Collaborative Metric Learning

2023 Winter DSAIL Internship

한재민

ISysE KAIST
Industrial & Systems Engineering

Contents

- Introduction
- Background
- Method
- Experiments
- Conclusion
- Implementation

Introduction

Metric

$(a)\ \ d(p,q) > 0\ if\ p \neq q;\ d(p,p) = 0;$

$(b)\ \ d(p,q) = d(q,p)$

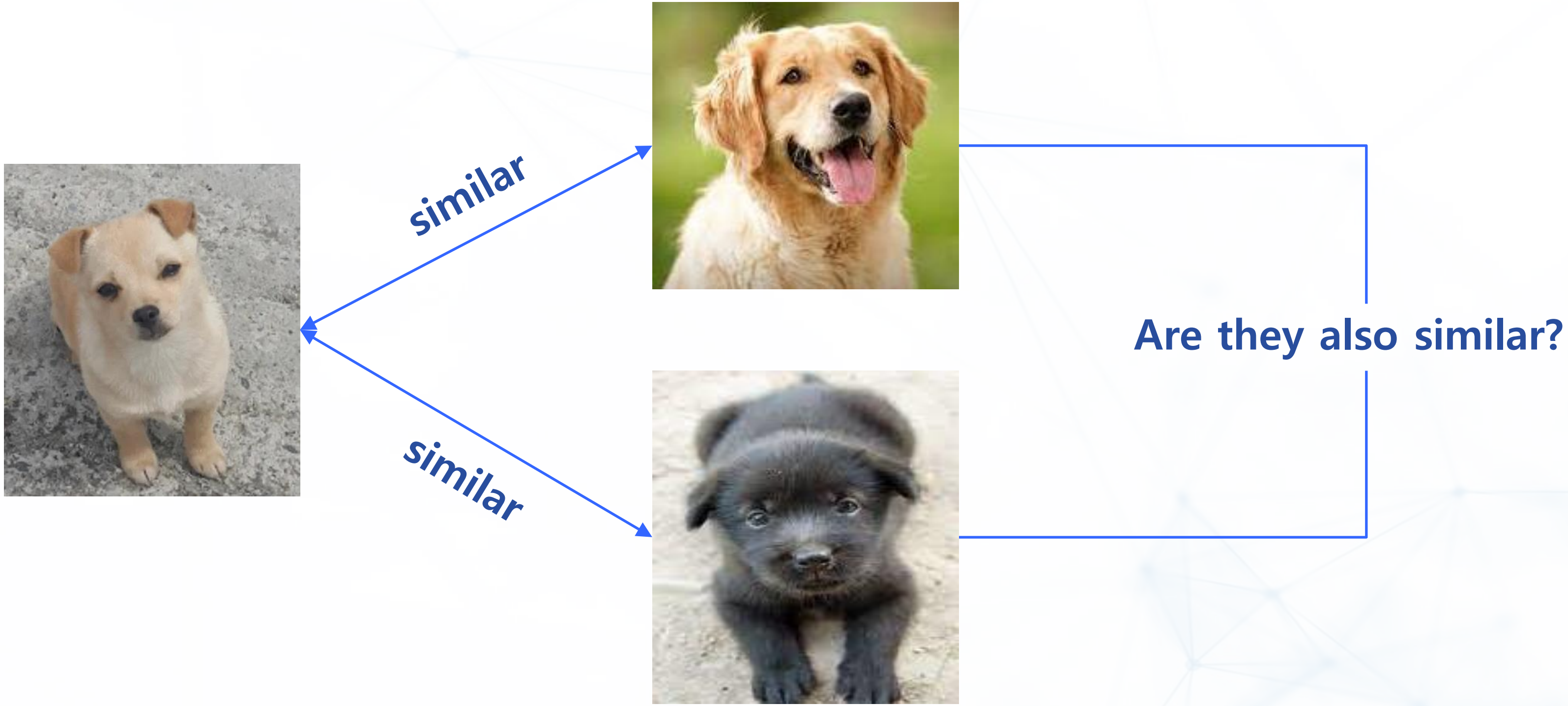
$(c)\ \ d(p,q) \leq d(p,r)\ + d(r,q)$ **Triangle Inequality**

metric = distance function

e.g. Manhattan Distance, Euclidean distance

Introduction

Similarity



Introduction

Similarity

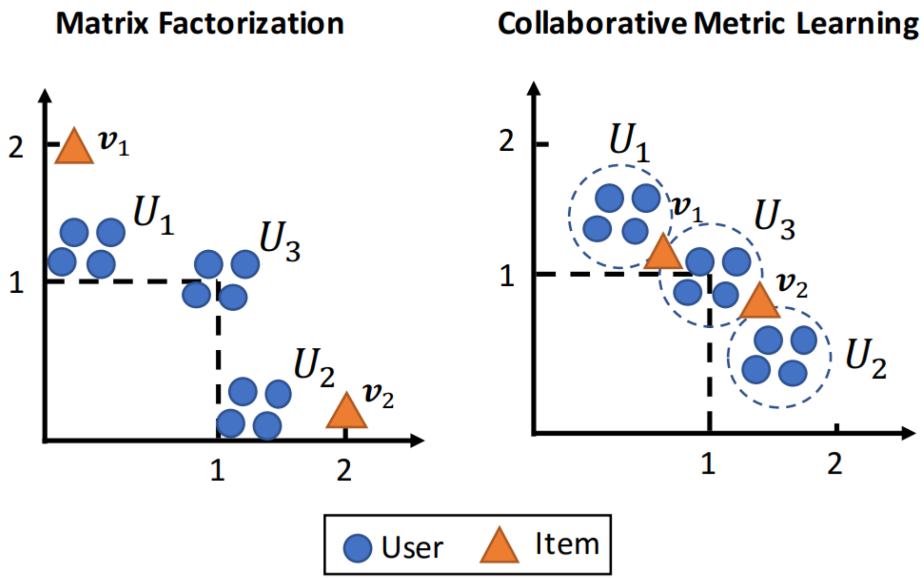
If we make a similarity metric, we can say that (previous slide) the pictures are also similar due to triangle inequality.

$$\because\ d(pic_2, pic_3) \leq d(pic_1, pic_2) + d(pic_1, pic_3)$$

→ Similarity propagation

Introduction

Matrix Factorization



	v_1	v_2
U_1	✓	
U_2		✓
U_3	✓	✓

Does not meet triangle inequality(dot product)

→ suboptimal performance

Background

Metric Learning

Distance is mostly used for representing relationship in ML field.

Metric Learning is an algorithm of achieving a suitable distance function which represents relationships between data well as we intended.

Background

Metric Learning

Label-specified Data

$$S = \{(x_i, x_j) |\ x_i \ and\ x_j\ are\ considered\ similar\}$$

$$D = \{(x_i, x_j) |\ x_i \ and\ x_j\ are\ considered\ dissimilar\}$$

$$d_A(x_i, x_j) = \sqrt{(x_i - x_j)^T A (x_i - x_j)}$$

Mahalanobis distance

Goal

$$\min_A \sum_{(x_i, x_j) \in \mathcal{S}} d_A(x_i, x_j)^2$$

$$\text{s.t.} \quad \sum_{(x_i, x_j) \in \mathcal{D}} d_A(x_i, x_j)^2 \geq 1 \text{ and } A \succeq 0.$$

Find such A

Background

Metric Learning for kNN

Pulling all similar pairs and pushing dissimilar pairs is not always feasible.

Make each object's k-nearest neighbors be the objects that share the same class label with that object.

Terminology

Target neighbors of x : Data points we desire to be the closest to x

Impostors : The differently labeled inputs that invade the perimeter

Background

LMNN(Large Margin Nearest Neighbor)

Two Loss functions

$$\mathcal{L}_{pull}(d) = \sum_{j \rightsquigarrow i} d(x_i, x_j)^2$$

$$\mathcal{L}_{push}(d) = \sum_{i, j \rightsquigarrow i} \sum_{k} (1 - y_{ik})[1 + d(x_i, x_j)^2 - d(x_i, x_k)^2]_+$$

$j \rightsquigarrow i$: j is input i's **target neighbor**

y_{ik} : **indicator** function(whether i and k are of the same class)

$[z]_+$: standard **hinge** loss

Background

Collaborative Filtering – Implicit Feedback

Original MF model

$$\min_{\mathbf{u}_*,\mathbf{v}_*} \sum_{r_{ij}\in\mathcal{K}} (r_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2 + \lambda_u\|\mathbf{u}_i\|^2 + \lambda_v\|\mathbf{v}_i\|^2$$

- Observing only positive feedback
- Cannot regard unobserved interactions as negative

Weighted Regularized Matrix Factorization(WRMF)

$$\min_{\mathbf{u}_*,\mathbf{v}_*} \sum_{r_{ij}\in\mathcal{K}} c_{ij}(r_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2 + \lambda_u\|\mathbf{u}_i\|^2 + \lambda_v\|\mathbf{v}_i\|^2$$

Background

Bayesian Personalized Ranking(BPR)

The notion of “ratings” become less precise in implicit feedback.

→ Modeling the relative preferences between different items

BPR

$$\min_{\mathbf{u}_*, \mathbf{v}_*} \sum_{i \in \mathcal{I}} \sum_{(j,k) \in \mathcal{D}_i} -log\ \sigma(\mathbf{u}_i^T \mathbf{v}_j - \mathbf{u}_i^T \mathbf{v}_k) + \lambda_u \|\mathbf{u}_i\|^2 + \lambda_v \|\mathbf{v}_j\|^2$$

D_i : set of item pairs (j, k) with following

i has interacted with j but not with item k

→ (Assume) user i might be more interested in j than item k

Background

Bayesian Personalized Ranking(BPR)

BPR does not sufficiently penalize the items that are at a lower rank.

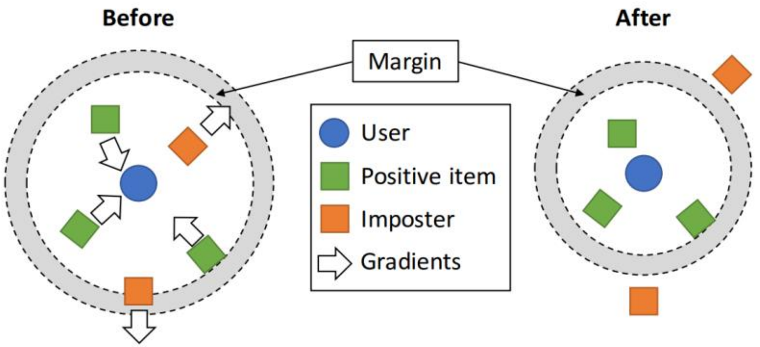
It produces suboptimal results for Top-K recommendation tasks.

Method

Model Formulation

$d(i,j) = \|u_i - v_j\|$ **Euclidean distance**

u_i : **user** vector, v_j : **item** vector



$$\mathcal{L}_m(d) = \sum_{(i,j)\in\mathcal{S}} \sum_{(i,k)\notin\mathcal{S}} w_{ij}[m + d(i,j)^2 - d(i,k)^2]_+$$

j : an item user i **liked**, k : did **not like**

w_{ij} : **ranking loss weight**

m : **safety margin size**

Method

Model Formulation

Difference with LMNN

- User's target neighbors are the items he liked
  - No target neighbor for items

- No pull loss term
  - An item can be liked by many users
  - Push loss pulls positive items (when there are impostors)

- Adopting Weighted ranking loss
  - Improving Top-K recommendations

Method

Approximated Ranking Weight

Weighted Approximate-Rank Pairwise(WARP)

$$w_{ij} = \log(rank_d(i,j) + 1)$$

d : given **metric**
J : **total number** of items
$rank_d(i,j)$: **rank** of item j in user i's recommendation

→ **It penalizes a positive item at a lower rank heavily.**

→ **However, computing $rank_d(i,j)$ at each step is expensive.**

Method

Approximated Ranking Weight

Weighted Approximate-Rank Pairwise(WARP)

Estimate $rank_d(i,j)$ through negative sampling

N times of sampling to find an impostor k

→ approximate $rank_d(i,j)$ as $\left\lfloor \frac{J}{N} \right\rfloor$

For U samples and M impostors,

approximate $rank_d(i,j)$ as $\left\lfloor \frac{J \times M}{U} \right\rfloor$

It follows Geometric distribution.

As it took N times, we can regard it as the mean of this distribution.

Thus, the probability is $\frac{1}{N}$, which implies the number of negatives(↔positive) would be around $\frac{J}{N}$.

We can accept that there are $\frac{J}{N}$ items behind j.

Therefore, the rank can be approximated as $\left\lfloor \frac{J}{N} \right\rfloor$

Method

Integrating Item Features

In metric learning, we obtained a matrix A.

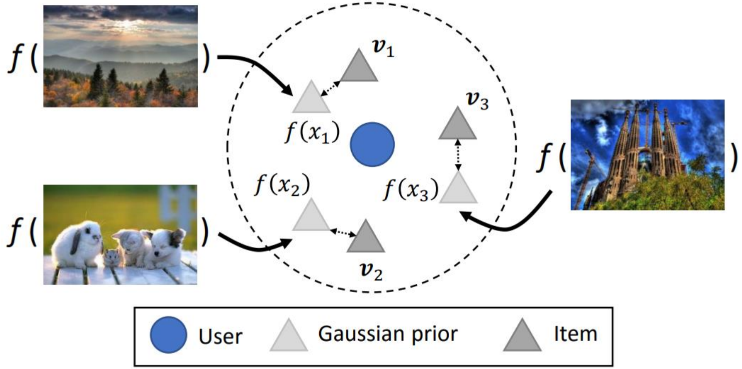
We can regard it a transformation function projecting raw input into Euclidean space.(Fundamental Theorem of Linear Algebra)

→ We can integrate item features into recommendation system.

Method

Integrating Item Features

$$\mathcal{L}_f(\theta, \mathbf{v}_*) = \sum_j \|f(\mathbf{x_j}, \theta) - \mathbf{v}_j\|^2$$



For a raw feature vector x_j of item j, we learn a transformation function f projecting x_j to the joint user-item space.

We treat $f(x_j)$ as a Gaussian prior to v_j and penalize j when v deviates from $f(x_j)$ with L2 loss function.

We choose Multi-Layer Perceptron(MLP) with dropout as a transformation function f

Method

Regularization

kNN-based model is ineffective in a high-dimensional space if the data points spread too widely(the curse of dimensionality)

→ Bound all the user, item with in a unit sphere.

i.e. $\|u_*\|^2 \leq 1$, $\|v_*\|^2 \leq 1$

Not regularizing L^2-norm, because origin has no specific meaning.

Method

Regularization

Covariance Regularization

To reduce the correlation between activations in a deep neural network

$$C_{ij} = \frac{1}{N}\sum_{n}({y_i}^n - \mu_i)({y_j}^n - \mu_j)$$

y^n : object's latent vector, n : batch index, N : batch size, $\boldsymbol{\mu_i} = \frac{1}{N}\sum_n {y_i}^n$

$$\mathcal{L}_c = \frac{1}{N}(\|C\|_f - \|diag(C)\|_2^2)$$

$\|\cdot\|$: Frobenius norm

Method

Training Procedure

Objective

$$\min_{\theta,\mathbf{u}_*,\mathbf{v}_*} \mathcal{L}_m + \lambda_f \mathcal{L}_f + \lambda_c \mathcal{L}_c \quad \text{s.t.} \quad \|\mathbf{u}_*\|^2 \leq 1 \text{ and } \|\mathbf{v}_*\|^2 \leq 1,$$

- λ_f, λ_c are **hyperparameters that control the weight of each loss term**
- **Minimize objective function with Mini-Batch SGD**
- **Control the learning rate using AdaGrad**

Method

Training Procedure

- Sample N positive pairs from S
- For each pair, sample U negative items and approximate $rank_d(i, j)$
- For each pair, keep the negative item k that maximizes the hinge loss and form a mini-batch of size N.
- Compute gradients and update parameters with a learning rate controlled by AdaGrad.
- Censor the norm of u_* and v_* by $y' = \frac{y}{max(\|y\|,1)}$
- Repeat this procedure until convergence.

Experiments

Datasets

Table 1: Dataset Statistics.

	CiteULike	BookCX	Flickr	Medium	MovieLens20M	EchoNest
Domain	Paper	Book	Photography	News	Movie	Song
# Users	7,947	22,816	43,758	61,909	129,797	766,882
# Items	25,975	43,765	100,000	80,234	20,709	260,417
# Ratings	142,794	623,405	1,372,621	2,047,908	9,939,873	7,261,443
Concentration[a]	33.47%	33.10%	13.48%	55.38%	72.52%	65.88%
Features Type	Tags	Subjects	Image Features	Tags	Genres, Keywords	NA
# Feature Dim.	10,399	7,923	2,048	2,313	10,399	NA

6 different domains with varying sizes and difficulties

Experiments

Evaluation Methodology

- Training 60%, validation 20%, test 20%
- Users with < 5 ratings are only in the training set.
- Ranking is evaluated on recall rates for Top-K recommendations.

- CML
  - WRMF, BPR, WARP

- CML+F(CML with item features)
  - FM, VBPR, CDL

Experiments

Recommendation Accuracy

Table 2: Recall@50 and Recall@100 on the test set. (# dimensions $r = 100$) The best performing method is boldfaced. $*$, $**$, $***$ indicate $p \leq 0.05$, $p \leq 0.01$, and $p \leq 0.001$ based on the Wilcoxon signed rank test suggested in [41].

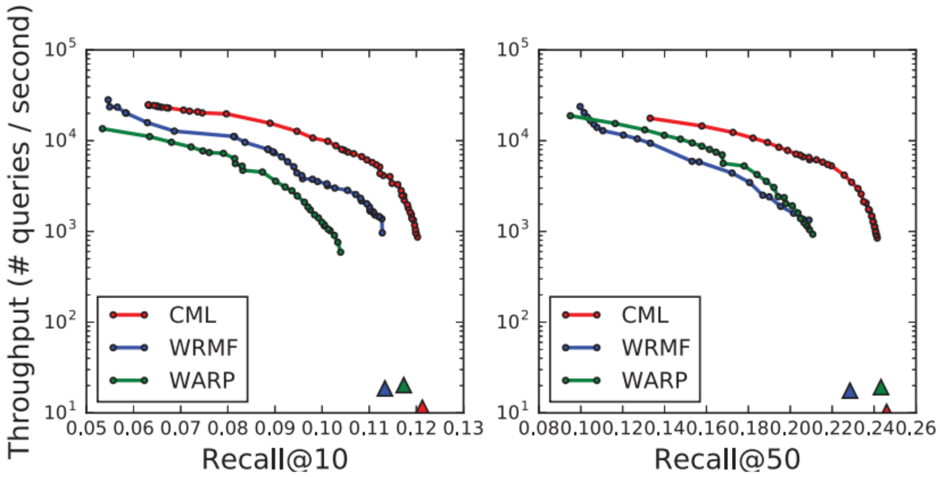
	WRMF	BPR	WARP	CML	*ours vs. best*	FM	VBPR	CDL	CML+F	*ours vs. best*
Recall@50										
CiteULike	0.2437	0.2489	0.1916	**0.2714*****	9.03%	0.1668	0.2807	**0.3375****	0.3312	-1.86%
BookCX	0.0910	0.0812	0.0801	**0.1037*****	13.95%	0.1016	0.1004	0.0984	**0.1147*****	12.89%
Flickr	0.0667	0.0496	0.0576	**0.0711*****	6.59%	NA	0.0612	0.0679	**0.0753*****	10.89%
Medium	0.1457	0.1407	0.1619	**0.1730*****	6.41%	0.1298	0.1656	0.1682	**0.1780*****	5.82%
MovieLens	0.4317	0.3236	0.4649	**0.4665**	0.34%	0.4384	0.4521	0.4573	**0.4617***	0.96%
EchoNest	0.2285	0.1246	0.2433	**0.2460**	1.10%	NA	NA	NA	NA	NA
Recall@100										
CiteULike	0.3112	0.3296	0.2526	**0.3411*****	3.37%	0.2166	0.3437	0.4173	**0.4255****	1.96%
BookCX	0.1286	0.1230	0.1227	**0.1436*****	11.66%	0.1440	0.1455	0.1428	**0.1712*****	17.66%
Flickr	0.0821	0.0790	0.0797	**0.0922*****	12.30%	NA	0.0880	0.0909	**0.1048*****	15.29%
Medium	0.2112	0.2078	0.2336	**0.2480*****	6.16%	0.1900	0.2349	0.2408	**0.2531*****	5.10%
MovieLens	0.5649	0.4455	0.5989	**0.6022**	0.55%	0.5561	0.5712	0.5943	**0.5976**	0.55%
EchoNest	0.2891	0.1655	0.3021	**0.3022**	0.00%	NA	NA	NA	NA	NA

Experiments

Top-K Recommendations with LSH

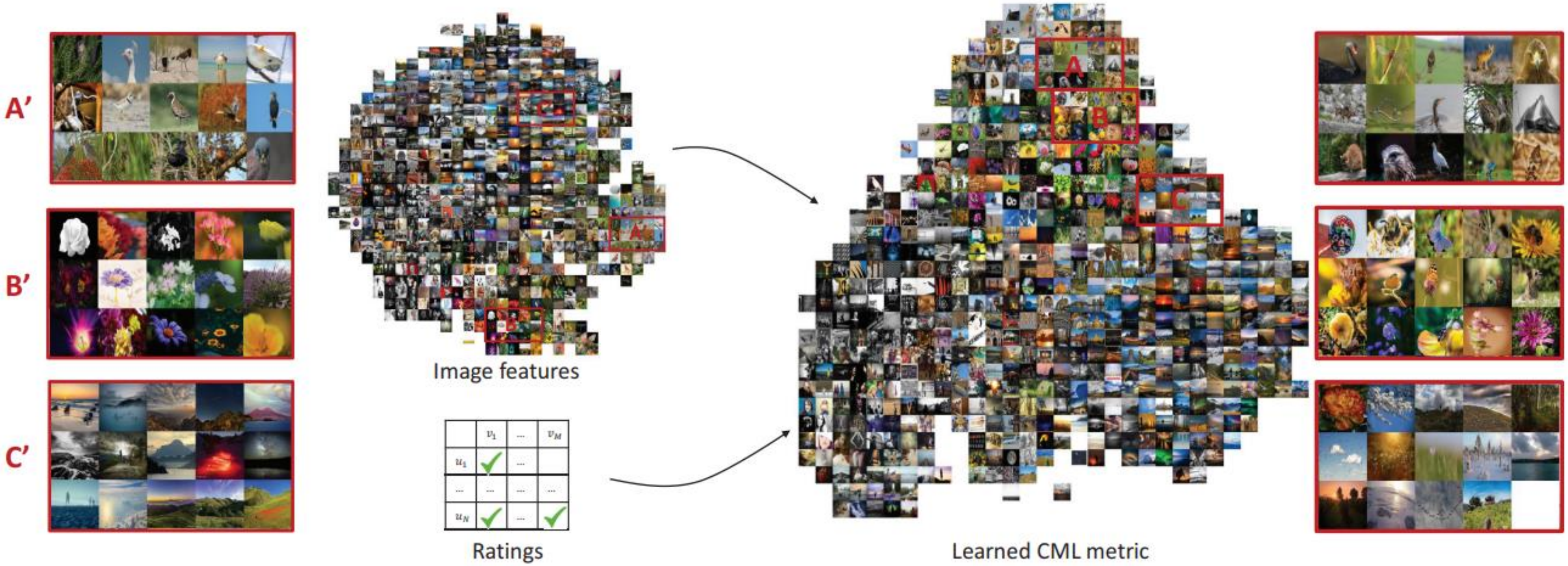
CML Brute-force search : slowest

CML + LSH : fastest



Experiments

Metric Visualization



Conclusion

- Improvement of accuracy in recommendation domain

- CML uncovers users' fine-grained preferences and the underlying preference spectrum.

- CML captures such relationships in a more intuitive way an can better propagate such information through user-item pairs.

Implementation

Data : MovieLens 100K

update하는 부분에서 구현 실패

```
def train_all(self):
    while not self.converged:
        self.CML_train()

def CML_train(self):
    self.sample_positive()      # Sample N positive pairs from S
    self.sample_negative()      # Sample Negative and approximate
    self.mini_batch()           # form a mini-batch
    self.update()               # gradient computing, update parameters
    self.censor()               # Censoring
    self.check_converged()      # check convergence
```

Implementation

- 논문 내 수식 이해, 증명하려고 노력
- 구현 중 실패
  - Update 부분을 제대로 구현하지 못함 -> 해당 부분에서 계속 이상한 값이 나옴.