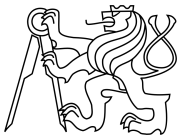
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY



ASSIGNMENT OF MASTER'S THESIS

Title:	Unsupervised Learning and Outlier Detection in Large Archives of Astronomical Spectra
Student:	Bc. Ksenia Shakurova
Supervisor:	RNDr. Petr Škoda, CSc.
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Theoretical Computer Science
Validity:	Until the end of winter semester 2017/18

Instructions

The current archive of the LAMOST telescope contains more than four million of automatically processed astronomical spectra. The goal of the thesis is to find reliable methods capable to identify scientifically interesting outliers, eliminate bad data, and group spectra of similar type using machine learning.
1) Make a survey of unsupervised techniques suitable for millions of feature vectors. Consider massively parallel processing (e.g., GPUs, Spark).
2) Select the best performing methods using the archive of Ondřejov 2m telescope containing about thirty thousand spectra of well identified stars.
3) Apply methods from 2) to the whole LAMOST archive in order to eliminate spurious and noisy spectra and to group similar spectral types.
4) Prepare a list of outliers for visual inspection and investigate their nature.
5) Run the same experiments with suitable dimensionality reduction.
6) Discuss the results with respect to precision and computing performance.

References

Will be provided by the supervisor.

L.S.

doc. Ing. Jan Janoušek, Ph.D.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague March 21, 2016

Czech Technical University in Prague
Faculty of Information Technology
Department of Theoretical Computer Science

Master's thesis

Unsupervised Learning and Outlier Detection in Large Archives of Astronomical Spectra

Bc. Ksenia Shakurova

Supervisor: RNDr. Petr Škoda, CSc.

10th May 2016

Acknowledgements

First of all I would like to thank my supervisor RNDr. Petr Škoda, CSc. for his valuable advices and assistance. Further, I am grateful to my colleague Bc. Andrej Palička for provided tools of data preprocessing.

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

Also I would like to thank my family for all help during the study. Especially, I am grateful to Bc. Nikita Orekhov for his support and advices.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on 10th May 2016 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Shakurova, Ksenia. *Unsupervised Learning and Outlier Detection in Large Archives of Astronomical Spectra.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2016.

Abstrakt

Tato práce se zabývá studiem vlastností několika populárních algoritmů shlukovací analýzy, např. DBSCAN, K-means, Biclustering a dalších pro astronomické účely. V práci se také zkoumají metody redukce dimenzionality a algoritmus LOF pro detekci odlehlých hodnot. Porovnání výkonnosti metod je zajištěno prostřednictvím experimentů na sadě snímků spekter z observatoře Ondřejova. Metody, které vykázaly nejlepší výsledky, jsou pak otestovány na větším archivu spekter získaných s teleskopu LAMOST. Výsledky experimentů jsou důkladně analyzovány.

Klíčová slova Učení bez učitele, Shluková analýza, Astroinformatika, Data mining, Astronomická spektroskopie, Big Data, Apache Spark

Abstract

In this thesis we examine popular clustering algorithms such as DBSCAN, Biclustering, K-means, etc., on the task of spectra clustering. In addition, we investigate several dimensionality reduction approaches and the algorithm LOF for the outliers detection. We conduct our experiment in order to resolve the problem of spectra clustering. We select the most promising methods according to their performance on the Ondřejov dataset and then apply them

on the larger LAMOST dataset. Next, we implement framework which incorporates mentioned algorithms including our implementation of LOF adapted for Apache Spark. Finally, we discuss obtained results.

Keywords Unsupervised learning, Cluster analysis, Astroinformatics, Data mining, Astronomical spectroscopy, Big Data, Apache Spark

Contents

List of Figures

List of Tables

Introduction

Motivation

Nowadays a tempo of collecting data is extremely high. This is especially evident in astronomy, where daily different observatories (state and private) collect petabytes of information, for example spectra and images of stars, quasars, galaxies, etc. But the biggest part of data is multidimensional radio data cubes (4D data cubes). The problem is that the data acquisition speed is not commensurate with the speed of processing this data. Some public databases are studied inside out by many researchers, but the most of databases are not available to the public and there is no much capabilities to explore them.

One of the issues of working with spectra is efficient classification of them. There are some pipelines that can roughly distribute objects between some big groups - stars with clearly distinguishable levels of brightness, galaxies, etc. These pipelines are not exact and usually cannot identify correctly unusual group representatives. These pipelines use previously computed templates of known types of stars and use some statistics to match spectra with the most similar template. At the same time stars and quasars have different pipelines and so on. That's why astronomers still have to classify them manually. Also while most telescopes observe certain objects, some unknown objects and artefacts can get recorded. Of course some noise will be also present.

Sometimes astronomers spot objects, that are not similar to any known groups - some new types of celestial objects.

Goal of the work

The goal of this work is to explore unsupervised algorithms and techniques that can be used for accurate grouping objects of similar types, for identifying scientifically interesting group's outliers that can be recognized as a new type of astronomical objects and for eliminating bad data.

This work will describe the process of analyzing and choosing clustering methods and methods for data dimensionality reduction. Then we will discuss results of experiments with partially labeled Ondřejov dataset and attempts to find out the most useful parameters by reducing feature space and apply chosen methods on the world's largest spectroscopic survey of Chinese telescope LAMOST [2]. The entire information about datasets is provided in section about data analysis 1.1 in the first chapter.

Chapter 1

Analysis

Unsupervised learning is a big part of machine learning and it's main goal is to solve task of inferring a function that would be able to describe hidden structure from unlabeled data. Since we have no information about labels, there is no error or reward signal to evaluate a potential solution. This distinguishes unsupervised learning from supervised learning.

This chapter contains description of the assigned task, which includes data analysis, comparison of clustering methods, data dimensionality reduction methods and techniques for searching outliers.

1.1 Data analysis

In this section we will give a definition of spectrum and the physical meaning of its curve, specificity of presenting data in astronomy and then description of datasets that are used in this work.

1.1.1 Spectrum definition and peculiarities

Spectrum obtained from detectors is an array of values of electromagnetic radiation intensity for every distinct wavelength (frequency point). Spectrum can have some peculiarities, which are essential to understand physical features of the celestial objects. Usually the majority of information contains in spectra lines – absorption lines, rarely emission lines or some kind of their mixture. Some simple examples of how stellar spectra can look like are presented in figure below 1.1.

There are some issues that are commonly encountered while attempting to use machine learning for spectra:

- The clusters would not be balanced because some types of objects are much more common than others. Furthermore, some clusters can contain just few stars while the others - thousands or even more. For example,

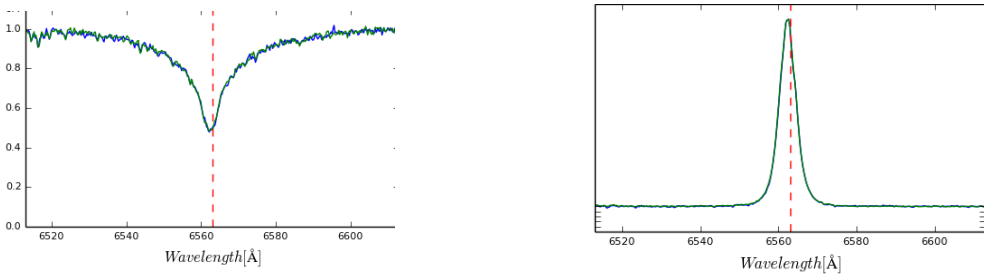


Figure 1.1: Spectrum with pure absorption line (left) and spectrum with one emission line (right). Source: `http://physics.muni.cz/~ssa/archive/`.

frequent stars are Be stars, because Ondřejov telescope was focused to observe Be stars for last 50 years (the electronic CCD detectors are used for last 15 year).

- Feature vector is very high-dimensional. It contains hundreds or thousands of points of spectrum's curve, so we would experiment with some limited areas of spectra, for example with some of the Balmer series as H_α [1]. The reason of choosing this part of spectrum is firstly that most of the observations from Ondřejov dataset contain exactly this area and also that a lot of interesting objects have emission in H_α. For example, the classical Be stars are defined as a non-supergiant B stars whose spectrum has (or had at some time) one or more Balmer lines in emission [3].

- Continuum normalization is needed to perform on the data. This normalization smooths bending of the whole spectra line to be horizontal (transform values to be about value 1), which means value of emission peak would be more than one and absorption would be less than one. Thus, profile of spectra would be identically scaled. Some emissions still can be very prominent – significantly higher than one. At the same time stars of the same type can have same geometry of emission but different values of emission peak.

As it was said before, there are some pipelines for classification of celestial objects, but they use some set of previously computed templates of known types of stars. Although they don't provide all needed information, for example identification of unusual objects, etc.

[1] The Balmer series or Balmer lines in atomic physics describe the spectral line emissions of the hydrogen atom

1.1.2 Data format

Flexible Image Transport System (FITS) is an open standard digital file format. It is highly used in astronomy as it can store almost every type of data [4]. FITS is useful for storage, transmission and processing of scientific and other images. Unlike many image formats, FITS is designed specifically for scientific data and hence includes many provisions for describing photometric and spatial calibration information, together with image origin metadata [5].

FITS file consists of two parts. The first one is a human-readable ASCII header with image metadata. However, a major feature of the FITS format is that any interested user can examine the headers. Each FITS file's header contains ASCII card images [2] that carry keyword/value pairs interleaved between data blocks. The second part of the FITS file is one or several extensions and each of these contains a data object in binary form. So it is possible to store several variants of image in the same file.

In general FITS files contain spectra (one spectra in its own FITS file) as an array of intensities for particular wavelengths. Values of wavelength for these intensities can be computed using reference point and difference given in metadata of a file. Also every spectra differs from the other in given grid of wavelengths. At the same time the unsupervised learning analysis uses intensities of particular wavelengths as features, which means we require same values of wavelengths (wavelength grid) for all spectra we analyze. Which in its turn means that data must be right and left cropped to smallest range and also scaled to the same intervals of intensities, i.e. reduced to the same grid. All these conversions are performed in preprocessing phase.

1.1.3 Description of datasets

This part will give common information on datasets used in this work and assignment of each of them. Different telescopes have different technical features, that's why datasets contain data that differ in observational features (e.g., depth, spatial coverage, resolution and spectral resolution).

Ondřejov dataset

This dataset is the archive provided by the Stellar department of the Astronomical Institute of the Czech Academy of Sciences. The spectra were obtained with a spectrograph of Ondřejov Observatory 2m Perek telescope.

It contains about seventeen thousand of spectra [6]. Part of them is manually divided to several groups so that's why this dataset is used for primary testing and verifying algorithms' efficiency. There are groups of normal spectra, spectra with some problems (damaged, etc) and non-sorted. The damaged spectra are frames with some instrumental artefacts like spectrograph error or

[2] Card image is an archaic term for a character string, usually 80 characters in length, that was, or could be, contained on a single punched card.

blur frame, etc. These spectra were immediately excluded from the testing set because of its negative influence on learning abilities of the algorithms. There is no scientific interest in training algorithms on set with priory noisy samples because it would be hard to obtain a good result. Also some spectra were excluded during the preprocessing, which scales spectra to the same grid of intensities.

Dataset of LAMOST telescope

Large Sky Area Multi-Object Fiber Spectroscopic Telescope (LAMOST) is a project of the National Astronomical Observatories of the Chinese Academy of Sciences. The scientific goal of LAMOST focuses on the extra-galactic observation, structure and evolution of the Galaxy and multi-wave identification [2].

The data set of LAMOST data release one (DR1) used in this work includes spectra of the pilot survey and spectra of the first year of the regular spectroscopic survey. This dataset has already been published for the public. The DR1 totally contains 2,204,860 spectra, including 717,660 spectra of pilot survey and 1,487,200 spectra of regular survey. In addition, they calculate the atmospheric parameters of 1,085,404 stars, which becomes the largest stellar spectral parameters catalog in the world at present. The spectra from that dataset were processed by pipelines. This process spread spectra between some common groups (stars, quasars, galaxies, etc.) [7]. This pipeline is far from absolute accuracy but knowledge obtained from pipeline will be used to isolate stars from other objects (for more information see description of experiments on LAMOST dataset 3.3).

Data release 3 (DR3) - the last full release of the LAMOST spectral survey, containing 5,755,126 spectra and 2,667 plates in total. The current archive (DR4) is an additional release to DR3 contains further more than one million of spectra [8]. All of them are automatically processed by pipelines.

1.2 Clustering techniques

Common technique in unsupervised learning is a *cluster analysis*. It groups objects of similar kinds into categories and sorts different objects into groups by their degree of association. Cluster analysis itself is not one specific algorithm but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a clusters and how to efficiently find them. Cluster analysis can be seen as iterative process of knowledge discovery. This section will describe clustering algorithms and taxonomy of them, methods of dimensionality reduction, manifold learning techniques and method for finding the outliers in data.

1.2.1 Taxonomy of clustering techniques

Categorization of the clustering algorithms is neither simple nor canonical [9]. We will mention only the most important and commonly used categorizations.

Flat vs. hierarchical clustering

Flat clustering creates a set of internally coherent clusters, which are clearly different [1]. This type of clustering does not provide explicit information about how should clusters relate to each other. The key point of this clustering is a requirement of input parameter - the number k of clusters. Choise of this parameter is usually a guess based on domain information or previous experiments. *Hierarchical clustering* creates a so-called *dendrogram* or tree of clusters (see figure 1.2). In other words it creates the hierarchy of relationships between clusters. This method does not need cluster number as input because it can have different stopping criterion for example the number of iterations.

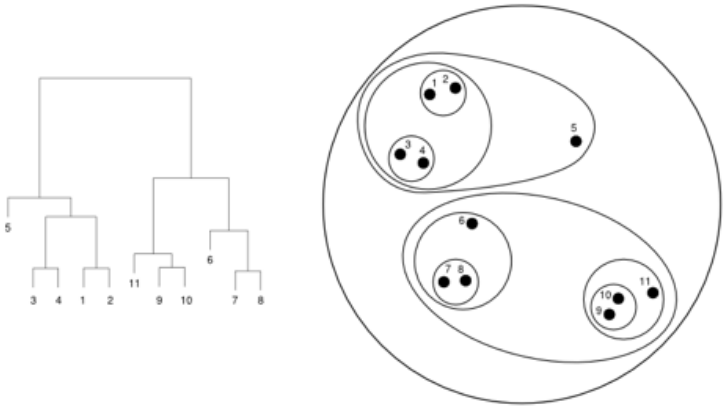


Figure 1.2: Typical example of hierarcical clustering: dendrogram (left) and view of clusters (rigth).

Partitional and non-partitional clustering

Algorithms that belong to *partitional* clustering perform a so-called *hard assignment* that means each object will be a member of only one cluster. The second type – *non-partitional* clustering - performs *soft clustering* (or *fuzzy clustering*). This approach assigns point to more than one cluster using a membership function [10]. Visualization of the difference between partitional and non-partitional clustering is given in figure 1.3.

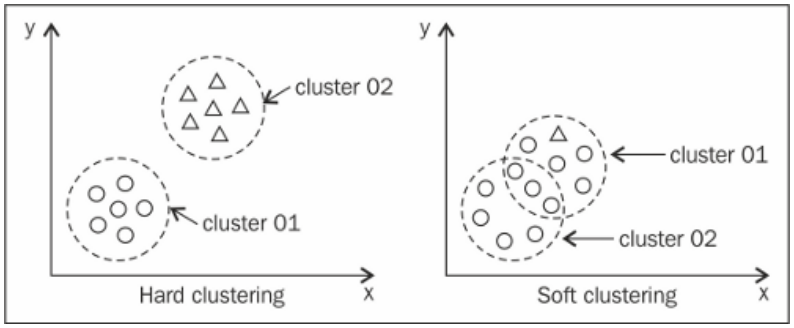


Figure 1.3: Example of partitional (left image) and non-partitional (right image) clustering.

One-way clustering vs. multi-dimensional clustering

In most cases input data is a data matrix of the size $n \times m$. One-way clustering algorithms perform clustering along only one dimension of this matrix. Multi-dimensional clustering performs simultaneous clustering along multiple dimensions [11]. This simultaneous clustering usually preformed along two dimensions. This approach has many advantages. For example, it demonstrates the ability to find relationships between clusters along all dimensions. Also it can act implicit dimensionality reduction and perform well with sparse data matrix [9]. Two-dimensional clustering is often called biclustering or co-clustering (for details see 1.2.2.4).

Basis of the cluster model definition

This view to taxonomy distinguishes several common types of clustering algorithms [12]:

- connectivity-based clustering is also known as hierarchical clustering, builds models based on distance connectivity;
- centroid-based clustering (K-Means, etc.) represents each cluster by a single mean vector;
- distribution-based clustering (Gaussian mixture models, etc.) establishes clusters using statistical distributions;
- density-based clustering (DBSCAN, etc.) defines clusters as connected dense regions;
- other algorithms, for example: BSAS (basic sequential algorithmic scheme), Biclustering.

1.2.2 Selected clustering algorithms

This section describes in details techniques that are considered in this work, comparison of these methods and possibility of using them.

Also taking into consideration peculiarity of the problem, we can identify the following sticking points:

- Flat types of clustering are complicated because we don't have any idea of how many clusters can be in dataset (especially in LAMOST dataset). These types of algorithms would require tuning of appropriate number of clusters.
- Stars of the same type can have very different value of emission peak (see subsection 1.1.1), that's why similar stars can be defined as different, for example by connectivity-based algorithms, which use simple Euclidean distance as measure.
- Density-based algorithms' tuning is complicated because of non-balanced data (for details see description of DBSCAN 1.2.2.2).

1.2.2.1 K-means

K-means is a commonly known flat clustering algorithm. It aims at separation of n observations to a certain number of clusters (assume k clusters) fixed a priori. The classic K-means clustering algorithm finds cluster centroids that minimize the distance between data points and the nearest centroid [13], usually within-cluster sum of squares (or inertia) used for this purposes. It is defined in such way

$$J = \sum_{j=0}^{k} \sum_{i=1}^{n} ||x_i^{(j)} - c_j||^2 \ , \tag{1.1}$$

where $||x_i^{(j)} - c_j||$ is a chosen measure between a data point $x_i^{(j)}$ and cluster center c_j [14]. A drawback is that this definition of distance makes the assumption that clusters are convex and isotropic, which is not always the case. It hardly detects elongated clusters or manifolds with irregular shapes. Usually *Euclidean distances* is used as a distance measure. But it tends to be inflated in very high-dimensional spaces, which is true in our case. That's why K-means is usually used with some dimensionality reduction methods like PCA, LLE, etc.

K-means starts with selecting as initial clusters' centers named the *seeds* from some randomly chosen objects. There is another method for seeds initializing called k-means++ (see section 2.1.1).

Mini-batch K-Means

Mini-batch K-Means is a variant of K-means that yields excellent clustering results with low computation cost on large data [15]. This method uses so-called *mini-batches* - set of randomly chosen points in space. Mini-batches of defined size are re-selected in each training iteration. This decreases amount of required computations. In each iteration for each sample the assigned centroid is updated by taking the streaming average of the sample and all previous samples assigned to that centroid. This has the effect of decreasing the rate of change for a centroid over time. Surely results will differ from K-means but only slightly. Also mini-batch variant converges faster than K-means [16].

1.2.2.2 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm, which creates clusters based on high and low density areas. Obtained clusters can have almost any shape (not only convex). The DBSCAN relies on a *density-based* notion, which is designed to discover clusters of an arbitrary shape [17]. A cluster is seen as a set of *core samples* (samples in areas of high density) and a set of *non-core samples*. Each core sample is close to another core sample and non-core samples are also close to a core sample.

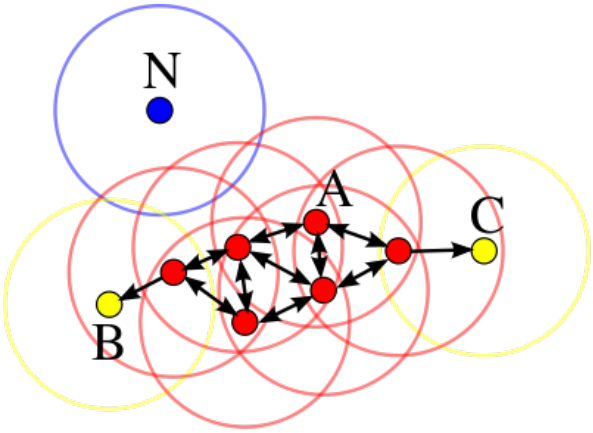


Figure 1.4: Minimum required number of point m is set to three. Point A and the other red points are *core points*, because at least three points surround them are in an ε radius. Points B and C are not core points but are reachable from A (via other core points) and thus belong to the cluster as well. Point N is a noise point because it is more distant than ε.

Process of clusters forming is controlled by two parameters: the distance ε and the minimum number of samples required to form cluster m. Figure 1.4

shows an example of the cluster, got from DBSCAN algorithm with minimum required number of point m is set to three.

Algorithm starts with an arbitrary starting point that has not been visited and then recursively takes a core sample, finding all of its core sample neighbors N. Then algorithm finds all of the neighbors of N that are also core samples and so on. Set of non-core samples, intuitively, are on the fringes of a cluster because they are close to core samples. Further, any cluster has at least m points in it, following the definition of a core sample. For any sample that is not a core sample, and does have a distance higher than ε to any core sample, it is considered as *outlier* by the algorithm.

Time complexity of DBSCAN depends on a number of visits of each point of the dataset that possibly happens several times (e.g., as candidates to different clusters). Practically, the time complexity is mostly governed by the number of the nearest neighbor search queries. DBSCAN executes exactly one such query for each point. The best option is to use some indexing structure. Then the average run time complexity of a single region query is $O(\log n)$ [17]. Further, if distance ε is chosen in a meaningful way, complexity $O(n \log n)$ can be achieved. The worst case is when no acceleration index is used and the distance ε is big enough, so that many points lie within. In this case the time complexity will reach $O(n^2)$. The memory complexity is $O(n^2)$ if distance matrix is used. In some non-matrix based implementations the memory complexity can be reduced to $O(n)$.

The algorithm has such advantages:

- DBSCAN does not require specified number of clusters;
- algorithm can find arbitrarily shaped clusters, it can even find a cluster completely surrounded by (but not connected to) a different cluster;
- algorithm has a notion of noise, and is robust to outliers.

It has to be also said that DBSCAN is designed for use with databases that can accelerate region queries, e.g. using an R*-tree [17]. This accelerates overall computation time. On the other side, the algorithm depends on several things that can cause unsatisfactory results:

- DBSCAN requires only two parameters, but if data are not well understood, selection of suitable parameters can be very difficult.
- DBSCAN is not completely deterministic [3]. Usually it is not sensitive to the ordering of points but, occasionally, if the ordering of the points is changed, points occurring on the edge of two different clusters might swap cluster membership. Although sets of core points will be the same, non-core points can belong to different clusters.

[3]Some deterministic implementations exist [18], but in general, algorithm can not be called deterministic.

- The quality of DBSCAN depends on the distance measure used for finding neighborhoods. Especially for high-dimensional data, commonly used Euclidean distance can be rendered almost useless due to the so-called "curse of dimensionality". However, this effect is present in any algorithm based on Euclidean distance.

- DBSCAN cannot cluster datasets well with large differences in densities since the combination of the distance ε and the required samples' number m cannot then be chosen right for all clusters. This means that some big clusters can be simply separated to smaller clusters because of small required number of samples. On the other side, bigger minimum number can cause ignoring of independence of very small clusters, etc.

1.2.2.3 Affinity propagation

Affinity propagation identifies the representative examples (exemplars) within the dataset by exchanging real-valued messages between all data points [19]. The dataset is then described by a small number of exemplars, which are identified as the most representative. The messages sent between pairs represent the suitability for one sample to be the exemplar of the other, which is updated in response to the values from other pairs. This updates happen iteratively until convergence. Hence the final clustering is given (for example see figure 1.5).

Affinity Propagation is interesting as it chooses the number of clusters based on the provided data. There are two important parameters: the *preference*, which controls how many exemplars are used, and the *damping factor*.

The main drawback of Affinity Propagation is its complexity. The algorithm has a time complexity of the order $O(N^2T)$, where N is the number of samples and T is the number of iterations until convergence. Further, if a dense similarity matrix is used, the memory complexity is $O(N^2)$. But it is reducible if a sparse similarity matrix is applied. This makes affinity propagation more appropriate for small or medium datasets.

Initially, all nodes are considered as exemplars. If no prior knowledge about favored exemplars is available, then the same preference value can be assigned to all nodes. The magnitude can be used to control cluster granularity. For each node i and each candidate exemplar k affinity propagation computes the *responsibility* $r(i,k)$, which indicates how well k suits for i as an exemplar (see equation 1.2) and the availability $a(i,k)$ reflects the evidence that i should choose k as an exemplar [19] (see equation 1.3).

$$r(i,k) \leftarrow s(i,k) - \max[a(i,\tilde{k}) + s(i,\tilde{k}), \forall \tilde{k} \neq k] \tag{1.2}$$

$$a(i,k) \leftarrow \min[0, r(k,k) + \sum_{\tilde{i} \notin \{i,k\}} \max(0, r(\tilde{i},k))] \tag{1.3}$$

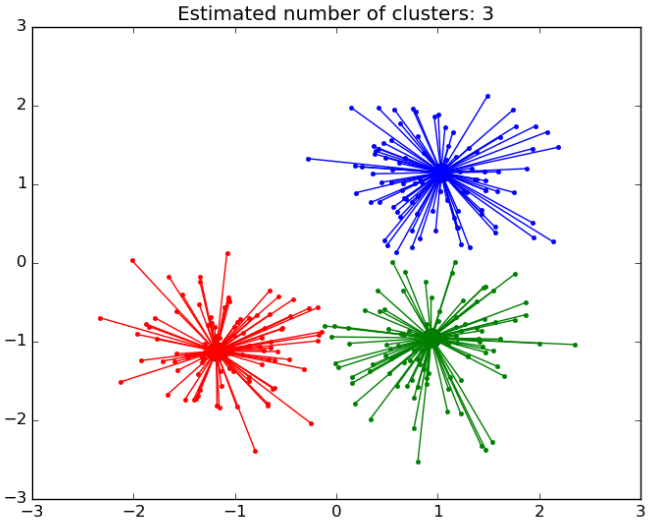


Figure 1.5: Example of affinity propagation resulting clusters.

1.2.2.4 Biclustering

The main goal of *Biclustering*, also known as *co-clustering* or *block clustering*, is to find biclusters (co-clusters) instead of simple clusters. A *co-cluster* or *bicluster* is a group of objects and features that are inter-related. The features in a co-cluster are the attributes used for choosing the objects to put in that co-cluster [4]. This means that algorithms simultaneously cluster rows and columns of a data matrix. These clusters of the rows and columns are known as biclusters. Each determines a submatrix of the original data matrix with some desired properties.

There are many different approaches for biclustering. The most interesting of them use K-means as a strategy [9]. The complexity of the biclustering problem depends on the exact problem formulation and particularly on the merit function used to evaluate the quality of a given bicluster.

Different biclustering algorithms have different definitions of bicluster:

- *constant values*, in other words a constant bicluster (is suitable for the tidy data but not for noisy data);
- *constant rows*, or *constant columns* algorithm will use variance of values and normalization;

[4]For example, in the analysis of DNA microarrays this means to find submatrices composed of subgroups of genes and subgroups of conditions, where the genes of a submatrix exhibit highly correlated activities for every condition in the same submatrix [9].

- unusually *high or low values*;
- sub-matrices with *low variance*;
- *correlated* rows or columns [5].

Biclustering algorithms also differ in how rows and columns may be assigned to biclusters, which leads to different bicluster structures. *Block diagonal* or *checkerboard* structures occur when rows and columns are divided into partitions. The first type occurs after rearranging the rows and columns of the data matrix, when each row and each column belongs to exactly one bicluster. The second type occurs in data, when each row belongs to all column clusters, and each column belongs to all row clusters.

1.2.2.5 Hierarchical clustering

Hierarchical clustering is a method of cluster analysis, which used to build a hierarchy of relationships between the clusters. The strategies of hierarchical clustering generally fall into two types:

1. Agglomerative ("bottom up" approach): each observation starts in its own cluster and pairs of clusters are merged as one moves up the hierarchy-.
2. Divisive ("top down" approach): starts with all objects in one cluster and subdividing them into smaller pieces. Divisive methods are not generally available and are rarely applied [21].

In general, the time complexity of hierarchical clustering is too high for large datasets: $O(n^3)$ for agglomerative clustering and $O(2^n)$ for divisive clustering with an exhaustive search. However, for some special cases optimal efficient agglomerative methods with time complexity $O(n^2)$ are known. This type of clustering because of its time complexity it less suitable for our issue.

1.2.2.6 Birch

Birch (Balanced Iterative Reducing and Clustering using Hierarchies) is an unsupervised data mining algorithm used to perform hierarchical clustering over particularly large data-sets. The advantage of BIRCH is its ability to incrementally and dynamically cluster incoming multi-dimensional metric data points to try to produce the best clustering quality with the available resources (i. e., available memory and time constraints) [22]. In most cases, BIRCH only requires a single scan of the database. For the given data Birch builds

[5] In Cheng and Churchs' theorem a bicluster is defined as a subset of rows and columns with almost same score. The similarity score is used to measure the coherence of the rows and columns [20]

a tree called the *Characteristic Feature Tree (CFT)*. The data are essentially lossy compressed to a set of *Characteristic Feature Nodes (CF Nodes)*. The CF Nodes have a number of subclusters called *Characteristic Feature subclusters*. These CF Subclusters located in the non-terminal CF Nodes can have CF Nodes as children. The CF Subclusters hold the necessary information for clustering which prevents the need to store the entire input data in memory. This information includes:

- *number of samples* in a subcluster,
- *linear sum* - a n-dimensional vector holding the sum of all samples,
- *squared sum* - sum of the squared Euclidean norm of all samples,
- *centroids* which help to avoid recalculation,
- *squared norm* of the centroids.

Characteristic Feature Tree is a height-balanced tree with two parameters: *branching factor* and *threshold* [22]. The threshold limits the distance between the entering sample and the existing subclusters. The branching factor limits the number of subclusters in a node.

Since Birch reduces input data to subclusters (stored in leaves of Characteristic Feature Tree), it can be seen as a version of data reduction method. Among the disadvantages of Birch there is a bad scalability to high dimensional data.

1.3 Dimensionality reduction methods

Dimensionality reduction or *dimension reduction* is the process of reducing the number of considered variables. It can be divided into *feature selection* and *feature extraction* [23].

Feature selection approaches try to find a subset of the original variables (also called features or attributes). There are three strategies: filter (e.g. information gain), wrapper (e.g. search guided by accuracy) and embedded (features are selected to add or be removed while building the model based on the prediction errors) approaches.

Feature selection techniques should be strictly distinguished from feature extraction. Feature extraction creates *new features* from functions of the original features, whereas feature selection returns a *subset of the original features.* Feature selection techniques are often used in domains where there are many features and comparatively few samples (or data points). So in our work we will consider only feature extraction methods.

As said before, feature extraction transforms the data from the high-dimensional space to a space of fewer dimensions. Methods for data transformation are *linear*, as in principal component analysis (PCA), or *non-linear.*

All methods differ in complexity and recommended usages (different algorithms work differently on the same data). Further follow descriptions of some dimensionality reduction methods (linear and non-linear):

- Principal component analysis (PCA),
- Locally Linear Embedding (LLE),
- t-distributed Stochastic Neighbor Embedding (t-SNE),
- Multidimensional scaling (MDS),
- Isomap (Isometric Mapping) Embedding,
- Spectral Embedding.

1.3.1 PCA

This technique is common linear technique for dimensionality reduction. It seeks way to decompose a dataset into a linear combination of a small number of *principal components*. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.

PCA is equivalent to finding a best-fit linear subspace to the entire dataset, such that the variance of the data projection into a subspace is maximized. Over last two decades, the PCA has been applied to a wide range of spectral classification problems [24].

1.3.2 LLE

LLE also begins by finding a set of the nearest neighbors of each point. It then computes a set of weights for each point that best describe the point as a linear combination of its neighbors. Finally, it uses an eigenvector-based optimization technique to find the low-dimensional embedding of points, such that each point is still described with the same linear combination of its neighbors [25].

In other words it's seeks to find low-dimensional projection of data set that best preserves geometry of local neighborhoods within a data. LLE can be thought of as a non-linear generalization of PCA [24].

LLE tends to handle non-uniform sample densities poorly because there is no fixed unit to prevent the weights from drifting as various regions differ in sample densities. LLE has no internal model.

1.3.3 t-SNE

t-distributed stochastic neighbor embedding (t-SNE) is a machine learning algorithm for dimensionality reduction developed by Laurens van der Maaten and Geoffrey Hinton (2008). It is a nonlinear dimensionality reduction technique that is particularly well suited for embedding high-dimensional data into a space of two or three dimensions. Specifically, it models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points [26].

The t-SNE algorithm comprises two main stages. First, t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked, whilst dissimilar points have an infinitesimal probability of being picked. Second, t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the Kullback–Leibler divergence between the two distributions with respect to the locations of the points in the map.

1.3.4 MDS

Multidimensional scaling (MDS) seeks a low-dimensional representation of the data in which the distances respect well the distances in the original high-dimensional space. In general, is a technique used for analyzing similarity or dissimilarity data. MDS attempts to model similarity or dissimilarity data as distances in a geometric spaces [25].

There are three types of MDS algorithm: *classical*, which is basically identical to PCA, *metric* and *non-metric*. In metric MDS, the input similarity matrix arises from a metric. Then the distances between two output points are set to be as close as possible to the similarity/dissimilarity data. Non-metric version of the algorithm is trying to hold the order of the distances, and hence seeks for a monotonic relationship between the distances in the embedded space and the similarities(dissimilarities).

1.3.5 Isomap

Isometric Mapping is one of the earliest approaches to manifold learning. Isomap can be viewed as an extension of Multi-dimensional Scaling (MDS) or Kernel PCA [16]. Isomap seeks a lower-dimensional embedding which maintains geodesic distances between all points.

1.3.6 Spectral Embedding

Spectral Embedding (also known as Laplacian Eigenmaps) builds a graph incorporating neighborhood information of the dataset. A low dimensional representation of the data are computed using notion of the Laplacian of the

graph [27]. The generated graph can be considered as a discrete approximation of the low dimensional manifold in the high dimensional space. Minimization of a cost function based on the graph guarantees that the closest points mapped close to each other in the low dimensional space and local distances are preserved.

1.4 Outliers detection

In statistics an outlier is an observation point that is distant from other observations. In data mining and machine learning outliers are often called *points of noise* - points that do not belong to any cluster.

Outlier detection aims to detect objects which behave in an unexpected way or have abnormal properties. It can find rare, unknown, or bad data. For example an outlier may indicate an experimental error. The used techniques are commonly divided into six methods, i.e., distribution, depth, distance, clustering, density and deviation based methods.

Usually clustering do not identify outliers as some special points but distribute them among all or by some clusters. Unsupervised anomaly detection techniques detect anomalies in an unlabeled test dataset under the assumption that the majority of the instances in the dataset are normal by looking for instances that seem to fit least to the remainder of the dataset.

The approach processes the data as a static distribution, pinpoints the most remote points, and flags them as potential outliers [28].

In our work we will consider only local outlier factor algorithms for outliers searching.

1.4.1 Local outlier factor

Local outlier factor (LOF) is an algorithm proposed by Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng and Jörg Sander in 2000. These algorithms are used to find anomalous data points by measuring the local deviation of a given data point with respect to its neighbors [29].

LOF is based on a concept of a local density so it shares some ideas with DBSCAN and OPTICS such as the *core distance* and the *reachability distance*. These distances are used for local density estimation and locality is given by *k nearest neighbours*. The main idea is to compare local density of an object to the local densities of its neighbours so algorithm can identify areas with similar density and points with significantly lower density than their neighbours. These points are named outliers.

The local density is estimated by the typical distance at which a point can be "reached" from its neighbors. The definition of reachability distance used in LOF is an additional measure to produce more stable results within clusters.

One of the key notions for LOF is the *k-distance* defined in such way: for any $k > 0$ the k-distance of object p is the distance $d(p, o)$ between p and an object $o \in D$ such that:

- for at least k objects $o' \in D$ p it holds that $d(p, o') \leq d(p, o)$;
- for at most $k - 1$ objects $o' \in D$ p it holds that $d(p, o') < d(p, o)$.

Simply put this means the distance of the object p to the k-th nearest neighbor, but set of the k nearest neighbor ($N_k(p)$) includes all objects at this distance, so it can contain more than k objects. Using k-distance the *reachability distance* can be defined as

$$\text{reach-distance}_k(p, o) = \max(\text{k-distance}(o), d(p, o)) \tag{1.4}$$

In another words the reachability distance of an object p from o is the true distance between the two objects but at least the k-distance of o. Figure 1.6 illustrates the idea of reachability distance with k = 4.

Then the *local reachability density* of object p is defined as

$$lrd_k(p) = 1/\left(\frac{\sum_{o \in N_{k(p)}} \text{reach-distance}_k(p, o)}{|N_k(p)|}\right) \tag{1.5}$$

that is the inverse of the average reachability distance of the object p from its neighbors. Note that it is not the average reachability of the neighbors from p (which by definition would be the k-distance(p)), but the distance at which it can be "reached" from its neighbors. With duplicate points this value can become infinite.

The (local) outlier factor of p is defined as

$$LOF_k(p) = \frac{\sum_{o \in N_{k(p)}} \frac{lrd_k(0)}{lrd_k(p)}}{|N_k(p)|} \tag{1.6}$$

The outlier factor of object p captures the degree according to which we call p an outlier. It is the average of the ratio of the local reachability density of p and those of p's MinPts-nearest neighbors. A value of approximately 1 indicates that the object is comparable to its neighbors (and thus not an outlier). A value below 1 indicates a denser region (inliers), while values significantly larger than 1 indicate outliers.

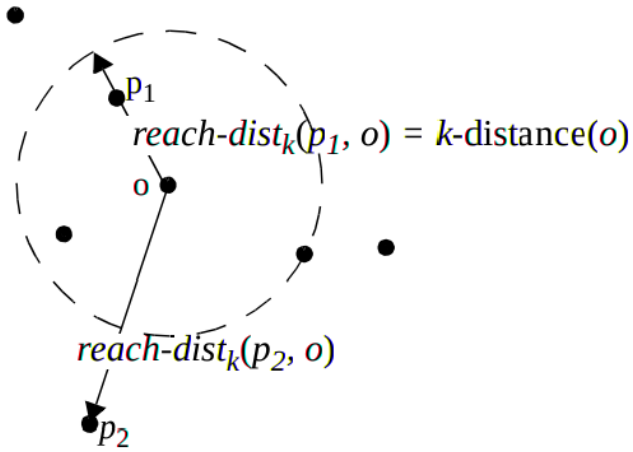


Figure 1.6: reach-dist(p_1,o) and reach-dist(p_2,o), for k=4

Local approach gives LOF an ability to identify outliers in a dataset that would not be outliers in another area of the dataset. For example points that are near to a very dense cluster might be an outlier or vice versa a point within a sparse cluster might not be an outlier.

The algorithm can be applied in many contexts. It has experimentally been shown to work very well in numerous setups, often outperforming the competitors, for example in network intrusion detection and on processed classification benchmark data [30].

The problem may arise while interpreting result of the algorithm. There is no clear rule for calling point an outlier. In some datasets and setups (with strong local fluctuations) a values of more than 1 could still be an inlier. These differences can also occur within a dataset due to the locality of the method. There are some improved extensions of LOF like Local Outlier Probability (LoOP) - a method using inexpensive local statistics to become less sensitive to the choice of the parameter k having resulting values scaled range of [0:1], etc.

1.5 Clustering validity methods

Clustering algorithms define clusters, that are not known a priori. Independently of the clustering method, that requires some kind of special appropriate evaluation [31]. The issues of clustering result validation (like number of clusters in dataset, fitting of resulting scheme and searching for a better partitioning) are the subjects of methods' discussion. They aim at the quantitative evaluation of the results of the clustering algorithms and are known under the general term *cluster validity* methods.

The main subject of cluster validity is finding out partitioning (optimal clustering scheme) that best fits the underlying data.

In general all clustering algorithms search for clusters which members are most close to each other (have the highest degree of similarity) and at the same time well separated. In most cases algorithms' evaluations conducted on 2D or 3D datasets that fit better for human visual verification. For datasets with high dimensionality (more then three) it is difficult to make any kind of visualization, that's why dimensionality reduction techniques can be used.

Behavior of clustering algorithms depends on:

- the features of data
- the input parameters values

The first point is about geometry of data and density distribution of clusters. As for the second point, it can be for example the issue of choosing cluster number for K-means clustering. In figure 1.7 is shown difference between results of same K-means applied with different input parameter number of clusters. This should visualize that unsuitable input parameters can lead any (even suitable) algorithm to inappropriate outcome.

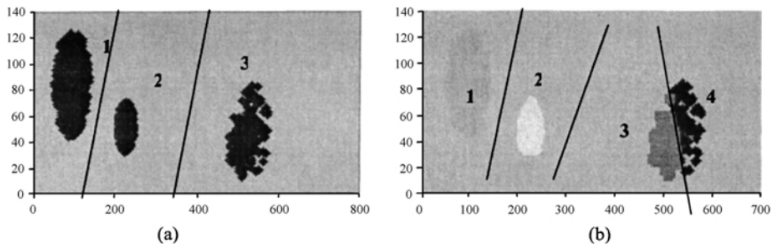


Figure 1.7: Resulting clusters after application K-means with input parameter number of clusters set to a) three and b) four clusters.

In general terms there are three approaches to probe clustering quality. The first approach is based on *external criteria*, which means we use previously known structure that is imposed on data and reflects our assumptions about clustering structure of the dataset. The second approach is based on *internal criteria*. This approach means that we evaluate results of the clustering by quantities that contains in data vectors themselves. In other words we use algorithms' optimization criterion such as proximity matrix, etc. The last approach is based on so-called *relative criteria*. The basic idea of this approach is to compare results with another clustering schemes obtained from same clustering algorithm but using another input parameters.

1.5.1 External criteria

In most cases user-based evaluation can cost much, so some benchmarks with a good level of inter-judge agreement can be used as surrogate for human judgment. Some criterion can evaluate how much clustering scheme fit a gold standard. This subsection will describe some types of external criteria.

Purity

This is a simple and transparent type of criteria. To compute it each cluster is assigned to the class which is the most frequent in it and then just says the number of correctly assigned object divided by n. Formally it can be written as

$$purity(\Omega, \Gamma) = \frac{1}{n} \sum_{k} \max_{j} |w_j \cap c_j| \tag{1.7}$$

where $\Omega = w_1, w_2, \ldots, w_k$ is the set of clusters and $\Gamma = c_1, c_2, \ldots, c_J$ is the set of classes.

Purity has values from 0 to 1 where bad clustering has values close to 0 and perfect clustering is closer to 1. Big value of purity does not necessary mean good clustering. Sometimes big number of clusters gives a priori better results for example if algorithm will assign one object to its own cluster. Thus, purity criterion can't be used to trade off the quality of the clustering against number of clusters.

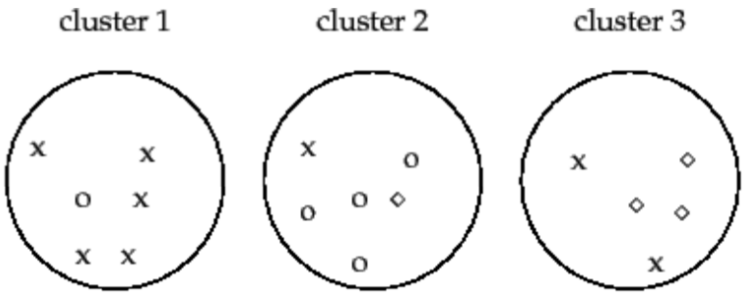


Figure 1.8: Purity is an external criterion for evaluation of clustering scheme quality. The major class and the number of members of this class in each cluster is: X, k (cluster 1); ∘, 4 (cluster 2); ⋄, 3 (cluster 3). Then purity is $(1/17) * (5 + 4 + 3) \approx 0.71$ [1]

Normalized Mutual Information (NMI)

NMI has advantages over purity because it allows us to make a trade-off between quality and number of clusters. Contrary to inertia 1.1, MI-based measures require the knowledge of the ground truth classes while almost never

available in practice or requires manual assignment by human annotators (as in the supervised learning setting). It can be defined as

$$NMI(\Omega, \Gamma) = \frac{I(\Omega, \Gamma)}{[H(\Omega) + H(\Gamma)]/2} \tag{1.8}$$

where I is a *mutual information* that measures mutual dependence between the two variables

$$I(\Omega; \Gamma) = \sum_k \sum_j P(\omega_k \cap c_j) \log \frac{P(\omega_k \cap c_j)}{P(\omega_k)P(c_j)} \tag{1.9}$$

and H is the *entropy*

$$H(\Omega) = -\sum_k P(\omega_k) \log P\omega_k \tag{1.10}$$

and where $P(\omega_k)$ and $P(c_j)$ are the probabilities of an object being in a ω_k, c_j, and $P(\omega_k \cap c_j)$ is a probability of an object being in their intersection.

The particular form of the denominator is chosen because $[H(\Omega)+H(\Gamma)]/2$ is the tight upper bound on mutual information $I(\Omega; \Gamma)$. Thus, NMI is always a number between 0 and 1.

Mutual information and also the normalized mutual information are not adjusted w.r.t probability and will tend to increase as the number of different labels (clusters) increases, regardless of the actual amount of "mutual information" between the label assignments. The equation from Vinh, Epps, and Bailey (2009) can be used for obtaining the expected value for the mutual information. In this equation, $a_i = |U_i|$ (the number of elements in U_i) and $b_j = |V_j|$ (the number of elements in V_j). This expected value between two clusterings U and V is describes as

$$E[I(U,V)] = \sum_{i=1}^{R} \sum_{j=1}^{C} \sum_{n_{ij}=\max(a_i+b_j-N,0)}^{\min(a_i,b_j)} \frac{n_{ij}}{N} log\left(\frac{N.n_{ij}}{a_i b_j}\right) \times \frac{a_i! b_j! (N-a_i)! (N-b_j)!}{N! n_{ij}! (a_i - n_{ij})! (b_j - n_{ij})! (N - a_i - b_j + n_{ij})!}$$

As suggested by Hubert and Arabie (1985), the general form of a similarity index corrected for chance is given by *Adjusted Mutual Information*:

$$AMI = \frac{I(U,V) - EI(,V)}{\max(H(U), H(V)) - E[I(U,V)]}, \tag{1.11}$$

which is upper-bounded by 1 and equals 0 when the index equals its expected value.

Rand Index (RI)

This criteria shows an alternative for information theoretic interpretation. This approach is to view a clustering scheme as a series of decisions for each of the $N(N-1)/2$ pairs of objects in the collection. We expect that two objects will be assigned to the same cluster if they are similar. In this way we define *True Positive (TP)* decision assigns two similar objects to the same cluster, a *True Negative (TN)* decision assigns two dissimilar documents to different clusters. There are two types of errors we can commit: a *False Positive (FP)* decision assigns two dissimilar documents to the same cluster and a *False Negative(FN)* decision assigns two similar documents to different clusters.

The *Rand index (RI)* measures the percentage of decisions that are correct. That means this metric is simply accuracy measure (used for example in the evaluation of the classification tasks) and it will be

$$RI = \frac{TP + TN}{TP + FP + TN + FN} \tag{1.12}$$

The negative side of this criteria is a fact that Rand Index gives equal weight to false positive and false negative decisions, but in some kind of tasks separating similar objects is worse than putting dissimilar objects to the same cluster [1]. For explicit penalizing false negative decisions we can use so-called *F measure*.

F-measure

F measure is a harmonic mean of two measures called *precision* and *recall*, where precision is a percentage of positive predictions that are correct

$$P = \frac{TP}{TP + FP} \tag{1.13}$$

and recall is a percentage of positive labeled objects that were predicted as positive

$$R = \frac{TP}{TP + FN} \tag{1.14}$$

F-measure that is harmonic mean of that two metrics will be defined as

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{1.15}$$

where β is a constant, that we can use to penalize false negatives more strongly than false positives by selecting its value > 1, thus giving more weight to recall [1].

Homogeneity, Completeness and V-measure

A clustering result satisfies *homogeneity* if all of its clusters contain only data points which are members of a single class. Meanwhile a clustering result satisfies completeness if all the data points that are members of a given class

are elements of the same cluster. The homogeneity and completeness are in some ways opposed: increasing the homogeneity of a clustering solution often results in decreasing its completeness. For example two degenerate clustering solutions can happen. To start with, assigning every data point into a single cluster guarantees perfect completeness. However, this cluster would be as inhomogeneous as possible, since all classes are united. Another solution suggests to assign each data point to a distinct cluster. This guarantees perfect homogeneity — each cluster trivially contains only members of a single class. However, completeness will be very low.

V-measure is an entropy-based measure which explicitly measures how successfully the criteria of homogeneity and completeness have been satisfied. V-measure is the harmonic mean of distinct homogeneity and completeness scores[32], just as precision and recall are commonly combined into F-measure [33].

If we assume a data set comprising N data points, and two partitions of these: a set of classes, $C = ci|i = 1, ..., n$ and a set of clusters, $K = ki|1, ..., m$, then homogeneity and completeness scores are formally given by:

$$h = 1 - \frac{H(C|K)}{H(C)} \tag{1.16}$$

$$c = 1 - \frac{H(K|C)}{H(K)} \tag{1.17}$$

where $H(C|K)$ is the conditional entropy of the classes given the cluster assignments. With n the total number of samples, n_c and n_k the number of samples respectively belonging to class c and cluster k, and finally $n_{c,k}$ the number of samples from class c assigned to cluster k, conditional entropy is given by

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{n_{c,k}}{n}\log\left(\frac{n_{c,k}}{n_k}\right) \tag{1.18}$$

and entropy of the classes $H(C)$ is given by

$$H(C) = -\sum_{c=1}^{|C|}\frac{n_c}{n}\log\left(\frac{n_c}{n}\right) \tag{1.19}$$

While V-measure is the weighted harmonic mean of homogeneity and completeness, it can be defined as

$$V_\beta = \frac{(1+\beta)\cdot h\cdot c}{(\beta\cdot h)+c} \tag{1.20}$$

Same as F-measure, if β is greater than 1 completeness is weighted more strongly, and if β is less than 1, homogeneity would be weighted more strongly.

1.5.2 Internal criteria

Typically clustering algorithm must formalize the goal of achieving high similarity between objects in same cluster and low similarity between objects from different clusters. This is the internal criteria for clustering quality. Common variants are minimization of average squared distance of objects from their cluster centers, minimization of the loss in mutual information between the input data matrix and its approximation based on clustering, minimization of Renyi's entropy, etc [9].

1.5.3 Relative criteria

While constructing an index for evaluation of clustering and selection of an optimal clustering scheme we have two main criteria [9]:

1. *Compactness*, the members of each cluster should be as close to each other as possible. A common measure of compactness is the variance, which should be minimized.

2. *Separation*, the clusters themselves should be widely spaced. There are three common approaches measuring the distance between two different clusters:

   - *Single linkage*: It measures the distance between the closest members of the clusters.
   - *Complete linkage*: It measures the distance between the most distant members.
   - *Comparison of centroids*: It measures the distance between the centers of the clusters

There will be described some popular indices: C-index and Silhouette coefficient.

C-index

Assume that p is the number of all pairs of samples that are located in the same cluster. S will be the sum of the distances between samples in those pairs. Then we find p of all pairs in dataset with the smallest distances (then S_{min} will be the sum of them) and p of them with the biggest (S_{max} will be the sum of them).

Then *C-index* is defined as

$$C = \frac{S - S_{min}}{S_{max} - S_{min}} \tag{1.21}$$

From that equation it follows, that if more pairs of samples with the small distances are in the same cluster, the numerator is smaller. That means that

value of C-index will be also small, thus, better clustering scheme [9]. The form of denominator is chosen for normalization, causing normalized values of $C \in [0, 1]$.

This parameter is generally higher for convex clusters than other concepts of clusters, such as density based clusters like those obtained through DBSCAN algorithm.

Silhouette coefficient

Silhouette coefficient is close in meaning to C-index such that it is also internal criteria, which gives us interpretation and validation of consistency within clusters [16]. For every sample i it is defined as

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \tag{1.22}$$

where:

- a_i - the mean distance between a sample and all other points in the same class;
- b_i - the mean distance between a sample and all other points in the next nearest cluster;

what causes values in range of [-1;1], where negative values mean bad clustering and positive values mean good clustering. The average s_i over all data of a cluster is a measure of how tightly grouped all the data in the cluster are. The Silhouette Coefficient is generally higher for convex clusters than other concepts of clusters, such as density-based clusters like those obtained through DBSCAN.

Chapter 2

Implementation and used technologies

As the main goal of this work is to study the effectiveness of different unsupervised methods for applying in astronomy, exactly for spectra studying, we tried to cover as much methods as possible. This chapter contains information about:

- used libraries and technologies,
- implemented algorithms (sequential and parallelized),
- tools used for post-processing.

2.1 Scikit-learn

Python open source library scikit-learn contains simple and efficient tools for data mining and data analysis reusable in various contexts, also built on NumPy, SciPy and matplotlib Python libraries [34]. This library was the main tool used in this work. Further, we will describe algorithms, techniques and indexes that has some implementation features in scikit-learn library.

2.1.1 Method for initializing centroids - K-means++

Traditionally, K-means (and also Mini-batch K-means) algorithm starts with k arbitrary centers, typically chosen randomly. Each point is then assigned to the nearest center, and each center is recomputed as the center of mass of all points assigned to it [35]. The last two steps are repeated until stabilization.

K-means++ is an initializing method that chooses centers at random from the data points, but assigns weights to the data points according to their squared distance squared to the closest center already chosen.

Choosing centers in this way is both fast and simple, and it already achieves guarantees that k-means cannot [35]. This combined algorithm is also called *k-means++*.

This is one of the proposed methods for K-means and Mini-batch K-means in scikit-learn. As we show in the experiments, this method helps to significantly reduce computation time.

2.1.2 Adaptation of cluster validity indexes

For the result estimation we used several criteria described in section 1.5. Among them are: purity, inertia, homogeneity, completeness, v-measure, rand index, mutual information and silhouette coefficient.

Should be also mentioned that for cluster validation we have used Adjusted Mutual Information instead of Normalized Mutual Information and adjusted variant of Rand Index that is defined in a similar way as Adjusted Mutual Information (see 1.5.1).

Purity is not provided in scikit-learn. This validity index was implemented in Python within our framework (see 2.3).

2.1.3 Biclustering implementations

Biclustering algorithms simultaneously cluster rows and columns of a data matrix. So, in our case, this method could help us to find the similarity between spectra based on subsequences of intensities that have some correlation between them. In other words this method helps to find out spectra that are similar in different sub-arrays of intensities (belong to same row bicluster).

We have tested two biclustering algorithms, provided by scikit-learn library: Spectral Biclustering (Kluger, 2003) and Spectral Co-Clustering (Dhillon, 2001).

Co-Clustering algorithm finds biclusters with values higher than in corresponding rows and columns. It creates biclustering, where each row and each column belongs to exactly one bicluster [36], so we can ask algorithms to find the exact number of biclusters, which after rearranging would appear at *diagonal* of matrix. The algorithm treats the input data matrix as a bipartite graph: the rows and columns of the matrix correspond to the two sets of vertices and each entry corresponds to an edge between a row and a column. The goal of the algorithm is to approximate the normalized cut of this graph to find heavy subgraphs. An approximate solution to the optimal normalized cut may be found via the generalized eigenvalue decomposition of the Laplacian of the graph. When input matrix A has size $m \times n$, Laplacian matrix has the size of $(m + n) \times (m + n)$. More efficient is to use the initial matrix, so it should be normalized as:

$$A_n = R^{-\frac{1}{2}} A C^{-\frac{1}{2}}$$

where R is the diagonal matrix with entry i equal to $\sum_j A_{ij}$ and C is the diagonal matrix with entry j equal to $\sum_i A_{ij}$. The singular value decomposition $A_n = U\Sigma V^\top$ provides the partitions of the rows and columns of A. A subset of the left singular vectors gives the row partitions, and a subset of the right singular vectors gives the column partitions. These singular vectors provide the desired information.

Spectral Biclustering creates biclusters assuming that data matrix contains some hidden *checkboard structure*. That's why the rows and columns may be partitioned to biclusters, which entries are approximately constant in the Cartesian product of row clusters and column clusters.

This method uses different types of normalization:

- *Bi-stochastization* makes both rows and columns sum to the same constant by repeated row and column normalization until convergence;
- *Log normalization (log-interactions)*: takes logarithm of data matrix as $L_{ij} = \log A_{ij}$. Defining *column mean* as $\overline{L_{i\cdot}} = \frac{1}{m}\sum_{i=1}^{m} L_{ij}$,similarly *row mean* $\overline{L_{\cdot j}}$, and *overall mean* $\overline{L_{\cdot\cdot}} = \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m} L_{ij}$, we then can define final matrix as $K_{ij} = L_{ij} - \overline{L_{i\cdot}} - \overline{L_{\cdot j}} + \overline{L_{\cdot\cdot}}$

This biclustering makes a separation where each row belongs to all column biclusters and each column belongs to all row biclusters.

2.2 Apache Spark and MetaCentrum

MetaCentrum project is an activity of the CESNET association. It operates and manages distributed computing infrastructure consisting of computing and storage resources owned by CESNET as well as those of co-operative academic centers within the Czech Republic. MetaCentrum is responsible for building the National Grid and its integration to related international activities [37]. Users registered in MetaCentrum get free access to a wide range of application software and utilities such as Matlab, Maple or Hadoop.

Hadoop is a well-known open-source framework for distributed storage and processing of large volumes of data. Hadoop is especially useful for processes, which involve a MapReduce algorithm.

Apache Spark is a fast and general engine for large-scale data processing working with *Hadoop Distributed File System (HDFS)*. HDFS is the primary distributed storage used by Hadoop applications. A HDFS cluster primarily consists of a NameNode that manages the file system metadata and DataNodes that store the actual data.

We decided that Apache Spark would be appropriate for parallel computations of big data from LAMOST dataset. Apache Spark provides API for several languages: Scala, SparkR, Java and Python, this simplifies integration to our framework. It also contains scalable machine learning library MLlib.

Unfortunately, MLlib contains only few clustering algorithms, among them are k-means and Gaussian mixtures (GMM) [38]. Also on servers with Hadoop is prohibited to have more than 2^{20} files in one directory [39]. This limit along with another causes (see subsection "Experiments on LAMOST Dataset" 3.3) forced us to reduce test group of LAMOST dataset.

2.3 Implementations

We implemented framework that includes several algorithms for clustering and dimensionality reduction from scikit-learn library to facilitate a test process. It also contains our own implementation of *purity score* and two variants of *local outlier factor* - sequential and parallel adapted for Apache Spark.

Purity
Implemented framework contains sequential algorithm for computing purity score in Python according to description given in 1.5.1. Purity score is an external index, so as an input it requires not only assigned clusters but true classes of samples.

Local Outlier Factor algorithms
Local outlier factor (described in 1.4.1) is an algorithm that has several main stages of computing :

1. initializing: this step can be expensive on computation time if the normalization is conducted;
2. computing of local outlier factor for every object:
    a) k distance computing: computation time depends on the computation time of distance function. Using precomputed distances can reduce time but would be expensive on memory in case of a big dataset;
    b) computing of reachability distances: time cost depends on input parameter k;
    c) local reachability density: computation time depends on the closeness of objects in space (set of the k nearest neighbor ($N_k(p)$) can contain more than k objects)
    d) computing of local outlier factor using ratio arrays of the nearest neighbors depends also on size of $N_k(p)$.

The most expensive step in terms of computational time is the computing of distances the between elements, so total time complexity is $O(n^2)$. The main problem of algorithm parallelization for Spark is the necessity of storing many information on every cluster since the distances between every pair of samples must be calculated.

2.4 Post-processing

Post-processing phase includes cluster validation and visualization of clusters. Cluster validity methods are applied if ground truth classes are given. Output also contains a cluster assignments for all samples.

Output clusters are visualized with libraries *matplotlib* and *plotly*. Also for several visualizations we used program SPLAT-VO - a powerful graphical tool for displaying, comparing, modifying and analyzing astronomical spectra, as well as searching and retrieving spectra from services around the world using Virtual Observatory (VO) protocols and services [40].

Chapter 3

Experiments and discussion

The main complication of unsupervised learning is a hard verification of results. In our case this mean visual validation of clusters and outliers which is time-consuming [6]. Generally there is no other way to verify results but to do it manually. That's why small labeled group of Ondřejov spectra was the first testing group for us. All details will be fully described below, but generally experiments were carried out in this stages:

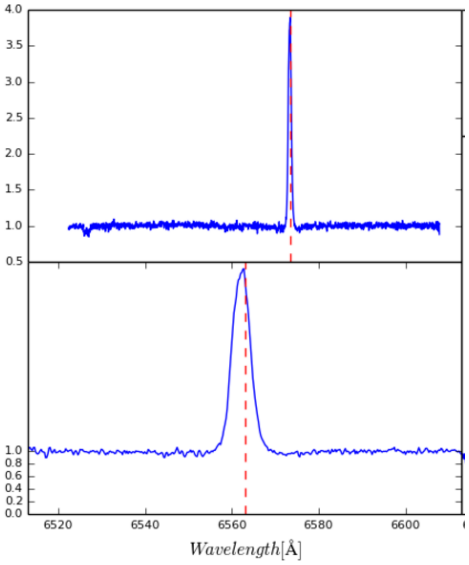
- experiments on a small part of data from Ondřejov (testing set) with verification using criteria described in 1.5,
- experiments on all data from Ondřejov with manual verification of result,
- experiments on data from LAMOST.

This chapter contains a description of the whole experiments and discussion of obtained results. For the sake of briefness we provide only most important results' visualizations. The whole experiments output – almost all images and estimations of experiments are given in attachment of enclosed CD.

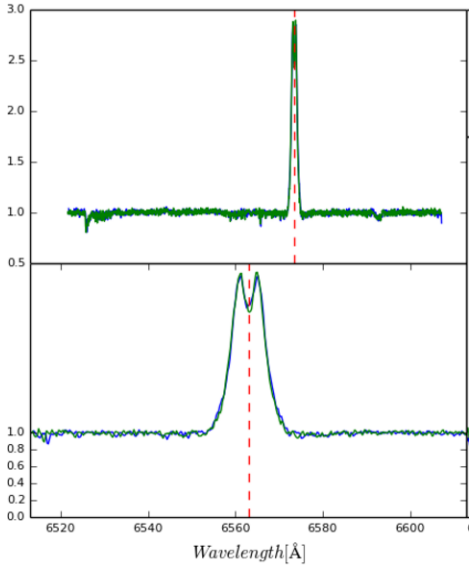
3.1 Experiments on labeled part of Ondřejov dataset

Telescope of Ondřejov observatory is already nearly half a century focused on observation Be stars. Some of these observations were unified to catalog of stars with sets of their spectra. That's why Ondřejov dataset contains a small group of spectra that were manually labeled, this means exact w.r.t. astronomy. So there is a group of *1726* spectra separated to *5* groups. We must notice that the last group contains spectra of stars, which some way evolved in time.

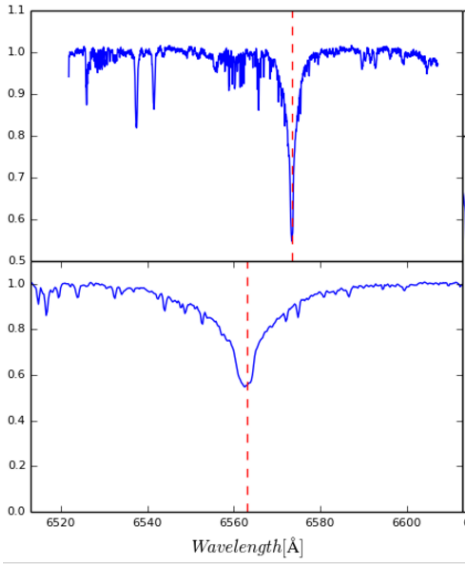
[6] Should be noticed, that visual validation was conducted on random chosen exemplars. We tried to take the most different spectra from all parts of datasets.



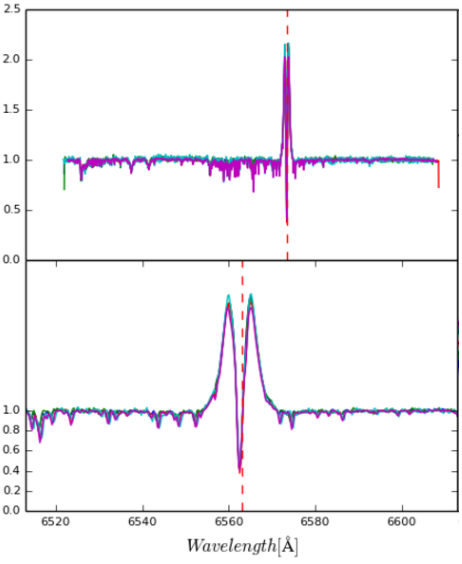
significant single peak - pure emission



double emissions



pure absorption



absorption line with an emission

Figure 3.1: Samples of groups 1-4 from labeled Ondřejov dataset in two variants: whole spectrum and zoom in area near H_α line (Source: `http://physics.muni.cz/~ssa/archive/`)

Many of them have absorptions or absorptions with emission, so strictly speaking it is not a fully independent group of spectra. The 5th group have arisen by cataloging of stars, which demonstrated some radical changes, so visually changed their class. Exemplars of the first four groups are represented

in figure 3.1. The number of samples in every group is 178, 172, 1188, 56, 132 respectively, percentage of every group of spectra is: 1st group - 10.3 %; 2nd group - 9.9 %, 3rd group - 68.8 %, 4th group - 3.3 %, 5th group - 7.7 %. This groups can be identified as spectra with:

1. significant peak without double emission,
2. double emission,
3. absorption,
4. absorption with emission,
5. chaos in emission (time evolving objects).

This spectra (further named as *testing set*) were used for the initial testing of the algorithms because this testing set provides information for clusters' evaluations. We have applied some common clustering algorithms and used several criteria, described in 1.5. Also we have used several methods of dimensionality reductions to define the most appropriate for this kind of data.

3.1.1 Clustering algorithms

Applied algorithms were selected to cover the greatest variety of techniques. Among them are: K-means, Mini-batch K-means, Affinity Propagation, Biclustering and DBSCAN.

3.1.1.1 K-means and Mini-batch K-means

A typical problem of K-means algorithm is non-balanced data and as we can determine clusters are not located separately in space (w.r.t. geometric distances – Euclidean distance). Another important point is initial seeding of centroids, because random seeding can take elements in immediate closeness. While K-means tend to stuck in local minimum, random seeding can lead to poor results. Initial seeding has also influence on time. Exists several methods for initial seeding, that can improve results. We tested K-means and Mini-batch K-means with random seeding, *K-means++* seeding (see section 2.1.1) and initial seeds obtained from PCA components.

K-means algorithms haven't shown good ability to cluster this type of data. Best K-means run (with random initial seed) have formed five clusters with quantitative distribution close to reality (92, 200, 181, 3, 1250), but some of them (especially bigger cluster) are mixture of spectra with emission and spectra with absorption in H_α. At the same time PCA, that provides *linear transformation* of data and give information about initial seeding, usually demonstrated three almost empty clusters and two big, which caused high

Table 3.1: The results of K-means and Mini-batch K-means testing on more appropriate found parameters. Green cells indicate the best runs of algorithms. Yellow cells indicates the deceptive measured results (deceptive w.r.t. the whole result of the run)

method		**init parameters**	**time**	**purity**	**inertia**	**homo**	**compl**	**v-meas**	**ARI**	**AMI**	**silh**
K-Means 5 clusters 200 iterations		seed: k-means++	3.16s	0.805	29824	0.374	0.632	0.470	0.593	0.372	0.768
		seed: random	3.51s	0.849	40165	**0.591**	**0.686**	**0.635**	**0.838**	**0.590**	0.661
		seed: PCA-based	0.71s	0.743	89878	0.138	0.637	0.227	0.187	0.135	0.929
Mini-batch K-Means	5 clusters	seed: random, batch size: 20	0.25s	0.823	248457	**0.622**	0.676	**0.648**	**0.850**	0.620	**0.626**
		seed: random, batch size: 45	0.30s	0.859	197649	0.614	0.674	0.643	0.847	0.612	0.645
		seed: k-means++, batch size: 45	0.23s	0.851	204032	**0.641**	0.658	**0.649**	**0.878**	**0.640**	0.542
	4 clusters	seed: random, batch size: 45	0.27s	0.814	199919	0.603	**0.701**	**0.648**	0.845	0.602	0.618
		seed: k-means++, batch size: 45	0.26s	0.843	200608	0.606	**0.703**	**0.651**	**0.848**	0.605	**0.666**
		seed: PCA-based batch size: 45	0.23s	0.804	563741	0.514	0.492	0.503	0.592	0.490	0.397
		seed: PCA-based batch size: 90	0.30s	0.843	334061	**0.612**	0.514	0.559	0.421	0.512	0.305

silhouette coefficient. Giving a hint about similarity between the groups four and five didn't improved results.

Mini-batch K-means shows by contrast a better result – it can be easily seen from results table - and a better quantitative distribution: for example with batch size 45 algorithms creates cluster of 1254, 95, 139, 193, 45 samples respectively. Also Mini-batch is faster by several orders of magnitude. This is explained by using mini-batches. Smaller size of mini-batch influences computation time positively. This only can be violated by some kind of additional improved stop criteria like a number of iteration without improvement, etc.

Using not random seeds for initializing centroids together with stop criteria described above, we can reduce computation time, what is seen for Mini-Batch with configuration with initial seed got from K-means++ and batch size equals 45. With this configuration algorithm runs faster than with same configuration but with random initializing. Also it was faster than algorithm with batch size 20.

The source of the imprecision of the clustering scheme obtained with K - means and Mini-batch K-means is the fact that from the point of view of algorithm samples with high emission peak are more distant from entire single-emission group that samples with some kind of small double emission peak, etc. Also the implementation provided by the scikit-learn library is adjusted to Euclidean distance. Using other metrics is allowed, but requires big memory overhead [34]. That's why these algorithms were excluded from further experiments.

3.1.1.2 Affinity propagation

All tests conducted with affinity propagation algorithm didn't give acceptable results. The main disadvantage of this method was a formation of the enormous number of clusters (more than hundred). Whereas purity and homogeneity almost reached maximum value all other indexes were very low. High purity and homogeneity is explained just by the large number of clusters. Also computation time was several times longer (configuration with 200 iterations takes about 11 seconds). We decided to exclude this method from further experiments.

3.1.1.3 Biclustering

In our research we tried a different usage of both Spectral Co-clustering and Spectral Biclustering algorithms described in 2.1.3 to reach the best clustering scheme. Figure 3.2 shows an initial state of the testing set.

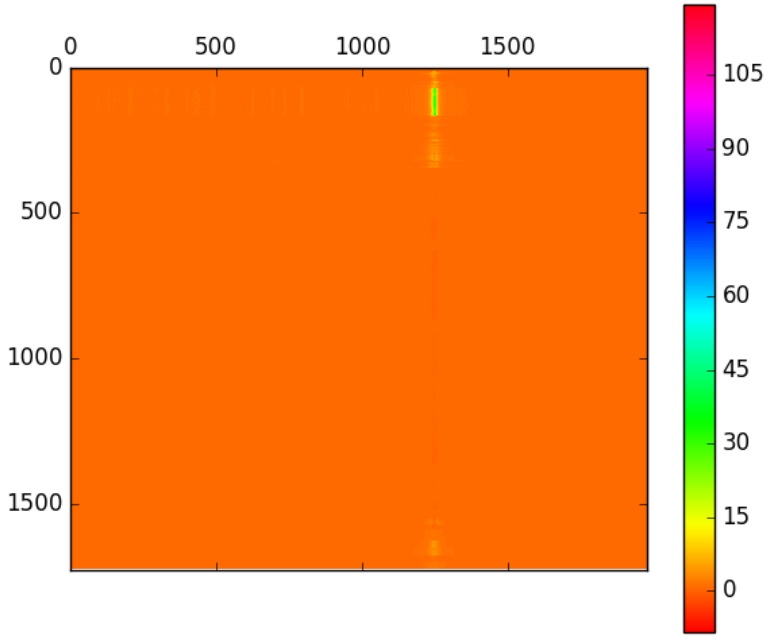


Figure 3.2: The initial view of data in *testing set*. The columns are wavelengths and each row is the spectrum itself. The colors reflect values of intensities. Most of all intensities have value about 1 because of the normalization, which is displayed as red color. It is seen that some spectra have higher values of emission (yellow and green spots).

From testing we made a conclusion, that both algorithms have shown tend to create one bigger cluster with all values around 1 and several smaller clusters with the extremely high emission peaks' values compared to the first cluster. The disadvantage of this clustering is in fact, that spectra with small emission peak and spectra with absorption are put into the same cluster (same as with K-means). Example figure 3.3 represents a view of this whole big cluster and figure 3.4 represents some random chosen spectra to show the difference between spectra put in that cluster (result obtained from the Spectral Biclustering algorithm with input configuration $(5, 8)$). This happens when naive configurations are used, such as attempt to obtain exactly 5 clusters. As number of biclusters is not enough for algorithm, it prefers to isolate some far different spectra with great value of emission to separating spectra which have "small" value difference in one small area. Example of this far different spectra is the spectra with an emission peak in both H_α ($\sim$6560 angstroms) and a He I 6678A line ($\sim$6678 angstroms, example of this spectrum is presented in figure 3.5). These spectra are separated to the particular group from spectra with only one standard peak in H_α and in term of astronomy they are the outliers, because this type of spectra usually belongs to very hot star.

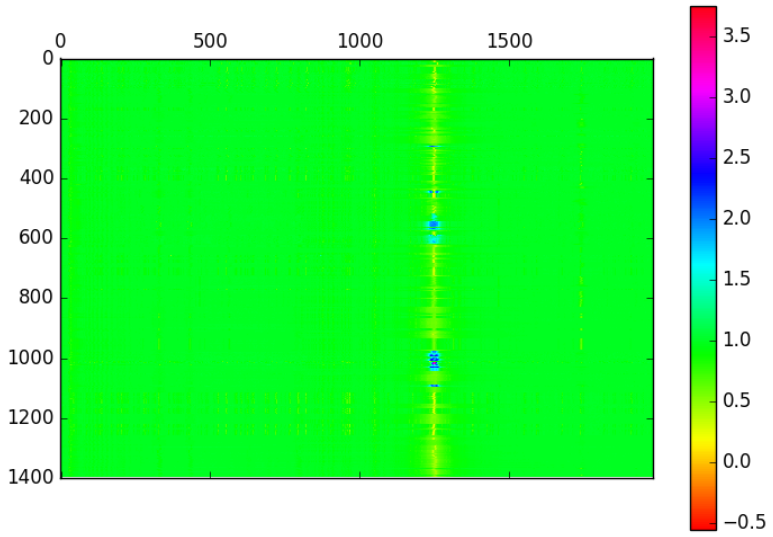


Figure 3.3: The view of the whole mixed cluster of spectra with small values of emission (blue spots) and absorption (yellow spots)

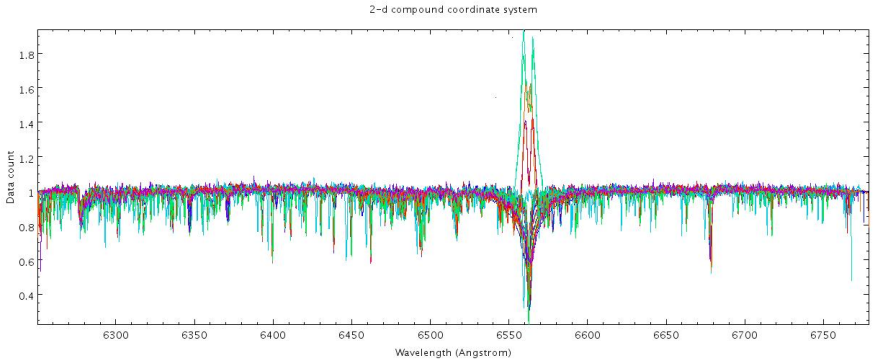


Figure 3.4: Comparison of some random chosen spectra from the mixed cluster. It can be easily seen that some spectra have emission peak and some spectra have absorption.

Table 3.2: Spectral Biclustering results

time	purity	homo	compl	v-meas	ARI	AMI	silhouette	number of biclusters	
								row	columns
0.94s	0.804	0.373	0.631	0.469	0.592	0.371	0.777	5	5
1.11s	0.804	0.373	0.631	0.469	0.592	0.371	0.773	5	10
1.10s	0.841	0.584	0.655	0.618	0.829	0.582	0.612	7	8
1.06s	0.849	0.616	0.633	0.625	0.846	0.613	0.614	8	8
1.12s	0.849	0.617	0.634	0.625	0.846	0.614	0.589	9	9
1.11s	0.874	0.656	0.647	0.651	0.851	0.644	0.647	10	6
1.11s	0.886	0.692	0.626	0.657	0.865	0.622	0.574	12	6
1.17s	0.884	0.707	0.442	0.544	0.430	0.437	0.340	14	6

Table 3.3: Spectral Co-clustering results. Input parameter of bicluster number is not the same as output bicluster number, because it implies number of diagonal clusters.

input # of biclusters	time	purity	homo-genity	complet-ness	v-measure	ARI	AMI	silhu	output biclusters number	
									row	column
7	0.41s	0.800	0.318	0.565	0.407	0.544	0.316	0.782	5	5
10	0.38s	0.791	0.389	0.370	0.379	0.410	0.367	0.346	6	8
13	0.44s	0.808	0.443	0.378	0.408	0.424	0.374	0.299	7	10
17	0.52s	0.799	0.427	0.245	0.312	0.191	0.241	0.152	11	13

Table 3.2 contains result of Spectral Biclustering with different input configuration. In that table bi-stochastic normalization of Spectral Biclustering (Kluger, 2003) is not represented among others due to lack of any significant differences in effectiveness compared to log normalization of the same algorithm. From this table it is seen, that simple increasing of amount of biclusters can improve results. Also it is seen, that number of column biclusters have less impact than number of row bicluster. The best found configuration of Spectral Biclustering is $(10, 6)$, which is confirmed by visual comparison of clusters (see 3.5). This clustering scheme is such that most of absorption spectra are in the particular group and spectra with emissions are divided by height of peak, etc.

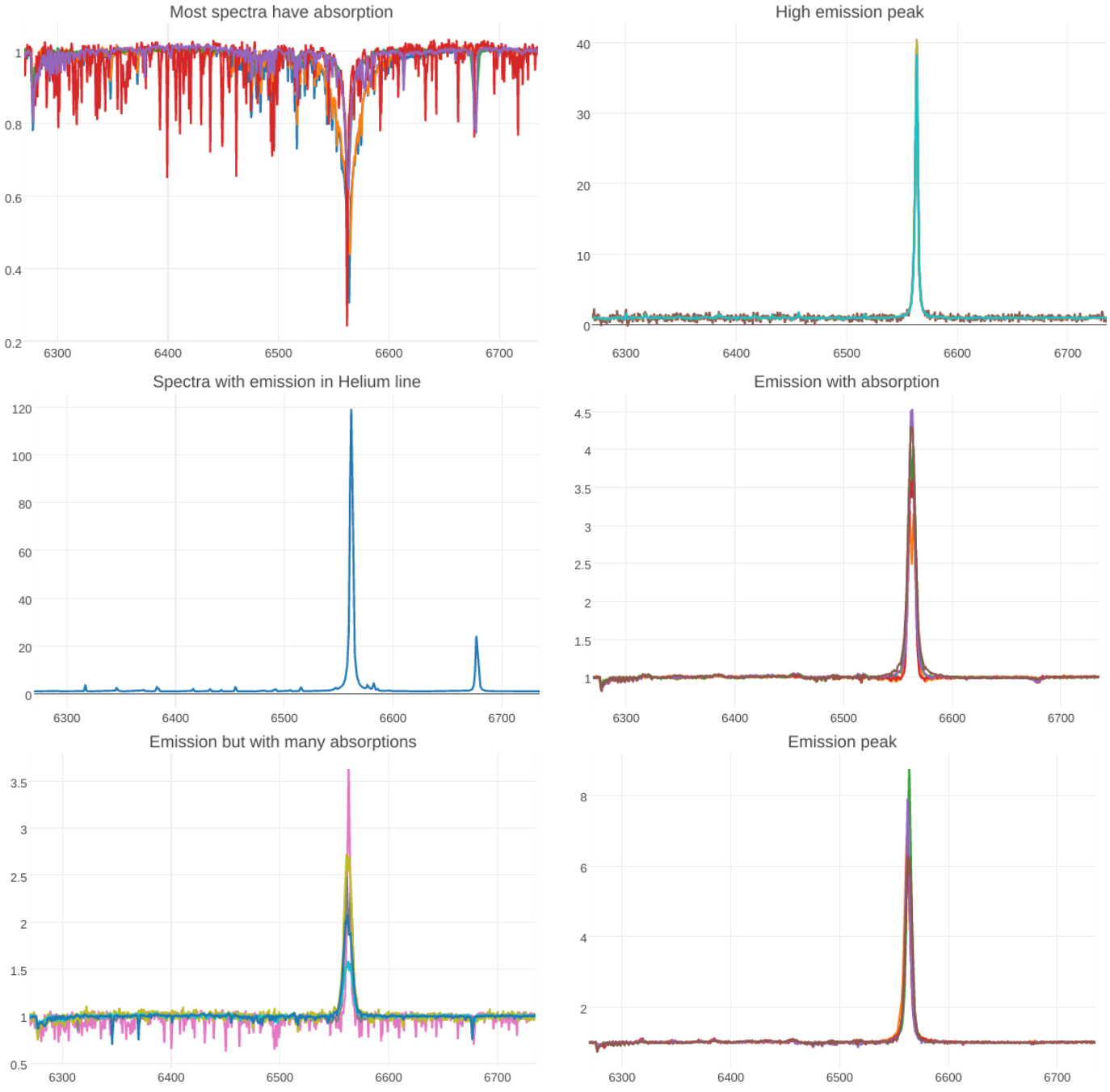


Figure 3.5: Result clusters (6 of 10) of Spectral Biclustering with configuration (10,6) on *testing set*. There are 5 spectra of each cluster on images (and only single spectrum with emission in helium line). Other 4 clusters are simply spectra with emission peak of different values of heights, etc.

Spectral Co-clustering usually also formed one big cluster with absorption and small emission peaks. Best result according to validity indexes 3.3 is the one with 7 diagonal clusters (5 row and 5 column output biclusters). The most interesting clustering scheme was obtained with input number of biclusters equaled 17: some clusters contained mixture of spectra with absorption together with spectra with double peak emission, but all of emissions and absorptions had almost the same width.

In our work we also tried to find subsequences of attributes that can represent physical feature of object. Biclustering have demonstrated the ability not only to cluster spectra with some kind of logic, but also gave us the information about connections and correlations between every wavelength. In most cases wavelengths were divided in such way: the biggest bicluster contains all wavelengths except diapason $\sim 6560 - 6580$, other biclusters contain parts of this diapason, i.e. it distinguishes the areas where emission and absorption lines usually appear.

Although results of Spectral Co-clustering does not seem convincing enough to continue experiments with it.

3.1.1.4 DBSCAN

DBSCAN algorithms can use any type of measure for nearest neighbors search. We tried two different measures: classic euclidean distance and cosine distance. Euclidean distance have shown better results, so we will start with describing clustering schemes obtained with cosine distance. *Cosine distance* is a term signifying the complement to cosine similarity in positive space.

Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. Cosine similarity applied in positive space gives values bounded in $[0, 1]$. Results obtained from external and internal validity criteria are given in the table 3.4.

Estimation of density-based techniques like DBSCAN does not need such parameters as *inertia*, because does not consider distance as connectivity or centroid-based algorithms do (detail in section 1.2.2.2). But this algorithm defines number of clusters itself, so this would be important for our study. Also the Silhouette Coefficient is generally higher for convex clusters than other concepts of clusters, such as clusters obtained through DBSCAN. Later we will discuss results of DBSCAN more detailed. But it should be mentioned, that high silhouette coefficient values is deceptive with respect to formed clusters. Also it can be seen from table, that not only silhouette coefficient but high purity can be obtained with bad formed clusters.

ε, m	Config.	Time	Purity	Hom.	Coml.	V-meas.	ARI	AMI	Silh.	N_o	N_c
Euclidean distance	1.2;20	20.46s	0.800	0.526	0.610	0.565	0.703	0.523	0.050	502	6
	1.0;20	18.06s	0.744	0.438	0.485	0.460	0.506	0.435	0.068	605	5
	1.2;35	19.58s	0.727	0.408	0.640	0.499	0.604	0.408	0.407	630	2
	1.5;50	21.86s	0.761	0.466	0.763	0.578	0.714	0.465	0.483	555	2
Cosine distance	0.01;50	7.70s	0.749	0.154	0.595	0.245	0.215	0.152	0.852	19	3
	0.05;30	7.78s	0.743	0.138	0.642	0.227	0.187	0.136	0.922	3	3
	0.05;50	7.53s	0.743	0.138	0.642	0.227	0.187	0.136	0.922	3	3
	0.005;10	7.58s	0.749	0.158	0.576	0.248	0.227	0.156	0.918	26	3
	0.005;50	7.70s	0.761	0.193	0.580	0.290	0.281	0.191	0.847	54	3

Table 3.4: Results of DBSCAN using euclidean distance and cosine distance. N_o and N_c are the number of outliers and clusters respectively. Green cells indicate best result for measure. Yellow cells of Silhouette coefficient for DBSCAN with cosine distance is used to show deceptive high values.

We tested different configurations of DBSCAN using cosine distance, but this method has not shown to be a useful tool for this kind of research. Probably best configuration ($\varepsilon = 0.005$ and $m = 50$) was interesting just because it segregated familiar mixed group and frequent high emissions and named 54 outliers. Disadvantage is that outliers are often noisy spectra, asymmetric double emission or extremely high peak.

With the euclidean distance we obtained better results. Furthermore with configuration $\varepsilon = 1.2$ and $m = 20$) we obtained well formed clusters, where absorptions were segregated to two different clusters. So no mixing of absorption and emission spectra were noticed. Exemplar of obtained clusters are shown in figure 3.6

Most of outliers here are spectra with very high emissions, that were not included to any cluster because of great distance along axes that describes H_α ($\sim$6560 angstroms) emission line. Another outliers are very noisy spectra with absorption in H_α.

Figure 3.6: Result clusters of DBSCAN with configuration $\varepsilon = 1.2$ and $m = 20$ (using euclidean distance). 10 random exemplars of each cluster is presented for clusters contains spectra: with narrow absorption; with absorptions; with not high double peak; with small emission; with pure emissions.

3.1.1.5 Birch

Version of Scikit-learn library provided on MetaCentrum does not contain any implementation of Birch algorithm. That's why we decided to focus on the other clustering methods, that have proved effectiveness (like DBSCAN and Spectra Biclustering).

3.1.2 Dimensionality reduction

With respect to characteristics of data we applied several methods of dimensionality reduction (LLE, Isomap, PCA and MDS) to find the most appropriate for this spectra data. We evaluated results by using them as input data for algorithms and by visual control since we have truth labels of samples.

Although linear local embedding demonstrated good performance as a tool while classification of SLOAN galaxy spectra [24], we could not achieve same good results. This is due to the difference between data. Galaxies usually has different shape of the spectrum and also experiments on SLOAN were conducted with smaller feature space.

Unfortunately we could not achieve visible improvements of clutering with any of the methods on the testing dataset. Mostly we obtained linear placement of samples and few detached samples. Also due to time and resources constraints we have no ability to implement parallel algorithm for dimensionality reduction or apply them to bigger datasets sequentially (overall time complexity of most algorithm is $O(n^2)$ or $O(n^2 \log n)$ [16]).

3.2 Experiments on the entire Ondřejov dataset

With respect to found information about how would algorithms work on such a data we tested DBSCAN and Spectral Biclustering on whole Ondřejov dataset.

3.2.1 DBSCAN

As DBSCAN uses two input parameters - distance ε and minimum number for cluster m, we tried to predict appropriate parameters for entire Ondřejov dataset. As number of element of specific groups (like spectra with pure absorption, etc.) will be larger for larger dataset, some types of objects like nova [7] can still be rare. Distance ε has another properties. As it is a distance in space between elements and DBSCAN uses simple euclidean distance, then we suggested, that distances in entire dataset should be about the same value. We applied different configurations of this parameters and stay with $\varepsilon = 3.0$ and $m = 15$ (output clustering contain 7 clusters and about 1400 outliers).

[7] Nova usually has emissions in wider range.

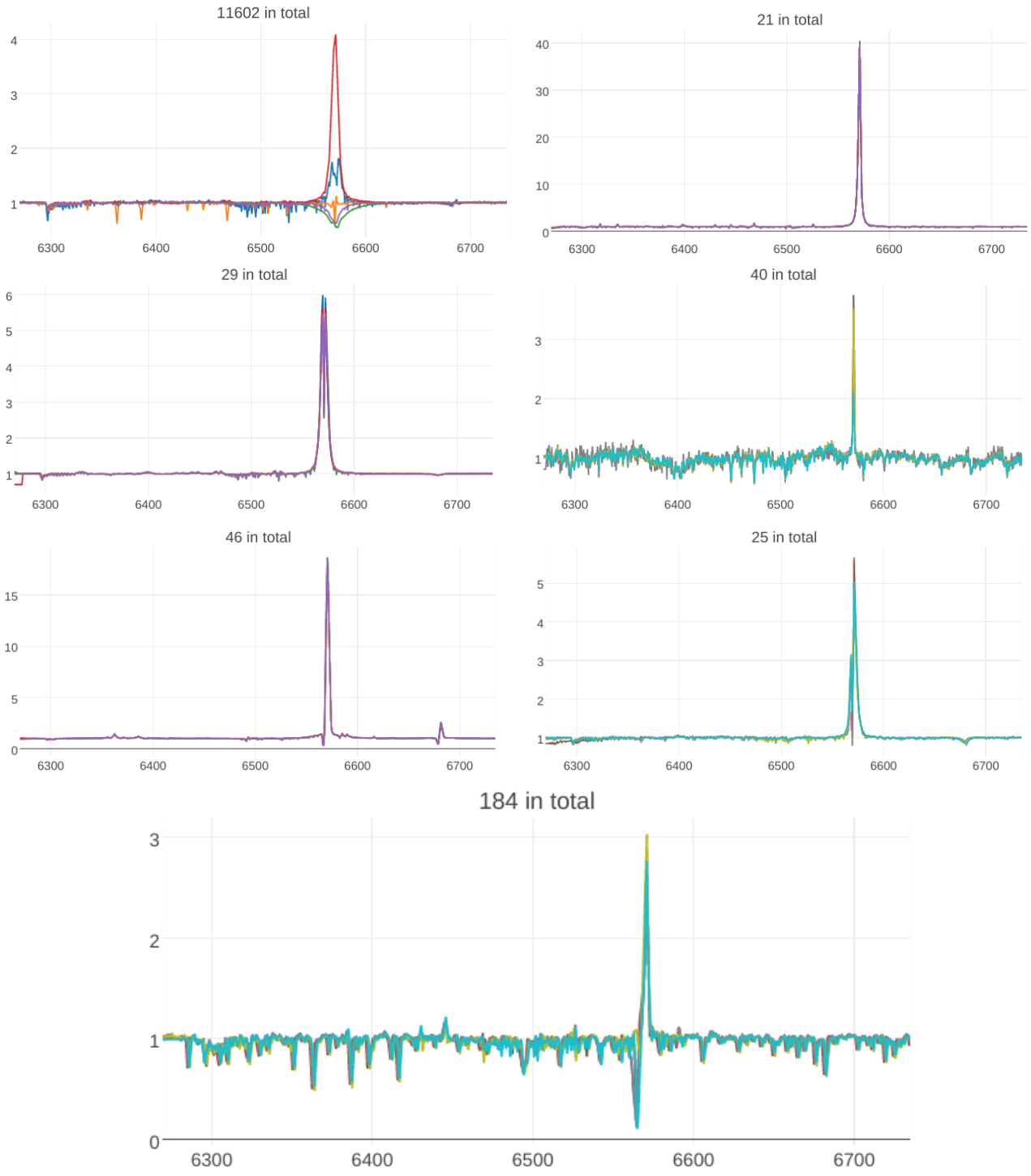


Figure 3.7: DBSCAN ($\varepsilon = 3.0$ and $m = 15$) result clusters of *entire Ondřejov dataset*. First image shows "mixed" cluster of spectra with small emissions and absorptions. Last image is an example of well separated cluster.

Exemplars of clusters obtained with $\varepsilon = 3.0$ and $m = 15$ are shown in figure 3.7. Obtained outliers are generally:

- spectra with high emission not related to any of the clusters;
- spectra of nova stars;
- noise or device artefact;
- asymmetrical double emission.

Setting larger m have led to forming less clusters, big number of outliers (although there are interesting exemplar between them) and one big cluster with spectra with absorption or small emissions. Smaller ε caused distribution of samples to more clusters and more outliers appeared. We checked smaller clusters and discovered, that most of them contains spectra of only few similar stars. So bigger number of clusters can interesting as it can be used to observe evolving of the objects. Since the object belongs to several clusters, it has changed somehow, e.g. an emission has appeared. On the other side bigger number of outliers is an outcome of excluding some spectra with emission from clusters, which contains them when value of ε is larger. At the same time this configuration didn't save us from "big mixed" cluster problem.

3.2.2 Spectral Biclustering

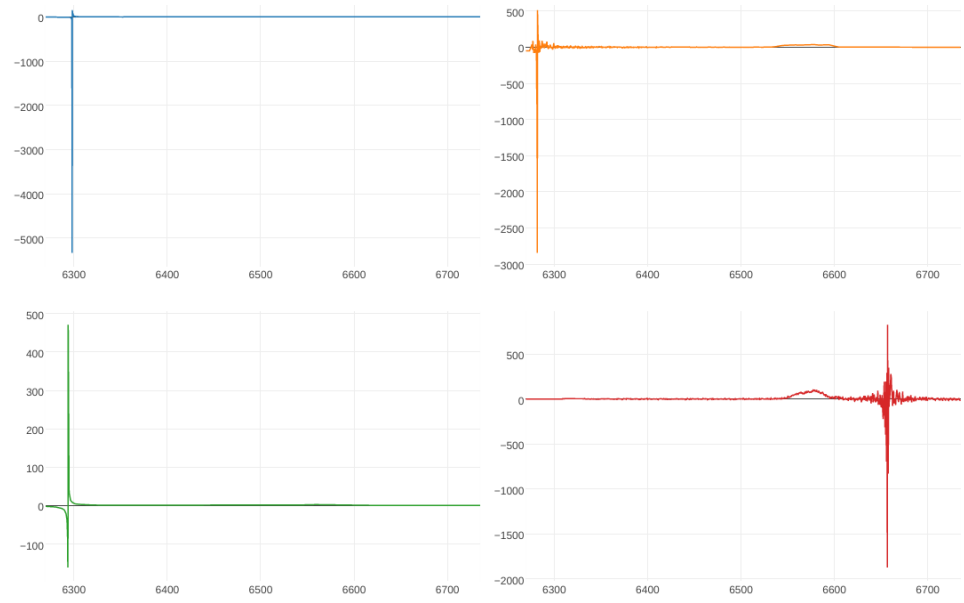


Figure 3.8: Defected spectra example found by using Spectral Biclustering.

During tests on labeled *testing set* of Ondřejov dataset Spectral Biclustering demonstrated ability to divide troubled "mixed" cluster and form sufficiently useful clusters. Although tests on *entire Ondřejov dataset* shows that in more variety and noisy dataset Spectral Biclustering is not same precise.

With the best found configuration $(14, 7)$ algorithm is able to segregate some interesting objects, such as novas, asymmetric double emission or spectra with several emissions to separate clusters. Examples of obtained clusters are given in figure 3.9. Further all clusters which contain single spectrum are obviously malfunction (see figure 3.8). This type of defect most likely causes by numeric error that appears during continuum normalization (see 1.1.1), which is method based on lest square density fitting [41].

Spectral Biclustering also could not divide spectra with absorption and spectra with small emission. Algorithm also put some noisy absorption spectra to the same cluster. Nevertheless, this algorithm was able to form groups of unusual objects that are similar to each other.

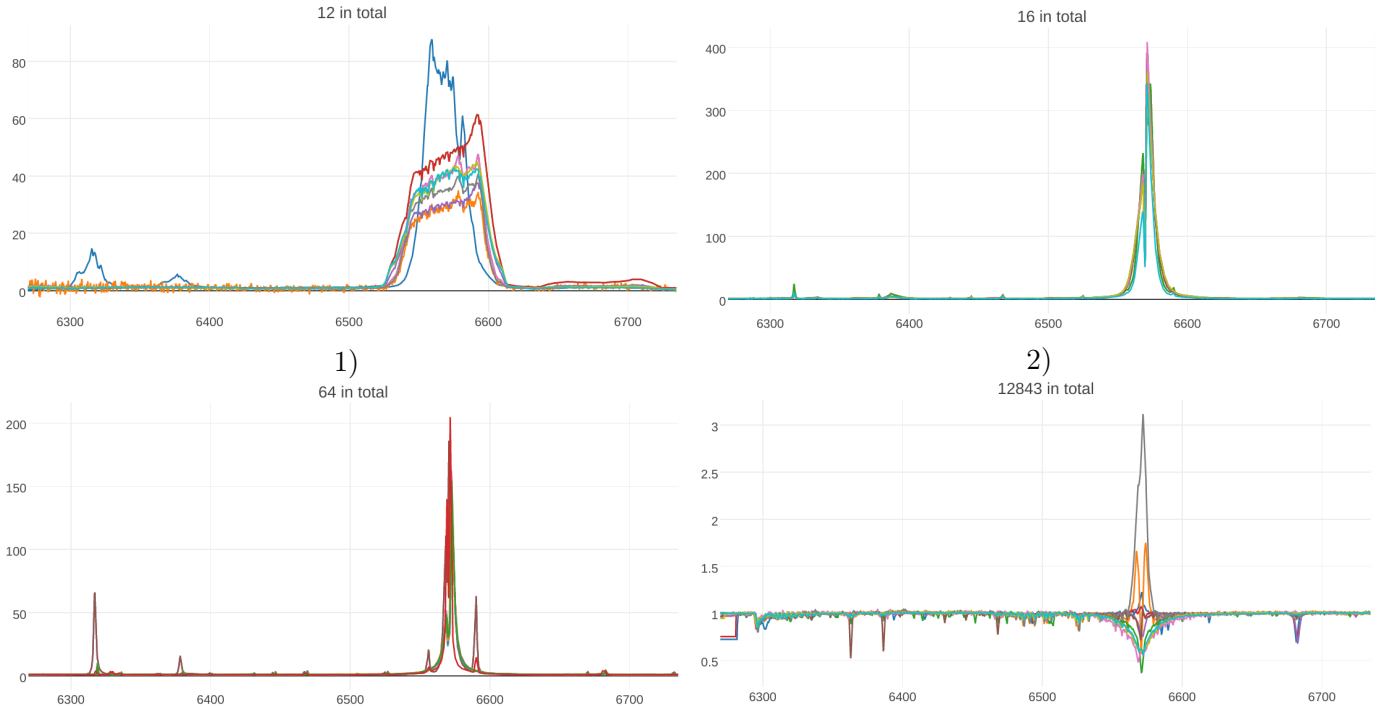


Figure 3.9: Examples of clusters obtained by Spectral Biclustering $(14, 7)$. There are 1) novas, 2) asymmetric extremely high double peak, 3) several high emission lines, 4) "mixed" cluster of spectra with absorptions and emissions

3.3 Experiments on LAMOST dataset

The original intentions of the studying LAMOST data were to apply methods which appeared useful on previous testing to the whole LAMOST dataset. However we faced with several problems, that forced us to focus on local problems.

First there is a limit of allowed file number on MetaCentrum servers, that make us use only a part of LAMOST dataset for experiments. We decided to choose spectra, that were labeled as stars by LAMOST pipelines. The reason for this was the fact that Ondřejov dataset basically contains only spectra of stars: Ondřejov telescope is focused to observe primarily Be stars.

To keep the purity of the experiment, spectra were cropped to H_α area same as spectra from Ondřejov dataset. Because of lower resolution of LAMOST telescope, there are only 546 points in H_α area. Thereby feature space for LAMOST data is almost four time smaller.

Also the Spark server by MetaCentrum provides old versions of some libraries, so integration of our framework was complicated. Then due to much memory consuming data and algorithms we had to reduce the size of testing subset. At final, we had about 100 thousands randomly chosen spectra. Also due to a lack of time there is no chance to implement all algorithms.

In our research the main problem was also to estimate results correctly. Estimation means saying if algorithms gave us truly results w.r.t. astronomical and physical definitions, what requires visual inspection. Because of no powers to proceed results verification on big data of LAMOST dataset, we decided to focus not on clustering but on outlier detection. For this purpose we used local outlier factor algorithm, that was described in subsection 1.4.1.

In general we applied two algorithms: Spectral Biclustering and LOF (in parallel). We will describe obtained results in this part.

LOF experiments

LOF is algorithm for finding outliers based on local density of samples. It was already said (see section 2.3) this algorithms makes a decision based on input parameter k used later to compute *k-distances*, reachability distances, etc. Then we call sample an outlier if it's local outlier factor is more than some threshold t(usually set to 1), but as it was said in section 1.4.1, value of this threshold can be different with different configurations, etc.

So we tried to determine this threshold by fixing value of k. We chose $k = 15$. First we set threshold t to 1, as it is define in classic algorithm description and then gradually increased the threshold. We found that more than 99% of spectra have t in range of less than 3. Little part of data has t even more that 30. Among observed outliers are spectra with:

- extreme falls and rises of values (we don't say emission and absorptions in this case, because this extreme values are obviously defects) or several

emissions or absorptions with high values, places not in H_α, see figure 3.10;

- smooth continuum and single (or double) emission, placed not in H_α region, see figure 3.11;
- noisy horizontal continuum without any emission or absorption (or almost imperceptible), see figure 3.13,
- chaotic spectra, see figure 3.12.

Decreasing of k led to getting more spectra that visually doesn't have emission or absorption. The explanation lies in fact, that LOF assigns scores according to *local* density depending on k-neighborhood. While all values of these spectra lies in small range, more probable they will be neighbors only to each other with relatively small k.

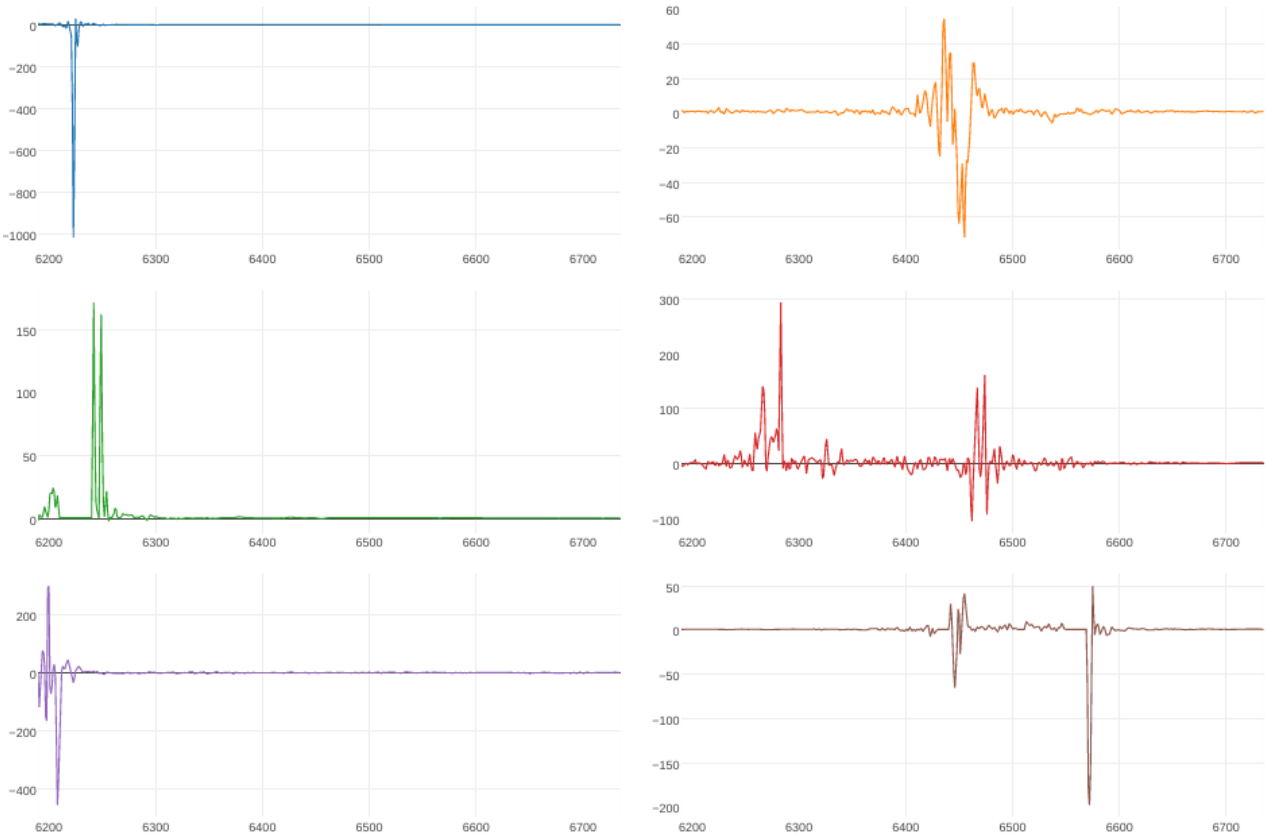


Figure 3.10: Extreme falls and rises of values(even less than zero) or several emissions or absorptions with high values.

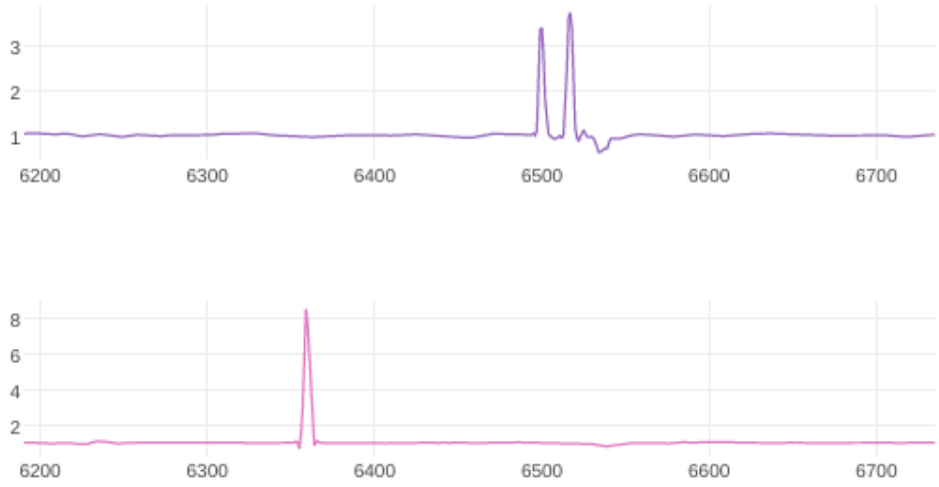


Figure 3.11: Smooth continuum and single (or double) emission placed not in H_α.

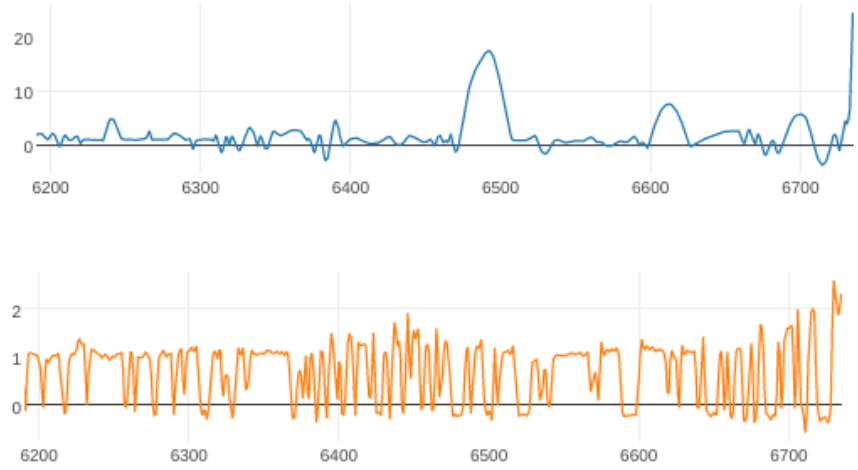


Figure 3.12: several chaotic emissions/absorptions of different width.

Figure 3.13: Noisy (in zoom) horizontal continuum without any emission or absorption (or almost imperceptible). We call it noisy, because deviations are almost same throughout spectrum.

LOF algorithm was parallelized for Apache Spark. Computation time for single launch for our testing subset was approximately 3 hours. Almost the same result was obtained by sequential algorithm. This comes from the fact that Apache Spark is not well-suited for such amount of data. It is designed for computation on large data, which in our case might be several millions of spectra.

Biclustering

Biclustering also was used on LAMOST data. But due to absence of the parallel algorithm designed for Spark test were conducted on a subset of about

100 thousands spectra only. Results obtained from Biclustering algorithm unfortunately were worse, than expected. Every applied configuration resulted in several very small clusters and one big, that contained almost 99% of all spectra. Some clusters were formed from actually similar spectra (see figure 3.14), but dividing this clusters from each other contraries the desired results. Also although the big cluster mostly contains spectra with absorption it is not pure cluster, because many outliers (from the point of view of the clusters) are found there.

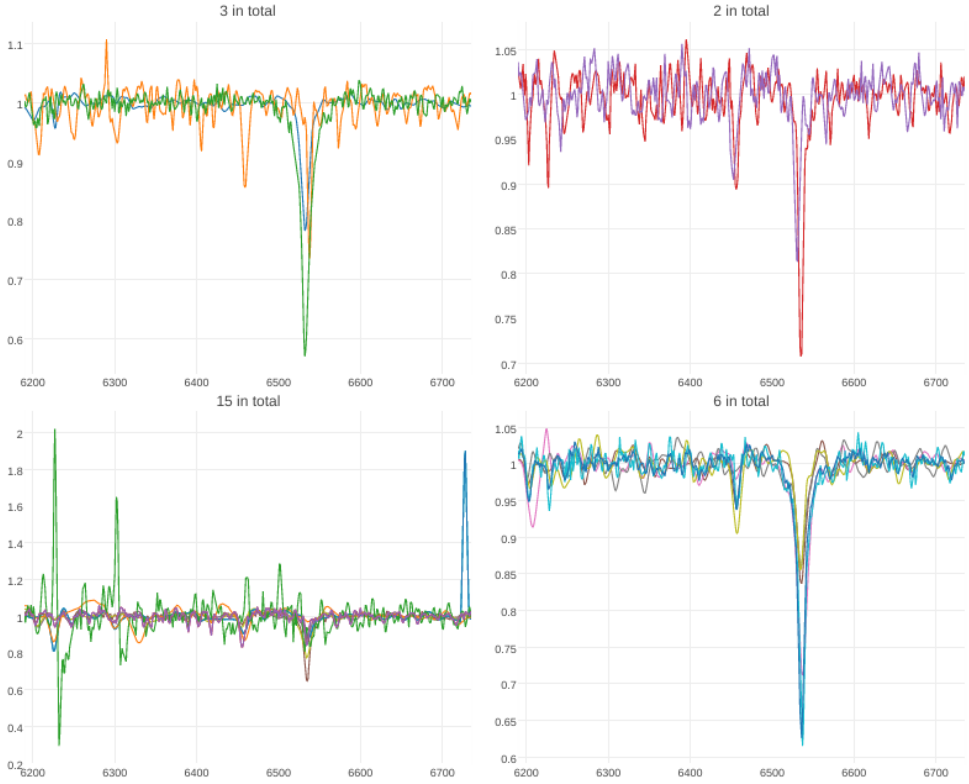


Figure 3.14: Biclustering on LAMOST: examples of small almost consistence clusters (we can see on the left bottom picture a cluster which is obviously not fully consistent

We verified single-element clusters (see figure 3.15) for distinctive features and found several spectra that are obviously damaged. Also it is probable, that some spectra belong to objects that can not be identified just by range of wavelengths, that we tested.

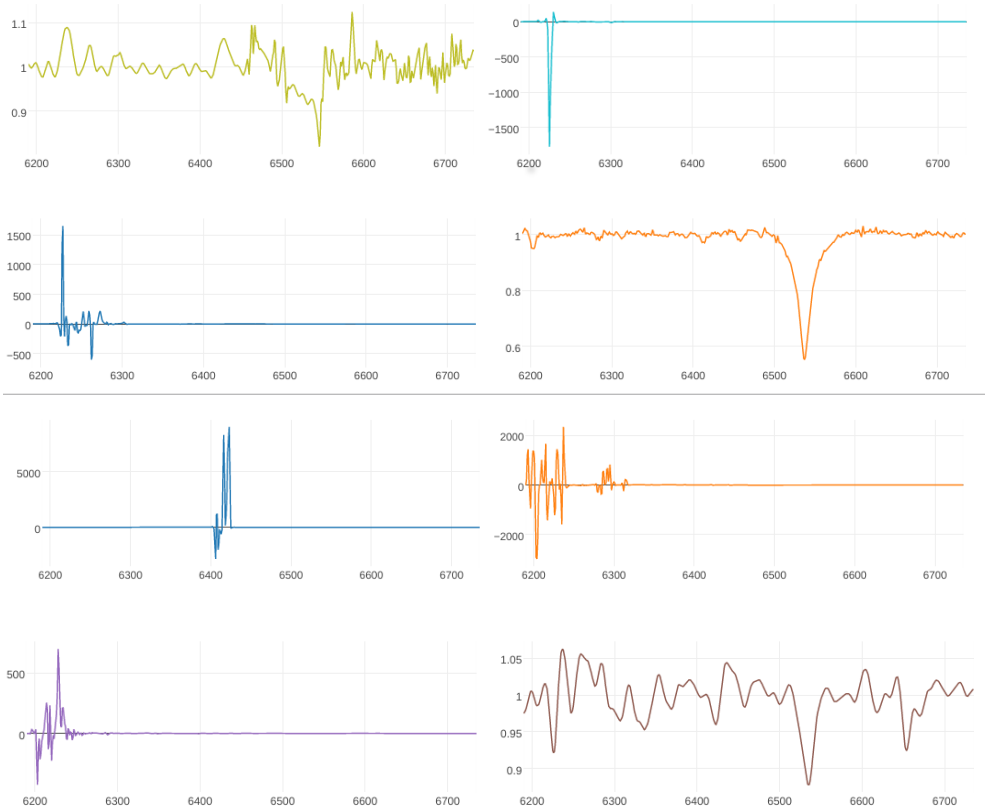


Figure 3.15: Biclustering on LAMOST: examples of single element clusters. Several obviously defected spectra (with leaps of values) are found. Bottom images are interesting as such shapes of spectra can be explained by the fact, that this spectra do not belong to star, but to quasar of galaxy of another object, that has major emission/absorption in different area.

An explanation of why algorithm did not demonstrated same good results as on previous datasets could be imbalance of the shape of input matrix. Test were conducted on about 100 thousands of spectra. On the other side, while the Ondřejov dataset contains spectra with almost two thousands of points in to H_α, LAMOST spectra have lower resolution, that's why spectra cropped to H_α area have only 584 points. And initially this method of biclustering was proposed for data with thousands of features[42].

3.4 Discussion

We attempted to adapt local outlier factor algorithm for Apache Spark, but we can not provide the true assessment of the algorithms' effectiveness. Firstly, Spark is obviously designed for Big Data researches, so in our case that should

be minimum several millions of spectra. For testing with small data amount such as we had there is no any reducing of computation time. On the contrary, time spend for resending data, synchronization, etc., leads to unnecessary waste of time.

On the other side we clarified the behavior of several methods (DBSCAN, Spectral Biclustering, LOF, etc.) on this kind of data and can make some conclusions. Firstly, Density-based DBCSAN was less effective on entire Ondřejov dataset, than on small consistent subset. We expect that adaptation of DBSCAN algorithm with respect to the peculiarities of the spectra might increase the efficiency.

Secondly, Spectral Biclustering is useful enough tool for clustering spectra, but it fast looses accuracy when data are too miscellaneous. This happens because this method expects checkboard structure of data. Such structure is more noticeable in Ondřejov data, so Spectral Biclustering works well on them. But LAMOST data are more varied. Also spectra from the test set from LAMOST are marked as stars by pipeline, but there is no confidence, that all of them are stars indeed. Furthermore we found several spectra, which are likely to be part of spectra of quasars or galaxy. These spectra can be also just damaged. Also we should take into consideration that our dataset of LAMOST contained lower dimensional feature space.

On the other side density-based method LOF shows good results in searching outliers even on LAMOST dataset. It allowed us to find the group of outliers. While some are them are likely instrumental artefacts, another part of them includes spectra that are obviously non-damaged. These spectra may be interesting for astronomers. We can conclude that methods using local density rates (used in LOF) are more appropriate for LAMOST data. Also all algorithms are able at least to detect damaged spectra, which is also useful for improving existing pipelines effectiveness.

Conclusion

In this thesis we examined different methods of unsupervised learning to figure out and understand capabilities of them in exploration of spectral data. We tested chosen methods on the Ondřejov dataset and on the LAMOST dataset and then discuss results. DBSCAN and Spectral Biclustering turn to be the most promising methods for spectra clustering. Also density based algorithm LOF has proved an ability to detect outliers in big datasets. This algorithms can be improved and then used for astronomical issues such as clustering, searching non-usual objects or even improving classification pipelines.

We also implemented some parallel algorithms for Spark and purity index (in Python) for our research. We combined some algorithms of the cluster analysis from scikit-learn library and our implementations of the Purity Score and Spectral Clustering to a single framework to simplify testing process.

For future work we would like to improve results obtained with Spectral Biclustering, DBSCAN and LOF by combining their features. Then we would like to apply reduction methods to LAMOST dataset (possibly adapt dimensionality reduction algorithms for Apache Spark). Also we would like to set our implementations for VO-Cloud environment(also named after its first application VO-KOREL [39]).

Bibliography

[1] Manning, C. D.; Raghavan, P.; Schütze, H. *Introduction to Information Retrieval. Cambridge University Press.* (2007).

[2] National Astronomical Observatories Chinese Academy of Sciences. LAMOST, *Data Release 1* [online]. May 2013, [Cited 2016-5-4]. Available from: `http://www.lamost.org/public/dr1?locale=en`

[3] Rivinius, T.; Carciofi, A. C.; Martayan, C. Classical Be stars. *The Astronomy and Astrophysics Review*, volume 21, no. 1, 2013: pp. 1–86.

[4] Lopatovský, L. Application of Self-Organizing Maps in Astroinformatics. 2014.

[5] Wells, D. C.; Greisen, E. W.; Harten, R. H. FITS - a Flexible Image Transport System. volume 44, June 1981: p. 363.

[6] Stellar department of the Astronomical Institute of the Czech Academy of Sciences. Available from: `http://vos2.asu.cas.cz/`

[7] Luo, A.-L.; Zhao, Y.-H.; Zhao, G.; et al. The first data release (DR1) of the LAMOST regular survey. *Research in Astronomy and Astrophysics*, volume 15, no. 8, 2015: p. 1095.

[8] National Astronomical Observatories Chinese Academy of Sciences. Large Sky Area Multi-Object Fiber Spectroscopic Telescope Data Release 3. [Cited 2016-5-4]. Available from: `http://dr3.lamost.org/`

[9] RUSSO, V. *STATE-OF-THE-ART CLUSTERING TECHNIQUES. Support Vector Methods and Minimum Bregman Information Principle.* Master's thesis, UNIVERSITÀ DEGLI STUDI DI NAPOLI, 2006/2007.

[10] Zadeh, L. A. Information and control. *Fuzzy sets*, volume 8, no. 3, 1965: pp. 338–353.

[11] Banerjee, A.; Dhillon, I.; Ghosh, J.; et al. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2004, pp. 509–514.

[12] Estivill-Castro, V. Why so many clustering algorithms: a position paper. *ACM SIGKDD explorations newsletter*, volume 4, no. 1, 2002: pp. 65–75.

[13] Coates, A.; Ng, A. Y. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*, Springer, 2012, pp. 561–580.

[14] Matteucci, M. A Tutorial on Clustering Algorithms. K-Means Clustering. Available from: `http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html`

[15] Sculley, D. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, ACM, 2010, pp. 1177–1178.

[16] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, volume 12, 2011: pp. 2825–2830.

[17] Ester, M.; Kriegel, H.-P.; Sander, J.; et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, 1996, pp. 226–231.

[18] Campello, R. J.; Moulavi, D.; Sander, J. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining*, Springer, 2013, pp. 160–172.

[19] Vlasblom, J.; Wodak, S. J. Markov clustering versus affinity propagation for the partitioning of protein interaction graphs. *BMC bioinformatics*, volume 10, no. 1, 2009: p. 99.

[20] Cheng, Y.; Church, G. M. Biclustering of expression data. In *Ismb*, volume 8, 2000, pp. 93–103.

[21] Matteucci, M. A Tutorial on Clustering Algorithms. Hierarchical Clustering Algorithms. Available from: `http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/hierarchical.html`

[22] Zhang, T.; Ramakrishnan, R.; Livny, M. BIRCH: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, ACM, 1996, pp. 103–114.

[23] Pudil, P.; Novovičová, J. Novel Methods for Feature Subset Selection with Respect to Problem Knowledge. 1998, inst. of Inf. Theory and Autom., Acad. of Sci., Prague, Czech Republic.

[24] VanderPlas, J.; Connolly, A. Reducing the dimensionality of data: Locally linear embedding of sloan galaxy spectra. *The Astronomical Journal*, volume 138, no. 5, 2009: p. 1365.

[25] Roweis, S. T.; Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, volume 290, no. 5500, 2000: pp. 2323–2326.

[26] Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, volume 9, no. 2579-2605, 2008: p. 85.

[27] Belkin, M.; Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, volume 15, no. 6, 2003: pp. 1373–1396.

[28] Hodge, V. J.; Austin, J. A Survey of Outlier Detection Methodologies. 2004.

[29] Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; et al. LOF: identifying density-based local outliers. In *ACM sigmod record*, volume 29, ACM, 2000, pp. 93–104.

[30] Campos, G. O.; Zimek, A.; Sander, J.; et al. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 2016: pp. 1–37.

[31] Rezaee, R.; Lelieveldt, B.; Reiber, J. *A New Cluster Validity Index for the Fuzzy c-Mean. Pattern Recognition Letters, 19, 237–246.* (1998).

[32] Rosenberg, A.; Hirschberg, J. V-Measure: A conditional entropy-based external cluster evaluation measure. Department of Computer Science Columbia University New York, NY 10027.

[33] Rijsbergen, C. J. V. Information Retrieval, 2nd edition. 1979, dept. of Computer Science, University of Glasgow.

[34] Buitinck, L.; Louppe, G.; Blondel, M.; et al. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[35] Arthur, D.; Vassilvitskii, S. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[36] Dhillon, I. S. Co-clustering documents and words using Bipartite Spectral Graph Partitioning. 2001.

[37] CESNET, z. s. p. o. Project MetaCentrum. [Cited 2016-5-4]. Available from: `https://www.metacentrum.cz/en/about/meta/index.html`

[38] The Apache Software Foundation. Machine Learning Library (MLlib) Guide. [Cited 2016-5-4]. Available from: `http://spark.apache.org/docs/latest/mllib-guide.html`

[39] Palička, A. *Semi-Supervised Learning of Millions of Astronomical Spectra.* Master's thesis, Czech Technical University in Prague, Faculty of Information Technology, 2016.

[40] Škoda, P.; Draper, P. W.; Neves, M. C.; et al. Spectroscopic analysis in the virtual observatory environment with SPLAT-VO. *Astronomy and Computing*, volume 7, 2014: pp. 108–120.

[41] Bukvić, S.; Spasojević, D.; Žigman, V. Advanced fit technique for astrophysical spectra-Approach insensitive to a large fraction of outliers. *Astronomy & Astrophysics*, volume 477, no. 3, 2008: pp. 967–977.

[42] Kluger, Y.; Basri, R.; Chang, J. T.; et al. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, volume 13, no. 4, 2003: pp. 703–716.

Appendix A

Acronyms

LAMOST Large Sky Area Multi-Object Fiber Spectroscopic Telescope

CCD Charge-Coupled Device

FITS Flexible Image Transport System

ASCII American standard code for information interchange

DR1 Data release one

DBSCAN Density-Based Spatial Clustering of Applications with Noise

BSAS Basic Sequential Algorithmic Scheme

Birch Balanced Iterative Reducing and Clustering using Hierarchies

CFT Characteristic Feature Tree

PCA Principal Component Analysis

LLE Locally Linear Embedding

t-SNE t-distributed Stochastic Neighbor Embedding

MDS Multidimensional scaling

LOF Local outlier factor

OPTICS Ordering points to identify the clustering structure

NMI Normalized Mutual Information

RI Rand Index

HDFS Hadoop Distributed File System

API Application programming interface

GMM Gaussian Mixture Model

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `src` ..... the directory of source codes
  - `vocloud_unsupervised` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format