf{0.38} & 0.32 & 0.22 & 0.22 \\ & \mathbf{0.57} & 0.30 & 0.23 \\ & & \mathbf{0.56} & 0.32 \\ & & & \mathbf{0.39} \end{pmatrix}$
μ_i	[0.06, 0.16, 0.24, 0.16, 0.06]
$\ell(w)$	0.06
ν_i	[**0.37**, 0.18, 0.18, 0.18, **0.38**]
λ_i	[0.12, 0.06, 0.06, 0.06, 0.12]

Table 14 Average gradient of three-layer NN for VD = Dim

Feature	Corresponding gradient(s)
(x_{ij}), (y_{ij}^{-1}), (η_{ij})	$\begin{pmatrix} 0.13 & 0.06 & 0.04 & 0.06 \\ & \mathbf{0.35} & 0.07 & 0.04 \\ & & \mathbf{0.31} & 0.06 \\ & & & 0.12 \end{pmatrix}$, $\begin{pmatrix} 0.10 & 0.04 & 0.04 & 0.05 \\ & 0.20 & 0.05 & 0.04 \\ & & 0.19 & 0.04 \\ & & & 0.10 \end{pmatrix}$, $\begin{pmatrix} 0.27 & 0.27 & 0.24 & 0.22 \\ & \mathbf{0.39} & 0.25 & 0.24 \\ & & \mathbf{0.38} & 0.28 \\ & & & 0.25 \end{pmatrix}$
μ_i	[0.08, 0.23, **0.34**, 0.22, 0.07]
$\ell(w)$	0.05
ν_i	[**0.31**, 0.21, 0.20, 0.20, **0.31**]
λ_i	[0.12, 0.07, 0.07, 0.07, 0.12]

coefficients shown in Table 12. The second model, a single-layer neural ReLU classification network with ten neurons, recorded an average training accuracy of 87.52% and an average test accuracy of 87.55%, with the average gradients shown in Table 13. The third model, a three-layer neural ReLU classification network with 20 neurons per layer, attained an average training accuracy of 96.66% and an average test accuracy of 96.67%, with the average gradients presented in Table 14.

Analysis. We see that $\eta_{i,j}$ are important features. We know that if $\eta(w)$ is small, for example, if $\eta(w)$ is a partial Coxeter element, then the dimension equals virtual dimension [30]. If $\eta(w)$ is large and close to the longest element, then dimension is unlikely to be equal to virtual dimension.

Table 15 Number of elements with dim = virdim fail

$\ell(\eta(w))$	0	1	2	3	4	5	6	7	8	9	10
# non-cordial	0	0	0	2696	8316	18,232	18,152	17,651	10,039	5284	1175
# cordial	1271	5742	11,191	21,255	24,754	31,172	24,780	21,292	11,155	5664	1225
Proportion	0	0	0	11%	25%	37%	42%	45%	47%	48%	49%

Moreover, it is known if $y = 1$ or, under additional hypotheses, if $x = w_0$, then dimension must also be equal to virtual dimension [46]. This explains the signs of the x_{ij} and y_{ij} in Table 12.

It is known in general, cf. (7.1), that the difference $d_w(b) - \dim X_w(b)$ is maximal for large $[b]$. This explains why the Newton point is an important feature.

Table 15 gives the proportion of elements w, grouped by the length of $\eta(w)$, where the virtual dimension is not equal to dimension for some b (i.e., the non-cordial elements in the sense of [46]).

6.5 Statistics of the Difference of Virtual dim. and dim.

The experiments and analysis above indicate that the virtual dimension is a good approximation of the dimension for non-empty $X_w(b)$. A natural question is to further study the difference between the dimension and virtual dimension for non-empty $X_w(b)$.

In this part, we will provide a numerical analysis of the the difference between the dimension and virtual dimension $\Delta_w(b) = d_w(b) - \dim X_w(b)$ for non-empty $X_w(b)$ of dataset 2, and hope that the analysis will guide us to get the pattern of difference.

As exhibited in Table 16, within the dataset, the virtual dimension represents the theoretical upper bound of the actual dimension. The maximum achievable accuracy rate was 65.82%. Furthermore, the predominant $\Delta_w(b)$ was of magnitude 1.

It seems that the percentage of pairs (w, b) with $\Delta_w(b)$ larger than a given bound decreases rapidly. We further expect that $\Delta_w(b)$ might be bounded from above by a constant depending only on n, i.e., the group SL_n. We will investigate this question by mathematical methods in Sect. 7.

6.6 Experiments on the Irreducible Components

In this section, we investigate the number of top-dimensional irreducible components of $X_w(b)$ up to the $\mathbf{J}_b$-action. The analogous question has been solved for affine

Table 16 $\Delta_w(b)$

	$\Delta_w(b) = 0$	$\Delta_w(b) = 1$	$\Delta_w(b) = 2$	$\Delta_w(b) = 3$	$\Delta_w(b) = 4$	All
Amount of data	2,020,909	922,482	166,386	9885	284	3,119,946
Proportion of data (%)	65	30	5	0.3	0.01	100

Deligne–Lusztig varieties in the affine Grassmannian [49, 69]. Here, the key ingredient is the dimension of the *weight space of the highest-weight Verma module* $V_\mu(\lambda)$. This is a commonly studied object in the representation theory of Lie algebras, and we refer the reader to any of the corresponding textbooks for the exact definition. For now, we remark that this dimension is a positive integer that is computable in terms of $\mu \in X_*(T)$ (which depends only on w) and $\lambda(b) \in X_*(T)$ (which only depends on b). Our first experiment uses the data set from the previous experiment, restricted to those ADLV whose dimension agrees with virtual dimension.

(1) **Data:** All $w = xt^\mu y \in W_a$ and $[b] \in B(\mathrm{SL}_5)$ with $\ell(w) < 30$ and $\dim X_w(b) = d_w(b)$. (Dataset size: 2,020,909)

Feature:

$$X = [x_{ij}, \mu_1, \ldots, \mu_5, y_{ij}^{-1}, \eta_{ij}, \ell(w), \nu_1, \ldots, \nu_5, \lambda_1, \ldots, \lambda_5, \dim V_\mu(\lambda)] \in \mathbb{R}^{47}.$$

Output: $Y = \sharp \mathbf{J}_b \backslash \Sigma^{\mathrm{top}} X_w(b)$

Result: Two models were utilized in our experiments, each executed 100 times. The first model, a single-layer neural ReLU classification network with 10 neurons, achieved an average training accuracy of 67.15% and an average training error of 0.48, an average test accuracy of 67.06% and an average test error of 0.48. The average gradients can be found in Table 17. The second model, a three-layer neural ReLU

Table 17 Average gradient of one-layer NN for irreducible components

Feature	Corresponding gradient(s)
$(x_{ij}), (y_{ij}^{-1}), (\eta_{ij})$	$\begin{pmatrix} 0.23\ 0.13\ 0.07\ 0.21 \\ 0.29\ 0.12\ 0.07 \\ 0.29\ 0.13 \\ 0.23 \end{pmatrix}, \begin{pmatrix} 0.25\ 0.14\ 0.04\ 0.19 \\ 0.29\ 0.15\ 0.04 \\ 0.29\ 0.14 \\ 0.25 \end{pmatrix}, \begin{pmatrix} 0.34\ 0.26\ 0.35\ \mathbf{0.62} \\ 0.41\ \mathbf{0.54}\ 0.35 \\ 0.43\ 0.26 \\ 0.34 \end{pmatrix}$
μ_i	[0.04, 0.11, 0.06, 0.11, 0.04]
$\ell(w)$	0.08
ν_i	[**0.56**, 0.30, 0.03, 0.30, **0.55**]
λ_i	[0.25, 0.13, 0.02, 0.13, 0.25]
$\dim V_\mu(\lambda)$	0.02

Table 18 Average gradient of three-layer NN for irreducible components

Feature	Corresponding gradient(s)
$(x_{ij}), (y_{ij}^{-1}), (\eta_{ij})$	$\begin{pmatrix} 0.25\ 0.16\ 0.10\ 0.17 \\ 0.31\ 0.15\ 0.10 \\ 0.31\ 0.16 \\ 0.25 \end{pmatrix}, \begin{pmatrix} 0.25\ 0.14\ 0.08\ 0.16 \\ 0.30\ 0.16\ 0.08 \\ 0.31\ 0.14 \\ 0.26 \end{pmatrix}, \begin{pmatrix} 0.54\ 0.43\ 0.63\ 0.55 \\ \mathbf{0.69}\ \mathbf{0.87}\ 0.62 \\ \mathbf{0.72}\ 0.42 \\ 0.51 \end{pmatrix}$
μ_i	[0.08, 0.28, 0.33, 0.27, 0.08]
$\ell(w)$	0.07
ν_i	[0.50, 0.30, 0.15, 0.30, 0.50]
λ_i	[0.24, 0.13, 0.06, 0.13, 0.24]
$\dim V_\mu(\lambda)$	0.03

classification network with 20 neurons per layer, reached an average training accuracy of 75.96% and an average training error of 0.33, an average test accuracy of 75.92%, and an average test error of 0.33. The average gradients are shown in Table 18.

Analysis. We observe that the variables ν_1, ν_5, η_{15}, η_{23}, η_{24}, η_{34}, μ_3 play a sensitive role in the approximated function $\hat{f}$. Interestingly, the simpler model tends to overlook η_{23}, η_{24}, η_{34}, μ_3, while these variables appear to be crucial for the more complex model. This suggests that the function f exhibits a complex relationship with respect to η_{23}, η_{24}, η_{34}, μ_3. Moreover, the accuracy achieved in this experiment is lower than that of the previous one, which focused on the dimension problem. This suggests that the problem of determining irreducible components presents greater complexity.

We remark that the feature $\dim V_\mu(\lambda)$ does not seem to be particularly influential. Repeating the experiment without this feature, we retain almost the same accuracy. To obtain further insight into the most well-behaved situations, we restrict our attention to a certain subset of elements, which are known to enjoy the most convenient properties.

(2) **Data:** All $w = xt^\mu y \in W_a$ where $y = 1$ and $[b] \in B(\mathrm{SL}_5)$ with $\ell(w) < 30$ and $X_w(b) \neq \emptyset$. In this case, it is known that $\dim X_w(b) = d_w(b)$. (Dataset size: 43,986)

Feature:

$$X = [x_{ij}, \mu_1, \ldots, \mu_5, \ell(w), \nu_1, \ldots, \nu_5, \lambda_1, \ldots, \lambda_5, \dim V_\mu(\lambda)] \in \mathbb{R}^{47}.$$

Output: $Y = \sharp \mathbf{J}_b \backslash \Sigma^{\mathrm{top}} X_w(b)$

Result: Two models were utilized in our experiments, each executed 100 times. The first model, a single-layer neural ReLU classification network with ten neurons, achieved an average training accuracy of 67.82% and an average training error of 0.49, an average test accuracy of 67.67%, and an average test error of 0.49. The average gradients can be found in Table 19. The second model, a three-layer neural ReLU classification network with 20 neurons per layer, reached an average training accuracy of 81.80% and an average training error of 0.26, an average test accuracy of 81.60%, and an average test error of 0.27. The average gradients are shown in Table 20.

Analysis. The restriction to $y = 1$ implies that $X_w(b)$ is equidimensional $\dim X_w(b) = d_w(b)$ whenever $X_w(b) \neq \emptyset$ [45]. In other words, all irreducible components are top-dimensional.

We see that the accuracy of the single-layer neural network does not change much compared to the previous experiment. However, the gradients are vastly different.

Table 19 Average gradient of one-layer NN for irreducible components

Feature	Corresponding gradient(s)
(x_{ij})	$\begin{pmatrix} 0.21\ 0.20\ 0.25\ \mathbf{0.29} \\ 0.16\ \mathbf{0.27}\ 0.25 \\ 0.17\ 0.20 \\ 0.20 \end{pmatrix}$
μ_i	[0.11, 0.13, 0.09, 0.12, 0.10]
$\ell(w)$	0.04
ν_i	[0.12, 0.10, 0.05, 0.10, 0.11]
λ_i	[0.07, 0.05, 0.03, 0.04, 0.07]
$\dim V_\mu(\lambda)$	**0.29**

Table 20 Average gradient of three-layer NN for irreducible components

Feature	Corresponding gradient(s)
(x_{ij})	$\begin{pmatrix} 0.21 & 0.20 & 0.23 & 0.20 \\ 0.21 & \mathbf{0.26} & 0.23 & \\ 0.21 & 0.20 & & \\ 0.21 & & & \end{pmatrix}$
μ_i	[0.10, 0.19, **0.25**, 0.20, 0.11]
$\ell(w)$	0.02
ν_i	[0.09, 0.09, 0.13, 0.09, 0.09]
λ_i	[0.05, 0.04, 0.05, 0.04, 0.05]
$\dim V_\mu(\lambda)$	**0.28**

The biggest contribution in Table 19 comes from the dimension of the weight space $\dim V_\mu(\lambda)$, which only made a tiny contribution in the previous experiment. In the case of three-layer neural networks, the restriction to $y = 1$ brings a substantial improvement in accuracy, and again, the contribution of $\dim V_\mu(\lambda)$ becomes significantly larger.

If $x = w_0$ is the longest element of the Weyl group, we know that the irreducible components of $X_w(b)$ correspond one-to-one to the irreducible components of the affine Deligne–Lusztig variety $X_\mu(b)$ inside the affine Grassmannian (following the proof of [25, Theorem 10.1]). For the latter kind of affine Deligne–Lusztig varieties, the number of $\mathbf{J}_b$-orbits of irreducible components has been predicted by Chen–Zhu, and their conjecture has been fully established by Zhou–Zhu [69] as well as Nie [49]. In this case, we know that $Y = \dim V_\mu(\lambda)$.

More generally, the same conclusion holds whenever x is the longest element of a Levi subgroup, following the Hodge–Newton decomposition method of Görtz–He–Nie [19]. For general x, we may expect that the number of irreducible components is much smaller. Nonetheless, it should not be a surprise that $\dim V_\mu(\lambda)$ is the input feature with the highest overall contribution, as measured by the average gradient. It is not quite clear why this was not the case in the previous experiment, since these two cases are related by the partial conjugation method, which is independent of μ. This could be an artifact of our limited data set. However, the situation remains overall mysterious, and we invite the interested readers to further explore this phenomenon through mathematical insight or ML-assisted research.

Overall, we may summarize that the problem of enumerating irreducible components allows for a fairly accurate solutions using single-layer or three-layer neural networks. This gives hope that further mathematical progress on this problem should be possible. Moreover, the second subset does indeed seem to be better behaved for studying this problem.

7 Lower Bound on the Dimension

In Sect. 6, we developed machine learning models that not only enable us to recover previously known results, but also lead to new conjectures and research questions. In this section, we study the question on the lower bound of the dimension of ADLV

whenever it is non-empty. In other words, we study the upper bound of the difference $d_w(b) - \dim X_w(b)$ whenever $X_w(b) \neq \emptyset$. For $\mathbf{G} = \mathrm{SL}_n$, we will show that

$$\max_{X_w(b)\neq\emptyset} d_w(b) - \dim X_w(b) = \begin{cases} k(k-1), & n = 2k, \\ k^2, & n = 2k+1. \end{cases}$$

Since the dimension of ADLV is of general interest, we establish such a lower bound in the most general case, i.e., we no longer specialize to $\mathbf{G} = \mathrm{SL}_n$.

7.1 General Setup

Let F be a non-archimedean local field with residue field $\mathbb{F}_q$ and let $\breve{F}$ be the completion of the maximal unramified extension of F. We write Γ for $\mathrm{Gal}(\overline{F}/F)$, Γ_0 for the inertia subgroup of Γ and $\sigma \in \Gamma$ for the Frobenius. Let $\mathbf{G}$ be a connected reductive group over F and $\breve{G} = \mathbf{G}(\breve{F})$. Let σ be the Frobenius morphism on $\breve{G}$. We fix a σ-stable Iwahori subgroup $\breve{I}$ of $\breve{G}$. Let $Fl = \breve{G}/\breve{I}$ be the *affine flag variety*. Let $\tilde{W}$ be the Iwahori–Weyl group of $\breve{G}$. Then, we have a natural identification $\breve{I}\backslash\breve{G}/\breve{I} \cong \tilde{W}$ and the σ-action on $\breve{G}$ induces a natural action on $\tilde{W}$, which we still denote by σ. The extended affine Weyl group $\tilde{W}$ is the semidirect product of the finite Weyl group W_0 and the Γ_0-coinvariants of the cocharacter lattice $X_*(T)_{\Gamma_0}$.

For any $b \in \breve{G}$ and $w \in \tilde{W}$, we define the corresponding *affine Deligne–Lusztig variety* in the affine flag variety

$$X_w(b) = \{g\breve{I} \in \breve{G}/\breve{I};\, g^{-1}b\sigma(g) \in \breve{I}\dot{w}\breve{I}\} \subset Fl.$$

It is known that the affine Deligne–Lusztig variety $X_w(b)$ is a (probably empty) locally closed (perfect) scheme of locally finite type over the residue field of $\breve{F}$. It is a general fact that one may reduce all questions regarding the geometry of affine Deligne–Lusztig varieties to the case where the group $\mathbf{G}$ is quasi-split and of adjoint type [19, Sect. 2]. Hence, we assume from now on that $\mathbf{G}$ satisfies these assumptions. In particular, this means that the finite Weyl group W_0 is stable under the action of σ.

We denote the Borovoi fundamental group of $\mathbf{G}$ to be $\pi_1(\mathbf{G}) = X_*(T)/\mathbb{Z}\Phi^\vee$, where $\Phi^\vee$ is the set of coroots. Then, the Kottwitz point of $[b] \in B(\mathbf{G})$ is denoted by $\kappa(b) \in \pi_1(\mathbf{G})_\Gamma$, which characterizes the connected components of the affine flag variety up to σ-action.

Write $w = xt^\mu y$ with $x, y \in W_0$, $\mu \in X_*(T)_{\Gamma_0}$ and $t^\mu y \in {}^{\mathbb{S}}\tilde{W}$. Then, we put $\eta_\sigma(w) = \sigma^{-1}(y)x$. Let $\nu_b \in X_*(T)_{\Gamma_0} \otimes \mathbb{Q}$ denote the dominant Newton point, $\mathrm{def}(b) \in \mathbb{Z}_{\geq 0}$ the defect and $2\rho \in X^*(T)^\Gamma$ the sum of the positive roots. Then, we define the *virtual dimension* of the pair $(w, [b])$ to be

$$d_w(b) = \frac{1}{2}\big(\ell(w) + \ell(\eta_\sigma(w)) - \langle \nu_b, 2\rho\rangle - \mathrm{def}(b)\big).$$

Here, we write ℓ for the length function of $\tilde{W}$ and W_0, and write ℓ_R for the *reflection length* on W_0. The reflection length of an element $u \in W_0$ is defined to be the smallest

number n, such that u is a product of n reflections (not necessarily simple) in W_0. It is denoted by

$$\ell_R(u) = \min\{n \mid \exists \alpha_1, \ldots, \alpha_n \in \Phi \text{ such that } u = s_{\alpha_1} \cdots s_{\alpha_n}\}.$$

Write $\mathcal{O} = \{w^{-1} w_0 \sigma(w) \mid w \in W_0\}$ for the σ-conjugacy class of the longest element w_0 and $\ell_R(\mathcal{O}) = \min\{\ell_R(u) \mid u \in \mathcal{O}\}$. For a complete description of $\ell_R(\mathcal{O})$, we refer to [31].

Set $B(\mathbf{G})_w = \{[b] \in B(\mathbf{G});\ X_w(b) \neq \emptyset\}$ and let $[b] \in B(\mathbf{G})_w$. We know that $\dim X_w(b) \leq d_w(b)$, and the goal of this section is to give a bound of the difference $d_w(b) - \dim X_w(b)$.

Theorem 2 *The maximum of the difference between virtual dimension and dimension, for all pairs $(w, [b])$ such that the corresponding affine Deligne–Lusztig variety is non-empty, is precisely given by*

$$\max_{\substack{w \in \tilde{W} \\ [b] \in B(\mathbf{G})_w}} d_w(b) - \dim X_w(b) = \frac{\ell(w_0) - \ell_R(\mathcal{O})}{2}.$$

We summarize our proof strategy as follows. There is a unique maximal element in $B(\mathbf{G})_w$, denoted by $[b_w]$. It is called the *generic* σ-conjugacy class of w, since it is the unique σ-conjugacy class, such that the intersection $[b_w] \cap \check{I} w \check{I}$ is dense in $\check{I} w \check{I}$. The existence of $[b_w]$ and a useful combinatorial characterization are obtained by Viehmann in [61, Corollary 5.6].

By a deep result in arithmetic geometry, we will see that the difference of virtual dimension and dimension reaches its maximum, over $B(\mathbf{G})_w$, at $[b] = [b_w]$.

The advantage of working with $[b_w]$ is that we have an explicit formula for the dimension of $X_w(b_w)$. Combined with a description of $[b_w]$ via a certain combinatorial model, we then compute the difference $d_w(b_w) - \dim X_w(b_w)$. Finally, we re-write that upper bound in terms of the length and reflection length functions of the finite Weyl group W_0.

For $\mathbf{G} = \mathrm{SL}_n$, one evaluates $\ell(w_0) = n(n-1)/2$ and $\ell_R(w_0) = \lfloor (n-1)/2 \rfloor$ to obtain the upper bound as stated in the beginning of this section.

7.2 Step 1: A Purity Result

We denote the usual dominance order on $B(\mathbf{G})$ by $\leq$. For $[b], [b'] \in B(\mathbf{G})$, this means that $[b] \leq [b']$ if and only if the Kottwitz points $\kappa(b), \kappa(b')$ agree as elements of $\pi_1(G)_\Gamma$ and the difference of Newton points $\nu_{b'} - \nu_b$ is a $\mathbb{Q}_{\geq 0}$-linear combination of positive coroots. Geometrically, this means that the subset $[b] \subset \check{G}$ lies inside the closure of $[b'] \subset \check{G}$.

One now may study increasing chains $[b] = [b_1] < [b_2] < \cdots < [b_{n+1}] = [b']$ in $B(\mathbf{G})$. By the work of Chai [10, Theorem 7.4], we know that all maximal chains have

the same length, given by

$$\text{length}([b],[b']) = \frac{1}{2}\big(\langle \nu_{b'} - \nu_b, 2\rho\rangle + \text{def}(b) - \text{def}(b')\big).$$

One may similarly ask for geometric properties of σ-conjugacy classes not inside $\breve{G}$, but inside the smaller subset $\breve{I}w\breve{I}$. The intersection $[b] \cap \breve{I}w\breve{I}$ is infinite-dimensional, but it is admissible in the sense of [26], so there is a well-defined notion of codimension in $\breve{I}w\breve{I}$ and closure inside $\breve{I}w\breve{I}$. There is a notion of relative dimension of $[b] \cap \breve{I}w\breve{I}$, allowing us to express the dimension of $X_w(b)$ in terms of $\dim [b] \cap \breve{I}w\breve{I}$.

For $[b] \in B(\mathbf{G})_w$, the closure of the Newton stratum $[b] \cap \breve{I}w\breve{I}$ is contained in the union of all Newton strata $[b'] \cap \breve{I}w\breve{I}$ for $[b] \geq [b'] \in B(\mathbf{G})_w$, but one cannot in general expect to have an equality.

To compare the dimensions of different Newton strata inside $\breve{I}w\breve{I}$, we use the purity theorem. This is a deep result in arithmetic geometry, developed by many experts, including de Jong, Viehmann, and Hamacher. We will not recall the statement nor the proof, due to the level of technicalities involved, and instead refer to the discussion in [62].

The statement we use here is due to Viehmann [62, Lemma 5.12]. It states that the codimension of $[b] \cap \breve{I}w\breve{I}$ inside $\breve{I}w\breve{I}$ is at most $\text{length}([b],[b_w])$. By [26, Theorem 2.23], we know that this codimension is equal to $\dim X_w(b_w) - \dim X_w(b) + \langle \nu_{b_w} - \nu_b, 2\rho\rangle$. Thus

$$\begin{aligned}
\dim X_w(b_w) - \dim X_w(b) &\leq \text{length}([b],[b_w]) - \langle \nu_{b_w} - \nu_b, 2\rho\rangle \\
&= \frac{1}{2}\big(\langle \nu_{b_w} - \nu_b, 2\rho\rangle + \text{def}(b) - \text{def}(b_w)\big) - \langle \nu_{b_w} - \nu_b, 2\rho\rangle \\
&= \frac{1}{2}\big(\langle \nu_b - \nu_{b_w}, 2\rho\rangle + \text{def}(b) - \text{def}(b_w)\big) \\
&= d_w(b_w) - d_w(b). \qquad (7.1)
\end{aligned}$$

Hence, $d_w(b) - \dim X_w(b) \leq d_w(b_w) - \dim X_w(b_w)$. In other words, the function

$$B(\mathbf{G})_w \to \mathbb{Z}, \quad [b] \mapsto d_w(b) - \dim X_w(b)$$

reaches its maximum at $[b] = [b_w]$. We now focus on this special case.

7.3 Step 2: The Quantum Bruhat Graph

Let $w \in \tilde{W}$. Recall that $[b_w] \in B(G)_w$ denotes the generic σ-conjugacy class associated with w. In this step, we calculate the difference $d_w(b_w) - \dim X_w(b_w)$ for arbitrary elements $w \in \tilde{W}$.

It is known that $\dim X_w(b_w) = \ell(w) - \langle \nu_{b_w}, 2\rho\rangle$, cf. [26, Theorem 2.23]. Thus, we compute

$$d_w(b_w) - \dim X_w(b_w) = \frac{1}{2}\big(-\ell(w) + \ell(\eta_\sigma(w)) + \langle \nu_{b_w}, 2\rho\rangle - \text{def}(b_w)\big).$$

By [56, Proposition 3.9], we know that

$$\langle \nu_b, 2\rho \rangle - \mathrm{def}(b) = \langle \lfloor b \rfloor, 2\rho \rangle,$$

where $\lfloor b \rfloor \in X_*(T)_\Gamma$ is the best integral approximation of $[b]$ in the sense of [21]. More specifically, $\lfloor b \rfloor$ is the unique element in $X_*(T)_\Gamma$, such that

- $\kappa(b) = \kappa(\lfloor b \rfloor)$ in $\pi_1(\mathbf{G})_\Gamma$ and
- $0 \leq \langle \nu_b - \lfloor \nu_b \rfloor, \omega_o \rangle < 1$ for any $o \in \mathbb{S}/\langle\sigma\rangle$. Here, $\omega_o \in \mathbb{Q}\Phi$ is the unique weight whose pairing with a simple root α is given by 1 if $s_\alpha \in o$ and 0 otherwise.

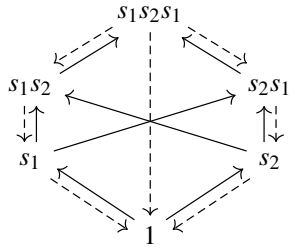
It remains to compute this approximation $\lfloor b_w \rfloor$ for arbitrary elements $w \in \tilde{W}$. This is a result of Schremmer [56, Theorem 4.2], generalizing earlier results which compute this quantity in special cases.

To understand these generic σ-conjugacy class $[b_w]$, the tool of choice for Schremmer's result and its predecessors is a finite combinatorial object associated with $\mathbf{G}$, known as *quantum Bruhat graph*. This graph was originally introduced by Brenti, Fomin, and Postnikov [5] as a consequence of certain solutions to Yang–Baxter equations. While originally intended to study quantum cohomology, especially the quantum Chevalley–Monk formula, it has since been found useful in a number of contexts, such as Kirillov–Reshetikhin crystals [39] and Bruhat order of affine Weyl groups [38]. The calculation of the generic σ-conjugacy class of affine Weyl group elements is related to the Bruhat order via a result of Viehmann [61, Corollary 5.6]. The resulting connection between the quantum Bruhat graph and the generic σ-conjugacy class of sufficiently regular affine Weyl group elements was first discovered by Milićević [45].

The quantum Bruhat graph is defined as follows: By definition, $\mathrm{QBG}(\Phi)$ is a directed graph, whose set of vertices is given by the finite set W_0. Its edges are of the form $w \to ws_\alpha$ for $w \in W_0$ and $\alpha \in \Phi^+$ whenever one of the following conditions is satisfied:

- $\ell(ws_\alpha) = \ell(w) + 1$ or
- $\ell(ws_\alpha) = \ell(w) + 1 - \langle \alpha^\vee, 2\rho \rangle$.

Edges satisfying the first condition are called *Bruhat* edges, whereas edges satisfying the second condition are called *quantum* edges (hence the graph's name). It is common to draw the graph with the vertical position of the vertices corresponding to the length, with the longest element on top and the identity element at the bottom. Then, the Bruhat edges go upwards, whereas the quantum edges go downwards. This is the quantum Bruhat graph of type A_2:



It is known that the quantum Bruhat graph is (strongly) connected. Hence we may write $d_{\mathrm{QBG}}(u, v) \in \mathbb{Z}_{\geq 0}$ for the length of a shortest path from u to v in the quantum Bruhat graph, where $u, v \in W_0$.

The description of the generic σ-conjugacy class $[b_w]$ in terms of w uses the quantum Bruhat graph and the decomposition $w = xt^{\mu}y$ as above. The latter decomposition is not canonical if w is not very regular, so we might have to vary it slightly. This can be done using the notion of length positive elements, as introduced by Schremmer [56].

Let $w = t^{\lambda}z \in \tilde{W}$. We say that $v \in W_0$ is *length positive* with respect to w if all positive roots $\alpha \in \Phi^+$ satisfy

$$\langle z^{-1}\lambda, v\alpha\rangle + \delta(zv\alpha) - \delta(v\alpha) \geq 0.$$

Denote the set of length positive elements by $\mathrm{LP}(w) \subseteq W_0$. If we write $w = xt^{\mu}y$ with $t^{\mu}y \in {}^{\mathbb{S}}\tilde{W}$ as above, then y^{-1} is always length positive, i.e., $y^{-1} \in \mathrm{LP}(w)$.

With this setup, we can summarize the main result of [56] as follows:

Theorem 3 [56, Theorem 4.2 and Lemma 4.4] *Let $w = t^{\lambda}z \in \tilde{W}$. Let $[b_w]$ the maximal element in $B(\mathbf{G})_w$. Then*

$$\langle \lfloor b_w \rfloor, 2\rho\rangle = \ell(w) - \min_{v \in \mathrm{LP}(w)} d_{\mathrm{QBG}}\big(v, \sigma(zv)\big).$$

In view of the above calculation, we obtain

$$d_w(b_w) - \dim X_w(b_w) = \frac{1}{2}\Big(\ell(y\sigma(x)) - \min_{v \in \mathrm{LP}(w)} d_{\mathrm{QBG}}\big(v, \sigma(zv)\big)\Big). \tag{7.2}$$

A priori, since there are infinitely many elements in $\tilde{W}$, it is not clear whether or not the left-hand side of the above equation has an upper bound. However, since W_0 is a finite group, the right-hand side (and thus also the left-hand side) of Eq. (7.2) has an upper bound.

7.4 Step 3: The Reflection Length as an Upper Bound

From Eqs. (7.1) and (7.2), we see that

$$\max_{[b] \in B(\mathbf{G})_w} d_w(b) - \dim X_w(b) = \frac{1}{2}\Big(\ell(y\sigma(x)) - \min_{v \in \mathrm{LP}(w)} d_{\mathrm{QBG}}\big(v, \sigma(zv)\big)\Big).$$

It remains to compute the maximum of this expression over all $w \in \tilde{W}$. A major difficulty in explicitly evaluating the right-hand side of the above expression is the condition $v \in \mathrm{LP}(w)$. In this section, we relax this condition to $v \in W_0$, thus obtaining the upper bound

$$\max_{[b] \in B(\mathbf{G})_w} d_w(b) - \dim X_w(b) \leq \frac{1}{2} \max_{x,y,v \in W}\Big(\ell(y\sigma(x)) - d_{\mathrm{QBG}}\big(v, \sigma(xyv)\big)\Big).$$

From the definition of the quantum Bruhat graph, we see

$$\min_{v\in W_0} d_{\mathrm{QBG}}\big(v,\sigma(xyv)\big) \geq \min_{v\in W_0} \ell_R\big(v^{-1}\sigma(xyv)\big).$$

We summarize

$$\max_{\substack{w\in\tilde{W}\\ [b]\in B(\mathbf{G})_w}} d_w(b) - \dim X_w(b) \leq \max_{v,x,y\in W_0} \frac{1}{2}\Big(\ell(y\sigma(x)) - \ell_R\big(v^{-1}\sigma(xyv)\big)\Big).$$

Writing $u = yv$ and $\eta = y\sigma(x)$, we can re-write this as

$$\begin{aligned}\cdots &= \max_{u,x,y\in W_0} \frac{1}{2}\Big(\ell(y\sigma(x)) - \ell_R\big(u^{-1}y\sigma(x)\sigma(u)\big)\Big)\\ &= \max_{u,\eta\in W_0} \frac{1}{2}\Big(\ell(\eta) - \ell_R\big(u^{-1}\eta\sigma(u)\big)\Big).\end{aligned}$$

If $\eta \neq w_0$, we find a simple reflection s with $\ell(\eta s) = \ell(\eta) + 1$. Then, certainly $\ell_R(u^{-1}\eta s\sigma(u)) \leq \ell_R(u^{-1}\eta\sigma(u)) + 1$. So when searching for the above maximum, we may replace η by ηs until $\eta = w_0$. Thus, we can simplify the above expression to

$$\cdots = \max_{u\in W_0} \frac{1}{2}\Big(\ell(w_0) - \ell_R\big(u^{-1}w_0\sigma(u)\big)\Big).$$

We proved that

$$\begin{aligned}\max_{\substack{w\in\tilde{W}\\ [b]\in B(\mathbf{G})_w}} d_w(b) - \dim X_w(b) &= \frac{1}{2}\Big(\ell(y\sigma(x)) - \min_{v\in \mathrm{LP}(w)} d_{\mathrm{QBG}}\big(v,\sigma(zv)\big)\Big)\\ &\leq \frac{\ell(w_0) - \ell_R(\mathcal{O})}{2},\end{aligned}$$

obtaining the upper bound claimed in Theorem 2.

The reader will find a peculiar similarity to the paper [31] of He and Yu. They study a similar maximization problem in [31, Lemma 4.3, Theorem 5.1], proving that

$$\max_{x,y\in W_0}\Big(\ell\big(\sigma^{-1}(y)x\big) - d_{\mathrm{QBG}}\big(x,y^{-1}\big)\Big) = \ell(w_0) - \ell_R(\mathcal{O}).$$

7.5 Step 4: Explicit Construction of the Lower Bound

We saw in the previous step that

$$\begin{aligned}\max_{\substack{w\in\tilde{W}\\ [b]\in B(\mathbf{G})_w}} d_w(b) - \dim X_w(b) &= \frac{1}{2}\Big(\ell(y\sigma(x)) - \min_{v\in \mathrm{LP}(w)} d_{\mathrm{QBG}}\big(v,\sigma(zv)\big)\Big)\\ &\leq \frac{\ell(w_0) - \ell_R(\mathcal{O})}{2}.\end{aligned}$$

In this section, we prove that equality holds, by explicitly constructing an element $w \in \tilde{W}$ and $v \in \mathrm{LP}(w)$, such that $\eta_\sigma(w) = y\sigma(x) = w_0$ and $d_{\mathrm{QBG}}\big(v, \sigma(xyv)\big) = \ell_R(\mathcal{O})$. We do this construction in a case-by-case fashion, first considering the case where $\mathbf{G}$ is quasi-simple over $\breve{F}$, meaning that the root system is connected.

7.5.1 The Split and $\sigma = \mathrm{Ad}(w_0)$ Cases

Consider first the case where the action of σ on Φ is either the identity map (i.e., $\mathbf{G}$ is residually split) or equal to the action of $-w_0$. In either case, we obtain $\ell_R(\mathcal{O}) = \ell_R(w_0)$. From [55, Sect. 5], we know that $\ell_R(w_0) = d_{\mathrm{QBG}}(w_0, 1)$. Then, $w := t^{2\rho^\vee} w_0$ satisfies $\eta_\sigma(w) = w_0$ and $\mathrm{LP}(w) = \{w_0\}$, from which one obtains $d_w(b_w) - \dim X_w(b_w) = \frac{\ell(w_0) - \ell_R(\mathcal{O})}{2}$, completing the proof of the theorem.

Following the classification of root systems, one sees that σ is given by one of the above choices *unless* $\mathbf{G}$ is non-split of type D_n with n even.

7.5.2 The Case ${}^2D_{2k}$

Suppose that $\mathbf{G}$ is of type D_{2k} with $k \geq 2$ and the image of σ in $\mathrm{Aut}(\Phi)$ has order 2. Label the simple roots as $\alpha_1, \ldots, \alpha_{2k}$ such α_i is connected to α_{i+1} in the Dynkin diagram of D_{2k} for all $i = 1, \ldots, 2k-2$. Then, σ interchanges the roots α_{2k} and α_{2k-1}, while fixing all other roots. The element $w_0 \in W_0$ is central. One computes $\ell_R(\mathcal{O}) = 2k-2$ [31, Sect. 5.7].

Define $x = s_{2k-1}$, $y = w_0 s_{2k}$, $v = w_0 \in W_0$ and $\mu \in X_*(T)_{\Gamma_0}$ as $\mu = 2\rho_K^\vee$, the sum of all positive coroots of the sub-root system spanned by $K = \{\alpha_1, \ldots, \alpha_{2k-1}\}$. Then, $w = xt^\mu y \in \tilde{W}$ satisfies $y = y^{-1} \in \mathrm{LP}(w)$ and $\ell(w, y\alpha_{2k}) = 0$, hence $v \in \mathrm{LP}(w)$. We get $\eta_\sigma(w) = y\sigma(x) = w_0$.

It suffices to show that $d_{\mathrm{QBG}}(v, \sigma(xyv)) \leq \ell_R(\mathcal{O}) = 2k-2$. For this, we compute

$$\begin{aligned} d_{\mathrm{QBG}}(v, \sigma(xyv)) &= d_{\mathrm{QBG}}(w_0, s_{2k-1}s_{2k}) = d_{\mathrm{QBG}}(s_{2k-1}w_0, s_{2k}) \\ &= d_{\mathrm{QBG}}(s_{2k}s_{2k-1}w_0, 1), \end{aligned}$$

where we applied [39, Lemma 7.7] twice. Since w_0, s_{2k-1} and s_{2k} centralize each other, we write $s_{2k}s_{2k-1}w_0 = w_0 s_{2k-1}s_{2k}$.

Denote the longest root of Φ^+ by θ. Applying [39, Lemma 7.7] again, we conclude

$$d_{\mathrm{QBG}}(w_0 s_{2k-1}s_{2k}, 1) = 1 + d_{\mathrm{QBG}}(s_\theta w_0 s_{2k-1}s_{2k}, 1).$$

Let $J = \{\alpha_1, \alpha_3, \ldots, \alpha_{2k}\}$, such that the longest element of W_J is equal to $s_\theta w_0$. Then, $s_\theta w_0 s_{2k-1}s_{2k} = s_1 w_0(J') s_{2k-1}s_{2k}$ where $J' = J \backslash \{\alpha_1\}$. We conclude

$$d_{\mathrm{QBG}}(w_0 s_{2k-1}s_{2k}, 1) = 2 + d_{\mathrm{QBG}}(w_0(J') s_{2k-1}s_{2k}, 1).$$

If $k = 2$, one checks that $w_0(J')s_{2k-1}s_{2k} = 1$, so that the desired identity follows. Otherwise, we have $k \geq 3$ and observe that J' defines a σ-stable sub-root system of type D_{2k-2}. In an inductive sense, we may assume that $d_{\mathrm{QBG}}(w_0(J')s_{2k-1}s_{2k}, 1) =$

$2(k-1)-2$ has already been established. Then also $d_{\mathrm{QBG}}(w_0 s_{2k-1} s_{2k}, 1) = 2k-2$ follows immediately.

This completes the proof that for the w constructed above, we have $d_w(b_w) - \dim X_w(b) = \frac{\ell(w_0)-\ell_R(\mathcal{O})}{2}$, establishing Theorem 2 for groups of type D_{2k} where σ has order 2.

7.5.3 The Case ${}^3D_4$

If $\mathbf{G}$ is of type D_4 and σ has order 3, enumerate the simple roots $\alpha_1, \ldots, \alpha_4$, such that $\sigma(\alpha_1) = \alpha_3$, $\sigma(\alpha_3) = \alpha_4$ and $\sigma(\alpha_4) = \alpha_1$. Then, one chooses $x = s_4$, $y = w_0\sigma^{-1}(x) = w_0 s_1$ and $v = w_0$ as above. Moreover, we choose $\mu = 2\rho_J^\vee$ with $J = \{\alpha_2, \alpha_3, \alpha_4\}$. Then, one chooses $w = xt^\mu y \in \tilde{W}$ and $v \in \mathrm{LP}(w)$ to get the same conclusion as in the example above.

7.5.4 The General Case

Following [31, Sect. 5.3], we can reduce to the case where $\mathbf{G}$ is quasi-simple over F. In other words, we assume that σ acts transitively on the set of irreducible components of W_0. We have $W_0 = W_0' \times \cdots \times W_0'$ with ℓ irreducible components. There is a length-preserving group automorphism σ' on W_0' with $\sigma(w_1, \ldots, w_\ell) = (\sigma'(w_\ell), w_1, \ldots, w_{\ell-1})$. Let w_0' be the longest element of W_0' and let $\mathcal{O}'$ be the σ'-conjugacy class of w_0' in W_0'. We distinguish the ℓ even case and the ℓ odd case as in [31, Sect. 5.4].

If ℓ is even. Let $u = (1, w_0', 1, w_0', \ldots, 1, w_0')$, then $w_0 = u\sigma(u)$, and hence, $\ell_R(\mathcal{O}) = 0$. Let $x = y = (w_0', 1, w_0', 1, \ldots, w_0', 1)$ and $\mu = (2\rho^\vee, 0, 2\rho^\vee, 0, \ldots, 2\rho^\vee, 0)$. Consider $w = xt^\mu y$. Then, $y\sigma(x) = w_0$, $v = y^{-1}u \in \mathrm{LP}(w)$ and $d_{\mathrm{QBG}}(y^{-1}u, \sigma(xu)) = d_{\mathrm{QBG}}(w_0, w_0) = 0$. Therefore, w satisfies the condition.

If ℓ is odd. Then, by [31, Sect. 5.4.2], $\ell_R(\mathcal{O}) = \ell_R(\mathcal{O}')$. By the result proved above in Sects. 7.5.1–7.5.3, one can find $w' = x' t^{\mu'} w_0' \sigma(x'^{-1})$ and $u' \in W_0'$ such that $\sigma(x') w_0' u' \in \mathrm{LP}(w')$ and $d_{\mathrm{QBG}}(\sigma(x') w_0' u', \sigma'(x'u')) = \ell_R(\mathcal{O}')$. Now, let

$$
\begin{aligned}
u &= (u', 1, 1, \ldots, 1),\\
v &= (\sigma'(x') w_0' u', x' w_0', x', x' w_0', x', \ldots, x' w_0', x'),\\
x &= (x', x' w_0', x', x' w_0', \ldots, x', x' w_0', x'),\\
y &= (w_0' \sigma'(x'^{-1}), w_0' x'^{-1}, x'^{-1}, w_0' x'^{-1}, x'^{-1}, \ldots, w_0' x'^{-1}, x'^{-1}),\\
\mu &= (\mu', \rho_{J_1}^\vee, \rho_{J_2}^\vee, \rho_{J_1}^\vee, \rho_{J_2}^\vee, \ldots, \rho_{J_1}^\vee, \rho_{J_2}^\vee),
\end{aligned}
$$

where $J_1 = \{i; x' w_0'(\alpha_i) < 0\}$, $J_2 = \{i; x'(\alpha_i) < 0\}$ and $\rho_{J_1}^\vee$ and $\rho_{J_2}^\vee$ are sum of fundamental coweights corresponding to J_1 and J_2, respectively. Consider $w = xt^\mu y$. Then, $y\sigma(x) = w_0$, $y^{-1}u \in \mathrm{LP}(w)$ and

$$
d_{\mathrm{QBG}}(y^{-1}u, \sigma(xu)) = d_{\mathrm{QBG}}(\sigma(x') w_0' u', \sigma'(x'u')) = \ell_R(\mathcal{O}') = \ell_R(\mathcal{O}).
$$

Therefore, w satisfies the condition. Theorem 2 is fully proved.

7.6 Final Comments

This concludes the theoretical discussion of the difference $d_w(b) - \dim X_w(b)$. It is noteworthy that, even though the dimension of affine Deligne–Lusztig varieties can be fully determined by the combinatorial algorithm presented in Sect. 2.3, we had to employ tools from quantum cohomology and algebraic geometry to prove Theorem 2. This situation is typical in research on affine Deligne–Lusztig varieties and illustrates why the field encompasses more than just analyzing a single algorithm on affine Weyl groups.

8 Conclusion

We used machine learning to study a central unsolved problem in pure mathematics, namely the geometry of affine Deligne–Lusztig varieties. In this section, we want to discuss the potential of this new research method and share some practical insights on the interdisciplinary collaboration.

Our project required an interdisciplinary research group, consisting of experts in machine learning and specialized researchers of the mathematical problem at hand. This joint expertise allowed us to find machine learning models using subject-specific knowledge, as well as interpret their behavior from the perspective of a subject matter expert. After exchanging explanations on the machine learning models used and the mathematical problem to be studied, we established a common understanding of the material. We could even delve into highly technical questions about modeling specific discrete functions, such as the dimension of affine Deligne–Lusztig varieties, using neural networks.

While the research method employed in this project cannot lead to new mathematical *proofs*, it still offers new insights, raises intriguing questions, and leads to conjectures. Starting with a naive view of the mathematical problem, we were able to rediscover some of the most crucial tools and invariants developed by the mathematical community with substantial amount of time and effort. This demonstrates that this pipeline has the potential to accelerate research.

Additionally, we identified new avenues of research that could be explored using established mathematical methods. By analyzing the linear model developed in Sect. 4, we formulated a new conjecture that has since been proven in Sect. 6.

We would like to emphasize some important requirements that contributed to our success, as a recommendation to other mathematical researchers (even from very different fields) who might benefit from this new research method:

- **Selection of Problems**: Choose problems that are easy to generate data for but difficult to find patterns in. In our case, the mathematical problem we studied was computable for a large number of examples. While the general goal of "better understanding the geometry of affine Deligne–Lusztig varieties" is too vague, we were able to define specific numerical invariants, such as non-emptiness, dimension, and the number of irreducible components. With a substantial amount

of computed data points ranging from thousands to millions, machine learning becomes a powerful tool to study the patterns.

- **Machine Learning Model Selection**: The choice of the machine learning model is crucial. While large neural networks may offer higher accuracy, they can reduce interpretability and make identifying patterns challenging. Complex function forms can also introduce noise in sensitivity analysis. Hence, it is important to strike a balance between model complexity and interpretability. Furthermore, the choice of loss functions and training algorithms may also play a vital role.
- **Leveraging Prior Knowledge**: Prior knowledge plays a significant role in the success of this approach. It aids in selecting appropriate features and models and helps researchers explain and evaluate the results. Understanding the problem itself is crucial for distinguishing meaningful patterns from noise, especially when facing counterintuitive results.
- **Moderate Technical Requirements**: The technical requirements are relatively low, and researchers do not need an overly complex machine learning setup. In our experiments, we found that even with millions of data points, these small networks could acquire well-trained models in just a few minutes when utilizing a single GPU. Training times on conventional laptops were generally less than an hour. However, the generation of data can be time-consuming. Despite not having an effective parallel implementation, generating millions of data points on a single CPU may still take several days.
- **Flexibility in Concerned Function**: One of the advantages of machine learning models is their ability to handle functions that are not smooth, continuous, or even discrete. In our case, the problem did not satisfy some common assumptions in machine learning, such as input feature independence. While this introduced occasional challenges, we could often find a way to alleviate it, squeezing out useful information about the machine learning model and the underlying mathematical problem in the process.
- **Effective Communication between Groups**: Interdisciplinary communication, particularly between groups steeped in pure mathematics and machine learning domains, is of paramount importance. In our research, we found that in-depth discussions on data generation, regularization, and fidelity terms significantly enhanced our comprehension of the numerical outcomes.

  While individual experiments can be executed swiftly, the subsequent analysis and interpretation of results often demand substantial time. The process inherently necessitates periodic revisions and discussions, leading to a cyclical pattern of repeated experimentation. Remarkably, the analysis and interpretation phase tends to consume the lion's share of time in this iterative process.

  Enhancing the mutual understanding between these expert groups could significantly expedite this process. A deeper appreciation of each other's domains can foster clearer communication, leading to more efficient analysis and potentially quicker identification of necessary experimental modifications. This cross-disciplinary understanding, therefore, serves as a catalyst for accelerating the overall research process. Conversely, the interdisciplinary research process itself improves the mutual understanding naturally over time.

Acknowledgements The authors would like to thank the anonymous referee for the detailed comments and suggestions. BD and PJ are partially supported by NSFC 12090022. XH is partially supported by the New Cornerstone Science Foundation through the New Cornerstone Investigator Program and the Xplorer Prize, and by Hong Kong RGC under Grant No. 14300220. All authors are fully supported by their enthusiasm toward the emerging field of AI in Mathematics.

Data Availability The data that support the findings of this study are openly available in https://github.com/Jinpf314/ML4ADLV/.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Aitken, A.C.: IV.—on least squares and linear combination of observations. Proc. R. Soc. Edinb. **55**, 42–48 (1936)
2. Bach, F., Jenatton, R., Mairal, J., Obozinski, G.: Convex optimization with sparsity-inducing norms. In: Optimization for Machine Learning, pp. 19–53. MIT Press, Cambridge, MA (2011)
3. Beazley, E.T.: Codimensions of Newton strata for $SL_3(F)$ in the Iwahori case. Math. Z. **263**(3), 499–540 (2009)
4. Berglund, P., Campbell, B., Jejjala, V.: Machine learning Kreuzer–Skarke Calabi–Yau threefolds. arXiv:2112.09117 (2021)
5. Brenti, F., Fomin, S., Postnikov, A.: Mixed Bruhat operators and Yang-Baxter equations for Weyl groups. Int. Math. Res. Not. IMRN **1999**(8), 419–441 (1999)
6. Brown, N., Sandholm, T.: Superhuman AI for multiplayer poker. Science **365**(6456), 885–890 (2019)
7. Bruhat, F., Tits, J.: Groupes réductifs sur un corps local, II. Schémas en groupes. Existence d'une donnée radicielle valuée. Publ. Math. l'Inst. Hautes Études Sci. **60**, 5–184 (1984)
8. Brunton, S.L., Proctor, J.L., Kutz, J.N.: Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci. USA **113**(15), 3932–3937 (2016)
9. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Disc. **2**(2), 121–167 (1998)
10. Chai, C.-L.: Newton polygons as lattice points. Am. J. Math. **122**(5), 967–990 (2000)
11. Chmiela, S., Tkatchenko, A., Sauceda, H.E., Poltavsky, I., Schütt, K.T., Müller, K.-R.: Machine learning of accurate energy-conserving molecular force fields. Sci. Adv. **3**(5), e1603015 (2017)
12. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**, 273–297 (1995)
13. Davies, A., Velicković, P., Buesing, L., Blackwell, S., Zheng, D., Tomasev, N., Tanburn, R., Battaglia, P., Blundell, C., Juhász, A., et al.: Advancing mathematics by guiding human intuition with AI. Nature **600**(7887), 70–74 (2021)
14. Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., et al.: Magnetic control of tokamak plasmas through deep reinforcement learning. Nature **602**(7897), 414–419 (2022)
15. Deligne, P., Lusztig, G.: Representations of reductive groups over finite fields. Ann. Math. **103**(1), 103–161 (1976)
16. Görtz, U., Haines, T.J., Kottwitz, R.E., Reuman, D.C.: Dimensions of some affine Deligne–Lusztig varieties. Ann. Sci. l'École Normale Supérieure **39**(3), 467–511 (2006)

17. Görtz, U., Haines, T.J., Kottwitz, R.E., Reuman, D.C.: Affine Deligne–Lusztig varieties in affine flag varieties. Compos. Math. **146**(5), 1339–1382 (2010)
18. Görtz, U., He, X.: Dimension of affine Deligne–Lusztig varieties in affine flag varieties. Doc. Math. **15**, 1009–1028 (2010)
19. Görtz, U., He, X., Nie, S.: P-alcoves and nonemptiness of affine Deligne–Lusztig varieties. Ann. Sci. l'École Normale Supérieure **48**(3), 647–665 (2015)
20. Hamacher, P.: The geometry of Newton strata in the reduction modulo p of Shimura varieties of PEL type. Duke Math. J. **164**(15), 2809–2895 (2015)
21. Hamacher, P., Viehmann, E.: Irreducible components of minuscule affine Deligne–Lusztig varieties. Algebra Number Theory **12**(7), 1611–1634 (2018)
22. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv:1510.00149 (2015)
23. He, X.: Minimal length elements in some double cosets of Coxeter groups. Adv. Math. **215**(2), 469–503 (2007)
24. He, X.: A subalgebra of 0-Hecke algebra. J. Algebra **322**(11), 4030–4039 (2009)
25. He, X.: Geometric and homological properties of affine Deligne–Lusztig varieties. Ann. Math. **179**(1), 367–404 (2014)
26. He, X.: Hecke algebras and p-adic groups. In: Current Developments in Mathematics 2015, pp. 73–135. International Press, Somerville, MA (2016)
27. He, X.: Some results on affine Deligne–Lusztig varieties. In: Proceedings of the International Congress of Mathematicians—Rio de Janeiro 2018. Vol. II. Invited Lectures, pp. 1345–1365. World Scientific, Hackensack, NJ (2018)
28. He, X.: Cordial elements and dimensions of affine Deligne–Lusztig varieties. Forum Math. Pi **9**, e9 (2021)
29. He, X., Nie, S.: Minimal length elements of extended affine Weyl groups. Compos. Math. **150**(11), 1903–1927 (2014)
30. He, X., Nie, S., Yu, Q.: Affine Deligne–Lusztig varieties with finite Coxeter parts. arXiv:2208.14058 (2022)
31. He, X., Yu, Q.: Dimension formula for the affine Deligne–Lusztig variety $X(\mu, b)$. Math. Ann. **379**(3–4), 1747–1765 (2021)
32. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems 33 (NeurIPS 2020), pp. 6840–6851. Curran Associates Inc., Red Hook, NY (2020)
33. Jia, W., Wang, H., Chen, M., Lu, D., Lin, L., Car, R., E, W., Zhang, L.: Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning. In: SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–14. IEEE (2020)
34. Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Zidek, A., Potapenko, A., et al.: Highly accurate protein structure prediction with AlphaFold. Nature **596**(7873), 583–589 (2021)
35. Kates-Harbeck, J., Svyatkovskiy, A., Tang, W.: Predicting disruptive instabilities in controlled fusion plasmas through deep learning. Nature **568**(7753), 526–531 (2019)
36. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv:1412.6980 (2014)
37. Kottwitz, R.E.: Isocrystals with additional structure. Compos. Math. **56**(2), 201–220 (1985)
38. Lam, T., Shimozono, M.: Quantum cohomology of G/P and homology of affine Grassmannian. Acta Math. **204**(1), 49–90 (2010)
39. Lenart, C., Naito, S., Sagaki, D., Schilling, A., Mark, S.: A uniform model for Kirillov-Reshetikhin crystals I: lifting the parabolic quantum Bruhat graph. Int. Math. Res. Not. IMRN **2015**(7), 1848–1901 (2015)
40. Long, Z., Lu, Y., Dong, B.: PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. J. Comput. Phys. **399**, 108925 (2019)
41. Long, Z., Lu, Y., Ma, X., Dong, B.: PDE-Net: learning PDEs from data. Proc. Mach. Learn. Res. **80**, 3208–3216 (2018)
42. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv:1711.05101 (2017)
43. Miethke, M., Pieroni, M., Weber, T., Brönstrup, M., Hammann, P., Halby, L., Arimondo, P.B., Glaser, P., Aigle, B., Bode, H.B., et al.: Towards the sustainable discovery and development of new antibiotics. Nat. Rev. Chem. **5**(10), 726–749 (2021)

44. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: representing scenes as neural radiance fields for view synthesis. Commun. ACM **65**(1), 99–106 (2021)
45. Milićević, E.: Maximal newton points and the quantum Bruhat graph. Mich. Math. J. **70**(3), 451–502 (2021)
46. Milićević, E., Viehmann, E.: Generic Newton points and the Newton poset in Iwahori-double cosets. Forum Math. Sigma **8**, e50 (2020)
47. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML'10), pp. 807–814. Omnipress, Madison, WI (2010)
48. Nelder, J.A., Wedderburn, R.W.M.: Generalized linear models. J. R. Stat. Soc. Ser. A **135**(3), 370–384 (1972)
49. Nie, S.: Irreducible components of affine Deligne–Lusztig varieties. Camb. J. Math. **10**(2), 433–510 (2022)
50. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019)
51. Rapoport, M.: A guide to the reduction modulo p of Shimura varieties. Astérisque **298**, 271–318 (2002)
52. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. **65**(6), 386–408 (1958)
53. Ruder, S.: An overview of gradient descent optimization algorithms. arXiv:1609.04747 (2016)
54. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
55. Sadhukhan, A.: Affine Deligne–Lusztig varieties and quantum Bruhat graph. Math. Z. **303**(1), 21 (2023)
56. Schremmer, F.: Generic Newton points and cordial elements. arXiv:2205.02039 (2022)
57. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
58. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
59. Stokes, J.M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N.M., MacNair, C.R., French, S., Carfrae, L.A., Bloom-Ackermann, Z., et al.: A deep learning approach to antibiotic discovery. Cell **180**(4), 688–702 (2020)
60. Viehmann, E.: The dimension of some affine Deligne–Lusztig varieties. Ann. Sci. l'École Normale Supérieure **39**(3), 513–526 (2006)
61. Viehmann, E.: Truncations of level 1 of elements in the loop group of a reductive group. Ann. Math. **179**(3), 1009–1040 (2014)
62. Viehmann, E.: On the geometry of the Newton stratification. In: Shimura Varieties, pp. 192–208. Cambridge University Press, Cambridge (2020)
63. Viehmann, E.: Minimal Newton strata in Iwahori double cosets. Int. Math. Res. Not. IMRN **2021**(7), 5349–5365 (2021)
64. Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., et al.: Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature **575**(7782), 350–354 (2019)
65. Von Lilienfeld, O.A., Burke, K.: Retrospective on a decade of machine learning for chemical discovery. Nat. Commun. **11**(1), 4895 (2020)
66. Williamson, G.: Is deep learning a useful tool for the pure mathematician? Bull. Am. Math. Soc. **61**(2), 271–286 (2024)
67. Zhang, L., Han, J., Wang, H., Car, R., E, W.: Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. Phys. Rev. Lett. **120**(14), 143001 (2018)
68. Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al.: A survey of large language models. arXiv:2303.18223 (2023)
69. Zhou, R., Zhu, Y.: Twisted orbital integrals and irreducible components of affine Deligne–Lusztig varieties. Camb. J. Math. **8**(1), 149–241 (2020)
70. Zhu, X.: Affine Grassmannians and the geometric Satake in mixed characteristic. Ann. Math. (2) **185**(2), 403–492 (2017)