Non-Parametric Few-Shot Learning

CS 330

Course Reminders

Homework 1 due **tonight**.

Homework 2 released, due Mon 10/24.

Project mentors to be assigned this week.

Project proposal due next Weds 10/19.

(graded lightly, for your benefit)

Plan for Today

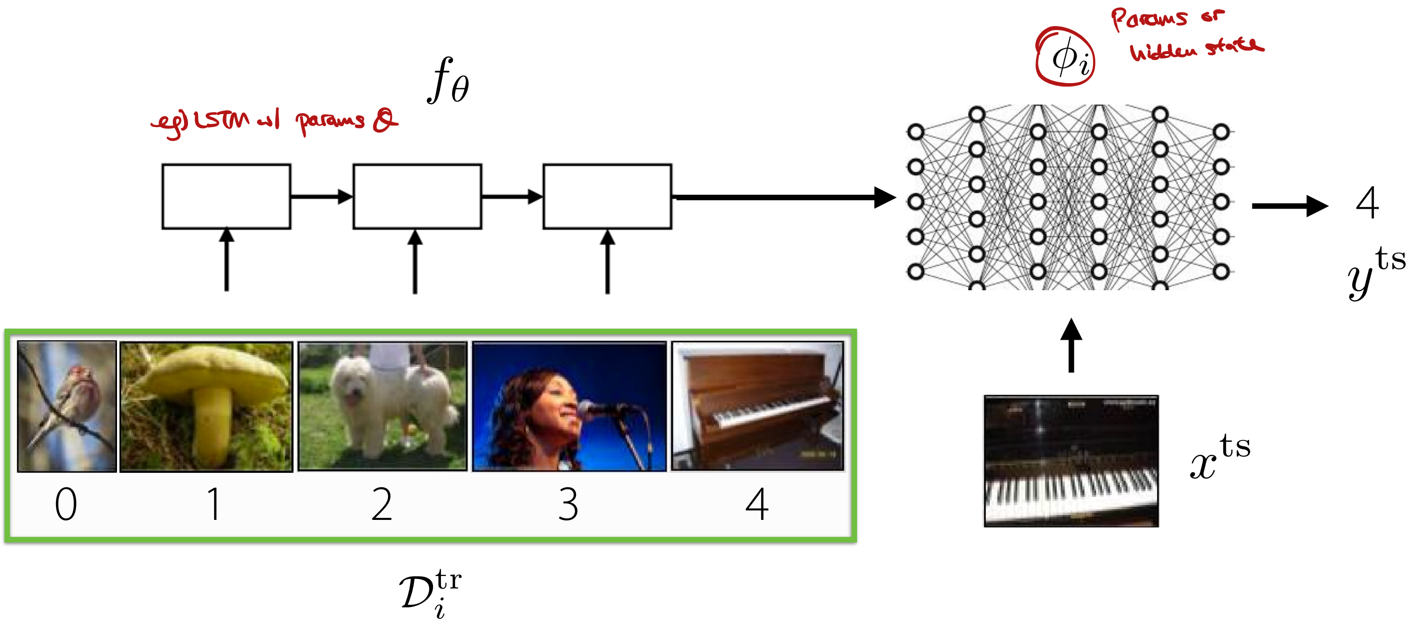
Non-Parametric Few-Shot Learning
- Siamese networks, matching networks, prototypical networks
- Case study of few-shot student feedback generation

Properties of Meta-Learning Algorithms
- Comparison of approaches

Example Meta-Learning Applications
- Imitation learning, drug discovery, motion prediction, language generation

Goals for by the end of lecture:
- Basics of non-parametric few-shot learning techniques (& how to implement)
- Trade-offs between black-box, optimization-based, and non-parametric meta-learning
- Familiarity with applied formulations of meta-learning

Recap: Black-Box Meta-Learning



Key idea: parametrize learner as a neural network

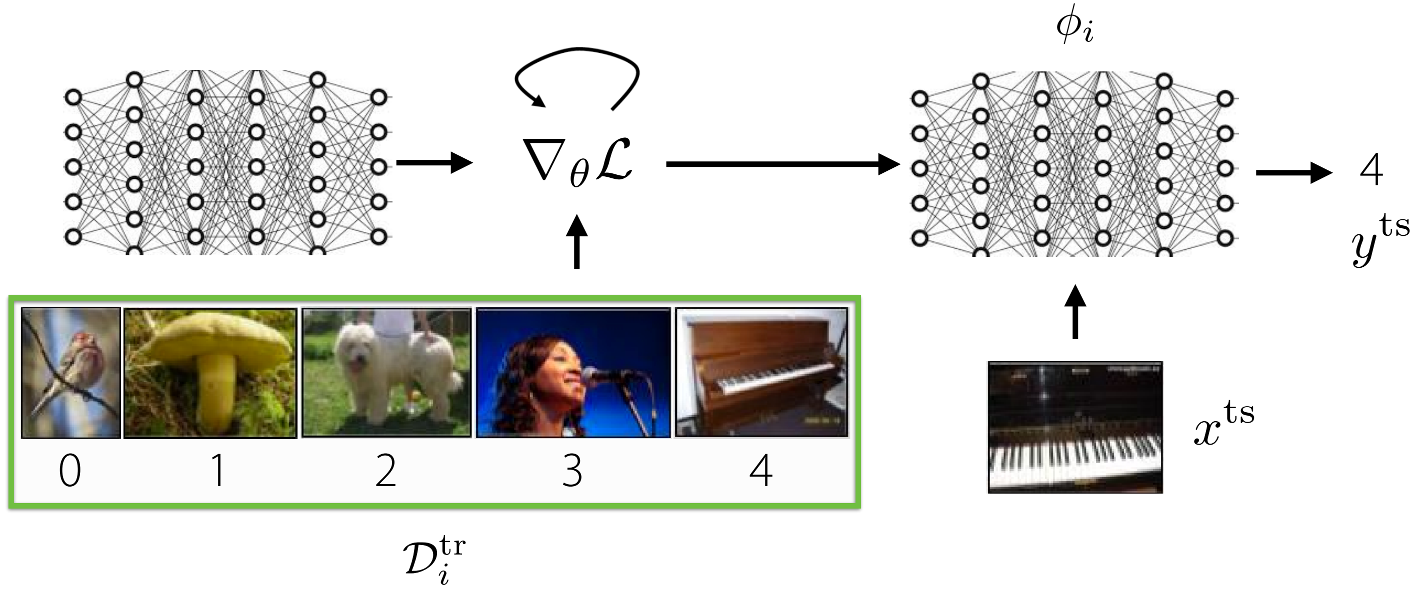
can represent diff. learning procedures

\+ **expressive**

Sensitive to hparams

\- **challenging optimization** problem

Recap: Optimization-Based Meta-Learning



Key idea: embed optimization inside the inner learning process

Use algos like nearest neighbors

+ **structure** of **optimization** embedded into meta-learner

- typically requires **second-order optimization**

Hessian-vector product helps

Computationally heavy.

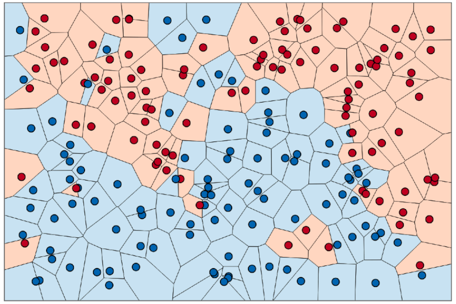
Today: Can we embed a learning procedure *without* a second-order optimization?

6/7 나중에 다시 봐야한다..
지금은 이해 x.

So far: Learning parametric models.

Computationally efficient
∵ small no. of comparisons
to make for the algo to work.

In low data regimes, **non-parametric** methods are simple, work well.



During **meta-test time**: few-shot learning <-> low data regime

During **meta-training**: still want to be parametric

→ we still have a large no. of tasks
→ we want good representations from data

Can we use **parametric meta-learners** that produce effective **non-parametric learners**?

the fuck?

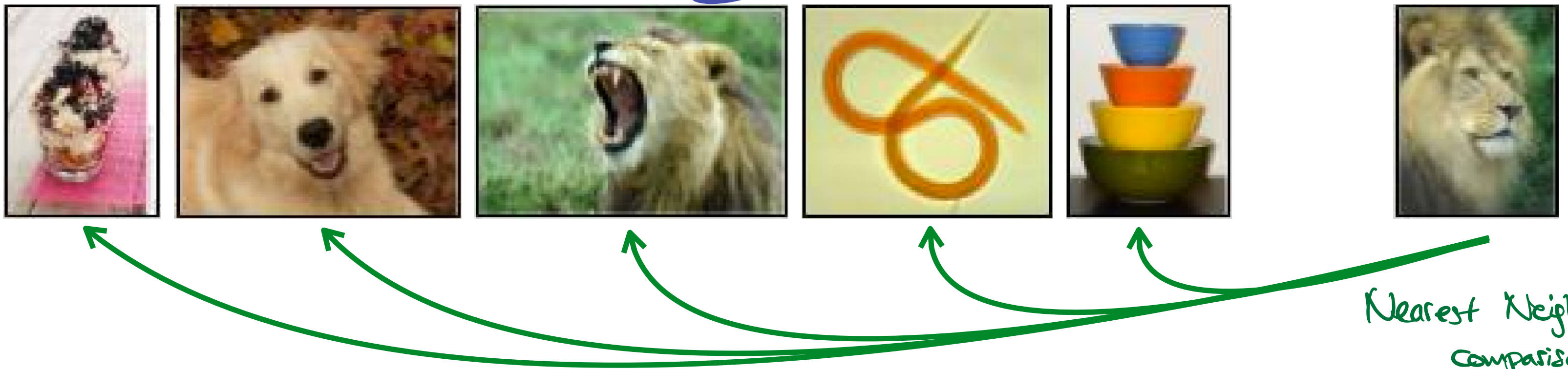
Note: some of these methods precede parametric approaches

Non-parametric methods

Key Idea: Use non-parametric learner.

$\{x_i^{tr}, y_i^{tr}\}$

training data $\mathcal{D}_i^{\mathrm{tr}}$ test datapoint x^{ts}



Nearest Neighbors-type comparisons

Compare test image with training images

In what space do you compare? With what distance metric?

ℓ_2 distance in pixel space?

Bad idea...

In what space do you compare? With what distance metric?

ℓ_2 distance in pixel space?

This is much closer to ⊛ in L2 space

⇒ Counterintuitive results... not a good metric

Zhang et al. (arXiv 1801.03924)

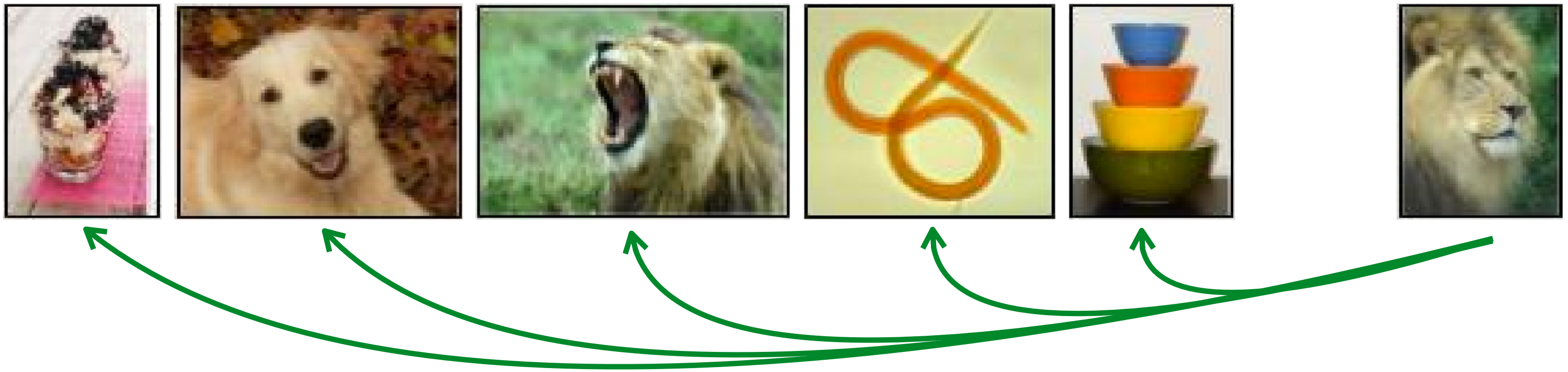
Non-parametric methods

Q: If you're choosing a metric, isn't that a parameter of the method?

A: Choice of distance func. is parametric, yes.
Once you embed, though, everything else is non-parametric.

Key Idea: Use non-parametric learner.

training data $\mathcal{D}_i^{\text{tr}}$ test datapoint x^{ts}



Compare test image with training images

In what space do you compare? With what distance metric?

~~ℓ_2 distance in pixel space?~~

Question: What distance metric would you use instead?

Idea: Learn to compare using meta-training data

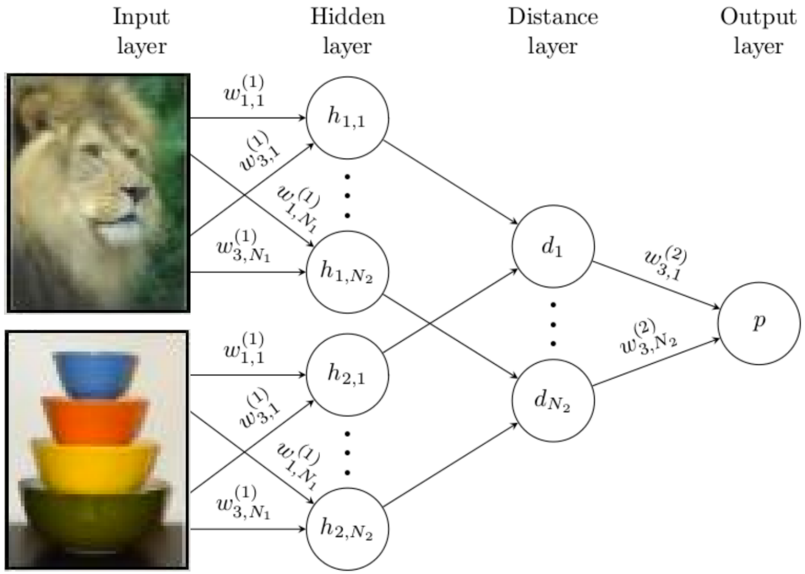
- Embeddings from a trained neural network
- Some metric learned from the training data
- Extracted key points
- Cosine similarity in feat. space

Non-parametric methods

· 2 NNs have the same parameters (shared params)
· Train the NN to output whether the two input images have the same class or not

Key Idea: Use non-parametric learner.

train Siamese network to predict whether or not two images are the same class



label

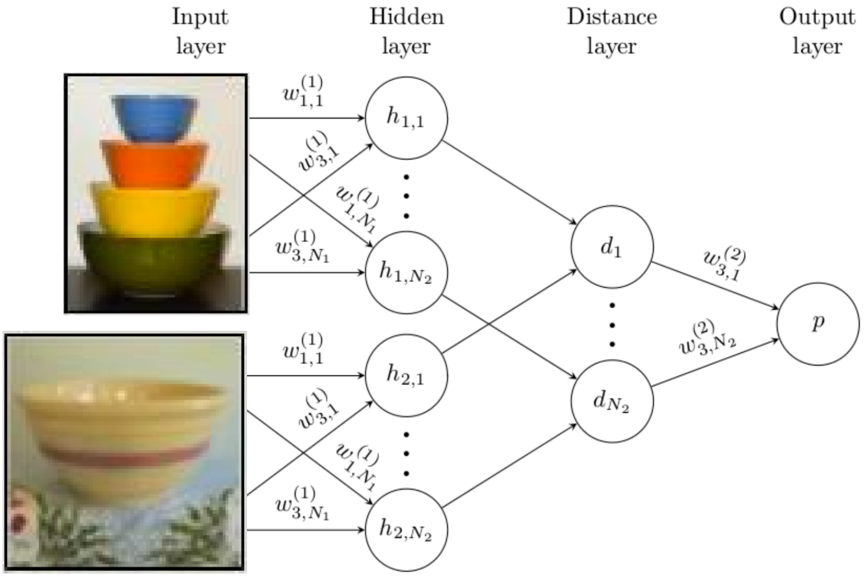
0 → diff. class

Koch et al., ICML '15

Non-parametric methods

Key Idea: Use non-parametric learner.

train Siamese network to predict whether or not two images are the same class



label

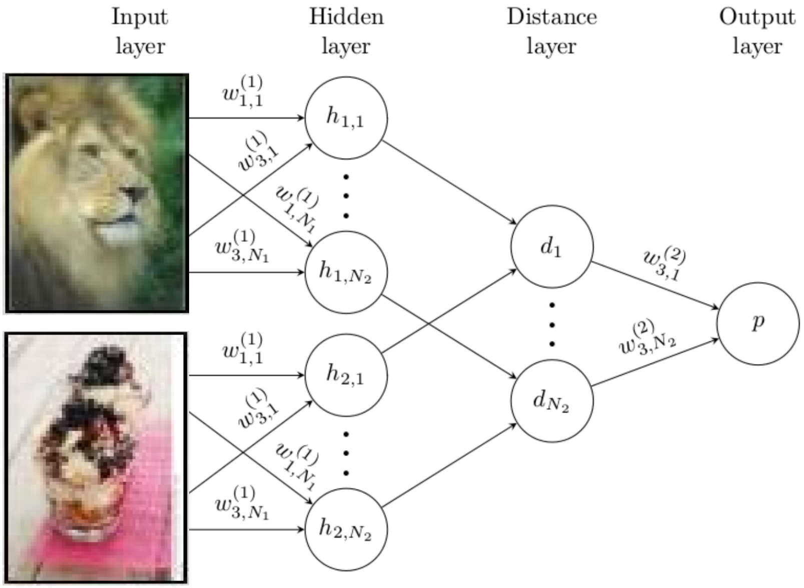
1 → Same class!

Koch et al., ICML '15

Non-parametric methods

Key Idea: Use non-parametric learner.

train Siamese network to predict whether or not two images are the same class



label

0

Koch et al., ICML '15

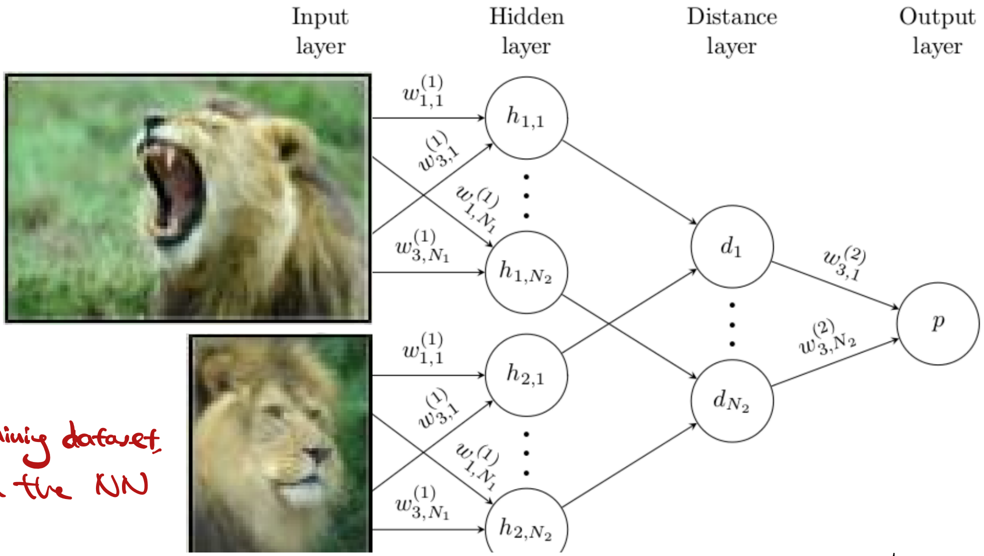
Non-parametric methods

· Need one example per class in the meta training dataset (same as before)

Key Idea: Use non-parametric learner.

train Siamese network to predict whether or not two images are the same class

Once this is trained, use it as a similarity/ distance metric.

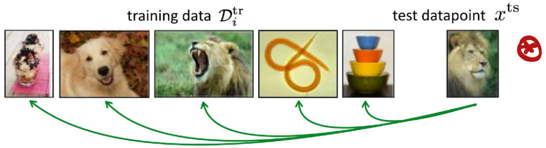


label

1

If N×K samples on the training dataset
N×K forward passes on the NN

Ask which of these images is the closest to ⊛



Meta-test time: compare image $\mathbf{x}_{\text{test}}$ to each image in $\mathcal{D}_j^{\text{tr}}$

mismatch btw the two.

Meta-training: Binary classification
Meta-test: N-way classification

Can we **match** meta-train & meta-test?

⊛ See notes next slide

Koch et al., ICML '15

Can use diff. encoders for train & test examples.

⇒ Choice of their architecture... can put it thru same NN.

Non-parametric methods

$$\hat{y}^{ts} = \sum_{x_k, y_k \in \mathcal{D}^{tr}} \mathbb{1}[f_\theta(x^{ts}, x_k) \geq 0.5]\, y_k$$

Indicator transform ⇒ Hard to differentiate

test example

each training example

not exactly, but close enough. → @ meta test time

↳ Do sth like this to backprop to θ, instead of doing binary clf. ⇒ can match what happens in meta train & test times.

Key Idea: Use non-parametric learner.

Can we **match** meta-train & meta-test?

Nearest neighbors in learned embedding space

↳ Restricted to classification tasks!

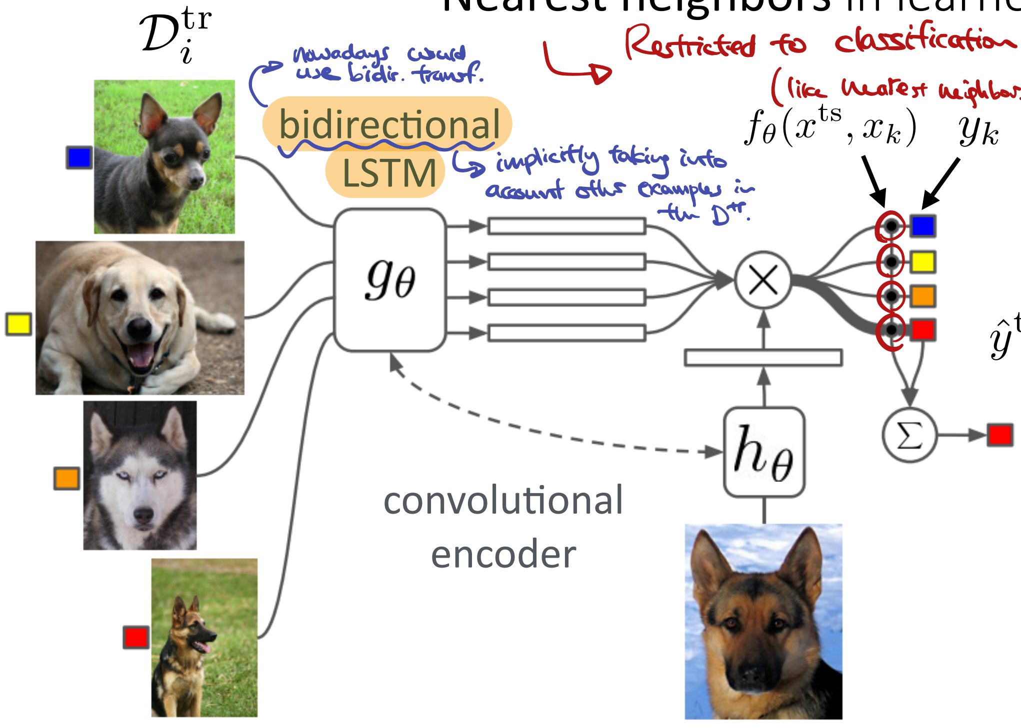
(like nearest neighbors)

Do something softer instead, like:

$$\hat{y}^{ts} = \sum_{x_k, y_k \in \mathcal{D}^{tr}} f_\theta(x^{ts}, x_k)\, y_k$$

black dots in diag.

⇒ Can differentiate (backprop).



$\mathcal{D}_i^{ts}$

1. Sample Task (3 char)
2. Sample 2 imgs / char → D_i^{tr}, D_i^{ts}
3. $\min_\theta \left[y_{ts} \log \hat{y}_{ts} + (1-y_{ts}) \log(1-\hat{y}_{ts}) \right]$

$$\hat{y}^{ts} = \sum_{x_k, y_k \in \mathcal{D}^{tr}} f_\theta(x^{ts}, x_k) y_k$$

Trained **end-to-end.**

Meta-train & meta-test time **match.**

↳ much better performance than Siamese Networks.

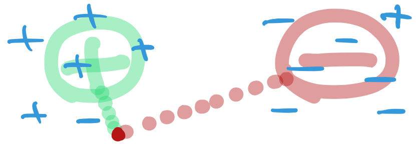
Vinyals et al. Matching Networks, NeurIPS ‘16

All feed-forward; no grad descent ⇒ v. lightweight.
↳ @ meta-test time

Non-parametric methods

#3: Instead of computing parameters for the task, directly make predictions for the test examples.

Key Idea: Use non-parametric learner.



⇒ Nearest Neighbor will predict (−), ignoring the info from its general vicinity. → predicting independently

⇒ why aggregating info is preferred. → aggregating
↳ eg. calculating distance from "prototypes"

General Algorithm:

~~Black box approach~~ Non-parametric approach (matching networks)

1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\mathrm{tr}}, \mathcal{D}_i^{\mathrm{test}}$ from $\mathcal{D}_i$
3. ~~Compute $\phi_i \leftarrow f_\theta(\mathcal{D}_i^{\mathrm{tr}})$~~ Compute $\hat{y}^{\mathrm{ts}} = \sum_{x_k, y_k \in \mathcal{D}^{\mathrm{tr}}} f_\theta(x^{\mathrm{ts}}, x_k) y_k$
4. ~~Update θ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\mathrm{test}})$~~ Update θ using $\nabla_\theta \mathcal{L}(\hat{y}^{\mathrm{ts}}, y^{\mathrm{ts}})$

(Parameters ϕ integrated out, hence non-parametric)

Matching networks will perform comparisons **independently**

What if >1 shot?
- calculate avg. embedding
- voting scheme → all votes are cast independently.

Can we **aggregate class information** to create a **prototypical embedding**?

Non-parametric methods

- Original paper uses Euclidean or cosine distances. → similar to matching networks
- In practice, you can also learn a distance function

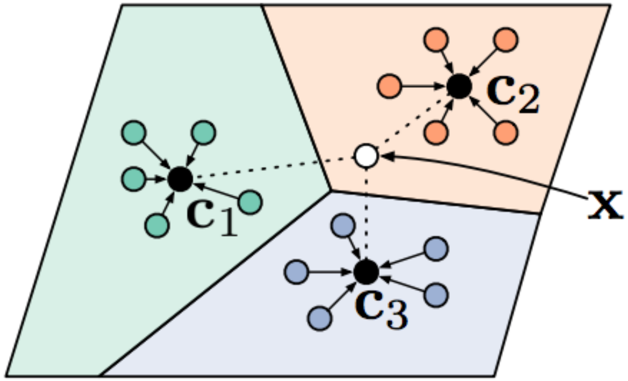
⇒ use $\text{softmax}(-d_{\theta_d}(c_n, x_{ts}))$ & do $\min_{\theta_f, \theta_d}$

→ θ_d: distance function model

→ θ_f: original model

where $c_n = \frac{1}{K} \sum_{x,y \in \mathcal{D}_i^{tr}} \mathbb{1}(y=n) f_{\theta}(x)$

Key Idea: Use non-parametric learner.



avg over the embeddings for the given class n

$$\mathbf{c}_n = \frac{1}{K} \sum_{(x,y) \in \mathcal{D}_i^{\text{tr}}} \mathbb{1}(y = n) f_\theta(x)$$

→ $\text{softmax}(-d(f_\theta(x), c_n))$

$$p_\theta(y = n|x) = \frac{\exp(-d(f_\theta(x), \mathbf{c}_n))}{\sum_{n'} \exp(-d(f_\theta(x), \mathbf{c}_{n'}))}$$

Typo!

d: Euclidean, or cosine distance

Snell et al. Prototypical Networks, NeurIPS '17

Non-parametric methods

So far: Siamese networks, matching networks, prototypical networks

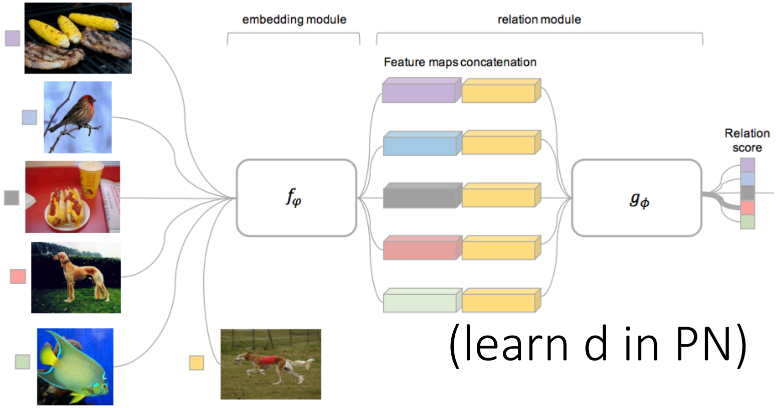
Embed, then nearest neighbors.

Challenge

What if you need to reason about more complex relationships between datapoints?

→ multiple prototypes per class. eg) breed of cat that looks like dog

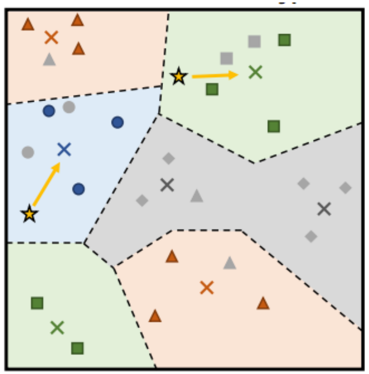
Idea: Learn non-linear relation module on embeddings



(learn d in PN)

Sung et al. Relation Net

Idea: Learn infinite mixture of prototypes.

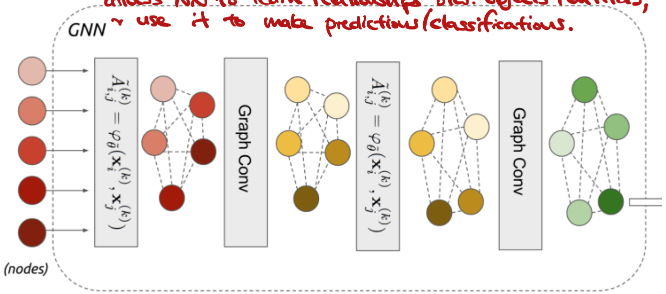


adaptive number of clusters

Allen et al. IMP, ICML '19

Idea: Perform message passing on embeddings

↳ process data that's represented as a graph.
allows NN to learn relationships btw. objects (entities,
& use it to make predictions/classifications.



Garcia & Bruna, GNN

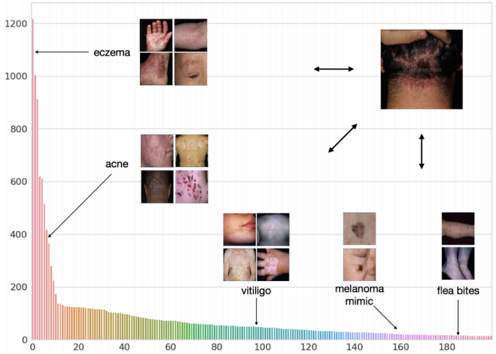
Previous Year's Case Study

Prototypical Clustering Networks for Dermatological Image Classification

Viraj Prabhu [*,1] Anitha Kannan[3] Murali Ravuri[3] Manish Chablani[3]
David Sontag[2] Xavier Amatriain[3]
[1]Georgia Tech [2]MIT [3]Curai
virajp@gatech.edu dsontag@mit.edu {anitha, murali, manish, xavier}@curai.com

Machine Learning for Healthcare Conference 2019

Link: https://arxiv.org/abs/1811.03066



Challenges:
- hard to get data
- data is long-tailed

Goal: Acquire accurate classifier on all classes

This Year's Case Study

Meta-Learning Student Feedback to 16,000 Solutions

- **Problem:** Providing high-quality feedback to students at scale is a challenging problem in computer science education. Traditional approaches to feedback, such as human grading, are time-consuming and expensive. Automated approaches, such as rule-based systems, can be brittle and difficult to scale.
- **Approach:** The authors propose a meta-learning approach to providing student feedback. Their approach, called ProtoTransformer, is a few-shot learning algorithm that can learn to give feedback to student code from just a few examples. ProtoTransformer first learns a representation of student code that captures both the code's syntax and semantics. It then uses this representation to learn a set of feedback templates. When given a new piece of student code, ProtoTransformer uses its learned representation to find the most similar feedback template. It then generates feedback based on the template.
- **Evaluation:** The authors evaluated ProtoTransformer on a dataset of student code from an introductory computer science course. They found that ProtoTransformer was able to generate high-quality feedback that was comparable to feedback generated by human instructors. ProtoTransformer was also able to generate feedback at scale, providing feedback to 16,000 student solutions in a matter of hours.
- **Conclusion:** The authors argue that ProtoTransformer is a promising new approach to providing student feedback. It is able to generate high-quality feedback at scale, which could help to improve the quality of computer science education.

Mike Wu, Chris Piech, and Chelsea Finn

July 20, 2021

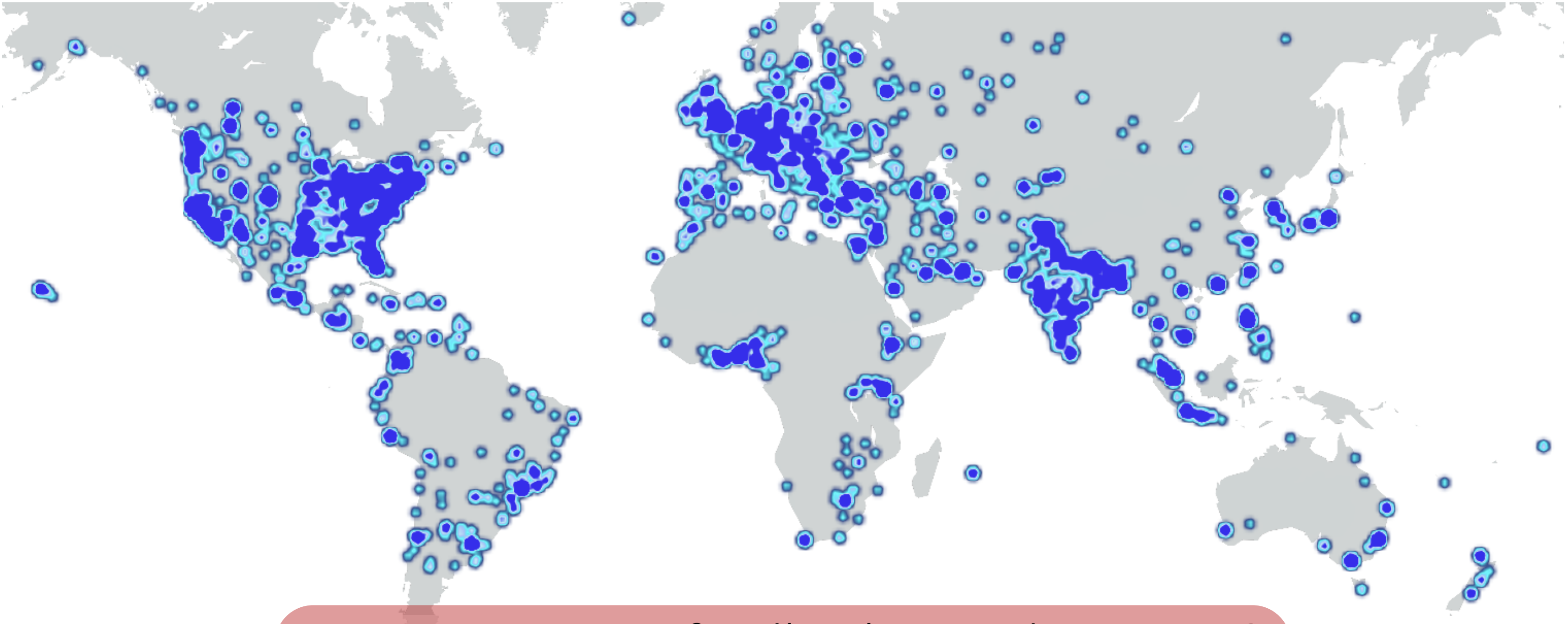
https://ai.stanford.edu/blog/prototransformer/

https://arxiv.org/abs/2107.14035

- Few-shot
- Learn from a few examples of feedback to students' submissions (code)
- Generates human-like feedback.

The Feedback Problem

Code-in-Place 2021: Free intro to CS course, 12,000+ students from 150+ countries



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

The Feedback Challenge

Formulate it as a clf problem.

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

INPUT

```
# print 1 to n w/ loop

def my_solution(n)

        print(1)

        print(2)

        print(3)
```

Grading rubric LABEL

[x] Incorrect Syntax

[x] Did not loop

[ ] Uses "print" fn

Predict!

Same rubrics that instructors use to give their feedback.

The Feedback Challenge

Why is this a hard problem for ML?

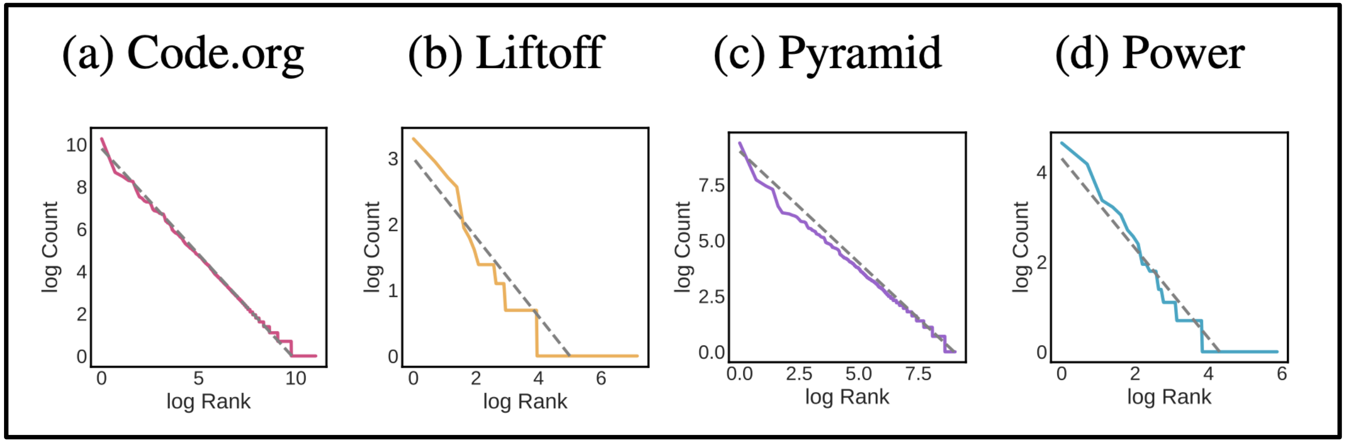
- **Limited annotation:** grading student work takes expertise and is very time consuming.

  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 *many* ways.



Generative Grading: Neural Approximate Parsing for Verifiable Automated Student Feedback (Malik et. al. 2020)

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 *many* ways.
- **Changing curriculums:** instructors constantly edit assignments and exams. Student solutions and instructor feedback look different year to year.

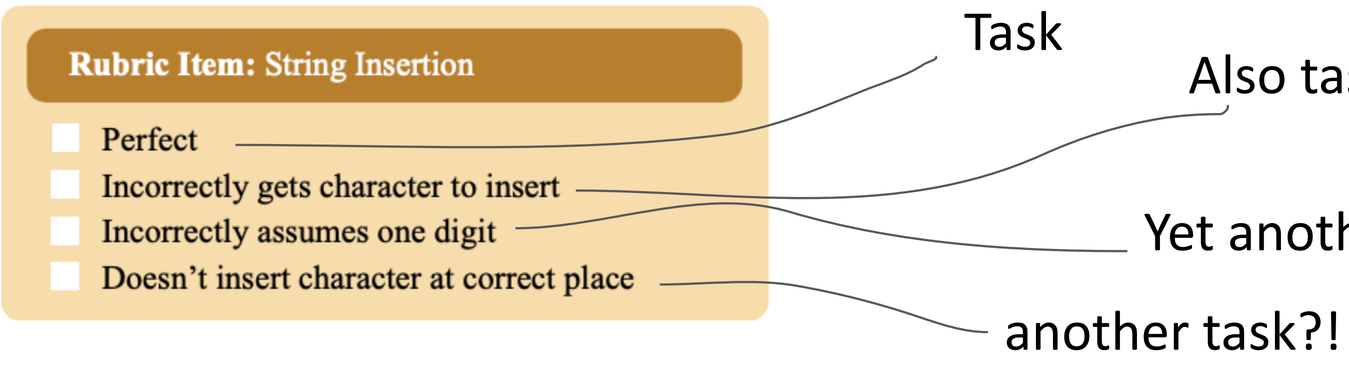
Framing it as a Meta-Learning Problem

Meta-Training Dataset: 4 final exams and 4 midterm exams from CS106.

- 63 questions and 24.8k student solutions.
- Every student solution has feedback via a rubric.

A rubric has several items. Each item has several options that you may pick as true.

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



Each rubric item as a task

Prototypical Networks for Few-shot Learning (Snell et. al. 2017)

ProtoTransformer

$$p_\theta(y = n|x) = \frac{\exp(-d\,(f_\theta(x), \mathbf{c}_n))}{\sum_{n'} \exp(d(f_\theta(x), \mathbf{c}_{n'}))}$$

Typo!

- x = (x_1, x_2, ..., x_T) is a **sequence of discrete tokens** (e.g. code, language).
- The embedding f_Θ: X→R^d is **a RoBERTa model** (stacked transformers) where token embeddings are averaged into a single vector. → takes Python code → embedding
- Applying this out of the box fails. wat

Attention is **not** all you need.😮

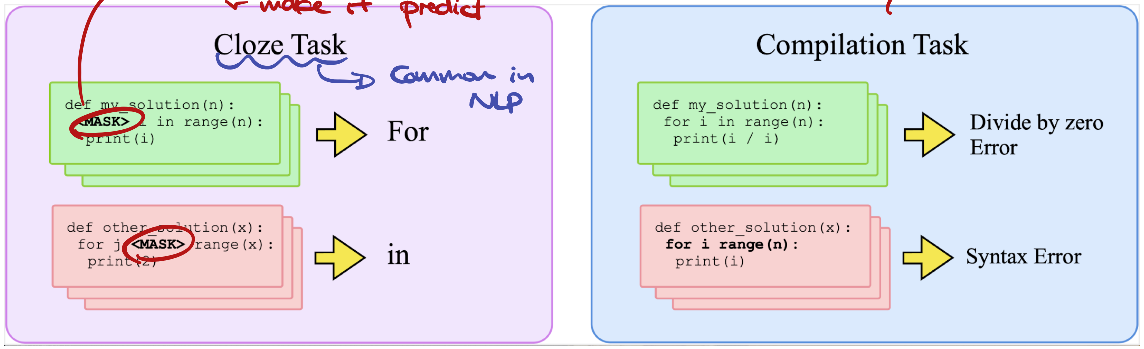
Ñooooo

nice

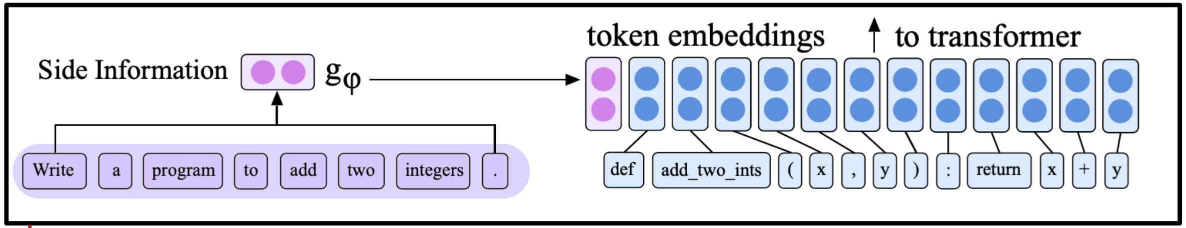
Trick #1: Augment rubric tasks with self-supervised tasks

Construct a task by predicting the compilation error Python returns when you run the code

mask out parts of code + make it predict



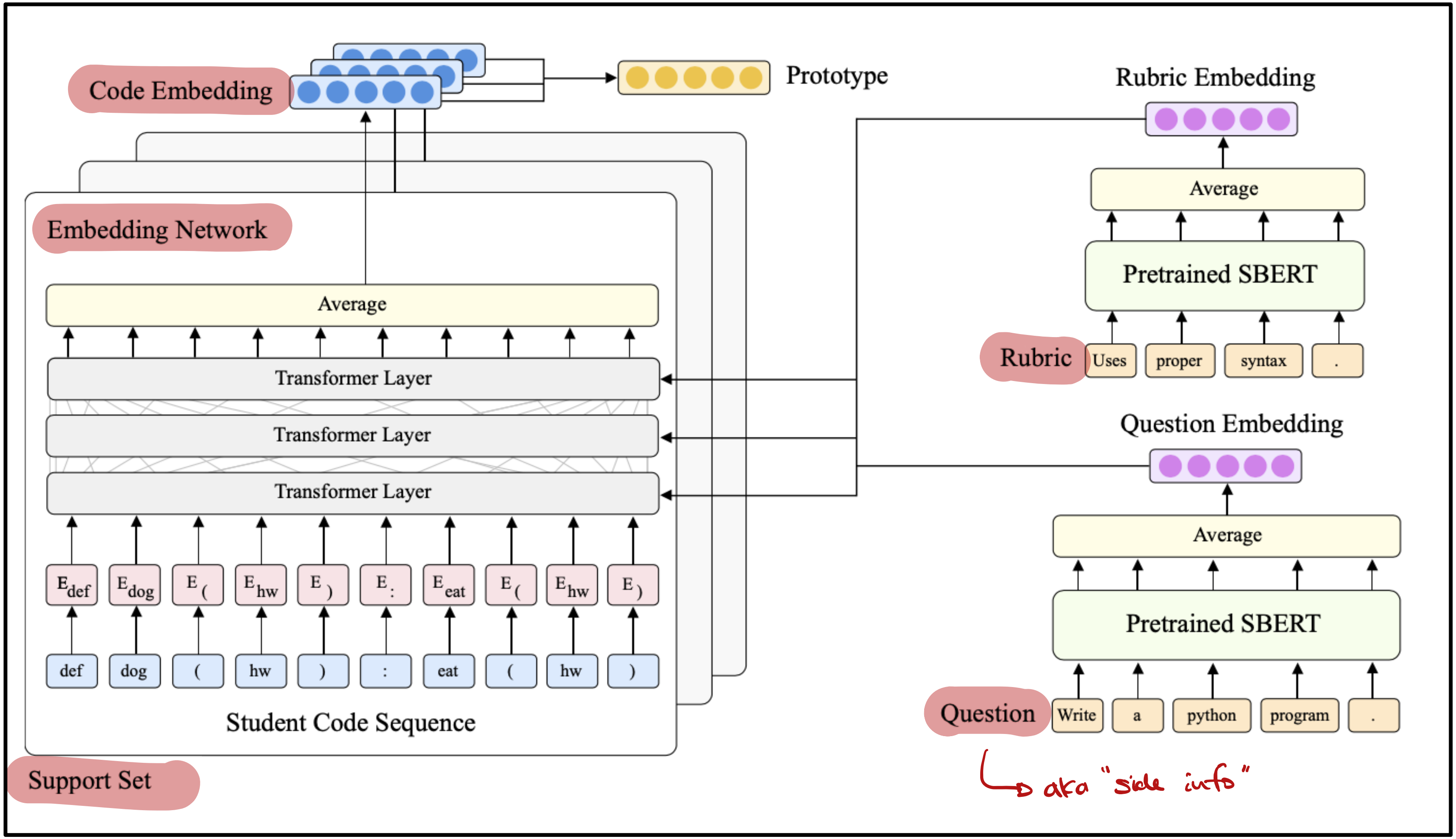
Trick #2: Reduce few-shot ambiguity by incorporating side information (rubric option name, question text)



inform the network what the task might be.

Trick #3: Pre-train on unlabeled Python code. How is this done, exactly?

CodeBERT: A Pre-Trained Model for Programming and Natural Languages (Feng et. al. 2020)
CodeSearchNet Challenge: Evaluating the State of Semantic Code Search (Husain et.al. 2020)



Code Embedding
Prototype
Rubric Embedding
Embedding Network
Average
Transformer Layer
Transformer Layer
Transformer Layer
E def
E dog
E (
E hw
E)
E :
E eat
E (
E hw
E)
def
dog
(
hw
)
:
eat
(
hw
)
Student Code Sequence
Support Set
Average
Pretrained SBERT
Rubric
Uses
proper
syntax
.
Question Embedding
Average
Pretrained SBERT
Question
Write
a
python
program
.
aka "side info"

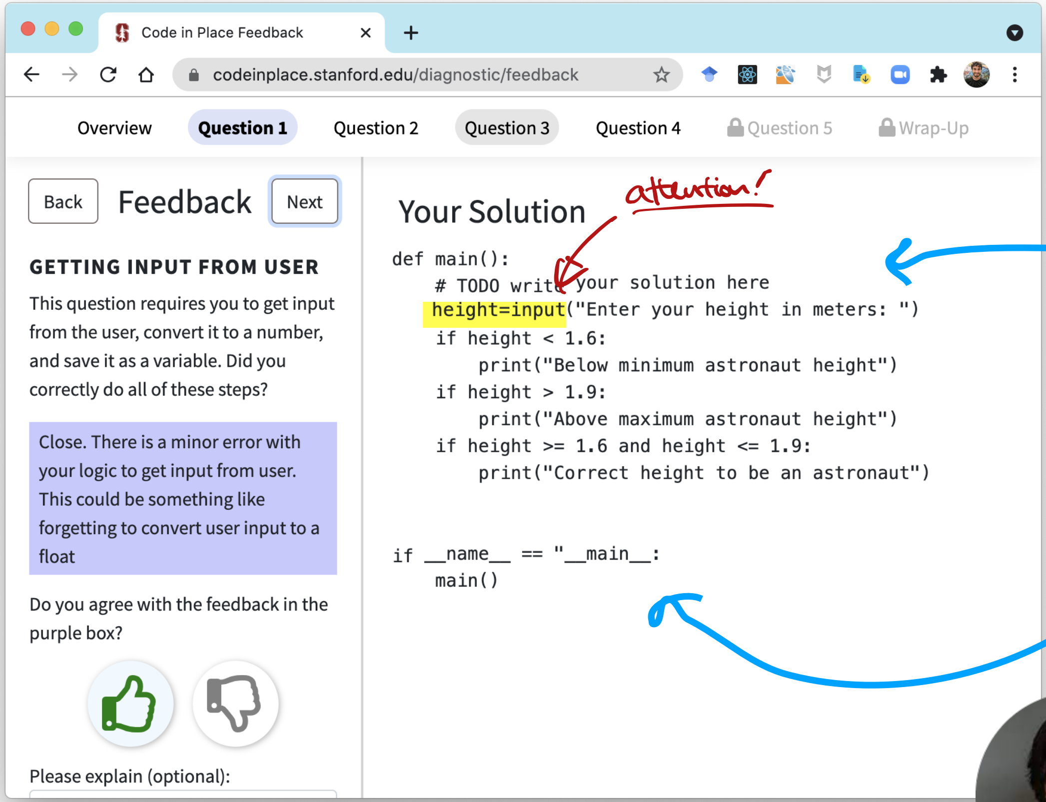
Main Offline Results

Model	AP	P@50	P@75	ROC-AUC
	Held-out rubric			
ProtoTransformer	**84.2** (±1.7)	**85.2** (±3.8)	**74.2** (±1.4)	**82.9** (±1.3)
Supervised	66.9 (±2.2)	59.1 (±1.7)	53.9 (±1.5)	61.0 (±2.1)
Human TA	82.5	–	–	–
	Held-out exam			
Model	AP	P@50	P@75	ROC-AUC
ProtoTransformer	**74.2** (±1.6)	**77.3** (±2.7)	**67.3** (±2.0)	**77.0** (±1.4)
Supervised	65.8 (± 2.1)	60.1 (±3.0)	54.3 (±1.8)	60.7 (±1.6)
Human TA	82.5	–	–	–

- Outperforms supervised learning by 8-17%
- More accurate than human TA on held-out rubric
  ↳ not that good
- Room to grow on held-out exam
  ↳ might include tokens unseen during training

Live Deployment to Code-in-Place Students

May 10th, 2021: Students took diagnostic.



Algorithm uses attention to highlight where the error arises

AI generated feedback

Syntax error here would prevent unit tests from being useful

Students evaluate the feedback

UI designed by Alan Cheng & Chris Piech

Blind, randomized trial *with real students*

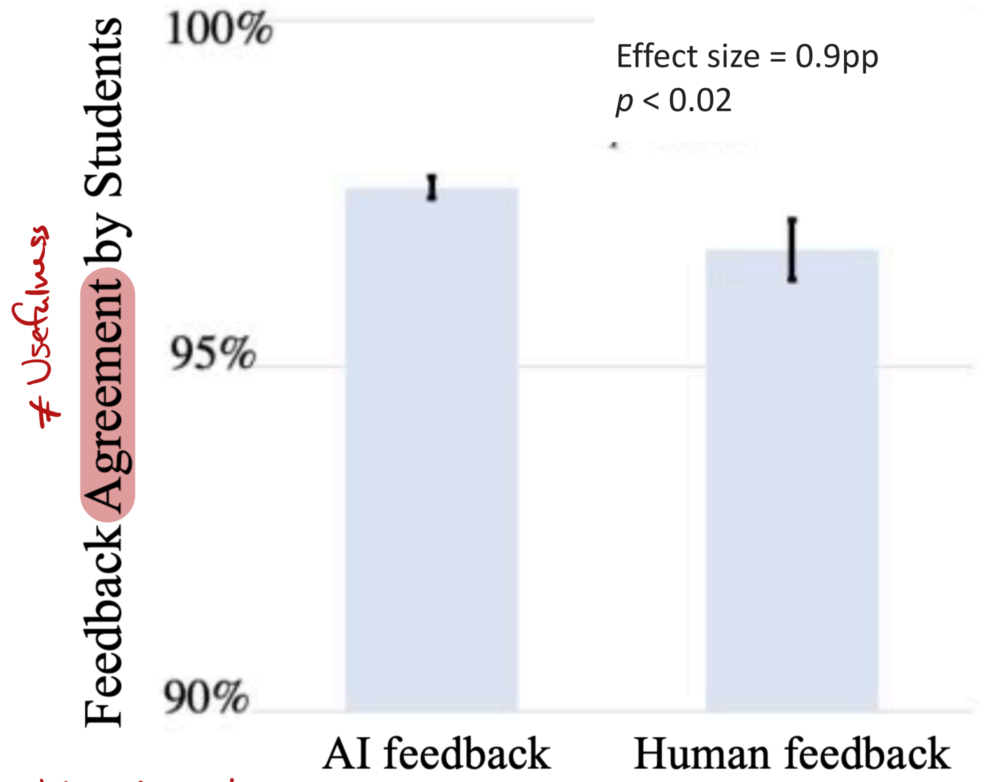
Humans gave feedback ~1k answers.
AI gave feedback on the remaining ~**15k**.

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

Volunteer TAs

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

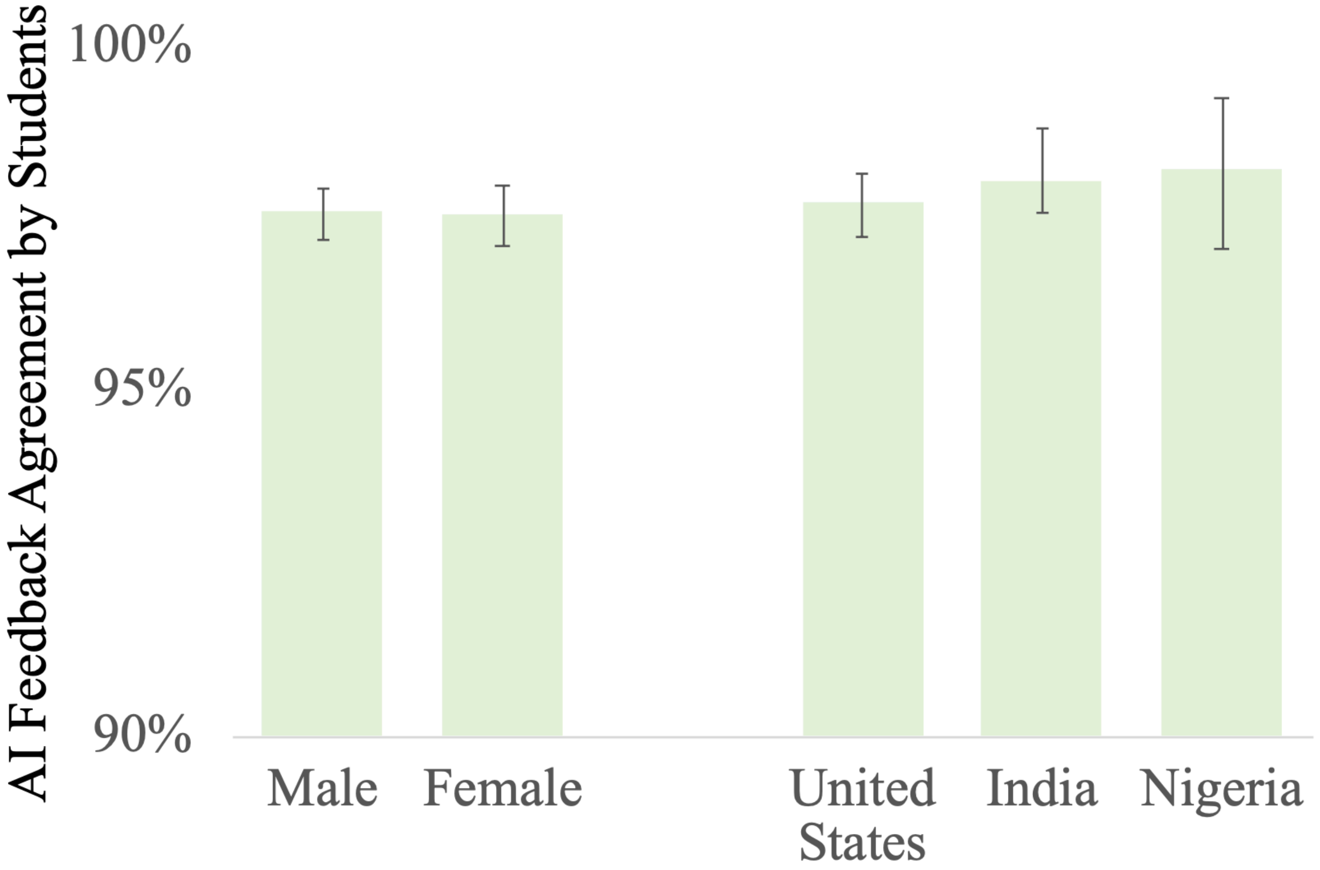
How??



Similar btw AI & Humans

Average holistic rating of usefulness by students was **4.6 ± 0.018 out of 5**.

No signs of bias by demographics



No signs
of bias

Plan for Today

Non-Parametric Few-Shot Learning
- Siamese networks, matching networks, prototypical networks
- Case study of few-shot student feedback generation

Properties of Meta-Learning Algorithms
- Comparison of approaches

Example Meta-Learning Applications
- Imitation learning, drug discovery, motion prediction, language generation

How can we think about how these methods compare?

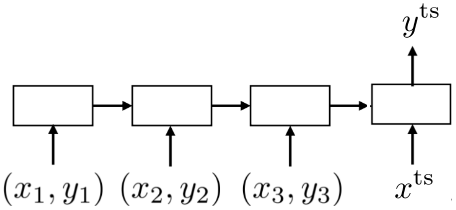
Black-box vs. Optimization vs. Non-Parametric

Directed graph where { nodes : mathematical operations; edges : flow of data btw. ops

***Computation graph* perspective**

Black-box

$$y^{\text{ts}} = f_\theta(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$$



Optimization-based

$$y^{\text{ts}} = f_{\text{MAML}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$$
$$= f_{\phi_i}(x^{\text{ts}})$$
$$\text{where } \phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})$$

Non-parametric

$$y^{\text{ts}} = f_{\text{PN}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$$
$$= \text{softmax}(-d\left(f_\theta(x^{\text{ts}}), \mathbf{c}_n\right))$$
$$\text{where } \mathbf{c}_n = \frac{1}{K} \sum_{(x,y)\in\mathcal{D}_i^{\text{tr}}} \mathbb{1}(y=n) f_\theta(x)$$

Note: (again) Can mix & match components of computation graph

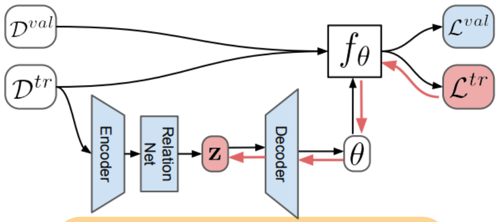
BB + optim.

Both condition on data & run gradient descent.

Jiang et al. CAML '19

optim. + nonparametric

Gradient descent on relation net embedding.



Rusu et al. LEO '19

MAML, but initialize last layer as ProtoNet during meta-training

Triantafillou et al. Proto-MAML '19

Black-box vs. Optimization vs. Non-Parametric

***Algorithmic properties* perspective**

query set을 integrate를 잘 할 수 있느냐...?

Expressive power — the ability for f to represent a range of learning procedures

Why? scalability, applicability to a range of domains

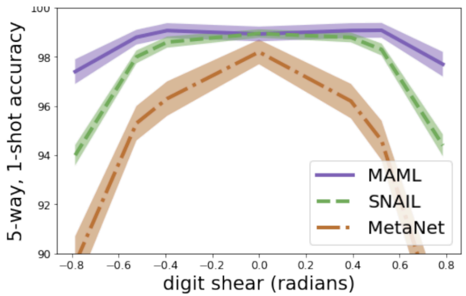
esp. where we don't have good learning algorithms.

Consistency — learned learning procedure will monotonically improve with more data

implies the algo will generalize better, but not vice versa.

Why? reduce reliance on meta-training tasks, good OOD task performance

Recall:



Better consistency.

These properties are important for most applications!

Black-box vs. Optimization vs. Non-Parametric

Too much compr. will not make it consistent ∵ in the embedding space, two points will not be separated enough to distinguish

Black-box

+ **complete expressive power**

- **not consistent** → won't necessarily get better w/ more data

+ easy to combine with **variety of learning problems** (e.g. SL, RL)

- **challenging optimization** (no inductive bias at the initialization)

- often **data-inefficient**

(★) Useful in reinforcement learning scenarios (because no good inner-loop optimiz. exists for RL)

Optimization-based

+ **consistent, reduces to GD** → grad descent

~ **expressive** for **very deep models***

+ **positive inductive bias** at the start of meta-learning

+ handles **varying** & **large K** well

+ **model-agnostic**

- **second-order optimization** ☹

- usually **compute** and **memory** intensive

Non-parametric

+ **expressive** for **most architectures**

~ **consistent** under **certain conditions** → if your embedding func. doesn't compress too much about the input.

+ entirely **feedforward** @ meta test time (inner-loop)

+ **computationally fast** & **easy to optimize**

- **harder to generalize** to **varying K**

- hard to scale to **very large K** ← empirical observation

- so far, **limited to classification** → if not clf, use optimization-based (recommended)

Generally, well-tuned versions of each perform **comparably** on many few-shot benchmarks!

(likely says more about the benchmarks than the methods)

→ also useful for scenarios where you have a lot of data, or don't care too much about data efficiency. e.g.) GPT-3 used BB approach.

Which method to use depends on your **use-case**. → also the case for ★ MTL vs. Meta-learning

*for supervised learning settings

Q: In all these methods we need many tasks. What to do when task no. is also small? (along w/ the no. of data pts)? Use multi-task learning over meta-learning?

A: Small no. of tasks → consistency is v. important, ∵ it's running GD on new tasks → less reliant on having a large no. of meta training tasks.

⇒ Can also do task augmentation (like in the case study) if possible.

Feature	Multi-task Learning	Meta-learning
Goal	Improve performance on a new task by learning shared features from multiple related tasks	Learn how to learn new tasks quickly
Training data	Set of multiple related tasks	Set of tasks that are all similar in some way
Performance	Better for tasks where the new tasks are similar to the tasks that the model was trained on	Better for tasks where the new tasks are not similar to the tasks that the model was trained on
Implementation difficulty	Easier	More difficult

Black-box vs. Optimization vs. Non-Parametric

***Algorithmic properties* perspective**

Expressive power

the ability for f to represent a range of learning procedures

Why? scalability, applicability to a range of domains

Consistency

learned learning procedure will monotonically improve with more data

Why? reduce reliance on meta-training tasks,
good OOD task performance

Uncertainty awareness

Will cover later

ability to reason about ambiguity during learning

Why? active learning, calibrated uncertainty, RL
principled Bayesian approaches

We'll discuss this in 2 weeks!

Plan for Today

Non-Parametric Few-Shot Learning

- Siamese networks, matching networks, prototypical networks
- Case study of few-shot student feedback generation

Properties of Meta-Learning Algorithms

- Comparison of approaches

Example Meta-Learning Applications

- Imitation learning, drug discovery, motion prediction, language generation

Application: One-Shot Imitation Learning

(Yu*, Finn* et al. One-Shot Imitation from Observing Humans. RSS 2018)

Tasks:

manipulating different objects

$\mathscr{D}_i^{\mathrm{tr}}$: video of a human

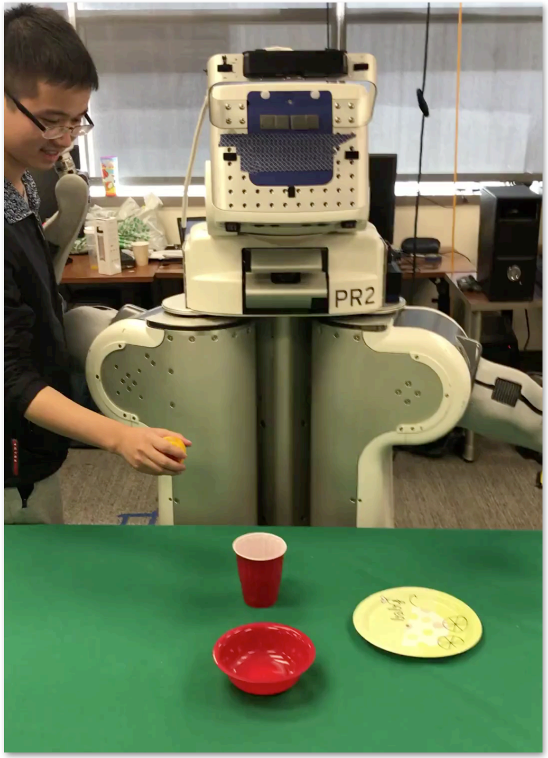
$\mathscr{D}_i^{\mathrm{ts}}$: teleoperated demonstration

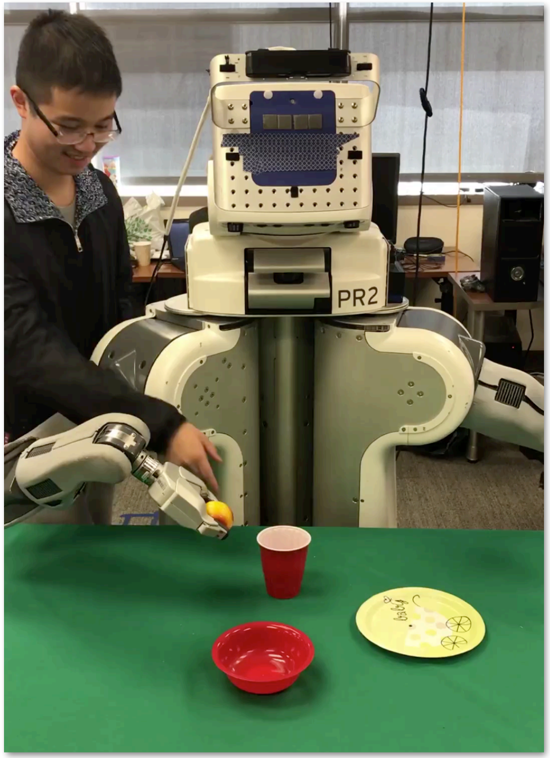
Model: optimization-based

MAML with *learned* inner loss

video of human don't have labels,
just input images (actions the robot should take)

used to run GD in the inner loop.





Needed for drug discovery

Application: Low-Resource Molecular Property Prediction

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

[potentially useful for low-resource drug discovery problems]

Tasks:

Predicting properties & activities of different molecules

$\mathscr{D}_i^{\text{tr}}, \mathscr{D}_i^{\text{ts}}$: different instances

Model: optimization-based

variant of MAML: only updates the last layer in the inner loop.

MAML, first-order MAML, ANIL

Gated graph neural net base model

ChEMBL ID	k-NN	Finetune-All	Finetune-Top	FO-MAML	ANIL	MAML
2363236	0.316 ± 0.007	0.328 ± 0.028	0.329 ± 0.023	$\mathbf{0.337 \pm 0.019}$	0.325 ± 0.008	0.332 ± 0.013
1614469	0.438 ± 0.023	0.470 ± 0.034	0.490 ± 0.033	0.489 ± 0.019	0.446 ± 0.044	$\mathbf{0.507 \pm 0.030}$
2363146	0.559 ± 0.026	0.626 ± 0.037	$\mathbf{0.653 \pm 0.029}$	0.555 ± 0.017	0.506 ± 0.034	0.595 ± 0.051
2363366	0.511 ± 0.050	0.567 ± 0.039	0.551 ± 0.048	0.546 ± 0.037	0.570 ± 0.031	$\mathbf{0.598 \pm 0.041}$
2363553	$\mathbf{0.739 \pm 0.007}$	0.724 ± 0.015	0.737 ± 0.023	0.694 ± 0.011	0.686 ± 0.020	0.691 ± 0.013
1963818	0.607 ± 0.041	0.708 ± 0.036	0.595 ± 0.142	0.677 ± 0.026	0.692 ± 0.081	$\mathbf{0.745 \pm 0.048}$
1963945	0.805 ± 0.031	$\mathbf{0.848 \pm 0.034}$	0.835 ± 0.036	0.779 ± 0.039	0.753 ± 0.033	0.836 ± 0.023
1614423	0.503 ± 0.044	0.628 ± 0.058	0.642 ± 0.063	0.760 ± 0.024	0.730 ± 0.077	$\mathbf{0.837 \pm 0.036}^*$
2114825	0.679 ± 0.027	0.739 ± 0.050	0.732 ± 0.051	0.837 ± 0.042	0.759 ± 0.078	$\mathbf{0.885 \pm 0.014}^*$
1964116	0.709 ± 0.042	0.758 ± 0.044	0.769 ± 0.048	0.895 ± 0.023	0.903 ± 0.016	$\mathbf{0.912 \pm 0.013}$
2155446	0.471 ± 0.008	0.473 ± 0.017	0.476 ± 0.013	0.497 ± 0.024	0.478 ± 0.020	$\mathbf{0.500 \pm 0.017}$
1909204	0.538 ± 0.023	0.589 ± 0.031	0.577 ± 0.039	0.592 ± 0.043	0.547 ± 0.029	$\mathbf{0.601 \pm 0.027}$
1909213	0.694 ± 0.009	0.742 ± 0.015	$\mathbf{0.759 \pm 0.012}$	0.698 ± 0.024	0.694 ± 0.025	0.729 ± 0.013
3111197	0.617 ± 0.028	0.663 ± 0.066	0.673 ± 0.071	0.636 ± 0.036	0.737 ± 0.035	$\mathbf{0.746 \pm 0.045}$
3215171	0.480 ± 0.042	0.552 ± 0.043	0.551 ± 0.045	0.729 ± 0.031	0.700 ± 0.050	$\mathbf{0.764 \pm 0.019}$
3215034	0.474 ± 0.072	0.540 ± 0.156	0.455 ± 0.189	$\mathbf{0.819 \pm 0.048}$	0.681 ± 0.042	0.805 ± 0.046
1909103	0.881 ± 0.026	$\mathbf{0.936 \pm 0.013}$	0.921 ± 0.020	0.877 ± 0.046	0.730 ± 0.055	0.900 ± 0.032
3215092	0.696 ± 0.038	0.777 ± 0.039	0.791 ± 0.042	0.877 ± 0.028	0.834 ± 0.026	$\mathbf{0.907 \pm 0.017}$
1738253	0.710 ± 0.048	0.860 ± 0.029	0.861 ± 0.025	0.885 ± 0.033	0.758 ± 0.111	$\mathbf{0.908 \pm 0.011}$
1614549	0.710 ± 0.035	0.850 ± 0.041	0.860 ± 0.051	0.930 ± 0.022	0.860 ± 0.034	$\mathbf{0.947 \pm 0.014}$
Avg. Rank	5.4	3.5	3.5	3.1	4.0	**1.7**

Application: Few-Shot Human Motion Prediction

(Gui et al. Few-Shot Human Motion Prediction via Meta-Learning. ECCV 2018)

[potentially useful for human-robot interaction, autonomous driving]

Tasks:

Different human users & motions

$\mathscr{D}_i^{\mathrm{tr}}$: past K time steps of motion

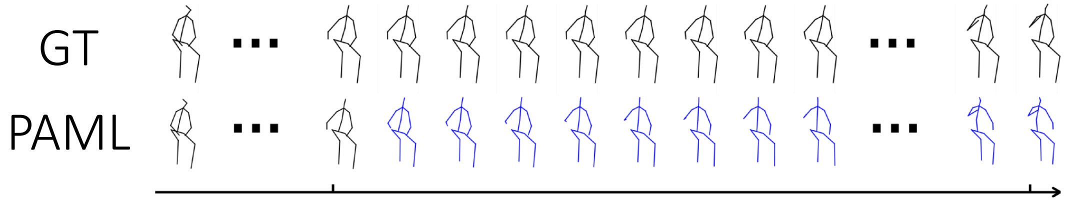
$\mathscr{D}_i^{\mathrm{ts}}$: future second(s) of motion

Model:

optimization-based/black-box hybrid

MAML with additional learned update rule

Recurrent neural net base model



		Walking						Eating					
milliseconds		80	160	320	400	560	1000	80	160	320	400	560	1000
residual sup. [32] w/ (Baselines)	$\mathrm{Scratch}_{\mathrm{spec}}$	1.90	1.95	2.16	2.18	1.99	2.00	2.33	2.31	2.30	2.30	2.31	2.34
	$\mathrm{Scratch}_{\mathrm{agn}}$	1.78	1.89	2.20	2.23	2.02	2.05	2.27	2.16	2.18	2.27	2.25	2.31
	$\mathrm{Transfer}_{\mathrm{ots}}$	0.60	0.75	0.88	0.93	1.03	1.26	0.57	0.70	0.91	1.04	1.19	1.58
	Multi-task	0.57	0.71	0.79	0.85	0.96	1.12	0.59	0.68	0.83	0.93	1.12	1.33
	$\mathrm{Transfer}_{\mathrm{ft}}$	0.44	0.55	0.85	0.95	0.74	1.03	0.61	0.65	0.74	0.78	0.86	1.19
Meta-learning (Ours)	PAML	**0.35**	**0.47**	**0.70**	**0.82**	**0.80**	**0.83**	**0.36**	**0.52**	**0.65**	**0.70**	**0.71**	**0.79**

		Smoking						Discussion					
milliseconds		80	160	320	400	560	1000	80	160	320	400	560	1000
residual sup. [32] w/ (Baselines)	$\mathrm{Scratch}_{\mathrm{spec}}$	2.88	2.86	2.85	2.83	2.80	2.99	3.01	3.13	3.12	2.95	2.62	2.99
	$\mathrm{Scratch}_{\mathrm{agn}}$	2.53	2.61	2.67	2.65	2.71	2.73	2.77	2.79	2.82	2.73	2.82	2.76
	$\mathrm{Transfer}_{\mathrm{ots}}$	0.70	0.84	1.18	1.23	1.38	2.02	0.58	0.86	1.12	1.18	1.54	2.02
	Multi-task	0.71	0.79	1.09	1.20	1.25	1.23	0.53	0.82	1.02	1.17	1.33	1.97
	$\mathrm{Transfer}_{\mathrm{ft}}$	0.87	1.02	1.25	1.30	1.45	2.06	0.57	0.82	1.11	1.11	1.37	2.08
Meta-learning (Ours)	PAML	**0.39**	**0.66**	**0.81**	**1.01**	**1.03**	**1.01**	**0.41**	**0.71**	**1.01**	**1.02**	**1.09**	**1.12**

mean angle error w.r.t. prediction horizon

Closing note for today

! will cover later.

$\mathscr{D}_i^{\mathrm{tr}}$ and $\mathscr{D}_i^{\mathrm{ts}}$ do not need to be sampled independently from $\mathscr{D}_i$.

eg). want to learn a good cat/dog classifier (for natural images) using hand sketches.

$\rightarrow \begin{cases} D_i^{tr} : \text{hand sketches} \\ D_i^{ts} : \text{natural images} \end{cases}$

$\mathscr{D}_i^{\mathrm{tr}}$ could have:

much more challenging

- noisy labels
- weakly supervised
- domain shift
- etc.

v. important for the test set to be a well-formed ML objective
$\because$ it drives the optimization process

$\rightarrow$ meta-learn for learning procedures that are
- more robust to noisy labels,
- can learn from weak supervision
- etc

Plan for Today

Non-Parametric Few-Shot Learning
- Siamese networks, matching networks, prototypical networks
- Case study of few-shot student feedback generation

Properties of Meta-Learning Algorithms
- Comparison of approaches

Example Meta-Learning Applications
- Imitation learning, drug discovery, motion prediction, language generation

Goals for by the end of lecture:
- Basics of non-parametric few-shot learning techniques (& how to implement)
- Trade-offs between black-box, optimization-based, and non-parametric meta-learning
- Familiarity with applied formulations of meta-learning

Course Logistics

Lecture Topics

Done with meta-learning algorithms!

Next week: unsupervised pre-training

Coursework

Homework 1 due **tonight**.

Homework 2 released, due Mon 10/24.

Project mentors to be assigned this week.

Project proposal due next Weds 10/19.

(graded lightly, for your benefit)