MURMURATIONS of ELLIPTIC CURVES EXPERIMENTS

Martin Lotz

Warwick University

January 26, 2024

Setup

Let $E/\mathbb{Q}$ be an elliptic curve and fix $M \in \mathbb{N}$. Define the vector

$$v(E) = \left(\frac{a_{p_1}(E)}{2\sqrt{p_1}}, \ldots, \frac{a_{p_M}(E)}{2\sqrt{p_M}} \right),$$

where p_n is the n-th prime number and

$$a_p(E) := p + 1 - \#E(\mathbb{F}_p).$$

We are interested in properties of sets $\{v(E)\}$ as E ranges over elliptic curves in a given conductor range.
See Robin's talk for more background and motivation.

Murmurations

Let $\mathcal{E}_r(N_1, N_2)$ denote the set of elliptic curves of rank r with conductor between N_1 and N_2 (consider one representative per isogeny class).

Define the averages

$$f_r(n) := \frac{1}{\#\mathcal{E}_r(N_1, N_2)} \sum_{E \in \mathcal{E}_r(N_1, N_2)} \frac{a_{p_n}(E)}{2\sqrt{p_n}}$$

Murmurations

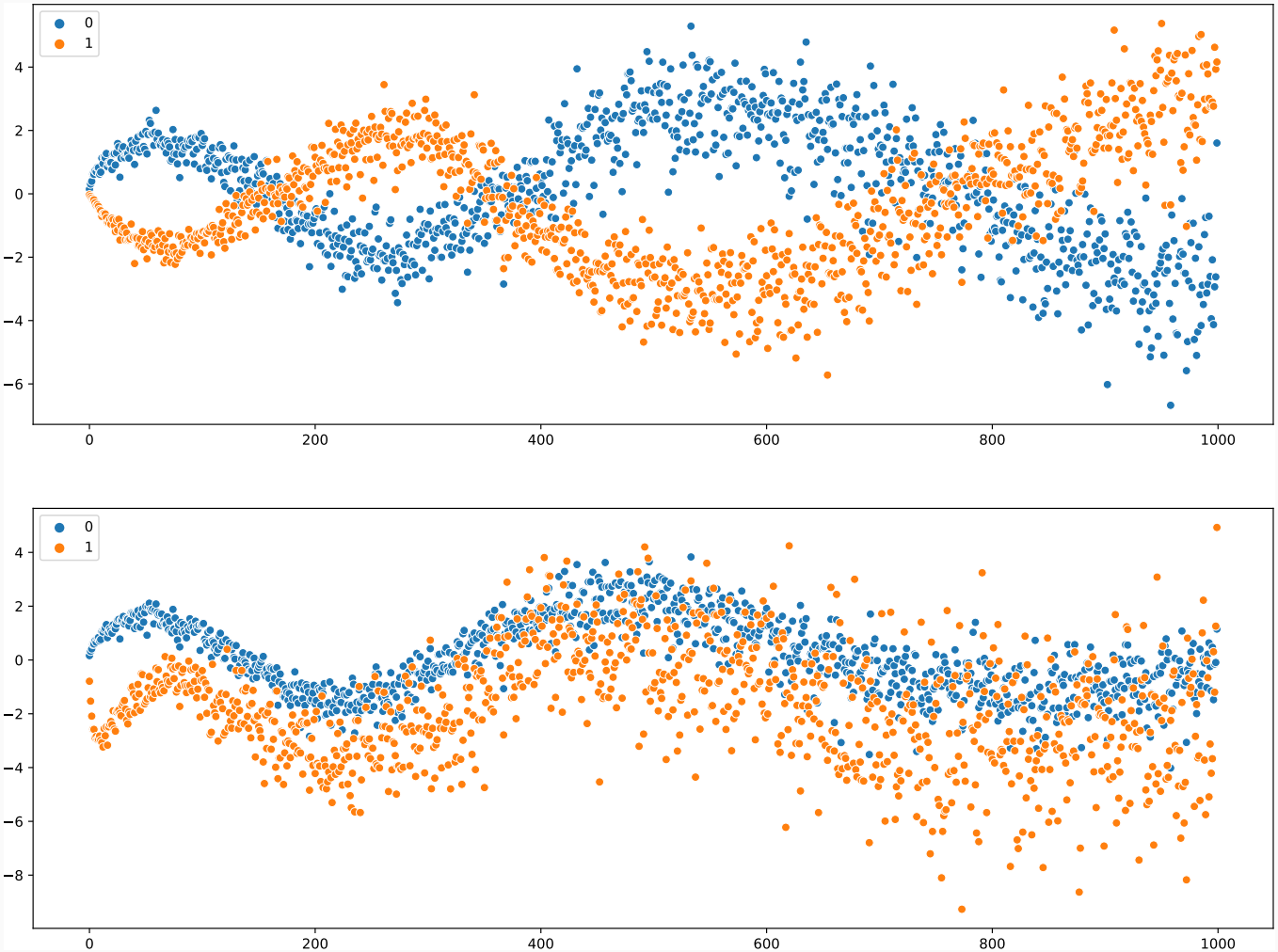
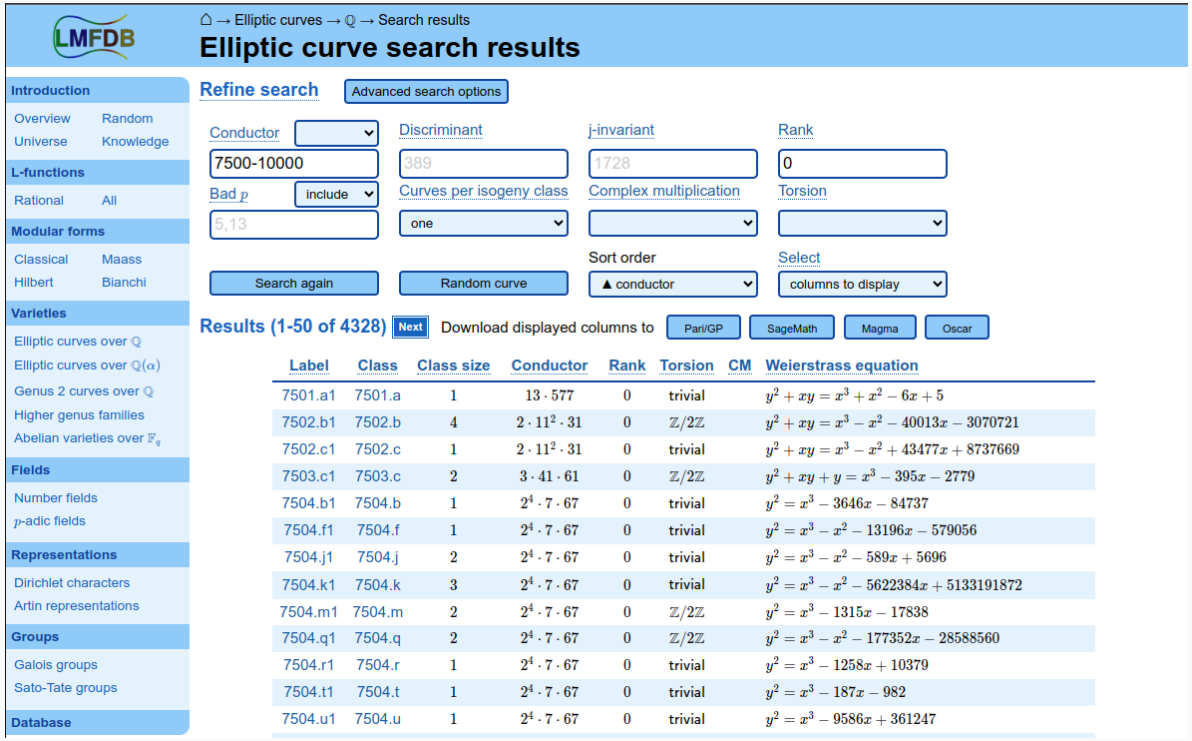


Figure 1: Top: $f_0(n)$ and $f_1(n)$ for curves in $\mathrm{E}_r(7500, 10000)$.Bottom: $f_0(n)$ and $f_2(n)$ for curves in $\mathrm{E}_r(5000, 10000)$.

Data source

Elliptic curves are sourced from John Cremona's EC database, via the The L-functions and modular forms database (LMFDB).



LMFDB

⌂ → Elliptic curves → ℚ → Search results

Elliptic curve search results

Introduction: Overview, Random, Universe, Knowledge

L-functions: Rational, All

Modular forms: Classical, Maass, Hilbert, Bianchi

Varieties: Elliptic curves over ℚ; Elliptic curves over $\mathbb{Q}(\alpha)$; Genus 2 curves over ℚ; Higher genus families; Abelian varieties over $\mathbb{F}_q$

Fields: Number fields; p-adic fields

Representations: Dirichlet characters; Artin representations

Groups: Galois groups; Sato-Tate groups

Database

Refine search | Advanced search options

Conductor: 7500-10000 | Discriminant: 389 | j-invariant: 1728 | Rank: 0

Bad p: include; 5,13 | Curves per isogeny class: one | Complex multiplication | Torsion

Sort order: ▲ conductor | Select: columns to display

Search again | Random curve

Results (1-50 of 4328) Next | Download displayed columns to Pari/GP SageMath Magma Oscar

Label	Class	Class size	Conductor	Rank	Torsion	CM	Weierstrass equation
7501.a1	7501.a	1	$13 \cdot 577$	0	trivial		$y^2 + xy = x^3 + x^2 - 6x + 5$
7502.b1	7502.b	4	$2 \cdot 11^2 \cdot 31$	0	$\mathbb{Z}/2\mathbb{Z}$		$y^2 + xy = x^3 - x^2 - 40013x - 3070721$
7502.c1	7502.c	1	$2 \cdot 11^2 \cdot 31$	0	trivial		$y^2 + xy = x^3 - x^2 + 43477x + 8737669$
7503.c1	7503.c	2	$3 \cdot 41 \cdot 61$	0	$\mathbb{Z}/2\mathbb{Z}$		$y^2 + xy + y = x^3 - 395x - 2779$
7504.b1	7504.b	1	$2^4 \cdot 7 \cdot 67$	0	trivial		$y^2 = x^3 - 3646x - 84737$
7504.f1	7504.f	1	$2^4 \cdot 7 \cdot 67$	0	trivial		$y^2 = x^3 - x^2 - 13196x - 579056$
7504.j1	7504.j	2	$2^4 \cdot 7 \cdot 67$	0	trivial		$y^2 = x^3 - x^2 - 589x + 5696$
7504.k1	7504.k	3	$2^4 \cdot 7 \cdot 67$	0	trivial		$y^2 = x^3 - x^2 - 5622384x + 5133191872$
7504.m1	7504.m	2	$2^4 \cdot 7 \cdot 67$	0	$\mathbb{Z}/2\mathbb{Z}$		$y^2 = x^3 - 1315x - 17838$
7504.q1	7504.q	2	$2^4 \cdot 7 \cdot 67$	0	$\mathbb{Z}/2\mathbb{Z}$		$y^2 = x^3 - x^2 - 177352x - 28588560$
7504.r1	7504.r	1	$2^4 \cdot 7 \cdot 67$	0	trivial		$y^2 = x^3 - 1258x + 10379$
7504.t1	7504.t	1	$2^4 \cdot 7 \cdot 67$	0	trivial		$y^2 = x^3 - 187x - 982$
7504.u1	7504.u	1	$2^4 \cdot 7 \cdot 67$	0	trivial		$y^2 = x^3 - 9586x + 361247$

Data preparation

Useful principles when dealing with data:

- Make it easy to load and manipulate data.
- Make it fast.
- Make it generalizable.

This is achieved by:

- Transform Sage files into Polars dataframes.
- Precompute a_p invariants and store everything in an efficient format.
- Create class template for fast loading of data in format suitable to machine learning (Python) packages.
- Put on data scientist hat.

Some tools


NumPy
matplotlib
PyTorch
pandas
SciPy
TensorFlow
scikit
learn
seaborn

Data loading

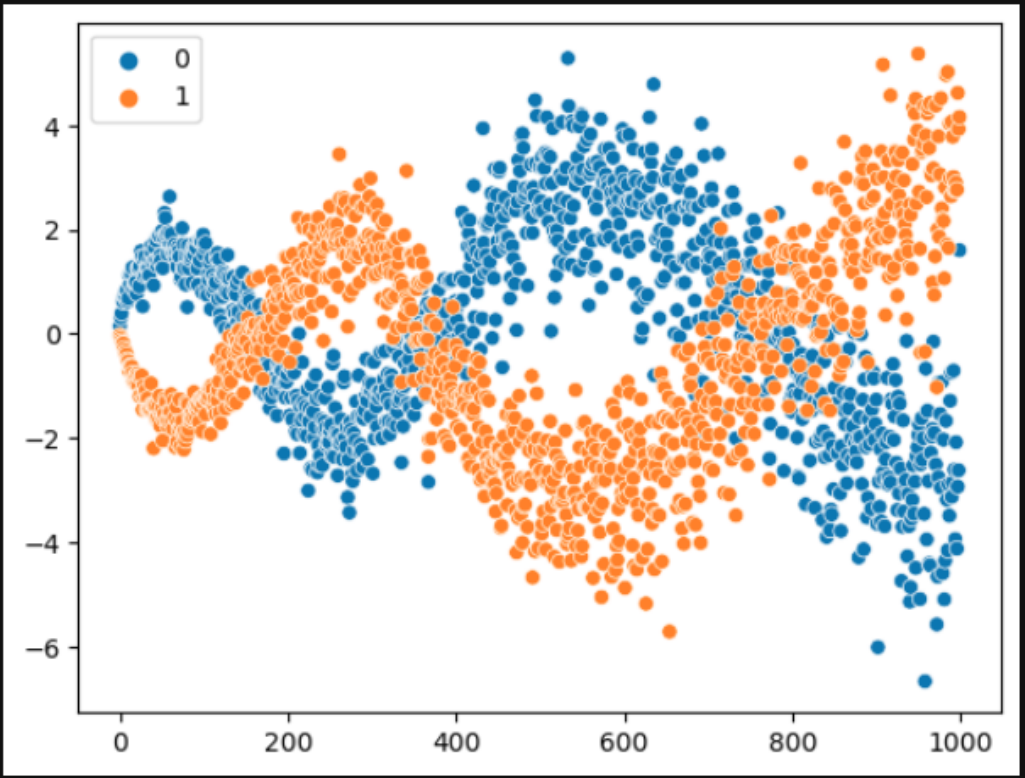
[4]:
```
# Load data into memory
source = 'data/ec_a.parquet'
ecdf = pl.read_parquet(source)
print(ecdf.shape)
```

```
(214070, 1005)
```

[5]:
```
EC = ECDataLoader(ecdf, N = 1000)
```

[6]:
```
murmurations = EC.load_averages(ranks = [0,1], conductor_range = [7500, 10_000])
```

[7]:
```
sns.scatterplot(data = murmurations, markers = ['o', 'o'])
plt.show()
```



Data loading

```
[8]: X, y = EC.load(ranks = [0, 1, 2],
                    conductor_range = [int(10000), int(40000)],
                    target = 'rank',
                    n_samples = 36000,
                    normalized = True,
                    balanced = True)

     print(X.shape)

     (36000, 1000)
```

Figure 2: Get the $v(E)$ vectors for 36000 random elliptic curves in a given conductor range with ranks 0, 1 and 2. Balanced means that the amount of curves with each rank is the same. Each vector $v(E)$ is a point in $\mathbb{R}^{1000}$.

Dimensionality Reduction

Dimensionality reduction is a set of techniques used to reduce the number of features (dimensions) in a dataset while retaining as much relevant information as possible. It is used for improved performance of algorithms (circumvent "curse of dimensionality"), noise reduction, data visualization and data compression.
Methods include:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Uniform Manifold Approximation and Projection (UMAP)
- t-distributed Stochastic Neighbor Embedding (t-SNE)

as well as neural network-based methods such as autoencoders.

Principal Component Analysis

Assume our data consists of the vectors $\{v_i\}_{i \leq m} \subset \mathbb{R}^M$ (for example, $M = 1000$). The goal is to find a linear subspace L with $\dim L = k$ that approximates the data as well as possible, in the sense that

$$\sum_{i=1}^{m} \operatorname{dist}(v_i, L)^2$$

is minimal among k-dimensional linear subspaces L.

Solution: Let X be the matrix with the v_i as rows, and $S = X^T X$. The eigenvector u_i corresponding to the i-th largest eigenvalue of S is called the i-th principal component. Moreover, $L = \operatorname{span}\langle u_1, \ldots, u_k \rangle$ is the best-approximating k-dimensional linear subspace.

Principal Component Analysis

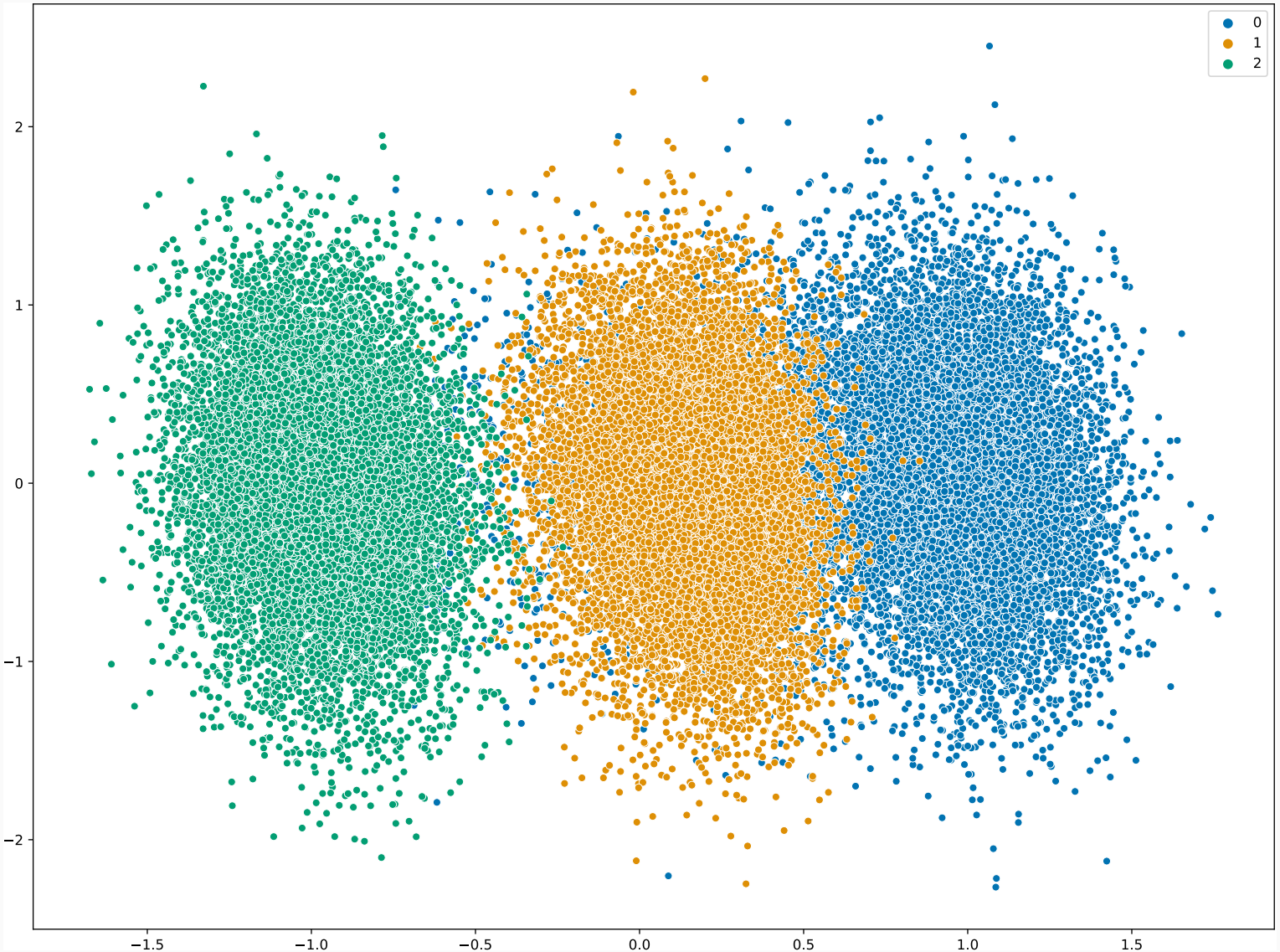


Figure 3: First two principal components.

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a statistical method used in machine learning and pattern recognition for both dimensionality reduction and classification.

- LDA is a supervised method. It aims to find a projection that maximizes the distance between different labels.

Linear Discriminant Analysis

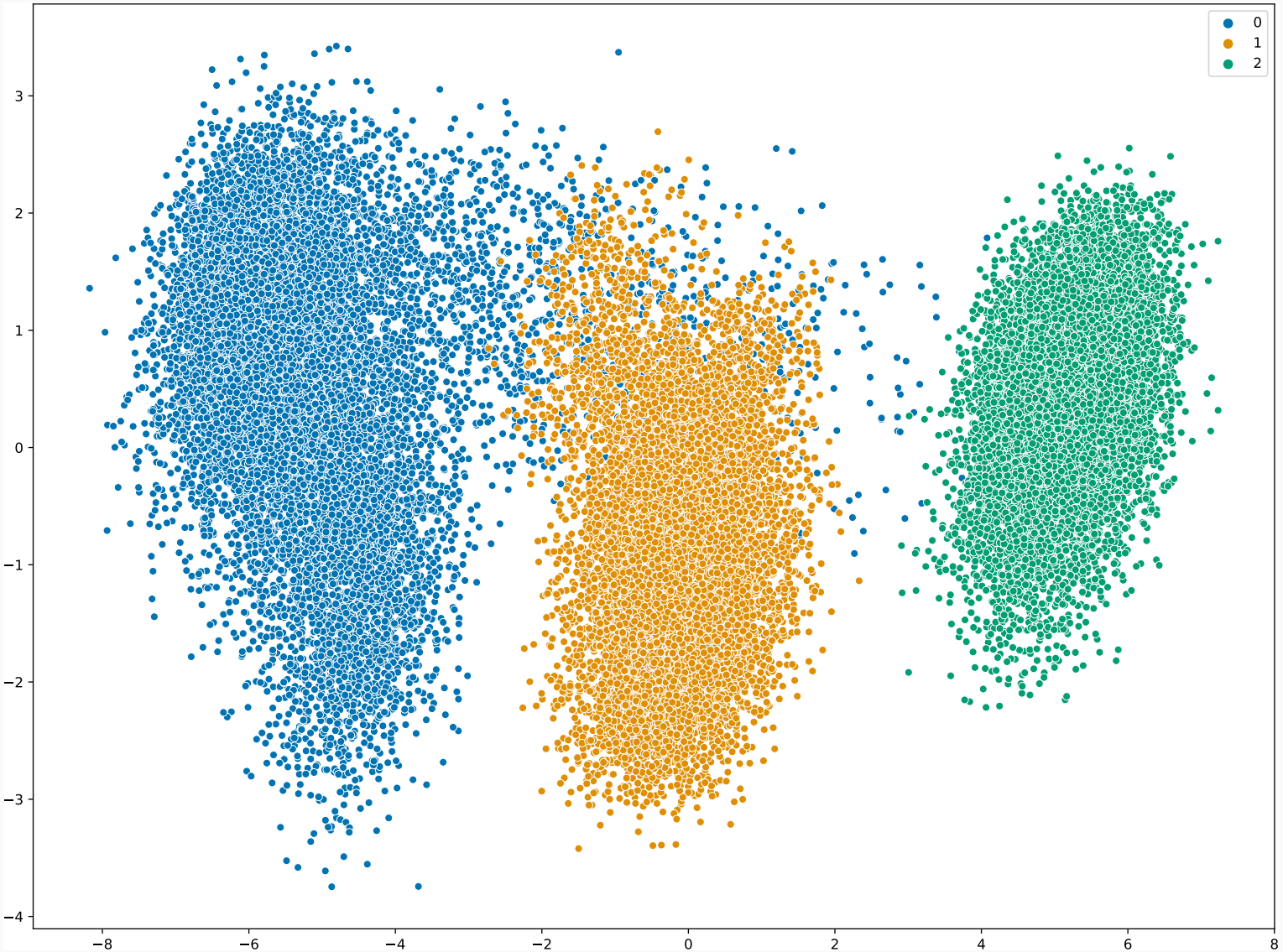


Figure 4: Linear discriminant analysis.

Prediction and classification

Machine learning methods can be used to predict the rank of an elliptic curve. This is an example of a classification problem (like recognizing items from images). Some approaches:

- Logistic regression (used in HLOP Murmurations paper)
- Support Vector Machines (used in this talk)
- Tree-based methods (random forests, XGBoost)
- Neural networks

Support Vector Machines

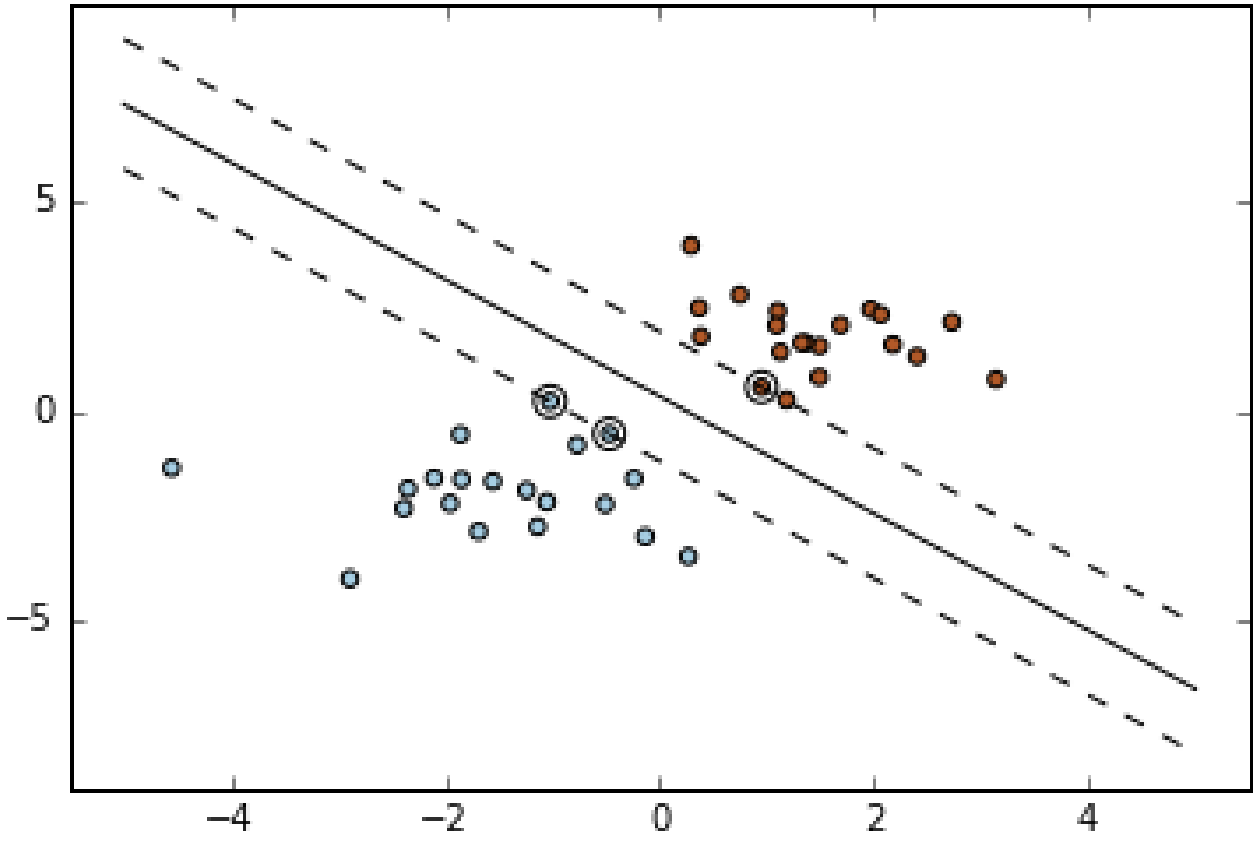


Figure 5: Support vector machine

Support Vector Machines

Assume a set of points $\{v_i\} \subset \mathbb{R}^d$ with labels $\{y_i\} \in \{-1, 1\}$ is linearly separable. A support vector machine (SVM) is a linear hyperplane

$$h(x) = w^T x + b$$

such that for all data points v_i,

$$w^T v_i + b \geq 1 \text{ if } y_i = 1, \quad w^T v_i + b \leq -1 \text{ if } y_i = -1.$$

A separating hyperplane of maximal margin can be found by solving the optimization problem:

$$\begin{aligned} &\text{minimize } \frac{1}{2}\|w\|^2 \\ &\text{subject to } y_i(w^T v_i + b) - 1 \geq 0, \quad 1 \leq i \leq n. \end{aligned}$$

Support Vector Machines

SVMs generalize to

- multiple classes,
- non-linearly separable data (by penalizing errors in the optimization problem, thus trying to minimize the misclassification error),
- non-linear separation via the kernel trick.

The dual problem of the SVM optimization problem depends only on the inner products $\langle v_i, v_j \rangle$ of the data. We can therefore embed the data into a Hilbert space, $\varphi \colon \mathbb{R}^d \to \mathcal{H}$, such that

$$\langle \varphi(v_i), \varphi(v_j) \rangle = K(v_i, v_j)$$

for a kernel function K. Example: the radial basis function (RBF) kernel

$$K(v_i, v_j) = \mathrm{e}^{-\frac{\|v_i - v_j\|^2}{2\sigma^2}}.$$

Support Vector Machines

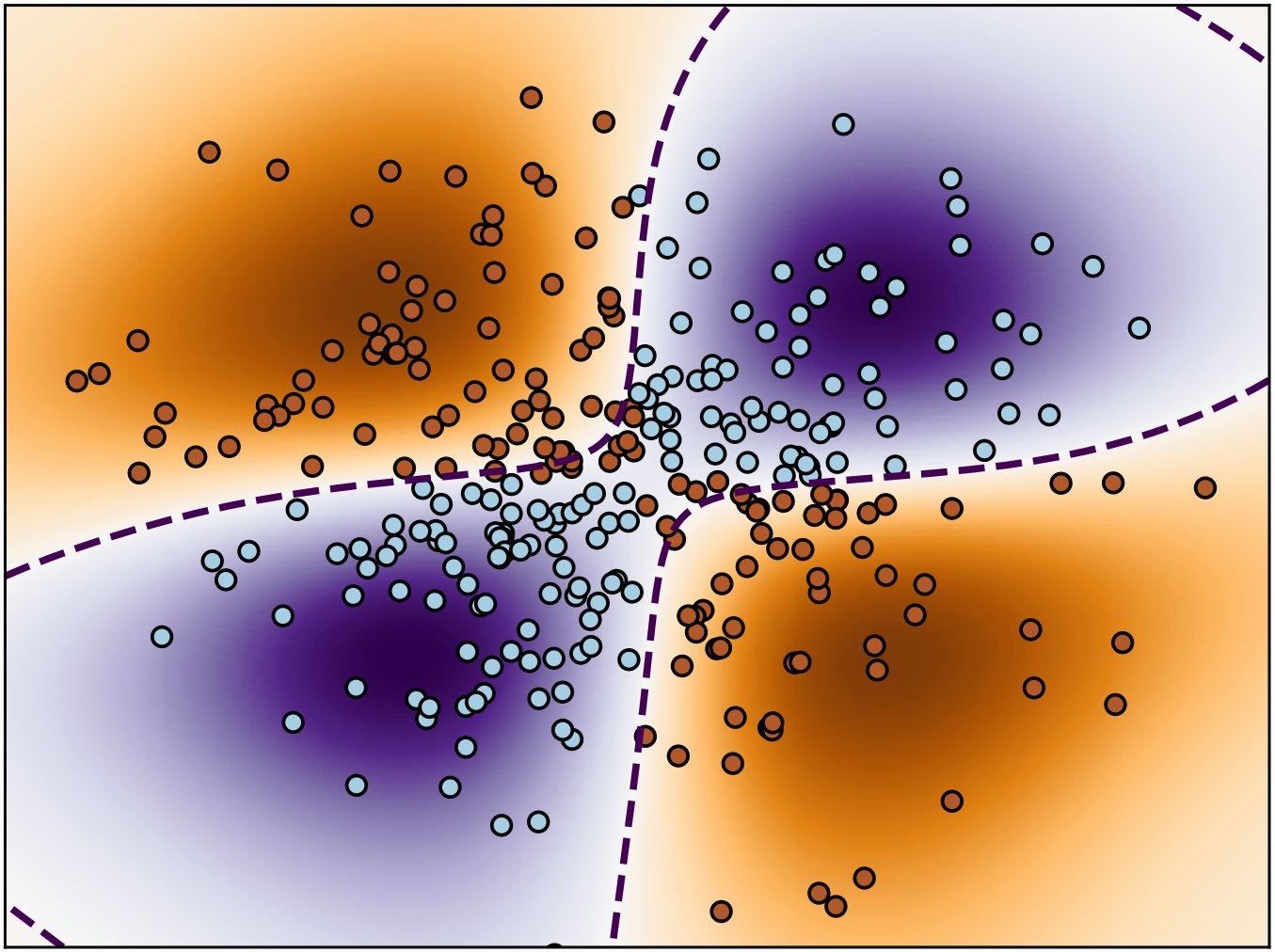
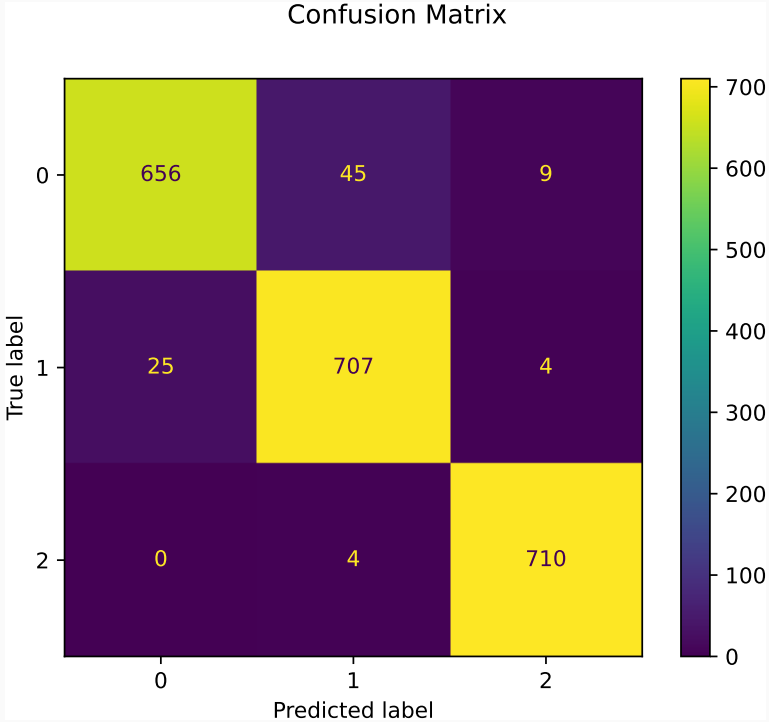


Figure 6: Non-linear SVM (from scikit-learn)

Support Vector Machines

We can apply SVMs to predict the rank of elliptic curves.

```
[40]: # Split data into 50% train and 50% test subsets
      X_train, X_test, y_train, y_test = train_test_split(
          X, y, test_size=0.3, shuffle=True
      )

[41]: clf = svm.SVC(kernel = 'sigmoid')
      clf.fit(X_train, y_train)

[41]: SVC
      SVC(kernel='sigmoid')

[42]: predicted = clf.predict(X_test)

[44]: print(
          f"Classification report for classifier {clf}:\n"
          f"{metrics.classification_report(y_test, predicted)}\n"
      )

Classification report for classifier SVC(kernel='sigmoid'):
              precision    recall  f1-score   support

           0       0.96      0.92      0.94       710
           1       0.94      0.96      0.95       736
           2       0.98      0.99      0.99       714

    accuracy                           0.96      2160
   macro avg       0.96      0.96      0.96      2160
weighted avg       0.96      0.96      0.96      2160
```



For distinguishing between rank 0 and 2 we get accuracy of 0.99.

Thank You!