How Many Dissimilarity/Kernel Self Organizing Map Variants Do We Need?

Fabrice Rossi

SAMM, Université Paris 1

WSOM 2014
Mittweida

How Many Dissimilarity/Kernel Self Organizing Map Variants Do We Need?

Fabrice Rossi

SAMM, Université Paris 1

WSOM 2014
Mittweida
"a little bit small compared to Paris"

Data complexity is increasing

Modern data are complex

- text everywhere (comments, messages, status, etc.)
- images everywhere
- relations (friends/contact, like/plus, ad hoc discussion, etc.)
- mixed data (buyers/items, listeners/songs, etc.)

Data complexity is increasing

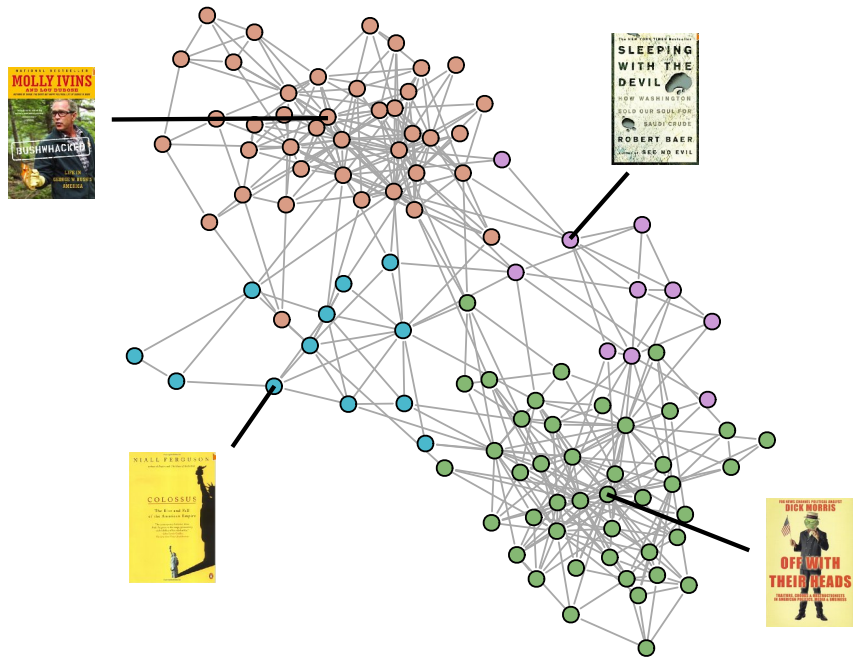
Modern data are complex

- text everywhere (comments, messages, status, etc.)
- images everywhere
- relations (friends/contact, like/plus, ad hoc discussion, etc.)
- mixed data (buyers/items, listeners/songs, etc.)

Data complexity is increasing

Modern data are complex

- text everywhere (comments, messages, status, etc.)
- images everywhere
- relations (friends/contact, like/plus, ad hoc discussion, etc.)
- mixed data (buyers/items, listeners/songs, etc.)



Data complexity is increasing

Modern data are complex

- text everywhere (comments, messages, status, etc.)
- images everywhere
- relations (friends/contact, like/plus, ad hoc discussion, etc.)
- mixed data (buyers/items, listeners/songs, etc.)

The vector model...

- in which all objects $(x_i)_{1 \leq i \leq N}$ live in a fixed vector space $\mathbb{R}^p$
- ...is less and less relevant

Solutions

1. specific solutions (e.g., probabilistic models for relational data)
2. generic solutions via a comparison measure

Dissimilarity/Kernel Data

Data model

- a data space $\mathcal{X}$ (might be implicit)
- N observations $(x_i)_{1 \leq i \leq N}$ from $\mathcal{X}$ (possibly with no attached description)

Dissimilarity

- a symmetric dissimilarity d function from $\mathcal{X}^2$ to $\mathbb{R}^+$
- or a symmetric matrix $D = (d(x_i, x_j))_{1 \leq i \leq N, 1 \leq j \leq N}$

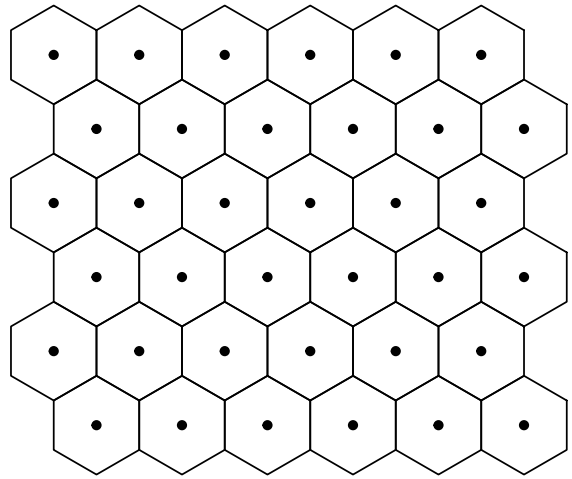
Kernel

- a kernel function k from $\mathcal{X}^2$ to $\mathbb{R}$, symmetric and positive definite
- or a symmetric positive definite matrix $K = (k(x_i, x_j))_{1 \leq i \leq N, 1 \leq j \leq N}$

SOM

Low dimensional prior structure

- a regular lattice of K units/neurons in $\mathbb{R}^2$: $(r_k)_{1 \leq k \leq K}$
- a time dependent neighborhood function $h_{kl}(t)$, e.g. $h_{kl}(t) = exp\left(-\frac{\|r_k - r_l\|^2}{2\sigma^2(t)}\right)$



Mapping

- each neuron r_k is associated to a prototype/model m_k in the data space
- each m_k/r_k is responsible of a cluster of data points, the C_k: quantization/clustering aspect
- if r_k and r_l are close according to h_{kl} then m_k and m_l should be close: topology preservation aspect

Training Algorithms

Stochastic/Online SOM

1. select a random data point x
2. find its *best matching unit*

$$c = \arg \min_{k \in \{1,\ldots,K\}} \|x - m_k(t)\|^2$$

3. update all prototypes

$$m_k(t+1) = m_k(t) + \epsilon(t) h_{kc}(t)(x - m_k(t))$$

4. loop to 1 until convergence

Training Algorithms

Batch SOM

1. compute the *best matching unit* for all data points

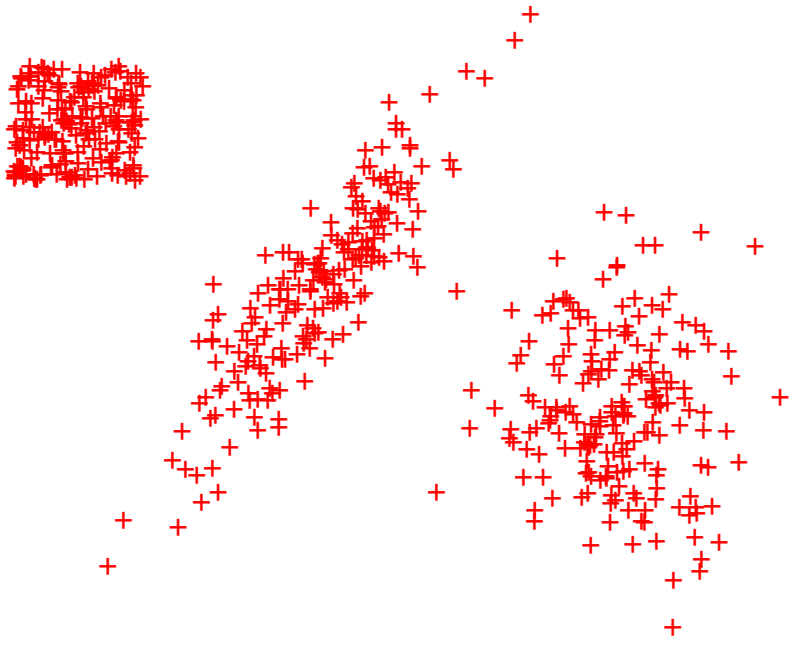
$$c_i(t) = \arg \min_{k \in \{1,\ldots,K\}} \|x_i - m_k(t)\|^2$$

2. update all prototypes

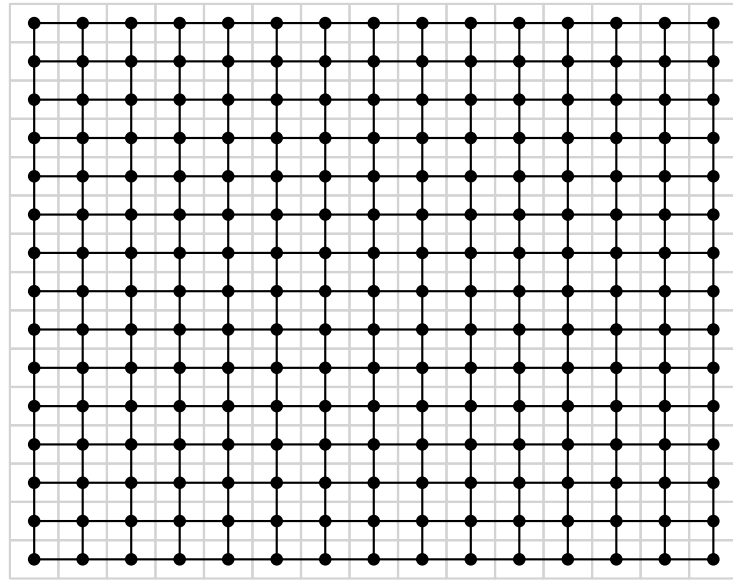
$$m_k(t+1) = \frac{\sum_{i=1}^{N} h_{kc_i(t)}(t) x_i}{\sum_{i=1}^{N} h_{kc_i(t)}(t)}$$

3. loop to 1 until convergence

Demo

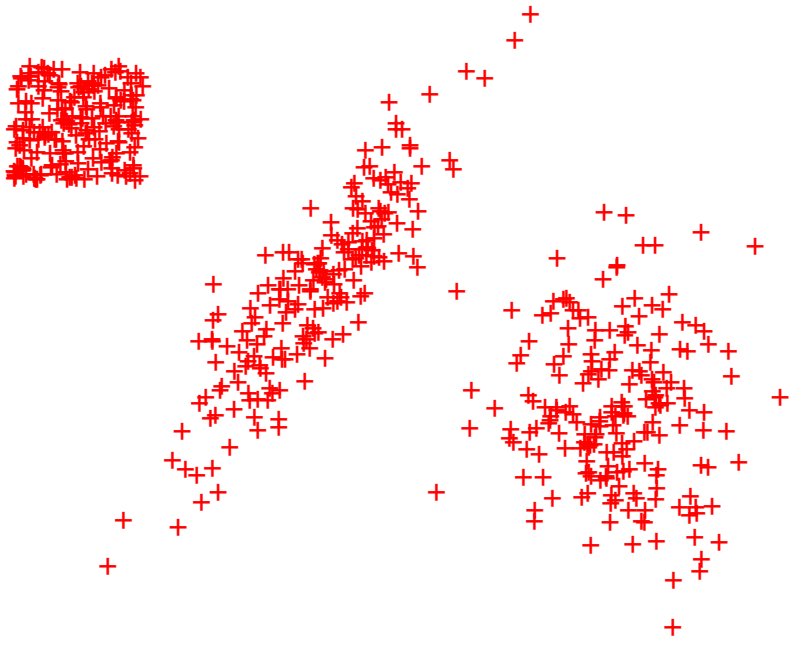


A simple 2D dataset

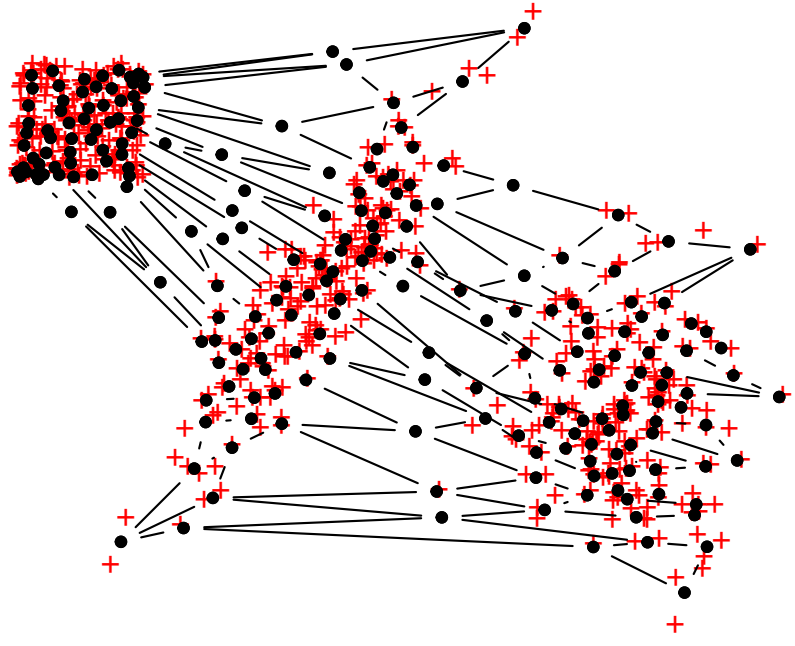


The original grid

Demo



A simple 2D dataset

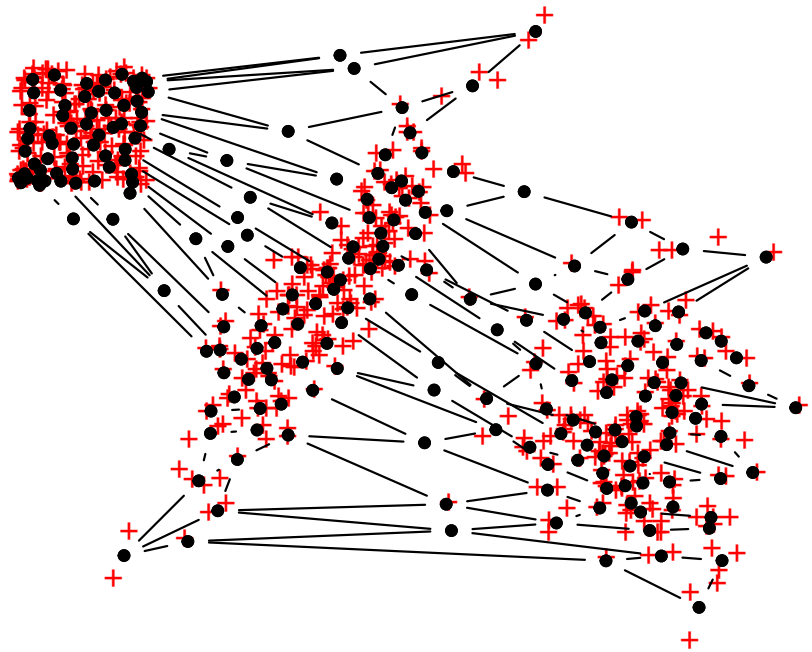


Prototype positions in the data space

Why does the SOM shine?

The SOM is a visualization framework

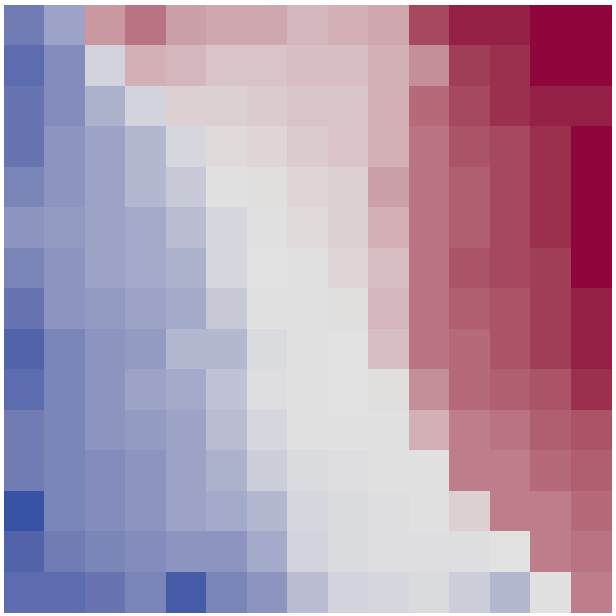
- glyph based visualization
- component planes
- hit map (data histograms)
- U matrix
- you name it...

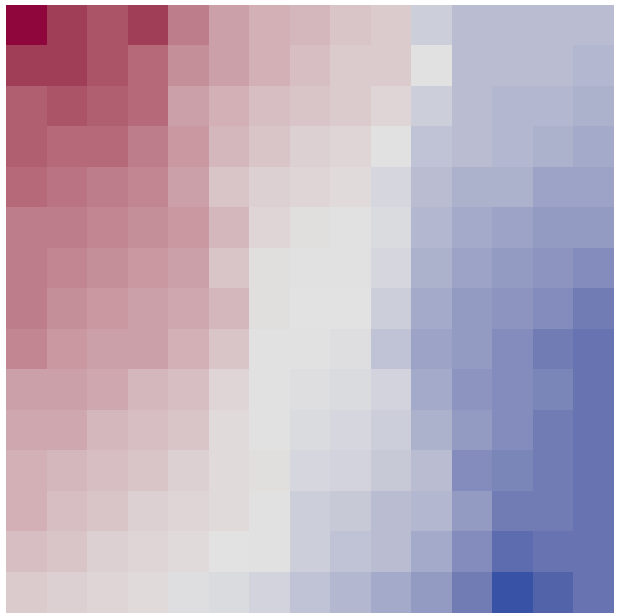


Why does the SOM shine?

The SOM is a visualization framework

- glyph based visualization
- component planes
- hit map (data histograms)
- U matrix
- you name it...

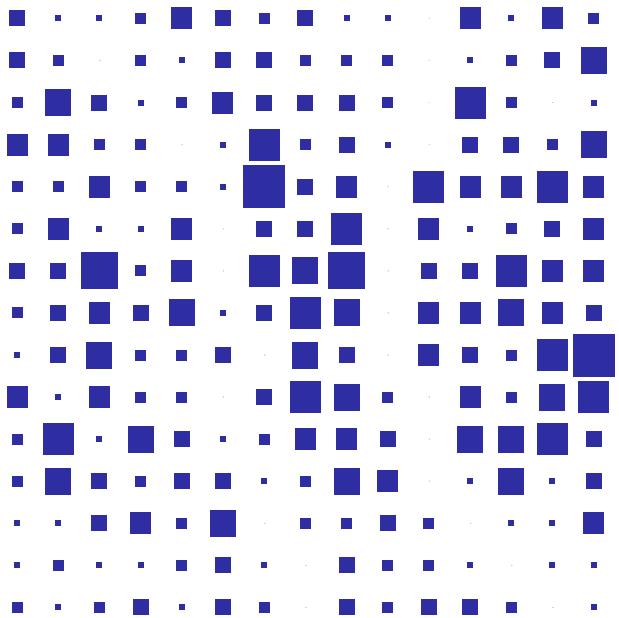




Why does the SOM shine?

The SOM is a visualization framework

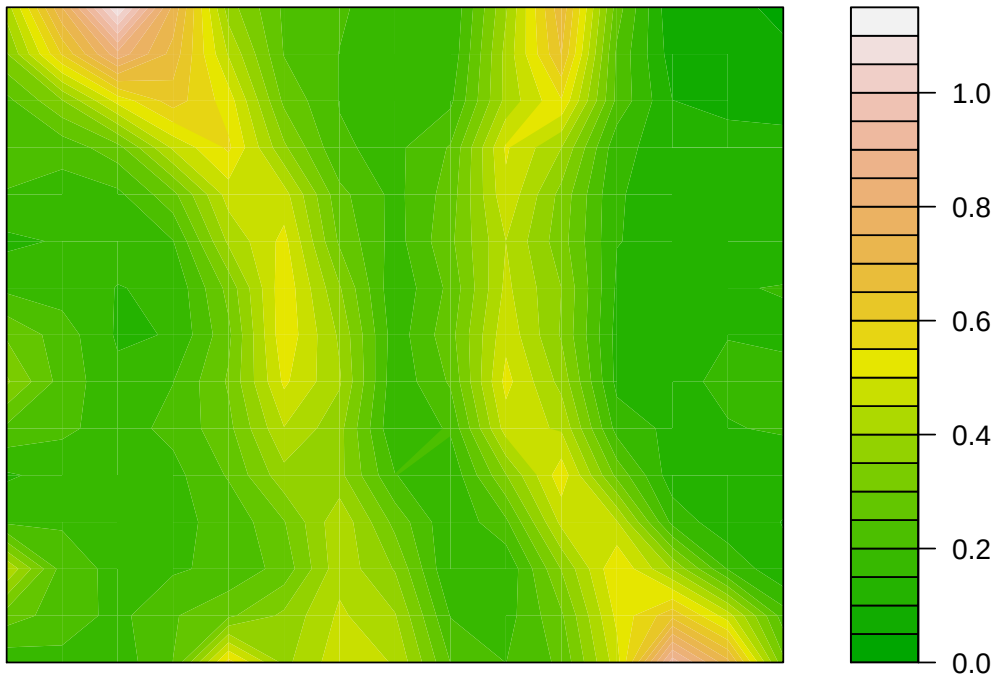
- glyph based visualization
- component planes
- hit map (data histograms)
- U matrix
- you name it...



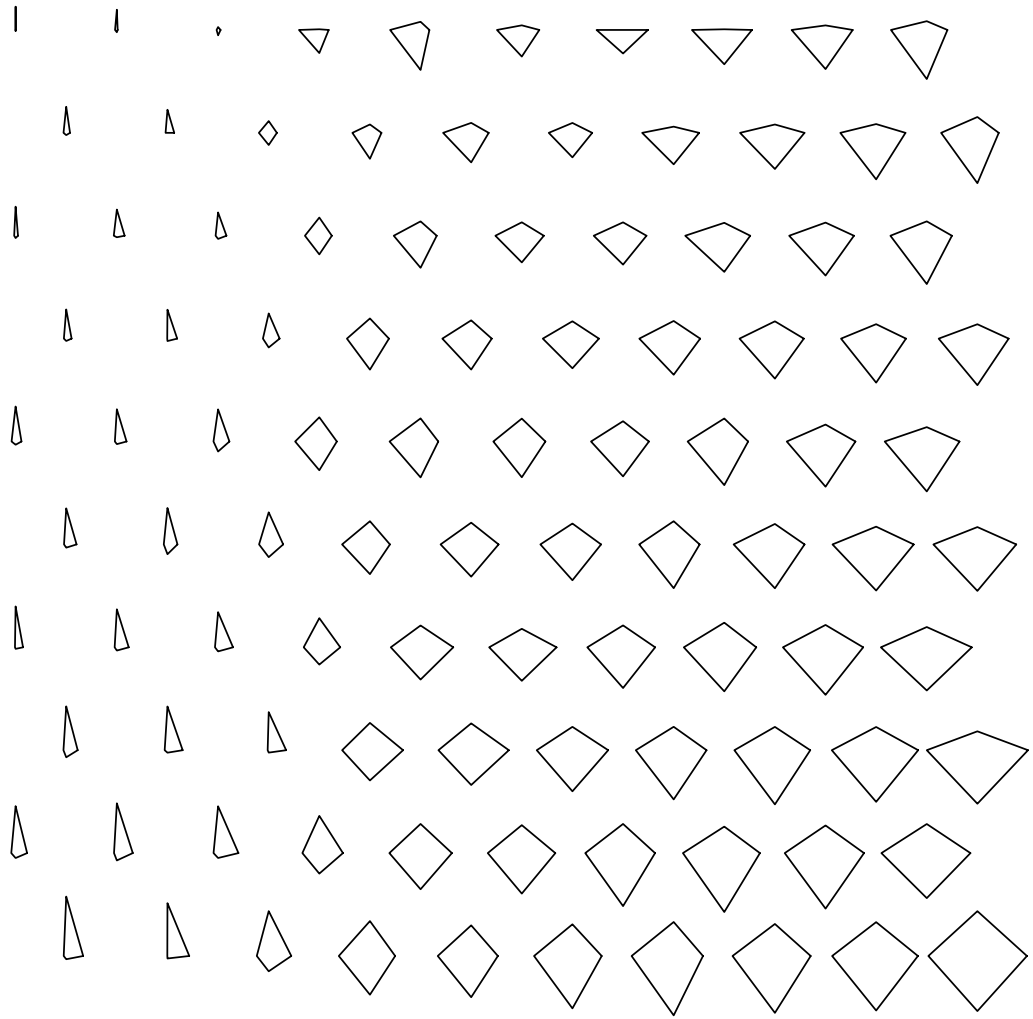
Why does the SOM shine?

The SOM is a visualization framework

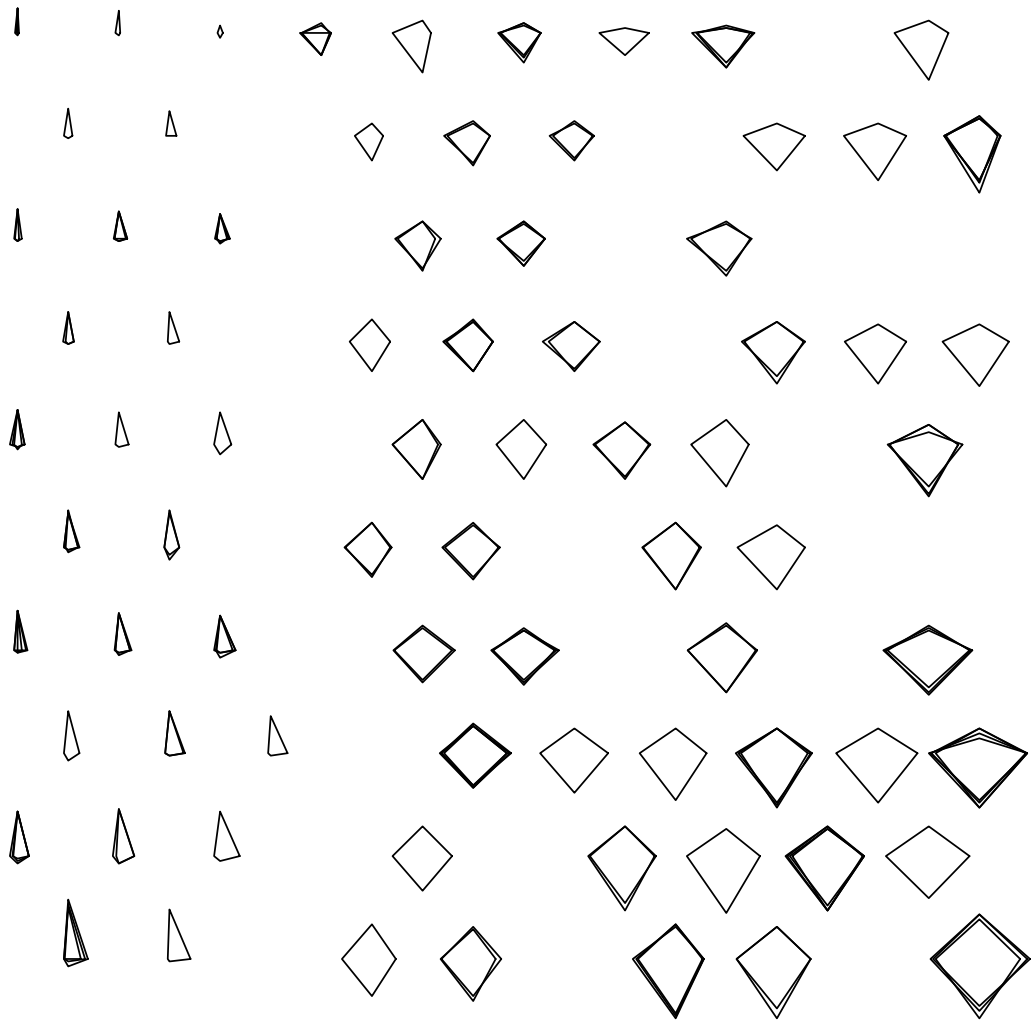
- glyph based visualization
- component planes
- hit map (data histograms)
- U matrix
- you name it...



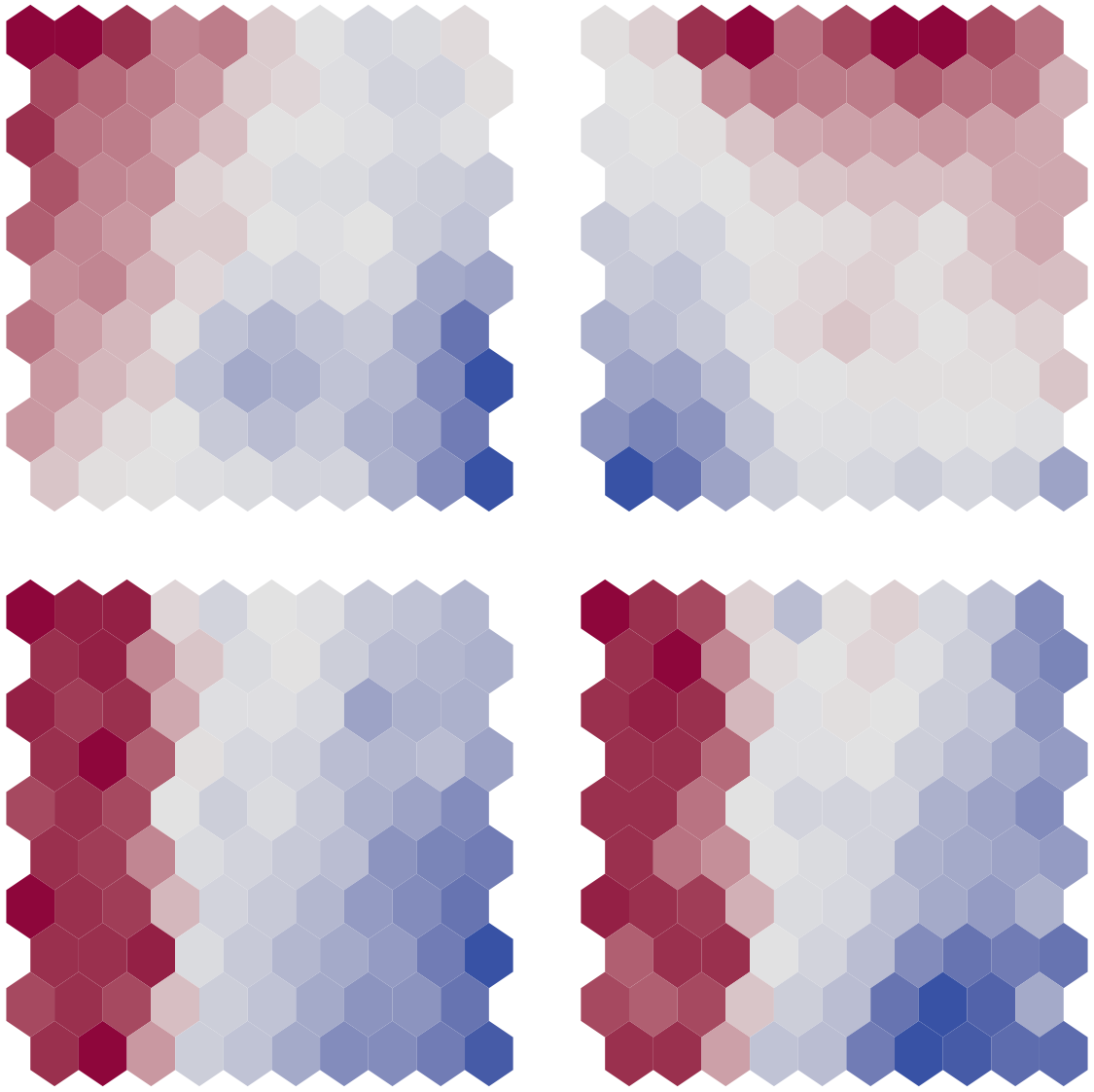
Mystery Dataset



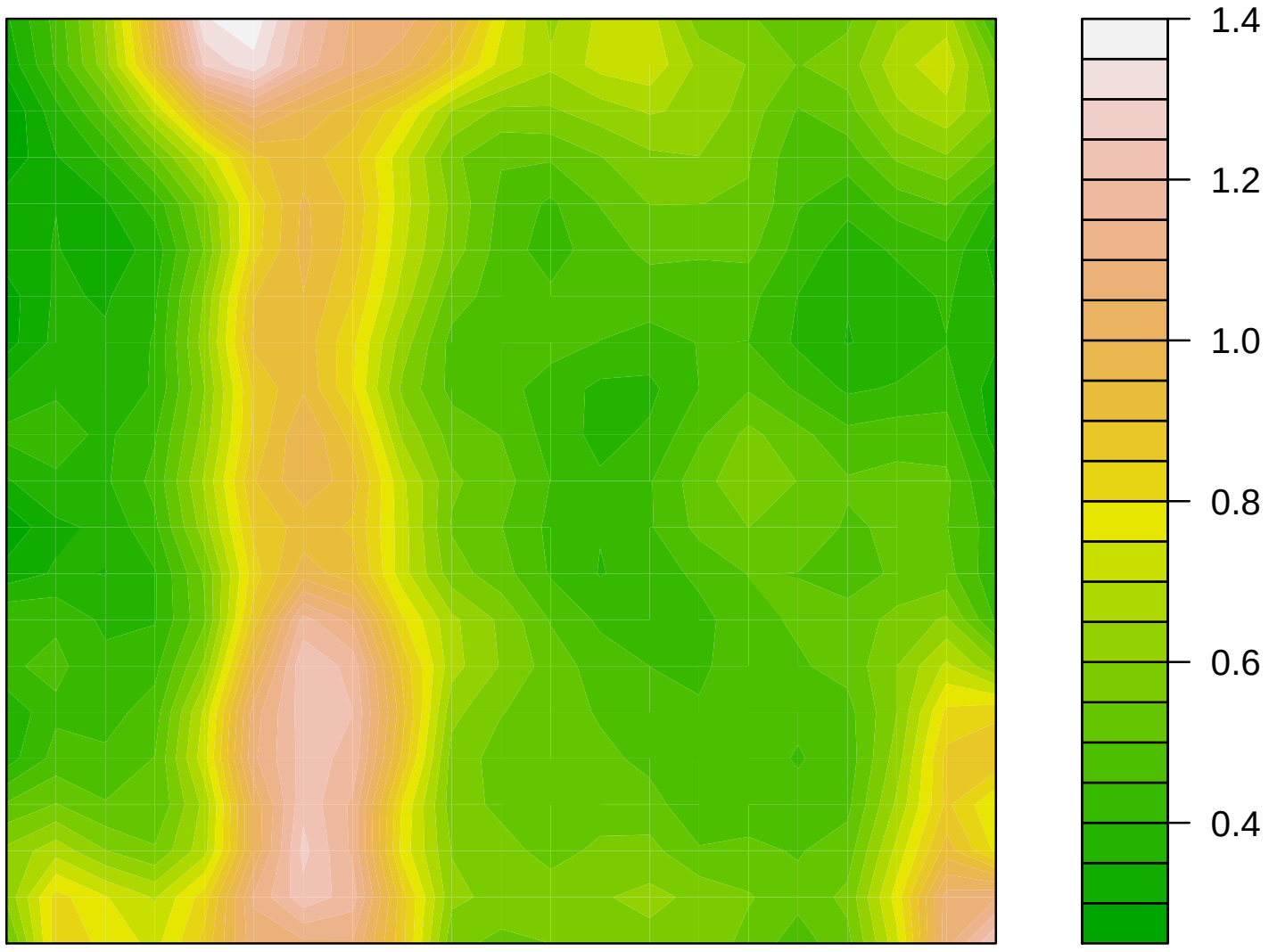
Mystery Dataset



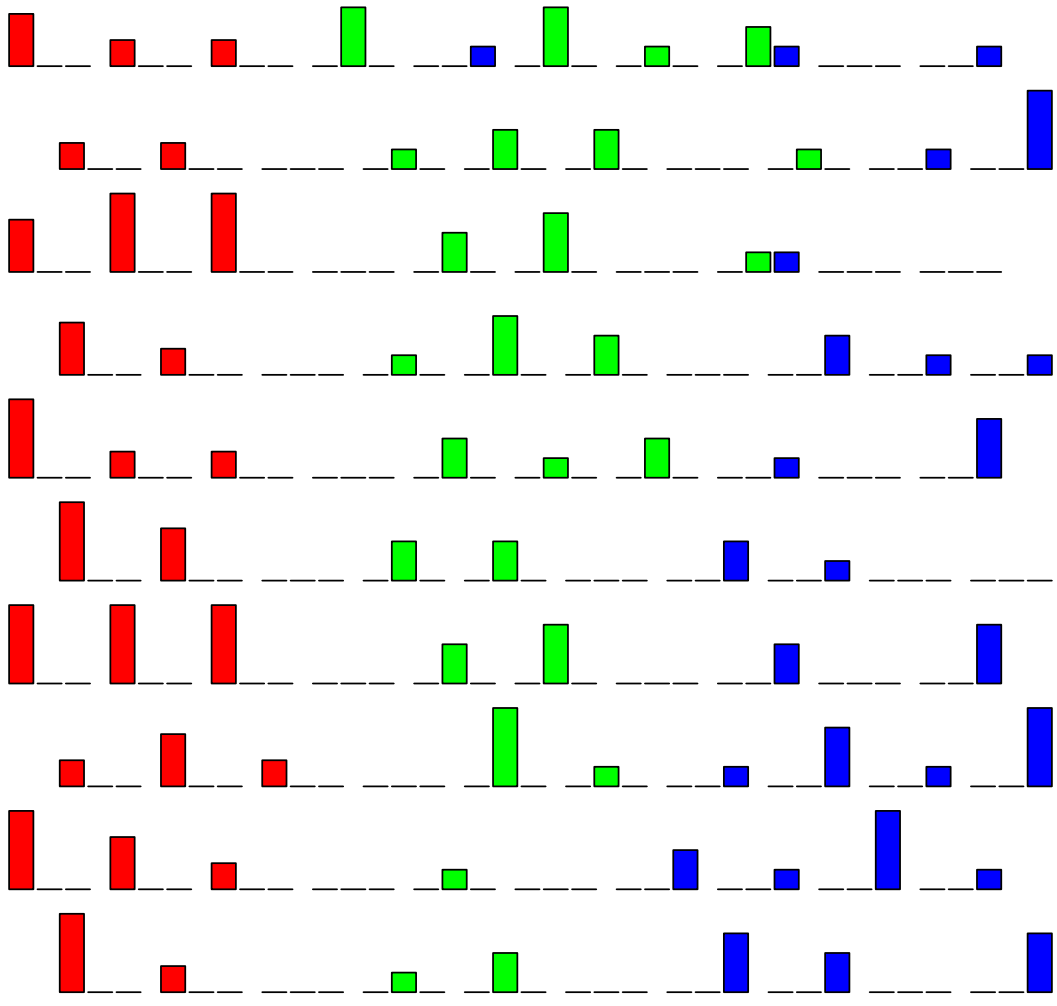
Mystery Dataset



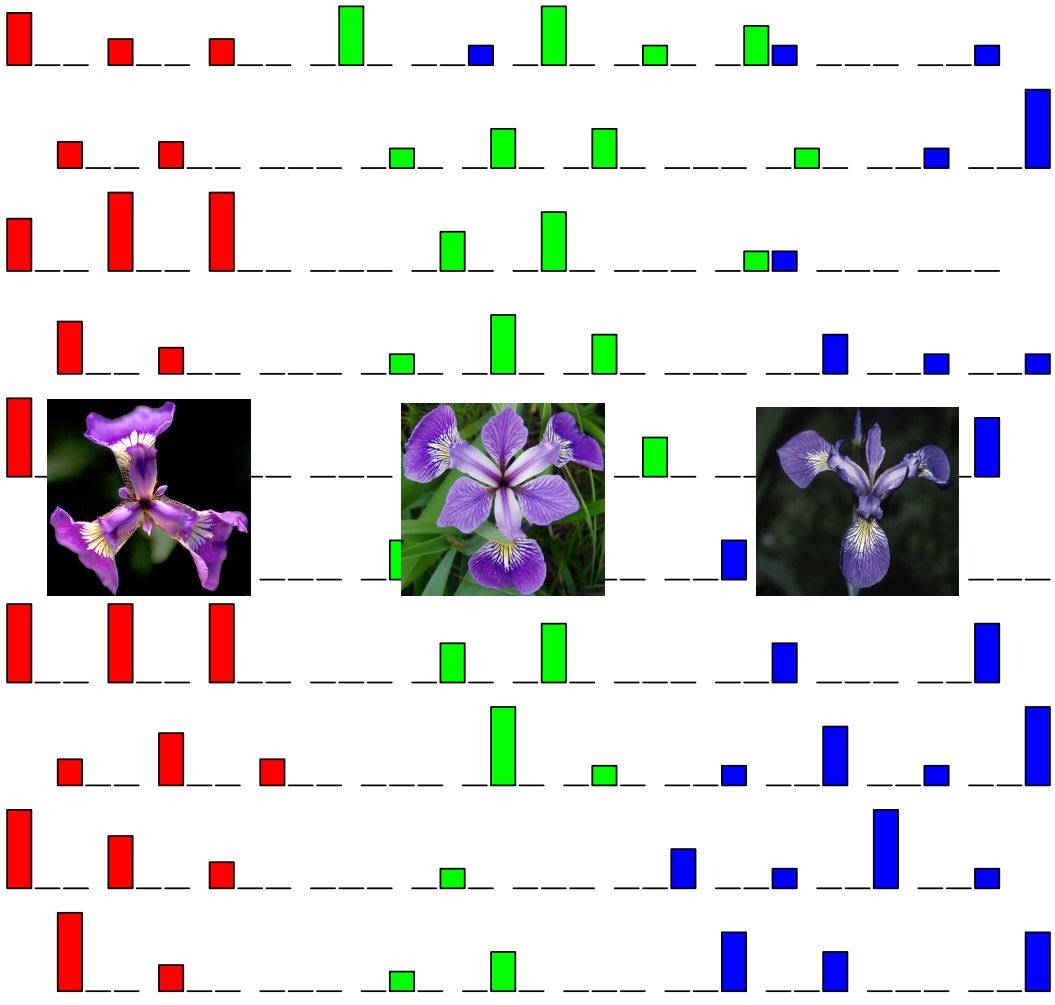
Mystery Dataset



Mystery Dataset



Mystery Dataset



Adapting to non vector data

Vector space algorithms

- BMU: $\|x - m_k(t)\|^2$
- prototype update: $h_{kc_i(t)}(t)x_i$

Vector space visualizations

- glyph based visualisation: direct use of coordinates
- component planes: direct use of coordinates
- U matrix and variants: $\|m_k - m_l\|^2$

Median SOM

[Kohonen, 1996, Kohonen and Somervuo, 1998]

Prototype update as an optimization problem

$$m_k(t+1) = \frac{\sum_{i=1}^{N} h_{kc_i(t)}(t) x_i}{\sum_{i=1}^{N} h_{kc_i(t)}(t)}$$

is equivalent to

$$m_k(t+1) = \arg\min_{m \in \mathbb{R}^p} \sum_{i=1}^{N} h_{kc_i(t)}(t) \|m - x_i\|^2.$$

A simple solution

- replace $\|m - x_i\|^2$ by $d(m, x_i)$
- constraint the m_k to be chosen in $\{x_1, \ldots, x_N\}$

Median SOM

[Kohonen, 1996, Kohonen and Somervuo, 1998]

Prototype update as an optimization problem

$$m_k(t+1) = \frac{\sum_{i=1}^{N} h_{kc_i(t)}(t) x_i}{\sum_{i=1}^{N} h_{kc_i(t)}(t)}$$

is equivalent to

$$m_k(t+1) = \arg\min_{m \in \mathbb{R}^p} \sum_{i=1}^{N} h_{kc_i(t)}(t) \|m - x_i\|^2.$$

A simple solution

- replace $\|m - x_i\|^2$ by $d(m, x_i)$
- constraint the m_k to be chosen in $\{x_1, \ldots, x_N\}$
- or not if the search in $\mathcal{X}$ is doable [Somervuo, 2003].

Median SOM

[Kohonen, 1996, Kohonen and Somervuo, 1998]

Batch Median SOM

1. compute the *best matching unit* for all data points

$$c_i(t) = \arg \min_{k \in \{1,\dots,K\}} d(m_k(t), x_i)$$

2. update all prototypes

$$m_k(t+1) = \arg \min_{m \in \mathcal{X}} \sum_{i=1}^{N} h_{kc_i(t)}(t) d(m, x_i)$$

3. loop to 1 until convergence

Numerous variants

- stochastic variation [Ambroise and Govaert, 1996]
- BMU variation [Kohonen and Somervuo, 2002, El Golli et al., 2004]
- collision avoidance [Rossi, 2007]

Median SOM

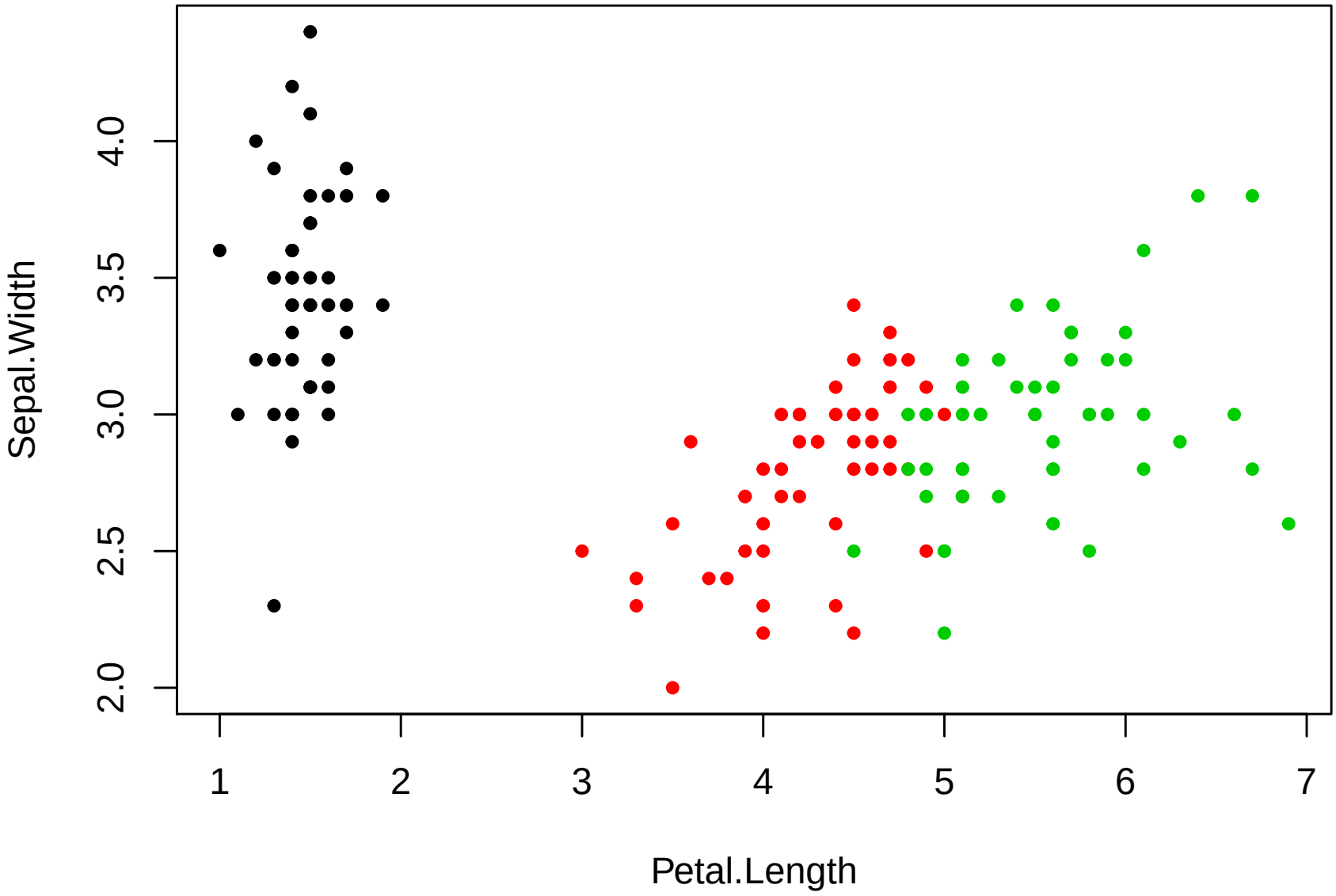
Pros

- straightforward (slow) implementation
- no approximation and no assumption on d

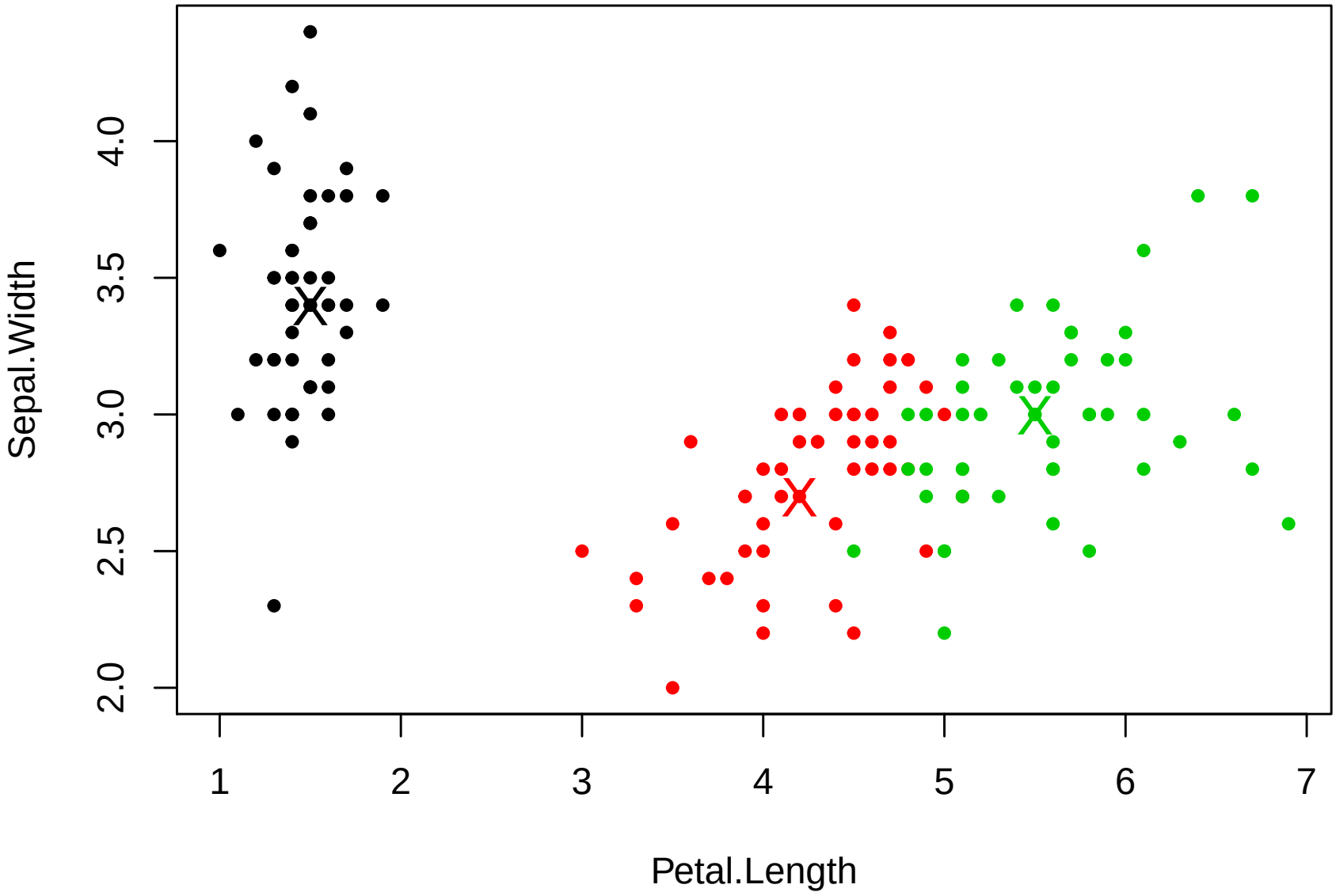
Cons

- slow: $O(N^2 + NK^2)$ per iteration with a fast implementation [Conan-Guez et al., 2006, Conan-Guez and Rossi, 2007]
- quantization quality limitation
- no interpolation effect
- massive folding (prototype collision [Rossi, 2007])

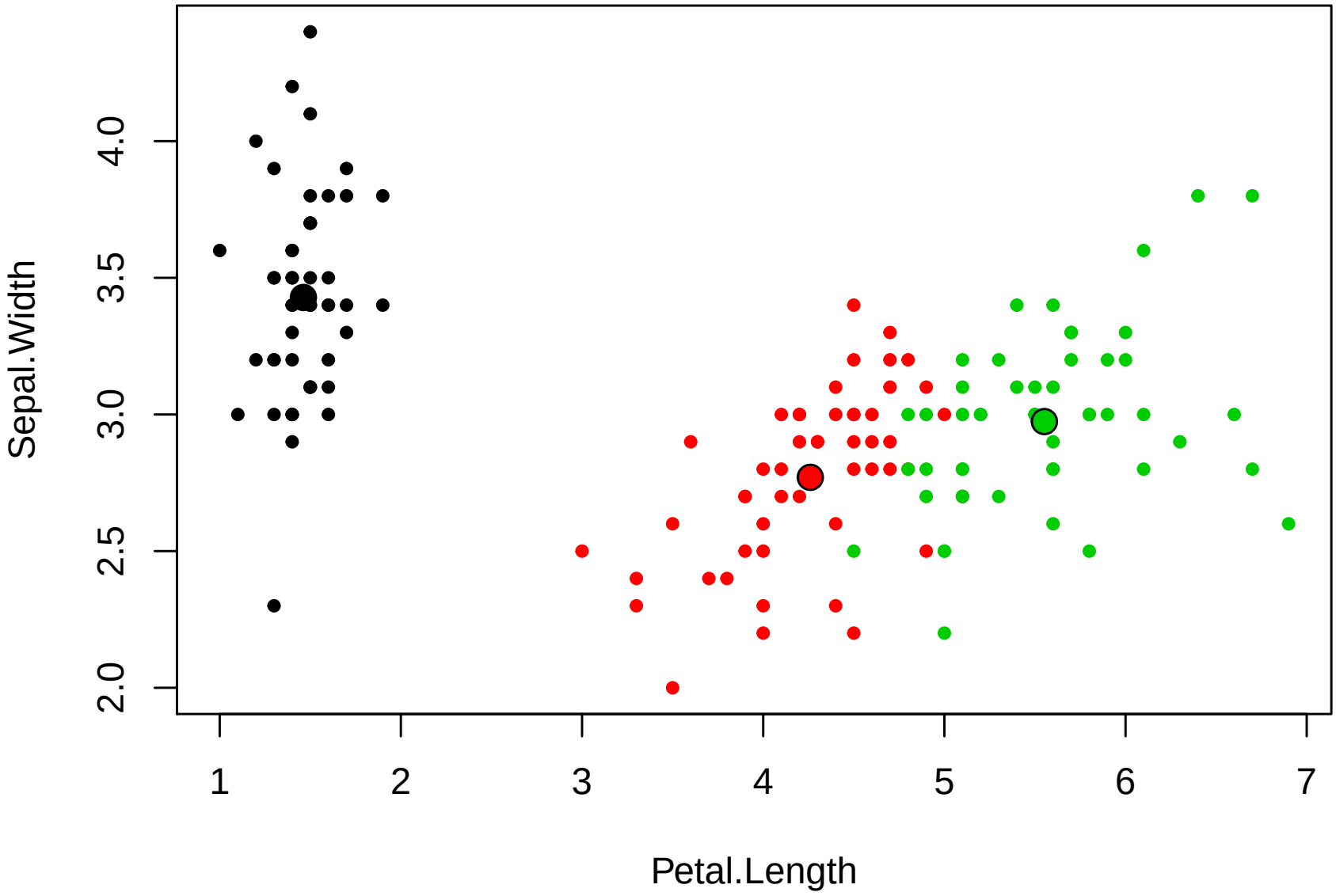
Quantization limit



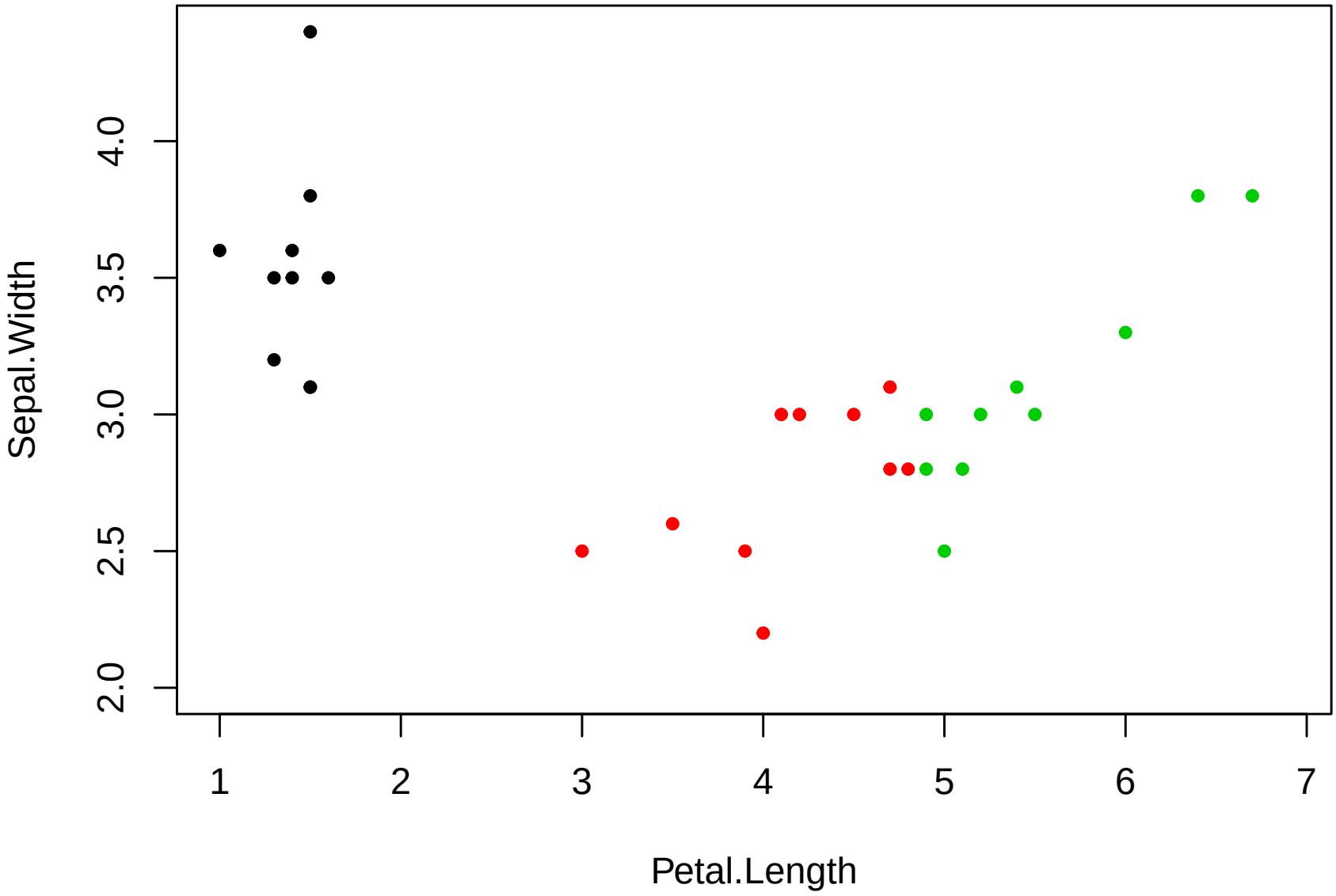
Quantization limit



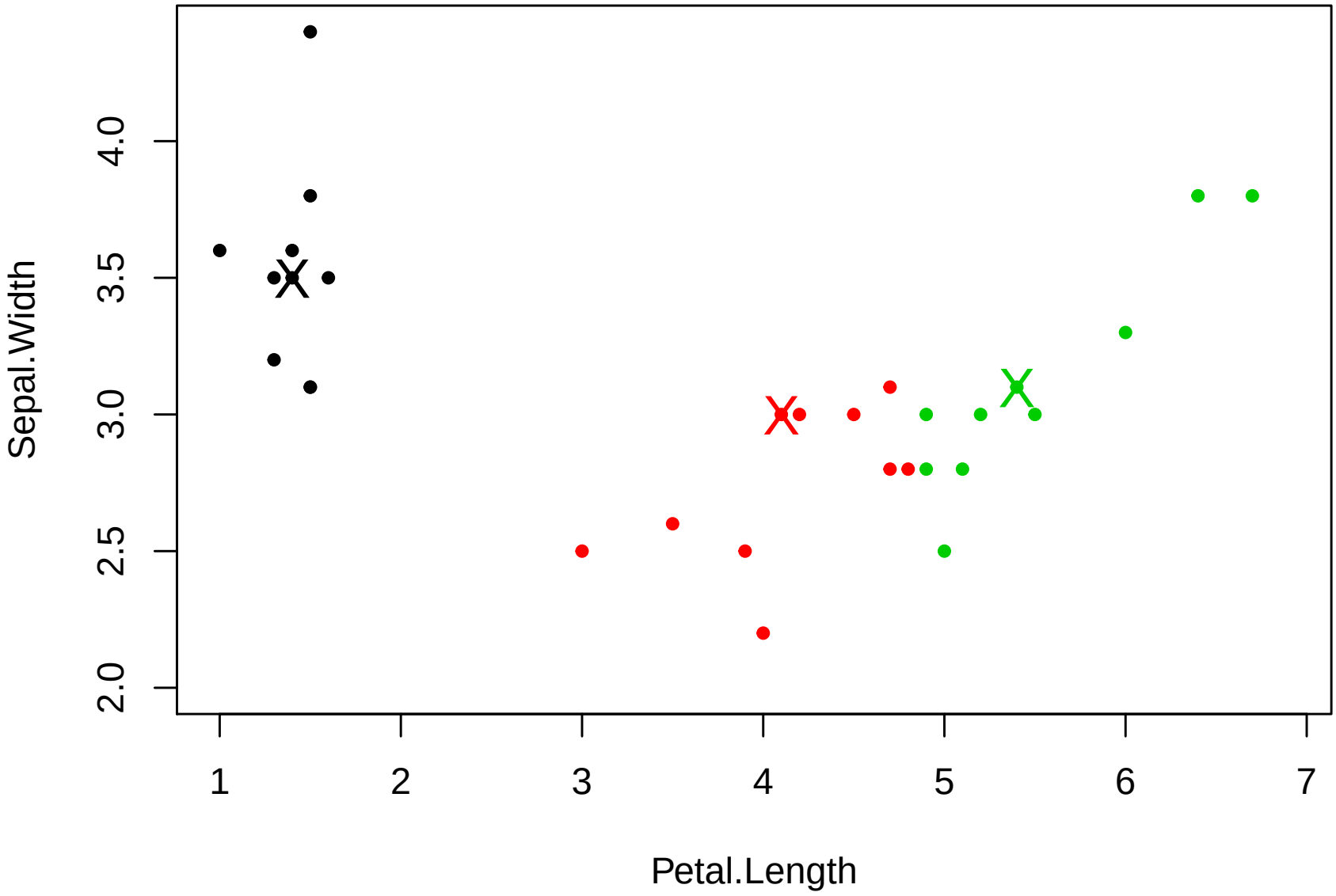
Quantization limit



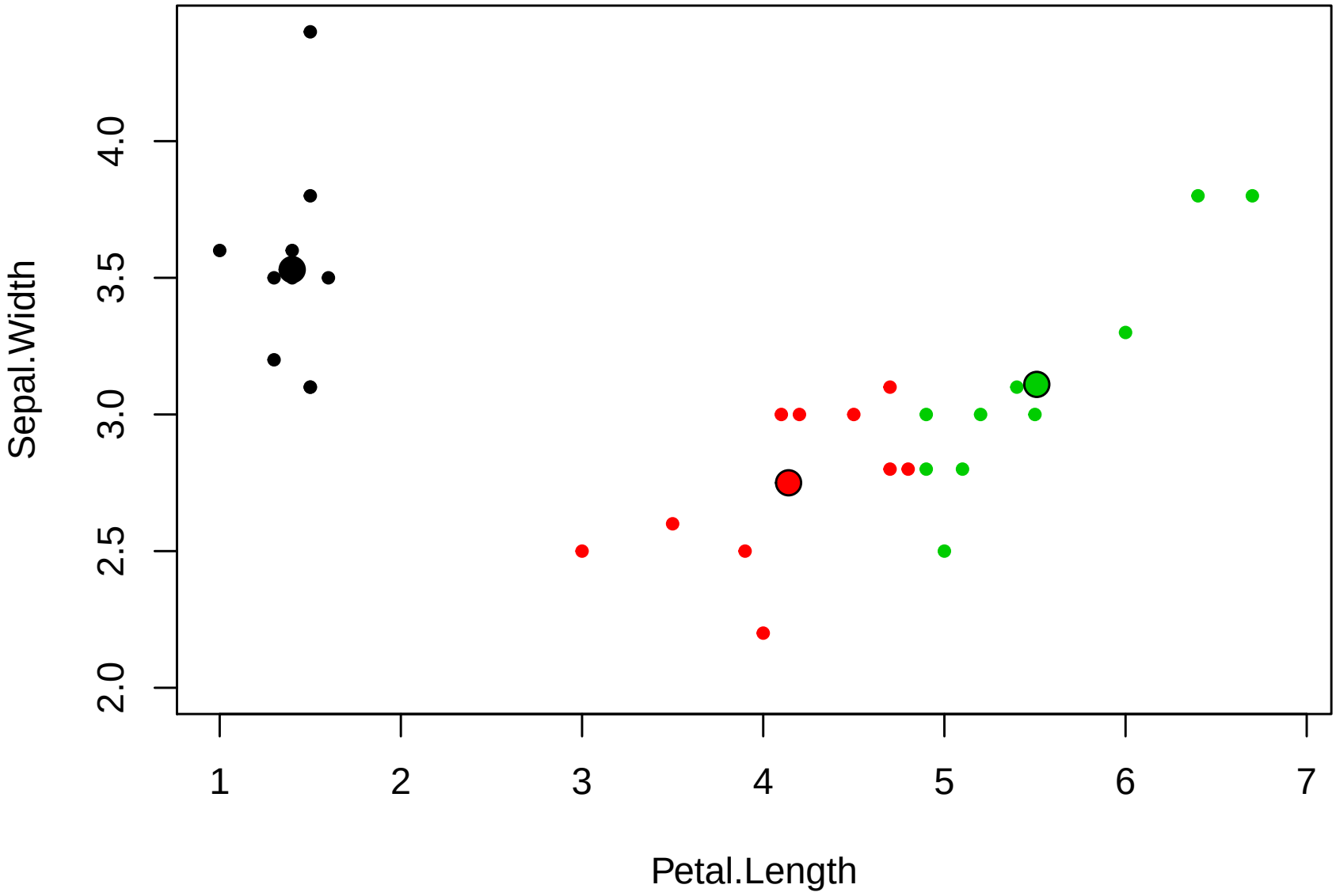
Quantization limit



Quantization limit



Quantization limit



Iris demo

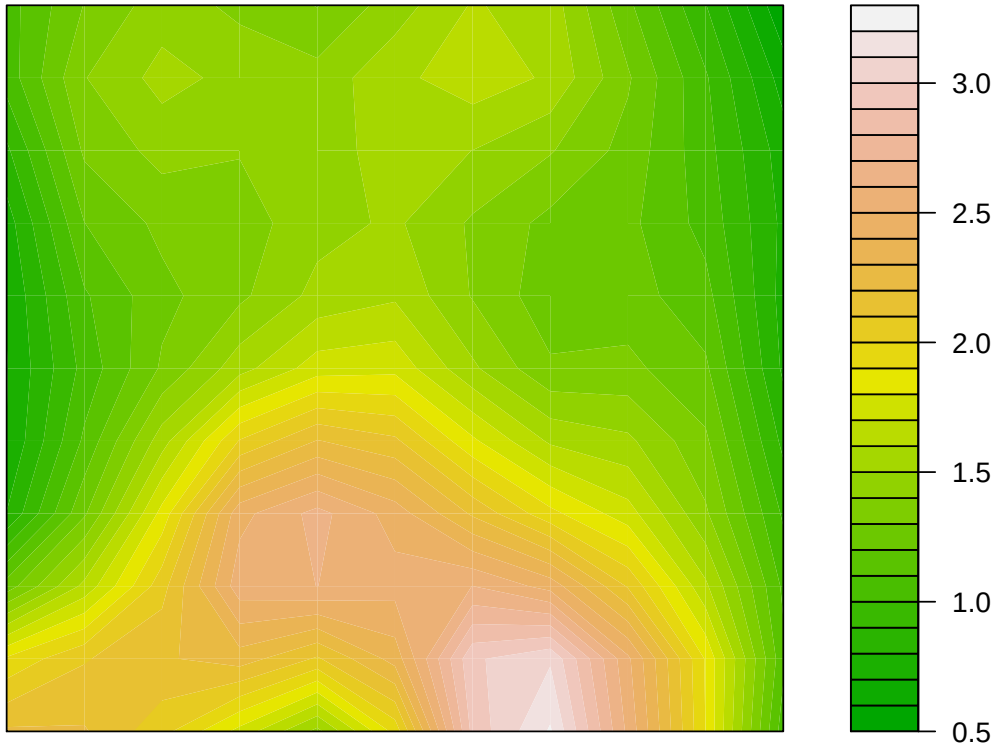
Strong limit on K

- at least one observation per unit
- test with $K = 25$ (5×5 grid)

Iris demo

Strong limit on K

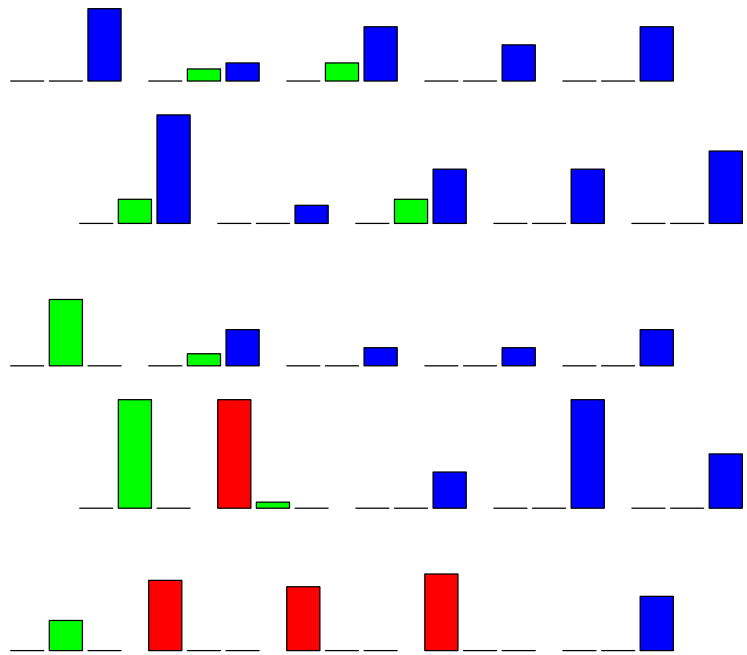
- at least one observation per unit
- test with $K = 25$ (5×5 grid)



Iris demo

Strong limit on K

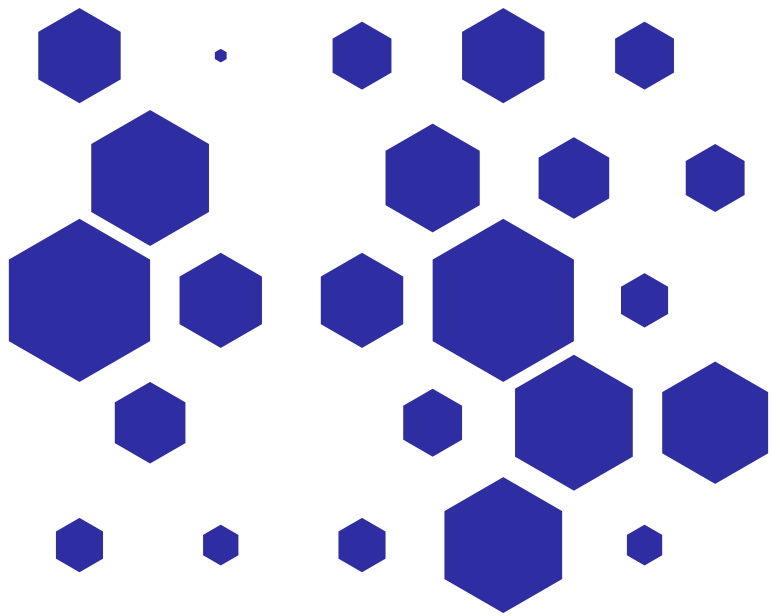
- at least one observation per unit
- test with $K = 25$ (5×5 grid)



Iris demo

Strong limit on K

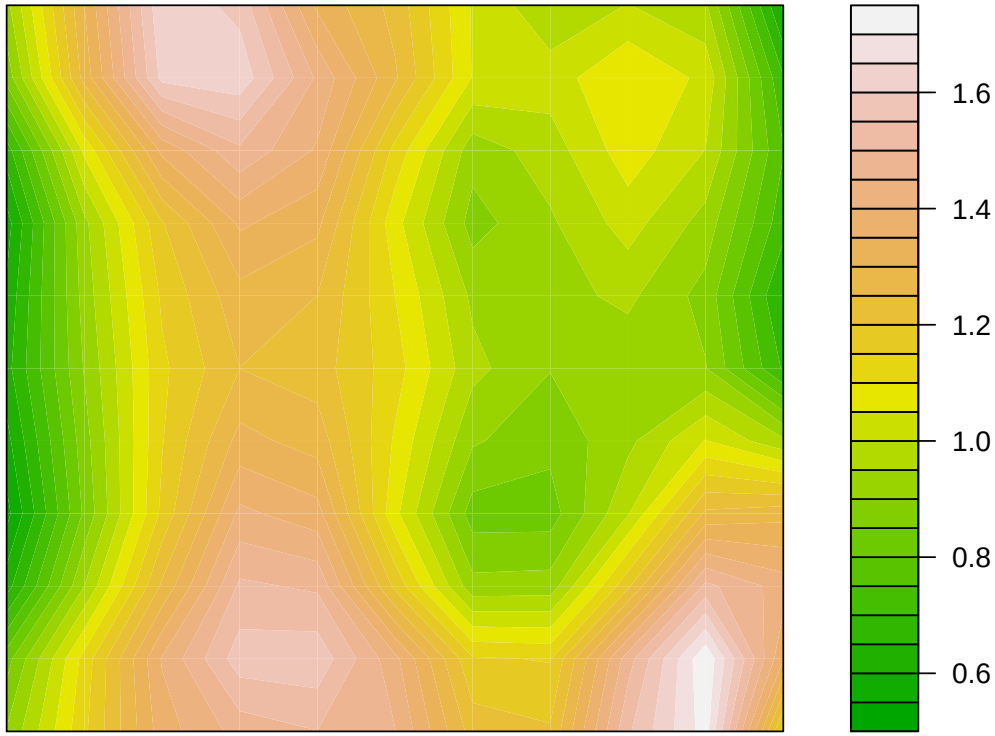
- at least one observation per unit
- test with $K = 25$ (5×5 grid)



Iris demo

Strong limit on K

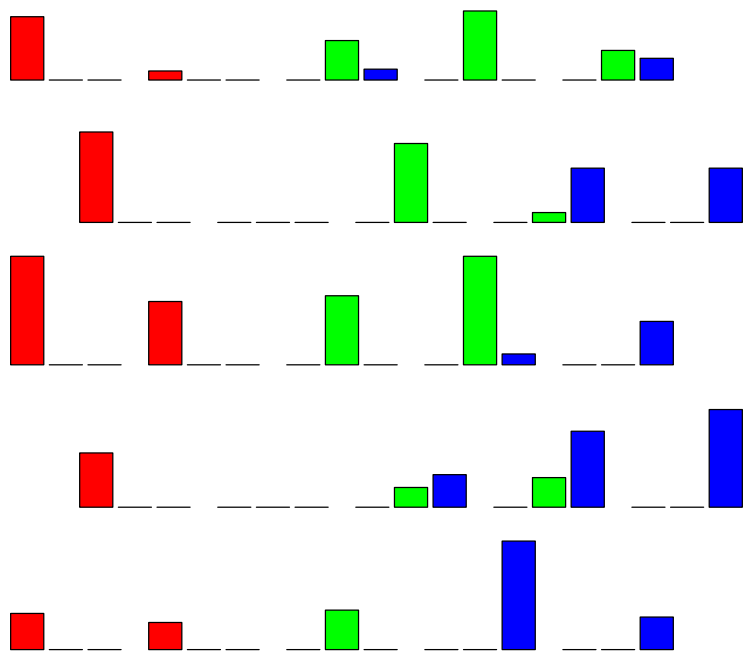
- at least one observation per unit
- test with $K = 25$ (5×5 grid)



Iris demo

Strong limit on K

- at least one observation per unit
- test with $K = 25$ (5×5 grid)



Energy functions

Heskes' Energy function

A variant of the SOM can be obtained by trying to solve the following optimization problem [Heskes and Kappen, 1993]

$$(m(t), c(t)) = \arg\min_{m,c} \sum_{k=1}^{K} \sum_{i=1}^{N} h_{kc_i}(t) \|m_k - x_i\|^2.$$

The BMU is now $c_i(t) = \arg\min_{k \in \{1,\ldots,K\}} \sum_{l=1}^{K} h_{kl}(t) \|x_i - m_l(t)\|^2$.

Energy functions

Heskes' Energy function

A variant of the SOM can be obtained by trying to solve the following optimization problem [Heskes and Kappen, 1993]

$$(m(t), c(t)) = \arg\min_{m,c} \sum_{k=1}^{K} \sum_{i=1}^{N} h_{kc_i}(t) \|m_k - x_i\|^2.$$

The BMU is now $c_i(t) = \arg\min_{k \in \{1,\ldots,K\}} \sum_{l=1}^{K} h_{kl}(t) \|x_i - m_l(t)\|^2$.

Equivalent problem

this is equivalent to solving
[Graepel et al., 1998, Graepel and Obermayer, 1999]

$$c(t) = \arg\min_{c} \frac{1}{2} \sum_{k=1}^{K} \frac{1}{\sum_{i=1}^{N} h_{kc_i}(t)} \sum_{i=1}^{N} \sum_{j=1}^{N} h_{kc_i}(t) h_{kc_j}(t) \|x_i - x_j\|^2.$$

Dissimilarity version

Graepel et al.'s proposal

Rather than optimizing

$$\sum_{k=1}^{K}\sum_{i=1}^{N} h_{kc_i}(t)d(m_k, x_i)$$

with coordinate descent over m and c, optimize

$$\frac{1}{2}\sum_{k=1}^{K}\frac{1}{\sum_{i=1}^{N} h_{kc_i}(t)}\sum_{i=1}^{N}\sum_{j=1}^{N} h_{kc_i}(t)h_{kc_j}(t)d(x_i, x_j)$$

with *deterministic annealing*.

No more equivalence

- equivalence only in a Euclidean space
- if d does not fulfill the triangular inequality, potentially they lead to very different solutions:
  - $d(m_k, x_i)$ is a *quantization* oriented measure
  - $d(x_i, x_j)$ is a *clustering* oriented measure

Soft Topographic Mapping for Proximity Data

[Graepel et al., 1998, Graepel and Obermayer, 1999]

Features

- based on a mean field $\simeq$ prototypes
- soft assignments
- two loops: EM like algorithm embedded into an annealing loop

Pros

- leverage the good properties of deterministic annealing
- no assumption on d

Cons

- sophisticated algorithm in which annealing control is crucial
- fixed neighborhood (effects of on the fly modifications are unclear)
- slow: $O(N^2K + NK^2)$ per iteration in two loops!

Relational approach

The relational idea

- N points $(x_i)_{1,\dots,N}$ in a Hilbert space $\mathcal{H}$
- N real valued coefficients $\alpha^T = (\alpha_i)_{i=1,\dots,N}$ with $\sum_{i=1}^N \alpha_i = 1$
- then we have [Hathaway et al., 1989]

$$\left\| x_i - \sum_{i=1}^N \alpha_i x_i \right\|_{\mathcal{H}}^2 = (D\alpha)_i - \frac{1}{2}\alpha^T D\alpha,$$

with $D_{ij} = \|x_i - x_j\|_{\mathcal{H}}^2$.

The relational trick

- $(x_i)_{i=1,\dots,N}$ in $(\mathcal{X}, d)$
- define a set of "pseudo linear combination" $\mathcal{A} = \{\alpha \in \mathbb{R}^N | \sum_{i=1}^N \alpha_i = 1\}$
- extend d to $\mathcal{A} \times \mathcal{X}$ via $d_r(\alpha, x_i) = (D\alpha)_i - \frac{1}{2}\alpha^T D\alpha$.

Relational Variants

Prototypes based methods

- in the batch SOM, $m_k(t+1) = \sum_{i=1}^{N} \alpha_k(t+1)_i x_i$ with

$$\alpha_k(t+1)_i = \frac{h_{kc_i(t)}(t)}{\sum_{i=1}^{N} h_{kc_i(t)}(t)}.$$

- then $\alpha_k(t+1) \in \mathcal{A}$ and we can define $d(m_k(t+1), x_i)$ as $d_r(\alpha_k(t+1), x_i)$

Variants

- c-means [Hathaway et al., 1989]
- batch SOM and batch neural gas [Hammer et al., 2007]
- online SOM [Olteanu et al., 2013]

Relational SOM

[Hammer et al., 2007]

Batch version

1. compute the *best matching unit* for all data points

$$c_i(t) = \arg \min_{k \in \{1,\ldots,K\}} d_r(\alpha_k(t), x_i),$$

   where d_r is the relational extension of the d
2. update all prototypes

$$\alpha_k(t+1)_i = \frac{h_{kc_i(t)}}{\sum_{l=1}^{N} h_{kc_l(t)}}.$$

3. loop to 1 until convergence

Theoretical justification

- corresponds to an embedding of D/d into a pseudo Euclidean space
- details in [Hammer and Hasenfuss, 2010]

Relational SOM

[Hammer et al., 2007]

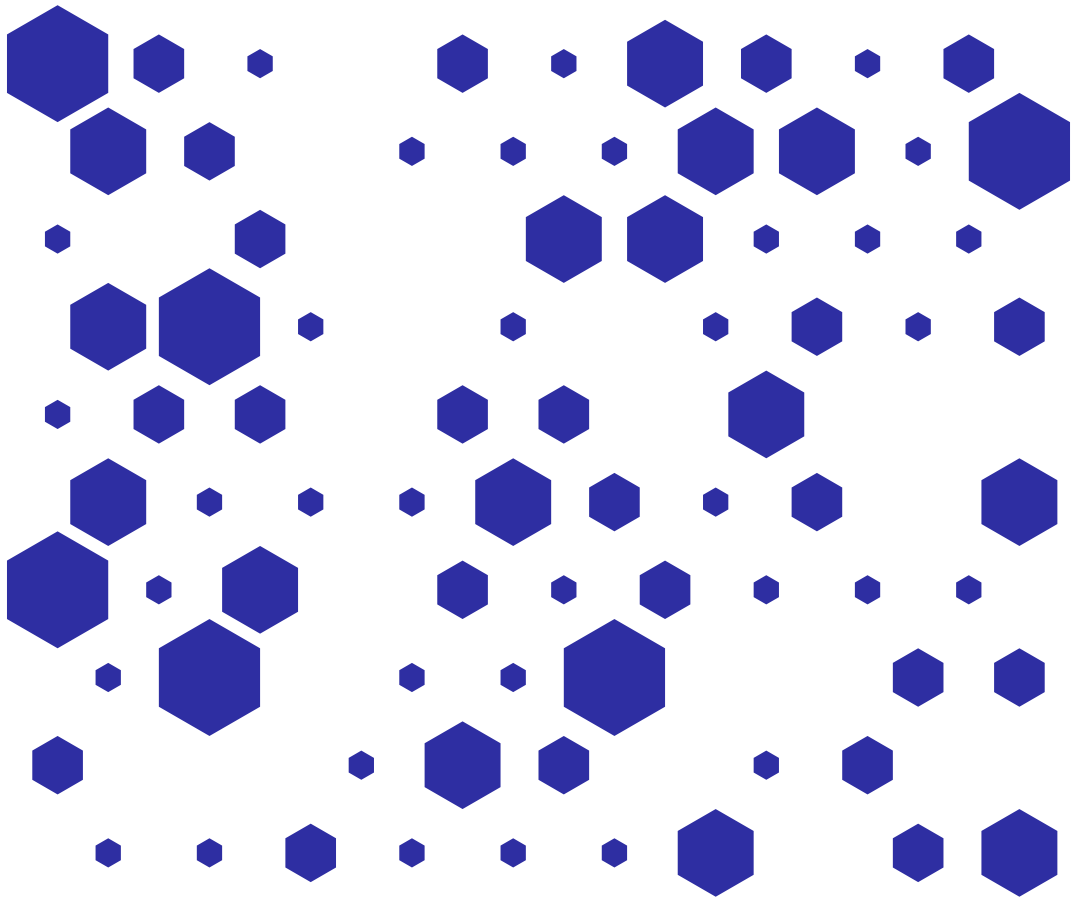
Pros

- straightforward implementation
- no approximation and no assumption on d
- theoretical guarantees

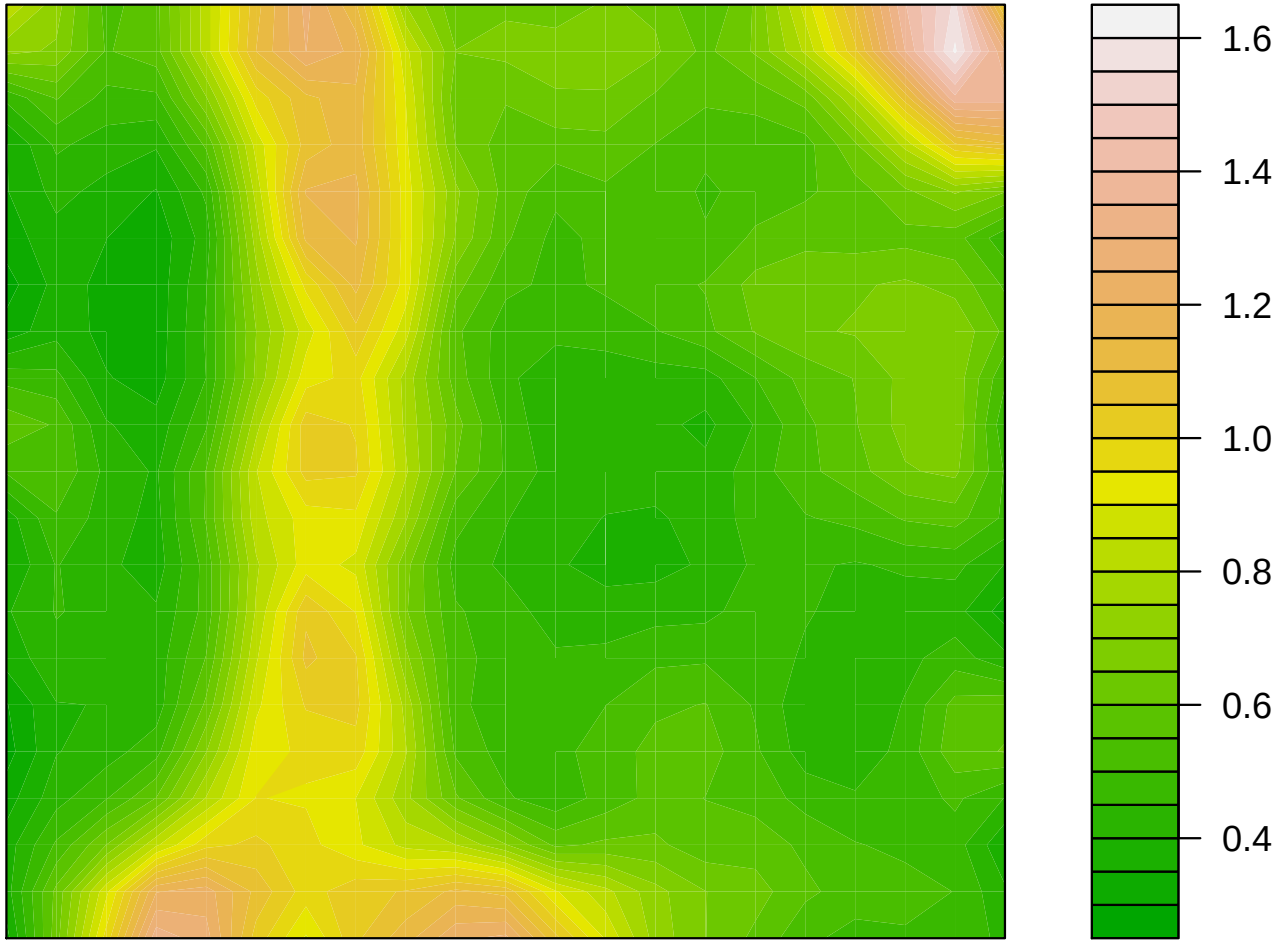
Cons

- slow: $O(KN^2)$ per iteration
- prototypes are meaningless

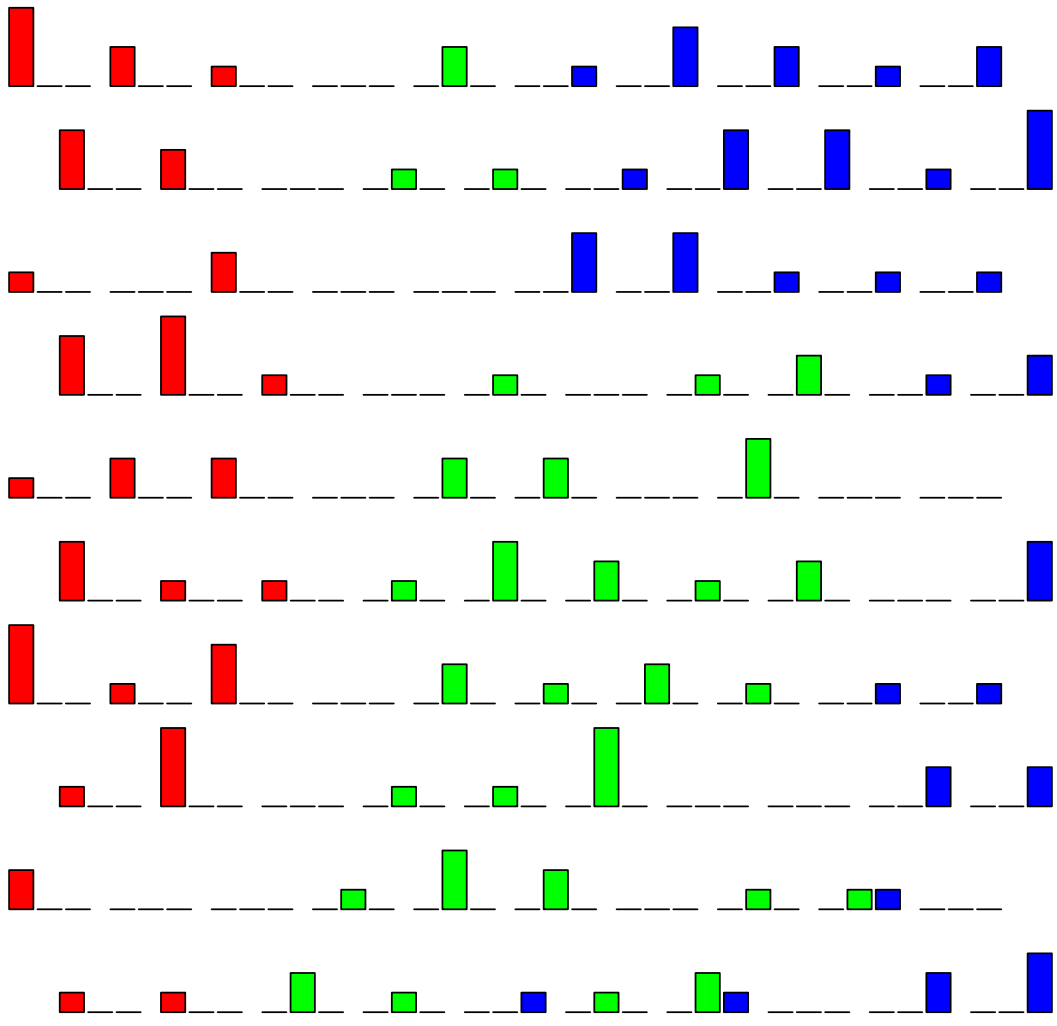
Iris Demo



Iris Demo



Iris Demo



An easier road

Kernel data

- easier to deal with because of the stronger assumption on K/k
- a kernel on $\mathcal{X}$ is associated to a Hilbert space $\mathcal{H}$ via a mapping ϕ
- main idea: implement a SOM in $\mathcal{H}$

Kernel trick

- standard tool of kernel methods
- first used for the SOM in [Graepel et al., 1998]
- if $m_k(t) = \sum_{i=1}^{N} \alpha_{ki}(t)\phi(x_i)$, then

$$\begin{aligned}\|\phi(x_i) - m_k(t)\|_{\mathcal{H}}^2 =& k(x_i, x_i) - 2\sum_{j=1}^{N} \alpha_{kj}(t)k(x_k, x_j) \\ &+ \sum_{j=1}^{N}\sum_{l=1}^{N} \alpha_{kj}(t)\alpha_{kl}(t)k(x_j, x_l).\end{aligned}$$

Kernel SOM

Numerous variants

- optimized via deterministic annealing in [Graepel et al., 1998]
- online kernel SOM [Mac Donald and Fyfe, 2000]
- batch kernel SOM [Martín-Merino and Muñoz, 2004, Villa and Rossi, 2007, Boulet et al., 2008]

Pros

- straightforward implementation
- theoretical guarantees (it's a SOM in the kernel space!)

Cons

- slow: $O(N^2K)$ per iteration
- prototypes are meaningless

Equivalence

Relational = kernel

- if K is a kernel matrix, define a dissimilarity matrix by $D_{ij} = K_{ii} + K_{jj} - 2K_{ij}$
- then for $\alpha \in \mathbb{R}^N$ such that $\sum_{i=1}^{N} \alpha_i = 1$

$$\underbrace{(D\alpha)_i - \frac{1}{2}\alpha^T D\alpha}_{\text{Relational BMU}} = \underbrace{K_{ii} - 2\sum_{j=1}^{N} K_{ij}\alpha_j + \sum_{j=1}^{N}\sum_{l=1}^{N} \alpha_j \alpha_l K_{jl}}_{\text{Kernel BMU}} .$$

- absolutely identical results
- the relational SOM is a (strict) extension of the kernel SOM

Soft Topographic Mapping for Proximity Data

STMP internal loop equivalent formulation

1. compute weighted dissimilarities $e_{ik}(t) = \sum_{s=1}^{K} h_{ks} d_r(\alpha_s(t), x_i)$ where d_r is the relational extension of d
2. compute soft assignment

$$\gamma_{ik}(t) = \frac{\exp(-\beta(t) e_{ik}(t))}{\sum_{s=1}^{K} \exp(-\beta(t) e_{is}(t))}.$$

3. update the prototypes

$$\alpha_s(t)_j = \frac{\sum_{k=1}^{K} \gamma_{jk}(t) h_{ks}}{\sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_{ik}(t) h_{ks}}.$$

Deterministic annealing

- is an optimization technique
- SMTP = DA relational SOM

There can be only one

Data type

- Vector data: euclidean SOM
- Dissimilarity/Kernel data: relational SOM

Optimization strategy

- online
- batch
- deterministic annealing

Arbitrary combination

- kernel data + DA: STMK [Graepel et al., 1998]
- dissimilarity data + online: online relational SOM [Olteanu et al., 2013]
- etc.

Computational costs

Cost for one iteration

Algorithm	Assignment cost	Prototype update cost
Batch SOM	$O(NKp)$	$O(NKp)$
Online SOM	$O(Kp)$	$O(Kp)$
Median SOM	$O(NK)$	$O(N^2 + NK^2)$
Batch relational SOM	$O(N^2K)$	$O(NK)$
Online relational SOM	$O(N^2K)$	$O(NK)$
STVQ	$O(NKp + NK^2)$	$O(NKp + NK^2)$
STMK/STMP	$O(N^2K + NK^2)$	$O(NK^2)$

Remarks

- processing one data point in the online relational SOM is as costly as processing the full data set in the batch relational SOM
- dual loop for the ST$\alpha\beta$ variants

And the winner is...

And the winner is...

The batch relational SOM

- generic (includes the kernel case)
- interpolation effects and good quantization
- not as costly as the STMP
- faster than the online relational SOM (but needs a proper initialization)

Visualization

- neither component planes nor glyph based visualisation
- hit map
- u matrix and variants (using the relational trick to compute dissimilarities between prototypes)

Open issues

Optimization

- no extensive comparison of STMP to batch relational SOM exists
  - see [Hammer and Hasenfuss, 2010] for Neural Gas
- can we optimize directly the clustering cost?
  - relational K means is outperformed by such an approach, see [Conan-Guez and Rossi, 2012]

Algorithmic cost

- $O(N^2K)$ is unacceptable for large data
  - for $N = 20\,000$ and $K = 10 \times 10$, one iteration can cost several seconds for a standard implementation
  - for $N = 100\,000$ and $K = 20 \times 20$: several minutes
- Nyström approximation [Williams and Seeger, 2001]?
  - see [Gisbrecht et al., 2012] for GTM

What about zero dissimilarity SOM?

Usability of the results

- Reduced visualization possibilities (compared to vector SOM)
- No user based evaluation available
- is it really useful on a data exploration point of view?

What about zero dissimilarity SOM?

Usability of the results

- Reduced visualization possibilities (compared to vector SOM)
- No user based evaluation available
- is it really useful on a data exploration point of view?

Embedding then vector SOM?

- compute a vector embedding of D into $\mathbb{R}^p$ and then apply a vector SOM
- cost based embedding methods are in $O(N \log N)$ per iteration with Barnes and Hut approximation or $O(N^2)$ without
- total cost dominated by $O(N^2)$ if p is small

Thanks!
Questions?

References I

Ambroise, C. and Govaert, G. (1996).
Analyzing dissimilarity matrices via Kohonen maps.
In *Proceedings of 5th Conference of the International Federation of Classification Societies (IFCS 1996)*, volume 2, pages 96–99, Kobe (Japan).

Boulet, R., Jouve, B., Rossi, F., and Villa, N. (2008).
Batch kernel SOM and related Laplacian methods for social network analysis.
Neurocomputing, 71(7–9):1257–1273.

Conan-Guez, B. and Rossi, F. (2007).
Speeding up the dissimilarity self-organizing maps by branch and bound.
In Sandoval, F., Prieto, A., Cabestany, J., and Graña, M., editors, *Computational and Ambient Intelligence (Proceedings of 9th International Work-Conference on Artificial Neural Networks, IWANN 2007)*, volume 4507 of *Lecture Notes in Computer Science*, pages 203–210, San Sebastián (Spain). Springer Berlin / Heidelberg.

Conan-Guez, B. and Rossi, F. (2012).
Dissimilarity clustering by hierarchical multi-level refinement.
In *Proceedings of the XXth European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2012)*, pages 483–488, Bruges, Belgique.

Conan-Guez, B., Rossi, F., and El Golli, A. (2006).
Fast algorithm and implementation of dissimilarity self-organizing maps.
Neural Networks, 19(6–7):855–863.

El Golli, A., Conan-Guez, B., and Rossi, F. (2004).
Self organizing map and symbolic data.
Journal of Symbolic Data Analysis, 2(1).

References II


Gisbrecht, A., Mokbel, B., Schleif, F.-M., Zhu, X., and Hammer, B. (2012).
Linear time relational prototype based learning.
Int. J. Neural Syst., 22(5).


Graepel, T., Burger, M., and Obermayer, K. (1998).
Self-organizing maps: Generalizations and new optimization techniques.
Neurocomputing, 21:173–190.


Graepel, T. and Obermayer, K. (1999).
A stochastic self-organizing map for proximity data.
Neural Computation, 11(1):139–155.


Hammer, B. and Hasenfuss, A. (2010).
Topographic mapping of large dissimilarity data sets.
Neural Computation, 22(9):2229–2284.


Hammer, B., Hasenfuss, A., Rossi, F., and Strickert, M. (2007).
Topographic processing of relational data.
In *Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 07)*, Bielefeld (Germany).


Hathaway, R. J., Davenport, J. W., and Bezdek, J. C. (1989).
Relational duals of the c-means clustering algorithms.
Pattern Recognition, 22(2):205–212.

References III


Heskes, T. and Kappen, B. (1993).
Error potentials for self-organization.
In *Proceedings of 1993 IEEE International Conference on Neural Networks (Joint FUZZ-IEEE'93 and ICNN'93 [IJCNN93])*, volume III, pages 1219–1223, San Francisco, California. IEEE/INNS.


Kohonen, T. (1996).
Self-organizing maps of symbol strings.
Technical report A42, Laboratory of computer and information science, Helsinki University of technology, Finland.


Kohonen, T. and Somervuo, P. J. (1998).
Self-organizing maps of symbol strings.
Neurocomputing, 21:19–30.


Kohonen, T. and Somervuo, P. J. (2002).
How to make large self-organizing maps for nonvectorial data.
Neural Networks, 15(8):945–952.


Mac Donald, D. and Fyfe, C. (2000).
The kernel self organising map.
In *Proceedings of 4th International Conference on knowledge-based intelligence engineering systems and applied technologies*, pages 317–320.


Martín-Merino, M. and Muñoz, A. (2004).
Extending the som algorithm to non-euclidean distances via the kernel trick.
In Pal, N., Kasabov, N., Mudi, R., Pal, S., and Parui, S., editors, *Neural Information Processing*, volume 3316 of *Lecture Notes in Computer Science*, pages 150–157. Springer Berlin Heidelberg.

References IV

Olteanu, M., Villa-Vialaneix, N., and Cottrell, M. (2013).
On-line relational som for dissimilarity data.
In Estévez, P. A., Príncipe, J. C., and Zegers, P., editors, *Advances in Self-Organizing Maps*, volume 198 of *Advances in Intelligent Systems and Computing*, pages 13–22. Springer Berlin Heidelberg.

Rossi, F. (2007).
Model collisions in the dissimilarity SOM.
In *Proceedings of XVth European Symposium on Artificial Neural Networks (ESANN 2007)*, pages 25–30, Bruges (Belgium).

Somervuo, P. J. (2003).
Self-organizing map of symbol strings with smooth symbol averaging.
In *Workshop on Self-Organizing Maps (WSOM'03)*, Hibikino, Kitakyushu, Japan.

Villa, N. and Rossi, F. (2007).
A comparison between dissimilarity som and kernel som for clustering the vertices of a graph.

In *Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 07)*, Bielefeld (Germany).

Williams, C. and Seeger, M. (2001).
Using the nyström method to speed up kernel machines.
In *Advances in Neural Information Processing Systems 13*.