Measure concentration in computer vision applications

Thesis submitted for the degree of
Doctor of Philosophy
at the University of Leicester

By
Konstantin Sofeikov
Department of Mathematics
University of Leicester

October 2017

Abstract

We live in the Information Age. In this age technological industry allows individuals to explore their personalized needs, therefore simplifying the procedure of making decisions. It also allows big global market players to leverage amounts of information they collect over time in order to excel in the markets they are operating in. Huge and often incomprehensive volumes of information collected to date constitute the phenomenon of Big Data. Big Data is a term used to describe datasets that are not suitable for processing by traditional software. To date, the commonly used way to get value out of Big Data is to employ a wide range of machine learning techniques. Machine learning is genuinely data-driven. The more data are available the better, from statistical point of view. This enables creation of an existing range of applications for broad spectrum of modeling and predictive tasks.

Traditional methods of machine learning (e.g. linear models) are easy to implement and give computationally cheap solutions. These solutions, however, are not always capable to capture the underlaying complexity of Big Data. More sophisticated approaches (e.g. Convolution Neural Networks in computer vision) are show empirically to be reliable, but this reliability bears high computational costs. A natural way to overcome this obstacle appears to be reduction of Data Volume(the number of factors, attributes and records). Doing so, however, is an extremely tedious and non-trivial task itself.

In this thesis we show that, thanks to well-known concentration of measure effect, it is often beneficial to keep the dimensionality of the problem high and use it to your own advantage. Measure concentration effect is a phenomenon that can only be found in high dimensional spaces. One of theoretical findings of this thesis is that using measure concentration effect allows one to correct individual mistakes of Artificial Intelligence(AI) systems in a cheap and non-intrusive way. Specifically we show how to correct AI systems errors with linear functional while not changing their inner decision making processes. As an illustration of how one can benefit from this we have developed Knowledge Transfer framework for legacy AI systems. The development of this framework is also an answer to a fundamental question: how a legacy "student" AI system could learn from "teacher" AI system without complete retraining. Theoretical findings are illustrated with several case studies in the area computer vision.

Preface

This thesis is a product of 4 years long journey I had a privilege and pleasure to take at the Department of Mathematics in the University of Leicester. Being an externally funded industrial project, this work was motivated initially by real-life problems arising from various computer vision applications. Later, however, it has become clear that these problems lead to interesting theoretical generalizations. This interface made the work particular interesting and rewarding for the author. The main focus of this work is in dealing with high-dimensional data and challenges associated with it.

This thesis consists of 4 chapters. In the introductory Chapter 1 we discuss what does Big Data mean in the modern world as well as what challenges does it bring. Not only Big Data brings its challenges and issues, it also brings benefits if used correctly.

Chapters 2 and 3 discuss sources of errors in Big Data Analytical Systems. These errors are exemplified with two most commonly used approaches to deal with large volumes of data in presence of uncertainty: machine learning and dimensionality reduction. In particular, it contains an overview of machine learning problem statement[1] and description of issues one has to face when dealing with big volumes of data. We also give a brief introduction to Curse and Blessing of Dimensionality terms.

Chapter 4 contains main theoretical body of the work. In includes separation theorems and computational algorithms that are built on these theorems. We also show what place do these approaches take in the context of modern Artificial Intelligence theory. In addition, we show what practical applications do these results find in the field of modern computer vision.

[1]Problem statement is adopted from [85]

Chapter 5 demonstrates how results from Chapter 4 can be used in real world applications. Several case studies are included here. These examples are arranged in the order of increasing complexity.

Acknowledgments

First and foremost I wish to thank my supervisor Dr. Ivan Tyukin. I still can not imagine how lucky I was to be his student. I can't express how grateful I am for his support and encouragements during our numerous meetings in his office. One can always feel his passion for knowledge and science and what is truly beautiful, he is able to share his passion with you.

I would also like to thank Prof. Alexander Gorban, my co-supervisor, for introducing me to the field of Artificial Intelligence and Data Analysis. I'm deeply indebted to him for numerous technical and deep scientific discussions, which we had over the course of the project. Those were truly amazing meetings.

Special thanks go to Dr. Ilya Romanenko, who was my line manager at Apical Ltd and ARM Ltd. He is the most talented and perhaps the best engineer I have ever met.

My appreciation also goes to Prof. Valery Terekhov thanks to whom I got interested in applied mathematics. To my parents Igor and Natalia for their trust in me and all the effort they have put into to rise me. Finally, I want to thank my adorable wife Viktoria for her patience, love and support she has surrounded me with.

Contents

Abstract **1**

Preface **2**

Acknowledgments **4**

Notation **7**

1 Introduction **8**

2 Source of errors in Big Data Analysis Systems **14**

2.1 Machine learning problem statement 14

2.2 Issues of Machine Learning 17

2.3 Specific issues of Machine Learning in Computer Vision domain . . . 18

2.4 High dimensional data issues 19

2.5 Curse of dimensionality 20

3 Dimensionality reduction **22**

3.1 Full spectral Techniques 23

3.2 Sparse Spectral Techniques 28

3.3 Non-convex Techniques 30

3.4 Random projections 32

3.5 The blessing of dimensionality 33

4 Blessing of dimensionality in Pattern Recognition **39**

4.1 Statistical properties of high-dimensional data 39

4.2 Measure concentration revisited 41

4.3 Almost orthogonality in high dimensions 46

4.4 Stochastic separation theorems 54
4.4.1 Extreme points of a random finite set 57
4.4.2 Two-functional (two-neuron) separation in finite sets 58
4.5 Knowledge transfer between legacy Artificial Intelligence systems . . . 65
4.5.1 K-tuple separation theorems 66
4.5.2 Artificial Intelligence Knowledge Transfer Framework 74
4.5.3 Knowledge Transfer Algorithms 76

5 Case studies **80**
5.1 Low dimensional data - simple classifiers 80
5.2 Low dimensional data - more advanced classifier 86
5.3 Knowledge transfer in Artificial Intelligence systems 96

Conclusion **102**

Appendices **104**

A A very brief history of Machine Learning **105**

B Ill-posed problems **109**

Notation

Throughout the thesis the following notational agreements are used

- $\mathbb{N}$ is the set of natural numbers;
- $\mathbb{R}$ denotes the field of real number;
- $\mathbb{R}^n$ stands for the n-dimensional real space; unless stated otherwise symbol n is reserved to denote dimension of the underlying linear space;
- let $x \in \mathbb{R}^n$, then $\|x\|$ is the Euclidean norm of x: $\|x\| = \sqrt{x_1^2 + \cdots + x_n^2}$;
- if x, y are two non-zero vectors from $\mathbb{R}^n$ then $\angle(x, y)$ denotes the smallest angle between these vectors;
- the symbol $|\cdot|_{\mathbb{R}^n}$ is reserved to denote an arbitrary norm in $\mathbb{R}^n$;
- $\mathcal{S}^{n-1}(R)$ denotes an $n-1$-sphere of radius R centred at 0: $\mathcal{S}^{n-1}(R) = \{x \in \mathbb{R}^n |\ \|x\| = R\ \}$;
- μ is the normalized Lebesgue measure on $\mathcal{S}^{m-1}(1)$: $\mu(\mathcal{S}^{m-1}(1)) = 1$;
- $\mathcal{B}^n(R)$ denotes a n-ball of radius R centered at 0: $\mathcal{B}^n(R) = \{x \in \mathbb{R}^n |\ \|x\| \leq R\}$;
- $\mathcal{V}(\Xi)$ is the Lebesgue volume of $\Xi \subset \mathbb{R}^n$;
- $\mathcal{D}^{n-1}(R)$ stands for a $(n-1)$-disc in the n-ball $\mathcal{B}^n(R)$ corresponding to its largest equator, and $\mathcal{D}_\delta^{n-1}(R)$ is its δ-thickening;
- Let $f : [0,1]^d \to \mathbb{R}$ be a continuous function, then

  $$\|f\|^2 = \langle f, f \rangle = \int_{[0,1]^d} f(x) f(x) dx,$$

  denotes the L_2-norm of f;
- "iid" stands for "independent and identically distributed";
- $\mathcal{M}$ is an i.i.d. sample equidistributed in $B_n(1)$;
- M is the number of points in $\mathcal{M}$, or simply the cardinality of the set $\mathcal{M}$.

Chapter 1

Introduction

Every day massive amounts of data are created. These data have to be captured, stored, analyzed and processed. While the amount of data collected has a potential to help to "spot business trends, prevent diseases, combat crime and so on" [28], it also gives rise to new challenges in data analysis. Big Data is the term used to describe datasets that are too large or complex for traditional data processing software. Big Data gives rise to new challenges which are often referred as the 5V of Big Data [40]:

1. Volume. This characteristic refers to sheer quantity of data. In this context, Big Data is synonymous to volumes of data processing which are beyond the ability of current state-of-the art tools and methods [45]. The data here includes not only the actual data samples at hand but also the parameters required to identify and describe the data samples.

2. Variety. Variety refers to datasets which have broad range of data types and formats. Examples are the datasets including simultaneous combinations of e.g. business transactions, information from social media platforms, sensors readings, and other heterogeneous streams of data.

3. Velocity. The velocity parameter of Big Data points to extraordinary speed with which data are generated and delivered.

4. Veracity. Veracity refers to uncertainty that is associated with the data. Examples include incorrect or noisy measurements, incomplete and missed data.

5. Value. Value is an indicator of costs and gains that Big Data bears. Examples of costs are, for example, infrastructural costs. Without an adequate infrastructure it is very hard to store and process large amounts of data. Gains can be exemplified as increasing understanding of customers needs for companies, improved health care and security.

Systems that have characteristics of Big Data already exist. Despite the fact that building these systems rises very serious challenges, they find a lot of applications in modern world. For example, governments can process Big Data for learning how electorate responses on government's actions. Big Data is collected in banking and securities domains. Applications of Big Data systems in these domains include card fraud detection, tick-level market data analysis and credit risk analysis. In media sphere Big Data is stored and analyzed on order to better understand customer insights. If done correctly, this allows one to give good content recommendations and deliver right content to a right target audience.

There is another area where Big Data systems only start finding its applications: video surveillance domain[1]. CCTV cameras deployed world wide are natural source of Big Data. It combines at least 4 out of 5 V's described above: Volume, Variety, Velocity and Value [17]. Recent progress in data storage development and networking interfaces allows one to address most of these challenges. Video from its sources can be quickly delivered, compressed and stored. However, to date there are no *efficient* ways to analyze video streams coming from these cameras. Effectively, if some information is to be found in a video stream, either a person has to seat and watch it in real-time, or hundreds hours of CCTV footage has to be looked through almost in manual mode. Of course, nowadays CCTV footage is only recorded if there is movement happening, but this almost does not help to solve the problem. If a camera is installed in a public space(e.g. public transport, shopping malls, etc.) then this approach is not helpful at all: there is always something happening.

During the past 2-3 years the market has started seeing new products created specifically to address this type of issue. Companies developing these solutions aim

[1]The first CCTV system was installed in year 1942 in Germany [23]. Back then it was used to observer launch of V-2 rockets and had extremely limited amount of installations. First widely-deployed cameras were used by the police to fight crime and by businesses that were prone to theft. To date there are about 250 million CCTV cameras installed world-wide [48] and about 6 million cameras in the UK alone.

to provide highly efficient(in terms of processing speed and power consumptions) solutions at a low price. Low price, low power consumption and high processing performance will allow the deployment of these products in each CCTV camera at almost no additional cost. Usually the main goal these companies are trying to achieve is to make CCTV footage analysis easier by only recording valuable pieces of video streams. Since it is not enough to switch on recording when *something* is happening, one has to analyze *what* is happening. Scene analysis may include people detection and tracking (including both full human figure and face detection), car detection, object recognition (including face or number plates recognition).

There is also hidden complexity in this task: the signal coming from cameras is always in RAW format. In this context RAW image means raw signals coming from camera sensors, Bayer pattern or mosaic pattern. All three are used interchangeably here. This means that before anyone is able to perform any computer vision task, one has to transform incoming images from RAW to RGB format. In order to get correct colors in the scene, besides bayer pattern interpolation, one also has to do color and gamma correction and automatic white balance setting . If colors are not set up properly, computer vision algorithms are not guaranteed to work adequately.

Let us exemplify some particular difficulties appearing in computer vision systems:

1. Pixelwise image processing tasks (Automatic White Balance(AWB) setting, image quality adjustment). In this task for a standard High Definition image one deals with almost 6 million pixels at a time. In this kind of task, operations on pixels are supposed to be done in real-time mode. For example, tasks of AWB setting and image quality adjustments are done on the camera's side therefore creating serious challenges for on-camera algorithms.

2. Object detection in live video streams. A video stream is a sequence of still images (frames) that are often received at a high frames per second (FPS) rate. Practically, the complexity of this task is at least FPS times more than the image processing task. One of the difficulties in these tasks is the number of proposals being tested for each frame. In some applications the number of proposals can be up to several hundred thousands. There is added complexity as well: in the majority of video processing applications requirements are very

high. For example, computer vision systems are often installed on autonomous cars where mistakes made by this kind of systems may lead to injuries and casualties.

3. Object recognition. A natural extension for object detection in live video streams task. Not only does it include the complexity of a previous task, but also has its own: the number of identities. For example, in a face recognition task one face has to be recognized among thousands of others. Combined with high precision requirements for biometric systems this makes the problem of object recognition a Big Data task.

As we can see, smart CCTV footage analysis employing computer vision techniques certainly a Big Data task with at least 4 of 5 Vs associated with it. Given the complexity of the task, different types of errors in these systems are not uncommon. While, errors in performing the pixelwise image processing task come at almost no price, errors made in scene analysis may have different consequences. If the computer vision system is used on a CCTV camera, false or missed detections may lead to false alarms. However, if a computer vision system is installed on an autonomous car, bad decisions made by an artificial intelligence vision system may lead to driver death [35].

While any system works well, almost no one asks why does it work well. However, should any problem occur, where intelligent systems are to be blamed, serious discussions on how does one avoid these and similar in the future arise. The very general question here is why do these artificial intelligence systems malfunctions happen? Of course, there is no simple answer. However, some clues can be found if one looks closer at how these systems work. Almost any decision being made in this kind of systems is based on many parameters, often thousands of them. As was said, quite often there are serious requirements for processing performance of these systems. That means that one has to sacrifice the complexity of decision making systems in order to get higher processing speeds. What happens next is that not always a simpler system is capable of understanding the underlying logic of the relationships between that many parameters.

As we can see, big data volumes and number of parameters (also called *dimensionality*) involved into a decision making process cause problems that somehow

need to be addressed. There are two fundamental ways to do it. The first and the simplest one is to eliminate the problem itself: do not use systems that rely on Big Data. The second way is to accept that Big Data is a reality and try to do something to address some of the issues described above. In this thesis we chose the second path. Therefore, **the main goal of this thesis** is to develop tools and methods that benefit from high dimensionality rather than suffer from it.

The following steps were identified as crucial for achieving the goal of the thesis.

1. Investigate how measure concentration effects can be used in various computer vision applications.

2. Develop and investigate usability of multilayer Artificial Intelligence system's correctors proposed by the authors in [31].

3. Develop a practical approach for knowledge transfer between two Artificial Intelligence systems. Usability of the approach developed must not depend on the structure and nature of computations in the Artificial Intelligence system being corrected.

Methods. This thesis makes use of the following domains of knowledge: linear algebra, statistics, data mining, machine learning framework, including Discriminant Analysis, Support Vector Machines and Artificial Neural Networks. Software development has mainly been done in the Matlab environment. Also, the following third-party software packages were used: libsvm [16], liblinear [71]. The computer vision framework which was used as a playground for experiments was developed in Apical Ltd.

The novelty of this work is in demonstration of how measure concentration effects can be used in practical computer vision applications, including the developed theory and algorithms of knowledge transfer between two artificial intelligence systems.

The applicability of the developed approaches has been tested in a series of practical computer vision tasks. However, applicability is not restricted to the computer vision domain only. Suggested algorithms of knowledge transfer can be implemented for and deployed in any Big Data system that make any sort of decisions

based on a large number of variables. The only requirement is that one must have access to/simulate internal variables of such AI systems, and these variables, if combined together, are elements of some topological vector field.

Practical and applied testing of developed approaches has been done during my work in Apical Ltd.(now ARM Ltd.) in years 2013-2017.

Structure of the thesis. In Chapter 2 we outline major sources of errors and uncertainty in Big Data Analysis Systems. In the same chapter we describe issues one has to face when dealing with high dimensional data. This will be described in Section 2.4. Dimensionality reduction techniques are described in Chapter 3.

Chapter 4 consists of theoretical body of the work. The main result are answers to the questions stated above: how can one make use of high dimensionality? Why can it be beneficial to keep the dimensionality of a problem high rather than trying to reduce it? We also answer one practical question, specifically: how can measure concentration be used to improve existing computer vision systems?

In Chapter 5 several case studies are described. The main goal of these case studies is to show how theoretical tools developed in Chapter 4 can be used in practice in various computer vision applications. For example, in Sections 5.1 and 5.2 we see which techniques are applicable for solving low dimensional data problems. Section 5.3 describes practical methods of benefiting from measure concentration effects in the computer vision domain.

We now describe the authors contribution to the results presented. General vision, ambition and planning of the work is to be attributed to many of senior colleagues whom I had pleasure to work with. Theorems, statements, and proofs is a joint endeavor with my supervisor. It is very difficult to disentangle individual efforts from the final results. However, all experiments and numerical simulations, as well as result analysis have been done solely by myself.

Chapter 2

Source of errors in Big Data Analysis Systems

2.1 Machine learning problem statement

The general model for learning from examples can be described with three components [85]:

1. A generator G of random vectors $x \in R^n$ drawn randomly from a fixed but unknown probability distribution function $F(x)$

2. A supervisor S that returns an output value to every input vector x, according to conditional distribution function $F(y|x)$

3. A learning machine capable of implementing a set of functions $f(x, \alpha), \alpha \in \Lambda$, where Λ is a set of parameters.

The selection of the desired function is based on *training set* with l independent and identically distributed examples drawn according to $F(x, y) = F(x)F(y|x)$:

$$(x_1, y_1), \ldots, (x_l, y_l). \tag{2.1}$$

The problem of learning is to chose particular function $f(x, \alpha), \alpha \in \Lambda$ that approximates supervisor's response in the best way. In order to chose this function one has measure the *loss* $L(y, f(x, \alpha))$ between the response y of the supervisor and the response $f(x, \alpha)$ provided by the learning machine. Consider the expected value of

the loss, given by a *risk functional*:

$$R(\alpha) = \int L(y, f(x, \alpha))dF(x, y). \tag{2.2}$$

The goal is to find a function $f(x, \alpha_0)$ that minimizes the risk functional $R(\alpha)$ in the situation where joint probability $F(x, y)$ is unknown and the only available information is the training set.

This definition of a learning problem is very broad and it includes many specific problems. However, there are three main types of problems: the problem of pattern recognition, regression estimation and density estimation.

In pattern recognition the supervisors output can take only two values $y = \{0, 1\}$. Let $f(x, \alpha), \alpha \in \Lambda$, be a set of indicator functions. Consider the following loss function:

$$L(y, f(x, \alpha)) = \begin{cases} 0 & \text{if} \quad y = f(x, \alpha), \\ 1 & \text{if} \quad y \neq f(x, \alpha). \end{cases} \tag{2.3}$$

Equation 2.2 determines the probability of different answers given by the supervisor and the indicator function $f(x, \alpha)$. The case of different answers is called *classification error*.

The problem is to find function that minimizes the probability of classification error when the probability measure $F(x, y)$ is unknown, but the data (2.1) are given.

In *regression estimation* the supervisor's response is a real value and the set $f(x, \alpha), \alpha \in \Lambda$, is a set of real functions. This set contains *regression function*

$$f(x, \alpha_0) = \int y dF(y|x).$$

The regression function is the one that minimizes (2.2) with the following loss function:

$$L(y, f(x, \alpha)) = (y - f(x, \alpha))^2. \tag{2.4}$$

Thus the problem of finding regression estimation is the problem of minimizing the functional (2.2) with the loss function (2.4) when the distribution $F(x, y)$ is unknown but the data (2.1) are given.

Lastly, consider the problem of density estimation from the set of densities

$p(x, \alpha), \alpha \in \Lambda$. For this problem we consider the following loss function:

$$L(p(x, \alpha)) = -\log p(x, \alpha). \tag{2.5}$$

The desired density minimizes risk functional (2.2) with the loss function (2.5).

The general setting of a learning problem is described as follows. Let the probability measure $F(z)$ be defined on the set Z. Consider the set of functions $Q(z, \alpha), \alpha \in \Lambda$. The goal is to minimize the risk functional

$$R(\alpha) = \int Q(z, \alpha) dF(z), \quad \alpha \in \Lambda, \tag{2.6}$$

where the probability measure $F(z)$ is unknown, but an i.i.d. sample

$$z_1, \ldots, z_l \tag{2.7}$$

is given.

In order to minimize functional (2.6) with an unknown distribution function $F(z)$, the following inductive principle can be applied.

1. The risk functional $R(\alpha)$ is replaced by *empirical risk functional*:

$$R_{emp} = \frac{1}{l} \sum_{i=1}^{l} Q(z_i, \alpha), \tag{2.8}$$

   constructed on the basis of the training set (2.7).

2. One approximates the function $Q(z, \alpha_0)$ that minimizes risk (2.6) by the function $Q(z, a_l)$ minimizing the empirical risk (2.8).

This principle if called *empirical risk minimization inductive principle* (ERM principle).

The ERM principle is quite general. The classical method of solution of specific learning problems are realizations of the ERM principle for a specific loss function. For example, for the regression problem one has the following functional to be minimized:

$$R_{emp}(\alpha) = \frac{1}{l} \sum_{i=1}^{l} (y_i - f(x_i, \alpha))^2,$$

which forms the least squares method. This gives a brief introduction into the machine learning problem statement, including the pattern recognition problem.

2.2 Issues of Machine Learning

In practical applications there are several problems associated with minimization of risk functional (2.6). These problems are common for many domains where machine learning techniques are used. These problems include:

1. A lot of real world problems are ill-posed (see Appendix B);
2. Minimization of risk functional (2.6) for high dimensional data often involves heavy computations and takes a lot of time;
3. High dimensional data often causes iterative algorithms to stick in local minima;
4. Training of machine learning systems on high dimensional data often leads to overfitting.

There is a lot of research dedicated to solving some of these problems. While some of them can partly be solved by scaling up the amount of resources available for training, others (e.g. local minima sticking, overfitting) are more fundamental. Another problem is that the outcome of the function (2.6) minimization is random. There are several factors contributing to this randomness:

1. Functional (2.6) minimization is an iterative process. At each iteration parameters get updated. During the initialization, however, the parameter set is often drawn from some distribution [64];
2. Learning machines are often trained with mini-batches of data [66]. These mini batches are chosen randomly from the training set and therefore cause random fluctuations of the risk functional (2.6). Also, division of the dataset into training and test sets is random.

These two factors are fundamental constraints of machine learning systems. Due to the amount of parameters being optimized, machine learning systems will always

make mistakes. For avoiding mistakes completely, one has either to have an amount of data representing every possible situation or to spend significant amounts of time sampling the function space $f(x_i, \alpha)$.

2.3 Specific issues of Machine Learning in Computer Vision domain

A particular domain, where machine learning techniques are widely used is computer vision. In the computer vision domain the general definition of a machine learning problems given by (2.6) is also suitable, except that vectors x_i in (2.1) are replaced with images I_i:

$$(I_1, y_1, \ldots, I_i, y_i). \tag{2.9}$$

One thus aims to minimize the following empirical functional using the least squares method:

$$R_{emp}(\alpha) = \frac{1}{l} \sum_{i=1}^{l} (y_i - f(I_i, \alpha))^2.$$

To date state of the art methods for minimizing this risk functional for a wide range of computer vision tasks are methods based on convolutional neural networks (CNN). In recent years this approach has proved itself to be a very efficient tool in visual image recognition tasks [51, 62, 79, 38]. Despite this fact, there are may hidden difficulties in application of this approach:

1. The problem of CNN training is ill-posed (see Appendix B);

2. Extremely long training times: sometimes it takes weeks for an optimization process to converge. That means that one is not able to sample the function space $f(x_i, \alpha)$ quickly enough;

3. High requirements for training set (2.9): to train a large CNN one has to have millions of images. For example, as of 2017 the ImageNet dataset [62] contains at least 10 million images. It is extremely hard to ensure that the training set is balanced correctly;

4. High probability to get stuck in a poor local minimum.

5. Due to the amount of parameters CNNs are prone to overfitting;

6. Lower efficiency of training of deep architectures [38].

All in all, despite the amount of model parameters, state of the art Learning Machines are still prone to errors.

There is a lot of research dedicated to solving some of these problems. Several techniques have been developed and applied in order to improve stability of the training process and reduce overfitting. However, some of these techniques introduce even more randomness into the training process. For example, the Dropout technique [15] prohibits some gradients to propagate back to the network. Choice of gradients that are nullified at each iteration is random.

The dimensionality of the data in computer vision tasks is also a problem. In practical applications RGB images are normally rescaled to a standard size $N \times N$ [1]. Given that each pixel has three components, one has to deal $N \times N \times 3$-dimensional data. Not only can visual data have such a big dimensionality: a lot of real world problems also do.

While there is not much that can be done against the complexity of the training process, it is possible to reduce complications introduced by the high dimensionality of the data. In what follows we will briefly discuss difficulties associated with high dimensional data and classical methods of dimensionality reduction.

2.4 High dimensional data issues

As it was pointed out before, dealing with high-dimension data has its own complications. In this chapter we will be discussing those problems.

1. Any dimensionality higher than 4 is very hard to imagine and therefore do quick in-mind evaluations of ideas .

2. Visualization difficulties [75][44]. Any visualizations requires data projection to a low dimensional space, therefore, the higher the dimensionality of the space we project from, the more information is lost.

[1] Typical values of N would be something like 80 or 256

3. Computational difficulties. Large numerical and categorical descriptions of entities in large amount of machine learning tasks are computationally hard to deal with.

4. The concept of the distance is changing as the dimensionality grows. This will be discussed in Chapter 4.

5. Attribute correlation issues. When dealing with high dimensional data we can often meet highly correlated attributes. This issue leads to unstable or non robust models [83].

6. If the process of data acquisition is expensive, then having insufficient data easily leads to overfitting [60].

7. Trade-off between accuracy and computational complexity. In high dimensional data mining applications we often have to reject complicated and expensive solutions in order to get something reasonable in terms of performance [73].

Even though the list shown here is relatively short and not domain specific, it still reflects the most common and general problem we face in high dimensional data analysis.

2.5 Curse of dimensionality

Curse of dimensionality refers to cases that arise when trying to analyze huge amounts of data. Most often this term is mentioned when we speak about hundreds and thousands of dimensions. The expression was first introduced by Richard E. Bellman [11, 12].

Curse dimensionality phenomena can be met in different areas of knowledge and can mean many things.

The basic example is NP-hardness. Solutions for some problems can be easily found in two dimensions. However, same problem becomes NP-hard for any dimensionality higher than 2.

In Combinatorics we can meet the problem when dealing with discrete variables. If one wants to consider all possible combinations in order to find the best one,

then even for n binary variables we have to check 2^d combinations. The problem quickly gets unsolvable if the number of possible states for each variable grows and the number of variables increases.

Another "curse" of dimensionality refers to data sampling problem. As dimensionality grows, data points become sparser and sparser. In this case one needs more and more data to avoid overfitting. This is also a problem because many machine learning algorithms rely on nearest neighbor idea and proximity of data points.

In optimization we face the problem when minimized variable depends on hundreds of thousands of variables. Generally speaking, the problem is that the landscape of the target function is very complicated and in general case cannot be analyzed analytically. Therefore any solutions found can be too sensitive to the data configuration. As optimization often starts from random initial conditions, it impossible to analyze how far found solution is from an optimal one.

In Machine Learning the aim is to minimize risk functional (2.6). However, often object's nature and relationships between different parameters of the object are so complicated that it takes enormous amount of data to build a good and reliable model. Problem tends to get more complicated when the number of samples at hand is insufficient. As it was mentioned by Gordon F. Hughes in [43], the predictive power of the models may reduce as the number of dimensions grows. This is known as Hughes phenomenon.

One of the most obvious ways to avoid listed issues is to reduce dimensionality of the problem. This has proven to be beneficial in many cases [87][59][27]. In Chapter 3 we review some of these techniques and use cases they are useful in.

Chapter 3

Dimensionality reduction

Dimensionality reduction techniques allow to decrease the number of variables when data are high dimensional. Operating in a reduced feature space has several advantages, which include:

1. Reduction of the number of calculations required;
2. Models built in a lower dimensional space are more robust;
3. Some of these techniques allow to leave only most valuable dimensions along which data have the biggest variance;
4. In low dimensional spaces (e.g. $2D$, $3D$) visualization becomes possible.

The very basic dimensionality reduction techniques include:

1. Missing values ratio. The idea of the method is to remove columns from the dataset which contain too many missing values. The step can be explained by that assumption that fairly often these columns do not contain very much useful information. However, this method is very limited, because many datasets do not contain any missing values (e.g. computer vision and speech recognition tasks)
2. Low variance filter. This method is very similar to the previous one. Here we remove columns with low variance assuming that some columns with very small data variability also do not contain very much useful information. The main drawback of this method, as we shall see later, is that this assumption often does not hold.

3. High correlation filter. This method relies on an assumption that two different features with high correlation are likely to contain similar information and one of them can be removed.

4. Binary decision trees [68]. Being classifiers in the first place, the induce procedure for binary decision trees has an interesting byproduct: having a decision tree at hand we can see which variables appear more informative than the others and therefore assume that latter are less important and therefore can be removed.

5. Backward Feature Elimination [36]. In this algorithm we first calculate the error of the model with all features in place. Then we remove features one by one, retraining the model each time and observing the new error. Features causing the smallest error increase are removed.

6. Forward feature construction [36]. The idea is to follow the process opposite to Backward Feature Elimination.

   It is worth noticing that the latter two algorithms can be applied effectively in only two cases. Either building of a classifier is cheap in terms of time and computation resources required or the dataset is small. Otherwise, these two methods are not feasible due to time limitations.

The list above illustrates only the simplest dimensionality reduction algorithms possible. Jaap van den Herik, Laurens van der Maaten and co-authors published a comprehensive classification of dimensionality reduction algorithms including more sophisticated ones [84].

3.1 Full spectral Techniques

Here we discuss a set of techniques that are based on dataset correlation matrix analysis.

Principal components analysis

Principal component analysis (PCA) [42, 63] is a statistical procedure that takes a dataset as input and makes an orthogonal transformation of this dataset. Variables

of the dataset are transformed into a set of linearly uncorrelated variables. These variables are called *principal components*.

PCA can also be thought as fitting an N-dimensional ellipsoid to data. When the ellipsoid is fit, its axis represent the principal components of the data. Therefore, removing relatively small axes (principal components) from the data we loose relatively small amounts of information. The procedure of principal components calculation is sensitive to the initial data scaling. By writing down the process of finding principal components we shall see why this is the case.

Let us define the data matrix X. Rows and columns of this matrix stand for observations and features respectively. We assume that each column is transformed so that in the new coordinates it has zero empirical mean. As we say about orthogonal transformation, we use some matrix W to project the original dataset into a different space. Let k be the number of rows in W, then each new component of the transformed dataset is obtained by multiplying each column of transposed X by a row $\mathbf{w}_k = (w_1, \ldots, w_p)_{(k)}$. As a result, we get new set of features $\mathbf{T}$:

$$\mathbf{T} = \mathbf{XW}. \tag{3.1}$$

It is important to note that while applying this transformation we aim to keep the maximum possible variance of data along the given dimensions. The first column $\mathbf{w}_1$ satisfies the following condition:

$$\mathbf{w}_1 = \arg\max_{||\mathbf{w}||=1} \frac{\|\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}\|}{\mathbf{w}^T\mathbf{w}}. \tag{3.2}$$

Since $\mathbf{X}^T\mathbf{X}$ is a matrix, not a vector, the quantity above can be viewed as a *Rayleigh quotient*. It is a known result that the maximum possible value of this quotient is the maximum eigenvalue of the $\mathbf{X}^T\mathbf{X}$ which is achieved when $\mathbf{x}$ is the maximum eigenvector.

Further components can be found by subtracting the first $k-1$ components from $\mathbf{X}$:

$$\hat{\mathbf{X}}_k = \mathbf{X} - \sum_{s=1}^{k-1} \mathbf{X}\mathbf{x}_{(s)}\mathbf{x}_{(s)}^T. \tag{3.3}$$

After this we repeat the same process of maximizing the Rayleigh quotient for

the matrix $\hat{\mathbf{X}}_k$. It appears that the final transform can be made up of all eigenvectors of the matrix $\mathbf{W} = \mathbf{X}^T\mathbf{X}$. Therefore, the final PCA transformation is described by

$$\mathbf{T} = \mathbf{XW}. \tag{3.4}$$

PCA is a very useful tool for visualization. Once the data is projected to a $2D$ or $3D$ orthogonal space it can be visualized and analyzed. However, there are a few caveats with PCA. The following assumptions are usually made when one is using PCA:

1. Being linear in nature, PCA assumes that there is an effective linear dimensionality reduction transformation exists for the dataset;

2. The directions that have the maximum variance are the most informative ones.

In addition to providing a useful dimensionality reduction tool, PCA has several drawbacks that may require special consideration in application. First and foremost, the size of the covariance matrix is obviously proportional to the dimensionality of the data. This makes this approach infeasible in very high dimensional spaces. Secondly, as was mentioned before, PCA assumes that dimensions with the highest variance are the most useful ones. However, this is not always the case. For example, we mention the so called "data cake", which is depicted in Figure 3.1.

The last but not the least, rules for dimensions pruning are heuristic. For example, in the Kaiser-Guttman criterion we calculate the mean of all eigenvalues M_e and leave only directions with corresponding eigenvalues greater than M_e.

Kernel PCA

Kernel PCA is a reformulation of the classical PCA approach. The difference is that while PCA involve construction of covariance matrix, Kernel PCA constructs a kernel matrix [7]. After matrix is constructed, this approach is similar to PCA - calculate the eigenvectors of the kernel matrix and build a lower dimensional embedding of data if one is required.

Speaking more formally, Kernel PCA computes a kernel matrix K of the datapoints $\mathbf{x}_i$:

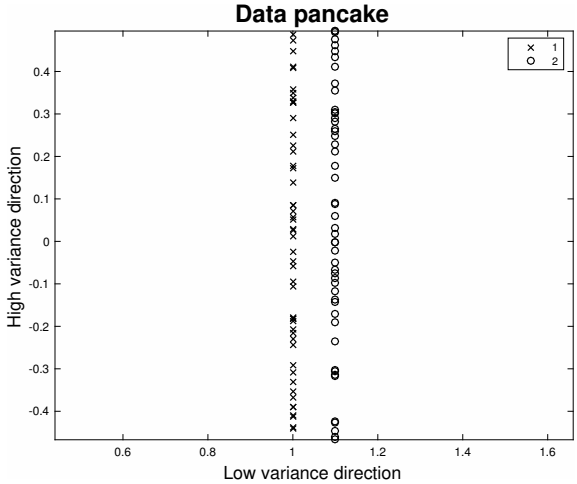


Figure 3.1: Example of where the direction with the highest variance is not the most useful one.

$$k_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j), \tag{3.5}$$

where κ is a kernel function [78]. Kernel PCA has pretty much the same weaknesses as classical PCA does: the size of kernel matrix is proportional to the number of points and the data manifold structure is not taken into account during dimensionality reduction.

Isomap

The principal component approach takes into account pairwise distances, but no information about dataset distribution is used. Therefore, if data are located on some curved manifold, PCA may consider two points to be close according to Euclidean distance calculations. However, if we take the form of the manifold data are distributed across, it might appear that points are located further from each other. The example of this situation is so called Swiss Roll [47] depicted in Figure 3.2. Isomap [47] techniques addresses this issue. Isomap is working not with Euclidean distance, but with *geodesic distance*. Geodesic distance is the distance measured between two points over the manifold. A further idea is to build a connected graph in which a point has connection only with the nearest N points. After the graph is constructed, the geodesic distance can be calculated as a shortest path between two points in this graph. After distances are obtained, the classical PCA approach can be used for further dimensionality reduction. Therefore, the main goal of Isomap is

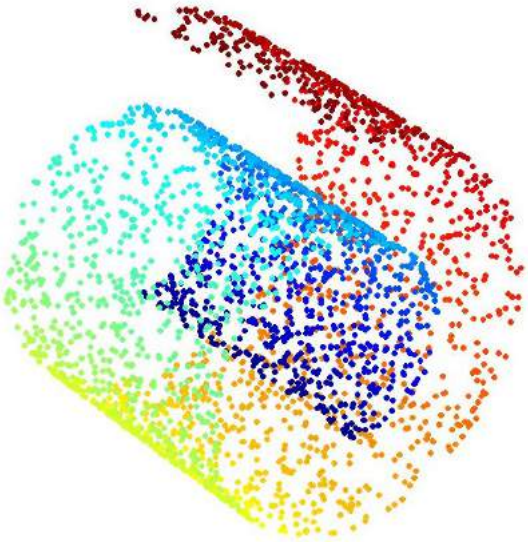


Figure 3.2: Swiss Roll dataset.

to embed some knowledge about the data manifold into a distance matrix.

However, if we further think about the graph construction process, we will see that in some cases Isomap will build a graph that does not adequately show the manifold structure. This is referred to as topological instability [8]. There are several approaches that aim to deal with this problem. For example, one can remove points that violate local graph linearity [1]. By dataset nature or after points were removed holes may appear in data. These kind of problems were also addressed in [54]. The third potential issue for Isomap is dealing with non-convex datasets, where is might be difficult to capture to general structure of the data manifold [82].

Diffusion Maps

Diffusion maps [52, 6] repeat Isomap in terms of building a data graph, but in this method a different approach to the distance calculation is used. In this approach we define a Markov random walk on the graph of the data. For each point we then count the number of times its neighborhood was visited and accept this counter as a measure of proximity between data point and its neighborhoods. This is called the *diffusion distance*. The advantage of this approach is that during similarity matrix construction more information about the graph structure is used. This type of distance if more robust than the geodesic distance used in Isomap.

Speaking more formally, we first construct a data graph with edges w_{ij}, where i

and j are indices of the data points. The matrix $\mathbf{W}$ contains the weights of edges in this graph which are defined as follows:

$$w_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}. \tag{3.6}$$

Since the weights of those edges are later used in the Markov random walk process, weights are supposed to represent the probability of visiting each edge and, therefore, have to add up to 1. Now we construct a new matrix $\mathbf{P}$ which is formed as follows:

$$p_{ij} = \frac{w_{ij}}{\sum_k w_{ik}}. \tag{3.7}$$

Then after t steps the elements of the diffusion matrix are defined as follows:

$$D^{(t)}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_k \frac{(p_{ik}^{(t)} - p_{jk}^{(t)})^2}{\psi(\mathbf{x}_k)}}, \tag{3.8}$$

where $\psi(\mathbf{x}_k) = \frac{m_i}{\sum_j m_j}, m_i = \sum_j p_{ij}$. It has been shown in [52] that a low dimensional representation $\mathbf{Y}$ that approximates $D^{(t)}(\mathbf{x}_i, \mathbf{x}_j)$ as well as possible is formed by d non trivial principal eigenvectors of the following problem:

$$\mathbf{P}\mathbf{v} = \lambda\mathbf{v}. \tag{3.9}$$

Since the data graph is fully connected and therefore each point is connected with itself, the largest eigenvalue is trivial and equal to 1. Therefore the next d eigenvectors give the following low-dimensional representation:

$$\mathbf{Y} = \{\lambda_2\mathbf{v}_2, \lambda_3\mathbf{v}_3, \ldots, \lambda_{d+1}\mathbf{v}_{d+1}\}. \tag{3.10}$$

3.2 Sparse Spectral Techniques

All techniques described before aim to solve a simple eigenvalue problem. The following class of techniques solves a generalized (sparse) eigenvalue problem which has the following form:

$$\mathbf{P}\mathbf{v} = \lambda\mathbf{B}\mathbf{v}. \tag{3.11}$$

Local Linear Embedding (LLE)

Local linear embedding [74] is somewhat similar to Isomap, but in contrast to this method it aims to preserve only the local structure of the data. This is achieved in the following way: take a point $\mathbf{x}_i$ and try to reconstruct this point as a linear combination of its k neighbors. These operations assume that the data manifold is locally linear. Since the reconstruction surface is linear, weights obtained during plane fitting procedure are preserved under affine transformations. Therefore, any of these coefficients are preserved under any linear operator A. In other words, such a transformation preserves local geometry of the manifold, and weights w_i. Therefore, finding a lower dimensional data representation Y leads to the minimization of the following cost function:

$$\phi(\mathbf{Y}) = \sum_i \|\mathbf{y}^2 - \sum_{j=1}^{k} w_{ij}\mathbf{y}_{i_j}\|^2 \text{ subject to } \|\mathbf{y}^{(k)}\|^2 = 1 \text{ for } \forall k. \tag{3.12}$$

It was shown in [74] that vectors $\mathbf{y}_i$ minimizing the cost function can be found by computing non-trivial eigenvectors of the following product:

$$(\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W}). \tag{3.13}$$

where $\mathbf{W}$ is $n \times n$ sparse matrix, whose entries are set to 0 if the i-th and j-th points are not connected in the data graph and the weight of the edge connecting them, otherwise.

Laplacian Eigenmaps

In Laplacian Eigenmaps [10] we also minimize a cost function, but this function is based on distances between nearest neighbors. Therefore, the function that is minimized is

$$\phi(\mathbf{Y}) = \sum_{ij} \|\mathbf{y}_i - \mathbf{y}_j\| w_{ij}. \tag{3.14}$$

Hessian Local Linear Embedding

This method is essentially a variation of the Local Linear Embedding method described earlier in this section. The idea here is to minimize the curvature of the manifold data points are embedded in.Hessian LLE [22] starts from taking k nearest data points for each data point $\mathbf{x}_i$ using Euclidean distance. Again, the manifold is assumed to be locally linear.

Local Tangent Space Analysis

Local Tangent Space Analysis [88] is a technique that describes local properties of the high-dimensional data using the local tangent space of each data point. The main idea is that if local linearity of the manifold is assumed, there exists a linear mapping from a high-dimensional datapoint to its local tangent space and hence there exists a linear mapping from the corresponding low-dimensional datapoint to its low-dimensional tangent space [88].

3.3 Non-convex Techniques

Sammon Mapping

The main weakness of the PCA approach is that it tries to retain large pairwise distances between distant points. While retaining them, PCA does not retain small pairwise distances. Quite often these small pairwise distances are more important for understanding of the data geometry. One of the approaches that address this issue is Sammon Mapping technique [76].

Sammon mapping uses weighted distance calculation. Contribution of each pair is weighted by inverse value of distance d_{ij} between two points. This type of scaling allows equal contribution to the cost function for both large and small pairwise distances. More formally the cost function is defined as:

$$\phi(\mathbf{Y}) = \frac{1}{\sum_{ij} d_{ij}} \sum_{i \neq j} \frac{(d_{ij} - \|\mathbf{y}_i - \mathbf{y}_j\|)^2}{d_{ij}}, \tag{3.15}$$

where d_{ij} is a pairwise distance between two points. The minimization of this cost function is usually done by a pseudo-Newton method [18].

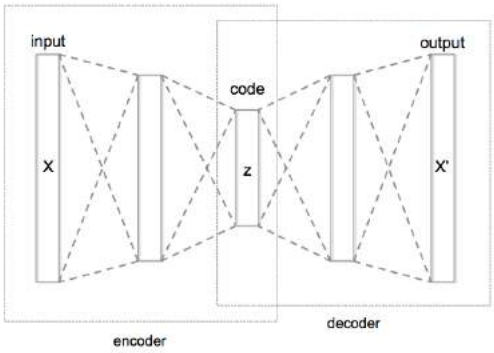


Figure 3.3: Autoencoder basic structure. Structure has equal number of inputs and outputs, middle layer represents input entity encoded.

Multilayer Autoencoders

Multilayer Autoencoders are essentially multilayer neural networks which have special structure. To start with, the number of inputs and outputs are equal in this structures and equal D. The number of layers is odd and middle layer has d dimensions and $d < D$. Then the idea is to minimize the mean-squared error between input and output. When the network is trained the middle layer with d dimensions represents an effective low dimensional representation of input data. The generic structure of autoencoder is shown in Figure 3.3.

The left part of the autoencoder is called the encoder and right part is called the decoder. Mathematically, the encoder maps the input to a lower dimensional space:

$$\phi : X \to F, \tag{3.16}$$

where F is a lower dimensional space, X is the input.

The decoder part does the opposite:

$$\psi : F \to X. \tag{3.17}$$

Therefore, mathematically, the task of building an autoencoder can be seen as

$$\arg\min_{\phi,\psi} \|X - (\psi \circ \phi)X\|^2. \tag{3.18}$$

Autoencoders have found many successful applications in data mining. For ex-

ample, they were used for building various image denoising algorithms. The idea is that a distorted signal is given as input and a clean signal is given as output. Then the autoencoder is trained for noise reduction. [86].

Autoencoders are not used only as a dimensionality reduction tool. This technique finds a lot of applications if different areas of computer vision. For example, autoencoders were used to generate natural language image descriptions [50, 72].

3.4 Random projections

Random projection is another dimensionality reduction technique used in machine learning. This methods maps a high dimensional data point to a lower dimensional one. The reduced dimensionality of the space allows us to perform faster calculations [2] on the data and get smaller and more robust models.

This method has been successfully applied in many applications including image and text processing [29, 70], compressive sensing [20], manifold learning, graph embedding [34] and dimensionality reduction.

The reason why the random projection method is a useful dimensionality reduction tool can be found in Johnson–Lindenstrauss lemma [49]: a small set of points in a high-dimensional space can be embedded into a space of lower dimensionality so that the pairwise distance between any two points is nearly preserved.

Suppose, we have an $M \times N$ matrix $\mathbf{F}$ whose rows contain M observations and columns contain N predictors. Let us also introduce a linear operator $\mathbf{P}$ which maps points described in $\mathbf{F}$ in a space of lower dimensionality. Lets say, its size will be $M_p \times N$. Therefore, the described operator maps our feature from feature space S to a space S_L:

$$\mathbf{P} : S \to S_L. \tag{3.19}$$

Now we can get projected features by using the following equation:

$$\mathbf{F}_p = (\mathbf{P}\mathbf{F}^T)^T. \tag{3.20}$$

It has been shown in [49] that such transformation introduces a limited distortion to the metric, which can be described by the following inequality (named the

Johnson–Lindenstrauss lemma):

$$(1-\varepsilon)||u-v||^2 \leq ||P(u)-P(v)||^2 \leq (1+\varepsilon)||u-v||^2. \tag{3.21}$$

In this inequality ε is a number from the interval $(0,1)$.

There are several things to note. First, the projection matrix P is not orthogonal, which can introduce significant distortions to the dataset. In some cases orthogonalization can be very expensive. However, as it was shown in [39], in high dimensional spaces two randomly chosen vectors are almost orthogonal, which makes P^TP close to the identity matrix.

Another important problem is choosing the matrix P. There are several approaches to handle this. One can generate matrix P using the Gaussian distribution [3]. The first row is sampled from the Gaussian distribution. The second row is chosen to be orthogonal to the first one and so on. A matrix generated in this way will have the following properties:

1. Spherical symmetry: $\mathbf{F}$ and $\mathbf{F}_p$ have the same distribution;
2. Orthogonality: The rows of $\mathbf{P}$ are pairwise orthogonal;
3. Normality: The row of $\mathbf{P}$ have unit length.

Another way to chose the matrix P was shown in [2]. One can use the following procedure:

$$R_{i,j} = \sqrt{3}\begin{cases} +1 & \text{with probability } \frac{1}{6}, \\ 0 & \text{with probability } \frac{2}{3}, \\ -1 & \text{with probability } \frac{1}{6}. \end{cases}$$

Projection matrices generated with this algorithm allow one to use integer calculations.

3.5 The blessing of dimensionality

So far we have emphasized difficulties which are caused by high dimensionality of a certain problem. However, there are theoretical benefits associated with high

dimensionality. Some important advantages of having data in high dimensional space will be discussed later. Now we will turn our attention to the well-known phenomenon called concentration of measure. We will briefly discuss known results which will partly be used in Chapter 4.

Concentration of measure term was first introduced by V. Milman. This is an important principle that finds many applications in different fields of combinatorics, functional and discrete analysis, statistics, probability theory, geometry, and statistical physics. Informally it says that "a random variable that smoothly depends on the influence of many independent random variables (but not too much on any of them) is essentially constant" [81]. A slightly different way to look at this statement is as follows: a function that depends on many variables does not deviate from some typical value, e.g. its mean or some other constant.

Let us give a few non-formal examples. Say, if we toss a coin once then the result is unpredictable: both heads and tails have equal probability. However, if we toss a coin large number of times(e.g. 10000), the result becomes highly predictable. Specifically, we can say that that number of tails will be somewhat around 5000. This is the simplest example of measure concentration phenomenon. Another example of this phenomenon is as follows. Our world consists of microscopic particles governed by laws of quantum and statistical physics [21]. Nevertheless, macroscopic properties determined by the ensembles of these particles are very deterministic. The reason for this is that possibilities observed on microscopic scale concentrate in a very narrow range. Similar examples can be found in computer science. Randomized algorithms that depend on many parameters will have (almost) constant execution time and memory consumption. On the other hand, even though parameters in a random algorithm are drawn at random, it is very likely that many runs of this algorithm will show almost deterministic result.

There are classical results that concern the same phenomenon: the central limit theorem, the law of large numbers and the theory of large deviations. The law of large numbers roughly says that if th same experiment is conducted many times, the average of the outcome will tend to concentrate around experiment's expected value. The central limit theorem says that "properly" normalized sums of random variables tend to normal distribution. Finally, the theory of large deviations studies

remote tails of random sequences. It says that the probability of an extreme "tail" events decreases with the exponential rate.

Despite the theoretical value of these results, the practical application of these theories is difficult since:

1. They are assymptotic limit laws applied in the infinite limit case. Most of practical applications concern the finite case;

2. These results say almost nothing about the rate of convergence.

Speaking more formally, we want to be able to answer the following question: given the random variable X with its mean value $E[X]$, what is the probability that X deviates far from $E[X]$?

Markov's inequality can serve as a basis for answering this question. Markov's inequality states the following fact:

$$P[X \geq \varepsilon] \leq \frac{E[X]}{\varepsilon}. \tag{3.22}$$

To see that this is true, imagine we have a non-negative function $f(x)$ and let us introduce the following function:

$$s(x) = \begin{cases} \varepsilon & \text{if} \quad f(x) \geq \varepsilon, \\ 0 & \text{if} \quad f(x) < \varepsilon. \end{cases}$$

Let μ be a measure and since $0 \leq s(x) \leq f(x)$, we have the following:

$$\int_X f(x)d\mu \geq \int_X s(x)d\mu = \varepsilon\mu(\{x \mid f(x) \leq \varepsilon\}).$$

Therefore,

$$\mu(\{x \mid f(x) \leq \varepsilon\}) \leq \frac{1}{\varepsilon}\int_X f(x)d\mu = \frac{E[X]}{\varepsilon}.$$

The latter proves Markov's inequality. Furthermore, let $X_c = |X - E[X]|$ and $\phi(x)$ denotes non-negative, non-decreasing function, then the following is also true:

$$P[X_c \geq \varepsilon] \leq P[\phi(X_c) \geq \phi(\varepsilon)] \leq \frac{E[\phi(X_c)]}{\phi(\varepsilon)}. \tag{3.23}$$

If we take $\phi(\varepsilon) = \varepsilon^2$, we obtain Chebyshev's inequality:

$$P[|X - E[X]| \geq \varepsilon] \leq \frac{Var[X]}{\varepsilon^2}.$$

Chernoff inequality allows one to obtain best possible bound for a tail probability, while using Markov's inequality. Let now $\phi(\varepsilon) = e^{\lambda\varepsilon}$. Then Markov's inequality becomes:

$$P[X \geq \varepsilon] \leq e^{-\lambda\varepsilon} E[e^{\lambda X}]. \tag{3.24}$$

For $\lambda > 0$ one can choose value of λ to minimize the upper bound 3.24. The Chernoff's inequality leads us to Hoeffding's inequality for sum of independent random variables. Let $X_1, \ldots, X_n$ be independent random variables such that X_i takes its values in $[a_i, b_i]$. Let $S = \sum_{i=1}^{N}(X_i - E[X_i])$, then:

$$P[S > \varepsilon] \leq e^{-\frac{2\varepsilon^2}{\sum_{i=1}^{N}(b_i - a_i)^2}}. \tag{3.25}$$

Another famous inequality is Minkowski's inequality. The best known form of this inequality is triangle inequality for L_p norms of vectors of random variables. Let X_1 and X_2 be two real-valued random variables, then for $q > 0$ we have the following:

$$E[|X_1 + X_2|^q]^{1/q} \leq E[|X_1|^q]^{1/q} + E[|X_2|^q]^{1/q}, \tag{3.26}$$

which has the more general form:

$$E_X[|E_Y[Z]|^q]^{1/q} \leq E_Y[(E_X[|Z|^q])^{1/q}], \tag{3.27}$$

where $Z = f(X, Y)$, f is a real-valued measurable function, $E_X[Z] = E[Z|Y]$ and $E_Y[Z] = E[Z|X]$.

One more inequality we will be referring to is the Brunn-Minkowski inequality. Here we consider sets $A, B \subset \mathbb{R}^n$. We also define *Minkowski sum* of A and B as the set of all vectors in $\mathbb{R}^n$ formed by sum of individual elements of A and B:

$$A + B = \{a + b \mid a \in A, b \in B\}. \tag{3.28}$$

Similarly, for $c \in \mathbb{R}$, let $cA = \{ca \mid a \in A\}$. We also denote $\mathcal{V}(A)$ the Lebesgue measure of $A \subset \mathbb{R}^n$. The Brunn-Minkowski inequality states the following:

$$\mathcal{V}((1-\lambda)A + \lambda B)^{1/n} \geq (1-\lambda)\mathcal{V}(A)^{1/n} + \lambda \mathcal{V}(B)^{1/n}. \tag{3.29}$$

The last statement we consider is so called Isoperemtric theorem. Consider the set $A \subset \mathbb{R}^n$. Now we introduce set A_t which will be called t-enlargement of A:

$$A_t = \{x \in \mathbb{R}^n \mid d(x, A) < t\}, \tag{3.30}$$

where $d(x, A)$ denotes the distance from x to set A. We define the surface area of A as:

$$\mathcal{V}(\partial A) = \lim_{t \to 0} \frac{\mathcal{V}(A_t) - \mathcal{V}(A)}{t}. \tag{3.31}$$

Also, if $B = \{x \in \mathbb{R}^n \mid \|x\| \leq 1\}$ (the unit open ball), then taking 3.28 into account, we have:

$$A_t = A + tB. \tag{3.32}$$

Now we can turn our attention to Isoperemetric theorem.

Theorem 1 (Isoperemetric theorem, [80]) *Let $A \subset \mathbb{R}^n$ be such that $\mathcal{V}(A) = \mathcal{V}(B)$. Then, for any $t > 0$, $\mathcal{V}(A_t) \geq \mathcal{V}(B_t)$.*

Proof. Using 3.29, we have the following:

$$\begin{aligned}
&\mathcal{V}(A_t)^{1/t} = \mathcal{V}(A + tB)^{1/t} \\
&\geq \mathcal{V}(A)^{1/t} + t\mathcal{V}(B)^{1/t} \\
&= \mathcal{V}(B)^{1/t}(1+t) = \mathcal{V}(B_t)^{1/t}.
\end{aligned}$$

The statements above are examples how measure concentration effect can be quantified in several ways. There are several other techniques developed that help to prove the concentration of measure phenomenon, which include:

1. The Martingale approach [58, 5, 41];
2. The entropy method and logarithmic Sobolev inequalities [53, 57];
3. Transportation-cost inequalities [53, 33];

4. Talagrand's inequalities for product measures [58, 81].

In the next chapter we will show how specific concentration effects influence performance of Machine Learning systems in computer vision applications. We will see how to make use of these effects to develop new Machine Learning technologies that not only are free from difficulties imposed by high dimensionality, but also are taking advantage of it.

Chapter 4

Blessing of dimensionality in Pattern Recognition

4.1 Statistical properties of high-dimensional data

Definition 1 *A system of vectors* $h_1, h_2, \ldots, h_m \in \mathbb{R}^n, i = 1, 2, \ldots, m$ *is said to be linearly dependent iff there exist* $c_1, c_2, \ldots, c_m, c_i \in \mathbb{R}, i = 1, 2, \ldots, m$ *such that*

$$h_1 c_1 + h_2 c_2 + \ldots + c_m h_m = 0, \tag{4.1}$$

and at least one of $c_1, c_2, \ldots c_m$ *is not equal to zero.*

The same can be written in vector-matrix notation. Let $H = (h_1, h_2, \ldots h_m)$ be an $n \times m$ matrix formed by $h_1, h_2, \ldots, h_m$.

$$\exists c \in \mathbb{R}^m, c \neq 0 : Hc = 0, \tag{4.2}$$

where

$$H = (h_1 h_2 \ldots h_m)$$

is an $n \times m$ matrix formed by $h_1, \ldots, h_m$.

Definition 2 *A system of vectors* $h_1, h_2, \ldots, h_m \in \mathbb{R}^n, i = 1, 2, \ldots, m$ *is said to be linearly independent if it is not linearly dependent:*

$$Hc \neq 0 \, \forall c \in \mathbb{R}^m, c \neq 0. \tag{4.3}$$

A simple fact follows from Definition 2

Proposition 1 *Consider a system of vectors $h_1, h_2, \ldots, h_m$ and let $H = (h_1, h_2, \ldots h_m)$. Let $|.|_{\mathbb{R}^n}$ be a norm on $\mathbb{R}^n$. Then the system $h_1, h_2, \ldots, h_m$ is linearly independent iff there exists an $\varepsilon > 0$ such that*

$$|Hx|_{\mathbb{R}^n} > \varepsilon \quad \forall \quad x \in S^{m-1}(1). \tag{4.4}$$

Proof. Suppose that the system $h_1, h_2, \ldots, h_m$ is linearly independent. Hence (4.3) holds, and given that $x \neq 0$ for all $x \in S^{m-1}(1)$ we thus obtain that $|Hx|_{\mathbb{R}^n} > 0$ for all $x \in S^{m-1}(1)$. Since $|.|_{\mathbb{R}^n}$ is continuous and $S^{m-1}(1)$ is compact we conclude that $|.|_{\mathbb{R}^n}$ takes it minimal and maximal values on $S^{m-1}(1)$. Let

$$\varepsilon = \min_{x \in S^{m-1}(1)} \|Hx\|_{\mathbb{R}^n}.$$

The minimum $|.|$ on $S^{m-1}(1)$ is separated from 0 since otherwise there will exist an $x \in S^{m-1}(1)$ such that $Hx = 0$. Thus (4.4) holds.

Let us now show that (4.4) implies (4.3). For any $c \in \mathbb{R}^m, c \neq 0$ there is an $x \in S^{m-1}(1)$ and an $\alpha \in \mathbb{R}, \alpha \neq 0$ such that $c = \alpha x$. Hence $|Hc|_{\mathbb{R}_n} = |\alpha||Hx|_{\mathbb{R}_n} > |\alpha|\varepsilon > 0$, which automatically assures that (4.3) holds. □

Quantification of linear independence

Standard notions of linear dependence and independence are not always easy to assess numerically when the values of ε in (4.3) are small. Furthermore, checking that for a given system of vectors $h_1, h_2, \ldots h_m$ and some ε, and all $x \in S^{n-1}(1)$, the following holds:

$$|Hx|_{\mathbb{R}^n} > \varepsilon \tag{4.5}$$

may not always be feasible or desirable. Two ways to relax and quantify the conventional notion of linear independence follow from Proposition 1. These are 1) the values of ε in (4.4), and 2) a possibility of introducing the finite measure on $S^{m-1}(1)$ that determines a proportion of $S^{m-1}(1)$ which satisfy (4.4).

Definition 3 *Let $h_1, h_2, \ldots h_m$ be a system of m normalized vectors from $\mathbb{R}^n$: $|h_i|_{\mathbb{R}^n} = 1, i = 1, 2, \ldots, m$. We will say that the system is (ε, θ)-linearly independent (almost*

linearly independent) with respect to μ if

$$\mu(\{x \in S^{m-1}(1) \mid |Hx|_{\mathbb{R}^n} \geq \varepsilon\}) \geq 1 - \theta. \tag{4.6}$$

Similarly, one can formulate a quantification of linear dependence:

Definition 4 *Let $h_1, h_2, \ldots h_m$ be a system of m normalized vectors from $\mathbb{R}^n$: $|h_i|_{\mathbb{R}^n} = 1, i = 1, 2, \ldots, m$. We will say that the system is (ε, θ)-linearly dependent (almost linearly dependent) with respect to μ if*

$$\mu(\{x \in S^{m-1}(1) \mid |Hx|_{\mathbb{R}^n} \leq \varepsilon\}) \geq 1 - \theta. \tag{4.7}$$

Notice that definitions of almost linear dependence and almost linear independence introduced in Definitions 3, 4 are consistent with conventional notions in the sense that the latter can be viewed as limiting cases of the former. Indeed, if μ is a surface area then setting $\theta = 0$ in Definition 3 and picking ε small enough one obtains the equivalent of Definition 2.

The above probabilistic quantification of linear independence has significant implications for data representation in applications. As we shall see in the next sections two seemingly exclusive extremes are likely to hold in higher dimensions. First of all, almost all points of an n-ball concentrate in an ϵ-thickening of an $n-1$ disc. This means that for m sufficiently large a family of randomly and independently chosen vectors $h_1, h_2, \ldots, h_m$ becomes almost linearly dependent. Yet, the values of m for which almost linear independence persists may be exponentially large. Furthermore, the latter situation holds with probability close to one. In other words, the number of almost orthogonal vectors grows exponentially with dimension. More formal statements are provided in Propositions 2, 3 below.

4.2 Measure concentration revisited

We begin with the following statement

Proposition 2 *Let $\mathbb{B}(R)$ be an n-ball of radius R in $\mathbb{R}^n$, and $0 < \sigma < R$ be a*

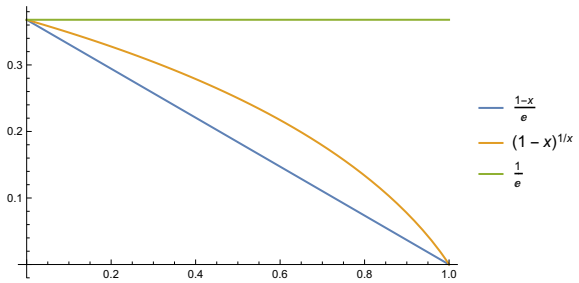


Figure 4.1: Illustration of $(1-x)\frac{1}{e}$, $(1-x)^{\frac{1}{x}}$ and $\frac{1}{e}$ behavior.

non-negative number. Then, for $n \gg 1$ the following estimate holds:

$$\frac{\mathbb{V}(\mathbb{B}^n(R)) - \mathbb{V}(\mathbb{B}^n(R-\sigma))}{\mathbb{V}(\mathbb{B}^n(R))} > 1 - e^{-\frac{n\sigma}{R}}. \tag{4.8}$$

Proof. Noticing that $\mathbb{V}(\mathbb{B}^n(R)) = C_n R^n$ where C_n is a constant independent on R we conclude that

$$\frac{\mathbb{V}(\mathbb{B}^n(R)) - \mathbb{V}(\mathbb{B}^n(R-\sigma))}{\mathbb{V}(\mathbb{B}^n(R))} = 1 - \frac{(R-\sigma)^n}{R^n} = 1 - \Big(\frac{R-\sigma}{R}\Big)^n.$$

The following inequality holds for all $0 < x < 1$:

$$(1-x)\frac{1}{e} < (1-x)^{\frac{1}{x}} < \frac{1}{e}. \tag{4.9}$$

With respect to the right part of (4.9), $(1-x)^{\frac{1}{x}} < e^{-1}$, we notice that $(1-x)^{\frac{1}{x}} < e^{-1} \Leftrightarrow 1 - x < e^{-x}$. The function $y = e^{-x}$ is strictly convex on $\mathbb{R}$, and $y = 1 - x$ is its first-order Taylor approximation at $x = 0$. Thus that $(1-x)^{\frac{1}{x}} < \frac{1}{e}$ holds true $0 < x < 1$ is the consequence of the strict convexity of the exponential e^{-x}.

In order to see that the left part of (4.9) holds as well, consider the following chain of equivalent inequalities for $0 < x < 1$:

$$(1-x)e^{-1} < (1-x)^{\frac{1}{x}} \Leftrightarrow e^{-x} < (1-x)^{1-x} \Leftrightarrow -x < (1-x)\ln(1-x). \tag{4.10}$$

Again, $y = (1-x)\ln(1-x)$ is strictly convex on $(-\infty, 1)$, and $y = -x$ is its first Taylor approximation at $x = 0$. Thus the left part of inequality is a consequence of the strict convexity of $(1-x)\ln(1-x)$. Also see Figure 4.1 for an additional illustration.

Using (4.9) one can show that

$$\left[e\Big(1-\frac{\sigma}{R}\Big)^{-1}\right]^{-\frac{n\sigma}{R}} < \Big(1-\frac{\sigma}{R}\Big)^{n} < e^{-\frac{n\sigma}{R}}. \tag{4.11}$$

To show this, one puts $x = \sigma/R$ in inequality 4.9. Thus we get:

$$\Big(1-\frac{\sigma}{R}\Big)e^{-1} < \Big(1-\frac{\sigma}{R}\Big)^{\frac{R}{\sigma}} < e^{-1}.$$

Exponentiating all members with the power of $\frac{n\sigma}{R}$, one obtains 4.11. Moreover, for σ/R sufficiently small we obtain

$$\Big(1-\frac{\sigma}{R}\Big)^{n} \sim e^{-\frac{n\sigma}{R}}.$$

with accuracy estimate following from (4.9). □

In accordance with Proposition 2 the volume of n-ball of Radius R, for n sufficiently large is concentrated in a thin layer around its surface. Furthermore, in this thin layer the volume of an n-ball is concentrated around the largest equator of the corresponding $n-1$ sphere, $S^{n-1}(R)$.

Proposition 3 *Let $\mathbb{B}^n(R)$ be an n-ball of radius R in $\mathbb{R}^n$, and $0, \sigma < R$. Let $\mathbb{D}^{n-1}_{\sigma}(R)$ be a σ-thickening of an $(n-1)$-disk $\mathbb{D}^{n-1}(R)$. Then*

$$\frac{\mathbb{V}(\mathbb{B}^n(R)) - \mathbb{V}(\mathbb{D}^{n-1}_{\sigma}(R))}{\mathbb{V}(\mathbb{B}^n(R))} < e^{-\frac{n\sigma^2}{2R^2}}.$$

Proof. Consider

$$\frac{\mathbb{V}(\mathbb{B}^n(R)) - \mathbb{V}(\mathbb{D}^{n-1}_{\sigma}(\sqrt{R^2-\sigma^2}))}{\mathbb{V}(\mathbb{B}^n(R))} = 1 - \Big(1-\frac{\sigma^2}{R^2}\Big)^{\frac{n}{2}}.$$

Using (4.9) we obtain

$$1 - \Big(1-\frac{\sigma^2}{R^2}\Big)^{\frac{n}{2}} > 1 - e^{-\frac{n\sigma^2}{2R^2}}$$

and the result follows from

$$\mathbb{V}(\mathbb{B}^n(R)) - \mathbb{V}(\mathbb{D}^{n-1}_{\sigma}(R)) \leq \mathbb{V}(\mathbb{D}^{n-1}_{\sigma}(\sqrt{R^2-\sigma^2})).$$

□

In high dimension the volume of the ball is concentrated in a thin layer near the sphere. Therefore, the estimate of the volume of the disk automatically provides an estimate of the surface of the corresponding waist of the sphere. Let us produce this estimate explicitly. The proportion of $\mathcal{S}^{n-1}(1)$ belonging to the cap (shaded part of the sphere in Fig. 4.2) equals the proportion of the solid ball that lies in the corresponding spherical cone (cf. [24], Fig. 11). The latter consist of two parts: one is the cone of height δ and radius of the base $\sqrt{1-\delta^2}$. The volume of the second part can be bounded from above by the half of the volume of the ball with radius $\sqrt{1-\delta^2}$. If we use the Stirling formula for the volume of high-dimensional ball $V_n(R) \sim \frac{1}{\sqrt{n\pi}} \left(\frac{2\pi e}{n}\right)^{n/2} R^n$ then we obtain that the fraction of the waist of the width 2δ is $1-(1+O(\delta/\sqrt{n}))e^{-\frac{n\delta^2}{2}}$. Indeed

$$\begin{aligned} V_n(R) &= \frac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)} R^n,\ V_n(R) = 2\pi \left(\frac{R}{\sqrt{n}}\right)^2 V_{n-2}(R), \\ V_n(R) &= R\sqrt{\pi}\frac{\Gamma(\frac{n+1}{2})}{\Gamma(\frac{n+1}{2}+\frac{1}{2})} V_{n-1}(R). \end{aligned} \tag{4.12}$$

The estimate now follows from

$$\Gamma(x) = x^{x-\frac{1}{2}} e^{-x} \sqrt{2\pi} \left(1 + \frac{1}{12x} + R_2(x)\right),$$

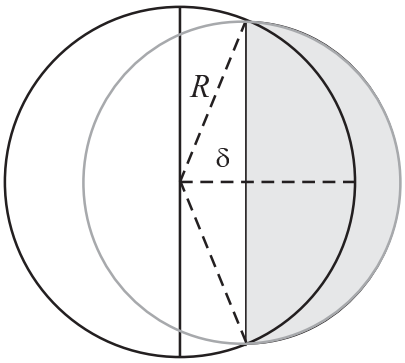
where the reminder $R_2(x)$ can be bounded as

$$|R_2(x)| \leq \frac{1+\frac{1}{6}\pi^2}{2\pi^3 x^2} \tag{4.13}$$

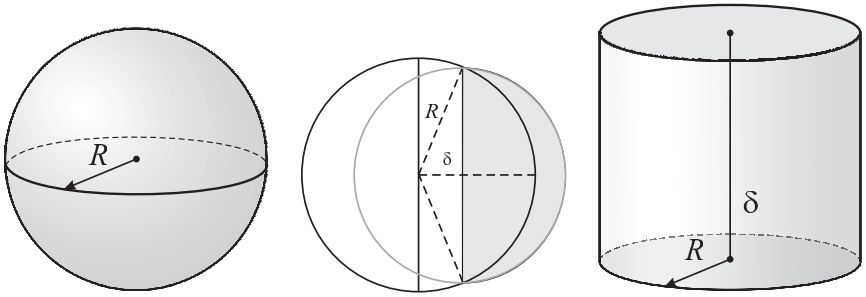
(see (3.11) from [13] for details). This estimate improves the textbook estimate $1-2e^{-\frac{n\delta^2}{2}}$ for large n [24].

The concentration effect described in Proposition 3 implies that in high dimensions nearly all independently and randomly drawn vectors will belong to a δ-thickening of a set of vectors that are linearly dependent: the $n-1$-disc $\mathcal{D}^{n-1}(R)$. On the other hand a set of $n-1$ vectors with probability close to one spans almost all vectors. We note that the latter property holds for systems of $n-k$, $k>1$ vectors too which follows immediately from [4]:

Theorem 2 (Theorem 3.1 in [4]) *Let E_k be a k-dimensional subspace of $\mathbb{R}^n$, and*



(a) Covering of an n-ball



(b) $\mathcal{V} \sim \sqrt{\frac{2\pi}{n+1}} R V_{n-1}(R)$ (c) $\mathcal{V} = \frac{1}{n+1}\delta V_{n-1}(R)$ (d) $\mathcal{V} = \delta V_{n-1}(R)$

Figure 4.2: Illustration to the estimate of the 2δ-width of the waist of the sphere. For (b), $n \gg 1$. More accurate estimate with evaluation of the reminder can be extracted from (4.12), (4.13).

denote by $\mu((E_k)_\varepsilon)$ *the Haar measure on the sphere* $\mathcal{S}(1)$ *of the set of points within a geodesic distance smaller than* ε *of* E_k. *We write* $k = \lambda n$. *Fix* $0 < \varepsilon < \pi/2$ *and* $0 < \lambda < 1$. *The following estimates hold as* $n \to \infty$

(i) If $\sin^2 \varepsilon > 1 - \lambda$, *then*

$$\mu((E_k)_\varepsilon) \simeq 1 - \frac{1}{\sqrt{n\pi}} \frac{\sqrt{\lambda(1-\lambda)}}{\sin^2 \varepsilon - (1-\lambda)} e^{\frac{n}{2} u(\lambda, \varepsilon)}.$$

(ii) If $\sin^2 \varepsilon < 1 - \lambda$, *then*

$$\mu((E_k)_\varepsilon) \simeq \frac{1}{\sqrt{n\pi}} \frac{\sqrt{\lambda(1-\lambda)}}{(1-\lambda) - \sin^2 \varepsilon},$$

where $u(\lambda, \varepsilon) = (1-\lambda) \log \frac{1-\lambda}{\sin^2 \varepsilon} + \lambda \log \frac{\lambda}{\cos^2 \varepsilon}$.

Since nearly all vectors in $\mathcal{B}^n(1)$ are concentrated in an ε-disc it is interesting to know how many pairwise almost orthogonal vectors can be found in this set. It turns out that this number is exponentially large. Detailed analysis and derivations are provided in the next section.

4.3 Almost orthogonality in high dimensions

Select two small positive numbers ϵ and θ. Let us generate randomly and independently N vectors $x_1, \ldots, x_N$ on $S^{n-1}(1)$. We are interested in the probability P that all N random vectors are pairwise ϵ-orthogonal, i.e. $|(x_i, x_j)| < \epsilon$ for $i, j = 1, \ldots, N$ and $i \neq j$. For which N is this $P > 1 - \theta$?

Propositions 2 and 3 suggest that for $n \gg 1$ almost the entire volume of n-ball $\mathbb{B}^n(1)$ is concentrated in ε-thickening of its largest equator. Moreover, for an arbitrarily chosen point p on the surface of $\mathbb{B}^n(1)$ almost all points of the ball belong to the set $\mathbb{DC}_\varepsilon^{n-1}(1)$ comprising of the difference of $\mathbb{D}_\varepsilon^{n-1}(1)$ the following estimate holds:

$$|\cos(\angle(p, x))| \leq \varepsilon \, \forall \, x \in \mathbb{DC}_\varepsilon^{n-1}(1).$$

this property means that the vector p is almost orthogonal to nearly all remaining points in $\mathbb{B}^n(1)$.

Another way of looking at this could be as follows. In accordance with Proposition 2, almost all points of the ball $\mathbb{B}$ are concentrated around a ε-thickening of the surface. At the same time they are also concentrated in $\mathbb{D}_{\varepsilon}^{n-1}(1)$. The length of such points, x, satisfy $1-\varepsilon \leq \|x\| \leq 1$, and hence

$$|\cos(\angle(p,x))| \leq \frac{|p^T x|}{1-\varepsilon} \leq \frac{\varepsilon}{1-\varepsilon}.$$

Let us determine the number of independent vectors which are pairwise ε-orthogonal with probability $1-\theta$. The volume taken by all vectors that are almost orthogonal to a given vector on a unit sphere can be estimated from Proposition 3. Consider the following products:

$$P(\varepsilon, N) = \prod_{k=1}^{N} (1 - ke^{-\frac{n\varepsilon^2}{2}}). \tag{4.14}$$

The value of $P(\varepsilon, N)$ is an estimate from below of the probability of a set of $N+1$ independent random vectors to be pairwise ε-orthogonal. Indeed, for one vector h_1 the fraction of vectors which are not ε-orthogonal to h_1 is evaluated as $e^{-n\sigma^2}$. Therefore, for k vectors $h_1, h_2, \ldots, h_k$, the fraction of vectors which are not ε-orthogonal to $h_1, h_2, \ldots, h_k$ is at most $ke^{-n\sigma^2}$. The probability to select randomly a vector h_{k+1}, which is ε-orthogonal to $h_1, h_2, \ldots, h_k$ is higher than $1-ke^{-n\sigma^2}$. Vectors are selected independently, therefore we have the estimate (4.14).

The value of $P(\varepsilon, N)$ in (4.14) can be estimated as follows. For $Ne^{-\frac{n\varepsilon^2}{2}} < 1$:

$$P(\varepsilon, N) > (1 - Ne^{-\frac{n\varepsilon^2}{2}})^N \sim e^{-N^2 e^{-\frac{n\varepsilon^2}{2}}}. \tag{4.15}$$

According to (4.15), if $P(\varepsilon, N)$ is set to be exceeding a certain value, $1-\theta$, the number of pairwise almost orthogonal vectors in $\mathbb{B}^n(1)$ will have the following asymptotic estimate: for

$$\boxed{N \leq e^{\frac{\varepsilon^2 n}{4}} \left[\log\left(\frac{1}{1-\theta}\right)\right]^{\frac{1}{2}}} \tag{4.16}$$

all random vectors $h_1, h_2, \ldots, h_k$ are pairwise ε-orthogonal with probability $P > 1-\theta$.

Estimate (4.15) of (4.14) can be refined if we apply log to the right hand size of

(4.14).

$$\log P(\varepsilon, N) = \sum_{k=1}^{N} \log\left(1 - ke^{-\frac{n\varepsilon^2}{2}}\right), \tag{4.17}$$

and estimate the above sum using the integral

$$J(z) = \int_0^z \log(1 - xr)dx = \frac{rz - 1}{r}\log(1 - rz) - z, r = e^{-\frac{n\varepsilon^2}{2}}. \tag{4.18}$$

Since $\log P(\varepsilon, N)$ is monotone for $e^{\frac{n\varepsilon^2}{2}} > N \geq 1$ we can conclude that $J(N+1) \leq \log P(\varepsilon, N) \leq J(N)$. Furthermore, given that

$$\log(1 - x) \leq -\frac{2x}{2 - x} \text{ for all } x \in [0, 1], \tag{4.19}$$

the following holds:

$$J(z) = \frac{2z(rz - 1)}{rz - 2} - z \tag{4.20}$$

for all $rz \in [0, 1]$. Hence

$$\frac{2(N + 1)(r(N - 1) - 1)}{r(N + 1) - 2} - (N + 1) \leq P(\varepsilon, N) = \log(1 - \theta). \tag{4.21}$$

Multiplying both sides by $r(N + 1) - 2$ we have

$$r(N + 1)^2 - \log(1 - \theta)r(N + 1) + 2\log(1 - \theta) \leq 0, \tag{4.22}$$

and solving this for N gives us the following estimate

$$\boxed{N \leq \sqrt{\frac{\log^2(1 - \theta)}{4} + 2\log\frac{1}{1 - \theta}e^{\frac{n\varepsilon^2}{2}}} + \frac{\log(1 - \theta)}{2}}. \tag{4.23}$$

Notice that the refined estimate (4.23) has asymptotic exponential rate of order $e^{\frac{n\varepsilon^2}{4}}$(with respect to dimensionality n) which is identical to the one derived in (4.16).

Estimates (4.23), (4.16) derived above suggest that, for θ sufficiently small a set of N randomly and independently chosen vectors in $\mathbb{B}^n(1)$ will be pairwise ε-orthogonal with probability $1 - \theta$ for

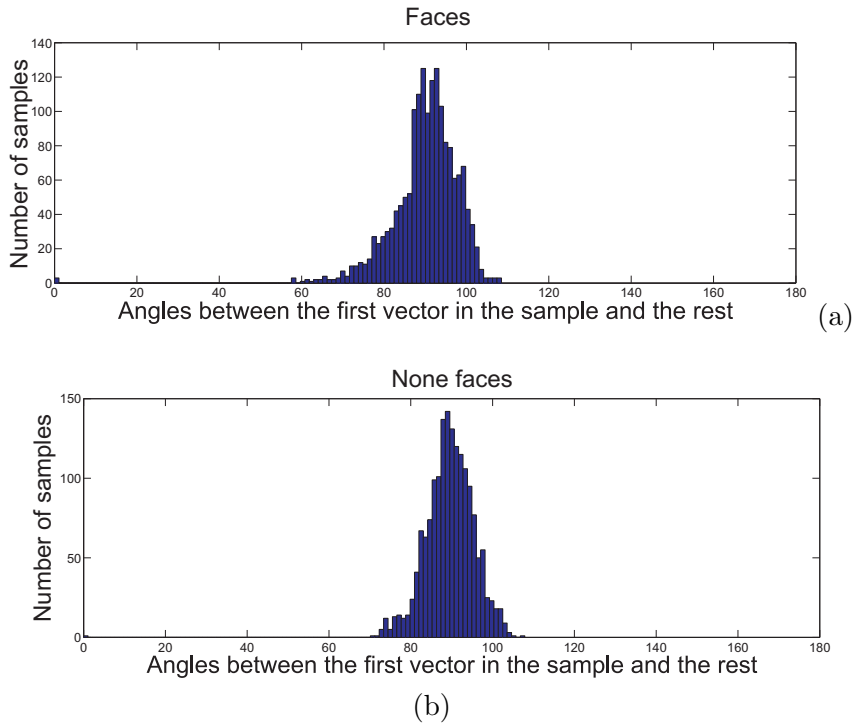


Figure 4.3: Measure concentration in high dimensions. Panel (a) shows histogram of angles between a randomly chosen feature vector in the set of "faces" and the rest of the vectors in the class. Panel (b) shows histogram of angels between a randomly chosen feature vector in the set of "non-faces" (negatives) and the rest of vectors in this class.

$$N < e^{\frac{\varepsilon^2 n}{4}} \theta^{\frac{1}{2}}. \tag{4.24}$$

Measure concentration effects

So far measure concentration effects have been discussed for idealized objects such as $\mathcal{S}^{n-1}(R)$ and $\mathcal{B}^n(R)$. The phenomenon, however, broadly applies to other objects whose geometric and formal description is not limited to the former.

In order to illustrate this point we analysed a database of HOG feature vectors [19] containing representations of images of faces[1] as well as the negatives (non-faces). Each feature vector has 1920 components, and hence belongs to $\mathbb{R}^n$ with $n = 1920$. Vectors of each classes have been centered and normalized so that they belong to the hypercube $[-1, 1]^n$. Fig. 4.3 shows distributions of angles between a randomly chosen vector (1-st) and that of the rest in their respective classes. As one can see from this figure, the angles concentrate in a vicinity of $\pi/2$ which is

[1]The database has been developed by Apical LTD.

consistent with our derivations for points in $\mathcal{B}^n(1)$.

Another interesting effect of measure concentration is exponential growth of the lengths of chains of randomly chosen vectors which are pairwise almost orthogonal. In order to illustrate and assess validity of our estimates (4.16), (4.23) the following numerical experiments have been performed. A point is first randomly selected in a hypercube $[-1,1]^n$ of some given dimension. The second point is randomly chosen in the same hypercube. These two points correspond to two vectors randomly drawn in $[-1,1]^n$. If the angle between the vectors was within $\pi/2 \pm 0.037\pi/2$ then the vector was retained. At the next step a new vector is generated in the same hypercube, and its angles with the previously generated vectors are evaluated. If these angles are within $0.037\pi/2$ of $\pi/2$ then the vector is retained. The process is repeated until the chain of almost orthogonality breaks, and the number of such pairwise almost orthogonal vectors (length of the chain) is recorded. Results are shown in Figure 4.4. Red line corresponds to the conservative theoretical estimate (4.16), green curve shows refined estimate, (4.23), and box plot shows lengths of pairwise almost orthogonal chains as a function of dimension. The value of θ was set to 0.1 for both theoretical estimates, and our choice of precision margins $\pi/2 \pm 0.037\pi/2$ corresponds to $\varepsilon = \cos(0.963\pi/2) = 0.0581$. As we can see from this figure our empirical observations are well aligned with theoretical predictions.

Approximation of a constant: dimensionality blowup

Another illustration of measure concentration and orthogonality effects belongs to the field of function approximation. Let suppose we want to approximate a given continuous function defined on interval $[0,1]$ be linear combinations of the following type:

$$f_N(x) = \sum_{i=1}^{N} c_i \phi(a_i, \sigma_i, x), \tag{4.25}$$

where the function ϕ is defined as follows:

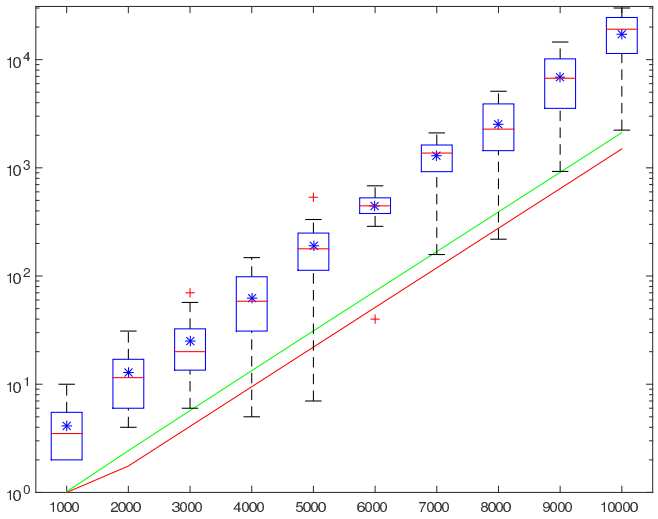


Figure 4.4: Lengths N of pairwise almost orthogonal chains of vectors that are independently randomly sampled from $[-1,1]^n$ as a function of dimension, n. For each n 20 pairwise almost orthogonal chains where constructed numerically. Boxplots show the second and third quartiles of this data for each n, red bars correspond to the medians, and blue stars indicate means. Red curve shows theoretical bound (4.16), and green curve shows refined estimate (4.23).

$$\phi(a,\sigma,x) = \begin{cases} 0, & x > a + \sigma/2, \\ 1, & a - \sigma/2 \leq x \leq a + \sigma/2, \\ 0, & x < a - \sigma/2. \end{cases}$$

For simplicity we suppose that this function is a constant on a defined interval:

$$f(x) = 1 \ \forall \ x \in [0,1].$$

A linear combination of $\phi(a_i,\sigma_i,x)$ can approximate any continuous function on $[0,1]$. Furthermore, the chosen function f can be represented by just a single element with $a = 0.5, \sigma = 0.5 : f(x) = \phi(0.5, 0.5, x)$. Since we make no assumptions about these parameters, we approximate the function f with linear combinations (4.25) in which values of a_i, σ_i were chosen randomly on the interval $[0,1]$ and the values of c_i were chosen as follows

$$c_1, c_2, \ldots, c_N = \arg \min_{c_1,c_2,\ldots,c_N} \int_0^1 (f(x) - f_N(x))^2 dx. \tag{4.26}$$

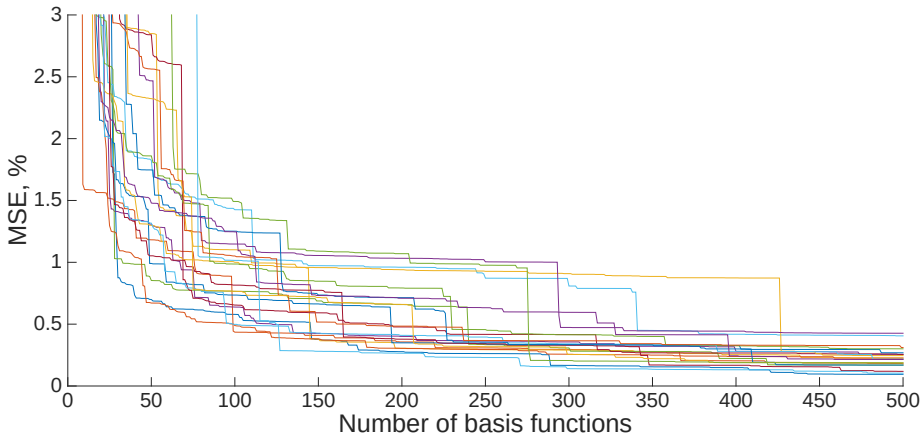


Figure 4.5: Errors of approximation of $f(x) = 1$ be linear combinations of $f_N(x)$, (4.25).

In order to evaluate the performance of approximation as a function of N the following iterative procedure has been used. In the first step the values of a_1 and σ_1 are randomly drawn from the interval $[0, 1]$. This is followed by finding the value c_1 in accordance with (4.26). Next values of a_1, σ_1 are drawn randomly from the interval $[0, 1]$, followed by finding an optimal pair of weights c_1, c_2. The L_2 error of approximation is recorded at each step. The process is run until $N = 500$.

Fig. 4.5 presents 20 different error curves corresponding to different growing systems of functions $\{\phi(a_i, \sigma_i, .)\}_{i=1}^N$. Even though the problem is both simple and has an explicit solution, the performance of such an approximation scheme is far from being ideal. One can see a big initial drop in error for values of $N < 100$. However, after this value the error decays very slowly and later the rate of the error almost stalls.

One potential explanation of this effect is as follows. Fig. 4.6 shows functions $f(x) - f_N(x)$ for $N = 5, 50$ and 500 along a single typical curve from Fig. 4.5. It is clear from the picture that error functions become more patchy and spiky as N grows. The individual spikes are randomly distributed in $[0, 1]$ and its thickness converges to zero. To compensate for this kind of error one needs to be able to generate very narrow $\phi(a_i, \sigma_i, .)$ which, in addition, have to be placed in a right location. However, in accordance with (4.16) one needs to accumulate an exponential number of functions to overcome this effect. This is reflected in a very slow convergence rate at the end of the process.

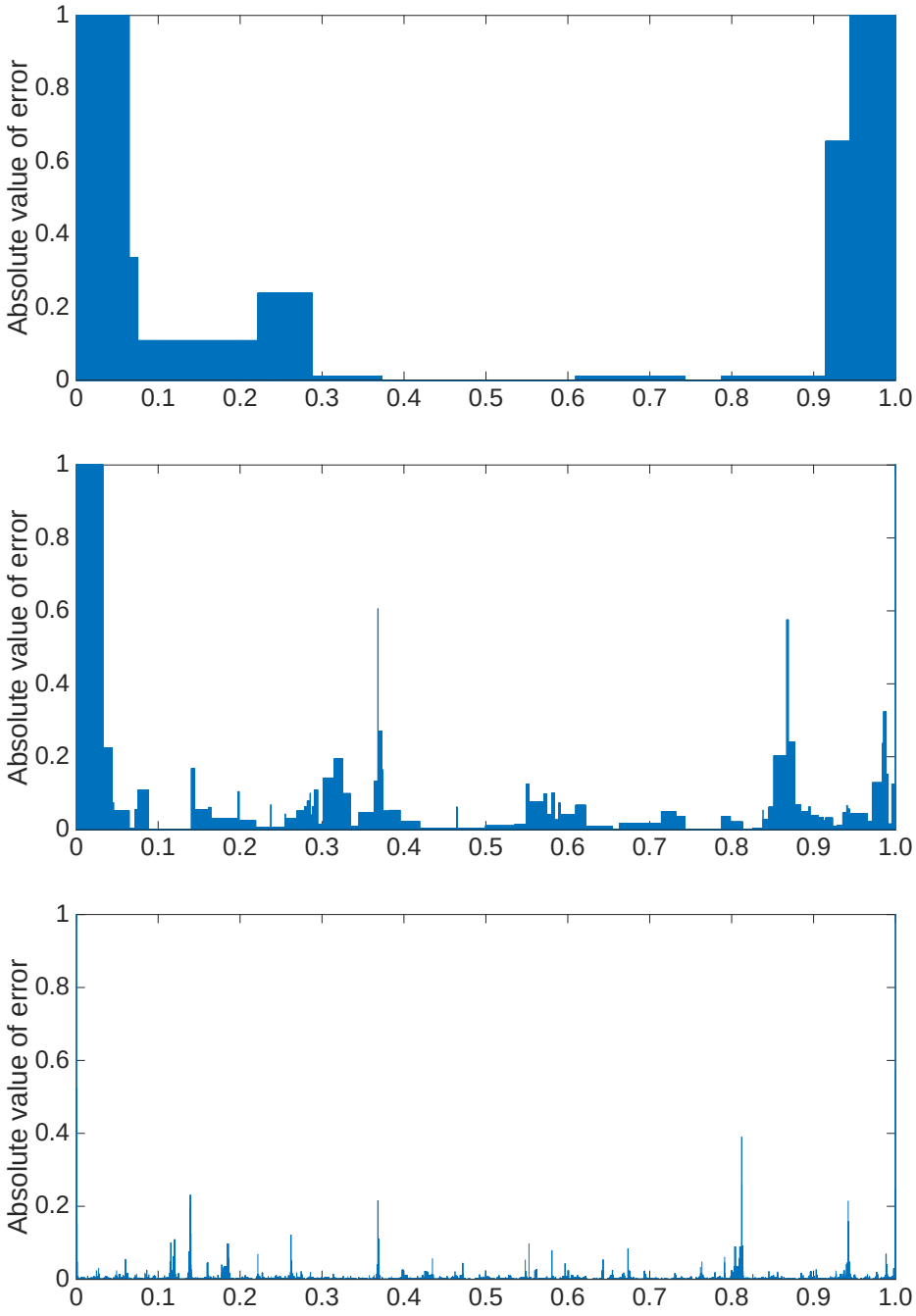


Figure 4.6: Error function $f(x) - f_N(x)$ for $N = 5$ (panel (a)), $N = 50$ (panel (b)) and $N = 500$ (panel (c)).

4.4 Stochastic separation theorems

In this section we consider an important consequence of the above result: stochastic separation theorems as developed in [31]. These theorems will be used in Section 4.5 and that is why their presentation is included here.

Let us consider a basic example where points are randomly equidistributed on the unit ball $B_n(1)$ in $\mathbb{R}^n$.

Definition 5 *Let X and Y be subsets of $\mathbb{R}^n$. We say that a linear functional l on $\mathbb{R}^n$ separates X and Y if there exists a $t \in B$ such that*

$$l(x) > t > l(y) \forall x \in X, y \in Y. \tag{4.27}$$

Let $\mathcal{M}$ be i.i.d sample drawn from the equidistribution on the unit ball $B_n(1)$. We begin with evaluating the probability that a single element x randomly and independently selected from the same equidistribution can be separated from $\mathcal{M}$ by a linear functional. This probability denoted as $P_1(\mathcal{M}, n)$, is estimated in the theorem below.

Theorem 3 ([31]) *Consider an equidistribution in a unit ball $B_n(1)$ in $\mathbb{R}^n$, and let $\mathcal{M}$ be an i.i.d. sample from the distribution. Then*

$$P_1(\mathcal{M}, n) \geq \max_{\varepsilon \in (0,1)} (1 - (1-\varepsilon)^n) \Big(1 - \frac{\rho(\varepsilon)^n}{2}\Big)^M, \quad \rho(\varepsilon) = (1 - (1-\varepsilon)^2)^{\frac{1}{2}}. \tag{4.28}$$

The proof of this theorem is mostly contained in the following lemma

Lemma 1 ([31]) *Let y be random point from an equidistribution on a unit ball $B_n(1)$. Let $x \in B_n(1)$ be a point inside the ball with $1 > |x| > 1 - \varepsilon > 0$. Then*

$$P\Big(\Big\langle \frac{x}{|x|}, y \Big\rangle < 1 - \varepsilon\Big) \geq 1 - \frac{\rho(\varepsilon)^n}{2}. \tag{4.29}$$

Proof. Recall that [55] $\mathcal{V}(B_n(r)) = r^n \mathcal{V}(B_n(1))$ for all $n \in \mathbb{N}$ $r > 0$. The point x is inside spherical cap $C_n(\varepsilon)$:

$$C_n(\varepsilon) = B_n(1) \cap \Big\{ \xi \in \mathbb{R}^n \Big| \Big\langle \frac{x}{|x|}, \xi \Big\rangle < 1 - \varepsilon \Big\}. \tag{4.30}$$

The volume of this cap can be estimated from above [9] as

$$\mathcal{V}(C_n(\varepsilon)) \leq \frac{1}{2}\mathcal{V}(B_n(1))\rho(\varepsilon)^n. \tag{4.31}$$

The probability that the point $y \in \mathcal{M}$ is outside of $C_n(\varepsilon)$ is equal to $1 - \mathcal{V}(C_n(\varepsilon))/\mathcal{V}(B_n(1))$. Probability (4.29) immediately follows from (4.31). □

Let us now return the proof of the theorem. If x is selected independently from the equidistribution on $B_n(1)$ then the probabilities that $x = 0$ or that it is on the boundary of the ball are 0. Let $x \neq 0$ be an interior of $B_n(1)$. According to Lemma 1, the probability that a linear functional l separates x from a point $y \in \mathcal{M}$ is larger than $1 - 0.5\rho(\varepsilon)^n$. Given that points of the set $\mathcal{M}$ are i.i.d. in accordance to the equidistribution on $B_n(1)$, the probability that l separates x from $\mathcal{M}$ is no smaller than $(1 - 0.5\rho(\varepsilon)^n)^M$.

On the other hand

$$P(1 > |x| > 1 - \varepsilon \,|\, x \in B_n(1)) = (1 - (1 - \varepsilon)^n).$$

Given that x and $y \in \mathcal{M}$ are independently drawn from the same equidistribution and that the probabilities of randomly selecting the point x exactly on the boundary of $B_n(1)$ or in its centre are zero, we can conclude that

$$P_1(\mathcal{M}, n) \geq (1 - (1 - \varepsilon)^n)(1 - 1/2\rho(\varepsilon)^n)^M. \tag{4.32}$$

Finally, noticing that (4.32) holds for all $\varepsilon \in (0, 1)$ including the value of ε maximizing the rhs of (4.32), we can conclude that (4.28) holds true too. □

Remark 1 ([31]) *For $\rho(\varepsilon)^n$ small enough the term $(1 - \frac{\rho^2}{2})^M$ can be approximated as $(1 - \rho^2/2)^M \approx e^{-M\frac{\rho^2}{2}}$. Thus Equation (4.28) becomes*

$$P_1(\mathcal{M}, n) \gtrsim \max_{\varepsilon \in (0,1)} (1 - (1 - \varepsilon)^n) e^{-M\frac{\rho^2}{2}}, \quad \rho(\varepsilon) \ll 1. \tag{4.33}$$

For example, for dimensionality $n = 50$, $\varepsilon = 1/5$ and $\rho = 3/5$, (4.33) becomes:

$$P_1(\mathcal{M}, 50) \gtrsim 0.99998 e^{(-4 \times 10^{-12} M)}.$$

For $M \leq 10^9$ this estimates gives $P_1(\mathcal{M}, 50) \gtrsim 0.996$. Therefore in dimension 50 or higher a randomly chosen point is linearly separable from a set of 10^9 points with probability 0.996.

Remark 2 ([31]) *If x is an element from sample $\mathcal{M}$ then the probability p that x is separable from all other points in the sample is bounded below by $P_1(\mathcal{M}, n)$.*

Remark 3 ([31]) *Let $x \in \mathcal{M}$ be a given query point. This query point determines the value of $\varepsilon = 1 - \|x\|$ as the least ε-thickening of the unit sphere containing x. With probability 1 the values of ε belong to the open interval $(0, 1)$. Let $p \in (0, 1)$ be the desired probability that x is separated from the rest of the sample $\mathcal{M}$. It is clear that the estimate $P_1(\mathcal{M}, n) \geq p$ holds for M from some interval $[1, \overline{M}]$. Interestingly, for n large enough, the maximal number $\overline{M}$ is exponentially large in dimension n. Indeed, let us fix the values of $\varepsilon \in (0, 1)$ and $p \in (0, 1)$. Then we find the estimate of the maximal possible sample size for which $P_1(\mathcal{M}, n) \geq p$ remains valid:*

$$\max\{\overline{M}\} \geq \frac{\ln(p)}{\ln\left(1 - \frac{\rho(\varepsilon)^n}{2}\right)} - \frac{\ln(1 - (1-\varepsilon)^n)}{\ln\left(1 - \frac{\rho(\varepsilon)^n}{2}\right)}.$$

Using

$$\frac{x}{x-1} \leq \ln(1-x) \leq -x$$

we conclude that

$$\max\{\overline{M}\} \geq \left(\frac{1}{\rho(\varepsilon)}\right)^n C(n, \varepsilon),$$

where

$$C(n, \varepsilon) = 2\left(|\ln(p)|\left(1 - \frac{\rho(\varepsilon)^n}{2}\right) - |\ln(1 - (1-\varepsilon)^n)|\right).$$

Observe that for any fixed $\varepsilon \in (0, 1)$ there is an $N(\varepsilon)$ large enough such that $C(n, \varepsilon) \geq |\ln(p)|$ for all $n \geq N(\varepsilon)$. Hence, for n sufficiently large the following estimate holds:

$$\max\{\overline{M}\} \geq e^{n \ln\left(\rho(\varepsilon)^{-1}\right)} |\ln(p)|. \tag{4.34}$$

Equation (4.34) can be viewed as a *separation capacity* estimate of linear functionals. This estimate links the level of desired performance specified by p, the maximal size of the sample, M, and parameters of the data, n and ε.

4.4.1 Extreme points of a random finite set

So far the question of separability of a single random point x, drawn from equidistribution on $B_n(1)$, from i.i.d. sample $\mathcal{M}$ has been discussed. However, in practice the data or a training set are fixed. It is important to know if the point linear separability property formulated in Theorem 3 persists in one form or another when the test point x belongs to the sample $\mathcal{M}$ itself. The question is then if the probability $P_M(\mathcal{M}, n)$ that each point $y \in \mathcal{M}$ is linearly separable from $\mathcal{M} \setminus \{y\}$ is close to one in high dimensions? If such a property holds then one can conclude that in high dimensions all points of some set $\mathcal{M}$ are vertices (extreme points) of the convex hull of $\mathcal{M}$ and none of $y \in \mathcal{M}$ is a convex combination of other points. The fact that this is the case follows from Theorem 4.

Theorem 4 ([31]) *Consider an equidistribution in a unit ball $B_n(1)$ in $\mathbb{R}^n$, and let $\mathcal{M}$ be i.i.d. sample from this distribution. Then*

$$P_M(\mathcal{M}, n) \geq \max_{\varepsilon \in (0,1)} \Big[(1 - (1-\varepsilon)^n) \Big(1 - (M-1) \frac{\rho(\varepsilon)^n}{2} \Big) \Big]. \tag{4.35}$$

Proof of Theorem 4. Let $P : \mathcal{F} \to [0,1]$ be a probability measure and $A_i \in \mathcal{F}, i = 1, \ldots, M$. It is known that

$$P(A_1 \vee A_2 \vee \ldots \vee A_M) \leq \sum_{i=1}^{M} P(A_i). \tag{4.36}$$

The probability that a test point y is in the ε-vicinity of the boundary of $B_n(1)$ is $1 - (1-\varepsilon)^n$. Fix $y \in \mathcal{M}$ and construct spherical caps $C_n(\varepsilon)$ for each element in $\mathcal{M} \setminus \{y\}$ as specified by (4.30) but with x replaced with corresponding points from $\mathcal{M} \setminus \{y\}$. According to (4.36) and Lemma 1, the probability that y is in any of these caps is no larger than $(M-1)\frac{\rho(\varepsilon)^n}{2}$. Hence the probability that a point $y \in \mathcal{M}$ is separable from $\mathcal{M} \setminus \{y\}$ is larger or equal to $(1-(1-\varepsilon)^n)\Big(1-(M-1)\frac{\rho(\varepsilon)^n}{2}\Big)$. Given that points of $\mathcal{M}$ are drawn independently and that there are exactly M points in $\mathcal{M}$, the probability that every single point is linearly separable from the rest satisfies (4.35). □

Remark 4 ([31]) *Note that employing (4.36) one can obtain another estimate of P_M:*

$$P_M(\mathcal{M}, n) \geq 1 - M(1 - P_1(\mathcal{M}, n)). \tag{4.37}$$

We can utilize this estimate together with (4.34) and estimate the maximal size of the sample from below. Indeed, if we require that $P_M(\mathcal{M}, n) \geq q$ for some probability q, $0 < q < 1$, then it is sufficient that $P_1(\mathcal{M}, n) > p$, where $1 - p = \frac{1}{M}(1 - q)$. Using $|\ln p| > 1 - p$ in (4.34) , we get that that $P_M(\mathcal{M}, n) > q$ if $M \leq \tilde{M}$ for some maximal value $\tilde{M}$, that satisfies the inequality

$$\tilde{M} \geq e^{n \ln (\rho(\varepsilon)^{-1})} \frac{1 - q}{\tilde{M}}.$$

Immediately from this inequality we get the explicit exponential estimate of the maximum of $\tilde{M}$ from below:

$$\max\{\tilde{M}\} \geq e^{\frac{1}{2} n \ln \rho(\varepsilon)^{-1}} \sqrt{1 - q}. \tag{4.38}$$

4.4.2 Two-functional (two-neuron) separation in finite sets

In the previous section we have provided estimates of the probabilities that a single linear classifier or a learning machine can separate a given point from the rest of data and showed that two disjoint weakly compact subsets of a topological vector space can be separated by small networks of perceptrons. Let us now see how employing small networks may improve probabilities of separation of a point from the rest of the data in high dimensions. In particular, we will consider the case of a two neuron separation in which the network is a simple cascade comprised of two perceptrons followed by a conjunction operation.

Before, however going any further we need to clarify and adjust the notion of separability of a point from a finite dataset by a network so that the question makes some practical and theoretical sense. Consider for example a problem of separating a test point by just two perceptrons. If one projects the data onto the 2D plane so that the projections of the test point and any other point from the rest of the data do not coincide, then the problem always has a solution. This is illustrated with the diagram in Fig. 4.7. According to this diagram any given point an arbitrary but finite dataset could be cut out from the rest of the data by just two lines that

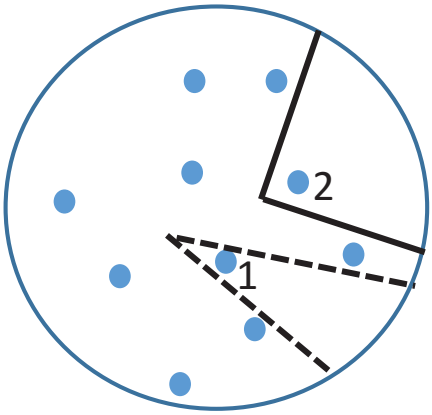


Figure 4.7: Two-neuron separation in finite sets. Every point can be separated by a sufficiently acute angle or highly correlated neurons: Point 1 is separated from other points by an acute angle (dashed lines). Point 2 is separated by a right angle (non-correlated neurons), but cannot be separated by a linear functional (i.e. by a straight line).

intersect at a sufficiently acute angle. Thus two hyperplanes $\{x|l_1(x) = \theta_1\}$ and $\{x|l_2(x) = \theta_2\}$ whose projections are already constitute the two-neuron separating cascade. The problem here is that because the angle is acute, the weights of the linear functionals are highly correlated. This implies that the robustness of such a solution is low and small perturbations in these coefficients can lead to the loss of separation. This motivates the alternate solution where the coefficients are uncorrelated and two hyperplanes are (almost) orthogonal.

Let us now analyze the problem of separation of a random i.i.d. finite sample $\mathcal{M}$ drawn from an equidistribution in $B_n(1)$ from a point x drawn independently from the same distribution. Formally, we are interested in the probability $\mathcal{P}_1(\mathcal{M}, n)$ that a two-neuron cascade with uncorrelated synaptic weights separates x from $\mathcal{M}$. An estimate of this probability is provided in the next theorem.

Theorem 5 ([31]) *Consider an equidistribution in a unit ball $B_n(1)$ in $\mathbb{R}^n$, and let $\mathcal{M}$ be an i.i.d. sample from this distribution. Then*

$$\begin{aligned}\mathcal{P}_1(\mathcal{M}, n) \geq \max_{\varepsilon\in(0,1)} (1-(1-\varepsilon)^n)\times \\ \left(1-\frac{\rho(\varepsilon)^n}{2}\right)^M e^{(M-n+1)\left[\frac{\frac{\rho(\varepsilon)^n}{2}}{1-\frac{\rho(\varepsilon)^n}{2}}\right]}\times \\ \left(1-\frac{1}{n!}\left((M-n+1)\frac{\frac{\rho(\varepsilon)^n}{2}}{1-\frac{\rho(\varepsilon)^n}{2}}\right)^n\right).\end{aligned} \tag{4.39}$$

Estimation (4.39) may look complicated compared to (4.28). It is different by only two factors. The first factor

$$\Big(1-\frac{1}{n!}\Big((M-n+1)\frac{\frac{\rho(\varepsilon)^n}{2}}{1-\frac{\rho(\varepsilon)^n}{2}}\Big)^n\Big)$$

is close to one for $(M-n+1)\frac{\rho(\varepsilon)^n}{2}<1$ and n sufficiently large. The second factor

$$e^{(M-n+1)\left[\frac{\frac{\rho(\varepsilon)^n}{2}}{1-\frac{\rho(\varepsilon)^n}{2}}\right]}$$

is more important. It compensates for the decay of the probability of separation keeping the right hand side of (4.39) close to 1 over large interval of values of M.

Remark 5 *Comparing the performance of single vs two-neuron separability in terms of the probabilities $\mathcal{P}_1(\mathcal{M},n)$ involves taking the maximum of*

$$\mathcal{P}_1(\mathcal{M},n,\varepsilon)=(1-(1-\varepsilon)^n)\Big(1-\frac{\rho(\varepsilon)}{2}\Big)^M, \tag{4.40}$$

and

$$\begin{aligned}\mathcal{P}_1(\mathcal{M},n)\geq\max_{\varepsilon\in(0,1)}(1-(1-\varepsilon)^n)\times\Big(1-\frac{\rho(\varepsilon)^n}{2}\Big)^M e^{(M-n+1)\left[\frac{\frac{\rho(\varepsilon)^n}{2}}{1-\frac{\rho(\varepsilon)^n}{2}}\right]}\times\\ \Big(1-\frac{1}{n!}\Big((M-n+1)\frac{\frac{\rho(\varepsilon)^n}{2}}{1-\frac{\rho(\varepsilon)^n}{2}}\Big)^n\Big)\end{aligned} \tag{4.41}$$

with respect to ε over $(0,1)$. In some situations, when the testing point is already given, the probabilities $\mathcal{P}_1$ are no longer relevant since the value of ε corresponding to the testing point is fixed. In this cases one needs to compare $\mathcal{P}_1(\mathcal{M},n,\varepsilon)$ defined by (4.40) and (4.41) instead. Performance of the corresponding separation schemes are illustrated with Fig. 4.8. Notice, that two-neuron cascade significantly out-performs the single neuron one over the large interval of values of M.

Remark 6 ([31]) *The probability $\mathcal{P}_M(\mathcal{M},n)$ that each point from $\mathcal{M}$ can be separated from other points by two uncorrelated neurons can be estimated like in Remark 4: $\mathcal{P}_M(\mathcal{M},n)\geq 1-M(1-\mathcal{P}_1(\mathcal{M},n))$.*

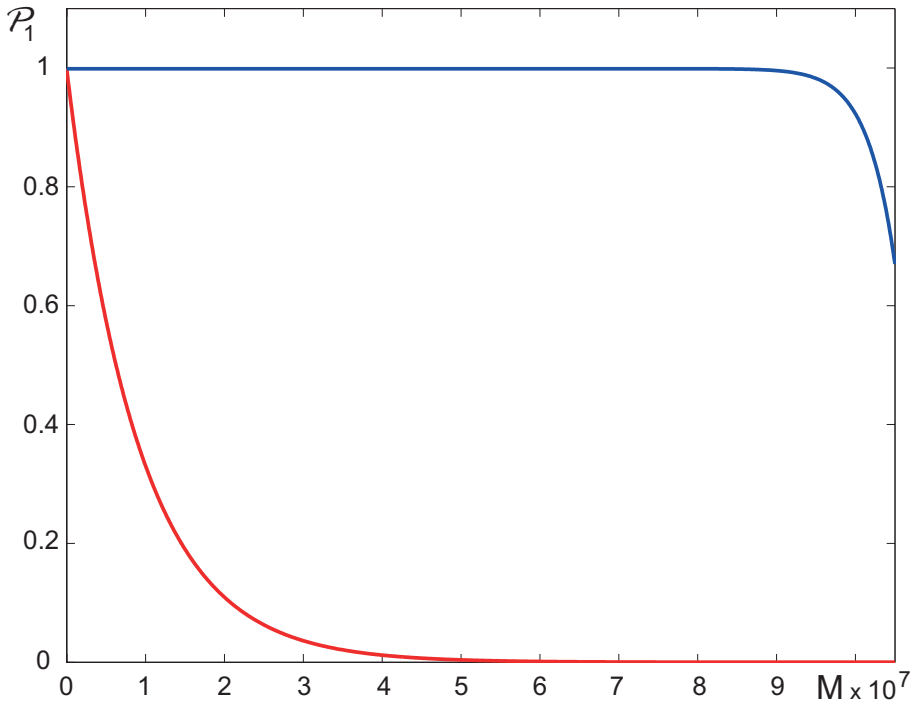


Figure 4.8: Illustration to Remark 5. Blue line shows an estimate of the rhs of (4.41) as a function of M at $\varepsilon = 1/5$, $\rho(\varepsilon) = 3/5$, and $n = 30$. Red line depicts an estimate of the rhs of (4.40) as a function of M for the same values of ε, $\rho(\varepsilon)$, and n.

Equidistribution in an ellipsoid

Let points in the set $\mathcal{M}$ be selected by independent trials taken from the equidistribution in a n-dimensional ellipsoid. Without loss of generality, we present this ellipsoid in the orthonormal eigenbasis.

$$E_n = \{x \in \mathbb{R}^n | \sum_{i=1}^{n} \frac{x_i^2}{c_i^2} \leq 1\} \tag{4.42}$$

where c_i are the semi-principal axes. The linear transformation

$$(x_1, \ldots, x_n) \mapsto \left(\frac{x_1}{c_1}, \ldots, \frac{x_n}{c_n}\right)$$

transforms the ellipsoid into the unit ball. The volume of every set in the new coordinates scales with the factor $1/(\prod_i c_i)$. Therefore the ratio of two volumes does not change, and the equidistribution in the ellipsoid is transformed into the equidistribution in the unit ball, hyperplanes are transformed into hyperplanes and the property of linear separability is not affected by nonsingular linear transformation. Thus the following corollaries hold:

Dimension, n	2	5	10	20	30
Ball (theoretical estimate)	$3.7 \cdot 10^{-5}$	0.0364	0.4096	0.9455	0.9975
Cube					
min:	$4 \cdot 10^{-4}$	0.0089	0.2580	0.9408	0.9986
median:	$4 \cdot 10^{-4}$	0.0110	0.2737	0.9469	0.9992
max:	$5 \cdot 10^{-4}$	0.0137	0.2847	0.9511	1.0
Gaussian					
min:	$5 \cdot 10^{-4}$	0.0122	0.1403	0.7559	0.9792
median:	$10 \cdot 10^{-4}$	0.0153	0.1568	0.7698	0.9817
max:	0.0014	0.0183	0.1778	0.7836	0.9848

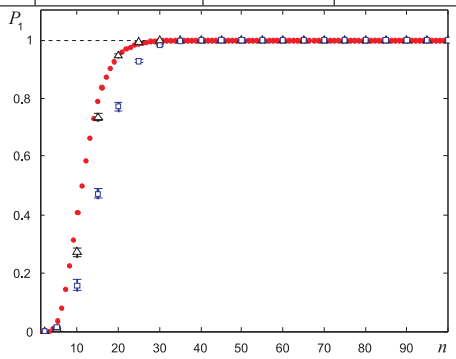


Figure 4.9: Numerical and theoretical estimates of $P_1(\mathcal{M}, n)$ for various distributions and n. *Left panel*, top row: theoretical estimate of $P_1(\mathcal{M}, n)$ for equidistributions in n-balls derived in accordance with (4.28) in the statement of Theorem 3. *Left panel*, rows 2, 3: numerical estimates $F_1(\mathcal{M}, n)$ of $P_1(\mathcal{M}, n)$ for both normal and equidistribution in an n-cube for various values of n. *Right panel:* solid circles show the values of theoretical estimates $P_1(\mathcal{M}, n)$, triangles show empirical means of $F_1(\mathcal{M}, n)$ for the samples drawn from equidistribution in the cube $[-1, 1]^n$, and squares correspond to empirical means of $F_1(\mathcal{M}, n)$ for the samples drown from the Gaussian (normal) distribution. Whiskers in the plots indicate maximal and minimal values in of $F_1(\mathcal{M}, n)$ in each group of experiments.

Corollary 1 ([31]) *Let $\mathcal{M}$ be formed by a finite number of i.i.d. trials taken from the equidistribution in a n-dimensional ellipsoid E_n (4.42), and let x be a test point drawn independently from the same distribution. Then x can be separated from $\mathcal{M}$ be a linear functional with probability $P_1(\mathcal{M}, n)$:*

$$P_1(\mathcal{M}, n) \geq (1 - (1 - \varepsilon)^n)\Big(1 - \frac{\rho(\varepsilon)^n}{2}\Big)^M$$

Corollary 2 ([31]) *Let $\mathcal{M}$ be formed by a finite number of i.i.d. trials taken from the equidistribution in a n-dimensional ellipsoid E_n. With the probability*

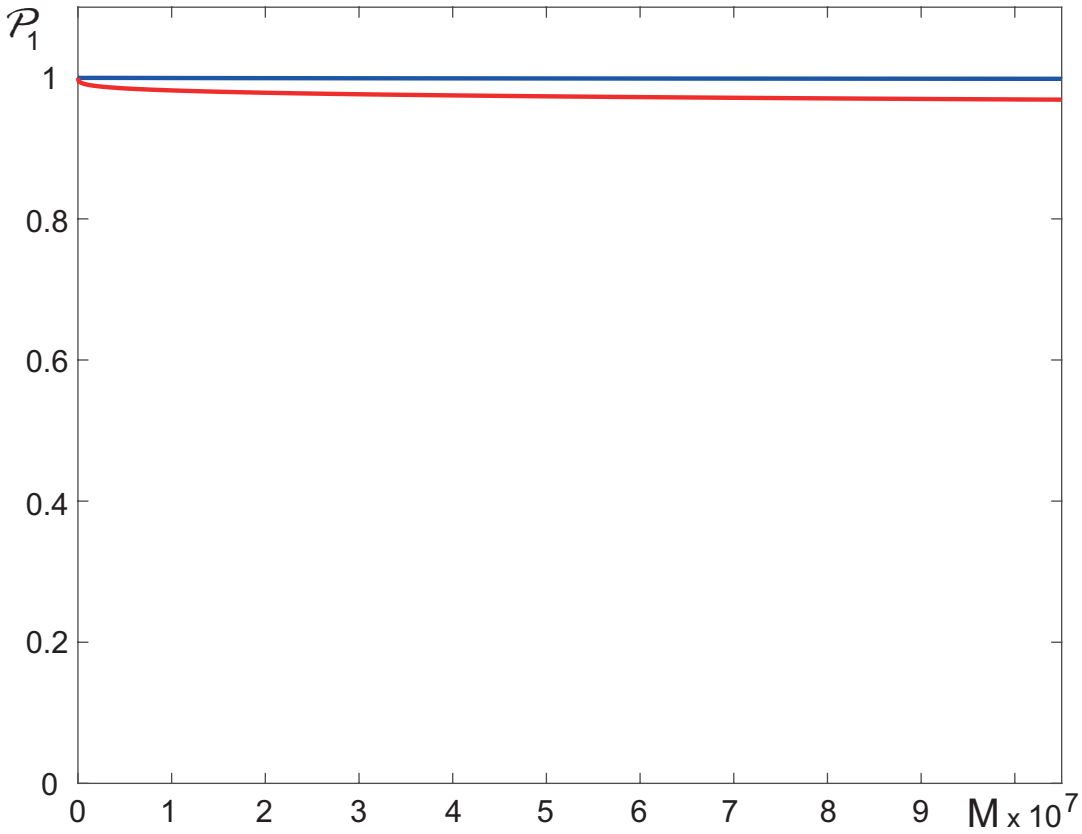


Figure 4.10: Illustration to Remark 5. Blue line shows an estimate of the rhs of (4.41) as a function of M at $\varepsilon = 1/5, \rho(\varepsilon) = 3/5$, and $n = 30$. Red line depicts an estimate of rhs of (4.40) as of function of M for the same values of $\varepsilon, \rho(\varepsilon)$, and n.

$$P_1(\mathcal{M}, n) \geq \Big[(1-(1-\varepsilon)^n)\Big(1-(M-1)\frac{\rho(\varepsilon)^n}{2}\Big)\Big]^M$$

each point y from the random set $\mathcal{M}$ can be separated from $\mathcal{M} \setminus \{y\}$ by a linear functional.

Equidistribution in a cube and normal distributions

It is well known that, for n sufficiently large, samples drawn from an n-dimensional normal distribution [55] concentrate near a corresponding n-sphere. Similarly, samples generated from an equidistribution in an n-dimensional cube concentrate near a corresponding sphere too. In this respect, one might expect that the estimates derived in Theorems 3,4 and 5 hold for these distributions too, asymptotically in n. Numerical experiments below illustrate that it is indeed the case.

The experiments are described as follows. For each distribution (normal distribution in $\mathbb{R}^n$ and equidistribution in the n-cube $[-1,1]^n$) and a given n an i.i.d. sample $\mathcal{M}$ of $M = 10^4$ vectors was drawn. For each vector y in this sample the functional $l_y(x) = \langle y, x\rangle - |y|^2$ was constructed. For each $y \in \mathcal{M}$ and $x \in \mathcal{M}$, $\neq y$ the sign of $l_y(x)$ was evaluated, and the total number N of instances when $l_y(x) < 0$ for all $x \in \mathcal{M}$, $x \neq y$ was calculated. The latter is a lower bound estimate of the number of points in $\mathcal{M}$ that linearly separable from the rest in the sample. This was followed from deriving the values of the success frequencies, $F_1(\mathcal{M}, n) = N/(M-1)$. For each n the experiment is repeated 50 times. Outcomes of this experiment are presented in Fig. 4.9. As we can see from Fig. 4.9 despite that the sample are drawn from different distributions, for $n > 30$ these differences do not significantly affect point separability properties.

One trial non-iterative learning

Basic model of linear separation of a given query point $y \in \mathcal{M}$ from any other $x \in \mathcal{M}$ from any other $x \in \mathcal{M}$, $x \neq y$ is

$$l_y(x) = \Big\langle \frac{y}{|y|, x} - |y| \Big\rangle < 0 \text{ for } x \in \mathcal{M} \setminus \{y\}. \tag{4.43}$$

Deriving these functionals is a genuine one-shot procedure and does not require iterative learning. The construction, however, assumes that the distribution from the sample $\mathcal{M}$ is drawn is close in some sense to an equidistribution in the unit ball $B_n(1)$.

In more general cases, e.g. when sampling from the equidistribution is an ellipsoid, the functionals $l_y(x)$ can be replaced with Fisher linear discriminants:

$$l_y(x) = \Big\langle \frac{\omega(y,\mathcal{M})}{|\omega(y,\mathcal{M})|}, x \Big\rangle - c, \tag{4.44}$$

where

$$\omega(y,\mathcal{M}) = \Sigma^{-1}\Big(y - \frac{1}{M-1}\sum_{x\neq y} x\Big),$$

Σ is the non-singular covariance matrix of the sample $\mathcal{M}$, and c is a parameter. The values of c could be chosen as $c = \Big\langle \frac{\omega(y,\mathcal{M})}{\|\omega(y,\mathcal{M})\|}, y \Big\rangle$. The procedure of generating the separating functional $l_y(x)$ remains non-iterative, but it does require knowledge of the covariance matrix Σ.

Note, that if $\mathcal{M}$ is centered at 0 then $\frac{1}{M-1}\sum_{x\neq y} x \simeq 0$, and (4.44) reduces to (4.43) after the corresponding Mahalanobis transformation: $x \mapsto \Sigma^{-1/2}x$. If the transformation $x \mapsto \Sigma^{-1/2}x$ transforms ellipsoid from which the sample $\mathcal{M}$ is drawn into the unit ball then (4.44) becomes equivalent to (4.43), and separation properties of Fisher discriminants (4.44) follow in the same way as stated in Corollaries 1 and 2.

In addition, or as an alternative, whitening or decorrelation transformations could be applied to $\mathcal{M}$ too. If the covariance matrix Σ is singular or ill-conditioned, projecting the sample $\mathcal{M}$ onto relevant principal components may be required.

4.5 Knowledge transfer between legacy Artificial Intelligence systems

We will now see how theoretical results developed in this chapter so far can be used to answer the fundamental question: how do we transfer knowledge between two AI systems?

4.5.1 K-tuple separation theorems

Let the set

$$\mathcal{M} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$$

be an i.i.d. sample from a distribution in $\mathbb{R}^n$. Pick another set

$$\mathcal{Y} = \{\boldsymbol{x}_{M+1}, \ldots, \boldsymbol{x}_{M+k}\}$$

from the same distribution at random. What is the probability that there is a linear functional separating $\mathcal{Y}$ from $\mathcal{M}$?

Below we provide three k-tuple separation theorems: for an equidistribution in $B_n(1)$ (Theorem 6 and 7) and for a product probability measure with bounded support (Theorem 8). These two special cases cover or, indeed, approximate a broad range of practically relevant situations including e.g. Gaussian distributions (reduce asymptotically to equidistribution in $B_n(1)$ for n large enough) and data vectors in which each attribute is a numerical and independent random variable.

The aim of this section is to show that theoretical results detailed in previous sections can be employed to develop a novel framework for automated, fast, and non-destructive process of knowledge creation in pre-trained AI systems. We will focus here on demonstrating capabilities rather than on creating a full-scale solution. For the purposes of this demonstration, we will assign a dedicated AI system to operate as an expert or an arbiter. The problem is related to the fundamental Big Data scalability challenge described in Chapter 1.

Theorem 6 *Let* $\mathcal{M} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ *and* $\mathcal{Y} = \{\boldsymbol{x}_{M+1}, \ldots, \boldsymbol{x}_{M+k}\}$ *be i.i.d. samples*

from the equidistribution in $B_n(1)$. *Then*

$$\mathcal{P}_1(\mathcal{M},\mathcal{Y}) \geq \max_{\delta,\varepsilon} \left(1-(1-\varepsilon)^n\right)^k \prod_{m=1}^{k-1} \left(1-m\left(1-\delta^2\right)^{\frac{n}{2}}\right) \left(1-\frac{\Delta(\varepsilon,\delta,k)^{\frac{n}{2}}}{2}\right)^M,$$

$$\Delta(\varepsilon,\delta,k) = 1-\left[\frac{(1-\varepsilon)\sqrt{1-(k-1)\delta^2}}{\sqrt{k}} - (k-1)^{\frac{1}{2}}\delta\right]^2.$$

Subject to :

$$\delta,\varepsilon \in (0,1)$$

$$1-(k-1)\delta^2 \geq 0$$

$$(k-1)(1-\delta^2)^{\frac{n}{2}} \leq 1$$

$$\frac{(1-\varepsilon)\sqrt{1-(k-1)\delta^2}}{\sqrt{k}} - (k-1)^{\frac{1}{2}}\delta \geq 0. \tag{4.45}$$

Proof of Theorem 6. Given that elements in the set $\mathcal{Y}$ are independent, the probability p_1 that $\mathcal{Y} \subset B_n(1) \setminus B_n(1-\varepsilon)$ is

$$p_1 = \left(1-(1-\varepsilon)^n\right)^k.$$

Consider an auxiliary set

$$\hat{\mathcal{Y}} = \left\{\hat{\boldsymbol{x}}_i \in \mathbb{R}^n \mid \hat{\boldsymbol{x}}_i = (1-\varepsilon)\frac{\boldsymbol{x}_{M+i}}{\|\boldsymbol{x}_{M+i}\|},\ i=1,\ldots,k\right\}.$$

Vectors $\hat{\boldsymbol{x}}_i \in \hat{\mathcal{Y}}$ belong to the sphere of radius $1-\varepsilon$ centered at the origin (see Figure 4.11, (b)). According to proof of Proposition 3 and estimate (4.14), the probability p_2 that for a given a given $\delta \in (0,1)$ all elements of $\hat{\mathcal{Y}}$ are pairwise $\delta/(1-\varepsilon)$-orthogonal, i.e.

$$|\cos\left(\hat{\boldsymbol{x}}_i,\hat{\boldsymbol{x}}_j\right)\rangle| \leq \frac{\delta}{1-\varepsilon} \text{ for all } i,j \in \{1,\ldots,k\},\ i \neq j, \tag{4.46}$$

can be estimated from below as:

$$p_2 \geq p_1 \prod_{m=1}^{k-1}\left(1-m\left(1-\delta^2\right)^{\frac{n}{2}}\right) = \left(1-(1-\varepsilon)^n\right)^k \prod_{m=1}^{k-1}\left(1-m\left(1-\delta^2\right)^{\frac{n}{2}}\right),$$

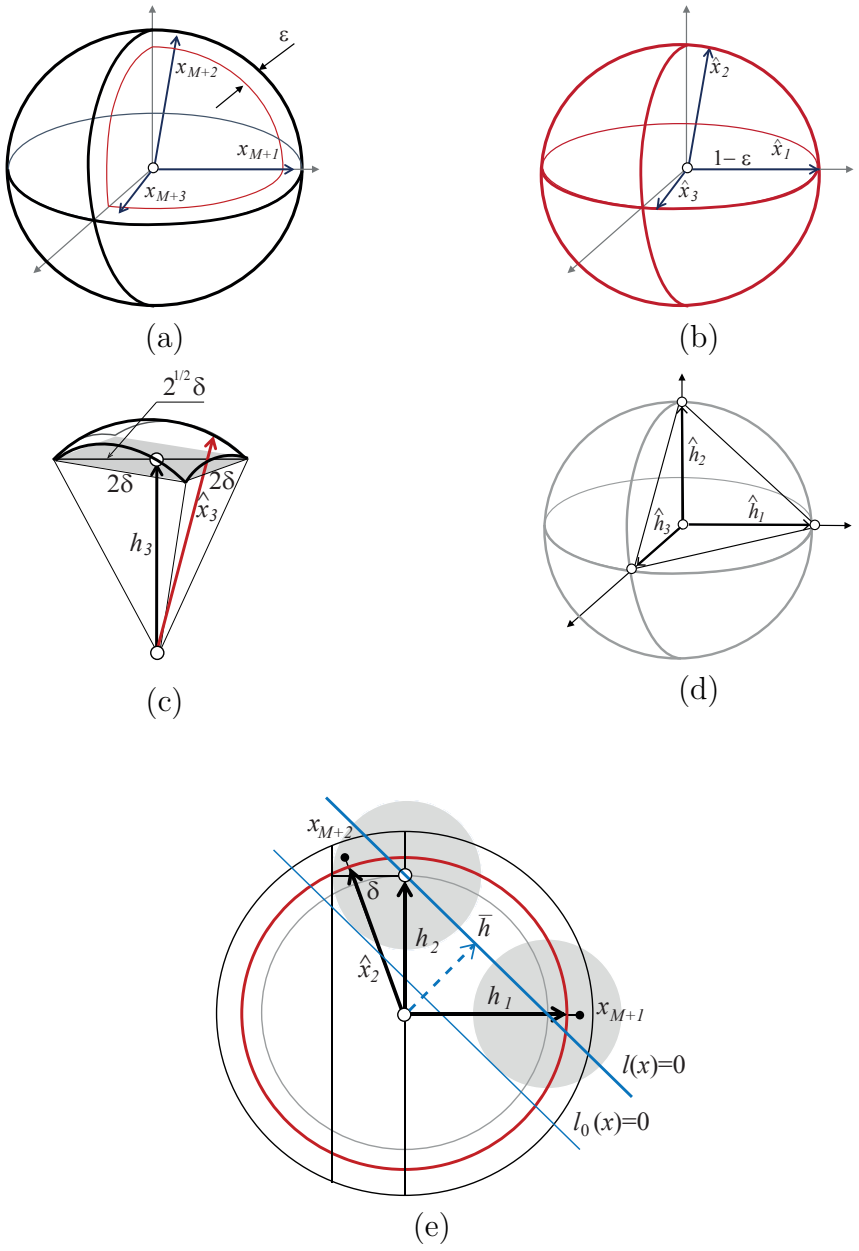


Figure 4.11: Illustration to the proof of Theorem 6. *Panel (a)* shows $\boldsymbol{x}_{M+1}$, $\boldsymbol{x}_{M+2}$ and $\boldsymbol{x}_{M+3}$ in the set $B_n(1) \setminus B_n(1-\varepsilon)$. *Panel (b)* shows $\hat{\boldsymbol{x}}_1$, $\hat{\boldsymbol{x}}_2$, and $\hat{\boldsymbol{x}}_3$ on the sphere $S_{n-1}(1-\varepsilon)$. *Panel (c)*: construction of $\boldsymbol{h}_3$. Note that $\|\boldsymbol{h}_3\| = \|\hat{\boldsymbol{x}}_3\|(1-2\delta^2)^{1/2} = (1-\varepsilon)(1-2\delta^2)^{1/2}$. *Panel (d)* shows simplex formed by orthogonal vectors $\hat{\boldsymbol{h}}_1, \hat{\boldsymbol{h}}_2, \hat{\boldsymbol{h}}_3$. *Panel (e)* illustrates derivation of functionals l and l_0.

for $(k-1)(1-\delta^2)^{\frac{n}{2}} \leq 1$. Suppose now that (4.46) holds true. Let δ be chosen so that $1-(k-1)\delta^2 \geq 0$. If this is the case than there exists a set of k pairwise orthogonal vectors

$$\mathcal{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_k\},\ \langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle = 0,\ i,j \in \{1,\ldots,k\},\ i \neq j,$$

such that (Figure 4.11, (c))

$$\|\hat{\boldsymbol{x}}_i - \boldsymbol{h}_i\| \leq (i-1)^{\frac{1}{2}}\delta,\ \|\boldsymbol{h}_i\| = (1-\varepsilon)(1-(i-1)\delta^2)^{\frac{1}{2}},\ \text{for all } i \in \{1,\ldots,k\}. \quad (4.47)$$

Finally, consider the set

$$\hat{\mathcal{H}} = \left\{ \hat{\boldsymbol{h}}_i \in \mathbb{R}^n \mid \hat{\boldsymbol{h}}_i = (1-\varepsilon)(1-(k-1)\delta^2)^{\frac{1}{2}} \frac{\boldsymbol{h}_i}{\|\boldsymbol{h}_i\|},\ i = 1,\ldots,k \right\}.$$

The set $\hat{\mathcal{H}}$ belongs to the sphere of radius $(1-(k-1)\delta^2)^{\frac{1}{2}}$, and its k elements are vertices of the corresponding $k-1$-simplex in $\mathbb{R}^n$ (Figure 4.11, (d)).

Consider the functional:

$$l(\boldsymbol{x}) = \left\langle \frac{\bar{\boldsymbol{h}}}{\|\bar{\boldsymbol{h}}\|}, \boldsymbol{x} \right\rangle - \frac{(1-\varepsilon)\sqrt{1-(k-1)\delta^2}}{\sqrt{k}},\ \bar{\boldsymbol{h}} = \frac{1}{k}\sum_{i=1}^{k} \hat{\boldsymbol{h}}_i.$$

Recall that if $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k$ are orthonormal vectors in $\mathbb{R}^n$ then $\|\boldsymbol{e}_1 + \boldsymbol{e}_2 + \cdots + \boldsymbol{e}_k\|^2 = k$. Hence $\left\|\sum_{i=1}^{k} \hat{\boldsymbol{h}}_i\right\| = \sqrt{k}(1-\varepsilon)\sqrt{1-(k-1)\delta^2}$, and we can conclude that $l(\hat{\boldsymbol{h}}_i) = 0$ and $l(\boldsymbol{h}_i) \geq 0$ for all $i = 1,\ldots,k$. According to (4.47), $\|\hat{\boldsymbol{x}}_i - \boldsymbol{h}_i\| \leq (k-1)^{\frac{1}{2}}\delta$ for all $i = 1,\ldots,k$. Therefore the functional

$$l_0(\boldsymbol{x}) = l(\boldsymbol{x}) + (k-1)^{\frac{1}{2}}\delta = \left\langle \frac{\bar{\boldsymbol{h}}}{\|\bar{\boldsymbol{h}}\|}, \boldsymbol{x} \right\rangle - \left(\frac{(1-\varepsilon)\sqrt{1-(k-1)\delta^2}}{\sqrt{k}} - (k-1)^{\frac{1}{2}}\delta \right) \quad (4.48)$$

satisfies the following condition: $l_0(\hat{\boldsymbol{x}}_i) \geq 0$ and $l_0(\boldsymbol{x}_{M+i}) \geq 0$ for all $i = 1,\ldots,k$. This is illustrated with Figure 4.11, (e).

The functional l_0 partitions the unit ball $B_n(1)$ into the union of two disjoint sets: the spherical cap $\mathcal{C}$

$$\mathcal{C} = \{\boldsymbol{x} \in B_n(1) \mid l_0(\boldsymbol{x}) \geq 0\} \quad (4.49)$$

and its complement in $B_n(1)$, $B_n(1)\setminus\mathcal{C}$. The volume $\mathcal{V}$ of the cap $\mathcal{C}$ can be estimated from above as

$$\mathcal{V}(\mathcal{C}) \leq \frac{\Delta(\varepsilon,\delta,k)^{\frac{n}{2}}}{2},$$
$$\Delta(\varepsilon,\delta,k) = 1 - \left[\frac{(1-\varepsilon)\sqrt{1-(k-1)\delta^2}}{\sqrt{k}} - (k-1)^{\frac{1}{2}}\delta\right]^2.$$

Hence the probability p_3 that $l_0(\boldsymbol{x}_i) < 0$ for all $\boldsymbol{x}_i \in \mathcal{M}$ can be estimated from below as

$$p_3 \geq \left(1 - \frac{\Delta(\varepsilon,\delta,k)^{\frac{n}{2}}}{2}\right)^M.$$

Therefore, for fixed $\varepsilon, \delta \in (0,1)$ chosen so that $\frac{(1-\varepsilon)\sqrt{1-(k-1)\delta^2}}{\sqrt{k}} - (k-1)^{\frac{1}{2}}\delta \geq 0$, the probability $p_4(\varepsilon,\delta)$ that $\mathcal{M}$ can be separated from $\mathcal{Y}$ by the functional l_0 can be estimated from below as:

$$p_4(\varepsilon,\delta) \geq (1-(1-\varepsilon)^n)^k \prod_{m=1}^{k-1}\left(1 - m\left(1-\delta^2\right)^{\frac{n}{2}}\right)\left(1 - \frac{\Delta(\varepsilon,\delta,k)^{\frac{n}{2}}}{2}\right)^M.$$

Given that this estimate holds for all feasible values of ε, δ, statement (4.45) follows. □

Figure 4.12 shows how estimate (4.45) of the probability $\mathcal{P}_1(\mathcal{M},\mathcal{Y})$ behaves, as a function of $|\mathcal{Y}|$ for fixed M and n. As one can see from this figure, when k exceeds some critical value ($k = 9$ in this specific case), the lower bound estimate (4.45) of the probability $\mathcal{P}_1(\mathcal{M},\mathcal{Y})$ drops. This is not surprising since the bound (4.45) is a) based on rough, L_∞-like, estimates, and b) these estimates are derived for just one class of separating functionals $l_0(\boldsymbol{x})$. Furthermore, no prior pre-processing and/or clustering was assumed for the $\mathcal{Y}$. An alternative estimate that allows us to account for possible clustering in the set $\mathcal{Y}$ is presented in Theorem 7.

Theorem 7 *Let $\mathcal{M} = \{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_M\}$ and $\mathcal{Y} = \{\boldsymbol{x}_{M+1},\ldots,\boldsymbol{x}_{M+k}\}$ be i.i.d. samples from the equidistribution in $B_n(1)$. Let $\mathcal{Y}_c = \{\boldsymbol{x}_{M+r_1},\ldots,\boldsymbol{x}_{M+r_m}\}$ be a subset of m elements from $\mathcal{Y}$ such that*

$$\beta_2(m-1) \leq \sum_{r_j,\ r_j \neq r_i} \langle \boldsymbol{x}_{M+r_i}, \boldsymbol{x}_{M+r_j}\rangle \leq \beta_1(m-1) \text{ for all } i = 1,\ldots,m. \qquad (4.50)$$

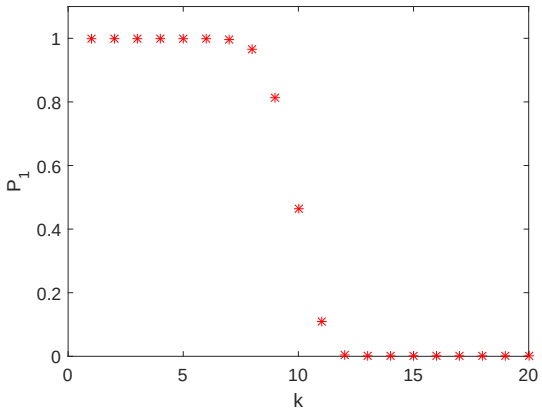


Figure 4.12: Estimate (4.45) of $\mathcal{P}_1(\mathcal{M}, \mathcal{Y})$ as a function of k for $n = 2000$ and $M = 10^5$.

Then

$$\begin{aligned} \mathcal{P}_1(\mathcal{M}, \mathcal{Y}_c) &\geq \max_{\varepsilon}(1-(1-\varepsilon)^n)^k \left(1 - \frac{\Delta(\varepsilon, m)^{\frac{n}{2}}}{2}\right)^M \\ \Delta(\varepsilon, m) &= 1 - \frac{1}{m}\left(\frac{(1-\varepsilon)^2 + \beta_2(m-1)}{\sqrt{1+(m-1)\beta_1}}\right)^2 \\ &\text{Subject to :} \\ &(1-\varepsilon)^2 + \beta_2(m-1) > 0 \\ &1 + (m-1)\beta_1 > 0. \end{aligned} \tag{4.51}$$

Proof of Theorem 7. Consider the set $\mathcal{Y}$. Observe that $\|\boldsymbol{x}_{M_i}\| \geq 1-\varepsilon$, $\varepsilon \in (0,1)$, for all $i = 1, \ldots, k$, with probability $p_1 \geq (1-(1-\varepsilon)^n)^k$. Consider now the vector $\bar{\boldsymbol{y}}$

$$\bar{\boldsymbol{y}} = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}_{M+r_i},$$

and evaluate the following inner products

$$\left\langle \frac{\bar{\boldsymbol{y}}}{\|\bar{\boldsymbol{y}}\|}, \boldsymbol{x}_{M+i} \right\rangle = \frac{1}{m\|\bar{\boldsymbol{y}}\|}\left(\langle \boldsymbol{x}_{M+r_i}, \boldsymbol{x}_{M+r_i} \rangle + \sum_{r_j,\ j\neq i} \langle \boldsymbol{x}_{M+r_i}, \boldsymbol{x}_{M+r_j} \rangle \right),\ i = 1, \ldots, m.$$

According to assumption (4.50),

$$\left\langle \frac{\bar{\boldsymbol{y}}}{\|\bar{\boldsymbol{y}}\|}, \boldsymbol{x}_{M+i} \right\rangle \geq \frac{1}{m\|\bar{\boldsymbol{y}}\|}\left((1-\varepsilon)^2 + \beta_2(m-1)\right)$$

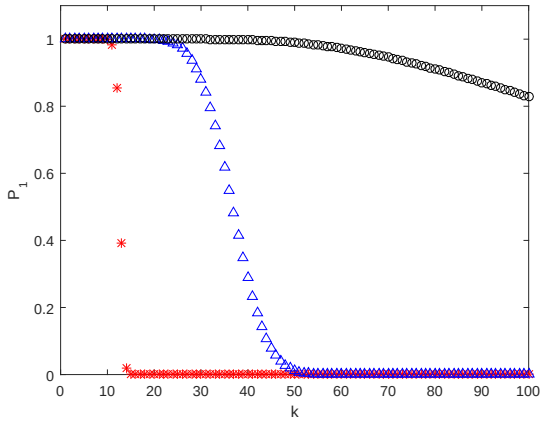


Figure 4.13: Estimate (4.51) of $\mathcal{P}_1(\mathcal{M}, \mathcal{Y})$ as a function of k for $n = 2000$ and $M = 10^5$. Red stars correspond to $\beta_1 = 0.5$, $\beta_2 = 0$. Blue triangles stand for $\beta_1 = 0.5$, $\beta_2 = 0.05$, and black circles stand for $\beta_1 = 0.5$, $\beta_2 = 0.07$.

and, respectively,

$$\frac{1}{m}\left(1 + (m-1)\beta_1\right) \geq \langle \bar{\boldsymbol{y}}, \bar{\boldsymbol{y}} \rangle \geq \frac{1}{m}\left((1-\varepsilon)^2 + \beta_2(m-1)\right)$$

Let $(1-\varepsilon)^2 + \beta_2(m-1) > 0$ and $(1-\varepsilon)^2 + \beta_1(m-1) > 0$. Consider the functional

$$l_0(\boldsymbol{x}) = \left\langle \frac{\bar{\boldsymbol{y}}}{\|\bar{\boldsymbol{y}}\|}, \boldsymbol{x} \right\rangle - \frac{1}{\sqrt{m}} \left(\frac{(1-\varepsilon)^2 + \beta_2(m-1)}{\sqrt{1 + (m-1)\beta_1}} \right). \tag{4.52}$$

It is clear that $l_0(\boldsymbol{x}_{M+r_i}) \geq 0$ for all $i = 1, \ldots, m$ by the way the functional is constructed. The functional $l_0(\boldsymbol{x})$ partitions the ball $B_n(1)$ into two sets: the set $\mathcal{C}$ defined as in (4.49) and its complement, $B_n(1) \setminus \mathcal{C}$. The volume $\mathcal{V}$ of the set $\mathcal{C}$ is bounded from above as

$$\mathcal{V}(\mathcal{C}) \leq \frac{\Delta(\varepsilon, m)^{\frac{n}{2}}}{2}$$

where

$$\Delta(\varepsilon, m) = 1 - \frac{1}{m}\left(\frac{(1-\varepsilon)^2 + \beta_2(m-1)}{\sqrt{1 + \beta_1(m-1)}} \right)^2.$$

Estimate (4.51) now follows. □

Examples of estimates (4.51) for various parameter settings are shown in Fig. 4.13. As one can see, in absence of pairwise strictly positive correlation assumption, $\beta_1 = 0$, the estimate's behavior, as a function of k, is similar to that of (4.45). However, presence of moderate pairwise positive correlation results in significant boosts to the values of $\mathcal{P}_1$.

Remark 7 Estimates (4.45), (4.51) for the probability $P_1(\mathcal{M}, \mathcal{Y})$ that follow from Theorems 6, 7 assume that the underlying probability distribution is an equidistribution in $B_n(1)$. They can, however, be generalized to equidistributions in ellipsoids and Gaussian distributions (cf. [31]).

Note that proofs of Theorems 6, 7 are constructive. Not only they provide estimates from below of the probability that two random i.i.d. drawn samples from $B_n(1)$ are linearly separable, but also they present the corresponding separating functionals explicitly as (4.48) and (4.52), respectively. The latter functionals are similar to Fisher linear discriminants. Whilst having explicit separation functionals is an obvious advantage from practical view point, the estimates that are associated with such functionals do not account for more flexible alternatives. In what follows we present a generalization of the above results that accounts for such a possibility as well as extends applicability of the approach to samples from product distributions. The results are provided in Theorem 8.

Theorem 8 *Consider the linear space* $E = \operatorname{span}\{\boldsymbol{x}_j - \boldsymbol{x}_{M+1} \mid j = M+2, \ldots, M+k\}$, *let the cardinality* $|\mathcal{Y}| = k$ *of the set* $\mathcal{Y}$ *be smaller than* n. *Consider the quotient space* $\mathbb{R}^n/E$. *Let* $Q(\boldsymbol{x})$ *be a representation of* $\boldsymbol{x} \in \mathbb{R}^n$ *in* $\mathbb{R}^n/E$, *and let the coordinates of* $Q(\boldsymbol{x}_i)$, $i = 1, \ldots, M+1$ *be independent random variables i.i.d. sampled from a product distribution in a unit cube with variances* $\sigma_j > \sigma_0 > 0$, $1 \leq j \leq n-k+1$. *Then for*

$$M \leq \frac{\vartheta}{3} \exp\left(\frac{(n-k+1)\sigma_0^4}{2}\right) - 1$$

with probability $p > 1 - \vartheta$ *there is a linear functional separating* $\mathcal{Y}$ *and* $\mathcal{M}$.

Proof of Theorem 8. Observe that, in the quotient space $\mathbb{R}^n/E$, elements of the set

$$\mathcal{Y} = \{\boldsymbol{x}_{M+1}, \boldsymbol{x}_{M+1} + (\boldsymbol{x}_{M+2} - \boldsymbol{x}_{M+1}), \ldots, \boldsymbol{x}_{M+1} + (\boldsymbol{x}_{M+k} - \boldsymbol{x}_{M+1})\}$$

are vectors whose coordinates coincide with that of the quotient representation of $\boldsymbol{x}_{M+1}$. This means that the quotient representation of $\mathcal{Y}$ consists of a single element, $Q(\boldsymbol{x}_{M+1})$. Furthermore, dimension of $\mathbb{R}^n/E$ is $n-k+1$. Let $R_0^2 = \sum_{i=1}^{n-k+1} \sigma_i^2$ and $\bar{Q}(\boldsymbol{x}) = \mathbb{E}(Q(\boldsymbol{x}))$. According to Theorem 2 and Corollary 2 from [32], for $\vartheta \in (0, 1)$

and M satisfying

$$M \leq \frac{\vartheta}{3} \exp\left(\frac{(n-k+1)\sigma_0^4}{2}\right) - 1,$$

with probability $p > 1 - \vartheta$ the following inequalities hold:

$$\frac{1}{2} \leq \frac{\|Q(\boldsymbol{x}_j) - \bar{Q}(\boldsymbol{x})\|^2}{R_0^2} \leq \frac{3}{2}, \left\langle \frac{Q(\boldsymbol{x}_i) - \bar{Q}(\boldsymbol{x})}{R_0}, \frac{Q(\boldsymbol{x}_{M+1}) - \bar{Q}(\boldsymbol{x})}{\|Q(\boldsymbol{x}_{M+1}) - \bar{Q}(\boldsymbol{x})\|} \right\rangle < \frac{1}{\sqrt{2}}$$

for all i, j, $i \neq M+1$. This implies that the functional

$$\ell_0(\boldsymbol{x}) = \left\langle \frac{Q(\boldsymbol{x}) - \bar{Q}(\boldsymbol{x})}{R_0}, \frac{Q(\boldsymbol{x}_{M+1}) - \bar{Q}(\boldsymbol{x})}{\|Q(\boldsymbol{x}_{M+1}) - \bar{Q}(\boldsymbol{x})\|} \right\rangle - \frac{1}{\sqrt{2}}$$

separates $\mathcal{M}$ and $\mathcal{Y}$ with probability $p > 1 - \vartheta$. □

4.5.2 Artificial Intelligence Knowledge Transfer Framework

In this section we show how Theorems 6, 7 and 8 can be applied for developing a novel one-shot AI knowledge transfer framework. We will focus on the case of transfer knowledge between two AI systems, a teacher AI and a student AI, in which input-output behavior of the student AI is evaluated by the teacher AI. In this setting, assignment of AI roles, i.e. student or teaching, is beyond the scope of this manuscript. The roles are supposed to be pre-determined or otherwise chosen arbitrarily.

General setup

Consider two AI systems, a student AI, denoted as AI_s, and a teacher AI, demoted as AI_t. These legacy AI systems process some *input* signals, produce *internal* representations of the input and return some *outputs*. We further assume that some *relevant* information about the input, internal signals, and outputs of AI_s can be combined into a common object, $\boldsymbol{x}$, representing, but not necessarily defining, the *state* of AI_s. The objects $\boldsymbol{x}$ are assumed to be elements of $\mathbb{R}^n$.

Over a period of activity system AI_s generates a set $\mathcal{S}$ of objects $\boldsymbol{x}$. Exact composition of the set $\mathcal{S}$ could depend on a task at hand. For example, if AI_s is an image classifier, we may be interested only in a particular subset of AI_s input-output data related to images of a certain known class. Relevant inputs and outputs of AI_s

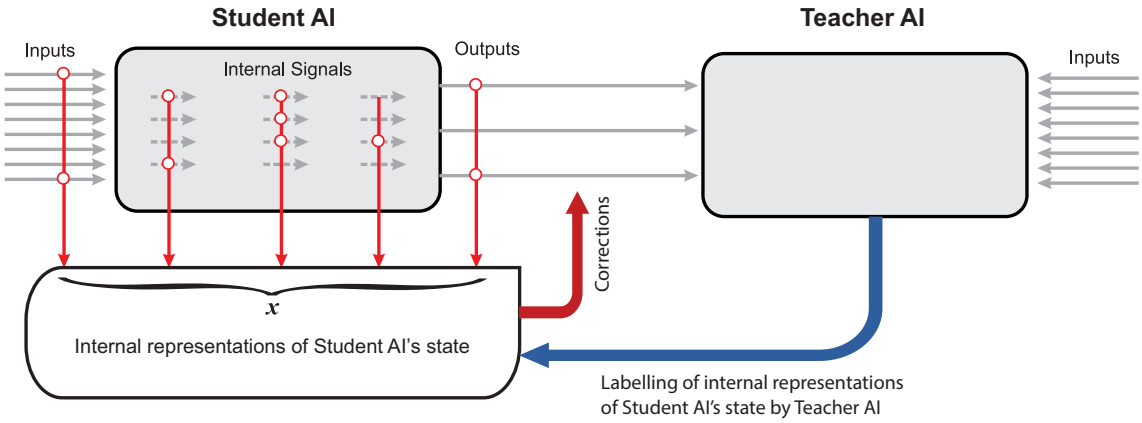


Figure 4.14: AI Knowledge transfer diagram. AI_s produces a set of its state representations, $\mathcal{S}$. The representations are labeled by AI_t into the set of correct responses, $\mathcal{M}$, and the set of errors, $\mathcal{Y}$. The student system, AI_s, is then augmented by an additional "corrector" eliminating these errors.

corresponding to objects in $\mathcal{S}$ are then evaluated by the teacher, AI_t. If AI_s outputs differ to that of AI_t for the same input then an error is registered in the system. Objects $\boldsymbol{x} \in \mathcal{S}$ associated with errors are combined into the set $\mathcal{Y}$. The procedure gives rise to two disjoint sets:

$$\mathcal{M} = \mathcal{S} \setminus \mathcal{Y}, \ \mathcal{M} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$$

and

$$\mathcal{Y} = \{\boldsymbol{x}_{M+1}, \ldots, \boldsymbol{x}_{M+k}\}.$$

A diagram schematically representing the process is shown in Fig. 4.14. The knowledge transfer task is to "teach" AI_s so that with

a) AI_s does not make such errors

b) existing competencies of AI_s on the set of inputs corresponding to internal states $\boldsymbol{x} \in \mathcal{M}$ are retained, and

c) knowledge transfer from AI_t to AI_s is reversible in the sense that AI_s can "unlearn" new knowledge by modifying just a fraction of its parameters, if required.

Two algorithms for achieving such transfer knowledge are provided below.

4.5.3 Knowledge Transfer Algorithms

Our first algorithm, Algorithm 1, considers cases when *Auxiliary Knowledge Transfer Units*, i.e. functional additions to existing student AI$_s$, are single linear functionals. The second algorithm, Algorithm 2, extends Auxiliary Knowledge Transfer Units to two-layer cascades of linear functionals.

The algorithms comprise of two general stages, pre-processing stage and knowledge transfer stage. The purpose of the pre-processing stage is to regularize and "sphere" the data. This operation brings the setup close to the one considered in statements of Theorems 6, 7. The knowledge transfer stage constructs Auxiliary Knowledge Transfer Units in a way that is very similar to the argument presented in the proofs of Theorems 6 and 7. Indeed, if $|\mathcal{Y}_{w,i}| \ll |\mathcal{S}_w \setminus \mathcal{Y}_{w,i}|$ then the term $(\mathrm{Cov}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i}) + \mathrm{Cov}(\mathcal{Y}_{w,i}))^{-1}$ is close identity matrix, and the functionals ℓ_i are good approximations of (4.52). In this setting, one might expect that performance of the knowledge transfer stage would be also closely aligned with the corresponding estimates (4.45), (4.51).

Remark 8 Note that the regularization step in the pre-processing stage ensures that the matrix $\mathrm{Cov}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i}) + \mathrm{Cov}(\mathcal{Y}_{w,i})$ is non-singular. Indeed, consider

$$\begin{aligned}
&\mathrm{Cov}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i}) = \tfrac{1}{|\mathcal{S}_w \setminus \mathcal{Y}_{w,i}|} \textstyle\sum_{\boldsymbol{x} \in \mathcal{S}_w \setminus \mathcal{Y}_{w,i}} (\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i}))(\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i}))^T \\
&= \tfrac{1}{|\mathcal{S}_w \setminus \mathcal{Y}_{w,i}|} \Big(\textstyle\sum_{\boldsymbol{x} \in \mathcal{S}_w \setminus \mathcal{Y}_w} (\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i}))(\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i}))^T + \\
&\textstyle\sum_{\boldsymbol{x} \in \mathcal{Y}_w \setminus \mathcal{Y}_{w,i}} (\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i}))(\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i}))^T \Big) .
\end{aligned}$$

Denoting $d = \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i}) - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_w)$ and rearranging the sum below as

$$\begin{aligned}
&\textstyle\sum_{\boldsymbol{x} \in \mathcal{S}_w \setminus \mathcal{Y}_w} (\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i}))(\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i}))^T = \\
&\textstyle\sum_{\boldsymbol{x} \in \mathcal{S}_w \setminus \mathcal{Y}_w} (\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_w) + d)(\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_w) + d)^T = \\
&\textstyle\sum_{\boldsymbol{x} \in \mathcal{S}_w \setminus \mathcal{Y}_w} (\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_w))(\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_w))^T + \\
&2d \textstyle\sum_{\boldsymbol{x} \in \mathcal{S}_w \setminus \mathcal{Y}_w} (\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_w))^T + |\boldsymbol{x} \in \mathcal{S}_w \setminus \mathcal{Y}_w| dd^T \\
&= \textstyle\sum_{\boldsymbol{x} \in \mathcal{S}_w \setminus \mathcal{Y}_w} (\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_w))(\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_w))^T + |\boldsymbol{x} \in \mathcal{S}_w \setminus \mathcal{Y}_w| dd^T
\end{aligned}$$

we obtain that $\mathrm{Cov}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i})$ is non-singular as long as the sum $\sum_{\boldsymbol{x} \in \mathcal{S}_w \setminus \mathcal{Y}_w} (\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_w))(\boldsymbol{x} - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_w))^T$ is non-singular. The latter property, however, is guaranteed by the regularization step in Algorithm 1.

Remark 9 Clustering at Step 2.a can be achieved by classical k-means algorithms [56] or any other method (see e.g. [26]) that would group elements of $\mathcal{Y}_w$ into clusters according to spatial proximity.

Remark 10 Auxiliary Knowledge Transfer Units in Step 2.b of Algorithm 1 are derived in accordance with standard Fisher linear discriminant formalism. This, however, need not be the case, and other methods such as e.g. Support Vector Machines [85] could be employed for this purpose there. It is worth mentioning, however, that support vector machines might be prone to overfitting [37] and their training often involves iterative procedures such as e.g. *sequential quadratic minimization* [65].

Furthermore, instead of the sets $\mathcal{Y}_{w,i}$, $\mathcal{S}_w \setminus \mathcal{Y}_{w,i}$ one could use a somewhat more aggressive division: $\mathcal{Y}_{w,i}$ and $\mathcal{S}_w \setminus \mathcal{Y}_w$, respectively.

Depending on configuration of samples $\mathcal{S}$ and $\mathcal{Y}$, Algorithm 1 may occasionally create knowledge transfer units, ℓ_i, that are "filtering" errors too aggressively. That is some $\boldsymbol{x} \in \mathcal{S}_w \setminus \mathcal{Y}_w$ may accidentally trigger non-negative response, $\ell_i(\boldsymbol{x}) \geq 0$, and as a result of this their corresponding inputs to A_s could be ignored or mishandled. To mitigate this, one can increase the number of clusters and knowledge transfer units, respectively. This will increase the probability of successful separation and hence alleviate the issue. On the other hand, if increasing the number of knowledge transfer units is not desirable for some reason, then two-functional units could be a feasible remedy. Algorithm 2 presents a procedure for such an improved AI Knowledge Transfer.

Algorithm 1 Single-functional AI Knowledge Transfer

1. **Pre-processing**

   (a) *Centering.* For the given set $\mathcal{S}$, determine the set average, $\bar{\boldsymbol{x}}(\mathcal{S})$, and generate sets $\mathcal{S}_c$

   $$\begin{aligned}\mathcal{S}_c &= \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} = \boldsymbol{\xi} - \bar{\boldsymbol{x}}(\mathcal{S}),\ \boldsymbol{\xi} \in \mathcal{S}\},\\ \mathcal{Y}_c &= \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} = \boldsymbol{\xi} - \bar{\boldsymbol{x}}(\mathcal{S}),\ \boldsymbol{\xi} \in \mathcal{Y}\}.\end{aligned}$$

   (b) *Regularization.* Determine covariance matrices $\mathrm{Cov}(\mathcal{S}_c)$, $\mathrm{Cov}(\mathcal{S}_c \backslash \mathcal{Y}_c)$ of the sets $\mathcal{S}_c$ and $\mathcal{S}_c \setminus \mathcal{Y}_c$. Let $\lambda_i(\mathrm{Cov}(\mathcal{S}_c))$, $\lambda_i(\mathrm{Cov}(\mathcal{S}_c \setminus \mathcal{Y}_c))$ be their corresponding eigenvalues, and $h_1, \ldots, h_n$ be the eigenvectors of $\mathrm{Cov}(\mathcal{S}_c)$. If some of $\lambda_i(\mathrm{Cov}(\mathcal{S}_c))$, $\lambda_i(\mathrm{Cov}(\mathcal{S}_c \setminus \mathcal{Y}_c))$ are zero or if the ratio $\frac{\max_i\{\lambda_i(\Sigma(\mathcal{S}_c))\}}{\min_i\{\lambda_i(\Sigma(\mathcal{S}_c))\}}$ is too large, project $\mathcal{S}_c$ and $\mathcal{Y}_c$ onto appropriately chosen set of $m < n$ eigenvectors, $h_{n-m+1}, \ldots, h_n$:

   $$\begin{aligned}\mathcal{S}_r &= \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} = H^T\boldsymbol{\xi},\ \boldsymbol{\xi} \in \mathcal{S}_c\},\\ \mathcal{Y}_r &= \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} = H^T\boldsymbol{\xi},\ \boldsymbol{\xi} \in \mathcal{Y}_c\},\end{aligned}$$

   where $H = (h_{n-m+1} \cdots h_n)$ is the matrix comprising of m significant principal components of $\mathcal{S}_c$.

   (c) *Whitening.* For the centered and regularized dataset $\mathcal{S}_r$, derive its covariance matrix, $\mathrm{Cov}(\mathcal{S}_r)$, and generate whitened sets

   $$\begin{aligned}\mathcal{S}_w &= \{\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{x} = \mathrm{Cov}(\mathcal{S}_r)^{-\frac{1}{2}}\boldsymbol{\xi},\ \boldsymbol{\xi} \in \mathcal{S}_r\},\\ \mathcal{Y}_w &= \{\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{x} = \mathrm{Cov}(\mathcal{S}_r)^{-\frac{1}{2}}\boldsymbol{\xi},\ \boldsymbol{\xi} \in \mathcal{Y}_r\},\end{aligned}$$

2. **Knowledge transfer**

   (a) *Clustering.* Pick $p \geq 1$, $p \leq k$, $p \in \mathbb{N}$, and partition the set $\mathcal{Y}_w$ into p clusters $\mathcal{Y}_{w,1}, \ldots \mathcal{Y}_{w,p}$ so that elements of these clusters are, on average, pairwise positively correlated. That is there are $\beta_1 \geq \beta_2 > 0$ such that:

   $$\beta_2(|\mathcal{Y}_{w,i}| - 1) \leq \sum_{\xi \in \mathcal{Y}_{w,i} \backslash \{\boldsymbol{x}\}} \langle \boldsymbol{\xi}, \boldsymbol{x} \rangle \leq \beta_1(|\mathcal{Y}_{w,i}| - 1) \text{ for any } \boldsymbol{x} \in \mathcal{Y}_{w,i}$$

   (b) *Construction of Auxiliary Knowledge Units.* For each cluster $\mathcal{Y}_{w,i}$, $i = 1, \ldots, p$, construct separating linear functionals ℓ_i:

   $$\begin{aligned}\ell_i(\boldsymbol{x}) &= \left\langle \tfrac{\boldsymbol{w}_i}{\|\boldsymbol{w}_i\|}, \boldsymbol{x} \right\rangle - c_i,\\ \boldsymbol{w}_i &= \left(\mathrm{Cov}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i}) + \mathrm{Cov}(\mathcal{Y}_{w,i})\right)^{-1} \left(\bar{\boldsymbol{x}}(\mathcal{Y}_{w,i}) - \bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i})\right)\end{aligned}$$

   where $\bar{\boldsymbol{x}}(\mathcal{Y}_{w,i})$, $\bar{\boldsymbol{x}}(\mathcal{S}_w \setminus \mathcal{Y}_{w,i})$ are the averages of $\mathcal{Y}_{w,i}$ and $\mathcal{S}_w \setminus \mathcal{Y}_{w,i}$, respectively, and c_i is chosen as $c_i = \min_{\boldsymbol{\xi} \in \mathcal{Y}_{w,i}} \left\langle \tfrac{\boldsymbol{w}_i}{\|\boldsymbol{w}_i\|}, \boldsymbol{\xi} \right\rangle$.

   (c) *Integration.* Integrate Auxiliary Knowledge Units into decision-making pathways of AI_s. If, for an $\boldsymbol{x}$ generated by an input to AI_s, any of $\ell_i(\boldsymbol{x}) \geq 0$ then report $\boldsymbol{x}$ accordingly (swap labels, report as an error etc.)

In Chapter 5 we illustrate the approach as well as the application of the proposed Knowledge Transfer algorithms in a relevant problem of a computer vision system design for pedestrian detection in live video streams.

Algorithm 2 Two-functional AI Knowledge Transfer

1. **Pre-processing**. Do as in Step 1 in Algorithm 1

2. **Knowledge Transfer**

   (a) *Clustering*. Do as in Step 2.a in Algorithm 1

   (b) *Construction of Auxiliary Knowledge Units*.

   1: Do as in Step 2.b in Algorithm 1. At the end of this step *first-stage* functionals ℓ_i, $i = 1, \ldots, p$ will be derived.

   2: For each set $\mathcal{Y}_{w,i}$, $i = 1, \ldots, p$, evaluate the functionals ℓ_i for all $\boldsymbol{x} \in \mathcal{S}_w \setminus \mathcal{Y}_{w,i}$ and identify elements $\boldsymbol{x}$ such that $\ell_i(\boldsymbol{x}) \geq 0$ and $\boldsymbol{x} \in \mathcal{S}_w \setminus \mathcal{Y}_w$ (incorrect error assignment). Let $\mathcal{Y}_{e,i}$ be the set containing such elements $\boldsymbol{x}$.

   3: **If** (there is an $i \in \{1, \ldots, p\}$ such that $|\mathcal{Y}_{e,i}| + |\mathcal{Y}_{w,i}| > m$) **then** increment the value of p: $p \leftarrow p + 1$, and return to Step 2.a.

   4: **If** (all sets $\mathcal{Y}_{e,i}$ are empty) **then** proceed to Step 2.c.

   5: For each pair of ℓ_i and $\mathcal{Y}_{w,i} \cup \mathcal{Y}_{e,i}$ with $\mathcal{Y}_{e,i}$ not empty, project orthogonally sets $\mathcal{Y}_{w,i}$ and $\mathcal{Y}_{e,i}$ onto the hyperplane $\ell_i(\boldsymbol{x}) = 0$ and form the sets $\mathcal{L}_i(\mathcal{Y}_{w,i})$ and $\mathcal{L}_i(\mathcal{Y}_{e,i})$:

$$\begin{aligned}
\mathcal{L}_i(\mathcal{Y}_{w,i}) &= \left\{ \boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{x} = \left(I_m - \tfrac{\boldsymbol{w}_i \boldsymbol{w}_i^T}{\|\boldsymbol{w}_i\|^2} \right) \boldsymbol{\xi} + \tfrac{c_i \boldsymbol{w}_i}{\|\boldsymbol{w}_i\|},\ \boldsymbol{\xi} \in \mathcal{Y}_{w,i} \right\}, \\
\mathcal{L}_i(\mathcal{Y}_{e,i}) &= \left\{ \boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{x} = \left(I_m - \tfrac{\boldsymbol{w}_i \boldsymbol{w}_i^T}{\|\boldsymbol{w}_i\|^2} \right) \boldsymbol{\xi} + \tfrac{c_i \boldsymbol{w}_i}{\|\boldsymbol{w}_i\|},\ \boldsymbol{\xi} \in \mathcal{Y}_{e,i} \right\}.
\end{aligned}$$

   6: Construct a linear functional $\ell_{2,i}$ separating $\mathcal{L}_i(\mathcal{Y}_{w,i})$ from $\mathcal{L}_i(\mathcal{Y}_{e,i})$ so that $\ell_{2,i}(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x} \in \mathcal{Y}_{w,i}$ and $\ell_{2,i}(\boldsymbol{x}) < 0$ for all $\boldsymbol{x} \in \mathcal{Y}_{e,i}$.

   (c) *Integration*. Integrate Auxiliary Knowledge Units into decision-making pathways of AI_s. If, for an $\boldsymbol{x}$ generated by an input to AI_s, any of the predicates $(\ell_i(\boldsymbol{x}) \geq 0) \wedge (\ell_{2,i}(\boldsymbol{x}) \geq 0)$ hold true then report $\boldsymbol{x}$ accordingly (swap labels, report as an error etc.).

Chapter 5

Case studies

In this chapter we look at several case studies and experiments that were done during the evaluation of ideas described in this thesis. In the Section 5.1 and 5.2 we will see how a complicated task of visual scene recognition can be mapped into a lower dimensional space and how one solves these tasks by employing simplest classification techniques.

In Section 5.3 we illustrate how ideas from Chapter 4 can be used to transfer knowledge between two or more AI systems.

5.1 Low dimensional data - simple classifiers

In this section we demonstrate how a visual scene recognition problem can be solved with simple techniques. To illustrate this idea we decided to use a binary decision tree as a classifier. The binary decision tree, being one of the simplest technique for understanding, remains very popular nowadays and finds a lot of applications in computer science domains. Among the main advantages of using binary decision trees we can name the following:

1. Binary trees are easily interpreted. Having a decision tree at hand one can easily follow the process of decision making with the full understanding of which variables were important for decision making.

2. The process of decision trees building is fast.

3. Ease of implementation. The process of binary decision tree implementation is fairly simple and there are not many places where errors can be made.

4. Ability to model non-linearities. Even though decision tree are not unique in being able to model non-linearities, still, going back to 1), these non-linearities can be explained in a natural language.

5. The decision process is fast and cheap. Indeed, each node of the decision tree contains only a binary function that usually just compares two numbers.

6. Some implementations easily handle irrelevant attributes. We will cover this later on.

7. Often one does not need to worry about data normalization.

Even though the list of advantages is quite extensive there are some disadvantages in this approach:

1. Ease of overfitting. One can easily overfit the model to noise. This happens due to the nature of the classifier building process. As we shall see later, it is always possible to fit a classifier to any data at hand and get 100% at test time.

2. On the other hand, it is still very easy to underfit your data. These two disadvantages originate from the fact that you are fully responsible for making early enough stop in the process of classifier building.

3. Quite often decision trees are extremely sensitive to data. Slight changes in the dataset can lead to a significant changes in decision tree structure.

4. This kind of algorithm is greedy by nature. To find the best sub-optimal solution you have to apply full search techniques and check the whole grid of points distributed across the data space.

The idea behind the algorithm is very simple yet very powerful: we are querying the data. At a time we ask one question that allows us to extract maximum information needed to make a decision. We now give a more formal definition for

such things as "question","maximum information" and "decision". Suppose we have data in the following form:

$$(x, c) = (x_1, x_2, \ldots, x_n, c) \tag{5.1}$$

where x_n are attributes and c is a target variable. Target values contains task-specific labels for instances: in medicine it can be positive of negative result of medical test; in finance it can be binary prediction for stock of whether it is going to rise or fall; in computer vision the task might be to tell whether or not the object of interest is in the frame. To be more specific let us take one practical task from the computer vision domain and set up the task in the following form: on the given picture decided into several dozens of regions mark those which contain patches of grass or sky. This task arises in the domain of image processing in digital cameras when one need to transform raw sensor signals to a RGB image. This task has a lot of stages and one of these stages is the one called Auto White Balance (AWB) setup. Consider a situation when someone takes pictures of landscapes in strong light on a sunny day in the middle of summer. Frames containing large patches of green grass or blue sky or both are not unusual in these circumstances. Every grass stem reflects sun rays, and since the relative amount of grass is large, such reflection gives rise to an additional source of light in the scene. This light, however, can easily be confused with reflection from a gray object under artificial light in standard (R/G, B/G) color metrics. Such confusion decreases reliability and correctness of Automatic White Balance (AWB) decisions in these situations. Furthermore, it suggests that using mere raw (R/G, B/G) data are hardly sufficient for producing correct AWB decisions. A similar scenario occurs in the presence of large areas of blue sky in images. Even though true color representation may be distorted in such images it is nevertheless desirable to be able to correctly represent true colors of objects in these scenes using AWB settings. For the time being we restrict ourselves only with grass detection. Them, recalling that we have target variable c, we say that for this task c to be in one of $\{grassy; non-grassy\}$. Each node in the binary tree contains some attribute name and some threshold value that attribute is be tested against. The tree is built by splitting the initial training set into subsets according to the attribute name and value. This process continues recursively until some stopping

criteria is reached. After this each area is classified as follows:

$$c(area) = \begin{cases} \text{grassy} & \text{if } \frac{N_g}{N_g+N_{ng}} > c_0 \\ \text{non-grassy} & \text{otherwise,} \end{cases}$$

where N_g is the number of grassy points in the subset and N_{ng} is the number of non-grassy points in the subset and c_0 is some threshold value. Now we shall discuss the criteria we use for splitting sets into subsets. There are several metrics available nowadays. Among the most populars are so called $ID3$ [67] and $C4.5$ [69]. For simplicity of implementation we will choose $ID3$.

The process of constructing decision tree with the $ID3$ algorithm can be described as follows. Let $x = (x_1, x2, \ldots, x_n)$ be the vector of attributes taking values in $\mathbb{R}$, $(x_{1,i}, x_{2,i}, \ldots, x_{n,i})$ be the training set, S be the initial training set which is to be recursively split as the algorithm proceeds. Also, for each attribute we introduce the set of thresholds $\{t_{j,1}, t_{j,2}, \ldots, t_{m,1}\}$ that are equally spaced in the interval $[\min x_j \max x_j]$. With each threshold we associate two sets: $S^+_{j,k} = \{x \in S | x_j > t_{j,k}\}$ and $S^-_{j,k} = \{x \in S | x_j < t_{j,k}\}$. Obviously, that $S = S^+_{j,k} \cup S^-_{j,k}$ and in this sense thresholds $t_{j,k}$ split the initial training set S into to disjoint subsets with respect to attribute value x_j and a set of thresholds associated with this attribute.

For the sets S, $S^+_{j,k}$ and $S^-_{j,k}$ we should introduce the measure that reflect how much variability is there in the subsets. For this purpose we will use Shannon entropy [77]:

$$\begin{aligned} H(S) &= -\textstyle\sum_{c \in C} p(c, S) \log_2 p(c, S), \\ H(S^+_{j,k}) &= -\textstyle\sum_{c \in C} p(c, S^+_{j,k}) \log_2 p(c, S^+_{j,k}), \\ H(S^-_{j,k}) &= -\textstyle\sum_{c \in C} p(c, S^-_{j,k}) \log_2 p(c, S^-_{j,k}). \end{aligned}$$

Obviously, when this measure is equal to zero then the subset contains object of only one class. Also we specify the conditional entropy $H(S|t_{j,k})$ in the following way:

$$H(S|t_{j,k}) = \frac{|S^+_{j,k}|}{|S|} H(S^+_{j,k}) + \frac{|S^-_{j,k}|}{|S|} H(S^-_{j,k}) \tag{5.2}$$

and Relative Information Gain $RIG(S|t_{j,k})$:

$$RIG(S|t_{j,k}) = \frac{H(S) - H(S|t_{j,k})}{H(S)} \tag{5.3}$$

Algorithm 1 *The algorithm for constructing binary decision trees can now be described as follows:*

1. *Consider initial set S;*
2. *Create a set of thresholds $\{t_{j,k}\}$;*
3. *For every $t_{j,k}$ calculate $RIG(S|t_{j,k})$;*
4. *Create a node with attribute x_l being a decision variable, and $x_l < t_{l,m}$, $x_l \geq t_{l,m}$ being its corresponding branching conditions; split the initial set S into two sets $S^+_{l,m}$ and $S^-_{l,m}$*
5. *Remove $t_{l,m}$ from the list of thresholds and repeat this procedure recursively for each subsequent subsets $S^+_{l,m}$ and $S^-_{l,m}$ until a stopping condition is met*

$$\begin{aligned} S^+(w) &= \{x \in S | \alpha(w,x) > 0\}, \\ S^-(w) &= \{x \in S | \alpha(w,x) \leq 0\}, \end{aligned} \tag{5.4}$$

where the function α

$$\alpha(w,x) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n, \tag{5.5}$$

$$\alpha : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}. \tag{5.6}$$

As a simple illustration we consider how algorithm works for a synthetic dataset that has only two attributes: x_1 and x_2. Let the dataset be the one that is depicted on the Figure 5.1. As we can see, dataset consists of points of two classes, marked with blue and green rectangles on the figure. The initial set S is the entire white rectangle containing all the points. Following step 2) of the Algorithm 1 we create a set of 10 thresholds $t_{1,j}$ and $t_{2,j}$ for attributes x_1 and x_2 which values lie in in the region $[0, 40]$. Suppose that maximum RIG value is achieved at $t_{1,14}$ (red line on Figure 5.1). This completes step 4. Proceeding to step 5, we split initial set S into $S^+_{1,14}$(on the right from the red line) and $S^-_{1,14}$(on the left from the red line). These steps are repeated until some stopping condition in each branch is met.

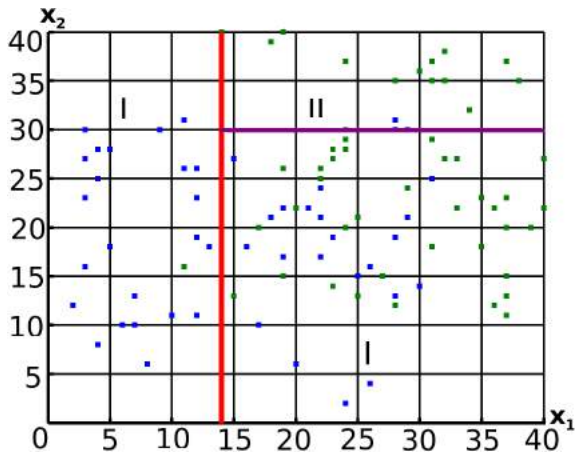


Figure 5.1: Points that belong to different classes are marked with crosses and circles respectively. The solid curve is the conditional entropy, $H(S|t_j)$ plotted as a function of the threshold values, t_j.

Considering this fairly simple synthetic dataset gives us initial intuition about the algorithm routines. Let us consider a real life dataset for the problem of AWB described above. For this problem dataset S consists of 6 variables. This variables are 1) average R/G value, 2) average B/G value, 3) variance of R/G value, 4) variance of B/G value, 5) Illumination of scene (lux value) 6) Intensity variation. The overall number of training points available was about 22000.

Some examples with results of algorithm can be seen on Figure 5.2.

As it was mentioned before, binary decision trees use one variable at a time, sometimes effectively eliminating useless parameters. Therefore one can look at the tree and to some extent estimate the usefulness of each parameter in the training set by simply counting the number of occurrences for each of them. As we see from the figure, some parameters seem to be less useful than others. For example such parameters as RGV and BGV are used less then color and texture characteristics. Recall that we use RIG value which is essentially telling us how much information we get if we know the value of this parameter alone. At this point natural question arises: could this be that this parameter is useless when used alone in the node of the tree? Could this be that combining parameter with any other increase its informativeness? We now proceed to a next part of this work which is aiming to investigate this question.

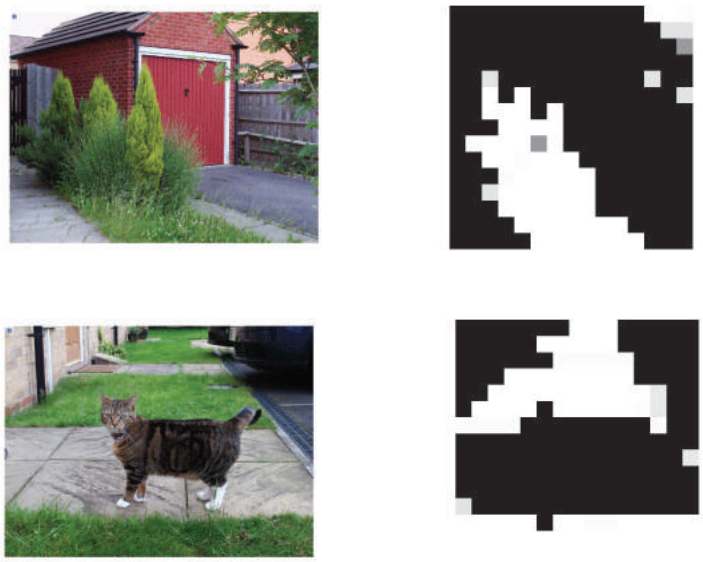


Figure 5.2: Examples of grass detection. Left column: source images. Right column: probability/likelihood maps.

5.2 Low dimensional data - more advanced classifier

As it was discussed early on, in classic binary tree approaches only one parameter is used at a time. This may lead to the situation when some parameters are ignored or almost not being used. What if we try to use more than one parameter at a time? The most obvious way to combine two parameters is to take its linear combination, then test this combination against some threshold value. Considering linear combination of only two parameters at a time is an equivalent to a projection any given dataset into $2D$ space. After this we consider the following set of linear combinations:

$$V = \omega_1 x_i + \omega_2 x_j \tag{5.7}$$

where ω_1 and ω_2 are coefficients, x_i and x_j, $i \neq j$, are two features taken from the dataset. By doing this we try to find the best projection that maximizes RIG. As for the x_i and x_j, we take all possible $2D$ projection of the feature space. This strategy does some improvement on the given training set. These steps can be summarized into 2.

Algorithm 2 *The algorithm for constructing binary decision trees can now be described as follows:*

1. *Consider initial set* S
2. *Consider all possible* $2D$ *projections of* S
3. *Create a set of thresholds* $\{t_{j,k}\}$
4. *For every* $t_{j,k}$ *calculate* $RIG(S|t_{j,k})$
5. *Determine* $t_{l,m} = \arg\max_{j=1,\ldots,n;k=1,\ldots,M} RIG(S|t_{j,k})$ *;*
6. *Create a node with attribute* x_l *being a decision variable, and* $x_l < t_{l,m}$, $x_l \geq t_{l,m}$ *being its corresponding branching conditions; split the initial set* S *into two sets* $S^+_{l,m}$ *and* $S^-_{l,m}$*;*
7. *Remove* $t_{l,m}$ *from the list of thresholds and repeat this procedure recursively for each subsequent subsets* $S^+_{l,m}$ *and* $S^-_{l,m}$ *until a stopping condition is met;*

Examples of improvements that were achieved with this approach can be found in Figure 5.3.

Low dimensional data - binary trees in the original ND space with modified entropy criterion

So far we have seen how simple classifiers can be applied for solving low dimensional classification problems. However, even looking at this kind of classifiers we could name several disadvantages of those approaches. First is the minor one: this type of classifiers work best for cases when data are presented as a set of categorical features, which is not the case in many applications. The second disadvantage is more important: training and testing performance are very sensitive to the training data. Let us look at the example on the picture below. On the Fig. 5.4 we can see a very narrow drop marked with an arrow. Initially it may look as a very attractive point to apply a split procedure. However, there are two possible reasons for this drop. The first one is that there is a gap between data points which is good for us, because we can get a lot of discrimination between data points by splitting the set at this point. However, this drop can also be caused by a data sensitivity. It

Figure 5.3: Examples of grass detection. White patches in the second and third columns stand for high grass probability in the current zone. First column from the left : source images. Centre column: probability/likelihood maps of grass obtained with $ID3$ algorithm. Third column: probability/likelihood maps of grass obtained with using of binary trees with combined attributes in nodes. We may see from the first row of this figure that the number of Positive-Negatives detections was reduced. From the second row we may conclude that the number of Negative-Positive detections has also been reduced. Moreover, we registered these improvement for nearly 75% of all images in our testing set.

is possible that the drop can be eliminated by just slightly varying dataset. In this case splitting dataset in this point leads us to overfitting problem. Another problem that arises later, when we try to apply this classifiers in high dimensional spaces is a computational one. If we try to use this classifier in a high dimensional space we may end up overfitting our model again. However, if we try to use projections, we'll quickly be overwhelmed by the number of combinations we need to check.

The next problem is the discontinuity of goodness measure. This is partly related to the overfitting problem described before. Discontinuity itself does not allow us to use optimization methods for finding local minima. This makes it impossible to move away from greedy algorithms and necessity to check all points in order to get suboptimal solution.

Let us consider how we can avoid partly this kind of issues. For the sake of simplicity let us return to a univariate case. With each point of $i = (x_i, c_i) = (x_{1,i}, x_{2,i}, ..., x_{n,i}, c_i)$ of the original data we associate an auxiliary integrable and non-negative smearing function $f_{\chi_i} : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$. Speaking about candidate functions we

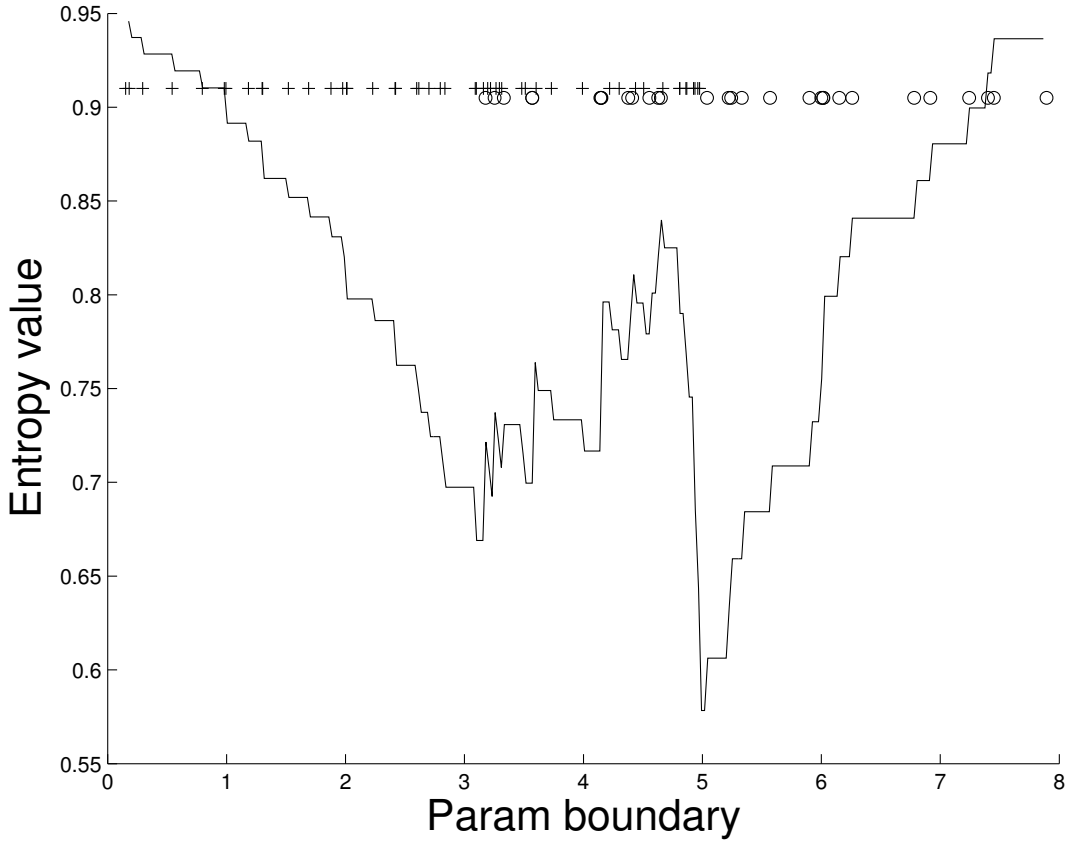


Figure 5.4: Points that belong to different classes are marked with crosses and circles respectively. Solid curve is the conditional entropy, $H(S|t_j)$ plotted as a function of the threshold values, t_j.

could use:

$$\begin{aligned}
\text{Gaussian}: \quad & \frac{1}{\sqrt{(2\pi)^n|\mathbf{\Sigma}|}} e^{\left(-\frac{1}{2}(x-\mathbf{x}_i)^T\mathbf{\Sigma}^{-1}(x-\mathbf{x}_i)\right)}, \\
\text{Inverse multiquadric}: \quad & \frac{1}{\sqrt{1+(\mathbf{x}-\mathbf{x}_i)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{x}_i)}} \\
\text{Delta}-\text{function}: \quad & \delta(x_1-x_1^i)\delta(x_2-x_2^i)\cdots\delta(x_n-x_n^i),
\end{aligned}$$

where $\mathbf{\Sigma}$ is a positive definite symmetric matrix, and $|\mathbf{\Sigma}|$ is the determinant of $\mathbf{\Sigma}$.

Having defined f_{χ_i} we introduce

$$\begin{aligned}
D(S) &= \textstyle\int_S \sum_{i=1}^N f_{\chi_i}(x)dx \\
D_c(S) &= \textstyle\int_S \sum_{i=1}^N f_{\chi_i}(x) I_c(\chi_i)dx,
\end{aligned}$$

where $I_c(\chi_i)$ is the indicator function:

$$I_c(\chi_i) = \begin{cases} 1, & c = c_i, \\ 0, & c \neq c_i. \end{cases}$$

Finally we define $p_f(c, S)$

$$p_f(c,S) = \frac{D_c(S)}{D(S)},$$

$$\begin{aligned}
H_f(S) &= -\textstyle\sum_{c\in C} p_f(c,S)\log_2 p_f(c,S), \\
H_f(S|t_{j,k}) &= \frac{D(S^+(t_{j,k}))}{D(S)} H_f(S^+(t_{j,k})) \\
&\quad + \frac{D(S^-(t_{j,k}))}{D(S)} H_f(S^-(t_{j,k}))
\end{aligned}$$

and

$$RIG_f(S|t_{j,k}) = (H_f(S) - H_f(S|t_{j,k}))/H_f(S). \tag{5.8}$$

Note that $D(S), D_c(S), p_f(c,S) \geq 0$, $\sum_c p_f(c,S) = 1$, $D(S^+(t_{j,k}))+D(S^-(t_{j,k})) = D(S)$. Replacing RIG with RIG_f in Algorithm 1 gives rise to the proposed modification.

The following characterizations of the newly introduced $RIG_f(S|\cdot)$ are immediate

Proposition 4

P1) Let $f_{\chi_i}(\cdot)$ be piece-wise continuous for all $i \in \{1, \ldots, N\}$, then $RIG_f(S|\cdot)$ is continuous. If f_{χ_i} are continuous then $RIG_f(S|\cdot)$ is differentiable.

P2) Let $f_{\chi_i}(\cdot)$ be the delta-function: $f_{\chi_i}(x) = \delta(x_1 - x_1^i)\delta(x_2 - x_2^i)\cdots\delta(x_n - x_n^i)$, then

$$RIG(S|t_{j,k}) = RIG_f(S|t_{j,k}) \text{ for all } t_{j,k}.$$

P3) Let $f_{\chi_i}(\cdot)$ be the indicator-function 1_S, then

$$RIG_f(S|t_{j,k}) = \text{const for all } t_{j,k}.$$

Properties P2), P3) are straightforward. Property P1 follows from that piece-wise continuity (continuity) implies that $p_f(c, S^+(t_{j,k}))$, $p_f(c, S^-(t_{j,k}))$, $D(S^+(t_{j,k}))$, and $D(S^-(t_{j,k}))$ are continuous (differentiable). Hence so are the functions $H_f(S|t_{j,k})$ (with respect to $t_{j,k}$) and, consequently, $RIG_f(S|t_{j,k})$.

According to the Proposition, using "broad" identical f_{χ_i} flattens the shape of $RIG_f(S|\cdot)$, $RIG_f(S|\cdot)$ with f_{χ_i} concentrated at χ_i resembles (in the limit) the shape of $RIG(S|\cdot)$. Figure 5.5 shows how $H_f(S|\cdot)$, derived for f_{χ_i} Gaussian, compares to $H(S|\cdot)$ for a randomly drawn data sample S (represented by $o, +$ in the figure). The function $RIG_f(S|\cdot)$, obviously, is just a scaled and translated version of $H_f(S|\cdot)$. Note that $H_f(S|\cdot)$ is a quite smooth curve, which agrees with property P1 in Proposition 4. Hence one can use a range of standard optimization methods to infer the optimal values of $t_{j,k}$. It is also clear that $H_f(S|\cdot)$ (and $RIG_f(S|\cdot)$) may still have a number of local minima. These can, however, be addressed by starting optimization procedures from various initial conditions. While this approach may increase the total number of calculations in total, it is generally more advantageous than direct search especially when nominal dimensionality of the data is high.

As we've seen this introducing smearing function allows us to reduce the number of local minima and therefore makes it possible to apply optimization techniques to decision trees inducers. It is also possible to use all features is decision trees node when making a decision, which is also supposed to increase the robustness of the classifier since the decision made in tree nodes are less sensitive to data points removal.

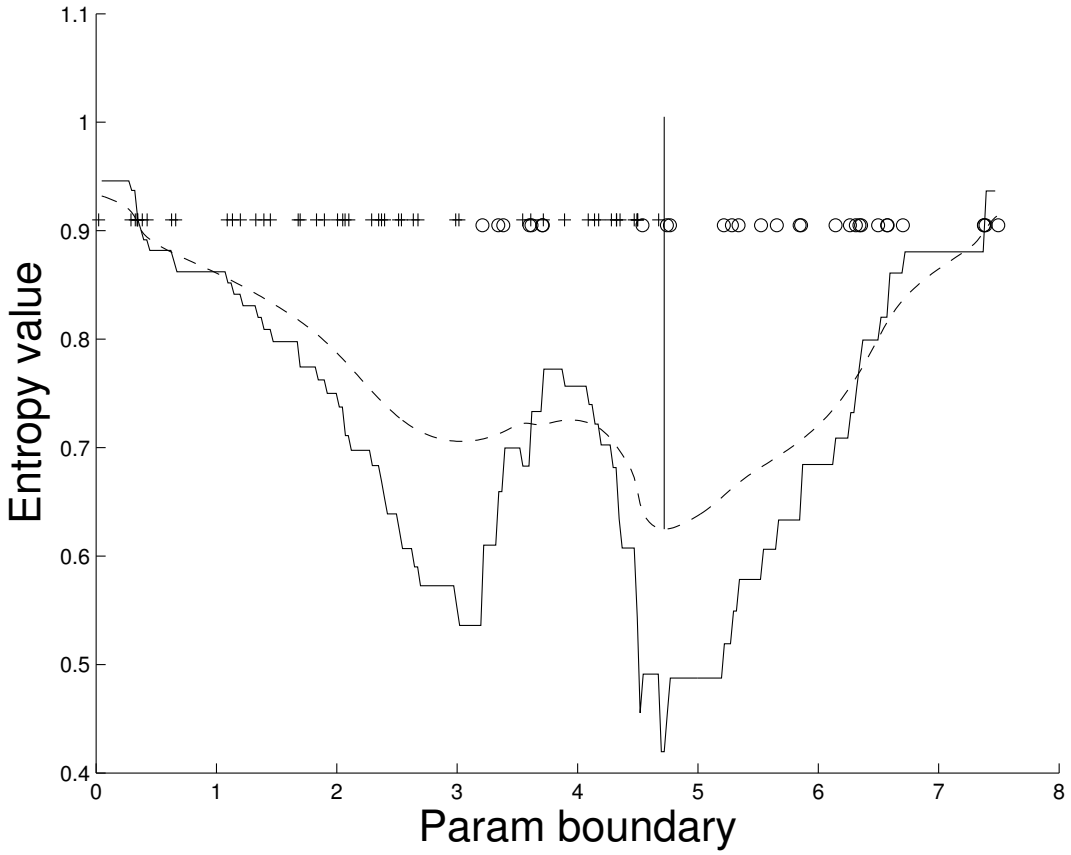


Figure 5.5: Points that belong to different classes are marked with crosses and circles. Solid curve shows conditional entropy curve. Dotted curve shows smoothed version of the same curve. Solid line that is perpendicular to parameter boundary axis indicates minimal value for the smoothed continuous version.

Note that the number of local minima may be controlled by the width of the smearing functions f_{χ_i}. Fig. 5.6 shows how the shape of $H_f(S|t_{j,k})$, changes with the width of f_{χ_i} for f_{χ_i} Gaussian.

Now, let us get back to the general multivariate case. The procedure described before can be easily extended to the general case in a very straightforward way. Indeed, consider splitting criterion (5.4), (5.6) and let

$$H_f(S|w) = \frac{D(S^+(w))}{D(S)} H_f(S^+(w)) + \frac{D(S^-(w))}{D(S)} H_f(S^-(w)),$$

$$RIG_f(S|w) = (H_f(S) - H_f(S|w))/H_f(S). \tag{5.9}$$

The following property of $RIG_f(S|w)$ is now immediate.

Proposition 5 *Suppose that for every value of w the set*

$$\mathcal{A}(w) = \{x \in S | \alpha(x, w) = 0\}$$

is an $n-1$ dimensional manifold, and it is such that that for any $\varepsilon > 0$ there is a $\delta > 0$: $\|w_1 - w_2\| < \delta$ implies that $\max_{x \in \mathcal{A}(w_1)} \operatorname{dist}(\mathcal{A}(w_2), x) < \varepsilon$. Furthermore, let f_{χ_i} be continuous. Then $RIG_f(S|\cdot)$ is differentiable.

Conclusion

In previous two sections we briefly studied the problem of basic scene recognition using simple statical tools available. We have shown that sometimes complicated task of scene recognition can be transfered to a low dimensional domain and successfully solved there. This approach may prove to be useful in scenarios when we are for example restricted in computational recourses available. Also, this kind of calculations does not require a lot of power consumption and implemented in silicon this type of solution may consume not more than several milliwatts. Even though we were able to reduce this problem to a low dimensional space, we will not be that lucky every time. For example, the problem of object detection and recognition can not be easily reduced to low dimensional one without sacrificing detection or recognition performance too much.

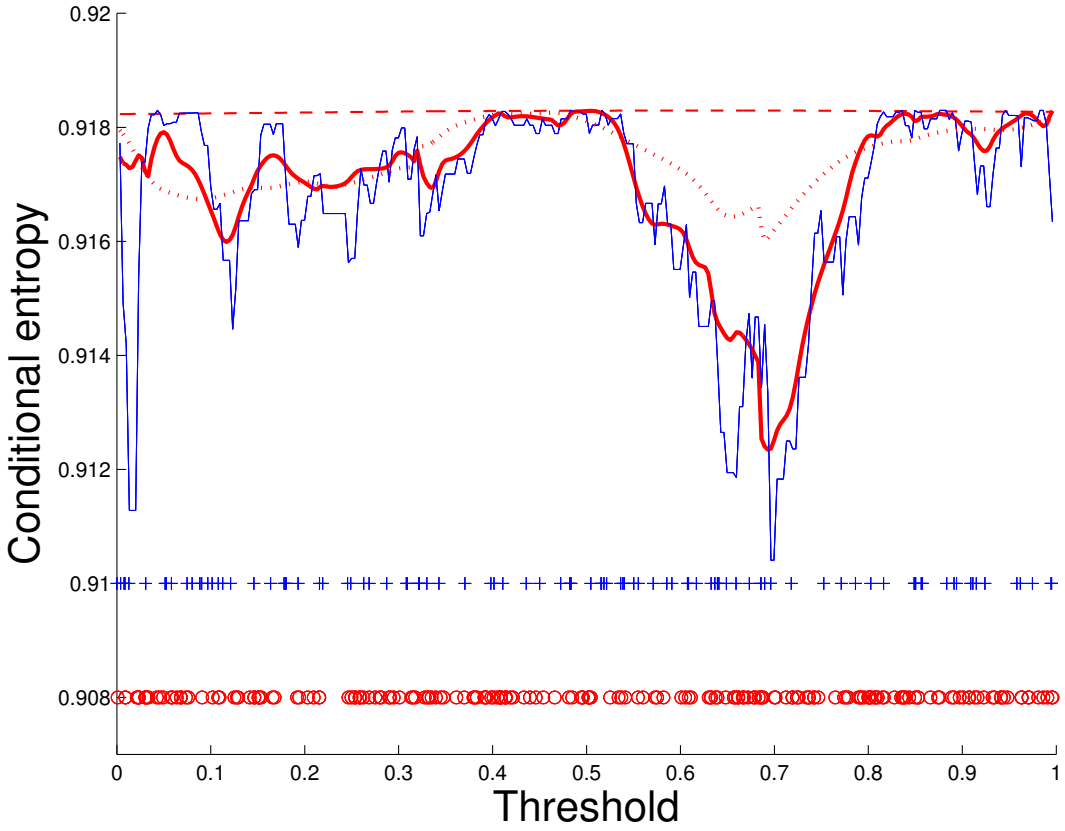


Figure 5.6: Dependence of $H_f(S|t_{j,k})$ on the width of f_{χ_i} (Gaussian, and σ is the corresponding width parameter). Solid blue curve depicts the original $H(S|t_{j,k})$. Thick red solid curve shows $H(S|t_{j,k})$ for $\sigma = 0.05$. Dotted line corresponds to the case of $\sigma = 0.01$, and dashed line stands for $\sigma = 0.0001$.

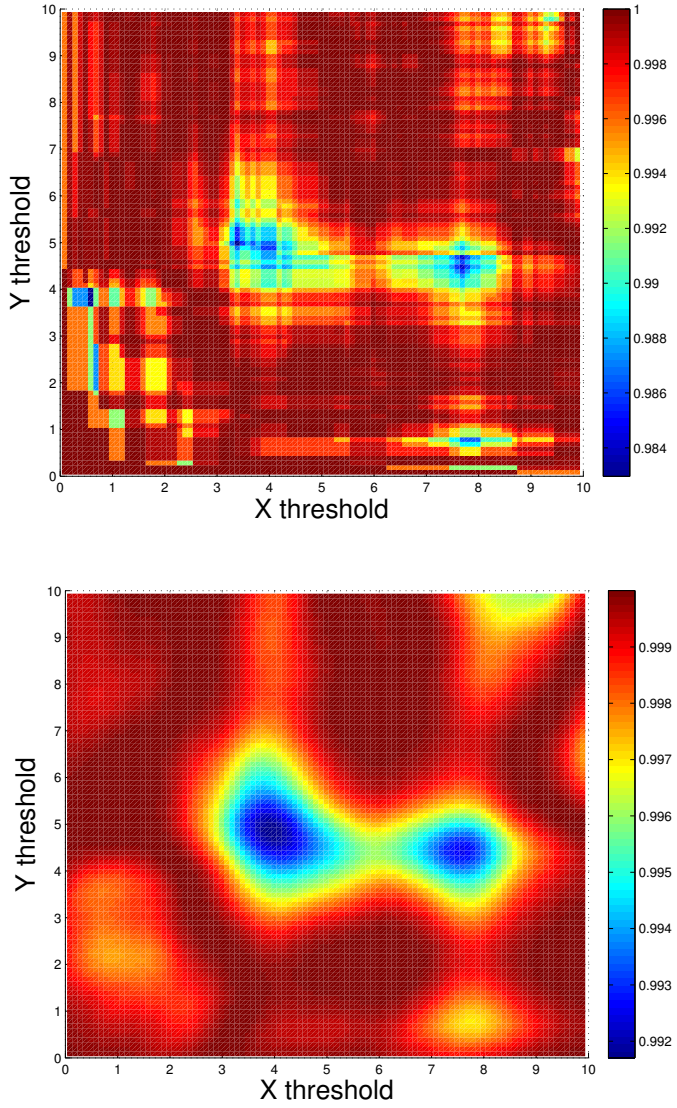


Figure 5.7: Contour plots of functions $H(S|\cdot)$ (upper panel) and $H_f(S|\cdot)$ (lower panel) for a randomly drawn sample of two-attribute dataset. The number of classes is 2. The number of local minima is drastically reduced in the lower panel.

In high dimensional space this gets significantly worse. For example, the attempt to use one parameter at a time as it was described in section — will quickly lead to overfitting if though one may observe a very good training set performance. Of course, binary decision trees with smooth entropy criterion will address this issues to some extent, but the optimization procedure may easily stuck on a local minima if σ_i is not chosen carefully. One can apply different method for hyper parameter optimization(see e.g. [46], [25]). Indeed, these methods can easily be applied to a relatively small dimensional tasks and technically there is nothing wrong in applying these methods in high dimensions. However, as volume(dimensionality and number of entries) grow, these methods may quickly become unpractical due to time constrains.

5.3 Knowledge transfer in Artificial Intelligence systems

Let us now show how ideas developed in section (4.5) can be used in practical computer vision applications for correcting errors of existing AI systems.

Let AI_s and AI_t be two systems developed, e.g. for the purposes of pedestrian detection in live video streams. Technological progress in embedded systems and availability of platforms such as e.g. Nvidia Jetson TX2 made hardware deployment of such AI systems at the edge of computer vision processing pipelines feasible. These AI systems, however, lack computational power to run state-of-the-art large scale object detection solutions such as e.g. ResNet [38] in real-time. Here we demonstrate that to compensate for this lack of power, AI Knowledge Transfer can be successfully employed. In particular, we suggest that the edge-based system is "taught" by the state-of-the-art teacher in a non-iterative and near-real time way. Since our building blocks are linear functionals, such learning will not lead to significant computational overheads. At the same time, as we will show later, the proposed AI Knowledge Transfer will result in a major boost to the system's performance in the conditions of the experiment.

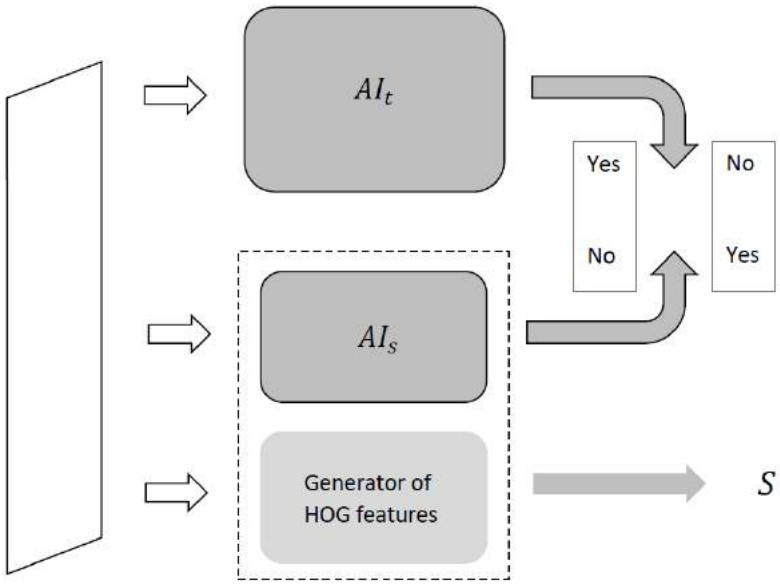


Figure 5.8: Knowledge transfer diagram between ResNet and HOG-SVM object detectors.

Definition of AI_s and AI_t and rationale

In our experiments, the teacher AI, AI_t, was modeled by a deep Convolutional Network, ResNet 18 [38] with circa 11M trainable parameters. The network was trained on a "teacher" dataset comprised of 5.2M non-pedestrian (negatives), and 600K pedestrian (positives) images. The student AI, AI_s, was modeled by a linear classifier with HOG features [19] and 2016 trainable parameters. The values of these parameters were the result of AI_s training on a "student" dataset, a sub-sample of the "teacher" dataset comprising of 55K positives and 130K negatives, respectively. This choice of AI_s and AI_t systems enabled us to emulate interaction between edge-based AIs and their more powerful counterparts that could be deployed on larger servers or computational clouds.

Moreover, to make the experiment more realistic, we assumed that internal states of both systems are inaccessible for direct observation. To generate sets $\mathcal{S}$ and $\mathcal{Y}$ required in Algorithms 1 and 2 we augmented system AI_s with an external generator of HOG features of the same dimension. We assumed, however, that covariance matrices of positives and negatives from the "student" dataset are available for the purposes of knowledge transfer. A diagram representing this setup is shown in Figure 5.8. A candidate image is evaluated by two systems simultaneously as well as by a HOG features generator. The latter generates 2016 dimensional vectors of

HOGs and stores these vectors in the set $\mathcal{S}$. If outputs of AI_s and AI_t do not match the corresponding feature vector is added to the set $\mathcal{Y}$.

Error types

In this experiment we consider and address two types of errors: false positives (Type I errors) and false negatives (Type II errors). The error types were determined as follows. An error is deemed as *false positive* if AI_s reported presence of a correctly sized full-figure image of pedestrian in a given image patch whereas no such object was there. Similarly, an error is deemed as *false negative* if a pedestrian was present in the given image patch but AI_s did not report it there.

In our setting, evaluation of an image patch by AI_t (ResNet) took 0.01 sec on Nvidia K80 which was several orders slower than that of AI_s (linear HOG-based classifier). Whilst such behavior was expected, this imposed technical limitations on the process of mitigating errors of Type II. Each frame from our testing video produced 400K image patches to test. Evaluation of all these candidates by our chosen AI_t is prohibitive computationally. To overcome this technical difficulty we tested only a limited subset of image proposals with regards to these error type. To get a computationally viable number of proposals for false negative testing, we increased sensitivity of the HOG-based classifier by lowering its detection threshold from 0 to -0.3. This way our linear classifier with lowered threshold acted as a filter letting through more true positives at the expense of large number of false positives. In this operational mode, Knowledge Transfer Unit were tasked to separate true positives from negatives in accordance with object labels supplied by AI_t.

Datasets

The approach was tested on two benchmark videos: LINTHESCHER sequence [30] created by ETHZ and comprised of 1208 frames and NOTTINGHAM video [14] containing 435 frames of live footage taken with an action camera. In what follows we will refer to these videos as ETHZ and NOTTINGHAM videos, respectively. ETHZ video contains complete images of 8435 pedestrians, whereas NOTTINGHAM video has 4039 full-figure images of pedestrians.

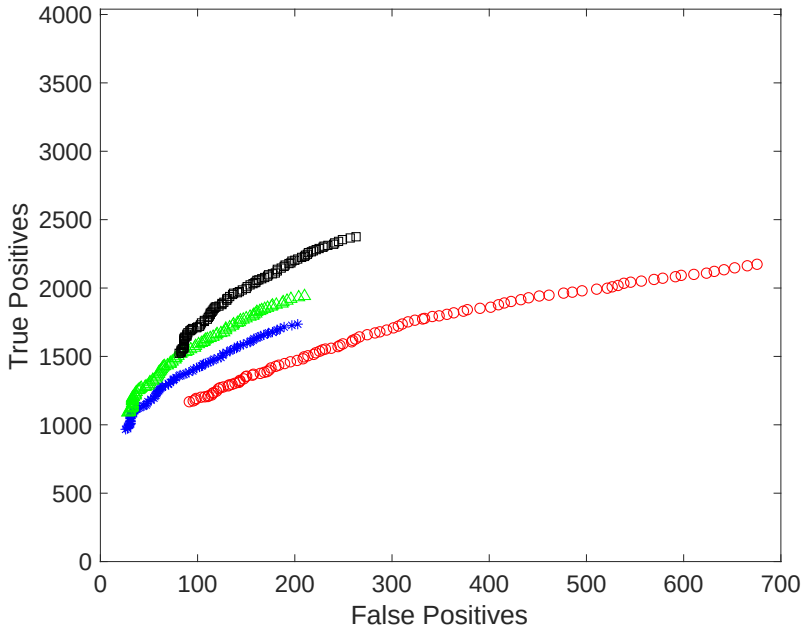


Figure 5.9: True positives as a function of false positives for NOTTINGHAM video.

Results

Performance and application of Algorithms 1, 2 for NOTTINGHAM and ETHZ videos are summarized in Fig. 5.9 and 5.10. Each curves in these figures is produced by varying the values of decision-making threshold in the HOG-based linear classifier. Red circles in Figure 5.9 show true positives as a function of false positives for the original linear classifier based on HOG features. Parameters of the classifier were set in accordance with Fisher linear discriminant formula. Blue stars correspond to AI_s after Algorithm 1 was applied to mitigate errors of Type I in the system. The value of p (number of clusters) in the algorithm was set to be equal to 5. Green triangles illustrate application of Algorithm 2 for the same error type. Here Algorithm 2 was slightly modified so that the resulting Knowledge Transfer Unit had only one functional ℓ_2. This was due to the low number of errors reaching stage two of the algorithm. Black squares correspond to AI_s after application of Algorithm 2 (error Type I) followed by application of Algorithm 2 to mitigate errors of Type II.

Figure 5.10 shows performance of the algorithms for ETHZ sequence. Red circles show performance of the original AI_s, green triangles correspond to AI_s supplemented with Knowledge Transfer Units derived using Algorithm 2 for errors of Type I. Black squares correspond to subsequent application of Algorithm 2 dealing with errors of Type II.

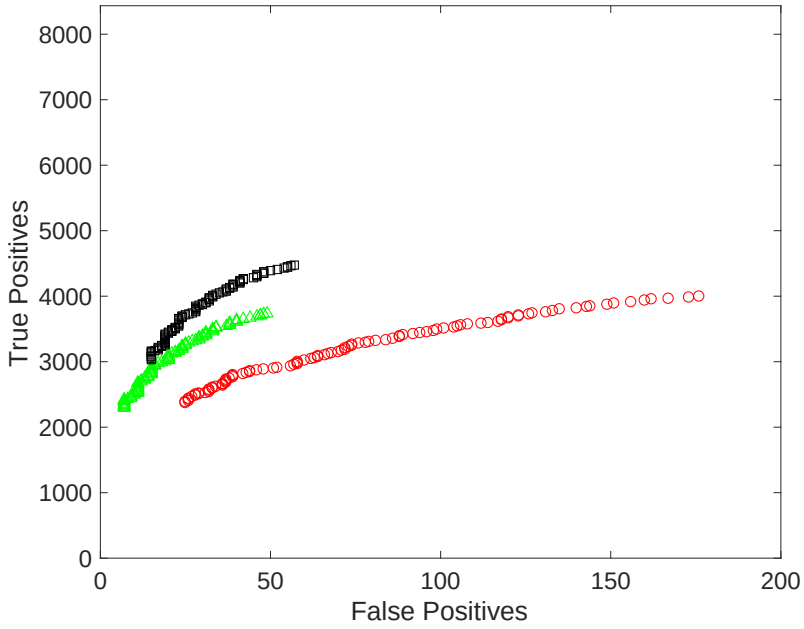


Figure 5.10: True positives as a function of false positives for ETHZ video.

In all these cases, supplementing AI_s with Knowledge Transfer Units constructed with the help of Algorithms 1, 2 for both error types resulted in significant boost to AI_s performance. Observe that in both cases application of Algorithm 2 to address errors of Type II has led to noticeable increases of numbers of false positives in the system at the beginning of the curves. Manual inspection of these false positives revealed that these errors are exclusively due mistakes of AI_t itself. For the sake of illustration, these errors for NOTTINGHAM video are shown in Fig. 5.11. These errors contain genuine false positives (images 12, 23-27) as well as mismatches by size (e.g. 1-7), and look-alikes (images 8,11,13,15-17).

Figure 5.11: False Positives induced by the teacher AI, AI_t.

Conclusion

As we have seen earlier in this thesis, high dimensionality bring uncertainty. This uncertainty appears because very often the underlying complexity of an object, that is being analyzed, is too high. In this case one has to employ more and more sophisticated machine learning techniques. More sophisticated techniques also bear additional costs: quite often the efficiency of the training process is reduced and more and more data are required to achieve good level of generalization. To achieve a good level of generalization in a task where relationships between objects parameters are complicated one has to increase the amount of parameters in order to increase descriptive power of a model. The latter increases training and evaluation time by a substantial margin.

Unfortunately, collecting more and more data and employing more sophisticated techniques does not guarantee that there will no errors in these systems.

We have shown, however, how does one take advantage of the high dimensionality of a problem. The key is to use concentration of measure effects: a phenomena that only can be found in high dimensional spaces. Theoretical findings in Chapter 4 allowed to establish the following fact. If the dimensionality of a problem is sufficiently high, one can remove individual errors of an AI system with almost no additional costs. In particular, thanks to the separations theorems in [31], these individual errors can be removed by linear functionals. The beauty of this approach is that, with very high probability, the overall performance of an AI system will not be hurt. Also, these linear functionals do not change the decision making process in AI system being corrected. If errors corrected with this approach are highly correlated, then the overall risk of errors in the corrected system is reduced.

The main consequence of these results is that it is clear now that *errors correction in non-linear systems is a linear problem.*

We also proposed a framework for instantaneous knowledge transfer between AI systems whose internal state used for decision-making can be described by elements of a high-dimensional vector space. The framework enables development of non-iterative algorithms for knowledge spreading between legacy AI systems with heterogeneous non-identical architectures and varying computing capabilities. Feasibility of the framework was illustrated with an example of knowledge transfer between two AI systems for automated pedestrian detection in video streams.

In the basis of the proposed knowledge transfer framework are separation theorems (Theorem 6 – 8) stating peculiar properties of large but finite random samples in high dimension. According to these results, $k < n$ random i.i.d. elements can be separated form $M \gg n$ randomly selected elements i.i.d. sampled from the same distribution by few linear functionals, with high probability. The theorems are proved for equidistributions in a ball and in a cube. The results can be trivially generalized to equidistributions in ellipsoids and Gaussian distributions. Generalizations to other meaningful distributions is the subject of future work.

Appendices

Appendix A

A very brief history of Machine Learning

This brief overview of the history of Machine learning tools was adopted from [85].

It is believed that the mathematical analysis of the learning process began when F. Rosenblatt suggested the first model of a learning machine, called the perceprtron. In 1957 perceptron algorithm was invented. The algorithm was designed to solve pattern recognition problems. In the simplest case this is a problem of separation of data points of two different categories. A separation rule is supposed to be built using given examples.

In Rosenblatt's model the perceptron has n inputs $x = (x^1, \ldots, x^n) \in X \subset R^n$ and one of two possible outputs $y = \{-1, 1\}$. Outputs and inputs are related through the following dependency:

$$y = \text{sign}\{(\omega \cdot x) - b\}.$$

From a geometrical perspective this functional divides space into two regions. The space it divided by a linear function and parameters ω and b fully define it. The goal of the learning process is to find suitable coefficients for a perceptron(or neuron).

In a more complicated case the output of one neuron can be input for a neuron in the next layer. Even though each neuron has only one output, this output can be sent to multiple neurons. While in the simplest case one neuron separates the space into two regions, a multilayer perceptron (Figure A.1) is capable of forming several

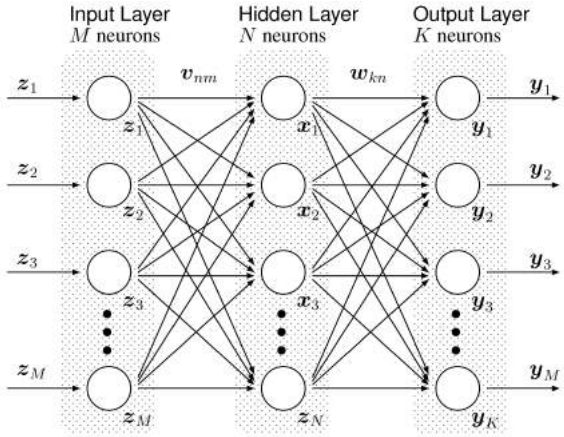


Figure A.1: Multilayer perceptron structure.

regions bounded by piecewise linear functions. Learning this kind of model means finding appropriate coefficients for all neurons in this structure.

Back then it was unclear how to train such models. Specifically, it was unclear how to find appropriate coefficients for this net of neurons. Rosenblatt's idea was to initialize coefficients of first layers randomly and fix them. This leaves only one set of trainable parameters: parameters of the output neuron. In this case initial neuron's input space X is translated into a new a one Z.

The following algorithm was proposed. Let

$$(x_1, y_1), \ldots, (x_l, y_l)$$

be the training data presented in the initial input space X. Then

$$(z_1, y_1), \ldots, (z_l, y_l)$$

will be training data in a transformed space Z. Elements of the training data are fed into the network one by one. Let ω be the coefficient vector of the last neuron in this set. Then the algorithm has the following steps

1. If a given example (x_k, y_k) if classified correctly, the coefficient vector of the last neuron is not changed.

2. If a given examples is not classified correctly, the coefficients vector is changed

according to the following rule:

$$w^{t+1} = t^t + y_k z_k$$

3. The initial state of the vector is zero, i.e.

$$w^1 = 0$$

Using this update rule perceptron has demonstrated the ability to generalize on simple examples.

In 1962 Novikoff proves the first perceptron theorem [61]. Given the following conditions

1. the norm of training vectors is bounded by some constant R: $|z| \leq R$;
2. the training data can be separated with margin ρ:

$$\sup_{\omega} \min_{i} y_i(z_i \cdot \omega) > \rho;$$

3. the training sequence must be presented to perceptron a sufficient number of times,

theorem states that after at most

$$N \leq \frac{R^2}{\rho^2}$$

steps a separating hyperplane will be constructed. This theorem is very important because it connects generalization capabilities of the perceptron with the number of errors on the training set.

Using the same technique it is possible to prove that if the data are separable, then after a finite number of steps the perceptron is able to separate an infinite sequence of data. Moreover, if one supplies the perceptron with the following stopping rule: learning process is stopped if

$$m_k = \frac{1 + 2\ln k - \ln\theta}{-\ln(1-\varepsilon)}$$

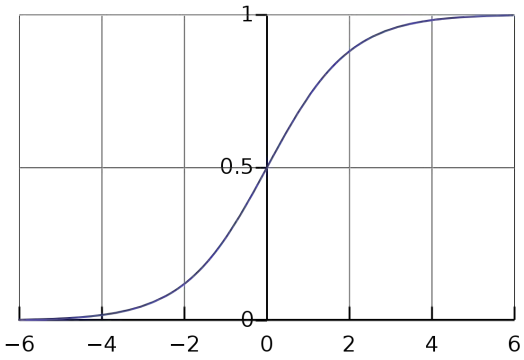


Figure A.2: Sigmoid function.

elements of the training data do not change the decision rule, then the perceptron will stop the learning process during the first

$$l \leq \frac{1 + 4 \ln \frac{R}{\rho} + \ln \theta}{-\ln(1-\varepsilon)} \left[\frac{R^2}{\rho^2}\right]$$

steps.

In this case the decision rule will have error rate less than ε with the probability $1 - \theta$.

The next step in constructing a general type of learning machine was done in 1986 when so-called back-propagation technique for finding the weights simultaneously for all neurons in the net was used.

In 1986 several authors proposed a method for simultaneous construction of the vector of coefficients for all neurons in perceptron called back-propagation algorithm. The idea of the method is simple: substitute the discontinuous sign $\{(w \cdot x) - b\}$ with a continuous sigmoid approximation:

$$t = S\{(w \cdot x) - b\}$$

where $S(-\infty) = -1$ and $S(\infty) = 1$ (see Figure A.2 for illustration). This substitution makes composition of neurons a smooth function and therefore for any fixed argument it has a gradient with respect to all of parameters of the neurons. Once gradients are calculated, one can apply gradient-based techniques for constructing a function that approximates the desired function. It is obvious, that a gradient-based technique can only find local minima.

Appendix B

Ill-posed problems

Under some circumstances the problem of solving operator equations

$$Af = F, f \in F$$

is ill-posed. Even if there exists a unique solution to this problem, a small deviation on the right-hand side of this equation (F_σ instead of F, where $\|F_\sigma - F\|$ can be arbitrary small) can cause large deviation in the solution. In this case functions f_σ that minimize the functional

$$R(f_\sigma) = \|Af - F_\sigma\|^2$$

do not guarantee the desired approximation level for the final solution, even if σ is close to zero. It was initially thought that real-life problems can not be ill-posed. However later it appeared that a lot of them are. It was discovered that if instead of the functional $R(f)$ one minimizes another so-called regularized functional

$$R^*(f) = \|Af - F_\sigma\|^2 + \gamma(\sigma)\Omega(f),$$

where $\Omega(f)$ is some functional and $\gamma(\sigma)$ is an appropriately chosen constant (this constant is adjusted according to the noise level σ), then it is possible to obtain a solution that converges to a desired one as σ tends to zero. For our discussion it is important that a lot of computer vision tasks may be ill-posed.

Bibliography

[1] A. Gupta A. Saxena and A. Mukerjee. Non-linear dimensionality reduction by locally linear isomaps. *Lecture Notes in Computer Science*, 2004.

[2] Dimitris Achlioptas. Database-friendly random projections. pages 274–281. ACM Press, 2001.

[3] Nir Ailon and Bernard Chazelle. The fast johnson-lindenstrauss transform and approximate nearest neighbors. *SIAM J. COMPUT*, pages 302–322, 2009.

[4] S. Artstein. Proportional concentration phenomena on the sphere. *Israel Journal of Mathematics*, 132:337–358, 2002.

[5] K. Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, 19.

[6] R.R. Coifman B. Nadler, S. Lafon and I.G. Kevrekidis. Diffusion maps, spectral clustering and the reaction coordinates of dynamical systems. *Applied and Computational Harmonic Analysis: Special Issue on Diffusion Maps and Wavelets*, 2006.

[7] A.J. Smola B. Scholkopf and K.-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 1998.

[8] M. Balasubramanian and E.L. Schwartz. The isomap algorithm and topological stability. *Science*, 2002.

[9] Keith Ball. An elementary introduction to modern convex geometry. In *in Flavors of Geometry*, pages 1–58. Univ. Press, 1997.

[10] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques

for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, 2001.

[11] Richard Ernest Bellman. *Dynamic programming*. Princeton University Press, 1957.

[12] Richard Ernest Bellman. *Adaptive control processes: a guided tour*. Princeton University Press, 1961.

[13] W.G.C. Boyd. Gamma function asymptotics by an extension of the method of steepest descents. *Proc. R. Soc. Lond. A*, 27:609–630, 1994.

[14] R. Burton. NOTTINGHAM video. `https://youtu.be/SJbhOJQCSuQ`, 2016. A test video for pedestrians detection taken from the streets of Nottingham by an action camera [Online; accessed 01-Feb-2018].

[15] Rich Caruana, Steve Lawrence, and Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *IN PROC. NEURAL INFORMATION PROCESSING SYSTEMS CONFERENCE*, pages 402–408, 2000.

[16] Chih-Jen Lin Chih-Chung Chang. LIBSVM: A Library for Support Vector Machines. `http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf`, 2001. [Online; accessed 01-Feb-2018].

[17] Seagate Company. Video Surveillance Storage: How Much Is Enough? `https://www.seagate.com/files/staticfiles/docs/pdf/whitepaper/video-surv-storage-tp571-3-1202-us.pdf`, 2012. [Online; accessed 01-Feb-2018].

[18] T. Cox and M. Cox. *Multidimensional scaling*. Chapman & Hall, 1994.

[19] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[20] M. Davenport. *The fundamentals of compressive sensing*. SigView, 2013.

[21] Alessandro Panconesi Devdatt P. Dubhashi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2012.

[22] D.L. Donoho and C. Grimes. Hessian eigenmaps: New locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 2005.

[23] Walter Dornberger. *V-2*. Ballantine Books, 1954.

[24] Walter Dornberger. An elementary introduction to modern convex geometry. *Flavors of Geometry*, 31, 1997.

[25] Ryan P. Adams Dougal Maclaurin, David Duvenaud. Gradient-based Hyperparameter Optimization through Reversible Learning. `https://www.robots.ox.ac.uk/~vgg/rg/papers/MaclaurinICML15.pdf`, 2015. [Online; accessed 01-Feb-2018].

[26] R. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley, 2000.

[27] R. Dupre, V. Argyriou, and D. Greenhil. Prediction of physics simulation using dimensionality reduction and regression. pages 99–110. 2013.

[28] The Economist. Data, data everywhere. `http://www.economist.com/node/15557443`, 2010. [Online; accessed 01-Feb-2018].

[29] Mannila Ella, Bingham; Heikki. Random projection in dimensionality reduction: Applications to image and text data. *KDD-2001: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: Association for Computing Machinery*, pages 245–250, 2015.

[30] A. Ess, B. Leibe, K. Schindler, and L. van Gool. A mobile vision system for robust multi-person tracking. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. DOI: 10.1109/CVPR.2008.4587581.

[31] A.N. Gorban, R. Burton, I. Romanenko, and Tyukin I. One-Trial Correction

of Legacy AI Systems and Stochastic Separation Theorems. https://arxiv.org/abs/1610.00494, 2016. [Online; accessed 01-Feb-2018].

[32] A.N. Gorban and I.Y. Tyukin. Stochastic Separation Theorems. https://arxiv.org/pdf/1511.07571v1.pdf, 2017. [Online; accessed 01-Feb-2018].

[33] N. Gozlan and C. Leonard. Transport inequalities: a survey. *Markov Processes and Related Fields*, 16, 2010.

[34] Thomas W Gross, Jonathan; Tucker. *Topological Graph Theory*. Dover Publications, 2001.

[35] The Guardian. Tesla driver dies in first fatal crash while using autopilot mode. https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk, 2016. [Online; accessed 01-Feb-2018].

[36] Isabelle Guyon. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[37] H. Han. Analyzing support vector machine overfitting on microarray data. In D.S. Huang, K. Han, and M. Gromiha, editors, *Intelligent Computing in Bioinformatics. ICIC 2014. Lecture Notes in Computer Science*, volume 8590, pages 148–156. Springer, Cham, 2014.

[38] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[39] R. Hecht-Nielsen. Context vectors: general purpose approximate meaning representations self-organized from raw data. *Computational Intelligence: Imitating Life*, pages 43–56, 1994.

[40] Martin Hilbert. BIG DATA FOR DEVELOPMENT:A Review of Promises and Challenges. http://www.martinhilbert.net/big-data-for-development/, 2015. [Online; accessed 01-Feb-2018].

[41] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 1963.

[42] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 1933.

[43] G.F Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14:55–63, 1968.

[44] W. Scott Spanglerb Inderjit S. Dhillona, Dharmendra S. Modhab. Class visualization of high-dimensional data with applications. *Computational Statistics Data Analysis*, pages 59–90, 2002.

[45] Anil Jain. The 5 Vs of Big Data. `https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data/`, 2016. [Online; accessed 01-Feb-2018].

[46] Yoshua Bengio James Bergstra. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[47] V. de Silva J.B. Tenenbaum and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2003.

[48] Niall Jenkins. 245 million video surveillance cameras installed globally in 2014. `https://technology.ihs.com/532501/245-million-video-surveillance-cameras-installed-globally-in-2014`, 2014. [Online; accessed 01-Feb-2018].

[49] Joram Johnson, William B.; Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Conference in Modern Analysis and Probability*, pages 189–206, 1982.

[50] Li Fei-Fe Justin Johnson, Andrej Karpathy. DenseCap: Fully Convolutional Localization Networks for Dense Captioning. `https://arxiv.org/pdf/1511.07571v1.pdf`, 2015. [Online; accessed 01-Feb-2018].

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classifica-

tion with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[52] S. Lafon and A.B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006.

[53] M. Ledoux. The concentration of measure phenomenon. *Mathematical Surveys and Monographs. American Mathematical Society*, 89, 2001.

[54] J.A. Lee and M. Verleysen. Nonlinear dimensionality reduction of data manifolds with essential loops. *Neurocomputing*, 2005.

[55] P. Le'vy. *Probl'emes concrets d'analyse fonctionnelle, 2nd ed.* Paris: Gauthier-Villars, 1951.

[56] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[57] G. Lugosi. Concentration of measure inequalities - lecture notes. 2009.

[58] C. McDiarmid. Concentration. *Probabilistic Methods for Algorithmic Discrete Mathematics*, 1996.

[59] Michal Valko-James Lyons-Weiler Milos Hauskrecht, Richard Pelikan. Feature selection and dimensionality reduction in genomics and proteomics. pages 149–172. 2011.

[60] Alex Krizhevsky-Ilya Sutskever-Ruslan Salakhutdinov Nitish Srivastava, Geoffrey Hinton. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. volume 15. 2014.

[61] Albert B.J. Novikof. On convergence proofs on perceptrons. page 615–622. 1962.

[62] H. Su-J. Krause-S. Satheesh-S. Ma Z. Huang A. Karpathy A. Khosla M. Bernstein A. C. Berg O. Russakovsky, J. Deng and L. FeiFei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis*, pages 1–42, 2014.

[63] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philiosophical Magazine*, 1901.

[64] Jeff Donahue-Trevor Darrell1 Philipp Krahenb, Carl Doersch. Data-dependent initializations of convolutional neural networks. 2016.

[65] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press, Cambridge, 1988.

[66] Ross Girshick-Pieter Noordhuis-Lukasz Wesolowski-Aapo Kyrola Andrew Tulloch Yangqing Jia Kaiming He Priya Goyal, Piotr Dollar. Accurate, large minibatch sgd: Training imagenet in 1 hour.

[67] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1,1:81–106, 1986.

[68] J.R. QUINLAN. Induction of decision trees. *Machine Learnin*, 1985.

[69] Quinlan J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc, 1993.

[70] J. Šalkevičius R. Damaševičius, M. Vasiljevas and M. Woźniak. Human activity recognition in aal environments using random projections. 2016.

[71] C.-J. Hsieh X.-R. Wang R.-E. Fan, K.-W. Chang and C.-J. Lin. "liblinear: A library for large linear classification. *Journal of Machine Learning Research*, page 1871–1874, 2008.

[72] Mike Schuster Noam Shazeer Yonghui Wu Rafal Jozefowicz, Oriol Vinyals. Exploring the Limits of Language Modeling. `http://arxiv.org/abs/1602.02410`, 2016. [Online; accessed 01-Feb-2018].

[73] Naftali Tishby Ran Gilad-Bachrach, Amir Navot. An information theoretic tradeoff between complexity and accuracy.

[74] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 2000.

[75] B. Wang P.-T. Bremer S. Liu, D. Maljovec and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. 2015.

[76] J.W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 1969.

[77] C. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[78] J. Shawe-Taylor and N. Christianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[79] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *An International Conference on Learning Representations*, 2015.

[80] Pascal Massart Stephane Boucheron, Gabor Lugosi. *Concentration inequalities: a nonasymptotic theory of independence*. Oxford University Press, 2016.

[81] Michel Talagrand. A new look at independence. *The Annals of Probability*, 24, 1996.

[82] J.B. Tenenbaum. Mapping a manifold of perceptual observations. *Advances in Neural Information Processing Systems*, 1998.

[83] Laura Tolosi and Thomas Lengauer. Classification with correlated features: unreliability of feature ranking and solutions. *Bioinformatics*, 27:1986–1994, 2011.

[84] L.J.P. van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality reduction: A comparative review, 2008.

[85] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 2000.

[86] Hugo; Lajoie Isabelle; Bengio-Yoshua; Manzagol Pierre-Antoine Vincent, Pascal; Larochelle. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 2011.

[87] Guy Lebanon Yi Mao, Krishnakumar Balasubramanian. Dimensionality reduction for text using domain knowledge. pages 801–809. 2010.

[88] Z. Zhang and H. Zha. Principal manifolds and nonlinear dimensionality reduction via local tangent space alignment. *SIAM Journal of Scientific Computing*, 2004.