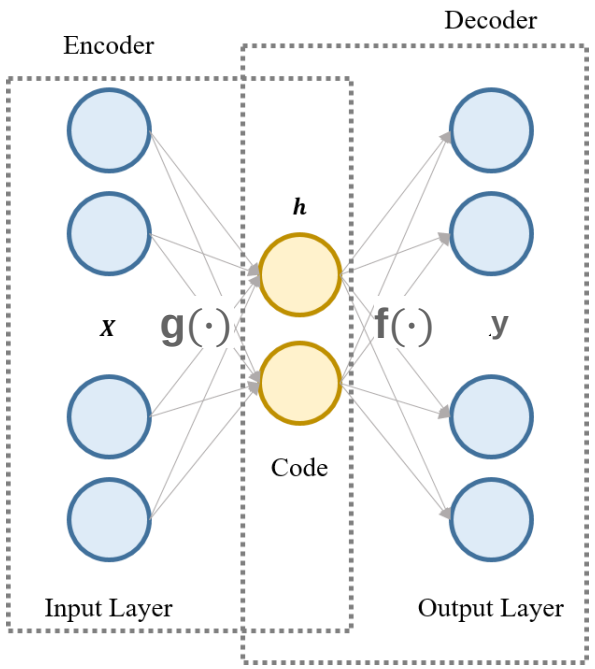
AutoRec : AutoEncoders Meet Collaborative Filtering

2020.11.21

윤이화

Autoencoder란

[AutoEncoder 모델 구조]



$h = g_\theta(x) = s(Wx + b)$

$y = f_{\theta'}(h) = s(W'y + b')$

$$L_{AE} = \sum_{x \in D} L(x, y)$$

모델 개요 및 특징

- Input과 Output이 동일한 **Self-supervised 학습**으로, Input을 저차원의 latent space에 압축하고 원상태로 복원하는 방식
- Input을 Output으로 변환하기 위한 중간 표현 상태를 잘 학습하는 것이 목표
- Aautoencoder는 **data-specific하기 때문에** 훈련된 데이터와 비슷한 데이터로만 압축될 수 있음

학습 방식

- Input Data를 Encoder에 통과시켜 h로 압축
- 압축된 h vector로부터 Input과 같은 크기의 출력 생성
- Loss값은 입력값 x와 Decoder를 통과한 y값의 차이로 정의
- Loss를 최소화하도록 경사하강법으로 최적화

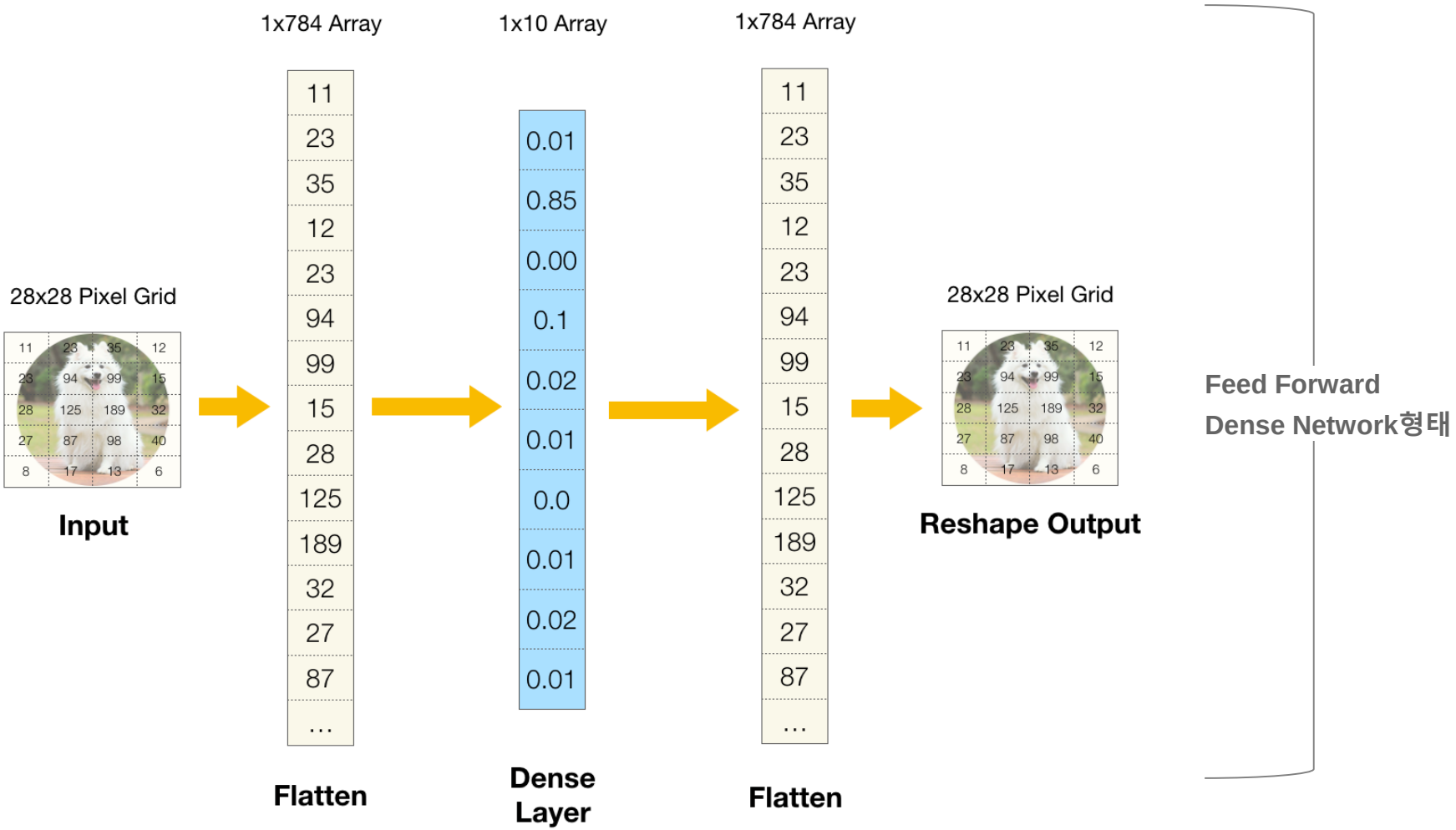
주요 활용 용도

Feature 추출

Dimension Reduction

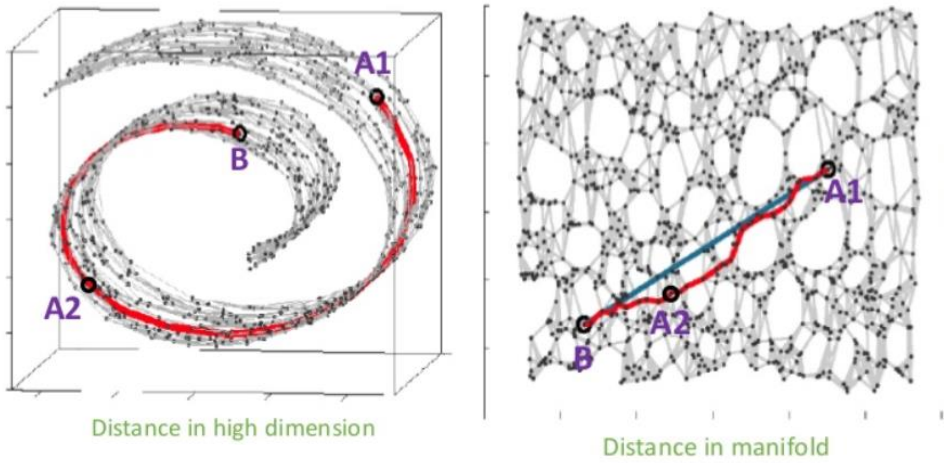
Network parameter 초기화

Autoencoder란



Autoencoder를 왜 사용할까?

Dimension Reduction



문제점

✓ 고차원에서 가까운 두 샘플은 의미적으로는 다를 수 있다

✓ 차원의 저주로 인해서 고차원에서 유의미한 거리 측정 방식을 찾기 어렵다

Manifold Assumption

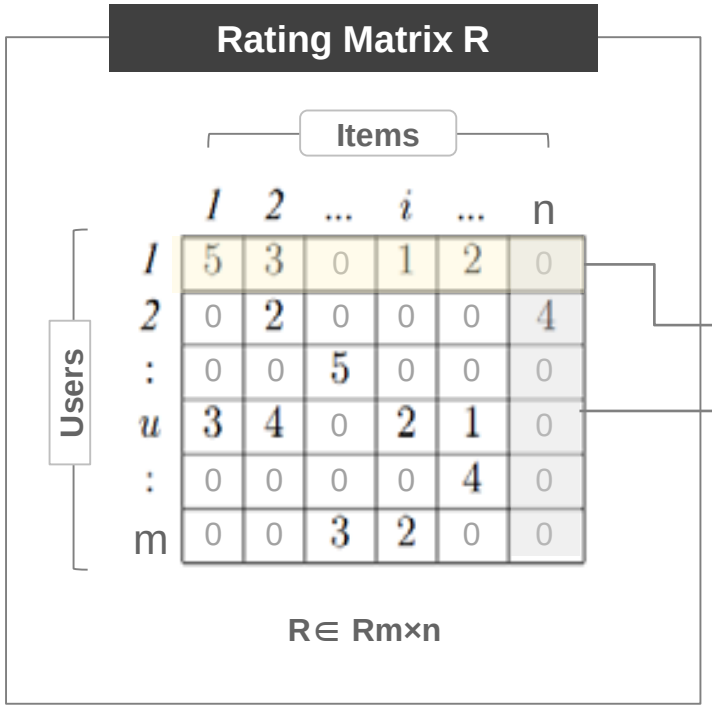
✓ 고차원의 데이터 밀도는 낮지만 이들의 집합을 포함하는 저차원의 Sub Space, 즉 Manifold가 있다
✓ manifold를 통해 우리는 샘플 데이터의 특징을 파악할 수 있다

Encoder 목적	고차원 데이터를 저차원의 데이터로 만들어서 원래의 데이터를 잘 설명하는 Manifold를 찾는 것
AE 차별점	MF, PCA등과 다르게 **Non-linear manifold**를 찾을 수 있다

고차원의 데이터를 잘 표현하는 manifold
≈ Encoder를 통해 나온 Vector

Encoder를 통해 학습한 Vector를 통한 유사도 산출
≈ Manifold상에서 유사한, Dominant Feature가 유사한 샘플을 찾겠다

AutoRec : 개요



1

✓ 모든 **unobserved data는 0으로 가정**한다.

→ Why? 학습에는 Observed data만 기여할 수 있도록 하였기 때문에

2

✓ 모든 아이템 i ∈ I = {1 . . . n}와 유저 u ∈ U = {1 . . . m]은 **partially observed vetor로 표현할 수 있다**.

$\mathbf{r}^{(u)} = (R_{u1}, \ldots R_{un}) \in \mathbb{R}^n$ → ex) $r^{user(1)} = (5, 3, 0, 1, 2, 0)$

$\mathbf{r}^{(i)} = (R_{1i}, \ldots R_{mi}) \in \mathbb{R}^m$ → ex) $r^{item(n)} = (0, 4, 0, 0, 0, 0, 0)$

3

✓ $r^{(i)}$또는 $r^{(u)}$를 저차원의 **Latent space에 투영하고 (encode), 다시 reconstruct (decode)**하여 평가되지 않은 항목을 예측하고 추천에 활용이 가능하다.

Objective : $\min_{\theta} \sum_{\mathbf{r} \in \mathbf{S}} ||\mathbf{r} - h(\mathbf{r}; \theta)||_2^2, \quad h(\mathbf{r}; \theta) = f\left(\mathbf{W} \cdot g(\mathbf{V}\mathbf{r} + \boldsymbol{\mu}) + \mathbf{b}\right)$

Matrix Completion을 Autoencoder 방식으로 구현

AutoRec : Model Formula

Item-based AutoRec 구조

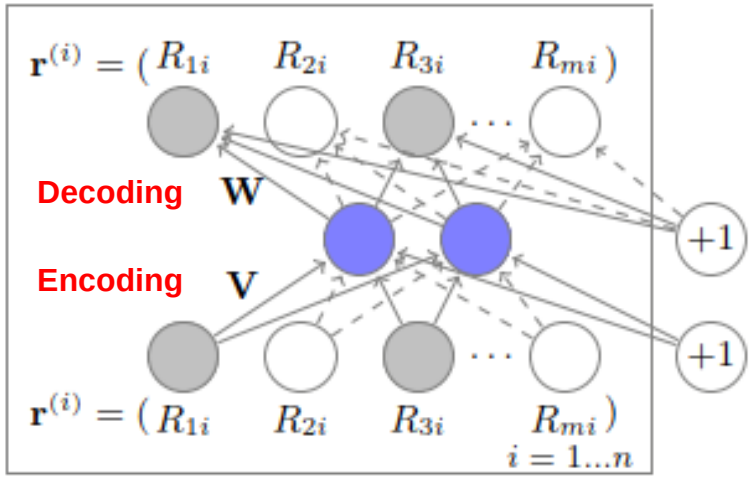


Figure 1: Item-based AutoRec model. We use plate notation to indicate that there are n copies of the neural network (one for each item), where $\mathbf{W}$ and $\mathbf{V}$ are tied across all copies.

K-dimension인 Hidden Layer 하나만 가지는 Autoencoder 구조

$$h(\mathbf{r};\theta) = f\left(\mathbf{W} \cdot g(\mathbf{V}\mathbf{r} + \boldsymbol{\mu}) + \mathbf{b}\right)$$

Notation

- **θ** : {W, V, μ, b}
- **Weight** : W ∈ Rd×k, V ∈ Rk×d,
- **Bias** : μ ∈Rk, b ∈ Rd
- **f(·), g(·)**: activation function

Objective

- 각 r(i)는 **observed input과 관련된 weight만 업데이트**함으로써 partially observed[1] 된다

→ Observed ratings만 영향을 준다

- 과적합 방지를 위해 **L2 정규화 적용**
- Input인 실제 Rating과 예측 Output 차이를 minimize

$$\min_{\theta} \sum_{i=1}^{n} \underbrace{||\mathbf{r}^{(i)} - h(\mathbf{r}^{(i)};\theta))||_{\mathcal{O}}^2}_{\text{Positive Rating만 사용}} + \frac{\lambda}{2} \cdot (||\mathbf{W}||_F^2 + ||\mathbf{V}||_F^2)$$

훈련 파라미터 및 결과

- **훈련 파라미터** V: mk + k, W : mk + m

→ 총 2mk + m+ k개 파라미터 훈련

- Hyperparameter : k, 람다, 배치사이즈 등

- User u의 아이템i에 대한 평가

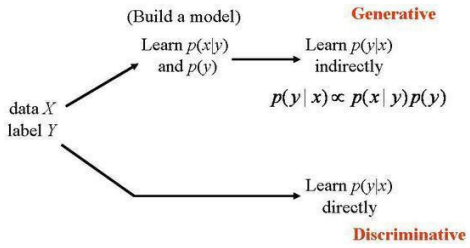
→ $\hat{R}_{ui} = (h(\mathbf{r}^{(i)};\hat{\theta}))_u.$

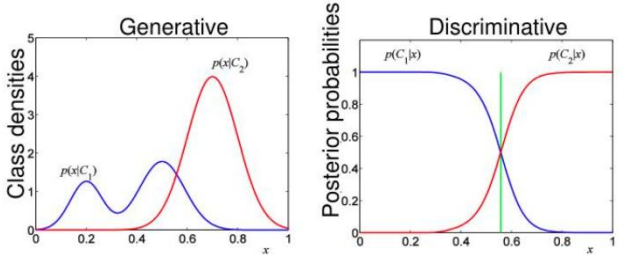
1) Fully Observable하면 출력 결과값만을 가지고 내부를 모두 추론할 수 있고, Partially observable하면 일부만 추론할 수 있음

AutoRec : 다른 모델과 차이점

RBM-CF와의 차이점

- **RBM-CF는 generative한 Probabilistic 모델이고, AutoRec은 Discriminative 모델이다**





Discriminative : P(Y|X)를 직접 도출, 데이터를 구분하는 Decision boundary를 학습하는 것이 목표

Generative : P(Y|X) ∝ P(X|Y)P(X)로 간접도출, 확률 분포를 학습하는 것이 목표

- RBM-CF는 Log likelihood를 maximize하고, AutoRec은 RMSE를 minimize하여 파라미터를 추정한다
- RBM-CF는 Contrastive divergence를 통해 학습하는 반면, **AutoRec은 비교적 학습이 빠른 gradient based backprop**으로 학습한다
- RBM-CF는 Discrete rating만 가능하며, 각 rating마다 파라미터를 추정해야하기 때문에 총 nkr개의 추정이 필요한데, **AutoRec은 r의 수에 영향 받지않아 더 적은 파라미터로 학습 가능하다.**

MF 기반 모델과 차이점

- MF 기반은 Item과 User를 같은 공간에 임베딩하는 반면, Item based AutoRec은 아이템만 임베딩한다.
- **AutoRec은 비선형 latent representation**이 가능하다

AutoRec : Experimental Evaluation

비교 모델

RBM-CF | BiasedMF | LLORMA

Dataset

Movielens 1M | Movielens 10M | Netflix

Setting

- Default rate of 3 for test users or items without training observations
- Split data 9:1 train test and 10% holdout for hyperparameter tuning
- For all baselines, we tuned the regularisation strength
- $\lambda \in \{0.001, 0.01, 0.1, 1, 100, 1000\}$ and the appropriate latent
- dimension $k \in \{10, 20, 40, 80, 100, 200, 300, 400, 500\}$
- Used for Rprop AutoRec propagation

Questions

1. User, Item based RBM과 AutoRec 성능 차이
2. Linear / Non-linear activation function 사용에 따른 AutoRec 성능 차이
3. Hidden Unit 개수에 따른 AutoRec 성능 차이
4. Baseline과 AutoRec 성능 차이
5. AutoRec 층을 깊게 쌓았을 때 성능 향상 여부

AutoRec : Experimental Evaluation

	ML-1M	ML-10M
U-RBM	0.881	0.823
I-RBM	0.854	0.825
U-AutoRec	0.874	0.867
I-AutoRec	**0.831**	**0.782**

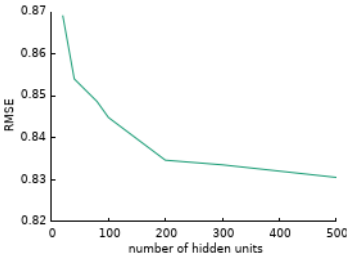
(a)

$f(\cdot)$	$g(\cdot)$	RMSE
Identity	Identity	0.872
Sigmoid	Identity	0.852
Identity	Sigmoid	**0.831**
Sigmoid	Sigmoid	0.836

(b)

	ML-1M	ML-10M	Netflix
BiasedMF	0.845	0.803	0.844
I-RBM	0.854	0.825	-
U-RBM	0.881	0.823	0.845
LLORMA	0.833	**0.782**	0.834
I-AutoRec	**0.831**	**0.782**	**0.823**

(c)



1. User, Item based RBM과 AutoRec 성능 차이 -- (a)
- AutoRec이 더 우수
- Item-based가 전반적으로 나은데, rating per item이 rating per user보다 더 많기 때문

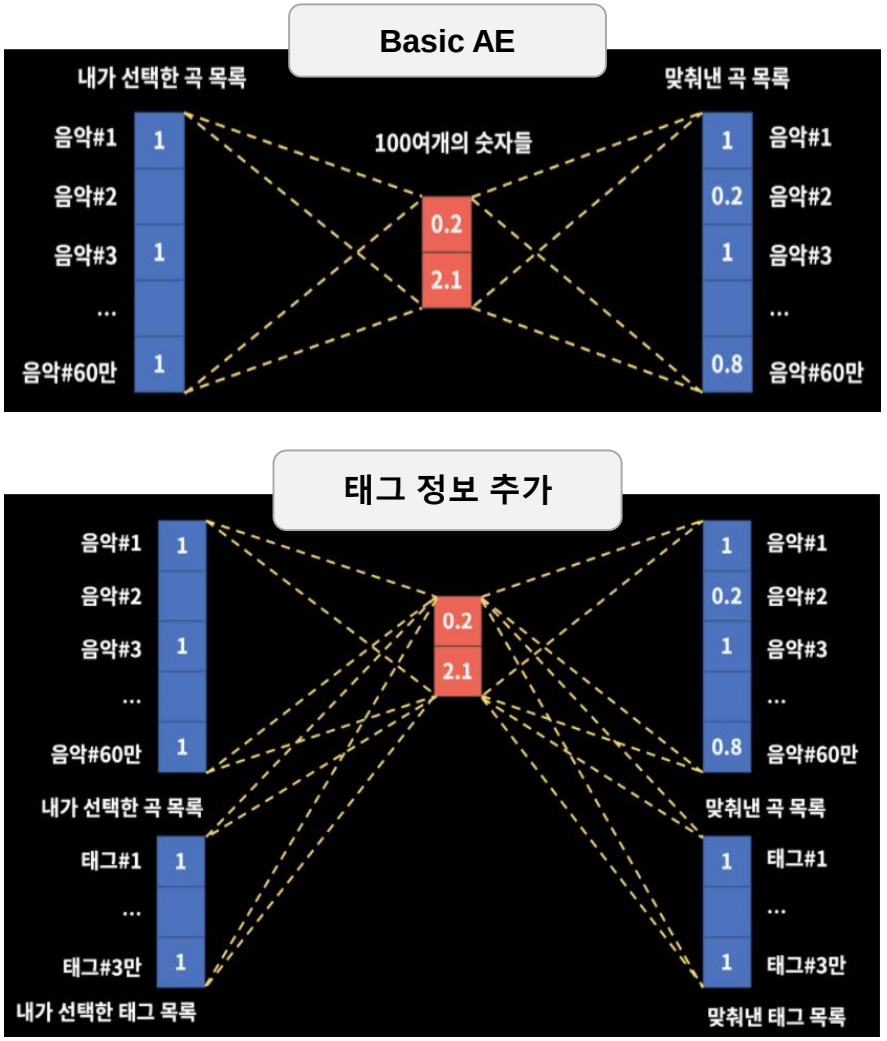
2. Linear / Non-linear activation function 사용에 따른 AutoRec 성능 차이 – (b)
- Hidden layer의 activation function의 비선형성이 성능에 영향을 많이 끼침
- Potential advantage over MF based models

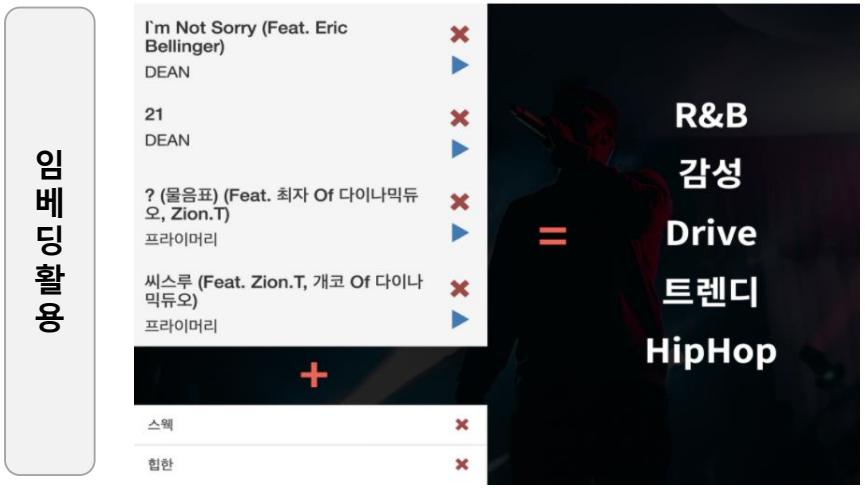
3. Hidden Unit 개수에 따른 AutoRec 성능 차이 -- graph
- 단조 증가, with diminishing return

4. Baseline과 AutoRec 성능 차이 – (c)
- AutoRec이 우수, LLORMA의 경우 50개의 근접 matrix를 사용한데 반해, AutoRec은 single latent representation 사용하였는데 동일한 결과

5. AutoRec 층을 깊게 쌓았을 때 성능 향상 여부
- (500, 250, 500) unit으로 3층 쌓았을 때 성능이 더 향상됐음

AutoRec : 활용 사례_멜론





https://tech.kakao.com/2020/04/29/kakaoarena-3rd-part1/