MIME
Master's degree ICT Internet Multimedia Engineering

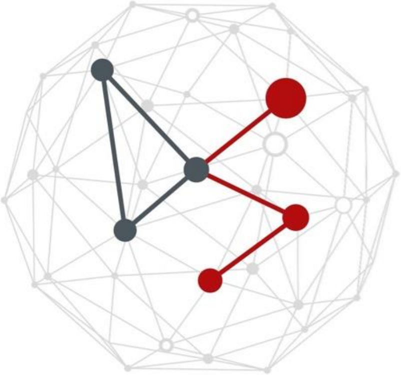
COMPETITIVE & UNSUPERVISED LEARNING FOR VECTOR QUANTIZATION

Michele Rossi

michele.rossi@unipd.it

Dept. of Information Engineering

University of Padova, IT



DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO
MATEMATICA

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Outline (1/2)

- Vector quantization - definition
- Self Organizing Maps (SOM)
  - SOM & the human brain
  - SOM’s goal
  - Input vectors & synaptic weights
  - 2D neural network example
- Learning steps
  - Competition
  - Cooperation
  - Synaptic adaptation
- Learning phases & experiments
- Pros & cons of SOM

Outline (2/2)

- Some useful concepts
  - Delaunay triangulation
  - Voronoi diagrams
- Topology preserving feature maps
  - Hebbian learning / Competitive Hebbian Learning (CHL)
  - Theorem “CHL preserves feature maps”
- Growing Neural Gas (GNG) networks
  - Algorithm
  - Remarks
  - Example results

Setup

- Input space, vectors of m elements

$$\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^T \quad \boldsymbol{x} \in \mathbb{R}^m$$

- i.i.d. sequentially sampled, one at a time,
  - from the same pdf $f(\boldsymbol{x})$
  - time is discrete n = 1, 2, …

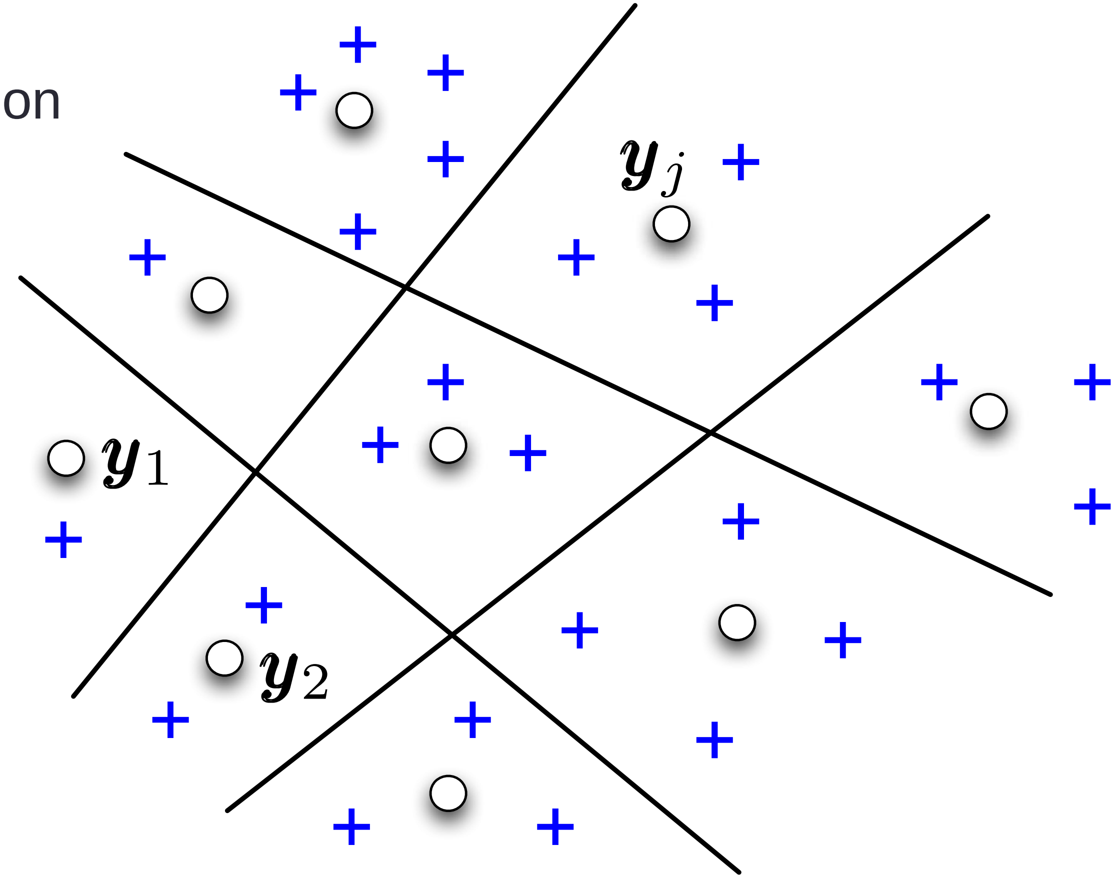
- A set of centroids is defined (e.g., cluster centers in K-means)

$$\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_\ell\}$$

- with: $\boldsymbol{y}_j \in \mathbb{R}^m$

Vector Quantization (1/2)

- Given an input data distribution (multi-dimensional vectors)
- We need to find:
  - A partition rule (regions)
  - A prototype for each region
    - Also called centroid



○ **Prototype (codeword)**

+ **Input data vector**

Vector Quantization (2/2)

- Distortion measure
  - Most common measure: Euclidean distance
  - Average distortion is quantified as the mean square error

$$E[d(\boldsymbol{x},\boldsymbol{y}_j)] = \sum_{j=1}^{\ell}\int_{I_j} \|\boldsymbol{a}-\boldsymbol{y}_j\| f(\boldsymbol{a})\,d\boldsymbol{a} \qquad (1)$$

Region j

Centroid

Input signal pdf

$\|\boldsymbol{x}\| \to \text{norm-2}$

- Optimal VQ
  - Find a set of centroids and a partition rule that minimize (1)
  - Nearest neighbor condition: the optimal partition is the one returning the minimum distortion (region j)

$$I_j = \{\boldsymbol{x} : d(\boldsymbol{x},\boldsymbol{y}_j) \leq d(\boldsymbol{x},\boldsymbol{y}_h), j \neq h\}$$

What self organizing maps are (1/2)

- Neural networks based on competitive learning
- Competitive learning
  - Output neurons compete among themselves to be activated (or *fired*)
  - Result: *only one neuron is active at any one time*
  - The neuron that wins the competition is called
    - The “winner-takes-all neuron” or simply,
    - the “winner”
- In a SOM
  - Neurons are usually placed at the nodes of a lattice (usually 1D or 2D)
  - The neurons become selectively tuned to the input patterns (*stimuli*) in the course of a competitive learning process
  - Learning is unsupervised

What self organizing maps are (2/2)

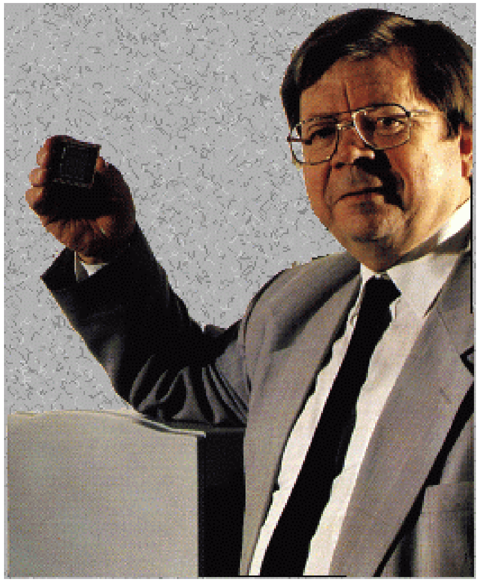
- Each neuron j in the SOM
  - Has a weight (or "feature vector") $\boldsymbol{w}_j = [w_{j1}, w_{j2}, \ldots, w_{jm}]^T$
  - The weights reflect the position of the neuron in the lattice
    - Also referred to as "coordinates" of the neuron
  - The spatial locations (i.e., the neuron's weights)
    - During the learning phase, become ordered with respect to each other in a way that a meaningful coordinate system is created over the lattice
    - The result is a topographic map of the input patterns, where the *coordinates* of the neurons in the lattice *are indicative of statistical features contained in the input patterns*

- Key properties
  - The SOM is inherently non-linear
  - Learning (adaptation of its weights) is unsupervised

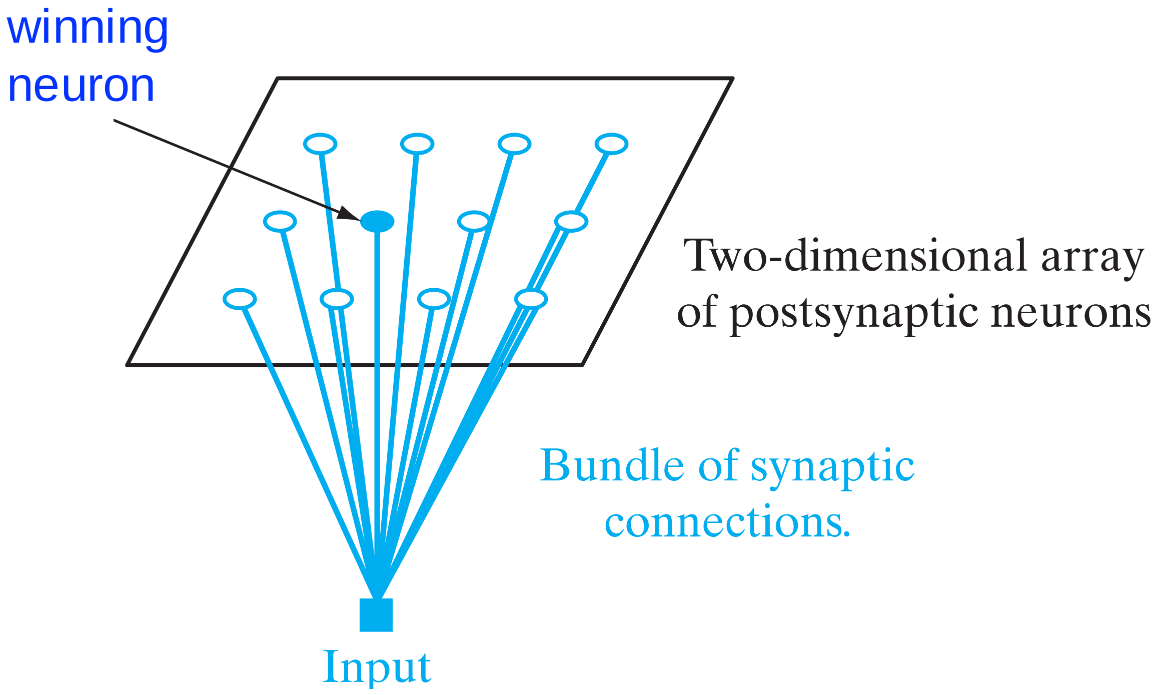
SOM & the human brain

- SOMs are motivated by features of the human brain
  - "The brain is organized in many places in a way that different sensory inputs are represented by topologically ordered computation maps"
- Key fact: sensory inputs such as
  - Tactile (Kaas, J.H., M.M. Merzenich, and H.P. Killackey, 1983. "The reorganization of somatosensory cortex following peripheral nerve damage in adult and developing mammals," *Annual Review of Neurosciences,* vol. 6, pp. 325–356)
  - Visual (Hubel, D.H., and T.N. Wiesel, 1962. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *Journal of Physiology,* vol. 160, pp. 106–154, London)
  - Acoustic (Suga, N., 1985. "The extent to which bisonar information is represented in the bat auditory cortex," in *Dynamic Aspects of Neocortical Function,* G.M. Edelman, W.E. Gall, and W.M. Cowan, eds. pp. 653–695, New York: Wiley)
- are mapped onto several areas of the *cerebral cortex* in a topologically ordered manner

SOM



Prof. Teuvo Kohonen


winning neuron
Two-dimensional array of postsynaptic neurons
Bundle of synaptic connections.
Input

- Key observation: the spatial location of an output neuron in a topographic map corresponds to a particular feature of the data drawn from the input space
- SOM: it is not meant to explain neurobiological details, but (i) *to capture the essential features* of computational maps in the brain and (ii) to remain *computationally tractable*
- [Kohonen1982] Teuvo Kohonen, “Self-organized formation of topologically correct feature maps,” *Biological Cybernetics,* vol. 43, pp. 59–69, 1982.

Self Organizing Map (SOM)

- Goal
  - Is to transform an incoming signal pattern of arbitrary dimension (m)
  - Into a one- or two-dimensional discrete map of neurons
- The algorithm in a nutshell
  - Initialization: the neuron's (synaptic) weights are initialized at random: no prior order is imposed on the map. Then, for each input vector DO:
  - 1. Competition: for each input pattern, each neuron computes a local value using a discriminant function. The neuron with the highest value wins the competition
  - 2. Cooperation: the winning neuron provides the location of a topological neighborhood of excited neurons. The neurons in this neighborhood will update their synaptic weights
  - 3. Synaptic adaptation: the winning neuron, as well as the neurons in its neighborhood update their synaptic weight, bringing it closer to the input pattern (vector)

Input vectors & synaptic weights

- Let m denote the dimension of the input data space
- Let **x** be a vector selected at random from the input space

$$\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^T$$

- The synaptic-weight vector of each neuron in the SOM has the same dimension of the input space. The weight vector associated with neuron j is:

$$\boldsymbol{w}_j = [w_{j1}, w_{j2}, \ldots, w_{jm}]^T, \; j = 1, 2, \ldots, \ell$$

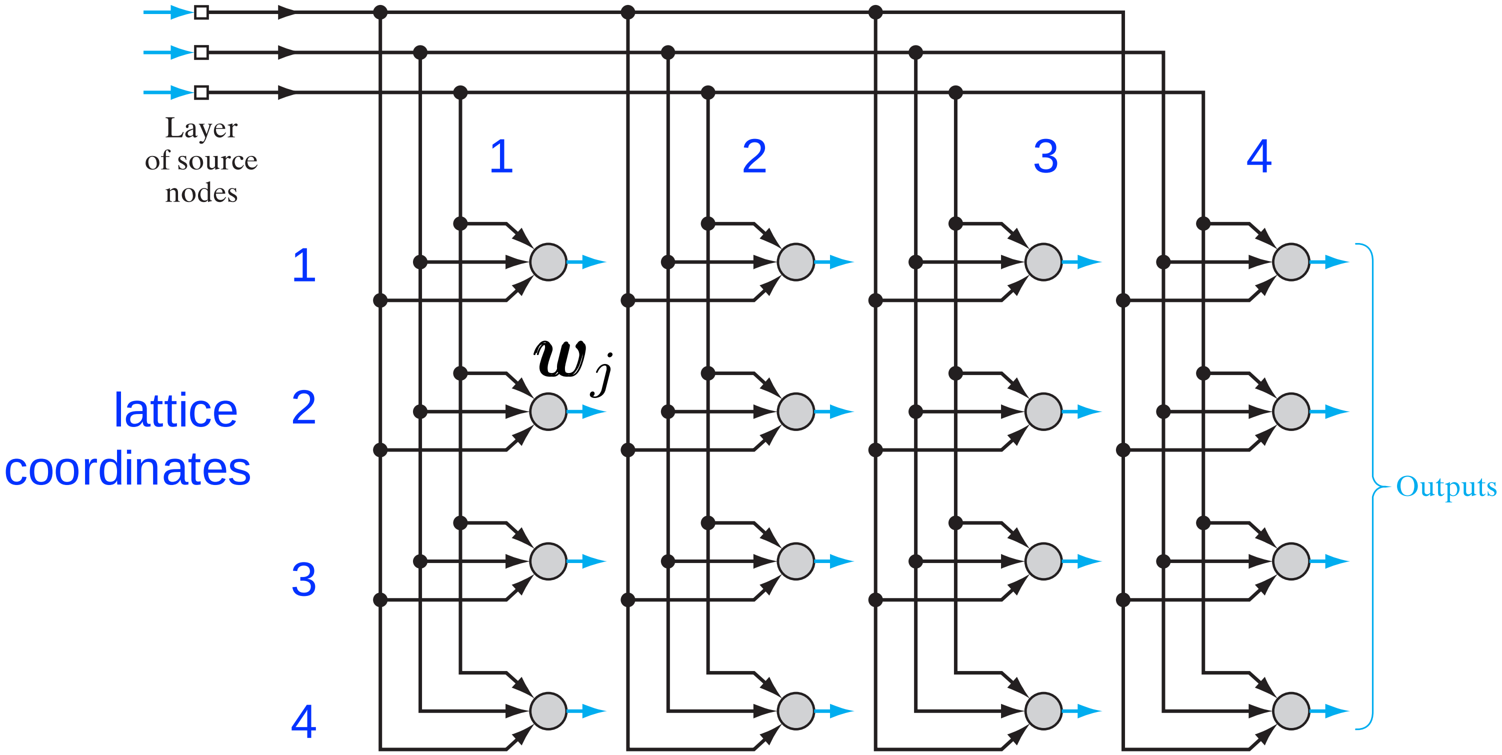
- ℓ represents the number of (output) neurons in the SOM

A 2D SOM example

2D lattice of $\ell = 4 \times 4 = 16$ neurons

Input patterns are vectors with three elements

$$\boldsymbol{x} = [x_1, x_2, x_3]^T$$



What SOM does

- It is a topological mapping algorithm
- “Optimally” places a fixed number of vectors (the neuron’s weights) into a higher dimensional space (of size m):
  - It is a dimensionality reduction algorithm (see below)
  - This means that is suitable for clustering (data compression)
- More specifically, the SOM maps
  - Input (feature) vectors of m elements $\boldsymbol{x} \in \mathbb{R}^m$ into
  - ℓ vectors of m elements $\boldsymbol{w}_j \in \mathbb{R}^m, \, j = 1, 2, \ldots, \ell$
  - with $\ell < m$

1. Competition

- Let **x** be a vector from the input space

$$\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^T$$

- All the neurons in the SOM lattice compete
  - The best fit neuron will win the competition
  - Let i(**x**) be the index associated with the winning neuron
  - This index i(**x**) is computed as:

$$\boxed{i(\boldsymbol{x}) = \arg\min_{j} \|\boldsymbol{x} - \boldsymbol{w}_j\|, \; j \in \mathcal{L}}$$

  - where $\mathcal{L}$ indicates the SOM lattice (*output space*)
  - and $\|\boldsymbol{x} - \boldsymbol{w}_j\|$ is the Euclidean distance
  - Observation: a *continuous* input space of activation patterns is mapped onto a *discrete output space* (lattice) of neurons by a process of competition among the neurons in the neural network

2. Cooperation (1/5)

- The winning neuron locates the center of a *topological neighborhood of cooperating neurons*
- The question now is: *"how do we define a neighborhood that is neuro-biologically correct?"*
- Answer: there is neurobiological evidence that
  - Lateral interaction occurs among excited neurons in the human brain
  - This means that: a neuron that fires tends to *excite more the neurons that are located in its immediate neighborhood* rather than those that are located away from it. This is also intuitively satisfying
- Hence: this observation leads us to define a topological neighborhood around the winning neuron $i(\mathbf{x})$ and make it decay smoothly with lateral distance

2. Cooperation (2/5)

- Topological neighborhood $h_{i,j}$
  - Centered on winning neuron i
  - Encompassing a set of neighboring neurons, denoted by j
  - Let $d_{i,j}$ be the lateral distance between neuron i and neuron j
- We assume that:
  - **1.** the topological neighborhood $h_{i,j}$ is symmetric around the maximum point, which is defined by $d_{i,j} = 0$. In other words, it attains its maximum value at the winning neuron i
  - **2.** the amplitude of the lateral neighborhood $h_{i,j}$ decreases monotonically with the distance to the point that it goes to zero as the distance diverges to infinity (this condition is key for the convergence of the learning phase)

$$h_{i,j}(n) \to 0 \text{ as } d_{i,j} \to +\infty$$

2. Cooperation (3/5)

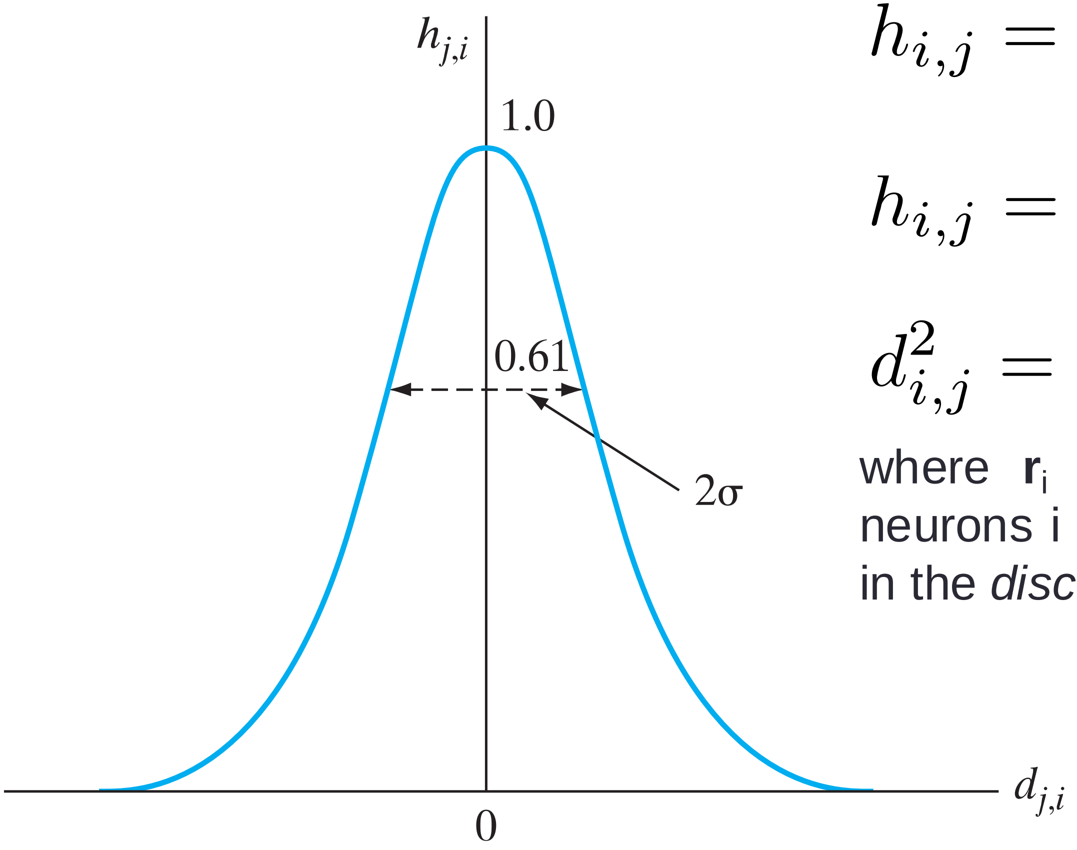
- A good choice for the neighborhood function is the Gaussian function

$$h_{i,j} = \exp\left(-\frac{d_{i,j}^2}{2\sigma^2}\right), \ j \in \mathcal{L}$$

- Observations
  - This function is translation invariant, i.e., it is independent of the index (location) of the winning neuron
  - parameter σ is the "effective width": it measures the degree to which the neurons in the neighborhood of the winner participate in the learning process
  - a Gaussian neighborhood function leads to faster convergence than, e.g., a rectangular neighborhood function
  - in the definition of neighborhood there is no "wrapping around"

2. Cooperation (4/5)

- The Gaussian neighborhood function



$$h_{i,j} = \exp\left(-\frac{d_{i,j}^2}{2\sigma^2}\right) \;, j \in \mathcal{L}$$

$$h_{i,j} = h_{j,i}$$

$$d_{i,j}^2 = \|\boldsymbol{r}_i - \boldsymbol{r}_j\|^2$$

where $\mathbf{r}_i$ and $\mathbf{r}_j$ are the positions of neurons i and j and are both measured in the *discrete output space* (the lattice)

2. Cooperation (5/5)

- Another unique feature of a SOM algorithm is that the size of the topological neighborhood is permitted to shrink with time n, i.e.,

$$\sigma(n) = \sigma_0 \exp\left(-\frac{n}{\tau_1}\right) \, , \, n = 0, 1, 2, \ldots$$

- Where τ_1 is a constant (empirically) picked by the designer. Hence, the topological neighborhood function assumes a time-varying form of its own:

$$h_{i,j}(n) = \exp\left(-\frac{d_{i,j}^2}{2\sigma(n)^2}\right) \, , \, j \in \mathcal{L} \, , \, n = 0, 1, 2, \ldots$$

3. Adaptation (1/3)

- Let n be the current time
- Time n is updated to n+1 at each new input
- Let $\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^T$ be the input vector at time n
- Let $i(\boldsymbol{x})$ be the index of the winning neuron
- Each neuron j (including the winning neuron i itself) updates its synaptic weight vector using:

$$\Delta \boldsymbol{w}_j = \eta(n) h_{i,j}(n)(\boldsymbol{x} - \boldsymbol{w}_j)\,, \quad \begin{cases} i: & \text{winning neuron} \\ j: & \text{excited neuron} \end{cases}$$

Delta update rule

3. Adaptation (2/3)

- Using discrete-time formalism
- The update equation for the weight vectors is:

$$\boldsymbol{w}_j(n+1) = \boldsymbol{w}_j(n) + \eta(n) h_{i,j}(n)(\boldsymbol{x} - \boldsymbol{w}_j)\ ,\ j \in \mathcal{L}$$

- $h_{i,j}(n)$: controls the size of the neighborhood *vs* space & time
- η(n): is a learning rate parameter
- This update equation is of a stochastic approximation type
  - The new weight vector **w**(n+1) is equal to the old one **w**(n) plus a Δ**w** (update) term, which depends on the distance between (i) the current input vector **x** and (ii) the weight vector $\mathbf{w}_j$
  - The intensity of the update depends on (i) how much **x** differs from $\mathbf{w}_j$, (ii) the learning rate parameter and (iii) the location of neuron j wrt to the winning neuron i(**x**) (note that j and i can also coincide)

3. Adaptation (3/3)

- Learning rate parameter
- Usually an exponentially decaying equation, although may not be optimal, has been (empirically) found to be adequate for the SOM algorithm to converge towards a correct topological map

$$\eta(n) = \eta_0 \exp\left(-\frac{n}{\tau_2}\right) \; , \; n = 0, 1, 2, \ldots$$

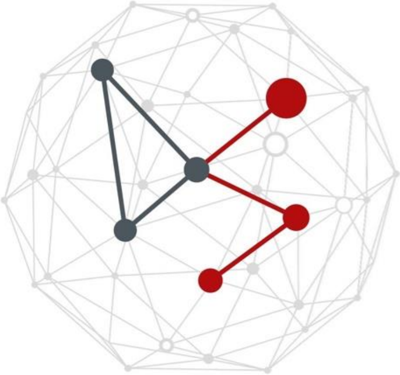
- Parameter τ_2 is set empirically by the designer

Observations

- The neighborhood function correlates neuron weights in the output space
  - A wide function (large $h_{i,j}(n)$) will correlate the directions of the weight (delta) updates for the neurons around the winner i(**x**), the closer the neuron j is, the more correlated its weight update will be with respect to that of the winner i
  - This, as learning evolves, creates topological ordering
  - This spatial correlation in the updates is the reason for the non-linearity of the SOM and makes its mathematical analysis hard
  - As time n goes by, the neighborhood function shrinks and, eventually, it will be a spike equal to 1 only for $d_{i,j}$=0 and zero otherwise:
    - At this point, learning is no longer correlated. Each neuron independently updates its own synaptic weight (fine tuning of local weights) – only at this point, SOM become mathematically tractable…

MIME
Master's degree ICT Internet Multimedia Engineering

REVIEW OF THE SOM ALGORITHM



DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

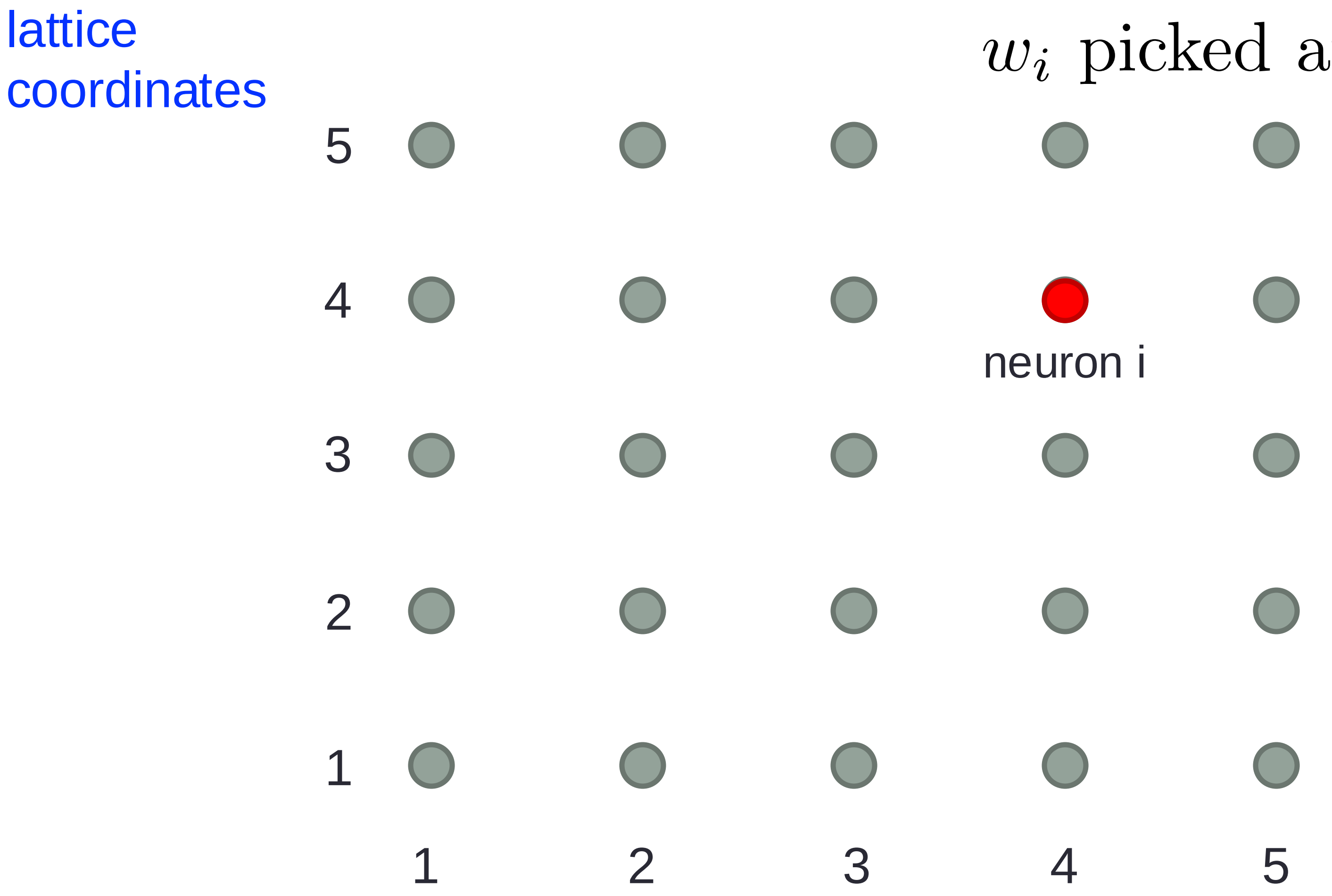
DIPARTIMENTO
MATEMATICA

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

STEP 1: initialize SOM

$$w_i = [w_{i1}\, w_{i2} \ldots w_{im}]^T$$

$$w_i \text{ picked at random}$$

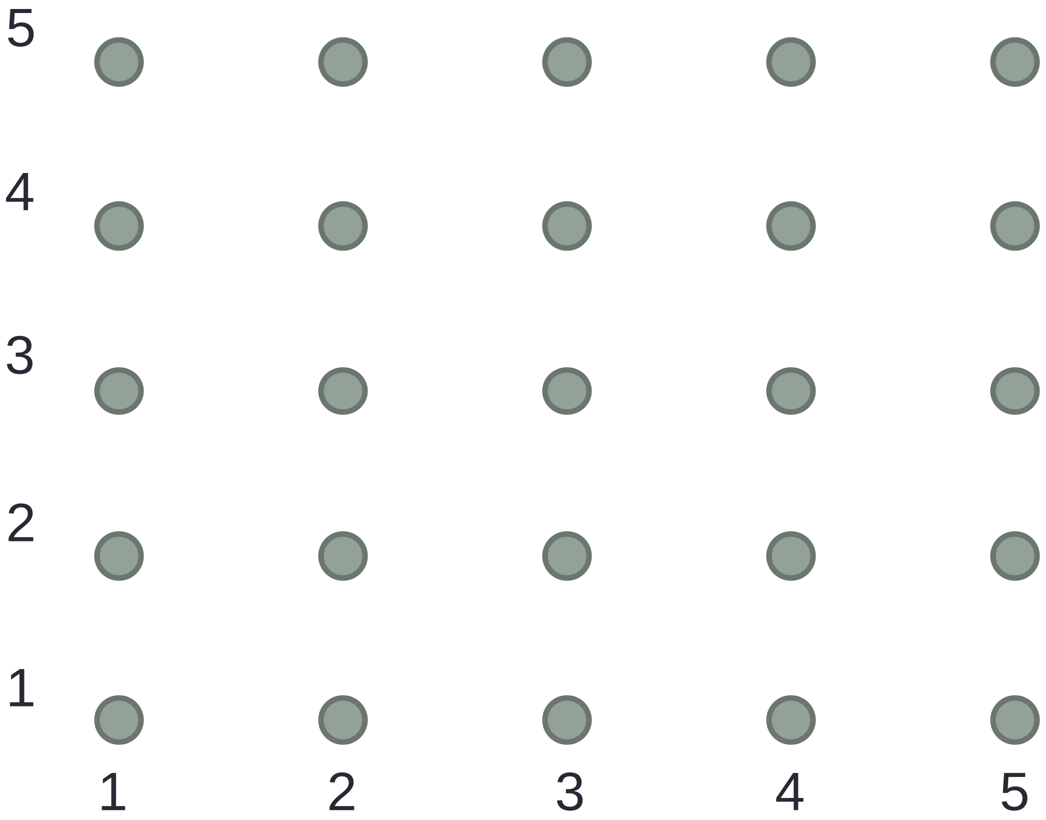


2D lattice (5x5)

STEP 2: sampling from input pdf

generic time n+1

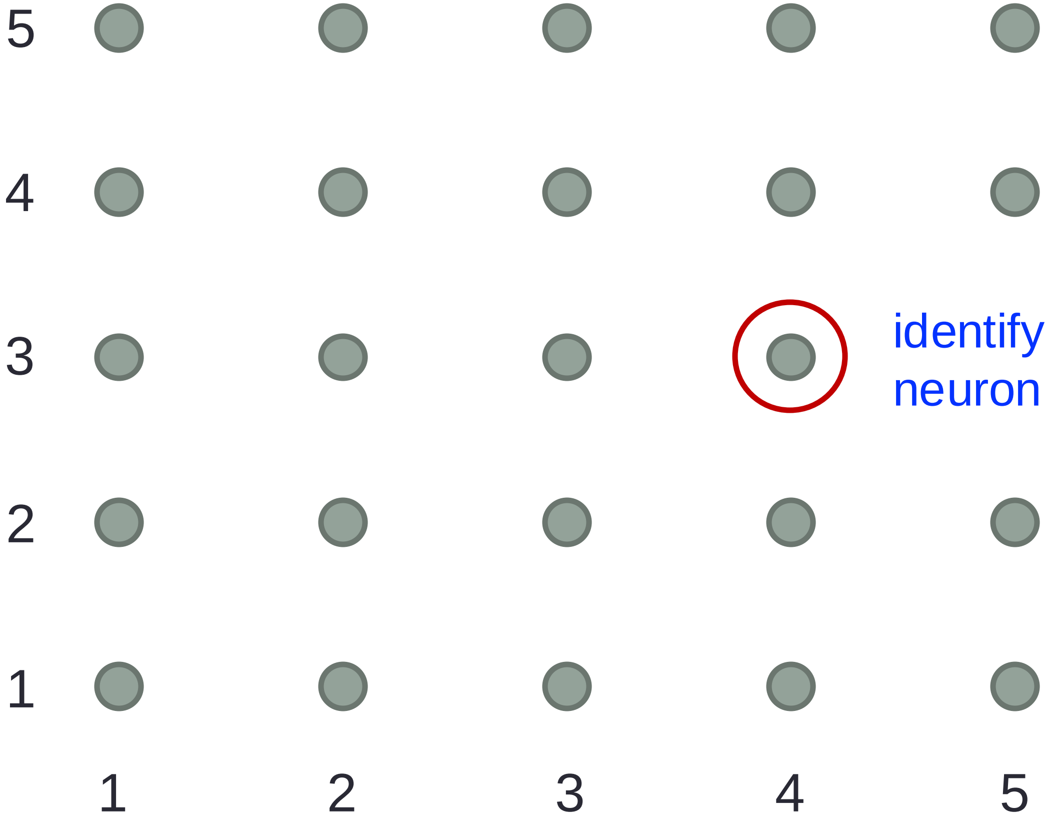
sample from input distribution $\boldsymbol{x}_{n+1} \leftarrow f(\boldsymbol{x})$



2D lattice (5x5)

STEP 3: similarity matching

all neurons compute local value $\left\|\boldsymbol{x}_{n+1} - \boldsymbol{w}_i\right\|, \forall i$

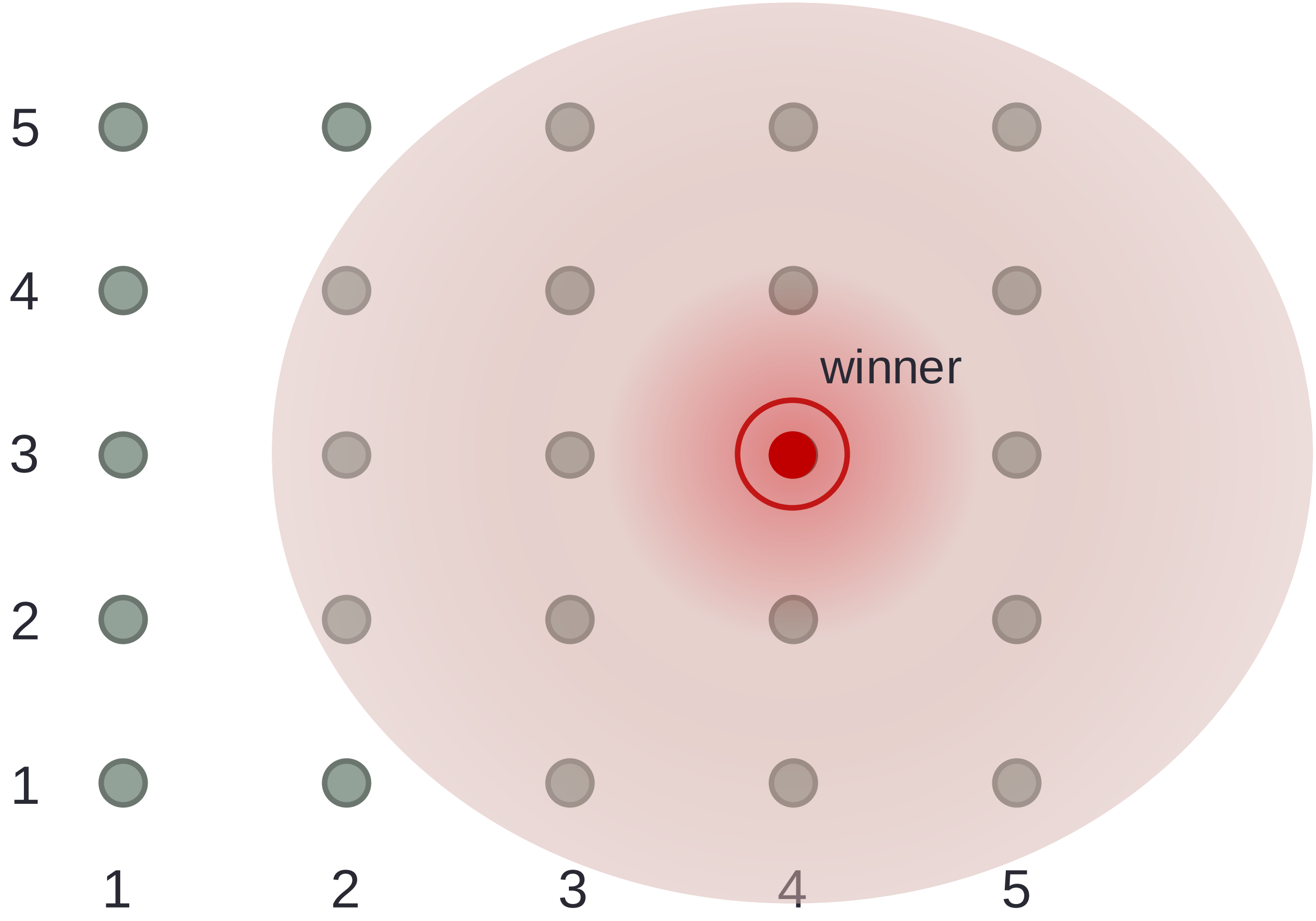


identify winner, as the neuron with closest weight

2D lattice (5x5)

STEP 3: similarity matching

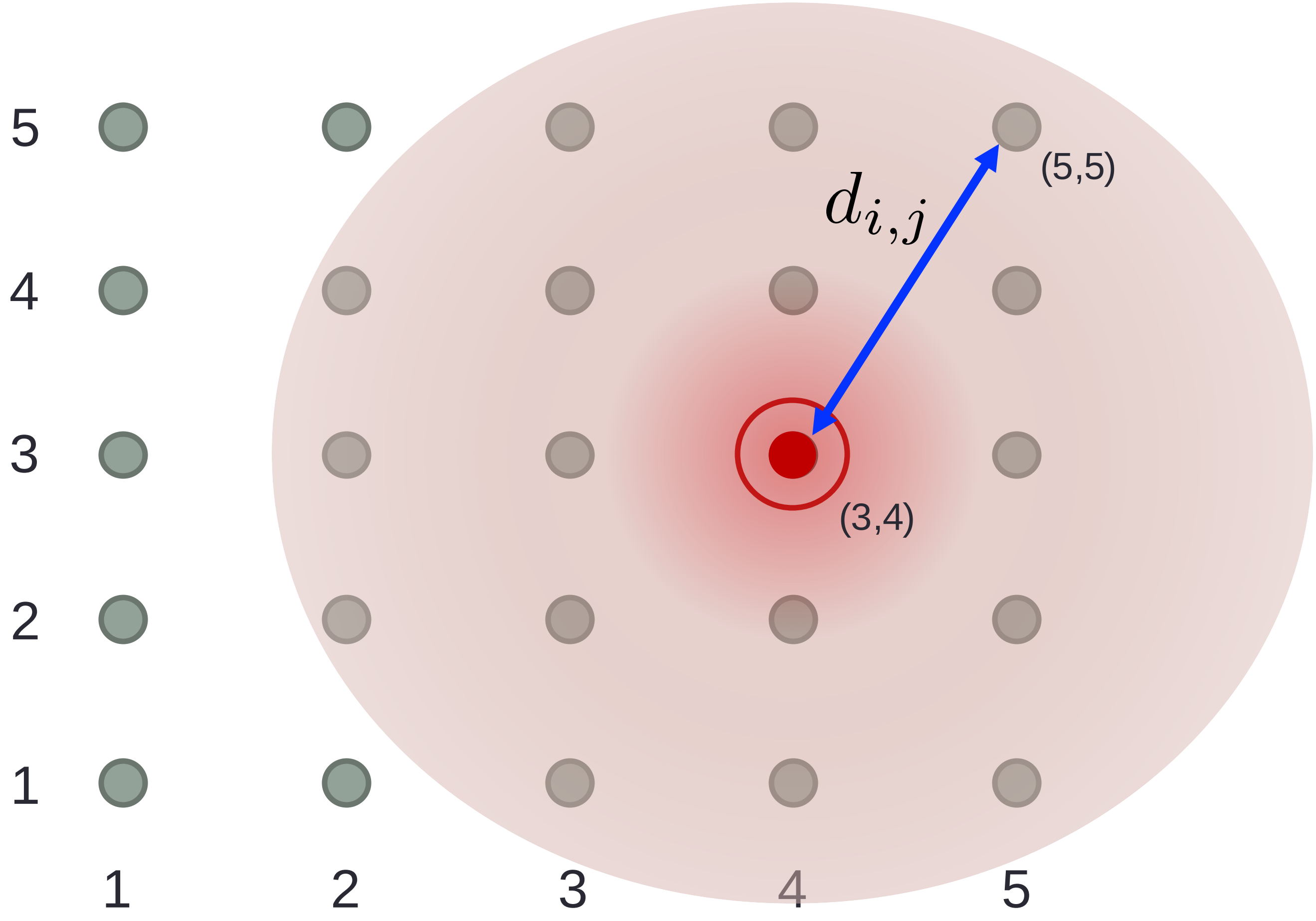
once winner identified → build Gaussian neighborhood around it



2D lattice (5x5)

STEP 3: similarity matching

compute neighborhood distances → using lattice coordinates

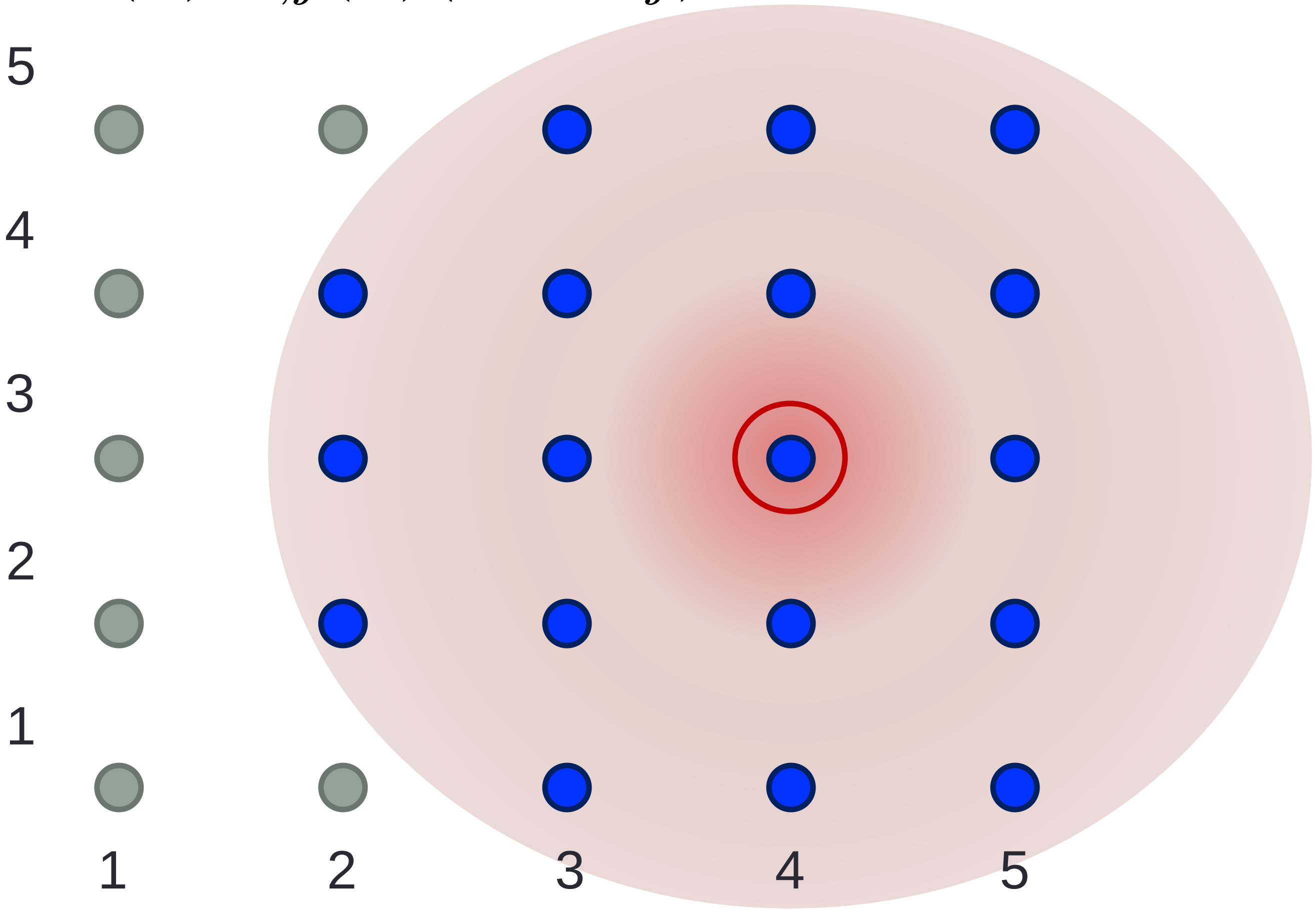


2D lattice (5x5)

STEP 4: update weights

update weights of all (blue) nodes in neighborhood

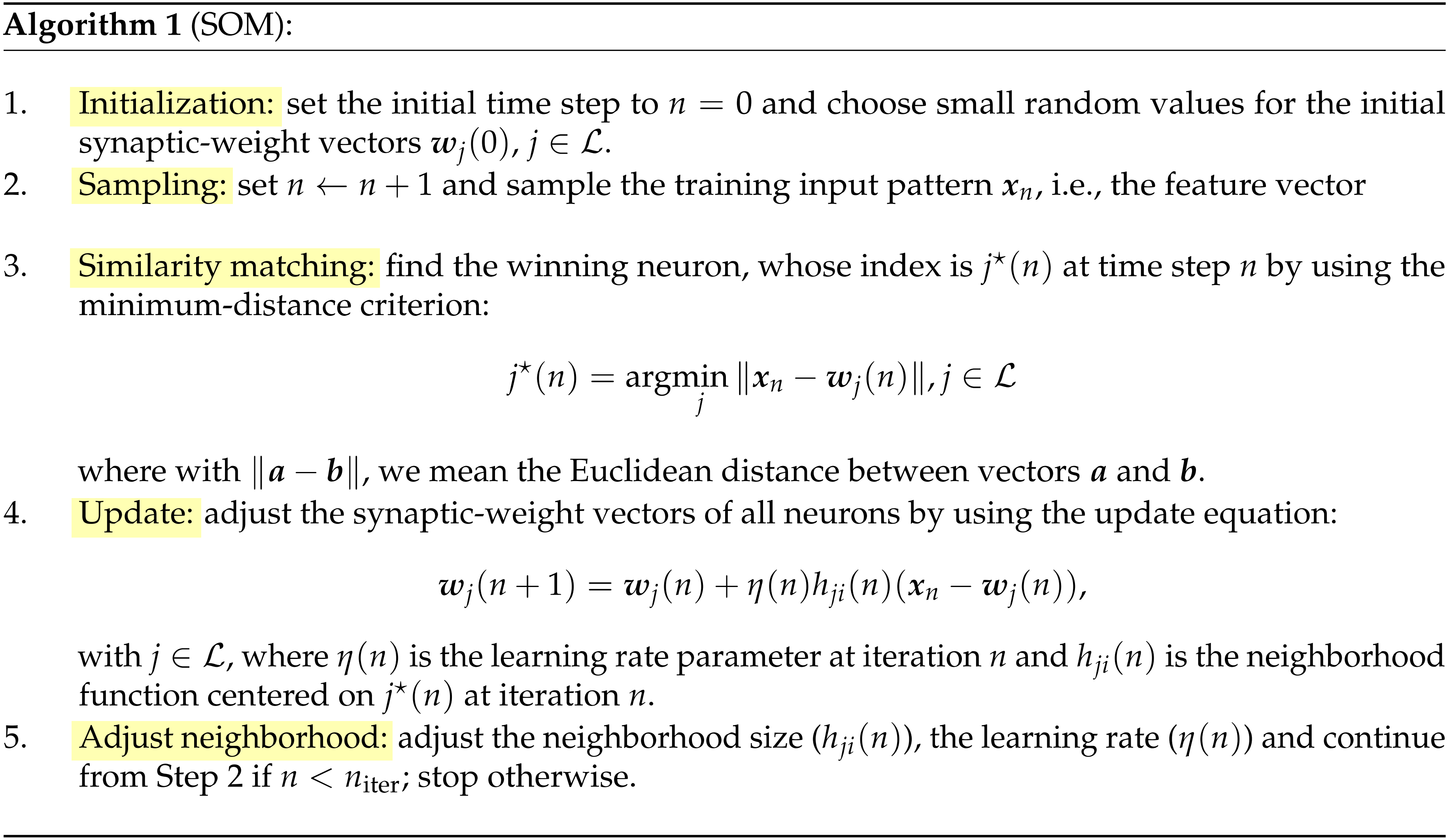
$$\boldsymbol{w}_j \leftarrow \boldsymbol{w}_j + \eta(n)h_{i,j}(n)(\boldsymbol{x} - \boldsymbol{w}_j)$$



STEP 5: update learning parameters

- Adjust neighborhood size $h_{i,j}(n+1)$
- Adjust learning rate $\eta(n+1)$
- Repeat from STEP 2

Review of the SOM algorithm



Algorithm 1 (SOM):

1. Initialization: set the initial time step to $n = 0$ and choose small random values for the initial synaptic-weight vectors $\boldsymbol{w}_j(0)$, $j \in \mathcal{L}$.
2. Sampling: set $n \leftarrow n + 1$ and sample the training input pattern $\boldsymbol{x}_n$, i.e., the feature vector
3. Similarity matching: find the winning neuron, whose index is $j^\star(n)$ at time step n by using the minimum-distance criterion:

$$j^\star(n) = \operatorname*{argmin}_j \|\boldsymbol{x}_n - \boldsymbol{w}_j(n)\|, j \in \mathcal{L}$$

   where with $\|\boldsymbol{a} - \boldsymbol{b}\|$, we mean the Euclidean distance between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

4. Update: adjust the synaptic-weight vectors of all neurons by using the update equation:

$$\boldsymbol{w}_j(n+1) = \boldsymbol{w}_j(n) + \eta(n)h_{ji}(n)(\boldsymbol{x}_n - \boldsymbol{w}_j(n)),$$

   with $j \in \mathcal{L}$, where $\eta(n)$ is the learning rate parameter at iteration n and $h_{ji}(n)$ is the neighborhood function centered on $j^\star(n)$ at iteration n.

5. Adjust neighborhood: adjust the neighborhood size ($h_{ji}(n)$), the learning rate ($\eta(n)$) and continue from Step 2 if $n < n_{\text{iter}}$; stop otherwise.

The two phases of the learning process: *ordering* and *convergence*

- Learning starts from a state of complete disorder
- Progressively, the SOM algorithm leads to an organized representation of patterns drawn from the input space (provided that good parameters are selected)
- The adaptation of the synaptic weights in the neural network *can be decomposed into two phases*:
  - 1. Self-organizing or ordering phase: during this phase the topological ordering of the weight vectors takes place
  - 2. Convergence phase: here the SOM map is fine tuned to provide an *accurate statistical quantification* of the input space

Self-organizing phase

- Rules of thumb [Haykin-2009]
  - The learning rate parameter $\eta(n)$ should begin with a value close to 0.1 and should decrease gradually but never getting smaller than 0.01
  - Considering that this phase takes approx. 1,000 iterations
  - Suitable values for the parameters are:

$$\begin{cases} \eta_0 = 0.1 \\ \tau_2 = 1,000 \end{cases}$$

  - The neighborhood function should initially include all the neurons in the network, centered on the winning neuron, and shrink slowly with time
  - Assuming a 2D lattice, we may set σ_0 equal to the "radius" of the lattice and for τ_1 we may use:

$$\tau_1 = \frac{1,000}{\log \sigma_0}$$

The SOM feature map ϕ (1/2)

- Once the SOM is trained, it defines a non-linear feature map
- The feature map ϕ
  - maps vectors **x** from the input space
  - into a corresponding neuron in the lattice (a “prototype” for VQ)

$$\phi : \mathbb{R}^m \rightarrow \mathcal{L}$$

- SOM map is frozen - for a new input vector **x**
  - we compute the best fit neuron in the SOM lattice
  - whose synaptic weight vector is the closest to **x**

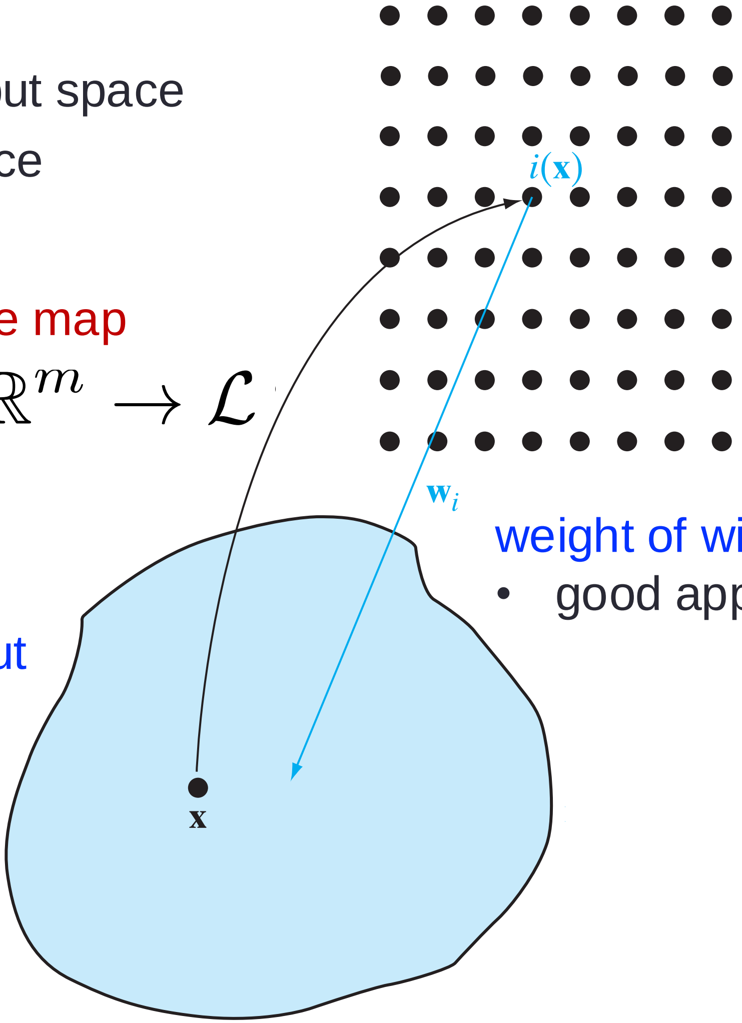
$$i(\boldsymbol{x}) = \arg\min_j \|\boldsymbol{x} - \boldsymbol{w}_j\|, \; j \in \mathcal{L}$$

  - **We have that:**

$$\phi(\boldsymbol{x}) \triangleq i(\boldsymbol{x})$$

The SOM feature map ϕ (2/2)

- A pointer is generated
  - Between the continuous input space
  - and the discrete output space



discrete output space $\mathcal{L}$

feature map

$$\phi : \mathbb{R}^m \to \mathcal{L}$$

weight of winning neuron

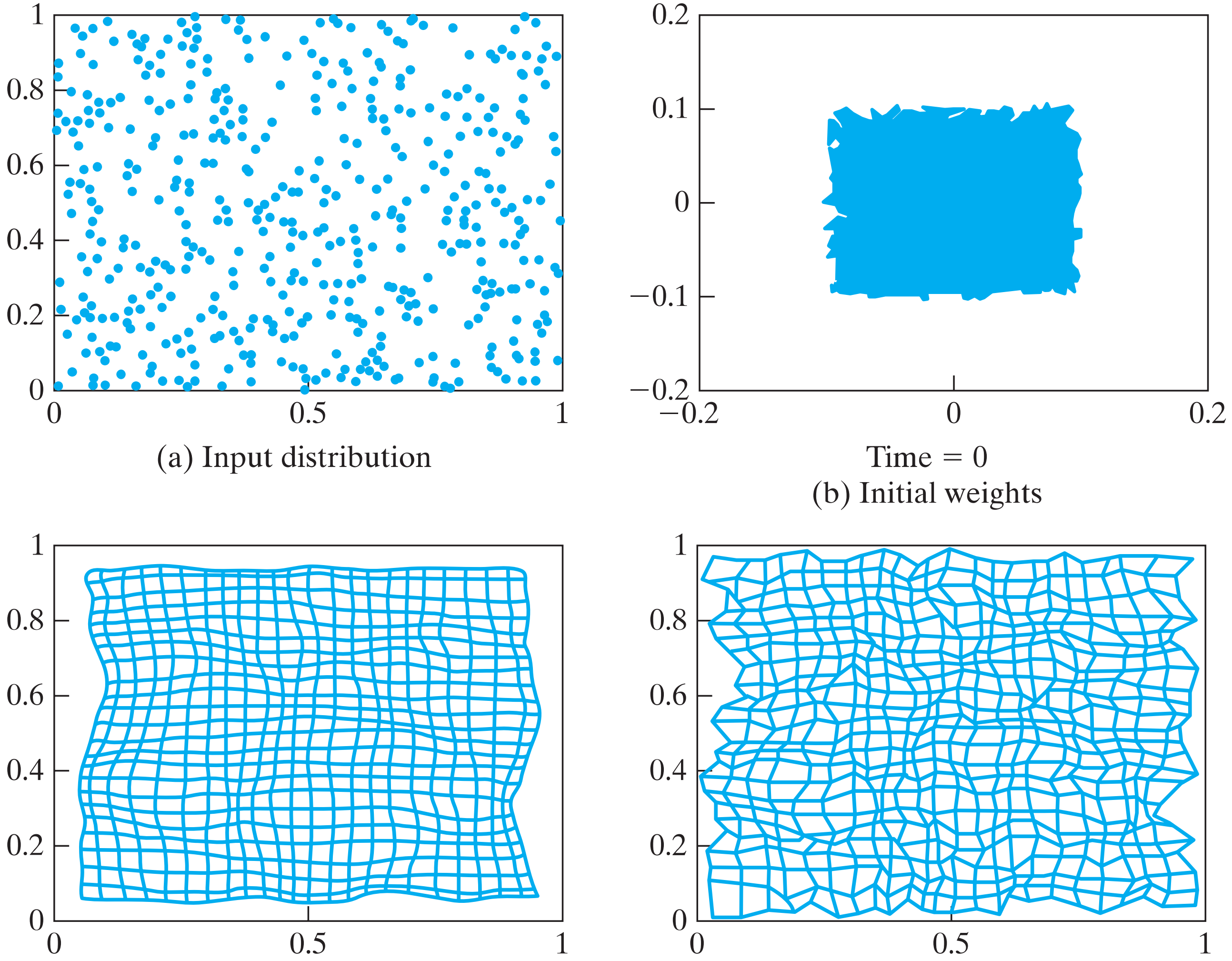
- good approximation for **x**

continuous input space $\mathbb{R}^m$

Simulation setup

- 576 neurons
  - Organized into a 2D lattice (24 rows & 24 columns)
  - Network is trained on 2D input vectors with uniform distribution
- SOM learning (initialization, ordering, convergence)
  - Initial weights are randomly assigned
  - Lines indicate neighboring neurons (synaptic weights are closest)
  - As learning progresses synaptic weights become ordered
  - In the refinement phase, the weights specialize
    - Mapping irregularities in the input distribution
    - Denser regions (more data points) tend to contain more neurons

Simulation results (2D input)



(a) Input distribution

Time = 0
(b) Initial weights

Time = 160 K
(c) Ordering phase

Time = 800 K
(d) Convergence phase

Intermission: variance unexplained (1/2)

- N input data points $x_{1,}$ x_2, …, x_N
- A regression function returns for each x_i an estimate

$$\hat{x}_i = f(x_i)$$

- We define:

Average:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Total sample variance:

$$\sigma^2_{\text{tot}} \triangleq \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

Variance of residuals:

$$\sigma^2_{\text{res}} \triangleq \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2$$

Intermission: variance unexplained (2/2)

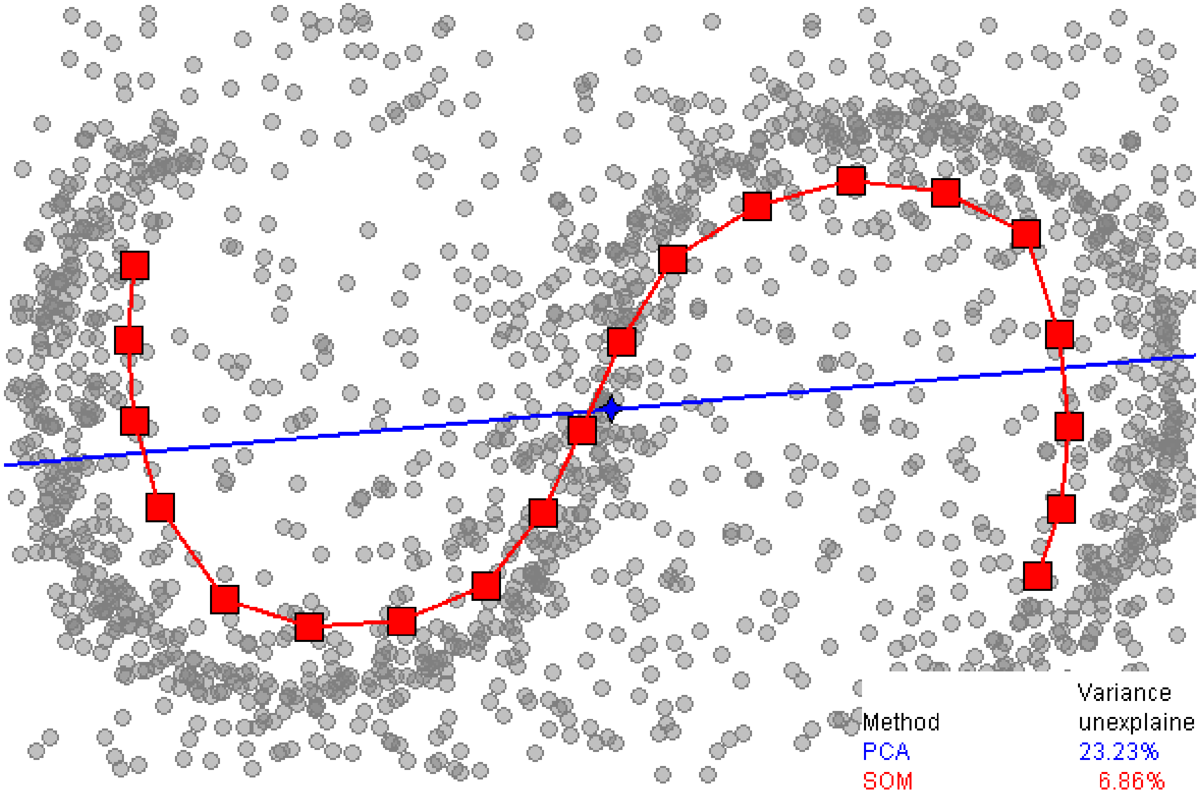
- Fraction of Variance Unexplained (FVU):

$$\mathrm{FVU} \triangleq \frac{\sigma^2_{\mathrm{res}}}{\sigma^2_{\mathrm{tot}}} = \frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{\sum_{i=1}^{N}(x_i - \bar{x})^2} = \frac{\sum_{i=1}^{N}(x_i - f(x_i))^2}{\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

- Is the fraction of the total variance that the regression model is not able to capture

1D SOM vs 1D PCA – FVU

- Data points rest on a non-linear manifold
- PCA performs poorly
- SOM does a better job



One-dimensional SOM versus principal component analysis (PCA) for data approximation. SOM is a red broken line with squares, 20 nodes. PCA is presented by a blue line. Data points are the small grey circles. For PCA, the fraction of variance unexplained in this example is 23.23%, for SOM it is 6.86%

SOM bibliography

Chapter 9 of Haikin's Book [Haykin-2009]

[Haykin-2009] Simon Haykin, "Neural Networks and Learning Machines," 3rd Edition, *Prentice Hall*, 2009. (+7k citations)

Further readings:

[Kohonen-2001] Teuvo Kohonen, "Self Organizing Maps," *Springer-Verlag*, 3rd Edition, 2001. (+22k citations)

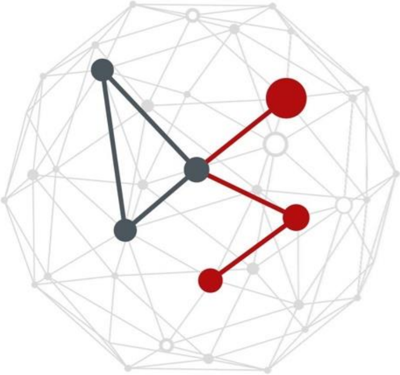
[Hosseini-2003] H. Shah-Hosseini, R. Safabakhsh, "TASOM: a new time adaptive self-organizing map," *IEEE Transactions on Systems, Man, and Cybernetics*, Part B, Vol. 33, No. 2, pp. 271-282, April 2003. (72 citations)

SOM pros and cons

- Pros
  - SOM learning leads to an online clustering algorithm
  - SOM learning is unsupervised, and precision can be tuned changing
    - training parameters,
    - number of neurons in the lattice
- Cons
  - Not clear how to “optimally” set the number of neurons
  - It would be ideal to *automatically* tune it
    - Add neurons until data pdf f(**x**) is represented within some error tolerance
- Desideratum – adaptation
  - Once the SOM map stabilizes, learning STOPS :-(
  - We would like to continuously adapt (add, move & remove weights)
  - Still, we want an unsupervised algorithm…

MIME
Master's degree ICT Internet Multimedia Engineering

GROWING NEURAL GAS (GNG) NETWORKS



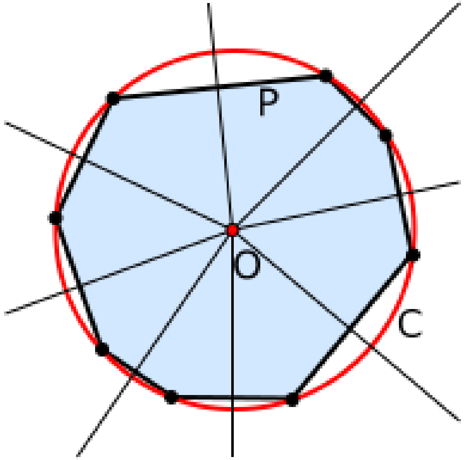
DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO
MATEMATICA

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Intermission: cirumcircle

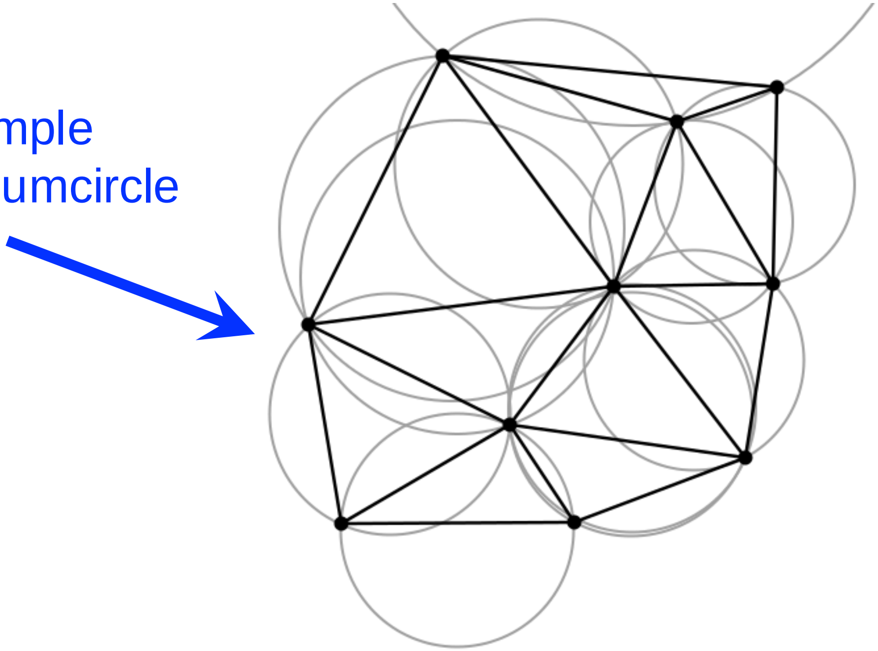
- Circumscribed circle – “circumcircle”
  - In geometry, the circumscribed circle or circumcircle of a polygon is a circle which passes through all the vertices of the polygon. The center of this circle is called the *circumcenter* and its radius is called the *circumradius*
  - A polygon which has a circumscribed circle is called a *cyclic polygon*. All regular simple polygons, all isosceles trapezoids, all triangles and all rectangles are *cyclic*



Intermission: Delaunay triangulation

- A **Delaunay triangulation** for a set **P** of discrete points in a plane is a triangulation DT(**P**) such that
  - Points in **P** are connected through triangles
  - no point in **P** is inside the circumcircle of any triangle in DT(**P**)
  - Delaunay triangulation: maximizes the minimum angle of all the angles of the triangles in DT(**P**)

Delaunay triangulation example
no point in P is inside a circumcircle



Voronoi diagram (1/6)

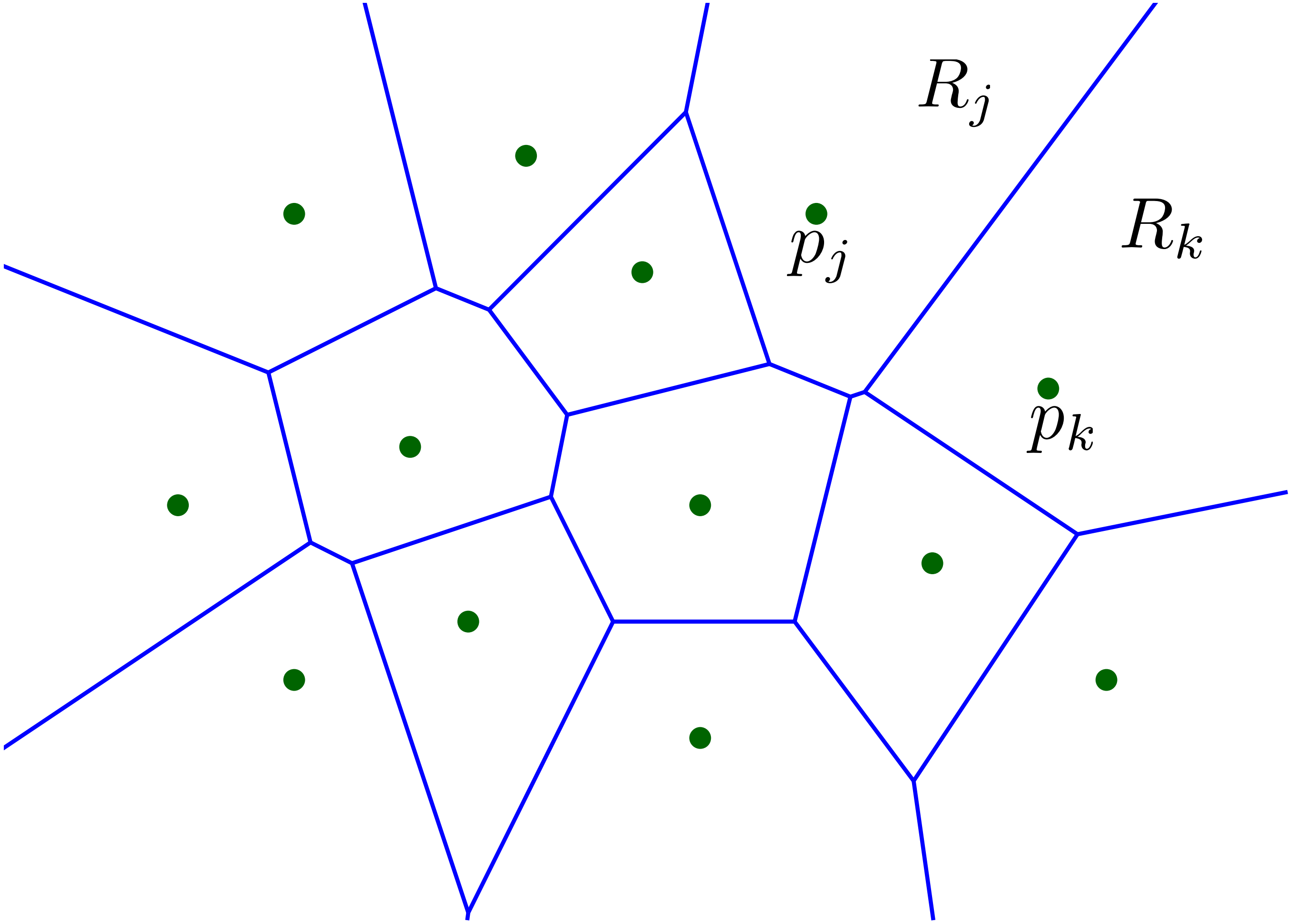
- Let X be a metric space
- Let $d(p_1, p_2)$ be a distance function between points p_1, p_2
- Let **P**=$\{p_k\}$ be a collection of K points in the space X
- The Voronoi region R_k associated with point p_k
  - Is the set of all points in X whose distance from p_k
    - Is no greater than their distance from any other point p_j
    - where j is any index different from k

- Formally:

$$R_k = \{x \in X \,|\, d(x, p_k) \leq d(x, p_j) \text{ for all } j \neq k\}$$

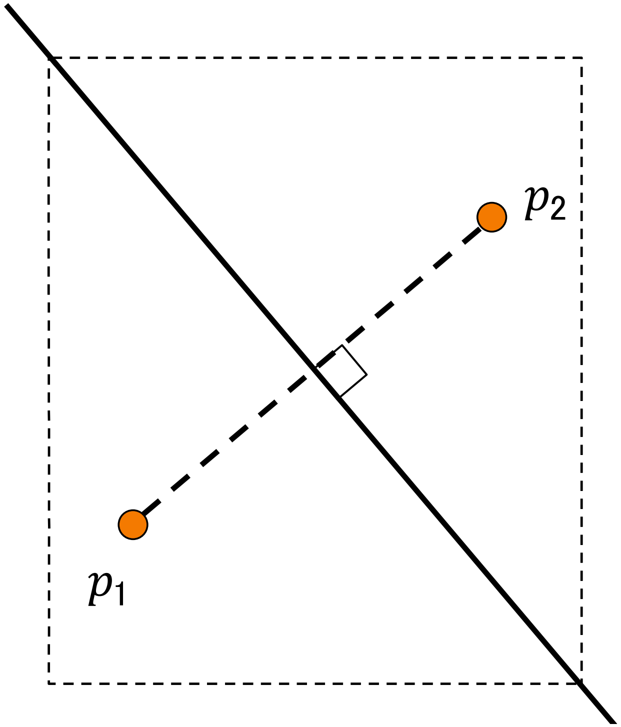
Voronoi diagram (2/6)

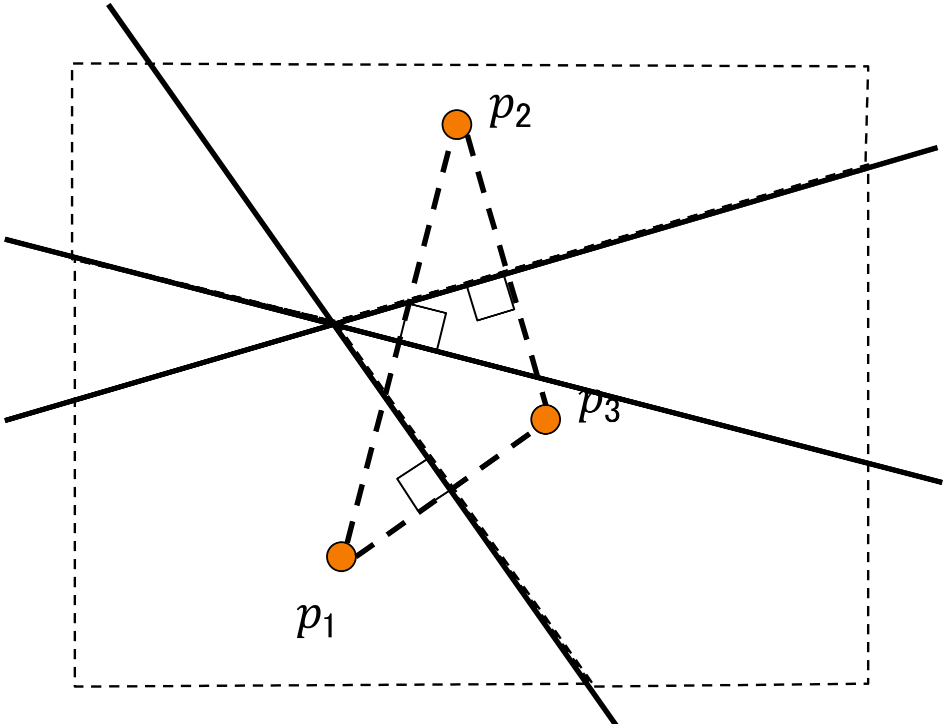
- Example: d(p_j,p_k) is the Euclidean distance



Voronoi diagram (3/6)

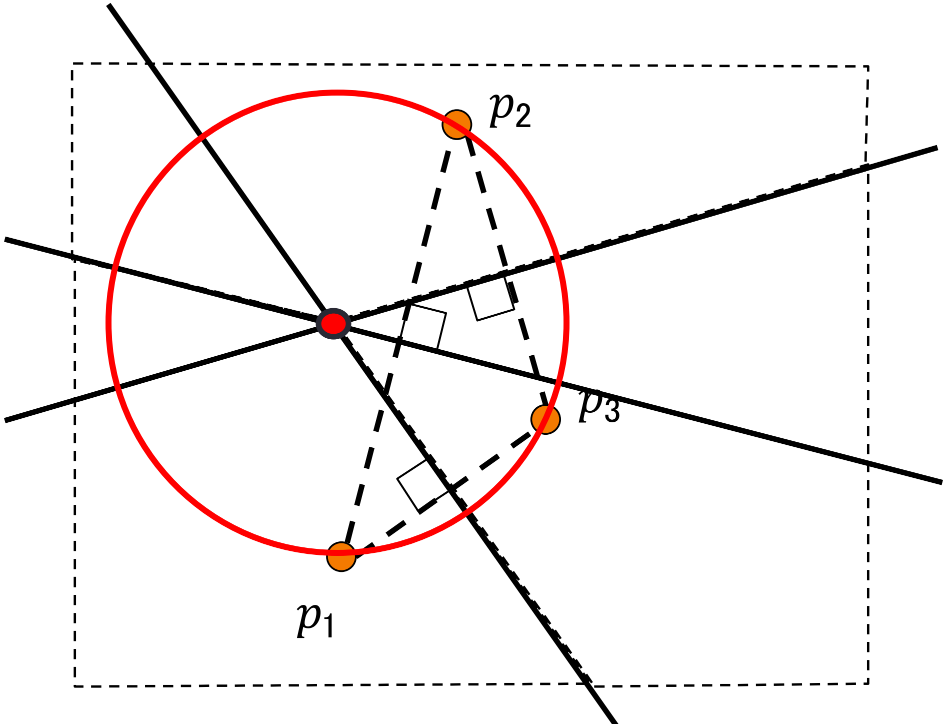
- 2 points: regions are defined by the perpendicular bisector
- 3 points: regions are the intersection of the half spaces $H(p_i, p_j)$ defined by the 3 perpendicular bisectors



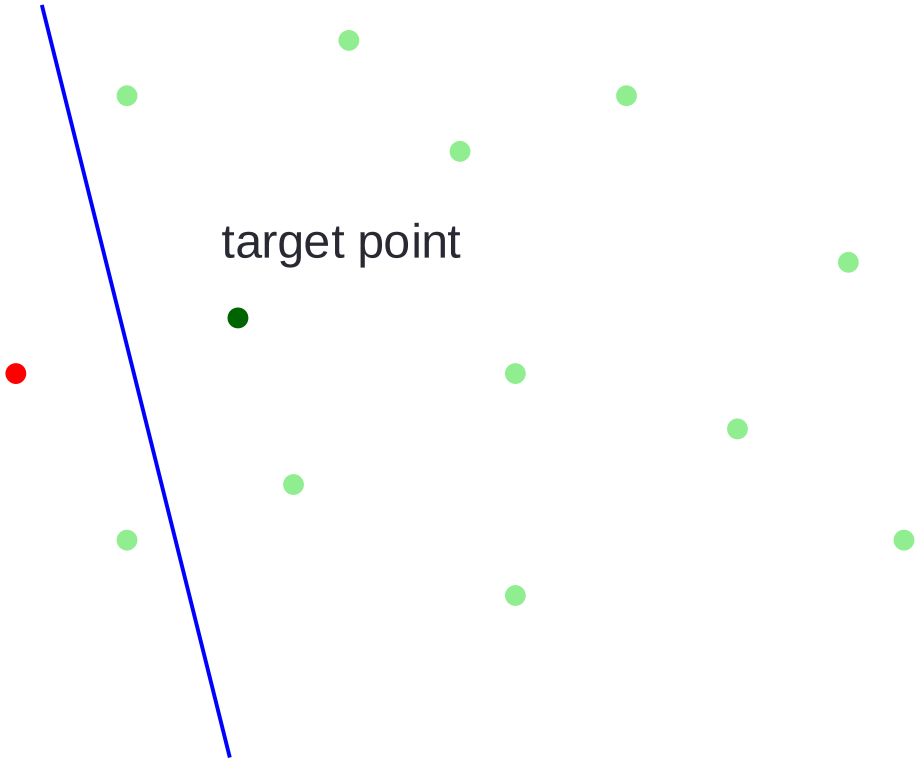


Voronoi diagram (4/6)

- 3 points: regions defined by the 3 perpendicular bisectors
  - The 3 bisectors intersect at a point (can be outside the triangle)
  - This point is the center of the circumcircle for the polygon (p_1, p_2, p_3)

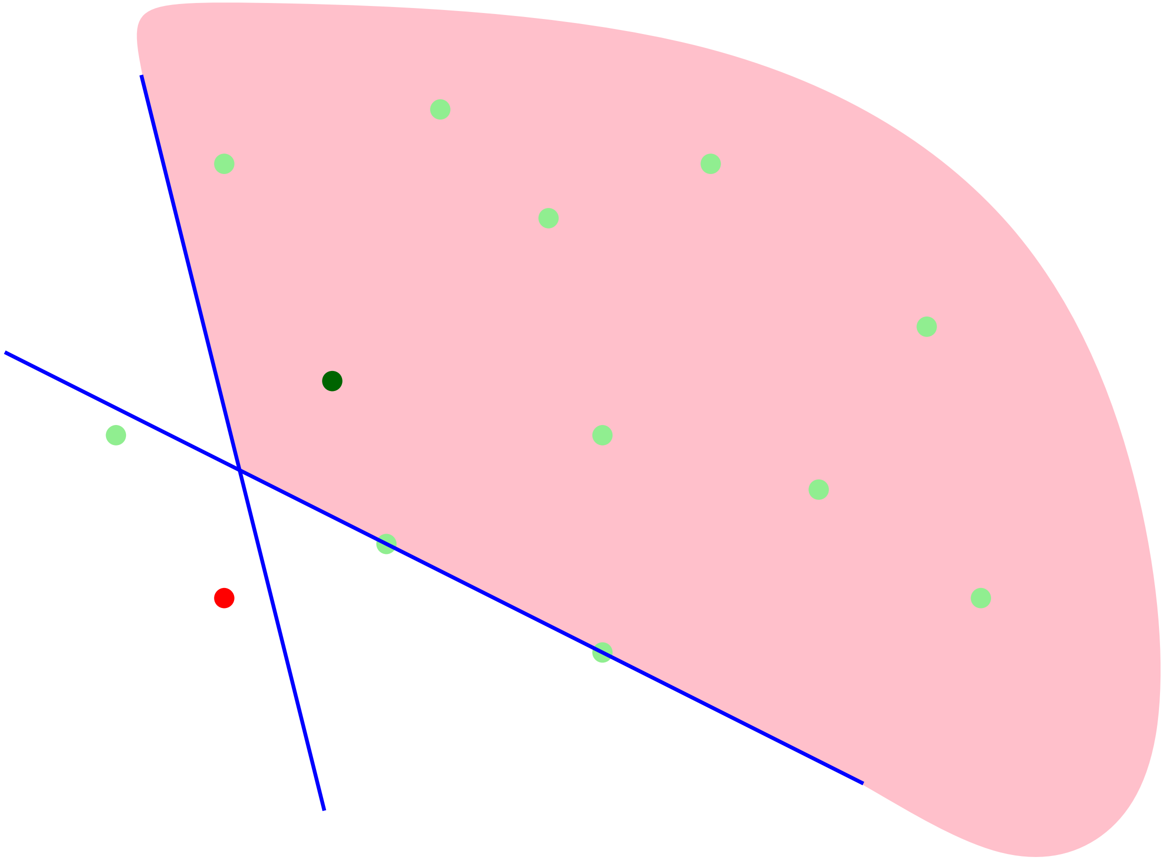


Faces of Voronoi diagram (1/5)



We consider all remaining points (differing from target), one at a time
Perpendicular bisector between target and red points → defines two half spaces

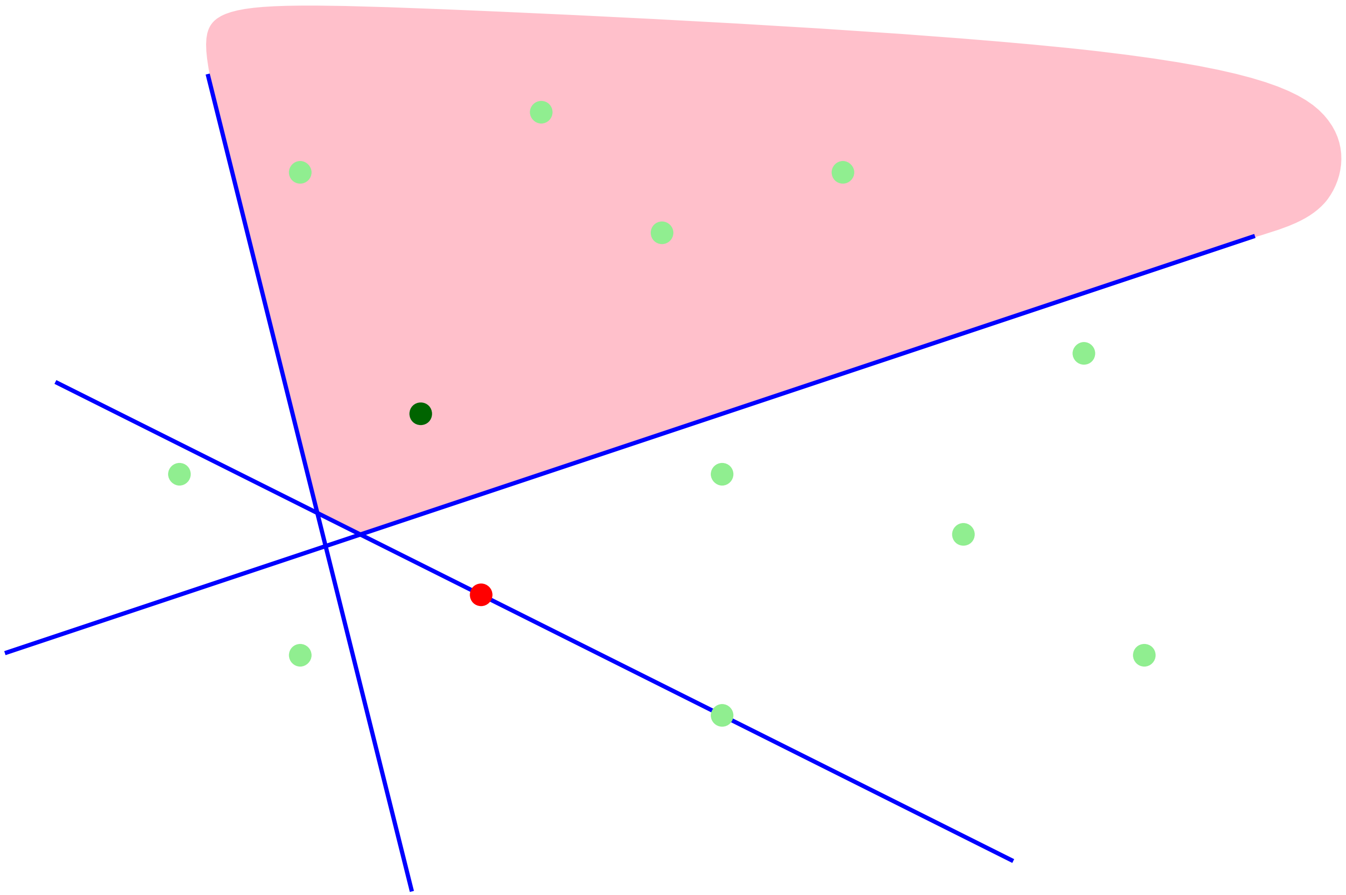
Faces of Voronoi diagram (2/5)



Perpendicular bisector between red and target points → defines two half spaces

Intersect those half spaces with previous one including the target point

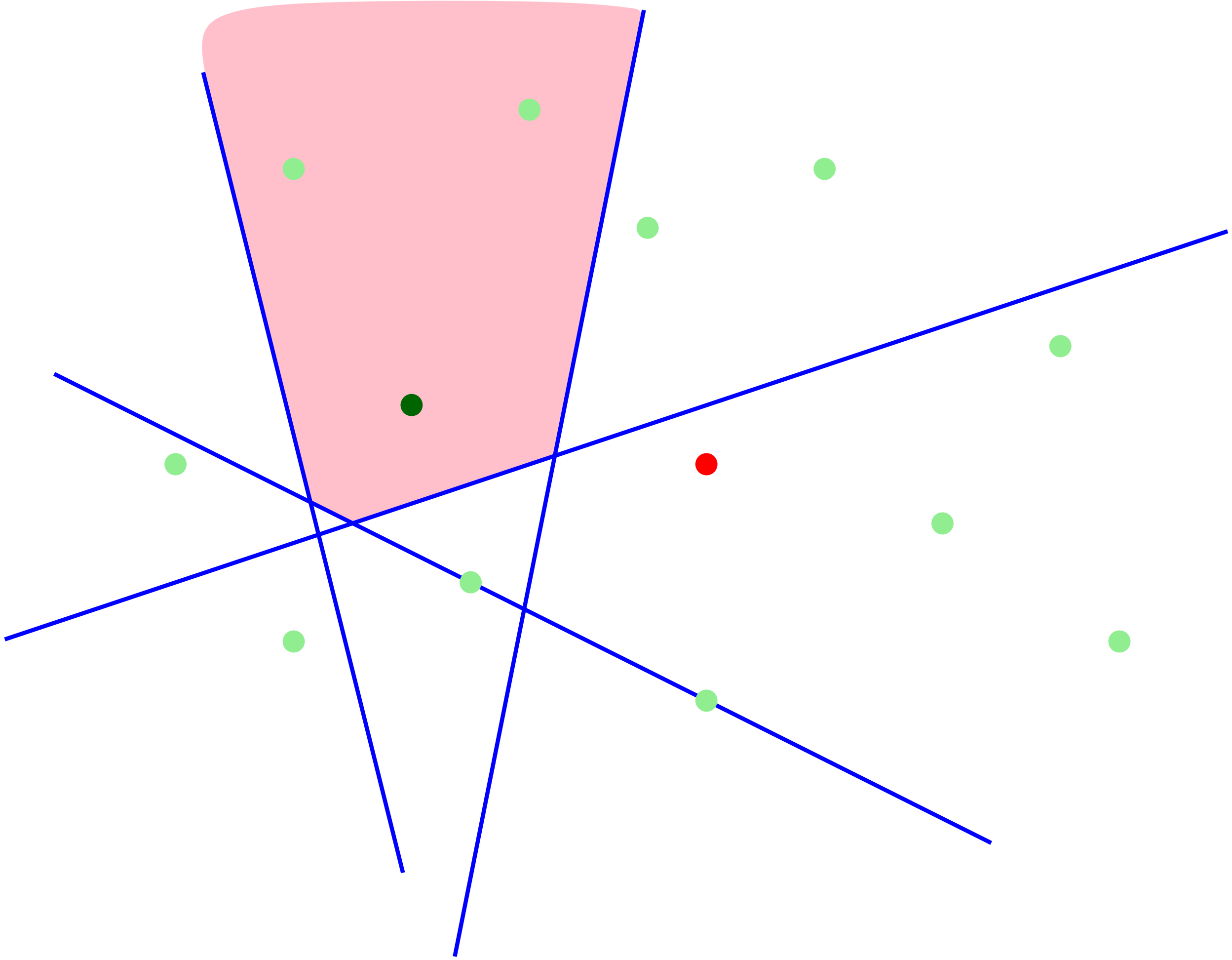
Faces of Voronoi diagram (3/5)



Perpendicular bisector between red and target points → define two half spaces

Intersect those half spaces with previous region including the target point

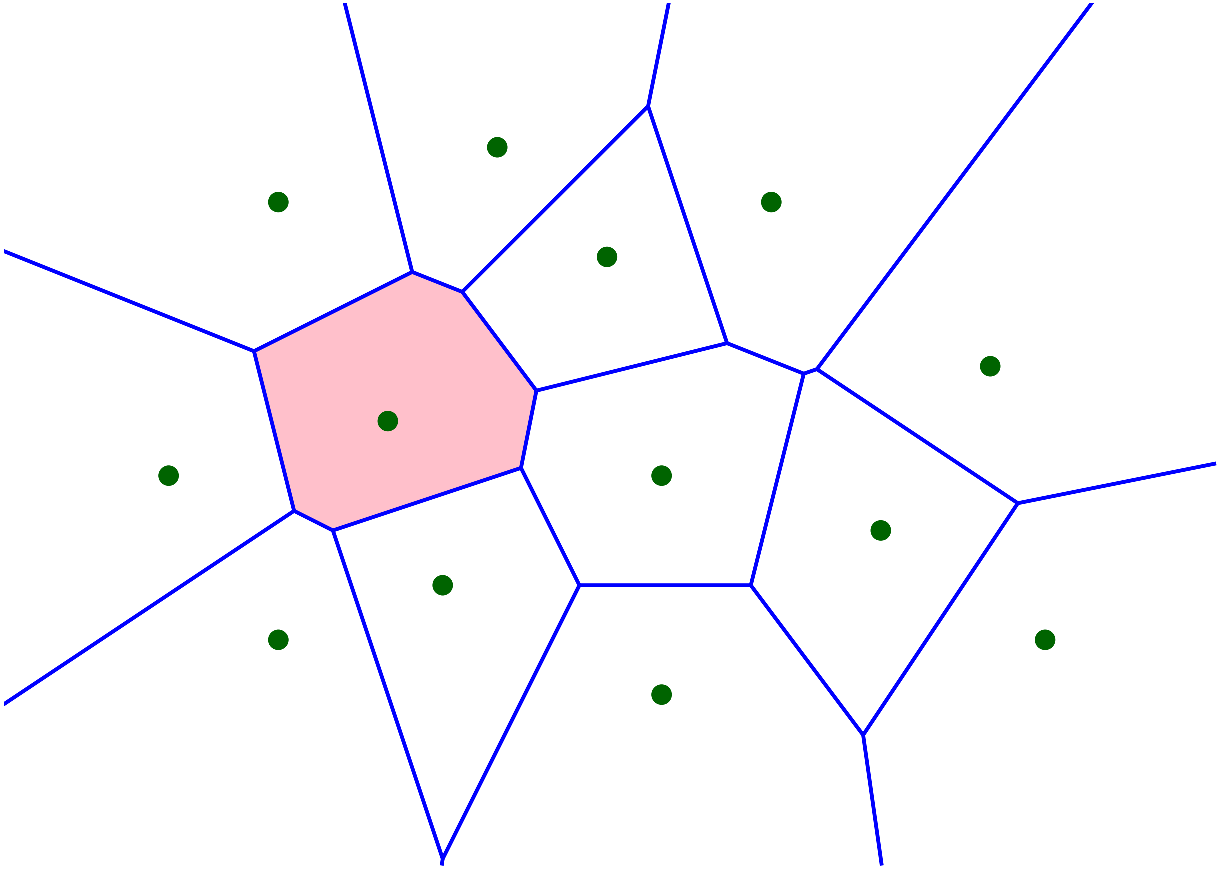
Faces of Voronoi diagram (4/5)



Perpendicular bisector between red and target points → define two half spaces

Intersect those half spaces with previous region including the target point

Faces of Voronoi diagram (5/5)



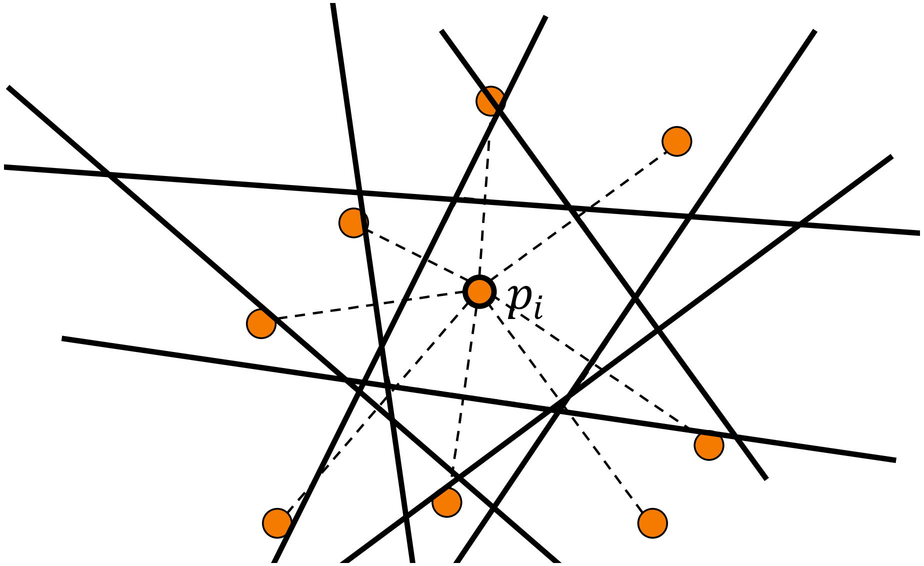
Upon cycling over all points differing from the target

The intersection of all the half spaces returns the Voronoi region for the target point

Voronoi diagram – continued (1/2)

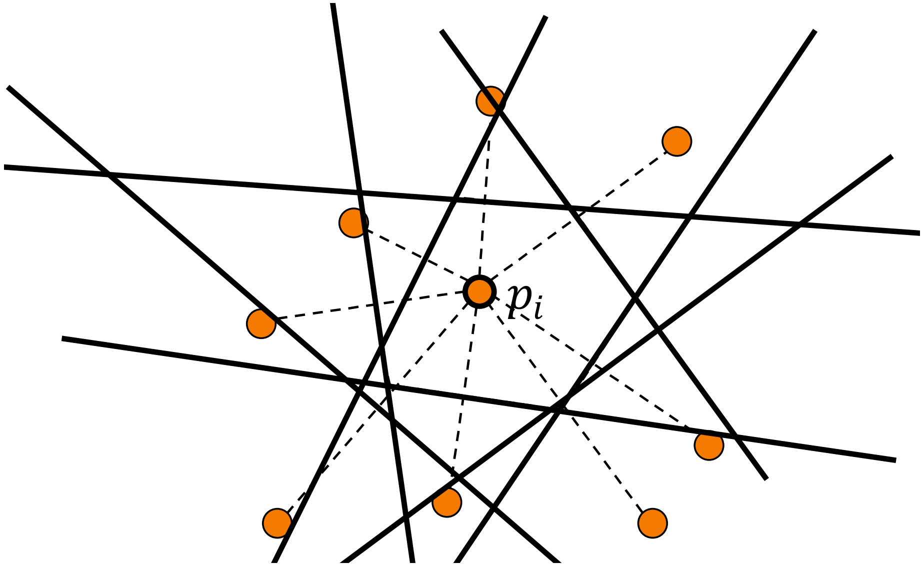
- N points: the Voronoi region R_i associated with point p_i corresponds to the intersection of the half spaces defined by the perpendicular bisectors

$$R_i = V(p_i) = \cap_{j \neq i} H(p_i, p_j)$$



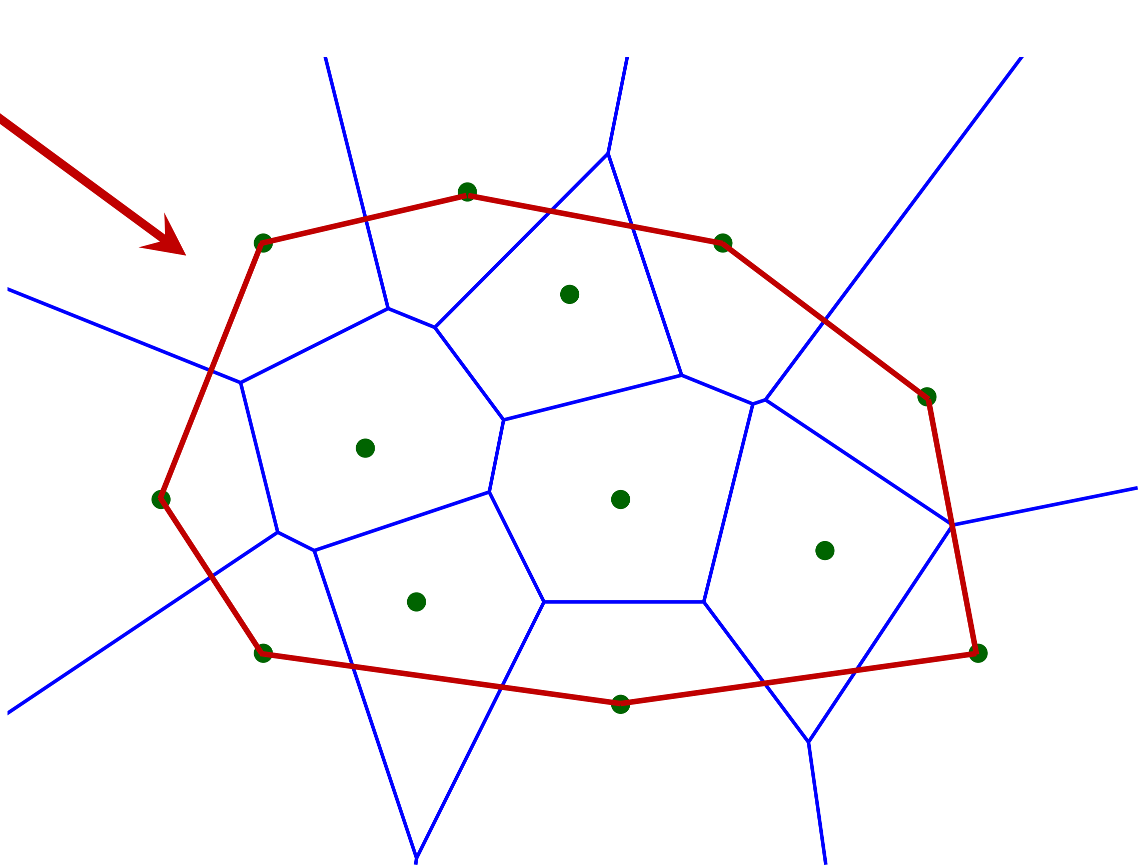
Voronoi diagram – continued (2/2)

- By construction, Voronoi regions are *convex* polygons
- There is 1 Voronoi region for each point
- Voronoi faces can be unbounded



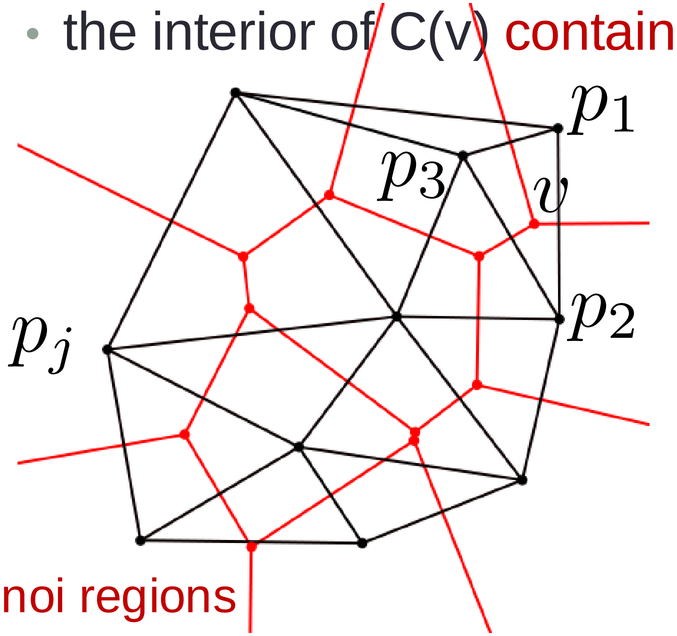
Properties of Voronoi diagrams (1/2)

- Given a set of data points **P**
- Convex hull of **P**: smallest *convex set* that contains **P**

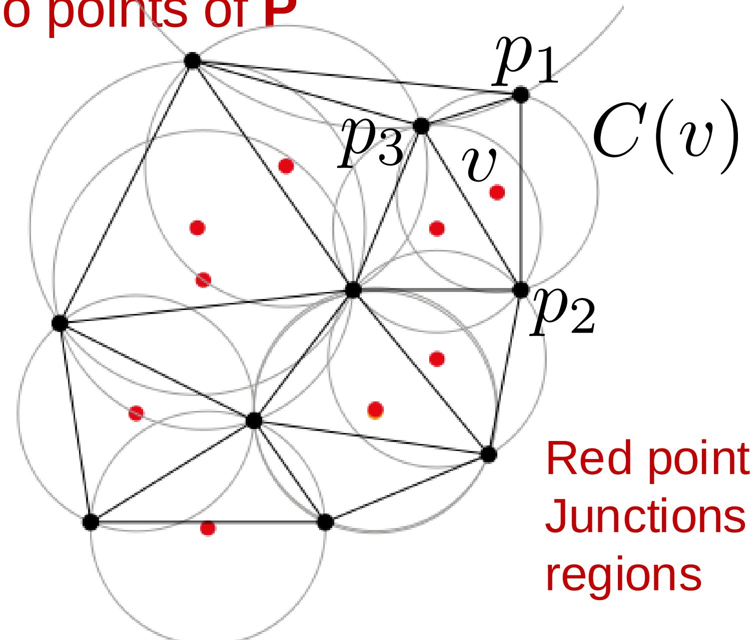


Properties of Voronoi diagrams (2/2)

- Each Voronoi region is convex
- R_i=V(p_i) is *unbounded* IFF p_i is on the boundary of the convex hull of **P**
- If v is at the junction of V(p_1), V(p_2), …, V(p_k) with k≥3
  - v is the center of a circle C(v) with p_1, p_2, ..., p_k on the boundary
  - the interior of C(v) contains no points of **P**



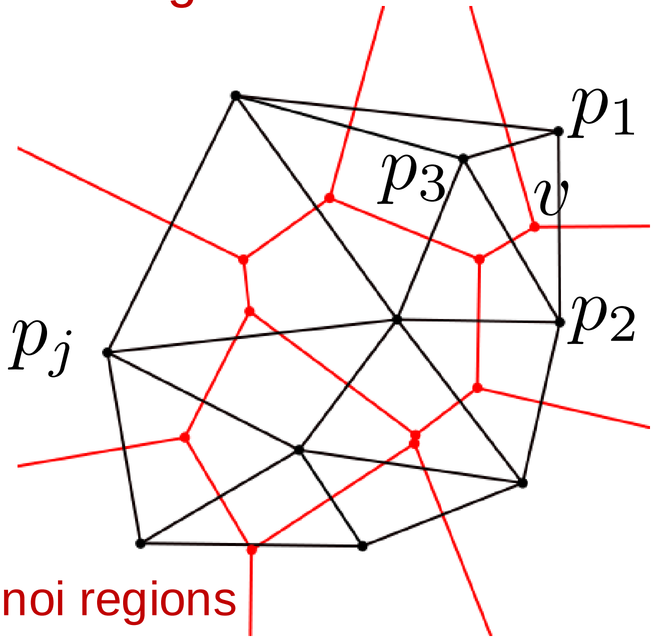
Voronoi regions



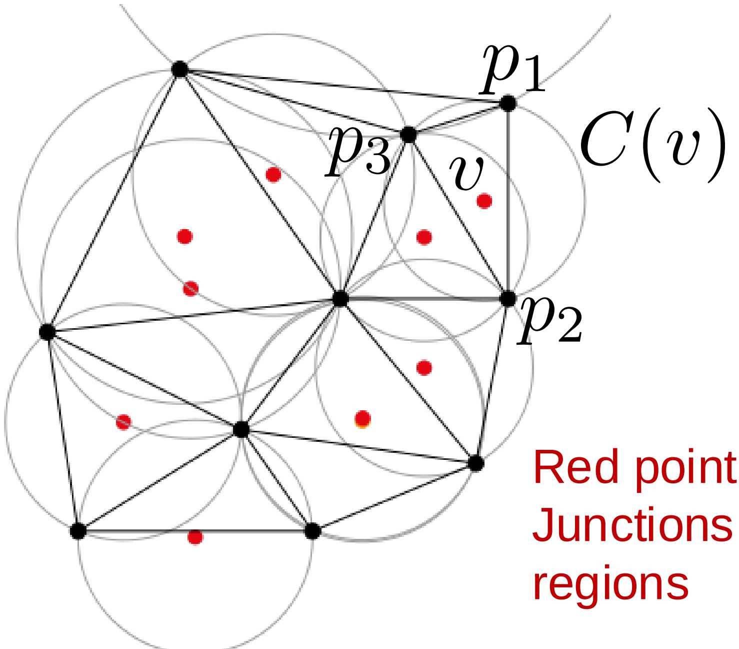
Red points are at the Junctions of Voronoi regions

Delaunay triangulation

- It is the straight line dual of the Voronoi diagram
  - It is represented by the black straight lines in the left plot below
  - Right plot: we see that DT(**P**) is a valid Delaunay triangulation
  - Starting from Voronoi diagram & connecting all the points *that share an edge* (a Voronoi face) → we get the Delaunay triangulation



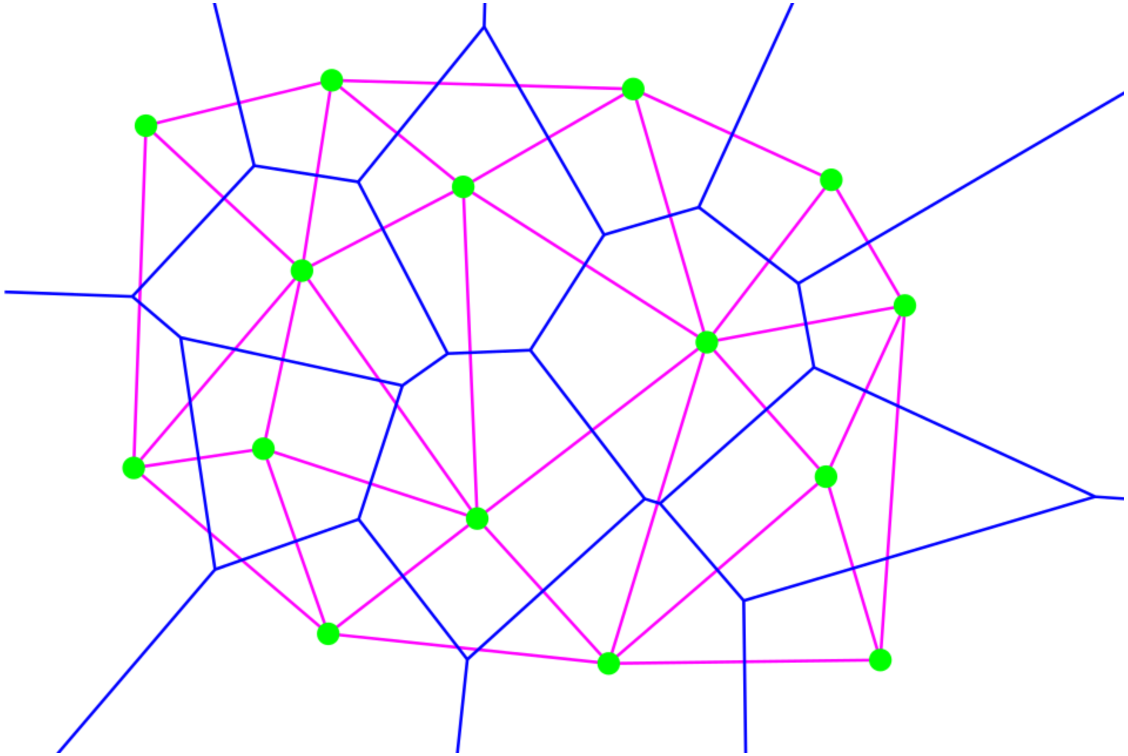
Voronoi regions



Red points are at the Junctions of Voronoi regions

Voronoi diag. *vs* Delaunay triangulation

- The Delaunay triangulation of a discrete point set **P** in general position corresponds to the dual graph of the Voronoi diagram for **P**



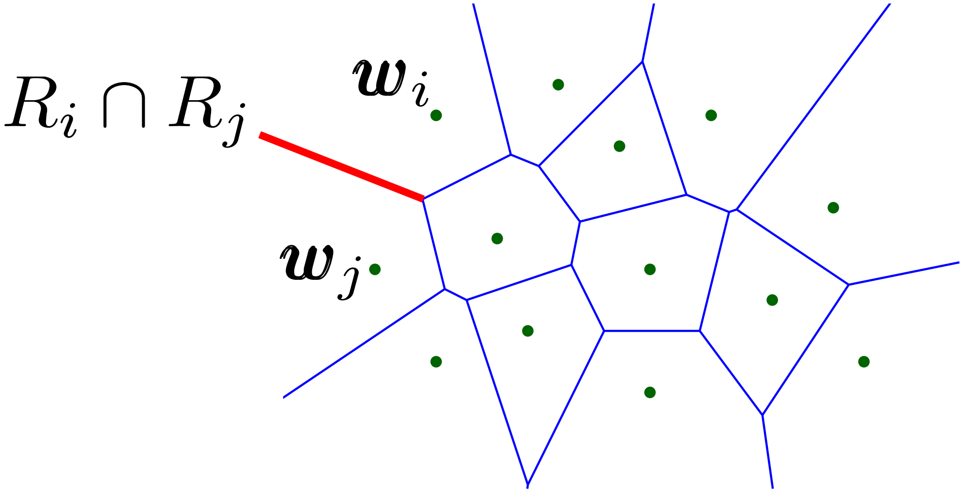
Data points: **green**, Voronoi diagram: blue, Delaunay triangulation: magenta

Topology preserving structures

- We are given a feature manifold $M \subseteq \mathbb{R}^m$
- And a set of points $\boldsymbol{w}_i \in M$
- Any two points $\boldsymbol{w}_i, \boldsymbol{w}_j$ are adjacent if
  - The intersection of their Voronoi regions is non-empty, i.e.,

$$R_i \cap R_j \neq \emptyset$$

  - This means that they have a Voronoi edge in common



Adjacency in M

- Definition.
  - Let $M \subseteq \mathbb{R}^m$ be a given manifold
  - And let $S = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N\}$ be a set of points $\boldsymbol{w}_i \in M$
  - The Voronoi regions of S are defined as $V(\boldsymbol{w}_i) \triangleq R_i$
  - Two points $\boldsymbol{w}_i, \boldsymbol{w}_j$ are adjacent in M if their masked Voronoi regions

$$R_i^{(M)} = R_i \cap M \quad R_j^{(M)} = R_j \cap M$$

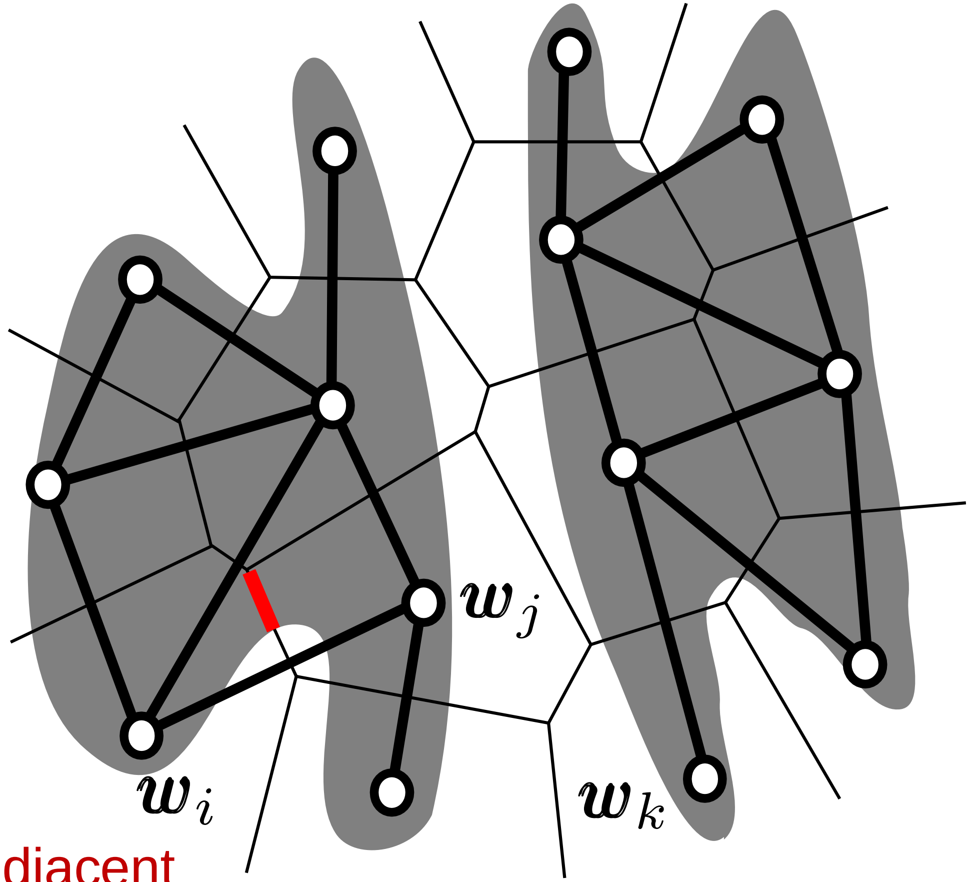
  - Share (at least) an element $\boldsymbol{v} \in M$
  - Or equivalently if:

$$R_i^{(M)} \cap R_j^{(M)} \neq \emptyset$$

Adjacency in M

- Example
  - M is the shaded area
  - Is formed of two disjoint sets
  - Here, $\boldsymbol{w}_i, \boldsymbol{w}_j$ are adjacent

$$R_i^{(M)} \cap R_j^{(M)} \neq \emptyset$$



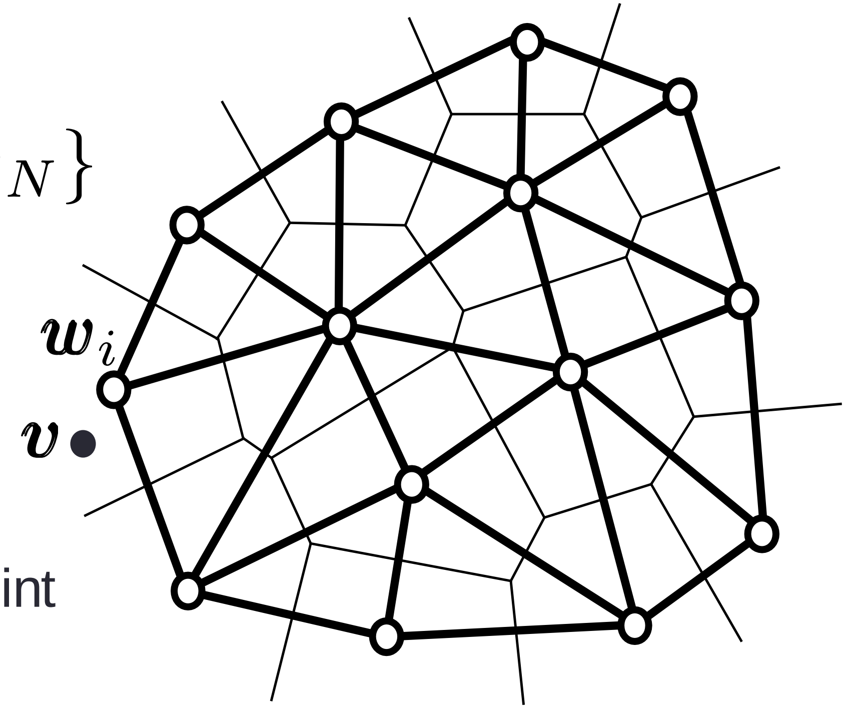
  - However, $\boldsymbol{w}_j, \boldsymbol{w}_k$ are NOT adjacent
    - Although their Voronoi regions have an edge in common
    - Their “masked” Voronoi regions (intersected with M) do not share any point

Topology preserving feature maps

- We are given a set of points

$$S = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N\}$$

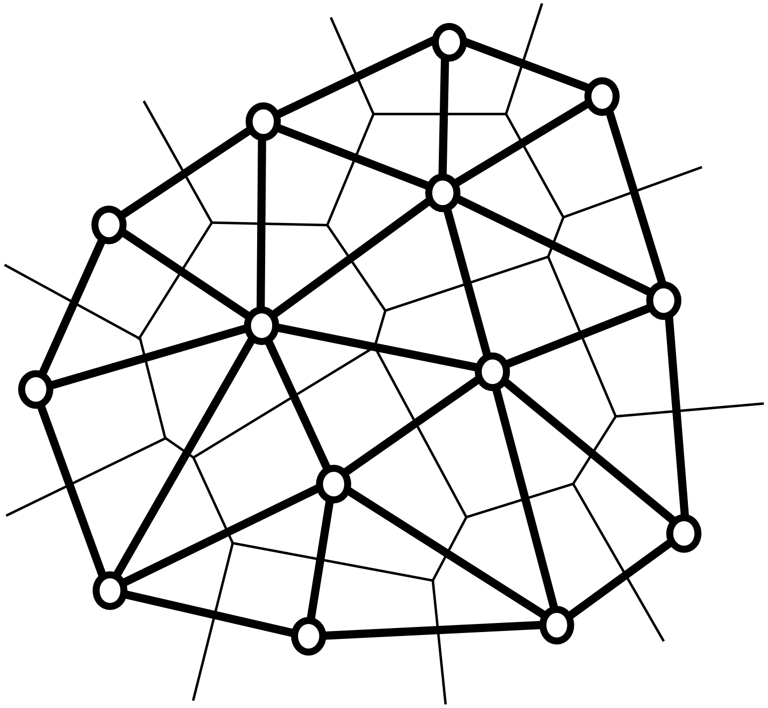
- These corresponds to *centroids*
  - Obtained via, e.g., SOM, K-means
  - from an input distribution
- If we sample **v** from M
  - this is associated with the closest point
  - in the figure this closest point is $\mathbf{w}_i$



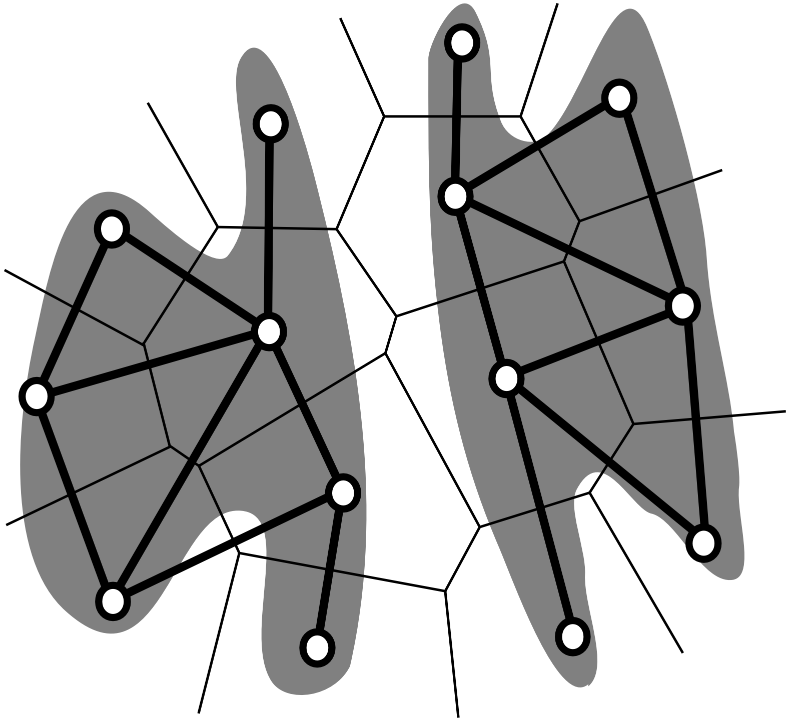
- Now, we would like to join the points in S *with edges*
  - to form a graph G (in the figure is obtained via Delaunay triangulation)
- Definition. Graph G forms a topology preserving map of M if points that are adjacent (edge) in G are also adjacent in M

Topology preserving feature maps

- Hence, we see that for the given manifold M (shaded area)
  - Graph G on the left is **NOT** topology preserving
  - Graph G on the right **IS** topology preserving



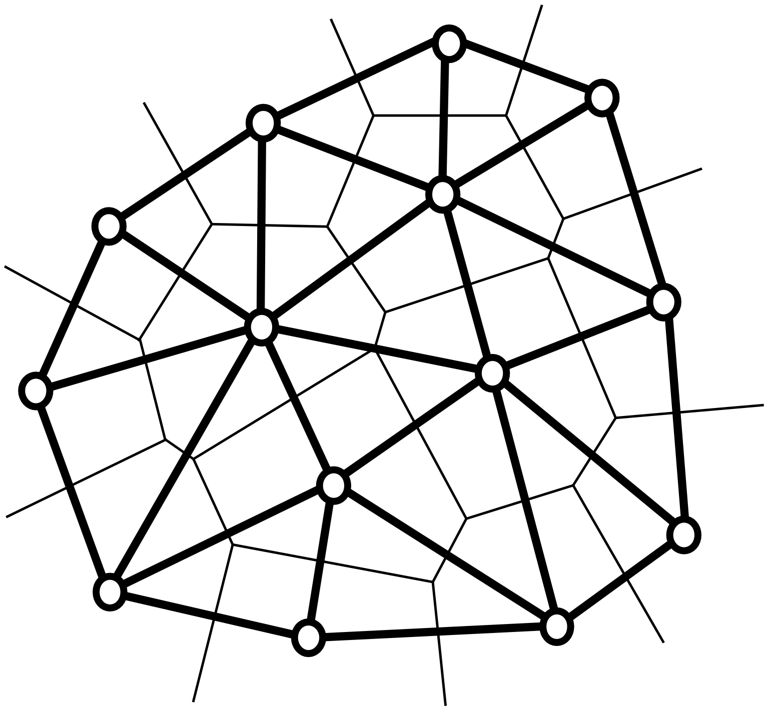
Delaunay triangulation



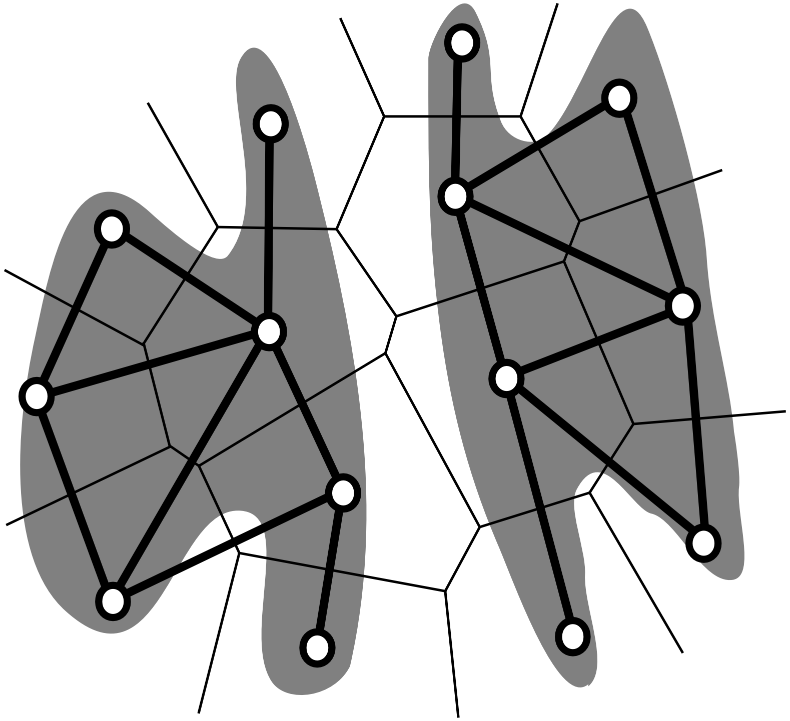
Masked Delaunay triangulation

Fundamental result

- The masked Delaunay triangulation
  - Leads to a topology preserving graph G
  - Preserves the topology relations in M



Delaunay triangulation



Masked Delaunay triangulation

Our objective

- For now, the points in S are given, where:

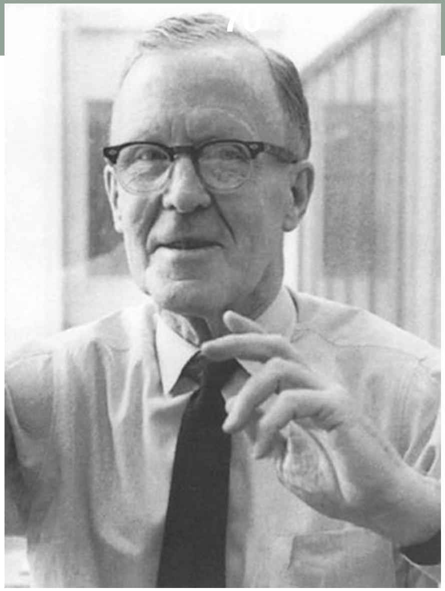
$$S = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N\}$$

- However, the edges connecting them to form G are unknown
- What we can do is sampling points **v** from M, *one at a time*

- From this procedure
  - We would like to automatically understand which edges are valid
  - So as to form a topology preserving map G

- From what we have seen this means that
  - The edges that we form in G must also be valid in M
  - That is: “adjacency” in G must correspond to “adjacency” in M and vice versa

Hebbian learning (1/3)



Donald O. Hebb - psychologist
(1908 – 1985)

- We assume a set of neural units $i = 1, 2, \ldots, \ell$
- Neural units are
  - Initially isolated
  - Develop lateral connections (edges) with neighbors
  - A unit i develops a lateral connection to a unit j through a synaptic link
- Lateral connections
  - Represent edges in the graph G
  - Are represented through a connection strength matrix **C** with elements $C_{ij}>0$
  - If $C_{ij}=0$ then units i and j are not connected
- The dynamics by which nearby units develop neuronal connections
  - Were first studied by Hebb [Hebb49] – *"neurons that fire together wire together"*
- [Hebb49] D.O. Hebb: The Organization of Behavior, Wiley, New York, 1949.

Hebbian learning (2/3)

'presynaptic' and 'postsynaptic' are used to indicate two neurons that are connected. Information flow in the nervous system basically goes one way. If one neuron fires (presynaptic cell) it can chemically activate another cell on which it “synapses” (the postsynaptic cell). A synapse is a neural pathway.

- Hebb’s postulate:
  - Unit i increases the strength of its synaptic link to unit j **IF** they are both active
    - “Neurons that fire together wire together”
    - This means that activities at units i and j are correlated
  - Provides a method to determine how to alter the weight between two neurons
  - Hence, the change in the synaptic strength is proportional to:

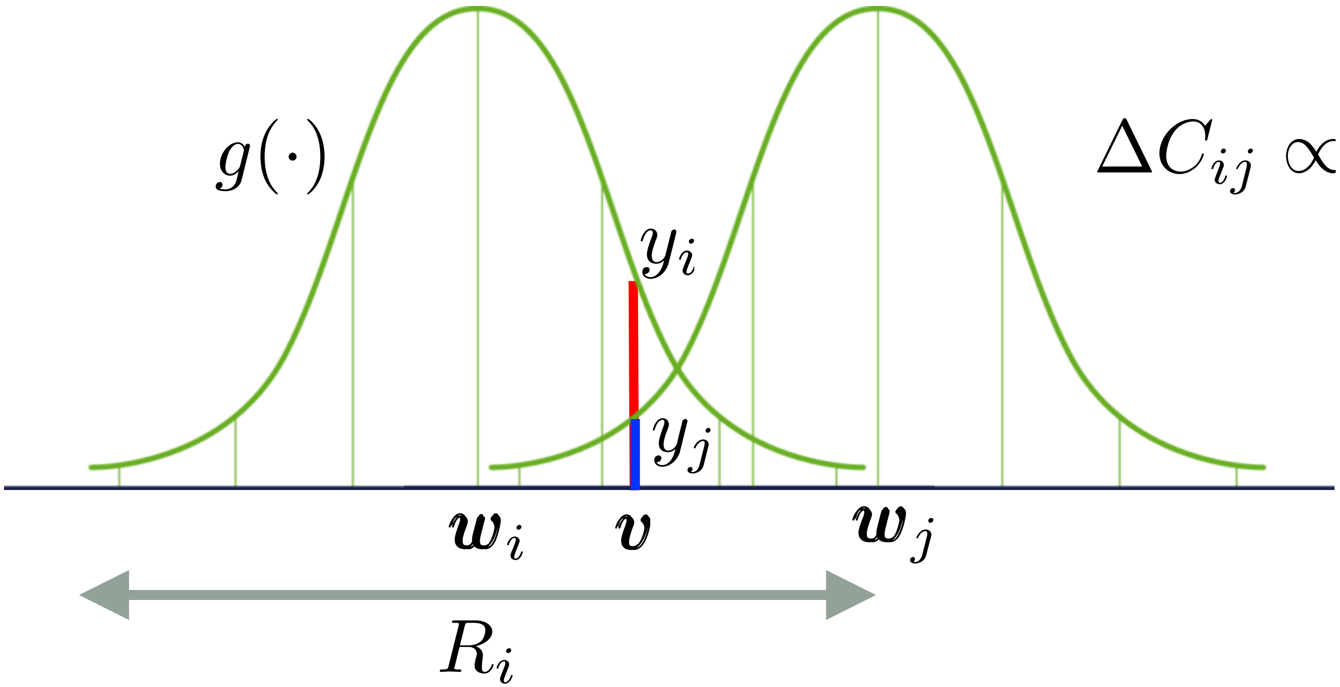
$$\Delta C_{ij} \propto y_i \times y_j$$

  - y_i and y_j indicate the intensity of the synaptic activity at units i and j

$$\boxed{C_{ij} \leftarrow C_{ij} + \eta y_i y_j}$$

Intermission: receptive field

- Each neuron i, has an associated receptive field R_i
  - The *center* of the receptive field is the neuron's weight
  - If any new input **v** falls within region R_i (centered on $\boldsymbol{w}_i$)
  - Neuron i fires with intensity: $y_i \propto g(\|\boldsymbol{w}_i - \boldsymbol{v}\|)$
  - Such intensity is proportional to the distance between **v** and $\mathbf{w}_i$



$$\Delta C_{ij} \propto y_i \times y_j$$

Hebbian learning (3/3)

- Verbatim application of Hebb’s rule
  - Would Lead to edges among all units (neurons)
  - With strength C_{ij} proportional to the overlap of receptive fields

$$C_{ij} \propto \int_{R_i \cap R_j} f(v) g(\|\boldsymbol{w}_i - \boldsymbol{v}\|) g(\|\boldsymbol{w}_j - \boldsymbol{v}\|) d\boldsymbol{v}$$

- This is not (yet) what we want
  - As it leads to the activation of all neurons at every new input
  - Does not involve competition
  - From SOM → we have learnt that competition leads to meaningful structure!!!

Definition “second order polyhedron”

- Definition – Second order polyhedron V_{ij}

- Second order Voronoi polyhedron V_{ij} with $i, j = 1, 2, \ldots, \ell$
- Is given by all the vectors $\boldsymbol{v} \in \mathbb{R}^m$
- For which $\boldsymbol{w}_i, \boldsymbol{w}_j$ are the two closest points to $\boldsymbol{v} \in \mathbb{R}^m$

$$V_{ij} = \{\boldsymbol{v} \in \mathbb{R}^m \mid \|\boldsymbol{v} - \boldsymbol{w}_i\| \leq \|\boldsymbol{v} - \boldsymbol{w}_k\| \cap \|\boldsymbol{v} - \boldsymbol{w}_j\| \leq \|\boldsymbol{v} - \boldsymbol{w}_k\| \, \forall \, k \neq i, j\}$$

Theorem 1

$$V_i \cap V_j \neq \emptyset \iff V_{ij} \neq \emptyset$$

Theorem 1

- Given a set of points $S = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N\}$
- It holds: $V_i \cap V_j \neq \emptyset \Longleftrightarrow V_{ij} \neq \emptyset$

Theorem 1

- Given a set of points $S = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N\}$
- It holds: $V_i \cap V_j \neq \emptyset \iff V_{ij} \neq \emptyset$

Sufficiency part ⇨

If $V_i \cap V_j \neq \emptyset$ is valid, then there exists $\boldsymbol{v} \in V_i, \boldsymbol{v} \in V_j$
For this vector it must hold (since it is in both sets)

$$\|\boldsymbol{v} - \boldsymbol{w}_i\| = \|\boldsymbol{v} - \boldsymbol{w}_j\| \leq \|\boldsymbol{v} - \boldsymbol{w}_k\| \text{ with } k \neq \{i, j\}$$

From the definition of second order polyhedron, this means that $\boldsymbol{v} \in V_{ij}$, which is therefore non empty

Theorem 1

- Given a set of points $S = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N\}$
- It holds: $V_i \cap V_j \neq \emptyset \iff V_{ij} \neq \emptyset$

Necessary part ⇐

We assume $V_{ij} \neq \emptyset$ and $V_i \cap V_j = \emptyset$

For each $\boldsymbol{v} \in V_{ij}$ the two nearest neighbors are $\boldsymbol{w}_i, \boldsymbol{w}_j$

Without loss of generality, we assume that for all $\boldsymbol{v} \in V_{ij}$ the nearest neighbor is $\boldsymbol{w}_i$

(**) Note that, if this were not true, there would be $\boldsymbol{v}^\star \in V_{ij}$ for which $\|\boldsymbol{v}^\star - \boldsymbol{w}_i\| = \|\boldsymbol{v}^\star - \boldsymbol{w}_j\|$ and then $V_i \cap V_j \neq \emptyset$

Theorem 1

Illustration of (**): if we had $\boldsymbol{v}_1, \boldsymbol{v}_2 \in V_{ij}$

with $\boldsymbol{v}_1$ closest to $\boldsymbol{w}_i$

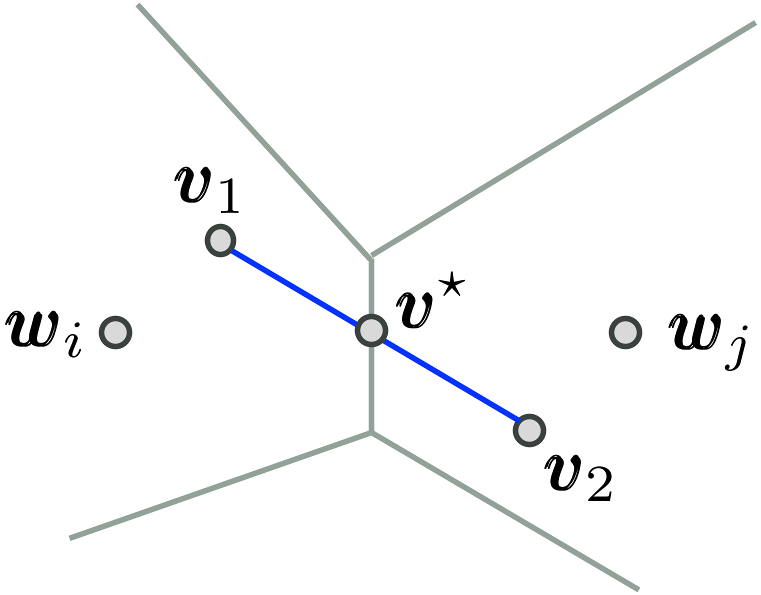
and $\boldsymbol{v}_2$ closest to $\boldsymbol{w}_j$

there must exist $\boldsymbol{v}^\star$ on the connecting line for which it holds

$$\|\boldsymbol{v}^\star - \boldsymbol{w}_i\| = \|\boldsymbol{v}^\star - \boldsymbol{w}_j\|$$

this implies $V_i \cap V_j \neq \emptyset$

hence, it is not permitted



Theorem 1

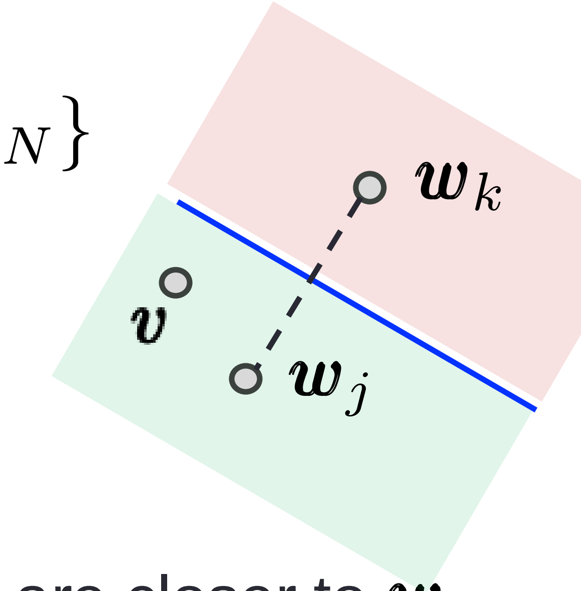
- Given a set of points $S = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N\}$
- It holds: $V_i \cap V_j \neq \emptyset \iff V_{ij} \neq \emptyset$

Necessary part ⇐

Since all points are closer to $\boldsymbol{w}_i$ it holds $V_{ij} \subseteq V_i$

However, it also (still) holds $\boldsymbol{v} \in V_{ij}$

Theorem 1



- Given a set of points $S = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N\}$
- It holds: $V_i \cap V_j \neq \emptyset \Longleftrightarrow V_{ij} \neq \emptyset$

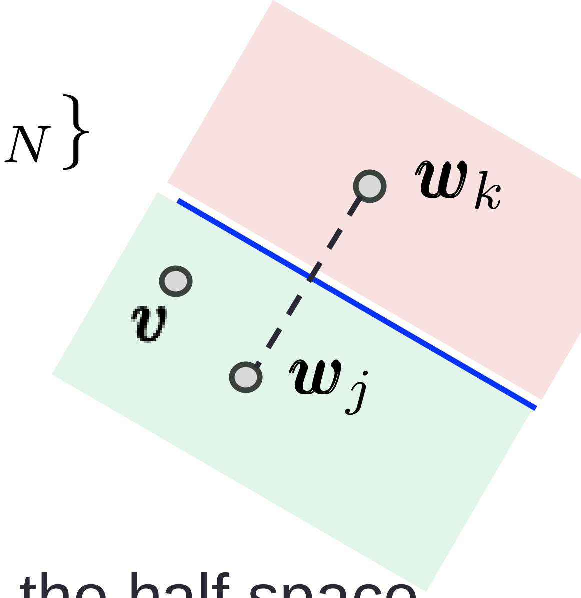
Necessary part

Hence, V_{ij} is given by all the $\boldsymbol{v} \in V_i$ which are closer to $\boldsymbol{w}_j$ with respect than all the other $\boldsymbol{w}_k \in S \setminus \{\boldsymbol{w}_i, \boldsymbol{w}_j\}$ (this directly descends from the def. of second order polyhedron)

Hence, V_{ij} is bounded by hyperplanes each of which is perpendicular to the connecting line between $\boldsymbol{w}_j$ and $\boldsymbol{w}_k$

Theorem 1

- Given a set of points $S = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N\}$
- It holds: $V_i \cap V_j \neq \emptyset \Longleftrightarrow V_{ij} \neq \emptyset$



Necessary part ⇦

For each of such hyperplanes $\boldsymbol{w}_j$ belongs to the half space which contains V_{ij}

Hence, it holds $\boldsymbol{w}_j \in V_{ij}$ and, from previous steps it also holds $\boldsymbol{w}_j \in V_{ij} \subseteq V_i$, which means $\boldsymbol{w}_j \in V_i$

However, since by definition it also holds $\boldsymbol{w}_j \in V_j$

We obtain $\boldsymbol{w}_j \in V_i \cap V_j$ which contradicts our assumption

QED

Competitive Hebbian learning (CHL)

- Intuition
  - Competition often leads to interesting structure (see, e.g., SOM)
- SOM
  - Competition is among the outputs of the units y_i (“winner-take-all” network)
- Here (Hebbian learning)
  - Competition is introduced among synaptic links
  - The correlated output activities are computed as: $Y_{ij} = y_i y_j$
  - The link with the highest Y_{ij} wins and *takes-all*, the remaining ones take nothing

- Competitive Hebbian learning among synaptic links

$$\Delta C_{ij} \propto \begin{cases} y_i y_j & y_i y_j \geq y_k y_l \,, \forall\, k, l \\ 0 & \text{otherwise} \end{cases}$$

Competitive Hebbian learning (CHL)

- Intuition
  - Competition often leads to interesting structure (see, e.g., SOM)
- SOM
  - Competition is among the outputs of the units y_i ("winner-take-all" network)
- Here (Hebbian learning)
  - Competition is introduced among synaptic links
  - The correlated output activities are computed as: $Y_{ij} = y_i y_j$
  - The link with the highest Y_{ij} wins and *takes-all*, the remaining ones take nothing

- Competitive Hebbian learning among synaptic links

$$\Delta C_{ij} \propto \begin{cases} y_i y_j & y_i y_j \geq y_k y_l \, , \forall \, k, l \\ 0 & \text{otherwise} \end{cases}$$

OBSERVATION:
A link (i,j) is reinforced (winner) IFF V_{ij} is non empty

Competitive Hebbian Learning (CHL)

- Result no. 1 – "CHL leads to Delaunay triangulation"
  - If we sequentially present to the network input patterns $\boldsymbol{v}$
  - Having a PDF $f(\boldsymbol{v})$ which has support that is non-zero everywhere in $\mathbb{R}^m$

  - Asymptotically, after a large number of input patterns have been collected
  - The weights converge to (due to Theorem 1):

$$\lim_{n \to +\infty} \theta(C_{ij}(n)) = A_{ij}$$

  - where:

$$\theta(\cdot) \text{ if the Heaviside step function}$$

- Adjacency matrix (V_i is the Voronoi region for neuron i)

$$A_{ij} = \begin{cases} 1 & V_i \cap V_j \neq \emptyset \\ 0 & V_i \cap V_j = \emptyset \end{cases}$$

This is the adjacency matrix of the Delaunay triangulation!!!

Def: S dense in M

- We recall – set of centroids is given

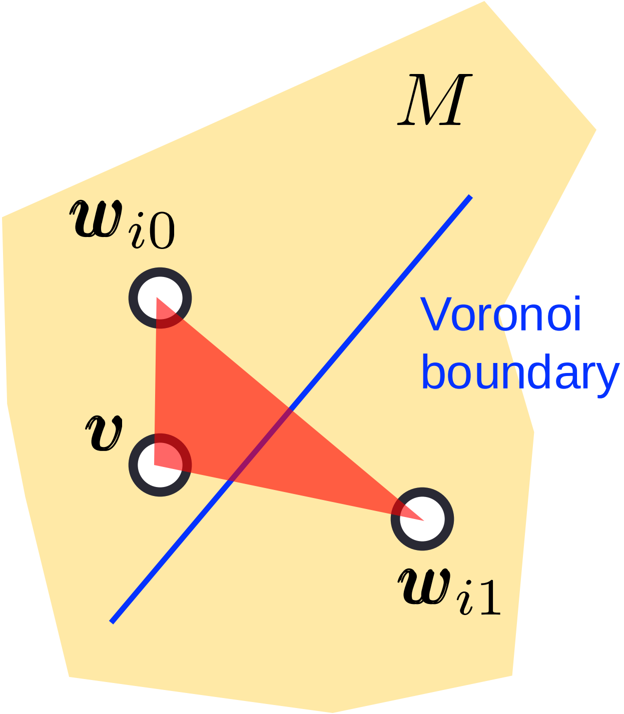
$$S = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N\}$$

- Definition – S is dense in M
  - Let **v** be an input pattern
  - $\mathbf{w}_{i0}$ is the closest weight (neuron)
  - $\mathbf{w}_{i1}$ is the second-closest weight (neuron)
  - The triangle formed by
    - **v**, $\mathbf{w}_{i0}$ and $\mathbf{w}_{i1}$ called

$$\Delta(\boldsymbol{v}, \boldsymbol{w}_{i0}, \boldsymbol{w}_{i1})$$

    - To be dense in M, Δ must be entirely contained in M

$$\Delta(\boldsymbol{v}, \boldsymbol{w}_{i0}, \boldsymbol{w}_{i1}) \subseteq M$$



$f(\boldsymbol{v})$ with support in M

- What happens when the input patterns PDF has support in M?

  (M is a subset of the input vector space)

- In this case
  - For some $V_{ij} \neq \emptyset$
  - We may have that:

$$V_{ij}^{(M)} \triangleq V_{ij} \cap M = \emptyset$$

(masked with M, there is no density in V_{ij})

- This means that

$$\int_{V_{ij}} f(\boldsymbol{v}) d\boldsymbol{v} \to 0$$

- That using CHL means that link (i,j) is not activated, i.e., $A_{ij} = 0$
  - This follows from Result 1, as none of the points in V_{ij} has probability strictly greater than 0 of being generated (the PDF is 0 in V_{ij})

Theorem 2

- "CHL leads to perfect topology preserving maps"
- This can be rephrased as:

$$V_i^{(M)} \cap V_j^{(M)} \neq \emptyset \Longleftrightarrow V_{ij}^{(M)} \neq \emptyset$$

- In words:
  - "masked Voronoi regions i and j have points in common (a link will be then created by CHL) iff the masked second order polyhedron is non empty"
- From previous slide: if the second order Voronoi polyhedron is $V_{ij}^{(M)} = \emptyset$
  - CHL will not create links between nodes i and j
  - But from Theorem 2 this implies that $V_i^{(M)} \cap V_j^{(M)} = \emptyset$
  - Which means that CHL is topology preserving (links will only be created between nodes that have Voronoi region with non-zero intersection in M, i.e., with common points in M)

Theorem 2

- We need to prove that: $V_i^{(M)} \cap V_j^{(M)} \neq \emptyset \Longleftrightarrow V_{ij}^{(M)} \neq \emptyset$

Sufficiency part

If $V_i^{(M)} \cap V_j^{(M)} \neq \emptyset \Rightarrow$ $\text{there exists } \boldsymbol{v} \text{ s.t.}$

$$\boldsymbol{v} \in V_i^{(M)}, \boldsymbol{v} \in V_j^{(M)}$$

and which also verifies,

$$\left\|\boldsymbol{v} - \boldsymbol{w}_i\right\| = \left\|\boldsymbol{v} - \boldsymbol{w}_j\right\| \leq \left\|\boldsymbol{v} - \boldsymbol{w}_k\right\|, \, \forall \boldsymbol{w}_k \in S$$

Which exactly means that: $v \in V_{ij}^{(M)}$

Theorem 2

- We need to prove that: $V_i^{(M)} \cap V_j^{(M)} \neq \emptyset \Longleftrightarrow V_{ij}^{(M)} \neq \emptyset$

Necessary part ⇦

If $V_{ij}^{(M)} \neq \emptyset$ then there exists a vector **v*** in the input space such that

$$\boldsymbol{v}^* \in V_{ij}^{(M)} \text{ s.t. } \Delta(\boldsymbol{v}^*, \boldsymbol{w}_i, \boldsymbol{w}_j) \subseteq M$$

(this holds as we assume that S is dense in M)

Note that, for each $\boldsymbol{v}^* \in V_{ij}^{(M)}$

$\boldsymbol{w}_i, \boldsymbol{w}_j$ are the two closest neighbors

Theorem 2: ⇦ continued

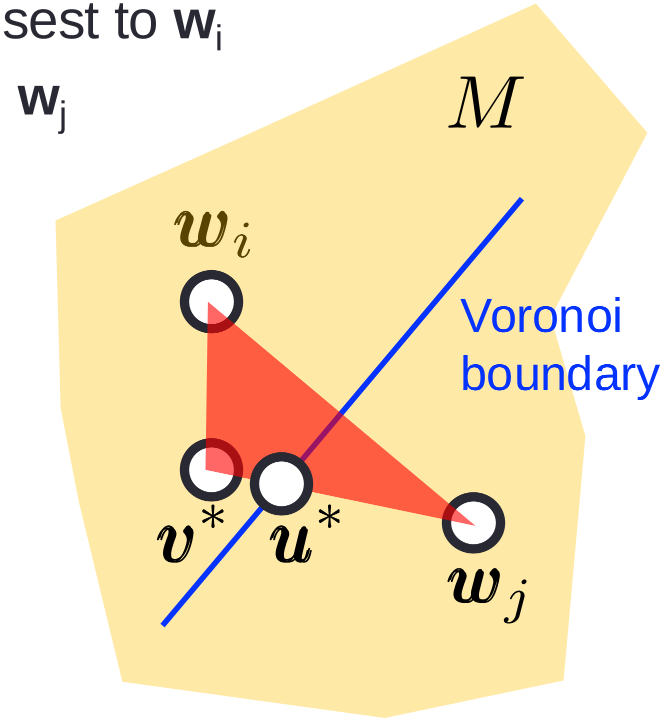
- Without loss of generality, assume that **v*** is closest to $\mathbf{w}_i$
- Let us take **u** in the segment connecting **v*** and $\mathbf{w}_j$
- Since
  - For **u**=**v*** → $\mathbf{w}_i$ is closest to **u**
  - For **u**=$\mathbf{w}_j$ → $\mathbf{w}_j$ is closest to **u**
- Then, there must exist an intermediate point **u***
  - In the segment seg(**v***, $\mathbf{w}_j$)
  - Such that:

$$\|\boldsymbol{u}^* - \boldsymbol{w}_i\| = \|\boldsymbol{u}^* - \boldsymbol{w}_j\|$$

  - Hence, we have:

$$\boldsymbol{u}^* \in V_i\ ,\ \boldsymbol{u}^* \in V_j\ ,\ \boldsymbol{u}^* \in \Delta(\boldsymbol{v}^*, \boldsymbol{w}_i, \boldsymbol{w}_j)$$

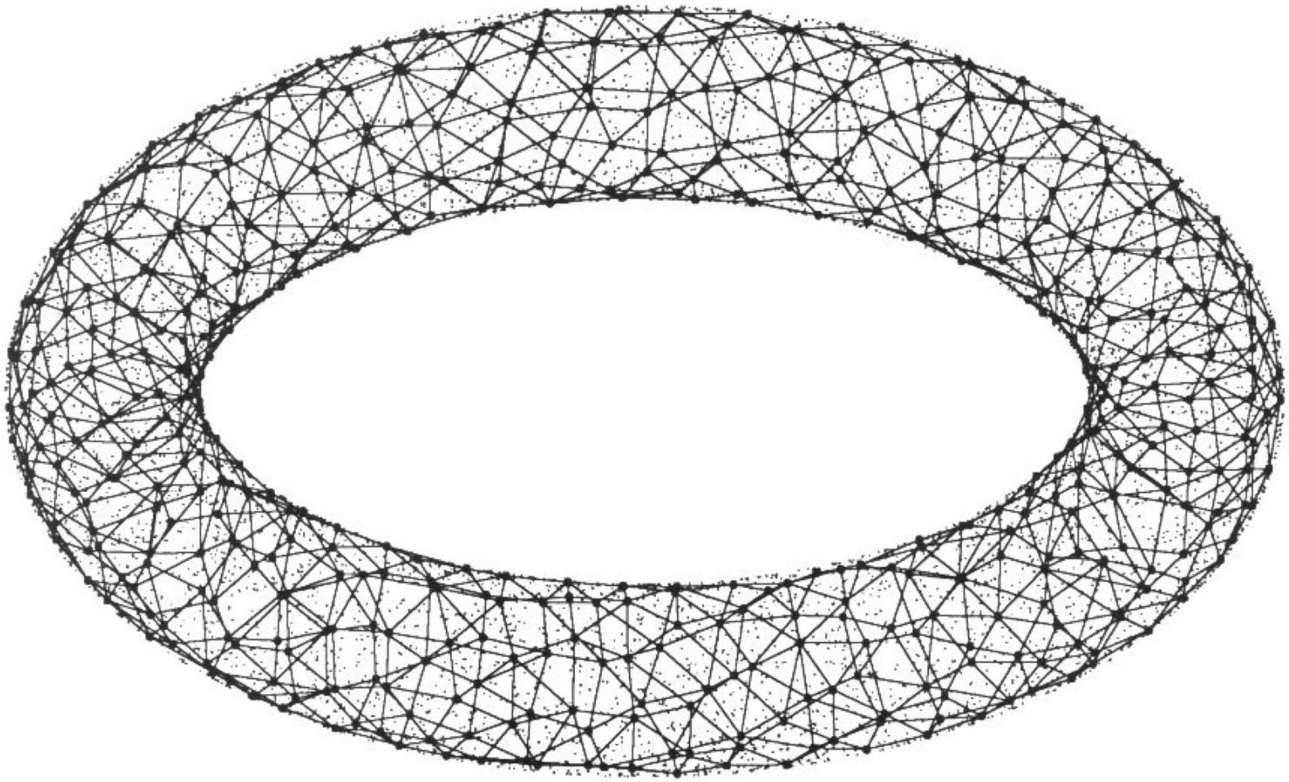
⇨ $$M \cap (V_i \cap V_j) \neq \emptyset \Rightarrow V_i^{(M)} \cap V_j^{(M)} \neq \emptyset$$



QED

Example

- Points in S pre-obtained through a vector quantization algorithm
- Links among them obtained using CHL
- Structure of the 3D manifold (torus) is preserved



Reference paper

Theorem 1 and Theorem 2 are in [Martinetz-93]

[Martinetz-93] Thomas Martinetz, “Competitive Hebbian Learning Rule Forms Perfectly Topology Preserving Maps,” *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 1993. (461 citations, Oct. 2024)

Growing Neural Gas (GNG) networks

- Unsupervised dimensionality reduction technique
  - Performs unsupervised clustering (or vector quantization)
- Number of neurons (*centroids*)
  - Not fixed a priori but dynamically tuned based on input statistics
- Uses Competitive Hebbian Learning (CHL)
  - To connect neighbors into a topology preserving manner
- Edges (connections among neurons)
  - Are not weighted
  - Their sole purpose is the definition of topological structure
  - An age count is maintained for each (addition wrt theory)
- Neurons
  - Have a local weight (feature vectors)
  - Addition and removal are possible as learning evolves

GNG – basic construct

- Basic construct
  - A network of nodes (neurons)
  - Each node has a weight $\boldsymbol{w}_i \in \mathbb{R}^m$
  - Connections are not weighted
  - Their sole purpose is the definition of topological structure

  - There is a (possibly infinite) number of m-dimensional input patterns $\boldsymbol{v} \in \mathbb{R}^m$
  - Input patterns obey some unknown probability density function $f(\boldsymbol{v})$
- Main idea
  - *Successively add new units to <u>an initially small network</u> by evaluating local statistical measures gathered during previous adaptation steps. Likewise, adapt the weight of existing units, possibly adding new or removing old ones.*

GNG – steps of the algorithm (1/3)

- 0) Start with two nodes with random weights $\boldsymbol{w}_a, \boldsymbol{w}_b$
- 1) Generate an input pattern $\boldsymbol{v}$ according to $f(\boldsymbol{v})$
- 2) Find the nearest and second-nearest nodes n_1, n_2
  - Closeness is measured as Euclidean distance between their weight and $\boldsymbol{v}$
- 3) Increment the age of all edges emanating from n_1
- 4) Add squared distance between the input pattern and the nearest node to a local variable

$$\text{error}(n_1) = \text{error}(n_1) + \|\boldsymbol{w}_{n_1} - \boldsymbol{v}\|^2$$

- 5) Move n_1 and its topological neighbors (all nodes that are linked with it) towards $\boldsymbol{v}$ by fractions ϵ_b, ϵ_n of their “distance” from **v**

$$\Delta\boldsymbol{w}_{n_1} = \epsilon_b(\boldsymbol{v} - \boldsymbol{w}_{n_1})$$

$$\Delta\boldsymbol{w}_{n_j} = \epsilon_n(\boldsymbol{v} - \boldsymbol{w}_{n_j}), \text{ for all neighbors } n_j \text{ of } n_1$$

GNG – steps of the algorithm (2/3)

- 6) if n_1, n_2 are already connected by an edge, set its age to zero, if this edge does not exist, create it (with zero age)
- 7) Remove edges with age larger than a_{max}. If this results in nodes having no emanating edges, remove these nodes as well
- 8) Structure refinement: if number of input patterns that were generated is an integer multiple of a parameter λ, insert a new node r as follows
  - Determine the node q with maximum accumulated error
  - Insert the new node r halfway between q and its neighbor f with the largest accumulated error variable

$$\boldsymbol{w}_r = 0.5(\boldsymbol{w}_q + \boldsymbol{w}_f)$$

  - Insert edges connecting the new node r with nodes q and f, and remove the original edge between q and f
  - Decrease the accumulated errors of q and f, by multiplying them by a constant α in (0,1). Initialize the error variable of node r with the new value of the error variable of node q

GNG – steps of the algorithm (3/3)

- 9) decrease the error variables at all nodes n, by multiplying them by a constant β (in (0,1), i.e., a sort of discount factor)

$$\mathrm{error}(n) \leftarrow \beta \mathrm{error}(n)$$

  - It effectively implements a low-pass filtering (averaging) of the local error
  - β can be seen as a forgetting factor for the past error
  - This creates stability and prevents the local error estimates to diverge

- 10) if some stopping criterion is not yet met (i.e., network size or some error measure, e.g., maximum error smaller than a threshold): GO TO 1

GNG – observations (1/2)

- The GNG algorithm leads to a general movement of all nodes (their weights) towards those areas of the input space where the input signal comes from (see step 5), i.e., where $f(\boldsymbol{v}) > 0$
- **IMPORTANT**
  - The insertion of edges (see step 6) between nearest and second-nearest node generates a connection of the “induced Delaunay triangulation” with respect to the current position of all nodes
- The removal of edges (step 7) is needed to obtain adaptation of the topology, when a region is no longer excited, get rid of it!!!
  - This is achieved through local edge aging around the nearest node (step 3), combined with age resetting of those edges that already exist between nearest and second-nearest nodes (step 6). This is competitive Hebbian learning among edges, i.e., only the winning edge is reinforced, in this case, by *resetting its age*

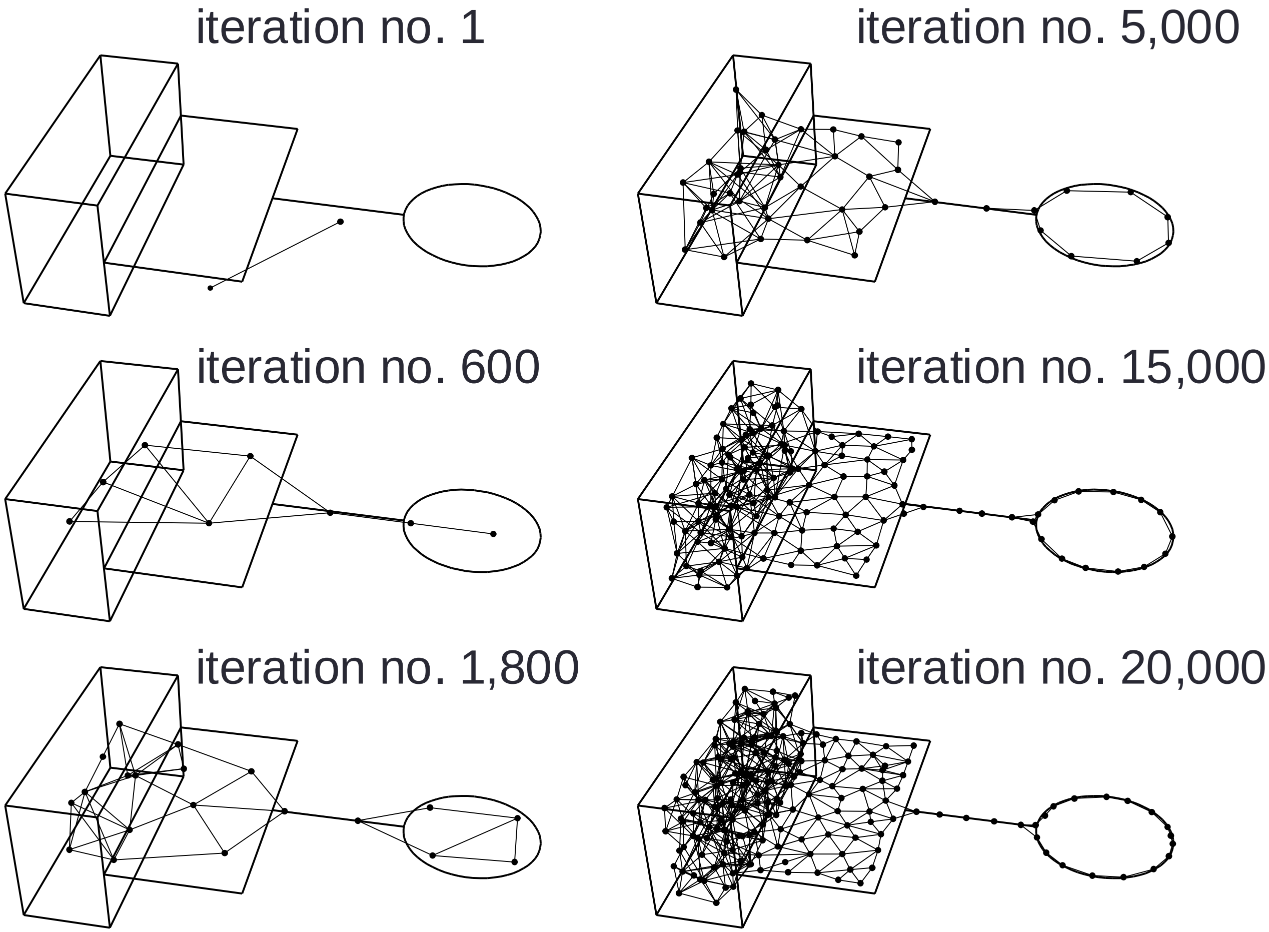
GNG – observations (2/2)

- The accumulation of squared error distances (step 4) during the adaptation helps identify those regions where the mapping causes a high error. To reduce this error, new nodes are inserted in such regions
  - New nodes are added at a constant pace (every λ steps)
- The maximum age a_{max} determines when an edge has to be removed
  - Age of an edge tells how many steps have been elapsed since the last activation of one of its end nodes
  - In other words, this represents the last time that the corresponding region in space was hit by an input pattern
  - Hence, increasing a_{max} corresponds to tracking smaller values of $f(\boldsymbol{v}) > 0$ (whose patterns occur less often)

Example results (1/2)

- Evolution of the GNG
  - Parameters are:

$$\begin{cases} \lambda = 100 \\ \alpha = 0.5 \\ \beta = 0.995 \\ \epsilon_b = 0.2 \\ \epsilon_n = 0.006 \\ a_{\max} = 50 \end{cases}$$



iteration no. 1

iteration no. 5,000

iteration no. 600

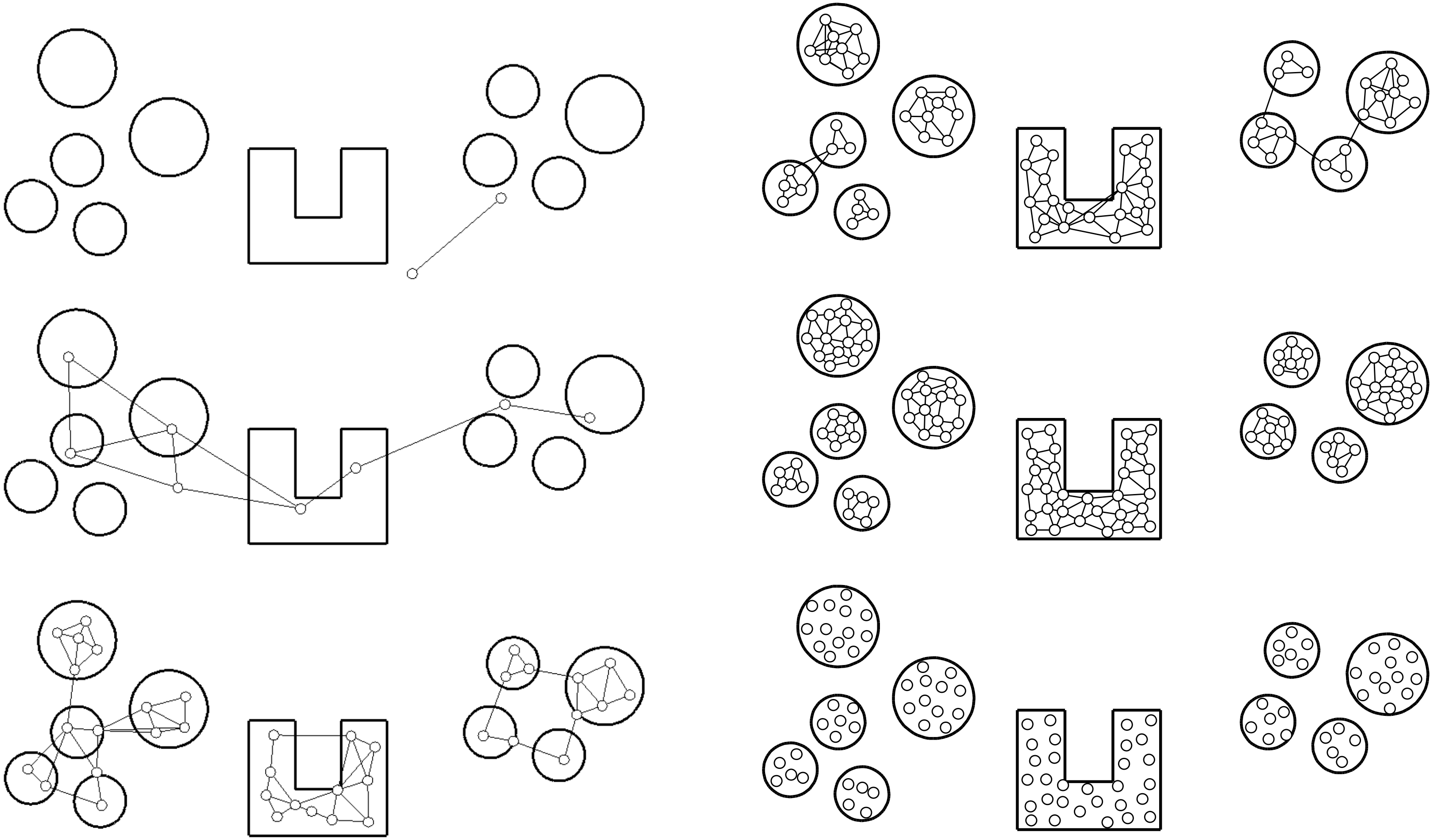
iteration no. 15,000

iteration no. 1,800

iteration no. 20,000

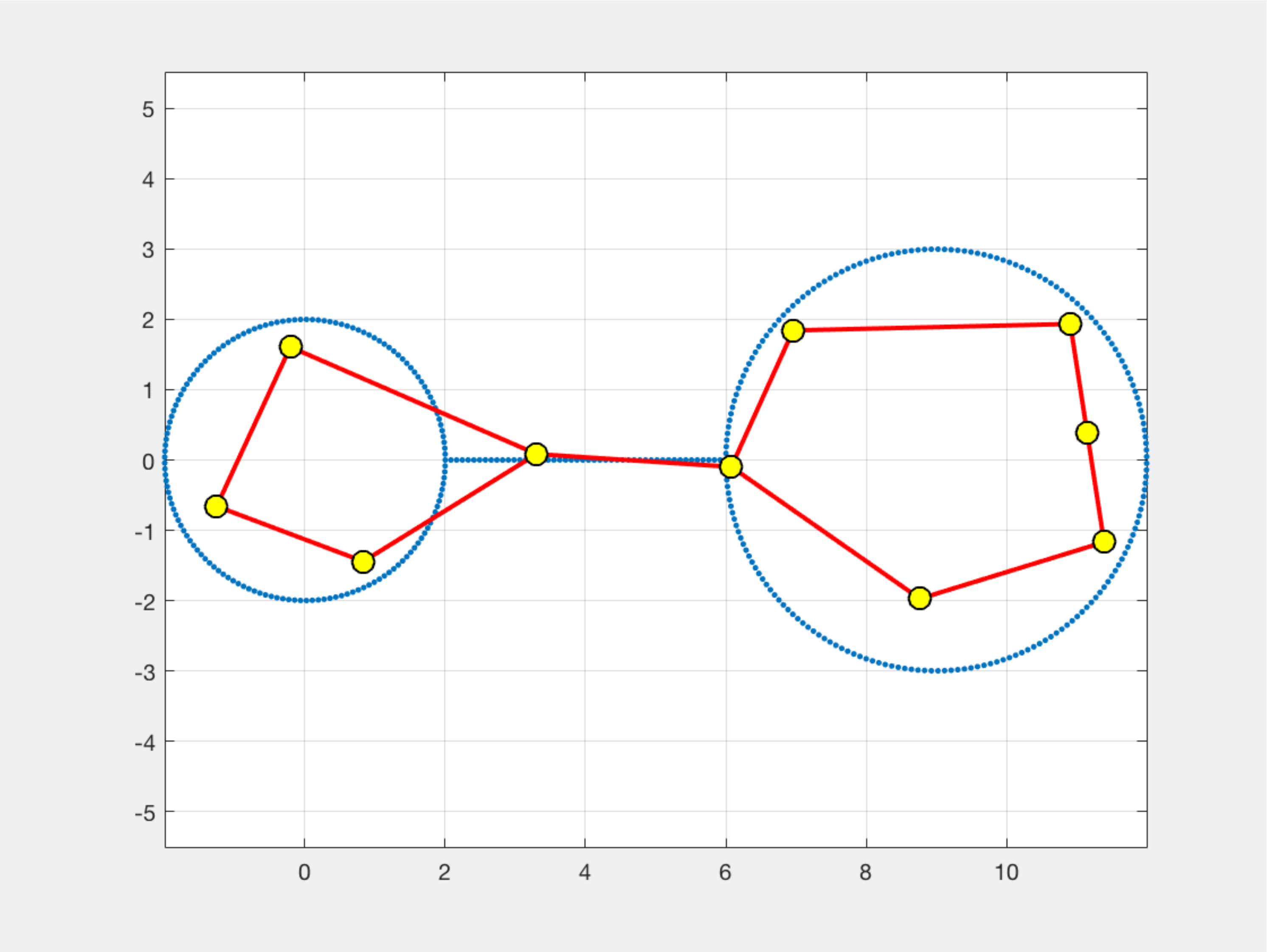
Example results (2/2) (10,000 steps, 100 nodes at the end)

“growing neural gas”
(uses “competitive Hebbian learning”)

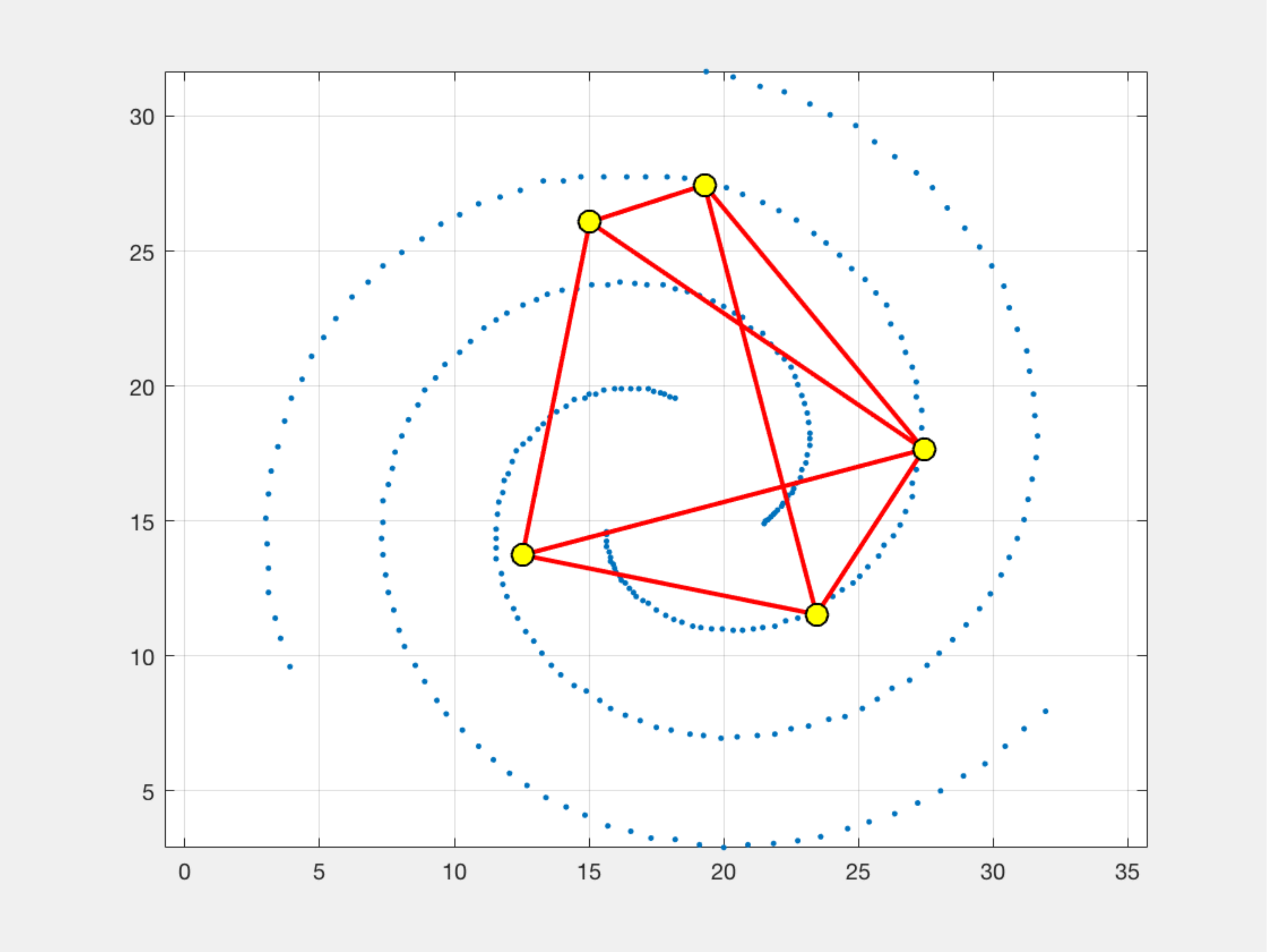


GROWING NEURAL GAS: ONLINE TRAINING EXAMPLES

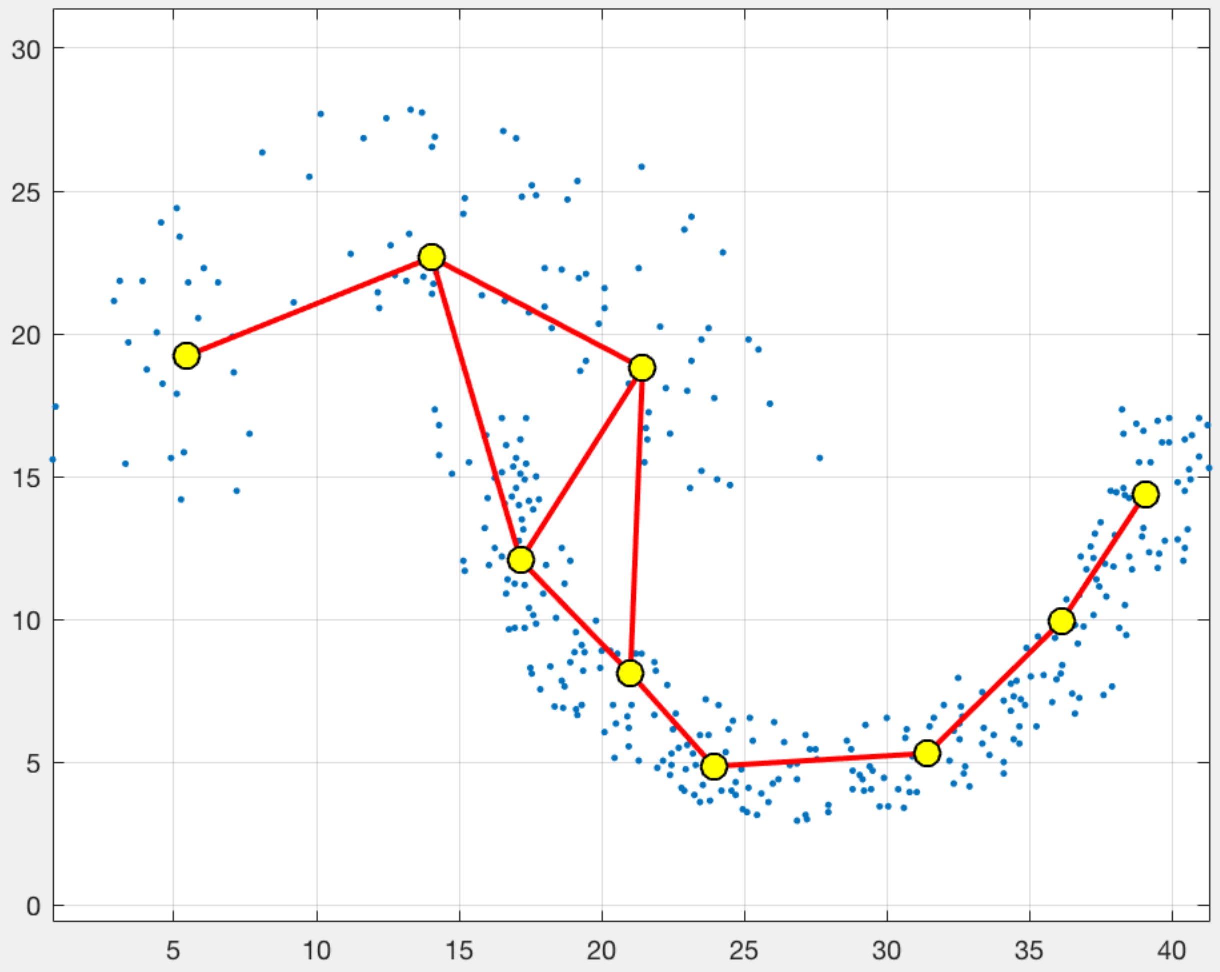
Vector quantization of PDF (circles/line)

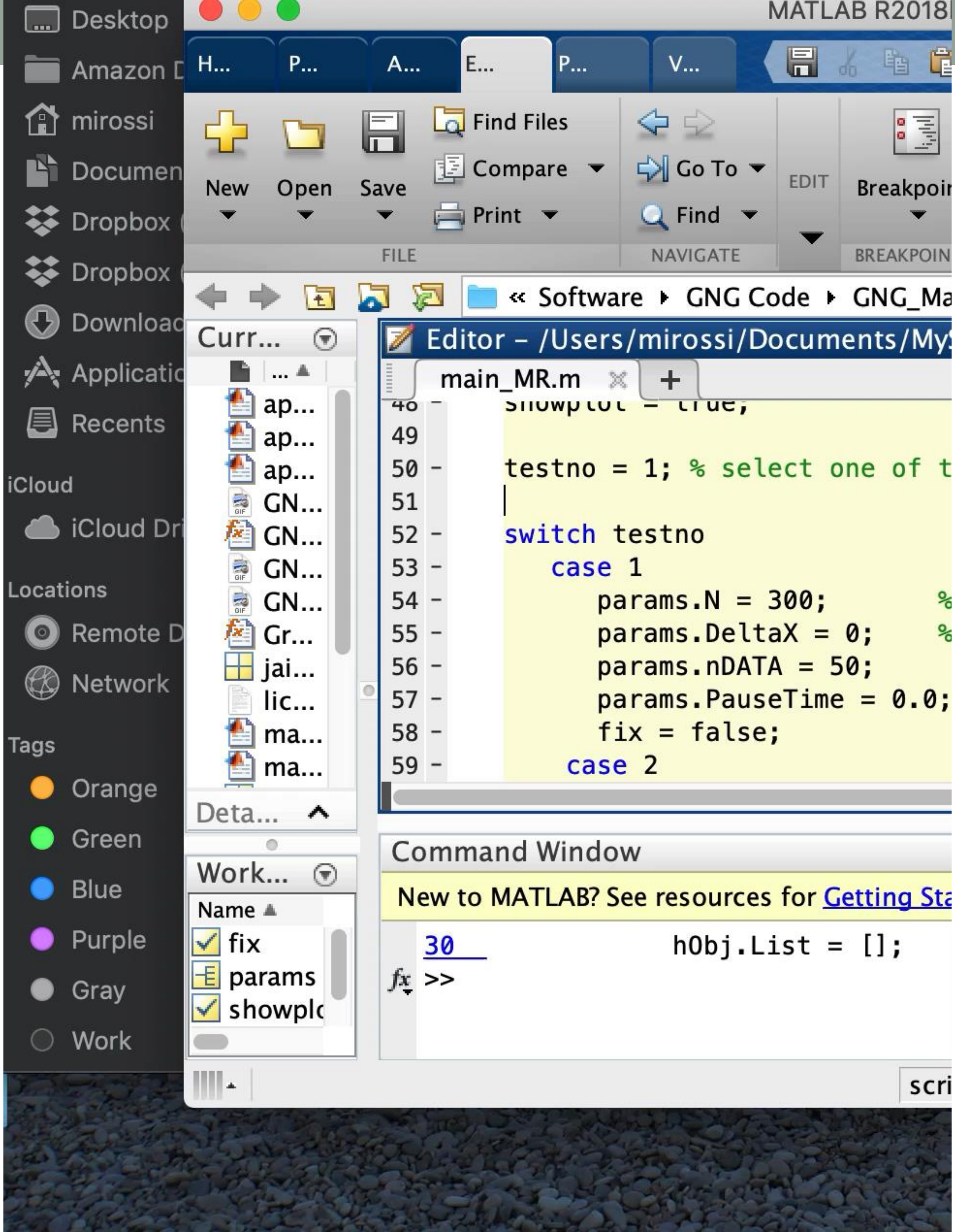


Vector quantization of PDF (spiral)



Vector quantization of PDF (data cloud)





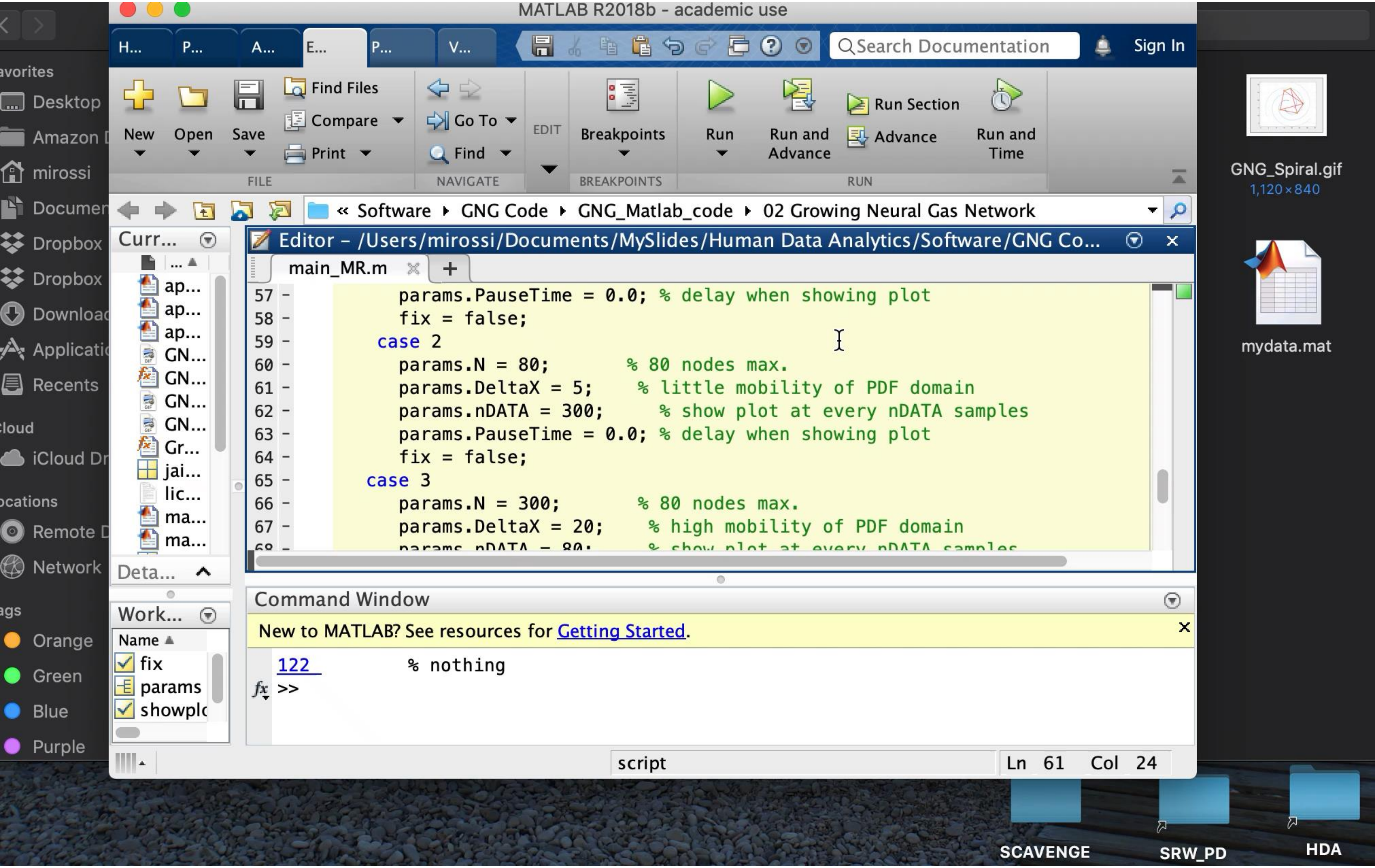
```
    showplot = true;

    testno = 1; % select one of t

    switch testno
       case 1
          params.N = 300;          %
          params.DeltaX = 0;       %
          params.nDATA = 50;
          params.PauseTime = 0.0;
          fix = false;
     case 2
```

New to MATLAB? See resources for Getting Sta

```
30            hObj.List = [];
>>
```



Editor – /Users/mirossi/Documents/MySlides/Human Data Analytics/Software/GNG Co...

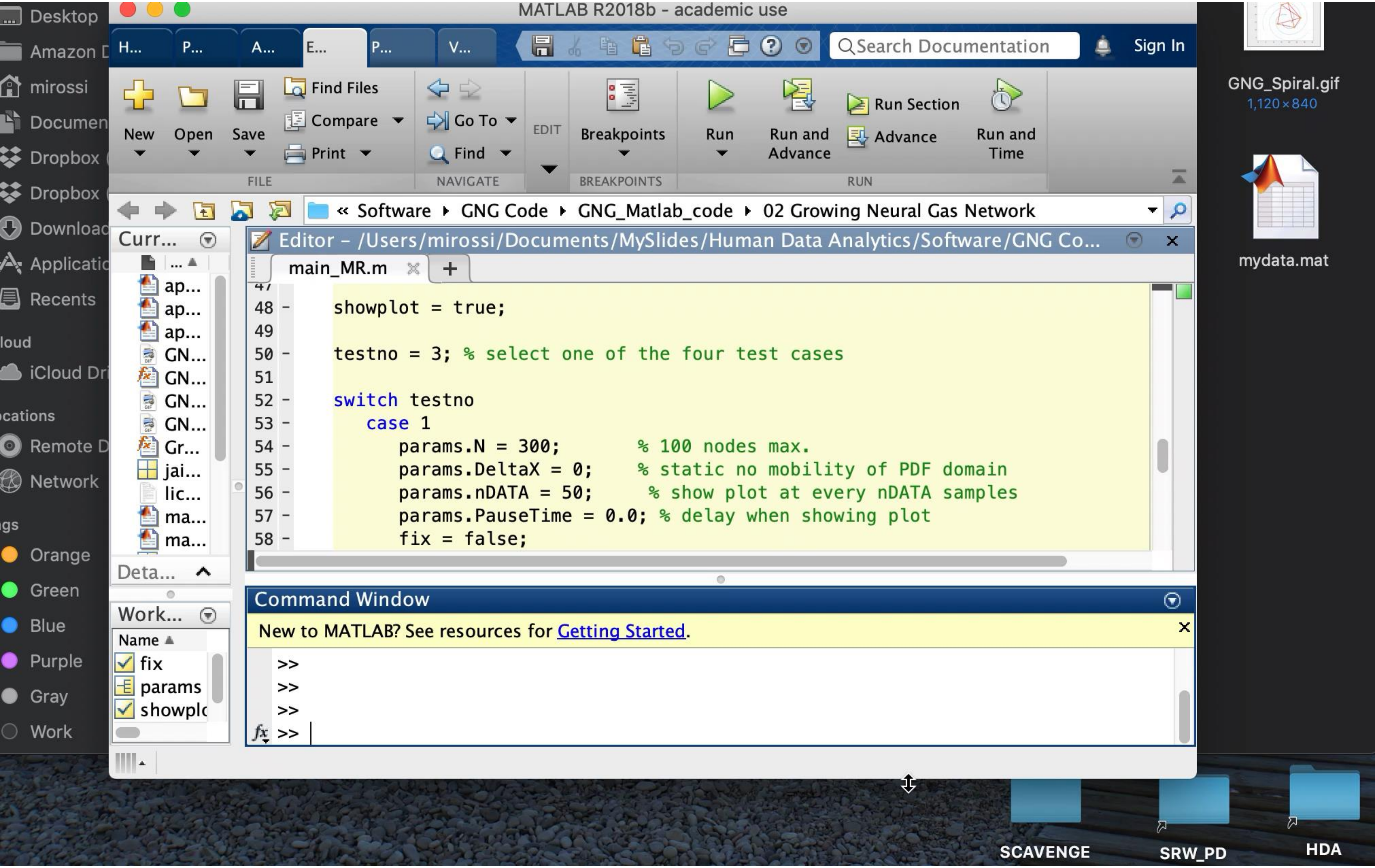
main_MR.m

```
      params.PauseTime = 0.0; % delay when showing plot
      fix = false;
    case 2
      params.N = 80;       % 80 nodes max.
      params.DeltaX = 5;    % little mobility of PDF domain
      params.nDATA = 300;     % show plot at every nDATA samples
      params.PauseTime = 0.0; % delay when showing plot
      fix = false;
   case 3
      params.N = 300;       % 80 nodes max.
      params.DeltaX = 20;    % high mobility of PDF domain
      params.nDATA = 80;     % show plot at every nDATA samples
```

New to MATLAB? See resources for Getting Started.

```
122          % nothing
>>
```

script | Ln 61 | Col 24



```
    showplot = true;

    testno = 3; % select one of the four test cases

    switch testno
       case 1
          params.N = 300;        % 100 nodes max.
          params.DeltaX = 0;     % static no mobility of PDF domain
          params.nDATA = 50;      % show plot at every nDATA samples
          params.PauseTime = 0.0; % delay when showing plot
          fix = false;
```

Command Window

New to MATLAB? See resources for Getting Started.

```
>>
>>
>>
>>
```

Gas neural networks bibliography

Lessons based on:

[Martinez1993] Thomas Martinetz, “Competitive Hebbian Learning Rule Forms Perfectly Topology Preserving Maps,” *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 1993. (461 citations, October 2024)

[Fritzke1994] Bernd Fritzke, “A Growing Neural Gas Network Learns Topologies,” *Proceedings of the 7th International Conference on Neural Information Processing Systems (NIPS)*, 1994. (2747 citations, Oct. 2024)

Further readings:

[Marslanda2002] Stephen Marslanda, Jonathan Shapiro, Ulrich Nehmzow, "A self-organising network that grows when required,” *Neural Networks Journal*, Vol. 15, No. 8-9, October 2002. (528 citations, Oct. 2024)

MiME
Master's degree ICT Internet Multimedia Engineering

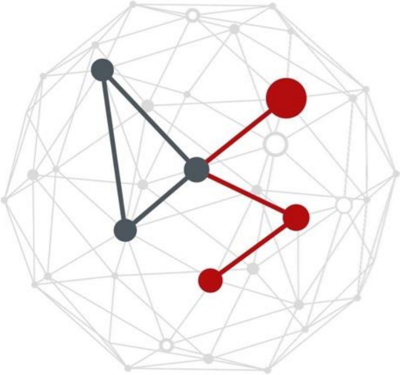
COMPETITIVE & UNSUPERVISED LEARNING FOR VECTOR QUANTIZATION

Michele Rossi

michele.rossi@unipd.it

Dept. of Information Engineering

University of Padova, IT



DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO
MATEMATICA

UNIVERSITÀ
DEGLI STUDI
DI PADOVA