Payment Rules through Discriminant-Based Classifiers

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

Manuscript by Nicolas Küchler

May 12, 2017

1 Introduction

Payment Rules through Discriminant-Based Classifiers is a paper from 2012 where Paul Dütting, Felix Fischer, Pichayut Jirapinyo, John K. Lai, Benjamin Lubin and David C. Parkes present a new method for constructing payment rules in *Mechanism Design*.
In *Mechanism Design* we are looking at situations where agents hold private informations about their preferences over different outcomes. Agents submit reports and the mechanism chooses an outcome and a payment.

1.1 Classical Approach for Mechanism Design

The classical approach is to start by imposing an incentive compatibility IC constraint (e.g. DSIC: dominant-strategy IC or BNIC: Bayesian-Nash IC) ensuring that agents report their preferences in equilibrium truthfully. Subject to that constraint, an outcome- and a payment rule are chosen. The classical approach however, brings some challenges:

Analytical Complexity It is difficult to find optimal mechanisms in *multi-dimensional domains*, where agents private information is described through more than a single number.

Exclusion of Mechanisms Incentive compatibility as a hard constraint might preclude mechanisms with useful economic properties.

Computational Complexity Payment rules can become computational intractable.

1.2 New Approach for Mechanism Design

In the new approach presented in the paper, we start from a given outcome rule g (e.g. optimal outcome rule) and a distribution over agent type profiles D. Then we use machine learning ML to identify a payment rule p that has good incentive properties relative to this outcome rule g. Instead of insisting on a hard incentive compatibility constraint, we minimize a metric called expected ex-post regret. This measures an agents regret reporting truthfully rather than any other false report that could have led to a personally preferred outcome.

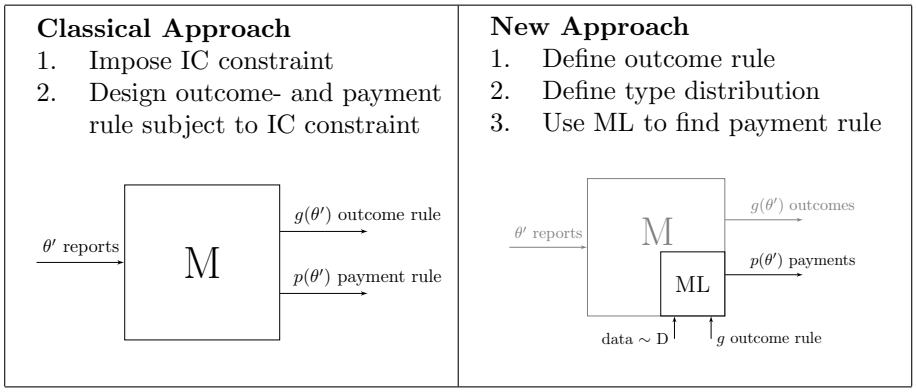

Figure 1: Comparison of the two approaches for *Mechanism Design*.

1.3 Structure

After introducing the new approach, the authors show the connection between finding a payment rule in *Mechanism Design* and the *Multi-Class Classification* problem of *Machine Learning*. Later they present a framework using *Structural Support Vector Machines* and apply it in experiments to two different domains.

2 Preliminaries

2.1 Ex-Post Regret

The *ex-post regret* an agent has for truthfully reporting in a given instance is the amount by which its utility could be increased through a misreport. Formally, the *ex-post regret* of agent i, given true type $\theta_i \in \Theta_i$ and the other agents reported types $\theta'_{-i} \in \Theta_{-i}$ is:

$$rgt_i(\theta_i, \theta'_{-i}) = \max_{\theta'_i \in \Theta_i} \underbrace{u_i((\theta'_i, \theta'_{-i}), \theta_i)}_{\text{utility misreport}} - \underbrace{u_i((\theta_i, \theta'_{-i}), \theta_i)}_{\text{utility truthful report}}$$

When a mechanism has zero *ex-post regret* for all inputs (θ_i, θ_{-i}), then it is *strategyproof*. Apart from that, *ex-post regret* does not have any further known

direct implications for equilibrium properties. The support for *expected ex-post regret* rather comes from a simple model, where agents face a certain cost for strategic behaviour. If this cost is higher than the expected gain, then the agents are assumed to behave truthfully.

2.2 Ex-Post Violation of IR

Another concept that is useful to understand some of the material covered later is the *ex-post violation of individual rationality*. It occurs, when an agent ends up with negative utility when reporting his true type. This means that he would have preferred not taking part in the mechanism at all. Ending up with a non-negative utility simply leads to a violation of zero. Formally, the *ex-post violation of individual rationality* of agent i, given true type $\theta_i \in \Theta_i$ and the other agents reported types $\theta'_{-i} \in \Theta_{-i}$ is:

$$irv_i(\theta_i, \theta'_{-i}) = |min(u_i((\theta_i, \theta'_{-i}), \theta_i), 0)|$$

3 Payment Rules from Multi-Class Classifiers

To show how finding a payment rule with good incentive properties is connected with the multi-class classification problem is really the essence of the paper but also not that straightforward to understand. Before building the connection, we start by looking at characteristics of a strategyproof mechanism and introduce the concept of a multi-class classifier.

3.1 Truthful Mechanisms

It is known that a mechanism is truthful if and only if two properties hold:

Agent-Independent Price Payment p_i is independent of agent i's reported type θ_i:

$$p_i(\theta) = t_i(\theta_{-i}, g_i(\theta)) \qquad \forall i \in N$$

Agent-Optimizing Outcome The chosen outcome $g_i(\theta)$ simultaneously maximizes the utility of all agents:

$$g_i(\theta) \in \arg\max_{o_i \in \Omega_i} v_i(\theta_i, o_i) - t_i(\theta_{-i}, o_i) \qquad \forall i \in N$$

3.2 Multi-Class Classifiers

A multi-class classifier takes an input x and finds the best fitting class y:

$$h_w(x) \in \arg\max_{y \in Y} f_w(x, y)$$

in our case the classifier is parametrized by a learned weight vector $w \in \mathbb{R}^m$. Learning the classifier h_w is nothing more than learning the function f_w.

Classifier in Mechanism Design It might sound really surprising that even though we are eventually looking for a payment rule p, we try to find a classifier for the outcome $o_i = g_i(\theta)$. So in our case the classifier receives as input a type profile θ and classifies an outcome o_i:

$$h_w(\theta) \in \arg\max_{o_i \in \Omega_i} f_w(\theta, o_i)$$

3.3 Connection

Assuming agent symmetry, we build the connection by focusing on a partial outcome rule g_1 and a classifier h_w for agent 1. Agent symmetry means informally that without loss of generality the problem is the same for every agent and it is thus enough to only consider it from the perspective of one agent.
Now let's look at the connection between truthful mechanisms and multi-class classifier.
Comparing classifier and *agent-optimizing outcome* property we see that they are really similar; both are looking for an optimal outcome o_i in a function of a type profile θ.
These similarities become even more striking when we consider only classifiers with a special discriminant-based structure of value - price, resembling utility of an agent:

$$\begin{array}{lllllll} \textit{Agent-Optimizing Property:} & g_1(\theta) \in \arg\max_{o_1 \in \Omega_1} & v_1(\theta_1, o_1) & - & t_1(\theta_{-1}, o_1) \\ \textit{Discriminant-Based Classifier:} & h_w(\theta) \in \arg\max_{o_1 \in \Omega_1} & w_1 v_1(\theta_1, o_1) & + & w_{-1}^T \psi(\theta_{-1}, o_1) \end{array}$$

Feature mapping function ψ maps θ_{-1} and o_1 into a single vector $\in \mathbb{R}^m$.
Due to this discriminant-based structure, learning the function f_w, becomes learning the payment rule p_w. We can see this by defining the associated price function:

$$t_w(\theta_{-1}, o_1) = -\frac{1}{w_1} w_{-1}^T \psi(\theta_{-1}, o_1)$$

And now with agent symmetry, we can obtain the complete payment rule $p_w(\theta)$ by plugging in this price function from each agent:

$$p_w(\theta) = (t_w(\theta_{-1}, g_1(\theta)), t_w(\theta_{-2}, g_2(\theta)), ..., t_w(\theta_{-n}, g_n(\theta)))$$

The *agent-independent price* property of a truthful mechanism can be directly inferred from the associated price function $t_w(\theta_{-1}, o_1)$, because it doesn't depend on an agents own report θ_1.

3.4 Truthful Mechanisms with a Perfect Classifier

To complete the connection between strategyproof mechanisms and multi-class classifiers we take a brief look at a Theorem from the paper:

Theorem 3.2 Let g be an agent symmetric outcome rule, h_w an admissible classifier, and p_w the payment rule corresponding to h_w. If h_w is a perfect classifier for the partial outcome rule g_1, then the mechanism (g, p_w) is strategyproof.

$$h_w \text{ is a } \textit{perfect classifier} \text{ for } g_1 \Rightarrow \text{mechanism } (g, p_w) \text{ is strategyproof}$$

A classifier is called a *perfect classifier* if it is able to classify all inputs correctly. In terms of our problem that means the trained classifier always selects the same outcome as the outcome rule.
Unfortunately identified mechanisms (g, p_w) are hardly ever strategyproof because a perfect classifier is rather rare. However, the authors show that the result extends gracefully to situations where no perfect classifier is found by relating *expected ex-post regret* of a mechanism to a measure of the generalization error of a classifier for g. Essentially this allows us to see, that minimizing the generalization error of the classifier (minimizing the number of miss-classifications), leads to minimizing the *expected ex-post regret* of the mechanism.

3.5 Example Single Item Auction

To gain some more intuition, let's consider a simple mechanism; the *Single Item Auction*. There are two possible outcomes $o_i \in \{0, 1\}$ for each bidder, either the item is allocated to agent i or it is not. Each agent reports his type θ_i, representing his value for the item, to the mechanism and the outcome rule allocates the item to the agent with the highest value.
The structure of the classifier $h_w(\theta) \in \arg\max_{o_1 \in \Omega_1} w_1 v_1(\theta_1, o_1) + w_{-1}^T \psi(\theta_{-1}, o_1)$ has to make sure that the outcome o_i matching the outcome rule $g(\theta)$ has to be optimal for every agent i. Concretely this means when agent 1 has a higher valuation for the item than the other agents, then the outcome $o_1 = 1$ (allocate the item to agent 1) needs to maximize the classifier. Otherwise outcome $o_1 = 0$ (not allocate the item to agent 1) needs to maximize the classifier.
In our example this can be achieved by setting:

$$w_1 = 1 \quad \text{and} \quad w_{-1}^T \psi(\theta_{-1}, o_1) = \begin{cases} -max(\theta_{-1}) & \text{if } o_1 = 1 \\ 0 & \text{if } o_1 = 0 \end{cases}$$

As we can see when plugging this into the associated price function $t_w(\theta_{-1}, o_1) = -\frac{1}{w_1} w_{-1}^T \psi(\theta_{-1}, o_1)$, this leads to the well known second price payment rule.

To clarify lets consider a small *Single Item Auction* instance with two agents. The optimal outcome is $o_1 = 0$ and $o_2 = 1$ both for the outcome rule of the mechanism and for each agent individually.

$$h_w(\theta) = \arg\max_{o_i} 1 * v_i(\theta_i, o_i) + \begin{cases} -max(\theta_{-1}) & \text{if } o_i = 1 \\ 0 & \text{if } o_i = 0 \end{cases}$$

	Outcome	**Value**	**Payment**	**Classifier**	**Optimal**
Agent 1	$o_1 = 1$	$v_1(\theta_1, 1) = 6$	8	$6 - 8 = -2$	
	$o_1 = 0$	$v_1(\theta_1, 0) = 0$	0	$0 - 0 = 0$	X
Agent 2	$o_1 = 1$	$v_2(\theta_2, 1) = 8$	6	$8 - 6 = 2$	X
	$o_1 = 0$	$v_2(\theta_2, 0) = 0$	0	$0 - 0 = 0$	

3.6 Consolidate Connection

To consolidate the connection between payment rules and multi-class classifiers lets summarize it in a sentence: *We train a classifier to learn the outcome rule that we already know but by doing so we are able to use a special structure within the classifier to derive a payment rule which provides good incentive properties.*

4 Structural SVM for Mechanism Design

The *Structural Support Vector Machine (SSVM)* is an extension of the standard *SVM* learning algorithm which allows to generalize the classifier to any kind of structured output and is thus an ideal fit for the problem of finding a discriminant-based classifier.

4.1 Training Data

In order to be able to learn the weight vector w of the classifier we need some training data to train the *SSVM*:

$$\text{training data: } \{(\theta^1, o_1^1), (\theta^2, o_1^2), ..., (\theta^l, o_1^l)\}$$

In the paper they generate this training data by first sampling from a defined type distribution and then apply the outcome rule to the sampled types. However, there are potentially other ways to get training data.
The authors used training sets of sizes 100, 300, and 500, and validation and test sets of size 1000 in their experiments. This should give a magnitude of how much training data is actually necessary.

4.2 Solve the Optimization Problem of the SSVM

Training a *SSVM* is nothing more than solving a constraint optimization problem. So after all the required information is provided, we can learn a weight vector w by plugging in the training data (θ_1^k, o_1^k) into the following optimization problem:

$$
\begin{aligned}
\underset{w,\xi}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{l}\sum_{k=1}^{l}\xi^k \\
\text{subject to} \quad & (w_1 v_1(\theta_1^k, o_1^k) + w_{-1}^T\psi'(\theta_{-1}^k, o_1^k)) - (w_1 v_1(\theta_1^k, o_1) + w_{-1}^T\psi'(\theta_{-1}^k, o_1)) \\
& \geq \mathcal{L}(o_1^k, o_1) - \xi^k \ , \forall k = 1, ..., l \ , \ o_1 \in \Omega_1 \\
& \xi^k \geq 0 \ , \forall k = 1, ..., l
\end{aligned}
$$

4.2.1 Intuition behind Optimization Problem

Constraints The first constraint basically says that the outcome o_1^k of the training sample has to be better than any other outcome o_1 by some margin $\mathcal{L}$. This makes sense since we defined the classifier such that it has to select the optimal outcome. The error term ξ^k makes sure that some minor missclassifications are possible but at a cost in the objective term. This leads to a soft-margin classifier which is less prone to overfitting to the training data.

Objective Term Minimizing $\|\mathbf{w}\|^2$ is in the objective term of every *SVM* and makes sure that the distance between the training samples and the decision boundaries between the different classes is maximized.

4.2.2 Valid Price Function

If $w_1 > 0$ then the learned weights w together with the feature map ψ define a price function $t_w(\theta_{-1}, o_1) = -\frac{1}{w_1} w_{-1}^T \psi(\theta_{-1}, o_1)$ that can be used to define payments $p_w(\theta)$, however the constraint $w_1 > 0$ is not enforced in the optimization problem directly for computational reasons. Instead hypothesis where the result of training is $w_1 \leq 0$ are simply discarded.

4.2.3 Flexible Payment Structure

We have seen function ψ the first time when we made the connection between payment rules and multi-class classifiers. Now in the optimization problem ψ appears again. As previously said, function ψ maps θ_{-1} and o_1 into a single vector. This mapping of input and output is something the mechanism designer has to specify and has an effect on the structure of the payments. We are not going into details here, however, I would like to mention that because of this mapping, the payment structure is flexible.

4.3 Problems of the computed Payment Rule

The computed payment rule from the optimization might have some problems but they can be removed or significantly weakened by slightly adjusting the result. The authors presented two problems and proposed possible fixes.

Computed payments could be negative One issue with the framework as stated is that the payments p_w computed from the solution to the optimization problem could be negative. This issue can be removed by normalizing the computed payments to an area where they are all positive.

Violation of individual rationality Another issue is that agents revealing their true type might end up with negative utility and regret taking part in the mechanism. As shown in the beginning, this is called a violation of *individual rationality*. The paper presents three ideas that can all be applied together to combat this issue:

- introduce payment offsets
- adjust the loss function $\mathcal{L}$
- introduce deallocation when such an IR violation occurs

Unfortunately none of them is able to completely solve the problem.

5 Experimental Evaluation

As mentioned in the beginning, the authors conducted a series of experiments over different domains and with various configurations but we will focus mainly on the most significant results in the domain of *multi-minded Combinatorial Auctions* with an optimal outcome rule.

5.1 Multi-Minded Combinatorial Auctions

A *multi-minded CA* like a normal *CA* allocates a set of r items $\{A, B, ...\}$ among n agents $\{1, ..., n\}$ that are allowed to express their valuation for each possible bundle separately. However, in a multi-minded CA, each agent is interested in at most b bundles for some constant b.
In *CA* type θ_i is a vector that specifies agent i's value for each possible bundle. ($\rightarrow \theta_i = (v_i(\emptyset), v_i(A), v_i(B), v_i(C), v_i(AB), v_i(AC), v_i(BC), v_i(ABC))$)
Types of all agents combined give a type profile $\theta = (\theta_1, \theta_2, ..., \theta_n)$. Outcome o_1 specifies in *CA* the allocation for agent 1. ($\rightarrow o_1 = 101 \triangleq$ agent 1 receives item A and C)
In order to study the effects of correlation and complementarity between items, the authors generated training samples (θ^k, o_1^k) using a type distribution that allowed them to vary these parameters.

5.2 Performance of the framework

In general we can say that the framework performed pretty well and the most significant findings are listed below.

Accuracy of the trained classifier in predicting the outcome is negatively correlated with regret and violation of individual rationality (Fig. 2).

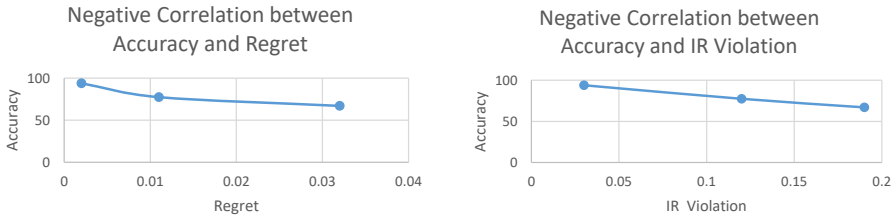

Figure 2: Correlation between accuracy and regret / ir violation

Degree of complementarity has a major effect on the results. Regret is higher for low complementarity between the items and it generally leads to less predictable allocations (Fig. 3).

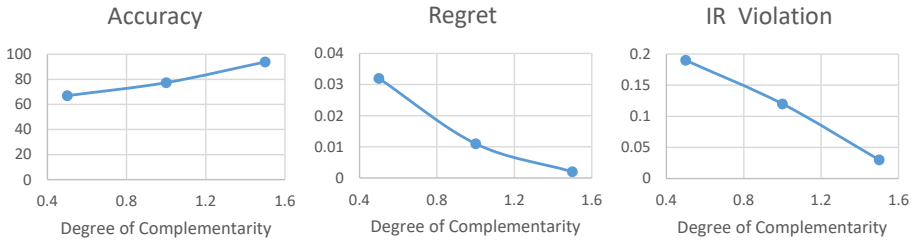

Figure 3: Effect of degree of complementarity on accuracy, regret and ir violation

Choice of the outcome rule in combination with the choice of an attribute mapping function also leads to significant differences in the outcome (Fig. 4). Attribute mapping function χ maps input and output into a single vector.

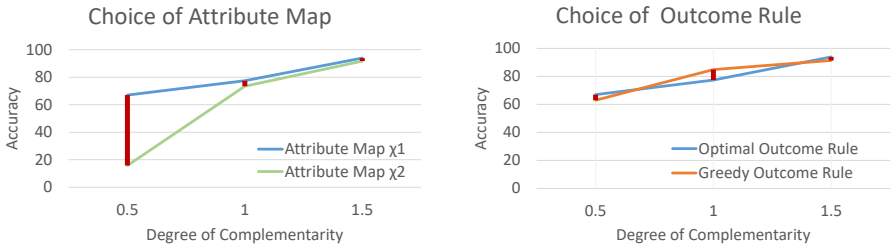

Figure 4: Effect of choosing a different outcome rule or attribute map

Increase the size of the training set leads to overall better results. In the experiments training sets with the sizes 100, 300 and 500 were used.

IR Fixes payment offset, adjusting loss function and introducing deallocation decrease *ir-violation* but unfortunately regret tends to move in the opposite direction.

6 Conclusion

In the introduction we were looking at three challenges that the classical approach brings: computational complexity, exclusion of mechanisms and analytical complexity. Now we are going to reflect back on if and how the new approach overcomes them.

Analytical Complexity It is hard to tell in general if this complexity is really reduced with the new approach. Because it is still not a framework that a mechanism designer can plug into any situation without additional effort. As an example, the mechanism designer has to find a good attribute mapping function and as we've seen this choice has a significant effect on the outcome.

Exclusion of Mechanisms Mechanisms with useful economic properties that were precluded because of the hard IC constraint are possible now with the new approach because a hard IC constraint doesn't exist any more.

Computational Complexity The computational cost associated with the new approach occurs offline during training. The learned payment rules have a succinct description and can be evaluated quickly in a deployed mechanism.

To summarize the paper, the authors have introduced a new paradigm for computational mechanism design, in which statistical machine learning is adopted to design payment rules for given outcome rules, and have shown encouraging experimental results. However, there are still quite a few directions of interest that have to be investigated in the future.