Introduction

2) Data set:

Supernova (SN) are classified based on their spectra by the presence and lack-there of certain elements, most-prominently Hydrogen. We have obtained 134 light curves of spectroscopically classified stripped-envelope SNe from the intermediate Palomar Transient Facility. These include Type Ib, Ic, IIb, and some potential contamination from other types of SNe. Our data-set greatly varies in quality, with some SNe sampled very regularly while others have very few points at all. The SNe light curves are in either g-band (14) or r-band (130), where for each SN a single filter has been used. Both spectroscopically and in their light curves, type Ib/c SNe (which do not show Hydrogen in their spectra) are quite similar, and might even be from the same class, while Type IIb supernovae do show some evidence of Hydrogen, and can evolve differently.

1) Question:

We would like to investigate our data-set for differentiation in the light curves belonging to these classes. Clustering (or sub-classifications) in the light curves could indicate that some of these spectroscopically similar SNe may actually arise from differing progenitor scenarios or explosion mechanisms. As a machine learning problem, we originally wanted to use the raw light curves, binned to a standardized high-dimensional data set consisting of 20 phase bins. However, many of our bins were devoid of data, and thus any clustering in the bins across the light curves were dominated simply by the sampling of our data instead of any features of the light curves themselves. Thus, we had to first employ a physically relevant dimensionality reduction technique and instead work on the features obtained from such a method.

Since we know as a prior that these SNe are expected to behave relatively similarly, we employed a template fitting method, where a spectroscopic template, either Ib/c or IIb, was fit to each SN. These templates came from either average of several SNe as in the case of the Ib/c template, or from a very well observed low-extinction nearby prototypical IIb SN, as in the case for the IIb template. The templates were linearly interpolated to match the sampling of whichever SN we are fitting. The templates had 3 parameters for each fit, a linear additive shift in time, a linear additive scale in magnitude space in brightness, and a multiplicative stretch factor, where the sampling of the template was stretched out by being multiplied by a constant, effectively either stretching out or squeezing the typical bell-shaped light curve of a SN. (See Fig. 1 for an example of a template fit to real SN data.)

For the fitting procedure, we used a regression method to minimize the residuals of the fit. So our cost function was simply the the difference of the interpolated magnitude of the template and the magnitude of the real SN light curve. We used the Levenberg–Marquardt algorithm for the regression. At each step, we fed in to the algorithm the residuals from the fit/initial-guess, and the algorithm searches for a step in all 3 parameters that minimizes sum of square errors. As the book notes, the Levenberg–Marquardt algorithm is prone to getting stuck in local minima, and as a result, requires a good guess of the initial parameters or a relatively smooth likelihood function surface of the parameters. As a result, the hyper-parameters used in the fit, both as initial guesses, but also as minimum and maximum limits, are very important for getting a good fit. Naively, if all the light curves look like each other after correcting for distance and time-dilation effect from the redshift, one would expect tightly clustered parameters, with values near zero for shift and scale, and near 1.0 for stretch. Of course, both the shift and the scale will have an uncertainty due to sampling bias. For example, if the beginning of the light curve is not caught, we would expect a larger absolute shift value, and if the peak is not caught, we would expect a larger absolute scale value. This is because the zero level for the shift is set by the first detection of the SN, and the zero level of the scale is set by the average of the 3 brightest points.

After we obtained these parameters, we decided to use the substantially fewer g-band model fits for verification. We manually inspected all the fits, and attempted to get better fits by modifying the original initial guesses and parameters limits from the model fitting algorithm. From the 14 light curves fit by the algorithm with the original set of assumptions (hyper-parameters), 1 was visibly a very poor fit, and for an additional 2 we were able to obtain a slightly better fit with

different hyper-parameters. As a result, the model fitting procedure was fully successful on 11/14, partial successful on 13/14, and failed on 1/14 of the g-band SN light curves. We then additionally visually classified these g-band SNe as either normal, broad, or poorly-observed. 12 were normal, one was broad, and one was poorly-observed. We will use the manually classified broad and poorly-observed SN to validate our clustering algorithms in the next section. We would naively expect these to be in different clusterings.

Finally, in order to answer our original question of sub-categories in the SN data, we implemented two unsupervised learning clustering algorithms. First was K-means, which is a flat clustering algorithm, meaning that one provides the number of clusters to be found in the data. The second was Density-Based Spatial Clustering of Applications with Noise (DBSCAN), implemented in SKLearn under the same name, which is a hierarchical clustering algorithm, meaning that it decides the number of cores to use as a part of the algorithm. With these methods, we investigate our now 3-dimensional dataset.

3) Clustering Algorithms:

There is a potential problem with using all 3 model parameters as our features in the clustering analysis. The parameter which sets the "shift" of the model to match the light curve can be effected by sampling bias to a much greater degree than is possible for the other two. The "scale" parameter setting the height will be approximately constrained by even a single detection, while the "stretch" of the model is similarly constrained by the need to fit several points. However, the shift can be very high if the SN was discovered later in its evolution, due to the way we set the zero point. (The other two parameters can be approximately correct even if the SN was discovered late.) In either case, all of our features need to be scaled as the possible values for shift vary between -100 to +100, the scale factor can vary between -10 to +10, while the stretch is tightly constrained between 0.1 and 3. Since we are originally doing unsupervised learning, we have no labels for our data. However, our g-band validation set has 3 labels derived from intuition as specified above. We will not try to get our data to fit in to that arbitrary classification scheme, (and end up doing supervised learning.) We would like to discover the labels from the data itself.

The quest for a suitable cost function is more difficult. However, we went with the K-means algorithm (and its sum-of-squares minimization cost function; Eq.1) for several reasons. First, we naturally expect to find at least one cluster of normal SNe, which are known to look like each other. We would then like to know if there are any other relatively cleanly separated clusters in the data, which the K-means method can find rather easily. These spherical clusters could serve as a means of sub-classifying the data. Furthermore, this algorithm is guaranteed to converge, is rather robust, and is generally applied when searching for clustering in data points that tend to be close to each other but well separated. If the SNe come from different progenitors, we could expect such a scenario.

For the hierarchical algorithm DBSCAN, the cost function essentially tries to minimize the distance between nearby points. (See Ester et al. 1996 for details.) It finds cores that are more dense than the surrounding, and grows them under the constraint of the cost function. it also has a noise model, and whatever it does not accept in to the cores, it labels as noise. It is an agglomerative procedure. Since we are interested in finding dense regions representing possible different SN sub-classes, we could have used many alternative cost-functions. For example the Gonzales Algorithm finds the maximum radius of a cluster by minimizing the sum of square errors from the center of each cluster. However, if our SN classes are not spherical, this is a weaker method.

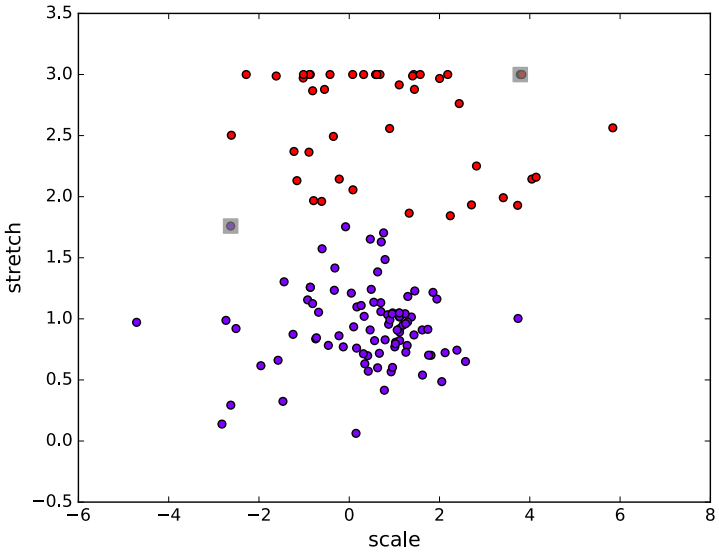
The two algorithms have a number of hyper parameters, however SKLearn library has suitable default values for most of them. For K-means, the randomly chosen cores, their distribution and number of randomly taken core samples, some parameters for how many times to run the algorithm along with maximum number of iterations before each run has to converge, the tolerance for conversion, are all hyper parameters that the KMeans function uses with some default values. For us the most important hyper parameter, which we vary, is the number of cores to use. The DBSCAN algorithm also uses several hyper-parameters when executing that use some default values, however there are 2 important ones which we vary. The minimum number of clustered points to use to start a high-density core, and a tolerance parameter (Epsilon) for

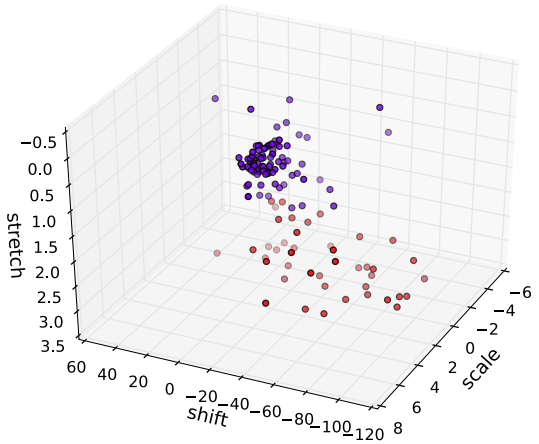
accepting or declining nearby points in to the core. In effect, the higher this tolerance parameter, the greater the distance between points which can be added to a single core. So if the data are linked, a very high value for this parameter can cause a slightly over-dense region to dominate in core size, but a too small parameter will reject most lower density cores as well as labeling many points as noise. For the number of clusters hyper-parameter from K-means, and the Epsilon parameter from DBSCAN, we used a few different values and show its effects on our data in the figures below. The optimal values depend very much on the distribution of our data and the results obtained. In this regard, having a three dimensional dataset was a positive, since we could easily see the effects of varying these parameters on the entire dataset it its space.

To evaluate our results, we use the previously identified broad and poorly-observed SN in the g-band to essentially train our algorithm parameters. We look for the clustering algorithms to separate these from the normal SN cluster. However, we did not have a true validation set. But we finally use the identified SNe (including the normal ones) as a test set, to confirm that they end up in a single cluster of "normal" SNe. Since we had a three-dimensional dataset, we can also visually inspect our results. We show that looking at 2D slices of a 3D distribution is already misleading, and the effects would be even greater with higher dimensional data. So in that case, both a training and a validation set, as well as a test set is important. In the future, this dataset can be used as the training and validation sets for an even larger SN light curve dataset, where we train with about 70-80% of our data, and use the rest for validation to prevent over-fitting.

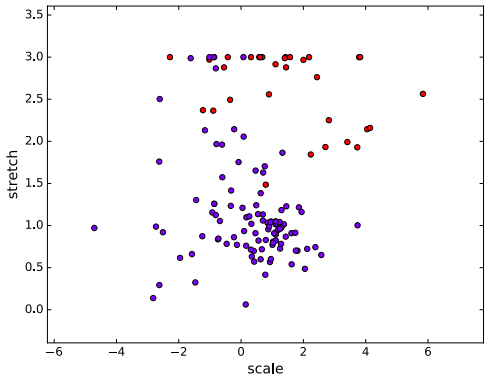
4) Applying the Algorithm and Results

We run both the K-means and DBSCAN algorithms. For the K means, we first only use the stretch and scale features (based on our intuition about the shift feature being potentially misleading). Here, the squares are the broad and poorly-observed SN from left to right. We can see that

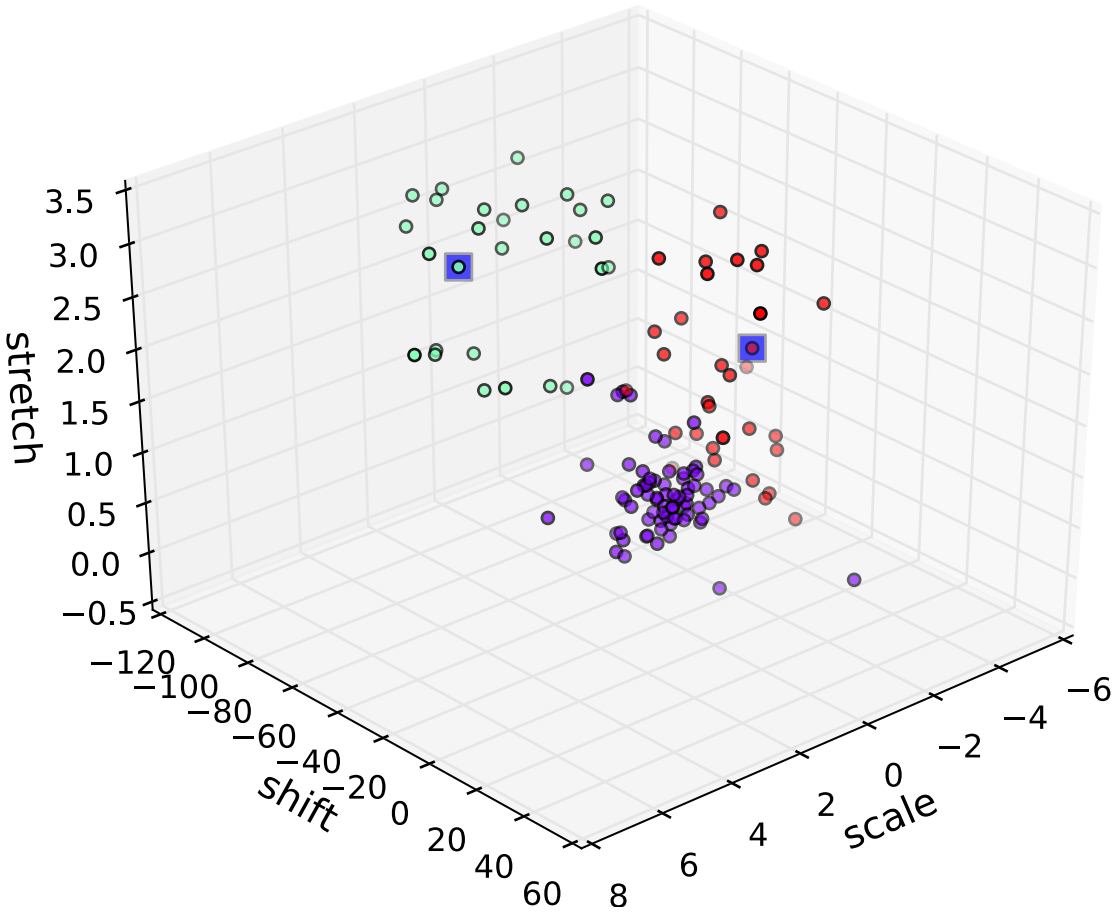




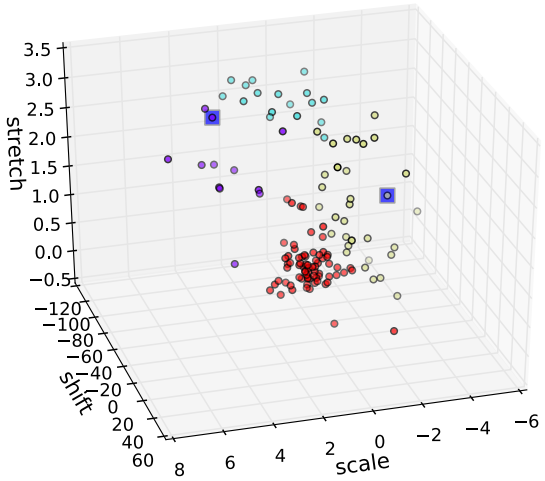
a bivariate distribution is found. However on the right, a 3D visualization with all 3 parameters shows that there may be further structure within our data. Now when using the shift parameter as a feature as well, we see that a simple 2D visualization will not be sufficient (below). Similarly, while the two training cases are still in separate classes, they are not distinguished from the core of normal SNe (in blue/purple) with a 2 core model.

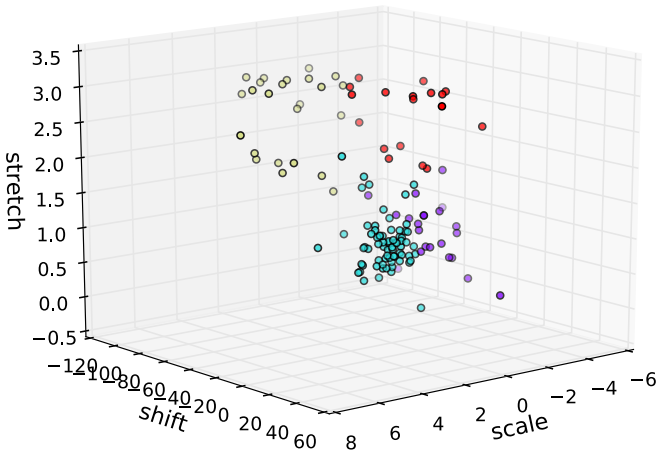


On the next page, we use a 3 core model, and we can see that our broad and poorly-observed SN are in their own respective classes, separate from the purple core of normal SNe. However, it did seem like that labels were somewhat mixed. We then decided to try with 4 cores to see if any further subclassifications could be found without over-fitting.

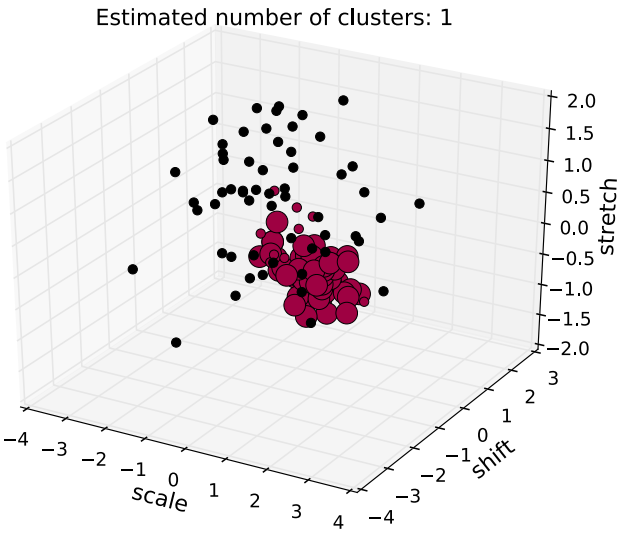


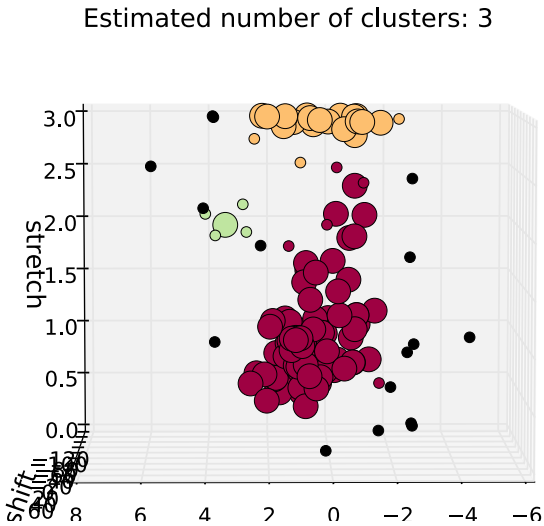
The two figures from two distinct runs of the 4 core K-means algorithm are below. In both of them, the normal, the poorly-observed, and the broad SN have different labels, (and thus are in different classes.) Though the two figures illustrate that the boundary between the classes may not be very exact, or that even more classes but with much fewer members may exist in between.



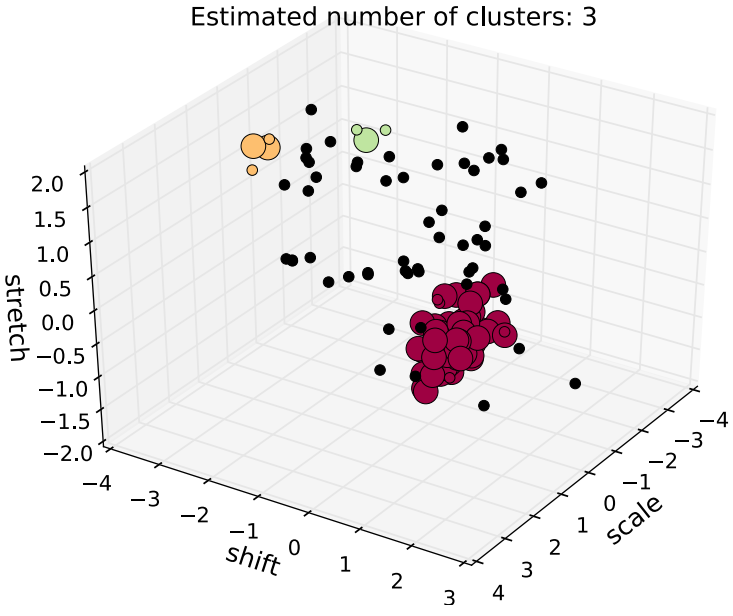


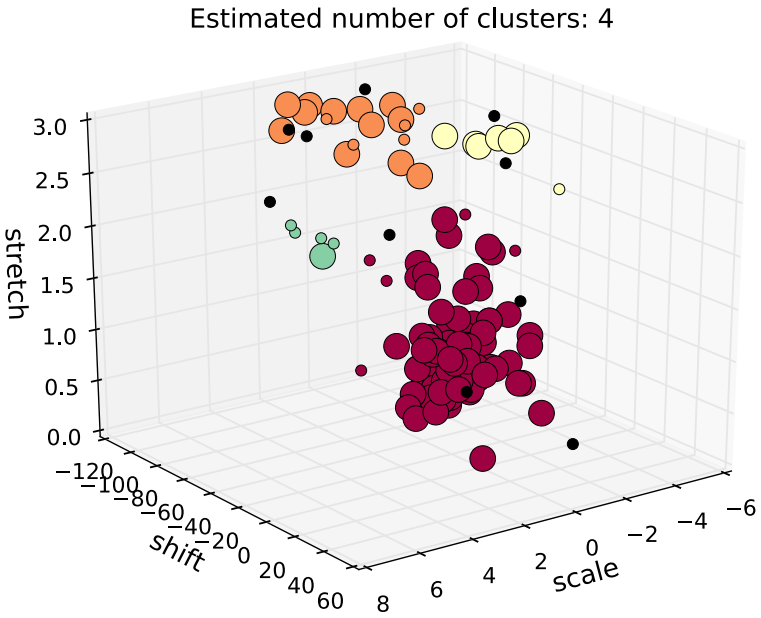
Finally, we show DBSCAN, the hierarchical algorithm run with several parameters. With a high value for the number of samples needed to form a core, and a low value for Epsilon, we obtain a single cluster as shown the right. However, collapsing the space in to 2D and not using the shift parameter for a similar set of hyper-parameters gives us a different result shown below.





Finally, we use all 3 dimensions of the dataset with the default value of the number of points needed to start a core, but first a slightly lower Epsilon tolerance value, (below left), and then a slightly higher epsilon tolerance value (below right.) We get an estimated 3-4 cores for these values of epsilon (from 0.3 to 0.9)
It is interesting to note that the first DBSCAN is also quite a good fit to the data. It might just be that we have





normal SN, and what is essentially a "noise"
of more unique different ones. None of the other classes are as convincing as the cluster of normal SN correctly identified as a single core in all of the DBSCAN clustering algorithms, as well as in all of the K-means algorithms. Physically we can conclude that there are a large group of stripped-envelope SN with light curves that all are very similar. However, there also exists quite a few others, which may or may not be part of another sub-class, with a large spread in light curve parameters.

Eq.1

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_j - \mu_i||^2)$$

Reference:
Ester M, et al. 1996. aaai.