UNIVERSITÄT BONN

Self-Organizing Maps (SOM) in High Energy Physics

27.04.2022

Kai Habermann, Eckhard von Toerne

Maschine Learning

Supervised

Unsupervised

Classification

Dimensionality reduction
PCA

Clustering
Kmeans

ANN, MLP, BDT, …
Trained with MC

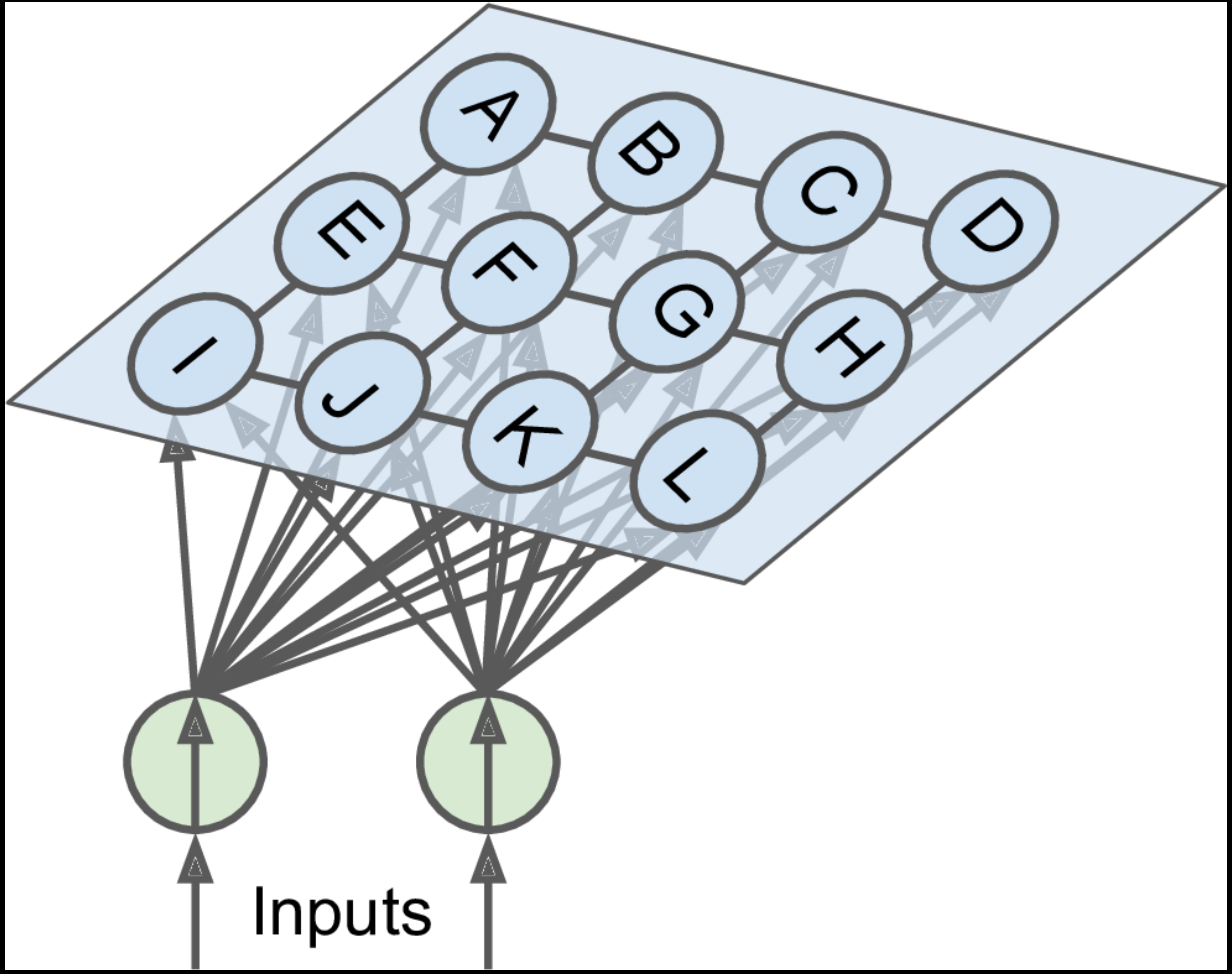
Self-Organizing Maps

Why SOM in Physics?

- Visualization
- Data driven Analysis
  - Simulation independent data exploration
  - Problem: complex data
- Clustering for feature detection
- Aid in the search for rare processes

What is SOM?

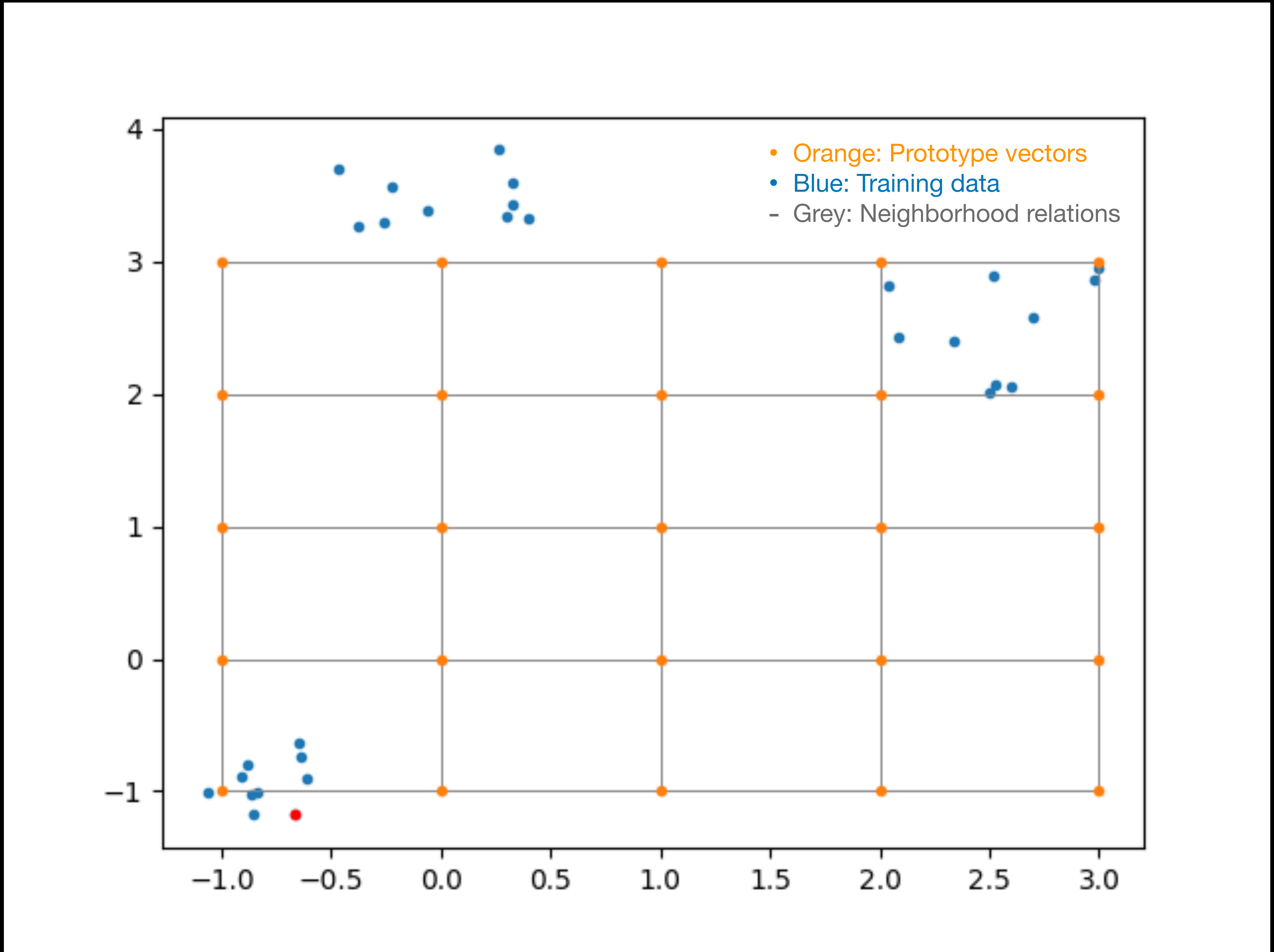
- Input Layer:
  - Size of input vectors
- Output Layer:
  - Layer of neurons organized in a 2 dimensional lattice
- Prototype vector the size of the input for each neuron of output layer
- Topology preservation via ordering of neurons in lattice
- Mapping via distance to Prototype vectors



„Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems", Aurelien Geron

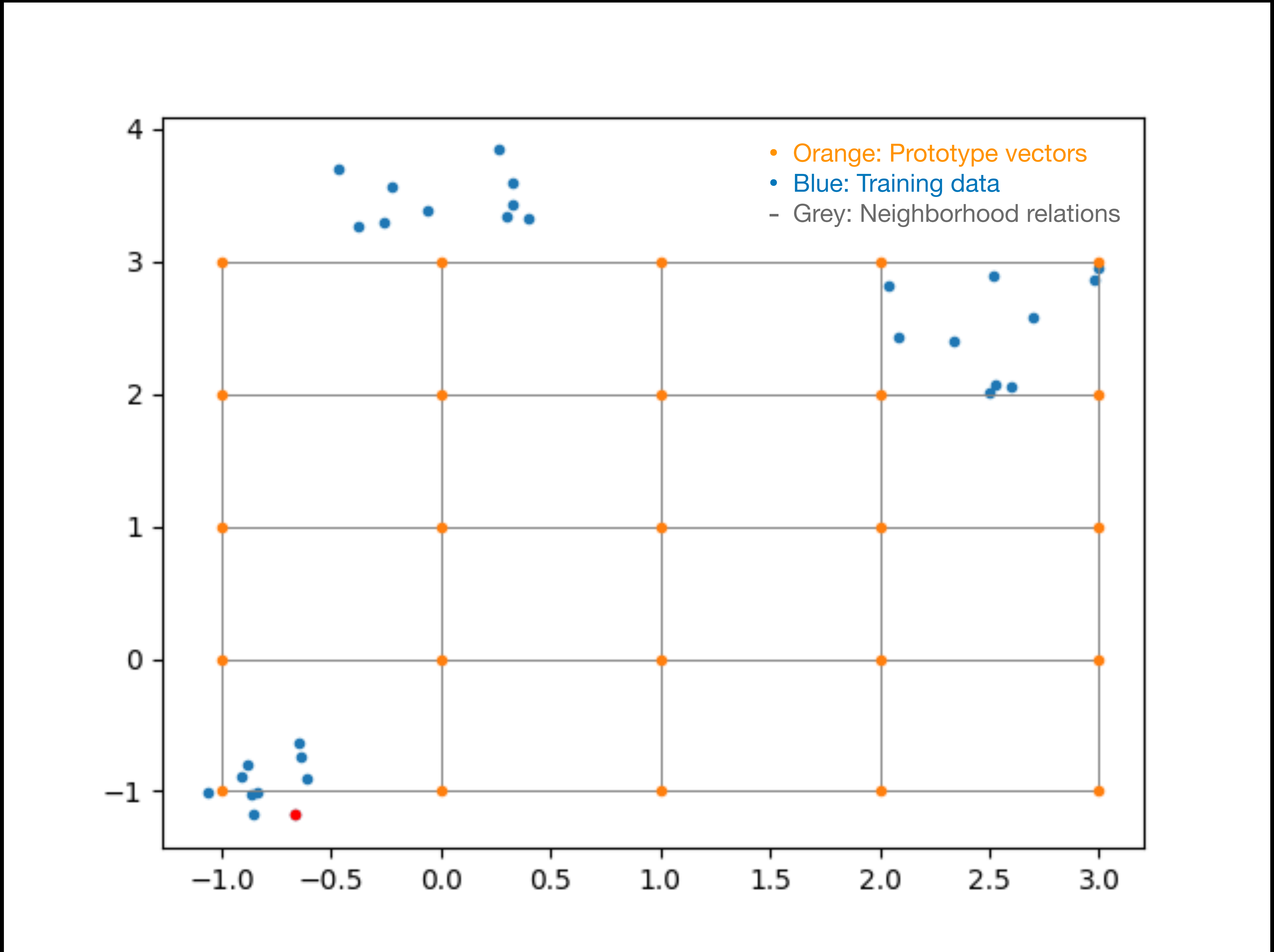
Training

- Goal:
  - Find best prototype vectors
  - Preserve Topology and distance relations
- Method:
  - Competitive + collaborative
  - Neurons compete for inputs + update neighbors [1]



Training

- Goal:
  - Find best prototype vectors
  - Preserve Topology and distance relations
- Method:
  - Competitive + collaborative
  - Neurons compete for inputs + update neighbors [1]



Data Sample

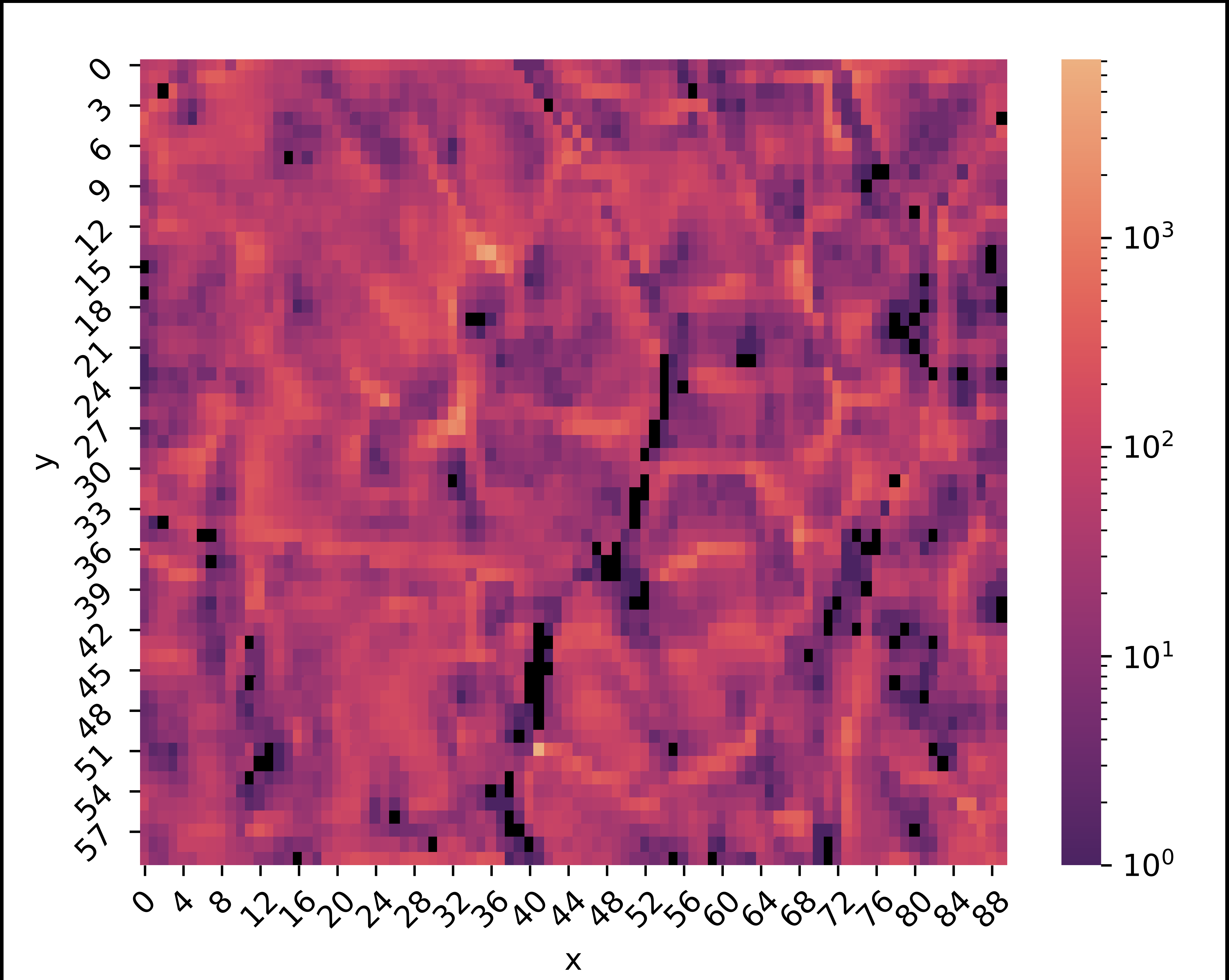
ATLAS Open Data[1] with $\sqrt{s} = 13\,\mathrm{TeV}$

- Openly available dataset from ATLAS (meant for educational use)
- We chose electron-muon dilepton final states + $N_{jet} \geq 0$ (500k events)
  - Contributions from $t\bar{t}$, $Z \to \tau\bar{\tau}$, WW and Higgs
- Remove 77 events with energies of more then $13\,\mathrm{TeV}$
- Opposite charge for leptons
- $m_T^{ll} > 70\,\mathrm{GeV}$, Isolation < 0.1 (applied only after training)
- Quantile transform to pull outliers in

[1]http://opendata.atlas.cern/release/2020/documentation/index.html

Mapped out data

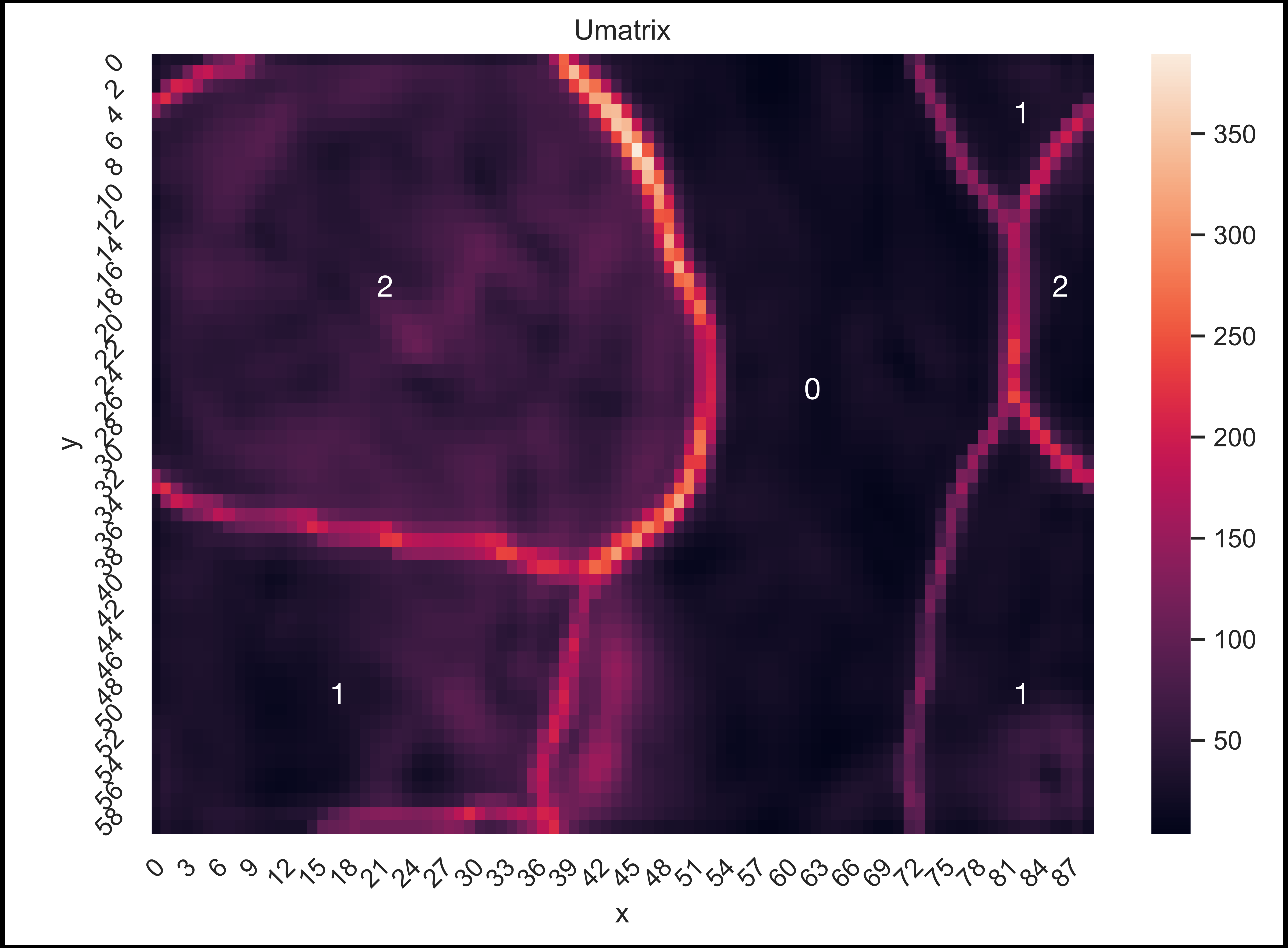
- SOM with 60x90 neurons
- Training with batches of 500 datapoints
- Each pixel represents a single neuron
- z-axis shows events per neuron



Unified Distance Matrix (U-Matrix)

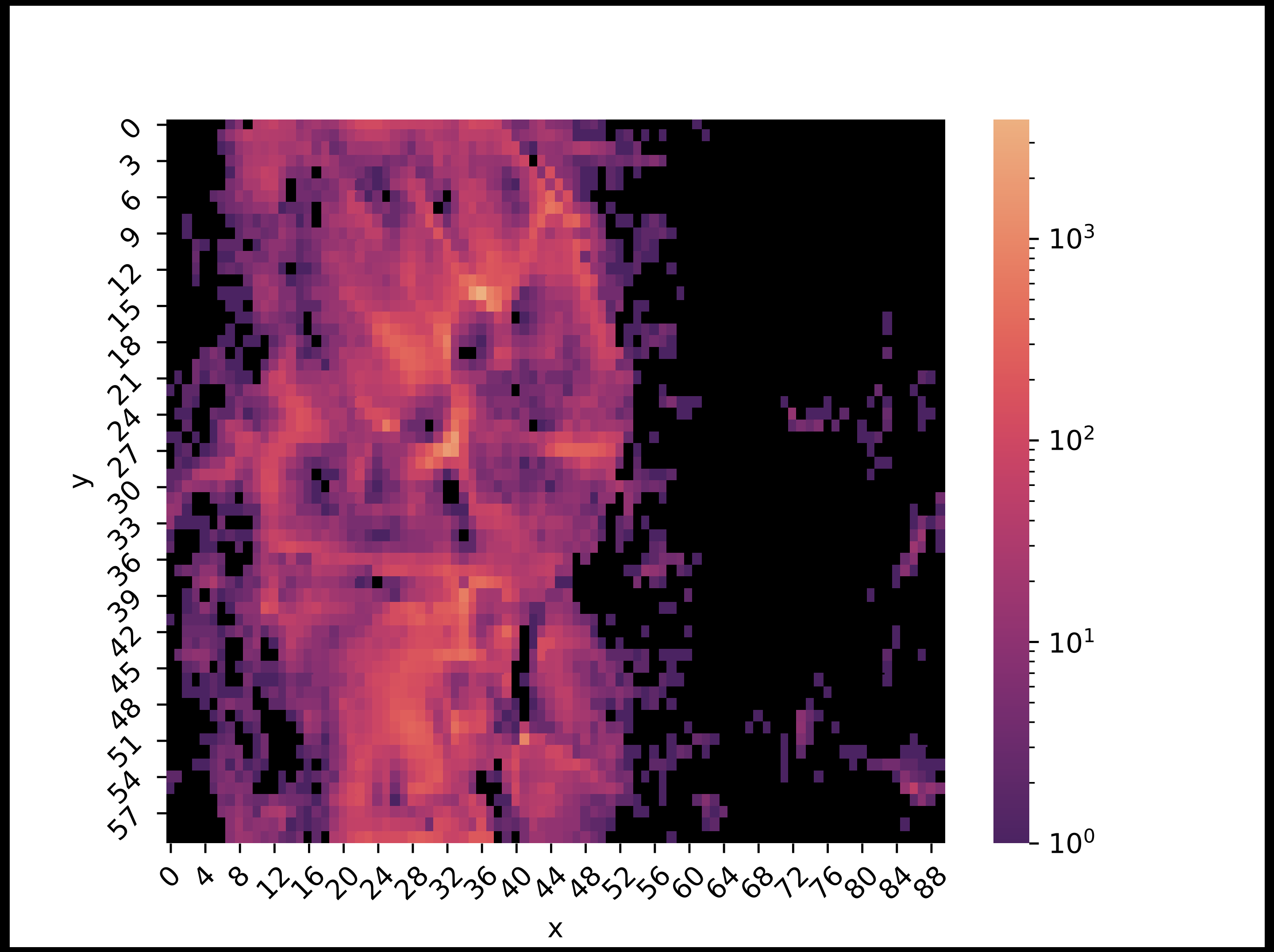
$$u_{ij} = \sum_{k=i-1}^{i+1} \sum_{m=j-1}^{j+1} |w_{km} - w_{ij}|$$

- w_{ab} are weight vectors of the neurons.
- Clusters of different amounts of jets
- 0: No jets
- 1: One jet
- 2: 2 or more jets

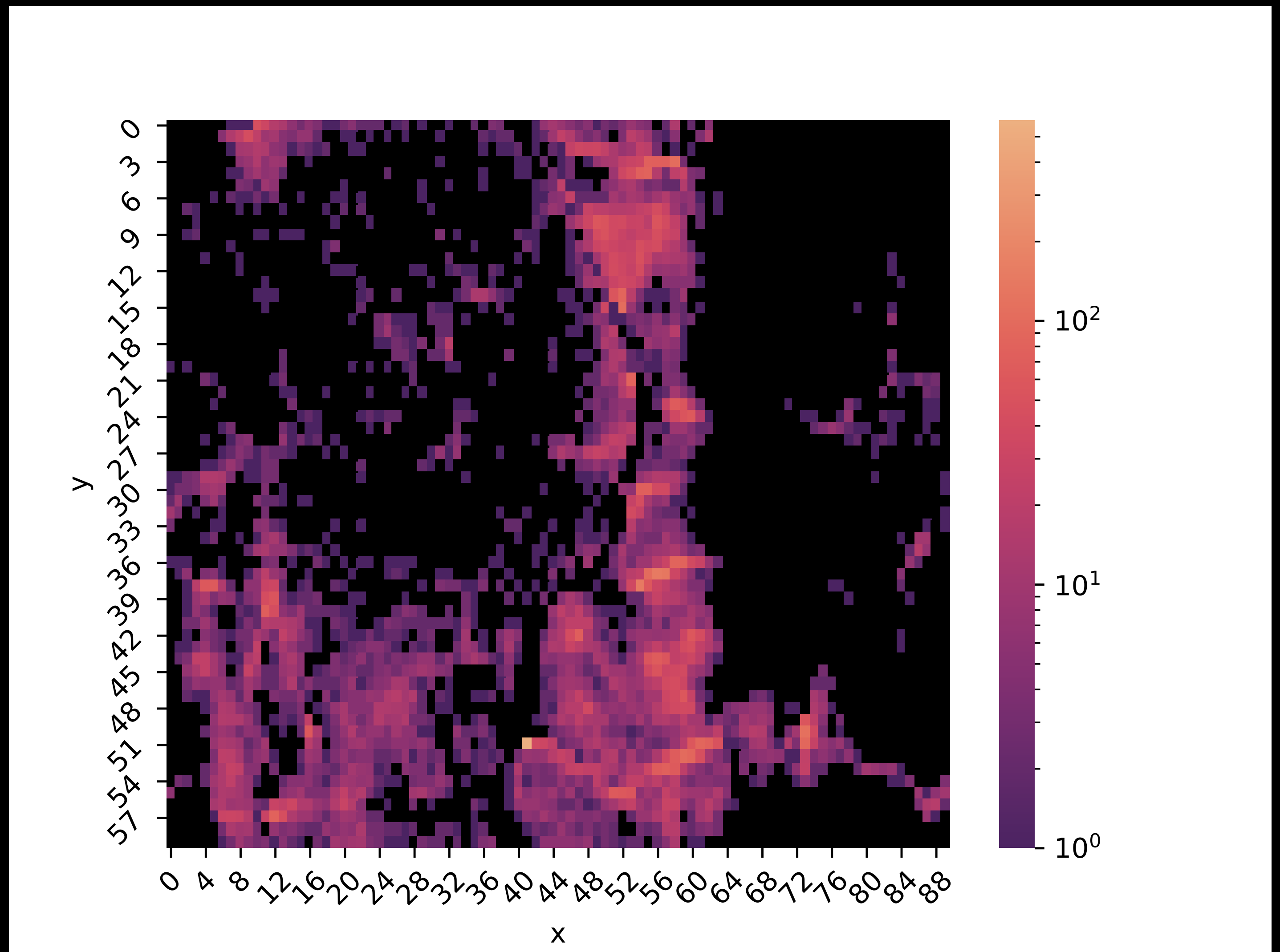


Mapped MC data (m_T^{ll} cut included)

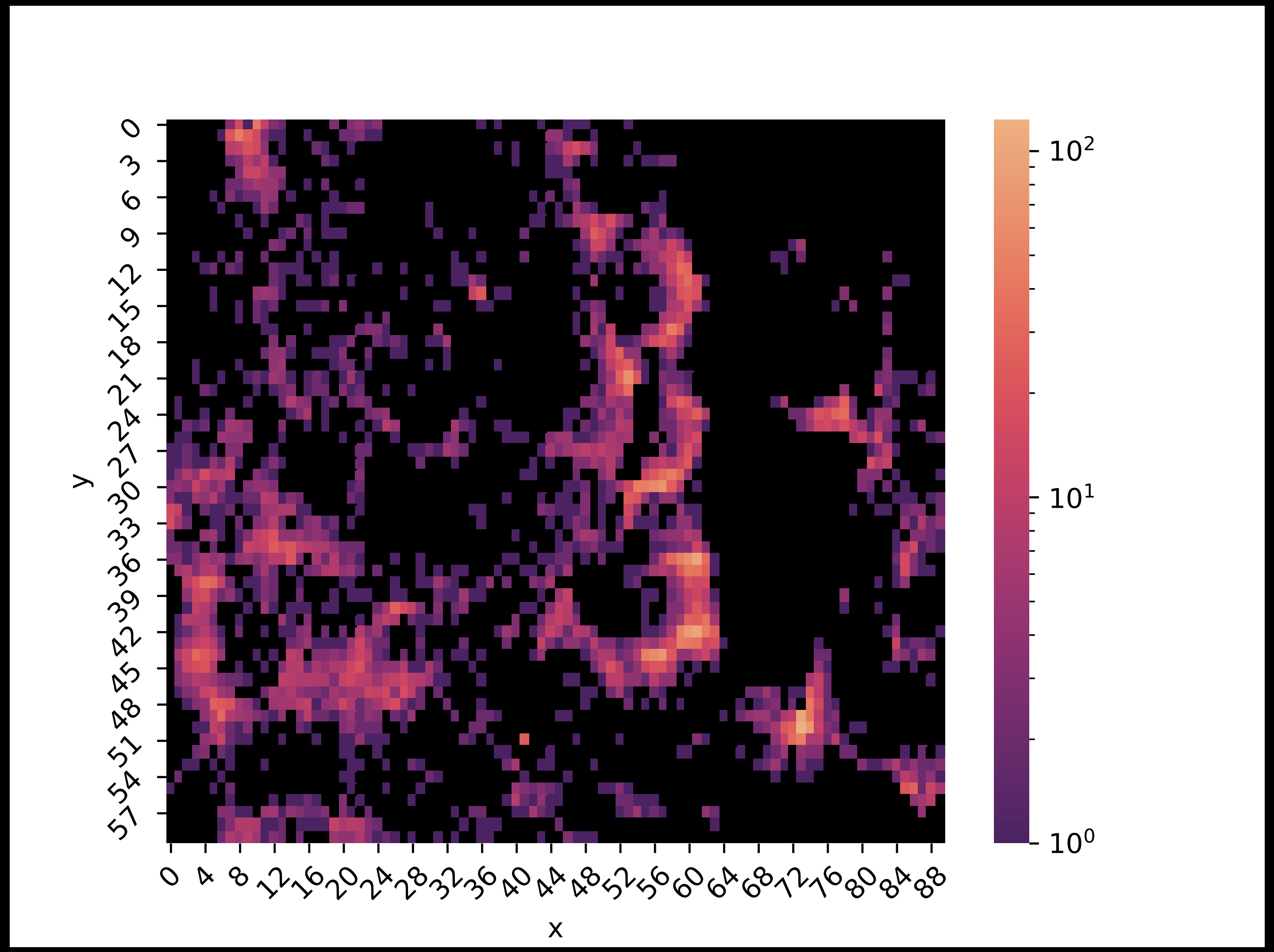
Single Top



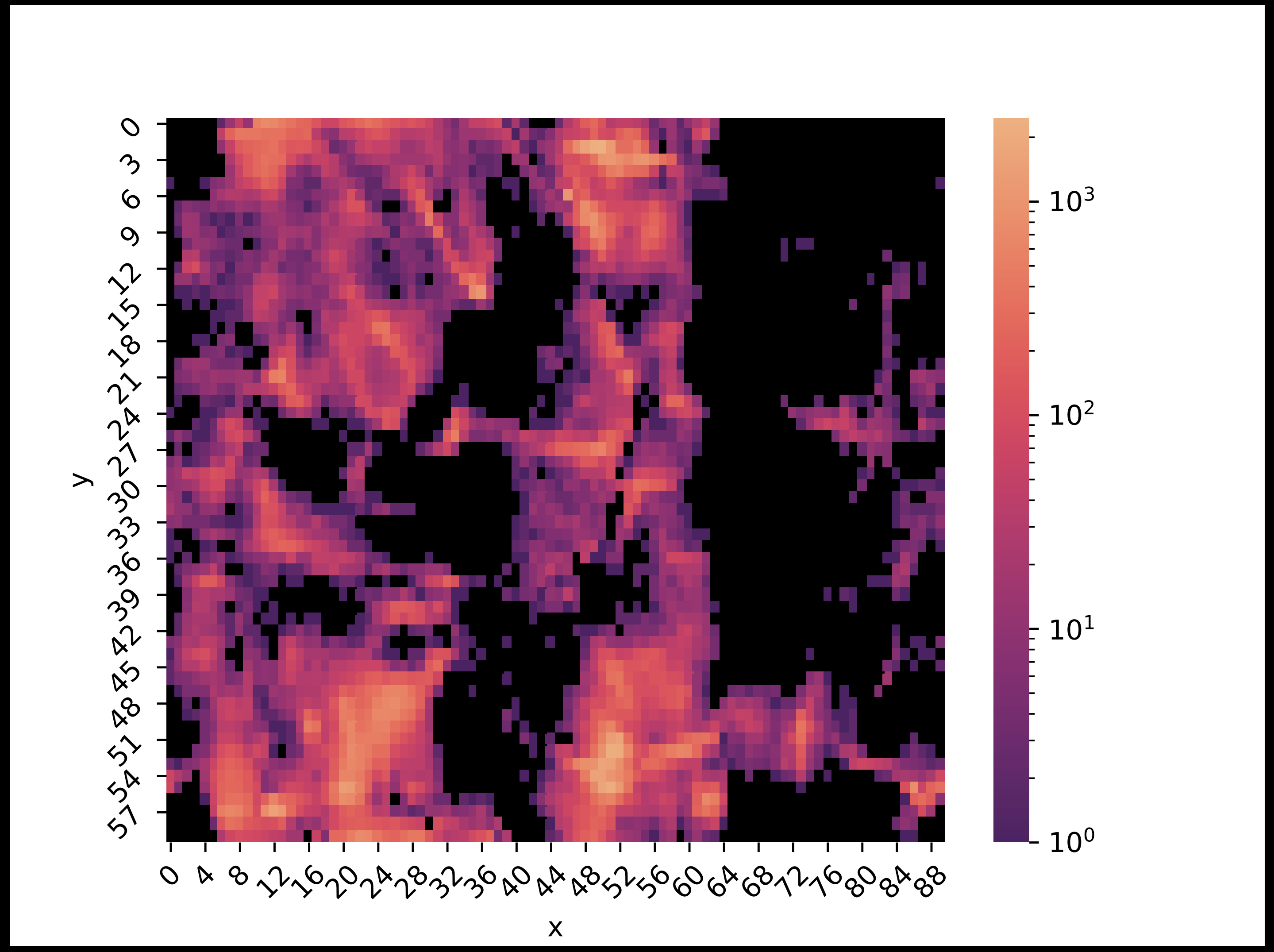
Single W



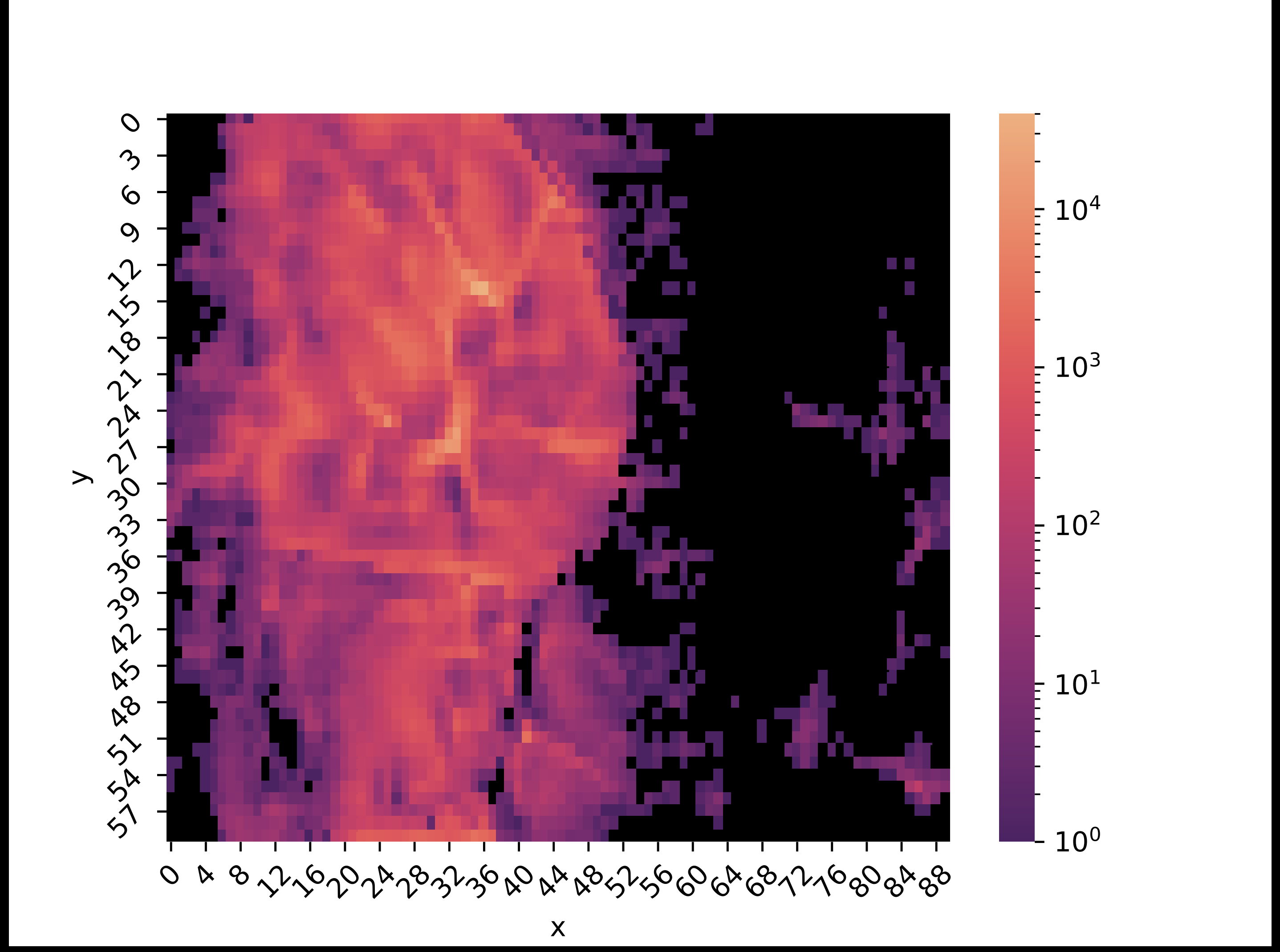
$Z \to \tau\bar{\tau}$



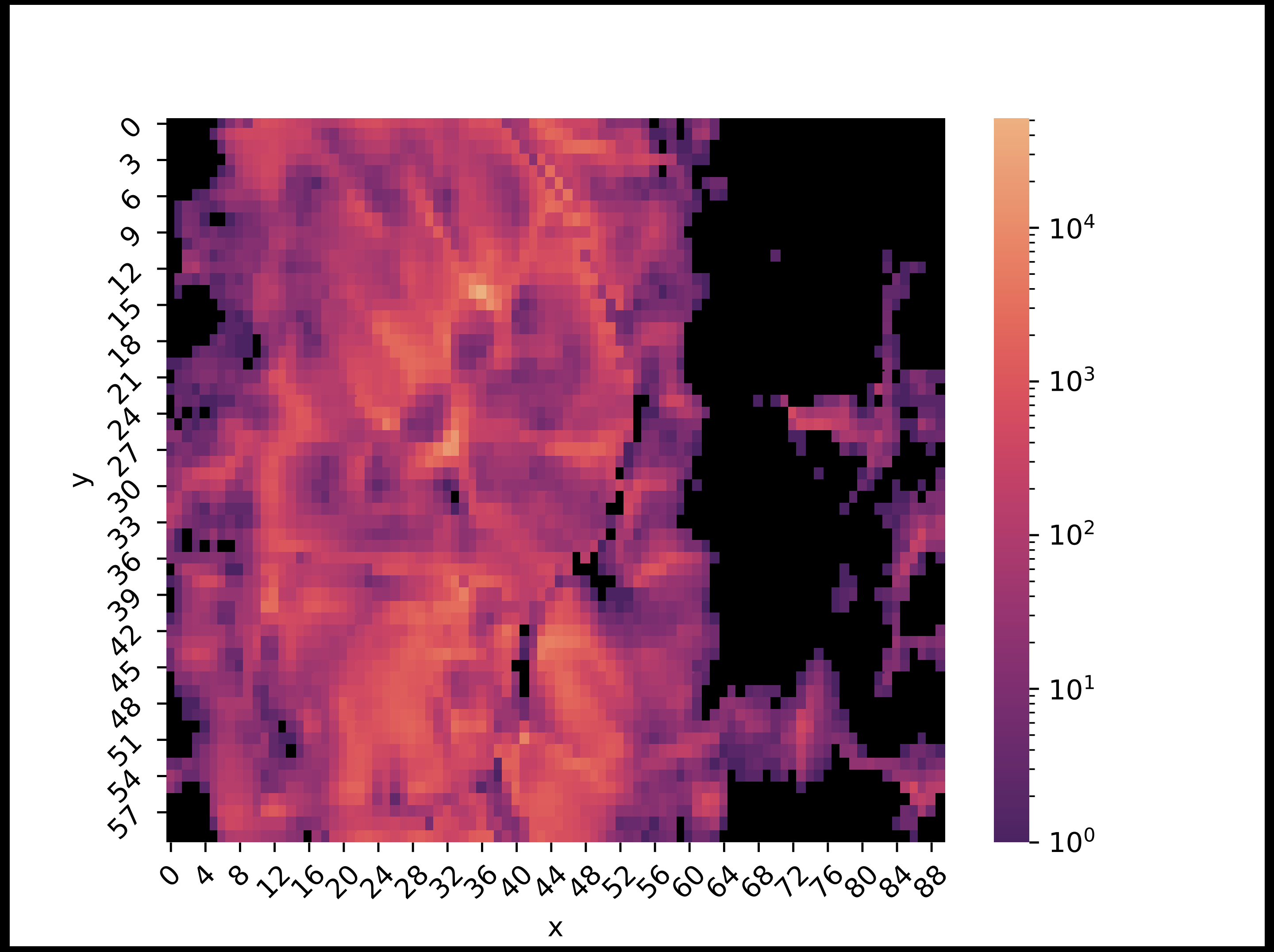
$H \to WW^*$



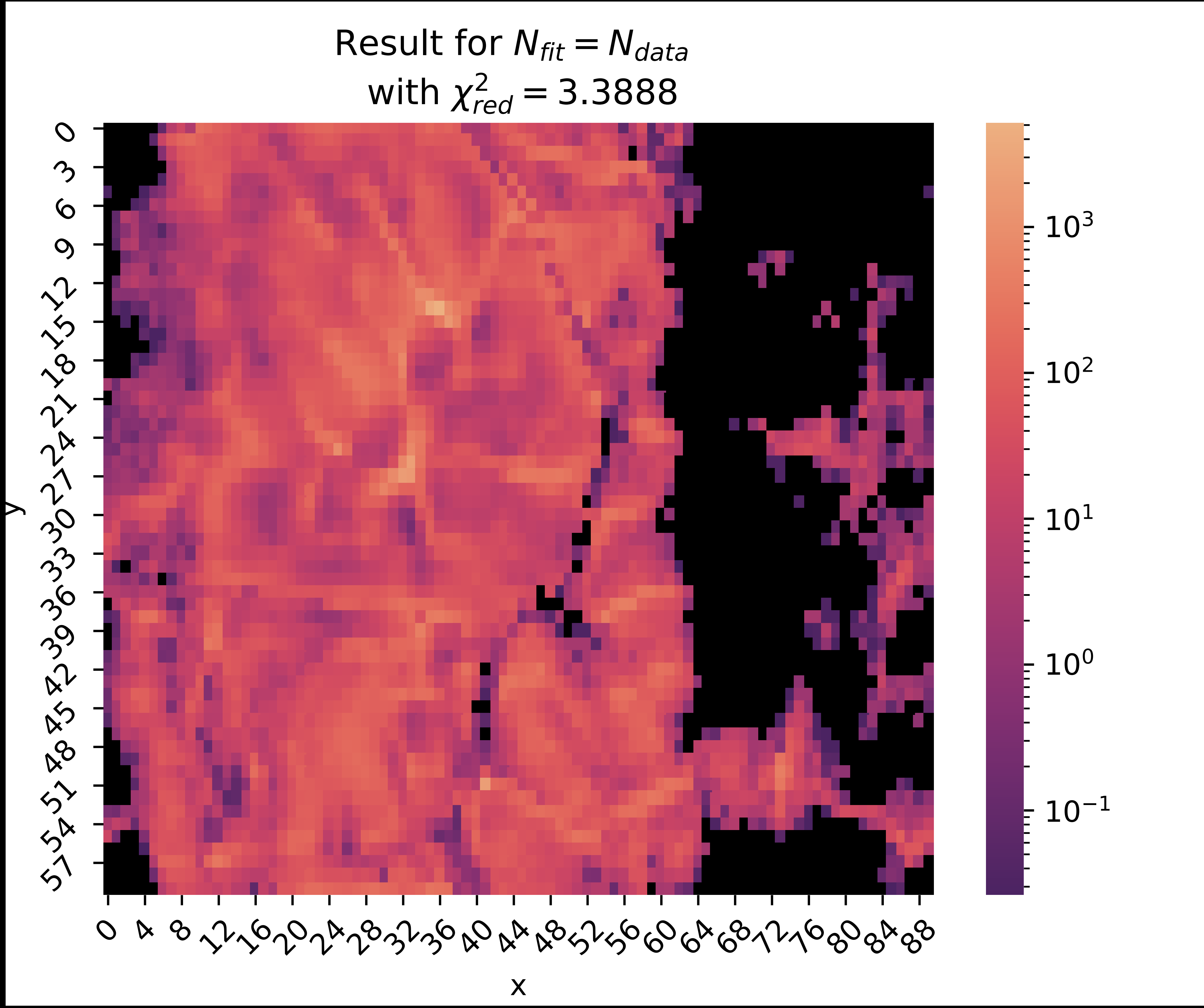
$t\bar{t}$



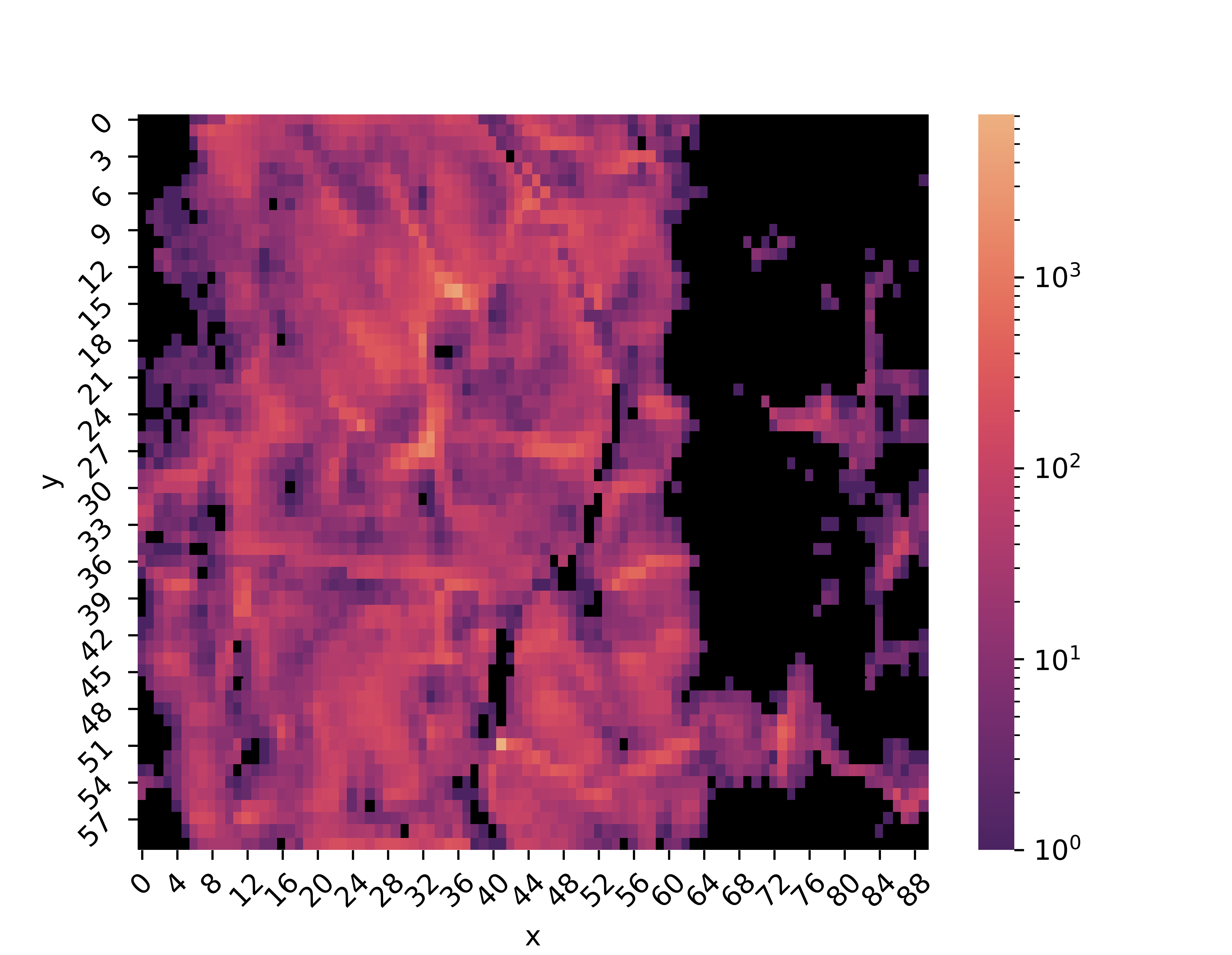
$W\overline{W}$



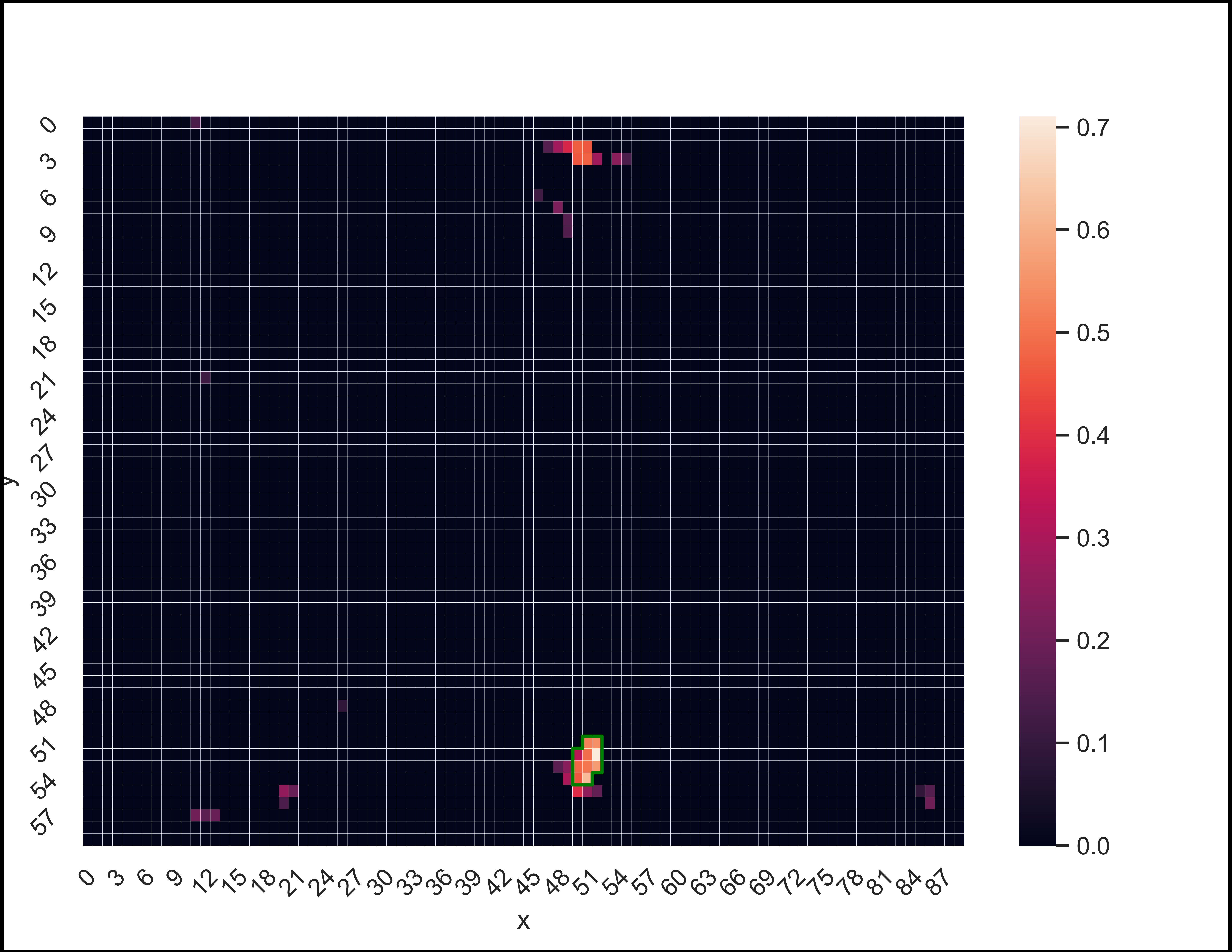
Fit result



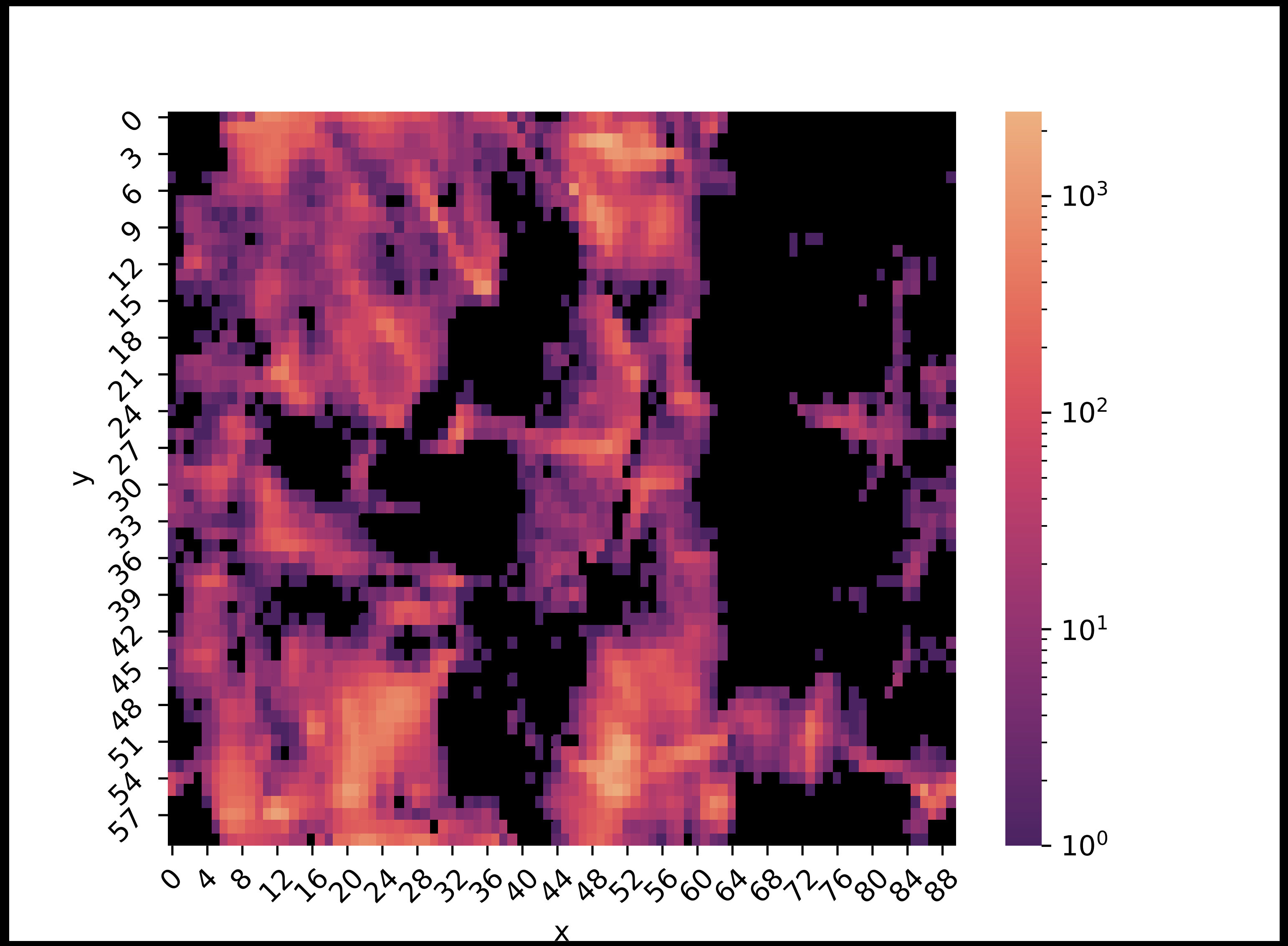
Real data



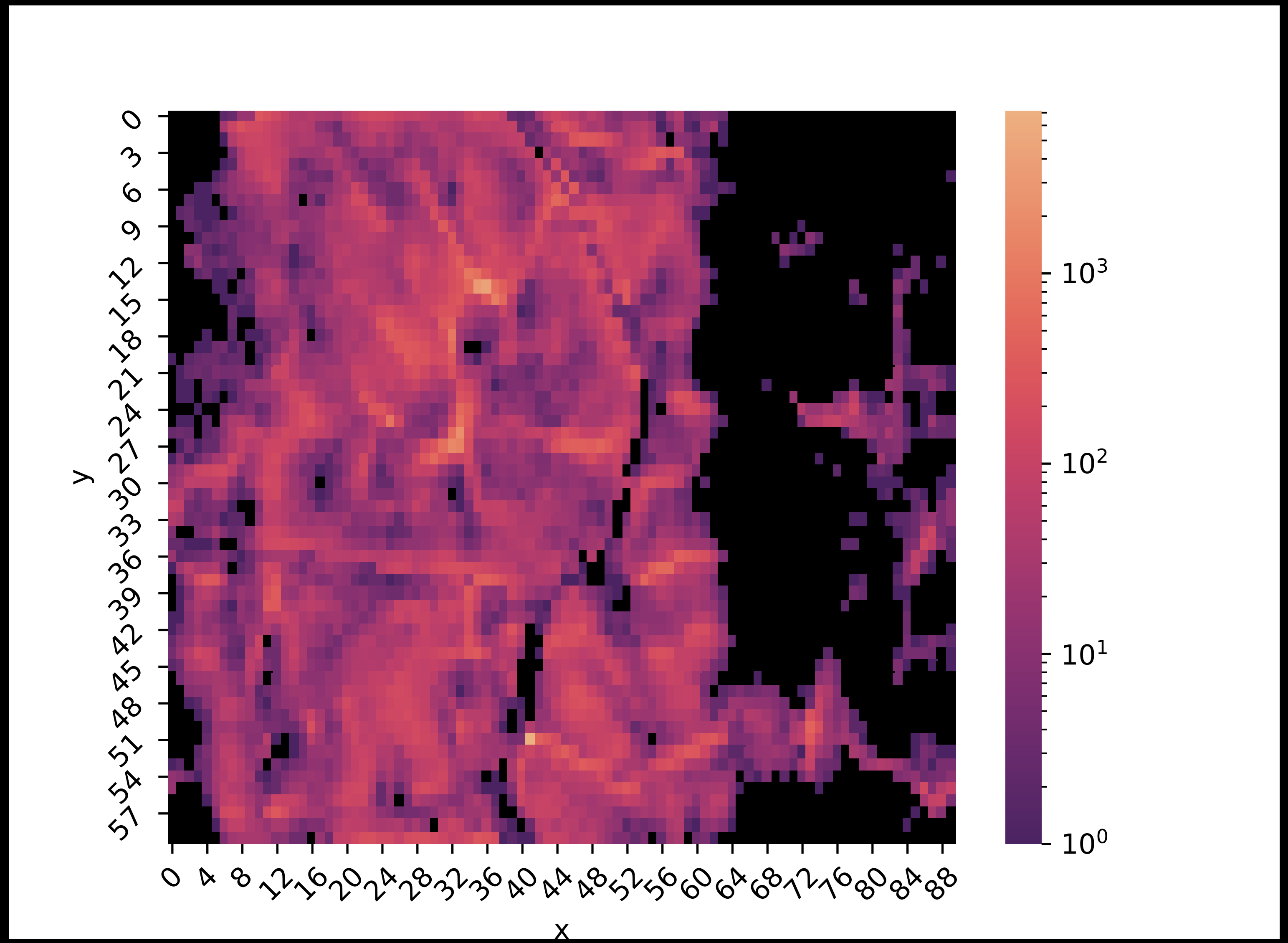
Relative amount of $H \to WW^*$

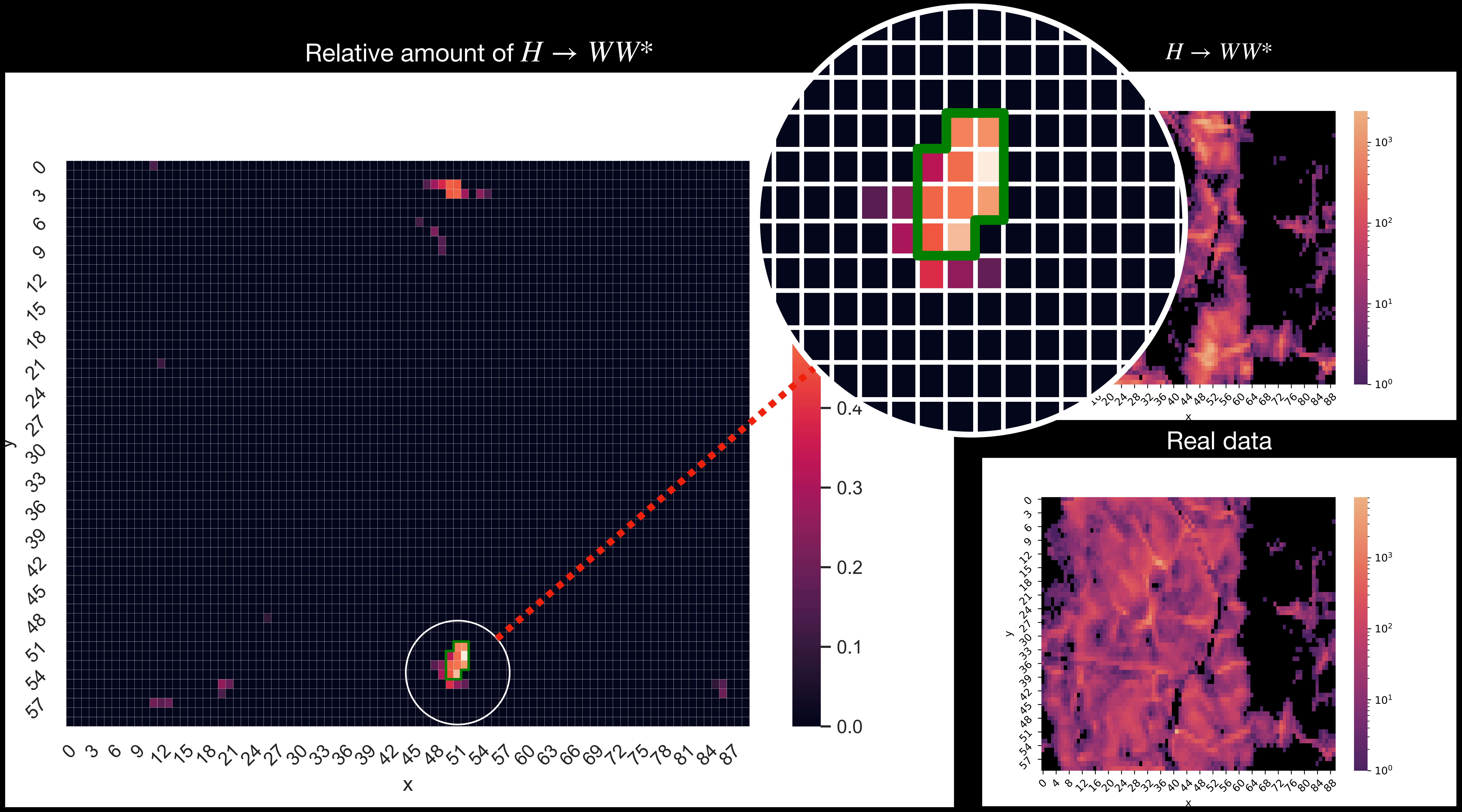


$H \to WW^*$



Real data



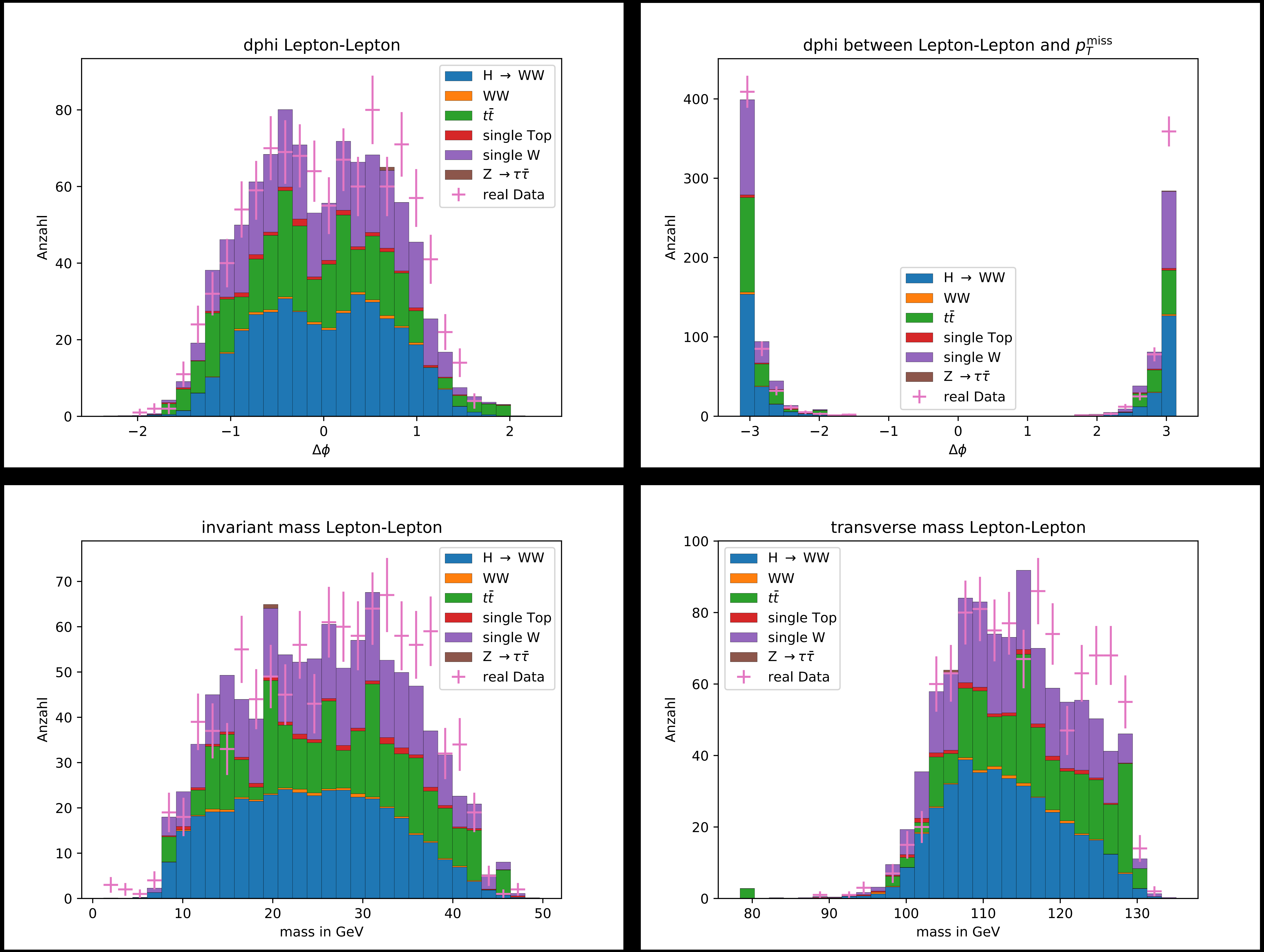


Relative amount of $H \rightarrow WW^*$
$H \rightarrow WW^*$
Real data
x
y
0.0
0.1
0.2
0.3
0.4

Isolated data

- 6099 $H \rightarrow WW^*$ events in total
- 396 $H \rightarrow WW^*$ events in subdataset
- Purity of 0.39

$$M_T^{ll} = \sqrt{2\, E_T^{ll}\, E_T^{\text{miss}}\, (1 - \cos(\theta_{ll,\text{miss}}))}$$



Summary and conclusion

- SOM can create low dimension map of data
- We can find clusters in the data
- SOM can be utilized to perform cuts on the data:
  - We can isolate 396 out of 6099 $H \rightarrow WW^*$ processes with a purity of 0.39

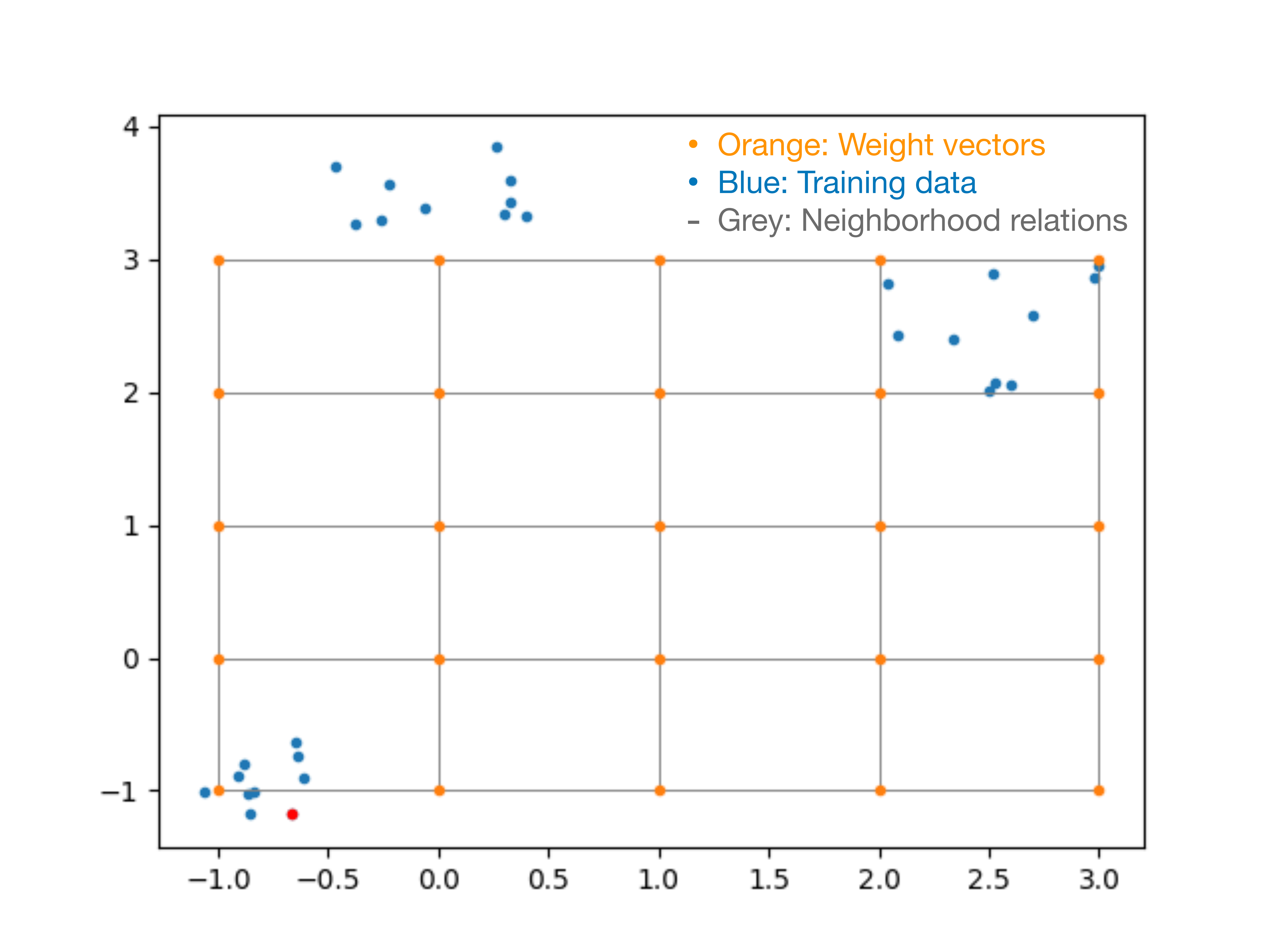
Thank you for listening!

Sources

1. Kohonen, T. Self-organized formation of topologically correct feature maps. *Biol. Cybern.* **43,** 59–69 (1982). https://doi.org/10.1007/BF00337288
2. Used Code: https://github.com/kai-git-stuff/SOM

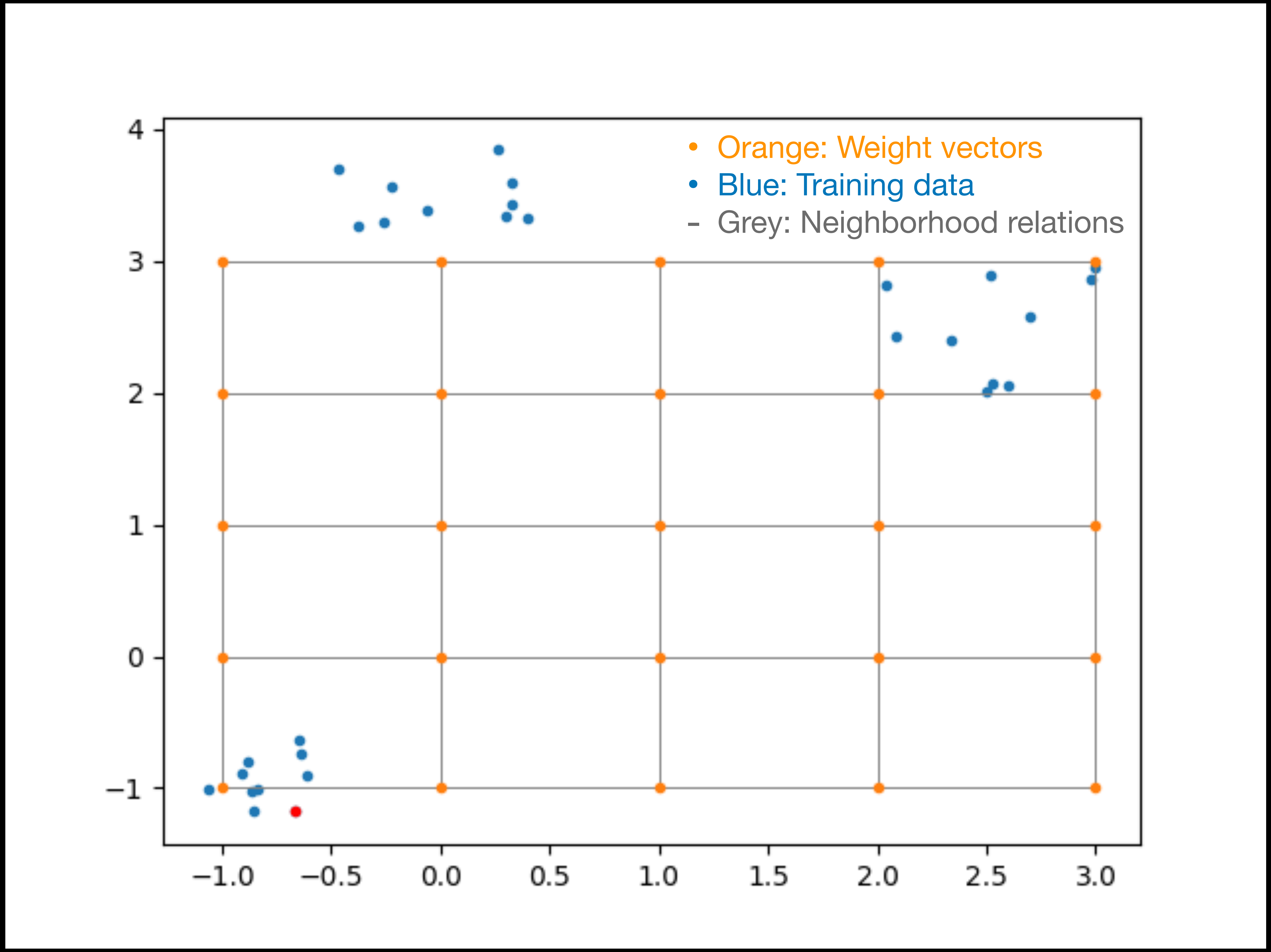
Training algorithm

1. Initalize weight vectors.
2. Select one input vector from training sample v_j.
3. Calculate euclidean distance $\|v_j - w_i\|$ to the weight vector of each neuron n_i and choose neuron n_{best} with lowest distance.
4. Renew weights depending on n_{best}.
5. Repeat steps 2-4 until iteration limit is reached.



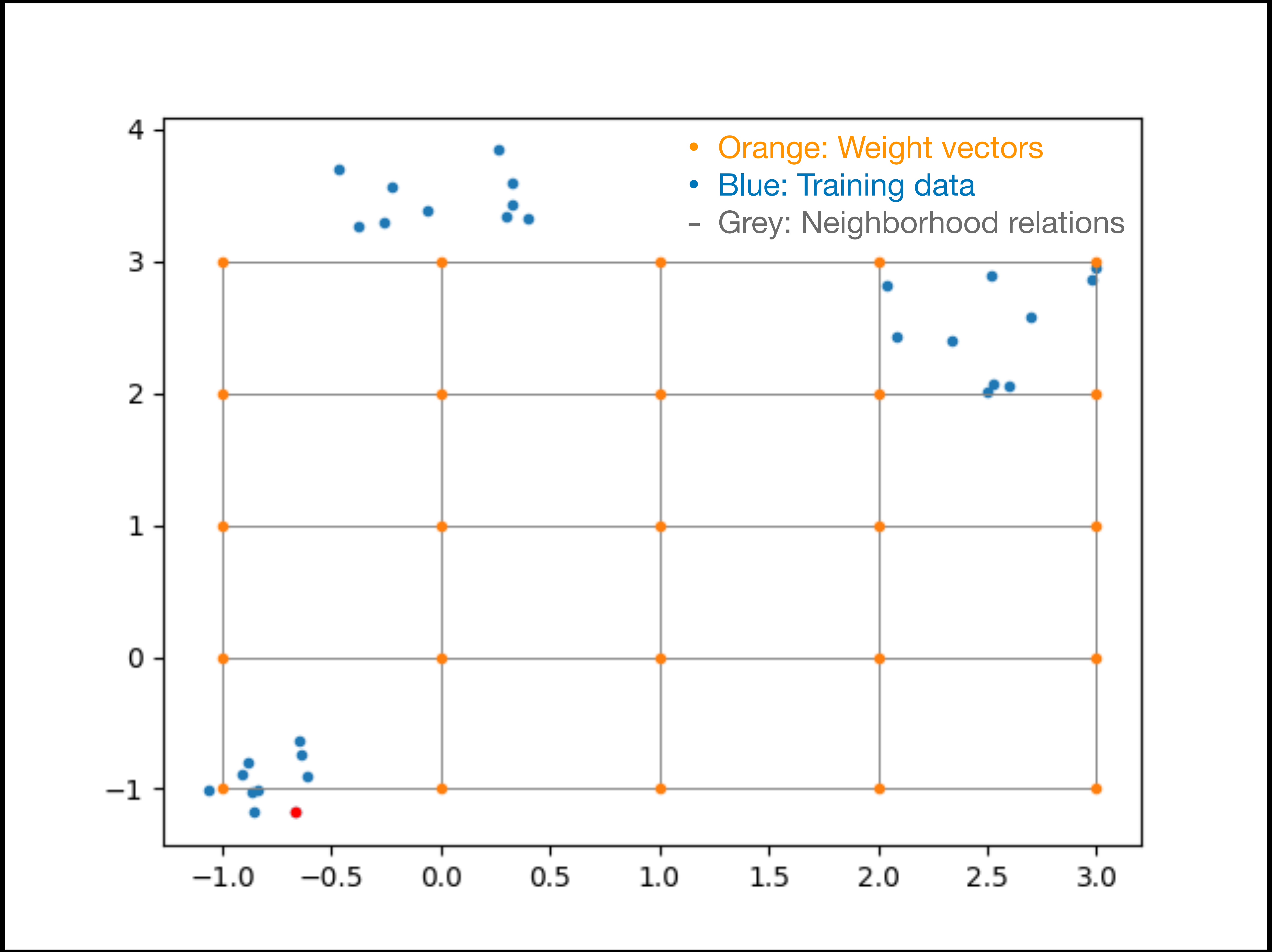
Training algorithm

1. Initalize weight vectors.
2. Select one input vector from training sample v_j.
3. Calculate euclidean distance $\| v_j - w_i \|$ to the weight vector of each neuron n_i and choose neuron n_{best} with lowest distance.
4. Renew weights depending on n_{best}.
5. Repeat steps 2-4 until iteration limit is reached.



Training algorithm

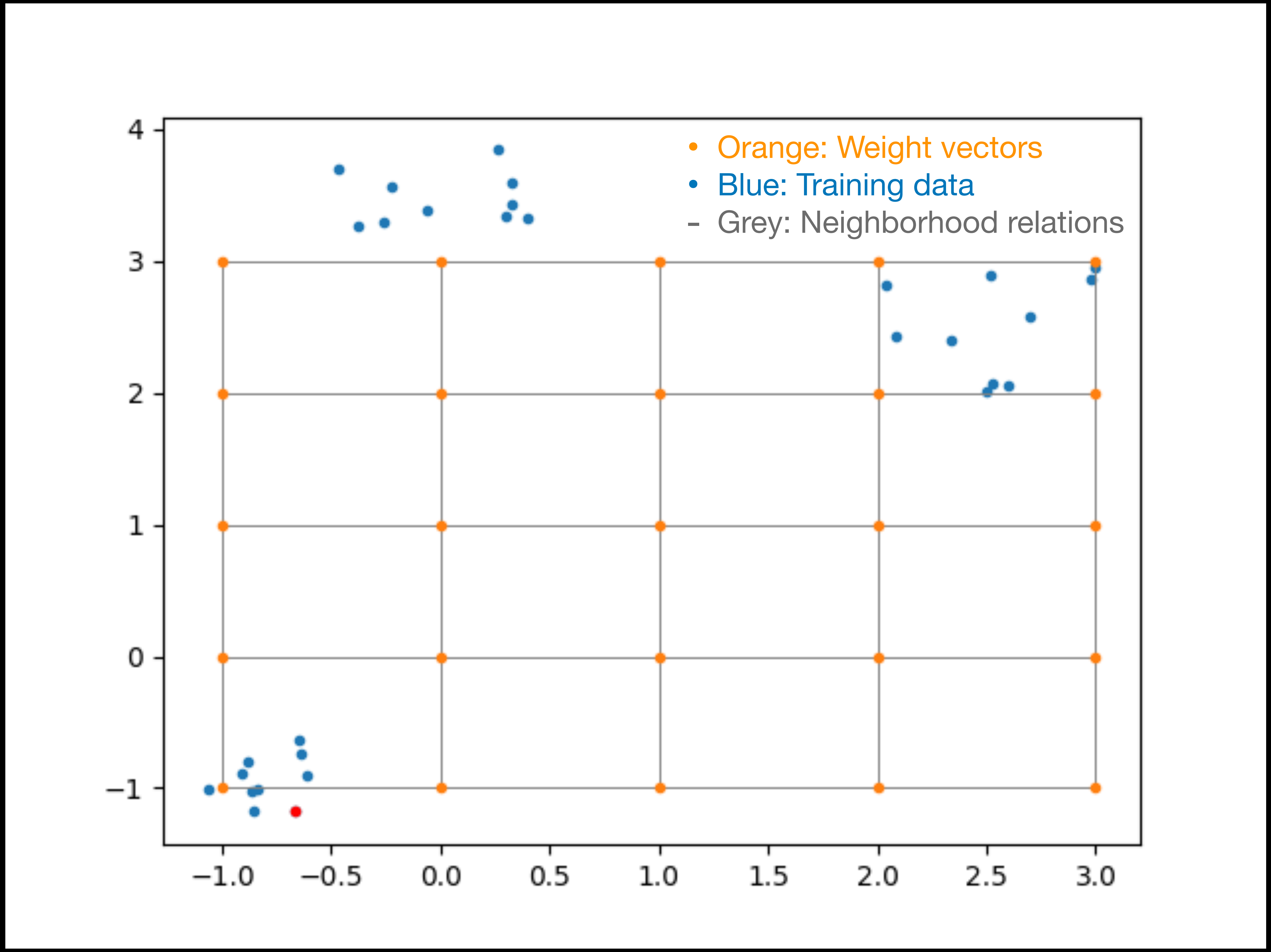
$$w_k^{t+1} = w_k^t + (v_j - w_k^t) \cdot \beta(r_k, \sigma) \cdot l$$



Training algorithm

$$w_k^{t+1} = w_k^t + \boxed{(v_j - w_k^t)} \cdot \beta(r_k, \sigma) \cdot l$$

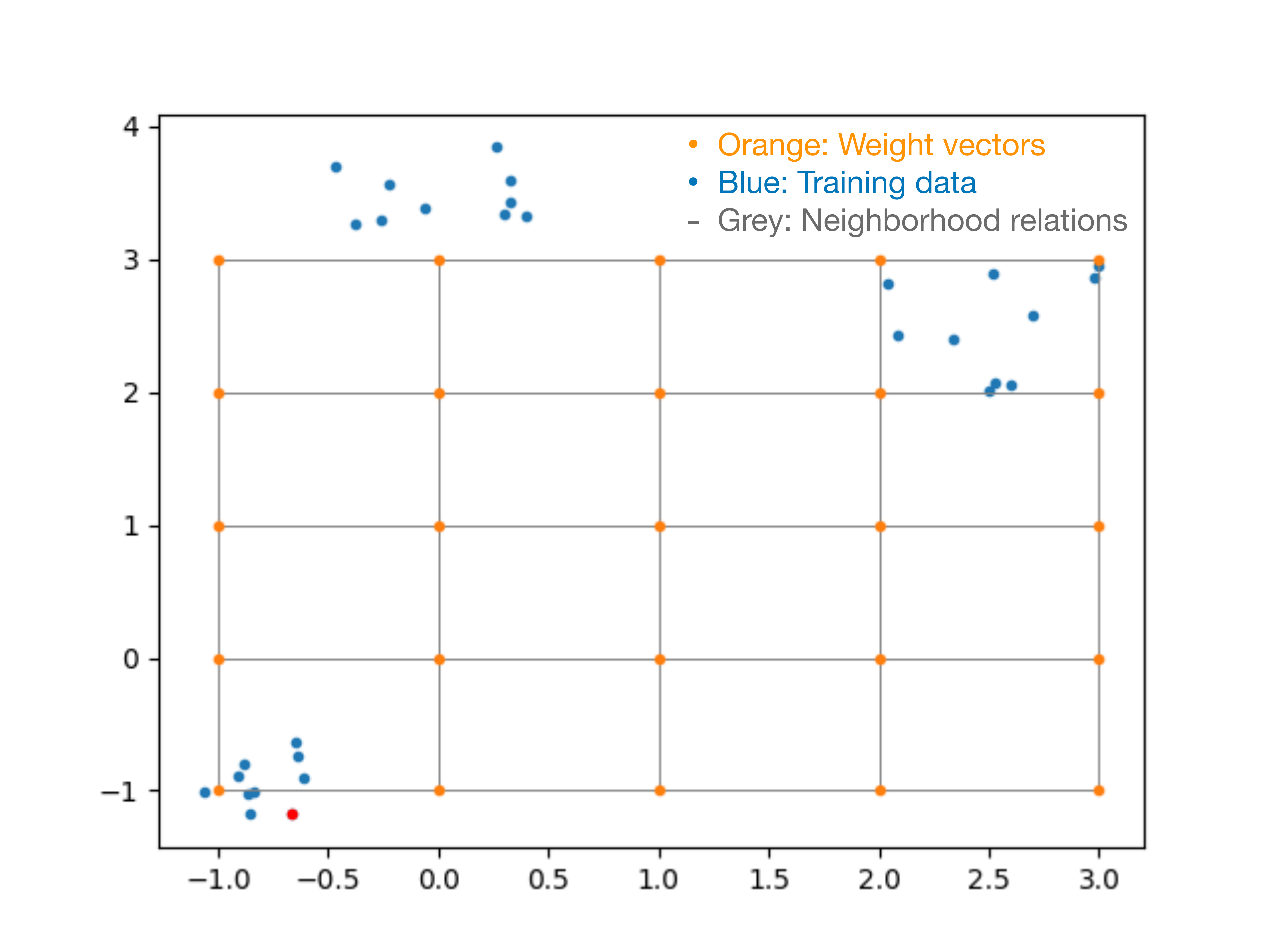
- Weights will be moved closer to v_j.



Training algorithm

$$w_k^{t+1} = w_k^t + (v_j - w_k^t) \cdot \boxed{\beta(r_k, \sigma)} \cdot l$$

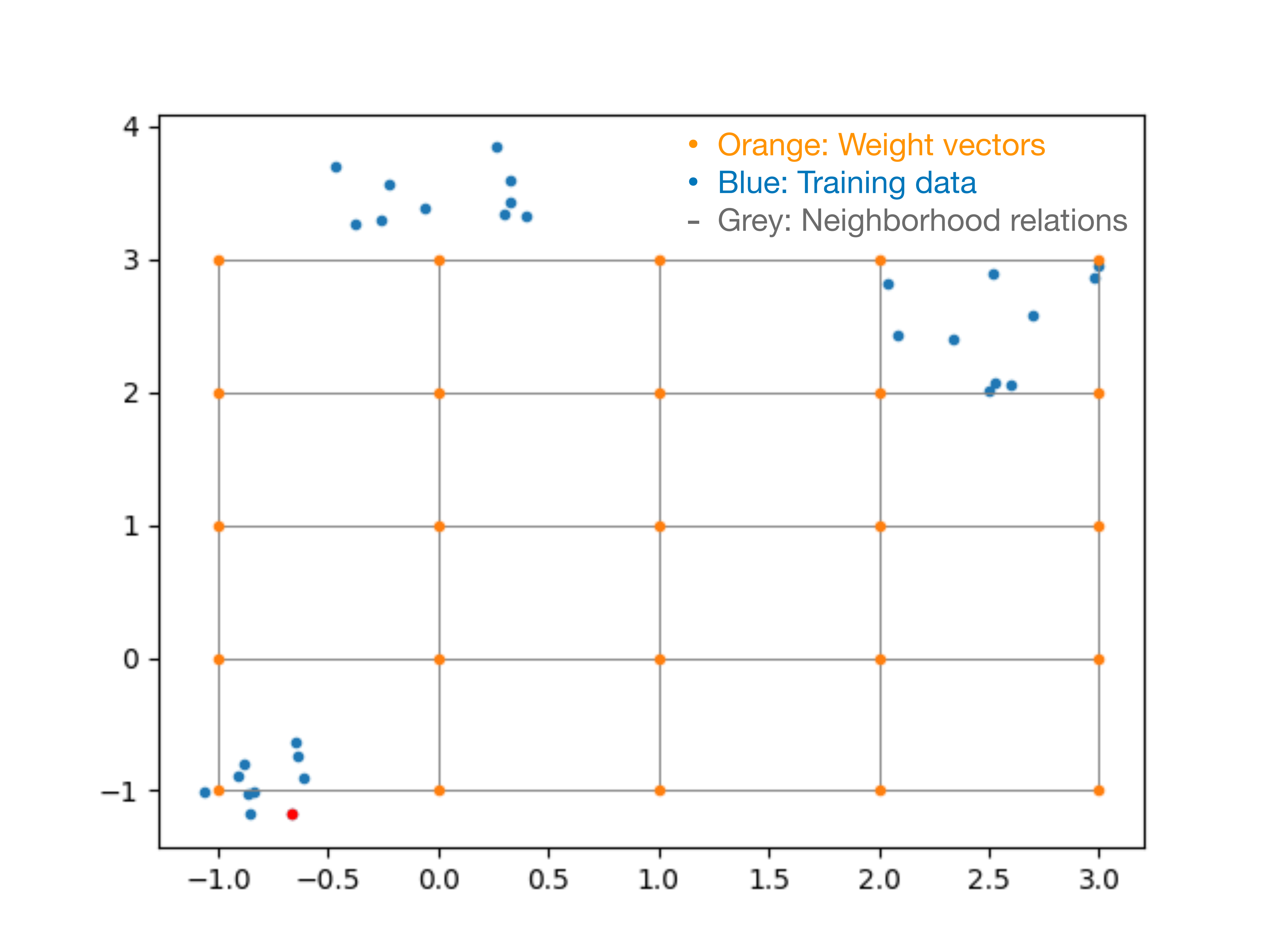
- Weights will be moved closer to v_j.
- $\beta(r, \sigma) = F_{\text{gauss}}(r, \sigma)$, r distance to n_{best} in the output space.



Training algorithm

$$w_k^{t+1} = w_k^t + (v_j - w_k^t) \cdot \boxed{\beta(r_k, \sigma)} \cdot l$$

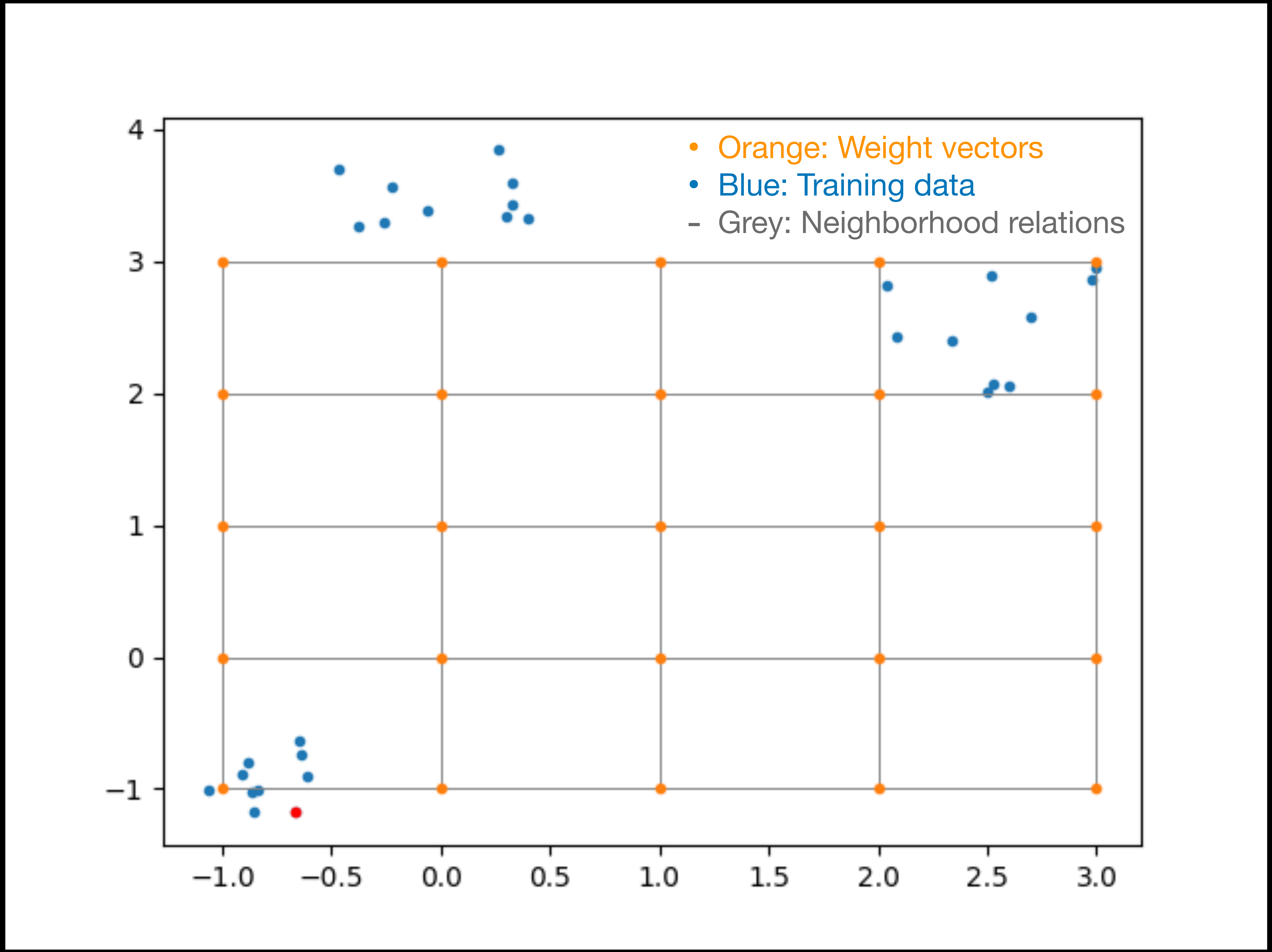
- Weights will be moved closer to v_j.
- $\beta(r, \sigma) = F_{\text{gauss}}(r, \sigma)$, r distance to n_{best} in the output space.
- Immediate neighbors of n_{best} will be altered the most.



Training algorithm

$$w_k^{t+1} = w_k^t + (v_j - w_k^t) \cdot \boxed{\beta(r_k, \sigma)} \cdot l$$

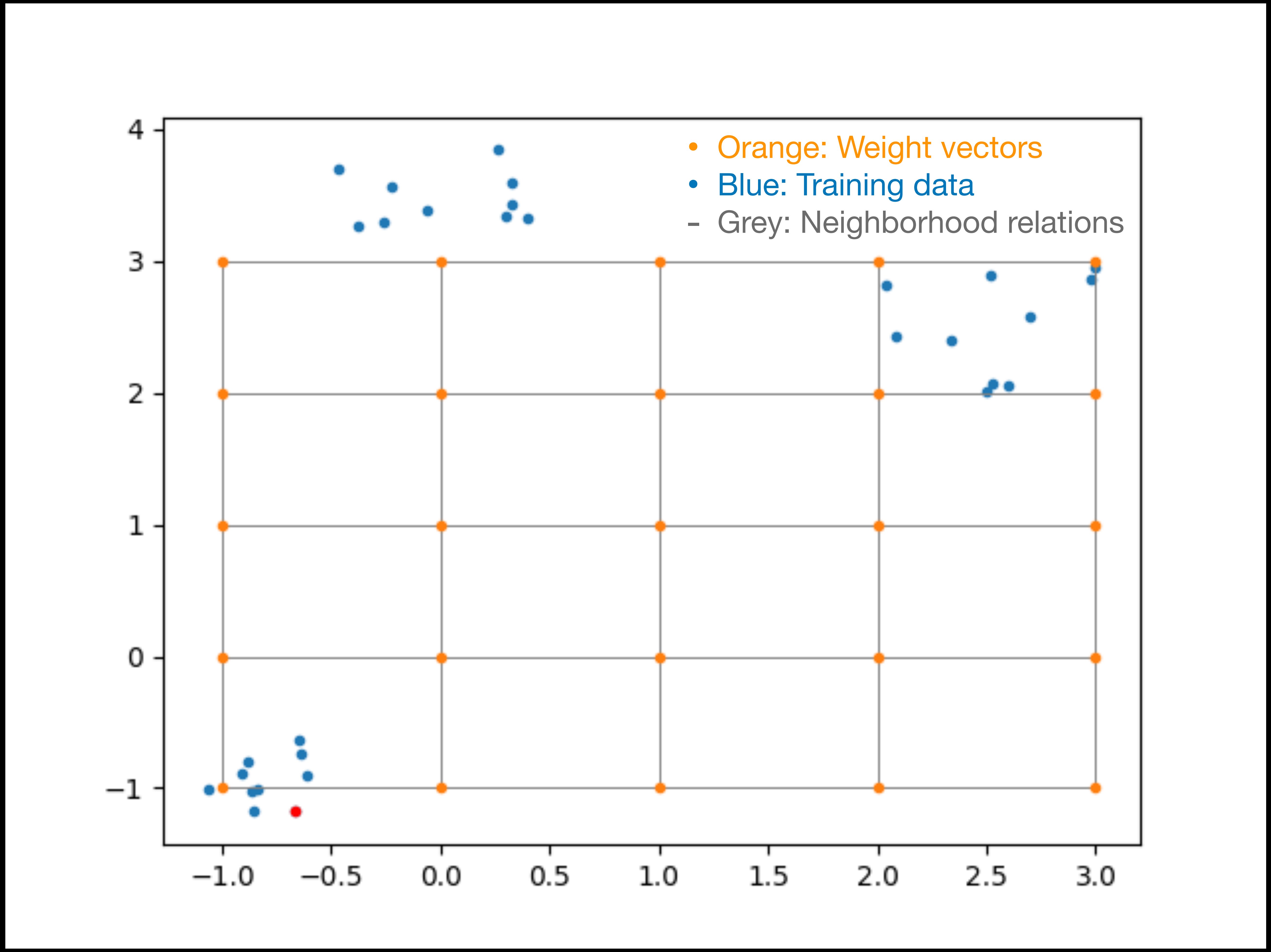
- Weights will be moved closer to v_j.
- $\beta(r, \sigma) = F_{\text{gauss}}(r, \sigma)$, r distance to n_{best} in the output space.
- Immediate neighbors of n_{best} will be altered the most.
- Distance relations from input space are mostly conserved.



Training algorithm

$$w_k^{t+1} = w_k^t + (v_j - w_k^t) \cdot \boxed{\beta(r_k, \sigma)} \cdot l$$

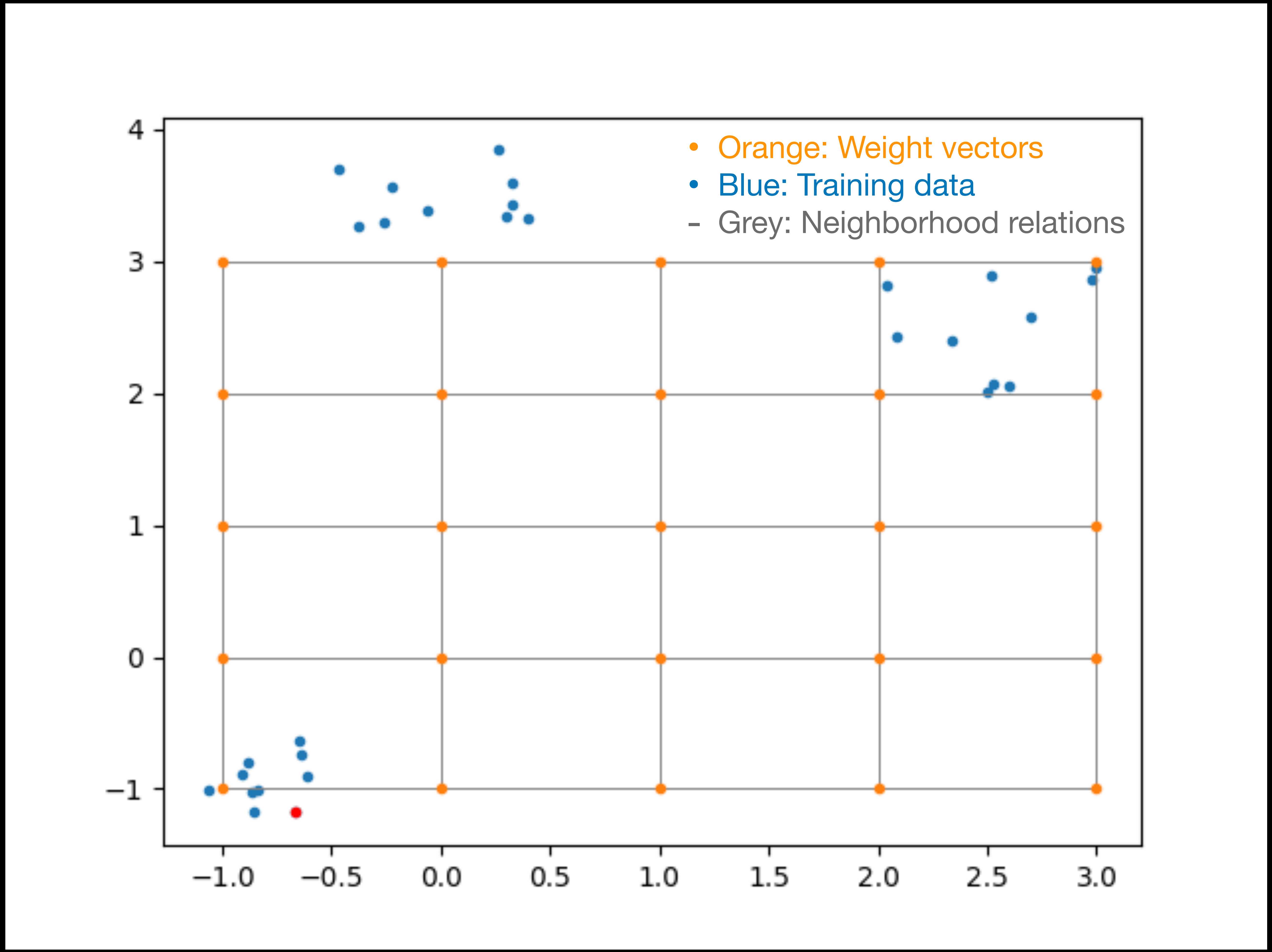
- Weights will be moved closer to v_j.
- $\beta(r, \sigma) = F_{\text{gauss}}(r, \sigma)$, r distance to n_{best} in the output space.
- Immediate neighbors of n_{best} will be altered the most.
- Distance relations from input space are mostly conserved.
- Clustering is encouraged.



Training algorithm

$$w_k^{t+1} = w_k^t + (v_j - w_k^t) \cdot \boxed{\beta(r_k, \sigma) \cdot l}$$

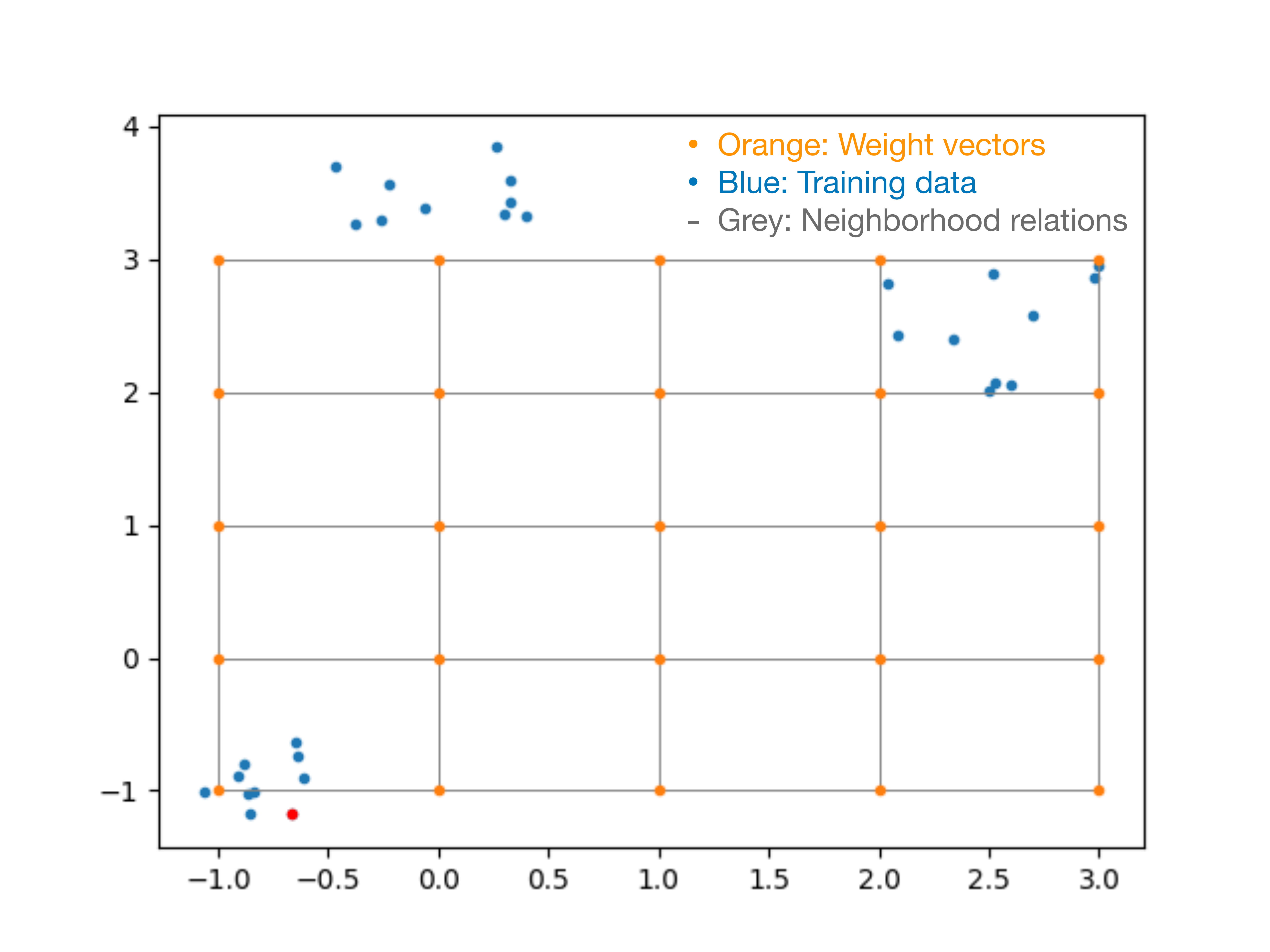
- Weights will be moved closer to v_j.
- $\beta(r, \sigma) = F_{\text{gauss}}(r, \sigma)$, r distance to n_{best} in the output space.
- Immediate neighbors of n_{best} will be altered the most.
- Distance relations from input space are mostly conserved.
- Clustering is encouraged.
- Convergence through decreasing l and σ.



Training algorithm

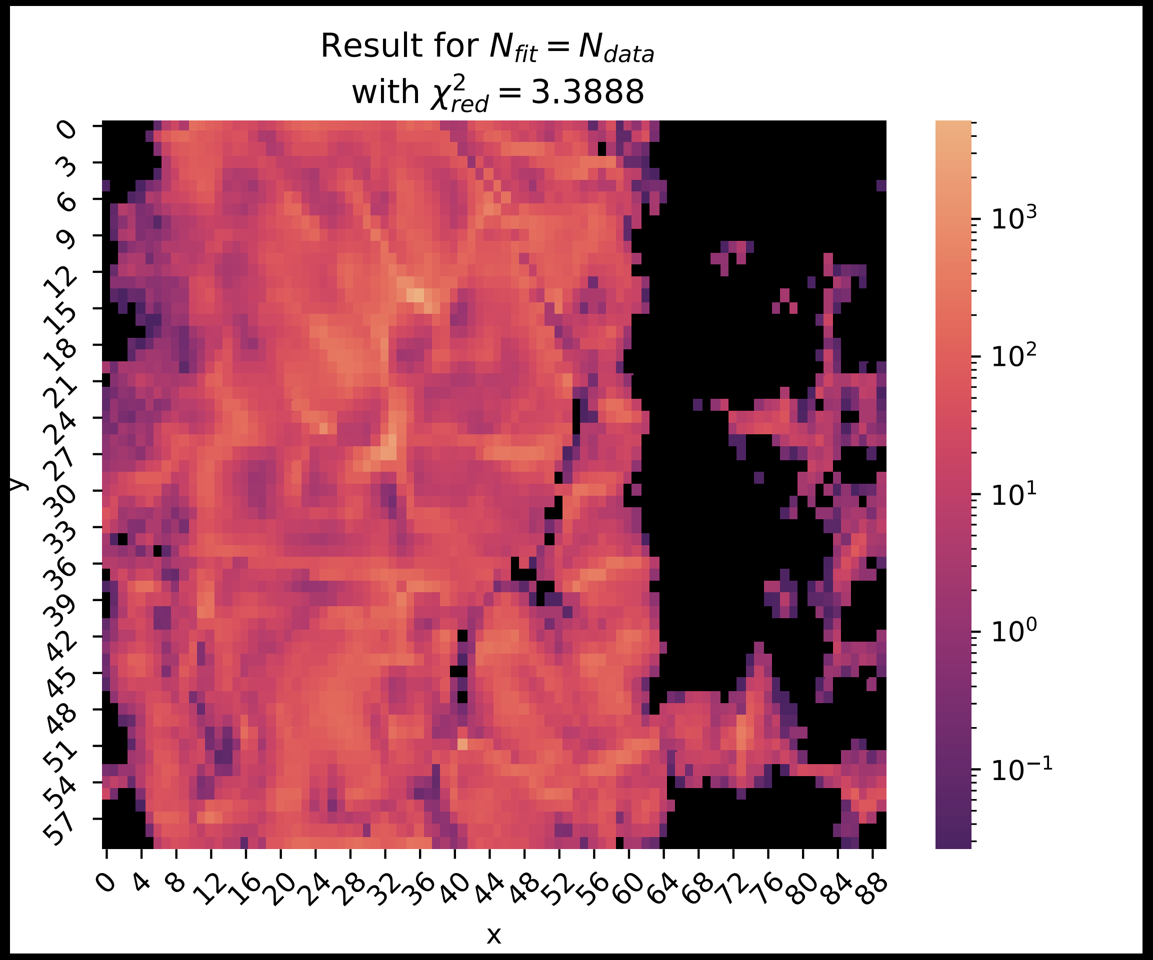
$$w_k^{t+1} = w_k^t + (v_j - w_k^t) \cdot \beta(r_k, \sigma) \cdot l$$

- Weights will be moved closer to v_j.
- $\beta(r, \sigma) = F_{\text{gauss}}(r, \sigma)$, r distance to n_{best} in the output space.
- Immediate neighbors of n_{best} will be altered the most.
- Distance relations from input space are mostly conserved.
- Clustering is encouraged.
- Convergence through decreasing l and σ.

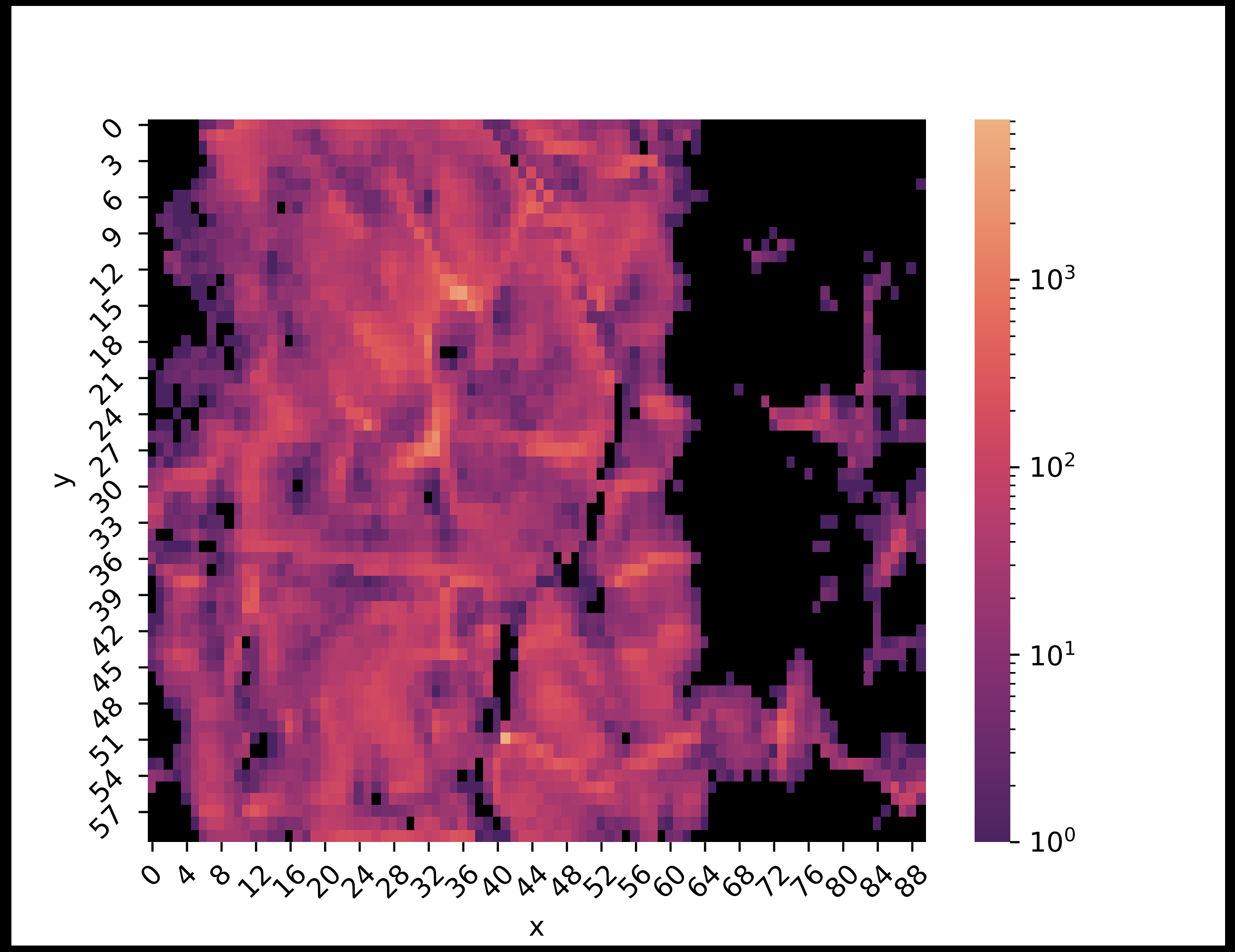


Fit procedure

Fitresult



Real data



- Map out MC-data on pre-trained SOM
- Normalize histograms
- Fit normalized real data histogram as weighted sum of normalized MC-data

Fit Results

- Isolation has to be smaller than 0.1 to reduce QCD
- $m_T^{ll} > 70\,\mathrm{GeV}$ to reduce Drell Yan $\tau\bar{\tau}$
  $\dfrac{\chi^2}{N_{df}} = 3.39$
- Bad χ^2 probably due to missing QCD

$H \rightarrow WW^*$	0.027 ± 0.002
Single top	0.039 ± 0.008
Single W	0.245 ± 0.003
$t\bar{t}$	0.382 ± 0.006
WW	0.189 ± 0.007
$Z \rightarrow \tau\bar{\tau}$	0.037 ± 0.002
$\sum$	0.919 ± 0.027