Few-Shot Learning in the Real World
Meta-Learning for Giving Feedback to Students
Chelsea Finn
By Braque or Cezanne?
How did you accomplish this?

Through previous experience.
How might you get a machine to accomplish this task?

- Modeling image formation
  - Geometry
  - SIFT features, HOG features + SVM
  - Fine-tuning from ImageNet features
  - Domain adaptation from other painters

Can we explicitly learn priors from previous experience that lead to efficient downstream learning?

Can we learn to learn?
What can meta-learning enable?

Adapting to new objects provided demo resulting policy

Adapting to new molecules

Yu*, Finn*, Xie, Dasari, Zhang, Abbeel, Levine. One-Shot Imitation from Observing Humans. RSS 2018

Adapting to new regions of the world

Nguyen et al. Meta-Learning GNN Initializations for Low-Resource Molecular Property Prediction. 2020

Adapting from simulation to real

Adapt from simulation to real

Song, Yang, Choromanski, Caluwaerts, Gao, Finn, Tan. Rapidly Adaptable Legged Robots via Evolutionary Meta-Learning. IROS 2020
Can we deploy few-shot learning algorithms in the real world?

- the feedback problem
- can we approach the problem using meta-learning?
- can deploy the approach to real students?
- reflections on real-world meta-learning
Few-shot learning to give feedback to student code

Mike Wu, Chris Piech, Noah Goodman, Chelsea Finn
The Feedback Problem

How can we give feedback on a diagnostic?

Submissions: open-ended Python code snippets

Estimated 8+ months of human labor

Piech & Sahami & Zelenski,
Stanford University
This problem isn’t unique to Code-in-Place.

What does feedback look like in MOOCs?
```javascript
main.js

1. getReminder();
2. function getReminder() {
3.     console.log(`Forgot my string markers`);
4. }
5. ```

/home/ccuser/workspace/learn-javascript-functions-functions-function-declarationV3/main.js:4
console.log(Forgot my string markers);

SyntaxError: missing ) after argument list
at createScript (vm.js:53:10)
at Object.runInThisContext (vm.js:95:10)
at Module._compile (module.js:543:28)
at Object.Module._extensions..js (module.js:580:10)
at Module.load (module.js:408:32)
at tryModuleLoad (module.js:447:12)
at Function.Module._load (module.js:439:3)
at Module.runMain (module.js:605:10)
at run (bootstrap_node.js:427:7)
at startup (bootstrap_node.js:151:9)
Welcome to VPython! First, try to make a single square using the turn right block and move forward block. Each side should be 100 pixels long.

Not quite. You have to use a block you aren’t using yet.
Module 1 Review Quiz

3/10 points earned (30%)

You haven't passed yet. You need at least 80% to pass. Review the material and try again! You have 3 attempts every 8 hours.

1. Part of motivation is a feeling of competence. Both Stephen Krashen and Leo Vygotsky believe students work best just a little above their performance level. Stephen Krashen calls this...

2. Vygotsky's theory of the Zone of Proximal Development has students working slightly above their level so they feel comfortable yet challenged. To assist students in this zone, teachers offer support - scaffolding - as they master a skill. Which of the following scenarios is an example of scaffolding?

3. In order to scaffold correctly, a teacher needs to break down difficult concepts by...
The Feedback Challenge

- Train a model to infer student misconceptions, $y$, from the student solution, $x$.

```python
# print 1 to n w/ loop
def my_solution(n):
    print(1)
    print(2)
    print(3)
```

- [x] Incorrect Syntax
- [x] Did not loop
- [ ] Uses “print” fn

Predict!
The Feedback Challenge

Why is this a hard problem for ML?

- **Limited annotation**: grading student work takes expertise and is very time consuming.

  *Example*: annotating 800 blockly codes took **25 hrs**
The Feedback Challenge

Why is this a hard problem for ML?

- **Limited annotation**: grading student work takes expertise and is very time consuming.
- **Long tailed distribution**: students solve the same problem in many ways.

![Graphs showing log count vs. log rank for different programs: (a) Code.org, (b) Liftoff, (c) Pyramid, (d) Power.]
The Feedback Challenge

Why is this a hard problem for ML?

- **Limited annotation**: grading student work takes expertise and is very time consuming.
- **Long tailed distribution**: students solve the same problem in many ways.
- **Changing curriculums**: instructors constantly edit assignments and exams. Student solutions and instructor feedback look different year to year.
Naive methods don’t work

- **Crowdsourcing human labor:** in 2014, Code.org got 1000s of instructors to label 55k student solutions to “artist” problems. But this barely covered the distribution and new solutions were frequent.

https://code.org/hints
Naive methods don’t work

- **Crowdsourcing human labor:** in 2014, Code.org got 1000s of instructors to label 55k student solutions to “artist” problems.

- **Supervised learning:** dataset of a few 1000 examples (at best) + long tail make this really hard.

<table>
<thead>
<tr>
<th>Model</th>
<th>Body F1</th>
<th>Tail F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output CNN [26]</td>
<td>0.10</td>
<td>0.10</td>
</tr>
<tr>
<td>Human</td>
<td>0.68</td>
<td>0.69</td>
</tr>
</tbody>
</table>

Block-based programming

<table>
<thead>
<tr>
<th>Model</th>
<th>Body Acc</th>
<th>Tail Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeuralNet [28]</td>
<td>0.20</td>
<td>0.21</td>
</tr>
<tr>
<td>Human</td>
<td>0.81</td>
<td>0.80</td>
</tr>
</tbody>
</table>

CS106A Graphics programming

<table>
<thead>
<tr>
<th>Model</th>
<th>Avg F1</th>
<th>Tail F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handcrafted [6]</td>
<td>0.58</td>
<td>-</td>
</tr>
<tr>
<td>T&amp;N Best [17]</td>
<td>0.55</td>
<td>-</td>
</tr>
<tr>
<td>Human</td>
<td>0.97</td>
<td>0.90</td>
</tr>
</tbody>
</table>

free response
Can we use prior data & formulate this as a meta-learning problem?

Prior experience

10 years of feedback from Stanford midterms and finals

Meta-test task

Give feedback on new problems with small amount of labeled examples
CS106A Dataset

Contains 4 final exams and 4 midterm exams from CS106.

- Total of 63 questions and 24.8k student solutions.
- Every student solution has feedback via a rubric.
- 10% of questions were annotated by more than 1 TA, which we use to compute human accuracy.

```
Question Description

Write a function `find_diff_chars(string_1, string_2)` that takes in two strings (guaranteed to be of equal length), and returns a list of all the indices where those two strings have different characters.

Student Solution

```python
def find_diff_chars(string_1, string_2):
    indices = []
    for i in range(len(string_1)):
        if string_1[i] == string_2[i]:
            indices.append(i)
    return indices
```
CS106A Dataset

A rubric has several items, each describes a misconception. Each item has several options that an grader may pick to be true.

- More than one option can be true.
- Every problem has its own (possibly unique) rubric items and options.

<table>
<thead>
<tr>
<th>Rubric Item: Problem Setup</th>
<th>Rubric Item: General Deductions</th>
<th>Rubric Item: String Insertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfect</td>
<td>Perfect</td>
<td>Perfect</td>
</tr>
<tr>
<td>Minor issue</td>
<td>1 syntax error</td>
<td>Incorrectly gets character to insert</td>
</tr>
<tr>
<td>Major issue</td>
<td>2 syntax errors</td>
<td>Incorrectly assumes one digit</td>
</tr>
<tr>
<td>No attempt</td>
<td>&gt;2 syntax errors</td>
<td>Doesn’t insert character at correct place</td>
</tr>
<tr>
<td></td>
<td>Variable scoping issue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Null pointer exception</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We treat every rubric option as a task.
- Every task is a binary classification problem!
- Total of 259 tasks ($K = 10$, $Q = 10$)
ProtoTransformer

Support and query sets:

\[ S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{KxN}, y_{KxN})\} \]

\[ Q = \{(x_1^*, y_1^*), (x_2^*, y_2^*), \ldots, (x_{QxN^*}, y_{QxN^*})\} \]
ProtoTransformer

Support and query sets:

$S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{KxN}, y_{KxN})\}$

$Q = \{(x_1^*, y_1^*), (x_2^*, y_2^*), \ldots, (x_{QxN^*}, y_{QxN^*})\}$

Use the support set $S$ to derive a prototype embedding for each class. Try to classify each example in query set $Q$ by distance to each prototype.
ProtoTransformer

Support and query sets:
S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{K\times N}, y_{K\times N})\}
Q = \{(x_1^*, y_1^*), (x_2^*, y_2^*), \ldots, (x_{Q\times N}^*, y_{Q\times N}^*)\}

Use the support set S to derive a prototype embedding for each class. Try to classify each example in query set Q by distance to each prototype.

\[ \mathcal{L}(x^*, y^*) = -\log \frac{\exp\{-\text{dist}(f_\theta(x^*), p_{y^*})/\tau\}}{\sum_{c=1}^N \exp\{-\text{dist}(f_\theta(x^*), p_c)/\tau\}} \]

\( p_c \) is the average embedding over examples in the support set with label c.

**L_2 norm**

\[ \text{temperature} \]
ProtoTransformer

\[
\mathcal{L}(x^*, y^*) = - \log \frac{\exp\{-\text{dist}(f_\theta(x^*), p_{y^*})/\tau\}}{\sum_{c=1}^{N} \exp\{-\text{dist}(f_\theta(x^*), p_c)/\tau\}}
\]

- We assume \( x = (x_1, x_2, \ldots, x_T) \) a sequence of discrete tokens (e.g. code, language).
- The embedding \( f_\theta: X \rightarrow \mathbb{R}^d \) is a RoBERTa model (stacked transformers) where non-padded token embeddings are averaged (single vector).
ProtoTransformer

\[ \mathcal{L}(x^*, y^*) = -\log \frac{\exp\{-\text{dist}(f_\theta(x^*), p_{y^*})/\tau\}}{\sum_{c=1}^{N} \exp\{-\text{dist}(f_\theta(x^*), p_c)/\tau\}} \]

- We assume \( x = (x_1, x_2, \ldots, x_T) \) a sequence of discrete tokens (e.g. code, language).
- The embedding \( f_\theta: X \rightarrow \mathbb{R}^d \) is a RoBERTa model (stacked transformers) where non-padded token embeddings are averaged (single vector).
- Applying this “out-of-the-box” fails. We needed several “tricks” to get past the small data size.

Attention is not all you need. 😐
Trick #1: Task Augmentation

259 is not a lot of tasks. Meta-learning often operates on 1000s of tasks. We apply the “data augmentation” idea to coding tasks!
Trick #2: Side Information

A task is only composed of 10 or 20 examples, leaving a lot of ambiguity.

Suppose we have “side information” $z = (z_1, z_2, ..., z_T)$ about each task: **rubric option name** and **question text**. How do we add this side information into our embedding function $f_\theta$?
Trick #2: Side Information

A task is only composed of 10 or 20 examples, leaving a lot of ambiguity.

Suppose we have “side information” $z = (z_1, z_2, ..., z_T)$ about each task: rubric option name and question text.

Prepend side information as a first token.
Trick #3: Code Pre-training

Can we utilize large unlabeled datasets of code to help the model learn a good prior for code?

In practice, we initialize the embedding network from pretrain weights and finetune top M layers.
### Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Held-out rubric</th>
<th>Held-out exam</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AP</td>
<td>P@50</td>
</tr>
<tr>
<td>ProtoTransformer</td>
<td>84.2</td>
<td>85.2</td>
</tr>
<tr>
<td></td>
<td>(±1.7)</td>
<td>(±3.8)</td>
</tr>
<tr>
<td>Supervised</td>
<td>66.9</td>
<td>59.1</td>
</tr>
<tr>
<td>Human TA</td>
<td>82.5</td>
<td>–</td>
</tr>
</tbody>
</table>

*Room to grow!*
Ablations

Legend

- task aug
- preprocessing
- architecture
- side info
- pretraining
- meta algo
- supervised
- best
Embeddings

Visualize “prototype” embeddings to interpret student ability and question quality.

Color shows the numeric grade (not used by model ever) given to student (darker is lower).
Can we deploy few-shot learning algorithms in the real world?

- the feedback problem

- can we approach the problem using meta-learning?

- can **deploy** the approach to **real students**?

- **reflections** on real-world meta-learning
Can we deploy this to Code-in-Place?

May 10th, 2021: Students took diagnostic.
Syntax error here would prevent unit tests from being useful.

Algorithm uses attention to highlight where in the code the error comes from.

AI generated feedback

Students evaluate the feedback

Your Solution

```python
def main():
    # TODO write your solution here
    height=input("Enter your height in meters: ")
    if height<1.6:
        print("Below minimum astronaut height")
    if height>1.9:
        print("Above maximum astronaut height")
    if height>=1.6 and height<=1.9:
        print("Correct height to be an astronaut")

if __name__ == "__main__":
    main()
```

designed by Alan Cheng & Chris Piech
Blind, randomized trial evaluated by real students

Humans gave feedback ~1k answers.
AI gave feedback on the remaining ~15k.

~2k could be auto-graded and were not included in analysis.

Humans gave good feedback.
ML model gave slightly better feedback.

Average holistic rating of usefulness by students was $4.6 \pm 0.018$ out of 5.
No signs of bias by demographics
Can we deploy few-shot learning algorithms in the real world?

- the feedback problem

- can we approach the problem using meta-learning?

- can deploy the approach to real students?

- reflections on real-world meta-learning
A first for education

First successful deployment of ML-driven **feedback** to open ended student work

* to the best of our knowledge

A first for ML

First successful deployment of prototypical networks in live application.
What was hard and different?

1. **Limited** meta-training tasks.
   - task augmentation can help
   - regularization may help
   
   Also see:
   Bansal et al. SMLMT ‘20
   Murty et al. DRECA ‘21

   Also see: Yao et al. MLTI ‘21

2. Where does the **support set** come from?
   - active learning? expert-designed support sets?

3. Can the model **defer harder examples** for the instructor?
   - calibration, selective classification

4. **Domain shift** between meta-training & deployment.
   
   Also see: Koh*, Sagawa* WILDS ’21
Want to learn more about meta-learning?

Stanford CS330: Deep Multi-Task and Meta Learning

cs330.stanford.edu

All lecture videos online!