Knowledge Tracing Machines: Factorization Machines for Knowledge Tracing

Jill-Jênn Vie   Hisashi Kashima

京都大学 KYOTO UNIVERSITY

The University of Hawaiʻi at Mānoa, January 24, 2019

https://arxiv.org/abs/1811.03388

Who is this person?

. . . Actually, she does not exist

5 years of GAN progress on face generation

A Style-Based Generator Architecture for Generative Adversarial Networks (Karras, Laine and Aila, 2018)
`arxiv.org/abs/1812.04948`

AI for Social Good

AI can:

- generate fakes
- recognize images/speech
- predict the next word
- play go (make decisions)

as long as you have enough data.

Can it also:

- generate exercises
- improve education
- predict student performance
- optimize human learning

as long as you have enough data?

My research

Use machine learning techniques on data that comes from humans

Outline

1. Introduction to machine learning
2. Knowledge Tracing and existing models
3. Knowledge Tracing Machines
    - Encoding data into sparse features
    - Running logistic regression or factorization machines
4. Experiments and results

What is machine learning?

Train a model on some data (e.g. pictures of cats)
Test it on new data
Tweak the parameters to optimize a value (e.g., maximize accuracy)

How to choose the best model?

Train model on some of your data, keep the remaining data hidden
Try to predict the remaining data

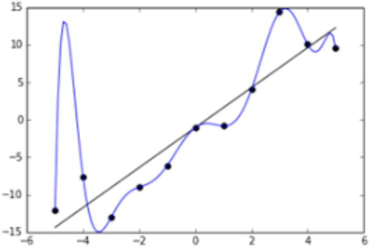
What is a good machine learning model?

Can reconstruct training data at 90%
predicts new points correctly 80% of the time

What is a bad machine learning model?

Can reconstruct training data at 100%
predicts new points correctly 20% of the time

It is called overfitting



Formally

We have existing data $(\boldsymbol{x}, y)$
and we want to learn a function $y = f(\boldsymbol{x})$.

Examples

classification	$\boldsymbol{x}$ can be an image and $y \in \{man, woman\}$
regression	$\boldsymbol{x}$ can be the temperature over the last 7 days and y is the temperature of the next day

Two typical examples

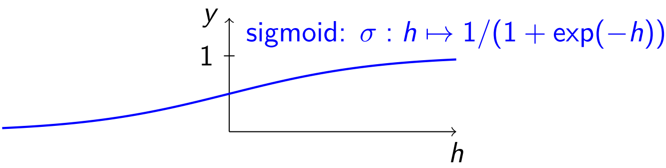
For regression: linear regression

We have existing data $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \mathbb{R}$ and we want to learn a function $y = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = \sum_i w_i x_i + b$.

Example if $d = 1$: we want to learn $wx + b = y$, an affine function!

For classification: logistic regression

We have existing data $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \{0, 1\}$ and we want to learn a function $y = \sigma(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b)$.



Is accuracy a good metric for classification?

If 99% of people are in good health, and you are making a machine learning algorithm that always predicts a person is healthy. . .
It will have accuracy 99%.

Is this a good machine?

Better metrics: AUC, F1 score, etc.

Students try exercises

Math Learning

Items	$5 - 5 = ?$	$17 - 3 = ?$	$13 - 7 = ?$
New student	○	○	×

Language Learning

	PRON	VERB	PRON	NOUN	CONJ	PRON	VERB	PRON	NOUN
correct:	She	is	my	mother	and	he	is	my	father
student:	she	is		mader	and	he	is		fhader
label:	O	O	✖	✖	O	O	O	✖	✖

Challenges

- Users can attempt a same item multiple times
- Users learn over time
- People can make mistakes that do not reflect their knowledge

Predicting student performance: knowledge tracing

Data

A population of users answering items

- Events: "User i answered item j correctly/incorrectly"

Side information

- We know the skills required to solve each item *e.g.*, $+$, $\times$
- Class ID, school ID, etc.

Goal: classification problem

Predict the performance of new users on existing items

Method

Learn parameters of questions from historical data *e.g., difficulty*
Measure parameters of new students *e.g., expertise*

Existing work

Model	Basically	Original AUC	Fixed AUC
Bayesian Knowledge Tracing (Corbett and Anderson 1994)	Hidden Markov Model	0.67	0.63
Deep Knowledge Tracing (Piech et al. 2015)	Recurrent Neural Network	0.86	0.75
Item Response Theory (Rasch 1960) (Wilson et al., 2016)	Online Logistic Regression		0.76

$$\underbrace{\text{PFA}}_{\text{LogReg}} \leq \underbrace{\text{DKT}}_{\text{LSTM}} \leq \underbrace{\text{IRT}}_{\text{LogReg}} \leq \underbrace{\text{KTM}}_{\text{FM}}$$

Limitations and contributions

- Several models for knowledge tracing were developed independently
- In our paper, we prove that our approach is more generic

In this paper

- Knowledge Tracing Machines unify most existing models
  - Encoding student data to sparse features
  - Then running logistic regression or factorization machines
- Better models found
  - It is better to estimate a bias per item, not only per skill
  - Side information improves performance more than higher dim.

Our small dataset

- User 1 answered Item 1 correct
- User 1 answered Item 2 incorrect
- User 2 answered Item 1 incorrect
- User 2 answered Item 1 correct
- User 2 answered Item 2 ???

user	item	correct
1	1	1
1	2	0
2	1	0
2	1	1
2	2	???

`dummy.csv`

Our approach

- Encode data to sparse features

user	item	correct
2	2	1
2	2	0
2	2	0
2	3	0
2	3	1
1	2	???
1	1	???

data.csv

encode →

KTM: Items, Skills, Wins, Fails
PFA: Skills, Wins, Fails
IRT: Users, Items

Users		Items			Skills			Wins			Fails		
1	2	Q_1	Q_2	Q_3	KC_1	KC_2	KC_3	KC_1	KC_2	KC_3	KC_1	KC_2	KC_3
0	1	0	1	0	1	1	0	0	0	0	0	0	0
0	1	0	1	0	1	1	0	1	1	0	0	0	0
0	1	0	1	0	1	1	0	1	1	0	1	1	0
0	1	0	0	1	0	1	1	0	2	0	0	1	0
0	1	0	0	1	0	1	1	0	2	0	0	2	1
1	0	0	1	0	1	1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0

sparse matrix X

- Run logistic regression or factorization machines

Model 1: Item Response Theory

Learn abilities θ_i for each user i
Learn easiness e_j for each item j such that:

$$Pr(\text{User } i \text{ Item } j \text{ OK}) = \sigma(\theta_i + e_j) \quad \sigma : x \mapsto 1/(1 + \exp(-x))$$
$$\text{logit}\, Pr(\text{User } i \text{ Item } j \text{ OK}) = \theta_i + e_j$$

Really popular model, used for the PISA assessment

Logistic regression

Learn $\boldsymbol{w}$ such that $\text{logit}\, Pr(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$

Graphically: IRT as logistic regression

Encoding "User i answered Item j" with sparse features:

	abilities		easinesses	
$\boldsymbol{w}$	$\boxed{\theta_i}$		$\boxed{e_j}$	
	U_i		I_j	
$\boldsymbol{x}$	1		1	
	Users		Items	

$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle = \theta_i + e_j = \operatorname{logit} Pr(\text{User } i \text{ Item } j \text{ OK})$$

Encoding into sparse features

Users			Items		
U_0	U_1	U_2	I_0	I_1	I_2
0	1	0	0	1	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	1	0	1	0
0	0	1	0	0	1

Then logistic regression can be run on the sparse features.

Oh, there's a problem

	Users			Items				
	U_0	U_1	U_2	I_0	I_1	I_2	y_{pred}	y
User 1 Item 1 OK	0	1	0	0	1	0	0.575135	1
User 1 Item 2 NOK	0	1	0	0	0	1	0.395036	0
User 2 Item 1 NOK	0	0	1	0	1	0	0.545417	0
User 2 Item 1 OK	0	0	1	0	1	0	0.545417	1
User 2 Item 2 NOK	0	0	1	0	0	1	0.366595	0

We predict the same thing when there are several attempts.

Count number of attempts: AFM

Keep a counter of attempts at skill level:

user	item	skill	correct	attempts (for the same skill)
1	1	1	1	0
1	2	2	0	0
2	1	1	0	0
2	1	1	1	1
2	2	2	0	0

$$\boldsymbol{w} \quad \overbrace{\qquad \boxed{\beta_i} \qquad}^{\text{easiness of skill}}\ \overbrace{\qquad \boxed{\gamma_j} \qquad}^{\text{bonus per attempt}}$$

$$\boldsymbol{x} \quad \begin{array}{|c|c|} \multicolumn{1}{c}{S_k} & \multicolumn{1}{c}{N_{ik}} \\ \hline \qquad 1 \qquad & \qquad 4 \qquad \\ \hline \multicolumn{1}{c}{\text{Skills}} & \multicolumn{1}{c}{\text{Attempts}} \end{array}$$

Count successes and failures: PFA

Count separately successes W_{ik} and fails F_{ik} of student i over skill k.

user	item	skill	correct	wins	fails
1	1	1	1	0	0
1	2	2	0	0	0
2	1	1	0	0	0
2	1	1	1	0	1
2	2	2	0	0	0

	easiness of skill	bonus per success	bonus per failure
$\boldsymbol{w}$	$\boxed{\beta_i}$	$\boxed{\gamma_j}$	$\boxed{\delta_j}$
	S_k	W_{ik}	F_{ik}
$\boldsymbol{x}$	1	1	1
	Skills	Wins	Fails

Model 2: Performance Factor Analysis

W_{ik}: how many successes of user i over skill k (F_{ik}: #failures)

Learn β_k, γ_k, δ_k for each skill k such that:

$$\text{logit } Pr(\text{User } i \text{ Item } j \text{ OK}) = \sum_{\text{Skill } k \text{ of Item } j} \beta_k + W_{ik}\gamma_k + F_{ik}\delta_k$$

Skills			Wins			Fails		
S_0	S_1	S_2	S_0	S_1	S_2	S_0	S_1	S_2
0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0

Better!

	Skills			Wins			Fails				
	S_0	S_1	S_2	S_0	S_1	S_2	S_0	S_1	S_2	y_{pred}	y
User 1 Item 1 OK	0	1	0	0	0	0	0	0	0	0.544	1
User 1 Item 2 NOK	0	0	1	0	0	0	0	0	0	0.381	0
User 2 Item 1 NOK	0	1	0	0	0	0	0	0	0	0.544	0
User 2 Item 1 OK	0	1	0	0	0	0	0	1	0	0.633	1
User 2 Item 2 NOK	0	0	1	0	0	0	0	0	0	0.381	0

Test on a large dataset: Assistments 2009

346860 attempts of 4217 students over 26688 items on 123 skills.

model	dim	AUC	improvement
PFA: skills, wins, fails	0	0.685	+0.07
AFM: skills, attempts	0	0.616	

Model 3: a new model (but still logistic regression)

model	dim	AUC	improvement
KTM: items, skills, wins, fails	0	0.746	+0.06
IRT: users, items	0	0.691	
PFA: skills, wins, fails	0	0.685	+0.07
AFM: skills, attempts	0	0.616	

Here comes a new challenger

How to model pairwise interactions with side information?

Logistic Regression

Learn a 1-dim bias for each feature (each user, item, etc.)

Factorization Machines

Learn a 1-dim bias and a k-dim embedding for each feature

How to model pairwise interactions with side information?

If you know user i attempted item j on mobile (not desktop)
How to model it?

y: score of event "user i solves correctly item j"

IRT

$$y = \theta_i + e_j$$

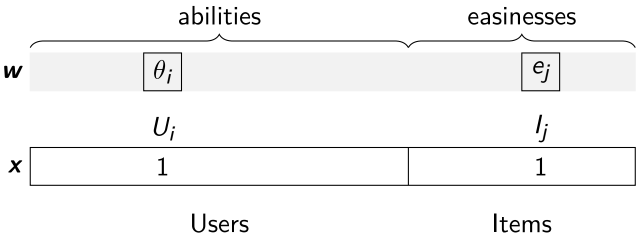
Multidimensional IRT (similar to collaborative filtering)

$$y = \theta_i + e_j + \langle \boldsymbol{v}_{\text{user } \boldsymbol{i}}, \boldsymbol{v}_{\text{item } \boldsymbol{j}} \rangle$$

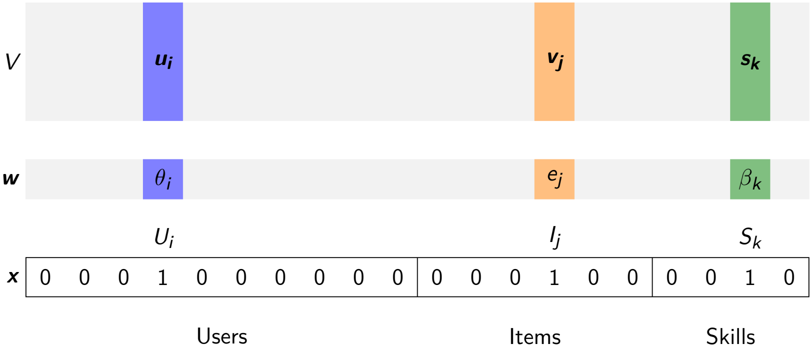
With side information

$$y = \theta_i + e_j + w_{\text{mobile}} + \langle \boldsymbol{v}_{\text{user } \boldsymbol{i}}, \boldsymbol{v}_{\text{item } \boldsymbol{j}} \rangle + \langle \boldsymbol{v}_{\text{user } \boldsymbol{i}}, \boldsymbol{v}_{\text{mobile}} \rangle + \langle \boldsymbol{v}_{\text{item } \boldsymbol{j}}, \boldsymbol{v}_{\text{mobile}} \rangle$$

Graphically: logistic regression

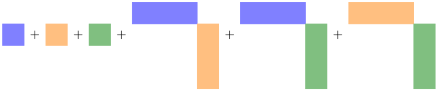


Graphically: factorization machines



Formally: factorization machines

Each user, item, skill k is modeled by bias w_k and embedding $\boldsymbol{v_k}$.



$$\operatorname{logit} p(\boldsymbol{x}) = \mu + \underbrace{\sum_{k=1}^{N} w_k x_k}_{\text{logistic regression}} + \underbrace{\sum_{1 \leq k < l \leq N} x_k x_l \langle \boldsymbol{v_k}, \boldsymbol{v_l} \rangle}_{\text{pairwise relationships}}$$

Steffen Rendle (2012). "Factorization Machines with libFM". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 3.3, 57:1–57:22. DOI: 10.1145/2168752.2168771

Training using MCMC

Priors: $w_k \sim \mathcal{N}(\mu_0, 1/\lambda_0) \quad \boldsymbol{v_k} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$
Hyperpriors: $\mu_0, \ldots, \mu_n \sim \mathcal{N}(0, 1), \lambda_0, \ldots, \lambda_n \sim \Gamma(1, 1) = U(0, 1)$

Algorithm 1 MCMC implementation of FMs

for each iteration **do**
  Sample hyperp. $(\lambda_i, \mu_i)_i$ from posterior using Gibbs sampling
  Sample weights $\boldsymbol{w}$
  Sample vectors $\boldsymbol{V}$
  Sample predictions $\boldsymbol{y}$
end for

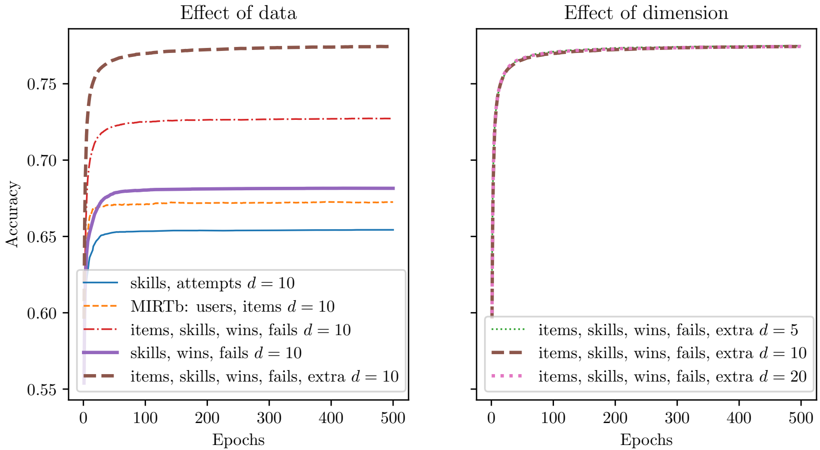
Implementation in C++ (libFM) with Python wrapper (pyWFM).

Steffen Rendle (2012). "Factorization Machines with libFM". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 3.3, 57:1–57:22. DOI: 10.1145/2168752.2168771

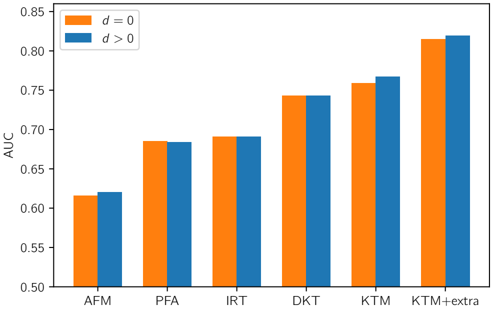
Datasets

Name	Users	Items	Skills	Skills/i	Entries	Sparsity	Attempts/u
fraction	536	20	8	2.800	10720	0.000	1.000
timss	757	23	13	1.652	17411	0.000	1.000
ecpe	2922	28	3	1.321	81816	0.000	1.000
assistments	4217	26688	123	0.796	346860	0.997	1.014
berkeley	1730	234	29	1.000	562201	0.269	1.901
castor	58939	17	2	1.471	1001963	0.000	1.000

Accuracy results on the Assistments dataset

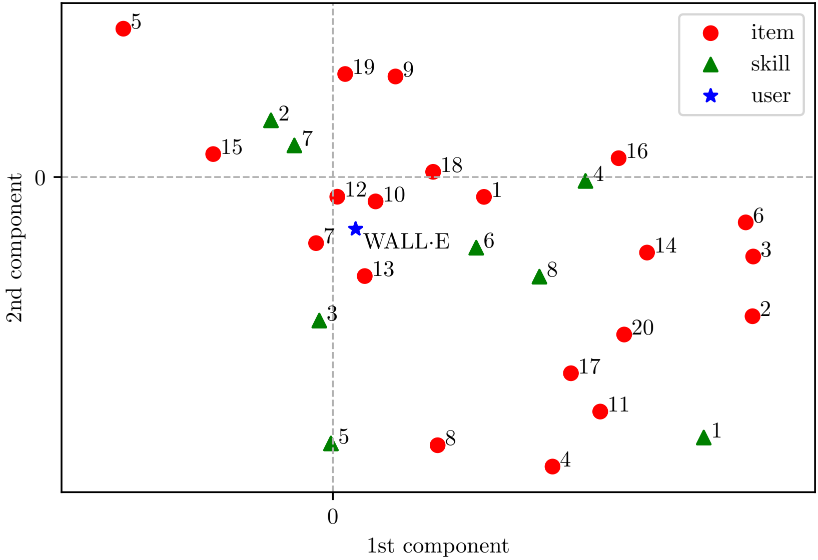


AUC results on the Assistments dataset



model	dim	AUC	improvement
KTM: items, skills, wins, fails, extra	5	0.819	
KTM: items, skills, wins, fails, extra	0	0.815	+0.05
KTM: items, skills, wins, fails	10	0.767	
KTM: items, skills, wins, fails	0	0.759	+0.02
DKT (Wilson et al., 2016)	100	0.743	+0.05
IRT: users, items	0	0.691	
PFA: skills, wins, fails	0	0.685	+0.07
AFM: skills, attempts	0	0.616	

Bonus: interpreting the learned embeddings



Take home message

Knowledge tracing machines unify many existing EDM models

- Side information improves performance more than higher d
- We can visualize learning (and provide feedback to learners)

It has better results than deep neural networks

- It is important to find simple models that fit data
- Instead of monster models that overfit data

Do you have any questions?

Read our article:

Knowledge Tracing Machines

https://arxiv.org/abs/1811.03388

Try the code:

https://github.com/jilljenn/ktm

Feel free to chat (or tell me about your favorite manga or anime):

vie@jill-jenn.net

Corbett, Albert T and John R Anderson (1994). "Knowledge tracing: Modeling the acquisition of procedural knowledge". In: *User modeling and user-adapted interaction* 4.4, pp. 253–278.

Piech, Chris, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein (2015). "Deep knowledge tracing". In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 505–513.

Rasch, Georg (1960). "Studies in mathematical psychology: I. Probabilistic models for some intelligence and attainment tests.". In:

Rendle, Steffen (2012). "Factorization Machines with libFM". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 3.3, 57:1–57:22. DOI: 10.1145/2168752.2168771.