SF2A 2019
P. Di Matteo, O. Creevey, A. Crida, G. Kordopatis, J. Malzac, J.-B. Marquette, M. N'Diaye, O. Venot (eds)

DEEP LEARNING FOR THE SELECTION OF YOUNG STELLAR OBJECT CANDIDATES FROM IR SURVEYS

D. Cornu[1] and J. Montillaud[1]

Abstract. The robust identification of YSOs is an important step for characterizing star-forming regions. Such classification is commonly performed with infrared suveys using straight cuts in CMD diagrams. However, Machine Learning algorithms may outperform these methods with adaptive and non-linear separations in any number of dimensions. In this paper we present our methodology to implement a supervised deep neural network for YSO classification with various datasets built from well-known regions. We detail the tuning of the network parameters, taking into account the specificities of this classification. Then we focus on the reliability of the classification and address difficulties due to the strong dilution of YSOs against contaminants.

Keywords: Young Stellar Objects, Spitzer, Infrared, Machine Learning, ANN, Classification, Protostars

1 Introduction

Observing Young Stellar Objects (YSOs) in stellar clusters can be used to characterize star-forming regions. Their presence testifies the star-formation activity and allow one to quantify its efficiency. They can also be a probe of cloud density which can be combined with astrometric surveys like Gaia to recover the 3D structure and motion of star forming clouds (Grossschedl et al. 2018). They are often identified with a classification method that mainly rely on their Spectral Energy Distribution (SED) in the infrared (IR). One can also separate them in evolutionary steps from the star-forming phase to the main sequence (class 0 to III), granting even more information on the structure and evolution of star-forming regions.

In this context we aim to design a classification method for YSOs, relying on large surveys and taking advantage of modern statistics analysis. We choose to use Artificial Neural Networks (ANN) which are popular supervised machine learning methods, and have shown strong performance on a wide variety of problems. Since it belongs to supervised methods we rely on an original classification that provides targets for the training phase. For this we use the popular method by Gutermuth et al. (2009) which describes a multi-step classification scheme using the *Spitzer* space telescope (Werner et al. 2004) surveys (using 5 bands between 3 and 24 μm) . In this study we focus on bright star-forming regions observed with Spitzer whose physical properties are well constrained. In this paper we show results based on a combination of three different dataset, namely, Orion (Megeath et al. 2012), NGC 2264 (Rapson et al. 2014), and a sample of clouds nearer than 1 kpc that exclude the two previous regions (Gutermuth et al. 2009). We briefly describe our implementation of a Multi Layer Perceptron (MLP), a kind of ANN, with a special emphasis on how to properly link the physical problem to the network. Therefore, we detail common good practices for the data preparation and a proper representation of results.

2 Deep Learning method

2.1 Deep Artificial Neural Network : Multi Layer Perceptron

The core element of ANN is a mathematical model of neuron (McCulloch & Pitts 1943) that performs the weighted sum h of an input vector. It is then given to an activation function $g(h)$ which usually resembles a

[1] Institut UTINAM - UMR 6213 - CNRS - Univ Bourgogne Franche Comté, France, OSU THETA, 41bis avenue de l'Observatoire, 25000 Besançon, France - email : david.cornu@utinam.cnrs.fr

binary activation. One usual activation function is the Sigmoïd : $a = g(h) = 1/(1 + \exp(-\beta h))$, where a is the result of the activation and β an hyperparameter of the network. One neuron alone only performs a linear separation. Therefore, they can be added in the form of a *deep* ANN. For this, neurons can be added in two different ways. The first one is by adding neurons independently in a layer. Each neuron is then fully connected to the input layer with no connection between neurons of the same layer. The second one is to add multiple layers on top of each other. In this case a layer will take as its own input vector the result of the activation of the neurons from the previous layer.

Such an ANN is a universal function approximator (Cybenko 1989), called a Multi Layer Perceptron (MLP) (Rumelhart et al. 1986), which is able to perform classification, regression, clustering, time series prediction, compression, image recognition, ... ANN are supervised machine learning methods, therefore they need to be trained on previously labeled data. To learn, the network will compare its own output with the expected output using an error function and will modify its weights accordingly. To do so one has to propagate the error from the output layer back to the input layer by using the well-known "Back-Propagation" algorithm. The general formula of this gradient computation is described as follows:

$$\omega_{ij} \leftarrow \omega_{ij} - \eta \frac{\partial E}{\partial \omega_{ij}} \qquad \frac{\partial E}{\partial \omega_{ij}} = \frac{\partial E}{\partial h_j}\frac{\partial h_j}{\partial \omega_{ij}} \qquad \delta_l(j) \equiv \frac{\partial E}{\partial h_j} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial h_j}. \tag{2.1}$$

The indices i and j go through the input dimensions and the neurons respectively, for the current layer. These equations are the same for each layer with adjustment to the size of the corresponding layer and associated inputs. δ_l is a local error term that can be defined for each layer. The η parameter is a learning rate, that must be fine tuned to allow both fast convergence and precision of the generalization. The final network contains only one hidden layer with 25 hidden neurons and a learning rate $\eta = 7 \times 10^{-5}$. These numbers are found based on performance criteria (Cornu et. al in prep.).

2.2 Classification with ANN

The problem we want to solve is a YSO classification. Therefore, the output layer is set with one neuron per class. The expected output for a 3-class example would be in the form of : *A: (1, 0, 0), B: (0, 1, 0), C: (0, 0, 1).* Combined with an appropriate activation function it allows one to recover a membership probability (based on the network knowledge) for each class. We use the common Softmax function : $a_k = g(h_k) = exp(h_k)/(\sum_{k=1}^{N} exp(h_k))$, where k is the number of neurons in the output layer. The result of each neuron will always be between 0 and 1, and the sum of all the output neurons will always equal 1. The object is then classified regarding the neuron with the maximum output. An example of results given by the network is: (0.1, 0.05, 0.85) for a rather clear class C, and (0.3, 0.4, 0.3) for a way more confuse object still considered as class B. This brings the network in the range of Probabilistic Neural Network (PNN) (Specht 1990).

3 Data preparation and network settings

3.1 Definition of the classes and labeled sample

We construct our training sample based on a simplified version of the method by Gutermuth et al. (2009) (hereafter G09), where only Spitzer data are used. We use the $3.6, 4.5, 5.8, 8$ μm bands from IRAC and the 24 μm band from MIPS combined with their respective uncertainty as input features. The G09 method performs a multi-step extraction based on straight cuts in color-color and color-magnitude diagrams (CMDs), which allows one to extract objects that are likely contaminants and to recover YSO candidates (Fig. 1). YSO classes range from class 0, that are the youngest one, up to class III that are near the main sequence (Allen et al. 2004). However, using the previous bands we are only able to recover class I (protostars) and class II (pre-main sequence with thick disk) YSOs. For the sake of simplicity we adopt only 3 categories in our classification : YSO CI, YSO CII, and Other/Contaminants.

To get the maximum generalization capacity out of our network we need to ensure the diversity of our training sample. For that we choose to use well-known and well constrained star forming regions observed with Spitzer : Orion (Megeath et al. 2012), NGC2264 / Mon OB1 (Rapson et al. 2014), and a sample of many regions nearer than 1 kpc (Gutermuth et al. 2009). This is a major point since different star forming regions cover the input feature space differently and we want our network to be able to make predictions on various regions. We define our "labeled sample" by applying the G09 method on these catalogs (Table 1).

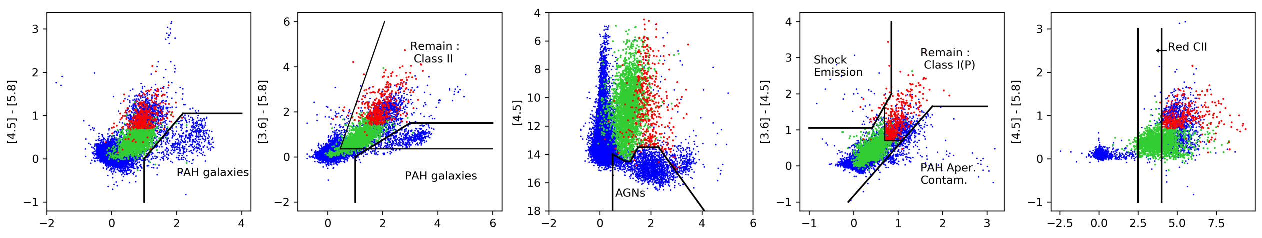

Fig. 1. Usual CMDs used for this classification. Retrieved class II candidates in green, class I candidates in red and others in blue for the Orion cloud.

Dataset	Pre-selection		Detailed contaminants					Output classes		
	Total	Selected	Gal.	AGN	Shocks	PAH	Stars	YSO CI	YSO CII	Other
Labeled sample	311407	29074	522	1448	34	89	21799	784	4396	23897
Test		5377	104	278	6	17	4359	82	531	4764
Train γ_i:			0.5	0.7	0.05	0.15	4.0	1.0	6.3	
Train		6286	331	463	27	70	2648	662	2085	3539

Table 1. Result of the application of our simplified G09 method. The third category of columns give the labels in use in the learning phase with the last "Other" column being the sum of all the detailed contaminants. The bottom lines shows the test and training sets distribution, along with the γ_i corresponding values for the train set.

3.2 Data separation : training and test sets

Most of the labeled dataset must be used for the learning process (training set), but we also need to keep aside a smaller part of it to assess the quality of the prediction of our network with objects it has never seen before (test set). This test set must be as representative as possible of the true problem we want to solve. It means that it must preserve observational proportions. Without this precaution, the results would only represent a numerical performance and not the capacity of the network to generalize a problem. We define a parameter θ so that for each subclass the number of objects kept away in the test set is $N_i * \theta$. For the results in this paper $\theta = 0.2$.

In contrast, the training set does not need to have observational proportions. Actually, re-balancing the number of objects given for each subclass is a way to force the network to give more representative strength to some subclass. It is especially important due to the strong dilution of our labeled sample. With observational proportions, contaminants (especially more evolved stars) would be so numerous than YSOs would be considered as noise. We then have to reduce the number of contaminants in our training sample, but we can not choose an equal ratio. Doing so would not take into account that some subclass of contaminants cover a larger input feature space than the YSOs, or require a more complex separation to be distinguished. Moreover, we need to keep the majority of our representative strength for the most dominant subclass as they will be the main factor of contamination of the other classes (as illustrated by the results in Sect. 4). Still, there are no exact best solutions for those proportions as it depends on what one wants to recover. We choose to put the emphasis on class I YSO candidates, since they are not recovered by many other methods, but with the objective of preserving the quality of class II candidates. It means that the number of class I YSOs in the training sample has to be maximized. The other objects are proportional to the number of class I which defines a proportion factor γ_i for each of them. The proportions we found to provide good results are presented in the bottom of Table 1. More details on this selection in Cornu et al. (in prep.).

4 Results

After the training, we can apply our network on the test set to recover the predicted result for each object for which we also have the expected answer. But to assess the quality we need a proper way to represent them. For classification it is recommended to use a Confusion Matrix. It is defined as a two dimensional table with rows corresponding to the original class distribution (labels/targets), and columns corresponding to the classes given to the same objects by our network classification (output). The "recall" represents the proportion of

Actual \ Predicted: Class	YSO CI	YSO CII	Other	Recall
YSO CI	75	3	4	91.5%
YSO CII	6	515	8	97.0%
Other	8	42	4714	99.0%
Precision	84.3%	92.0%	99.7%	98.6%

	CI	CII	Gal	AGN	Shock	PAH	Stars
YSO CI	75	6	0	2	2	2	2
YSO CII	3	515	2	2	4	3	31
Other	4	10	101	274	0	12	4327

Table 2. Left: Confusion matrix for the test set (observational proportions). **Right:** Subclass distribution from the G09 original attribution for each of the output of the network.

objects from a given class that were recovered as expected and is given at the end of each line. The "precision" represents the fraction of correctly classified objects in a class as predicted by our network and is given at the bottom of each column. And finally a global "Accuracy" quantity that gives the proportion of objects properly classified over the total number of objects given at the bottom right corner of the matrix. The corresponding matrix for our trained network applied on our test set in observational proportions is presented in Table 2.

These results show that we get more than 90% recall on the class I YSO candidates, but also by preserving a 97% recall on the class II candidates. The precision drops at 84% for class I. Looking at the detailed subclass results in Table 2, we can see that the contamination for class I is mainly due to the confusion with class II, while contamination for class II is mainly coming from the more evolved stars that are the vastly dominant class in the sample. Those results are highly competitive with previous studies (Marton et al. 2016; Miettinen 2018), especially regarding the precision that is greatly improved, due to our more detailed analysis and management of the training proportions.

5 Conclusions

We showed that, using a rather simple neural network, we could achieve excellent results on infrared YSO classification. However, such methods require meticulous preparation of the data and proper understanding of their generalization biases. Using such a network we aim at providing a wide catalog of Spitzer YSO candidates. Since our current limitation is the number of YSOs in our sample, we plan to use simulated data to try improving the current results. More detailed information on this research will be soon available in Cornu et al. (in prep.).

We thank PCMI (Programme National "Physique et Chimie du Milieu Interstellaire" of CNRS/INSU with INC/INP co-funded by CEA and CNES) for giving us the opportunity to present this research during their 2019 SF2A session, and for their financial support through the GALETTE project. D. Cornu acknowledge that this research is done under a PhD funding from the CNES and the "région Bourgogne Franche-Comté".

References

Allen, L. E., Calvet, N., D'Alessio, P., et al. 2004, The Astrophysical Journal Supplement Series, 154, 363

Cybenko, G. 1989, Mathematics of Control, Signals and Systems, 2, 303

Grossschedl, J. E., Alves, J., Meingast, S., et al. 2018, Astronomy & Astrophysics, 619, A106, arXiv: 1808.05952

Gutermuth, R. A., Megeath, S. T., Myers, P. C., et al. 2009, ApJS, 184, 18

Marton, G., Tóth, L. V., Paladini, R., et al. 2016, Mon Not R Astron Soc, 458, 3479

McCulloch, W. S. & Pitts, W. 1943, The bulletin of mathematical biophysics, 5, 115

Megeath, S. T., Gutermuth, R., Muzerolle, J., et al. 2012, AJ, 144, 192

Miettinen, O. 2018, Astrophysics and Space Science, 363, 197

Rapson, V. A., Pipher, J. L., Gutermuth, R. A., et al. 2014, The Astrophysical Journal, 794, 124

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. 1986, in Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, ed. D. E. Rumelhart, J. L. McClelland, & C. PDP Research Group (Cambridge, MA, USA: MIT Press), 318–362

Specht, D. F. 1990, Neural Networks, 3, 109

Werner, M. W., Roellig, T. L., Low, F. J., et al. 2004, The Astrophysical Journal Supplement Series, 154, 1