IJCAI-17
MELBOURNE

Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks

Jun Xiao[1], **Hao Ye**[1], Xiangnan He[2], Hanwang Zhang[2], Fei Wu[1], Tat-Seng Chua[2]

[1]College of Computer Science
Zhejiang University

[2]School of Computing
National University of Singapore

Example: Predicting Customers' Income

- Inputs:
  a) Occupation = { banker, engineer, … }
  b) Level = { junior, senior }
  c) Gender = { male, female }

Junior bankers have a lower income than junior engineers, but this is the reverse case for senior bankers

#	Occupation	Level	Gender
1	Banker	Junior	Male
2	Engineer	Junior	Male
3	Banker	Junior	Female
4	Engineer	Junior	Female
5	Banker	Senior	Male
6	Engineer	Senior	Male
7	Banker	Senior	Female
8	Engineer	Senior	Female
…	…	…	…

One-hot Encoding

Feature vector **X**									Target y
#	Occupation			Level		Gender			
	B	E	…	J	S	M	F		
1	1	0	…	1	0	1	0		0.4
2	0	1	…	1	0	1	0		0.6
3	1	0	…	1	0	0	1		0.4
4	0	1	…	1	0	0	1		0.6
5	1	0	…	0	1	1	0		0.9
6	0	1	…	0	1	1	0		0.7
7	1	0	…	0	1	0	1		0.9
8	0	1	…	0	1	0	1		0.7
…	…			…		…			

Linear Regression (LR)

- Model Equation: $\hat{y}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = \sum_{i=1}^{n} w_i x_i$
- Example:

Occupation	Level	Gender
Banker	Junior	Male

$$\hat{y}(\mathbf{x}) = \mathbf{w}_{Banker} + \mathbf{w}_{Junior} + \mathbf{w}_{Male}$$

- Drawbacks: Cannot learn cross-feature effects like:

"Junior bankers have lower income than junior engineers, while senior bankers have higher income than senior engineers"

Factorization Machines (FM)

- Model Equation:

$$\hat{y}_{FM}(\mathbf{x}) = \underbrace{w_0 + \sum_{i=1}^{n} w_i x_i}_{\text{linear regression}} + \underbrace{\sum_{i=1}^{n} \sum_{j=i+1}^{n} \hat{w}_{ij} x_i x_j}_{\text{pair-wise feature interactions}}$$

  a) $\hat{\omega}_{ij} = \mathbf{v}_i^T \mathbf{v}_j$

  b) $\mathbf{v}_i \in \mathbb{R}^k$: the embedding vector for feature i

  c) k: the size of embedding vector

- Example:

Occupation	Level	Gender
Banker	Junior	Male

$$\hat{y}(\mathbf{x}) = \mathbf{w}_{Banker} + \mathbf{w}_{Junior} + \mathbf{w}_{Male} + \langle \boldsymbol{v}_{Banker}, \boldsymbol{v}_{Junior} \rangle + \langle \boldsymbol{v}_{Banker}, \boldsymbol{v}_{Male} \rangle + \langle \boldsymbol{v}_{Junior}, \boldsymbol{v}_{Male} \rangle$$

- Drawbacks: Model all factorized interactions with the same weight.
- For example, the *gender* variable above is less important than others for estimating the target.

Attentional Factorization Machines (AFM)

- Our main contribution:
  a) Pair-wise Interaction Layer
  b) Attention-based Pooling

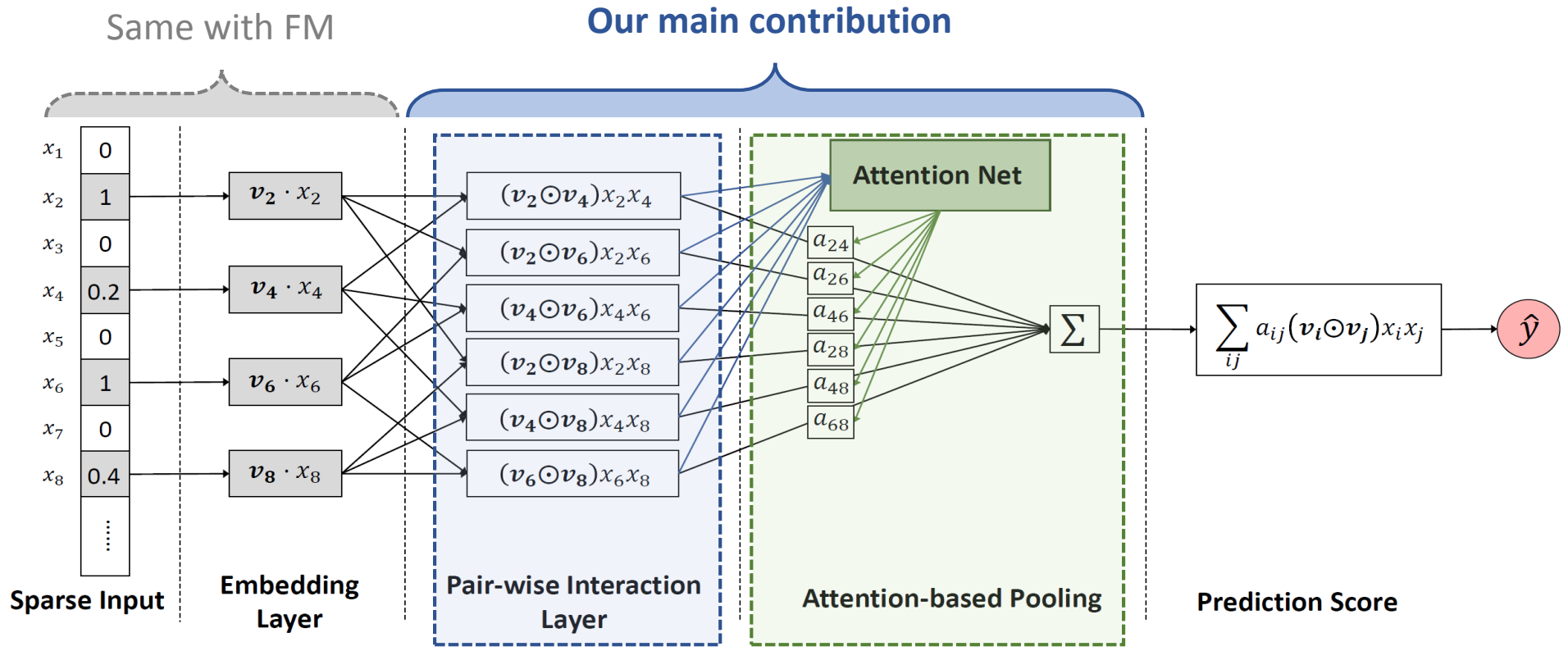


Figure 1: The neural network architecture of our proposed Attentional Factorization Machine model.

Contribution #1: Pair-wise Interaction Layer

- Layer Equation: $f_{PI}(\mathcal{E}) = \{(\mathbf{v}_i \odot \mathbf{v}_j)x_i x_j\}_{(i,j)\in\mathcal{R}_x}$

Where:

a) $\odot$: element-wise product of two vectors

b) $\mathcal{R}_x = \{(i,j)\}_{i\in\mathcal{X},j\in\mathcal{X},j>i}$

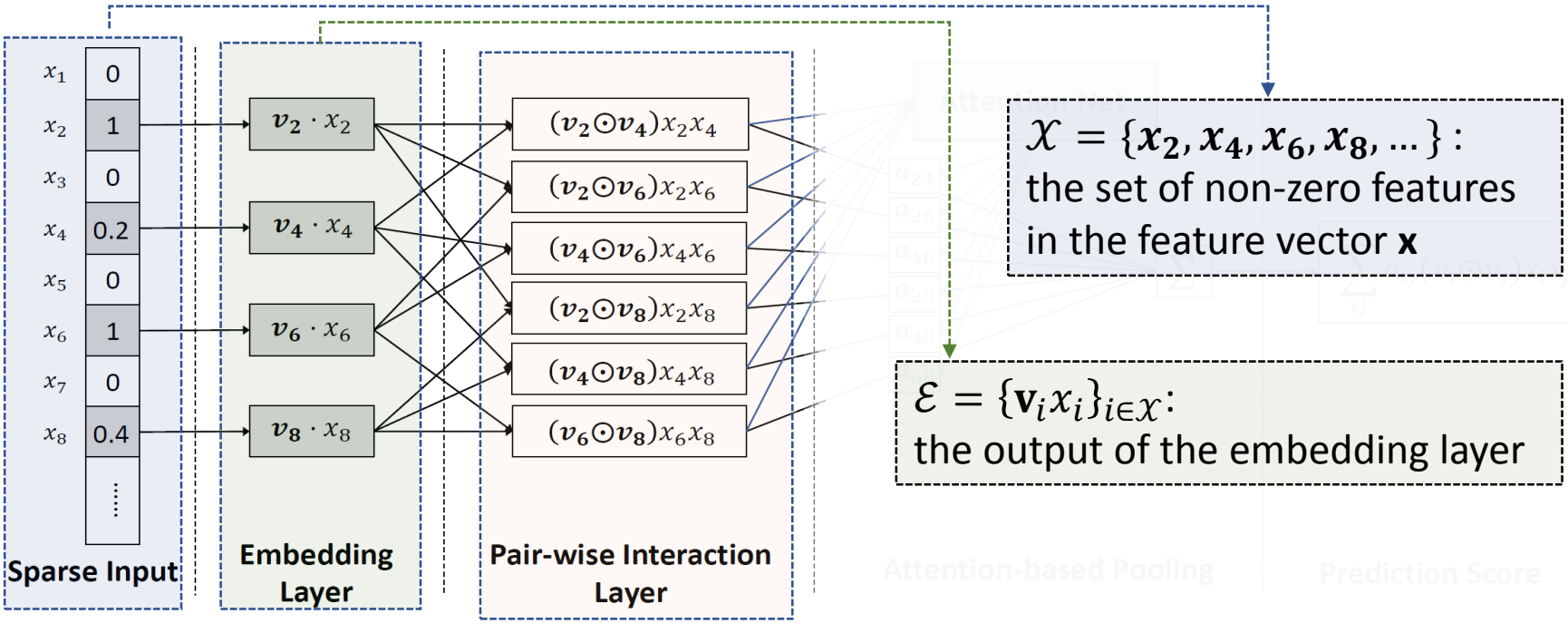


Figure 1: The neural network architecture of our proposed Attentional Factorization Machine model.

Express FM as a Neural Network

- Sum pooling over pair-wise interaction layer:

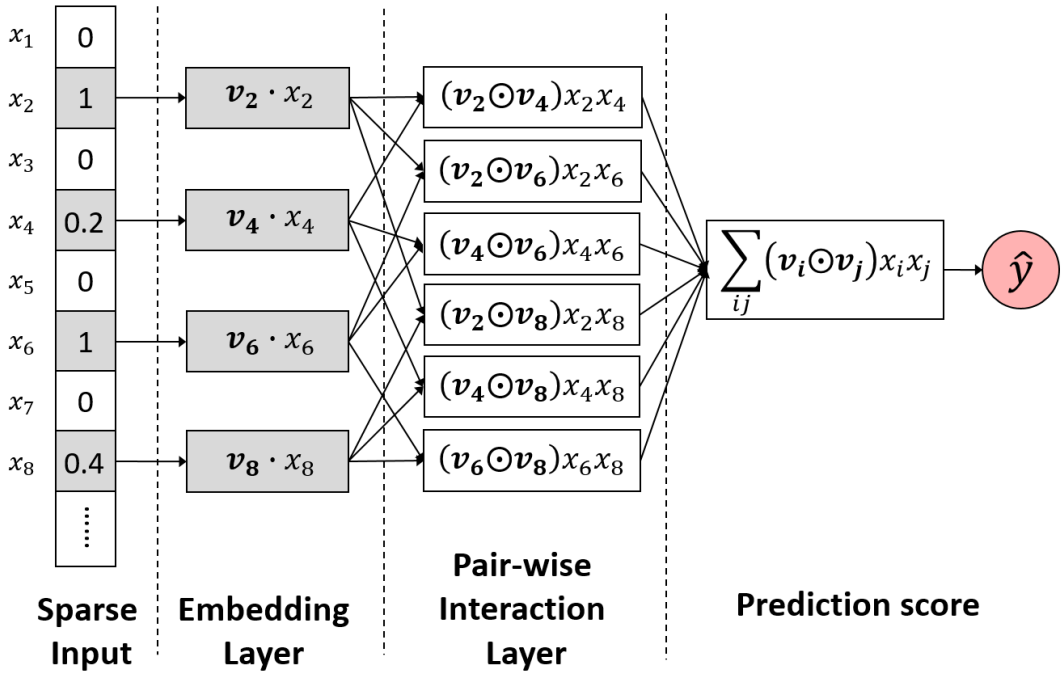
$$\hat{y} = \mathbf{p}^T \sum_{(i,j)\in\mathcal{R}_x} (\mathbf{v}_i \odot \mathbf{v}_j) x_i x_j + b$$

Where:

a) $\mathbf{p}^T \in \mathbb{R}^k$: weights for the prediction layer

b) b: bias for the prediction layer

- By fixing **p** to **1** and *b* to 0, we can exactly recover the FM model



Contribution #2: Attention-based Pooling Layer

- The idea of attention is to **allow different parts contribute differently** when compressing them to a single representation.
- Motivated by the drawback of FM, We propose to employ the attention mechanism on feature interactions **by performing a weighted sum on the interacted vectors.**

$$f_{Att}(f_{PI}(\mathcal{E})) = \sum_{(i,j)\in\mathcal{R}_x} a_{ij}(\mathbf{v}_i \odot \mathbf{v}_j)x_i x_j$$

Attention score for feature interaction $\langle i, j \rangle$

$$f_{PI}(\mathcal{E}) = \{(\mathbf{v}_i \odot \mathbf{v}_j)x_i x_j\}_{(i,j)\in\mathcal{R}_x}$$

Attention-based Pooling Layer

- Definition of attention network:

$$a'_{ij} = \mathbf{h}^T ReLU(\mathbf{W}(\mathbf{v}_i \odot \mathbf{v}_j)x_i x_j + \mathbf{b}),$$

$$a_{ij} = \frac{\exp(a'_{ij})}{\sum_{(i,j)\in\mathcal{R}_x} \exp(a'_{ij})},$$

Where:

a) $\mathbf{W} \in \mathbb{R}^{t\times k}$, $\mathbf{b} \in \mathbb{R}^t$, $\mathbf{h} \in \mathbb{R}^t$: parameters

b) t: ***attention factor***, denoting the hidden layer size of the attention network

- The output a_{ij} is a k dimensional vector

Summarize of AFM

- The overall formulation of AFM:

$$\hat{y}_{AFM}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \mathbf{p}^T \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{ij}(\mathbf{v}_i \odot \mathbf{v}_j) x_i x_j$$

- For comparison, the overall formulation of FM in neural network is:

$$\hat{y}_{FM}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \mathbf{p}^T \sum_{i=1}^{n} \sum_{j=i+1}^{n} (\mathbf{v}_i \odot \mathbf{v}_j) x_i x_j$$

- Attention factors bring AFM stronger representation ability than FM.

Experiments

- Task #1: Context-aware App Usage Prediction
  a) Frappe data: userID, appID, and 8 context variables (sparsity: 99.81%)
- Task #2: Personalized Tag Recommendation
  a) MovieLens data: userID, movieID and tag (sparsity: 99.99%)

Table 1: Statistics of the evaluation datasets.

Dataset	Instance#	Feature#	User#	Item#
Frappe	288,609	5, 382	957	4,082
MovieLens	2,006,859	90, 445	17,045	23,743

- Randomly split: 70% (training), 20% (validation), 10% (testing)
- Evaluated prediction error by RMSE (lower score, better performance).

Baselines

- 1. LibFM:
  - The official implementation of second-order FM
- 2. HOFM:
  - A 3rd party implementation of high-order FM.
  - We experimented with order size 3.
- 3. Wide&Deep:
  - Same architecture as the paper: 3 layer MLP: 1024->512->256
- 4. DeepCross:
  - Same structure as the paper: 10 layer (5 ResUnits): 512->512->256->128->64)
- k (the size of embedding feature) is set to 256 for all baselines and our AFM model.

I. Performance Comparison

- For Wide&Deep, DeepCross and AFM, **pretraining** their feature embeddings with FM leads to a lower RMSE than end-to-end training with a random initialization.

	Frappe		MovieLens	
Method	**Param#**	**RMSE**	**Param#**	**RMSE**
LibFM	1.38M	0.3385	23.24M	0.4735
HOFM	2.76M	0.3331	46.40M	0.4636
Wide&Deep	4.66M	0.3246	24.69M	0.4512
DeepCross	8.93M	0.3548	25.42M	0.5130
AFM	**1.45M**	**0.3102**	**23.26M**	**0.4325**

M means *million*

1. Linear way of high-order modelling has minor benefits.

2. Wide&Deep slightly betters LibFM while DeepCross suffers from overfitting.

3. AFM significantly betters LibFM with fewest additional parameters.

II. Hyper-parameter Investigation

- Dropout ratio (on embedding layer) = *Best
- **λ (L_2 regularization on attention network) = ?**
- Attention factor = 256 = k (size of embedding size)

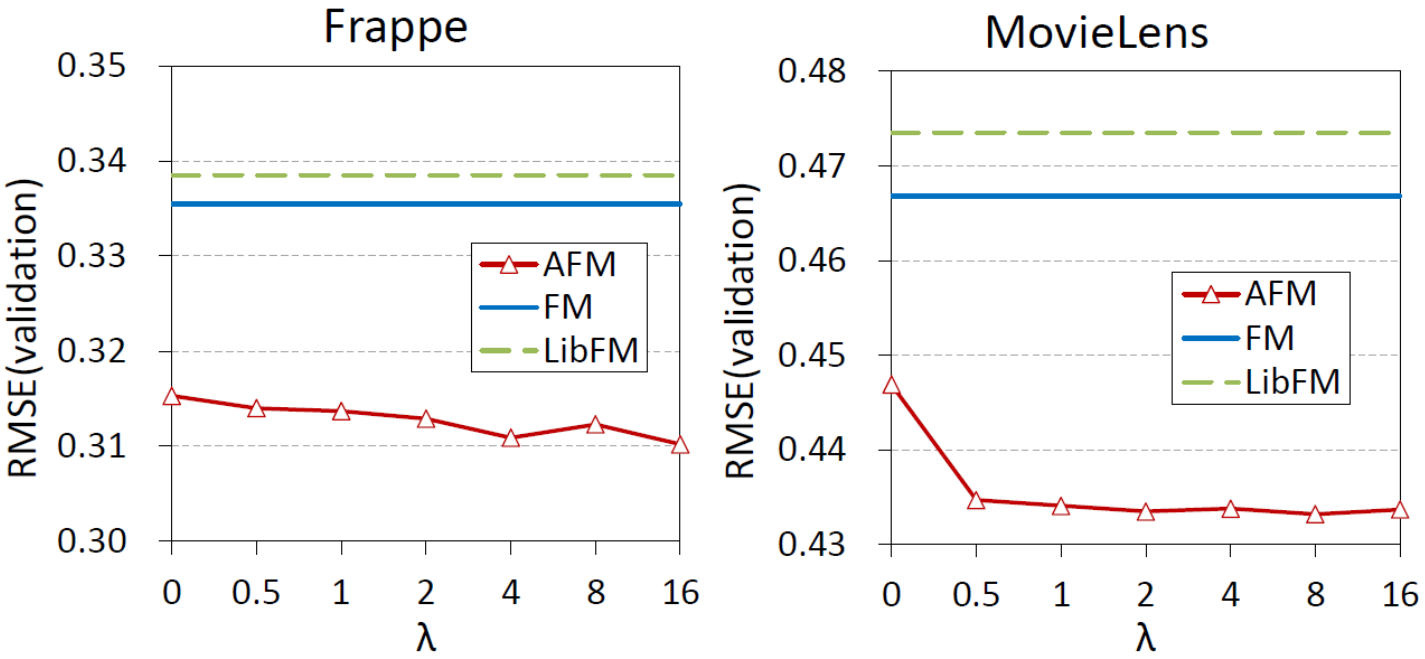


Figure 3: Validation error of AFM *w.r.t.* different regularization strengths on the attention network

II. Hyper-parameter Investigation

- Dropout ratio = *Best
- λ (L_2 regularization on attention network) = *Best
- **Attention factor = ?**

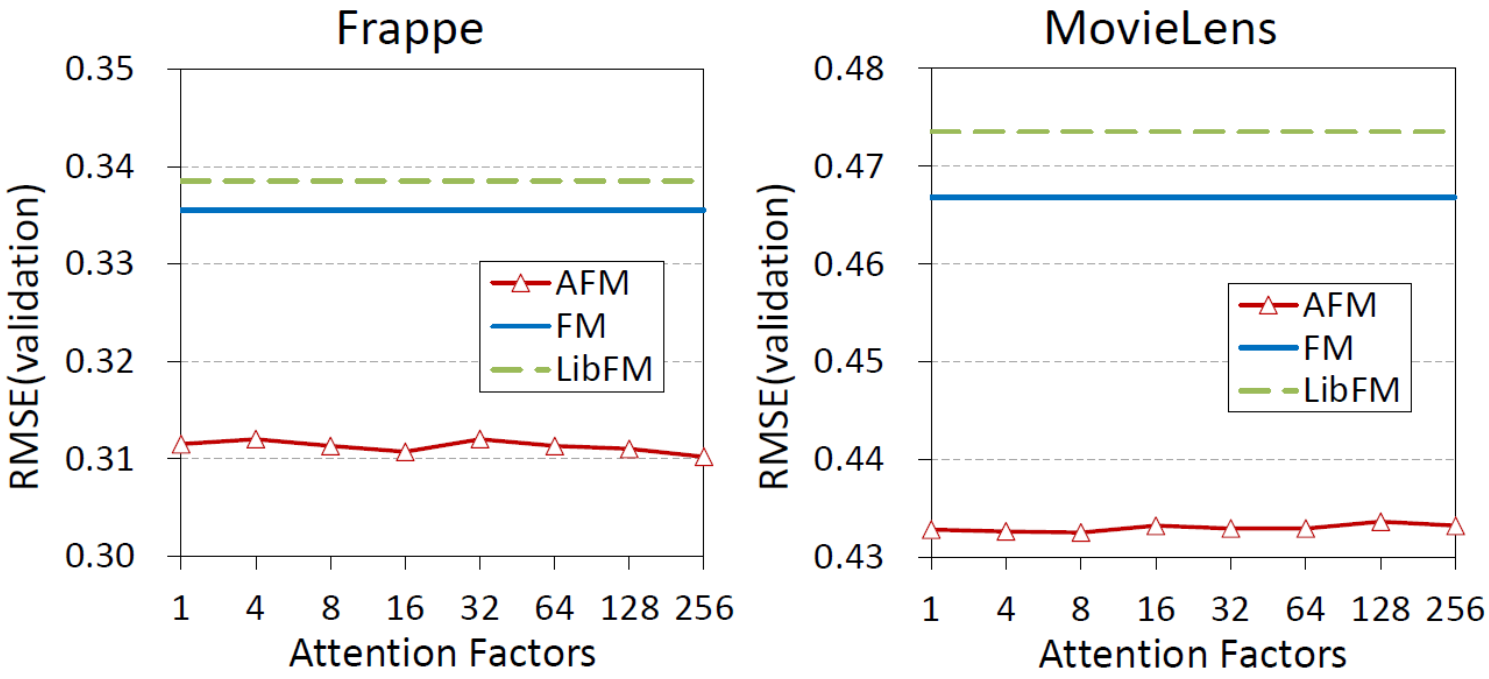


Figure 4: Validation error of AFM *w.r.t.* different attention factors

II. Hyper-parameter Investigation

- Dropout ratio = *Best
- λ (L_2 regularization on attention network) = *Best
- Attention factor = *Best

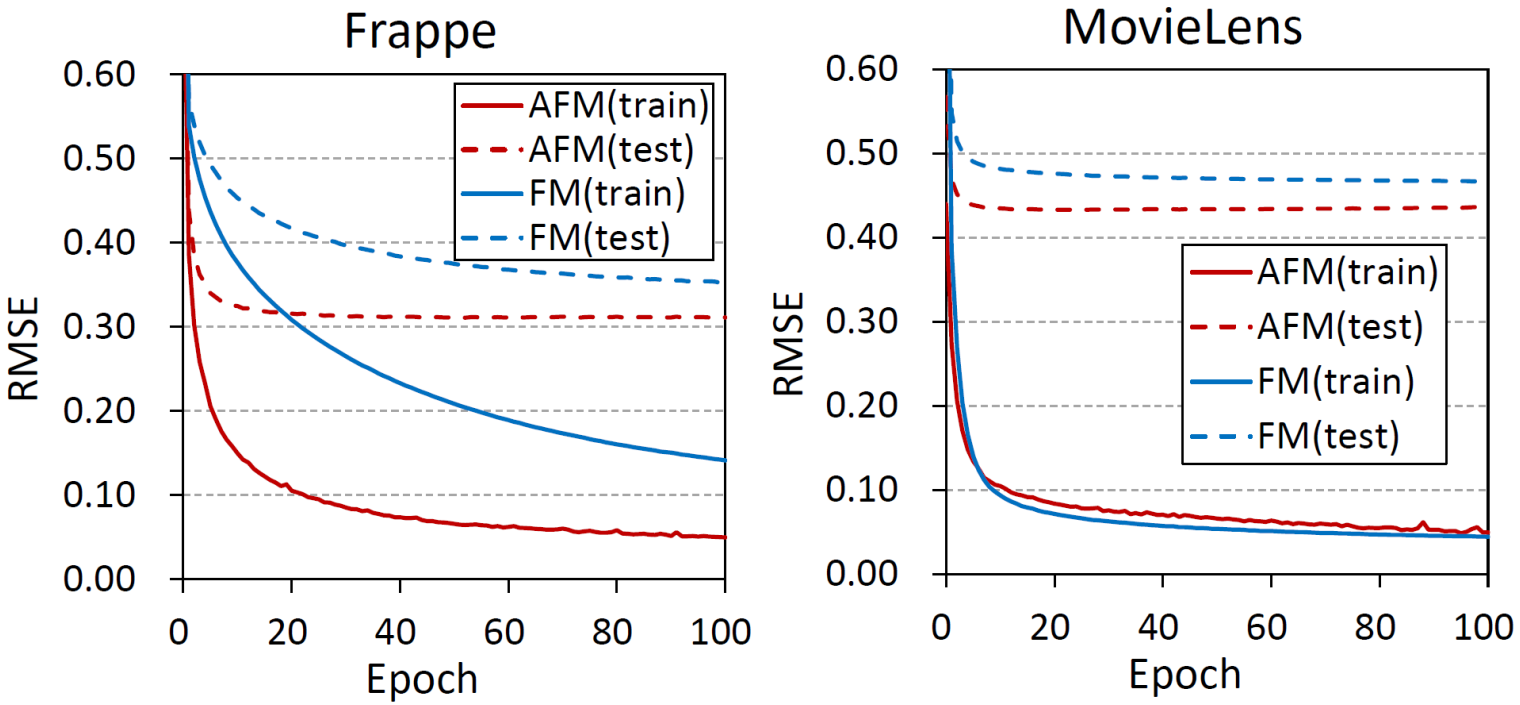


Figure 5: Training and test error of each epoch

III. Micro-level Analysis

- **FM**: Fix a_{ij} to a uniform number $\frac{1}{|\mathcal{R}_x|}$
- **FM+A:** Fix the feature embeddings pretrained by **FM** and train the attention network only.
- **AFM is more explainable by learning the weight of feature interactions**
- The performance is improved about 3% in this case.

Table 1: The attention_score*interaction_score of each feature interaction of three test examples on MovieLens.

#	Model	User-Item	User-Tag	Item-Tag	$\hat{y}$
1	FM	0.33*-1.81	0.33*-2.65	0.33*4.55	0.03
	FM+A	0.34*-1.81	0.27*-2.65	0.38*4.55	0.39
2	FM	0.33*-1.62	0.33*-1.00	0.33*3.32	0.24
	FM+A	0.38*-1.62	0.20*-1.00	0.42*3.32	0.56
3	FM	0.33*-1.40	0.33*-1.26	0.33*4.68	0.67
	FM+A	0.33*-1.40	0.29*-1.26	0.37*4.68	0.89

Conclusion

- Our proposed AFM enhances FM by learning the importance of feature interactions with an attention network, and **achieved a 8.6% relative improvement**.
  - improves the representation ability
  - improves the interpretability of a FM model

- This work is orthogonal with our recent work on neural FM [He and Chua, SIGIR-2017]
  - in that work we develops deep variants of FM for modelling high-order feature interactions

Future works

- Explore deep version for AFM by stacking multiple non-linear layers above the attention-based pooling layer
- Improve the learning efficiency by using learning to hash and data sampling
- Develop FM variants for semi-supervised and multi-view learning
- Explore AFM on modelling other types of data for different applications, such as:
  a) Texts for question answering,
  b) More semantic-rich multi-media content

Thanks!

Codes:

https://github.com/hexiangnan/attentional_factorization_machine