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

April 21, 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- or outcome 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 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.

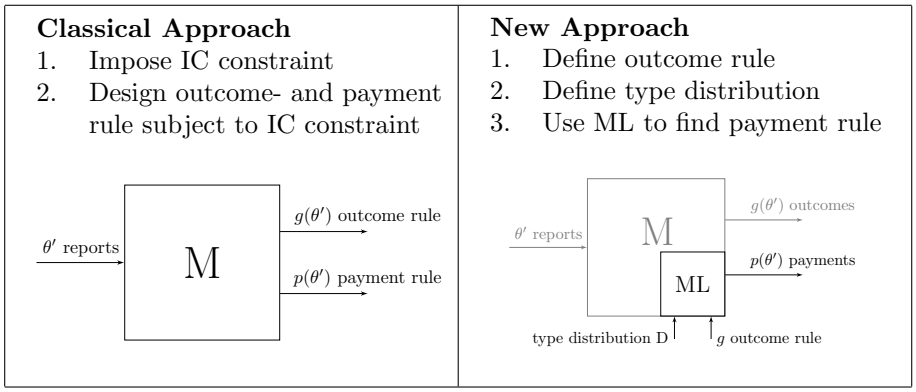


Figure 1: Comparison of the two approaches for *Mechanism Design*.

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 rather comes from a simple model, where agents face a certain cost for strategic behaviour. If this cost is higher than the *ex-post regret*, 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

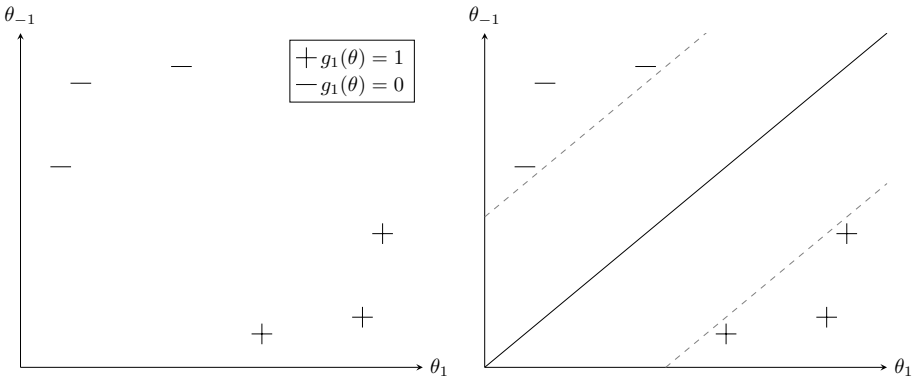
Now let's look at how the problem of multi-class-classification can be used to find a payment rule. For that we consider a simple mechanism; the *Single Item Auction*. There are two possible outcomes $o_i \in \{0, 1\}$ for each bidder, either the item is allocated to agent i or it is not. Each agent 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.

3.1 Classification Problem

Assuming agent symmetry, we focus on a partial outcome rule g_1 and train a classifier to mimic the outcome rule for agent 1. Agent symmetry means informally that the problem is the same for every agent and it is enough to only consider it from the perspective of one agent.
To be able to train a classifier we need a finite set of training data. This can be generated by drawing types θ^k from the type distribution D and applying the outcome rule g_1.

$$\text{training data: } \{(\theta^1, g_1(\theta^1)), ..., (\theta^l, g_1(\theta^l))\}$$

In figure 2a we see how such a training set with six samples could look like. Samples where type θ_1 has a high value are labeled with outcome $o_1 = 1$ and samples where the other agents type is higher valued are labeled with outcome $o_1 = 0$.
A *Support Vector Machine SVM* can be used to learn a classifier h that separates the two outcomes by a decision boundary. There are infinitely many possible decision boundaries with the given data but as we see in figure 2b a *SVM* chooses the one that informally speaking: *"maximizes the width of the street"*.



(a) six samples of θ with their outcome (b) decision boundary separating samples

Figure 2: SVM Classification Problem

3.2 Discriminant-Based Classifiers

The key idea is that by imposing a special structure on the classifier, it is possible to derive a payment rule for the mechanism that has good incentive properties. So we define a classifier h_w to be admissible if and only if it is defined in terms of a discriminant function where w is a weight vector that is learned and ψ is a feature mapping function:

$$h_w(\theta) = \arg\max_{o_1} \underbrace{w_1 v_1(\theta_1, o_1)}_{\text{correlates to value}} + \underbrace{w_{-1}^T \psi(\theta_{-1}, o_1)}_{\text{correlates to - price}}$$

This basically means for agent 1 the outcome o_1 is choosen that maximizes his utility. (e.g. if the price is higher than his value, then agent 1 would rather not be allocated and therefore $o_1 = 0$ is ideal)
We can see that the first term only depends on the type of agent 1 and increases in its valuation for outcome o_1, while the remaining terms ignore θ_1 entirely. This restriction allows us to directly infer *agent-independent prices* from a trained classifier which is known to be a property a *strategyproof* mechanism has to satisfy.
From this classifier h_w we can define the associated price function for agent 1:

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

and because of agent symmetry, we can obtain the complete payment rule $p_w(\theta)$ by plugging in this price-function from each agent:

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

3.3 The Classifier in the Single Item Auction

In our example of the *Single Item Auction* the structure of the classifier 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, 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	$0 - 0 = 0$	

3.4 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.
In the following we look at the process of building a mechanism with the new approach step by step. To see how the framework can be applied, we are only considering the domain of *multi-minded Combinatorial Auctions (CA)*
A *multi-minded CA* like a normal *CA* allocates a set of r items $\{A, B, ...\}$ among agents $\{1, ..., n\}$ that can express their valuation for bundles. However, in a multi-minded CA, each agent is interested in at most b bundles for some constant b. To keep it simple we are considering a setup with three agents and three items:

$$\textbf{CA Setup:} \quad agents = \{1, 2, 3\} \quad \& \quad items = \{A, B, C\}$$
$$\rightarrow \quad bundles = \{A, B, C, AB, AC, BC, ABC\}$$

4.1 Provide Outcome Rule and Type Distribution

First a type distribution has to be defined which represents agents preferences in the domain. In our example setup, the chosen type distribution provides parameters to control the correlation and complementarity between items.
After that the design objective of the mechanism (e.g. social welfare, revenue or some other notion of fairness) is selected and an outcome rule that achieves this objective is constructed. For the *multi-minded CA* we use the optimal outcome rule, which selects a feasible allocation with maximum total value and thus maximizes social welfare.

4.2 Generate Training Data

As a next step we need some data to train the *SSVM*. We generate training data by first sampling from the previously defined type distribution and then apply the outcome rule to the sampled types.
In our *multi-minded CA* with 3 items and 3 agents we draw 100 type profiles θ^k composed of the type of each agent θ_i^k. The type θ_i^k is a vector that specifies the value of an agent i for each possible bundle. And then the partial outcome rule g_1 is applied to the type profile to obtain the outcome for agent 1 o_1^k. The k-th type profile and outcome build together a training sample (θ^k, o_1^k).

Sample (θ^k, o_1^k)	**A** i	**Type** θ_i^k	**Outcome** o_1^k
$k = 1$ $\rightarrow (\theta^1, o_1^1)$	1	$\theta_1^1 = (0,4,4,4,4,4,4,4)$	$g_1(\theta^1) = o_1^1 = 010$
	2	$\theta_2^1 = (0,2,2,2,6,6,6,6)$	
	3	$\theta_3^1 = (0,2,2,2,4,4,4,6)$	
.			
.			
$k = 100$ $\rightarrow (\theta^{100}, o_1^{100})$	1	$\theta_1^{100} = (0,2,2,2,4,4,4,6)$	$g_1(\theta^{100}) = o_1^{100} = 000$
	2	$\theta_2^{100} = (0,2,2,2,6,6,6,6)$	
	3	$\theta_3^{100} = (0,4,4,4,4,4,4,4)$	

$\theta_i^k = (v_i(\emptyset), v_i(A), v_i(B), v_i(C), v_i(AB), v_i(AC), v_i(BC), v_i(ABC))$

4.3 Define Required Information for SSVM

A few parameters and configurations need to be defined in order to use a *Structural Support Vector Machine*. We will settle for a brief overview just to get an intuition for what a mechanism designer has to provide.

Attribute map χ is a function that generates an attribute vector that combines input and output data into a single object.
In our example of the CA the attribute map needs to combine agents types and the outcome into a single vector. There are multiple possibilities explored by the authors but we focus on one version where we stack vectors $\theta_i \setminus o_1$, obtained from θ_i by setting the entries for all bundles that intersect with o_1 to 0. This captures the fact that agent i cannot be allocated any of the bundles that intersect with o_1 if o_1 is allocated to agent 1.
Here in the example we see that each bundle that contains the item B which is part of the outcome o_1, has a value of 0.

$$\chi(\theta_{-1}, o_1) = \begin{bmatrix} \theta_2 \setminus o_1 \\ \theta_3 \setminus o_1 \end{bmatrix} \qquad \rightarrow$$

$$\chi(\theta_{-1}, 010) = \begin{bmatrix} 0 & 2 & 0 & 2 & 0 & 6 & 0 & 0 & 0 & 2 & 0 & 2 & 0 & 4 & 0 & 0 \end{bmatrix}^T$$

Kernel Function The computational efficiency of a *Support Vector Machine* comes from a mathematical trick in the dual formulation of the optimization problem where instead of computing the feature map ψ directly, the feature map only appears in a special form that can be replaced with a kernel function. There are a lot of different possible kernels but as an example we take a brief look at a *polynomial kernel*:

$$K_{polyd}(z, z') = \langle z, z' \rangle^d \ , \ d \in \mathbb{N}^+ \qquad \rightarrow \qquad \langle \chi(\theta_{-1}, 010), \chi(\theta_{-1}, 111) \rangle^d = 0$$

Parameter In addition some training parameters such as the regularization parameter C, and a kernel parameter d have to be defined.

4.4 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 generated training data (θ_1^k, o_1^k) into the following optimization problem:

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

4.4.1 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 *"width of the street"* is maximized.

4.4.2 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.

4.5 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.

4.6 Mechanism in Action

When a valid payment function was found and adjusted in off-line training, then it is ready to be used in the mechanism. Whenever agents report their types, they can be plugged into the learned payment function and the payments can be computed very efficiently.
So in our example of the CA, agents report their valuation for the different bundles, the winner determination problem is solved and the learned payment function applied to calculate the payments.

5 Experimental Evaluation

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*.

5.1 Metrics

We begin by briefly looking at the performance metrics that were used.

Classification Accuracy measures the accuracy of the trained classifier in predicting the outcome. It is the fraction of where predicted outcome matches the actual outcome:

$$\text{accuracy} = 100 * \frac{\sum_{k=1}^{l} \sum_{i=1}^{n} I(h_w(\theta_i, \theta_{-i}) = o_i^k)}{nl}$$

Ex post regret sums over the ex post regret experienced by all agents:

$$\text{regret} = \frac{\sum_{k=1}^{l} \sum_{i=1}^{n} rgt_i(\theta_i^k, \theta_{-i}^k)}{nl}$$

Individual rationality violation measures the fraction of IR violations over all agents:

$$\text{ir-violation} = \frac{\sum_{k=1}^{l} \sum_{i=1}^{n} I(irv_i(\theta_i, \theta_{-i}) > 0)}{nl}$$

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 is negatively correlated with regret and 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.

Choice of the outcome rule in combination with the choice of an attribute mapping function also leads to significant differences in the outcome.

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

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.