# Modern Trends in Recommender Systems

Debapriyo Majumdar

debapriyo@isical.ac.in

June 2021

# Modern recommendation approaches

- Amount of data is huge in scale
- Computing resources have become more powerful
- Some representative new approaches
  - Extreme multiclass classification for recommendation
  - Neural collaborative filtering
  - Wide and deep learning
  - Autoencoders

# Deep Neural Networks for YouTube Recommendation

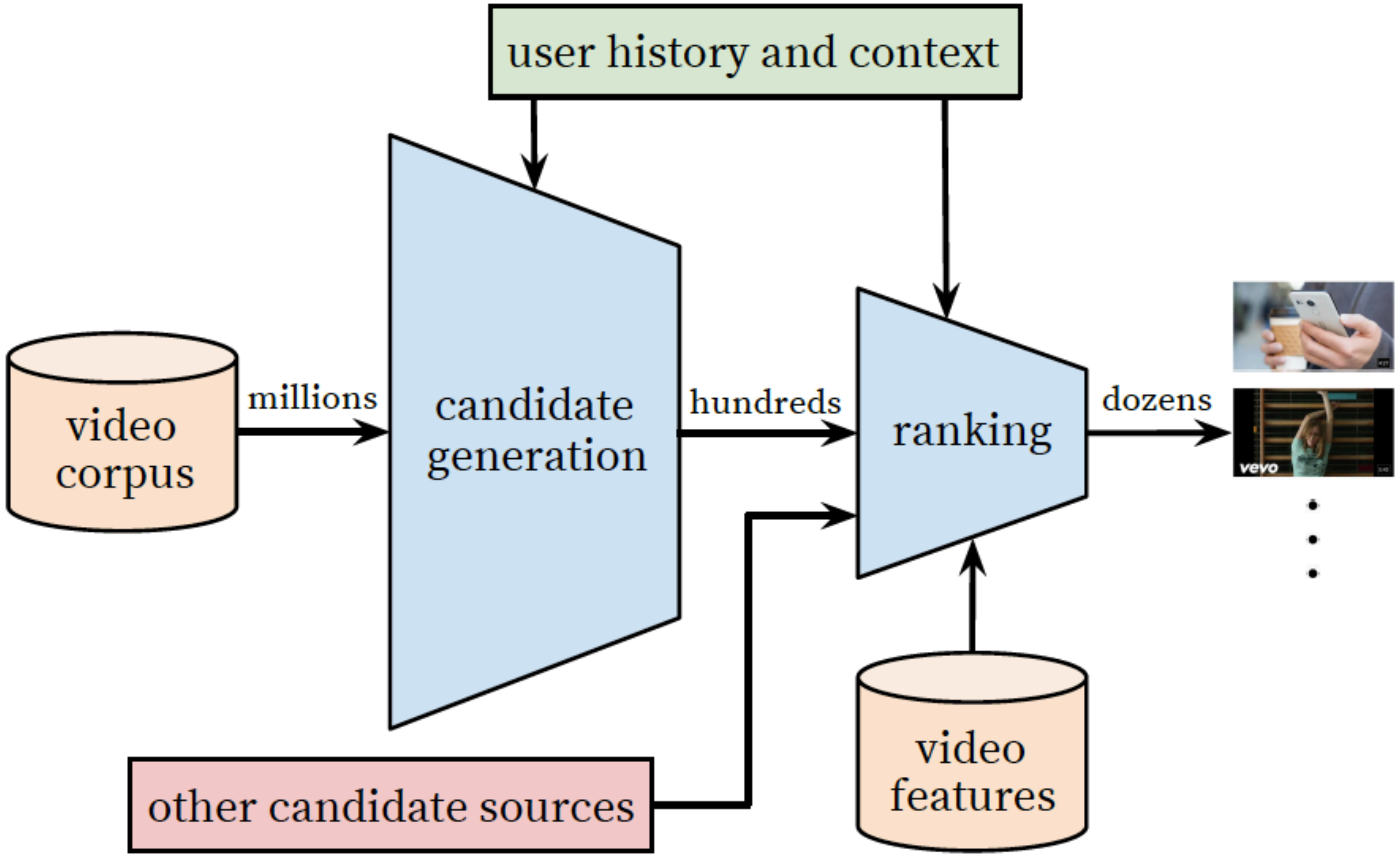
Paul Covington, Jay Adams, Emre Sargin
Google

In *Proceedings of the 10th ACM conference on recommender systems*, pp. 191-198. 2016.

# Recommending videos on YouTube: Challenges

- Scale: *many* existing recommendation algorithms work on small scale, but not on YouTube
- Freshness: Many hours of video uploaded per second
  - Need to balance recommendation of old + new content
- Noise
  - Explicit user satisfaction rarely available
  - Metadata poorly structured (note: making it too structured would make it difficult for the users)

# System overview



# Recommendation as classification

Extreme multiclass classification

When #classes and #dimensions are very high

$u =$ high dimensional embedding of user-context pair

$$P(w_t = i \mid U, C) = \frac{e^{v_i u}}{\sum_{j \in V} e^{v_j u}}$$

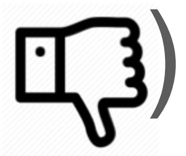
Probability of watching a particular video $i$ at time $t$, given user $U$ and context $C$

$v_j =$ high dimensional embedding of video $j$

Task:

- Learn *u* as a function of user's history and context
- Classify using the *softmax* classifier

# Training data

- How to get positive or negative examples?
  - Explicit (or ) feedback is very sparse
  - Implicit: Users are shown recommended vidoes
    A. Clicked vidoes are *positive.*
    Annotated with watch time
    B. Unclicked vidoes are negative
    C. Search history combined with watch history

**Number of negative classes (hence samples) are $\gg$ number of positive samples**

# Efficient training

- Uses a technique proposed by Jean, Cho, Memisevic, Benjio (ACL 2015) and also by Benjio and Senecal (2008)

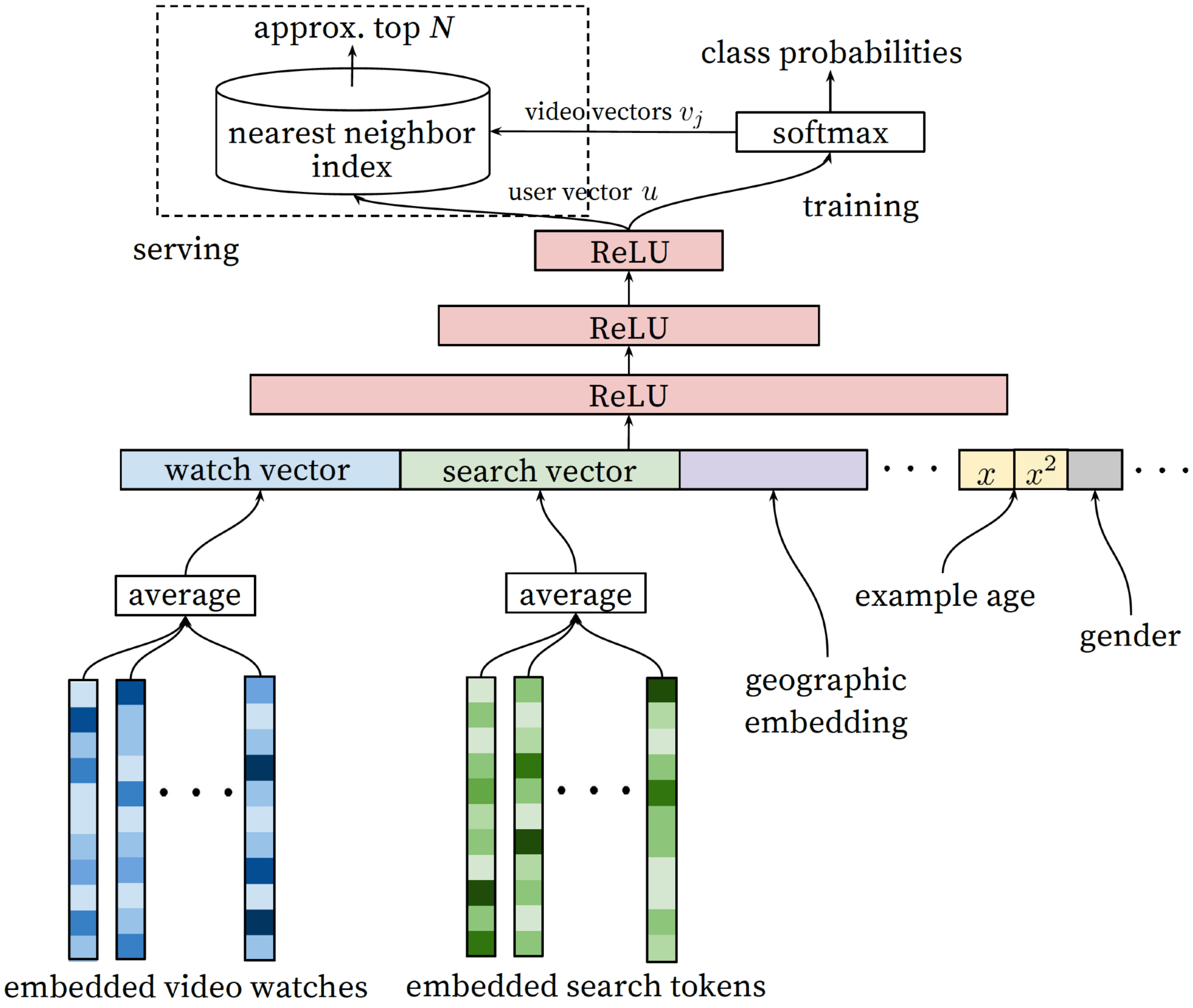
- Negative sampling: At every step, cross entropy loss is minimized for the positive examples and only a sample of the negative examples.
- Practice: ~ several thousand negative examples are sampled.
- Result: more than 100 times speedup.

# Goal: at serving time

- Compute the most likely $N$ classes (videos) for the user
- Scoring millions of items in about ~ 50-100 ms
- Requires a scheme sublinear in the number of classes
- Acceptable to be approximate

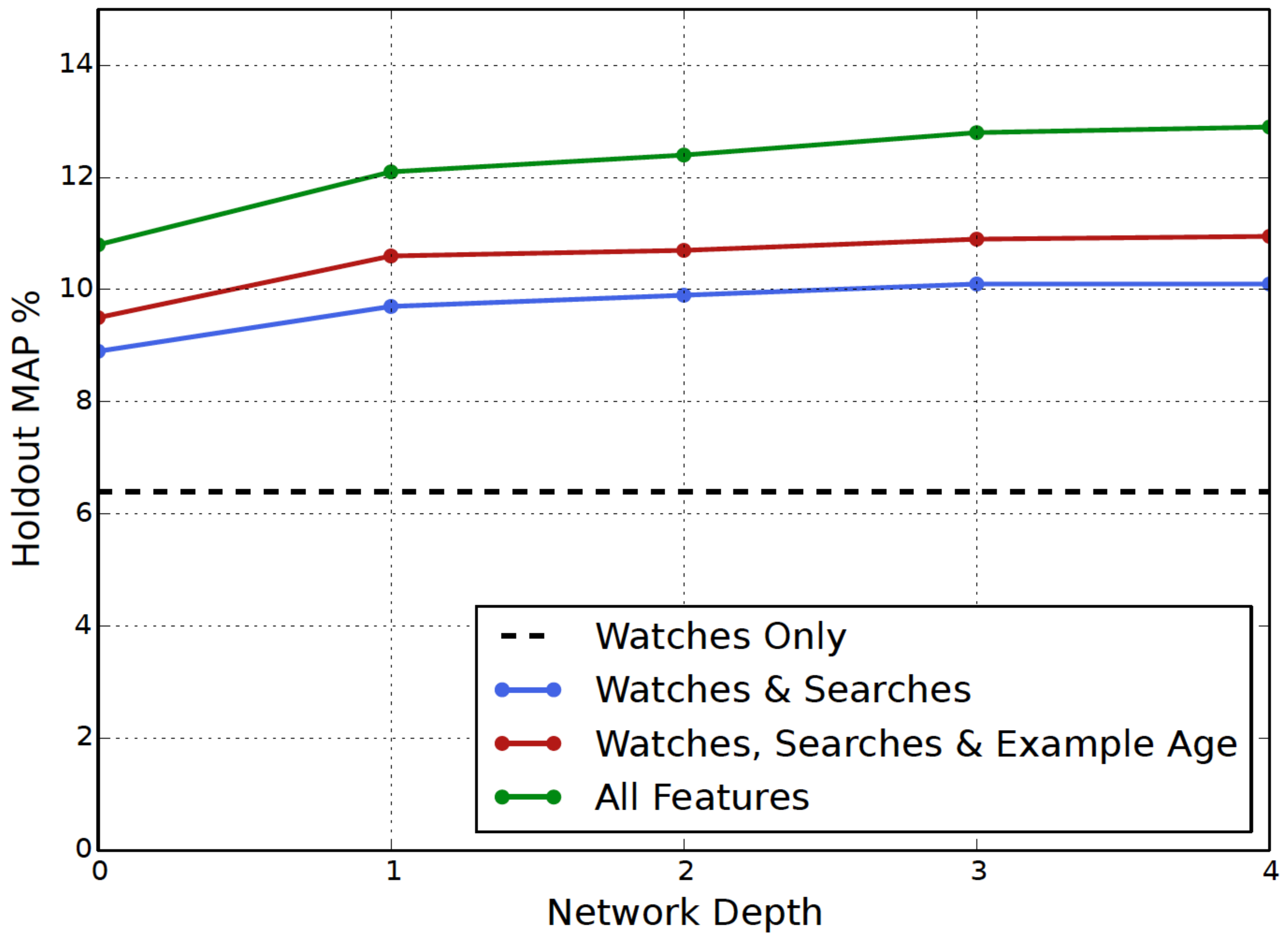
- Previous approach in YouTube: hashing
- Approach here: nearest neighbor search in the dot product space (not using calibrated softmax scores)

# The practical architecture



# Feature engineering

- Some feature engineering still required



# Result of width and depth

- The bottom of the network is the widest
- Depth 0: A linear layer simply transforms the concatenation layer to match the softmax dimension of 256
- Depth 1: 256 ReLU
- Depth 2: 512 ReLU — 256 ReLU
- Depth 3: 1024 ReLU — 512 ReLU — 256 ReLU
- Depth 4: 2048 ReLU — 1024 ReLU — 512 ReLU — 256 ReLU

| Hidden layers | weighted, per-user loss |
|---|---|
| None | 41.6% |
| 256 ReLU | 36.9% |
| 512 ReLU | 36.7% |
| 1024 ReLU | 35.8% |
| 512 ReLU $\rightarrow$ 256 ReLU | 35.2% |
| 1024 ReLU $\rightarrow$ 512 ReLU | 34.7% |
| 1024 ReLU $\rightarrow$ 512 ReLU $\rightarrow$ 256 ReLU | 34.6% |

# Neural Collaborative Filtering

# Recall: The latent factor model

$\leftarrow M$ users $\rightarrow$

| | | | |
|---|---|---|---|
| 8 | | | 5 |
| | | 7 | |
| 7 | 2 | | |
| | 4 | | |
| | | | 3 |

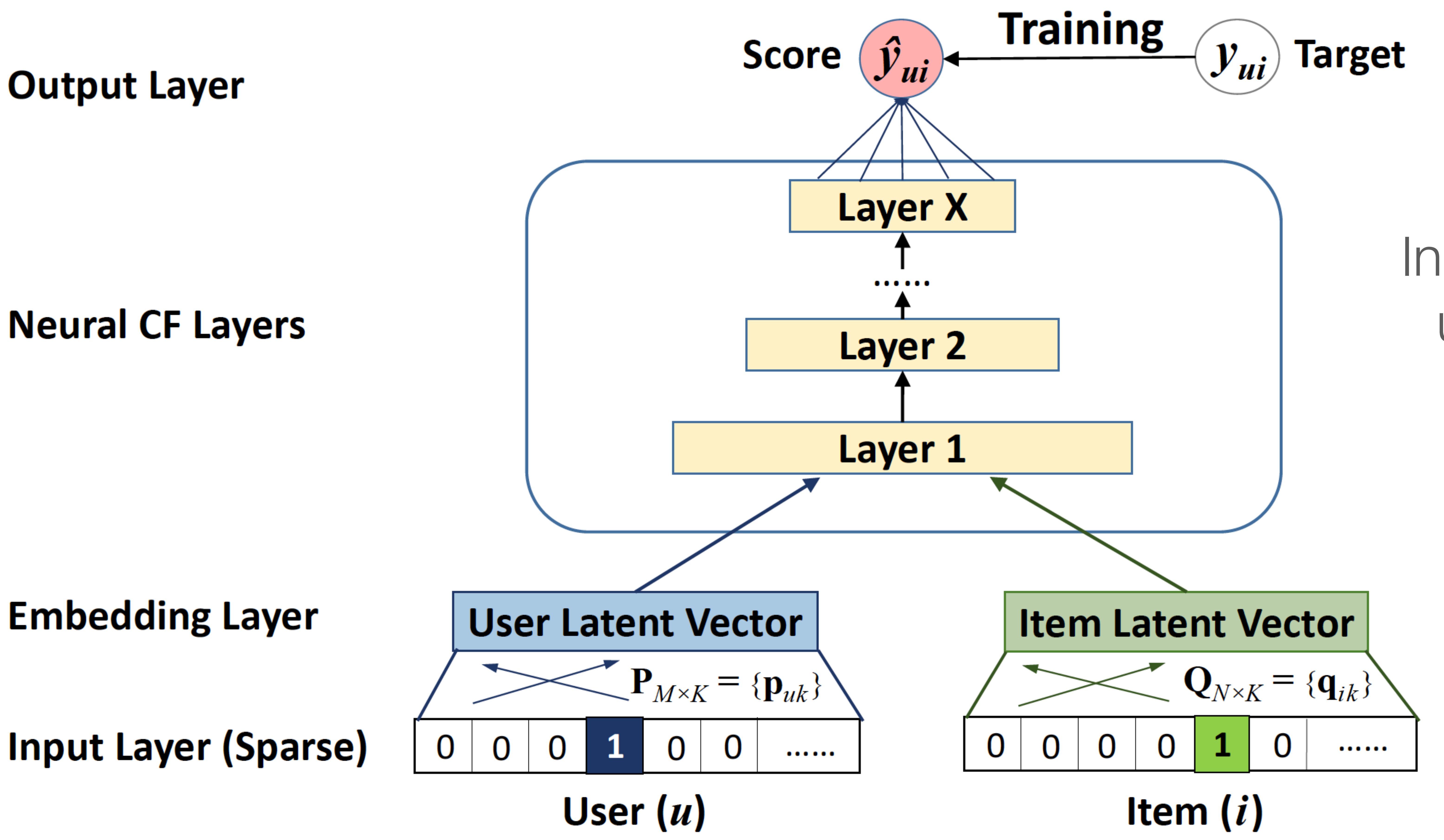
$\leftarrow N$ items $\rightarrow$

- An interaction matrix $Y \in \mathbb{R}^{M \times N}$, predicted interaction matrix $\hat{Y}$
  - Most entries of $Y$ are blank (unknown)
- Let $y_{ui}$ be preference score of $u$ to item $i$, $\hat{y}_{ui}$ be the predicted $y_{ui}$
- Observed set $O$, unobserved set $O^{-}$
- Goal: predict the missing values of $Y$

- Latent factor model: the users and items are represented in a $k$ dimensional space ( $k \ll M, N$ )
- User $u$ represented by $\mathbf{p}_u \in \mathbb{R}^k$ and item $i$ represented by $\mathbf{q}_i \in \mathbb{R}^k$
- Denote the corresponding matrices by $P \in \mathbb{R}^{M \times k}$ and $Q \in \mathbb{R}^{N \times k}$
- Traditional MF-method: estimation of the interaction by inner-product of the corresponding vectors:

$$\hat{y}_{ui} = f(u, i \mid \mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^T \mathbf{q}_i$$

# Architecture for NCF




Instead of a simple inner-product use a multi-layer-network with parameters $\Theta$

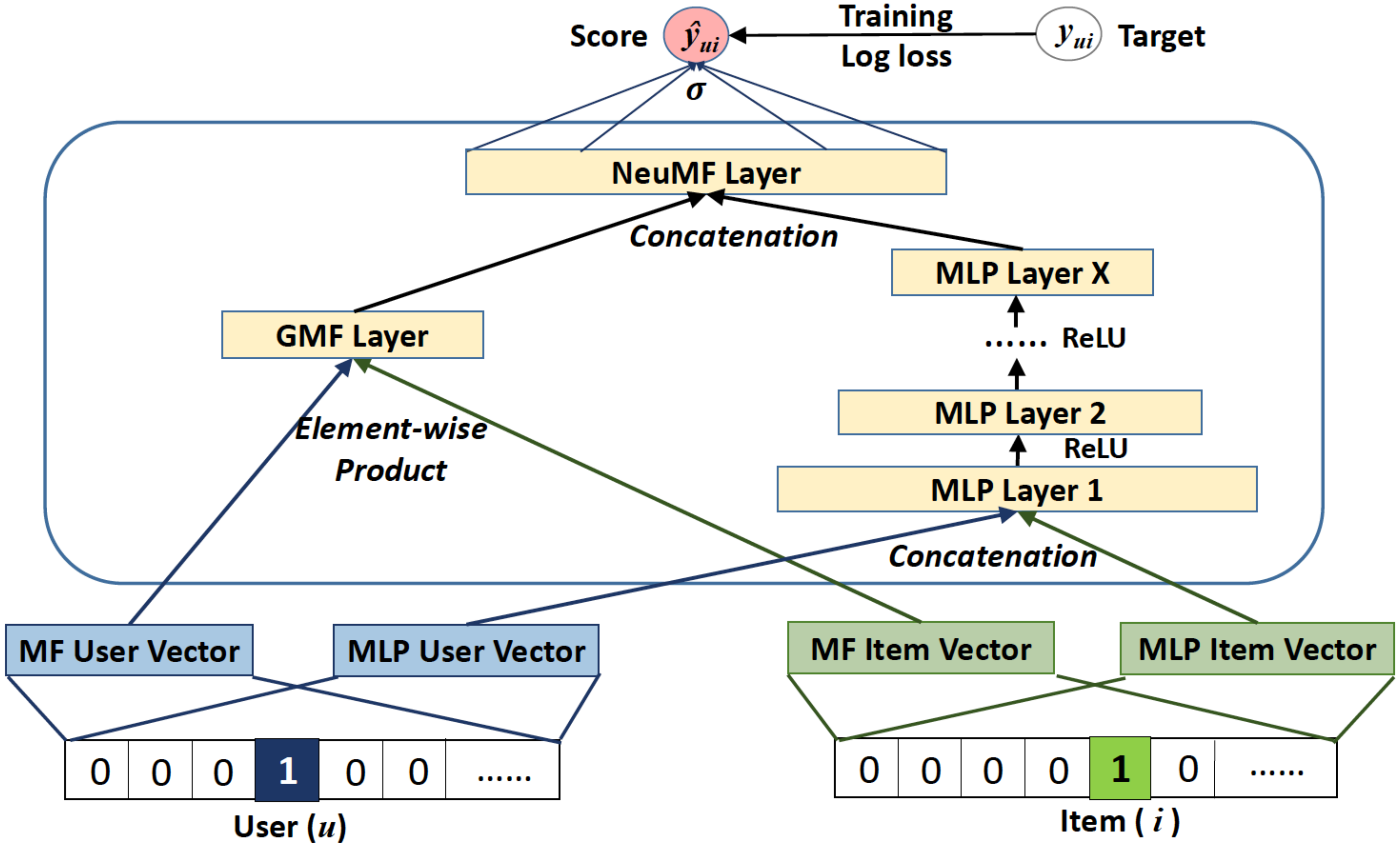
Customized input representation can be used

Denote them by $v_u^U$ (user $u$) and $v_j^I$ (item $i$) respectively

# Neural Collaborative Filtering (NCF)

- Most traditional recommendation models are linear
- Extend with MLP, add non-linearity
- Scoring function: $\hat{y}_{ui} = f(P^T v_u^U, Q^T v_i^I \,|\, P, Q, \Theta)$
- Matrix factorization based collaborative filtering can be viewed as a special case of NCF (without non-linearity)
- Training by binary cross-entropy loss (the interaction matrix is scaled to $[0,1]$)
- $$\mathscr{L} = - \sum_{u,i \in O \cup O^-} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log(1 - \hat{y}_{ui})$$
- Negative samples are picked uniformly randomly from unobserved instances

# Fusion between generalized MF and NMF




Li, Xiaopeng, and James She. "Collaborative variational autoencoder for recommender systems." In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 305-314. 2017.

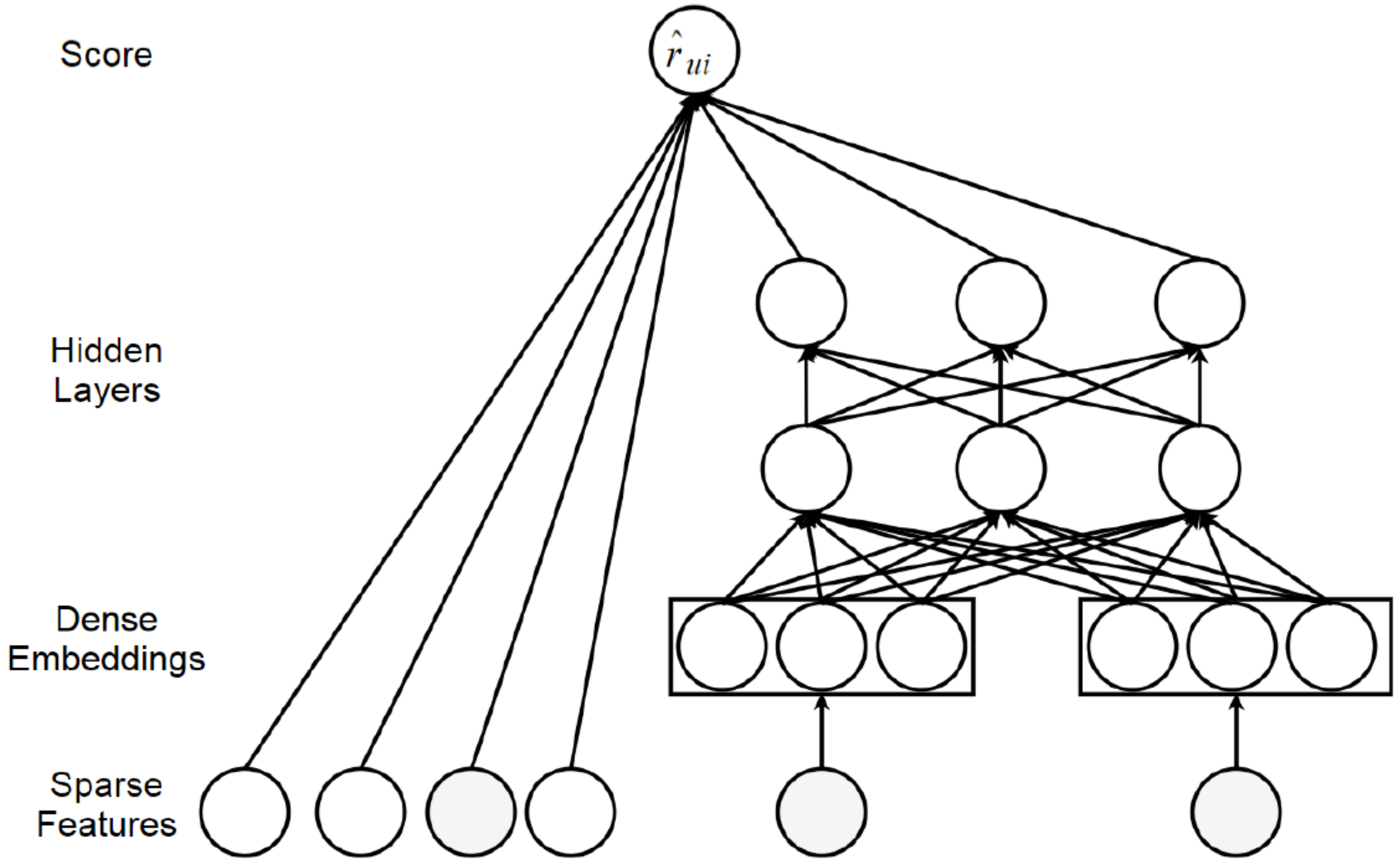
# Wide and Deep Learning

# Wide and Deep Learning

- Initially introduced in Google play store
- Can solve both regression and classification problems
- Wide component: single layer perceptron
  - Generalized linear model
  - Captures memorization
- Deep component: multi layer perceptron
  - More general and abstract representations

# Wide and Deep Learning

$$P(\hat{r}_{ui} = 1|x) = \sigma(W_{wide}^T\{x, \phi(x)\} + W_{deep}^T a^{(l_f)} + bias)$$




$$\alpha^{(l+1)} = f(W_{deep}^{(l)} a^{(l)} + b^{(l)})$$

Each layer of the deep component

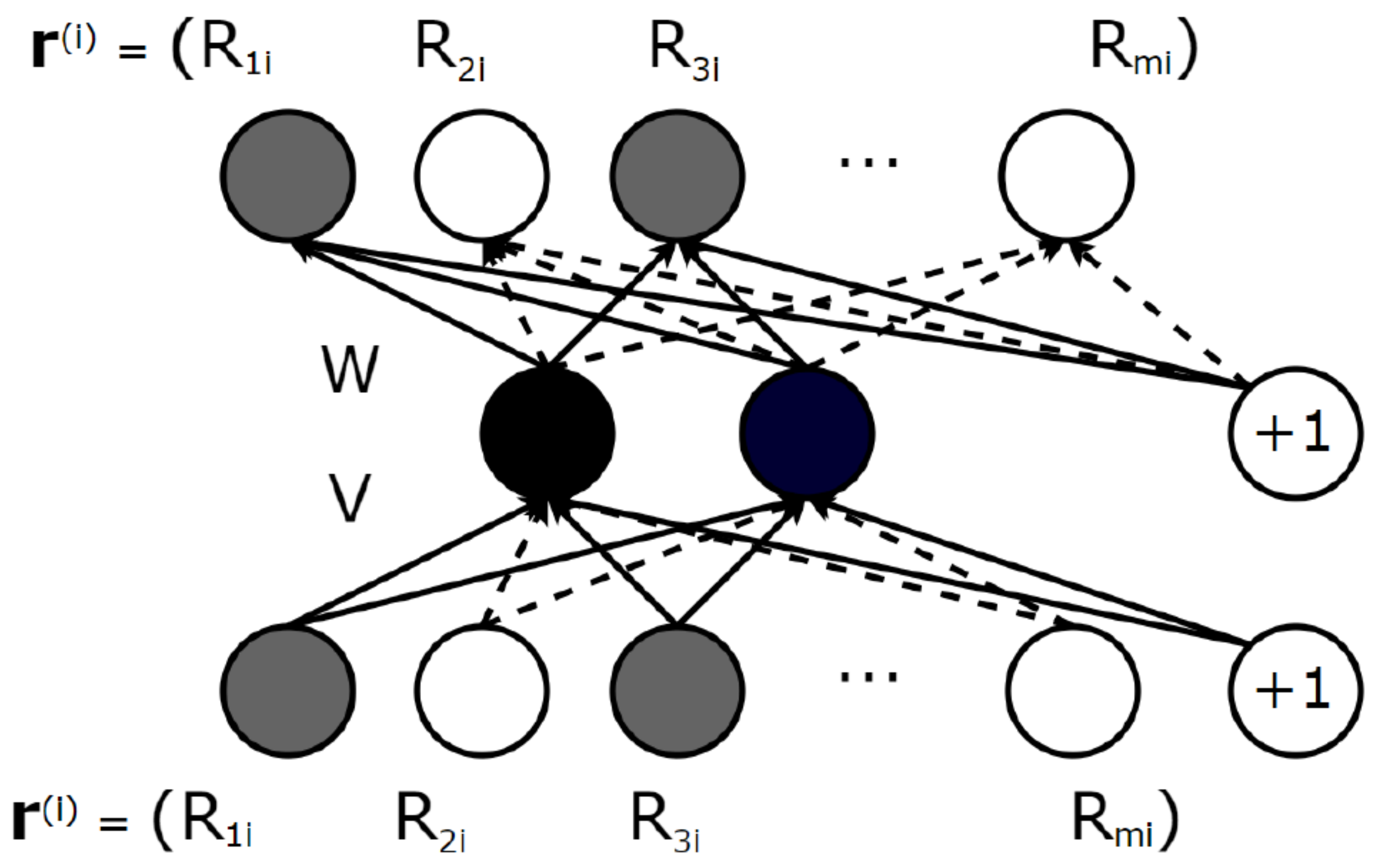
$$y = W_{wide}^T\{x, \phi(x)\} + b$$

Wide learning

# Recommendation using Autoencoders

# AutoRec: Autoencoder based Collaborative Filtering

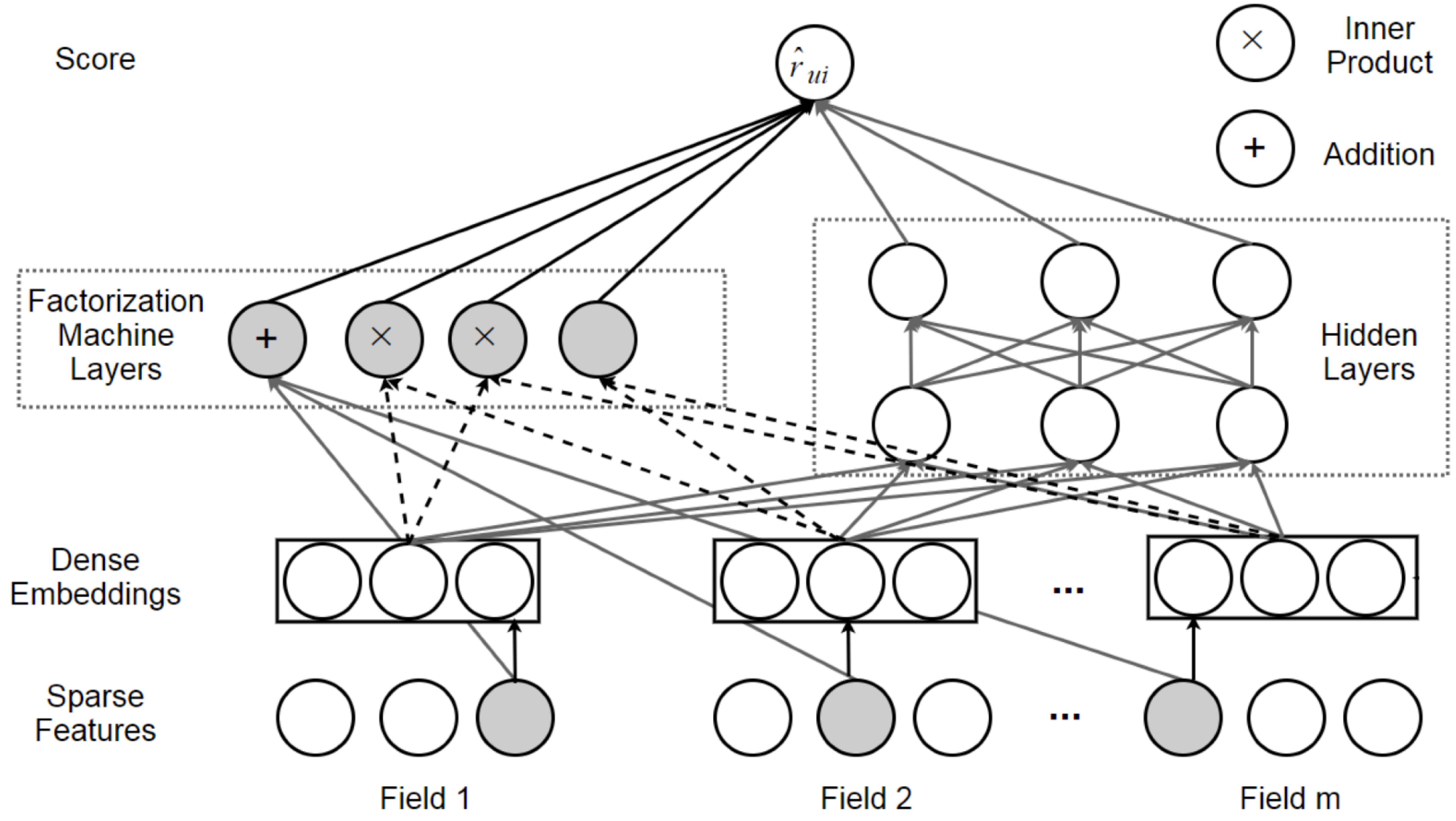
- Aim: take *partial* user or item vectors $\mathbf{r}^{(u)}$ or $\mathbf{r}^{(i)}$ and reconstruct them in the output layer
- Item-based or user-based AutoRec
- I-AutoRec reconstruct: $$h(\mathbf{r}^{(i)}; \theta) = f(W \cdot g(V \cdot \mathbf{r}^{(i)} + \mu) + b)$$
- *f* and *g* are activation functions, $$\theta = \{W, V, \mu, b\}$$
- Objective fu $$\underset{\theta}{argmin} \sum_{i=1}^{N} \| \mathbf{r}^{(i)} - h(\mathbf{r}^{(i)}; \theta) \|_{\mathcal{O}}^2 + \lambda \cdot reg$$




- I-AutoRec better than U-AutoRec, because $\mathbf{r}^{(u)}$ have higher variance
- Different combination of *f* and *g* important
- Deeper = better

# Deep Factorization Machine

- Seamless integration of factorization and MLP
- Factorization machine: model lower order interactions
- Deep network: non-linear activations and higher order interactions
- Prediction score: $$\hat{r}_{ui} = \sigma(y_{FM}(x) + y_{MLP}(x))$$



# References


- He, Xiangnan, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. "Neural collaborative filtering." In *Proceedings of the 26th international conference on world wide web*, pp. 173-182. 2017.
- Li, Xiaopeng, and James She. "Collaborative variational autoencoder for recommender systems." In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 305-314. 2017.