Tracking with Hashing

27th November 2024

Jeremy Couthures

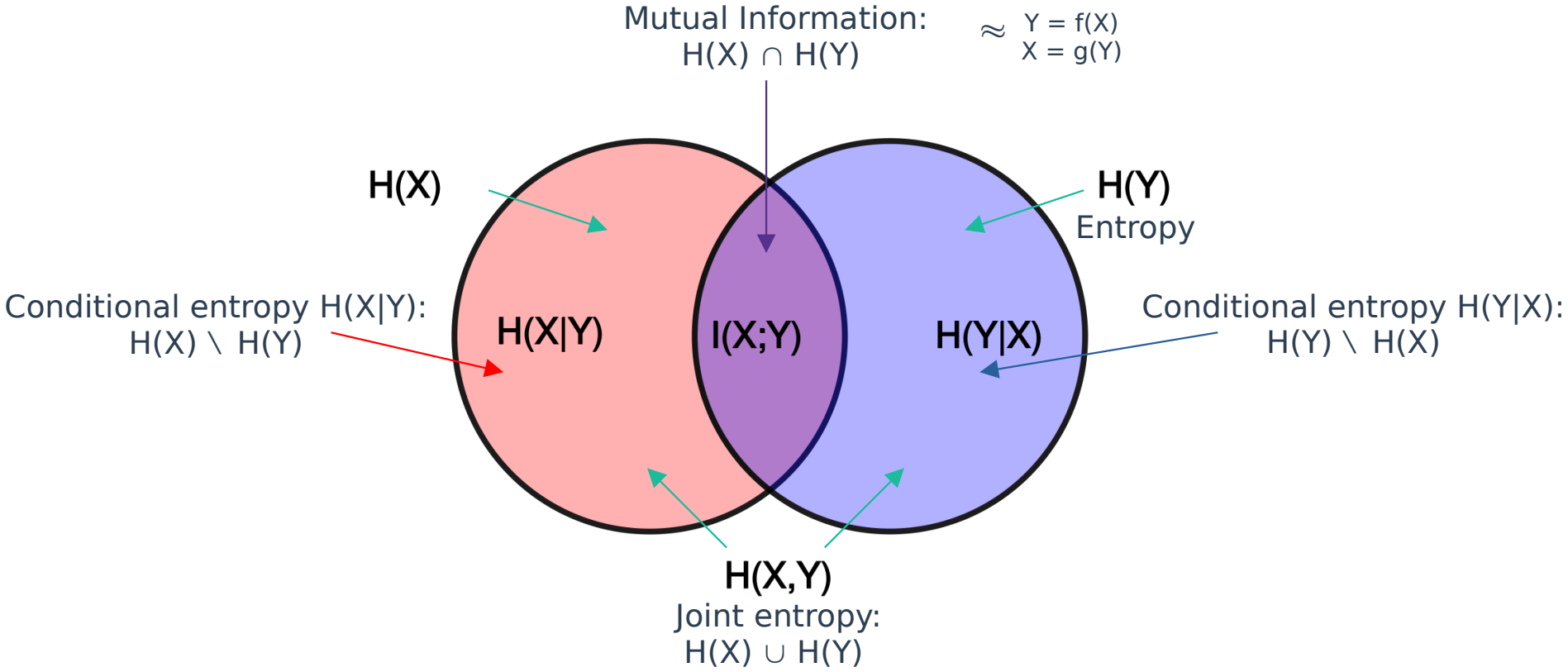
Known high-level features

- **Assumption:**

  the model is using high-level features in the output latent space (12 neurons)

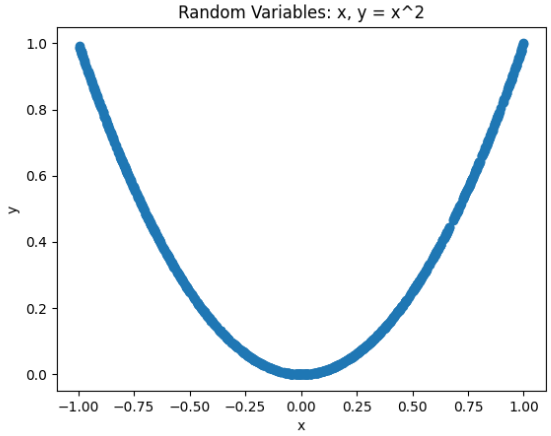
- **Approach:**
  - Information theory: conditional entropy of high-level features conditioned on the output latent space → gives how much of the high-level feature can be predicted from the latent space alone

Entropy


Mutual Information:
H(X) ∩ H(Y)
≈ Y = f(X)
X = g(Y)
H(X)
H(Y)
Entropy
Conditional entropy H(X|Y):
H(X) \ H(Y)
H(X|Y)
I(X;Y)
H(Y|X)
Conditional entropy H(Y|X):
H(Y) \ H(X)
H(X,Y)
Joint entropy:
H(X) ∪ H(Y)

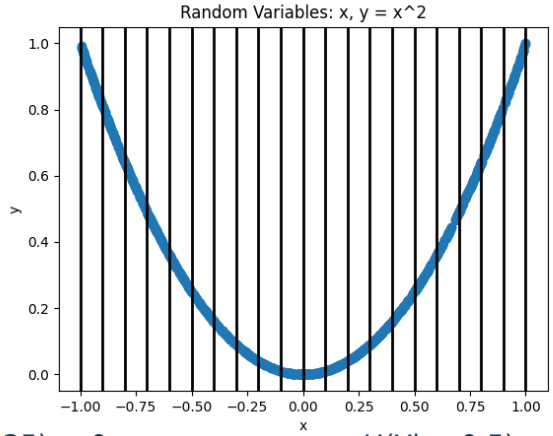
Conditional entropy

$$\mathrm{H}(Y|X) \ = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)}$$

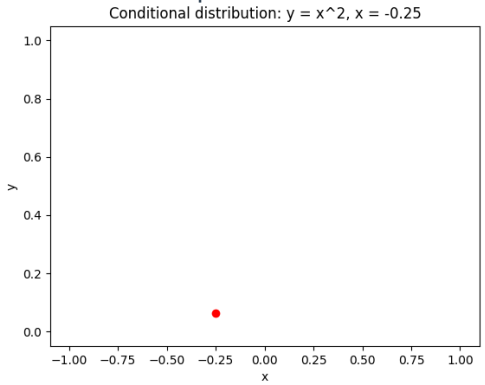


Conditional entropy

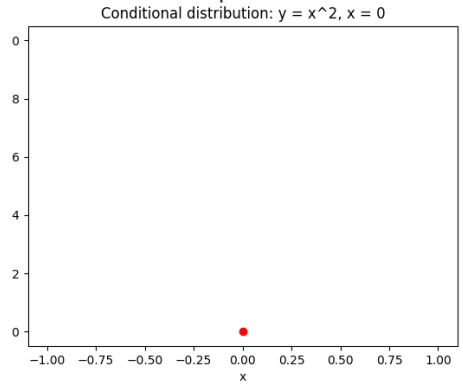
$$\mathrm{H}(Y|X) \ = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)}$$



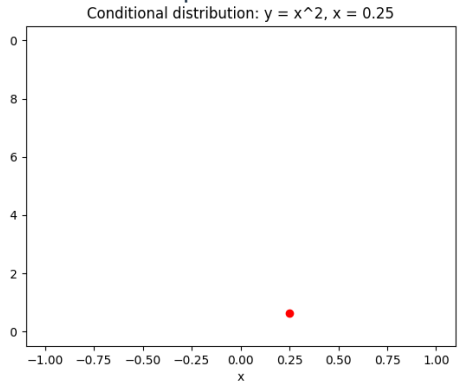
H(Y|x=-0.25) = 0



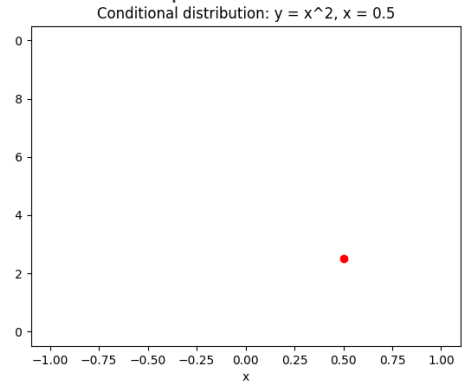
H(Y|x=0) = 0



H(Y|x=0.25) = 0

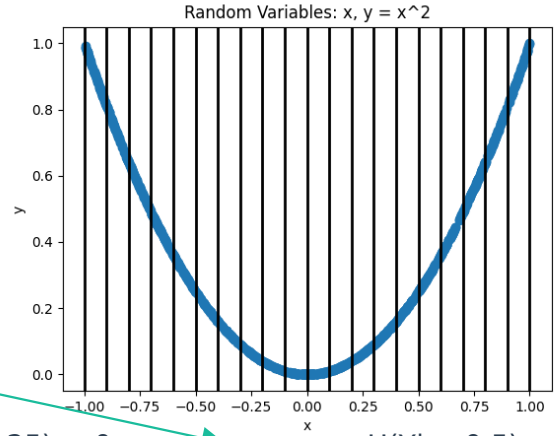


H(Y|x=0.5) = 0



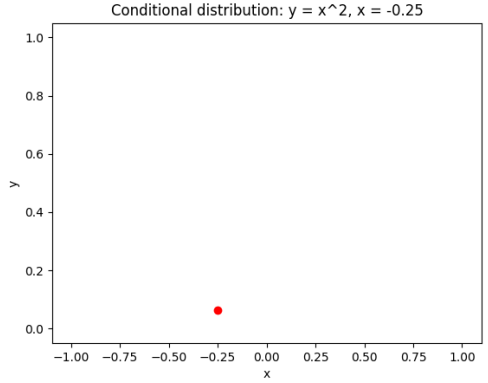
Conditional entropy

$$\mathrm{H}(Y|X) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)}$$

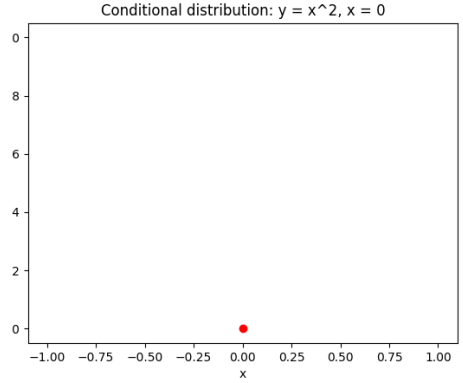


H(Y|X) = 0

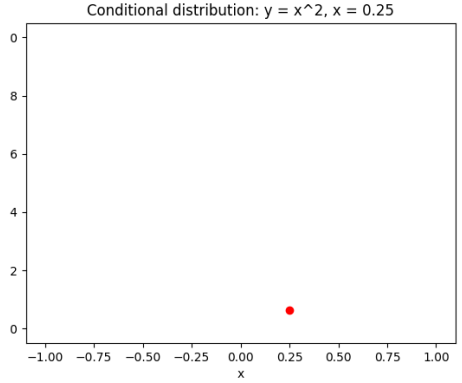
H(Y|x=-0.25) = 0



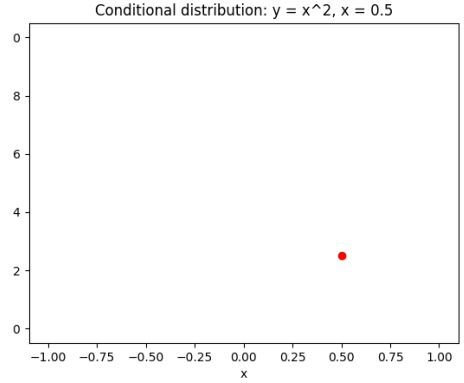
H(Y|x=0) = 0



H(Y|x=0.25) = 0



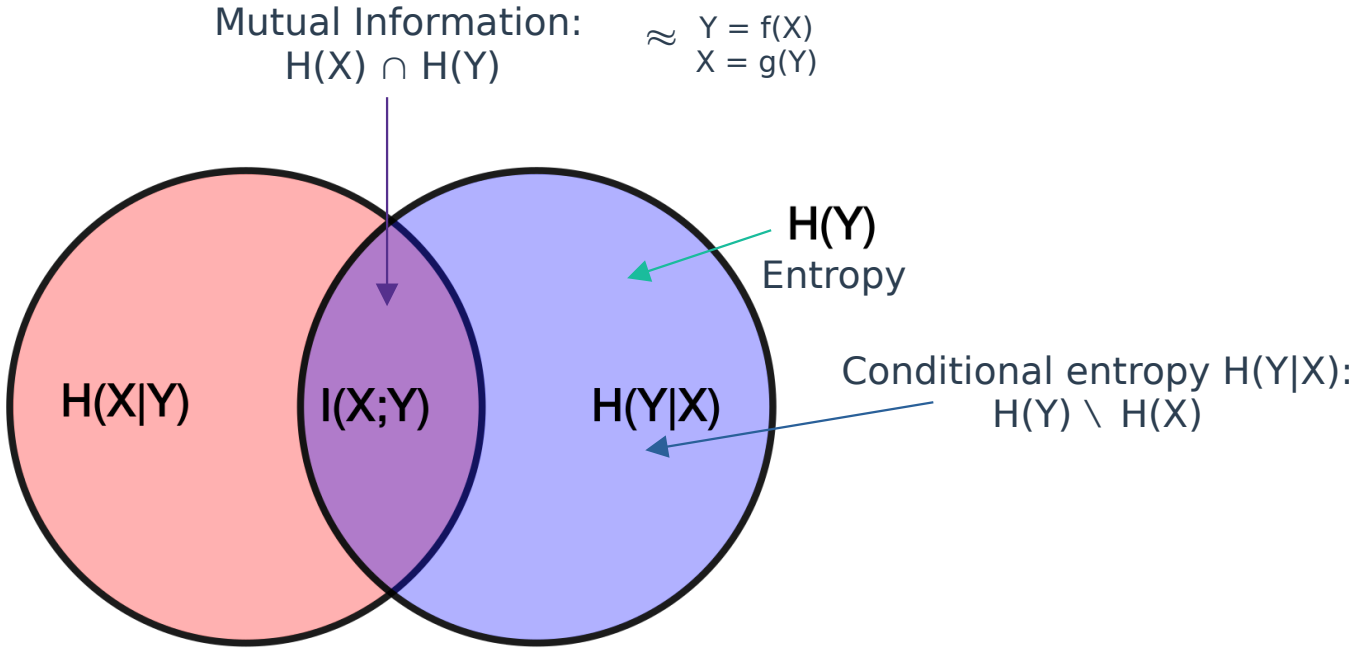
H(Y|x=0.5) = 0



Information coverage

Information coverage
(*not an official name*):

$$\frac{H(Y) - H(Y|X)}{H(Y)}$$



= 1 when H(Y|X) = 0 (full dependency)
= 0 when H(Y|X) = H(Y) (independency)

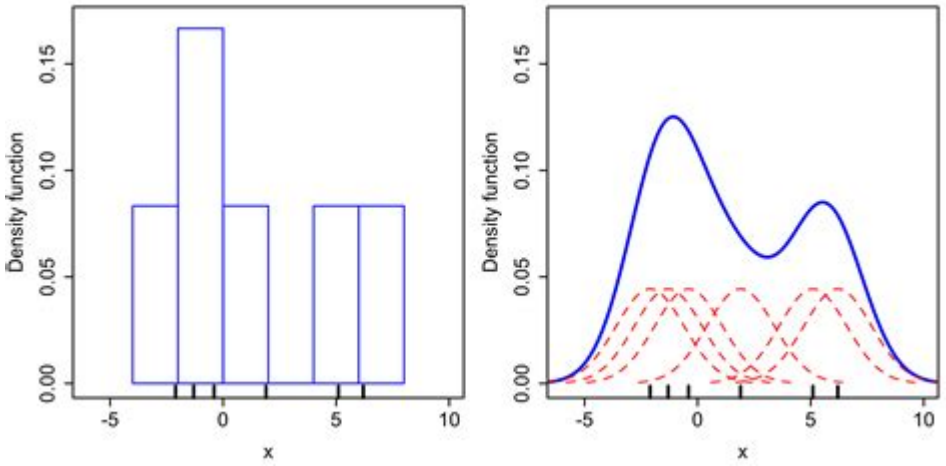
Kernel Density Estimation

- **Need to have the probability distribution**

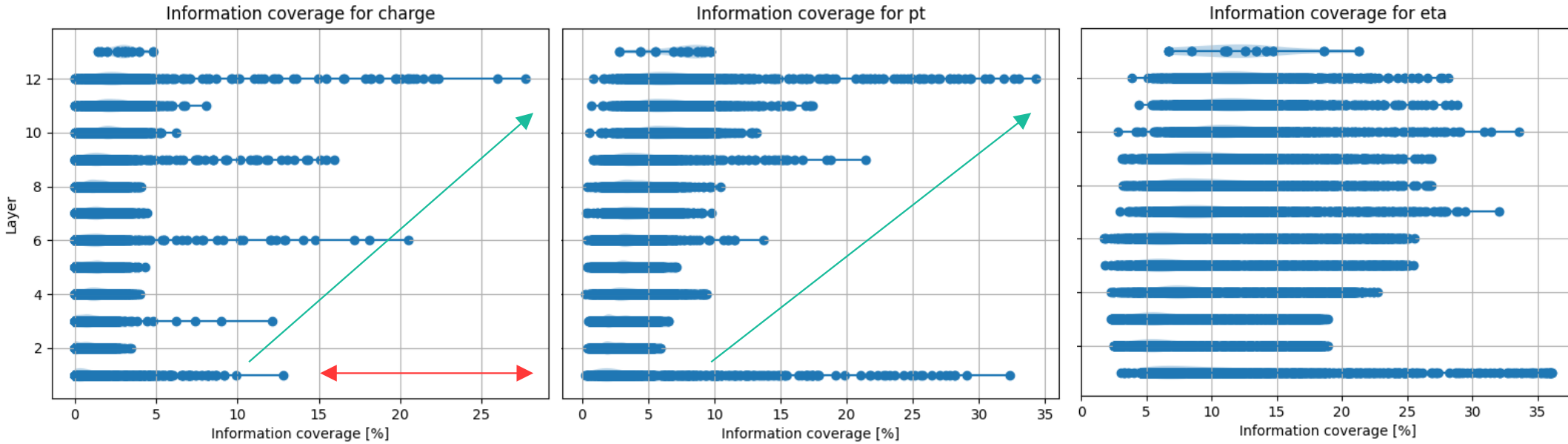
- **Kernel density estimation:**
  - Fit parameter: h
  - Put a Kernel function K(x,h) in each point and sum them to get the density estimation

  - Gaussian kernel (`kernel = 'gaussian'`)

$$K(x;h) \propto \exp(-\frac{x^2}{2h^2})$$

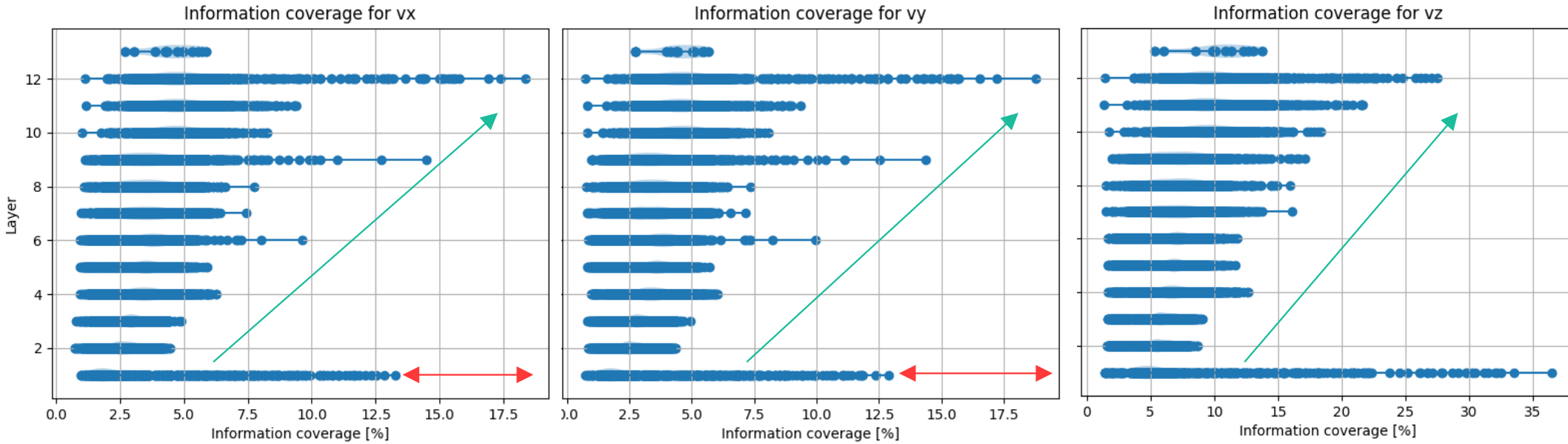


High-level variables single neuron



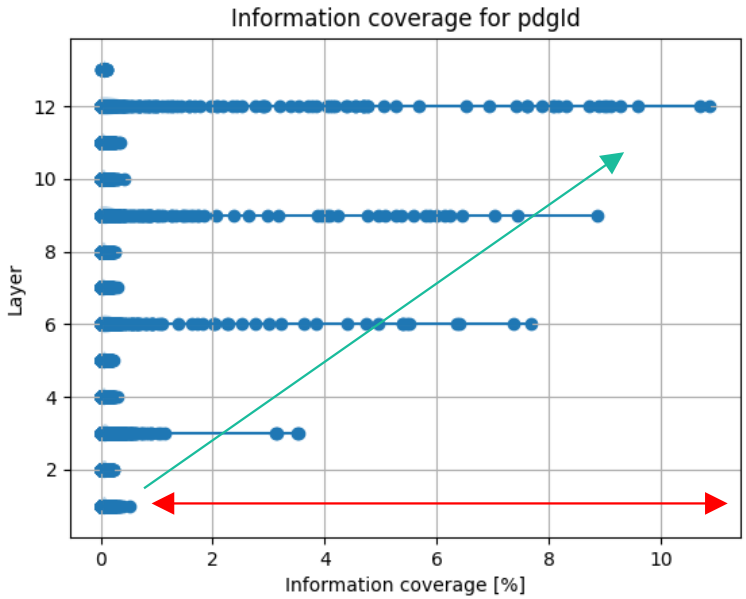
h = 0.02

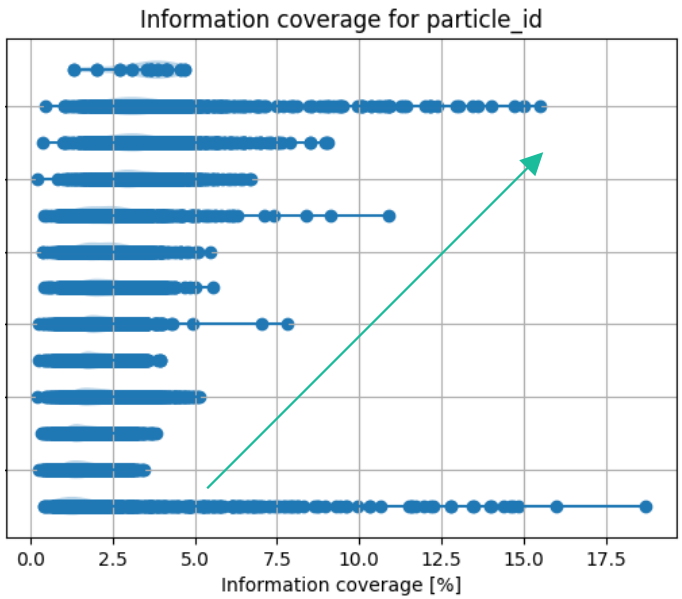
High-level variables single neuron

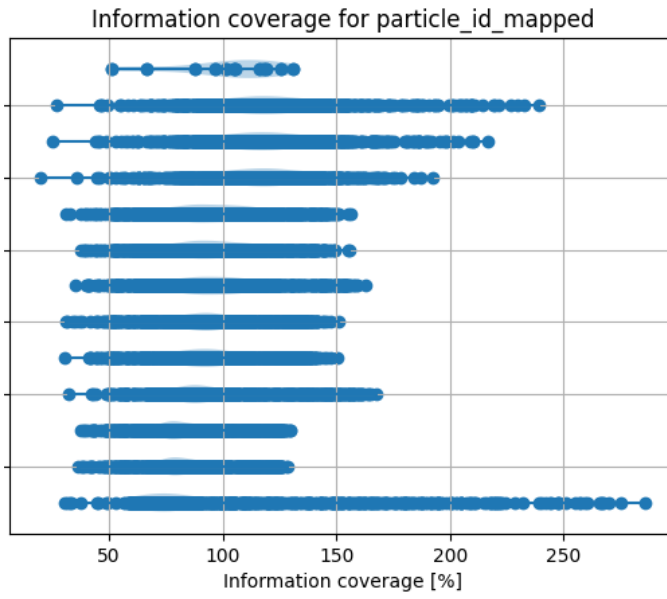


h = 0.02

High-level variables single neuron





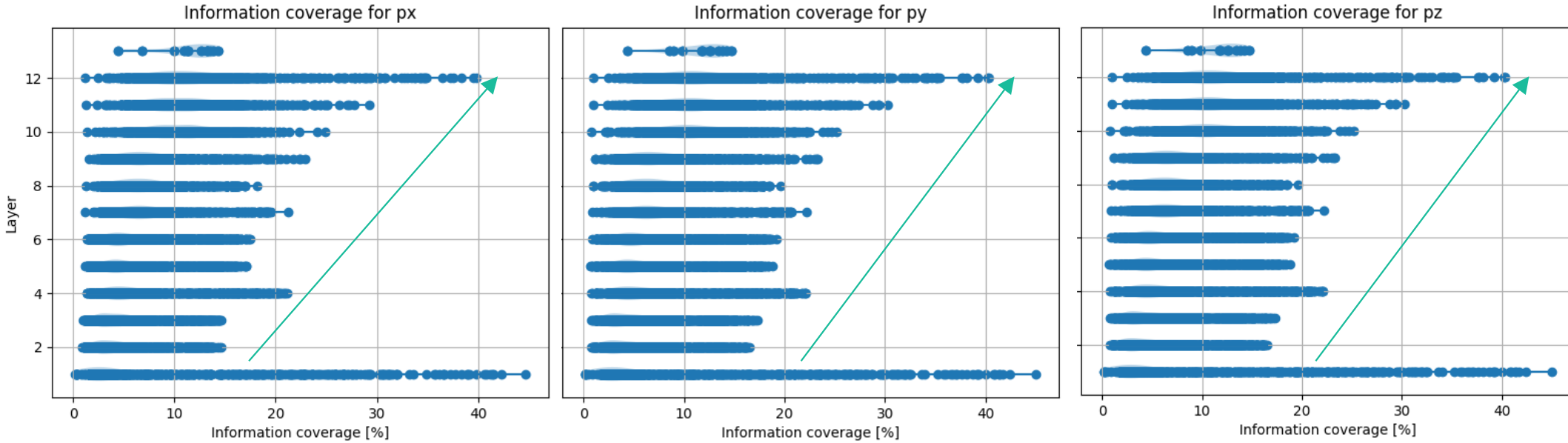


Learned to isolate some particles

Broken

h = 0.02

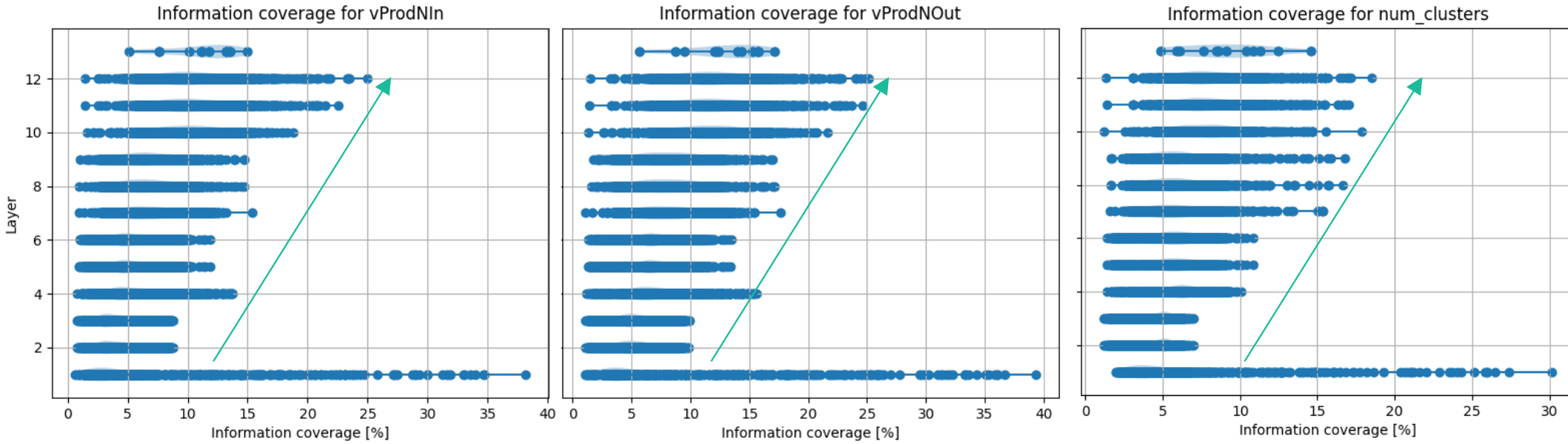
High-level variables single neuron



Same

h = 0.02

High-level variables single neuron



h = 0.02

Layer information coverage

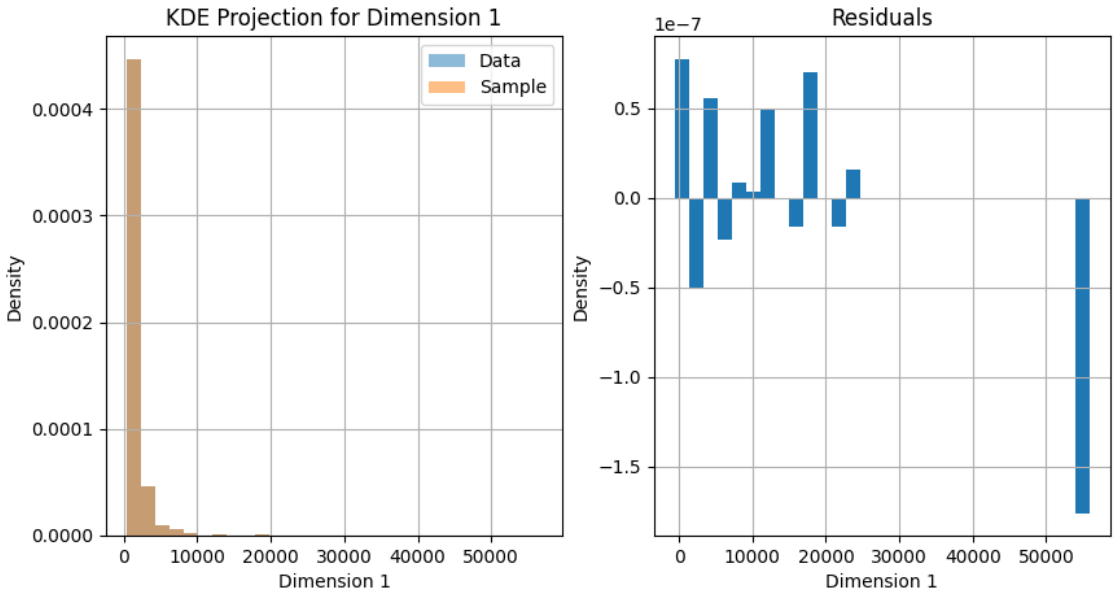
- **Information coverage varies with h**

h=0.5

layer	particle_id	subevent	barcode	px	py	pz	pt	eta	vx	vy	vz	radius	status	charge	pdgId	vProdNIn	vProdNOut
12	0.57	0.38	0.55	0.54	0.54	0.54	0.51	0.56	0.38	0.38	0.38	0.35		0.09	0.17	0.31	0.34

Goodness of fit

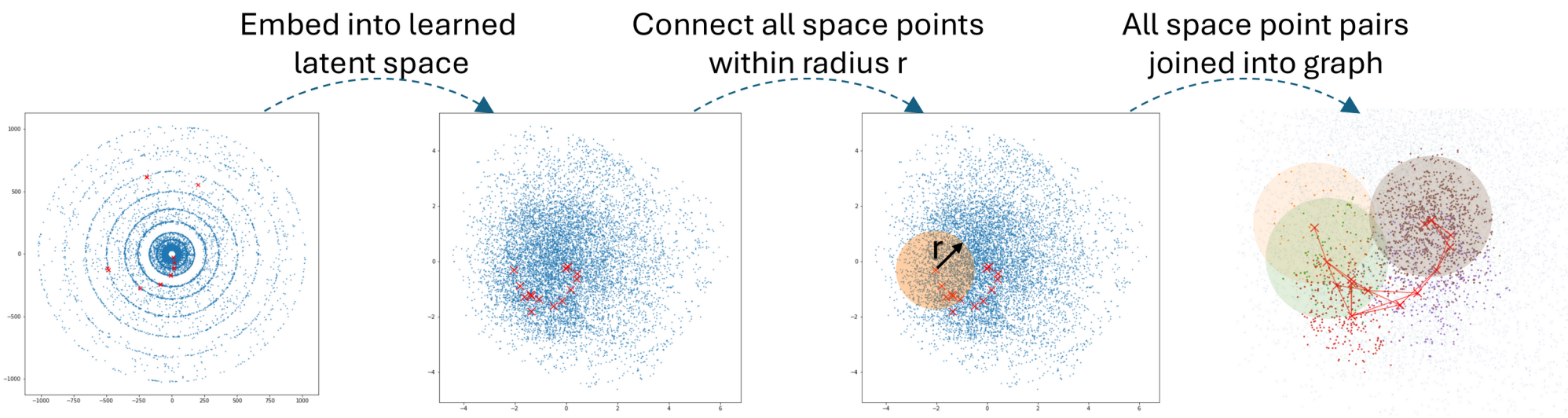
- **Sample from the fit KDE**



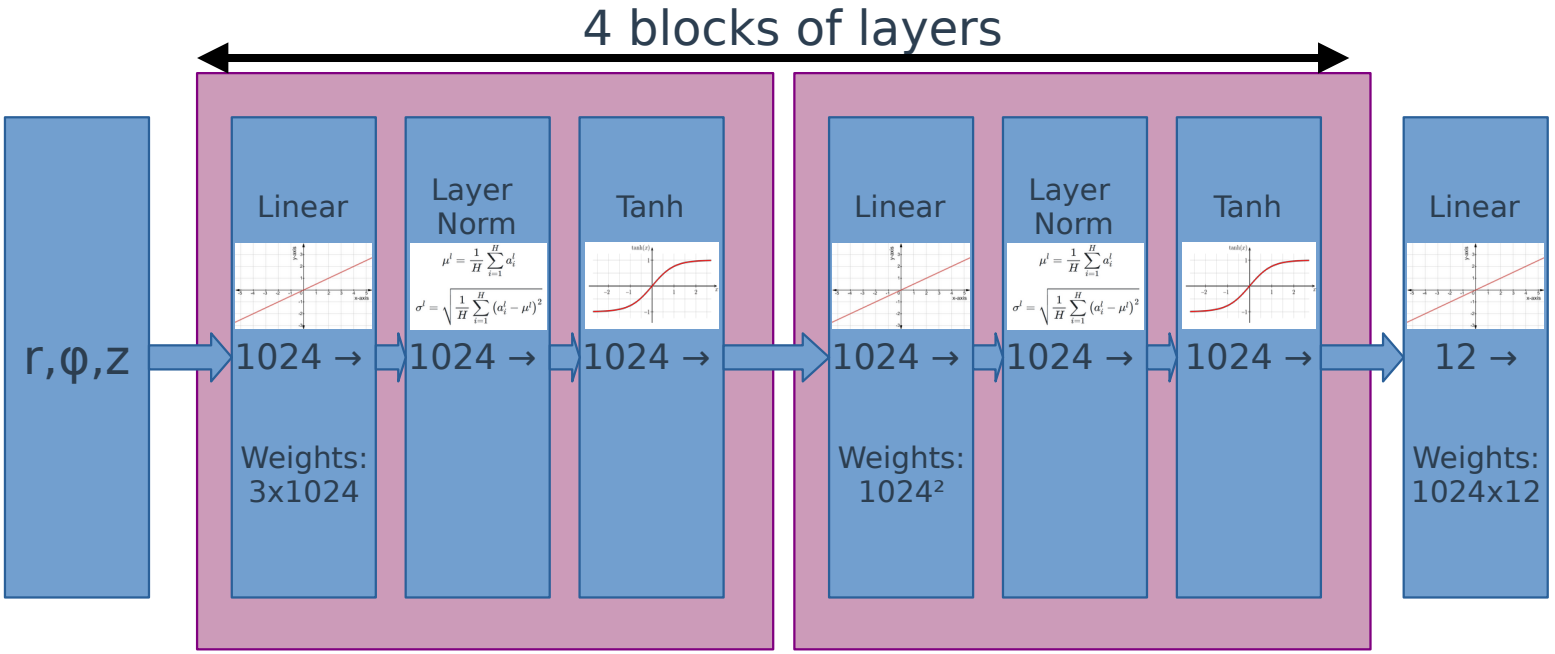
BACKUP

- **Plots en fonction de r-phi-z = pas la bonne approche**
- **Cherche à faciliter la reconstruction des traces → ce que les points d'une même trace ont en commun → doit regarder en fonction des paramètres des traces et comparer pour différentes traces**

GNN Metric Learning



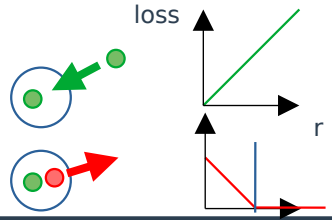
Architecture



Latent space: 12 variables

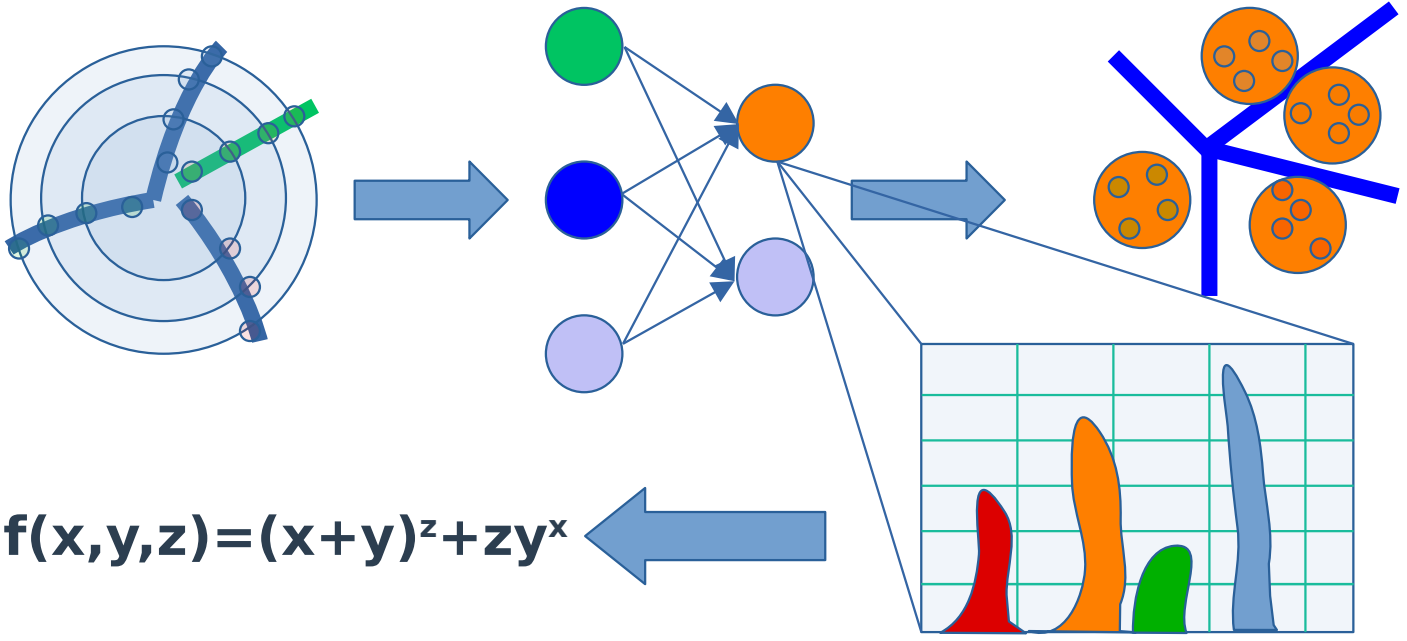
Hinge Loss

$$l_n = \begin{cases} x_n, & \text{if } y_n = 1, \\ \max\{0, margin - x_n\}, & \text{if } y_n = -1, \end{cases}$$



Interpretability

- study the internal representation of the problem by the model



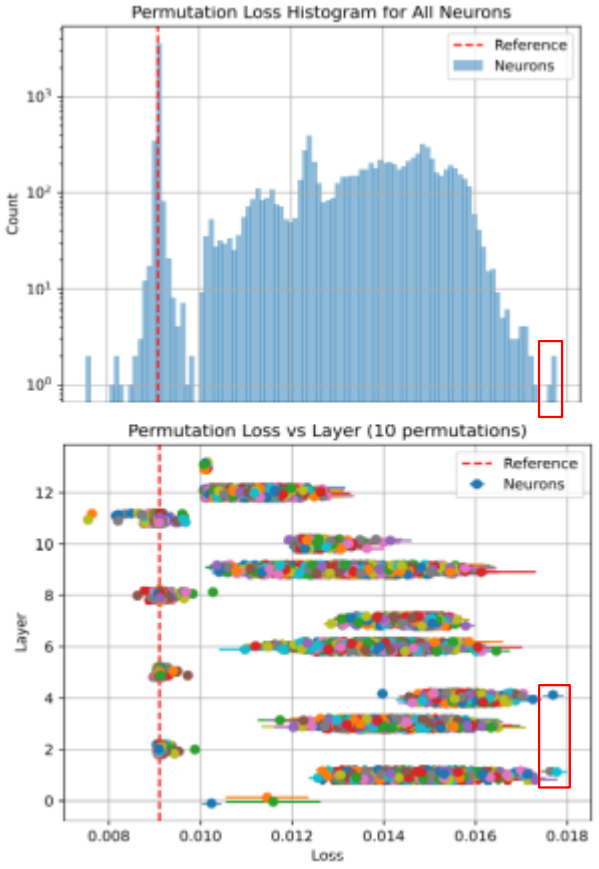
$f(x,y,z)=(x+y)^z+zy^x$

Software implementation

Identify High Level variables

Neuron identification: Permutation loss

- **3 promising neurons:**
  - 2 on layer 1 (***Linear*** with input layer)
  - 1 on layer 4 (More complex)
- **Normalization Layers (3n-1) not perturbed by permutation → Information is shared among neurons**



Model

- **Example 2**

1. First, we build our input data from the raw Athena events:

```
acorn infer data_reader.yaml
```

2. We start the graph construction by training the Metric Learning stage:

```
acorn train metric_learning_train.yaml
```

3. Then, we build graphs using the Metric Learning in inference:

```
acorn infer metric_learning_infer.yaml
```

```yaml
# Model inference parameters
r_infer: 0.1
knn_infer: 1000
```

```yaml
hard_cuts:
  pt: [1000, .inf]

# Model parameters
undirected: True
node_features: [r,    phi,  z]
node_scales:   [1000, 3.14, 1000]
emb_hidden: 1024
nb_layer: 4
emb_dim: 12
activation: Tanh
randomisation: 1
points_per_batch: 50000
r_train: 0.1
knn: 50
knn_val: 1000

# Training parameters
warmup: 5
margin: 0.1
lr: 0.01
factor: 0.7
patience: 10
max_epochs: 100
metric_to_monitor: f1
metric_mode: max
```

Performance