Payment Rules through Discriminant-Based Classifiers

Paul Dütting
Stanford University

Felix Fischer, Pichayut Jirapinyo, John. K. Lai, Benjamin Lubin, and David C. Parkes

Econ Theory Lunch @ Stanford
March 31, 2014
Introduction

Mechanism Design

- Agents have private values for different outcomes
  - Agents have quasi-linear utilities, i.e., utility = value - payment
- Mechanism solicits reports, computes outcome and payments
  - Mechanism optimizes objective, e.g., welfare = sum of true values
Mechanism Design

- Agents have private values for different outcomes
  - Agents have quasi-linear utilities, i.e., utility = value - payment
- Mechanism solicits reports, computes outcome and payments
  - Mechanism optimizes objective, e.g., welfare = sum of true values

Example Applications

- Allocation of arrival/departure slots to airlines
- Reallocating spectrum rights
- Selling online ads
Classic Approach to Mechanism Design

- Find mechanism to optimize desiderata
- Subject to incentive constraints
Introduction

Classic Approach to Mechanism Design

- Find mechanism to optimize desiderata
- Subject to incentive constraints

Example: Strategyproof Welfare Maximization

- Mechanism (approximately) maximizes welfare
  - Sum of values is (approximately) maximal
- Mechanism is strategyproof
  - Each agent maximizes utility by reporting truthfully
  - No matter what reports of other agents are
Example: Strategyproof Welfare Maximization (cont’d)

- If welfare can be optimized exactly
  - Can use Vickrey-Clarke-Groves (VCG) mechanism
- If welfare cannot be optimized exactly
  - Generally approximation plus VCG payments won’t work

Exemption: Maximal-in-range algorithms (Nisan and Ronen 2002)
Example: Strategyproof Welfare Maximization (cont’d)

- Strategyproofness in single-parameter settings
  - Outcome rule must be monotone, and then threshold payments
- Strategyproofness in multi-parameter settings
  - Implementable outcome rules characterized by cyclic monotonicity
  - Some partial characterizations, e.g., certain LP-based algorithms

See: (Archer and Tardos 2001), (Ashlagi et al. 2010), (Lavi and Swamy 2005)
Introduction

Example: Strategyproof Welfare Maximization (cont’d)

- Strategyproofness in single-parameter settings
  - Outcome rule must be monotone, and then threshold payments
- Strategyproofness in multi-parameter settings
  - Implementable outcome rules characterized by cyclic monotonicity
  - Some partial characterizations, e.g., certain LP-based algorithms

See: (Archer and Tardos 2001), (Ashlagi et al. 2010), (Lavi and Swamy 2005)

**Bottom line:** Classic approach requires *de novo* design
Question: Can we somehow automate this process?

Our Approach: Fix Heuristic, Learn Payment Rule

- Start with any heuristic for achieving objective
  - Possible objectives are welfare, fairness, etc.
  - Revenue considerations should be secondary
- Learn payments that make it maximally strategyproof
  - Payments will minimize expected ex post regret
Introduction

Question: Can we somehow automate this process?

Our Approach: Fix Heuristic, Learn Payment Rule

- Start with any heuristic for achieving objective
  - Possible objectives are welfare, fairness, etc.
  - Revenue considerations should be secondary
- Learn payments that make it maximally strategyproof
  - Payments will minimize expected ex post regret
- Enabled by surprisingly close connection between
  - (Approximately) strategyproof mechanism design
  - (Approximately) exact classification
Introduction

**Question:** Can we somehow automate this process?

**Our Approach: Fix Heuristic, Learn Payment Rule**

- Start with any heuristic for achieving objective
  - Possible objectives are welfare, fairness, etc.
  - Revenue considerations should be secondary
- Learn payments that make it maximally strategyproof
  - Payments will minimize expected ex post regret
- Enabled by surprisingly close connection between
  - (Approximately) strategyproof mechanism design
  - (Approximately) exact classification

Upshot: Only need to worry about heuristic, not incentives
Related Approach #1: Automated Mechanism Design

- Formulate search for mechanism that optimizes objective subject to incentives as a mathematical program
- Use the computer to solve it

See: (Conitzer and Sandholm 2002), (Conitzer and Guo 2010)

Related Approach #2: Black Box Reductions

- Turn any approximation algorithm into Bayes-Nash incentive compatible (BNIC) mechanism with essentially the same performance guarantee

See: (Bei and Huang 2010), (Hartline et al. 2010), (Cai et al. 2013)
Outline of this Talk

1 - Model and Definitions

2 - Theoretical Results

3 - Experimental Results

4 - Conclusion and Future Work
Model and Definitions

First Part: Mechanism design definitions

Toy example: Single-item, second price auction
Model and Definitions

First Part: Mechanism design definitions

Toy example: Single-item, second price auction

Mechanism Design Problem

- Agents: $N = \{1, \ldots, n\}$
- Outcomes: $\Omega = \prod_i \Omega_i$
- Types: $\Theta = \prod_i \Theta_i$
- Valuations: $v_i : \Theta_i \times \Omega_i \rightarrow \mathbb{R}_{\geq 0}$
- Distribution on types: $\Delta$
Model and Definitions

First Part: Mechanism design definitions

**Toy example:** Single-item, second price auction

**Mechanism Design Problem**

- **Agents:** $N = \{1, \ldots, n\}$
- **Outcomes:** $\Omega = \prod_i \Omega_i$
- **Types:** $\Theta = \prod_i \Theta_i$
- **Valuations:** $v_i : \Theta_i \times \Omega_i \to \mathbb{R}_{\geq 0}$
- **Distribution on types:** $\Delta$

**(Direct) Mechanism**

- **Outcome rule:** $g : \Theta \to \prod_i \Omega_i$
- **Payment rule:** $p : \Theta \to \mathbb{R}_{\geq 0}$
Strategyproofness

- Each agent maximizes its utility by reporting its true type, no matter what other agents report
- \( \forall i \in N, \theta_i \in \Theta_i \), and \( \theta' \in \Theta : u_i((\theta_i, \theta'_{-i}), \theta_i) \geq u_i((\theta'_i, \theta'_{-i}), \theta_i) \)
**Model and Definitions**

**Strategyproofness**

- Each agent maximizes its utility by reporting its true type, no matter what other agents report
- \( \forall i \in N, \theta_i \in \Theta_i, \text{ and } \theta' \in \Theta : u_i((\theta_i, \theta'_{-i}), \theta_i) \geq u_i((\theta'_i, \theta'_{-i}), \theta_i) \)

**Ex Post Regret**

- How much can agent \( i \) gain if instead of reporting type \( \theta_i \) it reported type \( \theta'_i \) instead?
- \( \forall i \in N, \theta \in \Theta : r_i(\theta_i, \theta_{-i}) = \max_{\theta'_i \in \Theta_i} u_i((\theta'_i, \theta_{-i}), \theta_i) - u_i((\theta_i, \theta_{-i}), \theta_i) \)
Model and Definitions

Strategyproofness

- Each agent maximizes its utility by reporting its true type, no matter what other agents report
- \( \forall i \in N, \theta_i \in \Theta_i, \text{ and } \theta' \in \Theta : u_i((\theta_i, \theta'_-i), \theta_i) \geq u_i((\theta'_i, \theta'_-i), \theta_i) \)

Ex Post Regret

- How much can agent \( i \) gain if instead of reporting type \( \theta_i \) it reported type \( \theta'_i \) instead?
- \( \forall i \in N, \theta \in \Theta : r_i(\theta_i, \theta_-i) = \max_{\theta'_i \in \Theta_i} u_i((\theta'_i, \theta_-i), \theta_i) - u_i((\theta_i, \theta_-i), \theta_i) \)

Hint: Think about regret as quantifiable relaxation of strategyproofness
Characterization of Strategyproofness

Mechanism \((g, p)\) is strategyproof if and only if for every \(\theta \in \Theta\),

- \(p_i(\theta) = t_i(\theta_{-i}, g_i(\theta))\) for all \(i \in N\), and
- \(g_i(\theta) \in \arg \max_{\omega_i' \in \Omega_i} (v_i(\theta_i, \omega_i') - t_i(\theta_{-i}, \omega_i'))\) for all \(i \in N\),

for a price function \(t_i : \Theta_{-i} \times \Omega_i \rightarrow \mathbb{R}_{\geq 0}\).
Model and Definitions

Second Part: Classification definition

**Toy example:** Pictures of animals, “cat” or “no cat”
Second Part: Classification definition

Toy example: Pictures of animals, “cat” or “no cat”

Classification Problem

- Input domain: $X$
- Output domain: $Y$
- Distribution on inputs: $D$
- Target function: $h^*: X \rightarrow Y$
Model and Definitions

Second Part: Classification definition

Toy example: Pictures of animals, “cat” or “no cat”

Classification Problem

- Input domain: $X$
- Output domain: $Y$
- Distribution on inputs: $D$
- Target function: $h^*: X \rightarrow Y$

Discriminant-Based Classifier

- Classifier: $h(x) = \arg \max_{y \in Y} f(x, y)$
- Discriminant function: $f: X \times Y \rightarrow \mathbb{R}$
Model and Definitions

Exact Classification

- Outcome of classifier and outcome of target function always coincide
- $\forall x \in X: h(x) = h^*(x)$
Model and Definitions

Exact Classification

- Outcome of classifier and outcome of target function always coincide
- $\forall x \in X: h(x) = h^*(x)$

Classification Error

- Classifier minimizes generalization error = expected discriminant loss
- $R_D(h) = \int_X [f(x, h(x)) - f(x, h^*(x))] D(x) \, dx$
Theoretical Results

Observation: Structurally almost identical problems (!)

Mapping between Classification and Mechanism Design

- Input domain: $X$ - Types: $\Theta$
- Output domain: $Y$ - Outcomes: $\Omega$
- Distribution on inputs: $D$ - Distribution on types: $\Delta$
- Target function: $h^*$ - Outcome rule: $g$

For simplicity: Assume agent symmetry for remainder of talk
Strategyproof Mechanism

- \( \forall \theta \in \Theta : g_1(\theta) = \arg \max_{\omega_1 \in \Omega_1} (v(\theta_1, \omega'_1) - t_1(\theta_{-1}, \omega'_1)) \)

Exact Discriminant-Based Classifier

- \( \forall x \in X : h^*(x) = h(x) = \arg \max_{y \in Y} f(x, y) \)
Theoretical Results

Strategyproof Mechanism

- \( \forall \theta \in \Theta: g_1(\theta) = \arg\max_{\omega_1' \in \Omega_1} (v(\theta_1, \omega_1') - t_1(\theta_1, \omega_1')) \)

Exact Discriminant-Based Classifier

- \( \forall x \in X: h^*(x) = h(x) = \arg\max_{y \in Y} f(x, y) \)
- Rename: \( X \rightarrow \Theta, h^* \rightarrow g_1, Y \rightarrow \Omega_1 \)
Theoretical Results

Strategyproof Mechanism

- $\forall \theta \in \Theta: g_1(\theta) = \arg\max_{\omega'_1 \in \Omega_1} (v(\theta_1, \omega'_1) - t_1(\theta_{-1}, \omega'_1))$

Exact Discriminant-Based Classifier

- $\forall x \in X : h^*(x) = h(x) = \arg\max_{y \in Y} f(x, y)$
- Rename: $X \rightarrow \Theta$, $h^* \rightarrow g_1$, $Y \rightarrow \Omega_1$
- $\forall \theta \in \Theta : g_1(\theta) = h(\theta) = \arg\max_{\omega'_1 \in \Omega_1} f(\theta, \omega'_1)$
Theoretical Results

Idea: Impose additional structure on discriminant function

Admissible Discriminant-Based Classifier

- Restrict how discriminant function can depend on input and output
- \( \forall \theta \in \Theta, \omega_1 \in \Omega_1: f(\theta, \omega_1) = v_1(\theta_1, \omega_1) - t_1(\theta_{-1}, \omega_1) \)

Links Classification Accuracy to Regret

- Discriminant loss = Ex post regret
- Generalization error = Expected ex post regret
Theoretical Results

Putting it all together, this suggests the following approach:

1. Given type space $\Theta$, distribution over types $\Delta$, and outcome rule $g_1 : \Theta \rightarrow \Omega_1$
Putting it all together, this suggests the following approach:

1. Given type space $\Theta$, distribution over types $\Delta$, and outcome rule $g_1: \Theta \rightarrow \Omega_1$.
2. Generate training samples consisting of types-outcome to agent 1 pairs $(\theta^t, g_1(\theta^t))_{t=1,...,T}$. 

Paul Dütting

Payment Rules

March 31, 2014
Putting it all together, this suggests the following approach:

1. Given type space $\Theta$, distribution over types $\Delta$, and outcome rule $g_1 : \Theta \rightarrow \Omega_1$

2. Generate training samples consisting of types-outcome to agent 1 pairs $(\theta^t, g_1(\theta^t))_{t=1,\ldots,T}$

3. Train admissible discriminant-based classifier $h(\theta) = \arg\max_{\omega_1 \in \Omega_1} (v_1(\theta_1, \omega_1) - t_1(\theta_{-1}, \omega_1))$ to predict outcome rule $g_1$
Putting it all together, this suggests the following approach:

1. Given type space $\Theta$, distribution over types $\Delta$, and outcome rule $g_1 : \Theta \rightarrow \Omega_1$
2. Generate training samples consisting of types-outcome to agent 1 pairs $(\theta^t, g_1(\theta^t))_{t=1,...,T}$
3. Train admissible discriminant-based classifier $h(\theta) = \arg \max_{\omega_1 \in \Omega_1} (v_1(\theta_1, \omega_1) - t_1(\theta_{-1}, \omega_1))$ to predict outcome rule $g_1$
4. Use $t = (t_1, \ldots, t_1)$ as payment rule in mechanism
Theoretical Results

Summary of Theoretical Results

**Theorem 1.** Exact admissible classifier

\[ h(\theta) = \arg \max_{\omega_1 \in \Omega_1} f(\theta, \omega_1) \]

with \( f(\theta, \omega_1) = v_1(\theta_1, \omega_1) - t_1(\theta_{-1}, \omega_1) \)

for agent-symmetric outcome rule \( g \) induces strategyproof mechanism \((g, t)\).
Theoretical Results

Summary of Theoretical Results

**Theorem 1.** Exact admissible classifier \( h(\theta) = \arg \max_{\omega_1 \in \Omega_1} f(\theta, \omega_1) \) with \( f(\theta, \omega_1) = v_1(\theta_1, \omega_1) - t_1(\theta_{-1}, \omega_1) \) for agent-symmetric outcome rule \( g \) induces strategyproof mechanism \((g, t)\).

**Theorem 2.** Admissible classifier \( h(\theta) = \arg \max_{\omega_1 \in \Omega_1} f(\theta, \omega_1) \) with \( f(\theta, \omega_1) = v_1(\theta_1, \omega_1) - t_1(\theta_{-1}, \omega_1) \) for agent-symmetric outcome rule \( g \) that minimizes generalization error induces mechanism \((g, t)\) that minimizes expected ex post regret.
Theoretical Results

Summary of Theoretical Results

**Theorem 1.** Exact admissible classifier \( h(\theta) = \arg \max_{\omega_1 \in \Omega_1} f(\theta, \omega_1) \)
with \( f(\theta, \omega_1) = v_1(\theta_1, \omega_1) - t_1(\theta_{-1}, \omega_1) \) for agent-symmetric outcome rule \( g \) induces strategyproof mechanism \((g, t)\).

**Theorem 2.** Admissible classifier \( h(\theta) = \arg \max_{\omega_1 \in \Omega_1} f(\theta, \omega_1) \)
with \( f(\theta, \omega_1) = v_1(\theta_1, \omega_1) - t_1(\theta_{-1}, \omega_1) \) for agent-symmetric outcome rule \( g \)
that minimizes generalization error induces mechanism \((g, t)\) that minimizes expected ex post regret.

**Remarks:** Agent symmetry can be dropped, in fact if and only if
Experimental Results

Question: Is there a machine learning framework that does what we want?
Experimental Results

Question: Is there a machine learning framework that does what we want?

Support Vector Machines

- Discriminant-based classifiers for binary classification
- Support efficient, non-linear classification via “kernel trick”

See: (Vapnik 1979), (Vapnik and Cortes 1995)
Experimental Results

**Question:** Is there a machine learning framework that does what we want?

**Support Vector Machines**

- Discriminant-based classifiers for binary classification
- Support efficient, non-linear classification via “kernel trick”

See: (Vapnik 1979), (Vapnik and Cortes 1995)

**Structural Support Vector Machines**

- Discriminant-based classifiers for multi-class classification
- Support efficient, non-linear classification via “kernel trick”

See: (Tsochantaridis et al. 2005), (Joachims et al. 2009)
Experimental Results

Question: How good is proposed approach in practice?

Experiments with three Applications

- Single-item auction with welfare maximizing allocation rule
- Multi-minded combinatorial auction with greedy outcome rule
- Assignment problem with max-min fair allocation rule

Note: First is implementable, last two are not
Experimental Results

Single-Item Auction, Welfare Maximization

- Two agents, one item
- Values independent, uniform on [0, 1]
- RBF kernel with parameters $C \in \{10^4, 10^5\}, \gamma \in \{0.01, 0.1, 1\}$
- Training set size 300, validation test size 1000
Experimental Results

Single-Item Auction, Welfare Maximization (Cont’d)

![Graph showing learned payment vs. value of agent 2 with lines for $\chi_1$ and second price]
Experimental Results

Multi-Minded Combinatorial Auction, Greedy

• Five agents, five items, three bundles each
• To determine bundles:
  ▶ With $p = 1/4$ chose item uniformly from remaining ones
  ▶ With $p = 3/4$ stop
• To determine values:
  ▶ Let $c$ and $d_i$ be $m$-dimensional vectors with entries chosen uniformly from $(0, 1]$
  ▶ Represent $j$-th bundle by vector $S_{i,j} \in \{0, 1\}^m$
  ▶ Let value $v(S_{i,j}) = \min_{S'_{i,j} \subseteq S_{i,j}} \left( \frac{\langle S'_{i,j}, \beta c + (1-\beta)d_i \rangle}{m} \right)^\zeta$
  ▶ Where $\beta = 1/2$ and $\zeta$ controls degree of complementarity ($\zeta < 1$ means substitutes)
• RBF kernel with parameters $C \in \{10^4, 10^5\}, \gamma \in \{0.01, 0.1, 1\}$
• Training set size 100, 300, 500, validation test size 1000
Multi-Minded Combinatorial Auction, Greedy (Cont’d)
Assignment Problem, Max-Min Fair

- Variable number of $n$ agents and $n$ items
- Values uniformly from $[0, 1]$
- RBF kernel with parameters $C \in \{10, 10^3, 10^5\}$ and $\gamma \in \{0.1, 0.5, 1\}$
- Training set size 600, validation set size 1000
Experimental Results

Assignment Problem, Max-Min Fair (Cont’d)

![Graph showing regret vs number of agents for different functions](image-url)
Conclusion and Future Work

Conclusion

- New paradigm for approximately strategyproof mechanism design
- Based on surprisingly close connection between
  - (Approximately) exact classification
  - (Approximately) strategyproof mechanism design
- Low ex post regret in experiments, even for non-implementable rules

Future Work

- Use this approach to design new mechanisms
- Improve and extend the proposed approach
  - Constrain properties of payment rules (e.g., budgets)
  - Additional design goals (e.g., interim regret)
- Adopt similar approach for setting without money
Conclusion and Future Work

Conclusion

- New paradigm for approximately strategyproof mechanism design
- Based on surprisingly close connection between
  - (Approximately) exact classification
  - (Approximately) strategyproof mechanism design
- Low ex post regret in experiments, even for non-implementable rules

Future Work

- Use this approach to design new mechanisms
- Improve and extend the proposed approach
  - Constrain properties of payment rules (e.g., budgets)
  - Additional design goals (e.g., interim regret)
- Adopt similar approach for setting *without* money
References

Payment Rules for Discriminant-Based Classifiers
P. Dütting, F. Fischer, P. Jirapinyo, J. K. Lai, B. Lubin, and D. C. Parkes
Conference on Electronic Commerce (EC’12), Valencia, Spain, 2012

Payment Rules for Discriminant-Based Classifiers
P. Dütting, F. Fischer, P. Jirapinyo, J. K. Lai, B. Lubin, and D. C. Parkes
Transactions on Economics and Computation (TEAC), to appear