1

Machine Learning for Matching Markets

Zhe Feng[a], David C. Parkes[a] and Sai Srivatsa Ravindranath[a]

[a] Paulson School of Engineering and Applied Sciences, Harvard University, 150 Western Ave, Boston, MA 02134, USA. {zhe_feng,parkes,saisr}@g.harvard.edu

Edited by

1.1 Introduction

In this chapter, we demonstrate the use of machine learning for the automated design of matching markets. This extends the reach of optimal design to problems that are challenging to solve analytically and provides new directions for economic theory, identifying gaps in current understanding.

This is a data-driven approach and assumes access to samples of agent values or preferences and makes use of differentiable representations of the rules of matching markets in enabling gradient-based optimization. We refer to this research agenda as that of *differentiable economics*. The framework involves the following four steps:

1. Design an *artificial neural network architecture* that provides a differentiable representation of a mapping from inputs such as preference reports to outcomes such as a distribution on matchings.
2. Formulate a *loss function* and define other quantities of interest, for example the degree to which incentive-compatibility is violated.
3. Adopt a suitable *training procedure* to minimize expected loss while incorporating constraints such as incentive compatibility.
4. Evaluate performance against baselines and interpret the learned mechanisms.

We first provide a primer on artificial neural networks in Section 1.2. Section 1.3 applies the framework to one-sided matching, and in particular to the design of revenue-optimal multi-item auctions. Section 1.4 applies the framework to a two-sided matching, and in particular to understand the design frontier between stability and strategy-proofness. In Section 1.5, we outline a number of open problems and interesting future directions.

1.2 Artificial Neural Networks

An *artificial neural network* (ANN) is a non-linear model of computation inspired by the brain that is commonly used in machine learning. Each unit in an ANN consists of a non-linear *activation function* applied to a weighted sum of inputs. See Figure 1.1 left, where linear sum $\ell = w_0 + \sum_{i=1}^{J} w_i q_i$, for inputs $q_1, \ldots, q_J$, with weights $w_1, \ldots, w_J$ and *bias term* w_0. $\sigma : \mathbb{R} \to \mathbb{R}$ denotes the *activation function* and the output is $\sigma(\ell)$. Some commonly used activation functions are the *sigmoid*, *tanh*, and the *ReLU* or *LeakyReLU* activation functions (see Figure 1.2).

In a fully-connected, feed-forward ANN, several such units are stacked together and organized in layers, such that the outputs of some units become inputs to others (see Figure 1.1 right).

Let $x \in \mathbb{R}^d$ denote the *input* to an ANN and $y \in \mathbb{R}^n$ the *output*. $w^{(1)}, w^{(2)}, \ldots w^{(R)}$ denote the weights corresponding to each of $R \geq 1$ hidden layers, with J_r units in layer r. $w^{(R+1)}$ denotes the weights in the output layer. The weights $w =$

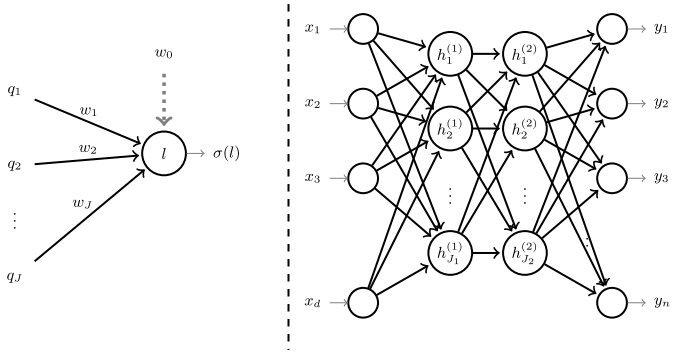


Figure 1.1 **Left**: A single unit in an artificial neural network. **Right**: A two-hidden layer, fully-connected, feed-forward artificial neural network.

$\{w^{(1)}, w^{(2)}, \ldots, w^{(R+1)}\}$ are the *parameters* to learn, with the network defining non-linear function $y = f^w(x)$.

Let $w_{ij}^{(r)}$ be the weight associated with the input from unit i in layer $r-1$ (or input x_i if $r = 1$) to unit j in layer r (or output unit y_j if $r = R+1$), and let $w_{0j}^{(r)}$ denote the associated bias term. The output $y \in \mathbb{R}^n$ is computed as follows:

$$h_j^{(1)} = \sigma\left(w_{0j}^{(1)} + \sum_{i=1}^{d} w_{ij}^{(1)} x_i\right), \qquad \forall j \in \{1, \ldots, J_1\} \tag{1.1}$$

$$h_j^{(r)} = \sigma\left(w_{0j}^{(r)} + \sum_{i=1}^{J_{r-1}} w_{ij}^{(r)} h_i^{(r-1)}\right), \qquad \forall j \in \{1, \ldots, J_r\},\ \forall r \in \{2, \ldots, R\} \tag{1.2}$$

$$y_j = \sigma\left(w_{0j}^{(R+1)} + \sum_{i=1}^{J_R} w_{ij}^{(R+1)} h_i^{(R)}\right), \quad \forall j \in \{1, \ldots, n\} \tag{1.3}$$

Learning is formulated as finding parameters w that minimizes a *loss function* with respect to a distribution F on inputs x. A common approach is *supervised learning*, where there is a *target function* f^* and the loss function is

$$\mathcal{L}(w) = \mathbb{E}_{x \sim F}\left[loss(f^w(x), f^*(x))\right], \tag{1.4}$$

where *loss* is a differentiable function that quantifies how well f^w approximates f^*. In application to economic design, the loss function will not come from supervision but directly captures the economic concept of interest.

ANNs are trained by updating parameters w to minimize loss on *training data* through a gradient descent procedure. A typical approach is to repeatedly sample a set of examples (a *mini-batch*) from the training data, with the parameters updated

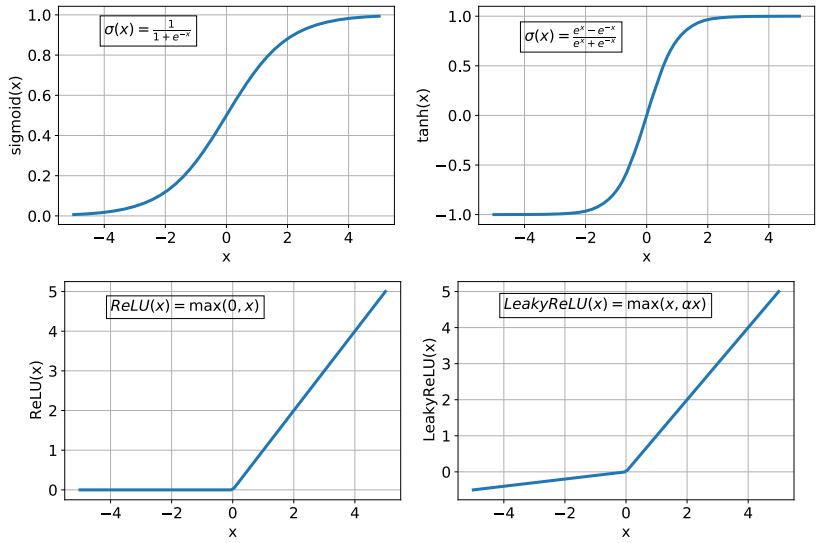


Figure 1.2 **Top-Left**: The *sigmoid* function maps a real number to the range $[0, 1]$. **Top-Right**: The *tanh* function is a scaled *sigmoid* function and is zero-centered and maps real numbers to the range $[-1, 1]$. **Bottom**: The *ReLU* and *LeakyReLU* are piece-wise linear functions that map reals to reals.

on mini-batch τ through a single iteration of stochastic gradient descent (SGD):

$$w_{ij}^r := w_{ij}^r - \alpha \times \nabla_{w_{ij}^r} \mathcal{L}^{(\tau)}(w), \quad \text{for each } i, j, \text{ and } r \in \{1, \ldots, R+1\}. \tag{1.5}$$

Here, $\alpha > 0$ is the *learning rate* and $\mathcal{L}^{(\tau)}(w)$ denotes the average loss on the examples in the τth mini-batch. Open-source software frameworks such as *PyTorch* and *TensorFlow* can be used to define and train ANNs.

1.3 Optimal Auction Design

We first illustrate the framework of differentiable economics on one-sided matching and the design of *revenue-optimal auctions.* This is a suitable problem to study because the optimal auction for the sale of two items is not fully understood from an analytical viewpoint.

An ANN is used to provide a differentiable representation for the allocation rule and payment rule. The aim is to learn the rules of an auction that minimize negated expected revenue while providing a close approximation to strategy-proofness.

1.3.1 Preliminaries

Let B denote a set of n *buyers* and G a set of m *items*. For *additive valuations,*et $v_{ij} \geq 0$ denote the value of buyer i for item j, so that buyer i's total value for a set of items S is $\sum_{j \in S} v_{ij}$. Let V denote the *valuation domain*, with buyer i's valuation represented through $v_i = (v_{i1}, \ldots, v_{im}) \in V$. Let $v = (v_1, \ldots, v_n)$ denote a *valuation profile*.

For a buyer with a *unit-demand valuation,* its value for a set S is $\max_{j \in S} v_{ij}$ and its value for the most preferred item. In effect, a unit-demand buyer is interested in buying at most one item. We work in a probabilistic setting, where the valuation v_i of buyer i is distributed i.i.d. according to distribution function F_V.

Let X denote the set of *feasible allocations*, i.e., the set of allocations in which each item is allocated at most once. For $x \in X$, let $x_i = (x_{i1}, \ldots, x_{im})$ denote buyer i's allocation, with $x_{ij} \in \{0, 1\}$ to indicate whether or not it is allocated item j.

An *auction* $A = (g, p)$ is defined by an *allocation rule* $g : V^n \to \triangle(X)$ and a *payment rule* $p : V^n \to \mathbb{R}^n$. $\triangle(X)$ is the probability simplex on feasible allocations, and g maps a reported valuation profile $\hat{v}$ to a possibly randomized allocation $g(\hat{v})$. The payment rule defines, for each buyer i, the expected payment $t_i = p(\hat{v})$.

We assume *quasilinear utility*, so that a buyer's utility is equal to the expected value minus payment. The following quantity plays an important role in learning approximately strategy-proof auctions.

Definition 1 (Regret) Buyer i's *regret* for truthful bidding on valuation profile v in auction $A = (g, p)$, and when all the other buyers are truthful, is

$$\text{regret}_i(v) = \max_{v_i' \in V} \big[(v_i(g(v_i', v_{-i})) - p_i(v_i', v_{-i})) - (v_i(g(v)) - p_i(v)) \big], \tag{1.6}$$

where $v_{-i} = (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$.

A buyer's regret is the maximum amount by which it can increase its utility relative to truthful reporting by reporting a non-truthful valuation. This connects with the notion of strategy-proofness (see Chapter **??**). An auction is *strategy-proof* if and only if every buyer has zero regret on every valuation profile.

1.3.2 Methodology

Step 1: Design an artificial neural network

We use a single neural network with two feed-forward components, namely the *allocation component* g^w and the *payment component* p^w (see Figure 1.3). Each component consist of multiple hidden layers ($h^{(r)}$ and $c^{(r)}$ in the figure) and an output layer. The components communicate through the value of the allocation.

This is the *RegretNet* architecture. The input is $n \times m$ reals, corresponding to bids. We write $b = (b_1, \ldots, b_n)$, where $b_i = (b_{i1}, \ldots, b_{im})$, and b_{ij} denotes the reported value (bid) of buyer i for item j. The network provides a differentiable

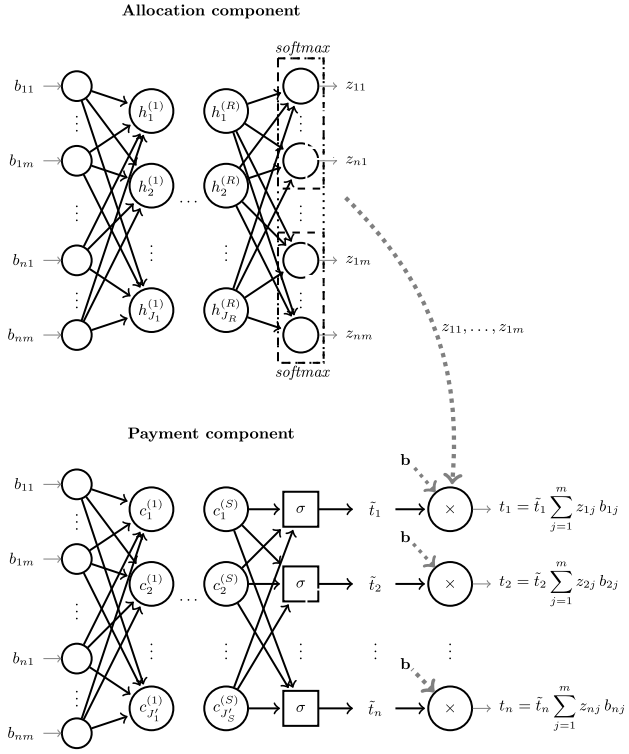


Figure 1.3 The allocation and payment component of the RegretNet architecture for a multi-item auction with n additive buyers and m items. The allocation component is a feed-forward network with $R(= 2)$ layers and softmax activation functions to determine the randomized allocation z. The payment component is a feed-forward network with $S(= 2)$ layers and sigmoid functions to determine the payment of each buyer as a fraction of the buyer's expected value (making use of the randomized allocation z).

representation of the auction rules. Let w_g and w_p denote the parameters in the allocation and payment component, respectively, with $w = (w_g, w_p) \in \mathbb{R}^d$, for d parameters.

The allocation component is function $g^w : V^n \to \triangle(X)$ and consists of two fully-connected hidden layers, with 100 units in each layer, each with *tanh* activations, and a fully-connected output layer. For each item $j \in G$, this outputs a vector $z_{1j}, \ldots, z_{nj}$ with $\sum_{i=1}^{n} z_{ij} \leq 1$. z_{ij} is the probability of allocating item j to buyer i. Each output unit uses a *softmax activation function*, where

$$z_{ij} = softmax_i(z'_{1j}, \ldots, z'_{nj}, z'_{n+1,j}) = \frac{e^{z'_{ij}}}{\sum_{k=1}^{n+1} e^{z'_{kj}}}, \tag{1.7}$$

where z'_{ij}, for each $i \in [n]$, is the result of taking a weighted sum of the outputs from the previous layer and $z'_{n+1,j}$ controls the probability that the item is not allocated ($e \approx 2.72$ is Euler's number).

Remark 2 The network architecture can represent allocation rules that tend to bundle items together, for example the value of z_{1j} may be high when z_{1k} is high, for items $j \neq k$ and buyer 1.

The payment component is function $p^w : V^n \to \mathbb{R}^n$ and consists of two fully-connected hidden layers, with 100 units in each layer each with *tanh* activations and a fully-connected output layer. For each buyer $i \in [n]$, there is an output unit that represents the expected payment $t_i \in \mathbb{R}$ for the bid inputs.

To ensure *individual rationality* (IR), with a buyer not charged more than its reported value, the network computes a *fractional payment*, $\tilde{t}_i \in [0, 1]$ for buyer i, which is the fraction of the buyer's reported value that it will be charged. This comes from a sigmoid function $\sigma(\ell_i)$ applied to the weighted sum from the previous layer. Given allocation z, the expected payment by buyer i is

$$t_i = \tilde{t}_i \times \sum_{j=1}^{m} z_{ij}\, b_{ij}. \tag{1.8}$$

Remark 3 The expected payment from RegretNet can be interpreted as a lottery on payments, charging buyer i the amount $\tilde{t}_i \times \sum_{j=1}^{m} x_{ij} b_{ij}$ for 0-1 allocation x. In this way, the buyer's payment is no greater than its bid value for the realized allocation.

For buyers with *unit-demand valuations*, we modify the allocation component to allocate at most one item to each buyer. The payment component is unchanged. The modified allocation component outputs two scores, $s \in \mathbb{R}^{(n+1)\times m}$ and $s' \in \mathbb{R}^{n\times(m+1)}$, and computes the row-wise softmax and column-wise softmax of each of s and s', respectively. The probability z_{ij} is given by the minimum of the corresponding normalized scores:

$$z_{ij} = \min\left\{ \frac{e^{s_{ij}}}{\sum_{k=1}^{n+1} e^{s_{kj}}}, \frac{e^{s'_{ij}}}{\sum_{k=1}^{m+1} e^{s'_{ik}}} \right\}. \tag{1.9}$$

Step 2: Formulate a loss function and quantify the violation of strategy-proofness

The loss function is the expected negated revenue and minimizing loss is equivalent to maximizing revenue. For training data $\mathcal{D} = \{v^{(1)}, \ldots, v^{(L)}\}$, consisting of L valuation profiles sampled i.i.d. from the valuation distribution, the empirical loss is

$$\mathcal{L}(w) \;=\; -\frac{1}{L} \sum_{\ell=1}^{L} \sum_{i=1}^{n} p_i^w(v^{(\ell)}). \tag{1.10}$$

Let $\text{regret}_i(v; w)$ denote the regret to buyer i at valuation profile v given an auction with parameters w. Given that regret is non-negative, the auction is strategy-proof up to zero measure events if and only if, for all buyers $i \in [n]$, we have

$$\mathbb{E}_{v \sim F^V \times \ldots \times F^V} \left[\text{regret}_i(v; w)\right] = 0. \tag{1.11}$$

We quantity the *violation of strategy-proofness* to buyer i as

$$rgt_i(w) \;=\; \frac{1}{L} \sum_{\ell=1}^{L} \text{regret}_i(v^{(\ell)}; w). \tag{1.12}$$

The training problem is:

$$\begin{aligned} &\min_{w} \; \mathcal{L}(w) \\ &\text{s.t.} \quad rgt_i(w) \;=\; 0, \quad \forall i \in [n]. \end{aligned} \tag{1.13}$$

Step 3: Adopt a training procedure

The training procedure uses gradient descent and *augmented Lagrangian optimization*. This solves a sequence of unconstrained optimization problems, for each of step $k \in 0, 1, \ldots$, where the constraints on regret are incorporated within the objective. The kth step seeks parameters $w^{(k)}$ to minimize

$$C(w; \lambda_{rgt}^{(k)}, \rho) \;=\; \mathcal{L}(w) \;+\; \sum_{i \in [n]} \lambda_{rgt,i}^{(k)} \times rgt_i(w) \;+\; \frac{\rho}{2} \times \sum_{i \in [n]} rgt_i(w)^2, \tag{1.14}$$

where $\lambda_{rgt,i}^{(k)} \in \mathbb{R}$ is the *Lagrangian multiplier* for buyer i. $C(w; \lambda_{rgt}^{(k)}, \rho)$ augments a Lagrangian function with a quadratic penalty term, with parameter $\rho > 0$. This modified objective penalizes revenue by a quantity that depends on the degree of violation of strategy-proofness.

We initialize $\lambda_{rgt,i}^{(0)} = 0$. Given solution $w^{(k)}$ in step k, the Lagrangian multipliers are updated according to rule

$$\lambda_{rgt,i}^{(k+1)} := \lambda_{rgt,i}^{(k)} \;+\; \rho \times rgt_i(w^{(k)}). \tag{1.15}$$

For each step k, the training procedure uses multiple SGD mini-batch iterations to approximately solve

$$w^{(k+1)} \in \arg\min_w \; C(w;\, \lambda_{rgt}^{(k)}, \rho). \tag{1.16}$$

The gradient of revenue with respect to w is straightforward to calculate. For regret, the gradient is complicated by the nested maximization (1.6). To handle this, we first find a *defeating valuation*, $\hat{v}_i^{(\ell)}$, for buyer i at valuation profile $v^{(\ell)}$ (or just $v_i^{(\ell)}$ if there is no such mis-report). This is a valuation that provides better utility than reporting truthfully. Given this, we approximate the gradient of regret

as the gradient of the difference in utility to the buyer at report $\hat{v}_i^{(\ell)}$ compared to its true report:

$$\nabla_w \Big[(v_i^{(\ell)}(g^w(\hat{v}_i^{(\ell)}, v_{-i}^{(\ell)})) - p_i^w(\hat{v}_i^{(\ell)}, v_{-i}^{(\ell)})) - (v_i^{(\ell)}(g^w(v^{(\ell)})) - p_i^w(v^{(\ell)})) \Big] . \quad (1.17)$$

For a given valuation profile $v^{(\ell)}$ we search for a defeating valuation for buyer i by following gradient ascent in input space, considering the buyer's utility with respect to its reported value (fixing the network parameters). This is in the style of adversarial machine learning. For each buyer, we use multiple random starting valuations and take the best mis-report as the defeating valuation.

Although the training problem is non-convex, we have found that Lagrangian optimization with SGD, together with gradient-ascent on inputs to find defeating valuations, can reliably learn auctions with near-optimal revenue and a very small violation of strategy-proofness.

1.3.3 Illustrative Experimental Results

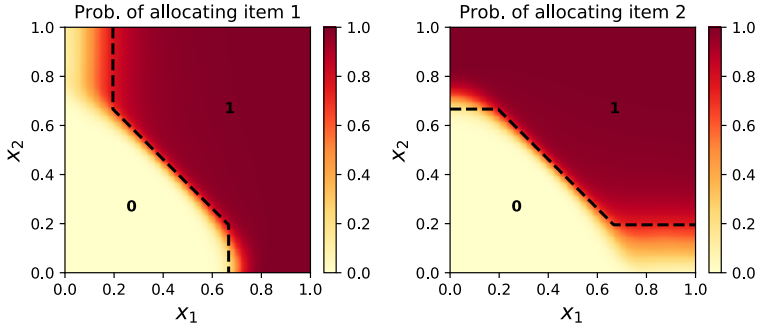
We first present results for single-buyer, two-item environments, for which there exist optimal designs from auction theory:

- 2 items, a single additive buyer, with item values $x_1, x_2 \sim U[0, 1]$, on item 1 and item 2, respectively. See Figure 1.4 (a).
- 2 items, a single additive buyer, with item 1 value $x_1 \sim U[4, 16]$ and item 2 value $x_2 \sim U[4, 7]$. See Figure 1.4 (b).
- 2 items, a single unit-demand buyer, with item values $x_1, x_2 \sim U[0, 1]$, on item 1 and item 2, respectively. See Figure 1.4 (c).
- 2 items, a single unit-demand buyer, with item values $x_1, x_2 \sim U[2, 3]$, on item 1 and item 2, respectively. See Figure 1.4 (d).

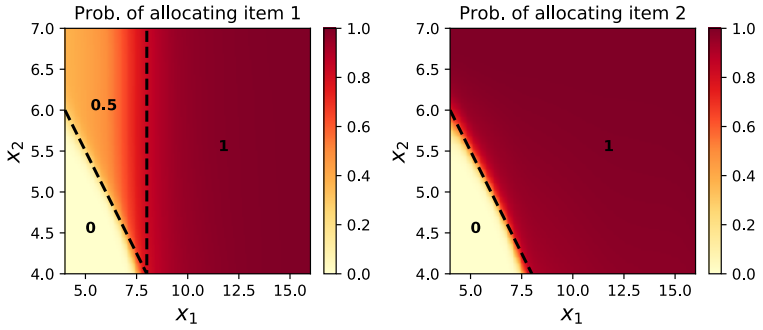
Table 1.1 summarizes the revenue and per-agent regret for the learned auctions, as evaluated on test data. For all four environments, the revenue is very close to the optimal revenue. For the additive $U[0, 1]$ environment the revenue in RegretNet is slightly higher than optimal, reflecting that it is not quite strategy-proof (while assuming truthful reports for evaluating revenue).

Figures 1.4 (a)–(d) compare the learned allocation rules with the optimal designs from economic theory. We super-impose on the density plots for RegretNet the optimal allocation rule, with different regions delineated by dashed lines and the number in a region giving the probability the item is allocated by the optimal rule. Not only is the revenue very close to optimal (Table 1.1), the allocation rules also capture the structure of the optimal designs.

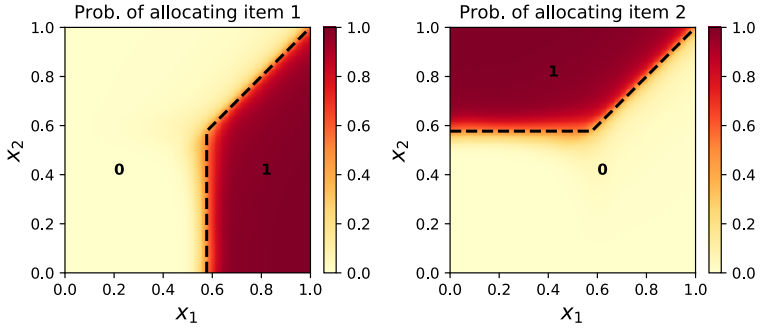
Figure 1.5 gives results for a setting with two items and two additive buyers where the item values are i.i.d. uniform on interval $[0, 1]$. This is complicated enough that there is no known theoretically-optimal design. As a baseline we compare



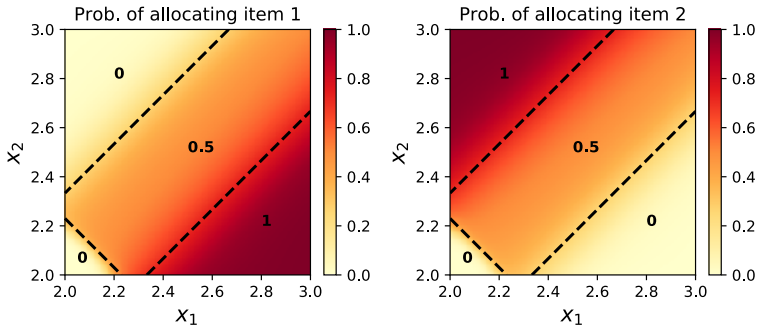
(a) Single buyer, additive valuation, item values $x_1, x_2 \sim U[0, 1]$



(b) Single buyer, additive valuation, item values $x_1 \sim U[4, 16]$, $x_2 \sim [4, 7]$



(c) Single buyer, unit-demand valuation, item values $x_1, x_2 \sim U[0, 1]$



(d) Single buyer, unit-demand valuation, item values $x_1, x_2 \sim U[2, 3]$

Figure 1.4 The allocation rule of a learned RegretNet auction for four different settings. We plot the probability of allocating item 1 (Left) and item 2 (Right), as a function of the buyer's value. The theoretically-optimal allocation rule is superimposed, with different allocation regions in the optimal rule delineated by dashed lines (the number in a region gives the probability the item is allocated in the optimal rule).

Economic environment	Optimal rev	RegretNet rev (norm)	RegretNet regret
2 item, 1 additive buyer, $x_1, x_2 \sim U[0,1]$	0.550	0.554 (100.7%)	< 0.001
2 item, 1 additive buyer, $x_1 \sim U[4,16]$, $x_2 \sim U[4,7]$	9.781	9.734 (99.5%)	< 0.001
2 item, 1 unit-demand buyer, $x_1, x_2 \sim U[0,1]$	0.384	0.384 (100.0%)	< 0.001
2 item, 1 unit-demand buyer, $x_1, x_2 \sim U[2,3]$	2.137	2.137 (100.0%)	< 0.001

Table 1.1 *Expected revenue and expected, per-agent regret from RegretNet in single-buyer auction settings, comparing with the theoretically-optimal revenue (and also giving the normalized revenue, as a fraction of the optimal revenue).*

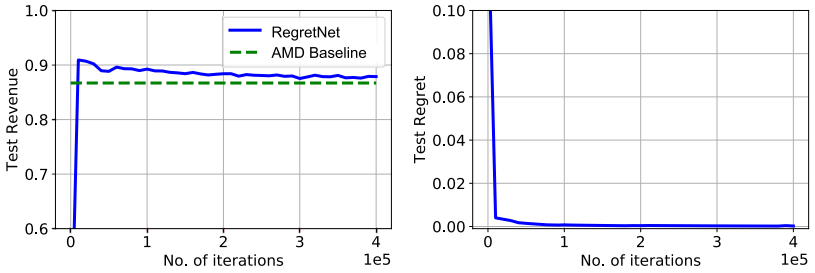


Figure 1.5 The test revenue and test regret from RegretNet as a function of training iterations (number of mini-batch updates) for an auction with two items and two additive buyers, with values $U[0,1]$ for each item. The baseline represents the performance of the previous best result from automated mechanism design.

with the previous best results from automated mechanism design (AM), which searches for the best auction in a parametrized family of strategy-proof auctions. The revenue falls during training, reflecting an improvement in strategy-proofness. The network learns an auction with essentially zero regret and expected revenue of 0.878, compared with revenue of 0.867 from the AMD baseline.

1.4 Two-Sided Matching

In this section, we turn to the automated design of two-sided matching markets. It is well known to be impossible to achieve both strategy-proofness and stability in two-sided matching **??**. And yet little is known about how to tradeoff between these two properties. Here, we illustrate the use of differentiable economics to explore the design frontier between strategy-proofness and stability and suggest new targets for economic theory.

1.4.1 Preliminaries

Let W denote a set of n *workers* and F denote a set of m *firms*. Each worker can be matched to at most one firm and each firm to at most one worker.

A *matching* μ is a set of (worker, firm) pairs, with each worker and firm participating in at most one match. Let $\mathcal{B}$ denote the set of all matchings. If a worker or firm remains unmatched, we will say that it is matched to $\perp$. If $(w, f) \in \mu$ then μ matches w to f, and we write $\mu(w) = f$ and $\mu(f) = w$. We write $(w, \perp) \in \mu$ (resp. $(\perp, f) \in \mu$) to denote that w (resp. f) is unmatched.

Each worker has a *strict preference order* $\succ_w$ over the set $\overline{F} = F \cup \{\perp\}$. Each firm has a *strict preference order* $\succ_f$ over the set $\overline{W} = W \cup \{\perp\}$. An agent prefers to be unmatched than matched to others ranked below $\perp$ (these are *unacceptable*, while others are *acceptable*). If worker w prefers firm f to f' then we represent this as $f \succ_w f'$, and similarly for firms. Let P denote the domain of preference profiles, with profile $\succ = (\succ_1, \ldots, \succ_n, \succ_{n+1}, \ldots, \succ_{n+m}) \in P$.

A pair (w, f) forms a *blocking pair* for matching μ if w and f prefer each other to their partners in μ (or $\perp$ in the case that either or both are unmatched). A matching μ is *stable* if and only if there are no blocking pairs. A matching μ satisfies *individual rationality* (IR) if it is not blocked by any single agent, i.e., no worker or firm finds its partner unacceptable.

Remark 4 Stability is not satisfied by an empty matching. For example, if a matching μ leaves a worker w and a firm f unmatched, where w finds f acceptable and f finds w acceptable, then (w, f) is a blocking pair to μ.

1.4.2 Randomized matchings

A *randomized matching mechanism* g takes a reported preference profile $\succ$ and maps this to a distribution $g(\succ) \in \triangle(\mathcal{B})$ on matchings. $\triangle(\mathcal{B})$ denotes the probability simplex on matchings. $r \in [0, 1]^{(n+1)\times(m+1)}$ denotes the *marginal probability* $r_{wf} \geq 0$ with which worker w is matched with firm f, for $w \in \overline{W}$ and firm $f \in \overline{F}$. We require $\sum_{f' \in \overline{F}} r_{wf'} = 1$ for all $w \in W$, and $\sum_{w' \in \overline{W}} r_{w'f} = 1$ for all $f \in F$.

Theorem 5 (Birkhoff–von Neumann) *Given a randomized matching r, there exists a distribution on matchings with marginal probabilities equal to r.*

We also write $g_{wf}(\succ)$ to denote the marginal probability of matching worker w (or $\perp$) and firm f (or $\perp$) at reported preference profile $\succ$. The following definition generalizes the concept of stability to randomized matchings.

Definition 6 (Ex ante justified envy) A randomized matching r causes *ex ante justified envy* if:

(1) some worker w prefers f over some (fractionally) matched firm f' (including

$f' = \perp$) and firm f prefers w over some (fractionally) matched worker w' (including $w' = \perp$) ("w has envy towards w'" and "f has envy towards f'"), or

(2) some worker w finds a (fractionally) matched $f' \in F$ unacceptable, i.e. $r_{wf'} > 0$ and $\perp \succ_w f'$, or some firm f finds a (fractionally) matched $w' \in W$ unacceptable, i.e. $r_{w'f} > 0$ and $\perp \succ_f w'$.

A randomized matching r is *ex ante stable* if and only if it does not cause any ex ante justified envy. Ex ante stability reduces to the standard concept of stability for a deterministic matching. Part (2) of the definition requires that a randomized matching r should satisfy IR; i.e., for any worker w with $\perp \succ_w f'$ for firm f' then $r_{wf'} = 0$, and any firm f with $\perp \succ_f w'$ for worker w' then $r_{w'f} = 0$.

To define strategy-proofness, say that $u_w : \overline{F} \to \mathbb{R}$ is a $\succ_w$-*utility* for worker w when $u_w(f) > u_w(f')$ if and only if $f \succ_w f'$, for all $f, f' \in \overline{F}$. We similarly define a $\succ_f$-*utility* for firm f. The following concept of ordinal strategy-proofness provides a strong version of incentive compatibility for randomized matching markets.

Definition 7 (Ordinal strategy-proofness) A randomized matching mechanism g satisfies *ordinal strategy-proofness* if and only if, for all agents $i \in W \cup F$, for any preference profile $\succ$, any $\succ_i$-utility u_i for agent i, and all reports $\succ_i'$, we have

$$\mathbb{E}_{\mu \sim g(\succ_i, \succ_{-i})} \left[u_i(\mu(i)) \right] \geq \mathbb{E}_{\mu \sim g(\succ_i', \succ_{-i})} \left[u_i(\mu(i)) \right]. \tag{1.18}$$

By this definition, no worker or firm can improve its expected utility by mis-reporting its preferences whatever the utility function consistent with its preferences. For a deterministic mechanism, ordinal strategy-proofness reduces to strategy-proofness and the requirement that no agent has an improving mis-report. In Section 1.4.4 we introduce a weaker notion of strategy-proofness that we use for learning making tradeoffs between strategy-proofness and stability.

1.4.3 Deferred Acceptance and RSD

Deferred-acceptance (DA) algorithms provide stable mechanisms and are strategy-proof for the proposing side of the market.

Theorem 8 *Worker- or firm-proposing DA is stable, but not ordinal strategy-proof.*

See Chapter **??**. We also define the following simple mechanism (see also *Random Priority* in Chapter **??**).

Definition 9 (Random serial dictatorship (RSD)) Sample a *priority order* π on the set $W \cup F$ uniformly at random, such that $\pi_1, \pi_2, \ldots, \pi_{m+n}$ is a permutation on $W \cup F$ in decreasing order of priority. Proceed as follows:

- Initialize matching μ to the empty matching.
- In round $k = 1, \ldots, m + n$:

– If participant $\pi_k \in W \cup F$ is not yet matched in μ, then add to matching μ the match between π_k and its most preferred, unmatched agent, or $(\pi_k, \perp)$ if all remaining possibilities are unacceptable.

Theorem 10 *RSD is ordinal strategy-proof, but not stable.*

Proof For ordinal strategy-proofness, observe that an agent's own report has no effect on the choices of higher-priority agents. Truthful reporting ensures that it obtains its most preferred match of those available when it is its turn. RSD is not stable because an agent (say f) may select as its match an agent on the other side of the market for whom f is unacceptable.

□

1.4.4 Methodology

Step 1: Design an artificial neural network

We use a neural network with parameters $\theta \in \mathbb{R}^d$ to represent a matching mechanism, $g^\theta : P \to \triangle(\mathcal{B})$. We use a fully-connected, feed-forward neural network with 4 hidden layers, 256 units in each layer, leaky ReLU activations, and a fully-connected output layer. See Figure 1.6.

We represent preference orders at the input by adopting the *equi-spaced utility function*, which is an evenly spaced valid utility. For preference profile $\succ$, the equi-spaced utility for worker $w \in W$ and firm $f \in F$ is denoted as $p_w^\succ = (p_{w1}^\succ, \ldots, p_{wm}^\succ)$ and $q_f^\succ = (q_{1f}^\succ, \ldots, q_{nf}^\succ)$ respectively. We also define $p_{w\perp}^\succ = 0$ and $q_{\perp f}^\succ = 0$. For example, we have:

- For a preference order $\succ$ with $w_1 : f_1, f_2, \perp, f_3$, we have $p_{w_1}^\succ = (\frac{2}{3}, \frac{1}{3}, -\frac{1}{3})$.
- For a preference order $\succ$ with $w_1 : f_1, f_2, \perp, f_3, f_4$, we have $p_{w_1}^\succ = (\frac{2}{4}, \frac{1}{4}, -\frac{1}{4}, -\frac{2}{4})$.
- For a preference order $\succ$ with $f_1 : w_2, w_1, w_3, \perp$, we have $q_{f_1}^\succ = (\frac{2}{3}, 1, \frac{1}{3})$.

In this way, the vector $(p_{11}^\succ, \ldots, p_{nm}^\succ, q_{11}^\succ, \ldots, q_{nm}^\succ)$ constitutes the input to the network ($2 \times n \times m$ numbers). The output of the network is a vector $r \in [0,1]^{n \times m}$ with $\sum_{j=1}^m r_{wj} \leq 1$ and $\sum_{i=1}^n r_{if} \leq 1$ for every every $w \in [n]$ and $f \in [m]$. This describes the marginal probabilities in a randomized matching for this input profile. The network first outputs two sets of scores $s \in \mathbb{R}^{(n+1)\times m}$ and $s' \in \mathbb{R}^{n \times (m+1)}$. We apply the *softplus function* (denoted by σ_+) element-wise to these scores, where $\sigma_+(x) = \ln(1 + e^x)$. To ensure IR, we first construct a Boolean mask variable β_{wf}, which is zero only when the match is unacceptable to one or both the worker and firm, i.e., when $\perp \succ_w f$ or $\perp \succ_f w$. We set $\beta_{n+1,f} = 1$ for $f \in F$ and $\beta_{w,m+1} = 1$ for $w \in W$. We multiply the scores s and s' element-wise with the corresponding Boolean mask variable to compute $\bar{s} \in \mathbb{R}_{\geq 0}^{(n+1)\times m}$ and $\bar{s}' \in \mathbb{R}_{\geq 0}^{n \times (m+1)}$.

For each $w \in \overline{W}$, we have $\bar{s}_{wf} = \beta_{wf} \ln(1 + e^{s_{wf}})$, for all $f \in F$. For each $f \in \overline{F}$, we have $\bar{s}'_{wf} = \beta_{wf} \ln(1 + e^{s'_{wf}})$, for all $w \in W$. We normalize $\bar{s}$ along the rows and

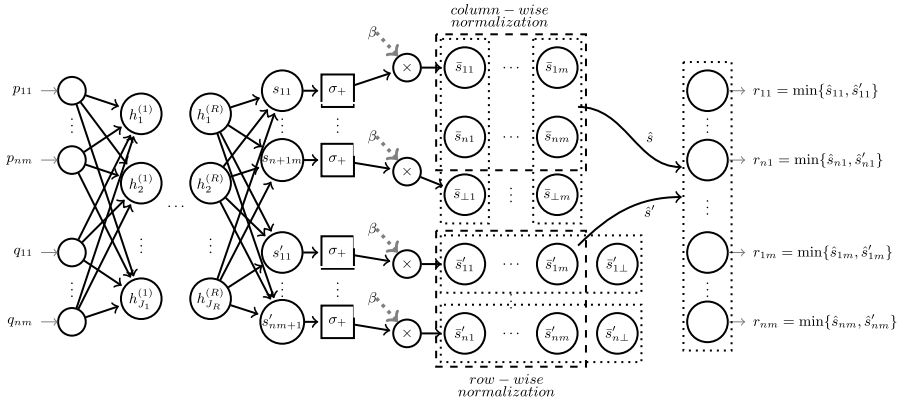


Figure 1.6 Matching network g for a set of n workers and m firms. Given inputs $p, q \in \mathbb{R}^{n\times m}$ the matching network is a feed-forward neural network with R hidden layers that uses softplus activation to generate non-negative scores and normalization to compute the marginal probabilities in the randomized matching. We additionally generate a Boolean mask matrix, β, and multiply it with the score matrix before normalization to ensure IR by making the probability of matches that are unacceptable zero.

$\bar{s}'$ along the columns to obtain *normalized scores*, $\hat{s}$ and $\hat{s}'$ respectively. The match probability r_{wf}, for worker $w \in W$ and firm $f \in F$, is computed as the minimum of the normalized scores:

$$r_{wf} = \min\left(\frac{\bar{s}_{wf}}{\sum_{f'\in\overline{F}} \bar{s}_{wf'}}, \frac{\bar{s}'_{wf}}{\sum_{w'\in\overline{W}} \bar{s}'_{w'f}}\right). \tag{1.19}$$

We have $r_{wf} = 0$ whenever $\beta_{wf} = 0$, and every matching in the support of the distribution will be IR. Based on this construction, the allocation matrix r is always weakly doubly stochastic (allowing for workers and firms to go unmatched) and can be decomposed to a convex combination of 0-1 weakly doubly stochastic matrices.

Step 2: Formulate a loss function and quantify the violation of strategy-proofness and stability

To train the neural network we formulate a loss function $\mathcal{L}$ on training data $D = \{\succ^{(1)}, \ldots, \succ^{(L)}\}$, with each preference profile sampled i.i.d. from a distribution on profiles. The loss function is designed to represent the trade-off between strategy-proofness and stability.

Recall that $g^\theta(\succ) \in [0,1]^{n\times m}$ denotes the randomized matching. We write $g^\theta_{w\perp}(\succ) = 1 - \sum_{f=1}^m g^\theta_{wf}(\succ)$ and $g^\theta_{\perp f}(\succ) = 1 - \sum_{w=1}^n g^\theta_{wf}(\succ)$ to denote the probability of worker w and firm f being unmatched, respectively. For worker w and firm f, we

define the *stability violation* at profile $\succ$ as

$$stv_{wf}(g^\theta,\succ)=\left(\sum_{w'=1}^{n} g^\theta_{w'f}(\succ)\cdot\max\{q^{\succ}_{wf}-q^{\succ}_{w'f},0\}+g^\theta_{\perp f}(\succ)\cdot\max\{q^{\succ}_{wf},0\}\right) \quad (1.20)$$
$$\times\left(\sum_{f'=1}^{m} g^\theta_{wf'}(\succ)\cdot\max\{p^{\succ}_{wf}-p^{\succ}_{wf'},0\}+g^\theta_{w\perp}(\succ)\cdot\max\{p^{\succ}_{wf},0\}\right).$$

This captures the first kind of ex ante justified envy in Definition 6, which is in regard to fractionally matched partners. We ignore the second kind of ex ante justified envy in defining the loss function because the learned mechanisms satisfy IR through the use of the Boolean mask matrix.

The overall *stability violation* of mechanism g^θ on profile $\succ$ is defined as

$$stv(g^\theta,\succ)=\frac{1}{2}\left(\frac{1}{m}+\frac{1}{n}\right)\sum_{w=1}^{n}\sum_{f=1}^{m} stv_{wf}(g^\theta,\succ). \quad (1.21)$$

The *expected stability violation* is $STV(g^\theta)=\mathbb{E}_{\succ}\left[stv(g^\theta,\succ)\right]$. We also write $stv(g^\theta)$ to denote the average stability violation across multiple profiles.

Theorem 11 *A randomized matching mechanism g^θ is ex ante stable up to zero-measure events if and only if $STV(g^\theta)=0$.*

Proof Since $stv(g^\theta,\succ)\geq 0$, then $STV(g^\theta)=\mathbb{E}_{\succ}\left[stv(g^\theta,\succ)\right]=0$ if and only if $stv(g^\theta,\succ)=0$ except on zero measure events. Moreover, $stv(g^\theta,\succ)=0$ implies $stv_{wf}(g^\theta,\succ)=0$ for all $w\in W$, all $f\in F$. This is equivalent to no ex ante justified envy. For firm f, this means $\forall w'\neq w$, $q^{\succ}_{wf}\leq q^{\succ}_{w'f}$ if $g^\theta_{w'f}>0$ and $q\succ_{wf}\leq 0$ if $g^\theta_{\perp f}>0$. Then there is no ex ante justified envy for firm f. Analogously, there is no ex ante justified envy for worker w. If g^θ is ex ante stable, it trivially implies $STV(g^\theta)=0$ by definition. □

Example 12 Consider a market with 3 workers and 3 firms, and the following preference profile ($\succ$):

$$\begin{aligned} w_1&: f_2,f_3,f_1,\perp \quad & f_1&: w_1,w_2,w_3,\perp\\ w_2&: f_2,f_1,f_3,\perp \quad & f_2&: w_2,w_3,w_1,\perp\\ w_3&: f_1,f_3,f_2,\perp \quad & f_3&: w_3,w_1,w_2,\perp \end{aligned}$$

The matching found by worker-proposing DA is $(w_1,f_3),(w_2,f_2),(w_3,f_1)$. This is a stable matching. Now consider the matching under RSD. We generate all possible priority orders and calculate the marginal matching probabilities as

$$r=\begin{pmatrix} \frac{11}{24} & \frac{1}{4} & \frac{7}{24}\\ \frac{1}{6} & \frac{3}{4} & \frac{1}{12}\\ \frac{3}{8} & 0 & \frac{5}{8} \end{pmatrix}$$

Here, f_2 and w_2 are the most preferred options for w_2 and f_2 respectively, and they would prefer to be matched with each other always rather than being fractionally matched. Thus (w_2, f_2) has ex ante justified envy and RSD is not stable.

The stability violations are (1.20):

$$stv_{wf}(g^{RSD}, \succ) = \begin{pmatrix} 0 & 0 & \frac{11}{2592} \\ \frac{1}{288} & \frac{1}{54} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

By (1.21), the overall stability violation is $stv(g^{RSD}, \succ) = \frac{17}{1944}$.

We turn now to strategy-proofness. For this, we relax the requirement of ordinal strategy-proofness and consider incentive alignment for the equi-spaced utility function, denoted $u_i^{(eq)}$ for agent i. We say that a randomized mechanism is *strategy-proof for the equi-spaced utility* (SP) if and only if, for all agents $i \in W \cup F$, for any preference profile $\succ$, and all reports $\succ_i'$, we have

$$\mathbb{E}_{\mu \sim g^\theta(\succ_i, \succ_{-i})}\left[u_i^{(eq)}(\mu(i))\right] \geq \mathbb{E}_{\mu \sim g^\theta(\succ_i', \succ_{-i})}\left[u_i^{(eq)}(\mu(i))\right]. \tag{1.22}$$

Let $u_i^{(eq)}(r; \succ_i) = \mathbb{E}_{\mu \sim r}\left[u_i^{(eq)}(\mu(i))\right]$ denote the expected utility for randomized matching r. We define the *regret* to agent i at preference order $\succ$ as

$$\text{regret}_i(g^\theta, \succ) = \max_{\succ_i' \in P}\left[u_i^{(eq)}(g^\theta(\succ_i', \succ_{-i}); \succ_i) - u_i^{(eq)}(g^\theta(\succ); \succ_i)\right], \tag{1.23}$$

where $\succ_{-i} = (\succ_1, \ldots, \succ_{i-1}, \succ_{i+1}, \ldots, \succ_{n+m})$. The per-agent regret on profile $\succ$ is

$$regret(g^\theta, \succ) = \frac{1}{2}\left(\frac{1}{n}\sum_{w \in W} \text{regret}_w(g^\theta, \succ) + \frac{1}{m}\sum_{f \in F} \text{regret}_f(g^\theta, \succ)\right). \tag{1.24}$$

We define the *expected regret* as $RGT(g^\theta) = \mathbb{E}_\succ\left[regret(g^\theta, \succ)\right]$, and write $rgt(g^\theta)$ to denote the average per-agent regret across multiple profiles. We also refer to this as the *strategy-proof violation*.

Theorem 13 *A randomized matching mechanism g^θ is strategy-proof for equi-spaced utility up to zero-measure events if and only if $RGT(g^\theta) = 0$.*

The proof is similar to Theorem 11. Also, if the mechanism is deterministic, then zero expected regret implies that no agent can improve its preference ranking for any mis-report (again up to measure zero events).

The training problem for trading off stability and strategy-proofness, with $\lambda \in [0, 1]$ to control the trade-off, is

$$\min_\theta \lambda \cdot stv(g^\theta) + (1 - \lambda) \cdot rgt(g^\theta). \tag{1.25}$$

Step 3: Adopt a training procedure

As with RegretNet, we make use of SGD for training. The gradient of the violation of stability with respect to parameters θ is straightforward to calculate. For regret, the gradient is again made complicated by the nested maximization.

Given that two-sided matching has discrete types, for small problems we can compute the best mis-report by enumeration. For agent i, this is

$$\hat{\succ}_i^{(\ell)} \in \text{argmax}_{\succ_i'} \left[u_i^{(eq)}(g^\theta(\succ_i', \succ_{-i}^{(\ell)}); \succ_i^{(\ell)}) - u_i^{(eq)}(g^\theta(\succ^{(\ell)}); \succ_i^{(\ell)}) \right]. \tag{1.26}$$

We find the gradient of regret to agent i with respect to parameters θ by fixing its mis-report accordingly and adopting truthful reports for others.

In addition, we define the *per-agent IR violation* at input profile $\succ$ as

$$irv(g^\theta, \succ) = \sum_{w=1}^{n} \sum_{f=1}^{m} g_{wf}^\theta(\succ) \cdot (\max\{-q_{wf}, 0\} + \max\{-p_{wf}, 0\}). \tag{1.27}$$

We write $irv(g^\theta)$ to denote the average per-agent IR violation of a mechanism across multiple profiles. This *degree of IR violation* captures the second kind of ex ante justified envy in Definition 6 and is zero for the learned mechanisms and DA but non-zero for RSD.

1.4.5 Illustrative Experimental Results

We study both uncorrelated and correlated preferences:

- *Uncorrelated preference orders.*

  For each worker or firm, first sample uniformly at random from all preference orders. Then, with probability $p_{\text{trunc}} \geq 0$ (the *truncation probability*), choose at random a position at which to truncate this agent's preference order such that all subsequent positions are unacceptable.
- *Correlated preference orders.*

  First sample a preference order for each agent as in the uncorrelated case. Also sample, uniformly at random and independently and with the same truncation probability, a special worker preference order $\succ_{w*}$ and a special firm preference order $\succ_{f*}$. Each agent adopts the special preference order with probability $p_{\text{corr}} > 0$, i.e., $\succ_{w*}$ or $\succ_{f*}$ as appropriate to its side of the market.

We consider the following environments:

- $n = 4$ workers and $m = 4$ firms with uncorrelated preferences and $p_{\text{trunc}} = 0.5$.
- $n = 4$ workers and $m = 4$ firms with correlated preferences, and varying $p_{\text{corr}} = \{0.25, 0.5, 0.75\}$ and $p_{\text{trunc}} = 0.5$.
- $n = 4$ workers and $m = 4$ firms with uncorrelated preferences and $p_{\text{trunc}} = 0$.

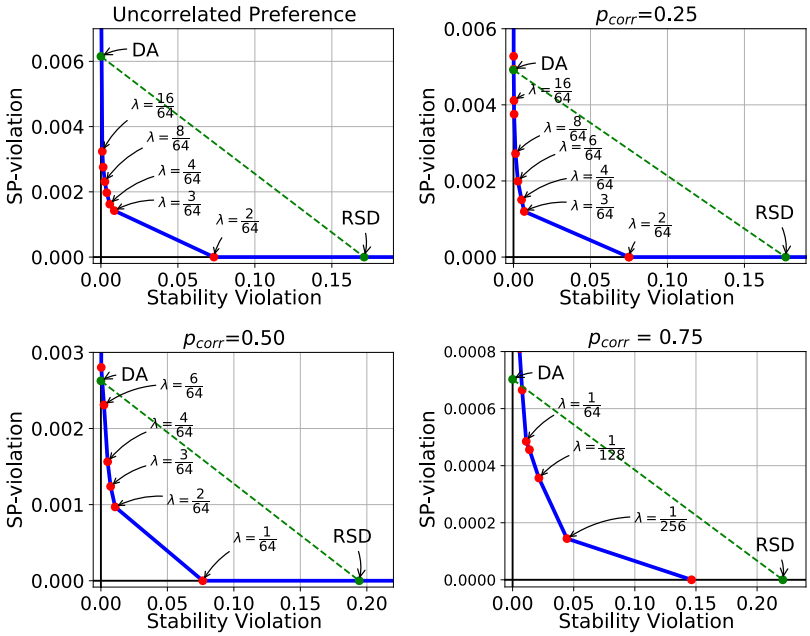


Figure 1.7 Truncated preference orders. The design frontier for the learned mechanisms for different choices of λ (red dots), and also showing RSD and the best of worker- and firm-proposing DA for SP violation. The sub-figures vary in the assumed correlation on preferences. The stability violation includes IR violation for RSD.

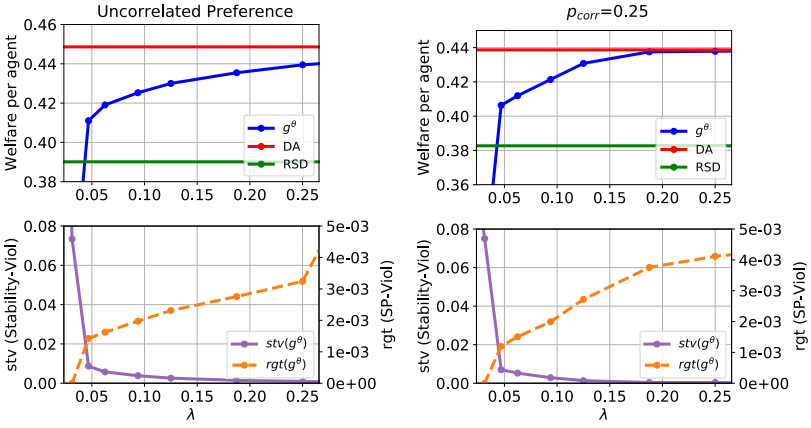


Figure 1.8 Truncated preference orders. **Top**: Per-agent welfare compared with RSD and the best of firm- and worker-proposing DA for welfare. **Bottom**: Stability violation and SP violation. For DA this considers the best of firm- and worker-proposing DA for SP violation. The stability violation for RSD is 0.171 and 0.176 for uncorrelated and correlated preferences, respectively. The SP-violation for DA is 6e-03 and 5e-03 for uncorrelated and correlated preferences, respectively. The stability violation includes IR violation for RSD.

We vary parameter λ between 0 to 1 in exploring the design frontier. In addition to stability and strategy-proofness, we also calculate the expected per-agent welfare for the equi-spaced utility function. We compare the learned mechanisms with RSD and DA. Because RSD does not guarantee IR, we include its IR violation as part of the reported stability violation. DA and the learned mechanisms satisfy IR.

We also calculate the similarity with DA. For preference profile $\succ$, the similarity to the worker-proposing DA (with matching $g^{w\text{-DA}}$) is the fraction of the matching of the DA mechanism that is retained in the learned mechanism g^θ, and

$$sim(g^\theta, \succ) = \frac{\sum_{(w,f):g_{wf}^{w\text{-DA}}(\succ)=1} g_{wf}^\theta(\succ)}{\sum_{(w,f):g_{wf}^{w\text{-DA}}(\succ)=1} 1}. \tag{1.28}$$

We define this analogously for firm-proposing DA, average the similarity across multiple profiles for each of worker- and firm-proposing DA, and define similarity $sim(g^\theta)$ as the maximum of these two quantities.

We also quantify the amount of randomization by computing the *expected normalized per-agent entropy*. For preference profile $\succ$, we calculate the normalized per-agent entropy (zero for a deterministic mechanism) as

$$H(g^\theta, \succ) = -\frac{1}{2n} \sum_{w \in W} \sum_{f \in \overline{F}} \frac{g_{wf}^\theta(\succ) \log_2 g_{wf}^\theta(\succ)}{\log_2 m} - \frac{1}{2m} \sum_{f \in F} \sum_{w \in \overline{W}} \frac{g_{wf}^\theta(\succ) \log_2 g_{wf}^\theta(\succ)}{\log_2 n}. \tag{1.29}$$

See Figures 1.7 and 1.8 for the results in environments with truncation. For Figure 1.7, we adopt as the DA baseline whichever of worker- and firm-proposing DA is best in terms of average SP violation on test data. The learned design frontier dominates the convex combination of DA and RSD.

For Figure 1.8, we adopt as the DA baseline whichever of worker- and firm-proposing DA is best in terms of per-agent welfare on test data. Figure 1.9 shows that for larger values of λ (≥ 0.2) the ANN tends to learn a mechanism that is deterministic and equivalent to a DA mechanism.

Considering Figures 1.7 and 1.8, we see that for very small λ values (emphasizing strategy-proofness) we learn mechanisms with welfare similar to that of RSD and with very small SP violations but with better stability than RSD. For slightly larger values of λ, say around 3/64 for small correlations between preferences, we learn mechanisms that are almost as stable as DA (stv ≤ 0.01) but with much lower SP violation (rgt ≤ 0.001). These mechanisms have welfare that is intermediate between RSD and DA. Comparing the scale of the y-axes in Figure 1.7, we see that higher correlation between preferences has little effect on stability but tends to remove opportunities for strategic behavior and improve strategy-proofness.

Figure 1.10 presents results for the environment without truncation (we still allow truncation for mis-reports). Compared with the results in Figure 1.7, the SP violation of DA is worse while the stability violation of RSD is better (because

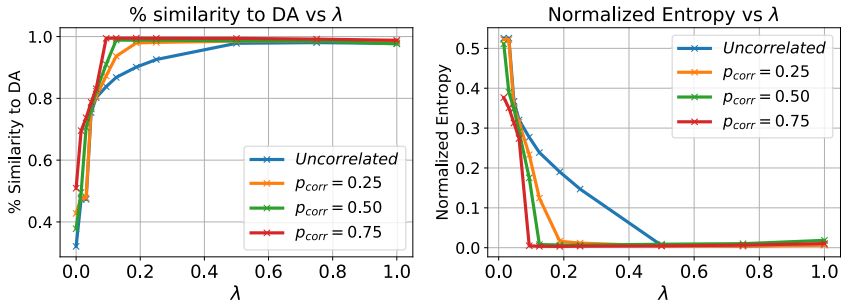


Figure 1.9 Non-truncated preferences. **Left**: Similarity of the learned mechanisms with DA. **Right**: Expected normalized per-agent entropy of the learned mechanisms.

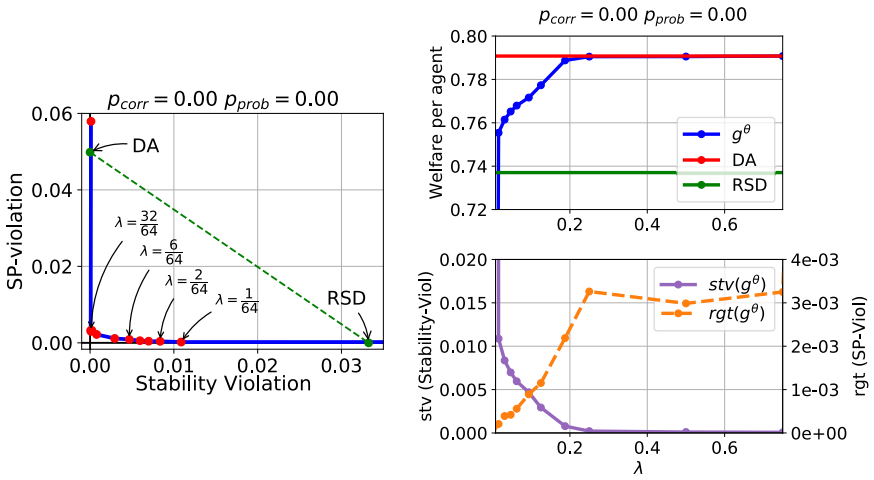


Figure 1.10 Non-truncated and uncorrelated preferences. **Left**: The design frontier for the learned mechanisms for different choices of λ (red dots) in an environment without preference truncation, and also showing RSD and the best of worker- and firm-proposing DA for SP violation. **Right**: Per-agent welfare, stability violation, and SP-violation. For welfare, this considers the best of worker- and firm-proposing DA for welfare. For SP-violation, this considers the best of worker- and firm-proposing DA for SP-violation. The stability violation for RSD is 0.033 and the SP-violation for DA is 0.05.

there can be no violation of IR). It is interesting that the learned mechanisms are able to achieve similar performance to the settings with truncation: very low stability violation (stv ≤ 0.005) and very low SP violation (rgt ≤ 0.001). Whereas DA is stable on all inputs, including truncated inputs, the learned mechanisms can adopt different behaviors on profiles with truncations, recognizing that this does

not affect in-distribution stability. To illustrate this, suppose firm f_1 has preference order $w_2, w_1, w_3, w_4, \perp$ and suppose DA assigns the firm to w_4, its least preferred worker. Now, f_1 may truncate to $w_2, w_1, \perp, w_3, w_4$, in which case a DA might match the firm to w_1, its second preferred worker. By comparison, the ANN can learn not to match the firm to any worker after this truncation, improving SP without compromising stability.

Taken as a whole, these results suggest new directions for economic theory. For example, are there mechanisms that for some distributions are provably almost as stable as DA and yet considerably more strategy-proof, and can these mechanisms and distributions be characterized? Are there mechanisms that for some distributions are provably almost as strategy-proof as RSD and yet considerably more stable, and can these mechanisms and distributions be characterized?

1.5 Discussion

We have seen that ANNs are a flexible tool with which to design and study matching markets. In addition to the matching problems described in this chapter, ANNs have been applied to the following settings:

- Small combinatorial auctions, which are auctions in which a buyer's value for a package of items can be super-additive in its value for individual items.
- Bayesian incentive-compatible auctions for buyers with budget constraints.
- Incentive-aligned social choice mechanisms such as multi-facility location.
- Auctions that allocate items efficiently while also minimizing the total expected payment collected from buyers.

Differentiable economics has also led to the discovery of provably-optimal auctions by making use of an architecture (RochetNet) that provides exact strategy-proofness for single-buyer settings. Methodological developments include the use of methods from robust machine learning to certify the worst-case violation of strategy-proofness of a learned mechanism as well as ANN architectures that impose symmetry on auction rules.

There are a number of interesting future directions, including new applications, for example to contract design, double auctions, or collusion-proof auctions. We need architectural innovations that incorporate characterization results from economic theory, and methods that provide robustness to adversarial inputs and transform learned mechanisms with approximate properties of interest to those with exact properties. Scaling to larger combinatorial auction problems will require the use of succinct representations of inputs and outputs. The communication of results back to economic theory will be aided through methods to interpret learned mechanisms, as well as lower- and upper-bounds on performance that serve to confirm opportunities for new theoretical development.

1.6 Chapter Notes

The agenda of automated mechanism design (AMD) was introduced by Conitzer and Sandholm (2002) and early work makes use of integer programming and linear programming. Later, there was work on optimizing within parametrized classes of mechanisms, leading for example to the AMD baseline in Figure 1.5 (Guo and Conitzer, 2010; Sandholm and Likhodedov, 2015).

The network architecture discussed in Section 1.3 is the *RegretNet* architecture from Dütting et al. (2019), who also provide generalization bounds for regret and revenue and experimental results for small combinatorial auctions; see also Dütting et al. (2020) for the single buyer *RochetNet* architecture, which is exactly strategy-proof and has been used to support conjectures and discover new, provably-optimal auctions. Shen et al. (2019) also derive theoretically optimal designs using a variation on RochetNet. Earlier, Cole and Roughgarden (2014) introduced the study of the sample complexity for optimal auction design. Dütting et al. (2015) introduced expected *ex post* regret to quantify approximate strategy-proofness in studying the application of support-vector machines to AMD.

Differentiable economics has been applied to budget-constrained auction design (Feng et al., 2018), multi-facility location (Golowich et al., 2018), and payment-minimizing mechanism design (Tacchetti et al., 2019). Curry et al. (2020) demonstrate how to develop certificates for approximate strategy-proofness. Rahme et al. (2021) introduce permutation-equivariance into ANN architectures as a way to impose symmetry on learned auctions. Working with Bayesian incentive compatibility rather than dominant-strategy incentive compatibility, Daskalakis and Weinberg (2012) and Conitzer et al. (2020) provide transforms of ϵ-IC mechanisms into IC mechanisms.

The development in this chapter of the application to two-sided matching design follows Ravindranath et al. (2021), who adopt *ordinal SP* and quantify the violation of first-order stochastic dominance rather than adopt ex post regret and equi-utility as in the present chapter. The results are qualitatively similar to those presented here. The decomposition theorem is discussed in Chapter **??**, and Budish et al. (2013) provide a general result that applies when some agents are unmatched. The concept of ex ante justified envy is due to Kesten and Ünver (2015). Narasimhan et al. (2016) apply support-vector machines to the design of stable matching mechanisms, finding rules on the stable matching polytope (see Chapter **??**).

The results presented here use the *PyTorch* deep learning library and the Adam optimizer with a learning rate of $\alpha = 0.001$. For the auction results, we use augmented Lagrangian solver across 80 steps (with $\rho = 1$), where each step involves 5,000 mini-batch SGD iterations, each mini-batch consisting of 128 valuation profiles. For each profile, 25 iterations of gradient ascent are used to find a defeating valuation for each buyer. For the matching results, we train for 50,000 mini-batch based SGD iterations, with each mini-batch consisting of 1,024 preference profiles.

Acknowledgments

We would like to thank the reviewers for their useful comments as well as our wonderful collaborators, Paul Dütting, Harikrishna Narasimhan, Scott Kominers, Shira Li, Jonathan Ma, and Noah Golowich. This work is supported in part by the Harvard Data Science Initiative and an award from AWS.

References

Budish, Eric, Che, Yeon-Koo, Kojima, Fuhito, and Milgrom, Paul. 2013. Designing Random Allocation Mechanisms: Theory and Applications. *American Economic Review*, **103**(2), 585–623.

Cole, Richard, and Roughgarden, Tim. 2014. The Sample Complexity of Revenue Maximization. Pages 243–252 of: *Proceedings of the 46th ACM Symposium on Theory of Computing*.

Conitzer, Vincent, and Sandholm, Tuomas. 2002. Complexity of Mechanism Design. Pages 103–110 of: *Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence*.

Conitzer, Vincent, Feng, Zhe, Parkes, David C., and Sodomka, Eric. 2020. Welfare-Preserving ε-BIC to BIC Transformation with Negligible Revenue Loss. *CoRR*, **abs/2007.09579**.

Curry, Michael, Chiang, Ping-yeh, Goldstein, Tom, and Dickerson, John. 2020. Certifying Strategyproof Auction Networks. Pages 4987–4998 of: *Advances in Neural Information Processing Systems*, vol. 33.

Daskalakis, Constantinos, and Weinberg, Seth Matthew. 2012. Symmetries and optimal multi-dimensional mechanism design. Pages 370–387 of: *Proceedings of the 13th ACM conference on Electronic commerce*.

Dütting, Paul, Fischer, Felix, Jirapinyo, Pichayut, Lai, John K., Lubin, Benjamin, and Parkes, David C. 2015. Payment Rules through Discriminant-Based Classifiers. *ACM Transactions on Economics and Computation*, **3**(1), 1–41.

Dütting, Paul, Feng, Zhe, Narasimhan, Harikrishna, Parkes, David C., and Ravindranath, Sai Srivatsa. 2019. Optimal Auctions through Deep Learning. Pages 1706–1715 of: *Proceedings of the 36th International Conference on Machine Learning, ICML*.

Dütting, Paul, Feng, Zhe, Narasimhan, Harikrishna, and Parkes, David C. 2020. Optimal Auctions through Deep Learning. *CoRR*, **abs/1706.03459**.

Feng, Zhe, Narasimhan, Harikrishna, and Parkes, David C. 2018. Deep Learning for Revenue-Optimal Auctions with Budgets. Pages 354–362 of: *Proceedings of the 17th Conference on Autonomous Agents and Multi-Agent Systems*.

Golowich, Noah, Narasimhan, Harikrishna, and Parkes, David C. 2018. Deep Learning for Multi-Facility Location Mechanism Design. Pages 261–267 of: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*.

Guo, M., and Conitzer, V. 2010. Computationally Feasible Automated Mechanism Design: General Approach and Case Studies. In: *Proceedings of the 24th AAAI Conference on Artificial Intelligence*.

Kesten, Onur, and Ünver, M. Utku. 2015. A theory of school-choice lotteries. *Theoretical Economics*, **10**(2), 543–595.

Narasimhan, Harikrishna, Agarwal, Shivani, and Parkes, David C. 2016. Automated Mechanism Design without Money via Machine Learning. In: *Proceedings of the 25th International Joint Conference on Artificial Intelligence*.

Rahme, Jad, Jelassi, Samy, Bruna, Joan, and Weinberg, S. Matthew. 2021. A Permutation-Equivariant Neural Network Architecture For Auction Design. Pages 5664–5672 of: *Thirty-Fifth AAAI Conference on Artificial Intelligence*.

Ravindranath, Sai Srivatsa, Feng, Zhe, Li, Shira, Ma, Jonathan, Kominers, Scott D., and Parkes, David C. 2021. Deep Learning for Two-Sided Matching. *CoRR*, **abs/2107.03427**.

Sandholm, Tuomas, and Likhodedov, Anton. 2015. Automated Design of Revenue-Maximizing Combinatorial Auctions. *Operations Research*, **63**(5), 1000–1025.

Shen, Weiran, Tang, Pingzhong, and Zuo, Song. 2019. Automated Mechanism Design via Neural Networks. In: *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*.

Tacchetti, Andrea, Strouse, D. J., Garnelo, Marta, Graepel, Thore, and Bachrach, Yoram. 2019. A Neural Architecture for Designing Truthful and Efficient Auctions. *CoRR*, **abs/1907.05181**.