Prerequisite Skill Structures in ASSISTments

Michel Desmarais Sameer Bhatnagar

August 4, 2015

Abstract

At the 2015 Artificial Intelligence in Education conference, Seth Adjei and Neil Heffernan presented their work on scrutinizing expert-defined prerequisite skill graphs. Using randomized controlled trials in PLACEments, the computer-adaptive-testing feature in the ASSISTments learning environment, they were able to identify some prerequisite skill arcs that were not supported by data.
This proposal outlines a technique that could be used to achieve the goal. We will start with a basic introduction to Partial Order Knowledge Structures (POKS), then try to mathematically formalize what a prerequisite skill might mean in this framework. We will then show a basic example as "proof of concept".

1 POKS

Partial Order Knowledge Structures (POKS) determine which items in a test are prerequisite to others. POKS are derived from the theory of Knowledge Spaces, where such prerequisite relationships are written as $A \to B$, which means if a student got item A correct, they likely will get item B also correct; said differently, item B is a prerequisite of item A.

1.1 Working Example

Let us look at a classical dataset from Tatsuoka, which is well studied in de la Torre 2009.

1.1.1 POKS Network Induction

Here is the raw test data, where each of the 536 rows represents a student, and each column is their success/failure on one of the 15 items of a test on subtraction of fractions.

```
fraction <- data.fraction1
head(fraction$data)
```

```
##   T01 T02 T03 T04 T05 T06 T07 T08 T09 T10 T11 T12 T13 T14
## 1   0   1   0   1   0   1   1   1   1   1   1   0   1   1
## 2   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 3   1   1   1   1   0   0   0   0   1   1   1   0   0   0
## 4   1   1   1   0   0   1   1   1   1   0   1   0   1   0
## 5   0   1   1   0   0   0   0   1   0   0   0   0   0   0
## 6   0   0   1   0   1   0   0   0   1   0   0   0   0   0
##   T15
## 1   1
## 2   1
## 3   0
## 4   1
## 5   0
## 6   0
```

```
tail(fraction$data)
```

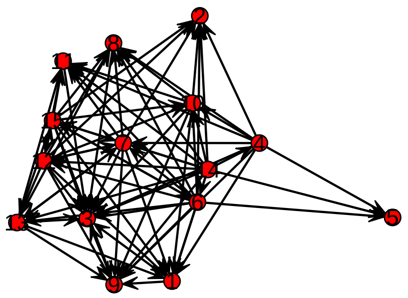
```
##     T01 T02 T03 T04 T05 T06 T07 T08 T09 T10 T11 T12 T13 T14
## 531   0   0   0   0   0   0   1   0   0   0   0   0   0   0
## 532   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## 533   1   1   1   1   0   1   1   1   1   0   1   1   1   0
## 534   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## 535   1   1   1   1   1   1   1   1   1   1   1   0   1   0
## 536   1   1   1   0   0   0   0   1   1   1   1   0   0   0
##     T15
## 531   0
## 532   0
## 533   0
## 534   0
## 535   0
## 536   0
```

If we run the POKS code on this data, we get the following adjacency matrix.

```
source("lib-poks-3.R")
ks <- ks.init(as.matrix(fraction$data), p.min = 0.8)
ks$m
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    0    0    1    0    0    0    0    0    1     0
##  [2,]    0    0    0    0    0    0    0    0    0     0
##  [3,]    0    0    0    0    0    0    0    0    0     0
##  [4,]    1    1    1    0    1    0    1    1    1     1
##  [5,]    0    0    0    0    0    0    0    0    0     0
##  [6,]    1    1    1    0    1    0    1    1    1     1
##  [7,]    1    1    1    0    0    0    0    1    1     0
##  [8,]    0    0    1    0    0    0    0    0    0     0
##  [9,]    0    0    1    0    0    0    0    0    0     0
## [10,]    1    1    1    0    0    0    0    1    1     0
## [11,]    0    0    1    0    0    0    0    0    0     0
## [12,]    1    1    1    0    0    0    1    1    1     1
## [13,]    1    0    1    0    0    0    1    1    1     0
## [14,]    1    1    1    1    1    1    1    1    1     1
## [15,]    1    1    1    0    0    1    1    1    1     1
##       [,11] [,12] [,13] [,14] [,15]
##  [1,]     0     0     0     0     0
##  [2,]     0     0     0     0     0
##  [3,]     0     0     0     0     0
##  [4,]     1     0     0     0     0
##  [5,]     0     0     0     0     0
##  [6,]     1     1     1     0     1
##  [7,]     1     1     1     0     0
##  [8,]     0     0     0     0     0
##  [9,]     0     0     0     0     0
## [10,]     1     0     0     0     0
## [11,]     0     0     0     0     0
## [12,]     1     0     1     0     1
## [13,]     1     0     0     0     0
## [14,]     1     1     1     0     1
## [15,]     1     1     1     0     0
```

To read this adjacency matrix, note that the row points to the column. For example, item 1 should have links pointing to items 3 and 9. (This means if a student gets item 1 correct, they should get items 3 and 9 correct as well) This structure can be visualized as follows:



What stands out is item five, which has no prerequisites (no edges leaving node 5). Let us use this as a place to work out some numbers to demonstrate how the POKS structure is induced. We see that mastery of item 4 implies mastery of item 5 from the table and the graph. Let us check the contingency table for these two items.

```
item5 <- fraction$data$T05
item4 <- fraction$data$T04
table(item4, item5)
```

```
##      item5
## item4   0   1
##     0 149 173
##     1  40 174
```

Now we can calculate $P(item_5|item_4)$ which should be greater than some threshold, as should $P(\neg item_4|\neg item_5)$ (threshold default is 0.5).

We must also ensure that the distributions over the two items are actually interacting with one another using a chi-square test.

This is repeated pairwise for all the items until a item-item structure is built, with directed edges indicating prerequisite relations.

2 Prerequisite Skills

It is important to recognize that to go from item-item structures, to skill-skill structure, we need an item-skill mapping. This has been coined the Q-matrix, where each item may require one or more skills. Here is the expert-defined q-matrix from our dataset:

```
fraction$q.matrix

##     QT1 QT2 QT3 QT4 QT5
## T01   1   0   0   0   0
## T02   1   1   1   1   0
## T03   1   0   0   0   0
## T04   1   1   1   1   1
## T05   0   0   1   0   0
## T06   1   1   1   1   0
## T07   1   1   1   1   0
## T08   1   1   0   0   0
## T09   1   0   1   0   0
## T10   1   0   1   1   1
## T11   1   0   1   0   0
## T12   1   0   1   1   0
## T13   1   1   1   1   0
## T14   1   1   1   1   1
## T15   1   1   1   1   0
```

The five skills (columns) are

QT1 performing basic fraction-subtraction operation

QT2 simplifying/reducing

QT3 separating whole number from fraction

QT4 borrowing one from whole number to fraction

QT5 converting whole

The proposal below will try to derive the prerequisite graph of these skills, and answer the question: which skills should be learned before the others?

2.1 Deriving Prerequisite skills - proposal

Multiplication of the student-response matrix, with the Q-matrix, gives what is called the skill-mastery matrix, wherein we can see which students have mastered which skills.

```
skill.succ <- as.matrix(fraction$data) %*% as.matrix(fraction$q.matrix)
skill.fail <- as.matrix(1 - fraction$data) %*% as.matrix(fraction$q.matrix)
```

These two lines of code separate out the calculation of which students succeedded in acquiring the skill, and which students failed in doing so. (This helps deal with missing values, as there might be some students who didn't get tested on a particular skill).

This is a projection of the student test data onto the skills: the higher the value for a student, the greater our confidence that they mastered that skill. This can be normalized by the maximum possible score the student could have had for any particular skill, which gives a skill-mastery-probability matrix:

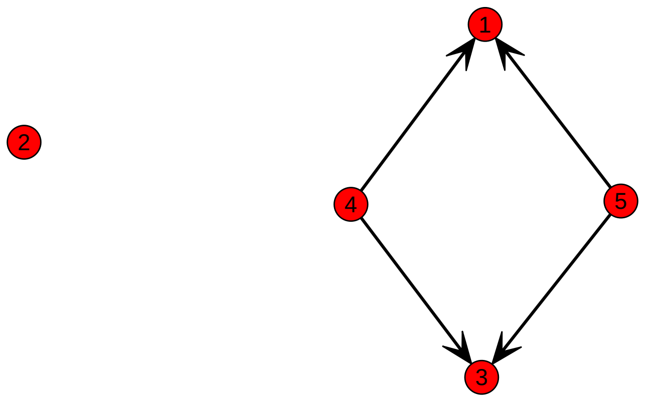
```
skill.mast.data <- skill.succ/(skill.succ + skill.fail)

head(skill.mast.data)

##       QT1  QT2   QT3  QT4  QT5
## [1,] 0.79 1.00 0.833 0.89 1.00
## [2,] 1.00 1.00 1.000 1.00 1.00
## [3,] 0.50 0.25 0.417 0.33 0.67
## [4,] 0.71 0.75 0.583 0.56 0.00
## [5,] 0.21 0.25 0.083 0.11 0.00
## [6,] 0.14 0.00 0.167 0.00 0.00

tail(skill.mast.data)

##          QT1  QT2   QT3  QT4  QT5
## [531,] 0.071 0.12 0.083 0.11 0.00
## [532,] 0.000 0.00 0.000 0.00 0.00
## [533,] 0.786 0.75 0.667 0.67 0.33
## [534,] 0.000 0.00 0.000 0.00 0.00
## [535,] 0.786 0.75 0.750 0.67 0.67
## [536,] 0.500 0.25 0.333 0.22 0.33
```

Now if we consider each skill as an item, we get something that looks a lot like another item-response matrix (just like the raw test data we started with). We can now run the same POKS code as before, but on this new skill mastery matrix: the difference is that now, instead of determining which *items* are prerequisites of each other, we will get relationships describing mastery of which *skills* imply other *skills*.

```
ks.skills <- ks.init(as.matrix(skill.mast.data), p.min = 0.9)
ks.skills$m
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    0    0    0
## [2,]    0    0    0    0    0
## [3,]    0    0    0    0    0
## [4,]    1    0    1    0    0
## [5,]    1    0    1    0    0
```

Once again, this is an adjacency matrix, where a 1 represents that the item of that row can should have a link directed towards the item of that column. This can be visualized as follows (remembering that each node now represents a skill)



The threshold probability pmin was raised to 0.9 in deriving this skill graph. Comparing this graph to the original list of skills, we believe there is sufficient face-value to this approach to try further validation (skill QT2 - "simplifying/reducing" is seperate not related to any of the others: one could argue that this skill has less to do with fractions, and more to do with really knowing your multiplication tables).

3 Validation

Once a prerequisite skill structure is determined, it can be validated by seeing if running a simulation of a computer adaptive test, knowledge of the skill structure would improve predictions of performance over POKS alone. This would require a slightly modified model. Another approach would be to generate synthetic data and see if this method could recover the prequisite skill structure that generated the data.