Uncertainty reduction by gradient descent

Victor Estrade[1], *Cécile Germain*[1], *Isabelle Guyon*[1], *David Rousseau*[2]

[1]*LRI, University of Paris-Sud, University of Paris-Saclay*

[2]*LAL-Orsay, University of Paris-Sud, University of Paris-Saclay*

Error sources

$$measure = value \pm \sigma_{stat} \pm \sigma_{syst}$$

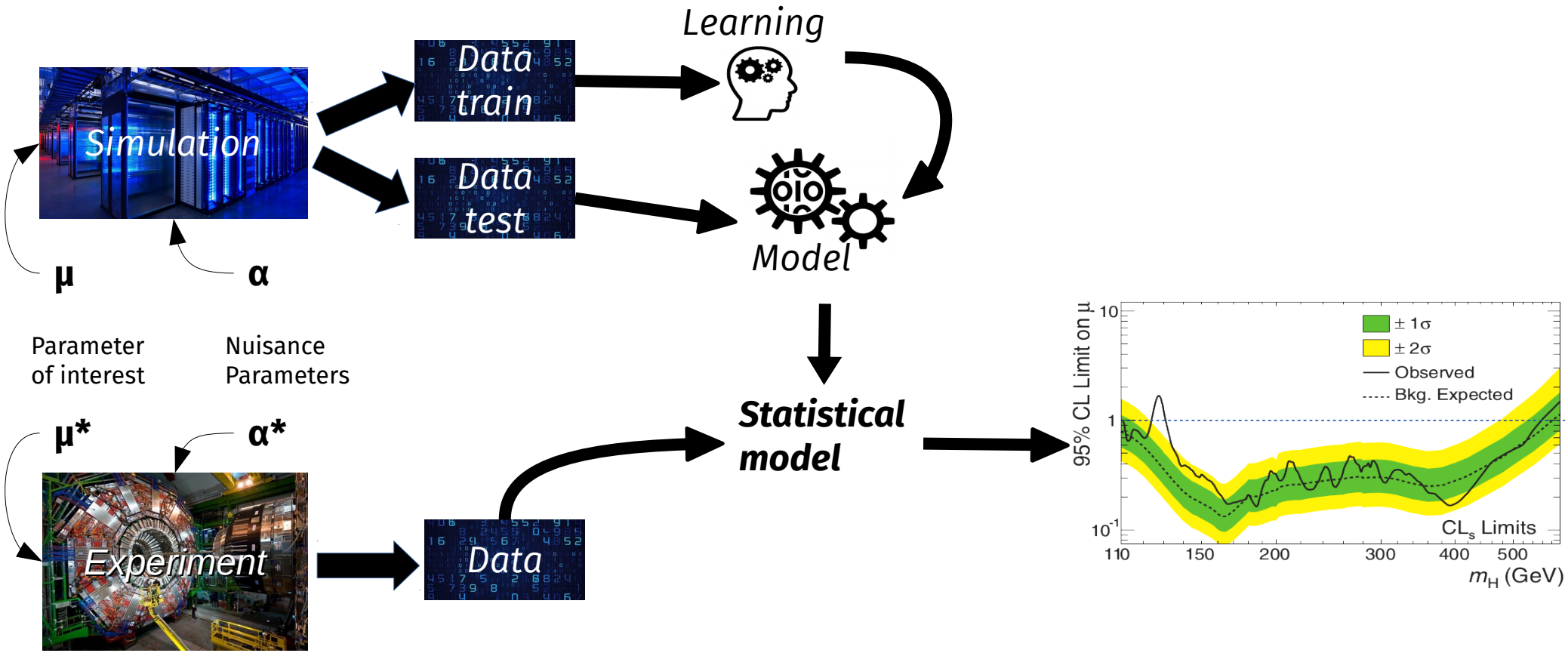
Statistical errors

- *Lack of data*
  (empirical ≠ asymptotic)
- *Noise*

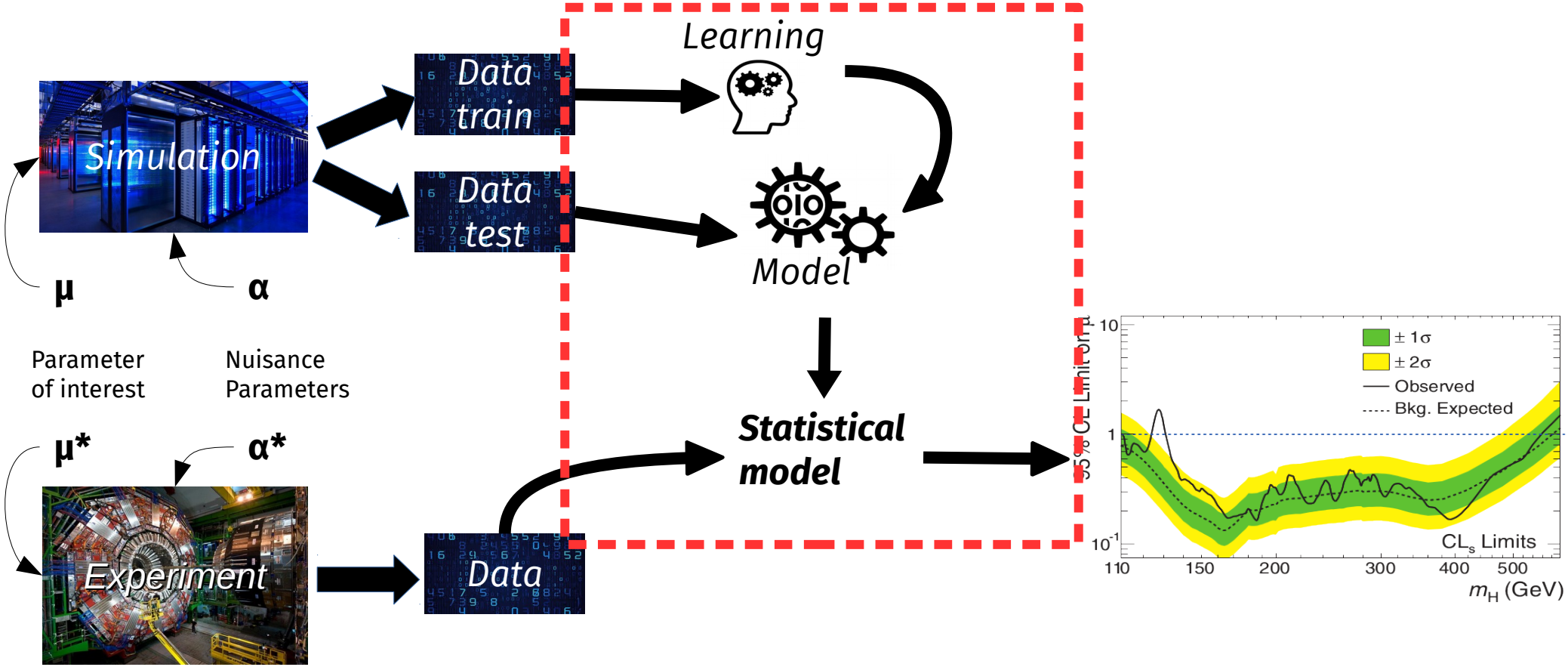
Systematic errors « known unknowns »

- *Apparatus responses*
- *Simulation impefections*
- *Lack of theoretical knowledge*
- *Chosen analysis tool*
- *etc*

Pipeline


Simulation
Data train
Data test
Learning
Model
μ
α
Parameter of interest
Nuisance Parameters
μ*
α*
Experiment
Data
Statistical model
95% CL Limit on μ
± 1σ
± 2σ
Observed
Bkg. Expected
CLs Limits
mH (GeV)

Pipeline


Simulation
Data train
Data test
Learning
Model
μ
α
Parameter of interest
Nuisance Parameters
μ*
α*
Experiment
Data
Statistical model
± 1σ
± 2σ
Observed
Bkg. Expected
CLs Limits
95% CL Limit on μ
mH (GeV)

Zoom in statistical model

- Looking for **confidence interval** of the maximum likelihood estimator

$$(\hat{\mu},\hat{\alpha})=\underset{\mu,\alpha}{argmax}\ \underbrace{p(D|\mu,\alpha)}_{intractable}$$

- Likelihood is intractable → numerical approximation
- Avoid dimensionality curse : machine learning to reduce dimension and compute summary statistic *F(D)*

$$p(F(D)|\mu,\alpha)\approx p(D|\mu,\alpha)$$

Measuring a cross section

- Poisson counting prossess

$$n \sim Poisson(\mu s + b)$$

$$L(\mu) = Poisson(n | \mu s + b)$$

$$L(\mu, \alpha) = Poisson(n | \mu s(\alpha) + b(\alpha)) L_c(\alpha)$$

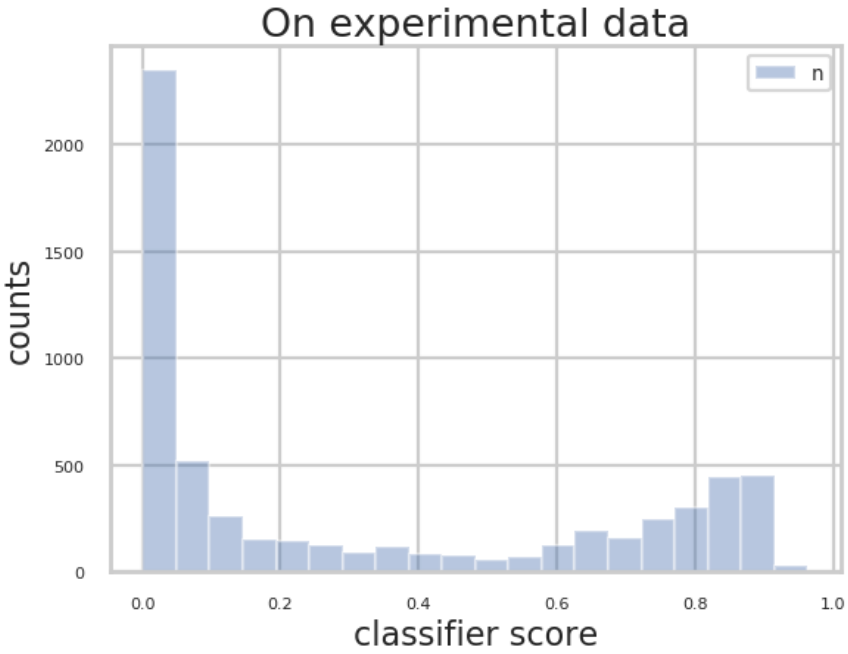
n = # events in experimental data
s = # expected signal events
b = # expected background events
μ = deviation from SM
(parameter of interest)

- Machine learning and simulation hidden in *s* and *b*

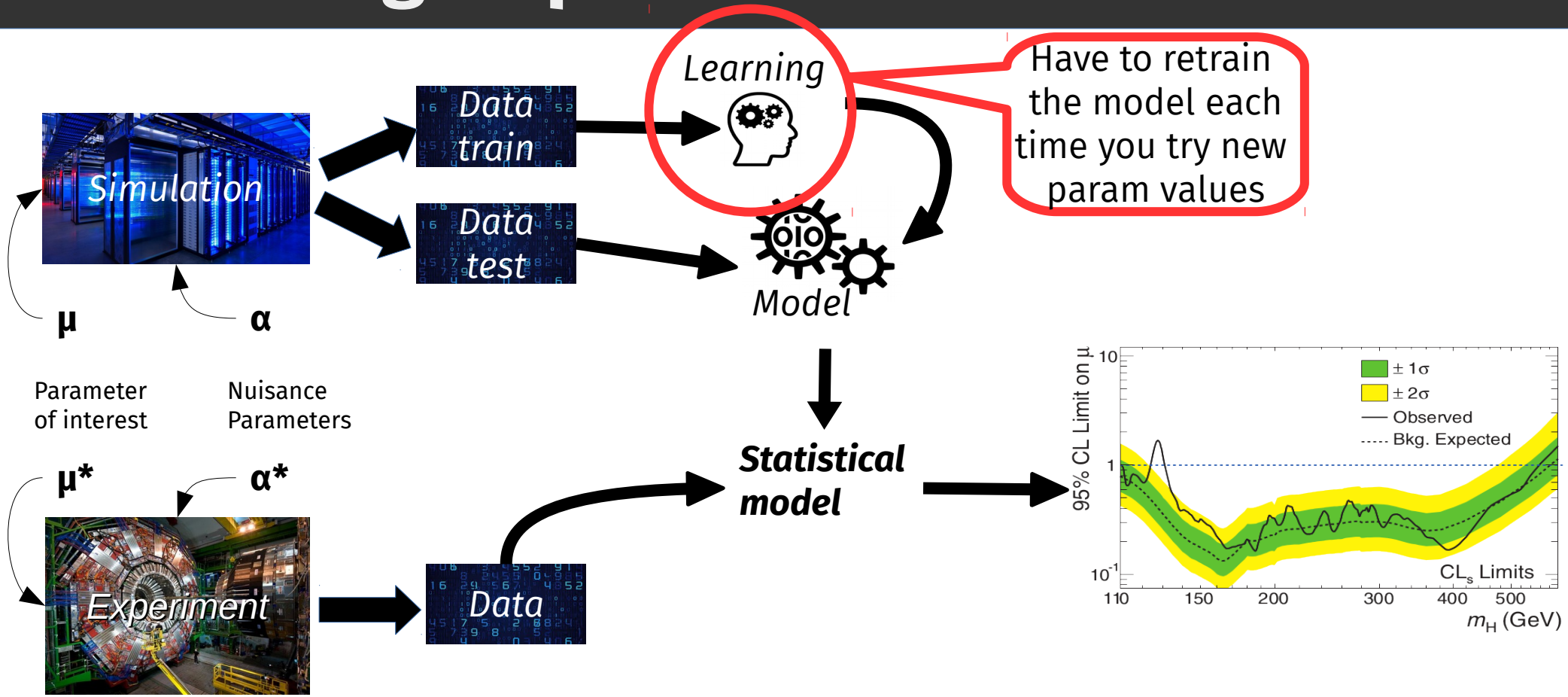
Standard method : Classifier

[Cranmer et al., 2015]

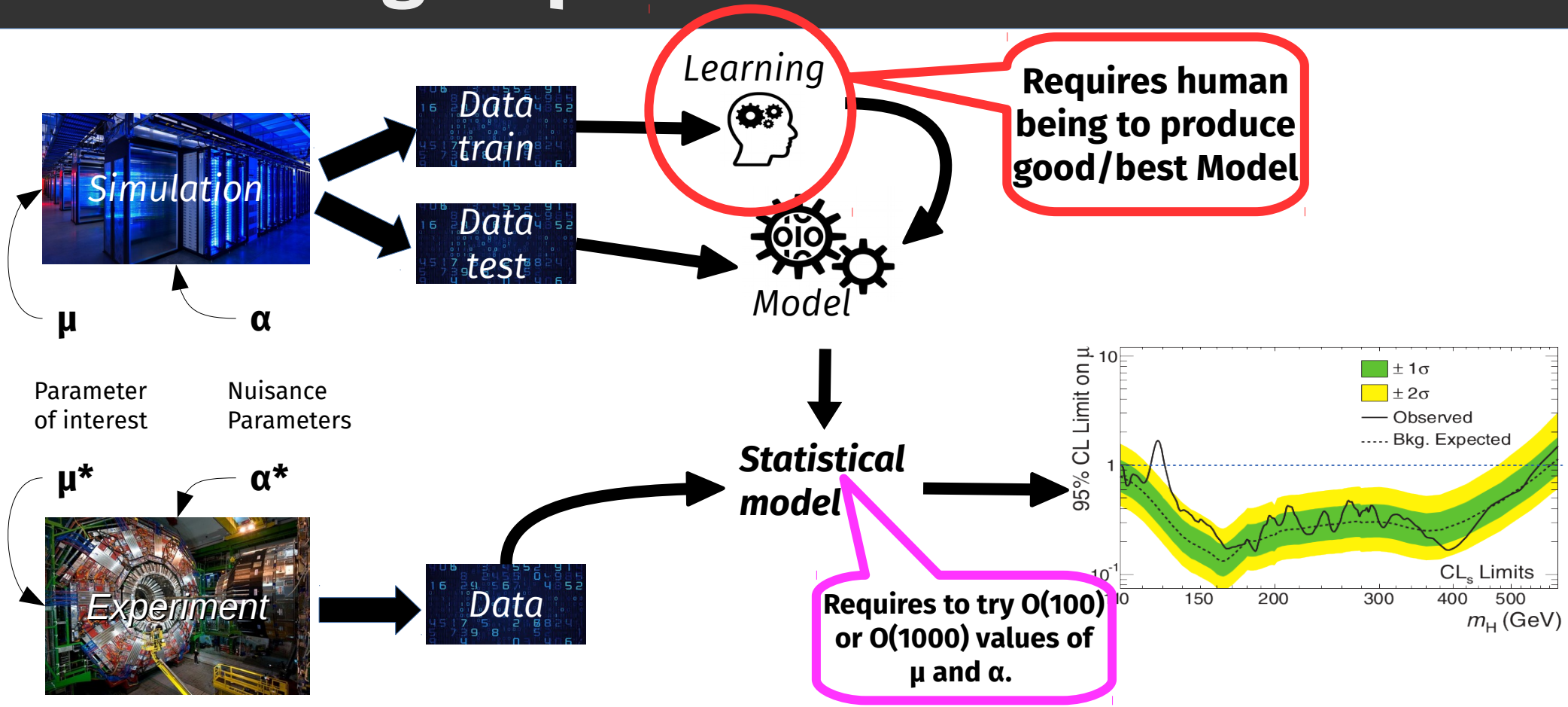
- Classifier score function as sufficent summary statistic



Learning requires human intervention



Learning requires human intervention



Naive ways

Ignore it at training time !
(baselines)

- Simple neural network classifier
- Simple BDT : Gradient Boosting

Ignore sensitive features !
(not always possible)

- Loose valuable information

Data augmentation

- Sample nuisance parameter and generate data
- Train on a mixture of simulator
- Hope the model becomes robust

Tangent Propagation in a nutshell

[Simard et al., 1991] [Rifai et al., 2011]

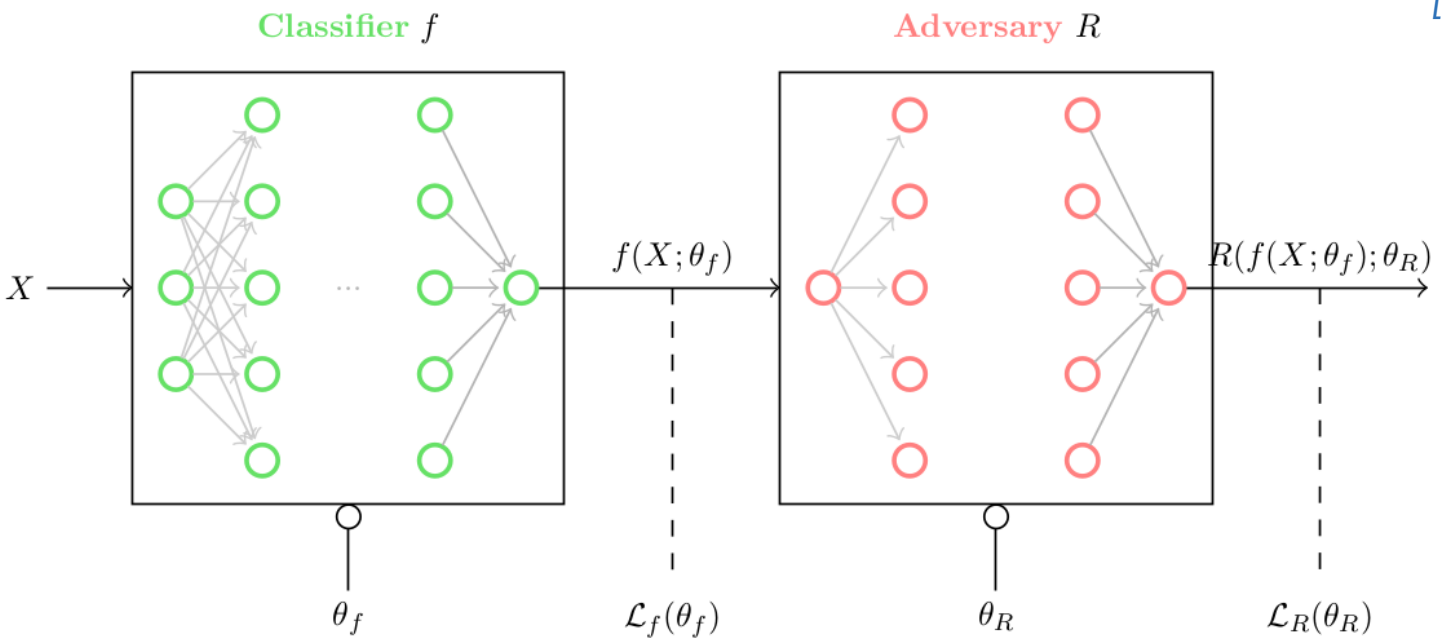
- Output of the model should be invariant according to some known transformation of the input
- Regularize the derivative of the model according to the parameter of the transformation

$$loss = E_{usual} + \lambda \sum_{x \in Data} \left| \frac{\partial f(d(x, \alpha))}{\partial \alpha} \right|_{z=0}^{2}$$

- Less data intensive than data augmentation
- Easier nowdays to compute with automatic differentiation software

Pivot Adversarial Neural Network

[Louppe et al., 2016]

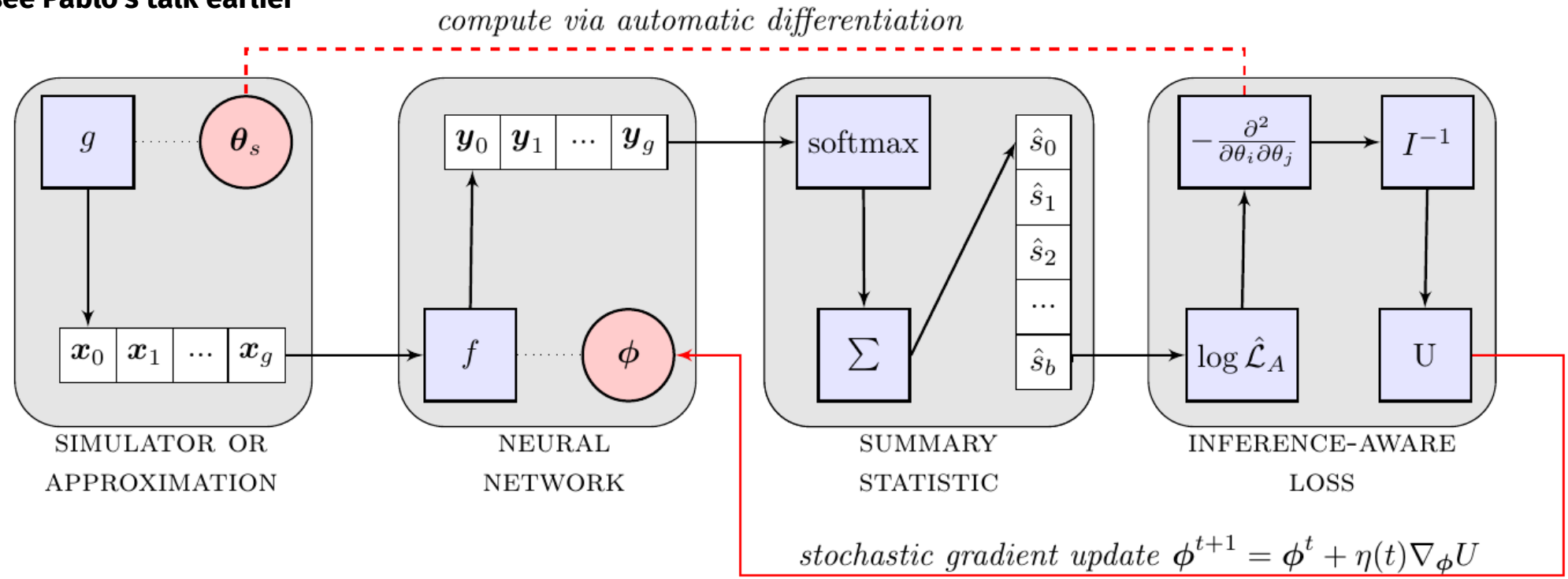


- Learn the regularization (adversarial training)
- Makes it impossible to reconstruct *z* from the model's output

INFERNO

[de Castro et al., 2018]

see Pablo's talk earlier



Other challengers

- QBDT *[Li-Gang Xia, 2018]*
- Direct optimisation of the discovery significance *[Elwood et al., 2018]*
  - (Poster session 1)
- SaBDT (see 1st talk of this afternoon)
- Baldi stuff : parametrized model
- Mining gold
- Others ?

Higgs ML Challenge open dataset

(Dataset) [Adam-Bourdarios et al., 2014]

- Simulation of H→ ττ decay
- 818238 events 13 DER & 17 PRI features
- We included 5 systematic effects
  - Tau energy scale
  - Jet energy scale
  - Lep energy scale
  - Soft MissingET term
  - Small but poorly known background

- Assumed calibration gives gaussian constrains with :

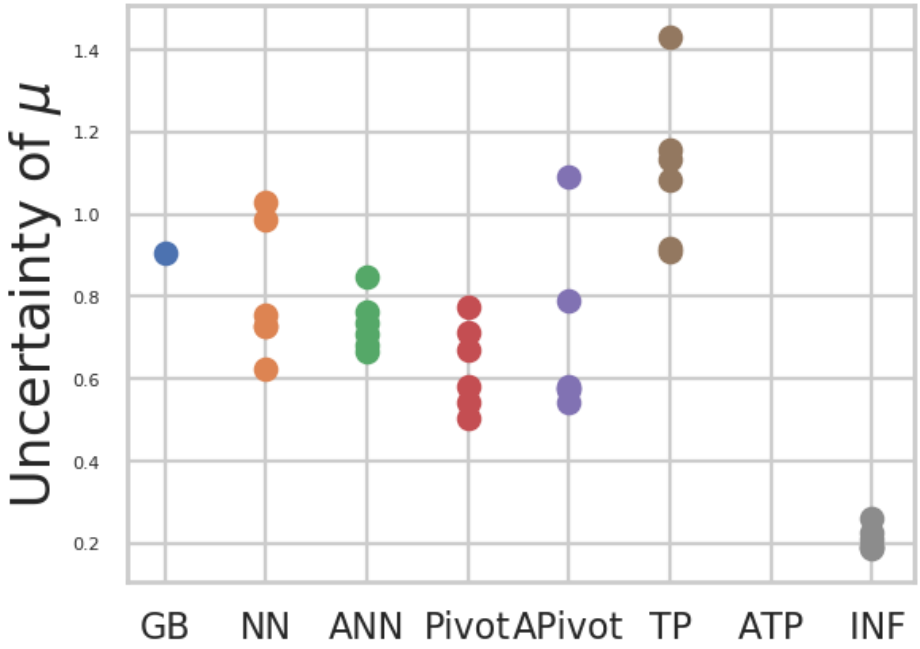
Systematic	mean	std
Tau ES	1.0	0.05
Jet ES	1.0	0.05
Lep ES	1.0	0.01
Soft term	2.7	0.5
Nast Bkg	1.0	0.5

$$L_c(\alpha) = \prod_i Normal(\alpha_i | mean_i, std_i)$$

Computing confidence interval

- Big thanks to Pablo for giving his help and code !
- In the future : use iminuit package

Preliminary results



Where to go from here

- `Use iminuit` package
- Add missing challengers in the benchmark
  - or turn it into a ML challenge to make other people work on it 😎
- Try another dataset
- Always more systematic effects
- Try unbinned likelihood

Conclusion

- Should make it easier to compare models in a realistic pipeline and get actual performances instead of surrogate (AUC & other)

  Take home message :

  Looking for properties at the dataset level ? Build a model that works at the dataset level !

Thank you for your attention !

Questions ?

Tangent Propagation in a nutshell

- Output of the model should be invariant according to some known transformation of the input
- Regularize the derivative of the model according to the parameter of the transformation

$$loss = E_{cross\,entropy} + \lambda \sum_{x \in Data} \left| \frac{\partial f(d(x,z))}{\partial z} \right|_{z=0}^{2}$$

- Less data intensive than data augmentation
- Jacobian vector product trick : compute this derivative with a forward propagation through a ”linearised” network.

$$\frac{\partial f(d(x,z))}{\partial z} = \nabla_x f(x) \frac{\partial d(x,z)}{\partial z}$$