REPRESENTATION WORKSHOP

SUMMARY

Savannah Thais
Learning to Discover
04/28/2022
Thanks to our wonderful speakers!
Physics and ML are concerned with characterizing the true probability distributions of nature, how do we represent truth, data, and models to best enable learning these distributions?
Geometric Deep Learning
Representation Priors

The Curse of Dimensionality

Geometric Priors

Symmetry Prior

Scale Separation Prior
Graph Neural Networks

Adjacency matrix $n \times n$

Feature matrix $n \times d$

arbitrary ordering of nodes

graph function $f(X, A)$

node function $F(X, A)$

$n!$ permutations
Graph Neural Networks

Graph function \( f(X, A) \)

Node function \( F(X, A) \)

Arbitrary ordering of nodes

Multiset of neighbour features

\( X_{N_i} = \{x_j \in N_i \} \)

Local function

\( \phi \left( \begin{array}{c} x_i \\ x_{N_i} \end{array} \right) \)

Permutation invariant
Graph Neural Networks

Graph function $f(\mathbf{X}, \mathbf{A})$

Node function $\mathbf{F}(\mathbf{X}, \mathbf{A})$

Arbitrary ordering of nodes

Multiset of neighbour features

Permutation-invariant aggregation operator, e.g., sum

$f(\mathbf{x}_i) = \phi(\mathbf{x}_i, \psi(\mathbf{x}_j)_{j \in \mathcal{N}_i})$

$\phi \left( \mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_j) \right)$

"convolutional"

$\phi \left( \mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j) \right)$

"message passing"

$\phi \left( \mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j) \psi(\mathbf{x}_j) \right)$

"attentional"
Extensions of GNNs

Transformers

$$\phi \left( x_i, \prod_{j=1}^{n} a(x_i, x_j, p_i, p_j) \psi(x_j) \right)$$

Graph Substructure Network

$$\phi \left( x_i, \prod_{j \in \mathcal{N}_i} \psi(x_i, x_j, p_i) \right)$$

Equivariant Graph Neural Networks

Graph $G = (V, E)$

Node features $X(G)$

Functions $F(X(\Omega))$

Graph Rewiring

Decouple input graph from information propagation graph (at the expense of link to WL)

- Neighbourhood sampling (GraphSAGE)$^1$
- Multi-hop filters (SIGN)$^1$
- Complete graph$^1$
- Topology diffusion (DIGL)$^1$
- Learnable graph (Dynamic Graph CNN)$^1$

Permutation group $\Sigma_n$

Permutation matrix $P$

Equivariant message passing

Rotation $R$

$$F(PXR, PAP^T) = PF(X, A)R$$
GDL for Physics Tasks
Tracking

High luminosity: how? Cannot reduce distance between bunches any further. More protons/bunch!

Charged particles leave hits in the detector

Represent the data using a graph

Goal: classify the edges of the graph

High classification score => high probability that the edge is part of a track
Low classification score => low probability that the edge is part of a track
GNNs for Tracking

Particles leaving hits

Module map creation

Graph creation

Embed into learned latent space

Connect all spacepoints within radius r

All spacepoint connections joined into graph

Input graph

$N_t$

Transformer

$H_t$

Node Block

Edge Block

Interaction Network

$N_{edges}$

Edge scores

Transforms the D-dimensional space of each edge into a classification score for each edge

$N_{nodes}$

$N_{edges}$

$\Delta \phi$

$\Delta \varphi$

$\Delta \phi$
GNNs for Tracking

Particles leaving hits

Module map creation

Graph creation

Embed into learned latent space

Connect all spacepoints within radius r

All spacepoint connections joined into graph

Edge efficiency

Track efficiency

Input graph

$N_t$

Encoders $\rightarrow H_t$ $\rightarrow$ Edge Block $\rightarrow$ Node Block $\rightarrow H_{t+1}$ $\rightarrow$ Decoder $\rightarrow N_{edges}$

Transforms the D-dimensional space of each edge into a classification score for each edge

$N_{nodes}$

$N_{edges}$

$\Delta r$

$\Delta \phi$

$\Delta \eta$

$\Delta \xi$

Edge scores

Track reconstruction efficiency

$t\bar{t}$, $p_T>0.5$ GeV, $\mu=200$, $\sqrt{s}=14$ TeV

$N_{edges}$

$p_T$ (GeV)

Track efficiency

Edge efficiency
Reconstruction

Multilayered detectors

See the excellent talk by Jan Stark earlier today.

Simulation to reconstruction

Simulation model

- Simulation input particles
- Decay products
- Detector hits

Reconstruction model

- Clusters of hits
- Reconstructed particles
Clustering

- Segment the energy deposits (hits) according to the originator particles
- The hits are embedded in a complicated feature space (Cartesian position, energy, signal significance, timing, layer information, ...)
- Showers from different particles may overlap spatially
- Standard heuristic approaches based on seeding & collecting neighbors, typically iterative

Set-to-set problem

Each particle is described by a multi-class label, and is embedded in a complex, problem-dependent feature space.
Clustering

- Segment the energy deposits (hits) according to the originator particles
- The hits are embedded in a complicated feature space (Cartesian position, energy, signal significance, timing, layer information, ...)
- Showers from different particles may overlap spatially
- Standard heuristic approaches based on seeding & collecting neighbors, typically iterative

Object condensation

**Boundedness:** the number of truth particles usually cannot be larger than the number of inputs (typically it's much smaller).

\[
L_V = \frac{1}{N} \sum_{j=1}^{N} \sum_{k=1}^{K} (M_{jk} \hat{V}_k(x_j) + (1 - M_{jk}) \tilde{V}_k(x_j))
\]

Each input represents exactly one truth particle, with attractive/repulsive potentials in a learned space \(x_j\) between correct/incorrect assignments.
Particle Flow

Machine learned particle flow (MLPF)

Baseline PF, adapted from B. Mangano for CMS, 2013

“True” or generated particles

Particle interaction & detection

HCAL Clusters

ECAL Clusters

Tracks

Detector measurements

Graph neural network

Extract features

MLPF candidates

Compare via loss function

Machine-learned particle-flow reconstruction

PF candidates

charged hadrons

neutral hadron

muon

photon

charged hadrons

neutral hadron

muon

photon
Particle Flow

Event as input set

\[ X = \{ x_i \} \]

Event as graph

\[ X = \{ x_i \}, A = A_{ij} \]

Transformed inputs

\[ H = \{ h_i \} \]

Graph building

Graph building

\[ \mathcal{F}(X | w) = A \]

Message passing

\[ \mathcal{G}(X, A | w) = H \]

Target set \( Y = \{ y_j \} \)

Output set \( Y' = \{ y'_j \} \)

Elementwise loss \( L(y_j, y'_j) \)

classification & regression

Decoding

\[ \mathcal{D}(x_j, h_j | w) = y'_j \]

\[ x_i = \{ \text{type}, p_T, E_{ECAL}, E_{HCAL}, \eta, \phi, \eta_{outer}, \phi_{outer}, q, \ldots \}, \quad \text{type} \in \{ \text{track, cluster} \} \]

\[ y_j = \{ \text{PID}, p_T, E, \eta, \phi, q, \ldots \}, \quad \text{PID} \in \{ \text{none, charged hadron, neutral hadron, } \gamma, e^\pm, \gamma, e^\pm \} \]

\[ h_i \in \mathbb{R}^{256} \]

Trainable neural networks: \( \mathcal{F}, \mathcal{G}, \mathcal{D} \)

- track, - calorimeter cluster, - encoded element
- target (predicted) particle, - no target (predicted) particle
Applicability

In a realistic environment

Computational scalability

Interpretability

- What inputs are relevant for a particular model output?
- Compute layerwise relevance scores \( R \)
- Aggregate along the graph
Adding Physics In
Deep Learning Accelerates Scientific Simulations up to Two Billion Times

1. Neural networks are good at interpolation, bad at extrapolation
2. Learned physics models often don’t learn anything close to the underlying physical equations
3. There’s no way we can build a dataset that covers the input space of a general-purpose simulator
Physics Inspired Priors/Inductive Biases

A simple inductive bias: Inertial dynamics

\[ x^{t+1} = \mathbf{NN}(x^t, v^t) \]

Static prior

\[ x^{t+1} = x^t + \mathbf{NN}(x^t, v^t) \]

Inertial prior

\[ x^{t+1} = x^t + \Delta t \cdot v^t + \mathbf{NN}(x^t, v^t) \]

Has to learn to predict static motion

Trivial to predict static motion

Has to learn to predict inertial motion

Trivial to predict inertial motion!
Physics Inspired Priors/Inductive Biases
Learning Physical Simulators

same model, same hyperparameters can simulate many systems

- And many ways to improve:
  - Adding noise, ask update function to remove, improves stability
  - Remeshing (scale prior) improves precision/speed
  - Adaptive remeshing improves precision/compute utilization
Learning System Design

**Inner loop**: forward model rollout

**Outer loop**: design optimization process

Gradient-based with learned models

Can we use a GNN based model pre-trained on physical dynamics for inverse design?

Constraint-Based GNNs

Invalid state: bounced too far

Invalid state: Two ball overlap

Run gradient descent on $f_C$ to find $\hat{X}_{t+1}$
Constraint-Based GNNs

Invalid state:
Two ball overlap

Invalid state:
bounced too far

Run gradient descent on $f_C$ to find $\hat{X}_{t+1}$

At test time optimize $f_C(X_{≤t}, X_{t+1}) + f_{\text{obstacle}}(X_{t+1})$

a learned constraint

a user-defined constraint
(e.g. new obstacle)

No collisions were ever observed at training time!
Getting Physics Back Out
Symbolic Regression

Distilling Free-Form Natural Laws from Experimental Data

Michael Schmidt\textsuperscript{1} and Hod Lipson\textsuperscript{2,3,}\textsuperscript{*}

For centuries, scientists have attempted to identify and document analytical laws that underlie physical phenomena in nature. Despite the prevalence of computing power, the process of finding natural laws and their corresponding equations has resisted automation. A key challenge to finding analytic relations automatically is defining algorithmically what makes a correlation in observed data important and insightful. We propose a principle for the identification of nontriviality. We demonstrated this approach by automatically searching motion-tracking data captured from various physical systems, ranging from simple harmonic oscillators to chaotic double-pendula. Without any prior knowledge about physics, kinematics, or geometry, the algorithm discovered Hamiltonians, Lagrangians, and other laws of geometric and momentum conservation. The discovery rate accelerated as laws found for simpler systems were used to bootstrap explanations for more complex systems, gradually uncovering the “alphabet” used to describe those systems.
Symbolic Regression

Repeats process iteratively to yield set of candidate equations
Learning Astrophysics

1. Our inputs are the positions of the bodies

2. They are converted into pairwise distances

3. Our model tries to guess a mass for each body

4. It then also guesses a force, that is a function of distance and masses

5. Using Newton’s laws of motion ($\sum \overrightarrow{F} = M \overrightarrow{a}$) it converts the forces into accelerations

6. Finally, it compares this predicted acceleration, with the true acceleration from the data

$$\text{Minimize} \quad \left| \overrightarrow{a}^{\text{pred}} - \overrightarrow{a}^{\text{true}} \right|^2$$
Inductive Biases

- Translational symmetry
- Rotational symmetry
- Newton’s second law
  \[ \sum \vec{F} = M \vec{a} \]
- Newton’s third law
  \[ \vec{F}_{ij} = - \vec{F}_{ji} \]
- Choice of reference frame, units, etc.
Learning Astrophysics

Predicted masses

Graph Network
Extracting the Physics

- Apply symbolic regression with a constraint to balance accuracy and equation complexity
- Can substitute learned equation for the force guess to improve the simulator
Discussion
Highlights
How Can We Make This Usable?

• Graph construction is critical for effective learning and meeting computing constraints
  • Are there ways to do effective segmentation or hierarchical graphs
  • How do we balance information sharing with size
• Incorporating inductive biases can improve stability, generalizability, and model efficiency
  • Equivariant GNNs could reduce training resources, generalize
  • Attention mechanisms weight physically important information
  • Are there other types of (intermediate) functions we could model
  • Constrained problems may be harder to solve in some cases
• We need to ensure the problem is truly physical
  • In high pileup overlapping tracks can share hits and even segments
  • How do we handle noise, missing information, detector effects
• Hardware-based acceleration is likely necessary
Does This Help Us Do Physics?

- Graphs seem to be the most effective representation of particle physics experiment data
  - Reduced information loss, allows hierarchical representations
  - But are we fully exploiting this
- Symbolic regression can help understand if a model is learning the true physics of the universe
  - Potentially help us refine physical laws
- Interpretability of GNNs is extremely under-studied in physics
  - Attention mechanisms and relevance propagation are proxies but are not precise
  - Other methods like black box methods, disentangle representation learning have not been studied
- Central debate in ML for physics: do we care about getting the physics back (data-driven science)
Are There New Directions to Explore?

- Transformers are effective on many problem types
  - positional encoding/graph substructure models
- Study graph rewiring/nonphysical graphs/message passing only edges/information aggregating nodes
- Incorporate additional priors/inductive biases
  - Loss function constraints (number of decay products, consistency with true tracks)
  - Constraint-based GNNs
  - Graph level conservation laws
- Apply these methods to more physics tasks
  - Underexplored for simulation
  - Full hierarchical reconstruction
  - Experimental design optimization (trigger operations, detector/accelerator design)
- Represent existing problems in new ways
  - Tracking as denoising VAE or mesh generation
A Note on Datasets

• The TrackML dataset is not realistic for several reasons
• A new open data detector is nearly ready
• Can we create other benchmark/open datasets
  • Particularly that are designed for GDL
  • Even benchmark GNN models
• Always the concern of mismatch between data and simulation
  • Are there ways we can train directly on data
Thank you to all participants!