DS 574 Algorithmic Mechanism Design **Lecture #19**
Fall 2025 Prof. Kira Goldner

Machine Learning and Incentives

ML algorithms for decision-making are almost everywhere nowadays. Examples from headlines and responses to them:

- Whether you qualify for a loan or not.
- Whether you qualify for probation.
- Whether you get invited for an onsite interview after a video screening call.
- Whether students get admitted to college.

Are these responses honest effort exertion or gaming?

Problem: If ML algorithms ignore this strategic behavior, they risk making policy decisions that are incompatible with the original policy's goal.

The goal of policy makers' and mechanism designers' in using ML for decision-making is to learn from human data to create better decisions.

However, those who have a stake in the outcome can also learn and manipulate their data.

Example: Strategic Classification

Consider a university trying to classify student applicants as qualified (positive) or unqualified (negative) for admission. A datapoint represents a student's features: SAT score, GPA, class ranking, etc.

Suppose the university solves this problem. Now, given this classifier (trained on training data), what will happen?

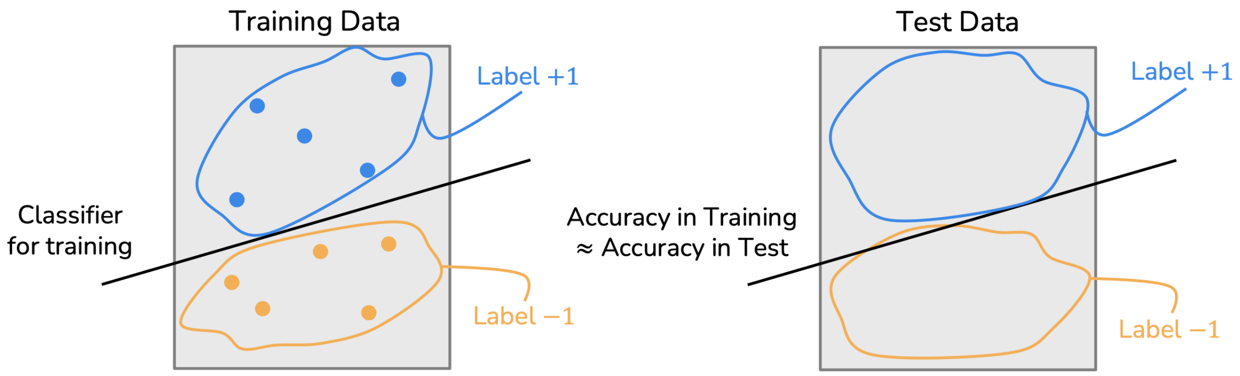

Figure 1: Example of Strategic Classification with students. Left: Setting a classifier for the training data. Right: What will then happen on the test data.

What will happen:

Root of the problem: Data corresponds to individuals who have **agency** and want to affect the decisions made on them by the ML algorithms.

The Offline Model

Let's formalize the above example with a university and student applicants into a *game*, as was first done by Hardt et al. [2016]. This will be a Stackelberg game, where the main player we worry about goes first and acts in a way that is defensive against all possible best-responses.

The players:

- University:
- Individual students:

The game:

1. Nature draws each agent's features (e.g., SAT score, class ranking, ...)

2. The learner commits to classifier

3. An agent observes

4. An agent reports to learner feature vector

5. The learner observes

6. The learner gets utility:

Let

$$\hat{x} = \arg\max_{y \in \mathcal{X}} \mathbb{E}_x[\mathbb{1}[\alpha(y) = 1] - c(x, y)].$$

The learner's goal is to select the best classifier given strategic responses, that is, to compute a Stackelberg Equilibrium:

$$\alpha^* = \arg\max_{f \in \mathcal{H}} \Pr_{x \sim \mathcal{D}}[\qquad\qquad\qquad\qquad]$$

The main result of Hardt et al. [2016] is there is a uniform strategy-robust learning algorithm for α^* in time and sample complexity $poly(m, 1/\varepsilon, \log(1/\delta))$ where the concept class is learnable from m examples up to error ε and confidence $1-\delta$ (for separable cost functions).

Follow up work by Zrnic et al. [2021] shows that the order of play is crucial: they study a setting where both parties learn, and show that how fast each adapts to the other impacts who's the "leader" in the game, and thus what the equilibrium is.

The Online Model

We'll now look at the model adapted for a *dynamic* or an *online* learning setting. The players are the same. The game is updated slightly.

For round $t \in [T]$:

1. Nature chooses an agent's features (e.g., SAT score, class ranking, ...) $x_t \in \mathcal{X} \subseteq [0, 1]^d$.
2. The learner picks classification rule $\alpha_t \in \mathcal{A} \subseteq [-1, 1]^{d+1}$.
3. The agent observes the classifier α_t and the datapoint (x_t, y_t) where $y_t \in \{-1, 1\}$.

4. The agent reports to the learner a feature vector $\hat{x}_t(\alpha_t)$ $(\neq x_t)$
5. The learner observes true label y_t.
6. The learner incurs classification loss $\ell(\alpha_t, \hat{x}_t(\alpha_t))$.

The learner's goal is to minimize Stackelberg Regret:

$$R(T) =$$

Assumptions are critical:

- What does the cost function to manipulate the agent's data look like?
- What does the learner's loss function look like, how does it depend on y? Binary? [Chen et al., 2020, Ahmadi et al., 2021] Hinge? Logistic? [Dong et al., 2018]

Big critiques:

- This is only for gaming, not honest effort.
- Agents have exact knowledge of the classifier. In reality, most classifiers are not exactly known, and we usually only have sample or probe access.

Fairness with Heterogenous Populations

So far we've been imagining that all students come from the same population, but that's obviously not true in reality. Two concurrent papers examine this problem when there are two populations: an *advantaged* population A and a *disadvantaged* population B.

We'll assume that the two populations are drawn from two different unknown distributions, which might even be the same. That is, both populations have qualified individuals.

For the advantaged population A, they have more access to tools to manipulate their feature vector, i.e., SAT prep classes and retakes. That is, it is less *costly* for those in A to change from x to $\hat{x}$ than it for those in B: $c_A(x, y) \leq c_B(x, y)$.

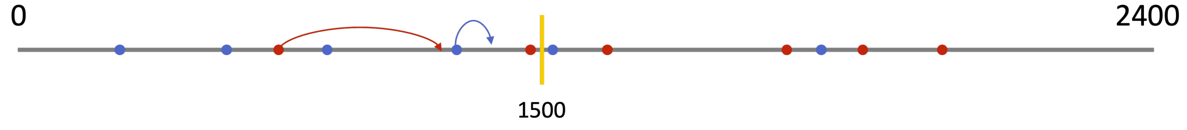

Figure 2: Example of Strategic Classification with two disparate populations.

Main question of Hu et al. [2019]: What if you subsidize the disadvantaged population by some β (fractionally *or* additively) so that it's less expensive for the population to manipulate? The learner pays for the subsidies, doesn't want false positives to be disproportionate between the two groups, but their primary objective is to maximize accuracy.

Result: There are examples where subsidies are not Pareto-improving for both groups A and B. That means that there exist individuals in both groups who are worse off after the subsidies.

Discussion:

Milli et al. [2019] shows a necessary trade-off between agent utility and learner utility (accuracy), and that this disproportionately impacts the disadvantaged population.

Other Examples of Incentivizing Effort

It turns out there are many areas of EconCS that can be thought of as [dis]incentivizing effort.

- Contracts
- Delegation
- Crowdsourcing: using information elicitation and scoring rules!

Algorithmic Contract Design

This is called a principal-agent problem: There is a "principal" hiring workers to do a job, and "agents" being hired to do the job.

Outcomes: There is a set of m possible outcomes for the principal. You can imagine binary outcomes: job completed, job not completed. Each outcome j has a reward r_j associated with it for the principal.

Actions: Each agent has a bunch of actions they can take. Think of this as different effort levels they could put in, or different methods they could use to get the job done. Each action i has a cost c_i for the agent to take it. Each action also stochastically leads to an outcome based on a probability matrix: q_{ij} is the probability of outcome j when action i is taken.

The main tension in contract design is that principals observe and pay for *outcomes*, but agents choose and bear the cost of *actions*.

Welfare? Utility? Design?

- The expected reward of action i is $R_i =$

  Then the expected welfare from action i is $W_i =$
- A *contract* is a payment rule $\mathbf{t}$ that consists of m non-negative payments or transfers $(t_1, \dots, t_m)$, one for each outcome j. All payments $t_j \geq 0$ must be non-negative, this is called "limited liability."
- Then for action i, $T_i = \mathbb{E}_{j \sim \mathbf{q}_i}[t_j] = \sum_{j \in [m]} q_{ij} t_j$ denotes the expected payment for action i.
- We now write expected utility for each party as a function of a taken *action* i:

  the principal's expected utility is $u_P(i \mid \mathbf{t}) =$

  and the agent's expected utility is $u_A(i \mid \mathbf{t}) =$

Then the agent will choose an action i to maximize their utility, and the goal of the principal is to design a contract $\mathbf{t}$ to maximize the principal's utility.

What does this remind you of?

Acknowledgements

This lecture was developed by using materials from Chara Podimata, and in particular, her 2021 tutorial on Incentive-Aware Machine Learning at ACM STOC.

References

Saba Ahmadi, Hedyeh Beyhaghi, Avrim Blum, and Keziah Naggita. The strategic perceptron. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 6–25, 2021.

Yiling Chen, Yang Liu, and Chara Podimata. Learning strategy-aware linear classifiers. *Advances in Neural Information Processing Systems*, 33:15265–15276, 2020.

Jinshuo Dong, Aaron Roth, Zachary Schutzman, Bo Waggoner, and Zhiwei Steven Wu. Strategic classification from revealed preferences. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 55–70, 2018.

Moritz Hardt, Nimrod Megiddo, Christos Papadimitriou, and Mary Wootters. Strategic classification. In *Proceedings of the 2016 ACM conference on innovations in theoretical computer science*, pages 111–122, 2016.

Lily Hu, Nicole Immorlica, and Jennifer Wortman Vaughan. The Disparate Effects of Strategic Manipulation. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 259–268, 2019.

Smitha Milli, John Miller, Anca D Dragan, and Moritz Hardt. The Social Cost of Strategic Classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 230–239, 2019.

Tijana Zrnic, Eric Mazumdar, Shankar Sastry, and Michael Jordan. Who leads and who follows in strategic classification? *Advances in Neural Information Processing Systems*, 34:15257–15269, 2021.