Lecture 16: Distribution shift

CS 182 (“Deep Learning”)
Today’s lecture

• This week, we focus on the general problem setting of distribution shift: when the test data comes from a different distribution than the training data.

• The real world is full of distribution shift; often it is benign, sometimes it is harmful.

• When models encounter harmful shifts, not only may their performance/accuracy become worse, but they may also exhibit other types of degradations.
  
  • E.g., worse calibration, worse fairness, …

• Most of the examples we will use are from computer vision (image classification in particular), but there are many other domains that are worthy of greater study.
Recall: true risk and empirical risk

- **Risk** is defined as expected loss: $R(\theta) = \mathbb{E}[\ell(\theta; X, Y)]$

- This is sometimes called **true risk** to distinguish from empirical risk below

- **Empirical risk** is the average loss on the training set: $\hat{R}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(\theta; x_i, y_i)$

- Supervised learning is oftentimes **empirical risk minimization (ERM)**

- Why (and when) does ERM make sense as a learning objective?
The ERM assumption

• ERM is based on the assumption that the test data distribution is the same as the training data distribution

• Under this assumption (and some others involving, e.g., regularization), we can derive generalization bounds of how well we expect models to generalize

  • Even for deep neural networks! This is an active area of research

• This simplifying assumption is used by almost all supervised learning methods

• This assumption was also once referred to as “the big lie of machine learning” by Prof. Zoubin Ghahramani, Sr. Director of Google Brain
Distribution shift in the real world

- Distribution shift in the real world is not the *exception*, it’s the *norm*

- E.g.: in continuous deployment settings, your model will likely encounter future scenarios not represented in the training data

- E.g.: if your model interacts with end users, some users will likely be atypical and will challenge your model in unpredictable ways
Characterizing real-world distribution shift
Distribution shift benchmarks

• In designing benchmarks for distribution shift, we have multiple objectives
  • We want benchmarks that are diverse and representative of real applications
  • We also want benchmarks that are easy to use and evaluate on
  • Let’s look at two general examples that prioritize these objectives somewhat differently: ImageNet challenge test sets and the WILDS benchmark
  • These examples are meant to be illustrative and representative, not exhaustive! Presenting an exhaustive list would take a very long time
ImageNet challenge test sets

- ImageNet challenge test sets are a popular way to measure model **robustness** to different distribution shifts.
- These test sets are designed to **stress test** models by simulating extreme or highly unusual events (stressors).
- These test sets contain the same classes as ImageNet (or a subset), therefore any model trained on ImageNet can easily be evaluated on these test sets.
- And because so much deep learning research focuses on ImageNet, these test sets are widely used.
ImageNet challenge test sets

- **ImageNet-C**
  - Gaussian Noise
  - Shot Noise
  - Impulse Noise
  - Defocus Blur
  - Flicker
  - JPEG
  - Motion Blur
  - Zoom Blur
  - Snow
  - Frost
  - Fog
  - Brightness
  - Contrast
  - Elastic
  - Pixelate
  - JPEG

- **ImageNet-R**
  - Painting
  - Origami

- **ImageNet-A**
  - Chair by rotation
  - Chair by background
  - Chair by viewpoint

- **ImageNet-Sketch**
  - Stylized ImageNet
The **WILDS** benchmark

https://wilds.stanford.edu

- Having easy to use and standardized challenge test sets is important
  - But is it the full picture?
- Are they representative of the distribution shift problems faced by practitioners?
- **WILDS** aims to curate a suite of problems that faithfully represent how distribution shift manifests in real world applications
  - E.g., shifts resulting from medical images collected from a different hospital at test time, or shifts caused by deploying models into different countries
## The WILDS benchmark

**https://wilds.stanford.edu**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>iWildCam</th>
<th>Camelyon17</th>
<th>RxRx1</th>
<th>OGB-MolPCBA</th>
<th>GlobalWheat</th>
<th>CivilComments</th>
<th>FMoW</th>
<th>PovertyMap</th>
<th>Amazon</th>
<th>Py150</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Train example</strong></td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>Test example</strong></td>
<td><img src="image12.png" alt="Image" /></td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
<td><img src="image21.png" alt="Image" /></td>
</tr>
<tr>
<td>Domain (d)</td>
<td>camera</td>
<td>hospital</td>
<td>batch</td>
<td>scaffold</td>
<td>location, time</td>
<td>demographic</td>
<td>time, region</td>
<td>country, rural-urban</td>
<td>user</td>
<td>git repository</td>
</tr>
</tbody>
</table>

```python
import numpy as np
...
norm=np.__
```

```python
import subprocess
as sp
p=sp.Popen()
stdout=p.__
```
In NLP: the ANLI dataset

- *Natural language inference* is the task of determining if a premise sentence and hypothesis sentence are related through contradiction, neutrality, or entailment.

- The **adversarial natural language inference (ANLI) dataset** consists of crowdsourced hypotheses written to fool state-of-the-art models.

- To construct the dataset: an annotator is asked to write a hypothesis given a premise and a condition (contradiction, neutrality, or entailment).
  - If the model correctly predicts the condition, the annotator is asked to try again.
  - If the model predicts incorrectly, the hypothesis is verified by other annotators.
Robustifying against distribution shift
Improving model robustness

• How do we actually make models more robust to distribution shift?

• For WILDS, the training datasets come with additional information (domains) which we can leverage — more on this next time.

• For the ImageNet challenge test sets, the story is different — we do not get any additional information for training as part of the problem statement.

• Here, some techniques have proved quite useful for improving robustness:
  • Training larger models on larger, more diverse datasets (perhaps unsurprising).
  • Using heavy data augmentations and alternative/additional training objectives.
Training larger models on larger datasets
Improves “robustness”?  

Pretrained Transformers Improve Out-of-Distribution Robustness

<table>
<thead>
<tr>
<th>Pretrained Transformers Improve Out-of-Distribution Robustness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dan Hendrycks(^1)*</td>
</tr>
<tr>
<td>Adam Dziedzic(^3)</td>
</tr>
<tr>
<td>(^1)UC Berkeley</td>
</tr>
<tr>
<td>{hendrycks,ericwallace,dawnsong}@berkeley.edu</td>
</tr>
</tbody>
</table>

Robustness properties of Facebook’s ResNeXt WSL models

<table>
<thead>
<tr>
<th>Emin Orhan</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:eo41@nyu.edu">eo41@nyu.edu</a></td>
</tr>
<tr>
<td>New York University</td>
</tr>
</tbody>
</table>

Table 4: Top-1 accuracy and confidence miscalibration scores on ImageNet-A. Note that lower RMS-CE and higher AURRA values indicate better calibrated models. On all three metrics, the largest WSL model performs the best.

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1 acc.</th>
<th>RMS-CE</th>
<th>AURRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>resnext101_32x8d</td>
<td>10.2</td>
<td>54.5</td>
<td>12.3</td>
</tr>
<tr>
<td>resnext101_32x8d_wsl</td>
<td>45.4</td>
<td>26.8</td>
<td>66.3</td>
</tr>
<tr>
<td>resnext101_32x16d_wsl</td>
<td>53.1</td>
<td>22.8</td>
<td>75.0</td>
</tr>
<tr>
<td>resnext101_32x32d_wsl</td>
<td>58.1</td>
<td>19.0</td>
<td>80.2</td>
</tr>
<tr>
<td>resnext101_32x48d_wsl</td>
<td>61.0</td>
<td>17.6</td>
<td>82.4</td>
</tr>
</tbody>
</table>
Data augmentations

- **Mixup** produces element-wise convex combinations of data points and improves corruption robustness.

- **AutoAugment** learns complex augmentation strategies from basic data augmentation operations by training tens of thousands of deep neural networks.

- **AugMix** mixes together random augmentations, using many of the same operations from AutoAugment.

- **PixMix** is a recent strategy that mixes in a separate image dataset (such as fractals) and results in consistently good performance across several metrics.
Current state of the art: masked autoencoders

- The current state of the art numbers for ImageNet-C, R, A, and Sketch are obtained with ViT models pretrained with a masked autoencoding objective.

- Supervised learning on the original ImageNet training set after this pretraining phase leads to the best results amongst models that do not get additional data.

<table>
<thead>
<tr>
<th>dataset</th>
<th>ViT-B</th>
<th>ViT-L</th>
<th>ViT-H</th>
<th>ViT-H(_{448})</th>
<th>prev best</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN-Corruption ↓[27]</td>
<td>51.7</td>
<td>41.8</td>
<td><strong>33.8</strong></td>
<td>36.8</td>
<td>42.5 [32]</td>
</tr>
<tr>
<td>IN-Adversarial [28]</td>
<td>35.9</td>
<td>57.1</td>
<td>68.2</td>
<td><strong>76.7</strong></td>
<td>35.8 [41]</td>
</tr>
<tr>
<td>IN-Rendition [26]</td>
<td>48.3</td>
<td>59.9</td>
<td>64.4</td>
<td><strong>66.5</strong></td>
<td>48.7 [41]</td>
</tr>
<tr>
<td>IN-Sketch [60]</td>
<td>34.5</td>
<td>45.3</td>
<td>49.6</td>
<td><strong>50.9</strong></td>
<td>36.0 [41]</td>
</tr>
</tbody>
</table>
Detecting distribution shift
Anomaly and out-of-distribution detection

- Why do we care about detecting anomalies and out-of-distribution (OOD) data?
- When machine learning systems encounter an anomaly, we may wish to trigger a “conservative” mode or failsafe in order to avoid catastrophes.
- We may wish to detect malicious use of machine learning systems, e.g., hackers.
- Or other potential dangers, e.g., dangerous novel microorganisms.
Anomaly detection: the basics

• We would like for our model to assign an anomaly score to every input $\mathbf{x}$ — the higher the score, the more anomalous the model thinks the example is.

• An intuitive idea would be to try and learn a model of $p(\mathbf{x})$ (a generative model) and treat an $\mathbf{x}$ as anomalous if it has low $p(\mathbf{x})$ according to the model.

  • This currently does not work well! Modern deep generative models often still do poorly at anomaly detection using this scheme for complex input spaces.

• There are some ways to make deep generative models useful for anomaly detection, though they are more complex and require additional assumptions.
A simple baseline for anomaly detection

- A better approach that does not involve training a generative model is to use the model’s **confidence** \( \max_k p_{\theta}(y = k \mid x) \) to detect anomalies

  - Specifically, use \(- \max_k p_{\theta}(y = k \mid x)\) as the anomaly score

  - In some contexts, \(- \max_k z_k\) (negative of max logit) may work better

- This simple baseline works reliably across computer vision, NLP, and speech recognition classification tasks, though it can’t detect *adversarial examples* (next week)

- Some more advanced techniques we don’t have time to discuss, but you can go look up: *likelihood ratios*, *outlier exposure*, *virtual logit matching*
Benchmarks for anomaly detection

- In some sense, there is a much larger “search space” for constructing anomaly detection benchmarks — train a model on one dataset, and treat any other dataset as anomalous

- E.g., train on CIFAR-10, evaluate on SVHN (a digit recognition dataset)

- Or, train on CIFAR-10, evaluate on CIFAR-100; or vice versa
  - The classes between these two datasets are mutually exclusive

- Or, train on ImageNet-22K, evaluate on Species
An aside: evaluating binary classifiers

• We can think of anomaly detection as a binary classification problem

• What might be the issue of just evaluating the accuracy of anomaly detectors?

  • What if we have 1 anomaly, 99 “normal” examples, and a detector that always predicts “usual”? What is its accuracy? Is this a good detector?

• Evaluating anomaly detectors, and binary classifiers in general, often consider more detailed metrics than just accuracy

  • These metrics are generally based on the number of true positives, false positives, true negatives, and false negatives — (usually) covered in CS 189
Model calibration

- Another concept related to the general reliability of machine learning models, but not tied to distribution shift, is model **calibration**

- E.g., consider a weather model that predicts “70% chance of rain” for a certain set of inputs — does it rain for 70% of those inputs?

- We measure calibration by comparing the model’s confidence against its accuracy

- Well calibrated models are more trustworthy, easier to integrate, and more interpretable
Calibration under distribution shift