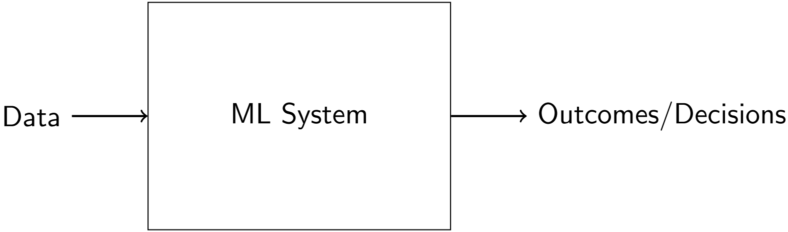
Learning in the Presence of Strategy

Dr. Ganesh Ghalme

Indian Institute of Technology Hyderabad

December 16, 2025

Classical ML



Classical ML

Historical Loans →

→ Loan decision

Classical ML

Medical History →  → Health Risk

Learning in the Presence of Strategy

- Agents with different goals, information and strategies interact with AI/ML algorithms

Learning in the Presence of Strategy

- Agents with different goals, information and strategies interact with AI/ML algorithms
- Agents are rational; take action that is best given their information
- Misalignment in objectives (also, sometimes information) leads to inefficiency

Strategic Learning

Classical vs Strategic Learning

- Let f be a binary classifier

Strategic Learning

Classical vs Strategic Learning

- Let f be a binary classifier
- Deployment time: $(x, y) \sim D$ is seen as is by f in classical setting.
- Strategic setting: $(x, y) \to (x', y')$.
    - The change (in feature value) is made by strategic agents (more on it later) in response to deployed classifier f

Strategic Learning

Classical vs Strategic Learning

- Let f be a binary classifier
- Deployment time: $(x, y) \sim D$ is seen as is by f in classical setting.
- Strategic setting: $(x, y) \rightarrow (x', y')$.
  - The change (in feature value) is made by strategic agents (more on it later) in response to deployed classifier f
  - If $y' = y$ then it is called gaming [a]

Strategic Learning

Classical vs Strategic Learning

- Let f be a binary classifier
- Deployment time: $(x, y) \sim D$ is seen as is by f in classical setting.
- Strategic setting: $(x, y) \to (x', y')$.
  - The change (in feature value) is made by strategic agents (more on it later) in response to deployed classifier f
  - If $y' = y$ then it is called gaming [a]
  - If also $y' \neq y$; the rule f is called performative prediction

[a] Strategic Classification, Hardt, Maggiddo, Papadimitriou, Wooters. ITCS 2016

Strategic Modification of data at deployment time

- Agents/Users want favourable outcome; 1 if classified positively and 0 otherwise.

Strategic Modification of data at deployment time

- Agents/Users want favourable outcome; 1 if classified positively and 0 otherwise.
- Classifier designed to predict true label accurately;
- Users optimal response to f

$$\Delta_f(x) \in \arg\min_{x' \in \mathcal{X}} \Big(\underbrace{f(x')}_{\text{classfier}} - \underbrace{c(x, x')}_{\text{cost}} \Big)$$

Strategic Modification of data at deployment time

- Agents/Users want favourable outcome; 1 if classified positively and 0 otherwise.
- Classifier designed to predict true label accurately;
- Users optimal response to f

$$\Delta_f(x) \in \arg\min_{x' \in \mathcal{X}} \Big(\underbrace{f(x')}_{\text{classfier}} \;-\; \underbrace{c(x, x')}_{\text{cost}} \Big)$$

- $c(x, x')$: cost of reporting x as x'.

Strategic Modification of data at deployment time

- Agents/Users want favourable outcome; 1 if classified positively and 0 otherwise.
- Classifier designed to predict true label accurately;
- Users optimal response to f

$$\Delta_f(x) \in \arg\min_{x' \in \mathcal{X}} \Big(\underbrace{f(x')}_{\text{classfier}} - \underbrace{c(x, x')}_{\text{cost}} \Big)$$

- $c(x, x')$: cost of reporting x as x'.
- cost is non-negative, truthful reports incur zero cost

Strategic Modification of data at deployment time

- Agents/Users want favourable outcome; 1 if classified positively and 0 otherwise.
- Classifier designed to predict true label accurately;
- Users optimal response to f

$$\Delta_f(x) \in \arg\min_{x' \in \mathcal{X}} \Big(\underbrace{f(x')}_{\text{classfier}} \; - \; \underbrace{c(x, x')}_{\text{cost}} \Big)$$

- $c(x, x')$: cost of reporting x as x'.
- cost is non-negative, truthful reports incur zero cost

- System's payoff: $\mathbb{E}_{(x,y)\sim\mathcal{D}}[f(\Delta_f(x)) = y]$.
  Throughout this talk we will consider strategic error.

$$f_s^* \in \arg\min_{f \in \mathcal{F}} \mathbb{E}_{(x,y)\sim\mathcal{D}}[f(\Delta_f(x)) \neq y]$$

Systems goal: Find f^* that adjusts to distribution shift in test data.

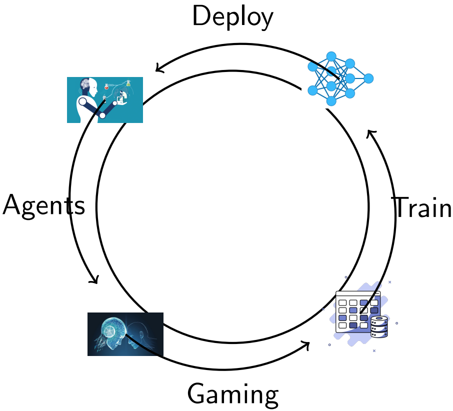
Applications

1. Health risk predictions
2. Bank loan approvals
3. Corporate hiring/promotions ...

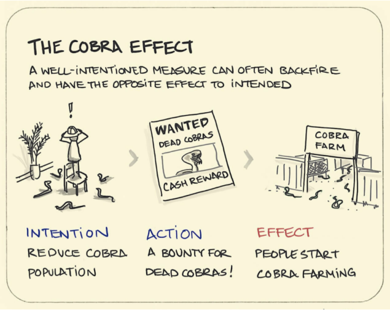
Applications

1. Health risk predictions
2. Bank loan approvals
3. Corporate hiring/promotions ...

Gaming:

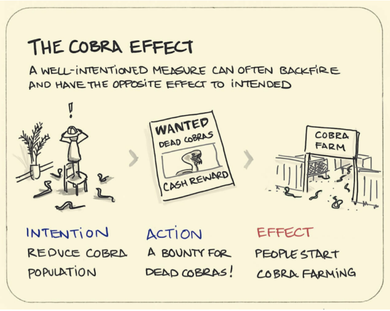


Gaming

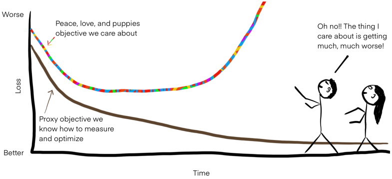


THE COBRA EFFECT
A WELL-INTENTIONED MEASURE CAN OFTEN BACKFIRE AND HAVE THE OPPOSITE EFFECT TO INTENDED
WANTED
DEAD COBRAS
CASH REWARD
COBRA FARM
INTENTION
REDUCE COBRA POPULATION
ACTION
A BOUNTY FOR DEAD COBRAS!
EFFECT
PEOPLE START COBRA FARMING

Gaming



THE COBRA EFFECT
A WELL-INTENTIONED MEASURE CAN OFTEN BACKFIRE AND HAVE THE OPPOSITE EFFECT TO INTENDED
WANTED
DEAD COBRAS
CASH REWARD
COBRA FARM
INTENTION
REDUCE COBRA POPULATION
ACTION
A BOUNTY FOR DEAD COBRAS!
EFFECT
PEOPLE START COBRA FARMING



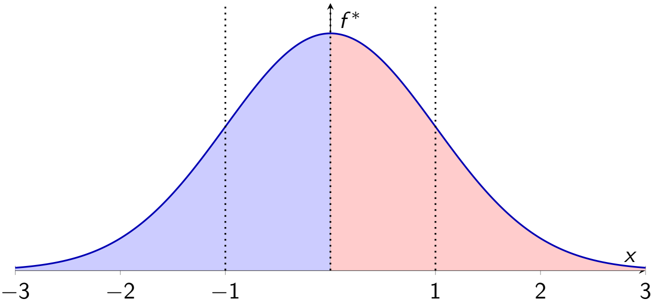
Worse
Peace, love, and puppies objective we care about
Oh no!! The thing I care about is getting much, much worse!
Loss
Proxy objective we know how to measure and optimize
Better
Time

Warmup

- Classical vs strategic classification: $\mathcal{D}_{train} \neq \mathcal{D}_{test}$ in strategic classification

Warmup

- Classical vs strategic classification: $\mathcal{D}_{train} \neq \mathcal{D}_{test}$ in strategic classification
- $\mathcal{D}_{test}$ is a obtained from implemented classifier f and $\mathcal{D}_{train}$



Strategic Classification

- Game theoretic interpretation: Two players, System and User(s) play following Stackelberg game

Strategic Classification

- Game theoretic interpretation: Two players, System and User(s) play following Stackelberg game
  - System learns a classifier f from training data $\mathcal{D}_{train}$

Strategic Classification

- Game theoretic interpretation: Two players, System and User(s) play following Stackelberg game
  - System learns a classifier f from training data $\mathcal{D}_{train}$
  - System declares f publicly

Strategic Classification

- Game theoretic interpretation: Two players, System and User(s) play following Stackelberg game
  - System learns a classifier f from training data $\mathcal{D}_{train}$
  - System declares f publicly
  - User, on observing f, misreport (at cost) her features to obtain the desired outcome from f

Goal: To minimize risk under strategic data distribution shift (strategic error).

Learning Objective

Objective Function

$$\min_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{n} \mathbb{1}[f(\Delta_f(x_i)) \neq y_i]$$

$$\textit{with} \quad \Delta_f(x) = \arg\max_{x'} f(x') - c(x, x')$$

- cost function important ($\Delta_f(x)$ unique?)
- Strategic Agents: Move to the point y on the boundary of the classifier at cost $c(x, y)$ only if $c(x, y) \leq 2$.
- Nested min-max problem

Examples

- $c(x,y) = ||y - x||_2$, $\mathcal{F}$ linear classifiers; $f(x) = w^T x$. Then

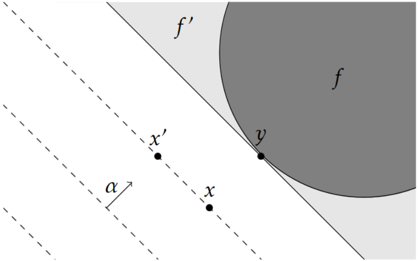
$$\Delta_w(x) = \begin{cases} x & \text{if } w^T x \geq 0 \text{ or } c(x;w) > 2. \\ proj(x;w) & \text{otherwise} \end{cases} \quad (1)$$

Examples

- $c(x,y) = ||y - x||_2$, $\mathcal{F}$ linear classifiers; $f(x) = w^T x$. Then

$$\Delta_w(x) = \begin{cases} x & \text{if } w^T x \geq 0 \text{ or } c(x;w) > 2. \\ proj(x;w) & \text{otherwise} \end{cases} \quad (1)$$

- (Separable) $c(x,y) = \max(0, c_2(y) - c_1(x))$ with $c_2(X) \subseteq c_2(X)$ e.g., $\langle \alpha, y - x \rangle_+$.

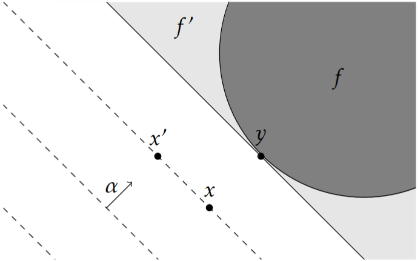


Examples

- $c(x, y) = ||y - x||_2$, $\mathcal{F}$ linear classifiers; $f(x) = w^T x$. Then

$$\Delta_w(x) = \begin{cases} x & \text{if } w^T x \geq 0 \text{ or } c(x; w) > 2. \\ proj(x; w) & \text{otherwise} \end{cases} \quad (1)$$

- (Separable) $c(x, y) = \max(0, c_2(y) - c_1(x))$ with $c_2(X) \subseteq c_2(X)$ e.g., $\langle \alpha, y - x \rangle_+$.



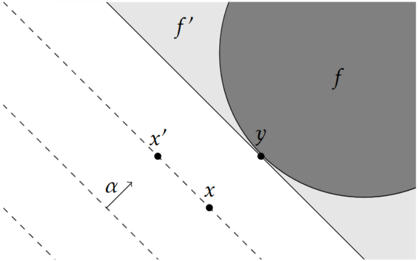
- (Decomposable) $c(x, y) = \sum_{i=1}^{d} \alpha_i |g_i(y_i) - g_i(x_i)|_+$

Examples

- $c(x,y) = ||y-x||_2$, $\mathcal{F}$ linear classifiers; $f(x) = w^T x$. Then

$$\Delta_w(x) = \begin{cases} x & \text{if } w^T x \geq 0 \text{ or } c(x;w) > 2. \\ proj(x;w) & \text{otherwise} \end{cases} \quad (1)$$

- (Separable) $c(x,y) = \max(0, c_2(y) - c_1(x))$ with $c_2(X) \subseteq c_2(X)$ e.g., $\langle \alpha, y - x \rangle_+$.



- (Decomposable) $c(x,y) = \sum_{i=1}^{d} \alpha_i |g_i(y_i) - g_i(x_i)|_+$ more on it later

Learnability Issues [1]

- Induced class $\mathcal{F}_\Delta = \{f(\Delta_f(x)) : f \in \mathcal{F}\}$.
- $SVC(\mathcal{F}) := VC(\mathcal{F}_\Delta)$.
- Cost is important
  - instance-invariant cost (cost of altering x is the same for all x); $SVC(\mathcal{F}) \approx VC(\mathcal{F})$ for linear f
  - instance-wise cost: SVC is ∞ even for linear classifiers.

[1] Sundaram et al. PAC-learning for Strategic Classification, JMLR 2023

Learning Results: Strategic ERM

Definition (Strategy Robust Learning)

An algorithm $\mathcal{A}$ is a strategy-robust learning algorithm for a class of cost functions $\mathcal{C}$ if for all distributions $\mathcal{D}$, for all class probabiliy functions η, all $c \in \mathcal{C}$ and for all ε and δ, given a description of c and access to labeled examples of the form $(x, \eta(x))$, where $x \sim \mathcal{D}$, $\mathcal{A}$ produces a classifier $f : \mathcal{X} \to \{-1, 1\}$ so that, with probability at least $1 - \delta$ over the samples,

$$Pr_{x \sim \mathcal{D}}(\eta(x) = f(\Delta_f(x))) > OPT(\mathcal{D}, c) - \varepsilon$$

Theorem

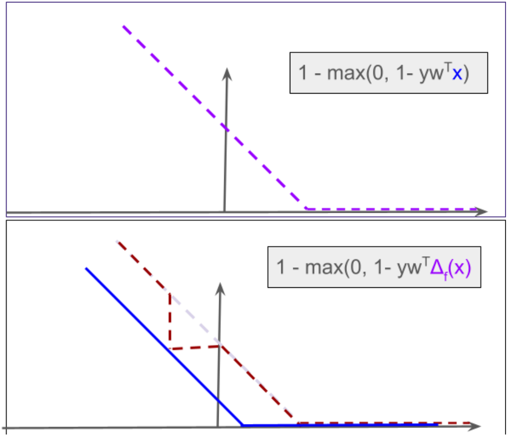
Let $\mathcal{H}$ be a concept class, $\mathcal{D}$ be a distribution and c be a separable cost function. Further, let m denote the number of samples and suppose

$$\mathcal{R}_m(\mathcal{H}) + 2\sqrt{\frac{\log(m+1)}{m}} + \sqrt{\frac{\log(2/\delta)}{8m}} \leq \frac{\varepsilon}{8}. \tag{2}$$

Then SERM outputs f such that w.p. atleast $1-\delta$,
$\mathbb{P}_{x \in \mathcal{D}}(\eta(x) = f(\Delta(x))) \geq \textsc{Opt}(\mathcal{D}, c) - \varepsilon.$

More on optimization Problem

- Why cant we use surrogate loss functions (such as hinge loss, log loss)



Variation 1: SC in the Dark [2]

[2] Ghalme et al. Strategic Classification in the Dark, ICML 2021.

SC in the Dark

1. Agent(s) may not have complete access to f;

SC in the Dark

1. Agent(s) may not have complete access to f;
2. Agents may have access to decisions by f; Example: OpenShufa

SC in the Dark

1. Agent(s) may not have complete access to f;
2. Agents may have access to decisions by f; Example: OpenShufa

Definition (Strategic error in the dark)

$$\mathrm{ERR}(f, \widehat{f}) = \mathbb{P}_{x \sim \mathcal{D}}(y \neq f(\Delta_{\widehat{f}}(x))) \quad (3)$$

SC in the Dark

1. Agent(s) may not have complete access to f;
2. Agents may have access to decisions by f; Example: OpenShufa

Definition (Strategic error in the dark)

$$\text{Err}(f, \widehat{f}) = \mathbb{P}_{x \sim \mathcal{D}}(y \neq f(\Delta_{\widehat{f}}(x))) \tag{3}$$

Who is in the dark?

SC in the Dark

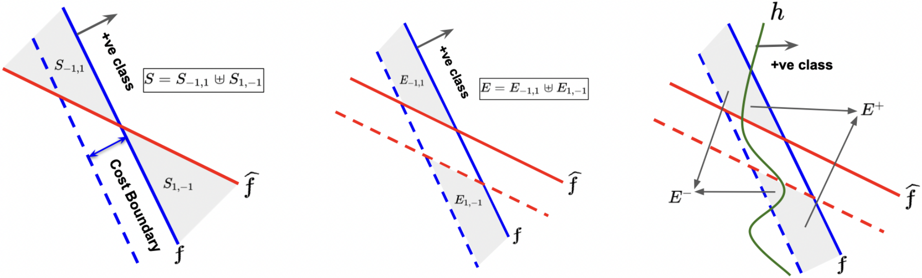
1. Agent(s) may not have complete access to f;
2. Agents may have access to decisions by f; Example: OpenShufa

Definition (Strategic error in the dark)

$$\textsc{Err}(f, \widehat{f}) = \mathbb{P}_{x \sim \mathcal{D}}(y \neq f(\Delta_{\widehat{f}}(x))) \tag{3}$$

Who is in the dark? By making f public, System can anticipate agents' response better (and construct robust f). By keeping f private, System is also in the dark as partially informed users may lead to unpredictable response.

Price of Opacity



Main Results

Definition (Price of Opacity (POP))

$$POP(f, f') := \textsc{Err}(f, f') - \textsc{Err}(f, f).$$

Here f is the System's classifier and f' is the classifier Agents' classifier (Agent responds to f').

Main Results

Definition (Price of Opacity (POP))

$$POP(f, f') := \textsc{Err}(f, f') - \textsc{Err}(f, f).$$

Here f is the System's classifier and f' is the classifier Agents' classifier (Agent responds to f').

Theorem (POP characterization)

If $\mathbb{P}_{x \sim \mathcal{D}}(x \in E) > 2\textsc{Err}(f^*, f^*) + 2\varepsilon$*, then* $POP > 0$.

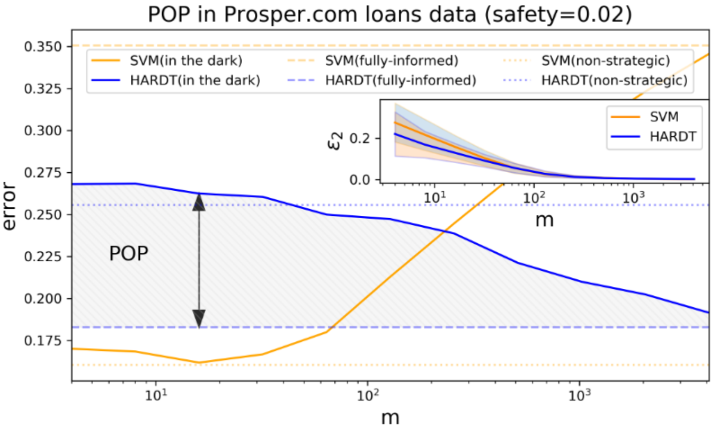


Figure: Price of Opacity is positive and decreases with the training samples m used to construct $\widehat{f}$.

Performative Prediction [3]

[3] Perdomo et al. Performative Prediction. ICML 2020

Variation 2: Performative Prediction

- SC assumption: Labels are immutable
- Performative Prediction: The (joint) distribution $\mathcal{D}$ changes to $D(\theta)$ where θ represents the parameters of the classification rule.
- Predictive Optimal policy $PO = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{Z \sim \mathcal{D}(\theta))} \ell(Z; \theta)$

Performative Prediction

- The deployed classifier has performative effect on qualification (improves with the cost/effort spent on obtaining positive outcome).
- Let classifier paramaters be θ then, $\mathcal{D}_{deploy} = \mathcal{D}(\theta)$.

Performative Prediction

- The deployed classifier has performative effect on qualification (improves with the cost/effort spent on obtaining positive outcome).
- Let classifier paramaters be θ then, $\mathcal{D}_{deploy} = \mathcal{D}(\theta)$.

Definition (Performative Risk)

$$PR(\theta) = \mathbb{R}_{Z \sim \mathcal{D}(\theta)} \ell(Z; \theta)$$

Performative Prediction

- The deployed classifier has performative effect on qualification (improves with the cost/effort spent on obtaining positive outcome).
- Let classifier paramaters be θ then, $\mathcal{D}_{deploy} = \mathcal{D}(\theta)$.

Definition (Performative Risk)

$$PR(\theta) = \mathbb{R}_{Z \sim \mathcal{D}(\theta)} \ell(Z; \theta)$$

Definition (Iterative Version)

$$\theta_{t+1} = \arg\min_{\theta} \mathbb{E}_{Z \sim \mathcal{D}(\theta_t)} \ell(Z; \theta)$$

Performative Prediction

- The deployed classifier has performative effect on qualification (improves with the cost/effort spent on obtaining positive outcome).
- Let classifier paramaters be θ then, $\mathcal{D}_{deploy} = \mathcal{D}(\theta)$.

Definition (Performative Risk)

$$PR(\theta) = \mathbb{R}_{Z\sim\mathcal{D}(\theta)}\ell(Z;\theta)$$

Definition (Iterative Version)

$$\theta_{t+1} = \arg\min_{\theta} \mathbb{E}_{Z\sim\mathcal{D}(\theta_t)}\ell(Z;\theta)$$

Definition (Performative Stability)

A model $f_{\theta_{ps}}$ is called performatively stable if

$$\theta_{PS} = \arg\min_{\theta} \mathbb{E}_{Z\sim\mathcal{D}(\theta_{PS})}\ell(z;\theta)) \tag{4}$$

Results: Performative Predictions

Theorem (Informal)

If the loss is smooth, strongly convex, and the mapping $\mathcal{D}(.)$ is sufficiently Lipschitz, then repeated risk minimization converges to performative stability at a linear rate.

Results: Performative Predictions

Theorem (Informal)

If the loss is smooth, strongly convex, and the mapping $\mathcal{D}(.)$ is sufficiently Lipschitz, then repeated risk minimization converges to performative stability at a linear rate.

Theorem (Informal)

If the loss is Lipschitz and strongly convex, and the map $\mathcal{D}()$ is Lipschitz, all stable points and performative optima lie in a small neighbourhood around each other.

Improvement Aware Strategic Classification

Improvement Aware SC

- Assume that η is component-wise increasing in x [4] and the cost function is decomposable.

[4] Feature value indicates qualification; higher the better.

Improvement Aware SC

- Assume that η is component-wise increasing in x [4] and the cost function is decomposable.
- Define
  - $\text{ERR}_s(f) := \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathbb{1}(f(\Delta_f(x)) \neq y)]$

[4] Feature value indicates qualification; higher the better.

Improvement Aware SC

- Assume that η is component-wise increasing in x [4] and the cost function is decomposable.
- Define
  - $\mathrm{ERR}_s(f) := \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathbb{1}(f(\Delta_f(x)) \neq y)]$
  - $\mathrm{ERR}_{\mathtt{imp}}(f) := \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathbb{1}(f(\Delta_f(x)) \neq y')]$ where $y' \sim \mathit{Bernoulli}(\eta(\Delta_f(x)))$
- Goal: Find optimal Strategy aware classifier i.e., $f^*_{\mathtt{imp}} \in \arg\min_{f\in\mathcal{F}} \mathrm{ERR}_{\mathtt{imp}}(f)$
- Also assume that the Bayes optimal classifier is also Linear.

[4] Feature value indicates qualification; higher the better.

Results

Theorem

$f^*(x; w, b) = f_s^*(x; w, b + \phi)$ *with* $\phi = \max_i \frac{w_i}{\alpha_i}$.

- f^* represents an optimal (linear) classifier with pristine, non-manipulated data; a naive approach

Results

Theorem

$f^*(x; w, b) = f_s^*(x; w, b + \phi)$ *with* $\phi = \max_i \frac{w_i}{\alpha_i}$.

- f^* represents an optimal (linear) classifier with pristine, non-manipulated data; a naive approach
- Optimal classifier with manipulated data with no-improvement; a pessimistic approach

Theorem

$f^*_{imp(x)} = f^*(x; w, b + \phi')$ *where* $\phi' \in [0, \max_i \frac{w_i}{\alpha_i}]$. *Furthermore,* $err_{imp}(f_s^*) \leq err_{imp}(f^*)$.

- The improvement aware classifier lies between naive and pessimistic classifiers.
- Also, the pessimistic classifier is a more reliable proxy for improvement aware classifier over a naive classfier.

Takeaways

- Traditional ML algorithms perform poorly in a strategic setting

Takeaways

- Traditional ML algorithms perform poorly in a strategic setting
- The other extreme; overfit to strategic nature

Takeaways

- Traditional ML algorithms perform poorly in a strategic setting
- The other extreme; overfit to strategic nature
- Strategic classifiers are learnable under reasonable assumptions on cost functions

Takeaways

- Traditional ML algorithms perform poorly in a strategic setting
- The other extreme; overfit to strategic nature
- Strategic classifiers are learnable under reasonable assumptions on cost functions
- Many questions: Heterogeneous Users, Social Burden, Information disparity, Herd Behavior....

Takeaways

- Traditional ML algorithms perform poorly in a strategic setting
- The other extreme; overfit to strategic nature
- Strategic classifiers are learnable under reasonable assumptions on cost functions
- Many questions: Heterogeneous Users, Social Burden, Information disparity, Herd Behavior....
- Beyond SC: Ranking, clustering, Online learning...

Takeaways

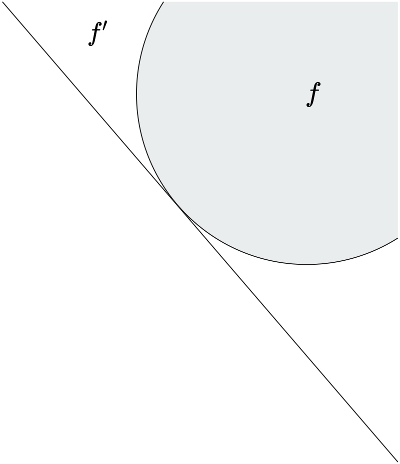
- Traditional ML algorithms perform poorly in a strategic setting
- The other extreme; overfit to strategic nature
- Strategic classifiers are learnable under reasonable assumptions on cost functions
- Many questions: Heterogeneous Users, Social Burden, Information disparity, Herd Behavior....
- Beyond SC: Ranking, clustering, Online learning...
- System Manipulation: strategic representation, User targeting, Persuasion ...

Takeaways

- Traditional ML algorithms perform poorly in a strategic setting
- The other extreme; overfit to strategic nature
- Strategic classifiers are learnable under reasonable assumptions on cost functions
- Many questions: Heterogeneous Users, Social Burden, Information disparity, Herd Behavior....
- Beyond SC: Ranking, clustering, Online learning...
- System Manipulation: strategic representation, User targeting, Persuasion ...

Thank you!

Strategic ERM



Algorithm 1: $\mathcal{A}$: gaming-robust classification algorithm for separable cost functions

1. **Inputs:** Labeled examples $(x_1, h(x_1)), \ldots, (x_m, h(x_m))$ from $x_i \sim \mathcal{D}$ i.i.d.. Also, a description of a separable cost function $c(x,y) = \max\{0, c_2(y) - c_1(x)\}$. ⇦
2. For $i = 1, \ldots, m$, let

$$t_i := c_1(x_i)$$

$$s_i := \begin{cases} \max\left(c_2(X) \cap [t_i, t_i + 2]\right) & c_2(X) \cap [t_i, t_i + 2] \neq \emptyset \\ \infty & c_2(X) \cap [t_i, t_i + 2] = \emptyset \,. \end{cases}$$

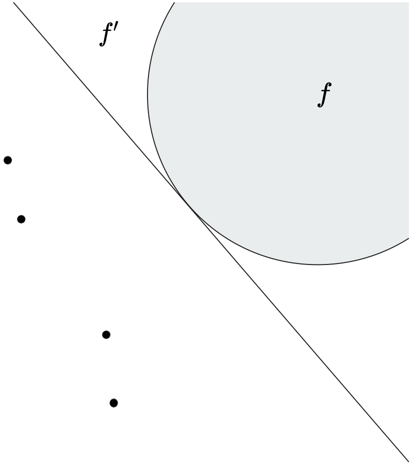
   For convenience, set $s_{m+1} = \infty$.
3. Compute

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

4. Find i^*, $1 \leqslant i^* \leqslant m+1$, that minimizes $\widehat{\mathrm{err}}(s_i)$.
5. **Return:** $f := c_2[s_{i^*}]$.

1

Strategic ERM



Algorithm 1: $\mathcal{A}$: gaming-robust classification algorithm for separable cost functions

1 **Inputs:** Labeled examples $(x_1, h(x_1)), \ldots, (x_m, h(x_m))$ from $x_i \sim \mathcal{D}$ i.i.d.. Also, a description of a separable cost function $c(x,y) = \max\{0, c_2(y) - c_1(x)\}$. ⇦

2 For $i = 1, \ldots, m$, let

$$t_i := c_1(x_i)$$

$$s_i := \begin{cases} \max(c_2(X) \cap [t_i, t_i + 2]) & c_2(X) \cap [t_i, t_i + 2] \neq \emptyset \\ \infty & c_2(X) \cap [t_i, t_i + 2] = \emptyset \,. \end{cases}$$

For convenience, set $s_{m+1} = \infty$.

3 Compute

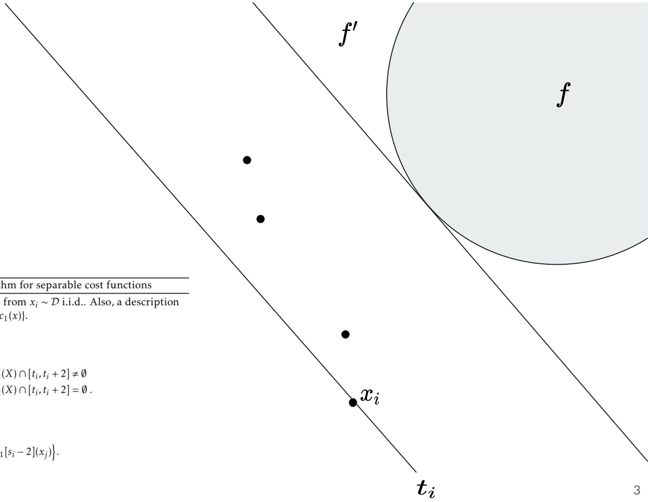
$$\widehat{\text{err}}(s_i) := \frac{1}{m} \sum_{j=1}^{m} \mathbf{1}\left\{h(x_j) \neq c_1[s_i - 2](x_j)\right\}.$$

4 Find i^*, $1 \leqslant i^* \leqslant m+1$, that minimizes $\widehat{\text{err}}(s_i)$.

5 **Return:** $f := c_2[s_{i^*}]$.

2

Strategic ERM



Algorithm 1: $\mathcal{A}$: gaming-robust classification algorithm for separable cost functions

1 **Inputs:** Labeled examples $(x_1, h(x_1)), \ldots, (x_m, h(x_m))$ from $x_i \sim \mathcal{D}$ i.i.d.. Also, a description of a separable cost function $c(x,y) = \max\{0, c_2(y) - c_1(x)\}$.

2 For $i = 1, \ldots, m$, let

$$t_i := c_1(x_i) \ \Leftarrow$$

$$s_i := \begin{cases} \max(c_2(X) \cap [t_i, t_i + 2]) & c_2(X) \cap [t_i, t_i + 2] \neq \emptyset \\ \infty & c_2(X) \cap [t_i, t_i + 2] = \emptyset \,. \end{cases}$$

For convenience, set $s_{m+1} = \infty$.

3 Compute

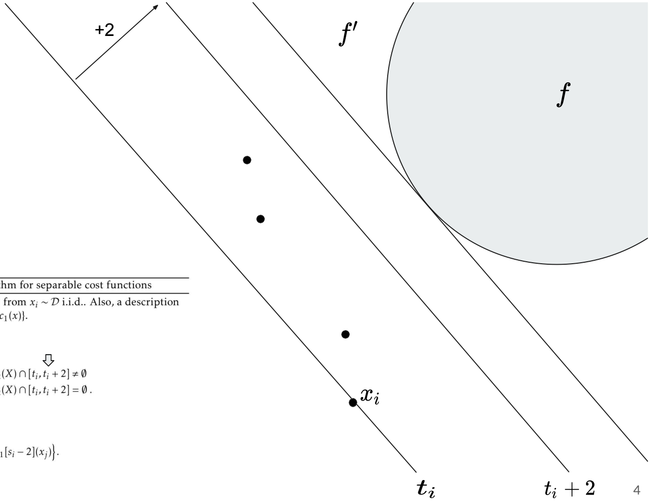
$$\widehat{\text{err}}(s_i) := \frac{1}{m} \sum_{j=1}^{m} \mathbf{1}\left\{h(x_j) \neq c_1[s_i - 2](x_j)\right\}.$$

4 Find i^*, $1 \leqslant i^* \leqslant m+1$, that minimizes $\widehat{\text{err}}(s_i)$.

5 **Return:** $f := c_2[s_{i^*}]$.

3

Strategic ERM



Algorithm 1: $\mathcal{A}$: gaming-robust classification algorithm for separable cost functions

1 **Inputs:** Labeled examples $(x_1, h(x_1)), \ldots, (x_m, h(x_m))$ from $x_i \sim \mathcal{D}$ i.i.d.. Also, a description of a separable cost function $c(x,y) = \max\{0, c_2(y) - c_1(x)\}$.

2 For $i = 1, \ldots, m$, let

$$t_i := c_1(x_i)$$

$$s_i := \begin{cases} \max(c_2(X) \cap [t_i, t_i + 2]) & c_2(X) \cap [t_i, t_i + 2] \neq \emptyset \\ \infty & c_2(X) \cap [t_i, t_i + 2] = \emptyset \,. \end{cases}$$

For convenience, set $s_{m+1} = \infty$.

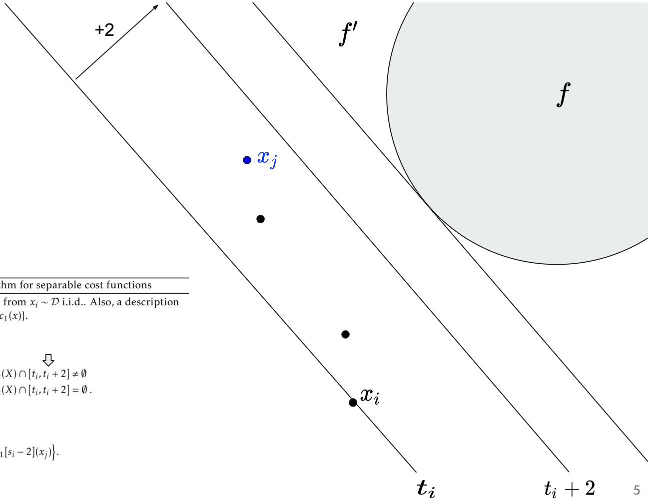
3 Compute

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

4 Find i^*, $1 \leqslant i^* \leqslant m+1$, that minimizes $\widehat{\mathrm{err}}(s_i)$.

5 **Return:** $f := c_2[s_{i^*}]$.

Strategic ERM



5

Algorithm 1: $\mathcal{A}$: gaming-robust classification algorithm for separable cost functions

---

1 **Inputs:** Labeled examples $(x_1, h(x_1)), \ldots, (x_m, h(x_m))$ from $x_i \sim \mathcal{D}$ i.i.d.. Also, a description of a separable cost function $c(x, y) = \max\{0, c_2(y) - c_1(x)\}$.

2 For $i = 1, \ldots, m$, let

$$t_i := c_1(x_i)$$

$$s_i := \begin{cases} \max\left(c_2(X) \cap [t_i, t_i + 2]\right) & c_2(X) \cap [t_i, t_i + 2] \neq \emptyset \\ \infty & c_2(X) \cap [t_i, t_i + 2] = \emptyset \,. \end{cases}$$

For convenience, set $s_{m+1} = \infty$.

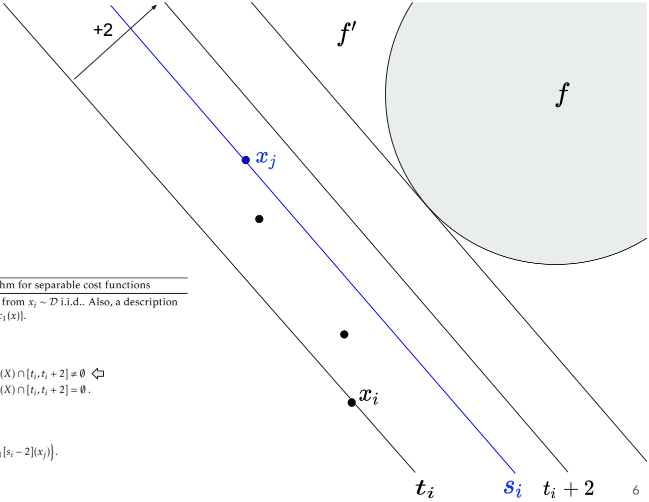
3 Compute

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

4 Find i^*, $1 \leqslant i^* \leqslant m + 1$, that minimizes $\widehat{\text{err}}(s_i)$.

5 **Return:** $f := c_2[s_{i^*}]$.

Strategic ERM



Algorithm 1: $\mathcal{A}$: gaming-robust classification algorithm for separable cost functions

1 **Inputs:** Labeled examples $(x_1, h(x_1)), \ldots, (x_m, h(x_m))$ from $x_i \sim \mathcal{D}$ i.i.d.. Also, a description of a separable cost function $c(x,y) = \max\{0, c_2(y) - c_1(x)\}$.

2 For $i = 1, \ldots, m$, let

$$t_i := c_1(x_i)$$

$$s_i := \begin{cases} \max(c_2(X) \cap [t_i, t_i + 2]) & c_2(X) \cap [t_i, t_i + 2] \neq \emptyset \\ \infty & c_2(X) \cap [t_i, t_i + 2] = \emptyset \,. \end{cases} \Leftarrow$$

For convenience, set $s_{m+1} = \infty$.

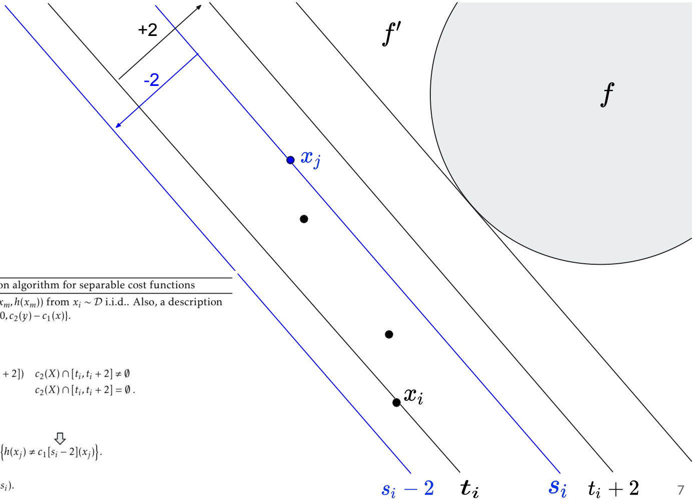
3 Compute

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

4 Find i^*, $1 \leqslant i^* \leqslant m+1$, that minimizes $\widehat{\text{err}}(s_i)$.

5 **Return:** $f := c_2[s_{i^*}]$.

Strategic ERM



Algorithm 1: $\mathcal{A}$: gaming-robust classification algorithm for separable cost functions

1. **Inputs:** Labeled examples $(x_1, h(x_1)), \ldots, (x_m, h(x_m))$ from $x_i \sim \mathcal{D}$ i.i.d.. Also, a description of a separable cost function $c(x, y) = \max\{0, c_2(y) - c_1(x)\}$.
2. For $i = 1, \ldots, m$, let

$$t_i := c_1(x_i)$$

$$s_i := \begin{cases} \max(c_2(X) \cap [t_i, t_i + 2]) & c_2(X) \cap [t_i, t_i + 2] \neq \emptyset \\ \infty & c_2(X) \cap [t_i, t_i + 2] = \emptyset \,. \end{cases}$$

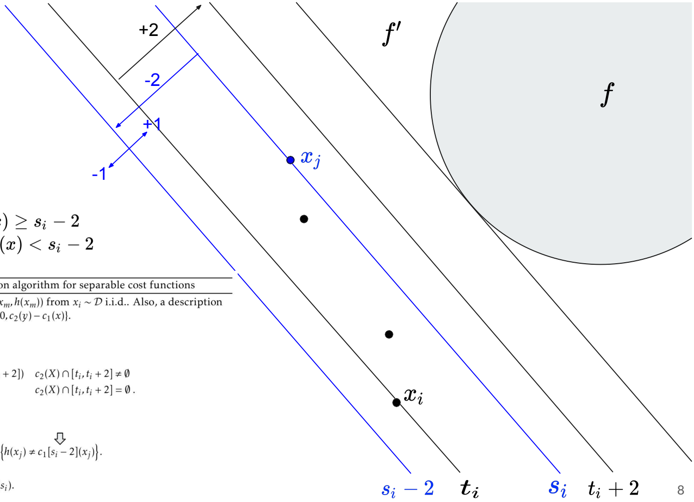
For convenience, set $s_{m+1} = \infty$.

3. Compute

$$\widehat{\text{err}}(s_i) := \frac{1}{m} \sum_{j=1}^{m} \mathbf{1}\left\{h(x_j) \neq c_1[\overset{\Downarrow}{s_i - 2}](x_j)\right\}.$$

4. Find i^*, $1 \leqslant i^* \leqslant m + 1$, that minimizes $\widehat{\text{err}}(s_i)$.
5. **Return:** $f := c_2[s_{i^*}]$.

Strategic ERM



$$c_1[s_i - 2](x) = \begin{cases} 1, \text{ if } c_1(x) \geq s_i - 2 \\ -1, \text{ if } c_1(x) < s_i - 2 \end{cases}$$

Algorithm 1: $\mathcal{A}$: gaming-robust classification algorithm for separable cost functions

1. **Inputs:** Labeled examples $(x_1, h(x_1)), \ldots, (x_m, h(x_m))$ from $x_i \sim \mathcal{D}$ i.i.d.. Also, a description of a separable cost function $c(x, y) = \max\{0, c_2(y) - c_1(x)\}$.
2. For $i = 1, \ldots, m$, let

$$t_i := c_1(x_i)$$

$$s_i := \begin{cases} \max(c_2(X) \cap [t_i, t_i + 2]) & c_2(X) \cap [t_i, t_i + 2] \neq \emptyset \\ \infty & c_2(X) \cap [t_i, t_i + 2] = \emptyset \, . \end{cases}$$

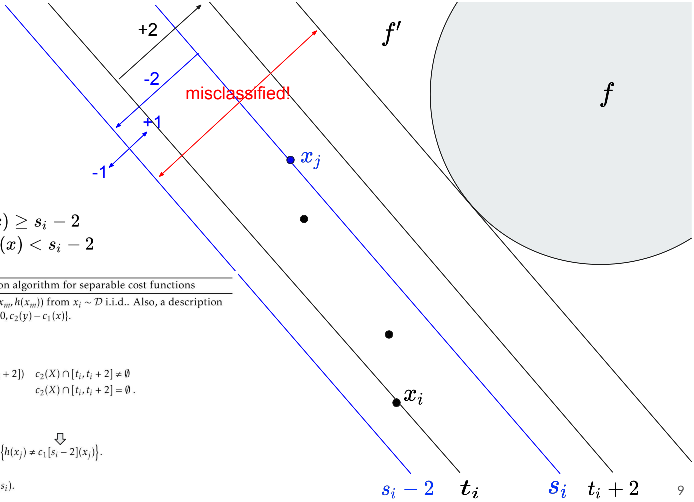
   For convenience, set $s_{m+1} = \infty$.

3. Compute

$$\widehat{\text{err}}(s_i) := \frac{1}{m} \sum_{j=1}^{m} \mathbf{1}\left\{h(x_j) \neq \overset{\Downarrow}{c_1[s_i - 2](x_j)}\right\}.$$

4. Find i^*, $1 \leqslant i^* \leqslant m+1$, that minimizes $\widehat{\text{err}}(s_i)$.
5. **Return:** $f := c_2[s_{i^*}]$.

Strategic ERM



$$c_1[s_i - 2](x) = \begin{cases} 1, & \text{if } c_1(x) \geq s_i - 2 \\ -1, & \text{if } c_1(x) < s_i - 2 \end{cases}$$

Algorithm 1: $\mathcal{A}$: gaming-robust classification algorithm for separable cost functions

1. **Inputs:** Labeled examples $(x_1, h(x_1)), \ldots, (x_m, h(x_m))$ from $x_i \sim \mathcal{D}$ i.i.d.. Also, a description of a separable cost function $c(x, y) = \max\{0, c_2(y) - c_1(x)\}$.
2. For $i = 1, \ldots, m$, let

$$t_i := c_1(x_i)$$

$$s_i := \begin{cases} \max(c_2(X) \cap [t_i, t_i + 2]) & c_2(X) \cap [t_i, t_i + 2] \neq \emptyset \\ \infty & c_2(X) \cap [t_i, t_i + 2] = \emptyset\,. \end{cases}$$

   For convenience, set $s_{m+1} = \infty$.
3. Compute

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

4. Find i^*, $1 \leqslant i^* \leqslant m + 1$, that minimizes $\widehat{\mathrm{err}}(s_i)$.
5. **Return:** $f := c_2[s_{i^*}]$.

Strategic ERM

Algorithm 1: $\mathcal{A}$: gaming-robust classification algorithm for separable cost functions

Inputs: Labeled examples $(x_1, h(x_1)), \ldots, (x_m, h(x_m))$ from $x_i \sim \mathcal{D}$ i.i.d.. Also, a description of a separable cost function $c(x,y) = \max\{0, c_2(y) - c_1(x)\}$.

For $i = 1, \ldots, m$, let

$$t_i := c_1(x_i)$$

$$s_i := \begin{cases} \max(c_2(X) \cap [t_i, t_i + 2]) & c_2(X) \cap [t_i, t_i+2] \neq \emptyset \\ \infty & c_2(X) \cap [t_i, t_i+2] = \emptyset \,. \end{cases}$$

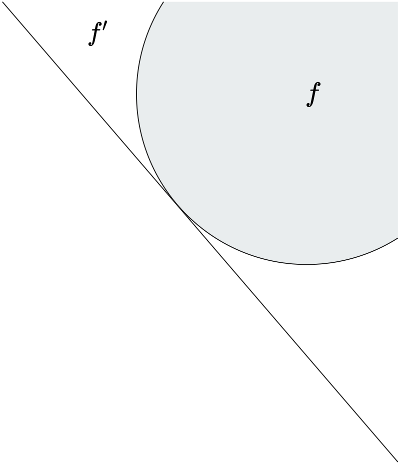
For convenience, set $s_{m+1} = \infty$.

Compute

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

Find i^*, $1 \leqslant i^* \leqslant m+1$, that minimizes $\widehat{\mathrm{err}}(s_i)$.

Return: $f := c_2[s_{i^*}]$.



10

$$c_2[s_i^*](x) = \begin{cases} 1 \text{ if } c_2(x) \geq s_i^* \\ -1 \text{ if } c_2(x) < s_i^* \end{cases}$$

Algorithm 1: $\mathcal{A}$: gaming-robust classification algorithm for separable cost functions

1 **Inputs:** Labeled examples $(x_1, h(x_1)), \ldots, (x_m, h(x_m))$ from $x_i \sim \mathcal{D}$ i.i.d.. Also, a description of a separable cost function $c(x,y) = \max\{0, c_2(y) - c_1(x)\}$.

2 For $i = 1, \ldots, m$, let

$$t_i := c_1(x_i)$$

$$s_i := \begin{cases} \max(c_2(X) \cap [t_i, t_i+2]) & c_2(X) \cap [t_i, t_i+2] \neq \emptyset \\ \infty & c_2(X) \cap [t_i, t_i+2] = \emptyset \,. \end{cases}$$

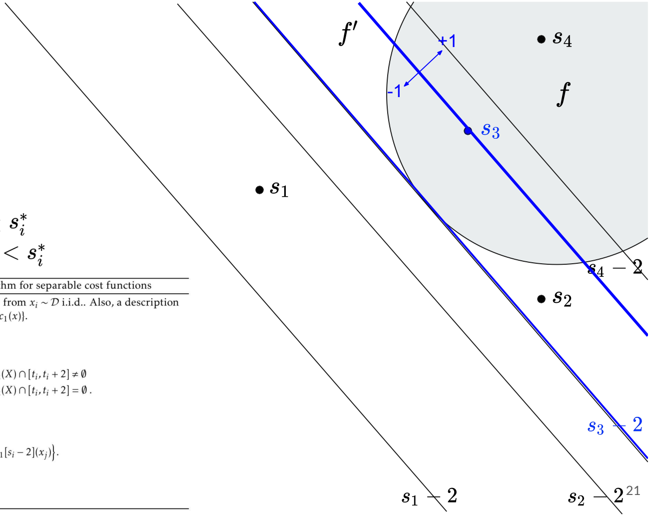
For convenience, set $s_{m+1} = \infty$.

3 Compute

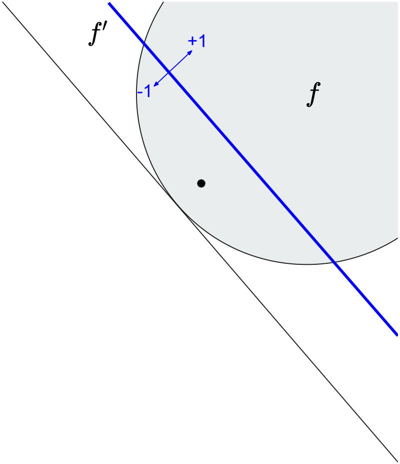
$$\widehat{\mathrm{err}}(s_i) := \frac{1}{m} \sum_{j=1}^{m} \mathbf{1}\left\{h(x_j) \neq c_1[s_i - 2](x_j)\right\}.$$

4 Find i^*, $1 \leqslant i^* \leqslant m+1$, that minimizes $\widehat{\mathrm{err}}(s_i)$.

5 **Return:** $f := c_2[s_{i^*}]$.

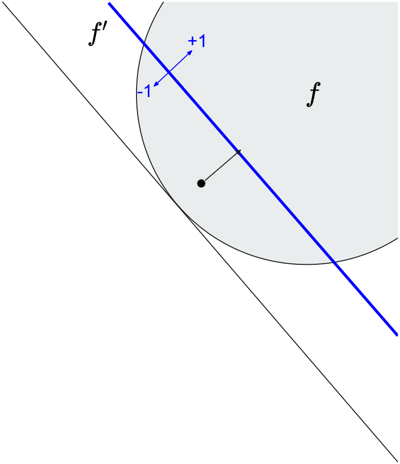


New input 1



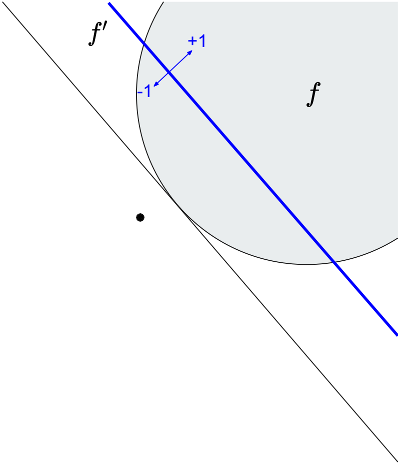
22

New input 1



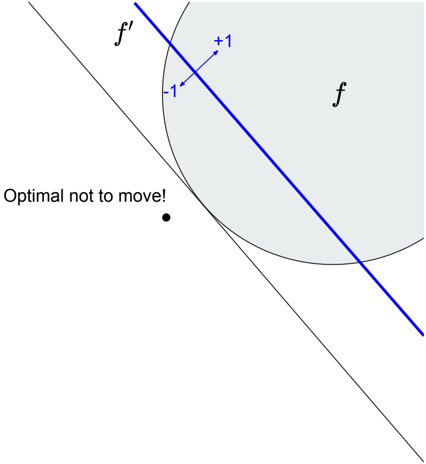
23

New input 2



24

New input 2



25