Purdue University

Master Thesis Report

Feature extraction and classification on Time Series

Author:
Miguel Méndez

Supervisor:
Bruno Ribeiro

June 29, 2017

Abstract

Time is involved in almost every scientific field one can think on. Observations of a phenomena are collected with the aim of study or explain its behavior. This collections lead to organized data called time series.

Data mining community has spent a reasonable amount of time studying time series, in order to extract all meaningful knowledge from them. Humans are generally good comparing time series, but still, our capabilities are not scalable and we need to design algorithms and techniques that allow us to deal with high dimensional data and other problems.

In this work we will focus in a specific problem, extracting valid features of unlabeled time series obtained from aircraft sensors. These must serve as a summary of a flight and they also must include relevant details that serve to characterize it. This information will be used to feed an algorithm which can learn to classify flights in groups, reducing the number of necessary labeled data to obtain the desired accuracy using an active learning approach.

Contents

1 Introduction

The main goal of this project is being able to classify hundreds of flights in different groups or clusters, using time series data obtained from the on board sensors of the planes. One of the main problems we will face is the absence of labeled data, for this reason it will be necessary to design and implement an unsupervised learning method which can allow us to extract conclusions and patterns from each of the different clusters.

We could for instance classify them by pilots skills depending on the information recorded by the aircraft sensors. For example, a experimented pilot knows how to reduce the bouncing of the aircraft during landing or what to do in a windy situation in order to avoid problems related with pitch and roll. Hence, flights containing characteristics like this should be clustered together and labeled, for example, as *safe flights*.

In this work we will use a dataset that consists of 1068 flights. It is worth it to highlight that our data embraces very different characteristics, some flights show a classic pattern (take off, cruise and landing) while as other ones correspond with instruction flights where cruise time is minimum and multiple take offs and landings are carried on. More information and descriptions about the available data will be given in 2, where we will also talk about the different parameters we can use, which include from position and velocity information to engine and fuel state.

One of the most important task in Machine Learning is to do a proper data preprocessing. In this project we need to convert all the previous described information into a manageable one that can be used to feed and train our algorithm. This feature extraction procedure will be covered in Chapter 3 where we will explain what Dynamic Time Warping is and how have we applied it on this project.

Once we have simplified our dataset we can proceed to the clustering part, which will be covered in Chapter 4. The clusters obtained after this unsupervised technique will serve us as the starting point of the Active Learning approach we have developed. The aim of this phase is to select those flights that contain more information, this is because they are either cluster centroids or important outliers, and ask an expert about them. In this way, we can do a good classification without labeling the whole dataset and maximizing the obtained information. More details about this will be covered in Chapter 5.

2 Dataset

In this second chapter we will introduce and study the dataset we have available for achieving the different targets of this project. When we speak about Machine Learning or Data Mining we have to put special attention in the data and we need to study and understand it as much as possible before deciding the next steps we want to follow.

We have a dataset with a total of 1068 flights, that contains on board sensor information. We can, for instance, cite altitude, pitch, roll, pressure, outside air temperature or ground speed, between many others. The total number of sensors depends on the aircraft model, since, some of them can for example have more engines than others. In this project we are not going to use specific aircraft information. In other words we will assume that all of them are the same model. For this reason we are only going to use those sensors that are common to all planes.

Let's now to take a look into the real data and see how is it structured and what kind of information do we have available. In table below we can observe a sample of sensors that are related with the position of the plane:

Lcl Time	**Latitude**	**Longitude**	**AltMSL**	**AltGPS**	**AltB**
hh:mm:ss	**degrees**	**degrees**	**ft msl**	**ft wgs**	**ft Baro**
8:26:15	40.415	-86.935	599.4	487.2	677.1
8:26:16	40.415	-89.935	598	485.8	676.1
8:26:17	40.415	-89.936	597.7	485.5	676.1
8:26:18	40.415	-89.936	597.5	485.4	676.0

Table 2.1: Location related sensors sample

We can observe that each of the columns represent a different sensor in a time series format. Time series can be defined as a collection of observations made sequentially in time. They are used in stats, signal processing, econometric, weather forecasting and almost everywhere. They are usually studied in order to obtain information about the underlying process, to describe the features of the series and to be able to make predictions about future time values.

Left to right we have local time, latitude, longitude, altitude over mean sea level, altitude GPS and barometric altitude. The second column corresponds with the units used by each of the sensors and the rest of them are just the sampled measurements. This table can serve us to give a brief idea of the amount of time series we have per each of the flights and that some of them are very correlated, we must think that for example the three

altitudes we have collected in the table show the same pattern since they are measuring the same using different scales.

Nevertheless, not all of the sensor data we have is as easy to process and understand as the one collected in the previous example. For us, as humans is very easy to think in terms of positioning and imagine how is the plane behaving by for example looking for abrupt changes in the altitude. One of the advantages of working with time series is that we can plot them an have an intuition of how has a certain fight evolved with time. If we for example plot the altitude over mean sea level data (see figure 2.1) we can clearly distinguish the three different phases of the flight repeated two times. We can then think that the pilot could have done a refuel or break stop and then continue his trip.

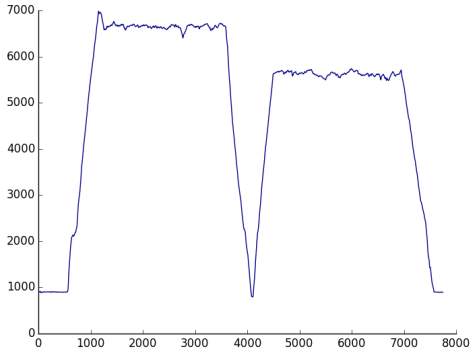


Figure 2.1: Altitude over mean sea level time series

But what would happen if we focus in more complicated and abstract time series, where information extraction is a difficult task even for experts in the field?. In order to understand better this problem let's take a look in the following table where we have collected a new set of the available sensors.

deg	**deg**	**deg C**	**gph**	**Hg**	**deg F**
Pitch	**Roll**	**OAT**	**E1 FFlow**	**E1 MAP**	**E1 CHT1**
0.11	0.58	-9	2.42	16.4	234.9
0.13	0.59	-9	2.4	16.44	235.29
0.12	0.57	-9	2.4	16.45	235.64
0.11	0.58	-9	2.4	16.46	235.51

Table 2.2: On board sensor time series sample from database

From left to right we have pitch and roll, which are related with the inclination of the plane in the vertical and horizontal plane. We can also see outside air temperature followed by three different sensors which retrieve

information from engine one. These are fuel flow consumption, manifold absolute pressure and cylinder head temperature.

Clearly, for this type of sensors we will require expert knowledge, since they are very specific and measure with different units. If we, for instance, try to see what is going on with the pitch information in the same flight that we show in figure 2.1 we will find the following plot:

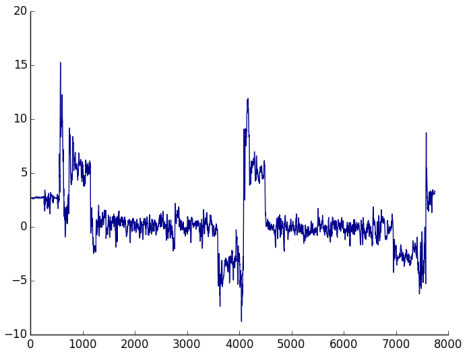


Figure 2.2: Roll sensor information

We can easily observe a correlation between both graphics, since peaks and depressions are equally located, but it is difficult to know what is going on during the rest of the trip. This fact leads to one of the most important points of this project, we need to find a metric that allows us to compare time series. With this information we can summarize a flight using just a small number of features and feed a machine learning algorithm.

In the following chapters we will cover these new targets and we will explain how have we proceed and what decisions have we taken during this project. Nevertheless, before moving to the next chapter, it is important to remember a problem that we have previously mentioned. Though we decide to discard some of the sensors and work only with a subset of them that are common to all of the flights, the flights have very different characteristics between them. One of this important differences is their duration. Some of them can last for hours, while as another ones can be very short leading to time series of very different length. Our metric must be able to compare them without care about the length, this fact will clearly reduce our options making our target harder to achieve.

3 Feature extraction

3.1 Goal

We must properly define the goal behind this second phase of our work which will determine our future work and results. As we said in the previous chapter, we must summarize our dataset which contains redundant and unnecessary information for our purposes. At the same time this summary must conserve most of its original properties so it can still represent the raw data. This new data representation must be robust working consistently with flights of diverse characteristics and durations.

The obtained new data will be used to feed an unsupervised learning algorithm. One of the priorities in the feature extraction procedure is to obtain a new data representation which structure can properly work with this type of algorithms. This has been the main reason that made us to decide that our new dataset would consist in a *pair-wise distance matrix*. These matrices are specially popular in graph theory, we can define them as square symmetric matrices containing the distances, taken pairwise, between the elements of a set. If there are N elements, this matrix will have NxN size.

Hence, each of the cells in our distance matrix will represent the distance (or dissimilarity) between two of the flights in our original dataset. In other words, the position M_{ij}, where i and j correspond with row and column index respectively, will contain the dissimilarity between flights i and j.

In order to create this distance matrix we need to define a metric or dissimilarity measure. It is worth it to note that it should be computed with respect to one or more sensors since, for instance, two flights might have very similar altitude data but very different roll data due to the weather conditions or to the pilot experience. So our selection should have into account the fact of including information from multiple sensors.

Each of the different sensors represent a time series. Therefore the metric we are looking for will allow us to compare time series of very different lengths in an efficient way.

3.2 Dynamic Time Warping

3.2.1 Introduction

Time series and their extraordinary ubiquity has resulted in important efforts from the data mining community (and other scientific communities) in

investigating and developing algorithms that can mine time series archives efficiently, retrieving fast and accurate results. The great of this effort has been mainly focused in a very specific, but at the same time very important, task. This is to find the best distance measure to use for a specific domain.

Many different techniques and methods have been proposed, but during the last years data mining community has leaned towards Dynamic Time Warping [1]. DTW is an algorithm for measuring similarity between two time series which may vary in timing. Among its interesting properties we must highlight the invariance to warping which in fact is the key behind this algorithm. It has been used in a large number of fields as robotics, medicine, bioinformatics, video games, music, image processing among others.

The bast amount of experimental research related with this algorithm can serve as a proof of its popularity. The variety of the different domains where it has been applied can serve as a proof of time series ubiquity. Examples of this can be the use of DTW in dogs activity recognition [2] or in the study of star light curves [3], which can serve to discover pulsars or extra solar planets...

Nevertheless DTW has a very important drawback, the original implementation is quadratic on time which can be a terrible characteristic when working with large volumes of data. Although our dataset cannot be considered as very large, we have high dimensional problem, so the quadratic complexity would affect our work. This problem has been the focus of attention for the community and has lead to significant improvements which have resulted in better and more efficient versions of the algorithm [4][5][6]. These techniques often rely on a deal between speed and accuracy, so we must decide which one of them are we more willing to sacrifice.

In the other hand, between its advantages we should mention that time series classification based on on Nearest Neighbor Dynamic Time Warping (NN DTW) is very hard to beat. Rival methods introduce high complexity and a large time and space overhead, while as DTW maintains itself simple, retrieving accurate results. In [7], authors show that the untenable lethargy of DTW clustering can be mitigated by casting it as a *anytime algorithm*. These algorithms trade execution time for quality of results but they always have a best-so-far answer available which improves with more execution time.

However the techniques proposed in the previous paper cannot be applied to our work, since our target is compute all pair-wise distances instead of the nearest neighbor, so we need to apply some modifications.

3.2.2 Algorithm

Dynamic Time Warping can be imagined as a special Euclidean Distance. Actually a special case of DTW is the original Euclidean distance. In general, DTW is a method that calculates the optimal match between two time series.

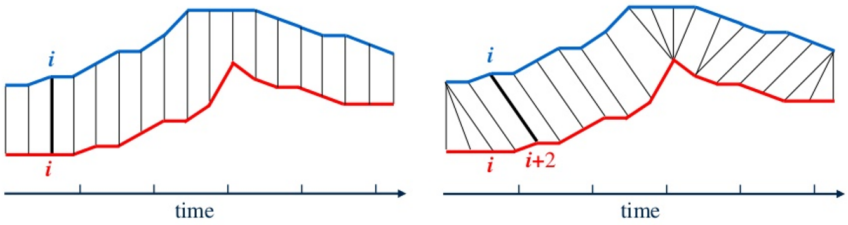


Figure 3.1: Euclidean distance comparison and Dynamic Time Warping, from Tsiporkova [8]

In figure 3.1, we can observe the differences between themin. In the left image we see the Euclidean distance match, where the i-th point on one time series is always aligned with the i-th point on the other. In the right graphic of the same figure we how DTW works. We can appreciate an elastic alignment, which results in a more intuitive similarity measure. For doing this, the series are warped non-linearly in the time dimension so the similarity of them will be independent to non-linear variation in this dimension.

The computation of Dynamic Time Warping is based on a dynamic programming algorithm inspired in the ones that were developed to solve the DNA sequence alignment problems. The first step is to create a matrix with of size mxn, where m represent the length of the first time series and n the length of the second one.

Each of the cells of the matrix will be filled with the cumulated distance between every pair of points we can extract from both time series. The recursive function that determines this cumulative distance and the dynamic programming algorithm is the following one:

$$\gamma(i,j) = d(q_i, c_j) + min \begin{cases} \gamma(i-1, j-1) \\ \gamma(i-1, j,) \\ \gamma(i, j-1) \end{cases} \tag{3.1}$$

When the whole matrix is filled with the distances, we can find a path on it that will represent the distance between both time series. We must think that every possible warping is a path trough the matrix so to find the path between c and q one need to find the path through the grid which minimizes the total distance between them:

$$P = \arg\min(D(c,q)) = p_1, p_2, \ldots, p_s, \ldots, p_k$$

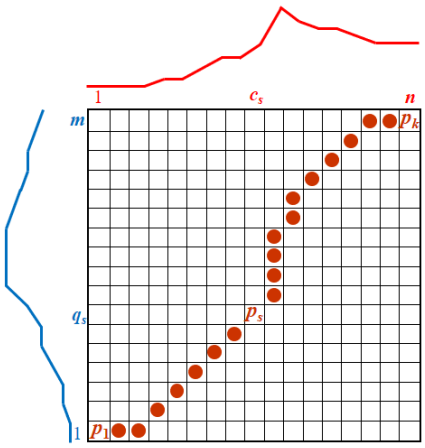


Figure 3.2: DTW matrix computation from Tsiporkova [8]

This the the original version of the algorithm in its recursive version. Nevertheless the associated complexity and the memory cost related with this implementation, are too high. The computation of DTW using this naïve version of the algorithm would take years to compare our flight data. So we will not use this recursion in practice, we will explore different optimization and tricks, that have been developed to deal with this problems.

So let's now move, once we have understood the utility of Dynamic Time Warping and its basic implementation, and learn how to compute in a fastest way. We will discuss two important optimization that allow us to obtain results in manageable time.

3.2.3 DTW Optimizations

There are two main optimization of DTW algorithm applied in this project that must be mentioned. The first one is a simple technique that allows to reduce the number of necessary computations to obtain the distance measure. The second one, solves an important problem of the original algorithm. We will also explain how to reduce the memory use and use parallelization to take advantage of cpu power.

Warping Constraint

The warping constraint, also called Sakoe-Chiba band (S-C band) [6], shown in figure 3.3, is the most popular method to constraint the matching between two time series. It was initially developed for speech community but it became very popular mainly due to its simplicity.

The idea is to limit the possible matches, this is, that two points that that are very far in the time dimension, should not be matched. The constraint

runs along the main diagonal of the matrix and has a fixed (horizontal and vertical) width.

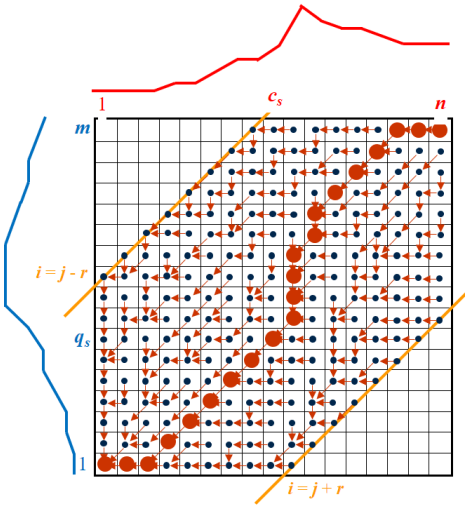


Figure 3.3: DTW constraint effect from Tsiporkova [8]

This constraint implies that an element x_n in the first time series can be aligned only to certain elements of the second one. In other words, the warping constraint represents the maximum amount the warping path is allow to deviate from the diagonal. Typical values are around the 10% of possible deviation, but it must be specifically tunned depending on the data [9].

This tunning procedure is usually based on cross-validation approaches. However our dataset lacks of labels and classes, so we cannot use a validation set to check the performance of different bands. For this reason we will manually tune it, by observing the relation between the different sets or clusters that will be created, after applying our unsupervised learning algorithmn.

Another important detail is related with the length of the time series. An avid reader would note that adding this new constraint will require both time series to have the same duration. One easy solution for this problem, is to apply a simple interpolation to the smallest time series. In this way we can re-interpolate the time series to have the same length without experiencing differences in our results.

Endpoints importance

This second optimization is based on [10] work. There the authors explain how they noted an importance detail that degrade the performance of DTW when was applied to real world datasets. As the authors explain in their work, the issue is that DTW's eponymous invariance to warping is only true

for the main "body" of the two time series being compared. However, for the "head" and "tail" of the time series, the algorithm affords no warping invariance.

The applied optimization consists simply in modify the endpoint constraint of the algorithm, in order to provide endpoint variance. Thanks to this DTW will be able to ignore some leading/trailing values in one or both of the two time series under comparison.

Although it can look like a very simple idea, the importance of it in this work is indubitable and, all of this without adding new complexities.

In Figure 3.4 we can understand better this endpoint importance and see how a little difference in the beginning of the time series will increase the total obtained distance.

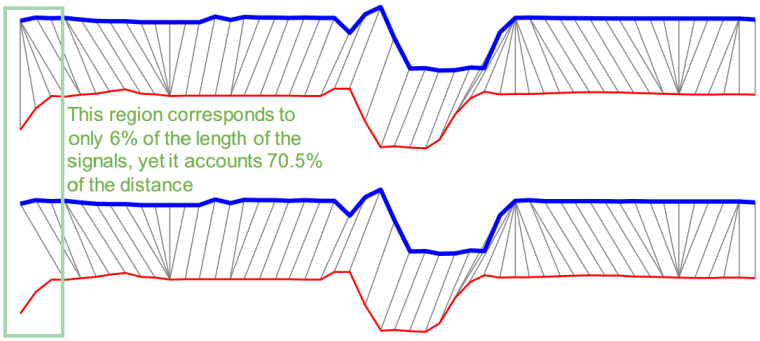


Figure 3.4: Endpoint effect from Silva et al. [10]

3.3 A different application of DTW

After all the above, the reader should have a clear idea of what Dynamic Time Warping is and the advantages that can proportion its use. Besides, one can start to imagine new ways where to apply this method.

The warping step, allow to match similarly shaped time series, even if they are out of phase. Think for instance in the sin and cos functions, which could be warped and matched perfectly by this algorithm.

This fact motivated another experiment during this project. Apart from the raw data collected by the on-board sensors of the plane, we developed an Android application that could retrieve information from the smartphone sensors, as the GPS, accelerometer, barometer, etc. The idea behind this was to simplify the way of obtaining data and also to increase our dataset. Nevertheless, we wanted to compare the accuracy and the difference between this two ways to capture data, and for this we use DTW.

An example of this can be observed in figure 3.5 located below. There we can see a total of three different plots.

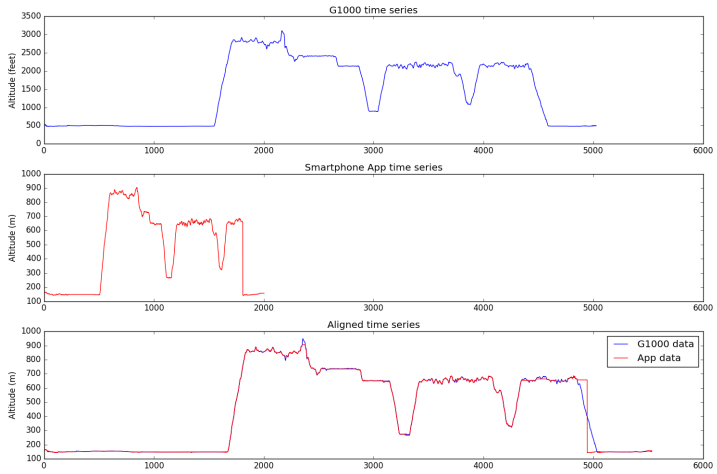


Figure 3.5: DTW matching of smartphone data and on board plane sensors data

If we look them from up to down, we will see first the time series that represents the altitude data recorded by one of the on-board sensors of the plane. We can clearly distinguish three different flights (takeoff, cruise and landing) in there.

The second or middle one, collects also altitude data of exactly the same flight. But this data has been recorded by the smartphone sensors through the application we have specially designed and developed for this project. These data are more sensible to noise and also the capture frequency is much lower than the one achieved by the on-board sensors.

Finally, the bottom and last graphic shows the alignment or matching returned after applying DTW between the two upper time series. We can see that the match is pretty good and that most of the time both time series are merged. Thanks to this method we can also compare time series and evaluate the amount of noise.

3.4 Other approaches

Although Dynamic Time Warping is the best method we have found for our purposes, during the development of this project we have studied and implemented other algorithms that could help to extract valid features from the flight dataset, different from the distance matrix approach.

The obtained results were not satisfactory and far from the values obtained by DTW but we still will mention two of them. We want to explain why we decide to use them and why did they not work or perform bad, to motivate future research.

The first one of them was based in Hidden Markov Models (HMM). The

main idea behind it was to use a similar approach to the one used in speech recognition [11]. Using a HMM we could obtain a transition matrix that we will use as a feature set, this would result in a non-intuitive feature set which "luckily" would summarize our flight.

The obtained transition matrix would be different each time we learn a HMM, they states of our model are represented as rows and this are not required to be constant, so we would need to find a rotation and shift invariant decomposition for our matrix. Nevertheless, it was not necessary to go so far, since HMM have problems dealing with our type of data. Markov assumption does not fit well with our type of data due to the large size of our time series that tend to adapt constant states during long times (specially during the cruise phase).

In order to solve this problem we also developed a method based on Linear Dynamical Systems(LDS) with the same goal as for the HMM case but with some important variation. Now the hidden states, as well as the observed variables are multi-variate Gaussian distributions. This model also show problems learning the patterns of the flights so we decide to not go further with it.

4 | Unsupervised Learning

Throughout this chapter we will explain in what consists our unsupervised learning approach for this first part of the project. The previous chapter has served us as an introduction to the problem, to understand it and to transform it into a valid representation. This new data format, will be used to feed our clustering algorithm. The intention behind this is to find hidden relations between flights, that we could not appreciate before, having into account the vast amount of data we had to process.

In an ideal situation, we could for example obtain different clusters, where each of them would be, for instance, directly related with the skills its the pilot. So our algorithm could detect those unstable and short flights, with numerous take-offs and landings, as training flights, and classify them in a single cluster. In the other hand, flights with a long cruise phase would be related with more expertise pilots. It is worth it to note, that this is not our real target but it can serve as an example to understand why do we need this clustering step.

The final target is then, to obtain some labels (the different clusters), that can serve us to classify flights by certain conditions, with the aim of applying later an active learning step, with experts mediation, that can improve the learning and the final accuracy. We will study this in the next chapter, let's focus now in the clustering of our dataset.

4.1 Input data

The input data we are going to use for feeding this algorithm is the all-pair wise distance matrix that we describe in the previous chapter. In this matrix each row will contain the Dynamic Time Warping distance between one specific flight and all the rest.

Therefore each instance or flight will be represented with the set of distances with respect to the other flights and itself. This fact implies that one specific column j will also contain the pair wise distances between the flight j and all the others. We are then dealing with an Hermitian or symmetric matrix whose main diagonal is formed by zeros (the distance of a flight with itself is zero).

In table below we have collected a small and reduced sample of our dataset with the intention of facilitate the reader's comprehension:

	Flight 1	**Flight 2**	**Flight 3**	**Flight 4**	**Flight 5**
Flight 1	0	0.21	0.45	0.62	0.34
Flight 2	0.21	0	0.3	0.15	0.1
Flight 3	0.45	0.3	0	0.13	0.53
Flight 4	0.62	0.15	0.13	0	0.712
Flight 4	0.34	0.1	0.53	0.712	0

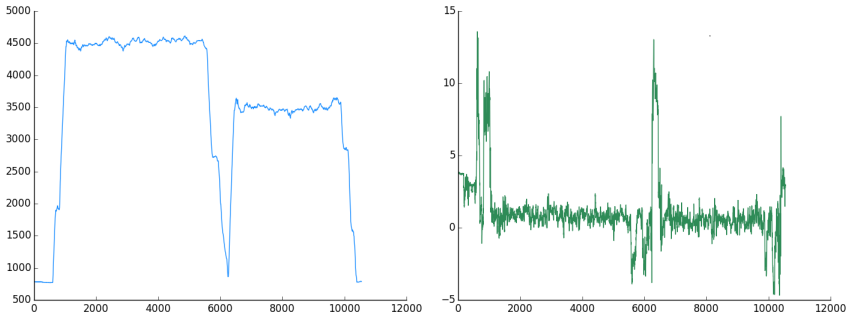
Table 4.1: Example of the all pair-wise distance matrix

Nevertheless we have to remember one important fact that he have been obviating for maintaining the problem as simple as possible.

We must remember that we have more than one time series per flight. We actually have tens of them. Also we must remember that this information was collected from multiple sensors. The distance matrix is at this time being constructed with the data of a specific sensor, for instance, the differences in the altitude data.

However, if we only take information from one of the sensors, our results will be biased, since we will put away valuable data. Let's think that we only use altitude data. In this case, two flights might look very similar because the realized the classic three phases, take off, cruise and landing. But this similarity might be far from being true. If we have a look on pitch or roll data, we might find that the one flight experienced a lot of turbulences while the other one was very soft.

We need then to combine as many sensors as possible to make our results as consistent as we can. If we observe figure 4.1 we can clearly see the differences between each of the sensor time series and the different information that they provide. In the other hand, if we combine all of the sensors, it will be hard to interpret the obtained results. If we observe once more the figure below, we will easily understand the altitude time series but we hardly will know what happens with the pitch or the roll.



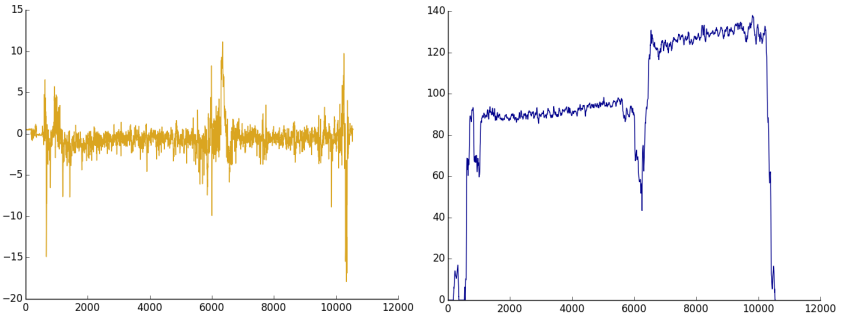


Figure 4.1: Different time series obtained from sensors. From left to right and up to down we have Altitude, Pitch, Roll and Ground Speed

One plausible option for tackling this issue could be to compute a different matrix for each of the sensors we want to use. Once these matrices are obtained we can simply do a weighted combination of all of them and use the resulting matrix as our dataset.

Imagine we have a total of n sensors we would like to use. We would then obtain a total of n different distance matrices D_i with $i = 1 \dots n$. Finally we could combine then applying a normalized weight w_i, between $[0, 1]$, to each matrix obtaining a weighted average of them.

$$X = w_1 D_1 + w_2 D_2 \dots + w_n D_n \tag{4.1}$$

We could also give more importance to certain sensors which might be more useful to classify the flights in clusters, or give the same importance to all of them.

4.2 Agglomerative Clustering

It is time now to describe the unsupervised learning technique we have applied. We basically decide to use Agglomerative Clustering [12], which is one of the simplest clustering algorithms we can find in Machine Learning literature. It a specific type of hierarchical clustering, which aims to build a hierarchy of clusters as the own name indicates.

There are two different fashions inside Hierarchical clustering [13], the agglomerative on which consists in a "bottom-up" approach, where each observation starts in it own cluster and pairs of clusters as one moves up the hierarchy. The second one is called "top-down" or divisive fashion, which consist exactly in the opposite. At the very first moment all observations belong to the same cluster, which is recursively divided with time until certain conditions are reached.

The decision of using this method is motivated mainly by its simplicity. It is one of the most basic clustering algorithms and with the lowest require-

ments. The use of more complex and popular methods is also possible. More important algorithms as K-means can be proposed as future work but we

We decide to use this method for its simplicity and low requirements. Nevertheless, more complex and exigent algorithms as K-means can be also used and it is proposed as future work.

4.2.1 Choosing K

When we deal with unsupervised clustering techniques we have to deal with the the unknown number of classes or labels. We do not now how many clusters do we really have in our dataset but we need to figure it out, or almost get as closer as possible. Otherwise, the obtained results will not reflect the properties of our data.

Most of the algorithms require the number of cluster as a parameter, this is, that we must preselect the number of clusters beforehand. Since we do not have any labeled data at this point we will require some kind of metric that can tell us how good or accurate is the obtained clustering. We can then use this metric to compare different results and have certain security in our final decision.

In this project we will rely in the Silhouette coefficient which refers to a method of interpretation and validation of consistency withing clusters. This metric compares the intra cluster similarity, this is the similarity between all the points that are in the same cluster, with the highest similarity to any other cluster. Let $a(i)$ be the average dissimilarity of i with all other data within the same cluster and $b(i)$ as the lowest average dissimilarity to any other cluster of which i is not a member. We can define now a silhouette:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{4.2}$$

Therefore if we want to decide the optimal number of clusters, we can compare the obtained silhouette coefficient for each different k we test. This is what we have done and in figure below we can observe the results for different values of k.

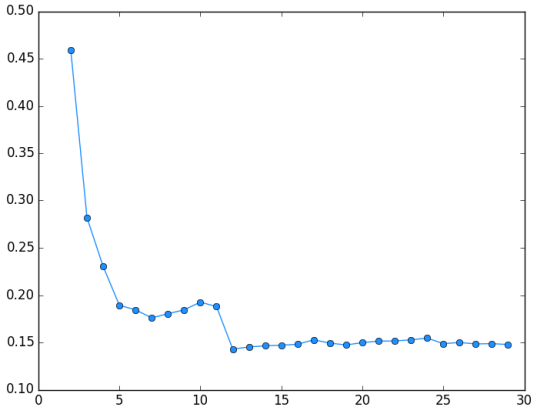


Figure 4.2: Number of cluster vs silhouette coefficient

The obtained graphic can be confusing and we need to study it carefully. In a first glimpse we can think that the best results are obtained when the number of clusters is reduced. Actually, the best results so far, are obtained when $k = 2$.

Nevertheless, we would like to obtain a higher number of clusters, since the target of this clustering step, is to obtain valid labels that we can use in the future active learning approach. If we use only two clusters, it is going to be difficult to highlight the differences between flights. A second glimpse to the previous figure, would note that there is another peak around $k = 10$. After this, the coefficient decreases and keeps itself stable.

Let's now observe with more detail what is actually happening with these two clusterings. This can be done in two different ways. The first one is visually examine and compare the flights in the clusters and try to determine if there is actually a relation, but our data is too large for this. The second approach is to examine the number of flights per cluster and the individual silhouette coefficient for each of them. Clearly, we will follow this second approach to obtain some insights about the clustering we have done.

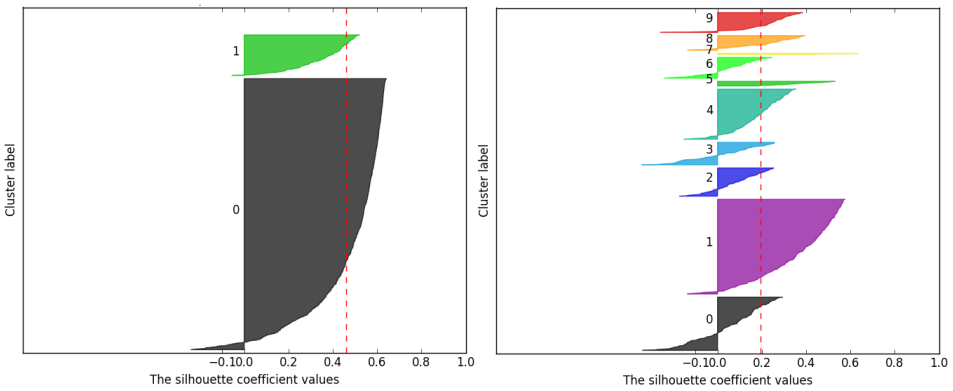


Figure 4.3: Silhouette coefficient details for K=2 and K=10

The left plot corresponds with the silhouette coefficient for a total of two clusters. We can observe the silhouette coefficient per each of the flights, which can be positive or negative. The dashed red line represent the average between all the coefficients we have obtained. We can see two different colors which are related with each of the clusters. If we focus in the size of each of the clusters, we realize that the cluster number zero has almost five times the size of cluster one. This unbalance does not benefit us, since we are looking for very detailed classification, trying to split the flights as much as possible but conserving also the most common characteristics that can form a cluster. Nevertheless we must note that, in this case, both clusters show compactness, this is, that only a few samples have obtained a negative silhouette coefficient, which might indicate that those points have been assigned to the wrong cluster.

The right plot shows the same summary for the silhouette coefficient for ten different clusters. In here we see a different pattern with many more negative samples than before. Also the average, represented by the dashed line, is considerably lower than in the left plot.

We must note that the silhouette coefficient is not perfect since it tends to return better results for those cases where the number of cluster is very low.

However we must remember that we are looking for labels that we can use as a starting point of our active learning procedure. Using a total of ten clusters, we would find very well classified points (related with high silhouette coefficient) and also outliers (negative coefficient). We would use this data points or flights as a starting set. We have to imagine that a very well classified point can be considered as the centroid of a cluster. In the other hand, a very bad classified point might indicate that we should create a new cluster for it and another similar points. The idea is to ask, to an oracle or experts in the field, about this specific points that contain a lot

of uncertainty, so our algorithm could learn a lot with the minimum labeled data.

For this reason we can conclude that $k = 10$ seems to be a good starting point for us. Clusters are better balanced and we will correct the higher number of wrong classified points in the future.

4.2.2 Clustering results

At this time, we have decided the unsupervised learning procedure we are going to use, which is agglomerative clustering. In the previous section we have explained how to choose a value for the number of clusters k. This parameter will be very important at this time, and as we said in the previous section, we have chosen to run the algorithm with a value of $k = 10$.

We will now show some details about the obtained results after applying the cluster procedure. The idea is to visually observe if there exist some relation between the flights that have been assigned to the same cluster and also to distinguish the difference between flights from different clusters. We must understand the importance of both, if flights in different clusters look very similar, is probably due to we are using a value of k greater than necessary. In the other hand if flights belonging to the same cluster, does not share a similar pattern we will need to increase the value of k.

For simplicity reason and with the aim of easing the interpretation, we will show only the Altitude GPS sensor, which can be easily understood. We just need to imagine the flight of the aircraft in a two dimensional plane. For this we will show a random sample obtained from some of the clusters. We have collected four different flights that have been classified to a certain cluster so we can compare them and analyze if they should actually be in the cluster or not.

The first of this examples are collected in figure 4.4 which represents the cluster number zero. Remember that we are using a total of 10 different clusters. The similarity between these flights is indubitable and obvious. But let's study them a little more in detail. The four of them consist in flights were there was a stop and then, they continue flying. Another important detail that we must focus on, is if there exists a bouncing pattern in the altitude. In other words, if the altitude data, specially during cruise phase, experience peaks and depressions instead of being almost a straight line.

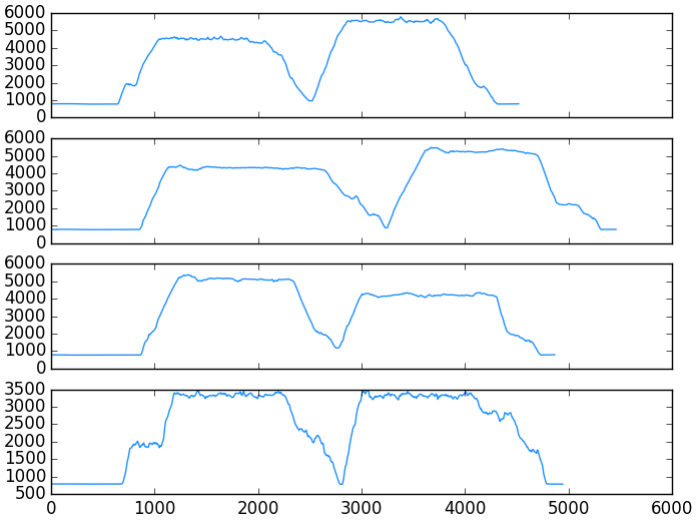


Figure 4.4: Sample of flights in cluster 0

This pattern might indicate that some sensors, as pitch and roll, experienced abrupt changes in their values and this is usually related with weather related problems but specially with the inexperience of the pilot. In this previous graphic we see that the four flights have a clean pattern, though maybe the bottom one is considerable different from the previous three. Still we can conclude that our algorithm is properly working up to this moment, since this four flights should be always in the same cluster.

The previous example is very intuitive and we can give us an idea of how the algorithm is performing. We just need a glimpse to understand they similarities between those flights. However, this does not always happen and sometimes we need to spend more time researching about the patterns of the clustered flights to get important insights.

Let's now to observe figure 4.5. There we have a new sample that consist of four new flights which belong to cluster number three. It is interesting to appreciate the difference between this figure and the previous one. Though these flight look very different at simple sight we can observe some important relation between them. If we focus in the first one and the third (upper down), we see two flights where the cruise phase has been considerably large and that with a lot of variance in the stability of the altitude data. The similarities between flights two and four are clear, the simply show the same pattern.

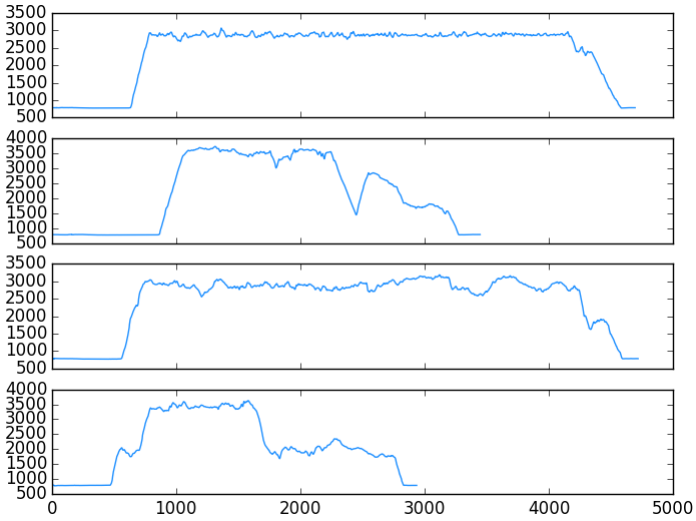


Figure 4.5: Sample of flights in cluster 3

But we can also relate the four flights together and find a pattern than wraps the four of us and that make the algorithm to classify them in the same cluster. The four flights have a very similar pattern but differently scaled, specially during landing phase. The pilots start to descend and the stop this procedure after some time for continuing with it later. Sometimes this stop is large, as for example in the bottom flight and some times is very short as for example in the upper plot. But Dynamic Time Warping deals with this time differences and retrieves the hidden similarity between them.

We must note that the differences between cluster zero and cluster three are also important and that we could not move a flight from one cluster to other, since it would not be related with the other flights. We can say that our algorithm is being consistent up to this moment.

We would like to highlight some curious fact that we have obtained unintentionally. After observing the different clusters we find a very special one. This is cluster number seven, with has a very specific type of flight (see figure 4.6)

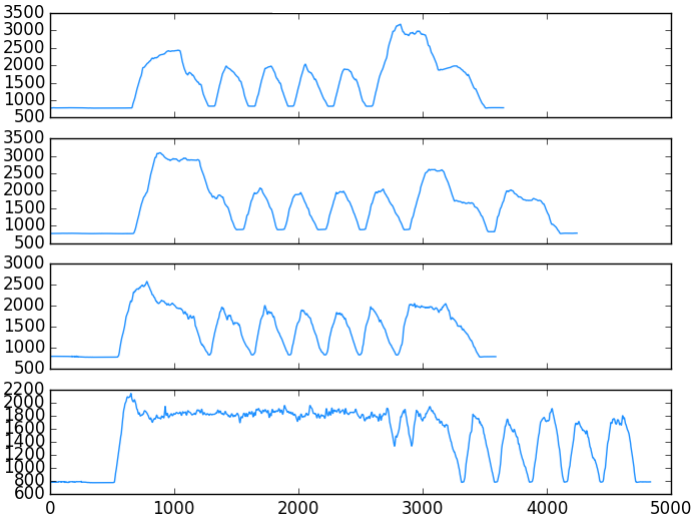


Figure 4.6: Sample of flights in cluster 7

This are strange patterns and they look very different from the previous examples. They are actually different from any flight we can think on and that is because these are training flights. They belong to flight lessons, where an experienced pilot shows a new one different techniques like taking off or landing and they practice them during hours flying in the same location. Our algorithm has been able to detect this pattern and has classified all this learning or training flights in the same cluster, putting together all those flights that have this special characteristics. This is a clear indicator of the correct performance of our method.

5 | Active Learning

This is the last and final part of this project and the most important at the same time. Since we started to deal with the problem of classifying flights, we thought about the use of active learning, to solve the issue of the unlabeled data. But although until this moment we have mentioned active learning numerous time, we did not proportion a valid definition yet.

Active learning is a special case of semi-supervised machine learning, where the learning algorithm is able to query the user or some oracle, to obtain the desired labels for some data points. It also known as optimal experimental design. The ideal use of this technique is in those situations where unlabeled data is abundant and manually labeling them is very expensive. The aim of the method is then to minimize the number of queries maximizing at the same time the performance of the algorithm [14][15][16]

In this project, the target is to create an algorithm that can classify our time series dataset. However, we do not have labels and we can manually label them because this will require a vast amount of time, even for experts in the field. What we can do is try to reduce as much as possible the number of flights that we need to label using this active learning approach.

It is worth it to note, that in our last step, we applied a clustering procedure to our unlabeled dataset. So, at this moment, we have classified our data into ten different clusters. We will use them as the starting point of our new procedure.

5.1 Using clusters as labels

We can imagine that each of the ten different clusters we have obtained is a class or label of our dataset. This means that we have ten different types of flights.

These labels will be used to feed a supervised learning algorithm which will be trained over the whole dataset. In other words, the new algorithm will try to classify the flights by clusters by finding some relation between the input features and the labels. This classifier must meet a condition, its has to output probabilities. We need to count with this special feature since, once our classifier has been trained trained it must be able to assign a certain label to a specific flight, and we would like to know which are the points with highest probabilities.

Once we have trained the algorithm with our dataset, we can use it once more to detect those data points that have achieved the largest probability in each of the clusters. These points can be considered as centroids, this is, that

they are a good representation of the whole set of points that are contained in a cluster. Having this into account we can use this set of centroids as the starting point of the active learning procedure and ask to label them first since this will maximize the amount of information that we are going to obtain.

Finally, we exchange the cluster labels with the new obtained labels and train another classifier with only this centroids and a very strong regularization. Then we will predict new labels for the whole dataset and once more we will ask the oracle to tag some new points. This process can be repeated as many times as wanted and the accuracy of our method can only maintain or increase, if and only the labeling step is properly done (we will clarify this later).

In the following section we will explain how to deal with some important problems as scalability. We will also give more details about the classifier and we will show the graphic interface we have developed for the labeling.

5.2 New data representation

Previously we have explained how did we transform the original dataset formed by tens of time series per flight, into a manageable one by using Distance Time Warping algorithm. We had used the all pair-wise distance matrix as the input dataset for training our agglomerative clustering algorithm. Nevertheless, we cannot keep using this data representation because, although it represents and summarizes the flights in a proper way, it is not scalable at all. We must think that, for each new flight we could receive, we would have to compare it with each one that we already had. So this would end in a unmanageable situation with time. Our idea is to reduce the number of comparison to a constant number and for this we will propose the following strategy.

We will randomly sample three flights from each of the ten different clusters. We will consider these set of flights as a summary of the entire cluster, so we can imagine that each set of three will act as a "centroid". After this step, the new feature vector will consist in the distance from one flight to each of these "centroids". Since we have a total of ten clusters and we extract three centroids of each them, the number of final features in our new representation will be thirty.

In a simpler way, if we have a flight f_i, its first feature will be the distance from f_i and the first cluster centroids of the first cluster, the second feature will be the distance with respect to the second centroid of also the first cluster, the third one will represent the distance with respect to the third centroid of the first cluster and so on. In figure below is collected a small scheme that might explain this idea better:

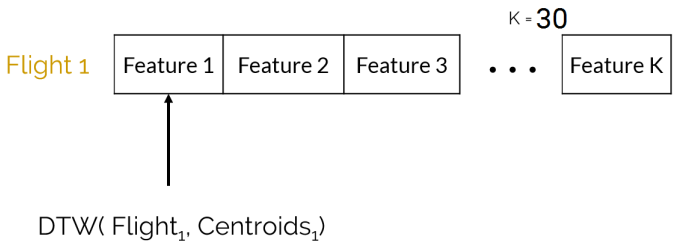


Figure 5.1: Example of dataset creation

It also worth it to note that after we use this new dataset of thirty features, to discover the real centroids in the clusters through the method described in section 5.1, the labels will be determined by the result of the query to the user. This means that the "fake" cluster labels that we had until this moment will be discarded. Nevertheless we decide to do not drop these data, and we just simply add the cluster as a new feature in the traditional one hot encoding format.

It is important to insist in the necessity of applying these new representation to reduce the complexity and make this solution feasible for larger datasets which we will have to deal in the future.

5.3 Classifier and centroids

When we are working with clusters we must always think about their centroids. These are special points which represent either, an average of the cluster, or the most representative points on it. If we find the centroid of the clusters and we ask for its real label to our oracle, we will obtain very important information which can be useful to merge clusters and it can facilitate a lot the inference task of our machine learning algorithm.

Hence, a good idea would be to identify this centroids in our dataset. Remind that we have created the new representation of our dataset based on some centroids that were randomly sampled. Clearly these points are not the real centroids so we will use the following method to identify them.

There are multiple classification algorithms in the machine learning literature that could be valid for our task. From support-vector machines to neural networks, we could use almost any model but we will focus only on those that can output probabilities. This is that they can estimate the probability that an instance has of being classified with certain label. For simplicity's sake, we will choose the classic Multinomial Logistic Regression (see equation 5.1) classifier which can be easily implemented and understood.

Multinomial Logistic Regression is the linear analysis to conduct when

the dependent variable is discrete and has more than two possible distinct values. It is simply an extension to the original logistic regression algorithm, which aims to analyze dichotomous dependents. In other words, we can say it is a model that is used to predict the probabilities of the different possible outcomes of a categorically distributed dependent variable, given a set of independent variable or features. It is also know as softmax regression and it became very popular with the rise of neural networks.

It is very important to highlight one of its main assumptions which differentiate this method from other as, for example, a naive Bayes classifier. The multinomial logistic model assumes that the dependent variable or label, cannot be perfectly predicted from the independent variables for any case. Its formula can be expressed as:

$$P(y = j|x) = \frac{e^{\mathbf{x}^T \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^T \mathbf{w}_k}} \tag{5.1}$$

where the vector x would represent the input data, the vector w would correspond with the parameters of the regression model which has usually the length of the input vector and one more element to add the intercept term.

5.3.1 Detecting the centroids

Once we have clear the method or algorithm we are going to use for our semi-supervised learning part, we need to determine which are the points from the whole dataset that can be considered as centroids of each of the clusters.

There multiple techniques for achieving this task and mostly of them are based on intra-cluster distance, but we are going to use a simpler method which will have into account the "opinion" of our multinomial logistic regression classifier. What we are going to do is simple. We start by training the algorithm will the whole dataset, all possible flights that we have available, this is a total of 1068. The input will consist in the data representation that we have explained in this chapter, in section 5.2. Once the classifier is trained, we will use it to predict new labels for the same data points but using the recently learned knowledge. The classifier outputs for each data point will be the probabilities of belonging to each of the clusters so we will save the point with the highest probability for each of the clusters.

These new set, of a total of ten points, will be queried to the oracle, so we can have labels for them. From this moment, we will only have ten labeled flights and we will start the active learning procedure, asking for more and more until we decide to stop. The stopping criteria can be based, for example, in some accuracy metric.

The fact of trusting in the classifier output to decide the centroids of each cluster can be difficult to interpret and for this reason we have created

figure 5.2. In there we can observe the following: the upper flight represents the obtained centroid, this is, the point or flight should be a "summary" or an average of all the others points in the cluster. The other three flights are random samples from that cluster.

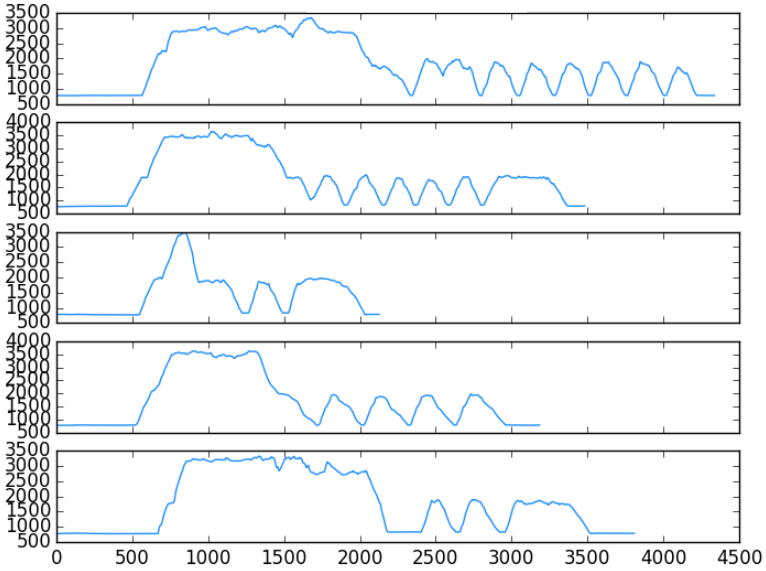


Figure 5.2: Obtained centroid of cluster 9 corresponds with the upper flights, the other flights are random samples from the cluster

If we focus in the image, we can really appreciate what we have previously mentioned. The upper flight mixes the characteristics of the four below as the centroid should do. It is surprising how well this performs, since this centroid is a real flight on our dataset not a artificially created one. Our algorithm was able to classify the flights in clusters and to obtain a representative flight from each of them.

Let's now move on and clarify how have we designed the iterative labeling phase that corresponds with the real active learning, how did we implemented it and how it performs.

5.4 Iterative labeling

This is the last step of this work and before going into deeper details we need to clarify one thing. This work is part of PEGASAS project from Federal Aviation Administration which aim is to enhance general aviation safety, accessibility and sustainability. This project had to presented on PEGASAS 2017 annual meeting and for this reason we had to develop an user interface that could serve us as a demo which show in public. In this section we will

present the iterative labeling phase together with the graphical user interface (GUI) that was developed for this meeting.

Our previous work consisted in ask to the oracle the labels of the cluster centroids and this gave us a set of labels to start with the iterative labeling phase. From here on we will imagine that there are only three possible true labels, we have made this decision following experts advice and also with the aim of simplifying the task. This three labels are *pattern work*, *local maneuvers* and *cross-county*, although it is possible to add more as through the following menu:

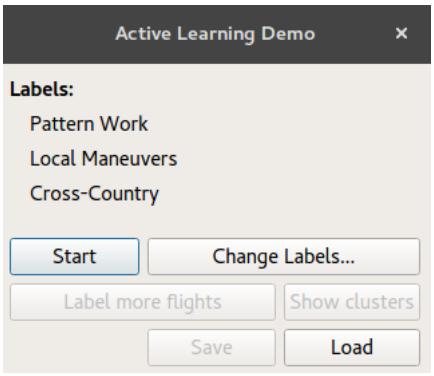


Figure 5.3: Simple menu with screenshot

After this we can start the active learning part. After pressing start the classifier will find the flight with the higher uncertainty in the whole dataset, this is, that after training itself only with the labeled data (in this first step only the ten centroids) and a strong regularization, it is going to predict labels for all the rest. Those flight that have obtained the lowest probability of belonging to certain label, have also a lot of uncertainty. Our algorithm does not actually know what to do with them so the will be queried in the next round of our active learning procedure.

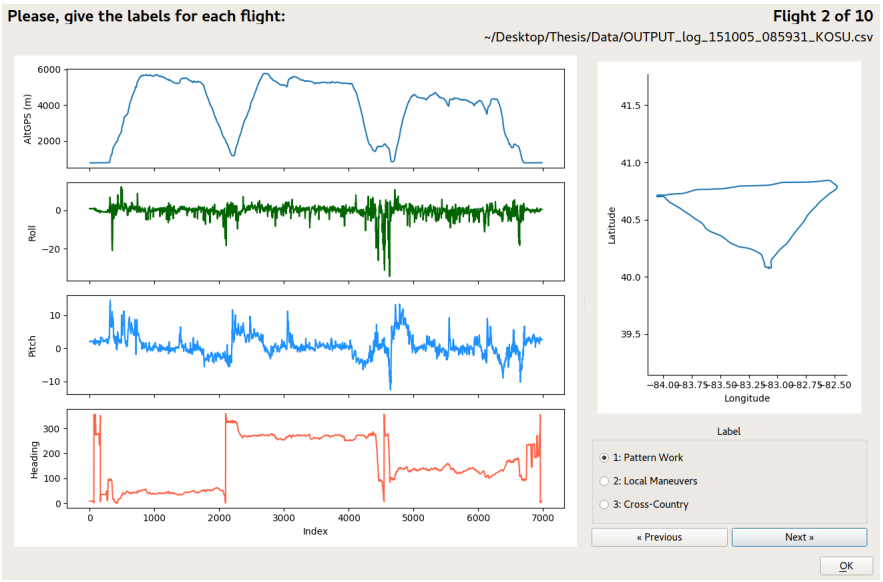


Figure 5.4: Decision or labeling screen

We need to proportion data to the user in order he can decide how to classify the flight in an accurate way. For this reason we have develop another screen that can be observed in figure 5.4. In this screen we can observe a total of five different plots. In the left window from up to down we can see altitude data, roll, pitch and heading data obtained from plane sensors. In the right panel we se a plot of longitude versus latitude, so we can form or make an idea of the route of the plain.

The flight that we see in that figure is very easy to categorize using only that data. If we focus in the altitude (the upper graphic) we can see that the plane took-off a total of three times. Furthermore if we observe the graphic in the right we can see that it move in a triangular way. This kind of flights are considered as cross-country.

After one iteration of the active learning we can also see the results and observe how the algorithm is classifying the different flights based on the input we are giving to him. This is very interesting and also very useful to ensure that the algorithm decisions are totally dependent on the labels we assign.

In figure 5.5 we show the screen that was developed with this aim. In there we can see the altitude data of each of the flights that the algorithm classifies with a certain label.

The use of active learning creates a total dependency between the algorithm and the oracle. So along the whole iterative process we must keep certain consistency, we must be clear about the type of labels we want to use and we also must have an idea of how to detect a flight of this type. After all, what the algorithm is going to do is to learn something that we are teaching

him, so we cannot change our criteria in the middle of the execution.

It is also very interesting how the algorithm evolves and asks more problematic flights each time. After a few rounds of labeling we start facing very difficult flight which are difficult to tag even for experts. There is also usually a certain relation between the queried flights. So once we give information to the algorithm he can extrapolate it and label with a lot of precision many flights.

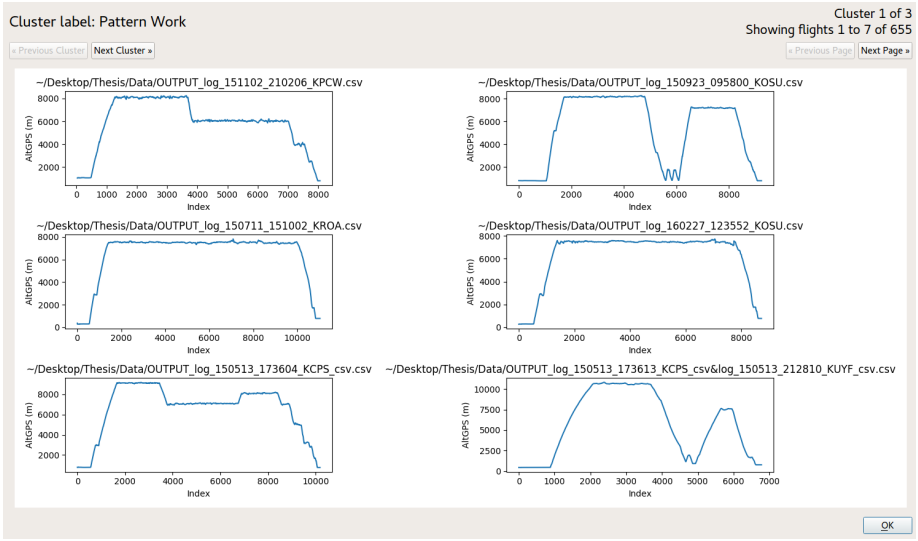


Figure 5.5: Show clusters screen

Conclusion

In this work we have started working with an unlabeled dataset formed by on board sensor data. These type of data are known as time series and for dealing with them, in a proper way, we need special techniques.

In this project we have shown how to use Dynamic Time Warping to reduce the whole dataset into a small matrix, composed of instances and features, that we used to feed a hierarchical clustering algorithm. The obtained labels from this step, were used to discover the most important flights in our dataset, which we called centroids. For doing this we use a Multinomial Logistic Regressor which was trained over the whole dataset. For obtaining the centroids we simply selected those flights that were associated with the highest probability of belonging to each of the classes. Finally, we developed an active learning procedure which queries the user about labels for the new flights and uses this information to improve the classification task.

This work has been presented at the Federal Aviation Administration (FAA) 2017 meeting in Texas. It is worth it to note that after three rounds of ten labels in the active learning or iterative part, aviation experts were having as much trouble as the algorithm in the classification task.

References

[1] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[2] Tatsuya Kiyohara, Ryohei Orihara, Yuichi Sei, Yasuyuki Tahara, and Akihiko Ohsuga. Activity recognition for dogs based on time-series data analysis. In *International Conference on Agents and Artificial Intelligence*, pages 163–184. Springer, 2015.

[3] ARUN Debray and RAYMOND Wu. Astronomical implications of machine learning, 2013.

[4] Eamonn Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 406–417. VLDB Endowment, 2002.

[5] Sang-Wook Kim, Sanghyun Park, and Wesley W Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 607–614. IEEE, 2001.

[6] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[7] Qiang Zhu, Gustavo Batista, Thanawin Rakthanmanon, and Eamonn Keogh. A novel approximation to dynamic time warping allows anytime clustering of massive time series datasets. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 999–1010. SIAM, 2012.

[8] Elena Tsiporkova. Dynamic time warping algorithm for gene expression time series. *Power point presentation. Retrieved September*, 11, 2010.

[9] Vit Niennattrakul and Chotirat Ann Ratanamahatana. Learning dtw global constraint for time series classification. *arXiv preprint arXiv:0903.0041*, 2009.

[10] Diego F Silva, Gustavo EAPA Batista, and Eamonn Keogh. Prefix and suffix invariant dynamic time warping. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 1209–1214. IEEE, 2016.

[11] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[12] Fionn Murtagh and Pierre Legendre. WardâĂŹs hierarchical agglomerative clustering method: which algorithms implement wardâĂŹs criterion? *Journal of Classification*, 31(3):274–295, 2014.

[13] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32 (3):241–254, 1967.

[14] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[15] Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118. ACM, 2001.

[16] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.