Lecture 18: Out-of-distribution generalization

Speaker: Stefanie Jegelka

```
“pig” + 0.005 x “airliner”
```
Standard ML setting

training distribution = test distribution
... vs the real world

deploy model on data from a different distribution

e.g.:
  - perturbed data
  - different label distribution
  - other shifts (sequence/graph size, weather, country/city, source of measurement,...)
Concrete Problems in AI Safety

Dario Amodei*  Chris Olah*  Jacob Steinhardt  Paul Christiano
Google Brain  Google Brain  Stanford University  UC Berkeley

John Schulman  Dan Mané
OpenAI  Google Brain

extremely negative reward, and covering both physical and abstract catastrophes, might help in the development of safe exploration techniques for advanced RL systems. Such a suite of environments might serve a benchmarking role similar to that of the bAbI tasks [163], with the eventual goal being to develop a single architecture that can learn to avoid catastrophes in all environments in the suite.

7 Robustness to Distributional Change

All of us occasionally find ourselves in situations that our previous experience has not adequately prepared us to deal with—for instance, flying an airplane, traveling to a country whose culture is very different from ours, or taking care of children for the first time. Such situations are inherently difficult to handle and inevitably lead to some missteps. However, a key (and often rare) skill in dealing with such situations is to recognize our own ignorance, rather than simply assuming that the heuristics and intuitions we’ve developed for other situations will overcome perfectly. Machine
Outline for today

• Adversarial examples and training: small perturbations

• Distributionally robust training

• "Extrapolation"
Adversarial examples
Adversarial examples

“pig”

91% confidence
Adversarial examples

“pig”

91% confidence

+ 0.005 x
Adversarial examples

- ML model predictions are (mostly) accurate but can be brittle

example: Szegedy et al 2013, obtained from https://gradientscience.org/intro_adversarial/
Adversarial examples

Papernot et al 2017, Practical black-box attacks against machine learning
Adversarial stickers
Adversarial stickers

Adversarial stickers

Adversarial stickers

Adversarial examples 3D-printed

image: Athalye et al 2018, Synthesizing robust adversarial examples
Adversarial examples 3D-printed

classified as turtle  classified as rifle  classified as other

image: Athalye et al 2018, Synthesizing robust adversarial examples
Speech recognition example

"it was the best of times, it was the worst of times"

"it is a truth universally acknowledged that a single"
Hmmmm....

• Are our models completely useless?

• Why does this happen?

• Can one prevent it?
History of adversarial examples / brittleness

Legend
- Pioneering work on adversarial machine learning
- Work on security evaluation of learning algorithms
- Work on evasion attacks (a.k.a. adversarial examples)
- \ldots in malware detection (PDF / Android)

2004-2005: pioneering work
- Dalvi et al., KDD 2004
- Lowd & Meek, KDD 2005

2006-2010: Barreno, Nelson, Rubinstein, Joseph, Tygar
The Security of Machine Learning
(and references therein)

2006: Globerson & Roweis, ICML
2009: Kolcz et al., CEAS
2010: Biggio et al., IJMLC

2013: Srndic & Laskov, NDSS

2013: Biggio et al., ECML-PKDD
- demonstrated vulnerability of nonlinear algorithms
to gradient-based evasion attacks, also under limited knowledge
Main contributions:
1. gradient-based adversarial perturbations (against SVMs and neural nets)
2. projected gradient descent / iterative attack (also on discrete features from malware data)
3. transfer attack with surrogate/substitute model
4. maximum-confidence evasion (rather than minimum-distance evasion)

2014: Biggio et al., IEEE TKDE
Main contributions:
- framework for security evaluation of learning algorithms
- attacker’s model in terms of goal, knowledge, capability

2014: Srndic & Laskov, IEEE S&P
used Biggio et al.’s ECML-PKDD ’13 gradient-based evasion attack to demonstrate vulnerability of nonlinear PDF malware detectors

Biggio & Roli 2018, Wild Patterns: ten years after the rise of adversarial machine learning
How do you create an adversarial example?

• want: small perturbation that does not change meaning to a human, but to ML model

• model outputs $P_\theta(y|\mathbf{x})$ (softmax)

• adversarial example:

$$\max_{\delta \in \Delta} P_\theta(y_{\text{target}}|\mathbf{x}+\delta)$$

small perturbation, e.g.

$$\Delta = \{\delta \in \mathbb{R}^d | \|\delta\|_\infty < \epsilon\}$$
How do you create an adversarial example?

- want: small perturbation that does not change meaning to a human, but to ML model
- model outputs $P_\theta(y | x)$
- adversarial example:

\[
\max_{\delta \in \Delta} P_\theta(y_{\text{target}} | x + \delta)
\]

- equivalently:

\[
\max_{\delta \in \Delta} \text{Loss}(f_\theta(x + \delta), y)
\]

vs training:

\[
\min_\theta \text{Loss}(f_\theta(x), y)
\]
How to find an adversarial example?

- e.g. Projected gradient ascent (we update data perturbation $\delta$):
  1. take a step in the direction of the gradient:
     $$
     \delta^{(t+1)} = \delta^{(t)} + \eta \cdot \nabla_\delta P_\theta(y_{\text{target}} | x + \delta)
     $$
  2. project the result back into the feasible set $\Delta$
  3. repeat steps 1 & 2
How to “defend” against adversarial examples?

Recall:

- Adversarial example versus standard training:

$$\max_{\delta \in \Delta} \text{Loss}(f_\theta(x + \delta), y) \quad \text{versus} \quad \min_{\theta} \text{Loss}(f_\theta(x), y)$$
How to “defend” against adversarial examples?

- **Standard training:**
  
  via (stochastic) gradient descent

- **Adversarial training / robust optimization:**

  \[
  \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(f_{\theta}(x^{(i)}), y^{(i)})
  \]

  \[
  \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{\delta \in \Delta} \text{Loss}(f_{\theta}(x^{(i)} + \delta), y^{(i)})
  \]

  “adaptive data augmentation”
Adversarial training

• Key question: what is the gradient of the robust loss?

\[ \mathcal{L}_{\text{robust}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \max_{\delta \in \Delta} \text{Loss}\left(f_{\theta}(x^{(i)} + \delta), y^{(i)}\right) \]

• **Danskin’s theorem**: find the optimal solution \( \delta^* \) for the inner maximization problem. Then the gradient wrt \( \theta \) is:

\[ \nabla_{\theta} \mathcal{L}_{\text{robust}}(\theta) = \frac{1}{n} \nabla_{\theta} \left[ \sum_{i=1}^{n} \text{Loss}\left(f_{\theta}(x^{(i)} + \delta^*), y^{(i)}\right) \right] \]
Adversarial training with stochastic gradient descent

repeat until convergence:

1. sample a data point \((x, y)\)

2. compute the optimal adversarial perturbation \(\delta^*\) \(\text{(approximately)}\)

3. compute the gradient \(g = \nabla_{\theta} \text{Loss}(f_{\theta}(x + \delta^*), y)\)

4. update \(\theta\) with the gradient \(g\)
What do adversarial examples tell us?
What do adversarial examples tell us?

- something about the input “features” that are critical for the model’s decision
- Example:

![Training data: classify 4 vs 9](image)
What do adversarial examples tell us?

• something about the input “features” that are critical for the model’s decision
• Example:

![Training data: classify 4 vs 9](image)

![Adversarial perturbations](image)

_images: Hongzhou Lin_
Predictive features

- Many features may be correlated with the label and hence predictive and help with accuracy, beyond what humans would use.
Where do these correlations come from?

• e.g., the way we create datasets …

“Fish” from the ImageNet training set
Where do these correlations come from?

• e.g. the way we create datasets…

**Ideal world:**
- Real-world images
- Expert annotators
- Perfect annotations
- Meaningful benchmark

**Real world:**
- Flickr/scraped images
- Automated + Crowd Labels
- Noisy, biased annotations
- Easy-to-optimize benchmark
It’s all “shortcuts”

- Shortcuts: features correlated with label in the training data, but not under realistic distribution shifts
- Models will use them and not generalize if features are no longer correlated

[Diagram illustrating the concept of shortcuts with examples of images and their classifications by typical humans and neural networks.]

Illustration: Geirhos et al 2020
It’s all “shortcuts”

• Shortcuts: features correlated with label in the training data, but not under realistic distribution shifts

• Models will use them and not generalize if features are no longer correlated

• This is a reason due to data, not model: hence, adversarial examples transfer across models trained on the same dataset
Examples of “shortcuts”

A herd of sheep grazing on a lush green hillside
Tags: grazing, sheep, mountain, cattle, horse

NeuralTalk2: A flock of birds flying in the air
Microsoft Azure: A group of giraffe standing next to a tree

Examples of “shortcuts”
Examples of “shortcuts”

“CNNs were able to detect where an x-ray was acquired [...] and calibrate predictions accordingly.”

“...if an image had a ruler in it, the algorithm was more likely to call a tumor malignant...”

[Esteva et al. 2017]

“CNNs were able to detect where an x-ray was acquired [...] and calibrate predictions accordingly.”

[Zech et al. 2018]

not all predictive patterns are desirable
Many more…

*i.id.*

- Domain shift: Wang 2018
- Adversarial examples: Szegedy 2013
- Distortions: Dodge 2019, Pose: Alcorn 2019, Texture: Geirhos 2019, Background: Beery 2018

*o.o.d.*

*Same category for humans but not for DNNs (intended generalization)*

*Same category for DNNs but not for humans (unintended generalization)*

Illustration: Geirhos et al 2020, Shortcut learning in deep neural networks
Transformers Learn Shortcuts to Automata

Bingbin Liu\textsuperscript{1*}  Jordan T. Ash\textsuperscript{2}  Surbhi Goel\textsuperscript{2,3}  Akshay Krishnamurthy\textsuperscript{2}  Cyril Zhang\textsuperscript{2}

\textsuperscript{1}Carnegie Mellon University  \textsuperscript{2}Microsoft Research NYC  \textsuperscript{3}University of Pennsylvania
bingbinl@cs.cmu.edu, \{ash.jordan, goel.surbhi, akshaykr, cyrilzhang\}@microsoft.com

Abstract

Algorithmic reasoning requires capabilities which are most naturally understood through recurrent models of computation, like the Turing machine. However, Transformer models, while lacking recurrence, are able to perform such reasoning using far fewer layers than the number of reasoning steps. This raises the question: \textit{what solutions are these shallow and non-recurrent models finding?} We investigate this question in the setting of learning automata, discrete dynamical systems naturally suited to recurrent modeling and expressing algorithmic tasks. Our theoretical results completely characterize \textit{shortcut solutions}, whereby a shallow Transformer with only $o(T)$ layers can exactly replicate the computation of an automaton on an input sequence of length $T$. By representing automata using the algebraic structure of their underlying transformation semigroups, we obtain $O(\log T)$-depth simulators for all automata and $O(1)$-depth simulators for all automata whose associated groups are \textit{simple}. Synthetic experiments by training Transformers to simulate a wide variety of shortcut solutions can be learned via standard training. We further investigate solutions and propose potential mitigations.

parallel solutions generalize within-distribution, but not out-of-distribution
Effect of adversarial training

- Model output should be stable under adversarial perturbations
  => teaches invariance to non-robust features
Effect of adversarial training

Loss gradients with respect to input pixels (most important features) show: robust model relies less on “non-robust” features, and more on human-intuitive features.

Adversarial examples for standard and robust models

(Tsipras et al. 2019, Robustness may be at odds with accuracy.)
Effect of adversarial training: transfer learning

- adversarially trained models transfer better to other datasets

(Salman et al. 2020, Do adversarially robust ImageNet models transfer better?)
Distributionally robust optimization
A twist: distributionally robust optimization

- So far: allowed to perturb each datapoint by a limited amount

- Alternative: we can perturb the entire training distribution (sample) by a certain amount, together
Distributionally robust optimization

• Standard training:
\[
\frac{1}{n} \sum_{i=1}^{n} \text{Loss}(f_\theta(x^{(i)}), y^{(i)}) = \mathbb{E}_{(x,y) \sim P} [\text{Loss}(f_\theta(x), y)]
\]

• Distributionally robust optimization (DRO):
\[
\min_{\theta} \max_{Q, D(Q, P_n) < \epsilon} \mathbb{E}_{(x,y) \sim Q} [\text{Loss}(f_\theta(x), y)]
\]
e.g. re-weight or perturb training data points

• Various choices of measuring “distance” between probability distributions: \(\chi^2\)-distance, Wasserstein distance, maximum mean discrepancy (MMD)…

allow a small perturbation of training sample (discrete distribution)
DRO and generalization

\[
\min_{\theta} \max_{Q, D(\mathcal{Q}, \mathcal{P}_n) < \varepsilon} \mathbb{E}_{(x, y) \sim Q}[\text{Loss}(f_{\theta}(x), y)]
\]

- DRO optimizes for a set of training data sets/distributions
- Say underlying data distribution is \( \mathcal{P} \)
- Empirical training data is \( \hat{\mathcal{P}}_n \)
- If \( D(\mathcal{P}, \hat{\mathcal{P}}_n) < \varepsilon \), then we are guaranteed to perform well on \( \mathcal{P} \) too, i.e., generalize!
Another connection to generalization: one can show that DRO with different divergences ("distances" between probabilities) corresponds to different regularization methods!

E.g. ridge regression (MMD), penalizing \( \| \nabla_x \ell(f(x)) \| \) (Wasserstein distance), penalizing \( \text{Var}_{P_n}(\ell(f(x))) \) (\( \chi^2 \)-distance).
Application: DRO and class imbalance

- Assume population has K sub-groups (example: K=2).
- Usually: minimize "Empirical Risk" (average error)

\[
\min_{\theta} \frac{1}{n} \left( \sum_{i \text{ in group 1}} \text{Loss}(x_i; \theta) + \sum_{j \text{ in group 2}} \text{Loss}(x_j; \theta) \right)
\]

- Here, 50% error on minority group makes only 10% average error. (+ statistical patterns for minority may be different)
- We can "ignore" minority group and still get decent loss!
DRO and class imbalance

- Idea: automatically re-weight data via DRO => pay more attention to minority class

\[
\min_{\theta} \max_{Q, D(Q, P_n) < \epsilon} \mathbb{E}_{(x, y) \sim Q} [\text{Loss}(f_{\theta}(x), y)]
\]

(Hashimoto et al. 2018, Fairness without demographics in repeated loss minimization)
Inductive biases and extrapolation
Predicting on data in a very different range...
Predicting on data in a very different range...

- Prediction depends on what function we are fitting (here, e.g., activation function)
Example: ReLU fully connected network

• Neural network extrapolates as a directionally linear function

Xu et al 2021, How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks.
Example: size generalization for graphs

- E.g.: train on small graphs, test on large graphs
  train on small degree graphs, test on large degree graphs

- ReLU MLPs parameterize the aggregation function in GNNs:

\[
\begin{align*}
    h_v^{(t)} &= \sum_{u \in N(v)} \text{FNN}(h_u^{(t-1)}, h_v^{(t-1)}, w(u, v))
\end{align*}
\]
Example: learning a shortest path algorithm with GNNs

Shortest Path: 
\[ \text{dist}[t][v] = \min_{u \in \mathcal{N}(v)} \text{dist}[t - 1][u] + w(u, v) \]

GNN: 
\[ h_v^{(t)} = \sum_{u \in \mathcal{N}(v)} \text{FNN}(h_u^{(t-1)}, h_v^{(t-1)}, w(u, v)) \]

Need FNN (MLP) to be nonlinear!

\[ \text{dist}[t][v] = \min_{u \in \mathcal{N}(v)} \text{dist}[t - 1][u] + w(u, v) \]

\[ h_v^{(t)} = \sum_{u \in \mathcal{N}(v)} \text{FNN}(h_u^{(t-1)}, h_v^{(t-1)}, w(u, v)) \]

\[ \text{dist}[t][v] = \min_{u \in \mathcal{N}(v)} \text{dist}[t - 1][u] + w(u, v) \]

\[ h_v^{(t)} = \sum_{u \in \mathcal{N}(v)} \text{FNN}(h_u^{(t-1)}, h_v^{(t-1)}, w(u, v)) \]

\[ \text{dist}[t][v] = \min_{u \in \mathcal{N}(v)} \text{dist}[t - 1][u] + w(u, v) \]

\[ h_v^{(t)} = \sum_{u \in \mathcal{N}(v)} \text{FNN}(h_u^{(t-1)}, h_v^{(t-1)}, w(u, v)) \]

\[ \text{dist}[t][v] = \min_{u \in \mathcal{N}(v)} \text{dist}[t - 1][u] + w(u, v) \]

\[ h_v^{(t)} = \sum_{u \in \mathcal{N}(v)} \text{FNN}(h_u^{(t-1)}, h_v^{(t-1)}, w(u, v)) \]
Example: learning a shortest path algorithm with GNNs

Shortest Path:
\[ \text{dist}[t][v] = \min_{u \in \mathcal{N}(v)} \text{dist}[t - 1][u] + w(u, v) \]  

GNN:
\[ h_v^{(t)} = \sum_{u \in \mathcal{N}(v)} \text{FNN}(h_u^{(t-1)}, h_v^{(t-1)}, w(u, v)) \]
\[ h_v^{(t)} = \max_{u \in \mathcal{N}(v)} \text{FNN}(h_u^{(t-1)}, h_v^{(t-1)}, w(u, v)) \]

Task-specific nonlinearities help.
empirically reflected in many works

Need FNN (MLP) to be nonlinear!

Xu et al 2021, How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks.
Side note: when does size generalization work “immediately”?
Summary

• Out-of-distribution generalization: big challenge, but helps understand what NN learn. Many instances:
  
  • adversarial examples / training
  
  • distributional robustness
  
  • “extrapolation” (large shifts)