Learning preferences in an accumulation-to-threshold model of decision making

Taher Rahgooy[1], K. Brent Venable[1, 2], Jerome R. Busemeyer[3]
[1]University of West Florida, Pensacola, FL, USA
[2]Institute for Human and Machine Cognition, Pensacola, FL, USA
[3]Indiana University, Bloomington, IN, USA
trahgooy@students.uwf.edu, bvenable@ihmc.org, jbusemey@indiana.edu

Abstract

Understanding human decision processes has been a topic of intense study in different disciplines including psychology, economics, and artificial intelligence. Indeed, modeling human decision making plays a fundamental role in the design of intelligent systems capable of rich interactions. Decision Field Theory (DFT) [3] provides a cognitive model of the deliberation process that precedes the selection of an option. DFT is grounded in psychological principles and has been shown to be effective in modeling several behavioral effects involving uncertainty and interactions among alternatives. In this paper, we address the problem of learning the internal DFT model of a decision maker by observing only his final choices. In our setting choices are among several options which are evaluated according to different attributes. Our approach, based on Recurrent Neural Networks, extracts underlying preferences compatible with the observed choice behavior and, thus, provides a method for learning a rich preference model of an individual which encompasses psychological aspects and which can be used as a more realistic predictor of future behavior.

1 Introduction

Preferences play a fundamental role in the understanding of human behavior and in the design of intelligent systems. On the one side, they lie at the core of decision making, a task which is central in describing how humans function in everyday life. On the other hand, qualitative and quantitative measures of satisfaction or utility, are at the basis of optimization in complex problems, a challenge which artificial agents have been successful in tackling.

In this paper we address the challenge of automatically extracting information about an individuals preferences by observing his choice behavior. Preference learning [6, 18, 15, 11] has been a topic of intense investigation by the AI community. This body of work has been crucial in enabling the use of preference models developed in the context of artificial agents as their direct definition is often impractical [6, 19]. However, these approaches are less suitable to model human behavior when it comes to decision making. Several properties which are necessary for optimization, such as, transitivity for example, infringe their ability to accommodate behavioral violations of rational principles.

On the other hand, in the area of psychology, cognitive computational models have been designed for the purpose of faithfully capturing human behavior in decision making. In these models, the parameters are often defined by hand in order to accurately replicate average behavior of individuals and, to the best of our knowledge, no method capable of automatically inferring them has been proposed.

In this paper we focus on Multialternative Decision Field Theory (MDFT) a dynamic-

cognitive approach to decision making based on the idea that the process of deliberation consists of a sequential sampling and accumulation of information over time. MDFT formally generalizes other models of decision making such as the classical multi-attribute decision model [9] and preferential choice model [5, 21]. MDFT is able to replicate fundamental aspects of human decision making such as, for example, violation to transitivity, preference reversal under time pressure and the well known effects of similarity, attraction and compromise [3].

In this paper we present a method based on machine learning and, in particular Recurrent Neural Networks, capable of inferring an underlying MDFT model compatible with the observed choice behavior on an individual. A key aspect of our method is that it learns in the presence of uncertainty and partial information. In fact, the training data contains only examples of choices and deliberation times and does not include how attention was allocated to the attributes during the deliberation process. In our experimental results we compare the original and learned models in terms of similarity of both the produced choice distributions and the initial evaluations for the options. As shown in Section 7, our learning approach is able to recover a model which is extremely close to the original one in terms of both measures.

Our work is novel as it tackles the problem of learning automatically a cognitive architecture of human decision making. From the preference learning perspective, it provides a way to extract multi-attribute preferences in the context of a complex systems involving stochasticity and bounded rationality. From a cognitive standpoint, our method allows to use these architectures at the level of the individual. Learned MDFT models can be used as behaviorally more accurate predictors of future choices in the context of recommender systems. Our work is also useful in settings where sets of options are presented simultaneously to a user (e.g., option slates). In fact, since the learned model inherits the characteristics of MDFT it is able to predict how choices change if different options are presented as competitors. Moreover, our results are relevant in the context of artificial personal assistants which often need to recover a model of the supported individuals by observing their behavior. The capability of automatically inferring an MDFT model of the user can allow the agent to have more realistic representation of his decision making behavior. The paper is organized as follows. The first two sections provide background on MDFT and Recurrent Neural Networks. In Section 4 we discuss related work. In Section 5 we formally define the learning problem which we tackle in Section 6. In Section 7 we describe the results of our experimental study and we then conclude, in the last section, with future work directions.

2 Multialternative Decision Field Theory

Decision field theory (DFT) is a dynamic-cognitive approach that models human decision making based on psychological principles [3]. DFT models the preferential choice as an accumulative process in which the decision maker attends to a specific attribute at each time to derive comparisons among options and update his preferences accordingly. Ultimately the accumulation of those preferences forms the decision maker's choice. DFT has been extended by [17] to multialternative preferential choice (denoted MDFT, for Multialternative DFT), where an agent is confronted with multiple options and equipped with an initial personal evaluation for them according to different criteria called attributes. For example, a student who needs to chose a main course among those offered by the cafeteria will have in mind an initial evaluation of the options in terms of how tasty and healthy they look. More formally, MDFT, in its basic formulation [17], is composed of:

Personal Evaluation: We assume a set of options $\{o_1, \ldots, o_n\}$ and a set of attributes $\{A_1, \ldots, A_J\}$. The subjective value of option o_i on attribute A_j is denoted by m_{ij} and

stored in matrix $\mathbf{M}$ for all options and attributes. In our example, let us assume that the cafeteria options for main course are *Salad (S)*, *Burrito (B)* and *Vegetable pasta (V)* and that the attributes considered are *Taste* and *Health*. Matrix $\mathbf{M}$ containing the student's initial preferences for the three options according to the two attributes could be defined as follows:

$$\mathbf{M} = \begin{bmatrix} 1 & 5 \\ 5 & 1 \\ 2 & 3 \end{bmatrix}$$

In this matrix the rows correspond to the options in order (S, B, V) and the columns to the attributes $Taste$ and $Health$. For example, we can see that $Burrito$ has a high preference in terms of taste but low in terms of nutritional value.

Attention Weights: Attention weights are used to express how much attention is allocated to each attribute at each particular time t during the deliberation process. We denote them by a one-hot column vector $\mathbf{W}(t)$ where $W_j(t)$ is a value denoting the attention to attribute j at time t. We adopt the common simplifying assumption that, at each point partial, the decision maker attends to only one attribute. Thus, $W_j(t) \in \{0, 1\}, \forall t, j$. In our example, where we have two attributes, at any point in time t, we will have $\mathbf{W}(t) = [1, 0]$, or $\mathbf{W}(t) = [1, 0]$, representing that the student is attending to, respectively, $Taste$ or $Health$. In general, the attention weights change across time according to a stationary stochastic process with probability distribution $\mathbf{w}$, where w_j is the probability of attending to attribute A_j. In our example, defining $w_1 = 0.55$ and $w_2 = 0.45$ would mean that at each point in time, the student will be attending $Taste$ with probability 0.55 and $Health$ with probability 0.45. In other words, $Taste$ matters slightly more to this particular student than $Health$.

Contrast Matrix: Contrast matrix $\mathbf{C}$ is used to compute the advantage (or disadvantage) of an option with respect to the other options. For example, $\mathbf{C}$ can be defined by contrasting the initial evaluation of one alternative against the average of the evaluations of the others. In this case, for three options, we have:

$$\mathbf{C} = \begin{bmatrix} 1 & -1/2 & -1/2 \\ -1/2 & 1 & -1/2 \\ -1/2 & -1/2 & 1 \end{bmatrix}$$

At any moment in time, each alternative in the choice set is associated with a **valence** value. The valence for option o_i at time t, denoted $v_i(t)$, represents its momentary advantage (or disadvantage) when compared with other options on some attribute under consideration. The valence vector for n options $o_1, \ldots, o_n$ at time t, denoted by column vector $\mathbf{V}(t) = [v_1(t), \ldots, v_n(t)]^T$, is formed by:

$$\mathbf{V}(t) = \mathbf{C} \times \mathbf{M} \times \mathbf{W}(t) + \epsilon(t) \tag{1}$$

Where $\epsilon(t) \sim N(0, \sigma^2)$ is random noise with zero mean. In our example, assuming zero noise, the valence vector at any time point in which $\mathbf{W}(t) = [1, 0]$, is $\mathbf{V}(t) = [(1-7)/2, (5-3)/2, (2-6)/2]^T$.

In MDFT preferences for each option are accumulated across the iterations of the deliberation process until a decision is made. This is done by using the **Feedback matrix**, which defines how the accumulated preferences affect the preferences computed at the next iteration. This interaction depends on how similar the options are in terms of their initial evaluation contained in matrix $\mathbf{M}$. Intuitively, the new preference of an option is affected positively and strongly by the preference it had accumulated so far, it is strongly inhibited by the preference of other options which are similar and this lateral inhibition decreases as the dissimilarity between options increases.

Clearly, the concept of similarity plays a crucial role. A common way to define it is to project the initial evaluations contained in $\mathbf{M}$ on the directions of indifference and dominance respectively [8]. Formally, this is done by defining distance matrix $\mathbf{D} = [D_{ij}]$ where $D_{ij} = (\mathbf{M}_i - \mathbf{M}_j) \times \mathbf{H}_b \times (\mathbf{M}_i - \mathbf{M}_j)^T$ and $\mathbf{H}_b$ is defined as

$$\mathbf{H}_b = \frac{1}{2} \times \begin{bmatrix} b+1 & b-1 \\ b-1 & b+1 \end{bmatrix} \tag{2}$$

where constant b determines the ratio of emphasis on the dominance direction with respect to the indifference direction.

Intuitively, the difference between competitive options will have a larger component along the line of indifference while the difference between two options where one is dominating will have a stronger component along the line of dominance. In order to simulate the common human behavior where dominated options are rapidly discarded during a decision process, more emphasis should be given to differences in the dominance direction than in the indifference direction. This is achieved whenever $b > 1$. At this point matrix $\mathbf{S}$ can be defined by mapping the distance via Gaussian function:

$$\mathbf{S} = \delta - \phi_2 \times \exp\left(-\phi_1 \times \mathbf{D}^2\right) \tag{3}$$

where δ, ϕ_1, ϕ_2 are the identity matrix, the decay parameter, and the sensitivity parameter respectively. The identity matrix δ is used to model positive self feedback. We can also see that when $i \neq j$, and, thus, $\delta_{ij} = 0$, we have a lateral inhibition $-\phi_2 \times \exp\left(-\phi_1 \times \mathbf{D}^2\right)$ which depends on the distance $\mathbf{D}$ between options. In our example, if we set $b = 10$, $\phi_1 = 0.01$ and $\phi_2 = 0.1$ we get the following $\mathbf{S}$ matrix:

$$\mathbf{S} = \begin{bmatrix} +0.9000 & -0.0000 & -0.0405 \\ -0.0000 & +0.9000 & -0.0047 \\ -0.0405 & -0.0047 & +0.9000 \end{bmatrix}$$

At any moment in time, the preference of each alternative is calculated by

$$\mathbf{P}(t+1) = \mathbf{S} \times \mathbf{P}(t) + \mathbf{V}(t+1) \tag{4}$$

where $\mathbf{S} \times \mathbf{P}(t)$ is the contribution of the past preferences and $\mathbf{V}(t+1)$ it the valence computed at that iteration. Usually the initial state $\mathbf{P}(0)$ is defined as $\mathbf{0}$, unless defined otherwise due, for example, to prior knowledge on past experiences.

As, feedback matrix $\mathbf{S}$ is a function of personal evaluations $\mathbf{M}$ and parameters $\theta = \langle \phi_1, \phi_2, b \rangle$ and for us $\mathbf{C}$ is always defined as described above, we will refer to an MDFT model by $\langle \mathbf{M}, \theta, \mathbf{w}, \sigma^2 \rangle$. Given an MDFT model one can simulate the process of deliberating among the options by accumulating the preferences for a number of iterations. The process can be stopped either by setting a threshold on the preference value and selecting whichever option reaches it first or, by fixing the number of iterations and then selecting the option with highest preference at that point. In general, different runs of the same MDFT model may return different choices because of the uncertainty on the attention weights distribution. This allows MDFT to effectively replicate behaviors observed in humans [4].

3 Recurrent neural networks

Artificial neural networks [20] are one of the most powerful and expressive learning models that are successfully used in many real world problems. However, conventional neural networks are not suitable for sequential problems with variable length where the current

prediction depends on the previous predictions. Recurrent neural networks (RNNs)[20] are an extension of conventional neural networks designed to deal with such problems. In their simplest form, at any given point in time t, RNNs maintain a state variable h_t as a function of current input x_t and the value of the previous state h_{t-1}, that is, $h_t = f_\eta(x_t, h_{t-1})$ where η is the set of parameters to be learned. In addition, a loss function $L(y_t, h_t)$, where y_t is the observed target value at time t, is appropriately defined to quantify the error of prediction at each time. The learning objective is to minimize an aggregate function of losses (e.g. the average, the sum or, simply, the last loss value) subject to parameters η. This objective is achieved by iteratively running back-propagation algorithm [20], or one of its improved versions [1], on the training data.

4 Related work

Preference learning [6] is about inducing predictive preference models from empirical data. Although learning preferences can be reduced to conventional machine learning approaches in some cases, in general, it is a more challenging task because of the complexity of the output (which usually takes the form of rankings or partial orders) and the incompleteness of the input (such as indirect feedback or implicit preference information) [6]. Most preference learning tasks are defined by an option input space and a label output space. The output space is used to define the orderings. For example in *label ranking* the goal is to learn a ranking function mapping the input space into permutations of the labels. The training data for these tasks is usually a set of pairwise comparisons. Our setting is significantly different from those considered in these tasks. First of all our training data is made of choices, and not pairwise comparison. Furthermore, these choices are only the final outcome of a dynamic process, which we aim at replicating as a whole.

In another related research, [2] presents a model which combines utility-based models of preference in economics with Bayesian inference to learn complex structured preferences. Their model predicts the users choice based on previous choices and also on the relationships between new and old options. Our work is different in that we focus on cognitive models of preferences, rather than economic models, and, while they assume prior knowledge about relations between options, we do not rely on any local information on options' preferences.

The most closely related work is presented in [16] where the authors study the problem of learning personal evaluations in a MDFT model using RNNs. What is presented here differs from [16] in several fundamental ways. We consider a threshold on preference as a stopping criterion instead of the number of iterations. This drastically complicates the problem of inducing personal evaluations as noted in [3] in the case of a maximum likelihood estimation approach. Moreover, our RNN architecture allows to both learn personal evaluations as well as the attention weights. In other words, we are able to learn how much an individual cares about a given attribute, in addition to how much the individual likes an option with respect to that attribute. Finally, we present a more extensive experimental study and we provide a comparison in terms of time efficiency and solution quality with respect to a maximum likelihood method based on [17].

5 Problem formulation

In this section we formalize the problem of learning parameters compatible with a decision maker's observed choice behavior assuming an underlying MDFT model.

Problem formulation Given a set of options $O = \{o_1, \ldots, o_n\}$ we consider a setting in which a user is confronted with k choice problems. In each problem, choosing among a subset of O, namely option-set, taking into account a set of attributes $\{A_1, \ldots, A_J\}$.

Given a dataset of observed choice distributions $\mathcal{D} = \{\langle d_1, \tau_1\rangle, \ldots, \langle d_N, \tau_N\rangle\}$, where d_i is the observed distribution of choices for ith option-set and τ_i is the deliberation preference threshold and assuming fixed values for θ and σ^2 parameters, we want to find a $\hat{\mathbf{M}}$ and/or $\hat{\mathbf{w}}$ in such a way that the MDFT $\hat{\mathcal{M}} = \langle \hat{\mathbf{M}}, \theta, \hat{\mathbf{w}}, \sigma^2\rangle$ generates a similar choice distribution as the one observed in $\mathcal{D}$.

Evaluation One option to compare two choice probability distributions is the Kullback-Leibler (KL) divergence [12]. It is a measure that quantifies in bits how far a probability distribution $p = \{p_i\}$ is from another distribution $q = \{q_i\}$, formally $D_{kl}(p||q) = \sum_i p_i \log(\frac{p_i}{q_i})$.

However, a problem with using KL-Divergence as a distance measure is that it is not symmetric. We, thus, use the Jensen-Shannon Divergence [13] which is an extension of KL-Divergence that overcomes this problem:

$$D_{js}(p||q) = \frac{D_{KL}(p||\frac{p+q}{2}) + D_{KL}(q||\frac{p+q}{2})}{2} \tag{5}$$

Let, $h_{\mathcal{D}}, h_{\hat{\mathcal{M}}}$ denote the histogram of choices in $\mathcal{D}$ and generated by $\hat{\mathcal{M}}$, respectively. In the next sections we will describe a learning approach and show how it minimizes $D_{js}(h_{\mathcal{D}}||h_{\hat{\mathcal{M}}})$.

6 Learning preference in an MDFT model

We recall that each time an MDFT model is run on a set of options $\{o_1, \ldots, o_k\}$, an option is selected after a sequence of deliberation steps, where, at each step, attention to attributes is allocated according to a specific attention vector (see Section 2). We can thus associate the deliberation sequence with a sequence of attention vectors: $\langle \mathbf{W}(1), \ldots, \mathbf{W}(T)\rangle$, which we will call attention sequence for short. In our running example, if the student first considered health, then taste and then health again, the attention sequence would be $\langle [1,0], [0,1], [1,0]\rangle$. It is easy to see, that if, in addition to dataset $\mathcal{D}$ we also had the associated attention sequences, then we could easily map the problem of learning an MDFT model of the user's behavior into a multi-class classification problem. Options $1, \ldots n$ would correspond to k classes, each sequence of attention vectors $\langle \mathbf{W}(1), \ldots, \mathbf{W}(T)\rangle$ would be an input, and the selected option would be the output. In what follows we propose a learning approach based on this intuition designed to bypass the absence of information on the attention sequences.

6.1 Learning architecture

As a first step we show how we can map an MDFT model into a recurrent neural network architecture as the one depicted in Figure 1. A pass over the network, from input to output, at time t corresponds to one iteration of the MDFT model. The network is structured into two sub-networks, respectively at the top and bottom of Figure 1. The top sub-network is responsible for computing the valences (as defined in Equation 1), while the bottom one corresponds to the preference update (as defined in Equation 4). As it can be seen, there is a one-to-one correspondence between networks' weights and the parameters of the MDFT. More in detail, starting from the top sub-network, the first layer of nodes, denoted with $\{1, \ldots, J\}$, corresponds to the attributes and the weights of their inputs, $\{w_1^t, \ldots w_J^t\}$ are the elements of the attention weight vector (at time t). The weights of the connections between this first layer of nodes and the second one correspond to the initial evaluations contained in matrix $\mathbf{M}$ and the weights between the second and third layer correspond to the values of contrast matrix $\mathbf{C}$. Notice that the third layer of nodes corresponds to the valence values, $v_1^t, v_2^t, \ldots, v_k^t$, one for each option. The inputs of the bottom sub-network are the preferences from the previous iteration $p_1^{t-1}, p_2^{t-1}, \ldots, p_n^{t-1}$, and the weights between the

two layers are the values of the **S** matrix. The output of the two sub-networks are combined to synthesize the new preferences at time t.

One deliberation process of length T is simulated by cycling over the network for T times and obtaining final preferences $\{P_1^T, P_2^T, \ldots, P_k^T\}$. We treat the final preferences as scores of a multi-class classifier and use hinge loss to optimize the parameters:

$$\text{loss}(\mathbf{P}, c) = \frac{\sum_{i \neq c}^{|\mathbf{P}|} \max(0, \text{m} - P_c + P_i))}{|\mathbf{P}| - 1} \tag{6}$$

where $\mathbf{P}$ is the predicted preference vector, c is the ground-truth choice with predicted preference P_c, and m is the margin parameter of hinge loss. During training the **C** parameters are fixed as defined in the description of the contrast matrix **C** in Section 2. **M** or **w** parameters are updated by error propagation and **S** parameters are recomputed given the new **M** or **w** parameters and constants ϕ_1 and ϕ_2 (see Equation 3).

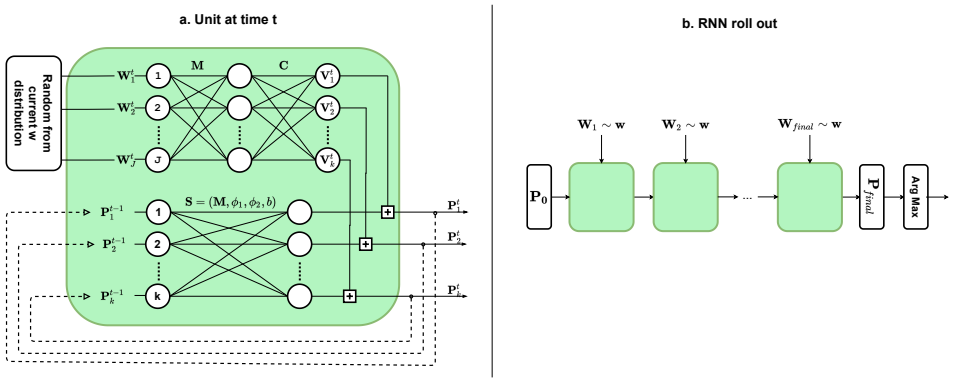

Figure 1: Recurrent neural network representing an MDFT model. a - The architecture of a single unit of the network. b - A roll-out representation of the deliberation process using the RNN. At each time step a sample of **w** is used as input.

6.2 Training

It is easy to see that it is not possible to train the RNN described above using choice dataset $\mathcal{D}$ because we don't have any information about the attention sequences that generated the choices. However, attention sequences are the only way in which uncertainty is manifested in the MDFT model, that is, it is a difference in attention sequences which makes it possible to obtain different choices from the same set of initial evaluations. Given this, we conjecture that sequences that generate the same options will have a common trend in terms of overall attention allocated to each attribute during deliberation. We corroborate our hypothesis with an extensive experimental study. In Figure 2 we show results for 4 samples, generated from a single MDFT model with three options and two attributes, and consisting each of 100 deliberation simulations involving 100 iterations each. The plots show, for each option and for each iteration, the percentage of previous iterations at which attention was allocated to the first attribute (i.e., when the attention vector was W_2, as defined in Section 2). The different colors correspond to different choices (o_1, o_2 and o_3, respectively) and the legends in each of the 4 sub-figures show the number of times that a particular option was returned over the 100 trials. From the figure we can see that each option has an attention allocation trend that, on average, is completely different from the other options. We can also see

that different samples generate very similar trends for the same options, confirming our hypothesis.

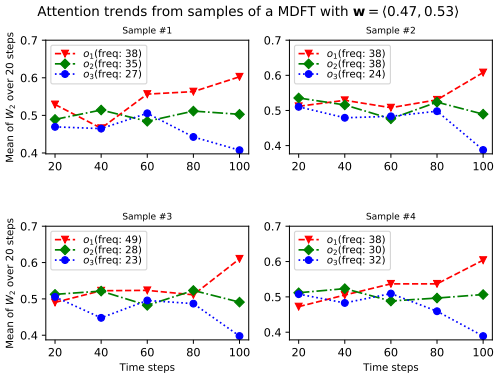

Figure 2: Average percentage of iterations at which attribute A_1 is attended for a given number of deliberation steps. Each sub-figure is the average over 100 samples. All samples are generated by the same MDFT.

In conclusion we see that the final choice will be the same for all attention sequences that follow a particular option trend. Therefore, we can use any subset of those sequences interchangeably for training our model as long as the average trend approximately remains the same. Based on this, we design an algorithm that adjusts the output of the RNN so that it aligns with the frequencies of returned options in the dataset.

We explain this algorithm with a simplified example. Consider an MDFT with three options $\{o_1, o_2, o_3\}$ and two attributes A_1 and A_2, similar to the one described in Section 2. Let us assume that we run this model for 10 times and that three options are returned with frequencies $\langle 3, 2, 5 \rangle$. In other words, o_1 was selected 3 times out of 10, o_2 2 times and o_3 the remaining 5. Now let us assume that we run the estimated model represented by RNN 10 times and that the three options are returned with the following frequencies: $\langle 6, 1, 3 \rangle$. In Table 1 (first three columns) we show an example of returned sample set with final preferences and choices. In the first phase of the algorithm, we sort the samples generated for each otpion by the RNN in decreasing order of the preference they assign to the selected option (as shown in Table 1). The intuition is that samples with higher preference are better representatives for the selected option.

Sample #	Predicted Preferences	Predicted Choice	Aligned with
1	$\langle 5, 3, 1.5 \rangle$	1	1
2	$\langle 5, -3, -1.5 \rangle$	1	1
3	$\langle 2.5, 1, 1.5 \rangle$	1	1
4	$\langle 2, .3, -1.5 \rangle$	1	3^*
5	$\langle 1, 0.3, -1.5 \rangle$	1	2^*
6	$\langle 1, 1.4, .5 \rangle$	1	3^*
7	$\langle 1, 3, 1.5 \rangle$	2	2
8	$\langle -1, 3, 10 \rangle$	3	3
9	$\langle 2, 4, 7 \rangle$	3	3
10	$\langle -1, 1, 2.5 \rangle$	3	3

Table 1: Samples generated from the learned model. * indicates re-assigned choice.

Next, we reassign some of the samples to a different option so to align with the frequencies observed in the data set. In our estimated sample set we have 6 samples where o_1 is selected, 1 where o_2 is selected and 3 where o_3 is selected, whereas we want to have 3 for o_1, 2 for o_2 and 5 for o_3. In order to obtain this we keep the first three samples for o_1 and we reassign the last three, which are in excess and weaker representatives for o_1, assigning one of them to o_2 and the other two to o_3. Reassignments can be done randomly (as in our implementation), or based on the preference assigned by the sample to the choice they have been reassigned to.

The result of the reassignment is shown in the last column of Table1.

At this point the hinge loss error defined in Equation 6 can be computed and propagated (e.g., for sample 4 of Table 1 we have $P_c = -1.5$ and $P_1 = 1$). In this way, the training procedure penalizes the miss-aligned samples and the error propagation causes the personal evaluation of more frequent options to shift towards most frequent attention sequences and vice-versa. The training algorithm, thus, iteratively improves the alignments and finds the appropriate attention sequence for each option.

Summarizing, given dataset $\mathcal{D} = \{\langle d_1, \tau_1\rangle, \ldots, \langle d_N, \tau_k\rangle\}$ of size N, training proceeds as follows:

1. **Initialization:** All parameters of the RNN network are initialized randomly, except for the contrast parameters $\mathbf{C}$ and θ and σ^2 which are fixed as previously described in Section 5.

2. **Sample generation and preference prediction:** The RNN is used to generate new samples for each option-set corresponding to $d_1, \ldots, d_N$. Each sample in option-set i is generated by cycling through the network until the maximum preference exceeds the deliberation preference threshold τ_i. For each sample, the selected choice is computed by applying **argmax** to the final preferences.

3. **Alignment**: An alignment between each sample in the data set and one in the new sample set is obtained as described above.

4. **Parameter update**: Hinge error for all samples is computed using Equation 6 and finally personal evaluation values $\mathbf{M}$ and/or attention weights $\mathbf{w}$ are updated accordingly using back-propagation algorithm. Once the $\mathbf{M}$ are computed, the $\mathbf{S}$ values are updated accordingly.

7 Experimental Results

In this section we report experimental results obtained from synthetic datasets generated from MDFT models.

Data Generation We generated 4 datasets each containing 10 problems. In each problem a set of options of size n_o is randomly generated using a uniform distribution in $[1, 10]$ and then n_s distinct sub-sets of size 3 are chosen as option-sets. Using these option-sets we create MDFT models with parameters generated randomly according to uniform distributions in $[0.002, 0.05]$, $[1, 20]$, $[0, 1]$, $[0.3, 0.7]$ for ϕ_1, ϕ_2, b, σ^2, and w, respectively. Given an MDFT model, we use it alongside a random threshold sampled uniformly from interval $[5, 15]$ to obtain the final choice distributions and create the *(choice, deliberation preference threshold)* pairs as defined in Section 2. Table 2 summarizes the characteristics of the datasets.

Baseline Method [17] provide the likelihood function formula for choosing an option from a set based on the MDFT model when the deliberation time is known. Based on this, we develop a method to use as a comparison to the RNN-based approach.

Dataset	# options (n_o)	# sub-sets (n_s)
#1	5	10
#2	7	10
#3	10	10
#4	10	20

Table 2: Four different datasets are generated with different option sizes. From each set of options, n_s sub-sets with size 3 are selected.

Given the likelihood function, it is possible to analytically obtain the parameter estimates from data using, for example, a maximum likelihood approach. This is not applicable when the deliberation time is unknown. As an alternative, we use a simulation method to estimate the likelihood function and a general purpose optimizer to find the maximum likelihood estimate for **w** and **M** parameters. We denote this approach by **MLE**.

Implementation We use the *Pytorch* library [14] to implement the Recurrent Neural Network described in Section 6, *RMSprop* optimizer [7] with learning rate 0.005, and multi-margin hinge loss with margin 0.01. We train each model for 150 iterations using 100 samples per MDFT in each iteration. We also stop training early whenever the loss error drops under 10^{-5}.

MLE is implemented in MATLAB and pre-compiled to speed-up the calculations. MATLAB function *fminsearch* is used to find the parameter estimates that maximize the likelihood function. We set the maximum number of iterations to 1000, and of simulations to 10^4. Initial values are randomly generated.

After training is completed, the learned models are used to produce the choice distributions over the options (denoted with $h_{\hat{\mathcal{M}}}$ in Section 5).

7.1 Learning Personal Evaluations M

In Table 3 we show results comparing the choice distributions of the learned models with the ones produced by the original MDFTs using distance D_{js} (defined in Section 5). As it can be seen, the **RNN** method consistently outperforms **MLE** by a wide margin.

Dataset	**RNN**		**MLE**	
	mean	std	mean	std
#1	0.012	0.015	0.103	0.129
#2	0.011	0.007	0.149	0.140
#3	0.012	0.012	0.173	0.091
#4	0.011	0.009	0.104	0.072

Table 3: Mean and standard deviations of Jensen-Shannon distance D_{js} between choice distributions for estimating **M**.

We recall that our primary goal is to learn an MDFT model capable of generating a decision making behavior similar to the one observed. However, since our data is generated starting from MDFTs, we also consider the similarity of the orderings of the options, with respect to each attribute, between the ground truth **M** matrix and in the learned matrix $\hat{\mathbf{M}}$. To do this we consider Kendall's τ ranking correlation coefficient [10]. This measure quantifies the distance between two rankings as the ratio of the number of pairwise miss-orderings to the number of all possible pairs. Table 4 shows the results of this coefficient on the datasets. As it can be seen, our **RNN** comes very close to recovering the initial rankings for options. It is interesting to observe that, even when the information about the relations between the options is sparse as in dataset #3 and #4, **RNN** is able to recover a

significant percentage of the ordering between options, while **MLE** under-performs in this respect.

Dataset	**RNN**		**MLE**	
	mean	std	mean	std
#1	0.220	0.121	0.305	0.137
#2	0.305	0.086	0.348	0.122
#3	0.298	0.073	0.420	0.051
#4	0.309	0.037	0.369	0.109

Table 4: Mean and standard deviations of Kendall's τ distance between the estimated $\hat{\mathbf{M}}$ and the actual $\mathbf{M}$.

7.2 Learning Attention Weights w

We recall that attention weights represent the importance given to each attribute. We use the Jensen-Shannon distance between the estimated attention weights and the original ones to measure the quality of the learned parameters. The results, depicted in Table 5, show how both methods are able to fully recover the attention weights. Similar results (which we omit for the sake of space) hold for the divergence between the predicted choices and actual choices. As expected, our results show that learning attribute importance is an easier task than recovering personal evaluations.

Dataset	**RNN**		**MLE**	
	mean	std	mean	std
#1	5.4e-5	3.0e-5	2.9e-5	1.5e-5
#2	6.8e-5	3.7e-5	3.0e-5	1.5e-5
#3	4.2e-5	2.7e-5	3.0e-5	1.6e-5
#4	3.0e-5	1.5e-5	2.7e-5	1.0e-5

Table 5: Mean and standard deviations of Jensen-Shannon distance D_{js} between the estimated $\hat{\mathbf{w}}$ and the actual $\mathbf{w}$.

7.3 Learning Personal Evaluations M and Attention Weights w simultaneously

The results shown in Table 6 demonstrate that **RNN** is able to find a combination of personal evaluations $\mathbf{M}$ and attention weights $\mathbf{w}$ in such a way that the resulting MDFT produces a choice distribution very close to the original MDFT. In this respect it outperforms **MLE**. However, we observe that, for both methods, these results don't necessarily guarantee convergence of the individual parameters, as can be seen for $\mathbf{w}$ in Table 7. In fact, the increased number of free parameters results in more combinations generating similar choice distributions.

We conclude this section with results comparing the two methods in terms of time efficiency. As it can be seen in Table 8, **RNN** is faster than **MLE** when learning personal evaluations $\mathbf{M}$ alone or jointly with $\mathbf{w}$, whereas the opposite holds when only attention weights $\mathbf{w}$ are learned. An explanation of this is that the task of learning $\mathbf{w}$ is simpler and that the overhead involved in the **RNN** method infringes its competitiveness in this case.

Dataset	RNN		MLE	
	mean	std	mean	std
#1	0.011	0.018	0.023	0.029
#2	0.012	0.011	0.169	0.149
#3	0.014	0.016	0.169	0.190
#4	0.010	0.008	0.097	0.084

Table 6: Jensen-Shannon distances D_{js} between choice distributions for estimating $\mathbf{M}$ and $\mathbf{w}$.

Dataset	RNN		MLE	
	mean	std	mean	std
#1	0.070	0.050	0.029	0.042
#2	0.040	0.041	0.022	0.026
#3	0.051	0.045	0.019	0.030
#4	0.040	0.036	0.019	0.021

Table 7: Mean and standard deviations of Jensen-Shannon distance D_{js} between the estimated $\hat{\mathbf{w}}$ and the actual $\mathbf{w}$.

Dataset	RNN			MLE		
	M	w	Mw	M	w	Mw
#1	69.1	29.7	89.0	109.6	7.8	115.4
#2	143.4	19.7	132.2	330.6	5.2	251.7
#3	64.3	14.1	114.0	242.6	3.8	195.8
#4	190.3	73.1	167.6	368.5	7.9	472.7

Table 8: Average training time in seconds. Where M, w, and Mw denote, respectively, learning $\mathbf{M}$, $\mathbf{w}$, and both $\mathbf{M}$ and $\mathbf{w}$.

8 Conclusion and Future Work

Current preference learning approaches build on assumptions that are sometimes in contrast with cognitive models of decision making. In this paper we present a method that combines a cognitive model of choice with recurrent neural networks to learn the decision maker's preferences from his final choices. We evaluate its performance and compare it with another method drawn from the state of the art on MDFT. Results show that the RNN-based method is effectively and efficiently learning preferences and attributes' importance which are compatible with ground-truth behavior.

In the future, we plan to address the issue of parameter convergence when both personal evaluation and attention weights are learned jointly by constraining personal evaluations to specific classes of functions (such as sigmoids or exponentials). We also plan to extend our experimental study to include behavioral data. Finally we will investigate online learning scenarios where parameters are updated as single choices are observed.

References

[1] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. Advances in optimizing recurrent networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8624–8628. IEEE, 2013.

[2] L. Bergen, O. Evans, and J. Tenenbaum. Learning structured preferences. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 32, 2010.

[3] J. R. Busemeyer and A. Diederich. Survey of decision field theory. *Mathematical Social Sciences*, 43(3):345–370, 2002.

[4] J. R. Busemeyer and J. T. Townsend. Decision field theory: a dynamic-cognitive approach to decision making in an uncertain environment. *Psychological review*, 100(3):432, 1993.

[5] G. De Soete, H. Feger, and K. C. Klauer. *New developments in probabilistic choice modeling*. North-Holland, Amsterdam, 1989.

[6] J. Fürnkranz and E. Hüllermeier. *Preference learning*. Springer, 2010.

[7] G. Hinton, N. Srivastava, and K. Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.

[8] J. M. Hotaling, J. R. Busemeyer, and J. Li. Theoretical developments in decision field theory: Comment on tsetsos, usher, and chater (2010). 2010.

[9] R. L. Keeney and H. Raiffa. *Decisions with multiple objectives: Preference and value tradeoffs*. Wiley, New York, 1976.

[10] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[11] F. Koriche and B. Zanuttini. Learning conditional preference networks. *Artif. Intell.*, 174(11):685–703, 2010.

[12] S. Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[13] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

[14] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[15] L. D. Raedt, A. Passerini, and S. Teso. Learning constraints from examples. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, pages 7965–7970. AAAI, 2018.

[16] T. Rahgooy and K. B. Venable. Learning preferences in a cognitive decision model. In *proceedings of Joint Workshop on Human Brain and Artificial Intelligence, HBAI2019. To appear in Springer-Nature Communications in Computer and Information Science (CCIS).*, 2019.

[17] R. M. Roe, J. R. Busemeyer, and J. T. Townsend. Multialternative decision field theory: A dynamic connectionst model of decision making. *Psychological review*, 108(2):370, 2001.

[18] F. Rossi and A. Sperduti. Learning solution preferences in constraint problems. *J. Exp. Theor. Artif. Intell.*, 10(1):103–116, 1998.

[19] F. Rossi, K. Venable, and T. Walsh. *A Short Introduction to Preferences: Between Artificial Intelligence and Social Choice*. Morgan and Claypool, 2011.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[21] L. L. Thurstone. *The measurement of values*. University of Chicago Press, 1959.