

A Framework for Comparing Models for Adaptive Testing

Jill-Jênn Vie

February 19, 2016

Models for Adaptive Testing

Framework, Experiment, Results

NEW! Adaptive Submodularity

Models for Adaptive Testing

Computerized Adaptive Testing (CAT)

Asking the right questions to the right people.

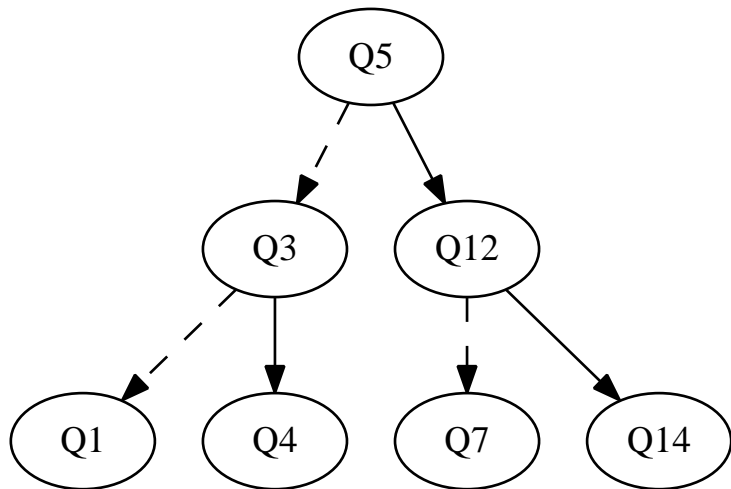


Figure 1: An adaptive test.

First of all

Assumptions

- ▶ Dichotomous items (either answered correctly or incorrectly)
- ▶ We do not care about item exposure (yet)

Goals

- ▶ We want to ask **as few questions as possible** in a test.
- ▶ Lots of different models. Which ones fit our data the most?

1. Rasch Model (catR)

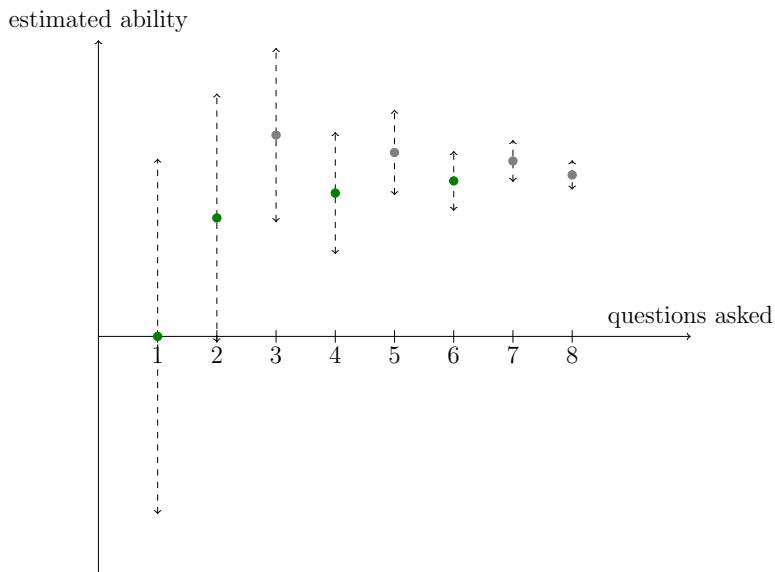


Figure 2: Example of CAT using the Rasch model.

An example of CAT simulated with catR

We ask question 42 to the examinee.

Correct!

We ask question 48 to the examinee.

Correct!

We ask question 82 to the examinee.

Incorrect.

We ask question 53 to the examinee.

Correct!

We ask question 78 to the examinee.

Incorrect.

We ask question 56 to the examinee.

Correct!

We ask question 76 to the examinee.

Incorrect.

We ask question 58 to the examinee.

Incorrect.

RPy2: R bindings for Python

```
from rpy2.robjects import r
r('library(catR)')
r('one <- sample(1, 100, T)')
r('itembank <- cbind(one, c(1:100)/100, 1 - one, one)')
pattern = [1, 1, 0, 1, 0, 1, 0, 0]
ql = [42]
for t in range(len(pattern)):
    print('We ask question %d to the examinee.' % ql[-1])
    print('Correct!' if pattern[t] else 'Incorrect.')
    questions = ','.join(map(str, ql))
    answers = ','.join(map(str, pattern[:t + 1]))
    r('theta <- thetaEst(matrix(itembank[c(%s),]',
        'nrow=%d), c(%s))' % (questions, t + 1, answers))
    q = r('nextItem(itembank, NULL, theta, x = c(%s),'
        'out = c(%s))$item' % (answers, questions))[0]
    ql.append(q)
```


2. Cognitive Diagnosis (CDM) aka Rule-Space Method

Mapping knowledge components (KC) to items in order to diagnose **misconceptions**.

2. Cognitive Diagnosis (CDM) aka Rule-Space Method

Mapping knowledge components (KC) to items in order to diagnose **misconceptions**.

Example

- ▶ Solving Item 1 requires mastering KC 1 and 2 (or guessing)
- ▶ Solving Item 2 requires mastering KC 3
- ▶ ...

At the end of the test, we can provide a **feedback** to the examinee.

Example: DINA model aka q-matrix

DINA: Deterministic Input, Noisy “And” gate.

$$\frac{2}{3} + \frac{5}{6} = ?$$

KC 1 Put at same denominator

KC 2 Add two fractions of same denominator

$$\frac{1}{2} \times \frac{3}{4} = ?$$

KC 3 Multiply two fractions

We can provide **useful feedback** to examinees:

- ▶ “You seem to have KC 2 and KC 3 but not KC 1.”

Example: DINA model aka q-matrix

DINA: Deterministic Input, Noisy “And” gate.

$$\frac{2}{3} + \frac{5}{6} = ?$$

KC 1 Put at same denominator

KC 2 Add two fractions of same denominator

$$\frac{1}{2} \times \frac{3}{4} = ?$$

KC 3 Multiply two fractions

We can provide **useful feedback** to examinees:

- ▶ “You seem to have KC 2 and KC 3 but not KC 1.”

... Sorry, I said **useful**:

- ▶ “You seem to be able to add two fractions of same denominator and multiply two fractions, but not put two fractions at the same denominator.”


Note: You may not find the DINA model on Google

Google

Tous Images Vidéos Actualités Plus ▾ Outils de recherche

Environ 16 700 000 résultats (0,55 secondes)

[Images correspondant à DINA model](#) [Signaler des images inappropriées](#)



[Plus d'images pour DINA model](#)

Model Dina
dinamodelpage.tumblr.com/ ▾ [Traduire cette page](#)
Model from Germany 21 Years My measures: Size: 1,72m Clothing size: 32/34 (XS/S)
Shoe size: 37/38 Measure: 91-65-89.

Figure 3: Another DINA model.

What does a CD-CAT look like?

Cognitive Diagnosis Computerized Adaptive Testing.

Round 1 -> We ask question 9 to the examinee.

It requires KC: [0, 1, 0, 0, 0, 0, 0, 0]

Correct!

Examinee: [0.5, 0.74, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5]

Estimate: 00000101100000000000

Truth: 00011111111101001111

Round 2 -> We ask question 6 to the examinee.

It requires KC: [0, 0, 0, 0, 0, 0, 1, 0]

Correct!

Examinee: [0.5, 0.74, 0.5, 0.5, 0.5, 0.5, 0.91, 0.5]

Estimate: 00000101100101010000

Truth: 00011111111101001111

What does a CD-CAT look like?

Round 4 -> We ask question 2 to the examinee.

It requires KC: [0, 0, 0, 1, 0, 0, 1, 0]

Incorrect.

Examinee: [0.5, 0.74, 0.5, 0.06, 0.5, 0.5, 0.96, 0.87]

1 1 0 3 3 9 3 8 6 3 4 8 0 7 4 7 3 3 1 2

Estimate: 00000101100101010000

Truth: 00011111111101001111

Round 6 -> We ask question 10 to the examinee.

It requires KC: [0, 1, 0, 0, 1, 0, 1, 1]

Correct!

Examinee: [0.5, 0.99, 0.67, 0.06, 0.98, 0.5, 1.0, 0.99]

Estimate: 00010101111101011001

Truth: 00011111111101001111

3. Regression Trees (Yan, Lewis, Stocking)

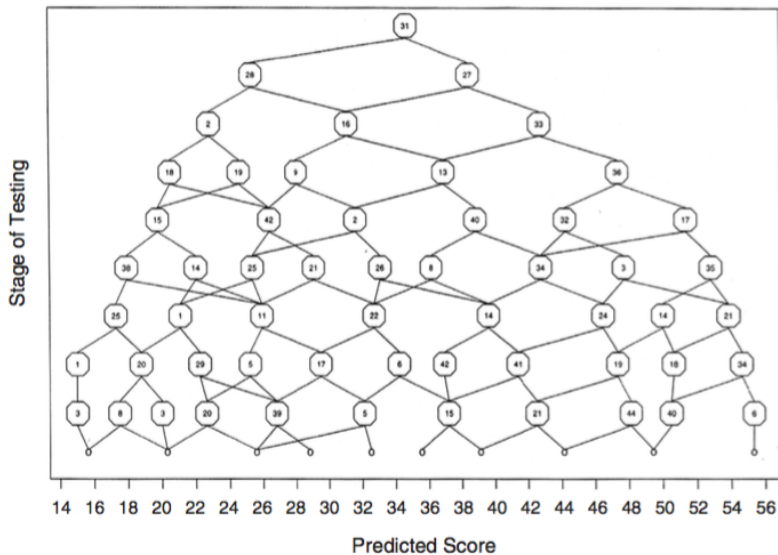


Figure 4: CAT using regression trees.

And many more

Multidimensional Item Response Theory

d latent traits instead of 1

MIRT + q-matrix

Measure one latent model per knowledge component

SPARFA: Sparse factor analysis

No access to full response patterns

Multistage testing

Asking questions k by k instead of one by one

And many more

Multidimensional Item Response Theory

d latent traits instead of 1

MIRT + q-matrix

Measure one latent model per knowledge component

SPARFA: Sparse factor analysis

No access to full response patterns

Multistage testing

Asking questions k by k instead of one by one

But **how to compare them?**

They're all flowcharts! (Or binary decision trees.)

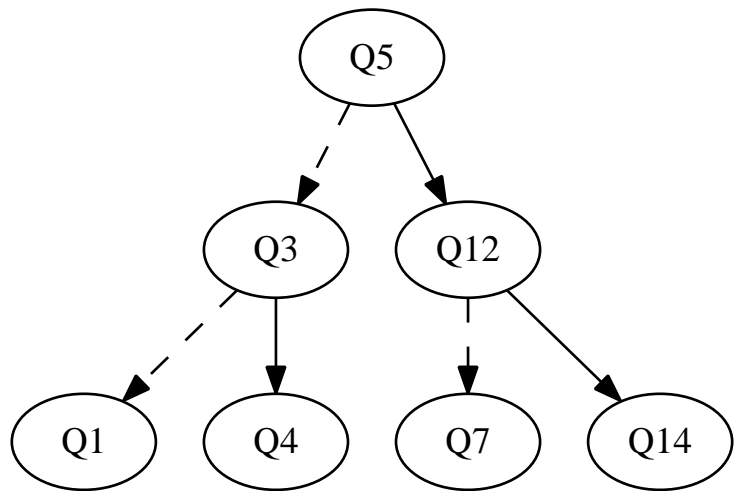
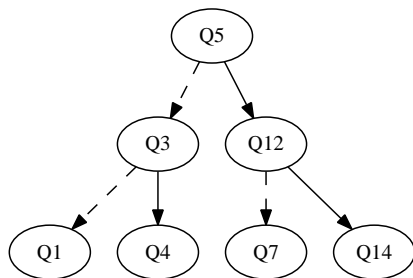


Figure 5: A binary decision tree.

Framework, Experiment, Results

Train/test datasets for both users and questions

- ▶ We train our models using a **train** dataset of student response patterns
- ▶ We evaluate them on models the following way:
 - ▶ We ask questions with the same criterion for all models (MFI)
 - ▶ And keep a validation question set.



Methods needed

- ▶ `training_step` over train dataset
- ▶ `init_test`
- ▶ `next_item` using questions and answers got so far
- ▶ `estimate_parameters` based on the last answer
- ▶ `predict_performance` of the model over the `validation_question_set`

Methods needed

- ▶ `training_step` over train dataset
- ▶ `init_test`
- ▶ `next_item` using questions and answers got so far
- ▶ `estimate_parameters` based on the last answer
- ▶ `predict_performance` of the model over the `validation_question_set`

Example: `mirt.py` calling `mirtCAT` package

```
def next_item(self, replied_so_far, results_so_far):
    next_item_id = mirtCAT.findNextItem(r.CATdesign)[0]
    return next_item_id - 1

def estimate_parameters(self, rep_so_far, res_so_far):
    r('CATdesign <- updateDesign(CATdesign, items=...)')
    r('CATdesign$person$update.thetas(CATdesign$design)')
```

Double cross-validation

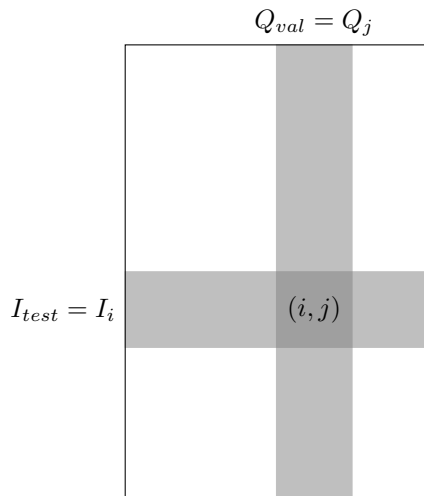


Figure 6: This is **not** a Belgian chocolate box.

Datasets

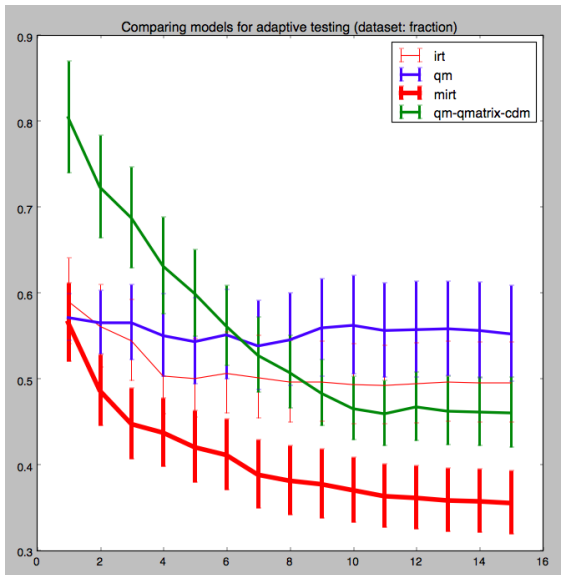
SAT test: 296 students, 40 questions

Multidisciplinary: Mathematics, Biology, World History, French.

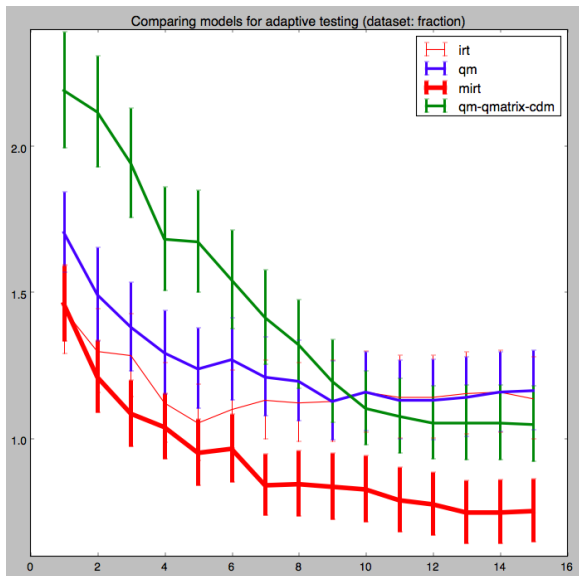
Fraction subtraction test: 536 students, 20 questions

KCs specified (*add fractions of same denominator*, etc.).

Results for the Fraction dataset: mean prediction error (negative log-likelihood)



Results for the Fraction dataset: mean number of questions predicted correctly



Discussion

Remarks

- ▶ After only 4 questions over 15, MIRT + q-matrix can predict correctly 4 out of 5
- ▶ Q-matrix (DINA) alone takes a long time to converge because first questions measure single KC
- ▶ In the early stages, Rasch Model performs well compared to 8-dim MIRT

Future work

- ▶ How to compare a flowchart with the optimal flowchart?
- ▶ A q-matrix is expensive to build. How helpful is it?
- ▶ How to compare CAT with MST?

NEW! Adaptive Submodularity

Adaptive Submodularity (Golovin and Krause, 2010)

Automated diagnosis

Suppose we have different hypotheses about the state of a patient, and can run medical tests to rule out inconsistent hypotheses. The goal is to adaptively choose tests to infer the state of the patient as quickly as possible.

This can be seen as a **Stochastic Set Cover** problem: we want to cover as many fake hypotheses as possible.

Adaptive submodular function

~ convexity over discrete domains (= subsets of items).

If the function to maximize (= information) has a certain property (*monotonic submodular*), a **greedy flowchart** builds a satisfying set: $(1 - 1/e) \simeq 67\%$ of the optimal flowchart in average.

Example 1: Vitamin C

Orange	Apple	Mango	Banana	Lemon
51 mg	8 mg	122 mg	10 mg	31 mg

- ▶ We want to find the **subset of k fruits** having biggest vitamin C.
- ▶ But vitamin C is an additive function:
$$\text{vitamin}(\{banana, apple\}) = \text{vitamin}(\{banana\}) + \text{vitamin}(\{apple\})$$
- ▶ Thus, taking the **best fruit at each step** is optimal.

What can be done with more generic functions?

$$f : 2^{E \times O} \rightarrow \mathbb{R}_{\geq 0}$$

is a function over subsets of pairs (*item*, *outcome*).

Monotonicity

The marginal benefit of selecting an item is always nonnegative

Submodularity

Selecting an item later never increases its marginal benefit

Our application

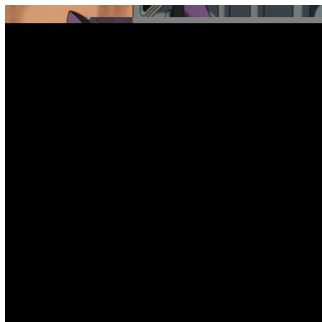
Any information function is supposed to be monotonic.

Submodularity is a stronger assumption: one can discuss.

Example 2: Maximizing Fisher information

- ▶ We want to compare catR's flowchart of depth k using MFI criterion with the optimal flowchart (achieving maximal Fisher information at the leaves).
- ▶ If the Fisher information function is monotone submodular,
- ▶ catR's greedy algorithm taking best item for MFI criterion performs in average $(1 - 1/e) \simeq 67\%$ as good as the best adaptive test. Good job David!

Thanks for listening!



Jill-Jênn Vie

jiji.cat

<http://github.com/jilljenn>

jjv@lri.fr

If you're interested in adapting a script for your uses, please drop me an issue :)