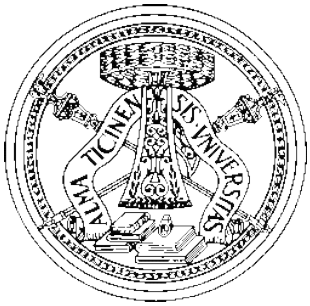

University of Pavia
Facoltà di Ingegneria

Master Degree in Computer Science and Multimedia

Towards a new meaning of modern basketball players positions

Supervisors:
Eng. Tullio Facchinetti

Candidate:
Federico Bianchi

Academic Year 2015/2016

"Analytics are like a bikini, they show a lot, but they don't show everything."

Bob Myers, Golden State Warriors general manager.

Acknowledgments

First of all I would like to thank my family, my parents in particular, who supported me with their love during my whole course of study, always letting me free to chose the best path for my career, but giving me the most precious advices, I always kept in my mind.

A special thank goes to my supervisor too, professor Tullio Facchinetti, who gave me a great opportunity to make a work of thesis which combines my biggest passion, basketball, with my working activity. He helped me a lot during the whole experience, being always present, supporting and motivating me to reach the final goal.

Along with professor Facchinetti I have to thank the people of MYAgonism, the Pavia startup I collaborated with for this work of thesis, in particular Paolo Raineri and Lorenzo Lanzieri, who have influenced me with their enthusiasm, and showed me how your biggest passion can become your job, if you really desire it. They created a great reality in MYAgonism and I will always wish the best for them in their future.

I cannot avoid to mention Leonardo, Valerio, Giacomo, Alessio and Riccardo, the colleagues I shared this five wonderful years of university with. We have been together for the whole experience and I cannot even imagine how difficult university could be without the support of great friends like you.

Lots of other friends, outside the university environment, made these five years an unforgettable period of my life. First of all I cannot forget to say the biggest thank to Cini and Seba, my two best friends, who are like brothers for me, and Susanna, my cousin, who have been the three people I knew I could always count on, in the good and in the bad moments.

Lots of other friends made this five years special, from Riki, Marcy, Steiner, Irene, Barni, Francesca, and Giulia, the people I always love to spend my free time with and the group of friends I shared my Pavia Wednesday nights with, in particular Alberto and Giuseppe.

An honorable mention goes to the staff of the "Bouvette dell'ingegnere", the university bar, Mara, Francesco and Riccardo, who helped me a lot in following all the lessons and passing my exams with the hundreds of good coffees I had thanks to them.

For sure I forgot to mention someone, but I know all the people I met in these five years had a special contribute in this experience, making my university years unforgettable.

Contents

List of Figures

List of Tables

Acronyms

PPG Average Points Per Game

MPG Average Minutes Played Per Game

TDA Topological Data Analysis

SOM Self Organizing Map

BMU Best Matching Unit

MDS Multidimensional Scaling

PCA Principal Component Analysis

MIT Massachusetts Institute of Technology

NBA National Basketball Association

CSV Comma Separated Values

TRB Average Total Rebounds per Game

BLK Average Blocks per Game

AST Average Assists per Game

STL Average Steals per Game

TOV Average Turnovers per Game

PF Average Personal Fouls per Game

TRB Average Total Rebounds per Game

ORB Average Offensive Rebounds per Game

DRB Average Defensive Rebounds per Game

3PP Average 3-point Shooting Percentage

2PP Average 2-point Shooting Percentage

FTP Average Free-Throw Shooting Percentage

OER Offensive Efficiency Rating

AAS All-Around All Star

SB Scoring Backcourt

SR Scoring Rebounder

PP Paint Protector

RP Role Player

OSR Outside Scoring Rebounder

ISR Inside Scoring Rebounder

OB Offensive Backcourt

ACK Acknowledge

ARQ Automatic Repeat reQuest

BD Bus Driver

BG Bus Guardian

BPLC Broadband Power Line Communication

BST Bosch Siemens Temic

CAN Controller Area Network

CC Communication Controller

CKP Clock Polarity

CKE Clock Edge

D2B Domestic Digital bus

DES Data Encryption Standard

DLL Data Link Layer

DSI Distributed System Interface

ECC Error Correction Code

ECU Electronic Control Unit

EMI Electromagnetic Interference

EMC Electromagnetic Compatibility

EVB Evaluation Board

FEC Forward Error Correction

FSK Frequency shift keying

GPIO General Purpose Input Output

IDE Integrated Development Environment

IE Inter Equipment

LIN Local Interconnect Network

LLC Logical Link Control

MAC Medium Access Control

MDI Medium Dependent Interface

MI Motorola Interconnect

MIC Message Identification Character

MML Mobile Multimedia Link

MOST Media Oriented System Transport

OBDII On Board Diagnostic II

ODFM Orthogonal Frequency Division Multiplexing

PL Physical Layer

PLC PowerLine Communication

PLS Physical Signaling

PMA Physical Medium Attachment

RISC Reduced Istruction Set Computer

RTD Round Trip Delay

RTOS Real-Time Operating System

SPI Serial Peripheral Interface

TDMA Time Division Multiple Access

UART Universal Asynchronous Receiver Transmitter

Chapter 1

Introduction

Basketball is a very complex sport and can be analyzed under many points of view. Nowadays advanced analytic techniques are applied in the study of any sport, but basketball is probably one of the most important in terms of statistical analysis, since a huge amount of data can be collected and analyzed from any basketball game.

Starting from the basic statistics anyone can find in any game box-score, like the number of scored points, grabbed rebounds or blocked shots, many more interesting data and information can be derived using advanced analytics.

Most of the professional teams around the world have some people in their coaching staff, whose job is to analyze statistics and data related to the team. Statistical analysis can bring a big value to the team and the coaching staff, since it can help coaches better understand the characteristics of their team and players, in order to exploit them and maximize performances.

Many teams in the NBA, the best league in the world, have many people in their coaching staff working on statistical analysis, in some cases even more then ten people. In recent years many elite teams of the league paid big attention to advanced analytics, not just in order to make a correct game plan according to the opposite team they are facing in the next game, but also using their advanced statistical data to plan their activities through the entire season, or even across different seasons. For example data collected from other teams can help in the selection of the correct players to be hired during the off-season in order to reinforce the team. Having a big amount of data helps coaches select properly the players that can fit best their team or game philosophy.

Advanced analytics are a very important part of the game nowadays, so having a proper team of analysts and the instruments to perform the analysis in the best possible way can make the difference between a winning season and a losing one.

Anyway, in order to perform advanced statistical analysis, sophisticated instruments and skilled people in the coaching staff are needed, but having both of them can be expensive for teams. Many non professional teams, or the ones that play in minor leagues may not be in the position to afford statistical analysis at high level, because of their low budget.

At the base of this thesis work there is a collaboration with MYAgonism, a Pavia born startup that works for providing advance analytic instruments to any basketball

coach in the world at low cost. The idea is to give to any coach or coaching staff the possibility to perform any statistical analysis they want, without the need of expensive equipment, and with the possibility to do it all by themselves, with the system performing the biggest part of the analysis in an automatic way.

MYAgonism is based on a web platform and a mobile application, that allow coaches to keep track and collect any kind of statistics about their team, analyze and use them in order to get the highest possible value from them, and improve the team performances thanks to those information.

1.1 The Problem of Positions

Advanced analytics can bring coaching staffs face many different problems, related with the performances of the team and of the single players. Problems can be related to any aspect of the game and of the coaching activity, from programming the correct training sessions in the different parts of the season, deciding the best strategy to be used against the next opponent team, or finding a way to improve some aspect of the team game.

Sometimes the problem to face can be not so deep and complex, but just an everyday activity, like selecting the best five players to put on the court together in order to perform best. Selecting the proper five players formation to put on the floor in any moment of the game can actually be a complex problem for a coach, since the decision can be influenced by many factors.

Generally, in basketball the five players on the court are selected according to their positions, so the coach select one player for each position to be covered on the court. Since Mr. James Naismith invented basketball in 1891, the same five positions have been used to define the role of players composing a team:

- Point Guard or Playmaker
- Shooting Guard
- Small Forward
- Power Forward
- Center

These five positions are not just labels assigned to players in order to decide the five ones that will start the game or will be on the court together, but they have always been associated to specific characteristics of the player playing that position. The Point Guard for example is considered to be usually the smallest player of the team, the one who has the responsibility to start the offensive possession with the ball in his hands, trying to create the best possible solution for the team to score. The Point Guard is supposed to be in some way the "brain" of the team, the one that "creates the game". On the opposite side there is the Center, which is usually the biggest player in the team, the one who grabs a lot of rebounds, protects the rim on the defensive end of the floor, blocking shots and takes advantage of his physical characteristics on the offensive end of the floor.

One of the most interesting problems regarding basketball analysis that raised during the last few years is the definition of new positions for basketball players. Basketball has changed a lot since its first ages, and nowadays it is much different from the game Mr. Naismith invented for his students. One of the most important changes in the game was the introduction of the three point line in the early 80's, an event which deeply changed the way of playing the game all over the world. Having a new kind of shot that brings one extra point if made, obviously made the idea of shooting from longer distances more interesting. This event changed the game a lot, raising the importance of outside shooting skills and giving more scoring responsibilities to players who played further from the basket, changing the way of playing the "outside" positions of the game.

The evolution of the game brought to a consequent evolution of the players, who adapted to it, becoming more athletic, more skilled and developing many different ways of playing the game. Due to this evolution, it can be difficult in some cases to assign a player to one of the five positions that have always been used in the history of basketball. It is easy to find players who can be considered to be halfway between two of the classical positions, or different players that are labeled to be playing the same position, but are actually playing the game in very different ways.

1.2 Goals of the project

The main focus of this thesis activity is trying to find a new way to define positions of basketball players. Players have always been assigned to the five classic positions according to subjective parameters or some prejudices related to their physical characteristics, but nowadays this kind of approach cannot be satisfying for many players around the world.

The goal of this work is not just to find a new way to assign a player to a predefined position, but to create new positions that will fit the player according to the way he plays the game, trying to cover all the possible ways the game is played, in order not to have ambiguity in the definition of a player's position.

The five classical positions have been totally abandoned, and the new ones are created on the base of a totally analytic and objective method, starting from the statistical profile of players. The idea is to start from the statistics that a player gathered during a whole season, in order to analyze his statistical profile and define his position according to it. Starting from the analysis of pure numbers, any subjective influence is removed and there is the assurance of defining a position that fits the way of playing of the analyzed player. Players with similar statistical profiles will fit in the same new position, while players with different statistical profiles will be separated in different positions.

At the end of the process it should be possible to recognize some groups of players with a similar statistical profile, in order to define a new position only on the base of the result of an automatic clustering, without any a-priori consideration. Refusing a-priori considerations removes any limit on the possible number of positions that will be defined, which could be more than five, according to the number of different statistical profiles that will be found.

The final goal of the work is to be able to find new positions that are faithfully and objectively describing the way a player plays the game, in order to give coaches

the possibility to immediately recognize the player who can give them what they need, when they need it.

1.3 Organization of the thesis

The following chapters of the thesis are organized as follows:

Chapter two will deal with the state of the art, with references to some existent works about definition of positions in sports. A particular focus will be done on the research from Muthu Alagappan, who tried to define new positions for basketball and was the main inspiration for the idea between this thesis.

Part of chapter two will also be dedicated to the presentation of the technologies that have been used during the thesis and some of their applications outside this work.

Chapter three will deal with a more detailed explanation of the technologies that have been used during the thesis under their functional point of view.

Self Organizing Maps, the main technology over which the entire work is based, will be explained in detail, followed by the Multi-Dimensional Scaling and Principal Component Analysis algorithms, used in the data visualization process.

In chapter four all the implemented solutions will be explained.

From the explanation of the programming language and all the external libraries used for the development, the chapter will go into a detailed analysis of the different modules developed to create the complete final architecture.

Chapter five will be dedicated to the presentation of all results.

Firstly the structure of the datasets used for the analysis will be explained, dealing also with the way they have been constructed.

In the second part of chapter five the results of all the experiments will be presented and analyzed in detail

Chapter six will be the final chapter of the thesis, dealing with some general conclusion on the work done and suggestion on possible future developments and applications.

Chapter 2

State of Art

In basketball, such as in many other sports, player's positions are very important and used to define the way players play the game. Other team sports that use positions to define roles of players in the team include soccer, American football, volleyball, hockey, rugby and many others.

In this chapter some works related to the relevance of positions in team sports will be analyzed, showing how the study of positions and of their evolution can help people interpret and understand the game, independently from the sport that is being considered.

Before starting with the explanation of the different case studies, the problem of the approach to sport analysis is discussed, since nowadays, even if sport analysis is becoming more and more important in many disciplines, a scientific approach to the problem seems to be still missing.

2.1 Towards a Scientific Approach

Statistical analysis is becoming more and more important and widespread in many sports, not only in basketball. Thanks to the technological development, fans and people who work in sport organizations are able to access huge amounts of statistics and data about any competition they want to follow, observe and study.

This possibility raises the interest of fans all over the world regarding statistics and data analysis about their favorite sport and players. Moreover, statistical data are used many times in TV programs or shows by the media to compare two teams or to comment the result of a specific game. For this reason, people is becoming rather familiar with these statistics and numbers.

A direct consequence of this spreading of statistics and numbers is the availability of many studies based on these data, which often bring to very interesting results, sometimes greatly helping teams and coaching staffs in their job.

Sport journalists and fans all over the world are continuously writing hundreds of articles and insights on dedicated magazines, websites or blogs about any aspect of sport analysis, from the more empirical one, to the more technical. Although many of these works are very detailed and well explained, just a few of them face the argument following a scientific method, and so scientific publications about sports analytics are limited to those fields that are related to other scientific disciplines.

For instance, many scientific articles related to sport deal with the evaluation of physical performance of athletes or the analysis of injuries and recovery from them. These works are mainly published on medicine journals.

On the other hand, studies on technical aspects of the game are usually developed in a non-scientific way. Actually, this is due to the fact that studies about the game itself can be more useful to people working within sport organizations or media organizations, who usually do not feel the need of a scientific approach to be followed in order to exploit the results of a specific study. In many cases, there is no scientific background to properly handle and exploit the results of these studies.

Studies on injuries or physical performance of players are instead mainly useful to people with some scientific background, and so they need to be carried out by following a scientific approach. Although these articles tackle the problems from the biological, and sometimes medical, viewpoint, relationships to more technical aspects of the game can be found to be useful. For example, some studies put into relationship the distributions of injuries with different player positions [**?**, **?**].

The idea of distinguishing players analysis according to their positions can be also applied in the case of physical performance evaluation. In a 2015 article from Haris Pojskic, Vlatko Separovic, Edin Uzicanin, Melika Muratovic and Samir Mackovic [**?**], evaluations on aerobic and anaerobic power consumptions are done on groups of basketball players playing difference positions. Some guards, forwards and centers are studied, making them perform the same 3 tests. Results are compared to determine whether there are differences in the power consumption of players from different positions. Actually, the results show that guards and forwards have better performance in the aerobic tests with respect to centers, while centers produce better performance in anaerobic tests.

In some few cases, scientific papers regarding empirical sport analysis can be found. Jennifer H. Fewell, Dieter Armbruster, John Ingraham, Alexander Petersen and James S. Waters, students from the Arizona State University [**?**] compare a basketball team to a strategic network, where players are represented as nodes and ball movements are represented as links between nodes. Computing some particular metrics about the network, it was possible to distinguish between some typical offensive strategies, based on different patterns of ball movements.

An interesting article about basketball advanced analytics has been published by Scott Bruce, a student from the Temple University of Philadelphia, in 2016 [**?**]. In this paper, Bruce uses the data coming from the NBA player tracking database to create a scalable framework that allows to efficiently compare players performance. The database provides information about positions and movements of players on the field, positions where actions have been made and data related to physical performance of players.

The main tool used to analyze these data is Principal Component Analysis (PCA), a technique that has also been used in the present work and will be explained in detail in Section 3.3. Thanks to the PCA, Scott found 4 principal components in the statistical data distribution that were analyzed from the NBA tracking database, and associated to each principal component a different meaning. For example principal component 1 was called "Inside vs. Outside", since its value showed how a player

is used to play closer or farther to the basket. Players who had a positive score in principal component 1 are the ones who play closer to the basket, while players with a negative score are used to play farther from it. Other principal components were related to other opposite aspects of the game[1]: the attitude to "assist and drive" is opposed to "catch and shoot" in principal component 2, "scoring and rebounding efficiency" is opposed to "speed" in principal component 3, "catch and pass/shoot" is opposed to "slash" in principal component 4.

Analyzing the combination of the principal components, or just the components alone, it is possible to find a direct way to compare players, based on some particular aspects of their game. For example, their attitude to shoot the ball more than to pass it, or the attitude to catch and shoot plays, more than driving to the basket.

Moreover, by aggregating data about the principal components of players, it is possible to apply the same analysis to teams, showing distribution of teams according to their principal components and, for example, their winning percentage. By analyzing the values of principal components related to winning teams, it is possible to assess the aspects of the game that can better influence the performance of the entire team.

As a conclusion, it can be noted that, even if many scientific researches regarding sport have been developed through the years, most of them are studying sport activities under a point of view that can be related to other fields of scientific research. Despite the increasing interest of people to sport analytics, a few studies are involved in applying a scientific method to the study of the game itself, with the only purpose of providing scientific-relevant tools to the sport community, to better understand the characteristics of sports and games or their evolution in time.

2.2 The Relevance of Positions

Most of the team sports are characterized by positions to which the players in a team are assigned, in order to attribute specific responsibilities during a game. In most cases, positions have been defined with the appearance of the sport they belong to. And many sports were born a long time ago. In Section 1.1 it was explained how basketball evolved and changed a lot through the time, becoming a very different sport with respect to its original conception. The same thing can be said about many other team sports. There are a lot of examples in which positions that were defined at the origin of a particular game, are now not so useful to describe the way players play the game, or at least players are changing the way of interpreting the meaning of their position.

Most of the time, it is difficult to find a complete change (we will use the term "revolution" for this change) of all the positions in a specific game, but there are many examples of great players in history of sports, who really changed the way of interpreting a position, or even created a new one because of the way they played the game.

[1]Speaking about "opposite" aspects of the game, Bruce refers to some couples of play types which usually are not both present in the way of playing of one single player. For example, "assist and drive" is opposed to "catch and shoot" because usually players who have attitudes in assisting team mates and driving to the basket do not have a good attitude in taking shots from plays made by teammates.

A first example comes from American Football, a sport in which positions are very well defined and each player has very different responsibilities according to his position.

The most well-known position in football is the Quarterback, the offensive player who receives the ball at the beginning of a play and has to pass it to his teammates to gain yards towards the goal. In the history of football, the most famous and successful quarterbacks were players with very good game vision and passing skills, but they were not so good athletes in terms of physical performance. Usually, the best athletes in the team played other position, and the role of the quarterback was quite static, with the "mind of the team" who had to be well protected by his offensive line in order to allow him to throw the ball to the best possible receiver.

In recent years, in the NFL, the USA National Football League, the best American football league in the world, a new category of quarterback started to shine. Players of this new category are elite athletes, but with excellent passing skills and game vision. The combination of athleticism and passing skills allows them to play outside the area that is protected by the offensive line, and to rush the ball in the case no suitable receiver is found for passing the ball.

In [**?**], Sam Monson tries to explain how "Mobile Quarterbacks" are redefining the position and the way of playing it. Monson highlights the fact that having a very athletic quarterback who can rush the ball besides passing it to receivers, creates a big threat for defenses, which have to take care of the passing game, but also of a possible running game from the quarterback himself. This is an option that was not available before the rise of this new category of athletes.

Two of the first examples of mobile quarterbacks were Michael Vick and Daunte Culpepper, who were very good quarterbacks in early 2000's. However, according to Monson, they never reached the elite of the league due to different reasons, and thus they were unable to actually re-define the way of playing the quarterback position. Nowadays, new young talented players arrived in the league as mobile quarterbacks, in particular Cam Newton of the Carolina Panthers, and Robert Griffin III of the Washington Redskins. Monson highlights that both of them are explosive athletes with very good passing skills, and he was wandering if they would be able to redefine the quarterback position, thus completing the "mission" that was failed by previous players. The current situation is that Robert Griffin III suffered an important knee injury that left him out of the field very much time. In the 2016-2017 season he will try to get back from the injury with his new team, the Cleveland Browns. Cam Newton instead has been playing great seasons with the Carolina Panthers, until the last one, in which he reached Super Bowl 50, the great final of the NFL, losing it against the Denver Broncos. Maybe the re-definition of the role is still not complete, but something is definitely changing in the American football.

A similar example of player re-defining a position comes from European football, or soccer. In this case the position under the spotlight is the goalkeeper, while the interested player is Manuel Neuer, German goalkeeper of FC Bayern Munich. In his 2016 insight, published on ESPN [**?**], Andrew Corsello tries to explain how Manuel Neuer can be the player re-defining the goalkeeper position. Neuer has been considered to be one of the best goalkeeper in the world for years, with his great athleticism and technical qualities, that brought him to be the starting goalkeeper of an elite European club and of the German national team, with which he won the

FIFA World Cup in Brazil in 2014.

What makes this player unique and different from all the other goalkeepers is its ability to play far away from the goal. He is worldwide known because of its plays far from his goal line. A typical situation where his unique qualities are highlighted is when there is a long pass from the opposite defense to launch a striker in the open field. In this kind of situations, the goalkeepers usually stay close to their goal line, hoping for an help from the defenders, getting ready for a one-on-one with the striker when he arrives rather close to the goal.

The Neuer approach is different. He usually starts running towards the ball, with the intention to reach it before the striker, even 30 or 40 meters far from his goal line. Afterwards, he throws the ball out and stop the play. This very peculiar (and somehow innovative and risky) actions made him very famous, but also changed the way of thinking at the position of the goalkeeper, that in his case, as Corsello said, can be renamed as "boxkeeper".

In his insight Corsello asked Neuer how can he play the position in this way. Neuer's answer to the question was just a single German word: "Nervenstarke", whose meaning is approximately "mind toughness", a characteristic that for Neuer is the key factor to be a perfect next-generation goalkeeper, as he is.

Football gives another example of a player who deeply changed the game, not re-defining an existing position, but even creating a new one: *Claude Makelele*.

In his article from 2015 [**?**], Zain Mahmood, explains how "The Makelele position" changed the English football forever, since he joined Chelsea FC in the summer of 2003. Makelele is a French midfielder, who has never been famous for his spectacular game or his technical skills. He made his career in football thanks to his athleticism, his spirit of sacrifice and his defensive attitude.

After the early years of his career in France, he joined Real Madrid in 2000, to play for the so called "Galacticos", a team full of sublime offensive players like Ronaldo, Raul, Figo and Zidane. Makelele had a very important role in such an offensive team, giving balance to it and being a defensive coordinator from the midfield.

After some successful seasons with Real Madrid, he was traded to the Chelsea FC, in London, where "The Makelele position" was born. Chelsea manager at that time was Jose Mourinho, the Portuguese manager who won the Champions League with Porto FC in the 2003-2004 season. Mourinho decided to change the module used by Chelsea, passing from the classic 4-4-2 module used by most of English teams, to a 4-3-3 module that was totally unused in the English Premier League.

Makelele was playing in the central position of the 3 midfield players, where he could continue playing his efficient defensive game with no offensive responsibilities. The 4-3-3 module puts 3 players in center midfield, against the 2 put by the 4-4-2 module, in which 2 of the 4 midfield players play on wings, and are usually devoted to the offensive part of the game. Facing a central midfield composed by 3 players instead of 2 was a massive problem for Chelsea opponents that year, and having an extra man in central midfield allowed Chelsea to dominate games for 90 minutes.

The Makelele position was officially born, a new type of midfielder, who has defensive responsibilities and can be considered a sort of "anti-footballer", as Mahmood says in his article. Someone whose primary instinct is to destroy opponents attack before creating some for his team.

Makelele had a huge role bringing Chelsea to win the Premier League title for two consecutive seasons, and he is one of the first examples in which the "dirty work" of a defensive midfielder has been recognized to be fundamental for the results of a great team.

The Makelele position has become fundamental and permanent in the recent history of football. In the 2015-2016 season, Leicester City FC won the Premier League, in one of the most intriguing "Cinderella story" in the history of sport. A key player of that team was a French defensive midfielder, N'Golo Kante, who played the Makelele position.

The last example of outstanding player in conflict with traditional positions in its sport, basketball. The player under the spotlight is *Draymond Green* of the Golden State Warriors. In an article published in February 2016 on SBNation [**?**], Paul Flannery defines Green as the player who is redefining the concept of NBA superstar, being totally different from all the other all star players that had a great impact on the league in the past.

Draymond Green was a college superstar at Michigan State University, where he was one of the best player in the nation, playing the power forward position during 4 successful seasons under coach Tom Izzo.

When he finally passed to professional sport, declaring for the NBA Draft, a problem was in the mind of every general manager of NBA teams: Green seemed to be too undersized to play the power forward position at NBA level, and too slow to play as a small forward or as a guard, he just seemed to be similar to many other undersized collegiate forward players who could not have any space in a league like the NBA.

He was selected by the Warriors with the 35th overall pick in the draft, at the beginning of the second round, where Warriors general manager Bob Myers decided to take a risk on him. In Flannery's article, Myers words about the decision of picking Draymond Green are reported: "We saw a guy that could help us win basketball games, Didn't fit the perfect profile of really any position, but he had succeeded at a really high level in college. Everyone we spoke to talked about his maturity, his professionalism, his basketball IQ, so we didn't try to overthink it." Actually even Myers had no idea of what could happen in the future of Draymond Green and, as he said, no one had idea of which position he could play in the NBA.

To find a way to introduce him in the team, Warriors head coach Steve Kerr asked himself a simple question: "What does he do really well?", trying to find some aspect of Green's game that could really help the team. The answer that Kerr gave to his own question was that Green does everything really well, so he could help the team in any aspect of the game. Green is an hybrid frontcourt[2] player, who can defend against opponents playing different positions, can make long range shots on the offensive side of the floor, and can pass the ball very well. To try to give a definition to what kind of player Draymond Green is, general manager Bob Myers said "In some ways he's pioneered the concept of a playmaking four", highlighting how Green's game can be compared to any of the classic basketball positions in some ways. He can protect the rim on defense like a center, score close to the basket and rebound like a power forward, defend on any opponent like some small forwards,

[2] A frontcourt player in basketball is the one who usually plays the power forward or center position, usually is one of the biggest players in the team.

make long range shots like a shooting guard and pass the ball like a point guard. "What position does he play was the big thing," Myers said. "He's taken that and turned it on his head and now the model is position-less basketball." As Warriors general manager said, what was a problem at the time of the draft selection has then turned into the most important strength of Draymond Green game, making him a unique player in the league.

In four years of professional career Green became a starter and a key player of the Golden State Warriors, he won the NBA championship in 2014-2015 season, and the team set the record for most wins in a single regular season in the 2015-2016 season, reaching the NBA Finals for the second consecutive year. Green was a very important part of this success, becoming an all-star player on both ends of the floor, the offensive and the defensive one, and creating the next prototype of NBA superstar, the most versatile player possible, who can do everything on the basketball court at the top level.

All of the examples that have been analyzed in this section regard some outstanding players, who have been able to change the way people think at how a particular position in a particular game is played. None of these example, anyway, tries to study a complete revolution or redefinition of all the positions in a game, which is the final goal of this thesis. There is a big difference between having single examples of players who play a position differently from the others and redefining all the positions, since the particular way a player plays a position may be relevant only for that player and not for any other player, while redefining all the positions in a game means trying to completely change the way people should think about the game. Actually, one attempt to the total redefinition of positions in basketball exists, and it is the main source of inspiration for this work of thesis.

2.3 Redefining basketball positions

Muthu Alagappan was born in England, raised in Houston, Texas and has always been a big fan of the Houston Rockets, the Houston's NBA team. He has been a student at Stanford University and after his graduation he started an internship with Ayasdi, a big data startup in Palo Alto, California. Here at Ayasdi he had the possibility to apply what he learned about big data and the tools of the startup to develop his new idea about advanced basketball analytics.

His idea was to apply Topological Data Analysis (TDA) on NBA players statistics in order to show differences between players statistical profile and to redefine basketball positions according to the results coming from TDA. TDA is a mathematical technique, mainly used to analyze high-dimensional datasets, putting the focus on the shape of data on the different dimensions, and showing differences between data samples through shape analysis. TDA allows to better understand these high-dimensional datasets analyzing them in a manner that is not influenced by some specific metrics, providing dimensional reduction, with no loss of information.

He presented his work at the 2012 MIT Sloan Sport Analytics Conference [**?**], the most important event in the world focused on sport analytics. His presentation was a great success, making him win the top prize in the "Evolution of Sport" category.

During his presentation, Alagappan explained how the idea behind his work came from his personal experience of non-professional basketball player, and basketball

fan. He explained how everytime he was going to play basketball with friends, people always made him play the point guard position, because of his size, since usually he was the shortest player on the court. He was actually a bad point guard, as himself declared, but he always thought to be a better player than he seemed to be, he was just a bad point guard, but not a bad player at all.

As explained in Section 1.1, basketball changed a lot as a game since its first ages, and classic positions are not anymore suitable for describing the way players play the game. In his talk, Alagappan focused on 2 main problems of the 5 positions of basketball everybody knows. The first problem is "oversimplification": any basketball expert, but also fans, can see how there are hundreds of different ways of playing basketball around the world, so why should we use just 5 positions to express the different ways the game is played?

The second problem is "incorrect classification": how can we objectively determine which position a player should play? Alagappan explained how usually players are labeled to be playing a certain position only because of some physical prejudice or some subjective factors, so he wanted to find an objective and analytical way to define new positions in basketball.

He decided to apply TDA to players statistics to find differences between players and the way they play the game only on the base of statistical data and of the differences between their "numbers". He learned and experienced how to apply TDA during his internship experience at Ayasdi, following the methods that the Palo Alto startup developed, as explained in the 2012 article[**?**] he contributed write. TDA allows to create the so called similarity networks, particular networks in which nodes are called "data points" and represent some elements of the dataset under consideration, while edges connecting nodes are created to be able to show the similarity relationships between nodes. Similar nodes will be displayed in close positions on the network, while nodes that are very different between each other are displayed far from one another on the network. Similarity networks are a very versatile tool, which can be used in many fields of study to show relationships between elements in any input space. An example of application of this kind of networks is the classification and visualization of relationship between protein families, as reported in a 2008 article[**?**].

Alagappan created a similarity network of NBA players, representing each player with his season average statistics, considering statistics from the 2010-2011 NBA regular season. Each node in the network represented one or more NBA players, and players were mapped on the network according to how similar or different their statistics are. He used 7 statistical category to define each representation of one of the 452 analyzed players: points, rebounds, assists, steals, turnovers, fouls and blocks, normalizing these statistics according to the Average Minutes Played Per Game (MPG) as explained in Section 5.2.1, to be able to compare players who played a different amount of MPG.

He ran the network expecting to find five main clusters describing the five positions of basketball, but what he actually found was much more than what he expected. The network provided him 13 clusters, corresponding to the 13 new positions of basketball he unveiled at the MIT Sloan Sport Analytics Conference:

1. Offensive Ball-Handler: A player who has very good ball-handling skills and is

specialized in scoring points, making free throws and taking many shots in a game. An Offensive Ball-Handler is below average in steals and blocks.

2. Defensive Ball-Handler: A defensive-minded player who is specialized in steals and assist, while he is just average in scoring points, making free throws or taking shots.

3. Combo Ball-Handler: A player who is average both on the defensive end of the floor and on the offensive one. These kind of players are good at doing all things, but are not shining in any particular aspect of the game.

4. Shooting Ball-Handler: A player who has a particular attitude for scoring. He is above average in field goal attempts and Average Points Per Game (PPG)

5. Role-Playing Ball-Handler: A player who plays a few minutes and do not have a big statistical impact on the game.

6. 3-Point Rebounder: A player who is a ball-handler and big man. He is above average in rebounds and three-pointers, compared to ball-handlers.

7. Scoring Rebounder: A player who is above average in rebounds and has particular attitude for scoring.

8. Paint Protector: A player who is above average in rebounds and blocks, but also fouls. A paint protector is also below average in points scored.

9. Scoring Paint Protector: A player who is above average in rebounding and blocking shots, but also in scoring. He has a good attitude on both end of the floor, the offensive and defensive one.

10. NBA 1st Team: A group of players who are simply very good in every statistical category. The similarity network groups them together because they are above average in every statistical category.

11. NBA 2nd Team: A group of players who do not have any particular statistical feature in their profile, they are close to average or a little above it in every statistical category.

12. Role Player: A role player is similar to a player of the NBA 2nd Team category, but with lower than average statistics and usually a few MPG.

13. One-of-a-Kind: Group made by just a few players, that are so good that TDA is not able to connect them to any other category. They are considered to be just outstanding and better than all the other players.

Defining the new basketball position using the similarity network it has been possible to associate every position with specific statistical features of players playing that position, as explained in the list. Moreover, TDA allows to compare players that are clustered in a certain position and analyze which ones are closer to the perfect prototype of player of a certain position, according to the distance between their statistical profile and the ones of the other players in the category. For instance,

Alagappan reports how the statistical profiles of Jason Terry and Tony Parker were the closest ones to the perfect Offensive Ball-Handler player, while Marcus Camby and Tyson Chandler were very good examples of Paint protectors.

Apart from showing the new position obtained by the map and how these positions describe the way players play the game, Alagappan was able to reconstruct on the map the spatial formation of players from specific teams of the 2010-2011 NBA season, showing which of the new positions players of these teams belong to. Studying how teams are built and which kind of players are in the different teams, with respect to the new positions, it could be possible to compare the composition of teams with winning records and teams with losing records, to see if there are some common patterns of players positions that influence the performance of teams. If it is possible to find a particular combination of players that is common to all the teams with best performance and win rate, it would be possible to build teams trying to recreate this particular combination that was very successful and effective for winning teams of the past seasons.

Alagappan stressed the fact having positions that are able to truly describe how a player performs, can help general managers to build more balanced teams and to to determine if a player fits in the team immediately, just looking at his "true position".

This way of interpreting positions can be a big help for coaches too, when they face the problem of building lineups or evaluating offensive and defensive strategies. Having more positions that better describe the way players interpret the game allows coaches to create strategies that can better combine the characteristics of their players, exploiting their strengths at the maximum potential.

With his presentation, Alagappan tried to show how a big change in the way of interpreting basketball is needed, to avoid to remain blocked on conventions that comes from the past. It is an undeniable fact that the changes and the evolution of basketball as a game made 5 positions is not suitable anymore for today's needs of coaches, general managers and even fans.

Alagappan's work was the main inspiration for this thesis, where other techniques and tools different from TDA were applied to redefine basketball positions. The idea is not just to validate Alagappan's results, but to create our own positions and define our own way of interpreting modern basketball, to help people understand the game better and being able to analyze it in the best possible way.

Chapter 3

Technologies

In this chapter the main technologies and techniques used in this work are presented and analyzed in details. The theory behind each method will be explained and some particular and interesting applications will be cited.

SOM is the main technology used in this work. It is used to obtain the clustering of players into different positions, which is the ultimate goal of the work. The input space of basketball players is made by vectors of statistics, each one referring to a single player, which can be made of different number of elements, according to the number of statistical categories used to describe the player.

The result of the SOM algorithm consists in the creation of a 2-D output layer made by neurons, the primary elements composing the SOM network. Each neuron is described by a weight vector which contains a number of elements that is equal to the number of statistical categories used to describe players, as will be explained in Section 3.1.1.

Even though the SOM performs the clustering algorithm on the elements of the input space, the result of the algorithm is still an high-dimensional space, since the 2-D output layer of the SOM is made by elements which are characterized by vectors of weights. Representing the output layer as a multi-dimensional array it may have a shape like $(N \times N \times m)$, where N is the dimension of the 2-D output lattice, while m is the number of weights describing a neuron.

Even if the clustering operation is completed just using the SOM, one main problem after the training of the network becomes the visualization of data. To visualize the clusters, we should be able to visualize the structure of the output layer of the SOM, which after the training process is still an high-dimensional space.

The other two technologies presented in this chapter, MDS and Principal Component Analysis (PCA) have been studied and applied to solve the visualization problem. Both techniques allow to reduce the dimensionality of an high-dimensional input space, allowing data visualization in a lower-dimensional output space. The application of MDS or PCA on the output layer of the trained SOM allows to obtain a low-dimensional output space which can be easily represented to show the effect of the clustering algorithm of the SOM.

3.1 Self Organizing Maps

A SOM is a particular kind of Artificial Neural Network (ANN), also known as Kohonen network from the name of the scientist who firstly proposed it, Teuvo Kohonen [**?**]. SOM are commonly used to represent and analyze high-dimensional datasets into a lower dimensional space, the map itself.

What makes SOM different from other neural networks is that SOM are trained using a particular unsupervised training technique, based on competitive learning, opposed to the error correction learning used by common neural networks, which are trained using supervised learning methods. Supervised learning techniques are based on giving to the neural network a particular set of inputs and the corresponding set of outputs. Giving to the network an adequate number of input-output pairs, it is possible to make the network learn how to generate the correct output, given a certain input. The strength of supervised neural networks is that they can successfully interpolate the input to generate correct output in presence of non-linear input-to-output transfer functions. To apply a supervised learning algorithm, a sufficient knowledge of what is expected as an output from the network, given a particular input, is required.

In this work, the final goal is to define new positions in basketball, which in other words means to define new clusters that can group basketball players in a different way with respect to the original basketball positions.

A very important point is that we do not start from some expected organization of new positions clusters, therefore a supervised approach is not viable. Instead we want a network which is able to determine a reasonable output itself, without any a-priori knowledge of the expected output provided from external sources. With its unsupervised training, SOM is a perfect tool for our purpose.

Unsupervised training based on competitive learning allows the SOM to learn the "shape" of the input space and the topological relationships between the elements in it, that in our case are vectors of floating point numbers representing a player statistics. The SOM is then able to reduce the dimensionality of the representation, while transferring the topological relationships between the elements of the high-dimensional input dataset in the lower dimensional output space.

3.1.1 Structure of a SOM

The standard structure of a SOM is made by 2 layers, a one dimensional input layer, and an output layer. These layers can have arbitrary dimensions, but most of the time a 2-dimensional output layer is used, since it ease the visualization of the input space in the dimensionally reduced output space.

The input layer can be made by an arbitrary number of elements, in the general example of Figure 3.1 it is made by n elements. Each element of the input layer is made by a vector of m elements. In our case each element of the input layer is the vector of statistics referring to a particular player. Therefore, the dimension of each element in the input space depends on the number of statistical categories considered for the categorization.

Elements in the output space are called neurons, recalling the name of the cells composing the human brain. Neural networks indeed were created to resemble the

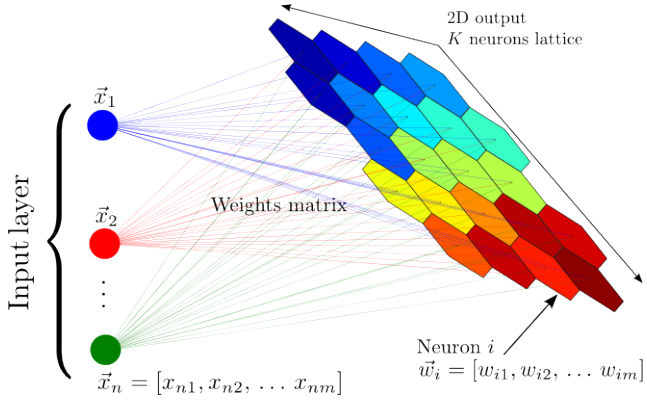

Figure 3.1: Structure of a Self-Organizing Map [**?**].

way the human brain works, with the collaboration of neuron cells through synapses. Elements in the output space of a neural network are called neurons to recall the human brain cells, since they are the "cells" which compose the neural network.

Each neuron is characterized by a vector of weights, whose dimension must be the same of the dimension of each element in the input set. In the general example shown in Figure 3.1, the dimension of neuron weight vectors is m. The output space is usually organized in a 2-dimensional lattice of neurons, that are not connected between each other, but have a fixed position in the lattice. The relative positions of neurons within the lattice is very important to maintain the topology of the input space elements on the output layer. Moreover, the position of neurons in the lattice is used to define the neighborhood relationships between them, which are used in the training process, as will be explained in Section 3.1.2.

The size of the lattice, which determine the number of neurons in it, can be arbitrary selected, according to the needs of the application. Neurons are not connected between each other, but each neuron is connected to all the elements of the input layer, and each connection is characterized by a weight. On the other hand, each weight vector $\vec{w}_i$ related to the $i-th$ neuron is initialized to random values when the map is created, since it will be tuned during the learning process, as will be explained in Section 3.1.2. In fact, values of weight vectors, $[w_{i1}, w_{i2}, ..., w_{im}]$, are gradually changed during the learning process, to obtain new values that are able to describe the topology of the input space, this process is called "tuning" of the parameters. The tuning of weight vectors of neurons makes the map able to maintain the topological features of the input space on the output layer.

3.1.2 Training Process and Learning Algorithm

The training process is very important for the successful behavior of any artificial neural network, and it becomes particularly important in the the case of a SOM. As explained in Section 3.1, a SOM uses a dedicated unsupervised type of training process, based on competitive learning.

Training is an iterative process that, depending on the size of the network, can require significant processing effort and time. Usually the training process takes most of the computational time of any algorithm that exploits SOM as a software module.

During the training process, each single element of the input set is given as an input to all the neurons of the output layer. A comparison between the input element and the weight vector of each neuron is made, to find the neuron whose weight vector has the closest value w.r.t. the input element. This neuron is called Best Matching Unit (BMU). Once the BMU has been found, its neighbors are computed according to a particular metric Afterwards, the weight vectors of the BMU and all its neighbors are tuned. The process is repeated for all the elements of the input space that are used for the training of the network, and the whole process is repeated multiple times. These iterations impact on the time required for the processing.

The learning process can be divided into 6 steps:

1. Initialization of neurons weight vectors. Weight vectors are usually initialized to small random values, that will be tuned during the learning process. Another possibility is to initialize vectors taking samples from the subspace spanned by the two principal component eigenvectors of the input space.

2. Selection of one of the elements of the input set. The selected element will be presented to the network and compared with the weight vector of each neuron.

3. Vector comparison and selection of the BMU. The selected input space element is compared to each neuron weight vector, to find the weight vector that is most similar to it. The metric used to compare input elements to weight vectors is usually the squared Euclidean distance:

$$EuclideanDist^2 = \sum_{i=0}^{m}(I_i - W_i)^2 \tag{3.1}$$

   Equation 3.1 expresses the squared euclidean distance between an element of the input space and a weight vector of a particular neuron of the output layer. The term I_i refers to the i-th element of the selected input vector used to feed the network, while W_i refers to the i-th element of the weight vector of the neuron that is considered for the comparison. The choice of using the squared euclidean distance allows to avoid repeating the square root operation, which is computational and time expensive. Skipping the square root operation does not affect the final result, since the same squared metric is used for all the comparisons. The BMU is the neuron which has the lowest value of squared euclidean distance with respect to the considered input vector.

4. Computation of the neighborhood of the BMU. The initial neighborhood radius is set to be as large as the output layer lattice. This radius is computed again at every iteration, using a specific metric:

$$\sigma(t) = \sigma_0 * e^{\frac{-t}{\lambda}} \tag{3.2}$$

   Equation 3.2 shows how the neighborhood radius at iteration t is computed. σ_0 is the radius of the output layer, while λ is a time constant, computed as the ratio between the total number of iterations of the learning process and the output layer radius, σ_0. Weight vectors of neurons that are within the

neighborhood radius of the BMU are tuned, while neurons that are outside the radius are not considered in the tuning process. As shown in Equation 3.2 the neighborhood radius is computed with an exponential decay formula, so the value of the radius gets smaller as the number of completed iteration increases, so as the learning process gets completed.

5. Tuning of the neighbors weight vectors. Once the neighborhood neurons have been identified using the neighborhood radius function, the tuning of the weight vectors can begin. The weight vectors of the BMU and of each neighbor neuron will be tuned. The weights associated to each neighbor neurons are updated according to their distance from the BMU. The variation of the weights of closer neurons to the BMU is larger. On the opposite, farther neurons from the BMU are less affected by the tuning process. This means that – obviously – the neuron whose weights are most strongly affected by the tuning process is the BMU itself. During the current iteration t, the new values of weights associated to the neurons to be used as starting point during the next iteration, $t+1$, are computed according to the following law:

$$W(t+1) = W(t) + \theta(t) * L(t) * (I(t) - W(t)) \tag{3.3}$$

In Equation 3.3 $W(t+1)$ is the new weight vector assigned to the neuron, while $W(t)$ is the weight vector at the current iteration. $\theta(t)$ expresses the distance of the considered neuron from the BMU; its value is obviously null for the BMU itself. This particular distance is computed considering the squared spatial distance between the considered neuron and the BMU in the grid, but also considering the neighborhood radius at the current iteration:

$$\theta(t) = e^{\frac{-SpatialDist^2}{2*\sigma^2(t)}} \tag{3.4}$$

If we express with (i, j) the coordinates of a generic neuron in the grid, the squared spatial distance between a neuron and the BMU is:

$$SpatialDist^2 = (bmu(i) - neuron(i))^2 + (bmu(j) - neuron(j))^2 \tag{3.5}$$

Thanks to Equation 3.4, different effects of tuning can be applied to neurons that are farther or closer to the BMU within the neighborhood radius.

Another important parameter of Equation 3.3 is $L(t)$, which refers to the learning rate of the training process:

$$L(t) = L_0 * e^{\frac{-t}{\lambda}} \tag{3.6}$$

As shown in Equation 3.6, the learning rate is computed as an exponential decay, so that its value decreases as the number of completed iterations of the learning process increases. Its initial value is set equal to L_0, the initial learning rate, which is usually assigned manually when the SOM is created. The time

constant λ, instead, is computed as the ratio between the total number of iterations of the training process and the radius of the output layer of the map.

Finally the term $(I(t) - W(t))$ in Equation 3.3 represents the difference between the input vector and the considered weight vector at the current iteration.

6. Repeat operations from Point 2 for an adequate number of iterations. It is worth to notice that the number of iterations can not be universally set. The number of iterations needed for a correct training process depends from many factors, like the size of the output layer, the size of the training set or the kind of application that is addressed.

After the completion of the learning process, all the weight vectors on the output layer of the SOM are tuned. At this stage, the map is ready to be used to classify new input elements that were never provided before to the network. The training process creates clusters of neurons in the output layer, in a way that reflects similarities and topological relationships between the elements of the input space.

After the training process, the forward mapping of elements from the input space to the output layer of the map can be performed. The input element that is given to the SOM is mapped in a specific position of the output layer, corresponding to the neuron that, after the training process, turns out to be the most similar to the input vector. This way the SOM is able to map similar elements of the input space in close positions on the output layer, allowing a simple and intuitive 2-D visualization of the clusters of the multi-dimensional input space. Input elements that are close in the input space, and similar between each other, will be mapped in close positions on the output layer of the map. This feature makes SOM a powerful tool for data clustering and classification of multi-dimensional input data.

3.1.3 Applications of the SOM

SOM are a very versatile tool that can be applied to many applications and in many fields. As explained in the previous sections, a SOM can be used in general to enable or simplify the visualization of features and characteristics of a multi-dimensional dataset, highlighting clusters of elements in the dataset that are characterized by similar features, if they are present.

The first application that is worth to mention is the most general one, i.e., the *clustering*. In this work a SOM is used to cluster basketball players into categories according to their statistics, but the SOM can be used to cluster any kind of data according to their features.

In their article [**?**], J. Vesanto and E. Alhoniemi explained how this kind of neural network can be a good instrument to be used in the exploratory phase of any data mining activity. The projection of the input space in the lower-dimensional output space can be used to highlight properties of the data, and so the clusters in which they are organized. The authors outlined that a larger number of neurons in the output layer can help the visualization of clusters and groups of the input space.

In the same paper, a hierarchical agglomerative clustering and partitive clustering using K-means is discussed. The authors analyzed the performance of the two-stage procedure of hierarchical clustering, in which prototypes are created by the SOM in the first stage, while they are clustered in the second one. This approach turned

out to perform better in terms of computation time with respect to direct clustering techniques.

Another interesting application of a SOM is reported in [**?**], where a collaboration of researchers from a German and a Romanian university applies for hand and full body tracking. The idea comes from the increasing spread of gestural interfaces for consumer applications, that are becoming the new frontier of Human-Machine Interaction (HMI).

Some of the first applications using this kind of gesture based interaction were in the field of video games, like Nintendo Wii, or Microsoft Kinect, which showed how this approaches could create very involving and interesting user experiences. Apart from video games, hand-tracking alone can be used in many different applications, and represents a milestone of gesture based HMI.

The advent of new devices for 3-D image acquisition provided a big improvement to movement-tracking applications, since having the possibility to detect hands or body positions in 3-D allows to better recognize the shape of any hand position in the acquired images.

Exploiting all those technologies, researchers tried to use the SOM algorithm to develop a new tracking application, that was able to track hand and even body positions while images were taken from the camera. The idea is that, thanks to a proper training phase, the neurons of a SOM can be clustered in particular positions related to the main feature-point that characterize the hand or body shape.

Applying the SOM algorithm it is possible to see if the clusters are correctly positioned with respect to the shape and position of the hand.

The selection of the configuration of points to be used to recognize the hand or body shape was a very critical problem, but the use of 3-D image acquisition devices allowed to extend the models, adding more features that could better describe positions and shapes, and could be easily recognized by 3-D cameras.

Having hardware components with good performance, like 3-D cameras, allowed the team also to try to apply a new extended kind of SOM and SOM algorithm to improve the tracking performance. This extended SOM introduces new features apart from the classical neurons of the output layer described in Section 3.1.1. These new features are 1-dimensional and 2-dimensional network segments that can be used to recognize fingers or palm shape in an hand image.

The main difference of the extended SOM algorithm with respect to the standard one, is that the tuning of parameters is not applied to single nodes anymore, but it can be applied to 1-D or 2-D segments in their whole, so that the map can recreate the shape of the hand as accurately as possible.

Even if the standard SOM was already able to recognize hand shape and different hand positions, the extended configuration allows to recognize them more precisely, due to its improved anatomical fidelity compared to the standard SOM topology.

The whole application was developed in an embedded system connected with the 3-D camera. This allowed to track hand shape and position in real time with high precision, thanks to the extended SOM model.

Another article written by Andrew Lee and Claus Rinner, from the Ryerson University of Toronto [**?**], shows the flexibility of the SOM, which can be used in very different fields. In their article, the two researchers of the Department of Geography applied a SOM to identify and visualize urban socio-economic changes

in the neighborhoods of the city of Toronto.

They leveraged the features of SOM to analyze high-dimensional data coming from the Canadian Census, about the socio-economic conditions of the Toronto population through the years. They selected 15 variables from different categories of the Census report to be used to measure the socio-economic changes in Toronto society.

Data from different years were collected and normalized to be comparable, and then grouped into a unique input dataset. Three particular time milestone (1996, 2001 and 2006) were selected to analyze the evolution of the Toronto society changes over the time.

By applying the SOM algorithm through a particular software, it was possible to have a correct and effective visualization of clusters, referring to the groups of social features present in the different Toronto neighborhoods.

The clustering configuration of each one of the "milestone years" was studied, to understand how the clustering changed through the years. Neurons that represent the Census year 1996 appear to have no clear gaps between them. This pattern suggests the existence of widespread socio-economic diversity. 2001 and 2006 neurons are more separated into clusters that are more recognizable from their different locations on the SOM plane.

Neurons of 2001 are grouped in 5 clusters, with 2 groups that are much bigger than the other 3. Group C is the largest group, which has a higher proportion of population with university education, white collar occupation and self-employed. Group D, the second largest one, has higher values in blue collar occupation, lone-parent families, rental tenure, average number of children, and visible minorities of black and Latin American.

Neurons related to 2006 are clustered into four groups. Group I is the largest one, which takes the most part of the network neurons and Census tracts. Since one single group occupied most of the neurons in the map plane, further analysis on this group was needed to discover the real clusters in it. This additional analysis showed how 3 more clusters could be found in Group I, but actually these clusters are not well defined and overlapping between each other.

Another kind of analysis has been made by highlighting temporal trajectories to determine how clusters moved into the map topology in different temporal intervals. Studying the trajectories it was possible to analyze how deeply different neighborhoods in the city changed in the time interval. Since in the lattice of the output layer of a SOM neurons that are different between each other are placed in distant positions of the SOM plane, the analysis of the length of time trajectories can be used to understand how deeply the socio-economic structure of a neighborhood changed. If two neurons referring to the same neighborhood get closer through the years, having short temporal trajectories between them, it means that the socio-economic change in this neighborhood was not so deep. On the contrary, if two neurons referring to the same neighborhood have divergent long trajectories through the years, it means that important socio-economic changes happened in the neighborhood.

This particular example showed the power of SOM not just in analyzing the topological structure of a fixed high-dimensional dataset, but exploited the SOM properties also to analyze the evolution of data through time, which can be a very

important application in many fields of study.

A proof of SOM versatility is the big amount of works based on its use, related to many fields of study.

For example, in [**?**] a use of SOM in finding possible relationships among genes and their possible contribution in some biological processes is presented. 282 genes have been selected, that have been shown through biological experiments to have an activity during the cell cycle. Thanks to the use of SOM, it was possible to find clusters of genes with similar behavior during the cycle.

In [**?**] the use of SOM aims to map the relative differences in competitive performance between some major global cities in a preselected set. The idea is to obtain a hierarchical benchmark analysis of these cities on the basis of their socio-economic power, expressed through variables which resemble how it is perceived by various groups of relevant urban stakeholders.

A similar application of SOM is reported in [**?**], where the automatic clustering algorithm is exploited to classify urban spatial structure in US cities. Six key variables have been selected from the urban literature: commuting costs, density, employment dispersion and concentration, land-use mix, polycentricity, and size, used to define measures to quantify the description of each city. Measures are then applied to 359 metropolitan areas from the 2000 US Census to obtain the classification through the SOM algorithm.

Finally an analysis on micro-finance institutions and their efficiency is reported in [**?**]. In this case, SOM is used to analyze how efficiency and social impact of micro-finance institutions are related between each other, to cluster the institutions on the base of variables describing those aspects of their activities. Investigating the association between social efficiency and financial performance, on a data set that includes 650 micro-finance institutions, using the SOM algorithm, it was possible to find evidence of a significant, positive relationship between social efficiency and financial performance.

3.2 Multidimensional Scaling

MDS is a set of data analysis techniques that are used to display the structure of distance-like data in a 2-D space as a geometrical picture or graph.

MDS can show the structure of a set of objects starting from data that approximate the distances between pairs of the elements of the set. The term "similarity" is generically used to refer to any metric used to express relationships between pairs of elements in the object set. A very important point about MDS is that input data used by MDS have to refer to similarity or dissimilarity metrics used to compare objects in the input set. The data can be an "objective" similarity measure or an index calculated from multivariate data.

In case of data referring to similarity between objects, the value of the metric will be higher as the elements are similar between each other, while in case of dissimilarity metrics, the value of the metric will be higher as compared elements are different between each other.

The input set of objects is usually made by elements that are represented by an array of values, which refer to different parameters, used to describe the characteristics of the considered element. The input data to be given to MDS must

be computed using a proper metric which is able to express the difference between two elements of the input set, for example the euclidean distance between two arrays. The input dataset is made by the values of the distance metric computed between each possible pair of elements of the input set, usually stored in a distance matrix.

The result of the MDS algorithm is a representation of the distances between the objects of the input set in a new lower-dimensional space, usually a 2-D space. That is, two similar objects are represented by two points that are close together in the low-dimensional output space, while two dissimilar objects are represented by two points that are far from each other.

MDS is a generic term that includes different algorithms. These algorithms can be classified according to the type of similarity data used for the computation: in case the data are qualitative, the algorithm is a non-metric MDS, otherwise, if similarities are quantitative, the algorithm is a metric MDS. The number of similarity matrices and the nature of the MDS model are other parameters used to classify MDS algorithms. Classical MDS algorithms use one matrix and an unweighted model, replicated MDS use several matrices and an unweighted model, while weighted MDS combine the use of several matrices with a weighted model.

MDS has been used in this work to obtain a correct representation of the output layer of the implemented SOM instance. In our application the output layer was made by a lattice of neurons characterized by weight vectors of 7 or 11 values. Each weight vector can be associated to an object of the typical MDS input set. Once the similarity matrix between neurons of the output layer has been computed, it can be used as an input for the MDS algorithm, to obtain a 2-D representation of the output layer, used to show the clusters obtained with the training of the map.

A particular MDS algorithm has been used to obtain the SOM output layer visualization, which will be explained in Section 3.2.1.

3.2.1 The Sammon Algorithm

Sammon mapping is a nonlinear metric MDS algorithm that maps an high-dimensional space to an output space of lower dimensionality. The main feature of the Sammon algorithm is preserving the structure of inter-point distances in high-dimensional space in the corresponding lower-dimension projection. The method was proposed by John W. Sammon in 1969 [**?**] and since its introduction, it became one of the most successful and widespread MDS algorithms.

Like for every other MDS algorithms, the input of the Sammon algorithm is a similarity matrix, which contains the values of a particular distance metric, computed for each possible couple of elements of the high-dimensional input space. The distance metric that is usually used is the Euclidean Distance between the elements of the input space.

To explain how the algorithm works it is necessary to introduce another metric, which has a fundamental role in the Sammon algorithm, called Sammon's Error or Sammon's Stress. Denoted the distance between i-th and j-th objects in the original space as d^*_{ij}, and the distance between their corresponding projections as d_{ij}, the Sammon's Error is computed as:

$$E = \frac{1}{\sum_{i<j} d^*_{ij}} \sum_{i<j} \frac{(d^*_{ij} - d_{ij})^2}{d^*_{ij}} \tag{3.7}$$

The goal of the Sammon algorithm is to minimize the error metric shown in Equation 3.7. The minimization can be performed in many ways, for example by means of gradient descent methods, or more commonly involving iterative methods. Coordinates of projections of input space elements in the lower-dimensional space are computed at each iteration using MDS and the iterative process continues until the error metric goes under a specified threshold. The number of iterations required to complete the process has to be experimentally determined and the convergence fo the solution is not always guaranteed.

In our application a dedicated software module implementing the Sammon mapping has been used to obtain the coordinates of neurons of the map in a 2-D output space, as explained in Section 4.2.3. The result of the algorithm was very good and an effective visualization of the output layer of the SOM was obtained, as shown in all the graphs of Section 6.2.

3.2.2 Applications of MDS

With reference to MDS, many different algorithms and technologies are related to high-dimensional data visualization and analysis. Due to this high number of possible implementation of the general MDS algorithm, the number of possible applications is huge too. The general MDS algorithm can be applied to almost any field of study in which analysis on high-dimensional data is needed, adapting it to the purpose of the research that is being developed. This section will present some applications of the general implementation of MDS, including a specific application of the Sammon algorithm.

The first considered application was proposed by Michael C. Hout et al. in [**?**].

The paper deals with visual search, one of the most widely studied topics in vision science, both as an independent topic, and as a tool for studying attention and visual cognition. The study on visual search focuses on how people are able to search for a particular visual target in an environment, analyzing how easy or difficult is to find the target, according to its particular features and the features of the surrounding non-target elements, called "distractors".

A crucial factor to determine the easiness and likelihood of success of a search is the similarity between the target and the surrounding elements. Any visual feature the human eye can track can be considered as a visual stimulus, so, according to its complexity, we may have a different number of features to consider to determine the similarity of two or more stimuli. For some simple stimulus characteristics, like color or orientation, it is easy to find a way to quantify the similarity, while for more complex stimuli it may be more difficult. Usually for complex stimuli, similarity is expressed on the base of consensus impressions from researchers, who try to find a way to define dichotomies between similar and different features.

In this article, MDS is proposed as a solution to quantify the similarity between different complex visual stimuli, characterized by multiple features. Conducting an MDS analysis on the stimuli that will be used during the visual search, it is possible to quantify the extent to which stimuli are perceived as resembling one another,

without the need of any dichotomy, but having a continuous scale of similarity. These quantifications can then be used in conjunction with simple accuracy or reaction time measures, as well as eye-tracking metrics.

As input, the MDS algorithm takes some raw similarity metrics, computed during some experiments with people asked to perform a visual search. The raw similarity metric can be obtained directly, asking people to indicate the level of similarity between two objects they perceive, or in an indirect manner, for example computing the time taken by people to find that two items are not identical. The output of MDS is a map that quantifies the perceived relationships among the stimuli. During the analysis, each item is moved in the output space to a location relative to the other items that, as best as possible, resembles the raw similarity metric results provided by people who did the visual search. The result of the algorithm provides coordinates for object shown during the experiment, which can be used to provide a representation of the map, in which distances between objects reflect their similarity.

Summarizing, the application can be divided in three steps:

1. Collect similarity ratings for all the stimuli proposed during the visual search experiment.

2. Apply MDS analysis to the similarity ratings.

3. Use the output to quantify the relationships among items and examine the results of the experiment considering the MDS-derived similarity features.

The main problem of this approach is that the starting point of the whole computation are subjective data, expressed through the similarity metrics collected during the experiments. To overcome this problem more objective metrics like eye-tracker data can be included in the features representing the different stimuli. What is more eye-tracking data are able to disclose aspects of the visual search task that cannot be collected using the subjective metrics coming from participants interviews.

MDS turned out to be a robust and simple method of quantifying similarity between visual stimuli in a totally objective manner. Although there was an important visual component to this task, the usage of MDS allowed researchers to quantify the semantic rather than visual similarity between items.

Another application of MDS is provided by Bojana et al. in [**?**], where a particular MDS algorithm was used to develop a new approach for searching and indexing software models repositories.

Software models are used to represent architecture and functionalities of software projects, they have a specific structure, which is expressed using a particular syntax of a modeling language. Each modeling language conforms to a metamodel, which defines the structure, semantics and constraints for building a model. Software model repositories are used to store important and useful software models, which can be reused by other developers as a reference to build new software projects with a similar structure or functionalities.

The main problem with this kind of repositories is to find an efficient way of indexing and searching in them. Most software model repositories allow users to apply a general-purpose keyword search, which is a scalable way of searching, but is not suitable for searching in software models, since it does not consider the structure

of the model to be retrieved. On the other hand, approaches that query the system considering the structure of the model can usually be applied only to specific domains, characterized by well-known and well-recognizable features, limiting the power of the search.

The proposed solution is applied to a Web Modeling Language (WebML) repository [1], and it is based on the transformation of software models into graphs. The solution applies MDS to turn graph nodes into points in a multidimensional space, to be able to perform an efficient and effective comparison between any query model and models in the repository.

WebML models are turned into attributed graphs, in a way that each element in the model is represented as a node of the graph, while containment relationships and links between the elements of the model are represented as edges of the graph. This kind of graph representation is the one that maintains the structure of the model best, thanks to attributes associated to both edges and nodes. Turning models into attributed graphs, the problem of searching into a collection of models turns into searching into a collection of graphs.

The main objective of the application is to find a ranked list of elements in the repository that are as similar as possible to the query model, or better to the query graph. Similarity is computed on the base of the text content of labels associated to nodes and edges of the compared graphs, but also on the base of the similarity between their topology.

Given a set of distance values between pairs of objects, MDS can assign coordinates to each object, such that distances computed from the assigned coordinates are as representative as possible to the distances between objects. This idea was exploited for clustering and then efficiently indexing nodes of the model graphs. Each node of any graph can be associated to particular coordinates, and the comparison between coordinates allows an easy comparison between whole graphs.

Considering the name, type and data attributes associated to nodes, comparison between nodes is achieved by transforming vertex attributes values into coordinates of points in a multidimensional space. The Euclidean distance between points is used as a metric to express the preserved distance between nodes. These computed distances allow to find all the graph nodes that are close to a given node according to a specific attribute. In other words, given a node with certain attributes, thanks to MDS it is possible to find nodes in the repository which have the same, or very similar attributes. Performing the same comparison on all nodes and edges of a query graph with all the graphs in a repository, it is possible to compare whole graphs, and so whole models.

The particular MDS algorithm used in this application was the Chalmers algorithm, chosen due to its low computational overhead and low noise generation. Parameters of the Chalmers algorithm such as number of iterations, max number of points in the random set and number of dimensions that are used for representing points in space, have been manually tuned to their optimal values.

Given a query, the developed application returns a set of modeling patterns ranked as relevant to the query. To assess the quality of the application, the relevance

[1] WebML is a Domain Specific Language (DSL) for designing complex web sites. It consists of two parts: data model, which describes the data requirements of an application, and web model, which describes the organization of the front-end interfaces of a web application.

of each returned modeling pattern was evaluated on the base of the researchers experience. However, a modeling pattern might span multiple project segments. In order to assess the overall model relevance to a query, all of the project segments that the modeling pattern spans are considered. The overall relevance is computed as an average of the relevance of each project segments.

The application turned out to have very good performance in terms of computation time, while it is not perfectly accurate in the retrieval of relevant result. A prospected future expansion of the work will consist in adding alternative objective functions for ranking, which should improve the performance in terms of accuracy.

Finally a particular case study is discussed, where the combination of a SOM and the Sammon algorithm has been applied in the analysis of groundwater-quality time-series [**?**].

Alluvial groundwater is an important source of drinking water in many countries, providing almost 50% of the global drinking-water supply. Many drinking-water extraction wells are located near rivers, exploiting the high-permeable deposits in the river valleys.

It is not always possible to avoid river water infiltration into drinking water extraction wells and increased infiltration of river water into an aquifer during high discharge events [2] can result in microbial contamination of wells. The potential of the contamination is influenced by many factors, like the position of the well with respect to the river, the composition of the river bed, but most important the nature of the high discharge event.

The influence of river discharge events on groundwater can reduce the quality of the extracted drinking water, even making it not drinkable anymore. The production of drinking water should be influenced by the study of groundwater quality, but it is not easy to find good qualitative parameters to be used for an effective analysis of the water. Moreover, to see how the river events influence the quality of groundwater through times, a huge amount of data referring to the different quality attributes is needed, and the problem of interpreting those data raises.

Once parameters to define quality were selected, time-series data were normalized to be comparable, subtracting the mean and dividing by the standard deviation.

SOM have been applied in this situation, to find some particular states in the analysis of the variables which influence groundwater quality through time. By analyzing how groundwater quality parameters change over the time, it was possible to identify some particular configurations of those parameters, corresponding to river events that affected groundwater quality. The Sammon algorithm was applied to the output of the SOM to associate a particular coordinate to each relevant state of groundwater quality and display the coordinates in a 2-D graph. Close points in the graph represented similar states of the groundwater quality, and each graph considered a different time period. Each time period was selected to be sure that at least one high discharge event was included.

The analysis of the clusters of points in the 2-d graphs it made possible to identify similar states related to different observations of the quality parameters. Connections between points in the graph represented time relationships between

[2] The discharge of a river (or stream) is the volume of water that flows past a point in the river course per second. High discharge events are conditions in which the river discharge increases a lot in a certain period of time.

observations, connecting consequent observations. Dividing observations referring to different periods of time in different graphs it was possible to find similar features in the graphs patterns.

Each graph referring to a different period including an high-discharge event, shows a similar pattern of observations, with points in the graph distributed along a V-shape or U-shape pattern. In all of the considered periods, the movement of observations in the graph started from the upper-right part of the graph, going to the lower-right part, and finishing the movement in the upper-left sector of the graph. This shows how the quality-variables considered in the clustering process evolved in a similar way during different high-discharge events, observed in different time periods.

By isolating the three main observation periods, it was also possible to highlight the evolution of specific variables in those periods, coloring the points related to the observations according to the value of the considered variable in that observation. For example it was possible to observe how higher values of the "groundwater head" variable were located in the lower part, while lower values of "groundwater head" were related to the observations in the upper parts of the graph, during all the observation periods. This result shows how an high discharge event increases the level of the "groundwater head" parameter, since the observations in the lower part of the graph correspond to the time period which is in the middle of the high discharge event. It was possible to apply the same approach to all the other variables, to find particular patterns in the evolution of different parameters during the high discharge event. Analysis on precise crucial parameters, some of which were not even included in the training of the SOM, disclosed particular patterns that were used to highlight potentially hazardous situation.

The use of SOM and Sammon algorithm was fundamental to allow the analysis of the evolution of the big amount of variables that influence the quality of groundwater. Thanks to those technologies it was possible to isolate and recognize particular states in time, studying the features that could make those states dangerous for the water quality.

3.3 Principal Component Analysis

PCA is a method of identifying patterns in data, expressing them in such a way to show their similarities and differences. PCA allows to highlight the principal components of high-dimensional data, to be able to analyze and visualize them, since structures and features in high-dimensional data are difficult to be found.

One of the main advantages of PCA is dimensional reduction, since once the principal components have been found, they can be used as an approximate representation of the initial high-dimensional data, which usually does not imply loss of information. PCA can be applied as a particular kind of compression technique in this sense.

PCA can be applied to any high-dimensional dataset and the complete algorithm can be divided in 6 steps:

1. Get some data. It all starts with a dataset to be analyzed. The dataset can be of any dimension, and on the base of the dimension PCA can be applied to

reduce it or just to normalize data.

2. Subtract the mean. It is important to apply PCA that all the data are normalized with respect to the mean. Suppose for example to have a dataset made of 2 dimensions, X and Y, where the mean referring to the X dimension is $\overline{X}$ and mean referring to the Y dimension is $\overline{Y}$. Each object in the set is a vector of two coordinates, so to complete the mean normalization, its X component have $\overline{X}$ subtracted, while its Y component have $\overline{Y}$ subtracted. The result of the mean subtraction is a new dataset whose mean is 0 across all the dimensions.

3. Calculate the covariance matrix. Covariance is a measure that express how much a dimension vary with respect to another. Covariance is always computed between two different dimensions of a dataset, otherwise, if it is computed between one dimension and itself it is called variance. Covariance between X and Y dimensions can be computed as:

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{n-1} \tag{3.8}$$

   In Equation 3.8 the term n refers to the number of elements in set X or Y, while $\overline{X}$ is the mean over the X dimension and $\overline{Y}$ is the mean over the Y dimension. Computing the covariance between all the possible couples of dimensions we obtain the elements of the covariance matrix, needed to apply PCA:

$$cov = \begin{bmatrix} cov(X,X) & cov(X,Y) \\ cov(Y,X) & cov(Y,Y) \end{bmatrix}$$

   Non-diagonal elements of the matrix can tell us the way a dimension change according to another dimension. In a 2×2 matrix if the non-diagonal terms are all positive it means that the x and y variables increase or decrease together.

4. Calculate the eigenvectors and eigenvalues of the covariance matrix. The covariance matrix is always a squared matrix, so eigenvectors and eigenvalues can be computed for it. Given a square matrix A, a vector V is an eigenvector of matrix A, if and only if $A * V = \lambda * V$, so the product between the matrix and the vector gives a result which is equal to the product of the vector V for a scalar value λ. The scalar value λ for which the equality is satisfied is the eigenvalue corresponding to eigenvector V.

   By computing eigenvectors and eigenvalues of the covariance matrix we obtain important information about the principal components. Each eigenvector represents a direction in the input space, and so a particular component or structure within our original data distribution. Each eigenvector represents a different direction within the distribution, so a different component and all eigenvectors are perpendicular or better orthogonal between each other. Therefore, by this process of taking the eigenvectors of the covariance matrix, we have been able to extract features that characterize the data. The next step of the algorithm consists in transforming the data, to represent them in terms of those features.

5. Choosing components and forming a feature vector. From the previous point it was possible to understand how eigenvectors represent the components that form our initial dataset, but the meaning of eigenvalues has not been explained yet. Each eigenvector is associated to a corresponding eigenvalue, and different eigenvectors correspond to different eigenvalues. The eigenvalue shows how significant a certain component is in the input space distribution. If the eigenvalue is high, there will be a large distribution of the elements of the input set along the component associated to the examined eigenvalue. If the eigenvalue is smaller, the distribution of elements along the corresponding component will be smaller too.

   According to this definition of eigenvalue, the component whose eigenvector is associated to the highest eigenvalue is the principal component. Since eigenvectors and eigenvalues are computed from the covariance matrix, the next step is to sort eigenvectors according to the correspondent eigenvalue, in decreasing order of eigenvalues. Once the eigenvectors have been ordered, it is possible to decide to apply a dimensional reduction or not. Choosing only some of the eigenvectors with higher corresponding eigenvalues we can reduce dimensionality assuring that we are picking the most significant components from the input space.

   The set of selected eigenvectors form the so called feature vector, which is an array of eigenvectors, sorted by decreasing eigenvalues.

6. Derive the new dataset. Once the feature vector has been computed and so the eigenvectors of the components to consider have been selected, it is possible to create the new dataset. To obtain the final result the feature vector and the original dataset have to be transposed, then the transposed feature vector has to be multiplied on the left of the transposed original dataset.

$$FinalData = RowFeatureVector \times RowDataAdjust \tag{3.9}$$

   $RowFeatureVector$ is a matrix in which each row corresponds to an eigenvector, from the most significant on the top, to the less significant in the bottom position. $RowDataAdjust$ is the initial dataset, normalized by the mean and transposed.

   This result represents the original dataset only in terms of the principal components that have been selected at the previous step of the process. If the selected components are not all of the original components, the new dataset will have lower-dimensionality with respect to the original one. If otherwise all of the eigenvectors have been used to create the feature vector, the result of the whole process is the same data distribution normalized by the mean and expressed in function of the eigenvectors directions, instead of the original axis. It is possible to invert the whole process and come back to the original data only in case all the eigenvectors have been included in the feature vector. In this case there is no loss of information, otherwise the dimensionality reduction causes a loss of information which is not reversible.

PCA has been applied in this work for the visualization of the output layer of the SOM. The idea was to obtain a new reduced output layer in which each neuron was characterized by a weight vector of 3 elements, to be associated to the RGB component of a specific color. Having a 2-D grid of RGB triplets it was possible to display the map as an image, considering each neuron as a pixel with its own color. This is a typical application of PCA, which has already been used for coloring the output layer of a SOM in an application reported in a paper by de Runz et al. [**?**]. The obtained image was expected to show the clusters of the output layer of the SOM as groups of close pixels with similar colors, but actually the result was not satisfying, since the clusters were not recognizable applying this method on our instance of the SOM.

3.3.1 Applications

The PCA, as the other technologies presented in this chapter, is very versatile and can be applied to study the structure of high-dimensional data coming from any source. Some applications that leverage the PCA related to different fields of study are presented in this section.

The first application presented comes from the results of the work of a team of French researchers of the Pasteur institute in Paris. In the article [**?**], the PCA is applied in the study of the shape of protein cavities and their evolution during molecular dynamics.

Proteins functions strongly depend from their shape, since proteins exploit their functions by mean of the physical interaction with other proteins, nucleic acids, substrates, or more in general other molecules. Usually the shape of a specific protein actually determines its function, so analyzing the shape it is possible to understand how to exploit the potential of a certain protein.

Protein cavities are one of the main features that determine proteins shape, and their evolution during the molecular dynamic process can affect the evolution of protein functionalities. A cavity is defined as the volume accessible to a small solvent probe delimited by a "solvent accessible surface", but inaccessible to the "bulk solvent" delimited with a larger probe. Anyway it is not simple to recognize and analyze the shape of proteins cavities, and most existing quantitative studies use exclusively simple descriptors of cavity geometry such as its surface area, its volume or the position of its geometric center.

Once the main attributes defining a protein cavity have been determined, it was possible to perform PCA on those attributes, finding the principal components that are in common with different proteins and different protein cavities. PCA was also applied to analyze the effects of the molecular dynamic process on proteins shape, since it is easy to compare the evolution of principal components during the process for different proteins.

Another important result obtained with PCA analysis is the possibility of the invert the process. The study on PCA was not used only to compare existing cavities, but it was also possible to design particular cavities shapes, starting from properly selected principal components. This is a very important result, which can have an important application in drug design. Having a more precise analysis on cavities and the possibility to recreate specific cavities on the base of principal component,

can improve drug design a lot, since combining proteins with specific shapes, it is possible to obtain functionalities close the desired one in an easier way.

Despite some technical limitations, like the big size of data produced [3], the method proved to be powerful and robust and opens new opportunities to visualize and explore the dynamics of protein cavities. This technique is a powerful and precise way to decompose and classify dynamical evolution of cavity geometry in molecular dynamics.

Another application was proposed in [**?**] in the agricultural domain. In their presentation, the Japanese team of researcher from the Osaka Prefecture University applied a combination of PCA and SOM technologies for visualizing and classifying fire risks in Indonesian forest regions.

Fires risks in forests are monitored thanks to data related to the so called hot spots, sections of forest or woods where fires frequently occur. By monitoring some parameters related to hot spots, it is possible to analyze the fire distribution and fire risks in a forest. The goal is to try to prevent fires or at least to reduce the amount of damages caused by them. Finding clusters of hot spots in a forest can be important to detect regions of the forest in which the fire risks are higher.

Detailed data about hot spot parameters were collected in a four-years period, from 2000 to 2003. Data were divided in 11 categories, referring to the 11 months from January to November. Data about the month of December were not available for year 2001, so researchers decided not to consider the month of December in the whole study, to have data of the same size referring to each year. The collection of data related to many hot spots spread across the Indian forest and the analysis on many months of observation, turned into the creation of an high-dimensional dataset.

PCA was applied to extract the main features and patterns in the fire risk distribution across the different areas of the forest and through the 11 months in each considered year. It was found that the first 2 principal components of the input dataset described the 84.72% of the variance of the whole data distribution. The higher percentage of variance indicated that these PCs are suitable used to explain the distribution pattern of the fire risk, according to hot spot details in forest regions for different months.

The analyzing of the two principal components relate with the 11 months allows to determine how September and March were the two months with higher fire risk, while January and November were the two months with lower risks. Moreover, considering the value of the two principal components related to each observation parameter, both months observed and forest region considered, it was possible to show a 2-D distribution of fire risks using a scatter plot. PCA plot shows the correlation pattern between samples (regions) relative to the certain input variable (months). For example, the regions with larger variance of fire risk occurrences are grouped at around the months with higher fire risk. On the contrary, regions with smaller variance of fire risk are grouped at around of the months with lower fire risk.

Even if the two principal components describe most of the variation in the input dataset, the PCA plot was not enough to show hot spot and fire risks distribution in a satisfying and effective way. The plot showed the correlation between the observed

[3]Decomposing a very complex shape in principal components can bring to the generation of some GB of data.

areas and the risk related to the observation months in an effective way, but to better visualize the fire risk distribution SOM was applied. Thanks to SOM it was possible to find, for each of the considered months, clusters of regions to be classified with a different level of fire risk. Studying the evolution of the SOM structure through the whole period it was possible to analyze how the fire risk level changed in the different forest areas.

This application effectively compares the results of the PCA and SOM technologies. PCA performed very well to explain the variance of dataset, since the extraction process provided a high percentage of the first two principal components. The results indicated that PCA has explained most the cumulative variance of data; unfortunately, the PCA projection was difficult to reveal a representative data pattern when the applied data available have large-scales. In contrast, SOM presented the variance of the input data based on a visual identification of the relative distances between the nodes network, showing data distribution patterns in a more effective way.

Chapter 4

Software Implementation

The solution to the problem of redefining basketball positions was implemented in a unique software project, completely written in Python language. In this chapter the architecture of the whole project will be presented, starting from the external tools integrated in the system, to conclude with a detailed explanation of all the software modules composing the final version of the project.

Python is an interpreted, dynamic programming language, born in the early 1990's, designed to emphasize code readability and to allow coders to write more code in a faster way thanks to its syntax. Python supports different programming paradigms, included the object-oriented one used in this project. In this thesis, the 2.7 version of Python was used, even if it is not the latest released one. This decision was made due to compatibility needs related to the main external library used to implement the SOM module, which was supported only by Python 2.7 and not by the latest 3.5 version.

4.1 External Libraries

As explained in Chapter 3, the technologies used to implement the final solution are quite consolidated. On the other hand, the goal of this thesis was not to research on them, but to exploit them as much as possible into a new field of application. Thanks to the wide diffusion of those technologies, it was possible to find external Python libraries to be integrated in the project, which provided good working implementation of the technologies we were interested in.

4.1.1 PyMVPA

PyMVPA is an open source Python package developed by the collaboration of 3 German researchers, an Italian and a US universities, integrated with the work of different external contributors. The PyMVPA package contains many modules developed to ease statistical learning analyses of large datasets. It offers a framework with an high-level interface to many algorithms for classification, regression, feature selection, data import/export, and it is designed to integrate well with already available Python software packages dealing with similar problems, such as scikit-learn, shogun or MDP. Even if it was firstly developed to be applied to the neuroimaging

domain, all of the algorithms are applicable to any field of study in which large datasets analysis is needed.

Among all the different modules of the PyMVPA package, the one used in this thesis was the `SimpleSOMMapper`, which provides a simple but effective implementation of a SOM.

This module is part of the "mappers" category, a set of components implementing general algorithms for data reversible transformation. A mapper is a module that can take an entire dataset or just an array as an input, and can performs some operations on it, modifying the input dataset. The operations on a dataset can be as simple as selecting a subset of it, or more complex, like data projection.

Mapping algorithms can be unsupervised, like the one implemented in the SOM module, but also supervised algorithms for data transformation are implemented in other modules of the mappers package. Three modes of operation can be implemented by the mappers:

- Train. All mappers can be trained according to a selected training dataset. The training phase does not modify the dataset, but allows the mapper to learn how the mapping operation has to be performed. There are mappers which do not perform any training phase in their algorithm, in this case the training command is just ignored.

- Forward-mapping. The mapper takes an input dataset, performs the required operations on it and returns the result of the computation, a new modified dataset. Operations are performed on the input dataset, and the output dataset will contain the modified elements, but also the elements of the input dataset which have not been modified by the algorithm. The input dataset is not saved automatically before the algorithm operations are performed, so it has to be saved manually if needed.

- Reverse-mapping. Some particular mappers support the reverse-mapping function, which allows to come back to the input dataset once the transformation of the forward mapping has already been performed.

The "SimpleSOMMapper" module implements a classic SOM algorithm, based on the unsupervised training technique described in Section 3.1.2.

The SOM object is initialized by specifying two required parameters: The shape of the output layer, defined by a tuple referring to the dimensions of the output lattice, and the number of iterations to be performed during the training phase. Conditional parameters can be specified too, in case a particular value of some parameter related to the SOM algorithm has to be manually set.

The conditional parameters are: "learning_rate", the initial learning rate of the training algorithm; "iradius", the initial neighborhood radius used during the training phase; "distance_metric", which allows to specify a particular metric to be used in the searching for the BMU during the training phase, if it is not set the default metric is the squared Euclidean one; "initialization_func", used to specify a particular initialization function to be used to set initial neuron weights.

Regards the attributes, the module presents many conditional attributes that can be activated while the object is instantiated. The most important attribute of

the mapper is the "K" attribute. The "K" attribute refers to the output layer of the SOM and so it is made by a two-dimensional array of neurons, where each neuron is represented with the corresponding array of weights. If the dimensions of the lattice is 30x30 and each neuron is characterized by 7 weights, the "K" attribute returns a multi-dimensional array of dimension 30x30x7. The possibility to directly access the output layer of the SOM is very important for some operations, such as data-visualization or statistical analysis of the clusters in the output layer.

Finally, since it is part of the mappers package, the "SimpleSOMMapper" provides all the typical methods of mappers previously described in this section. The training method is very important to perform the unsupervised competitive learning algorithm typical of the SOM, while the forward method allows to perform the mapping of some elements of the input set in specific positions of the output layer.

The PyMVPA "SimpleSOMMapper" module was the core of the final application, since it allowed to perform all the main computations needed to obtain the clustering of the input space elements in the output space. Anyway many other operations had to be performed both by preprocessing the input data before giving them to the SOM module and by analyzing or visualizing data after the clustering process.

4.1.2 Matplotlib

Even if the PyMVPA SOM module provides everything that is needed to perform the clustering algorithm, it does not provide a direct method for the visualization of the output layer. Luckily, as explained at the beginning of Section 4.1.1, the PyMVPA package was developed to work in a good collaboration with other external Python packages, one of the most important is matplotlib.

Matplotlib is a Python 2-D plotting library created by John Hunter in 2007 [**?**]. It allows to easily create any kind of graph or 2-D figure to be shown in an interactive environment, and it is used in application development, interactive scripting, and publication-quality image generation across user interfaces and operating systems. Matplotlib allows to generate any kind of graph, like histograms, power spectra, bar charts, error charts, scatter plots and many others, with just a few lines of code, providing a very good documentation and a big set of examples to take inspiration from.

Thanks to the matplotlib "imshow" method, for example, it was possible to visualize a first version of the SOM output layer representation, where each neuron was associated to a pixel of an image with its own RGB configuration. The result is shown in Figure 4.1, where it is possible to see the scale on the axis which report the dimension of the output layer, that in this case was a 15x15 neuron lattice. Each pixel of the image was a neuron of the lattice which was colored according to the cluster it belongs to. Thanks to additional matplotlib functionalities it was possible to show some labels in the position of neurons corresponding to particular elements of the input set, mapped on the output layer.

During the experiments made on the SOM during the whole work, at some point it has not been possible to associate each neuron to an RGB triplet, as explained in Section **??**.

Since the "imshow" matplotlib method requires a matrix of RGB triplets to perform the visualization, another method was needed to visualize the output layer

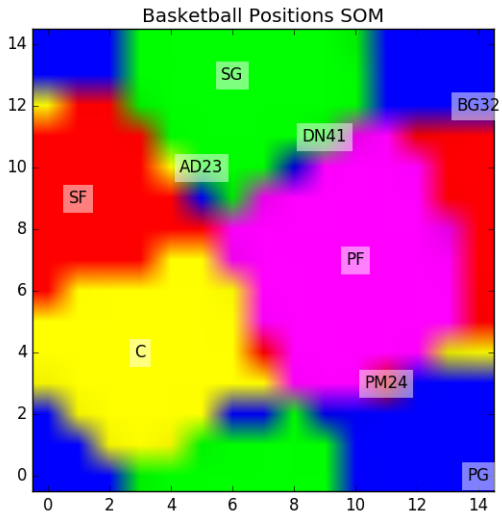

Figure 4.1: Example of SOM visualization made with matplotlib imshow command.

of the SOM. The neurons of the output layer can be seen as points in a 2-D grid, so the best solution for their visualization was a scatter plot, in which each neuron was one of the scatters. The matplotlib "scatter" command allows to create the desired scatter plot, just giving to the method as required parameters the X and Y coordinates of the scatters. Additional parameters of the function allow to specify also the size and the color of the scatters, and labels can be shown in the position corresponding to specific neurons, as in the case of the "imshow" command.

Two versions of the scatter plot visualization of the SOM have been developed. The first one, shown in Figure 4.2, uses the coordinates of neurons on the output lattice to place the scatters into the scatter plot. The result is a regular 2-D lattice that does not show the clusters between neurons as well as the application required. In the second visualization method, shown in Figure 4.3, the coordinates of scatters within the plot have been computed using the Sammon MDS algorithm described in Section 3.2.1. Thanks to the application of the Sammon algorithm, it was possible to assign coordinates to neurons in the 2-D space, in a way that the distance between any pair of scatters represents the actual difference between the corresponding neurons.

Matplotlib was the only module used in this project for the creation of output graphs and figures. It performed very well, considering the importance of the visualization to have an idea of the results obtained with the SOM algorithm. The data to be given to the visualization module had to be computed applying some of the algorithms described in Chapter 3, but the visualization provided by the matplotlib library tools was optimal for our application.

4.1.3 Scikit-learn

Data visualization was probably the biggest problem faced during the whole thesis activity. The solution presented in Section 4.1.2 only describes the way matplotlib

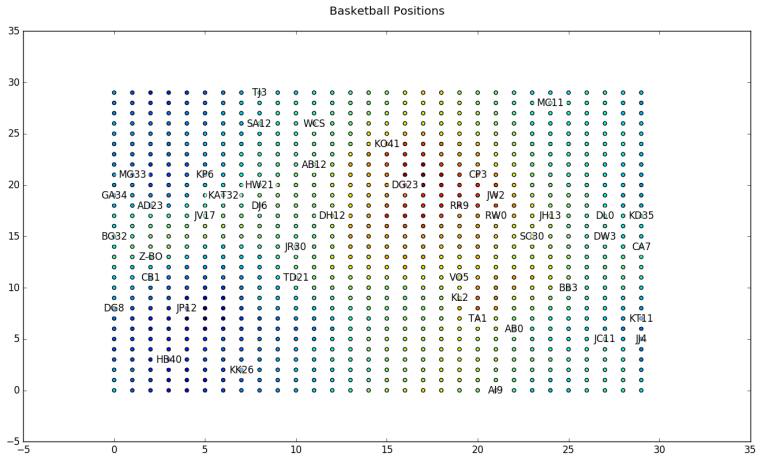

Figure 4.2: Scatter plot of the output layer with fixed positions of neurons in the lattice.

tools have been used to display the results, after some important computations have been made on the output data coming from the SOM module, but not the way those computations were performed.

Once the SOM clustering algorithm was applied, the output layer of the SOM had to be displayed. However, since the neurons in the layer were characterized by weight arrays made by 7 or 11, the elements the data to be visualized were still high-dimensional data. To solve the problem and reduce dimensionality of the output dataset to be displayed, different trials have been made, by applying some of the techniques explained in Chapter 3.

As for the SOM implementation, in this case it was possible to find some existing tools that implemented the interested algorithms, ready to be integrated in the final application. Scikit-learn is a Python open source library that provides many simple and efficient tools for data mining and data analysis, including modules for the application of PCA and MDS algorithms. The library was implemented to be efficiently integrated with other famous Python software packages for scientific computations, including matplotlib.

Firstly, the Scikit-learn PCA module was tried, to see if, thanks to principal components, it was possible to adapt the structure of the output layer of the SOM to make it a compatible input for the "imshow" command of matplotlib. Since the output layer of the SOM turns out to be a 2-D grid of arrays, each one composed by 7 or 11 elements, the idea was to apply PCA to reduce dimensionality and obtain a 2-D grid of elements characterized by arrays of size 3, to be used as RGB triplets for the "imshow" command of matplotlib. By reducing dimensionality to only 3 elements per weight vector it would be possible to associate each neuron to a pixel of an image and show the results of the clustering in a similar manner to the one shown in Figure 4.1.

Actually the result of the computation assigned to the neurons RGB triplets that were not describing the clusters in a proper way, since the output image

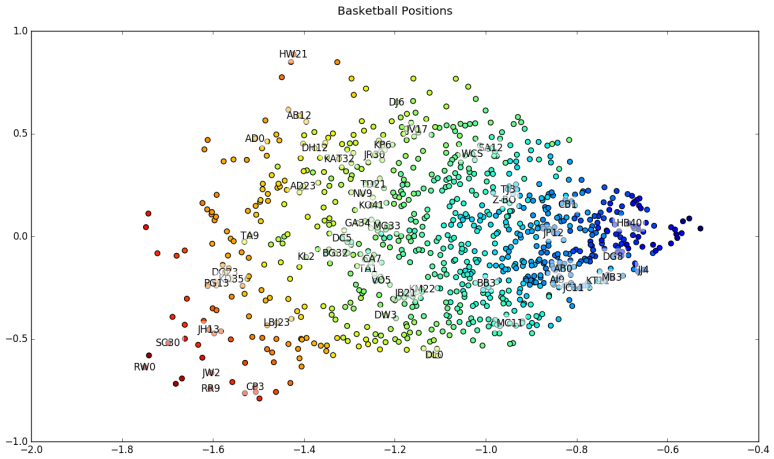

Figure 4.3: Scatter plot of the output layer with positions of neurons computed with Sammon algorithm.

obtained showed very different colors for adjacent neurons, showing actually no cluster and no pattern in the distribution on the output lattice. The main reason of the unsatisfactory result is probably to be searched in the way PCA had to be performed on the output layer. The PCA module of the Scikit-learn library requires as an input a one-dimension vector of elements of any size. To apply PCA to the whole output layer of the SOM, PCA had to be applied to one single row of the lattice at the time, and not on the whole grid. The reconstruction of the complete lattice provided a non-homogeneous result, which did not show any clustering pattern. This is due to the fact that the principal components computed in each row may be different for each row of the output layer, so even if the dimension of the result obtained by all the rows were the same, the principal component considered in each row may be different with respect to the original input set dimensions.

A second trial was done by applying the MDS algorithm to the output layer of the SOM, using the dedicated module of the Scikit-learn library. The MDS module can be initialized by specifying only the number of desired dimensions of the output space. However, the initialization method also allows to specify the number of iterations of the process, if the applied algorithm has to be a metric or non-metric MDS, a lower threshold for the error below which the algorithm has to stop and other non-required parameters to custom the algorithm as required by the specific application.

The "fit_transform" method is the most important one. By calling it, the module performs the MDS algorithm as specified during the initialization phase and returns the output space coordinates associated to the elements of the input set. In our application the output space was a 2-D space, thus the output coordinates were tuples made of two elements, representing the (x, y) coordinates of neurons inside the scatter plot. The coordinates obtained from the Scikit-learn MDS algorithm were passed as an input to the matplotlib "scatter" function, to obtain the type of visualization shown in Figure 4.4.

The obtained result was satisfying and allowed to have an initial idea of the

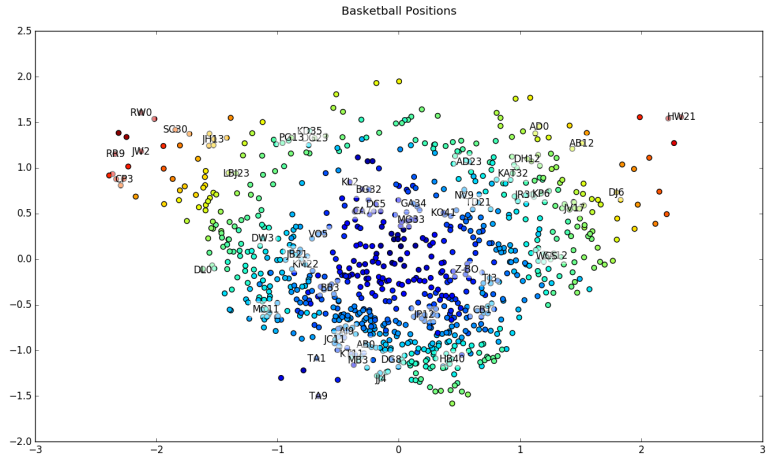

Figure 4.4: Result of the Scikit-learn MDS algorithm represented in a scatter plot.

clusters generated in the output layer of the SOM. Anyway, the representation is quite chaotic and there is an important section in the middle of the scatter plot where no labels are displayed, which means that no significant elements of the input set were mapped in that area. Clusters of neurons appear to be positioned around a sort of "central core", the are with dark blue scatters, where no elements are mapped, and this kind of visualization was not satisfying for our purposes, since it was not possible to recognize clusters in a proper way.

Even if it did not reach the optimal result, the Scikit-learn MDS module provided a good solution to the visualization problem, which was one of the most important issues of the whole project.

4.1.4 The Sammon Module

The final and optimal solution found for the visualization problem exploits the Sammon algorithm, a specification of the MDS algorithm presented in Section 3.2.1. A simple, but efficient and effective implementation of the Sammon algorithm has been found, provided in a 2014 open source project by Tom Pollard, an associate researcher at Massachusetts Institute of Technology (MIT). This project is composed by just one class, which contains the complete Python implementation of the algorithm, with all the needed functionalities.

The Sammon module contains one basic function to be called to perform the algorithm, and the only required parameter is the input multidimensional dataset on which the dimensionality reduction has to be applied. Thanks to other additional non-required parameters, it is possible to specify the dimension of the reduced output needed, a lower threshold of the Sammon error below which the algorithm stops, the maximum number of iterations and other variables used to adapt the algorithm to the application.

Calling the function, a set of 2-D coordinates are assigned to the neurons of the output layer, and returned as result of the computation. The coordinates are computed in a way that the distance between neurons in the 2-D output space

resembles the true difference between neurons weight vectors, to maintain the topological properties of the input dataset in the output space. As in the case of the Scikit-learn MDS algorithm, the resulting coordinates were used as inputs for the matplotlib scatter function, to show the result of the clustering operation in a scatter plot.

Setting the function parameters in a proper way, it was possible to obtain a very satisfactory result, shown in Figure 4.3. The figure shows an homogeneous scatter distribution across the graph, with almost no empty area in which no elements of the input set are mapped.

This visualization is the one that allowed to obtain the best results in terms of clusters definition and visualization, as will be shown in the results explanation in Section **??**. What is more, this solution provided the best results in terms of performance. Even if the applied algorithm is an iterative one, the computation of coordinates done using the Sammon algorithm was much faster w.r.t. the same computation made with the Scikit-learn MDS module.

4.1.5 Data Query Libraries

Two more standard Python libraries have been integrated in the application to handle the input dataset used to obtain the final result.

The input data on which the whole study has been made are the official National Basketball Association (NBA) statistics related to the regular season games[1] of different seasons through the years.

The data have been downloaded from a certified source and have been saved in two formats: firstly a MySQL database has been created, where each table contains data referring the statistics of all the players who played during a specific season. The MySQL database has been exploited to automatically create datasets to be used as training sets for the SOM. The creation of those training sets has been done y following precise criteria, as explained in Chapter 5, so MySQL queries have been used to retrieve the correct data according to the needs of the training set.

To be able to connect the Python application with the MySQL database, the standard "mysql.connector" Python library has been used. It allows to open a connection to the database, specifying credentials, the host and the database name. From a method of the connection object it is also possible to create a cursor object, which is responsible for executing the desired query and returning the results in an easy-to-parse format.

Initially the whole application was built for retrieving data directly from the MySQL database, even when the filtering operation on data to be retrieved was not particuarly complex. Complex filtering operations on data were needed only when players to be included in a dataset had to be selected according to precise parameters. Once the dataset has been set, retrieving data related to the same statistical categories for each player did not need any complex filtering operation.

[1]The NBA season is divided in two parts, the regular season, in which all the teams play 82 games between each other, and the playoffs, in which only the 16 best teams in the league play against each other in a tournament format, where 7-games series determine which team goes to the next round, until the Finals. The study has been applied to regular season statistics to consider all the players in the league.

Moreover, the connection of the Python application to the database was quite expensive in terms of computation time, so an alternative solution has been used to reduce the time needed to retrieve the input data.

The database tables have been exported in Comma Separated Values (CSV) files, a particular type of text file format that can store grid and tables. Python provides a dedicated "csv" library to handle CSV files, for both reading and writing them. Reading a CSV file becomes an easy task using the "csv" Python library, thanks to the "reader" object, which can be instantiated specifying the file to be read, the delimiter used in the file, and the character used to include char fields in a record. The "reader" object returns each row in the CSV file as an array, where each element corresponds to one of the comma-separated values in the row. Obviously the library contains also a "writer" module, that allows to convert data given to it in an array format into rows of a CSV file. Two other interesting modules of the "csv" library are "DictReader" and "DictWriter", which allow to map CSV files into Python dictionary structures and vice-versa, using the field names that are usually reported in the first row of a CSV file as keys for the dictionary.

The application has been modified to substitute the MySQL database with CSV files reporting the statistics related to the different seasons, which are saved locally inside the project. The main modules of the Python application retrieve the input data needed for the computations from the CSV files, through the use of the "csv" library, providing a good speedup of the whole execution, due to the fact that a remote connection with the database is not needed anymore.

4.2 Application Architecture

The whole application has been developed in a unique Python project, exploiting the object-oriented programming paradigm. The project is organized in classes and modules, written in a way that allows to separate different functionalities in different modules, integrating also the external libraries described in Section 4.1.

The core of the application is the "BasketballSOM" class, which implements the whole SOM algorithm, exploiting its own methods and specific calls to external support modules responsible for additional computations needed to let the whole application work. This section will explain the external modules with their specific responsibilities, and will conclude with a detailed explanation of the core class which exploits the smaller module for the final SOM implementation.

4.2.1 Data Retrieval Modules

As described in Section 4.1.5, the data used for the whole application have been saved in two formats, a MySQL database and some CSV files. To be able to deal with this two data formats, some components have been implemented to filter and retrieve needed data from the two different sources.

A "DBSearcher" class was implemented to retrieve players statistics from the MySQL database. The class has to be initialized by providing the database name and the name of the table to be queried, so that, thanks to the "mysql.connector" library, the module can open the connection with the correct table in the database.

Three methods are implemented in this class, which allow to retrieve statistics in different ways. The first one allows to return an average of the statistics of players playing a certain position. Given one of the five classical positions in basketball, the method queries the database for all the players labeled to be playing that position. An average of each statistical field is computed over all the players of the required positions, considering only those players who have a value of MPG which is above a certain threshold, which can be specified as a parameter of the method. The returning value is an array whose size is equal to the number of statistical categories needed for the application, and contains the average of each statistical category over the players of a certain position.

The second method allows to retrieve the season average statistics of a single player, specifying his name and surname, while the last one was implemented to retrieve average statistics of all the players in the league, with no particular filtering operation. The only restriction for players to be included in the dataset or not is a lower threshold on the value of MPG which can be specified as a parameter to the method.

The "CSVParser" class is the corresponding implementation of the "DBSearcher" module, which deals with CSV files. The class is initialized by passing the name of the CSV file to be parsed and the statistical categories to be retrieved. It contains three methods that allow to perform exactly the same operations described for the "DBSearcher" class, but acting on a CSV file instead of a MySQL database. The class was created after extracting data from the tables of MySQL database into CSV files, as described in Section 4.1.5.

Another data retrieval module was created for a different purpose. The "TrainingSetFileCreator" class has been implemented to write training set files automatically, including players according to the precise criteria described in Chapter 5. This module queries the MySQL database to find the correct players to be included in the training set. The players name and surname are saved in a text file put in a proper location; the text file will be read by other modules of the application, described in Section **??**, to retrieve statistics of the players that are included in the training set.

4.2.2 Dataset Creators

Two additional supporting modules are used to create the training set, so the set of data to be used to perform the training of the SOM, and the input set, the set of statistics referred to the players that have to be forward-mapped on the trained SOM.

The "TrainingSetCreator" module retrieves data from the database source according to the name of players included in the training set input file, which can be created using the "TrainingSetFileCreator" described in Section 4.2.1. The class is instantiated passing the path of the database, the statistical categories to be retrieved for each player and the training set file path as parameters. The module reads the name and surname of players to be included in the dataset from the training set input file, and exploits the "CSVParser" module, described in Section 4.2.1 to retrieve each player statistics from the CSV data source. The training set creator implements also a method that allows to create the training set including all the

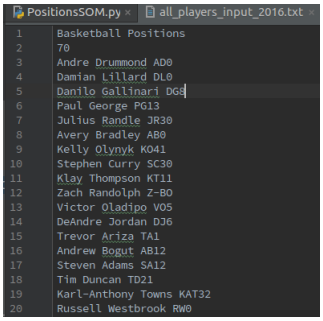

Figure 4.5: A screenshot showing part of the standard input set file.

players in the database, once again exploiting the method of the "CSVParser" that performs the operation.

The "InputSetCreator" has the same responsibility for the input set, so it has to retrieve the correct data from the CSV data source to be included in the input set. It is instantiated passing the same parameters used for the "TrainingSetCreator" class, so the data source path, the statistical categories to be retrieved and the input set file path. Its main functionality is to read the data related to players to be included in the set from the file, and retrieve statistics of the included players from the data source. Even if the overall behavior of the input set creator module is similar to the one of the training set creator, input set files are formatted in a more complex way, according to the purpose of the forward-mapping that has to be performed, and so different data-retrieval methods are included in the class. The first variant of input set creation method is called "single_plot_input" and deals with the most simple format of input file, used for a standard mapping with the aim of showing how clusters of players are positioned on the map.

As shown in Figure 4.5, the input file contains the title of the plot in the first row, an integer value in the second row, and the following rows are filled with data related to the players that have to be included in the input set, written in the following format: `"name surname label"`. The method saves the title of the graph and the integer number in the second row in separate variables. The integer number will be used for some filtering of the visualization of players labels on the final graph. The idea is to map on the output layer of the SOM all the players in the input set, but to display the position of only some of them in the final graph, if needed. The integer number in the file allows to specify how many players have to be actually shown on the final graph, and the application selects the players starting from the top of the list in the input set file. If the value of this parameter specified in the input set file is higher than the number of players in the input set, labels related to all of them will be displayed. For each row following the integer parameter, the method divides the name and surname of the player from his label and saves them in two separated lists. The list of players names is used to retrieve the statistics related to each player in the list, exploiting the method of the "CSVParser" class described in Section 4.2.1. The method returns the input set built from the data source, the array of labels to be displayed, the title of the graph and the integer parameter for limiting the number of labels to be displayed.

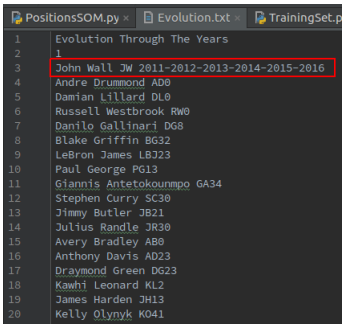

Figure 4.6: A screenshot showing part of the evolution input set file.

The second method is called "evolution_plot_input" and it is used to create the input set needed to show the evolution of a player position through different seasons on the map. The input file related to this functionality has a similar format with respect to the standard one, apart from one line. As shown in Figure 4.6, the first line contains the title of the graph, the second one an integer number, that in this case is set to default to 1, since the position of only one player must be displayed on the map. The third line contains data referring to the player whose evolution through years has to be displayed. The line contains his name and surname, the label associated to him, and a set of years, in the format reported in line 3 of the file shown Figure 4.6. The input set is made collecting first the statistics of the interested player from the seasons related to the years specified in the input set file. Since different seasons data are collected in different data sources, the module has to parse more files to retrieve all the data needed. Statistics referring to the performances of the considered player are placed in the top positions of the input set, then the algorithm works like in the case of the standard input set to complete the collection with statistics of the other players. When the visualization will be loaded, only the labels referring to the player of interest will be displayed on the final graph.

Finally, the "euroleague_plot_input" method has been implemented to analyze how players from an European competition [2] are positioned on the output layer of a SOM trained with data related to NBA players. The input set file of European players has the same format of the standard NBA input dataset file, and its path has to be passed to the method as a required parameter. What makes this method different from the others is that the final input set is actually obtained from the concatenation of two different input sets, described by two different input set files, one related to NBA players and the other one containing information about the European players. The first one is obtained retrieving statistics related to the European players from the dedicated data source, while the second one contains the statistics related to NBA players. Concatenating the two lists of statistics a big unique dataset is obtained, which allows to display NBA and European players on the same graph.

[2] The European competition we are referring to is the Euroleague, which is the top level basketball competition for European teams.

4.2.3 Visualization Loader

Some different techniques of visualization have been implemented in the application, both to answer to different visualization needs, but also to try to find the optimal solution for visualizing the results. The "VisualizationLoader" class was created to contain all the visualization methods implemented in the application, so that, during the main work-flow the correct visualization method can be called according to the needs. The class has to be initialized providing some required parameters used by the visualization methods: the current instance of the trained SOM, the input set of data to be mapped on the output layer, the labels to be displayed in positions corresponding with the mapped data, the title of the graph, the shape of the output layer[3] and the number of weights associated to each neuron, which corresponds to the number of statistical categories used to represent each player statistical profile.

The first implemented method is called "load_visualization", and was used to create a result similar to the one shown in Figure 4.1. It exploits the "imshow" method of the matplotlib library described in Section 4.1.2, so it needs as required parameter a color map, as the "imshow" method does. The color map is a matrix of arrays, made by 3 elements each, corresponding to the RGB triplets needed to create the final graph. Each element of the color map corresponds to a neuron of the output layer, with its associated color. The visualization method displays the color map, computes the positions of the input set elements on the output layer and displays the labels associated to each element of the input set in the corresponding position on the map. The result of the visualization is shown in Figure 4.1, and it is used to demonstrate how players are positioned on the map with respect to the five standard positions of basketball. An experiment was made using this visualization method to display the result of the clustering operation for the definition of the new positions, exploiting another method of the "VisualizationLoader" class. The additional method we are referring to is called "applyPCA", and takes as unique required parameter the whole output layer of the SOM. This method is used to create a color map from the already trained output layer of the SOM, applying the Scikit-Learn PCA algorithm described in Section 4.1.3. The idea is to apply PCA to reduce dimensionality of the output layer from a 2-D matrix of arrays made by 7 or 11 elements each, to a 2-D matrix of arrays made by 3 elements each, in a way that each element of the matrix can be used as a pixel of the output image with its own RGB color. Actually, even if the dimensionality reduction was applied with success, passing the obtained color map to the "load_visualization" method, the result was not satisfying and no cluster could be identified in the output layer topology.

The second implemented method is "load_visualization_scatter". It is the first trial made to try to display the output layer of the SOM as a scatter plot. This method requires as a unique parameter the integer value coming from the input set file described in Section 4.2.2, called "show_limit" and used to determine how many labels of forward-mapped players have to be displayed in the final result. Firstly, the positions of input set players on the output layer are computed. Afterwards, the method calls a small external module, which returns the coordinates of scatters and

[3]It is passed as a unique entire number, since in this application a squared output lattice is used, so giving the size of one dimension, l, the overall shape of the output layer is $(l \times l)$.

the colors associated to each of them. In this case the coordinate of each scatter in the output graph is simply obtained from its position in the output lattice of the SOM, while the color associated to each of them is obtained computing the norm of the weight vector that characterizes the neuron. The X and Y coordinates of scatters and their colors are passed to the matplotlib "scatter" function described in Section 4.1.2, and the labels associated to players within the "show_limit" are displayed in the correct positions. The final result of the visualization is shown in Figure 4.2, but it was just the first step through the final result, since it does not allow to see the clusters in the output layer in the most effective way.

The next step through the optimal result is the "load_MDS_scatter" method, which exploits the MDS algorithm of the Scikit-Learn library, described in Section 4.1.3, to compute the coordinates of scatters in the final plot. Firstly the method computes the mapped elements from the input set onto the output layer, then the output layer matrix of neuron is re-shaped into a one-dimensional array to be usable from the modules of the Scikit-Learn library. The similarity matrix needed to apply the MDS algorithm is computed from the output layer in array shape, and the MDS module is initialized. After a few preprocessing operations, the coordinates of neurons on the scatter plot are computed by the MDS module, and the coordinates are passed to the matplotlib "scatter" function. Colors associated to each scatter are obtained from the norm of the weight vector of the corresponding neuron, as in the previous case. After the scatters have been placed in the plot, the labels associated to the players to be displayed are shown in the correct positions. Figure 4.4 is an example of result of the computation, which is much better than the one obtained with the "load_visualization_scatter" method, but still not the optimal solution for the goal of clusters definition.

The final visualization solution is implemented in the "load_sammon_scatter" method. The only required parameter for the method is the "show_limit" value already used in the other methods, but another additional non-required parameter can be specified in this case. The parameter is called "mapped_coord_path" and allows to specify the path of a file where the method can save the coordinates of the mapped players. The default value of the parameter is "None", but if a different value is assigned to it, the method saves the coordinates of the scatters associated to the mapped players into the specified file. The saving of the coordinates of the players in the final scatter plot can be useful to perform some additional analysis on the relative distance between players, which is computed to resemble as best as possible the difference between two players and their way of playing. The method firstly computes the positions of the input set elements on the output layer, through the SOM forward-mapping operation, then the coordinates of scatters to be placed in the plot has to be computed too. In this case the Sammon MDS algorithm is exploited for the computation, thanks to the Sammon external module described in Section 4.1.4. The method calls the external module that returns the coordinates of neurons to be placed in the output scatter plot. The color associated to neurons is once again obtained from the computation of the norm of the weight vector associated to each neuron, and the result of the neurons distribution is displayed thanks to the "scatter" matplotlib function. Labels related to the mapped players

are shown as in the previous cases, taking into account the value of the "show_limit" parameter. Figure 4.3 reports an example of the result, which is the optimal one found for the visualization problem. This visualization is the one that allows to better distinguish clusters of players on the map and perform additional analysis on their positions, focusing on the relative distance between players and their absolute position in the scatter plot.

4.2.4 Basketball SOM

The "BasketballSOM" class is the core of the application, since it implements all the methods and the calls to external modules that allow to perform the whole SOM algorithm and present the results. The class has to be initialized specifying some parameters that are fundamental for the execution of the whole process. In particular, for the initialization of the SOM object, these parameters are:

- l. The size of the output lattice, since it will be a squared grid, one integer number is enough to express its dimension.
- *map_dim*. The tuple that expresses the real dimension of the output layer of the map, $(l \times l)$.
- *n_iter*. The number of iterations to be performed during the training algorithm.
- *learn*. The initial learning rate to be used in the training process.
- *stat_cat*. The statistical categories to be considered for the SOM process. This parameter is an array of small strings, referring to the labels used in the data source CSV file to express the meaning of data. Filtering on the labels assigned to columns it is possible to retrieve only data related to the statistical categories we are interested in.
- *neuron_size*. It expresses the dimension of each weight vector associated to a neuron in the map. The number of weights associated to each neuron is equal to the number of statistical categories used for the map characterization. Each input set element is characterized by a different value for each statistical category and the neuron weight vectors must have the same size of the elements of the input set, as explained in Section 3.1.2.
- *db_path*. The path of the data source file, containing all the statistics and data of the input space.
- *train*. The path of the training set file to be used to create the training set.
- *input*. The path of the input set file to be used to create the input set.
- *neigh_rad*. The only non-required parameter, used to express the initial value of the neighborhood radius to be used in the training phase. If no particular value is assigned to this parameter, the initial value of the neighborhood radius is set to the default value specified in the SOM module of the PyMVPA library.

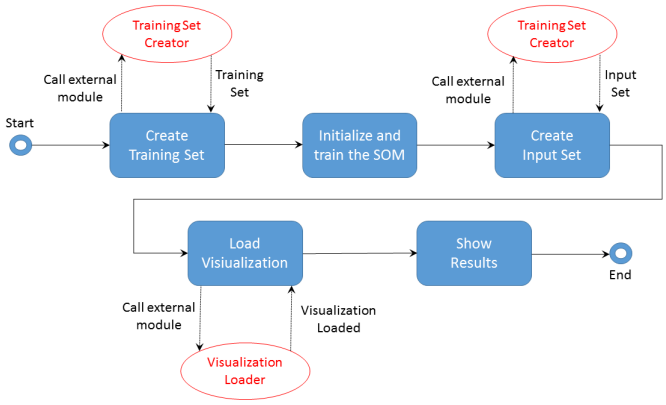

Figure 4.7: Scheme describing the complete workflow of the "computeSOM" method.

The class contains only two methods. One of them is the main method that performs the whole algorithm, the other is a supporting method, called from the most important one to perform some additional preprocessing computation needed to let the algorithm work.

The supporting method is called "normalize_list_of_lists", and takes a list of lists as unique input parameter. The list of lists in question is one of the two datasets used in the application, it can be the training set or the input set. Both sets are created retrieving statistics of the interested players from the proper data source, so the values contained in those lists of arrays may have a big variability. Each statistical category, from the PPG to the number of rebounds grabbed per game or the number of assists made per game, may vary in a range of values that usually goes from 0 to 50 most of the time.

The SOM algorithm can perform a correct clustering with any kind of input values coming from the input space, but it performs better if the values of the datasets involved in the process are between 0 and 1; this is the reason why the "normalize_list_of_lists" method was implemented. The method performs a normalization of the dataset given as input to it, transforming all the values in a range between 0 and 1, but keeping the relative distance between them, to feed the SOM module with a dataset that can optimize the clustering performances. Any time a dataset is created through the dedicated external modules, it is normalized before being given as an input to the SOM module for the computations.

The most important method of the whole application is the "computeSOM" method, which performs all the computations and external calls needed to make the application work. There is no required parameter to be passed to it, but some optional parameters can be set to particular values to customize its behavior.

The workflow of the method implementing the algorithm is shown in Figure 4.7. It can be divided in 5 steps:

1. Creation of the training set. The creation of the training set is the first step of the work-flow. A "TrainingSetCreator" object is instantiated, passing the required parameters described in Section 4.2.2 to it. The proper method to

obtain the training set is called and the returned dataset is normalized using the "normalize_list_of_lists" method.

2. Creation and training of the SOM. The SOM object of the PyMVPA library is initialized specifying the required parameters, as described in Section 4.1.1 and the training of the map is performed immediately after the creation of the new instance, feeding the object with the training set computed at the previous point.

3. Creation of the input set. An "InputSetCreator" object like the one described in Section 4.2.2 is instantiated, passing the required parameters to it. The correct method call to be performed from the "InputSetCreator" instance depends on two optional parameters of the "computeSOM" method. The "evolution" parameter can be set to a value different from 0 if the aim of the algorithm that is being executed is to show the evolution of the position of a player through different seasons. In this case the "evolution_plot_input" method of the "InputSetCreator" is called and the returned input set is normalized with the "normalize_list_of_lists" method. If the value of the "euroleague_path" parameter, which reports the path of the input set file containing names of European players, is different from "None", the "euroleague_plot_input" method of the "InputSetCreator" is called and the retrieved input set is normalized as usual. Finally, in the case none of the two parameters is set to a value different from its default, the standard "single_plot_input" is called to retrieve the input set to be normalized.

4. Load Visualization. The data visualization phase of the process starts with the initialization of the "VisualizationLoader" object, passing to it all the required parameters described in Section 4.2.3. According to the purpose of the process that is being executed, the proper visualization method is called to obtain the desired result.

5. Show Results. Finally the result of the whole process is displayed in an interactive window, calling the "show" method of matplotlib. The interactive widow allows to analyze the result in detail, moving the graph into the frame or zooming the visualization on some interesting area of the graph. Screenshots of the results can be saved in any image format.

The implementation of the whole architecture around the "BasketballSOM" core class allows to perform the whole process with just one method call. In a main script used to launch the application it is possible to set all the parameters that are needed for the execution of the process, instantiate the "BasketballSOM" class and launch the "computeSOM" method to perform the whole process. With just two simple operations the whole application runs according to the specified parameters. The use of the object-oriented paradigm and the separation of different modules for different functionalities allows to change the behavior of the external modules without changing the main core class. The addition of new external modules is possible in any moment, adding the proper calls in the main algorithm. The use of this architectural design makes the application more flexible and open to future expansions or changes.

Chapter 5

Training Set Characterization

The choice of the training set is a very important task when using the SOM. Indeed the training set used to train the network defines the clusters that will be found in the output layer after the training of the SOM. These clusters will be used later to understand to which category some elements of the input space take belong. During the experiments, different training sets have been used to explore different results from the SOM clustering. The main source of data used for the experimental validation of this work was the NBA database of statistics. In particular, the experiments discussed in this chapter, which were done to find the feature of a good training set for the map, were based on data referring to the 2015-2016 NBA regular season. The season includes 82 games played by the 30 teams in the league, involving 476 players.

5.1 Five Positions Training Set

The first trial was made by training the SOM using a set made by just five elements, one for each of the "classic" positions of basketball: Point Guard, Shooting Guard, Small Forward, Power Forward and Center. Starting from the 2015-2016 NBA regular season database, the records have been divided in five groups according to the positions. Once determined the five groups of records corresponding to the five positions, a single array has been created using the data belonging to these groups of records, containing for each statistical category the average of that category computed on all the players of that position. Actually each element of the training set describes the season average parameters of a player, that are obtained by computing the average of each statistical field, for all the players of the same position. Training the SOM with this kind of training set it was possible to obtain five clusters in the output layer.

Afterwards, some players have been mapped on the output layer of the SOM to test whether they were mapped in the cluster corresponding to the position they were supposed to be mapped, as shown by the results in Section 6.1.

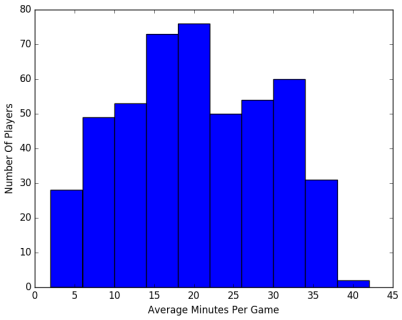

Figure 5.1: Players Average Minutes Played Per Game

5.2 New Positions Training Sets

Since the five classical positions of basketball have been found to poorly describe the way nowadays players play the game, further investigations were made using more sophisticated training sets.

To provide a new classification of positions of basketball, a new training set had to be chosen. This training set shall be used to define the clusters on the output layer of the SOM. Some trials have been made to find the training set that better describes the different ways of playing in modern basketball.

5.2.1 All Players Training Set

The first trial was made by including in the training set all the players of the NBA, to be sure to consider all the possible facets of the game.

At this point a problem raises: comparing statistics about season average numbers related to players who played a different amount of minutes per game may lead to errors in the evaluation. Season average statistics are computed in a way that does not take into account the number of minutes played.

Total statistics referring to each statistical category (points, rebounds, assists, blocks, etc.) of each game played by a player are summed, and then divided by the number of games played during the entire seasons, to find the average value of the specific statistical category across the different games of the season. For example, PPG are computed as:

$$PPG = \frac{\sum_{n=1}^{Ngames} PTS(n)}{Ngames} \tag{5.1}$$

Where "PTS(n)" is the number of points scored by the player in the n-th game of the season, while "Ngames" is the number of games played by the player during the season. This is also done for the number of minutes played, since the season MPG are computed by summing the minutes played by the player in each game and dividing the sum by the number of games played during the season. Using these values, a meaningful comparison among statistics of players can not be done without considering MPG, since it is obvious that a player who played more minutes per game has more possibilities to improve his statistical performance during the game.

Directly including the MPG as a modeling variable in the system, i.e., by setting this value as an input to the SOM, may not be a useful solution. In fact, using the minutes played as one of the variables will influence the clustering just like all the other variables, while MPG should somehow change the way we evaluate all the other statistics. For example, if a player has statistics that are in average with respect to the statistics of the training set, but a few minutes played per game, his statistics should be considered to be more valuable with respect to the ones of a player who has higher average statistics but played a lot of minutes per game. Bigger values in the statistics, i.e., better performance, in shorter time should be considered as a very positive aspect in the evaluation of a player.

To be comparable, all the statistics have been normalized by the minutes played by the players, using the typical normalization over 48 minutes, which consists in dividing all the other average statistics by the MPG, and then multiply the result by 48. For example, the PPG statistic is normalized over 48 minutes as follows:

$$normalizedPPG = \frac{PPG}{MPG} * 48 \tag{5.2}$$

This normalization allows to consider the average statistics of a player as he would have played 48 minutes in every game, so as he always played the whole game, that in the NBA lasts 48 minutes. This technique is already in use in basketball analytics, and in theory it allows to compare the impact of players that may have different impacts due to the different amount of time played in each game. In practice, however, there could be significant statistical distortions. In particular, a player may have played a very few minutes during the season, maybe in situations when the result of the game was no more in discussion. In this case, his statistics normalized on a 48 minutes base may be stretched too much. He may result to have statistics similar to the ones of an all-star player, while this is only due to the statistical distortion introduced by the normalization. Anyway, a normalization on the number of minutes played is necessary for our purpose, since the statistics of players with different average minutes played can not be compared in a meaningful manner. It is obvious indeed that a player who played less minutes will have lower averages in all the statistical categories, with respect to a player who averaged more minutes per game. Actually this problem was reflected in the training of the SOM made using all the NBA players as a training set. The obtained clusters were not well defined and big areas of the map were not corresponding to any player.

To face the problem, a lower limit on the average minutes played during the season by a player was set to make him "eligible" for the training set. To decide the value of this lower threshold, an analysis on the minutes played by all the players was done, plotting an histogram that shows the number of players averaging a different amount of minutes played during the season. Figure 5.1 shows the result of this analysis. From the figure it is clear how most of the players average from 10 to 30 minutes per game. From some tests on the SOM training phase, the lower threshold value was put at 10 minutes per game. In this way, all the players averaging less than 10 minutes played per game are excluded from any training set. This decision actually makes 77 players ineligible for any training set, over the total number of players, which is 476. So the 16.1% of the players in our database will not be used in any training set. Once the lower threshold of average minutes per game has been

decided other types of training sets had to be created.

The restriction on minutes played was not the only feature that was used to create the best possible training set for the definition of the new positions, so different trials were made.

5.2.2 Starting Five Training Set

The first trial was made by creating a dataset which contains only the starting five players for each of the 30 teams of the league, for a total of 150 players in the training set. The selection of the correct five players for each team was made according to the data given by the league site, which tells for each player the number of games played, and the number of games started during the 2015-2016 regular season. The five players with the higher number of games started for each team were inserted in the training set shown in Table 5.1.

Table 5.1: Starting Five Training Set

TEAM	PLAYERS				
ATL	Al Horford	Paul Millsap	Kent Bazemore	Kyle Korver	Jeff Teague
BOS	Amir Johnson	Jared Sullinger	Jae Crowder	Avery Bradley	Isaiah Thomas
BRK	Brook Lopez	Thaddeus Young	Joe Johnson	Bojan Bogdanovic	Jarrett Jack
CHI	Cody Zeller	Marvin Williams	Nicolas Batum	Courtney Lee	Kemba Walker
CHO	Pau Gasol	Taj Gibson	Nikola Mirotic	Jimmy Butler	Derrick Rose
CLE	Tristan Thompson	Kevin Love	LeBron James	J.R. Smith	Kyrie Irving
DAL	Zaza Pachulia	Dirk Nowitzki	Chandler Parsons	Wesley Matthews	Deron Williams
DEN	Nikola Jokic	Kenneth Faried	Danilo Gallinari	Gary Harris	Emmanuel Mudiay
DET	Andre Drummond	Marcus Morris	Tobias Harris	Kentavious Caldwell-Pope	Reggie Jackson
GSW	Andrew Bogut	Draymond Green	Harrison Barnes	Klay Thompson	Stephen Curry
HOU	Dwight Howard	Donatas Motiejunas	Trevor Ariza	James Harden	Patrick Beverley
IND	Ian Mahinmi	Myles Turner	Paul George	Monta Ellis	George Hill
LAC	DeAndre Jordan	Blake Griffin	Luc Mbah-a-Moute	J.J. Redick	Chris Paul
LAL	Roy Hibbert	Julius Randle	Kobe Bryant	Jordan Clarkson	DAngelo Russell
MEM	Marc Gasol	Zach Randolph	Matt Barnes	Tony Allen	Mike Conley
MIA	Hassan Whiteside	Chris Bosh	Luol Deng	Dwyane Wade	Goran Dragic
MIL	Greg Monroe	Jabari Parker	Giannis Antetokounmpo	Khris Middleton	Michael Carter-Williams
MIN	Karl-Anthony Towns	Gorgui Dieng	Andrew Wiggins	Zach LaVine	Ricky Rubio
NOP	Omer Asik	Anthony Davis	Alonzo Gee	Eric Gordon	Jrue Holiday
NYK	Robin Lopez	Kristaps Porzingis	Carmelo Anthony	Arron Afflalo	Jose Calderon
OKC	Steven Adams	Serge Ibaka	Kevin Durant	Andre Roberson	Russell Westbrook
ORL	Nikola Vucevic	Aaron Gordon	Victor Oladipo	Evan Fournier	Elfrid Payton
PHI	Jahlil Okafor	Nerlens Noel	Robert Covington	Jerami Grant	Ish Smith
PHO	Tyson Chandler	Alex Len	P.J. Tucker	Devin Booker	Brandon Knight
POR	Mason Plumlee	Ed Davis	Al-Farouq Aminu	C.J. McCollum	Damian Lillard
SAC	Willie Cauley-Stein	DeMarcus Cousins	Rudy Gay	Ben McLemore	Rajon Rondo
SAS	Tim Duncan	LaMarcus Aldridge	Kawhi Leonard	Danny Green	Tony Parker
TOR	Jonas Valanciunas	Patrick Patterson	DeMarre Carroll	DeMar DeRozan	Kyle Lowry
UTA	Rudy Gobert	Derrick Favors	Gordon Hayward	Rodney Hood	Raul Neto
WAS	Marcin Gortat	Otto Porter	Jared Dudley	Bradley Beal	John Wal

The results of the clustering obtained with this training set were not satisfactory, mainly for two reasons:

1. considering a set of 150 players, we are considering a subset of all the players in the league which is considered too small;

2. in most of the teams, the players who actually characterize the way the team plays are not necessarily the ones of the starting five. Many teams, especially the winning ones, have an higher number of players who give a good contribution to the team, playing a significant amount of minutes per game, even if some of

them is not playing the game from the beginning, which means their success is also based on the help that usually two or three players starting from the bench can provide to the team. Usually, these "key players" starting the game in the bench, play in average more than ten minutes per game, thus they are also eligible for the training set according to the threshold that has been set in Section 5.2.1.

This training set gives better results in the clustering with respect to the training set containing all the players, but the result in not clear enough for the purpose of this work, since the new positions do not show up clearly in the map, and some very famous players are mapped in positions that do not describe their style very well.

5.2.3 Key Players Training Set

As reported in Section 5.2.2, the training set composed by the five starting players of each team did not give good results in terms of clustering, and one of the reasons was that many teams in the NBA do not build their success only on the players belonging to the starting five, but also on the help coming from players starting from the bench. Considering this fact, a new training set was created.

In this case the 8 players with the highest number of average minutes played per game, for each of the 30 teams, were inserted in the set. Usually indeed the starting five players are the ones that average the highest number of minutes, but sometimes the players coming from the bench may play more minutes than some starters, giving a very important help to the team. Considering the 8 players with the highest averaged minutes per game we are able to consider all the players that actually play most of the minutes of the team. Searching in the NBA database for the 8 players with the highest average minutes per game for each team, the dataset shown in Table 5.2 was created.

Table 5.2: Key Players Training Set

TEAM	PLAYERS							
ATL	Al Horford	Paul Millsap	Kent Bazemore	Kyle Korver	Jeff Teague	Thabo Sefolosha	Dennis Schroder	Tim Hardaway
BOS	Amir Johnson	Jared Sullinger	Jae Crowder	Avery Bradley	Isaiah Thomas	Marcus Smart	Kelly Olynyk	Evan Turner
BRK	Brook Lopez	Thaddeus Young	Joe Johnson	Bojan Bogdanovic	Jarrett Jack	Wayne Ellington	Donald Sloan	Shane Larkin
CHI	Cody Zeller	Marvin Williams	Nicolas Batum	Courtney Lee	Kemba Walker	Al Jefferson	Jeremy Lin	Michael Kidd-Gilchrist
CHO	Pau Gasol	Taj Gibson	Nikola Mirotic	Jimmy Butler	Derrick Rose	Doug McDermott	Joakim Noah	Mike Dunleavy
CLE	Tristan Thompson	Kevin Love	LeBron James	J.R. Smith	Kyrie Irving	Matthew Dellavedova	Iman Shumpert	Mo Williams
DAL	Zaza Pachulia	Dirk Nowitzki	Chandler Parsons	Wesley Matthews	Deron Williams	Raymond Felton	J.J. Barea	Devin Harris
DEN	Nikola Jokic	Kenneth Faried	Danilo Gallinari	Gary Harris	Emmanuel Mudiay	Will Barton	Darrell Arthur	D.J. Augustin
DET	Andre Drummond	Marcus Morris	Tobias Harris	Kentavious Caldwell-Pope	Reggie Jackson	Ersan Ilyasova	Stanley Johnson	Anthony Tolliver
GSW	Andrew Bogut	Draymond Green	Harrison Barnes	Klay Thompson	Stephen Curry	Andre Iguodala	Festus Ezeli	Shaun Livingston
HOU	Dwight Howard	Donatas Motiejunas	Trevor Ariza	James Harden	Patrick Beverley	Terrence Jones	Corey Brewer	Josh Smith
IND	Ian Mahinmi	Myles Turner	Paul George	Monta Ellis	George Hill	Rodney Stuckey	Jordan Hill	C.J. Miles
LAC	DeAndre Jordan	Blake Griffin	Jeff Green	J.J. Redick	Chris Paul	Jamal Crawford	Austin Rivers	Paul Pierce
LAL	Roy Hibbert	Julius Randle	Kobe Bryant	Jordan Clarkson	DAngelo Russell	Lou Williams	Larry NanceJr.	Brandon Bass
MEM	Marc Gasol	Zach Randolph	Matt Barnes	Tony Allen	Mike Conley	Lance Stephenson	Jordan Farmar	Mario Chalmers
MIA	Hassan Whiteside	Chris Bosh	Luol Deng	Dwyane Wade	Goran Dragic	Justise Winslow	Josh Richardson	Gerald Green
MIL	Greg Monroe	Jabari Parker	Giannis Antetokounmpo	Khris Middleton	Michael Carter-Williams	Jerryd Bayless	O.J. Mayo	Greivis Vasquez
MIN	Karl-Anthony Towns	Gorgui Dieng	Andrew Wiggins	Zach LaVine	Ricky Rubio	Shabazz Muhammad	Tayshaun Prince	Nemanja Bjelica
NOP	Omer Asik	Anthony Davis	Alonzo Gee	Eric Gordon	Jrue Holiday	Tyreke Evans	Dante Cunningham	Norris Cole
NYK	Robin Lopez	Kristaps Porzingis	Carmelo Anthony	Arron Afflalo	Jose Calderon	Langston Galloway	Lance Thomas	Derrick Williams
OKC	Steven Adams	Serge Ibaka	Kevin Durant	Andre Roberson	Russell Westbrook	Enes Kanter	Randy Foye	Dion Waiters
ORL	Nikola Vucevic	Aaron Gordon	Victor Oladipo	Evan Fournier	Elfrid Payton	C.J. Watson	Brandon Jennings	Mario Hezonja
PHI	Jahlil Okafor	Nerlens Noel	Robert Covington	Jerami Grant	Ish Smith	Hollis Thompson	Isaiah Canaan	Nik Stauskas
PHO	Tyson Chandler	Alex Len	P.J. Tucker	Devin Booker	Brandon Knight	Eric Bledsoe	T.J. Warren	Mirza Teletovic
POR	Mason Plumlee	Ed Davis	Al-Farouq Aminu	C.J. McCollum	Damian Lillard	Allen Crabbe	Gerald Henderson	Meyers Leonard
SAC	Willie Cauley-Stein	DeMarcus Cousins	Rudy Gay	Ben McLemore	Rajon Rondo	Darren Collison	Omri Casspi	Marco Belinelli
SAS	Tim Duncan	LaMarcus Aldridge	Kawhi Leonard	Danny Green	Tony Parker	Patrick Mills	Manu Ginobili	Boris Diaw
TOR	Jonas Valanciunas	Patrick Patterson	DeMarre Carroll	DeMar DeRozan	Kyle Lowry	Cory Joseph	Terrence Ross	Bismack Biyombo
UTA	Rudy Gobert	Derrick Favors	Gordon Hayward	Rodney Hood	Raul Neto	Alec Burks	Trey Burke	Trevor Booker
WAS	Marcin Gortat	Otto Porter	Jared Dudley	Bradley Beal	John Wall	Markieff Morris	Garrett Temple	Ramon Session

Comparing the new training set with the previous one, it is immediately clear how this is actually an expansion of the previous one, as the starting players of each team are included in the eight players averaging the highest number of minutes

played per game. This new training set reflects better the true core of each team, made by the five starters and some key players coming from the bench. The training of the SOM with this dataset produced results that are much better in terms of clustering and clusters definition, as shown by the results reported in Chapter **??**.

5.2.4 Position Based Training Set

The last experiment was made considering the five classic positions of basketball. The main idea behind this trial is that these positions are not suitable for the different ways players play the game nowadays, and so we want to define new ones. In particular it is known how, between the set of players that are labeled to be playing a certain position, there are different ways of playing the same position by different players. For example, between the point guards there are some players who are very good in the the scoring part of the game, some others who are worse scorers but better passers, and some others who are worse offensive players but better defensive players. We can make similar distinctions between the players labeled in each of the five classic positions. This last training set is created taking a certain number of players from the group of players labeled in each of the five classical positions: some of the top point guards in the league, some of the top shooting guards, some of the top small forwards, some of the top power forwards and some of the top centers.

To obtain a suitable training set, there are 2 problems to solve:

1. Select the correct features or criteria to be used to order the players and select the top ones with respect to the selected criteria.

2. Selecting the correct number of players for each position, in order to create a proper training set.

To solve the first problem two different training sets were created, sorting players according to two different statistical categories: the MPG and the PPG. Very often the top players in a team or in a league are considered to be the ones that score more points per game, even if the value of a player should not be measured only by his scoring ability. Usually, the players who score more points are also the ones who play more minutes for their team. As regards the number of players per role to be inserted in the training set, it was decided to select 30 players per position, equal to the number of teams in the league.

As a first analysis, a comparison was made between the two training sets, obtained selecting the top 30 scoring players for each position and the top 30 players for each position based on the average minutes per game. These 2 training sets are made by the same number of players, since in both cases the first 30 players from 5 different sorted lists were selected, so the training sets are made by 150 players each. A small script was used to perform the comparison after the creation of the training sets, in order to see the number of players that take part to both of them. The result of the comparison showed that 129 players are part of both training sets, over the 150 total in each of them. So the 86% of players are in common between the two sets.

To verify whether the number of players taken for each position influences this percentage or not, the same training sets was formed by taking the top 40 players for

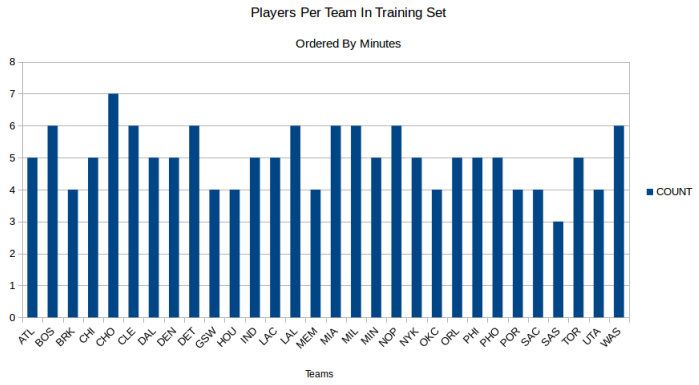

Figure 5.2: Players per team in the training set, ordered by minutes played

each position, both sorted by points per game and minutes per game. In this case each training set is made by 200 players. The number of players in common between the two sets is 174, the 87% of the total. The percentage of elements in common between the two training sets is almost the same of the previous case, thus the number of players selected by position is proved to not influence it so much. Increasing the number of considered players by position even more, the percentage actually increases too. However, the increment is negligible, since there is an increment of 1% by adding 10 players by position in the training set. Therefore, to have much better percentages we should consider a lot more players, but in this case this is not our purpose, since adding more players will make the training set lose its main feature, which is to be made by the best players for each position.

Another analysis made on the two training set consists in counting the number of players coming from each of the 30 teams in the league that are included in the training set. Since the final goal of this approach is to obtain a training set that well describes the way all the players in the league play, it is important that every team in the league is represented with some players in the training set. In the first experiment the two training sets considered in the analysis were built taking the top 30 players for each position, both sorted by minutes and points per game.

The results concerning the number of teams represented in the two training sets are shown in Figure 5.2 and Figure 5.3.

Figures 5.2 and 5.3 show how, taking the top 30 players for each position, all the 30 teams of the league have some players included in the training set, both sorting them by the average points per game and by the average minutes played per game. Looking at the two graphs it can be seen how the training set created by ordering players according to the minutes played has a lower variability in the number of players coming from each team. From the graph in Figure 5.2 it can be seen how only one team has 7 players in the training set, and only one team is represented by only 3 players in the set. In the remaining 28 teams there are 8 teams with 4 players in the set, 8 teams represented by 6 players and 12 teams providing 5 players to the set. There is actually a good balance in the distribution between the 30 teams in the league, which all contribute to the construction of the training set with almost the same "effort". Ordering the players on the basis of the averaged points per game

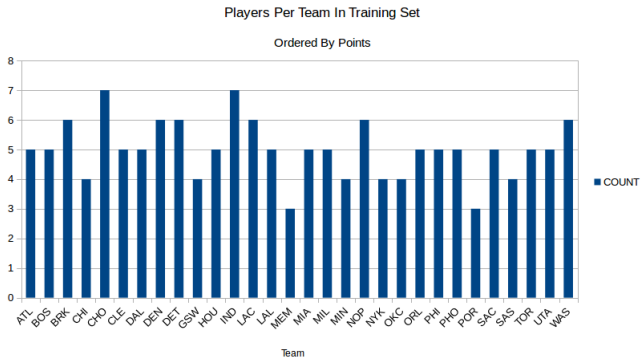

Figure 5.3: Players per team in the training set, ordered by points scored

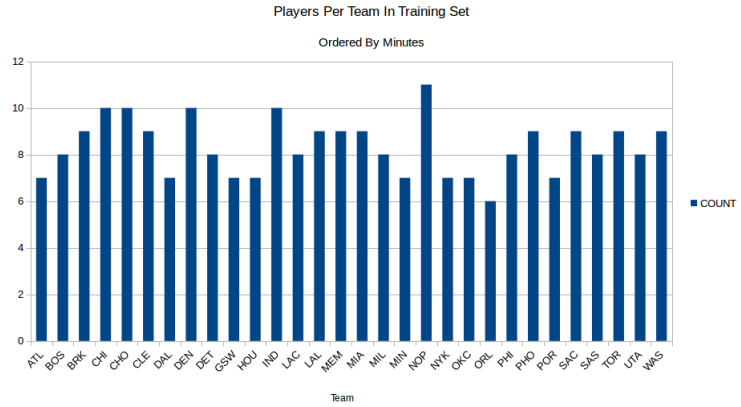

Figure 5.4: Players per team in the training set, ordered by minutes played

there is a different distribution in the training set. In the case shown in Figure 5.3 there are two teams that provide 7 players to the training set each, and two teams providing only 3 players to the set. Moreover there are 6 teams proving 4 players each and 6 teams providing 6 players each. Finally most of the teams, 14 over the total 30, provide 5 players each to the training set.

Obviously the average number of players per team in the training set is always 5, but ordering the players according to the averaged points per game achieves a greater variability in the number of players coming from each single team.

To further confirm this fact, the same training sets were built taking the top 50 players for each position, ordered according to the two criteria, and the same analysis was made to see how the 30 teams are represented in the training set.

Figure 5.4 and 5.5 report the histograms showing the team distribution in the two cases.

Taking the top 50 players for each position, it can be seen how the training set created by ordering the players according to the average minutes played per game have the smaller variability in the team distribution, as there is a lower number of teams that provide a number of players that is far from the expected average number of players per team, that in this case is between 8 and 9 players.

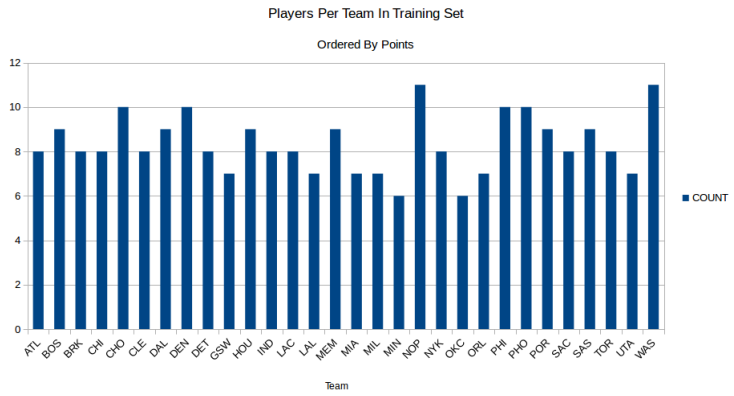

Figure 5.5: Players per team in the training set, ordered by points scored

The reason for the higher variability in the points-per-game-ordered training set may be in the way the different teams of the league play.

The teams that provide the lower number of players in the training set are for example the Portland Trail Blazers, or the Oklahoma City Thunder. A common feature of these two teams is that most of the points scored by the team are scored by the two most important players of the roster. The Portland back-court duo made by Damian Lillard and C.J. McCollum provided most of the team points during the whole season, while the Kevin Durant - Russell Westbrook duo in Oklahoma City is considered to be one of the best scoring couples in NBA history. Lilliard and McCollum combined for 45.9 points per game during the regular season, while the whole Portland Trail Blazer team averaged 105.1 points per game. Therefore, the two most important players of the team scored 43.7% of the team points during the regular season. Durant and Westbrook combined for 51.7 average points per game during the season, while the whole Oklahoma City team averaged 110.2 points per game. In this case the two players combined for the 46.9% of the average points of their team.

On the other side, teams that do not base their scoring only on a few players, provided the higher number of players in the training set, since in these teams there are more players who average a good number of points per game and the scoring statistics are more distributed between all of them. In this category of teams there are for example the Charlotte Hornets, which had 6 players averaging double figure[1] in scoring during the whole regular season.

On the basis of the above considerations, it is not possible to say that a better scoring distribution can give the team better results, since actually in the last season the Oklahoma City Thunder had a better record with respect to the Charlotte Hornets. This kind of considerations are out of the scope of this work.

As a conclusion of the analysis it was decided that the average minutes played per game is the best feature to be used to sort the players within a position, to create a training set that is made by players who can represent all the teams in the league in the best possible way. Regarding the number of players to be selected for

[1]The term "averaging double figure" in scoring means scoring an average of more than 10 points per game. The PPG statistic of a player scoring in double figure has 2 figures.

each position, the initial option of selecting 30 players per position turns out to be the best one. In fact, increasing too much the number of players per positions does not significantly affect change the features of the training set. Moreover, the goal of taking the best players in the NBA to characterize the training set is achieved by considering 30 players per position.

Chapter 6

Results

In this chapter all the results obtained from different experiments made during the thesis will be presented and explained in details.

Firstly, an analysis on classic basketball positions has been made, to prove how they are not suitable for describing the way modern basketball players play the game. The next step was the creation of new positions, based on the use of proper training sets, used to train the SOM and create automatic clusters that show how players are similar or different between each other. Once the new positions have been defined, the same analysis has been performed on previous seasons, from the 2010-2011 regular season, until the last season (2015-2016) on which the whole work is based. Since data from previous seasons have been retrieved, a further experiment was made to see how some properly selected players changed their way of playing through the years, and so changed their position according to this evolution.

All the listed experiments were made considering the 7 basic statistical categories that were used by Muthu Alagappan to define his new positions, as explained in Section 2.3. To complete our analysis, a new expanded set of statistical categories was created, made by 11 different statistics, to see if a more complete statistical description of players could provide a better definition of positions, and so a better clustering of players.

Finally the SOM algorithm for clustering players and creating new positions was applied to a different dataset, made by players from the Euroleague, the most important European basketball competition. The aim of this last experiment was to see if the approach can be used as a universal tool to create new positions for players of any league in the world.

6.1 Classic Positions Analysis

Before starting with the definition of the new positions, the traditional basketball positions have been analyzed to see how they are not able to describe modern players game anymore.

The proper training set to perform this analysis was created as explained in Section 5.1 and was composed by 5 elements, one for each position: Point Guard, Shooting Guard, Small Forward, Power Forward and Center. After the training process, the output layer of the SOM was divided in 5 clusters, each of them

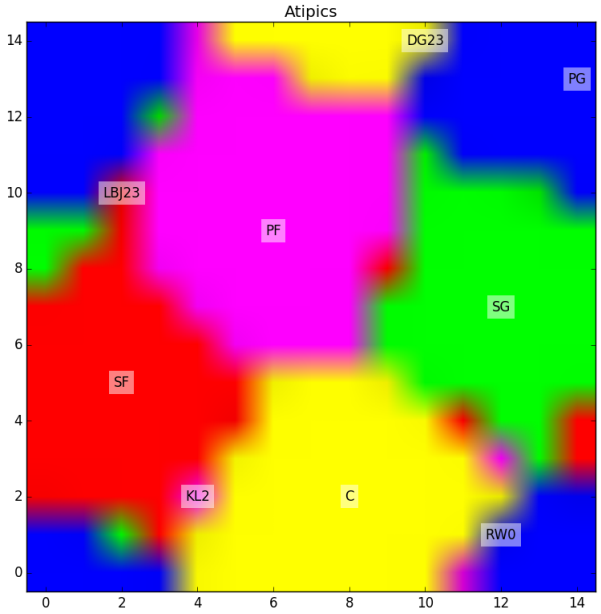

Figure 6.1: Forward mapping of some atypical players on the 5 positions trained SOM

corresponding to one position. A color was assigned to each position and also to each neuron. The color to be assigned to a neuron is selected according to its similarity with the elements of the training set. Since each element of the training set is a reference to a particular position, the color of the neuron will be one of the 5 colors assigned to the 5 positions.

The center of each cluster obtained on the output layer was the position in which the elements of the training set were mapped, so applying the forward mapping process feeding the SOM with the statistical profiles of some particular players, it is possible to see how they are placed in the map with respect to the 5 classic positions.

Firstly, some of the most atypical players in the league were mapped on the SOM. These players are some of the ones whose position have been largely debated during the last seasons, since, even if they are labeled to be playing a certain position, their way of playing that position is different from all the other players in the league. The selected players are LeBron James, Kawhi Leonard, Draymond Green and Russell Westbrook, their statistics are reported in Table 6.1.

Table 6.1: Atypical players statistics

PLAYER	TRB	BLK	AST	STL	TOV	PF	PTS
LeBron James	7.4	0.6	6.8	1.4	3.3	1.9	25.3
Kawhi Leonard	6.8	1.0	2.6	1.8	1.5	1.8	21.2
Russell Westbrook	7.8	0.3	10.4	2.0	4.3	2.5	23.5
Draymond Green	9.5	1.4	7.4	1.5	3.2	3.0	14.0

LeBron James is one of the most famous basketball players in the world, and he is considered to be a small forward. Actually he is taller and bigger than most of the other small forwards in the league, he can score like a shooting guard, pass the ball like a point guard, and grab rebounds like a power forward, and actually he played any of these positions in some particular situations during his career. Kawhi Leonard is considered a small forward too, but his rebounding, scoring and defensive skills, make him a unique player, and the future star of the San Antonio Spurs. Draymond Green is probably the most atypical of atypical players. His case was discussed in Section 2.2, since he can do almost anything on a basketball court and he was defined as the epitome of the position-less basketball player. Finally Russell Westbrook is labeled to be a point guard, but he is probably the best athlete who ever played the position in basketball history. He made 18 triple doubles[1] during the last regular season, tying the previous record made by Magic Johnson in the 1981-1982 regular season. A point guard is supposed to be a good scorer and a good assist-man, but not such a rebounder and shot blocker as Westbrook is.

Figure 6.1 shows the result of the forward mapping of the atypical players on the output layer of the SOM and it is immediately clear how classic positions cannot describe the way these players play.

LeBron James (LBJ23) is mapped in a position which is between the red area referring to small forward players, the purple area of power forwards, the blue area of point guards and the green area referring to shooting guards. Actually he is halfway between all the positions we cited previously to describe his way of playing.

Kawhi Leonard (KL2) is mapped between the small forward red area and the yellow area referring to centers, he is considered to be a small forward, but his rebounding and defensive skills place him close to the center cluster.

Russell Westbrook (RW0) and Draymond Green (DG23) are in the peripheral area of the point guard position, the blue area, but they are on the edge which is close to the center position. Actually for the way classic basketball positions are thought to be, point guards and centers should be two opposite types of players, with completely different responsibilities on the floor, and so completely different ways of playing the game, as explained in Section 1.1.

The fact that these players are labeled to be half-way between a point-guard and a center shows how the classic positions are not able to describe their way of playing anymore.

A further experiment made on the SOM clustered on the base of the 5 classic positions was made selecting some players labeled to be playing the same position, and forward-mapping them on the output layer. The idea is to show how the way of playing the same position can change a lot from one player to another.

The first trial was made by mapping some of the most famous power forwards in the league: Kevin Love, Paul Millsap, Blake Griffin, Anthony Davis, LaMarcus Aldridge and Dirk Nowitzki, whose statistics are reported in Table 6.2.

[1] A triple double happens when a player records a value that is greater or equal to 10 in three different statistical fields during one game. It means for example that in one game a player has to score at least 10 points, grab at least 10 rebounds and give at least 10 assists to make a triple double. A triple double can be made by any triplet of statistical categories, even blocks, rebounds and assists for example.

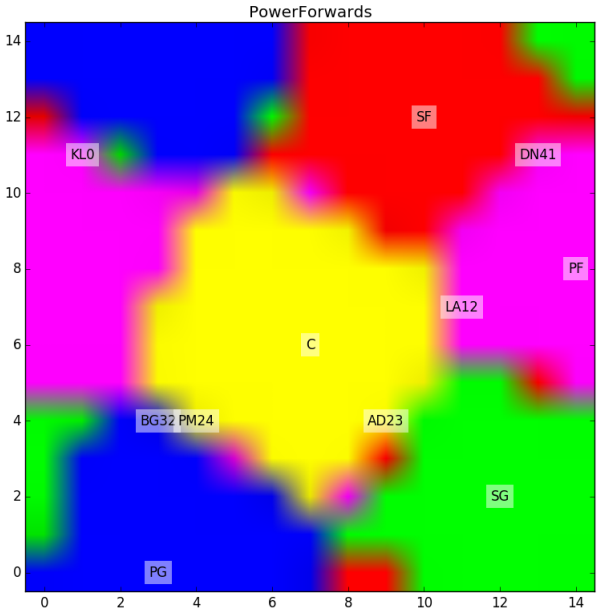

Figure 6.2: Forward mapping of some power forwards on the 5 positions trained SOM

Table 6.2: Power Forwards statistics

PLAYER	TRB	BLK	AST	STL	TOV	PF	PTS
Kevin Love	9.9	0.5	2.4	0.8	1.8	2.1	16.0
Paul Millsap	9.0	1.7	3.3	1.8	2.4	2.9	17.1
Blake Griffin	8.4	0.5	4.9	0.8	2.4	2.7	21.4
Anthony Davis	10.3	2.0	1.9	1.3	2.0	2.4	24.3
LaMarcus Aldridge	8.5	1.1	1.5	0.5	1.3	2.0	18.0
Dirk Nowitzki	6.5	0.7	1.8	0.7	1.1	2.1	18.3

Figure 6.2 shows the positions of players on the output layer on the SOM and it is immediately clear how they are not mapped in the center of the purple area related to power forwards, but are spread across different positions in the map, even if most of the them are on the edge of the power forward area.

The most interesting thing to point out is the relative distance between them, since it is the parameter that actually describes the difference between their way of playing. Paul Millsap (PM24) and Blake Griffin (BG32) are mapped in very close positions, since their Average Total Rebounds per Game (TRB), Average Turnovers per Game (TOV) and Average Personal Fouls per Game (PF) statistics have very close values. Moreover, they are the two players in the set with the highest values in Average Assists per Game (AST).

Anthony Davis (AD23), with his high values of TRB, Average Blocks per Game (BLK) and PPG is mapped in a position that is half-way between the center and the shooting guard positions, two roles that should have almost nothing in common,

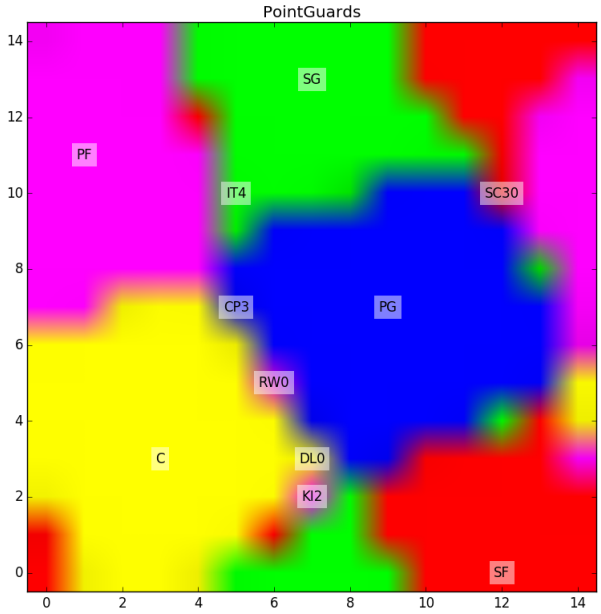

Figure 6.3: Forward mapping of some point guards on the 5 positions trained SOM

but Davis can be considered as a player who grabs rebounds and blocks shots like a center and actually scores like a shooting guard, so the power forward position does not fit his game at all.

LaMarcus Aldridge (LA12), Dirk Nowitzki (DN41) and Kevin Love(KL0) are the 3 players who are mapped close to the power forward cluster of the map, but they are very distant from each other. Love is a good scorer, but his game is characterized by an high number of TRB and a good value of AST. The good rebounding skills make him a good power forward, but his passing skills place him in a position of the map which is close to the point guard area, like in the case of Millsap and Griffin. Aldridge and Nowitzki instead have statistics that are characterized by good scoring and rebounding numbers, but no other excellent statistics. They are both placed on the edge of the power forward position, but Aldridge, who is a better rebounder is close to the Center region, because of his higher TRB and BLK statistics, while Nowitzki, who is a poorer rebounder and shot blocker is placed close to the small forward area.

The same kind of experiment was made on the point guard position, mapping some of the most famous point guards in the league on the output layer of the SOM.

The players who have been mapped in this experiment are Kyrie Irving, Isaiah Thomas, Damian Lillard, Stephen Curry, Chris Paul and Russell Westbrook, whose statistics are shown in Table 6.3.

Figure 6.3 shows how the players are placed on the output layer, and it is possible to see how they are less spread with respect to power forward players, since they are closer to the point guard blue region of the graph. Even if they are close tho the point guard region, the relative distances between them shows that there are some differences between their way of playing the position.

Russell Westbrook (RW0) and Chris Paul (CP3) for example are placed in close

Table 6.3: Point Guards statistics

PLAYER	TRB	BLK	AST	STL	TOV	PF	PTS
Kyrie Irving	3.0	0.3	4.7	1.1	2.3	2.0	19.6
Isaiah Thomas	3.0	0.1	6.2	1.1	2.7	2.0	22.2
Damian Lillard	4.0	0.4	6.8	0.9	3.2	2.2	25.1
Stephen Curry	5.4	0.2	6.7	2.1	3.3	2.0	30.1
Chris Paul	4.2	0.2	10.0	2.1	2.6	2.5	19.5
Russell Westbrook	7.8	0.3	10.4	2.0	4.3	2.5	23.5

position, between the point guard area, the power forward area and the center area. This is due to the fact that they are the two players with the highest values of AST and they have very similar statistics in Average Steals per Game (STL) and PF. They are both good scorers, but Westbrook is better in TRB, a feature that maps him closer to the center yellow region on a neuron of the map which is colored in purple, the color of the power forward position.

Kyrie Irving (KI2), Damian Lillard (DL0) and Isaiah Thomas (IT4) have similar statistical profiles, having low values of TRB, and STL and AST with respect to the other players in the set, in particular Westbrook and Paul, while having average values of PPG. This differences bring them to be mapped in the farthest positions with respect to the point guard area, since Thomas and Irving are closer to the shooting guard green area. Lillard is placed in a strange position half-way between the blue point guard, the green shooting guard and the yellow center areas. The main reason for being closer to the center area is that he is the point guard with the higher value in BLK in the set, he has also good rebounding ability compared to the other players in the set, even if not excellent, and an high number of AST. This mix of good statistics makes him more difficult to be mapped in a single cluster.

Finally Stephen Curry (SC30) is mapped on the opposite side of the point guard cluster with respect to the other players, closer to the red small forward and the green shooting guard clusters. From his statistics it can be seen how he has high values in all the statistical categories, being the best, or close to the best player in the set according to each category. He is the best scorer, with 30.1 PPG, a value that is much higher than the others[2]; moreover, he has very good numbers in AST, TRB and STL with respect to the other players. Having high numbers in almost all the statistical categories makes Curry an outstanding player, and his position is halfway between the blue point guard, the red small forward, the green shooting guard and the purple power forward areas, making him difficult to be assigned to a precise position too.

The latest two presented situations show how many different ways of playing the same position can be found, and statistics can describe the true difference between players, who should probably be separated into different categories.

The results proved how the classic basketball positions are not able to fit the way

[2]In the 2015-2016 regular season Stephen Curry has been the player with the highest PPG in the whole NBA.

many modern players play the game anymore, and many times it is not possible to completely describe a player with just one position. The results of these experiments highlighted the need of new positions and new criteria to classify players.

6.2 The New Positions

The need of new positions has been proved once again by training the SOM to obtain five clusters referring to the five classic positions, and mapping players on that kind of output layer. To define the new positions, the SOM has to be trained with a proper training set, chosen in a way that each way of playing the game is included and considered during the training process.

The clusters that will be obtained on the output layer of the SOM will show how players are divided according to their statistical profiles, automatically defining the new positions of modern basketball.

The data source used for the experiment included statistics reporting the season averages of all the players in the NBA during the 2015-2016 regular season, and the training sets used to create the clusters were computed following the criteria explained in Chapter 5. The statistical categories considered for the new positions definition are the same 7 used for the mapping of players on the five classic positions trained map.

It is important to underline that no particular clustering or shape of the output layer was expected, since the aim of the SOM is to obtain the clusters automatically, just on the base of the data characterizing the input space, in particular the training set used to feed the network during the learning process.

Some trials have been made, training the SOM with different training sets: the one in which the best players from each of the five classic positions are selected on the base of MPG, described in Section 5.2.4; another training set made by players from the starting five of each team in the league, built as described in Section 5.2.2; and finally the training set including the key players[3] of each NBA team, in which players to be included are selected as explained in Section 5.2.3.

The best result, shown in Figure 6.4, was obtained with the latest training set, the one made by key players of each team, which allowed to finally define the new positions, according to the clusters obtained from the training of the SOM.

The 5 areas highlighted in Figure 6.4 refer to the new obtained positions. The first surprising result is that there are still 5 positions, the same number of the classic ones. Anyway these new 5 positions are different from the classic ones and group players together in a different way. Players who were labeled to be playing the same classic position are now separated in different clusters, and each of the new groups includes players from more than one of the old classic positions.

Analyzing the outcome of the whole process, it is possible to define the new positions coming from the automatic clustering of the SOM as follows:

1. **All-Around All Star**. An All-Around All Star (AAS) is a player who is outstanding in many aspects of the game, and so his statistics are far beyond the league averages in many statistical fields. AAS players can make the

[3]The key players are the 8 players with the highest MPG in each team during the 2015-2016 regular season.

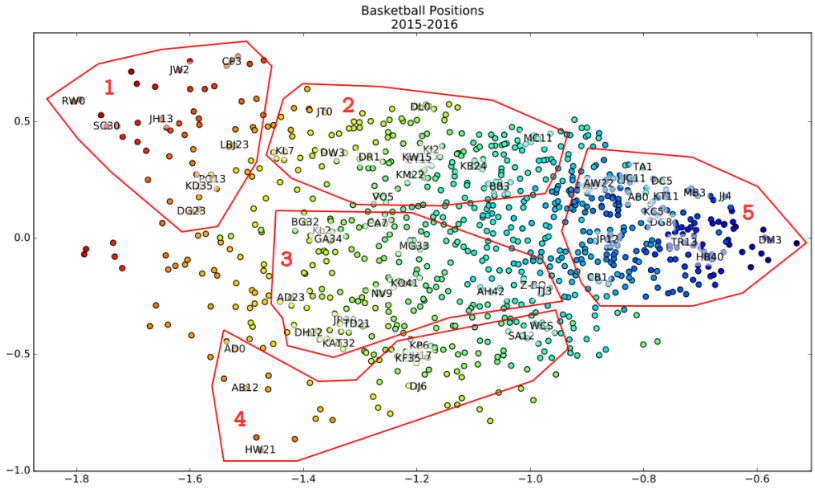

Figure 6.4: Output of the SOM with the clusters defining the new positions.

difference for their teams in many different ways. They are usually elite scorers, but combine the scoring ability with great passing skills, or big rebounding numbers, or great defensive skills, or even all of these characteristics together. Examples of those kind of players are LeBron James (LBJ23), Kevind Durant (KD35), Paul George (PG13), Stephen Curry (SC30) and James Harden (JH13).

2. **Scoring Backcourt**. A Scoring Backcourt (SB) player has in his offensive potential his main weapon on the floor. These kind of players have high values of PPG statistics, but usually they also have good passing skills, which means high values in AST. They are usually below average in statistical fields like TRB and BLK, since they are players who prefer to play away from the basket. Perfect examples of SB players are Kobe Bryant (KB24), Damian Lillard (DL0), Kyrie Irving (KI2) or Dwyane Wade (DW3).

3. **Scoring Rebounder**. A SR is a player who has high average statistics mainly in PPG and TRB statistical fields. These players are usually skilled scorers, both in low-post game[4] and facing the basket, with special rebounding abilities. They are more sized than the SB and are used to play closer to the basket. Examples of SR are Marc Gasol (MG33), Carmelo Anthony (CA7), Karl-Anthony Towns (KAT32), Anthony Davis (AD23) or Blake Griffin (BG32).

4. **Paint Protector**. A PP player is usually not a great scorer, with PPG statistics that are below the averages of the league. They are very good at rebounding and blocking shots, thanks to their size and excellent defensive skills, so their values related to TRB, BLK and STL are higher with respect to the other statistical categories. They usually have a more physical game

[4]A low-post scorer is a player who is able to score points playing closer to the basket, in the low-post position. Playing in the low post means being able to receive the ball close to the basket, giving the back to it. Thanks to peculiar technical skills, it is possible to take advantage of this position and of the size of the player to be able to score from a very short distance.

and play closer to the basket on both ends of the floor. Examples of this kind of players are DeAndre Jordan (DJ6), Andrew Bogut (AB12), Steven Adams (SA12) or Kenneth Faried (KF35).

5. **Role Players**. Finally there is the Role Player (RP), who is usually very good but not excellent in only one statistical category[5], or in some cases have numbers which are just below to the average of the league in all statistical categories. It is difficult to describe the way those players play in a unique way, since in this group players who are considered to be specialists of particular aspects of the game are included. They can be outside shooters, whose main responsibility on the floor is to score from the creations of other players, or defensive specialists, whose responsibility is to guard the best player of the opponent team. These kind of roles can be crucial in a team, but their importance is usually not reflected in their statistics, which are usually lower than other players, which is the reason why those players are clustered together on the output layer of the SOM. Examples of RP are Marco Belinelli (MB3), J.J. Redick (JJ4), Harrison Barnes (HB40), Jabari Parker (JP12) or Avery Bradley (AB0).

As shown in Figure 6.4 the clusters of the output layer of the SOM are not completely separated, so it is possible that a player is placed in a position which can be halfway between two groups. In this case, like for all the other experiments made with the SOM, the relative distance between mapped players on the output layer is the parameter which truly describes the difference between two elements, so analyzing their relative positions it is possible to better understand the meaning of the defined positions.

Figure 6.5 shows the result obtained forward-mapping all the players included in the training set on the output layer of the SOM. This kind of experiment was made after the positions definition, since having so many players, the labels describing them can overlap making it difficult to recognize the text of the label itself and so the identity of the mapped player. During the positions definition, some proper players have been included in the input set, in a way that allows to have the minimum number of overlapping labels, and so being able to recognize and identify the mapped players on the output layer. Even if the players distribution is more flat across the map, it is still possible to recognize the clusters that have been already defined on the output layer with a limited number of mapped players, confirming the new positions definition. A further confirmation comes from the fact that the cluster corresponding to the AAS players, which should contain the best players of the league, is the one in which there is lower concentration of mapped players, even mapping all the components of the training set.

6.2.1 Players Comparisons

Comparing the relative distance between players on the map it is possible to see how good the result of the SOM clustering is and how players are different from each other.

[5]Most of the times when a RP is very good but not excellent in just one statistical category, this category is PPG.

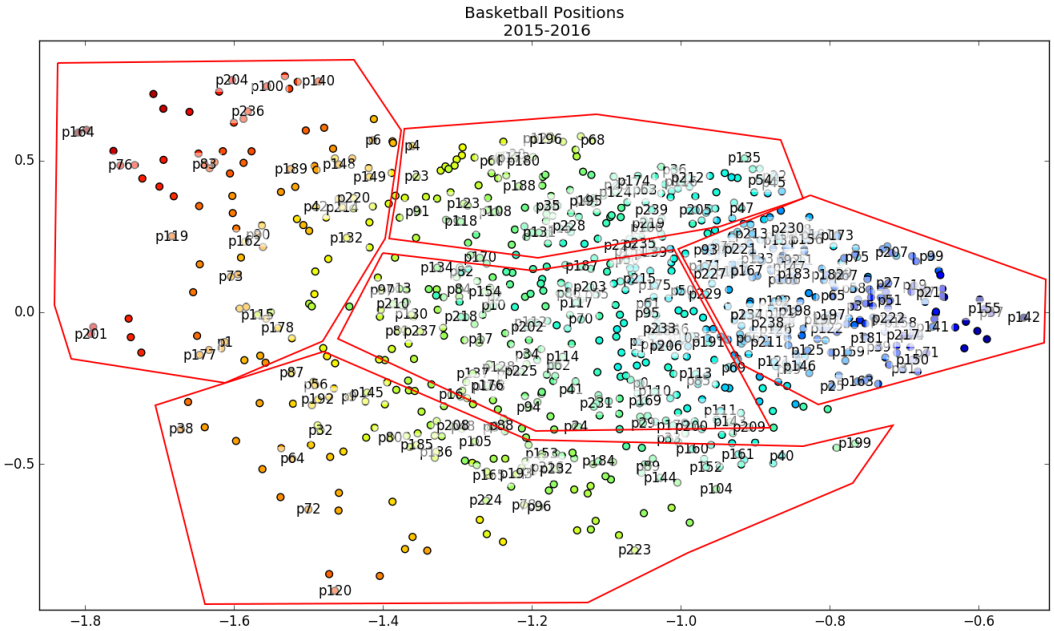

Figure 6.5: All players of the NBA training set, mapped on the output layer.

As a first explicative example of the effectiveness of the SOM mapping, the position of two particular players will be analyzed: DeAndre Jordan (DJ6), who was assigned by the SOM to the Paint Protector position, in the lower part of the map, and Damian Lillard (DL0), who is mapped in the Scoring Backcourt position, in the upper part of the map.

Figure 6.6 shows the positions of the two players on the map, and it is immediately clear how they are placed in opposite positions of the output layer, on the upper and lower edges of the central area of the map. This kind of distance between them resembles a great difference between their statistical profiles and so their way of playing the game.

Damian Lillard is assigned to the SB position, so he is mapped to be a player who can score a lot and give many assists, playing far from the basket and having low values of statistical fields like TRB and BLK.

On the other side, DeAndre Jordan is assigned to the PP position, a position made by players who usually have lower values in statistical categories like PPG and AST, but are very good at rebounding and blocking shots.

As a proof of the difference between their statistical profiles, Figure 6.7 reports the values related to their statistics during the 2015-2016 regular season in a graphical comparison. From the graphical comparison it is possible to see how Lillard has almost double of Jordan's PPG, reported in the comparison with the "PTS" label, and averaged 6.8 AST in front of Jordan's 1.2. On the other side Jordan's statistics about TRB, reported in the graph with the "REB" label, and BLK are more than 3 times higher with respect to Lillard ones.

This comparison proofs how Lillard and Jordan ways of playing the game are different and characterized by different values in the statistical categories considered during the training of the SOM. What is more the comparison shows how the relative

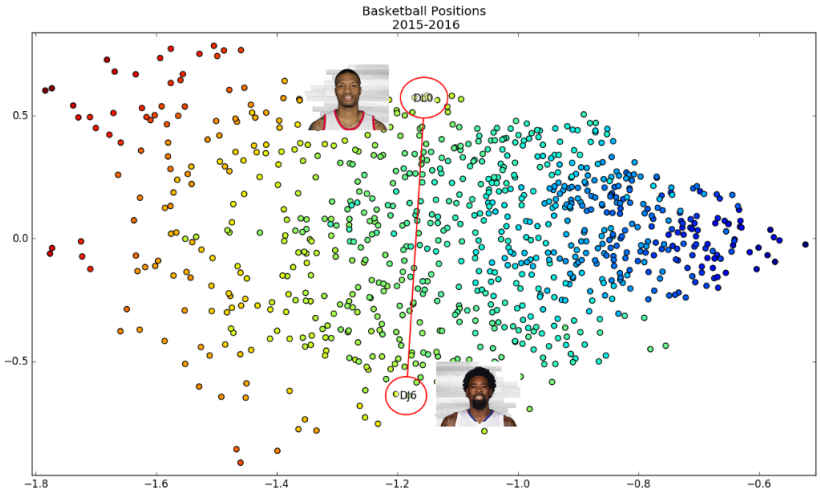

Figure 6.6: Output layer of the SOM, Damian Lillard and DeAndre Jordan highlighted, photos of players come from Source [**?**]

distance between players in the map truly resembles the difference between their statistical profiles, since the difference between Lillard and Jordan statistics have been proved to be as big as the distance which separates them on the the output layer of the SOM.

Once we proved how the relative distance between players is a good metric for showing the difference between the way they play, another kind of analysis on the clusters structure can be made. The output of the SOM algorithm does not provide very well-separated clusters, so as reported in conclusion of Section 6.2 a player may be mapped half-way between two clusters. Its distance with respect to another player shows the difference between his statistical profile and all the others, so if DeAndre Jordan and Damian Lillard are very different between each other, there must be some players between them who are half-way between Lillard way of playing the game and Jordan way.

Analyzing the position of players along the vertical direction between Lillard and Jordan, two more players have been selected, to show how the statistical profiles of players evolve moving through the map, gradually changing from one extreme to the other.

The two selected players in between are Karl-Anthony Towns (KAT32) and Carmelo Anthony (CA7), both mapped in the Scoring Rebounder area of the map. Being SR they should be players who have statistical profiles which are mainly characterized by high values of PPG and TRB.

Figure 6.8 shows the positions of the 4 players under analysis on the output layer of the SOM. It is possible to notice how, even if Towns and Anthony are both SR, Towns is mapped in a position of the output layer which is closer to Jordan's one, so he is in the lower part of the SR area, close to the PP area. Anthony on the other hand is closer to Lillard's position on the output layer, since he is mapped in the upper limit of the SR region, close to the SB area. This particular topology of players shows how some differences can be found within the same group, and the statistical profiles of players can be different within the same category, even if the

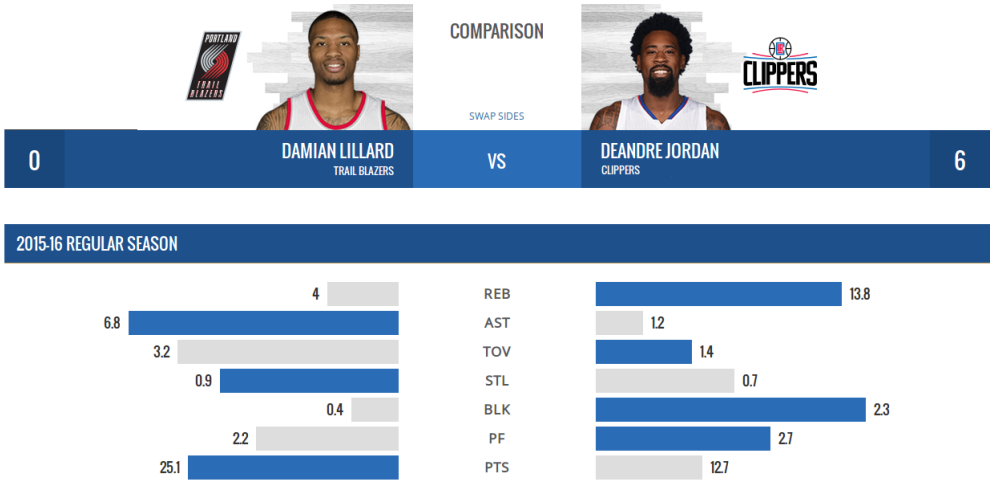

Figure 6.7: Comparison between Damian Lillard and DeAndre Jordan statistics, Source [**?**]

main features are the same.

Table 6.4: From Paint Protector to Scoring Backcourt Statistics

PLAYER	TRB	BLK	AST	STL	TOV	PF	PTS
DeAndre Jordan	13.8	2.3	1.2	0.7	1.4	2.7	12.7
Karl-Anthony Towns	10.4	1.7	2.0	0.7	2.2	3.0	18.3
Carmelo Anthony	7.7	0.5	4.2	0.9	2.4	2.5	21.8
Damian Lillard	4.0	0.4	6.8	0.9	3.2	2.2	25.1

An analysis on the statistical profiles was necessary to investigate this evolution of statistics across the map. Statistical profiles of the 4 players are reported in Table 6.4, and an histogram of the statistical categories which better describe the difference between players is shown in Figure 6.9.

The histogram in Figure 6.9 shows effectively how the statistical profiles of players change, moving from the bottom part of the map through the upper part. It is immediately clear how the values of the PPG and AST statistical categories, the blue and orange columns, gradually increase moving from the player mapped in the bottom part of the map, DeAndre Jordan, to the player mapped in the upper part of the map, Damian Lillard. On the other side, the values of the TRB and BLK statistical categories, represented by the yellow and green columns, decrease along the same path.

Towns, who is mapped closer to Jordan, has better values in PPG and AST than Jordan, but worse than Lillard and Anthony, while he has worse values of TRB and BLK than Jordan, but better values with respect to Lillard and Anthony.

In a similar way Anthony, who is mapped closer to Lillard, is a worse scorer and passer than Lillard, but better than Towns and Jordan, while he is a better rebounder and shot blocker than Lillard, but worse than Towns and Jordan.

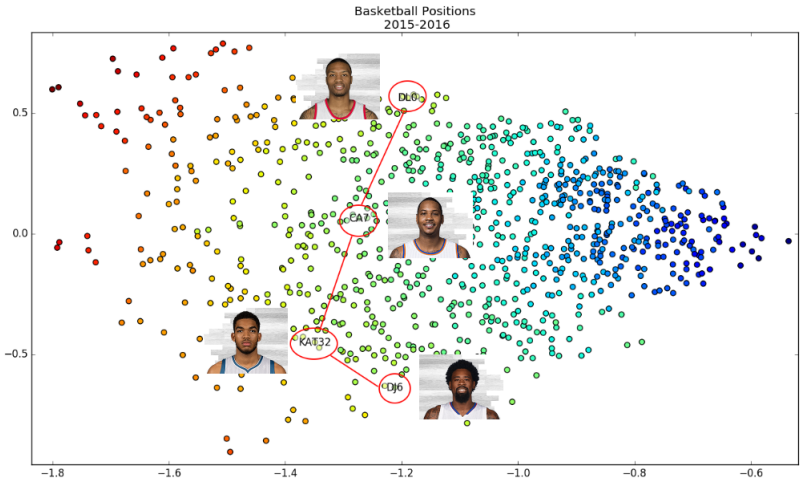

Figure 6.8: Players from Paint Protector to Scoring Backcourt positions, images of players are taken from Source [**?**]

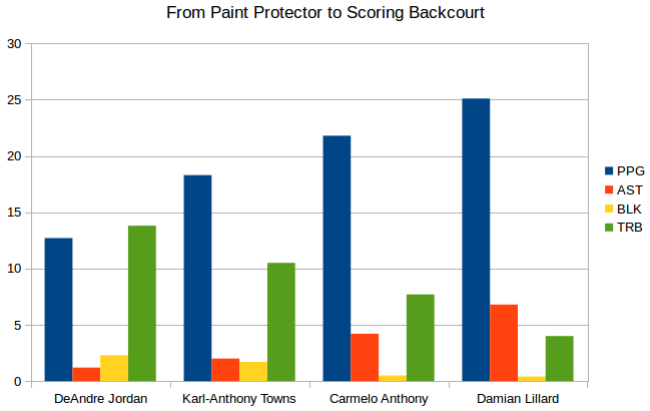

Figure 6.9: Histogram of the statistics of the 4 considered players, data reported in Table 6.4

The statistical profiles of Towns and Anthony are proved to fit perfectly in the gradual connection between Lillard and Jordan statistical profiles, since their statistics are half-way between the ones of the two most different players.

The experiments and analysis made on relative distances between players show how the mapping algorithm of the SOM is an efficient and effective descriptor of the difference between players statistical profiles.

This proves also the validity of the new positions defined in Section 6.2, which are able to describe the way a player plays in a more effective and objective way with respect to the classic basketball positions.

Despite the good results, also in the new positions there are some differences between players of the same cluster, but this is something that cannot be avoided, since each player will always have slightly different statistics with respect to the others, and to describe the way each single player plays thousands of variables would

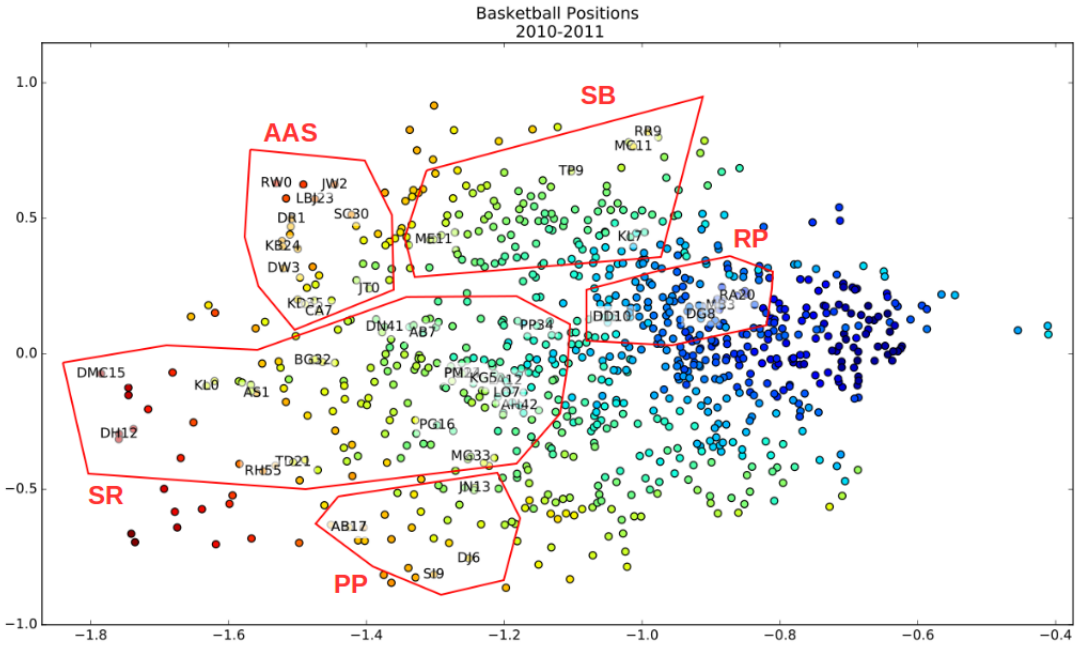

Figure 6.10: Positions definition for 2010-2011 regular season.

be needed.

Anyway the new positions have been proved to be able to describe the main features in the statistical profiles of players assigned to a certain position, which was the expected result of the SOM process.

6.3 Previous Seasons Analysis

To have a further confirmation of the validity of the new positions defined in Section 6.2, the same algorithm was applied on data referring to seasons of the past years. In particular the new datasets used as input space for the SOM algorithm, were the regular season average statistics of all the NBA players from the 2010-2011 regular season, until the 2014-2015 season.

The idea was to repeat the same computations to see if the same positions and clusters could be found in past seasons, even if some of the players involved in the process are different from the ones considered in the training of the 2015-2016 SOM.

The training set that was created using the same criteria of the one used to define the 2015-2016 regular season positions, as explained in Section 5.2.3. The statistical categories used to characterize players of the 2015-2016 season have been used to collect basketball statistics for decades, so it was possible to retrieve the same statistical categories for each of the past seasons.

The results shown from Figure 6.10 to Figure 6.14 provide the positions definition in each of the past seasons considered for the analysis. It is immediately clear how it was possible to find the same clusters, and so the same positions found for the 2015-2016 season in all the graphs referring to the previous seasons.

Each season presents clusters of different shape and size, due to the different

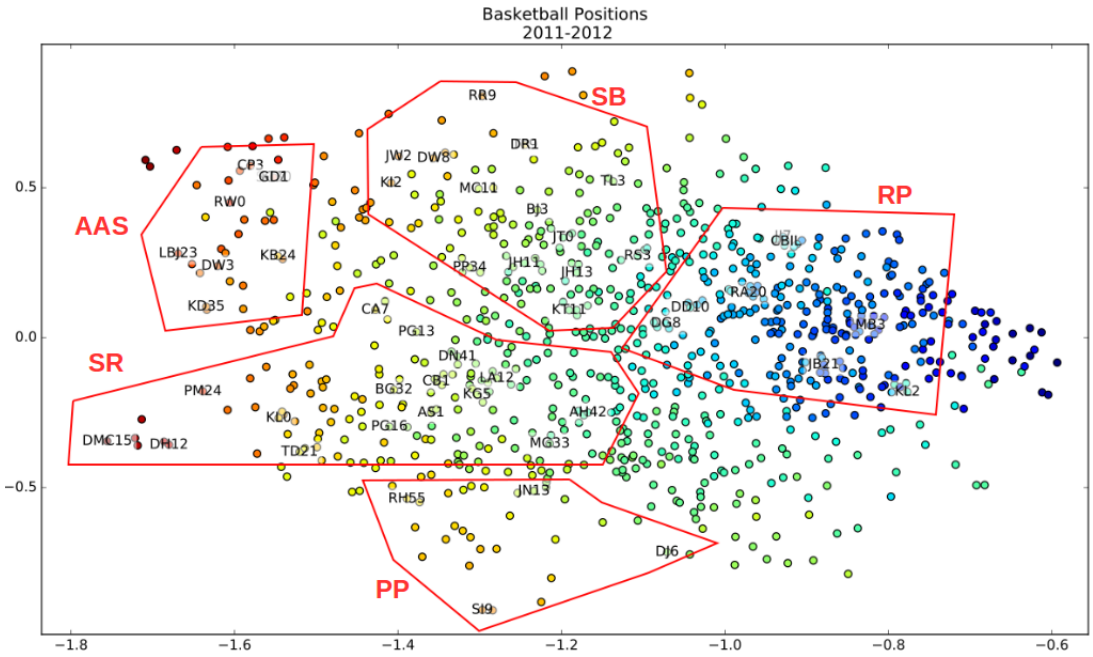

Figure 6.11: Positions definition for 2011-2012 regular season.

number of players mapped in a certain position and the relative distance between them.

For example in the 2010-2011 season graph, shown in Figure 6.10, the AAS cluster is very small and the players are mapped in very close positions, due to the similarity between their statistical profiles. Some of the players mapped in the 2010-2011 AAS position are Russell Westbrook, Lebron James, Kobe Bryant and Dwyane Wade, whose statistics are reported in Table 6.5. From the comparison of their statistical profiles it is possible to see how the all had high values in PPG, and very similar values in BLK, STL, TOV and PF. The differences in the numbers referring to TRB and AST are the features that allow to distinguish between them, mapping the players in different positions of the output layer, even if very close to each other.

Table 6.5: 2010-2011 AAS players statistics

PLAYER	TRB	BLK	AST	STL	TOV	PF	PPG
Russell Westbrook	4.6	0.4	8.2	1.9	3.9	2.5	21.9
LeBron James	7.5	0.6	7.0	1.6	3.6	2.1	26.7
Kobe Bryant	5.1	0.1	4.7	1.2	3.0	2.1	25.3
Dwyane Wade	6.4	1.1	4.6	1.5	3.1	2.6	25.5

Another example of very small cluster is the SB group of players of the 2012-2013 regular season, shown in Figure 6.12. Some of the players mapped in this cluster are Ty Lawson, Damian Lillard, Tony Parker and Deron Williams, whose statistics are

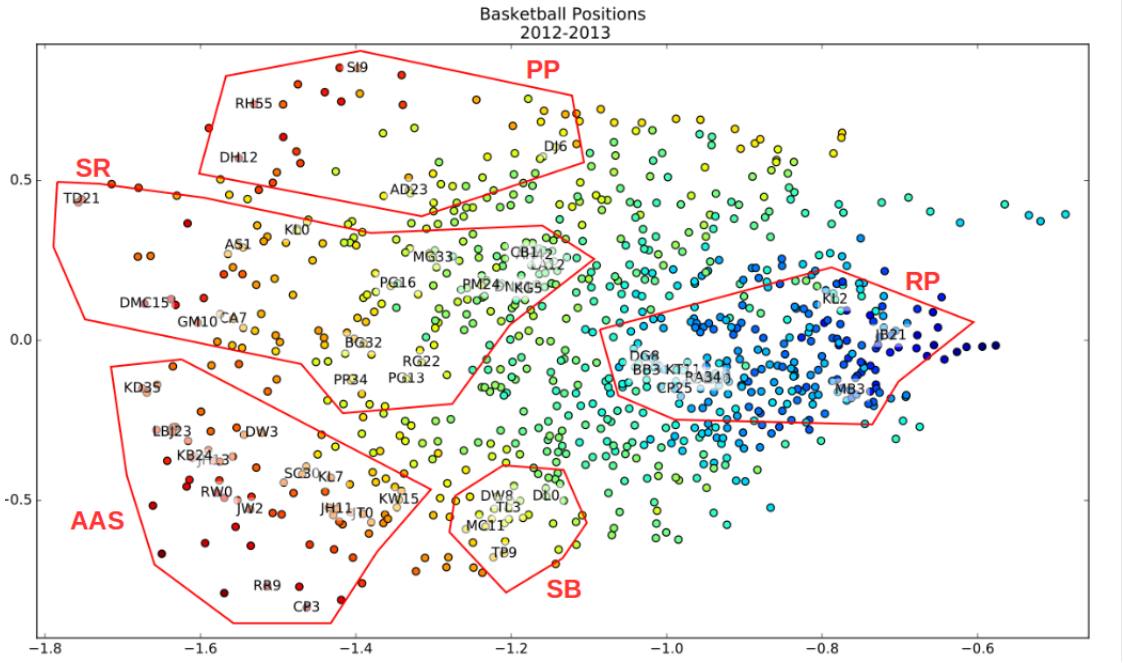

Figure 6.12: Positions definition for 2012-2013 regular season.

reported in Table 6.6. Also in this case it is possible to prove the similarity between their statistical profiles, since all the statistical categories provide very similar values for each player, with the highest variability in the PPG category, which is just 3.6 PPG between the lowest and the highest value.

Table 6.6: 2012-2013 SB players statistics

PLAYER	TRB	BLK	AST	STL	TOV	PF	PPG
Ty Lawson	2.7	0.1	6.9	1.5	2.5	1.8	16.7
Damian Lillard	3.1	0.2	6.5	0.9	3.0	2.1	19.0
Deron Williams	3.0	0.4	7.7	1.0	2.8	2.5	18.9
Tony Parker	3.0	0.1	7.6	0.8	2.6	1.4	20.3

Apart from the relative distance between single players, which have been proved to be an effective descriptor of the difference or similarity between them, there is another feature which can be recollected in all the graphs referring to the previous seasons: the relative positions of clusters between each other.

It is evident from all the graphs how the relative positions between clusters is maintained across different seasons. The AAS and RP clusters are positioned on opposite sides of the map, respectively on the left and right side. The SB and PP clusters are placed in opposite positions too, on the upper and lower parts of the map, while the SR cluster is placed in the middle-left part of the map, always separating the SR and PP clusters.

This particular pattern or shape of clusters within the map confirms the analysis

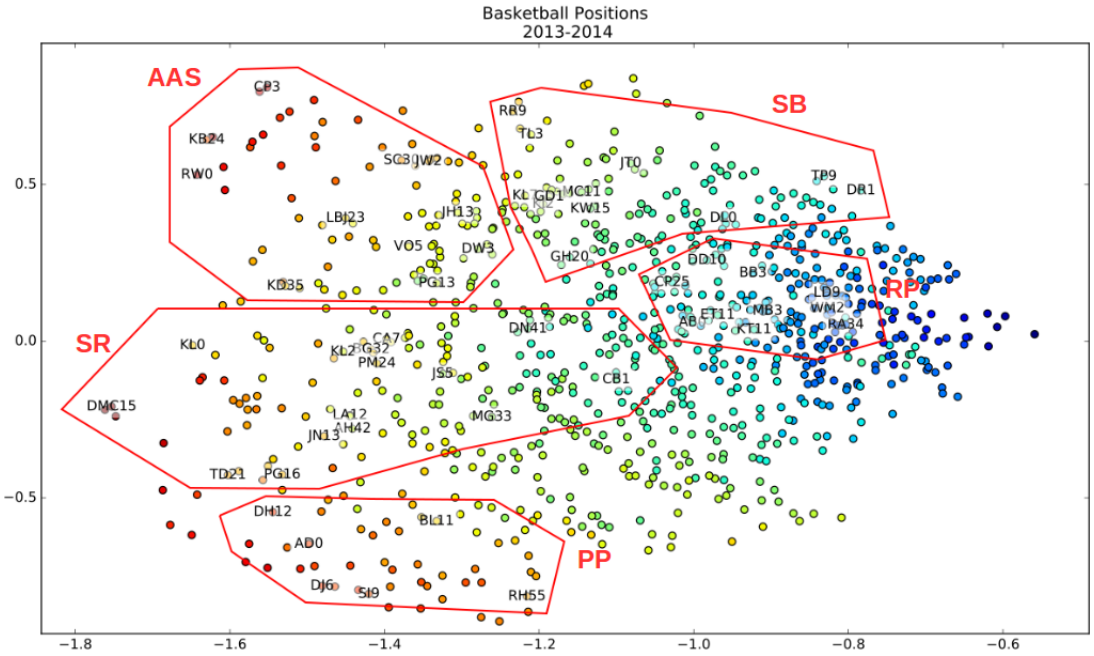

Figure 6.13: Positions definition for 2013-2014 regular season.

made in Section 6.2.1, since the same gradual evolution from one position to another can be found in all the maps of the previous seasons, for example having the SR players mapped in positions that are half-way between the PP and SB players.

This kind of analysis supports the quality of the new defined positions, but also confirms the fact that there are still differences between players clustered in the same position, even if the new positions are able to better describe their way of playing the game.

Another particular feature retrieved looking at the results from the previous seasons, is the presence of some players in the same cluster in each of the considered seasons. For example Stephen Curry (SC30) and Russell Westbrook (RW0) are always mapped in the AAS position, Pau Gasol (PG16) and Marc Gasol (MG33) are always placed in the middle of the map, in the SR category, or DeAndre Jordan (DJ6) never moved from the extreme upper or lower part of the map, in the cluster related to PP players.

Comparing statistics of players who were mapped in similar positions through different seasons, no particular change in their statistical profiles should be found, but not all the players mapped in different seasons maintained their position in the map through the years.

6.4 Players Evolution Through the Years

Analyzing the results of the mapping of the past seasons, it was possible to find the same new positions, defined in Section 6.2, in each of the previous years considered in the experiment. Clusters in the different years were not always composed by the same players, even if some of them "stayed" in the same position through the years.

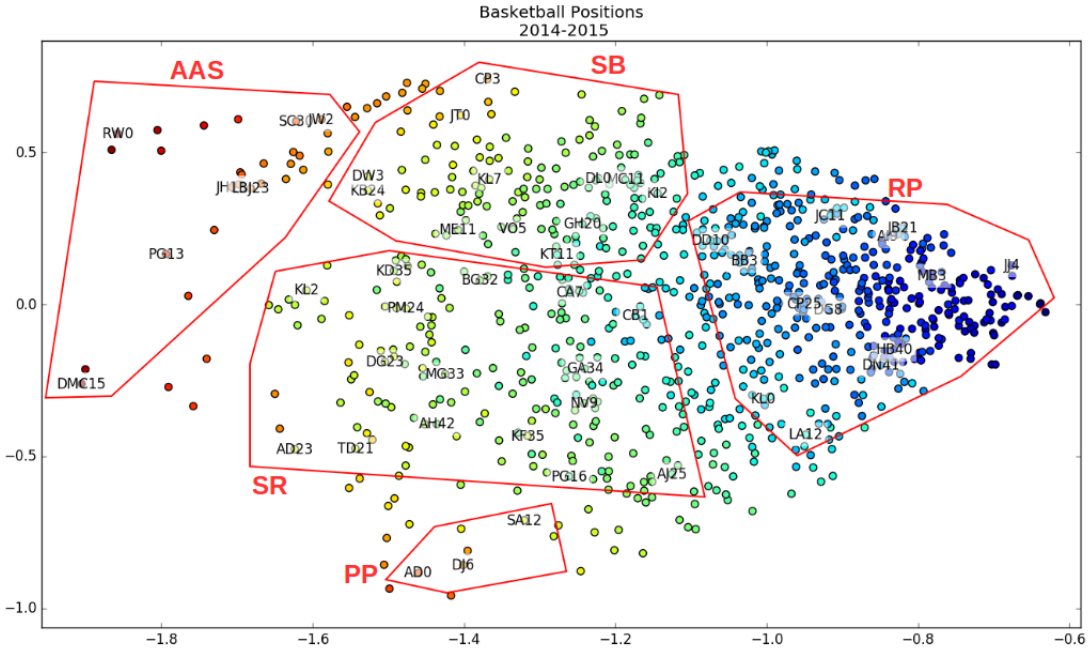

Figure 6.14: Positions definition for 2014-2015 regular season.

Studying how players can change their position on the output layer of the SOM, it is possible to understand how they changed their way of playing through the years, and so how their statistical profiles evolved in the different seasons.

To be able to show the evolution of a player across different seasons, the dedicated methods of the input set loader and visualization modules described in Section 4.2.4 must be used. The SOM is trained with the 2015-2016 key players training set described in Section 5.2.3, a proper input set is created by the `InputSetCreator` module described in Section 4.2.2, and the results of the visualization are loaded as explained in Section 4.2.3. The result allows to show the position of the statistical profiles describing the performances of a player in the different past seasons, on the output layer of the map trained with the 2015-2016 training set. The relative distance between the positions of the labels referring to different seasons, show how the player evolved his game through the years.

A first notable example will show how a player whose position is not moving in the output layer of the map through seasons, had very similar statistics through the years, and so his way of playing did not change a lot.

The player taken as an example of this "stability" is LeBron James, one of the most famous basketball player and athlete in the world, who has been playing in the NBA since 2003, being one of the best players in the league for many years. Table 6.7 reports his regular season average statistics in the last 6 seasons, from 2010-2011 to 2015-2016, and it is immediately clear how he maintained the high level of his game regularly through the years, with values of each statistical categories which are slightly different from one year and another one.

Table 6.7: LeBron James Evolution Through The Years

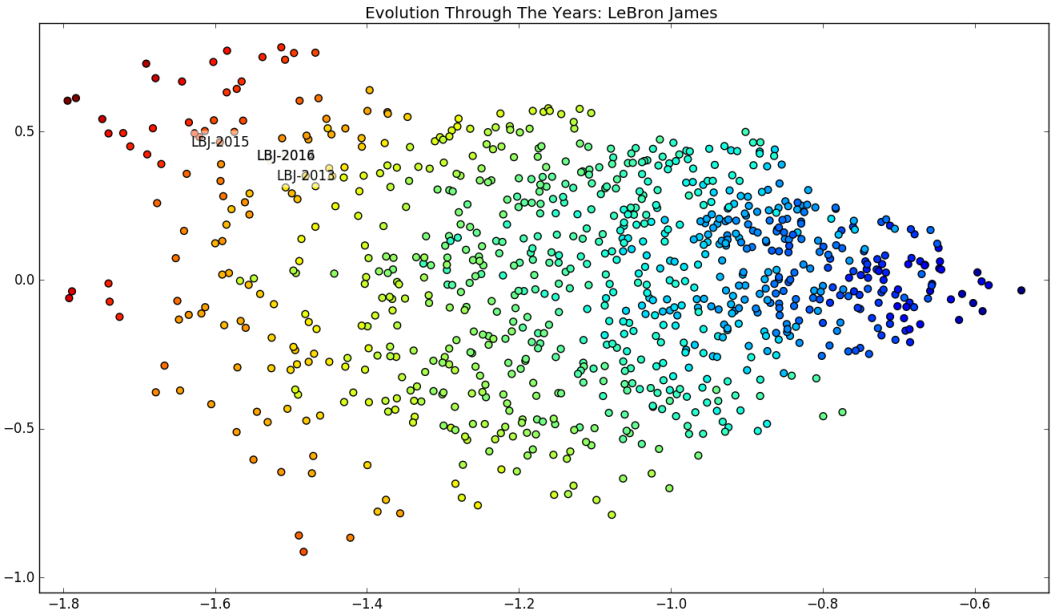

Figure 6.15: Evolution of LeBron James position on the output layer, from 2010-2011 to 2015-2016.

PLAYER	TRB	BLK	AST	STL	TOV	PF	PTS
LeBron James 2010-2011	7.5	0.6	7.0	1.6	3.6	2.1	26.7
LeBron James 2011-2012	7.9	0.8	6.2	1.9	3.4	1.5	27.1
LeBron James 2012-2013	8.0	0.9	7.3	1.7	3.0	1.4	26.8
LeBron James 2013-2014	6.9	0.3	6.3	1.6	3.5	1.6	27.1
LeBron James 2014-2015	6.0	0.7	7.4	1.6	3.9	2.0	25.3
LeBron James 2015-2016	7.4	0.6	6.8	1.4	3.3	1.9	25.3

Figure 6.15 shows the evolution of James position on the map through the years. Even if 6 different statistical profiles, one for each considered season, have been mapped on the output layer, only 3 labels are visible and displayed in separated positions. This strange visualization effect is due to the fact that his statistics in the different seasons are so close between each other that 4 of the elements of the input set have been mapped in the same position on the output layer, and so the corresponding labels are perfectly overlapped. In particular the "LBJ-2016" label, referring to the position of his statistics related to the 2015-2016 regular season, is overlapped with the labels referring to the statistics related to the 2010-2011, 2011-2012 and 2013-2014 seasons. The changes in his positions on the output layer of the SOM are really minimal, and he has been mapped in the AAS cluster in all the seasons that were analyzed, showing how his performances in the last 6 years have been constantly at really high levels.

Now some examples of players who changed their position on the map through the years will be reported to show how their "movements" on the map are actually reflecting the changes in their statistical profiles.

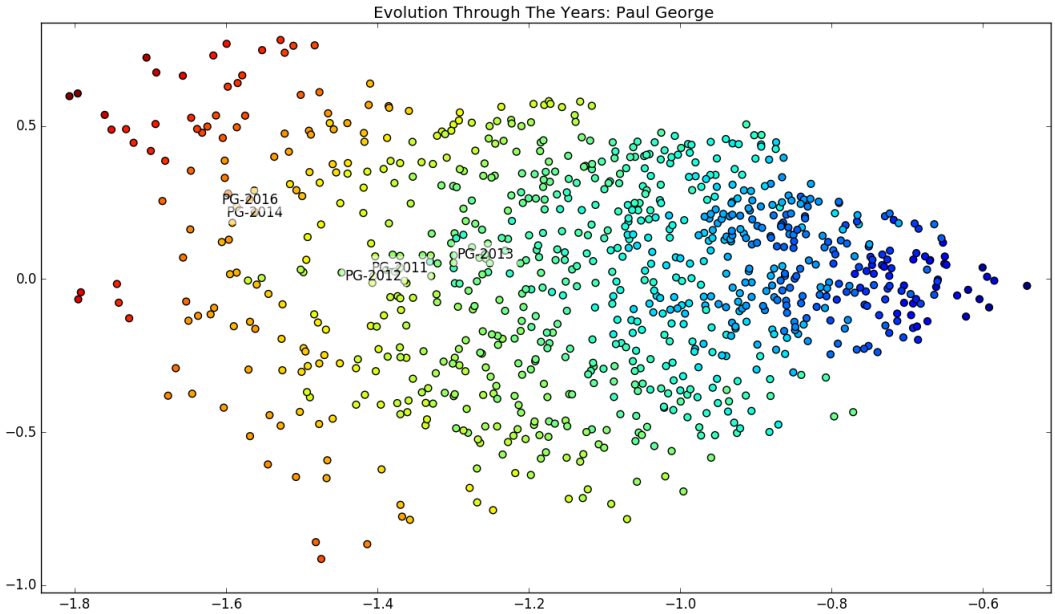

Figure 6.16: Evolution of Paul George position on the output layer, from 2010-2011 to 2015-2016.

The first "evolving" player under exam is Paul George, who has been playing in the NBA since the 2010-2011 season, after being drafted by the Indiana Pacers, which is the only team he played for since the beginning of his professional career.

Analyzing the evolution of his position since his first season in the league it is possible to see how much time it takes for a player from his rookie season[6] to become an all star player, or better to become an AAS player.

Table 6.8: Paul George Evolution Through The Years

PLAYER	TRB	BLK	AST	STL	TOV	PF	PTS
Paul George 2010-2011	3.7	0.4	1.1	1.0	1.1	2.1	7.8
Paul George 2011-2012	5.6	0.6	2.4	1.6	1.8	2.9	12.1
Paul George 2012-2013	7.6	0.6	4.1	1.8	2.9	2.9	17.4
Paul George 2013-2014	6.8	0.3	3.5	1.9	2.8	2.5	21.7
Paul George 2015-2016	7.0	0.4	4.1	1.9	3.3	2.8	23.1

The positions he assumed on the SOM through the different years are shown in Figure 6.16 and his statistics in all the considered seasons are reported in Table 6.8. His statistics referring to the 2014-2015 season have not been considered, since in that season he played only 6 games over the 82 of the whole regular season, due to a bad injury he had in the summer of 2014, during a game with the USA national basketball team, in which he unfortunately broke his leg.

[6]Rookies are the players at the first year in the league, the rookie season is their first season as professional players.

His statistics in the first two seasons of his career placed him in the SR position, due to the fact that the two most important statistical categories in his statistical profile were PPG and TRB and he had good numbers in BLK with respect to the other statistical categories. While his scoring and rebounding numbers kept increasing regularly, an increase in AST during his third season made him move closer to the upper part of the map, so closer to the SB area, due to an improvement in his passing game. The season before his injury and the last one were his two best seasons. His PPG statistics improved in both seasons, making him one of the best scorers in the league, what is more he maintained his numbers in TRB, AST and STL, but also the number of TOV increased during the last two seasons he played. The continuous improvement of his statistics made him move from the SR position to the AAS position, and actually nowadays he is considered to be one of the best players in the NBA.

The second player who had an interesting evolution in his game during the last few seasons is Draymond Green, from the Golden State Warriors, whose particular way of playing has already been described in Section 2.2. Green was drafted by the Warriors in the 2012-2013 season and he played all of the last 4 seasons with the same team.

As explained in Section 2.2 during his first year in the league, finding a position on the floor for him was a big problem for the Golden State Warriors head coach Steve Kerr, who made him play just a few MPG due to his particular characteristics, and so Green statistics during his first season were quite bad, as reported in Table 6.9.

Table 6.9: Draymond Green Evolution Through The Years

PLAYER	TRB	BLK	AST	STL	TOV	PF	PTS
Draymond Green 2012-2013	3.3	0.3	0.7	0.5	0.6	2.0	2.9
Draymond Green 2013-2014	5.0	0.9	1.9	1.2	1.1	2.8	6.2
Draymond Green 2014-2015	8.2	1.3	3.7	1.6	1.7	3.2	11.7
Draymond Green 2015-2016	9.5	1.4	7.4	1.5	3.2	3.0	14.0

As coach Kerr started to better understand his potential, his MPG increased during the next seasons and so his statistics. Coach Steve Kerr used to say that Green is very good at doing so many things on a basketball court, indeed the particular feature of Green statistical profile is that all the statistical categories describing his way of playing increased through the years, making him a better and better player as seasons passed.

This continuous improvement of his statistics brought to a consequent movement of his position on the output layer of the SOM, as shown in Figure 6.17. During his first season he was mapped in the SR cluster, but very close to the RP area, since his statistics were quite low and below the averages of the league. During the next 2 seasons a good increase in PPG, TRB and BLK made him move in the opposite left part of the SR area, close to the AAS cluster. Finally in the last season all the statistical categories improved again, but the doubling of the AST statistic was the main reason which made the SOM algorithm map him in the AAS position. Nowadays Draymond Green is considered by many people as one of the best players in the league, and for sure one of the most complete ones, being a crucial element

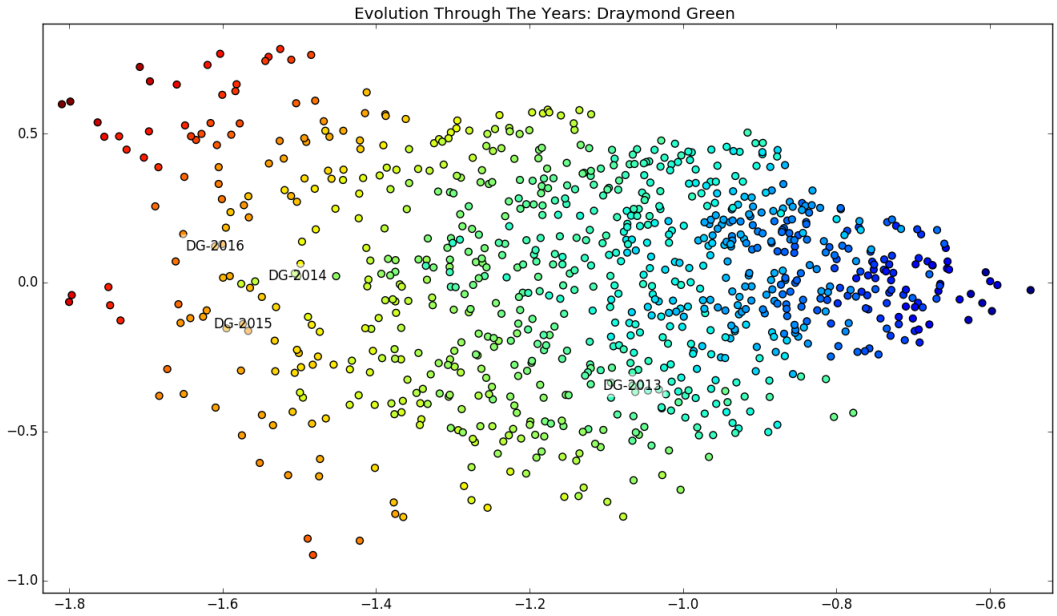

Figure 6.17: Evolution of Draymond Green position on the output layer, from 2012-2013 to 2015-2016.

of the Golden State Warriors, who won the championship in 2015 and reached the NBA finals in 2016.

The third an last example of player changing his way of playing during the last few seasons is James Harden of the Houston Rockets. Harden started his NBA career in Oklahoma, after being selected by the Oklahoma City Thunder in the 2009 NBA draft.

He played in Oklahoma City during the first 3 seasons of his career, being an important player for the team, but not the most important. The Oklahoma City Thunder are the team were Kevin Durant and Russell Westbrook have been playing until the 2015-2016 season, and those two player were the most important in the team, since they are actually two of the best players in the NBA[7].

Even if he was considered to be the third player in terms of importance for the team, starting almost all the games from the bench, Harden was able to have a big impact during the years he spent in Oklahoma, helping the team reach the NBA Finals in the 2011-2012 season, the season in which he also won the "Sixth Man Of The Year" award[8].

The talent and potential he showed during his first NBA seasons in Oklahoma made many people think that he deserved to be more than the third most important player of a team. During the summer of 2012 he moved from Oklahoma City to join the Houston Rockets, the team in which he plays nowadays. In Houston he

[7]As a further proof of their value, both Durant and Westbrook are mapped in the AAS position in the output layer of the SOM referring to all of the seasons considered during this work of thesis, from 2010-2011 to 2015-2016.

[8]The "Sixth Man Of The Year" award is assigned by the NBA to the player who was able to give the best contribute to his team starting from the bench. The "Sixth Man" in basketball is the first player who comes out of the bench to change the players who started the game.

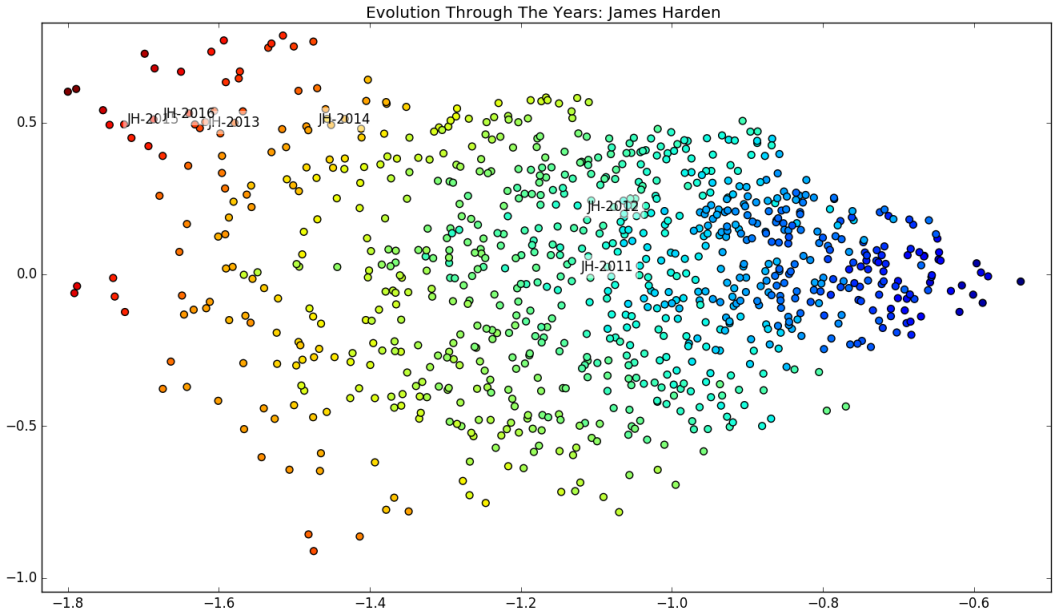

Figure 6.18: Evolution of James Harden position on the output layer, from 2010-2011 to 2015-2016.

was immediately considered the most important player in the team, and the general manager and coaching staff of the Houston Rockets built the team around him.

His statistics reported in Table 6.10 reflect this change of status from the first 2 seasons with the Thunder to the next 4 with the Rockets.

Table 6.10: James Harden Evolution Through The Years

PLAYER	TRB	BLK	AST	STL	TOV	PF	PPG
James Harden 2010-2011	3.1	0.3	2.1	1.1	1.3	2.5	12.2
James Harden 2011-2012	4.1	0.2	3.7	1.0	2.2	2.4	16.8
James Harden 2012-2013	4.9	0.5	5.8	1.8	3.8	2.3	25.9
James Harden 2013-2014	4.7	0.4	6.1	1.6	3.6	2.4	25.4
James Harden 2014-2015	5.7	0.7	7.0	1.9	4.0	2.6	27.4
James Harden 2015-2016	6.1	0.6	7.5	1.7	4.6	2.8	29.0

All of his statistics improved a lot after the transfer to Houston, in particular AST, STL and PPG above all, this is due to the fact that Harden had the possibility to be the most important player in the team for the Rockets, with an increasing number of possessions in his hands, more responsibilities and the occasion to show his complete potential in a team created to magnify his characteristics.

This statistical improvement turned into and improvement of his game and also of his position on the SOM. Figure 6.18 shows how in the two seasons playing for the Oklahoma City Thunder, Harden statistical profile made the SOM algorithm map him in areas close to the middle of the output layer, corresponding to the SB position, and even quite close to the RP position. After the trade and the passage

to the Houston Rockets he was selected to the All Star Game in all the seasons he played in Texas, being considered one of the best scorers in the league. This change of status and statistics is reflected in the change of his position on the output layer of the SOM, since his statistical profiles from the 2012-2013 season to the 2015-2016 season are all mapped in the AAS position.

Analyzing data from different years and seasons it was possible to better understand how the game evolved during the last few years, and above all to understand how some players game changed, as a consequence of their natural growth, or after particular events which made the change possible.

Being able to analyze and see the evolution of players through data and statistics can be a very important tool for coaching staffs and general managers, to make proper evaluations on players performances and take the best decisions to build a winning team.

6.5 Expand The Analysis With More Statistical Categories

Across the different experiments and analysis reported in this chapter, one variable never changed: the number of statistical categories used to describe the way a player plays the game.

The starting point for selecting which statistical categories should be used to describe a player profile comes from Muthu Alagappan and his research, presented in Section 2.3. Alagappan selected the 7 statistical categories that are the most commonly used to describe performances of basketball players all around the world, since they are a good choice for defining a kind of player profile that is suitable for players coming from any league in the world.

Anyway, nowadays statistical analysis is considered as a very important tool in many professional and non-professional leagues, and so many teams and leagues around the world started collecting more and more statistics about players and teams performances.

Since from official sources of leagues like the NBA and many others it is possible to easily retrieve more sophisticated data, with respect to the 7 statistical categories used by Alagappan to describe players, we decided to modify and increase the number of statistical categories used to describe each player's game.

The idea is to describe players in a more precise and sophisticated way, using more data which should better resemble the different aspects of players game, and see if the clustering coming out of a SOM, trained considering these new statistical categories, produces different results in terms of players positions.

First of all it was decided to separate the TRB statistic in two statistical categories: Average Offensive Rebounds per Game (ORB) and Average Defensive Rebounds per Game (DRB). Having two statistical categories instead of one, it is possible to separate rebounds grabbed on the offensive end of the floor, which come from a shot missed by a teammate, from the ones grabbed on the defensive end of the floor, which come from a shot missed by an opponent.

Three more statistical categories were added to players profile, related to players shooting percentage: Average 3-point Shooting Percentage (3PP), Average 2-point

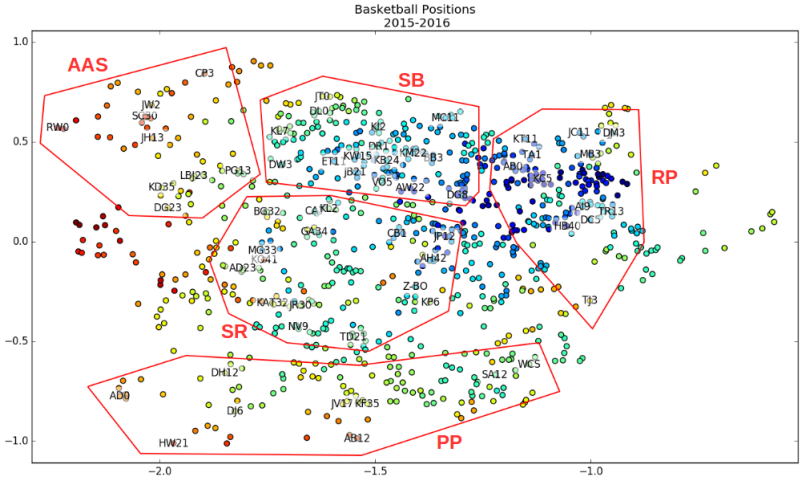

Figure 6.19: Output layer of the SOM trained considering 11 statistical categories.

Shooting Percentage (2PP) and Average Free-Throw Shooting Percentage (FTP). Analyzing these statistics referring to shooting percentages it is possible to understand a player offensive attitude. For example players who have very good 3PP and FTP statistics are usually players who prefer to play far away from the basket, while players who have worse number in 3PP, but very good numbers in 2PP are more devoted to the game in the paint[9], closer to the basket.

The new statistical categories are added to the other 7 that were used to characterized players profiles in the previous experiments, increasing the total number of considered categories from 7 to 11: PPG, ORB, DRB, AST, STL, TOV, BLK, PF, 3PP, 2PP and FTP.

Each element of the input space now becomes an array of 11 elements, and so the neurons of the SOM are now characterized by weight vectors of the same dimension. The whole software application and the SOM algorithm did not need to be changed after the increase of the number of statistical categories, since it has been developed in a parametric and scalable way that allows to perform the SOM algorithm on an input space of any dimension.

Passing the labels of the statistical categories to be used to the main "BasketballSOM" class, assuring that they are corresponding to the label used in the data source to refer the desired statistical categories, all data are retrieved correctly and the whole algorithm works as presented in Section 4.2.4.

The SOM has been trained with different training sets, created according to the criteria explained in Chapter 5, but in this case the training set which gave the best results in term of clustering is the one which includes players coming from the starting five of each team, described in Section 5.2.2.

Figure 6.19 shows the result of the clustering and the positions defined thanks to this result. Even though the number of statistical categories has been increased, the same 5 positions have been found on the output layer of the map. The main

[9] "To play in the paint" in basketball means to play close to the basket. "The paint" is the rectangle area around the basket, which is usually colored in a different way with respect to the other parts of the floor, which is the reason why it is called "paint". This is also the reason why "playing in the paint" is an expression used referring to players who play close to the basket.

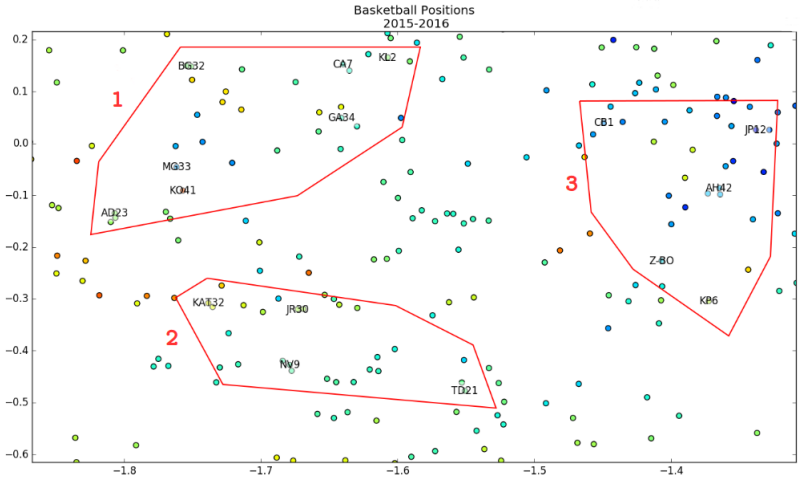

Figure 6.20: Zoom on the SR cluster of the map trained using 11 statistical categories.

reason of this result is probably the fact that 6 of the 11 statistical categories used to define the clusters are the same used in the previous experiments, so an important contribute to the SOM training and characterization is given by those categories which were already in use.

The new positions found training the SOM with 11 statistical categories are the same 5 presented in Section 6.2, but clusters are more separated and well defined on the output layer. Analyzing the single clusters it is also possible to notice how most of the players did not change position with the addition of the new statistical categories, which have not changed the relative distance between them in a significant way. For example players like LeBron James (LBJ23), Russell Westbrook (RW0) or Stephen Curry (SC30) are still mapped in the AAS cluster, Damian Lillard (DL0), Kyrie Irving(KI2) or Kobe Bryant (KB24) remained in the SB cluster, while DeAndre Jordan (DJ6) and Steven Adams (SA12) are still part of the PP position.

Anyway, analyzing the single clusters it is possible to see how the introduction of the new statistical categories actually had an effect on the clustering result of the SOM. Firstly the attention has to be focused on the SR cluster, which seems to be the one that better highlights the effect of the new statistical categories.

From Figure 6.19 it was already possible to notice an empty area with no player mapped in the middle of the SR cluster. Figure 6.20 reports a zoomed version of the cluster, and from this enlarged visualization it is possible to see how players are mapped in 3 main areas within the SR cluster. This particular feature allows to distinguish between players mapped in the SR position, which will probably have different statistical profiles and ways of playing, as was already pointed out in Section 6.2.1.

Table 6.11: Different Scoring rebounders

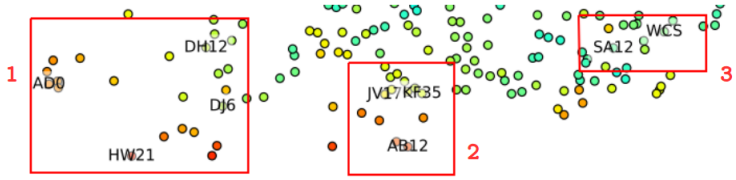

Figure 6.21: Zoom on the PP cluster of the map trained using 11 statistical categories.

PLAYER	ORB	DRB	3P_P	2P_P	FT_P	BLK	AST	STL	TOV	PF	PTS
Carmelo Anthony (1)	1.40	6.40	0.35	0.46	0.82	0.50	4.20	0.90	2.40	2.50	21.80
Blake Griffin (1)	1.50	6.90	0.40	0.50	0.73	0.50	4.90	0.80	2.40	2.70	21.40
Giannis Antetokounmpo (1)	1.40	6.20	0.29	0.54	0.72	1.40	4.30	1.20	2.60	3.20	16.90
Karl-Anthony Towns (2)	2.80	7.70	0.36	0.56	0.79	1.70	2.00	0.70	2.20	3.00	18.30
Tim Duncan (2)	1.90	5.40	0.00	0.49	0.71	1.30	2.70	0.80	1.50	2.00	8.60
Nikola Vucevic (2)	2.70	6.20	0.00	0.51	0.74	1.10	2.80	0.80	1.90	2.70	18.20
Chris Bosh (3)	0.90	6.50	0.36	0.51	0.78	0.60	2.40	0.70	1.50	1.90	19.10
Zach Randolph (3)	2.60	5.10	0.25	0.48	0.78	0.20	2.10	0.60	1.50	2.10	15.30
Al Horford (3)	1.80	5.50	0.35	0.56	0.81	1.50	3.20	0.80	1.30	2.00	15.20

Table 6.11 reports the statistical profiles of players mapped in the SR position, dividing them in the three sub-clusters shown in Figure 6.20. Comparing their statistical profiles it is possible to notice how players in the upper-left sub-cluster, labeled with number 1 have lower values in statistical categories like ORB or BLK, while having higher values in AST and TOV. This kind of statistical profiles describe players who have a better attitude playing far away from the paint, facing the basket and creating plays for teammates.

Players in the lower sub-cluster, labeled with number 2 instead have better statistics in ORB and BLK, while having poor 3PP statistics and lower numbers in AST. This sub-category of players are the SR who prefer to play closer to the basket, usually in low-post on the offensive end of the floor, taking advantage of their size. This features are more similar to the ones of PP players and indeed sub-cluster number 2 is the closest to the PP cluster.

Players in the third sub-cluster instead have average numbers in all statistical categories, with no particular feature which is prevalent in their statistical profile. Their statistics in AST, 3PP, ORB and BLK are not excellent but just average, their statistical profiles are only characterized by good numbers in scoring and rebounding categories, making them average SR players. Not having particular features apart from good scoring and rebounding statistics, their cluster is the one that is closer to the RP area, which contains players with similar characteristics, not shining in any particular statistical category.

Another cluster which presents similar features is the PP one, whose zoomed version is shown in Figure 6.21. From the figure it is possible to notice how the cluster was extended across the horizontal direction of the output layer and so the players were divided in 3 sub-clusters on the left and right extremes, and in the middle of the cluster.

Table 6.12: Different Paint Protectors

PLAYER	ORB	DRB	3P_P	2P_P	FT_P	BLK	AST	STL	TOV	PF	PTS
Andre Drummond (1)	4.90	9.90	0.00	0.52	0.36	1.40	0.80	1.50	1.90	3.00	16.20
Hassan Whiteside (1)	3.30	8.60	0.00	0.61	0.64	3.70	0.40	0.60	1.90	2.80	14.20
DeAndre Jordan (1)	3.50	10.30	0.00	0.70	0.44	2.30	1.20	0.70	1.40	2.70	12.70
Jonas Valanciunas (2)	3.10	6.10	0.00	0.57	0.75	1.30	0.70	0.40	1.40	2.60	12.80
Kenneth Faried (2)	3.50	5.20	0.00	0.56	0.61	0.90	1.20	0.50	1.40	2.50	12.50
Steven Adams (3)	2.70	3.90	0.00	0.62	0.56	1.10	0.80	0.50	1.10	2.80	8.00
Willie Cauley-Stein (3)	2.00	3.30	0.00	0.57	0.63	1.00	0.60	0.70	0.70	2.20	7.00

Table 6.12 reports the statistical profiles of some of the PP players mapped on the output layer of the SOM. Analyzing the statistical profiles some differences can be found to explain the different positions of players inside the cluster.

Players mapped in the left part of the clusters are reported in the first 3 rows of Table 6.12, their statistical profiles are characterized by high numbers in DRB, BLK and STL with respect to the other players, which are the statistical categories that should distinguish a good PP from players playing other positions, as explained in Section 6.2. What is more those players are better than the others in PPG, a feature that places them in the left part of the map, closer to the AAS cluster.

Players in the middle part of the clusters are reported in fourth and fifth rows and their statistical profiles have lower numbers with respect to players in the first three rows in almost all statistical categories, apart from FTP. Having good averages in FTP can be an important feature for players who play the PP position, since they are playing closer to the basket, where the game gets more physical, and the number of fouls is increasing. Being able to convert free throws into points with good percentages can be a big advantage for a PP player, who will probably have to shoot many free throws.

Finally players in the righten side of the cluster are reported in the last two rows of the table. In this case their numbers are lower than the other players in all the statistical categories and no particular feature is highlighted in their statistical profiles. A reason for this flat statistical profiles can be similar to the one given for players from sub-cluster 3 in the SR position, since the righten sub-cluster is the one which is closer to the RP position also in this case.

Increasing the number of statistical categories used to describe a player did not provide new results in terms of number of positions found on the output layer of the SOM. Anyway the effect of considering new statistics and so new data in the definition of positions had an effect at lower level, changing the shape of some of the single clusters describing positions. Adding new data to describe players and their ways of playing can highlight more and more aspects of their game, and even if the clusters will not be more separate between each other, their inner shape can change, showing some sub-clusters that can bring to the analytic definition of the different ways of playing the same position.

6.6 European Players Analysis

Once new positions of NBA players have been defined and analyzed, the final experiment made on our SOM application was trying to map players coming from a different league, to see if the mapping algorithm can be applied to players of any league.

As new reference data source, the statistics referring to the 2015-2016 Euroleague regular season have been considered. The Euroleague is the most important European basketball competition, in which all the best teams coming from the most important European Leagues face each other to win the most prestigious European title. All the best players coming from Europe, but also many American players who were not able to reach the NBA, play in the Euroleague competition.

Euroleague, like the NBA championship is divided in different phases: the first phase called regular season, in which all teams play against each other, and a second phase, the playoffs, in which the best teams play in a direct elimination format to reach the final.

Like in the case of the NBA statistics regarding the regular season have been used to define positions, to be able to include players coming from all the European teams involved in the competition for the positions definition.

The clustering process was performed following the same work-flow and algorithm used for the NBA clustering, and also the training sets for the different experiments were created using the criteria explained in Chapter 5.

As regards the statistical categories used to define players statistical profiles, a first trial was made using the 7 statistical categories taken from Alagappan's research, which gave good results for the NBA clustering.

Analyzing European players, the use of 7 statistical categories did not give good results in terms of clustering. The mapping on the output layer of the SOM provided an high number of players with similar relative distance between each other, making the definition of separate clusters too difficult and not significant.

To solve the problem another trial was made, using the 11 statistical categories described in Section 6.5, while maintaining the same training and input set.

Figure 6.22 shows the result of the clustering made using 11 statistical categories, which provides a much better outcome for positions definition. The neuron distribution is less sparse and cluster are easily recognizable and well divided between each other, even if there are still come empty areas, the mapping of players covers most of the output layer of the map, so it was possible to define new positions for European players.

The need of 11 statistical categories to obtain a proper clustering of European players comes from the differences between NBA basketball and the European one. Even if we are talking about the same sport, the way European teams play is very different from the American way of playing. First of all the game in the NBA is more physical and spectacular, so athleticism becomes a very important factor influencing the game. NBA players are for sure the best athletes between basketball players, and having such high level athletes turns in a consequent increase of the game pace. Playing at different levels of athleticism and rhythm of the game turns into different magnitude of statistics, since an higher number of possessions leads to higher values of any statistical categories, even considering the different amounts of minutes played and normalizing statistics according to it.

Moreover NBA players have a more "specialized" game, in the sense that American coaches prefer to coach players who have well defined responsibilities in the team, and can do very well some particular things; more than having complete players who can do many different things well, but nothing in an excellent way. On the other end European players are usually more complete and their game is usually less-specialized.

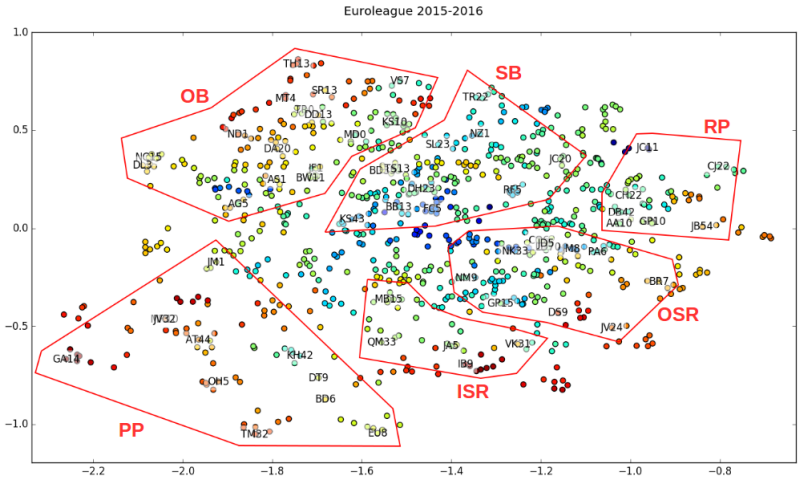

Figure 6.22: Positions of European players, described using 11 statistical categories.

This difference between the NBA and European basketball can be the reason why 11 statistical categories are needed to define European players positions. Having more complete players who are able to perform well in many aspects of the game turns into having statistical profiles which are not as well-shaped as the NBA ones. To find true differences between the way of playing of European players, more statistics are needed, to consider more aspects of the game and so be able to reconstruct a significant "shape" of their statistical profiles.

The result of the clustering made using 11 statistical categories is shown in Figure 6.22, and the positions we found in this case are 6 and not 5 anymore.

1. **Offensive Backcourt**. Offensive Backcourt (OB) is the first new category defined for European players. Players in this category have very good offensive skills, in particular scoring and passing, which make their statistical profiles characterized by high values in PPG and AST. They are usually the players who handle the ball in the decisive possessions, being able to make the best offensive play at any moment, to score themselves or assist a teammate in a better position. Examples of players in this category are Alessandro Gentile (AG5), Nando DeColo (ND1), Sergio Rodriguez (SR13) or Vassilis Spanoulis (VS7).

2. **Scoring Backcourt**. European SB players can be compared to American ones, since their responsibilities are very similar. SB players are more devoted to the scoring game with respect to the OB players whose offensive game is more complete, so SB players usually have lower values in AST statistics, while still being great scorers. Some players mapped in this position are Krunoslav Simon (KS43), Rudy Fernandez (RF5) or Sergio Llull(SL23).

3. **Outside Scoring Rebounder**. The SR position defined for NBA players is divided into two positions for European players. Outside Scoring Rebounder (OSR) are players who have statistical profiles mainly characterized by good numbers in PPG, ORB and DRB, as SR should. What makes them OSR is a better attitude for playing farther from the basket having good 3PP values.

They are players who can play physically close to the basket, but also being effective outside scorers. Players mapped in this category are Pero Antic (PA6), Dario Saric (DS9), Nicolo Melli (NM9) or Justin Doellman (JD5).

4. **Inside Scoring Rebounder**. Inside Scoring Rebounder (ISR) are players who have a more similar game to the definition of SR given for NBA players. Their game is characterized by high values in ORB, DRB and PPG, but they usually have low statistics in 3PP, while having higher values in 2PP, showing their attitude for playing closer to the basket. Players of this category are James Augustine (JA5), Miro Bilan (MB15) or Ioannis Bouroussis (IB9).

5. **Paint Protector**. The PP position is defined for players who prefer to play closer to the basket and have better defensive attitudes. They usually score less points with respect to ISR, but provide higher values in BLK and STL. Examples of PP are Ekpe Udoh (EU8), Bryan Dunston (BD6) or Othello Hunter (OH5).

6. **Role Players**. Finally there are RP, whose characteristics are similar to the players of the same position defined for the NBA. They can be players who have a more specialized game, like outside shooters, or even defensive specialist, whose impact on the game is not reflected by statistics. They are classified together because their statistical profiles are not so well-shaped. Examples of European RP are Davis Bertans (DB42), Charles Jenkins (CJ22) and Josh Carter (JC11).

Even if the positions defined for European players are not so different from the ones defined for NBA ones, some slight difference was found. In Europe it is more difficult to find players like AAS who are outstanding in all statistical categories or are much better than all the other teammates. European players are more complete and so usually each team bases its hopes of success on 4 or 5 fundamental players, while it is more difficult to find a team build around the performance of 1 or 2 superstar players, like in the NBA.

Figure 6.23 shows the result of the forward-mapping of all the players included in the training set, used to define European players positions. As in the case of the NBA mapping of all the training set players, explained at the end of Section 6.2, this kind of result have been shown after the positions definition because of labels overlapping problems. Also in this case anyway, the same clusters defined using a limited input set were found on the map containing all the players, even if there are some players who are mapped between different clusters.

Having less-specialized players made it more difficult to find proper clusters to divide them, so a good idea for future applications of this tool may be considering more advanced statistical categories during the clustering process, to be able to better describe all the aspects of players game.

Moreover this result shows how leagues around the world may have different characteristics. This is a very important point to keep in mind if the application will be used for defining positions of other leagues, since probably each league will have different parameters to be considered, and different levels of details in the statistical profiles will be needed to define proper positions for players of that league.

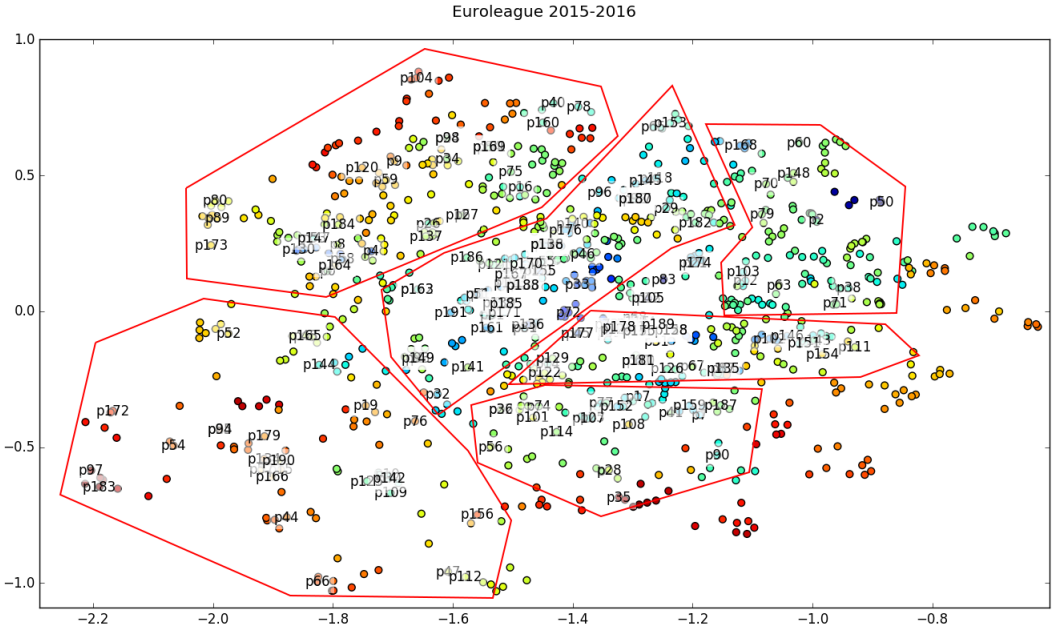

Figure 6.23: All players of the European league training set, mapped on the output layer.

Chapter 7

Conclusions

In this thesis an original method based on the use of SOM for redefining basketball players positions was presented. The goal of the thesis was to obtain new positions for basketball players which were able to better describe their way of playing with respect to the classic ones in use since the invention of the game.

The main source of inspiration comes from a 2012 research by Muthu Alagappan, a former Stanford student who, during an internship experience in a big data startup, applied the tools of the startup to analyze the differences between basketball players and their statistical profiles, redefining the positions in basketball. He has done his work on a dataset of statistics referring to regular season games of the NBA 2010-2011 season, so we decided to use the NBA regular season statistics as input space for our application too. This decision is due to the fact that official NBA statistics are easy to retrieve and the NBA is the most famous league in the world, so it is possible to make the research understandable and open to more people dealing with data referring to the most famous and well-known players in the world. Moreover, by applying our techniques on the same league Alagappan studied, it was possible to compare our results with his ones, even if the purpose of the thesis was not just validate Alagappan's research, but create our own new basketball positions.

Once the PyMVPA library for SOM implementation was found and tested, it was possible to start working on data of the input space. Finding a proper training set to feed the SOM, to obtain clusters that describe all the ways of playing basketball was not a trivial task, since feeding the network with all the players in the league did not provide satisfying results, as described in Section 5.2.1. Training set definition was an important part of the work and it took some experiments to find which one was including the highest possible number of players whose statistical profiles were relevant for positions definition.

After the best training set has been found, a proper input set had to be created for the forward mapping of players on the output layer of the SOM. It was an important task to select the proper players to be mapped, since it was important to select both players that were included in the training set and players who were not included, to see if the result was working for all the players in the league.

At this point the most difficult part of the work of thesis started: data visualization. The SOM algorithm performs the clustering by just tuning the weights associated to the neurons of the output layer, but not assigning particular colors or positions to neurons, to actually show the effect of the clustering. The PyMVPA module

which implements the SOM does not include a standard visualization method, so a proper visualization technique was needed to be able to analyze the quality of results obtained from the clustering. Since the output layer of the SOM is an high-dimensional space, as described at the beginning of Chapter 3, new technologies for the visualization of high-dimensional data had to be studied and integrated in our application.

To solve this problem, MDS and PCA technologies were applied, as described in Section 3.2 and Section 3.3. Both technologies allow to reduce dimensionality of an high-dimensional input space in a lower-dimensional output space, maintaining the topological features of the input space as much as possible. The dimensional reduction allows to display the output layer of the SOM, and so the results of the clustering, using standard Python libraries like the matplotlib described in Section 4.1.2, using a simple two-dimensional scatter plot representation. The visualization problem was probably the most difficult one to solve and also the most time expensive, since I had no experience dealing with high-dimensional datasets, their representation and the technologies used to solve this kind of problems.

After the implementation of all the software modules needed to perform the whole SOM algorithm and visualization the tests to define the new positions started. At first the low quality of classic positions describing the way modern players play the game was proved as described in Section 6.1, then it was possible to define the new basketball positions.

The result was surprising in terms of number of positions found, since the new positions defined by the SOM were 5, exactly the same number of the classic ones. Anyway analyzing the statistical profiles of players included in each cluster it was possible to discover how first of all the new positions are different from the classic ones, and how they are describing the way a player plays the game in a more accurate way, according to his statistics. Analyzing the relative distance between players it is possible to assign a new player to one of the new positions, comparing his position to the one of the closest well known players mapped on the output layer and already assigned to the new positions. The analysis of the single positions showed how, even if they are better describing the way players play, it is always possible to improve the level of detail in the description. Checking the relative distance of players of the same cluster, it is possible to find different ways players play a certain position. This feature does not allow to increase the number of defined positions with new additional roles, but it is an effective way to understand all the shades of a player game.

To validate the results an analysis on past seasons has been performed, and actually the same positions were found. Having data related to past seasons, it was also possible to analyze the evolution of single players across different seasons, which can be a very important tool of analysis for coaching staffs and recruiters to evaluate the growth of a player through the years.

7.1 Future Development

Even if the results of the work were quite satisfactory, any project or research activity can be improved. In this case there are different future possible developments and improvements which can be implemented.

First of all the application should be integrated in the MYAgonism platform, since the work of thesis starts with the collaboration with the Pavia startup, which provided the starting idea and followed the whole development. According to their needs, the application can be slightly changed to obtain the most useful results for coaches and people who work with the MYAgonism application and web environment. A direct consequence is the application of the SOM algorithm to different datasets and so to players profiles coming from different basketball leagues. The MYAgonism platform is used by coaches at any level, not only major leagues coaching staffs, but also people working for collegiate teams or minor leagues teams. It is important to give to all of them the definition of positions which are able to describe their players in a proper way with respect to the environment in which they are, and so with respect to the players of their league. If we map a minor league player in a SOM trained with NBA players statistics, his position will not be correct, since the level of the league in which he plays is different from the NBA and so statistical profiles of players will have different "shapes".

To improve and extend the analysis the statistical categories used to describe players can be put in discussed. The ones used during this work of thesis are simplest ones which are available for any league and championship in the world, but they are also the ones that describe the most fundamental aspect of the game, like scoring, rebounding, passing and defensive aspects of the game. The MYAgonism platform provides this kind of statistical categories for all the leagues and players who are recorded in the system, but integrates these statistics with other more sophisticated data used to give high-level analytic tools to coaches. An interesting experiment could be to use those sophisticated statistics to create players profiles and feed the SOM with a training set including those advanced data, to see if the clustering and the positions definition changes. Having more sophisticated statistics and a bigger amount of data may help considering aspects of the game which cannot be considered using only the simpler statistics we used in this work of thesis.

An example of those advanced statistics is the Offensive Efficiency Rating (OER) which can be computed by analyzing a player performance during a game. The OER is defined as:

$$OER = \frac{Points\_Scored}{OP} \tag{7.1}$$

Where OP refers to the number of offensive possessions played by the player during the game. The OP can be computed too, as the sum of all the statistics referring to offensive possessions during the game:

$$OP = 2PFG + 3PFG + (FT/2) + TOV \tag{7.2}$$

In Equation 7.2, the term $2PFG$ refers to the number of 2 points field goal attempts; $3PFG$ refers to the number of 3 points field goal attempts; FT is the number of free throw attempts, it is divided by 2 because usually a player shoots 2 free throws each time, so in the OP evaluation the number of times a player goes to the free throw line is counted; finally TOV is the number of turnovers in the game. The OER is a measure of the effectiveness of offensive possessions of a player, since it gives a measure of the number of points each possession of the player provided to the team. Derived data of this type are not always available in official league

data sources, but the MYAgonism platform provides them to players and coaches subscribing to it.

Apart from changing the statistical categories involved in the positions definition process, other future expansions of the application can be done adding functionalities which allow better players comparisons and a deeper analysis of players mapped in the same position. Analyzing the statistical profiles of players in the same position it is possible to better understand the difference between them, as explained in Section 6.5, so having an automatic functionality or tool that allows to visualize those differences would be very useful to coaches for players comparison.

Another possible future development can be a new visualization technique, since the used ones give good results but having the possibility to visualize the output layer of the SOM in a more precise way may reveal aspects of the clustering process that cannot be noted from the dimensional reduction that MDS and PCA apply.

7.2 Coaching Application

As already mentioned, the result of this thesis has the purpose to be a tool for coaches to better understand the way players play the game, without the need of knowing them. It can be useful for a coach to analyze those data in order to find the players who better adapt to his basketball philosophy.

A proposed application which can be implemented as a future expansion of this work of thesis is a new functionality which suggests the player a coach should recruit to be the best fit for his team.

The idea is to start from the coach basketball philosophy and find those players, or better those positions, that fit the way a coach wants his team to play. Given a coach philosophy, which should be defined by some input parameters the coach can specify, the system should provide a solution, giving to the coach a starting five made by players taken from the new positions, the ones who best fit his idea of basketball.

A first example of basketball philosophy can be the so called "Run & Gun"[1], preferred by coaches who want a very offensive team, which plays at high pace, trying to take a shot in the first seconds of the offensive possession and playing a lot of transition game to increase the pace of the game. Usually teams who play the "Run & Gun" model do not have very good defenses, take a lot of three point shots and do not have really complex offensive systems, since taking a fast shot means exploiting the first good solution that comes during a possession, and it usually comes from the instinctive plays of players on the floor.

Figure 7.1 shows a possible output of the application, given to a coach who wants to see his team playing the "Run & Gun" philosophy. On the half-court a possible starting five is displayed, composed by one AAS player, two SB players, one PP and a RP.

[1]The "Run & Gun" philosophy takes his name from the way teams following this philosophy play, since players of these teams usually run a lot up and down on the floor and take many shots from outside in the first seconds of the offensive possession. A famous example of team following this philosophy are the Phoenix Suns of the seasons from 2003-2004 to 2007-2008, whose coach was Mike D'Antoni.

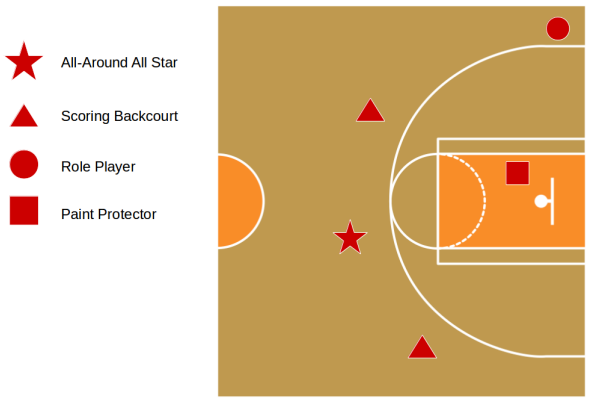

Figure 7.1: Example of output with the selection of players who fit the "Run & Gun philosophy".

The idea is to have five players on the court who are able to play at high-pace, increasing the number of possessions and having good scoring abilities. Having an AAS player on the court give to the team the scoring potential it needs, since usually AAS players are the best scorers in the league.

The two SB players have the responsibility to help the team with their natural offensive attitude, since their way of playing is characterized by good outside shooting and passing skills, which can help the team find a good offensive solution in the first seconds of the possession. The addition of a RP gives the possibility to adapt the five players on the court according to the opponents.

Since players playing the RP position are usually defensive specialists or pure shooters, selecting a proper RP a coach can decide to deploy a more balanced lineup to face the menace of good offensive players of the opposite team, or boost the offensive potential of his team, adding another shooter to the lineup.

Finally a PP is fundamental to guarantee rebounding potential and defensive presence to the lineup, what is more having an AAS, two SB and a RP the lineup needs a player who increases the average size, to face bigger players in the opponent team. The idea of choosing a PP and not a SR to complete the lineup comes from the fact that PP usually do not need to play many offensive possessions with the ball in their hands, so they do not need to stop the ball movement to be effective on the offensive end of the floor. Moreover they are usually better athletes with respect to some SR, so they can run the floor better, as required by the high pace of the team.

Opposite to coaches who prefer the "Run & Gun" philosophy there are other coaches who want their team to play a more organized kind of basketball, based on solid defense, and many precise and organized offensive solutions. We can call this way of playing basketball the "Defensive System" to simplify and oppose it to the "Run & Gun" philosophy. Teams following this defensive basketball philosophy play at slower pace, trying to stop any offensive solution of the opponent team for 24 seconds on the defensive end of the floor, and searching for the best offensive

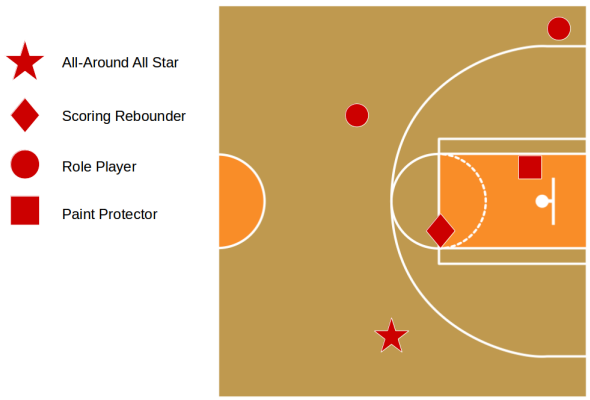

Figure 7.2: Example of output with the selection of players who fit a defensive system team.

solution between the many ones proposed in the complex schemes the coach designed on the offensive end.

Figure 7.2 shows a possible output given to a coach who wants to create a team based on strong defense and precise execution on offense. The lineup shown in the figure is composed by one AAS, two RP, one SR and a PP.

Having an AAS in the team is always a good selection, if it is possible to recruit a player of this category. In this case his responsibility is to be the main offensive resource of the team, scoring many PPG, but also creating offensive solutions for his teammates.

The SR should be the second offensive terminal of the team, giving an inside-scoring solution to be exploited during the schemes execution. Having two different offensive threats, one from outside and one from inside the paint, can be a very important feature for a team which wants to play an organized game, with many different offensive solutions to be exploited during the possessions, which usually last longer than the ones of a "Run & Gun" team.

The PP has very similar responsibility with respect to the case of the "Run & Gun" team, since he should be the best rebounder of the team, along with the SR, with a defensive minded attitude in his game, blocking shots and helping teammates reject opponents offensive initiative inside the paint.

Finally two RP have been included in the lineup to give versatility and more solutions to the coach. As we already mentioned and explained in Section 6.2, RP can be defensive minded players who play farther from the basket, or outside shooters who score mainly from teammates creations. Both kind of players can fit a "Defensive System" lineup, according to the needs the coach has in the different situations. Adding outside shooters can be a good solution to increase the offensive potential of the team, since they can exploit the space given by a defense which will concentrate its attention on the two primary offensive terminals, the AAS and the SR. A good scheme should provide a solution which exploits the offensive attitude of the AAS and SR to create a good shot for one of the RP, surprising the defense. On the other

hand, having defensive minded RP allow the team face the more offensive lineups of the opponents, but also increases the athleticism of the lineup, enabling fast break and open field offensive solutions coming from steals and ball recoveries the defensive RP guarantee.

Providing this kind of instrument to a coach allows to design the perfect lineup for the type of game he has in mind at any moment, even during a game, to face some particular situations or change the game trends changing the way his team plays. The immediate feedback given by a tool like the one presented can be useful and effective only if the positions used to compose the lineups are describing the way players play in a proper way, as our new positions do.

Bibliography

[1] NBA.com/stats. http://stats.nba.com/, Last Access 2016.09.02.

[2] NBA.com/stats On/Off court comparison. http://stats.nba.com/vs/#!/?PlayerID=203081&VsPlayerID=201599, Last Access 2016.09.02.

[3] SOMz: Self Organizing Maps and random atlas. http://matias-ck.com/mlz/somz.html, Last Access 2016.08.20.

[4] Muthu Alagappan. From 5 to 13: Redefining the Positions in Basketball (2012 EOS/Alpha Award Winner). http://www.sloansportsconference.com/content/the-13-nba-positions-using-topology-to-identify-the-different-types-of-players/, 2012. [Online; accessed 20-August-2016].

[5] Suwardi Annas, Takenori Kanai, and Shuhei Koyama. Principal component analysis and self-organizing map for visualizing and classifying fire risks in forest regions. *Agricultural Information Research*, 16(2):44–51, 2007.

[6] Daniel Arribas-Bel, Karima Kourtit, and Peter Nijkamp. Benchmarking of world cities through self-organizing maps. *Cities*, 31:248 – 257, 2013.

[7] Daniel Arribas-Bel and Charles R Schmidt. Self-organizing maps and the us urban spatial structure. *Environment and Planning B: Planning and Design*, 40(2):362–371, 2013.

[8] Holly J. Atkinson, John H. Morris, Thomas E. Ferrin, and Patricia C. Babbitt. Using sequence similarity networks for visualization of relationships across diverse protein superfamilies. *PLoS ONE*, 4(2):1–14, 02 2009.

[9] Bojana Bislimovska, Günes Aluç, M Tamer Özsu, and Piero Fraternali. Graph search of software models using multidimensional scaling. In *EDBT/ICDT Workshops*, pages 163–170, 2015.

[10] Scoot Bruce. A scalable framework for nba player and team comparisons using player tracking data. *Journal of Sports Analytics*, Preprint(Preprint):1–13, 2016.

[11] Rocio Chavez-Alvarez, Arturo Chavoya, and Andres Mendez-Vazquez. Discovery of possible gene relationships through the application of self-organizing maps to dna microarray databases. *PLoS ONE*, 9(4):1–13, 04 2014.

[12] Andrew Corsello. The Stunning, Strange, Beautiful Game of Manuel Neuer. http://www.espn.com/espn/feature/story/_/page/enterprise-neuer160525/bayern-munich-manuel-neuer-changing-means-goalie, 2016. [Online; accessed 19-August-2016].

[13] Michel Herbin Cyril de Runz, Eric Desjardin. Unsupervised visual data mining using self-organizing maps and a data-driven color mapping. *IEEE Transactions on Computers*, 2012.

[14] Nathan Desdouits, Michael Nilges, and Arnaud Blondel. Principal component analysis reveals correlation of cavities evolution and functional motions in proteins. *Journal of Molecular Graphics and Modelling*, 55:13 – 24, 2014.

[15] L.P. Graham D.J. Bokor, A. Sundaram. Influence of field position on rugby league players requiring shoulder reconstruction. *Int J Sports Med*, 37(6):489–492, 2016.

[16] Paul Flannery. Draymond Green is redefining NBA stardom. Even he didn't see that coming. http://www.sbnation.com/2016/2/16/10987022/draymond-green-warriors-nba-unexpected-star, 2016. [Online; accessed 19-August-2016].

[17] Sascha Klement Erhardt Barth Thomas Martinetz Foti Coleca, Andreea State. Self-organizing maps for hand and full body tracking. *Neurocomputing*, 147:174 – 184, 2012. Advances in Self-Organizing Maps Subtitle of the special issue: Selected Papers from the Workshop on Self-Organizing Maps 2012 (WSOM 2012).

[18] Edin Uzicanin Melika Muratovic Samir Mackovic Haris Pojskic, Vlatko Separovic. Positional role differences in the aerobic and anaerobic power of elite basketball players. *Journal of Human Kinetics*, 49(3):219–227, May 2015.

[19] Michael C. Hout, Hayward J. Godwin, Gemma Fitzsimmons, Arryn Robbins, Tamaryn Menneer, and Stephen D. Goldinger. Using multidimensional scaling to quantify similarity in visual search and beyond. *Attention, Perception, & Psychophysics*, 78(1):3–20, 2016.

[20] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[21] John Ingraham Alexander Petersen James S. Waters Jennifer H. Fewell, Dieter Armbruster. Basketball teams as strategic networks. *Public Library of Science*, 7(11):e47445, May 2012.

[22] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, Sep 1990.

[23] S. Hofstetter M. Lames L. Leventer, F. Eek. Injury patterns among elite football players: A media-based analysis over 6 seasons with emphasis on playing position. *Int J Sports Med*, 11(3):11, May 2016.

[24] Andrew C.-D. Lee and Claus Rinner. Visualizing urban social change with self-organizing maps: Toronto neighbourhoods, 1996-2006. *Habitat International*, 45, Part 2:92 – 98, 2014. Special Issue: Exploratory Spatial Analysis of Urban Habitats.

[25] Philippe Louis, Alex Seret, and Bart Baesens. Financial efficiency and social impact of microfinance institutions using self-organizing maps. *World Development*, 46:197 – 210, 2013.

[26] Lehman A. Ishkanov T. Vejdemo-Johansson M. Alagappan M. Carlsson J. Carlsson G Lum P. Y., Singh G. Extracting insights from the shape of complex data using topology. *Scientific Reports*, 3(1236), 2012.

[27] Zain Mahmood. How 'The Makelele Role' redefined English football. `http://www.sportskeeda.com/football/how-the-makelele-role-redefined-english-football`, 2015. [Online; accessed 19-August-2016].

[28] Sam Monson. Mobile Quarterbacks Redefining the Position. `http://bleacherreport.com/articles/1251778-mobile-quarterbacks-redefining-the-position`, 2012. [Online; accessed 19-August-2016].

[29] Rebecca M. Page, Peter Huggenberger, and Gunnar Lischeid. Multivariate analysis of groundwater-quality time-series using self-organizing maps and sammon's mapping. *Water Resources Management*, 29(11):3957–3970, 2015.

[30] J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, 1969.

[31] J. Vesanto and E. Alhoniemi. Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, 11(3):586–600, 2000.