A Semi-Supervised Self-Organizing Map for Clustering and Classification

Pedro H. M. Braga and Hansenclever F. Bassani

Center of Informatics
Federal University of Pernambuco
Recife, Brasil
{phmb4, hfb}@cin.ufpe.br

July, 2018

Centro de Informática U·F·P·E

Universidade Federal de Pernambuco

Outline

1. Introduction
2. Background
3. Semi Supervised Self-Organizing Maps (SS-SOM)
4. Experimental Setup
5. Experimental Results and Analysis
6. Conclusion and Future Work

Introduction

- In recent years, research on Artificial Neural Networks with supervised learning algorithms has made great advances (Robotics, Genomics, and Natural Language Processing).

- These methods require a large amount of properly labeled data for training and it makes their use in many applications impractical.

- At the current stage of research, it is of great importance to put forward methods that can benefit both from the (frequently large amounts of) unlabeled data available as well as from the smaller amounts of labeled data, what would expand the current range of machine learning applications.

- Semi-Supervised Learning (SSL) methods are used to achieve this (Zhu and Ghahramani, 2002)

Introduction

- Semi-Supervised Learning is halfway between supervised and unsupervised learning, being applied for both classification and clustering tasks (Basu et al., 2002)

- Prototype-based methods such as K-Means and Self-Organizing Maps (SOM) are examples that have been successfully applied in this area.

- Kohonen proposed two very influential prototype-based methods:
  - SOM – Unsupervised (Kohonen, 1990)
  - Learning Vector Quantization (LVQ) – Supervised (Kohonen, 1995)
  - Why not a hybrid model?

Semi-Supervised Self-Organizing Map (SSSOM), which is an extension of Local Adaptive Receptive Field Dimension Selective Self-Organizing Map (LARFDSSOM – Bassani and Araujo, 2015), created by introducing important modifications to incorporate semi-supervised learning.

Background

- Unsupervised Learning:
  - Can address the problem imposed by the high-dimensional and unlabeled data
  - The general task of clustering involves **not only clustering** the data **but also identifying subsets** of the input dimensions which are **relevant** to characterize each cluster.
  - One way to achieve this is by **applying local relevances to the input dimensions (each cluster have their set of relevances)**. The identification of which dimension is relevant or not is an important feature when working with high-dimensional data.
  - In this context, subspace clustering methods have been proposed aiming to determine clusters in subspaces of the input dimensions of a given dataset
  - DSSOM, LARFDSSOM

- Supervised Learning:
  - Some methods for classification were proposed to deal with high-dimensional data: SVMs, MLP, GRLVQ etc.

Background

- Semi-Supervised Learning:
  - Semi-supervised K-means-based methods were very successful demonstrating their advantages over standard approaches. It can be viewed as an instance of the EM algorithm. The labeled data provides prior information about the conditional distribution of hidden category labels, working as a guide for the clustering process.

  - Label propagation (LP) is another approach for SSL. LP methods operate on proximity graphs or connected structures to spread and propagate information about the class to nearby nodes according to a similarity matrix. It is based on the assumption that nearby entities should belong to the same class, in contrast to far away entities (Zhu and Ghahramani, 2002)

  - A similar alternative to LP is called Label Spreading (LS). It differs from LP in modifications to the similarity matrix. LP uses the raw similarity matrix constructed from the data with no changes, whereas LS minimizes a loss function that has regularization properties allowing it to be often better regarding robustness to noise (Zhou et al., 2004)

SS-SOM

$$s(\mathbf{x}) = \arg\max_j [ac(D_\omega(\mathbf{x}, \mathbf{c}_j), \omega_j)], \quad (1)$$

$$ac(D_\omega(\mathbf{x}, \mathbf{c}_j), \omega_j) = \frac{\sum_{i=1}^{m} \omega_{ji}}{\sum_{i=1}^{m} \omega_{ji} + D_\omega(\mathbf{x}, \mathbf{c}_j) + \epsilon}, \quad (2)$$

$$D_\omega(\mathbf{x}, \mathbf{c}_j) = \sqrt{\sum_{i=1}^{m} \omega_{ji}(x_i - c_{ji})^2}. \quad (3)$$

1. Initialization
2. Organization
3. Convergence

Algorithm 1: Hybrid Mode

```
1  Initialize parameters a_t, lp, β, age_wins, e_b, e_n, eβ, minwd,
     t_max, push_rate, N_max;
2  Initialize the map with one node with c_j initialized at the first
     input pattern x_0, ω_j ← 1, δ_j ← 0, wins_j ← 0 and class_j ←
     noClass or class(x_0) if available;
3  Initialize the variable nwins ← 1;
4  for t ← 0 to t_max do
5      Choose a random input pattern x;
6      Compute the activation of all nodes (Eq. 2);
7      Find the winner s_1 with the highest activation (a_s) (Eq. 1);
8      if x has a label then
9          Run the SupervisedMode(x, s_1) (Alg. 3);
10     else
11         Run the UnsupervisedMode(x, s_1) (Alg. 2);
12     if nwins = age_wins then
13         Remove nodes with wins_j < lp × age_wins;
14         Update the connections of the remaining nodes (Eq. 7);
15         Reset the number of wins of the remaining nodes:
16         wins_j ← 0;
17         nwins ← 0;
18     nwins ← nwins + 1;
19 Run the Convergence Phase;
```

SS-SOM

- Other Common Operations
  - Nodes removal
  - Neighborhood Update

$$\text{nodes } i \text{ and } j \text{ are } \begin{cases} \text{connected,} & \text{if } (\, class(i) = class(j) \text{ or } \\ & class(i) = noClass \text{ or } \\ & class(j) = noClass \,) \\ & \text{and } \|\omega_i - \omega_j\| < e\sqrt{m} \\ \text{disconnected,} & \text{otherwise} \end{cases} \tag{7}$$

Algorithm 1: Hybrid Mode

1 Initialize parameters a_t, lp, β, age_wins, e_b, e_n, $\epsilon\beta$, $minwd$, t_{max}, $push\_rate$, N_{max};
2 Initialize the map with one node with $\mathbf{c}_j$ initialized at the first input pattern $\mathbf{x}_0$, $\boldsymbol{\omega}_j \leftarrow \mathbf{1}$, $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$, $\text{wins}_j \leftarrow 0$ and $\text{class}_j \leftarrow$ *noClass* or $class(\mathbf{x}_0)$ if available;
3 Initialize the variable nwins $\leftarrow 1$;
4 **for** $t \leftarrow 0$ ***to*** t_{max} **do**
5 Choose a random input pattern $\mathbf{x}$;
6 Compute the activation of all nodes (Eq. 2);
7 Find the winner s_1 with the highest activation (a_s) (Eq. 1);
8 **if** ***x** has a label* **then**
9 Run the SupervisedMode($\mathbf{x}$, s_1) (Alg. 3);
10 **else**
11 Run the UnsupervisedMode($\mathbf{x}$, s_1) (Alg. 2);
12 **if** *nwins* = *age_wins* **then**
13 Remove nodes with wins$_j$ < *lp* × *age_wins*;
14 Update the connections of the remaining nodes (Eq. 7);
15 Reset the number of wins of the remaining nodes:
16 $\text{wins}_j \leftarrow 0$;
17 *nwins* $\leftarrow 0$;
18 *nwins* $\leftarrow$ *nwins* + 1;
19 Run the Convergence Phase;

SS-SOM

- Other Common Operations

Algorithm 4: Node Update

Input: Node s, Learning Rate lr

1 **Function** `UpdateNode` (s, lr) **:**
2 Update the distance vectors $\boldsymbol{\delta}_s$ of s according lr (Eq. 5);
3 Update the relevance vectors $\boldsymbol{\omega}_s$ of s (Eq. 6);
4 Update the weight vectors $\mathbf{c}_s$ of s (Eq. 4);

$$\mathbf{c}_j(n+1) = \mathbf{c}_j(n) + e(\mathbf{x} - \mathbf{c}_j(n)), \tag{4}$$

$$\boldsymbol{\delta}_j(n+1) = (1 - e\beta)\boldsymbol{\delta}_j(n) + e\beta(|\mathbf{x} - \mathbf{c}_j(n)|), \tag{5}$$

$$\omega_{ji} = \begin{cases} \dfrac{1}{1 + \exp\left(\dfrac{\delta_{ji\text{mean}} - \delta_{ji}}{s(\delta_{ji\text{max}} - \delta_{ji\text{min}})}\right)} & \text{if } \delta_{ji\text{min}} \neq \delta_{ji\text{max}} \\ 1 & \text{otherwise,} \end{cases} \tag{6}$$

SS-SOM

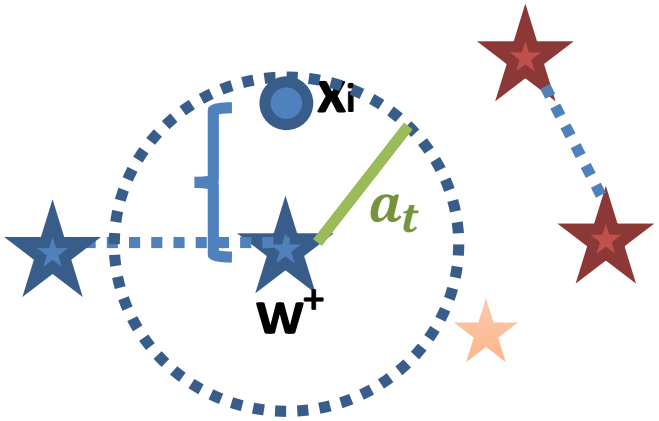
Algorithm 2: Unsupervised Mode

Input: Input pattern **x** and the first winner s_1;

if $a_{s_1} < a_t$ *and* $N < N_{max}$ **then**
  Create new node j and set: $\mathbf{c}_j \leftarrow \mathbf{x}$, $\boldsymbol{\omega}_j \leftarrow \mathbf{1}$, $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$, $\text{wins}_j \leftarrow 0$ and $\text{class}_j \leftarrow \textit{noClass}$;
  Connect j to the other nodes as per Eq. 7;
else
  Update the winner node and its neighbors: UpdateNode(s_1, e_b), UpdateNode($\textit{neighbors}(s_1)$, e_n) (Alg. 4);
  Set $\text{wins}_{s_1} \leftarrow \text{wins}_{s_1} + 1$;

SS-SOM

- When the winner node has the same class of a given sample:



Nodes of Class 1

Nodes of Class 2

Nodes of Undefined Class

Algorithm 3: Supervised Mode

and the first winner s_1;

1 **if** $class_{s_1} = class(x)$ **or** $class_{s_1} = noClass$ **then**

2 **if** $a_{s_1} < a_t$ *and* $N < N_{max}$ **then**

3 Create new node j and set: $\mathbf{c}_j \leftarrow \mathbf{x}$, $\boldsymbol{\omega}_j \leftarrow \mathbf{1}$, $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$, $wins_j \leftarrow 0$ and $class_j \leftarrow class(\mathbf{x})$;

4 Connect j to the other nodes as per Eq. 7;

5 **else if** $a_{s_1} \geq a_t$ **then**

6 Update the winner node and its neighbors: UpdateNode(s_1, e_b), UpdateNode($neighbors(s_1)$, e_n) (Alg. 4);

7 Set $class_{s_1} \leftarrow class(\mathbf{x})$;

8 Update s_1 connections as per Eq. 7;

9 Set $wins_{s_1} \leftarrow wins_{s_1} + 1$;

10 **else**

11 Try to find a new winner s_2 with *noClass* or the same class of $\mathbf{x}$ with activation $a_{s_2} \geq a_t$;

12 **if** s_2 *exists* **then**

13 Update the new winner node, its neighbors and the previous wrong winner: UpdateNode(s_2, e_b), UpdateNode($neighbors(s_2)$, e_n) and UpdateNode(s_1, -push_rate) (Alg. 4);

14 Set $wins_{s_2} \leftarrow wins_{s_2} + 1$;

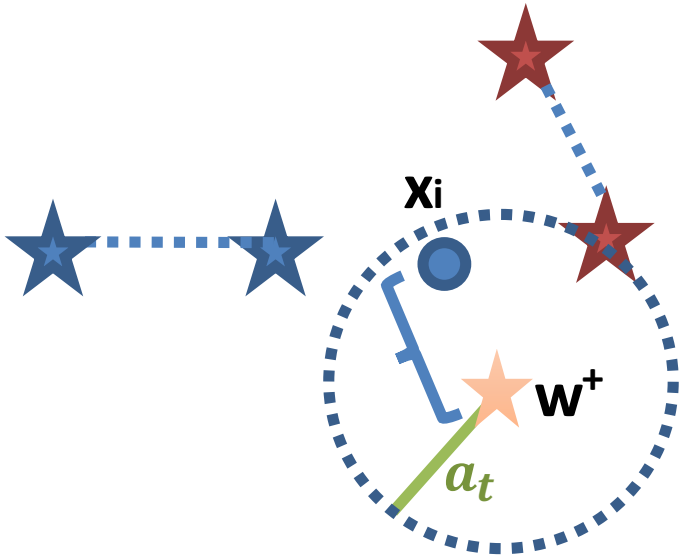
15 **else if** $N < N_{max}$ **then**

16 Create new node j and set: $\mathbf{c}_j \leftarrow \mathbf{x}$, $\boldsymbol{\omega}_j \leftarrow \mathbf{1}$, $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$, $wins_j \leftarrow 0$ and $class_j \leftarrow class(\mathbf{x})$;

17 Connect j to other nodes as per Eq. 7;

SS-SOM

- When the winner node class is not defined:



Nodes of Class 1

Nodes of Class 2

Nodes of Undefined Class

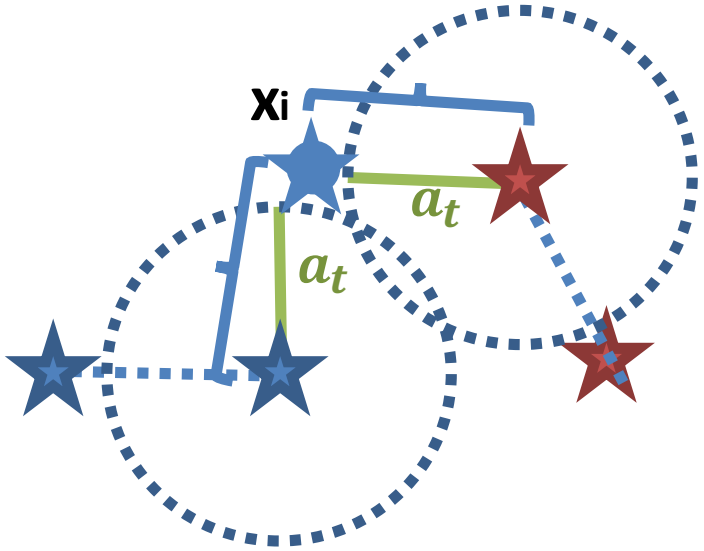
Algorithm 3: Supervised Mode

Input: Input pattern **x**

1 **if** $class_{s_1}$ = *class*(**x**) **or** $class_{s_1}$ = *noClass* **then**
2 **if** $a_{s_1} < a_t$ *and* $N < N_{max}$ **then**
3 Create new node j and set: $\mathbf{c}_j \leftarrow \mathbf{x}$, $\boldsymbol{\omega}_j \leftarrow \mathbf{1}$, $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$, $wins_j \leftarrow 0$ and $class_j \leftarrow$ *class*(**x**);
4 Connect j to the other nodes as per Eq. 7;
5 **else if** $a_{s_1} \geq a_t$ **then**
6 Update the winner node and its neighbors: UpdateNode(s_1, e_b), UpdateNode(*neighbors*(s_1), e_n) (Alg. 4);
7 Set $class_{s_1} \leftarrow$ *class*(**x**);
8 Update s_1 connections as per Eq. 7;
9 Set $wins_{s_1} \leftarrow wins_{s_1} + 1$;
10 **else**
11 Try to find a new winner s_2 with *noClass* or the same class of **x** with activation $a_{s_2} \geq a_t$;
12 **if** s_2 *exists* **then**
13 Update the new winner node, its neighbors and the previous wrong winner: UpdateNode(s_2, e_b), UpdateNode(*neighbors*(s_2), e_n) and UpdateNode(s_1, -push_rate) (Alg. 4);
14 Set $wins_{s_2} \leftarrow wins_{s_2} + 1$;
15 **else if** $N < N_{max}$ **then**
16 Create new node j and set: $\mathbf{c}_j \leftarrow \mathbf{x}$, $\boldsymbol{\omega}_j \leftarrow \mathbf{1}$, $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$, $wins_j \leftarrow 0$ and $class_j \leftarrow$ *class*(**x**);
17 Connect j to other nodes as per Eq. 7;

SS-SOM

- When a given labeled given sample has not a representative winner node:



Nodes of Class 1

Nodes of Class 2

Nodes of Undefined Class

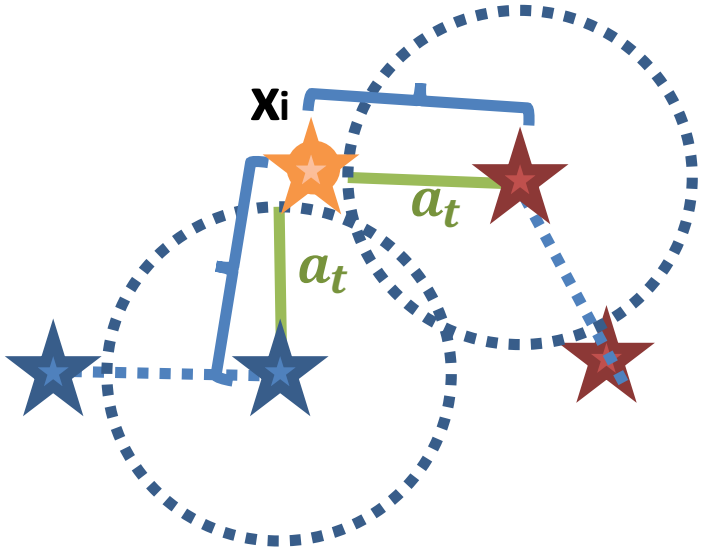
Algorithm 3: Supervised Mode

Input: Input pattern **x** and the first winner s_1;

1 **if** $class_{s_1} = class(\mathbf{x})$ **or** $class_{s_1} = noClass$ **then**
2 **if** $a_{s_1} < a_t$ *and* $N < N_{max}$ **then**
3 Create new node j and set: $\mathbf{c}_j \leftarrow \mathbf{x}$, $\boldsymbol{\omega}_j \leftarrow \mathbf{1}$, $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$, $wins_j \leftarrow 0$ and $class_j \leftarrow class(\mathbf{x})$;
4 Connect j to the other nodes as per Eq. 7;
5 **else if** $a_{s_1} \geq a_t$ **then**
6 Update the winner node and its neighbors: UpdateNode(s_1, e_b), UpdateNode($neighbors(s_1)$, e_n) (Alg. 4);
7 Set $class_{s_1} \leftarrow class(\mathbf{x})$;
8 Update s_1 connections as per Eq. 7;
9 Set $wins_{s_1} \leftarrow wins_{s_1} + 1$;
10 **else**
11 Try to find a new winner s_2 with *noClass* or the same class of **x** with activation $a_{s_2} \geq a_t$;
12 **if** s_2 *exists* **then**
13 Update the new winner node, its neighbors and the previous wrong winner: UpdateNode(s_2, e_b), UpdateNode($neighbors(s_2)$, e_n) and UpdateNode(s_1, -push_rate) (Alg. 4);
14 Set $wins_{s_2} \leftarrow wins_{s_2} + 1$;
15 **else if** $N < N_{max}$ **then**
16 Create new node j and set: $\mathbf{c}_j \leftarrow \mathbf{x}$, $\boldsymbol{\omega}_j \leftarrow \mathbf{1}$, $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$, $wins_j \leftarrow 0$ and $class_j \leftarrow class(\mathbf{x})$;
17 Connect j to other nodes as per Eq. 7;

SS-SOM

- When a given unlabeled given sample has not a representative winner node:



Nodes of Class 1

Nodes of Class 2

Nodes of Undefined Class

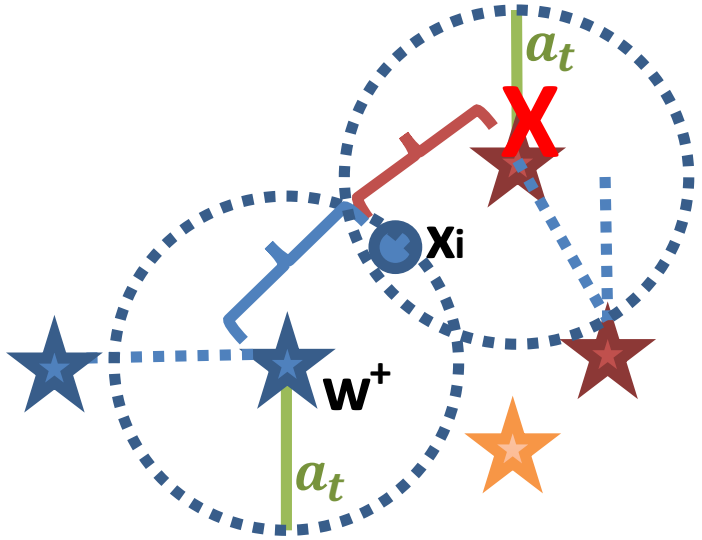
Algorithm 3: Supervised Mode

Input: Input pattern **x** and the first winner s_1;

1 **if** $class_{s_1}$ = *class*(**x**) ***or*** $class_{s_1}$ = *noClass* **then**
2 **if** $a_{s_1} < a_t$ *and* $N < N_{max}$ **then**
3 Create new node j and set: $\mathbf{c}_j \leftarrow \mathbf{x}$, $\boldsymbol{\omega}_j \leftarrow \mathbf{1}$, $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$, $wins_j \leftarrow 0$ and $class_j \leftarrow$ *class*(**x**);
4 Connect j to the other nodes as per Eq. 7;
5 **else if** $a_{s_1} \geq a_t$ **then**
6 Update the winner node and its neighbors: UpdateNode(s_1, e_b), UpdateNode(*neighbors*(s_1), e_n) (Alg. 4);
7 Set $class_{s_1} \leftarrow$ *class*(**x**);
8 Update s_1 connections as per Eq. 7;
9 Set $wins_{s_1} \leftarrow wins_{s_1} + 1$;
10 **else**
11 Try to find a new winner s_2 with *noClass* or the same class of **x** with activation $a_{s_2} \geq a_t$;
12 **if** s_2 *exists* **then**
13 Update the new winner node, its neighbors and the previous wrong winner: UpdateNode(s_2, e_b), UpdateNode(*neighbors*(s_2), e_n) and UpdateNode(s_1, -push_rate) (Alg. 4);
14 Set $wins_{s_2} \leftarrow wins_{s_2} + 1$;
15 **else if** $N < N_{max}$ **then**
16 Create new node j and set: $\mathbf{c}_j \leftarrow \mathbf{x}$, $\boldsymbol{\omega}_j \leftarrow \mathbf{1}$, $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$, $wins_j \leftarrow 0$ and $class_j \leftarrow$ *class*(**x**);
17 Connect j to other nodes as per Eq. 7;

SS-SOM

- When the winner node and a given labeled sample have different classes and a new valid winner is found:



Nodes of Class 1

Nodes of Class 2

Nodes of Undefined Class

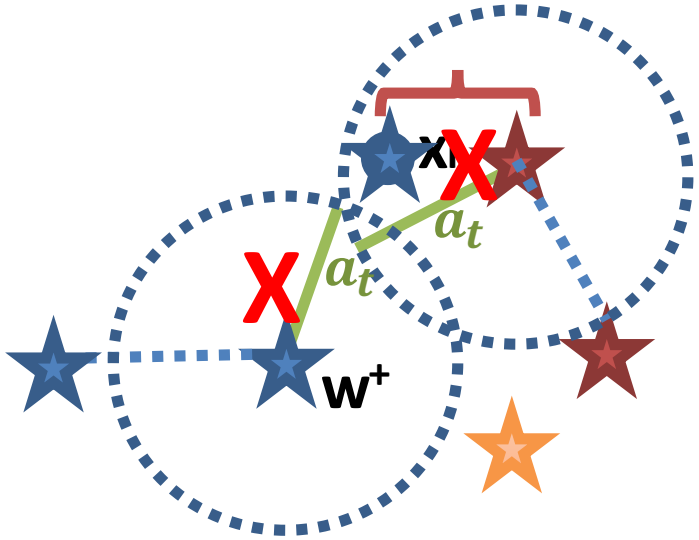
Algorithm 3: Supervised Mode

Input: Input pattern $\mathbf{x}$ and the first winner s_1;

if $class_{s_1} = class(\mathbf{x})$ **or** $class_{s_1} = noClass$ **then**
  if $a_{s_1} < a_t$ *and* $N < N_{max}$ **then**
    Create new node j and set: $\mathbf{c}_j \leftarrow \mathbf{x}$, $\boldsymbol{\omega}_j \leftarrow \mathbf{1}$, $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$, $wins_j \leftarrow 0$ and $class_j \leftarrow class(\mathbf{x})$;
    Connect j to the other nodes as per Eq. 7;
  else if $a_{s_1} \geq a_t$ **then**
    Update the winner node and its neighbors: UpdateNode(s_1, e_b), UpdateNode($neighbors(s_1)$, e_n) (Alg. 4);
    Set $class_{s_1} \leftarrow class(\mathbf{x})$;
    Update s_1 connections as per Eq. 7;
    Set $wins_{s_1} \leftarrow wins_{s_1} + 1$;
else
  Try to find a new winner s_2 with *noClass* or the same class of $\mathbf{x}$ with activation $a_{s_2} \geq a_t$;
  if s_2 *exists* **then**
    Update the new winner node, its neighbors and the previous wrong winner: UpdateNode(s_2, e_b), UpdateNode($neighbors(s_2)$, e_n) and UpdateNode(s_1, -push_rate) (Alg. 4);
    Set $wins_{s_2} \leftarrow wins_{s_2} + 1$;
  else if $N < N_{max}$ **then**
    Create new node j and set: $\mathbf{c}_j \leftarrow \mathbf{x}$, $\boldsymbol{\omega}_j \leftarrow \mathbf{1}$, $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$, $wins_j \leftarrow 0$ and $class_j \leftarrow class(\mathbf{x})$;
    Connect j to other nodes as per Eq. 7;

SS-SOM

- When the winner node and a given labeled sample have different classes, and there is no new valid winner:



Nodes of Class 1

Nodes of Class 2

Nodes of Undefined Class

Algorithm 3: Supervised Mode

Input: Input pattern $\mathbf{x}$ and the first winner s_1;

if $class_{s_1} = class(\mathbf{x})$ **or** $class_{s_1} = noClass$ **then**
  if $a_{s_1} < a_t$ *and* $N < N_{max}$ **then**
    Create new node j and set: $\mathbf{c}_j \leftarrow \mathbf{x}$, $\boldsymbol{\omega}_j \leftarrow \mathbf{1}$, $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$, $wins_j \leftarrow 0$ and $class_j \leftarrow class(\mathbf{x})$;
    Connect j to the other nodes as per Eq. 7;
  else if $a_{s_1} \geq a_t$ **then**
    Update the winner node and its neighbors: UpdateNode(s_1, e_b), UpdateNode($neighbors(s_1)$, e_n) (Alg. 4);
    Set $class_{s_1} \leftarrow class(\mathbf{x})$;
    Update s_1 connections as per Eq. 7;
    Set $wins_{s_1} \leftarrow wins_{s_1} + 1$;
else
  Try to find a new winner s_2 with *noClass* or the same class of $\mathbf{x}$ with activation $a_{s_2} \geq a_t$;
  if s_2 *exists* **then**
    Update the new winner node, its neighbors and the previous wrong winner: UpdateNode(s_2, e_b), UpdateNode($neighbors(s_2)$, e_n) and UpdateNode(s_1, -push_rate) (Alg. 4);
    Set $wins_{s_2} \leftarrow wins_{s_2} + 1$;
  else if $N < N_{max}$ **then**
    Create new node j and set: $\mathbf{c}_j \leftarrow \mathbf{x}$, $\boldsymbol{\omega}_j \leftarrow \mathbf{1}$, $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$, $wins_j \leftarrow 0$ and $class_j \leftarrow class(\mathbf{x})$;
    Connect j to other nodes as per Eq. 7;

Experimental Setup

- Compared it with:
  - 3 supervised methods: MLP, SVM and GRLVQ
  - 2 semi-supervised methods: Label Spreading and Label Propagation

- Real-word Datasets: Breast, Diabetes, Glass, Liver, Pendigits, Shape and Vowel

- For all methods, on each dataset, we used a cross-validation. Each method was trained and tested 500 times for each fold with different parameter values sampled according to a Latin Hypercube Sampling (LHS) (Helton et al., 2005)

- Mean and standard deviation of the best results were calculated separately for each dataset

Experimental Setup

- For studying the effects of the different levels of supervision, the semi-supervised methods were trained with the following percentages of labeled data: 1%, 5%, 10%, 25%, 50%, 75% and 100%.

- Classification and Clustering:
  - All nodes labeled: straightforward
  - Some nodes labeled: continue looking for another node with a defined class label, and an activation above the threshold
  - No nodes labeled: we can identify the clusters of the input patterns, but not their classes.

- For classification purposes, if available, we use the node class as the predicted class. Otherwise, it is straightforwardly considered as an error.

Experimental Setup

TABLE I
PARAMETER RANGES FOR GRLVQ

Parameters	min	max
Number of nodes	10	30
Positive learning rate	0.4	0.5
Negative learning rate	0.01	0.05
Weights learning rate	0.15	0.2
Learning Decay	0.000001	0.00002
Number of epochs	5000	10000

TABLE II
PARAMETER RANGES FOR SVM

Parameters	min	max
C	0.1	10
Kernel Function[1]	1	4
Degree of polynomial kernel function	3	5
Gamma of kernel functions 2, 3 and 4	0.1	1
Independent term in kernel functions 2 and 3	0.01	1

[1] 1: linear, 2: poly, 3: rbf and 4: sigmoid.

TABLE III
PARAMETER RANGES FOR MLP

Parameters	min	max
Number of neurons in each layer	1	100
Number of hidden layers	1	3
Learning rate	0.001	0.1
Momentum	0.85	0.95
Epochs	100	200
Optimizer[1]	1	3
Activation function[2]	1	3
Learning Decay[3]	1	3

[1] 1: lbfgs; 2: sgd; 3: adam; [2] 1: logistic; 2: tanh; 3: relu; [3] 1: constant; 2: invscaling; 3: adaptative.

TABLE IV
PARAMETER RANGES FOR LABEL SPREADING AND LABEL PROPAGATION

Parameters	min	max
Kernel Function[1]	1	2
γ (for RBF Kernel)	10	30
Number of Neighbors (for KNN Kernel)	1	100
α^*	0	1
Number of epochs	20	100

[1] 1: RBF and 2: KNN. * α is only used for label spreading.

TABLE V
PARAMETER RANGES FOR SS-SOM

Parameters	min	max
Activation threshold (a_t)	0.80	0.999
Lowest cluster percentage (lp)	0.001	0.01
Relevance rate (β)	0.001	0.5
Max competitions (age_wins)	$1 \times S^*$	$100 \times S^*$
Winner learning rate (e_b)	0.001	0.2
Wrong winner learning rate (e_w)	$0.01 \times e_b$	$1 \times e_b$
Neighbors learning rate (e_n)	$0.002 \times e_b$	$1 \times e_b$
Relevance smoothness ($\epsilon\beta$)	0.01	0.1
Connection threshold ($minwd$)	0	0.5
Number of epochs ($epochs$)	1	100

* S is the number of input patterns in the dataset.

Experimental Results

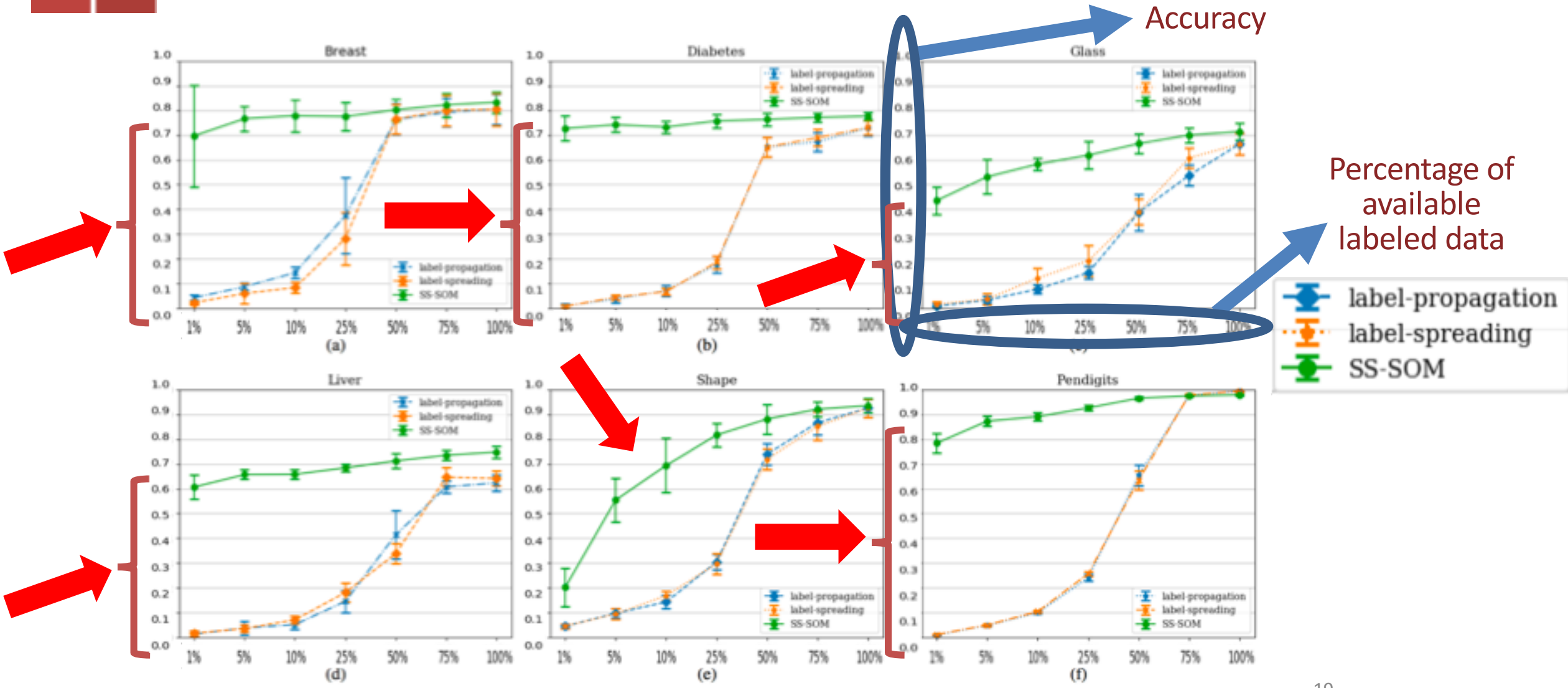


Fig. 1. Best mean accuracy and standard deviation as function of the percentage of supervision for each dataset

Experimental Results

TABLE VI
ACCURACY RESULTS FOR REAL-WORLD DATASETS WITH 100% OF THE LABELED DATA

Accuracy	Breast	Diabetes	Glass	Liver	Pendigits	Shape	Vowel
SS-SOM	**0.832 (0.044)**	**0.776 (0.016)**	**0.714 (0.033)**	**0.748 (0.025)**	0.978 (0.004)	**0.935 (0.029)**	0.876 (0.017)
Label Propagation	0.805 (0.063)	0.730 (0.031)	0.663 (0.044)	0.623 (0.036)	**0.994 (0.003)**	0.925 (0.036)	**0.948 (0.012)**
Label Spreading	0.805 (0.066)	0.729 (0.031)	0.663 (0.044)	0.640 (0.031)	**0.994 (0.003)**	0.925 (0.036)	**0.948 (0.012)**
MLP	**0.854 (0.032)**	**0.791 (0.017)**	**0.746 (0.031)**	**0.766 (0.031)**	0.993 (0.001)	0.923 (0.034)	0.874 (0.033)
SVM	0.850 (0.037)	0.788 (0.020)	0.718 (0.028)	0.746 (0.054)	**0.997 (0.001)**	**0.931 (0.030)**	**0.909 (0.022)**
GRLVQ	0.830 (0.049)	0.772 (0.020)	0.676 (0.027)	0.699 (0.022)	0.915 (0.004)	0.823 (0.061)	0.515 (0.027)

In bold, the best results for each dataset on each category: semi-supervised and supervised methods. The underlined results indicate the global best.

≈ 2%

Conclusion and Future Work

- This article presented an approach for classification and clustering with semi-supervised learning
- The behavior of SS-SOM was shown to have led to significant improvements in classification results for small amounts of labeled data
- Showed its robustness under this condition, being better than other semi-supervised, achieving impressive results even with only 1% of labeled data
- Using 100% of the labels, SS-SOM showed results better than or at least close to the best found in comparison with others supervised and semi-supervised methods, even with it not being the objective of this work

Conclusion and Future Work

- When there is no labeled sample available, SS-SOM works exactly as LARFDSSOM, inheriting its characteristics and performance. However, when labeled samples are given, the results can be even better.

- The self-organizing process is run for a number of epochs sampled from LHS, which is usually greater than the necessary to converge, even at the defined interval. An adequate stop criterion is an object of study for future versions in order to reduce the training time.

- With a small change, SS-SOM could also incorporate reinforcement learning, being, thus, capable of switching between three different learning approaches, to exploits several forms of information available.

Thank you all!

Pedro Braga

phmb4@cin.ufpe.br

UNIVERSIDADE
FEDERAL
DE PERNAMBUCO