Likelihood ratio in many dimensions

Using neural networks for effective field theory

Jonas Rübenach
Hamburg, 2019-07-12

Theory

Search for BSM physics at high energies

- Effective field theory (EFT)
  - Approximation for new physics at energies beyond the current scale

$$\mathcal{L}_{\mathrm{EFT}} = \mathcal{L}_{\mathrm{SM}} + \sum_i \frac{c_i}{\Lambda^2}\mathcal{O}_i + \mathcal{O}\left(\frac{1}{\Lambda^3}\right)$$

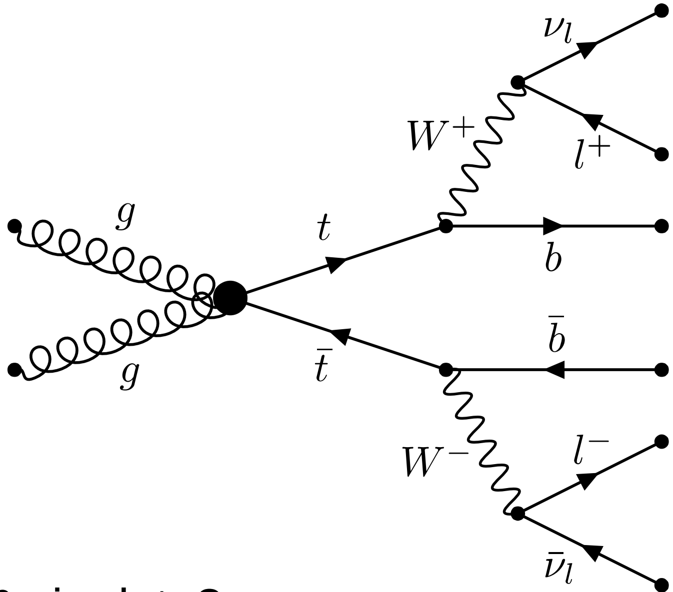
    - Wilson coefficient c_i
    - Cut-off energy scale $\Lambda = 1\ \mathrm{TeV}$
    - EFT operator $\mathcal{O}_i$

- **Goal**: Set limits on c_i
  - Note: Generally no signal/background distinction possible due an interference term

Process

Top-quark pair production and decay

- Top-quark pair decaying into electrons or muons
- Adding only one EFT operator $\mathcal{O}_{\mathrm{tG}}$
  - Introduces $gg t\bar{t}$ vertex and modifies top-gluon coupling
- Utilizing 23 “high-level” observables
- **Question**: Can we rule out the existence of $\mathcal{O}_{\mathrm{tG}}$ in data?



Likelihood ratio

How to probe for hypotheses

- Two different hypotheses, e. g. assuming two different values c_{tG} and c'_{tG}.
- Probability to observe specific values for a set of observables x

$$p(x|c_{tG})$$

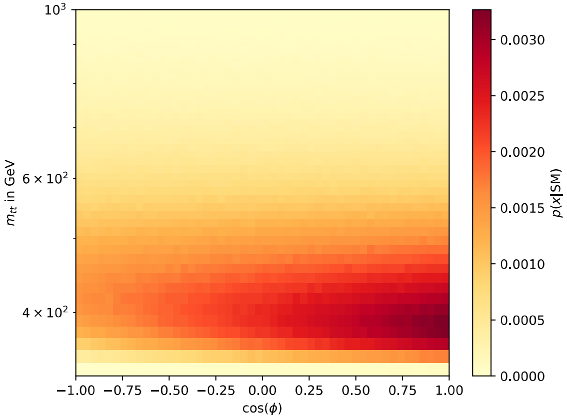
- Likelihood ratio

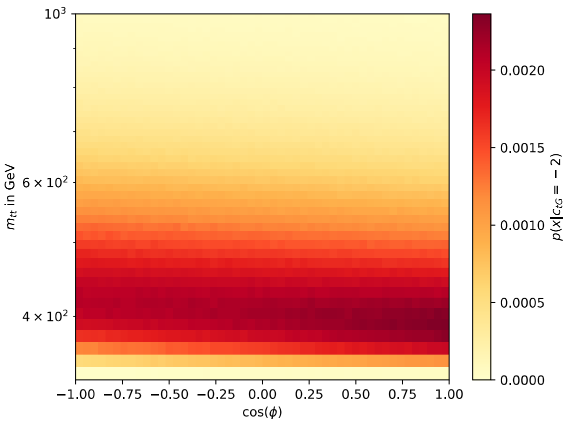
$$r(x|c_{tG}, c'_{tG}) = \frac{p(x|c_{tG})}{p(x|c'_{tG})}$$

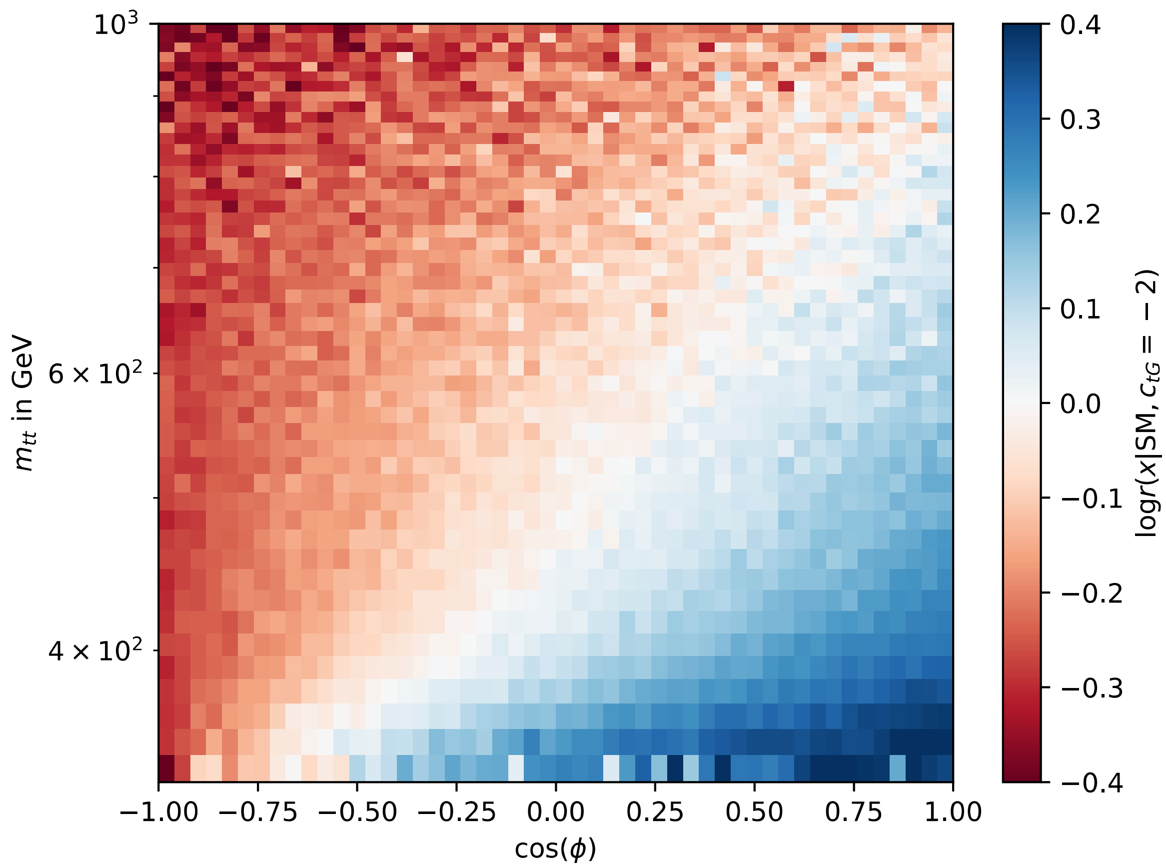
  - Provides most powerful tests
  - Hard to compute, especially for high-dimensional x

Likelihood ratio

Low-dimensional example: 15 million MC events in 2D histograms







Machine learning approach

Described in a recent paper

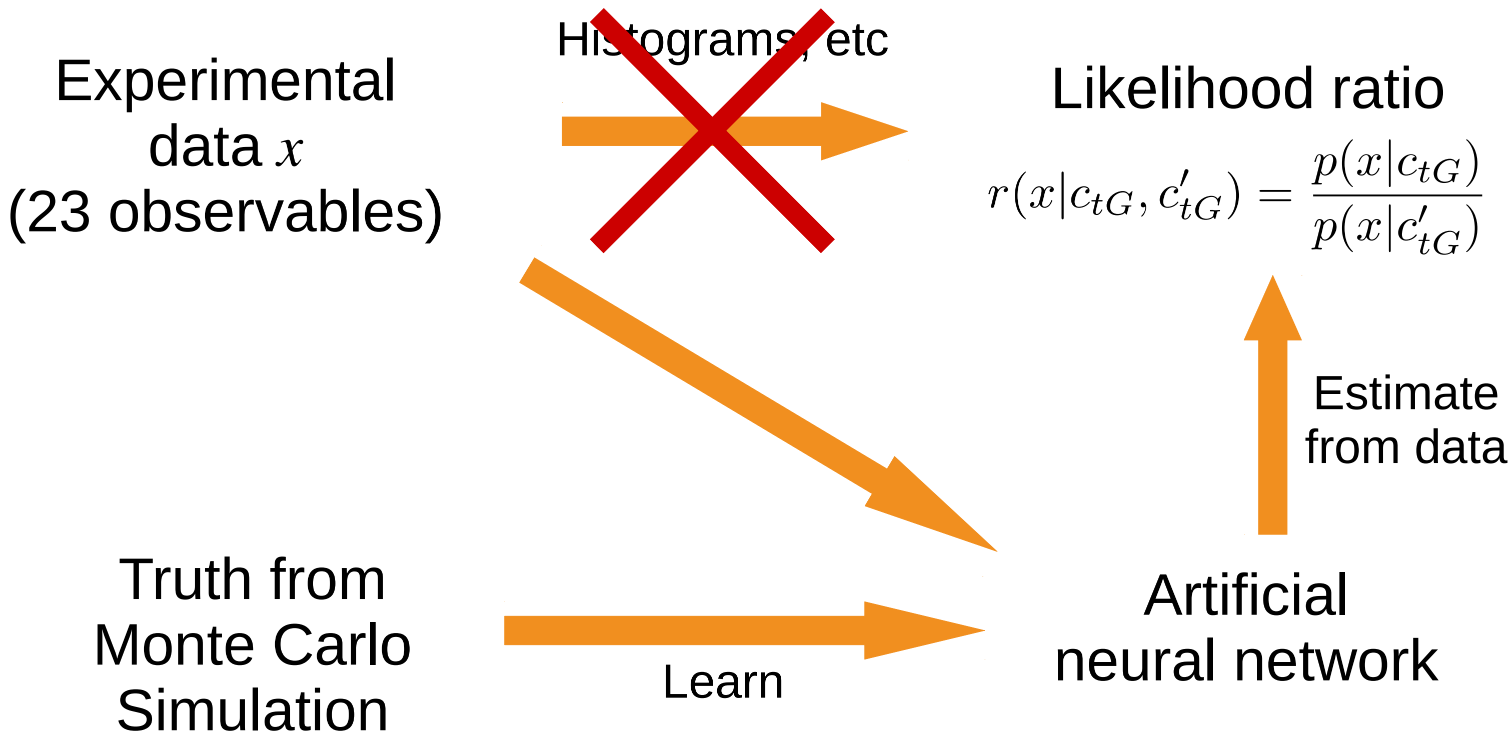
A Guide to Constraining Effective Field Theories with Machine Learning

Johann Brehmer,[1] Kyle Cranmer,[1] Gilles Louppe,[2] and Juan Pavez[3]
[1] *New York University, USA*
[2] *University of Liège, Belgium*
[3] *Federico Santa María Technical University, Chile*
(Dated: 30th July 2018)

- In a recent paper multiple approaches to estimate the likelihood ratio using neuronal networks were described (arXiv:1805.00020)
  - Classification
  - **Regression**
  - Local score and density estimation
- Works for a single EFT operator as well as for multiple ones
- Can also be used for hypothesis tests unrelated to EFT

Machine learning approach

Using Monte-Carlo truth to gain sensitivity



Machine learning approach

Training a neural network for a regression on the likelihood ratio

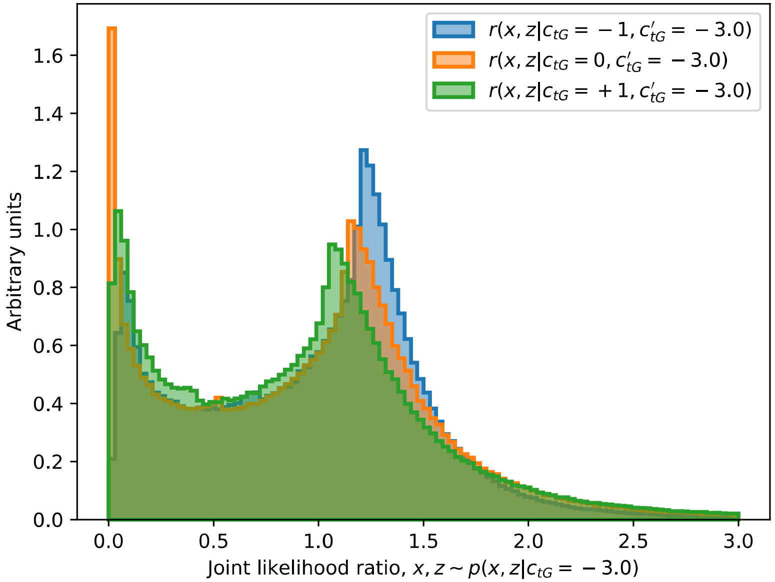
- **Idea**: Regress on the likelihood ratio using a neural network
  - High accuracy
  - Fast computation (in comparison to KDE or matrix element method)
- Problem: True likelihood ratio not available
  - Instead use joint-likelihood ratio, depending also on detector, shower and parton variables

$$
\begin{aligned}
r(x, z_{\mathrm{all}}|c_{tG}, c'_{tG}) &= \frac{p(x, z_{\mathrm{d}}, z_{\mathrm{s}}, z_{\mathrm{p}}|c_{tG})}{p(x, z_{\mathrm{d}}, z_{\mathrm{s}}, z_{\mathrm{p}}|c'_{tG})} \\
&= \frac{p(x|z_{\mathrm{d}}, z_{\mathrm{s}}, z_{\mathrm{p}})p(z_{\mathrm{d}}|z_{\mathrm{s}}, z_{\mathrm{p}})p(z_{\mathrm{s}}|z_{\mathrm{p}})p(z_{\mathrm{p}}|c_{tG})}{p(x|z_{\mathrm{d}}, z_{\mathrm{s}}, z_{\mathrm{p}})p(z_{\mathrm{d}}|z_{\mathrm{s}}, z_{\mathrm{p}})p(z_{\mathrm{s}}|z_{\mathrm{p}})p(z_{\mathrm{p}}|c'_{tG})} \\
&= \frac{p(z_{\mathrm{p}}|c_{tG})}{p(z_{\mathrm{p}}|c'_{tG})}
\end{aligned}
$$

Machine learning approach

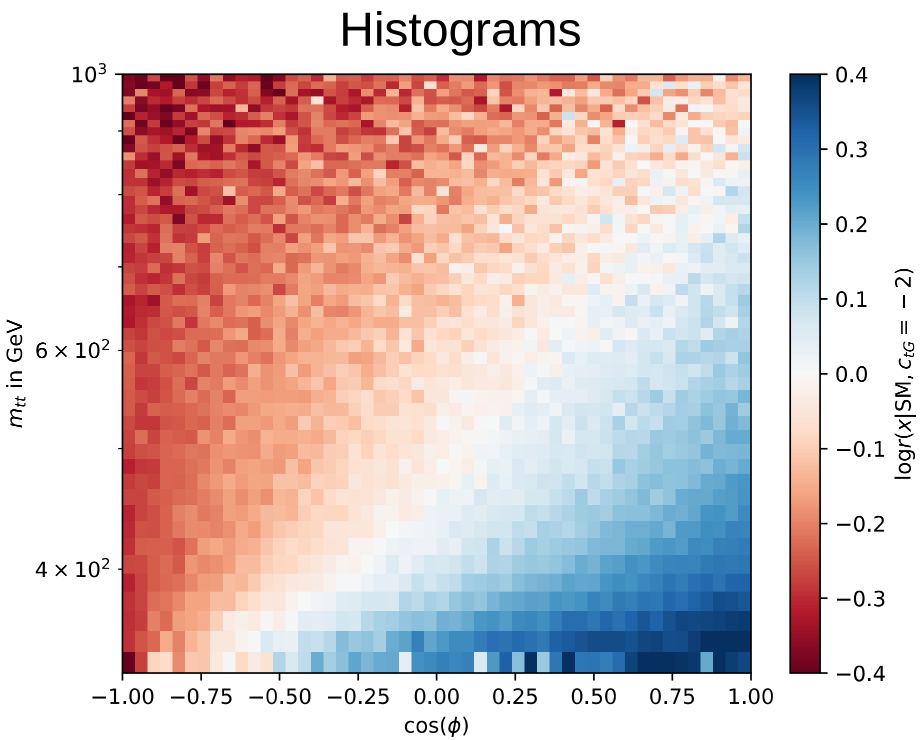
Training a neural network for a regression on the likelihood ratio

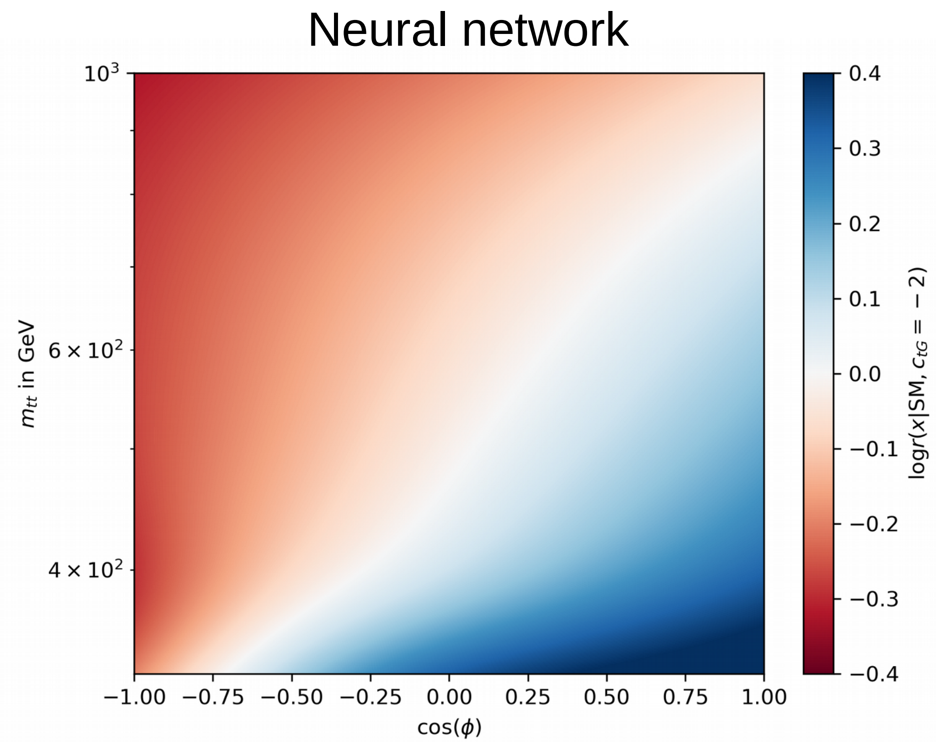
- Procedure
  - Compute joint-likelihood ratio from matrix elements of LO MC events using simple formula
  - Build a dense neural network with 3 to 6 hidden layers, tanh activations
  - Input event observables, regress on joint-likelihood ratio
  - Use mean squared error as loss
- Analytically it can be shown that the loss is minimized by $r(x|c_{\mathrm{tG}})$



Machine learning approach

2D example: Histogram vs neural network





Improvements for the neural network

- For limits, knowledge of $r(x|c_{tG},c'_{tG})$ for many c_{tG}-values is needed
  - Train independent neural network for every value **or**
  - "Parameterized model": Input c_{tG}-values alongside observables
- Input more information from the Monte Carlo, notably the score t
  - Joint score is available and behaves analogous to the joint-likelihood ratio

$$t(x|\hat{c}_{tG}) = \frac{\partial}{\partial c_{tG}} \log p(x|c_{tG})|_{\hat{c}_{tG}}$$

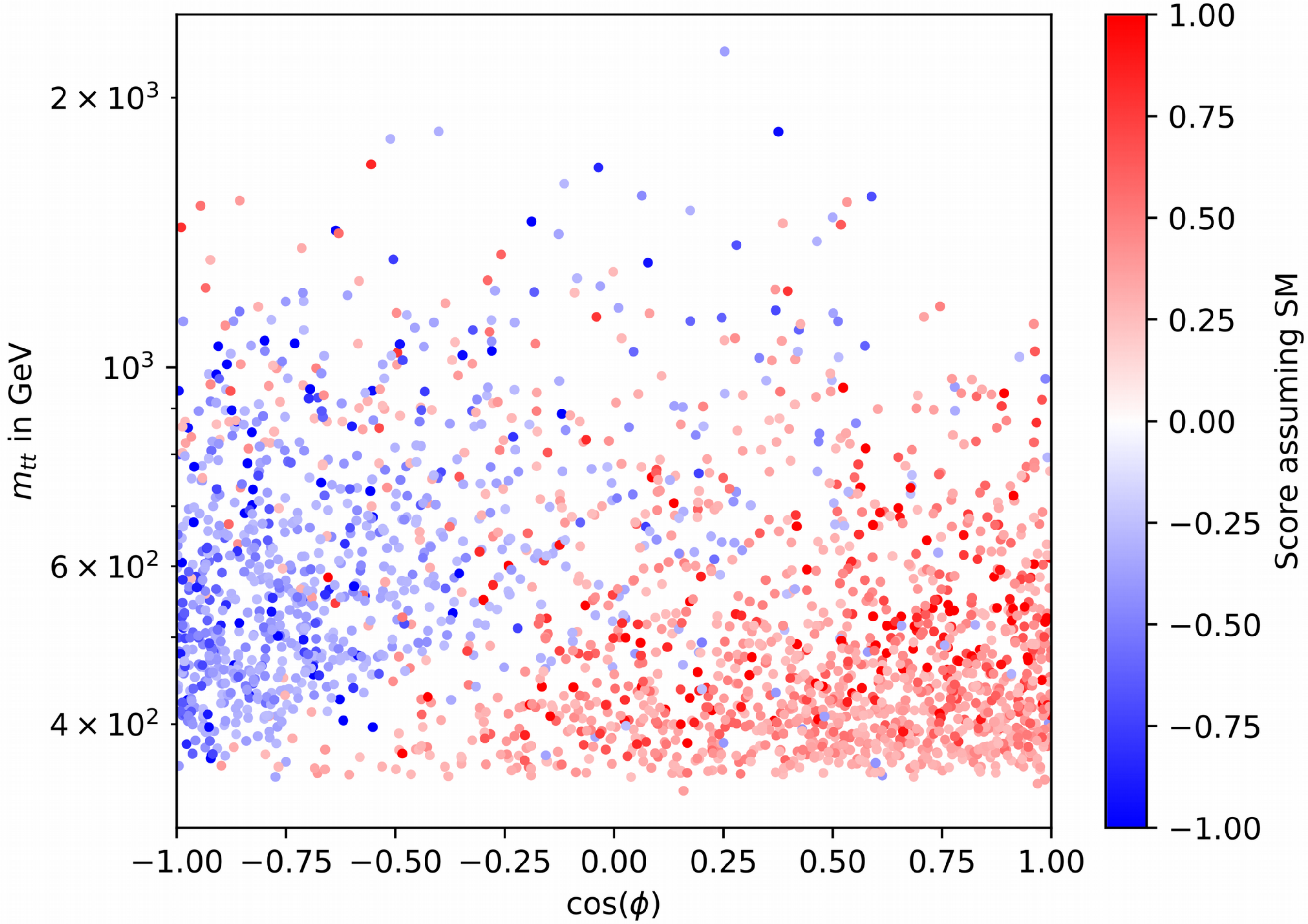
$$L = \mathrm{MSE}_r + \alpha \mathrm{MSE}_t$$

Improvements for the neural network

Joint score

$$t(x, z_{\mathrm{all}}|\hat{c}_{tG}) = \frac{1}{p(z_{\mathrm{p}}|\hat{c}_{tG})} \frac{\partial}{\partial c_{tG}} p(z_{\mathrm{p}}|c_{tG})|_{\hat{c}_{tG}}$$

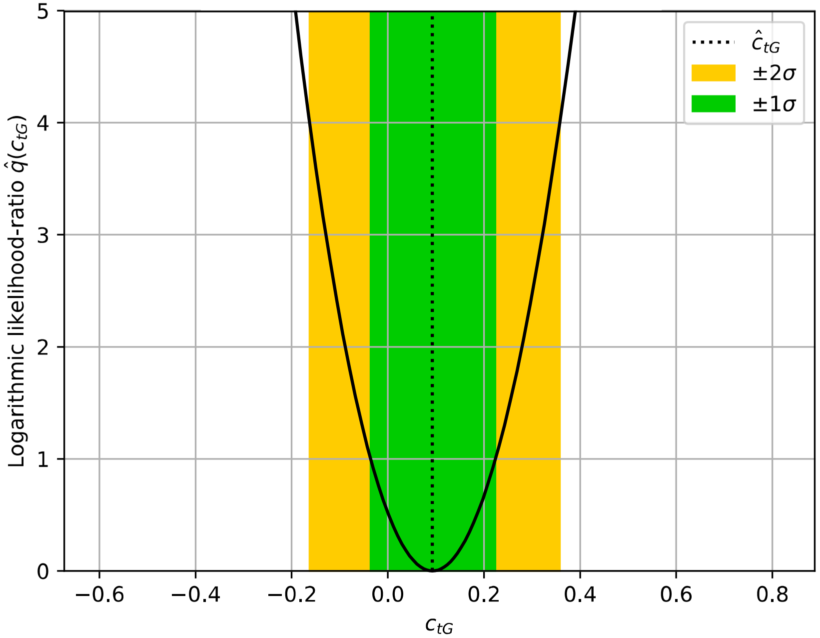
MC events with cut |t| > 0.25



Setting limits

Asymptotic properties of the likelihood ratio

- For a large number of events -2 log r is χ^2-distributed
  - Minimum gives estimate for the true c_{tG}-value
  - Deviations from 0 would point to new physics
- Evaluate neural network for a set of events to get limits
  - In this example, an independent set of 5000 standard model MC events were used



$$\hat{c}_{tG} = 0.093 \pm 0.13$$

Outlook

Pros

- Works for a large number of observables
- Observables can also be “low-level” four momenta
- Easily extendible for more operators
- Possible higher sensitivity than any histogram-based approach

Challenges for the future

- Tune hyperparamters
- Add detector simulation and nuance parameters
- Train on NLO simulation (instead of LO)
- Check performance on actual data
- Quality of the neural network itself hard to quantify