Graph Neural Networks for Neutrino Classification

Nicholas Choma and Joan Bruna

July 18, 2018

Agenda

IceCube Neutrino Observatory

- Project goal is to detect high-energy extraterrestrial neutrinos, originating from e.g. black holes and supernovae
- Neutrinos interact only through gravity and weak subatomic force, making them excellent intergalactic messengers
- Detection is made difficult due to overwhelming cosmic ray background noise

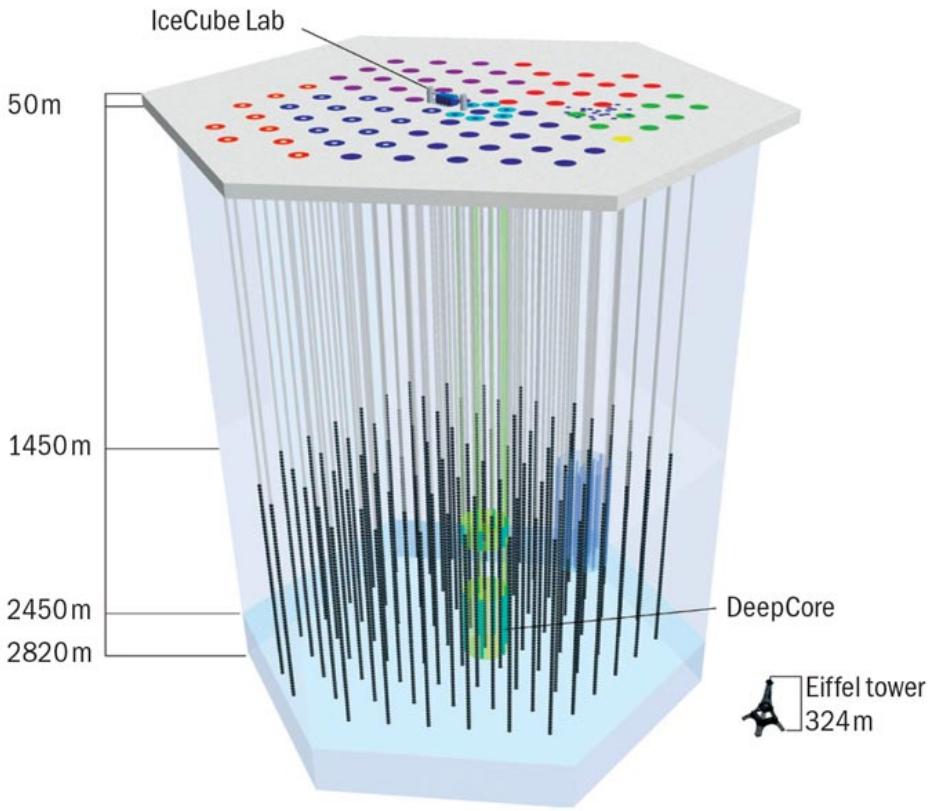


Figure: IceCube sensor array

IceCube Dataset

- Cubic km, irregular hexagonal grid of 5160 sensors for detecting neutrinos
- Each detection event involves only a subset of all sensors
- Data is generated by simulators using first principles from physics
- About 4x more background events than signal. Samples weighted based upon yearly frequency

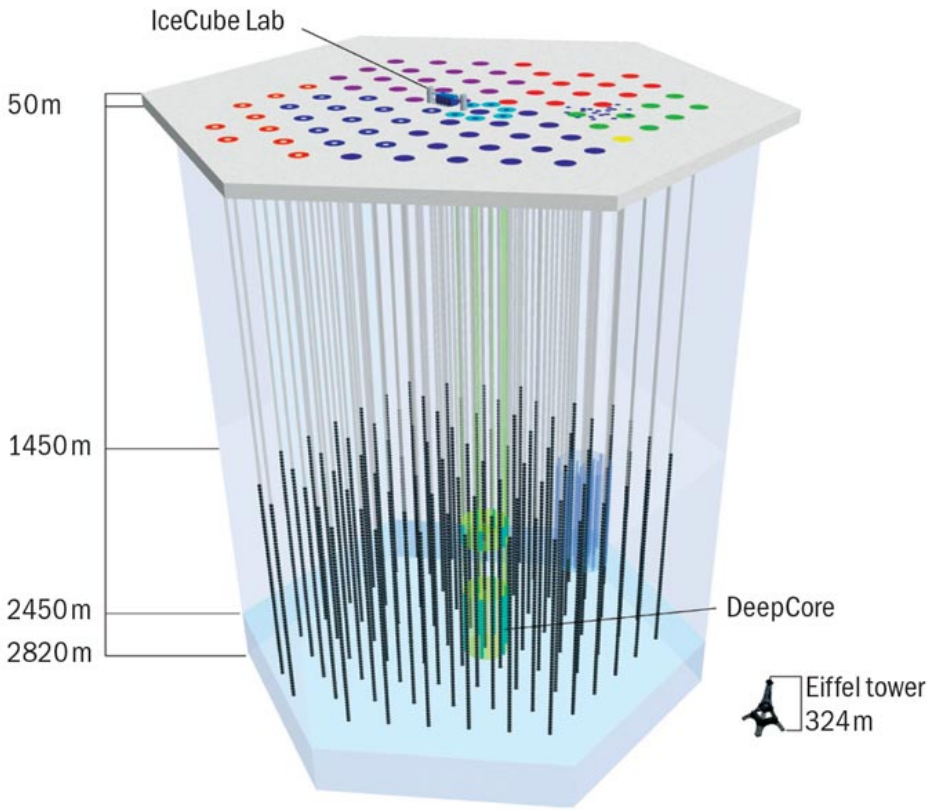


Figure: IceCube sensor array

IceCube Physics Baseline

- Sequence of cuts based upon energy loss stochasticity and energy vs. zenith angle used to obtain baseline
- Current baseline keeps:
  - 1 weighted signal event per year
  - 1:1 signal-noise ratio (SNR)

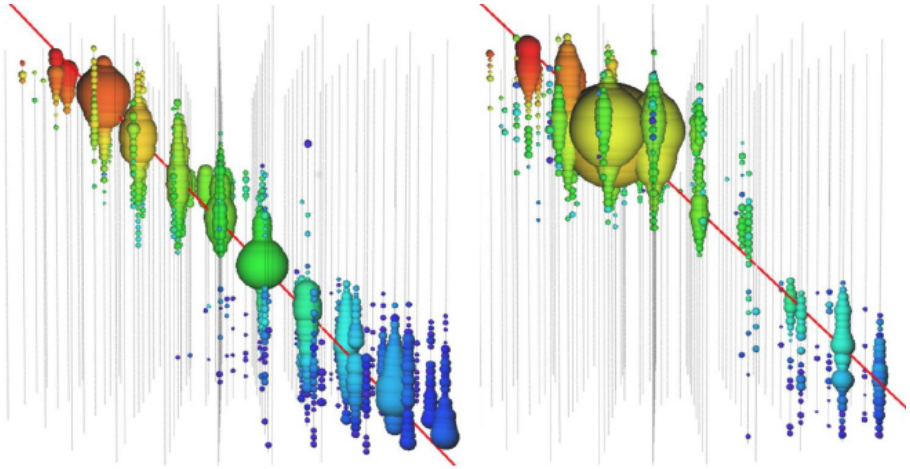


Figure: Background (left) and signal (right) events

Geometric Deep Learning

- Graph- and manifold-structured data
  - Point clouds
  - Social networks
  - 3D shapes
  - Molecules
- Graph neural network models:
  - Learned information diffusion processes
  - Convolution based upon spectral filters
  - Graphs performing local neighborhood operations
- See [Bronstein *et al.*, 2016] for Geometric Deep Learning survey

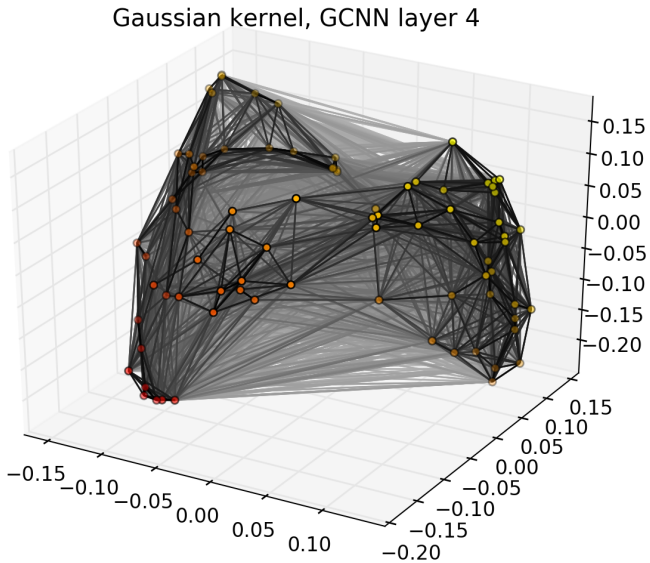


Figure: Point cloud embedded in 3D. Graph constructed using a Gaussian kernel.

Graph Neural Networks (GNN) and IceCube

- Graph constructed with DOMs as vertices, edges learned
- Computation restricted to active DOMs only
- GNN model able to use IceCube structure to learn efficiently
- Translation invariance not required as in 3D Convolutional Neural Network (CNN)

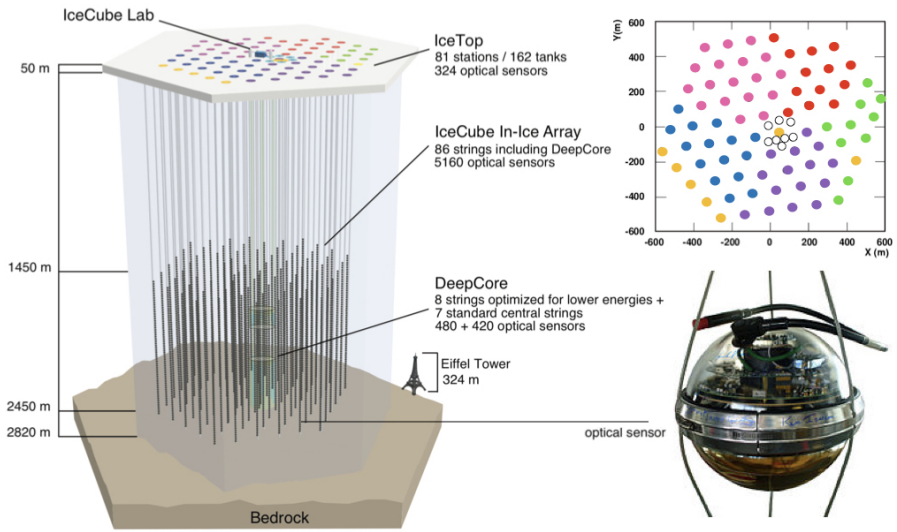


Figure: IceCube sensor array (left), overhead view (top right), and sensor (bottom right)

IceCube GNN Architecture

Task

Input: n x 6-tuple of (domx, domy, domz, first_charge, total_charge, first_time)
Output: Prediction $\in [0, 1]$

GNN Overview:

1. Compute adjacency matrix of pairwise distances between DOMs active in a given event
2. Apply graph convolution layers
3. Pool graph nodes and apply final network output layer on all features

Step 1: Compute adjacency matrix

- Pairwise distances are computed using a Gaussian kernel function
- Only spatial coordinates are used (domx, domy, domz)
- σ is a scalar, learned parameter

Gaussian kernel

$$d_{ij} = \exp(-\tfrac{1}{2}||x_i - x_j||^2/\sigma^2)$$

- A softmax function is applied to each row to get adjacency matrix A

Softmax

$$A_{ij} = \frac{\exp(d_{ij})}{\sum_k \exp(d_{ik})}$$

Step 2: Apply Graph Convolution Layers

- Model uses eight layers of graph convolution with 64 features each
- Each layer is divided into two 32-feature graph convolutions, one which has a pointwise nonlinearity (ReLU) applied
  - $\texttt{ReLU}(x) = \max(0, x)$
- Linear and nonlinear outputs are concatenated - denoted by $||$ symbol - to produce the layer output
- t indexes the graph convolution layer, d is the number of features

Graph Convolution Layer

Input: $X^{(t)} \in \mathbf{R}^{nxd^{(t)}}$
Output: $X^{(t+1)} \in \mathbf{R}^{nxd^{(t+1)}}$

$$X_{nlin} = \texttt{ReLU}(\texttt{GConv}(X^{(t)}))$$
$$X_{lin} = \texttt{GConv}(X^{(t)})$$
$$X^{(t+1)} = X_{nlin}||X_{nlin}$$

Step 2 (cont.): GConv, Operators and Transformation

- Signal is spread over graph via two operators
  - A, graph adjacency matrix
  - I, identity matrix
- Outputs of operators acting on graph signal are concatenated
- Linearly transformed by learned $\theta_w \in \mathbf{R}^{2d^{(t)} x \frac{d^{(t+1)}}{2}}$, $\theta_b \in \mathbf{R}^{\frac{d^{(t+1)}}{2}}$

GConv update

$$\texttt{Spread}(X^{(t)}) = AX^{(t)}||IX^{(t)}$$

$$\texttt{GConv}(X^{(t)}) = \texttt{Spread}(X^{(t)})\theta_w^{(t)} + \theta_b^{(t)}$$

Step 3: Final Readout Layer

- Final layer sums over all points to produce $X^{(end)} \in \mathbf{R}^d$
- Features are then linearly transformed by $\theta_w^{(end)} \in \mathbf{R}^d, \theta_b^{(end)} \in \mathbf{R}$
- A prediction $y_{pred} \in [0, 1]$ is output using a sigmoid function
  - $\texttt{Sigmoid}(x) = \frac{1}{1+e^{-x}}$

Readout

$$X_k^{(end)} = \sum_j X_{jk}^{(end-1)}$$

$$y_{pred} = \texttt{Sigmoid}(X^{(end)T}\theta_w^{(end)} + \theta_b^{(end)})$$

Results

- Final deep learning selection on the test set gives
  - 5.77 neutrinos per year
  - 1.94 cosmic muons per year

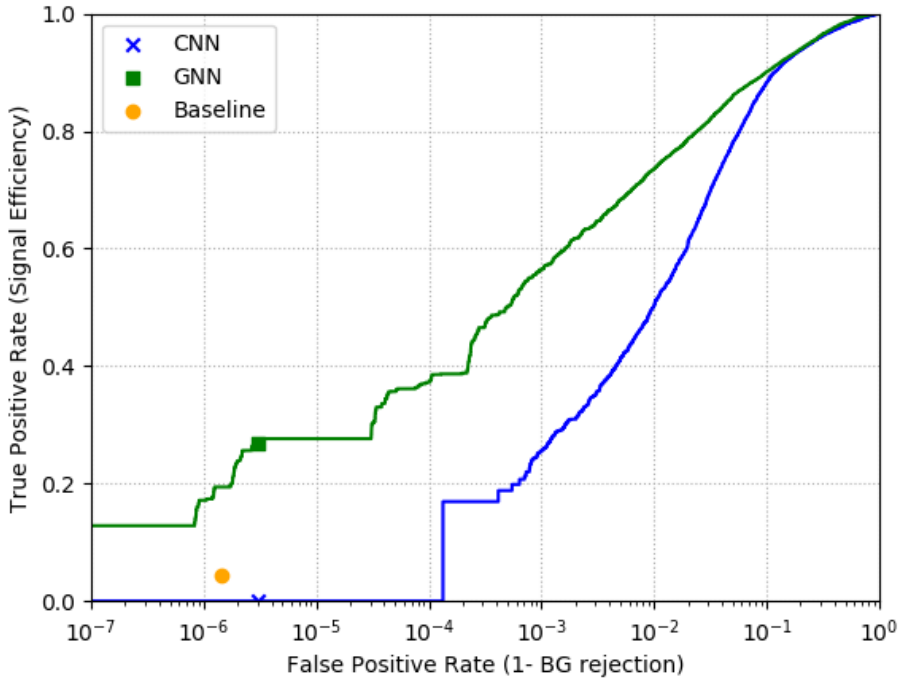


Figure: Receiver operating characteristics (ROC) curve

Future Directions, Performance

Models currently require ≈ 2 days to train. Future directions will address this with several ideas:

1. Parallelization using multiple compute nodes
2. Kernel adjacency matrix sparsity
3. O(nlogn) implementation using hierarchical clustering

Parallelization using multiple compute nodes

On each compute node, process subset of minibatch and compute gradients of parameters. Then combine all gradients for minibatch and take gradient step.

Benefits:

- Run larger minibatches (> 5 samples currently) for faster training
- Faster hyperparameter, architecture experimentation

Challenges:

- Long idle time for any compute node processing a small event
- Larger minibatches may affect model convergence
- No asymptotic speedup

Kernel adjacency matrix sparsity

Perform KNN-search for each graph node, or remove weighted edges from graph below cutoff threshold, to create sparse graph adjacency matrix.

Benefits:

- Reduces compute time (wall and asymptotic) once sparse adjacency matrix is built

Challenges:

- Still $O(n^2)$ cost in building sparse adjacency matrix
- Graph may no longer be connected

$O(nlogn)$ implementation using hierarchical clustering

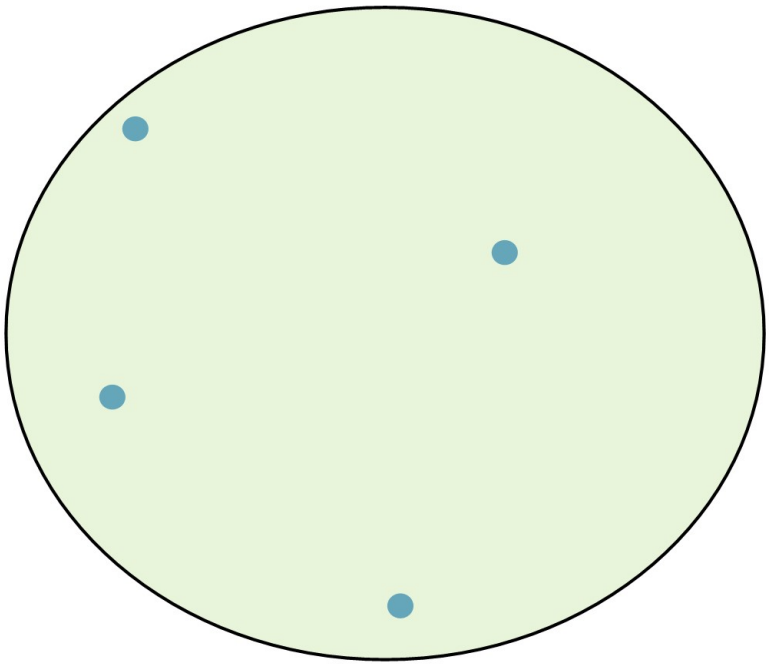
Idea

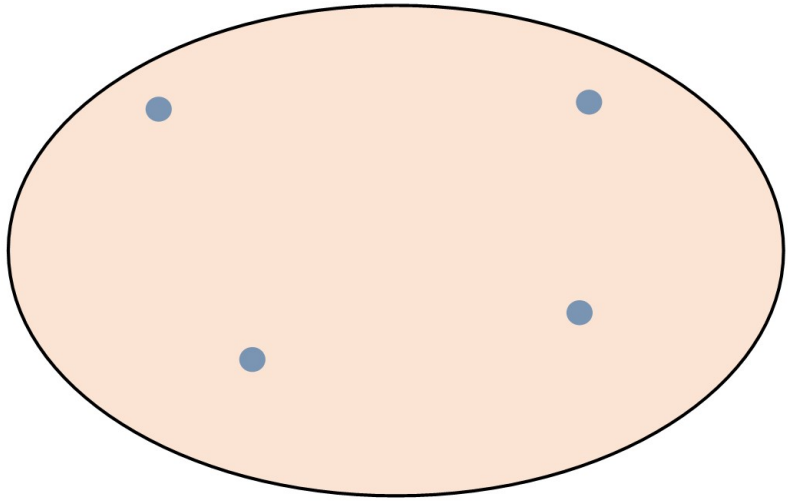
Create sparse graph which guarantees connectivity between distant vertices.

1. Recursively divide the graph into two subsets of vertices, building a binary tree with a unique subset of at most k vertices at each tree leaf
2. Internal tree nodes become new vertices in the graph and connect to all descendants, guaranteeing $O(n \log n)$ edges in constructed graph
3. Vertices within a tree leaf are densely connected

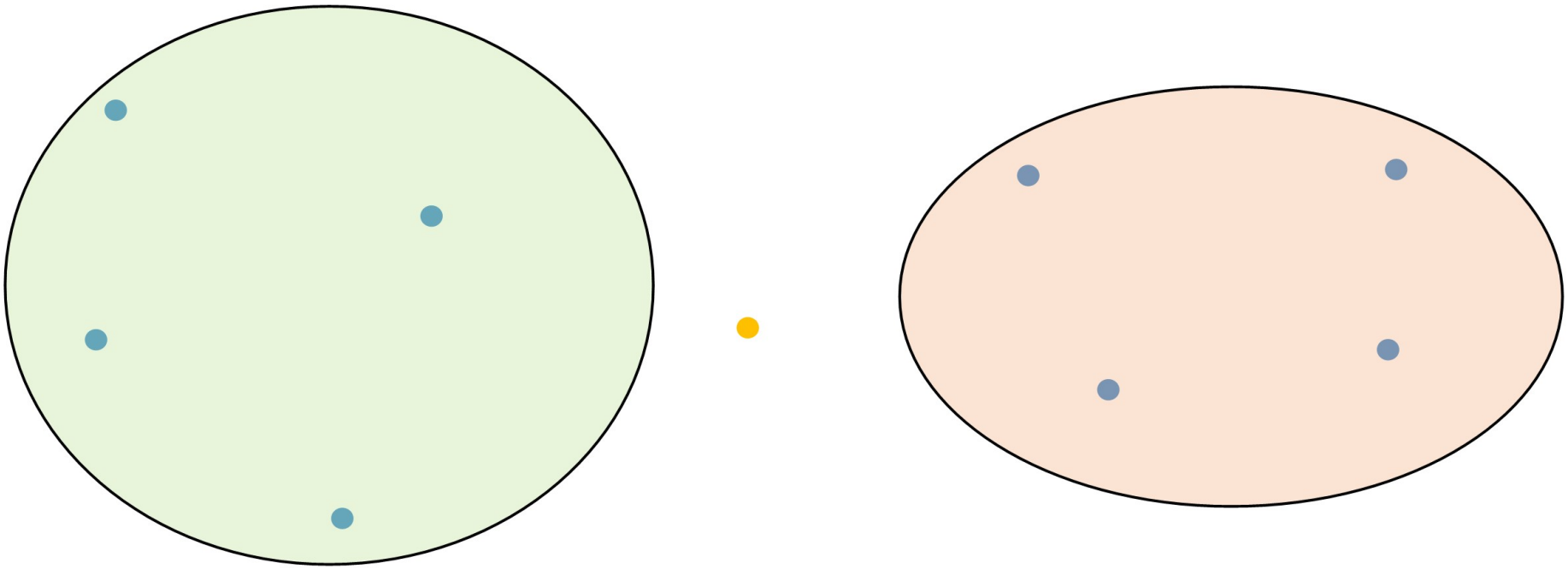
$O(nlogn)$ Graph construction example

$O(nlogn)$ Graph construction example

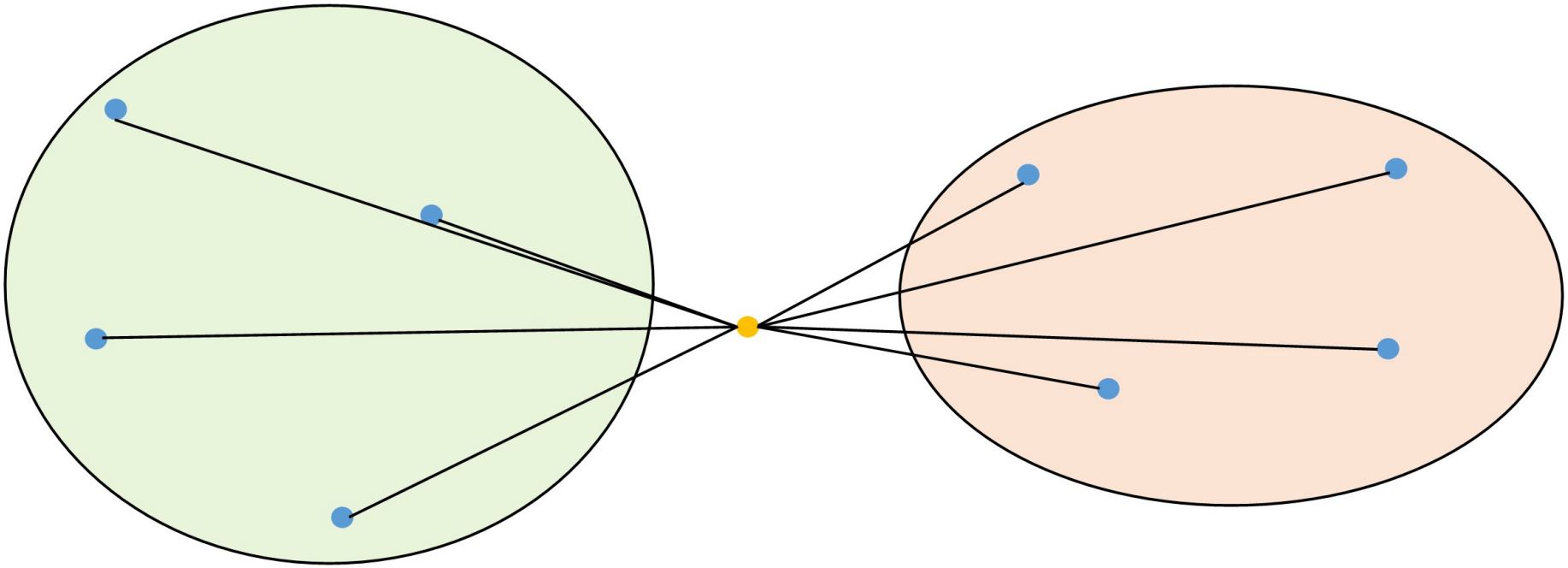




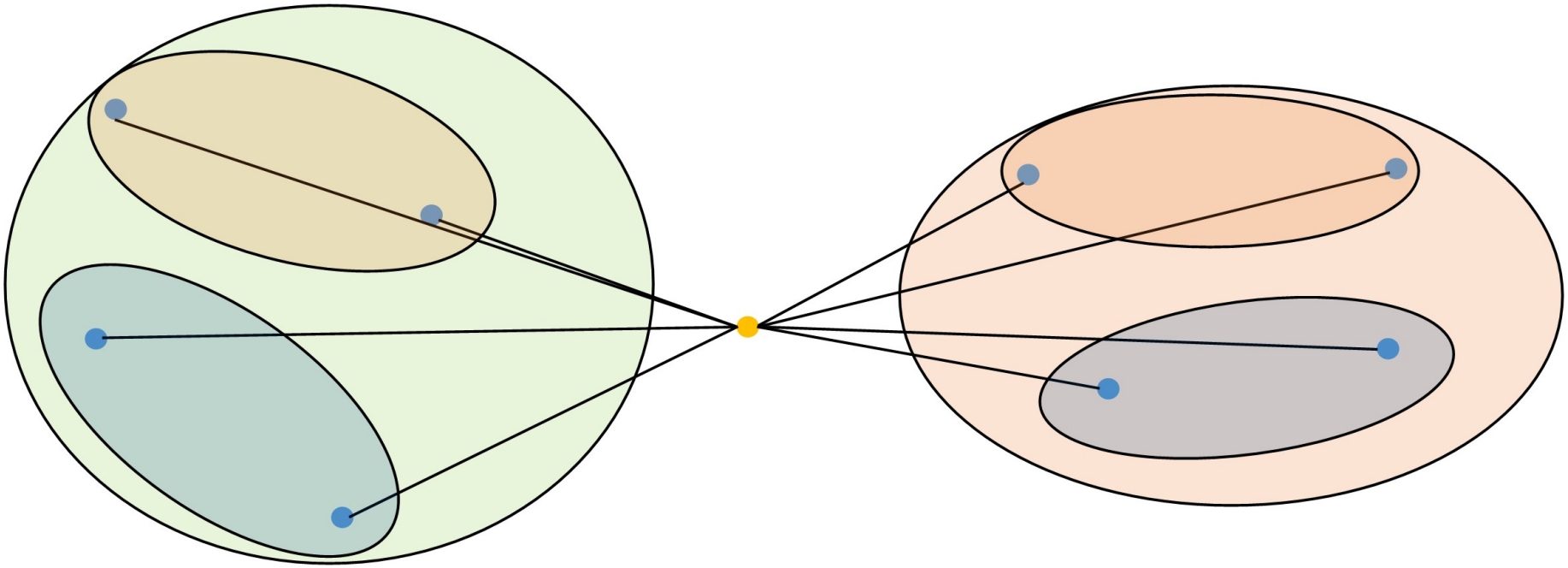
$O(nlogn)$ Graph construction example



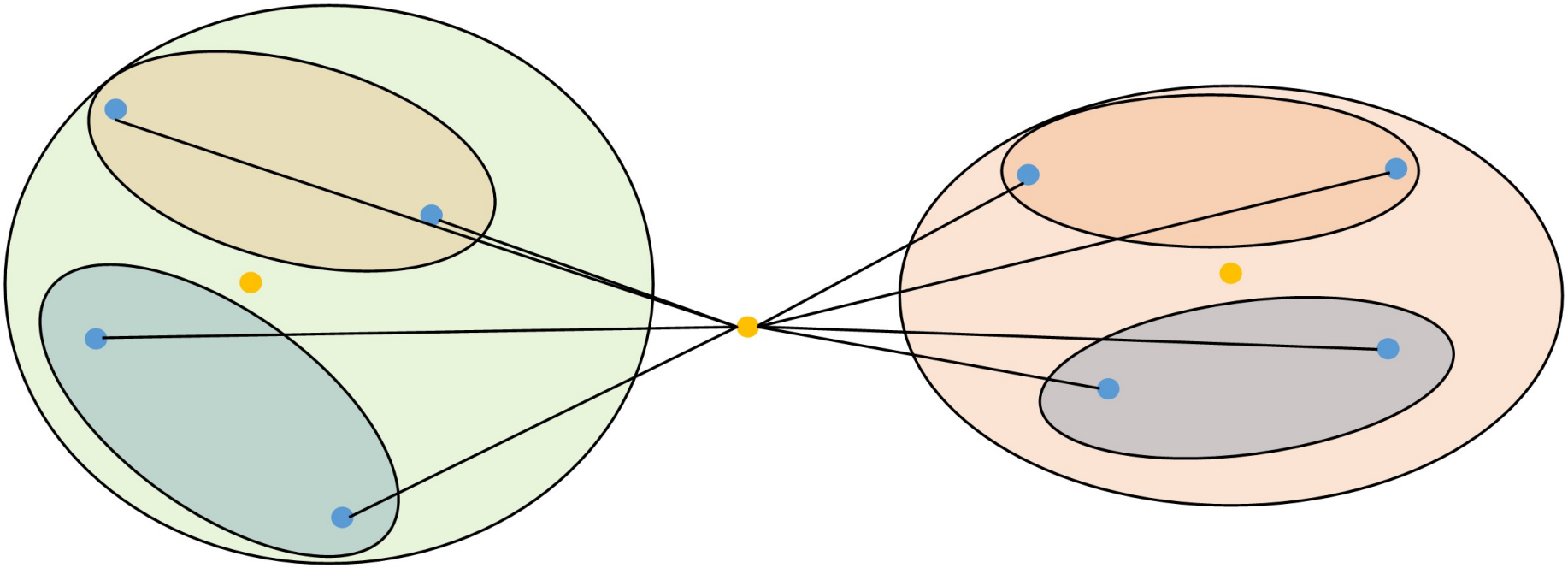
$O(nlogn)$ Graph construction example



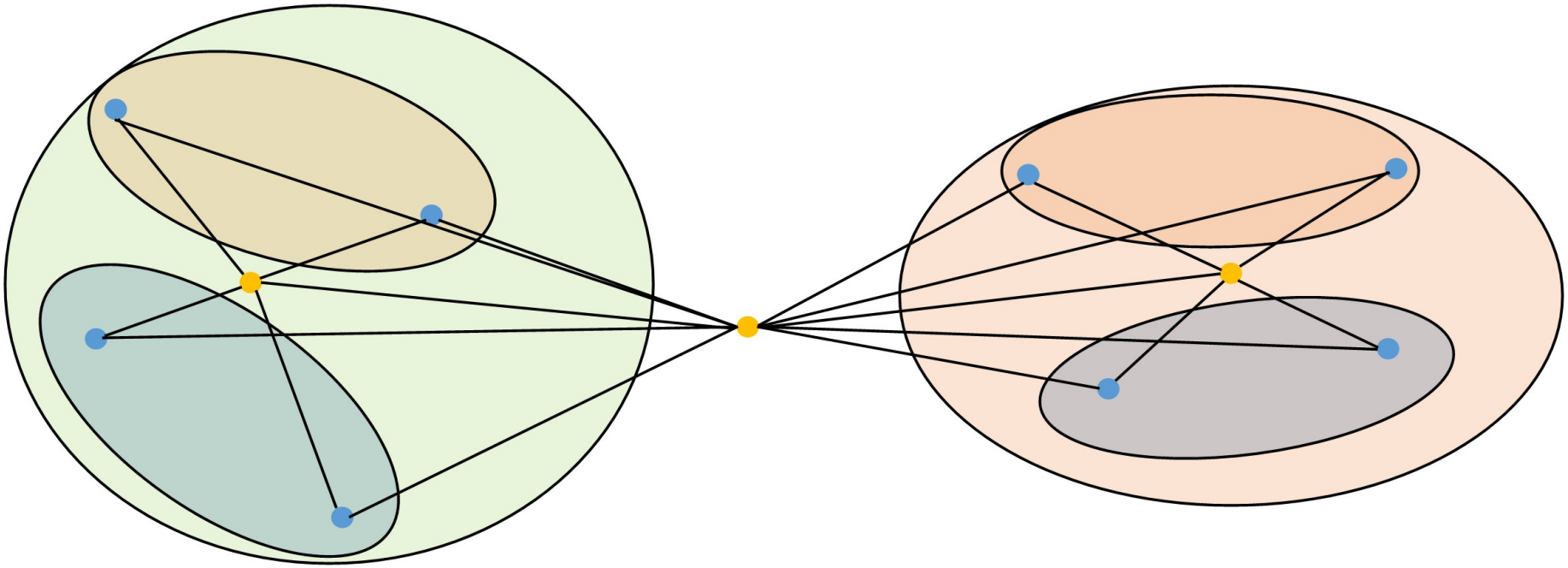
$O(nlogn)$ Graph construction example



$O(nlogn)$ Graph construction example



$O(nlogn)$ Graph construction example



$O(nlogn)$ implementation using hierarchical clustering

Benefits:

- Improved asymptotic time complexity for large graphs
- Wall time cost improved for graphs with ≈ 1000 nodes, as in IceCube
- No risk of having isolated subsets of the graph as in sparse case

Challenges:

- Preliminary results show promise, but worse than best GNN
- Need to use policy gradient to learn splitting procedure since discrete splits not differentiable
- Training on batches not straightforward

Future Directions, Next Tasks

Several additional interesting problems may be greatly improved upon through the use or continued development of GNNs:

1. Beyond Standard Model (BSM) jet classification
2. Quantum chemistry property estimation
3. Particle tracking, jet physics

Jet Classification

Goal: Classify a jet event as *interesting* - e.g., as resulting from the decay products of a Higgs Boson

Task

Input: An event, which consists of n particles in the point cloud. Each particle consists of 6 features derived from its 4-momenta
Output: Prediction $\in [0, 1]$

- Nearly identical format as the IceCube dataset
- Classification accuracy improved by using a custom kernel inspired by jet physics for creating the pairwise adjacency matrix

Quantum Chemistry

Goal: Predict quantum properties of organic molecules, resulting in a machine learning model which reduces compute time by orders of magnitude over traditional methods

Task

Input: A molecule, consisting of n atoms and their type (e.g. N, O, F), and the bonds between them
Output: Real value for a quantum chemistry property of interest

- Gilmer *et al.*, 2017 use message passing neural networks to successfully predict 11 of 13 properties considered
- Current work focuses on incorporating line graphs into GNN architectures for modeling of higher-order interactions

Particle Tracking

Goal: Reconstruct particle tracks from 3D points captured in particle detectors

Task

Input: An event, consisting of n detector *hits*, their (x, y, z) positions, and their charges

Output: n predictions $\in \{0, 1, \ldots, k\}$, where k is unique for each event

- Challenges
  - $n \approx 100,000$
  - $k \approx 10,000$, but unknown *a priori*
- Combines both hierarchical graph pooling and the $O(n \log n)$ model