Mechanism Design through Statistical Machine Learning: Part I (Auctions)

David C. Parkes

Computer Science
John A. Paulson School of Engineering and Applied Sciences
Harvard University

January 12, 2016

Optimization where the Inputs are Private

Example: allocate landing rights at Schiphol airport.

- An agent i's *type* x_i specifies her value for each possible *allocation* $y \in Y$
- Design goal: maximize sum total value
- Constraint: *incentive compaibility*

Optimization where the Inputs are Private

Example: allocate landing rights at Schiphol airport.

- An agent i's *type* x_i specifies her value for each possible *allocation* $y \in Y$
- Design goal: maximize sum total value
- Constraint: *incentive compaibility*

Optimization where the Inputs are Private

Example: allocate landing rights at Schiphol airport.

- An agent i's *type* x_i specifies her value for each possible *allocation* $y \in Y$
- Design goal: maximize sum total value
- Constraint: *incentive compaibility*

Optimization where the Inputs are Private

- Alternative $y \in Y$. n agents.
- *Valuation type* $x_i \in X$, defines *value* $v_i(x_i, y)$.
- *Design goal*:

$$\max_{y \in Y} \quad g(x_1, \ldots, x_n, y)$$
$$\text{s.t.} \quad \textit{incentive compatibility} \text{ (IC)}$$

- *IC*: truthful reporting is a *dominant strategy* for each agent
- Typical to use *payments* to align incentives.

Optimization where the Inputs are Private

- Alternative $y \in Y$. n agents.
- *Valuation type* $x_i \in X$, defines *value* $v_i(x_i, y)$.
- *Design goal*:

$$\max_{y \in Y} \quad g(x_1, \ldots, x_n, y)$$
$$\text{s.t.} \quad \textit{incentive compatibility } \text{(IC)}$$

- *IC*: truthful reporting is a *dominant strategy* for each agent
- Typical to use *payments* to align incentives.

Optimization where the Inputs are Private

- Alternative $y \in Y$. n agents.
- *Valuation type* $x_i \in X$, defines *value* $v_i(x_i, y)$.
- *Design goal*:

$$\max_{y \in Y} \quad g(x_1, \ldots, x_n, y)$$
$$\text{s.t.} \quad \textit{incentive compatibility} \text{ (IC)}$$

- *IC*: truthful reporting is a *dominant strategy* for each agent
- Typical to use *payments* to align incentives.

Example: Vickrey Auction

- An agent's *type* x_i specifies her value for an item.
  - e.g., values \$10, \$8, \$4 (agents 1, 2 and 3).
- *Design goal*: allocate to agent with maximum value
- Solution:
  - Receive reports $\hat{x}_1, \ldots, \hat{x}_n$
  - Allocate to highest bid, for second-highest bid amount
- Incentive compatible, optimal.

Example: Vickrey Auction

- An agent's *type* x_i specifies her value for an item.
  - e.g., values \$10, \$8, \$4 (agents 1, 2 and 3).
- *Design goal*: allocate to agent with maximum value
- Solution:
  - Receive reports $\hat{x}_1, \ldots, \hat{x}_n$
  - Allocate to highest bid, for second-highest bid amount
- Incentive compatible, optimal.

Example: Vickrey Auction

- An agent's *type* x_i specifies her value for an item.
  - e.g., values \$10, \$8, \$4 (agents 1, 2 and 3).
- *Design goal*: allocate to agent with maximum value
- Solution:
  - Receive reports $\hat{x}_1, \ldots, \hat{x}_n$
  - Allocate to highest bid, for second-highest bid amount
- Incentive compatible, optimal.

Example: Vickrey-Clarke-Groves mechanism

Example: allocate landing rights at Schiphol airport.

- An agent's type x_i specifies her value for each possible allocation $y \in Y$
- *Design goal*: maximize sum total value
- Solution:
  - Receive reports $\hat{x}_1, \ldots, \hat{x}_n$
  - Choose $y^* \in \arg\max_Y \sum_i v_i(\hat{x}_i, y)$
  - Charge $\sum_{j \neq i} v_j(\hat{x}_j, y^{-i}) - \sum_{j \neq i} v_j(\hat{x}_j, y^*)$ to agent i
- Incentive compatible, optimal.

Example: Vickrey-Clarke-Groves mechanism

Example: allocate landing rights at Schiphol airport.

- An agent's type x_i specifies her value for each possible allocation $y \in Y$
- *Design goal*: maximize sum total value
- Solution:
  - Receive reports $\hat{x}_1, \ldots, \hat{x}_n$
  - Choose $y^* \in \arg\max_Y \sum_i v_i(\hat{x}_i, y)$
  - Charge $\sum_{j \neq i} v_j(\hat{x}_j, y^{-i}) - \sum_{j \neq i} v_j(\hat{x}_j, y^*)$ to agent i
- Incentive compatible, optimal.

Example: Vickrey-Clarke-Groves mechanism

Example: allocate landing rights at Schiphol airport.

- An agent's type x_i specifies her value for each possible allocation $y \in Y$
- *Design goal*: maximize sum total value
- Solution:
  - Receive reports $\hat{x}_1, \ldots, \hat{x}_n$
  - Choose $y^* \in \arg\max_Y \sum_i v_i(\hat{x}_i, y)$
  - Charge $\sum_{j \neq i} v_j(\hat{x}_j, y^{-i}) - \sum_{j \neq i} v_j(\hat{x}_j, y^*)$ to agent i
- Incentive compatible, optimal.

An Incentive Mechanism

A *mechanism* $M = (f, t)$:

- Receive reports $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_n)$
- *Outcome rule*: choose $y^* = f(\hat{x})$
- *Payment rule*: charge each agent i an amount $t_i(\hat{x}, y^*) \in \mathbb{R}$

A mechanism is *incentive-compatible* if truthful reporting is a dominant strategy, for all reports of others.

An Incentive Mechanism

A *mechanism* $M = (f, t)$:

- Receive reports $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_n)$
- *Outcome rule*: choose $y^* = f(\hat{x})$
- *Payment rule*: charge each agent i an amount $t_i(\hat{x}, y^*) \in \mathbb{R}$

A mechanism is *incentive-compatible* if truthful reporting is a dominant strategy, for all reports of others.

Challenges in Mechanism Design

- *Single-dimensional mechanism design* well understood
  - Valuation types $x_i \in \mathbb{R}$, value for alternative y monotone in x_i
  - Myerson (1981)
- *Multi-dimensional mechanism design* largely unsolved
  - Vickrey-Clarke-Groves mechanism only general solution
    - limited to welfare-optimality, often intractable
  - Analytical bottleneck: conditions such as *cyclic-monotonicity* hard to work with (Rochet'81)
- Some *computational progress* (Cai et al.'13, Alaei et al.'12, Daskalakis et al.'15), but not for general problems.

Challenges in Mechanism Design

- *Single-dimensional mechanism design* well understood
  - Valuation types $x_i \in \mathbb{R}$, value for alternative y monotone in x_i
  - Myerson (1981)
- *Multi-dimensional mechanism design* largely unsolved
  - Vickrey-Clarke-Groves mechanism only general solution
    - limited to welfare-optimality, often intractable
  - Analytical bottleneck: conditions such as *cyclic-monotonicity* hard to work with (Rochet'81)
- Some *computational progress* (Cai et al.'13, Alaei et al.'12, Daskalakis et al.'15), but not for general problems.

Challenges in Mechanism Design

- *Single-dimensional mechanism design* well understood
  - Valuation types $x_i \in \mathbb{R}$, value for alternative y monotone in x_i
  - Myerson (1981)
- *Multi-dimensional mechanism design* largely unsolved
  - Vickrey-Clarke-Groves mechanism only general solution
    - limited to welfare-optimality, often intractable
  - Analytical bottleneck: conditions such as *cyclic-monotonicity* hard to work with (Rochet'81)
- Some *computational progress* (Cai et al.'13, Alaei et al.'12, Daskalakis et al.'15), but not for general problems.

Automated Design via Learning

$$\left.\begin{array}{ll} \max_{y \in Y} & g(x_1, \ldots, x_n, y) \\ \text{s.t.} & \text{IC} \end{array}\right\} \approx f(x)$$

Given f. Learn a payment rule to accompany f:

- Input: training examples $\{(x^k, f(x^k))\}_k$ (generated with $x \sim_{IID} D$)
- Learn a *payment rule* t_w such that $M = (f, t_w)$, is *approximately IC*

Benefits:

- Rule f can be an algorithm (address comput. intractability)
- Allows graceful degradation to approximate-IC; avoid the analytical bottleneck.

Automated Design via Learning

$$\left.\begin{array}{ll} \max_{y \in Y} & g(x_1, \ldots, x_n, y) \\ \text{s.t.} & \text{IC} \end{array}\right\} \approx f(x)$$

Given f. Learn a payment rule to accompany f:

- Input: training examples $\{(x^k, f(x^k))\}_k$ (generated with $x \sim_{IID} D$)
- Learn a *payment rule* t_w such that $M = (f, t_w)$, is *approximately IC*

Benefits:

- Rule f can be an algorithm (address comput. intractability)
- Allows graceful degradation to approximate-IC; avoid the analytical bottleneck.

Automated Design via Learning

$$\left.\begin{array}{ll} \max_{y \in Y} & g(x_1, \ldots, x_n, y) \\ \text{s.t.} & \text{IC} \end{array}\right\} \approx f(x)$$

Given f. Learn a payment rule to accompany f:

- Input: training examples $\{(x^k, f(x^k))\}_k$ (generated with $x \sim_{IID} D$)
- Learn a *payment rule* t_w such that $M = (f, t_w)$, is *approximately IC*

Benefits:

- Rule f can be an algorithm (address comput. intractability)
- Allows graceful degradation to approximate-IC; avoid the analytical bottleneck.

Automated Design via Learning

$$\left.\begin{array}{ll} \max_{y \in Y} & g(x_1, \ldots, x_n, y) \\ \text{s.t.} & \text{IC} \end{array}\right\} \approx f(x)$$

Given f. Learn a payment rule to accompany f:

- Input: training examples $\{(x^k, f(x^k))\}_k$ (generated with $x \sim_{IID} D$)
- Learn a *payment rule* t_w such that $M = (f, t_w)$, is *approximately IC*

Benefits:

- Rule f can be an algorithm (address comput. intractability)
- Allows graceful degradation to approximate-IC; avoid the analytical bottleneck.

Background: Characterization of IC mechanisms

Theorem 1

Mechanism (f, t) is IC if and only if:

- *Agent-independence*: price to i for y is $t_i(\hat{x}_{-i}, y)$
- *No-regret*: $\forall i: \ f(\hat{x}) \in \arg\max_y [v_i(\hat{x}_i, y) - t_i(\hat{x}_{-i}, y)]$

Example:

- Bids \$10, \$8, \$4.
- Price to agent 2 is \$10 for the item, \$0 o.w. $\Rightarrow$ no regret for not receiving item!

Background: Characterization of IC mechanisms

Theorem 1

Mechanism (f, t) is IC if and only if:

- *Agent-independence*: price to i for y is $t_i(\hat{x}_{-i}, y)$
- *No-regret*: $\forall i : \; f(\hat{x}) \in \arg\max_y [v_i(\hat{x}_i, y) - t_i(\hat{x}_{-i}, y)]$

Example:

- Bids $\$10, \$8, \$4$.
- Price to agent 2 is \$10 for the item, \$0 o.w. ⇒ no regret for not receiving item!

Approximate IC

Fix type x_i, reports of others. y^* is choice of mechanism. Let $t_i(y)$ denote price function. Two possibilities:

Case 1 (No regret):

$$v_i(x_i, y^*) - t_i(y^*) \geq \max_y [v_i(x_i, y) - t_i(y)]$$

Case 2 (Regret > 0):

$$regret_t(x) = \max_y [v_i(x_i, y) - t_i(y)] - (v_i(x_i, y^*) - t_i(y^*)) > 0$$

Definition 1

Payment rule t_w is *maximally-IC* given f and D if

$$t_w \in \arg\min_t \mathbb{E}_{x \sim D}[regret_t(x)]$$

- Expected regret $= 0 \iff$ mech is IC
- Generally interested in low expected regret

Approximate IC

Fix type x_i, reports of others. y^* is choice of mechanism. Let $t_i(y)$ denote price function. Two possibilities:

Case 1 (No regret):

$$v_i(x_i, y^*) - t_i(y^*) \geq \max_y [v_i(x_i, y) - t_i(y)]$$

Case 2 (Regret > 0):

$$regret_t(x) = \max_y [v_i(x_i, y) - t_i(y)] - (v_i(x_i, y^*) - t_i(y^*)) > 0$$

Definition 1

Payment rule t_w is *maximally-IC* given f and D if

$$t_w \in \arg\min_t \mathbb{E}_{x \sim D}[regret_t(x)]$$

- Expected regret = 0 $\iff$ mech is IC
- Generally interested in low expected regret

Approximate IC

Fix type x_i, reports of others. y^* is choice of mechanism. Let $t_i(y)$ denote price function. Two possibilities:

Case 1 (No regret):

$$v_i(x_i, y^*) - t_i(y^*) \geq \max_y [v_i(x_i, y) - t_i(y)]$$

Case 2 (Regret > 0):

$$regret_t(x) = \max_y [v_i(x_i, y) - t_i(y)] - (v_i(x_i, y^*) - t_i(y^*)) > 0$$

Definition 1

Payment rule t_w is *maximally-IC* given f and D if

$$t_w \in \arg\min_t \mathbb{E}_{x \sim D}[regret_t(x)]$$

- Expected regret $= 0 \iff$ mech is IC
- Generally interested in low expected regret

Approximate IC

Fix type x_i, reports of others. y^* is choice of mechanism. Let $t_i(y)$ denote price function. Two possibilities:

Case 1 (No regret):

$$v_i(x_i, y^*) - t_i(y^*) \geq \max_y [v_i(x_i, y) - t_i(y)]$$

Case 2 (Regret > 0):

$$regret_t(x) = \max_y [v_i(x_i, y) - t_i(y)] - (v_i(x_i, y^*) - t_i(y^*)) > 0$$

Definition 1

Payment rule t_w is *maximally-IC* given f and D if

$$t_w \in \arg\min_t \mathbb{E}_{x \sim D}[regret_t(x)]$$

- Expected regret = 0 $\iff$ mech is IC
- Generally interested in low expected regret

Approximate IC

Fix type x_i, reports of others. y^* is choice of mechanism. Let $t_i(y)$ denote price function. Two possibilities:

Case 1 (No regret):

$$v_i(x_i, y^*) - t_i(y^*) \geq \max_y [v_i(x_i, y) - t_i(y)]$$

Case 2 (Regret > 0):

$$regret_t(x) = \max_y [v_i(x_i, y) - t_i(y)] - (v_i(x_i, y^*) - t_i(y^*)) > 0$$

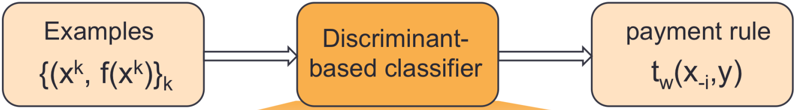
Definition 1

Payment rule t_w is *maximally-IC* given f and D if

$$t_w \in \arg\min_t \mathbb{E}_{x \sim D}[regret_t(x)]$$

- Expected regret $= 0 \iff$ mech is IC
- Generally interested in low expected regret

Overall Approach



$$f_w(x) \in \arg\max_y \left[v_i(x_i, y) - w^\top \psi(x_{-i}, y)\right]$$

Review: Discriminant-Based Classifier

- *Feature space* X^n.
- *Label space* Y.
- Learn *hypothesis* $h \in \mathcal{H}$.

For *parameters* w, define a *discriminant function*

$$H_w : X^n \times Y \mapsto \mathbb{R}.$$

The corresponding *classifier* is:

$$f_w(x) \in \arg\max_y H_w(x, y).$$

Review: Discriminant-Based Classifier

- *Feature space* X^n.
- *Label space* Y.
- Learn *hypothesis* $h \in \mathcal{H}$.

For *parameters* w, define a *discriminant function*

$$H_w : X^n \times Y \mapsto \mathbb{R}.$$

The corresponding *classifier* is:

$$f_w(x) \in \arg\max_y H_w(x, y).$$

Review: Discriminant-Based Classifier

- *Feature space* X^n.
- *Label space* Y.
- Learn *hypothesis* $h \in \mathcal{H}$.

For *parameters* w, define a *discriminant function*

$$H_w : X^n \times Y \mapsto \mathbb{R}.$$

The corresponding *classifier* is:

$$f_w(x) \in \arg\max_y H_w(x, y).$$

Exact Classifier gives an IC mechanism

Define a special *discriminant function*:

$$H_w(x,y) = v_i(x_i,y) - \underbrace{w^\top \psi(x_{-i},y)}_{\text{payment } t_w(x_{-i},y)},$$

for *features* $\psi(x_{-i},y) \in \mathbb{R}^m$.

Theorem 2

An exact classifier for outcome rule f provides IC mech. (f, t_w).

Proof: Let $y^* = f(x)$. Because classifier f_w is exact, then $v_i(x_i,y^*) - t_w(x_{-i},y^*) \geq v_i(x_i,y) - t_w(x_{-i},y)$.
$\Rightarrow (f, t_w)$ is IC (by Theorem 1).

Exact Classifier gives an IC mechanism

Define a special *discriminant function*:

$$H_w(x, y) = v_i(x_i, y) - \underbrace{w^\top \psi(x_{-i}, y)}_{\text{payment } t_w(x_{-i}, y)},$$

for *features* $\psi(x_{-i}, y) \in \mathbb{R}^m$.

Theorem 2

An exact classifier for outcome rule f provides IC mech. (f, t_w).

Proof: Let $y^* = f(x)$. Because classifier f_w is exact, then $v_i(x_i, y^*) - t_w(x_{-i}, y^*) \geq v_i(x_i, y) - t_w(x_{-i}, y)$.

$\Rightarrow (f, t_w)$ is IC (by Theorem 1).

Exact Classifier gives an IC mechanism

Define a special *discriminant function*:

$$H_w(x, y) = v_i(x_i, y) - \underbrace{w^\top \psi(x_{-i}, y)}_{\text{payment } t_w(x_{-i}, y)},$$

for *features* $\psi(x_{-i}, y) \in \mathbb{R}^m$.

Theorem 2

An exact classifier for outcome rule f provides IC mech. (f, t_w).

Proof: Let $y^* = f(x)$. Because classifier f_w is exact, then $v_i(x_i, y^*) - t_w(x_{-i}, y^*) \geq v_i(x_i, y) - t_w(x_{-i}, y)$.
$\Rightarrow (f, t_w)$ is IC (by Theorem 1).

A Risk-Optimal Classifier gives Maximal IC

Definition 2

The *discriminant loss function* is

$$L_w(x, f(x)) = H_w(x, f_w(x)) - H_w(x, f(x)) \geq 0.$$

Theorem 3

A classifier f_w that *minimizes exp loss* $\mathbb{E}_x[L_w(x, f(x))]$ provides a mechanism (f, t_w) that is *maximally-IC*.

Proof: The discriminant loss corresponds to regret, because

$$L_w(x, f(x)) = \max_y [H_w(x, y)] - H_w(x, y^*) = regret_{t_w}(x),$$

since $H_w(x, y) = v_i(x_i, y) - w^\top \psi(x_{-i}, y)$.

A Risk-Optimal Classifier gives Maximal IC

Definition 2

The *discriminant loss function* is

$$L_w(x, f(x)) = H_w(x, f_w(x)) - H_w(x, f(x)) \geq 0.$$

Theorem 3

A classifier f_w that *minimizes exp loss* $\mathbb{E}_x[L_w(x, f(x))]$ provides a mechanism (f, t_w) that is *maximally-IC*.

Proof: The discriminant loss corresponds to regret, because

$$L_w(x, f(x)) = \max_y [H_w(x, y)] - H_w(x, y^*) = regret_{t_w}(x),$$

since $H_w(x, y) = v_i(x_i, y) - w^\top \psi(x_{-i}, y)$.

Mechanism design via Risk-optimal Classifiers

The program:

1. Given training data $\{(x^k, f(x^k))\}_k$. Define feature map ψ.
2. Learn a classifier with discriminant function

$$H_w(x, y) = v_i(x_i, y) - w^\top \psi(x_{-i}, y)$$

that minimizes expected loss.

3. Obtain mechanism (f, t_w), with payment rule

$$t_w(x_{-i}, y) = w^\top \psi(x_{-i}, y)$$

Use *structural support vector machines* to solve the multi-class classification problem (Joachims et al. 2009).

Mechanism design via Risk-optimal Classifiers

The program:

1. Given training data $\{(x^k, f(x^k))\}_k$. Define feature map ψ.
2. Learn a classifier with discriminant function

$$H_w(x, y) = v_i(x_i, y) - w^\top \psi(x_{-i}, y)$$

   that minimizes expected loss.
3. Obtain mechanism (f, t_w), with payment rule

$$t_w(x_{-i}, y) = w^\top \psi(x_{-i}, y)$$

Use *structural support vector machines* to solve the multi-class classification problem (Joachims et al. 2009).

Structural SVMs

Training data $\{(x^k, y^k)\}_k$. *Feature map* ψ.

Training problem:

$$\min_{w,\xi\geq 0} \frac{1}{2} w^\top w + \frac{C}{\ell} \sum_k \xi^k \qquad \text{(QP)}$$

$$\text{s.t. } w^\top \psi(x^k, y^k) + \xi^k \geq \max_y w^\top \psi(x^k, y), \quad \forall k,$$

where ξ^k is a *slack variable*, and indicates a discriminant loss on example k. Impose 'admissible' structure on ψ.

The QP minimizes the regularized, empirical discriminant loss.

Structural SVMs

Training data $\{(x^k, y^k)\}_k$. *Feature map* ψ.

Training problem:

$$\min_{w,\xi\geq 0} \frac{1}{2} w^\top w + \frac{C}{\ell} \sum_k \xi^k \qquad \text{(QP)}$$

$$\text{s.t. } w^\top \psi(x^k, y^k) + \xi^k \geq \max_y w^\top \psi(x^k, y), \quad \forall k,$$

where ξ^k is a *slack variable*, and indicates a discriminant loss on example k. Impose 'admissible' structure on ψ.

The QP minimizes the regularized, empirical discriminant loss.

Structural SVMs

Training data $\{(x^k, y^k)\}_k$. *Feature map* ψ.

Training problem:

$$\min_{w,\xi\geq 0} \frac{1}{2} w^\top w + \frac{C}{\ell} \sum_k \xi^k \tag{QP}$$
$$\text{s.t. } w^\top \psi(x^k, y^k) + \xi^k \geq \max_y w^\top \psi(x^k, y), \quad \forall k,$$

where ξ^k is a *slack variable*, and indicates a discriminant loss on example k. Impose 'admissible' structure on ψ.

The QP minimizes the regularized, empirical discriminant loss.

Solving the Training Problem

1. Allow large attribute vector via *kernel trick*. Expand *attributes* q into features:

$$\psi(x_{-i}, y) = \phi(q) \in \mathbb{R}^m,$$

   Features ψ only appear in dual via inner product:

$$\langle(\phi(q), \phi(q')\rangle = K(q, q'),$$

   where K is the *kernel*.

2. Handle large outcome space Y via efficient separation (following Taskar et al., 2004).

Solving the Training Problem

1. Allow large attribute vector via *kernel trick*. Expand *attributes* q into features:

$$\psi(x_{-i}, y) = \phi(q) \in \mathbb{R}^m,$$

   Features ψ only appear in dual via inner product:

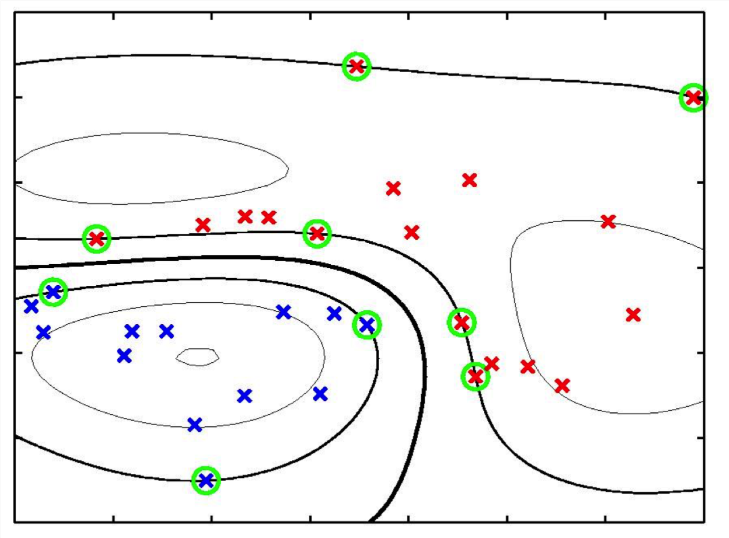
$$\langle(\phi(q), \phi(q')\rangle = K(q, q'),$$

   where K is the *kernel*.

2. Handle large outcome space Y via efficient separation (following Taskar et al., 2004).

Support Vectors in Original Attribute Space

Bishop (2007)



Empirical Results

Two problems that cannot be solved via classical methods

- *Multi-minded combinatorial auction:*
    - Auction multiple landing times at Schiphol
    - Each agent interested in multiple packages of landing slots
    - Vary degree of *complementarity* between items
    - f is a greedy allocation algorithm
- *Fair assignment problem:*
    - Each agent can receive at most one landing time
    - f maximizes *egalitarian welfare* (lex-max-min)

Benchmark: VCG-based rules (not IC!)

Empirical Results

Two problems that cannot be solved via classical methods

- *Multi-minded combinatorial auction*:
  - Auction multiple landing times at Schiphol
  - Each agent interested in multiple packages of landing slots
  - Vary degree of *complementarity* between items
  - f is a greedy allocation algorithm
- *Fair assignment problem*:
  - Each agent can receive at most one landing time
  - f maximizes *egalitarian welfare* (lex-max-min)

Benchmark: VCG-based rules (not IC!)

Empirical Results

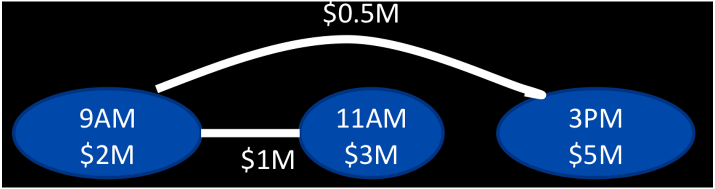
Two problems that cannot be solved via classical methods

- *Multi-minded combinatorial auction*:
  - Auction multiple landing times at Schiphol
  - Each agent interested in multiple packages of landing slots
  - Vary degree of *complementarity* between items
  - f is a greedy allocation algorithm
- *Fair assignment problem*:
  - Each agent can receive at most one landing time
  - f maximizes *egalitarian welfare* (lex-max-min)

Benchmark: VCG-based rules (not IC!)

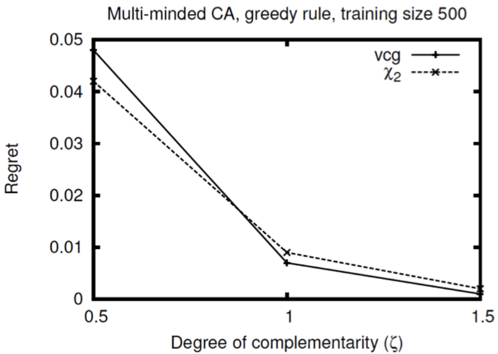
Succinct Valuation Representation for CA

- Need efficient separation problem
- Adopt *graphical valuations* (Conitzer & Sandholm 2005, Abraham et al. 2012)
  - Welfare-maximization NP-hard
  - Can solve separation problem (in dual QP)



Results I: Multi-Minded CAs

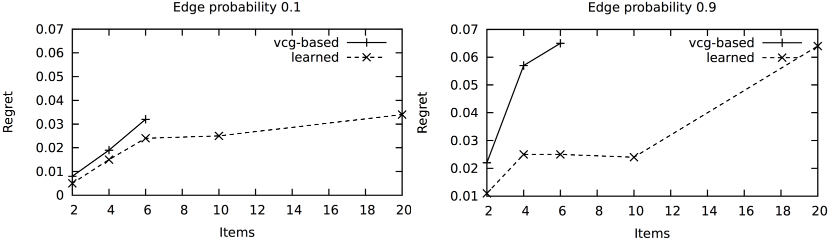
5 agents, 5 items. Without succinct valuations.



- *Performance comparable to that of VCG-based rule.*

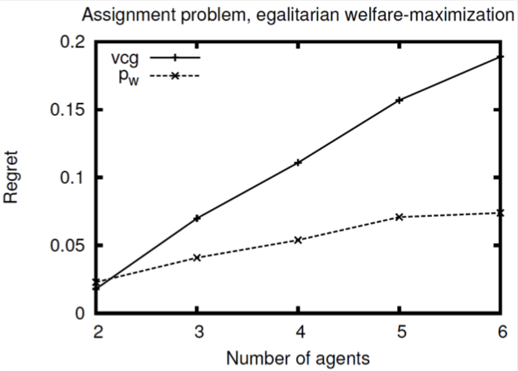
Results I: Multi-Minded CAs

Larger instances (6 agents, 2–20 items.) Succinct valuations.



- Higher edge density, higher complementarity.
- *Performance dominates that of VCG-based rule.*

Results II: Fair Assignment Problem



- *Performance dominates that of VCG-based rule.*

Conclusions

- We use *statistical learning* to find a payment rule that makes an outcome rule approx-IC
- Connect *discriminant-based classifiers* and incentive-compatible payment rules.
  - Roughly: look for a discriminant for agent i that is linear in x_i and non-linear in x_{-i}
- Circumvents analytical bottleneck, opens up *empirical approach* where an efficient outcome rule is matched (automatically) with a suitable payment rule.

Thank you

Reference: Payment Rules through Discriminant-Based Classifiers, P. Dütting, F. A. Fischer, P. Jirapinyo, J. K. Lai, B. Lubin, and D. C. Parkes, *ACM Transactions on Economics and Computation* **3**(1), 2014