Independence of NN classifier from a continuous parameter

i.e. fair neural networks

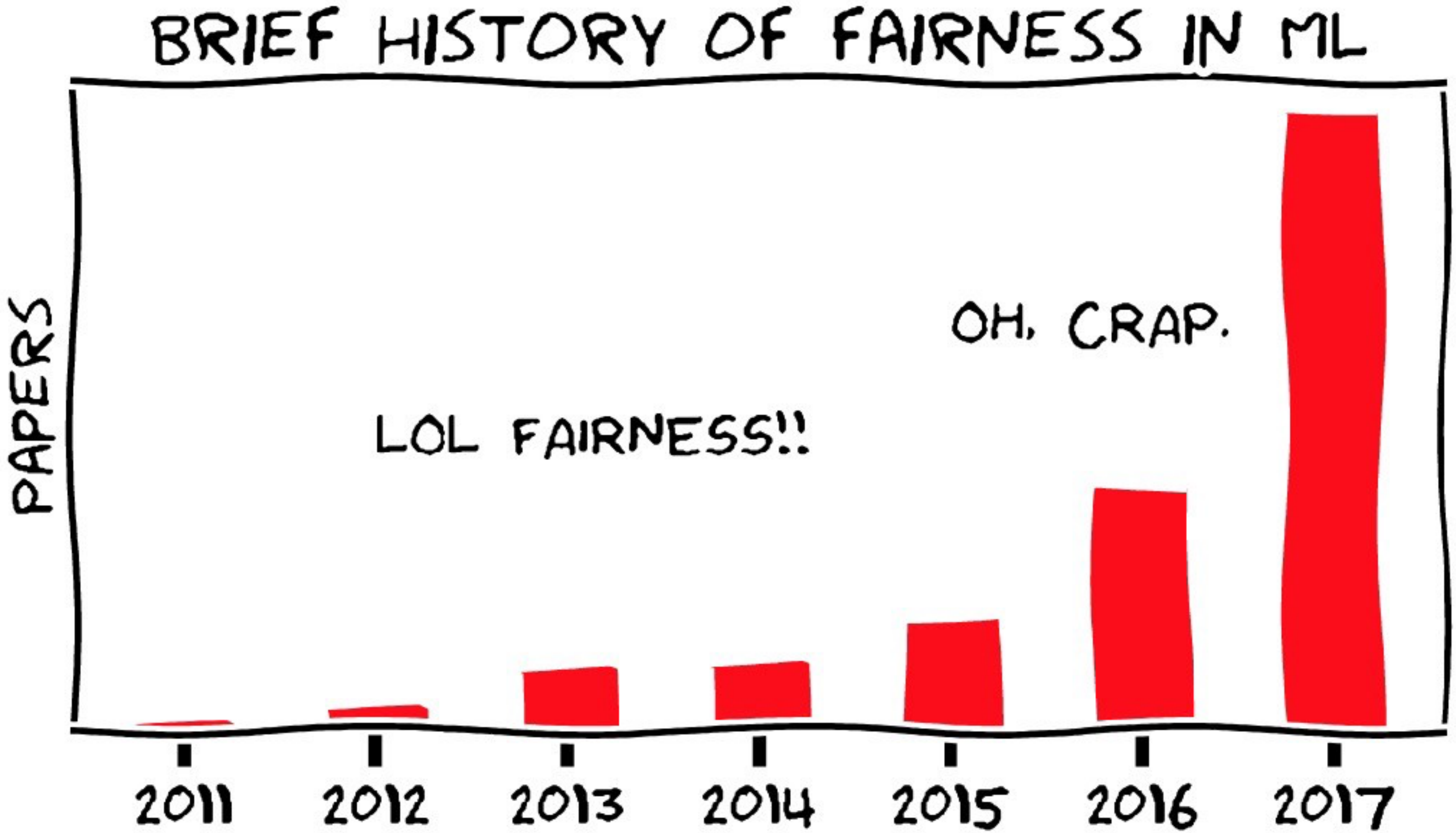
Miha Zgubič

(miha.zgubic@cern.ch)

MLHEP 2019

July 8, 2019

DESY, Hamburg


BRIEF HISTORY OF FAIRNESS IN ML
PAPERS
OH. CRAP.
LOL FAIRNESS!!
2011
2012
2013
2014
2015
2016
2017

Non-physics problem

You are a company selling insurance.

Your algorithm learns that ***for some reason*** men on average live shorter lives than women and sells them insurance less often.

This sparks public outrage and you want to make sure that the algorithm is fair w.r.t. gender.

This article is more than **8 months old**

Amazon ditched AI recruiting tool that favored men for technical jobs

Specialists had been building computer programs since 2014 to review résumés in an effort to automate the search process

BUSINESS NEWS OCTOBER 10, 2018 / 4:12 AM / 8 MONTHS AGO

Amazon scraps secret AI recruiting tool th
showed bias against women

A serious problem in many applications of machine learning:

Amazon learnt this the hard way!

IBM open source fairness toolkit

https://aif360.mybluemix.net

Fairness metrics,

data preprocessing,

re-weightings,

output postprocessing,

tutorials and examples

Physics problem

Decorrelated Jet Substructure Tagging using Adversarial Neural Networks

Chase Shimmin
Department of Physics and Astronomy, UC Irvine, Irvine, CA 92627 and
Department of Physics, Yale University, New Haven, CT

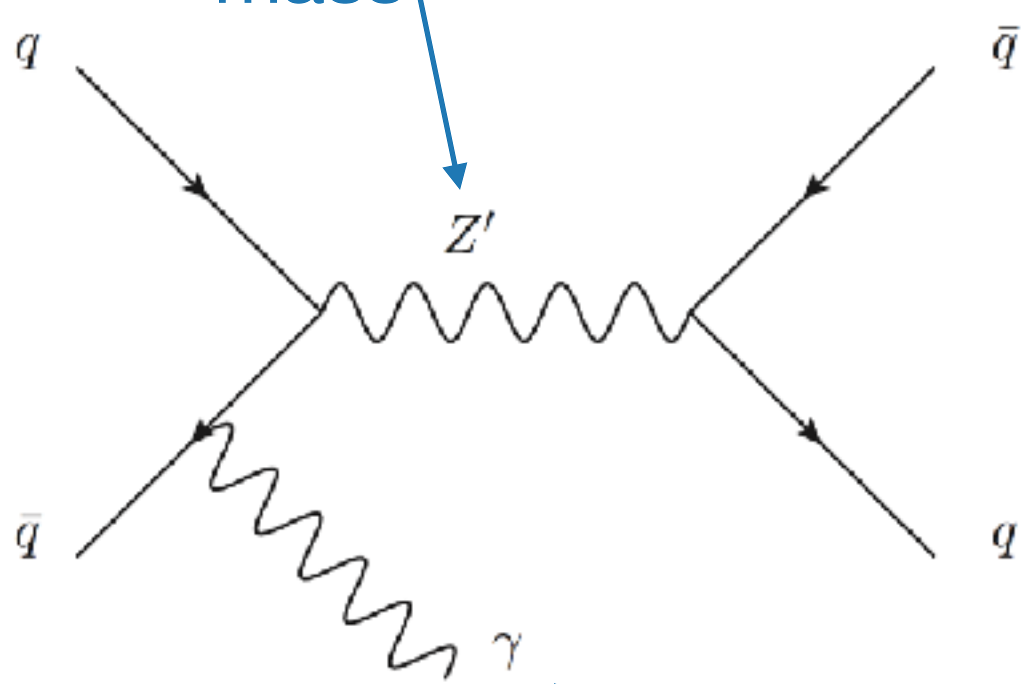
Peter Sadowski and Pierre Baldi
Department of Computer Science, UC Irvine, Irvine, CA 92627

Edison Weik and Daniel Whiteson
Department of Physics and Astronomy, UC Irvine, Irvine, CA 92627

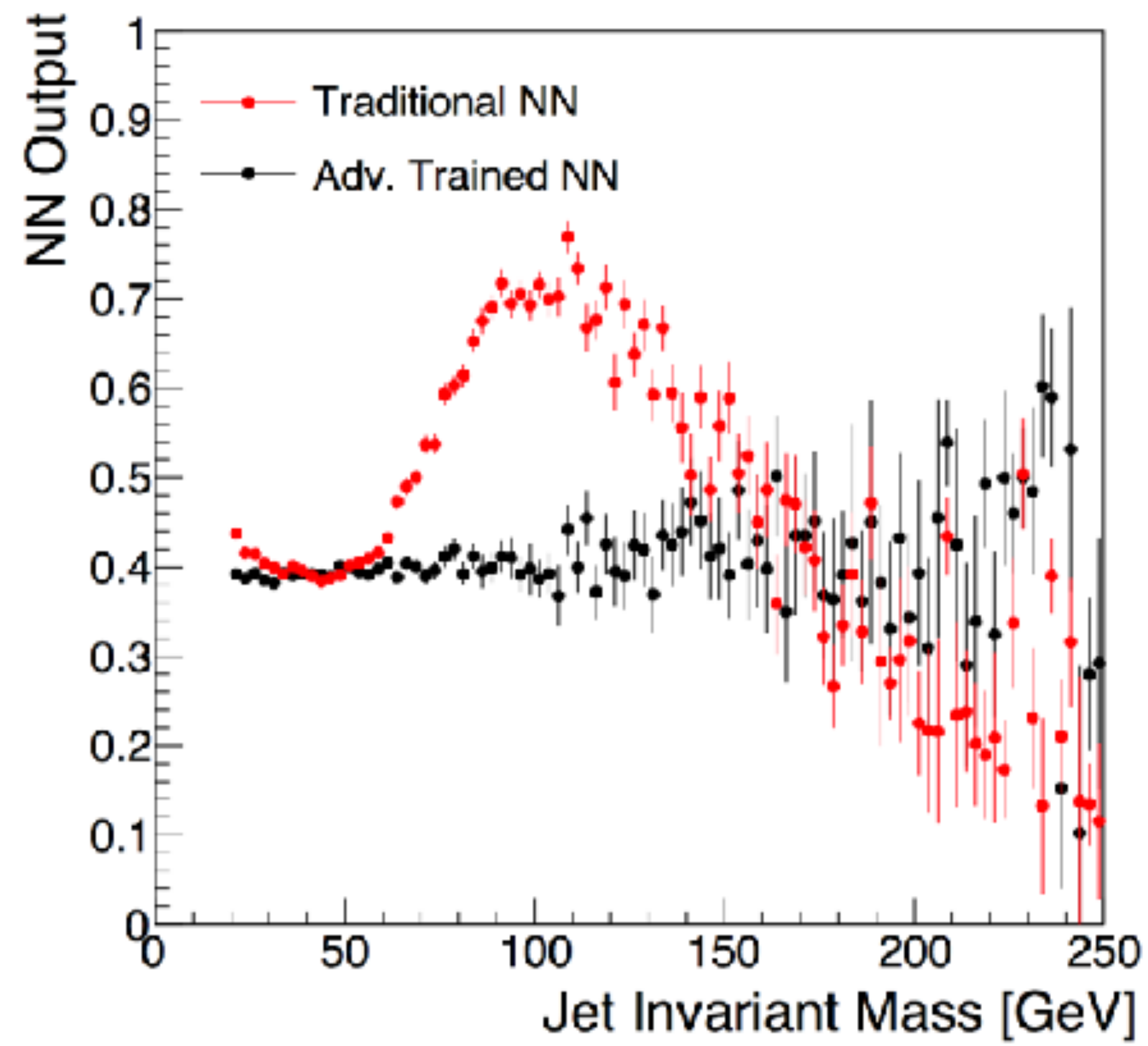
Edward Goul
Department of Physics, MIT, Cambridge, MA 02139

Andreas Søgaard
Department of Physics and Astronomy, University of Edinburgh, Edinburgh UK
(Dated: March 13, 2017)

Z' is a new particle with unknown mass



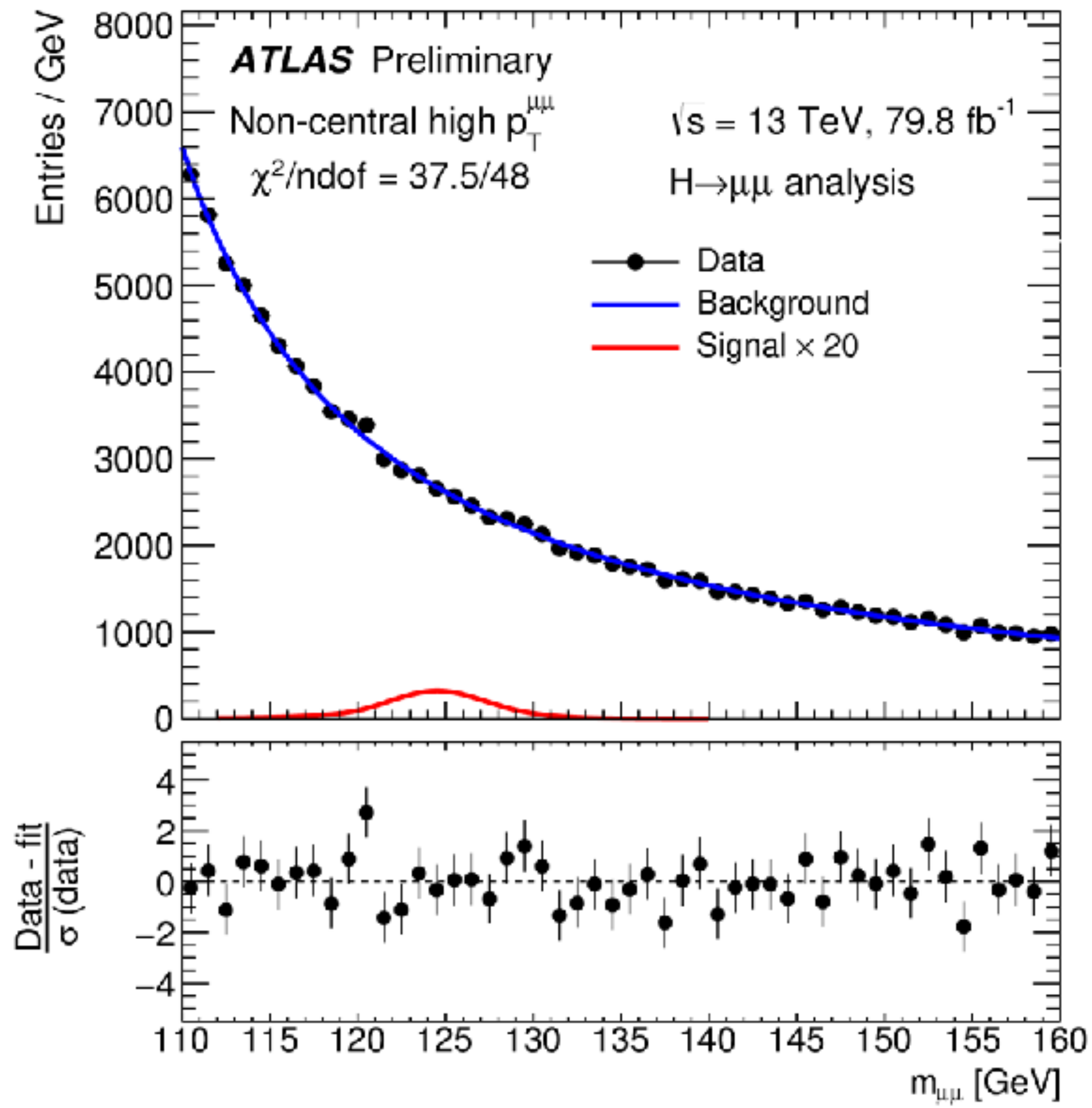
Recoils against a photon



Another physics problem

Higgs -> mu mu

Fit an analytical function to data spectrum.



Train a NN classifier to distinguish between H and Z (dominant background)

-> Without learning the invariant mass (keep mass spectrum intact)

Your application here

your text

more text

more text

your

impressive

figure

How to do it

2016

Learning to Pivot with Adversarial Networks

Gilles Louppe
New York University
g.louppe@nyu.edu

Michael Kagan
SLAC National Accelerator Laboratory
makagan@slac.stanford.edu

Kyle Cranmer
New York University
kyle.cranmer@nyu.edu

2018

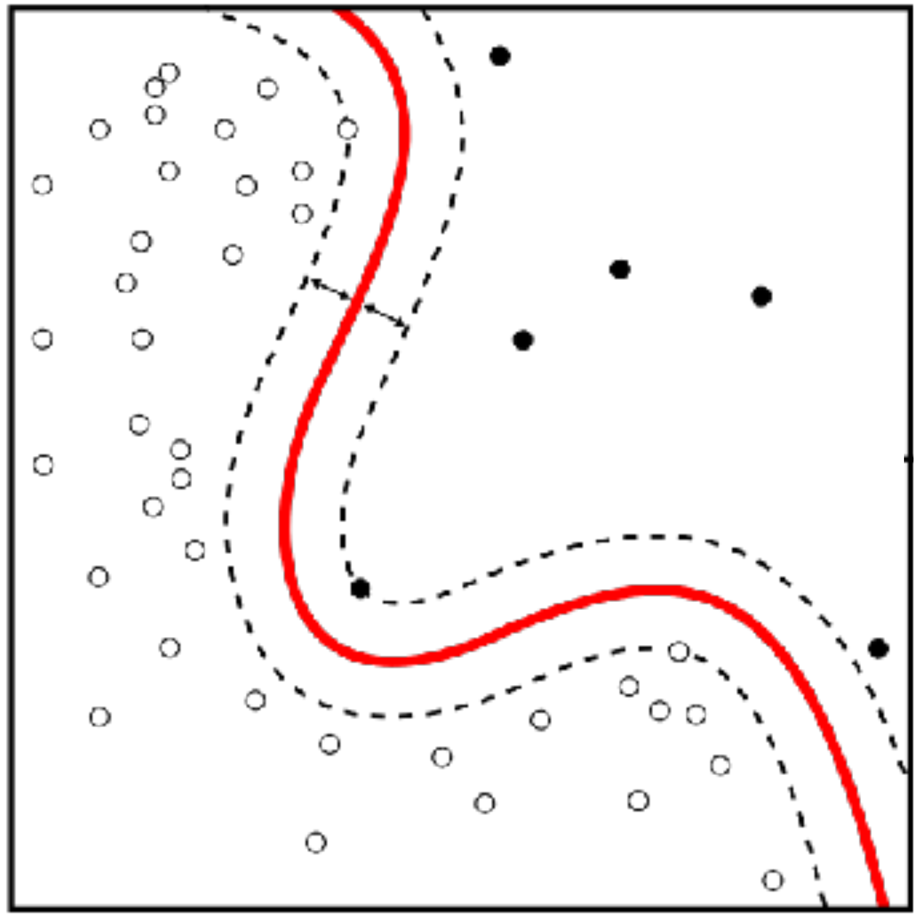
Mitigating Unwanted Biases with Adversarial Learning

Brian Hu Zhang
Stanford University
Stanford, CA
bhz@stanford.edu

Blake Lemoine
Google
Mountain View, CA
lemoine@google.com

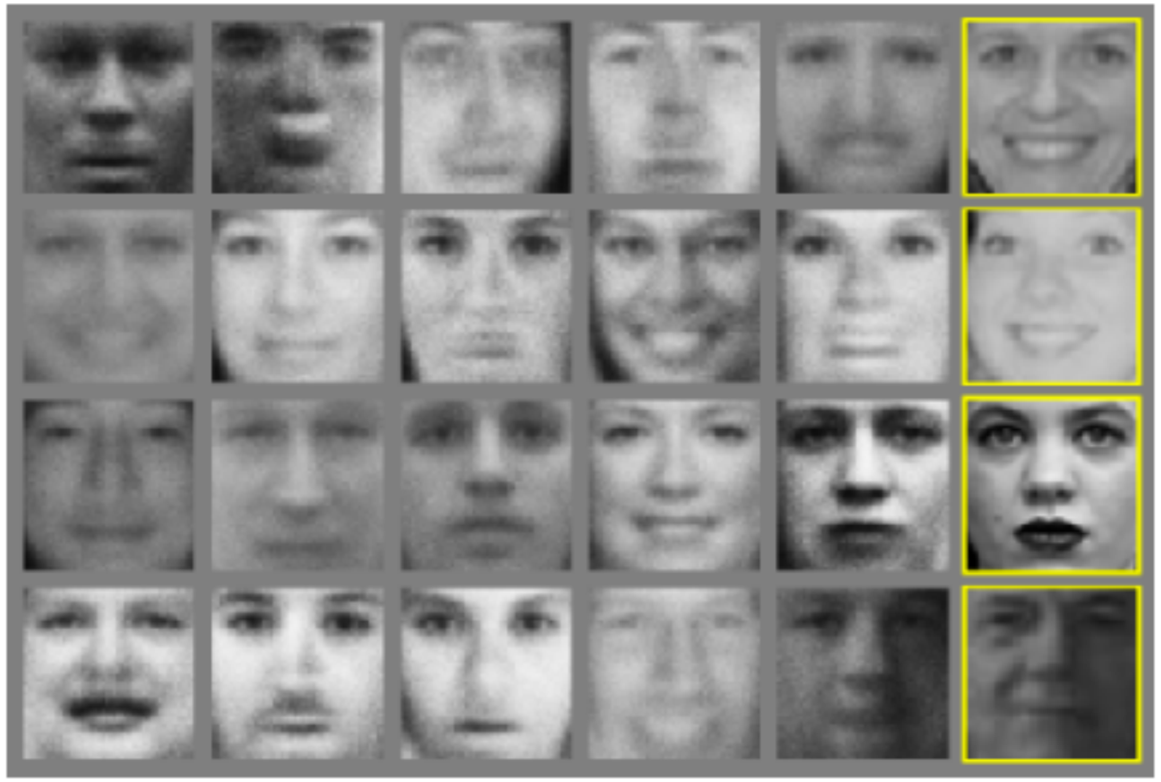
Margaret Mitchell
Google
Mountain View, CA
mmitchellai@google.com

Classifier



+

AN from GANs



= a CAN?

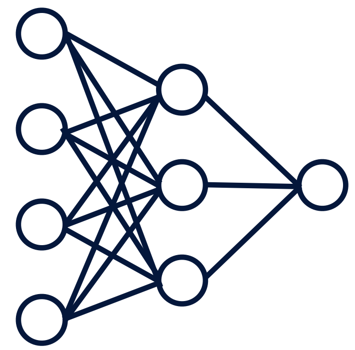
Problem setup

Classify instances with features X,
by predicting labels Y,
while being unbiased w.r.t. parameter Z

<u>in HEP, very often:</u>
features X = kinematics
labels Y = signal/background
parameter Z = mass

Classifier, parametrised by weights θ_{clf}:

<u>input</u>: kinematics X,
<u>output</u>: prediction for labels Y

X  $f_{\theta_{clf}}(X)$

$L_{classifier}$ = cross entropy between Y and $f_{\theta_{clf}}(X)$

Training (non-adversarial):

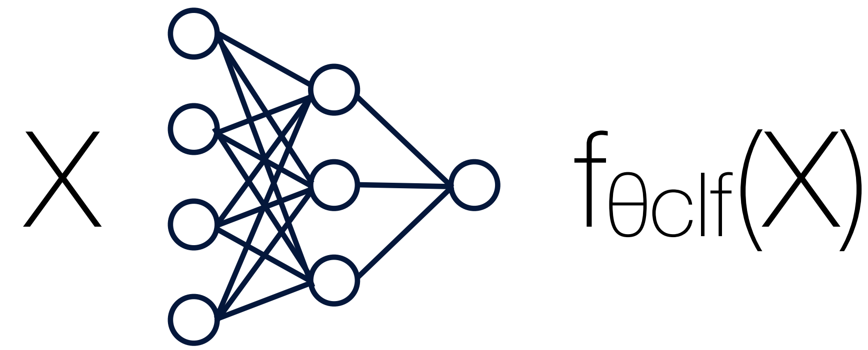
$$\theta_{clf} \leftarrow \theta_{clf} - \eta \nabla_{\theta_{clf}} L_{classifier}$$

Classifier

Classifier, parametrised by weights θ_{clf}:

input: kinematics X,

output: prediction for labels Y



$$L_{classifier} = \text{cross entropy between } Y \text{ and } f_{\theta_{clf}}(X)$$

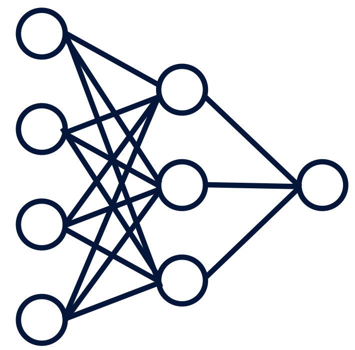
Training (non-adversarial):

$$\theta_{clf} \leftarrow \theta_{clf} - \eta \underbrace{\nabla_{\theta_{clf}} L_{classifier}}_{\substack{\text{direction which} \\ \text{improves classification}}}$$

Classifier

Classifier, parametrised by weights θ_{clf}:

<u>input</u>: kinematics X,
<u>output</u>: prediction for labels Y

X  $f_{\theta clf}(X)$

$L_{classifier}$ = cross entropy between Y and $f_{\theta clf}(X)$

Training (non-adversarial):

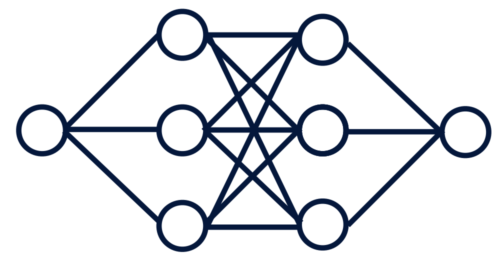
$$\theta_{clf} \leftarrow \theta_{clf} - \eta \underbrace{\nabla_{\theta clf} L_{classifier}}_{\substack{\text{direction which} \\ \text{improves classification}}} \boxed{- \underbrace{\text{indep\_term}}_{\substack{\text{direction which} \\ \text{improves independence}}}}$$

no analytic form:
parametrise by the adversary NN

Adversary

Adversary, parametrised by weights θ_{adv}:

<u>input</u>: classifier prediction $f_{\theta_{clf}}(X)$
<u>output</u>: prediction for parameter Z^*

$f_{\theta_{clf}}(X)$  Z_{pred}

$L_{adversary}$ = mean square difference between Z_{pred} and Z

Training (minimises the loss):

$$\theta_{adv} \leftarrow \theta_{adv} - \eta \nabla_{\theta_{adv}} L_{adversary}$$

*can be other things too, explained later

Modified loss

Modify classifier loss function:

$$L = L_{classifier} - \lambda \, L_{adversary}$$

cross-entropy
between Y and $f_{\theta clf}(X)$

mean square loss
between Z_{pred} and Z

Modified loss

Modify classifier loss function:

$$L = L_{classifier} - \lambda L_{adversary}$$

The classifier update rule becomes:

$$\theta_{clf} \leftarrow \theta_{clf} - \eta \nabla_{\theta_{clf}} (L_{classifier} - \lambda L_{adversary})$$

Modified loss

Modify classifier loss function:

$$L = L_{classifier} - \lambda L_{adversary}$$

The classifier update rule becomes:

$$\theta_{clf} \leftarrow \theta_{clf} - \eta \nabla_{\theta_{clf}} \big(\underbrace{L_{classifier}}_{\text{direction which improves classification}} - \lambda \underbrace{L_{adversary}}_{\text{direction which improves independence}} \big)$$

Modified loss

Modify classifier loss function:

$$L = L_{classifier} - \lambda L_{adversary}$$

The classifier update rule becomes:

minus sign: want to make classifier such that is maximises the adversary loss

tradeoff between accuracy and independence

$$\theta_{clf} \leftarrow \theta_{clf} - \eta \nabla_{\theta_{clf}} \Big(\underbrace{L_{classifier}}_{\substack{\text{direction which} \\ \text{improves classification}}} - \underbrace{\lambda L_{adversary}}_{\substack{\text{direction which} \\ \text{improves independence}}} \Big)$$

Algorithm

<u>Main training loop</u>

for N steps:

  update the classifier: $\theta_{clf} \leftarrow \theta_{clf} - \eta \nabla_{\theta clf} (L_{classifier} - \lambda \ L_{adversary})$

  for M steps:

    update the adversary: $\theta_{adv} \leftarrow \theta_{adv} - \eta \nabla_{\theta adv} L_{adversary}$

A few points

Classifier is unchanged between normal and adversarial training.

Adversary provides a gradient when training the classifier.

Adversary is not used at the inference stage, only at training.

Quick Summary

Problem:

Classify instances with features X by predicting labels Y, while keeping the predictions independent of the protected parameter Z

Solution:

Modify the loss function of the classifier during the training using an ancillary NN (which can be discarded at inference time)

In the practicals

Toy example, physics example (Hbb)

Both “normal” and adversarial training implemented.

Encouraged to play around with the classifier and the adversary.

Thanks to Philipp Windischhofer for preparing and sharing the MadGraph samples!

BONUS

Additional adversaries

Gaussian mixture model (GMM)

From the original paper. Predicts a probability density (GMM) over the parameter Z.

paper: https://arxiv.org/abs/1611.01046
code (Keras): https://github.com/glouppe/paper-learning-to-pivot

MINE

Mutual Information Neural Estimator. Minimises a functional which results in an estimate of Mutual Information between $f_{\theta_{Clf}}(X)$ and Z.

paper: https://arxiv.org/abs/1801.04062
code (Pytorch): https://github.com/MasanoriYamada/Mine_pytorch
code (TF): https://github.com/mzgubic/MINE

In my experience they perform similarly, but it probably depends on the problem so worth trying more than one!

Extending to multiple protected parameters

Always possible (any adversary):

$$L = L_{classifier} - \lambda L_{adv1} - \lambda L_{adv2}$$

But in principle this only guarantees that:

$p(f|Z1) = p(f)$

$p(f|Z2) = p(f)$

and <u>not</u> that:

$p(f|Z1, Z2) = p(f)$

MINE can extend naturally to multiple protected parameters by simply changing the number of input layers.

GMM needs to be rewritten for each number (to model N-dimensional pdf)