Week 1, Lecture 1 - Course Introduction

Note: MEL = meta-learning.

Topics

- Modern DL methods for learning across tasks
- Implementing these methods (MT, TL) in PyTorch
- Glimpse of building new algorithms

low-level descriptions:

- MT, TL
- Meta learning algos
- Advanced meta learning topics
- Unsupervised pre-training
  - FS learning
- Domain adaption
- Lifelong learning
- Open problems

Focus on DL, with case studies in things like NLP. - No RL! (see CS 224R)

1. Logistics

- Lectures are live-streamed and recorded
- two guest lectures
- Prereqs:
  - Sufficient background in ML (229)

Homeworks

50% of grade.

- 0: multi-task basics
- 1: multi-task data processing and BB-ML
- 2: gradient-based ML
- 3: fine-tuning pre-trained language models
- 4 (optional): Bayesian ML and meta overfitting

    - Replace 15% of hw/project
    - Not coding, all math

- 6 late days

Project

- Poster session, 50% of grade.
- Idea: ...

Now technical content...

2. Why study multi-task learning and meta-learning?

- How can we enable agents to learn a breadth of skills in the real world?
    - Because each time we have to train a supervised signal
        - So the goal is to learn representations across tasks
- Aside (common paradigm to learn representations): initialize well (not randomly) –> fine-tune on new task.
    - This is harder for RL than NLP because NLP has the entire wikipedia to use but robotic common sense representations are not as straightforward (maybe we need a common robot embedding?)

Evolution:

- Early in CV: hand-design features, train SVM on-top
- Modern CV: end-to-end training, no hand-engineering
    - Allows us to handle unstructured inputs without understanding it
- Now why meta-learning? Three reasons...
    - **Don't have large dataset** at the outset to pre-train on or use in end-to-end SL manner (med imaging, robotics, etc.)
        - Even more so: **long-tail data** samples (e.g., self-driving won't catch all edge cases)
            - MEL techniques can help with this (kinda... not the main focus tho)
    - **Quickly learn something new** (few-shot learning)
- Lots of open problems

Multi-task intro

- What is a task?
  - Dataset + loss objective –> model
  - Objects as "tasks"
  - Critical assumption: different tasks need to share some base structure (goal is to exploit shared structure)
    - But lots of tasks share structure (even as upstream as sharing the laws of physics!)
    - Question: can we learn a shared embedding space for e.g., text + images in one?
- Does MT learning reduce to single-task SL learning?
  - Somewhat (tho not for every problem)
  - Idea: sum loss and data:

$$\mathcal{D} = \bigcup \mathcal{D}_i \quad \mathcal{L} = \sum \mathcal{L}_i$$

Next up: a technical dive into the **multi-task** learning framework.