Multi-Task Learning & Transfer Learning Basics

CS 330
Logistics

Optional homework 0 due Monday 9/27.

PyTorch review session tomorrow at 6:00 pm PT.

Project guidelines posted.

Office hours start today
Plan for Today

Multi-Task Learning
- Problem statement
- Models, objectives, optimization
- Challenges
- Case study of real-world multi-task learning

Transfer Learning
- Pre-training & fine-tuning

Goals for by the end of lecture:
- Know the key design decisions when building multi-task learning systems
- Understand the difference between multi-task learning and transfer learning
- Understand the basics of transfer learning
Multi-Task Learning
Some notation

Single-task learning: \( \mathcal{D} = \{(x, y)_k\} \) [supervised]
\[ \min_\theta \mathcal{L}(\theta, \mathcal{D}) \]

Typical loss: negative log likelihood
\[ \mathcal{L}(\theta, \mathcal{D}) = -\mathbb{E}_{(x, y) \sim \mathcal{D}}[\log f_\theta(y | x)] \]

What is a task? (more formally this time)
A task:
\[ \mathcal{T}_i \triangleq \{p_i(x), p_i(y | x), \mathcal{L}_i\} \]
data generating distributions

Corresponding datasets:
\[ \mathcal{D}_i^{tr}, \mathcal{D}_i^{test} \]
will use \( \mathcal{D}_i \) as shorthand for \( \mathcal{D}_i^{tr} \):
Examples of Tasks

A task: \( \mathcal{T}_i \triangleq \{p_i(x), p_i(y | x), \mathcal{L}_i\} \)
- data generating distributions

Corresponding datasets: \( D_i^{tr}, D_i^{te} \)
- will use \( D_i \) as shorthand for \( D_i^{tr} \);

Multi-task classification: \( \mathcal{L}_i \) same across all tasks
- e.g. per-language handwriting recognition
- e.g. personalized spam filter

Multi-label learning: \( \mathcal{L}_i, p_i(x) \) same across all tasks
- e.g. CelebA attribute recognition
- e.g. scene understanding

When might \( \mathcal{L}_i \) vary across tasks?
- mixed discrete, continuous labels across tasks
- multiple metrics that you care about

\[ L_{tot} = w_{depth} L_{depth} + w_{kpt} L_{kpt} + w_{normals} L_{normals} \]
Vanilla MTL Objective: \[ \min_{\theta} \sum_{i=1}^{T} \mathcal{L}_i(\theta, \mathcal{D}_i) \]

Decisions on the model, the objective, and the optimization.

How should we condition on \( z_i \)? What objective should we use?

How to optimize our objective?

e.g. one-hot encoding of the task index

or, whatever meta-data you have

- personalization: user features/attributes
- language description of the task
- formal specifications of the task
Model

How should the model be conditioned on $z_i$?
What parameters of the model should be shared?

Objective

How should the objective be formed?

Optimization

How should the objective be optimized?
Conditioning on the task

Let’s assume $z_i$ is the one-hot task index.

**Question:** How should you condition on the task in order to share as little as possible?
Conditioning on the task

\[ y = \sum_j 1(z_i = j)y_j \]

\[ \rightarrow \text{independent training within a single network!} \]

with no shared parameters
The other extreme

Concatenate $z_i$ with input and/or activations

all parameters are shared
(except the parameters directly following $z_i$, if $z_i$ is one-hot)
An Alternative View on the Multi-Task Architecture

Split $\theta$ into shared parameters $\theta^{sh}$ and task-specific parameters $\theta^i$

Then, our objective is:

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

Choosing how to condition on $z_i$ equivalent to Choosing how & where to share parameters
Conditioning: Some Common Choices

1. **Concatenation-based** conditioning

   - **Input** → Concatenate → Linear → **Output**
   - Conditioning representation $Z_i$
   - Concatenation-based conditioning simply concatenates the conditioning representation to the input.
   - The result is passed through a linear layer to produce the output.

   These are actually equivalent!

2. **Additive** conditioning

   - **Input** → Linear → **Output**
   - Conditioning representation $Z_i$
   - Conditional biasing first maps the conditioning representation to a bias vector.
   - The bias vector is then added to the input.

**Question:** why are they the same thing? (raise your hand)

Concat followed by a fully-connected layer:
Conditioning: Some Common Choices

3. Multi-head architecture

Why might multiplicative conditioning be a good idea?
- more expressive per layer
- recall: multiplicative gating

4. Multiplicative conditioning

Multiplicative conditioning generalizes independent networks and independent heads.

Conditioning: More Complex Choices

Cross-Stitch Networks. Misra, Shrivastava, Gupta, Hebert ’16

Deep Relation Networks. Long, Wang ’15

Multi-Task Attention Network. Liu, Johns, Davison ’18

Perceiver IO. Jaegle et al. ’21
Unfortunately, these design decisions are like neural network architecture tuning:

- problem dependent
- largely guided by intuition or knowledge of the problem
- currently more of an art than a science
<table>
<thead>
<tr>
<th><strong>Model</strong></th>
<th>How should the model be conditioned on $z_i$?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>What parameters of the model should be shared?</td>
</tr>
<tr>
<td><strong>Objective</strong></td>
<td>How should the objective be formed?</td>
</tr>
<tr>
<td><strong>Optimization</strong></td>
<td>How should the objective be optimized?</td>
</tr>
</tbody>
</table>
Vanilla MTL Objective: \[ \min_{\theta} \sum_{i=1}^{T} L_i(\theta, D_i) \]

Often want to weight tasks differently: \[ \min_{\theta} \sum_{i=1}^{T} w_i L_i(\theta, D_i) \]

How to choose \( w_i \)?

- dynamically adjust throughout training
- manually based on importance or priority

**a. various heuristics**

encourage gradients to have similar magnitudes

(Chen et al. GradNorm. ICML 2018)

**b. optimize for the worst-case task loss**

\[ \min_{\theta} \max_i L_i(\theta, D_i) \]

(e.g. for task robustness, or for fairness)
<table>
<thead>
<tr>
<th>Component</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>How should the model be conditioned on $z_i$?</td>
</tr>
<tr>
<td></td>
<td>What parameters of the model should be shared?</td>
</tr>
<tr>
<td>Objective</td>
<td>How should the objective be formed?</td>
</tr>
<tr>
<td>Optimization</td>
<td>How should the objective be optimized?</td>
</tr>
</tbody>
</table>
Optimizing the objective

Vanilla MTL Objective: \( \min_{\theta} \sum_{i=1}^{T} \mathcal{L}_i(\theta, \mathcal{D}_i) \)

Basic Version:

1. Sample mini-batch of tasks \( \mathcal{B} \sim \{\mathcal{T}_i\} \)
2. Sample mini-batch datapoints for each task \( \mathcal{D}_i^b \sim \mathcal{D}_i \)
3. Compute loss on the mini-batch: \( \mathcal{L}(\theta, \mathcal{B}) = \sum_{\mathcal{T}_k \in \mathcal{B}} \mathcal{L}_k(\theta, \mathcal{D}_k^b) \)
4. Backpropagate loss to compute gradient \( \nabla_{\theta} \mathcal{L} \)
5. Apply gradient with your favorite neural net optimizer (e.g. Adam)

Note: This ensures that tasks are sampled uniformly, regardless of data quantities.

Tip: For regression problems, make sure your task labels are on the same scale!
Challenges
Challenge #1: Negative transfer

Negative transfer: Sometimes independent networks work the best.

<table>
<thead>
<tr>
<th>Multi-Task CIFAR-100</th>
<th>% accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>task specific, 1-fc (Rosenbaum et al., 2018)</td>
<td>42</td>
</tr>
<tr>
<td>task specific, all-fc (Rosenbaum et al., 2018)</td>
<td>49</td>
</tr>
<tr>
<td>cross stitch, all-fc (Misra et al., 2016b)</td>
<td>53</td>
</tr>
<tr>
<td>independent</td>
<td>67.7</td>
</tr>
</tbody>
</table>

(Yu et al. Gradient Surgery for Multi-Task Learning. 2020)

Why?

- optimization challenges
  - caused by cross-task interference
  - tasks may learn at different rates
- limited representational capacity
  - multi-task networks often need to be much larger than their single-task counterparts
If you have negative transfer, **share less** across tasks.

It’s not just a binary decision!

$$\min_{\theta^{sh}, \theta^1, \ldots, \theta^T} \sum_{i=1}^{T} \mathcal{L}_i(\{\theta^{sh}, \theta^i\}, \mathcal{D}_i) + \sum_{t'=1}^{T} \|\theta^t - \theta''\|$$

“soft parameter sharing”

+ allows for more fluid degrees of parameter sharing

- yet another set of design decisions / hyperparameters
Challenge #2: Overfitting

You may not be sharing enough!

Multi-task learning <-> a form of regularization

**Solution:** Share more.
Challenge #3: What if you have a lot of tasks?

Should you train all of them together? Which ones will be complementary?

**The bad news:** No closed-form solution for measuring task similarity.

**The good news:** There are ways to approximate it from one training run.

Fifty, Amid, Zhao, Yu, Anil, Finn. *Efficiently Identifying Task Groupings for Multi-Task Learning*. 2021
Plan for Today

Multi-Task Learning
- Problem statement
- Models, objectives, optimization
- Challenges
- Case study of real-world multi-task learning

Transfer Learning
- Pre-training & fine-tuning
Case study

**Recommending What Video to Watch Next: A Multitask Ranking System**

Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, Ed Chi

Google, Inc.

{zhezhao,lichan,liwei,jilinc,aniruddhnath,shawnandrews,aditeek,nlogn,xinyang,edchi}@google.com

**Goal:** Make recommendations for YouTube

![Figure 4: Recommending what to watch next on YouTube.](image)
Case study

**Recommending What Video to Watch Next: A Multitask Ranking System**

Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, Ed Chi
Google, Inc.
{zhezhao,lichan,liwei,jilinc,aniruddhnath,shawnandrews,aditeek,nlogn,xinyang,edchi}@google.com

**Goal:** Make recommendations for YouTube

- videos that users will rate highly
- videos that users they will share
- videos that user will watch

**Conflicting objectives:**

**Implicit bias caused by feedback:**
user may have watched it because it was recommended!
Framework Set-Up

**Input:** what the user is currently watching (query video) + user features

1. Generate a few hundred of candidate videos
2. Rank candidates
3. Serve top ranking videos to the user

**Candidate videos:** pool videos from multiple candidate generation algorithms
- matching topics of query video
- videos most frequently watched with query video
- And others

**Ranking:** central topic of this paper
The Ranking Problem

Input: query video, candidate video, user & context features

Model output: engagement and satisfaction with candidate video

Engagement:
- binary classification tasks like clicks
- regression tasks for tasks related to time spent

Satisfaction:
- binary classification tasks like clicking “like”
- regression tasks for tasks such as rating

Weighted combination of engagement & satisfaction predictions -> ranking score
score weights manually tuned

Question: Are these objectives reasonable? What are some of the issues that might come up?
The Architecture

Basic option: “Shared-Bottom Model"
(i.e. multi-head architecture)

-> harm learning when correlation
    between tasks is low
Instead: use a form of soft-parameter sharing “Multi-gate Mixture-of-Experts (MMoE)"

Allow different parts of the network to “specialize” expert neural networks $f_i(x)$

Decide which expert to use for input $x$, task $k$:

$$g^k(x) = \text{softmax}(W_{g^k}x)$$

Compute features from selected expert:

$$f^k(x) = \sum_{i=1}^{n} g_{(i)}^k(x) f_i(x)$$

Compute output:

$$y_k = h^k(f^k(x)),$$
Experiments

Set-Up

- Implementation in TensorFlow, TPUs
- Train in **temporal order**, running training continuously to consume newly arriving data
- **Offline** AUC & squared error metrics
- **Online A/B testing** in comparison to production system
  - live metrics based on time spent, survey responses, rate of dismissals
- Model **computational efficiency** matters

Results

<table>
<thead>
<tr>
<th>Model Architecture</th>
<th>Number of Multiplications</th>
<th>Engagement Metric</th>
<th>Satisfaction Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared-Bottom</td>
<td>3.7M</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>Shared-Bottom</td>
<td>6.1M</td>
<td>+0.1%</td>
<td>+1.89%</td>
</tr>
<tr>
<td>MMoE (4 experts)</td>
<td>3.7M</td>
<td>+0.20%</td>
<td>+1.22%</td>
</tr>
<tr>
<td>MMoE (8 Experts)</td>
<td>6.1M</td>
<td>+0.45%</td>
<td>+3.07%</td>
</tr>
</tbody>
</table>

*Table 1: YouTube live experiment results for MMoE.*

Found 20% chance of gating polarization during distributed training -> use drop-out on experts
Plan for Today

Multi-Task Learning
- Problem statement
- Models & training
- Challenges
- Case study of real-world multi-task learning

Transfer Learning
- Pre-training & fine-tuning
Multi-Task Learning vs. Transfer Learning

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

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

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

Key assumption: Cannot access data $D_a$ during transfer.

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

Question: In what settings might transfer learning make sense?
(answer in chat or raise hand)
Transfer learning via fine-tuning

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

(typically for many gradient steps)

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

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.

<table>
<thead>
<tr>
<th>Pre-trained Dataset</th>
<th>PASCAL</th>
<th>SUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</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>

Some common practices
- 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)

Pre-trained parameters

training data
for new task
Universal Language Model Fine-Tuning for Text Classification. Howard, Ruder. ‘18

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).

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

Upcoming lectures: few-shot learning via meta-learning
Plan for Today

Multi-Task Learning
- Problem statement
- Models, objectives, optimization
- Challenges
- Case study of real-world multi-task learning

Transfer Learning
- Pre-training & fine-tuning

Goals for by the end of lecture:
- Know the key design decisions when building multi-task learning systems
- Understand the difference between multi-task learning and transfer learning
- Understand the basics of transfer learning
Reminders

Next time: Meta-learning problem statement, Black-box meta-learning, GPT-3