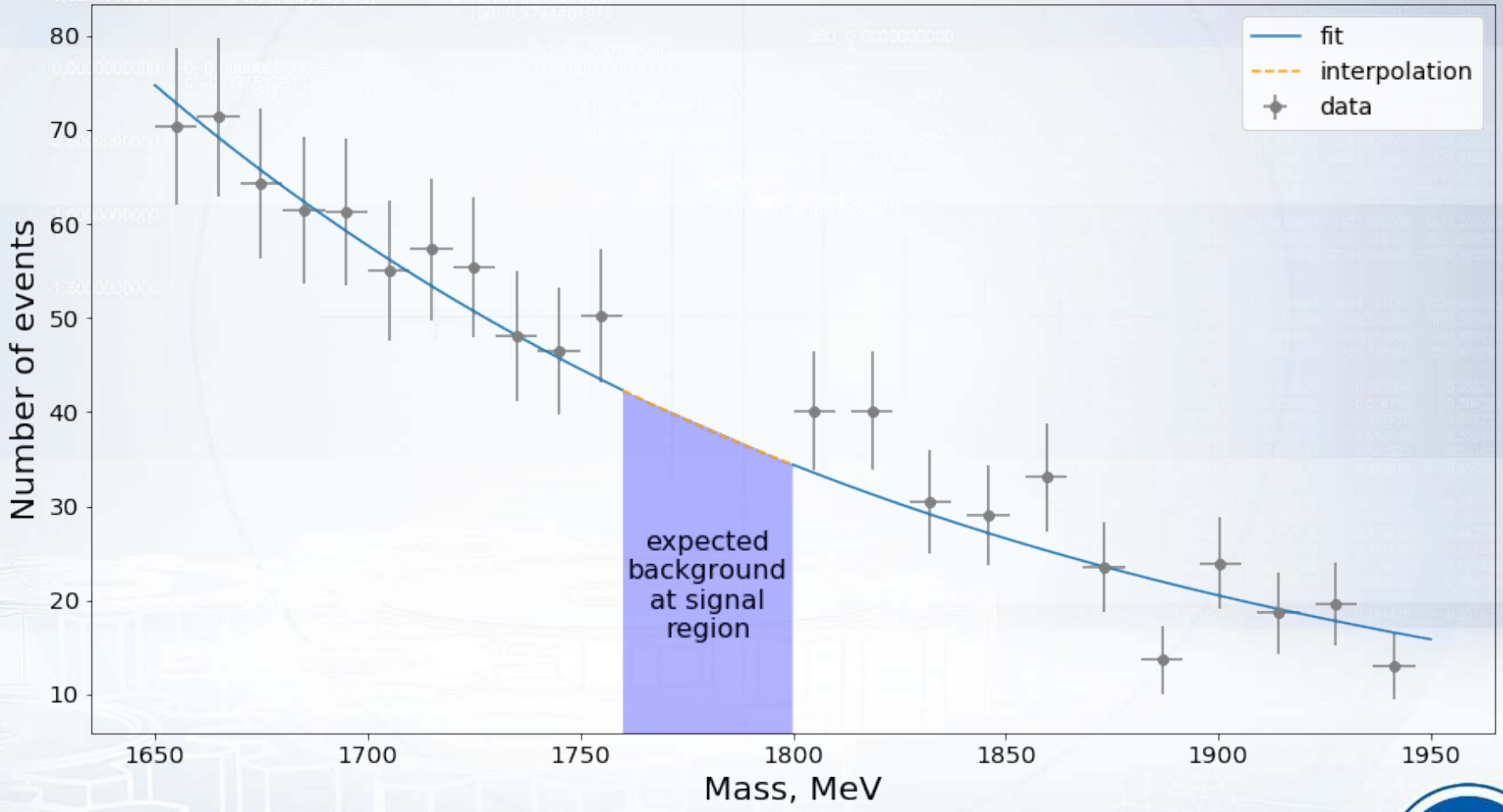
Searching for the best model

Andrey Ustyuzhanin

Searching for rare decays



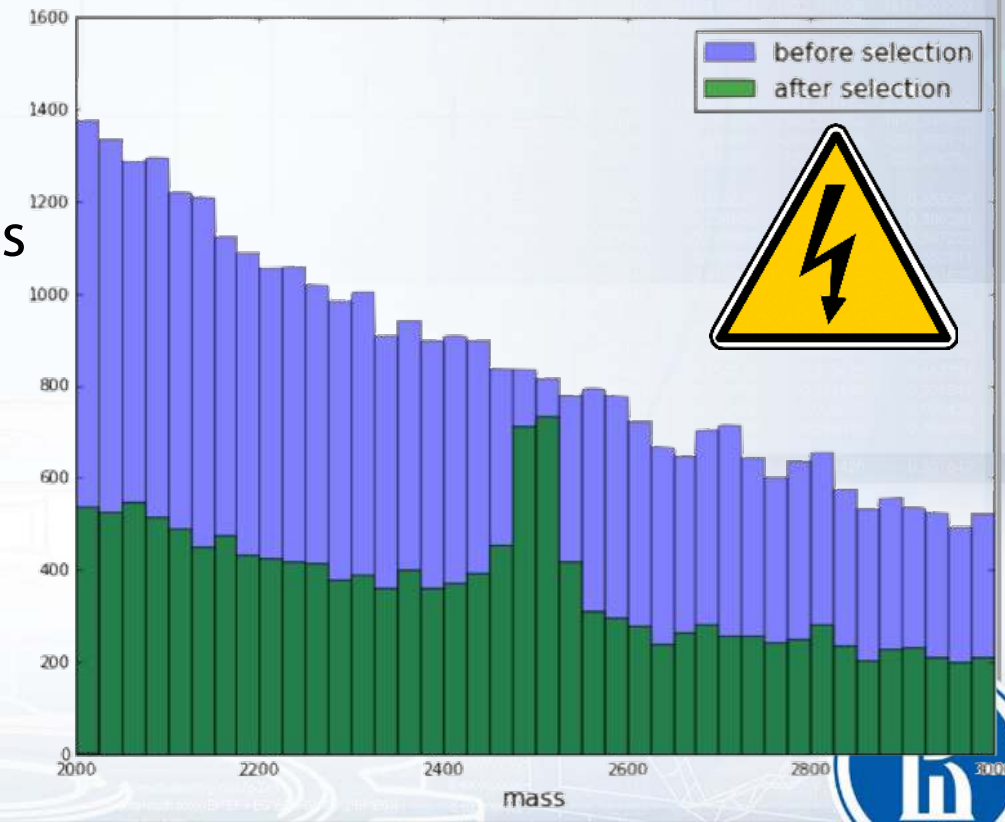
Classifier Restrictions

Uniformity

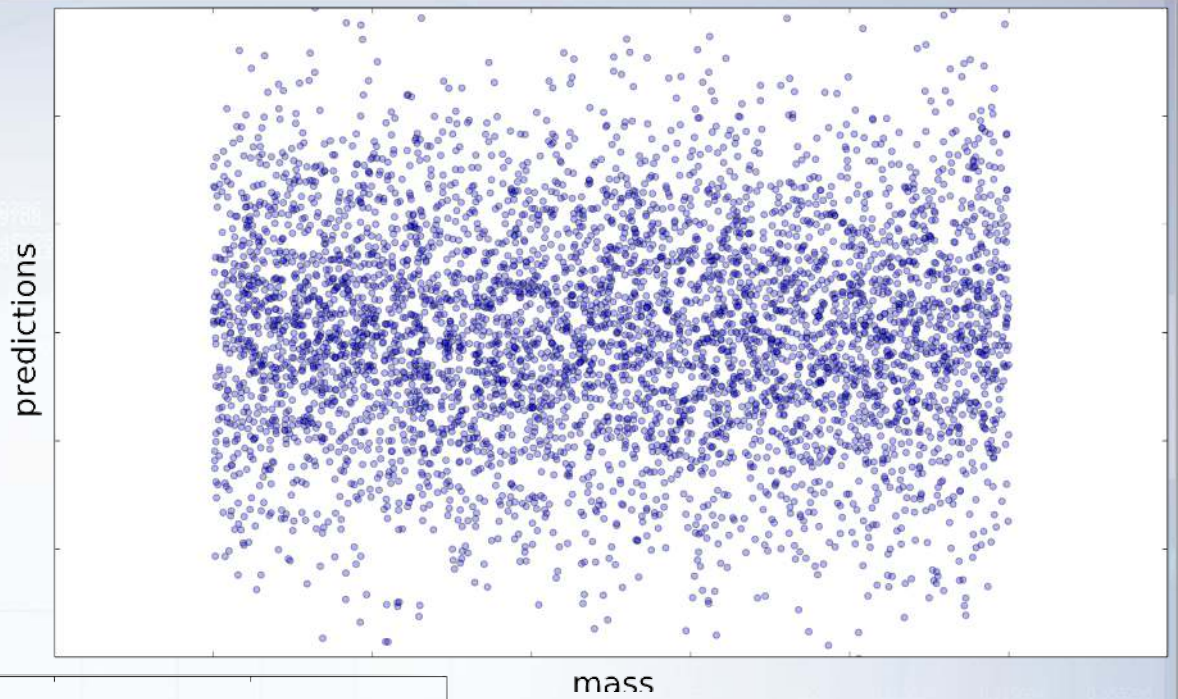
Correlation between classifier prediction and mass leads to false peaks which spoil (bias) event counting.

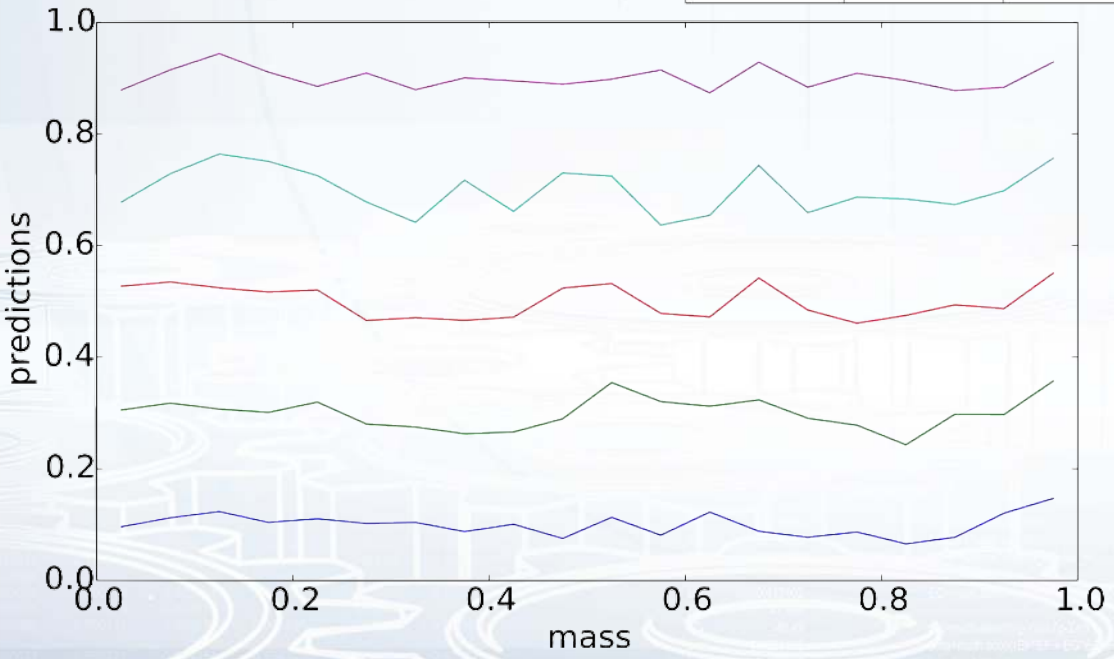
Agreement

In training data signal is represented by simulated events and background by real. Thus classifier might pick MC-specific features, which also bias counting.



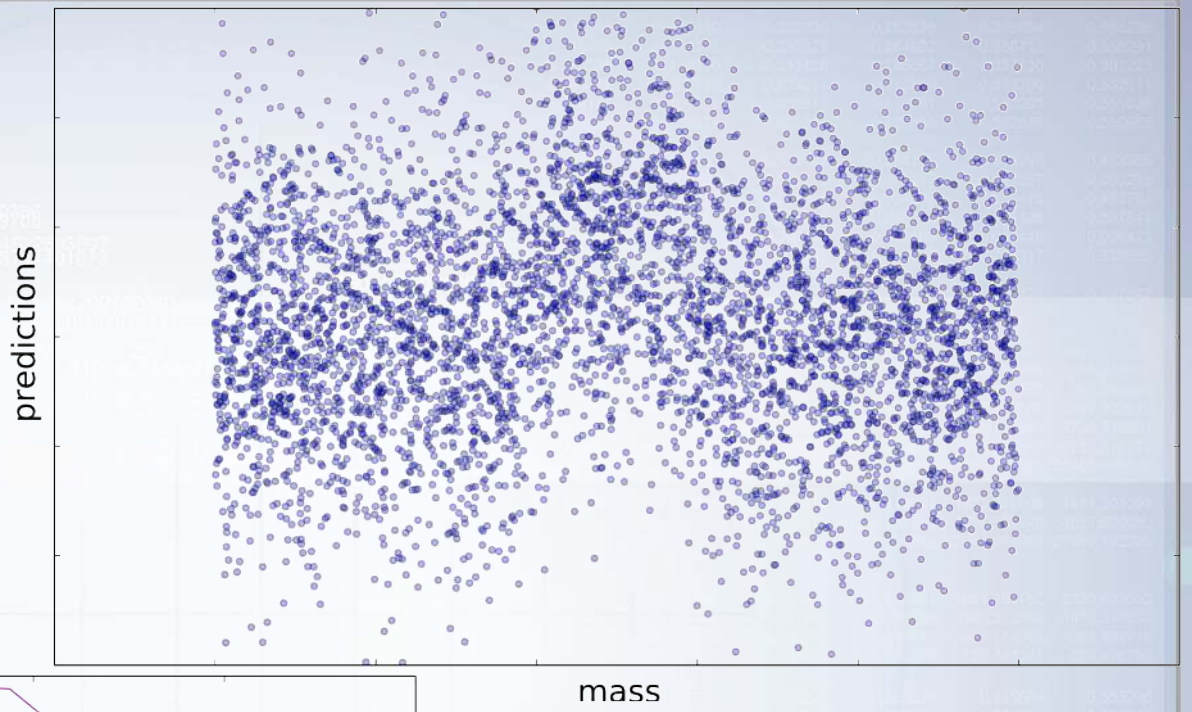
Uniformity

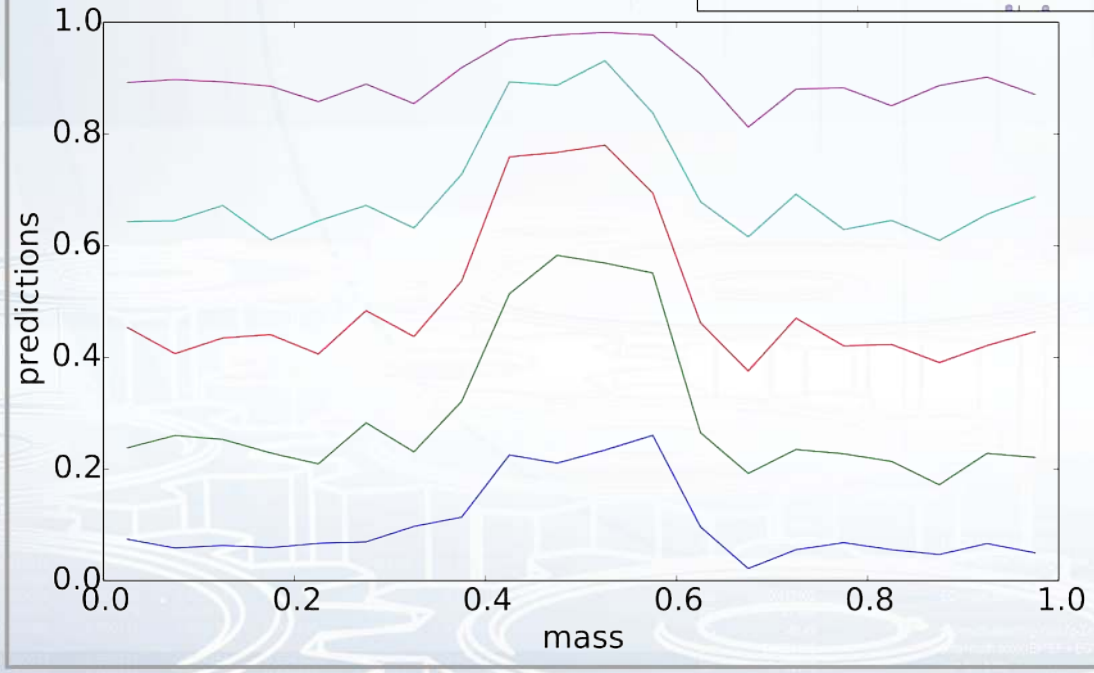




Each line shows PDF of events selected by
TPR = {0.1, 0.3, 0.5, 0.7, 0.9}

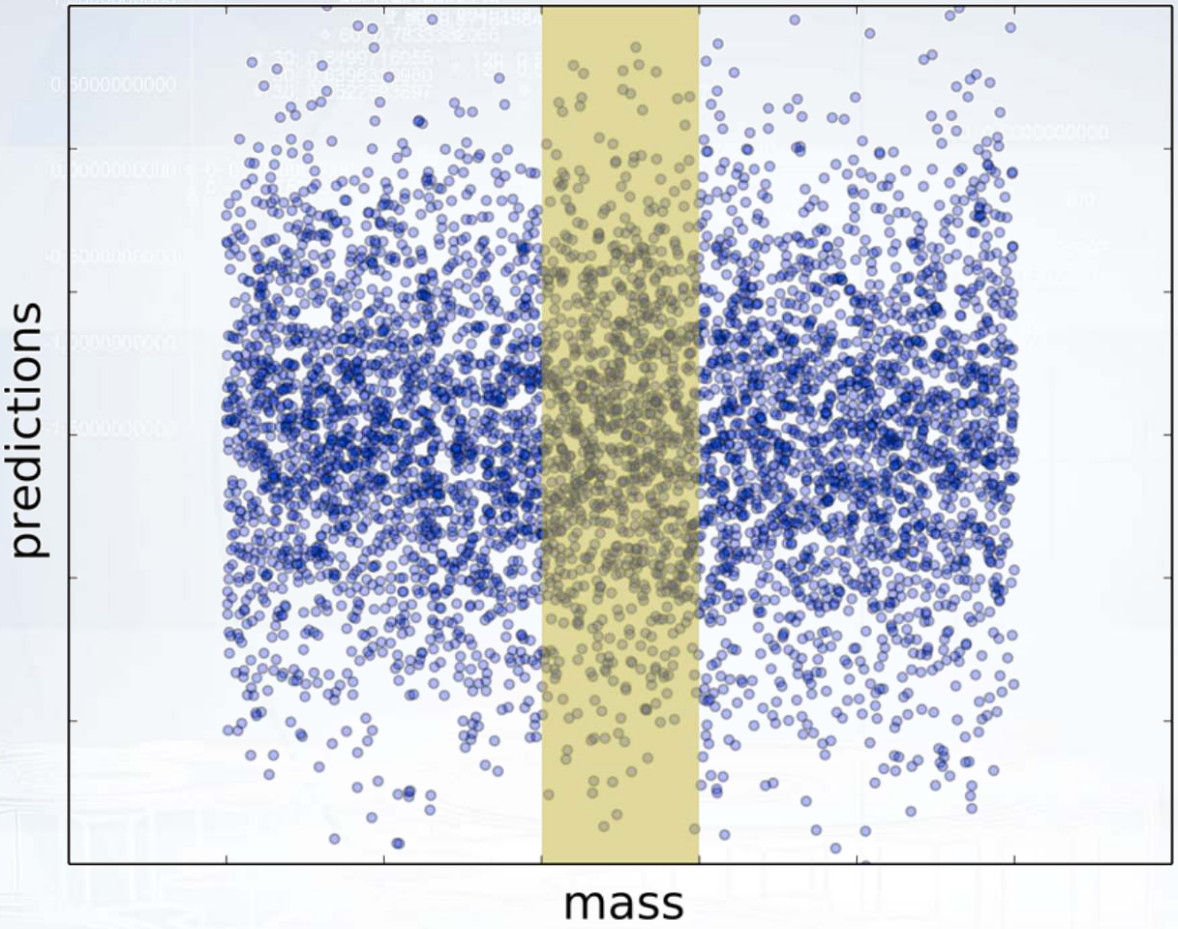
Non-Uniformity

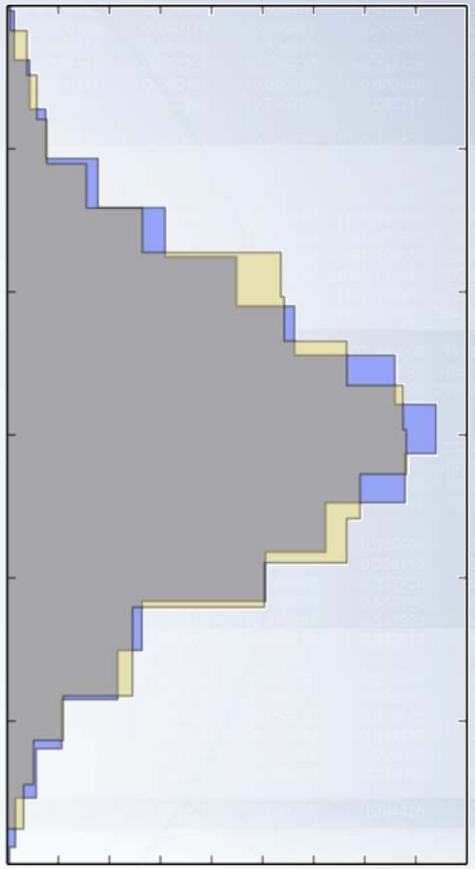




Each line shows PDF of events selected by
TPR = {0.1, 0.3, 0.5, 0.7, 0.9}

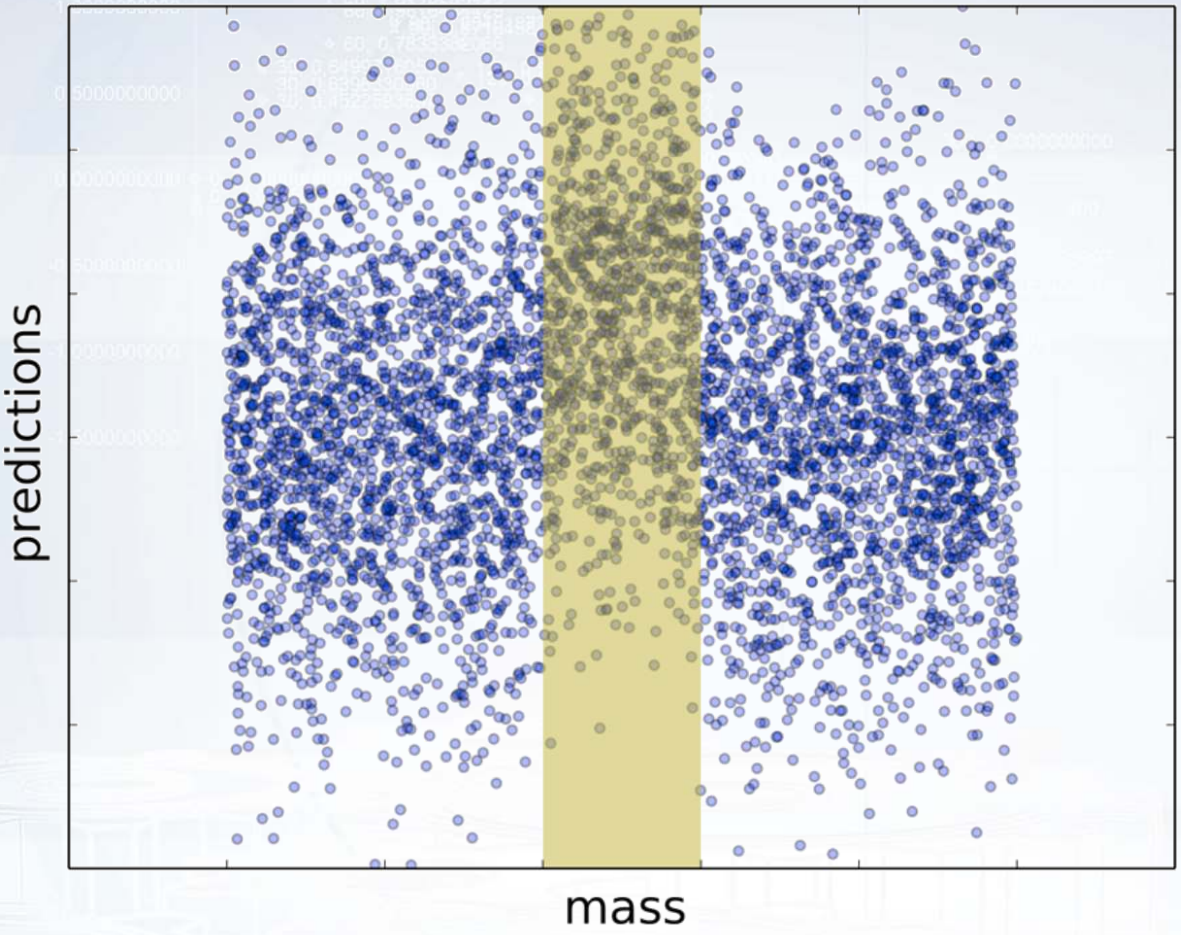
Non-Uniformity Measure

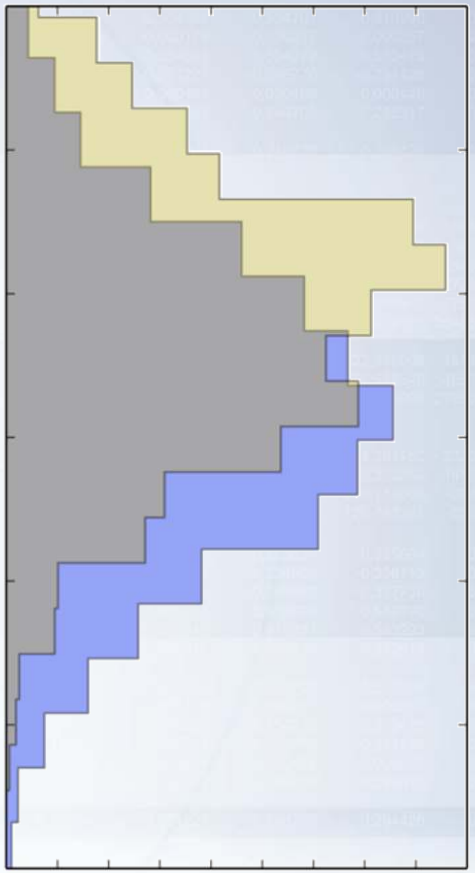




Uniform predictions

Non-Uniformity Measure

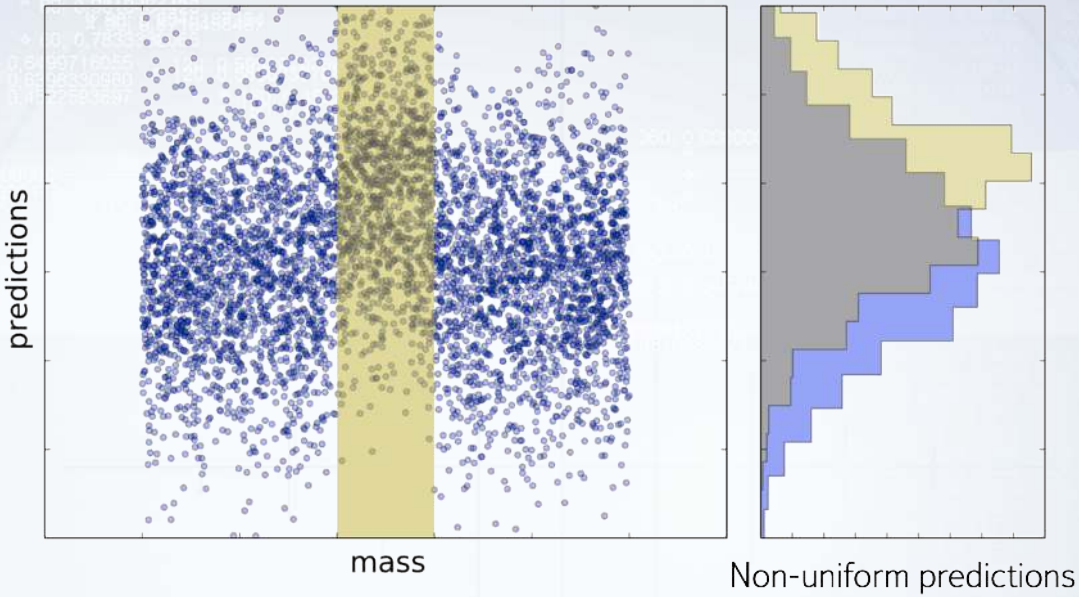




uniformity = no dependence between mass and predictions

Non-uniform predictions (peak in highlighted region)

Non-Uniformity Measure



$$\mathrm{CvM} = \sum_{\mathrm{region}} \int |F_{\mathrm{region}}(s) - F_{\mathrm{global}}(s)|^2 \, dF_{\mathrm{global}}(s)$$

Cramer-von Mises test (integral characteristic), where

F_{region} – CDF for region distribution (yellow)

F_{global} – CDF for global distribution (blue)

Uniformity Check

- random predictions and mass are independent variables;
- assume null-hypothesis: mass and predictions are independent;
- generate distribution of CvM value under null-hypothesis by repeating many times the following steps:
  - generate random predictions;
  - compute CvM value;
- choose p-value and compute corresponding CvM value.

Basic Approach

Reduce the set of features used in training: leave only those, which do not correlate with the mass

- It's simple and it works
- But omitting those features we loose information

Can we modify ML algorithm to use all the features, but provide uniform background efficiency (FPR)/signal efficiency (TPR) along the mass?

Gradient Boosting Recap

Gradient Boosting greedily builds an ensemble of estimators

$$D(x) = \sum_j \alpha_j d_j(x)$$

That minimize given loss function. Those losses could be:

- MSE: $\mathcal{L} = \sum_i (y_i - D(x_i))^2$
- AdaLoss: $\mathcal{L} = \sum_i e^{-y_i D(x_i)}, \qquad y_i = \pm 1$
- LogLoss: $\mathcal{L} = \sum_i \log(1 + e^{-y_i D(x_i)}), \qquad y_i = \pm 1$

Each term in the ensemble approximates the residuals between true y_i and all the preceding terms.

uBoostBDT

Aims to get **FPR**$_{region}$=const:

- Fix target efficiency, for example **FPR**$_{target}$=30%, and find corresponding threshold
- Train a tree, its decision function is $d(x)$
- Increase weight for misclassified: $w_i \leftarrow w_i \exp(-\alpha y_i d(x_i)))$
- Increase weight of background events in the regions with high **FPR**

$$w_i \leftarrow w_i \exp\left(\beta(\mathrm{FPR}_{\mathrm{region}} - \mathrm{FPR}_{\mathrm{target}})\right)$$

Thus we achieve **FPR**$_{region}$=30% in all regions

Computationally complex, and may get biased.

uBoostGB + FlatnessLoss

- Why not minimize CvM with Gradient Descent?
  … we can't compute the gradient!

- CvM approximation:

$$\mathscr{L}_{FL} = \sum_{region} \int |F_{region}(s) - F_{global}(s)|^2 ds$$

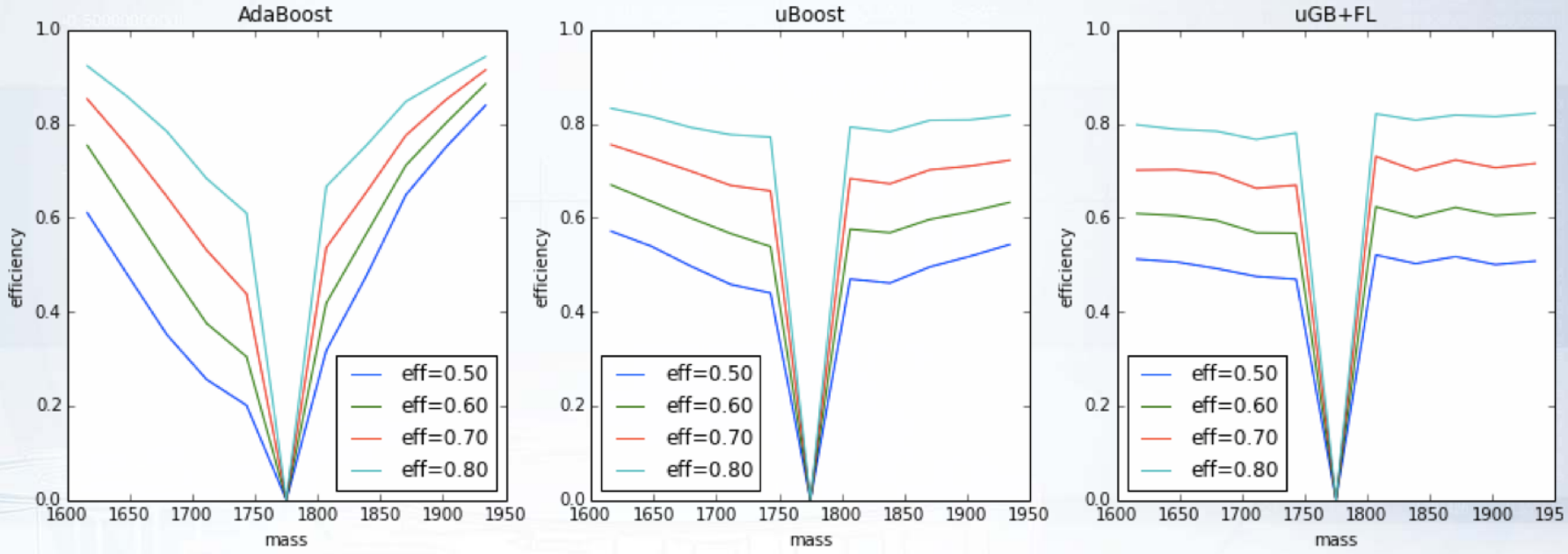
$$\frac{\partial}{\partial D(x_i)} \mathscr{L}_{FL} \sim 2(F_{region}(s) - F_{global}(s))|_{s=D(x_i)}$$

- Add approximate CvM to a loss function (regularize):

$$\mathscr{L} = \mathscr{L}_{adaloss} + \alpha \mathscr{L}_{FL}$$

arXiv:1410.4140

Improving Uniformity



- uBoostGB+FL is faster and allows for trade-off between quality / uniformity
- https://github.com/arogozhnikov/hep_ml/

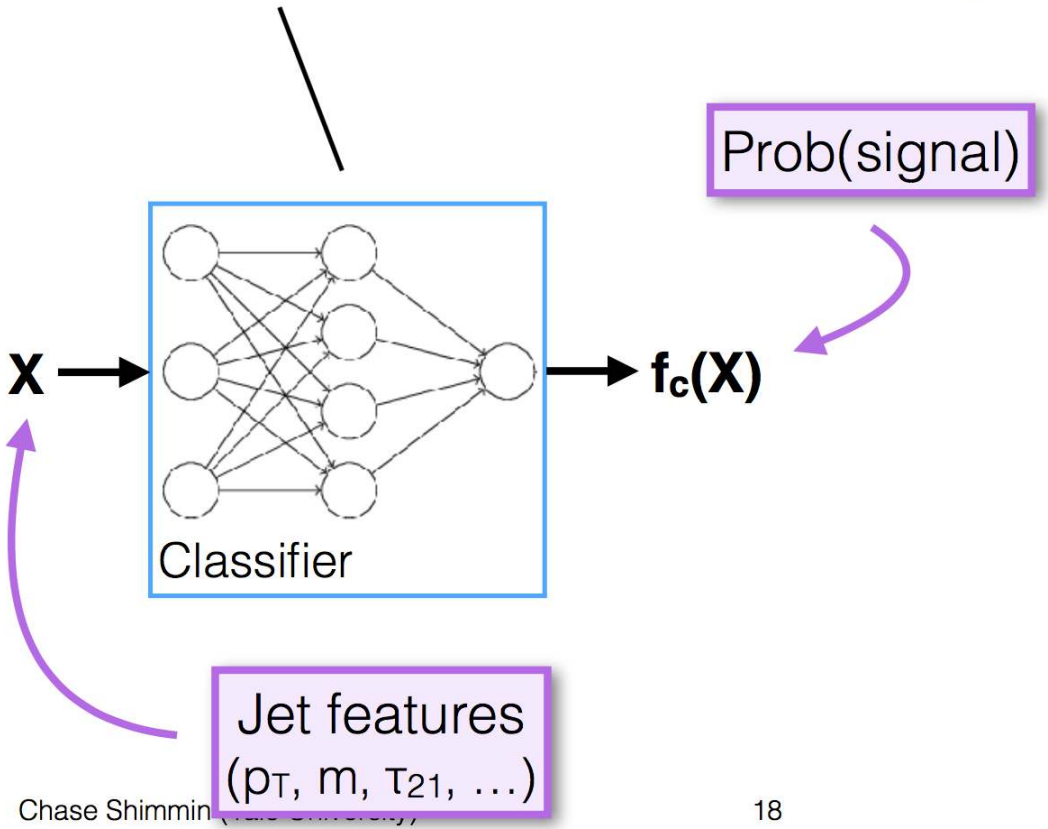
Adversarial Approach

Chase_2017.03.22_IML.pdf (стр. 18 из 37)

Adversarial Decorrelation

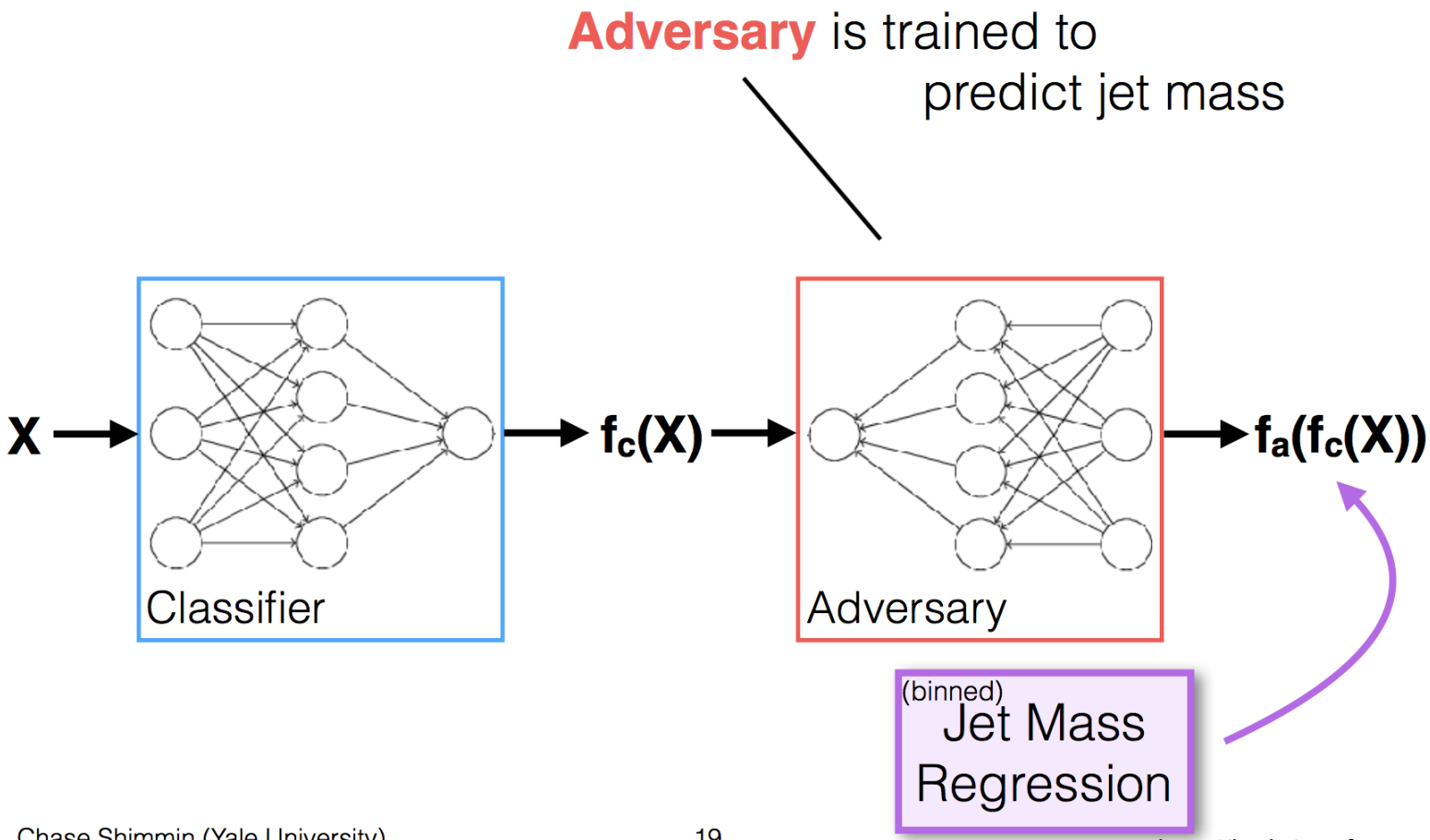
Basic idea:

Classifier is trained to identify signal jets



Adversarial Approach

Adversarial Decorrelation

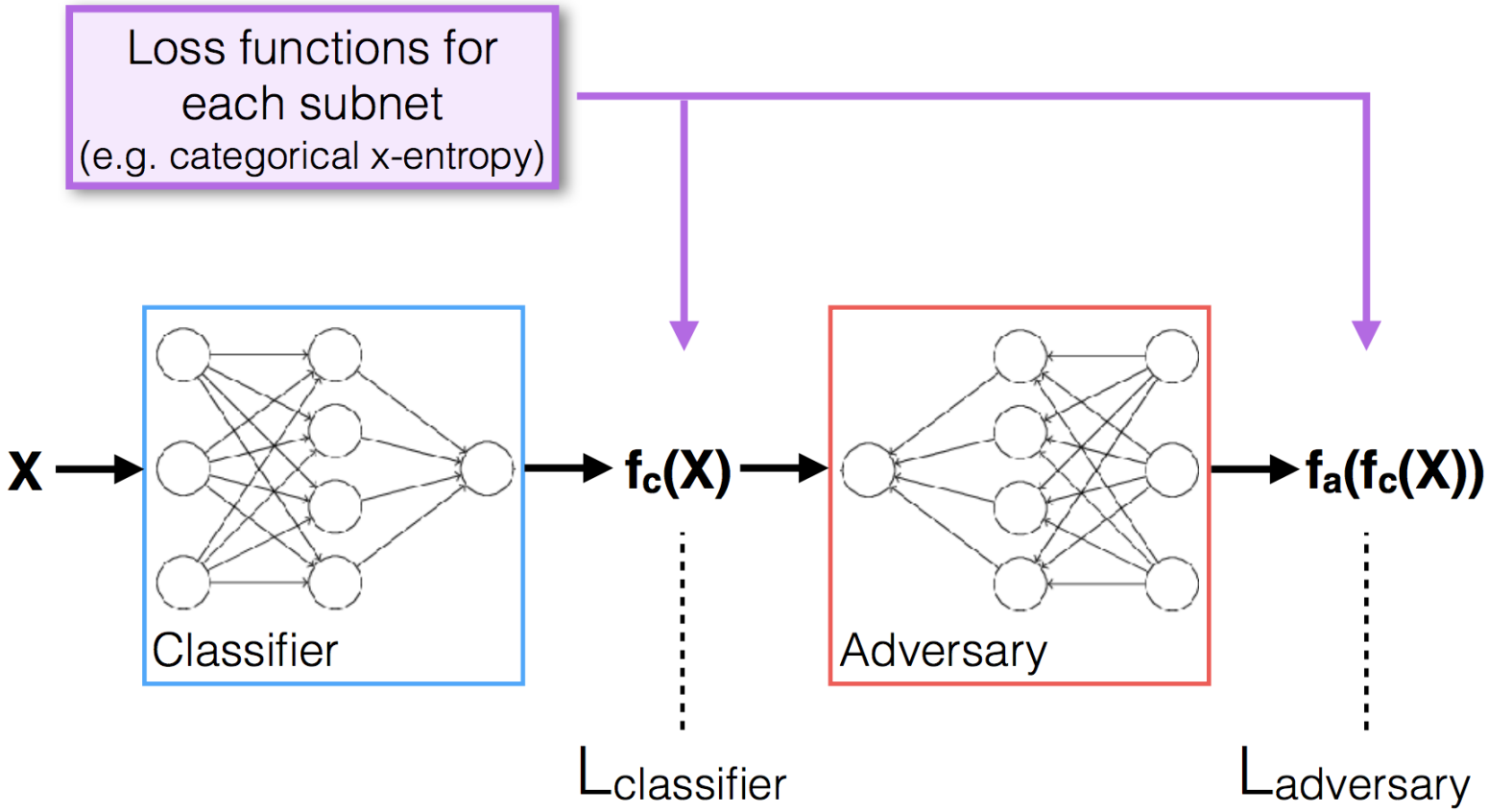


http://bit.ly/2GefGtq

Adversarial Approach

Chase_2017.03.22_IML.pdf (стр. 20 из 37)

Adversarial Decorrelation



Chase Shimmin (Yale University) 20 http://bit.ly/2GefGtq

Adversarial Approach

Chase_2017.03.22_IML.pdf (стр. 21 из 37)

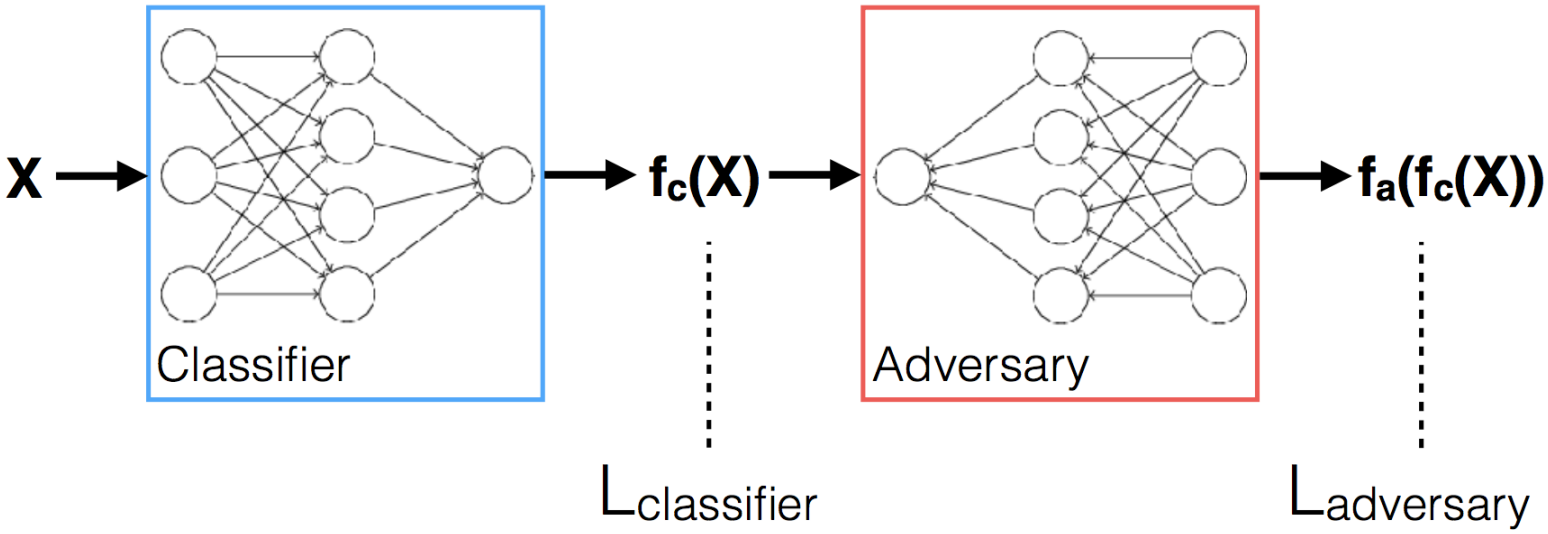
Adversarial Decorrelation

Simultaneously minimize:

$L_{adversary}$

and

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



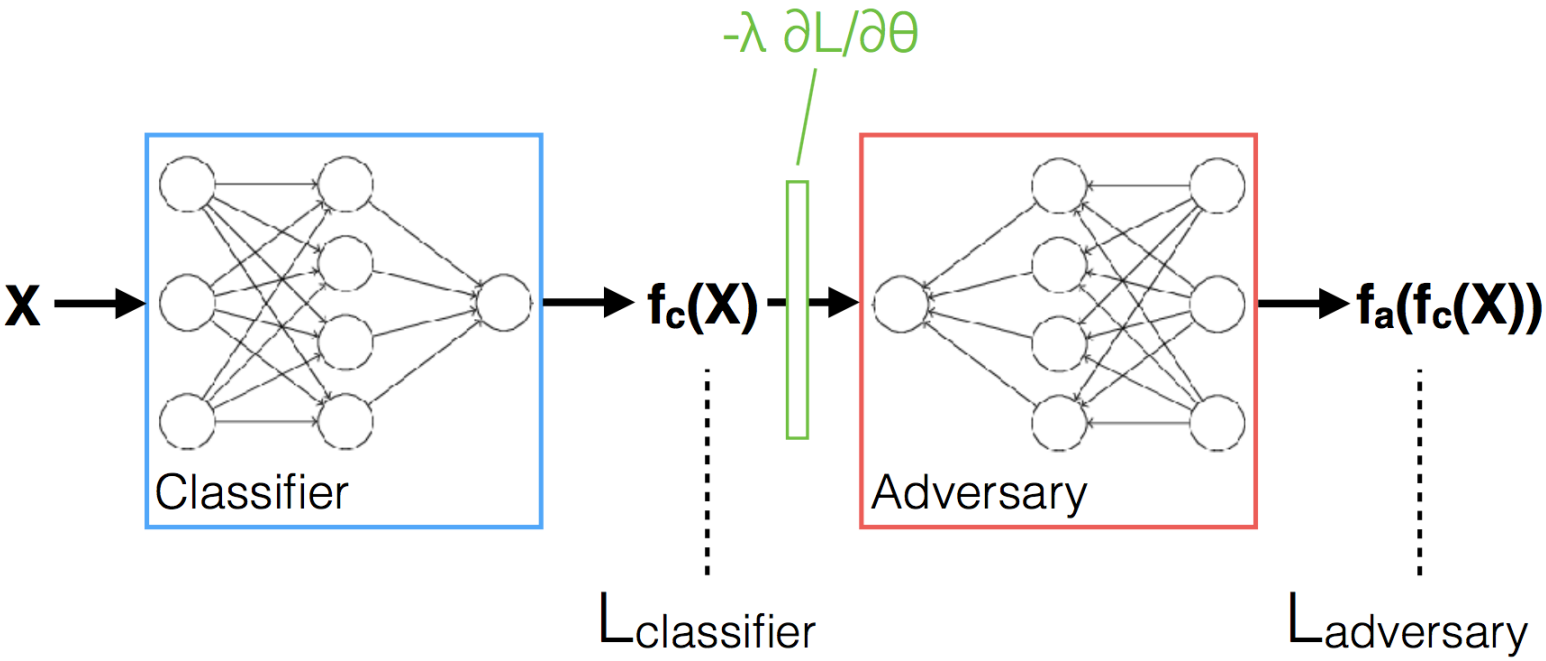
Chase Shimmin (Yale University) 21 http://bit.ly/2GefGtq

Adversarial Approach

Chase_2017.03.22_IML.pdf (стр. 23 из 37)

Training

- Simultaneous optimization achieved with **gradient scaling layer**
- Signal events are given zero weight in adversary loss



Chase Shimmin (Yale University) 23 http://bit.ly/2GefGtq

Adversarial Approach

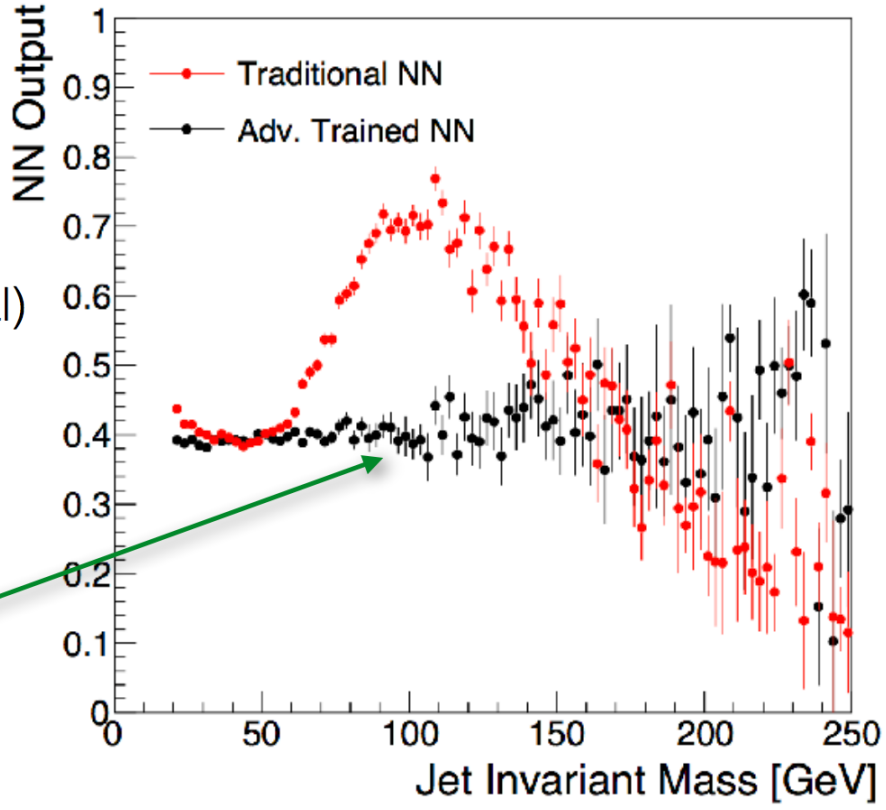
Chase_2017.03.22_IML.pdf (стр. 24 из 37)

Results

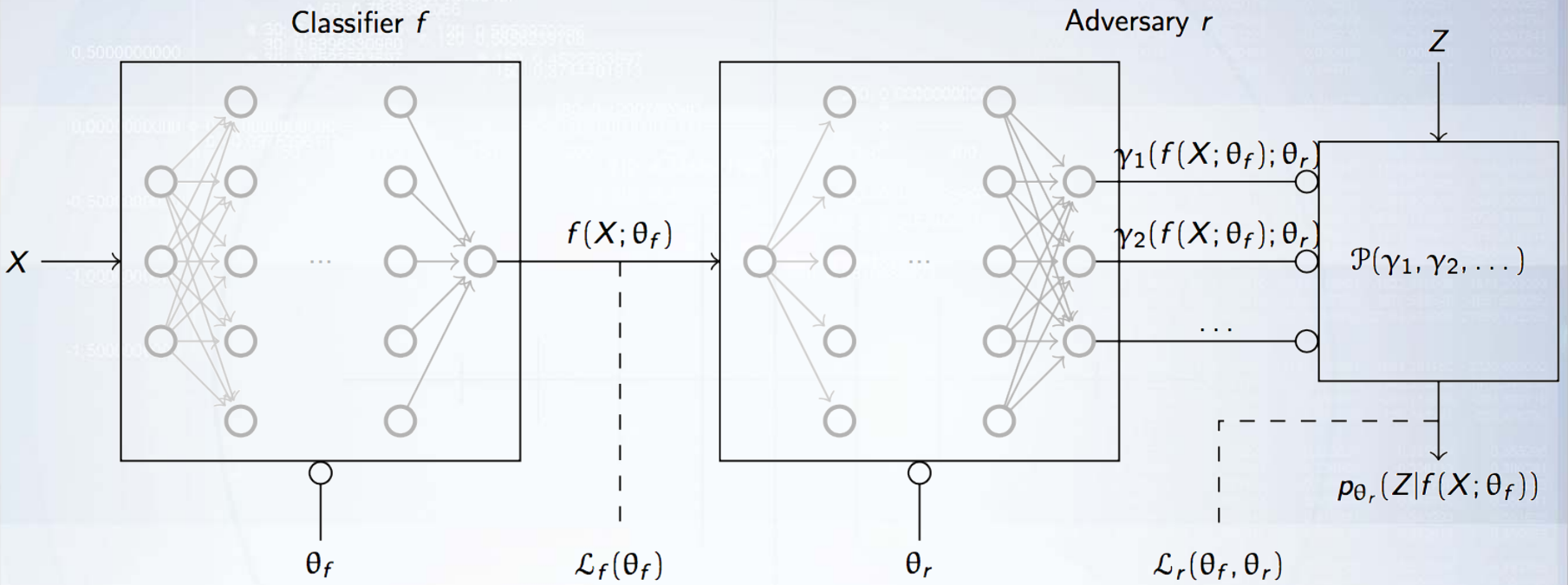
Training on ~200k MC events:

Sherpa γ+jet	(BG)
MG5 γ+Z'	(Signal)
Pythia + Delphes	(Both)

✓ Tagger profile much flatter



Goin Deeper with Adversarial Training



$$\hat{\theta}_f, \hat{\theta}_r = \arg\min_{\theta_f}\max_{\theta_r} \mathcal{L}_f(\theta_f) - \mathcal{L}_r(\theta_f,\theta_r)$$

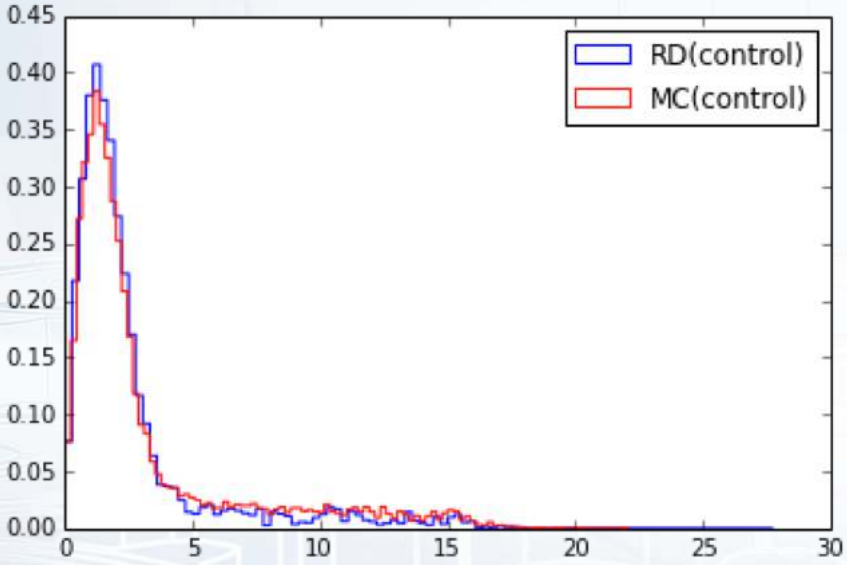
Here the adversary part identifies PDF *parameters* that can be used to infer Z (decorrelated feature) from classifier output.

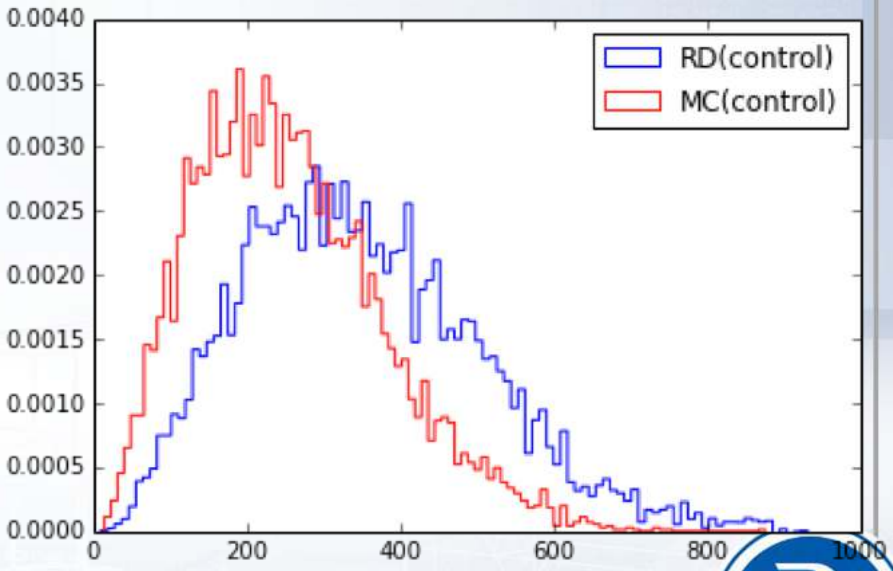
Intuitively, r penalizes f so it is impossible to reconstruct Z.

http://bit.ly/2GejPgY

Real Data vs Simulation Agreement

- Classifier is trained for simulated signal vs real background
- Not all features are perfectly simulated: MC and real data disagree (plots below)
- **Problem**: efficiency is overestimated





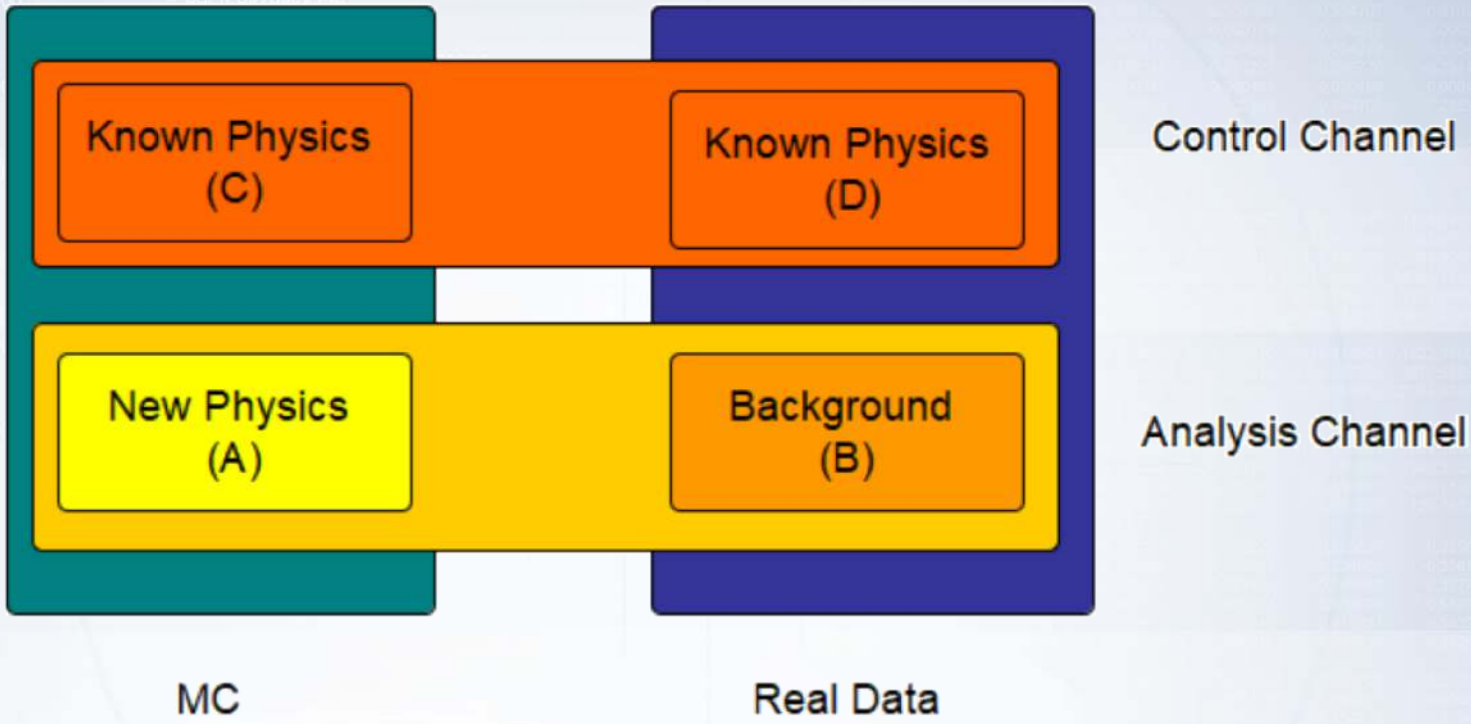
Approach

- Pick control channel of similar topology: $D_s \rightarrow \phi\,(\mu^+\mu^-)\,\pi^-$;
- $D_s \rightarrow \phi\,(\mu^+\mu^-)\,\pi^-$ is a much better known channel, can be extracted from the data;
- Compare performance of classifier on simulated and real samples using Kolmogorov-Smirnov test:

$$T = sup_x \left| F_1(x) - F_2(x) \right|$$

- Demand the distance to be below certain margin. How can we include this criteria in the training loop?

Data Doping

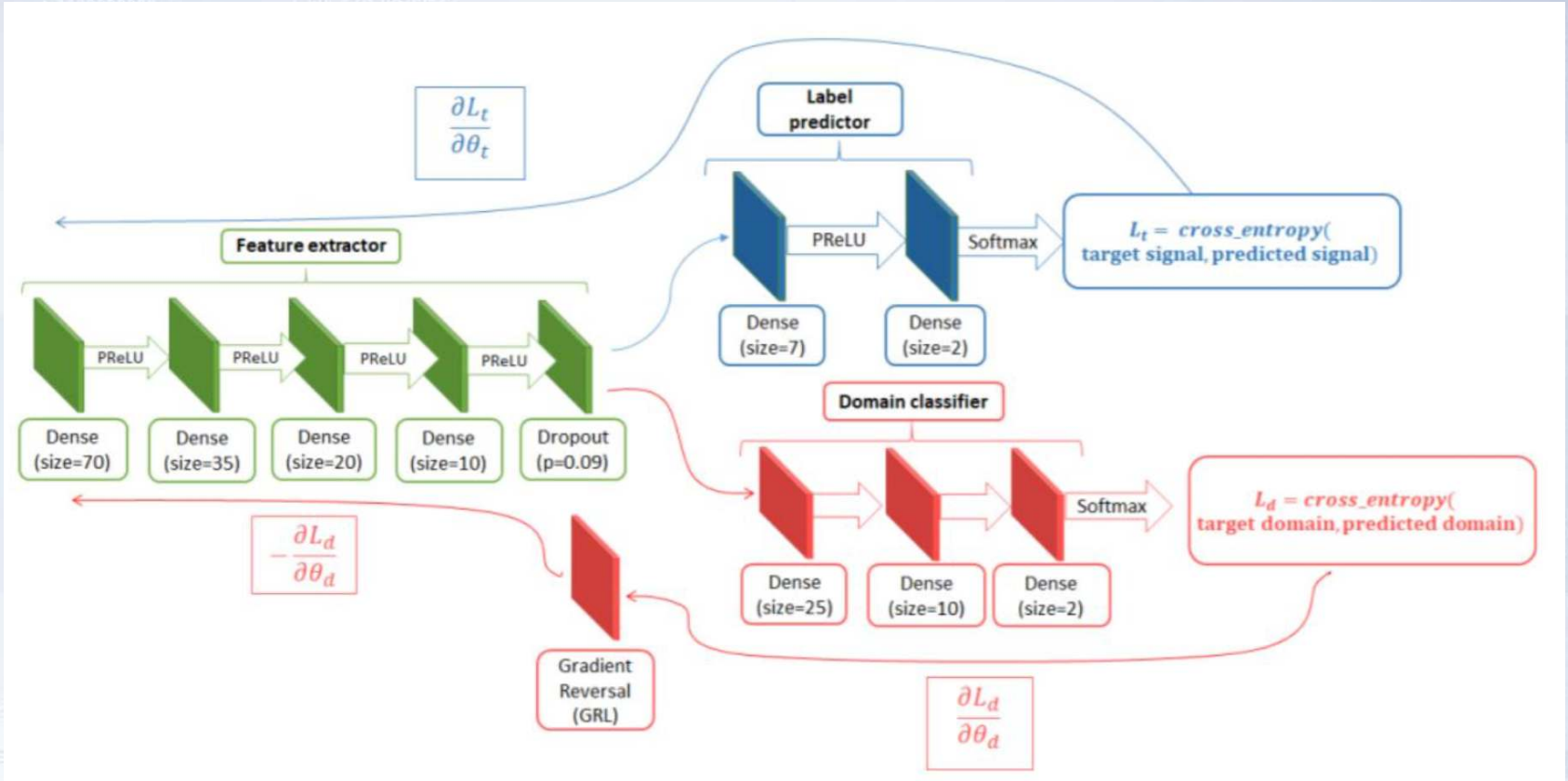


Let classifier discriminate A from B, not C from D:

- Add fraction of simulated signal (C) to the training sample with 'background' label ('*doping*')

Vicens Gaitan: http://bit.ly/2pLgVcy

Domain Adaptation by Gradient Reversal



- Feature extractor – builds meaningful representation (features);
- Label predictor – discriminates between signal and background;
- Domain classifier – discriminates between MC and real.

Domain Adaptation by Gradient Reversal

Let's compare the techniques on $\tau \rightarrow \mu\mu\mu$ dataset:

Metric \ Model	Mass-aware Classifier	Data Doping	Domain-adaptation
AUC (truncated)	**0.999**	0.9744	0.979
KS (< 0.09)	0.18	0.087	**0.06**
CvM (< 0.002)	0.0008	0.0011	**0.0008**

- By varying learning rate for feature extractor and domain classifier it is possible to trafeoff classifier quality to degree of agreement.
- More details you can find in the paper by A. Ryzhikov et al: http://bit.ly/2GHCs06

Bonus: Hacking Uniformity and Agreement Tests

In some cases it is possible to reconstruct the mass feature and technically avoid uniformity test by the following procedure:

- Train classifier fc that takes advanteage of the feature (presumably it will have unwanted peak). It will give good performance by fail uniformity test;
- Elevate fc output to very high power (e.g. 5000) and add random noise – it will pass uniformity test;
- Combine with poor random classifier – it will pass agreement test as well.
- Still will be physically meaningless.

https://www.kaggle.com/vicensgaitan/clipping-spreading

Conclusion

- Special cases of data analysis in HEP
  - Uniformity
  - Agreement
- Some ML techniques still can be helpful!
  - Ensembles of classifiers + special loss or Adversarial Networks