Momentum-based Gradient Methods in Multi-objective Recommender Systems

Blagoj Mitrevski[1]*, Milena Filipovic[1]*, Diego Antognini[1], Emma Lejal Glaude[2], Boi Faltings[1], Claudiu Musat[2]

[1]Ecole Polytechnique Fédérale de Lausanne, Switzerland
[2] Swisscom, Switzerland
firstname.lastname@{epfl.ch, swisscom.com}

Abstract

Multi-objective gradient methods are becoming the standard for solving multi-objective problems. Among others, they show promising results in developing multi-objective recommender systems with both correlated and uncorrelated objectives. Classic multi-gradient descent usually relies on the combination of the gradients, not including the computation of first and second moments of the gradients. This leads to a brittle behavior and misses important areas in the solution space.

In this work, we create a multi-objective *Adamize* method that leverage the benefits of the Adam optimizer in single-objective problems. This corrects and stabilizes the gradients of every objective before calculating a common gradient descent vector that optimizes all the objectives simultaneously. We evaluate the benefits of *Multi-objective Adamize* on two multi-objective recommender systems and for three different objective combinations, both correlated or uncorrelated. We report significant improvements, measured with three different Pareto front metrics: hypervolume, coverage, and spacing. Finally, we show that the *Adamized* Pareto front strictly dominates the previous one on multiple objective pairs.

1 Introduction

Decision-making relies on multiple factors. The world is complex, and many problems require an optimization for more than one objective. Multi-objective problems are present in fields like engineering, economics, finance, logistics, and many more. Multi-objective optimization is the area of decision-making in which we simultaneously optimize for more than one objective. We distinguish two types of objectives: the correlated and the conflicting ones. When the objectives are conflicting, the choice of the optimal decisions needs to be taken in the presence of trade-offs: choosing one objective usually comes at the expense of the others. In practice, the decision of choosing the best solution is left to the domain experts or the business stakeholders. Multi-objective optimization provides a data-driven alternative.

Recommenders are not only about relevance. One of the objectives of recommender systems is to be accurate, namely to successfully model the user's preferences. These systems are however not limited to accuracy. Another objective that can improve the user's experience with the recommender system is proposing more diverse content. It helps the user to escape their filter bubble that can reduce user creativity, learning, and connection (Nguyen et al. 2014). Also, promoting more recent content (Chakraborty et al. 2017; Gabriel De Souza, Jannach, and Da Cunha 2019) can bring social value by keeping the user up to date.

However, among the multiple stakeholders of the recommender system it is possible to encounter diverse and competing objectives. For instance, increasing the revenue for a company does not always mean the user will get a better and improved experience. If an application store puts more importance on recommending overpriced applications it may increase its revenue, but this strategy will hurt developers of free and cost-effective applications; also it will put a burden on the user's budget. This becomes more frequent, as more and more companies are becoming socially responsible (Varona 2020; Hatcher 2000; Vveinhardt and Andriukaitiene 2014), which can be unaligned with traditional business objectives.

From one to multiple objectives. Prior work (Poirion, Mercier, and Désidéri 2017) proposed the gradient-based multi-objective optimization algorithm, called the Stochastic Multi-Subgradient Descent Algorithm (SMSGDA), or an improved version for recommendation (Milojkovic et al. 2019). The method computes the gradients with respect to each objective and then constructs a common descent vector by taking a linear combination of the individual gradients. The weight of each gradient is computed by solving a quadratic constrained optimization problem. Finally, the model parameters are updated in the opposite direction of the common descent vector. The problem with stochastic single-objective optimization is the stochasticity that comes from using mini-batches or dropout regularization.

In single-objective settings, this problem is solved using optimizers like Adam (Kingma and Ba 2014) and RMSprop (Hinton, Srivastava, and Swersky 2012). These stabilize the computation and speed up to convergence. In a similar fashion, we introduce a simple yet effective *Adamize* trick for multi-objective problems. We keep track of the first and second moments of the gradients and use the momentums to correct the gradients and compute better gradient weights. Finally, we calculate a more stable common descent vector

*Work done while at EPFL and Swisscom.

using the corrected gradients.

In this work, we thus make the following contributions: we address the recommendation task with multiple-objectives, in which objectives can either be correlated or conflicting. We first present the *Adamize* trick to correct and stabilize the gradients of every objective before aggregating them into a common gradient descent. We then show that our novel multi-gradient descent method can be easily integrated into state-of-the-art recommender systems. We evaluate our method using two real-world recommendation datasets with up to three objectives. We then compare the results of the momentum-based optimization with the state of the art using three different metrics based on the resulting Pareto fronts. As the observed differences are stark, we complement our analysis with visualizations that further underline the usefulness of momentum-based multi-gradient descent in multi-objective recommender systems.

2 Background

There are different ways of solving the multi-objective optimization problem, such as evolutionary algorithms, re-ranking, and gradient-based solutions. In this work, we focus on the latter. We present the multi-gradient descent algorithm for multi-objective optimization (Milojkovic et al. 2019), the basis of the current work.

2.1 Definitions

Multi-Objective Optimization. The multi-objective optimization of a model can be formally defined as:

$$\min_{w \in \mathbb{R}^D} \mathcal{L}(w) = \min_{w \in \mathbb{R}^D} \mathcal{L}_1(w), \dots \mathcal{L}_n(w) \tag{1}$$

where w are the model parameters, D is the dimension of the model parameters, $\mathcal{L}(w) : \mathbb{R}^D \to \mathbb{R}^n$ is a vector valued objective function with continuously differentiable objective functions $\mathcal{L}_n(w) : \mathbb{R}^D \to \mathbb{R}$.

Common Descent Vector. The common descent vector (Désidéri 2012) is the core of the multi-gradient descent algorithm. It is computed with a linear combination of the gradients:

$$\nabla_w \mathcal{L}(w) = \sum_{i=1}^{n} \alpha_i \nabla_w \mathcal{L}_i(w) \tag{2}$$

with $\alpha_i \geq 0, i \in \{1, \dots, n\}$, and $\sum_{i=0}^{n} \alpha_i = 1$, where $\mathcal{L}_i(w)$ is the gradient of the i-th objective, α_i is the weight of the i-th gradient objective, n is the number of objectives, and w are the model parameters.

Pareto Stationary Solution. A solution w of the Equation 2 is Pareto stationary iff it satisfies the Karush–Kuhn–Tucker (KKT) conditions. In other words, there exists $\alpha_1 \dots \alpha_n$ that statisfy the three following constraints:

$$\alpha_1 \dots \alpha_n \geq 0, \sum_{i=0}^{n} \alpha_i = 1, \text{and} \sum_{i=1}^{n} \alpha_i \nabla_w \mathcal{L}_i(w) = 0$$

2.2 Multi-Gradient Descent Algorithm (MGDA)

After the definition of the common descent vector and the Pareto stationary solution, we present the multi-gradient descent algorithm (MGDA) (Désidéri 2012). The algorithm is deterministic and is proven to converge to a Pareto stationary solution. For an arbitrary number of objectives, this algorithm computes the alphas (i.e., weights of gradients, see Equation 2) to create a common descent vector. This vector is made such that the optimization step in the opposite direction of this common descent vector; all the objectives are simultaneously optimized. To compute the alphas, we need to solve the following quadratic constrained optimization problem (QCOP):

$$\min_{\alpha_1, \dots, \alpha_n} \left\{ \left\| \sum_{i=1}^{n} \alpha_i \nabla_w \mathcal{L}_i(w) \right\|^2 \mid \sum_{i=1}^{n} \alpha_i = 1, \alpha_i \geq 0 \right\} \tag{3}$$

After computing the alphas, we compute the final common descent vector $\nabla_w \mathcal{L}(w)$. If $\nabla_w \mathcal{L}(w) = 0$ the solution is Pareto Stationary. Otherwise, $\nabla_w \mathcal{L}(w) \neq 0$, the solution is not Pareto Stationary and thus, we apply an optimisation step in the opposite direction of the common descent vector, improving each objective at once.

It is worth noting that if there are two objectives, an analytical solution to the QCOP problem exists. Otherwise, the QCOP can be solved by using the Frank-Wolfe constrained optimization algorithm as in (Sener and Koltun 2018).

2.3 Stochastic Multi-Subgradient Descent Algorithm (SMSGDA)

The previous multi-gradient descent algorithm has few drawbacks to be used in real-world problems. A first one is the need to compute the full gradient at every optimization step which makes it computationally expensive and in some cases infeasible. A second one, the requirements do not allow to use non-smooth loss functions as objective functions. All of these drawbacks are solved by the Stochastic Multi-Subgradient Descent Algorithm (SMSGDA) presented in (Poirion, Mercier, and Désidéri 2017). The Stochastic Multi-Subgradient Descent Algorithm is similar to the Multi-Gradient Descent Algorithm, with the difference that instead of computing the gradients for every objective and then computing the alphas using the whole dataset, we are computing them on a subset of the dataset. Therefore, the stochasticity comes from the mini-batch.

2.4 Gradient Normalization

In real-world use-cases, the objectives for which we are optimizing may have different scales. This causes a problem for the MGDA and SMSGDA algorithms because they will favor the objectives that have a higher scale, leading to unbalanced solutions that perform well on certain objectives, but badly on the others.

To solve this problem, after computing the gradients, the authors normalize them to interval according to the maximal empirical loss for each objective:

$$\nabla_w \hat{\mathcal{L}}_i(w) = \frac{\nabla_w \mathcal{L}_i(w)}{\mathcal{L}_i(w_{init})} \tag{4}$$

Algorithm 1 SMSGDA with Gradient Normalization

$initialize()$
for $i \in 1, ..., n$ **do**
  $empirical\_loss_i = \mathcal{L}_i(w)$
end for
for $epoch \in 1, ..., M$ **do**
  for $batch \in 1, ..., B$ **do**
    $do\_forward\_pass()$
    for $i \in 1, ..., n$ **do**
      $calculate\, loss \;\; \mathcal{L}_i(w)$
      $calculate\, gradient \;\; \nabla_w \mathcal{L}_i(w)$
      $normalize\, gradient \;\; \nabla_w \hat{\mathcal{L}}_i(w) = \frac{\nabla_w \mathcal{L}_i(w)}{empirical\_loss_i}$
    end for
    $\alpha_1, \ldots, \alpha_n = \text{QCOPSolver}\left(\nabla_w \hat{\mathcal{L}}_1(w), \ldots, \nabla_w \hat{\mathcal{L}}_n(w)\right)$
    $\nabla_w \mathcal{L}(w) = \sum_{i=1}^{n} \alpha_i \nabla_w \hat{\mathcal{L}}_i(w)$
    $w = w - \eta \nabla_w \mathcal{L}(w)$
  end for
  $evaluate\_model()$
  $update\_pareto\_set()$
end for

where $\nabla_w \hat{\mathcal{L}}_i(w)$ is the resulting normalized gradient, $\nabla_w \mathcal{L}_i(w)$ is the original gradient of the i-th objective, $\mathcal{L}_i(w_{init})$ is the initial loss for the i-th objective which is used as approximation for the maximum empirical loss for the given objective. The final algorithm is shown in Algorithm 1.

3 Related work

3.1 Multi-Objective Recommendation

With the advances of neural approaches in other fields, they also found their way into recommendation systems. First, the introduction of Neural network-based Collaborative Filtering (He et al. 2017) showed promising results. Later, the Variational Autoencoders for Collaborative Filtering (Liang et al. 2018) became state-of-the-art and still keeps its title as the best collaborative filtering based recommender.

Recommender systems and ranking problems have similarities: learning a personalized recommender can be transformed as a ranking problem (Karatzoglou, Baltrunas, and Shi 2013). The multi-objective ranking optimization in (Carmel et al. 2020) is solved by label aggregation. This method collects the multiple labels of the training examples into a single label, and then use a single-objective optimizer to rank the aggregated label, solving the multi-objective problem.

Alternatively, the gradient-based methods can solve the multi-objective optimization problem. In (Désidéri 2012), the authors propose the Multi-Gradient Descent Algorithm (MGDA) for optimizing multi-objective based on the steepest descent method. This algorithm is an adjustment of the classical gradient descent algorithm to work with multiple objectives. The same authors of the MGDA algorithm extended it to Stochastic Multi-Subgradient Descent Algorithm (SMSGDA) (Poirion, Mercier, and Désidéri 2017). The SMSGDA is a stochastic version of the MGDA that could also work with non-smooth objective functions.

A more robust gradient-based multi-objective optimization algorithm that still works in cases when the exact gradients could not be computed is presented in (Peitz and Dellnitz 2018). To alleviate the inaccuracies, an additional condition is presented for the descent direction.

(Milojkovic et al. 2019) proposed a gradient-based algorithm for optimizing multi-objective recommender systems. Their solution is based on finding a common descent vector, which is a combination of the gradients of every objective. By taking an optimization step in the opposite direction of this common descent vector, the model is optimized for all objectives simultaneously. We build upon this work and improve convergence and stability of the optimization.

3.2 Multi-Task Learning

Multi-task learning is inherently a multi-objective problem because different tasks may conflict, requiring a trade-off (Sener and Koltun 2018). In multi-task learning, where one model gives multiple predictive outputs, the gradients per task usually have different magnitudes. In (Chen et al. 2018), the authors propose a technique that automatically balances the training procedure by dynamically tuning gradient magnitudes. Their GradNorm technique improves the accuracy of the models, reduces overfitting across multiple tasks, and decreases the number of hyperparameters. (Sener and Koltun 2018) cast the framework as a multi-objective optimization, the authors solve the multi-task problem by casting it into a multi-objective optimization problem, coupling the two problems together. They state that different tasks may be conflicting, requiring a multi-objective setup. The findings of our work are applicable to multi-task learning and support conflicting objectives.

4 The Adamize Trick for Multi-Objective Optimization

When optimizing models on a single objective, we are usually doing it in a stochastic fashion. The stochasticity comes from using mini-batch stochastic gradient descent where we use subsets of the data to compute the gradient, or use a dropout regularization (Srivastava 2013). The stochasticity in the optimization algorithm introduces noise in the gradient and may cause the algorithm to converge slower, or even diverge.

There exist multiple optimizers like Adam (Kingma and Ba 2014) and RMSprop (Hinton, Srivastava, and Swersky 2012) which aim to stabilize the gradients when doing an optimization step. They achieve the stabilization by keeping a running average of the first and second moments of the gradients and taking a step in the opposite direction of the corrected gradient by using the first and the second momentum. For example, the corrected gradient moves faster on steep slopes and oscillates less on valleys and thus, move faster to the optima. Following the intuition behind ADAM and RMSprop, it may be beneficial, when using the Stochastic Multi-Subgradient Descent Algorithm (SMSGD), available in Algorithm 1, to smooth the gradients from the different objectives before calculating alphas and combining them to get the final common descent vector. Intuitively, this may

Algorithm 2 SMSGDA with Gradient Normalization and Adamizing Every Objective

$initialize()$
for $i \in 1, ..., n$ **do**
  $empirical\_loss_i = \mathcal{L}_i(w)$
end for
for $epoch \in 1, ..., M$ **do**
  for $batch \in 1, ..., B$ **do**
    $do\_forward\_pass()$
    $evaluate\_model()$
    $update\_pareto\_set()$
    for $i \in 1, ..., n$ **do**
      $calculate\, loss \;\; \mathcal{L}_i(w)$
      $calculate\, gradient \;\; \nabla_w \mathcal{L}_i(w)$
      $normalize\, gradient \;\; \nabla_w \hat{\mathcal{L}}_i(w) = \frac{\nabla_w \mathcal{L}_i(w)}{empirical\_loss_i}$
      $\boldsymbol{\nabla_w \bar{\mathcal{L}}_i(w) = Adamize(\nabla_w \hat{\mathcal{L}}_i(w))}$
    end for
    $\alpha_1, \ldots, \alpha_n = \text{QCOPSolver}\left(\nabla_w \bar{\mathcal{L}}_1(w), \ldots, \nabla_w \bar{\mathcal{L}}_n(w)\right)$
    $\nabla_w \bar{\mathcal{L}}(w) = \sum_{i=1}^{n} \alpha_i \nabla_w \bar{\mathcal{L}}_i(w)$
    $w = w - \eta \nabla_w \bar{\mathcal{L}}(w)$
  end for
end for

lead to more stable alpha computations, faster convergence, and convergence to better solutions.

The vanilla SMSGD algorithm is presented in Section 2.3 and the pseudo-code is given in Algorithm 1. Our proposition is to use Adam based optimizers for every objective before computing the common descent vector. We directly add the Adam computation for every objective. Therefore, the difference with the vanilla SMSGD is that we are also keeping the running average for the gradient of every objective, instead of keeping only the average of the common descent vector. Since these are the gradients that affect the computations of the alphas, the final common descent vector is expected to be more stable. The pseudo-code of the *Adamize* trick for the gradients is presented in Algorithm 2 and Algorithm 3. The difference between Algorithm 1 and Algorithm 2 is in the bold line: instead of using the original gradients from every objective, we correct them using the first and second momentums, and we use the corrected gradients to compute the alphas and the common descent vector.

In terms of computation and memory requirement, the complexity is linear with respect to the number of objectives. We save the first and second momentums of every objective. As the number of objectives is small, the overhead of our method is insignificant.

5 Experiments

In this section, we assess the improvement of the proposed *Adamize* trick on two datasets and up to three correlated and conflicting objectives.[1]

[1]For simplicity, we will use interchangeably the words objectives and losses.

Algorithm 3 Adamizing a Gradient

Parameters: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Parameters: λ: Gradient correction magnitude parameter
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
procedure ADAMIZE($\nabla_w \mathcal{L}(w_t)$)
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_w \mathcal{L}(w_t)$ (The gradient w.r.t. stochastic objective at timestamp t)
  $m_t \leftarrow \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$ (Update biased first moment estimate)
  $v_t \leftarrow \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$ (Update biased second raw moment estimate)
  $\hat{m_t} \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
  $\hat{v_t} \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
  return $(1 - \lambda)\nabla_w \mathcal{L}(w_t) + \lambda * \hat{m_t} / (\sqrt{\hat{v_t}} + \epsilon)$ (Smoothed gradient)
end procedure

5.1 Objectives for Recommendation

A recommender system can be trained with different objectives and for different purposes. For example, for some companies, there might be an economic or strategic incentive to recommend newer, instead of older, content to the users. Other socially responsible companies would like that their recommender to learn a notion of fairness or awareness of social biases.

In this section, we present the objectives we employ in our experiments. As a use-case, we use the state-of-the-art variational autoencoder Mult-VAE$^{\text{PR}}$ of (Liang et al. 2018) to demonstrate how to integrate our objectives into an existing recommender training procedure. However, we emphasize that they are easily adapted to any other model that, as recommendation, outputs a vector of probabilities across all the items.

Relevance Objective. This loss measures the relevance of the predicted items for the given user. The idea is to compare the output of the model with the user's interactions and measure how good the model can predict the user's interactions. The relevance loss in variational autoencoders is simply the reconstruction loss, plus the KL divergence between the posterior and the prior. More formally, the loss is:

$$\mathcal{L}(\boldsymbol{x}; \theta, \phi) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \beta * KL(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \tag{5}$$

where $\boldsymbol{x}$ is the input vector for a user, θ and ϕ are model parameters, $\mathbf{z}$ is the variational parameter of the distribution, and β is the regularizer controlling how much weight to be given to the KL term.

Revenue Objective. Alongside the enhanced user experience, a company is incentivized to use a recommender to increase simultaneously the revenue. Therefore, the revenue loss can be used in the training process to boost the recommendations of expensive items, increasing the overall revenue generated. The loss is similar to the relevance loss of Section 5.1, with a difference that the input of the model is multiplied by a weight vector, representing the prices of the items. Before computing the log-likelihood for a given user, the input vector for a user is multiplied with the price vector:

$$\mathcal{L}(\boldsymbol{x}; \theta, \phi) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\pi * \boldsymbol{x}|\boldsymbol{z})] \tag{6}$$

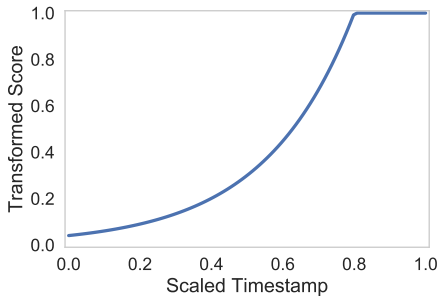


Figure 1: Our proposed recency function (Equation 7).

where $\boldsymbol{x}$ is the input vector for a user, π is the price vector, the $*$ symbol denotes element-wise multiplication between two vectors, θ and ϕ are model parameters, and $\mathbf{z}$ is the variational parameter of the variational distribution.

Recency Objective. From our practical experience, we came across a finding that users strongly prefer to interact with recently added content. Furthermore, the authors of (Ding, Li, and Orlowska 2006) have shown that with the introduction of recency we could get improved and more precise recommender systems.

Computing a recency score for items remains an open question. For a given dataset, we propose to leverage the timestamps of the items when they first became available. For an item, we scale its timestamp using a min-max normalization between the first and last interaction any user had with it. However, we claim that recency is not a linear function of the time. Since we want to promote more recent items, we propose to transform the scores according to the following function, also depicted in Figure 1:

$$f(x) = \begin{cases} 1, & \text{if } x \geq 0.8 \\ 0.3^{(0.8-x)*\frac{10}{3}}, & \text{otherwise} \end{cases} \tag{7}$$

Based on this transformation function, we proposed the recency objective which stimulates the model to recommend recent items. The input of the model is multiplied by a weight vector, which represents the recency score of the items, when the loss is computed. Similarly to the other losses, before computing the log-likelihood for a given user, the input vector for a user is multiplied with the recency vector, or mathematically:

$$\mathcal{L}(\boldsymbol{x}; \theta, \phi) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\rho * \boldsymbol{x}|\boldsymbol{z})] \tag{8}$$

where $\boldsymbol{x}$ is the input vector for a user, ρ is the recency vector, the $*$ symbol denotes element-wise multiplication between two vectors, θ and ϕ are model parameters, $\mathbf{z}$ is the variational parameter of the distribution.

5.2 Datasets

In order to assess the effectiveness of our proposed model, we first carried out experiments on the well-known Amazon Books dataset, being a subsample from the Amazon review dataset (He and McAuley 2016; Harper and Konstan 2015). Along with users preferences for books, it contains the book prices which can be used as a second revenue objective for multi-objective models (see Section 5.1). We also consider the MovieLens dataset (Harper and Konstan 2015).

In terms of objectives, we use the relevance, revenue, and recency objectives for the MovieLens dataset. For the Amazon Books dataset we used only the relevance and revenue objectives since the recency is not available.

Relevance Objective. We employ the definition of (Liang et al. 2018) to measure relevance. We quantify the proportion of relevant top-k items to a user with *Recall@k*:

$$Recall@k(u, \omega) := \frac{\sum_{r=1}^{k} \mathbb{I}[\omega(r) \in I_u]}{min(k, |I_u|)} \tag{9}$$

where $\omega(r)$ is the item at rank r, I_u the set of held-out items that user u interacted with, and $\mathbb{I}[\cdot]$ the indicator function.

Revenue Objective. To model the revenue objective for MovieLens, we enriched it with prices from the Amazon review dataset by doing a fuzzy joining on the titles of the movies. For Amazon Books dataset, the prices are already included. We denote the final price vector π.

Recency objective In the MovieLens dataset for every given rating, there is a timestamp indicating when the rating was given by the user. We assume that a given movie became available when the first rating was given for it. This enables us to create an additional objective, recency objective. A model trained with the recency objective is expected to prefer recommending recently available movies. Using these availability timestamps, we create a recency vector ρ.

Finally, we train the following combination of objectives:

1. Relevance + Revenue objectives;
2. Relevance + Recency objectives;
3. Revenue + Recency objectives;
4. Relevance + Revenue + Recency objectives.

5.3 Preprocessing

In our experiments, we consider implicit feedback. Therefore, we first binarize the ratings by converting ratings higher than or equal to 3.5 to positive interaction, and ratings lower than 3.5 to negative interaction. Then, we split the data in a way that 90% of the users with their interactions are used as training data, 5% are used as validation data, and the remaining 5% are used as testing data. Finally, we mask 20% of interactions per user in the validation and testing data. The remaining 80% of the interactions are used as input to the model, and the masked 20% are used as ground truth, to compare the model's output with.

5.4 Experimental Settings

We implemented the state-of-the-art variational autoencoder Mult-VAE^{PR} of (Liang et al. 2018) for collaborative filtering, and augmented the training loss with the objectives described in Section 5.1. Our VAE model contains an encoder and a decoder. The encoder consists of two linear layers of sizes 600 and 400. The decoder also consists of two linear

Hyperparameter	Value Range
Optimizer	Adam, AdamW
Gradient correction	1e-2, 1e-3, 1e-4, 1e-5
β_1	0.9, 0.99, 0.999
β_2	0.99, 0.999, 0.9999

Table 1: The hyperparameter search space to adamize the gradients.

Hyperparameter	ML-2	ML-3	AB-2
optimizer	AdamW	AdamW	AdamW
gradient correction	1e-4	1e-4	1e-3
β_1	0.999	0.999	0.9
β_2	0.999	0.9999	0.999

Table 2: The final hyperparemeters for MovieLens with two and three objectives, and the AmazonBook dataset with two objectives.

layers, both with a size of 600. The number of latent features, the bottleneck of the model is 200. We are also normalizing the input before we forward it through the model. As regularization, we use a dropout of 0.5 to the input[2].

Employing the *Adamize* trick for the gradients require hyperparameters that must be tuned. To find the most optimal hyperparameters, we are doing an extensive grid search. It has the following hyperparameters:

- Optimizer: the type of optimizers to use for every objective, it can be either Adam or AdamW. The AdamW (Loshchilov and Hutter 2017) is an improvement of Adam (Kingma and Ba 2014), fixing the way weight decay is implemented in Adam;
- Gradient correction magnitude: how much importance/weight to give to the Adamazing of the gradients of every objective;
- β_1: exponential decay rate for the first moment. estimate, used by the optimizers;
- β_2: exponential decay rate for the second estimate, used by the optimizers.

For more details about the hyperparameters, please refer to Section 4 and Algorithm 3. The values used in the grid search are summarised in Table 1 and the final hyperparamaters in Table 2.

5.5 Pareto Front Metrics

It is not straightforward to compare the multi-objective solution from different multi-objective algorithms and optimization strategies. The solutions from the methods of multi-objective optimization are in the form of Pareto sets. An initial comparison of two and three-dimensional Pareto sets is to plot them and inspect them visually. Although visual inspection can help us to rank and compare Pareto set solutions, we seek an objective and systematic way. Therefore, in this section, we present three metrics for measuring the quality of the Pareto set which can help us measure the performance of the multi-objective algorithms quantitatively.

[2]We will make the code available.

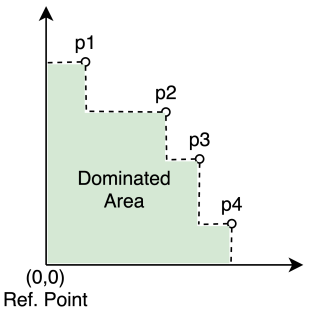


Figure 2: Hypervolume in two dimensions.

Hypervolume (Zitzler, Brockhoff, and Thiele 2007): One of the ways of measuring the quality of the Pareto set is to measure the area that is dominated by it. The intuition is, the larger the area the solution can dominate, the better the solution. Using the hypervolume to compute the area dominated by to solution, this intuition can be extended to more than two dimensions (Zitzler and Thiele 1999). Since we are interested in increasing the recommender system metrics, we are using the origin as a reference point for computing the hypervolume. An example of the hypervolume in two dimension space is shown in Figure 2, where $p1$, $p2$, $p3$, and $p4$ are points in the Pareto set, and the hypervolume is colored with green.

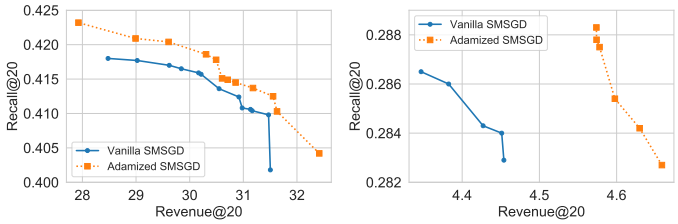
Coverage (Zitzler and Thiele 1999): The coverage is a metric that indicates the fraction of points from one Pareto set that are dominated by or equal to points from another Pareto set. If a one point $p1$ is dominated by or equal to another point $p2$, than it is said that $p2$ covers $p1$. If the coverage is 1.0, that means all the points from the second Pareto set are covered by points from the first one. Reverse, if the coverage is 0.0, that means none of the points from the second Pareto set are covered by points from the first set.

However, a drawback of the coverage is that it cannot tell us by how much one solution is better than the other one (Zitzler and Thiele 1999). If P_{S_1} is the first Pareto set, P_{S_2} is the second Pareto set, and with $p1 \geq p2$ we denote that solution point $p1$ covers solution point $p2$, then the coverage metric is defined as:

$$\mathcal{C}(P_{S_1}, P_{S_2}) = \frac{|\{p2 \in P_S 2; \exists p1 \in P_{S_1} : p1 \geq p2\}|}{|P_{S_2}|} \quad (10)$$

It is important to note that the coverage metric is not symmetric, and both $\mathcal{C}(P_{S_1}, P_{S_2})$ and $\mathcal{C}(P_{S_2}, P_{S_1})$ have to be examined when evaluating Pareto sets. In our experiments, we report both variants as we apply a pairwise comparison.

Spacing (Okabe, Jin, and Sendhoff 2003): The spacing is a distance-based metric that measures the spread of a given solution. The bigger the spacing metric is, the more diverse and the more spread are the solutions in the Pareto set. If having the best solutions in a Pareto set is important, the diversity of the solutions captures the range of choices available to the decision-makers. This is a concrete business advantage. If P_S is the Pareto set, d_i is the distance to the clos-



(a) MovieLens dataset. (b) Amazon dataset.

Figure 3: Visualization of Pareto fronts for two objectives.

Dataset	Method	Hypervolume	Coverage	Spacing
Movies	Vanilla	13.16	0.0	0.19
	Adamized	**13.68**	**1.0**	**0.34**
Books	Vanilla	1.28	0.0	**0.016**
	Adamized	**1.34**	**1.0**	0.014

Table 3: Pareto front metrics for MovieLens and Amazon Book datasets on two objectives.

est neighbour of the i-th point in the Pareto set, and $\bar{d}$ is the average of d_i, then the spacing is computed as:

$$\mathcal{SP}(P_S) = \sqrt{\frac{1}{|P_S| - 1} \sum_{i=1}^{|S|} (d_i - \bar{d})^2} \qquad (11)$$

6 Results

6.1 Two Objectives

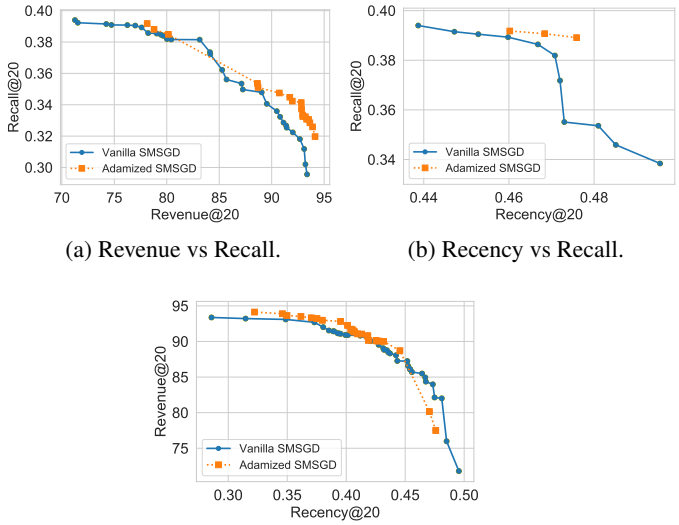
Figure 3 shows the Pareto front of the baseline and our method. From both visualizations we can clearly observe that *Adamizing* the gradients significantly improves the performance over the SMSGD algorithm. The Pareto front obtained with our method clearly dominates the vanilla SMSGD algorithm. On MovieLens, the Pareto fronts are more spread than in the Amazon dataset case.

To further inspect and quantify the results, we also present the metrics for measuring the quality of the Pareto set in Table 3. Our proposed algorithm outperforms the baseline significantly in terms of coverage (as can be seen on the visualization) also in terms of hypervolume, following the visualization. However, we observe that the spacing of our method nearly doubles in the MovieLens dataset, but perform similarly on the Amazon Book dataset.

6.2 Three objectives

For better visualization, we project the three-dimensional Pareto fronts on two objectives. Results are available in Figure 4. Still, from the plots we observe an improvement in all the three combination of objectives.

The Table 4 quantifies the improvement of our proposed method compared to the vanilla SMSGDA. We can see that the *Adamize* trick on our method dominates approximately half the solutions found by the vanilla SMSGDA, while being slightly more spread over the space. In terms of hypervolume, the vanilla SMSGDA performs slightly better.



(a) Revenue vs Recall. (b) Recency vs Recall.

(c) Recency vs Revenue.

Figure 4: Pareto fronts for MovieLens on three objectives.

Method	Hypervolume	Coverage	Spacing
Vanilla	**17.54**	0	0.15
Adamized	17.02	**0.49**	**0.24**

Table 4: Pareto front metrics for MovieLens on three objectives.

However, the difference is less significant than the two objectives case because of the curse of dimensionality.

Supported by the improvements on two different datasets, and using up to three different objectives, we can say that the *Adamize* trick leads on average to better solutions.

7 Conclusion

In this paper we introduced a novel method for multi-gradient descent that leverages a momentum-based optimizer. We applied the method on a problem with a growing importance - Multi-objective Recommender Systems. We benchmarked the novel optimization method against the state-of-the-art multi-gradient descent method and reported the results on three different metrics based on the resulting Pareto front: *hypervolume*, *coverage*, and *spacing*. The results show that the new Pareto fronts are significantly better from all three perspectives. We complemented the analysis with a visualization of the Pareto fronts that further emphasizes the gains obtained.

To the best of our knowledge, we are the first to use a momentum-based optimizer for each objective in a multi-objective setup. We hope that this will inspire research practitioners to test and produce other ideas in the direction of using momentum-based optimizers per objective in a multi-objective setup. Improving the gradient-based optimization could benefit all the multi-objective optimization problems, in all applicable fields.

References

Carmel, D.; Haramaty, E.; Lazerson, A.; and Lewin-Eytan, L. 2020. Multi-Objective Ranking Optimization for Product Search Using Stochastic Label Aggregation. In *Proceedings of The Web Conference 2020*, 373–383.

Chakraborty, A.; Ghosh, S.; Ganguly, N.; and Gummadi, K. P. 2017. Optimizing the recency-relevancy trade-off in online news recommendations. In *Proceedings of the 26th International Conference on World Wide Web*, 837–846.

Chen, Z.; Badrinarayanan, V.; Lee, C.-Y.; and Rabinovich, A. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, 794–803.

Désidéri, J.-A. 2012. Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique* 350(5-6): 313–318.

Ding, Y.; Li, X.; and Orlowska, M. E. 2006. Recency-based collaborative filtering. In *Proceedings of the 17th Australasian Database Conference-Volume 49*, 99–107.

Gabriel De Souza, P. M.; Jannach, D.; and Da Cunha, A. M. 2019. Contextual hybrid session-based news recommendation with recurrent neural networks. *IEEE Access* 7: 169185–169203.

Harper, F. M.; and Konstan, J. A. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5(4): 1–19.

Hatcher, T. 2000. The social responsibility performance outcomes model building socially responsible companies through performance improvement outcomes. *Performance Improvement* 39(7): 18–22.

He, R.; and McAuley, J. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, 507–517.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, 173–182.

Hinton, G.; Srivastava, N.; and Swersky, K. 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on* 14(8).

Karatzoglou, A.; Baltrunas, L.; and Shi, Y. 2013. Learning to rank for recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems*, 493–494.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Liang, D.; Krishnan, R. G.; Hoffman, M. D.; and Jebara, T. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, 689–698.

Loshchilov, I.; and Hutter, F. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* .

Milojkovic, N.; Antognini, D.; Bergamin, G.; Faltings, B.; and Musat, C. 2019. Multi-Gradient Descent for Multi-Objective Recommender Systems. *arXiv preprint arXiv:2001.00846* .

Nguyen, T. T.; Hui, P.-M.; Harper, F. M.; Terveen, L.; and Konstan, J. A. 2014. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web*, 677–686.

Okabe, T.; Jin, Y.; and Sendhoff, B. 2003. A critical survey of performance indices for multi-objective optimisation. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, volume 2, 878–885. IEEE.

Peitz, S.; and Dellnitz, M. 2018. Gradient-based multiobjective optimization with uncertainties. In *NEO 2016*, 159–182. Springer.

Poirion, F.; Mercier, Q.; and Désidéri, J.-A. 2017. Descent algorithm for nonsmooth stochastic multiobjective optimization. *Computational Optimization and Applications* 68(2): 317–331.

Sener, O.; and Koltun, V. 2018. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, 527–538.

Srivastava, N. 2013. Improving neural networks with dropout. *University of Toronto* 182(566): 7.

Varona, M. 2020. Incentives to Encourage Companies to Become Socially Responsible. *Nuevas Tendencias* (103): 30–40.

Vveinhardt, J.; and Andriukaitiene, R. 2014. Readiness of companies to become socially responsible: social behaviour of an organization and an employee from a demographic viewpoint. *Problems and perspectives in management* (12, Iss. 2 (contin.)): 215–229.

Zitzler, E.; Brockhoff, D.; and Thiele, L. 2007. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In *International Conference on Evolutionary Multi-Criterion Optimization*, 862–876. Springer.

Zitzler, E.; and Thiele, L. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation* 3(4): 257–271.