Fast Item Response Theory (IRT) Analysis by using GPUs

Lei Chen
lei.chen@liulishuo.com
Liulishuo Silicon Valley AI Lab
Outline

• A brief introduction of Item Response Theory (IRT)

• Edward, a new probabilistic programming (PP) toolkit

• An experiment of using Edward to do IRT model estimation on both CPU and GPU computing platforms

• Summary
A concise introduction of adaptive learning

- What's up with adaptive learning

"Adaptive learning is an education technology that can respond to a student's interactions in real-time by automatically providing the student with individual support."
Adaptive learning is hot in the eduTech market

• Increasing demands

• *Districts’ spending on adaptive learning products has grown threefold between 2013 and 2016*, according to a new analysis. EdWeek market brief 7/14/2017

• Increasing suppliers
Precisely knowing students' ability levels is important

- Adaptive learning needs correct inputs about students’ ability levels, which are latent.

- Assessment are developed for inferring latent abilities.

- For a Yes/No question, the probability a student provides a correct answer \( p(X=1) \) depends on
  - his/her latent ability (theta)
  - Also other related factors, e.g., item’s difficulty, making a lucky guess, carelessness …
Item Response Theory (IRT)

- IRT provides a principled statistical method to quantify these factors and has been widely used to build up modern assessment industry

- A widely used 2 parameter logistic model (2-PL)

\[ P(X = 1 | \theta, a, b) = \frac{e^{a(\theta - b)}}{1 + e^{a(\theta - b)}} \]
IRT with fewer or more parameters

• 1-PL
  • Only having b, assume all items share same a

• 3-PL
  • c for random guessing

• 4-PL
  • d for inattention

\[ P(X = 1|\theta, a, b, c) = c + (1 - c) \frac{e^{a(\theta-b)}}{1+e^{a(\theta-b)}} \]

\[ P(X = 1|\theta, a, b, c, d) = c + (d - c) \frac{e^{a(\theta-b)}}{1+e^{a(\theta-b)}} \]
IRT’s wide usages
IRT’s wide usages

• More precise description of item performance
IRT’s wide usages

- More precise description of item performance
- More precise scoring
IRT’s wide usages

- More precise description of item performance
- More precise scoring
- More powerful test assembly
IRT’s wide usages

• More precise description of item performance

• More precise scoring

• More powerful test assembly

• Supporting advanced linking & equating to make standard tests be possible
IRT’s wide usages

• More precise description of item performance

• More precise scoring

• More powerful test assembly

• Supporting advanced linking & equating to make standard tests be possible

• Supporting adaptive testing by placing examinees and items on the same scale
Concrete examples

• “Item response theory and computerized adaptive testing” presentation made for a hands-on workshop by Rust, Cek, Sun, and Kosinski from University of Cambridge The Psychometrics Center

• Very nice animations to explain IRT, how to use IRT to score, and CAT.
Item Response Function
Binary items

Parameters:
- Measured concept (theta)

Models:

Probability of getting item right

1

Measured concept (theta)
**Item Response Function**

**Binary items**

**Parameters:**
- Difficulty

**Models:**
- 1 Parameter

---

Probability of getting item right

---

Measured concept (theta)
Item Response Function

Binary items

Parameters:
- Difficulty
- Discrimination

Models:
- 1 Parameter
- 2 Parameter

Measured concept (theta)

Probability of getting item right

Discrimination (slope)

Difficulty
**Item Response Function**

**Binary items**

**Parameters:**
- Difficulty
- Discrimination
- Guessing

**Models:**
- 1 Parameter
- 2 Parameter
- 3 Parameter

- Probability of getting item right
- Measured concept (theta)

- Discrimination (slope)
- Difficulty
- Guessing
Item Response Function
Binary items

- Parameters:
  - Difficulty
  - Discrimination
  - Guessing
  - Inattention

- Models:
  - 1 Parameter
  - 2 Parameter
  - 3 Parameter
  - 4 Parameter

Probability of getting item right vs. Measured concept (theta)
Item Response Function

Binary items

Parameters:
- Difficulty
- Discrimination
- Guessing
- Inattention

Models:
- 1 Parameter
- 2 Parameter
- 3 Parameter
- 4 Parameter
- unfolding

Measured concept (theta)

Probability of getting item right

Discrimination (slope)

Difficulty

Guessing

Inattention

1
Scoring

Test:

![Graph showing probability against theta values]
Scoring

Test:
1. Normal distribution
Scoring

Test:
1. Normal distribution
2. q1 - Correct
Scoring

Test:
1. Normal distribution
2. q1 - Correct
Scoring

Test:
1. Normal distribution
2. q1 - Correct
3. q2 - Correct
Scoring

Test:
1. Normal distribution
2. q1 - Correct
3. q2 - Correct

Most likely score
Scoring

Test:
1. Normal distribution
2. q1 - Correct
3. q2 - Correct
4. q3 - Incorrect
Scoring

Test:
1. Normal distribution
2. q1 - Correct
3. q2 - Correct
4. q3 - Incorrect

Most likely score
Scoring

Test:
1. Normal distribution
2. q1 - Correct
3. q2 - Correct
4. q3 - Incorrect

Most likely score
Scoring

Test:
1. Normal distribution
2. q1 - Correct
3. q2 - Correct
4. q3 - Incorrect

Most likely score
Computer Adaptive Testing

- Standard tests
  - Containing fixed number of questions
  - Some are too simple and some are too difficult for a specific test-taker

- CAT
  - Items can be tailored
  - Save time/money
  - Measure test-taker’s ability more accurately
Example of CAT

Start the test:
Example of CAT

Start the test:
1. Ask first question, e.g. of medium difficulty
Example of CAT

Start the test:
1. Ask first question, e.g. of medium difficulty
2. Correct!
Example of CAT

Start the test:
1. Ask first question, e.g. of medium difficulty
2. Correct!
3. Score it

Probability

Theta

Normal distribution

Most likely score

13
Example of CAT

Start the test:
1. Ask first question, e.g. of medium difficulty
2. Correct!
3. Score it

Most likely score
Example of CAT

Start the test:
1. Ask first question, e.g. of medium difficulty
2. Correct!
3. Score it
4. Select next item with a difficulty around the most likely score (or with the max information)
Example of CAT

Start the test:
1. Ask first question, e.g. of medium difficulty
2. Correct!
3. Score it
4. Select next item with a difficulty around the most likely score (or with the max information)
5. And so on…. Until the stopping rule is reached
IRT model estimation

- Mostly used Marginal Maximum Likelihood (MMLE)
  - Finding the marginal distribution of the item parameters by **integrating over theta**
  - Estimate item parameters by MLE
  - Obtain theta by MLE based on estimated item parameters
  - For a more efficient estimation, use EM
- Other ways
  - Joint Maximum Likelihood (JML)
Bayesian solution

• Issues with MLE
  • Depends on distribution of data
  • Estimation is not accurate when samples are small-sized
  • Hard to handle ability distribution is not normal

• Bayesian solutions consider theta priors
MCMC

- Markov chain Monte Carlo (MCMC) used for Bayesian estimation

- Ultimate goal is approximate $p(\text{parameters}|\text{data})$ by sampling many data points from the posterior probability

- Hamiltonian MC is good at dealing with high-dimensional parameter spaces. HMC utilizes the geometry of the important regions of the posterior for making better proposals.
Variational Inference

• To approximate intractable distribution by using a family of distributions and finding the member of this family that can minimizes divergence to the true posterior

• By approximating the posterior with a simpler function, leading to faster estimation

• Kullback–Leibler (K-L) divergence was frequently used to measure two distributions’ closeness
Previous efforts of using GPUs for fast estimation


• C programming using CUDA

• Challenges
  • C/CUDA is not familiar to many data scientists
  • Low-level implementation
Edward

• A library for probabilistic modeling, inference, and criticism

• Developed by Dustin Tran and others at Columbia University

• Named in honor of innovative statistician George Edward Pelham Box.

• Created in 2016 but has attracted many users for doing probabilistic programming
Attractions of Edward

• Rich optimization/inference methods

• Make it very convenient for many users who are interested in trying PP but don’t want to be swamped by many math and statistics details

• Developed as a higher level abstraction on Tensorflow

• GPU running is enabled automatically by using Tensorflow
Edward uses Box’ loop

- a) Build a model (forward direction)
- b) Use observed data to infer posterior (backward direction)
- c) Criticize the model and revise (=> a)
A concrete example

• An example from Torsten Scholak, Diego Maniloff “Intro to Bayesian Machine Learning with PyMC3 and Edward” at PyCon 2017

• From coin toss sequence [H,T,H,T,H,H,T,H,…] to estimate prob(H)

• Model

```python
>>> pheads = Uniform(low=0.0, high=1.0)  # params: fairness of the coin.
>>> c = Bernoulli(probs=pheads, sample_shape=100)  # data: coin tosses.
```
Using Edward to infer

Approximate inference in the **PP language** Edward:

```python
# BACKWARD MODEL
q_pheads = ??

# INFERENCE
inference = ed.Inference(latent_vars={pheads: q_pheads}, data={c: c_train})

# define posterior vars
# bind latent to posterior vars
# bind random vars to data
```

Running inference:

```python
>>> inference.run()
```
Experiment

- Generate simulated test data
  - Binary answers from 2,000 students working on 250 test questions; need jointly estimate 2,000 + 250 ability and item parameters for a 1-PL model
  - Generated true ability (theta) and item difficulty (threshold) from two normal distributions
  - Based on examinee's ability and item difficulty, generate answer vector

```python
# DATA
nsubj = 2000
nitem = 250
trait_true = np.random.normal(size=[nsubj, 1])
thresh_true = np.random.normal(size=[1, nitem])
X_data = np.random.binomial(1, expit(trait_true - thresh_true))
```
Experiment : model

• Specify a generative model

• Treat priors of both trait and threshold to be normal

• Logit transformation is \( \ln(s/1-s); s = \exp(\text{logit})/1+\exp(\text{logit}) \), which is 1-PL IRT model where logit = trait - threshold

```python
# MODEL
trait = Normal(loc=tf.zeros([nsubj, 1]), scale=tf.ones([nsubj, 1]))
thresh = Normal(loc=tf.zeros([1, nitem]), scale=tf.ones([1, nitem]))
X = Bernoulli(logits=trait - thresh)
```
Experiment: inference

• Inference using HMC

• Posterior are from the samples generated by running HMC

```python
# INference
T = 5000  # number of posterior samples
go Trait = Empirical(params=tf.Variable(tf.zeros([T, nsubj, 1])))
go Thresh = Empirical(params=tf.Variable(tf.zeros([T, 1, nitem])))

inference = ed.HMC({trait: q_trait, thresh: q_thresh}, data={X: X_data})
inference.run(step_size=0.1)
```
Experiment: inference

- Inference using Variational Inference
  - For both trait and threshold, use normal distribution family with variational loc and scale
  - `ed.KLqp` do the backward inference to determine loc and scale

```python
# INFERENCEn
q_trait = Normal(
    loc=tf.Variable(tf.random_normal([nsubj, 1])),
    scale=tf.nn.softplus(tf.Variable(tf.random_normal([nsubj, 1]))))
q_thresh = Normal(
    loc=tf.Variable(tf.random_normal([1, nitem])),
    scale=tf.nn.softplus(tf.Variable(tf.random_normal([1, nitem]))))

inference = ed.KLqp({trait: q_trait, thresh: q_thresh}, data={X: X_data})
inference.run(n_iter=2500, n_samples=10)
```
Experiment: setups

• Metrics
  • Speed: running time in seconds
  • Estimation accuracy: MSE between true parameters and the estimated parameters

• Hardware
  • A game PC using Ubuntu Linux as OS
  • CPU: Intel Xeon E5-1660 v3
  • GPU: NVIDIA Titan X
Experiment: result

<table>
<thead>
<tr>
<th>Inference/Platform</th>
<th>Running time (Sec)</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMC/CPU</td>
<td>893</td>
<td></td>
</tr>
<tr>
<td>HMC/GPU</td>
<td>222</td>
<td>0.900</td>
</tr>
<tr>
<td>VB/CPU</td>
<td>116</td>
<td></td>
</tr>
<tr>
<td>VB/GPU</td>
<td>29</td>
<td>0.023</td>
</tr>
</tbody>
</table>

- For the two inference methods, GPU running is about 4X faster than CPU running
- Compared to MC, VB is much more time efficient
- In our simulated data, VB shows more accurate parameter estimation
Summary

• IRT models are acting as cornerstone for many educational applications

• Estimating a large of model parameters (on students’ ability levels and items) can be time consuming

• Edward, a probabilistic programming toolkit, provides a convenient way for doing IRT parameter estimation using Bayesian methods, and it enables fast GPU computations
Useful resources


- http://tscholak.github.io/assets/PyConEdward/#!/


- http://edwardlib.org/