# RevMan: Revenue-aware Multi-task Online Insurance Recommendation

Yu Li,[1] Yi Zhang, [2] Lu Gan, [2] Gengwei Hong, [2] Zimu Zhou, [3] Qiang Li [1,4*]

[1] College of Computer Science and Technology, Jilin University
[2] WeSure Inc.
[3] School of Information Systems, Singapore Management University
[4] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University
yuli19@mails.jlu.edu.cn, {jamesyzhang, matrixgan, weberhong}@wesure.cn, zimuzhou@smu.edu.sg, li_qiang@jlu.edu.cn

## Abstract

Online insurance is a new type of e-commerce with exponential growth. An effective recommendation model that maximizes the total revenue of insurance products listed in multiple customized sales scenarios is crucial for the success of online insurance business. Prior recommendation models are ineffective because they fail to characterize the complex relatedness of insurance products in multiple sales scenarios and maximize the overall conversion rate rather than the total revenue. Even worse, it is impractical to collect training data online for total revenue maximization due to the business logic of online insurance. We propose RevMan, a **Rev**enue-aware **M**ulti-t**a**sk **N**etwork for online insurance recommendation. RevMan adopts an adaptive attention mechanism to allow effective feature sharing among complex insurance products and sales scenarios. It also designs an efficient offline learning mechanism to learn the rank that maximizes the expected total revenue, by reusing training data and model for conversion rate maximization. Extensive offline and online evaluations show that RevMan outperforms the state-of-the-art recommendation systems for e-commerce.

## Introduction

Technological advances such as artificial intelligence and mobile computing have accelerated the digitization of the insurance industry. Online insurance, which sells insurance products via mobile apps, has emerged as a new sales mode to reach a wide range of customers with prompt and flexible services. Reports predict an exponential growth in the global online insurance market (Mordor 2019).

To provide personalized insurance products to customers without overwhelming their attention, online insurance companies such as Tencent WeSure (Tencent 2017), Acko General Insurance (Acko 2017), PingAn JinGuanjia (PingAn 2016) etc. often group their complex and diverse insurance products into *sales scenarios*, each targeting a different customer category. Fig. 1 shows an example of four sales scenarios in the mobile app of an online insurance company. It is crucial that the recommended insurance products in all the sales scenarios will convert to actual buying as much as possible to maximize the *total revenue*.

*Corresponding author.

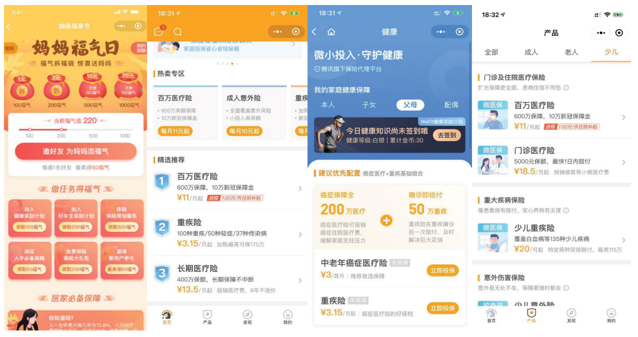
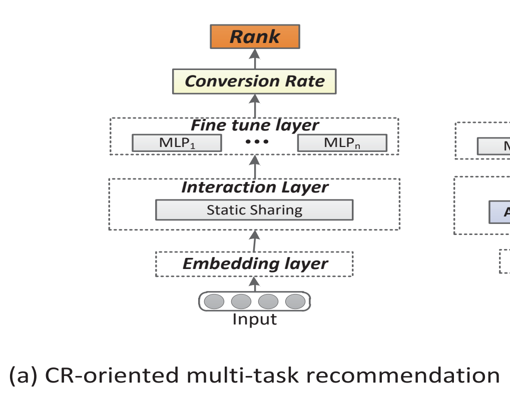

(a) activity (b) special sale (c) parent products (d) children products

Figure 1: Screenshots of sales scenarios in the mobile app of an insurance company: (a) activity page for new users; (b) special sale page for loyal customers; (c) products customized for parents; (d) products customized for children.

Compared with generic recommendation for e-commerce, designing an effective recommendation strategy for online insurance products encounters unique challenges due to the product complexity and business logic of online insurance. On one hand, data sparsity is severe in online insurance because of the common misunderstanding of the general public on insurance products (Bi et al. 2020). Compared with other daily goods, there are considerably fewer trading data of online insurance products to train an accurate recommendation model. Even worse, the few (compared with the amount of different products and customers) trading data available come from different sales scenarios. The data distributions of different sales scenarios often vary, making the training dataset for each sales scenario even smaller. On the other hand, there is a lack of training data for maximizing the total revenue. In the historical trading data, insurance products are ranked according to the expected conversion rate, thus failing to reflect the actual buying if the products are ranked based on the expected revenue. Nevertheless, it is expensive or impractical to collect new training samples where products are ranked for this goal due to the business logic of online insurance companies.

A natural solution to data sparsity is to jointly train multiple correlated tasks (sales scenarios in our case) via multi-task learning (Ruder 2017). Multi-task learning proves ef-



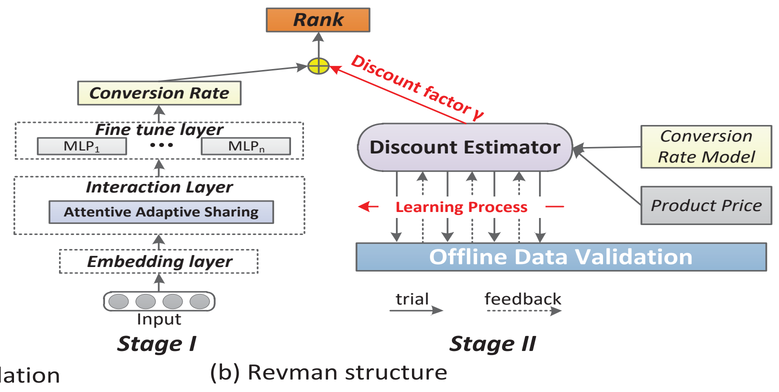


Figure 2: An illustration of previous CR-oriented multi-task recommendation models and RevMan. (a) Typical structure of existing multi-task recommendation models that maximize the conversion rate. (b) Workflow of RevMan. At stage I, RevMan utilizes a selective attentive multi-task network (SAMN) to learn feature representations of different tasks for modeling conversion rate. At stage II, RevMan learns from offline data the ranks that maximizes the total revenue via a discount factor $\gamma$. It conducts trials on the offline data and utilizes the results as feedbacks to optimize $\gamma$. The products are re-ranked based on the product of $\gamma$ and conversion rate to maximize the total revenue.

fective in recommendation systems for videos (Ma et al. 2018a), web search (Bai et al. 2009), online advertising (Pan et al. 2019), etc. It is also promising for online insurance recommendation because insurance products and sales scenarios are correlated. For example, a customer who buys a car accident insurance may also buy a health insurance. Moreover, over 40% of potential customers place high priority to insurance products of children and parents, which are often listed in different sales scenarios (Mordor 2019).

However, prior multi-task learning proposals for online product recommendation are sub-optimal for online insurance recommendation. *(i)* The feature sharing mechanisms in previous studies (Hu et al. 2018; Ma et al. 2019; Pan et al. 2019; Ma et al. 2018b; Wang et al. 2019a) fail to model the complex task relatedness in insurance products. Existing multi-task recommendation systems mainly adopt a three-layer design: an embedding layer, an interaction layer and a fine tune layer (Ruder 2017; Zhang and Yang 2017), as shown in Fig. 2(a). Feature sharing takes place either in the embedding layer (Ma et al. 2018b; Wang et al. 2019a) or in the interaction layer using fixed weights for each task pair, *i.e.*, static feature sharing (Hu et al. 2018; Ma et al. 2019; Pan et al. 2019). Feature sharing in the embedded layer enforces only simple, low-level feature sharing, while static sharing cannot characterize the diversity in task relatedness due to difference in user behaviours across sales scenarios (Yosinski et al. 2014). *(ii)* Most existing online product recommendation systems focus on conversion rate, rather than the total revenue (Kitada, Iyatomi, and Seki 2019; Bai et al. 2009; Chapelle et al. 2010; Zhang et al. 2019). Recommending high conversion-rate products does not necessarily maximize the total revenue, because products with high conversion rates seldom have high prices (Zhang et al. 2016).

To this end, we design RevMan, the first **Rev**enue-aware **M**ultit**a**sk recommendation **n**etwork for online insurance recommendation. As mentioned before, it is impractical to collect new training samples for total revenue in vivo. We can only rely on historical data ranked according to the conversion rate. We tackle this problem by transforming the objective of maximizing the total revenue to maximizing a modified version of conversion rate. Specifically, we assign a discount factor $\gamma$ to each conversion rate such that ranking products based on the discounted conversion rate is equivalent to ranking products based on the expected revenue. Then a recommender system that maximizes the (discounted) conversion rate also maximizes the total revenue. To realize this idea, we need to address the questions below.

- *How to learn an effective multi-task model for conversion rate from sparse historical data of multiple sales scenarios?* Our solution is SAMN, a selective attentive multi-task network to maximize the conversion rate. Instead of feeding the same training sample for all tasks as in previous studies, it adaptively utilizes the embedding vector of the training sample for different tasks.
- *How to define the discount factor $\gamma$ such that maximizing the discounted conversion rate also maximizes the total revenue?* Since it is difficult to derive a closed-form mapping from the conversion rate to the total revenue, we learn the discount factor via offline estimation. Specifically, we simulate revenue-based ranking by assigning different $\gamma$ to the conversion rate and test the new rankings on the historical data to collect revenue estimations. This forms a sequential decision process and thus we adopt reinforcement learning to learn the policies to assign discount factors that maximize the total revenue.

Our main contributions and results are as follows:

- To the best of our knowledge, RevMan is the first recommendation system that aims to maximize the total revenue for online insurance products.
- We design a two-stage model which *(i)* optimizes the conversion rates from sparse training data of multiple sales scenarios and *(ii)* learns to maximize the total revenue without collecting revenue-ranked training data.

- We evaluate RevMan via extensive offline and online experiments on real-world online insurance platforms. Offline experimental results show an improvement of 0.62% in average AUC and 7.05% in revenue over the state-of-the-arts (Mcmahan et al. 2013; Cheng et al. 2016; Ma et al. 2018b,a,a). A 7-day real-world A/B test also validates the effectiveness of RevMan in both modeling conversion rate of all tasks and increasing the total revenue.

## Primer on Total Revenue

Before diving into the details of design, we first present the primer on total revenue. In online retailing business, the total revenue is typically calculated based on the Gross Merchandise Volume (GMV), which is the total dollars of sold products in expectation (Bi et al. 2020; Pei et al. 2019):

$$Rev = \sum_{\forall i,j} I^j(i)_{CR} * CR^j(i) * P(i) \tag{1}$$

where $I$, $CR$ and $P$ are the impression, conversion rate and price of product $i$ in task $j$, respectively. By default, we treat $CR^j(i)$ as the average conversion rate for impressed product $i$ over all users, and set $P(i)$ identical for different tasks (Pei et al. 2019). $I_{CR}$ is the impression based on the rank of $CR$ and $I^j(i)_{CR}$ is the summation of $I^j_u(i)_{CR}$ on all user $u$. For different ranks, the conversion rate of the same product may vary due to position bias, leading to different $CR^j(i)$(Guo et al. 2019). For each user $u$, $I^j_u(i)_{CR}$ is defined as:

$$I^j_u(i)_{CR} = \begin{cases} 1, & rank(CR^j_u(i)) \leq M^j \\ 0, & otherwise \end{cases} \tag{2}$$

where $M^j$ is the number of impressed products in task $j$ and $rank$ is the reverse sort algorithm. As mentioned before, a task refers to a recommendation task in each sales scenario.

We make two observations from Eq. (1) and Eq. (2).

- A recommendation model that maximizes the conversion rates $\{CR^j_u(i)\}$ may not maximize the total revenue because *(i)* the relationship between conversion rates and total revenue is complex and *(ii)* the impression is based on the ranks of conversion rates rather than revenue.
- A model that maximizes the total revenue may be learned through samples collected for maximizing the conversion rate. This is because the rank of each product can be a bridge to estimate the relationship between $I$ and $CR$, *i.e.*, we can obtain $I^j_u(i)$ for each rank result on $CR^j_u(i)$. If we can simulate different ranks with $CR^j_u(i)$, then we can obtain a series of corresponding $E[Rev]$ to find the way to maximize the total revenue.

To obtain an effective model that maximizes the total revenue on basis of a model that optimizes the conversation rate, we need to *(i)* first model conversion rate for different tasks and *(ii)* design a method to learn the optimal ranks, which leads to the design of RevMan.

## RevMan Design

### Overview

RevMan consists of two stages to sequentially *(i)* model the conversion rate for different tasks and *(ii)* utilize the conversion rate to estimate the maximized revenue (see Fig. 2(b)).

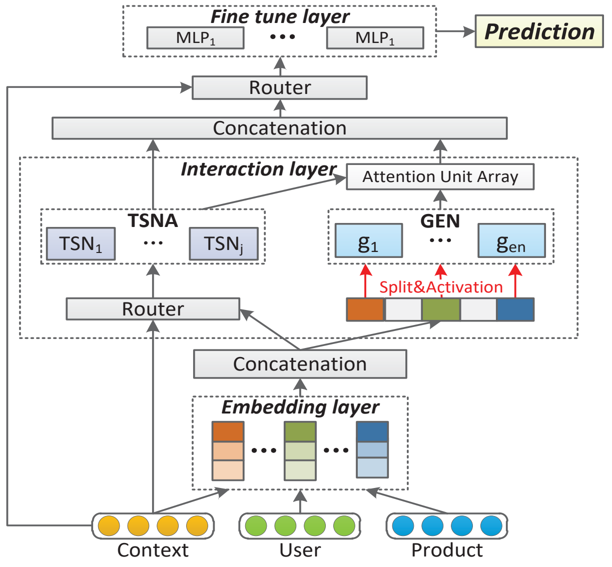


Figure 3: Architecture of Selective Attentive Multi-task Network (SAMN). Its key novelty is an attentive adaptive feature sharing mechanism. Specifically, an attention unit array is used to control the knowledge sharing flow among tasks based on the input from each task.

- **Stage I: SAMN.** We design a selective attentive multi-task network (SAMN) to adaptively share feature embedding among different tasks. SAMN makes full use of the sparse data by modeling the complex task relatedness and sharing useful feature representations among tasks.
- **Stage II: Discount Estimator.** The discount estimator learns a discount factor $\gamma$ on $CR$ by simulating the impact of ranks such that the total revenue is maximized.

### SAMN Design

SAMN adopts the three-layer architecture in mainstream multi-task recommendation systems (Ruder 2017; Zhang and Yang 2017), which consists of an embedding layer, an interaction layer and a fine tune layer. SAMN mainly differs from prior studies in the interaction layer (see Fig. 3).

**Embedding Layer** It converts raw samples into embedding vectors. A raw sample $i$ from task $j$ is a triad $\hat{x}^j_i = \{u, prd, ctx\}$. $u$ is user related features such as age, sexuality, and interest features, such as buy history, view records, etc. $prd$ is the product features such as name, type, coverage, etc. $ctx$ is the contextual features such as time, scenario ID, impression position, etc. Notice that the impression position is used in training while hold out for prediction to alleviate bias (Guo et al. 2019). Scenario ID is also used in the following layers to decide the activation route. We can write $\hat{x}^j_i$ as a multi-hot vector:

$$\hat{x}^j_i = [\underbrace{0, 0, ..., 1}_{user}, \underbrace{0, 1, ..., 0}_{product}, \underbrace{0, 1, ..., 1}_{context}] \tag{3}$$

Then the embedded vector $x^j_i$ can be calculated as:

$$x^j_i = \hat{x}^j_i * emb^T \in \mathbb{R}^{D \times K} \tag{4}$$

where $*$ is element-wise multiplication, $emb \in \mathbb{R}^{D \times K}$ is the embedding dictionary, $D$ is the feature space size and $K$ is the embedding vector length. As in previous studies (Ma et al. 2018b,a), we share the embedding dictionary across different tasks given overlapped feature space.

**Interaction Layer** This layer consists of two main feature learning subnetworks. One is the task specific network array (TSNA) and the other is general expert network (GEN). TSNA learns task-specific feature representations, while GEN learns general feature representations.

TSNA is an array of task-oriented multi-layer perceptrons (MLPs) to learn the specific feature representations for each task. By default, we set the number of MLPs to the task number based on the fact that the data distribution differs among tasks. Accordingly, the structures of components in TSNA can be tuned separately given different data distributions. Notice that TSNA is a general framework and other meta recommendation structure, like Wide&Deep (Cheng et al. 2016), deepFM (Guo et al. 2017), can be also applied in TSNA instead of MLP. In order to learn the task-specific features, we design a router for selective activation. Specifically, the router will direct each embedding vector $x_i^j$ to the corresponding MLP, based on the contextual feature $ctx$. Then the output of TSNA $F_T(x_i^j)$ can be calculated as:

$$F_T(x_i^j) = \begin{cases} T_k(x_i^j) & k = j \\ 0 & otherwise \end{cases} \tag{5}$$

where $T_k$ is the output of $k^{th}$ MLP in TSNA.

Different from TSNA, GEN is designed to learn global feature representations from all tasks. To extract richer feature representations from different tasks, we utilize the idea from (Vaswani et al. 2017) and design an array of subnetworks, *i.e.*, $g_i$, as the inner structure of GEN. That is

$$G(x_i^j) = [g_0(x_i^j), g_1(x_i^j), ..., g_{e_n}(x_i^j)] \tag{6}$$

where $G$ is the concatenated output of each component in GEN and $e_n$ is the total number of $g_k$. By default, we use MLP as the inner structure. In GEN, each $g_k$ only focuses on a specific range of $x_i^j$, so that feature representations in different space can be extracted more explicitly, *i.e.*,

$$x_i^j = [\underbrace{a_1, ..., a_{l_0}}_{g_0}, ..., \underbrace{a_{D \cdot K - l_{e_n}+1}, ..., a_{D \cdot K}}_{g_{e_n}}] \tag{7}$$

where the focusing rage $l_k$ can be adaptively tuned under different task settings. By default, we set $l_k$ as multiples of K to align with the embedding vectors.

To incorporate the output of GEN in TSNA, we design an array of bi-linear attention units to extract useful knowledge with the help of $F_T$ for each task $j$. By default, we set the number of attention units the same with task number. Accordingly, the weight vector $\boldsymbol{w}^j$ can be calculated as:

$$\begin{aligned} \mathbf{a}^j &= F_T(x_i^j) * W * G(x_i^j)^T \in \mathbb{R}^{1 \times e_n} \\ \boldsymbol{w}^j &= [w_1^j, w_2^j, ..., w_{e_n}^j] \\ w_i^j &= \frac{\exp(a_i^j)}{\sum_{\forall a_k^j \in \mathbf{a}^j} \exp(a_k^j)} \end{aligned} \tag{8}$$

where $W$ is the weight-mapping matrix and $w_i^j$ is the weight for each vector in $G$. Then we can derive the output of the interaction layer $Q(x_i^j)$ as follows:

$$Q(x_i^j) = [F_T(x_i^j), \frac{\sum_{k=0}^{e_n} w_k^j g_k(x_i^j)}{\eta}] \in \mathbb{R}^{1 \times [D_{F_T} + D_g]} \tag{9}$$

where $\eta$ is a scaling factor.

**Fine tune Layer** As in previous works (Ma et al. 2018a,b), we pass $Q(x_i^j)$ to the corresponding top MLP network for fine-tuning, *i.e.*,

$$\hat{y}_i^j = \mathcal{K}_j[Q(x_i^j)] \tag{10}$$

where $\mathcal{K}_j$ is top MLP for task $j$ and $\hat{y}$ is the predicted output. We use ReLU activation for its better performance than sigmoid and tanh (Krizhevsky, Sutskever, and Hinton 2012).

Since the importance of different scenarios on CRs varies, we use a weighted cross entropy as the loss functions and add $l1$-norm to control the model complexity, *i.e.*,

$$\mathcal{L} = -\sum_{\forall i,j} \beta_j \cdot [y_i^j \log \hat{y}_i^j + (1 - y_i^j) \log(1 - \hat{y}_i^j)] + \lambda \Omega(\theta) \tag{11}$$

where $\beta_j$ denotes the weight of task $j$ and $\Omega(\theta)$ is the regularization for network parameters.

## Discount Estimator Design

Discount Estimator learns a transformation (*i.e.*, discount factor $\gamma$) for the conversion rate to simulate different ranks. Given the complexity between impression and rank, it takes a reinforcement learning based approach to derive the transformation as a set of policies. This way, it can estimate the corresponding impression based on the predicted conversion rate for calculating the total revenue.

**Discounted Conversion Rates** We reformulate the calculation of total revenue in Eq. (1) by introducing $Z^j(i) = \gamma^j(i) CR^j(i)$, the discounted conversion rate, where $\gamma^j(i)$ is the discount factor for product $i$ in task $j$. Importantly, ranking and impression of products are based on the discounted conversion rates rather than the original ones. Accordingly, we can estimate the corresponding impression under different ranks by reusing the predicted $CR$ from historical data. Hence there is no need to deploy a recommendation system online to collect new training data for $Rev$, which may harm the actual revenue (Pei et al. 2019).

Ranking products based on the discounted conversion rate may change their impression orders and thus, affects $CR^j(i)$ (Craswell et al. 2008). Therefore, we cannot directly use the predicted $CR^j(i)$ after changing ranks to estimate the total revenue as in Eq. (1). To solve this problem, we use a debiased direct measure (DDM) based on the direct measure to estimate $CR^j(i)$ (Langford, Li, and K 2011). Specifically, we introduce $\rho$ to alleviate the CR bias incurred by impression positions. Suppose $H$ is the estimated conversion rate if products are ranked based on $Z$, then we have:

$$H^j(i) = \frac{\sum_{\forall u} \rho^j(rank(Z_u^j(i))) \cdot CR_u^j(i) \cdot I_u^j(i)_Z}{\sum_{\forall u} I_u^j(i)_Z} \tag{12}$$

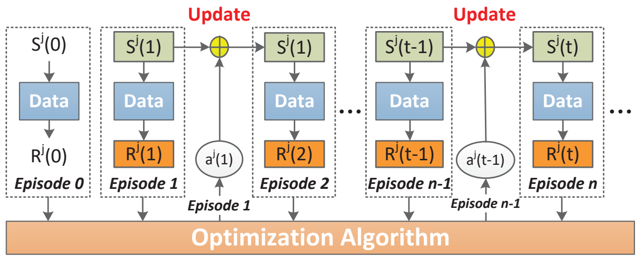


Figure 4: Workflow of discount estimator. It is initialized by a randomly chosen state $S^j(0)$. Then action $a^j(t)$ is updated based on the environmental feedback of episode $t$ and $t-1$. The process ends when $\sum_{\forall i,u} \Delta R^j_u(i) < \epsilon$.

where $I^j_u(i)_Z$ is the impression function based on $Z^j_u(i)$. $\rho^j$ is the position bias of impression estimated from the data, which can be obtained statistically (Guo et al. 2019).

Accordingly, the total revenue can be reformulated as:

$$Rev = \sum_{\forall i,j} I^j(i)_Z * H^j(i) * P(i) \tag{13}$$

**Learning Discount Factor** It is non-trivial to obtain a closed-form expression for $\gamma$ to directly optimize Eq. (13) since $I^j(i)_Z$ and $H^j(i)$ are not continuous. However, for each $\gamma$ and $CR^j(i)$, we can directly obtain $I^j(i)_Z$ and $H^j(i)$ from testing on the data set. By testing different $\gamma$, we can obtain a series of revenue values to select the optimal $\gamma$ that maximizes $Rev$. This process of learning $\gamma$ can be categorized as a sequential decision-making process, where $Rev$ is used as feedback to update the optimal $\gamma$ through trial and error. Thus, we formulate the problem as a reinforcement learning process and resort to REINFORCE algorithm (Williams 1992) to learn the discount factor. With out loss of generality, we use task $j$ to show the learning process, since the definitions and update schemes are identical for all tasks. Fig. 4 illustrates the workflow.

- **State $S$:** The triad $\{I^j(i)_Z, H^j(i), \gamma^j(i)\}$ denotes the state $S$ in the learning process. Following Eq. (2), we select top-$M^j$ in the candidate sets to calculate $I^j_u$.
- **Action $a$:** An action $a$ is defined as the change of $\gamma$. We denote the action for episode $t$ as $a^j(t) = \{\Delta\gamma^j(0,t), ..., \Delta\gamma^j(N,t)\}$, where $N$ denotes the number of total product candidates. The discount factor in episode $t+1$ can then be calculated as $\boldsymbol{\gamma}^j(t+1) = \boldsymbol{\gamma}^j(t) + a^j(t)$.
- **Reward $R$:** We define the reward as the total revenue in the current episode w.r.t $\gamma$. Notice that once we choose an action, new state should be tested on the whole data to obtain the reward. Accordingly, it is practical to treat the revenue provided by each sample equally. Therefore, the total reward can be calculated as:

$$R^j(t) = \sum_{\forall i,u} I^j(i,t)_Z * H^j_t(i,t) * P(i) \tag{14}$$

where variable $t$ denotes the calculation is based on $\gamma^j(t)$.

We use REINFORCE (Williams 1992) for the learning process, since the policies are learned from environmental feedbacks and we follow the policy gradient for optimization. In general, the optimization can be calculated as:

$$\gamma \leftarrow \gamma + \alpha \nabla_\theta \log \pi(\theta) v_t \tag{15}$$

where $\pi$ is the policy function and $v_t$ is the guiding vector for the update. In our case, we use the average reward change $\Delta \boldsymbol{R}^j(t) = [\Delta R^j(0,t), ..., \Delta R^j(N,t)]$ as the guiding vector, where $\Delta R^j(k,t)$ denotes the average signed revenue improvement for product $k$ in episode $t$. We also use the difference of $\gamma$ between two adjacent episodes to calculate $\nabla_\theta \log \pi(\theta)$. Combining Eq. (15) and Eq. (14), we have the following optimization equation:

$$\boldsymbol{\gamma}^j(t+1) \leftarrow \boldsymbol{\gamma}^j(t) + \alpha \Delta \boldsymbol{R}^j(t) \log \frac{\boldsymbol{\gamma}^j(t)}{\boldsymbol{\gamma}^j(t-1)} \tag{16}$$

where $\log \frac{\boldsymbol{\gamma}^j(t)}{\boldsymbol{\gamma}^j(t-1)}$ denotes the element-wise division of two $\gamma$. By moving $\boldsymbol{\gamma}^j(t)$ to the left, we can readily obtain the $a^j(t)$ for current episode. The learning process stops when $\sum_{\forall i} \Delta R^j(i,t) < \epsilon$, where $\epsilon$ is a predefined hyperparameter.

## Experiments

In this section, we first describe the basic settings for the experiments. Then we show both offline and online evaluation results to demonstrate the effectiveness of RevMan.

### Experimental Settings

**Datasets** Our dataset consists of 5.6 million online samples collected from the impression and conversion logs of a major online insurance platform in China.The samples come from four separate sales scenarios:

- *Special Sale Page (Scenario 1).* This sales scenario is for loyal customers who already know the platform. The sample size is $3,908,386$.
- *Paid User Page (Scenario 2).* This sales scenario is for subscribed users. The sample size is $183,848$.
- *Policy Info Page (Scenario 3).* This sales scenario is for users on the platform who have not subscribed a product. The sample size is $414,318$.
- *Children Product Page (Scenario 4).* This sales scenario is dedicated to customers who want to buy insurance products for their children. The sample size is $1,157,392$.

Note that for each impression on a user, there may exist multiple products. Like previous works (Pan et al. 2019; Ma et al. 2018b), we treat each pair of user and impressed product as an impression sample. As mentioned before, we also include contextual features, like sales scenario and time, in each sample. For conversion, we use the subscription of a product as a positive sample. We use the first $80\%$ samples for training, and the remaining $20\%$ for testing.

**Metrics** We conduct both online and offline evaluation. In offline evaluation, we first use AUC, Recall@N and NDCG to assess the ranking performance of RevMan, compared with different methods. In addition, we also evaluate the performance of RevMan in revenue improvement, *i.e.*, w/o

| **Dataset** | **Model** | **Scenario 1 (N=3)** | | | **Scenario 2 (N=2)** | | | **Scenario 3 (N=5)** | | | **Scenario 4 (N=3)** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | AUC | Recall@N | NDCG | AUC | Recall@N | NDCG | AUC | Recall@N | NDCG | AUC | Recall@N | NDCG |
| Train | LR | 0.8345 | 0.6205 | 0.6377 | 0.6902 | 0.3395 | 0.5107 | 0.8446 | 0.7211 | 0.5321 | 0.8516 | 0.8105 | 0.7754 |
| | Wide&Deep | 0.8842 | 0.6467 | 0.6767 | 0.7291 | 0.3815 | 0.5821 | 0.8655 | 0.7358 | 0.5526 | 0.8631 | 0.8249 | 0.7913 |
| | BSM | 0.8599 | 0.6454 | 0.6844 | 0.7180 | 0.3701 | 0.5703 | 0.8690 | 0.7346 | 0.5559 | 0.8653 | 0.8234 | 0.7906 |
| | CoNet | 0.8742 | 0.6316 | 0.6626 | 0.7045 | 0.3526 | 0.5370 | 0.8634 | 0.7554 | 0.5942 | 0.8593 | 0.7742 | 0.7838 |
| | MMoE | 0.8836 | 0.6483 | 0.6784 | 0.7231 | 0.3605 | 0.5634 | 0.8718 | 0.7843 | 0.6663 | 0.8731 | 0.8482 | 0.7936 |
| | SAMN | **0.8857** | **0.6586** | **0.6841** | **0.7336** | **0.4101** | **0.6118** | **0.8805** | **0.8601** | **0.6740** | **0.8768** | **0.8515** | **0.7947** |
| Test | LR | 0.8237 | 0.6097 | 0.6455 | 0.6351 | 0.3335 | 0.5109 | 0.7821 | 0.7703 | 0.4456 | 0.7952 | 0.7201 | 0.7355 |
| | Wide&Deep | 0.8268 | 0.6394 | 0.6569 | 0.6536 | 0.3734 | 0.5781 | 0.7958 | 0.7784 | 0.4936 | 0.8047 | 0.8076 | 0.7589 |
| | BSM | 0.8283 | 0.6458 | 0.6609 | 0.6459 | 0.3667 | 0.5286 | 0.8011 | 0.7816 | 0.5564 | 0.8032 | 0.7680 | 0.7535 |
| | CoNet | 0.8262 | 0.6328 | 0.6578 | 0.6405 | 0.3423 | 0.5235 | 0.8023 | 0.7927 | 0.5994 | 0.8001 | 0.7326 | 0.7488 |
| | MMoE | 0.8320 | 0.6407 | 0.6634 | 0.6444 | 0.3567 | 0.5255 | 0.8041 | 0.8196 | 0.6604 | 0.8077 | 0.8180 | 0.7600 |
| | SAMN | **0.8355** | **0.6543** | **0.6716** | **0.6597** | **0.3853** | **0.6026** | **0.8078** | **0.8278** | **0.6734** | **0.8127** | **0.8184** | **0.7628** |

Table 1: Comparison of ranking performance on the dataset of four scenarios. The best results are marked in bold.

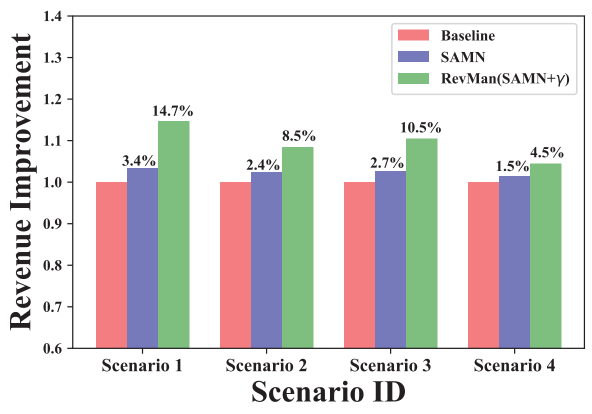


Figure 5: Offline evaluation in revenue improvement of RevMan. Compared with SAMN, RevMan achieves 11.3%, 6.1%, 7.8% and 3% revenue improvement in the four scenarios respectively.

the discount estimator. In online evaluation, we conduct A/B tests to validate the improvement of RevMan in conversion rate and revenue, compared with our online baseline. Further, we also use online behaviors of two popular products to illustrate the impacts of RevMan.

**Compared Methods** We compare the performance of RevMan with the following methods.

- LR (Mcmahan et al. 2013). Logistic regression is a widely used shallow model for CR prediction. It is also our online baseline. For each task in offline evaluation, we use the same parameter settings as online.
- Wide&Deep (Cheng et al. 2016). Wide&Deep is a deep neural network designed for CR prediction. We use its feature interaction method as the interaction layer.
- Bottom-Shared Network (BSM). We build a network from (Ma et al. 2018b), where the embedding layer is trained on all data. We adapt the model by building independent MLPs in fine tune layer for each task.
- CoNet (Hu et al. 2018). It uses a sparse weight matrix to enable feature sharing. We insert this matrix between the interaction layer and the fine tune layer for each task pair.
- MMoE (Ma et al. 2018a). It is similar to BSM, where a set of bottom networks are shared for all tasks. The difference is that the output vector from interaction layer is weighted sum over bottom network based on the input features.

During training, we use Adam (Kingma and Ba 2015) as the optimizer. For each model, the relevant hyper-parameters *e.g.*, neurons per layer, are empirically tuned. During testing, the learning rate is set to $lr = 0.002$ in order to control the update step. We assign $[0.5, 0.1, 0.2, 0.2]$ as the weight vector in the loss function Eq. (11) and set $\epsilon = 100$ as the stop condition.

## Offline Evaluation

First, we compare the ranking performance of SAMN, *i.e.*, the multi-task ranking model of RevMan at stage I, with the five methods above. Based on the best parameters derived from the offline trials, the experimental results are shown in Table 1. Compared with the best performance of the state-of-the-arts, SAMN achieves 0.42%, 0.93%, 0.46% and 0.62% AUC improvement in the four scenarios. The performance gain of SAMN over the rest is most notable in Scenario 2. This may be due to *(i)* Scenario 2 has the fewest samples since it focuses on paid users; *(ii)* the adaptive feature sharing mechanism in SAMN enables more effective experience sharing from other tasks. We can see that the improvement of Recall@N and NDCG is also the most significant in Scenario 2, demonstrating that the structure design in SAMN is effective in alleviating the data sparsity problem.

Then we show the effectiveness of RevMan in maximizing the total revenue. For fair comparison, we use SAMN as the baseline to check *whether the discount factor $\gamma$ can improve the total revenue*. We use Eq. (13) for revenue calculation and the position bias factor $\rho$ is obtained when $p < 0.05$. As shown in Fig. 5, by using $\gamma$, the revenue increases by 11.3%, 6.1%, 7.8% and 3% compared with SAMN. The discount factor $\gamma$ promotes the impression chance of those high-price products with less exposures properly, so that the total revenue can be increased.

## Online Evaluation

The online A/B tests are conducted in four scenarios for seven days. The incoming customers are randomly chosen to alleviate the bias of data distribution.

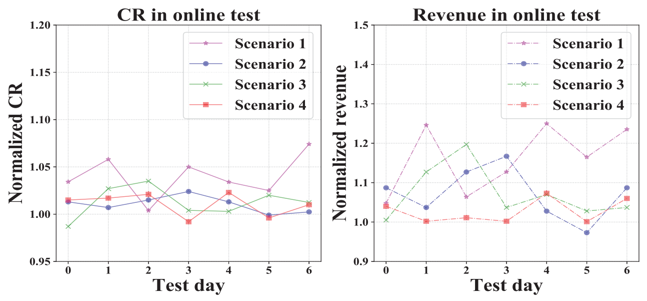


Figure 6: Online A/B Test. The performance of baseline is normalized to 1.0 for both CR and revenue.

First, we examine the overall performance of RevMan. As shown in Fig. 6, RevMan achieves 16.2%, 7.2%, 7.15% and 2.7% revenue improvement in those four scenarios compared with the baseline. RevMan also increases the conversion rate by 3.98%, 1.04%, 1.26% and 1.05% in the four scenarios. Although the performance of RevMan fluctuates, the p-value of collected improvement data is below 0.05, indicating the revenue improvement is reliable. The revenue improvement is lower in Scenario 4. This may be due to the limitations of product candidates given scenario design logic. With fewer products to recommend, the effectiveness of discount factor is less than those in other scenarios.

We then conduct a case study to examine the impression change after applying RevMan. We use Scenario 1 as the target since it has the largest number of visiting customers among the four scenarios. For comparison, we track two popular products: one has lower prices but high conversion rate, *i.e.*, G1, and the other G2 is the opposite. As shown in Fig. 7, the impression coverage of G1 is much higher than that of G2 before applying RevMan, since previous baseline focuses solely on conversion rate. After introducing RevMan, the results show that the recommendations are more effective. G2 gets more impression coverage, with slightly affected conversion rate. On the other hand, the conversion rate of G1 is also increased by over 60%. This observation indicates that RevMan can effectively improve the total revenue by promoting the impression of high-price products, while preserving overall conversion rate.

## Related Work

Our work is relevant to the following categories of research.

**Multi-task Recommendation.** Recently, lots of works have been proposed in this scope towards solving data sparsity (Ma et al. 2018b; Kitada, Iyatomi, and Seki 2019; Pan et al. 2019) or modeling multi-objectives (Ma et al. 2018a; Ma et al. 2019; Hu et al. 2018). In (Ma et al. 2018b), an embedding share method is proposed to jointly train click-through rate and conversion rate. The loss functions of those two tasks are combined to improve the performance for conversion rate prediction. CoNet (Hu et al. 2018) utilizes a weight matrix to control the feature sharing among different tasks. (Wang et al. 2019b) propose a knowledge-graph based approach to enable feature sharing among different tasks. However, it needs rich data to achieve acceptable performance, which is inapplicable in our scenario.

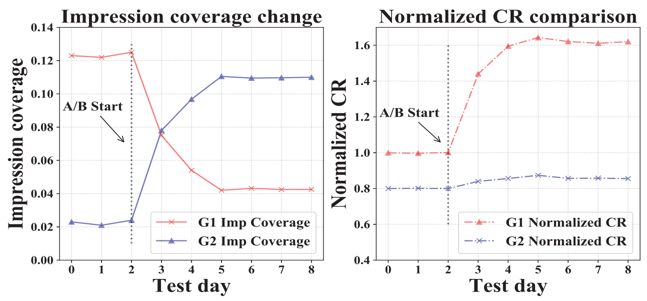


Figure 7: Case study of two products of scenario 1 in online A/B Test. G2, with higher price, gets more impression with RevMan. Comparison on CR shows RevMan improves the recommendation efficiency for both G1 and G2.

In summary, existing studies adopt static feature sharing among different tasks, which fails to capture the complex task relatedness. In contrast, we propose SAMN to adaptively learn the feature representations for each task, so as to improve the overall recommendation performance.

**Recommendation Economics.** The economics of recommendation is important since it is directly related to the revenue of companies. In (Zhang et al. 2016), an algorithm for surplus maximization is proposed. In (Zhao et al. 2017), researchers propose an algorithm focusing on maximizing the unit utility of recommended products. Similar idea is adopted in (Ge et al. 2019). A recent work (Pei et al. 2019) proposed to generalize the values of different behaviors together so that they can be modeled in the same framework.

These theoretical efforts are often impractical in practice since they fail to solve important revenue related issues, *e.g.*, modeling complex relationships between conversion rate and impression, etc. In our work, we propose a practical mechanism, *i.e.*, discount factor estimator, to fully consider those revenue related issues. Furthermore, we design an efficient RL algorithm to learn the best ranks for maximizing the total revenue.

## Conclusion

In this paper, we propose RevMan, the first online insurance recommendation system that maximizes the total revenue. RevMan is a two-stage design. At stage I, a Selective Attentive Multi-task Network (SAMN) is proposed to enable adaptive feature sharing among different tasks, so as to learn from sparse training data. Based on the model of stage I, we design a discount estimator to learn the discount factor $\gamma$ on the predicted conversion rate. With an offline RL algorithm, new ranks of products can be estimated towards maximizing the total revenue. Extensive offline and online experiments demonstrate the effectiveness of RevMan in comparison with the-state-of-the-arts.

## Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant No.61772229, 62072208.

## References

Acko. 2017. Acko General Insurance. https://www.acko.com/. Last accessed date: Sep 1st, 2020.

Bai, J.; Zhou, K.; Xue, G.; Zha, H.; Sun, G.; Tseng, B. L.; Zheng, Z.; and Chang, Y. 2009. Multi-task learning for learning to rank in web search. In *CIKM*, 1549–1552.

Bi, Y.; Song, L.; Yao, M.; Wu, Z.; Wang, J.; and Xiao, J. 2020. A Heterogeneous Information Network based Cross Domain Insurance Recommendation System for Cold Start Users. In *SIGIR*, 2211–2220.

Chapelle, O.; Shivaswamy, P.; Vadrevu, S.; Weinberger, K.; and Tseng, B. 2010. Multi-task learning for boosting with application to web search ranking. In *SIGKDD*, 1189–1198.

Cheng, H.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T. D.; Aradhye, H.; Anderson, G.; Corrado, G. S.; Chai, W.; Ispir, M.; et al. 2016. Wide & Deep Learning for Recommender Systems. In *RecSys*, 7–10.

Craswell, N.; Zoeter, O.; Taylor, M. J.; and Ramsey, B. 2008. An experimental comparison of click position-bias models. In *WSDM*, 87–94.

Ge, Y.; Xu, S.; Liu, S.; Geng, S.; and Zhang, Y. 2019. Maximizing Marginal Utility per Dollar for Economic Recommendation. In *WWW*, 2757–2763.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. *arXiv: Information Retrieval* .

Guo, H.; Yu, J.; Liu, Q.; Tang, R.; and Zhang, Y. 2019. PAL: a position-bias aware learning framework for CTR prediction in live recommender systems. In *RecSys*, 452–456.

Hu, G.; Zhang, Y.; Yang, Q.; and Guang-Neng, H. U. 2018. CoNet: Collaborative Cross Networks for Cross-Domain Recommendation. In *CIKM*, 667–676.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*. URL http://arxiv.org/abs/1412.6980.

Kitada, S.; Iyatomi, H.; and Seki, Y. 2019. Conversion Prediction Using Multi-task Conditional Attention Networks to Support the Creation of Effective Ad Creatives. In *SIGKDD*, 2069–2077.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NeurIPS*, 1097–1105.

Langford, J.; Li, L.; and K, M. D. 2011. Doubly Robust Policy Evaluation and Learning. In *ICML*, 1097–1104.

Ma, J.; Zhao, Z.; Chen, J.; Li, A.; Hong, L.; and Chi, E. H. 2019. SNR: Sub-Network Routing for Flexible Parameter Sharing in Multi-task Learning. In *AAAI*, 216–223.

Ma, J.; Zhao, Z.; Yi, X.; Chen, J.; Hong, L.; and Chi, E. H. 2018a. Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts. In *SIGKDD*, 1930–1939.

Ma, X.; Zhao, L.; Huang, G.; Wang, Z.; Hu, Z.; Zhu, X.; and Gai, K. 2018b. Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate. In *SIGIR*, 1137–1140.

Mcmahan, H. B.; Holt, G.; Sculley, D.; Young, M.; Ebner, D.; Grady, J. P.; Nie, L.; Phillips, T.; Davydov, E.; Golovin, D.; et al. 2013. Ad click prediction: a view from the trenches. In *SIGKDD*, 1222–1230.

Mordor, I. 2019. China Online Insurance Market - Growth, Trends, and Forecast (2019-2024). https://www.mordorintelligence.com/industry-reports/china-online-insurance-market.

Pan, J.; Mao, Y.; Ruiz, A. L.; Sun, Y.; and Flores, A. 2019. Predicting Different Types of Conversions with Multi-Task Learning in Online Advertising. In *SIGKDD*, 2689–2697.

Pei, C.; Yang, X.; Cui, Q.; Lin, X.; Sun, F.; Jiang, P.; Ou, W.; and Zhang, Y. 2019. Value-aware Recommendation based on Reinforced Profit Maximization in E-commerce Systems. *arXiv: Information Retrieval* .

PingAn. 2016. PingAn JinGuanJia. https://gj.pingan.com. Last accessed date: Sep 1st, 2020.

Ruder, S. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv: Learning* .

Tencent. 2017. Tecent WeSure. https://www.wesure.cn/index.html. Last accessed date: Feb 20th, 2021.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *NeurIPS*, 5998–6008.

Wang, B.; Zhou, J.; Qiu, M.; Wang, X.; and Cai, D. 2019a. A Minimax Game for Instance based Selective Transfer Learning. In *SIGKDD*, 34–43.

Wang, H.; Zhang, F.; Zhao, M.; Li, W.; Xie, X.; and Guo, M. 2019b. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In *WWW*, 2000–2010.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8(3-4): 229–256.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? In *NeurIPS*, 3320–3328.

Zhang, W.; Bao, W.; Liu, X. Y.; Yang, K.; Lin, Q.; Wen, H.; and Ramezani, R. 2019. Large-scale Causal Approaches to Debiasing Post-click Conversion Rate Estimation with Multi-task Learning. In *WWW*, 2775–2781.

Zhang, Y.; and Yang, Q. 2017. A Survey on Multi-Task Learning. *arXiv: Learning* .

Zhang, Y.; Zhao, Q.; Zhang, Y.; Friedman, D.; Zhang, M.; Liu, Y.; and Ma, S. 2016. Economic Recommendation with Surplus Maximization. In *WWW*, 73–83.

Zhao, Q.; Zhang, Y.; Zhang, Y.; and Friedman, D. 2017. Multi-Product Utility Maximization for Economic Recommendation. In *WSDM*, 435–443.