Neural Collaborative Filtering vs. Matrix Factorization Revisited

Steffen Rendle, Walid Krichene,
Li Zhang, John Anderson

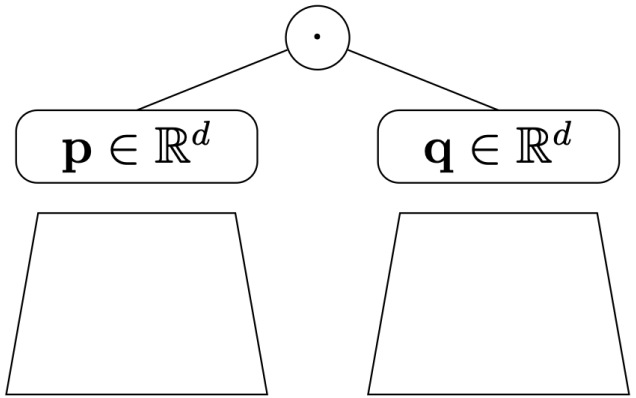
Dot product similarity

p – user embedding
q – item embedding

$$\phi^{\text{dot}}(\mathbf{p}, \mathbf{q}) := \langle \mathbf{p}, \mathbf{q} \rangle = \mathbf{p}^T \mathbf{q} = \sum_{f=1}^{d} p_f q_f.$$

$$\phi^{\text{dot}}(\mathbf{p}, \mathbf{q}) := b + p_1 + q_1 + \langle \mathbf{p}_{[2,\ldots,d]}, \mathbf{q}_{[2,\ldots,d]} \rangle$$

$$\phi^{\text{dot}}(\mathbf{p}, \mathbf{q}) = \langle \mathbf{p}, \mathbf{q} \rangle$$

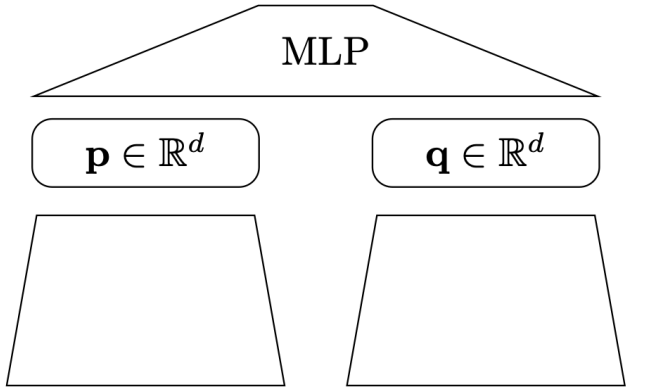


MLP similarity

p – user embedding
q – item embedding

$$\mathbf{f}: \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}: \quad \mathbf{f}_{W,\mathbf{b}}(\mathbf{x}) = \boldsymbol{\sigma}(W\,\mathbf{x} + \mathbf{b}), \quad \boldsymbol{\sigma}(\mathbf{z}) = [\sigma(z_1), \ldots, \sigma(z_{\text{out}})],$$
$$W \in \mathbb{R}^{\text{in} \times \text{out}}, \ \mathbf{b} \in \mathbb{R}^{\text{out}}$$

$$\phi^{\text{MLP}}(\mathbf{p}, \mathbf{q}) = \mathbf{f}_{W_l,\mathbf{b}_l}(\ldots \mathbf{f}_{W_1,\mathbf{b}_1}([\mathbf{p}, \mathbf{q}]) \ldots)$$



NeuMF

$$\phi^{\text{NeuMF}}(\mathbf{p}, \mathbf{q}) := \phi^{\text{MLP}}(\mathbf{p}_{[1,\dots j]}, \mathbf{q}_{[1\dots j]}) + \phi^{\text{GMF}}(\mathbf{p}_{[j+1\dots d]}, \mathbf{q}_{[j+1\dots d]}),$$

$$\phi^{\text{MLP}}(\mathbf{p}, \mathbf{q}) := \mathbf{f}_{W_l, \mathbf{b}_l}(\dots \mathbf{f}_{W_1, \mathbf{b}_1}([\mathbf{p}, \mathbf{q}]) \dots).$$

$$\phi^{\text{GMF}}(\mathbf{p}, \mathbf{q}) := \sigma(\mathbf{w}^T(\mathbf{p} \odot \mathbf{q})) = \sigma(\langle \mathbf{w} \odot \mathbf{p}, \mathbf{q} \rangle) = \sigma\left(\sum_{f=1}^{d} w_f p_f q_f\right)$$

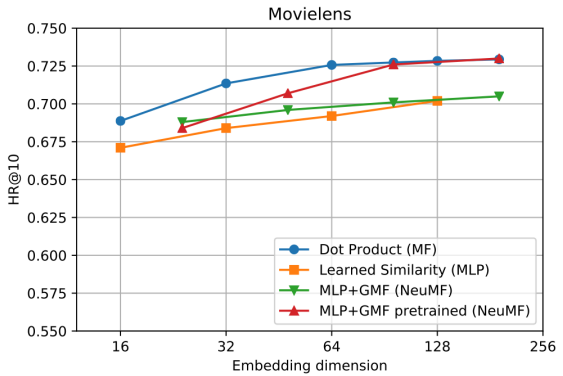
Matrix factorization

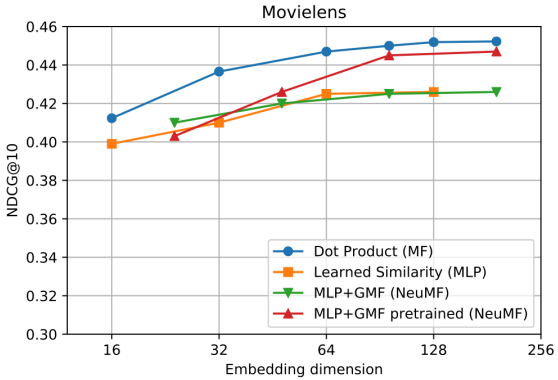
$$l(u,i,y) = -y \ln \sigma(\phi(\mathbf{p}_u, \mathbf{q}_i)) - (1-y) \ln(1 - \sigma(\phi(\mathbf{p}_u, \mathbf{q}_i))) + \lambda \|\mathbf{p}_u\| + \lambda \|\mathbf{q}_i\|$$

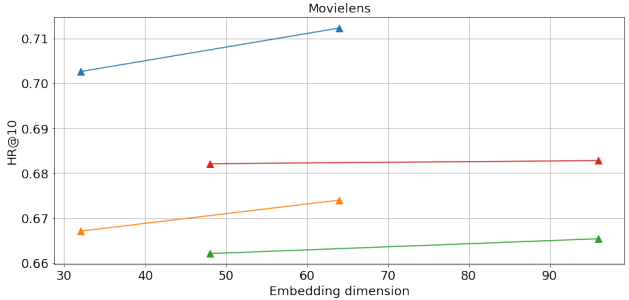
$$\begin{aligned}
p_{u,1} &\leftarrow p_{u,1} - \eta[(\sigma(\phi(\mathbf{p}_u, \mathbf{q}_i)) - y) + \lambda p_{u,1}] \\
q_{i,1} &\leftarrow q_{i,1} - \eta[(\sigma(\phi(\mathbf{p}_u, \mathbf{q}_i)) - y) + \lambda q_{i,1}] \\
\mathbf{p}_{u,[2,\ldots,d]} &\leftarrow \mathbf{p}_{u,[2,\ldots,d]} - \eta[(\sigma(\phi(\mathbf{p}_u, \mathbf{q}_i)) - y)\, \mathbf{q}_{i,[2,\ldots,d]} + \lambda \mathbf{p}_{u,[2,\ldots,d]}] \\
\mathbf{q}_{i,[2,\ldots,d]} &\leftarrow \mathbf{q}_{i,[2,\ldots,d]} - \eta[(\sigma(\phi(\mathbf{p}_u, \mathbf{q}_i)) - y)\, \mathbf{p}_{u,[2,\ldots,d]} + \lambda \mathbf{q}_{i,[2,\ldots,d]}]
\end{aligned}$$

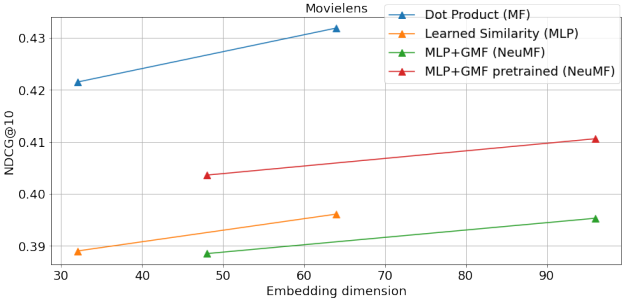
MLP-learned similarity vs dot product similarity

Hypotheses: Simple dot product yields better results



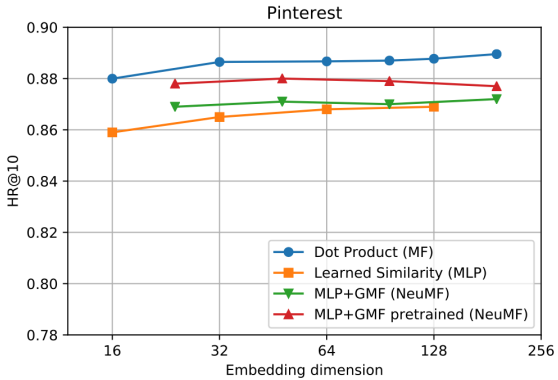


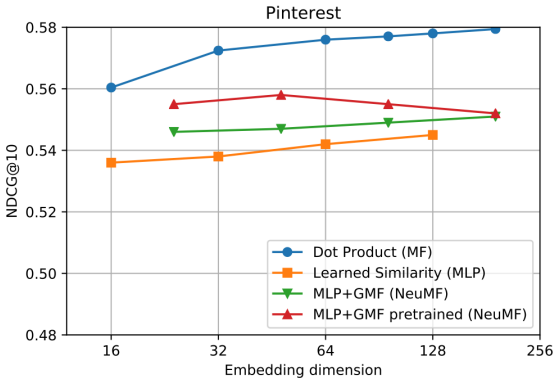


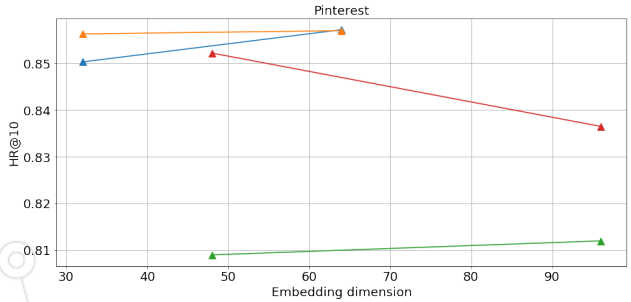


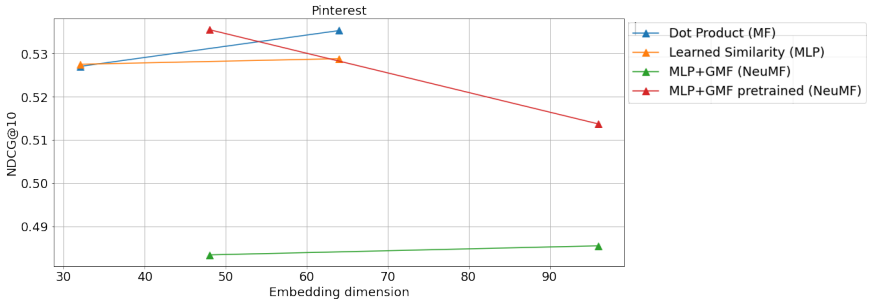
MLP-learned similarity vs dot product similarity

Hypotheses: Simple dot product yields better results









Summarization of experiments

Authors:

- Dot product is better that MLP or NeuMF
- Ensemble of GMF and MLP shows no more than ensemble is preformes better than a single model
- MLP has more parameters

SAS team:

- Speed up training!
- The hyperparameters for MF were better explored

	Negatives	Regularization
MF	8, 10	Yes
MLP, GMF, NeuMF	4	No

- We couldn't show the same superiority of MF as authors

Learning a Dot Product with MLP is Hard:

Experiment:
Input: $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$ from $\mathcal{N}(0, \sigma^2_{\text{emb}} I)$
Goal: $\hat{y} : \mathbb{R}^{2\,d} \to \mathbb{R}$ approximate y with $\hat{y}(\mathbf{p}, \mathbf{q})$
Labels: $y(\mathbf{p}, \mathbf{q}) = \langle \mathbf{p}, \mathbf{q} \rangle + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2_{\text{label}})$

3 datasets of tuples $(\mathbf{p}, \mathbf{q}, y)$

1) Training
2) Test

} Sample with P ~ 100 positives for each user

3) Test of fresh embeddings <- generalization!

Learning a Dot Product with MLP is Hard:

$$\sqrt{\mathrm{Var}(y)} = \sqrt{\sigma_{\text{label}}^2 + d\,\sigma_{\text{emb}}^4} \longrightarrow$$ Trivial RMSE on Netflix Prize 1.13

Trivial model predicts average rating ~ 0

Noise:

$\sigma_{label} = 0.85$

Netflix Prize

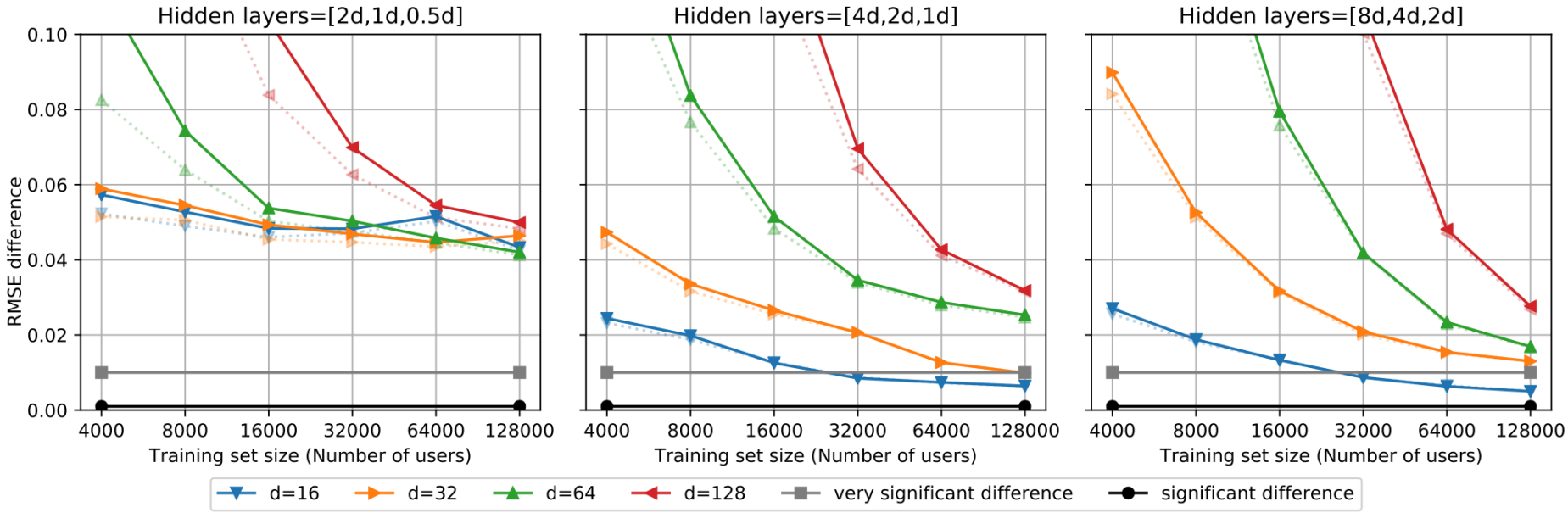
Trivial RMSE Best RMSE

$$\sigma_{emb}^2 = \sqrt{\frac{1.13^2 - 0.85^2}{d}}$$

RMSE Metric : Between MLP predictions and dot product

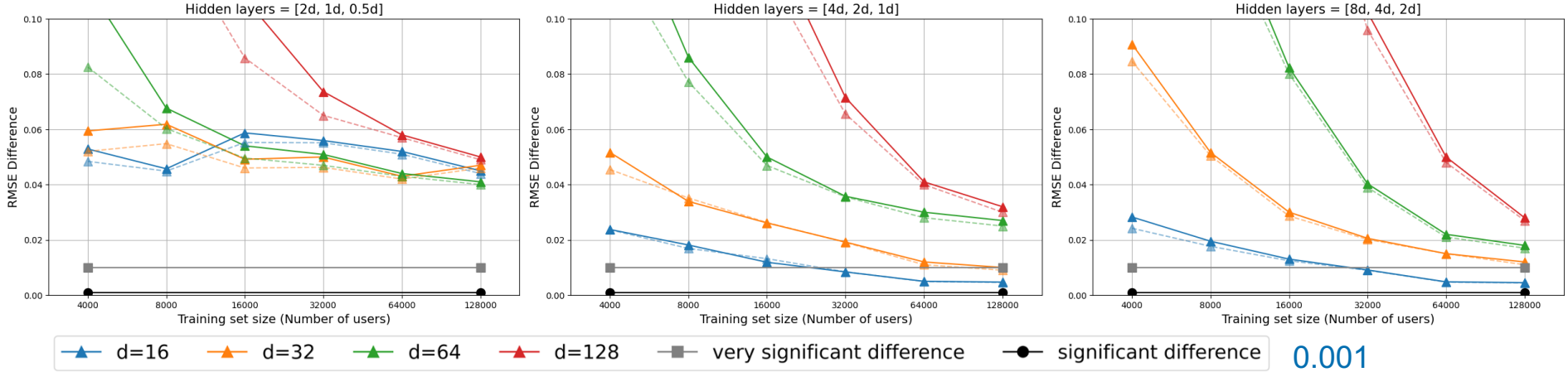
Choose σ_{label} and σ_{emb} to get acceptable values

Experiment results



Experiment results

Input layer size: 2d; MLP layers size: [4h, 2h, h, h]; h = d/2, d, 2d,



0.001

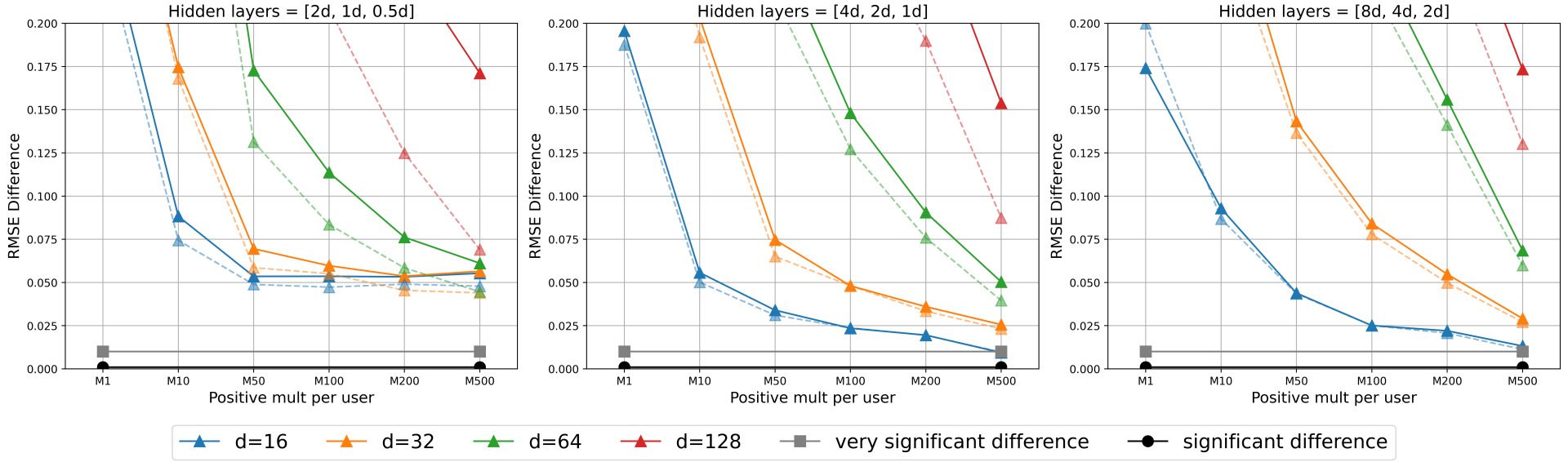
0.01

↓ RMSE on Netflix Prize - 1 year

MLP can approximate a dot product!!!

Increasing number of positives

Fixed Training size = 4000



But!

- It becomes increasingly difficult for an MLP to fit the dot product function with increasing dimensions

$$O(d/\epsilon)^{\alpha} \text{ for } 1 \leq \alpha \leq 2$$

- d = 128, with 128000 users, the error is still above 0.02, much higher than the very significant difference of 0.01!

Applicability of Dot Product Models ⌛ Time!

Dot product similarity:

- for 1 item: O(d)
- for n items: O(nd)

MLP-learned similarity:

- for 1 item: $O(d^2)$
- for n items: $O(nd^2)$

$d \sim 1e^2$, $n \sim 1e^6$

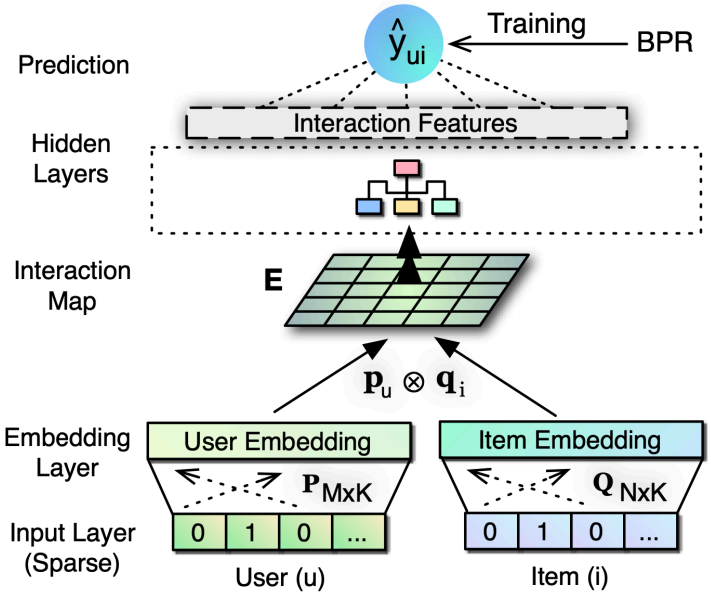
Both are impractical for retrieval applications

- Approximate nearest neighbor search or maximum inner product search
- is not applicable for real time!

Other approaches from Related Works

MLPs at the Output Layer of DNNs

- NeuMF = MLP + GMF
- MLP ↔ outerproduct + CNN

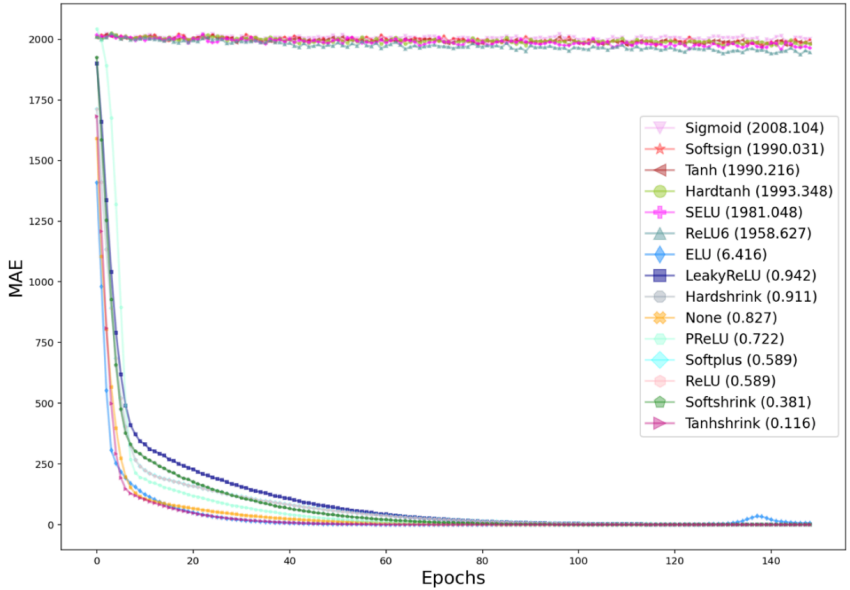


Conclusion

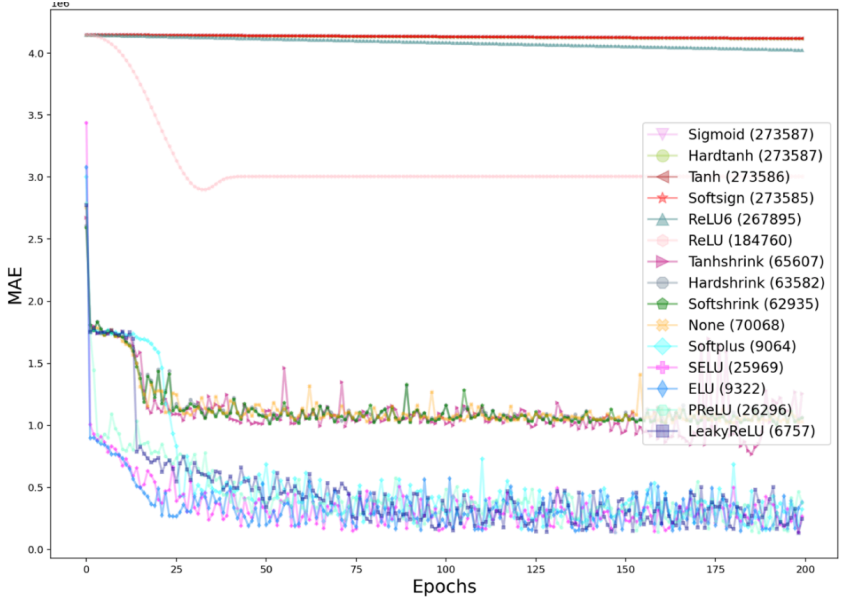
- Dot product is better that MLP or NeuMF:
  - More applicable (time)
  - No pretraining, No need for large datasets
  - Better alignment with other research areas such as NLP or CV

MLP for arithmetic operations

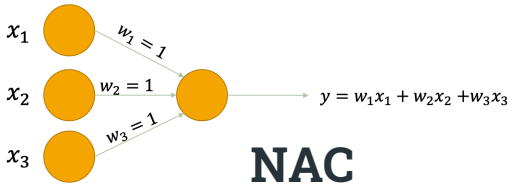
Addition



Multiplication



Special layers



$y = w_1x_1 + w_2x_2 + w_3x_3$

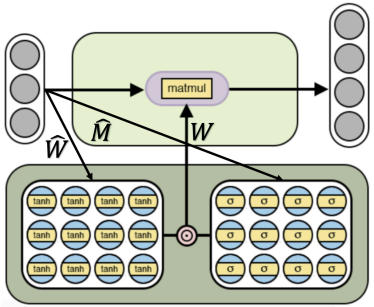
Addition

$y = x_1 + x_2 + x_3$

NAC

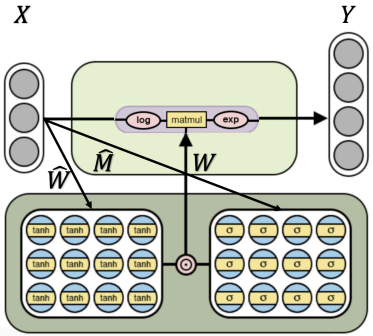
Simple:

$W = tanh(\hat{W}) \odot \sigma(\hat{M})$

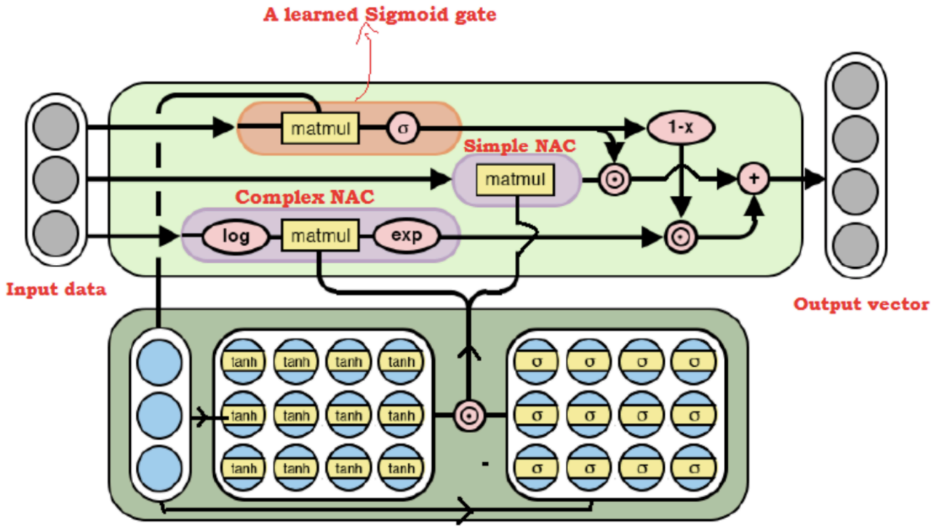


Complex:

$Y = exp(W \cdot (log(|x| + \varepsilon))$



NALU



	NAC	NALU
MAE	0.000009	0.15