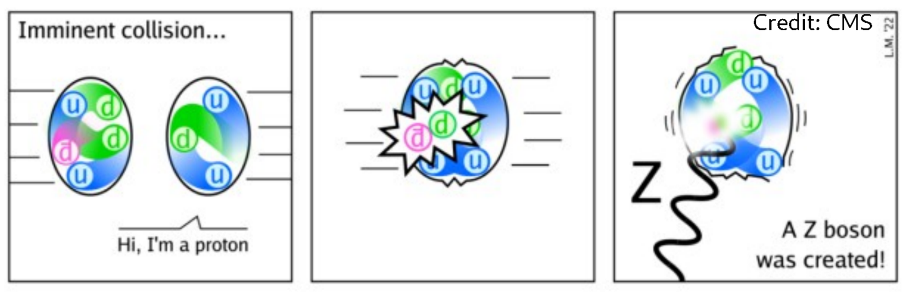
Normalising Flows for Phasespace Integration

Institute of Nuclear Physics Polish Academy of Sciences
Division NO4: Theoretical Physics

1. Large Hadron Collider (LHC) physics in a nutshell

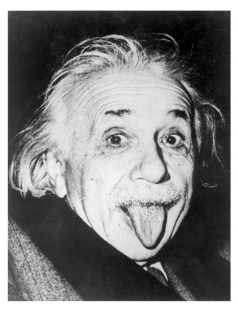
Collision of protons create particles



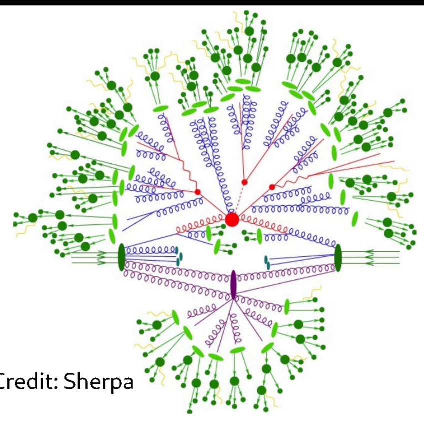
Energy ↔ Mass

E = mc^2

Credit: Geolino

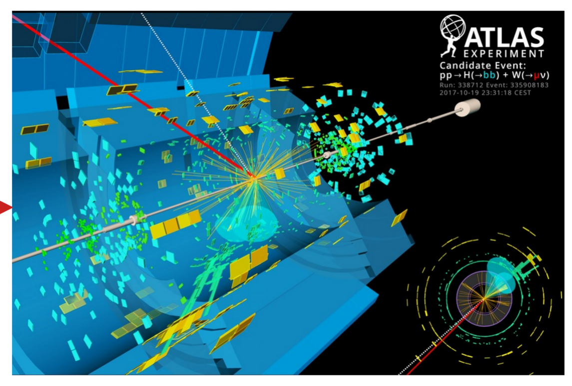


Quantum processes of this creation are **random** but some are more likely than others



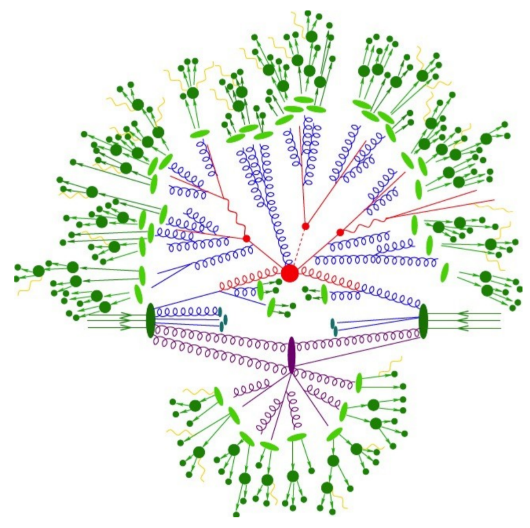
Credit: Sherpa

To study these processes we compare **predicted & measured** probabilities



2. How do we predict the outcome of particle collisions?

We have a model from that we can calculate all possible outcomes and their probability.
"Quantum Field Theory"



In practice we are interested in the sum of possibilities
→ called the "cross-section"

"We sum all possible outcomes"
=
"We integrate the phase space"

$$= \int \mathrm{d}\Phi \, |\mathcal{M}(\Phi)|^2$$

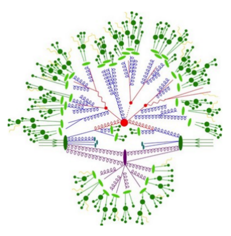
"Probability for a given outcome"
=
"Probability density"

3. Integration is a complicated problem!

If the "primitive function" is known integration is simple:

$$I = \int_a^b f(x)\, \mathrm{d}x = \Big[F(x)\Big]_{x=a}^{x=b} = F(b) - F(a)$$

What if you **don't** know the primitive function?
→ an unsolved and incredibly difficult problem!



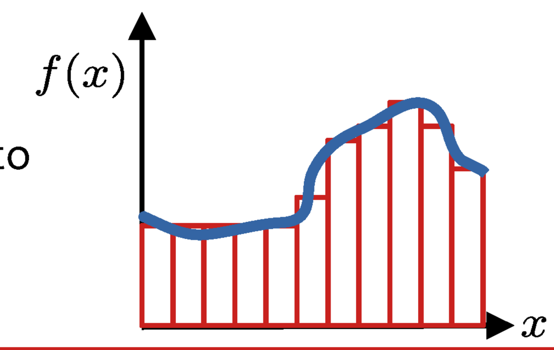
$$= \int \mathrm{d}\Phi \, |\mathcal{M}(\Phi)|^2$$

→ phase space has **many dimensions**
→ integrand f(x) is **complicated**

Solution: numerical approximations

Example:

Use squares to simplify

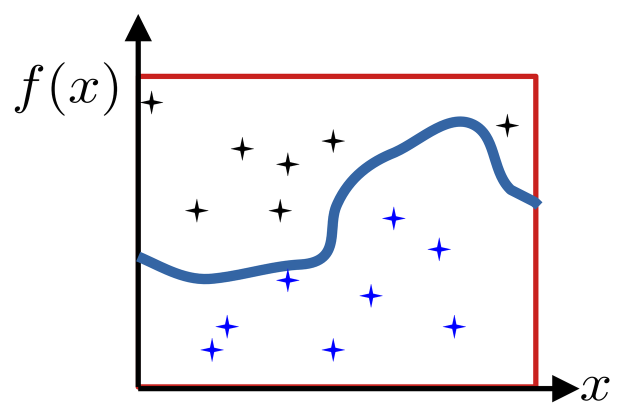


4. A numeric method: Monte Carlo

To estimate the integral we take N random of points (x_i,y_i) in the square and count how often $f(x_i) < y_i$

Wikipedia

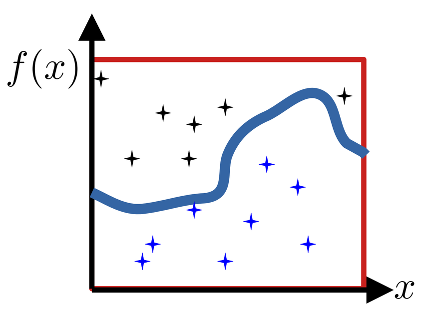
$$I \approx \hat{I} = \frac{\#\ \text{Hits}}{\#\ \text{Trials}} = \frac{1}{N}\sum_i^N f(x_i)$$



BUT: Estimate only good with **many** points, takes long even on supercomputers

Credit: eshopper.co

5. How to improve the Monte Carlo method?



Why do we need so many points?
Standard Monte Carlo is uniform in the area,
if f(x) is complicated it misses narrow peaks
until points are very dense!

Re-distribute points helps:
→ throw points where the function is large
→ needs less points for a good estimate of the integral

$$x_i \to z(x_i) = z_i \qquad I \approx \hat{I} = \frac{1}{N}\sum_i^N f(z_i)$$

Equivalent to substitution:

$$I = \int f(x)\, \mathrm{d}x = \int \left(\frac{\mathrm{d}x(z)}{\mathrm{d}z}\right) f(x(z))\, \mathrm{d}z \equiv \int \tilde{f}(z)\, \mathrm{d}z$$

But how do we know where we need the points?

6. Use iterative adaption techniques

1) Start with uniformly random points
2) Learn where f(x) was large
3) Re-distribute next set of random points
4) Repeat!

Various techniques are available, we want to use the advantage of **machine learning** to tackle the most difficult problems!

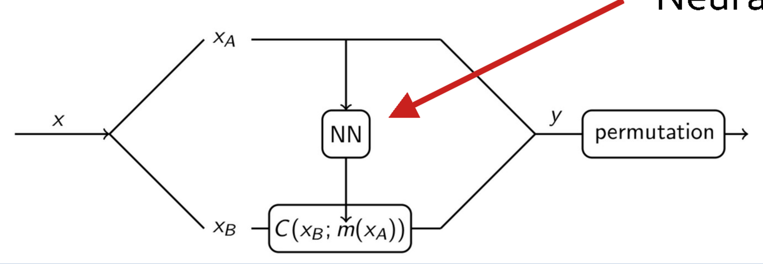
For experts: Coupling Layer-based Normalizing Flow (IFlow)

Build series of easy Mappings (bijections):

$$\vec{x}_K = c_K(c_{K-1}(\cdots c_2(c_1(\vec{x}))))$$

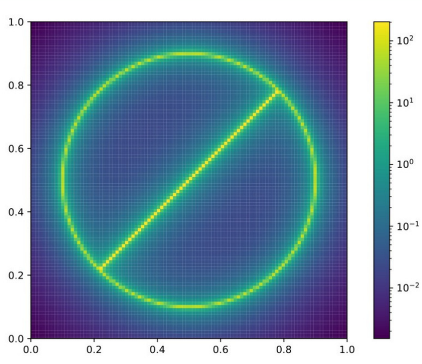
Structure of a single coupling layer:

Neural Network with millions of parameter!



$$x'_A = x_A, \qquad A \in [1, d]\,,$$
$$x'_B = C(x_B; m(\vec{x}_A)), \qquad B \in [d+1, D]\,.$$

7. Different adaption techniques in practice

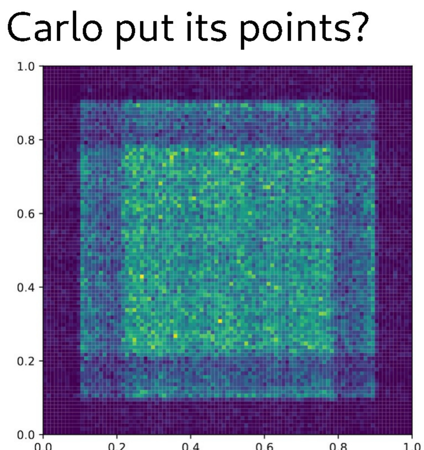


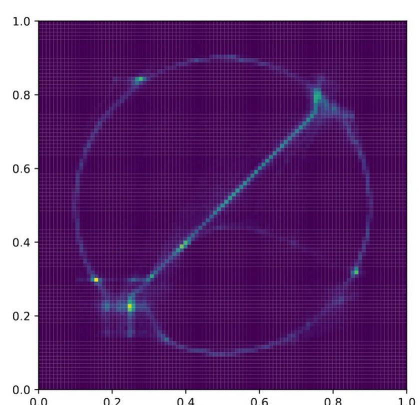
"Stop-sign" function: complicated 2 dimensional structure

$$f(x,y) = \frac{1}{2\pi^2} \cdot \frac{\Delta r}{\left(\sqrt{(x-x_0)^2+(y-y_0)^2} - r_0\right)^2 + (\Delta r)^2} \cdot \frac{1}{\sqrt{(x-x_0)^2+(y-y_0)^2}}$$
$$+ \frac{1}{2\pi r_0} \cdot \frac{\Delta r}{((y-y_0)-(x-x_0))^2 + (\Delta r)^2} \cdot \Theta\left(r_0 - \sqrt{(x-x_0)^2+(y-y_0)^2}\right)$$

Where does Monte Carlo put its points?

VEGAS
(approximation through squares, a widely used technique)

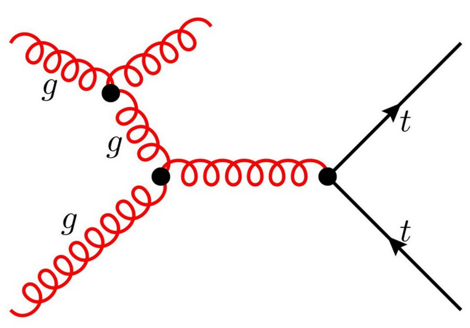




IFlow
machine learning method

8. Application to High Energy Physics

Production of top-quark pairs
(the heaviest known particle!)



10 Dimensions are hard to visualize
→ 2D projections