Modeling Risky Choices in Unknown Environments: Supplement

Ville Tanskanen VILLE.TANSKANEN@HELSINKI.FI
Chang Rajani CHANG.RAJANI@HELSINKI.FI
Department of Computer Science, University of Helsinki, Finland

Homayun Afrabandpey HOMAYUN.AFRABANDPEY@NOKIA.COM
Nokia, Finland

Aini Putkonen AINI.PUTKONEN@AALTO.FI
Aurélien Nioche NIOCHE.AURELIEN@GMAIL.COM
Department of Communications and Networking, Aalto University, Finland

Arto Klami ARTO.KLAMI@HELSINKI.FI
Department of Computer Science, University of Helsinki, Finland

This supplementary material provides the technical details used for the experiments of the main paper, using Section, Table and Figure numbering that does not overlap with the main manuscript to avoid confusion.

Sections 8-11 provide additional details for Experiments 1-3 carried out on synthetic data. Section 8 provides description of the neural network architectures and Section 9 describes the training details. Section 10 provides detailed specification of the simulated decision-making environment used in all experiments. Section 11 provides the details for the cumulative prospect theory (CPT) model used as example of white-box risk model, including high-level description on how its parameters can be interpreted.

Section 12 describes additional experiment to study the effect of the number of pseudo-labels to accuracy of the environment model. Section 13 provides the details for the case study on Counter-Strike: Global Offensive (CS:GO), explaining e.g. data acquisition, additional details about the game, and full description of the way the data was mapped into lottery sets.

We also provide full code implementing all models and all experiments as conducted in the main paper [1]. The source code package also includes the result files of the experiments.

8. Network Architectures

The neural networks used in the experiments were the same regardless of the experiment and the environment dimension.

For the MONDE, we used a network that had two hidden layers consisting of 64 and 63 units respectively for the feature network (the part of the network taking features x as its input), both followed by tanh-activation. The output of the second layer is concatenated with y to form a 64-dimensional hidden layer that is mapped to 32-dimensional layer, again

1. `https://version.helsinki.fi/MUPI/modeling-risky-choices-in-unknown-environments`

© 2021 V. Tanskanen, C. Rajani, H. Afrabandpey, A. Putkonen, A. Nioche & A. Klami.

with tanh activation, and then to a scalar with sigmoid transformation to produce the value of the CDF for the specific y. During training, y is the true outcome. For test instances – when using MONDE to provide the outcome representation $q_{\boldsymbol{p}}$ for new features x – the model is used by feeding in a sequence $\boldsymbol{y}$ of possible outcomes, each at a time, to compute the CDFs $P(Y < y|x)$ for all. We used a regular grid $\boldsymbol{y}$ of 200 points over the interval $[y_{min}, y_{max}]$, where the limits correspond to the smallest and largest outcome in all data. The actual output representation $q_{\boldsymbol{p}}$ is then formed by matching the probabilities $P(Y < y|x)$ with a sequence of probabilities $\boldsymbol{p}$ using nearest neighbour. The number of quantiles used for representing the CDF are provided in Table 4 for each experiment separately – we used 32 for others but 50 for the first one, but could as well have used the same number for all.

For the other two outcome representations we used a standard feedforward network with two hidden layers of 64 and 32 units, both with the ReLU activation. The 32-dimensional hidden vector was used as distributed outcome representation $f(x)$, and the point prediction $\hat{y}$ was constructed by mapping $f(x)$ to single dimension with linear mapping and identity activation.

The RNN architecture used two RNNs as described in Section 5.2. The first hidden layer had the hidden representation size $h = 10$, it was unidirectional and used ReLu-activations. We used the sum of the first RNN's last hidden layer as the initial hidden reprsentation h_0 for the second RNN. Whenever we used a packed sequence (during training) we had to pad the sequence in order to make the softmax work correctly over varying length sequences. Note, however, that the padding value is now easy to choose, as we can be certain that 0 is the smallest number a hidden state can hold, because of the ReLU activations. We used -2 as the padding value, but any negative value less than this creates negligible probabilities.

9. Details of Training Networks

All models were trained using standard stochastic gradient methods, in particular using the Adam optimizer as implemented in PyTorch. The hyperparameters used for the training are provided in Table 4, which lists for all experiments the number of epochs N_e, the batch size bs, the initial learning rate lr.

Pseudo-labeling in Experiment 3 was done so that the number of pseudo-labels linearly increased (over the training iterations) from 1 to 3, replacing the same number of random true observations in each batch (of 32 samples in total). For each pseudo-label we sampled the value from a quantile selected from uniform distribution between 0.3 and 0.5.

Throughout the experiments, each individual model (be it MONDE, RNN or CPT) was fast to train, typically taking only a few minutes on standard desktop or laptop machines. The exact time depends on the amount of data in conventional manner, and more detailed analysis of the computational cost is of no particular interest for this work. However, we note that for some of the experiments we average the results over relatively large number of repetitions, which makes the total computational cost larger.

Exp.	MONDE (N_e, bs, lr)	CPT (N_e, bs, lr)	RNN(N_e, bs, lr)	CDF len
Exp. 1	(150, 32, 0.001)	(50, 16, 0.02)	-	50
Exp. 2	(250, 32, 0.001)	(250, 32, 0.001)	(200, 32, 0.01)	32
Exp. 2(policies)	(250, 32, 0.001)	(250, 32, 0.001)	(200, 32, 0.01)	32
Exp. 3(c. budget)	(250, 32, 0.001)	(100, 16, 0.02)	-	32

Table 4: Training parameters for the experiments. N_e stands for the number of epochs, bs for batch size and lr for the initial learning rate of the optimizer (Adam). "CDF len" is the length of quantile encoding $q_{\boldsymbol{p}}$ for quantile encoding $\boldsymbol{p} = [q(p_1), \dots, q(p_{\text{CDF len}})]$ for MONDE. CPT init/true dist(p/λ) denotes the distribution of which the unconstrained parameters for ground truth CPTs or initial values for CPT training were sampled from. s_p denotes the parameters of sigmoid activation.

10. Environments of the Experiments

As described in the main paper, we generated the true outcomes always from the environment model

$$y = \frac{5}{\pi} \arcsin\Big(\sin(\frac{\pi}{2}\beta_2^T x)\Big) + \exp(\cos(\beta_1^T x) + 0.2)z,$$

where $z \sim N(0, 1)$. The first element of x is here fixed to 1, so that the corresponding weight models a bias term. Next we provide the exact parameter vectors β_1 and β_2 that were used in the experiments to facilitate reproducability.

Experiment 1 used

$$\beta_1 = [-0.6023, -1.3410]$$
$$\beta_2 = [0.0, -1.5583],$$

and the same parameters were used for creating Figure 2. For Experiment 2 and the pseudo-labeling part of Experiment 3 we used

$$\beta_1 = [0.1081, -0.4376, -0.7697]$$
$$\beta_2 = [-0.1929, -0.3626, -2.8451],$$

and for the shared-data part of Experiment 3 we used still a bit more complicated environment with

$$\beta_1 = [-0.3720, -0.6967, 0.5324, -0.0640, -0.3433]$$
$$\beta_2 = [0.3132, -1.1290, -1.2232, -0.2313, 0.9067].$$

11. Cumulative Prospect Theory details

For training the CPT, we optimized the parameters in unconstrained space, so that the unconstrained parameters were α_u, β_u, λ_u, γ_u^+ and γ_u^-. We map the unconstrained parameters by sigmoid mapping $c(p) = \frac{1}{1+\exp(-p)}$ for parameters $p \in \{\alpha_u, \beta_u, \gamma_u^+, \gamma_u^-\}$, and softplus mapping $\lambda = \text{softplus}(\lambda_u)$ for λ_u, to acquire the constrained versions of the parameters. The initial, as well as the ground truth (unconstrained) CPT parameter values, were sampled from Gaussian distributions. The λ parameter was sampled from $\lambda \sim |N(0,1)| + 1$ [2]and other parameters were sampled from $N(0,1)^2$.

Given the formulation of the main paper, the parameters of a CPT profile can be interpreted as follows:

- $\alpha < 1$: concave utility function in the gains,
- $\beta < 1$: convex utility function in the losses,
- $\lambda > 1$: the higher the value, the more losses hurt relative to the same amount of gain,
- $\gamma^+ < 1$: small probabilities are overestimated, and large ones underestimated in gains and
- $\gamma^- < 1$: small probabilities are overestimated, and large ones underestimated in losses.

Experiment 2 and Experiment 3 (pseudo-labels) used a generating CPT with parameter set $\theta = [0.7311, 0.6226, 10.00, 0.6225, 0.7047]$ describing a loss averse agent.

12. Additional Experiment on Pseudo-labels

We linearly increased the number of pseudo-labels from 1 to 3 in batches of size 32 in the Experiment 2. Another experiment was conducted that studies the effect of the number of pseudo-labels per batch to the accuracy of the environment model. Figure 7 shows the effect of number of pseudo-labels in a setup equal to Experiment 2 in the main paper, presenting average over 40 repetitions of the experiment. It compares the true distribution of the environment and the estimated distribution through mean absolute errors between quantiles $q \in \{0.05, 0.5, 0.95\}$ over randomly selected locations x. Formally, the measure is $\left|P^{-1}_{Y|x}(q) - \hat{P}^{-1}_{Y|x}(q)\right|$, where $P^{-1}_{Y|x}(q) = \inf\{y : P(Y < y|x) \geq q\}$ is the quantile function of the true environment and $\hat{P}$ refers to the estimated envrionment model. We used batch size of 32 to train the MONDE of which a number (x-axis) of samples were replaced with a pseudo-labels. The horizontal lines spanning the $[1, 3]$ region depicts the MAE when the number of pseudo-labels was linearly increased during training from 1 to 3. The choices that determine which outputs we obtain the true outputs were generated using CPT heuristic, just like in Table 2 of the original paper.

The result indicates that modest amount of pseudo-labels is beneficial for the environment model and that the choice of using $1-3$ pseudo-labels per batch explained in the main paper is consistently better than not using any. However, including too many pseudo-labels decreases the performance, and we recommend carefully validating the choice in practical applications.

Finally, we make a remark on the generality of the proposed pseudo-labeling approach. The method described in the main paper requires the environment model to have a probabilistic internal representation $\hat{P}(Y|x)$ because of the inverse transform sampling. The

2. Experiment 3 (constant budget) had standard deviation 1.1 instead of 1.

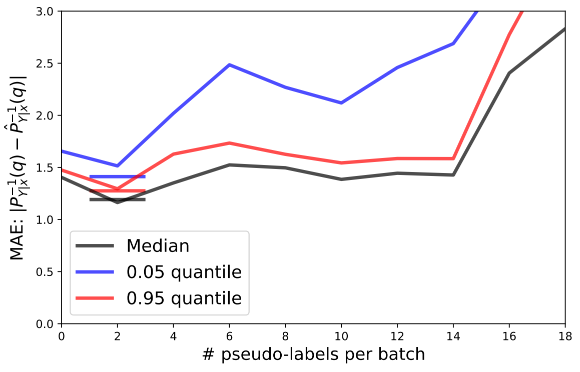

Figure 7: Supplementary experiment on pseudo-labels' effect on the environment model accuracy. Modest amounts of pseudo-labels suggest slightly increased performance. Adding more pseudo-labels hinders training, as majority of the samples do not carry real information.

framework, however, generalizes also for other models. For models without internal representation for $P(Y|x)$ we need to estimate what 'low' means from the data itself, for example using empirical estimate for the unconditional distribution $p(y)$ of the observed outcomes. We can then use random low quantiles of $p(y)$ for pseudo-labeling, or select some low quantile as explicit prior for the model itself.

13. Case Study Details

The game of GS:GO is played by taking one of two sides: the terrorists and the counter-terrorists. The terrorists aim to plant a bomb at one of two sites, which the counter-terrorists need to stop. At round 15 the teams are switched, so that both teams get to play both sides. At the beginning of each round the team decides which items to purchase from the in-game shop using currency earned during previous rounds. The in-game economy is fairly simple, with players getting money from kills and planting the bomb, as well as bonuses for winning a round, or losing many rounds in a row. We model the risk behavior of teams on the level of the decisions the teams make regarding investing the in-game money before each individual round. Note that we do not model risk behavior within the actual rounds that involve continuous real-time decisions of all individual players.

Data: We downloaded the last 100 matches for two competitive teams *Astralis* (a team that retained the highest ranking for a long time) and *Gambit Esports* (the team at the top of the ranking currently) from https://www.hltv.org/, so that for all matches the five players for each team were the same. Each match consists of 2-5 *maps*, each consisting of up-to 30 *rounds*, with the first team to win 16 rounds winning the game. Some maps had to be discarded due to errors in processing the game. For both teams we had approximately 3,200 lottery sets. Each half of the game starts with a "pistol round", where no major decisions can be made; these rounds were discarded from the data. We also removed the round before halftime, as well as the very last round, as the next round's equipment value (the pistols) does not reflect the outcome of the choices made. We did not include overtime rounds, as the teams receive full money at the beginning of each overtime, and thus always take a full buy.

Lottery sets: For the decisions, we consider the following the discretization bins given by *HLTV.org*:

- *Full eco* ($< 5k\$$) means that the team is very short on money, and decides to save money for the next round by buying almost nothing this round.
- *Semi eco* ($< 10k\$$) means that the team typically buys some pistols and kevlar, leaving sufficient money for the next round
- *Semi buy* ($< 20k\$$) means that the team cannot afford all the rifles and equipment they want, but decide to take a chance and try to win with worse equipment
- *Full buy* ($> 20k\$$) means the team can afford to buy everything they can. This is the most common decision in the data.

As inputs X for the environment model we use 12 features that describe the current game situation. The features are:

```
round, is_terrorist, won last, team previous eq. value,
enemy previous decision, team score, opponent score, team map point,
opponent map point, team eq. value roundstart, team money,
team decision available
```

where the last one encodes the different options available for the team depending on their current amount of money. Note that some features, such as *enemy previous decision* are not directly available to the players but we assume that professional teams can easily deduce them based on what they see during the previous round. We encode the options as ordinal nunbers, so that "eco" maps to the value 1.0 and "full-buy" to 4.0. All features were normalized to lie between 0 and 1 and the outcome was normalized to have zero mean and unit standard deviation. This feature representation was constructed as a reasonable attempt to characterize the game state, using features commonly used by e-sports commentators and analysts, but we note that it could be further improved for more detailed analysis of the risk behavior.

Learning: The environment model MONDE used the architecture described in Section 8 and was trained for 500 epochs with batch size 1024 and learning rate 0.1. The decision model was trained for 350 epochs with batch size equal to the size of the training data and learning rate of 0.01. To increase robustness for stochastic optimization, we re-trained the decision model 10 times and selected the one with the best internal objective. The RNN model used as predictive baseline had 32 hidden units and was trained for 200 epochs with 1024 batch size and learning rate of 0.02. The length of quantile vector $\boldsymbol{q_p}$ used as input for the decision model was 32.