remote sensing

MDPI

Article

Supervised and Semi-Supervised Self-Organizing Maps for Regression and Classification Focusing on Hyperspectral Data †

Felix M. Riese, Sina Keller * and Stefan Hinz

Institute of Photogrammetry and Remote Sensing, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany; felix.riese@kit.edu (F.M.R.); stefan.hinz@kit.edu (S.H.)

* Correspondence: sina.keller@kit.edu; Tel.: +49-721-608-41815

† This paper is an extended version of our paper published in arXiv:1903.11114.

Received: 29 November 2019; Accepted: 16 December 2019; Published: 18 December 2019

Abstract: Machine learning approaches are valuable methods in hyperspectral remote sensing, especially for the classification of land cover or for the regression of physical parameters. While the recording of hyperspectral data has become affordable with innovative technologies, the acquisition of reference data (ground truth) has remained expensive and time-consuming. There is a need for methodological approaches that can handle datasets with significantly more hyperspectral input data than reference data. We introduce the Supervised Self-organizing Maps (SuSi) framework, which can perform unsupervised, supervised and semi-supervised classification as well as regression on high-dimensional data. The methodology of the SuSi framework is presented and compared to other frameworks. Its different parts are evaluated on two hyperspectral datasets. The results of the evaluations can be summarized in four major findings: (1) The supervised and semi-Supervised Self-organizing Maps (SOM) outperform random forest in the regression of soil moisture. (2) In the classification of land cover, the supervised and semi-supervised SOM reveal great potential. (3) The unsupervised SOM is a valuable tool to understand the data. (4) The SuSi framework is versatile, flexible, and easy to use. The SuSi framework is provided as an open-source Python package on GitHub.

Keywords: machine learning; unsupervised learning; supervised learning; semi-supervised learning; land cover; soil moisture

1. Introduction

Hyperspectral remote sensing data have been used in many applications of environmental research during the last few decades [1,2]. Since hyperspectral sensors are mounted nowadays on unmanned aerial vehicles (UAVs) and satellites are also installed on handheld devices, the hyperspectral data acquisition has become more affordable [3]. Therefore, hyperspectral data are used increasingly to monitor physical parameters such as soil moisture or land cover over larger areas [4,5].

The increasing computing power and the availability of big data have enabled the development of artificial neural networks (ANNs) as standard tools for regression and classification tasks [6,7]. In machine learning (ML) included among other ANNs, different models are used depending on the tasks and the available data [8]. Currently, feed-forward neural networks and convolutional neural networks (CNN) are some of the most common types of ANN in remote sensing research. With several hidden layers, so-called deep ANNs are able to learn autonomously lower-level and higher-level features. These kinds of ANNs require a lot of training data. In environmental applications, we often

deal with limited reference data (ground truth) since its acquisition is extremely time-consuming and costly. The reference data can be limited, for example, in their amount, in their accuracy and quality as well as being limited in time. Therefore, ML approaches that depend on large training datasets such as deep ANNs are often not applicable.

Self-organizing maps (SOMs) are a type of ANN that can handle datasets with few references. SOMs are mostly used for unsupervised clustering and visualization; only a few studies have focused on supervised SOMs up to now (see Section 2.1). The SOM was introduced in [9–12]. It is a shallow ANN architecture consisting of an input layer and a two-dimensional (2D) grid as the output layer. The latter is fully connected to the input layer. The neurons on the output grid are interconnected to each other through a neighborhood relationship. Changes on the weights of one output neuron also affect the neurons in its neighborhood. This unique characteristic decreases overfitting of the training datasets. Furthermore, the 2D output grid visualizes the results of the SOM comprehensibly. This plain visualization does not exist in the majority of ANNs.

In this paper, we introduce the Supervised Self-organizing maps (SuSi) framework for regression and classification with hyperspectral data. Figure 1 illustrates the schema of the SuSi framework. The SuSi framework combines unsupervised, supervised, and semi-supervised learning for regression and classification. In [13], the supervised regression part of the SuSi framework has been initially introduced. It has been applied for the estimation of soil moisture [4] and the estimation of water quality parameters [14]. This initial SOM implementation has served as the basis for the SuSi framework that we have further improved in terms of its regression performance. In addition, we introduce novel parts of the SuSi framework in this contribution such as the supervised classification as well as the semi-supervised learning. To evaluate these novel parts, we apply the SuSi framework on two exemplary hyperspectral datasets and compare the respective regression and classification results to a random forest (RF) model. The actual framework is available as an open-source Python package on GitHub [15]. This ensures the maintainability and the opportunity for future upgrades by the authors as well as any member of the community.

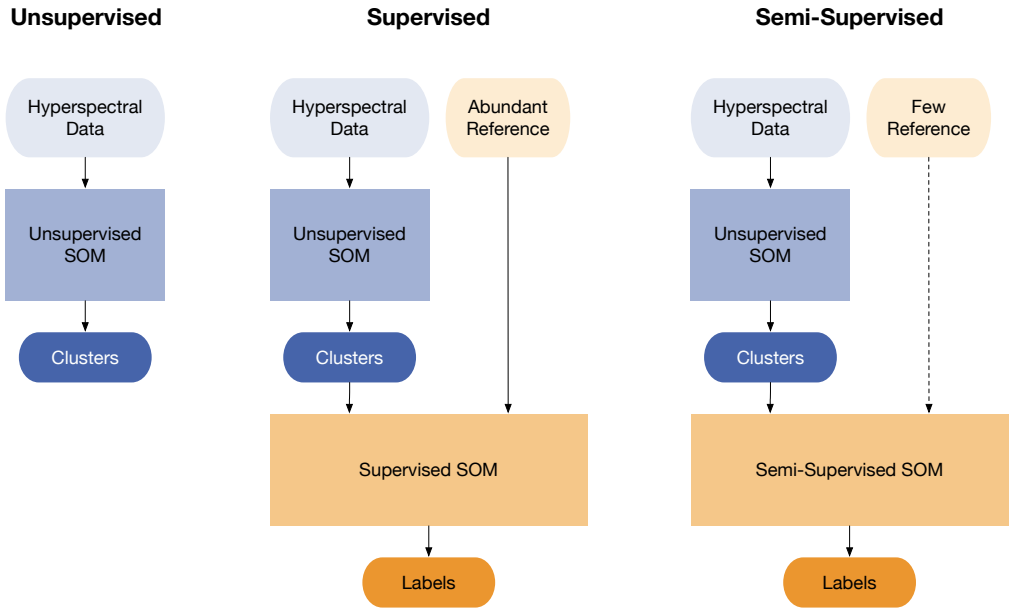


Figure 1. Schema of the Supervised Self-organizing Maps (SuSi) framework consisting of the unsupervised self-organizing map (SOM, blue) and the (semi-)supervised SOM (orange). Reduced reference data are illustrated with a dashed line between the references and the semi-supervised SOM.

The main contributions regarding methodological developments and software of this paper are summarized as follows:

- the mathematical foundations of the unsupervised and supervised SOM in the SuSi framework,
- a detailed evaluation of the supervised regression SOM on the example of soil moisture regression and of the supervised regression SOM on the example of land cover classification, and
- the introduction of the semi-supervised SOM followed by an evaluation of the semi-supervised regression and classification capabilities of the SuSi framework.

2. Materials and Methods

In this section, we give an overview of the research background, introduce the datasets used for the evaluation and describe in detail the mathematical foundations of the unsupervised, supervised, and semi-supervised SOM in the SuSi framework.

2.1. Research Background

In the following subsection, we give a brief overview of various SOM applications in different fields of research. Most SOMs are applied in unsupervised learning like clustering, visualization, and dimensionality reduction [16]. A comprehensive overview of SOMs as unsupervised learners and their applications in the research field of water resources is presented by [17]. SOMs are also used in the context of maritime environment research [18]. In addition, a main application of SOMs is clustering data [19] and cluster-wise regression [20].

SOMs can be combined with other ML techniques. For example, the output of the unsupervised SOM is used by Hsu et al. [21] as input for a support vector regressor to forecast stock prices. The clustering of the unsupervised SOM significantly improved the forecast. The authors in [22] present the combination of SOMs with linear regression in hydrology. The authors point out the value of the visualization capabilities of the unsupervised SOM. SOMs can also be used for data fusion, e.g., for plant disease detection [23]. Hagenbuchner and Tsoi [24] add majority voting to SOMs for the application as a supervised classifier. The combination of SOMs and nearest-neighbor classification is shown by Ji [25] for the classification of land use based on Landsat satellite data. Additional combinations of unsupervised SOMs and supervised algorithms used for classification are presented in [26–29]. One example for the application of SOMs to solve nonlinear regression tasks is presented by Hecht et al. [30] in the field of robotics. SOMs are also applied in the supervised regression of physical parameters in environmental research [4,13,14].

Semi-supervised learning (SSL) learns from labeled as well as from unlabeled data. An introduction to SSL is given in [31]. In [32], an unsupervised SOM and a supervised ANN are combined for the semi-supervised classification in software fault detection. The authors point out the generalization abilities of their proposed framework. An approach entirely based on SOMs is presented in [33]. A semi-supervised classification SOM (SS-SOM) is introduced and applied on several real-world datasets. The implementation of the SS-SOM is provided and can therefore be compared to existing packages.

Python and R are widely used programming language in the context of ML applications. Programming frameworks like scikit-learn [34] in Python have simplified the application of existing ML techniques considerably. While in the programming language R the *kohonen* package [35] provides a standardized framework for SOMs, several minor SOM packages exist in Python addressing only specific ML tasks.

In the following, we compare the introduced SuSi framework with existing Software packages in Python, R, and C. In our comparison, we include the Python packages *SOMPY* [36], *SimpSOM* [37], *MiniSom* [38], *TensorFlow SOM* [39], PyMVPA [40], NeuPy [41], the R *kohonen* package [35] and the C tool *SS-SOM* [33]. All entitled packages are freely available, regularly maintained (in 2019), and include unsupervised clustering. For example, the packages abhinavralhan/kohonen-maps and `lmjohns3/kohonen` are not regularly maintained and therefore left out in this study. The packages are analyzed on the basis of the following criteria:

- Syntax: Is the syntax simple to use, e.g., in the style of scikit-learn [34]?
- Documentation: Is the documentation provided in the form of a comprehensive paper or an online documentation?
- Code: Is the code well documented and structured? Are code examples and example plots available?
- ML: Is unsupervised clustering, supervised regression, or supervised classification included in the package?
- Installation: Is the installation simple for the user, e.g., with PyPI?
- Programming language: In what language can the package be used?

The analysis is performed in favor of the packages, meaning that in cases where a criterion is only partially satisfied, it is still considered as satisfied. Table 1 summarizes the analysis results. Thus far, no supervised SOM package for Python is available that matches the defined criteria and requirements (see Table 1).

Table 1. Analysis of the SuSi package with existing self-organizing map (SOM) packages. All packages are provided for the programming language Python (except kohonen and SS-SOM), they are freely available and regularly maintained (in 2019).

Package **Reference**	**SuSi**	**SOMPY** [36]	**SimpSOM** [37]	**MiniSom** [38]	**TensorFlow SOM** [39]	**PyMVPA** [40]	**NeuPy** [41]	**kohonen** [35]	**SS-SOM** [33]
Simple syntax	✓		✓	✓		✓	✓		
Useful documentation	✓			✓		✓	✓		✓
Well documented code	✓		✓	✓	✓		✓		
Unsupervised clustering	✓	✓	✓	✓	✓	✓	✓	✓	✓
Supervised regression	✓							✓	
Supervised classification	✓							✓	✓
Semi-supervised regression	✓								
Semi-supervised classification	✓								✓
Simple installation	✓		✓	✓		✓	✓	✓	
Python version	3	2	3	3	3	2	3	R	C

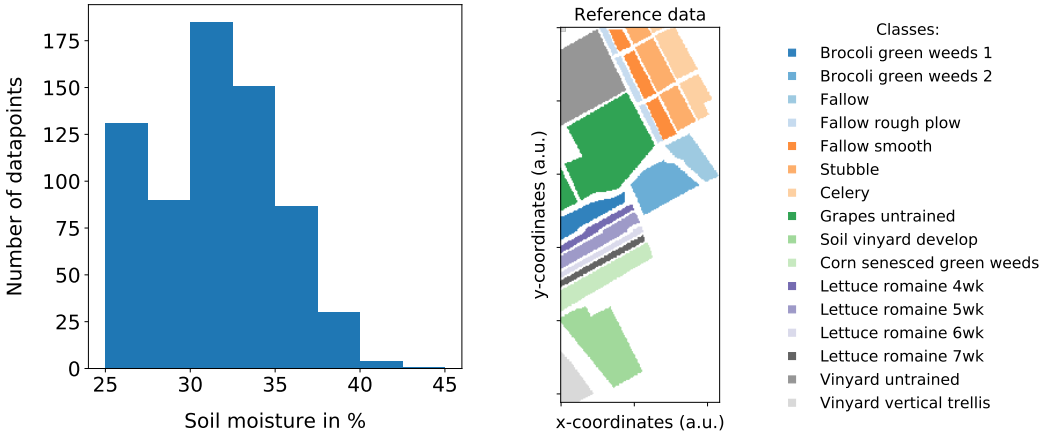
2.2. Hyperspectral Datasets for the Evaluation

The different evaluations of the SuSi framework performed in this paper are based on two existing hyperspectral datasets, one dataset with continuous labels (regression) and one dataset with discrete labels (classification). Both datasets are selected based on their free availability [42,43] and their high dimensionality of the input data.

A soil moisture dataset measured during a field campaign and published in [42] is used for the regression evaluations in this paper. The dataset consists of 679 datapoints collected by a Cubert UHD 285 camera. One datapoint consists of 125 bands between 450 nm to 950 nm. A soil moisture sensor measured the reference values in a range of 25% and 42% soil moisture. In Figure 2a, the distribution of the soil moisture measurements is shown.

The Salinas valley dataset [43] for classification is a freely available land cover dataset consisting of 512 × 217 pixels collected by the 224-band Airborne Visible Infrared Imaging Spectrometer (AVIRIS) in California, United States of America. The spatial resolution of this dataset is 3.7 m. Of the 224 bands in the range of 400 nm to 2500 nm, the 20 water absorption bands are discarded, namely the bands 108 to 112, 154 to 167 and 224. The dataset consists of 54,129 datapoints with reference data of 16 classes including several vegetation classes and bare soil. In Figure 2b, an overview of the dataset is illustrated. Compared to the regression dataset, this dataset is significantly larger.

For some studies, the hyperspectral data are normalized per feature to improve the performance of some estimators which are based on distance metrics. The mean of the respective dataset is therefore set to zero and the standard deviation to one.



(**a**) (**b**)

Figure 2. (**a**) histogram of the soil moisture distribution of the applied dataset [42]; (**b**) overview of the Salinas Airborne Visible Infrared Imaging Spectrometer (AVIRIS) dataset with 16 classes and a spatial resolution of 3.7 m [43]. The coordinates are not available and therefore illustrated in arbitrary units (a.u.).

2.3. Unsupervised SOM for Clustering

In the following subsection, we describe the architecture and mathematics of the unsupervised part of the SuSi framework. Further insights about the theory of SOMs can be found for example in [12]. The 2D grid of a SOM, the map, can be implemented in different topologies. In this paper, we use the simplest topology: the 2D rectangular grid consisting of $n_{\text{row}} \times n_{\text{column}}$ nodes. This grid is fully connected to the input layer. This means that each node is connected to all n input features via n weights. The variable naming conventions of the SuSi framework are given in Table A1. The training process of the unsupervised SOM is illustrated in Figure 3 and consists of the following steps:

1. Initialize the SOM.
2. Get random input datapoint.
3. Find best matching unit (BMU) (see Section 2.3.1).
4. Calculate learning rate (see Section 2.3.2) and neighborhood function (see Section 2.3.3).
5. Calculate neighborhood distance weight matrix (see Section 2.3.4).
6. Modify SOM weight matrix (see Section 2.3.5).
7. Repeat from step 2 until the maximum number of iterations is reached.

The initialization approach of a SOM mainly affects the speed of the training. We initialize the SOM weights of the SuSi framework randomly at this stage of development. Attik et al. [44], Akinduko et al. [45], for example, proposed more sophisticated initialization approaches like applying a principal component analysis. We describe the training of an unsupervised SOM in Section 2.3.1. For every part of the unsupervised SOM, several possible formulas exist. We provide the formulas most relevant to this paper in the following subsections while providing alternative formulas in Appendix B.

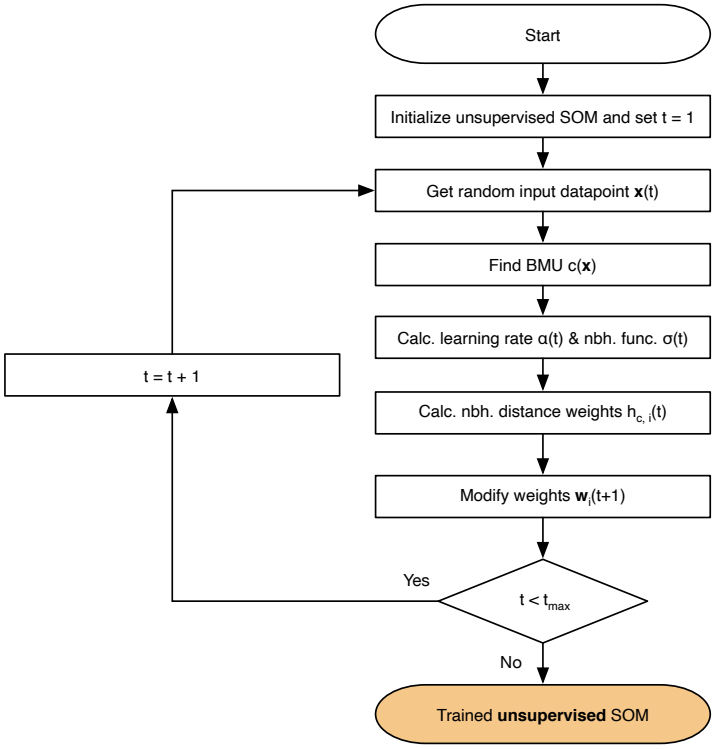


Figure 3. Flowchart of the unsupervised SOM algorithm resulting in the trained unsupervised SOM (orange).

2.3.1. Finding the Best Matching Unit

During the search for the best matching unit (BMU), the current input datapoint is compared to all n-dimensional weight vectors on the SOM grid. The SOM node that is the closest one to the input node according to the chosen distance metric is the BMU. Several distance metrics can be applied. The most common distance metric is the *Euclidean distance* defined as

$$d(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{\sum_{i=1}^{n} (x_i - x_i')^2}, \tag{1}$$

with a dimension n of the vectors $\boldsymbol{x}, \boldsymbol{x}'$. Further SOM distance metrics are provided in Appendix B.1. The Euclidean distance is the default distance metric of the SuSi framework. Note that, in Section 2.3.4, the Euclidean distance is applied on the 2D coordinates of the SOM nodes and not on the n-dimensional weight vectors.

2.3.2. Learning Rate

Decreasing learning rates are often implemented in ANNs for a faster convergence and to prevent oscillations. The learning rate α of the SOM training is a function that decreases from a value α_0 with an increasing number of iterations. In the following, we present the default implementation of the learning rate in the SuSi framework. Alternative learning rate functions are summarized in Appendix B.2. In Figure 4, examples for the behavior of the functions are plotted. The default implementation the learning rate is taken from [46] and includes a start value for the learning rate as well as an end value α_{end}:

$$\alpha(t) = \alpha_0 \cdot \left(\frac{\alpha_{\text{end}}}{\alpha_0} \right)^{t/t_{\max}}. \tag{2}$$

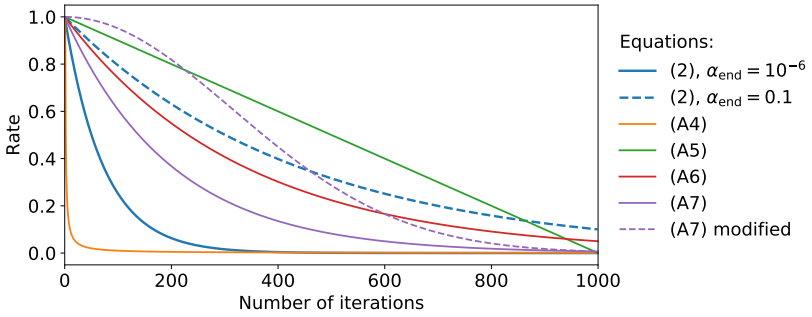


Figure 4. Comparison of different choices for the functional behavior of decreasing learning rates on the number of iterations t ($t_{max} = 1000, \alpha_0 = 1$).

2.3.3. Neighborhood Function

Similar to the learning rate, the neighborhood function is monotonically decreasing. In the following, we provide the default neighborhood function of the SuSi framework. Additional neighborhood functions can be found in Appendix B.3. The neighborhood function in [47] is defined, with σ_0 as initial value of the neighborhood function and is equivalent to Equation (A5), as

$$\sigma(t) = \sigma_0 \cdot \left(1 - \frac{t}{t_{max}}\right). \tag{3}$$

2.3.4. Neighborhood Distance Weight

The neighborhood distance weight is a function of the number of iterations and the distance $d(c, i)$ between the BMU c and every other node i on the SOM grid. The distance $d(c, i)$ between the BMU c and node i is defined as the Euclidean distance (see Equation (1)) on the 2D map grid. Note that this distance $d(c, i)$ is not the distance between the n-dimensional weights. In the following, we provide the default neighborhood distance weight formula of the SuSi framework. Another formula is provided in Appendix B.4. Matsushita and Nishio [47] proposed a *Pseudo-Gaussian* neighborhood distance weight. The weight between the BMU c and the node i on the SOM grid is defined as

$$h_{c,i}(t) = \exp\left(-\frac{d^2}{2 \cdot \sigma(t)^2}\right), \tag{4}$$

with the neighborhood function from Equation (3) and the Euclidean distance $d(c, i)$ on the SOM grid. The implications of the chosen neighborhood distance weight definitions on the SOM are investigated in e.g., [48,49].

2.3.5. Adapting Weights

The two most used approaches to adapt the SOM weights are the *online* and the *batch* mode. All weights of the SOM are adapted based on the learning rate and the neighborhood distance weight. The *online mode* is described in detail in [12]. After each iteration, all weights of the SOM are adapted to the current datapoint $\boldsymbol{x}(t)$ as follows:

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + \alpha(t) \cdot h_{c,i}(t) \cdot (\boldsymbol{x}(t) - \boldsymbol{w}_i(t)), \tag{5}$$

with neighborhood function $h_{c,i}(t)$, learning rate $\alpha(t)$, and weight vector $\boldsymbol{w}_i(t)$ of node i at iteration t. The *online* mode is the default mode of the SuSi framework. An implementation of the *batch mode* is described in Appendix B.5.

2.3.6. Trained Unsupervised SOM

After reaching the maximum number of iterations $t_{\max}$, the unsupervised SOM is fully trained. The unsupervised SOM can now calculate the BMU for every input datapoint, regardless if the datapoint is also part of the training dataset. This supervised approaches in Sections 2.4 and 2.5 are based on the trained unsupervised SOM. Since the number of nodes on the SOM grid can be larger than the number of datapoints in a specific dataset, several nodes can exist that are not linked to a datapoint of that respective dataset.

2.4. Supervised SOM for Classification and Regression

To apply the SuSi framework for solving supervised regression or classification tasks, we attach a second SOM to the unsupervised SOM. Figure 5 illustrates the flowchart of the second, supervised SOM. The two SOMs differ with respect to the dimension of the weights as well as their training. The weights of the unsupervised SOM are of the same dimension as the input data. Thus, adapting these weights often changes the BMU for each input datapoint. In contrast, the weights of the supervised SOM have the same dimension as the target variable of the respective task.

One has to distinguish between two cases: regression and classification. In the regression case, the weights of the supervised SOM are one-dimensional and contain a continuous number. In the classification case, the weights of the supervised SOM contain a class. By combining the unsupervised and the supervised SOM, the former is used to select the BMU for each datapoint while the latter links the selected BMU to a specific estimation. In the following, we describe the different implementations for regression and classification tasks.

2.4.1. Implementation of the Regression SOM

An initial implementation of the regression SOM is described in [13] using the example of the soil moisture regression based on hyperspectral data. The training of the regression SOM proceeds analogous to the unsupervised SOM: first, the SOM is initialized randomly. Again, it iterates randomly through the dataset (see Step 1). In each iteration, the BMU is found for the current datapoint based on the trained unsupervised SOM (see Steps 2 and 3). The BMUs do not change for the datapoints during the training since the unsupervised SOM is fully trained. Then, the neighborhood function, the learning rate, and the neighborhood distance weight matrix are calculated similarly to the algorithm of the unsupervised SOM (see Steps 4 and 5). Finally, the weights are adapted to the label $y(t)$ of the input datapoint $\boldsymbol{x}(t)$ (see Step 6).

In the case of the regression SOM, the label is a continuous value and the weights of the regression SOM can be modified similarly to the process described in Section 2.3.5. After the training (and in the case of a one-dimensional, target variable), the regression SOM consists of a map with a continuous distribution of the regression target variable. To apply the trained regression SOM to a new dataset, the BMUs needs to be found by the unsupervised SOM. For each datapoint in the new dataset, the estimated output value of the SuSi framework is the weight of the found BMU on the regression SOM. The regression SOM is illustrated in Figure 5.

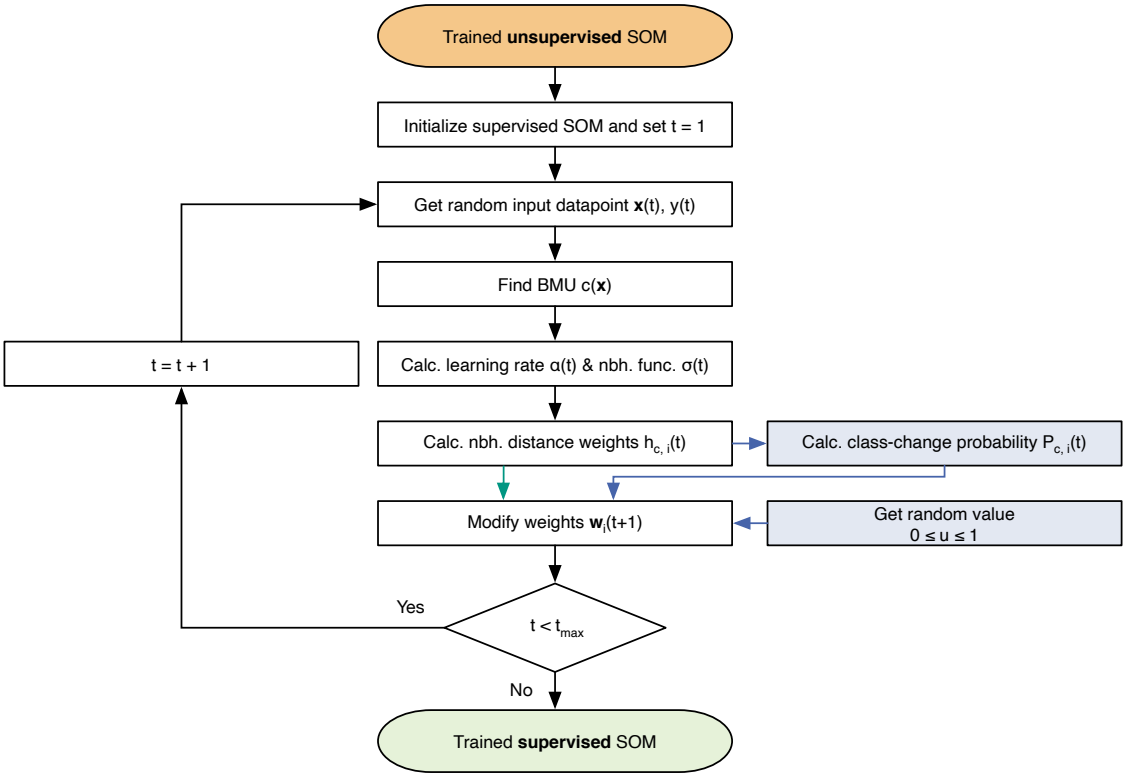


Figure 5. Flowchart of the algorithms for the regression SOM (black and cyan) and the classification SOM (black and blue). The "Trained unsupervised SOM" (orange) is the result of the unsupervised SOM algorithm illustrated in Figure 3.

2.4.2. Implementation of the Classification SOM

In the case of a classification task, the labels are discrete. In contrast to the commonly used majority voting approach (see [24]), we have implemented a training process similar to the adaptation approach of the unsupervised SOM (see Section 2.3.5):

1. Initialize the classification SOM.
2. Get random input datapoint with label.
3. Find BMU based on trained unsupervised SOM.
4. Calculate learning rate and neighborhood function.
5. Calculate neighborhood distance weight.
6. Calculate class-change probability matrix.
7. Modify classification SOM weight matrix.
8. Repeat from step 2 until the maximum number of iterations is reached.

The classification SOM is illustrated in Figure 5. The initialization in step 1 contains a simple majority vote: each node is assigned to the class representing the majority of datapoints allocated to the respective node. Steps 2 to 5 are implemented similarly to the regression SOM in Section 2.4.1. To modify the discrete weights of the classification SOM, we introduce the class-change probability $P_{c,i}(t)$ in step 6. In the regression SOM, the SOM nodes around the BMU are adapted to the current datapoint with a certain probability depending on the learning rate and the neighborhood distance weight. In the following, we explain our proposed modification in the case of discrete models.

For datasets with imbalanced class distributions, meaning datasets with significantly different number of datapoints per class, we provide the possibility to re-weight the dataset. The optional class weight is defined as

$$w_{\text{class}(j)} = \begin{cases} N/(n_{\text{classes}} \cdot N_j), & \text{if class weighting,} \\ 1, & \text{otherwise,} \end{cases} \tag{6}$$

with the number of datapoints N, the number of datapoints N_j of class j, and the number of classes n_{classes}. Similar to Equation (5), we define a term that affects the modification of the SOM weights. Since the modifications need to be discrete, we rely on probabilities. The probability for a class change of a node i with BMU $c(\boldsymbol{x})$ of the datapoint $\boldsymbol{x}(t)$ with label $y(t)$ is defined as

$$P_{c,i}(t) = w_{y(t)} \cdot \alpha(t) \cdot h_{c,i}(t), \tag{7}$$

with the class weight $w_{y(t)}$ (see Equation (6)), the learning rate $\alpha(t)$ (see Section 2.3.2), and the neighborhood distance weight $h_{c,i}(t)$ (see Section 2.3.4). To decide if a node changes its assigned class, a binary decision rule is created based on this probability. A simple threshold of e.g., 0.5 would lead to a static SOM after a certain number of iterations. Therefore, we include randomization into the decision process. For every node in every iteration, a random number $u_i(t)$ is generated that is uniformly distributed between 0 and 1. The modification of the weights is then implemented based on the class change probability $P_{c,i}(t)$ defined in Equation (7) as follows:

$$w_i(t+1) = \begin{cases} y(t), & \text{if } u_i(t) < P_{c,i}(t), \\ w_i(t), & \text{otherwise,} \end{cases} \tag{8}$$

with the label $y(t)$ linked the to datapoint $\boldsymbol{x}(t)$ of the current iteration t. After the maximum number of iterations is reached, the classification SOM is fully trained. Then, every node on the SOM grid is assigned to one class of the dataset. To apply the classification SOM on unknown data, the BMU needs to be found for each datapoint with the unsupervised SOM. This process is similar to the process of the trained regression SOM. The estimation of the classification SOM for this datapoint is equivalent to the weight of the neuron in the classification SOM at the position of the selected BMU.

2.5. Semi-Supervised SOM for Regression and Classification

In hyperspectral remote sensing, the acquisition of the reference data like soil moisture measurements is often costly and time-expensive. With hyperspectral sensors on UAVs or satellites, the hyperspectral input data can be recorded over large areas. This kind of data acquisition can lead to a dataset with significantly more hyperspectral input data than reference data. To use the possibly larger input dataset, *semi-supervised learning* can be applied that can learn from labeled as well as from unlabeled data. In comparison, supervised learning approaches (see Section 2.4) can only learn from labeled data.

The semi-supervised SOM is a combination of an unsupervised part (see Section 2.3) and a supervised part (see Section 2.4). The unsupervised part is applied on the full hyperspectral input dataset since it does not rely on labels. The supervised part afterwards is applied on the smaller, labeled part of the training dataset. To decrease the importance of unlabeled training datapoints, a sample weight is implemented which gives more weight to labeled datapoints than to unlabeled datapoints. This weighting is used for semi-supervised regression as well as for semi-supervised classification. A class weighting, as it is implemented in the supervised classification SOM (see Section 2.4.2), can only improve the semi-supervised classification performance, if the labeled datapoints are not balanced over the different classes.

3. Results and Discussion

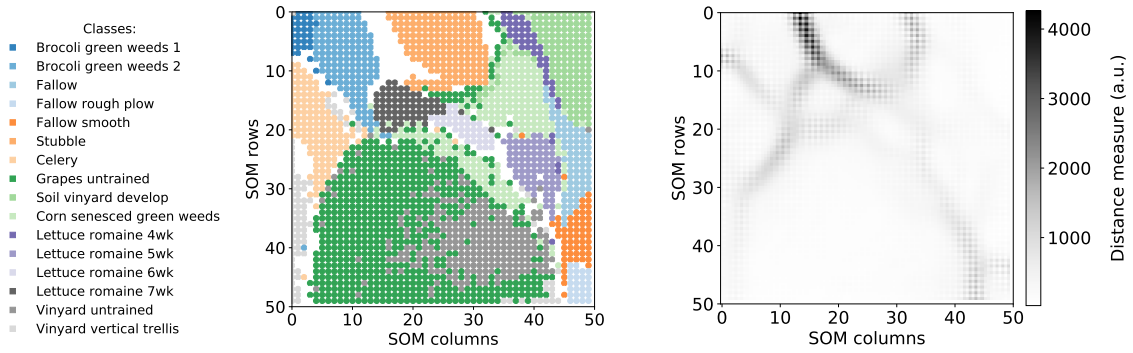
In this section, we present exemplary applications of unsupervised clustering, supervised, and semi-supervised soil moisture regression as well as land cover classification. Subsequently, we briefly discuss the respective applications of the SuSi framework and the derived results.

3.1. Unsupervised Clustering and Visualization

We exemplary apply the unsupervised SOM on the Salinas dataset described in Section 2.2. The hyperparameters for the unsupervised SOM are provided in Table A2. After the training of the unsupervised SOM, the dominant (most prevalent) class for every single SOM node is shown in Figure 6a. The SOM performs the clustering unsupervised, solely relying on the input data. The labels of the input data are only added in the unsupervised part due to visualization purposes. Similar classes form clusters of different shapes. With this visualization, information about similarities between the classes can be gained as well as possible mislabeled datapoints and classes [16]. Classes such as *celery* and *soil vinyard develop* form more isolated clusters while a class like *corn senesced green weeds* is more spread over the SOM grid.

In Figure 6b, the u-matrix is shown. The u-matrix contains the vector norms between the neighboring SOM nodes. Larger values imply larger spectral differences. The given u-matrix shows that the spectra of classes *stubble* and *celery* are significantly separated from the rest classes.

The SOM grid distribution per class is shown in Figure 7. For each class, mostly one or two soft clusters are visible. Differences of overlapping classes like *grapes untrained* and *vinyard untrained* are more visible in this visualization than in the dominant-class plot in Figure 6a.



(**a**) (**b**)

Figure 6. (**a**) dominant classes per SOM node on the grid of the unsupervised SOM. The color white is used if no datapoint is mapped to a SOM node; (**b**) u-matrix of the trained unsupervised SOM.

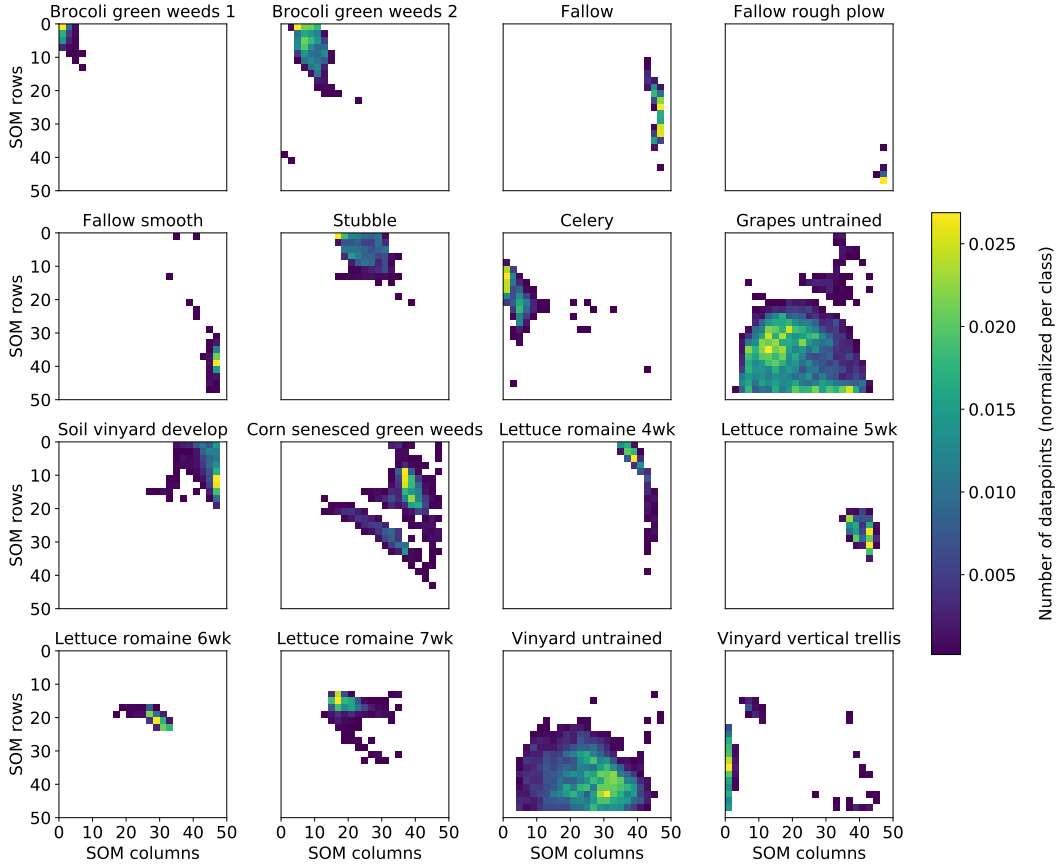


Figure 7. 2D histograms for every class of the Salinas dataset on the 2D grid of the unsupervised SOM.

3.2. Supervised Regression of Soil Moisture

The performance of the regression SOM is evaluated based on the soil moisture dataset. A similar evaluation is published in [13] with an initial version of this regression SOM. For the validation of the regression performance and the generalization capabilities of the SOM, the dataset is randomly divided into a training and a test subset in the ratio 1 : 1. The training of the regression SOM is performed on the training subset and the evaluation is performed on the test subset. The dataset is normalized per feature for the SOM regression; the RF regression does not improve with normalization.

The hyperparameters of the regression SOM are set after a minor optimization and are provided in Table A2. These hyperparameters can be further optimized depending on the applied dataset and underlying task. The results of the regression SOM are compared to the results of a RF regressor [50]. The RF regressor is set up with 10,000 estimators and the scikit-learn default hyperparameters (see [34]). For the evaluation, we choose the coefficient of determination R^2. To ensure generalization, the evaluation is repeated for five different random seeds which are applied on the SOM, the RF as well as the dataset split. The final results are provided as the mean R^2 its standard deviation.

The regression SOM achieves $R^2 = 95.3 \pm 0.8\%$ on the test subset. This score implies that the regression SOM is able to generalize very well on this dataset. Interestingly, the result for the training subset is only marginally better with $R^2_{\text{train}} = 97.7 \pm 0.6\%$. In comparison, the RF regressor results in $R^2 = 93.0 \pm 2.2\%$ on the test dataset and $R^2_{\text{train}} = 99.1 \pm 0.2\%$ on the training dataset. The SOM outperforms the RF while showing less overtraining, meaning a smaller difference between R^2 and R^2_{train}. In this case, the R^2_{train} score could function as *out-of-bag estimate* for the dataset similar to [50]. When dealing with small datasets, the SOM provides the advantage of not requiring a split of the dataset.

Figure 8a shows the distribution of the SOM BMUs of the soil moisture dataset. Instead of finding one single maximum, we recognize rather a random and uniform distribution. The u-matrix shows

the Euclidean distances between the SOM nodes (see Figure 8b). In summary, areas on the SOM grid with low output values show larger distances to the neighboring SOM nodes. Figure 8a illustrates further that, despite the fact that the dataset is smaller than the number of SOM nodes, the training takes advantage of the whole SOM grid. The spread over the whole SOM grid ensures a generalization. The continuous regression output for each SOM node is presented in Figure 8c. Although the SOM nodes outnumber the input datapoints, each SOM node is linked to a soil moisture value.

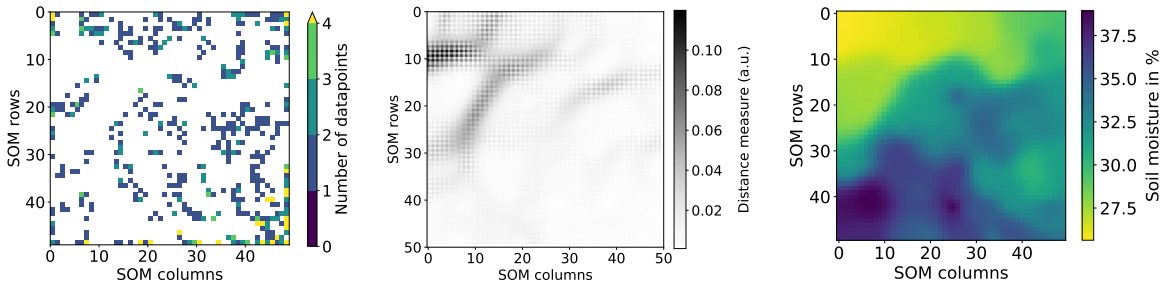


Figure 8. (**a**) regression SOM distribution of the BMUs of the dataset; (**b**) u-matrix; (**c**) distribution of the SOM regression output calculated for each node.

3.3. Supervised Classification of Land Cover

We use the Salinas land cover dataset (see Section 2.2) for the evaluation of the supervised classification SOM. An RF classifier is used as a baseline. The dataset is split into a training and a test dataset in the ratio 30 : 70. The split ratio is selected to ensure enough datapoints in the training dataset and to ensure a proper evaluation. The evaluation results are the average results of five different random seeds (see Section 3.2). Scaling is applied on the hyperspectral input data for the SOM as described in Section 2.2.

Similar to Section 3.2, the hyperparameters of the classification SOM are set after a minor optimization and are provided in Table A2. The hyperparameters of the classification SOM can be further optimized. The RF classifier is set up with 100 estimators and the scikit-learn default hyperparameters (see [34]). For the evaluation, we choose the metrics overall accuracy (OA), average accuracy (AA), and Cohen's kappa coefficient κ. The OA is defined as the ratio between the number of correctly classified datapoints and the size of the dataset. The AA is the sum of the recall of each class divided by the number of classes with the recall of a class being the number of correctly classified instances (datapoints) of this class, divided by the total number of instances of this class. Cohen's kappa coefficient κ is defined, with the hypothetical probability of chance agreement θ, as

$$\kappa = \frac{\mathrm{OA} - \theta}{1 - \theta}. \tag{9}$$

The classification results of the complete dataset are shown in the prediction map in Figure 9. The SOM achieves a test OA of $76.9 \pm 0.2\%$, AA $= 81.4 \pm 0.7\%$ and $\kappa = 74.3 \pm 0.2\%$. The training OA is $77.2 \pm 0.7\%$, the training AA is $81.6 \pm 0.9\%$, and the κ score on the training subsets is $74.6 \pm 0.8\%$. In contrast, the RF classifier achieves a test OA of $93.5 \pm 0.1\%$, AA $= 96.4 \pm 0.1\%$ and $\kappa = 92.7 \pm 0.2\%$, while the RF training metrics are all at about 100%. The RF classifier performs significantly better than the classification SOM. Note that the classification SOM of the SuSi framework has not yet been fully optimized. Similar to the regression results (see Section 3.2), the differences between test and training metrics are lower for the SOM. Two findings can be derived. Firstly, the SOM is robust against overtraining in this study implied by the only marginal difference between training and test accuracies. Secondly, the SOM is not optimized on this classification task in terms of the classification algorithm itself as well as the hyperparameters. To optimize the algorithm and the large number of different

hyperparameters for the classification SOM, we expect the potential users of the open-source SuSi framework [15] to share their experiences and to improve the framework.

Figure 10a shows the distribution of the classification output of the SOM. Nodes assigned to the same classes are closer together on the SOM grid due to the inclusion of the neighborhood during the training process. The u-matrix is shown in Figure 10b. Both plots in Figure 9 can be compared to the clustering results in Figure 6. Similar structures occur. Obvious differences can be explained by the different number of datapoints used for the clustering as well as by the application of different random seeds. The ability to differentiate between the 16 classes is illustrated in the confusion matrix in Figure 11a. Classes like *grapes untrained* and *vinyard untrained* are often confused with the respective other as is also found in Section 3.1. Figure 11b shows the confusion matrix of the RF is shown. As expected from the OA and AA, most of the classes are classified correctly to nearly 100%. The two classes *grapes untrained* and *vinyard untrained* are confused with each other, which is a comparable effect previously seen in the SOM.

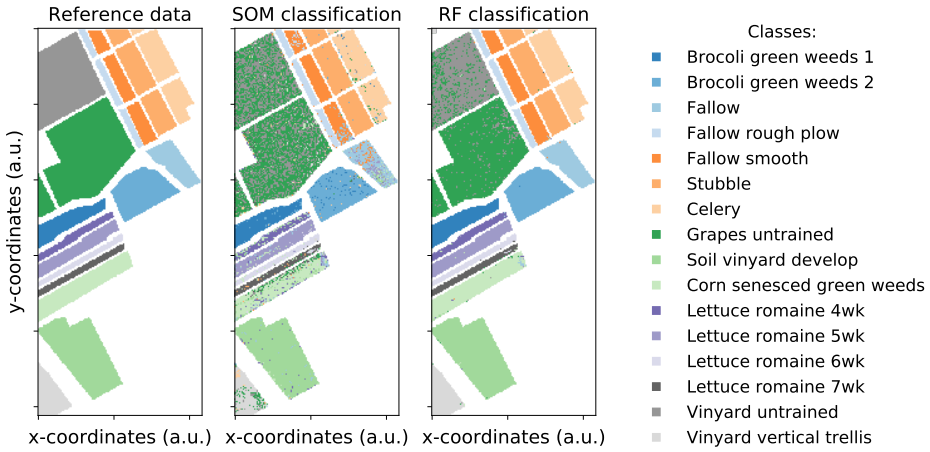
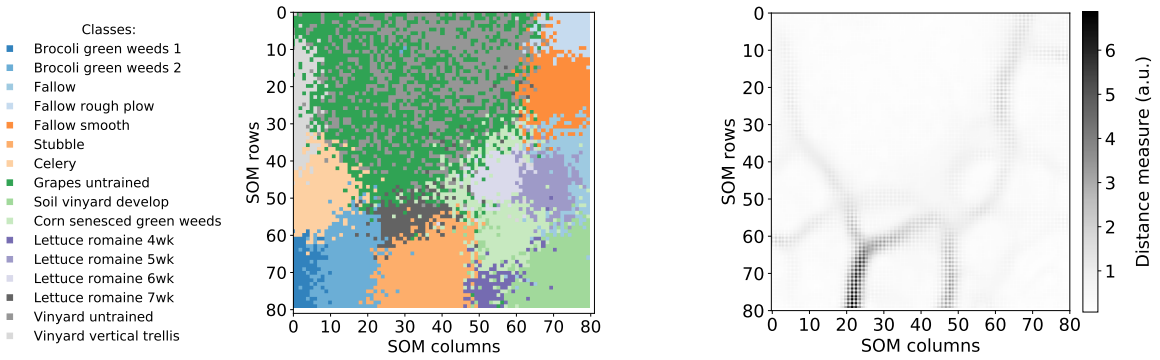
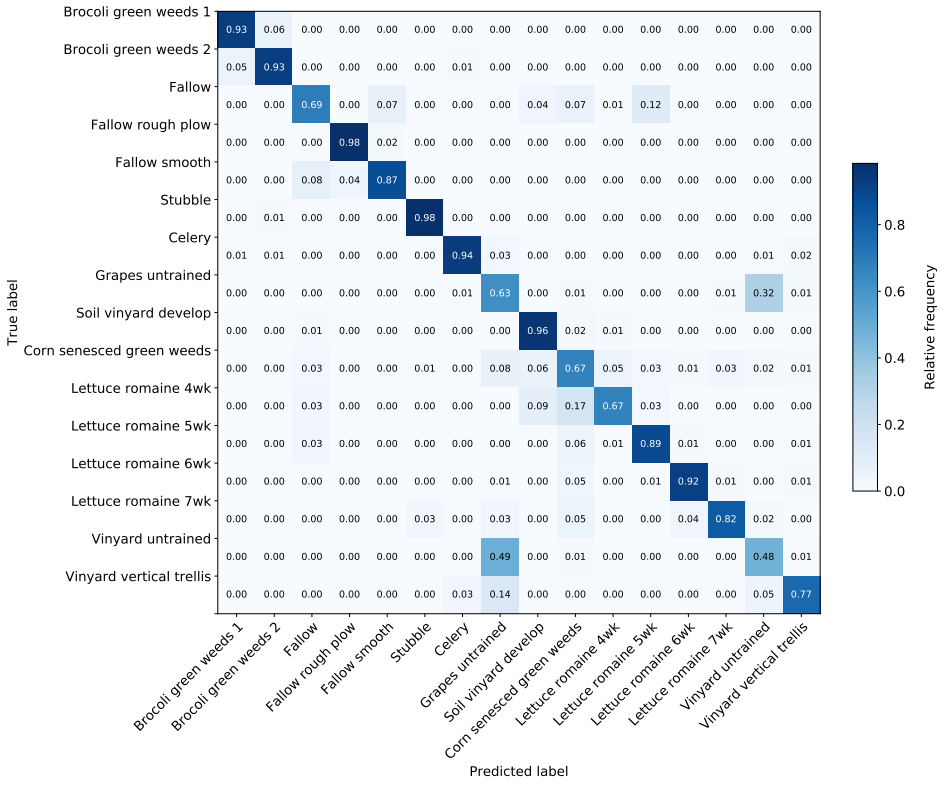


Figure 9. Map of the reference data (**left**) and the classification result of the classification SOM (**center**) and the random forest (RF) classifier (**right**) on the Salinas Valley dataset. The white area is ignored. The coordinates are not available and therefore illustrated in arbitrary units (a.u.).

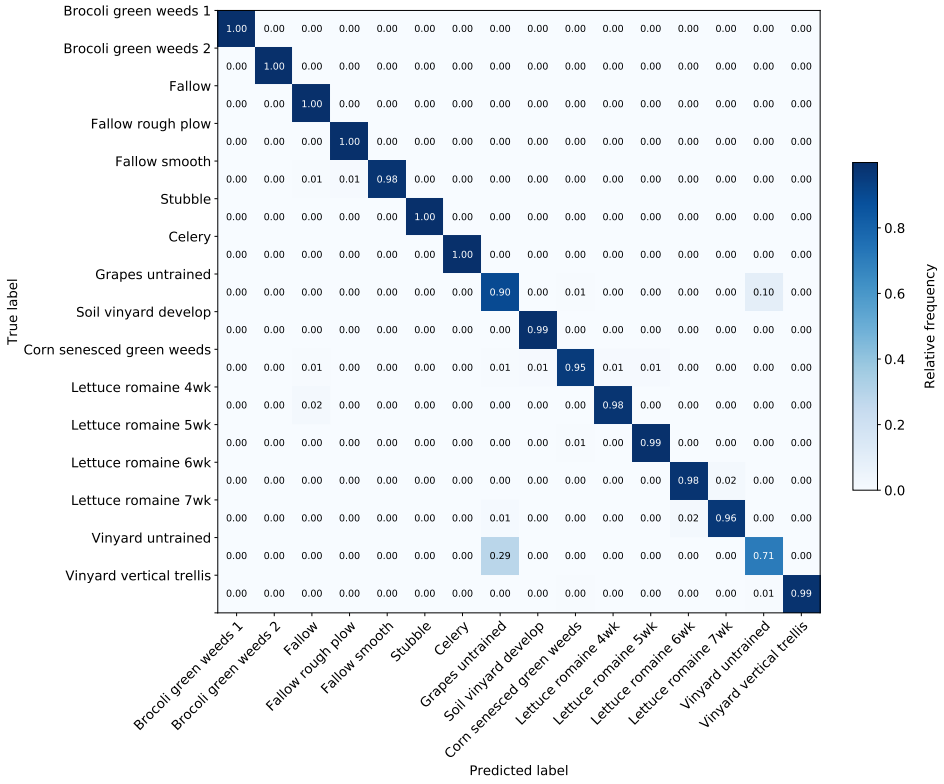


(**a**) (**b**)

Figure 10. (**a**) classification SOM distribution of the classes linked to each node as output of the classification calculated; (**b**) u-matrix of the classification SOM.



(**a**)



(**b**)

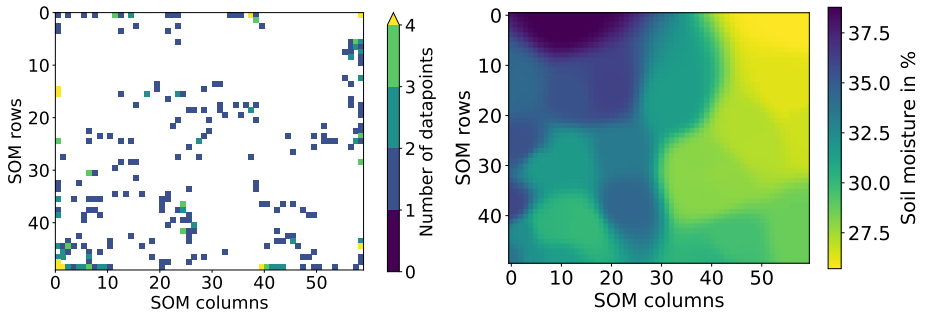
Figure 11. Normalized confusion matrix of (**a**) the classification SOM and (**b**) the RF classifier.

3.4. Semi-Supervised Regression of Soil Moisture with Minimal Reference Data

To evaluate the semi-supervised regression SOM, we use the soil moisture dataset and is applied in a supervised regression (see Section 3.2). For the training, the dataset is split into a training and a test subset in the ratio 1 : 1. Only 10% of the datapoints in the training dataset are labeled. Five different random seeds for the SuSi framework as well as for the dataset split are used to reduce effects of randomization. The hyperparameters of the semi-supervised regression SOM are set after a minor optimization and are provided in Table A2. For the semi-supervised regression SOM, the hyperspectral input data are normalized.

The semi-supervised regression SOM achieves $R^2 = 81.9 \pm 3.2\%$. The distribution of the BMUs of the soil moisture dataset is shown in Figure 12a (supervised regression: Figure 8a) and the continuous regression output for each SOM node can be found in Figure 12b (supervised regression: Figure 8c). Comparing the plots of the semi-supervised and the supervised case, we find that the distribution of the dataset over the full SOM grid as well as generating a continuous output map remain the same for the semi-supervised case.

To evaluate the regression performance of the semi-supervised SOM, we apply an RF regressor with five different random seeds and 1000 estimators. The RF is trained only on the labeled datapoints of the training dataset and tested on the full test dataset. It achieves $R^2 = 71.9 \pm 6.0\%$. This result shows that the semi-supervised regression SOM is able to learn additional information from the unlabeled data and outperforms the RF on such a small labeled dataset.



(**a**) (**b**)

Figure 12. Semi-supervised regression SOM distributions of (**a**) the best matching units (BMUs) of the test dataset and (**b**) the regression output calculated for each node.

3.5. Semi-Supervised Classification of Land Cover with Minimal Reference Data

In the following, we provide a proof-of-concept for the semi-supervised classification SOM. The Salinas dataset (see Section 3.3) provides the basis for the evaluation. To study the semi-supervised functionality of the SuSi framework, we generate a new training dataset consisting of the whole Salinas dataset with the following modification: only two datapoints of every class are labeled, which results in a labeled dataset with 32 labeled pixels (0.06%) and 54,097 unlabeled pixels. The semi-supervised SOM is evaluated based on the full Salinas dataset with all 54,129 labeled pixels. The hyperparameters of the semi-supervised classification SOM are set after a minor optimization and are provided in Table A2. Similarly to the studies before, five different random seeds are applied for the classifiers as well as for the randomized choice of the 32 labeled pixels. To evaluate the classification results of the semi-supervised SOM, a supervised RF is only trained on the 32 labeled datapoints and evaluated on the full datasets with all 54,129 labeled pixels.

The test OA of the semi-supervised classification SOM is $67.3 \pm 3.0\%$, the test AA is $78.6 \pm 2.6\%$, and the test $\kappa = 64.5 \pm 3.3\%$. Compared to the supervised classification results in Section 3.3, the OA is significantly lower while the AA does not differ much. Since the labeled training dataset consists of two datapoints of every class, the AA is expected to be larger than the OA for a very unbalanced

dataset as for the Salinas dataset. A κ of more than 60% shows that the semi-supervised SOM is significantly better than a random classification. The training metrics are calculated based on the 32 labeled datapoints and are for the OA and AA about $99.4 \pm 1.2\%$ and κ of $99.3 \pm 0.0\%$. This indicates the capability of the SOM to adapt to the poorly-labeled dataset.

The supervised RF achieves a test OA of $72.9 \pm 2.8\%$, a test AA of $81.3 \pm 2.0\%$, and a test κ of $70.2 \pm 3.0\%$. Similar to Section 3.3, the train metrics are about 100%. The RF classifier outperforms the semi-supervised approach only by a few percentage points. The differences between the performances of the SOM and the RF are significantly smaller than in the supervised case (see Section 3.3). This finding implies that a supervised approach might be a similar or even better choice in a case of a dataset with only a few labels.

Figure 13 shows the prediction map of the semi-supervised SOM with reasonable results. In general, the fields of the different classes in the Salinas dataset can be recognized as being of one major class. In Figure 14a, the output grid of the semi-supervised SOM with each node assigned to one class is illustrated. Compared to the supervised case in Figure 10a, clear circles can be seen that are artifacts of the early stages, and therefore larger learning rates, of the training of the semi-supervised SOM. With more labeled datapoints, a more diverse output grid could be trained. The distribution of the different labeled datapoints on the SOM grid is shown in Figure 14b. The dataset is spread over the full SOM grid with small clusters at the borders of the grid.

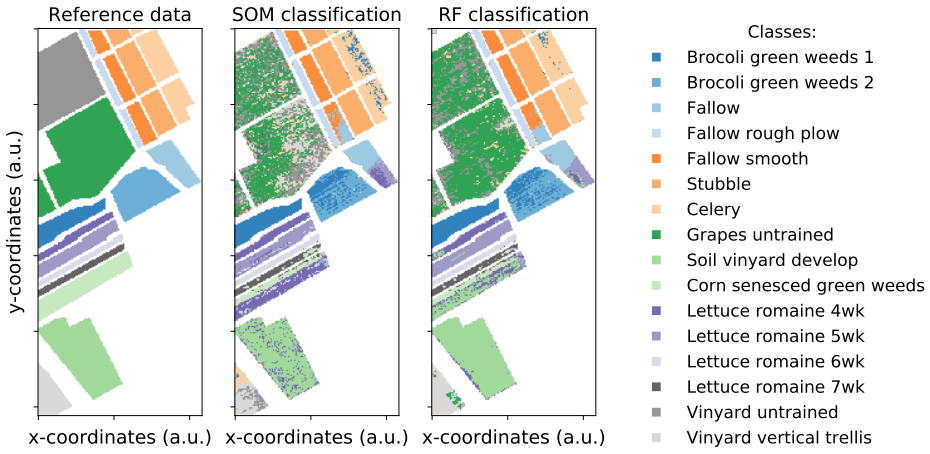


Figure 13. Prediction map of the semi-supervised SOM. The coordinates are not available and therefore illustrated in arbitrary units (a.u.).

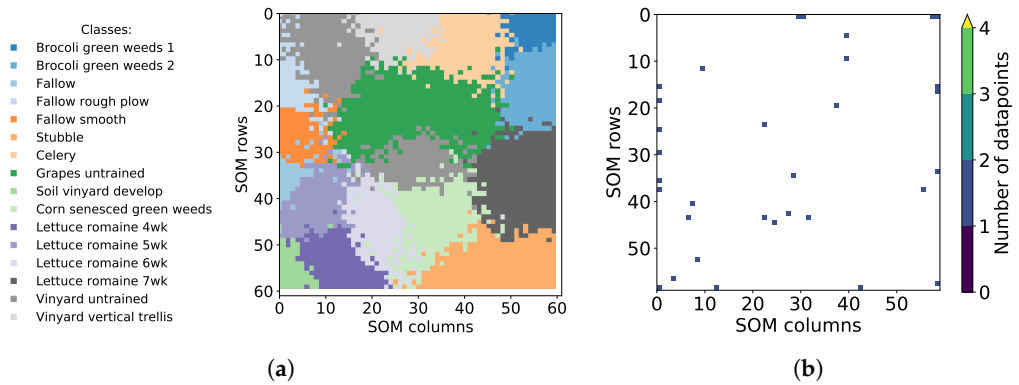


Figure 14. (**a**) distribution of the semi-supervised SOM classification output calculated for each node; (**b**) semi-supervised regression SOM distribution of the BMUs of the dataset.

4. Conclusions

SOMs are applied primarily for unsupervised clustering and visualization in various research fields. In this paper, we introduce a SOM-based framework, the **Su**pervised **S**elf-organ**i**zing maps (SuSi) framework, which can be used for supervised and semi-supervised regression and classification tasks as well as for unsupervised clustering and visualization. We compare the framework with existing Python, R and C packages in Section 2.1 based on a selected list of requirements and based on the ease of use. The mathematical concept of the framework is presented in Sections 2.3–2.5. Examples for clustering and visualization are provided in Section 3.1.

We demonstrate regression and classification results of the supervised SOM in the SuSi framework in Sections 3.2 and 3.3. The regression is performed on a small dataset while the classification SOM is applied on a relatively large dataset. All evaluation results of this paper are summarized in Table 2. The regression SOM outperforms the baseline classifier RF while simultaneously showing less overtraining. Similar to the supervised SOM regression performance, the supervised classification SOM achieves satisfactory results with potential to improve. We find that the performance metric based on the training dataset could function as an *out-of-bag estimate* for the dataset. This suggests that, in the case of the supervised regression and classification SOMs, we do not have to split the dataset necessarily, which might improve the training especially on small datasets.

The unsupervised and supervised capabilities of the SuSi framework are combined for solving semi-supervised tasks. Similar to the supervised regression and classification applications, we apply the semi-supervised regression and classification SOM on two different datasets to evaluate their performances. Both datasets are modified for the semi-supervised evaluation: only a few datapoints in the training dataset remain labeled. While the semi-supervised regression SOM clearly outperforms the RF baseline classifier, the semi-supervised classification SOM achieves satisfying results that are still below the RF performance.

In the future, the SuSi framework will be extended, optimized, and upgraded. In particular, the supervised and semi-supervised classification has great potential for methodological improvements. One promising approach is the batch mode [12] for adapting the SOM weights in Appendix B.5. The handling of missing and incomplete data, as described in [24], is one example for a possible extension. Another possible extension of the semi-supervised SOM is active learning [51]. In active learning, the ML model actively queries for unlabeled datapoints to be labeled that are most helpful for the underlying task. One further advantage of the SOM is the 2D output grid that can be used for visualization of the results of the SOM. This visualization can enhance the understanding for the underlying dataset. For example, the general ability to learn from datasets can be extended according to [22]. Furthermore, we plan to apply the SuSi framework on further datasets and to share our best practices in the context of handling the SuSi framework to ensure its effectiveness in hyperspectral remote sensing.

Table 2. Evaluation results of the soil moisture regression examples in Sections 3.2 and 3.4 as well as of the land cover classification examples in Sections 3.3 and 3.5.

	Metric	Dataset	Supervised		Semi-Supervised	
			SOM	RF	SOM	RF
Regression	R^2 in %	test	95.3 ± 0.8	93.0 ± 2.2	81.9 ± 3.2	71.9 ± 6.0
		training	97.7 ± 0.6	99.1 ± 0.2	-	-
Classification	OA in %	test	76.9 ± 0.2	93.5 ± 0.1	67.3 ± 3.0	72.9 ± 2.8
		training	77.2 ± 0.7	100	99.4 ± 1.2	-
	AA in %	test	81.4 ± 0.7	96.4 ± 0.1	78.6 ± 2.6	81.3 ± 2.0
		training	81.6 ± 0.9	100	99.4 ± 1.2	-
	κ in %	test	74.3 ± 0.2	92.7 ± 0.2	64.5 ± 3.3	70.2 ± 3.0
		training	74.6 ± 0.8	100	99.3 ± 0.0	-

Author Contributions: S.K. and F.M.R. prepared the methodological concept of this study, the original draft as well as the editing of the manuscript. F.M.R. designed the software, curated the data, performed the investigation, formal analysis, validation, and visualization of the results. S.K. and S.H. initialized the related research and provided didactic and methodological inputs. All coauthors contributed to the review of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Variable Naming Conventions

Table A1. Variable naming conventions of the Supervised Self-organizing Maps (SuSi) framework.

Variable	Description
n	Number of features of a datapoint
N	Number of datapoints
n_{row}	Number of rows on the SOM grid
n_{column}	Number of columns on the SOM grid
t	Number of current iteration
$t_{\max}$	Number of maximum iterations, $t < t_{\max}$
$\boldsymbol{x}(t)$	Datapoint at iteration t with $\boldsymbol{x} \in \mathbb{R}^n$
$y(t)$	Label of datapoint $\boldsymbol{x}(t)$
$c(\boldsymbol{x})$	Best matching unit (BMU) of datapoint $\boldsymbol{x}(t)$ with $c \in \mathbb{R}^2$
$\alpha(t)$	Function of the learning rate
α_0	Start value of the learning rate
$\sigma(t)$	Neighborhood function
σ_0	Start value of the neighborhood function
σ_{end}	End value of the neighborhood function
$h_{c,i}$	Neighborhood distance weight between BMU c and SOM node i
$\boldsymbol{w}_i(t)$	Weight of node i at iteration t with $\boldsymbol{w}_i \in \mathbb{R}^n$

Appendix B. Supplementary Mathematical Formulas

Appendix B.1. Additional Distance Metrics

In addition to the *Euclidean distance* defined in Equation (1) in Section 2.3.1, we provide other choices in the following. One possible choice is the *Manhattan distance* which is defined as the sum of the absolute distances per element:

$$d(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^{n} |x_i - x_i'|. \tag{A1}$$

The *Tanimoto distance* as a third option is defined as the distance or dissimilarity between two boolean (binary: T, F) vectors $\boldsymbol{x}, \boldsymbol{x}'$:

$$d(\boldsymbol{x}, \boldsymbol{x}') = \frac{R}{c_{TT} + c_{FF} + R} \qquad \text{with } R = 2(c_{TF} + c_{FT}), \tag{A2}$$

with c_{ij} as the number of occurrences of $x_k = i$ and $x_k' = j$ for all elements k as defined in [52]. The *Mahalanobis distance* between two one-dimensional, vectors $\boldsymbol{x}, \boldsymbol{x}'$ is defined, with the covariance matrix V of the two vectors, as

$$d(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{(\boldsymbol{x} - \boldsymbol{x}')V^{-1}(\boldsymbol{x} - \boldsymbol{x}')^T}. \tag{A3}$$

Appendix B.2. Additional Learning Rates

In addition to the default learning rate defined in Equation (2) in Section 2.3.2, we provide other choices in the following. In [53], different learning rates for SOMs are introduced:

$$\alpha(t) = \alpha_0 \cdot \frac{1}{t}, \tag{A4}$$

$$\alpha(t) = \alpha_0 \cdot \left(1 - \frac{t}{t_{\max}}\right), \tag{A5}$$

$$\alpha(t) = \alpha_0^{t/t_{\max}} \qquad \text{with } \alpha_0 = 0.005. \tag{A6}$$

The following learning rate was applied by [54]:

$$\alpha(t) = \alpha_0 \cdot \exp(-t/t_{\max}). \tag{A7}$$

Appendix B.3. Additional Neighborhood Functions

In addition to the default neighborhood function defined in Equation (3) in Section 2.3.3, we provide other choices in the following. The neighborhood function is implemented by [54] as

$$\sigma(t) = \sigma_0 \cdot \exp(-t/t_{\max}), \tag{A8}$$

equivalent to Equation (A7). The neighborhood function (see [46]) is defined similarly to Equation (2) as

$$\sigma(t) = \sigma_0 \cdot \left(\frac{\sigma_{\text{end}}}{\sigma_0}\right)^{t/t_{\max}}. \tag{A9}$$

Appendix B.4. Mexican Hat Neighborhood Distance Weight

In addition to the default neighborhood distance weight defined in Equation (4) in Section 2.3.4, we provide another implementation in the following. The *Mexican Hat* [11] neighborhood distance weight is defined, with neighborhood function of Equation (3) and Euclidean distance $d(c,i)$ on the SOM grid, as

$$h_{c,i}(t) = \left(1 - \frac{d^2}{\sigma^2}\right) \exp\left(-\frac{d^2}{2 \cdot \sigma(t)^2}\right). \tag{A10}$$

Appendix B.5. Batch Mode

In addition to the default *online mode* defined in Equation (5) in Section 2.3.5, we provide another implementation in the following. In the *batch mode* [47,55], the whole dataset consisting of N datapoints is used in every iteration. Each weight $\boldsymbol{w}_i$ is adapted, with the neighborhood function $h_{c,i}(t)$ from Equation (4) and the weight vector $\boldsymbol{w}_i(t+1)$ of node i at iteration $t+1$, as follows:

$$\boldsymbol{w}_i(t+1) = \frac{\sum_{j=1}^{N} h_{c,i}(t) \cdot \boldsymbol{x}_j}{\sum_{j=1}^{N} h_{c,i}(t)}. \tag{A11}$$

Appendix C. Hyperparameters

Table A2. Hyperparameters of the different SOM evaluations divided into general, unsupervised and supervised hyperparameters. The training time (last row) is not a hyperparameter but is included to present the computation times of the SuSi framework training. These training times are based on an Apple MacBook Pro 13-inch, 2017, with 3.1 GHz Dual-Core Intel Core i5 and 16 GB memory.

Hyperparameters		Unsupervised Clustering	Supervised Regression	Supervised Classification	Semi-Supervised Regression	Semi-Supervised Classification
General	Number of rows	50	50	80	50	60
	Number of columns	50	50	80	60	60
	Distance metric	(1)	(1)	(1)	(1)	(1)
	Learning rate start	0.5	0.7	0.7	0.6	0.6
	Learning rate end	0.05	0.07	0.07	0.07	0.07
	Nbh. distance weight	(4)	(4)	(4)	(4)	(4)
Unsupervised	Initialization	random	random	random	random	random
	Number of iterations	50,000	10,000	60,000	10,000	30,000
	Training mode	online	online	online	online	online
	Neighborhood mode	(3)	(3)	(3)	(A9)	(A9)
	Learning rate	(2)	(2)	(2)	(A7)	(A7)
Supervised	Initialization	-	random	random	random	random
	Number of iterations	-	20,000	60,000	50,000	70,000
	Training mode	-	online	online	online	online
	Neighborhood mode	-	(3)	(3)	(3)	(3)
	Learning rate	-	(2)	(2)	(2)	(2)
	Class weighting	-	-	no	-	no
	Training time in s	200	30	$>$ 1800	38	220

References

1. Treitz, P.M.; Howarth, P.J. Hyperspectral remote sensing for estimating biophysical parameters of forest ecosystems. *Prog. Phys. Geogr. Earth Environ.* **1999**, *23*, 359–390. [CrossRef]
2. Ali, I.; Greifeneder, F.; Stamenkovic, J.; Neumann, M.; Notarnicola, C. Review of Machine Learning Approaches for Biomass and Soil Moisture Retrievals from Remote Sensing Data. *Remote Sens.* **2015**, *7*, 16398–16421. [CrossRef]
3. Colini, L.; Spinetti, C.; Amici, S.; Buongiorno, M.; Caltabiano, T.; Doumaz, F.; Favalli, M.; Giammanco, S.; Isola, I.; La Spina, A.; et al. Hyperspectral spaceborne, airborne and ground measurements campaign on Mt. Etna: Multi data acquisitions in the frame of Prisma Mission (ASI-AGI Project n. I/016/11/0). *Quad. Geofis.* **2014**, *119*, 1–51.
4. Keller, S.; Riese, F.M.; Stötzer, J.; Maier, P.M.; Hinz, S. Developing a machine learning framework for estimating soil moisture with VNIR hyperspectral data. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* **2018**, *IV-1*, 101–108. [CrossRef]
5. Gislason, P.O.; Benediktsson, J.A.; Sveinsson, J.R. Random forests for land cover classification. *Pattern Recognit. Lett.* **2006**, *27*, 294–300. [CrossRef]
6. Camps-Valls, G.; Tuia, D.; Gómez-Chova, L.; Jiménez, S.; Malo, J. Remote sensing image processing. *Synth. Lect. Image, Video, Multimed. Process.* **2011**, *5*, 1–192. [CrossRef]
7. Gewali, U.B.; Monteiro, S.T.; Saber, E. Machine learning based hyperspectral image analysis: A survey. *arXiv* **2018**, arXiv:1802.08701.
8. Shalev-Shwartz, S.; Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*; Cambridge University Press: Cambridge, UK, 2014. [CrossRef]
9. Kohonen, T. Self-organized formation of topologically correct feature maps. *Biol. Cybern.* **1982**, *43*, 59–69. [CrossRef]
10. Kohonen, T. The self-organizing map. *Proc. IEEE* **1990**, *78*, 1464–1480. [CrossRef]
11. Kohonen, T. *Self-Organizing Maps*; Springer: Berlin/Heidelberg, Germany, 1995; Volume 30. [CrossRef]
12. Kohonen, T. Essentials of the self-organizing map. *Neural Netw.* **2013**, *37*, 52–65. [CrossRef]
13. Riese, F.M.; Keller, S. Introducing a Framework of Self-Organizing Maps for Regression of Soil Moisture with Hyperspectral Data. In Proceedings of the 2018 IEEE International Geoscience and Remote Sensing Symposium, Valencia, Spain, 22–27 July 2018; pp. 6151–6154. [CrossRef]

14. Keller, S.; Maier, P.M.; Riese, F.M.; Norra, S.; Holbach, A.; Börsig, N.; Wilhelms, A.; Moldaenke, C.; Zaake, A.; Hinz, S. Hyperspectral Data and Machine Learning for Estimating CDOM, Chlorophyll a, Diatoms, Green Algae, and Turbidity. *Int. J. Environ. Res. Public Health* **2018**, *15*, 1881. [CrossRef] [PubMed]
15. Riese, F.M. *SuSi: SUpervised Self-Organizing Maps in Python*; Zenodo: Geneve, Switzerland, 2019. [CrossRef]
16. Ultsch, A. Self-organizing neural networks for visualisation and classification. In *Information and Classification*; Springer: Berlin/Heidelberg, Germany, 1993; pp. 307–313.
17. Kalteh, A.; Hjorth, P.; Berndtsson, R. Review of the self-organizing map (SOM) approach in water resources: Analysis, modelling and application. *Environ. Model. Softw.* **2008**, *23*, 835–845. [CrossRef]
18. Lobo, V.J.A.S. Application of Self-Organizing Maps to the Maritime Environment. In *Information Fusion and Geographic Information Systems*; Popovich, V.V., Claramunt, C., Schrenk, M., Korolenko, K.V., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 19–36. [CrossRef]
19. Vesanto, J.; Alhoniemi, E. Clustering of the self-organizing map. *IEEE Trans. Neural Networks* **2000**, *11*, 586–600. [CrossRef] [PubMed]
20. Muruzábal, J.; Vidaurre, D.; Sánchez, J. SOMwise regression: A new clusterwise regression method. *Neural Comput. Appl.* **2012**, *21*, 1229–1241. [CrossRef]
21. Hsu, S.H.; Hsieh, J.P.A.; Chih, T.C.; Hsu, K.C. A two-stage architecture for stock price forecasting by integrating self-organizing map and support vector regression. *Expert Syst. Appl.* **2009**, *36*, 7947–7951. [CrossRef]
22. Hsu, K.; Gupta, H.V.; Gao, X.; Sorooshian, S.; Imam, B. Self-organizing linear output map (SOLO): An artificial neural network suitable for hydrologic modeling and analysis. *Water Resour. Res.* **2002**, *38*, 1–17. [CrossRef]
23. Moshou, D.; Bravo, C.; Oberti, R.; West, J.; Bodria, L.; McCartney, A.; Ramon, H. Plant disease detection based on data fusion of hyper-spectral and multi-spectral fluorescence imaging using Kohonen maps. *Real-Time Imaging* **2005**, *11*, 75–83. [CrossRef]
24. Hagenbuchner, M.; Tsoi, A.C. A supervised training algorithm for self-organizing maps for structures. *Pattern Recognit. Lett.* **2005**, *26*, 1874–1884. [CrossRef]
25. Ji, C.Y. Land-Use Classification of Remotely Sensed Data Using Kohonen Self-organizing Feature Map Neural Networks. *Photogramm. Eng. Remote Sens.* **2000**, *66*, 1451–1460.
26. Martinez, P.; Gualtieri, J.; Aguilar, P.; Plaza, A.; Pérez, R.; Preciado, J. Hyperspectral Image Classification using a Self-Organizing Map. In Proceedings of the Tenth JPL Airborne Earth Science Workshop, Pasadena, CA, USA, 27 February–2 March 2001, Volume 10, pp. 267–274.
27. Zaccarelli, N.; Zurlini, G.; Rizzo, G.; Blasi, E.; Palazzo, M. Spectral Self-Organizing Map for hyperspectral image classification. In *Sensors for Environmental Control*; World Scientific Publishing Company Incorporated: Singapore, 2003; pp. 218–223.
28. Zhong, Y.; Zhang, L.; Huang, B.; Li, P. An Unsupervised Artificial Immune Classifier for Multi/Hyperspectral Remote Sensing Imagery. *IEEE TGRS* **2006**, *44*, 420–431. [CrossRef]
29. Fessant, F.; Aknin, P.; Oukhellou, L.; Midenet, S. Comparison of Supervised Self-Organizing Maps Using Euclidian or Mahalanobis Distance in Classification Context. In *Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence*; Mira, J., Prieto, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2084, pp. 637–644. [CrossRef]
30. Hecht, T.; Lefort, M.; Gepperth, A. Using self-organizing maps for regression: The importance of the output function. In *European Symposium on Artificial Neural Networks (ESANN)*; Inria: Rocquencourt, France, 2015; pp. 1–6.
31. Chapelle, O.; Schölkopf, B.; Zien, A. *Semi-Supervised Learning*; Adaptive Computation and Machine Learning; MIT Press: Cambridge, MA, USA, 2006; p. 508. [CrossRef]
32. Abaei, G.; Selamat, A.; Fujita, H. An empirical study based on semi-supervised hybrid self-organizing map for software fault prediction. *Knowl.-Based Syst.* **2015**, *74*, 28–39. [CrossRef]
33. Braga, P.H.; Bassani, H.F. A Semi-Supervised Self-Organizing Map for Clustering and Classification. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8. [CrossRef]
34. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.

35. Wehrens, R.; Kruisselbrink, J. Flexible Self-Organizing Maps in kohonen 3.0. *J. Stat. Softw.* **2018**, *87*, 1–18. [CrossRef]
36. Moosavi, V.; Packmann, S.; Vallés, I. SOMPY: A Python Library for Self Organizing Map (SOM). 2018. Available online: https://github.com/sevamoo/SOMPY (accessed on 4 March 2019).
37. Comitani, F. SimpSOM: A Lightweight Implementation of Kohonen Self-Organising Maps. Release 1.3.3. 2018. Available online: https://github.com/fcomitani/SimpSOM (accessed on 4 March 2019).
38. Vettigli, G. MiniSom: Minimalistic and NumPy-Based Implementation of the Self Organizing Map. Release 2.1.5. 2019. Available online: https://github.com/JustGlowing/minisom (accessed on 4 March 2019).
39. Gorman, C. A Multi-Gpu Implementation of the Self-Organizing Map in TensorFlow. 2018. Available online: https://github.com/cgorman/tensorflow-som (accessed on 7 November 2018).
40. Hanke, M.; Halchenko, Y.O.; Sederberg, P.B.; Olivetti, E.; Fründ, I.; Rieger, J.W.; Herrmann, C.S.; Haxby, J.V.; Hanson, S.J.; Pollmann, S. PyMVPA: A unifying approach to the analysis of neuroscientific data. *Front. Neuroinform.* **2009**, *3*, 3. [CrossRef] [PubMed]
41. Shevchuk, Y. NeuPy. 2015. Available online: http://neupy.com/ (accessed on 1 October 2019).
42. Riese, F.M.; Keller, S. Hyperspectral benchmark dataset on soil moisture. 2018, doi:10.5281/zenodo.1227836.
43. Grupo de Inteligencia Computacional de la Universidad del País Vasco. Salinas AVIRIS Dataset. Available online: http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes#Salinas (accessed on 13 March 2019).
44. Attik, M.; Bougrain, L.; Alexandre, F. Self-organizing Map Initialization. In *Artificial Neural Networks: Biological Inspirations—ICANN 2005*; Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3696, pp. 357–362. [CrossRef]
45. Akinduko, A.A.; Mirkes, E.M.; Gorban, A.N. SOM: Stochastic initialization versus principal components. *Inf. Sci.* **2016**, *364-365*, 213–221. [CrossRef]
46. Barreto, G.A.; Araújo, A.F.R. Identification and Control of Dynamical Systems Using the Self-Organizing Map. *IEEE Trans. Neural Networks* **2004**, *15*, 1244–1259. [CrossRef]
47. Matsushita, H.; Nishio, Y. Batch-Learning Self-Organizing Map with Weighted Connections Avoiding False-Neighbor Effects. In Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, 18–23 July 2010; pp. 1–6. [CrossRef]
48. Ritter, H.; Martinetz, T.; Schulten, K. *Neural Computation and Self-Organizing Maps: An Introduction*; Addison-Wesley: Reading, MA, USA, 1992.
49. Horowitz, R.; Alvarez, L. Convergence properties of self-organizing neural networks. In Proceedings of the 1995 American Control Conference-ACC'95, Seattle, WA, USA, 21–23 June 1995; Volume 2, pp. 1339–1344.
50. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
51. Zhang, Y.; Yang, H.L.; Prasad, S.; Pasolli, E.; Jung, J.; Crawford, M. Ensemble multiple kernel active learning for classification of multisource remote sensing data. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2014**, *8*, 845–858. [CrossRef]
52. Jones, E.; Oliphant, T.; Peterson, P. SciPy: Open source scientific tools for Python. *Comput. Sci. Eng.* **2001**. [CrossRef]
53. Natita, W.; Wiboonsak, W.; Dusadee, S. Appropriate Learning Rate and Neighborhood Function of Self-organizing Map (SOM) for Specific Humidity Pattern Classification over Southern Thailand. *Int. J. Model. Optim.* **2016**, *6*, 61–65. [CrossRef]
54. de Sá, J.A.S.; da Rocha, B.R.P.; Almeida, A.; Souza, J.R. Recurrent Self-Organizing Map for Severe Weather Patterns Recognition. In *Recurrent Neural Networks and Soft Computing*; IntechOpen: Rijeka, Croatia, 2012; Chapter 8, pp. 151–174. [CrossRef]
55. Kohonen, T.; Somervuo, P. How to make large self-organizing maps for nonvectorial data. *Neural Netw.* **2002**, *15*, 945–952. [CrossRef]