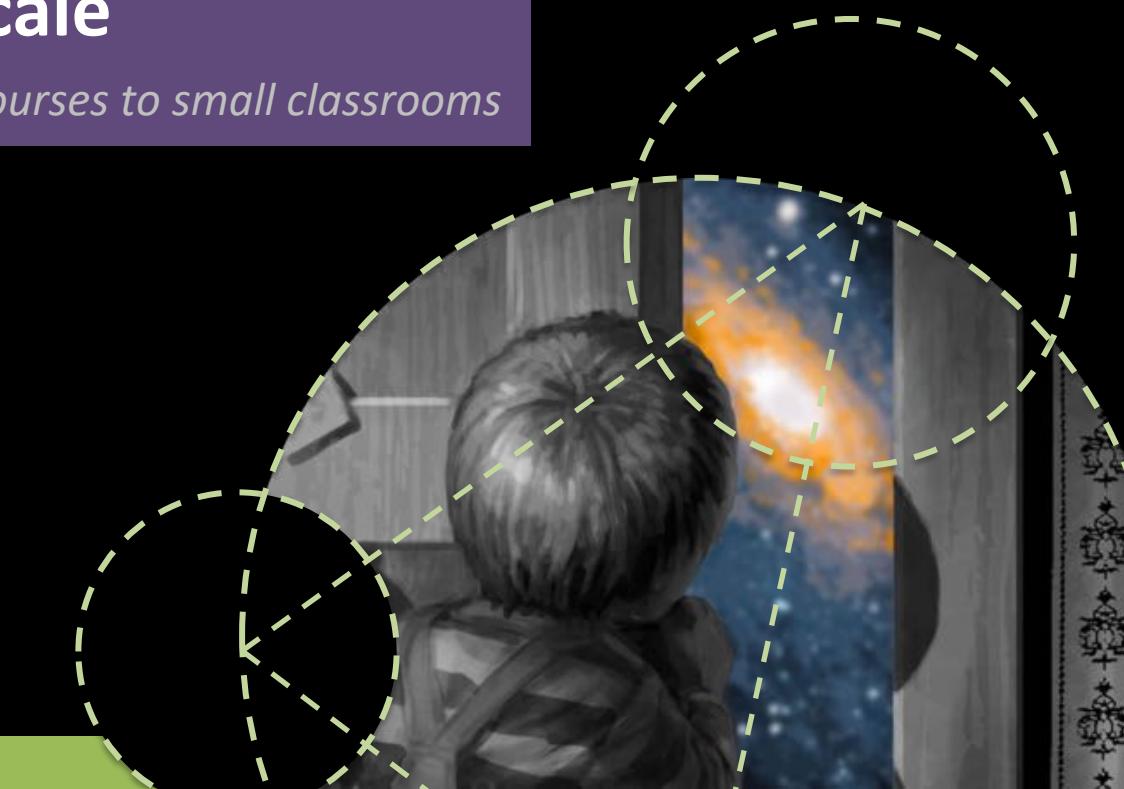


# Understanding Students at Scale

*From massive online courses to small classrooms*

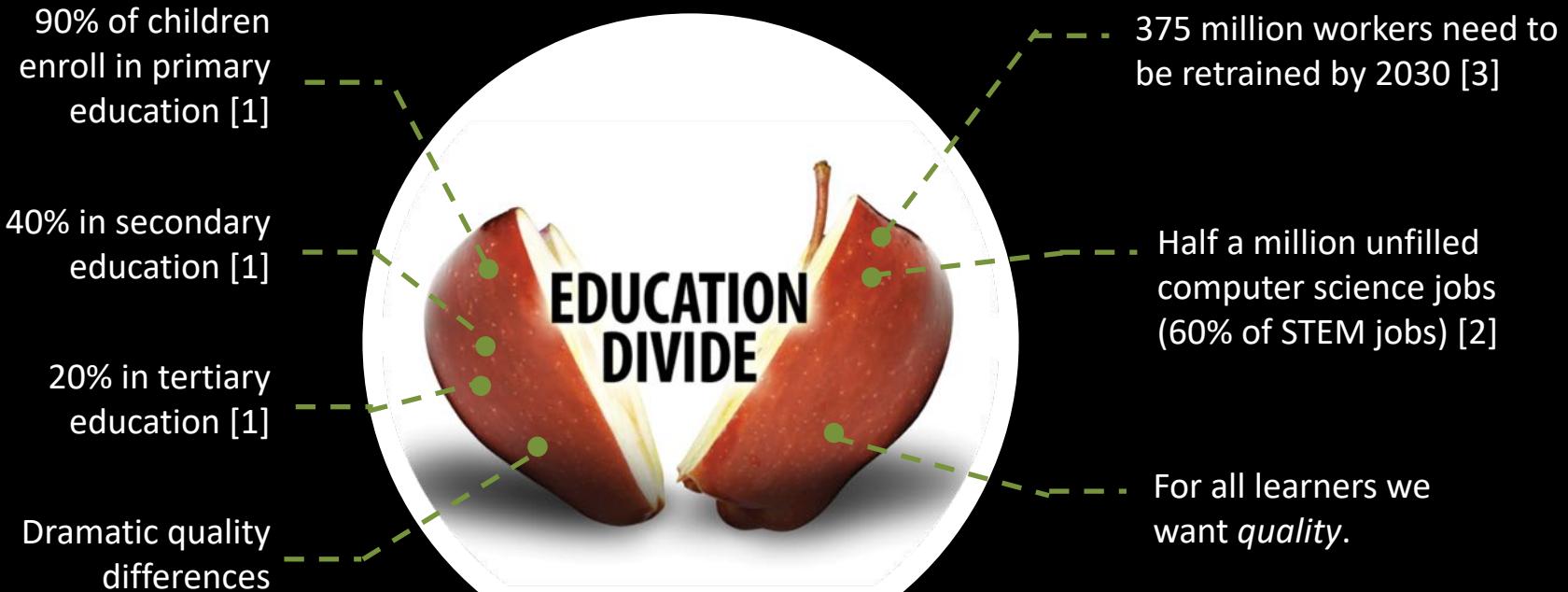
April 2019

Chris Piech



# Chapter 0: The Grand Challenge

# Quality Education Gap



[1] *World development indicators 2015*. World Bank Publications, 2015.

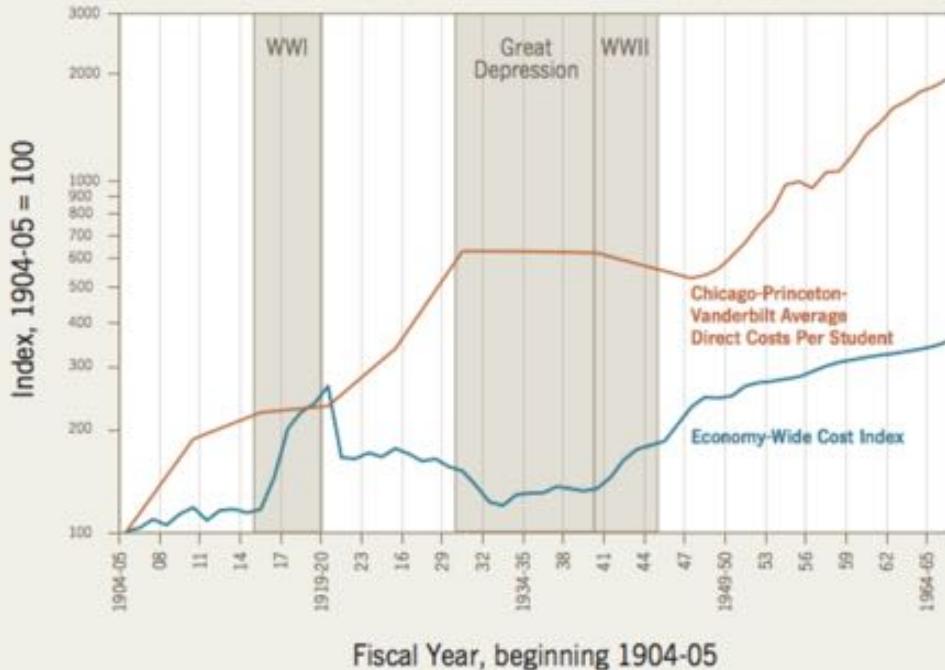
[2] USA Bureau of Labor Statistics Employment Projections , 2016.

[3] Jobs lost, jobs gained. McKinsey Global Institute, 2017.

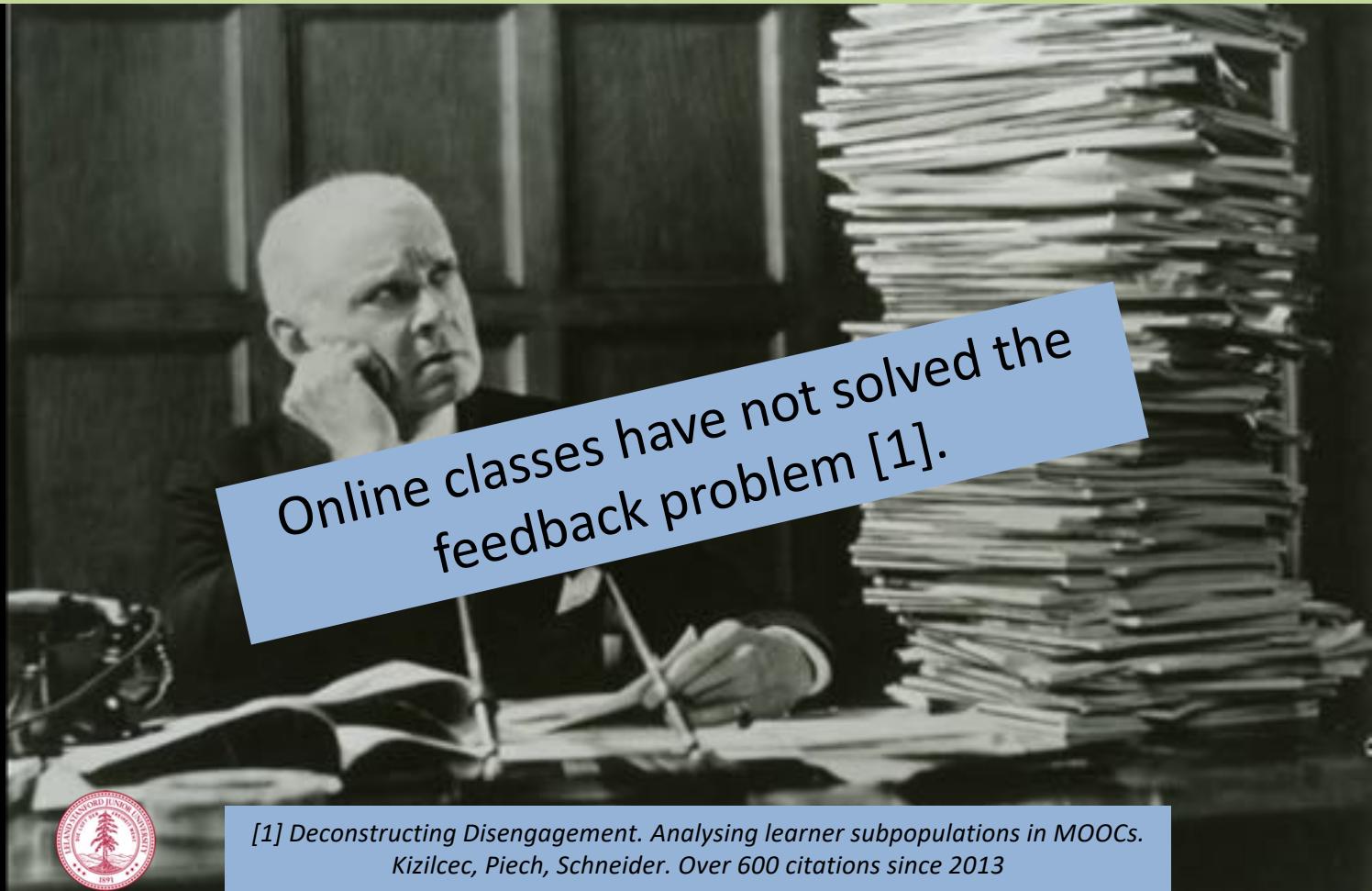


# Cost Disease of Education

Direct Costs Per Student,  
Compared with an Economy-Wide Cost Index



# Feedback is Labor Intensive



# Progress in other domains



# The Last Remaining Board Game



# Self Driving Cars



# Grand Challenge in Education

---

Clear  
Societal Need

Increase  
education  
productivity

New  
Datasets of  
Learning

MOOC  
assignments

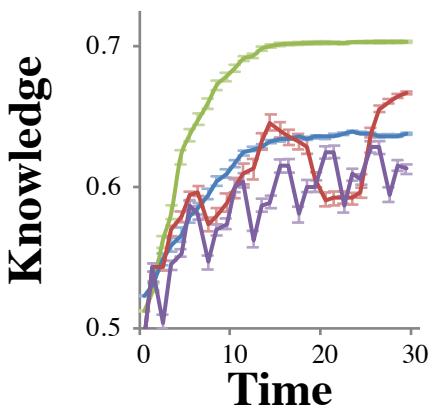
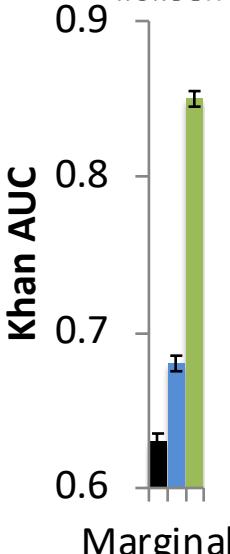
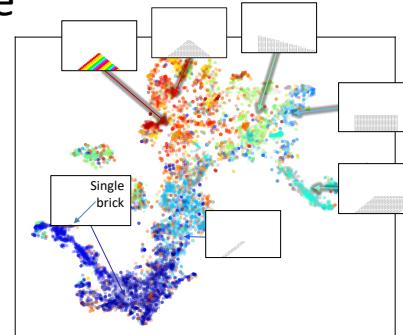
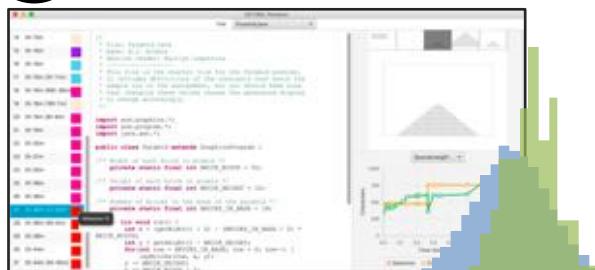
Deep  
Learning

Applied  
Math  
Renaissance

Autonomously support education by  
better understanding students.



# In Today's Talk

- 1 Temporal understanding of students
- 2 Zero-Shot Deep understanding of code
- 3 Real human impact
- 4 Taste of next steps for AI in Education





# Chapter 1: Deep Knowledge Tracing

## *Feedback in a simple context*



# Story of Riley



## Exercise Type:

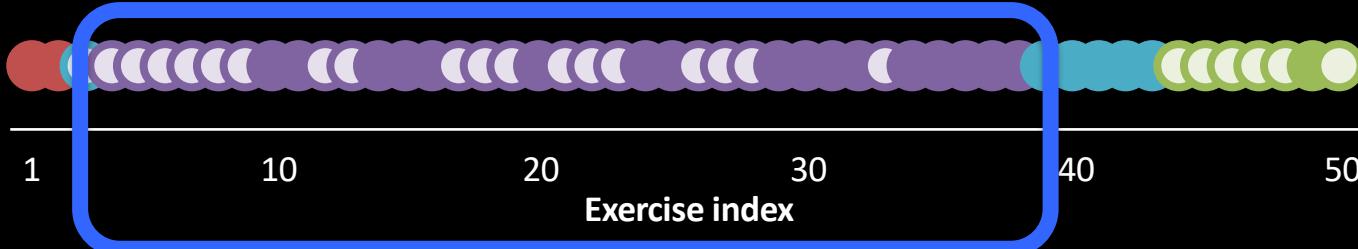
- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

## Answer:

- Correct
- Incorrect



# Story of Riley



## Exercise Type:

- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

## Answer:

- Correct
- Incorrect



# Story of Riley



## Exercise Type:

- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

## Answer:

- Correct
- Incorrect



# Story of Riley



What should Riley do next?

**Exercise Type:**

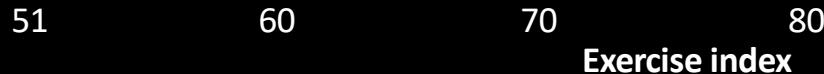
- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

**Answer:**

- Correct
- Incorrect



# Story of Riley



## Exercise Type:

- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots
- Slope of a line

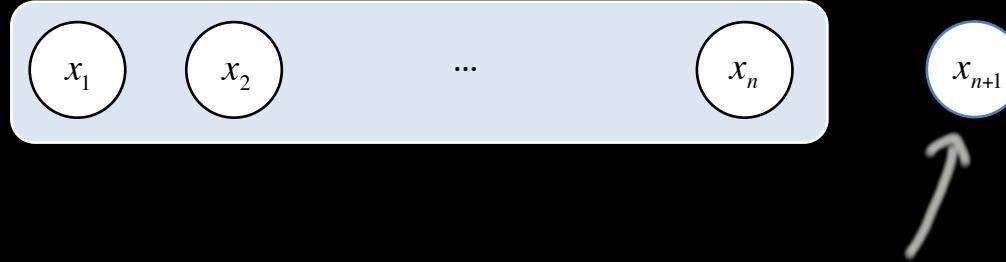
## Answer:

- Correct
- Incorrect



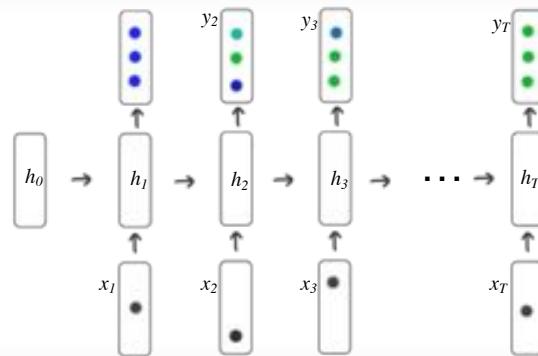
# Knowledge Tracing for Feedback

Given  $n$  historical answers:

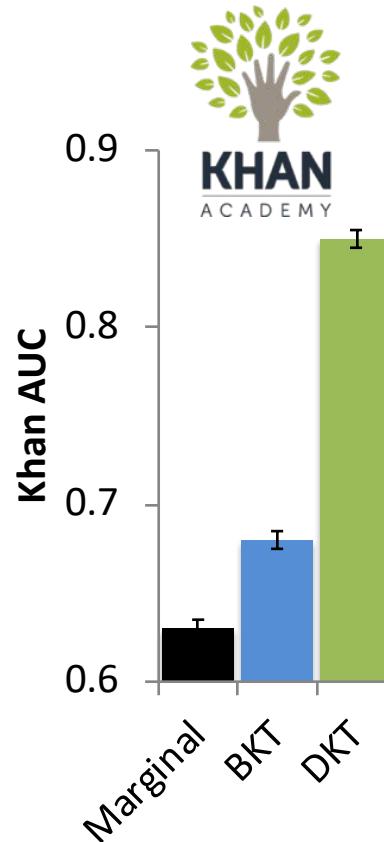
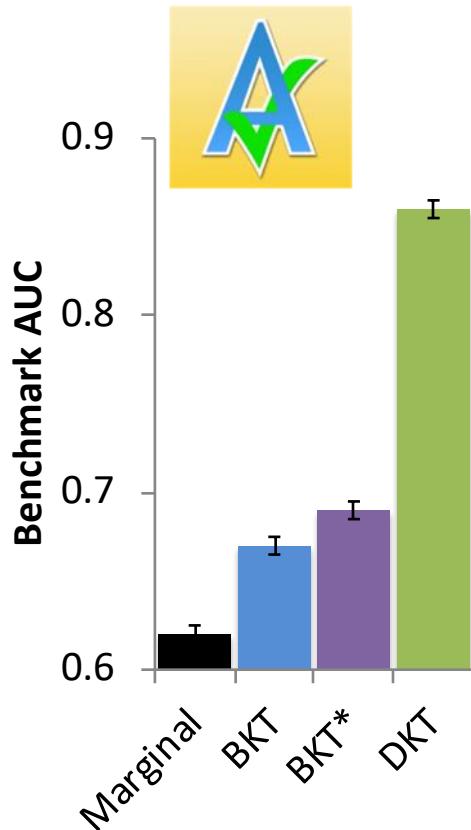


Predict the next  
one

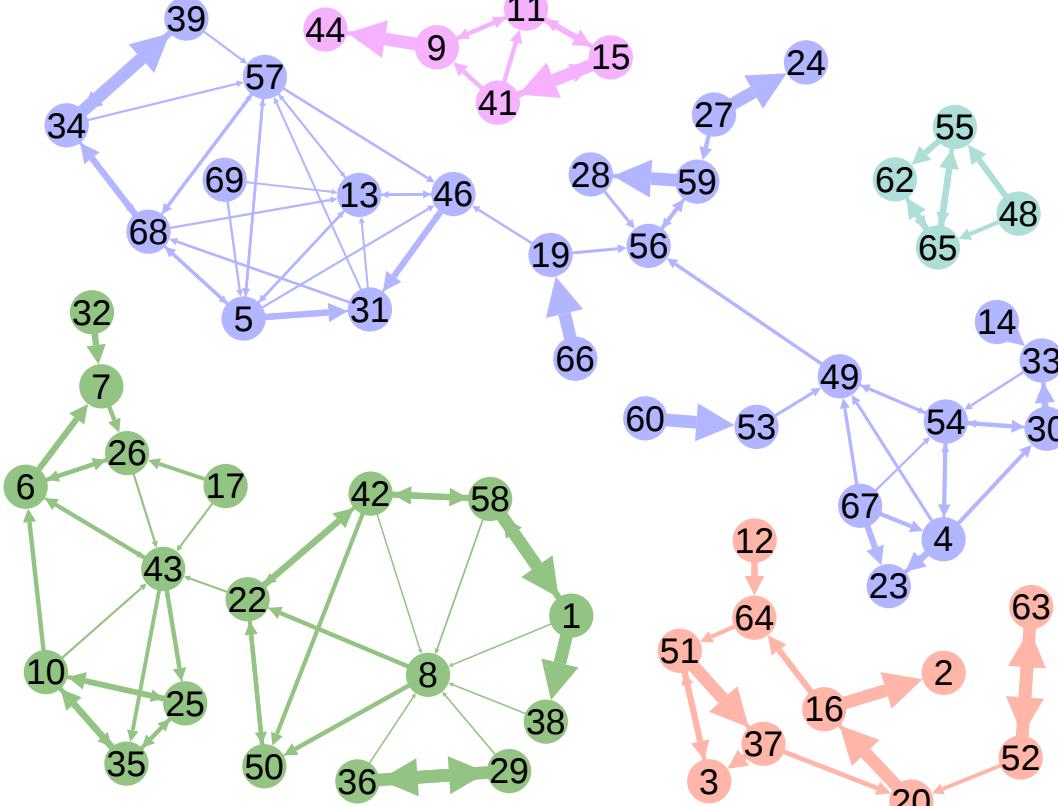
# Build the *first* deep learning algorithm for human learning



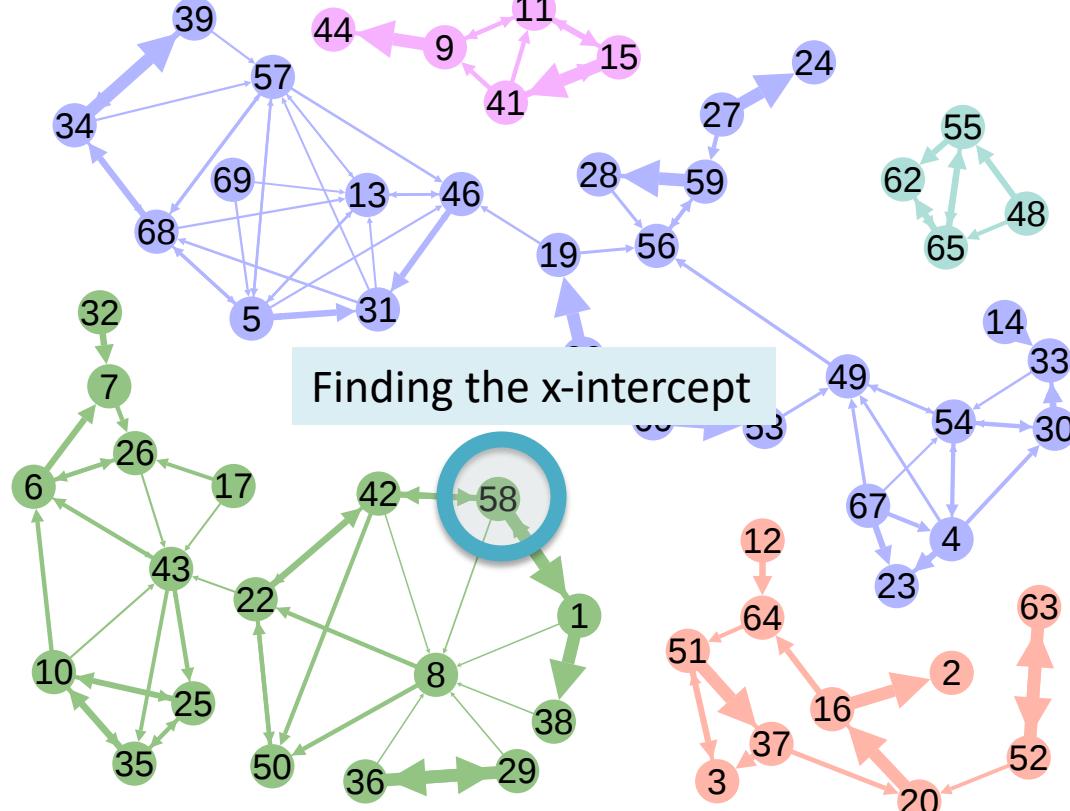
# Understanding Students



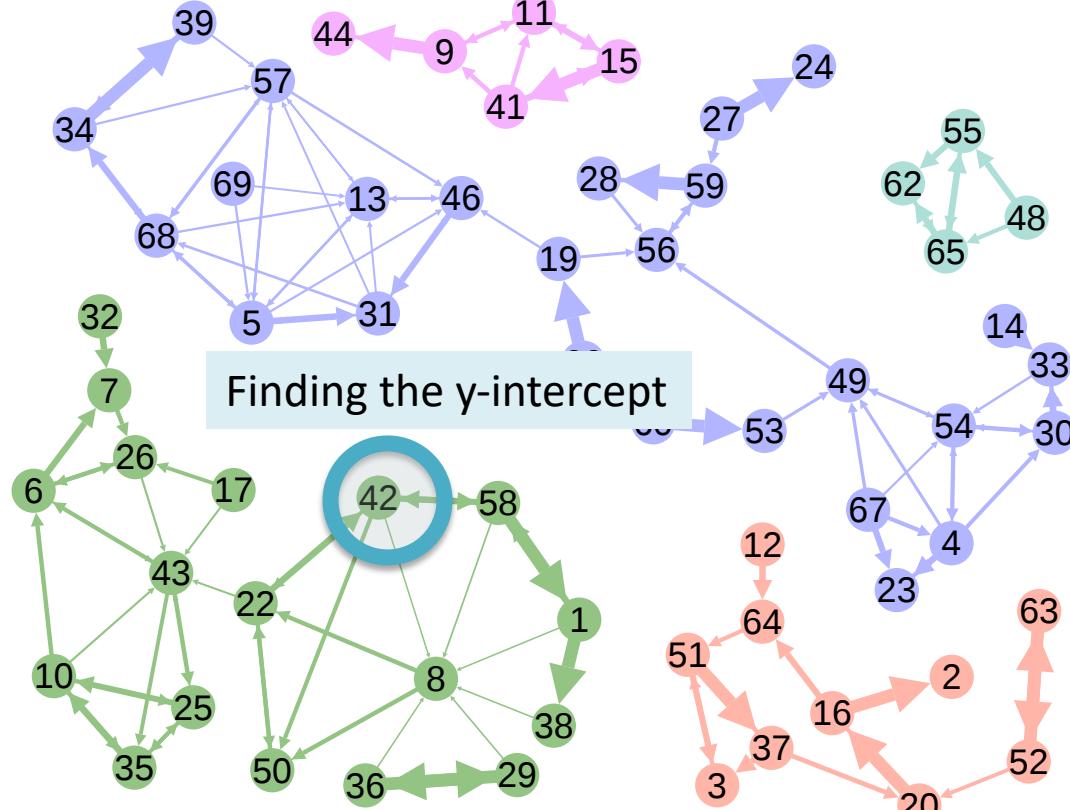
# Learns Concept Relationships



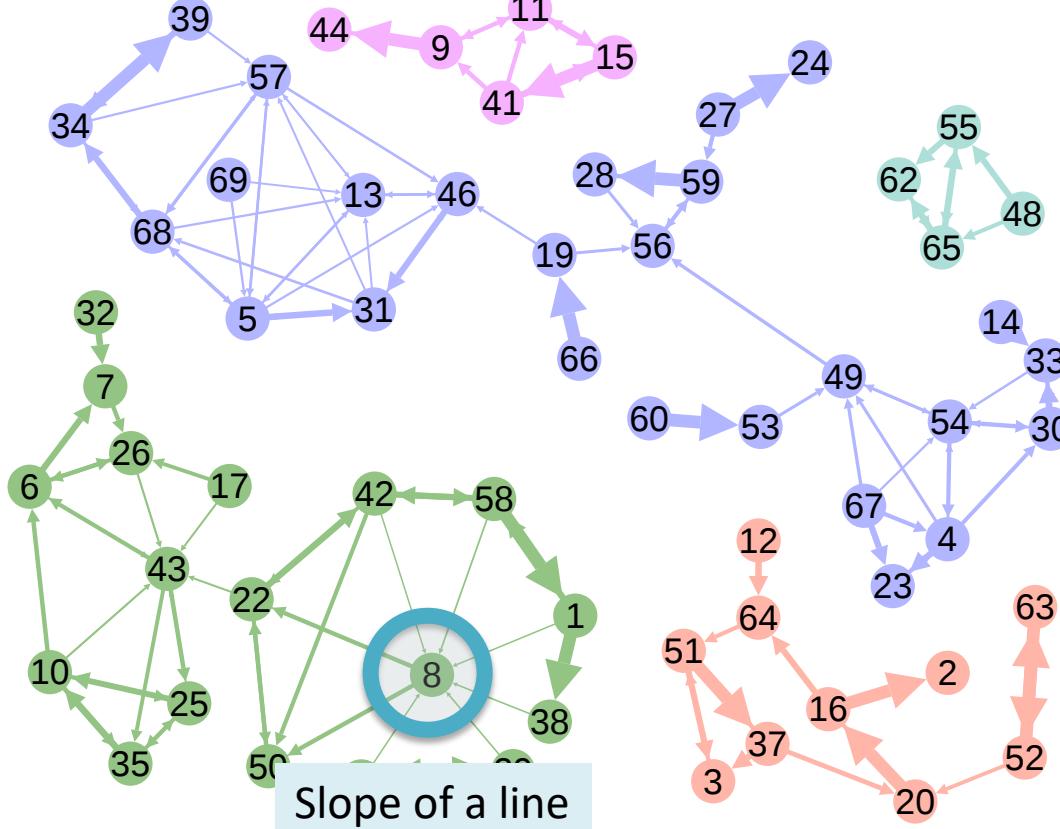
# Learns Concept Relationships



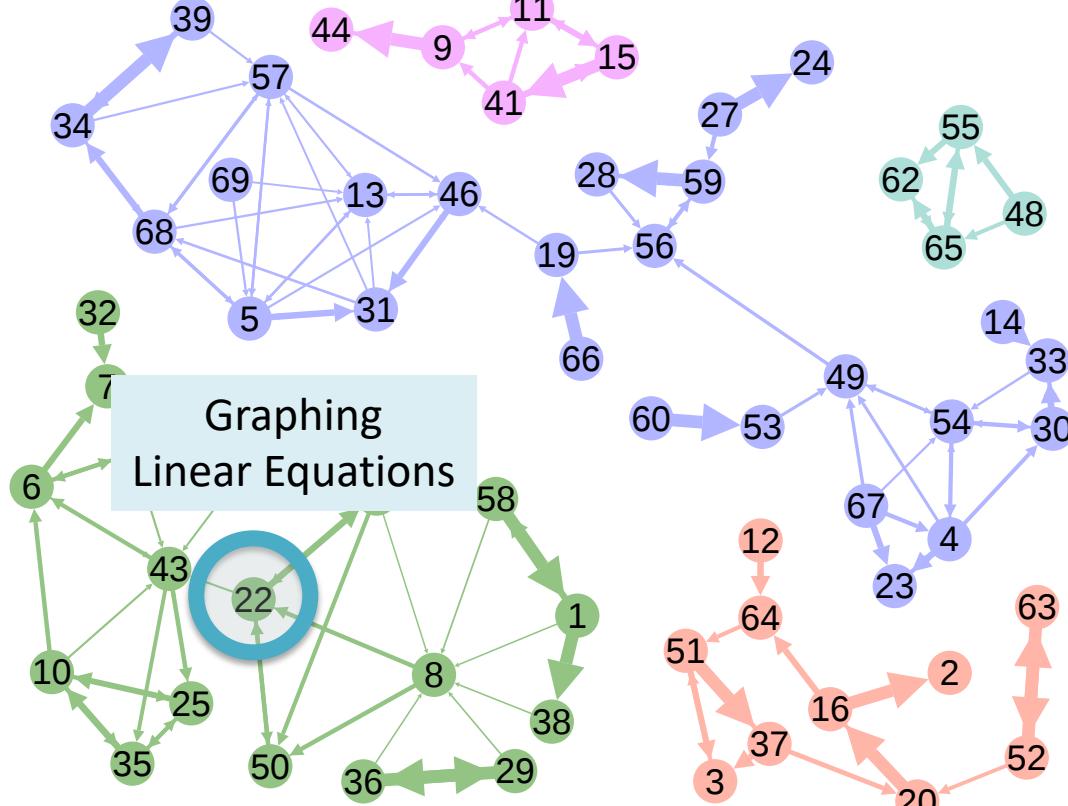
# Learns Concept Relationships



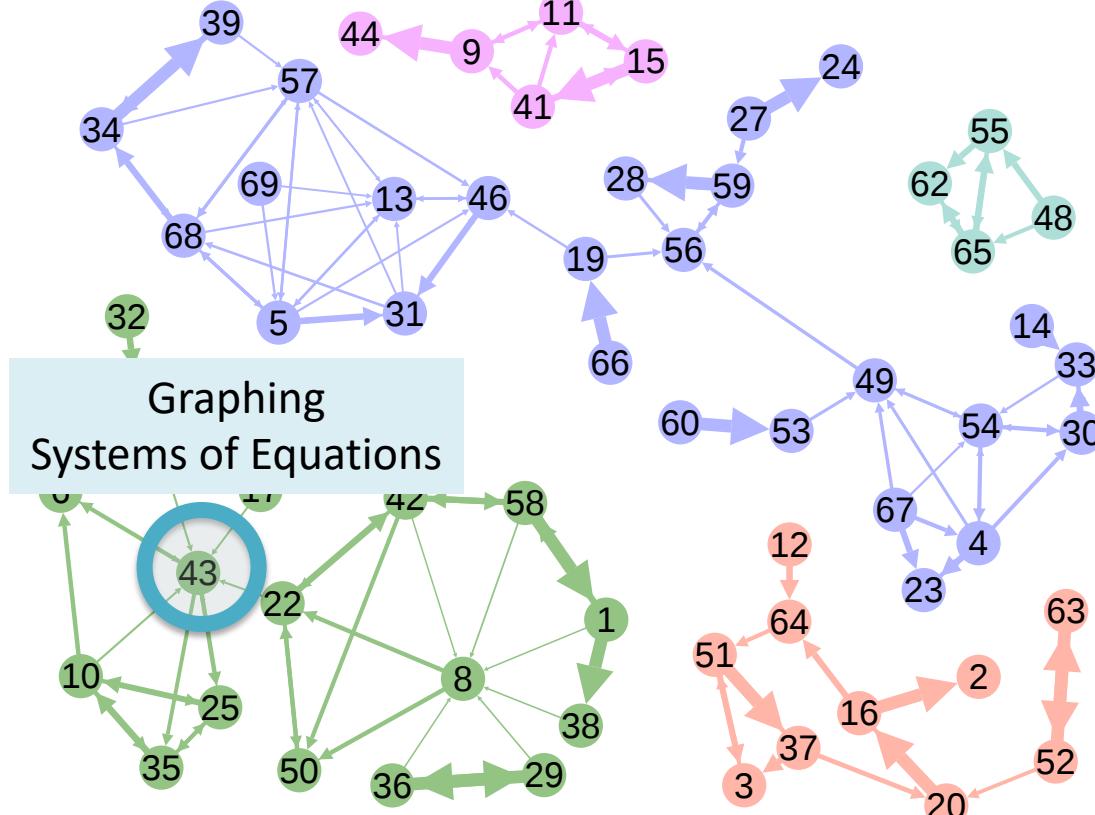
# Learns Concept Relationships



# Learns Concept Relationships

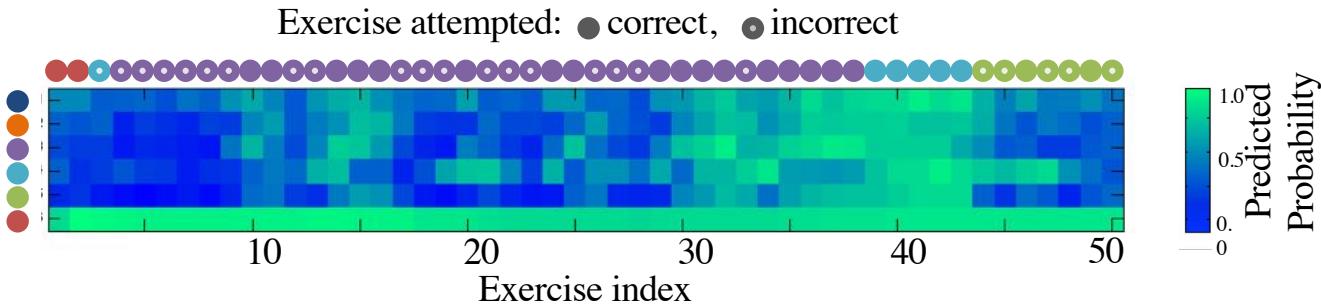


# Learns Concept Relationships



Can it help make decisions?

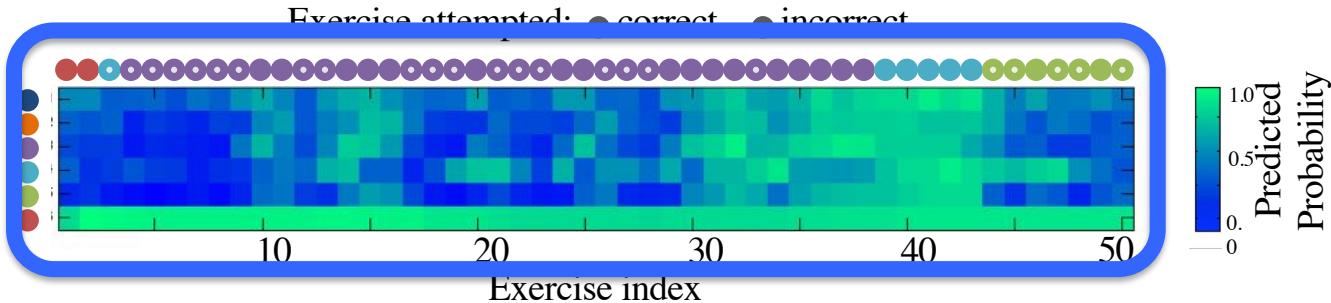
# Predicting Next Item



- Line graph intuition
- Slope of a line
- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots



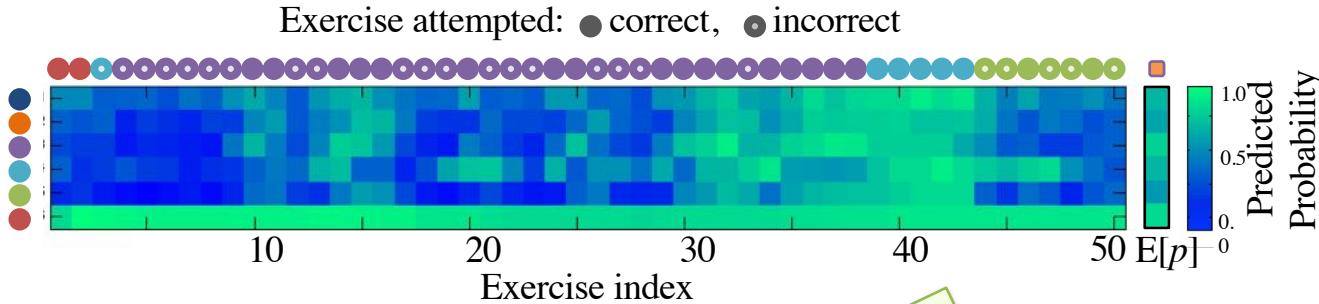
# Predicting Next Item



- Line graph intuition
- Slope of a line
- Solving for x-intercept
- Solving for y-intercept
- Graphing linear equations
- Square roots



# Predicting Next Item

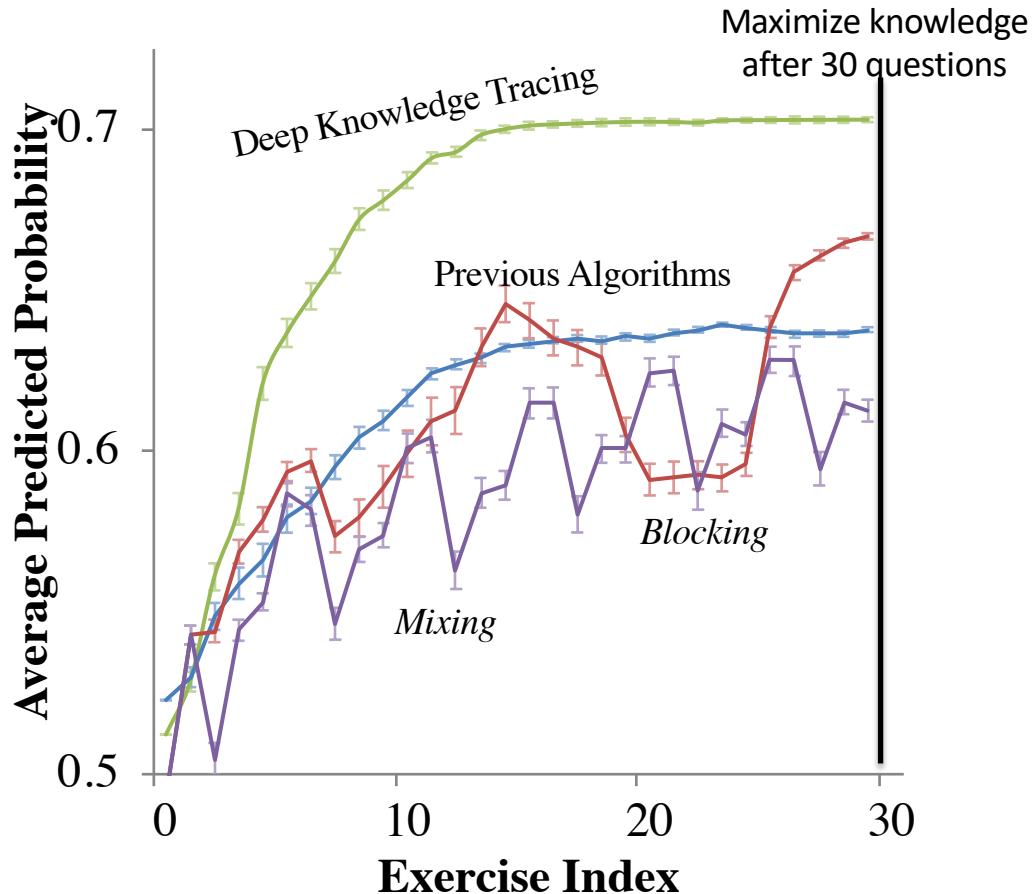


We can find what is the “deal maker”

- Line graph
- Slope
- Intercept
- For y-intercept
- Graphing linear equations
- Square roots



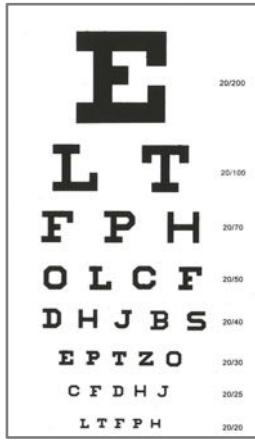
# Optimal Teaching



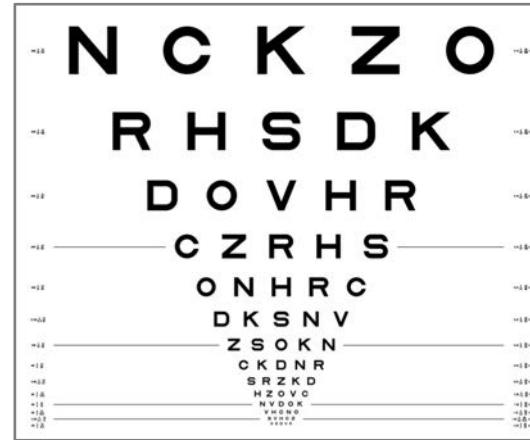
Aside: optimal psychometrics

# Visual Acuity Test

Snellen



ETDRS

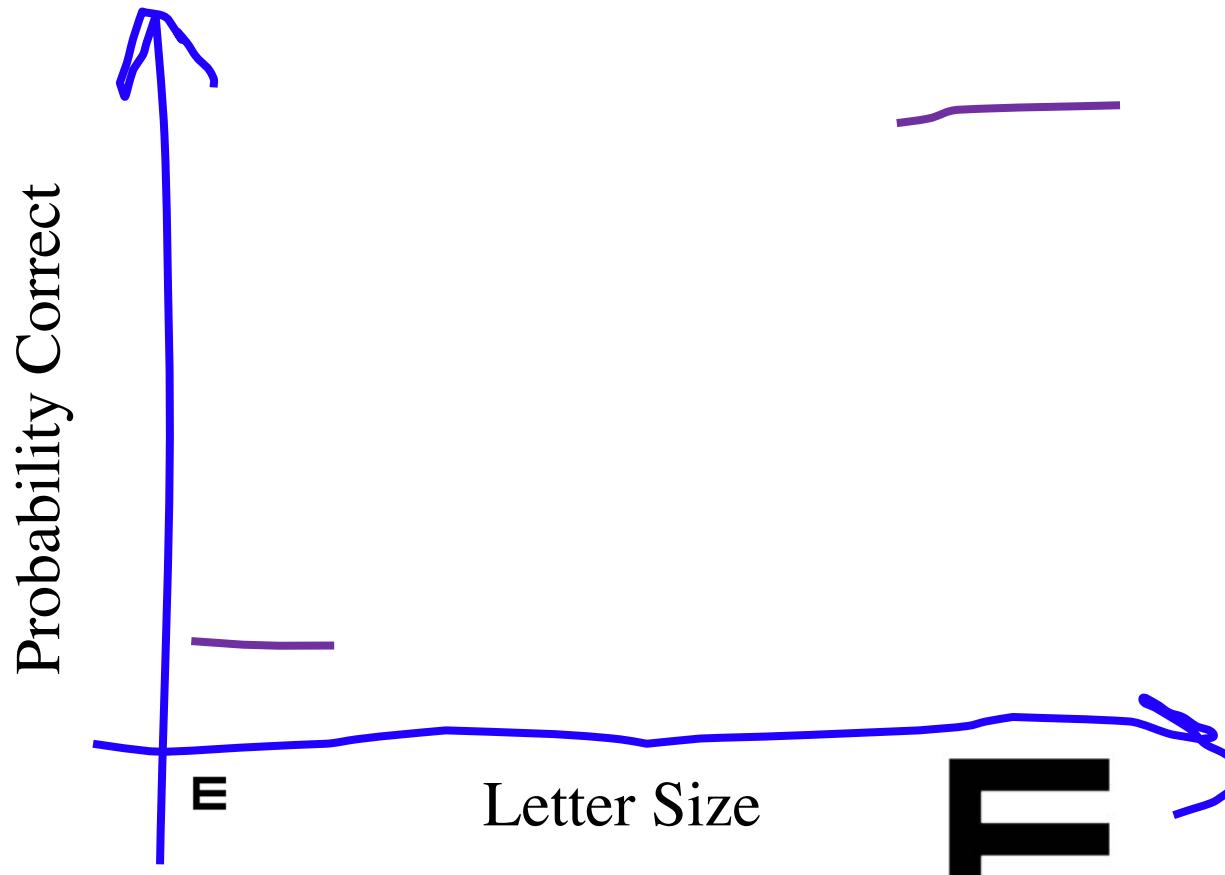


How can we make an accurate,  
adaptive eye exam?

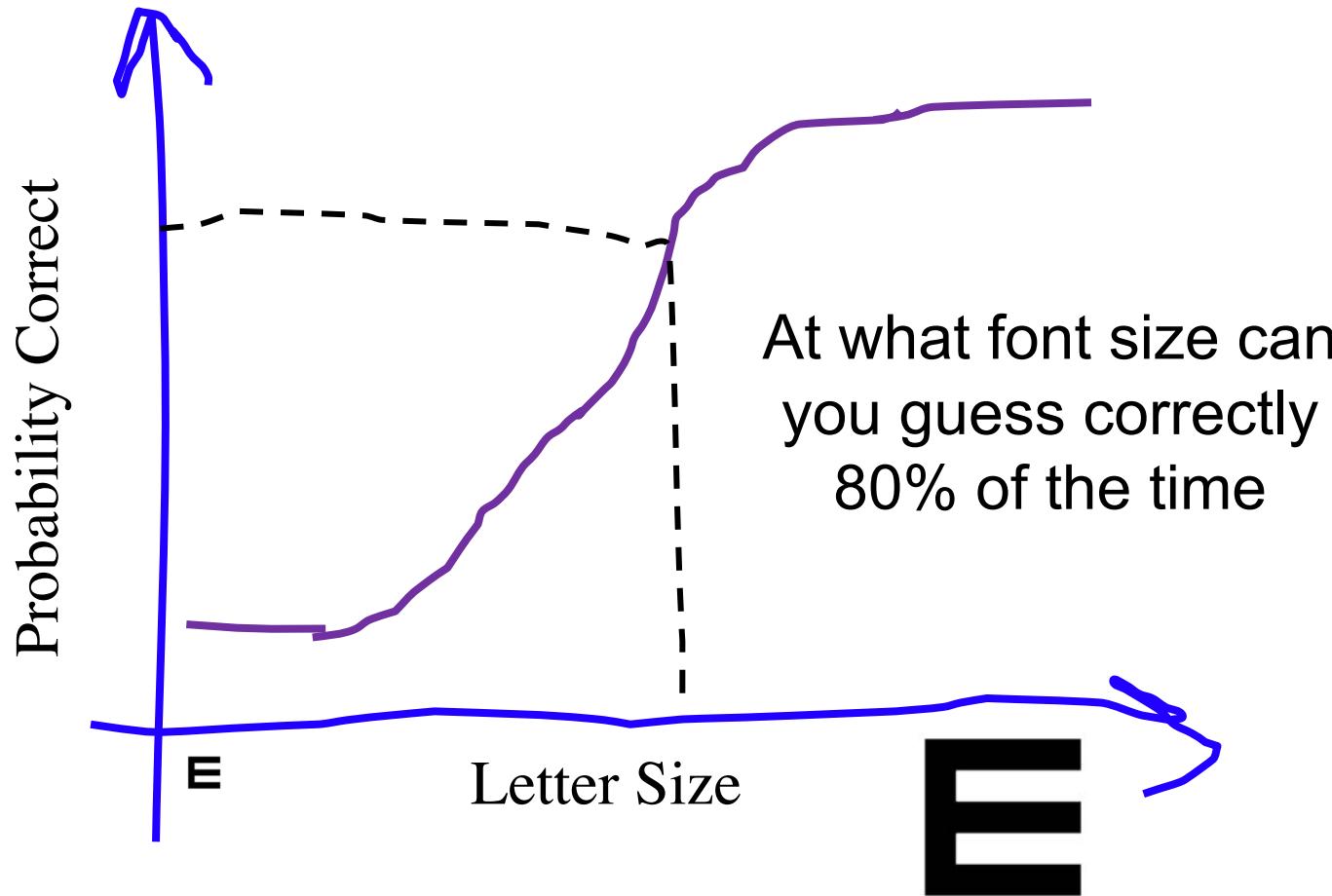
20 adaptively chosen letter sizes



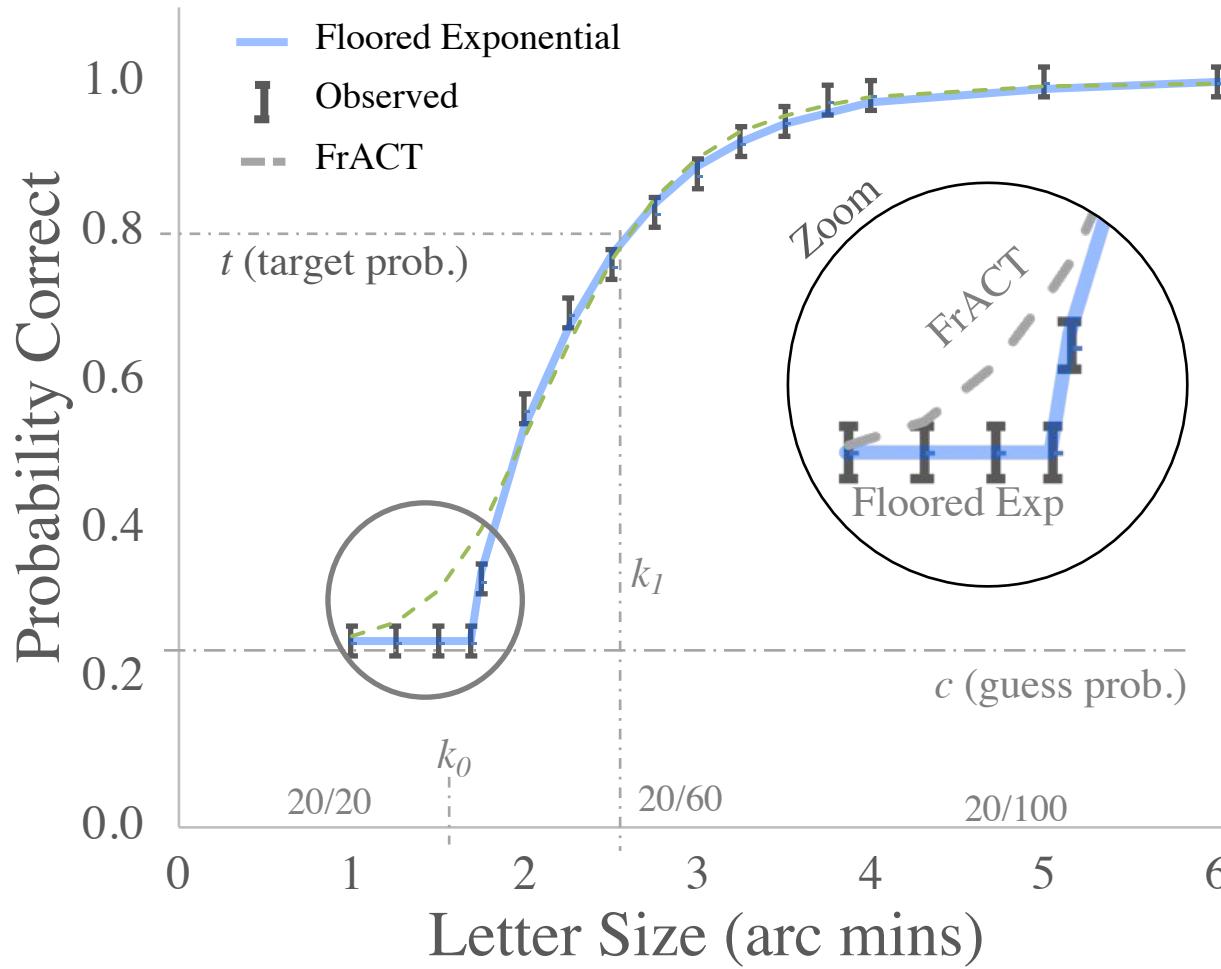
# Visual Acuity Response Function



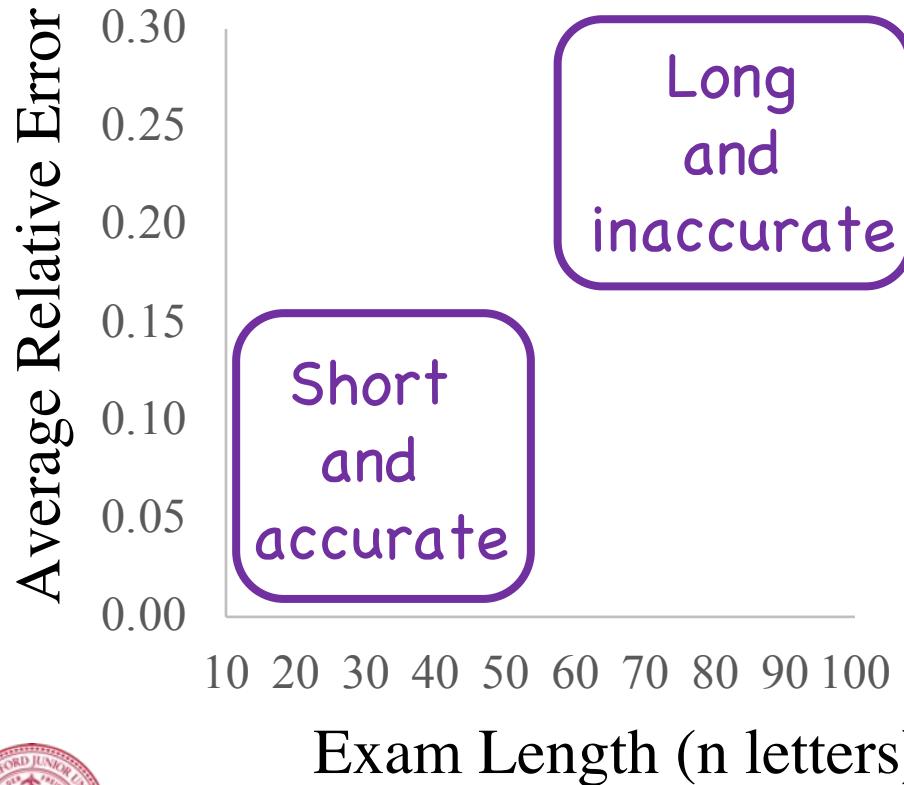
# Visual Acuity Response Function



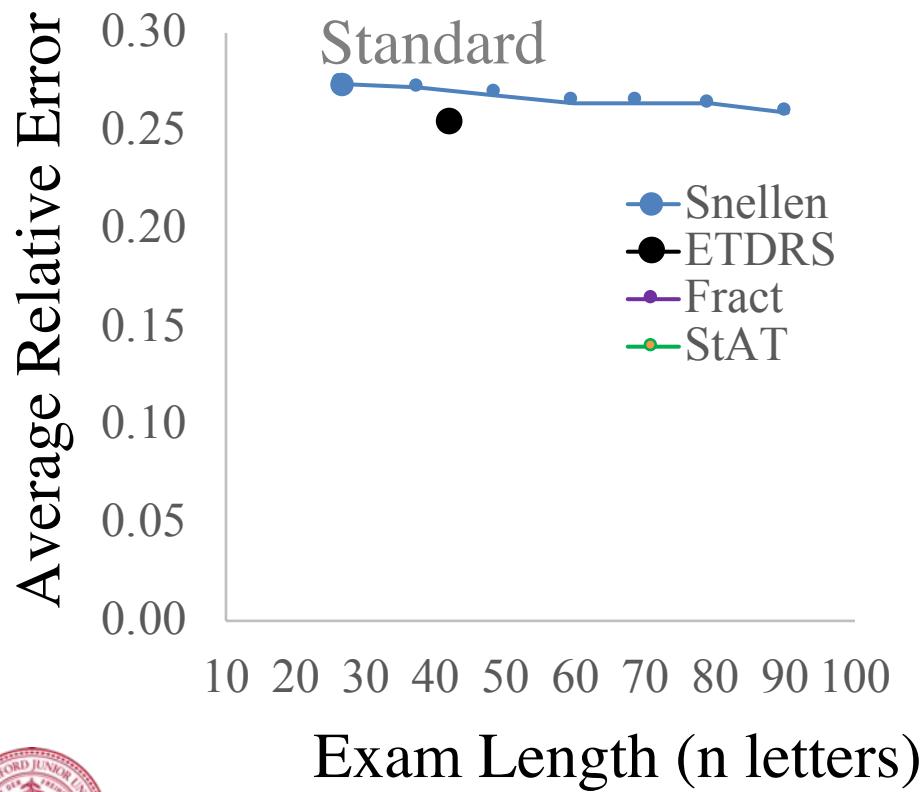
# Visual Acuity Response Function



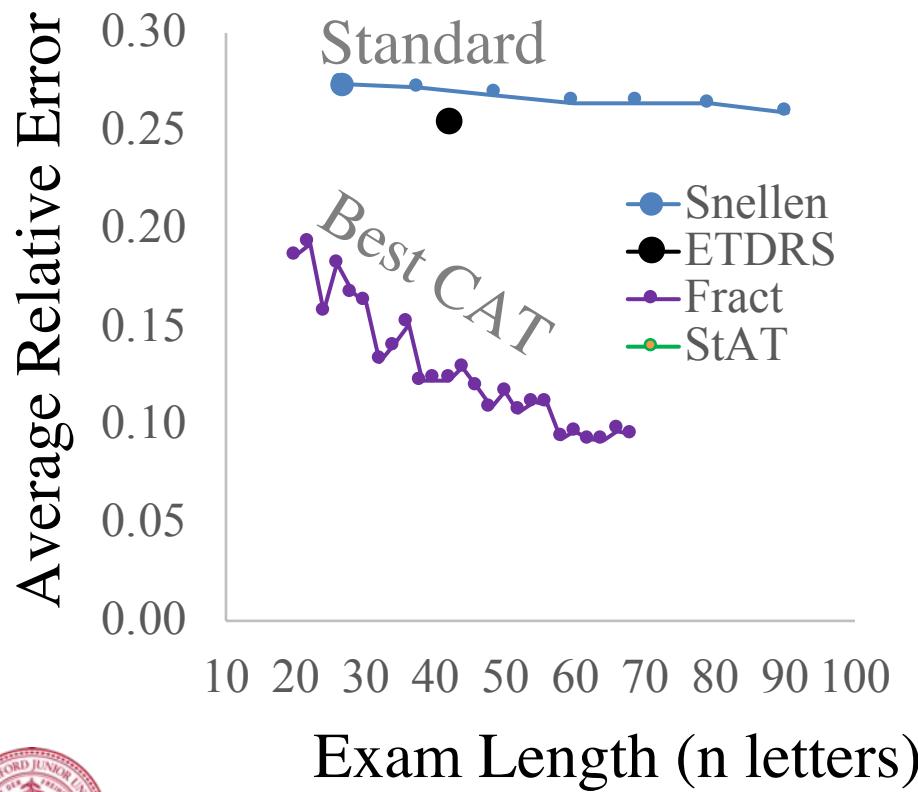
# Improved Eye Exam



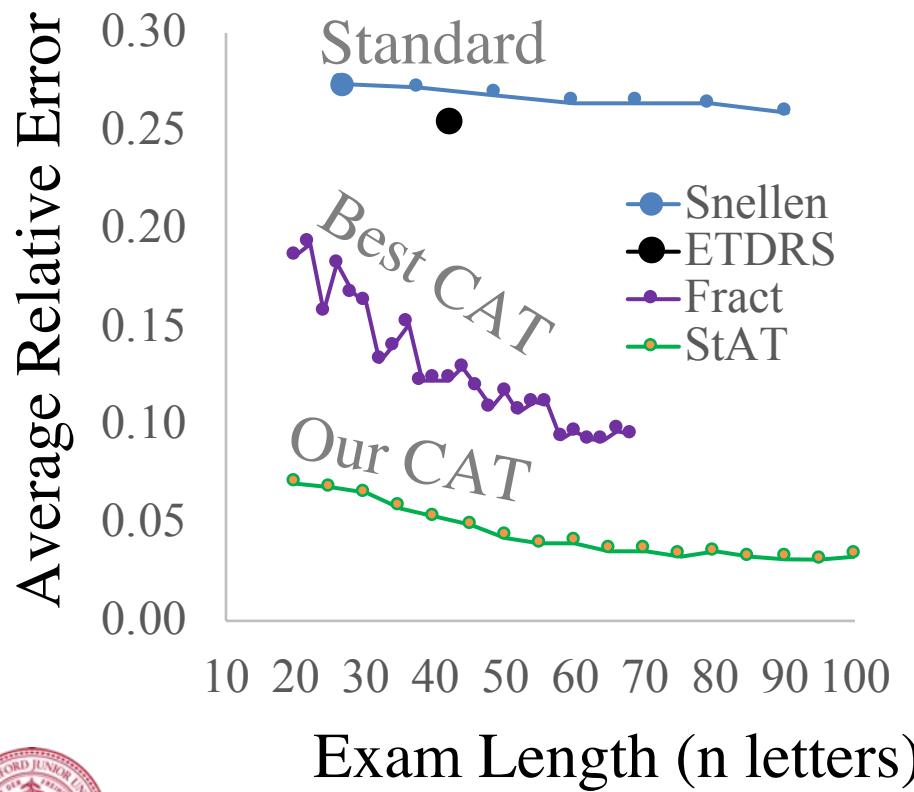
# Improved Eye Exam



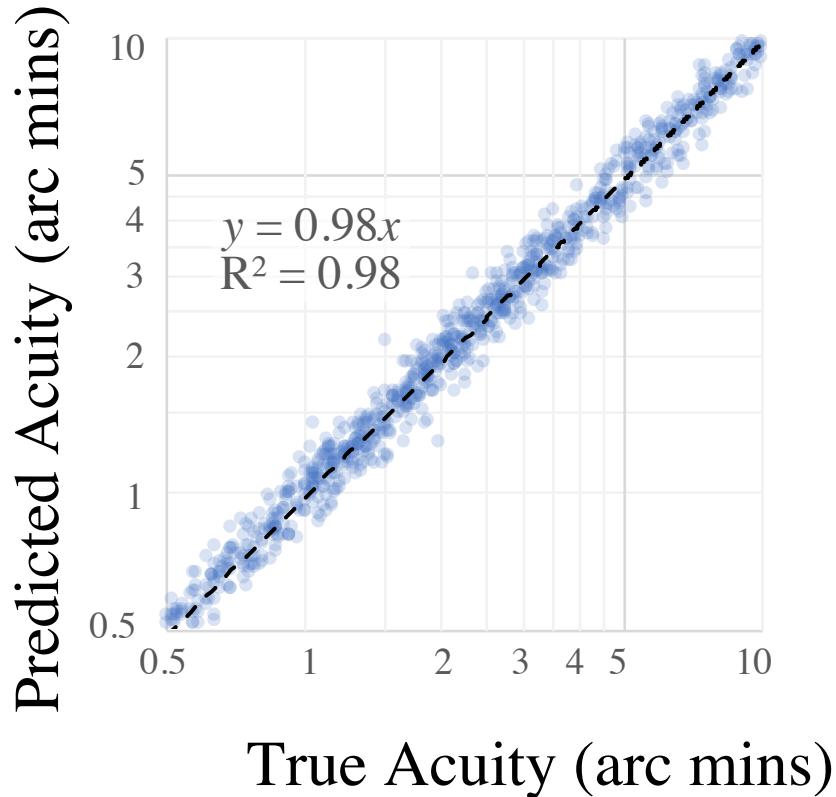
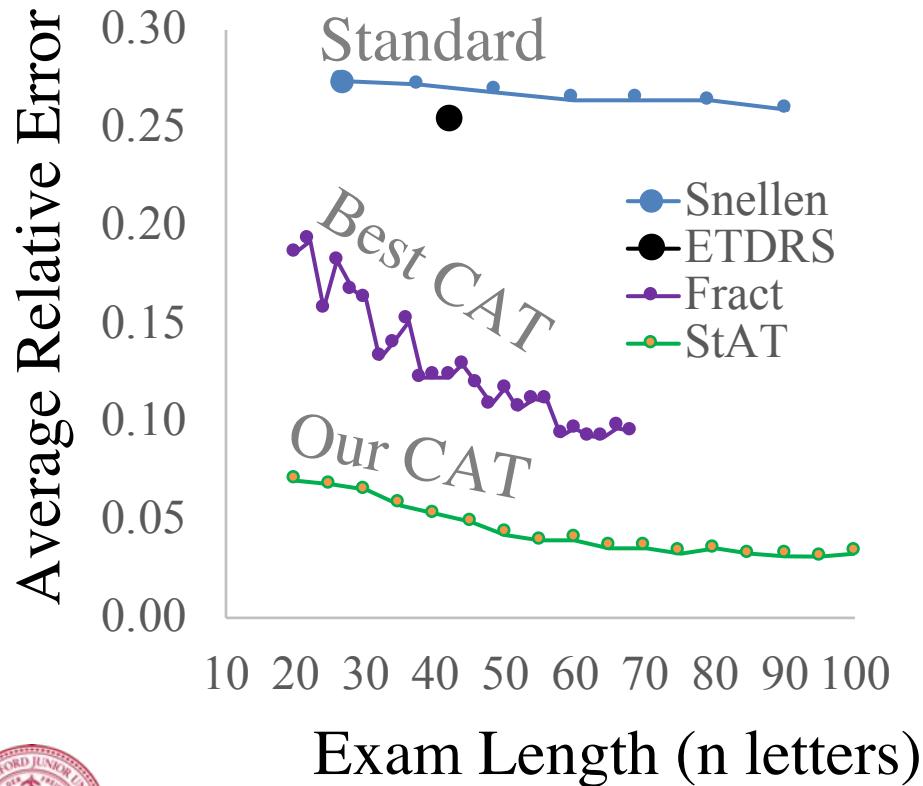
# Improved Eye Exam



# Improved Eye Exam



# Improved Eye Exam



Visus Test

Left Eye

StAT Algorithm

N done: 15

MAP acuity: 2.5 arcmin

Interval: [2.1, 3.6] arcmins

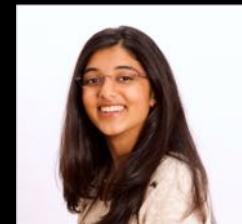
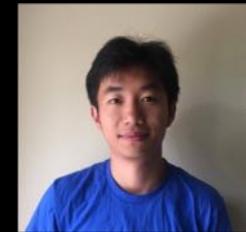
Likelihood of Acuity Scores:

Progress: 75%

Acuity (logMAP)

LELAND STANFORD JUNIOR UNIVERSITY  
1891

# Chapter 2: Zero Shot Feedback



*Best student paper at AAAI 2019*

# Online Classes Haven't Solved the Problem

Feedback is hard



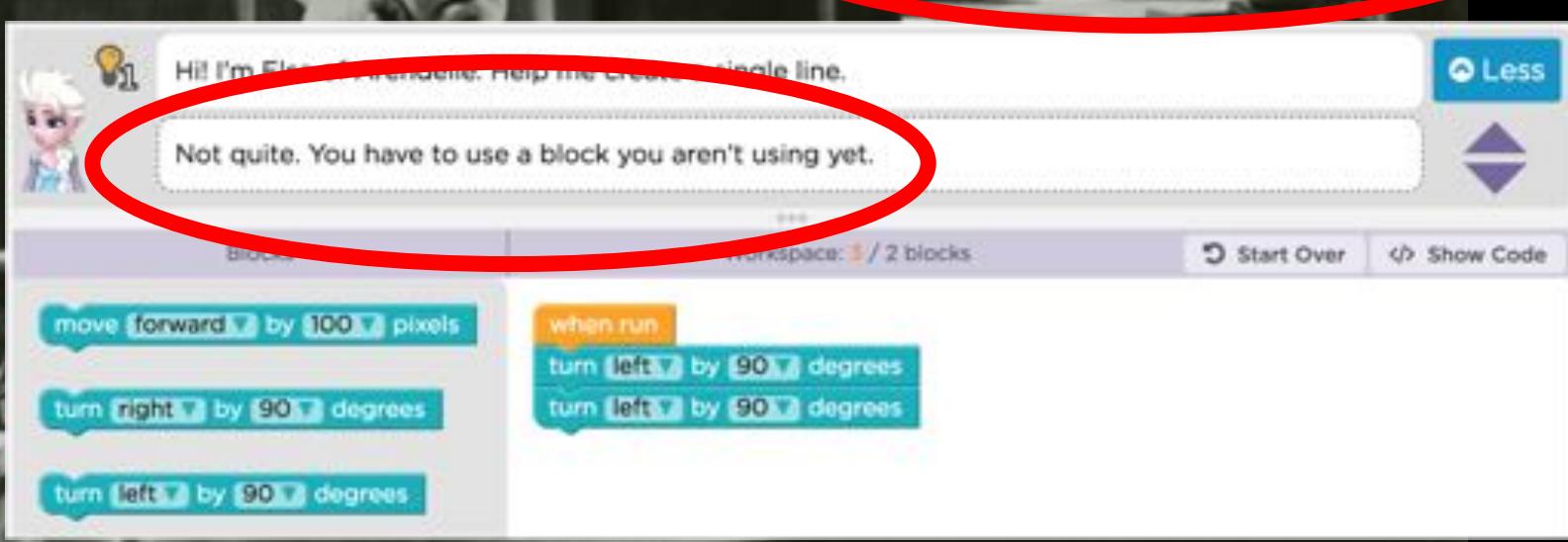
# Online Classes Haven't Solved the Problem

... hard

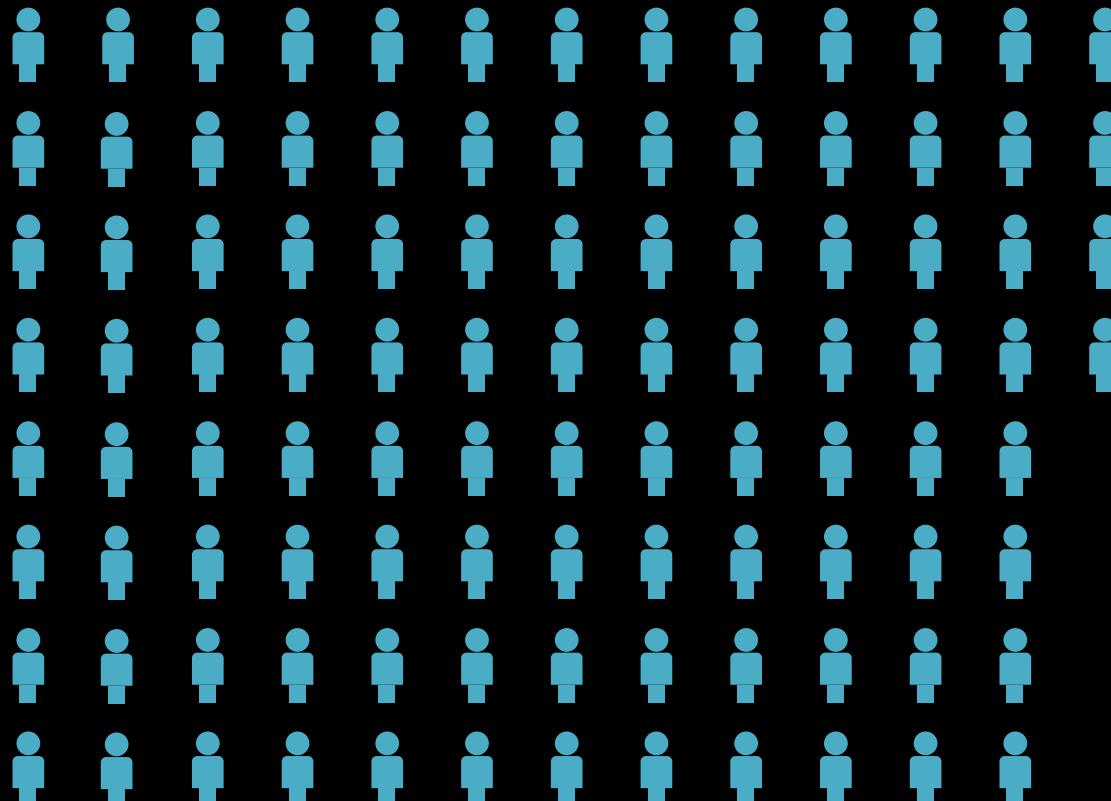
A screenshot of a code editor window titled "Introduction to Python". The file being edited is "script.py". The code contains three lines:

```
1 my_name = "Codecademy"
2 print("Hello and welcome " + my_name + "!")
3
```

The third line is highlighted in red. To the right of the code, the message "File "script.py", line 3" is displayed, followed by a red box containing the error message "SyntaxError: unexpected EOF while parsing".

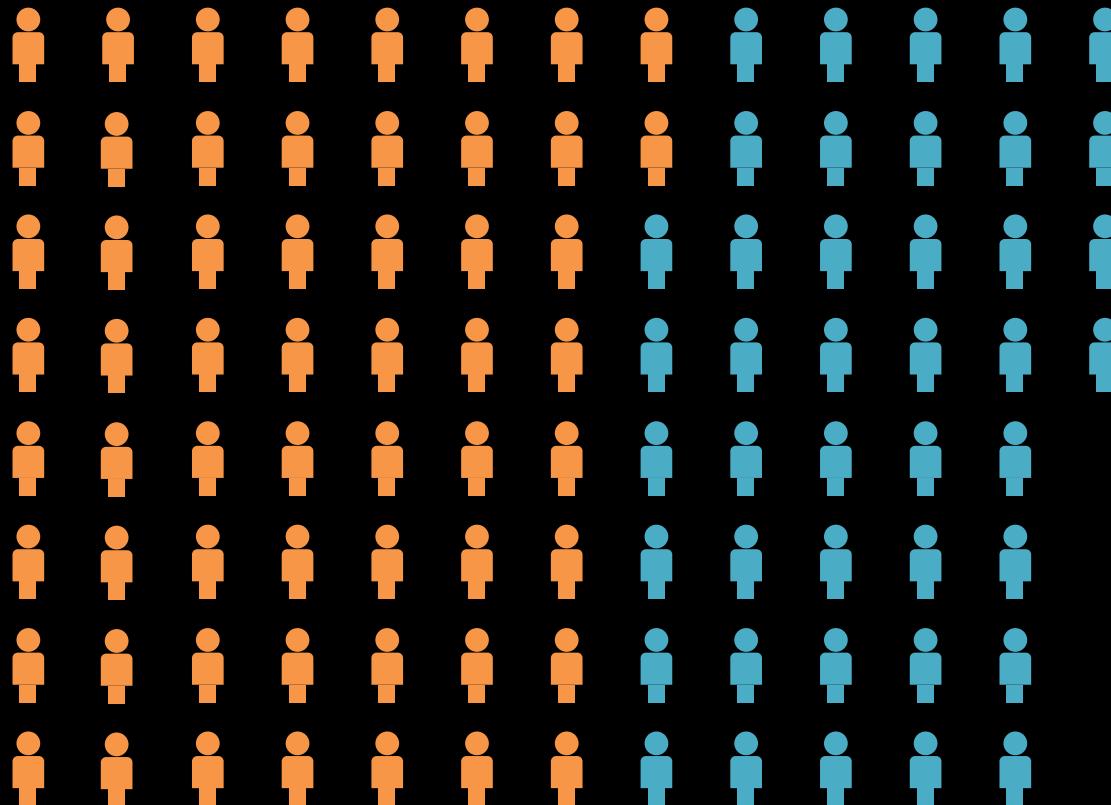


# US K12 Students



= 500,000 learners

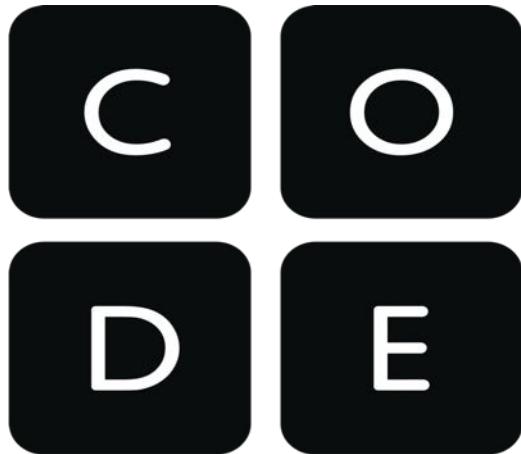
# Code.org



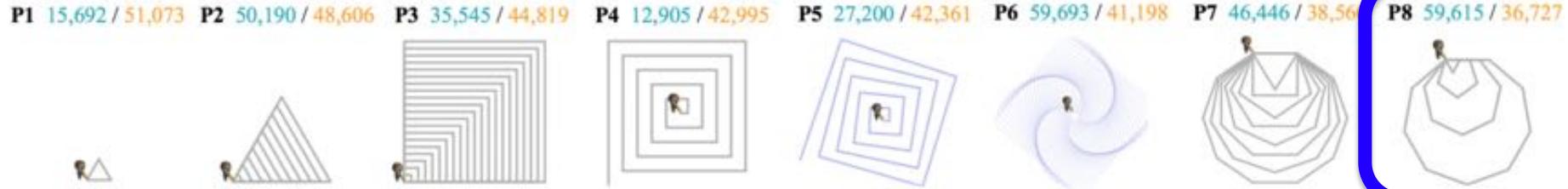
= 500,000 learners



# The Code.org Dataset



- Students learning **nested loops**
- 50k students with **1.5 million submissions** to a curriculum of **8 exercises**.
- **800 human labels** across 2 of the exercises.





Run



P1 15,692 / 51,073 P2 50,190 / 48,606 P3 35,545 / 44,819 P4 12,905 / 42,995 P5 27,200 / 42,361 P6 59,693 / 41,198 P7 46,446 / 38,561 P8 59,615 / 36,727



Let's put it all together!

Using your knowledge of `for` loops and the `counter` variable, create this drawing where each shape has two more sides than the last. Make sure that each side is 10 times as long as the number of sides in the polygon.

Very little of the code has been provided for you.

Blocks

Actions  
Math  
Loops  
Brushes

Workspace: 15 / 16 blocks

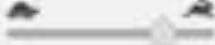
when run

Start Over

Show Code



Run



Let's put it all together!

Using your knowledge of `for` loops and the `counter` variable, create this drawing where each shape has two more sides than the last. Make sure that each side is 10 times as long as the number of sides in the polygon.

Very little of the code has been provided for you.

Blocks

Workspace: 15 / 15 blocks

Start Over

Show Code

Actions  
Math  
Loops  
Brushes

when run

for counter from 3 to 9 count by 2



P1 15,692 / 51,073

P2 50,190 / 48,606

P3 35,545 / 44,819

P4 12,905 / 42,995

P5 27,200 / 42,361

P6 59,693 / 41,198

P7 46,446 / 38,561

P8 59,615 / 36,727





Run



P1 15,692 / 51,073 P2 50,190 / 48,606 P3 35,545 / 44,819 P4 12,905 / 42,995 P5 27,200 / 42,361 P6 59,693 / 41,198 P7 46,446 / 38,561 P8 59,615 / 36,727



Let's put it all together!

Using your knowledge of `for` loops and the `counter` variable, create this drawing where each shape has two more sides than the last. Make sure that each side is 10 times as long as the number of sides in the polygon.

Very little of the code has been provided for you.

Blocks

Workspace: 15 / 15 blocks

Start Over

Show Code

when run

for counter from 3 to 9 count by 2

repeat 2 times

do



Run

P1 15,692 / 51,073 P2 50,190 / 48,606 P3 35,545 / 44,819 P4 12,905 / 42,995 P5 27,200 / 42,361 P6 59,693 / 41,198 P7 46,446 / 38,561 P8 59,615 / 36,727

Let's put it all together!

Using your knowledge of `for` loops and the `counter` variable, create this drawing where each shape has two more sides than the last. Make sure that each side is 10 times as long as the number of sides in the polygon.

Very little of the code has been provided for you.

Blocks

Workspace: 15 / 15 blocks

Start Over

Show Code

Actions  
Math  
Loops  
Brushes

when run

for counter from 3 to 9 count by 2

repeat [counter] times

do move forward [counter] by 10 pixels

turn right [counter] by 360 / counter degrees



# Human Labelled 800

Top Secret

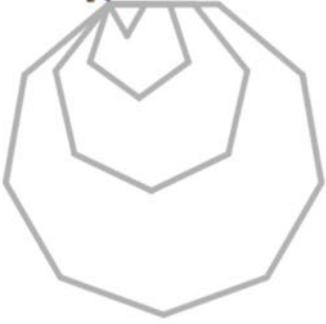
Not Secure stanford.edu/~cplech/restricted/codeDotOrgLabels/#/cjo

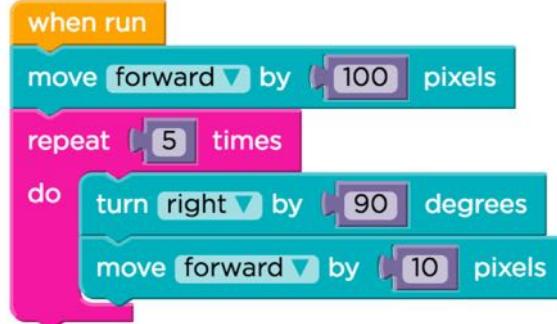
Question Solution

## Instructions

- If there are many moves, focus on the first one
- Random code strategy is for when the student seems to be trying things randomly
- Lookout for students who don't get nesting or pre/post conditions. Often extra blocks in a body is an indication that they don't get that the post of the loop has to match the precondition

## Question





```
when green flag clicked
  move [forward v] by [100 pixels]
  repeat (5)
    do
      turn [right v] by [90 degrees]
      move [forward v] by [10 pixels]
```

## Label Console

Num Done: 8273

### Strategy

Beeper Boundary (most people do this)  
 Triangle Strategy  
 Recursive Strategy

### Looping

Correct use of looping  
 Doesn't use a while  
 Doesn't have correct stop condition  
 Body is missing statements  
 Body has extra statements  
 Body order is incorrect  
 Sets up initial precondition  
 Does not get nesting  
 Loop post condition doesn't match precondition  
 Repetition of bodies

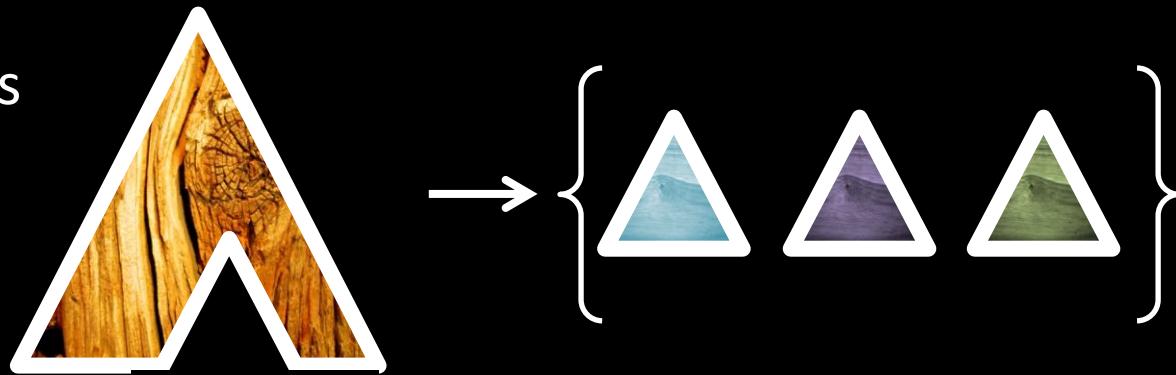
### Cleanup

Record label

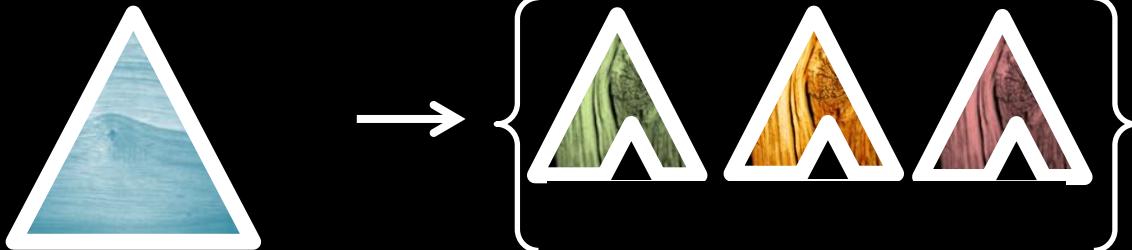
It is a very hard problem

# Terribly Clever Static Analysis

A student's  
program



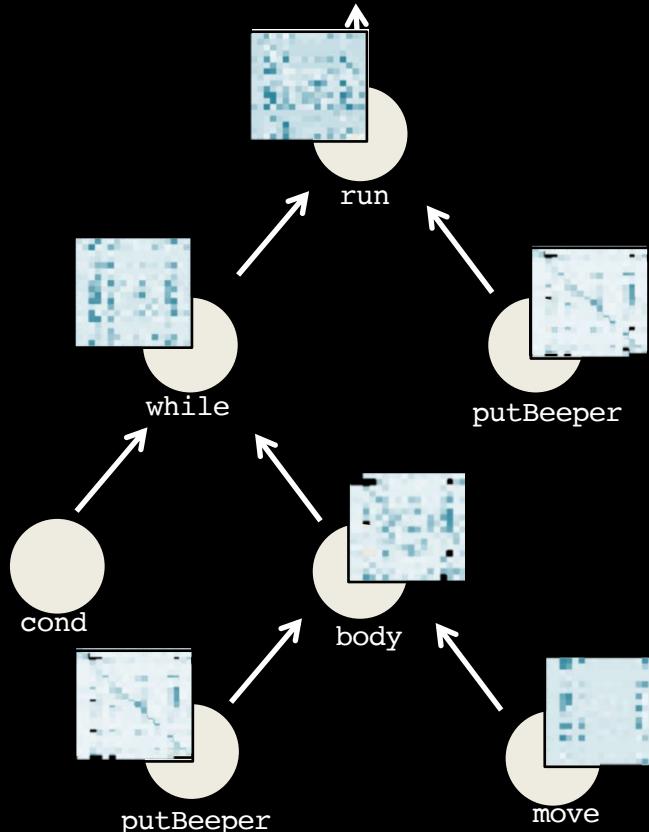
A bug



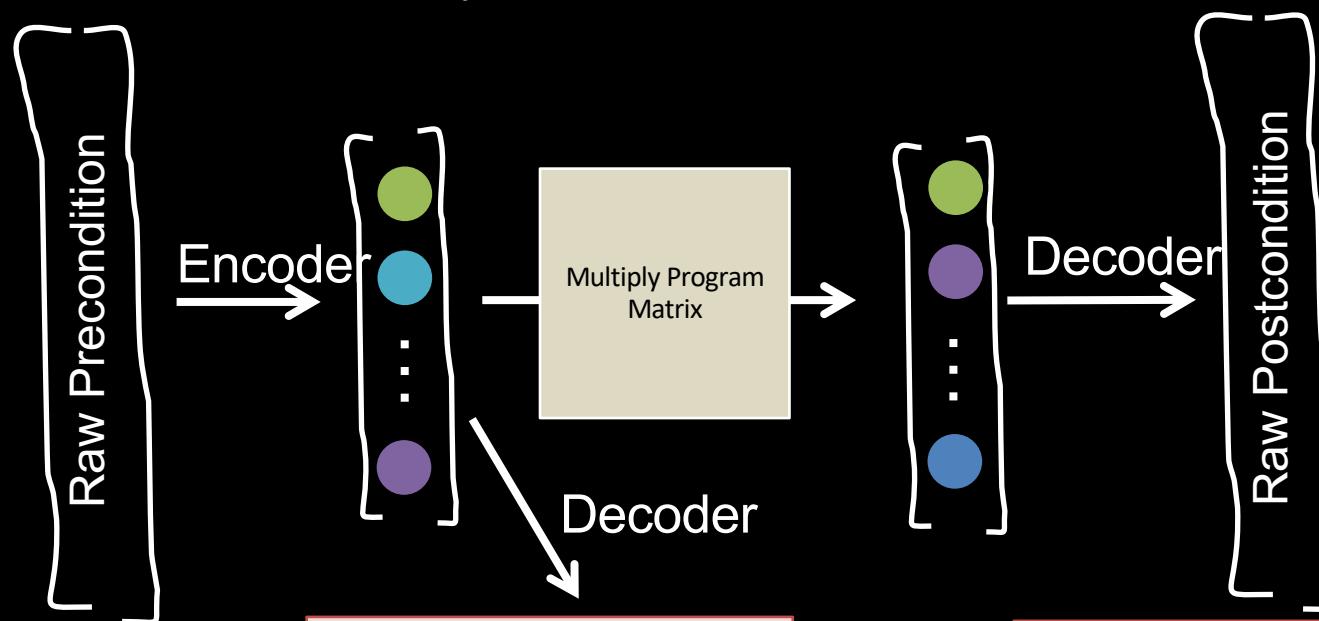
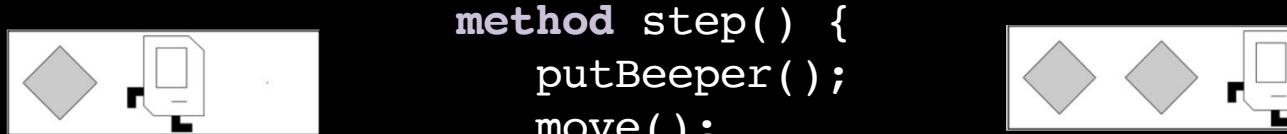
# Encode a Program

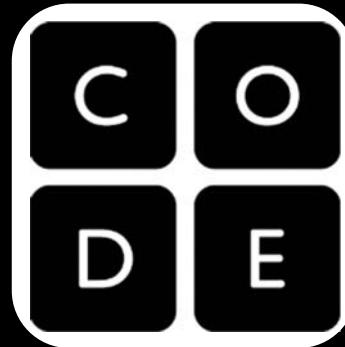
It looks like you have a fencepost error!

```
// User defined method
private void run() {
    while(isClear()){
        putBeeper();
        move();
    }
    putBeeper();
}
```



# Neural Network for Programs



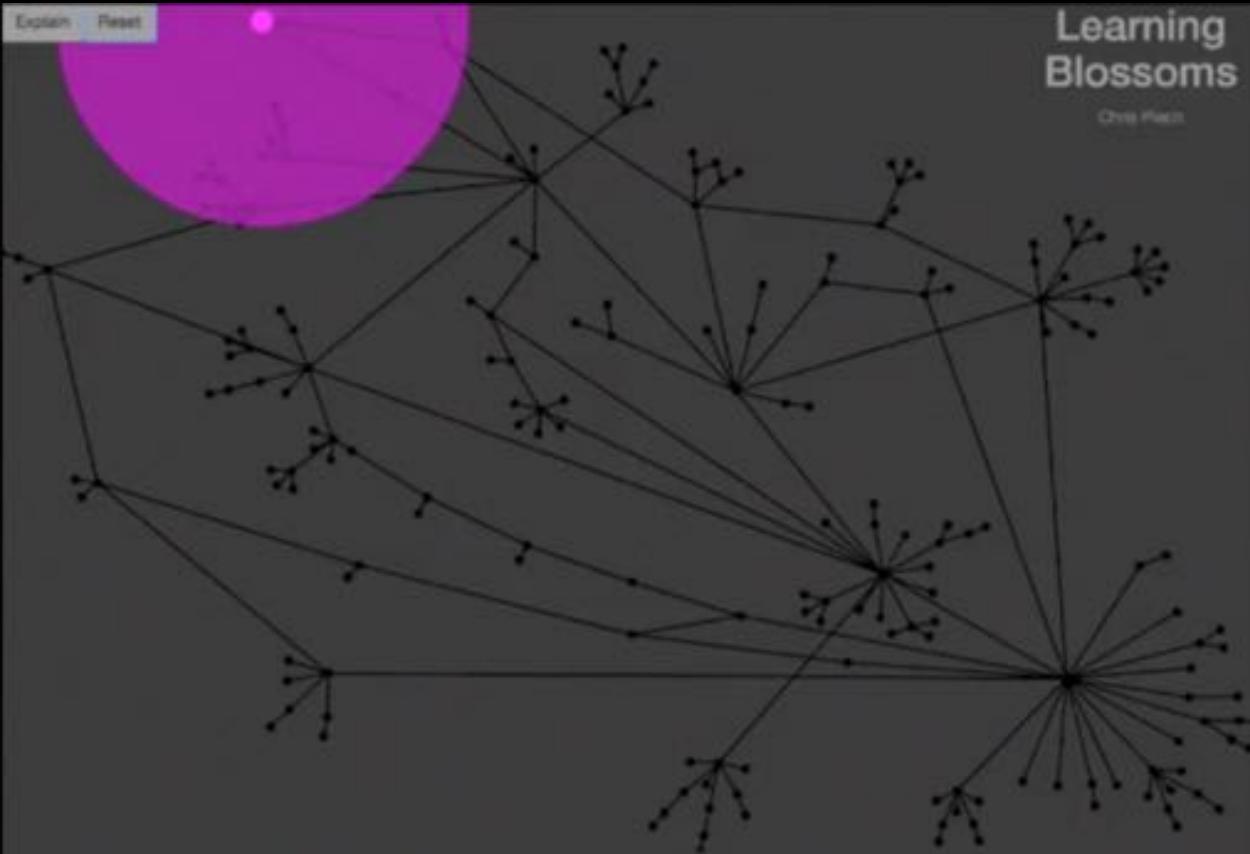


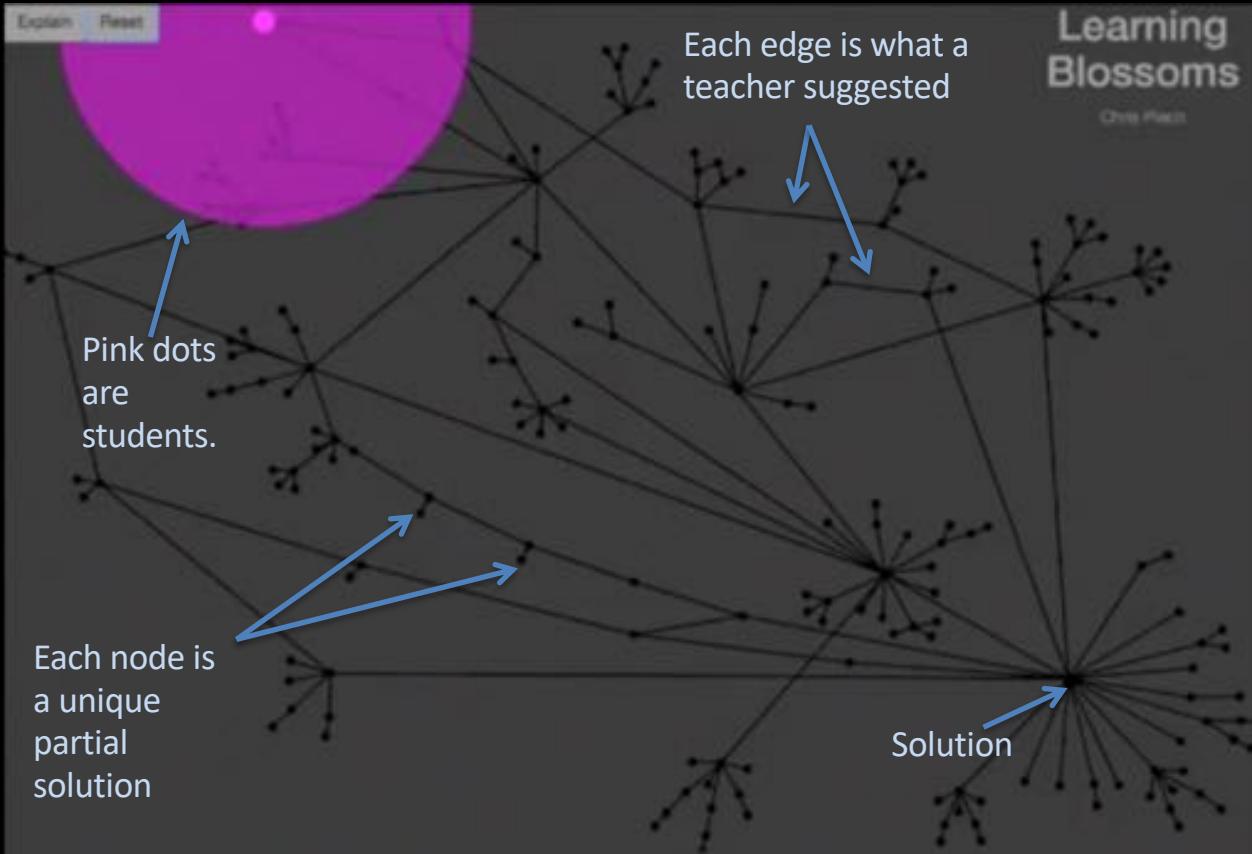
Can we provide feedback  
by dynamic analysis?



• • •







Explain Reset

# Learning Blossoms

Chris Pieno



# The Crowd is Un-wise

Temporal methods tried:

- Shortest path
- Min Time
- Expected Success
- Reinforcement learning
- Most Common Next
- Most Popular Path



```
when run
  move forward
  move forward
  turn left
```

A Scratch script consisting of four blocks: "when run", "move forward", "move forward", and "turn left". The "turn left" block has a counter-clockwise arrow icon.

```
when run
  move forward
  turn left
```

A Scratch script consisting of three blocks: "when run", "move forward", and "turn left". The "turn left" block has a counter-clockwise arrow icon.

```
when run
  move forward
  move forward
  turn right
```

A Scratch script consisting of five blocks: "when run", "move forward", "move forward", "turn right", and "move forward". The "turn right" block has a clockwise arrow icon.

```
when run
  move forward
  move forward
  turn left
  move forward
```

A Scratch script consisting of six blocks: "when run", "move forward", "move forward", "turn left", "move forward", and "move forward". The "turn left" block has a counter-clockwise arrow icon.

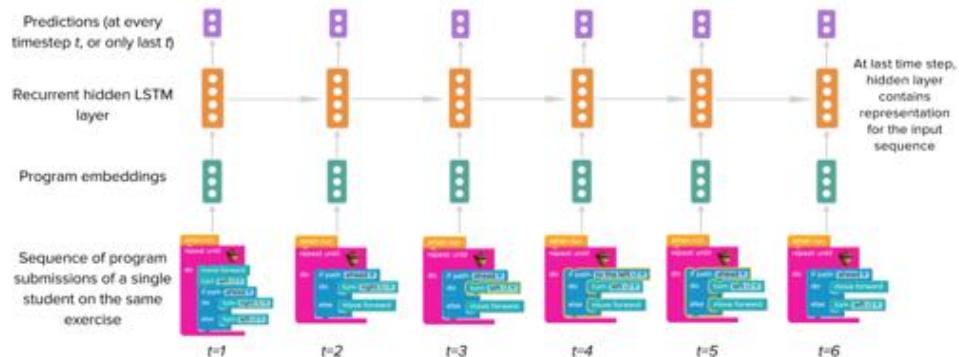
18%

45%

12%

# State of the Art of this well studied problem

SOTA uses RNNs to vectorize programs and classify among K feedback classes.



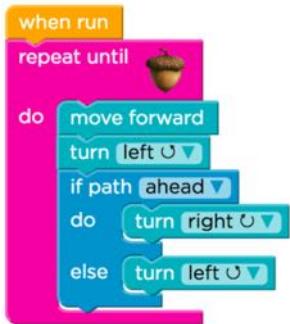
These models are ...

- Far from human accuracy.
- Uninterpretable... why pick the feedback it did?
- Require lots of labeled examples.

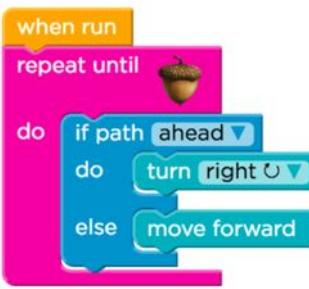


# What Matters for Students?

1. Two compound errors



2. Solves first error



3. Starts reasonable attempt



4. Completes attempt



5. Backtracks



6. Finds solution



# Highly Rates Grit

1. Two compound errors



2. Solves first error



3. Starts reasonable attempt



4. Completes attempt



5. Backtracks



6. Finds solution

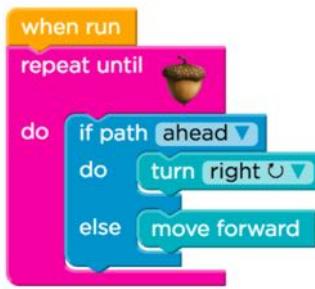


# Highly Rates Grit

1. Two compound errors



2. Solves first error



3. Starts reasonable attempt



4. Completes attempt



5. Backtracks



6. Finds solution

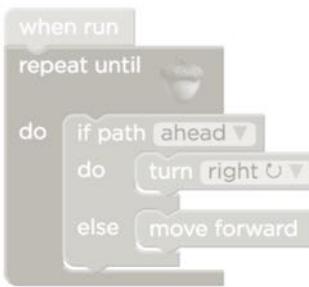


# Highly Rates Grit

1. Two compound errors



2. Solves first error



3. Starts reasonable attempt



4. Completes attempt



5. Backtracks



6. Finds solution

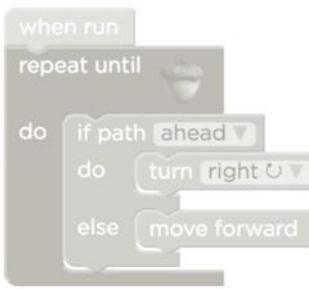


# Highly Rates Grit

1. Two compound errors



2. Solves first error



3. Starts reasonable attempt



4. Completes attempt



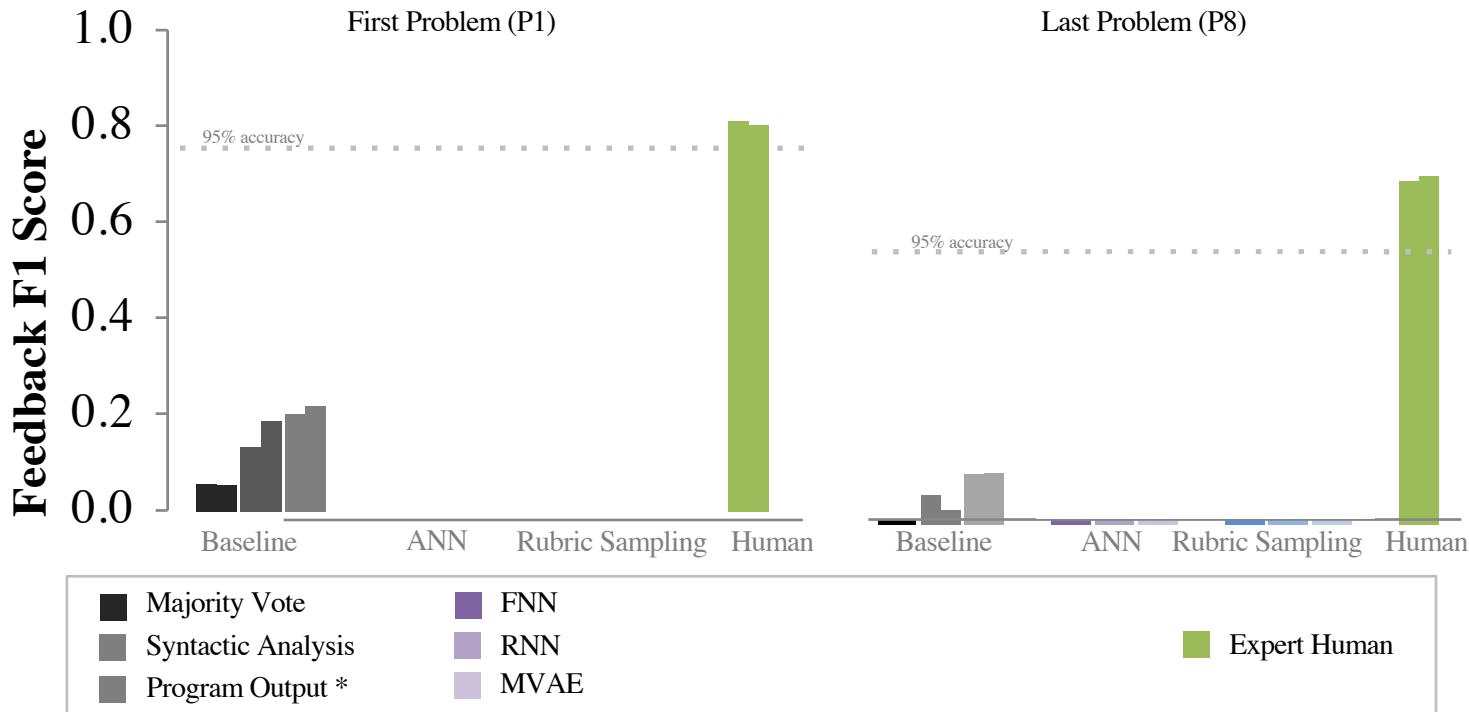
5. Backtracks



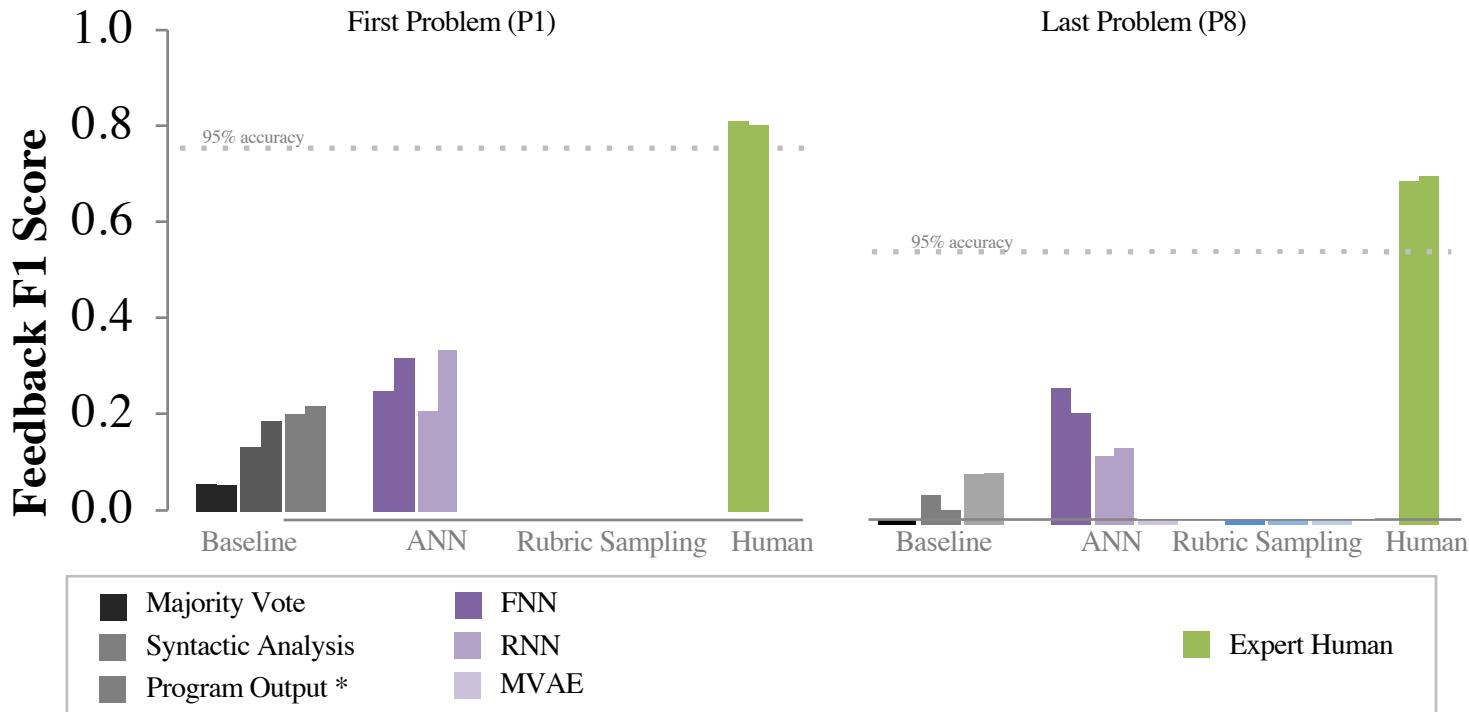
6. Finds solution



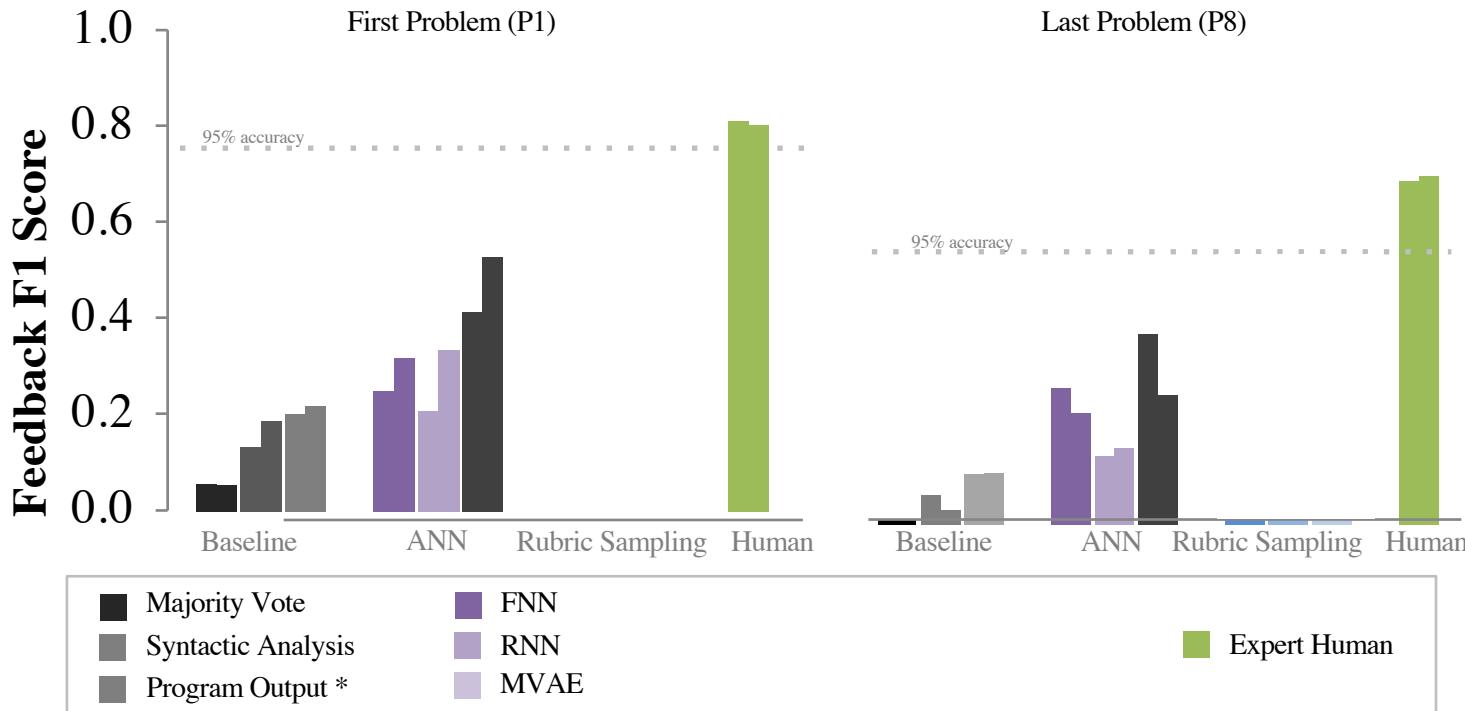
# Baselines Don't Work



# Neural Networks Don't Work



# Inaccurate, Uninterpretable, and Data Hungry



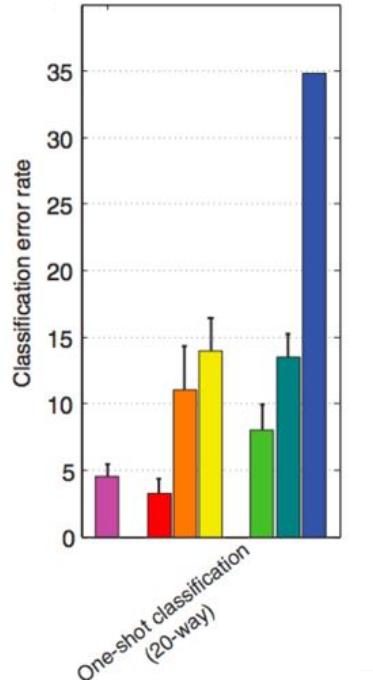
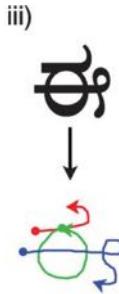
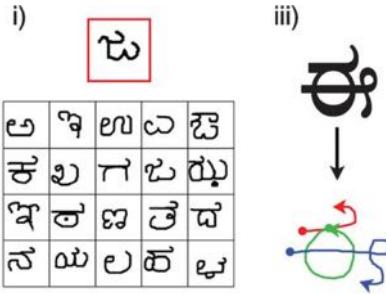
# Zero Shot Feedback Challenge

Real impact needs to be zero shot = *zero historical data + zero expert labels.*



[suspense]

# One Shot Learning with Characters



Bayesian Program Learning models

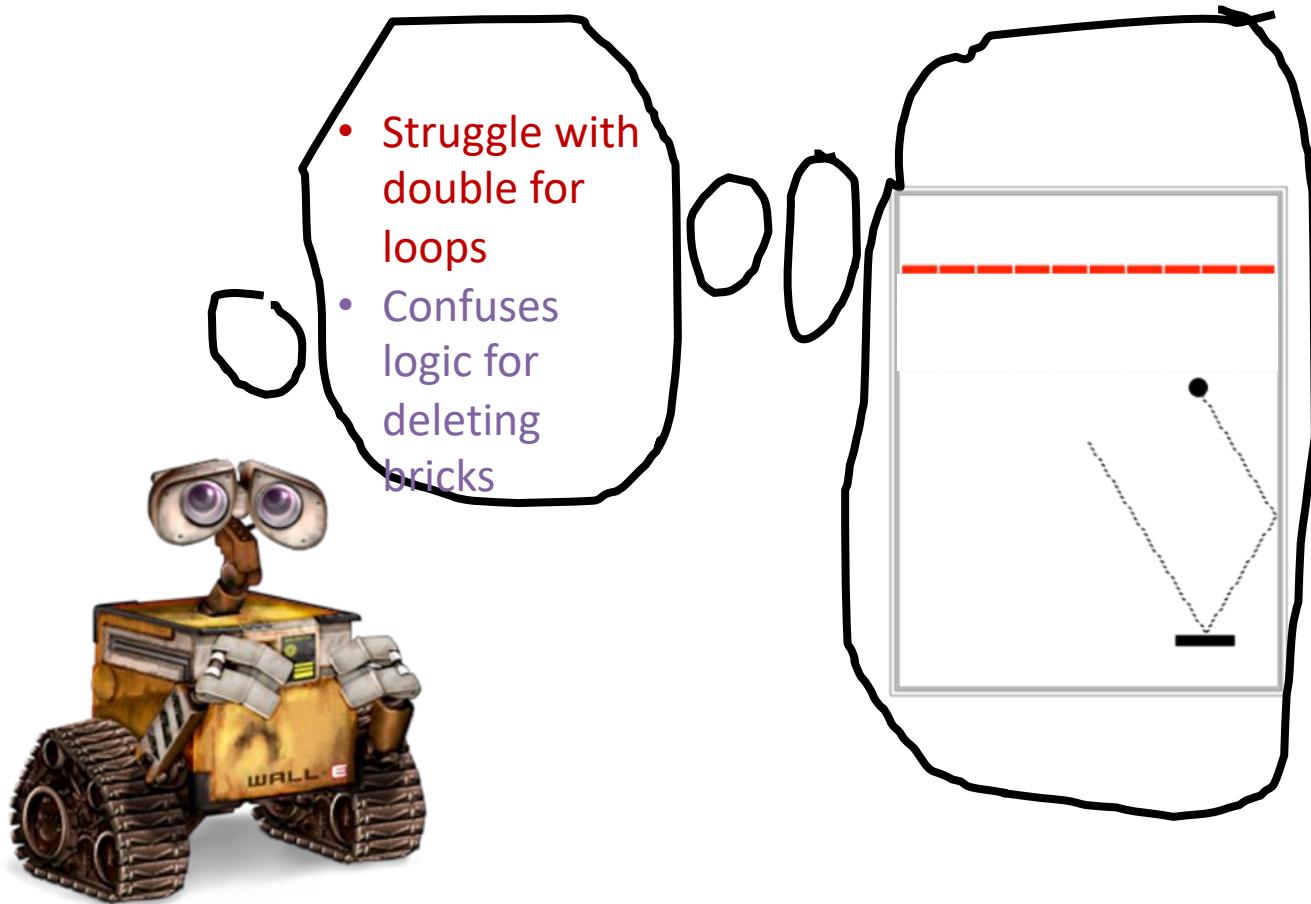
- BPL
- BPL Lesion (no learning-to-learn)
- BPL Lesion (no compositionality)

Deep Learning models

- Deep Siamese Convnet
- Deep Convnet
- Hierarchical Deep



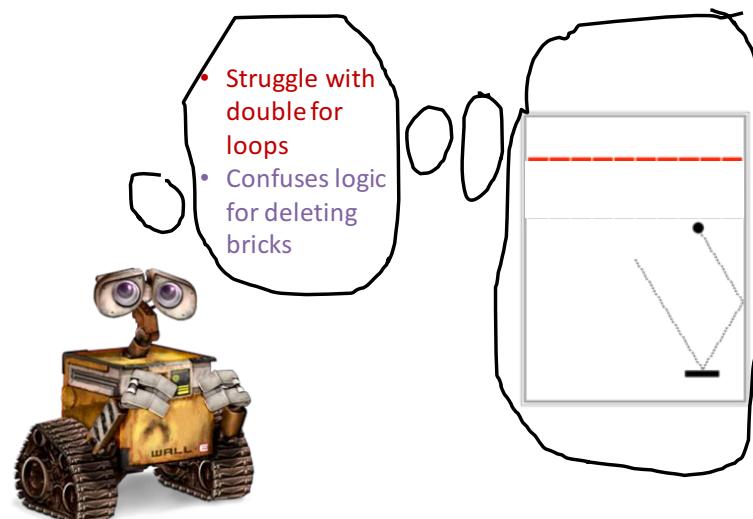
# Imagine Students



# Imagine Students

Priors on knowledge      Misconceptions      Program methods

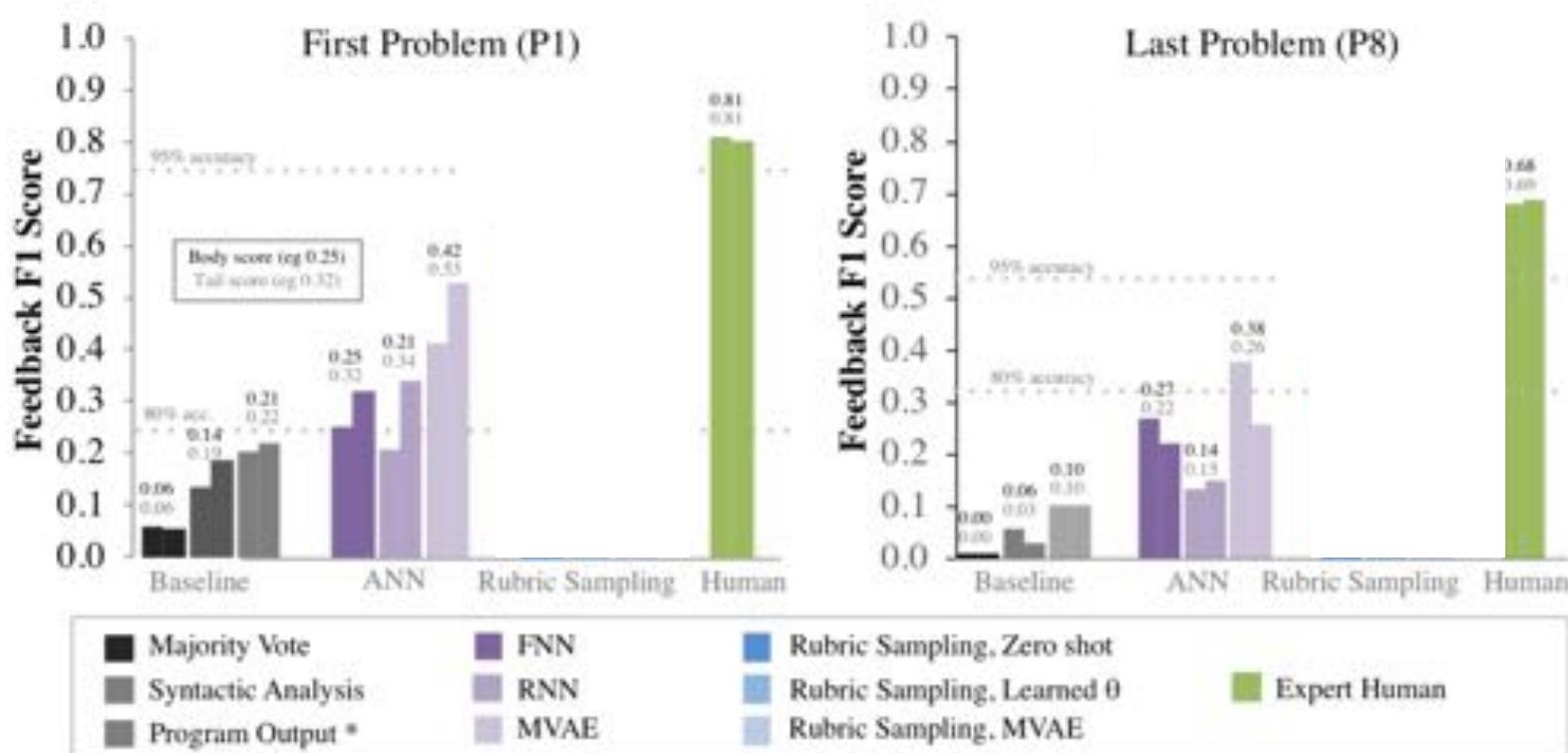
$$P(\psi, \theta^{(1)}, \dots, \theta^{(M)}, I^{(1)}, \dots, I^{(M)}) \\ = P(\psi) \prod_{m=1}^M P(I^{(m)} | \theta^{(m)}) P(\theta^{(m)} | \psi) *$$



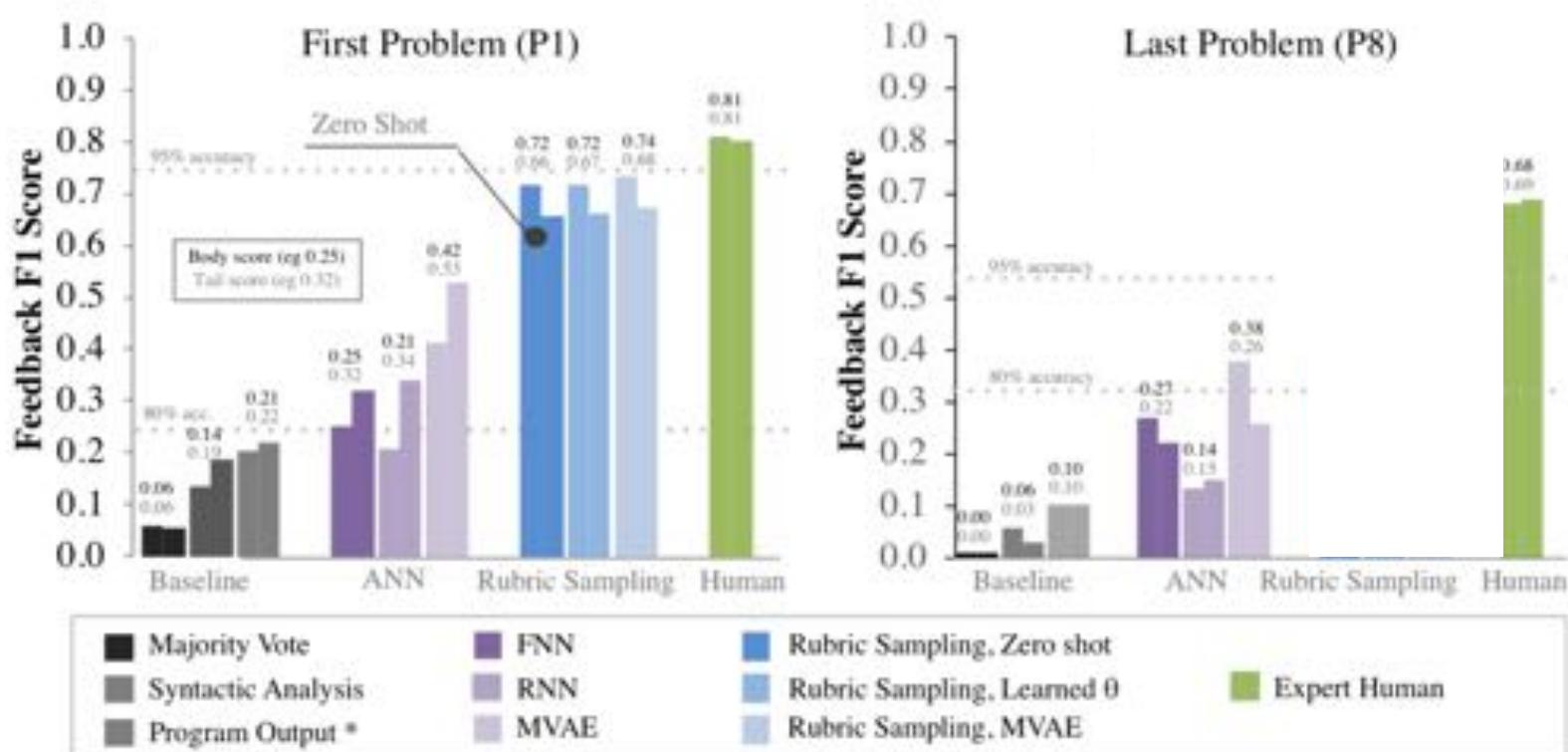
\* Encoded as a  
Bayesian Meta  
Program



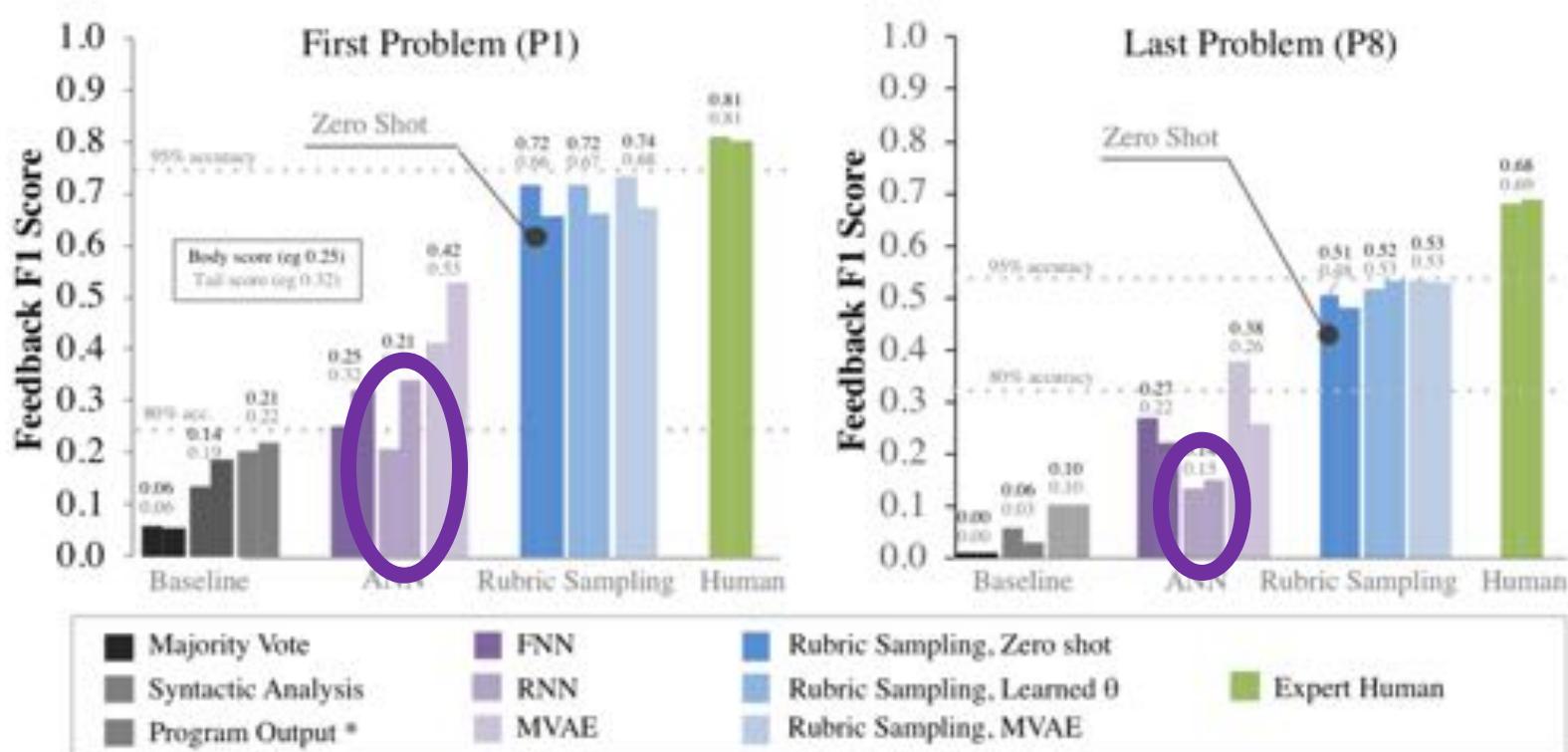
# Setting a new state-of-the-art



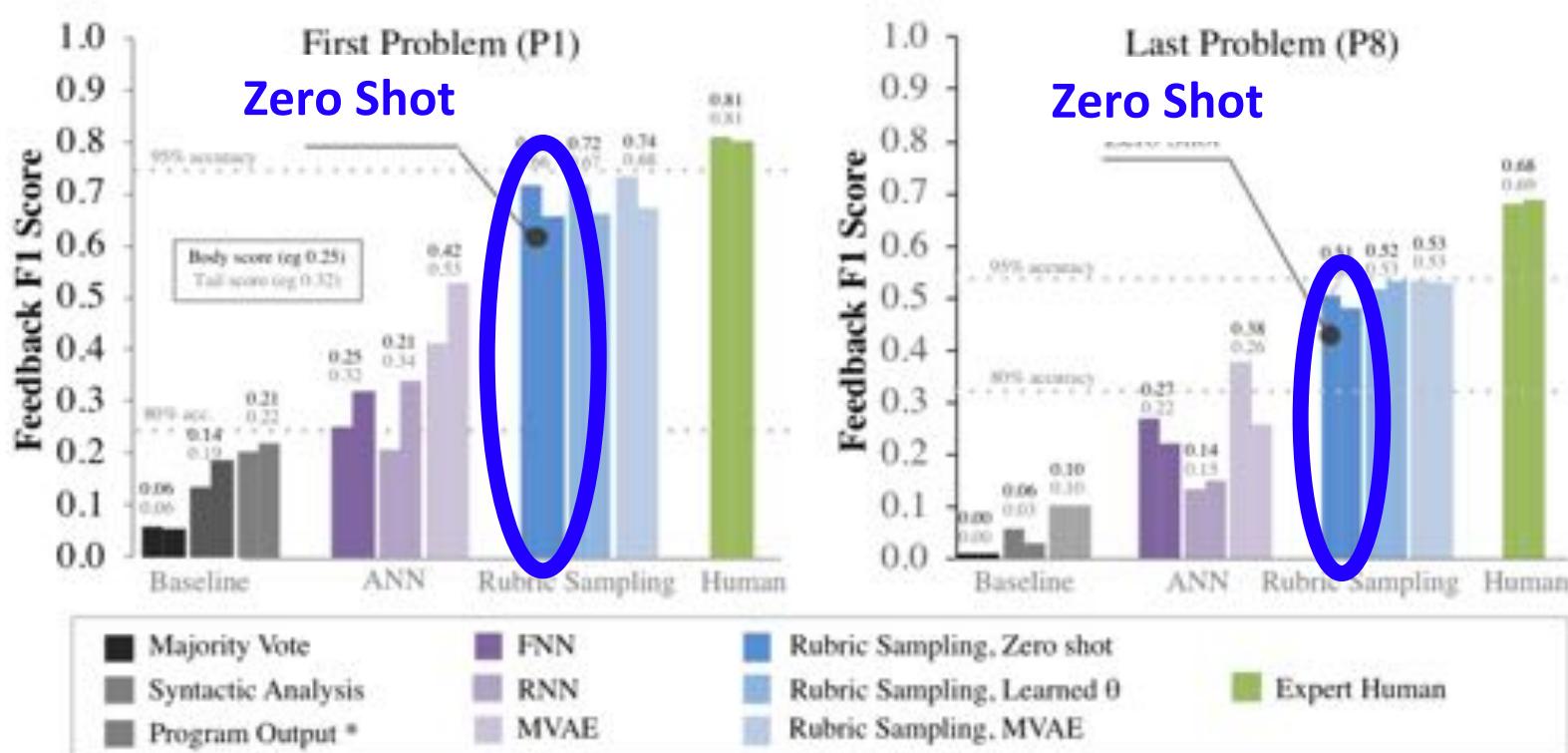
# Setting a new state-of-the-art



# Setting a new state-of-the-art



# Setting a new state-of-the-art



# Rubrics are interpretable

Inference is easy in a context free grammar: use A\*.

We can highlight parts of the code that caused the model to give a particular feedback.

Replace **compiler messages** with **code highlighting**.

## Does not know equilateral is 60

```
Repeat ( 300 ) { Move ( 60 ) TurnLeft ( 130 ) }
TurnRight ( 60 ) Move ( 50 )
Move ( 55 ) TurnLeft ( 75 )
Repeat ( 3 ) { TurnLeft ( 60 ) Move ( 50 ) TurnLeft ( 1200 ) }
Repeat ( 30 ) { TurnRight ( 120 ) Move ( 50 ) }
```

## Random move amount

```
Repeat ( 300 ) { Move ( 60 ) TurnLeft ( 130 ) }
TurnRight ( 60 ) Move ( 50 )
Move ( 55 ) TurnLeft ( 75 )
Repeat ( 3 ) { TurnLeft ( 60 ) Move ( 50 ) TurnLeft ( 1200 ) }
Repeat ( 30 ) { TurnRight ( 120 ) Move ( 50 ) }
```

## Looped

```
Repeat ( 300 ) { Move ( 60 ) TurnLeft ( 130 ) }
TurnRight ( 60 ) Move ( 50 )
Move ( 55 ) TurnLeft ( 75 )
Repeat ( 3 ) { TurnLeft ( 60 ) Move ( 50 ) TurnLeft ( 1200 ) }
Repeat ( 30 ) { TurnRight ( 120 ) Move ( 50 ) }
```

## Correct move direction

```
Repeat ( 300 ) { Move ( 60 ) TurnLeft ( 130 ) }
TurnRight ( 60 ) Move ( 50 )
Move ( 55 ) TurnLeft ( 75 )
Repeat ( 3 ) { TurnLeft ( 60 ) Move ( 50 ) TurnLeft ( 1200 ) }
Repeat ( 30 ) { TurnRight ( 120 ) Move ( 50 ) }
```

## Doesn't loop three times

```
Repeat ( 300 ) { Move ( 60 ) TurnLeft ( 130 ) }
TurnRight ( 60 ) Move ( 50 )
Move ( 55 ) TurnLeft ( 75 )
Repeat ( 3 ) { TurnLeft ( 60 ) Move ( 50 ) TurnLeft ( 1200 ) }
Repeat ( 30 ) { TurnRight ( 120 ) Move ( 50 ) }
```

## Correct turn direction

```
Repeat ( 300 ) { Move ( 60 ) TurnLeft ( 130 ) }
TurnRight ( 60 ) Move ( 50 )
Move ( 55 ) TurnLeft ( 75 )
Repeat ( 3 ) { TurnLeft ( 60 ) Move ( 50 ) TurnLeft ( 1200 ) }
Repeat ( 30 ) { TurnRight ( 120 ) Move ( 50 ) }
```

## Left/Right confusion

```
Repeat ( 300 ) { Move ( 60 ) TurnLeft ( 130 ) }
TurnRight ( 60 ) Move ( 50 )
Move ( 55 ) TurnLeft ( 75 )
Repeat ( 3 ) { TurnLeft ( 60 ) Move ( 50 ) TurnLeft ( 1200 ) }
Repeat ( 30 ) { TurnRight ( 120 ) Move ( 50 ) }
```

Can we **grade** my CS1  
**midterm** more accurately  
than Stanford TAs?

# Chapter 3: Helping Student Teachers



Lisa Yan



Annie Hu

*Presented at SIGCSE 2019*

Students should learn  
the *process* of how to solve  
programming problems.

But we aren't providing  
feedback on *process*.

(We are providing feedback on final  
solutions)

# Pensieve Tool

```
5  Oh 3m
6  Oh 4m
7  Oh 5m
8  Oh 5m
9  Oh 7m
10 Oh 8m
11 Oh 11m
12 Oh 12m
13 Oh 13m
14 Oh 13m
15 Oh 16m
16 Oh 16m
17 Oh 16m (Oh 15m)
18 Oh 16m (8h 38m)
19 Oh 16m (18h 7m)
20 Oh 16m (8h 9m)
21 Oh 16m
22 Oh 20m
23 Oh 21m
24 Oh 30m
25 Oh 36m
26 Oh 36m
27 Oh 36m (Th 25m)
28 Oh 36m (8h 5m)
29 Oh 38m
30 Oh 44m
31 Oh 44m (Oh 45m)
```

File: Pyramid.java

```
/*
 * File: Pyramid.java
 * Name:
 * Section Leader:
 */
/*
 * This file is the starter file for the Pyramid problem.
 * It includes definitions of the constants that match the
 * sample run in the assignment. But you should make sure
 * that changing these values causes the generated display
 * to change accordingly.
 */

import acm.graphics.*;
import acm.program.*;
import java.awt.*;

public class Pyramid extends GraphicsProgram {
    /** Width of each brick in pixels */
    private static final int BRICK_WIDTH = 30;

    /** Height of each brick in pixels */
    private static final int BRICK_HEIGHT = 12;

    /** Number of bricks in the base of the pyramid */
    private static final int BRICKS_IN_BASE = 18;

    public void run() {
        int x = getWidth() / 2 - (BRICKS_IN_BASE / 2) *
               BRICK_WIDTH;
        int y = getHeight() - BRICK_HEIGHT;
        for (int row = BRICKS_IN_BASE; row > 0; row--) {
            drawBricks(row, x, y);
            y += BRICK_HEIGHT;
            x += BRICK_WIDTH / 2;
        }
    }
}
```

Milestone 13: Perfect

A 3D perspective view of a pyramid made of small gray blocks, showing its base and several layers of bricks.

SourceLength

Time into Problem (hours)	Selection (Characters)	Comments (Characters)	Code (Characters)
0.0	100	100	100
0.1	200	150	150
0.2	700	150	150
0.4	500	150	150
0.5	450	150	150
0.6	450	150	150
0.7	450	150	150
0.8	650	650	650
0.85	600	600	600
0.9	350	350	350

Time into Problem (hours)

Selection Comments Code



# Pensieve Tool

File: `Pyramid.java`

```
5  0h 3m
6  0h 4m
7  0h 5m
8  0h 5m
9  0h 7m
10 0h 9m
11 0h 11m
12 0h 12m
13 0h 13m
14 0h 13m
15 0h 16m
16 0h 16m
17 0h 16m (0h 11m)
18 0h 16m (0h 38m)
19 0h 16m (1h 7m)
20 0h 16m (0h 9m)
21 0h 18m
22 0h 20m
23 0h 21m
24 0h 30m
25 0h 36m
26 0h 36m
27 0h 36m (0h 25m)
28 0h 36m (0h 5m)
29 0h 38m
30 0h 44m
31 0h 44m (0h 45m)
```

Milestone 10: Perfect

/\*  
 \* File: Pyramid.java  
 \* Name:  
 \* Section Leader:  
 \* ...  
 \* This file is the starter file for the Pyramid problem.  
 \* It includes definitions of the constants that match the  
 \* sample run in the assignment, but you should make sure  
 \* that changing these values causes the generated display  
 \* to change accordingly.  
 \*/  
  
import acm.graphics.\*;
import acm.program.\*;
import java.awt.\*;  
  
public class Pyramid extends GraphicsProgram {  
  
 /\*\* Width of each brick in pixels \*/
 private static final int BRICK\_WIDTH = 30;  
  
 /\*\* Height of each brick in pixels \*/
 private static final int BRICK\_HEIGHT = 12;  
  
 /\*\* Number of bricks in the base of the pyramid \*/
 private static final int BRICKS\_IN\_BASE = 18;  
  
 public void run() {
 int x = (getWidth() / 2) - (BRICKS\_IN\_BASE / 2) \*
BRICK\_WIDTH;
 int y = getHeight() - BRICK\_HEIGHT;
 for (int row = BRICKS\_IN\_BASE; row > 0; row--) {
 drawBricks(row, x, y);
 y += BRICK\_HEIGHT;
 x += BRICK\_WIDTH / 2;
 }
 }
}

The interface includes a code editor with syntax highlighting, a preview window showing a pyramid made of small squares, and a graph plotting 'Characters' (Y-axis, 0-700) against 'Time into Problem (hours)' (X-axis, 0.0-0.9). The graph tracks three metrics: Selection (orange), Comments (yellow), and Code (green). A vertical red line marks the completion of Milestone 10 at approximately 0.2 hours.

Time into Problem (hours)	Selection (Characters)	Comments (Characters)	Code (Characters)
0.0	100	100	100
0.1	150	150	200
0.2	700	150	200
0.4	150	150	500
0.5	150	150	450
0.6	150	150	450
0.7	150	150	450
0.8	650	650	650
0.85	600	600	680
0.9	350	350	680



# Students work faster and learn more

