Fisher-Weighted Merge of Contrastive Learning Models in Sequential Recommendation

Jung Hyun Ryu [*1] **Jaeheyoung Jeon** [*2] **Jewoong Cho** [*2] **Myungjoo Kang** [1 2]

Abstract

Along with the exponential growth of online platforms and services, recommendation systems have become essential for identifying relevant items based on user preferences. The domain of sequential recommendation aims to capture evolving user preferences over time. To address dynamic preference, various contrastive learning methods have been proposed to target data sparsity, a challenge in recommendation systems due to the limited user-item interactions. In this paper, we are the first to apply the Fisher-Merging method to Sequential Recommendation, addressing and resolving practical challenges associated with it. This approach ensures robust fine-tuning by merging the parameters of multiple models, resulting in improved overall performance. Through extensive experiments, we demonstrate the effectiveness of our proposed methods, highlighting their potential to advance the state-of-the-art in sequential learning and recommendation systems.

1. Introduction

With the exponential growth of online platforms and services, a significant amount of data is being generated daily. Recommendation systems have become crucial in utilizing this data effectively. The system aim to identify relevant items based on user preferences and interests. As user preferences evolve over time, sequential recommendation has gained attention as a subfield in this area. We address the problem of sequential recommendation as follows.

Let $\mathcal{U}$ be the set of users $\mathcal{U} = \{u_1, u_2, \cdots, u_{|\mathcal{U}|}\}$, and $\mathcal{V}$ be the set of items as $\mathcal{V} = \{v_1, v_2, \cdots, v_{|\mathcal{V}|}\}$. The sequence

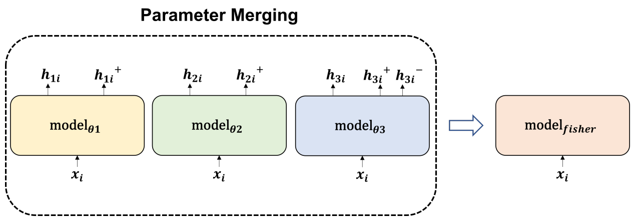

Figure 1. Overview of Parameter Merging. Sharing model architecture, each models differ in how the contrastive loss is constructed. We merge models by using a weighted sum, where the weights are determined based on the posterior distribution of each model's parameters, assuming a Gaussian distribution.

of user-item interaction for u_i is a list with chronological order, $s_i = [v_1^{u_i}, v_2^{u_i}, \cdots, v_t^{u_i}, \cdots, v_{n_{u_i}}^{u_i}]$. Here user $u_i \in \mathcal{U}$, $v_t^{u_i} \in \mathcal{V}$, and user u_i interact item $v_t^{u_i}$ in time step t. The length of sequence for user u_i is n_{u_i}, and our object is to build a model predicting the item with which user is interact in the next time step, i.e.

$$p\left(v_{n_{u_i}+1}^{u_i} = v \,\middle|\, s_i\right). \tag{1}$$

The previous methodologies typically employ similar model structures but utilize various learning frameworks (Xie et al., 2022; Qiu et al., 2022). Prior research has shown that ensemble methods yield significant benefits when multiple learning frameworks are employed (Gontijo-Lopes et al., 2021).

We propose a practical and feasible method to ensemble the parameters of models trained with different contrastive learning techniques in a sequential recommendation. The purpose of this study is to effectively aggregate the obtained parameters θ in various learning frameworks and hyperparameter settings, building on previous research and experiments.

Assuming the posterior distribution of parameters θ_m for each m-th model, Sec 3.1, we achieved more effective ensemble results. This approach allowed us to capture the uncertainty associated with each model's parameter estimates and leverage this information to enhance the ensem-

[*]Equal contribution [1]Interdisciplinary Program in Artificial Intelligence, Seoul National University, Seoul, Korea [2]Department of Mathematics, Seoul National University, Seoul, Korea. Correspondence to: Myungjoo Kang <mkang@snu.ac.kr>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

ble process. By considering the posterior distributions, we were able to account for the variability in parameter values across different models and obtain a more robust and comprehensive ensemble outcome.

2. Related Works

Researchers have explored various ensemble methods, including bootstrapping, bagging, and boosting, to improve model performance (Ganaie et al., 2022; Breiman, 1996; 2001; Natekin & Knoll, 2013; Liu et al., 2014). Ovadia et al. (Ovadia et al., 2019) demonstrated the accuracy of ensembles even in the presence of distribution shift, while Mustafa et al. (Mustafa et al., 2020) proposed a method that combines fine-tuned subsets of pre-trained models to achieve high accuracy and resilience to distribution shift. Parameter merging is another technique to reduce model size and computational requirements (Houlsby et al., 2019). However, ensemble methods often require additional training, which can be computationally expensive and time-consuming.

2.1. Diverse Learning Framework

Wenzel et al. (2020) and Zaidi et al. (2021) investigated the role of random hyperparamters and architectures in ensemble. Gontijo et al. (2021) demonstrated the ensemble effect across various training methodologies; initialization, hyperparameter, architecture, framework, and dataset levels. Diverse training methodologies exhibit different generalization capabilities, ultimately lead to uncorrelated errors. Models tend to specialize in subdomains within the data and highlights the crucial role of ensemble techniques in enhancing overall performance.

2.2. Merging Methods

Model Soup Model Soup (Wortsman et al., 2022) presents an effective approach for combining parameters without additional training. It demonstrates research findings that improve the performance of trained models by constructing a ”recipe” composed of diverse models and averaging their parameters. The study introduces three methods for creating the recipe: the uniform soup, which averages the parameter values of all models; the greedy soup, which sequentially adds models based on their performance ranking; and the learned soup, which identifies the optimal model interpolation through training. These approaches contribute to enhancing the overall performance of the model without the need for additional training.

Fisher Merging Within the scope of related works, parameter merging is interpreted as a process that maximizes the joint likelihood of model parameters' posteriors (Matena & Raffel, 2022). Previous study (Wortsman et al., 2022) consider averaging as a scenario where the posteriors of these models are assumed to follow an isotropic Gaussian distribution, and the joint likelihood is maximized accordingly. To refine this approach, efforts have been made to approximate the posterior of the model using Laplace approximation (Daxberger et al., 2021). In this case, the distribution of each model is modeled by assuming the mean as the observed, which can be interpreted as trained parameter and the variance as the Gaussian distribution's Fisher matrix. By employing this formulation, the joint likelihood is calculated.

2.3. Sequential Recommendation System

SASRec (Kang & McAuley, 2018) employ Transformer layers to dynamically assign weights to previous items. BERT4Rec (Sun et al., 2019) demonstrate an improvement by incorporating user behavior information from both directions using a bidirectional Transformer. CL4SRec (Xie et al., 2022) employed three data augmentation techniques, namely item cropping, item masking, and item reordering, to create pairs for contrastive learning. DuoRec (Qiu et al., 2022) integrated two types of contrastive loss. Firstly, it incorporated unsupervised augmentation using dropout-based model-level augmentation to generate positive pairs. Secondly, it incorporated supervised positive sampling, which involves creating pairs by considering sequences with the same target item as positive samples.

3. Methodology

We perform model ensemble based on different types of loss functions. BERT4Rec (Sun et al., 2019), CL4SRec (Xie et al., 2022), and DuoRec (Qiu et al., 2022) share the basic structure of BERT4Rec (Sun et al., 2019). However, they differ in the sense of constructing positive pairs, a key component of their learning framework of constrastive learning.

Figure 1 represents the overview of parameter merging process. By sharing the structure of the model, which is parameterized with diverse learning frameworks, we can leverage ensemble methods to our advantage. Furthermore, inspired by previous studies demonstrating the effectiveness of ensemble models trained using various learning methods, we apply parameter merging techniques, namely Parameter Averaging and Fisher-weighted Parameter Merging, described in Section 3.1, to combine these models.

3.1. Understanding Model Ensemble

We follow the work of Matena et al (2022). Consider a scenario where we have models with the same structure, $\text{model}_1, \text{model}_2, \cdots, \text{model}_M$, with corresponding parameters $\theta_1, \theta_2, \cdots, \theta_M$. Our objective is to find the parameter θ^* that maximizes the joint likelihood of the posteriors of these parameters.

The posterior of θ_m can be represented as $p\left(\theta|\theta_m\right)$. Since obtaining this posterior directly is generally challenging, it can employ approximation methods such as Laplace approximation to make assumptions and seek the parameter θ^* (MacKay, 1992; Daxberger et al., 2021). Let us interpret the process of finding θ^* as maximizing the joint likelihood, $\sum_m \log p\left(\theta \mid \theta_m\right)$. Assuming that $p\left(\theta \mid \theta_m\right)$ follows a Gaussian distribution, we set the mean of this Gaussian distribution as the observed θ_m and examine the procedure for averaging parameters and Fisher merging separately, depending on the method used to assume the variance.

Averaging Parameters Assume that the posterior $p\left(\theta \mid \theta_m\right)$ follows a Gaussian distribution $\mathcal{N}\left(\hat{\theta}_m, I\right)$. Here, $\hat{\theta}_m$ represents the parameters of the trained m-th model, and I denotes the identity matrix. In this case, the desired solution θ^* can be obtained as the average of the parameters of the candidate models, as shown in eq.2:

$$\theta^* = \arg\max_\theta \sum_m \log p\left(\theta \mid \theta_m, I\right) = \frac{1}{M} \sum_m \theta_m. \qquad (2)$$

Fisher Merging Let us consider the posterior $p\left(\theta \mid \theta_m\right)$ following a Gaussian distribution $\mathcal{N}\left(\hat{\theta}_m, H^{-1}\right)$. Here, $\hat{\theta}_m$ represents the parameters of the trained m-th model, and H corresponds to the Hessian matrix of θ_m obtained through the second-order Taylor expansion at the mode of the posterior. It has been established that the Hessian matrix in this distribution coincides with the Fisher information, but for computational efficiency, we only utilize the diagonal elements of the Fisher matrix (Kirkpatrick et al., 2017).

The desired solution θ^* can be expressed as eq.3, capturing the essence of the Fisher likelihood :

$$\theta^* = \arg\max_\theta \sum_m \lambda_m \log p\left(\theta \mid \theta_m, F_m\right), \qquad (3)$$

where $F_m = \mathbb{E}_x \mathbb{E}_{y \sim p_\theta(y|x)} \nabla_\theta \log p_\theta\left(y|x\right) \nabla_\theta \log p_\theta\left(y|x\right)^T$. The closed-form solution for θ^* can be obtained as shown in eq.4, which directly incorporates the Fisher matrix. In practice, we utilize an empirical estimate of the Fisher matrix, denoted as $\hat{F}$, as shown eq.4 (Kirkpatrick et al., 2017).

$$\theta^{*(j)} = \frac{\sum_m \lambda_m F_m^{(j)} \theta_m^{(j)}}{\sum_m \lambda_m F_m^{(j)}}, \qquad (4)$$

where $F_m = \frac{1}{N} \mathbb{E}_{y \sim p_\theta(y|x)} \left(\nabla_\theta \log p_\theta\left(y \mid x\right)\right)^2$ and $j = 1, \cdots, |\theta|$, considering as element-wise multiplication.

3.2. Applying Model Ensemble

By expressing the Fisher matrix we intend to compute in eq.4 in terms of recommendation factors, we can decompose it into the following components:

$$\begin{aligned} &\mathbb{E}_{x_i} \mathbb{E}_{y \sim p_\theta(y|x_i)} \left(\nabla_\theta \log p_\theta\left(y \mid x_i\right)\right)^2 \\ &= \frac{1}{N} \sum_i \sum_j p_\theta\left(y_j \mid x_i\right) \left(\nabla_\theta \log p_\theta\left(y_j \mid x_i\right)\right)^2 \\ &= \frac{1}{|\mathcal{U}|} \sum_i^{|\mathcal{U}|} \sum_j^{|\mathcal{V}|} p_\theta\left(v_j \mid s_i\right) \left(\nabla_\theta \log p_\theta\left(v_j \mid s_i\right)\right)^2. \end{aligned} \qquad (5)$$

There are two computational challenges associated with the above equation. First, calculations need to be performed for each individual sample s_i. Second, calculations need to be performed for each item v_j within a single sample. The reason why these points acts as a drawback in recommendation systems is due to the large number of users and items in the data. For instance, in the case of MovieLens-1M dataset (Harper & Konstan, 2015), there are about 6000 users and 3500 items. However, performing Fisher matrix calculations that require differentiation with respect to θ for each user and item becomes a computational burden.

3.2.1. Sampling sequences

Batch-wise Computation To address the first challenge of performing computations on individual samples, we reinterpret the equation and carry out the calculations on a batch basis. It should be noted that $p_\theta\left(v_j|s_i\right)$ can vary for each sample s_i. Therefore, we perform the sorting of $p_\theta\left(v_j|s\right)$ to address this variation, where BS indicates batch size:

$$\begin{aligned} &\sum_i^{|\mathcal{U}|} \sum_j^{|\mathcal{V}|} p_\theta\left(v_j \mid s_i\right) \left(\nabla_\theta \log p_\theta\left(v_j \mid s_i\right)\right)^2 \\ &= \sum_{\text{BS}_k} \sum_j^{|\mathcal{V}|} \left(\sum_i^{\text{BS}_k} p_\theta\left(v_j|s_i\right)\right) \left(\nabla_\theta \sum_i^{\text{BS}_k} \log p_\theta\left(v_j|s_i\right)\right)^2. \end{aligned} \qquad (6)$$

3.2.2. Sampling items

To alleviate the computational burden associated with iterating over all j values, which scales with $|\mathcal{V}|$, we employ a sampling-based approach within the methodology. This sampling strategy aims to reduce the computational cost while maintaining the representativeness of the calculations.

Random Sampling We compute the eq.3.2 by randomly sampling j from the total number of items. This process was performed to calculate the Fisher matrix without any specific assumptions or prior knowledge.

Top-k Sampling The probability which is output by the model can be interpreted as the preference or likelihood of the recommended items for a given sample. Based on this interpretation, we select a set of n items that are most likely to be of interest to the corresponding user, i.e. $p_\theta(v_j \mid s_i)$. Subsequently, we compute the Fisher matrix with these selected items as the focal points. By focusing on this subset of items that are expected to be of highest interest, we aim to capture the relevant information for optimizing the model's performance effectively.

$$\begin{aligned}&\sum_{j}^{|\mathcal{V}|} p_\theta(v_j \mid s)\left(\nabla_\theta \log p_\theta(v_j \mid s)\right)^2 \\ &\approx \sum_{j}^{\text{top-}k} p_\theta(v_j \mid s)\left(\nabla_\theta \log p_\theta(v_j \mid s)\right)^2 .\end{aligned} \tag{7}$$

Model-based Sampling To select a subset of items for further analysis, we randomly sampled items based on their conditional probability $p_\theta(v_j|s_i)$ using a weighted random selection process. The selection probability of each item was determined by its associated probability stored in the model's output. By selecting items with higher probabilities, we focused on a specific number of items that were more likely to align with the user's preferences or interests. This allowed us to analyze and evaluate the subset of items based on their associated probabilities obtained from the model's output. With N denotes the sample size, his approximation can be represented as:

$$\begin{aligned}&\mathbb{E}_{y\sim p_\theta(y|x)}\left(\nabla_\theta \log p_\theta(y \mid x)\right)^2 \\ &\approx \frac{1}{N}\sum_{v_j\sim p_\theta(v_j|s)}^{N}\left(\nabla_\theta \log p_\theta(v_j \mid s)\right)^2 .\end{aligned} \tag{8}$$

Calculate with target item We compute the Fisher matrix based on the target item, disregarding other items with limited direct relevance. By employing this approach, we focus solely on the target item and its associated information to calculate the Fisher matrix. Our rationale behind this decision is to prioritize the target item's impact on the model's optimization process, as it is directly linked to the specific objective or task at hand. Consequently, we exclude items with minimal direct relevance to ensure a more targeted and meaningful computation of the Fisher matrix.

$$p_\theta\left(v_j^* \mid s\right)\left(\nabla_\theta \log p_\theta\left(v_j^* \mid s\right)\right)^2 , \tag{9}$$

where v_j^* is the target item.

4. Experiments

We use MovieLens-1M dataset (Harper & Konstan, 2015) for experiments. For each user, we have sequential data consisting of movies purchased in chronological order. We adopt next-item prediction task (i.e. leave-one-out evaluation), following previous works (Sun et al., 2019; Xie et al., 2022; Qiu et al., 2022). The last movie is considered as the test set, and the validation data is used to predict the preceding movies. During training, we adopt a masked language modeling approach similar to BERT (Devlin et al., 2018), where we mask certain movies in the sequentially ordered list and task the model with predicting them.

The evaluation method used in this study is the Normalized Discounted Cumulative Gain at 10 (NDCG@10), which is a ranking-based evaluation approach (He et al., 2017). It ranks the top 10 items predicted by the model based on their perceived preference and considers the actual ranking of the preferred items. A higher NDCG value, closer to 1, indicates better performance. Different NDCG values can be obtained depending on the selection of items, such as from the full item pool, a random set of 100 items, or the top 100 most popular items.

4.1. Results of Model Merging

Examine the results through Table 1 and Table 2. Table 1 presents the results obtained by training models, namely BET4Rec (Sun et al., 2019), CL4SRec (Xie et al., 2022), and DuoRec (Qiu et al., 2022). We merge these models using Fisher methods. While Table 1 demonstrates the results of models trained solely from scratch. Table 2 represents the results of fine-tune setting. We train the baseline model without contrastive loss for 20 epochs, which is the convergence point of the baseline experiment without any additional contrastive loss, similar to BERT4Rec. Following this, each model; BERT4Rec (Sun et al., 2019), CL4SRec (Xie et al., 2022), DuoRec (Qiu et al., 2022), underwent fine-tuning according to their respective methods, and the results were merged using Fisher methods. In both conditions, we fine-tuned addtional epoch after merging process.

Fisher merge fails to improve the performance of individual models in baseline setting. When Fisher merge is applied during the fine-tuning setting, it leads to improved performance compared to individual models. This finding aligns with previously reported phenomena (Ganaie et al., 2022) where individual models tend to achieve higher performance than merged model in the baseline setting. However, the results of Fisher merge in the fine-tuning setting show comparable performance with the individual models in baseline setting, while the individual recipe models of fine-tuning setting do not exceed. Also, the results indicate that even for models that have not been sufficiently trained such as CL4SRec in our setting, merging parameters resulted in comparable performance to other models, demonstrating robustness.

Table 1. Results of parameter merging; Fisher-merge on Baseline Settings. The recipes used for merging were trained for the same number of epochs. 'POS.' refers to the method of constructing positive pair, 'sup' to supervised augmentation and '-' to supervised and unsupervised augmentation in subsection 2.3

MODEL	POS	FULL	RANDOM	POPULAR
BASELINE		**0.1398**	**0.5651**	**0.0482**
CL4SREC		0.0955	0.043	0.0429
DUOREC	SUP	0.1348	0.5575	0.044
	UNSUP	0.1301	0.5592	0.0438
	-	0.1382	0.5588	0.0464
FISHER		0.1289	0.5495	0.0472

Table 2. Results of parameter merging; Fisher-merge on fine-tune Settings. The recipes used for merging were trained on baseline model (without contrastive loss) and fine-tuned on each model.

MODEL	POS	FULL	RANDOM	POPULAR
BASELINE		0.135	0.5573	0.0426
CL4SREC		0.0585	0.0513	**0.0466**
DUOREC	SUP	0.1346	0.5547	0.0454
	UNSUP	0.1358	0.5594	0.0445
	-	0.1351	0.554	0.0423
FISHER		**0.1386**	**0.5618**	0.0428

4.2. The Validity of Batch-wise Computation

We performed batch-wise computations with the aim of implementing an efficient Fisher matrix calculation. Compared to computing on individual samples, grouping samples into batches allowed us to achieve computational efficiency.

The following Figure 4 in Appendix A.3 illustrates the method for minimizing errors when performing calculations on a batch basis. The figure demonstrates that within a batch containing 10 samples, denoted as s_i, there is a phenomenon where the probabilities of item v_j decrease in a similar manner. By sorting the samples s_i based on the probability of v_j, even when grouping them into batches, it is possible to minimize the error described by the eq.6. Furthermore, the figure illustrates the rationale behind top-k sampling. For the top-k items, the probabilities hold meaningful information, whereas for the remaining items, the probabilities are nearly zero or close to it.

4.3. Effect of Sampling Methods and Size

To investigate the effect of sampling methods, we conduct experiments by varying the number of sampled samples and the sampling techniques employed. Specifically, we consider three sample sizes: $n = 10, n = 30$ and $n = 50$, and four different sampling methods: random sampling, top-k sampling, model-based sampling, and calculate with target item. The results of these experiments can be observed in Table 3. The table provides insights into the performance of each sampling method under different sample sizes, allowing us to analyze their respective effects on the task at hand. Note that this result is calculated on batched data. To examine the results of parameter merging, we conducted experiments in fine-tuning setting, explained in 2.

The experiments revealed effective ensemble results, particularly showcasing the robust performance of CL4SRec (Xie et al., 2022). Despite having significantly lower performance compared to other models during the parameter merging process, the model with poor performance exhibited robust performance in the Fisher merge results. Regarding the sampling methods, top-k sampling demonstrated the best performance. This can be attributed to the concentration of probabilities assigned to specific items by the model, effectively approximating the Fisher criterion sought in the evaluation. Also, the model-based sampling method exhibits a more pronounced improvement in performance as the sampling size increases compared to other models. We interpret these results as being rooted in the direct interpretation of the equation defined for Fisher merging. Interestingly, despite the fact that calculating Fisher matrix on target item has a single sample, the method demonstrated sampling effectiveness by achieving good performance even with a small sample size. These findings shed light on the interpretation of experimental results in the context of deep learning research.

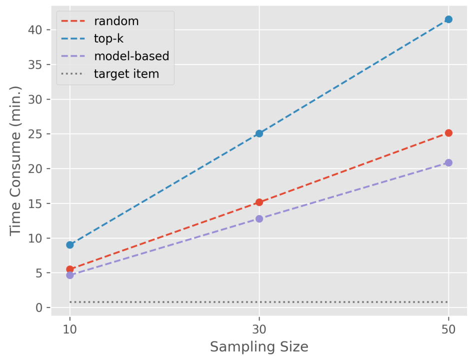

Figure 2. Measured Time Consumed for Each Sampling Method and Size. We sample items from MovieLens-1M dataset. Time is measured in batch-wise setting, where batch size is 256.

4.4. Computational Cost

Figure 2 demonstrates computational cost in terms of time consumed during calculating Fisher matrix for single model. The concept of parameter merging involves additional computation on the existing parameters. Therefore, it is important to ensure efficiency in this process. To achieve efficiency, considerations such as calculating the Fisher ma-

Table 3. Effect of Sampling Method and Sampling Size. We merge models in settings of Table 2, the fine-tune setting. We merged models with 4 sampling methods; random sampling, top-k sampling, model-based sampling, and calculate on target item, on 3 different sampling size; n=10, n=30 and n=50. **Bold** represents the best variant in each evaluation setting, and <u>underlines</u> indicates the second best variation.

	SAMPLE SIZE	FULL		RANDOM		POPULAR	
		NDCG10	NDCG20	NDCG10	NDCG20	NDCG10	NDCG20
BASELINE		0.135	0.1601	0.5573	0.5786	0.0426	0.0706
CL4SREC		0.0585	0.0751	0.0513	0.043	**0.0466**	0.0701
DUOREC (SUP.)		0.1346	0.1591	0.5547	0.58	0.0454	0.068
DUOREC (UNSUP.)		0.1358	0.1609	0.5594	0.5782	0.0445	0.0742
DUOREC (SUP.&UNSUP.)		0.1351	0.1599	0.554	0.5732	0.0423	0.0724
RANDOM SAMPLING	10	0.1379	<u>0.1638</u>	0.5606	<u>0.5825</u>	0.0457	0.0691
	30	0.1366	0.1624	0.5584	0.58	0.0477	**0.0726**
	50	<u>0.1386</u>	0.1636	0.5598	0.5813	0.0419	0.0419
TOP-K SAMPLING	10	0.1364	0.1624	0.5602	0.5817	0.0446	0.0689
	30	0.1373	0.1616	**0.5637**	**0.5835**	0.0457	0.0708
	50	**0.1387**	0.1635	0.5592	0.5807	0.0424	0.0672
MODEL-BASED SAMPLING	10	0.1358	0.1619	0.5564	0.5782	0.044	0.0696
	30	0.1385	**0.1646**	0.5579	0.5784	0.0446	0.0689
	50	0.138	0.1632	0.5605	0.5814	<u>0.0465</u>	0.0719
CALCULATE ON TARGET ITEM	1	<u>0.1386</u>	0.1628	<u>0.5618</u>	0.5806	0.0428	<u>0.0725</u>

trix in batch units and performing sampling are necessary. It is observed that, except for the calculation on the target item, the computational complexity increases linearly with the sampling size. As for the calculation on the target item, the sampling size remains fixed at 1 since each sequence has a single target item. Thus, our research is significant as it approximates the Fisher matrix calculation with a much smaller number of items (around 3000) compared to calculating it on the entire item set.

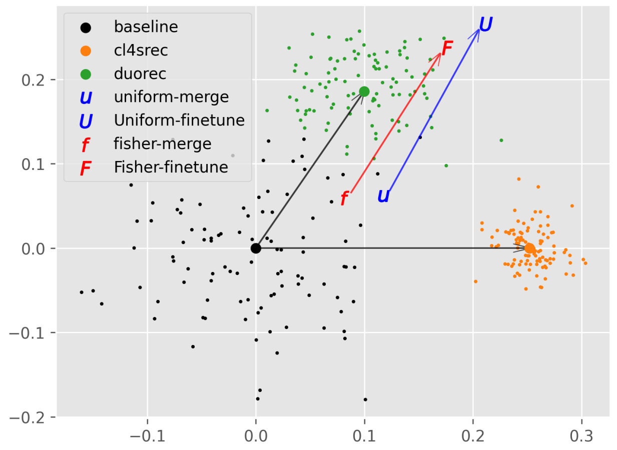

Figure 3. Visualization for Weights of Merged Models. Based on the plane containing 64-dimension parameters of three model, we visualised its weight, 100 samples from each posteriors and merged parameters

4.5. Visualization of Merged Weights

We present a visual illustration to aid in the intuitive understanding of the merged weights. Figure 3 represents the fine-tuning setting of 2, where the three centroids correspond to the weights of individual models. The plane visualized in 3 encompasses these three weights. The scattered points, projected onto the plane, depict 100 samples drawn from $\mathcal{N}(\theta_m, F_m)$. It is observed that the baseline weight exhibits the largest variance. This can be attributed to the experimental setup where the baseline is pre-trained and then fine-tuned with CL4SRec (Xie et al., 2022) and DuoRec (Qiu et al., 2022). The weights obtained through uniform merging are represented as the average of the three centroid points, while the weights obtained through Fisher merging take into account the variances of these recipe weights. It can be seen that the weights obtained through Fisher merging considered posterior and variance with Laplace approximation and provides nice initial point for fine-tune.

5. Conclusion

We apply ensemble technique, Fisher merging, for sequential models, enabling robust fine-tuning through parameter merging. Our experimental results demonstrate the effectiveness of these proposed methods in improving recommendation performance. These contributions have the potential to advance the field of sequential learning and recommendation systems, offering valuable insights for future research and practical applications.

Acknowledgements

This work was support by the NRF grant [2012R1A2C3010887] and the MSIT/IITP [1711117093, 2021-0-00077, 2021-0-01343, Artificial Intelligence Graduate School Program(SNU)].

References

Breiman, L. Bagging predictors. *Machine learning*, 24: 123–140, 1996.

Breiman, L. Random forests. *Machine learning*, 45:5–32, 2001.

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., and Hennig, P. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34:20089–20103, 2021.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Ganaie, M. A., Hu, M., Malik, A., Tanveer, M., and Suganthan, P. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115:105151, 2022.

Geirhos, R., Meding, K., and Wichmann, F. A. Beyond accuracy: quantifying trial-by-trial behaviour of cnns and humans by measuring error consistency. *Advances in Neural Information Processing Systems*, 33:13890–13902, 2020.

Gontijo-Lopes, R., Dauphin, Y., and Cubuk, E. D. No one representation to rule them all: Overlapping features of training methods. *arXiv preprint arXiv:2110.12899*, 2021.

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

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

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Kang, W.-C. and McAuley, J. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pp. 197–206. IEEE, 2018.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Liu, P., Han, S., Meng, Z., and Tong, Y. Facial expression recognition via a boosted deep belief network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1805–1812, 2014.

MacKay, D. J. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Matena, M. S. and Raffel, C. A. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

Mustafa, B., Riquelme, C., Puigcerver, J., Pinto, A. S., Keysers, D., and Houlsby, N. Deep ensembles for low-data transfer learning. *arXiv preprint arXiv:2010.06866*, 2020.

Natekin, A. and Knoll, A. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

Qiu, R., Huang, Z., Yin, H., and Wang, Z. Contrastive learning for representation degeneration problem in sequential recommendation. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pp. 813–823, 2022.

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., and Jiang, P. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 1441–1450, 2019.

Wang, T. and Isola, P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pp. 9929–9939. PMLR, 2020.

Wenzel, F., Snoek, J., Tran, D., and Jenatton, R. Hyperparameter ensembles for robustness and uncertainty quantification. *Advances in Neural Information Processing Systems*, 33:6514–6527, 2020.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pp. 23965–23998. PMLR, 2022.

Xie, X., Sun, F., Liu, Z., Wu, S., Gao, J., Zhang, J., Ding, B., and Cui, B. Contrastive learning for sequential recommendation. In *2022 IEEE 38th international conference on data engineering (ICDE)*, pp. 1259–1273. IEEE, 2022.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., and Lipson, H. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

Zaidi, S., Zela, A., Elsken, T., Holmes, C. C., Hutter, F., and Teh, Y. Neural ensemble search for uncertainty estimation and dataset shift. *Advances in Neural Information Processing Systems*, 34:7898–7911, 2021.

A. Appendix

A.1. Motivation : Error Inconsistency

Table 4. Effect of Constructing Pair of Contrastive Loss (%). We observe that models with divergent training methodologies exhibit distinct generalization behavior, resulting in highly uncorrelated errors.

	SIMILAR		DISSSIMILAR
CL4SREC (XIE ET AL., 2022)	8.05	$<$	11.41
DUOREC (QIU ET AL., 2022)	8.67	$<$	11.18

Previous research (Gontijo-Lopes et al., 2021; Yosinski et al., 2015) demonstrated the increased effectiveness of ensemble methods as error inconsistency grows. Building upon the existing research discourse, we conducted the current experiment. In this study, we analyze the impact of Fisher merging in the context of sequential recommendation systems, attributing its effectiveness to the selection of recipe models trained using different frameworks .

In our experiments, we employ a model based on the BERT4Rec (Sun et al., 2019) architecture as our baseline. To enhance the performance of the model, we apply various data augmentation techniques to enable contrastive learning.

To analyze the effects of contrastive loss, we divide the training frameworks into two categories: similar and dissimilar. The similar learning frameworks are trained using the same loss function but with slight variations such as different seeds and hyperparameters, indicating the relationship among models trained with small changes. On the other hand, the dissimilar learning frameworks involve different data augmentation techniques, resulting in variations in the construction of positive and negative pairs for contrastive loss (Wang & Isola, 2020).

Error inconsistency (Geirhos et al., 2020) refers to the percentage of data where two models have different classification results, with one model making correct predictions while the other model makes incorrect predictions. Since we are not dealing with classification, we considered a model to have made a correct prediction if the value of NDCG@10 is above 0.5.

By comparing the error inconsistency between similar framework and dissimilar framework, we observe the effectiveness of contrastive loss. An observation that can be inferred from Table 4 is that the constructing positive pair for contrastive loss significantly affects the similarity of the samples that the models predict accurately. As the method for constructing positive pair varies, the models demonstrate a considerable difference in their ability to predict samples correctly. This finding highlights the sensitivity of the models to the specific construction of the contrastive loss, which in turn impacts their predictive performance.

A.2. Robustness of Fisher Merging; Recipe Selection

We compared two different recipe selection in Table 5; Fisher merged parameters with least performance model and Fisher merged parameters without the model. In our experimental setup, CL4SRec did not exhibit superior performance compared to other models, considering the chosen hyperparameter settings and other factors. Therefore, we aim to leverage the elements of the recipe to demonstrate the robustness effect of Fisher merge. Our findings confirm that by removing underperforming models as individual components and applying Fisher merge, the resulting ensemble demonstrates robustness.

Table 5. Ablation Study of Results of parameter merging; Fisher-merge on fine-tune Settings. The recipes used for merging were trained on baseline model (without contrastive loss) and fine-tuned on each model. Ablation Study illustrates the situation of recipe without the model with least performance.

MODEL	POS	FULL	RANDOM	POPULAR
BASELINE		0.135	0.5573	0.0426
(CL4SREC)		(0.0585)	(0.0513)	(<u>0.0466</u>)
DUOREC	SUP	0.1346	0.5547	<u>0.0454</u>
	UNSUP	0.1358	0.5594	0.0445
	-	0.1351	0.554	0.0423
FISHER (WITH)		**0.1386**	**0.5618**	0.0428
FISHER (W.O.)		<u>0.1373</u>	<u>0.5603</u>	**0.0487**

A.3. Visualization of Sorted Probability

The figure displays the sorted probabilities of the top 50 items for 10 sequences, where single line represents single sequence. The cumulative probability values for sample sizes of 10, 30, and 50 are 0.381, 0.569, and 0.658, respectively. With the exception of a few largest ones, the majority of probabilities approximate 0.

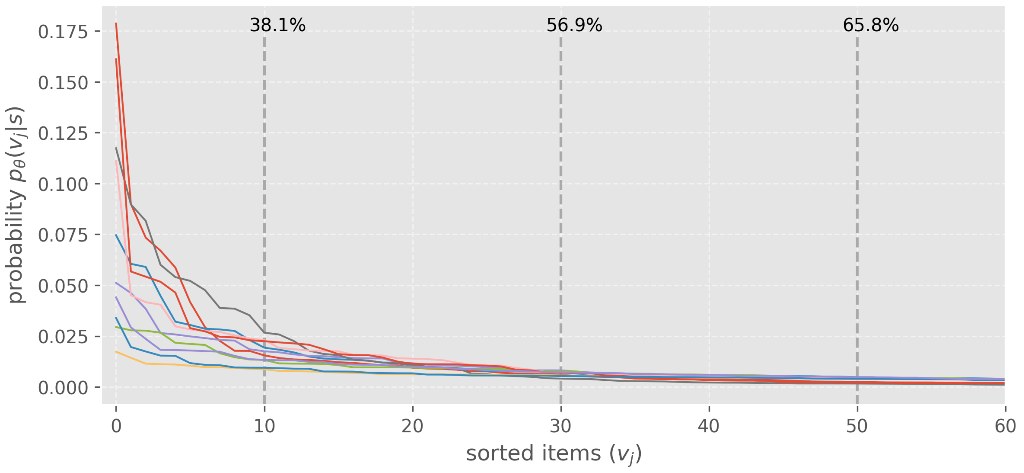

Figure 4. Sorted Probability $p_\theta\left(v_j|s_i\right)$.