MNRAS 000, 1–16 (2015) Preprint 22 December 2015 Compiled using MNRAS LATEX style file v3.0

Exploring the spectroscopic diversity of type Ia supernovae with DRACULA: a machine learning approach

M. Sasdelli[1,2⋆], E. E. O. Ishida[2,3], R. Vilalta[4], M. Aguena[5], V. C. Busti[5], H. Camacho[5], A. M. M. Trindade[6,7], F. Gieseke[8], R. S. de Souza[9], Y. T. Fantaye[10], and P. A. Mazzali[1,2], for the COIN collaboration

[1] *Astrophysics Research Institute, Liverpool John Moores University, Liverpool L3 5RF, UK*
[2] *Max-Planck-Institut für Astrophysik, Karl-Schwarzschild-Straße 1, 85748, Garching, Germany*
[3] *Laboratoire de Physique Corpusculaire de Clermont-Ferrand, F-63171 Aubière Cedex, France*
[4] *Department of Computer Science, University of Houston, 4800 Calhoun Rd., Houston TX 77204-3010, USA*
[5] *Departamento de Física Matemática, Instituto de Física, Universidade de São Paulo, CP 66318, CEP 05508-090, São Paulo - SP, Brazil*
[6] *Instituto de Astrofisica e Ciências do Espaço, Universidade do Porto, CAUP, Rua das Estrelas, PT4150-762 Porto, Portugal*
[7] *Departamento de Física e Astronomia, Faculdade de Ciências, Universidade do Porto, Rua do Campo Alegre 687, PT4169-007 Porto, Portugal*
[8] *Institute for Computing and Information Sciences, Radboud University Nijmegen, Toernooiveld 212, 6525 EC Nijmegen, Netherlands*
[9] *MTA Eötvös University, EIRSA "Lendulet" Astrophysics Research Group, Budapest 1117, Hungary*
[10] *Department of Mathematics, University of Rome Tor Vergata, Rome, Italy*

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT

The existence of multiple subclasses of type Ia supernovae (SNeIa) has been the subject of great debate in the last decade. One major challenge inevitably met when trying to infer the existence of one or more subclasses is the time consuming, and subjective, process of subclass definition. In this work, we show how machine learning tools facilitate the automatic discovery of sub-populations of SNIa; to that end we introduce the `DRACULA` Python package (Dimensionality Reduction And Clustering for Unsupervised Learning in Astronomy). Our approach is divided in three steps: (i) Transfer Learning, which takes advantage of all available spectra (even those without an epoch estimate) as an information source, (ii) dimensionality reduction through Deep Learning and (iii) unsupervised learning (clustering) using K-Means. Results match a previously suggested classification scheme, showing that the proposed method is able to grasp the main spectral features behind the definition of such subclasses. Moreover, our methodology is capable of automatically identifying a hierarchical structure of spectral features. This allows the confirmation of the velocity of lines as a first order effect in the determination of SNIa sub-classes, followed by 91bg-like events. In this context, SN Ia spectra are described by a space of 4 dimensions + 1 for the time evolution of objects. We interpreted this as evidence that the progenitor system and the explosion mechanism should be described by a small number of initial physical parameters. Given the expected data deluge in the forthcoming years, our proposed approach is essential to allow a quick and statistically coherent identification of subclasses (and outliers). `DRACULA` is publicly available within COINtoolbox (https://github.com/COINtoolbox/DRACULA).

Key words: supernovae: general – methods: machine learning, data analysis, statistical

1 INTRODUCTION

Type Ia supernovae (SNe Ia) are extremely bright objects, exhibiting a good degree of spectroscopic and photomet-

⋆ E-mail: m.sasdelli@ljmu.ac.uk (MS)

ric homogeneity. Among other characteristics, the fact that their luminosity correlates with a set of distance independent quantities constructed using multi-band light curves is particularly relevant. These relations enable the use of SNe Ia as standard candles, and combined with their strong luminosity, allows us to probe large cosmological distances. This facilitated the discovery of the accelerating expansion of the Universe in the late 20$^{\text{th}}$ century (Riess et al. 1998; Perlmutter et al. 1999).

Although most SNe Ia are spectroscopically quite uniform, there is a significant fraction of spectroscopically peculiar objects (Li et al. 2001), some of which are very different from the average SN Ia. At this moment, it is still unclear if there exists different subclasses in the space of SN Ia spectra that are truly distinct (e.g Benetti et al. 2005) or if subclasses defined in the literature are just extremes of a continuum distribution of properties (e.g Blondin et al. 2012).

Parallel to such considerations derived from the analysis of observed spectra theoretical developments also investigate multiple hypotheses to explain SN Ia diversity. A large number of possible progenitor systems and explosion mechanisms have been proposed (e.g. Hillebrandt et al. 2013) leading to an agreement that the origin of the majority of SNe Ia lies in the thermonuclear runaway of a CO white dwarf in a binary system. Nevertheless, the nature of the companion and the explosion mechanism are still fiercely debated. Proving the existence of well-defined and distinct subclasses would strongly support hypothesis of different progenitor systems or, qualitatively, different explosion mechanisms.

Trying to identify which spectral feature(s) might carry the signature of physically distinct subclasses (if they exists), a number of different classification schemes have been proposed. SNe Ia have been classified into High and Low Velocity Gradient based on the time gradient of the velocity of the Si II 6355Å line (Benetti et al. 2005). They have also been classified into Shallow and Broad-Silicon classes according to the Equivalent Width (EW) of the Si II 6355Å line, and have been named Cool when the ratio between the Si II 5972Å and the Si II 6355Å is above a certain value at B-band maximum (Branch et al. 2009). Many of these classification schemes do not exhibit well-separated groups when applied to a large number of observations (Blondin et al. 2012). However, most of them are based on a very small subset of spectral features. The situation might be quite different if all the information contained in the spectra is taken into account. Additionally, SNe Ia are classified in spectroscopic subclasses after the first peculiar object of a certain kind (e.g. Li et al. 2001). For example, 91T-like for the similarity with SN 1991T before maximum, 91bg-like when they are similar to the faint SN 1991bg and 02cx-like when they are similar to the faint and hot SN 2002cx. This is a non-quantitative criterion that complicates the study of subclass definition.

One main driving force behind the development of classification schemes based on individual spectral features was the difficulty in obtaining a large number of high quality observations. Having only a few observed objects of each category, the only viable approach was to minutely study the observations at hand, extrapolating their characteristics to the entire SN Ia population. Nowadays, the situation is rapidly changing. In the last few years, data released from a number of observation campaigns increased the number of available spectra by at least an order of magnitude (Blondin et al. 2012; Silverman et al. 2012; Folatelli et al. 2013). In this new paradigm, we face a different challenge: to develop methods and tools capable of dealing with a large number of spectra at once. The overwhelming volume of data defies dependence on human inspection of individual spectra; the process must be automatized.

Fortunately, similar problems are at the core of machine learning research; such tools can be adapted to a large variety of tasks, as has been reported in other fields (see e.g. Crisci et al. 2012; Libbrecht & Noble 2015; Vidyasagar 2015). Following this trend, the present work is an additional effort to popularize modern machine learning techniques within astronomy (see Ball & Brunner 2010; Krone-Martins et al. 2014; Ivezic et al. 2014, and references therein).

In what follows, we describe a series of machine learning tools and demonstrate how they can help automatize the visualization and classification of a large set of SN Ia spectra. Our approach involves two steps: reducing the dimensionality of an initially very large space and subsequently using unsupervised learning (clustering) to automatically reproduce a classification system based on individual spectral analysis. On each step we use state of the art machine learning techniques that lead to powerful insights on questions underlying SN Ia spectral features. The tools used here are implemented in the `DRACULA` Python package (Dimensionality Reduction And Clustering for Unsupervised Learning in Astronomy) and are publicly available within the COINtoolbox[1].

This paper is organized as follows: section 2 describes the data used for our analysis; section 3 explains our approach to dimensionality reduction; section 4 demonstrates how transfer learning can be used in the context of SN Ia spectral analysis; section 5 shows the improvement in dimensionality reduction achieved by Deep Learning in comparison with PCA results; section 6 gives a brief overview of the methods used for data visualization; section 7 reviews the K-Means algorithm; section 8 describes the `DRACULA` package, where our proposed tools are implemented; section 9 goes over our main results. Lastly, section 10 gives summary and discussion. In order to avoid confusion between similar expressions with distinct meanings in the machine learning and astronomy communities, we provide a small glossary in appendix A with the definitions used throughout this paper.

2 DATA

We compiled a set of publicly available SN Ia spectra from a variety of sources: the Berkeley Supernova Program (Silverman et al. 2012), the CfA spectroscopic release (Blondin et al. 2012) and the Carnegie Supernova Project (CSP) (Folatelli et al. 2013). Spectra have been collected from the SUSPECT [2] and the WISEREP (Yaron & Gal-Yam 2012) repositories. CSP spectra are published in rest frame; the remaining spectra were deredshifted using heliocentric redshifts from Blondin et al. (2012).

In order to build the input data matrix, the spectra need

[1] https://github.com/COINtoolbox
[2] http://www.nhn.ou.edu/~suspect

to be smoothed, binned in a homogeneous wavelength window, and systematics must be taken into account. Here we follow the procedure used by Sasdelli et al. (2015), smoothing the spectra through the use of the Savitzky-Golay filter (Morrey 1968) and applying the derivative over wavelength to the logarithm of the measured flux. The Savitzky-Golay filter is effective in reducing the noise and, at the same time, preserving the shape of the features present in the spectra. The use of derivative spectropy allows us to remove the systematics due to the uncertainty in the distance determination and in the global spectrum calibration. A possible alternative to the Savitzky-Golay filter and derivative approach is the use of a wavelet decomposition[3], discarding the coefficients heavily affected by reddening and noise (Arsenijevic 2011). We plan to investigate this further in a future work.

Once the pre-processing is done, it is necessary to design the initial data matrix as input to the dimensionality reduction algorithms, taking into account the drastic changes in SN spectra with time and the non-ideal epoch coverage. Time sampling of SN data is highly irregular, having large periods without observations, particular at very early and very late epochs. In Sasdelli et al. (2015) this problem was dealt with by concatenating spectra along the same line in the data matrix, thus taking into account the time evolution of each object. This strategy presents promising results, but generates a matrix with a large fraction of missing data. We propose an alternative approach that allows us to exploit most of the available spectroscopic information (regardless of the epoch of observation) to attain a stable low dimensional space (section 4). Before addressing the effectiveness of our proposal, we review in the next section the dimensionality reduction techniques to be tested.

3 DIMENSIONALITY REDUCTION

After the data have been pre-processed, the first step is to translate it to a low dimensional feature space. We briefly describe below the two main dimensionality reduction algorithms used in this work: Principal Component Analysis (PCA) and Deep Learning (DL).

3.1 Principal Component Analysis

PCA is a method to reduce the dimensionality of a multivariate dataset, by projecting it onto a lower dimensional feature space. Given its versatility, PCA and variations of it have been applied to a broad range of astronomical studies (e.g., Ferreras et al. 2006; Ishida & de Souza 2011; Mitra et al. 2011; Ishida et al. 2011; Ishida & de Souza 2013; Benitez-Herrera et al. 2013; De Souza et al. 2014a,b; Sasdelli et al. 2015).

The principal components (PCs) are computed diagonalizing the covariance matrix (Σ^2), with the eigenvectors being the PCs and the eigenvalues indicating the fraction of total variance *explained* by their corresponding PCs. The first eigenvector (PC1 - the component associated with the largest eigenvalue) indicates the direction of greatest variance, the second eigenvector (PC2 - component with second largest eigenvalue) points to the second direction holding highest variance subjected to being orthogonal to PC1, and so on.

Mathematically, this can be described as follows: given Γ measured features $y_1, \ldots, y_\Gamma$, all of them column vectors of dimension n (1 for each object in the data set), the first PC is obtained by finding a unit vector $\mathbf{a}$ that maximizes the variance, S, of the data projected onto it:

$$\mathbf{a_1} = \underset{||\mathbf{a}||=1}{\arg\max}\ S^2(\mathbf{a}^t y_1, \cdots, \mathbf{a}^t y_\Gamma), \tag{1}$$

where t is the transpose operation and $\mathbf{a_1}$ is the direction of the first PC[4]. Once we have computed the $(k-1)$th PC, the direction of the kth component, for $1 < k \leqslant \Gamma$, is given by

$$\mathbf{a_k} = \underset{||\mathbf{a}||=1, \mathbf{a}\perp\mathbf{a}_1, \cdots, \mathbf{a}\perp\mathbf{a}_{k-1}}{\arg\max}\ S^2(\mathbf{a}^t y_1, \cdots, \mathbf{a}^t y_\Gamma), \tag{2}$$

where the condition that each PC must be orthogonal to all previous PCs ensures a new uncorrelated basis.

It is possible to show that the above is equivalent of calculating the eigenvalues and eigenvectors of the Σ^2 (Jolliffe 1986, chapter 1). Once the PCs are calculated, one can use the percentage of total variance encoded in the eigenvalues in order to determine the dimensionality of the new feature (or PC) space.

3.2 Deep Learning

The idea behind DL is to take the input data $\mathbf{x} = x_1, x_2, ..., x_n$ and represent it in the form of a layer of nodes – or neurons–, where each node is a variable x_i (see Fig. 1, bottom layer). Additional layers of neurons above the original input signal are built to ensure that each new layer captures a more abstract representation of the original input signal. In DL, each layer constructs new features by forming nonlinear combinations of the features in the layer below. This hierarchical approach to feature construction has been effective in disentangling factors of variation in the data (Hinton & Salakhutdinov 2006; Bengio et al. 2013; LeCun et al. 2015). DL has contributed to a rapid advancement in the field of neural networks through new mechanisms to train architectures made of many layers of intermediate neurons.

To illustrate these ideas, consider the task of image processing. Specifically, assume we have an image with thousands of pixels where we wish to recognize the object depicted in the image. We can represent the entire image as a feature vector $\mathbf{x} = (x_1, x_2, ..., x_n)$, where each pixel x_i is a variable or feature. The resulting space is not only very large; in addition, each pixel contains low-level information about the main object in need of recognition. In DL we make each pixel x_i stand as one node along the first layer of the neural network. The second layer is made of nodes computing nonlinear combinations of all nodes (pixels) below. The third layer captures nonlinear combinations of the nodes on the second layer, and so on. Each layer captures more abstract, global structures of the object under analysis. Starting with low-level pixel information, upper layers

[3] Already successfully applied to other subjects within astronomy (see e.g. Paykari et al. 2014).

[4] $\underset{y}{\arg\max}\ f(y)$ is the set of values of y for which the function $f(y)$ attains its largest value.

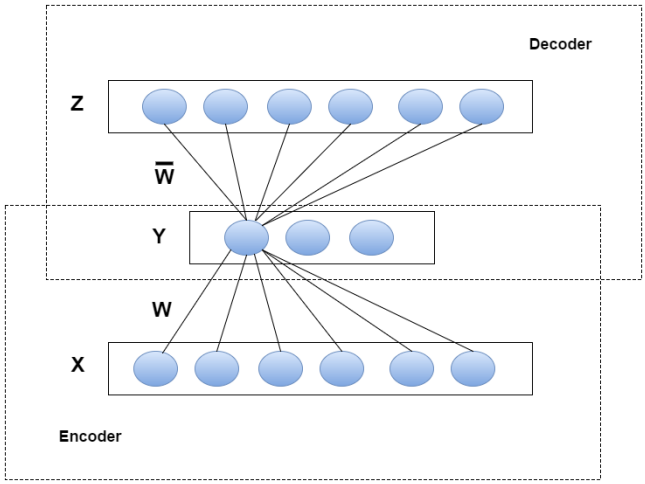

Figure 1. A simple autoencoder where the input **X** is reproduced in the ouptut layer **Z**. The middle layer **Y** "compresses" the input signal **X**, effectively reducing the dimensionality of the data.

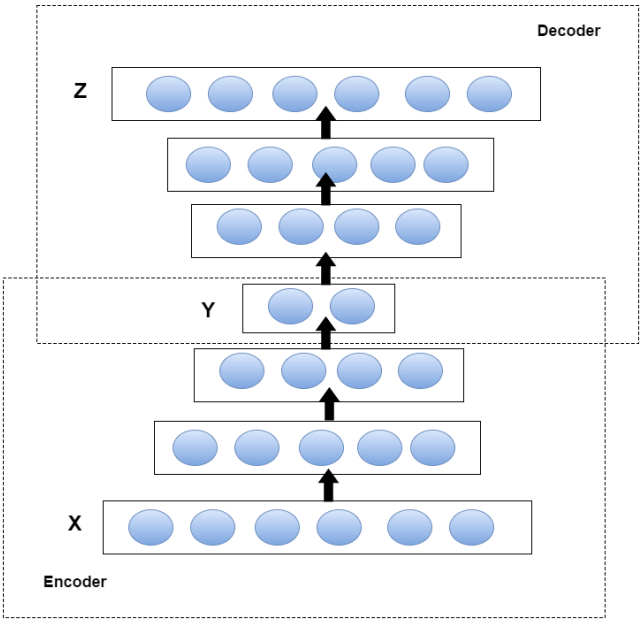

Figure 2. A deep autoencoder where intermediate layers provide increasingly more abstract representations of the input signal. The most abstract representation is stored in layer 4 (vector **Y**).

can gradually capture edges, motifs, and larger structures of the main object.

While different approaches exist to deal with DL architectures, we focus our attention on the problem of dimensionality reduction. In such unsupervised learning setting, auto-encoders have played a prominent role (Vincent et al. 2008). An illustration of a simple auto-encoder is shown in Figure 1. The goal here is to compress the input signal by a transformation that reduces the size of the feature space. Specifically, the first layer corresponds to input vector $\mathbf{x} \in \mathcal{R}^n$. The intermediate layer corresponds to a new vector $\mathbf{y} \in \mathcal{R}^d$, $d < n$, where each node computes a nonlinear combination of the input features. The last layer maps the internal representation back to the original dimensionality through a new vector $\mathbf{z} \in \mathcal{R}^n$, with the objective of reproducing the input vector $\mathbf{x}$ as best as possible, $\mathbf{x} \sim \mathbf{z}$. The weight parameters of the autoencoder are the weight matrices $\mathbf{W}$ and $\overline{\mathbf{W}}$ connecting nodes from one layer to the one above[5](see Fig. 1). The network is trained by adjusting the weights to minimize an error function that computes the distance between input and output: $\| \mathbf{x} - \mathbf{z} \|^2$. Specifically, the transformation from the first layer to the second layer "encodes" the input signal through a nonlinear transformation:

$$\mathbf{y_i} = f_i(\mathbf{x}) = \sigma(\mathbf{w_i}^t \mathbf{x}) \tag{3}$$

where $\mathbf{y_i}$ is one intermediate node, $\mathbf{w_i}$ is the weight vector (containing the bias term), and $\sigma(u)$ can vary in nature, a common choice being the sigmoid function $\sigma(u) = \frac{1}{1+e^{-u}}$. The new vector $\mathbf{y}$ is then "decoded" into a new vector $\mathbf{z}$ that reconstructs the input vector $\mathbf{x}$:

$$\mathbf{z_j} = g_j(\mathbf{y}) = \sigma(\overline{\mathbf{w_j}}^t \mathbf{y}) \tag{4}$$

[5] We assume weight matrices contain both weights and bias terms.

While learning to reproduce an input signal may appear as a trivial exercise, the interesting part of the auto-encoder is that the intermediate layer (vector $\mathbf{y}$) "abstracts" the representation of the input layer during the training phase, essentially compressing the input data through a combination of nonlinear representations. This is similar to the goal behind PCA, except here the combination of features is nonlinear, and there is no orthogonality constraint. Each of the intermediate nodes stands as a new variable in the reduced dimensionality space.

Notice that the autoencoder can be divided into two sections. The "encoder" section generates more compact representations (Fig. 1; layers 1 and 2), while the "decoder" section simply unfolds the compact representation in an attempt to reproduce the input signal (Fig. 1; layers 2 and 3).

3.2.1 Stacking Multiple Layers

The ideas above have been extended to "deep" architectures with many layers of neurons. Figure 2 shows an example of a deep autoencoder. Training such deep architectures can be done iteratively by stacking several auto-encoders in such a way that the intermediate layer of nodes becomes input to the next auto-encoder. Alternatively we can simply build a deep autoencoder directly, and let the optimization phase (gradient descent) look for the weight values that minimize the distance between input $\mathbf{x}$ and output $\mathbf{z}$. The net result is a vector $\mathbf{y}$ (middle layer) that in effect summarizes the input signal in a compact fashion. This technique was recently used by Huertas-Company et al. (2015) in the construction of a galaxy morphology catalogue, but its potential application in different areas of astronomy is still to be scrutinized.

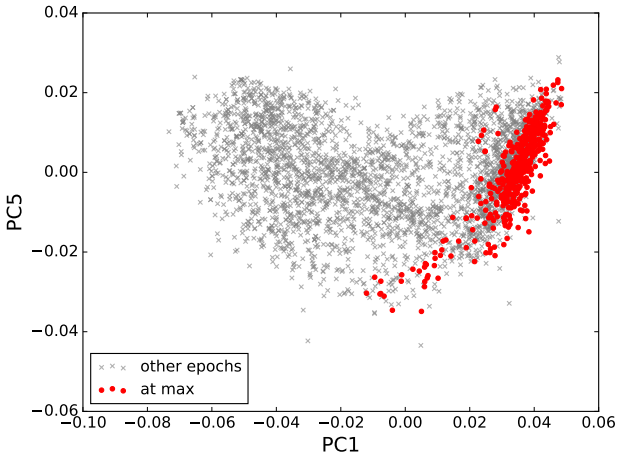

Figure 3. Distribution of spectra in the feature space formed by the 1^{st} and 5^{th} principal components. The red circles represent SN Ia spectra at maximum and the grey crosses denote SN Ia spectra in other epochs.

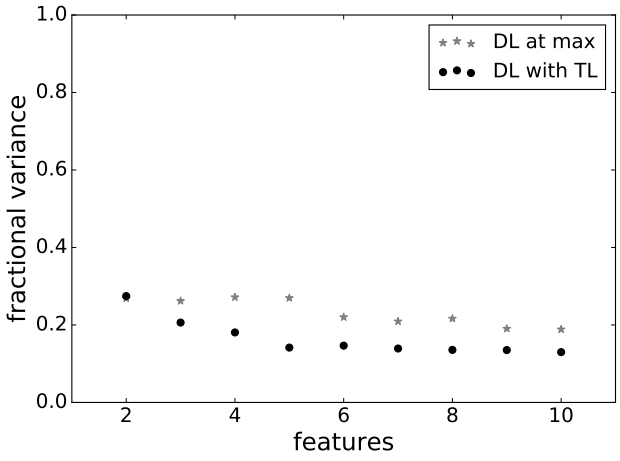

Figure 4. Variance between the deep learning reconstructions and observed SN Ia spectra at B max. Gray crosses correspond to a data configuration using only SNe Ia at maximum (no transfer learning) and black circles denote results from an initial data matrix containing spectra from all epochs (with transfer learning). The horizontal axis stands for the number of features; the vertical axis shows the deviation between real data and reconstruction.

This work represents the first effort to use DL techniques in the characterization of spectral features.

It is important to emphasize that unlike the eigenvectors from PCA, the elements of the middle layer within a deep network do not follow a natural ordering. Moreover, due to a random initialization of all weight parameters, there is no guarantee that the final result will be the same across runs. The power of DL resides on providing a compact representation of the initial data, preserving relevant information through layers of abstraction. This compact representation has extraordinary potential in characterizing the data, as will be made clear shortly.

4 TRANSFER LEARNING

We are interested in a data driven approach to investigate the diversity of SNe Ia at maximum brightness. Our strategy consists in reducing the dimensionality of the initial feature space and subsequently applying an unsupervised learning algorithm. However, if we follow the traditional approach of constructing the input matrix only with spectra at maximum (or in a certain epoch bin around it), we will end up with a very small matrix (∼150 objects). It would be difficult for any dimensionality reduction algorithm to grasp the details of a complex space starting with such a small matrix. Concatenating spectra according to the observed epoch bin is a good alternative, but requires a dimensionality reduction tool armed to cope with missing data, as has already been demonstrated in Sasdelli et al. (2015).

Here we choose to use *transfer learning*, a recent area in machine learning that deals with the general problem of exploiting information from a variety of different environments to help with learning, inference, and prediction in a new environment where training data is scarce (Quionero-Candela et al. 2009). A simple example is spam filter detection, where one could aim at using the feedback (i.e., labeled data) of a group of existing users to help generating a model for a new user (Pan & Yang 2010).

Our data scenario fits within the scope of transfer learning. One is given spectra from the same supernova at various epochs, which can be treated as different observations. Despite being the electromagnetic signature of different astrophysical conditions, spectra from different epochs share common properties (e.g., they all have absorption/emission lines). Consequently, if we consider each spectrum as an independent object (a different line in the data matrix) we can train our code to recognize spectral features, irrespective of the epoch of the observation.

Following this reasoning, our initial data matrix was built with all available SN Ia spectra, regardless of the epoch of the observation[6]. This allows us to exploit all available spectra, even those with unknown epoch of observation. Despite the increase in data volume, the resulting feature space is much more stable, less affected by the inclusion/removal of individual spectra and provides a safer ground for spectral feature recognition. Thus, each line in our data matrix holds the derivative of the flux for an observed spectrum between 4000 and 7000Å, sampled in bins of 10Å. The complete matrix contains 3677 lines and 300 columns and serves as input to the dimensionality reduction algorithms (section 3).

Once the low dimensional space was determined, we selected only those spectra of interest (within 3 days from maximum brightness, resulting in 486 individual spectra) for the unsupervised learning phase. Figure 3 illustrates how the spectra around maximum occupy a well defined region of the PCs (feature) space. This configuration can easily be used to study the time evolution of spectral features, as well as to estimate the epoch of a given spectrum. In this work,

[6] Closely related strategies, with different goals, were used by Richards et al. (2012) for semi-supervised photometric classification of SN curves, Kremer et al. (2015) for photometric redshift determination and Vilalta et al. (2013) for cepheids classification.

we focus on recognizing characteristics of SN Ia spectra at maximum.

Figure 4 shows how this approach impacts the reconstruction power of the DL feature space. The vertical axis represents residual variance between the reconstruction and the original data with (black circles) and without (grey stars) the transfer learning approach. To calculate variance, both data sets (one with all the spectra and another containing only spectra at B-maximum) were randomly split in a training (80%) and a test set (20%). DL was applied to both training sets with different number of features in the central layer. Resulting features were than used to reconstruct the spectra in the test set and the normalized residual variances between the predictions and the measured derivative spectra in the test set were then calculated. We observe that with transfer learning, 4 features suffice to converge to a stable fractional variance. Without transfer learning, the same performance level cannot be achieved, even if we employ 10 features along the intermediate layer.

5 COMPARING FEATURE SPACES

In order to quantify the impact of DL over the standard PCA algorithm, we provide a detailed comparison between these two feature spaces. Fig. 5 shows the reconstruction of the original spectra using PCA and DL. In black we show a few examples of SN Ia spectra at three representative epochs. The first four spectra are at ~ -13 days from B max. Three spectra are close to B max and, finally, three spectra are found in the nebular phase at $\sim +180$ days from maximum (from top left to bottom right).

Reconstructions using DL show very good agreement with observations at all epochs due to the non-linearity of its representations. It performs exceptionally well in the earliest and the latest spectra and on SNe with rare spectroscopic peculiarities such as, for example, SN 2005hk. The behaviour of the High Velocity Feature (Mazzali et al. 2005) of the Si II 6355Å in the early spectra (for example SN 2002dj and SN 2002bo) are also finely reproduced by DL when compared to PCA. The latter is competitive only away from the early and late epochs and on objects which are not too peculiar. For PCA to obtain reconstructions comparable to DL, we would need a large number of components. Using at least 15 PCs, reconstructions using PCA are, on average, competitive with DL (Fig. 5).

Fig. 6 shows the residual variances between reconstructions and original data using PCA and DL. Variances were calculated as explained in section 4. DL greatly outperforms PCA even when using a small number of features. Its reconstruction capability shows steady improvement until ~ 5 features, after that it remains approximately constant. PCA only achieves a comparable reconstruction with ~ 15 PCs. However, a large fraction of the variance explained by PCA can be traced to noise, and an unnecessary large number of components overfits the data. DL behaves robustly, less affected by noise, and preventing overfitting (the variance explained remains approximately constant for more than $\sim$4 features). This suggests that the intrinsic dimension in SN Ia spectra is approximately $4-5$. One of these dimensions is needed to explain the time evolution of the spectrum, leaving only $3-4$ hidden physical parameters to characterize the explosion. This hints to the apparent simplicity of the space of SN Ia spectra.

Results above are similar to those presented by Sasdelli et al. (2015), where 5 PCs were responsible for most of the data variance. Despite using a different data set and a different dimensionality reduction analysis, it is quiet remarkable that both analysis return the same order of magnitude for the number of features/PCs. This suggests a small number of important "hidden" parameters involved in SN Ia explosions.

6 DATA VISUALIZATION

Given that DL is a relatively new technology in machine learning, and has no precedence in the study of SN Ia spectra, we provide the reader with a couple of visualization tools to enable analysis of this feature space. The two algorithms described below are part of the field of dimensionality reduction, but here we employ them as visualization tools.

6.1 Self-organizing maps

Self-Organizing Maps (SOM) are a special kind of *artificial neural networks*, often invoked to visualize data in an unsupervised manner. They are commonly used to find a two-dimensional embedding of the data and have been extensively employed in astronomy, e.g., in stellar spectra (Mahdi 2011), light curve classification (Brett et al. 2004), and object selection and photometric redshift estimation (Way & Klose 2012; Geach 2012).

The main idea behind SOM is to construct a 2-dimensional representation of the data where similar objects are placed close to each other. The algorithm can be described as follows: consider data in $\mathbb{R}^d$, and a two-dimensional grid $M = S_1 \times S_2$. Initially each cell of the grid, $C \in M$, hosts a random vector $\mathbf{c} \in \mathbb{R}^d$, called prototype. One then chooses, at random, one element from the data set, $\mathbf{x} \in \mathbb{R}^d$, and compares it to all prototypes in the grid using, for example, the Euclidean distance in $\mathbb{R}^d$. Let $\mathbf{p}$ be the prototype in the map that is closest to $\mathbf{x}_i$ and $P \in M$ denote the grid cell hosting $\mathbf{p}$. Then, $\mathbf{p}$ and all prototypes $\mathbf{q}$ of neighbouring cells Q of P are updated according to the following rule:

$$\mathbf{q} = \mathbf{q} + \alpha \cdot h(P, Q) \cdot (\mathbf{x} - \mathbf{p}), \tag{5}$$

where h is a neighbourhood function (typical functions are $h \equiv 1$, $h(P,Q) = \|P - Q\|$, or $h(P,Q) = e^{-(\sqrt{2}\sigma)^{-2}\|P-Q\|^2}$) and α, named *learning rate*, is usually reduced during the iterative process. Thus, prototypes of P and its neighbouring cells are made "more similar" to $\mathbf{x}$. Another element from the data set, $\mathbf{x}'$, is then compared to this new grid configuration. In case this data vector is very similar to the first it will probably be allocated in P or one of its neighbouring cells. Otherwise, it will populate another cell, P', defining a new *locus* on the grid which will host its characteristics. The comparison is repeated for all objects in the data and for the entire data set until convergence. As a result, we are left with a 2-dimensional re-organization of the initial data, where similar objects are allocated in nearby cells and distinct ones occupy different extremes of the grid.

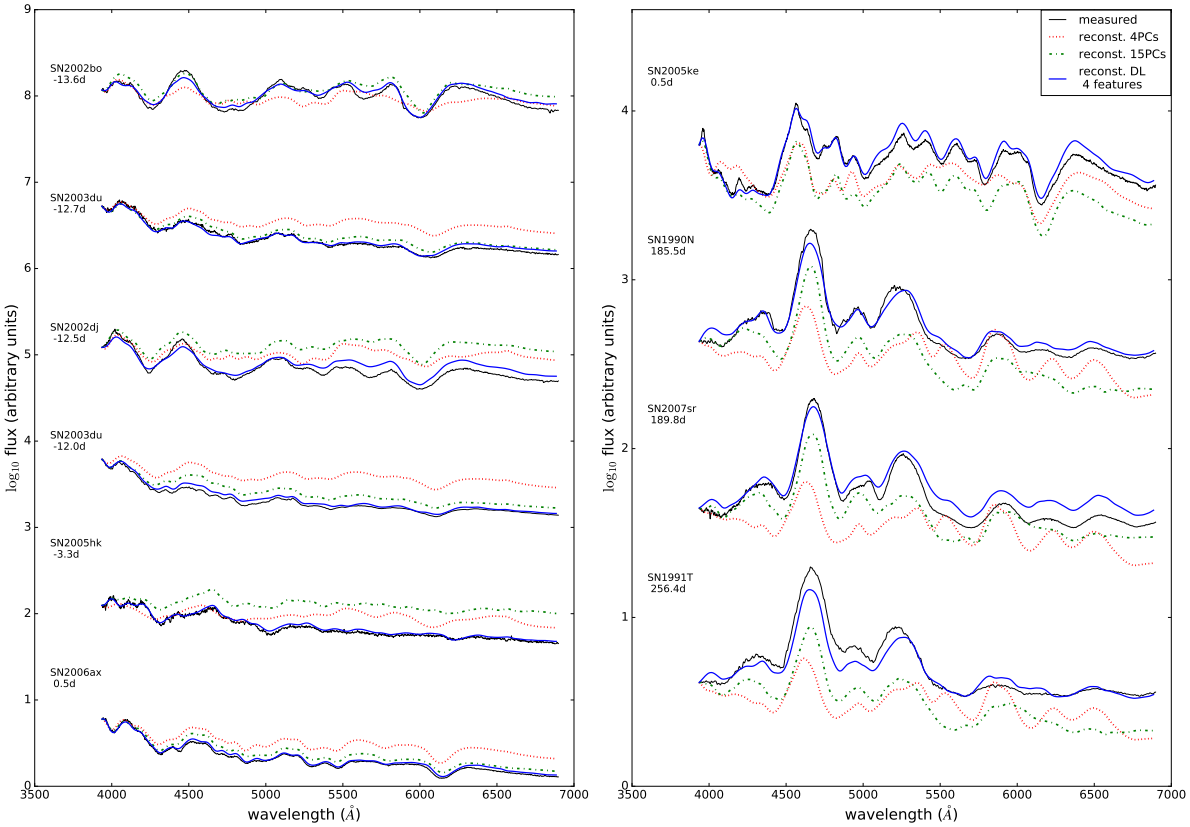

Figure 5. Reconstruction of a few examples of measured SN Ia spectra (black) using Deep Learning (thin blue) and PCA using 4 (dot-dashed red) and 15 (dashed green) PCs.

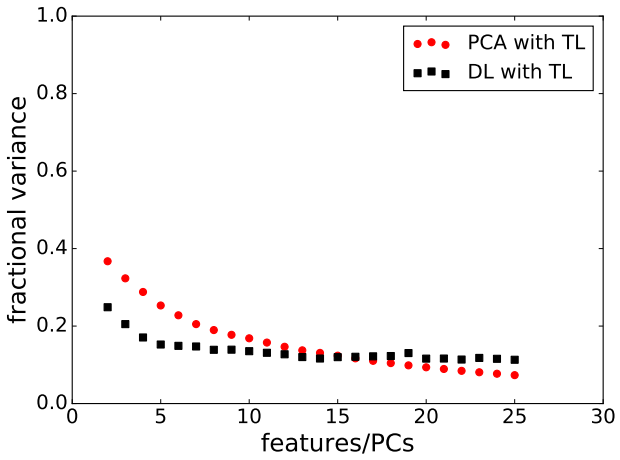

Figure 6. Comparison between PCA and Deep Learning in their capacity to reconstruct the original spectra. Horizontal axis stands for the number of PCs/features and vertical axis shows the deviation between real data and reconstruction.

Figure 6.1 shows results from applying SOM to the 4-dimensional DL feature space. As described in section 4, for this task we selected only spectra close to maximum. This grid contains 10×10 individual cells showing the mean spectra (black line), standard deviations (pink/purple) and the numbers of spectra allocated in each individual cell[7]. Different spectroscopic classes are arranged over different parts of the grid. Normal SNe Ia such as SN 1994D and SN 2011fe are close to the centre of the grid, the peculiar and faint 91bg-like cluster on the top left and the high velocity SNe are on the bottom left corner. We also recognize 91T-like SNe on the right side of the grid. From this configuration, we can already tell that the DL feature space is able to grasp crucial differences between different spectral features. Literature suggests that 91T-like, High Velocity and 91bg-like SNe are the extremes of SN Ia spectral variability, while 91bg-like SNe are possibly a more isolated group (Cormier & Davis 2011; Blondin et al. 2012). In this context, spectroscopically normal SNe Ia form the bulk of the data set, connecting the other groups together through an almost continuous change in spectral features. These finding are nicely confirmed by our SOM analysis, which gives us a glimpse of the potential of this feature space. In the following sections, we show how this first visual analysis is in concordance with the more quantitative results obtained using unsupervised learning.

[7] We emphasize that Figure 6.1 does not show the final prototypes in each cell, but the mean of all the spectra allocated in them. We chose this visualization because the prototypes in our case would relate to the 4-dimensional DL space, making it impossible to recognize spectral features.

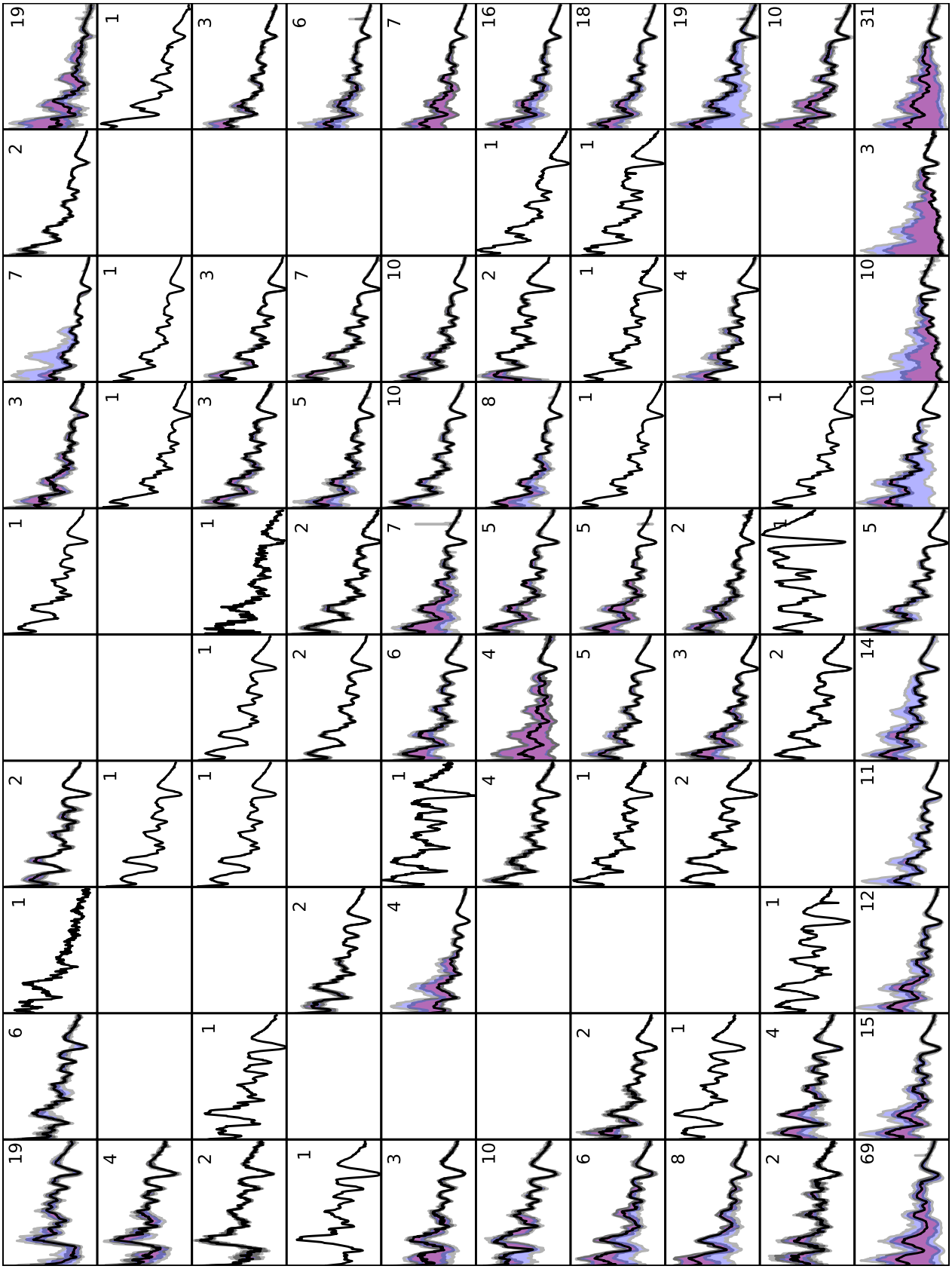

Figure 7. A self organized map of SN Ia spectra at maximum, constructed from the Deep Learning 4-dimensional feature space. Black lines correspond to the mean spectra of each cell, purple and blue bands correspond 1σ and 2σ respectively. Also shown are the number of spectra allocated in each individual cell.

6.2 Isomap

Isomap belongs to a broader class of dimensionality reduction techniques known as manifold learning. While PCA seeks to preserve the variance of the data, Isomap preserves its intrinsic geometry (Tenenbaum et al. 2000). More precisely, it can be seen as an extension of another classical dimensionality reduction method, called *multi-dimensional scaling* (MDS), which aims at finding a low-dimensional embedding of the data such that the distance between any pair of two points is preserved. Isomap generalizes this idea by resorting to "geodesic manifold distances", which are approximated via, e.g., a neighbourhood graph in which two points (nodes) are connected if one of the points is within the set of the K-nearest neighbours of the other one (Tenenbaum et al. 2000). The geodesic distance $d_M(i, j)$ between two points i and j can be defined as the shortest path between the two points in that graph. These distances are then used as in classical MDS, which usually resorts to the standard Euclidean distances ("straight lines"). Thus, in contrast to MDS, Isomap can also capture non-linear manifold structures. In astronomy, it was recently applied to spectroscopic classification by Bu et al. (2014).

In what follows, we use Isomap to provide a 2-dimensional visualization of the 4-dimensional feature space obtained with DL. It provides a much clearer view of the distribution of points in the deep learning feature space and facilitates its visual comparison with other classifications schemes.

7 UNSUPERVISED LEARNING

In previous sections we introduced efficient dimensionality reduction techniques. We now turn to the last step of our endeavour: unsupervised learning. Our main goal is to show the feasibility of automatically identifying sub-classes of SNe Ia with minimum assumptions about the physics and dynamics of the SN mechanism. Unsupervised learning techniques identify groups or clusters in a data set by maximizing the similarity among objects within the same cluster, and maximizing the dissimilarity among objects from different clusters.

The computational complexity of a clustering algorithm increases as the dimension of the data grows larger. This is because added dimensions increase the volume of space fast, turning the data sparse; the capacity of finding clusters then deteriorates. This effect is known as the *curse of dimensionality*. DL represents our solution to mitigate this effect.

While there are many clustering algorithms readily available, the K-Means is the most popular. We have compared different methods using simulated data and found that K-Means exhibits a good performance. We will therefore focus on this method in what follows.

7.1 K-means

The K-means algorithm is one of the most well-known clustering techniques, yielding intuitive solutions. In its original form (MacQueen 1967) it begins by choosing at random k vectors of the same dimensions of the data (with k chosen by the user). These will act as centres for potential groups definition. A distance (Euclidean) is then calculated between all vectors in the data set and the centre candidates. Each data point is assigned to the groups represented by its closer centre candidate. Once the first set of groups is defined, the centre candidates are updated to the centroid defined by all the members in each group. The process is repeated until the centroids are not changed due to further iterations. In practice, this local search strategy quickly converges to a solution (e.g., after 50 iterations).

One drawback of K-means is the need to explicitly state the number of clusters *a priori*. In our case, we wish to show that our approach is able to reproduce the classification proposed by Wang et al. (2009) and, as a consequence, we will always search for 4 clusters. A deeper analysis quantifying the degree of cluster coherence throughout different number of clusters will be addressed in a subsequent paper.

The methods described above have been made available in a single toolbox that enables quick analysis of a (potentially) large initial data set. We briefly describe this new toolbox next.

8 DRACULA

We present the `DRACULA` (Dimensionality Reduction And Clustering for Unsupervised Learning in Astronomy), where the methods discussed in this paper have been implemented. The toolbox is written in Python, is publicly available, and can be easily adapted to multiple applications. The software relies heavily on tools developed in scikit-learn (Pedregosa et al. 2011). The DL analysis used the H2O package (Arora et al. 2015) and the SOM routine used the Kajić et al. (2014) implementation. Thanks to its modular design, all main steps, e.g., dimensionality reduction, unsupervised learning (clustering) and plotting, can be run separately.

In what follows, we demonstrate how `DRACULA` can be invoked to obtain similar results to those presented in this paper[8]. We avoid a detailed description of all code functionalities, and focus on main procedures, the documentation[9] provides more detailed descriptions. Let us start assuming an initially big data matrix. In our case, this is composed of derivatives over the flux logarithm, as described in section 2. We now go through each step described in previous sections and demonstrate how the user can reproduce our results using `DRACULA`.

8.1 Dimensionality Reduction

To begin, the user needs to set up a configuration file (which must be named `config.py`) to define the type of desired analysis. This module contains four methods: PCA (Jolliffe 1986), Expectation Maximization PCA (Bailey 2012), Kernel PCA (Schölkopf et al. 1999) and DL (Deng & Yu 2014). Dimensionality reduction requires as input a list of observed features for each object (1 line per object, 1 column per feature). For clarity, input and output are defined next:

[8] If you want to be updated on `DRACULA` development send a request to `coin_dracula+subscribe@googlegroups.com`
[9] `https://github.com/COINtoolbox/DRACULA`

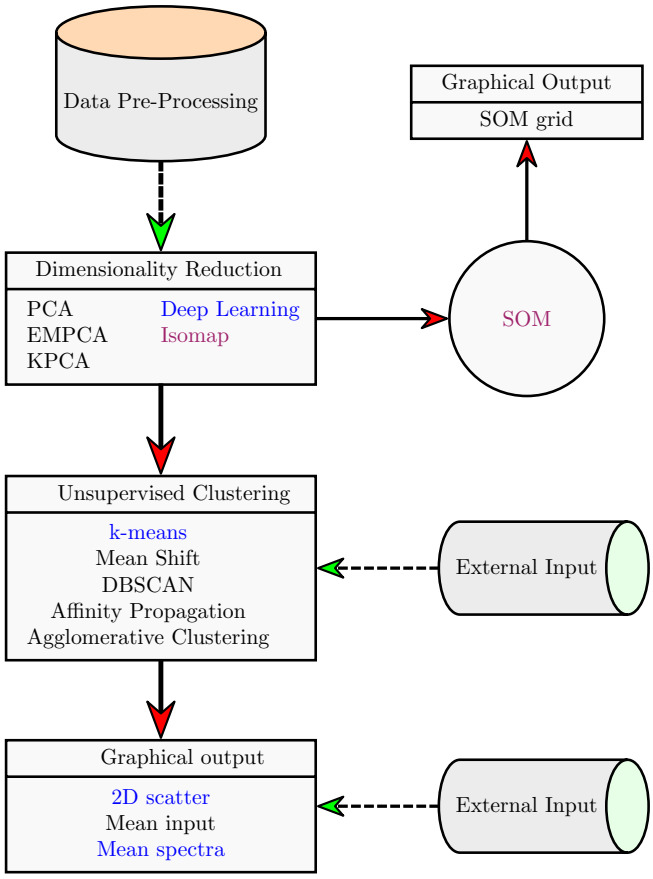

Figure 8. Flow chart describing the capabilities of the DRACULA package. Cylinders represent external inputs, the rectangles denote the tools which constitutes the package and circles represent modules which can be used independently. Red (full) arrows indicate the complete algorithm and green (dashed) arrows indicate points where external inputs can be inserted. The names marked in blue are the ones used to obtain the results in section 9 and the ones marked in purple are dimensionality reduction tools which we used here only for visualization.

- **input:** data amenable to reduction (ex: spectra derivatives)
- **output:** reduced data

In order to perform dimensionality reduction with DL[10] the configuration file must contain: `ORG_DATA`, that receives the path to the file with uncompressed data and `REDUCTION_METHOD` that determines the dimensionality reduction algorithm. If no other options are included in the configuration file, the code will run using default values[11]. The user might change default values by modifying the random seed, `DeepLearning_seed`, number and structure of hidden layers, `DeepLearning_n_layers` and `DeepLearning_hidden` respectively, in the configuration file.

[10] Note that the codes uses `R` as an interface, which requires installation of `R` (https://www.r-project.org/), h2o (http://h2o.ai/) and rpy2 (http://rpy.sourceforge.net/) packages.
[11] Initial random seed: `DeepLearning_seed = 1`; number of hidden layers `DeepLearning_n_layers=7`; structure of hidden layers: `DeepLearning_hidden='c(120,100,90,50,30,20,4,20,30,50,90,100,120)'`

One can run the chosen dimensionality reduction routine from the command line typing DRAC_REDUCTION. The output containing the reduced data is created in a folder called `red_data`.

8.2 Clustering

Clustering follows using the output file from the last section or using an externally reduced dataset. In the case of an external source, `ORG_DATA` should be commented out and the path to the externally reduced file must be given as

```
REDUCED_DATA_EXTERNAL = "<path to reduced data>"
```

The running format is the same as dimensionality reduction, where the input is the reduced data. After clusters are detected, the algorithm outputs their centres and the corresponding labels for each object. In short:

- **input:** reduced data
- **output:** centers of clusters and labels for each object in the reduced data file

Available options here are KMeans (MacQueen 1967; Arthur & Vassilvitskii 2007), Mean Shift (Comaniciu & Meer 2002), Agglomerative Clustering (Voorhees 1986), Affinity Propagation (Comaniciu & Meer 2002) and DBSCAN (Ester et al. 1996). For our particular case, the configuration file contains:

```
CLUSTERING_METHOD = "KMeans"
```

Analogous to the dimensionality reduction case, the user can change parameters. An example is the number of clusters using variable `KMeans_n_clusters`[12]. At this stage it is also possible to use a mask, which is specially important for cases where transfer learning is applied (e.g., when the dimensionality reduction algorithm is applied to a big diverse matrix and only a subset of the reduced data is intended for clustering). The mask consists of a file with the same number of lines as the reduced data; on each line "1" indicates objects included in the clustering analysis and "0" refers to all remaining objects. The path for the mask should be provided as follows:

```
MASK_DATA = '<path to mask file>'
```

The clustering routines can be run in the command line by typing DRAC_CLUSTERING.

8.3 Varying parameters

In cases where it is necessary to compare the output from a range of parameters, `DRACULA` allows the user do it automatically[13]. The parameters to be declared in the configuration file are

[12] Default value is `KMeans_n_clusters=4`.
[13] Beware that this functionality only permits to change one parameter at a time.

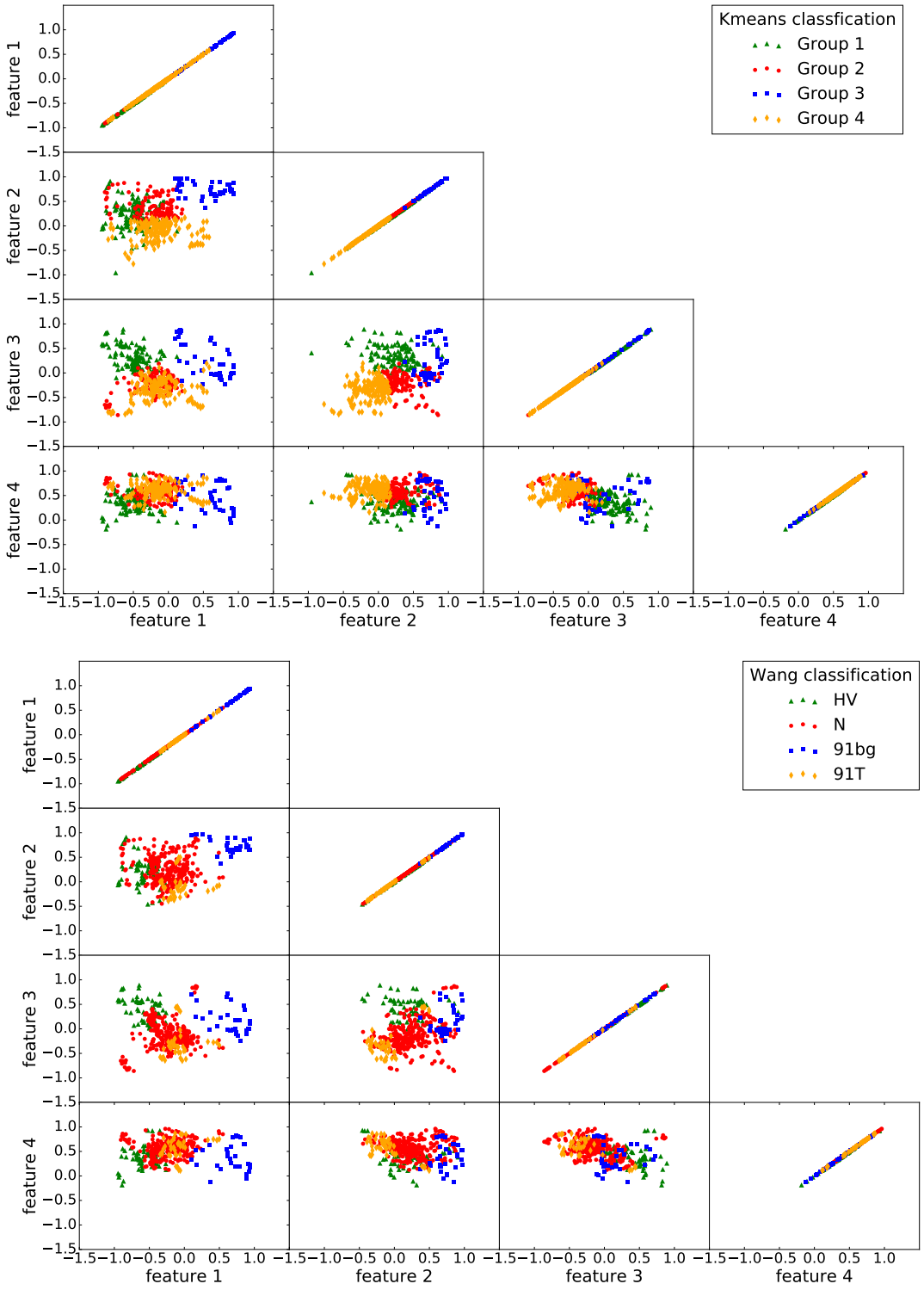

Figure 9. Four dimensional feature space resulting from Deep Learning. Each point corresponds to one SN Ia spectrum within −3 and +3 days since maximum. **Top:** Colours represent the groups found by the K-Means algorithm when 4 groups are required. **Bottom:** Colours correspond to the classification suggested by Wang et al. (2009). It does not show the objects classified as "peculiar".

```
VAR_TYPE = 'CLUSTERING' or 'REDUCTION'
VAR_PAR = 'REDUCTION_METHOD' or 'CLUSTERING_METHOD' or 'MeanShift_quantile' or ...
VAR_VALS = [1,2,3] or ['name1','name2','name2'] or [vec1,vec2,vec3]
```

This functionality uses the same configuration file (`config.py`) with the above extra keys, and can be run by typing DRAC_COMPARISON.

8.4 Plotting

It is useful to plot results to gain insight on the cluster quality. Plots can be generated for the whole process, from reduction to clustering, with input and output given by:

- **input:** reduced data, cluster centres and labels for each object in the reduced data file
- **output:** scatter plot of reduced data coloured according to labels

Scatter plots, similar to those shown in Figure 9, are generated by typing DRAC_PLOT. The format of the output files is selected by setting the keyword `PLOT_EXT`, in the configuration file.

If one wishes to use this tool for plotting clusters assigned by an external clustering algorithm, the path to the corresponding labels should be provided:

```
LABELS_DATA_EXTERNAL = "<path to labels file>"
```

Finally, it is useful to visualize the mean input data within each cluster. Features can then be compared with expected patterns. The structure in this case is

- **input:** reduced data and cluster labels
- **output:** plot of mean input data for each cluster

If the data for visualization is not in the file used to make the reduction (in our case the reduction is performed on the derivative of the spectra, but we would like to observe the mean spectrum of each group found by K-Means), it can be plotted by setting:

```
SPECTRAL_DATA_EXTERNAL = "<path to original data file>"
```

The format of the output file for the mean data (or mean spectrum) needs to be set separately through the keyword `PLOT_SPEC_EXT`. The plots can be generated using the command DRAC_PLOT_SPECS.

We emphasize that the above provides only a glimpse of the capabilities of DRACULA. The package has other functionalities (e.g. cluster validation routines) which will be fully explored, and described, in subsequent work.

8.5 SOM

The SOM module (section 6.1) is independent from the steps previously described. The structure is

- **input:** reduced data
- **output:** SOM grid

This module requires its own configuration file which must called `config_som.py`. The path to the reduced data file is given through the keyword `ORG_DATA`. When no other option is provided in the configuration file, the module will run a 10×10 matrix through 100 iterations, and the output plot will show the prototypes populating each cell. If the user wants the SOM grid to show the mean of the original data (in our case, the mean of observed spectra assigned to each cell), the keyword `SPECTRAL_DATA_EXTERNAL` must point to the original data file. The number of iterations can be set by adding the keyword `Niter` to the configuration file[14]. SOM module can be run by typing DRAC_SOM.

9 RESULTS

We used `DRACULA` to analyse the sample of SN Ia spectra introduced in section 2. Results presented below correspond to the 4-dimensional DL feature space (section 5) subjected to the K-Means algorithm. In order to compare our results with the classification proposed by Wang et al. (2009), we set the K-Means to search for 4 distinct groups (section 7).

Figure 9 shows 4-dimensional DL feature space configuration. In the upper panel, colours correspond to the groups found by K-Means and in the lower panel the points are identified according to the Wang et al. (2009) classification[15]. Although the two classification schemes are not in complete agreement, they share some basic characteristics. For example, the scatter plot in the feature space formed by features 1 and 2 are quite similar in the upper and lower panels. This leads to the idea that maybe group 3 (upper panel) can be associated with 1991bg-like SNe (lower panel).

A better visualization of this feature space is achieved by applying the isomap algorithm (sec. 6.2) to the 4-dimensional DL feature space. Fig. 10 shows the resulting 2-dimensional isomap space with groups identified by K-Means (left panel) and the ones suggested by Wang et al. (2009, right panel). This figure not only demonstrates how isomaps can be a powerful tool in high dimensional data visualization, but also clarifies the potential of combining DL with unsupervised learning algorithms.

However, our final goal is to be able to identify what different spectral characteristics separate the groups found with our method. Thus, we show in Fig. 8.4 the mean spectrum of each group in both classification schemes. The agreement between the mean spectra is proof that DL, when coupled with unsupervised learning algorithms is able to automatically identify important spectral features without the human screening. The method recovers classes similar to the ones defined by the visual inspection and this can be used to optimize, with a data driven approach, the current identification of classes or subclasses of SNe.

Beyond that, it is also capable of identifying a hierarchical structure within the data set. Figure 12 shows the mean spectrum of all SN Ia spectra at maximum (top-left panel)

[14] The grid shown in Figure 6.1 was constructed with `Niter = 10000000`.

[15] The lower panel of Fig. 9 does not show the SNe classified as "peculiar" by Wang et al. (2009) because there is not a unique underlying spectral characteristic which defines this group. They mainly were not able to fit in the other four categories.

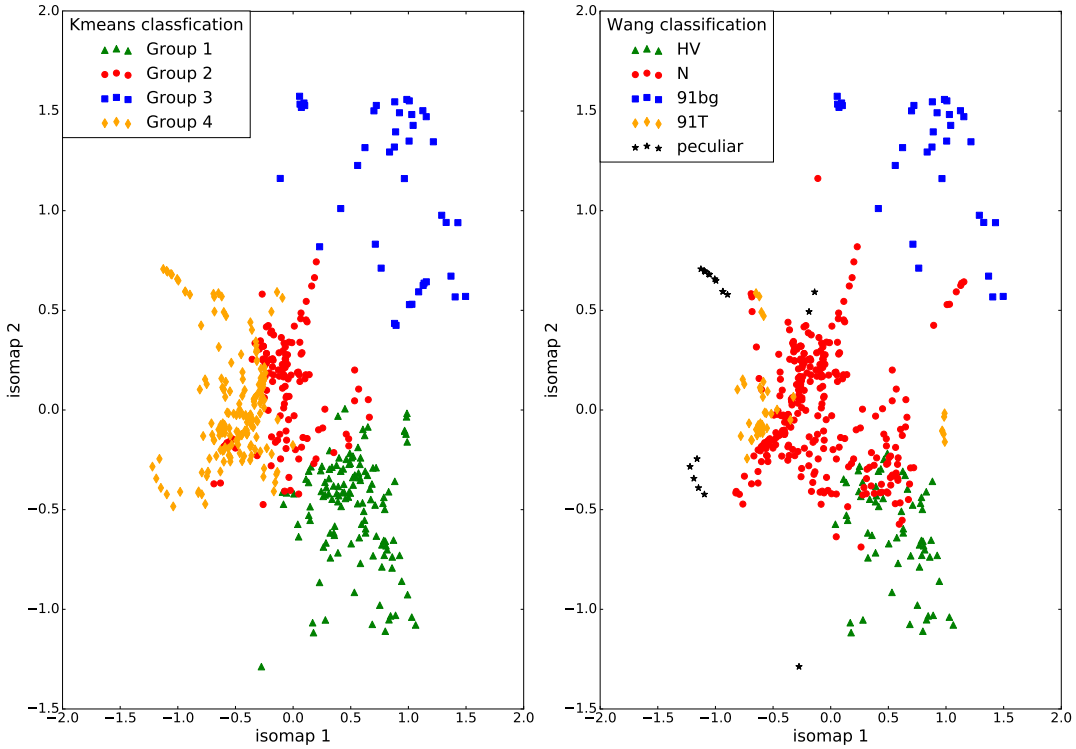

Figure 10. Four dimensional feature space from Deep Learning reduced to 2 dimensions through isomap. **Left:** Groups found by the K-Means algorithm when 4 groups are imposed. **Right:** Objects separated according to the classification proposed by Wang et al. (2009).

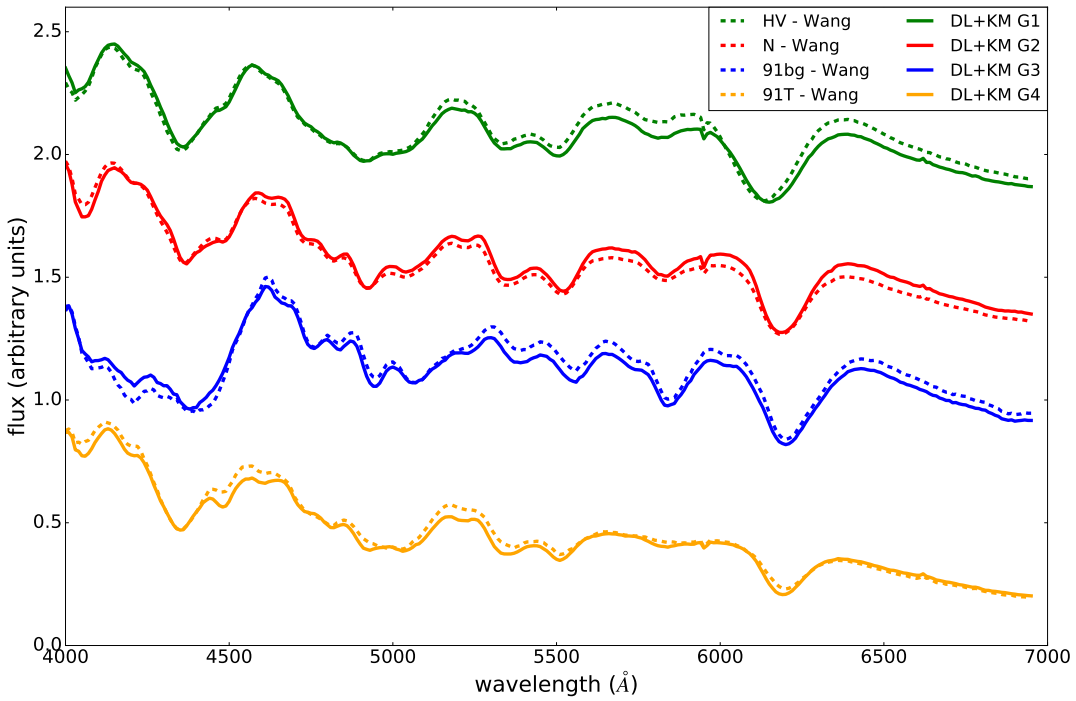

Figure 11. Comparison between the mean spectrum found by Deep Learning + KMeans (full lines) and mean spectrum of the groups defined by Wang et al. (2009, dashed lines).

as well as the mean spectrum of each group for the K-Means configurations with 2 to 4 groups, comparing it with the classification proposed by Wang et al. (2009). This shows that the velocity at which most lines form (the velocity of the photosphere) is the first order spectral characteristic which defines subgroups of SNe Ia (2 groups configuration). After this, the 1991bg-like objects are kept in a group on their own (3 group configuration) and finally the 1991T-like objects are separated. In order words, depending on the degree of specialization we demand from the data, we are able to recognize a sequence of spectral features which might be used to define sub-populations of SNe Ia.

10 DISCUSSION

We combined a series of contemporary machine learning techniques in a framework built to automatically identify classes within a set of measured spectra. As a first application of this tool, we investigate our ability to recover previously reported sub-classes of SNe Ia.

The set of public SN Ia spectra (section 2) was first submitted to the preprocessing described in Sasdelli et al. (2015) and the resulting matrix was used as input for the algorithm which can be summarized in 3 main steps: transfer learning, dimensionality reduction and unsupervised clustering.

The goal of transfer learning is to ensure the stability of the low dimensional feature space by adding a large variety of spectra in the original data matrix. In our example, although the clustering analysis is focused at B-maximum, we used spectra from all the available epochs for the training. However, once the low dimensional feature space was constructed, only the projections corresponding to spectra at maximum were selected for the next phase.

We introduce Deep Learning for dimensionality reduction on SN Ia spectra. This is a cutting edge technique only recently introduced in astronomy (Huertas-Company et al. 2015). We proved its effectiveness in spectroscopy and showed that it performs much better in the reconstruction of measured spectra than the commonly used PCA algorithm. Reducing the dimensionality of the feature space from ~ 300 hundreds wavelengths bins to 4.

Since this is the first application of Deep Learning for SN Ia spectra characterization, we used *Self- Organizing Maps* (SOM) in order to have a better idea of the potentialities of this new feature space. This visualization technique shows how spectral properties vary with continuity in the SN Ia spectra space. Specifically it allowed the visualization of the peculiarity of subgroups reported in the literature, like the 91bg-like SNe (Cormier & Davis 2011; Blondin et al. 2012).

Last but not least, we used unsupervised learning techniques to investigate the possibility of identifying spectroscopic features in subclasses of SN Ia spectra at maximum light. This allowed us to define a data-driven classification scheme *a posteriori* and analyse the groups separately in order to look for the spectroscopic characteristics which define them. This enables the classification of a fairly large data set requiring the astronomer to visually inspect only a handful of possibilities (the mean spectra). We use the low dimensional space from Deep Learning as an input for the K-Means algorithm. In order to provide a more friendly visualization of the four dimensional feature space, we use isomap as a further layer of dimensionality reduction. Here the separation between the groups are much more obvious and the identification with the Wang et al. (2009) groups is also much clearer.

We find that the time and spectral variability of SNe Ia can be summarized by a low dimensional space. The spectra starting from few days from the explosion up to more than a year after can be parametrized by a 5-dimensional space. The time evolution of the spectra, of course, uses one of these dimensions. This result suggests that only 4 underlying physical parameters are enough to describe SN Ia explosions and their spectroscopic variability, including the most "peculiar" objects, such as 91bg-like and the 02cx-like SNe. SNe Ia are well known to be mostly a uniform class of objects. Our results quantify this claim, and suggest that the progenitors should also be a "simple" system with no more than a handful of initial parameters.

The complete apparatus was built under `DRACULA`, a publicly available Python package and the complete algorithm applied to a public data set of SN Ia spectra.

Our results showed an impressive agreement between its mean spectra and the ones from Wang et al. (2009) classification scheme. Moreover, the method is also able to identify a hierarchical structure within the data set, confirming previous statements that the high velocity features are the first order effect in separating sub-samples of SNe Ia (Wang et al. 2013). Further developments along the lines presented here, like the analysis of time evolution in SN Ia spectra and the possibility to use these tools in the classification of other supernova types will be addressed in future works.

At this point, we showed that the combination of modern machine learning techniques can play a crucial role in the automatic treatment of a large number of observations, changing the relationship between the researcher and his/her first approach to the observational data. This work is only a glimpse on the kind of change which is about to come and the potentialities awaiting in further development of interdisciplinary fields.

ACKNOWLEDGEMENTS

We thank Bruce Basset and Paniez Paykari for insightful discussions. EEOI and RSS thank Petr Skoda for pointing out the capabilities of SOM. EEOI is partially supported by the Brazilian agency CAPES (grant number 9229-13-2). MA is supported by São Paulo Research Foundation (FAPESP) under grant number 2013/26612-2. VCB is supported by São Paulo Research Foundation (FAPESP)/CAPES agreement under grant number 2014/21098-1. HC is supported by CNPq. This work is a product of the 2nd COIN Residence Program. We thank Alan Heavens and Jason McEwen for encouraging the realization of this edition. The program was held in the Isle of Wight, UK in October/2015 and supported by the Imperial Centre for Inference and Cosmology (ICIC), Imperial College of London, UK, and by the Mullard Space Science Laboratory (MSSL) at the University College of London, UK. The IAA Cosmostatistics Initiative[16]

[16] https://asaip.psu.edu/organizations/iaa/iaa-working-group-of-cosmostatistics

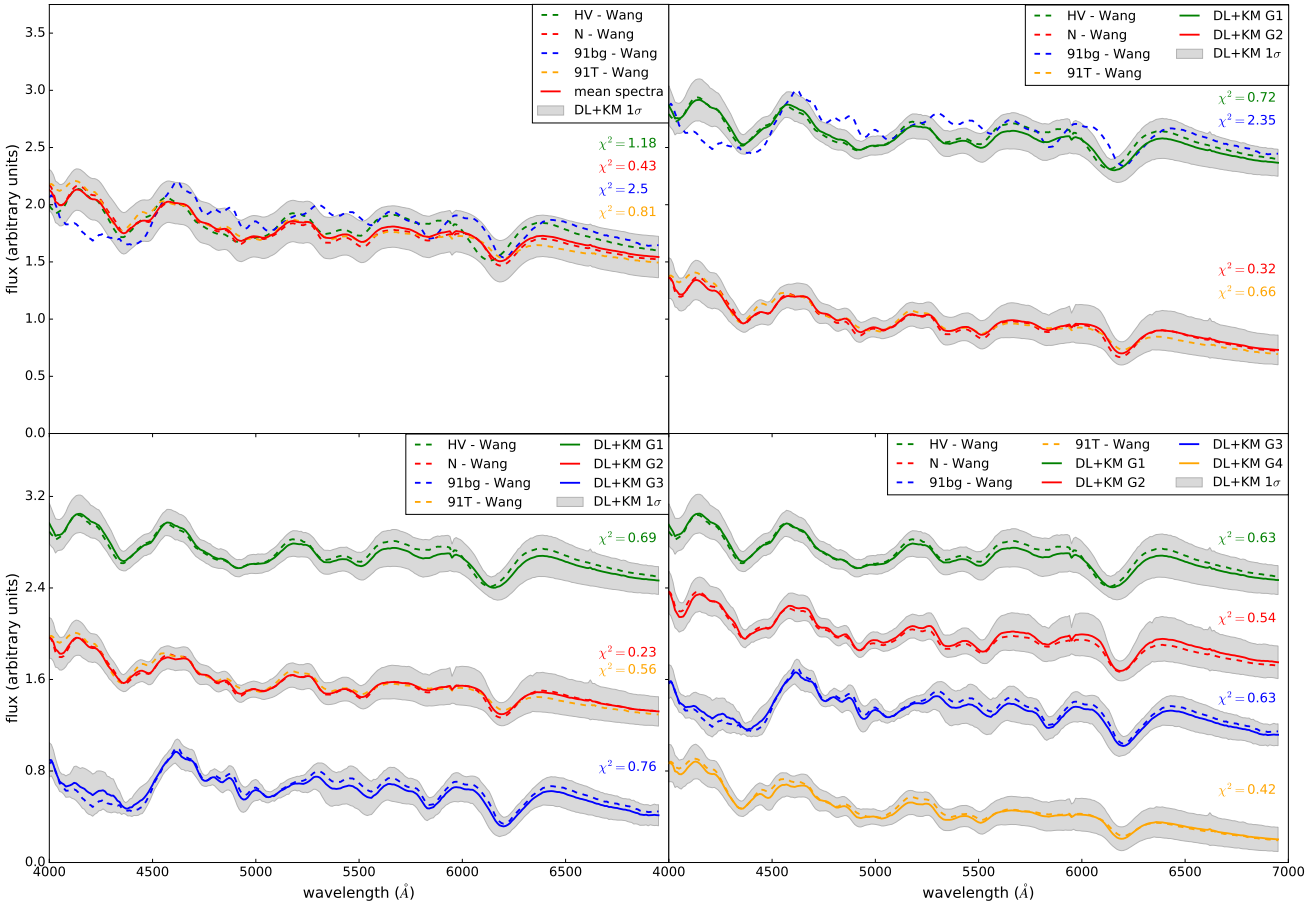

Figure 12. Representation of the hierarchical evolution of the groups found by K-Means algorithm (full lines) in comparison with the groups defined by Wang et al. (2009) (dashed lines). The top-left panel shows the mean of all SN Ia spectra between −3 and +3 days since B-maximum (full red line). Other panels show the mean spectra for different configurations of the K-Means algorithm, from 2 to 4 groups. Also shown are the deviations between the mean spectra between groups identified with the K-Means algorithm and the mean spectra for the groups defined by Wang et al. (2009). The grey region denotes 1σ scatter for the groups defined with the K-Means algorithm.

(COIN) is a non-profit organization whose aim is to nourish the synergy between astrophysics, cosmology, statistics and machine learning communities. This work was written on the collaborative `Overleaf` platform[17], and made use of the GitHub[18], a web-based hosting service, the `git` version control software, and Slack[19] a team collaboration platform.

APPENDIX A: GLOSSARY

In order to avoid confusion due to different nomenclatures used in the machine learning and astronomy communities, we provide below a list of the terms used in this paper:

- Feature (Machine Learning): it is a measured property or an observation. In our context it corresponds to the flux (or the derivative of the flux) in a given wavelength bin. It maintains the same name after the dimensionality reduction.
- Feature space (Machine Learning): commonly known in astronomy as parameter space. This space is high dimensional in the beginning of our analysis ($\sim$ 300 wavelength bins) and after going through the Deep Learning machinery it becomes 4-dimensional.
- Parameter (Computing): the input value of a variable of a computer program.
- Physical parameter (Supernova Physics): it is a generic term to describe one of the initial conditions of the supernova explosion.
- Prototype (SOM): vectors populating each cell of a SOM. Initially these are random but as the code iterates they become more representative of the data vectors assigned to their cell.
- Spectral feature (Spectroscopy): absorption and/or emission feature in the spectrum. It originates from a group of atomic lines with similar energy. It is usually dominated by the lines of a single ion.
- Weight parameter (Deep Learning): weight matrices

[17] www.overleaf.com
[18] www.github.com
[19] https://slack.com

connecting adjacent layers of the Deep Learning network. They get optimized during the training phase using the entire training sample.

REFERENCES

Arora A., Candel A., Lanford J., LeDell E., Parmar V., 2015, Deep Learning with H2O. http://h2o.ai/resources

Arsenijevic V., 2011, MNRAS, 414, 1617

Arthur D., Vassilvitskii S., 2007, in Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '07. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp 1027–1035, http://dl.acm.org/citation.cfm?id=1283383.1283494

Bailey S., 2012, PASP, 124, 1015

Ball N. M., Brunner R. J., 2010, International Journal of Modern Physics D, 19, 1049

Benetti S., et al., 2005, ApJ, 623, 1011

Bengio Y., Courville A., Vincent P., 2013, IEEE Transactions on Pattern Analysis and Machine Intelligence, 35, 1798

Benitez-Herrera S., Ishida E. E. O., Maturi M., Hillebrandt W., Bartelmann M., Röpke F., 2013, MNRAS, 436, 854

Blondin S., et al., 2012, AJ, 143, 126

Branch D., Dang L. C., Baron E., 2009, PASP, 121, 238

Brett D. R., West R. G., Wheatley P. J., 2004, Monthly Notices of the Royal Astronomical Society, 353, 369

Bu Y., Chen F., Pan J., 2014, New Astronomy, 28, 35

Comaniciu D., Meer P., 2002, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24, 603

Cormier D., Davis T. M., 2011, MNRAS, 410, 2137

Crisci C., Ghattas B., Perera G., 2012, Ecological Modelling, 240, 113

De Souza R. S., Maio U., Biffi V., Ciardi B., 2014a, MNRAS, 440, 240

De Souza R. S., Ishida E. E. O., Whalen D. J., Johnson J. L., Ferrara A., 2014b, MNRAS, 442, 1640

Deng L., Yu D., 2014, Found. Trends Signal Process., 7, 197

Ester M., Kriegel H.-P., Sander J., Xu X., 1996, in Proc. of 2nd International Conference on Knowledge Discovery and. pp 226–231

Ferreras I., Pasquali A., de Carvalho R. R., de la Rosa I. G., Lahav O., 2006, MNRAS, 370, 828

Folatelli G., et al., 2013, ApJ, 773, 53

Geach J. E., 2012, MNRAS, 419, 2633

Hillebrandt W., Kromer M., Röpke F. K., Ruiter A. J., 2013, Frontiers of Physics, 8, 116

Hinton G. E., Salakhutdinov R. R., 2006, Science, 313, 504

Huertas-Company M., et al., 2015, ApJS, 221, 8

Ishida E. E. O., de Souza R. S., 2011, A&A, 527, A49

Ishida E. E. O., de Souza R. S., 2013, MNRAS, 430, 509

Ishida E. E. O., de Souza R. S., Ferrara A., 2011, MNRAS, 418, 500

Ivezic Z., Connolly A. J., VanderPlas J. T., Gray A., 2014, Statistics, Data Mining, and Machine Learning in Astronomy: A Practical Python Guide for the Analysis of Survey Data, stu - student edition edn. Princeton University Press, http://www.jstor.org/stable/j.ctt4cgbdj

Jolliffe I., 1986, Principal Component Analysis. Springer Verlag

Kajić I., Schillaci G., Bodiroža S., Hafner V. V., 2014, in Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction. HRI '14. ACM, New York, NY, USA, pp 192–193, doi:10.1145/2559636.2559816, http://doi.acm.org/10.1145/2559636.2559816

Kremer J., Gieseke F., Steenstrup Pedersen K., Igel C., 2015, Astronomy and Computing, 12, 67

Krone-Martins A., Ishida E. E. O., de Souza R. S., 2014, MNRAS, 443, L34

LeCun Y., Bengio Y., Hinton G., 2015, Nature, 521, 436

Li W., Filippenko A. V., Treffers R. R., Riess A. G., Hu J., Qiu Y., 2001, ApJ, 546, 734

Libbrecht M. W., Noble W. S., 2015, Nat Rev Genet, 16, 321

MacQueen J. B., 1967, in Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. pp 281–297

Mahdi B., 2011, preprint, (arXiv:1108.0514)

Mazzali P. A., et al., 2005, ApJ, 623, L37

Mitra S., Choudhury T. R., Ferrara A., 2011, MNRAS, 413, 1569

Morrey J. R., 1968, Analytical Chemistry, 40, 905

Pan S. J., Yang Q., 2010, IEEE Trans. on Knowl. and Data Eng., 22, 1345

Paykari P., Lanusse F., Starck J.-L., Sureau F., Bobin J., 2014, A&A, 566, A77

Pedregosa F., et al., 2011, Journal of Machine Learning Research, 12, 2825

Perlmutter S., et al., 1999, ApJ, 517, 565

Quionero-Candela J., Sugiyama M., Schwaighofer A., Lawrence N. D., 2009, Dataset Shift in Machine Learning. The MIT Press

Richards J. W., Homrighausen D., Freeman P. E., Schafer C. M., Poznanski D., 2012, MNRAS, 419, 1121

Riess A. G., et al., 1998, AJ, 116, 1009

Sasdelli M., et al., 2015, MNRAS, 447, 1247

Schölkopf B., Smola A. J., Müller K.-R., 1999, MIT Press, Cambridge, MA, USA, Chapt. Kernel Principal Component Analysis, pp 327–352, http://dl.acm.org/citation.cfm?id=299094.299113

Silverman J. M., et al., 2012, MNRAS, 425, 1789

Tenenbaum J. B., Silva V. d., Langford J. C., 2000, Science, 290, 2319

Vidyasagar M., 2015, Annual Review of Pharmacology and Toxicology, 55, 15

Vilalta R., Gupta K. D., Macri L., 2013, Astronomy and Computing, 2, 46

Vincent P., Larochelle H., Bengio Y., Manzagol P. A., 2008, in Proceedings of the International Conference on Machine Learning..

Voorhees E. M., 1986, Inf. Process. Manage., 22, 465

Wang X., et al., 2009, ApJ, 699, L139

Wang X., Wang L., Filippenko A. V., Zhang T., Zhao X., 2013, Science, 340, 170

Way M. J., Klose C. D., 2012, PASP, 124, 274

Yaron O., Gal-Yam A., 2012, PASP, 124, 668

This paper has been typeset from a TEX/LATEX file prepared by the author.