「MeLQS 準拠の多肢選択問題の STACK における実装について」

吉冨 賢太郎

大阪府立大学

2021.11.13

解答過程解析を中心とする理数系 e ラーニングの
分析・設計・運用に関する総合研究
第 1 回シンポジウム
ー数式自動採点システムの活用事例ー

オンラインテスト教材の目的

オンライン学習 ・・・ 自宅での学習

オンラインテスト

・・・ CBT，自宅や端末室でのテスト

動画と並ぶ自習教材の要

自習教材設計のポイント

1 常に考えさせる←集中力の維持

2 途中でつまづか(せ)ない←継続性

1. 常に考えさせる工夫🤔

- 解説動画では例, 例題や問掛けを潤沢に
- 例題の実践問題をオンライン問題で提供
- 動画は重い➠問題メインでもいいかも

2. 途中でつまづかせないための問題設計

a. 負荷を大きくしない

- 他にも学生にはやることがある (教員もだけど)🙄
- 計算がむっちゃ大変なのは勘弁...(猛省 😅)
- 入力が大変なのも勘弁😕➠ 多肢選択の混用で回避

b. フィードバックで気付きを誘導

適切なフィードバック (後述) が重要

c. バグのない問題

形成的評価利用 ➠ 学生を人柱にしない配慮が必要😾

バグのない問題開発は神経使う ➠ 今後も使えるので👍

良いフィードバックとは？

★「正解か間違いか」のみを表示

➠ 単なる計算練習ならOK

➠概念理解を共なう場合はNG

★「何が原因で間違えたか？」「何をすべきか？」を表示

アンケートに多い意見

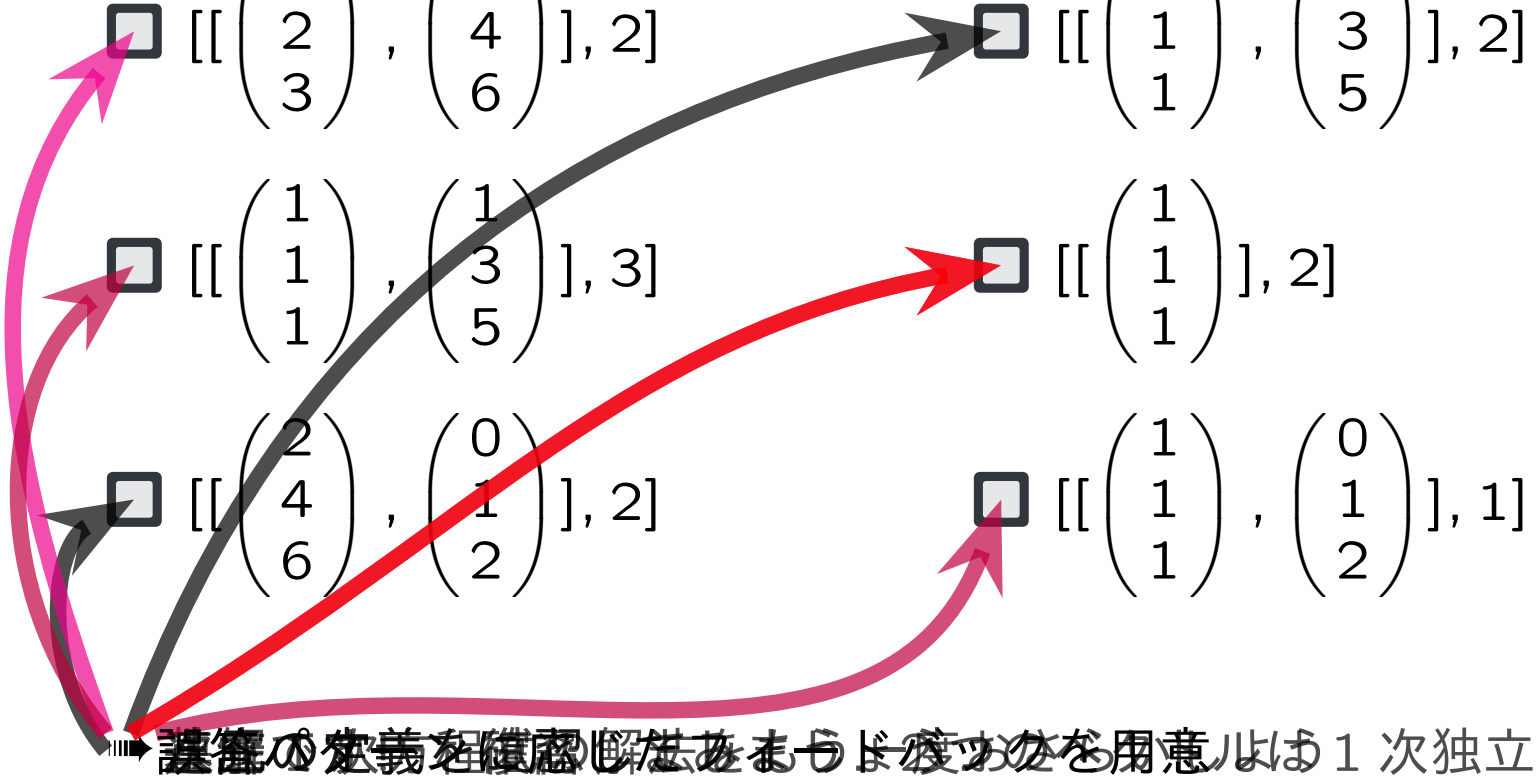
「どこが違うかわからないので説明して欲しい」

「問題の解説動画が欲しい」etc.

例

問題 行列 A の列ベクトルが... である行列を被約階段行列に変形すると... となるとき，A の列空間の基底と次元の組合せとして正しいものをすべて選びなさい．

- $[[\begin{pmatrix}1\\2\\3\end{pmatrix}, \begin{pmatrix}2\\4\\6\end{pmatrix}], 2]$
- $[[\begin{pmatrix}1\\1\\1\end{pmatrix}, \begin{pmatrix}1\\3\\5\end{pmatrix}], 3]$
- $[[\begin{pmatrix}2\\4\\6\end{pmatrix}, \begin{pmatrix}0\\1\\2\end{pmatrix}], 2]$
- $[[\begin{pmatrix}1\\1\\1\end{pmatrix}, \begin{pmatrix}1\\3\\5\end{pmatrix}], 2]$
- $[[\begin{pmatrix}1\\1\\1\end{pmatrix}], 2]$
- $[[\begin{pmatrix}1\\1\\1\end{pmatrix}, \begin{pmatrix}0\\1\\2\end{pmatrix}], 1]$



➠ 講義の定義を復習しながらオーラドバックを用意しよう

➠ 講義の定義を復習しながらオーラドバックを用意しよう

MeLQSの観点から

MeLQS ＝ Mathematics e-Learning Question Specification

オンラインテスト問題の構成要素の標準を提唱

1. 問題変数 (ランダム変数) と問題文・解答欄

2. 解答変数と解答判定・フィードバック

※ 解答判定とフィードバックが教材の要

※ 異種システム間での教材共有を目的

※ STACK はこの仕様に適合する

※ 多肢選択問題は条件付きで適合可

誤答パターンに基づく選択肢と大量自動生成・ランダム出題の併用

STACKのメリット・デメリット

★ STACKのメリット😍

誤答分析によりフィードバックの改善も随時可

乱数生成をその場でプレビューして確認可

STACK独自の便利な関数あり

万一のバグも、受験開始後も修正対応可能

★ STACKのデメリット😢

プログラムに慣れるまで開発が面倒?

複雑なPRTの構成はやや時間がかかる

ちょっと重い😅

Maxima の関数がそのまま使えるとは限らない😕

システムのバグもときにはある😅

多肢選択問題のメリット・デメリット

★ 多肢選択問題のメリット

・Moodle 標準、他の LMS との互換性も期待

・動作が軽い 😀

・学生の入力負荷も軽い 😀

★ 多肢選択問題のデメリット

・乱数化できない←ランダム出題利用には大量生成が必要

・大量生成・ランダム出題の場合バグがあると致命的

受験開始後に 100 題中 35 題にバグ ⇒ 手動修正困難

・選択肢が複雑・多いと遅い ⇒KaTeX フィルタ

STACKによる多肢選択問題

STACKでも多肢選択問題を作成可能

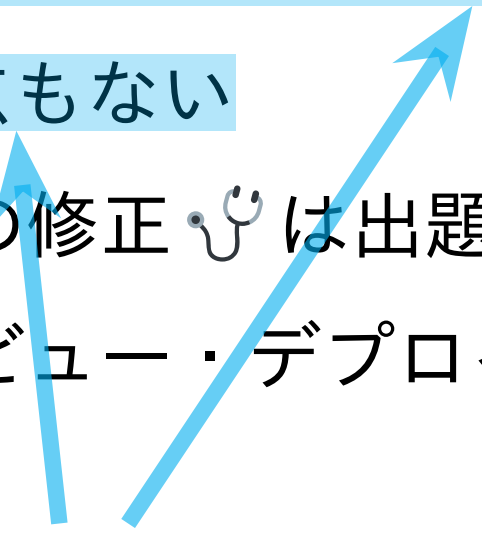
が，ドキュメント通りの実装だと...

🤔フィードバックができない(正解判定のみ)

😿部分点もない

🤫バグの修正🩺は出題後でも可能

🙂プレビュー・デプロイで事前確認容易



解決方法を紹介😺

STACKにおける多肢選択問題の実装

★ 多肢選択問題で用いるリストは２タイプ

タイプ１.[[opt1,true],[opt2,false],[opt3,true],...]

表示する式と真偽の組のリスト

タイプ２.[[1,true,opt1],[2,false,opt2],[3,true,opt3],...]

[解答識別子, 真偽, 表示する式] のリスト

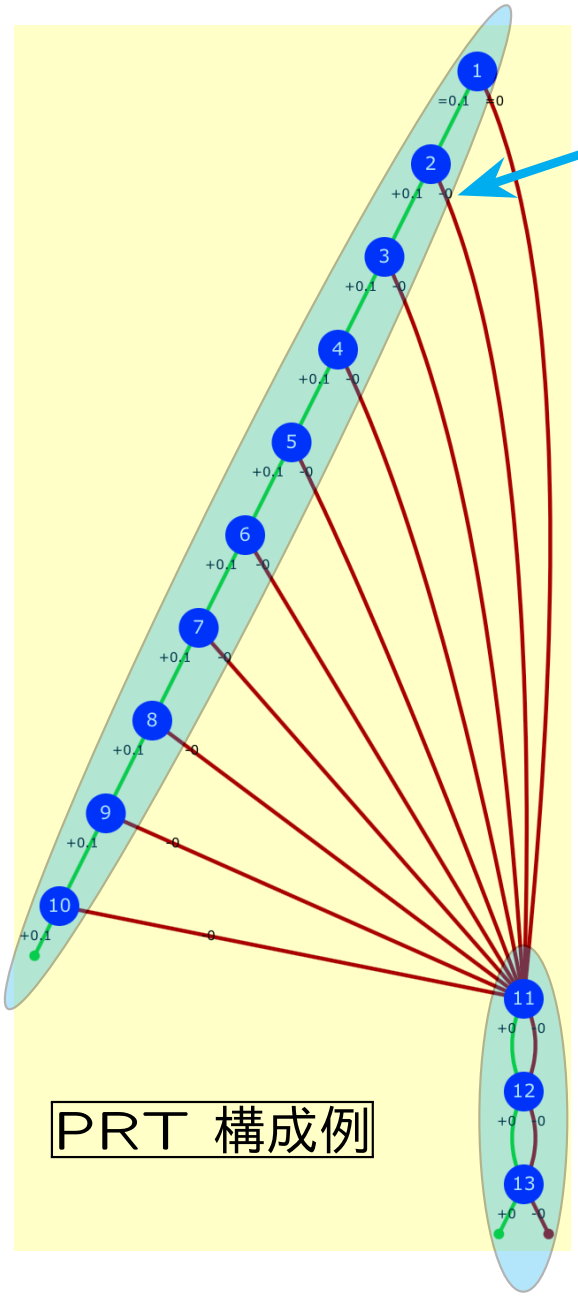
➠ タイプ２ を用いる.

誤答パターン:「解答識別子」の数値範囲を利用

部分点: 正解数/正解総数-誤答数/誤答総数による配列で設定

➠ 具体例で説明します 😅

STACK 多肢選択問題の実装 (PRT)



PRT 構成例

前半: 部分点処理

後半: flag によるフィードバック処理

実際のコード(問題概要)

STACKの問題の整頓 | 問題のテストとデプロ

行列Aを行に関して基本変形して, 被約階段行列(行簡約形)に変形すると, $\begin{pmatrix} 0 & 1 & 0 & -5 & -11 & 0 & -1 \\ 0 & 0 & 1 & 0 & -3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$になるという. このとき, この行列の列空間(列ベクトルで生成される空間)の基底を列ベクトルから選ぶとき, 基底になるベクトルの組合せとして正しいものをすべて選びなさい.

- ☐ $[a_2, a_3]$
- ☐ $[a_1, a_3, a_5]$
- ☐ $[a_3, a_4, a_6]$
- ☐ $[a_5]$
- ☐ $[a_5, a_6, a_7]$
- ☐ $[a_3, a_6, a_7]$
- ☐ $[a_7]$

実際のコード (準備)

```
cl:rand([7,8]);
rw:rand([4,5,5,6]);
rk:rand([2,3,3,4]);
/* Randechelon (STACK) */
RndNZ(n):=(-1)^(rand(2))*(rand(n)+1);
/* Return which column is first non-zero entry of i-th row vector */
getFirstNZ(M,i):=block([t,p,q],[p,q]:matrix_size(M),if M[i]=makelist(0,q) then
return(q+1) else for t: 1 thru q do (if not M[i,t]=0 then return(t)));
/* reduced echelon form */
redeche(M):=block([p,q,r,i,j,t],[p,q]:matrix_size(M),M:echelon(M),r:rank(M),if r=1
then return (M),for i:r thru 2 step -1 do (j:getFirstNZ(M,i),for t:1  thru i-1 do
(M:rowop(M,t,i,M[t,j]))),M);
/* randechelon : random echelon form */
randechelon(i1,j1,pv,cL):=block([i,j,ret],redeche(genmatrix(lambda([i,j],if i>length(pv)
 then 0 elseif j < pv[i] then 0 elseif j=pv[i] then 1 else rand(cL)),i1,j1)));
pvL:random_permutation(makelist(i,i,1,cl));
pv:sort(makelist(pvL[i],i,1,rk));
matF:randechelon(rw,cl,pv,[-5,-4,-3,-2,-1,-1,0,0,1,1,2,3,4,5]);
```

実際のコード (選択肢生成)

```
/* MC setting */
cwnL:random_permutation(rand([[1,3,3],[2,2,3]]));
cn1:cwnL[1];cn:cn1;
wn1:cwnL[2];wn2:cwnL[3];wn:wn1+wn2;
/* common function */
chkRank(pp,rr):=is(rank(apply(addcol,makelist(col(matF,pp[i]),i,1,length(pp))))=rr);
/* ListAL1 */
ListAL1:[makelist(a[pv[i]],i,1,rk)];/* default */
mkAL1(kmax):=block([k,k1,pL,pp],k1:0,for k:0 while (k<kmax and length(unique(ListAL1))
<=cn1+2) do (k1:k,pL:random_permutation(makelist(i,i,1,cl)),pp:
sort(makelist(pL[i],i,1,rk)),if chkRank(pp,rk) then ListAL1:append(ListAL1,
[makelist(a[pp[i]],i,1,rk)])),return(k1));
tmp:mkAL1(1000);
ListAL1:unique(ListAL1);
/* ListBL1: insufficient case */
ListBL1:makelist([a[i]],i,1,cl);
ListBL1:if rk>2 then append(ListBL1,[[a[pv[1]],a[pv[2]]]]) else ListBL1;
ListBL1:if rk>2 then append(ListBL1,[[a[pv[1]],a[pv[3]]]]) else ListBL1;
ListBL1:if rk>2 then append(ListBL1,[[a[pv[2]],a[pv[3]]]]) else ListBL1;
ListBL1:if rk>3 then append(ListBL1,[[a[pv[1]],a[pv[2]],a[pv[3]]]]) else ListBL1;
ListBL1:unique(ListBL1);
```

⇐正解数、各誤答パターン数の決定

⇐リストの重複を除く (重要)

⇐リストの重複を除く (重要)

実際のコード (選択肢生成 2)

```
/* ListBL2: rank mismatch */
ListBL2:[];
mkBL2(kmax):=block([k1,k,pL,pp],for k:0 while (k<kmax and length(unique(ListBL2))<=wn2+2) do
 (k1:k,pL:random_permutation(makelist(i,i,1,cl)),pp:sort(makelist(pL[i],i,1,rk)),if not
 chkRank(pp,rk) then ListBL2:append(ListBL2,[makelist(a[pp[i]],i,1,rk)])),return(k1));
tmp:mkBL2(1000);ListBL2:unique(ListBL2);
/* Correct List */
CorL1:random_permutation(makelist([i,true,ListAL1[i]],i,1,length(ListAL1)));
/* CorL2:random_permutation(makelist([i+20,true,ListAL2[i]],i,1,length(ListAL2))); */
/* Wrong List */
WrgL1:random_permutation(makelist([100+i,false,ListBL1[i]],i,1,length(ListBL1)));
WrgL2:random_permutation(makelist([200+i,false,ListBL2[i]],i,1,length(ListBL2)));
va:random_permutation(append(makelist(CorL1[i],i,1,cn1),makelist(WrgL1[i],i,1,wn1),
  makelist(WrgL2[i],i,1,wn2)));
Cans1:mcq_correct(va);
```

正当パターンは i ($i < 100$)

※ 区別する必要があるときは, 例えば 20 毎に区切る

誤答パターン 1 は $100 + i (i = 1, \ldots)$ の解答識別子

誤答パターン 2 は $200 + i (i = 1, \ldots)$ の解答識別子

識別子を 100 で割った商で正当誤答の判別が可能

実際のコード (解答変数)

```
/* ansL は id 識別子を 10 で割った正誤と誤答パターンのリスト */
ansL:map(lambda([u],floor(u/100)),ans1);
/* alen チェックを入れた数 */
alen:length(ans1);
/* CansL [0,0,...,0] ansL と一致すれば正解 */
Calen:cn;
CansL:makelist(0,i,1,Calen);
count0(L):=block([ct,i],ct:0,for i:1 thru alen while i <= alen do if L[i] = 0 then ct:ct+1,retu
/* ct0 チェックのうちの正解の数 */
ct0:count0(ansL);
/* sc0 通常多肢選択問題で期待される部分点 */
sc0:ct0/cn-(alen-ct0)/wn;
/* scL STACK の分岐による部分点に変換するための T/F flag リスト */
scL:makelist(is(sc0>=0.1*i),i,1,10);
/* flg1,... 誤答パターン検出フラグ */
flg1:member(1,ansL);
flg2:member(2,ansL);
flg0:is(ct0 < cn);
```

flg0 : 正解をすべて選択していない

flg1 : 誤答パターン 1 を選択

flg2 : 誤答パターン 2 を選択

scL: 正解数に応じた true,false の配列

➠PRT で部分点の決定

実際のコード(PRT)


ノード 1
評価関数 代数等価
評価対象 scL[1]
評価基準 true
オプション
抑制 Yes
真の場合のノード1
計算 = 点数 0.1 減点 次のノード ノード 2 解答記録 prt1-1-T
ノード'1'の真のフィードバック
偽の場合のノード1
計算 = 点数 0 減点 次のノード ノード 11 解答記録 prt1-1-F
ノード'1'の偽のフィードバック
イタリック [Cmd + i]
ノード 1の削除
ノード 2
評価関数 代数等価
評価対象 scL[2]
評価基準 true
オプション
抑制 No
真の場合のノード2
計算 + 点数 0.1 減点 次のノード ノード 3 解答記録 prt1-2-T
ノード'2'の真のフィードバック
偽の場合のノード2
計算 - 点数 0 減点 次のノード ノード 11 解答記録 prt1-2-F
ノード'2'の偽のフィードバック
ノード 2の削除
ノード 3
評価関数 代数等価
評価対象 scL[3]
評価基準 true
オプション
抑制 No

まとめと今後の予定

- STACK の多肢選択問題も PRT の工夫で MeLQS 準拠に
- フィードバックも問題変数で設定は可能 (文字化け問題の解決が必要)

  多肢選択問題自動生成への応用が可能?⇓
- 条件設定による雛形のダウンロードサイトの構築
- STACK で生成した問題変数から

  Moodle 多肢選択問題を掃き出す仕掛け (を開発すれば)

  ⇓

  バグ free(less?) 多肢選択問題大量生成 😌

最後に

📧 yositomi@las.osakafu-u.ac.jp

- 📝 多肢選択問題のアイデア募集 ➠ 作ります
- 🗄️ 開発済み XML 提供します 🖖🏽

http://www.las.osakafu-u.ac.jp/~yositomi/moodle_xml/STMC/

良質な問題データの共有 ↔ 教育に関する経験知の共有 👍