Learned Representations from Lower-Order Interactions for Efficient Clustering

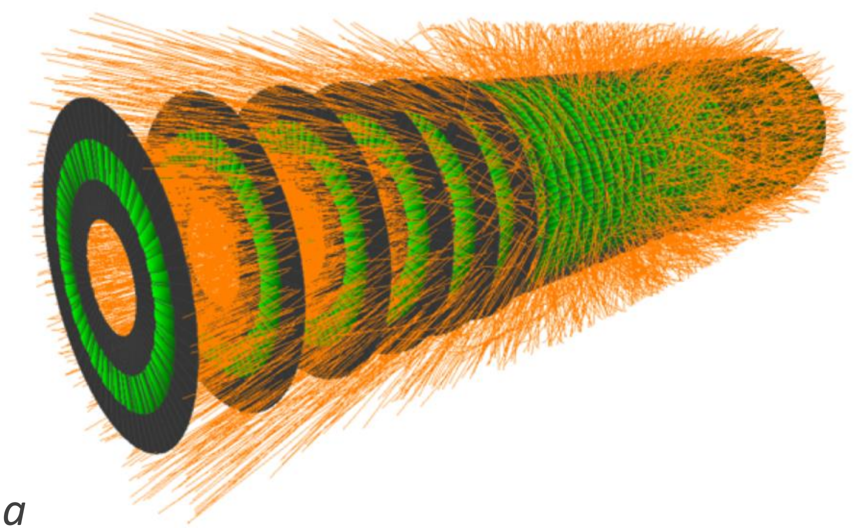
NICHOLAS CHOMA

EXA.TRKX, LAWRENCE BERKELEY NATIONAL LAB

CONNECTING THE DOTS 2020

Particle Tracking, TrackML

Given a set of $\approx 10^5$ hits created by $\approx 10^4$ particle tracks, cluster hits such that each cluster is associated with one track.



Input:

- N hits (x, y, z and detector ID)
- Detector cell pattern for each hit

Output:

- k clusters of the N hits

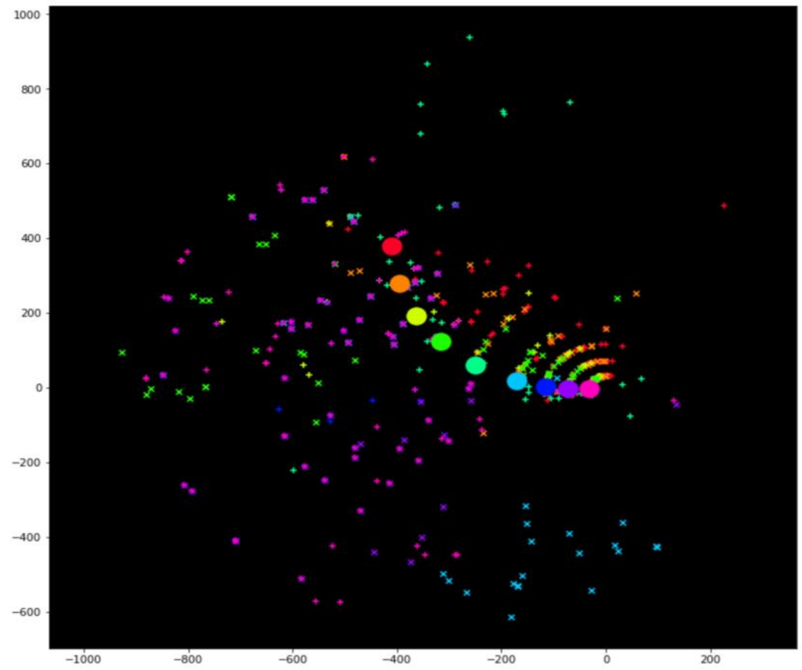
Challenges:

- Variable number of tracks which is not *a priori* known
- Inference must be efficient

Baseline Methods, Motivation

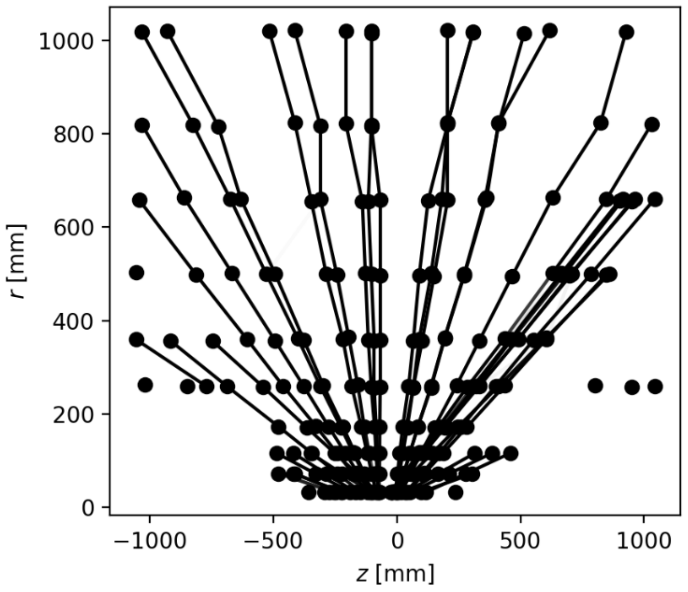
Top solutions to kaggle's TrackML competition generally operate in three stages:

1. Identify candidate **pairs of hits on adjacent detector layers**
2. Construct tracks based on candidate pairs
3. Refine track selections
   - Reject tracks below certain threshold
   - Extend tracks with any appropriate leftover hits



GNN Baseline, Motivation

- Farrell et al. proposed a GNN-based deep learning approach which connects hits through learned edge labels
  1. Construct a graph which connects promising **hit pairs in adjacent layers** with edges
  2. Run hits and graph through a GNN to classify edge labels
  3. Traverse graph through high-scoring edges to build tracks



[Farrell et al., 2018]

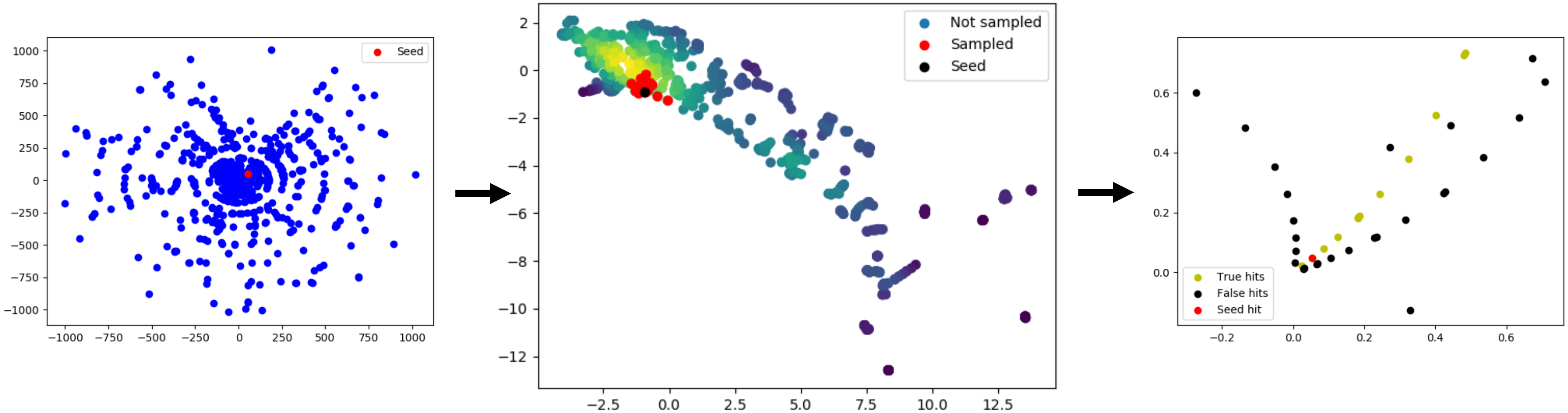
Learned Embeddings

- Leading solutions begin by a process of **doublet building**, relying on implementing a series of heuristic physics-based cuts
- **Key idea**: build doublets by learning a representation of hits, where the new space has a Euclidean metric, hits belonging to the same track are close together, and otherwise far apart
- Learn an embedding model as follows:
  - Dataset $X = \left\{\left(x_1^{(1)}, x_2^{(2)}\right), \ldots, \left(x_n^{(1)}, x_n^{(2)}\right)\right\}$, x_i in original space
  - Embedding model $\phi(x) \in \mathbb{R}^m$, where ϕ is an MLP and m is the embedding dimension
  - Supervise MLP with hinge loss l:

Let $d = \|\phi\left(x^{(1)}\right) - \phi\left(x^{(2)}\right)\|_2$ be the distance between $x^{(1)}$ and $x^{(2)}$. Then

$$l\left(x^{(1)}, x^{(2)}\right) = \begin{cases} d & \text{if } x^{(1)}, x^{(2)} \text{ belong to same track} \\ \max(0, 1-d) & \text{else} \end{cases}$$

Learned Embeddings

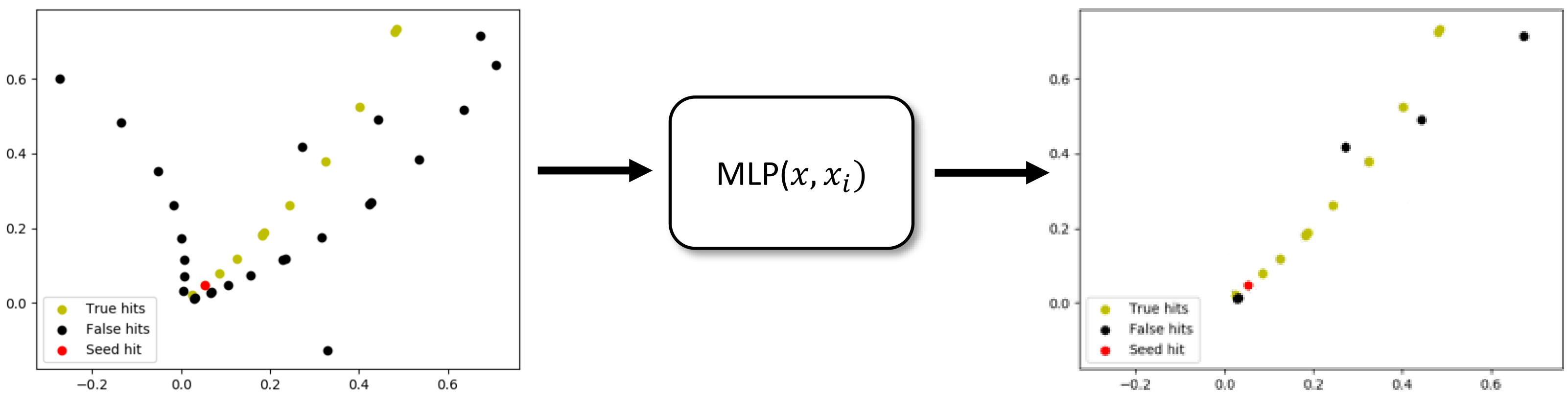


- Transform hits from the original space (left) to an embedded space
- Given a seed hit, query its neighborhood in the embedded space (center) using KD-trees
- Visualize the neighborhood in the original space (right)
  - Note that all of the hits in the seed hit's track are present

Doublet Refinement

- Graph neural networks are expensive and require large footprints in GPU memory
- Far more information content in doublet than in single hit
- **Key idea**: Filter doublets with small MLP that rejects hit pairs belonging to different tracks, greatly increasing the purity of the produced graph
- Learn a filter model as follows:
  - Dataset $X = \left\{\left(x_1^{(1)}, x_2^{(2)}\right), \ldots, \left(x_n^{(1)}, x_n^{(2)}\right)\right\}$, x_i in original space
  - Targets $Y = \{y_1, \ldots, y_n\}$, $y_i = 1$ if $x_i^{(1)}, x_i^{(2)}$ belong to the same track, else 0
  - Supervise MLP with binary cross entropy

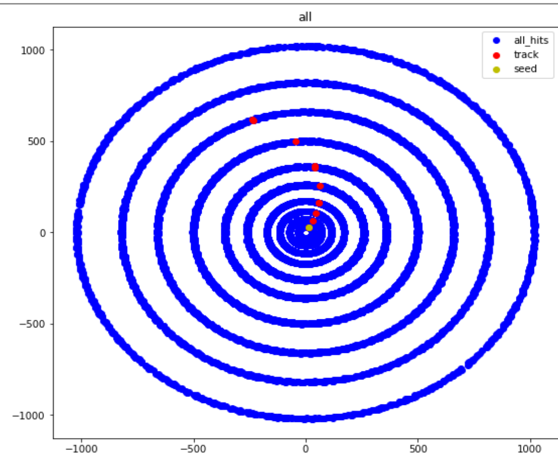
Doublet Refinement



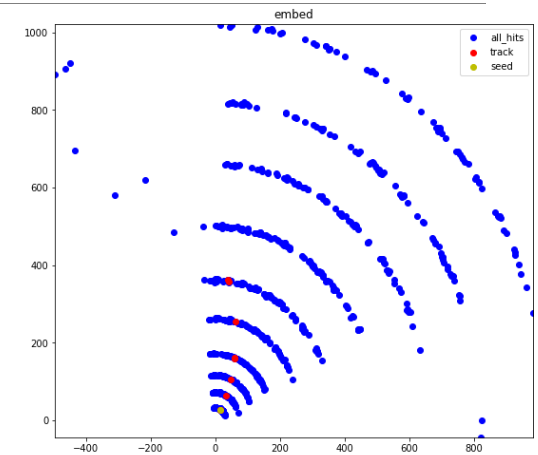
- Given seed x, sample neighbors $x_i \in N(x)$ from the embedded space
- Pass x concatenated with x_i to an MLP classifier, trained to reject hit pairs which belong to different tracks
- Remove pairs below threshold, refining neighborhood selection
  - Pairwise hit information improves precision by ≈17%, greatly reducing the number of pairs produced

Doublet Pipeline, example

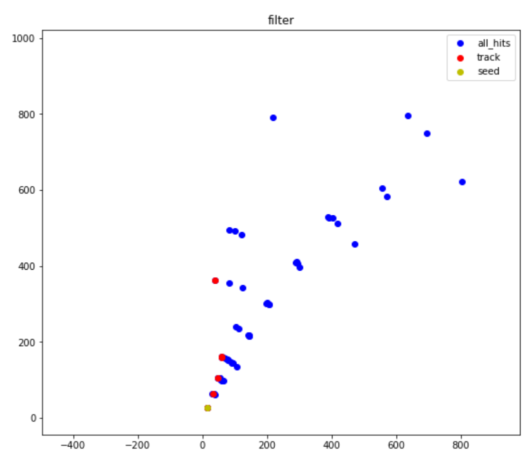
1. Seed doublet and true track



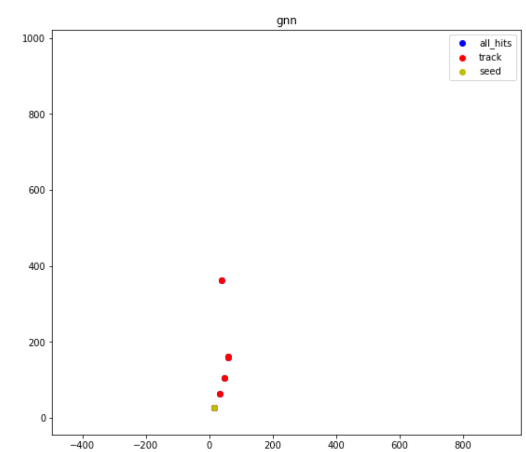
2. Embed hits, query neighbors



3. Filter neighbors as doublets against seed



4. Further filtering possible with e.g. GNN to produce tracks



Learned Embeddings, Results

- Learned representations find twice as many true doublets, while presenting one third as many false doublets
- Adaptive neighborhood size allows variable efficiency, purity
- This doublet seeding strategy is plug-and-play with the baselines
  - Results presented here are with restriction that true pairs are between adjacent layers only

Method	**Range (GeV)**	**Efficiency**	**Purity**
Learned	All	0.961	0.303
	>0.5	0.985	
	>1.0	0.983	
Cut-based	All	0.428	0.087
	>0.5	0.944	
	>1.0	0.993	

Learned Embeddings, Speed

- Graph building per event can be greatly reduced with good precision, recall
- Half precision and smaller networks offer promise to build entire graphs in under one second
- Challenges remain in training at half precision in stable manner

Precision	Net Size	Efficiency	Purity	Speed (s)
Full	4L/2048H	0.962	0.305	18
Full	3L/512H	0.961	0.303	3
Half	3L/512H	0.962	0.288	1.5

Generalizing to triplets

- Given a set of triplets, how can they be stitched together to form tracks?
- **Key idea**: build tracks by learning an embedding of triplets, where the new space has a Euclidean metric, triplets belonging to the same track are close together, and otherwise far apart
- Learn an embedding model as follows:
  - Dataset $T = \left\{\left(t_1^{(1)}, t_2^{(2)}\right), \ldots, \left(t_n^{(1)}, t_n^{(2)}\right)\right\}$, t_i a triplet of hits
  - Embedding model $\phi(x) \in \mathbb{R}^m$, where ϕ is an MLP and m is the embedding dimension
  - Supervise MLP with hinge loss l:

Let $d = \|\phi(t^{(1)}) - \phi(t^{(2)})\|_2$ be the distance between $t^{(1)}$ and $t^{(2)}$. Then

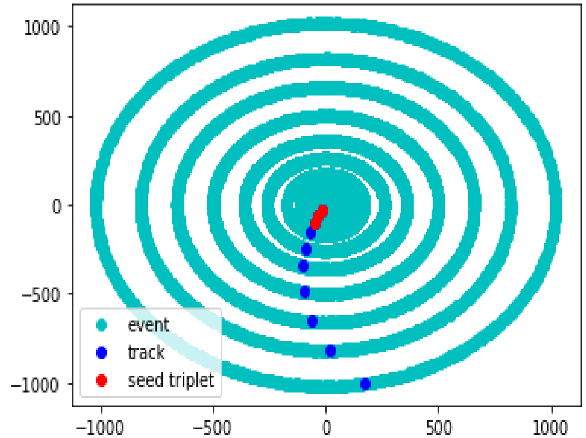
$$l(t^{(1)}, t^{(2)}) = \begin{cases} d & \text{if } t^{(1)}, t^{(2)} \text{ belong to same track} \\ \max(0, 1 - d) & \text{else} \end{cases}$$

Track building with triplets

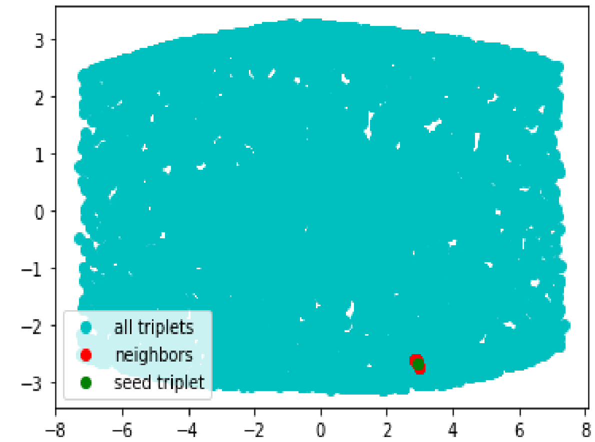
1. Receive triplets with high purity – hits inside triplets all come from the same track - from seeding algorithm
2. Embed triplets using learned embedding model
3. Query neighborhoods, filter pairs of triplets (akin to doublet filtering)
4. Build tracks iteratively, assigning hits to clusters as seeded by a triplet

Track building, triplets: Ex 1

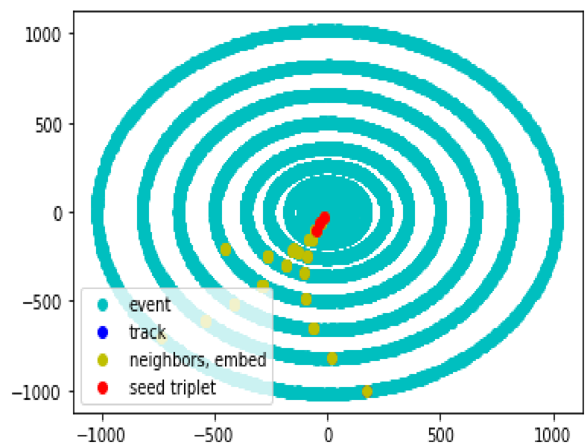
1. Seed triplet and true track



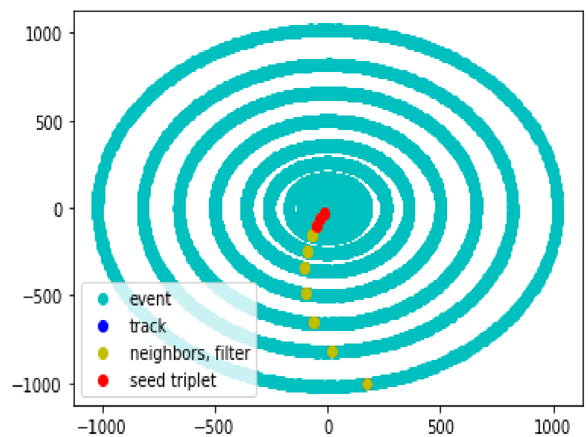
2. Embed triplets, query neighbors



3. Query neighbors from embedding

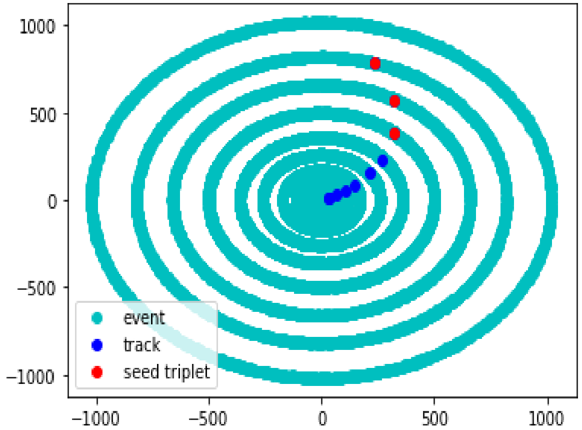


4. Filter triplets pairwise, producing track

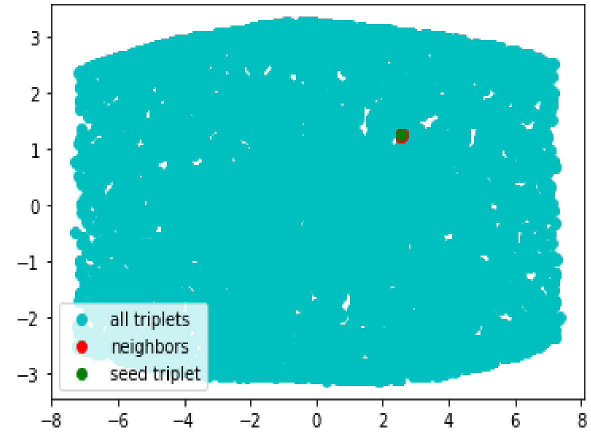


Track building, triplets: Ex 2

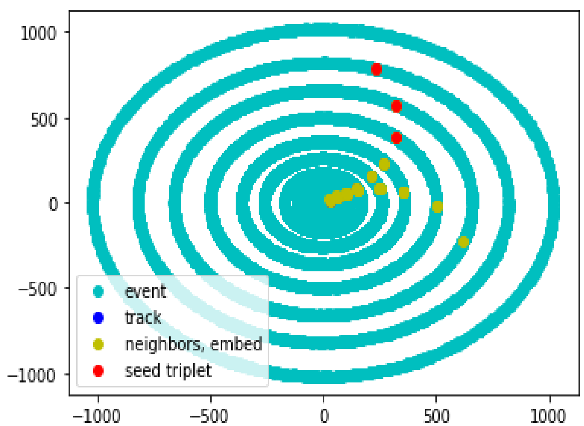
1. Seed triplet and true track



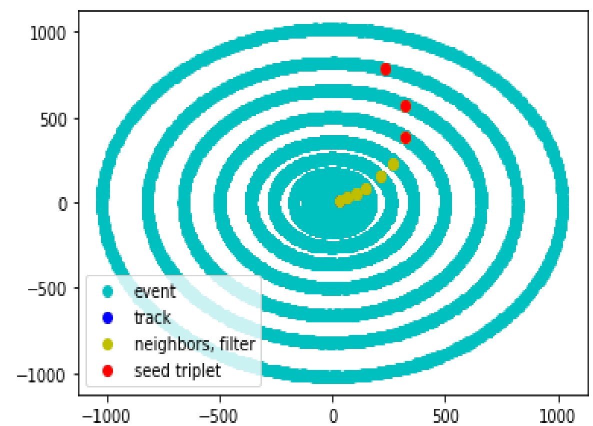
2. Embed triplets



3. Query neighbors from embedding



4. Filter triplets pairwise, producing track



TrackML score from triplets

- Given a set of triplets from triplet classifier shown in Daniel Murnane's talk, baseline TrackML score is ≈ 0.82
  - Assign all hits contained in triplets to their true particle ID
  - Any hit not assigned to a triplet is given cluster ID 0
- Building triplets using embeddings achieves a **TrackML score of ≈ 0.78**
  - Able to stitch together tracks which had a gap in Daniel's layerwise graph
  - Does not account for hits not included in any triplet

Generalizing to Tracklets

- Many ways to parameterize a track segment (curvature, angle in z-r plane, etc.) to find nearby hits, matching track segments
- **Key idea**: learn a representation of track segments using a GNN, where the new space has a Euclidean metric, hits and track segments belonging to the same track are close
- Similar to the previous learned representation model:
  - Dataset $X = \left\{\left(c_1^{(1)}, c_2^{(2)}\right), \ldots, \left(c_n^{(1)}, c_n^{(2)}\right)\right\}$, c_i is a cluster of variable size, containing hits which belong to the same track
  - Embedding model $\Psi(c) \in \mathbb{R}^m$, where Ψ is a message-passing GNN and m is the embedding dimension
  - Supervise GNN with hinge loss l:

Let $d = \|\Psi\left(c^{(1)}\right) - \Psi\left(c^{(2)}\right)\|_2$ be the distance between $c^{(1)}$ and $c^{(2)}$. Then

$$l\left(c^{(1)}, c^{(2)}\right) = \begin{cases} d & \text{if } c^{(1)}, c^{(2)} \text{ represent the same track} \\ \max(0, 1 - d) & \text{else} \end{cases}$$

Future Directions

- Recover TrackML score through learned metric between triplets and hits
- Stitch together triplets with a GNN, incorporating neighborhood information in addition to pairs of triplets
- Alternative training strategies for embeddings
  - Contrastive triplet loss, or n-pair loss for better convergence
  - Angular loss allows for scale invariance in embedded space
- Optimize graph building to balance speed with efficiency
  - Vary neighborhood size against filtering threshold
  - Reduce MLP size, incorporate half-precision training