Strategic Classification

Gaming-Robust Learning under Strategic Behavior

Benjamin de Castro Patel Priyansh

Paper reference

M. Hardt, N. Megiddo, C. Papadimitriou, and M. Wootters, *Strategic Classification*, in *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science (ITCS)*, 2016.

Main question: How should a classifier be designed when people can change features strategically after seeing it?

Motivating example

School admissions

1. Book ownership correlates with academic success
2. Train an admissions classifier using book ownership
3. Applicants buy books to game the classifier

Goodhart's Law

"If a measure becomes the public's goal, it is no longer a good measure."

Secrecy is a poor solution

- Contestants can infer the rule from outcomes
- Transparency is often preferable or required

Paper goal: formalize gaming in classification and design classifiers that are optimal against strategic response.

Full Information Game

Definition 1.1

Fix a population $\mathcal{X}$, distribution $\mathcal{D}$ over $\mathcal{X}$, cost function

$$c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+,$$

and target classifier

$$h : \mathcal{X} \to \{-1, 1\}.$$

1. Jury knows $c, \mathcal{D}, h$ and publishes a classifier $f : \mathcal{X} \to \{-1, 1\}$.
2. Contestant knows $c, \mathcal{D}, h, f$, is given $x \sim D$, and chooses a response map $\Delta : \mathcal{X} \to \mathcal{X}$.

Jury payoff:

$$\Pr_{x \sim \mathcal{D}} \big[h(x) = f(\Delta(x))\big].$$

Contestant payoff:

$$\mathbb{E}_{x \sim \mathcal{D}} \big[f(\Delta(x)) - c(x, \Delta(x))\big].$$

Full Information Game: Best Response and Equilibrium

Contestant best response

For each original point x,

$$\Delta(x) \in \arg\max_{y \in \mathcal{X}} \; f(y) - c(x, y).$$

Definition 1.2: Strategic maximum

$$\mathrm{OPT}_h(\mathcal{D}, c) = \max_{f:\mathcal{X}\to\{-1,1\}} \Pr_{x\sim\mathcal{D}} \big[h(x) = f(\Delta(x))\big].$$

Stackelberg competition:

- Jury moves first and commits to f
- Contestant best-responds
- Jury wants the best classifier under that response
- Best Jury play assuming Contestant best response is Stackelberg equilibrium

Remark 1.3

Assume $c(x, x) = 0$.

Characterization

If $f(x) = 1$, then staying put is already accepted, so

$$\Delta(x) = x.$$

If $f(x) = -1$, let

$$y = \arg \min_{y:\, f(y)=1} c(x, y).$$

Then

$$\Delta(x) = \begin{cases} y, & c(x, y) < 2, \\ x, & c(x, y) > 2. \end{cases}$$

Intuition:

- Acceptance changes payoff from -1 to $+1$
- That gain is exactly 2
- So Contestant moves only if the move costs strictly less than 2

Statistical Classification Game

Definition 1.4

Same game, but Jury no longer knows h or $\mathcal{D}$ exactly.

1. Jury knows only the cost function c and may request labeled samples $(x, h(x))$ with $x \sim \mathcal{D}$. She outputs f.
2. Contestant knows c and f, then chooses Δ.

Jury payoff:

$$\Pr_{x \sim \mathcal{D}} \big[h(x) = f(\Delta(x))\big].$$

Contestant payoff:

$$\mathbb{E}_{x \sim \mathcal{D}} \big[f(\Delta(x)) - c(x, \Delta(x))\big].$$

Information split

- Jury gets c and labeled examples $(x, h(x))$
- Contestant gets c and the published classifier f

Strategy-Robust Learning

Definition 1.5

An algorithm A is a **strategy-robust learning algorithm** for a class of cost functions C if, for all $\mathcal{D}$, all classifiers h, all $c \in C$, and all ε, δ: given labeled examples $(x, h(x))$ with $x \sim \mathcal{D}$, A outputs a classifier f such that, with probability at least $1 - \delta$,

$$\Pr_{x \sim \mathcal{D}}[h(x) = f(\Delta(x))] > \mathrm{OPT}_h(\mathcal{D}, c) - \varepsilon.$$

Meaning: learn a classifier whose strategic payoff is nearly as good as the best possible classifier.

Uniform Strategy-Robust Learning

Definition 1.6

Let $\mathcal{H}$ be a concept class. An algorithm A is a **uniform strategy-robust learning algorithm** for $(\mathcal{H}, C)$ if, with probability at least $1 - \delta$ over the sample draw, the following holds *simultaneously for all* $h \in \mathcal{H}$:

$$\Pr_{x \sim \mathcal{D}}[h(x) = f(\Delta(x))] > \mathrm{OPT}_h(\mathcal{D}, c) - \varepsilon.$$

Difference from Definition 1.5

- Definition 1.5: guarantee for one fixed unknown h
- Definition 1.6: one sample must work for all $h \in \mathcal{H}$ at once

Why Definition 1.5 $\not\Rightarrow$ Definition 1.6: A Small Example

Setup:

- Population $X = \{a, b, c, d\}$, uniform distribution.
- Concept class $\mathcal{H} = \{h_1, h_2, h_3, h_4\}$ where each h_i labels exactly one point $+1$.
- Algorithm draws $m = 2$ samples.

For any *fixed* h_i, 2 samples suffice (Def 1.5). But no 2-sample draw covers all four targets at once (Def 1.6).

The targets:

	a	b	c	d
h_1	$+1$	-1	-1	-1
h_2	-1	$+1$	-1	-1
h_3	-1	-1	$+1$	-1
h_4	-1	-1	-1	$+1$

Suppose we draw samples $\{a, b\}$:

- ✓ Can learn h_1: saw a labeled $+1$.
- ✓ Can learn h_2: saw b labeled $+1$.
- × Cannot distinguish h_3 from h_4: never saw c or d.

This example assumes trivial gaming $\Delta(x) = x$ for understanding the difference

Contributions of the paper

1. **Efficient algorithms for separable costs**
   - Strategy-robust learning for any target h (Theorem 1.8)
   - Uniform strategy-robust learning when $\mathcal{H}$ is statistically learnable (Theorem 1.7)
2. **Extension to minima of separable costs**
   - Theorem 1.9
3. **Hardness for general costs**
   - Even metrics need not admit an efficient strategy-robust learning algorithm
   - Theorem 1.10

Main Theorems (1.7–1.10)

	Thm 1.7	Thm 1.8	Thm 1.9	Thm 1.10
Cost class	Separable $\mathcal{S}$	Separable $\mathcal{S}$	$\min$ of k sep.	Gen. metrics
Guarantee	Uniform	Non-uniform	Uniform	Hardness
Time	$\mathrm{poly}(m, 1/\varepsilon, \log 1/\delta)$	Poly	$\mathrm{poly}(m, e^k, 1/\varepsilon, \log 1/\delta)$	NP-hard
Samples	$\mathrm{poly}(m, 1/\varepsilon, \log 1/\delta)$	Poly	$\mathrm{poly}(m, k, 1/\varepsilon, \log 1/\delta)$	—
Requirement	$\mathcal{H}$ stat. learnable	None on h	$\mathcal{H}$ stat. learnable	No approx. within $\lvert X\rvert^{-\eta}$, any $\eta > 0$

All positive results: payoff $\geq \mathrm{OPT}_h(\mathcal{D}, c) - \varepsilon$ w.p. $\geq 1 - \delta$.

Statistically learnable

A concept class is statistically learnable if enough samples suffice to learn it with small generalization error, regardless of computational efficiency.

Separable Cost Functions

Definition 2.1

A cost function is **separable** if it can be written as

$$c(x, y) = \max\{0,\ c_2(y) - c_1(x)\},$$

for functions $c_1, c_2 : \mathcal{X} \to \mathbb{R}$ satisfying

$$c_1(\mathcal{X}) \subseteq c_2(\mathcal{X}).$$

Interpretation

- Each state has its own score/cost level
- Transition cost is the increase from the old state to the new one
- Nonnegativity comes from the $\max\{0, \cdot\}$
- Not moving is always an option $c_1(\mathcal{X}) \subseteq c_2(\mathcal{X})$

Algorithm 1: Threshold Classifier Motivation

For separable costs, the classifier can be restricted to threshold functions of the form

$$c_2[t](x) = \begin{cases} 1, & c_2(x) \geq t, \\ -1, & c_2(x) < t. \end{cases}$$

Why this matters

Instead of searching over all classifiers $f : \mathcal{X} \to \{-1, 1\}$, Jury only searches over one real threshold t.

Consequence: the strategic optimization problem collapses to a one-dimensional threshold search.

Algorithm 1: Statement

Input: Labeled examples $(x_1, h(x_1)), \ldots, (x_m, h(x_m))$ with $x_i \sim \mathcal{D}$ i.i.d. Cost function $c(x, y) = \max\{0,\, c_2(y) - c_1(x)\}$.

---

1. For $i = 1, \ldots, m$, compute:

$$t_i := c_1(x_i), \qquad s_i := \begin{cases} \max\big(c_2(X) \cap [t_i,\, t_i + 2]\big) & \text{if } \neq \emptyset \\ \infty & \text{otherwise} \end{cases}$$

   Set $s_{m+1} = \infty$.

2. For each s_i, compute empirical error:

$$\widehat{\mathrm{err}}(s_i) = \frac{1}{m} \sum_{j=1}^{m} \mathbf{1}\{h(x_j) \neq c_1[s_i - 2](x_j)\}$$

3. Find i^* minimizing $\widehat{\mathrm{err}}(s_i)$.
4. **Return:** $f := c_2[s_{i^*}]$.

Algorithm 1: Annotations

Step 1: $t_i = c_1(x_i)$ measures how "advantaged" point x_i is. s_i is the most expensive reachable destination — the critical gaming threshold.

Step 2: $c_1[s-2]$ accepts all x with $c_1(x) \geq s-2$. This is exactly $\Gamma(c_2[s])$ — the set that ends up accepted after Contestant best-responds.

Steps 3–4: Pick the threshold with lowest empirical error. Output is a simple threshold on c_2.

Because any classifier f can be replaced by a threshold on c_2 with the same payoff to Jury. Only $m+1$ candidate thresholds matter, so runtime is $O(m^2)$.

Linear Cost Functions

Special case: $c(x, y) = \langle \alpha,\, y - x \rangle_+$ for $\alpha \in \mathbb{R}^n$.

- α assigns per-attribute cost for *increasing* that attribute.
- Moving **perpendicular** to α: free.
- Moving **opposite** to α: free (truncation at 0).
- Moving **along** α: costs $\|\alpha\|$ per unit.

Consequence: Any classifier f can be replaced by a halfspace f' with normal α, without changing payoff. Search reduces to one threshold:

$$f'(x) = \begin{cases} +1 & \langle \alpha, x \rangle \geq t \\ -1 & \text{otherwise} \end{cases}$$

This is a special case of separable with $c_1(x) = \langle \alpha, x \rangle$, $c_2(y) = \langle \alpha, y \rangle$.

Algorithm 1: Geometric Intuition with Linear Cost Functions

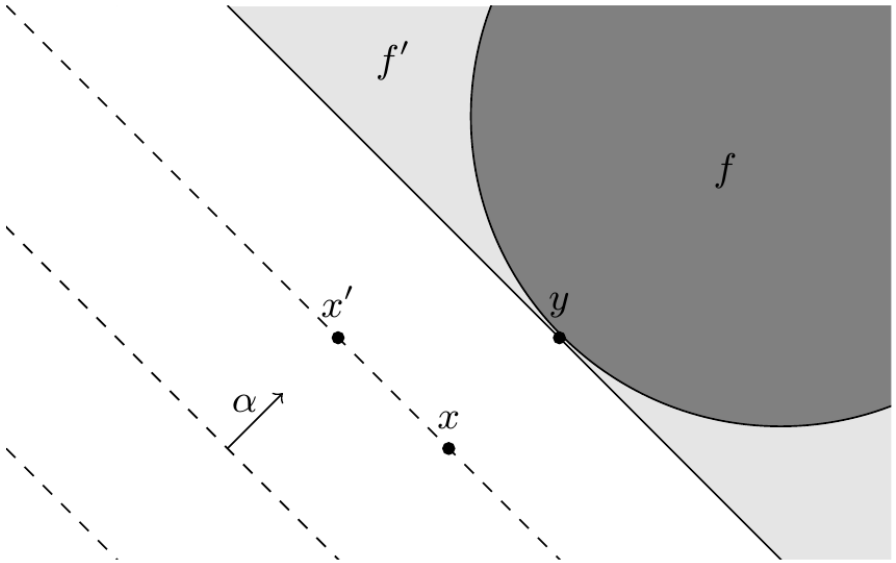


Figure 1: Suppose the optimal classifier for Jury is f (which accepts the dark gray region), and the cost function is $c(x, y) = \langle \alpha, y - x \rangle_+$. Because moving perpendicular to α is free for Contestant, then the payoff for Jury if she plays f' (shown above, which accepts the light gray region) is the same as her payoff if she plays f. Indeed, suppose that the agent x shown above would be willing to move to y to get accepted by f. Then x' would also be willing to move to y, because the cost is the same. Thus,

Key Lemma: Replacing f with a Threshold

Core Idea

Any classifier f can be replaced by a simpler threshold classifier f' without changing Jury's payoff.

For separable costs, define the threshold value:

$$s^* = \min\{c_2(y) : f(y) = 1\}$$

and the threshold classifier:

$$f'(y) = \begin{cases} 1, & c_2(y) \geq s^*, \\ -1, & c_2(y) < s^*. \end{cases}$$

Then:

$$\Gamma(f) = \Gamma(f') \quad \Longrightarrow \quad \Pr_{x \sim \mathcal{D}}[h(x) = f(\Delta(x))] = \Pr_{x \sim \mathcal{D}}[h(x) = f'(\Delta(x))]$$

Proof: $\Gamma(f) = \Gamma(f')$

Recall that for separable costs:

$$c(x, y) = \max\{0,\ c_2(y) - c_1(x)\}$$

Step 1. Write out $\Gamma(f)$:

$$\begin{aligned}\Gamma(f) &= \{x : c_1(x) > \min\{c_2(y) : f(y) = 1\} - 2\} \\ &= \{x : c_1(x) > \min\{c_2(y) : f'(y) = 1\} - 2\} \\ &= \Gamma(f')\end{aligned}$$

Step 2. Use symmetric difference $\triangle = \{x : (x \in A) \oplus (x \in B)\}$ to rewrite payoff:

$$\begin{aligned}\Pr[h(x) = f(\Delta(x))] &= \Pr\{x \in (\Gamma(f) \triangle \{y : h(y) = 1\})^c\} \\ &= \Pr\{x \in (\Gamma(f') \triangle \{y : h(y) = 1\})^c\} \\ &= \Pr[h(x) = f'(\Delta(x))]\end{aligned}$$

Consequence

It suffices to search over threshold classifiers $c_2[s]$, one per value $s \in c_2(\mathcal{X}) \cup \{\infty\}$.

Strategic Optimum as a Threshold Search

Optimal payoff (Theorem 2.3 proof):

$$\mathrm{OPT}_h(\mathcal{D}, c) = 1 - \inf_{s \in S} \mathrm{err}(s), \quad \mathrm{err}(s) := \Pr\big[h(x) \neq c_1[s-2](x)\big]$$

where $S := c_2(\mathcal{X}) \cup \{\infty\}$ and $c_1[t](x) = \mathbf{1}[c_1(x) > t]$.

Algorithm 1 candidate thresholds: for each sample point x_i, set

$$t_i = c_1(x_i), \qquad s_i = \max\big(c_2(\mathcal{X}) \cap [t_i,\, t_i + 2]\big), \quad s_{m+1} = \infty$$

Empirical error for each candidate:

$$\widehat{\mathrm{err}}(s_i) = \frac{1}{m} \sum_{j=1}^{m} \mathbf{1}\big[h(x_j) \neq c_1[s_i - 2](x_j)\big]$$

Output: $f = c_2[s_{i^*}]$ where $i^* = \arg\min_i \ \widehat{\mathrm{err}}(s_i)$.

Why only $m + 1$ candidates? (Claim 2.7)

$\widehat{\mathrm{err}}(s)$ only changes value when $c_1[s-2](x_j)$ changes for some j — which happens exactly at the s_i values. So the finite search is complete.

Claims 2.7 & 2.8: From Empirical to True Error

Claim 2.7 (Discrete search is fine)

$$\widehat{\mathrm{err}}(s^*) = \inf_{s \in S} \widehat{\mathrm{err}}(s)$$

The empirical minimizer over $m+1$ finite candidates achieves the global empirical minimum.

Claim 2.8 (Uniform convergence)

With probability at least $1-\delta$, for all $h \in \mathcal{H}$ and all $s \in S$:

$$\left|\widehat{\mathrm{err}}(s) - \mathrm{err}(s)\right| \;\leq\; 4R_m(\mathcal{H}) + 8\sqrt{\frac{\ln(m+1)}{m}} + \sqrt{\frac{2\ln(2/\delta)}{m}}$$

where $R_m(\mathcal{H})$ is the Rademacher complexity of $\mathcal{H}$.

Combining:

$$\mathrm{err}(s^*) \;\leq\; \mathrm{OPT}_h(\mathcal{D}, c) + \varepsilon$$

$\Longrightarrow$ Algorithm 1 is a **strategy-robust learning algorithm**. $\square$

Section 3: General Cost Functions (Overview)

Positive extension

- A general cost can be represented as the minimum of separable costs
- This gives Theorem 1.9

But in full generality:

- representation may require many separable pieces
- complexity can become large

Message

Separable structure is what makes efficient strategic learning possible.

Section 3: General Cost Functions

Can we go beyond separable costs?

Proposition 3.1 (Any cost = min of separable costs)

Let $\mathcal{X}$ be any finite set and $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ any mapping. Let $D > \max\{c(x, y) : x, y \in \mathcal{X}\}$. Then:

$$c(x, y) = \min_{w,z \in \mathcal{X}} \{c(w, z) + D \cdot \mathbf{1}[x \neq w] + D \cdot \mathbf{1}[y \neq z]\}$$

Each function $c_{w,z}(x, y) = c(w, z) + D \cdot \mathbf{1}[x \neq w] + D \cdot \mathbf{1}[y \neq z]$ is separable.
Consequence: any c can be written as the min of $|\mathcal{X}|^2$ separable functions.

Proposition 3.2 (Metric setting, ε-net)

Let c be a metric on $\mathcal{X}$ and S an ε-net of $\mathcal{X}$ (every $x \in \mathcal{X}$ has some $s \in S$ with $c(x, s) \leq \varepsilon$). Then for every $x, y \in \mathcal{X}$:

$$c(x, y) \;\leq\; \min_{w,z \in S} \{c(x, w) + c(w, z) + c(z, y)\} \;\leq\; c(x, y) + 4\varepsilon$$

The number of separable pieces needed depends only on $|S|^2$

Algorithm 2: Gaming-Robust Classification for Min-of-Separable Costs

Inputs: Labeled sample $(x_1, h(x_1)), \ldots, (x_m, h(x_m))$; description of k separable cost functions $b(x, y) = \max\{0,\, b_2(y) - b_1(x)\}$ for $b \in \mathcal{B}$.

Step 1. For each $i = 1, \ldots, m$ and $b \in \mathcal{B}$:

$$t_{i,b} = b_1(x_i), \qquad s_{i,b} = \begin{cases} \max\big(b_2(\mathcal{X}) \cap [t_{i,b},\ t_{i,b} + 2]\big) & \text{if non-empty} \\ \infty & \text{otherwise} \end{cases}$$

Set $s_{m+1,b} = \infty$ for all $b \in \mathcal{B}$.

Step 2. For each $\boldsymbol{s} \in \bigoplus_{b \in \mathcal{B}} \{s_{i,b} : i = 1, \ldots, m+1\}$, compute:

$$\widehat{\mathrm{err}}(\boldsymbol{s}) = \frac{1}{m} \sum_{j=1}^{m} \mathbf{1}\{h(x_j) \neq \min\{b_1[s_b - 2](x_j) : b \in \mathcal{B}\}\}$$

Step 3. Find $\boldsymbol{s}^*$ minimizing $\widehat{\mathrm{err}}(\boldsymbol{s})$. **Return:** $f(x) = \min\{b_2[s_b^*](x) : b \in \mathcal{B}\}$

Key idea

Run Algorithm 1's threshold logic independently per component b; the acceptance region is the *intersection* of all b-wise threshold regions.

Theorem 3.3 and Remark 3.4

Theorem 3.3 (informal)

Suppose $c(x, y) = \min_{b \in \mathcal{B}} b(x, y)$ where each $b \in \mathcal{B}$ is separable. Let $\mathcal{D}$ be a distribution and assume Algorithm 2 is run on m samples.
With probability at least $1 - \delta$, for all $h \in \mathcal{H}$ simultaneously:

$$\operatorname{err}(f) \ \leq \ \operatorname{OPT}_h(\mathcal{D}, c) + \varepsilon,$$

provided m satisfies

$$R_m(\mathcal{H}) + 2\sqrt{\frac{|\mathcal{B}| \ln(m+1)}{m}} + \sqrt{\frac{\ln(2/\delta)}{8m}} \ \leq \ \frac{\varepsilon}{8}$$

The running time of Algorithm 2 is $O(m^{|\mathcal{B}|})$.

Remark 3.4

When $|\mathcal{B}|$ is small, Theorem 3.3 gives a nice bound. However, if $|\mathcal{B}|$ is large, these guarantees are not so good. The term $\sqrt{|\mathcal{B}| \ln(m+1)/m}$ may be replaced by $R_m(\mathcal{H})$, where

Algorithm 2: Geometric Intuition with 2 Linear Cost Functions

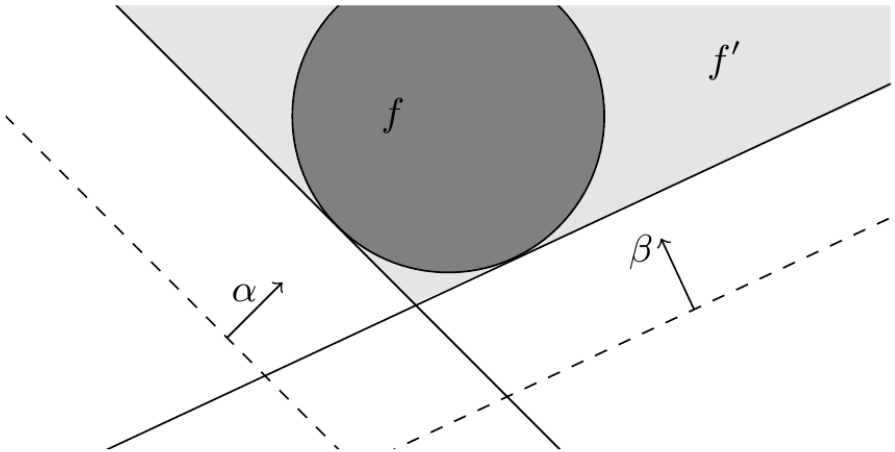


Figure 2: Suppose the the optimal classifier for Jury is f (which accepts the dark grey region), and the cost function is $c(x, y) = \min\left\{\langle\beta, y - x\rangle_+, \langle\alpha, y - x\rangle_+\right\}$. For the same reasoning as in Figure 1, the classifier f' has the same payoff to Jury as f does. Thus, Jury may restrict his/her search to classifiers f that are the intersections of two affine halfspaces.

so that m satisfies

NP-Completeness (Theorem 4.1)

For general cost functions, approximating the strategic optimum is **NP-hard** — even for **metrics**.

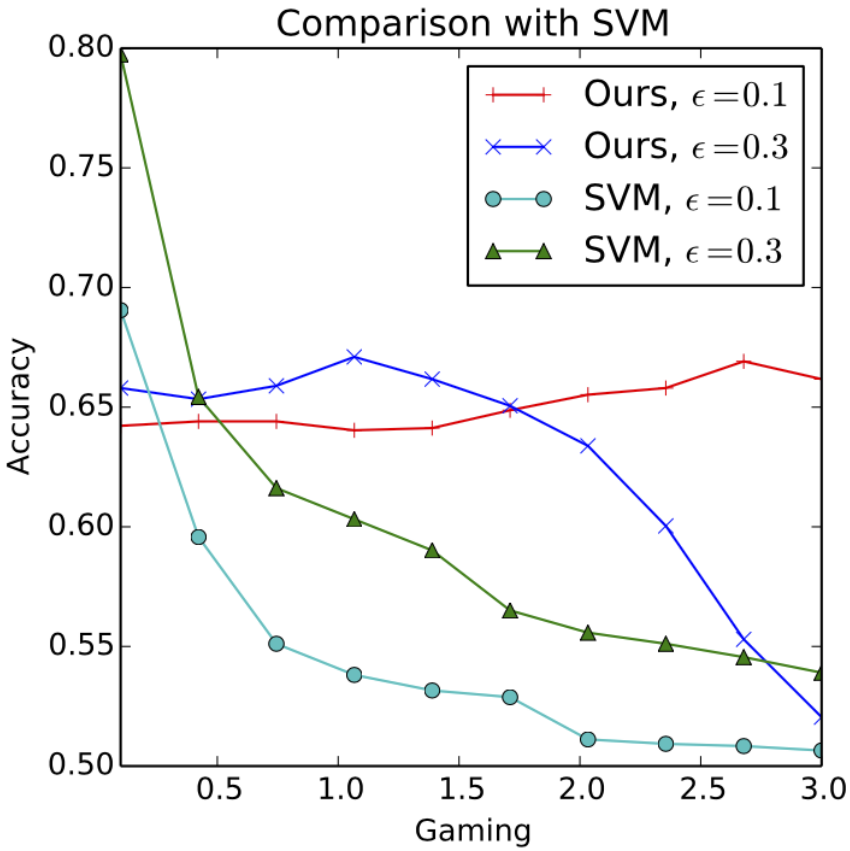
Cost class	**Complexity**
Separable	Poly
$\min$ of k separable	$\exp(k)$
Metrics	NP-hard

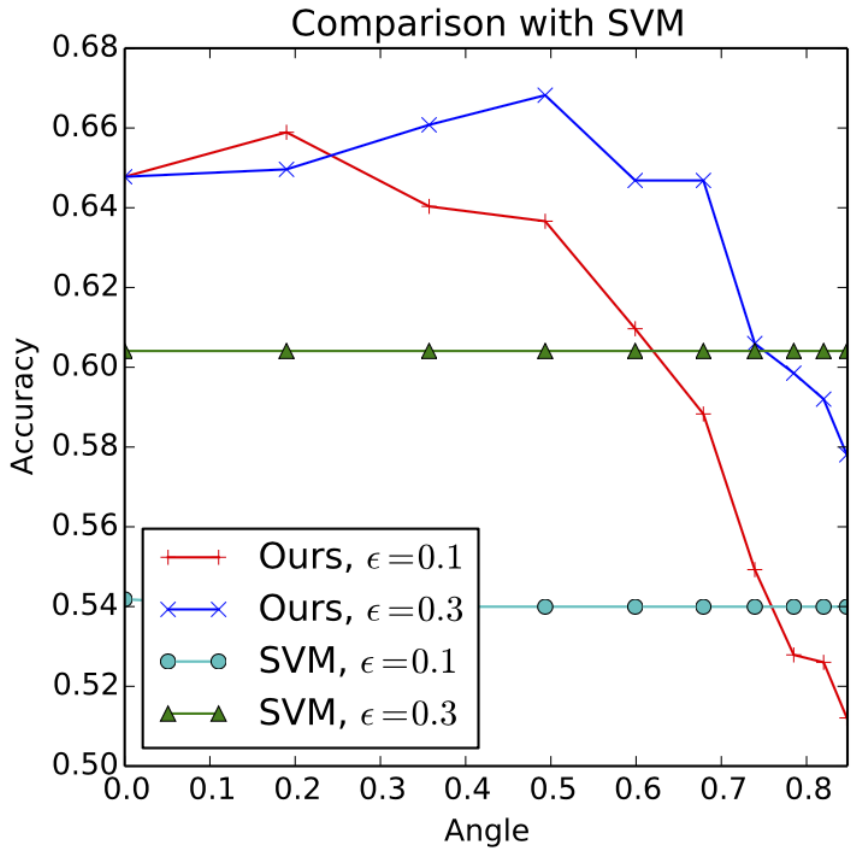
Metrics are nonneg., symmetric, satisfy triangle inequality. E.g.:

- Euclidean distance $\|x - y\|_2$
- Hamming Distance (Edit distance between strings)
- Shortest-path distance in a graph

Not separable: symmetry prevents writing as $\max\{0, c_2(y) - c_1(x)\}$.

Experimental Results: Comparison with SVM





- SVM accuracy drops sharply with even small amounts of gaming.
- Robustness holds even under **two sources of model error**: wrong cost function form ($\varepsilon > 0$) and wrong cost direction ($\alpha' \neq \alpha$).

Takeaways

1. Standard classification breaks when people can react strategically
2. The paper models this as a Stackelberg game
3. For separable costs, the optimal strategic search reduces to thresholding
4. Yielding efficient strategy-robust and uniform strategy-robust learning algorithms
5. For general costs, efficient near-optimal learning is impossible in general

Thank You!

Questions