Clustering of Satellite Image Time Series with SITS

Lorena Santos *National Institute for Space Research (INPE), Brazil*
Karine Ferreira *National Institute for Space Research (INPE), Brazil*
Rolf Simoes *National Institute for Space Research (INPE), Brazil*
Gilberto Camara *National Institute for Space Research (INPE), Brazil*

One of the key challenges when using samples to train machine learning classification models is assessing their quality. Noisy and imperfect training samples can have a negative effect on classification performance. Therefore, it is useful to apply pre-processing methods to improve the quality of the samples and to remove those that might have been wrongly labeled or that have low discriminatory power. Representative samples lead to good classification maps. `sits` provides support for two clustering methods to test sample quality, which are agglomerative hierarchical clustering (AHC) and self-organizing maps (SOM).

Introduction: Clustering for sample quality control

One of the key challenges when using samples to train machine learning classification models is assessing their quality. Noisy and imperfect training samples can have a negative effect on classification performance [Frénay and Verleysen, 2013]. Therefore, it is useful to apply pre-processing methods to improve the quality of the samples and to remove those that might have been wrongly labeled or that have low discriminatory power. Representative samples lead to good classification maps. `sits` provides support for two clustering methods to test sample quality: (a) Agglomerative Hierarchical Clustering (AHC); (b) Self-organizing Maps (SOM).

Hierachical clustering

Cluster analysis has been used for many purposes in satellite image time series literature ranging from unsupervised classification [Petitjean et al., 2011], and pattern detection [Romani et al., 2011]. Here, we are interested in the second use of clustering, using it as a way to improve training data to feed machine learning classification models. In this regard, cluster analysis can assist the identification of structural *time series patterns* and anomalous samples [Romani et al., 2011], [Chandola et al., 2009].

Agglomerative hierarchical clustering (AHC) is a family of methods that groups elements using a distance function to associate a real value to a pair of elements. From this distance measure, we can compute the dissimilarity between any two elements from a data set. Depending on the distance functions and linkage criteria, the algorithm decides which two clusters are merged at each iteration. AHC approach is suitable for the purposes of samples data exploration due to its visualization power and ease of use [Keogh et al., 2003]. Moreover, AHC does not require a predefined number of clusters as

an initial parameter. This is an important feature in satellite image time series clustering since defining the number of clusters present in a set of multi-attribute time series is not straightforward [Aghabozorgi et al., 2015].

The main result of AHC method is a *dendrogram*. It is the ultrametric relation formed by the successive merges in the hierarchical process that can be represented by a tree. Dendrograms are quite useful to decide the number of clusters to partition the data. It shows the height where each merging happens, which corresponds to the minimum distance between two clusters defined by a *linkage criterion*. The most common linkage criteria are: *single-linkage*, *complete-linkage*, *average-linkage*, and *Ward-linkage*. Complete-linkage prioritizes the within-cluster dissimilarities, producing clusters with shorter distance samples. Complete-linkage clustering can be sensitive to outliers, which can increase the resulting intracluster data variance. As an alternative, Ward proposes a criteria to minimize the data variance by means of either *sum-of-squares* or *sum-of-squares-error* [Ward, 1963]. Ward's intuition is that clusters of multivariate observations, such as time series, should be approximately elliptical in shape [Hennig, 2015]. In `sits`, a dendrogram can be generated by `sits_dendrogram()`. The following codes illustrate how to create, visualize, and cut a dendrogram (for details, see `?sits_dendrogram()`).

After creating a dendrogram, an important question emerges: *where to cut the dendrogram?* The answer depends on what are the purposes of the cluster analysis [Hennig, 2015]. If one is interested in an unsupervised classification, it is common to use *internal validity indices*, such as silhouettes [Rousseeuw, 1987], to help determine the best number of clusters. However, if one is interested in understanding the structure of a labeled data set, or in the identifying sample anomalies, as we are here, one can use *external validity indices* to assist the semi-supervised procedure in order to achieve the optimal correspondence between clusters and classes partitions. In this regard, we need to balance two objectives: get clusters as large as possible, and get clusters as homogeneous as possible with respect to their known classes. To help this process, `sits` provides `sits_dendro_bestcut()` function that computes an external validity index *Adjusted Rand Index* (ARI) for a series of different number of generated clusters. This function returns the height where the cut of the dendrogram maximizes the index.

In this example, the height optimizes the ARI and generates 6 clusters. The ARI considers any pair of distinct samples and computes the following counts: (a) the number of distinct pairs whose samples have the same label and are in the same cluster; (b) the number of distinct pairs whose samples have the same label and are in different clusters; (c) the number of distinct pairs whose samples have different labels and are in the same cluster; and (d) the number of distinct pairs whose samples have the different labels and are in different clusters. Here, a and d consist in all agreements, and b and c all disagreements. The ARI is obtained by:

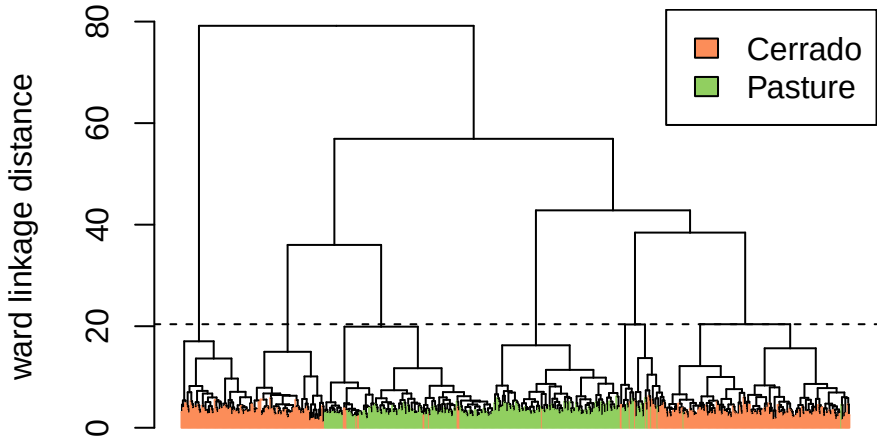
$$ARI = \frac{a+d-E}{a+d+b+c-E},$$

where E is the expected agreement, a random chance correction calculated by

$$E = (a+b)(b+c)+(c+d)(b+d).$$

Unlike other validity indexes such as Jaccard ($J = a/(a+b+c)$), Fowlkes-Mallows ($FM = a/(a^2 + a(b+c) + bc)^{1/2}$), and Rand (the same as ARI without the E adjustment) indices, ARI is more appropriate either when the number of clusters is outweighed by the number of labels (and *vice versa*) or when the amount of samples in labels and clusters is imbalanced [Hubert and Arabie, 1985], which is usually the case.

```
# take a set of patterns for 2 classes
# create a dendrogram object, plot, and get the optimal cluster based on ARI index
clusters.tb <- sits::sits_cluster_dendro(cerrado_2classes, bands = c("ndvi", "evi"))
```




```
# show clusters samples frequency
sits::sits_cluster_frequency(clusters.tb)
```

```
##
##              1   2   3   4   5   6 Total
##   Cerrado  203  13  23  80   1  80   400
##   Pasture    2 176  28   0 140   0   346
##   Total    205 189  51  80 141  80   746
```

Note in this example that almost all clusters has a predominance of either "Cerrado" or "Pasture" classes with the exception of cluster 3. The contingency table plotted by `sits_cluster_frequency()` shows how the samples are distributed across the clusters and helps to identify two kinds of confusions. The first is relative to those small amount of samples in clusters dominated by another class (*e.g.* clusters 1, 2, 4, 5, and 6), while the second is relative to those samples in non-dominated clusters (*e.g.* cluster 3). These confusions can be an indication of samples with poor quality, an inadequacy of selected parameters for cluster analysis, or even a natural confusion due to the inherent variability of the land classes.

The result of the `sits_cluster` operation is a `sits_tibble` with one additional column, called "cluster". Thus, it is possible to remove clusters with mixed classes using standard `R` such as those in the `dplyr` package. In the example above, removing cluster 3 can be done using the `dplyr::filter` function.

```
# remove cluster 3 from the samples
clusters_new.tb <- dplyr::filter(clusters.tb, cluster != 3)

# show new clusters samples frequency
sits_cluster_frequency(clusters_new.tb)
```

```
##
##              1   2   4   5   6 Total
##   Cerrado  203  13  80   1  80   377
##   Pasture    2 176   0 140   0   318
##   Total    205 189  80 141  80   695
```

The resulting clusters still contained mixed labels, possibly resulting from outliers. In this case, users may want to remove the outliers and leave only the most frequent class. To do this, one can use `sits_cluster_clean()`, which removes all minority samples, as shown below..

```
# clear clusters, leaving only the majority class in each cluster
cleaned.tb <- sits_cluster_clean(clusters.tb)
# show clusters samples frequency
sits_cluster_frequency(cleaned.tb)
```

```
##
##              1   2   3   4   5   6 Total
##   Cerrado  203   0   0  80   0  80   363
##   Pasture    0 176  28   0 140   0   344
##   Total    203 176  28  80 140  80   707
```

Self-organizing maps

As an alternative for hierachical clustering for quality control of training samples, SITS provides a clustering technique based on self-organizing maps (SOM). SOM is a dimensionality reduction technique [Kohonen et al., 2001], where high-dimensional data is mapped into two dimensions, keeping the topological relations between data patterns. The input data is a high dimset of tranining samples which are typically of a high dimension. For example, a time series of 25 instances of 4 spectral bands is a 100-dimensional data set. The general idea of SOM-based clustering is that, by projecting the high-dimensional data set of training samples into a 2D map, the units of the map (called "neurons") compete for each sample. It is expected that good quality samples of each class should be close together in the resulting map.

The process of clustering with SOM is done by `sits_cluster_som()` which creates a self-organizing map. First, a 2D grid of neurons is initialized randomly; each neuron is associated to a weight vector of the same dimension as the input space. For each time series sample, the algorithm finds the neuron with the smallest distance based on its

weight vector. After the match, the neuron's weight vector and those of its neighbors are then updated. After all training samples are associated with neurons, each neuron is labeled using a majority vote, taking the most frequent class from the samples associated with it. In this way, SOM splits the output space into regions. To increase the reliability of quality control procedures, the SOM clustering can be executed several times. Using this iterative procedure, the algorithm computes the probability that a sample belongs to each of the resulting clusters. From these probabilities, samples with similar phenological patterns are grouped together. This allows using SOM to detect and remove outliers; samples whose set of probabilities has a high variance are discarded.

More formally, for each neuron j SOM uses a vector of weights, $w_j = [w_{j1}, \ldots, w_{jn}]$,that has the same dimension of the input vector of time series samples $x(t) = [x(t)_1, \ldots, x(t)_n]$, where n is the time series dimension. In the begining, the neurons are initialized randomly. At each training step, a time series sample $x(t)$ is presented to the network in order to find the neuron whose weight vector has the smaller distance to the sample. Distance metrics D_j compute the distance between input sample $x(t)$ and the weight vector of a neuron. This is computed for each neuron j in the output layer. The most commonly used metric is Euclidean distance, shown below.

$$D_j = \sum_{i=1}^{n} \sqrt{(x(t)_i - w_{ji})^2}.$$

The neuron that contains the shortest distance, defined here as b, is denoted as the best matching unit (BMU):

$$d_b = min\left\{D_1, \ldots, D_J\right\}.$$

Once the BMU is selected for a given sample, its neighborhood must be updated. The weight of each neighbour is adjusted according to the similarity with input vector. The equation for updating the weight vector is given by:

$$w_j(t+1) = w_j(t) + \alpha(t) * h_{b,j}(t)[x(t)_i - w_j(t)],$$

where $\alpha(t)$ is the learning rate, which must be set as $0 < \alpha(t) < 1$, and $h_{b,j}(t)$ is a neighbourhood function.

As an example of the use of SOM clustering for quality control of samples, we take a data set 617 time series samples for the combination of the LANDSAT images for an area of the Brazilian Amazon rain forest. This area corresponds to the LANDAT WRS 226/04. The LANDSAT pixels have been combined with the MOD13Q1 collection 5 images, to fill the gaps where there is too much cloud in the LANDSAT data. The data set has the following classes (and samples per class): Deforestation_2014 (146 samples), Deforestation_2015 (198 samples), Forest (128 samples), and Pasture (145 samples). We first run the SOM cluster function.

```
# clustering time series using SOM
som_cluster <-
    sits::sits_som_map(
```

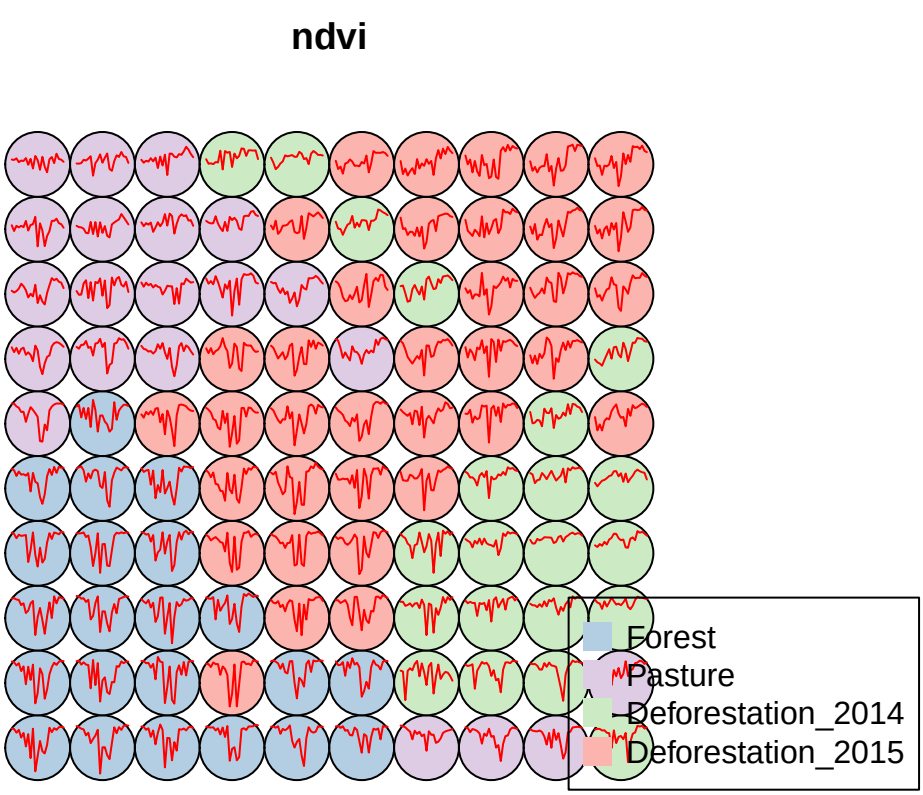
```
        prodes_226_064,
        grid_xdim = 10,
        grid_ydim = 10,
        alpha = 1.0,
        distance = "euclidean",
        iterations = 40
    )
```

The output of the `sits_som_map` is a list with 4 tibbles:

- the original time series with four additional columns: `id_sample` (the original id of each sample), `som_label` (the cluster label assigned to the respective neuron on the SOM map) `cluster_probability` (the probability of a samble belonging to that cluster) and `total_probability`. Each input sample can be assigned to two or more clusters with different probabilities.
- one tibble with the association between the labels of original samples and the labels of the neurons.
- one tibble with the association bewteen each neuron and its neighbours.

To plot the kohonen map, use `plot.som_map()`, which can be called directly from `plot()`. The neurons are labelled using the majority voting.

```
plot(som_cluster)
```



From the statistics obtained from function `sits_som_map`, a new dataset can be generated removing the samples with bad quality. The function `sits_som_clean_samples()` removes the bad samples based on the probability a samples belongs to a cluster.

```
new_samples <- sits_som_clean_samples(som_cluster)
new_samples
```

```
## # A tibble: 529 x 9
##    id_sample longitude latitude start_date end_date   label cube
##        <int>     <dbl>    <dbl> <date>     <date>     <chr> <chr>
##  1         1     -53.4    -6.53 2014-08-04 2015-07-22 Defo~ mixl~
##  2         2     -53.4    -6.51 2014-08-04 2015-07-22 Defo~ mixl~
##  3         4     -53.9    -6.23 2014-08-04 2015-07-22 Defo~ mixl~
##  4         5     -52.8    -6.29 2014-08-04 2015-07-22 Defo~ mixl~
##  5         6     -53.5    -6.17 2014-08-04 2015-07-22 Defo~ mixl~
##  6         7     -53.7    -6.07 2014-08-04 2015-07-22 Defo~ mixl~
##  7         8     -53.7    -6.08 2014-08-04 2015-07-22 Defo~ mixl~
##  8         9     -53.0    -6.35 2014-08-04 2015-07-22 Defo~ mixl~
##  9        10     -53.7    -6.09 2014-08-04 2015-07-22 Defo~ mixl~
## 10        11     -53.7    -6.10 2014-08-04 2015-07-22 Defo~ mixl~
## # ... with 519 more rows, and 2 more variables: time_series <list>,
## #   probability <dbl>
```

To verify the quality of the clusters generated by SOM, a confusion matrix, the overall accuracy and the statistics about each class is evaluated.

```
cluster_overall <- sits_som_evaluate_cluster(som_cluster)
cluster_overall$confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##
##                    Deforestation_2014 Deforestation_2015 Forest Pasture
##   Deforestation_2014                124                  5      2      15
##   Deforestation_2015                 19                159     14       7
##   Forest                              1                  4    120       5
##   Pasture                            16                  1      2     126
##
## Overall Statistics
##
##                Accuracy : 0.8532
##                  95% CI : (0.8229, 0.8802)
##     No Information Rate : 0.2726
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.8036
##
##  Mcnemar's Test P-Value : 0.002917
##
## Statistics by Class:
##
##                     Class: Deforestation_2014 Class: Deforestation_2015
## Sensitivity                            0.7750                    0.9408
## Specificity                            0.9522                    0.9113
## Pos Pred Value                         0.8493                    0.7990
## Neg Pred Value                         0.9241                    0.9762
## Prevalence                             0.2581                    0.2726
## Detection Rate                         0.2000                    0.2565
## Detection Prevalence                   0.2355                    0.3210
## Balanced Accuracy                      0.8636                    0.9261
##                      Class: Forest Class: Pasture
## Sensitivity                 0.8696         0.8235
## Specificity                 0.9793         0.9593
## Pos Pred Value              0.9231         0.8690
## Neg Pred Value              0.9633         0.9432
## Prevalence                  0.2226         0.2468
## Detection Rate              0.1935         0.2032
## Detection Prevalence        0.2097         0.2339
## Balanced Accuracy           0.9244         0.8914
```

Besides that, a table showing how a table showing how pure the cluster is computed.

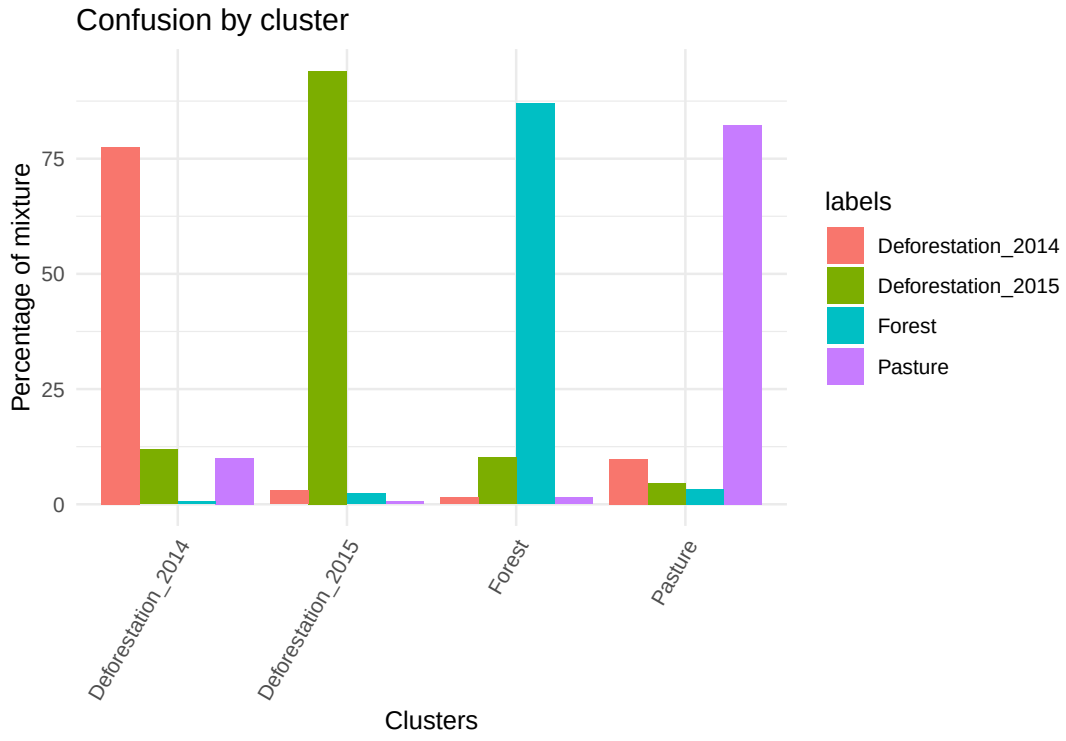
```
cluster_overall$mixture_cluster
```

```
## # A tibble: 16 x 4
##    id_class cluster            original_class     mixture_percentage
##       <int> <chr>              <chr>                           <dbl>
##  1        1 Deforestation_2014 Deforestation_2014              77.5
##  2        1 Deforestation_2014 Deforestation_2015              11.9
##  3        1 Deforestation_2014 Pasture                         10
##  4        1 Deforestation_2014 Forest                           0.625
##  5        2 Deforestation_2015 Deforestation_2015              94.1
##  6        2 Deforestation_2015 Deforestation_2014               2.96
##  7        2 Deforestation_2015 Forest                           2.37
##  8        2 Deforestation_2015 Pasture                          0.592
##  9        3 Forest             Forest                          87.0
## 10        3 Forest             Deforestation_2015              10.1
## 11        3 Forest             Deforestation_2014               1.45
## 12        3 Forest             Pasture                          1.45
## 13        4 Pasture            Pasture                         82.4
## 14        4 Pasture            Deforestation_2014               9.80
## 15        4 Pasture            Deforestation_2015               4.58
```

```
## 16       4 Pasture            Forest                        3.27
```

Finally, a graphic showing the mixture within each cluster is presented.

```
sits_som_plot_clusters(cluster_overall, "Confusion by cluster")
```



References

Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.

Christian Hennig. Clustering strategy and method selection. In Christian Hennig, Marina Meila, Fionn Murtagh, and Roberto Rocci, editors, *Handbook of cluster analysis*. CRC Press, 2015.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2 (1):193–218, 1985.

Eamonn Keogh, Jessica Lin, and Wagner Truppel. Clustering of time series subsequences is meaningless: Implications for previous and future research. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 115–122, 2003.

T. Kohonen, M. R. Schroeder, and T. S. Huang. *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition, 2001.

François Petitjean, Jordi Inglada, and Pierre Gançarskv. Clustering of satellite image time series under time warping. In *Analysis of Multi-temporal Remote Sensing Images (Multi-Temp), 2011 6th International Workshop on the*, pages 69–72. IEEE, 2011.

LAS Romani, RRV Gonçalves, BF Amaral, DYT Chino, J Zullo, C Traina, EPM Sousa, and AJM Traina. Clustering analysis applied to NDVI/NOAA multitemporal images to improve the monitoring process of sugarcane crops. In *Analysis of Multi-temporal Remote Sensing Images (Multi-Temp), 2011 6th International Workshop on the*, pages 33–36. IEEE, 2011.

Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

Joe H Ward. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.