1. Motivation

Classical Approach to Mechanism Design
- Impose strategyproofness as a hard constraint
- Identify mechanism that optimizes given objective

Challenges: Multi-dimensional types, NP-hard, SP ⇒ undesirable properties

Alternative Approach to Mechanism Design
- Impose outcome rule as a hard constraint
- Learn payment rule that minimizes expected ex post regret

Main advantage: Outcome rule ⇒ computational/economic properties

Applications: Optimal rule is NP-hard, SP is unavailable/undesirable

2. Problem Statement

Input: Distribution over type space \( \Theta \), outcome space \( \Omega \)

Outcome rule \( g : \Theta \to \Omega \)

Payment rule \( p : \Theta \to \mathbb{R}_{\geq 0} \)

Goals: Minimum expected ex post regret
- Agent-independent prices
-\( utility: u_i(\theta', \theta) = v_i(\theta, g_i(\theta')) - p_i(\theta')\)
-\( regret: \max_{\theta \in \Theta} u_i((\theta', \theta'), \theta) - u_i((\theta', \theta'), \theta)\)

Note: Strategyproof ⇒ zero expected ex post regret

3. Mechanisms as Classifiers

Characterization of Strategyproof Mechanisms

Mechanism \((g, p)\) is strategyproof if and only if
-\( p_i(\theta) = t_i(\theta_i - \psi(\theta, \theta_i))\)
-\( g_i(\theta) = \arg \max_{\theta \in \Theta} f_i(\theta, \theta_i) - t_i(\theta_i - \psi(\theta, \theta_i))\)

for a price function \( t_i : \Theta_i \times \Theta_{-i} \to \mathbb{R}_{\geq 0} \)

Observation: Can interpret \( g_i \) as discriminant-based classifier

Example: Vickrey’s Auction

Discriminant-based classifier:
\[ h_i : \Theta \to \Omega \]
\[ h_i(\theta) = \arg \max_{\theta \in \Theta} f_i(\theta, \theta_i) \]

Discriminant function:
\[ f_i(\theta, \theta_i) = v_i(\theta, \theta_i) - t_i(\theta_i - \psi(\theta, \theta_i)) \]

Price function:
\[ t_i(\theta_i - \psi(\theta, \theta_i)) = \max_{\theta \neq \theta_i} v_j(\theta_j, +) \]
\[ t_i(\theta_i - \psi(\theta, \theta_i)) = 0 \]

Idea: Turn this around, train classifier and read off payment rule

4. The General Approach

Step 1: Generate training data for outcome rule \( g_i \):
\[ \{(\theta(i), g_i(\theta(i)), \ldots, \theta(i), g_i(\theta(i))\} \]

Step 2: Train admissible classifier \( h_i \) to predict outcome rule \( g_i \):
\[ h_i(\theta) = \arg \max_{\theta \in \Theta} f_i(\theta, \theta_i) + \frac{1}{2} w^T \psi(\theta_i - \psi(\theta, \theta_i)) \]
\[ = \arg \max_{\theta \in \Theta} f_i(\theta, \theta_i) - t_i(\theta_i - \psi(\theta, \theta_i)) \]

Step 3: Read off payment rule \( p_i(\theta, \theta_i) = t_i(\theta_i - \psi(\theta, \theta_i)) \)

Note: Agent-independent prices

5. Theoretical Properties

Exact Classification and Strategyproofness

Perfect admissible classifiers \( h_1, \ldots, h_n \Rightarrow SP \) mechanism \((g, p)\)

Approximate Classification and Ex Post Regret

Admissible classifiers \( h_1, \ldots, h_n \) with min generalization error
\( \Rightarrow \) mechanism \((g, p)\) with min expected ex post regret

6. Applying the Framework

Structural Support Vector Machines
- Train classifier \( h(\theta) = \arg \max_{\theta \in \Theta} w^T \psi(\theta, \theta_i) \)
- For admissibility impose \( \psi(\theta, \theta_i) = (v_i(\theta_i, \theta_i), \psi(\theta, \theta_i)) \)
- Use decomposable feature map \( \psi(\theta, \theta_i) = \phi(\chi(\theta_i, \theta_i)) \)

Note: Non-linear prices via “kernel trick”

Dealing with Individual Rationality Violations
- Payment offset: Decrease all payments by given amount
- Null loss fix: Penalize false “null outcome” predictions more

7. Experimental Results

Multi-Minded Combinatorial Auction
- Agents get bundles of items, are interested in fixed # of bundles
- Outcome rule greedily maximizes social welfare

8. Conclusion

Results
- New paradigm for computational mechanism design
- Encouraging experimental results

Future Directions
- Constrain properties of payment rule (e.g., budgets)
- Additional design goals (e.g., interim regret)