A Support Vector Machine Pixel Classifier Applied to Radar Sounder Data

Jordy Dal Corso
Department of Engineering and Computer Science
University of Trento
Trento, Italy
jordy.dalcorso@unitn.it

***Abstract*—Radar sounders (RS) are nadir-looking satellite instruments that acquire information on the subsurface of planetary bodies by transmitting electromagnetic pulses and processing the received echoes. Different models have beed developed for the classification of pixels within radar sounder data products and they are mainly based on machine or deep learning. In the field of radar sounder data analysis, very few authors provided open source software implementation of their works and this lack of reproducibility poses a challenge to validate, test and enhance novel methodologies. With this report, we attempt at replicating the SVM-based pixel classifier presented in [3] and we further analyze the model by performing feature selection and an analysis of the computational time of different SVM kernels. We test our replication on the dataset from the radar sounder MCORDS1 provided by [3].**

***Index Terms*—Radar sounder, support vector machine, SVM, classifier, supervised**

I. Introduction

There is a growing importance of RS in the field of planetary exploration. They are particularly because they are one of the few ways to investigate the subsurface of earth and other planetary bodies. The science products obtained by RS are especially useful for understanding the past history of planets, their recent and past geological activity and the composition of the uppermost rocky layers. Moreover, RS appear very important in mapping the subsurface of icy scenario like Antarctica and Greenland. The data obtained by RS can be aggregated in products called *radargrams*, which are 2D matrices where each column (or *rangeline*) corresponds to the echo of a chirp transmitted toward the target and received by the radar, sampled at the ADC frequency. Chirps penetrate the subsurface of target planetary bodies and their echo contains information about the dielectric discontinuities and the different materials which composes the subsurface, hence the radargram ends up being a depiction of a slice of the subsurface scenario.

As the number of operating remote sensing (RS) systems increases, so does the volume of data generated. Consequently, there is a greater demand for automated methods to analyze this data. Early automatic methods mainly focused on tracking the surface and the bedrock layer in icy scenarios [1]. A different perspective was investigated by [3], in which authors presented a pixel-wise classification method to obtain a full segmentation of the input radargrams. Segmentation maps are more informative than layer tracking because they allow to fit distributions on pixels of target areas and eventually individuate hidden subsurface formations. However, the Support Vector Machine (SVM) classifier presented in [3] is based on hand-crafted predictors and no public code is available to replicate the work, thus preventing the application of the method to new datasets and possible enhancements. With an effort to overcome these limitations, with this short paper we implement the SVM classifier of [3] and we replicate its results on a dataset obtained from the radar sounder MCoRDS1 [4]. We write codes to compute the hand-crafted features of a given arbitrary radargram, to apply the SVM classifier using these features and to test the trained model on newly acquired data. Our contribution is twofold. Firstly, with our implementation we allow researchers to replicate the work of [3], compare it with other new methods from the literature and enhance it. Secondly, we verify the importance of each hand-crafted feature. The rest of the paper is organized as follows. Section II introduces the work of [3] with a particular focus on how to compute the hand-crafted features. Section III presents the experiments we made to validate the model and further ablations on feature selection. We draw conclusions in Section IV.

II. Methodology

We define a radargram as an array $R \in \mathbb{R}^{N \times M}$ where $N \in \mathbb{N}^+$ is the number of samples per rangelines and $M \in \mathbb{N}^+$ is the number of rangelines. Each entry of R represents the intensity of the received echo at a particular time instant. We show an example of radargram from the radar sounder MCoRDS in Figure **??** with annotations on the main features. Our objective is the *semantic segmentation* of the radargram. In particular, given a set of $n_C \in \mathbb{N}^+$ semantic classes, given arbitrary indices $i \in \{1, ..., N\}$, $j \in \{1, ..., M\}$, we aim at finding a segmentation map $S \in \{1, ..., n_C\}^{NxM}$ where the entry (i, j) of S is the class of pixel (i, j) of the radargram R. The classes proposed by [3] are *free space*, *ice layers*, *bedrock line* and *noise*. This is a common choice when dealing with an icy scenario. The free space is the space composed by air which is at the top of radargrams and it is usually the easiest class to detect: it is sufficient to track the surface line (i.e. the brightest return of each rangeline) and set to free space every pixel above the surface. The ice layers class is

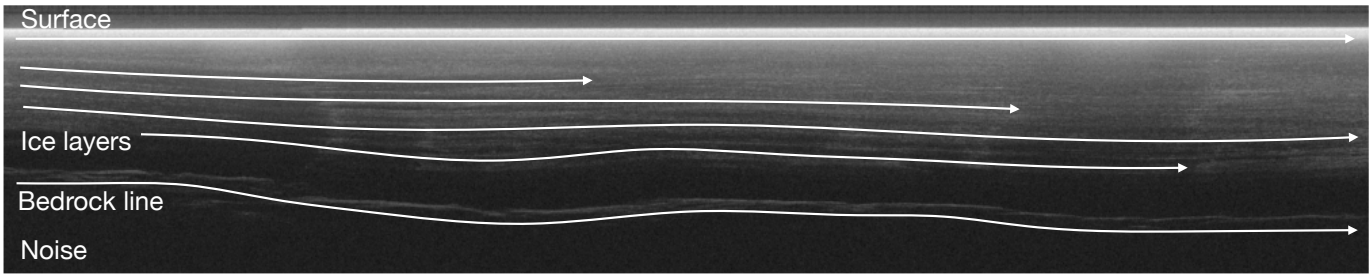

Fig. 1. Example of radargram obtained by MCoRDS1 in Antarctica. Columns are rangelines and rows represents ADC samples of each echo. We can see how it represent a slice of the subsurface scenarion and how bright features develop horizontally within the radargram. The surface is the brightest return and it is at the top of the radargram. Ice layers can be found right under the surface line. The bedrock line can be seen in the lower part.

commonly right under the surface and comprises the layers of ice created by the accumulation and freezing of snow during each year. The seasonality of ice is depicted within this class as multiple bright layers. The bedrock line class is the second brightest return after the surface since it comprises the dielectric discontinuity between the ice and rock. It is usually surrounded by the noise class since in the lower part of the bedrock the echo is too weak to return to the radar and right above the bedrock line there is the so called Echo Free Zone (EFZ).

In order to obtain the segmentation S, authors of [3] employ an SVM-based pixel classifier $\mathcal{M}_\theta : \mathbb{R}^{n_F} \rightarrow \{1, ..., n_C\}$ such that, given a vector $\mathbf{z} \in \mathbb{R}^{n_F}$ of $n_F \in \mathbb{N}^+$ features representing the pixel (i, j) of R, we have:

$$\mathcal{M}_\theta(\mathbf{z}) = \hat{s} \tag{1}$$

where $\hat{s} \in \{1, ..., n_C\}$ is the prediction for the class of pixel (i, j), i.e. $S(i, j)$. A core point in the aforementioned formulation is the derivation of the feature vector $\mathbf{z}$. In [3] $n_F = 7$ features are employed and we briefly introduce them in the following paragraphs. A number of features is computed employing the intensities of the neighboring pixels hence we define the neighborhood A_x of a radargram pixel x as the pixels within the rectangle of dimension $W_r \times W_a$ surrounding x.

a) Intensity: The first feature is the intensity of each pixel. It consists in the entry R(i,j) itself, as the intensity of each pixel is the only non-derived information we have when operating on the radargram. It is common to attribute the noise class to low intensity pixels, i.e. when the intensity of the returning echo does not surpass the instrument thermal noise floor. Conversely, high intensity pixels are considered *signal*, thus potentially a class different from noise.

b) Depth: We define the depth for any pixel of each rangeline as a number $d \in [0, 1]$ where $d = 0$ and $d = 1$ correspond respectively to the top and the bottom of the radargram. In other terms, we are associating to each pixel its row number (up to a normalization constant equal to N). The depth is an important feature for a radargram pixel due to the radar scenario of acquisition. Classes have a fixed vertical order dictated by physics, where free space is on top and the bedrock is at the bottom.

c) Entropy: The entropy associated to a pixel x is defined as the entropy of its neighborhood, which is:

$$E(x) = \sum_{y \in A_x} p(y) log(p(y)) \tag{2}$$

As stated in [3], the entropy is good at characterising the level of noise within a neighborhood. An high entropy is associated to high variations of intensities within the neighborhood.

d) Parameters of the best fitting distribution: In **??**, authors found that Gamma distributions fit well the various classes within MCoRDS1 radargrams. Hence they define as features the parameters α and β of the Gamma distribution fitting the neighborhood of each pixel.

e) Noise KL Divergence: A Gamma distribution is fit to a large portion of noise pixels (i.e. the lower part of the radargram) and for each pixel of the radargram they compute as a feature the KL divergence KL between the Gamma fitting the pixel neighborhood and the noise Gamma. The divergence between the two can be found analytically as:

$$D_{KL}\left(\Gamma(\alpha_1, \beta_1) | \Gamma(\alpha_2, \beta_2)\right) = a + b + c + d \tag{3}$$

$$\begin{aligned} a =& \alpha_1 \ln\left(\frac{\beta_1}{\beta_2}\right) \\ b =& (\alpha_2 - \alpha_1)\psi(\alpha_1) \\ c =& -\ln\Gamma(\alpha_2) \\ d =& \frac{\beta_1}{\beta_2} \end{aligned}$$

This feature represent how a pixel neighborhood is different from the noise distribution, hence how a pixel is likely not noise if its neighborhood is different from the noise distribution.

f) Relational feature: A relational feature depicts the vertical relationships existing between classes. For example, by physical constraints we are certain that the free space is on top of ice layers in radargram acquired on an icy scenario. Moreover, the bedrock is in the lower part of the radargram and it is surrounded by noise. In order to represent these relationships, authors of [3] threshold the KL divergence feature KL in order to obtain a binary map KL_{bin} as:

$$\text{KL}_{\text{bin}} = \begin{cases} 1 & \text{if} \quad KL > \text{thr}_{\text{KL}} * \mu_{\text{noise}} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Then, given a rangeline $\mathbf{r} \in \mathbb{R}^N$, they compute the relational feature Rel of each pixel i as:

$$\text{Rel}(\mathbf{r}(i)) = \text{cumsum}\left(\text{KL}_{\text{bin}}(\mathbf{r})\right)(i) \tag{5}$$

Hence the relational feature of each pixel is the cumulative sum of the binary KL divergence of pixels appeared previously in the same rangeline.

Given these features, in the next section we briefly introduce the dataset we utilised for our experiments and the settings we employed for the SVM classifier.

III. Experiments

A. Dataset

As dataset, we use the same utilized by [3]. In particular, we have a radargram of $M = 27350$ rangelines and $N = 410$ samples per rangeline. The radargram has been obtained by the radar sounder MCoRDS during a campaign in Antarctica in 2010. We compute the previously introduced features for the radargram, obtaining, for each pixel $x \in R$, a representation $\mathbf{z} \in \mathbb{R}^7$ where $\mathbf{z} = (\text{Int}, \text{Depth}, \text{Ent}, \alpha, \beta, \text{KL}_{\text{noise}}, \text{Rel})$.

B. SVM Classifier

As previously mentioned, we use a pixel-wise SVM classifier to classify each pixel of the radargram R into the correct class. We do not report here the theory of SVM but readers can see e.g. [2] for a complete chapter on the topic. We implement our classifier in Python following the definition of hyperparameters provided by [3]. In particular, we use the RBF kernel and we set $C = 100$ and $\gamma = 2$. The parameter C tunes the level of regularization of the model. An high C leads to low regularization, hence strict boundaries between classes and consequently less generalization capabilities, but higher precision. The parameter γ is related to the shape of the RBF kernel.

C. Experimental schedule

In order to replicate the results of [3], we select randomly the 1% of pixels from the dataset, after removing the first 50 rows in order to remove the free space class, whose classification is trivial. We obtain a dataset of:

$$\#\text{pixels} = ((410 - 50)) \times 27350)/100 = 98460$$

which is $\sim$ 98460 pixels with their respective features as predictors and class ground truths provided by [3]. We end up with a problem with $n_C = 3$ classes, which are *Ice*, *Bedrock* and *Noise*. We further reduce the number of training samples using a $20 - 80$ train-test split. We do K-fold cross validation with $K = 5$ to validate the performance of the model. We test the model with both RBF (as per [3]) and Linear kernel. The latter leads to faster computation time but less accuracy. We assess our results quantitatively by showing the precision, recall and F1 score of each class, as well as the overall accuracy of the classification.

We further analyze the performances of the two kernels by fixing the hyperparameters and training the SVM on a growing number of training samples. We train the model on percentages $p \in \{0.01, 0.05, 0.10, 0.25, 0.50, 0.75, 1.00\}$ of the total dataset, where the higest value 1.00 corresponds to training on the full 1% of pixels randomly extracted from the radargram (as per [3]).

TABLE I
MEAN CLASSIFICATION METRICS ON THE K-FOLD TRAINING SETS USING THE RBF KERNEL

Class	Precision	Recall	F1
Ice	1.00	1.00	1.00
Bedrock	1.00	0.93	0.96
Noise	0.99	0.99	0.99
Accuracy			0.99

TABLE II
MEAN CLASSIFICATION METRICS ON THE K-FOLD TEST SETS USING THE RBF KERNEL

Class	Precision	Recall	F1
Ice	1.00	1.00	1.00
Bedrock	0.96	0.89	0.93
Noise	0.98	0.99	0.99
Accuracy			0.98

We conclude our experiments by performing feature selection on the 7 features presented by [3], with the aim of validating their findings and individuating the most explanatory features. We perform backward feature selection by starting with a model with all the predictors and we iteratively remove the predictor whose removal lead to the best bedrock F1 score on the test set. We rely to this score as criterion since it encapsulate performance on the most challenging class.

D. Results

Quantitative results obtained using the RBF kernel are shown in Table I and Table II. In particular, we see that there is a very small mismatch in performance between train and test. Main flaws occur in classifying the bedrock class,

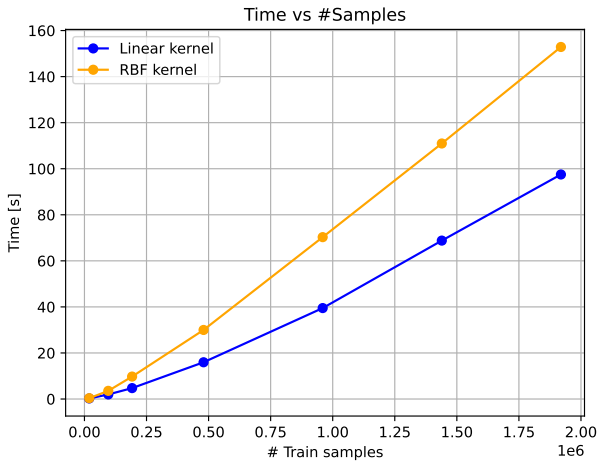

Fig. 2. Time elapsed during model fitting on training sets with different number of samples. The comparison between the two kernels clearly shows how the linear kernel is faster but the comparisons

TABLE III
MEAN CLASSIFICATION METRICS ON THE K-FOLD TRAINING SETS USING THE LINEAR KERNEL

Class	Precision	Recall	F1
Ice	1.00	1.00	1.00
Bedrock	0.96	0.75	0.84
Noise	0.97	0.99	0.98
Accuracy			0.99

TABLE IV
MEAN CLASSIFICATION METRICS ON THE K-FOLD TEST SETS USING THE LINEAR KERNEL

Class	Precision	Recall	F1
Ice	1.00	0.99	0.99
Bedrock	0.93	0.75	0.83
Noise	0.97	0.99	0.98
Accuracy			0.98

which is the most challenging one due to its low support and high irregularity. Results are in accordance with the ones obtained by [3] with a very similar setting. The results obtained using the Linear kernel are shown in Table IV and Table IV. As we can see, performances are worse than the RBF kernel, especially on the challenging bedrock class. Anyway, the overall K-Fold accuracy is 0.99 ± 0.001 for the RBF kernel and 0.98 ± 0.001 for the Linear kernel, with no particular variations within classes.

Regarding the comparison between the linear and RBF kernels, we plot the results of our ablation in Figure **??**. As expected, the linear kernel is faster compared to the RBF as the number of training sample grows, despite its lower performance on the bedrock class. This opens up a problem in choosing which kernel to use: there is clearly a trade-off between computation time and performance and practitioners may go for the linear kernel in the case of scarce computational resources.

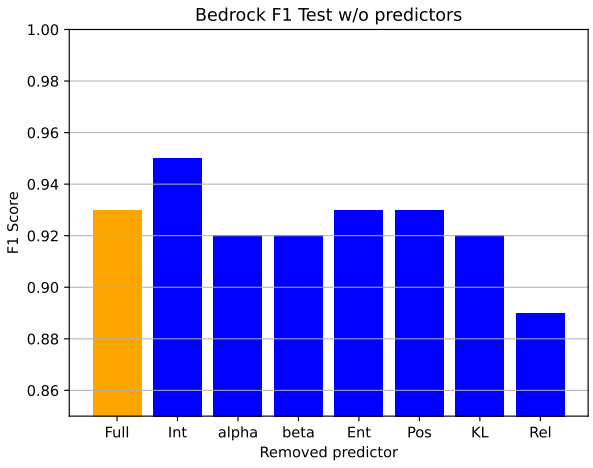

Fig. 3. Results of the first step of backward selection. We compare the Bedrock F1 of the full model and the models with a predictor removed. We can see how the model performs better without the bare intensity and performs way worse without the relational feature.

Lastly, results on feature selection are shown in Figure 3. We show only the first step of backward selection since further steps lead to a too high deterioration of the evaluation metrics. We can see how the removal of the relational predictor leads to the worst performance. This is probably because the predictor is very informative in terms of vertically discriminating the different classes. Interestingly, we see an increase in performance when removing the pixel intensity. We interpret this as the capacity of the other derived predictors to encapsulate the correct information to perform segmentation. Apparently the bare intensity does not help at discriminating well the ice and bedrock classes.

IV. CONCLUSIONS

With this work, we succesfully replicated the method presented by [3]. In particular, we developed our software implementation using Python and we obtained very similar results with respect to the original work. Furthermore, we analysed the computational training time in terms of the SVM kernel utilized and we performed backward feature selection to rank the importance of predictors. We assessed that speed of using the linear kernel instead of the RBF one, despite a low degradation of performance. This goes in accordance with the SVM theory and our expectations, highlighting the trade-off between accuracy and computational training time. Moreover, the results of feature selection shows how the intensity predictor becomes irrelevant when employing the other derived predictors, and how removing the relational feature strongly degrade performance. We plan to use this implementation to compare our future works with [3], further validating their performance and possibly using the hand-crafted features presented by authors to enhance predictions of our models.

REFERENCES

[1] Christopher M Gifford, Gladys Finyom, Michael Jefferson, MyAsia Reid, Eric L Akers, and Arvin Agah. Automated polar ice thickness estimation from radar imagery. *IEEE Transactions on Image Processing*, 19(9):2456–2469, 2010.

[2] Trevor Hastie, Robert Tibshirani, Jerome Friedman, Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Support vector machines and flexible discriminants. *The elements of statistical learning: data mining, inference, and prediction*, pages 417–458, 2009.

[3] Ana-Maria Ilisei and Lorenzo Bruzzone. A system for the automatic classification of ice sheet subsurface targets in radar sounder data. *IEEE Transactions on Geoscience and Remote Sensing*, 53(6):3260–3277, 2015.

[4] Fernando Rodriguez-Morales, Sivaprasad Gogineni, Carlton J Leuschen, John D Paden, Jilu Li, Cameron C Lewis, Benjamin Panzer, Daniel Gomez-Garcia Alvestegui, Aqsa Patel, Kyle Byers, et al. Advanced multifrequency radar instrumentation for polar research. *IEEE Transactions on Geoscience and Remote Sensing*, 52(5):2824–2842, 2013.