An Algorithm for Learning Switched Linear Dynamics from Data.

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present an algorithm for learning switched linear dynamical systems in discrete time from data that may include noisy observations of the full system state or output observations. Switched linear systems use multiple linear dynamical modes to explain the data within some desired tolerance. They arise quite naturally in many applications such as robotics and cyber-physical systems. Learning these dynamics from given data is a NP-hard problem that is very closely related to the k-linear regression problem of fitting $k > 1$ linear models to given data. A direct 0–1 mixed integer linear programming approach yields time complexity that is exponential in the number of data points.

In this paper, we modify the problem formulation to yield an algorithm that is linear in the size of the data while being exponential in the number of state variables and the desired number of modes. To do so, we combine classic ideas from the ellipsoidal method for solving convex optimization problems, and well-known oracle separation results in non-smooth optimization. We demonstrate our approach on a set of microbenchmarks and a few interesting real-world datasets. Our evaluation on a prototype implementation suggests that the benefits of this algorithm can be made practical even against highly optimized off-the-shelf solvers.

1 Introduction

Switched linear systems model dynamical systems that arise in diverse areas including natural sciences (biological models) [3, 7, 15], robotics [2], and cyber-physical systems [1]. They are characterized by a finite set of modes, wherein each mode features a different set of governing equations for the future states in terms of the current state of the system. In this paper, we study algorithms for inferring the dynamics of a switched linear system from data that consists of full-state or output observations. Specifically, given a set of data points $\{(\mathbf{x}_i, \mathbf{x}'_i)\}_{i=1}^N$ of current and next state observations, and a number of modes m, we wish to find matrices $A_1, \ldots, A_m$ such that each data point $(\mathbf{x}_i, \mathbf{x}'_i)$ is "explained away" by some matrix A_j. I.e, $(\forall i)\ (\exists j)\ ||\mathbf{x}'_i - A_j\mathbf{x}_i|| \leq \epsilon||\mathbf{x}_i|| + \tau$, for given constants $\epsilon, \tau > 0$. However, the switching signal which governs the current mode associated with each data point is latent. This renders the problem NP-hard and the best known approaches have exponential complexity in terms of the number of data points and the desired number of modes. The exponential complexity in the number of data points makes this problem especially challenging since many examples feature thousands of data points but with tens of modes and state variables.

The key insight in this paper is that by reformulating this problem to incorporate a "gap" in the tolerance, we can significantly improve the complexity to be *linear* in the number of data points N and exponential in the number of modes and the dimensionality of the state space. Rather than providing a yes/no answer for a single tolerance bound ϵ, our approach either finds matrices that satisfy the tolerance bound for some $\epsilon_2 > 0$ or conclude that no matrices exist that can explain

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

the data with tolerance ϵ_1 ($< \epsilon_2$). However, if the underlying data could have been explained away with some tolerance that lies in the gap $[\epsilon_1, \epsilon_2]$, our algorithm may succeed or fail with no guarantees provided. Our technical approach exploits the gap in our formulation to argue that any set of feasible solutions to our problem must have a lower bound to its volume. Furthermore, by using a careful strategy for exploring the solution space, we guarantee that each step of our approach will shrink the volume of our remaining solution space by some factor. This is achieved using ideas that are similar to separation oracles in combinatorial optimization [19], or the so-called "cutting plane" argument [9]. However, our approach applies these inside a branch-and-bound algorithm to bound the depth of the overall tree. Thus, we achieve a bound on the running time that is exponential in the state dimension and the number of modes but linear in the data size.

We demonstrate that our ideas can be practically realized using a prototype implementation that is compared against a state-of-the-art mixed integer linear programming solver (Gurobi) [17]. Our evaluation results confirm that the theoretical insights also apply in practice to yield an algorithm that is often orders of magnitude faster, especially as the amount of data increases. We also demonstrate our approach using two interesting applications that include modeling human writing on a tablet in order to predict the alphabet from a small number of samples, and human highway driving scenarios from a data set gathered from drivers along a busy segment of US-101 near Los Angeles, CA.

1.1 Related Work

The problem of switched and hybrid system identification has been widely studied using a variety of approaches going back to the early 1980s [28]. We discuss a few representative approaches, refering the reader to the monograph by Lauer et al [21] for further details.

The identification approach may be *exact* — seeking a global solution that minimizes the discrepancy between the data and the model prediction, or *approximate* — wherein the optimization problem is solved approximately or assumptions about the nature of the switching signal are used to simplify the problem. Our approach here is *exact* but over a formulation that allows for a gap. Likewise, many approaches, including ours, focus on identifying the dynamics of individual modes while assuming that the switching signal is *exogenous*; on the other hand, other approaches, such as *piecewise affine system identification* [4, 25] or *hybrid automata learning* [26], attempt to identify the dynamics of each mode along with rules for transitioning between modes.

Vidal et al present an exact approach for finding matrices that fit the data in the absence of noise by posing the problem as one of finding zeros of a multivariate polynomial [30]. By exploiting the problem structure, they provide a criterion based on the rank of a matrix to identify the number of modes needed and use randomization to recover the individual model parameters for each mode. The approach provides some heuristics for the case when the data has noise. This is subsequently refined by Ozay et al [24] to handle the case of noisy data, by using sum-of-squares relaxation to a semi-definite programming problem. Assumptions on the nature of the switching signal can drastically simplify the problem at hand. For instance, Ozay shows that the problem can be solved in polynomial time using a dynamic programming algorithm if the number of mode switches in a trajectory is to be minimized [23]. Another approximate approach involves clustering to identify data points that may belong to the same mode, and thus solve the problem of identifying the dynamics for each mode given this information [12, 20]. The performance of clustering algorithms can vary depending on the nature of the data and no guarantees are available.

The problem of switched linear system identification has received attention from the machine learning community as well. For instance, piecewise linear systems can be modeled naturally using neural networks with ReLU units and back-propagation can then be used to train network parameters from given data [16]. However, the number of possible modes can be exponential in the network size. Furthermore, switched systems often involve discontinuous switching whereas neural network learning approaches seek to smooth such discontinuities using activation functions such as sigmoid.

Ly and Lipson present an approach for learning hybrid automata from data using evolutionary techniques [22]. Their approach combines symbolic regression for learning the dynamics of the individual modes with techniques for segmenting the traces into modes and inferring the guards for the state-dependent switching. Elsewhere, the problem of combining hidden markov models with linear systems, called Jump Markov Models (JMMs), has been well studied in the machine learning community using a variety of inference techniques. A common strategy for identifying JMMs modifies

the expectation-maximization (EM) algorithm [11]. Gharamani and Hinton propose a technique wherein the difficulties in the E-step are resolved using variational inference [14], whereas Blake et al propose a sampling-based scheme [6]. A key difference between our approach and JMMs is that we focus on the algorithmic efficiency for learning the dynamics of each mode from data. The problem of learning the Markov model that generates the data is thus decoupled in our approach. However, our example on handwriting recognition presented in Section 4 illustrates how off-the-shelf approaches for learning a finite-state model can be combined with our approach in a fruitful manner. Our approach also side-steps the problem of estimating the number of modes which is often an issue. This problem can be addressed by adding a penalty term accounting for the "model complexity" to the overall objective function to be minimized [22] or by using more sophisticated non-parametric Bayesian approaches wherein the prior distribution specifies unbounded number of modes and the varying number of model parameters varying as the number of modes increases [13].

2 Problem Formulation

We will use bold, lower case symbols ($\mathbf{x}, \mathbf{y}, \mathbf{w}$) to denote vectors, and upper case letters (A, B, C) to denote matrices, unless otherwise mentioned. Let $\mathbb{R}$ denote the set of real numbers, and $\mathbb{N}$ the set of natural numbers. For $m \in \mathbb{N}$, let $[m]$ denote the set $\{1, \ldots, m\}$.

A linear dynamical system in discrete time is defined by its state $\mathbf{x}(t) \in \mathbb{R}^n$ and output $\mathbf{y}(t) \in \mathbb{R}^p$, at each time $t \in \mathbb{N}$. The state evolves using the dynamical rule: $\mathbf{x}(t+1) = A\mathbf{x}(t)$, wherein $A \in \mathbb{R}^{n \times n}$. The output is obtained as $\mathbf{y}(t) = C\mathbf{x}(t)$, wherein $C \in \mathbb{R}^{p \times n}$.

Definition 1 (Switched Linear Dynamical System)**.** *A switched linear dynamical system with state* $\mathbf{x} : \mathbb{N} \to \mathbb{R}^n$ *and output* $\mathbf{y} : \mathbb{N} \to \mathbb{R}^p$ *with* $m \geq 1$ *dynamical modes is defined by* m *linear dynamical systems* $A_1, \ldots, A_m$*. For each time* $t \in \mathbb{N}$*, there exists a mode* $\sigma(t) \in [m]$ *s.t.* $\mathbf{x}(t+1) = A_{\sigma(t)}\mathbf{x}(t)$*.*

The switching signal $\sigma : \mathbb{N} \to [m]$ *chooses a mode for each time* t *such that the continuous state* $\mathbf{x}(t)$ *evolves according to this mode. The output is given by* $\mathbf{y}(t) = C\mathbf{x}(t)$ *according to a given matrix* $C \in \mathbb{R}^{p \times n}$*. We can generalize this to have output maps that also depend on the mode.*

Switched *affine* dynamical systems, wherein $\mathbf{x}(t+1) = A_j\mathbf{x}(t) + \mathbf{b}_j$ for $\mathbf{b}_j \in \mathbb{R}^n$, are handled by augmenting $\mathbf{x}$ with an extra component that is always 1 and treating $\mathbf{b}_j$ as a new column of A_j.

The goal of this paper is to learn a switched linear dynamical system model from observation data. The simplest type of data consists of full state observations involving pairs of states of the form $\{(\mathbf{x}_i, \mathbf{x}'_i)\}_{i=1}^N$, wherein each $\mathbf{x}_i = \mathbf{x}(t_i)$ and $\mathbf{x}'_i = \mathbf{x}(t_i + 1)$ are states observed at two successive time instants t_i and t_i+1. In particular, the switching signal σ is *not* observable. We also assume that the number m is known. However, knowing an upper bound M on the number modes is sufficient: we perform binary search, iterating our algorithm for $O(\log(M))$ steps, each with fixed m.

The assumption of full state observation is often cumbersome and impractical. However, our techniques extend to output observations as well. The key idea is to use a fixed-length history of output observations $\mathbf{y}(t-p), \ldots, \mathbf{y}(t)$ and find a switched auto-regressive (SARX) [32] model of the form:

$$\mathbf{y}(t+1) = \sum\nolimits_{k=0}^{p} B_{\sigma(t),k}\mathbf{y}(t-k), \text{ wherein } \sigma(t) \in [m],\ B_{j,k} \in \mathbb{R}^{p \times p}\,.$$

Therefore, in the following, we will focus our description on the full state data.

Switched Linear Dynamical System Identification: The central problem of interest is therefore to learn matrices $A_1, \ldots, A_m$ that "explain away" a set of full state observation data. First, we define the notion of "explaining away". Let us fix the data $D : \{(\mathbf{x}_i, \mathbf{x}'_i)\}_{i=1}^N$.

Definition 2 (Explaining Away With Tolerance ϵ)**.** *A set of matrices* $A_1, \ldots, A_m$ *"explains away" the data* D *with* error bound $\tau > 0$ *and* tolerance $\epsilon > 0$ *iff*

$$(\forall\, (\mathbf{x}_i, \mathbf{x}'_i) \in D)\ (\exists\, j \in [m])\ \|\mathbf{x}'_i - A_j\mathbf{x}_i\|_\infty \leq \epsilon\|\mathbf{x}_i\|_\infty + \tau\,. \tag{1}$$

The L_∞ norm can be replaced by the L_2 or L_1 norm, but some of the details of our approach will need to change. This issue will be discussed further in Appendix A (cf. supplementary materials). Let E_n denote the $n \times n$ matrix will all entries being 1.

The SLS-ID problem inputs the number of modes $m \geq 2$, data $D : \{(\mathbf{x}_i, \mathbf{x}'_i)\}_{i=1}^N$, absolute error bound: $\tau > 0$, relative error tolerance: $\epsilon > 0$ and a absolute limit γ. It outputs a set of matrices $A_1, \ldots, A_m$ that explains away D with error bound τ and tolerance ϵ and satisfying

$$-\gamma E_n \leq A_j \leq \gamma E_n \,,\ \forall\, j \in [m]\,,\ \text{i.e, each entry of } A_j \text{ lies in } [-\gamma, \gamma]\,, \tag{2}$$

or outputs INFEASIBLE if no such matrices exist. The SLS-ID problem is known to be NP-hard [21, Theorem 5.1]. We now formulate a mixed integer linear program for solving the SLS-ID problem.

We choose $m \times n^2$ decision variables which form the unknown entries of the matrices $A_1, \ldots, A_m$. We also use binary indicator variables $w_{i,1}, \ldots, w_{i,m} \in \{0, 1\}$ wherein $i = 1, \ldots, N$ ranges over each data item in D. The indicator variable $w_{i,j}$ denotes that the ith data item in D is explained away by the matrix A_j. Each data item must be explained away by one matrix, i.e., $\sum_{j=1}^m w_{i,j} = 1\,,\ \forall\, i \in [N]$. We add constraint (2) to set limits on the entries of the matrices A_j. Finally, we add the constraint that each data item is explained away by the corresponding matrix:

$$\|\mathbf{x}'_i - A_j \mathbf{x}_i\|_\infty \leq \epsilon \|\mathbf{x}_i\|_\infty + \tau + (1 - w_{i,j})M\,,\ \forall\, (\mathbf{x}_i, \mathbf{x}'_i) \in D,\ \forall\, j \in [m]\,. \tag{3}$$

Here, $M > 0$ is a constant that is larger than $\gamma\|\mathbf{x}_i\|_1 + \|\mathbf{x}'_i\|_\infty$ for all $(\mathbf{x}_i, \mathbf{x}'_i) \in D$, implying that (3) is trivially satisfied if $w_{i,j} = 0$ ("big-M" method).

The set of constraints above along with $w_{i,j} \in \{0, 1\}$ form the constraints of an MILP. The objective can be set in many ways: one common approach is to minimize the sum of the absolute value of the entries of each matrix, as a regularization term.

Lemma 1. *A set of matrices $A_1, \ldots, A_m$ is feasible for the MILP defined above (for some valuation of the binary variables $w_{i,j}$) iff they explain away the data D with error bound τ and tolerance ϵ.*

The MILP has mN binary variables and thus its solution using a common approach such as *branch-and-bound* will require solving 2^{mN} linear programs. This is exponential in the data size N ($= |D|$).

2.1 Identification with a Tolerance Gap

We now formulate a "relaxed version" of the SLS-ID problem with a "tolerance gap".

Definition 3 (Switched Linear System Identification with Gap (SLS-ID-GAP))**.** *We define the* SLS-ID-GAP *problem for given $\epsilon_2 > \epsilon_1 \geq 0$, $\tau > 0$ and $\gamma > 0$. Let $\epsilon_{\mathsf{gap}} : \epsilon_2 - \epsilon_1 > 0$.*

INPUTS: *Number of modes $m \geq 2$, Data $D : \{(\mathbf{x}_i, \mathbf{x}'_i)\}_{i=1}^N$, absolute error bound: $\tau > 0$,* two *relative error tolerances: $\epsilon_2 > \epsilon_1 \geq 0$ and limit: γ (for the constraint (2)).*

OUTPUT: FEASIBLE *along with a set of matrices $A_1, \ldots, A_m$ that explains away D with error bound τ and tolerance ϵ_2, or* INFEASIBLE *if no set of m matrices satisfying (2) with $\gamma - \epsilon_{\mathsf{gap}}$ can explain away the data with tolerance ϵ_1 and error bound τ.*

For any given data set D, let us define $\epsilon_{\min}(D)$ as the least tolerance ϵ for which it can be explained away with fixed error bound τ and limit γ. It is easy to show that $\epsilon_{\min}(D)$ is well defined for any data set D. Thus (ignoring the technicality related to the limit γ for now), algorithm for SLS-ID-GAP problem guarantees that (a) If $\epsilon_{\min}(D) \leq \epsilon_1$ then the algorithm will return FEASIBLE with matrices that explain the data with tolerance ϵ_2; (b) If $\epsilon_{\min}(D) > \epsilon_2$, algorithm will return INFEASIBLE; (c) If $\epsilon_{\min}(D) \in (\epsilon_1, \epsilon_2]$, the algorithm may return FEASIBLE or INFEASIBLE (without any guarantees). The main result of the paper is as follows:

Theorem 1. *There exists an algorithm for solving the* SLS-ID-GAP *problem with complexity $O\big(m^{Cmn^3|\log(\gamma/\epsilon_{\mathsf{gap}})|} N \mathsf{poly}(m, n)\big)$, wherein $\mathsf{poly}(m, n)$ denotes a polynomial function of m, n and C is a constant factor.*

The significance of this approach is that it is *linear* in the size of the data set N but exponential in the number of modes m and the dimension of the state space n.

3 Tree-Based Exploration Algorithm

We will now present our algorithm for finding matrices $A_1, \ldots, A_m$ that explain away the data, in stages. First, we present the algorithm as a tree-based exploration approach, wherein each node of

Algorithm 1: Algorithm to expand a non-terminal leaf node in the tree.

Data: Leaf node ν with polyhedra $P_1, \ldots, P_m$, data set U and assignment map μ.
Result: New leaf nodes, or matrices $Q_1, \ldots, Q_m$ that explain away all the data.

1 **Choose** candidate matrices (feasible solutions) $Q_1, \ldots, Q_m$ s.t. for all $j \in [m]$, $Q_j \in P_j$
2 **Find** $(\mathbf{x}_i, \mathbf{x}'_i) \in U$ s.t. for all $j \in [m]$, $\|\mathbf{x}'_i - Q_j \mathbf{x}_i\|_\infty > \epsilon_2 \|\mathbf{x}_i\|_\infty + \tau$
3 **if *not found*** **then**
4 **return** FEASIBLE, $(Q_1, \ldots, Q_m)$ `/* ` $Q_1, \ldots, Q_m$ `explain all the data */`
5 **else**
6 $\hat{U} \leftarrow U \setminus \{(\mathbf{x}_i, \mathbf{x}'_i)\}$ `/* Remove` $(\mathbf{x}_i, \mathbf{x}'_i)$ `from unassigned data. */`
7 **for** $j \in [m]$ **do**
  `/* Constrain polyhedron` P_j `s.t. matrix` A_j `explains` $(\mathbf{x}_i, \mathbf{x}'_i)$`. */`
8   $\widehat{P}_j \leftarrow P_j \cap \{A_j : \|\mathbf{x}'_i - A_j \mathbf{x}_i\|_\infty \leq \epsilon_2 \|\mathbf{x}_i\|_\infty + \tau\}$
9   **if** $\widehat{P}_j$ *is feasible* **then**
10   **Create** new child node ν_j of ν.
11   Associate ν_j with polyhedra $P_1, \ldots, P_{j-1}, \widehat{P}_j, P_{j+1}, \ldots, P_m$
12    $\hat{\mu}_j \leftarrow \mu \cup \{(\mathbf{x}_i, \mathbf{x}'_i) \mapsto j\}$ `/* Associate` $(\mathbf{x}_i, \mathbf{x}'_i)$ `with mode` j`. */`
13   Associate node ν_j with set $\hat{U}$ and map $\hat{\mu}_j$

the tree represents a set of m convex polyhedra. Subsequently, we will show how a careful choice of the tree exploration strategy will guarantee a bound on the maximum length of each branch of the tree. This, in turn, yields the desired algorithm and its complexity guarantee. The proofs of all the lemmas are outlined: detailed proofs are provided in Appendix B (cf. supplementary material).

The central data structure maintained by our algorithm is a tree T, wherein each node of the tree has the following associated information:

1. Convex polyhedra $P_1, \ldots, P_m$, represented as systems of linear inequalities, describing a set of possible solutions for $A_1, \ldots, A_m$ respectively.
2. A subset $U \subseteq D$ containing data points that have not been "assigned" a mode yet.
3. A assignment map $\mu : D \setminus U \to [m]$ mapping each data point in $D \setminus U$ to a mode in $[m]$.

The root of the tree is initialized by m convex polyhedra $P_1, \ldots, P_m$, wherein P_j encodes the limits on the entries of the matrix A_j: $-\gamma E_n \leq A_j \leq \gamma E_n$. The set U at the root is the full data set D and the associated map is the "empty map" since its domain is empty.

A node of the tree is said to be *terminal* if either one of the associated polyhedron P_j is empty (infeasible) or the associated data set is empty ($U = \emptyset$). Terminal nodes will always be leaves in the tree maintained by our algorithm.

Expanding a Node: Starting from the root node, our algorithm iteratively expands the tree T until termination. Each expansion step involves choosing a non-terminal leaf ν of the tree and carrying out the steps outlined in Algorithm 1. Expanding ν begins with choosing feasible solutions $Q_1, \ldots, Q_m$ from polyhedra $P_1, \ldots, P_m$, respectively (line 1). Next, we scan through all the "unassigned" data U to find a data point $(\mathbf{x}_i, \mathbf{x}'_i) \in U$ that cannot be explained away by $Q_1, \ldots, Q_m$ (line 2). If no such data point can be found then we have, in fact, explained away all the data using $Q_1, \ldots, Q_m$. Therefore, we can terminate (line 4). If a data point $(\mathbf{x}_i, \mathbf{x}'_i) \in U$ is found in the previous step, then we create m child nodes $\nu_1, \ldots, \nu_m$ for the current node. Each child ν_j will have associated polyhedra $P_1, \ldots, P_{j-1}, \hat{P}_j, P_{j+1}, \ldots, P_m$, wherein, for $j' \neq j$, $P_{j'}$ remains the same as in node ν, and the new polyhedron $\hat{P}_j$ is defined in order to force the matrix A_j to explain the data point $(\mathbf{x}_i, \mathbf{x}'_i)$ that could not be explained by the previous candidate matrices (line 8). Node ν_j will have associated data set $\hat{U} : U \setminus \{(\mathbf{x}_i, \mathbf{x}'_i)\}$ and mode assignment map $\hat{\mu}_j : \mu \cup \{(\mathbf{x}_i, \mathbf{x}'_i) \mapsto j\}$ (in other words, the newly assigned data point is assigned to mode j in node ν_j).

Thus, the process of expanding a node of the tree results either in termination with a feasible solution $Q_1, \ldots, Q_m$ that explains away the entire data, or in the possible addition of m new leaf nodes to the tree. We will now establish important properties of this algorithm.

Lemma 2. *Let $Q_1, \ldots, Q_m$ be the chosen candidates while expanding some node ν. For each child ν_j of ν with associated polyhedra $P_1, \ldots, P_{j-1}, \hat{P}_j, P_{j+1}, \ldots, P_m$, we have that $Q_j \notin \hat{P}_j$.*

The proof is to simply show that the constraints added to $\hat{P}_j$ in line 8 are not satisfied by Q_j.

The tree-based exploration algorithm is *complete*: if there is a set of matrices explaining away the data, then there is a leaf with associated polyhedra containing these matrices.

Lemma 3. *Suppose the set of matrices $A_1^*, \ldots, A_m^*$ explains away all the data in D with error bound τ and tolerance $\epsilon_1 \leq \epsilon_2$ and satisfies the limit constraint (2), then there exists a leaf node ν in the tree with associated polyhedra $P_1, \ldots, P_m$ such that for all $j \in [m]$, $A_j^* \in P_j$.*

Proof is by induction on the number of iterations of the algorithm. At the beginning, we show that the root node satisfies the conditions of the lemma. We prove that at each step if a leaf satisfying the conditions of the lemma is expanded, then one of the children "inherits" the property.

Thus far, the algorithm is nothing but a *branch-and-bound* tree-search algorithm which will continue to have a worst case complexity of 2^{mN} times the cost of solving linear programs with mn^2 variables and $O(Nn + mn^2)$ constraints. In the next subsection, we present a volume-based approach for pruning nodes in the tree, that will exploit the presence of the tolerance gap to reduce the complexity.

Volume-Based Pruning: Thus far, the tree exploration stops only if each leaf in the tree is terminal or a feasible set of matrices is discovered in the process of expanding some leaf node. We establish a bound $V_{\min}$ such that if for some leaf node and some $j \in [m]$, the associated polyhedron P_j satisfies $\mathsf{vol}(P_j) < V_{\min}$, then we can mark the node as terminal and remove it from further consideration.

Suppose the data $D : \{(\mathbf{x}_i, \mathbf{x}_i')\}_{i=1}^N$ has been generated from *ground truth* matrices $A_1^*, \ldots, A_m^*$ and an assignment map $\mu^* : D \to [m]$ such that for all $i \in [N]$, the mode $j : \mu^*(\mathbf{x}_i, \mathbf{x}_i') \in [m]$ satisfies $\|\mathbf{x}_i' - A_j^* \mathbf{x}_i\|_\infty \leq \epsilon_1 ||\mathbf{x}||_\infty + \tau$. I.e, they explain away the data with tolerance $\epsilon_1 < \epsilon_2$. In particular, $\mu^*(\mathbf{x}_i, \mathbf{x}_i')$ tells us which among the m matrices explains away the ith data item with error bound τ and tolerance ϵ_1. For technical reasons, we also assume that the ground truth matrices satisfy a "stronger version" of the limit constraint (2): $-(\gamma - \epsilon_{\mathsf{gap}})E_n \leq A_j^* \leq (\gamma - \epsilon_{\mathsf{gap}})E_n$, $\forall\, j \in [m]$.

By Lemma 3, at each iteration of the algorithm, there exists a leaf node ν in the tree that satisfies that for all $j \in [m]$, $A_j^* \in P_j$, wherein $P_1, \ldots, P_m$ are the polyhedra associated to ν. We now refine this property by showing that A_j^* is in the interior of P_j at a distance of at least ϵ_{gap} from the boundary. Recall the definition of the induced (operator) L_∞ norm of a matrix: $\|A\|_\infty^{\mathsf{ind}} : \max_{\|\mathbf{x}\|_\infty \leq 1} \|A\mathbf{x}\|_\infty$.

Note that $\|A\|_\infty^{\mathsf{ind}} = \max(\|A_{1:}\|_1, \ldots, \|A_{n:}\|_1)$ (= maximum among the L_1 norm of each row) [18]. Given $A \in \mathbb{R}^{n \times n}$ and $\epsilon \geq 0$, we let $B_\infty^{\mathsf{ind}}(A, \epsilon)$ be the ball of radius ϵ (w.r.t. $\|\cdot\|_\infty^{\mathsf{ind}}$) centered at A.

Lemma 4. *With $A_1^*, \ldots, A_m^*$ as above, at each iteration of the algorithm, there exists a leaf node ν in the tree with associated polyhedra $P_1, \ldots, P_m$ such that for all $j \in [m]$, $B_\infty^{\mathsf{ind}}(A_j^*, \epsilon_{\mathsf{gap}}) \subseteq P_j$.*

The crux of the proof is to show that if a matrix A_j satisfies $\|A_j - A_j^*\|_\infty^{\mathsf{ind}} \leq \epsilon_{\mathsf{gap}}$, then we can show that, for each data item $(\mathbf{x}_i, \mathbf{x}_i') \in D$ with $\mu^*(\mathbf{x}_i, \mathbf{x}_i') = j$, $\|\mathbf{x}_i' - A_j \mathbf{x}_i\|_\infty \leq \|\mathbf{x}_i' - A_j^* \mathbf{x}_i\|_\infty + \epsilon_{\mathsf{gap}} \|\mathbf{x}_i\|_\infty \leq \epsilon_2 \|\mathbf{x}\|_\infty + \tau$. As an immediate corollary, we have the following fact.

Lemma 5. *At each iteration of the algorithm, there exists a leaf node ν in the tree with associated polyhedra $P_1, \ldots, P_m$ such that for all $j \in [m]$, $\mathsf{vol}(P_j) \geq \frac{(2\epsilon_{\mathsf{gap}})^{n^2}}{(n!)^n}$.*

We note that any node containing $B_\infty^{\mathsf{ind}}(A_j^*, \epsilon_{\mathsf{gap}})$ will satisfy this minimum volume property. The ball $B_\infty^{\mathsf{ind}}(A_j^*, \epsilon_{\mathsf{gap}})$ is the product of n unit L_1-norm balls (one for each row) each scaled with a factor ϵ_{gap}. The rest follows from the well-known formula for a unit L_1-norm ball in n dimensions.

This motivates our main idea for pruning nodes in the tree. If the volume of a polyhedron P_j of a leaf node ν is less than $V_{\min} : \frac{(2\epsilon_{\mathsf{gap}})^{n^2}}{(n!)^n}$, then we consider the node to be terminal. We note that the ground truth matrices are definitely not part of such a node.

Cutting Plane Argument: We will now prove an effective bound on the depth of our tree using the so-called cutting plane argument from non-smooth optimization [9]. First, we refine line 1 of our algorithm wherein we choose candidates $Q_j \in P_j$ for $j \in [m]$. We will specifically choose Q_j as the center of the maximum volume inscribed ellipsoid (MVE) of P_j. The MVE center of a polyhedron can be computed efficiently by using semi-definite programming (SDP) [5, Proposition 4.9.1]. Now consider the child node ν_j of ν, wherein we have a polyhedron $\hat{P}_j \subsetneq P_j$. We show that the volume reduces by at least a factor $\alpha : (1 - \frac{1}{n^2}) < 1$.

Lemma 6. $\mathsf{vol}(\hat{P}_j) \leq (1 - \frac{1}{n^2})\mathsf{vol}(P_j)$.

The above follows from the classical cutting-plane argument [27] applied on the polyhedron $\hat{P}_j$, which is obtained by cutting P_j with a hyperplane excluding the MVE center.

Lemma 7. *The depth of the tree is* $O(mn^4 \log(n\gamma/\epsilon_{\mathsf{gap}}))$.

The proof observes that each time we expand a node in the tree, one polyhedron in the child node has its volume reduced by a factor α as stated in Lemma 6. Also, when the overall volume becomes smaller than $V_{\min}$ as in Lemma 5, the node is pruned. This places a bound on the depth of the tree, as stated in the lemma.

Theorem 2. *The overall size of the tree cannot exceed* $m^{O(mn^4 \log(\gamma/\epsilon_{\mathsf{gap}}))}$ *nodes, wherein the complexity of expanding each node is linear in the size* N *of the data set and involves solving* m *SDPs each with* n^2 *variables and* $O(mn^4)$ *constraints.*

In fact, the n^4 term in the exponent can be reduced to n^3 using the observation that each polyhedron P_j for each node ν is the cartesian product of n polyhedra $P_{j,k}$, for $k \in [n]$, involving the variables from the kth row of the unknown matrix A_j. This "fine-grained" analysis is provided in Appendix C.

4 Experimental Evaluation

Thus far, we have described our algorithm for solving the switched linear system identification problem. In this section, we will describe an evaluation of our approach meant to answer two key questions: (a) *Do the theoretical guarantees translate into superior empirical performance when compared to highly optimized MILP solvers?* We compare our approach against the MILP solver Gurobi [17] over a set of "microbenchmarks" of varying dimensions, number of modes and data sizes. Gurobi is available free but under restricted license for academic use. (b) *Does the approach yield interesting results on real-life datasets?* We illustrate our approach on handwriting and highway driving data-sets.

Implementation: We implemented the proposed approach in Python 3.8, using Gurobi [17] to encode and solve linear programs. We assume that the number of modes m (or an upper bound on it) is given. Also, our implementation fixes $\epsilon_1 = 0$ and $\epsilon_2 = \epsilon_{\mathsf{gap}} = \epsilon > 0$. Therefore, throughout this section, we will report the value of ϵ. Our implementation coincides with the description in Section 3 with one important modification: we compute the *Chebyshev* center instead of the center of the maximum volume inscribed ellipsoid (MVE). The Chebyshev center can be computed very efficiently and reliably using Linear Programming [8], and provides a good approximation of the MVE center. Although the theoretical guarantees on the termination of the process using the Chebyshev center are weaker than the ones with the MVE center (Lemma 6), the use of Chebyshev center in this context provides a powerful heuristic in practice [9, §4.4].

Evaluation on Microbenchmarks: We evaluate the proposed approach and the MILP approach on three microbenchmarks. For each benchmark, the linear dynamics at each mode are obtained by sampling a random Hurwitz matrix of appropriate dimensions. At each step, the mode is chosen uniformly at random. Further details are provided in Appendix D (cf. supplementary material). We generate $k \in \{1, 10, 100, 1000\}$ traces, each consisting of $T = 10$ time steps starting from an initial state uniformly sampled from the region $[-1, 1]^n$. Additionally, we add uniformly sampled noise in the range $[-\tau, \tau]$ to the states. We consider three noise amplitudes $\tau \in \{0.0, 0.05, 0.1\}$. As a result, each data set consists of $N = kT$ data points. We report the time taken by the MILP approach and the proposed approach (TS) and the L_∞ error between the predicted and actual states in Table 1. Each entry in the table reports average times and errors over 3 runs.

Table 1: Performance of MILP vs. our approach (TS) on a set of microbenchmarks. All timings are reported in seconds on a MACbook pro running OSX 10.15 with 16 GB RAM.

		$\tau = 0.00$				$\tau = 0.05$				$\tau = 0.10$			
	N	MILP Time [s]	MILP Error	TS Time [s]	TS Error	MILP Time [s]	MILP Error	TS Time [s]	TS Error	MILP Time [s]	MILP Error	TS Time [s]	TS Error
Benchmark #1	10	0.03	0.00	0.16	0.00	0.02	0.05	0.02	0.04	0.02	0.10	0.04	0.04
$n = 2, m = 3$	100	0.95	0.00	0.55	0.00	0.20	0.05	0.98	0.05	0.17	0.10	0.28	0.10
$\epsilon = 0.05$	1000	6.37	0.00	0.54	0.00	2.80	0.05	3.47	0.05	3.09	0.10	112.90	0.10
	10000	781.03	0.97	1.07	0.00	440.85	0.05	138.11	0.05	332.03	0.19	461.58	0.10
Benchmark #2	10	0.07	0.16	0.14	0.02	0.06	0.29	0.09	0.07	0.07	0.35	0.03	0.25
$n = 3, m = 5$	100	0.69	1.76	0.64	1.02	0.58	1.84	0.47	0.99	0.57	1.25	0.36	0.96
$\epsilon = 0.1$	1000	6.54	1.46	172.17	1.22	7.08	1.40	0.99	1.09	7.43	1.79	0.96	1.16
	10000	3075.34	3.1	39.45	2.87	2067.17	3.38	341.96	3.3	10098.96	3.16	510.71	3.44
Benchmark #3	10	0.11	4.20	0.17	1.80	0.12	4.25	0.21	1.61	0.12	4.23	0.20	1.41
$n = 5, m = 8$	100	1.78	15.13	0.17	4.02	1.94	12.98	0.19	4.08	1.94	18.22	0.18	4.14
$\epsilon = 5$	1000	16.41	63.05	0.25	71.36	16.56	65.42	0.23	72.28	17.51	68.10	0.22	73.2
	10000	845.29	64.00	0.58	71.36	859.01	54.74	0.56	72.28	831.64	63.56	0.56	73.2

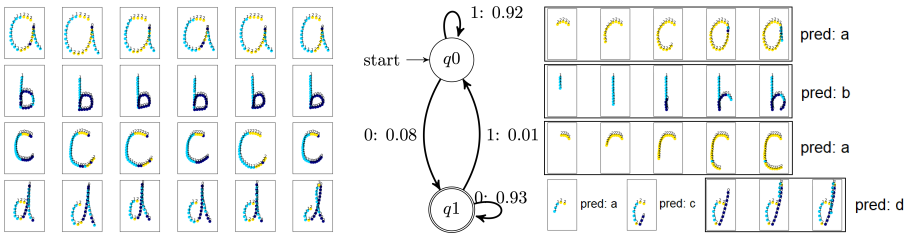

Figure 1: **Left:** Handwriting samples with identified modes shown in different colors, **Middle:** JMM model for letter b and **Right:** predicting letters from partial observations using the JMM.

Table 1 shows that the MILP approach rapidly becomes more expensive as we increase the number of data points N. The time taken for $N = 10000$ by our approach is at most 500 seconds, whereas the MILP approach takes up to 10000 seconds. The errors reported in the Table are dependent on the value of ϵ, and are comparable for both approaches. The microbenchmarks provide preliminary confirmation that the theoretical results do bear out in practice, especially when compared against a highly optimized MILP solver whose performance rapidly degrades as the number of data points increases, in contrast to our approach whose performance is affected more by the number of modes and the dimension of the state space rather than the number of data points.

Handwriting Recognition: We now evaluate our approach on a human handwriting data on a tablet: our goal is to identify various modes with dynamics that describe how letters are traced out on the tablet. We generated our own handwriting data set by having an author trace out the letters $a - d$ using their fingers on their mouse pad of their laptop. We collected the (x, y) locations of the handwritten letters over time. We generated our final data set by interpolation in order to ensure that the samples are roughly equidistant from each other. We also scaled the (x, y) values so that the letters fit within a square of side length 1. The process of data generation is described in detail in Appendix E. We collected 10 samples for each letter (Fig. 1 left).

We applied our approach to learn $m = 3$ matrices that explain away the data with $\epsilon = 0.05$ and $\tau = 0.1$. For each sample, we obtain a corresponding switching sequence from this process. We apply a well-known grammatical inference approach to learn a jump markov model (JMM), wherein the various dynamical modes are represented on the edges of the automaton [10, 31]. The estimated mode for each data point of each letter is represented in Fig.1. Each color represents a different mode. We note that the modes naturally corresponds to different parts of the letters.

Fig. 1 (middle) shows an example of JMM for the letter b. Each transition in the JMM represents a mode and a probability of taking the edge. At state q_0, Mode 1 is observed for 92% of the time, and at q_1, Mode 0 is observed for 93% of the time. We then tried to predict a class for a new sampled handwritten letter, based on the estimated JMMs (Fig. 1 (right)). Each column represents 20%,

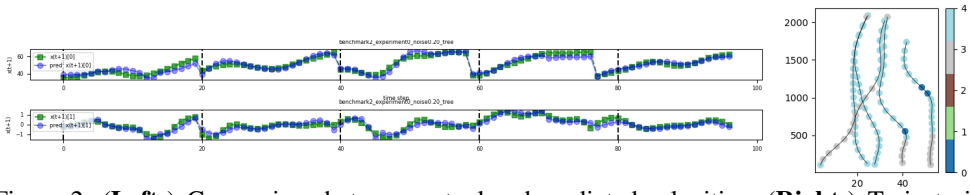

Figure 2: (**Left:**) Comparison between actual and predicted velocities. (**Right:**) Trajectories with estimated modes in color.

40%, 60%, 80%, 95% part of a letter. We use the JMMs and Bayesian inference to compute the probability of a letter given a partial sequence of states. The letters a, b, d are correctly predicted but the letter c is classified as an a: a visual inspection of Fig 1 (right) explains why this is the case.

In conclusion, we show that our switched linear system identification algorithm works on the noisy handwriting data and can be used to identify a JMM from sequences of identified modes which in turns can be useful for data classification and prediction.

Highway Data-set: In this experiment, our objective is to model a human driver's choice of acceleration based on the behavior of the vehicles in front and behind the ego car. Our goal is to find out if the modes of the switching system can naturally correspond to different qualitative behaviors such as staying in the same lane or switching lanes. For this purpose, we used the publicly available US-101 human driver dataset [29]. The data set contains vehicle IDs, coordinates and velocities over time. We model the choice of velocity at next time step as a switched linear system, where we assume that all mode dynamics are linear in the car's forward velocity v_y and lateral velocity v_x, the preceding car's forward velocity v_y^p and the distance d to the preceding car.

$$v_x(t+1) = f_x(v_x(t), v_y(t), v_y^p(t), d(t)),\ v_y(t+1) = f_y(v_x(t), v_y(t), v_y^p(t), d(t))\,,$$

where f_x, f_y are unknown linear functions over their arguments. We consider several modes described by different functions f_x, f_y to model how the velocity choice depends on the driving context such as going straight, turning for changing lane, or the preceding car changing lanes.

The parameters we used for the algorithm are: $m = 5$, $\epsilon = 10^{-3}$ and $\tau = 0.2$. The identified modes as well as the comparison between the predicted and actual outputs are shown in Figure 2. We note that the model predicted velocities are accurate estimates of the actual velocities. Figure 2 shows five vehicle trajectories, color coded with the identified modes. Three modes out of the five are actually used and each mode represents a qualitatively different behavior of the driver (stay/change lanes). In conclusion, we show that our algorithm can be used to identify various decision making modes from actual driving data.

5 Conclusions: Limitations, Future Work and Broader Impacts

In summary, we have presented an approach that improves the time complexity of switched linear system identification using a novel problem formulation with a "gap" and using ideas from the ellipsoidal method in combinatorial optimization. The resulting complexity is linear in the number of data points. Limitations include (a) Our approach uses full state information and extends to output observations using SARX models. However, the number of modes in the SARX model can be exponentially larger when compared to the state-space model. (b) Our approach finds the matrices explaining away dynamics but does not infer the generator for these matrices, although we present an example using Jump Markov Models. Therefore, our approach may find a solution that may lead to unsatisfactory hybrid system models. However, we may use backtracking in our approach to generate further solutions; (c) Finally, our implementation performs the tree exploration without leveraging other information about the system or using heuristics that have proved powerful in branch-and-bound algorithms. Nevertheless, it provides a proof of concept that it can be competitive against highly-optimized MILP solvers. We will expand on these ideas in our future work.

In terms of broader impacts, model identification is a very important machine learning problem that has very important and beneficial applications in areas such as controls, autonomous systems and medical applications. However, our approach can be used to improve tasks that may prove detrimental to individuals and societies such as improving existing surveillance systems.

References

[1] Rajeev Alur. *Principles of Cyber-Physical Systems*. MIT Press, 2015.

[2] Aaron D. Ames. Human-inspired control of bipedal walking robots. *IEEE Transactions on Automatic Control*, 59(5):1115–1130, 2014.

[3] E. Bartocci, F. Corradini, M. R. Di Berardini, E. Entcheva, S. A. Smolka, and R. Grosu. Modeling and simulation of cardiac tissue using hybrid i/o automata. *Theor. Comput. Sci.*, page 3149–3165, aug 2009.

[4] Alberto Bemporad, Andrea Garulli, Simone Paoletti, and Antonio Vicino. A bounded-error approach to piecewise affine system identification. *IEEE Transactions on Automatic Control*, 50(10):1567–1580, 2005.

[5] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2001.

[6] Andrew Blake, Ben North, and Michael Isard. Learning multi-class dynamics. In *Advances in Neural Information Processing Systems*, volume 11. MIT Press, 1998.

[7] Luca Bortolussi and Alberto Policriti. Hybrid systems and biology. In Marco Bernardo, Pierpaolo Degano, and Gianluigi Zavattaro, editors, *Formal Methods for Computational Systems Biology*, pages 424–448, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[8] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[9] Stephen Boyd and Lieven Vandenberghe. Localization and cutting-plane methods. *From Stanford EE 364b lecture notes*, 2007.

[10] Colin De la Higuera. *Grammatical inference: learning automata and grammars*. Cambridge University Press, 2010.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[12] Giancarlo Ferrari-Trecate, Marco Muselli, Diego Liberati, and Manfred Morari. A clustering technique for the identification of piecewise affine systems. In *Hybrid Systems: Computation and Control*, page 218–231. Springer-Verlag, 2001.

[13] Emily Fox, Erik Sudderth, Michael Jordan, and Alan Willsky. Nonparametric bayesian learning of switching linear dynamical systems. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

[14] Zoubin Ghahramani and Geoffrey E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864, 2000.

[15] Ronojoy Ghosh, Keith Amonlirdviman, and Claire J. Tomlin. A hybrid system model of planar cell polarity signaling in drosophila melanogaster wing epithelium. In *IEEE Conference on Decision and Control (CDC)*, pages 1588–1594. IEEE, 2002.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[17] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022.

[18] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[19] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 2010.

[20] Fabien Lauer. Estimating the probability of success of a simple algorithm for switched linear regression. *Nonlinear Analysis: Hybrid Systems*, 8:31–47, 2013.

[21] Fabien Lauer and Gérard Bloch. Hybrid system identification: Theory and algorithms for learning switching models. *Springer*, 2019.

[22] Daniel L. Ly and Hod Lipson. Learning symbolic representations of hybrid dynamical systems. *Journal of Machine Learning Research*, 13(115):3585–3618, 2012.

[23] Necmiye Ozay. An exact and efficient algorithm for segmentation of arx models. In *American Control Conference (ACC)*, pages 38–41, 2016.

[24] Necmiye Ozay, Constantino Lagoa, and Mario Sznaier. Robust identification of switched affine systems via moments-based convex optimization. pages 4686–4691, 12 2009.

[25] Sadra Sadraddini and Calin Belta. Formal guarantees in data-driven model identification and control synthesis. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*, pages 147–156, 2018.

[26] Miriam García Soto, Thomas A Henzinger, and Christian Schilling. Synthesis of hybrid automata with affine dynamics from time-series data. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.

[27] S. P. Tarasov, L. G. Khachiyan, and I. I. Èrlikh. The method of inscribed ellipsoids. In *Dokl. Akad. Nauk SSSR*, volume 37, pages 226–230, 1988.

[28] Jitendra K. Tugnait. Detection and estimation for abruptly changing systems. *Automatica*, 18(5):607–615, 1982.

[29] US Federal Highway Administration (FHWA). US-101 human driven vehicle trajectory dataset.

[30] René Vidal, Stefano Soatto, Yi Ma, and Sankar Sastry. An algebraic geometric approach to the identification of a class of linear hybrid systems. In *42nd IEEE International Conference on Decision and Control*, volume 1, pages 167–172 Vol.1, 2003.

[31] Kandai Watanabe, Nicholas Renninger, Sriram Sankaranarayanan, and Morteza Lahijanian. Probabilistic specification learning for planning with safety constraints. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6558–6565. IEEE, 2021.

[32] Siep Weiland, A Lj Juloski, and B Vet. On the equivalence of switched affine models and switched ARX models. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 2614–2618. IEEE, 2006.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] This was discussed partly in the related work and partly in the concluding Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] see Section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] The proofs are sketched in the main paper but provided in full details as part of the appendix.
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We averaged across multiple runs. Error bars are not informative for our empirical evaluation.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A] Our experiments each ran on a single CPU on a desktop. We did not use GPUs, clusters or cloud computing resources.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] Our approach is built on top of Gurobi [17] optimization tool.

(b) Did you mention the license of the assets? [Yes]

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We are using publicly available data that can be downloaded by everyone freely.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]