6.S897/HST.956 Machine Learning for Healthcare

Lecture 24: Robustness to Dataset Shift

Instructors: David Sontag, Peter Szolovits

1 Introduction

This lecture is primarily a discussion on things that can go wrong when deploying a machine learning model, even one that may have trained and tested well prior to deployment.

- Adversarial perturbations of inputs & security vulnerabilities
- Natural changes in the data
- **Machine Learning breaks when the test and training distributions are different**

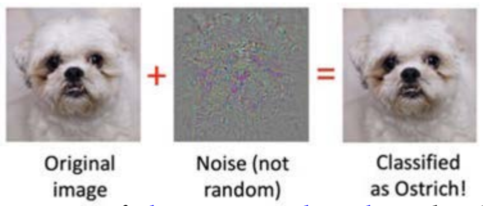
  Example: Assume some data X has been used to predict some label Y, such as predicting Type 2 Diabetes. This model will have difficulty having high accuracy if tested on label ”Diabetes”, which might include Type I.

  – This is because $P(Y|X)$ changes (two distributions $P_0(Y|X)$ vs $P_1(Y|X)$). This is known as *dataset shift*.

  – Today we will focus on what happens when $P(X)$ changes ($P_0(X)$ vs. $P_1(X)$), a phenomenon called *covariate shift*.

1.1 Adversarial Perturbations

- Adding even a small amount of noise to an image can result in a classifier completely mis-classifying the original image. This can be used to incorrectly diagnose patients to get reimbursed by positive results for procedures. [Finlayson et al., ”Adversarial Attacks Against Medical Deep Learning Systems”, Arxiv 1804.05296, 2018]
- Different ways to approach this, both in the context of perturbing the training data (if access to the model is available), or perturbing test inputs (which does not need access to the data)
- A famous example of an adversarial perturbation is the following:



Courtesy of Christian Szegedy et al. Used under CC BY.

Figure 1: A dog (original image) goes from being correctly classified to being mis-classified as an ostrich after an adversarial perturbation even though the original and modified images are indistinguishable to the human eye.

1.2 Natural Perturbations

Data can be non-stationary, e.g the significance of features may change over time. Quite often, different hospitals may use different naming conventions or change their definitions of diseases over time. This is something to keep in mind when training a model in one institution and implementing it in a different institution. For example, the population in central Boston may be older on average than the population in Waltham, and this may come with a different distribution of diseases. A model trained to predict healthcare costs on the younger population may have to change which features it considers most heavily when applied to the older population.

2 Population-level Checks for deployment/transfer

- Every study in an MLHC application should have a "Table 1", reflecting the population demographics for the institution being studied.
- In a case study for a sample of an institution, there should be two columns, one each for the population at large and one more for the sample.

Example: Facial Recognition - take basic information about the differences in faces between one institution and another before starting work at the new institution. i

3 Transfer Learning

Transfer learning is a technique to re-purpose a model trained on one dataset to work on another, related dataset e.g moving a model from UCSF to NYU. You usually have a lot of data from $P(Y|X)$ and a little from $Q(Y|X)$ where P is the distribution the model was trained on and Q is the new distribution to learn. How can we quickly adapt our old model?

3.1 Linear models: original representation, modify weights

One way to adapt a linear model is to learn the old weights using $P(Y|X)$ and then when learning using data from $Q(Y|X)$, instead of using L1 or L2 regularization, use $\|w - w_{old}\|_2^2$ or $\|w - w_{old}\|_1$ where w_{old} denotes the old weights.

3.2 Linear models: manually choose a good shared representation

Another way to adapt a linear model is to come up with a good shared representation that works for both $P(Y|X)$ and $Q(Y|X)$. Take for example a model that we want to apply across two different electronic health record systems, EHR 1 and EHR2 . We have a few options. We could learn models on EHR 1 and apply them to EHR 2, but concepts important in EHR 1 may not appear in EHR 2, and vice versa. We could learn models only using EHR 2, but this results in throwing away a lot of valuable data. Or we could develop a model only on the intersection of elements in EHR 1 and EHR 2, but this could remove the majority of clinical concepts in both EHRs from our model.
The solution is to map semantically similar items to a shared vocabulary as in the below figure.

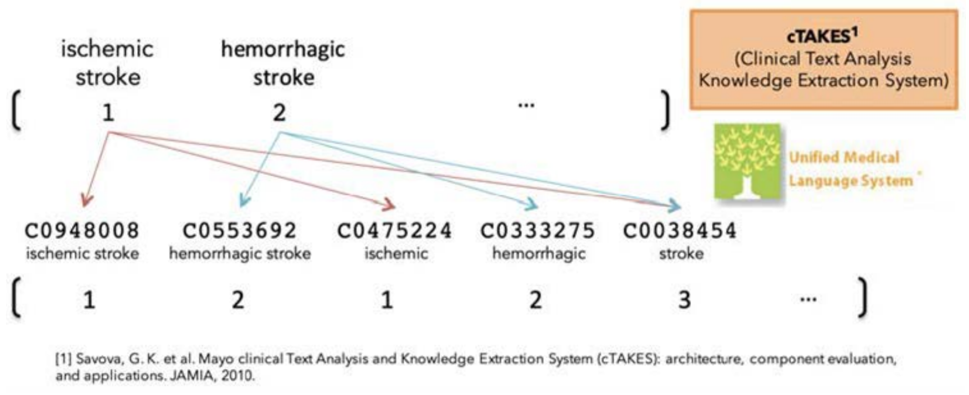
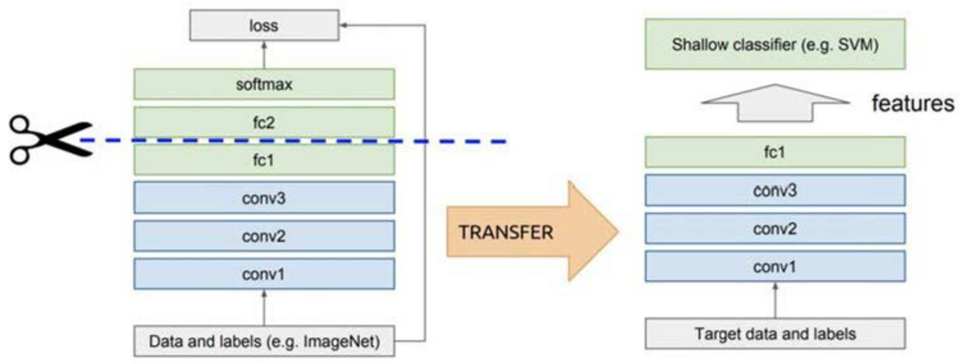

ischemic stroke
hemorrhagic stroke
1
2
...
cTAKES[1] (Clinical Text Analysis Knowledge Extraction System)
Unified Medical Language System
C0948008 ischemic stroke
C0553692 hemorrhagic stroke
C0475224 ischemic
C0333275 hemorrhagic
C0038454 stroke
1 2 1 2 3 ...
[1] Savova, G. K. et al. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation, and applications. JAMIA, 2010.

Figure 2: Mapping EHR specific events to a shared vocabulary. Here, "ischemic stroke" gets mappped to two more general representations, "ischemic" and "stroke" separately, making it more robust.

3.3 Deep models: re-use part of the learned representation, fine-tune

Applying a widely used in computer vision, we take a pre-trained model, chop off the top few layers, and train a new shallow model on the induced representation.



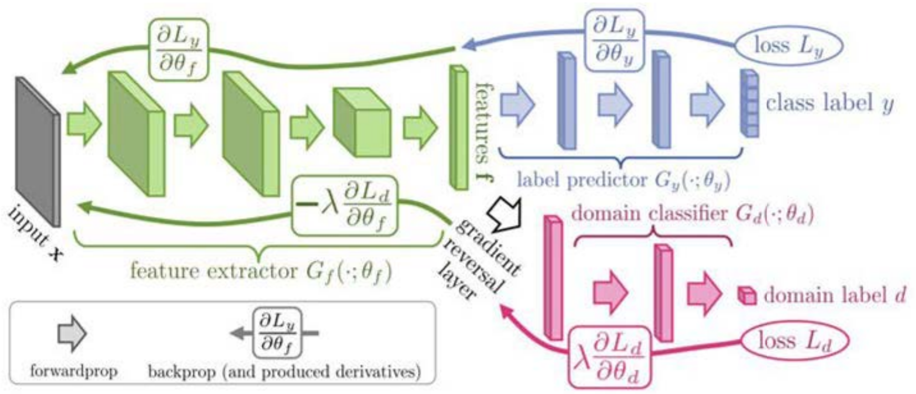
© TelebcomBCN. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use/

Figure 3: An example of cutting off the first few layers of a model and re-purposing the rest of it for transfer learning.

For recurrent neural networks, we carry over the word embeddings, but relearn $S^{s,x}$ so that they change with the new institution.

3.4 Deep models: automatically find a good shared representation

Deep learning models are very good at finding behind-the-scenes representations.


$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features $\mathbf{f}$
label predictor $G_y(\cdot;\theta_y)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
domain classifier $G_d(\cdot;\theta_d)$
input x
gradient reversal layer
feature extractor $G_f(\cdot;\theta_f)$
domain label d
loss L_d
$\lambda \frac{\partial L_d}{\partial \theta_d}$
forwardprop
backprop (and produced derivatives)

© Ganin et al. All rights reserved. This content is excluded from our Creative Commons license. For more information, see https://ocw.mit.edu/help/faq-fair-use/

Figure 4: Guided by learning theory (Ben-David et al. 06), recent work shows how to do domain adaptation without labels in target set:

4 Unsupervised Domain Adaptation

Intuition: learn parameters that minimize the loss under the training domain D_1 and also maintain that there be a small distance betweeen D_1 and D_2. The first few layers of the networks minimize the feature space.

5 Summary: Building a Robust Model

We phrase finding a robust model as a new optimization problem where we minimize θ while maximizing $\delta \in \Delta$ w.r.t. $loss(\theta, x + \delta, y)$ where δ is a perturbation used to increase loss on the correct label y to make the model more robust and protect against adversarial attacks.

References

[BSOM$^+$14] David W Bates, Suchi Saria, Lucila Ohno-Machado, Anand Shah, and Gabriel Escobar. Big data in health care: using analytics to identify and manage high-risk and high-cost patients. *Health Affairs*, 33(7):1123–1131, 2014.

[WSW14] Xiang Wang, David Sontag, and Fei Wang. Unsupervised learning of disease progression models. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 85–94. ACM, 2014.

MIT OpenCourseWare
https://ocw.mit.edu

6.S897 / HST.956 Machine Learning for Healthcare
Spring 2019

For information about citing these materials or our Terms of Use, visit: https://ocw.mit.edu/terms