Tracking with ML

11th December 2024

Jeremy Couthures

Reconstructed 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

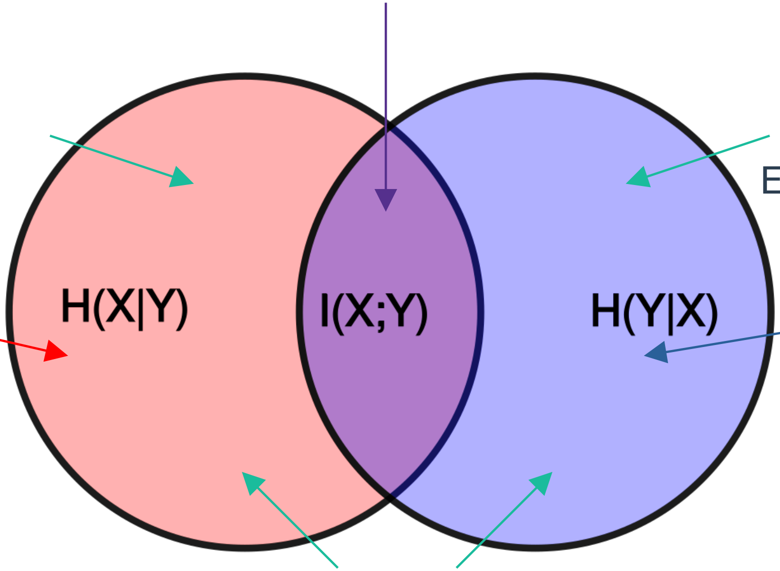
Mutual Information:
$H(X) \cap H(Y)$

$\approx$ $Y = f(X)$
$X = g(Y)$

H(X)

H(Y)
Entropy

Conditional entropy $H(X|Y)$:
$H(X) \setminus H(Y)$



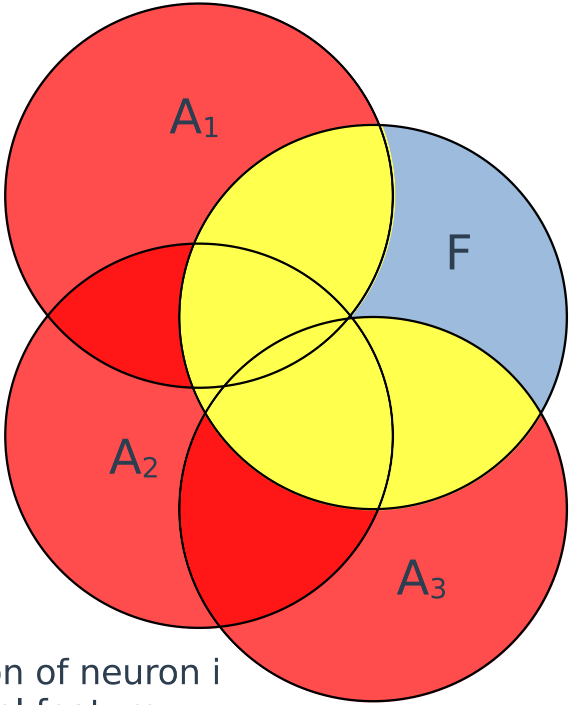
Conditional entropy $H(Y|X)$:
$H(Y) \setminus H(X)$

H(X,Y)
Joint entropy:
$H(X) \cup H(Y)$

Some properties of entropy

- $H(X)$ is an expected value, only cares about probabilities
- $H(X)$ is maximum when X follows a uniform law
- $\max(H(X), H(Y)) \leq H(X,Y) \leq H(X) + H(Y)$
- $H(Y|X) \leq H(Y)$
- $I(X;X) = H(X)$
- $0 \leq I(X;Y) \leq \min(H(X), H(Y))$
- Equalities when:
  - $X=Y$
  - X and Y are independent

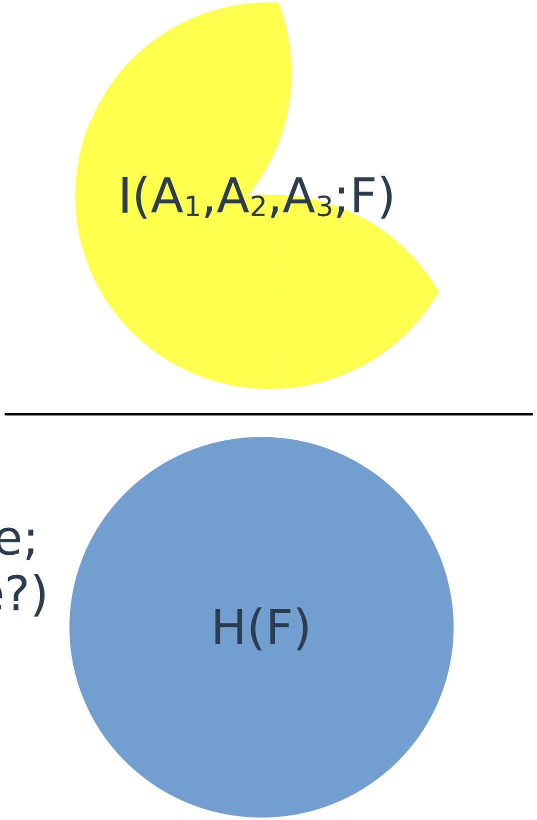
Proficiency



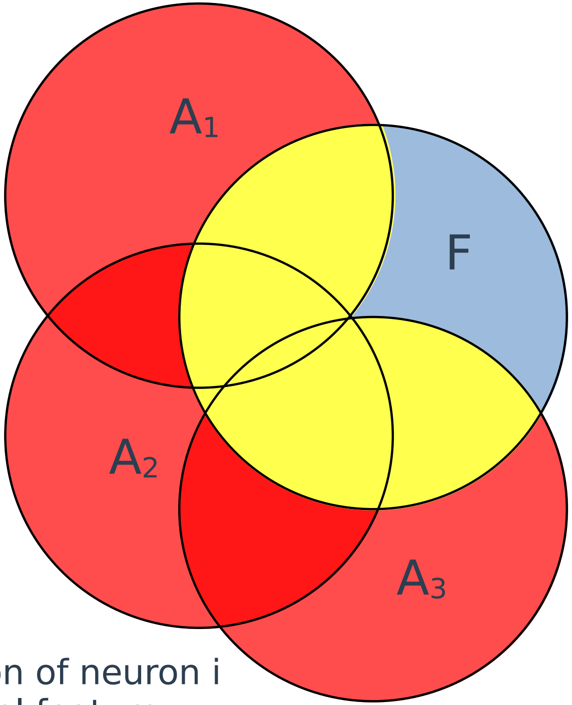
Aᵢ: attention of neuron i
F: high-level feature

Proficiency =

(= what I called information coverage; can I keep the name?)



Proficiency



A_i: attention of neuron i
F: high-level feature

$$\text{Proficiency} = \frac{H(F) - H(F|A_1,A_2,A_3)}{H(F)}$$

Computing entropy

- **Discrete (Shannon):**

$$H[X] = -\sum_{i=1}^{n} P(x_i) \log P(x_i)$$

- **Continuous (Shannon):**

$$H(\mathbf{y}) = -\int f(\mathbf{y}) \log f(\mathbf{y}) \mathrm{d}\mathbf{y}$$

- **Continuous (Edwin Thompson Jaynes):**
  Limiting density of discrete points

$$\lim_{N \to \infty} H_N(X) = \log(N) - \int p(x) \log \frac{p(x)}{m(x)} \, dx.$$

Computing entropy: continuous issue

- **Continuous (Shannon):** $H(\mathbf{y}) = -\int f(\mathbf{y}) \log f(\mathbf{y}) \mathrm{d}\mathbf{y}$
  - not invariant under a change of variables (Ex: H(y) != H(2*y))
  - can become negative (Ex: uniform between 0 and ½)
  - not dimensionally correct ([f(y)] = [1/dy] → log([f(y)]) = log([1/dy]))
- **Quantization:**

continuous function f discretised into bins of size Δ

$$\int_{-\infty}^{\infty} f(x)\mathrm{d}x = \lim_{\Delta \to 0} \sum_{i=-\infty}^{\infty} \Delta f(x_i)$$

Shannon entropy of the discretised density H_Δ

$$H_\Delta := -\sum_{i=-\infty}^{\infty} \Delta f(x_i) \log(f(x_i)) - \sum_{i=-\infty}^{\infty} \Delta f(x_i) \log(\Delta)$$

$$\lim_{\Delta \to 0} H_\Delta = \underbrace{-\int_{-\infty}^{\infty} f(x) \log(f(x))\mathrm{d}x}_{\text{differential entropy}} \underbrace{-\log(0)}_{\text{infinty offset}}$$

Fix →

$$\underbrace{-\int_{-\infty}^{\infty} f(x) \log(f(x))\mathrm{d}x}_{h(X)} = -\lim_{\Delta \to 0} \sum_{i=-\infty}^{\infty} \Delta f(x_i) \log(f(x_i)) = H_\Delta + \log(\Delta)$$

Implementation tests

- **Multivariate normal distributions: 10 000 points**

  Analytical solution: $h(x) = \frac{n}{2}\ln(2\pi) + \frac{1}{2}\ln|\Sigma| + \frac{1}{2}n$

  n is the dimension

N	Theory	Histogram	KDE + Histogram
3	3.91	3.82	4.02
4	5.09	4.43	5.21
5	6.26	4.30	6.30
6	7.41	3.68	7.23
13	18.45	-3.21	18.43

Bias

KDE + Histogram:

KDE Fit the points (**gaussian** kernel)
→ 10^{10} samples from KDE
Histogram from the 10^{10} samples

= Histogram(KDE(10 000), 10^{10})

20 bins are use for histograms

Computing entropy: continuous fix

- **Continuous (Edwin Thompson Jaynes):**
  Limiting density of discrete points

$$\lim_{N\to\infty} \frac{1}{N} (\text{number of points in } a < x < b) = \int_a^b m(x)\, dx$$

$$\lim_{N\to\infty} H_N(X) = \log(N) - \int p(x) \log \frac{p(x)}{m(x)}\, dx.$$

- Invariant with change of variables
- Positive
- Dimensionally correct

Relative entropy:

$$D(p \parallel m) = \int p(x) \log \frac{p(x)}{m(x)}\, dx$$

KNN estimation

- **Estimate mutual information**

$$||z - z'|| = \max\{||x - x'||, ||y - y'||\}$$

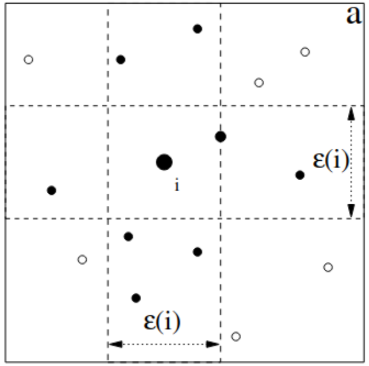
$$\langle \ldots \rangle = N^{-1} \sum_{i=1}^{N} \mathsf{E}[\ldots(i)]$$

$$I^{(1)}(X,Y) = \psi(k) - \langle \psi(n_x+1) + \psi(n_y+1) \rangle + \psi(N)$$

Here, $\psi(x)$ is the digamma function, $\psi(x) = \Gamma(x)^{-1} d\Gamma(x)/dx$. It satisfies the recursion $\psi(x+1) = \psi(x) + 1/x$ and $\psi(1) = -C$ where $C = 0.5772156\ldots$

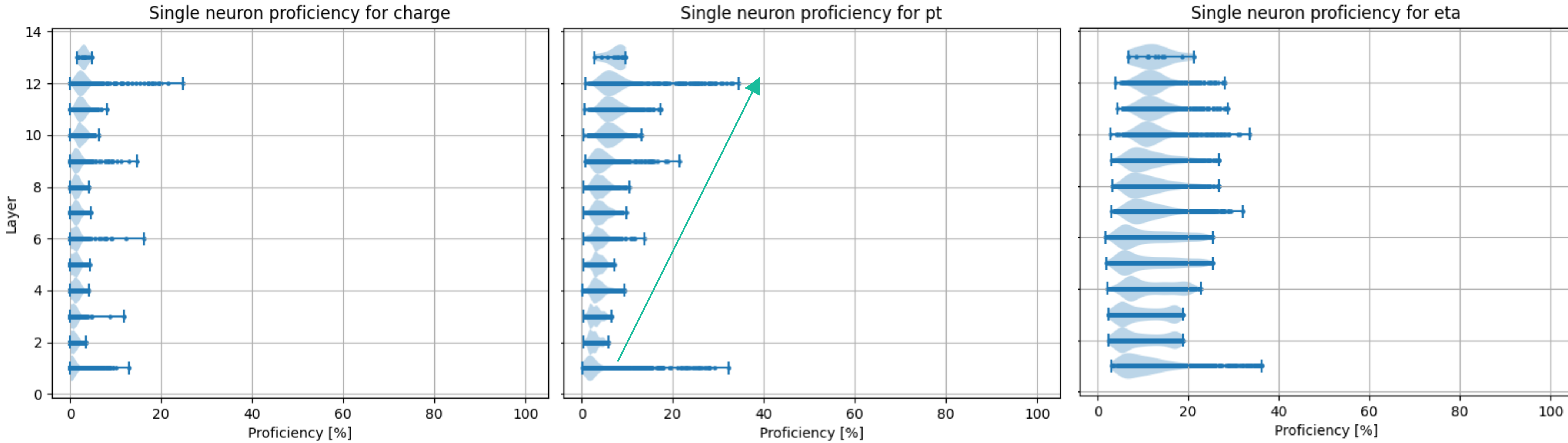
Let us denote by $\epsilon(i)/2$ the distance from z_i to its k-th neighbour, and by $\epsilon_x(i)/2$ and $\epsilon_y(i)/2$ the distances between the same points projected into the X and Y subspaces. Obviously, $\epsilon(i) = \max\{\epsilon_x(i), \epsilon_y(i)\}$.

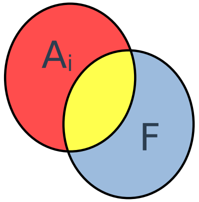
In the first algorithm, we count the number $n_x(i)$ of points x_j whose distance from x_i is strictly less than $\epsilon(i)/2$, and similarly for y instead of x.



Used by scikit-learn

High-level variables single neuron

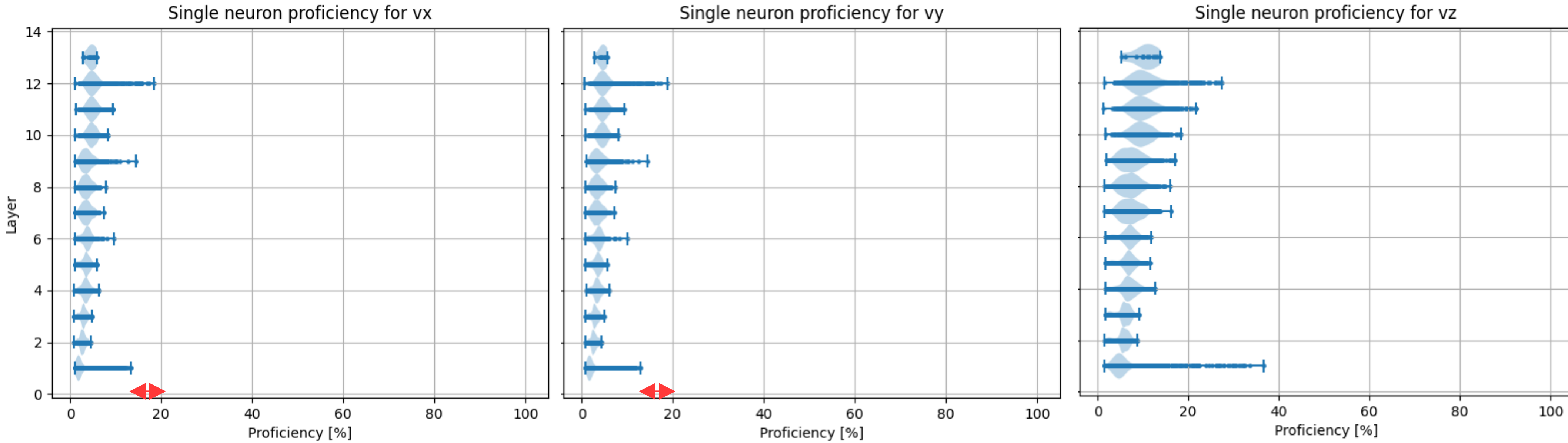


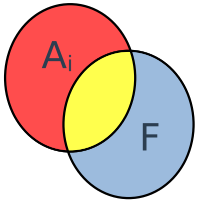


From Scikit-learn Mutual information calculation

1 event: 14183 hits

High-level variables single neuron



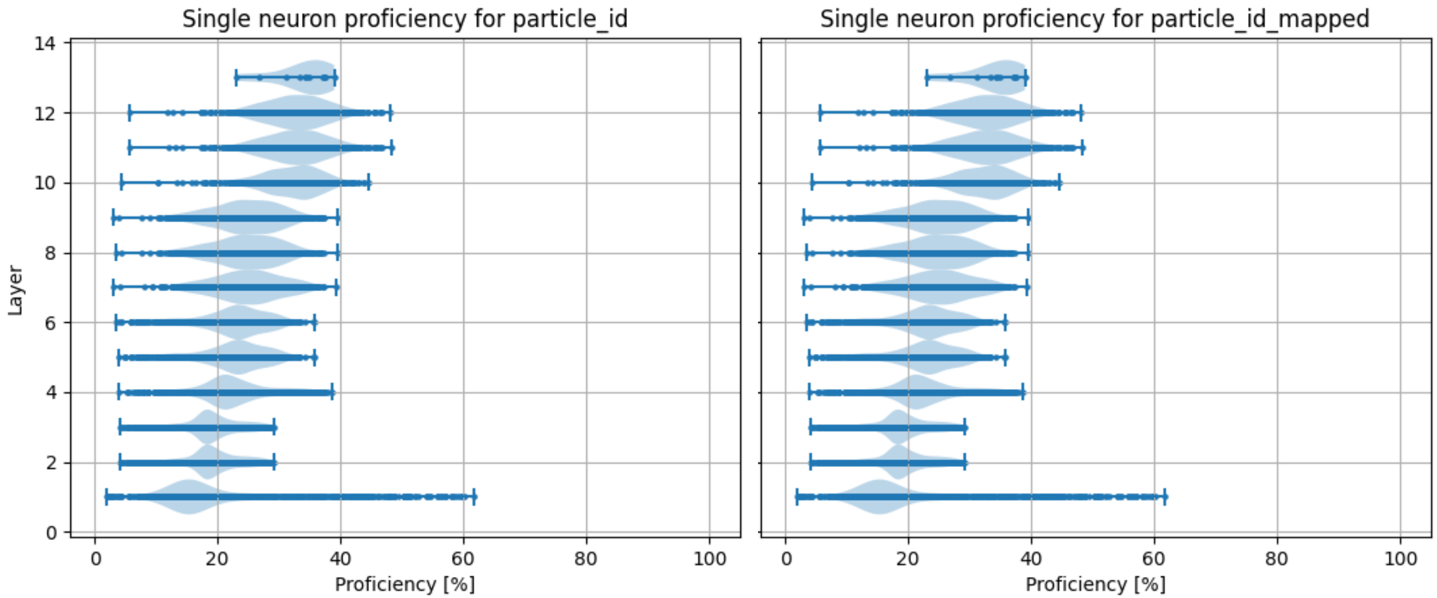


Same

1 event: 14183 hits

From Scikit-learn Mutual information calculation

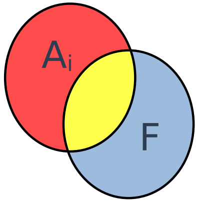
Change of variable impact



Should be close to 0:
Any particle can
hit any cell

Indicates not enough
data

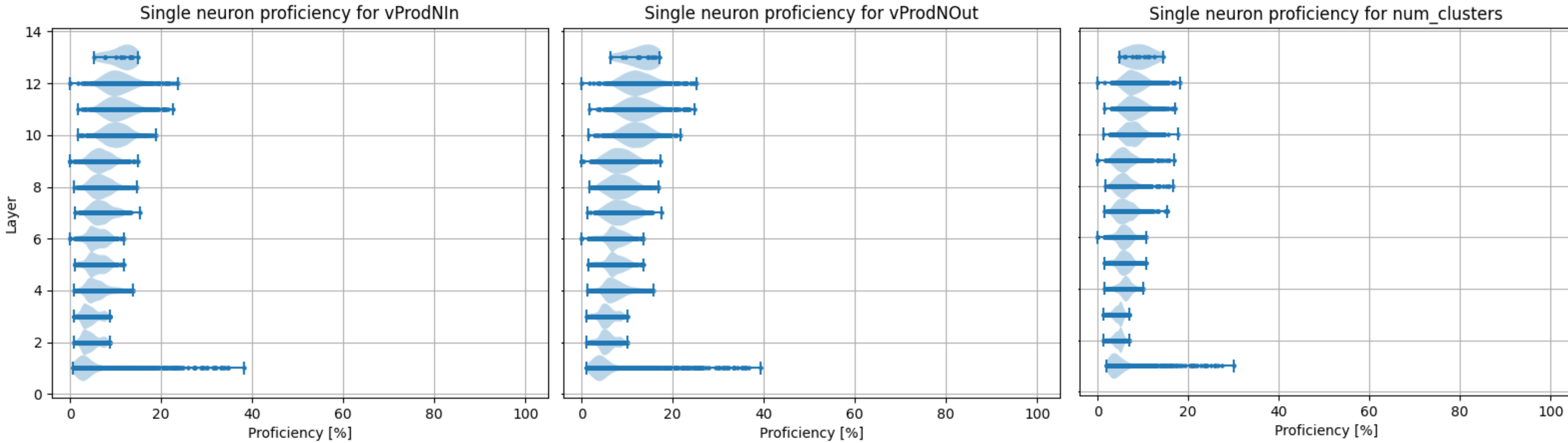
Same

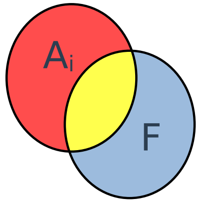


From Scikit-learn Mutual information calculation

1 event: 14183 hits

High-level variables single neuron

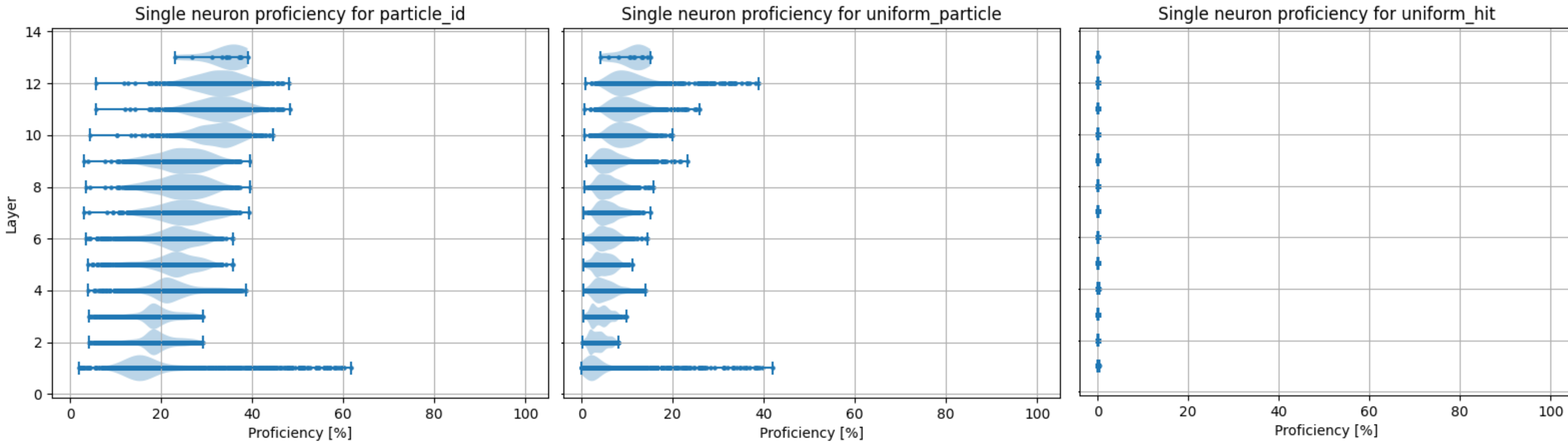


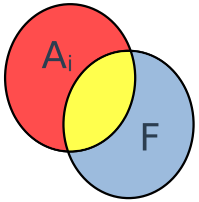


From Scikit-learn Mutual information calculation

1 event: 14183 hits

Random variables single neuron: uniform

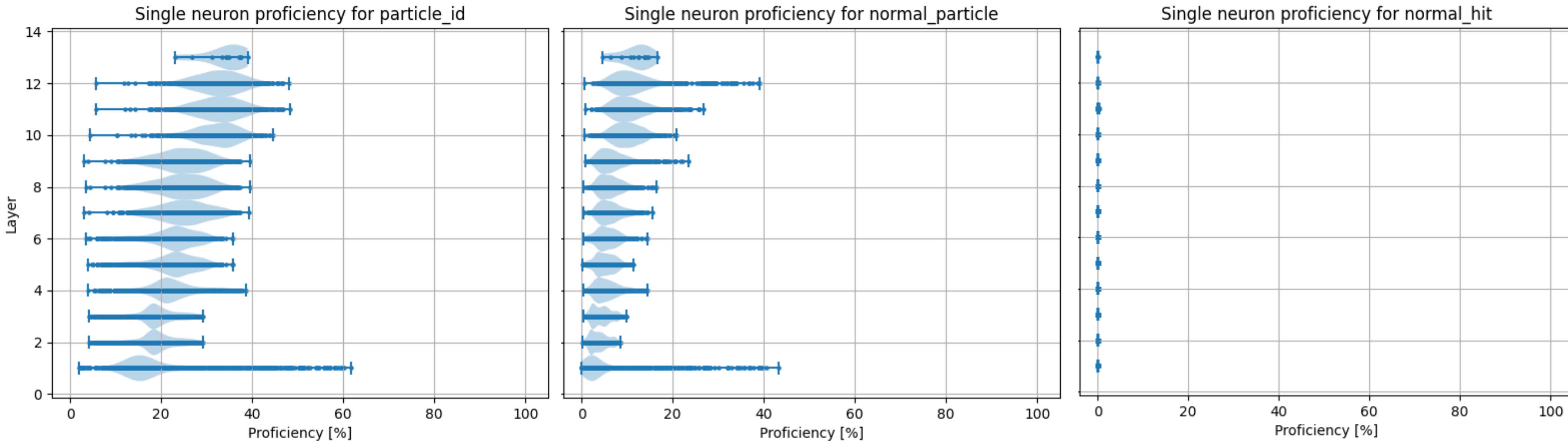


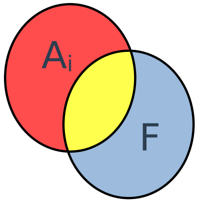


From Scikit-learn Mutual information calculation

1 event: 14183 hits

Random variables single neuron: normal

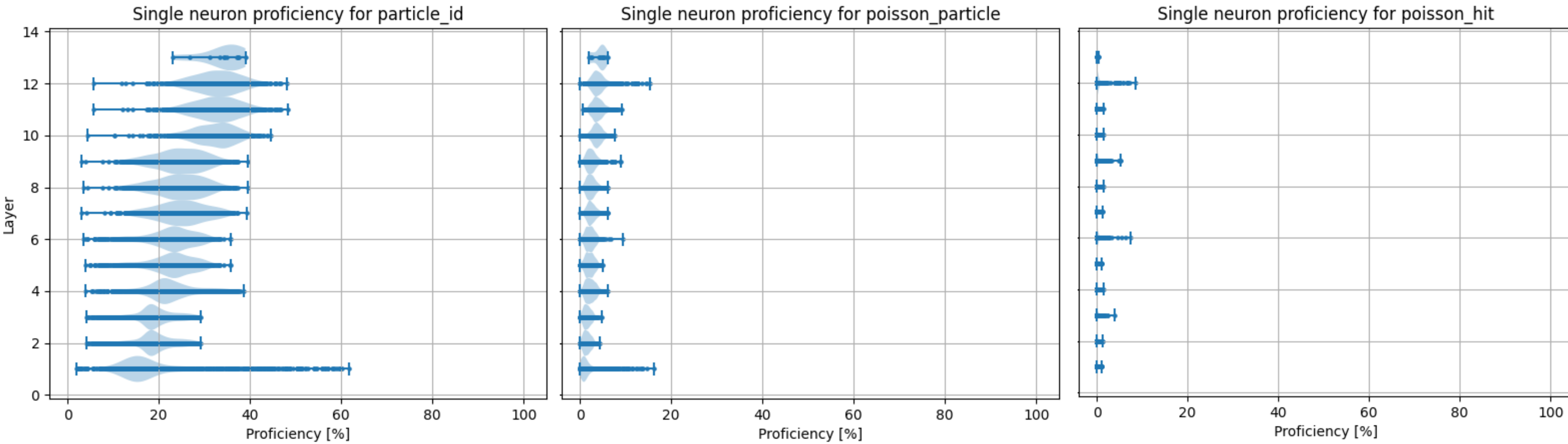


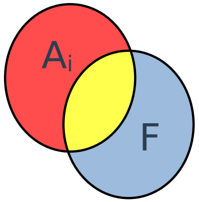


From Scikit-learn Mutual information calculation

1 event: 14183 hits

Random variables single neuron: poisson





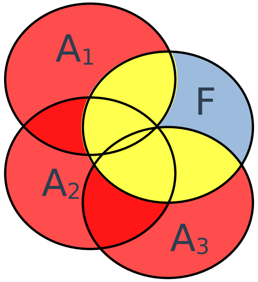
From Scikit-learn Mutual information calculation

1 event: 14183 hits

Layer proficiency

- **Scikit-learn do only single dimensional X and Y**

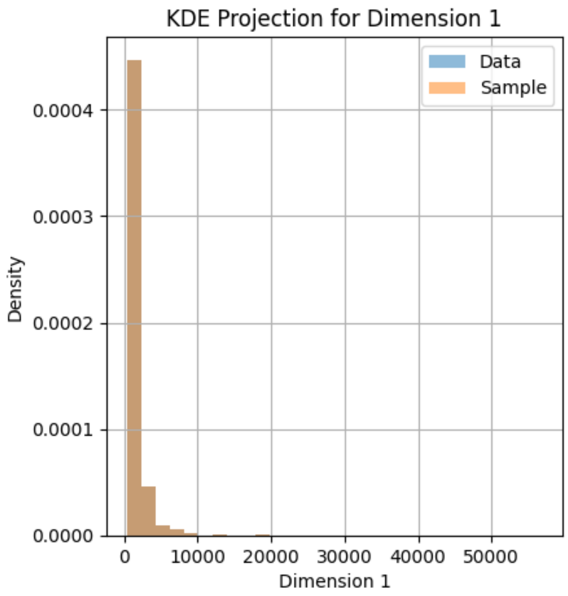
Theory: To estimate the joint MI between $\{X_1, X_2, \ldots, X_m\}$ and Y, the high-dimensional variables $\{X_1, X_2, \ldots, X_m\}$ should be treated as a whole and n_x would be defined as the number of points in the m-dimensional space.

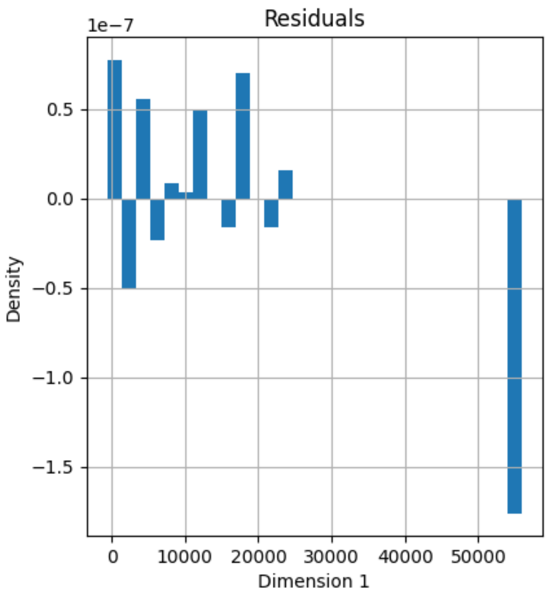


BACKUP

Goodness of fit

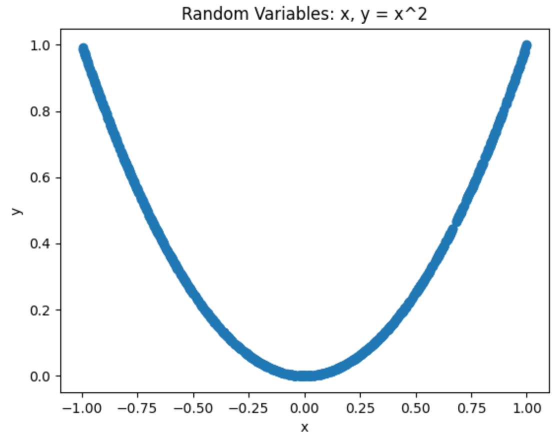
- **Sample from the fit KDE**





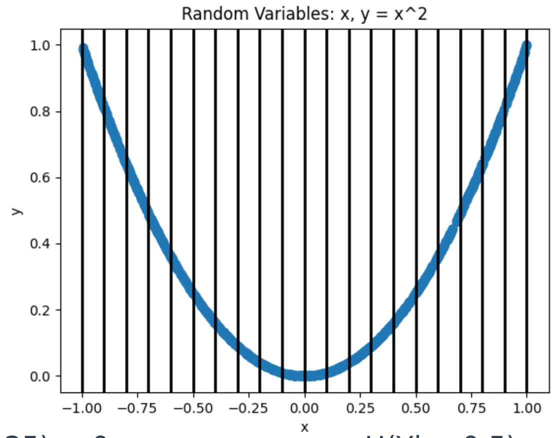
Conditional entropy

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



Conditional entropy

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

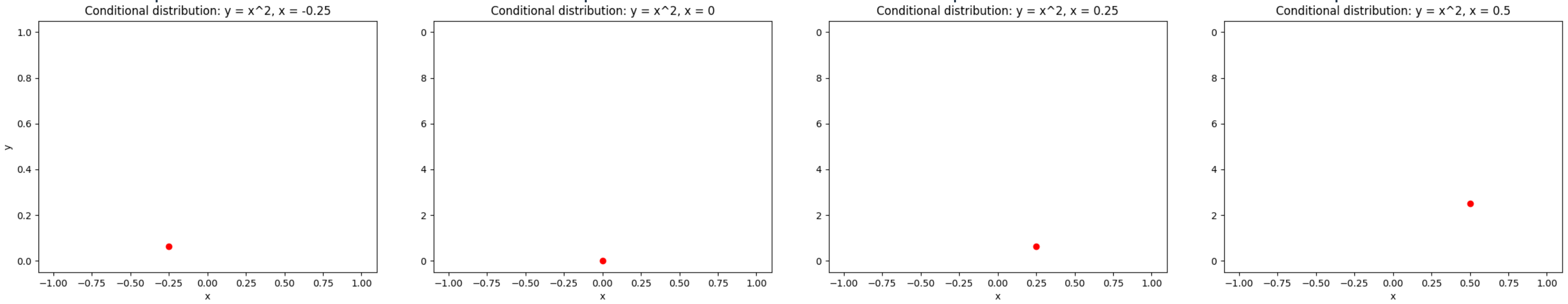


H(Y|x=-0.25) = 0

H(Y|x=0) = 0

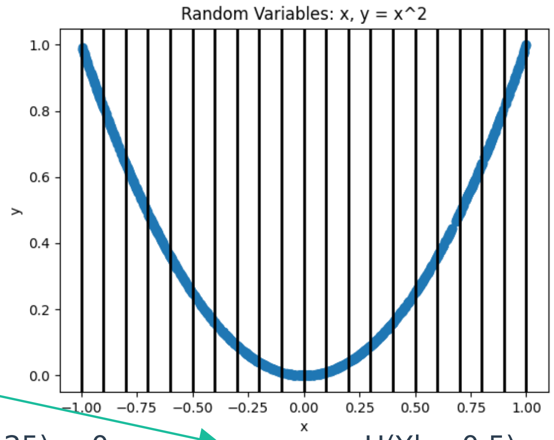
H(Y|x=0.25) = 0

H(Y|x=0.5) = 0



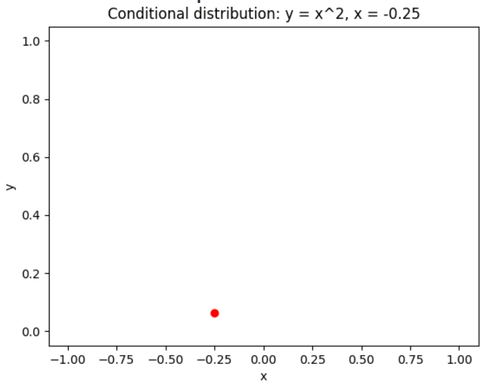
Conditional entropy

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

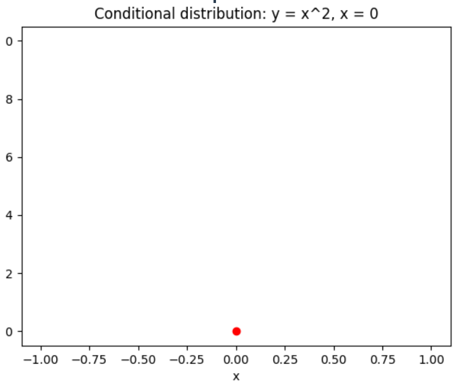


H(Y|X) = 0

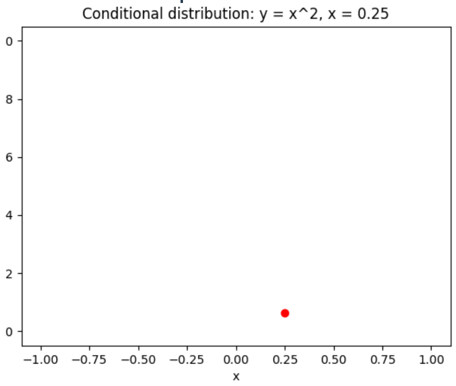
H(Y|x=-0.25) = 0



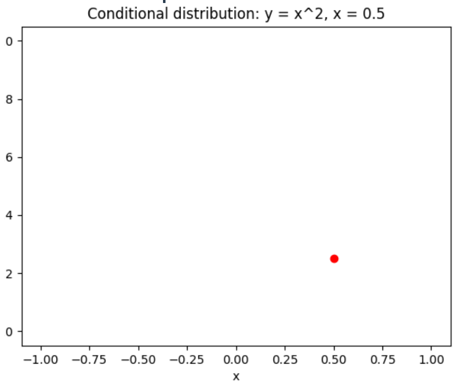
H(Y|x=0) = 0



H(Y|x=0.25) = 0



H(Y|x=0.5) = 0



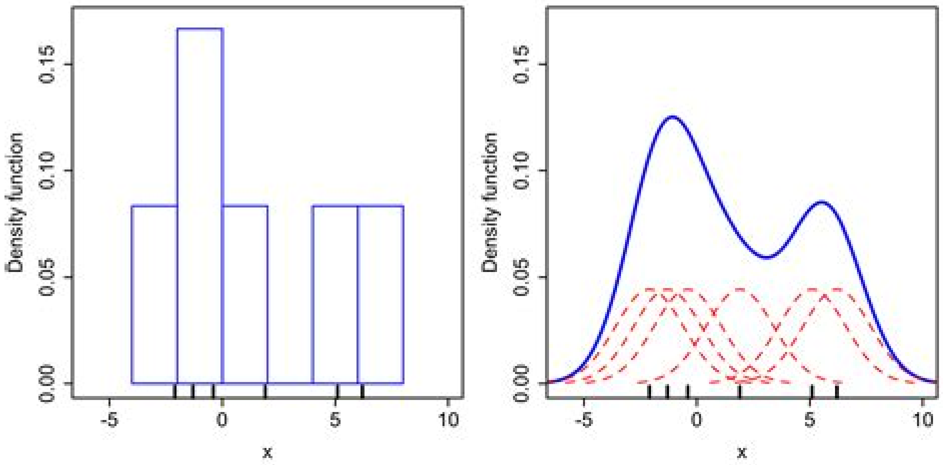
Kernel Density Estimation

- **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'`)

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

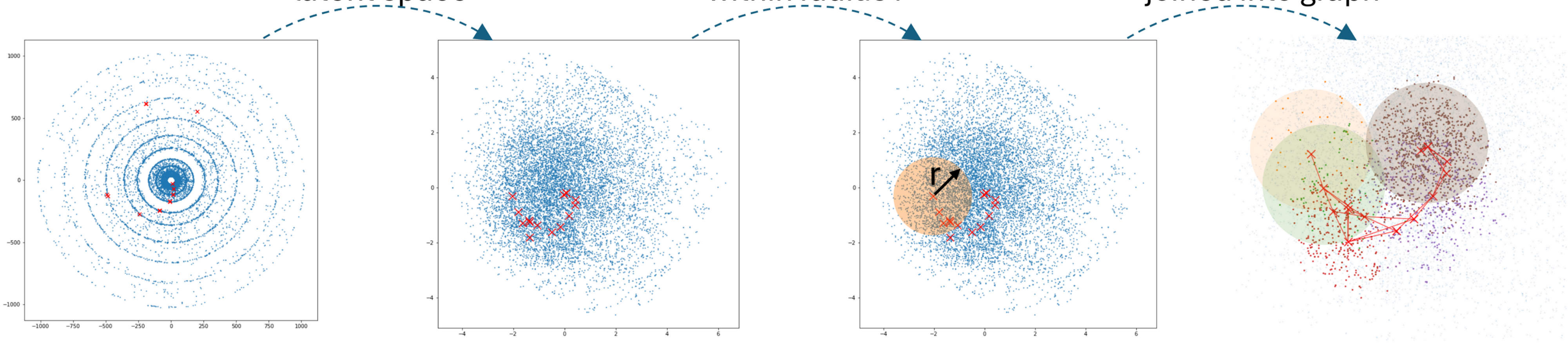


GNN Metric Learning

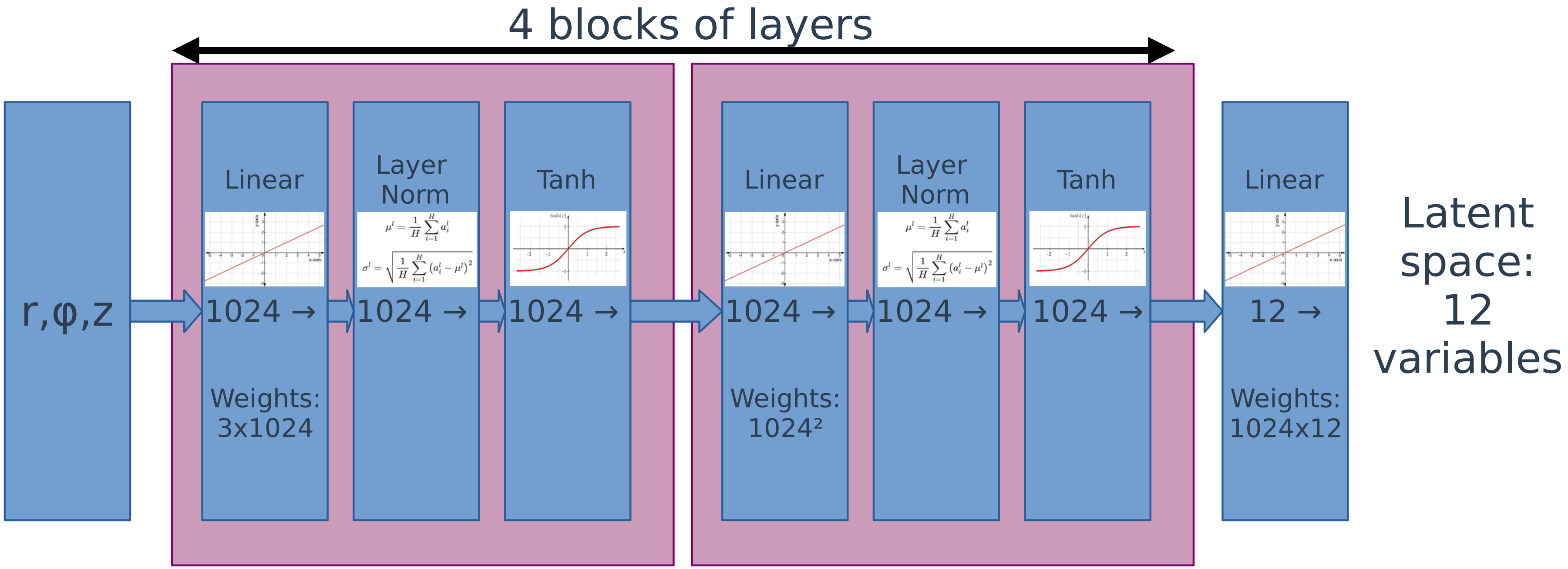
Embed into learned latent space

Connect all space points within radius r

All space point pairs joined into graph

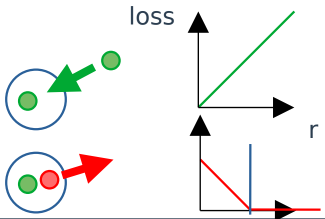


Architecture



Hinge Loss

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



Performance