Systematic Uncertainties and Domain Adaptation

Michael Kagan
SLAC

Hammers and Nails
July, 2017
The Goal

The Higgs Boson!

ATLAS Preliminary

H → ZZ* → 4l
13 TeV, 36.1 fb⁻¹

Events/2.5 GeV

*Data
Higgs (m_H = 125.09 GeV)
ZZ*
tt+V, VVV
Z+jets, tt
Uncertainty

The Higgs Boson!
The Goal

- Want to reject the “No Higgs” hypothesis (red + purple)
- Hopefully measure the size of the Signal
The Goal

ATLAS Preliminary

$H \rightarrow ZZ^* \rightarrow 4l$

13 TeV, 36.1 fb$^{-1}$

- Data
- Higgs ($m_H = 125.09$ GeV)
- ZZ$^*$
- $t\bar{t}+V$, VVV
- $Z$+jets, $t\bar{t}$
- Uncertainty

The Higgs Boson!

- Need accurate predictions for **Signal** and **Background**
The Goal

- And we need to know how wrong these predictions could be... Systematic Uncertainties
Sources of Systematic Uncertainty

- **Experimental**
  - How well we understand the detector and our ability to identify / measure properties of particles given that they are present

- **Theoretical**
  - How well we understand the underlying theoretical model of the physical processes

- **Modeling uncertainties**
  - How well can I constrain backgrounds in alternate data measurements / control regions

<table>
<thead>
<tr>
<th>Source of uncertainty</th>
<th>( \sigma_\mu )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>0.39</td>
</tr>
<tr>
<td>Statistical</td>
<td>0.24</td>
</tr>
<tr>
<td>Systematic</td>
<td>0.31</td>
</tr>
</tbody>
</table>

**Experimental uncertainties**

<table>
<thead>
<tr>
<th>Source of uncertainty</th>
<th>( \sigma_\mu )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jets</td>
<td>0.03</td>
</tr>
<tr>
<td>( E_T^{\text{miss}} )</td>
<td>0.03</td>
</tr>
<tr>
<td>Leptons</td>
<td>0.01</td>
</tr>
<tr>
<td>b-jets</td>
<td>0.09</td>
</tr>
<tr>
<td>c-jets</td>
<td>0.04</td>
</tr>
<tr>
<td>light jets</td>
<td>0.04</td>
</tr>
<tr>
<td>extrapolation</td>
<td>0.01</td>
</tr>
</tbody>
</table>

**Theoretical and modelling uncertainties**

<table>
<thead>
<tr>
<th>Source of uncertainty</th>
<th>( \sigma_\mu )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pile-up</td>
<td>0.01</td>
</tr>
<tr>
<td>Luminosity</td>
<td>0.04</td>
</tr>
<tr>
<td>Signal</td>
<td>0.17</td>
</tr>
<tr>
<td>Floating normalisations</td>
<td>0.07</td>
</tr>
<tr>
<td>Z+jets</td>
<td>0.07</td>
</tr>
<tr>
<td>W+jets</td>
<td>0.07</td>
</tr>
<tr>
<td>( t\bar{t} )</td>
<td>0.07</td>
</tr>
<tr>
<td>Single top-quark</td>
<td>0.08</td>
</tr>
<tr>
<td>Diboson</td>
<td>0.02</td>
</tr>
<tr>
<td>Multijet</td>
<td>0.02</td>
</tr>
<tr>
<td>MC statistical</td>
<td>0.13</td>
</tr>
</tbody>
</table>
Error Propagation

• Random variables $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ distributed according to $p(\mathbf{x})$
  
  - Assume we only know $E[\mathbf{x}] = \mu$ and $\text{Var}[x_i, x_j] = V_{ij}$

• For some function $f(\mathbf{x})$, first order approximation

  $$f(\mathbf{x}) \approx f(\mu) + \sum_{i=1}^{N} \left. \frac{\partial f}{\partial x_i} \right|_{\mathbf{x}=\mu} (x_i - \mu_i)$$

• Then

  $$E[f(\mathbf{x})] \approx f(\mu)$$

  $$\sigma_f^2 = E[(f(\mathbf{x}) - f(\mu))^2] \approx \sum_{i, j=1}^{N} \left[ \frac{\partial f}{\partial x_i} \frac{\partial f}{\partial x_j} \right]_{\mathbf{x}=\mu} V_{ij}$$
Error Propagation

• Often, we only have samples of $x$

• But we know how $x$ varies under some uncertainty
  
  $\rightarrow x \pm \Delta x$

  Where $\Delta x$ is a 1 standard deviation change in $x$

• Then often approximate the error on $f(x)$ with
  
  $\pm \Delta f(x) = f(x \pm \Delta x) - f(x)$

• So we vary $x$ and see how $f(x)$ changes
  
  Often histogram the distributions, showing $\pm \Delta f(x)$ as a band on the histogram bin counts
Hypothesis Testing

\[ \lambda(\mathcal{D}; \theta_0, \theta_1) = \prod_{x \in \mathcal{D}} \frac{p(x|\theta_0)}{p(x|\theta_1)} \]

• Likelihood \( p(x|\theta) \) parameterized by \( \theta \)

• Likelihood ratio used as test statistic for hypothesis testing between null hypothesis \( (\theta_0) \) and alternative hypothesis \( (\theta_1) \)

• Parameterization \( \theta=(\mu, \nu) \) includes parameters of interest \( (\mu) \) and nuisance parameters \( (\nu) \)

[For more details, see Glen Cowan’s talk]
Nuisance Parameters and the Likelihood

\[ L(\mu) = \prod_{i=1}^{N} \frac{(\mu s_i + b_i)^{n_i}}{n_i!} e^{-(\mu s_i + b_i)} \]

- Comparing binned counts between observed data and prediction
  - \( n_i \) is data counts in bin \( i \)
  - \( s_i \) is predicted signal (diamond) counts in bin \( i \)
  - \( b_i \) is predicted background (rocks) counts in bin \( i \)
  - \( m \) is parameter of interest, the “signal strength”

[For more details, see Glen Cowan’s talk]
Nuisance Parameters and the Likelihood

\[ L(\mu, z) = \prod_{i=1}^{N} \frac{(\mu s_i + \nu b_i)^{n_i}}{n_i!} e^{-(\mu s_i + b_i)} N(\nu; 1, \sigma) \]

\[ L_{profile}(\mu) = \max_{\nu} L(\mu, \nu) \]

• With nuisance parameters
  – \( \nu \) is the nuisance parameter, where uncertainty parameterized by \( \nu \) follows some prior (here a normal distribution)
  – Frequently use profile likelihood, maximizing over nuisance for each value of \( \mu \)

[For more details, see Glen Cowan’s talk]
Effect of Systematics

- Systematics can have a big effect, want to mitigate them as much as possible

[For more details, see Glen Cowan’s talk]
How do we get these predictions?

- Standard Model tells us how to compute interaction rates and their distributions
  
  - But we can only compute and approximation

- Several different “generators” that simulate how to compute the primary interactions and evolve them into multi-body states

- The dimensionality of the problem becomes enormous, can’t compute the full PDF

- Simulating these process is done through Monte Carlo sampling
Different generators can sometimes give rather different predictions.
Uncertainty in the Theory Model

- **Model true distributions has approximations in it**
  
  - $x_t = g(z_t; y_t=c, \nu_t)$
  - $g(\ldots)$ is the generator of events, that can be conditioned on class
  - $z_t$ is some latent random numbers (random seeds)
  - $\nu_t$ are parameters

- **Don’t know the exact model, but can quantify the uncertainty of the approximation**
  
  - E.g. parameters have been constrained by other experiments
  - Theoretical physicists give bounds on parameters
  - Variations of parameters $\nu_t$ can be done to get alternative true distributions

- $p(x) \rightarrow p(x, \nu_t) = p(x | \nu_t) p(\nu_t)$
Very Big Detectors: The ATLAS Experiment

$\sim 10^8$ detector channels

Data:
$\sim 300$ MB / sec
$\sim 3000$ TB / year

Weight:
7000 tons

Size:
46 m long,
25 m high,
25 m wide
From Theory to Experiment

- Particles interact with custom detectors
  - At the LHC, detectors can have $10^8$ sensors
• Particles interact with custom detectors
  – At the LHC, detectors can have $10^8$ sensors

• Develop algorithms to identify particles from detector signals
From Theory to Experiment

- Particles interact with custom detectors
  - At the LHC, detectors can have $10^8$ sensors

- Develop algorithms to identify particles from detector signals

- Develop small set ($\sim 20-40$) engineered features that describe the interesting properties of the event...

- Train classifier to separate signal and background
Getting it right… we need

• An accurate model of the underlying physics, including the properties of the proton, primary interaction, and the “showering” process

• An accurate and detailed model of the detector

• Accurate models of particle interactions with materials

• Must tune parameters in both the underlying theory model and the particle interactions with matter

• And we need to correct any mismodeling as best we can → Calibrations / Residual corrections
Simulating the experiment

- Experiment simulation includes geometry, (mis)alignment, material properties, etc...

- Detector / particle-material interactions simulated with GEANT
Simulating the experiment

- Experiment simulation includes geometry, (mis)alignment, material properties, etc...

- Detector / particle-material interactions simulated with GEANT

Uncertainty in our simulated electron energy measurements due to our uncertainty in the amount of material in the detector
Calibration / Modeling Corrections

• If we want to calibrate the modeling of a particle, find a pure sample of that particle and derive corrections to the simulation to match data.

• Example
  – Z bosons decays to 2 electrons that have a combined mass of $\approx 90 \text{ GeV}$
  
  – $p(\text{particle}=\text{electron} \mid m_{ee} \approx 90) \approx 99\%$
  
  – So we can select (nearly) pure electron sample by looking for electron pairs consistent with Z boson
  
  – Use known properties of Z boson to calibrate
Calibration / Modeling Corrections

- For samples **x from the simulated, a posteriori** adjust the value
  \[ p_i \rightarrow s \cdot p_i \cdot (1+z) \]
  - \( s = \text{constant} \) scaling parameter to match the mean of simulation and data
  - \( z \sim N(z; 0, \sigma_z) \) resolution smearing, match variance in simulation and data

- Measure \( s \pm \Delta s \) and \( \sigma_z \pm \Delta z \) with finite precision \( \rightarrow \) family of distributions \( p(p_i, s, \sigma_z) \)
  \[ s \sim N(s; s_0, \Delta_s) \quad z \sim N(z; 0, \sigma_z) \quad \sigma_z \sim N(\sigma_z; \sigma_{z0}, \Delta_z) \]

**Diagram:**
- True distribution; no detector
- Simulated distribution with detector
- Corrected simulation, with s and \( \sigma_z \) variations
- Observed Data

\[ m_{ee}^2 = (E_1 + E_2)^2 - (\bar{p}_1 + \bar{p}_2)^2 \]
Calibration / Modeling Corrections

• Correct simulated mean and variance to match data

Scale and smear momentum $p_i \rightarrow s*p_i*(1+z)$ to get matching mass distributions

$$m_{ee}^2 = (E_1 + E_2)^2 - (\vec{p}_1 + \vec{p}_2)^2$$
Moving Forward

- Our known unknowns, our **Systematic Uncertainties**, will diminish our sensitivity to new physics

- We want to reduce them where we can
  - Improve our models with new inputs from new measurements

- And diminish their impact as much as possible
  - Can we do this when training a classifier to separate signal from background?
• Train classifier, evaluate on nominal sample
• Run systematically varied samples through classifier and see how it changes score distribution
  – Only evaluate systematics after training complete
• No notion of systematics in training
Typical Binary Classification Problem

Training

• Data: \( \{x_i, y_i\}_{i=1}^{N} \)
  - \( x_i \) in \( \mathbb{R}^d \)
  - \( y_i \) in \( \{0, 1\} \)

- \( x_i, y_i \sim p(x, y) \)

• Learn classifier \( h(x) \)
  - Model \( p(y|x) \) and label \( \{x_i\} \)

Testing / Evaluation

• Data: \( \{x_i\} \)

- \( x_i \sim p(x) \)
Domain Adaptation

Source domain

Target domain

Probability Distribution Functions over the domains

[N. Courty, https://mathematical-coffees.github.io/mc01-ot/]
Training on source data only won’t generalize well!


Training

• Data:
  \[ S = \{ x^{s_i}, y^{s_i} \}_{i=1}^{N} \]
  \[ U_t = \{ x^{t_i} \} \]

  \(- x^{s_i}, y^{s_i} \sim p_s(x, y) \)
  \(- x^{t_i} \sim p_t(x) \)

• Distribution of source and target domains not the same,
  \[ p_s(x, y) \neq p_t(x, y) \]

• Setting
  \(- \) Perform the same classification in both domains
  \(- \) Distributions of target and source are closely related

Testing / Evaluation

• Data: \[ \{ x^{t_i} \} \]

  \(- x^{t_i} \sim p_t(x) \)
Domain Adaptation

- How can we learn a classifier from Source data and have a classifier that works well on Target data?

- Depends on what we know about the problem!
  - We assume the tasks are related!
  - Do we know anything about what has changed between domains?
  - We can use the Target data to help in the training process!

Theoretical bounds [Ben-David et al., 2010]
Error of a classifier in the target domain is upper bounded by the sum of three terms:

- Error of classifier in source domain
- Divergence (or “distance”) between the feature distributions $p_s(x)$ and $p_t(x)$
- How closely related the classification tasks are
Domain Adaptation

• Several methods to address domain adaptation problems
  – Differ depending on the details of the problem

Approaches

• Instance weighting
  – Often in Covariate Shift setting

• Self-Labeling approaches
  – Iteratively predict target labels from source and learn
  – Not going to discuss today

• Feature representation approaches
  – Learn a feature representation $f(x)$ that has the same distribution for target and source data
  – Adversarial domain adaptation takes this approach!
Covariate Shift

• General situation:
  \[ p_s(x, y) \neq p_t(x, y) \]

• In this setting assume:
  – Class probabilities conditional on features the same:
    \[ p_t(y \mid x) = p_s(y \mid x) = p(y \mid x) \]
  – Marginal Distributions different:
    \[ p_s(x) \neq p_t(x) \]
Covariate Shift

\[ p_{\text{test}}(x) \]

\[ p_{\text{train}}(x) \]

Input density

Function

Training samples

Test samples

Target function

[M. Sugiyama, MLSS 2012]
• Just estimate $p(y \mid x)$ directly from source?

  — *Model misspecification:* if classifier model $f(x; \theta)$ can not module true decision boundary for any $\theta$, classifier performance will differ between source and target
Covariate Shift

- Training: minimize average loss \( l(x_i, y_i, \theta) \) when learning with source data

\[
\frac{1}{N_s} \sum_{i=1}^{N_s} l(x_i^s, y_i^s, \theta) \xrightarrow{n \to \infty} \int l(x, y, \theta)p_s(x, y)dx\,dy
\]
Covariate Shift

- Training: minimize average loss $l(x_i, y_i, \theta)$ when learning with source data

$$\frac{1}{N_s} \sum_{i=1}^{N_s} l(x_i^s, y_i^s, \theta) \xrightarrow{n \to \infty} \int l(x, y, \theta) p_s(x, y) dx dy$$

- But want to minimize average loss $l(x_i, y_i, \theta)$ on target data

$$\int l(x, y, \theta) p_t(x, y) dx dy$$
Covariate Shift

- Training: minimize average loss $l(x_i, y_i, \theta)$ when learning with source data

$$
\frac{1}{N_s} \sum_{i=1}^{N_s} l(x_s^i, y_s^i, \theta) \xrightarrow{n \to \infty} \int l(x, y, \theta) p_s(x, y) dx dy
$$

- But want to minimize average loss $l(x_i, y_i, \theta)$ on target data

$$
\int l(x, y, \theta) p_t(x, y) dx dy
$$

- Can use density ratio

$$
\frac{p_t(x, y)}{p_s(x, y)} = \frac{p(y|x)p_t(x)}{p(y|x)p_s(x)} = \frac{p_t(x)}{p_s(x)}
$$
Covariate Shift

- Training: minimize average loss \( l(x_i, y_i, \theta) \) when learning with source data

\[
\frac{1}{N_s} \sum_{i=1}^{N_s} l(x_i^s, y_i^s, \theta) \xrightarrow{n \to \infty} \int l(x, y, \theta) p_s(x, y) dx dy
\]

- But want to minimize average loss \( l(x_i, y_i, \theta) \) on target data

\[
\int l(x, y, \theta) p_t(x, y) dx dy
\]

- Can use density ratio

\[
\frac{p_t(x, y)}{p_s(x, y)} = \frac{p(y|x)p_t(x)}{p(y|x)p_s(x)} = \frac{p_t(x)}{p_s(x)}
\]

Then:

\[
\frac{1}{N_s} \sum_{i=1}^{N_s} \frac{p_t(x)}{p_s(x)} l(x_i^s, y_i^s, \theta) \to \int \frac{p_t(x)}{p_s(x)} l(x, y, \theta) p_s(x, y) dx dy
\]

\[
= \int l(x, y, \theta) p_t(x, y) dx dy
\]
Covariate Shift

• How to estimate \( \frac{p_t(x)}{p_s(x)} \)

• Several methods, e.g. train classifier \( h(x) \) to tell the difference between target and source data, then optimal classifier gives:

\[
  h^*(x) = \frac{p_t(x)}{p_s(x) + p_t(x)} = \frac{1}{1 + \frac{p_s(x)}{p_t(x)}},
\]
Covariate Shift

• How to estimate \( \frac{p_t(x)}{p_s(x)} \)

• Several methods, e.g. train classifier \( h(x) \) to tell the difference between target and source data, then optimal classifier gives:

\[
h^*(x) = \frac{p_t(x)}{p_s(x) + p_t(x)} = \frac{1}{1 + \frac{p_s(x)}{p_t(x)}}
\]

• Difficulties with this approach
  – Don’t know ratio exactly, only can estimate from samples
  – Weighting causes variance of training data to increase

\[
\min_{\theta} \frac{1}{N_s} \sum_{i=1}^{N_s} \left( \frac{p_t(x)}{p_s(x)} \right)^\lambda l(x_i^s, y_i^s, \theta)
\]

\( \lambda = 0 \) \hspace{1cm} \( \lambda = 0.5 \) \hspace{1cm} \( \lambda = 1 \)
Feature Representation

- Setting: $p_s(x,y) \neq p_t(x,y)$
- Don’t know if $p_s(y|x) \neq p_t(y|x)$
Feature Representation

- Setting: $p_s(x,y) \neq p_t(x,y)$
- Don’t know if $p_s(y \mid x) \neq p_t(y \mid x)$
- Do our best, at least try to make feature distributions match
• **Setting:** $p_s(x,y) \neq p_t(x,y)$

• Don’t know if $p_s(y|x) \neq p_t(y|x)$

• Find representation of features through mapping $g(x) = z$
Feature Representation

- Want similar distributions of source and target features after mapping:

\[ p_s(z) = p_t(z) \]
• Build a classifier $h(z)$ that classifies the source well under the new representation
  – If the target task is not far from source task, then this classifier should do well on target task

• Many ways to do this, focus on Adversarial Networks
• Generative Adversarial Network (GAN)
  – Generator (G) is Neural Network that produces images from random numbers (z)
  – Discriminator (D) tries to classify images as real or fake data
  – When training
    • Generator tries to fool discriminator, and is penalized when discriminator succeeds

\[
\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_{z}(z)} [\log (1 - D(G(z)))]
\]

arXiv:1406.2661
Adversarial Learning

- Domain Adversarial Neural Network (DANN)
  - Classifier for signal vs background task
  - Discriminator uses hidden representation to tell the difference between Source and Target data
  - When training:
    - Classifier tries to fool discriminator, and is penalized when discriminator succeeds

\[
E(\theta_f, \theta_y, \theta_d) = \frac{1}{n} \sum_{i=1}^{n} L_y^i(\theta_f, \theta_y) - \lambda \left( \frac{1}{n} \sum_{i=1}^{n} L_d^i(\theta_f, \theta_d) + \frac{1}{n'} \sum_{i=n+1}^{N} L_d^i(\theta_f, \theta_d) \right),
\]

\[
(\hat{\theta}_f, \hat{\theta}_y) = \underset{\theta_f, \theta_y}{\arg\min} E(\theta_f, \theta_y, \hat{\theta}_d),
\]

\[
\hat{\theta}_d = \underset{\theta_d}{\arg\max} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d).
\]
Domain Adversarial Neural Networks

(a) Standard NN. For the “domain classification”, we use a non adversarial domain regressor on the hidden neurons learned by the Standard NN. (This is equivalent to run Algorithm 1, without Lines 22 and 31)

(b) DANN (Algorithm 1)
Problem Statement

Training
- Simulated Data:
  \[ S = \{x^s_i, y^s_i, z^s_i\}_{i=1}^N \]
  - \( x^s_i, y^s_i, z^s_i \sim p(x, y, z) \)

- Family of data generating processes \( p(X,Y,Z) \)
  - \( X \) is the input data
  - \( Y \) is the target (e.g. class target for classifier)
  - \( Z \) is the value of the nuisance parameter

- We want to learn the mapping \( f(x; \theta_f) \) to make predictions of \( y \)

Testing / Evaluation
- Data: \( \{x^t_i\} \)
  - \( x^t_i \sim p_t(x) \)
Problem Statement

Training

- Simulated Data:
  \[ S = \{ x^s_i, y^s_i, z^s_i \}_{i=1}^N \]

  \[ x^s_i, y^s_i, z^s_i \sim p(x, y, z) \]

Testing / Evaluation

- Data: \{x^t_i\}

  \[ x^t_i \sim p_t(x) \]

- Believe there is some \( z \) that would make simulated distribution match observed data

  - Often have the situation that we believe \( p_s(y) = p_t(y) \), i.e. priors (fraction of backgrounds) are not changing, only their feature distributions… but won’t use that information here

- So we want to be robust to changes in \( z \)

  - We want \( f(\ldots) \) to be a “pivotal quantity”, such that

    \[ p(f=s \mid z) = p(f=s \mid z') \]
• Classifier built to solve problem at hand
Adversarial Networks

- Adversary is built to predict the value of \( Z \) given the classifier output
  - If adversary can predict \( Z \), than there is information in \( f(\ldots) \) about nuisance parameter, i.e. \( f(\ldots) \) and \( Z \) are correlated in some way

Adversarial Networks

\[ \hat{\theta}_f, \hat{\theta}_r = \arg\min_{\theta_f} \max_{\theta_r} E(\theta_f, \theta_r). \]

\[ E_\lambda(\theta_f, \theta_r) = \mathcal{L}_f(\theta_f) - \lambda \mathcal{L}_r(\theta_f, \theta_r), \]

- Loss encodes performance of both classifier and adversary
  - Classifier is penalized when adversary does well at predicting \( Z \)
- Hyper-parameter \( \lambda \) controls the trade-off between performance of \( f(\ldots) \) and its independence w.r.t. the nuisance parameter
  - Large \( \lambda \) enforces \( f(\ldots) \) to be pivotal
  - Small \( \lambda \) allows \( f(\ldots) \) to be more optimal
Adversary Architecture

• The adversary is designed to model the posterior probability of $Z$ given $f(\cdot)$:
  $$p(z \mid f(x) = s)$$

• If $Z$ is categorical, the posterior can be modeled with a standard classifier
  – logistic / softmax output with cross-entropy loss

• If $Z$ is continuous, posterior can be modeled with a mixture density network

• No constraints on the prior $p(z)$
Mixture Density Network

• Typically for regression, \( f(\ldots) \) predicts only the value of \( Y \) and comparing with square error loss \( L = \sum (f-y)^2 \)

• Instead, have \( f(x_i) \) output two values, \( \mu(x_i) \) and \( \sigma(x_i) \)
  – Compare with prediction using Gaussian \( G(y_i \mid \mu(x_i), \sigma(x_i)) \)

• Since we don’t know the actual distribution of \( p(Z \mid f(x)=s) \), model using a sum of Gaussians as a more flexible family of distributions than a single Gaussian
Learning to Pivot: Toy Example

- **2D example**

\[ x \sim \mathcal{N}\left((0, 0), \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}\right) \quad \text{when } Y = 0, \]

\[ x \sim \mathcal{N}\left((1, 1 + Z), \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) \quad \text{when } Y = 1. \]
Learning to Pivot: Toy Example

- **2D example**
  
  \[ x \sim \mathcal{N}\left((0,0), \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}\right) \quad \text{when } Y = 0, \]
  
  \[ x \sim \mathcal{N}\left((1, 1 + Z), \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) \quad \text{when } Y = 1. \]

- **Without adversary** (top) large variations in network output with nuisance parameter

- **With adversary** (bottom) performance is independent!

Discriminate between boosted W jets (Y=1) and QCD (Y=0)

- Using data from Baldi et al, arXiv:1603.09349

Nuisance parameter is the amount of pileup

- Z=0 for no pileup
- Z=1 for $<\mu>=50$

Will tune $\lambda$ to optimize the significance of finding W jets

- Normalize samples to that we have 1000 QCD jets and 100 W jets

[Baldi et al, arXiv:1603.09349]
Learning to Pivot: Physics Example


Optimal tradeoff of performance vs. robustness

Non-Adversarial training

- $\lambda=0$, $Z=0$
  - Standard training with no systematics during training, evaluate systematics after training

- $\lambda=0$
  - Training samples include events with systematic variations, but no adversary used

- $\lambda=10$
  - Trading accuracy for robustness results in net gain in terms of statistical significance

[This is AMS, an estimate of significance, See Glen’s talk]
Decorrelating Variables

- Same adversarial setup can decorrelate a classifier from a chosen variable (rather than nuisance parameter)

- In this example, decorrelate classifier from jet mass, so as not to sculpt jet mass distribution with classifier cut

[See Daniel Whiteson’s talk]
Weak Supervision

Training

• Simulated Data:
  \[ S = \{x^s_i, y^s_i\}_{i=1}^N \]
  \[ U = \{x^t_i\} \]

  - \( x^s_i, y^s_i \sim p_s(x, y) \)
  - \( x^t_i \sim p_t(x) \)

Testing / Evaluation

• Data: \( \{x^t_i\} \)
  - \( x^t_i \sim p_t(x) \)

• In this setting, we believe we know that the simulation can accurately predict the class labels conditioned on an auxiliary variable \( w \)
  - \( p_s(y \mid w) = p_t(y \mid w) \) where \( \text{Cov}(w, x) = 0 \)

• Can use known class fractions in bins of \( w \) as targets
Weakly Supervised Training on Data

- Train directly on data, many analysis uncertainties can be avoided.

\[
f_{\text{weak}} = \arg\min_{f':\mathbb{R}^n \to [0,1]} \ell \left( \frac{1}{N} \sum_{i=1}^{N} \frac{f'(x_i)}{N} - y \right)
\]

arXiv:1702.00414
Conclusion

• Systematics are our known unknowns that cause variations in our distributions, and are parameterized by nuisance parameters.

• Domain adaptation techniques provide insights into how we might mitigate the effects of systematics.

• Developed Adversarial techniques for training a classifier to avoid being sensitive to systematics.
  – Can also be used to induce independence of a classifier w.r.t. some variable.

• **But should we even use this within the setting of hypothesis testing with a profile likelihood test statistic?**
  – If we know well how a systematic works, is it better to simply profile the systematic in a statistical analysis?
  – For systematics that we don’t understand, e.g. generator uncertainties, this may be very useful!
Backup
Particle Detection at ATLAS
Particle Interactions with Material

- Particles from the primary interaction interact with the detector
  - This is how we can measure their energy and trajectory
  - We have to simulate this process... at the atomic level

- **Bethe formula**: mean energy loss per unit of material traversed
  - Losing energy in material is a stochastic process!
  - Use Monte Carlo techniques to compute energy loss of particle as it passes through a slice of material
• Sometimes, we may not trust the theoretical model of the background

• Find another sample with only background-like events and measure the background like distribution directly
  
  – Is new distribution exactly like our real background distribution?  
  • If not, can we correct it? Can we quantify how it is different?
  
  – Typically compare several background-like samples to get an idea of how the distribution may vary
Building a Background Model From Data

- Find signal-free samples with different features
  - $\text{Cov}(v_1, v_2) = 0$

- Change one feature to build model, change other feature to test

![Graphs showing probability distributions and ratios for different v1 and v2 values.](image)
Training Algorithm: Alternating SGD

1: for $t = 1$ to $T$ do
2:    for $k = 1$ to $K$ do
3:       Sample minibatch $\{x_m, z_m, s_m = f(x_m; \theta_f)\}_{m=1}^M$ of size $M$; ▷ Update $r$
4:       With $\theta_f$ fixed, update $r$ by ascending its stochastic gradient $\nabla_{\theta_r} E(\theta_f, \theta_r) :=$
5:          $\nabla_{\theta_r} \sum_{m=1}^M \log p_{\theta_r}(z_m|s_m)$;
6:    end for
7:    Sample minibatch $\{x_m, y_m, z_m, s_m = f(x_m; \theta_f)\}_{m=1}^M$ of size $M$; ▷ Update $f$
8:    With $\theta_r$ fixed, update $f$ by descending its stochastic gradient $\nabla_{\theta_f} E(\theta_f, \theta_r) :=$
9:       $\nabla_{\theta_f} \sum_{m=1}^M \left[ -\log p_{\theta_f}(y_m|x_m) + \log p_{\theta_r}(z_m|s_m) \right]$,
10:      where $p_{\theta_f}(y_m|x_m)$ denotes $1(y_m = 0)(1 - s_m) + 1(y_m = 1)s_m$;
11: end for
Calibration / Modeling Corrections

- Given some electron vs. non-electrons classifier, check classifier true positive rate with electrons coming from Z boson