Contributions to Unsupervised and Supervised Learning with Applications in Digital Image Processing

Ana I. González Acuña

Department of Computer Science and Artificial Intelligence
The University of the Basque Country

PhD Advisor: D. Manuel Graña Romay

2012

Contents

Section Contents

1. Introduction and overview of Thesis contributions
   - Unsupervised learning
   - Supervised learning

Unsupervised learning

- Non-Stationary Clustering
- Generalized Learning Vector Quantization
- Local Stochastic Learning Rule
- Evolution Strategy
- Occam filters to determine optimal codebook size
- Convergence analysis of the SOM in the GNC theory context

Non-Stationary Clustering

State the problem of Non-Stationary Clustering and related Adaptive Vector Quantization in the context of Color Quantization of image sequences.

Generalized Learning Vector Quantization

Detailed empirical and analytical study of the convergence properties of an unsupervised learning rule, the Generalized Learning Vector Quantization.

- Proposed by Pal, Bezdek & Tsao as a generalization of the SCL algorithm with superior insensitivity to the initial conditions.
- GLVQ sensitivity to the number of clusters and the input space scale.
- Conditions for inconsistent and undesired behaviour of GLVQ.

Local Stochastic Learning rule

Formulation and demonstration of a Local Stochastic Learning Rule (LSLR).

- Variation of the SCL based on **Local Stochastic Competition** (LSC) decision for the encoding phase of VQ.
- Discussion of convergence of the LSC to the Nearest Neighbor assignment.
- A great potential for **speeding up** the codification process, with an affordable loss of codification quality.

Evolution Strategy I

Formulation of a Michigan-like Evolution Strategy for Clustering.

- Specific representation of the problem, where all the **population represents a complete solution**, each individual chromosome represents a component of the solution.
- Existence of a **global fitness** function for the entire population, on top of the individual fitness functions.
- **Mutation** is the **only operator** that introduces evolution-like variability.
- **Greedy selection operators** extract the next population from the pool of parents and offspring

Evolution Strategy II

Application of Evolution Strategies:

- Design of vector quantizers applied to the Color Quantization of image sequences.
- Design of VQ Bayesian Filters applied to noise removal and region segmentation of Magnetic Resonance Imaging data.

Occam filters to determine optimal codebook size

Application of an Occam filter approach to determine de optimal number of clusters in an application of the VQ Bayesian Filters.

- Occam filters use the fact that signal noise can be **cancelled out** by the signal loss produced by a lossy compression algorithm. Seeks the balance between the noise cancelation and the signal loss.
- In VQBF, the **compression control parameter is the codebook size**. Tuning this parameter to obtain noise cancellation is equivalent to determine the number of classes.
- Optimal codebook size is the **inflexion point of a rate-distortion curve**, computed using SOM.
- Test in a unsupervised segmentation process on a 3D MRI data.

Convergence analysis of the SOM in the GNC theory context

Discussion of the convergence of the Self Organizing Map and Neural Gas from the point of view of Graduated Non-Convexity methods.

1. Introduction and overview of Thesis contributions
   - Unsupervised learning
   - Supervised learning

Supervised learning

- Supervised SOM for color face localization
- Vector Quantization Bayesian Filtering for MRI tissue segmentation
- High Order Boltzmann Machines
- Relevance Dendritic Computing

Supervised SOM for color face localization

Development of a color based face localization system.

- Algorithm for face localization in image sequences:
  - First stage: **Localize** the head region based on the analysis of the **signatures** of temporal difference images.
  - Second stage: Provide **confirmation** of the head hypothesis through the **color analysis** of the head subimage.
- Color analysis as a Color Quantization process: color representatives are computed through a **supervised version of the SOM**.

VQBF for MRI tissue segmentation

Application of Vector Quantization Bayesian Filtering (VQBF) to the supervised segmentation in 3D Magnetic Resonance Images (MRI) of a region of interest (ROI).

- Hybrid System:
  - Filtering layer: VQBF performs an **unsupervised preprocessing** of the image to **reduce signal variability** across individual data volume. We use SOM to compute codebook required by VQBF.
  - Supervised Classification layer: Multi-Layer Perceptron is applied to VQBF-slices giving a **prediction** of the ROI. Ground truth is stablished over some selected slices where ROI is manually drawn.

High Order Boltzmann Machines

Generalization of the learning rule of the HOBM.

- Use of **categorical and continuous units** to reduce network complexity and speedup of the learning process.
- Use of **high order connection** to model high order interactions between variables instead of hidden units.
- Without hidden units, the Kullback-Leibler divergence is a convex function, therefore, **learning is robust** against bad initial conditions.

Relevance Dendritic Computing

Application of the Sparse Bayesian Learning to the Dendritic Computing to obtain Relevant Dendritic Computing parsimonious classifiers.

Presentation organization

Selected contributions:

- Non-Stationary Clustering
- Convergence analysis of the SOM and NG in the GNC theory context
- Relevance Dendritic Computing

Conclusions at the end of each sections

Section Contents

Introduction

- **Definition of problem**: Non-Stationary Clustering / Vector Quantization
- **Solution:** Competitive Neural Networks as Adaptive VQ algorithms
- **Application:** Color Quantization of image sequences

Non-Stationary Clustering I

Given a sequence of sample datasets of the time varying population

$$\mathbf{X}(\tau) = \{\mathbf{x}_1(\tau), \ldots, \mathbf{x}_N(\tau)\}\,;\ \tau = 0, 1, \ldots$$

obtain a sequence of disjoint partitions of the input space with corresponding induced sequence of sets of disjoint clusters on the sample datasets

$$P(\mathbf{X}(\tau)) = \{\mathcal{X}_1(\tau), \ldots, \mathcal{X}_M(\tau)\}$$

minimizing a criterium function along time

$$\xi = \sum_{\tau \geq 0} \xi(\tau)$$

Non-Stationary Clustering II

A solution

$$\mathbf{Y}(\tau) = \{\mathbf{y}_1(\tau), \ldots, \mathbf{y}_M(\tau)\}$$

where input space partitions are:

$$\mathbf{x}_j(\tau) \in \mathcal{X}_i(\tau) \Leftrightarrow i = \arg\min_{k=1,\ldots,M} \left\{ \|\mathbf{x}_j(\tau) - \mathbf{y}_k(\tau)\|^2 \right\}$$

At each time step, the criterium function is the within-cluster distortion:

$$\xi(\tau) = \sum_{j=1}^{N} \sum_{i=1}^{M} \|\mathbf{x}_j(\tau) - \mathbf{y}_i(\tau)\|^2 \delta_i(\mathbf{x}_j(\tau), \mathbf{Y}(\tau))$$

$$\delta_i(\mathbf{x}_j(\tau), \mathbf{Y}(\tau)) = \begin{cases} 1 & i = \arg\min_{k=1,\ldots,M} \left\{ \|\mathbf{x}_j(\tau) - \mathbf{y}_k(\tau)\|^2 \right\} \\ 0 & \text{otherwise} \end{cases}$$

Non-Stationary Clustering III

Search for a sequence of codebooks

$$\mathbf{Y}(\tau) = \{\mathbf{y}_1(\tau), \ldots, \mathbf{y}_M(\tau)\}$$

minimizing:

$$\min_{\{\mathbf{Y}(\tau)\}} \sum_{\tau \geq 0} \xi(\tau)$$

Frame-Based Adaptive Vector Quantization

To solve the previous stochastic minimization problem, we propose adaptive algorithms based in two simplifying assumptions:

1. **Time independence**: The minimization of the sequence of time dependent error function can be done independently at each time step.

$$\min_{\{\mathbf{Y}(\tau)\}} \sum_{\tau \geq 0} \xi(\tau) = \sum_{\tau \geq 0} \min_{\{\mathbf{Y}(\tau)\}} \xi(\tau)$$

2. **Bounded variation of the optimal codebook between successive time steps**. Then the set of representatives obtained after adaptation in a time step can be used as the initial conditions for the next time step.

$$\mathbf{Y}(\tau, 0) = \mathbf{Y}(\tau - 1, N)$$

FBAVQ procedure

Algorithm Frame-Based Adaptive VQ procedure

1. Assume an initial codebook $\mathbf{Y}(0)$, $\tau = 0$
2. Update the clock $\tau = \tau + 1$ and take the next sample of size N

$$\mathbf{X}(\tau) = \{\mathbf{x}_1(\tau), \ldots, \mathbf{x}_N(\tau)\}$$

3. Assume as the initial codebook the result of the adaptation at the previous time instant

$$\mathbf{Y}(\tau, 0) = \mathbf{Y}(\tau - 1, N)$$

4. Compute the sequence of adaptations of the codebook

$$\{\mathbf{Y}(\tau, t); t = 1, \ldots, N\}$$

   applying CNN (or ES) to $\mathbf{x}(t)$ extracted from $\mathbf{X}(\tau)$

5. Resume the process from step 2

Section

Competitive Neural Networks I

CNN algorithms are adaptive algorithms performing stochastic gradient descent (SGD) on a distortion-like criterium function to solve Clustering/VQ problems.

- Simple Competitive Learning (SCL) is the **basic** competitive learning rule derived from the minimization of the Euclidean distortion.
- Self-Organizing Maps (SOM), Fuzzy Learning VQ (FLVQ), Neural Gas (NG) and Soft Competition Scheme (SCS) are **instances of a general competitive learning rule**.
- These CNN are **robust initialization procedures** for the SCL when the goal is the minimization of the Euclidean distortion

Competitive Neural Networks II

General competitive learning rule for CNN :

$$\mathbf{y}_i(t+1) = \mathbf{y}_i(t) + \alpha_i(t)\ \Phi_i(\mathbf{x}(t), \mathbf{Y}(t), t)\ (\mathbf{x}(t) - \mathbf{y}_i(t))$$

where

- $\mathbf{x}(t) \in \mathbf{X}$: set of sample vectors,
- $y_i(t) \in \mathbf{Y}$: codebook,
- $\alpha_i(t)$: (local) learning rate,
- $\Phi(.)$: neighboring function.

Neighboring function

SCL: $$\Phi_i(\mathbf{x}, \mathbf{Y}, t) = \delta_i(\mathbf{x}, \mathbf{Y}) = \begin{cases} 1 & i = \arg \min_{k=1,\dots,M} \left\{ \|\mathbf{x} - \mathbf{y}_k\|^2 \right\} \\ 0 & \mathit{otherwise} \end{cases}$$

SOM: $$\Phi_i(\mathbf{x}, \mathbf{Y}, t) = \begin{cases} 1 & |w(\mathbf{x}, \mathbf{Y}) - i| \leq v(t) \\ 0 & \mathit{otherwise} \end{cases}$$

NG: $$\Phi_i(\mathbf{x}, \mathbf{Y}, t) = e^{\left(-k_i(\mathbf{x}, \mathbf{Y})/\lambda(t)\right)}$$

FLVQ: $$\Phi_i(\mathbf{x}, \mathbf{Y}, t) = (u_i(\mathbf{x}, \mathbf{Y}))^{m(t)} = \left(\sum_{k=1}^{M} \left(\frac{\|\mathbf{x}-\mathbf{y}_i\|^2}{\|\mathbf{x}-\mathbf{y}_k\|^2} \right)^{\frac{1}{m(t)-1}} \right)^{-m(t)}$$

SCS: $$\Phi_i(\mathbf{x}, \mathbf{Y}, t) = e^{-\frac{1}{2}\|\mathbf{x}-\mathbf{y}_i\|^2 \sigma(t)^{-2}} \left(\sum_{k=1}^{M} e^{-\frac{1}{2}\|\mathbf{x}-\mathbf{y}_k\|^2 \sigma(t)^{-2}} \right)^{-1}$$

Functional convergence I

The general learning rule perform a cascade of minimizations over a **sequence of objective functions**

$$\xi_{\Phi}(t) = \sum_{i=1}^{M} \int \int -\Phi_i(\mathbf{x}, \mathbf{Y}, t)(\mathbf{x} - \mathbf{y}_i)\, p(\mathbf{x})\, d\mathbf{y}_i d\mathbf{x}$$

The limit of this sequence of objective functions will be determined by the limit of their respective neighboring functions:

$$\lim_{t\to\infty} \Phi_i(\mathbf{x}, \mathbf{Y}, t) = \Phi_i^*(\mathbf{x}, \mathbf{Y}) \Longrightarrow \lim_{t\to\infty} \xi_{\Phi}(t) = \xi_{\Phi^*}$$

The application of the general learning rule is, therefore, a **minimization procedure** for the **limit objective function** ξ_{Φ^*}.

Functional convergence II

Functional convergence is controlled by the specific annealing control parameter of the neighboring function

$$\lim_{t\to\infty} \Phi_i\left(\mathbf{x}, \mathbf{Y}, t\right) = \delta_i\left(\mathbf{x}, \mathbf{Y}\right)$$

Functional convergence to the Euclidean distortion:

$$\lim_{t\to\infty} \xi_{\Phi(\cdot)}\left(t\right) \approx \xi_E^2$$

Learning realizations

Online and batch realizations:

- Input data set is presented **several times**
- Control parameters are modified after each input data **set** presentation

Adaptation of codebook:

- Online realization: after each input data **sample** presentation
- Batch realization: after presentation of the whole input data **set**

One-pass learning realizations

To approach real time performance we impose a *one-pass* adaptation at each time step, and small sample datasets.

One-pass online realization:

- Each input data is presented **at most once**
- Control parameters are modified after each input data **sample** presentation
- Adaptation of codebook after each input data **sample** presentation

Scheduling of learning rate

- The **local** learning rate schedule for each unit $i = 1, .., M$

$$\alpha_i (t) = 0.1 (1 - t_i / N)$$

- The **global** learning rate schedule has the same value for all units

$$\alpha (t) = \alpha_0 \left(\frac{\alpha_N}{\alpha_0} \right)^{\frac{t}{N}}$$

Scheduling of neighborhood size

The **rate of functional convergence** to the null neighborhood is denoted r.

$$\Phi_i(\mathbf{x}, \mathbf{Y}, t) = \delta_i(\mathbf{x}, \mathbf{Y}) \quad t \geq \frac{N}{r}$$

Scheduling of the neighborhood control parameter $\left(t < \frac{N}{r}\right)$:

$$\text{SOM:} \quad v(t) = \left\lceil (v_0 + 1)^{\left(1 - \frac{r}{N} t\right)} \right\rceil - 1$$

$$\text{NG:} \quad \lambda(t) = \lambda_0 \left(\frac{0.01}{\lambda_0}\right)^{\frac{r}{N} t}$$

$$\text{FLVQ:} \quad m(t) = m_0 \left(\frac{1.1}{m_0}\right)^{\frac{r}{N} t}$$

$$\text{SCS:} \quad \sigma(t) = (\sigma_0 + 1)^{\left(1 - \frac{r}{N} t\right)} - 1$$

Section

Color Quantization

Vector Quantization in a color space (RGB)

Encoder:

$$C: \quad f(x,y) \in [0,1]^3 \quad \rightarrow \quad f^M(x,y) \in \{1,\ldots,M\}$$

Decoder:

$$\hat{f}(x,y) = \mathbf{y}_i \Leftrightarrow f^M(x,y) = i.$$

Quality measure:

$$E = \sum_{x,y} \left\| \hat{f}(x,y) - f(x,y) \right\|^2$$

Non-Stationary Color Quantization I

Given an image sequence

$$\{f_\tau(x,y)\,;\ \tau = 1,2,\ldots\}$$

the searched partitions are the CQs of the images in the sequence

$$\left\{f_\tau^M(x,y)\,;\ \tau = 1,2,\ldots\right\}$$

and the infinite time horizon criterion function is the **accumulative CQ distortion**

$$E = \sum_{\tau\geq 0} E(\tau) = \sum_{\tau\geq 0}\sum_{x,y}\left\|\hat{f}_\tau(x,y) - f_\tau(x,y)\right\|^2$$

Non-Stationary Color Quantization II

Non-Stationary CQ looks for the optimal sequence of color palettes

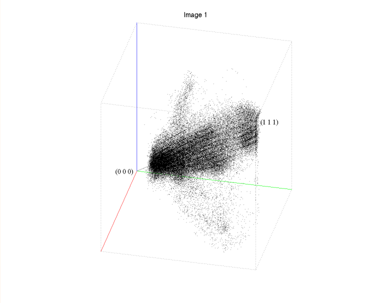
$$\mathbf{Y}(\tau) = \{\mathbf{y}_1(\tau), \ldots, \mathbf{y}_M(\tau)\}$$

minimizing the accumulated CQ distortion using AVQ algorithms

$$\min_{\{\mathbf{Y}(\tau)\}} \sum_{\tau \geq 0} E(\tau)$$

Section

Dataset example

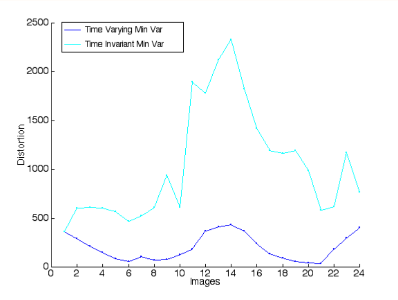


Benchmark results I

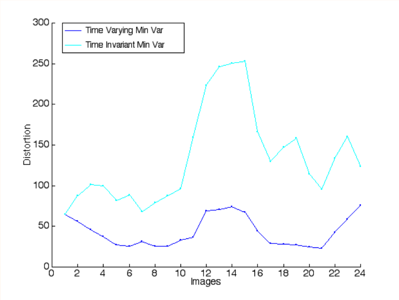
Benchmark non adaptive algorithm: Minimum Variance Heckbert's algorithm

- Application to the entire images in the sequence in two ways:
  - **Stationary assumption**: the color representatives obtained for the first image are used for the CQ of the remaining images in the sequence (*Time Invariant Min Var*)
  - **Non-stationary assumption**: applying it to each image independently (*Time Varying Min Var*)

Benchmark results II



16



256

Experiments

Experimental results report a sensitivity analysis to:

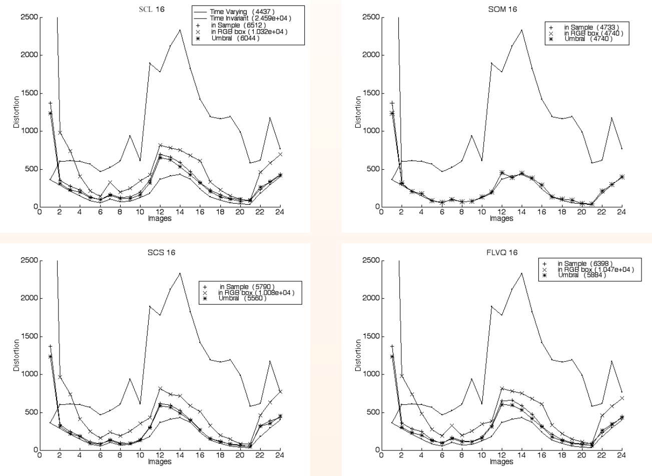
- Neighboring function control parameters
- Convergence ratio to SCL
- Initial conditions
- Codebook size
- Time subsampling
- Sample size
- Learning rate scheduling

Sensitivity to neighboring function control parameters

	SOM		FLVQ				SCS		
	$v_0 =$		$m_0 =$				$\sigma_0 =$		
	1	8	10	7	4	2	0.1	2	$\widehat{\sigma_{i,0}}$
$r = 1$	102.20	106.20	99.07	96.87	93.00	84.04	95.31	442.2	236.20
$r = 2$	72.08	71.49	91.47	90.94	86.96	84.74	85.85	149.2	85.24
$r = 4$	70.80	70.15	85.34	85.91	84.87	84.56	81.77	125.8	85.62
$r = 6$	72.70	**69.37**	85.39	85.31	85.48	**84.66**	**81.67**	113.7	84.43
$r = 8$	74.11	70.48	87.18	86.15	87.22	86.01	82.43	108.8	82.57

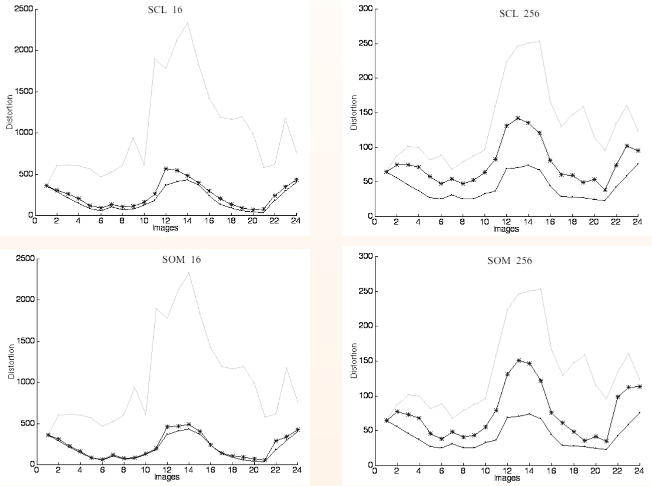
Accumulated distortion results

Robustness to initial conditions



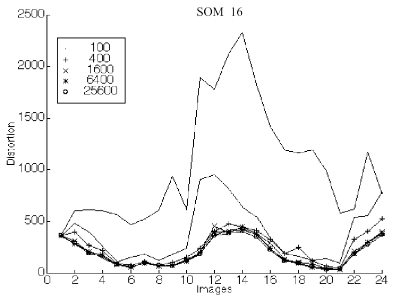
Per image distortion results

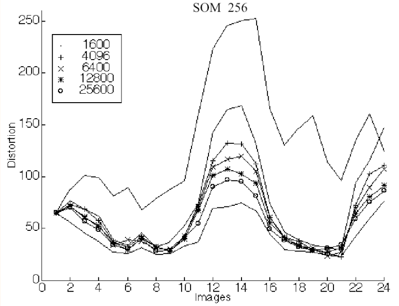
Sensitivity to codebook size



Per image distortion results (samples of 1600 pixels)

Sensitivity to sample size





Per image distortion results using sequences of samples of diverse size

Section

Conclusions

- Proposition of **Non-Stationary Clustering**, with a paradigm demonstration, the CQ of image sequences.
- Proposal of a general approach to their solution, the **Frame-Based Adaptive VQ**.
- Formulation of the most important CNN in a common framework as **instances of a general competitive learning rule**
- Implementation of **one-pass realizations** of learning schedules for CNN
- Exhaustive test of the CNN on the CQ of image sequences, proving the **robustness of the FBAVQ** performed by the CNN.

Section Contents

3. Convergence of the SOM from the point of view of GNC
    - Introduction
    - SOM and NG as GNC algorithms
    - Experimental results
    - Conclusions

Introduction I

Convergence of the SOM and NG is usually contemplated from the point of view of stochastic gradient descent (SGD) algorithms of an energy function.

- SGD algorithms have **slow convergence rate**, they are local minimization algorithms and thus very **dependent on the initial conditions.**
- However SOM and NG can be very **insensitive to the initial conditions.**

The empirical evidence leads us to propose the theory of **Graduated Non-Convexity** (GNC) methods as framework of the convergence analysis of the SOM and NG.

Introduction II

GNC algorithms try to solve the minimization of a non-convex objective function by the sequential search of the minima of a **one-parameter family of functionals**, which are morphed from a convex function up to the non-convex original function.

- In the SOM and NG the **neighborhood control parameters** may be understood as performing the role of graduating the non-convexity of the energy function minimized by the algorithm.
- The training of both the SOM and the NG can be seen as a **continuation process** of the minimum of a sequence of energy functions starting from a convex one and ending with the highly non-convex quantization distortion function.

Graduated Non-Convexity definitions

Let a sampled surface corrupted by additive noise,

$$M(x) = D(x) + N(x)$$

GNC approach seeks the **MAP estimate** of

$$p(R = D \,|M)$$

obtained **minimizing the energy**:

$$E[R] = -\log p(M\,|D = R) - \log p(D = R) = \boldsymbol{E_d[R] + E_s[R]}$$

GNC function general formulation:

$$E[R] = \sum_x (M(x) - R(x))^2 + f_\sigma(R)$$

GNC definitions

GNC method:

- Define a one-parameter family of functionals $E_\sigma [R]$, $\sigma \in [0, 1]$
  - **Initial functional** $E_{\sigma=1} [R]$ is **convex**
  - $E_\sigma [R]$ varies continuously as σ decreases from 1 to 0
  - Final functional $E_{\sigma=0} [R] = E [R]$ is the original function to be minimized
- Minimization of the whole sequence of functionals, using de optimal vector result of one minimization as the initial condition for the next.
- **No bifurcations in the continuation process** to track the global minimum of the initial functional to a global minimum of the target functional.

SOM functional

Original functional:

$$E_{SOM}(\mathbf{X},\mathbf{Y},v)=\sum_{i=1}^{M}\sum_{j=1}^{N}\Phi_i(\mathbf{x}_j,\mathbf{Y},v)\left\|\mathbf{x}_j-\mathbf{y}_i\right\|^2$$

$$\Phi_i(\mathbf{x},\mathbf{Y},v)=\begin{cases}1 & |w(\mathbf{x},\mathbf{Y})-i|\leq v\\ 0 & \text{otherwise}\end{cases}$$

Reorganized:

$$\begin{aligned}E_{SOM}(\mathbf{X},\mathbf{Y},v) &= \sum_{j=1}^{N}\left\|\mathbf{x}_j-\mathbf{y}_{w(\mathbf{x}_j)}\right\|^2\\ &+ \sum_{j=1}^{N}\sum_{\substack{i=1\\ i\neq w(\mathbf{x}_j)}}^{M}\Phi_i(\mathbf{x}_j,\mathbf{Y},v)\left\|\mathbf{x}_j-\mathbf{y}_i\right\|^2\end{aligned}$$

NG functional

Discretized functional:

$$E_{ng}\left(\mathbf{X},\mathbf{Y},\lambda\right)=\frac{1}{2C\left(\lambda\right)}\sum_{i=1}^{M}\sum_{j=1}^{N}\Phi_i\left(\mathbf{x},\mathbf{Y},\lambda\right)\left\|\mathbf{x}_j-\mathbf{y}_i\right\|^2,$$

Reorganized:

$$\begin{aligned}E_{ng}\left(\mathbf{X},\mathbf{Y},\lambda\right) &= \sum_{j=1}^{N}\left\|\mathbf{x}_j-\mathbf{y}_{w\left(\mathbf{x}_j\right)}\right\|^2 \\ &+ \sum_{j=1}^{N}\sum_{\substack{i=1\\ i\neq w\left(\mathbf{x}_j\right)}}^{M}\Phi_i\left(\mathbf{x}_j,\mathbf{Y},\lambda\right)\left\|\mathbf{x}_j-\mathbf{y}_i\right\|^2,\end{aligned}$$

Convexity of SOM and NG initial functionals

Conditions for convexity, regarding the neighborhood parameters

$$\nabla_i^2 E_{SOM}(\mathbf{X}, \mathbf{Y}, v) = \frac{1}{2} \sum_{j=1}^{N} \Phi_i(\mathbf{x}_j, \mathbf{Y}, v)$$

$$\forall \mathbf{X}; \forall i; \nabla_i^2 E_{SOM}(\mathbf{x}, \mathbf{Y}, v) > 0.$$

- For SOM, setting the neighborhood radius to encompass the whole network ensures that condition.
- NG neighborhood function is always positive, thus any non zero temperature will ensure theoretical convexity.

Continuation process of SOM and NG functionals

Absence of bifurcations in the continuation process:

- NG, successive functionals minimized are convex up to the limit of neighbouring control parameter.
- SOM, this is not demonstrated.

3 Convergence of the SOM from the point of view of GNC

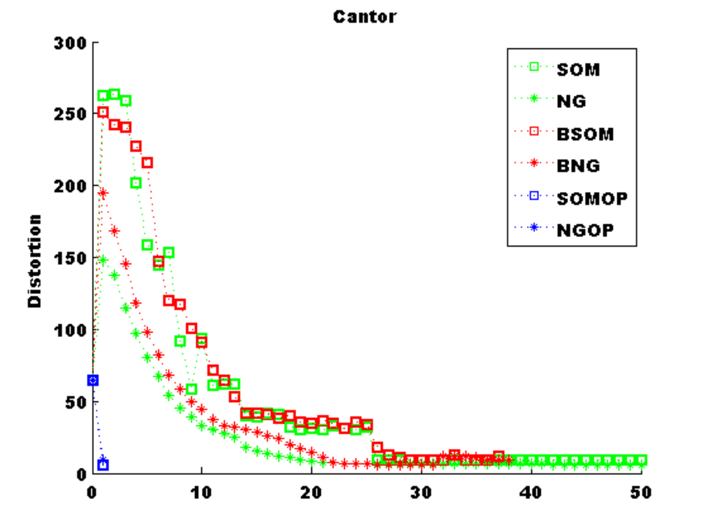
- Introduction
- SOM and NG as GNC algorithms
- **Experimental results**
- Conclusions

Experimental results

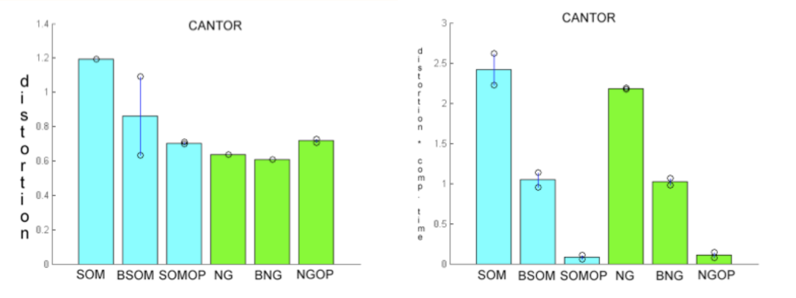
Experimental results to stress the idea that SOM and NG must be considered as a kind of GNC algorithms.

Comparison of conventional online and batch realizations versus one-pass realization.

Evolution of the distortion



Distortion and computation efficiency



SOM:	Online SOM	NG:	Online NG
BSOM:	Batch SOM	BNG:	Batch NG
SOMOP:	One-pass SOM	NGOP:	One-pass NG

3 Convergence of the SOM from the point of view of GNC

- Introduction
- SOM and NG as GNC algorithms
- Experimental results
- Conclusions

Conclusions

- One-pass realization can obtain **competitive performance** in terms of distortion, and much better in terms of computational efficiency.
- **Training** of the SOM and the NG can be seen as a **continuation of the minimization process** over a sequence of functionals tuned by the neighborhood control parameter.

Section Contents

Introduction

- **Motivation**: Improve generalization of Dendritic Computing (DC)
- **Framework**: Sparse Bayesian Learning –> Relevance Vector Machines (RVM)
- **Proposition**: Relevance Dendritic Computing (RDC) embedding DC in SBL

Dendritic Computing

Dendritic Computing $\subset$ Lattice Computing

Single Neuron Lattice model with DC :

- computes a **perfect approximation** to any data distribution
- suffers from **over-fitting problems**
- lack of **regularization**

Single Neuron Lattice model with DC

Given

$$\left(\mathbf{x}^{\xi}, c_{\xi}\right),\ \mathbf{x}^{\xi} \in \mathbb{R}^{d},\ c_{\xi} \in \{0,1\},\ \xi = 1, \dots, m$$

Response of the j-th dendrite:

$$\tau_j\left(\mathbf{x}^{\xi}\right) = p_j \bigwedge_{i \in I_j} \bigwedge_{l \in L_{ij}} (-1)^{1-l}\left(x_i^{\xi} + w_{ij}^{l}\right)$$

Complete **neuron activation**:

$$\tau\left(\mathbf{x}^{\xi}\right) = \bigwedge_{k=1}^{j} \tau_k\left(\mathbf{x}^{\xi}\right)$$

Output classification prediction:

$$\hat{c}^{\xi} = f\left(\tau\left(\mathbf{x}^{\xi}\right)\right)$$

Algorithm learning

The neuron activation function $\tau_j\left(\mathbf{x}^{\xi}\right)$ has **not derivatives defined**.

To develop learning algorithms is not possible to apply gradient based approaches.

An alternative is use **constructive learning algorithms**.

Algorithm learning

Algorithm Constructive algorithm pseudocode

1. Build a hyperbox enclosing all pattern samples of class 1.
2. Add dendrites to remove misclassified patterns of class 0 that fall inside this hyperbox:
   (a) Select at random one misclassified pattern.
   (b) Compute the minimum Chebyshev distance to a class 1 pattern.
   (c) Uses the patterns that are at this distance from the misclassified pattern to build a hyperbox that is removed from the initial hyperbox.
   (d) If one of the bounds is not defined, then the hyperbox spans to infinity in this dimension.

Sparse Bayesian Learning

Sparse Bayesian Learning is a general Bayesian framework for obtaining **sparse solutions** to regression and classification tasks.

A popular instance of this approach is the Relevance Vector Machine (RVM).

Model specification I

Given a binary classification problem, where $\{\mathbf{x}_n, t_n\}_{n=1}^{N}$ are the training input-target class pairs, $t_n \in \{0, 1\}$.

The **linear model function** is:

$$y(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{N} w_i K(\mathbf{x}, \mathbf{x}_i) + w_0$$

To obtain a prediction of the *a posteriori* probability of class 1

$$p(t = 1|\mathbf{x}) = \sigma(y(\mathbf{x}; \mathbf{w})), \quad \sigma(y) = \frac{1}{1+e^{-y}}$$

Model specification II

Dataset *likelihood*:

$$P\left(\mathbf{t}\left|\mathbf{w}\right.\right)$$

A priori distribution probability of the parameters w:

$$p\left(\mathbf{w}\left|\boldsymbol{\alpha}\right.\right)$$

A priori non-informative distribution probability of hyperparameters α:

$$p\left(\boldsymbol{\alpha}\right)$$

Bayesian inference

Estimation of the model parameters and hyperparameters corresponds to the **computation of the posterior distribution**:

$$p(\mathbf{w}, \boldsymbol{\alpha} | \mathbf{t}) = p(\mathbf{w} | \mathbf{t}, \boldsymbol{\alpha})\, p(\boldsymbol{\alpha} | \mathbf{t})$$

where

$$p(\mathbf{w} | \mathbf{t}, \boldsymbol{\alpha}) \propto p(\mathbf{t} | \mathbf{w})\, p(\mathbf{w} | \boldsymbol{\alpha})$$

$$p(\boldsymbol{\alpha} | \mathbf{t}) \propto p(\mathbf{t} | \boldsymbol{\alpha})\, p(\boldsymbol{\alpha})$$

Relevance Dendritic Computing I

Rewrite the dendritic neuron activation similar to a linear model function

$$\tau(\mathbf{x}) = \bigwedge_{n=1}^{N} \lambda_n(\mathbf{x}, \mathbf{x}_n)$$

where $\lambda_k(\mathbf{x}, \mathbf{x}_n)$ assumes the role of a **lattice-based kernel function**

$$\lambda_n(\mathbf{x}, \mathbf{x}_n) = \bigwedge_{i=1}^{d} (x_i - x_{n,i})\,\pi_{n,i}$$

The factor $\pi_{n,i} \in \{-1, 1, \infty\}$ models the contribution of the i-th component of the n-th sample training vector to the neural activation function

Relevance Dendritic Computing II

Gaussian prior distributions of the weights must be formulated over the **inverses of the weights**:

$$\mathbf{w} \equiv \left\{ \pi_{n,i}^{-1} \right\} = 1/\boldsymbol{\pi}$$

$$p\left(\mathbf{w} \,|\boldsymbol{\alpha}\right) = \prod_{n=1}^{N} \prod_{i=1}^{d} \mathcal{N}\left(\pi_{n,i}^{-1} \,\middle|\, 0, \alpha_{n,i}^{-1}\right)$$

RDC algorithm

Algorithm The Relevance Dendritic Computing

1. Initialize uniform weight prior hyperparameters $\alpha_{n,i} = \frac{1}{Nd}$.

2. Search for the most probable weights $\boldsymbol{\pi}_{MP}$ minimizing the log-posterior weight distribution

$$\begin{aligned} \log p(\boldsymbol{\pi} \,|\, \boldsymbol{\alpha}, \mathbf{t}) &\propto \log\{p(\mathbf{t} \,|\, \boldsymbol{\pi})\, p(\boldsymbol{\pi} \,|\, \boldsymbol{\alpha})\} \\ &= \sum_{n=1}^{N} [t_n \log y_n + (1 - t_n)\log(1 - y_n)] - \frac{1}{2}\mathbf{w}^T \mathbf{A} \mathbf{w} \end{aligned}$$

with $y_n = \sigma(\tau(\mathbf{x}_n))$, by Monte-Carlo Methods. Obtain relevant estimations of $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ from Monte-Carlo generated data.

3. Apply relevance updating

$$\alpha_{n,i}^{\text{new}} = \frac{\gamma_{n,i}}{\mu_{n,i}^2}, \qquad \text{with} \quad \gamma_{n,i} = 1 - \alpha_{ni}\Sigma_{ni,ni}$$

4. Remove irrelevant weights ($\alpha_{n,i} > \theta$) setting them to infinity

5. Test convergence. If not converged, repeat from step 2.

RDC algorithm

Algorithm The Relevance Dendritic Computing

1. Initialize uniform weight prior hyperparameters $\alpha_{n,i} = \frac{1}{Nd}$.

2. Search for the most probable weights $\boldsymbol{\pi}_{MP}$ minimizing the log-posterior weight distribution

$$\begin{aligned} \log p(\boldsymbol{\pi} \,|\boldsymbol{\alpha}, \mathbf{t}) &\propto \log \{p(\mathbf{t} \,|\boldsymbol{\pi})\, p(\boldsymbol{\pi} \,|\boldsymbol{\alpha})\} \\ &= \sum_{n=1}^{N} [t_n \log y_n + (1 - t_n) \log(1 - y_n)] - \frac{1}{2}\mathbf{w}^T \mathbf{A}\mathbf{w} \end{aligned}$$

with $y_n = \sigma(\tau(\mathbf{x}_n))$, by Monte-Carlo Methods. Obtain relevant estimations of $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ from Monte-Carlo generated data.

3. Apply relevance updating

$$\alpha_{n,i}^{\text{new}} = \frac{\gamma_{n,i}}{\mu_{n,i}^2}, \qquad \text{with} \quad \gamma_{n,i} = 1 - \alpha_{ni}\Sigma_{ni,ni}$$

4. Remove irrelevant weights ($\alpha_{n,i} > \theta$) setting them to infinity

5. Test convergence. If not converged, repeat from step 2.

RDC algorithm

Algorithm The Relevance Dendritic Computing

1. Initialize uniform weight prior hyperparameters $\alpha_{n,i} = \frac{1}{Nd}$.
2. Search for the most probable weights $\boldsymbol{\pi}_{MP}$ minimizing the log-posterior weight distribution

$$\log p(\boldsymbol{\pi} \,|\boldsymbol{\alpha}, \mathbf{t}) \propto \log \{p(\mathbf{t} \,|\boldsymbol{\pi}) \, p(\boldsymbol{\pi} \,|\boldsymbol{\alpha})\}$$
$$= \sum_{n=1}^{N} [t_n \log y_n + (1 - t_n) \log (1 - y_n)] - \frac{1}{2} \mathbf{w}^T \mathbf{A} \mathbf{w}$$

with $y_n = \sigma(\tau(\mathbf{x}_n))$, by Monte-Carlo Methods. Obtain relevant estimations of $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ from Monte-Carlo generated data.
3. Apply relevance updating

$$\alpha_{n,i}^{\text{new}} = \frac{\gamma_{n,i}}{\mu_{n,i}^2}, \qquad \text{with} \quad \gamma_{n,i} = 1 - \alpha_{ni} \Sigma_{ni,ni}$$

4. Remove irrelevant weights ($\alpha_{n,i} > \theta$) setting them to infinity
5. Test convergence. If not converged, repeat from step 2.

RDC algorithm

Algorithm The Relevance Dendritic Computing

1. Initialize uniform weight prior hyperparameters $\alpha_{n,i} = \frac{1}{Nd}$.
2. Search for the most probable weights $\boldsymbol{\pi}_{MP}$ minimizing the log-posterior weight distribution

$$\begin{aligned}\log p(\boldsymbol{\pi} \,|\boldsymbol{\alpha}, \mathbf{t}) &\propto \log\{p(\mathbf{t}\,|\boldsymbol{\pi})\, p(\boldsymbol{\pi}\,|\boldsymbol{\alpha})\} \\ &= \sum_{n=1}^{N} [t_n \log y_n + (1-t_n)\log(1-y_n)] - \frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w}\end{aligned}$$

   with $y_n = \sigma(\tau(\mathbf{x}_n))$, by Monte-Carlo Methods. Obtain relevant estimations of $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ from Monte-Carlo generated data.
3. Apply relevance updating

$$\alpha_{n,i}^{\text{new}} = \frac{\gamma_{n,i}}{\mu_{n,i}^2}, \qquad \text{with} \quad \gamma_{n,i} = 1 - \alpha_{ni}\Sigma_{ni,ni}$$

4. Remove irrelevant weights ($\alpha_{n,i} > \theta$) setting them to infinity
5. Test convergence. If not converged, repeat from step 2.

RDC algorithm

Algorithm The Relevance Dendritic Computing

1. Initialize uniform weight prior hyperparameters $\alpha_{n,i} = \frac{1}{Nd}$.
2. Search for the most probable weights $\boldsymbol{\pi}_{MP}$ minimizing the log-posterior weight distribution

$$\begin{aligned} \log p(\boldsymbol{\pi} \,|\, \boldsymbol{\alpha}, \mathbf{t}) &\propto \log \{ p(\mathbf{t} \,|\, \boldsymbol{\pi})\, p(\boldsymbol{\pi} \,|\, \boldsymbol{\alpha}) \} \\ &= \sum_{n=1}^{N} [t_n \log y_n + (1 - t_n) \log (1 - y_n)] - \frac{1}{2} \mathbf{w}^T \mathbf{A} \mathbf{w} \end{aligned}$$

   with $y_n = \sigma(\tau(\mathbf{x}_n))$, by Monte-Carlo Methods. Obtain relevant estimations of $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ from Monte-Carlo generated data.

3. Apply relevance updating

$$\alpha_{n,i}^{\text{new}} = \frac{\gamma_{n,i}}{\mu_{n,i}^2}, \qquad \text{with} \quad \gamma_{n,i} = 1 - \alpha_{ni} \Sigma_{ni,ni}$$

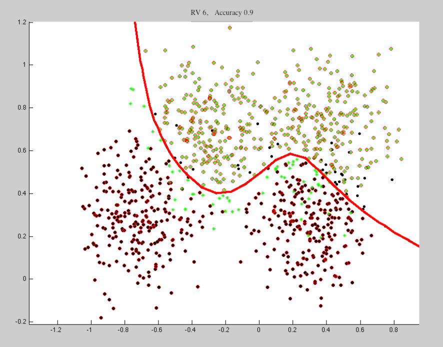
4. Remove irrelevant weights ($\alpha_{n,i} > \theta$) setting them to infinity
5. Test convergence. If not converged, repeat from step 2.

Experimental results I

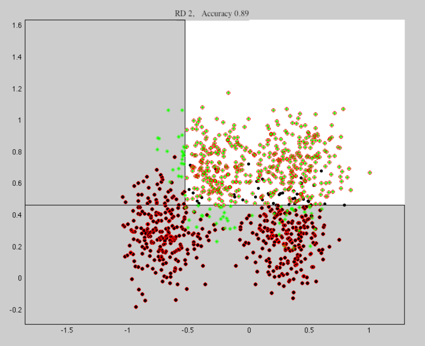
	accuracy	sensitivity	specificity	#rel. par.
RDC	0.89	0.86	0.92	2
RVM	0.90	0.87	0.92	6

Test results on the *Ripley* dataset (provided by Tipping)

Experimental results II



RVM (6)



RDC (2)

Conclusions

- Application of Sparse Bayesian Learning to a **new kind of classification systems** based on Dendritic Computing.
- **Lattice kernel** classification model.
- RDC finds **comparable results** with much **more parsimonious models** than RVM.

Section Contents

Summary

Several contributions to unsupervised and supervised learning:

- Proposition of a paradigm of Clustering problems, the **Non-Stationary Clustering**
- Presentation of a framework for the **analysis of the convergence** of SOM and NG.
- Application of the Sparse Bayesian Learning to the Dendritic Computing to obtain **Relevant Dendritic Computing** classifiers.

Further work: Improve RDC, trying different lattice-based kernel functions and other optimization techniques

Thanks for your attention¡