Transfer Learning +
Start of Meta-Learning

CS 330
Lecture 3
Logistics

Homework 0 due **Wednesday 10/4 at 11:59 pm PT**.
Homework 1 out on Wednesday.

Project ideas posted on Ed.
Recap from Last Time

A task: \( \mathcal{T}_i \triangleq \{ p_i(x), p_i(y \mid x), \mathcal{L}_i \} \)

Corresponding datasets: \( \mathcal{D}^{tr}_i, \mathcal{D}^{test}_i \)

Learning a task: \( \mathcal{D}^{tr}_i \rightarrow \theta \)

\[
\min_{\theta} \sum_{i=1}^{T} w_i \mathcal{L}_i(\theta, \mathcal{D}_i)
\]

- Choice of task weighting \( w_i \) affects prioritization of tasks.

Multi-task learning learns neural network conditioned on task descriptor \( z_i \)

\[
f_{\theta}(y \mid x, z_i)
\]

- Choice of how to condition on \( z_i \) affects how parameters are shared.
  - If you observe negative transfer, share less.
  - If you observe overfitting, try sharing more.
Plan for Today

**Transfer Learning**
- Problem formulation
- Fine-tuning

**Start of Meta-Learning**
- Problem formulation
- General recipe of meta-learning algorithms

What you’ll learn:
- How can you **transfer** things learned from one task to another?
- What does it mean for two tasks to have *"shared structure"*?
- What is **meta-learning**?

} Part of Homework 1!
Multi-Task Learning vs. Transfer Learning

**Multi-Task Learning**
Solve multiple tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$ at once.

$$\min_{\theta} \sum_{i=1}^{T} \mathcal{L}_i(\theta, \mathcal{D}_i)$$

**Transfer Learning**
Solve target task $\mathcal{T}_b$ after solving source task(s) $\mathcal{T}_a$ by transferring knowledge learned from $\mathcal{T}_a$.

Common assumption: Cannot access data $\mathcal{D}_a$ during transfer.

Transfer learning is a valid solution to multi-task learning. (but not vice versa)

**Question:** What are some problems/applications where transfer learning might make sense?

- when $\mathcal{D}_a$ is very large (don’t want to retain & retrain on $\mathcal{D}_a$)
- when you don’t care about solving $\mathcal{T}_a$ & $\mathcal{T}_b$ simultaneously
Transfer learning via fine-tuning

\[ \phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{tr}) \]

( typically for many gradient steps )

Parameters pre-trained on \( \mathcal{D}_a \)

training data for new task \( \mathcal{F}_b \)

Where do you get the pre-trained parameters?
- ImageNet classification
- Models trained on large language corpora (BERT, LMs)
- Other unsupervised learning techniques
- Whatever large, diverse dataset you might have

Pre-trained models often available online.

Common design choices
- Fine-tune with a smaller learning rate
- Smaller learning rate for earlier layers
- Freeze earlier layers, gradually unfreeze
- Reinitialize last layer
- Search over hyperparameters via cross-val
- Architecture choices matter (e.g. ResNets)

What makes ImageNet good for transfer learning? Huh, Agrawal, Efros. ‘16

<table>
<thead>
<tr>
<th>Pre-trained Dataset</th>
<th>PASCAL</th>
<th>SUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet</td>
<td>58.3</td>
<td>52.2</td>
</tr>
<tr>
<td>Random</td>
<td>41.3 [21]</td>
<td>35.7 [2]</td>
</tr>
</tbody>
</table>
When might this common wisdom break?

Unsupervised pre-training objectives may not require diverse data for pre-training.

When might this common wisdom break?

Yoonho’s (rough) thought process

1. Fine-tuning **only the last layer** works well.
2. Is there anything special about the last layer?
3. For fine-tuning to low-level image corruptions, maybe the first layer might be better?

Result: Fine-tuning the first or middle layers can work better than the last layers.
Chelsea’s recommended default

Train last layer, then fine-tune entire network

Kumar, Raghunathan, Jones, Ma, Liang. Fine-Tuning Can Distort Pre-Trained Features and Underperform Out-of-Distribution. ICLR 2022
How does fine-tuning work with varying target dataset sizes?

Figure 3: Validation error rates for supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

Universal Language Model Fine-Tuning for Text Classification. Howard, Ruder. ‘18

Fine-tuning doesn’t work well with very small target task datasets

This is where meta-learning can help.
Plan for Today

*Transfer Learning*
- Problem formulation
- Fine-tuning

*Meta-Learning*
- Problem formulation
- General recipe of meta-learning algorithms
From Transfer Learning to Meta-Learning

**Transfer learning**: Initialize model. Hope that it helps the target task.

**Meta-learning**: Can we explicitly *optimize* for transferability?

Given a set of training tasks, can we optimize for the ability to learn these tasks quickly? so that we can learn *new* tasks quickly too.

Learning a task: $\mathcal{D}_i^{tr} \rightarrow \theta$

Can we optimize this function? (for small $\mathcal{D}_i^{tr}$)
Two ways to view meta-learning algorithms

Mechanistic view

➢ Deep network that can read in an entire dataset and make predictions for new datapoints
➢ Training this network uses a meta-dataset, which itself consists of many datasets, each for a different task

Probabilistic view

➢ Extract shared prior knowledge from a set of tasks that allows efficient learning of new tasks
➢ Learning a new task uses this prior and (small) training set to infer most likely posterior parameters
How would Bayes view it?

Graphical model for multi-task learning & meta-learning. (whiteboard)

What does “structure” mean? statistical dependence on shared latent information $\theta$

If you condition on that information,
- task parameters become independent
  i.e. $\phi_{i_1} \perp \phi_{i_2} | \theta$
  and are not otherwise independent $\phi_{i_1} \perp \phi_{i_2}$
- hence, you have a lower entropy
  i.e. $H(p(\phi_i | \theta)) < H(p(\phi_i))$

Thought exercise #1: If you can identify $\theta$ (i.e. with meta-learning), when should learning $\phi_i$ be faster than learning from scratch?

Thought exercise #2: what if $H(p(\phi_i | \theta)) = 0 \ \forall i$?
How would Bayes view it?

Graphical model for multi-task learning & meta-learning. (whiteboard)

What does “structure” mean? statistical dependence on shared latent information $\theta$

What information might $\theta$ contain...

...in a multi-task sinusoid problem?

$\theta$ corresponds to family of sinusoid functions (everything but phase and amplitude)

...in multi-language machine translation?

$\theta$ corresponds to the family of all language pairs

Note that $\theta$ is narrower than the space of all possible functions.
Two ways to view meta-learning algorithms

Mechanistic view

➢ Deep network that can read in an entire dataset and make predictions for new datapoints
➢ Training this network uses a meta-dataset, which itself consists of many datasets, each for a different task

Probabilistic view

➢ Extract shared prior knowledge from a set of tasks that allows efficient learning of new tasks
➢ Learning a new task uses this prior and (small) training set to infer most likely posterior parameters

For rest of lecture: Focus primarily on the mechanistic view.

(Bayes will be back later)
How does meta-learning work? An example.

Given 1 example of 5 classes:

- training data $D_{\text{train}}$

Classify new examples:

- test set $X_{\text{test}}$
How does meta-learning work? An example.

Given 1 example of 5 classes:

training data $D_{\text{train}}$

Can replace image classification with: regression, language generation, skill learning, any ML problem

meta-training $\mathcal{T}_{1}$

$\mathcal{T}_{2}$

:::

training classes

meta-testing $\mathcal{T}_{\text{test}}$

Classify new examples

test set $X_{\text{test}}$
Meta-Learning Problem
Transfer Learning with Many Source Tasks

Given data from $\mathcal{T}_1, \ldots, \mathcal{T}_n$, solve new task $\mathcal{T}_{\text{test}}$ more quickly / proficiently / stably

**Key assumption:** meta-training tasks and meta-test task drawn i.i.d. from same task distribution $\mathcal{T}_1, \ldots, \mathcal{T}_n \sim p(\mathcal{T}), \mathcal{T}_j \sim p(\mathcal{T})$

Like before, tasks must share structure.

What do the tasks correspond to?
- recognizing handwritten digits from different languages (see homework 1!)
- giving feedback to students on different exams
- classifying species in different regions of the world
- a robot performing different tasks

How many tasks do you need? The more the better. (analogous to more data in ML)
Some terminology

- **task training set** $\mathcal{D}_{i}^{tr}$ “support set” “context”
- **task test dataset** $\mathcal{D}_{i}^{test}$ “query set”

**k-shot learning**: learning with k examples per class (or k examples total for regression)

**N-way classification**: choosing between N classes

**Question**: What are k and N for the above example?
Problem Settings Recap

**Multi-Task Learning**
Solve multiple tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$ at once.

$$\min_{\theta} \sum_{i=1}^{T} \mathcal{L}_i(\theta, \mathcal{D}_i)$$

**Transfer Learning**
Solve target task $\mathcal{T}_b$ after solving source task(s) $\mathcal{T}_a$ by transferring knowledge learned from $\mathcal{T}_a$.

**Meta-Learning Problem**
Transfer Learning with Many Source Tasks

Given data from $\mathcal{T}_1, \ldots, \mathcal{T}_n$, solve new task $\mathcal{T}_{test}$ more quickly / proficiently / stably

In transfer learning and meta-learning: generally impractical to access prior tasks

In all settings: tasks must share structure.
Plan for Today

Transfer Learning
- Problem formulation
- Fine-tuning

Meta-Learning
- Problem formulation
- General recipe of meta-learning algorithms
One View on the Meta-Learning Problem

Supervised Learning:

Inputs: \( x \)
Outputs: \( y \)

\[ y = g_\phi(x) \]

Data: \( \{(x, y)_i\} \)

Meta Supervised Learning:

Inputs: \( D^{tr} \) \( x^{ts} \)
Outputs: \( y^{ts} \)

\[ y^{ts} = f_\theta(D^{tr}, x^{ts}) \]

Data: \( \{D_i\} \)
\[ D_i : \{(x, y)_j\} \]

Why is this view useful?
Reduces the meta-learning problem to the design & optimization of \( f \).

General recipe

How to design a meta-learning algorithm

1. Choose a form of $f_\theta(D^{tr}, x^{ts})$

2. Choose how to optimize $\theta$ w.r.t. max-likelihood objective using meta-training data

meta-parameters
Lecture Recap

How can you transfer things learned from one task to another?

**Fine-tuning**: initialize on source task(s) then optimize on target task

Being careful to not destroy initialized features (e.g. smaller learning rate, train last layer first)

What does it mean for tasks to have "shared structure"?

- Statistical dependence on shared latent information $\theta$

**Meta-learning** aims to learn shared structure, use it to learn new tasks quickly.
Plan for Today

Transfer Learning
- Problem formulation
- Fine-tuning

Meta-Learning
- Problem formulation
- General recipe of meta-learning algorithms

What you’ll learn:
- How can you transfer things learned from one task to another?
- What does it mean for two tasks to have “shared structure”?
- What is meta-learning?
Roadmap

Next five lectures on core methods

Meta-learning methods (3 lectures) (homework 1 & 2)
Unsupervised pre-training methods (2 lectures) (homework 3)

Homework 0 due **Wednesday 10/4 at 11:59 pm PT**.
Homework 1 out on Wednesday.

Project ideas posted on Ed.