

# A Machine Learning Theory Perspective on Strategic Litigation

Melissa Dutz  

Toyota Technological Institute at Chicago, IL, USA

Han Shao<sup>1</sup>   

University of Maryland, College Park, MD, USA

Avrim Blum   

Toyota Technological Institute at Chicago, IL, USA

Aloni Cohen   

The University of Chicago, IL, USA

---

## Abstract

*Strategic litigation* involves bringing a case to court with the goal of having an impact beyond resolving the particular dispute at hand. In a common law system, one way a case may have far-reaching impact is by establishing new legal precedent that later courts must follow. In this paper, we explore strategic litigation from the perspective of machine learning theory. We consider an abstract model of a common law legal system where a lower court decides new cases by applying a decision rule learned from a higher court's past rulings. In this model, we explore the power of a *strategic litigator*, who strategically brings cases to the higher court to influence the decision rule applied by the lower court in future cases. We explore questions including: What impact can a strategic litigator have? Which cases should a strategic litigator bring to court? Does it ever make sense for a strategic litigator to bring a case when they are sure the court will rule against them? We show that this strategic case selection problem has interesting structure, with even simple settings exhibiting counterintuitive phenomena. When cases are represented by points in one dimension and the lower court's learning algorithm is nearest neighbor, or as points in  $d$  dimensions and the lower court's learning algorithm is a support vector machine, we characterize the set of inducible decision rules and develop algorithms for selecting an optimal set of cases to bring to the higher court given the strategic litigator's objectives.

**2012 ACM Subject Classification** Theory of computation → Machine learning theory; Theory of computation → Algorithmic game theory

**Keywords and phrases** Strategic Litigation, Machine Learning Theory, Law

**Digital Object Identifier** 10.4230/LIPIcs.FORC.2026.19

**Funding** *Melissa Dutz*: Melissa Dutz was supported in part by an NSF CSGrad4US fellowship under Grant No. 2313998. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. National Science Foundation.

*Han Shao*: Han Shao was supported in part by Harvard CMSA.

*Avrim Blum*: Avrim Blum was supported in part by the National Science Foundation under grant CCF-2212968, and by ECCS-2216899. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. National Science Foundation.

---

<sup>1</sup> Work mostly done while at Toyota Technological Institute at Chicago



© Melissa Dutz, Han Shao, Avrim Blum, and Aloni Cohen;  
licensed under Creative Commons License CC-BY 4.0

7th Symposium on Foundations of Responsible Computing (FORC 2026).

Editor: Huijia (Rachel) Lin; Article No. 19; pp. 19:1–19:17



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Strategic litigation (also called impact litigation) involves bringing a legal case to court with the goal of creating some broader societal impact outside of resolving the case itself: for example, shaping future law. In a common law system, a decision on one case can establish precedent that future cases must follow. Proponents of a particular legal doctrine might therefore want to carefully select a case likely to be ruled on favorably, so that similar but perhaps more ambiguous cases will be decided favorably as well. A famous example of strategic litigation is *Brown v. Board of Education*; although this case concerned school desegregation specifically, it was crafted to undermine the “separate but equal” doctrine of racial discrimination much more broadly.

The United States has a hierarchical court system in which higher courts’ rulings establish binding precedent that lower courts must follow. Lower courts must somehow determine how precedent applies to new cases. This process can be viewed as the lower courts learning a decision rule from precedent: a way to map information about a new case to a ruling based on precedent. This perspective motivates studying the application of precedent through the lens of learning theory, the study of learning functions from data.

In this paper, we make an initial foray into studying strategic litigation from a machine learning theory perspective. We introduce a highly stylized, abstract model of a common-law legal system with three actors: a high court, a lower court, and a strategic litigator. The lower court decides new cases by applying a decision rule learned from a higher court’s past rulings. The strategic litigator strategically brings cases to the higher court to influence this learned decision rule, thereby affecting future cases. We explore questions including: What impact can a strategic litigator have? Which cases should a strategic litigator bring to court? Does it ever make sense for a strategic litigator to bring a case when they know the court will rule against them? Under two variants of our model, we characterize the set of decision rules a strategic litigator can induce, and provide algorithmic results for selecting an optimal set of cases to bring to court given the strategic litigator’s objectives. Of course, studying such a simplistic model cannot necessarily tell us anything about the US court system in all its complexity. However, we hope it can still offer some insight into the dynamics of strategic litigation.

This work has connections to other topics in machine learning theory, including teaching dimension ([3], [5]) and clean-label data attacks ([8], [1]). Previous works have also modeled legal decision making as a classifier (e.g., [6], [2], [4]) but to our knowledge this is the first exploration of strategic impact litigation in a machine learning setting. It will be helpful to present our model before discussing these connections in more detail, so we will revisit them in Section 3.

## 2 Setting

In this section, we describe our basic abstract model of the legal system, specify the instantiations of our model that we will use as case studies throughout this paper, and give an example illustrating strategic litigation under our model.

### 2.1 Abstract Model of the Legal System

We introduce our basic abstract model of cases, courts and strategic litigators below. Our model does not aim to realistically represent a real-world court system. Instead, our goal is to describe a simple model of a common-law legal system in which strategic litigation is possible. This allows us to study the dynamics of strategic litigation with minimal confounding complexity. We will extend our model in Section 6 to consider overturning precedent.

**Modeling Case Information.** The key facts of cases are referred to as *case fact patterns*. We represent case fact patterns as data points in some instance space.

**Modeling the Courts.** We consider a simple two-level court system comprised of a high court and lower courts. Courts act as binary classifiers on case fact patterns, with positive and negative labels corresponding to decisions for and against the plaintiff, respectively. The high court and the low court decide cases differently. The high court decides cases according to some binary labeling function  $f^*$ . The lower court learns a binary labeling function  $f$  from precedent (those cases already decided/labeled by the higher court) using a deterministic learning algorithm  $\mathcal{A}$ . Denoting the set of historical labeled data  $S_h = \{(x_1, f^*(x_1)), (x_2, f^*(x_2)), \dots, (x_n, f^*(x_n))\}$ , the classification function learned by the lower courts is  $f \leftarrow \mathcal{A}(S_h)$ . Note that each time a new case is decided by the high court, the set of precedent  $S_h$  is updated and the function learned by the lower courts is updated accordingly. We assume that  $\mathcal{A}$  always produces a function consistent with all precedent (e.g.  $f(x) = y \forall (x, y) \in S_h$ ). Finally, we assume that  $f^*$  is realizable, meaning that  $f^*$  is in the hypothesis class  $\mathcal{H}$  of functions that  $\mathcal{A}$  returns.

**Modeling a Strategic Litigator.** We assume a strategic litigator has some goal function  $g \in \mathcal{H}$  which describes how they would like cases to be decided, and we assume they know the high court's labeling function  $f^*$  and the learning algorithm  $\mathcal{A}$  used by the lower courts. Given  $g$ , the strategic litigator would like to bring some cases to be decided by the high court, such that the precedent they create influences the lower courts to rule on new cases according to  $g$ .

More precisely, the strategic litigator would like to select a set of case fact patterns to be labeled by  $f^*$  (call this labeled set  $S_l$ ), such that the function learned by the lower courts based on the union of the existing labeled set  $S_h$  and the added set  $S_l$  is close to their goal function  $g$ . Note that in general, it may not be possible to force  $\mathcal{A}$  to output exactly  $g$ . Therefore, the general goal of the strategic litigator is to select a set of case fact patterns such that the function  $f$  output by  $\mathcal{A}$  minimizes classification error (as defined by  $g$ ) with respect to some distribution  $\mathcal{D}$  of case fact patterns that the strategic litigator cares about. Formally:

$$\text{Given } f^*, g, S_h, \text{ find } S_l \text{ minimizing } \mathbb{E}_{x \sim \mathcal{D}} [\mathbb{1}\{g(x) \neq f(x)\}] \text{ where } f \leftarrow \mathcal{A}(S_h \cup S_l). \quad (1)$$

We primarily focus on the setting where the strategic litigator can select cases from some finite pool of case fact patterns,  $P$ . This captures the fact that only a finite set of existing cases can be brought to court (one cannot litigate a hypothetical case). However, we will also discuss the case where the strategic litigator can bring *any* point in the instance space when this enables more interesting results.

## 2.2 Case Studies

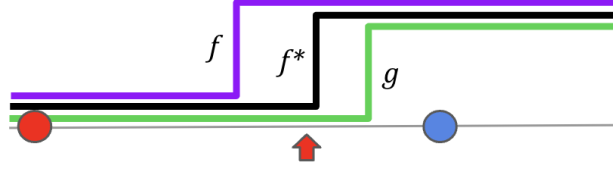
Throughout this paper, we will focus on two particular instantiations of our model:

1. **1-Dimensional Nearest Neighbor Setting:** Case fact patterns are represented by points in 1-dimensional space, the hypothesis class  $\mathcal{H}$  consists of all Boolean functions over  $\mathbb{R}$  that make a finite number of alternations between positive and negative labels, and the lower court's learning algorithm  $\mathcal{A}$  is a *nearest neighbor classifier*, which labels a new point the same way as the closest point in its training data.

2.  **$d$ -Dimensional Linear Separator Setting:** Case fact patterns are represented by points in  $d$ -dimensional space, the hypothesis class  $\mathcal{H}$  consists of linear separators, and the lower court's learning algorithm  $\mathcal{A}$  is a *support vector machine* (SVM). Recall that SVM finds a maximum-margin linear separator (a separator which maximizes the distance between itself and the closest point from each class). The position of the separating hyperplane which SVM learns is completely determined by training points which lie on the margin (e.g., the points closest to the separator), which are called support vectors.

We study nearest neighbor classifiers and maximum-margin linear separators because they are simple and natural heuristics for decision-making based on historical data. For example, nearest neighbor captures the intuition that similar cases should be decided similarly. A linear separator is natural as it is the simplest way to divide a  $d$ -dimensional space into two classes, such as winning and losing cases. By maximizing the distance between the boundary and the most ambiguous “edge cases” in each class, maximum-margin linear separators are intuitively robust among possible linear separators.

Figure 1 shows a simple example of strategic litigation in our 1-Dimensional Nearest Neighbor setting.



■ **Figure 1**  $f^*$  is the labeling function used by the high court,  $f$  is the output of Nearest Neighbor given the two historical points labeled by  $f^*$  (the red point is negative, the blue point is positive), and  $g$  is the strategic litigator's goal function. To make Nearest Neighbor output  $g$ , the strategic litigator can bring a point where indicated by the red arrow, which will be labeled negative by  $f^*$ .

### 3 Related Work

As mentioned earlier, our work is related to the concept of the *teaching dimension* in learning theory. Introduced by [3], the teaching dimension measures the complexity of *teaching*. Suppose a teacher wants to select the most helpful examples to efficiently teach any student a concept in a given concept class; the teaching dimension measures the minimum number of examples required. More formally, the teaching dimension is the minimum number of examples necessary to teach *any* consistent learning algorithm a target function  $f_t$  from a given hypothesis class  $\mathcal{H}$  (e.g., it is the minimum number of examples such that  $f_t$  is the only remaining consistent function in  $\mathcal{H}$ ). In our setting, the strategic litigator faces a similar problem to the teacher: they would like to find a small set of examples to “teach”  $g$  to the lower court's learning algorithm  $\mathcal{A}$ . This differs from the classic teaching dimension setting in two main ways. First, in the classic teaching setting, examples are labeled by the teacher's target function  $f_t$ . In our case, examples are not labeled by the strategic litigator's target function  $g$  – they are labeled by  $f^*$ . Second, the classic teaching problem requires a set of examples which will teach *any* consistent learner, while in our model the strategic litigator can focus on teaching a particular, known learner ( $\mathcal{A}$ ). More recent work ([5]) has studied the teaching dimension given a known learner, including SVM. In particular, [5] show that the number of examples required to teach any particular linear separator to SVM is 2. This corresponds to our setting when  $\mathcal{A}$  is SVM in the special case where  $f^* = g$ , there is no historical data, and the strategic litigator can select any case fact pattern in the instance space.

This work is also connected to work on clean label data poisoning attacks (see [8], [1]). In a clean-label data poisoning attack, an attacker seeks to cause the learning algorithm to make a mistake on a target test instance by injecting strategically chosen, correctly-labeled examples into the algorithm’s training set. In our work, the strategic litigator similarly seeks to manipulate the output of a learning algorithm ( $\mathcal{A}$ ) by injecting correctly labeled examples (case fact patterns labeled by  $f^*$ ) into its training set. The clean-label data poisoning model would correspond to a variant of our setting in which the strategic litigator’s goal is to cause the lower courts to rule favorably on a particular future case, rather than trying to bring the function  $\mathcal{A}$  learns close to some goal function  $g$  overall.

A number of previous works such as [6], [2] and [4] have also modeled legal decisions as the output of some learning algorithm; the key novelty of our work lies in exploring strategic impact litigation under a learning model. Like us, [6] model legal decision making as a nearest neighbor classifier; they also consider in more detail how to represent case fact patterns as points in an instance space. The work of [4] is perhaps most similar to ours in this vein: they also provide a machine learning perspective on the common law system and model strategic agents within this system (although these agents are not strategic litigators in the sense of strategic impact litigation we are focused on). As in our setting, [4] models courts as deciding cases brought by strategic agents according to some decision rule, but unlike in our setting, this rule is unknown. The agents in [4] are utility maximizers. They choose to bring a case or settle out of court depending on which has higher expected utility, taking account of the cost of litigation, utility of various outcomes, and the agents’ posterior over the decision rule given past cases. The paper explores whether the system as a whole converges to the unknown decision rule. Our focus on strategic impact litigation requires a different model. First, we consider a hierarchical system with a higher court establishing precedent for a lower court to follow. Second, the agent’s goal is to affect the lower court’s decision rule for future cases. The utility associated with the individual case is not relevant.

## 4 Strategic litigation in one dimension

Despite its simplicity, the 1-Dimensional Nearest Neighbor setting is already quite rich. We begin by highlighting some interesting structural observations exhibited by our model, even when the litigator is able to bring any fact pattern in the instance space as a case. Then we give a dynamic programming algorithm for optimally selecting cases from a finite set of fact patterns.

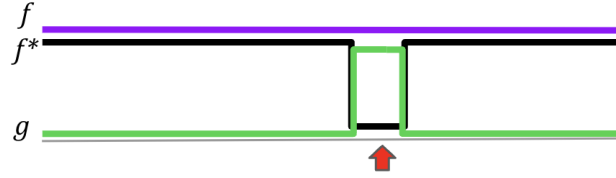
We adopt a geometric view that simplifies the exposition throughout this section. Recall that the strategic litigator’s with goal function  $g$  wants to minimize  $\mathbb{E}_{x \sim \mathcal{D}}[\mathbb{1}[g(x) \neq f(x)]]$ , where  $f \leftarrow \mathcal{A}(S_h \cup S_l)$  is the classifier applied by the lower court and  $\mathcal{D}$  is some distribution of case fact patterns that the strategic litigator cares about. In 1 dimension, we can view the probability mass under  $\mathcal{D}$  as a measure of distance. By doing so, the function the strategic litigator wants to minimize is the total length of all intervals along which  $f$  and  $g$  disagree. We call this quantity the *discrepancy*.

### 4.1 Structural Observations

In these examples, we assume the instance space is the real interval  $[0, 1]$  for simplicity, and we assume the strategic litigator is able to bring any fact pattern in the instance space as a case. Without loss of generality, we assume the output of Nearest Neighbor when there is no historical data is the all-positive function. Recall that  $f^*$  is the labeling function of the high court,  $f$  is the output of  $\mathcal{A}$  (Nearest Neighbor) on any historical points labeled by  $f^*$ , and  $g$  is the goal function of the strategic litigator.

## 19:6 A Machine Learning Theory Perspective on Strategic Litigation

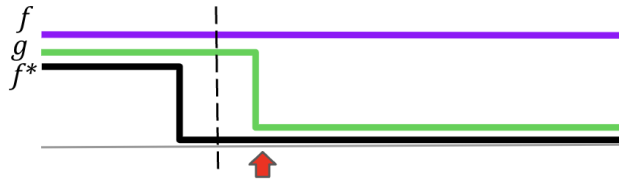
► **Observation 1.** *It can be arbitrarily helpful for the strategic litigator to bring a case where the high court will rule against them.*



■ **Figure 2** An example illustrating Observation 1: choosing a losing case (indicated by the red arrow) brings the classifier produced by Nearest Neighbor much closer to the strategic litigator's goal function.

For example, in Figure 2,  $f^*$  and  $g$  disagree on every point, so no matter which case the strategic litigator brings, the high court will rule against them. However, since  $f$  is all-positive and  $g$  is mostly negative, it benefits the strategic litigator to bring a case which will be classified negatively (such as the point indicated by the red arrow). This will cause Nearest Neighbor to output the all-negative function, which is much closer to  $g$  than the current all-positive  $f$ . Choosing this point could have arbitrarily high benefit as the positive region of  $g$  (and the corresponding negative region of  $f^*$ ) makes up a smaller and smaller fraction of the overall interval.

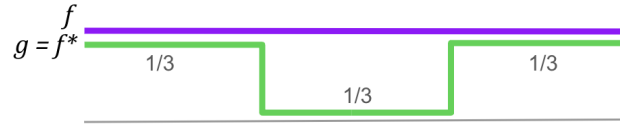
► **Observation 2.** *Choosing a case myopically can cause unnecessary, irreversible error.*



■ **Figure 3** An example illustrating Observation 2: greedily choosing a currently-optimal case (indicated by the red arrow) causes irreversible error that could have been avoided by planning ahead.

One natural question is whether a strategic litigator can use a simple greedy strategy, choosing cases one at a time by selecting the single most helpful case at each step. Figure 3 shows that such a strategy can create unnecessary, irreversible error. In particular, since the current  $f$  is all-positive, bringing any positive point does not alter  $f$ , but bringing any negative point will bring  $f$  closer to  $g$  since it will flip  $f$  to the all-negative function. Myopically, then, the optimal choice is any negative point. Suppose the strategic litigator even tie-breaks among these options in a smart way by choosing one of the negative points which  $g$  would also label negatively. The strategic litigator might then choose the negative point indicated by the arrow. Choosing this point would create irreversible error: no matter what positive points they bring in the future, the strategic litigator can no longer bring the boundary of  $f$  further to the right than the dashed line. This error is unnecessary as it would have been possible to exactly achieve  $g$  (by bringing any positive point to the left of  $g$ 's boundary, and a negative point an equal distance to the right of  $g$ 's boundary).

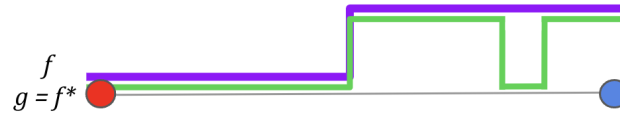
► **Observation 3.** *1-step lookahead is also not enough to be optimal.*



■ **Figure 4** An example illustrating Observation 3: greedily choosing a currently-optimal pair of cases cannot improve the strategic litigator’s error, even though the strategic litigator’s goal function can be achieved.

One might observe that choosing the best pair of points would be sufficient to avoid the problem in the previous example. Since 2 points define a boundary under Nearest Neighbor, it’s natural to wonder whether choosing the best pair of points might be enough to overcome the pitfalls of myopia in general. However, even looking ahead to choose the best pair of points, the strategic litigator can fail to minimize their error. For example in Figure 4, no pair of points can improve the strategic litigator’s error, so a strategic litigator who chooses pairs of points myopically will not choose any, failing to achieve  $g$  even though it is possible (since  $g = f^*$ ). If the strategic litigator brings 2 positive points,  $f$  does not change. If the strategic litigator brings 2 negative points,  $f$  will become the all-negative function, making the strategic litigator worse off. If the strategic litigator brings a positive and a negative point,  $f$  will become some function with a single boundary between a negative and positive region. At best, the strategic litigator will improve with respect to the entire negative third of  $g$  by making this region of  $f$  negative, but  $f$  must then be negative throughout one of the neighboring positive thirds of  $g$ , so the total discrepancy between  $f$  and  $g$  does not change.

► **Observation 4.** *The strategic litigator might need to make things worse in the short term in order to make things better in the longer term.*



■ **Figure 5** An example illustrating Observation 4: bringing any individual case increases the strategic litigator’s error, even though the strategic litigator’s goal function can be achieved by bringing several cases. The red and blue dots represent historical negative and positive points, respectively.

In the example shown in Figure 5, bringing any individual case makes the strategic litigator worse off (it increases the discrepancy between  $f$  and  $g$ ), but bringing several cases benefit the strategic litigator (by making  $f$  equal  $g$ ). Bringing any negative point will shift the boundary of  $f$  to the right, increasing the discrepancy between  $f$  and  $g$ ; any positive point will shift the boundary of  $f$  to the left, increasing the discrepancy between  $f$  and  $g$ . However, since  $g = f^*$ , the strategic litigator can make  $f$  match  $g$  exactly by bringing several points. Note that this also highlights the pitfalls of a myopic strategy, since a myopic strategic litigator would not bring any points at all, failing to achieve  $g$  even though it is possible.

## 4.2 Optimally selecting cases in one dimension

This section presents a dynamic programming algorithm for optimally selecting a subset of cases  $X$  from a finite pool of fact patterns  $P$  for the 1-Dimensional Nearest Neighbor setting, given labeled historical precedent  $S_h$ , goal function  $g$ , and distribution  $\mathcal{D}$  over case

fact patterns. The high level intuition is as follows: when choosing a set of points one by one from left to right, under Nearest Neighbor the error of the solution after adding a new point  $p$  is unchanged from the previous solution over the region from the left boundary of the domain to the midpoint  $m$  between  $p$  and the closest point to its left. To the right of  $m$ , the error is now determined only by  $p$  and any relevant historical precedent (points from  $S_h$  to the right of  $p$ ). This allows us to build an error-minimizing solution iteratively.

► **Theorem 5.** *In the 1-Dimensional Nearest Neighbor setting, for any goal function  $g$ , set of case fact patterns  $P$ , distribution  $\mathcal{D}$  over  $\mathbb{R}$ , set of historical precedent  $S_h$ , a strategic litigator can select a subset of case fact patterns  $\{x_1, x_2, \dots\} \subseteq P$  such that  $f \leftarrow \mathcal{A}(S_h \cup \{(x_1, f^*(x_1)), \dots\})$  minimizes  $\mathbb{E}_{x \sim \mathcal{D}}[\mathbb{1}[g(x) \neq f(x)]]$  over all possible subsets of  $P$  in  $O(|P|^2)$  calls to a nearest-neighbor oracle and a discrepancy oracle which computes the error of the current function with respect to  $g$ .*

The proof of Theorem 5 can be found in Appendix A.

► **Remark 6.** If the strategic litigator also wants to limit the number of cases they bring, Algorithm 1 can be easily extended to find the optimal set of at most  $k$  case fact patterns. We can simply extend the DP table to have  $k$  rows of length  $p$  instead of a single one.  $T[j][i]$  would then store the minimum error achievable with at most  $j$  points when the  $i$ th point is the rightmost point selected, and the table could be filled row by row, from left to right.

## 5 Strategic litigation in $d$ dimensions

This section considers the  $d$ -Dimensional Linear Separator setting. We first characterize which goal functions  $g$  are *achievable* in principle, assuming the strategic litigator is free to bring any cases of his choosing. We show that all achievable  $g$  are achievable using only 2 cases. We then discuss two more challenging settings: when  $g$  is not achievable, and when the strategic litigator is restricted to bringing cases from a finite pool  $P$ .

### 5.1 Which goal functions are achievable?

In an ideal case, the strategic litigator would like to select a set of case fact patterns such that function output by  $\mathcal{A}$  (the lower court's learning algorithm) is exactly  $g$ . When can the strategic litigator force  $\mathcal{A}$  to produce exactly  $g$ , assuming they can select any case fact pattern in the space? We answer this question in Theorem 8 below.

► **Definition 7** (Achievable function). *We call a function  $g$  achievable if there exists some set  $\{x_1, \dots, x_m\}$  of case fact patterns such that  $g \leftarrow \mathcal{A}(\{(x_1, f^*(x_1)), \dots, (x_m, f^*(x_m))\} \cup S_h)$ , where  $f^*$  is the labeling function of the high court,  $\mathcal{A}$  is the learning algorithm of the low court, and  $S_h$  is the set of historical data.*

► **Theorem 8.** *Let  $S_h$  be the set of historical cases labeled by  $f^*$ . Let  $g$  be some linear function and let  $\theta$  be the angle between the normal vectors of  $g$  and  $f^*$  (call these  $w_g$  and  $w_{f^*}$  respectively) in degrees. Assume the strategic litigator can bring any case fact pattern in the space. Then,  $g$  is achievable if and only if one of the following conditions hold:*

- $0 < \theta < 90$ , there is no historical data in the disagreement region of  $f^*$  and  $g$  (i.e.  $\{(x, y) \in S_h : f^*(x) \neq g(x)\} = \emptyset$ ), and there are no points directly on the boundary of  $g$ .
- $\theta = 0$  (i.e.  $f^*$  and  $g$  are parallel and positive in the same direction), and there is no historical data less than distance  $\delta$  from  $g$ , where  $\delta$  is the maximum of the distances from  $g$  to the positive region defined by  $f^*$  and to the negative region defined by  $f^*$ .

Moreover, if  $g$  is achievable, it is achievable using 2 cases.

■ **Algorithm 1** Select a subset of case fact patterns  $\{x_1, x_2, \dots\} \subseteq P$  such that  $f \leftarrow \mathcal{A}(S_h \cup \{(x_1, f^*(x_1)), \dots\})$  minimizes  $\mathbb{E}_{x \sim \mathcal{D}}[\mathbb{1}[g(x) \neq f(x)]]$ .

---

**Inputs:**

- $f^*$ : the high court’s labeling function.
- $g$ : the strategic litigator’s goal function.
- $\mathcal{D}$ : the strategic litigator’s distribution of case fact patterns of interest.
- $S_h$ : historical labeled data.
- $P$ : collection of available case fact pattern points.

**Output:** A subset of case fact patterns  $X \subseteq P$  which minimizes the strategic litigator’s expected error with respect to  $g$  and  $\mathcal{D}$ .

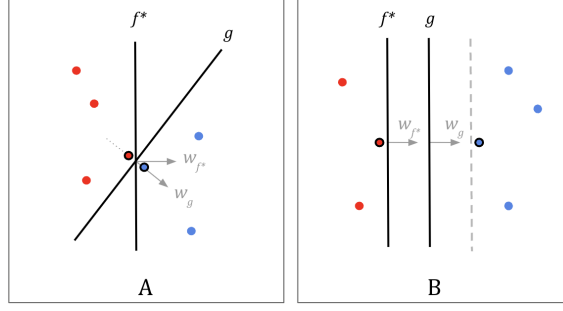
**Utilities:**

- Let **nearest\_neighbor**(*interval*,  $S$ ) compute the standard nearest neighbor classifier over the interval *interval* given points in  $S$ .
  - Let **discrepancy**( $f, g, \text{interval}, \mathcal{D}$ ) compute the sum of the lengths of intervals within *interval* throughout which  $f(x) \neq g(x)$ , where units of distance are probability mass under  $\mathcal{D}$ .
- 1: Rescale coordinates so all inputs (points, functions,  $\mathcal{D}$ ) are with respect to the interval  $[0, 1]$
  - 2: Sort  $P$  from leftmost to rightmost point
  - 3: Construct a DP table  $T$  which is a list of length  $p$ , where  $p$  is the number of points in  $P$ .  $T[j]$  will store the strategic litigator’s minimum total error if  $P[j]$  is the rightmost point selected.
  - 4:  $f_0 = \text{nearest\_neighbor}([0, 1], S_h \cup \{(P[0], f^*(P[0]))\})$  *output of Nearest Neighbor when choosing only  $P[0]$*
  - 5:  $T[0] = \text{discrepancy}(f_0, g, [0, 1], \mathcal{D})$  *error when choosing only  $P[0]$*
  - 6: **for**  $j = 1$  to  $p - 1$  **do**
  - 7:    $f_j = \text{nearest\_neighbor}([0, 1], S_h \cup \{(P[j], f^*(P[j]))\})$  *output of Nearest Neighbor when choosing only  $P[j]$*
  - 8:    $\text{err\_only\_j} = \text{discrepancy}(f_j, g, [0, 1], \mathcal{D})$  *error when choosing only  $P[j]$*
  - 9:    $\text{err\_j\_after\_i} = \min_{i \in [j-1]} \text{error}(i, j)$  *calculate the lowest possible error when choosing point  $j$  if  $i$  is the closest chosen point to its left*
  - 10:    $T[j] = \min\{\text{err\_only\_j}, \text{err\_j\_after\_i}\}$
  - 11: **end for**
  - 12:  $\text{min\_err\_with\_points} = \min_j T[j]$
  - 13:  $f_\emptyset = \text{nearest\_neighbor}([0, 1], S_h)$  *output of Nearest Neighbor when choosing no points*
  - 14:  $\text{err\_no\_points} = \text{discrepancy}(f_\emptyset, g, [0, 1], \mathcal{D})$  *error when choosing no points*
  - 15: **if**  $\text{err\_no\_points} \leq \text{min\_err\_with\_points}$  **then**
  - 16:   **return**  $\emptyset$
  - 17: **else**
  - 18:   **return** the set of points corresponding to  $\min_j T[j]$ . We can find these by retracing the entries of  $T$  we used in the minimum cost solution, and adding each of the corresponding points to our output set.
  - 19: **end if**

*Local helper function which computes the minimum error when the  $i$ th and  $j$ th points are the two rightmost selected points (the  $j$ th point is to the right of the  $i$ th point)*

**procedure**  $\text{error}(i, j)$ :

- 1:  $f_i = \text{nearest\_neighbor}([P[i], 1], S_h \cup \{(P[i], f^*(P[i]))\})$  *output of Nearest Neighbor from  $P[i]$  onward when  $P[i]$  is the rightmost selected point*
  - 2:  $\text{err\_after\_i\_no\_j} = \text{discrepancy}(f_i, g, [P[i], 1], \mathcal{D})$  *error to the right of  $P[i]$  without  $j$*
  - 3:  $f_{ij} = \text{nearest\_neighbor}([P[i], 1], S_h \cup \{(P[i], f^*(P[i])), (P[j], f^*(P[j]))\})$  *output of Nearest Neighbor from  $P[i]$  onward when we choose  $P[j]$  next after  $P[i]$*
  - 4:  $\text{err\_after\_i\_with\_j} = \text{discrepancy}(f_{ij}, g, [P[i], 1], \mathcal{D})$  *error to the right of  $P[i]$  when we choose  $P[j]$  next*
  - 5: **return**  $T[i] - \text{err\_after\_i\_no\_j} + \text{err\_after\_i\_with\_j}$  *minimum overall error when choosing  $P[j]$  immediately after  $P[i]$*
-



■ **Figure 6** Frame A illustrates the first case from Theorem 8. The two points with bolded border lying on  $w_g$  and  $-w_g$  will become the support vectors, producing  $g$  (the other points illustrate historical data). Frame B illustrates the second case; again, the two points with bolded border will become the support vectors, producing  $g$ . The dashed line marks distance  $\delta$  away from  $g$  on the right.

**Proof.** First, we show that SVM cannot produce any  $g$  whose normal vector  $w_g$  has angle  $\geq 90$  degrees to the normal vector of  $f^*$ . SVM produces some linear classifier whose normal vector is a linear combination of the data points, where the weights of the data points have the same sign as the their label. This implies that the normal vector of any function SVM can produce using data labeled by  $f^*$  has a positive inner product with  $w_{f^*}$ . Two vectors with a positive inner product have angle  $< 90$  degrees.

The rest of the negative results are easy to see: If any point  $p$  lies directly on  $g$ , we cannot teach  $g$  itself to SVM (although we can get arbitrarily close). Suppose WLOG  $p$  is positive; since SVM produces a maximum-margin classifier, the classifier it produces can only ever be as close to  $p$  as to the closest negative point the strategic litigator can bring; this distance cannot be 0 when  $g$  is a linear separator. Furthermore, SVM cannot produce any  $g$  which is inconsistent with any historical data, because SVM is guaranteed to find a consistent function in  $\mathcal{H}$  if one exists, and  $f^*$  is such a function. Finally, suppose  $f^*$  and  $g$  are parallel and positive in the same direction, and there is historical data within distance  $\delta$  of  $g$ , where  $\delta$  is the maximum of the distances from  $g$  to the positive region defined by  $f^*$  and to the negative region defined by  $f^*$ . If any point is less than  $\delta$  away from  $g$ , WLOG let it be positive. The closest negative point must be at least  $\delta$  away from  $g$  by definition of  $\delta$ , so SVM cannot produce  $g$  as it will produce some classifier with equal minimum margin on both sides.

It remains to show that we can produce any  $g$  which does not violate the above constraints, and do so by bringing just 2 points. In the special case where  $f^* = g$  and there is no historical data, the results of [5] apply: they showed that the number of points required to teach a particular separator to SVM is 2. In general, if  $f^*$  and  $g$  are parallel in the same direction, we can choose one positive and one negative point equidistant from  $g$  along the normal vector of  $g$ , at least as close to  $g$  as any historical data (any pair of equidistant points at least  $\delta$  away on either side of  $g$  will suffice; we can find such points since we assume that there are no historical points less than  $\delta$  away from  $g$ ). If  $f^*$  and  $g$  intersect, without loss of generality let the intersection point be the origin. Then it suffices to choose one positive and one negative point along the normal vector of  $g$ , arbitrarily closer to  $g$  than the distance of any historical point to  $g$ . In either case, since the two points we choose have different labels, and are equidistant from  $g$  on either side of  $g$  along a line perpendicular to  $g$ ,  $g$  is the maximum margin separator with respect to only this pair of points. The two points we choose must be closer to each other than any other pair of points from different classes are to each other, so they will become the only support vectors (the only points which determine the output of SVM). Therefore, SVM will produce  $g$  with the addition of these two points. ◀

## 5.2 Approximating non-achievable goals functions

In case the strategic litigator's goal function  $g$  is not achievable (see Definition 7), the strategic litigator should choose an achievable function  $g'$  which minimizes the strategic litigator's classification error (the error according to  $g$ , with respect to whatever distribution  $\mathcal{D}$  of case fact patters the strategic litigator cares about). Once the strategic litigator has selected this achievable proxy goal  $g'$ , our results from Theorem 8 apply, and the strategic litigator can achieve  $g'$  by bringing two case fact patterns as described in the proof of Theorem 8 above.

The remaining task in this case is therefore to find an achievable  $g'$  which minimizes  $\mathbb{E}_{x \sim \mathcal{D}}[\mathbb{1}[g(x) \neq g'(x)]]$ . The strategic litigator can find  $g'$  using the following simple brute force method. Draw a sample  $S$  of points from  $\mathcal{D}$ , and enumerate all achievable (according to the constraints specified in Theorem 8) SVM classifiers which correspond to distinct labelings of points in  $S$ . Choose as  $g'$  a classifier in this set which minimizes the number of misclassified points in  $S$ . Note that by the Sauer-Shelah Lemma [7], the number of effective classifiers is polynomial in  $|S|$ . When the size of the sample is large enough, this approximates the best achievable classifier with respect to error on  $\mathcal{D}$ . Using a standard uniform convergence argument and the fact that the VC dimension of a  $d$ -dimensional linear classifier is  $d + 1$ ,  $|S| = \tilde{O}(\frac{d}{\epsilon^2})$  is enough to guarantee that the error of any classifier  $g'$  on  $S$  is  $\epsilon$ -close to the true error over  $\mathcal{D}$  for some  $\epsilon$ .

## 5.3 Optimally selecting cases from a finite pool

Suppose the strategic litigator can only select case fact patterns from a limited pool  $P$ . The output function of SVM is determined entirely by its support vectors (points on the margin). In  $d$  dimensions, there can be at most  $d + 1$  linearly independent support vectors, so any function achievable using *any* subset of  $P$  can also be achieved using a subset of size at most  $d + 1$ . Trivially, then, the strategic litigator can enumerate the set  $F$  of all achievable functions by enumerating the functions induced by each of the  $O(n^{d+1})$  subsets of size  $\leq d + 1$  in  $P$  (for a given subset  $\{x_1, x_2, \dots\}$ , the corresponding separator is  $\mathcal{A}(S_h \cup \{(x_1, f^*(x_1)), (x_2, f^*(x_2)), \dots\})$ ). It follows that if  $g \in F$ , it is achievable and the strategic litigator has already found the corresponding subset of  $P$  needed to achieve it. Otherwise, if  $g \notin F$ ,  $g$  is not achievable. In that case, just as described above in Section 5.2, the strategic litigator can choose a  $g' \in F$  which approximately minimizes their classification error on their distribution of interest ( $\mathcal{D}$ ) by choosing  $g'$  which minimizes their error on a sufficiently large sample from  $\mathcal{D}$  (the same sample complexity bound holds here). In this case, the strategic litigator has already found the corresponding subset of  $P$  to teach  $g'$ .

## 6 Extension: Overturning Precedent

We also study the case where there is some historical precedent which does not agree with the high court's current reasoning  $f^*$ , and model a process for overturning precedent. A change in the high court's labeling function could correspond to a change in justices, or simply to a change in the existing justices' views.

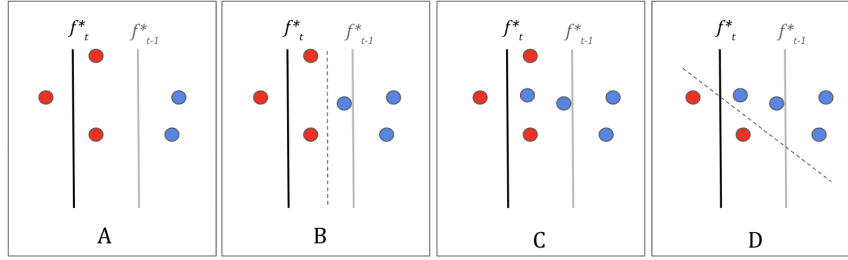
Let  $S_h$  be the set of all historical cases and their labels. These cases could have been labeled by  $f^*$  or by earlier high court function(s).

We assume the current high court has some limited power to throw out precedent which does not agree with their current reasoning represented by  $f^*$  (e.g, points whose label does not match the label  $f^*$  would assign). If the high court throws out some precedent (a set of labeled data points), those data points are removed from the set of precedent  $S_h$ .

We assume the high court only throws out precedent when deciding a new case (e.g. classifying a new fact pattern  $x$ ), and then only if a certain condition holds: the high court only throws out any precedent if when the newly labeled point  $(x, f^*(x))$  is added to the data set  $S_h$ , there is no longer any function in the hypothesis class  $\mathcal{H}$  which is consistent with all the data in  $S_h$ .

When the high court does throw out precedent, we assume they only remove points whose label disagrees with  $f^*$ . Moreover, we assume they do not gratuitously throw out precedent: we assume they throw out some minimal set such that there exists a function in  $\mathcal{H}$  which is consistent with the remaining data.

► **Example.** Figure 7 illustrates our model of overturning precedent.



■ **Figure 7** Frame A shows historical data labeled by a previous high court function  $f_{t-1}^*$ , when a new high court function  $f_t^*$  arrives. Frame B shows a new point arriving which is classified positively by  $f_t^*$ . No precedent is overturned as there still exists a linear separator consistent with the data (e.g., the dashed line). Frame C shows another point arriving which is classified positively by  $f_t^*$ . Now there is no longer a linear separator consistent with the data, so the high court must remove some subset of the points whose labels it disagrees with (the two negative points to the right of  $f_t^*$ ). Frame D illustrates if the high court chooses (arbitrarily) to remove the top negative point. It does not remove both because it only removes a minimal set such that there is a linear separator consistent with the remaining data (e.g., the dashed line).

We now return to considering whether the strategic litigator can achieve their goals in this extended setting. Notice that the strategic litigator may now need to strategize about how to force the high court to overturn precedent which disagrees with  $g$ . We assume the strategic litigator knows the high court's model for overturning precedent as described above, but does not know which minimal set of precedent the high court will choose to remove if there are multiple possibilities.

## 6.1 Results in 1-Dimensional Nearest Neighbor Setting

This extension is trivial in our 1-Dimensional Nearest Neighbor setting. Under a Nearest Neighbor classifier and our model for overturning precedent, the only way to get the high court to remove a historical point is to bring that exact point again and have it labeled the opposite way. If any existing points whose label disagrees with  $f^*$  are in the pool of case fact patterns the strategic litigator can choose from, Algorithm 1 is already set up to evaluate whether the strategic litigator should choose them – only a minor modification is needed to make sure we compute our Nearest Neighbor error with respect to the new label when a duplicate point is chosen.

## 6.2 Results in $d$ -Dimensional Linear Separator Setting

Now we consider this extension in our  $d$ -dimensional linear separator setting, assuming the strategic litigator can choose any case fact pattern in the space. If  $g$  is achievable according to the constraints specified in Theorem 8 when taking into account only historical data labeled by  $f^*$ ,  $g$  is still achievable with the addition of historical data whose labels disagree with  $f^*$ , so long as these new points also conform to the achievability constraints described in Theorem 8 (e.g., there is no data within distance  $\delta$  of  $g$  in the case where  $g$  is parallel to  $f^*$ , and no data in the disagreement region of  $f^*$  and  $g$  in the case where  $f^*$  and  $g$  intersect).<sup>2</sup>

Trivially, the strategic litigator can achieve  $g$  in this case by bringing up all the historical points whose label disagrees with both  $g$  and  $f^*$ . They will all be relabeled by  $f^*$  and therefore their labels will also agree with  $g$ , returning the strategic litigator to precisely the achievable setting described in Theorem 8. The strategic litigator can then bring 2 points as described in the proof of Theorem 8 to achieve  $g$ . However, this could potentially require bringing many points. The strategic litigator can also achieve  $g$  using at most  $2d + 1$  points, as stated in Theorem 9.

► **Theorem 9.** *In our  $d$ -dimensional linear separator setting, any linear classifier  $g$  is achievable if it satisfies the conditions in Theorem 8. Moreover, if  $g$  is achievable, it is achievable by bringing at most  $2d + 1$  points.*

**Proof.** The proof idea is to bring a set  $N$  of points satisfying that

- any linear classifier consistent with  $N$  will label the existing points in the same way as  $f^*$ ;
- and similar to Theorem 8,  $N$  contains two points to teach  $g$ .

Let  $S_{dis}$  denote the set of existing points that disagree with  $f^*$  and  $g$ . For each linear classifier  $w^\top x + b \geq 0$ , we can represent it by  $(w, b)$ . Then we can denote the set of all linear separators by  $\{(w, b) \mid \|w\|_2 = 1\}$  (here we restrict  $\|w\|_2 = 1$  since the scaling doesn't matter). We first prove the statement for the first condition, where **the angle between  $f^*$  and  $g$  satisfies  $0 < \theta < 90$** .

**Proof for the case of  $d = 2$ .** Let's start with 2-dimensional case. We pick a coordinate system such that  $f^*$  is represented as  $e_1^\top x \geq 0$  and  $g$  is represented as  $w_g^\top x \geq 0$  with  $w_{g,1} > 0, w_{g,2} \geq 0$ , where  $e_1, e_2, \dots$  denote the standard basis of this coordinate system. Let  $w_g^\perp = \frac{e_1 - (e_1^\top w_g)w_g}{\|e_1 - (e_1^\top w_g)w_g\|_2}$  denote the normalized vector perpendicular to  $w_g$ . When there are no existing disagreeing points lie on the boundary of  $f^*$ , we bring 4 points  $N = \{(\epsilon w_g, +1), (-\epsilon w_g, -1), (\epsilon w_g + \alpha w_g^\perp, +1), (\alpha e_2, +1)\}$  for some large  $\alpha$  and small  $\epsilon$ . For any negative disagreeing point  $(x, -1)$  (satisfying  $f^*(x) = g(x) = +1$  and not on the boundary of  $g$ ), the feature  $x$  can be decomposed as  $x = \alpha_1 w_g^\perp + \alpha_2 e_2$  for some  $\alpha_1, \alpha_2 > 0$  (since  $e_1^\top x = \alpha_1 \frac{1 - (e_1^\top w_g)w_{g,1}}{\|e_1 - (e_1^\top w_g)w_g\|_2} > 0$  and  $w_g^\top x = \alpha_2 w_{g,2}$ ). Hence, when  $\epsilon \rightarrow 0$  and  $\alpha \rightarrow \infty$ ,  $x$  must lie inside the convex hull of  $\epsilon w_g, \epsilon w_g + \alpha w_g^\perp, \alpha e_2$  with some margin (i.e.,  $x$  is not on the boundary of this convex hull). Any linear separator consistent with  $N$  will label this convex hull as positive and thus  $x$  as positive by some positive margin (which depends on  $x$  only) and thus  $(x, -1)$  will be deleted. Similarly, for any positive disagreeing point  $(x, +1)$  (satisfying  $f^*(x) = g(x) = -1$ ), when  $\epsilon \rightarrow 0$  and  $\alpha \rightarrow \infty$ ,  $-x$  must lie in the convex hull of  $\epsilon w_g, \epsilon w_g + \alpha w_g^\perp, \alpha e_2$ . Hence any consistent linear separator will label  $-x$  as positive by some positive margin. Thus  $2\epsilon w_g - x$  will also be labeled as positive when  $\epsilon$  is small enough.

<sup>2</sup> In the case where  $f^*$  and  $g$  intersect, this restriction is actually not necessary: any data in the disagreement region of  $f^*$  and  $g$  whose label disagrees with  $f^*$  must be labeled according to  $g$ . Additional points whose labels agree with  $g$  have no affect on our earlier results in this case. However, we will focus on the more restricted case for simplicity in directly applying our results from Theorem 8.

If there are negative disagreeing points lying on the boundary of  $f^*$ , pick the one closest to the origin and denote its distance by  $c$ . Then we add  $(\frac{c}{2}e_2, +1)$  to  $N$ . Then all existing negative points in  $S_{dis}$  points on the boundary will lie in the convex hull of  $\frac{c}{2}e_2$  and  $\alpha e_2$ . Thus, these points will be deleted. Hence, in the 2-d case, we can delete all disagreeing points by bringing 5 points. The two points  $(\epsilon w_g, +1)$  and  $(-\epsilon w_g, -1)$  will teach  $g$  as we showed in the proof of Theorem 8.

**Proof for the high dimensional case.** Now we reduce the high dimensional case to the 2-d case. Again we pick a coordinate system such that  $f^*$  is  $e_1^\top x \geq 0$  and  $g$  is  $w^\top x \geq 0$  with  $w_1 > 0, w_2 \geq 0, w_i = 0, \forall i \geq 3$ . Similar to the 2-d case, we add  $(\epsilon w_g, +1), (-\epsilon w_g, -1)$  to  $N$ . Then any linear classifier consistent with these two points must have  $b \in [-\epsilon, \epsilon]$ . Otherwise it will label these two points in the same way and be inconsistent with  $N$ . Then we add points  $(\epsilon w_g + \infty \cdot e_3, +1)$  and  $(\epsilon w_g - \infty \cdot e_3, +1)$  to  $N$ . Any linear separator  $w$  with  $w_3 \neq 0$  will label these two points in two different ways. Hence, any linear separator consistent with  $N$  will satisfy  $w_3 = 0$ . Similarly, we can also add such pair of points for other dimensions. Hence, by adding  $2(d-2)$  points, we reduce the high dimensional case to the 2-dimensional case.

**Under the second condition where  $\theta = 0$  and there are no historical data with in distance  $\delta$  of  $g$ .** Pick the coordinate system such that  $g$  is  $e_1^\top x \geq 0$  and  $f^*$  is  $e_1^\top x \geq -\delta$ . Let  $e_1, e_2, \dots$  denote the standard basis of this coordinate system. Then we can add  $N = \{(-(\delta+\epsilon)e_1, -1), ((\delta+\epsilon)e_1, +1), ((\delta+\epsilon)e_1 + \infty \cdot e_2, +1), ((\delta+\epsilon)e_1 - \infty \cdot e_2, +1), \dots, ((\delta+\epsilon)e_1 + \infty \cdot e_d, +1), ((\delta+\epsilon)e_1 - \infty \cdot e_d, +1)\}$ . By adding  $N$ , we can guarantee that any consistent linear separator will label the entire hyperplane  $e_1^\top x = \delta + \epsilon$  as positive. For any negative disagreeing point  $(x, -1)$  with  $x_1 > \delta$ , when  $\epsilon \rightarrow 0$ , the segment with ending points  $-(\delta+\epsilon)e_1$  and  $x$  will intersect with the hyperplane  $e_1^\top x = \delta + \epsilon$ . Then any consistent linear separator will not label  $x$  as negative; otherwise both  $-(\delta+\epsilon)e_1$  and  $x$  are labeled as negative and thus, the intersecting point is also labeled as negative, which is a contradiction. ◀

## 7 Discussion

This work is a preliminary exploration of strategic impact litigation from a machine learning theory perspective. We hope it can serve as a jumping off point for further study in this direction. Our model of the legal system is highly stylized, and there are numerous ways to enrich it. We will mention a few potentially interesting variations here.

One direction could be to model uncertainty about decisions in the legal process. Throughout this paper, we assumed the strategic litigator knows the function  $f^*$  which the high court uses to decide cases. We could alternatively consider a setting where, while the high court still makes decisions according to some  $f^*$ , the strategic litigator only knows a distribution over possible functions the high court could use. To think about uncertainty another way, we could also model the courts as making noisy decisions; this could capture the fact that a variety of unpredictable factors outside of the case fact pattern itself – from the random selection of potential jury members, to whether the judge is hungry while making the decision – can impact the outcome of a case.

We could also explore a more complex model of the courts; for example, the high court might provide some explanation of their reasoning about decisions in addition to the decision itself, which could then be used by lower courts to inform the classifier they learn.

Another interesting direction might be to model additional litigators. In this work, while there might be historical cases the strategic litigator did not bring, the strategic litigator benefits from being the only currently active litigator: the strategic litigator can bring as many strategically selected cases as they like without any cases being brought by others

throughout this process. New cases being brought by other agents might render the some of the cases initially selected by the strategic litigator less helpful or potentially even harmful. We could consider what outcomes might be achieved in a setting with multiple strategic litigators with competing interests, or perhaps in a less competitive setting where cases drawn at random from some distribution arrive periodically.

## References

- 1 Avrim Blum, Steve Hanneke, Jian Qian, and Han Shao. Robust learning under clean-label attack. In *Conference on Learning Theory*, pages 591–634. PMLR, 2021. URL: <http://proceedings.mlr.press/v134/blum21a.html>.
- 2 L Karl Branting, Craig Pfeifer, Bradford Brown, Lisa Ferro, John Aberdeen, Brandy Weiss, Mark Pfaff, and Bill Liao. Scalable and explainable legal prediction. *Artificial Intelligence and Law*, 29:213–238, 2021. doi:10.1007/S10506-020-09273-1.
- 3 S.A. Goldman and M.J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995. doi:10.1006/jcss.1995.1003.
- 4 Jason D. Hartline, Daniel W. Linna, Liren Shan, and Alex Tang. Algorithmic learning foundations for common law. In *Proceedings of the 2022 Symposium on Computer Science and Law*, CSLAW '22, pages 109–117, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3511265.3550438.
- 5 Ji Liu, Xiaojin Zhu, and H. Gorune Ohannessian. The teaching dimension of linear learners. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 117–126, New York, New York, USA, 20–22 June 2016. PMLR. URL: <https://proceedings.mlr.press/v48/liua16.html>.
- 6 Ejan Mackaay and Pierre Robillard. *Predicting judicial decisions: The nearest neighbour rule and visual representation of case patterns*, pages 302–331. De Gruyter, Berlin, Boston, 1974. doi:10.1515/9783112320594-012.
- 7 N Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972. doi:10.1016/0097-3165(72)90019-2.
- 8 Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suci, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 6106–6116, Red Hook, NY, USA, 2018. Curran Associates Inc. URL: <https://proceedings.neurips.cc/paper/2018/hash/22722a343513ed45f14905eb07621686-Abstract.html>.

## A Proof of Theorem 5

Here we'll give a proof of the correctness and runtime of Algorithm 1 (Theorem 5).

**Proof.** First, we'll prove the correctness of Algorithm 1. We will assume the utility functions are correct, so `nearest_neighbor(interval, S)` correctly computes a standard nearest neighbor classifier over the interval *interval* given points in *S*, and `discrepancy(f, g, interval, D)` correctly computes the sum of the lengths of intervals within *interval* throughout which  $f(x) \neq g(x)$ , where units of distance are probability mass under  $\mathcal{D}$ . Then, `discrepancy(f, g, interval, D)` also computes the strategic litigator's error for producing *f* over the interval *interval*, since this is the total probability mass on points *f* misclassifies over the interval *interval*. These will be used throughout the rest of the proof.

Recall that we begin by re-scaling coordinates so all inputs ( $f$ ,  $g$ , all points, and  $\mathcal{D}$ ) are with respect to the interval  $[0, 1]$ , so throughout the rest of this proof,  $[0, 1]$  is the entire interval and  $[p, 1]$  is the entire interval including and to the right of some point  $p$ .

The minimum-error set of points is either empty or contains some points, and we return the set corresponding to the minimum error between these two options (lines 15-18). To show the algorithm is correct, then, it remains to show that the error in each case is correctly computed and corresponds to a valid set of points.

When the strategic litigator does not choose any points, the strategic litigator's error is the error corresponding to the output of Nearest Neighbor over the entire interval  $[0, 1]$  given only the set of historical points  $S_h$ , so we correctly compute the minimum error in this case. When this case gives the minimum error overall, we correctly return  $\emptyset$  (provided we also correctly compute the minimum error for the case when the strategic litigator does choose some points, which we will show next).

If  $T[j]$  correctly stores the minimum error when  $P[j]$  is the rightmost point selected, we correctly compute the minimum error in the case where the strategic litigator selects at least one point as  $\min_j T[j]$ , since some selected point must be the rightmost selected point. We'll prove that  $T[j]$  stores the minimum error when  $P[j]$  is the rightmost selected point by induction. Recall that we start by sorting the collection of points  $P$  from left to right, so when we refer to the  $i$ th point, it is the  $i$ th point from the left.

Consider the base case,  $T[0]$ . When choosing  $P[0]$  as the rightmost point, it must be the only selected point. This corresponds to the error of the output of Nearest Neighbor over the entire interval  $[0, 1]$  given  $P[0]$  labeled by  $f^*$  and the rest of the historical points, which is exactly what we compute. Clearly this case corresponds to a valid set of points ( $\{P[0]\}$ ).

Next, assuming  $T[k]$  correctly stores the minimum error when the  $k$ th point is the rightmost selected point for some  $k \geq 0$ , we'll show  $T[k+1]$  is also computed correctly. When the  $k+1$ th point is the rightmost selected point, either it is the only selected point or there is at least one other selected point to its left, so the minimum error overall is the minimum of these two cases.

The first case corresponds to the error of the output of Nearest Neighbor over the entire interval  $[0, 1]$  given  $P[k+1]$  labeled by  $f^*$  and the set of historical points, which is exactly what we compute. Clearly this case corresponds to a valid set of points ( $\{P[k+1]\}$ ).

Now consider the second case where we select point  $P[k+1]$  and at least one other point  $P[i]$  which is the closest selected point to its left. The selection of the  $P[k+1]$  does not alter the Nearest Neighbor classifier over the interval  $[0, P[i]]$ , and only  $P[i]$  and points to its right affect the classifier over the interval  $[P[i], 1]$ . Therefore, the minimum error achievable in this case is the total minimum error when  $P[i]$  is the rightmost point over the entire interval  $[0, 1]$  less the error to the right of  $P[i]$  when  $P[i]$  is the rightmost selected point, plus the error to the right of  $P[i]$  when  $P[k+1]$  is the rightmost selected point. Since by our inductive hypothesis  $T[i]$  correctly stores the total minimum error when  $P[i]$  is the rightmost point over the entire interval  $[0, 1]$ , we correctly compute the minimum error in this case (see lines 1-5 of the helper function **error**). Again since the selection of the  $P[k+1]$  does not alter the Nearest Neighbor classifier over the interval  $[0, P[i]]$ , and only  $P[i]$  and points to its right affect the classifier over the interval  $[P[i], 1]$ , simply adding  $P[k+1]$  to the set of points corresponding to the best error when  $P[i]$  is the rightmost point is a valid set of points corresponding to the minimum error we store in  $T[k+1]$ . This concludes the proof of correctness.

That the algorithm uses  $O(|P|^2)$  oracle calls is easy to see: the table has  $|P|$  entries, and to compute each entry  $T[i]$  for  $0 \leq i \leq |P|$ , we do a constant number of oracle calls to **nearest\_neighbor** and **discrepancy** involving every entry to the left of  $T[i]$ . Although it depends on implementation, we'd expect that each call to **nearest\_neighbor** would take up to  $O(|S_h|)$  time as used here, while the time complexity of each call to **discrepancy** would be at most linear in the total number of “alternations” in both functions being compared (i.e., the number of times each function switches from negative to positive or vice versa). Here, the number of alternations is linear in  $|S_h|$  plus the number of alternations in  $g$ , which is arbitrarily selected by the strategic litigator. ◀