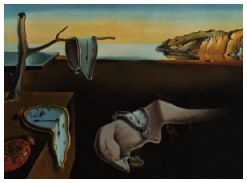

Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

Bayesian inference for smoothing of remote sensing image classification using machine learning

Gilberto Camara
National Inst for Space Research, Brazil

Rolf Simoes
National Inst for Space Research, Brazil

Renato Assunção
Federal Univ of Minas Gerais, Brazil

Alexandre Carvalho
Inst for Applied Economic Research, Brazil

Felipe Souza
National Inst for Space Research, Brazil

Pedro Ribeiro de Andrade
National Institute for Space Research, Brazil

Abstract

The abstract of the article.

Keywords: Bayesian smoothing, image classification, machine learning, R.

1. Introduction

Image classification post-processing has been defined as *"a refinement of the labelling in a classified image in order to enhance its classification accuracy"* (Huang, Lu, Zhang, and Plaza 2014). In remote sensing image analysis, these procedures are used to combine pixel-based classification methods with a spatial post-processing method to remove outliers and misclassified pixels. For pixel-based classifiers, post-processing methods enable the inclusion of spatial information in the final results.

Smoothing methods are an important complement to machine learning algorithms for image classification. Methods such as support vector machines (Mountrakis, Im, and Ogole 2011), random forests (Belgiu and Dragut 2016) and deep learning (Ma, Liu, Zhang, Ye, Yin, and Johnson 2019) are becoming increasingly popular. Since these methods are mostly pixel-based,it is useful to complement them with post-processing smoothing to include spatial information in the result.

A traditional choice for smoothing classified images is the majority filter, where the class of the central pixel is replaced by the most frequent class of the neighbourhood. This technique is rather simplistic; more sophisticated methods use class probabilities. For each pixel, machine learning and other statistical algorithms provide the probabilities of that pixel belonging to each of the classes. As a first step in obtaining a result, each pixel is assigned to the class whose probability is higher. After this step, smoothing methods use class probabilities to detect and correct outliers or misclassified pixels.

Probability-based smoothing methods include Gaussian and edge-aware filtering (Schindler 2012), modal filters (Ghimire, Rogan, and Miller 2010) and probabilistic relaxation (Gong and Howarth 1989). Huang *et al.* (2014) propose a relearning method based on co-ocurrence matrices; these matrices represent the joint distribution of class labels in the neighborhood of each pixel. In the current work, we introduce a Bayesian smoothing method, which provides the means to incorporate prior knowledge in data analysis.

Bayesian inference can be thought of as way of coherently updating our uncertainty in the light of new evidence. It allows the inclusion of expert knowledge on the derivation of probabilities. As stated by Spiegelhalter and Rice (2009): “In the Bayesian paradigm, degrees of belief in states of nature are specified. Bayesian statistical methods start with existing ‘prior’ beliefs, and update these using data to give ‘posterior’ beliefs, which may be used as the basis for inferential decisions”.

Bayesian smoothing methods have become an established technique for image restoration and reconstruction (Hanson 1993). More recently, Wu, Du, Cui, and Zhang (2017) proposed a Bayesian approach for soft fusion applied to change detection. However, to the best of our knowledge this work is the first proposal to use Bayesian smoothing for post-processing of machine learning pixel based classification.

% TODO: include a paragraph about R packages following the style below In R, . . . include here a discussion of what’s available in R, and include mention to the **sits** package

The **sits** package uses satellite image time series for land classification, using
a time-first, space-later approach Simoes, Camara, Queiroz, Souza, Andrade, Santos, Carvalho, and Ferreira (2021). In the data preparation part, collections of big Earth observation images are organized as data cubes. Each spatial location of a data cube is associated with a time series. Locations with known labels train a machine learning classifier, which classifies all time series of a data cube. The package has tools for analysis, visualization, and classification of satellite image time series.

2. Methods

2.1. Conversion from probabilities to logits

The proposed post-classification smoothing model considers the output of a machine learning algorithm that provides the probabilities of each pixel in the image to belong to target classes. More formally, let $p_{i,k}$ be the probability of pixel i belonging to class k, $k = 1,\ldots,K$, then

$$\sum_{k=1}^{K} p_{i,k} = 1, p_{i,k} > 0. \tag{1}$$

To more the inference tractable, the class probability values $p_{i,k}$ are converted to log-odds values. The logit function converts probability values from 0 to 1 to values from $-\infty$ to ∞. The conversion from probabilities to logit values is useful to support the assumption of normal distribution for the data. The conversion is expressed by

$$x_{i,k} = \ln\left(\frac{p_{i,k}}{1-p_{i,k}}\right) \tag{2}$$

Confidence in pixel classification increases with logit. There are situations, such as border pixels or mixed ones, where the logit of different classes are similar in magnitude. These are cases of low confidence in the classification result. To assess and correct these cases, the post-classification smoothing method borrows strength from the neighbors and reduces the variance of the estimated class.

2.2. Bayesian inference

After the transformation of probability values into logits, the problem can be expressed in a Bayesian context. Suppose that $x \in X$ has a distribution $\pi(x|\theta)$ with an unknown $\theta \in \Theta$ parameter. Consider two random variables for each pixel i and class$k$: (a) $x_{i,k}$, the observed class logits; (b) $\mu_{i,k}$, the inferred logit values based on the observations. From the output of the machine learning classification is $x_{i,k}$, the aim is to estimate the actual values $\mu_{i,k}|x_{i,k}$. The Bayesian inference procedure can be expressed as

$$\pi(\mu|x) \propto \pi(x|\mu)\pi(\mu). \tag{3}$$

To estimate the conditional posterior distribution $\pi(\mu|x)$, we combine two distributions: (a) the distribution $\pi(x|\mu)$, known as the likelihood, which expresses how measured values $x_{i,k}$ depend in the underlying values $\mu_{i,k}$; and (b) $\pi(\mu)$, which is our guess on the actual data distribution, known as the prior. For simplicity, independence between the different classes k is assumed. Therefore, the update will be performed for each class k separately.

The method assumes that the likelihood $x_{i,k}|\mu_{i,k}$ follows a normal distribution $N(\mu_{i,k},\sigma_k^2)$ with parameters $\mu_{i,k}$ and σ_k^2. The variance σ_k^2 will be estimated based on user expertise and taken as a hyperparameter to control the smoothness of the resulting estimate. Therefore

$$x_{i,k}|\mu_{i,k} \sim N(\mu_{i,k},\sigma_k^2) \tag{4}$$

is the likelihood. We will also assume a normal local prior for the parameter $\mu_{i,k}$ with parameters $m_{i,k}$ and $s^2_{i,k}$:

$$\mu_{i,k} \sim N(m_{i,k}, s^2_{i,k}). \tag{5}$$

To calculate the prior, the method assumes that the class probabilities in the spatial neighbourhood of the pixel have the same distribution. Estimating the local means and variances for the prior distribution considers a spatial neighbourhood N_i close to pixel p_i. Let $\#(N_i)$ be the number of elements in the neighbourhood N_i. The mean value is given by

$$m_{i,k} = \frac{\sum_{(j)\in N_i} x_{j,k}}{\#(N_i)} \tag{6}$$

and the variance by

$$s^2_{i,k} = \frac{\sum_{(j)\in N_i} [x_{j,k} - m_{i,k}]^2}{\#(N_i) - 1}. \tag{7}$$

Given these assumptions, the Bayesian update for the expected conditional mean $E[\mu_{i,k}|x_{i,k}]$ is given by:

$$E[\mu_{i,k}|x_{i,k}] = \frac{m_{i,t} \times \sigma^2_k + x_{i,k} \times s^2_{i,k}}{\sigma^2_k + s^2_{i,k}} \tag{8}$$

which can be expressed as a weighted mean

$$E[\mu_{i,k}|x_{i,k}] = \left[\frac{s^2_{i,k}}{\sigma^2_k + s^2_{i,k}}\right] \times x_{i,k} + \left[\frac{\sigma^2_k}{\sigma^2_k + s^2_{i,k}}\right] \times m_{i,k} \tag{9}$$

where

- $x_{i,k}$ is the logit value for pixel i and class k.
- $m_{i,k}$ is the average of logit values for pixels of class k in the neighborhood of pixel i.
- $s^2_{i,k}$ is the variance of logit values for pixels of class k in the neighborhood of pixel i.
- σ^2_k is the prior variance of the logit values for class k, and is an user-derived hyperparameter.

The above equation is a weighted average between the value $x_{i,k}$ for the pixel and the mean $m_{i,k}$ for the neighboring pixels. When the variance $s^2_{i,k}$ for the neighbors is too high, the algorithm gives more weight to the pixel value $x_{i,k}$. On the other hand, when the noise σ^2_k increases, the method gives more weight to the neighborhood mean $m_{i,k}$.

2.3. Effect of the hyperparameter

The parameter σ_k^2 controls the level of smoothness. If σ_k^2 is zero, the smoothed value $E[\mu_{i,k}|x_{i,k}]$ will be equal to the pixel value $x_{i,k}$. Making σ_k^2 high leads to much smoothness. Values of the prior variance σ_k^2, which are small relative to the local variance $s_{i,k}^2$ increase our confidence in the original probabilities. Conversely, prior variances σ_k^2 which are large relative to the local variance $s_{i,k}^2$ increase our confidence in the average probability of the neighborhood.

Thus, the parameter σ_k^2 expresses our confidence in the inherent variability of the distribution of values of a class k. The smaller the parameter σ_k^2, the more we trust the estimated probability values produced by the classifier for class k. Conversely, higher values of σ_k^2 indicate lower confidence in the classifier outputs and improved confidence in the local average values.

Consider the following two-class example. Take a pixel with probability 0.4 (logit $x_{i,1}$ = -0.4054) for class A, and probability 0.6 (logit $x_{i,2} = 0.4054$) for class B. Without post-processing, the pixel will be labeled as class B. Consider that the local average is 0.6 (logit $m_{i,1} = 0.4054$) for class A and 0.4 (logit $m_{i,2}$ = -0.4054) for class B. This is a case of an outlier classified originally as class B in the midst of a set of pixels of class A. Take the local variance of logits to be $s_{i,1}^2 = 5$ for class A and $s_{i,2}^2 =$ 10 and for class B. This difference is to be expected if the local variability of class A is smaller than that of class B.

To complete the estimate, we need to set the parameter σ_k^2, representing prior belief in the variability of the probability values for each class. If we take both σ_A^2 for class A and σ_B^2 for class B to be both 10, the Bayesian estimated probability for class A is 0.52 and for class B is 0.48. In this case, the pixel will be relabeled as being class A. However, if our belief in the original values is higher, we will get a different result. If we set σ^2 to be 5 for both classes A and B, the Bayesian probability estimate will be 0.48 for class A and 0.52 for class B. In this case, the original label will be kept.

2.4. Defining the neighbourhood

The intuition for Bayesian smoothing is that homogeneous neighbourhoods should have a dominant class that has both higher average probabilities and lower variance than the others. In these neighbourhoods, a pixel of a different class is likely to be associated to lower average probabilities and higher local variance. Thus, the post-processing will not change the most likely class for pixels in spatially homogenous regions.

The rationale behind Bayesian smoothing is that areas with similar characteristics have a dominant class that has higher average probabilities and lower variance than other classes. In such regions, a pixel with a different class is expected to have lower average probabilities and higher local variance. Post-processing should not change the most likely class for pixels in these regions.

When pixels are located at the borders between areas with different classes, they contain signatures of two classes, which poses a classification problem. For local class statistics to be reliable, only pixels likely to belong to this class should be included. When windows are centered on border pixels, only some of the pixels in the window belong to the same class as the central pixel; the others belong to a different class. To account for border pixels, sits uses a special definition of a neighbourhood to estimate the prior class distribution.

To estimate prior distribution using spatial neighbourhood, the algorithm only uses pixels with high probability of belonging to the class. The logic is that pixels with low probabilities are unlikely to belong to the same class distribution as those with high probability values. This procedure ensures a more reliable estimate of the prior.

3. Software and examples

3.1. Reading a probability data cube

The input for the **sits** post-classification is an image file with the probabilities produced by a machine learning algorithm. This file should be multi-band, where each band contains the pixel probabilities of a single class. The image should be GDAL-readable with INT2S data type with values ranging from [0..10000]. The file name should have information on reference dates and includes a version number. The following code creates a probability cube from a file produced by a random forests classification applied to a data cube of Sentinel-2 images, covering tile "20LLQ" in the Brazilian Amazonia covering the period from 2020-06-04 to 2021-08-26. The original analysis-ready files were obtained from Microsoft Planetary Computer. The training data has six classes: (a) `Forest` for natural tropical forest; (b) `Water` for lakes and rivers; (c) `Wetlands` for areas where water covers the soil in the wet season; (d) `ClearCut_Burn` for areas where fires cleared the land after tree removal. (e) `ClearCut_Soil` where the forest has been removed; (f) `ClearCut_Veg` where some vegetation remains after most trees have been removed.

The first step is informing the location of the file. In this case, the image containing the probabilities is provided by the **sitsdata** package.

```
R> data_dir <- system.file("/extdata/Rondonia-20LLQ/", package = "sitsdata")
R> list.files(data_dir)

[1] "SENTINEL-2_MSI_20LLQ_2020-06-04_2021-08-26_probs_v1.tif"
```

The class labels should also be informed by the user, since they are not stored in image files.

```
R> labels <- c("Water", "ClearCut_Burn", "ClearCut_Soil",
+              "ClearCut_Veg", "Forest", "Wetland")
```

In **sits**, `sits_cube()` builds an object describing the image; its main parameters are: (a) `source`: data provider. Examples are "AWS" and "MPC"; (b)`collection`: image collection on the data provider, e.g., "SENTINEL-2-L2A" (Sentinel-2 collection from MPC); (c)`data_dir`: directory where the image is available; (d)`bands`: Band names associated with the data, in this case `probs`; (e) `labels`: Labels with the training classes; (f) `version`: version of the result; (g) `parse_info`: how to extract `tile`, `band`, `date` and `version` information from the file name. It includes designators such as `X1` and `X2` which are place holders for parts of the file name that is not relevant.

```
R> probs_cube <- sits_cube(
+   source = "MPC",
+   collection = "SENTINEL-2-L2A",
```

```
+    bands = "probs",
+    labels = labels,
+    parse_info = c(
+      "X1", "X2", "tile", "start_date", "end_date", "band", "version"
+    ),
+    data_dir = data_dir
+ )
```

The output of `sits_cube()` is a tibble that contains metadata about the probability maps and associated file. Figure 1 shows the layers of the data cube. The map for class `Forest` shows high probability values associated with compact patches and linear stretches in riparian areas. Class `ClearCut_Soil` is mostly composed of dense areas of high probability whose geometrical boundaries result from forest cuts. By contrast, the probability maps for classes `Water`, `ClearCut_Burn` and `ClearCut_Veg` have mostly low values.

```
R> plot(probs_cube)
```

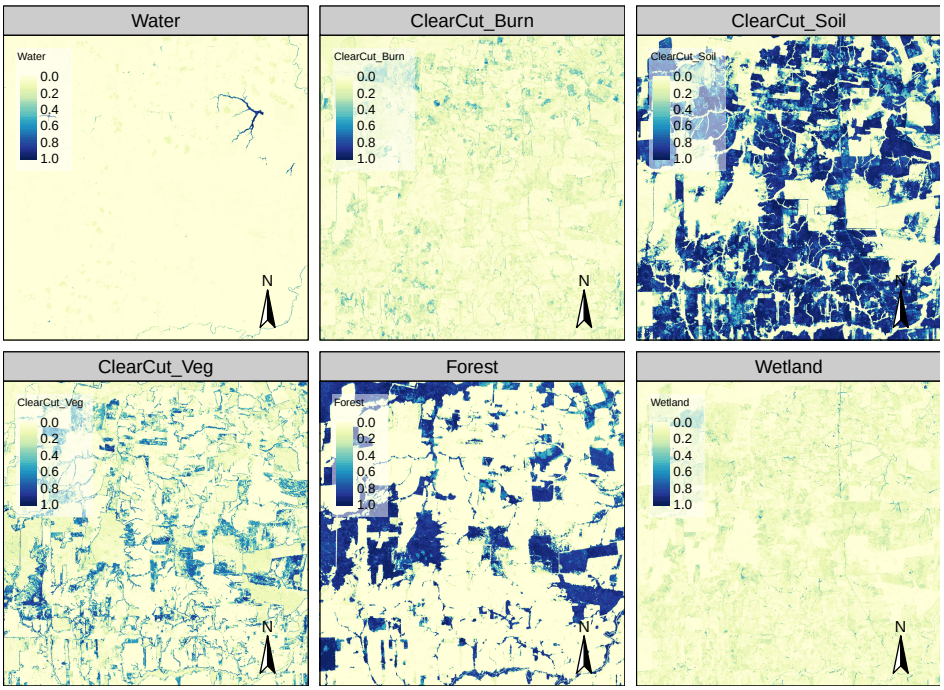

Figure 1: Class probabilities produced by random forest algorithm.

To show the need for post-processing, it is useful to produce a non-smoothed labelled map. The map is obtained by taking the class of higher probability for each pixel as its label without considering the spatial context. This is done by `sits_label_classification()` whose main parameters are: (a) `cube`, a probability cube; (b) `output_dir`, directory for storing results; (c) `version`.

```
R> label_cube_no_smooth <- sits_label_classification(
+      cube = probs_cube,
```

```
+      output_dir = tempdir(),
+      version = "no_smooth")
```

Figure 2 shows the resulting map, which contains a significant number of outliers and pixels likely to be misclassified.

```
R> plot(label_cube_no_smooth)
```

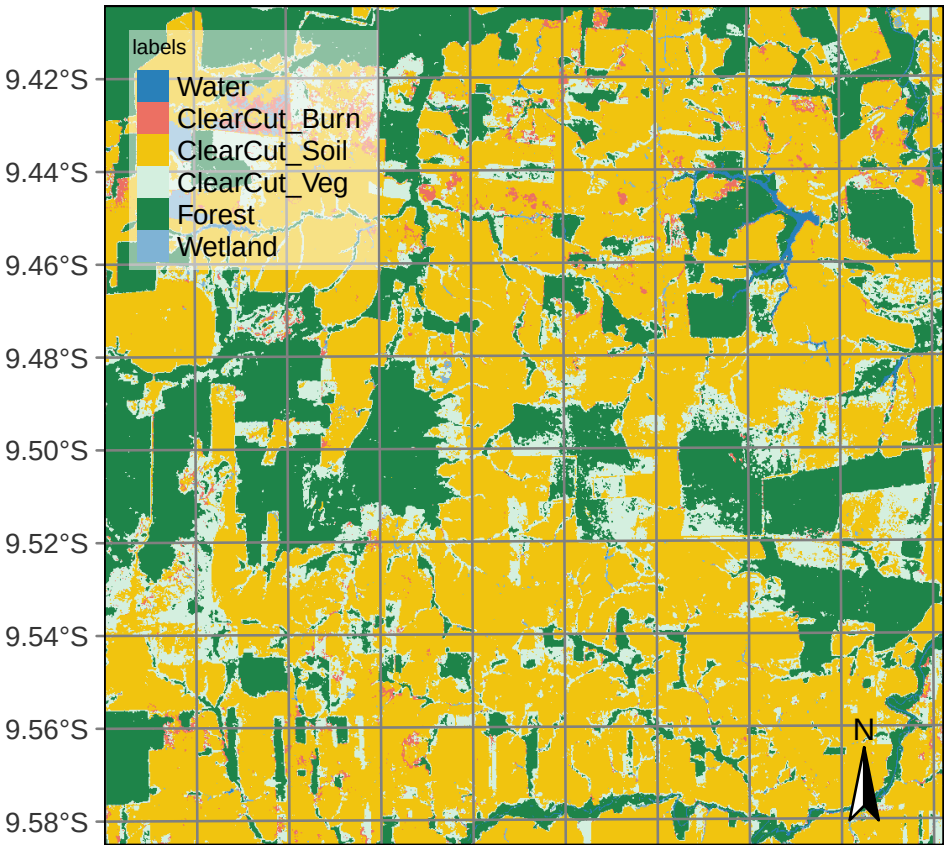

Figure 2: Labelled map without smoothing.

3.2. Estimating the local logit variances

Before applying the Bayesian smoothing procedure to remove outliers, it is useful to consider the spatial distribution and the histogram of the logit variances, which correspond to the $s^2_{i,k}$ parameter in the Bayesian estimator. For this purpose, `sits_variance()` estimates the local logit variances. Its main parameters are: (a) `cube`, a probability cube; (b) `window_size`, the dimension of the local neighbourhood; (c) `neigh_fraction`, the percentage of pixels in the neighbourhood used to calculate the variance; (d) `output_dir`, directory where results will be stored. The example uses half of the pixels of a 7x7 window to estimate the variance. The chosen pixels will be those with the highest probability pixels to be more representative of the actual class distribution. The output values are the logit variances in the vicinity of each pixel.

```
R> var_cube <- sits_variance(
+     cube = probs_cube,
+     window_size = 7,
+     neigh_fraction = 0.5,
+     output_dir = tempdir())
R> summary(var_cube, only_stats = TRUE)
```

```
 Water            ClearCut_Burn     ClearCut_Soil     ClearCut_Veg
 Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.000
 1st Qu.: 0.090   1st Qu.: 0.0600   1st Qu.: 0.0700   1st Qu.: 0.110
 Median : 1.560   Median : 0.1300   Median : 0.1700   Median : 0.390
 Mean   : 2.303   Mean   : 0.3076   Mean   : 0.5014   Mean   : 1.213
 3rd Qu.: 4.420   3rd Qu.: 0.2800   3rd Qu.: 0.4600   3rd Qu.: 1.280
 Max.   :24.450   Max.   :12.4400   Max.   :21.7000   Max.   :22.530
 Forest           Wetland
 Min.   : 0.000   Min.   : 0.0000
 1st Qu.: 0.060   1st Qu.: 0.0500
 Median : 0.500   Median : 0.1200
 Mean   : 1.844   Mean   : 0.4373
 3rd Qu.: 3.030   3rd Qu.: 0.2700
 Max.   :28.240   Max.   :12.0600
```

The choice of the 7 x 7 window size is a compromise between having enough values to estimate the parameters of a normal distribution and the need to capture local effects for class patches of small sizes. Classes such as `Water` and `ClearCut_Burn` tend to be spatially limited; a bigger window size could result in invalid values for their respective normal distributions.

The summary statistics show that most local variance values are low, which is an expected result. Areas of low variance correspond to pixel neighbourhoods of high logit values for one the classes and low logit values for the others. High values of the local variances are relevant in areas of confusion between classes. Figure 3 shows the values of local logit variances for classes `ClearCut_Veg` and class `Forest`, considering only the 4th quartile of the distribution. Only the top 25% of the values for each class are shown, emphasizing areas of high local variability.

```
R> plot(var_cube, percentile = 0.75, labels = c("ClearCut_Veg", "Forest"))
```

Comparing the logit variance maps of Figure 3 with the probability maps of Figure 1 shows the relevance of expert knowledge. The areas of high probability of class `Forest` are mostly made of compact patches; areas of high local variance occur near the borders of these patches. By contrast, class `ClearCut_Veg` represents a transition between natural forest areas and places where all trees have been cut, which are associated to class `ClearCut_Soil`. Class `ClearCut_Veg` has a high spectral variability, since the extent of remaining vegetation after most trees have been removed is not uniform. For this reason, the local variance of class `ClearCut_Veg` is mostly patch-based, while that of class `Forest` is mostly border-based.

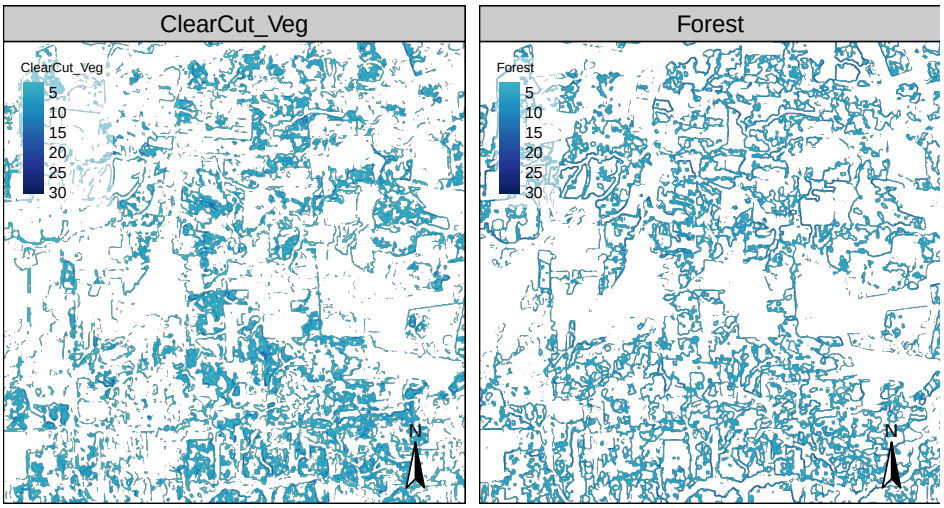

Figure 3: Logit variance map showing values above the 3rd quartile.

Further insights into the local variance are provided by Figure 4 which shows the histograms of local variances per class. The values shown correspond to the 4th quartile (greater that 75% of all values) . The distribution of logit variances is uneven between the classes. Class `ClearCut_Veg` has a more balanced distribution, while most values in the 4th quartile of the `ClearCut_Soil` class have low values.

```
R> plot(var_cube, type = "hist")
```

3.3. Applying Bayesian smoothing to remove outliers

To remove the outliers in the classification map, **sits** uses `sits_smooth()`. Its main parameters are: (a) `cube`, a probability cube; (b) `window_size`, the dimension of the local neighbourhood; (c) `neigh_fraction`, the percentage of pixels in the neighbourhood used to calculate the variance; (d) `smoothness`, a vector with prior logit variances for each class; (e) `output_dir`, directory where results will be stored; (f) `version`, a string to distinguish different runs.

As discussed above, the effect of the Bayesian estimator depends on the values of the a priori variance σ^2_k set by the user and of the neighbourhood definition used to compute to local variance $s^2_{i,1}$ for each pixel. To show the effects of different σ^2 values we consider two cases: (a) setting σ^2 to a high value close to the maximum value of local logit variance; (b) setting σ^2 to a lower value close to the minimum value of the 4rd quartile.

```
R> smooth_cube_high <- sits_smooth(
+     cube = probs_cube,
+     window_size = 7,
+     neigh_fraction = 0.5,
+     smoothness = c(25, 10, 20, 20, 25, 10),
+     output_dir = tempdir(),
+     version = "smooth_high")
```

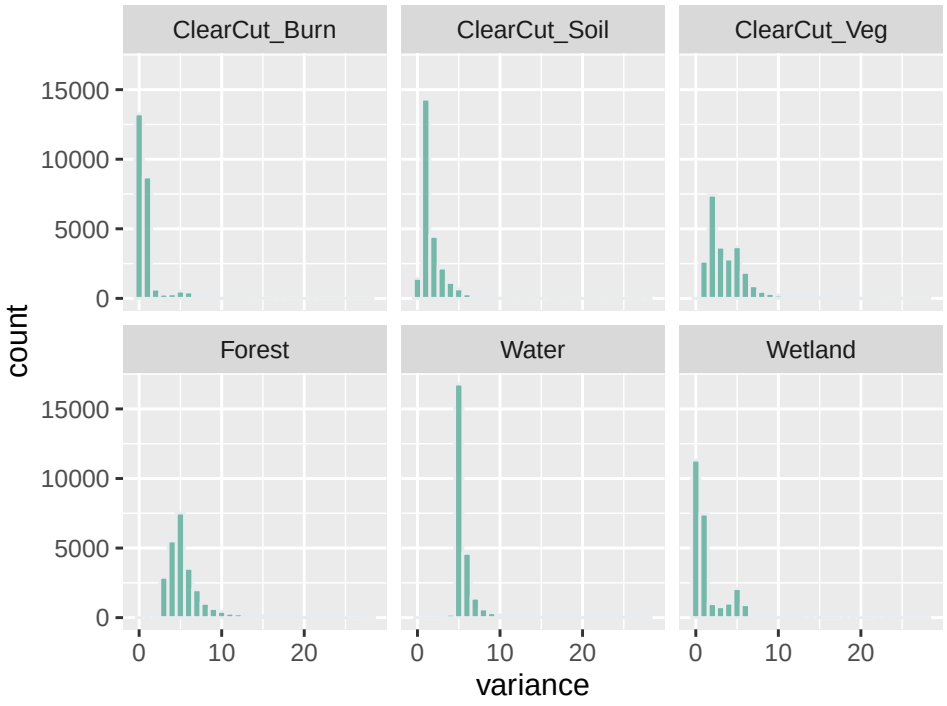

Figure 4: Histogram of logits of class variances above the 3rd quartile.

The impact of Bayesian smoothing can be best captured by producing a labelled map using `sits_label_classification()` taking the smoothed image as its input. Figure 5 shows that the outliers and isolated pixels have been removed.

```
R> label_cube_smooth_high <- sits_label_classification(
+    cube = smooth_cube_high,
+    output_dir = tempdir(),
+    version = "smooth_high"
+ )
R> plot(label_cube_smooth_high)
```

In the smoothed map, the outliers have been removed by expanding forest areas. Forests have replaced small corridors of water and soil encircled by trees. This effect is due to the high probability of forest detection in the training data. Compare the smoothing with high values with the smoothing with values close to the minimum value of the 4th quartile of the local variance for each class, as computed below.

```
R> smooth_cube_low <- sits_smooth(
+      cube = probs_cube,
+      window_size = 7,
+      neigh_fraction = 0.5,
+      smoothness = c(5, 1, 1, 2, 4, 1),
+      output_dir = tempdir(),
+      version = "smooth_low")
```

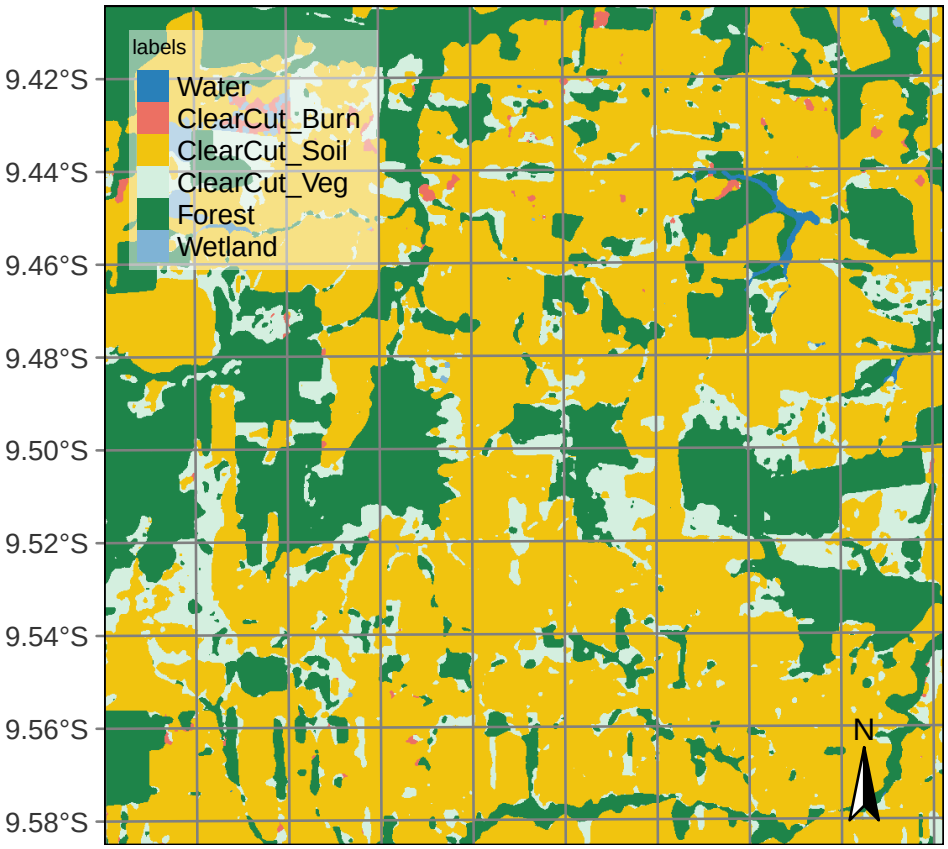

Figure 5: Labelled map with smoothing with high smoothness values.

To see the impact of small σ^2, we compute the labelled map.

```
R> label_cube_smooth_low <- sits_label_classification(
+    cube = smooth_cube_low,
+    output_dir = tempdir(),
+    version = "smooth_low"
+ )
R> plot(label_cube_smooth_low)
```

Comparing the class areas of the non-smoothed and smoothed maps shown below, the most frequent classes (`ClearCut_Soil` and `Forest`) increased their areas at the expense of the others. As shown in Figure **??**, these classes occur in more compact patches than the others.

```
R> sum1 <- summary(label_cube_no_smooth, only_stats = TRUE)
R> names(sum1) <- c("class", "area_k2_no_smooth")
R> sum2 <- summary(label_cube_smooth_high, only_stats = TRUE)
R> names(sum2) <- c("class", "area_k2_smooth_high")
R> sum3 <- summary(label_cube_smooth_low, only_stats = TRUE)
R> names(sum3) <- c("class", "area_k2_smooth_low")
R> dplyr::inner_join(sum1, sum2, by = "class") %>%
+    dplyr::inner_join(sum3, by = "class")
```

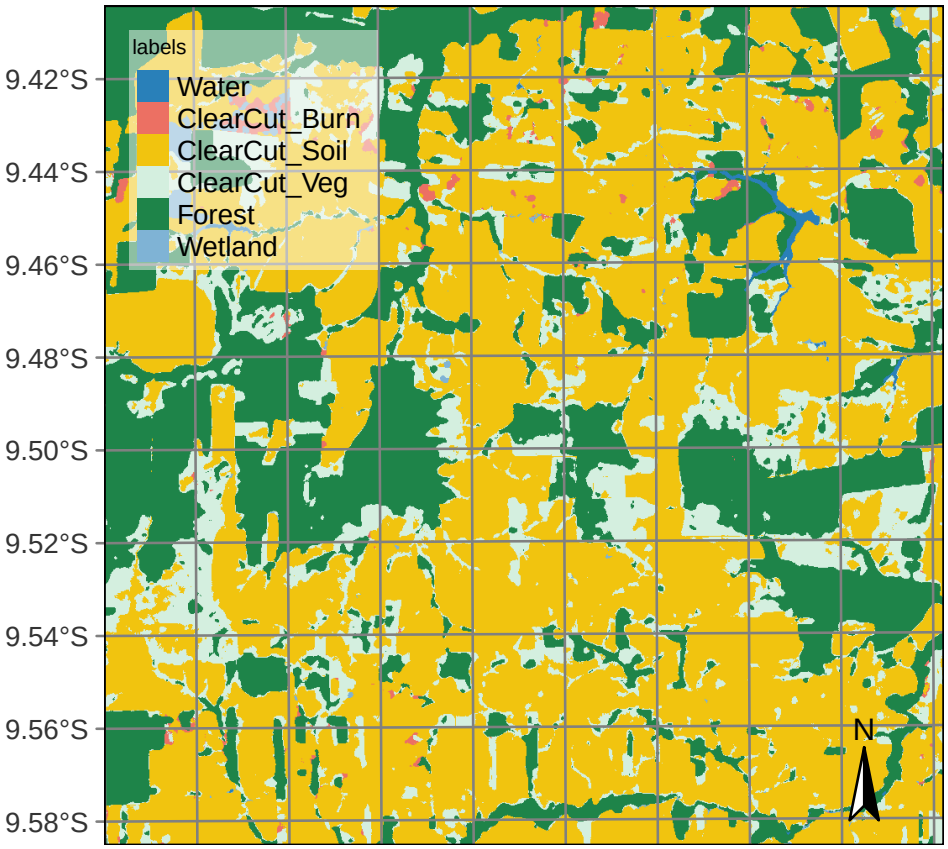

Figure 6: Labelled map with small smoothing parameters.

	class	area_k2_no_smooth	area_k2_smooth_high	area_k2_smooth_low
1	Water	1.767	1.030	1.106
2	ClearCut_Burn	7.743	2.043	2.771
3	ClearCut_Soil	215.400	234.500	224.700
4	ClearCut_Veg	65.350	47.620	57.470
5	Forest	105.100	112.500	111.200
6	Wetland	4.667	2.311	2.834

In the smoothed map, the outliers have been removed by expanding forest areas. Forests have replaced small corridors of water and soil encircled by trees. This effect is due to the high probability of forest detection in the training data. To keep the water areas and reduce the expansion of the forest area, a viable alternative is to reduce the smoothness (σ^2) for the "Forest" and "Water" classes. In this way, the local influence of the forest in the other classes is reduced. As for the water areas, since they are narrow, their neighborhoods will have many low probability values, which would reduce the expected value of the Bayesian estimator.

Based on expert knowledge, there are two main options for setting the σ_k^2 parameter:

1. To increase the neighbourhood influence compared with the probability values for each pixel, set the σ_k^2 parameter with high values (20 or above) . Classes whose probabilities have strong spatial autocorrelation will tend to replace outliers of classes with higher spatial variance variability.

2. To reduce the neighbourhood influence compared with the probabilities for each pixel of class k, set the σ_k^2 parameter with low values (5 or below) . In this way, classes with low spatial autocorrelation are more likely not to be relabelled.

Consider the case of forest areas and watersheds. If an expert wishes to have compact areas classified as forests without many outliers inside them, she will set the σ^2 parameter for the class "Forest" to be high. For comparison, to avoid that small watersheds with few similar neighbors being relabeled, it is advisable to avoid a strong influence of the neighbors, setting σ^2 to be as low as possible. Therefore, choice of σ^2 depends on the effect intended by the expert in the final clasified map.

To keep the water areas and reduce the expansion of the forest area, a viable alternative is to reduce the smoothness (σ^2) for the "Forest" and "Water" classes. In this way, the local influence of the forest in the other classes is reduced. As for the water areas, since they are narrow, their neighborhoods will have many low probability values, which would reduce the expected value of the Bayesian estimator.

4. Comparison with other methods

Conclusion

Belgiu M, Dragut L (2016). "Random Forest in Remote Sensing: A Review of Applications and Future Directions." *ISPRS Journal of Photogrammetry and Remote Sensing*, **114**, 24–31.

Ghimire B, Rogan J, Miller J (2010). "Contextual Land-Cover Classification: Incorporating Spatial Dependence in Land-Cover Classification Models Using Random Forests and the Getis Statistic." *Remote Sensing Letters*, **1**(1), 45–54.

Gong P, Howarth PJ (1989). "Performance Analyses of Probabilistic Relaxation Methods for Land-Cover Classification." *Remote Sensing of Environment*, **30**(1), 33–42. ISSN 0034-4257. doi:10.1016/0034-4257(89)90045-X.

Hanson KM (1993). "Introduction to Bayesian Image Analysis." In *Medical Imaging 1993: Image Processing*, volume 1898, pp. 716–731. SPIE. doi:10.1117/12.154577.

Huang X, Lu Q, Zhang L, Plaza A (2014). "New Postprocessing Methods for Remote Sensing Image Classification: A Systematic Study." *IEEE Transactions on Geoscience and Remote Sensing*, **52**(11), 7140–7159.

Ma L, Liu Y, Zhang X, Ye Y, Yin G, Johnson BA (2019). "Deep Learning in Remote Sensing Applications: A Meta-Analysis and Review." *ISPRS Journal of Photogrammetry and Remote Sensing*, **152**, 166–177. ISSN 0924-2716. doi:10.1016/j.isprsjprs.2019.04.015.

Mountrakis G, Im J, Ogole C (2011). "Support Vector Machines in Remote Sensing: A Review." *ISPRS Journal of Photogrammetry and Remote Sensing*, **66**(3), 247–259.

Schindler K (2012). "An Overview and Comparison of Smooth Labeling Methods for Land-Cover Classification." *IEEE transactions on geoscience and remote sensing*, **50**(11), 4534–4545.

Simoes R, Camara G, Queiroz G, Souza F, Andrade PR, Santos L, Carvalho A, Ferreira K (2021). "Satellite Image Time Series Analysis for Big Earth Observation Data." *Remote Sensing*, **13**(13), 2428. doi:10.3390/rs13132428.

Spiegelhalter D, Rice K (2009). "Bayesian Statistics." *Scholarpedia*, **4**(8).

Wu C, Du B, Cui X, Zhang L (2017). "A Post-Classification Change Detection Method Based on Iterative Slow Feature Analysis and Bayesian Soft Fusion." *Remote Sensing of Environment*, **199**, 241–255. ISSN 0034-4257. doi:10.1016/j.rse.2017.07.009.

Affiliation:

Gilberto Camara
National Inst for Space Research, \ Brazil
Avenida dos Astronautas, 1758
12227-001 Sao Jose dos Campos, Brazil
E-mail: gilberto.camara"'inpe.br

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd