Pearson

IRT and beyond

What to do when you want to customise a model but a package doesn’t let you do that?

Krzysztof Jędrzejewski
Advanced Computing & Data Science Lab

eRum, 15 May 2018

Image by: Lauren Rowling

Code examples and slides

github.com/kjedrzejewski/eRum2018

IRT

- *Item Response Theory*
- Used in psychometrics to estimate the **difficulty of a test question** (and a learner's skill level)
- Can also be used in other areas, e.g. to assess **ad clickability**

1PL IRT model

- 1-parameter logistic (1PL) is the most basic IRT model
- **Assumption:** the probability of answering a test question correctly depends only on the difference between a student's skill and that question's difficulty
- **Observed data:**
  - which question was answered?
  - by which student?
  - was the answer correct or incorrect?

Many ways to estimate parameter values with R

- Using a dedicated IRT package, e.g. *TAM*
- As random effects in logistic regression model, e.g. with *lme4*
- Using gradient descent, e.g. with *TensorFlow*
- Using probabilistic programming, e.g. with *stan* or *greta*

TensorFlow

stan

greta

Using a dedicated IRT package

+ We just need to convert the data to the expected format and **call a function**
+ Usually **the fastest way** to estimate model parameters

- Such packages almost always support only most popular models
- **Doesn’t let us to estimate a custom model parameters**

Example packages: *TAM*, *eRm*, *mirt*

Example code: github.com/kjedrzejewski/eRum2018/blob/master/1pl_irt.R

http://www.edmeasurementsurveys.com/TAM/Tutorials/

Using logistic regression with random effects

Question difficulties and skill levels are random effects related to questions and students

+ Allows us to add **additional variables and parameters** to the model

- The model needs to remain a **linear combination** of observed variables

Example packages: *lme4*

Example code: github.com/kjedrzejewski/eRum2018/blob/master/1pl_me.R

https://github.com/lme4/lme4

Using gradient descent (e.g. with *TensorFlow*)

Maximum Likelihood Estimation of model parameters using cross entropy and gradient descent based optimisers

+ Allows us to have **non-linear components** in the model
+ Can use **GPU to speed up** computations

- We need to write **a lot of code** to describe dependencies between data and model parameters, and to establish the optimisation process
- We need to create our **own stop condition**

TensorFlow

NVIDIA CUDA

Example packages: *tensorflow*

Example code: github.com/kjedrzejewski/eRum2018/blob/master/1pl_tf.R

https://tensorflow.rstudio.com

Using probabilistic programming (with *stan*)

+ Provides credible intervals of estimated model parameters, which gives us **information about the precision of our estimates**

- Model needs to be expressed in the ***stan* language**

- Sampling is **time-consuming**, esp. for big datasets

Example packages: *rstan*

Example code: github.com/kjedrzejewski/eRum2018/blob/master/1pl_stan.R

http://mc-stan.org/users/interfaces/rstan

Using probabilistic programming (with *greta*)

- \+ Also gives us **information on the precision** of our estimates (like *stan*)
- \+ We define the model using **native R syntax** (unlike *stan*)
- \+ It's built on top of TensorFlow, so it **can leverage GPU** for computation

- \- Sampling is still **time-consuming**

Example packages: *greta*

Example code: https://github.com/kjedrzejewski/eRum2018/blob/master/1pl_greta.R

https://greta-dev.github.io/greta/

Benchmark, 1PL, small sample

100 questions, 1000 people => 100 000 observations

	Macbook Pro (CPU-only calculations)
TAM	0.9 s
lme4	24.3 s
tensorflow	4.5 min.
greta	18.9 min.
stan	32.2 min.

https://github.com/kjedrzejewski/eRum2018/blob/master/tests_1pl.R

Benchmark, 1PL, small sample

100 questions, 1000 people => 100 000 observations

	Macbook Pro (CPU-only calculations)	AWS p3.2xlarge nVidia Tesla V100	GPU speed-up
TAM	0.9 s		
lme4	24.3 s		
tensorflow	4.5 min.	1.7 min.	**~2.65x**
greta	18.9 min.	11.9 min.	**~1.59x**
stan	32.2 min.		

https://github.com/kjedrzejewski/eRum2018/blob/master/tests_1pl.R

Benchmark, 1PL, large sample

500 questions, **5000** people => **2 500 000** observations

	Macbook Pro (CPU-only calculations)	AWS p3.2xlarge nVidia Tesla V100	GPU speed-up
TAM	47.3 s		
lme4	30.2 min.		
tensorflow	42.4 min.	3.8 min.	**~11.16x**
greta	5.8 h	39.4 min.	**~8.83x**
stan	too long :(		

https://github.com/kjedrzejewski/eRum2018/blob/master/tests_1pl.R

Takeaways

- TensorFlow may be used for other tasks than deep learning
- GPU may be used to speed up parameter estimation of a large group of models
- For large samples, it may be faster to estimate parameters of a linear model using TensorFlow with GPU, than using specialized regression libraries
- Speed-up offered by GPU increases with data size

ALWAYS LEARNING

ioki.pl/category/data-science/