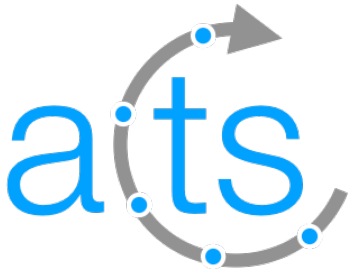
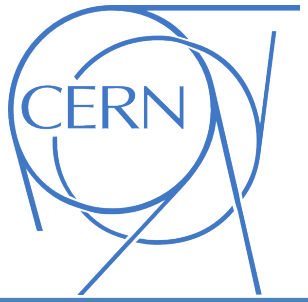
Machine Learning for Ambiguity Resolution in Acts

Corentin Allaire
Hadrien Grasland
David Rousseau
Françoise Bouvet



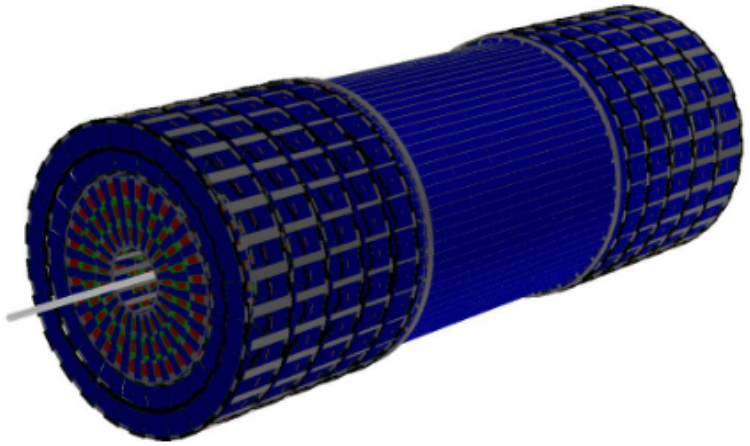


Acts : A Common Tracking Software

- Open source tracking software:
  https://github.com/acts-project/acts
- Experiment independent toolkit :
  - ATLAS, ALICE, BVG, FASER
  - LDMX, sPHENIX, EIC, …

Testing environment for new tracking algorithms:

- Open Data Detector (ODD) :
  - Virtual detector : benchmarking
  - Based on the **Track ML challenge**
  - Full silicon design (similar to ATLAS ITk)
  - Realistic detector material

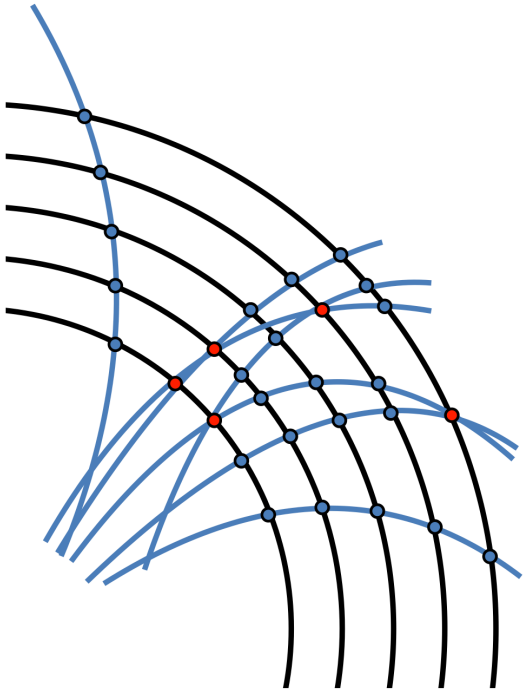


- Used in a full tracking chain
  ➡ Tracking **performances** evaluation

- Excellent environment for developing/testing **machine learning** based algorithms

What is Ambiguity Resolution?

- After track reconstruction : many tracks are **duplicates** (same associated truth particle)
- **Fake** tracks : combination of arbitrary hits (different truth particles)
- Ambiguity solver : **remove both** and handle **hits shared** by multiple tracks

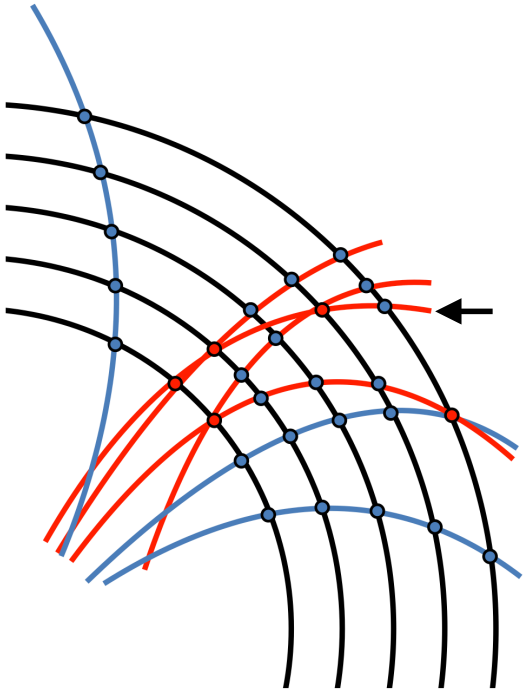


- Implemented in Acts : **Greedy Solver** (decent but is relatively slow)
  - Select tracks with too many shared hits (> 2 in practice)
  - Remove the tracks with the highest **relative shared hits**
  - **Recompute** shared hit and **repeat**

- Classification problem : excellent opportunity for a **ML-based solution** in Acts

What is Ambiguity Resolution?

- After track reconstruction : many tracks are **duplicates** (same associated truth particle)
- **Fake** tracks : combination of arbitrary hits (different truth particles)
- Ambiguity solver : **remove both** and handle **hits shared** by multiple tracks

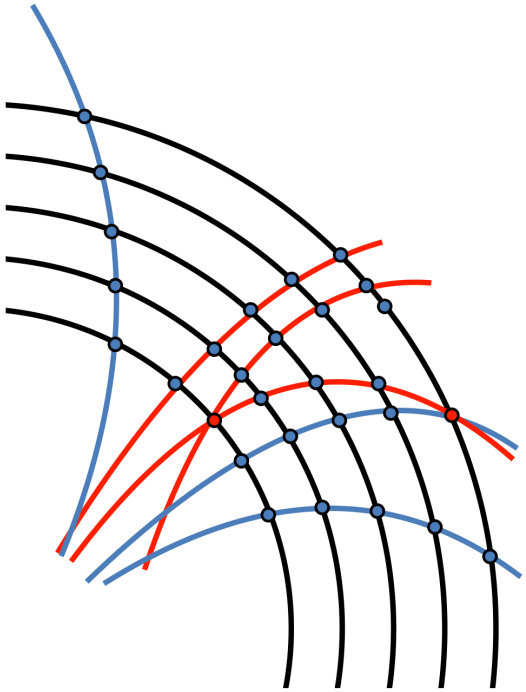


- Implemented in Acts : **Greedy Solver** (decent but is relatively slow)
  - Select tracks with too many shared hits (> 2 in practice)
  - Remove the tracks with the highest **relative shared hits**
  - **Recompute** shared hit and **repeat**

- Classification problem : excellent opportunity for a **ML-based solution** in Acts

What is Ambiguity Resolution?

- After track reconstruction : many tracks are **duplicates** (same associated truth particle)
- **Fake** tracks : combination of arbitrary hits (different truth particles)
- Ambiguity solver : **remove both** and handle **hits shared** by multiple tracks

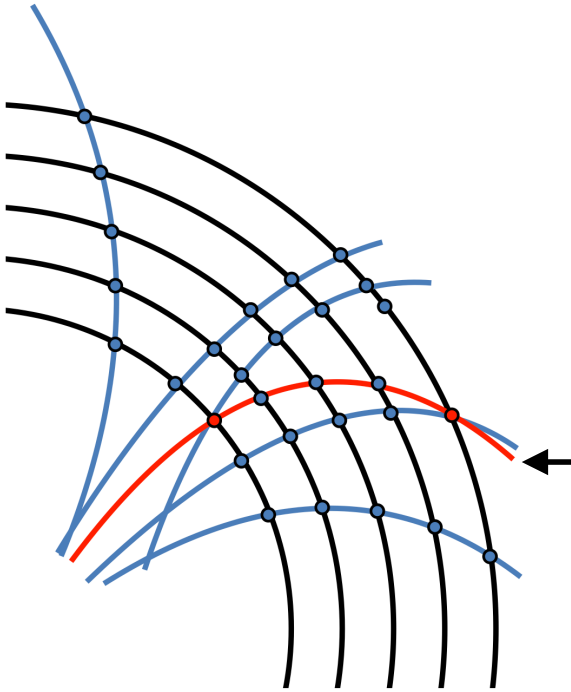


- Implemented in Acts : **Greedy Solver** (decent but is relatively slow)
  - Select tracks with too many shared hits (> 2 in practice)
  - Remove the tracks with the highest **relative shared hits**
  - **Recompute** shared hit and **repeat**

- Classification problem : excellent opportunity for a **ML-based solution** in Acts

What is Ambiguity Resolution?

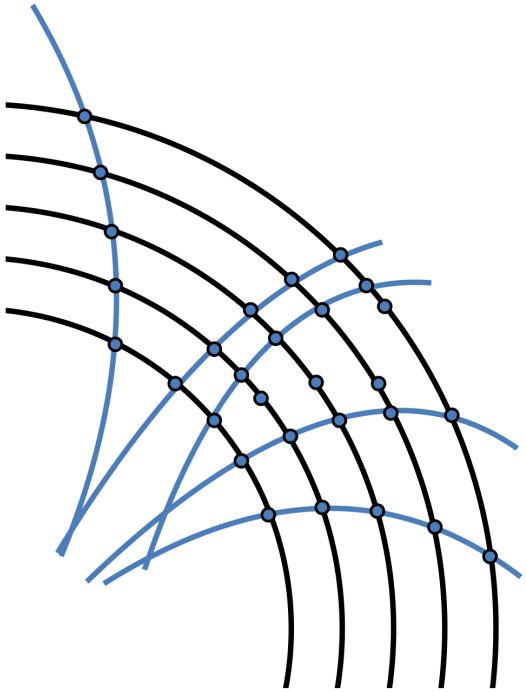
- After track reconstruction : many tracks are **duplicates** (same associated truth particle)
- **Fake** tracks : combination of arbitrary hits (different truth particles)
- Ambiguity solver : **remove both** and handle **hits shared** by multiple tracks



- Implemented in Acts : **Greedy Solver** (decent but is relatively slow)
  - Select tracks with too many shared hits (> 2 in practice)
  - Remove the tracks with the highest **relative shared hits**
  - **Recompute** shared hit and **repeat**
- Classification problem : excellent opportunity for a **ML-based solution** in Acts

What is Ambiguity Resolution?

- After track reconstruction : many tracks are **duplicates** (same associated truth particle)
- **Fake** tracks : combination of arbitrary hits (different truth particles)
- Ambiguity solver : **remove both** and handle **hits shared** by multiple tracks



- Implemented in Acts : **Greedy Solver** (decent but is relatively slow)
  - Select tracks with too many shared hits (> 2 in practice)
  - Remove the tracks with the highest **relative shared hits**
  - **Recompute** shared hit and **repeat**

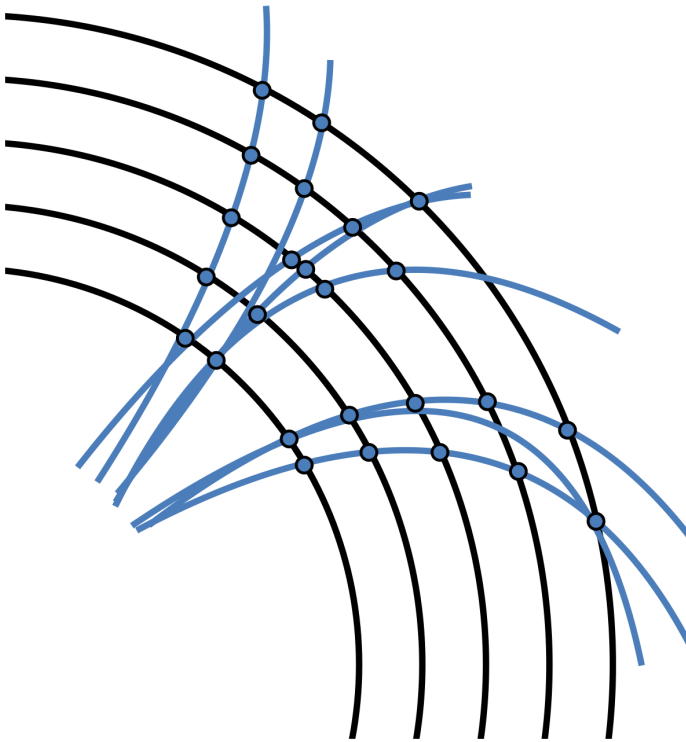
- Classification problem : excellent opportunity for a **ML-based solution** in Acts

Machine learning based Ambiguity Solving

Cluster Tracks together → Network to determine the best track per cluster

Clustering: cluster together nearby tracks,
1 cluster = 1 truth particle

- Tracks from the same particle should share hits
  - **shared hit based clustering :**
    - Select **leading track** for cluster : **most hits**, no hits shared with existing cluster
    - Hits shared with a leading track : add to the **cluster**

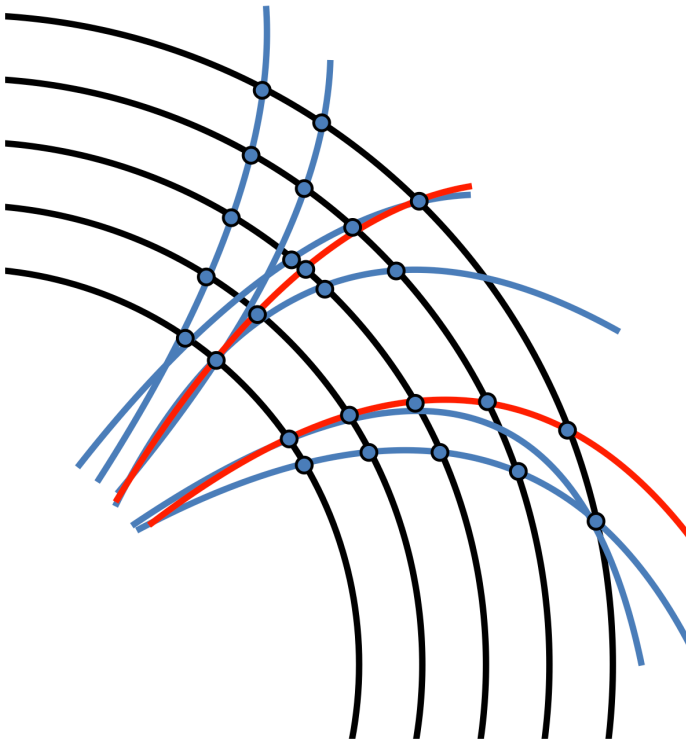


Machine learning based Ambiguity Solving

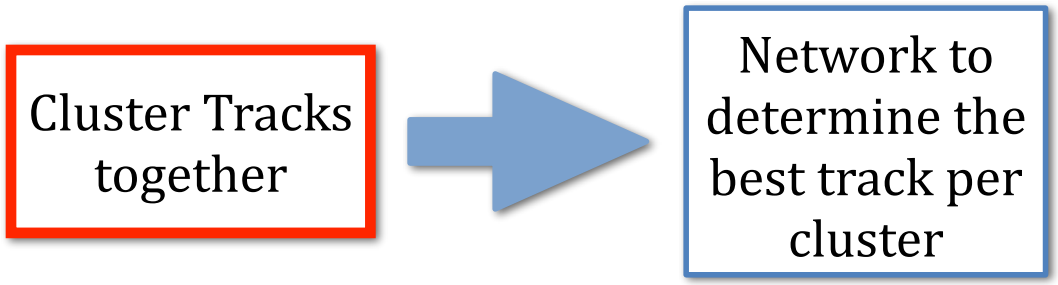
Cluster Tracks together → Network to determine the best track per cluster

Clustering: cluster together nearby tracks,
1 cluster = 1 truth particle

- Tracks from the same particle should share hits
  ➧ **shared hit based clustering :**
  - Select **leading track** for cluster : **most hits**, no hits shared with existing cluster
  - Hits shared with a leading track : add to the **cluster**

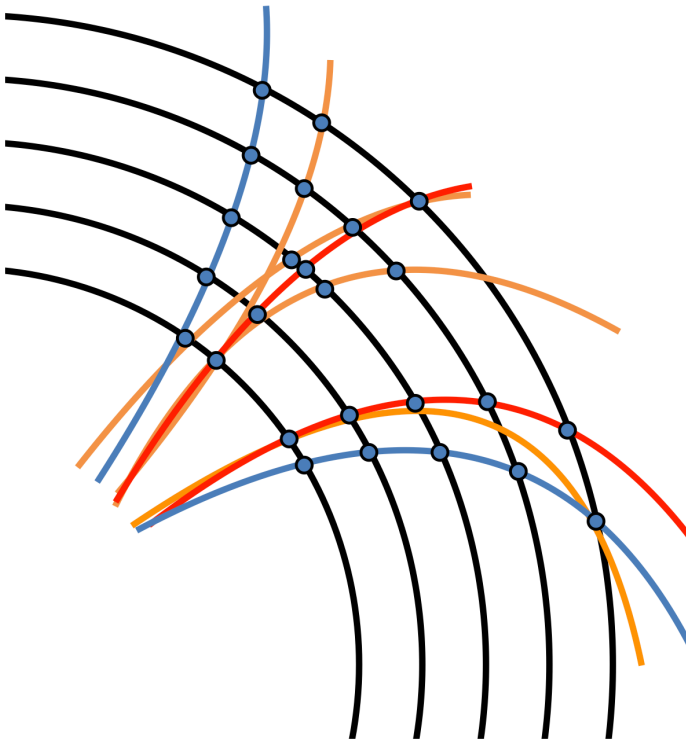


Machine learning based Ambiguity Solving

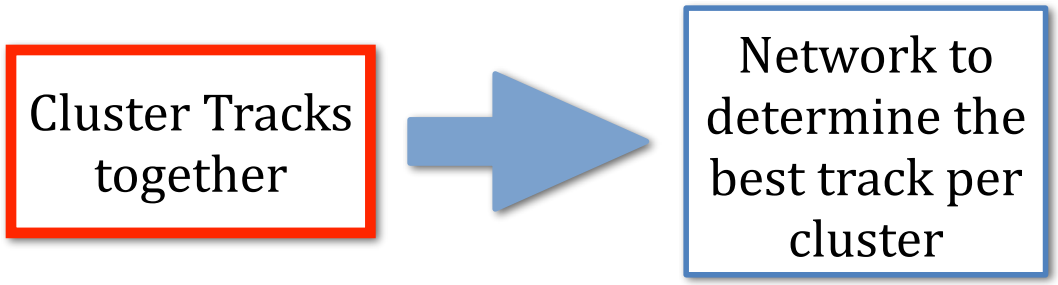


Clustering: cluster together nearby tracks,
1 cluster = 1 truth particle

- Tracks from the same particle should share hits
  ➧ **shared hit based clustering :**
  - Select **leading track** for cluster : **most hits**, no hits shared with existing cluster
  - Hits shared with a leading track : add to the **cluster**

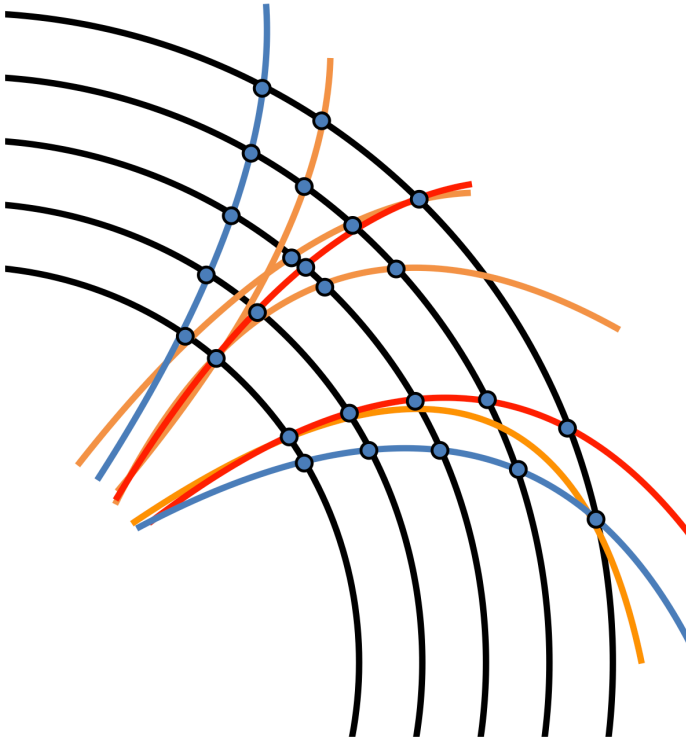


Machine learning based Ambiguity Solving



Clustering: cluster together nearby tracks,
1 cluster = 1 truth particle

- Tracks from the same particle should share hits
  ➧ **shared hit based clustering :**
  - Select **leading track** for cluster : **most hits**, no hits shared with existing cluster
  - Hits shared with a leading track : add to the **cluster**
- Other **clustering** approaches tested : **DBScan**
  - Pre-cluster track (still need shared hit clustering after)
  - Slight efficiency increase but 50% slower
  - Available in Acts but not shown today



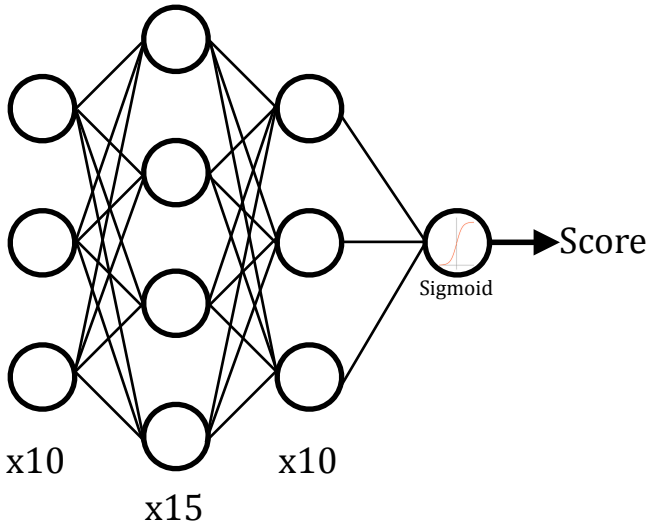
Machine learning based Ambiguity Solving

Cluster Tracks together

Network to determine the best track per cluster

Neural Network: Score each track, keep highest score per cluster :

- Three layers NN, used for ranking
- Use 8 track variables as input
- **One score per track ➡ Select the best one**



Advantages :

- No parameter tuning, only a short training
- Available in Acts vis **Onnxruntime**

Input parameters :

- Number of states
- Number of measurements
- Number of Outliers
- Number of Holes
- NDF
- Chi2/NDF
- Eta
- Phi

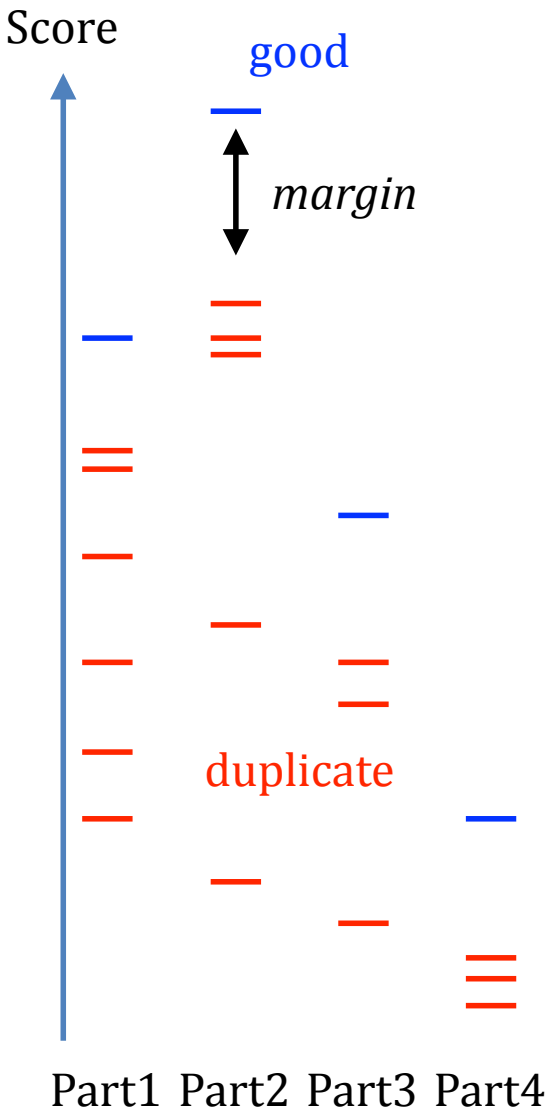
Ranking Neural Network

Goal : one score per track, highest for the **good** one

- Training without clustering ➔ use **truth** matching instead
- Compute one loss per **truth particle**
- Use a **Margin Ranking Loss :**

$$loss_{part} = \frac{1}{N_{tracks}} \sum^{tracks} max(0,\ x - y + margin)$$

- x: track score; y: good track score; margin = 0.05
- If $score_{bad} < (score_{good} - margin)$, then loss = 0
  Else, loss = score difference minus the margin
- Try to **separate the good and bad scores** by at least a *margin*



Performances : definition

- Full tracking chain with the ODD in Acts ➔ Compare the **Greedy Solver** and the **ML Solver**
- Geant4 + Pythia simulation of 1000 ttbar events (+ 1000 for training)
- Timing measured on a local server ➔ focus on **ratio** more than absolute value
- Only consider tracks with ≥7 measurements

Definition :

- **Good track**: for a truth particle, track with the most **truth match measurements**, then **fewer outliers**, then the **smallest chi2**
- **Duplicate** : > 50% truth-matched measurements
- **Fake**: < 50 % truth-matched measurements

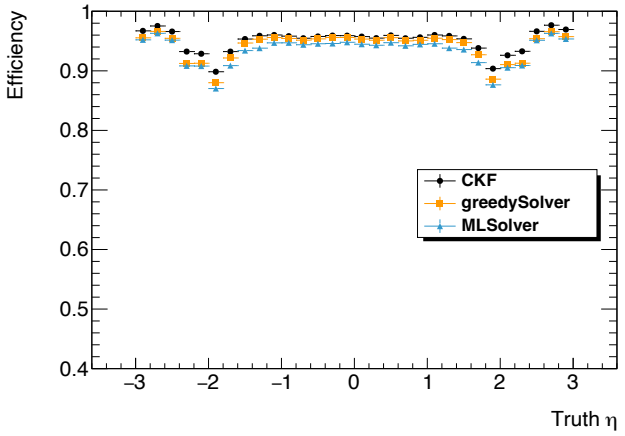
Performances : Efficiency

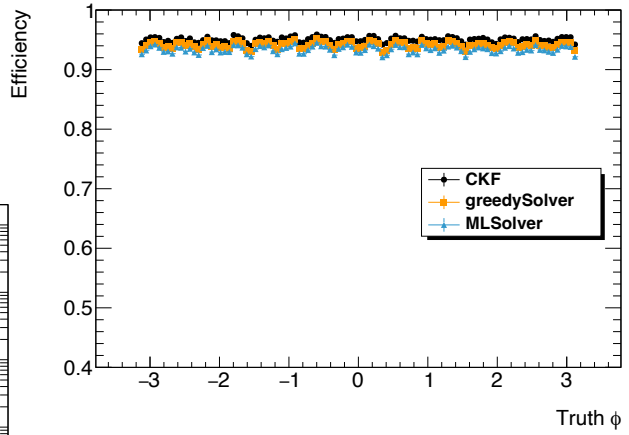
- Reference : output of the Acts **Combinatorial Kalman Filter** (CKF)
- Rates are with respect to the number of tracks at the current step
- **Efficiency (good tracks)** : Fraction of the original good tracks still present
  ➡ Quality of the **selection** (ranking network)
- **Efficiency (truth tracks)** : Fraction of the original truth particles still present
  ➡ Quality of the **clustering**

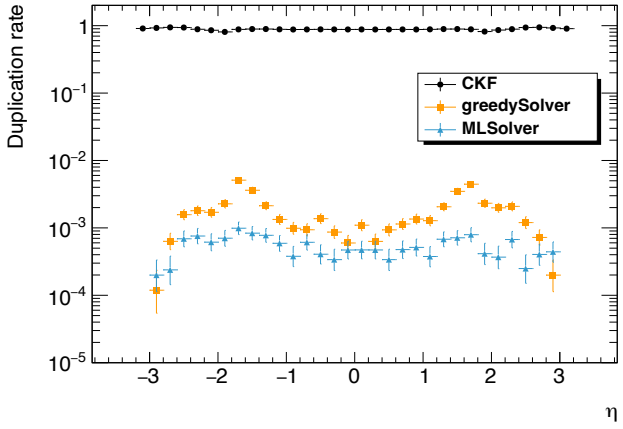
	Number of tracks	Number of truth particles	Efficiency (good tracks)	Efficiency (truth tracks)	Duplicate Rate	Fake Rate	Solver speed [ms/event]
CKF	7995	834.7	100 %	100 %	89.5 %	0.06 %	0
CKF + Greedy Solver	823.6	821.4	81.5 %	98.4 %	0.17 %	0.10 %	184
CKF + ML Solver	811.7	810.7	84.2 %	97.1 %	0.05 %	0.06 %	41.2

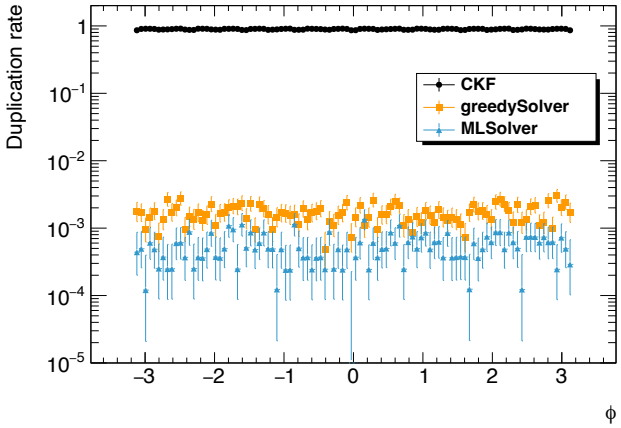
Performances : Efficiency

- Slightly worst efficiency with the ML Solver
  ➧ Effect consistent across the full detector
- Outperform the Greedy solver in duplicate removal in the **forward region**
- **Perform constantly** across the entire detector range









Summary

- **ML Ambiguity solver**: Combine **clustering** and a **Ranking** neural network
- The ML solver shows great performances: ~5 time faster than the classical one and similar performances
- Available right now in Acts with an example to run it with the ODD, can be tested by any experiment using Acts

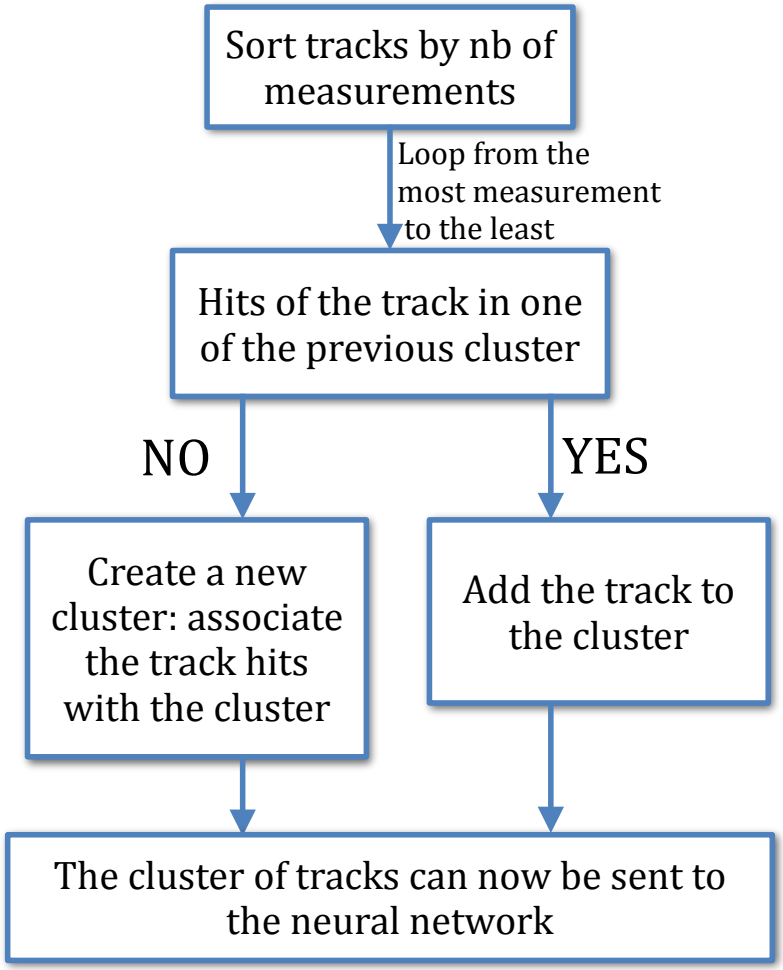
Outlook

- Fine-tuning of the Clustering and the Network
- Generalisation to seeds: Select the most promising seed before track finding
  ➡ potential significant speedup

BACKUP

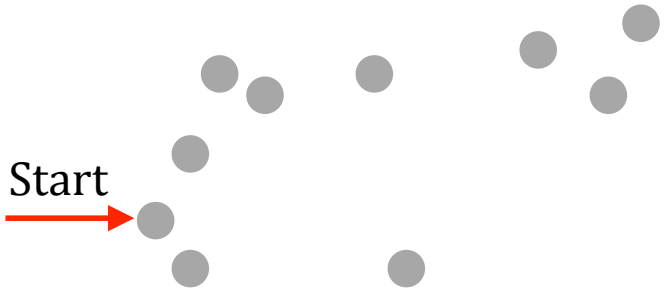
Shared hits based clustering

- Idea : 1 cluster = 1 truth particle
- Still needed in the DBScan case to create a sub-cluster with hits-sharing tracks
- Based purely on unordered_map, the speed shouldn't decrease with the number of tracks
- 1 Cluster ~ 1 track with a large number of measurements (More measurements ➡ , better track)
- Add track to the cluster if they share a hit with the primary track



DBScan clustering

- Idea : 1 cluster = 1 truth particle
- sklearn in Python; mlpack in C++
- Clustering based on data density
- Use 2 parameters :
  - ε: Max distance between neighbour
  - Min_{sample}: Min number of elements per cluster
- More than Min_{sample} neighbour ➡ Create a cluster
- For each element of the cluster, do the same ➡ extend the cluster
- In the Ambiguity Solver :
  - distance in (η,ϕ); ε=0.07 ; Min_{sample}=2



Min_{sample}=3