Multi-Task Learning Basics

CS 330
Logistics

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

High-resolution feedback starts this week — we value your feedback!

**Project ideas**: sharing ideas from survey, project titles from past years
Plan for Today

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

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

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

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

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: \( \mathcal{T}_i \triangleq \{ p_i(x), p_i(y|x), L_i \} \)

data generating distributions

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

will use \( \mathcal{D}_i \) as shorthand for \( \mathcal{D}_i^{\text{tr}} \):

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. face 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
Vanilla MTL Objective
\[
\min_\theta \sum_{i=1}^{T} L_i(\theta, D_i)
\]

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
What parameters of the model should be shared?

Model: How should the model be conditioned on $z_i$?

How should the objective be formed?

Objective

How should the objective be optimized?

Optimization
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 $\mathbf{z}_i$ with input and/or activations

all parameters are shared
(except the parameters directly following $\mathbf{z}_i$, if $\mathbf{z}_i$ 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

These are actually equivalent!

2. **Additive** conditioning

*Question:* why are they the same thing? (think-pair-share)

Concat followed by a fully-connected layer:

*Diagram sources:* distill.pub/2018/feature-wise-transformations/
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
What parameters of the model should be shared?

How should the model be conditioned on $z_i$?

How should the objective be formed?

How should the objective be optimized?

Model

Objective

Optimization
Vanilla MTL objective:
\[
\min_{\theta} \sum_{i=1}^{T} \mathcal{L}_i(\theta, D_i)
\]

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

How to choose $w_i$?
- manually based on importance or priority
- *dynamically* adjust throughout training

a. various heuristics
encourage gradients to have similar magnitudes
(Chen et al. GradNorm. ICML 2018)

b. optimize for the worst-case task loss
\[
\min_{\theta} \max_{i} \mathcal{L}_i(\theta, D_i)
\]
(e.g. for task robustness, or for fairness)
How should the model be conditioned on $z_i$?

What parameters of the model should be shared?

How should the objective be formed?

How should the objective be optimized?
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: \( \hat{\mathcal{L}}(\theta, \mathcal{B}) = \sum_{\mathcal{T}_k \in \mathcal{B}} \mathcal{L}_k(\theta, \mathcal{D}_k^b) \)
4. Backpropagate loss to compute gradient \( \nabla_\theta \hat{\mathcal{L}} \)
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

multi-head architectures
cross-stitch architecture
independent training
If you have negative transfer, **share less** across tasks.

It’s not just a binary decision!

\[
\min_{\theta^s, \theta^1, \ldots, \theta^T} \sum_{i=1}^{T} \mathcal{L}_i(\{\theta^s, \theta^i\}, \mathcal{D}_i) + \lambda \sum_{i'=1}^{T} ||\theta^i - \theta^{i'}||
\]

“soft parameter sharing”

+ allows for more fluid degrees of parameter sharing
- yet another set of design decisions / hyperparameters
- more memory intensive
Challenge #2: Overfitting

You may not be sharing enough!

Multi-task learning $\leftrightarrow$ 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.


Xie, Pham, Dong, Du, Liu, Lu, Liang, Le, Ma, Yu. *DoReMi: Optimizing Data Mixtures Speeds Up Language Model Pretraining.* 2023
A task: \( \mathcal{T}_i \triangleq \{ p_i(x), y_i(x), \mathcal{L}_i \} \)

Corresponding datasets: \( \mathcal{D}_{i, tr}, \mathcal{D}_{i, test} \)

Model Architecture
- multiplicative vs. additive conditioning on \( z_i \)
- share more vs. less depending on observed transfer

Objective & Optimization

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

- choosing task weights
- stratified mini-batches
Plan for Today

Multi-Task Learning
- Problem statement
- Models, objectives, optimization
- Challenges
- Case study of real-world multi-task learning
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.
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 related to time spent

Satisfaction:
- binary classification tasks like clicking “like”
- regression 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)

- harms 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
- **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
Lecture Recap

- Multi-task learning learns neural network conditioned on task descriptor $z_i$
- Choice of task weighting $w_i$ affects prioritization of tasks.
- 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.

Goals for by the end of lecture:
- Understand the key design decisions when building multi-task learning systems
Reminders

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

**Next time:** Transfer learning basics, meta-learning problem statement